A Relaxed Approach to RDF Querying

Carlos A. Hurtado
churtado@dcc.uchile.cl
Department of Computer Science
Universidad de Chile

Alexandra Poulovassilis, Peter T. Wood
{ap,ptw}@dcs.bbk.ac.uk
School of Computer Science and Information Systems
Birkbeck, University of London

Motivation

- W3C RDF data access group has emphasized the need to enhance RDF query languages to solve real problems:
  - “it must be possible to express a query that does not fail when some specified part of the query fails to match”
- motivated the `OPTIONAL` clause in the emerging SPARQL W3C proposal
- `OPTIONAL` clause allows a query to return matchings that fail to match some conditions in the query

Motivation

Consider the following SPARQL-like query:

$$
\begin{aligned}
?Y \quad \leftarrow \quad & (?X, \mathit{hasName}, ?Y), (?X, \texttt{type}, \mathit{Wine}), \\
& (?X, \mathit{locatedIn}, ?Z), \\
& \texttt{OPTIONAL}(?Z, \texttt{type}, \mathit{MalboroughRegion}).
\end{aligned}
$$

- returns names of wines located in Malborough region
- head of query is a single variable
- body is a graph pattern comprising 4 triple patterns
- because last triple pattern is inside an `OPTIONAL` clause, query also returns names of all wines (located somewhere)

Motivation

- the conditions of a query could be relaxed in ways other than simply dropping optional triple patterns
  - by replacing constants with variables
  - by using the type and predicate hierarchies in an ontology associated with the data
- `OPTIONAL` clause lacks a notion of ranking answers; hence users cannot establish how closely answers match original query

Example of relaxation

Consider the following query, with a new RELAX clause:

$$?Y \quad \leftarrow \quad (?X, hasName, ?Y), \\ \texttt{RELAX}(?X, \texttt{type}, SauvignonBlanc).$$

- returns names of wines of type Sauvignon Blanc
- assume *SauvignonBlanc* is a subclass of *WhiteWine*
- 2nd triple pattern can be relaxed to $(?X, \texttt{type}, WhiteWine)$, for example
- names of Sauvignon Blanc wines can be returned to the user before names of white wines

Outline

- Related work
- Definitions
  - RDF graphs, RDFS ontologies, entailment, graph patterns, conjunctive queries
- Relaxing triple patterns
- Relaxation graph of a triple pattern
- Algorithm for computing ranked, relaxed answers
- Conclusion and future work

Related Work

- the idea of making queries more flexible by the logical relaxation of their conditions is not new
- e.g., Gaasterland, Godfrey and Minker proposed such a mechanism in the context of deductive databases and logic programming, and called it *query relaxation*
- there are many other proposals for flexible querying
- we believe this is the first proposal for flexible querying of RDF which also includes ranking

Definitions—RDF graphs

- we work with RDF graphs which may mention the RDFS vocabulary
- assume there are infinite sets I (IRIs), B (blank nodes), and L (RDF literals)
- elements in $I \cup B \cup L$ are called RDF *terms*
- a triple $(v_1, v_2, v_3) \in (I \cup B) \times I \times (I \cup B \cup L)$ is called an RDF triple
- v_1 is called the subject, v_2 the predicate and v_3 the object
- an RDF graph is a set of RDF triples

Definitions—RDFS ontologies

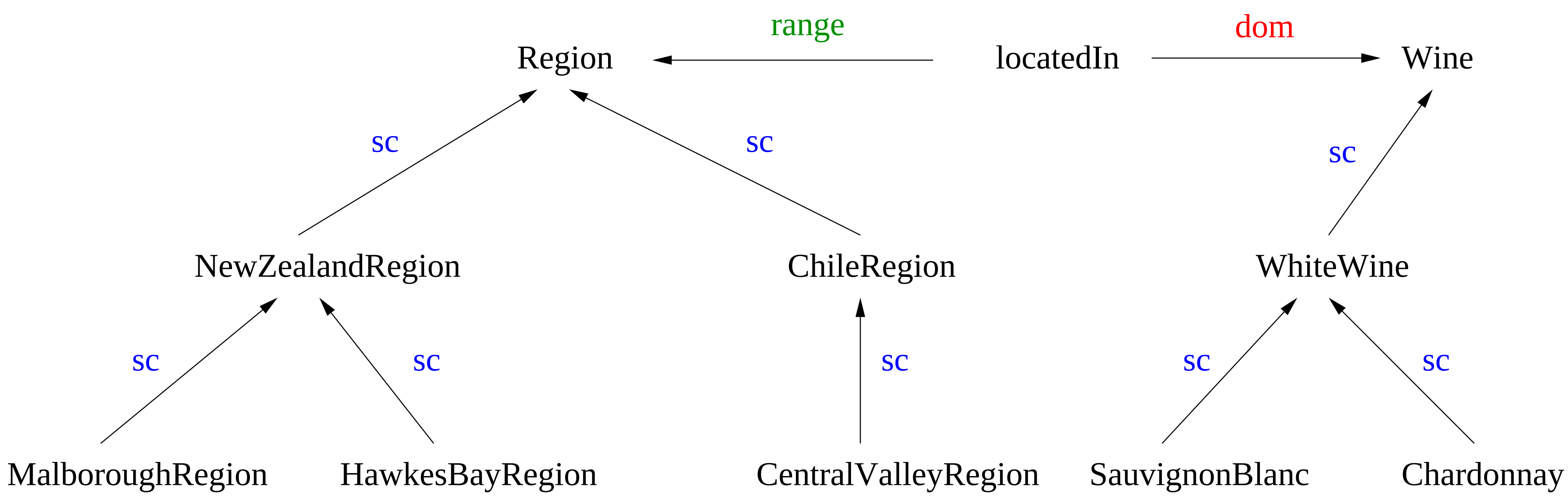
- we assume an ontology is modeled as an RDF graph with interpreted RDFS vocabulary
- RDFS vocabulary defines classes and properties, used to describe related resources and their relationships
- we use a small fragment of the RDFS vocabulary:

  `rdfs:range` [range], `rdfs:domain` [dom],

  `rdf:type` [type], `rdfs:subClassOf` [sc],

  `rdfs:subPropertyOf` [sp]
- we assume that `sc` and `sp` are acyclic
- we also assume there are no blank nodes in the ontology
- we omit all other vocabulary including `rdf:Property`, `rdfs:Class`, and `rdfs:Resource`

Fragment of RDFS wine ontology



Definitions—simple entailment

- decompose entailment into simple and RDFS entailment
- simple entailment depends only on the basic logical form of RDF graphs and therefore holds for any vocabulary
- given two RDF graphs G_1, G_2, a map from G_1 to G_2 is a function μ from terms of G_1 to terms of G_2, preserving IRIs and literals, such that for each triple $(a,b,c) \in G_1$ we have $(\mu(a), \mu(b), \mu(c)) \in G_2$
- RDF graph G_1 simply entails G_2, denoted $G_1 \models_{\texttt{simple}} G_2$, if and only if there exists a map from G_2 to G_1
- simple entailment is captured by rule 7 on next slide

RDFS Inference Rules

Group A (Subproperty) $(1)\,\frac{(a,\texttt{sp},b)\;\;(b,\texttt{sp},c)}{(a,\texttt{sp},c)}$ $(2)\,\frac{(a,\texttt{sp},b)\;\;(x,a,y)}{(x,b,y)}$

Group B (Subclass) $(3)\,\frac{(a,\texttt{sc},b)\;\;(b,\texttt{sc},c)}{(a,\texttt{sc},c)}$ $(4)\,\frac{(a,\texttt{sc},b)\;\;(x,\texttt{type},a)}{(x,\texttt{type},b)}$

Group C (Typing) $(5)\,\frac{(a,\texttt{dom},c)\;\;(x,a,y)}{(x,\texttt{type},c)}$ $(6)\,\frac{(a,\texttt{range},d)\;\;(x,a,y)}{(y,\texttt{type},d)}$

(Simple Entailment) (7) For a map $\mu : G' \longrightarrow G : \frac{G}{G'}$

Definitions—RDFS entailment

- RDFS entailment captures the semantics added by the RDFS vocabulary
- we write that $G_1 \models_{\texttt{rule}} G_2$ if G_2 can be derived from G_1 by iteratively applying rules in groups (A), (B) and (C) on the previous slide
- closure of an RDF graph G, denoted $\mathrm{cl}(G)$, is the closure of G under the rules in groups (A), (B) and (C)
- we have that $G_1 \models_{\texttt{rule}} G_2$ if and only if $G_2 \in \mathrm{cl}(G_1)$
- it turns out that G_1 RDFS-entails G_2, written $G_1 \models_{\texttt{RDFS}} G_2$, iff there is a graph G such that $G_1 \models_{\texttt{rule}} G$ and $G \models_{\texttt{simple}} G_2$

Definitions—graph patterns

- assume set of variables V disjoint from the sets I, B, and L
- a triple pattern is a triple $(v_1, v_2, v_3) \in (I \cup V) \times (I \cup V) \times (I \cup V \cup L)$
- a graph pattern is a set of triple patterns
- we denote by $\mathtt{var}(P)$ the variables mentioned in P
- variables are indicated by a leading question mark
- the notions of map and entailment can be generalized to graph patterns by treating variables like blanks

Definitions—conjunctive queries

- a conjunctive query Q is an expression $T \leftarrow B$
  - B is a graph pattern
  - $T = \langle T_1, \ldots, T_n \rangle$ is a list of variables in $\mathtt{var}(B)$
- we denote T by $\mathtt{Head}(Q)$, and B by $\mathtt{Body}(Q)$
- a query Q may be formulated over an ontology O
- matching is a function from $\mathtt{var}(\mathtt{Body}(Q))$ to $(I \cup B \cup L)$
- for matching Θ, $\Theta(\mathtt{Body}(Q))$ denotes the graph resulting from replacing each variable X in $\mathtt{Body}(Q)$ by $\Theta(X)$
- given RDF graph G, the answer of Q, denoted $\mathtt{ans}(Q, O, G)$, is the set of tuples defined as follows:
  - for each Θ such that $\Theta(\mathtt{Body}(Q)) \subseteq \mathrm{cl}(O \cup G)$, return $\Theta(\mathtt{Head}(Q))$

Relaxing triple patterns

- relaxation will be defined in the context of an ontology, denoted by O, and a set of fixed variables, denoted by F
- we model relaxation as a combination of two types of relaxations, ontology relaxation and simple relaxation
- let t_1, t_2 be triple patterns, where $t_1 \notin \mathrm{cl}(O)$, $t_2 \notin \mathrm{cl}(O)$, and $\mathtt{var}(t_1) = \mathtt{var}(t_2) \subseteq F$
- ontology relaxation is defined as follows:
  $t_1 \prec^*_{\mathtt{onto}} t_2$ if $\{t_1\} \cup O \models_{\mathtt{rule}} t_2$
- e.g., let O be wine ontology and let $F = \{?X\}$
  - $(?X, \mathtt{type}, \mathit{SauvignonBlanc}) \prec^*_{\mathtt{onto}} (?X, \mathtt{type}, \mathit{Wine})$
  - $(?X, \mathit{locatedIn}, \mathit{Maipo}) \prec^*_{\mathtt{onto}} (?X, \mathtt{type}, \mathit{Wine})$
  - $(?X, \mathit{locatedIn}, ?Y) \not\prec^*_{\mathtt{onto}} (?X, \mathtt{type}, \mathit{Wine})$

Relaxing triple patterns

- simple relaxation is defined as follows: $t_1 \prec^*_{\mathtt{simple}} t_2$ if $t_1 \models_{\mathtt{simple}} t_2$ via a map that preserves F
- e.g., with $F = \{?X\}$
  - $(?X, \mathtt{type}, \mathit{Wine}) \prec^*_{\mathtt{simple}} (?X, \mathtt{type}, ?Z)$
  - $(?X, \mathtt{type}, \mathit{Wine}) \prec^*_{\mathtt{simple}} (?X, ?W, \mathit{Wine})$
- relaxation is defined as follows. We say that t_2 relaxes t_1, denoted $t_1 \prec^* t_2$, if one of the following holds:
  1. $t_1 \prec^*_{\mathtt{onto}} t_2$,
  2. $t_1 \prec^*_{\mathtt{simple}} t_2$, or
  3. there exists a t such that $t_1 \prec^* t$ and $t \prec^* t_2$.
- denote by $\prec$ (direct relaxation) the reflexive and transitive reduction of $\prec^*$ (relaxation)

Relaxation graph of a triple pattern

- want to relax each triple pattern that occurs inside the `RELAX` clause of a query
- adapt the relaxation relationship to use relaxation “above” a given triple pattern
- relaxation relation “above” a triple pattern t, denoted by $\prec_t^*$, is $\prec^*$ restricted to triple patterns t' such that $t \prec^* t'$, and where $F = \texttt{var}(t)$ (i.e., the variables of t are the fixed variables in the relaxation)
- the relaxation graph of a triple pattern t is the directed acyclic graph induced by $\prec_t$

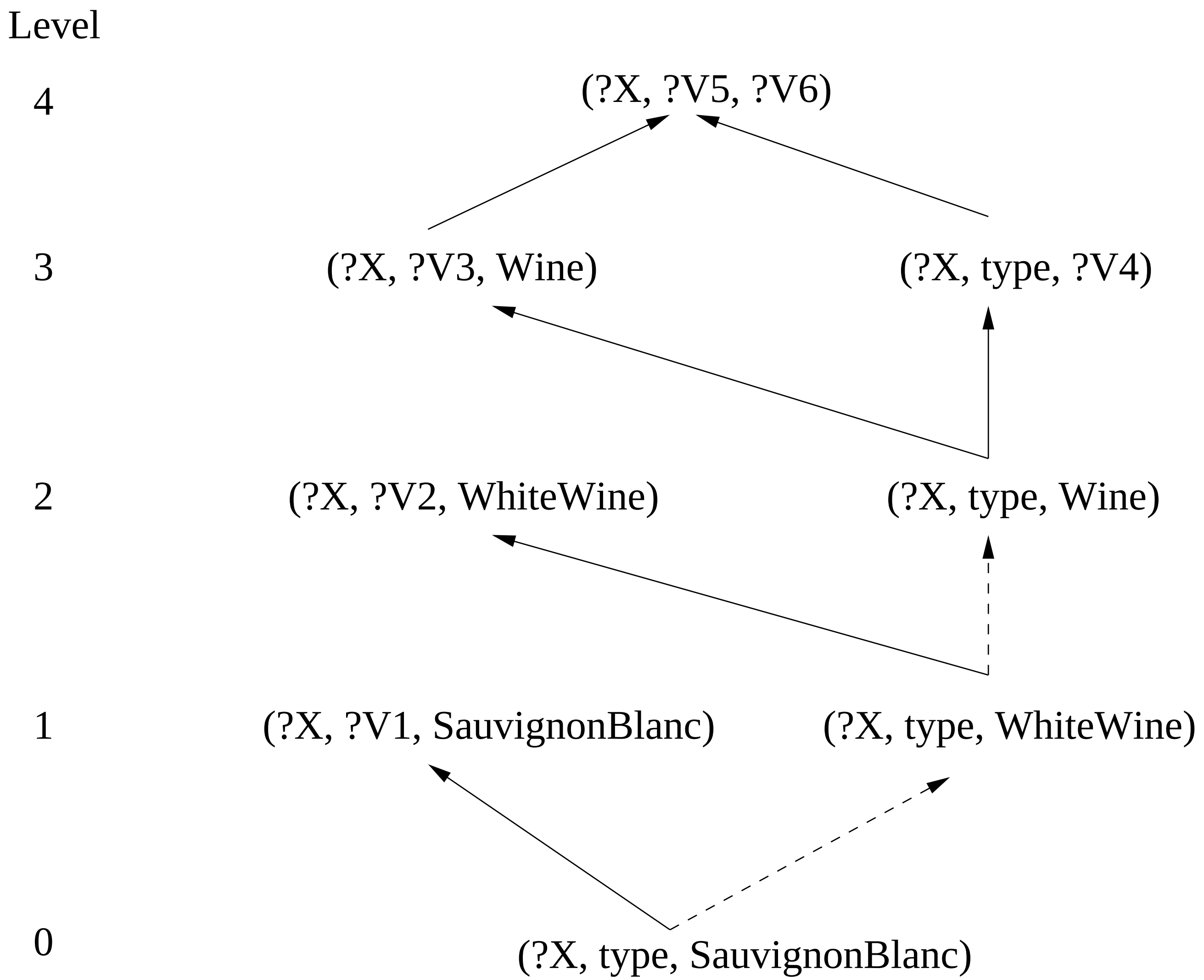
Example

Consider the following query:

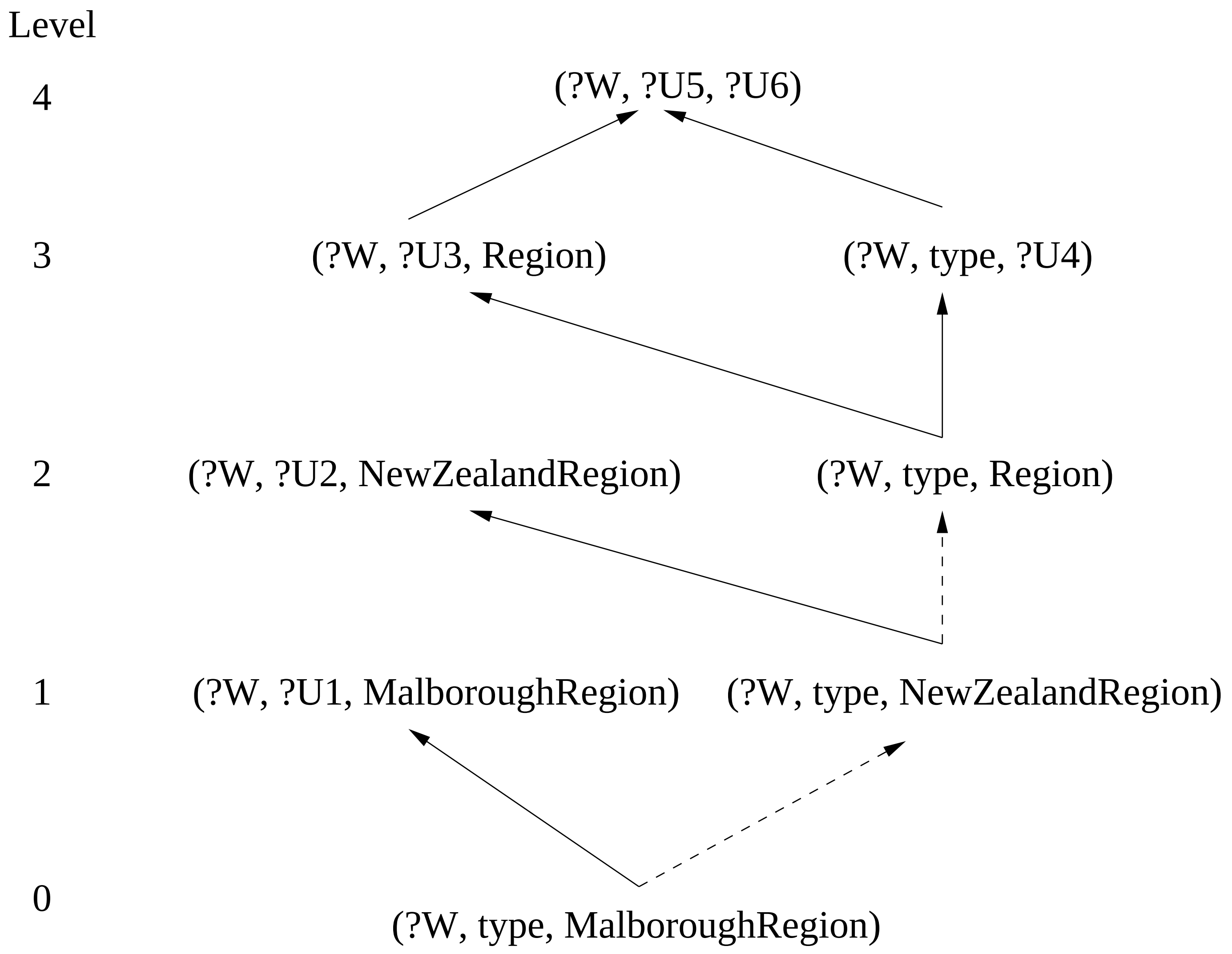
$$\begin{aligned} ?Y, ?Z \quad \leftarrow \quad & (?X, \mathit{hasName}, ?Y), (?X, \mathit{hasPrice}, ?Z), \\ & \mathtt{RELAX}(?X, \mathtt{type}, \mathit{SauvignonBlanc}), \\ & (?X, \mathit{locatedIn}, ?W) \\ & \mathtt{RELAX}(?W, \mathtt{type}, \mathit{MalboroughRegion}). \end{aligned}$$

- returns names and prices of wines of type Sauvignon Blanc from the Malborough region
- relaxation graphs of $(?X, \mathtt{type}, \mathit{SauvignonBlanc})$ and $(?W, \mathtt{type}, \mathit{MalboroughRegion})$ on next 2 slides

Relaxation graph of (?X, type, SauvignonBlanc)



Relaxation graph: (?W, type, MalboroughRegion)



Algorithm for computing ranked, relaxed answers

- assumes triples of RDF graph are stored in a single statement table G
- assumes an operator $\texttt{deltaFind}(t, G)$ that, given triple pattern t and table G, returns triples in G that match t but no triple pattern below t in its relaxation graph

Input: a query Q (interpreted over an ontology O), where $\texttt{Body}(Q) = \{t_1, \ldots, t_n\}$, a statement table G, and an integer *maxLevel*

Output: the set $\texttt{ans}_{\texttt{relax}}(Q, G, \mathit{maxLevel})$ where new answers are returned successively at each level of the relaxation graph.

Algorithm for computing ranked, relaxed answers

1. $k := 0$, $\mathit{stillMore} := \mathit{true}$
2. For each triple pattern $t_i \in \texttt{Body}(Q)$, compute the relaxation graph R_i of t_i up to level $\mathit{maxLevel}$
3. While ($k \leq \mathit{maxLevel}$ and $\mathit{stillMore}$) do
   (a) For each combination $t'_1 \in R_1, \ldots, t'_n \in R_n$ such that $\sum_i \mathit{level}(t'_i, R_i) = k$ output

   $$\pi_H(\texttt{deltaFind}(t'_1, G) \bowtie \ldots \bowtie \texttt{deltaFind}(t'_n, G))$$

   (b) $k := k + 1$
   (c) $\mathit{stillMore} :=$ there exist nodes $t'_1 \in R_1, \ldots, t'_n \in R_n$ such that $\sum_i \mathit{level}(t'_i, R_i) = k$

Example of relaxed query—level 0

$$
\begin{aligned}
?Y, ?Z \quad \leftarrow \quad & (?X, hasName, ?Y), (?X, hasPrice, ?Z), \\
& (?X, \texttt{type}, SauvignonBlanc), \\
& (?X, locatedIn, ?W) \\
& (?W, \texttt{type}, MalboroughRegion).
\end{aligned}
$$

- returns names and prices of wines of type Sauvignon Blanc from the Malborough region
- from now on, we just consider the 3rd and 5th triple patterns, the ones being relaxed

Example of relaxed queries—level 1

Ontology relaxations give rise to the following 2 queries:

$$(?X, \texttt{type}, \textit{SauvignonBlanc}), (?W, \texttt{type}, \textit{NewZealandRegion})$$

- Sauvignon Blanc wines from New Zealand, e.g. from Hawkes Bay

$$(?X, \texttt{type}, \textit{WhiteWine}), (?W, \texttt{type}, \textit{MalboroughRegion})$$

- wines from the Malborough region of type WhiteWine, e.g. a Chardonnay

Example of relaxed queries—level 1

Simple relaxations give rise to the following 2 queries:

$$(?X, \texttt{type}, \mathit{SauvignonBlanc}), (?W, ?U_1, \mathit{MalboroughRegion})$$

- Sauvignon Blanc wines located in regions that are directly connected in some way to the Malborough region

$$(?X, ?V_1, \mathit{SauvignonBlanc}), (?W, \texttt{type}, \mathit{MalboroughRegion})$$

- wines from the Malborough region that are directly connected in some way to Sauvignon Blanc

Conclusion and future work

- developed a framework for query relaxation and answer ranking for RDF
  - useful when user lacks knowledge of the ontology
  - data represents concepts with heterogeneous properties
- potentially applicable to other languages such as OWL
- we would like to generalize relaxation
  - to graph patterns rather than triple patterns
  - to queries involving disjunction, . . .

Conclusion and future work

- we could include other forms of relaxation
  - e.g., breaking join dependencies i.e. shared variables
  - adding triple patterns $(?X_i, \texttt{equal}, ?X_j)$
  - each equality clause can now also be subject to relaxation
  - e.g., to find resources connected by some path
- notion of ranking can be made much more sophisticated
  - to include similarity measures, e.g.