Constrained Regular Expressions in SPARQL

Faisal Alkhateeb, Jean-François Baget, Jérôme Euzenat

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Constrained Regular Expressions in SPARQL

Faisal Alkhateeb — Jean-François Baget — Jérôme Euzenat

N° 6360

October 2007

Thème SYM

ISSN 0249-6399 ISRN INRIA/RR--6360--FR+ENG

Constrained Regular Expressions in SPARQL

Faisal Alkhateeb , Jean-François Baget , Jérôme Euzenat

Thème SYM — Systèmes symboliques
Projet EXMO

Rapport de recherche n° 6360 — October 2007 — 23 pages

Abstract: RDF is a knowledge representation language dedicated to the annotation of resources within the Semantic Web. Though RDF itself can be used as a query language for an RDF knowledge base (using RDF consequence), the need for added expressivity in queries has led to the definition of the SPARQL query language. SPARQL queries are defined on top of graph patterns that are basically RDF (and more precisely GRDF) graphs. To be able to characterize paths of arbitrary length in a query (*e.g.*, "does there exist a trip from town A to town B?"), we have already proposed the PRDF (for Path RDF) language, effectively mixing RDF reasonings with database-inspired regular paths. However, these queries do not allow expressing constraints on internal nodes (*e.g.*, "Moreover, one of the stops must provide a wireless connection."). To express these constraints, we present here an extension of RDF, called CRDF (for Constrained paths RDF). For this extension of RDF, we provide an abstract syntax and an extension of RDF semantics. We characterize query answering (the query is a CRDF graph, the knowledge base is an RDF graph) as a case of CRDF entailment that can be computed using a particular of graph homomorphism. Finally, we use CRDF graphs to generalize SPARQL graph patterns, defining the CSPARQL extension of that query language, and prove that the problem of query answering using only CRDF graphs is an NP-hard problem.

Key-words: semantic web, RDF, SPARQL, constrained regular expressions, graph homomorphism, paths, SPARQL extensions.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38334 Montbonnot Saint Ismier (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Expressions régulières contraintes dans SPARQL
Rapport de recherche
INRIA

Résumé : RDF est un langage de représentation de connaissances dédié à l'annotation de ressources dans le cadre du web sémantique. Bien que RDF lui-même peut être utilisé comme un langage de requête pour une base de connaissances RDF (utilisant conséquence RDF), la nécessité d'ajouter expressivité dans les requêtes a conduit à la définition du langage de requête SPARQL. Les requêtes SPARQL sont définies au sommet des graphes patterns qui sont fondamentalement RDF (et plus précisément GRDF) graphes. Pour exprimer les chemins d'un longueur arbitraire dans une requête (*e.g.*, "existe-t-il un chemin de la ville A à la ville B?"), Nous avons déjà proposé le langage PRDF (pour Path RDF). Cependant, les requêtes PRDF ne permettent pas d'exprimer des contraintes sur les nIJuds internes (*e.g.*, "En outre, l'un des arrêts doit fournir une connexion sans fil."). Pour exprimer ces contraintes, nous présentons ici une extension de RDF, appelé CRDF (pour Constrained Path RDF). Pour cette extension de RDF, nous proposons une syntaxe abstraite et une extension de RDF sémantique. Nous caractérisons la réponse à la requête (la requête est un graphe CRDF, et la base de connaissances est un graphe RDF) comme un cas particulier de la conséquence CRDF qui peut être calculé en utilisant une sorte d'homomorphisme. Enfin, nous utilisons les graphes CRDF de généraliser SPARQL, en définissant l'extension CSPARQL de ce langage des requêtes, et de prouver que le problème de répondre à des requêtes en utilisant uniquement les graphes CRDF est un NP difficile problème.

Mots-clés : web sémantique, RDF, SPARQL, expressions régulières contraintes, homomorphisme, chemins, extensions de SPARQL.

Contents

1 Introduction **4**

2 Motivating Examples – Introducing CPSPARQL **5**

3 GRDF **6**
3.1 GRDF syntax 6
3.2 GRDF semantics 7
3.3 Inference mechanism for GRDF 8

4 CPRDF: syntax **10**

5 CPRDF: semantics **11**
5.1 Generated language 11
5.2 Interpretations and models of CPRDF graphs 13

6 Inference mechanism for CPRDF **14**

7 CPSPARQL **19**
7.1 Syntax 19
7.2 Answers to CPSPARQL queries 19

8 Related Work **21**

9 Conclusion **21**

1 Introduction

RDF (Resource Description Framework, [20]) is a knowledge representation language dedicated to the annotation of documents and more generally of resources within the Semantic Web. In its abstract syntax, an RDF document is a set of triples *(subject, predicate, object)*, that can be represented by a directed labeled graph (hence the name *"RDF graph"*). The language is provided with a model-theoretic semantics [16], that defines the notion of consequence between two RDF graphs. Answers to an RDF query (the knowledge base and the query are RDF graphs) are determined by consequence, and can be computed using a particular *map* (a mapping from terms of the query to terms of the knowledge base that preserves constants), a *graph homomorphism* [14, 5].

SPARQL [22] is one of the languages developed in order to query an RDF knowledge base [15]. The heart of a SPARQL query, the graph pattern, is an RDF graph (and more precisely a Generalized RDF graph allowing variables as predicates, as done in [23]). The maps that are used to compute answers to a graph pattern query in an RDF knowledge base are exploited by [8] to define answers to the more complex, more expressive SPARQL queries (using, for example, disjunctions or functional constraints).

For added expressivity, we have proposed an extension of RDF, that allows expressing paths of arbitrary length [2, 3]. This language, called PRDF (for Path RDF), allows using regular expressions as predicates in an RDF triple. As done before in databases [11, 12, 1, 6], these regular expressions can encode *regular paths* in an RDF graph (the concatenation of the labels of arcs in each path form a word that belongs to the language generated by the regular expression). As a particular case of PRDF entailment, PRDF queries (the query is a PRDF graph, the knowledge base is an RDF graph) can be computed using maps. Using PRDF queries, we can ask questions of the form: "does there exist a trip from town A to town B?".

However, PRDF does not allow specifying properties on the nodes that belong to a path defined by a regular expression. It is thus impossible, for example, to enrich the previous query by "One of the stops must provide a wireless access.". This paper presents an extension of PRDF, called CPRDF (for Constrained path RDF), that allows such constraints. As done for PRDF, we provide an abstract syntax for the language, extend the model-theoretic semantics of RDF, and characterize answers to a CPRDF query in an RDF knowledge base as entailment.

Since CPRDF query answers are maps, we can use the SPARQL definition framework of [21] to extend SPARQL: by considering the graph patterns of SPARQL as CPRDF graphs, we obtain the CPSPARQL extension of that query language. We have already implemented this language[1].

The remainder of the report is structured as follows. We start in Sect. 2 with some motivating examples which cannot be expressed by SPARQL and require to constrain paths. Sect. 3 is devoted to the presentation of the GRDF language, an extension of RDF with variables as predicates. The presentation framework of the CPRDF language is as follows: we first define the abstract syntax of the language in Sect. 4, then its semantics by extending the standard RDF model-theoretic semantics in Sect. 5. We characterize a particular case of entailment in the considered language, where the knowledge base is an RDF graph in Sect. 6. In this case, query answering can be computed by maps. CPRDF graphs (respectively, the maps) are used in Sect. 7 to extend the SPARQL query language to

[1] `http://psparql.inrialpes.fr/`

CPSPARQL (respectively, to answer CPSPARQL queries). After a review of related work (Sect. 8), we conclude in Sect. 9.

2 Motivating Examples – Introducing CPSPARQL

The following example queries attempt to give an insight of CPSPARQL.

Example 1 *Consider the RDF graph G of Fig. 1, that represents the transportation means between cities, the type of the transportation mean, and the price of tickets. For example, the existence of two triples like (*`flight`*,* `ex:from`*,* `C1`*) and (*`flight`*,* `ex:to`*,* `C2`*) means that* `C2` *is directly reachable from* `C1` *using* `flight`*.*

Suppose someone wants to go from Roma to a city in one of the Canary Islands. The following SPARQL query finds the name of such city with only direct trips (no paths):

```
SELECT ?City
WHERE { ?Trip ex:from ex:Roma . ?Trip ex:to ?City .
        ?City ex:cityIn ex:CanaryIslands . }
```

Nonetheless, SPARQL cannot express indirect trips with variable length paths. We can express that using regular expressions with the following (C)PSPARQL query:

```
SELECT ?City
WHERE { ex:Roma (ex:from-.ex:to)+ ?City .
        ?City ex:cityIn ex:CanaryIslands . }
```

Where "-" is the inverse operator. For example, given the RDF triple (`ex:Paris`*,* `ex:from`*,* `ex:flight1`*), we can deduce (*`ex:flight1`*,* `ex:from-`*,* `ex:Paris`*).*

Suppose that he want to use only planes. To do that, we first define a constraint that consists of a name, interval delimiters to include or exclude path node extremities, a quantifier, and a variable is used to be substituted by nodes, and a graph to be matched. For example, the name of the constraint in the following query is `const1`*, it is open from left and universal which ensures that all trips are of type plane.*

```
SELECT ?City
WHERE { CONSTRAINT const1 ]ALL ?Trip]: { ?Trip rdf:type ex:Plane . }
        ex:Roma (ex:from-%const1%.ex:to)+ ?City .
        ?City ex:cityIn ex:CanaryIslands . }
```

Moreover, if the user cannot go out the European union, e.g., for the visa problem, then we will require all intermediate stops to be cities in Europe.

```
SELECT ?City
WHERE { CONSTRAINT const1 ]ALL ?Trip]: { ?Trip rdf:type ex:Plane . }
        CONSTRAINT const2 ]ALL ?Stop]: { ?Stop ex:cityIn ?Country .
                                         ?Country ex:partOf ex:Europe . }
        ex:Roma (ex:from-%const1%.ex:to%const2%)+ ?City .
        ?City ex:cityIn ex:CanaryIslands . }
```

The price of each direct trip is no more than 500*:*

```
SELECT ?City
WHERE { CONSTRAINT const1 ]ALL ?Trip]: { ?Trip rdf:type ex:Plane .
                                         ?Trip ex:price ?Price .
                                         FILTER (?Price < 500) }
        CONSTRAINT const2 ]ALL ?Stop]: { ?Stop ex:cityIn ?Country .
                                         ?Country ex:partOf ex:Europe . }
        ex:Roma (ex:from-%const1%.ex:to%const2%)+ ?City .
        ?City ex:cityIn ex:CanaryIslands . }
```

Suppose we want that the price of the whole trip is no more than 1000, then we can use the `SUM` *function in the following query:*

```
SELECT ?City
WHERE { CONSTRAINT const1 SUM(?Sum1,?Price) ]ALL ?Trip]:
                                        { ?Trip rdf:type ex:Plane .
                                          ?Trip ex:price ?Price .
                                          FILTER (SUM(?Sum1,?Price) < 1000) }
        CONSTRAINT const2 ]ALL ?Stop]: { ?Stop ex:cityIn ?Country .
                                         ?Country ex:partOf ex:Europe . }
        ex:Roma (ex:from-%const1%.ex:to%const2%)+ ?City .
        ?City ex:cityIn ex:CanaryIslands . }
```

As we can see, CPSPARQL is definitely a more expressive language than SPARQL. We will now present it in details.

3 GRDF

The section is dedicated to the presentation of GRDF, an extension of *simple RDF* (RDF language with simple semantics [16]), that allows the use of variables as predicates in triples (as done in [23]). The decision to use simple RDF as the basic building block for our extensions (and not RDF or RDFS) is justified by the fact that RDF and RDFS entailments are obtained from simple RDF entailments by applying rules to the knowledge base (a polynomial procedure) [16]. The same framework could easily be applied to CPRDF to extend, for example, our languages to CPRDFS. For the sake of clarity and brevity, we do not discuss these extensions in this paper. Moreover, to simplify notations, and without loss of generality, we do not distinguish here between simple and typed literals.

3.1 GRDF syntax

RDF graphs are usually constructed over the set of urirefs, blanks, and literals [7]. "Blanks" is a vocabulary specific to RDF. Because we want to stress the compatibility of the RDF structure with classical logic, we will use the term variable instead. The specificity of a blank with regard to variables is their quantification. Indeed, a blank in RDF is an existentially quantified variable. We prefer to retain this classical interpretation which is useful when an RDF graph is put in a different context.

Terminology. An *RDF terminology*, noted $\mathcal{T}$, is a union of 3 pairwise disjoint infinite sets of *terms*: the set $\mathcal{U}$ of *urirefs*, the set $\mathcal{L}$ of *literals* and the set $\mathcal{B}$ of *variables*. We call *vocabulary* and use $\mathcal{V} = \mathcal{U} \cup \mathcal{L}$ to denote the set of *names*. From now on, we use the following notations for the

elements of these sets: a variable will be prefixed by `?` (like `?x1`), a literal will be between quotation marks (like `"27"`), remaining elements will be urirefs (like `ex:price`).

Excluding variables as predicates and literals as subject was an unnecessary restriction in the RDF design, that has been relaxed in many RDF extensions. Relaxing these constraints simplifies the syntax and neither changes the RDF semantics nor the computational properties of reasoning. In consequence, we adopt such an extension introduced in [17] and called *generalized RDF graphs*, or simply GRDF graphs. Using variables as predicates in triples is required also for the SPARQL graph patterns: note that it does not cause additional complexity in the entailment problem.

Definition 1 (GRDF graph) *An* RDF graph *is a set of triples of* $(\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times \mathcal{T}$. *A* GRDF graph *(for generalized RDF) is a set of triples of* $\mathcal{T} \times (\mathcal{U} \cup \mathcal{B}) \times \mathcal{T}$.

So, every RDF graph is a GRDF graph. If G is a GRDF graph, we use $\mathcal{T}(G)$, $\mathcal{U}(G)$, $\mathcal{L}(G)$, $\mathcal{B}(G)$, $\mathcal{V}(G)$ to denote the set of terms, urirefs, literals, variables or names that appeared at least in a triple of G. In Sect. 6, we extend these notations to also take into account the terms which appear in the constrained regular expressions. In a triple (s, p, o), s is called the subject, p the predicate and o the object. It is possible to associate to a set of triples G a labeled directed graph[2] where the set of nodes is the set of terms appearing as a subject or object at least in a triple of G, the set of arcs is the set of triples of G, (*i.e.*, if (s, p, o) is a triple, then $s \xrightarrow{p} o$ is an arc). By drawing these graphs, the nodes resulting from literals are represented by rectangles while the others are represented by rectangles with rounded corners. In what follows, we conflate the two views of RDF syntax (as sets of triples or labeled directed graphs). We will then speak interchangeably about its nodes, its arcs, or the triples which make it up.

3.2 GRDF semantics

By providing GRDF with formal semantics, we express the conditions under which a GRDF graph truly describes a particular world (*i.e.*, an interpretation is a model for the graph) [16]. The usual notions of validity, satisfiability and consequence are entirely determined by these conditions.

Definition 2 (Interpretation of a vocabulary) *Let* $V \subseteq \mathcal{U} \cup \mathcal{L}$ *be a vocabulary. An* interpretation *of* V *is a tuple* $I = (I_R, I_P, I_{EXT}, \iota)$ *where:*

- I_R *is a set of* resources *that contains* $V \cap \mathcal{L}$*;*
- $I_P \subseteq I_R$ *is a set of properties;*
- $I_{EXT} : I_P \rightarrow 2^{I_R \times I_R}$ *associates to each property a set of pairs of resources called the* extension *of the property;*
- *the interpretation function* $\iota : V \rightarrow I_R$ *associates to each name in* V *a resource of* I_R*, if* $v \in \mathcal{L}$*, then* $\iota(v) = v$*.*

[2] In fact, a GRDF graph can be represented as a directed labeled multigraph, since there can be many arcs between two given nodes.

Definition 3 (Model of a GRDF graph) *Let $V \subseteq \mathcal{V}$ be a vocabulary, and G be a GRDF graph such that $\mathcal{V}(G) \subseteq V$. An interpretation $I = (I_R, I_P, I_{EXT}, \iota)$ of V is a* model *of G iff there exists a mapping $\iota' : \mathcal{T}(G) \rightarrow I_R$ that extends ι (i.e., $t \in V \cap \mathcal{T}(G) \Rightarrow \iota'(t) = \iota(t)$) such that for each triple (s, p, o) of G, $\iota'(p) \in I_P$ and $(\iota'(s), \iota'(o)) \in I_{EXT}(\iota'(p))$. The mapping ι' is called a* proof *of G in I.*

The following definition is the standard model-theoretic definition of satisfiability validity and consequence. It will be used for CPRDF graphs.

Definition 4 (Satisfiability, validity, consequence) *A graph G is* satisfiable *iff it admits a model. G is* valid *iff for every interpretation I of a vocabulary $V \supseteq \mathcal{V}(G)$, I is a model of G. A graph G' is a* consequence *of a graph G, denoted by $G \models G'$, iff every model of G is also a model of G'.*

In what follows, we use $\models_{\text{GRDF}}$ (respectively, $\models_{\text{CPRDF}}$) to denote GRDF (respectively, CPRDF) consequences.

Proposition 1 (Satisfiability) *Every GRDF graph is satisfiable. The only valid GRDF graph is the empty graph.*

Proof. (Satisfiability) To each GRDF graph G we associate an interpretation of $\mathcal{V}(G)$, denoted $I_{iso}(G)$, called an isomorphic model of G [5, 23]. We prove that $I_{iso}(G)$ is a model of G. It follows that every GRDF graph admits a model, so it is satisfiable. The construction of $I_{iso}(G) = (I_R, I_P, I_{EXT}, \iota)$ can be made as follows:

(i) the set of resources in $I_{iso}(G)$ is the set of terms of G, *i.e.*, $IR = term(G)$;

(ii) the set of properties in $I_{iso}(G)$ is the set of predicates of G, *i.e.*, $IP = pred(G)$;

(iii) the identy $\forall x \in \mathcal{V}(G)$, $\iota(x) = x$;

(iv) $\forall p \in IP$, $I_{EXT}(p) = \{\langle s, o\rangle \in IR \times IR \mid \langle s, p, o\rangle \in G\}$.

Let us prove that $I_{iso}(G)$ is a model of G. Consider the extension ι' of ι to $\mathcal{B}(G)$ defined by $\forall x \in term(G)$, $\iota'(x) = x$. The condition of Def. 3 immediately follows from the construction of $I_{iso}(G)$. Note that ι is a bijection between $term(G)$ and IR.

(Validity) a non empty GRDF graph has no proof in an interpretation in which all properties are interpreted by I_{EXT} as an empty set.

3.3 Inference mechanism for GRDF

The consequence in GRDF is of utmost importance, since it is the basis for query answering. As done in [14], we use maps and homomorphisms to prove consequence and answer queries.

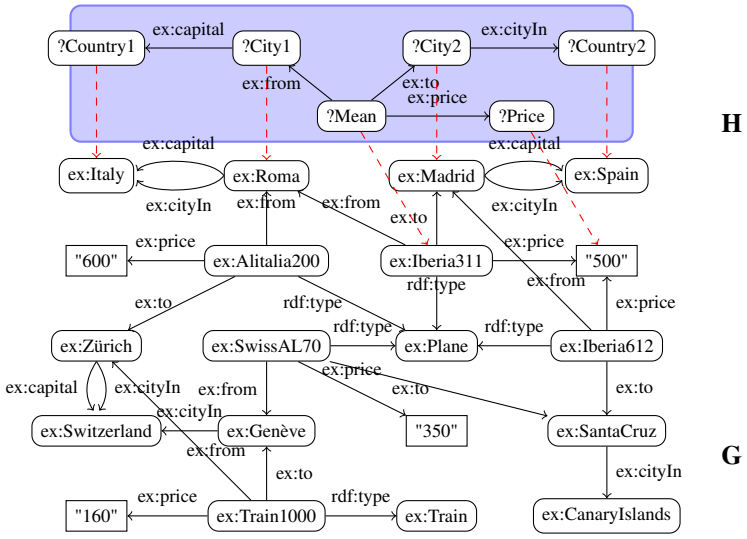

Figure 1: A GRDF homomorphism.

Definition 5 (Map) *Let $V_1 \subseteq \mathcal{T}$, and $V_2 \subseteq \mathcal{T}$ be two sets of terms. A map from V_1 to V_2 is a mapping $\mu : V_1 \rightarrow V_2$ such that $\forall x \in (V_1 \cap \mathcal{V})$, $\mu(x) = x$ (i.e., that preserves urirefs and literals).*

A map μ and an extension ι' of an interpretation function ι are two different mappings, *i.e.*, μ is a mapping from terms to terms that preserves urirefs and literals while ι' is a mapping from terms to resources that preserves the values of ι.

Definition 6 (GRDF homomorphisms) *Let G and G' be two GRDF graphs. A GRDF* homomorphism *from G' into G is a map $\pi : \mathcal{T}(G') \rightarrow \mathcal{T}(G)$ that preserves triples, i.e., such that $\forall (s, p, o) \in G', (\pi(s), \pi(p), \pi(o)) \in G$.*

Theorem 1 *Let G and G' be two GRDF graphs. Then $G \models_{GRDF} G'$ iff there exists a GRDF homomorphism from G' into G.*

The definition of GRDF homomorphisms (Def. 6) is similar to the *map* defined in [14] for RDF graphs. [14] provides, without proof, an equivalence theorem (Theorem 3) between RDF entailment and maps. A proof is provided in [5] for RDF graphs, but the homomorphism involved is a mapping from nodes to nodes, and not from terms to terms. In RDF, the two definitions are equivalent. However, the terms-to-terms version is necessary to extend the theorem of GRDF (Theorem 1) to the CPRDF graphs studied in the rest of this paper. The proof of Theorem 1 will be a particular case of the proof of Theorem 2 for CPRDF graphs.

Proposition 2 (Complexity) *[16] The problem of deciding, given two GRDF graphs G and G', if $G \models_{GRDF} G'$ is* NP*-complete.*

This problem is called RDF ENTAILMENT, and is NP-complete for RDF graphs [16, 14]. For GRDF, the complexity remains unchanged (*cf.* [21]). Polynomial subclasses of the problem can be exhibited based upon the structure or labeling of the query: when the query is ground [23], or more generally when it has a bounded number of variables; when the query is a tree or admits a bounded decompositions into a tree, according to the methods in [13] as shown in [5].

Example 2 *The map π defined by {(`?Country1`, `ex:Italy`), (`?Country2`, `ex:Spain`), (`?City1`, `ex:Roma`), (`?City2`, `ex:Madrid`), (`?Mean`, `ex:Iberia311`), (`?Price`, `"500"`)} is a GRDF homomorphism from the GRDF graph H into the RDF G of Fig. 1.*

4 CPRDF: syntax

To be able to express properties on nodes that belong to a regular path, we extend PRDF [3] by adding constraints to a regular expression. For the sake of simplicity and without loss of generality, we restrict the constraints in this section to be GRDF graphs. Then, we parametrize the CPRDF language in the way that allows us to naturally extend it to include more general constraints.

Definition 7 (GRDF constraint) *A* GRDF constraint *is written $\dagger_1 Q x \dagger_2 : C$ where C is a GRDF graph, $\dagger_1$ and $\dagger_2$ are one of the interval delimiters [and], Q is a quantifier either `ALL` or `EXISTS`, and x is a variable (or a term) that labels a node of C.*

A constraint consists of interval delimiters which are used to include or exclude the extremities of a path, a quantifier either `ALL` or `EXISTS`, a variable, and a GRDF graph that must be satisfied by the internal nodes. For example, the constraint defined by]`ALL ?Trip`]: {(`?Trip`, `rdf:type`, `ex:Plane`) } when applied to a regular expression R ensures that all nodes except the source extremity in a path satisfying R are of type plane. Intuitively, a path p satisfies a regular expression R if the word formed by concatenating the labels of the arcs along the path belongs to the language generated by R.

In what follows, we use Φ_{GRDF} to denote the set of GRDF constraints. When this restriction is not necessary, we use Φ to denote a constraint language.

Let Σ be an alphabet. A *language* over Σ is a subset of Σ^*: its elements are sequences of elements of Σ called *words*. A word (non empty) $(a_1, \ldots, a_k)$ is denoted $a_1 \cdot \ldots \cdot a_k$. If $A = a_1 \cdot \ldots \cdot a_k$ and $B = b_1 \cdot \ldots \cdot b_q$ are two words over Σ, then $A \cdot B$ is the word over Σ defined by $A \cdot B = a_1 \cdot \ldots \cdot a_k \cdot b_1 \cdot \ldots \cdot b_q$. A constrained regular expression over $(\mathcal{U}, \mathcal{B}, \Phi)$ can be used to define the language over $(\mathcal{U} \cup \mathcal{B})$.

Definition 8 (Constrained regular expressions) *A* constrained regular expression *over $(\mathcal{U}, \mathcal{B}, \Phi)$ (denoted by $R \in \mathcal{R}(\mathcal{U}, \mathcal{B}, \Phi)$) is defined inductively by:*

- *if $u \in \mathcal{U}$, then u and $u^- \in \mathcal{R}(\mathcal{U}, \mathcal{B}, \Phi)$;*
- *if $b \in \mathcal{B}$, then $b \in \mathcal{R}(\mathcal{U}, \mathcal{B}, \Phi)$;*
- *if $R \in R(\mathcal{U}, \mathcal{B}, \Phi)$, then $(R^+) \in \mathcal{R}(\mathcal{U}, \mathcal{B}, \Phi)$;*

- *if* $R_1, R_2 \in \mathcal{R}(\mathcal{U}, \mathcal{B}, \Phi)$*, then* $(R_1 \cdot R_2)$ *and* $(R_1|R_2)$ *are elements of* $\mathcal{R}(\mathcal{U}, \mathcal{B}, \Phi)$.
- *if* $R \in \mathcal{R}(\mathcal{U}, \mathcal{B}, \Phi)$ *and* $\psi \in \Phi$ *is a constraint, then* $R\%\psi\% \in \mathcal{R}(\mathcal{U}, \mathcal{B}, \Phi)$.

The inverse operator $^-$ handles only atomic expressions. It specifies the orientation of arcs in the paths retrieved (*i.e.*, it inverses the matching of arcs). Moreover, the constraints are not necessarily grouped together and we can have a constrained regular expression of the form $R\%\psi_1\% \ldots \%\psi_k\%$. This allows us to specify at each grouped block different constraint with(out) different variable(s), which is more flexible and general than grouping all constraints in one block.

Informally, a CPRDF[Φ] graph is a graph whose arcs are labeled with constrained regular expressions whose constraints are elements of Φ.

Definition 9 (CPRDF graphs) *A CPRDF[*Φ*] triple is an element of* $(\mathcal{T} \times \mathcal{R}(\mathcal{U}, \mathcal{B}, \Phi) \times \mathcal{T})$*. A CPRDF[*$\Phi$*] graph is a set of CPRDF[*Φ*] triples.*

Example 3 *The CPRDF[*Φ_{RDF}*] graph* H *represented by the following triples:*

```
{(?City1 ex:cityIn ex:Italy), (?City2 ex:cityIn ex:CanaryIslands),
 (?City1 (ex:from-%]ALL ?Trip]: {?Trip rdf:type ex:Plane}%.ex:to)+ ?City2) }
```

when used as a query, finds pairs of cities (`?City1`*,* `?City2`*), one in Italy and the other in the Canary Islands, such that* `?City2` *is reachable from* `?City1` *using only planes.*

5 CPRDF: semantics

To be able to express the semantics of CPRDF[Φ] graphs, we have first to define the language generated by a regular expression. The derivation trees used here are just a visual representation of the more usual inductive definition of derivation [3]. The internal nodes of these trees will be used to define the semantics of constraints.

5.1 Generated language

Definition 10 (Derivation tree) *Let* $R \in \mathcal{R}(\mathcal{U}, \mathcal{B}, \Phi)$ *be a constrained regular expression. A rooted labeled tree with ordered subtrees* A *is called a derivation tree of* R *(denoted* $A \in \mathcal{DT}(R)$*) iff* A *can be constructed inductively in the following way:*

1. *if* $R = a \in (\mathcal{B} \cup \mathcal{U})$*, then* A *is the tree of Fig. 2(a);*
2. *if* $R = (R'^{+})$ *and* $A_1, \ldots, A_k$ *(*$k \geq 1$*) are a set of derivation trees of* $\mathcal{DT}(R')$*, then* A *is the tree of Fig. 2(b);*
3. *if* $R = (u^-)$*, then* A *is the tree of Fig. 2(c);*
4. *if* $R = (R_1 \cdot R_2)$*,* $A_1 \in \mathcal{DT}(R_1)$ *and* $A_2 \in \mathcal{DT}(R_2)$*, then* A *is the tree of Fig. 2(d);*
5. *if* $R = (R_1|R_2)$ *and* $A' \in \mathcal{DT}(R_1) \cup \mathcal{DT}(R_2)$*, then* A *is the tree of Fig. 2(e);*

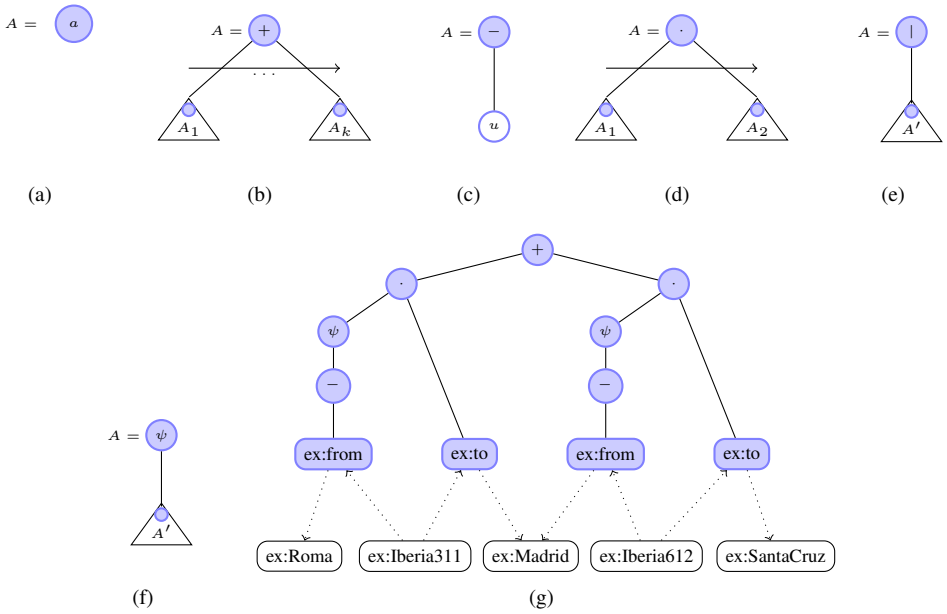

Figure 2: Constructing a derivation tree of a constrained regular expression.

6. if $R = (R'\!<\!\psi\!>)$ and $A' \in \mathcal{DT}(R')$, then A is the tree of Fig. 2(f).

The elements of a derivation tree are quantified using path labels in a given graph, and will be illustrated later through an example.

Definition 11 (Word) *To a derivation tree A we associate a unique word $w(A)$, obtained by concatenating the labels of the leaves of A, totally ordered by the depth-first exploration of A determined by the order of its subtrees. We use $\rho(A, i)$ to denote the i^{th} leaf of A, according to that order.*

The word associated to a derivation tree A of a constrained regular expression R belongs to the language generated by R, as usually defined by $L^*(R) = \{w \in (\mathcal{U} \cup \mathcal{B})^+ \mid \exists A \in \mathcal{DT}(R), w = w(A)\}$. With regard to a more traditional definition of the language generated by a regular expression, our definition ranges over $(\mathcal{U} \cup \mathcal{B})$. This is necessary when extending our work to RDF with variables as predicates [23].

5.2 Interpretations and models of CPRDF graphs

A CPRDF interpretation of a vocabulary $V \subseteq \mathcal{V}$, is an RDF interpretation of V. However, an RDF interpretation must meet specific conditions to be a model for a CPRDF[Φ] graph (Def. 14). These conditions are the transposition of the classical path semantics within the RDF semantics (Def. 12); and the satisfaction of the constraints by the resources of RDF interpretations (Def. 13).

Definition 12 (Support of a constrained regular expression in an interpretation) *Let $I = (I_R, I_P, I_{EXT}, \iota)$ be an interpretation of a vocabulary V, and $R \in \mathcal{R}(\mathcal{U}, \mathcal{B}, \Phi)$ be a constrained regular expression such that $\mathcal{U}(R) \subseteq V$. Let ι' be an extension of ι to $\mathcal{B}(R)$, and $w(A) = a_1 \cdot \ldots \cdot a_k$ be a word of $L^*(R)$. A tuple $(r_0, \ldots, r_k)$ of resources of I_R is called a proof of w in I according to ι' iff $\forall 1 \leq i \leq k$:*

- $\langle r_i, r_{i-1} \rangle \in I_{EXT}(\iota'(a_i))$ *if $\rho(A, i)$ has an ancestor labeled by $^-$;*
- $\langle r_{i-1}, r_i \rangle \in I_{EXT}(\iota'(a_i))$, *otherwise.*

The first item of this definition handles the inverse operator ($^-$): if the ancestor of a_i is labeled by $^-$ (*i.e.*, it is equivalent to a_i^-), then we inverse the two resources that belong to the extension of the property of $\iota'(a_i)$. This definition is used for defining CPRDF models in which it replaces the direct correspondence that exists in RDF between a relation and its interpretation (see first item of Def. 14), by a correspondence between a constrained regular expression and a sequence of relation interpretations. This allows to match constrained regular expressions with variable length paths.

Definition 13 (Satisfaction of a constraint in an interpretation) *Let $\psi = \dagger_1 Q x \dagger_2 : C$ be a constraint of Φ_{RDF}, and $I = (I_R, I_P, I_{EXT}, \iota)$ be an interpretation of a vocabulary V. A resource r of I_R satisfies ψ iff there exists a proof $\iota' : \mathcal{T} \rightarrow I_R$ of C such that $\iota'(x) = r$.*

Now we are ready to define when an interpretation is a model of a CPRDF[Φ_{RDF}] graph.

Definition 14 (Model of a CPRDF graph) *Let $I = (I_R, I_P, I_{EXT}, \iota)$ be an interpretation of a vocabulary V, and G be a CPRDF[Φ_{RDF}] graph such that $\mathcal{U}(G) \subseteq V$. We say that I is a* model *of G iff there exists an extension ι' of ι such that for each triple (s, R, o) of G, there exists a tuple $T = (r_0, \ldots, r_k)$ of resources of I_R ($\iota'(s) = r_0$ and $\iota'(o) = r_k$) and a word $w(A) = a_1 \cdot \ldots \cdot a_k \in L^*(R)$ such that:*

- *T is a proof of w in I according to ι';*
- *for each node z labeled by a constraint $\psi = \dagger_1 Qx \dagger_2 : C$ in A, rooting a subtree A' with $a_p \cdot \ldots \cdot a_{p+q} = w(A')$, then $Q\ r \in \dagger_1 r_{p-1}, \ldots, r_{p+q} \dagger_2$, r satisfies ψ.*

We also say that $\langle \iota'(s), \iota'(o) \rangle$ satisfies R in I according to ι'.

Proposition 3 (Satisfiability) *A CPRDF[Φ_{GRDF}] graph G is satisfiable iff $\forall (s, R, o) \in G$, $L^*(R) \neq \emptyset$.*

Proof. Let G be a CPRDF[Φ_{GRDF}] graph such that $\forall (s, R, o) \in G$, $L^*(R) \neq \emptyset$. To proof that G is satisfiable, we build a canonical model of G as follows:

1. Builds a graph G' by replacing each triple (s, R, p) in G by a set of triples (s, p_1, v_1), $\ldots$, (v_{n-1}, p_n, o) such that $p_1 \cdot \ldots \cdot p_n$ is an arbitrary word in the language generated by R, and $v_{i's}$ are all new distinct variables; and for each constraint $\psi = \dagger_1 Qx \dagger_2 : C$ in R, add to G' the graph C_n^x for each node n in $\{s, v_1, \ldots, v_{n-1}, o\}$, where C_n^x is the graph obtained by substituting each occurrence of x by n.
2. Construct the isomorphic model of G' (see Prop. 1).

6 Inference mechanism for CPRDF

Two conditions must be satisfied for the notion of homomorphism to be able to find the answers to a CPRDF[Φ] query in an RDF knowledge base (Def. 17): instead of proving an arc (a triple) of the query by an arc in the knowledge base, we prove it by a path in the knowledge base (Def. 15); and the satisfaction of the corresponding node(s) in the path of the knowledge base to the corresponding constraint(s) (Def. 16).

Definition 15 (Path word) *Let G be an RDF graph of vocabulary $V \subseteq \mathcal{V}$, and $R \in \mathcal{R}(\mathcal{U}, \mathcal{B}, \Phi)$ be a constrained regular expression such that $\mathcal{U}(R) \subseteq V$. Let $\mu : \mathcal{B}(R) \rightarrow V$ be a map from the variables of R to V, and $w(A) = a_1 \cdot \ldots \cdot a_k$ be a word of $L^*(R)$. A tuple $(n_0, \ldots, n_k)$ of nodes of G is called a* path *of w in G according to μ iff $\forall 1 \leq i \leq k$:*

- *$(n_i, \mu(a_i), n_{i-1}) \in G$ if $\rho(A, i)$ has an ancestor labeled by $^-$;*
- *$(n_{i-1}, \mu(a_i), n_i) \in G$, otherwise.*

As done for the interpretation (Def. 12), the first item handles the inverse operator: if the ancestor of a_i is labeled by $^-$, then we inverse the orientation of the arc.

Example 4 *Figure fig:aderivationtree shows a possible derivation tree of the constrained regular expression R =(ex:from-%ψ%ex:to)$^+$ of the CPRDF[Φ_{GRDF}] graph H of Ex. 3, where ψ =]ALL ?Trip]: {(?Trip, rdf:type, ex:Plane)}. The nodes in white color, which correspond to the path of nodes in the RDF graph G of Figure fig:rdfhomomorphism, together with the path labels are used to quantify the elements of the tree. The tuple T=(ex:Roma, ex:Iberia311, ex:Madrid, ex:Iberia612, ex:SantaCruz) of nodes in the RDF graph G of Figure fig:rdfhomomorphism is a path of the word w=(ex:from-· ex:to·ex:from-·ex:to) $\in L^*(R)$ according to the empty map.*

The following definition gives the condition(s) when a constraint of Φ_{GRDF} is satisfied. This definition can be extended based on the constraints (*cf.* Note 2).

Definition 16 (Satisfaction of a constraint in a GRDF graph) *Let G be a GRDF graph, $\psi = \dagger_1 Q\ x \dagger_2 : C$ be a constraint of Φ_{GRDF}, and s a term of G. We say that s satisfies ψ in G if there exists a GRDF homomorphism π from C into G such that $\pi(x) = s$.*

Intuitively, in CPRDF[Φ] homomorphisms, each internal node labeled by a constraint ψ of a derivation tree determines the subtree (not necessary the whole tree, since a constraint ψ may be applied to a partial part of a constrained regular expression, Def. 8) whose corresponding nodes in the knowledge base graph must satisfy ψ (see the second item of the following definition).

Definition 17 (CPRDF homomorphism) *Let G be an RDF graph and H be a CPRDF[Φ] graph. A* CPRDF[Φ] homomorphism *from P into G is a map $\pi : \mathcal{T}(H) \rightarrow \mathcal{T}(G)$ such that $\forall (s, R, o) \in H$, there exists a tuple $T = (n_0, \ldots, n_k)$ of nodes of G ($\pi(s) = n_0$ and $\pi(o) = n_k$) and a word $w(A) = a_1 \cdot \ldots \cdot a_k \in L^*(R)$ such that:*

- *T is a path of w in G according to π;*
- *for each node z labeled by a constraint $\psi = \dagger_1 Q x \dagger_2 : C$ in A, rooting a subtree A' with $a_p \cdot \ldots \cdot a_{p+q} = w(A')$, then $Q\ n \in \dagger_1 n_{p-1}, \ldots, n_{p+q} \dagger_2$, n satisfies ψ.*

We say that $\langle \pi(s), \pi(o) \rangle$ satisfies R in G according to π.

The existence of a CPRDF[Φ] homomorphism is exactly what is needed for deciding entailment between RDF and CPRDF[Φ] graphs.

Theorem 2 (CPRDF-RDF entailment) *Let G be an RDF graph, and H be a CPRDF[Φ_{GRDF}] graph. Then $G \models_{CPRDF} H$ iff there exists a CPRDF[Φ_{GRDF}] homomorphism from H into G.*

Proof. Let G be an RDF graph, H be a CPRDF[ϕ_{GRDF}] graph and $I = (I_R, I_P, I_{EXT}, \iota)$ be an interpretation of a vocabulary $V = U \cup L$ such that $\mathcal{V}(G) \subseteq V$ and $\mathcal{V}(H) \subseteq V$. We prove both directions of the theorem as follows. We first add to G, for each triple (s, p, o) in G, the triple (s, p^-, o). This way we can ignore the first item of Def. 17 and Def. 14.

($\Rightarrow$) Suppose that there exists a CPRDF[Φ_{GRDF}] homomorphism from H into G, *i.e.*, $\pi : term(H) \rightarrow term(G)$. We want to prove that $G \models_{\text{CPRDF}} H$, *i.e.*, that every model of G is a model of H.

If I is a model of G, then there exists an extension ι' of ι to $\mathcal{B}(G)$ such that $\forall \langle s, p, o\rangle \in G$, $\langle \iota'(s), \iota'(o)\rangle \in I_{EXT}(\iota'(p))$ (Def. 3). We want to prove that I is also a model of H, *i.e.*, there exists an extension ι'' of ι to $\mathcal{B}(H)$ such that $\forall \langle s, R, o\rangle \in H$, $\langle \iota''(s), \iota''(o)\rangle$ supports R in I according to ι''.

Let ι'' be the map defined by:

$$\forall x \in \mathcal{T}, \iota''(x) = \begin{cases} (\iota' \circ \pi)(x) & \text{if } \pi \text{ is defined;} \\ \iota'(x) & \text{otherwise.} \end{cases}$$

.

We show that ι'' verifies the following properties:

1. I is an interpretation of $\mathcal{V}(H) \cap nodes(H)$.[3]
2. ι'' is an extension to variables of H, *i.e.*, $\forall x \in \mathcal{V}(H) \cap \mathcal{V}(G)$, $\iota''(x) = \iota(x)$.
3. ι'' satisfies the conditions of CPRDF[Φ_{GRDF}] models (Def. 14), *i.e.*, for every triple $\langle s, R, o\rangle \in H$, the pair of resources $\langle \iota''(s), \iota''(o)\rangle$ supports R in I according to ι''.

Now, we prove the satisfaction of these properties:

1. Since each term $x \in \mathcal{V}(H) \cap nodes(H)$ is mapped by π to a term $x \in \mathcal{V}(G)$ and I interprets all $x \in \mathcal{V}(G)$, I interprets all $x \in \mathcal{V}(H) \cap nodes(H)$.
2. Since π is a map (Def. 17), we have $\forall x \in \mathcal{V}(H) \cap \mathcal{V}(G)$, if π is defined, $\pi(x) = x$ (Def. 5). Hence, we have $\iota''(x) = (\iota' \circ \pi)(x) = \iota'(x) = \iota(x)$, $\forall x \in \mathcal{V}(H) \cap \mathcal{V}(G)$.
3. It remains to prove that for every triple $\langle s, R, o\rangle \in H$, the pair of resources $(\iota''(\pi(s)), \iota''(\pi(o)))$ supports R in ι'' (Def. 14). By the definition of CPRDF[Φ_{GRDF}] homomorphism (Def. 17), we have:

   (i) $\forall (s, R, o) \in H$, there exists a tuple $T = (n_0, \ldots, n_k)$ of nodes of G (with $\pi(s) = n_0$ and $\pi(o) = n_k$) and a word $w(A) = a_1 \cdot \ldots \cdot a_k \in L^*(R)$ such that T is a path of w in G according to π. From the definition of path (Def. 15), $(n_{i-1}, \pi(a_i), n_i) \in G$ such that $n_0 = \pi(s)$, $n_k = \pi(o)$. It follows that $(\iota'(\pi(s)), \iota'(n_1)) \in I_{EXT}(\iota'(\pi(a_1)))$, …, $(\iota'(n_{k-1}), \iota'(\pi(o))) \in I_{EXT}(\iota'(\pi(a_k)))$ (Def. 3, GRDF models). So, by Def. 12, the tuple of resources T_r defined by $T_r = (\iota''(\pi(s)) = \iota'(n_0) = r_0, r_1, \ldots, r_{k-1}, r_k = \iota'(n_k) = \iota''(\pi(o)))$ (with $r_i = n_i$, $1 \leq i \leq k-1$) is a proof of w in I according to $(\iota' \circ \pi)$. Since $\iota'' = (\iota' \circ \pi)$, we have T_r is also a proof of w in I according to ι''.

[3] Note that an interpretation I can be a model of a given CPRDF[ϕ] graph H even it does not interpret all terms of H. This is due to the disjunction operator that occurs inside constrained regular expressions.

(ii) For each node z labeled by a constraint $\phi = \dagger_1 Qx\dagger_2 : C$ in A, rooting a subtree A' with $a_p \cdot \ldots \cdot a_{p+q} = w(A')$, then $Q\ n \in \dagger_1 n_{p-1}, \ldots, n_{p+q}\dagger_2$, n satisfies ϕ. By Def. 16, n satisfies ϕ in G if there exists a GRDF homomorphism π_1 from C into G such that $\pi_1(x) = n$. Using Theorem 1 and Def. 3, there exists a proof $\iota_G : \mathcal{T} \rightarrow I_R$ of C such that $\iota_G(x) = \iota'(n)$. So, $Q\ r \in \dagger_1 r_{p-1}, \ldots, r_{p+q-1}\dagger_2$, r satisfies ϕ (with $r_i = \iota'(n_i)$).

The conditions of CPRDF[Φ_{GRDF}] models are satisfied. Hence, *i.e.*, every model of G is a model of H.

($\Leftarrow$) Suppose that $G \models_{\text{CPRDF}} H$. We want prove that there is a CPRDF[Φ_{GRDF}] homomorphism from H into G.

Every model of G is also a model of H. In particular, $I_{iso} = (I_R, I_P, I_{EXT}, \iota)$ the isomorphic model of G, where there exists a bijection ι' between $term(G)$ and IR (see Prop. 1). ι' is an extension of ι to $\mathcal{B}(G)$ such that $\forall \langle s, p, o\rangle \in G$, $\langle \iota'(s), \iota'(o)\rangle \in I_{EXT}(\iota'(p))$ (Def. 3). Since I_{iso} is a model of H, there exists an extension ι'' of I_{iso} to $\mathcal{B}(H)$ such that $\forall \langle s, R, o\rangle$, $\langle \iota''(s), \iota''(o)\rangle$ supports R in ι'' (Def. 14). Let us consider the function $\pi = (\iota'^{-1} \circ \iota'')$. To prove that π is a CPRDF[Φ_{GRDF}] homomorphism from H into G, we must prove that:

1. π is a map from $term(H)$ into $term(G)$;
2. $\forall x \in \mathcal{V}(H)$, $\pi(x) = x$;
3. $\forall \langle s, R, o\rangle \in H$, the pair of nodes $(\pi(s), \pi(o))$ satisfies R in G according to π.

Let us prove these properties.

1. Since ι'' is a map from $term(H)$ into I_R and ι'^{-1} is a map from I_R into $term(G)$, $\pi = (\iota'^{-1} \circ \iota'')$ is clearly a map from $term(H)$ into $term(G)$ ($term(H) \xrightarrow{\iota''} IR \xrightarrow{\iota'^{-1}} term(G)$).
2. From the definition of an extension: $\forall x \in \mathcal{V}(H)$, $\iota''(x) = \iota(x)$. Since ι' is a bijection, $\forall x \in \mathcal{V}(H)$, $(\iota'^{-1} \circ \iota'')(x) = (\iota'^{-1} \circ \iota)(x) = x$.
3. Since ι'' is a proof of H, by definition of CPRDF[Φ_{GRDF}] models (Def. 14), we have:

   (i) For each triple (s, R, o) of H, there exists a tuple $T = (r_0, \ldots, r_n)$ of resources of I_R (with $\iota''(s) = r_0$ and $\iota''(o) = r_n$) and a word $w(A) = a_1 \cdot \ldots \cdot a_k \in L^*(R)$ such that T is a proof of w in I according to ι''. By Def. 12, $\langle r_{i-1}, r_i\rangle \in I_{EXT}(\iota''(a_i))$ with $\iota''(s) = r_0$ and $\iota''(o) = r_n$, $1 \leq i \leq k$. It follows that $\langle n_{i-1}, p_i, n_i\rangle \in G$ with $n_i = \iota'^{-1}(r_i)$, and $p_i = (\iota'^{-1} \circ \iota'')(a_i)$ (construction of $I_{iso}(G)$, see Prop. 1). We have, $(\iota'^{-1} \circ \iota'')(s) = \iota'^{-1}(r_0) = n_0$, $(\iota'^{-1} \circ \iota'')(o) = \iota'^{-1}(r_k) = n_k$, and the word w defined by $w = p_1 \cdot \ldots \cdot p_k \in L^*((\iota'^{-1} \circ \iota'')(R))$. So the tuple of nodes T_n defined by $T_n = ((\iota'^{-1} \circ \iota'')(s) = \iota'(r_0) = n_0, n_1, \ldots, n_{k-1}, n_k = (\iota'^{-1} \circ \iota'')(o))$ is a path of w in G according to $(\iota'^{-1} \circ \iota'') = \pi$.

(ii) For each node z labeled by a constraint $\phi = \dagger_1 Qx\dagger_2 : C$ in A, rooting a subtree A' with $a_p \cdot \ldots \cdot a_{p+q} = w(A')$, then $Q\ r \in \dagger_1 r_{p-1}, \ldots, r_{p+q}\dagger_2$, r satisfies ϕ. By Def. 13, r satisfies ϕ iff there exists a proof $\iota_G : \mathcal{T} \rightarrow I_R$ of G such that $\iota_G(x) = r$. Using the equivalence between GRDF homomorphism and RDF entailment (Theorem 1), there exists a GRDF homomorphism π_1 from C into G such that $\pi_1(x) = \iota'^{-1}(r) = n$.

Hence, π is a CPRDF[Φ_{GRDF}] homomorphism from H into G.

Proposition 4 (Complexity) *The problem of deciding, given an RDF graph G and a CPRDF[Φ_{GRDF}] G', if $G \models_{CPRDF} G'$ is* NP*-complete.*

Proof. Checking if $G \models_{\text{CPRDF}} G'$ is equivalent to checking the existence of a CPRDF[Φ_{GRDF}] homomorphism from G' into G (Theorem 2). So, it is sufficient to show that checking the existence of a CPRDF[Φ_{GRDF}] homomorphism from G' into G is NP-complete.

When G' does not contain constraints, *i.e.*, G' is a PRDF graph, then the problem is NP-complete [3]. We describe an algorithm showing that adding constraints does not change this complexity as follows:

- We first add to G, for each triple (s, p, o) in G, the triple (s, p^-, o) (which can be done in polynomial time in size of G).
- Calculate all necessary homomorphisms from the graphs of constraints of G' into G a priori only one time (the problem of evaluating a union of GRDF graphs is a NP-complete [21]). Suppose that $\Gamma = \{\psi_i \mid \ \psi_i \text{ is a constraint in } G'\}$, and Ω_i is the set of homomorphisms from the graph of the constraint ψ_i into G.
- Now, testing whether if each node n satisfies a given constraint ψ_i in the knowledge base is equivalent to testing if the there exists an homomorphism from the graph of ψ_i into the knowledge base, $\pi \in \Omega_i$ with $\pi(x) = n$, where x is the variable in ψ_i. The later can be done in linear time in the size of Ω_i (if we assume that checking if $\pi(x) = n$ can be done in $\mathcal{O}(1)$, otherwise it can be in polynomial time).

Example 5 *Consider the CPRDF[Φ_{GRDF}] graph H of Ex. 3, the RDF graph G of Fig. 1, and the map π defined by {(`?City1`,`ex:Roma`), (`?City2`,`ex:SantaCruz`), (`ex:from`,`ex:from`), (`ex:to`,`ex:to`), (`ex:cityIn`,`ex:cityIn`), (`?Country`,`ex:CanaryIslands`)}. According to Def. 17, the first condition is satisfied by π (see Ex. 4), and the stops along the path between (`ex:Roma`,`ex:SantaCruz`) are all cities in Europe (see Fig. 2(g)). So, π is a CPRDF[Φ_{GRDF}] homomorphism from H into G.*

7 CPSPARQL

[21] presents an alternate characterization of query answering with the SPARQL query language that relies upon operations on maps from the graph patterns of a query into an RDF knowledge base. We use this framework to extend SPARQL to CPSPARQL, by defining graph patterns as CPRDF[Φ] graphs. Analogously, the set of answers to a CPSPARQL query is defined inductively from the set of maps of the CPRDF[Φ] graphs of the query into the RDF knowledge base.

7.1 Syntax

In CPSPARQL there are several functions that can be used for capturing the values along the paths like `SUM` for summation of values along paths, `AVG` for the average, `COUNT` for counting nodes satisfying constraints. For the sake of simplicity, we have not introduced these function, and illustrate them with examples (*cf.* Sect. 2). Moreover, since the graph patterns in the SPARQL query language are shared by all SPARQL query forms and that our proposal is based upon extending these graph patterns, we illustrate our extension using the SELECT ... FROM ... WHERE ... query form[4]. Our extension can then be applied to other query forms.

CPSPARQL graph patterns are built on top of CPRDF in the same way that SPARQL is built on top of RDF.

Definition 18 (CPSPARQL graph patterns) *A* CPSPARQL[Φ] graph pattern *is defined inductively by:*

- *every CPRDF[Φ] graph is a CPSPARQL[Φ] graph pattern;*
- *if P_1, P_2 are CPSPARQL[Φ] graph patterns and R is a SPARQL constraint, then (P_1 `AND` P_2), (P_1 `UNION` P_2), (P_1 `OPT` P_2), and (P_1 `FILTER` R) are CPSPARQL[Φ] graph patterns.*

Note 1 The parametrization of CPSPARQL[Φ] by Φ allows us to extend naturally its graph patterns to more general constraints. For example, if Φ_{SPARQL} denotes the set of all possible SPARQL graph patterns, then a CPRDF[Φ_{SPARQL}] graph could be a CPSPARQL[Φ_{SPARQL}] graph pattern.

CPSPARQL query. A CPSPARQL[Φ] query is of the form `SELECT` $\vec{B}$ `FROM` u `WHERE` P such that P is a CPSPARQL[Φ] graph pattern.

7.2 Answers to CPSPARQL queries

We first need to introduce some notations and operations in maps. If μ is a map, then the domain of μ, denoted by $dom(\mu)$, is the subset of $\mathcal{T}$ where μ is defined. If P is a graph pattern, then $\mu(P)$ is the graph pattern obtained by the substitution of $\mu(b)$ to each variable $b \in \mathcal{B}(P)$. Two maps μ_1 and μ_2 are *compatible* when $\forall x \in dom(\mu_1) \cap dom(\mu_2)$, $\mu_1(x) = \mu_2(x)$. If $\mu_1 : T_1 \rightarrow \mathcal{T}$ and

[4] SPARQL provides several result forms that can be used for formating the query results. For example, CONSTRUCT that can be used for building an RDF graph from the set of answers, ASK that returns `TRUE` if there is a answer to a given query and `FALSE` otherwise, and DESCRIBE that can be used for describing a resource RDF graph.

$\mu_2 : T_2 \to \mathcal{T}$ are two compatible maps, then we use $\mu = \mu_1 \oplus \mu_2 : T_1 \cup T_2 \to \mathcal{T}$ to denote the map defined by: $\forall x \in T_1, \mu(x) = \mu_1(x)$ and $\forall x \in T_2, \mu(x) = \mu_2(x)$. Analogously to [21], we define the *join* and *difference* of two sets of maps Ω_1 and Ω_2 as follows:

- *(join)* $\Omega_1 \bowtie \Omega_2 = \{\mu_1 \oplus \mu_2 \mid \mu_1 \in \Omega_1, \mu_2 \in \Omega_2 \text{ are compatible}\}$;
- *(difference)* $\Omega_1 \setminus \Omega_2 = \{\mu_1 \in \Omega_1 \mid \forall \mu_2 \in \Omega_2, \mu_1 \text{ and } \mu_2 \text{ are not compatible}\}$.

As in the case of SPARQL, the answer to a query reduced to a CPRDF[Φ] graph is also given by a map. The definition of an answer to a CPSPARQL query will be thus identical to the one given for SPARQL [21], but it will use CPRDF[Φ] homomorphisms.

Definition 19 (Answers to a CPSPARQL graph pattern) *Let P be a CPSPARQL[Φ] graph pattern and G be an RDF graph, then the set $\mathcal{S}(P, G)$ of answers of P in G is defined inductively by:*

- *if P is a CPRDF[Φ] graph, $\mathcal{S}(P, G) = \{\mu \mid \mu$ is a CPRDF[Φ] homomorphism from P into $G\}$;*
- *if $P = (P_1$ AND $P_2)$, $\mathcal{S}(P, G) = \mathcal{S}(P_1, G) \bowtie \mathcal{S}(P_2, G)$;*
- *if $P = (P_1$ UNION $P_2)$, $\mathcal{S}(P, G) = \mathcal{S}(P_1, G) \cup \mathcal{S}(P_2, G)$;*
- *if $P = (P_1$ OPT $P_2)$, $\mathcal{S}(P, G) = (\mathcal{S}(P_1, G) \bowtie \mathcal{S}(P_2, G)) \cup (\mathcal{S}(P_1, G) \setminus \mathcal{S}(P_2, G))$;*
- *if $P = (P_1$ FILTER $R)$, $\mathcal{S}(P, G) = \{\mu \in \mathcal{S}(P_1, G) \mid \mu(R) = \top\}$.*

Note 2 When CPSPARQL graph patterns are constructed over CPRDF[Φ_{SPARQL}] graphs, then we need only to extend Def. 16 in the following way: Let G be a graph, P be a SPARQL graph pattern, $\psi = \dagger_1 Q x \dagger_2 : P$ be a constraint, and s a term of G. We say that s satisfies ψ in G if there exists a map $\mu \in \mathcal{S}(P, G)$ such that $\mu(x) = s$. The definition of CPRDF[Φ] homomorphism (Def. 17) and first item of Def. 19 remain unchanged.

Answers to a CPSPARQL[Φ] query are the instantiations of the set of maps from its graph patterns into the graph representing the knowledge base(s).

Definition 20 (Answers to a CPSPARQL query) *Let $Q =$*SELECT $\vec{B}$ FROM u WHERE P *be a CPSPARQL[Φ] query. Let G be the RDF graph identified by the URL u, and Ω the set of answers of P in G. Then the answers to the query Q are the projections of elements of Ω to $\vec{B}$, i.e., for each map π of Ω, the answer of Q associated to π is $\{(x, y) \mid x \in \vec{B}$ and $y = \pi(x)$ if $\pi(x)$ is defined, otherwise* `null`$\}$.

Proposition 5 *Let G be an RDF graph, P be a CPRDF[Φ_{GRDF}] graph and $\vec{B}$ be a tuple of variables appearing in P, an answer to the CPSPARQL[Φ_{GRDF}] query $Q =$*SELECT $\vec{B}$ FROM u WHERE P *is a CPRDF[Φ_{GRDF}] homomorphism μ such that $G \models_{CPRDF} \mu(P)$.*

This proposition is a straightforward consequence of Def. 19. It is based on the fact that the answers to Q are the restrictions to $\vec{B}$ of the set of CPRDF[Φ_{GRDF}] homomorphisms from P into G which, by Theorem 2, correspond to CPRDF-RDF entailment.

8 Related Work

There are many query languages dealing with paths: G and G+ [11, 12], GraphLog [9], Lorel [1], UnQL [6], WebSQL [19], Corese [10] including our own extension to SPARQL [2, 3]. None of them deal with constraints.

Two extensions of SPARQL, which are closely similar to PSPARQL, have been recently defined based on our initial proposal [2]: SPARQLeR [18] and SPARQ2L [4]. Both languages extend SPARQL by allowing query graph patterns involving path variables. Each path variable is used to capture paths in RDF graphs, and is matched against any arbitrary composition of RDF triples between two given nodes. The constraints in these extensions are simple, *i.e.*, restricted to testing the length of paths and testing if a given node is in the resulting path. The queries in CPSPARQL are examples that can be emulated by neither SPARQ2L nor SPARQLeR. Several problems are shared by the two extensions when we evaluate such graph patterns. In particular, the strategy of obtaining paths and then filtering them is inefficient since it can generate a large number of paths. Multiple uses of same path variable is not fully defined: it is not specified which path is to be returned or if is it enforced to be the same. The effects of paths variables in the DISTINCT clause are not treated. Since SPARQLeR is not defined with a formal semantics, its use of path variables in the subject position is unclear, in particular, when they are not bound. It seems that the algorithms used in SPARQ2L are not complete with regard to their intuitive semantics, since the set of answers can be infinite in absence of constraints for using shortest or acyclic paths.

A kind of constrained regular expressions is used in XPath [8]. However, XPath data model relies on trees (not graphs) and defines only monadic queries (not polyadic).

To our knowledge no other language for querying graphs supports constraints on paths. CPSPARQL allows filtering constraints on the fly (during path search) and not a posteriori, and is not restricted to simple paths. This relaxation is not only useful for many applications (*cf.* [4] for some examples), but also provides polynomial classes for the regular expression satisfiability problem (*i.e.*, when they do not contain variables). The originality of our proposal lies in our adaptation of the RDF model-theoretic semantics to take into account constrained regular expressions, effectively combining the expressiveness of these two languages, and the integration of this combination on top of the most important query language for RDF, SPARQL, providing a wide range of querying paradigms.

9 Conclusion

Our initial proposal, the PSPARQL language, extends SPARQL with PRDF graphs to allow expressing variable length paths. Since PSPARQL and SPARQL do not allow specifying characteristics of the nodes traversed by a regular path, we have extended the PSPARQL language syntax and semantics to handle constraints, and have characterized answers to a CPRDF query in an RDF knowledge base as maps. This property was sufficient to extend the SPARQL query language to CPSPARQL, combining the expressiveness of both SPARQL and CPRDF. We have provided a sound and complete inference mechanism for answering CPSPARQL queries over RDF graphs as well as algorithms for calculating these answers.

The proposed language, CPSPARQL has several advantages. First of all, it allows expressing variable length paths which can be qualified through the use of constraints. It may enhance efficiency, since the task of evaluating path expressions is heavyweight and exhaustive, and the use of predefined constraints inside regular expressions prunes irrelevant paths during the evaluation process and not a posteriori. The constraints in CPSPARQL are extensible (*i.e.*, it can be extended to include constraints that can be more general, *cf.* Sect. 7), partial (*i.e.*, can be applied to a part of a regular expression, see examples in Sect. 2). The use of regular expressions supports a meaningful and natural use of inverse paths through the use of inverse operator. As done for SPARQL, CPRDF graphs can be adapted and integrated in other graph-based query languages.

As it is shown along the paper, we go far beyond the trivial constraints, *i.e.*, testing simple paths and the existence of a node along the path. Extending RDF to RDFS (RDF Schema) does not change the computational properties of the language: finding consequences in RDFS is reduced polynomially to finding consequences in RDF [16]. So, our work extends naturally to RDFS thanks to this reduction. Finally, we have implemented a CPSPARQL query engine that is available for both download and online test.

References

[1] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. L.Wiener. The Lorel query language for semistructured data. *Journal on Digital Libraries*, 1(1):68–88, 1997.

[2] F. Alkhateeb, J.-F. Baget, and J. Euzenat. Complex path queries for RDF graphs. In *ISWC, poster paper*, 2005.

[3] F. Alkhateeb, J.-F. Baget, and J. Euzenat. RDF with regular expressions. Research report 6191, INRIA, Montbonnot (FR), 2007.

[4] K. Anyanwu, A. Maduko, and A. P. Sheth. SPARQ2L: towards support for subgraph extraction queries in RDF databases. In *Proceedings of the 16th international conference on World Wide Web (WWW'07)*, pages 797–806, 2007.

[5] J.-F. Baget. RDF entailment as a graph homomorphism. In *Proceedings of the 4th International Semantic Web Conference (ISWC'05), Galway (IE)*, pages 82–96, 2005.

[6] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A query language and optimization techniques for unstructured data. In *Proceedings of the ACM SIGMOD International Conference on the Management of Data*, pages 505–516, 1996.

[7] J. J. Carroll and G. Klyne. RDF concepts and abstract syntax. W3C recommendation, 2004.

[8] J. Clark and S. DeRose. XML Path Language (XPath). W3C Recommendation, 1999.

[9] M. P. Consens and A. O. Mendelzon. Graphlog: a visual formalism for real life recursion. In *Proceedings of the 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 404–416, 1990.

[10] O. Corby, R. Dieng-Kuntz, and C. Faron-Zucker. Querying the semantic web with corese search engine. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI'2004), sub-conference (PAIS'2004), Valencia (Spain)*, pages 705–709, 2004.

[11] I. F. Cruz, A. O. Mendelzon, and P. T. Wood. A graphical query language supporting recursion. In *Proceedings of the ACM SIGMOD*, pages 323–330, 1987.

[12] I. F. Cruz, A. O. Mendelzon, and P. T. Wood. G+: Recursive queries without recursion. In *Proceedings of the Expert Database Conference*, pages 355–368, 1988.

[13] G. Gottlob, N. Leone, and F. Scarcello. A comparison of structural CSP decomposition methods. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence IJCAI '99*, pages 394–399, 1999.

[14] C. Gutierrez, C. Hurtado, and A. O. Mendelzon. Foundations of semantic web databases. In *ACM Symposium (PODS)*, pages 95–106, 2004.

[15] P. Haase, J. Broekstra, A. Eberhart, and R. Volz. A comparison of RDF query languages. In *Proceedings 3rd International Semantic Web Conference*, pages 502–517, 2004.

[16] P. Hayes. RDF semantics. W3C Recommendation, February 2004.

[17] H. J. Horst. Completeness, decidability and complexity of entailment for RDF schema and a semantic extension involving the OWL vocabulary. *Journal of Web Semantics*, 3(2):79–115, 2005.

[18] K. Kochut and M. Janik. SPARQLeR: Extended sparql for semantic association discovery. In *Proceedings of 4th European Semantic Web Conference ESWC'07*, pages 145–159, 2007.

[19] A. O. Mendelzon, G. A. Mihaila, and T. Milo. Querying the world wide web. *Int. J. on Digital Libraries*, 1(1):54–67, 1997.

[20] E. Miller, R. Swick, and D. Brickley. Resource description framework RDF. W3C Recommendation, 2004.

[21] J. Perez, M. Arenas, and C. Gutierrez. Semantics and complexity of SPARQL. In *Proceedings of the 5th International Semantic Web Conference*, pages 30–43, 2006.

[22] E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. W3C Working draft, 2007.

[23] H. J. ter Horst. Extending the RDFS entailment lemma. In *Proceedings of the Third International Semantic web Conference (ISWC2004)*, pages 77–91, 2004.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399