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° ????

May 2007

Thème SYM

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

Constrained Regular Expressions in SPARQL

Faisal Alkhateeb , Jean-François Baget , Jérôme Euzenat

Thème SYM — Systèmes symboliques
Projet EXMO

Rapport de recherche n° ???? — May 2007 — 20 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 using only trains and buses?"), 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 the 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 particular case of CRDF entailment that can be computed using some kind 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, and query answering thus remains a PSPACE-complete problem for CSPARQL.

Key-words: semantic web, query language, RDF, SPARQL, constrained regular expressions.

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 en utilisant seulement les trains et les bus?"), 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, et le problème de répondre à la requête reste donc un PSPACE-complet pour CSPARQL.

Mots-clés : web sémantique, langage de requête, RDF, SPARQL, expressions régulières contraintes.

Contents

1 Introduction

RDF (Resource Description Framework, [22]) is a knowledge representation language dedicated to the annotation of documents and more generally of resources within the framework of 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 model-theoretic semantics [16], that defines the notion of consequence between two RDF graphs, *i.e.*, does an RDF graph G entails an RDF graph H (RDF ENTAILMENT), which is a NP-hard problem [14]. 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 [24] 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 GRDF graph allowing variables as predicates, as done in [17]). 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, [3] has proposed an extension of GRDF, that allows expressing paths of arbitrary length. This language, called PRDF (for Paths RDF), allows using regular expressions as predicates in an RDF triple. As done before in databases [9, 10, 11, 20, 1, 6], these regular expressions can encode paths in an RDF graph (the concatenation of the labels of arcs in this 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, and form a NP-hard problem. Using PRDF queries, we can ask questions of the form: "does there exist a trip from town A to town B using only trains and buses?".

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 CRDF (for Constrained paths RDF), that allows to express such constraints. As done for PRDF, we provide an abstract syntax for the language, extend model-theoretic semantics of RDF, and characterize entailment to define answers to a CRDF query in an RDF knowledge base. The problem remains NP-hard.

Finally, since both PRDF and CRDF queries answers are maps, we can use the SPARQL definition framework of [23] to extend SPARQL: by considering the graph patterns of SPARQL as PRDF or CRDF graphs, we obtain the PSPARQL and CSPARQL extensions of that query language. We have already implemented the PSPARQL and CSPARQL languages[1].

Paper outline. Sect. 2 is devoted to the presentation of the GRDF language. The presentation framework of the CRDF language is as follows: we first define the abstract syntax of the language in Sect. 3, then its model-theoretic semantics for CRDF, by extending the standard RDF/GRDF semantics Sect. 4. We characterize a particular case of entailment in the considered language, where the knowledge base is an RDF graph Sect. 5. In this case, query answering can be computed by

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

maps (a NP-hard problem). These maps are used in Sect. 6 to extend the SPARQL query language to CSPARQL. After a review of related works (Sect. 7), we conclude in Sect. 8.

2 GRDF

In this section, we present 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 [17]). 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 semantic rules to the knowledge base (a polynomial procedure). The same framework could easily be applied to PRDF or CRDF to extend, for example, our languages to PRDFS or CRDFS. 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.

The abstract syntax, model-theoretic semantics and a characterization of GRDF entailment as a particular map (a graph homomorphism [14, 5]) are presented successively.

2.1 GRDF syntax

Terminology. The *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 foaf:Person or ex:friend).

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. When switching to SPARQL, variables and blanks have different behaviors in complex cases. For example, a blank shared in different simple patterns of a group query pattern has a local scope which is easier to describe as changing the quantification scope of a variable than changing a blank into a variable. So, for the purpose of this paper and without loss of generality, we have chosen to follow [23] to not distinguish between variables and blanks, and use variables instead.

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 [18] 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 (RDF and GRDF graphs) *A* 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 an RDF or 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. 5, we extend these notations to also take into account the terms which appear in the (constrained) regular expression. 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.*, a triple (s, p, o), then $s \xrightarrow{p} o$). By drawing these graphs, the nodes resulting from literals are represented by rectangles while the others are represented as 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.

Example 1 *The RDF graph G, which is presented graphically in Fig. 1, represents the transportation means between cities, the type of the transportation mean, and the price of the tickets.*

2.2 GRDF semantics

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

Definition 2 (Interpretation of a vocabulary) *Let $V \subseteq \mathcal{V} = \mathcal{U} \cup \mathcal{L}$ 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 $\iota(v) = v$.*

Definition 3 (Model of a GRDF graph) *Let $V \cup \mathcal{V}$ be a vocabulary, and G be a GRDF graph such that every name appearing in G is also in V ($\mathcal{V}(G) \subseteq V$). 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 all kind of graphs (*i.e.*, RDF, GRDF, PRDF and CRDF) presented in this paper.

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

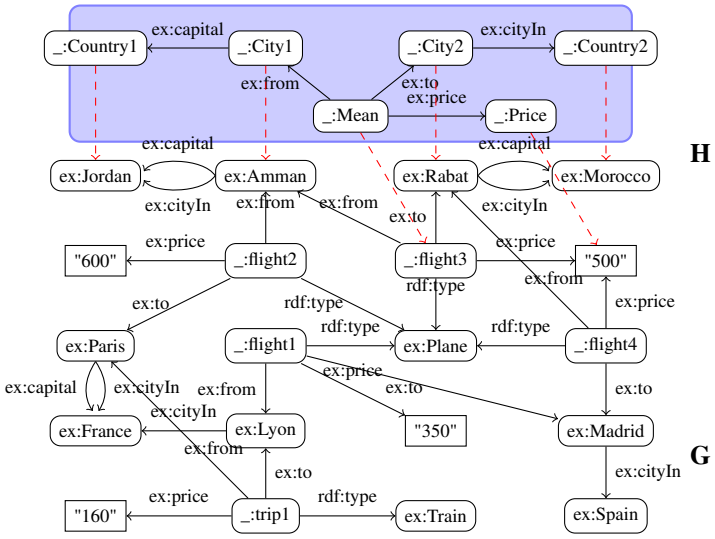

Figure 1: An RDF homomorphism.

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, noted $G \models_{RDF} G'$, iff every model of G is also a model of G'.*

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

PROOF (Satisfiability) [5] builds the isomorphic model of a GRDF graph. (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.

2.3 Inference mechanism for GRDF

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

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 (RDF homomorphisms) *Let G and G' be two GRDF graphs. An RDF* 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 $G \models_{RDF} G'$ iff there exists an RDF homomorphism from G' into G.*

The definition of RDF 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] also 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 RDF (Theorem 1) to the CRDF graphs studied in the rest of this paper.

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

This problem is called RDF ENTAILMENT, and it shown to be NP-complete for RDF graphs [16, 14]. For GRDF, the complexity remains unchanged. Polynomial subclasses of the problem can be exhibited based upon the structure or labeling of the query: when the query is ground [17], 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:Jordan), (_:Country2, ex:Morocco), (_:City1, ex:Amman), (_:City2, ex:Rabat), (_:Mean, _:flight3), (_:Price, "500")$\}$ *is an RDF homomorphism from RDF graph H into G of Fig. 1.*

3 CRDF: 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 parametrize the CRDF language in the way that allows us to naturally extend it to include more general constraints as done in Sect. 6.

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

For example, the constraint defined by $\langle]\forall$_:N$] : \{$(_:N, rdf:type, ex:Plane)$\}\rangle$ is a universal and open from the left hand-side.

In what follows, we use ϕ_{GRDF} to denote the set of constraints restricted to GRDF graphs.

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 noted $a_1 \cdot \ldots \cdot a_k$. If $A =$

$a_1 \cdot \ldots \cdot a_k$ et $B = b_1 \cdot \ldots \cdot b_q$ 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 $(\mathcal{U}, \mathcal{B}, \phi_{\text{GRDF}})$ can be used to define the language over $(\mathcal{U} \cup \mathcal{B})$.

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

- *if* $u \in \mathcal{U}$*, then* u*,* $(!u)$ *and* $(u^-) \in \mathcal{R}(\mathcal{U}, \mathcal{B}, \phi_{GRDF})$*;*
- *if* $b \in \mathcal{B}$*, then* $b \in \mathcal{R}(\mathcal{U}, \mathcal{B}, \phi_{GRDF})$*;*
- *if* $R \in R(\mathcal{U}, \mathcal{B}, \phi_{GRDF})$*, then* $(R^+) \in \mathcal{R}(\mathcal{U}, \mathcal{B}, \phi_{GRDF})$*;*
- *if* $R_1, R_2 \in \mathcal{R}(\mathcal{U}, \mathcal{B}, \phi_{GRDF})$*, then* $(R_1 \cdot R_2)$*, and* $(R_1|R_2)$ *are elements of* $\mathcal{R}(\mathcal{U}, \mathcal{B}, \phi_{GRDF})$*.*
- *(5) if* $R \in \mathcal{R}(\mathcal{U}, \mathcal{B}, \phi_{GRDF})$ *and* $\phi \in \phi_{GRDF}$ *is a GRDF constraint, then* $R\langle\phi\rangle \in \mathcal{R}(\mathcal{U}, \mathcal{B}, \phi_{GRDF})$*.*

Note that the negation ! and $^-$ are restricted to atomic urirefs to have a reasonable complexity for solving regular expression problems (one can chose to apply the negation to the outermost regular expression as done in [8]). The inverse operator $^-$ handles only atomic expressions. Its effect will be on the matching process (*i.e.*, to specify the orientation of arcs in the paths retrieved). For example, the triple following $(\texttt{ex:Amman}, \texttt{ex:from}^-, \texttt{\_:Flight})$ matches $(\texttt{\_:Flight}, \texttt{ex:from}, \texttt{ex:Amman})$. Moreover, The constraints are not necessarily grouped together and we can have a constrained regular expression of the form $R\langle\phi_1\rangle \ldots \langle\phi_k\rangle$. This way, we can 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 CRDF[ϕ_{GRDF}] graph is a graph whose arcs are labeled with constrained regular expressions whose constraints are elements of ϕ_{GRDF}.

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

Example 3 *Suppose we want to go from Jordan to Spain using only plane. This query is represented by the CRDF[*ϕ_{GRDF}*] graph* H *of Fig. 2.*

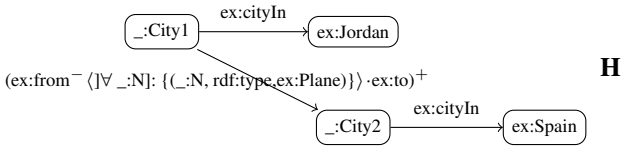

Figure 2: A CRDF graph.

4 CRDF: semantics

To be able to express the semantics of CRDF 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.

4.1 Generated language

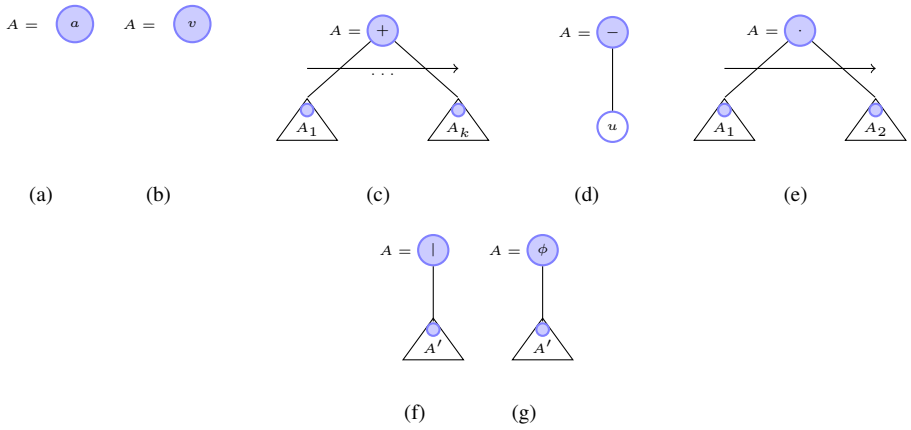

Figure 3: Constructing a derivation tree of a constrained regular expression.

Definition 10 (Derivation tree) *Let $R \in \mathcal{R}(\mathcal{U}, \mathcal{B})$ be a regular expression. A rooted labeled tree with ordered subtrees A is called a derivation tree of R (and we denote $A \in \mathcal{DT}(R)$) 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. 3(a);*
2. *if $R = !u$ and $v \in \mathcal{U}$ with $v \neq u$, then A is the tree of Fig. 3(b);*
3. *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. 3(c);*
4. *if $R = (u^{-})$, then A is the tree of Fig. 3(d);*
5. *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. 3(e);*
6. *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. 3(f);*

7. if $R = (R'[\phi])$ and $A' \in \mathcal{DT}(R')$, then A is the tree of Fig. 3(g).

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 the concatenating the labels of the leaves of A, totally ordered by the depth-first exploration of A determined by the ordering 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 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 because variables may match predicate variables in GRDF graphs.

4.2 Interpretations and models in CRDF

A CRDF interpretation of a vocabulary $V \subseteq \mathcal{V}$, is an RDF interpretation of V. However, an RDF interpretation has specific conditions to be a model for a CRDF 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 $R \in \mathcal{R}(\mathcal{U}, \mathcal{B}, \phi_{GRDF})$ be a 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 $^-$;*
- *$\langle r_{i-1}, r_i \rangle \in I_{EXT}(\iota'(a_i))$, otherwise.*

The first item of this definition handles the constructor ($^-$): 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 CRDF models in which it replaces the direct correspondences 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 expression with variable length paths.

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

Now we are ready to define when an interpretation is a model of a CRDF graph.

Definition 14 (Model of a CRDF graph) *Let $I = (I_R, I_P, I_{EXT}, \iota)$ be an interpretation of a vocabulary V, and G be a CRDF[ϕ_{GRDF}] 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 $T = (r_0, \ldots, r_n)$ of resources of I_R ($\iota'(s) = r_0$ 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';*
- *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 $Q\ r \in \dagger_1 r_{p-1}, \ldots, r_{p+q-1} \dagger_2$, r satisfies ϕ.*

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

PROOF (sketch). If G is a CRDF[ϕ_{GRDF}] graph such that $L^*(R) \neq \emptyset$ for all predicates R of the graph, then build a graph G' obtained by replacing all arcs r of G by a path encoding an arbitrary word of the language generated by the label of r, and adding all informations required to satisfy the constraints. See that any model of G' (their existence is proven by Prop. 1) is a model of G.

5 Inference mechanism for CRDF

Two conditions must be satisfied in the notion of homomorphism to be able to find the answers to a CRDF[ϕ_{GRDF}] 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 (Def. 15); and the satisfaction of the corresponding node(s) in the path of the knowledge base G to the graph C of the constraint (Def. 16).

Definition 15 (Path word) *Let G be an RDF graph of a vocabulary $V \cup \mathcal{V}$, and $R \in \mathcal{R}(\mathcal{U}, \mathcal{B})$ be a 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 $(x_0, \ldots, x_k)$ of nodes of G is called a* path *of w in G according to μ iff $\forall 1 \leq i \leq k$:*

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

Example 4 *For example, the tuple $T = (\texttt{ex:Amman}, \texttt{\_:flight2}, \texttt{ex:Rabat}, \texttt{\_:flight4}, \texttt{ex:Madrid})$ of nodes in the RDF graph G of Fig. 1 is a path of the word $w = (\texttt{ex:from}^- \cdot \texttt{ex:to} \cdot \texttt{ex:from}^- \cdot \texttt{ex:to}) \in L^*((\texttt{ex:from}^- \langle]\forall \texttt{\_:N}] : \{(\texttt{\_:N}, \texttt{rdf:type}, \texttt{ex:Plane})\}\rangle \cdot \texttt{ex:to})^+)$ according to the empty map.*

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.

Definition 16 (Satisfaction of a constraint in an RDF graph) *Let G be a graph, $\phi = \dagger_1 Qx \dagger_2 : C$ be a constraint, and s a term of G. Then s satisfies ϕ in G if there exists an RDF homomorphism π from C into G such that $\pi(x) = s$.*

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

Definition 17 (CRDF homomorphism) *Let G be an RDF graph and H be a CRDF[ϕ_{GRDF}] graph. A* CRDF 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 $T = (n_0, \ldots, n_m)$ of nodes of G ($\pi(s) = n_0$ and $\pi(o) = n_m$) 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 $\phi = \dagger_1 Qx \dagger_2 : C$ in A, rooting a subtree A' with $a_p \cdot \ldots \cdot a_{p+q} = w(A')$, then $Q\ n \in \dagger_1 n_{p-1}, \ldots, n_{p+q-1} \dagger_2$, n satisfies ϕ.*

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

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

See that the map from G' into G, along with all necessary homomorphisms from the constraints of G' into G is still a polynomial certificate.

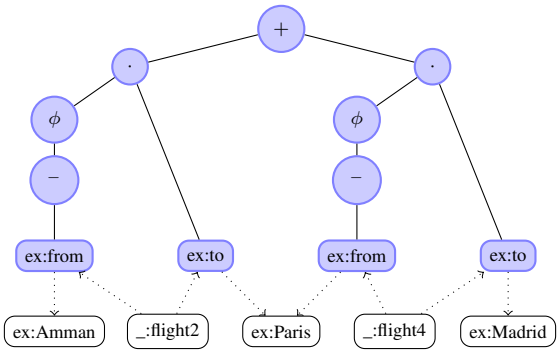

Figure 4: A possible derivation tree of the constrained regular expression of Ex. 5. The nodes in white color, which correspond to the path of nodes in the RDF graph G, together with the path labels are used to quantify the elements of the tree.

Example 5 *Consider the PRDF graph H of Fig. 2, the RDF graph G of Fig. 1. According to Def. 17, the first condition is satisfied (see Ex. 4), and the transportation means along the path are of type plane (see also Fig. 4). So the map π defined by* $\{(\texttt{\_:City1}, \texttt{ex:Amman}), (\texttt{ex:from}, \texttt{ex:from}), (\texttt{ex:to}, \texttt{ex:to}), (\texttt{ex:cityIn}, \texttt{ex:cityIn}), (\texttt{\_:City2}, \texttt{ex:Madrid}), (\texttt{\_:Country}, \texttt{ex:Spain})\}$ *is a CRDF homomorphism from H into G.*

6 CSPARQL

[23] presents an alternate characterization of query answering with the SPARQL query language that relies upon operations on maps from the GRDF graph pattern that form the heart of a query into an RDF knowledge base. We use this framework to extend SPARQL to CSPARQL, by defining graph patterns as CRDF graphs. This method allows us to combine the expressive power of (constrained) regular paths with the SPARQL query language.

6.1 Syntax

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 ... queries[3]. Our extension can then be applied to other query forms. So, we will not present the keywords which allow, for example, to filter (FILTER), to order (ORDER BY), or to limit (LIMIT and/or OFFSET) the answers of a query. The reader can refer to the SPARQL draft [24] or to [23] for formal semantics of SPARQL queries.

CSPARQL graph patterns are built on top of CRDF in the same way that SPARQL is built on top of RDF. In CSPARQL there are several functions that can be used for capturing the values along the paths like `SUM` for summation of values, `AVG` for the average, `COUNT` for counting nodes satisfying constraints. For the sake of simplicity, we have not introduced these function in the syntax of the language, and illustrate them with examples in Sect. 6.3.

Definition 18 (CSPARQL graph patterns) *A* CSPARQL graph pattern *is defined by:*

- *every CRDF[ϕ_{GRDF}] graph is a CSPARQL graph pattern;*
- *if P_1, P_2 are CSPARQL 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 CSPARQL graph patterns.*

Note 1 The parametrization of CRDF[ϕ_{GRDF}] by [ϕ_{GRDF}] allows us to extend naturally CSPARQL graph patterns to any set of constraints (for example, SPARQL or CSPARQL graph patterns), and the complexity of the language will be also parametrized by CRDF[ϕ_{GRDF}]. This way, if ϕ_{SPARQL} denotes the set of all possible SPARQL graph patterns, then the first item could be replaced by the following one: - every CRDF[ϕ_{SPARQL}] graph is a CSPARQL graph pattern.

[3] 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.

CSPARQL query. A *CSPARQL query* is of the form `SELECT` $\vec{B}$ `FROM` u `WHERE` P. The only difference with a SPARQL query is that, this time, P is a CSPARQL graph pattern.

6.2 Answers to CSPARQL 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 μ_1 and μ_2 are two compatible maps, then we use $\mu = \mu_1 \oplus \mu_2 : T_1 \cup T_2 \rightarrow \mathcal{T}$ 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 [23], we define the *join* and *difference* of two sets of maps Ω_1 and Ω_2 as follows:

- *(join)* $\Omega_1 \bowtie \Omega_2 = \{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1, \mu_2 \in \Omega_2$ are compatibles $\}$;
- *(difference)* $\Omega_1 \setminus \Omega_2 = \{\mu_1 \in \Omega_1 \mid \forall \mu_2 \in \Omega_2, \mu_1$ and μ_2 are not compatibles$\}$.

As in the case of RDF/GRDF, the answer to a query reduced to a CRDF[ϕ_{GRDF}] graph with constraints is also given by a map. The definition of an answer to a CSPARQL query will be thus identical to that given for SPARQL [23], but it will use CRDF homomorphisms.

Definition 19 (Answers to a CSPARQL graph pattern) *Let P be a CSPARQL graph pattern and G be an RDF graph. The set $\mathcal{S}(P, G)$ of answers of P in G is defined inductively by:*

- *if P is a CRDF[ϕ_{GRDF}] graph, $\mathcal{S}(P, G) = \{\mu \mid \mu$ is a CRDF 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 CSPARQL graph patterns are constructed over CRDF[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, $\phi = \dagger_1 Q x \dagger_2 : P$ a constraint, and s a term of G. Then s satisfies ϕ in G if there exists a map $\mu \in \mathcal{S}(P, G)$ such that $\mu(x) = s$. The definition of CRDF homomorphism (Def. 17) and first item of Def. 19 remain unchanged.

Let $Q =$`SELECT` $\vec{B}$ `FROM` u `WHERE` P be a CSPARQL 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 a GRDF graph, P be a PRDF graph and $\vec{B}$ be a tuple of variables appearing in P, an answer to the CSPARQL query Q =*SELECT $\vec{B}$ FROM u WHERE P *is a CRDF homomorphism μ such that $G \models_{CRDF} \mu(P)$.*

This property 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 CRDF homomorphisms from P into G which, by Theorem 2, corresponds to CRDF-GRDF entailment.

6.2.1 CSPARQL Complexity

Since CSPARQL queries are the same as SPARQL queries with the difference of the kind of basic graph patterns (*i.e.*, GRDF vs CRDF[ϕ_{GRDF}]) and CSPARQL QUERY EVALUATION for CRDF[ϕ_{GRDF}] graphs is in NP, our extension does not increase the worst case complexity of SPARQL, *i.e.*, PSPACE-complete [23].

6.3 CSPARQL by examples

The following examples attempts to give a feel of CSPARQL.

Example 6 *Suppose one wants to go from Paris to a city in Switzerland such that he cannot go out the union european, for example the visa problem. The following query finds the name of such city:*

SELECT _:City _:Country
FROM $<$ http : //example.org/index1.ttl $>$
WHERE {{(ex:Paris, ex:from$^-$⟨[∀_:Stop] : {(_:Stop, ex:cityIn, _:Country), (_:Country, ex:partOf, ex:Europe
ex:to$^+$), _:City), (_:City, ex:cityIn, ex:Switzerland)}}

Moreover, the price of each direct trip is no more than 500*:*

SELECT _:City _:Country
FROM $<$ http : //example.org/index1.ttl $>$
WHERE {{(ex:Amman, ex:from$^-$⟨]∀_:Trip] : (_:Trip, ex:price, _:Price) FILTER (_:Price $<$ 500)⟩)· ex:to$^+$), _:City)}}

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 _:Country
FROM $<$ http : //example.org/index1.ttl $>$
WHERE {(ex:Amman, ex:from$^-$⟨ SUM(_:Sum1, _:Price)[∀_:Trip] : (_:Trip, ex:price, _:Price)
FILTER (SUM(_:Sum1, _:Price) $<$ 1000)⟩)· ex:to$^+$, _:City)}}

7 Related Work

G and its extension G+ are two languages for querying structured databases. These query languages support only graphical queries similar to PRDF queries, which are limited to finding simple paths (cycle-free paths). They do not provide SQL-like functionalities, for example, for filtering, ordering,

projection, selection and other useful features. Graphlog — a visual query language which has been proven equivalent to linear Datalog [8] — extends G+ by combining it with the Datalog notation. Lorel [1] and UnQL [6] are two SQL-like languages for querying semi-structured documents, and use regular expression patterns to find simple paths. STRUQL [12], a query language for a web-site management system, incorporates regular expressions and has precisely the same expressive power as stratified linear Datalog. WebSQL [21], incorporates regular expressions for querying distributed collection of documents connected by hypertext links. It has a cost based query evaluation mechanism, *i.e.*, it evaluates how much of the network must be visited to answer a particular query. The main problems with finding only simple paths are that: the complexity of solving regular expressions is NP-complete if they do not contain variables [25]; and there are situations in which answers to such queries are all non simple, *e.g.*, if the only paths matching a regular expression pattern have cycles [4]. Moreover, the above languages do not support constraints in paths and/or internal nodes.

Two extensions of SPARQL, which are closely similar to PSPARQL, have been recently defined based on our initial proposal [2]: SPARQLeR [19] 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 given two 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 found path.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 several times 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.

CSPARQL generalizes all the above mentioned languages to allow inline qualifying regular expressions with variables, and is not restricted to simple paths. This relaxation not only useful for many applications (*cf.* [4] for some examples), but also provides polynomial classes of the satisfiability problem of regular expressions (*i.e.*, when they don not contain variables). The originality of our proposal lies in our adaptation of 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, that provides a wider range of querying paradigms than the above mentioned languages.

8 Conclusion

This paper attempts to enrich our initial proposal, PRDF, that extends RDF graphs with the ability of expressing paths. Since PRDF does not allow to specify characteristics of the nodes traversed by a regular path, we have extended PRDF to CRDF: Constrained RDF. We have extended the language semantics to handle constraints, and have characterized answers to a CRDF query as maps. This property was sufficient to extend the SPARQL query language to CSPARQL, combining this time

the expressiveness of both SPARQL and constrained regular expressions. We have provided a sound and complete inference mechanism for answering CSPARQL queries over RDF graphs as well as algorithms for calculating these answers. We have also proven that adding constraints to our initial proposal does not increase the computation properties of the language, *i.e.*, the problem of checking if there exists a homomorphism from a CRDF graph into an RDF graph is NP-complete, and thus the problem of evaluating CSPARQL queries over RDF graphs, remains PSPACE-complete. As it 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), provide more sophisticated, useful, and inline constraints (during the evaluation process).

Though our language is presented with GRDF (or SPARQL graph patterns) as constraints, it can be extended to include constraints that can be more general. As done for SPARQL, CRDF language can be adapted and integrated in other graph-based query languages. Extension of RDF to RDFS (RDF Schema) does not change the computational properties of the language: the consequence in RDFS is reduced polynomially to the consequence in RDF [16]. So, our work extends naturally to RDFS thanks to this reduction. Finally, we have implemented a CSPARQL query engine that is available for downloads, and provided an applet for 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, 05 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 *Proc. of the ISWC*, pages 82–96, 2005.

[6] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A query language and optimization techniques for unstructured data. In *Proc. 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. Recommendation, W3C, February 2004.

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

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

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

[11] O. de Moor and E. David. Universal regular path queries. *Higher-Order and Symbolic Computation*, 16(1-2):15–35, 2003.

[12] M. Fernandez, D. Florescu, A. Levy, and D. Suciu. A query language and processor for a website management system. In *Proceedings Workshop on Management of Semistructured Data*, Tucson, 1997.

[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 *Proc. of the ISWC*, pages 502–517, 2004.

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

[17] H. J. Horst. Extending the $RDFS$ entailment lemma. In *Proc. of the ISWC*, pages 77–91, 2004.

[18] 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.

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

[20] Y. A. Liu, T. Rothamel, F. Yu, S. Stoller, and N. Hu. Parametric regular path queries. In *Proc. of the ACM SIGPLAN*, pages 219–230, 2004.

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

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

[23] J. Perez, M. Arenas, and C. Gutierrez. Semantics and complexity of SPARQL. In *Proc. of the ISWC*, pages 30–43, Athens (GA US), 2006.

[24] E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. Working draft, W3C, 2006.

[25] P. T. Wood. *Queries on Graphs*. PhD thesis, Department of Computer Science, University of Toronto, 1988.

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