# Quick introduction to RDFPath

Stefan Kokkelink

WORKING DRAFT

THIS VERSION:

March 15, 2001

The purpose of RDFPath is to localize information in an RDF graph. It provides for a general technique to specify paths between two arbitrary nodes of an RDF graph. An RDFPath expression basically is build upon the following three constructs:

1. A *primary selection* selects an initial set of nodes of an given RDF graph. For example, the query `resource('http://mydomain.org')` selects the resource with URI `http://mydomain.org`. In most cases the returned objects of a primary selection serve as context objects for further queries.

2. A *location step* specifies a set of nodes which can be reached by 'one step' from a given context object. A composition of location steps (seperated by the character `/`) defines a *location path*. For example, the query `child(dc:creator)/child(vCard:FN)` selects all nodes that could be reached by a composition of `dc:creator` and `vCard:FN` arcs starting at the current context (which should be a resource in this case).

3. A *filter* selects a subset of a given set of objects. For example the filter `[child(dc:creator) = "Karl Mustermann"]` selects all resources of a given set of nodes that have a `dc:creator` child whose string representation is equal to `Karl Mustermann`.

There are different types of possible parameters for primary selections and location steps. An URI can be expressed by a qualified name (e.g. `dc:creator`) or in absolute form within quotation marks (e.g. `'http://www.myorg.org'`). In the former case the implementation of RDFPath has to take care for a correct namespace binding. In general an RDFPath expression is a composition of a primary selection and several location steps and filters.

```
pselec/locstep1[filter1]/locstep2/locstep3[filter2][filter3]
```

An RDFPath query is the evaluation of an RDFPath expression relative to a given context object. The following list gives an overview over some available primary selections and location steps.

Primary selections:

- `resource()`. Selects all resources of an RDF graph. If a URI is given as a parameter (e.g. `resource(rdf:Bag)`) *the* resource named by the URI is returned.
- `literal()`. Selects all literals of an RDF graph. If a string is given as a parameter (e.g. `literal('Stefan')`) *the* literal with the given value is returned.

Locations steps:

- `child()`. Selects the childs of the context node. The childs are the target nodes of all arcs starting at the current context. If a URI is given as a parameter (e.g. `child(dc:creator)`) only those childs are returned which can be reached by a `dc:creator` arc.
- `parent()`. Selects the parents of the context node. The parents are the source nodes of all arcs ending at the current context. If a URI is given as a parameter (e.g. `parent(rdf:type)`) only those parents are returned which can be reached by an `rdf:type` arc.
- `element()`. Selects the childs of the context node that can be reached by an `rdf:_i` arc ($i \in \mathbb{N}$).
- `container()`. Selects the parents of the context node that have the current context node as target of an `rdf:_`$i$ arc ($i \in \mathbb{N}$).
- `self()`. Selects the current context node.
- `property()`. Selects the labels (as strings) of arcs starting at the current context node.

There are several further parameters available for the described location steps which refine their semantics, see [Kokkelink 01] for details. Every primary selection or location step selects a set of objects relative to the current context. In RDFPath *filters* are used to refine these sets of objects. For example, the filter `[child(dc:creator)]` selects all the nodes of the current object set that are source of a `dc:creator` arc. There are several predicates available, e.g. the equality of strings. Every object in RDFPath is supposed to have a string representation. For example, the RDFPath query `resource()[child(dc:creator)='Karl Mustermann']` selects all resources that have a `dc:creator` child whose string representation

equals `Karl Mustermann`.

*(ToDo: Explain 'powers': rdf:value{2}=rdf:value/rdf:value , rdf:value{*}=descendants , rdf:value{2-3} = ranges , rdf:value{2-*} )* !

The RDF Schema specification [Brickley & Guha 00] defines several axioms for statements that are *implicitly* contained in an RDF graph. For example, if `A` is a subclass of `B` and `a` is of type `A`, then `a` implicitly is also of type `B`:

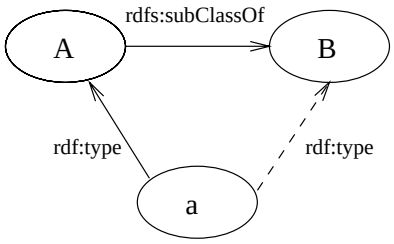


Figure 1: Implicit statements

RDFPath supports these kind of axioms by using the *extension operator* `^`. For example, the query `resource()[child(^rdf:type)=rdf:Bag]` selects all instances (explicit or implicit) of `rdf:Bag`.

In order to support rapid query creation RDFPath provides for a compact abbreviated syntax. The follwing table lists the main abbreviations:

| Standard | Abbreviation |
|---|---|
| `child()` | `*` |
| `child(dc:creator)` | `dc:creator` |
| `self()` | `.` |

Examples:

- `dc:creator`
- `dc:creator/vCard:FN`
- `dc:creator[vCard:FN='Karl Mustermann']`
- `dc:creator[vCard:FN='Karl Mustermann']/vCard:EMAIL`

*(ToDo: Provide some examples!)* !

## References

**Lassila, Ora; Swick, Ralph R.** (1999). *Resource Description Framework (RDF) Model and Syntax Specification.* W3C Recommendation 22 February 1999, `http://www.w3.org/TR/REC-rdf-syntax/`

**Brickeley, Dan; Guha, R.V.** (2000). *Resource Description Framework (RDF) Schema Specification 1.0.* W3C Candidate Recommendation 27 March 2000, `http://www.w3.org/TR/rdf-schema/`

**Kokkelink, Stefan** (2001). *RDFPath: A path language for RDF.* In preparation.