shax – a quickstart

Version: 2018-02-08

shax is a command-line tool for processing and generating SHAX models – for example translating a SHAX model into a SHACl model, or generating a SHAX model from XSDs. For related general information see the paper "Combining graph and tree: writing SHAX, obtaining SHACL, XSD and more" (doc/shax.pdf).

The tool is written in XQuery, version 3.1. For the time being, please use the XQuery processor BaseX for executing shax; BaseX can be downloaded from here:

http://basex.org/products/download/all-downloads/

Comprehensive documentation of shax is under construction. The quickstart document serves the sole purpose of enabling a quick trying out of some core functionality:

- Transformation: SHAX => SHACL
- Transformation: SHAX => XSD
- *Transformation: SHAX => JSON Schema (release planned for 2018-02-28)*
- Transformation: XSD => SHAX

Prerequisites

In order to try out shax, please download or clone the repository. The XQuery command-line tool is provided by `$SHAX_HOME/bin/shax.xq`

If you have not yet installed the BaseX XQuery processor, do so now.

General command-line syntax

The XQuery command-line tools is provided by `$SHAX_HOME/bin/shax.xq`. Using the BaseX XQuery processor, a call of `shax.xq` has the following general structure:

```
basex -b "request=operation?param1=…,param2=…"   $SHAX_HOME/bin/shax.xq
```

where

- `operation` identifies the tool operation (e.g. `shacl`, `xsd`, `jschema`, `xsd2shax`)
- `param1`, `param2`, … are operation-specific parameter names (e.g. shax, xsd)

Operation and parameter names can be abbreviated, as long as the prefix used is long enough to identify exactly one operation or parameter unambiguously. Operation and parameter names can be entered in any mixture of upper and lower case.

While the set of available parameters is in principle operation specific, SHAX models to be processed are always specified by a parameter named `shax`. The parameter type is `docFOX`, which means that the value can be a FOXpath expression identifying one or more SHAX documents. See section "Paramter type `docFOX`" for details about paramter syntax and semantics. Similarly, XSD documents to be processed are always specified by a parameter named `xsd`, which has parameter type `docFOX`.

Some parameters have the type `nameFilter` – e.g. the parameter `ename`. See section “Parameter type `nameFilter`” for details about syntax and semantics.

Parameters typed `xs:boolean` can be specified using an abbreviated syntax (see section “Boolean parameters”).

Parameter types

Parameter type `docFOX`

The parameter value is a FOXpath expression. Such expressions can select file system resources with an XPath-like syntax. A general introduction to the FOXpath language is given in doc/foxpath-intro.pdf. Examples:

/shax-base/ota/OpenTravel_2013A_XML/OTA_AirAvailRQ.shax
/shax-base /ota//OTA_AirAvailRQ.shax
/shax-base /ota//*airavail*.shax
/shax-base /ota//(*avail* except *pkg*)
/shax-base /ota//*avail*[not(self~::*pkg*)]

Parameter type `nameFilter`

The parameter value is a whitespace-separated sequence of name patterns. A *name pattern* is a string of literal characters and/or wildcards. There are two kinds of wildcards: “*” matches a sequence of zero or more arbitrary chacters, and “?” matches exactly one arbitrary character. If the name pattern is not preceded by “~”, the pattern selects all names matching the pattern; if the pattern is preceded by “~”, the pattern is “negative”, that is, it selects all names *not* matching the name pattern.

A namePattern can have a postfix modifying the evaluation of the pattern:
Postfix: #s – case sensitive (default is case insensitive)
Postfix: #r – the pattern is interpreted as a regular expression, rather than a pattern

Examples:

- “*” - matches any name
- “*rq” – matches any name ending with “rq”
- “~*test*” – matches any name not containing the string “test”
- “*rq ~*test*” – matches any name ending with “rq” and not containing the string “test”
- “*RQ#c” – matches any name ending with “RQ”, case sensitively
- “*\d{3}#r” – matches any name ending with three digits

Boolean parameters

For parameters types `xs:boolean`, an abbreviated syntax is available: `paramname` is short for `paramname=true`; `~paramname` is short for `paramname=false`.

Translate SHAX into SHACL

Summary

Transforms one or more SHAX documents into a single SHACL document. The SHACL model is written using Turtle syntax.

Usage

Operation: shacl

Parameters:

- shax – a FOXpath expression specifying one or more SHAX documents; note that imports are resolved recursively, so that it suffices to provide the "head" model(s)

Examples

```
basex  -b "request=shacl?shax=/shax-base/ota//*airavail*"
          /tt/shax/bin/shax.xq     > airAvailability-shacl.ttl

basex  -b "request=shacl?s=/shax-base/ota/(*airavail* *ground*)"
          /tt/shax/bin/shax.xq     > airAvailability-shacl.ttl
```

Concepts

SHACL is a schema language for RDF (https://www.w3.org/TR/shacl/). It is one of the schema languagues into which SHAX models can be translated, the others being XSD and JSON Schema.

You can test the validation of RDF data against a SHACL model here: http://shacl.org/playground

Translate SHAX into XSD

Summary

Transforms one or more SHAX documents into XSD documents. Per namespace used in the SHAX model(s), one XSD is generated which uses the namespace as target namespace.

Usage

Operation: xsd

Parameters:

- shax – a FOXpath expression specifying one or more SHAX documents; note that imports are resolved recursively, so that it suffices to provide the "head" model(s)

- odir – an output folder into which to write the XSDs; this parameter MUST be used if the SHAX model(s) use more than one namespace, as in this case several XSDs are generated
- ofile – file name to be used for the generated XSD; the value "#stdout" means that the XSD should be written to standardout, rather than to a file. Note that in the case that several XSDs are written, the file names are constructed from the parameter values of $ofile and $osuffixes.
- osuffixes – name suffixes used if several XSDs are generated, which is the case when the SHAX model(s) use more than one namespace. In this case, the names of the XSDs are constructed by appending to $ofile (after stripping the file extension) a suffix and the extension ".xsd", where the suffix depends on the position of the target namespace in the list of sorted target namespaces. If the parameter is not specified, the suffixes are "1", "2", … If the parameter is specified, it is split at whitespaces and the resulting tokens are used as the first, second, … suffix. Example 1: ofile="foo.xsd", parameter osuffixes not used, two XSDs are generated; there names are "foo1.xsd" and "foo2.xsd". Example 2: xsd="foo.xsd" osuffixes="-bar -wow", two XSDs are generated; there names are "foo-bar.xsd" and "foo-wow.xsd".

Examples

```
basex  -b "request=xsd?shax=/shax-base/booking.shax, ofile=booking.xsd"
           /tt/shax/bin/shax.xq

basex  -b "request=xsd?shax=/shax-base/booking.shax, ofile=#stdout"
           /tt/shax/bin/shax.xq     > booking.xsd

basex  -b "request=xsd?s=/shax-base/booking.shax" ofile="travels.xsd" odir="/xsd"
           /tt/shax/bin/shax.xq

basex  -b "request=xsd?s=/shax-base/booking.shax" ofile="travels.xsd" odir="/xsd"
                          osuffixes="-core –suppl"
           /tt/shax/bin/shax.xq
```

Concepts

XSD is one of the schema languagues into which SHAX models can be translated, the others being SHACL and JSON Schema.

The XML data described and validated by the XSD are equivalent to the RDF data described and validated by the SHACL model generated from the SHAX model(s).

Translate SHAX into JSON Schema

Summary

Transforms one or more SHAX documents into a JSON Schema document.

Usage

Operation: jschema

Parameters:

- shax – a FOXpath expression specifying one or more SHAX documents; note that imports are resolved recursively, so that it suffices to provide the “head” model(s)
- oname – a name filter identifying the name(s) of the SHAX object types to be described by the JSON Schema

Example

```
basex  -b "request=jschema?shax=/shax-base/booking.shax, oname=booking"
           /tt/shax/bin/shax.xq     > booking.jschema.json

basex  -b "request=jschema?shax=/shax-base/(booking.shax,traveller.shax), oname=booking"
           /tt/shax/bin/shax.xq     > booking.jschema.json
```

Concepts

JSON Schema is one of the schema languagues into which SHAX models can be translated, the others being SHACL and XSD.

The JSON data described and validated by the JSON Schema are equivalent to the RDF data described and validated by the SHACL model generated from the SHAX model(s).

Translate XSD into SHAX

Summary

Transforms one or more XSD documents into a single SHAX document.

Usage

Operation: xsd2shax

Parameters:

- xsd – a FOXpath expression specifying one or more XSD documents; note that includes and imports are resolved recursively, so that it suffices to provide the “head” schema(s)

Examples

```
basex  -b "request=xsd2shacl?xsd=/xsd-base/ota//*airavail*"
           /tt/shax/bin/shax.xq     > travelAvailability.shax
```

```
basex  -b "request=xsd2s?x=/xsd-base/ota/(*airavail* *ground*)"
             /tt/shax/bin/shax.xq     > travelAvailability.shax
```

Concepts

XSD is translated into SHAX which captures approximately the XSD model content. Note that a precise specification of what is captured how is work in progress.