Mapping YANG to DSDL draft-ietf-netmod-dsdl-map-01

Ladislav Lhotka
⟨*lhotka@cesnet.cz*⟩

Rohan Mahy
⟨*rohan@ekabal.com*⟩

Sharon Chisholm
⟨*schishol@nortel.com*⟩

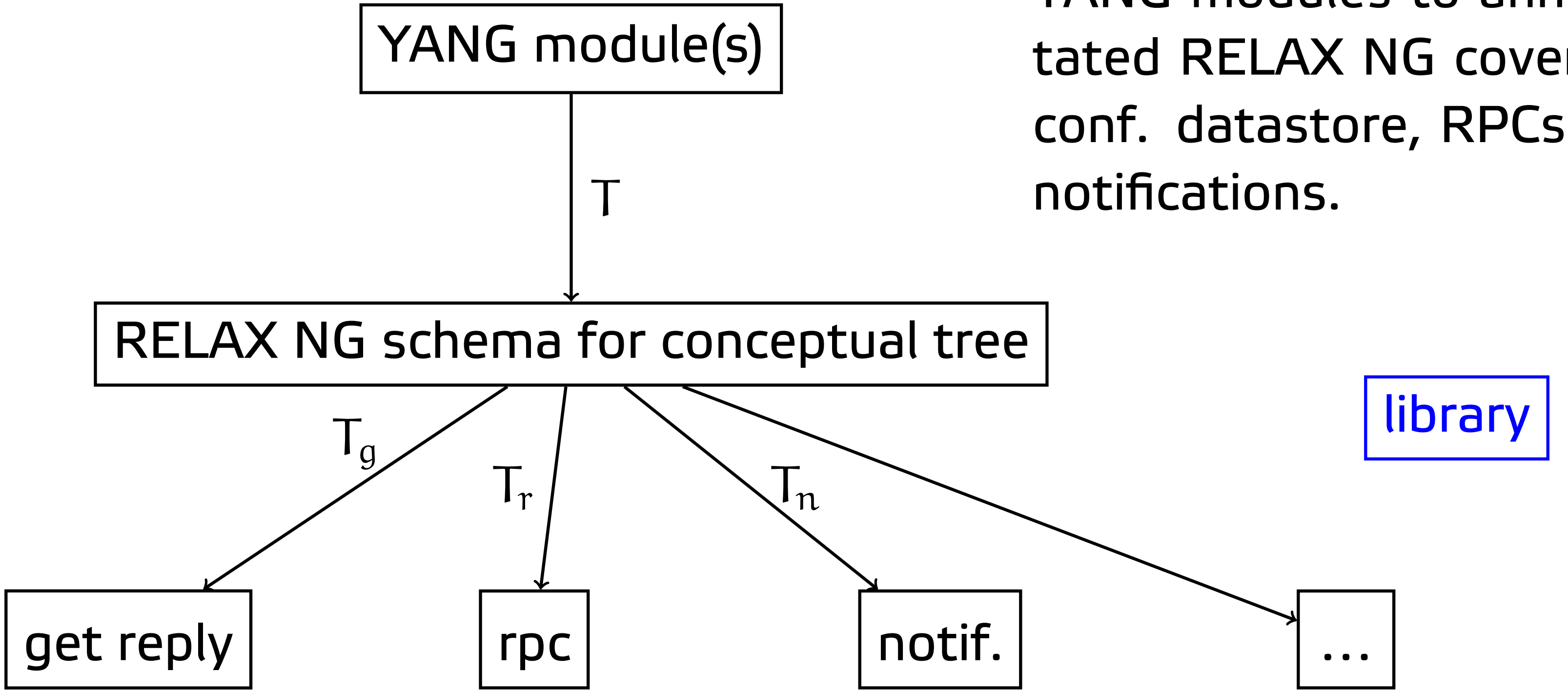
23 March 2009

Main changes between -00 and -01

- New text describing the second mapping step (conceptual tree schema $\rightarrow$ DSDL schemas)
- A complete DHCP example in Appendix C.
- `xmlns:xxx` is now the only way for declaring target namespaces in RELAX NG
- Additional underline character prepended to mangled names of groupings (typedef and groupings used separate namespaces in YANG)
- Names of list keys and all components of node identifiers in **unique** have an explicit NS prefix

Two-step procedure

Step #1 maps one or more YANG modules to annotated RELAX NG covering conf. datastore, RPCs and notifications.



Step #2 produces standard DSDL schemas (RELAX NG, Schematron, DSRL) for the requested XML document type and context (features, datastore), including the NETCONF envelope (e.g., `<rpc-reply>`).

Conceptual tree is just a way for structuring the RELAX NG schema so that the contents of input YANG modules are represented in a single schema.

Conceptual tree schema is still quite readable.

Description of the mapping is considerably easier.

However, implementations needn’t internally follow this structuring.

Conceptual tree schema

RELAX NG with three types of annotations:

- Dublin Core for metadata (references to input YANG modules)

  ```
  <dc:source>YANG module 'dhcp'</dc:source>
  ```

- RELAX NG DTD Compatibility for documentation

  ```
  <a:documentation>
    A reusable list of subnets
  </a:documentation>
  ```

- NETMOD-specific annotations – XML attributes and elements corresponding to YANG statements that cannot be represented in RELAX NG.

YANG language extensions (declared via **extension**) MAY be inserted in the schema, too, in YIN format.

NETMOD-specific annotations

Namespace URI:
`urn:ietf:params:xml:ns:netmod:dsdl-annotations:1`

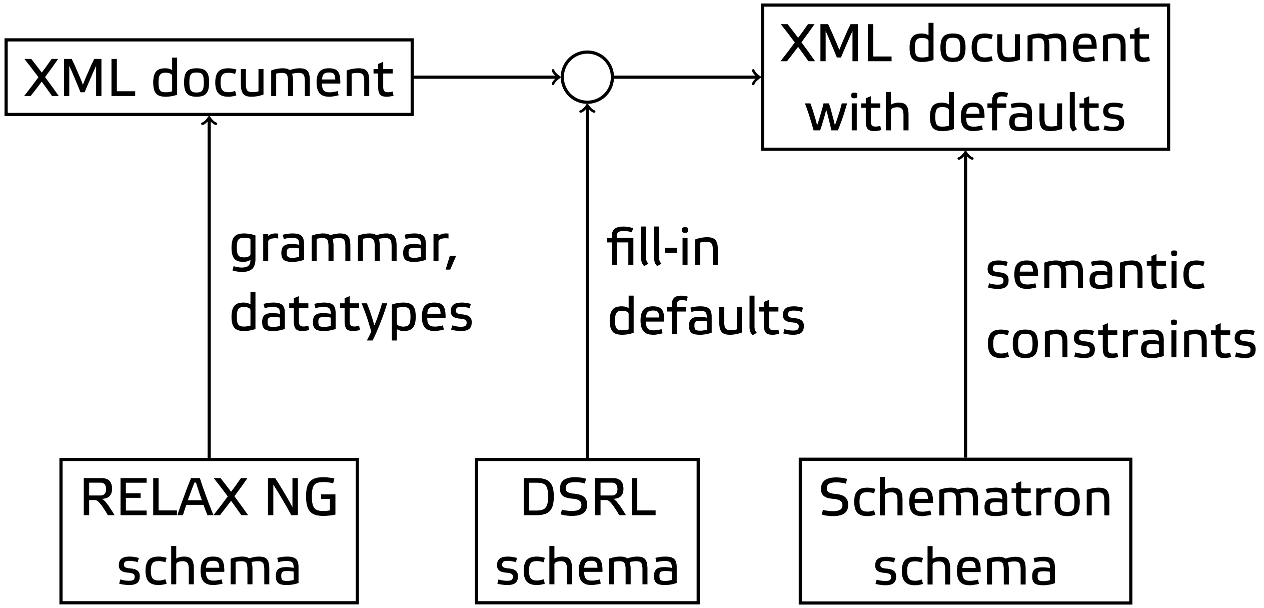
Attributes	*Elements*
`config`	`must`
`default`	`error-app-tag`
`default-case`	`error-message`
`key`	`instance-identifier`
`min-elements`	`leafref`
`max-elements`	
`ordered-by`	
`status`	
`unique`	
`units`	
`when`	

Second mapping step

Conceptual tree schema is transformed to standard RELAX NG, Schematron and DSRL.

- `config` annotations is used for filtering non-config (status) data nodes when the target document type is *get-config* reply;
- `default` and `default-case` annotations are mapped to DSRL element maps;
- `key`, `min-elements`, `max-elements`, `unique`, `when`, `must` & `error-message`, `instance-identifier` and `leafref` are mapped to Schematron rules.
- `ordered-by`, `status`, `units` and `error-app-tag` are not used.

Validation procedure



Schematron relies on grammatic validity (RELAX NG, then Schematron).

Defaults have to be substituted for missing leaf values *before* Schematron validation because of the way how YANG defines the context for evaluating XPath in **must**, **when**, etc.

Schematron phases

Phases are used for specifying subsets of Schematron rules that can be applied selectively.

The draft currently defines two phases:

full	(default) all rules are checked;
noref	referential integrity is not checked – useful e.g., for validating *candidate* datastore;

Phases will also be used for handling **if-feature**.

Current status

- Step 1 (YANG $\rightarrow$ conceptual tree schema) complete except **deviation** and **if-feature**
- Step 2 (validation schemas) complete except mapping defaults to DSRL and *instance-identifier* annotation.

Implemented in *pyang*: step 1 in Python, step 2 in XSLT.

RELAX NG validation tested with *Jing*, *libxml2*, *nxml-mode*.

Schematron validation tested with the official Schematron implementation from *http://www.schematron.com* (XSLT 1.0 distribution, used with *xsltproc*).

Independent testing with other tools is very welcome.

Open issue #1: if-feature

Another annotation in the conceptual tree schema.

Available features supplied as arguments for the second step.

- RELAX NG: elements depending on a feature delared as optional
- Schematron: extra pattern for each feature, validation phases have to be set up for all combination of features.
- DSRL: ??? - defaults mustn't be substituted for nonexistent leafs

Open issue #2: deviation

Proposal: modules specifying deviations are submitted as input modules for step 1 so that the deviations are already applied in the conceptual schema tree.

Open issue #3: instance-identifier type

Checking the presence of the instance is a task for Schematron but it (probably) cannot be implemented if XSLT 1.0 is used as the query language. Options: EXSLT or XPath 2.0

Proposal: EXSLT

Open issue #4: other targets for validation

The following XML instance document types can be validated:

- get/get-config replies
- specific RPC – request or reply
- specific notification
- combinations of the above – server and client part separately

Are there any other document types to be validated or other applications for the DSDL schemas?