Working with YANG Data Models and Instances Using (Mainly) *pyang*

Ladislav Lhotka
⟨lhotka@nic.cz⟩

20 July 2014

Agenda

- Required software,
- Editing YANG modules,
- *pyang* plugins,
- Preparing a sample instance document,
- DSDL-based validation of instance documents,
- Converting XML instance documents to JSON.

An extended version of this tutorial is available at
https://code.google.com/p/pyang/wiki/Tutorial

Required Software

- *pyang*

  https://code.google.com/p/pyang/

- Libxml2 tools (*xmllint*, *xsltproc*). Packages available for most operating systems and distributions.

  http://www.xmlsoft.org/

Optional:

- *Jing* and *Trang*

  https://code.google.com/p/jing-trang/

- *GNU Emacs* or *Aquamacs*

About *pyang*

Command-line tool written in Python, XSLT and sh/bash.

Extensible via plugins.

Project site: *https://code.google.com/p/pyang/*

Under active development, new plugins and bugfixes only available in SVN.

Last stable version: 1.4.1 (2013-11-11).

RTFM: Unix man pages

- pyang (1)
- yang2dsdl (1)

Editing YANG Modules

Commercial editors and development environments exist but standard editors mostly suffice.

Special support for popular editors:

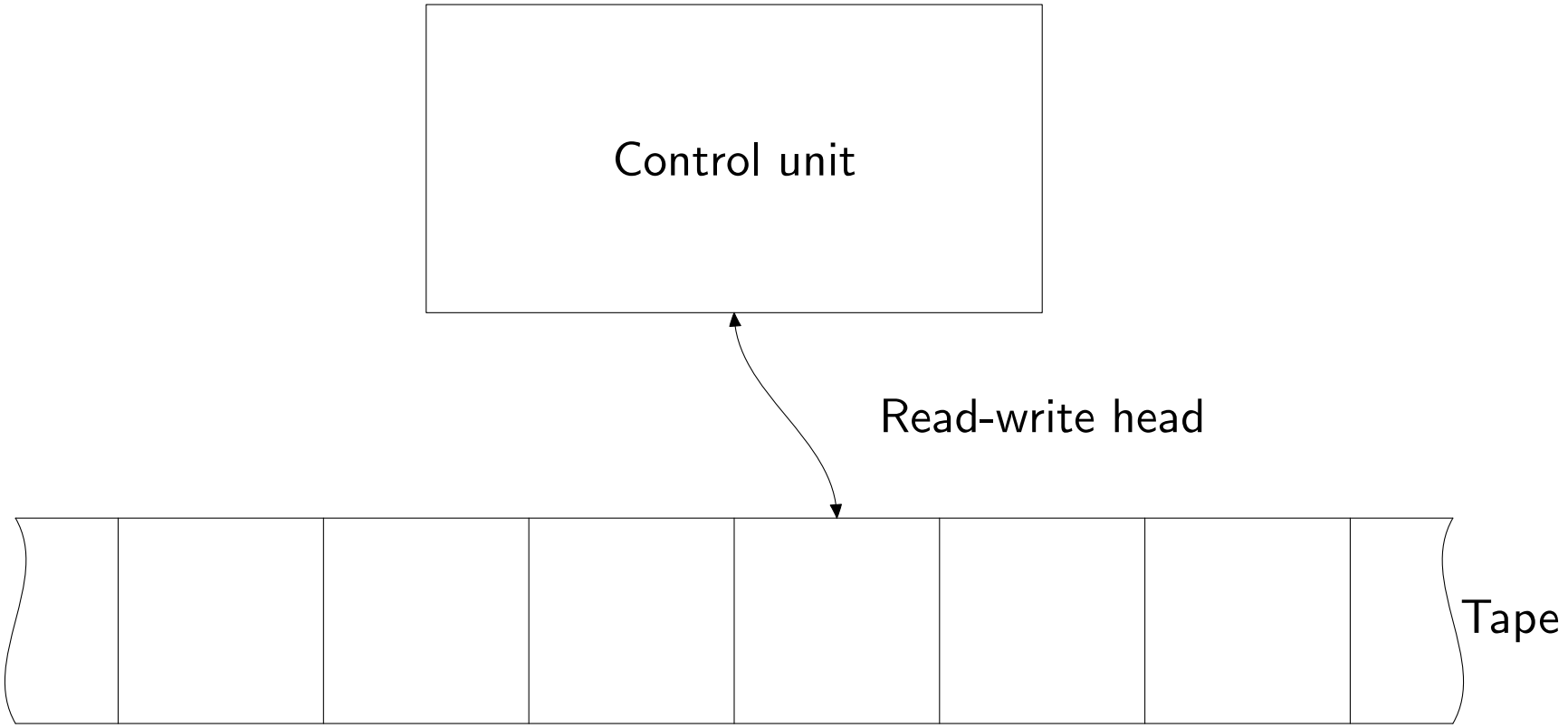
http://www.yang-central.org/twiki/bin/view/Main/YangTools

- *Emacs* – yang-mode
- *Vim* syntax file

With Emacs and nXML mode, it is also quite effective to use YIN syntax as the source format, see

https://gitlab.labs.nic.cz/labs/yang-tools/wikis/editing_yang

Turing Machine



YANG module: `turing-machine.yang`

Checking Module Correctness

```
$ pyang turing-machine.yang
```

Validation according to RFC 6087 rules:

```
$ pyang --ietf turing-machine.yang
```

Plugins

Conversions to various formats, activated with `-f`.

Most plugins have specific command-line switches and arguments.

- `yin`, `yang` – YIN and YANG syntax
- `dsdl` – DSDL hybrid schema (RFC 6110)
- `xsd` – W3C XML Schema (incomplete, deprecated)
- `tree` – schema tree (ASCII art)
- `xmi`, `uml` – UML diagrams
- `jstree` – HTML/JavaScript YANG browser
- `hypertree` – Hyperbolic YANG browser, to be used with Treebolic
- `jsonxsl`, `jtox` – XML↔JSON instance document conversion
- `sample-skeleton` – skeleton of a sample instance document

TM Schema Tree

```
$ pyang -f tree turing-machine.yang
```

Help on tree symbols:

```
$ pyang --tree-help
```

state data

configuration

RPC

notification

```
module: turing-machine
   +--rw turing-machine
      +--ro state                    state-index
      +--ro head-position            cell-index
      +--ro tape
      |  +--ro cell* [coord]
      |     +--ro coord     cell-index
      |     +--ro symbol?   tape-symbol
      +--rw transition-function
         +--rw delta* [label]
            +--rw label     string
            +--rw input
            |  +--rw state     state-index
            |  +--rw symbol    tape-symbol
            +--rw output
               +--rw state?       state-index
               +--rw symbol?      tape-symbol
               +--rw head-move?   head-dir
rpcs:
   +---x initialize
   |  +--ro input
   |     +--ro tape-content?   string
   +---x run
notifications:
   +---n halted
      +--ro state    state-index
```

leaf type

list key

optional node

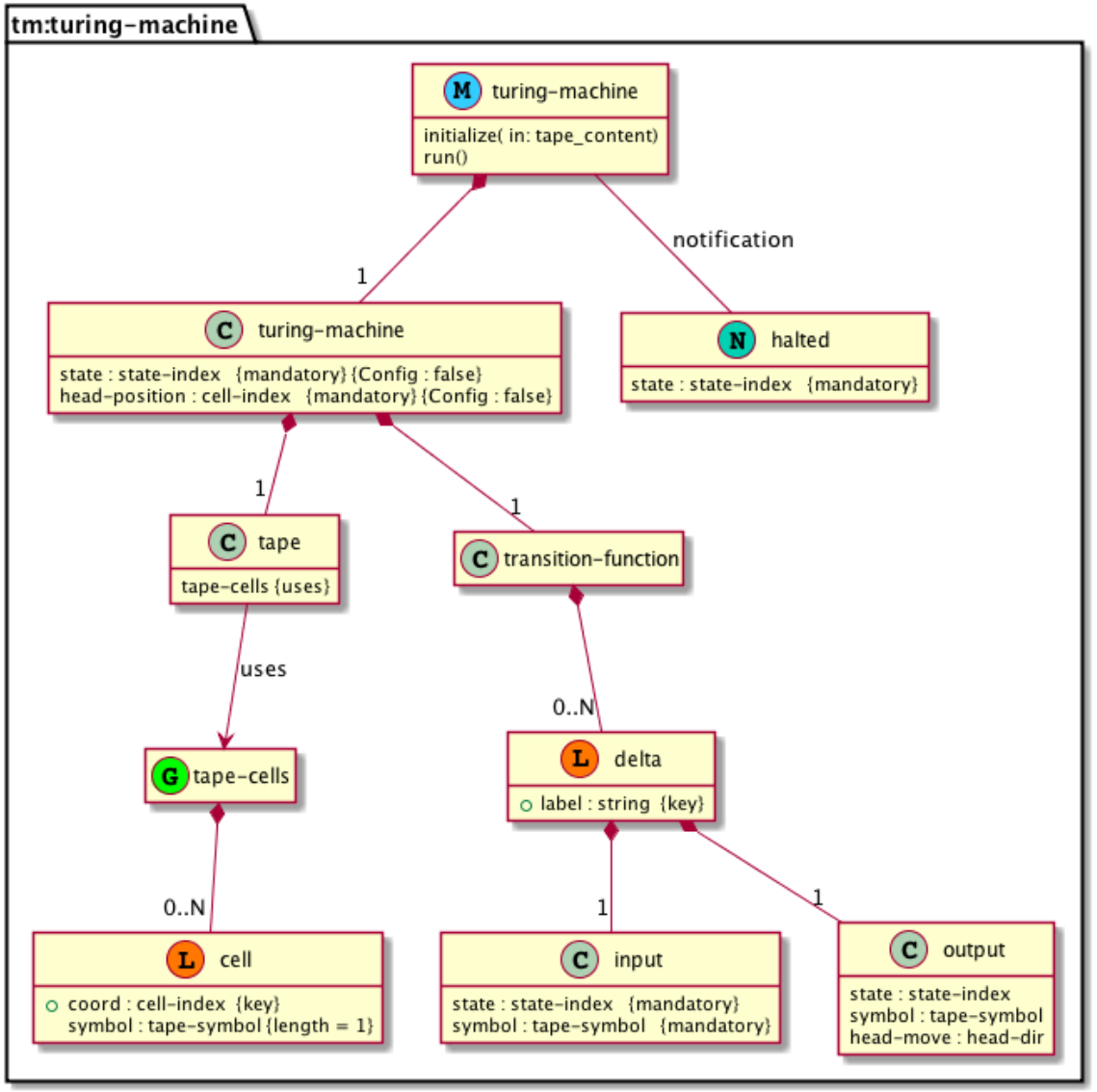
UML Diagram

```
$ pyang -f uml -o tm.uml turing-machine.yang \
> --uml-no=stereotypes,annotation,typedef
```

Conversion to PNG:

```
$ plantuml tm.uml
```

turing-machine



UML Generated : 2014-07-18 12:50

DSDL Schemas

DSDL = Document Schema Definition Languages

International Standard ISO/IEC 19757, see *http://dsdl.org*.

RFC 6110 defines the mapping of YANG data models to three schemas of the DSDL family:

- RELAX NG – schema (grammar) and types
- Schematron – semantic constraints
- DSRL (Document Schema Renaming Language) – defaults

```
$ yang2dsdl -t config turing-machine.yang
== Generating RELAX NG schema './turing-machine-config.rng'
Done.
== Generating Schematron schema './turing-machine-config.sch'
Done.
== Generating DSRL schema './turing-machine-config.dsrl'
Done.
```

Target for DSDL Schemas

DSDL schemas can be generated for different target document types selected by the `-t` option:

- `data` – configuration&state data, encapsulated in `<nc:data>` (default).
- `config` – configuration data, encapsulated in `<nc:config>`
- `get-reply` – complete reply to NETCONF *get* operation,
- `get-config-reply` – reply to *get-config* operation,
- `edit-config` – *edit-config* message,
- `rpc` – RPC request defined in the data model,
- `rpc-reply` – RPC reply defined in the data model,
- `notification` – event notification defined in the data model.

Preparing Sample XML Instance Document

In an I-D describing a data model, it is often useful to include a sample document showing instance data such as the contents of a configuration datastore.

1. Generate a skeleton document:

   ```
   $ pyang -f sample-skeleton turing-machine.yang \
   > --sample-skeleton-annotations --sample-skeleton-doctype=config | \
   > xmllint -o turing-machine-config.xml --format -
   ```

   The skeleton document has to be edited!

2. Convert the RELAX NG schema to the compact syntax:

   ```
   $ trang -I rng -O rnc turing-machine-config.rng turing-machine-config.rnc
   ```

3. Load `turing-machine-config.xml` into *Emacs*.

Schema-based Validation

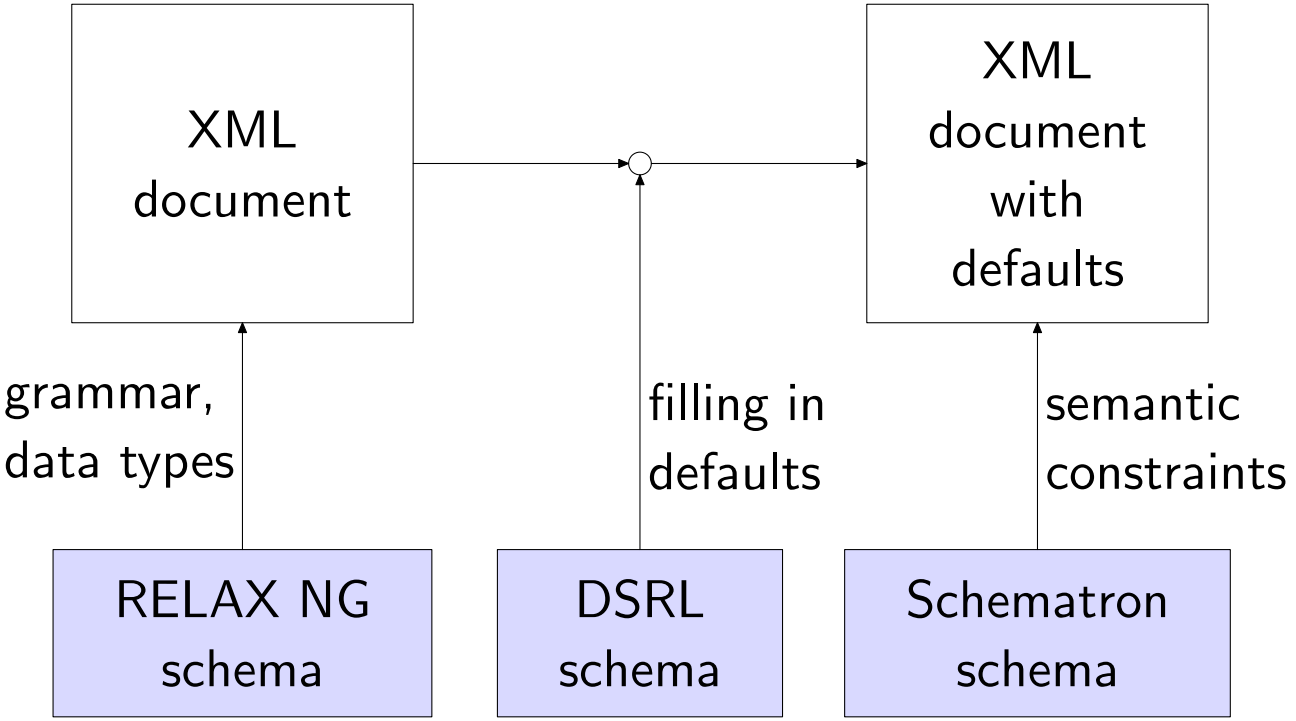
use pre-generated schemas schema name base

use *jing* XML instance to validate

```
$ yang2dsdl -s -j -t config -b turing-machine -v turing-machine-config.xml
== Using pre-generated schemas
== Validating grammar and datatypes ...
turing-machine-config.xml validates.
== Adding default values... done.
== Validating semantic constraints ...
No errors found.
```

Without `-j`, *xmllint* is used by default for RELAX NG validation – it works, too, but often gives inferior/wrong error messages.

DSDL Validation Procedure



Converting XML Instances to JSON

XML↔JSON mapping is defined in *draft-ietf-netmod-yang-json-00*.

JSON is optional media type in RESTCONF:
http://tools.ietf.org/html/draft-ietf-netconf-restconf-01

1. Generate XSLT 1.0 stylesheet with `jsonxsl` plugin:

   ```
   $ pyang -f jsonxsl -o tmjson.xsl turing-machine.yang
   ```

2. Apply the stylesheet to a valid XML instance document:

   ```
   $ xsltproc tmjson.xsl turing-machine-config.xml
   ```

   The same stylesheets works for **all** document types.

The `jtox` plugin performs the opposite conversion.

Further Information

1. NETMOD WG:

   http://datatracker.ietf.org/wg/netmod/documents/

2. NETCONF Central

   http://www.netconfcentral.org/

3. *pyang* wiki

   https://code.google.com/p/pyang/w/list

4. YANG Central

   http://www.yang-central.org/twiki/bin/view/Main/WebHome