Xcos-Converter

The aim of this project is to provide a software which converts a Scilab 5.5.2 file to Scilab 6.0.2 file. The code is written in python and the package used is Element tree. Python Element tree is a lxml package and it is a XML.

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The design goals of XML emphasize simplicity, generality, and usability across the Internet. XML is a markup language much like HTML. XML was designed to store and transport data. XML was designed to be self-descriptive.

XPath (XML Path Language) is a query language for selecting nodes from an XML document. XPath uses "path like" syntax to identify and navigate nodes in an XML document. XPath contains over 200 built-in functions. The XPath language is based on a tree representation of the XML document, and provides the ability to navigate around the tree, selecting nodes by a variety of criteria.

Xcos is a Scilab toolbox for modeling and simulation of dynamic (continuous and discrete) systems. Although the main purpose is to simulate dynamic systems, Xcos can be used for signal generation, data visualization or simple algebraic operations.

The relationship between Xcos and Xml is that Xcos is written in the same format as XML with Xcos specific tags. So, we can parse Xcos files as XML files.

The available Python Modules are:

xml.etree.ElementTree :

The *xml.etree.ElementTree* module implements a simple and efficient API for parsing and creating XML data.

Element.findall() method :

Element.findall() finds only elements with a tag which are direct children of the current element.

Signature :

Element.findall (path, namespaces:None)

Beautiful-Soup (bs4) :

Beautiful-Soup is a Python library for pulling data out of HTML and XML files. It works with parsers to provide idiomatic ways of navigating, searching, and modifying the parse tree.

BeautifulSoup.find_all() method :

The BeautifulSoup.find_all() method looks through a tag's descendants and retrieves all descendants that match filters.

Signature :

BeautifulSoup.find_all(name, attrs, recursive, string, limit, **kwargs)

Lxml :

Lxml is a Python library which allows for easy handling of XML and HTML files. It provides a very simple and powerful API for parsing XML and HTML. It supports one-step parsing as well as step-by-step parsing using an event-driven API.

Findall() method :
Finds all matching subelements, by tag name or path. The optional namespaces argument accepts a prefix-to-namespace mapping that allows the usage of XPath prefixes in the path expression.

Signature :
findall(path, namespaces)

The files used in this project are:

1. **Parser.py**

   The parser file uses conf.py to get the xpath and the rule. The function of conf.py will be explained later in the document.
   The first part of the code deals with xpath. Xpaths are needed to find the exact nodes we have to work with , it guarantees that no other nodes which might have similar tag names are affected.
   Inorder to build the xpath , it refers to the variable path in the conf file.
   Once the xpath is built it goes through the rule variable in the conf file to find out which operation has to be applied to the node found using xpath.
   The rules in parser.py are as follows :

   - DOUBLE_TO_INTEGER : In this rule , the ScilabDouble tags which have been selected using xpath the tag name is changed to ScilabInteger. A missing attribute, intPrecision = sci_int32 is also added to the node. Further in the child nodes, attribute ‘realPart’ is changed to ‘value’ and the respective values are changed from double to integer as well.

   - DELETE_ATTRIB : In this rule , the attribute mentioned in the rule variable if its present in the selected node is deleted.

- MAIN_BLOCK : In this rule, in the BasicBlock tag two tags are added . Also two attributes , dependOnT and dependOnU are added to BasicBlock tag if not present.

- ADD_SUB_SUBTAG : In this rule , we have to add nodes inside the tag mxGeometry . The rule will search through children nodes of mxGeometry and if there's no mxPoint tag present it will add them.

- DELETE_SUBTAG : In this rule, we remove the mxGeometry tag from ExplicitInputPort and ExplicitOutputPort tags.

- DOUBLE_TO_INTEGER_AND_SWAP : In this rule , if height attribute is present in ScilabDouble and its value is greater than zero the values of height and width are exchanged. The values of attribute line and column which are present in the subtags are also exchanged. Further operations are the same as mentioned in DOUBLE_TO_INTEGER.

- DELETE_SUB_ATTRIB : In this rule, the attributes 'x' and 'y' are deleted from the mxGeometry tag which is a subtag of either ExplicitLink tag or ImplicitLink tag.

- DELETE_SUBSUB_ATTRIB : In this rule , the attribute 'scilabclass' is deleted from the tag Array which is a subtag under mxGeometry.

- ADD_ATTRIB : In this rule, the dictionary of attributes mentioned in the rule section is added to the node selected using xpath.
- DELETE_TAG : In this rule , if the selected node is not None it is deleted.

- REPLACE_ATTRIB : In this rule attribute present in the path variable is replaced with the attribute present in rule variable.

- ADD_TAG : In this rule , if the new tag mentioned isn't present in the selected node then it is added as a child tag to the selected node.

- BLOCK_TYPE_H : This rule is specifically aimed for the SuperBlockDiagram tag. This rule adds tags to the SuperBlockDiagram tag. The rule is divided in a way to tackle the fact that tags have to be added in different levels i.e child nodes, grandchild nodes etc.

2. Conf.py

In this file we have globally defined all variables. All the operations that are to be performed are mentioned at the beginning of this file.
This file contains two lists i.e. *path* and *rule*.

Path :

Path has nodes like tag, attribute, subtag, sub attributes that are to be changed. It deals with Xpath in the parser.py file. This nodes are declared using pattern ‘KEY_PATH_*’.
Initially, path list is declared empty. Later on nodes get appended to this list according to the requirement of Xpath written in the parser.py file.

Rule :

Rule has a list of nodes with changed values. This nodes are declared using pattern ‘KEY_RULE_*’.

At first, this list is empty. When Xpath in the parser.py file gets the desired node, the operation related to it is performed. And the list gets appended with changed nodes. The operations are declared using the pattern ‘KEY_RULE_OP’.

The list of operations performed are as follows:

- DOUBLE_TO_INTEGER
- DELETE_ATTRIB
- MAIN_BLOCK
- ADD_SUB_SUBTAG
- DELETE_SUBTAG
- DOUBLE_TO_INTEGER_AND_SWAP
- DELETE_SUB_ATTRIB
- DELETE_SUBSUB_ATTRIB
- ADD_ATTRIB
- DELETE_TAG
- REPLACE_ATTRIB
- ADD_TAG
- BLOCK_TYPE_H

Future Scope :

ElementTree only gives child nodes by getchildren() method. To get all nodes, a findall method is used. But sometimes the findall method fails to give elements which are of NoneType.

lxml.etree offers a lot more functionality, such as XPath, XSLT and XML Schema support, which ElementTree does not offer. lxml.etree allows navigation to the parent of a node by the getparent() method and to the siblings by calling getnext() and getprevious(). This is not possible in ElementTree as the underlying tree model does not have this information.