Deployment of contraval (Contract Validator): (User's Guide Ver 1.2, 12 Jul 2018)

Carlos Molina–Jimenez[1] and Ioannis Sfyrakis[2]

[1] Computer Laboratory, University of Cambridge,
carlos.molina@cl.cam.ac.uk
[2] School of Computing Science, Newcastle University, UK,
giannisfyrakis@gmail.com

Abstract. This document is a walk through description of the deployment and usage of version 1.2 of *contraval*. *Contraval* is a software tool for validating the logical consistency of contracts conceived as a set of ECA (Event Condition Actions) rules.
The ECA rules that represent the contract are writen in the EPROMELA language and validating with the SPIN model checker.
Version 1.1 was implemented at Newcastle University, UK in 2012 and it did not offer support for generating executions sequences mechanically.
This version (1.2) incorporates a unix shell scrip and a python script that the user can use for generating execution sequences (test cases) for testing smart contracts.

1 Introduction

The repository includes several examples located in the examples folder. Each example can be run independently.

2 Data buyer–data seller Contract Example

The *datasellercontract* folder contains the *datasellercontract.pml* example to explain how use *contraval*.

To run the examples your local computer needs to have:

- *Spin:* Spin Version 6.1.0 or newer deployed on your local computer. Version of SPIN for Linux, Win and Mac users can be downloaded from the SPIN's home page [1].
- *Python:* We run these experiments with Python 2.7.15.

To run the example:

1. Copy one of the folders (say datasellercontract) into a folder of your local computer.
2. The folder includes several files:

   - example independent macros: setting.h and BizOperation.h vector.lpr and for.h. These macros implement the business operations added to PROMELA and are used in all examples.
   - *datasellercontract.pml* and *rule.h*: These files are example specific. In this example, they contain, respectively, the code for the events involved in the contract and the rules that are triggered by these events.
   - *pro2seq:* This is a unix shell script (promela 2 sequence).
   - *parser-filtering.py:* This is python program that is called by the pro2seq shell script to generate the execution sequences out from the Spin output files.
   - The folder includes two subfolder: *datasellercontract.pmlExecSeq* and *datasellercontract.pmlExecSeq-xml* that the script *pro2seq* produced from previous runs. These subfolders are deleted and re–produced by each run of *pro2seq*.
3. To run the code type:

```
% pro2seq
```

4. The *datasellercontract.pml* model is provided with a LTL formulae that instructs spin to generate all the execution sequences implicitly encoded in the epromela model.
5. The sequences are store in a the folder datasellercontract.pmlExecSeq-xml.

3 Buyer–Store Contract Example

The *BuyerStoreContract* folder contains the *BuyerStoreContract.pml* example. We used it in version 1.1 to explain how use *contraval*. The folder does not include the *pro2seq* or *parser-filtering.py* scrypts. The user is expected to generate the execution sequences manually.

To run the examples you need to have Spin Version 6.1.0 or newer deployed on your local computer. Version of SPIN for Linux, Win and Mac users can be downloaded from the SPIN's home page [1].

We will assume a Linux machine, but the examples can be run in Win or Mac as well.

1. Copy one of the folders (say BuyerStoreContract) into a folder of your local computer.
2. The folder includes several files:
   - example independent macros: setting.h and BizOperation.h vector.lpr and for.h. These macros implement the business operations added to PROMELA and are used in all examples.
   - BuyerStoreContract.pml and rule.h. These files are example specific. In this example, they contain, respectively, the code for the events involved in the contract and the rules that are triggered by these events.
3. To run the code type:

```
% spin -a BuyerStoreContract.pml
% cc -o pan pan.c
% pan -a
```

4. The *BuyerStoreContract.pml* model is provided with several LTL formulae. You can edit the file to comment and uncomment them as needed. Keep in mind that SPIN can verify only a single LTL at a time.

A detailed explanation of the *BuyerStoreContract.pml* can be found in a technical report ([2]) which is also include in the *BuyerStoreContract* folder.

4 Implementation

Details about the implementation of *contraval* and additional case studies can be found in Abubkr Abdelsadiq's PhD dissertation that originated this work [3].

5 Licence

The *contraval* tool is released under the Apache License, Version 2.0[4], which is available from Apache's web pages. Also, you can find a *txt* copy from our home page [5].

6 Implementation History

Table 1. Contraval–implementation history.

Version	Date	Contributors	Key features
1.2	Jul 2018	Carlos and Ioannis	Implementation of a unix shell script and a python script for generating execution sequences (test cases) mechanically from epromela models.
1.1	Sep 2012	Abubkr Abdelsadiq	Implementation of macros for the specification of business operations.

Acknowledgment

Carlos Molina-Jimenez is currently collaborating with the HAT Community Foundation under the support of Grant RG90413 NRAG/536.

Ioannis Sfykaris was partly supported by the EU Horizon 2020 project PrismaCloud (https://prismacloud.eu) under GA No. 644962.

References

1. SPIN: On-the-fly, ltl model checking with spin. http://spinroot.com (visited in Jul 2012 2012)
2. Molina-Jimenez, C., Shrivastava, S.: Establishing conformance between contracts and choreographies. Technical Report CS-TR-1383, School of Computing Science, Newcastle Univ. UK (April 2013)
3. Abdelsadiq, A.A.: A Toolkit for model checking of electronic contracts. PhD thesis, School of Computing Science, Newcastle University, UK (September 2012)
4. Foundation, T.A.S.: Apache license version 2.0, january. http://www.apache.org/licenses (2004)
5. Molina-Jimenez, C.: Carlos molina–jimenez home page. http://homepages.cs.ncl.ac.uk/carlos.molina (2012)