The RAVEN PRA Plugin
- User Manual -

D. Mandelli, C. Wang, A. Alfonsi

May 8, 2019

Contents

1 Introduction

This document provides a detailed description of the PRA plugin for the RAVEN [1,2] code. The features included in this plugin are:

- Event Tree (ET) Model (see Section 2)
- Fault Tree (FT) Model (see Section 3)
- Markov Model (see Section 4)
- Reliability Block Diagram (RBD) Model (see Section 5)
- Data Classifier (see Section 6)
- ET Data Importer (see Section 7)
- FT Data Importer (see Section 8)

2 ET Model

This model is designed to read from file the structure of the ET and to import such Boolean logic structure as a RAVEN model. The ET must be specified in a specific format: the OpenPSA format (https://github.com/open-psa). As an example, the ET of Fig. 1 is translated in the OpenPSA format as shown below:

Listing 1: ET of Fig. 1 in OpenPSA format.

```
<define-event-tree name="eventTree">
    <define-functional-event name="ACC"/>
    <define-functional-event name="LPI"/>
    <define-functional-event name="LPR"/>
    <define-sequence name="1"/>
    <define-sequence name="2"/>
    <define-sequence name="3"/>
    <define-sequence name="4"/>
    <initial-state>
        <fork functional-event="ACC">
            <path state="0">
                <fork functional-event="LPI">
                    <path state="0">
                        <fork functional-event="LPR">
                            <path state="0">
                                <sequence name="1"/>
                            </path>
                            <path state="+1">
                                <sequence name="2"/>
                            </path>
                        </fork>
                    </path>
                    <path state="+1">
                        <sequence name="3"/>
                    </path>
                </fork>
            </path>
            <path state="+1">
                <sequence name="4"/>
            </path>
        </fork>
    </initial-state>
</define-event-tree>
```

The ET of Fig. 1 and defined in Listing 1 can be defined in the RAVEN input file as

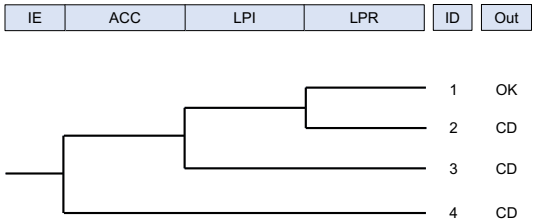

Figure 1: Example of ET.

follows:

Listing 2: ET model input example.

```
<Models>
  ...
  <ExternalModel name="ET" subType="ETModel">
    <variables>
      statusACC,statusLPI,statusLPR,sequence
    </variables>
    <map var="statusACC">ACC</map>
    <map var="statusLPI">LPI</map>
    <map var="statusLPR">LPR</map>
    <sequenceID>sequence</sequenceID>
  </ExternalModel>
  ...
</Models>
```

All the specifications of the ET model are given in the `<ExternalModel>` block. Inside the `<ExternalModel>` block, the XML nodes that belong to this models are:

- `<variables>`, ***string, required parameter***, a list containing the names of both the input and output variables of the model
- `<sequenceID>`,***string, required parameter***, the name of the alias variable that indicate the branch ID
- `<map>`,***string, required parameter***, the name ID of the ET branching variable
  - `var`, ***required string attribute***, the ALIAS name ID of the ET branching variable

Provided this definition and the ET model of Fig. 1 and described in Listing 1, the resulting model in RAVEN is characterized by these variables:

- Input variables: statusACC, statusLPI, statusLPR
- Output variable: sequence

2.1 ET model reference tests

- test_ETmodel.xml
- test_ETmodel_TD.xml

3 FT Model

This model is designed to read from file the structure of the FT and to import such Boolean logic structure as a RAVEN model. The FT must be specified in a specific format: the OpenPSA format (https://github.com/open-psa). As an example, the FT of Fig. 2 is translated in the OpenPSA as shown below:

Listing 3: FT in OpenPSA format.

```
<opsa-mef>
    <define-fault-tree name="FT">
        <define-gate name="TOP">
            <or>
                <gate name="G1"/>
                <gate name="G2"/>
                <gate name="G3"/>
            </or>
        </define-gate>
        <define-component name="A">
            <define-gate name="G1">
                <and>
                    <basic-event name="BE1"/>
                    <basic-event name="BE2"/>
                </and>
            </define-gate>
            <define-gate name="G2">
                <and>
                    <basic-event name="BE1"/>
                    <basic-event name="BE3"/>
                </and>
            </define-gate>
            <define-basic-event name="BE1">
                <float value="1.2e-3"/>
            </define-basic-event>
            <define-component name="B">
                <define-basic-event name="BE2">
                    <float value="2.4e-3"/>
                </define-basic-event>
                <define-basic-event name="BE3">
                    <float value="5.2e-3"/>
                </define-basic-event>
            </define-component>
        </define-component>
        <define-component name="C">
```

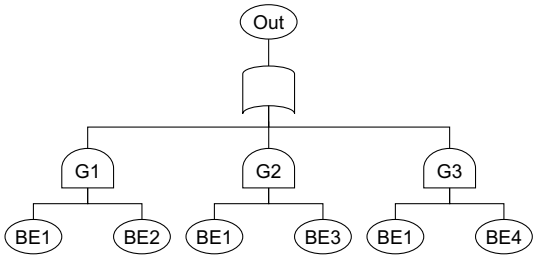

Figure 2: Example of FT.

```
            <define-gate name="G3">
                <and>
                    <basic-event name="BE1"/>
                    <basic-event name="BE4"/>
                </and>
            </define-gate>
            <define-basic-event name="BE4">
                <float value="1.6e-3"/>
            </define-basic-event>
        </define-component>
    </define-fault-tree>
</opsa-mef>
```

The FT of Fig. 2 and defined in Listing 3 can be defined in the RAVEN input file as follows:

Listing 4: FT model input example.

```
  <Models>
    ...
    <ExternalModel name="FT" subType="FTModel">
      <variables>
        statusBE1,statusBE2,statusBE3,statusBE4,TOP
      </variables>
      <topEvents>TOP</topEvents>
      <map var="statusBE1">BE1</map>
      <map var="statusBE2">BE2</map>
      <map var="statusBE3">BE3</map>
      <map var="statusBE4">BE4</map>
    </ExternalModel>
    ...
  </Models>
```

All the specifications of the FT model are given in the `<ExternalModel>` block. Inside the `<ExternalModel>` block, the XML nodes that belong to this models are:

- <variables>, ***string, required parameter***, a list containing the names of both the input and output variables of the model
- <topEvents>,***string, required parameter***, the name of the alias Top Event
- <map>,***string, required parameter***, the name ID of the FT basic events
  - var, ***required string attribute***, the ALIAS name ID of the FT basic events

Provided this definition, the FT model of Fig. 1 and described in Listing **??**, the resulting model in RAVEN is characterized by these variables:

- Input variables: statusBE1, statusBE2, statusBE3, statusBE4
- Output variable: TOP

3.1 FT model reference tests

- test_FTmodel.xml
- test_FTmodel_TD.xml

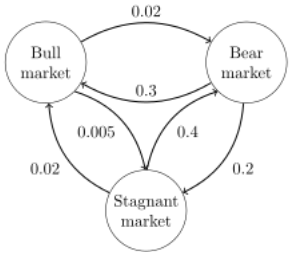

Figure 3: Example of continuous time Markov chain (source Wikipedia: https://en.wikipedia.org/wiki/Markov_chain).

4 Markov Model

This model is designed to import a generic Markov chain as a RAVEN model. As an example, the Markov chain of Fig. 3 is translated in the OpenPSA as shown below:

Listing 5: Markov model input example.

```
<Models>
  <ExternalModel name="markov" subType="MarkovModel">
    <variables>initialState,finalState</variables>
    <initState>initialState</initState>
    <finState>finalState</finState>
    <endTime>1000</endTime>
    <state name="1"> <!-- Bull market -->
      <transition type="lambda" value="0.02" >2</transition>
      <transition type="lambda" value="0.005">3</transition>
    </state>
    <state name="2"> <!-- Bear market -->
      <transition type="lambda" value="0.3">1</transition>
      <transition type="lambda" value="0.2">3</transition>
    </state>
    <state name="3"> <!-- Stagnant market -->
      <transition type="lambda" value="0.02">1</transition>
      <transition type="lambda" value="0.4" >2</transition>
    </state>
  </ExternalModel>
</Models>
```

All the specifications of the Markov model are given in the `<ExternalModel>` block. Inside the `<ExternalModel>` block, the XML nodes that belong to this model are:

- `<variables>`, ***string, required parameter***, a list containing the names of both the input and output variables of the model

- `<initState>`, ***string, required parameter***, variable ID corresponding to initial state
- `<finState>`, ***string, required parameter***, variable ID corresponding to final state
- `<endTime>`, ***float, required parameter***, time horizon to evaluate Markov chain transition history
- `<state>`, specifies a single node; inside a `<state>` all possible transitions OUT of this state must be specified in the `<transition>` xml sub-nodes:
  - `transition`, ***required string attribute***, arrival state
  - `type`, ***required string attribute***, type of transition. Allowed transition types are: The ET of Fig. 1 and defined in Listing **??** can be defined in the RAVEN input file as follows: lambda, tau, instant and unif (see below)
  - `value`, ***required string attribute***, value associated to the particular transition

The following transition types are available:

- lambda: classical continuous time Markov chain transition rate in λ form
- tau: classical continuous time Markov chain transition rate in the $\tau = \frac{1}{\lambda}$ form
- instant: deterministic transition out of particular state; the exact transition time is provided in input
- unif: transition time is uniformly sampled between the two provided values in the `value` node

4.1 Markov model reference tests

- test_markovModel_2states_tau.xml
- test_markovModel_2states.xml
- test_markovModel_3states_complexTrans.xml
- test_markovModel_3states_instantTrans.xml
- test_markovModel_3states.xml

5 RBD Model

This model is designed to read from file the structure of the RBD and import such Boolean logic structure as a RAVEN model. The RBD must be specified in a specific format; as an example, the RBD of Fig. 4 is translated in the RAVEN format as shown below:

Listing 6: RBD input file.

```
<Graph name="testGraph">
  <node name="CST">
    <childs>1</childs>
  </node>
  <node name="1">
    <childs>2,3,4</childs>
  </node>
  <node name="2">
    <childs>5</childs>
  </node>
  <node name="3">
    <childs>5</childs>
  </node>
  <node name="4">
    <childs>5</childs>
  </node>
  <node name="5">
    <childs>6,7,8</childs>
  </node>
  <node name="6">
    <childs>SG1</childs>
  </node>
  <node name="7">
    <childs>SG2</childs>
  </node>
  <node name="8">
    <childs>SG3</childs>
  </node>
  <node name="SG1">
  </node>
  <node name="SG2">
  </node>
  <node name="SG3">
  </node>
</Graph>
```

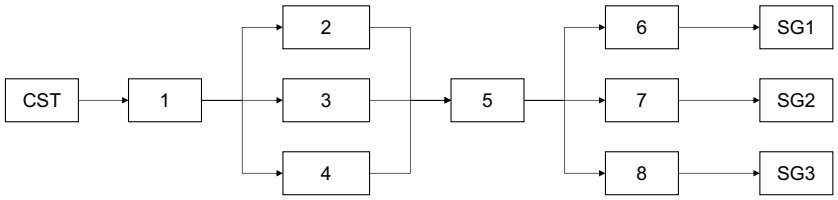

Figure 4: Example of RBD.

The FT of Fig. 4 and defined in Listing 6 can be defined in the RAVEN input file as follows:

Listing 7: RBD model input example.

```
<Models>
  ...
  <ExternalModel name="graph" subType="GraphModel">
    <variables>
      status2,status3,status4,status5,
      statusSG1,statusSG2,statusSG3
    </variables>
    <modelFile>graphTest</modelFile>
    <nodesIN>CST</nodesIN>
    <nodesOUT>SG1,SG2,SG3</nodesOUT>
    <map var="status2">2</map>
    <map var="status3">3</map>
    <map var="status4">4</map>
    <map var="status5">5</map>
    <map var="statusSG1">SG1</map>
    <map var="statusSG2">SG2</map>
    <map var="statusSG3">SG3</map>
  </ExternalModel>
  ...
</Models>
```

All the specifications of the RBD model are given in the `<ExternalModel>` block. Inside the `<ExternalModel>` block, the XML nodes that belong to this models are:

- `<variables>`, ***string, required parameter***, a list containing the names of both the input and output variables of the model
- `<modelFile>`,***string, required parameter***, the name of the file that provide the RBD structure
- `<nodesIN>`,***string, required parameter***, the name of the input nodes

- <nodesOUT>,***string, required parameter***, the name of the output nodes
- <map>,***string, required parameter***, the name ID of the RBD node
  - var, ***required string attribute***, the ALIAS name ID of the RBD node

Provided this definition, the RBD model of Fig. 4 and described in Listing 6, the resulting model in RAVEN is characterized by these variables:

- Input variables: status2, status3, status4, status5
- Output variable: statusSG1, statusSG2, statusSG3

5.1 RBD model reference tests

- test_graphModel.xml
- test_graphModel_TD.xml

6 Data Classifier

The **DataClassifier** post-processor is specifically used to classify the data stored in the DataObjects. The details about this post-processors can be found in raven user manual subsection **PostProcessor** of section **Models**.

6.1 Data Classifier reference tests

- test_dataClassifier_postprocessor.xml
- test_dataClassifier_postprocessor_HS.xml

7 ET Data Importer

This Post-Processor is designed to import an ET as a PointSet in RAVEN. The ET must be specified in a specific format: the OpenPSA format (https://github.com/open-psa). The details about this post-processors can be found in raven user manual subsection **PostProcessor** of section **Models**.

7.1 ET Importer reference tests

- test_ETimporter.xml
- test_ETimporterMultipleET.xml
- test_ETimporterSymbolic.xml
- test_ETimporter_expand.xml
- test_ETimporter_DefineBranch.xml
- test_ETimporter_3branches.xml
- test_ETimporter_3branches_NewNumbering.xml
- test_ETimporter_3branches_NewNumbering_expanded.xml

8 FT Data Importer

This Post-Processor is designed to import a FT as a PointSet in RAVEN. The FT must be specified in a specific format: the OpenPSA format (https://github.com/open-psa). The details about this post-processors can be found in raven user manual subsection **PostProcessor** of section **Models**.

8.1 FT Importer reference tests

- test_FTimporter_and_withNOT_embedded.xml
- test_FTimporter_and_withNOT_withNOT_embedded.xml
- test_FTimporter_and_withNOT.xml
- test_FTimporter_and.xml
- test_FTimporter_atleast.xml
- test_FTimporter_cardinality.xml
- test_FTimporter_component.xml
- test_FTimporter_doubleNot.xml
- test_FTimporter_iff.xml
- test_FTimporter_imply.xml
- test_FTimporter_multipleFTs.xml
- test_FTimporter_nand.xml
- test_FTimporter_nor.xml
- test_FTimporter_not.xml
- test_FTimporter_or_houseEvent.xml
- test_FTimporter_or.xml
- test_FTimporter_xor.xml

Document Version Information

This document has been compiled using the following version of the plug-in git repository: c5d7b2e5bfa7b6d5c9e94acbad8d0082d5a64ce6 Congjian Wang Wed, 8 May 2019 09:23:10 -0600

References

[1] C. Rabiti, A. Alfonsi, J. Cogliati, D. Mandelli, R. Kinoshita, S. Sen, C. Wang, and J. Chen, "Raven user manual," Tech. Rep. INL/EXT-15-34123, March 2017.

[2] A. Alfonsi, C. Rabiti, D. Mandelli, J. Cogliati, C. Wang, P. W. Talbot, D. P. Maljovec, and C. Smith, "Raven theory manual and user guide," tech. rep., Idaho National Laboratory (INL), 2017.