CRADLE

CRADLE is an online plan recognition algorithm. It extents the PHATT algorithm by Geib and Goldman. It is constraining the space of possible explanations. We designed several “filters” that reduce the size of the explanation set in a way that reflects the intended use of plan recognition in exploratory environments. Specifically, the filters aim to produce complete, parsimonious and coherent explanations. Its acronyms stand for Cumulative Recognition of Activities and Decreasing Load of Explanations.

CRADLE is an incremental algorithm. It calculates one observation at a time, based on the explanations produced so far. For each observation, it performs two stages of processing:

1. Incrementally combine the new observation to existing explanations in every possible way.
2. Measure the features of all explanations and filter away explanations with features below the thresholds.

CRADLE was written in python v.2.7 and is recommended to run with the JIT compiler pypy. It was tested on Windows 7 and Ubuntu 12.04.

CRADLE was first published in EDM 2014 and the paper can be found in the root folder of this project.

Installation

In order to run CRADLE, you must have python 2.7 or pypy installed.

Running from source

The algorithm main module is called CRADLE.py and is located in the root folder.

Each CRADLE run requires 2 different files (an example of these files can be found in the folder "Example Domain"):

1. Domain file – which describes the grammar , the goals and the possible symbols to observe
2. Observation file – which describes a set of actions performed in the domain.

Running under python:

- Install Python 2.7
- Usage: python.exe CRADLE.py <domain file> <observations file>

Running under pypy:

- Install PyPy 3.2.1
- pypy.exe CRADLE.py <domain file> <observations file>

How to read the output

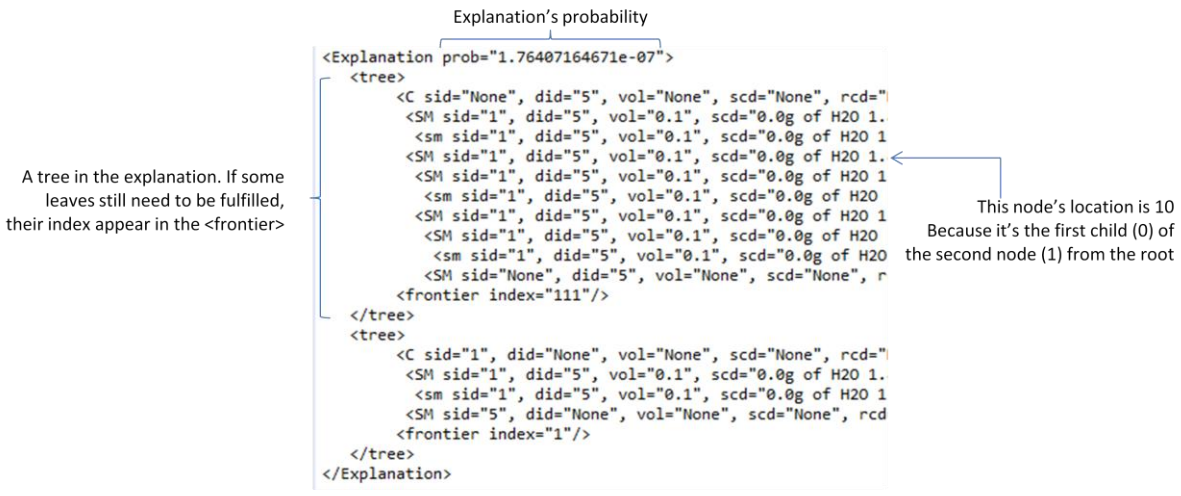
CRADLE output is divided into 3 sections:

1. Verification of the input
2. Results
3. Statistics

The verification just prints back to the screen all information it received – this is the plan library created from the domain file and the list of observations from the observations file.

The results section prints the most probable explanation using uniform probability on all the rules, the amount of all explanations produced and how many of which were with no open frontier.

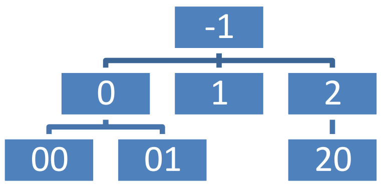
This is a description of the contents of an outputted explanation:


Explanation's probability
A tree in the explanation. If some leaves still need to be fulfilled, their index appear in the <frontier>
This node's location is 10 Because it's the first child (0) of the second node (1) from the root

```
<Explanation prob="1.76407164671e-07">
   <tree>
        <C sid="None", did="5", vol="None", scd="None", rcd="
         <SM sid="1", did="5", vol="0.1", scd="0.0g of H2O 1.
          <sm sid="1", did="5", vol="0.1", scd="0.0g of H2O 1
         <SM sid="1", did="5", vol="0.1", scd="0.0g of H2O 1.
          <SM sid="1", did="5", vol="0.1", scd="0.0g of H2O 1
           <sm sid="1", did="5", vol="0.1", scd="0.0g of H2O
          <SM sid="1", did="5", vol="0.1", scd="0.0g of H2O 1
           <SM sid="1", did="5", vol="0.1", scd="0.0g of H2O
            <sm sid="1", did="5", vol="0.1", scd="0.0g of H2O
           <SM sid="None", did="5", vol="None", scd="None", r
        <frontier index="111"/>
   </tree>
   <tree>
        <C sid="1", did="None", vol="None", scd="None", rcd="
         <SM sid="1", did="5", vol="0.1", scd="0.0g of H2O 1.
          <sm sid="1", did="5", vol="0.1", scd="0.0g of H2O 1
         <SM sid="5", did="None", vol="None", scd="None", rcd
        <frontier index="1"/>
   </tree>
</Explanation>
```

The most confusing part is probably the encoding of a location along the tree. A location is represented using a string which is read from left to right – each digit represents the child's index in a specific level.

In this figure, each node is labeled by its location:



Of course, this representation creates a limit to the algorithm – in its current stage **you cannot represent grammars with more than 10 symbols on the right side of a rule.**

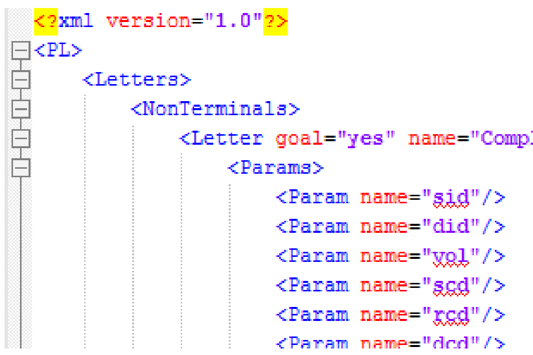
If you need this functionality, please contact me ☺

The last part of the output is the statistics generated by the profiler.

Domain file

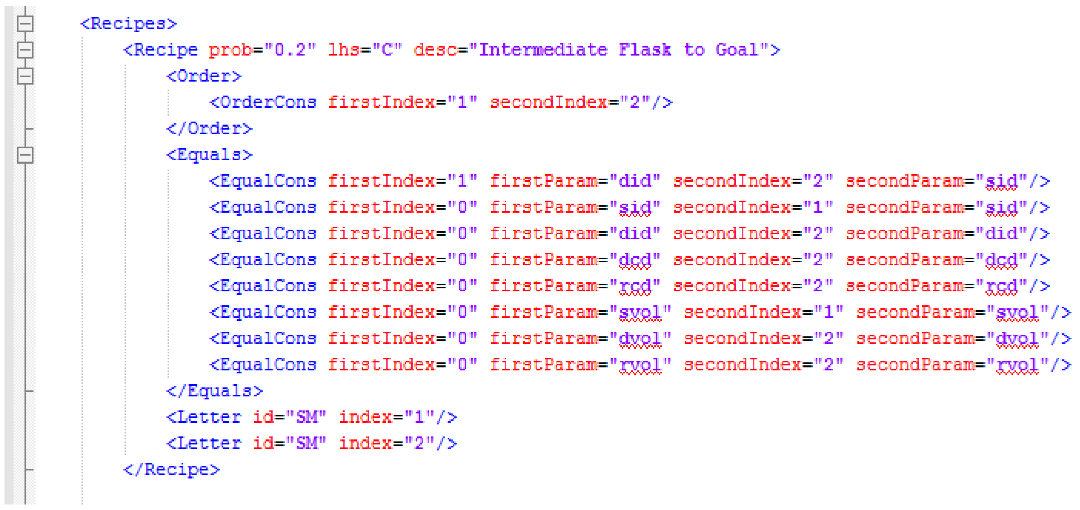
The domain file needs to contain 2 parts: letters and rules. The letters also contains two parts: terminal symbols (Sigmas), and non-terminal symbols (NTs).

This is how a domain file should look like:



```xml
<?xml version="1.0"?>
<PL>
    <Letters>
        <NonTerminals>
            <Letter goal="yes" name="Complex Mix" id="C">
                <Params>
                    <Param name="sid"/>
                    <Param name="did"/>
                    <Param name="vol"/>
                    <Param name="scd"/>
                    <Param name="rcd"/>
                    <Param name="dcd"/>
```

Above is the beginning of the letter section, containing a description of a non-terminal action called "C", which can be the goal.



```xml
    <Recipes>
        <Recipe prob="0.2" lhs="C" desc="Intermediate Flask to Goal">
            <Order>
                <OrderCons firstIndex="1" secondIndex="2"/>
            </Order>
            <Equals>
                <EqualCons firstIndex="1" firstParam="did" secondIndex="2" secondParam="sid"/>
                <EqualCons firstIndex="0" firstParam="sid" secondIndex="1" secondParam="sid"/>
                <EqualCons firstIndex="0" firstParam="did" secondIndex="2" secondParam="did"/>
                <EqualCons firstIndex="0" firstParam="dcd" secondIndex="2" secondParam="dcd"/>
                <EqualCons firstIndex="0" firstParam="rcd" secondIndex="2" secondParam="rcd"/>
                <EqualCons firstIndex="0" firstParam="svol" secondIndex="1" secondParam="svol"/>
                <EqualCons firstIndex="0" firstParam="dvol" secondIndex="2" secondParam="dvol"/>
                <EqualCons firstIndex="0" firstParam="rvol" secondIndex="2" secondParam="rvol"/>
            </Equals>
            <Letter id="SM" index="1"/>
            <Letter id="SM" index="2"/>
        </Recipe>
```

Above is an example of a rule in the domain – each rule has a probability as an attribute and a left hand side letter as an attribute named "lhs". Moreover, each rule contains the element "order", "equals" and one or more element of the type "letter". The "order" element describes the ordering constraints between the letters (for example, in the above case, the first SM must appear before the second SM). The "equals" element describes the equality constraints on letter's parameters (for example, in the above case, the value of the parameter named "did" of the first action must be the same as the value of the parameter named "sid" of the second action. Notice that the index 0 refers to parameters of the left hand side action).

An example domain file can be found in "ExampleDomain/Domain.xml"

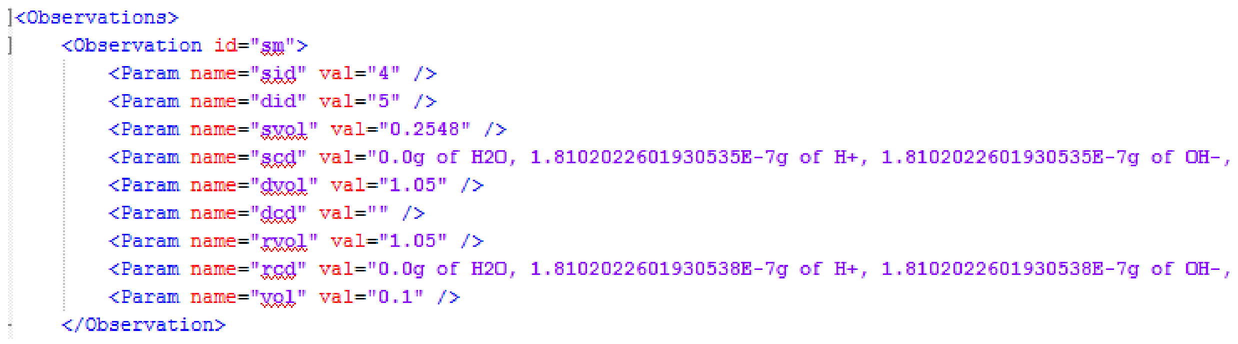
Observations file

The observation file is in xml format. It contains a root attribute called "observations", which holds a list of observations; each is a different element.

Each observation should have the "id" attribute (these should always be an id from the domain's Sigmas) and then a list of parameters and their assignments. The list should be represented as described below:

- Each parameter is a different element
- The parameter's name is represented by the attribute "name"
- The parameter's value is represented by the attribute "value"
- Attribute values should always be wrapped with quotes

An example of an observation:



```
<Observations>
    <Observation id="sm">
        <Param name="sid" val="4" />
        <Param name="did" val="5" />
        <Param name="svol" val="0.2548" />
        <Param name="scd" val="0.0g of H2O, 1.8102022601930535E-7g of H+, 1.8102022601930535E-7g of OH-,
        <Param name="dvol" val="1.05" />
        <Param name="dcd" val="" />
        <Param name="rvol" val="1.05" />
        <Param name="rcd" val="0.0g of H2O, 1.8102022601930538E-7g of H+, 1.8102022601930538E-7g of OH-,
        <Param name="vol" val="0.1" />
    </Observation>
```

An example of observation file can be found in "ExampleDomain/Observations.xml"

Copyrights and Licensing

Contact Information

If you have any questions, bug reports or ideas, please contact me at:

reuthde@gmail.com