Rule Engine Challenge

Problem Statement

Build a rule engine that will apply rules on streaming data. Your program should be able to perform following tasks, at minimum:

- Allow users to create rules on incoming data stream
- Execute rules on incoming stream and show the data that violates a rule.

Incoming data stream is a tagged data stream. Each incoming data is a hashmap with following syntax

```
{ 'signal': 'ATL1', 'value': '234.98', 'value_type': 'Integer'}
{ 'signal': 'ATL2', 'value': 'HIGH', 'value_type': 'String'}
{ 'signal': 'ATL3', 'value': '23/07/2017 13:24:00.8765', 'value_type': 'Datetime'}
...
...
...
```

In general, a data unit would have three keys

- **signal**: This key specifies the source ID of the signal. It could be any valid alphanumeric combo. ex: ATL1, ATL2, ATL3, ATL4
- **value**: This would be the actual value of the signal. This would always be a string. ex: '234', 'HIGH', 'LOW', '23/07/2017'
- **value_type**: This would specify how the value is to interpreted. It would be one of the following
    - *Integer*: In this case the value is interpreted to be an integer. Ex: '234' would be interpreted as 234
    - *String*: In this case the value is interpreted to be a String. Ex: 'HIGH' would be interpreted as 'HIGH'
    - *Datetime*: In this case the value is interpreted to be a Date Time.

Rules can be specified for a signal and in accordance to the value_type. Some examples of rules are:

- ATL1 value should not rise above 240.00
- ATL2 value should never be LOW
- ATL3 should not be in future

Your solution should have minimal external dependencies, preferably none (testing frameworks and build systems are allowed). You should cap your effort to one day worth of work on the coding challenge itself, prioritising according to what you believe matters most.

Discussion Questions

In addition to a code submission, please provide short answers to the following discussion questions in a README (plain-text or Markdown format):

- Briefly describe the conceptual approach you chose! What are the trade-offs?
- What's the runtime performance? What is the complexity? Where are the bottlenecks?
- If you had more time, what improvements would you make, and in what order of priority?

Preparation

Download the raw_signal.json file. It contains the test case on which your solution would be evaluated.

Technical Criteria

Your solution would be tested on a Unix like system (Linux or Mac OS X). For each input data signal, only emit the name of the source that was violated by a rule specified. Your rules need to be stored on a persistent storage format (text/yaml/ini file, database, etc.) independent of the source code.

Assessment Criteria

In no specific order:

- If your solution satisfies the requirements

- How the code and functionality is tested
- The understandability and maintainability of your code
- The cleanliness of design and implementation
- Time performance on a standard laptop
- Answers to the discussion questions.

raw_data.json