A scripting language to analyse semi-structured textual data

Guillaume Baudart, Louis Mandel, Olivier Tardieu, Mandana Vaziri

Jamie Jennings

IBM T.J. Watson

IBM Cloud
### Goal
- Monitoring: report alerting state
- Troubleshooting: find causes of a crash

### Challenges
- Lots of semi-structured data
- Format can change over time
A scripting language

Goal

- Monitoring: report alerting state
- Troubleshooting: find causes of a crash

Challenges

- Lots of semi-structured data
- Format can change over time

Built on top of JavaScript with dedicated log analysis features
Programming Model

- **source**
  - get input data
  - disk, web, databases...

- **group**
  - re-structure data

- **match**
  - extract structure

- **stream**
  - react when match

- **block, restart**
  - triggered only once

- Repeat and compose
Repeat and Compose

Serial composition

Ordered parallel composition

CloudLens: default parallel composition + explicit restart
match {
    "(?<failed>.* > .* FAILED"
}

var failed = 0;

stream (entry) when (entry.failed) {
    print("FAILED:", entry.failed)
    failed++;
}

{ print(failed, "failed tests") }
match {
    "(?<failed>.*\>) \.* FAILED"
}

var failed = 0;

stream (entry) {
    print("FAILED:", entry.failed)
    failed++;
}

{ print(failed, "failed tests") }
var dateFormat = "Date[yyyy-MM-dd' 'HH:mm:ss.SSS]";

match {
    "Starting test (?<desc>.*) at (?<start>.*)# start:" + dateFormat;
    "Finished test (?<desc>.*) at (?<end>.*)# end:" + dateFormat
}

var start;

stream (entry) {
    start = entry.start;
}

stream (entry) {
    entry.dur = entry.end - start;
    if (entry.dur > 12000) {
        print(entry.dur, "\t", entry.desc);
    }
}
system.basic.WskBasicTests > Wsk Action CLI should reject delete of action that does not exist FAILED
  org.scalatest.exceptions.TestFailedException: "error: Unable to delete action: Request failure: The requested resource does not exist. (code 914)"
did not include substring that matched regex error: The requested resource does not exist. \(\text{code} \ d+\)
  at org.scalatest.MatchersHelper$.newTestFailedException(MatchersHelper.scala:160)
  at org.scalatest.Matchers$ResultOfIncludeWordForString.regex(Matchers.scala:2201)
  at system.basic.WskBasicTests$$anonfun$25.apply$mcV$sp(WskBasicTests.scala:295)
  at system.basic.WskBasicTests$$anonfun$25.apply(WskBasicTests.scala:295)
  at system.basic.WskBasicTests$$anonfun$25.apply(WskBasicTests.scala:295)
  at org.scalatest.Transformer$$anonfun$apply$1.apply$mcV$sp(Transformer.scala:22)
  at org.scalatest.OutcomeOf$class.outcomeOf(OutcomeOf.scala:85)
  at org.scalatest.OutcomeOf$.outcomeOf(OutcomeOf.scala:104)
  at org.scalatest.Transformer.apply(Transformer.scala:22)
  at org.scalatest.Transformer.apply(Transformer.scala:20)
  at org.scalatest.FlatSpecLike$$anon$1.apply(FlatSpecLike.scala:1647)
  at org.scalatest.FlatSpec.withFixture(FlatSpec.scala:1683)
  at org.scalatest.FlatSpecLike$class.invokeWithFixture$1(FlatSpecLike.scala:1644)
  at org.scalatest.FlatSpecLike$class.invokeWithFixture$1(FlatSpecLike.scala:1644)
  at org.scalatest.FlatSpecLike$non$1.apply(FlatSpecLike.scala:1656)
  at org.scalatest.FlatSpecLike$non$1.apply(FlatSpecLike.scala:1656)
  at org.scalatest.FlatSpecLike$non$runTests$1.apply(FlatSpecLike.scala:1656)
  at org.scalatest.FlatSpecLike$non$runTests$1.apply(FlatSpecLike.scala:1656)
  at org.scalatest.FlatSpecLike$non$runTests$1.apply(FlatSpecLike.scala:1656)
  at org.scalatest.FlatSpecLike$non$runTests$1.apply(FlatSpecLike.scala:1656)
  at system.basic.WskBasicTests.org$scalatest$BeforeAndAfterEachTestData$super$runTest(WskBasicTests.scala:50)
  at org.scalatest.BeforeAndAfterEachTestData$class.runTest(BeforeAndAfterEachTestData.scala:193)
  at system.basic.WskBasicTests.runTest(WskBasicTests.scala:50)
system.basic.WskBasicTests > Wsk Action CLI should reject delete of action that does not exist FAILED
org.scalatest.exceptions.TestFailedException: "error: Unable to delete action: Request failure: The requested resource does not exist. (code 914)"
  did not include substring that matched regex error: The requested resource does not exist. \d+
  at org.scalatest.MatchersHelper$.newTestFailedException(MatchersHelper.scala:160)
  at org.scalatest.Matchers$ResultOfIncludeWordForString.regex(Matchers.scala:2201)
  at org.scalatest.Matchers$ResultOfIncludeWordForString.regex(Matchers.scala:2173)
  at system.basic.WskBasicTests$$anonfun$25.apply$mcV$sp(WskBasicTests.scala:295)
  at system.basic.WskBasicTests$$anonfun$25.apply(WskBasicTests.scala:295)
  at system.basic.WskBasicTests$$anonfun$25.apply(WskBasicTests.scala:295)
  at org.scalatest.Transformer$$anonfun$apply$mcV$sp(Transformer.scala:22)
  at org.scalatest.OutcomeOf$class.outcomeOf(OutcomeOf.scala:85)
  at org.scalatest.Translate$$anonfun$apply$mcV$sp(Translator.scala:85)
  at org.scalatest.Transformer.apply(Transformer.scala:22)
  at org.scalatest.Transformer.apply(Transformer.scala:20)
  at org.scalatest.FlatSpecLike$$anonfun$runTest$1.apply(FlatSpecLike.scala:1656)
  at org.scalatest.FlatSpecLike$$anonfun$runTest$1.apply(FlatSpecLike.scala:1656)
  at org.scalatest.Transformer.runTestImpl(Engine.scala:306)
  at org.scalatest.FlatSpecLike$$anonfun$runTest$1.runTest(FlatSpecLike.scala:1656)
  at system.basic.WskBasicTests.runTest(WskBasicTests.scala:50)
  at system.basic.WskBasicTests.runTest(WskBasicTests.scala:50)

• **source**
  - stream of JSON object

• **group**
  - combine consecutive entry into arrays

• **lens**
  - define CloudLens functions
match {
    "(?<failed>.* > .*) FAILED";
}

group {
    "^[^ ]"
}

lens stackCheck() {
    match {
        "at .\((?<whisk>Wsk.*)\)"
    }

    stream (line) {
        print(' at', line.whisk)
    }
}

stream (entry) when (entry.failed) {
    print("FAILED", entry.failed);
    stackCheck(entry.group)
}
lens testStart() {
  match {
    "Starting test (?<start>.* ) at (?<date>.* )"
  }

  stream (entry) {
    print("Starting", entry.start)
  }
}

lens testStop() {
  match {
    "Finished test (?<end>.* ) at (?<date>.* )"
  }

  stream (entry) {
    print("Finished", entry.end)
  }
}

{ testStart();
  testStop();
}
lens testStart () {
    match {
        "Starting test (?<start>.*+) at (?<date>.*+)
    }

    stream (entry) {
        print("Starting", entry.start)
    }
}

lens testStop() {
    match {
        "Finished test (?<end>.*+) at (?<date>.*+)
    }

    stream (entry) {
        print("Finished", entry.end)
    }
}

run testStart()
run testStop()
Formal Semantics

Stage

• restart and block, or
• Pipeline
  - succession of group, match, and stream

Program execution \( E \vdash p \rightarrow E' \)
Stage elaboration \( E, p \vdash p' \downarrow E' \)
Stage execution \( E \vdash p \rightarrow E' \)

\[
\frac{E, \varnothing \vdash p \downarrow E'}{E \vdash p \rightarrow E'} \quad \frac{E, p :: \text{match} \{ \text{patterns} \} \vdash p' \downarrow E'}{E, p \vdash \text{match} \{ \text{patterns} \} p' \downarrow E'}
\]

\[
\frac{E \vdash p \rightarrow E'}{E, p \vdash \varnothing \downarrow E'} \quad \frac{E \vdash p \rightarrow E' \quad E' \vdash p' \rightarrow E''}{E, p \vdash \text{restart} p' \downarrow E''}
\]
Implementation

Java 8 and Javascript
• Popular programming language
• Nashorn runtime in the JVM
• Fast prototyping

Two execution modes
• Monitoring: on-the fly processing
• Troubleshooting: table processing

Closely follows the semantics
1) Build stages (block and pipelines)
2) Execute stages
Handle environment with JavaScript closures
Execution

Each section returns a list of continuations

```plaintext
lens f(x) {
    stream {a1(x)}
    {a2(x)}
    stream {a3(x)}
}

stream {p1}
run f(42)
stream {p2}
```
Execution

Each section returns a list of continuations

```javascript
lens f(x) {
    stream {a1(x)}
    {a2(x)}
    stream {a3(x)}
}

stream {p1}
run f(42)
stream {p2}
```
Execution

Each section returns a list of continuations

```plaintext
lens f(x) {
    stream {a1(x)}
    {a2(x)}
    stream {a3(x)}
}
stream {p1}
run f(42)
stream {p2}
```
Execution

Each section returns a list of continuations

```plaintext
lens f(x) {
    stream {a1(x)}
    {a2(x)}
    stream {a3(x)}
}
stream {p1}
run f(42)
stream {p2}
```
Execution

Each section returns a list of continuations

```plaintext
lens f(x) {
  stream {a1(x)}
  {a2(x)}
  stream {a3(x)}
}

stream {p1}
run f(42)
stream {p2}
```
Execution

Each section returns a list of continuations

```
len(x) {
  stream {a1(x)}
  {a2(x)}
  stream {a3(x)}
}

stream {p1}
run f(42)
stream {p2}
```
Each section returns a list of continuations

```
lens f(x) {
    stream {a1(x)}
    {a2(x)}
    stream {a3(x)}
}

stream {p1}
run f(42)
stream {p2}
```
Execution

Each section returns a list of continuations

```
lens f(x) {
    stream {a1(x)}
    {a2(x)}
    stream {a3(x)}
}

stream {p1}
run f(42)
stream {p2}
```

```
script  lens
f(x)    |
        |
stream  run
p1      |
        |
stream  block
a1(42)  |
        |
stream  |
        |
a2(42)  |
        |
```
Execution

Each section returns a list of continuations

```
lens f(x) {
    stream {a1(x)}
    {a2(x)}
    stream {a3(x)}
}

stream {p1}
run f(42)
stream {p2}
```
Execution

Each section returns a list of continuations

```
lens f(x) {
    stream {a1(x)}
    {a2(x)}
    stream {a3(x)}
}
stream {p1}
run f(42)
stream {p2}
```
Execution

Each section returns a list of continuations

```
lens f(x) {
    stream {a1(x)}
    {a2(x)}
    stream {a3(x)}
}

stream {p1}
run f(42)
stream {p2}
```
Execution

Each section returns a list of continuations

```plaintext
lens f(x) {
    stream {a1(x)}
    {a2(x)}
    stream {a3(x)}
}

stream {p1}
run f(42)
stream {p2}
```

![Diagram showing the execution flow and stages](diagram.png)
Connections

Logical time

one step = one line of log
lens and run: concurrency
syntactic scheduling

stream (entry) when ( condition )
Connections

Logical time

one step = one line of code
lens and run: concurrency
syntactic scheduling

Reactive domains [MPP15]

group to define sub-streams
lens execution over sub-streams

[MPP15] Mandel, Pasteur, Pouzet
Time Refinement in a Functional Synchronous Language
Applications

IBM Bluemix OpenWhisk

Logs from Travis
CloudLens as a whisk action...
...to analyze whisk builds
Applications

IBM Bluemix OpenWhisk
Logs from Travis
CloudLens as a whisk action...
...to analyze whisk builds
Applications

IBM Bluemix OpenWhisk

Logs from Travis

CloudLens as a whisk action...
...to analyze whisk builds

The build for PR #1023 failed
Failed tests

system.basic.WskBasicNodeTests

Powered by CloudLens & OpenWhisk
Applications

IBM Bluemix OpenWhisk
Logs from Travis
CloudLens as a whisk action...
...to analyze whisk builds
Applications

IBM Bluemix OpenWhisk

Logs from Travis
CloudLens as a whisk action...
...to analyze whisk builds
Applications

IBM Bluemix OpenWhisk
Logs from Travis
CloudLens as a whisk action...
...to analyze whisk builds

Review code repositories
Comments analysis
Applications

IBM Bluemix OpenWhisk
Logs from Travis
CloudLens as a whisk action...
...to analyze whisk builds

Review code repositories
Comments analysis

Notebook integration
CloudLens, un langage de script pour l'analyse de données semi-structurées

http://cloudlens.github.io