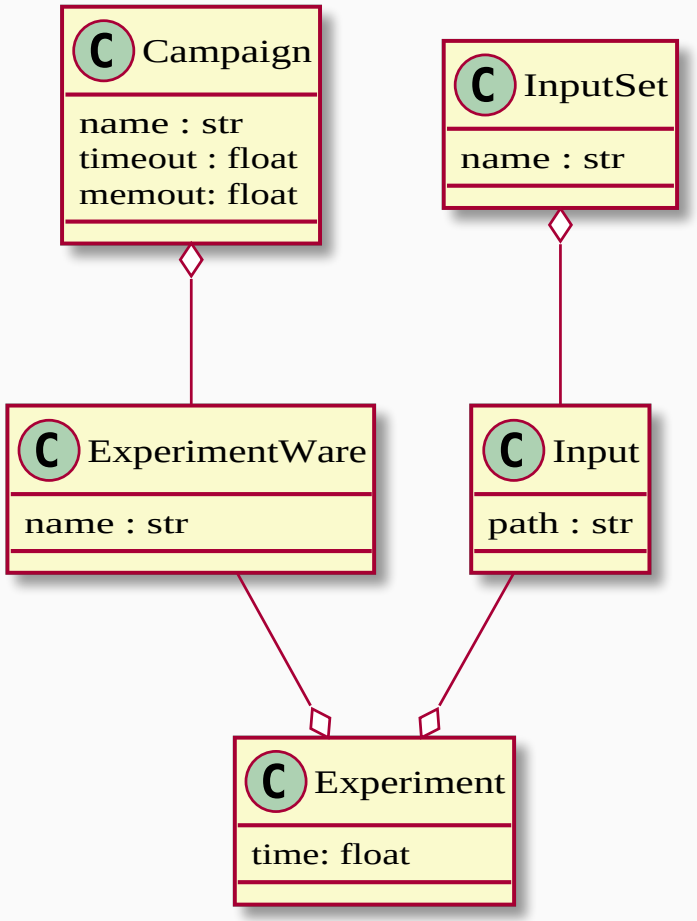
Metrics: A Unified Library for Experimenting Solvers

Thibault Falque[1], Romain Wallon[2], Hugues Wattez[2]

Séminaire au CRIL – 17 et 24 septembre 2020

[1] Exakis Nelite
[2] CRIL, Univ Artois & CNRS

metrics-core



Example

Example

Let us consider a *campaign* in which we would like to compare the two solvers *Sat4j* and *Glucose*. These solvers are our *experimentwares*. Suppose that we want to compare them on two *inputs*: a sudoku instance `sudoku.cnf` and a pigeonhole problem `pigeonhole.cnf`. The *input-set* we consider is composed of these two instances. The *experiments* of this campaign are thus:

- the execution of *Sat4j* on `sudoku.cnf`,
- the execution of *Sat4j* on `pigeonhole.cnf`,
- the execution of *Glucose* on `sudoku.cnf`, and
- the execution of *Glucose* on `pigeonhole.cnf`.

extraCt dAta of exPeriments from softwarE Logs

metrics-scalpel - YAML Configuration (Experimental Setup)

```yaml
name: My Awesome Campaign
date: 2020/09/17

setup:
  os: Linux CentOS 7 (x86_64)
  cpu: Intel XEON X5550
  ram: 32GB
  timeout: 1200
  memout: 16384
```

You must specify `name`, `timeout` and `memout`

`metrics-scalpel` - YAML Configuration (Experiment-Wares)

You may manually list the experiment-wares used for your experiments

```yaml
experiment-wares:
  - my-awesome-solver
  - ...
```

You may also let `scalpel` retrieve them, if you do not need to collect additional data

metrics-scalpel - YAML Configuration (Inputs)

A first approach for listing the benchmarks is to list them in the YAML

```yaml
input-set:
  name: My Input Set
  family: -3
  input-name: -1
  type: file-list
  path-list:
      - /path/to/my/benchmarks/of/family-a/sat/toto.cnf
      - ...
      - /path/to/my/benchmarks/of/family-n/unsat/titi.cnf
```

metrics-scalpel - YAML Configuration (Inputs)

You may also retrieve the benchmarks from a file hierarchy containing these benchmarks

```yaml
input-set:
  name: My Input Set
  family: -2
  input-name: -1
  type: hierarchy
  extensions:
    - .cnf.xz
    - .cnf.bz2
    - .cnf
  path-list:
      - /path/to/my/benchmarks
```

`metrics-scalpel` - YAML Configuration (Inputs)

You may also let `scalpel` retrieve the inputs, if you do not need to collect additional data

`metrics-scalpel` - YAML Configuration (Source)

`scalpel` may parse a wide variety of source files to retrieve experimental data, such as CSV files

```yaml
source:
  path: /path/to/my/file.csv
```

`metrics-scalpel` - YAML Configuration (Source)

`scalpel` may parse a wide variety of source files to retrieve experimental data, such as "evaluation" files

```yaml
source:
  format: evaluation
  path: /path/to/my/file.txt
```

`metrics-scalpel` - YAML Configuration (Source)

`scalpel` may parse a wide variety of source files to retrieve experimental data, even log files!

```yaml
source:
  format: deep-dir
  path: /path/to/root/directory/of/xp
  hierarchy-depth: 2
  experiment-ware: 1
```

An example of hierarchy described by the configuration above is following

```
- xp
    - my-solver-a
        - output-on-instance-1
            - ...
        - ...
    - ...
```

metrics-scalpel - YAML Configuration (Source)

scalpel may parse a wide variety of source files to retrieve experimental data, even log files!

```yaml
source:
  format: flat-dir
  path: /path/to/root/directory/of/xp
```

An example of hierarchy described by the configuration above is following

```
- xp
    - output-of-solver-a-on-instance-1
    - ...
    - output-of-solver-b-on-instance-n
```

`metrics-scalpel` - YAML Configuration (Source)

If `scalpel` does not allow to parse your files, you may implement your own parser, and tell `scalpel` to use it.

```yaml
source:
  path: /path/to/my/very/specific/file
  parser: my.own.parser
```

metrics-scalpel - YAML Configuration (Data)

If your data (especially, from a CSV file) do not follow scalpel's naming convention, you may map the names of your file to the identifiers recognized by scalpel

```yaml
data:
  mapping:
    experiment_ware:
      - solver
      - configuration
    cpu_time:
      - solver time
    input:
      - benchmark
```

metrics-scalpel - YAML Configuration (Raw Data)

You can describe how to extract data from log files as follows

```yaml
data:
  raw-data:
    - log-data: memory
      file: mysolver.log
      regex: "c Memory usage: (\d+) Mo"
      group: 1
    - log-data: cpu_time
      file: mysolver.log
      pattern: "c CPU time: {real} seconds"
```

A log file could have the following form

```
c This is an example of log file from my awesome solver
c
c Memory usage: 3000 Mo
c CPU time: 12.34 seconds
```

`metrics-scalpel` - YAML Configuration (Data Files)

If your solver has produced files using common formats, `scalpel` can parse them without having to describe them

```yaml
data:
  data-files:
    - output.json
    - output.csv
```

Previously...

With `metrics-scalpel`, it is possible to extract data:

- from CSV files
- from evaluation files
- from solver log files

Since last week...

We took your remarks into account, and have implemented some new features:

- Boolean values are supported as simplified patterns (`true` and `false`, case insensitive)
- Extraction of the name of the solver and input from the name of the file being parsed
- Parsing of (multiple) custom CSV files, with header or not

We are currently working on other new features:

- Parsing of multiple files having the same name but different extensions (almost done)
- Extraction of multiple data on the same line (almost done)
- Exploration of file hierarchy with arbitrary depth

Today...

Let us talk about figures!

Automated tooL for expLoiting Experimental resulTs

- Static Plot: matplotlib library
- Dynamic Plot: Plotly library

Demonstration

Because demonstration is better than words!

Already Available!

Install `metrics` with `pip`

```
$ pip install crillab-metrics
```

Already Available!

Install `metrics` with `pip`

```
$ pip install crillab-metrics
```

Download the source code from GitHub

https://github.com/crillab/metrics/

Metrics: A Unified Library for Experimenting Solvers

Thibault Falque[1], Romain Wallon[2], Hugues Wattez[2]

Séminaire au CRIL – 17 et 24 septembre 2020

[1] Exakis Nelite
[2] CRIL, Univ Artois & CNRS