UFR IM²AG

UNIVERSITÉ Grenoble Alpes

A short introduction to ParTraP

Yves Ledru

Université Grenoble Alpes

Laboratoire d'Informatique de Grenoble

Yves.Ledru@imag.fr

2020

UFR IM²AG

UNIVERSITÉ Grenoble Alpes

ParTraP introduction

- ParTraP is a language for the expression of properties on event traces.
- Traces of software systems are easy to produce manually, or using a logging framework such as log4J or java.util.logging.
- As traces get long, it becomes mandatory to use automated tools, such as ParTraP to analyse them.

UFR IM²AG

UNIVERSITÉ Grenoble Alpes

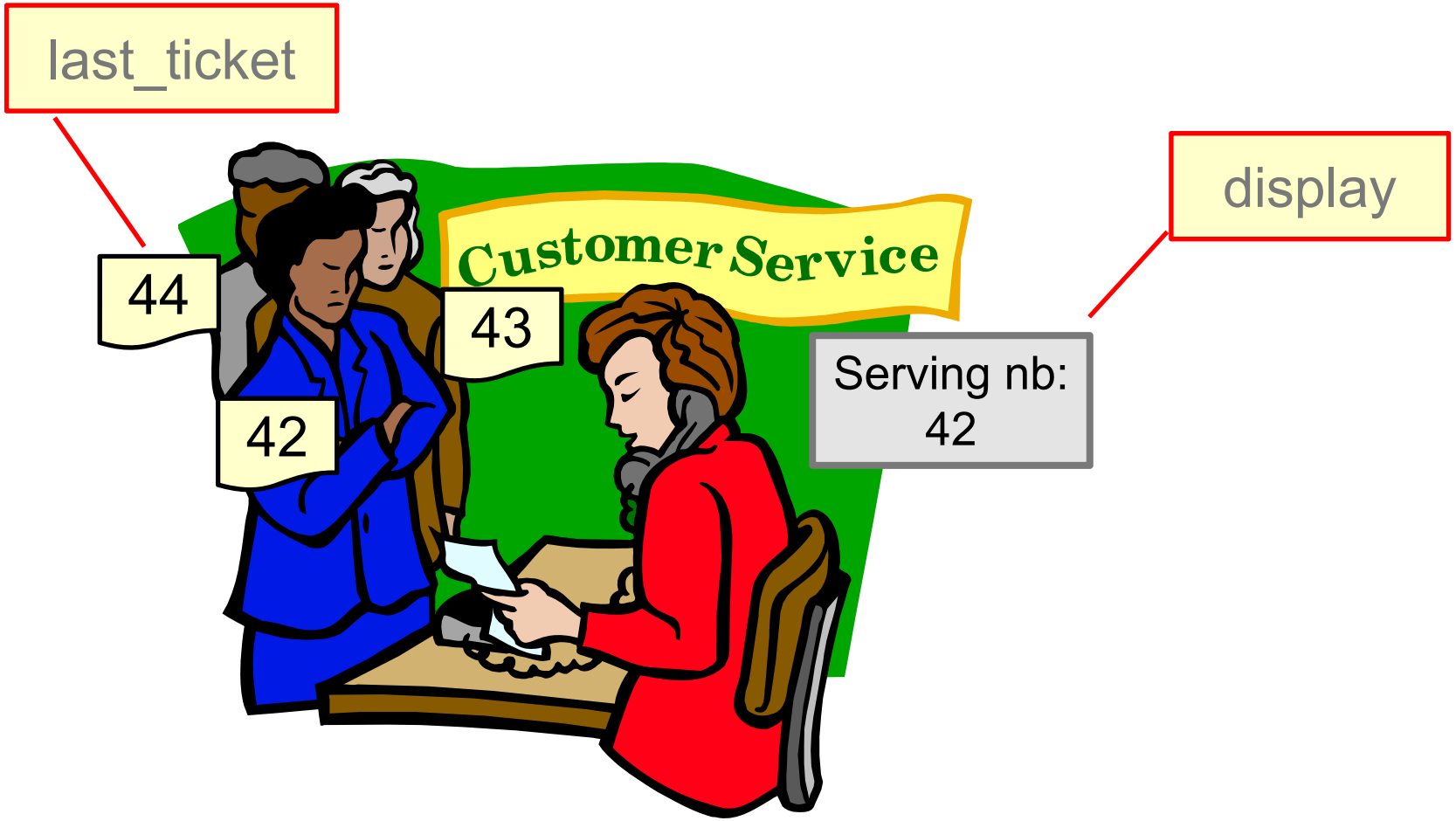
LOGGING

UFR IM²AG

UNIVERSITÉ Grenoble Alpes

Example : Take a Number

- The following slides use a trace of a ticketing system.



UFR IM²AG

UNIVERSITÉ Grenoble Alpes

JSON traces

- ParTraP verifies properties on JSON traces.
- The following trace logs Entry and Return events of methods take_ticket and serve_next.


```
[
{"time":1550229599836,"level":"INFO","sourceClass":"TANMain2","id":"Starting","arg":" a run"}
,{"time":1550229599878,"level":"FINER","sourceClass":"TakeANumber","id":"take_ticket_ENT"}
,{"time":1550229599878,"level":"FINER","sourceClass":"TakeANumber","param0" : 1,"id":"take_ticket_RET"}
,{"time":1550229599879,"level":"FINER","sourceClass":"TakeANumber","id":"take_ticket_ENT"}
,{"time":1550229599879,"level":"FINER","sourceClass":"TakeANumber","param0" : 2,"id":"take_ticket_RET"}
,{"time":1550229599879,"level":"FINER","sourceClass":"TakeANumber","id":"serve_next_ENT"}
,{"time":1550229599879,"level":"FINER","sourceClass":"TakeANumber","param0" : 1,"id":"serve_next_RET"}
,{"time":1550229599879,"level":"FINER","sourceClass":"TakeANumber","id":"take_ticket_ENT"}
,{"time":1550229599879,"level":"FINER","sourceClass":"TakeANumber","param0" : 3,"id":"take_ticket_RET"}
,{"time":1550229599880,"level":"FINER","sourceClass":"TakeANumber","id":"serve_next_ENT"}
,{"time":1550229599880,"level":"FINER","sourceClass":"TakeANumber","param0" : 2,"id":"serve_next_RET"}
,{"time":1550229599882,"level":"FINER","sourceClass":"TakeANumber","id":"serve_next_ENT"}
,{"time":1550229599882,"level":"FINER","sourceClass":"TakeANumber","param0" : 3,"id":"serve_next_RET"}
,{"time":1550229599882,"level":"FINER","sourceClass":"TakeANumber","id":"serve_next_ENT"}
,{"time":1550229599882,"level":"FINER","sourceClass":"TakeANumber","param0" : 4,"id":"serve_next_RET"}
]
```

UFR IM²AG

UNIVERSITÉ Grenoble Alpes

PARTRAP

UFR IM²AG

UNIVERSITÉ Grenoble Alpes

ParTraP : a language for the specification of properties of parametric traces

- In ParTraP
  - A trace is a sequence of events
  - Events have a name (« id ») and parameters
  - Traces are encoded in JSON

- ParTraP allows to express properties of traces
- Evaluating a property on a trace returns true or false

- ParTraP is available as an Eclipse plugin
  http://vasco.imag.fr/tools/partrap/

UFR IM²AG

UNIVERSITÉ Grenoble Alpes

JSON trace

- Each event is a set of parameter names associated to a value
- (the value may be structured (e.g. list or record).
- One of the parameters is the « time »
- Another parameter is « id », the type of the event


```
[
{"time":1550229599836,"level":"INFO","sourceClass":"TANMain2","id":"Starting","arg":" a run"}
,{"time":1550229599878,"level":"FINER","sourceClass":"TakeANumber","id":"take_ticket_ENT"}
,{"time":1550229599878,"level":"FINER","sourceClass":"TakeANumber","param0" : 1,"id":"take_ticket_RET"}
,{"time":1550229599879,"level":"FINER","sourceClass":"TakeANumber","id":"take_ticket_ENT"}
,{"time":1550229599879,"level":"FINER","sourceClass":"TakeANumber","param0" : 2,"id":"take_ticket_RET"}
,{"time":1550229599879,"level":"FINER","sourceClass":"TakeANumber","id":"serve_next_ENT"}
,{"time":1550229599879,"level":"FINER","sourceClass":"TakeANumber","param0" : 1,"id":"serve_next_RET"}
,{"time":1550229599879,"level":"FINER","sourceClass":"TakeANumber","id":"take_ticket_ENT"}
,{"time":1550229599879,"level":"FINER","sourceClass":"TakeANumber","param0" : 3,"id":"take_ticket_RET"}
,{"time":1550229599880,"level":"FINER","sourceClass":"TakeANumber","id":"serve_next_ENT"}
,{"time":1550229599880,"level":"FINER","sourceClass":"TakeANumber","param0" : 2,"id":"serve_next_RET"}
,{"time":1550229599882,"level":"FINER","sourceClass":"TakeANumber","id":"serve_next_ENT"}
,{"time":1550229599882,"level":"FINER","sourceClass":"TakeANumber","param0" : 3,"id":"serve_next_RET"}
,{"time":1550229599882,"level":"FINER","sourceClass":"TakeANumber","id":"serve_next_ENT"}
,{"time":1550229599882,"level":"FINER","sourceClass":"TakeANumber","param0" : 4,"id":"serve_next_RET"}
]
```

UFR IM²AG

UNIVERSITÉ Grenoble Alpes

Simplified trace

```
00 [
01  {"id":"Starting","arg":" a run"}
02 ,{"id":"take_ticket_ENT"}
03 ,{"param0" : 1,"id":"take_ticket_RET"}
04 ,{"id":"take_ticket_ENT"}
05 ,{"param0" : 2,"id":"take_ticket_RET"}
06 ,{"id":"serve_next_ENT"}
07 ,{"param0" : 1,"id":"serve_next_RET"}
08 ,{"id":"take_ticket_ENT"}
09 ,{"param0" : 3,"id":"take_ticket_RET"}
10 ,{"id":"serve_next_ENT"}
11 ,{"param0" : 2,"id":"serve_next_RET"}
12 ,{"id":"serve_next_ENT"}
13 ,{"param0" : 3,"id":"serve_next_RET"}
14 ,{"id":"serve_next_ENT"}
15 ,{"param0" : 4,"id":"serve_next_RET"}
16 ]
```

UFR IM²AG

UNIVERSITÉ Grenoble Alpes

ParTraP unary properties

- Absence_of <event>
  is true if the event is not present in the trace
- Occurrence_of <event>
  is true if the event is present in the trace

- Where <event> ::= event_name [ident [where <expr>]]
  An event is characterized by a name, it can also introduce a local identifier and a boolean expression involving this identifier, which must be satisfied by the event in the trace.

Examples of unary properties (1)

occurrence_of take_ticket_ENT;

Is satisfied by the trace (events 2, 4, 8)

The property means that there is at least one event

In the trace whose name is take_ticket_ENT.

```
00 [
01  {"id":"Starting","arg":" a run"}
02 ,{"id":"take_ticket_ENT"}
03 ,{"param0" : 1,"id":"take_ticket_RET"}
04 ,{"id":"take_ticket_ENT"}
05 ,{"param0" : 2,"id":"take_ticket_RET"}
06 ,{"id":"serve_next_ENT"}
07 ,{"param0" : 1,"id":"serve_next_RET"}
08 ,{"id":"take_ticket_ENT"}
09 ,{"param0" : 3,"id":"take_ticket_RET"}
10 ,{"id":"serve_next_ENT"}
11 ,{"param0" : 2,"id":"serve_next_RET"}
12 ,{"id":"serve_next_ENT"}
13 ,{"param0" : 3,"id":"serve_next_RET"}
14 ,{"id":"serve_next_ENT"}
15 ,{"param0" : 4,"id":"serve_next_RET"}
16 ]
```

Examples of unary properties (2)

absence_of take_ticket_RET tic where tic.param0 <= 0;

Is satisfied by the trace

```
00 [
01  {"id":"Starting","arg":" a run"}
02 ,{"id":"take_ticket_ENT"}
03 ,{"param0" : 1,"id":"take_ticket_RET"}
04 ,{"id":"take_ticket_ENT"}
05 ,{"param0" : 2,"id":"take_ticket_RET"}
06 ,{"id":"serve_next_ENT"}
07 ,{"param0" : 1,"id":"serve_next_RET"}
08 ,{"id":"take_ticket_ENT"}
09 ,{"param0" : 3,"id":"take_ticket_RET"}
10 ,{"id":"serve_next_ENT"}
11 ,{"param0" : 2,"id":"serve_next_RET"}
12 ,{"id":"serve_next_ENT"}
13 ,{"param0" : 3,"id":"serve_next_RET"}
14 ,{"id":"serve_next_ENT"}
15 ,{"param0" : 4,"id":"serve_next_RET"}
16 ]
```

The property means that there is no event named take_ticket_RET whose parameter param0 is a negative value.

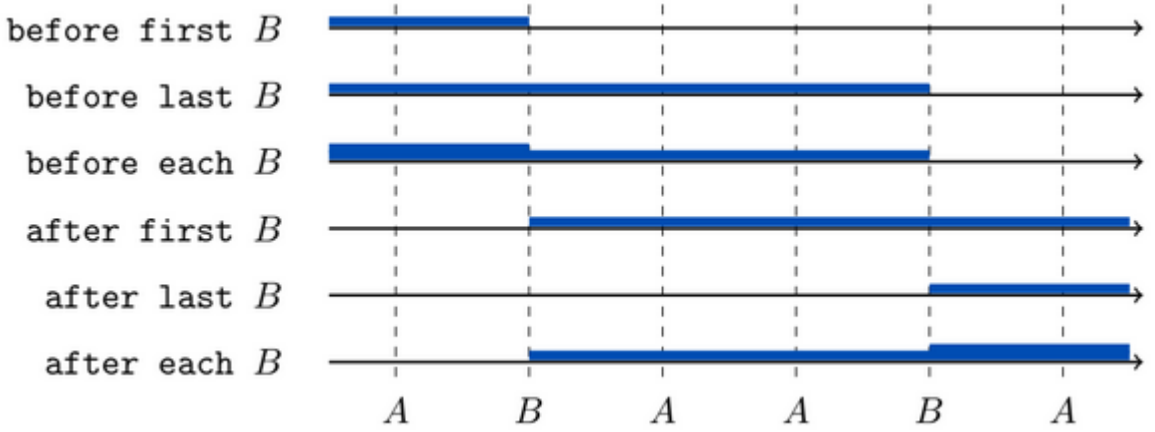
Absence_of is useful to express invariant properties

UFR IM²AG

UNIVERSITÉ Grenoble Alpes

Unary scopes

- Scopes define a subtrace where the property will be evaluated.
- There are six unary scopes :
  - after first , after last , after each
  - before first , before last , before each
- « each » leads to several subtraces
- The event which delimitates the scope is not included in the scope



UFR IM²AG

UNIVERSITÉ Grenoble Alpes

Examples of unary scopes (1)

after first Starting, occurrence_of take_ticket_RET

After the first event named Starting, there exists at least one event take_ticket_RET

It is satisfied by lines 01 and 03.

```
00 [
01  {"id":"Starting","arg":" a run"}
02 ,{"id":"take_ticket_ENT"}
03 ,{"param0" : 1,"id":"take_ticket_RET"}
04 ,{"id":"take_ticket_ENT"}
05 ,{"param0" : 2,"id":"take_ticket_RET"}
06 ,{"id":"serve_next_ENT"}
07 ,{"param0" : 1,"id":"serve_next_RET"}
08 ,{"id":"take_ticket_ENT"}
09 ,{"param0" : 3,"id":"take_ticket_RET"}
10 ,{"id":"serve_next_ENT"}
11 ,{"param0" : 2,"id":"serve_next_RET"}
12 ,{"id":"serve_next_ENT"}
13 ,{"param0" : 3,"id":"serve_next_RET"}
14 ,{"id":"serve_next_ENT"}
15 ,{"param0" : 4,"id":"serve_next_RET"}
16 ]
```

UFR IM²AG

UNIVERSITÉ Grenoble Alpes

Examples of unary scopes (2)

after each take_ticket_RET ret,
absence_of take_ticket_RET ret2
where ret.param0 >= ret2.param0;

The ticket numbers are strictly ascending.

In other words, the param0 of each take_ticket_RET event, is not greater than the param0 of subsequent take_ticket_RET events

This scope applies to three subtraces

```
00 [
01  {"id":"Starting","arg":" a run"}
02 ,{"id":"take_ticket_ENT"}
03 ,{"param0" : 1,"id":"take_ticket_RET"}
04 ,{"id":"take_ticket_ENT"}
05 ,{"param0" : 2,"id":"take_ticket_RET"}
06 ,{"id":"serve_next_ENT"}
07 ,{"param0" : 1,"id":"serve_next_RET"}
08 ,{"id":"take_ticket_ENT"}
09 ,{"param0" : 3,"id":"take_ticket_RET"}
10 ,{"id":"serve_next_ENT"}
```

UFR IM²AG

UNIVERSITÉ Grenoble Alpes

Absence of events in scope

- When the event which delimits the scope is absent, the property is always true.

- For example

after first Start, occurrence_of take_ticket_RET

- If there is no event named Start the property is true.

UFR IM²AG

UNIVERSITÉ Grenoble Alpes

Unary scopes can be nested

```
after each take_ticket_RET r,
    after each take_ticket_RET r2,
      before first serve_next_RET ss where ss.param0==r.param0,
        absence_of serve_next_RET s2 where s2.param0==r2.param0;
```

This property means that ticket r2 was taken after ticket r, so it should be served after it.

Here the property expresses the absence of a serve which would satisfy s2 (same param as r2) before ss .

UFR IM²AG

UNIVERSITÉ Grenoble Alpes

Properties can be connected by boolean operators

- <prop> and <prop> , <prop> or <prop> ,
  <prop> equiv <prop> , <prop> implies <prop>
- Examples:

after first Starting,
(occurrence_of take_ticket_RET)
and
(occurrence_of serve_next_RET) ;

(after first Starting, occurrence_of take_ticket_RET)
and
(after first take_ticket_RET, occurrence_of serve_next_RET);

UFR IM²AG

UNIVERSITÉ Grenoble Alpes

Additional operators

- They are build from combinations of the above operators.
  - Properties:
    - A followed_by B
    - B preceded_by A
    - A prevents B
  - Scopes:
    - Between A and B
    - Since A until B

UFR IM²AG

UNIVERSITÉ Grenoble Alpes

Time variants, Python

- The timestamps of the event can be taken into account by specialized versions of the operators

- For example:
  Within 2ms before each A, absence_of B

- Properties appearing in a where clause can be expressed in Python, a language familiar to numerous software engineers.