From BDD Scenarios to Test Case Generation

Combining Model-Based Testing and Behavior-Driven Development

Tannaz Zameni
PhD candidate at the University of Twente
t.zameni@utwente.nl

TiCToC

Testing in Times of Continuous Change

zetta scale

InTraffic

Malvern Panalytical
a spectris company

NWO

ESI

ASML

Canon

axini

Radboud Universiteit
IN DEI NOMINE FELICITER

UNIVERSITY OF TWENTE.

CGI

AKKODIS

ICT smarter solutions

Capgemini

TOPIC
EMBEDDED SYSTEMS

MBT

- No widespread use of MBT in the industry, yet

+ Automated, algorithmic generation of large amounts of valid test cases

+

BDD

- No underlying theory providing formal semantics

+ Collaborative exploration of the requirements Structured, readable scenarios in natural language

An Example

A controller job of type production is moved to the printed jobs the moment printing completes

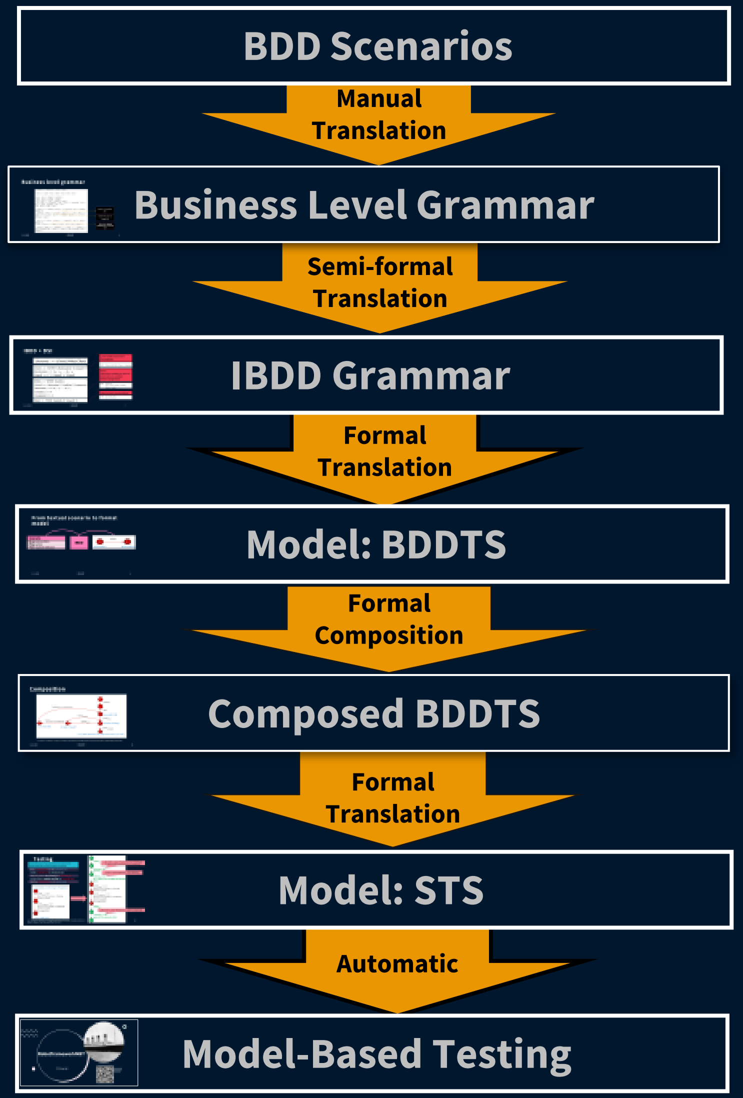
Given a **Controller job** is in the **scheduled jobs**

and the **Controller job** is a Production job

When the printer **starts printing** the **Controller job**

and the printer **completes printing** the **Controller job**

Then the **Controller job** is in the **printed jobs**



BDD Scenarios

Manual Translation

Business Level Grammar

Semi-formal Translation

IBDD Grammar

Formal Translation

Model: BDDTS

Formal Composition

Composed BDDTS

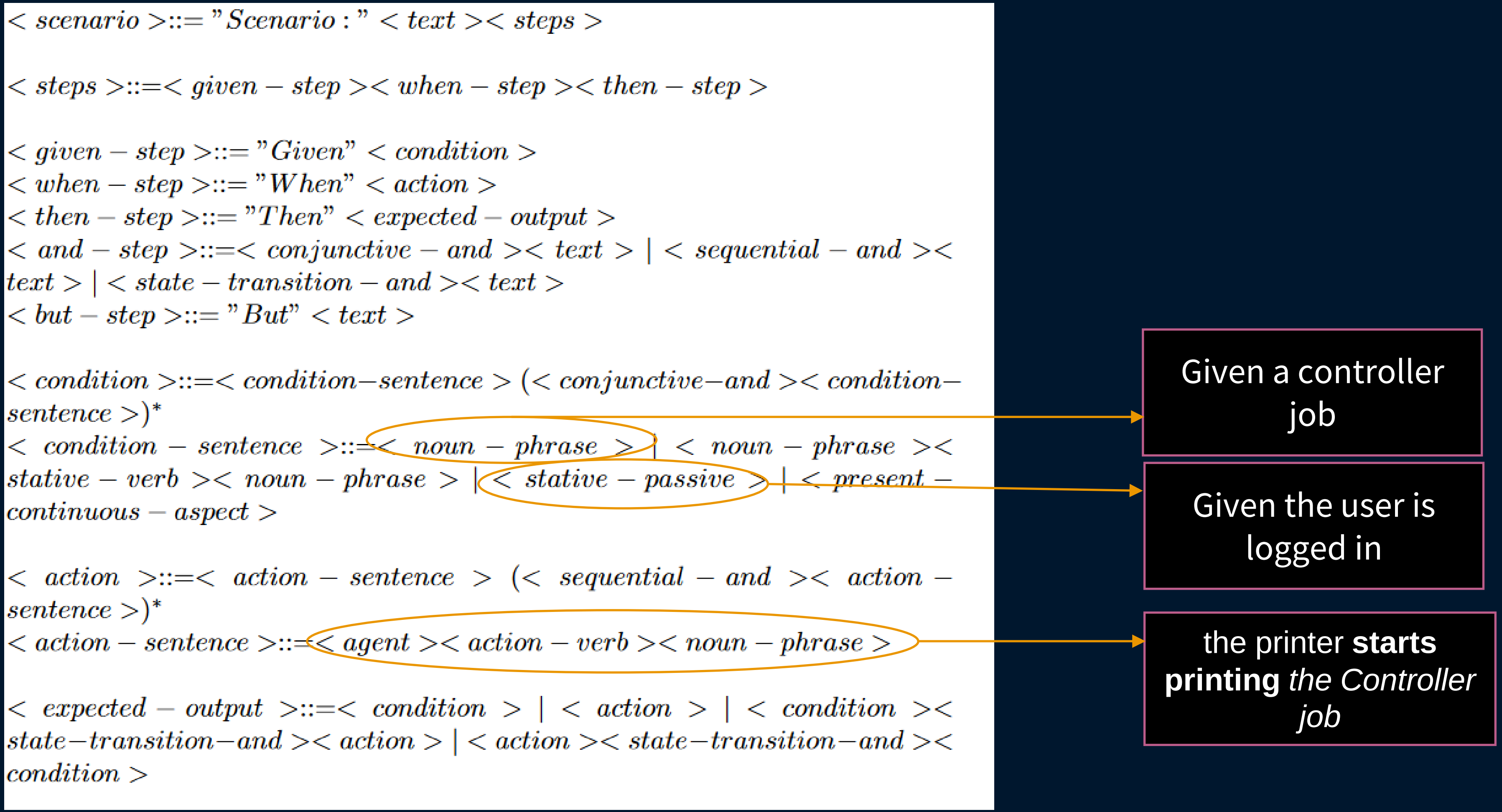
Formal Translation

Model: STS

Automatic

Model-Based Testing

Business level grammar



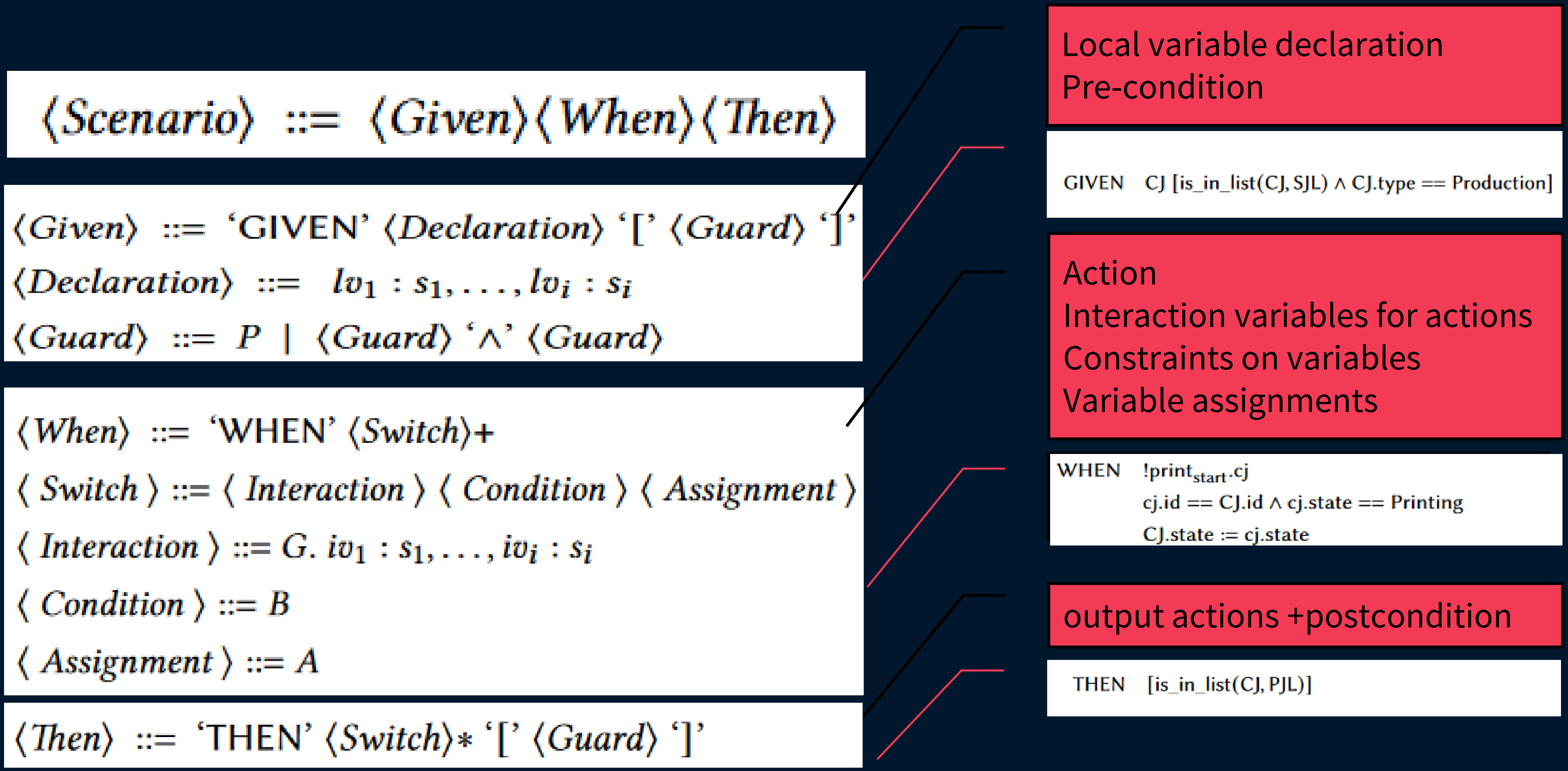
```
< scenario >::= "Scenario : " < text >< steps >

< steps >::=< given − step >< when − step >< then − step >

< given − step >::= "Given" < condition >
< when − step >::= "When" < action >
< then − step >::= "Then" < expected − output >
< and − step >::=< conjunctive − and >< text > | < sequential − and ><
text > | < state − transition − and >< text >
< but − step >::= "But" < text >

< condition >::=< condition−sentence > (< conjunctive−and >< condition−
sentence >)*
< condition − sentence >::=< noun − phrase > | < noun − phrase ><
stative − verb >< noun − phrase > | < stative − passive > | < present −
continuous − aspect >

< action >::=< action − sentence > (< sequential − and >< action −
sentence >)*
< action − sentence >::=< agent >< action − verb >< noun − phrase >

< expected − output >::=< condition > | < action > | < condition ><
state−transition−and >< action > | < action >< state−transition−and ><
condition >
```

Given a controller job

Given the user is logged in

the printer **starts printing** *the Controller job*

IBDD + DSI



$\langle \mathit{Scenario} \rangle ::= \langle \mathit{Given} \rangle \langle \mathit{When} \rangle \langle \mathit{Then} \rangle$

$\langle \mathit{Given} \rangle ::= \text{`GIVEN'}\ \langle \mathit{Declaration} \rangle\ \text{`['}\ \langle \mathit{Guard} \rangle\ \text{`]'}$

$\langle \mathit{Declaration} \rangle ::= lv_1 : s_1, \ldots, lv_i : s_i$

$\langle \mathit{Guard} \rangle ::= P \mid \langle \mathit{Guard} \rangle\ \text{`}\wedge\text{'}\ \langle \mathit{Guard} \rangle$

$\langle \mathit{When} \rangle ::= \text{`WHEN'}\ \langle \mathit{Switch} \rangle +$

$\langle \mathit{Switch} \rangle ::= \langle \mathit{Interaction} \rangle \langle \mathit{Condition} \rangle \langle \mathit{Assignment} \rangle$

$\langle \mathit{Interaction} \rangle ::= G.\ iv_1 : s_1, \ldots, iv_i : s_i$

$\langle \mathit{Condition} \rangle ::= B$

$\langle \mathit{Assignment} \rangle ::= A$

$\langle \mathit{Then} \rangle ::= \text{`THEN'}\ \langle \mathit{Switch} \rangle *\ \text{`['}\ \langle \mathit{Guard} \rangle\ \text{`]'}$

Local variable declaration
Pre-condition

GIVEN CJ [is_in_list(CJ, SJL) ∧ CJ.type == Production]

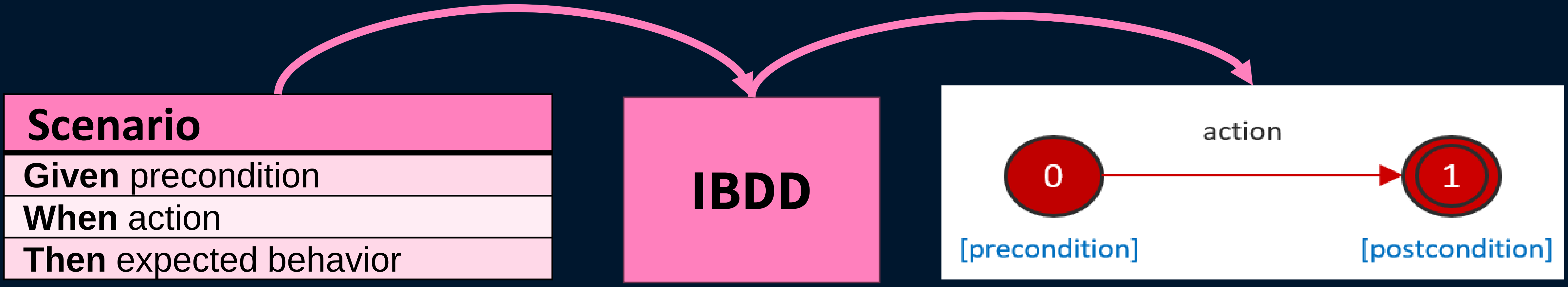
Action
Interaction variables for actions
Constraints on variables
Variable assignments

WHEN $!\mathrm{print}_{start}.\mathrm{cj}$
cj.id == CJ.id ∧ cj.state == Printing
CJ.state := cj.state

output actions +postcondition

THEN [is_in_list(CJ, PJL)]

From textual scenario to formal model



Scenario

Given precondition

When action

Then expected behavior

IBDD

A controller job of type production is moved to the printed jobs the moment printing completes

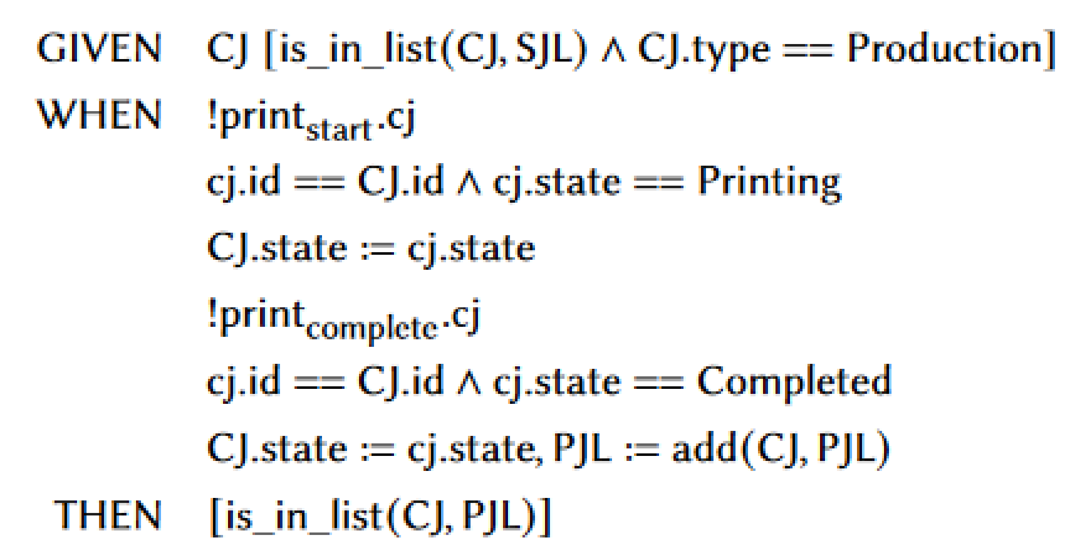
Given a Controller job is in the scheduled jobs

and the Controller job is a Production job

When the printer **starts printing** the Controller job

and the printer **completes printing** the Controller job

Then the Controller job is in the printed jobs



GIVEN CJ [is_in_list(CJ, SJL) ∧ CJ.type == Production]

WHEN $!print_{start}.cj$

cj.id == CJ.id ∧ cj.state == Printing

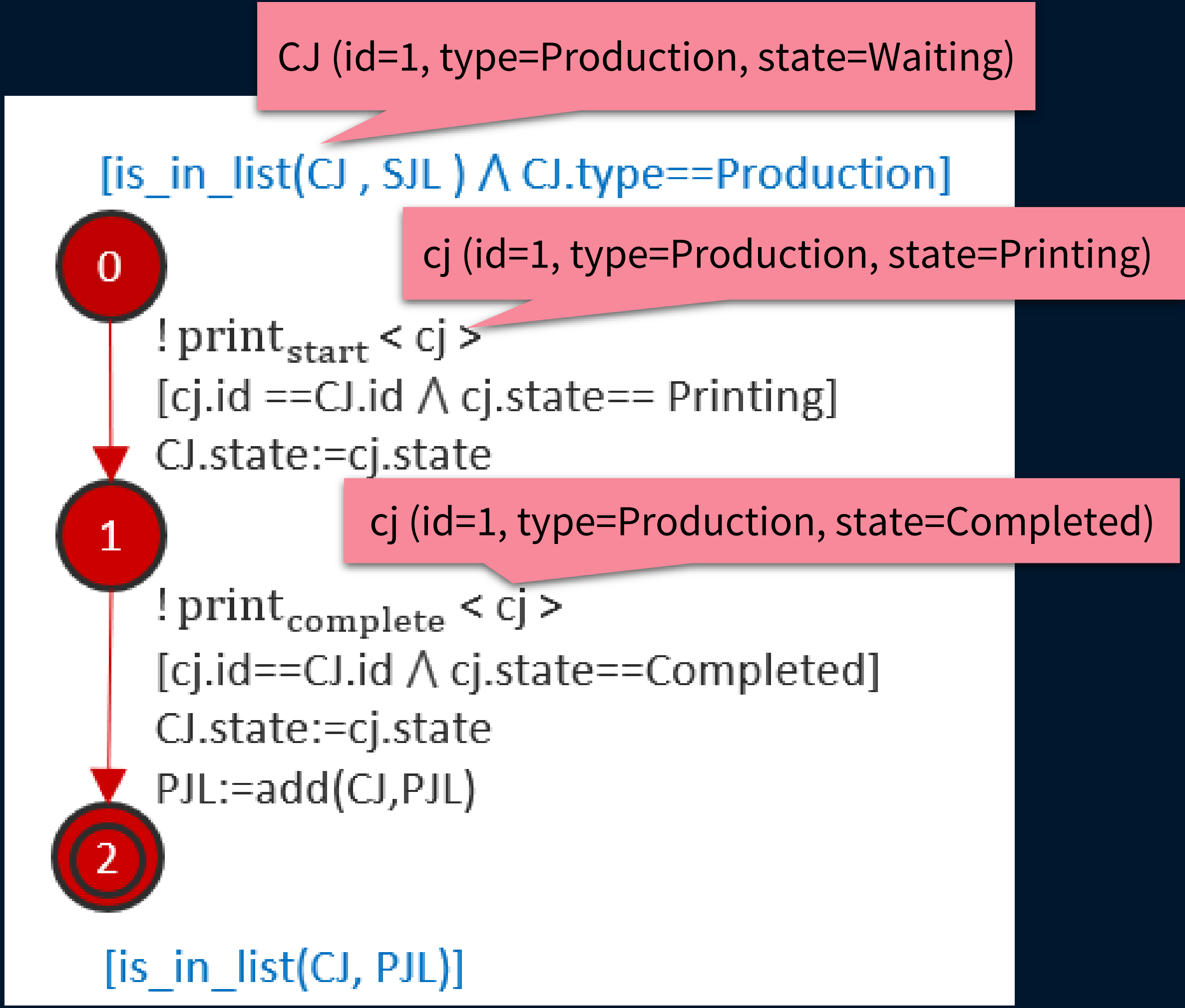
CJ.state := cj.state

$!print_{complete}.cj$

cj.id == CJ.id ∧ cj.state == Completed

CJ.state := cj.state, PJL := add(CJ, PJL)

THEN [is_in_list(CJ, PJL)]



CJ (id=1, type=Production, state=Waiting)

[is_in_list(CJ , SJL) ⋀ CJ.type==Production]

cj (id=1, type=Production, state=Printing)

$! print_{start} < cj >$

[cj.id ==CJ.id ⋀ cj.state== Printing]

CJ.state:=cj.state

cj (id=1, type=Production, state=Completed)

$! print_{complete} < cj >$

[cj.id==CJ.id ⋀ cj.state==Completed]

CJ.state:=cj.state

PJL:=add(CJ,PJL)

[is_in_list(CJ, PJL)]

1-A controller job is added to the scheduled jobs after a job is submitted

Given a **Job file**

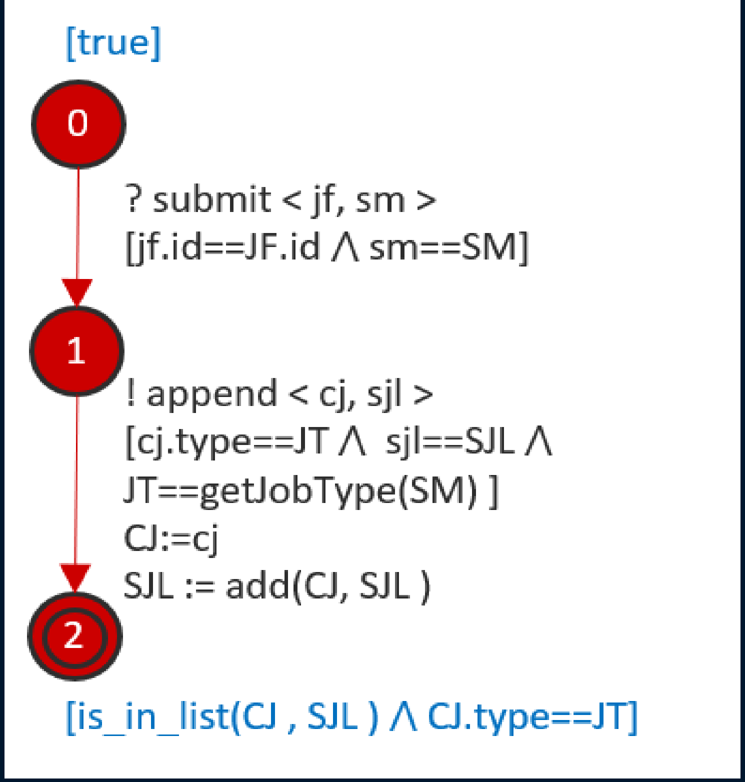
When the operator ***submits*** the Job file using a **<Submission method>**

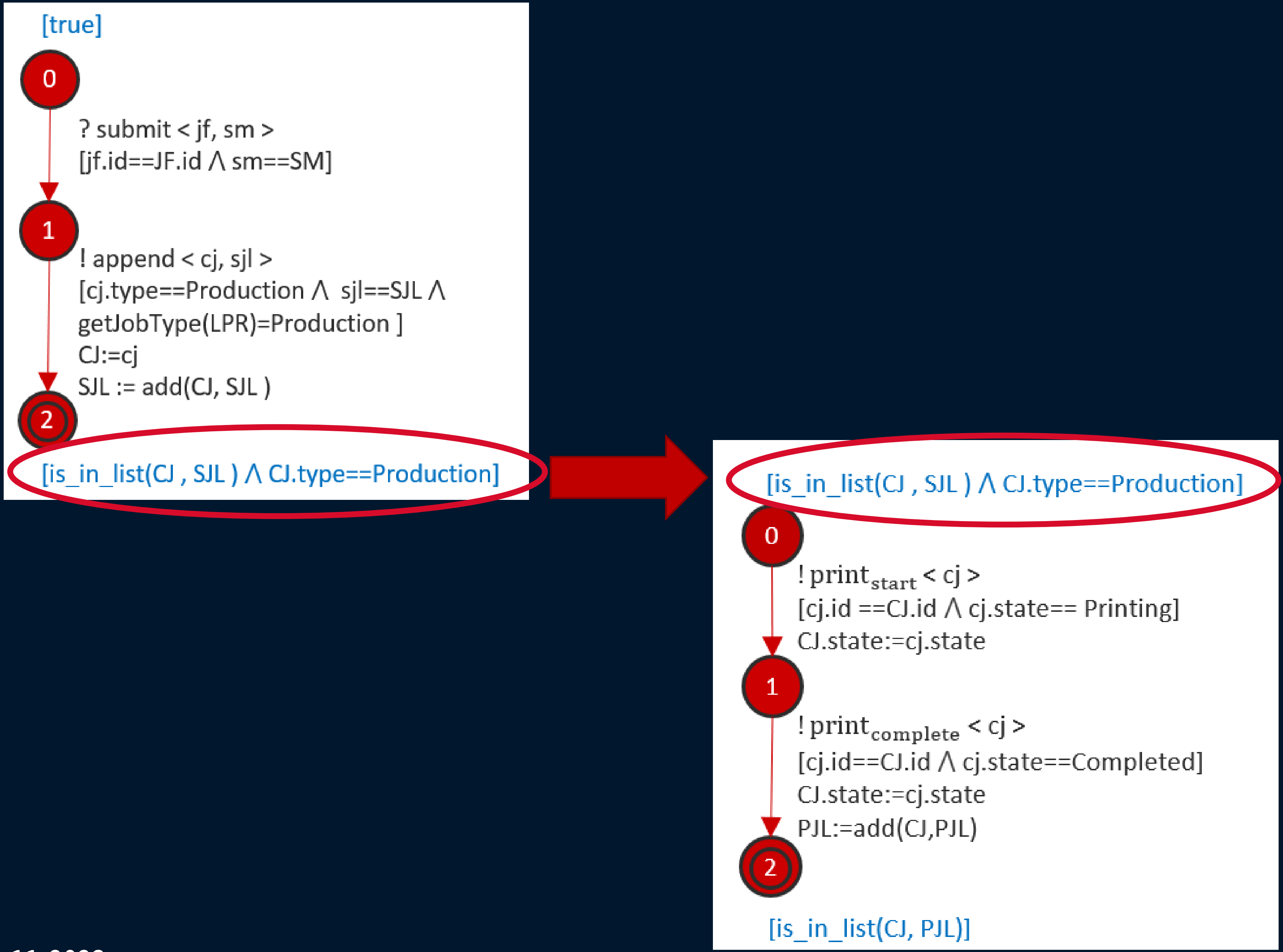
Then the printer ***adds*** a new Controller job to the scheduled jobs

and the Controller job is of type **<job type>**

Scenario

Submission method	**Job type**
LPR	Production job
IPP	Production job
JMF	Production job
Socket	Streaming job





[true]

0

? submit < jf, sm >
[jf.id==JF.id Λ sm==SM]

1

! append < cj, sjl >
[cj.type==Production Λ sjl==SJL Λ
getJobType(LPR)=Production]
CJ:=cj
SJL := add(CJ, SJL)

2

[is_in_list(CJ , SJL) Λ CJ.type==Production]

[is_in_list(CJ , SJL) Λ CJ.type==Production]

0

! $\text{print}_{\text{start}}$ < cj >
[cj.id ==CJ.id Λ cj.state== Printing]
CJ.state:=cj.state

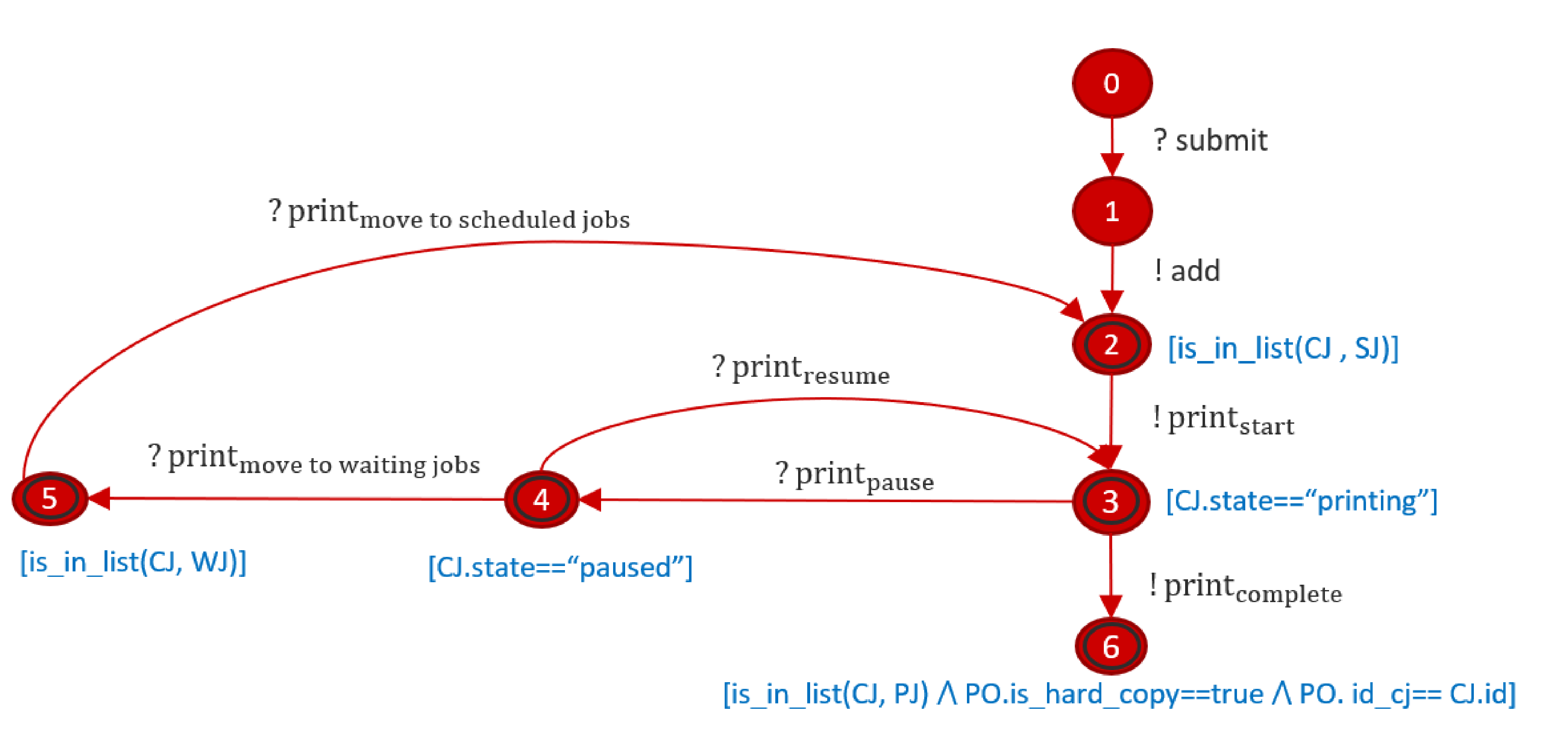
1

! $\text{print}_{\text{complete}}$ < cj >
[cj.id==CJ.id Λ cj.state==Completed]
CJ.state:=cj.state
PJL:=add(CJ,PJL)

2

[is_in_list(CJ, PJL)]

Composition


0
? submit
1
! add
2
[is_in_list(CJ , SJ)]
? print$_{\text{move to scheduled jobs}}$
? print$_{\text{resume}}$
! print$_{\text{start}}$
? print$_{\text{move to waiting jobs}}$
? print$_{\text{pause}}$
5
4
3
[CJ.state==“printing”]
[is_in_list(CJ, WJ)]
[CJ.state==“paused”]
! print$_{\text{complete}}$
6
[is_in_list(CJ, PJ) Λ PO.is_hard_copy==true Λ PO. id_cj== CJ.id]

T. Zameni, P. van Den Bos, J. Tretmans, J. Foederer and A. Rensink, "From BDD Scenarios to Test Case Generation," *(ICSTW)*, Dublin, Ireland, 2023.

Testing

A controller job of type production is moved to the printed jobs the moment printing completes

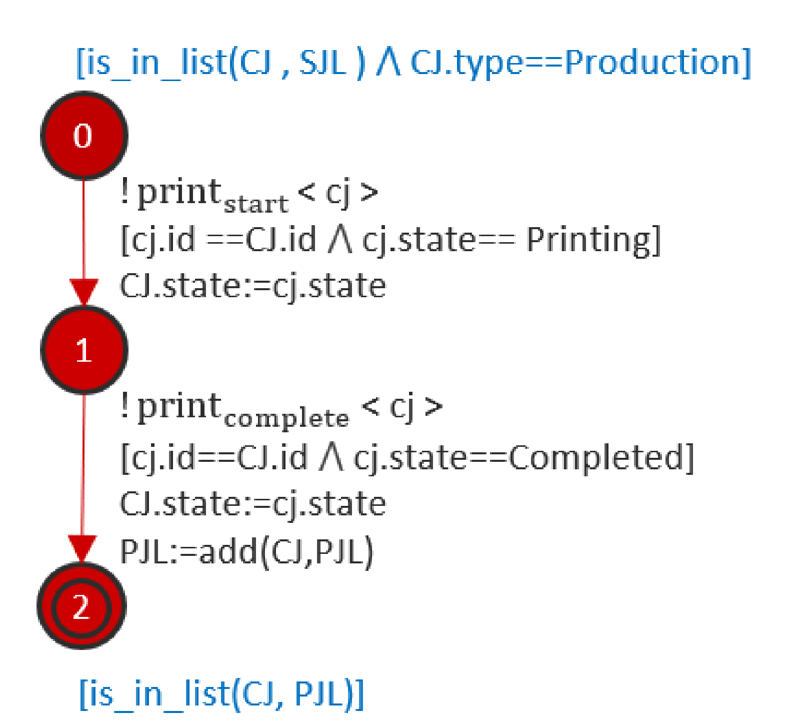
Given a Controller job is in the scheduled jobs

and the Controller job is a Production job

When the printer **starts printing** the Controller job

and the printer **completes printing** the Controller job

Then the Controller job is in the printed jobs

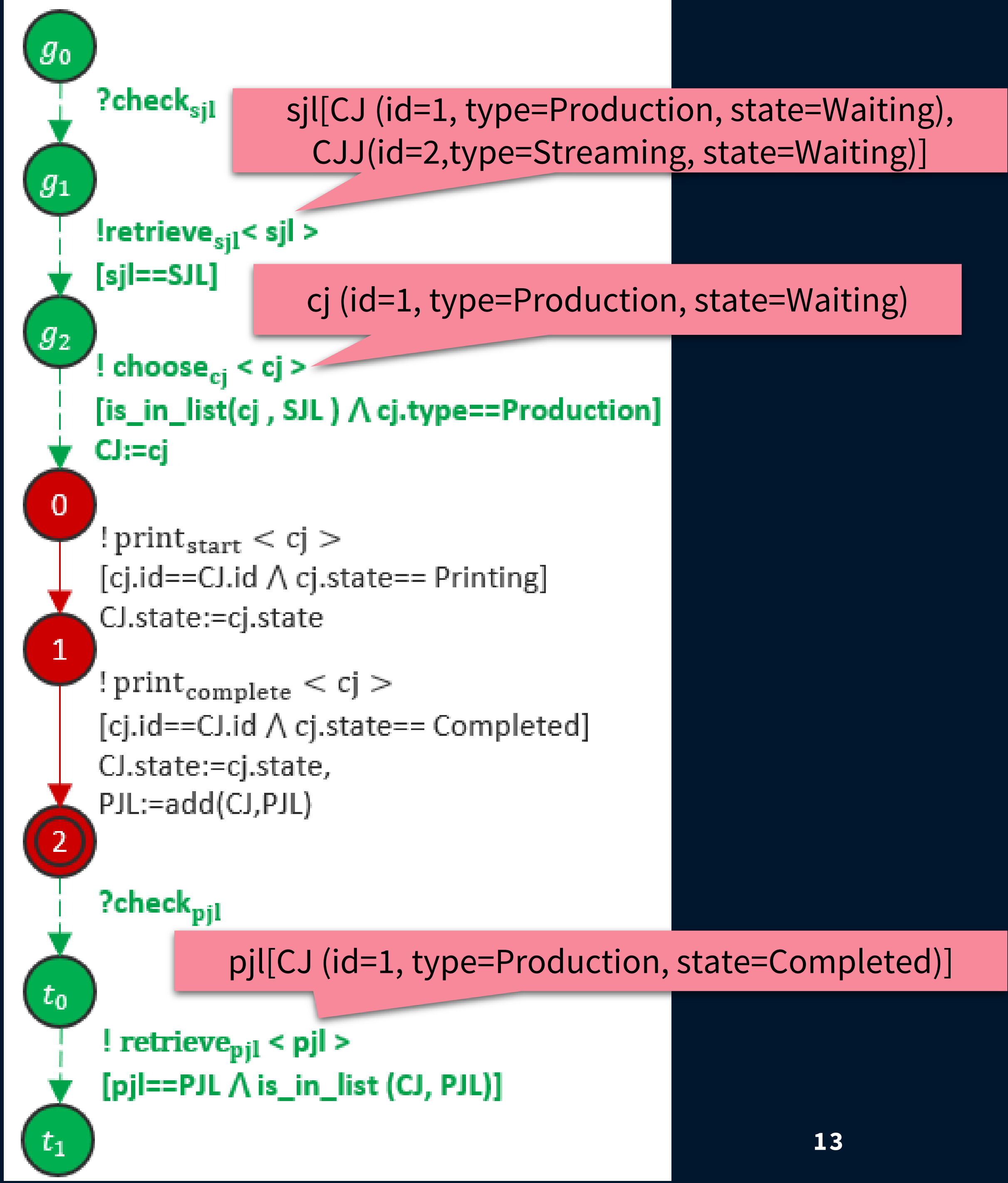


[is_in_list(CJ , SJL) ∧ CJ.type==Production]

$! \text{print}_{start}$ < cj >
[cj.id ==CJ.id ∧ cj.state== Printing]
CJ.state:=cj.state

$! \text{print}_{complete}$ < cj >
[cj.id==CJ.id ∧ cj.state==Completed]
CJ.state:=cj.state
PJL:=add(CJ,PJL)

[is_in_list(CJ, PJL)]



$?\text{check}_{sjl}$

sjl[CJ (id=1, type=Production, state=Waiting),
CJJ(id=2,type=Streaming, state=Waiting)]

$!\text{retrieve}_{sjl}$< sjl >
[sjl==SJL]

cj (id=1, type=Production, state=Waiting)

$! \text{choose}_{cj}$ < cj >
[is_in_list(cj , SJL) ∧ cj.type==Production]
CJ:=cj

$! \text{print}_{start}$ < cj >
[cj.id==CJ.id ∧ cj.state== Printing]
CJ.state:=cj.state

$! \text{print}_{complete}$ < cj >
[cj.id==CJ.id ∧ cj.state== Completed]
CJ.state:=cj.state,
PJL:=add(CJ,PJL)

$?\text{check}_{pjl}$

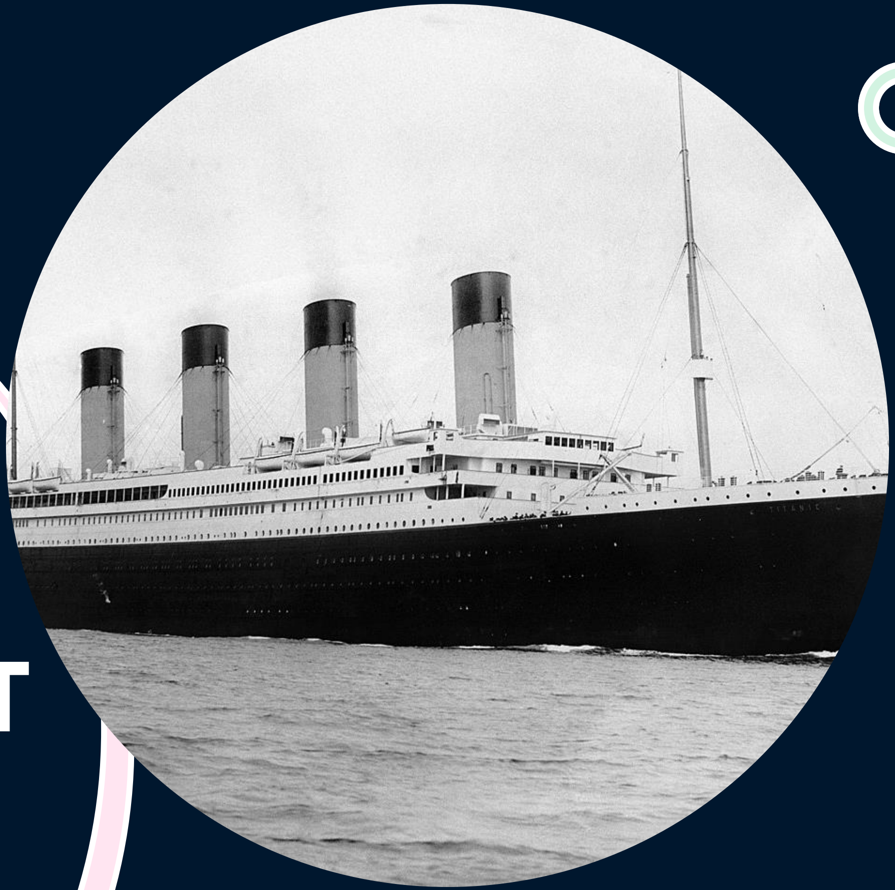
pjl[CJ (id=1, type=Production, state=Completed)]

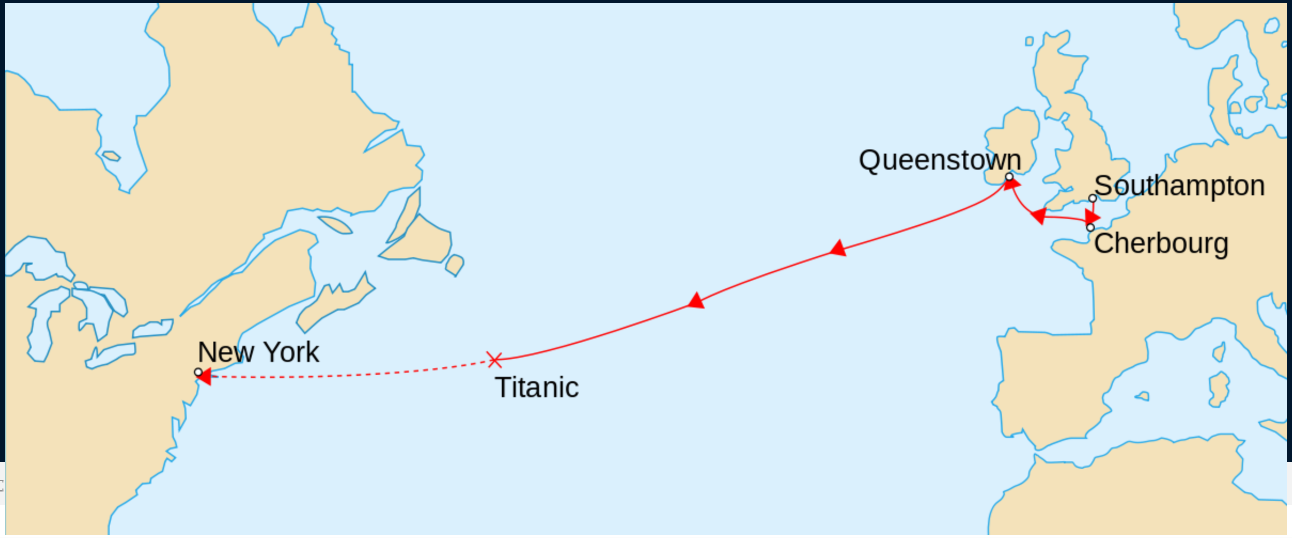
$! \text{retrieve}_{pjl}$ < pjl >
[pjl==PJL ∧ is_in_list (CJ, PJL)]

van den Bos, P., Tretmans, J. Coverage-Based Testing with Symbolic Transition Systems. In: Beyer, D., Keller, C. (eds) Tests and Proofs. TAP 2019.

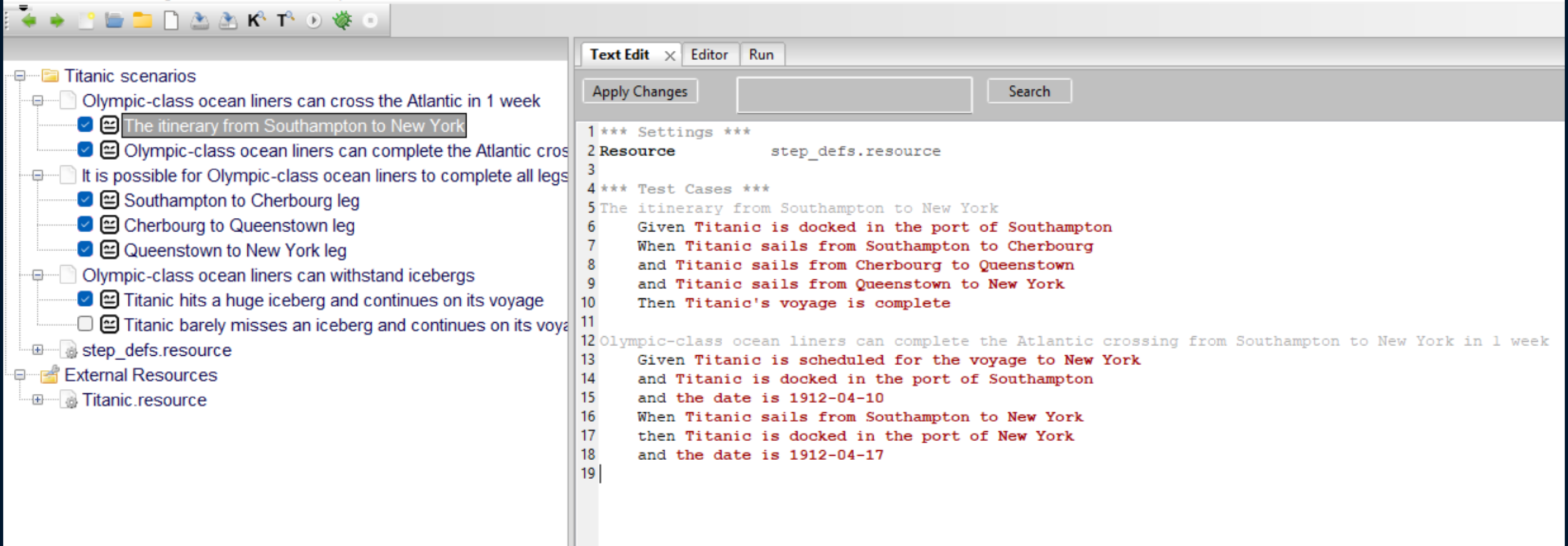
RobotframeworkMBT

Titanic





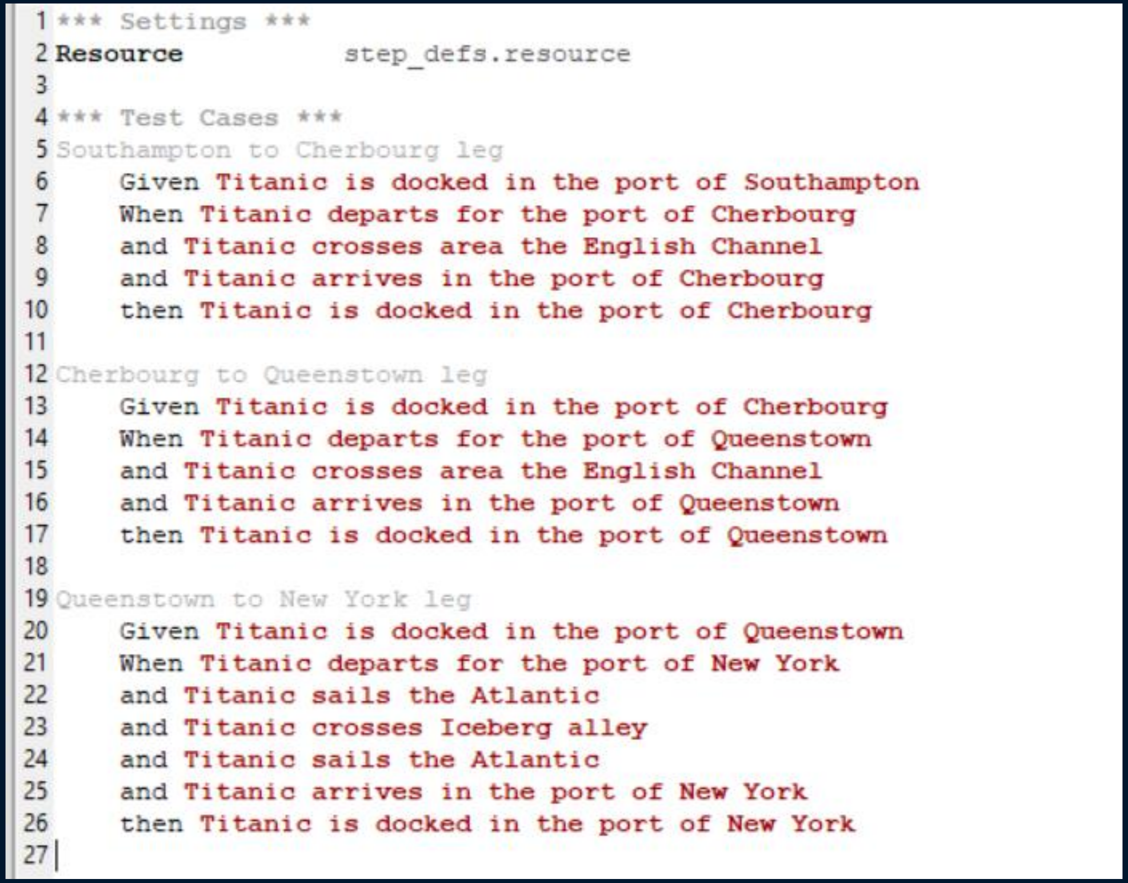
RIDE - Olympic-class ocean liners can cross the Atlantic



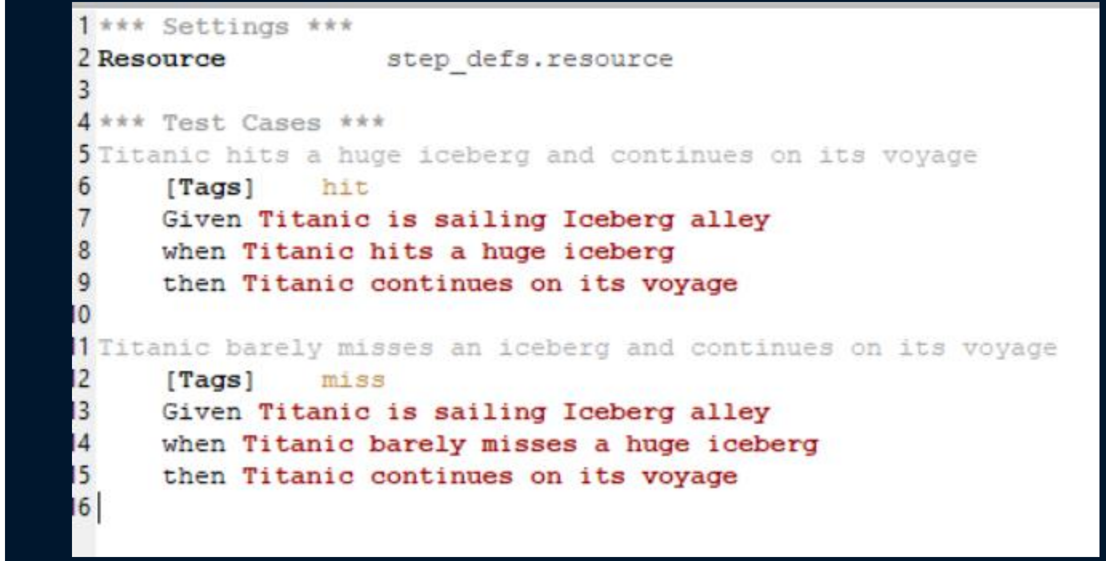
- Titanic scenarios
  - Olympic-class ocean liners can cross the Atlantic in 1 week
    - ☑ The itinerary from Southampton to New York
    - ☑ Olympic-class ocean liners can complete the Atlantic cros
  - It is possible for Olympic-class ocean liners to complete all legs
    - ☑ Southampton to Cherbourg leg
    - ☑ Cherbourg to Queenstown leg
    - ☑ Queenstown to New York leg
  - Olympic-class ocean liners can withstand icebergs
    - ☑ Titanic hits a huge iceberg and continues on its voyage
    - ☐ Titanic barely misses an iceberg and continues on its voya
  - step_defs.resource
- External Resources
  - Titanic.resource

```
*** Settings ***
Resource          step_defs.resource

*** Test Cases ***
The itinerary from Southampton to New York
    Given Titanic is docked in the port of Southampton
    When Titanic sails from Southampton to Cherbourg
    and Titanic sails from Cherbourg to Queenstown
    and Titanic sails from Queenstown to New York
    Then Titanic's voyage is complete

Olympic-class ocean liners can complete the Atlantic crossing from Southampton to New York in 1 week
    Given Titanic is scheduled for the voyage to New York
    and Titanic is docked in the port of Southampton
    and the date is 1912-04-10
    When Titanic sails from Southampton to New York
    then Titanic is docked in the port of New York
    and the date is 1912-04-17

```

Titanic Scenarios

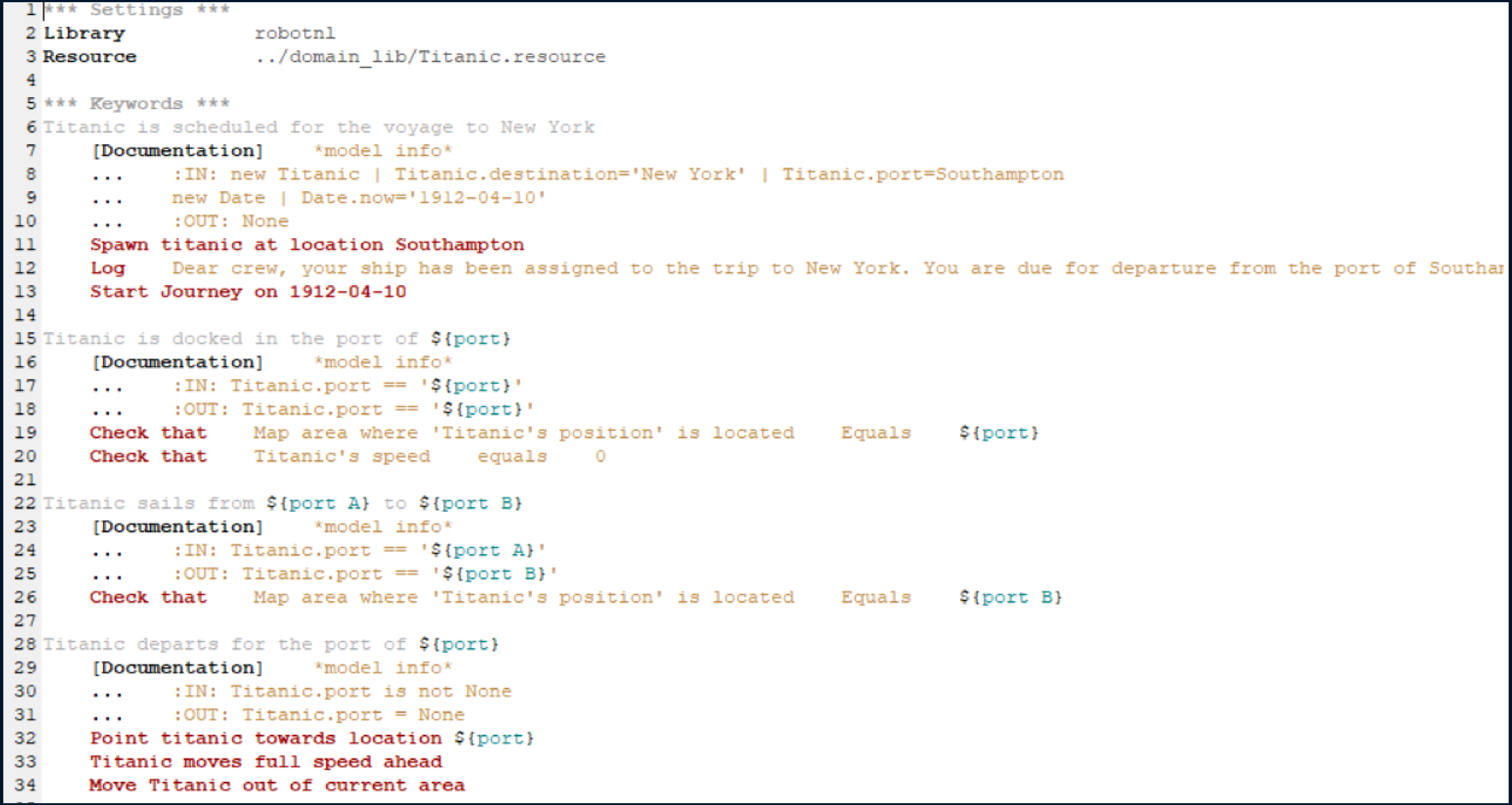


```
*** Settings ***
Resource          step_defs.resource

*** Test Cases ***
Southampton to Cherbourg leg
    Given Titanic is docked in the port of Southampton
    When Titanic departs for the port of Cherbourg
    and Titanic crosses area the English Channel
    and Titanic arrives in the port of Cherbourg
    then Titanic is docked in the port of Cherbourg

Cherbourg to Queenstown leg
    Given Titanic is docked in the port of Cherbourg
    When Titanic departs for the port of Queenstown
    and Titanic crosses area the English Channel
    and Titanic arrives in the port of Queenstown
    then Titanic is docked in the port of Queenstown

Queenstown to New York leg
    Given Titanic is docked in the port of Queenstown
    When Titanic departs for the port of New York
    and Titanic sails the Atlantic
    and Titanic crosses Iceberg alley
    and Titanic sails the Atlantic
    and Titanic arrives in the port of New York
    then Titanic is docked in the port of New York
```



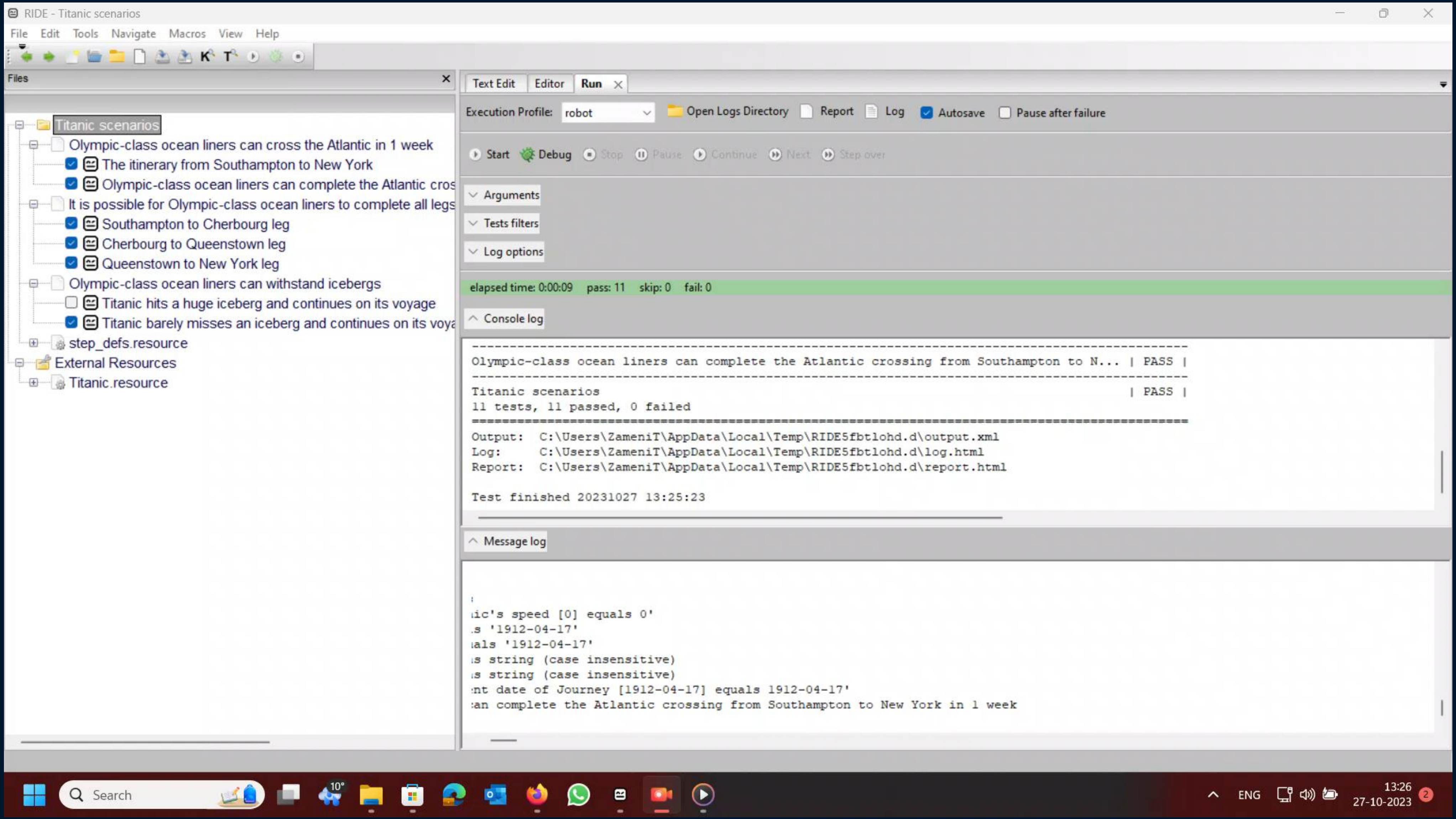
```
*** Settings ***
Resource          step_defs.resource

*** Test Cases ***
Titanic hits a huge iceberg and continues on its voyage
    [Tags]    hit
    Given Titanic is sailing Iceberg alley
    when Titanic hits a huge iceberg
    then Titanic continues on its voyage

Titanic barely misses an iceberg and continues on its voyage
    [Tags]    miss
    Given Titanic is sailing Iceberg alley
    when Titanic barely misses a huge iceberg
    then Titanic continues on its voyage
```

Modelling information



```
*** Settings ***
Library            robotnl
Resource           ../domain_lib/Titanic.resource

*** Keywords ***
Titanic is scheduled for the voyage to New York
    [Documentation]    *model info*
    ...    :IN: new Titanic | Titanic.destination='New York' | Titanic.port=Southampton
    ...    new Date | Date.now='1912-04-10'
    ...    :OUT: None
    Spawn titanic at location Southampton
    Log    Dear crew, your ship has been assigned to the trip to New York. You are due for departure from the port of Southar
    Start Journey on 1912-04-10

Titanic is docked in the port of ${port}
    [Documentation]    *model info*
    ...    :IN: Titanic.port == '${port}'
    ...    :OUT: Titanic.port == '${port}'
    Check that    Map area where 'Titanic's position' is located    Equals    ${port}
    Check that    Titanic's speed    equals    0

Titanic sails from ${port A} to ${port B}
    [Documentation]    *model info*
    ...    :IN: Titanic.port == '${port A}'
    ...    :OUT: Titanic.port == '${port B}'
    Check that    Map area where 'Titanic's position' is located    Equals    ${port B}

Titanic departs for the port of ${port}
    [Documentation]    *model info*
    ...    :IN: Titanic.port is not None
    ...    :OUT: Titanic.port = None
    Point titanic towards location ${port}
    Titanic moves full speed ahead
    Move Titanic out of current area
```

Demo Video attached



- Titanic scenarios
  - Olympic-class ocean liners can cross the Atlantic in 1 week
    - ☑ The itinerary from Southampton to New York
    - ☑ Olympic-class ocean liners can complete the Atlantic cros
  - It is possible for Olympic-class ocean liners to complete all legs
    - ☑ Southampton to Cherbourg leg
    - ☑ Cherbourg to Queenstown leg
    - ☑ Queenstown to New York leg
  - Olympic-class ocean liners can withstand icebergs
    - ☐ Titanic hits a huge iceberg and continues on its voyage
    - ☑ Titanic barely misses an iceberg and continues on its voya
  - step_defs.resource
- External Resources
  - Titanic.resource

Execution Profile: robot ☑ Autosave ☐ Pause after failure

elapsed time: 0:00:09 pass: 11 skip: 0 fail: 0

```
------------------------------------------------------------------------------------------------
Olympic-class ocean liners can complete the Atlantic crossing from Southampton to N... | PASS |
------------------------------------------------------------------------------------------------
Titanic scenarios                                                                      | PASS |
11 tests, 11 passed, 0 failed
================================================================================================
Output:  C:\Users\ZameniT\AppData\Local\Temp\RIDE5fbtlohd.d\output.xml
Log:     C:\Users\ZameniT\AppData\Local\Temp\RIDE5fbtlohd.d\log.html
Report:  C:\Users\ZameniT\AppData\Local\Temp\RIDE5fbtlohd.d\report.html

Test finished 20231027 13:25:23
```

```
:
ic's speed [0] equals 0'
s '1912-04-17'
als '1912-04-17'
s string (case insensitive)
s string (case insensitive)
nt date of Journey [1912-04-17] equals 1912-04-17'
an complete the Atlantic crossing from Southampton to New York in 1 week
```