Automated Generation of BPEL Adapters

International Conference on Service Oriented Computing, 2006

Antonio Brogi and Razvan Popescu

2009.05.27

Presented by In-Gwon Song

KAIST SE LAB
KAIST Software Engineering Laboratory

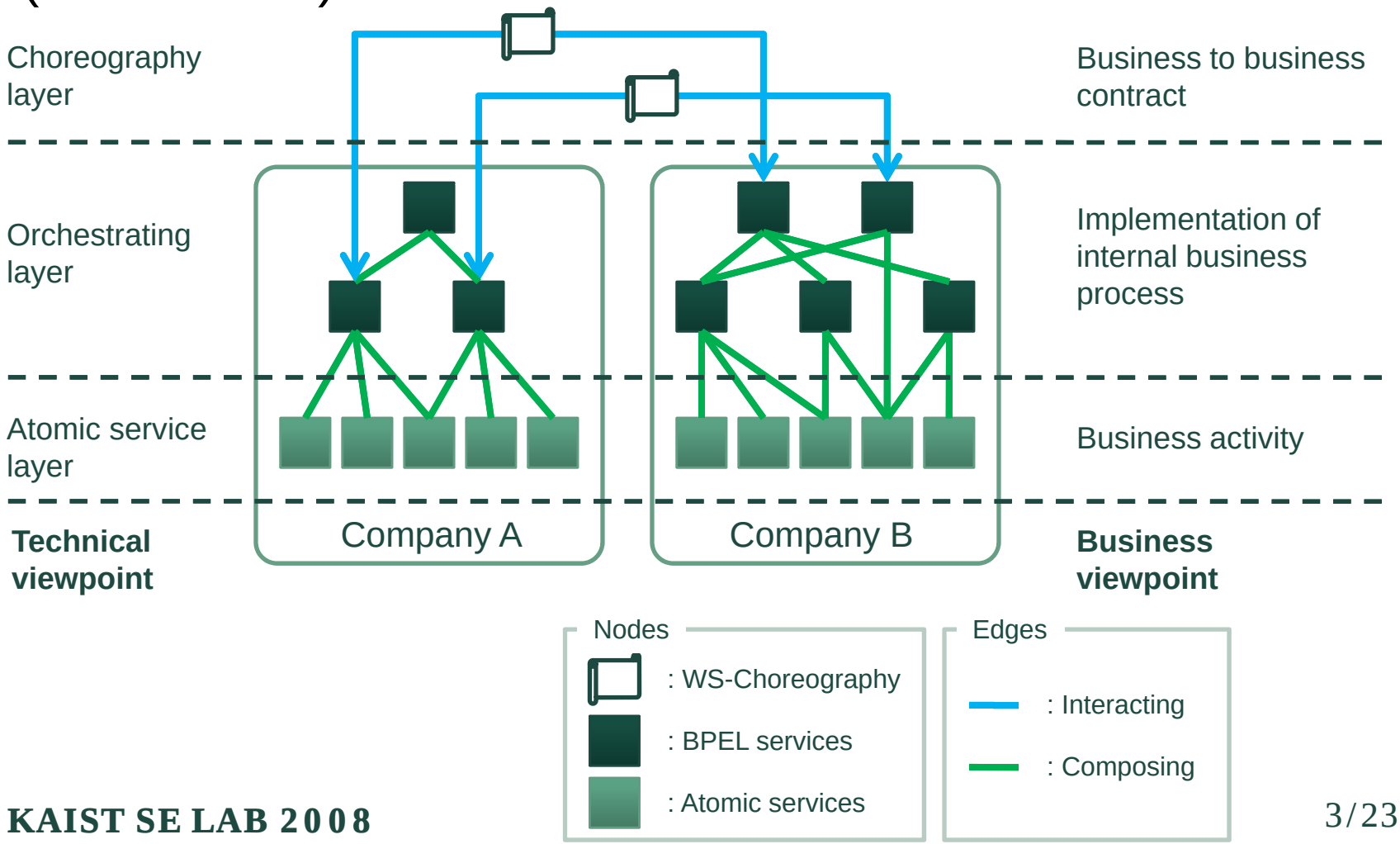
ⓒ KAIST SE LAB 2008

Contents

- Introduction
- Background
- Overall approach
- Automated generation of BPEL adapters
  - Service translation
  - Adapter generation
  - Adapter deployment
- Conclusion
- Discussion

Introduction(1/2)

- Composition hierarchy of service oriented architecture (web service)



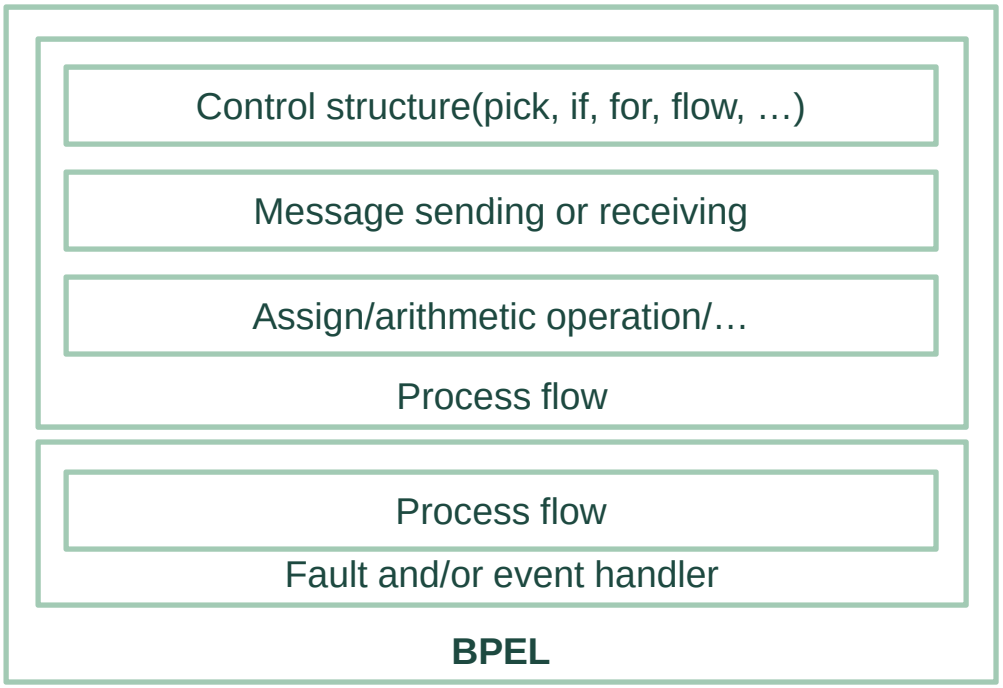
Introduction(2/2)

- Motivation
  - In most cases, BPEL services are not matched with each other
    - BPEL services are independently developed
  - BPEL services cannot be changed to match with other services
    - BPEL services reflect BPM(Business Process Model)

- Research goal
  - Generating an adapter between given two BPEL services for resolving the behavioral mismatch

Background(1/3)

- BPEL(Business Process Execution Language)
  - Derived from BPM(in general)
  - Composing services by a central BPEL service
    - BPEL engine execute BPEL as a service
  - Overall structure



Background(2/3)

- Example of BPEL

```
<process name="CommandCentre"><sequence>
    <receive op="ExecTask" from Client />
    <invoke op="Login" of MarsExplorer />
    <assign>...</assign>
    <invoke op="SetCoords" of MarsExplorer />
    <assign>...</assign>
    <invoke op="SetJob" of MarsExplorer />
    <pick>
      <onMsg op="SubmitRep" from MarsExplore><sequence>
            <receive op="JobID" from MarsExplorer />
            <invoke op="Logout" of MarsExplorer />
            <reply op="ExecTask" of Client /></sequence>
      </onMsg>
      <onMsg op="SubmitErr" from MarsExplorer><sequence>
            <invoke op="Logout" of MarsExplorer/>
            <assign>...</assign>
            <reply op="ExecTask" of Client/></sequence>
      </onMsg>
    </pick>
</sequence></process>
```

BPEL for command center service

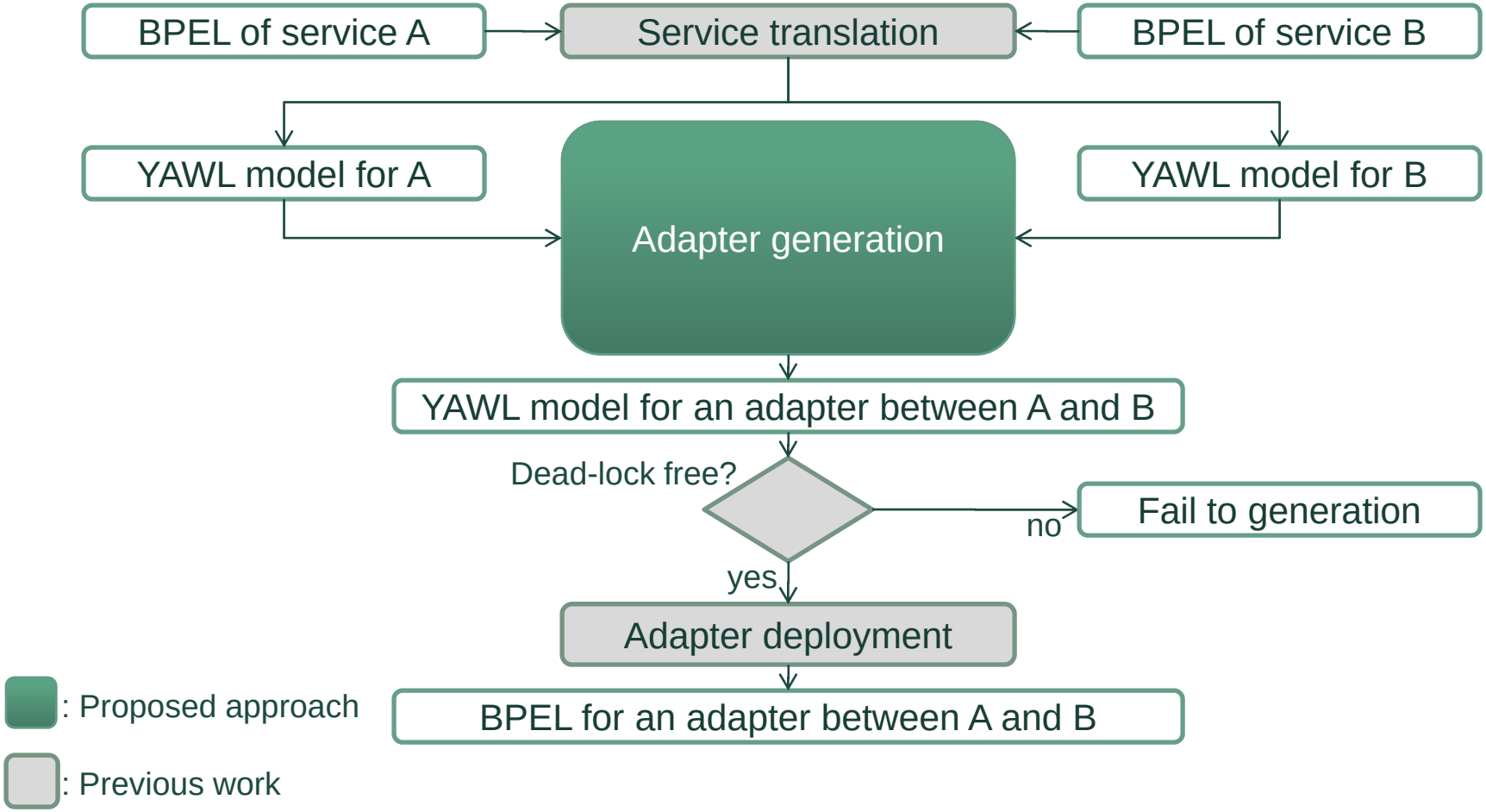
Background(3/3)

- Classification of mismatch
  - Technical level
    - Data encoding, OS, language, …
  - Signature level
    - Operation name, type name, …
  - Behavioral level
    - Interaction protocol
  - Quality of service level
    - Non functional requirements(performance, security, …)
  - Semantic level(conceptual level)
    - Functional requirements

In this paper, authors focus on behavioral level.

Overall approach

- Generating BPEL adapter



YAWL: Yet Another Workflow Language

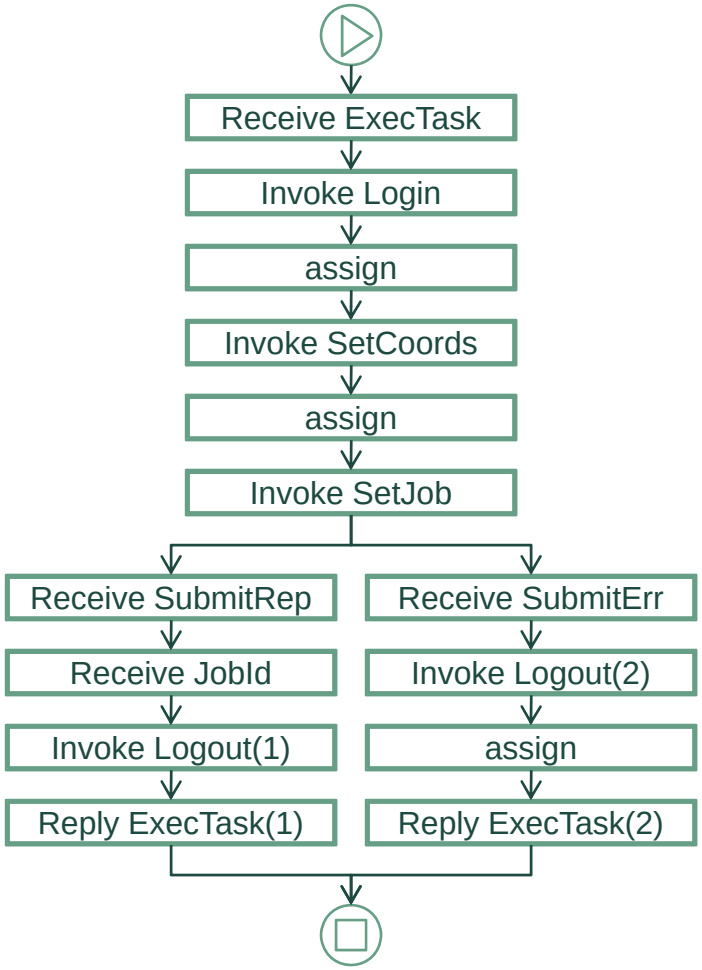
Service translation(1/2)

- Command center

```
<process name="CommandCentre"><sequence>
    <receive op="ExecTask" from Client />
    <invoke op="Login" of MarsExplorer />
    <assign>...</assign>
    <invoke op="SetCoords" of MarsExplorer />
    <assign>...</assign>
    <invoke op="SetJob" of MarsExplorer />
    <pick>
      <onMsg op="SubmitRep" from MarsExplore><sequence>
            <receive op="JobID" from MarsExplorer />
            <invoke op="Logout" of MarsExplorer />
            <reply op="ExecTask" of Client /></sequence>
      </onMsg>
      <onMsg op="SubmitErr" from MarsExplorer><sequence>
            <invoke op="Logout" of MarsExplorer/>
            <assign>...</assign>
            <reply op="ExecTask" of Client/></sequence>
      </onMsg>
    </pick>
</sequence></process>
```

BPEL

Translation



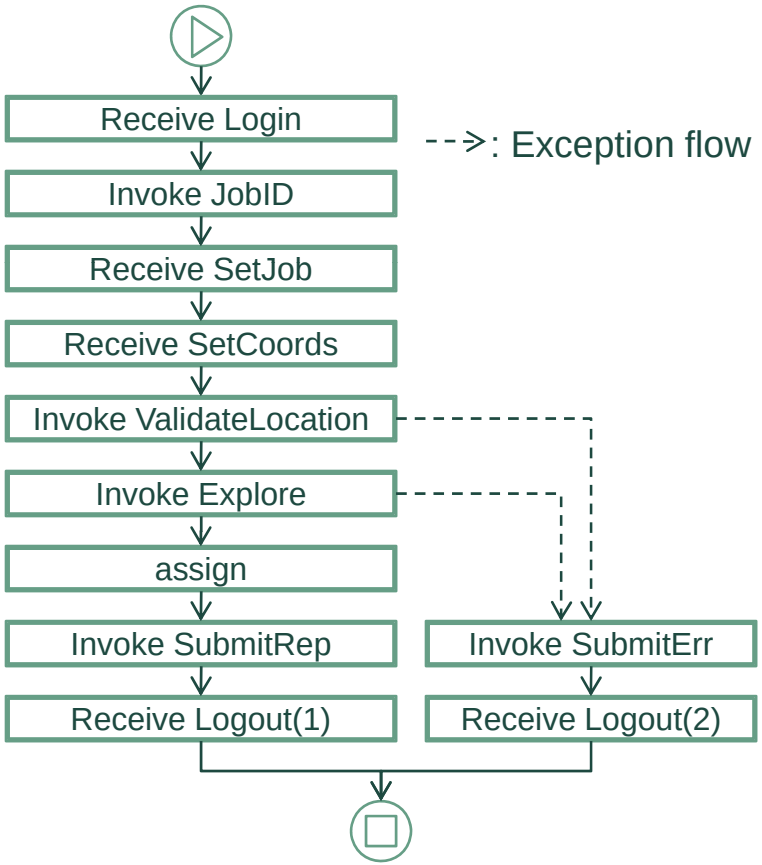
YAWL(abstracted)

ⓒ KAIST SE LAB 2008

Service translation(2/2)

- Mars explorer

```
<process name="MarsExplorer">
    <sequence>
        <receive op="Login" from CommandCentre />
        <invoke op="JobID" of CommandCentre />
        <receive op="SetJob" from CommandCentre />
        <receive op="SetCoords" from CommandCentre />
        <invoke op="ValidateLocation" of LoggerService />
        <invoke op="Explore" of ExplorerService />
        <assign> … </assign>
        <invoke op="SubmitRep" of CommandCentre />
        <receive op="Logout" from CommandCentre />
    </sequence>
  <faultHandlers>
    <catch faultName="Task Error" faultVar="error">
        <sequence>
            <invoke op="SubmitErr" of CommandCentre />
            <receive op="Logout" from CommandCentre />
        </sequence>
    </catch>
  </faultHandlers>
</process>
```



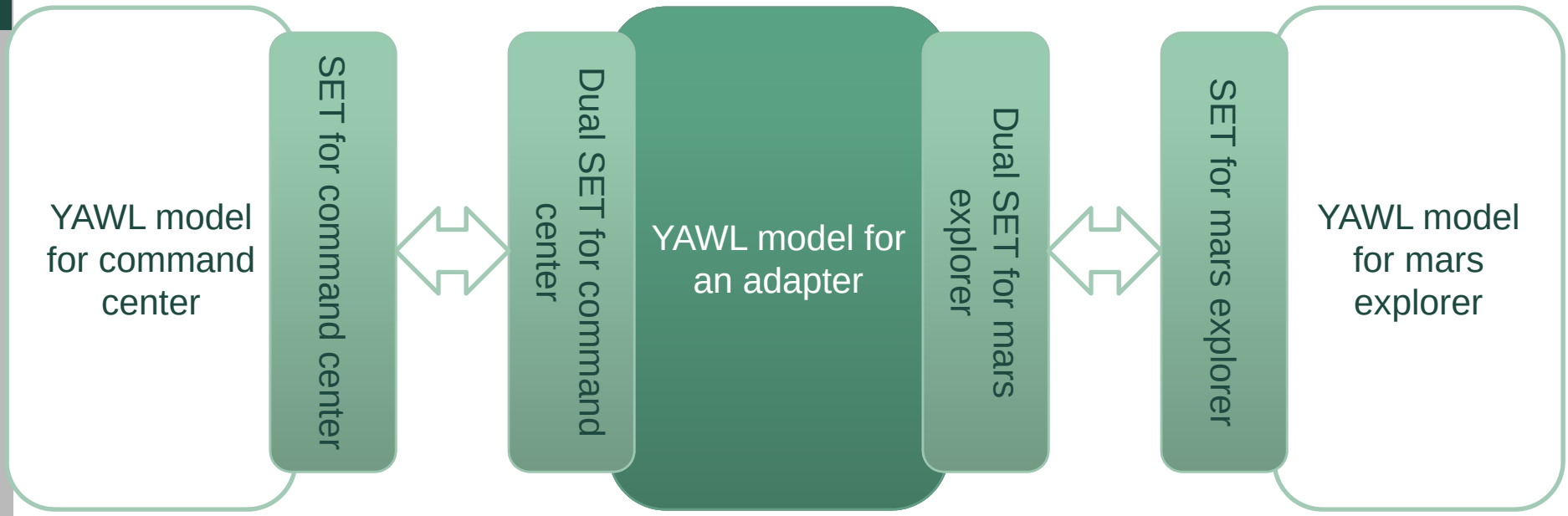
BPEL

Translation

YAWL(abstracted)

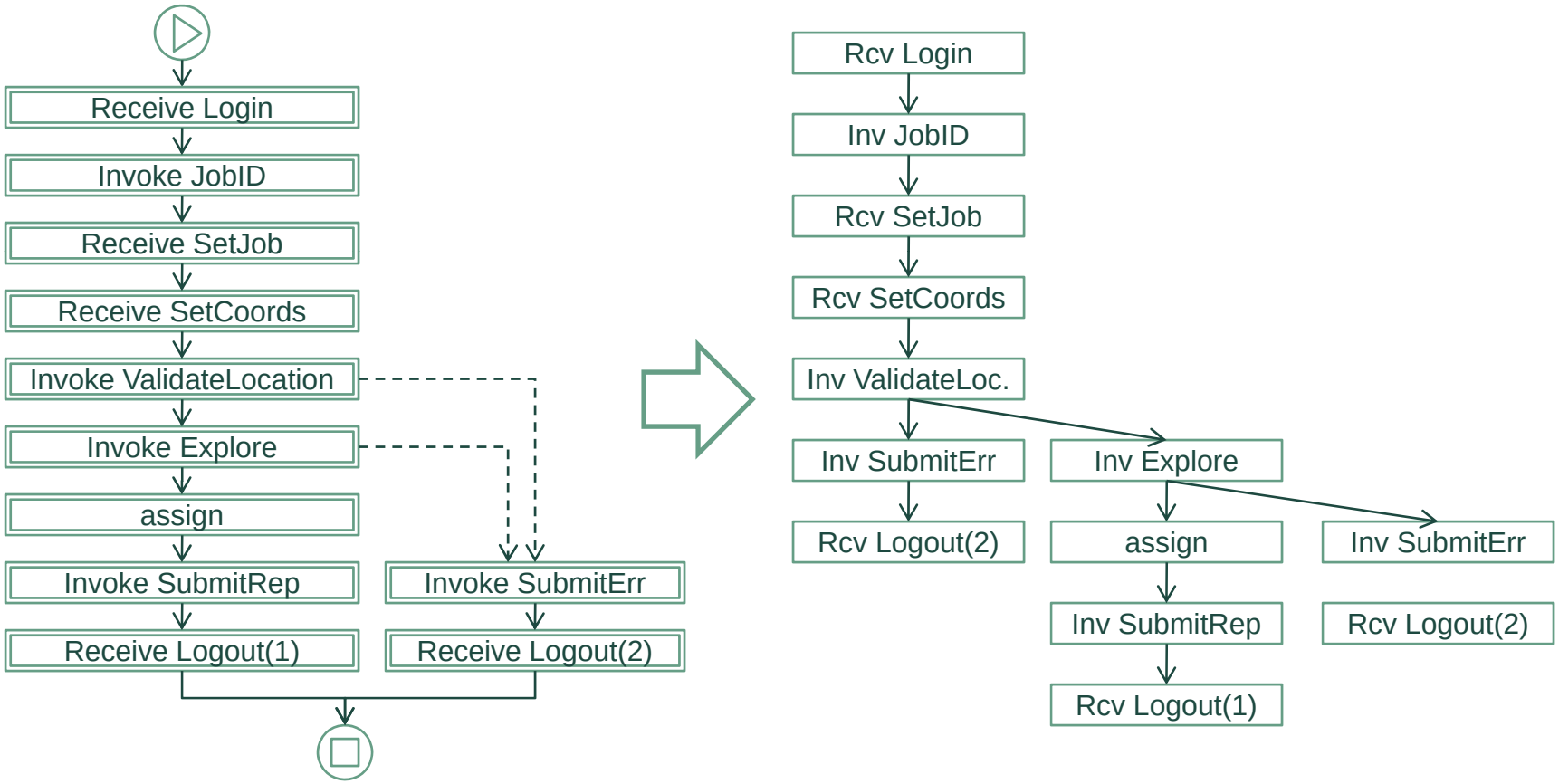
Adapter generation

- Overall procedure
  - Extracting a SET(Service Execution Tree) from each YAWL model
  - Deriving a dual SET for the SET of each YAWL model
  - Deriving a SET of the adapter
  - Generating a YAWL model for the adapter



Extracting SETs(1/3)

- Extracting possible execution traces

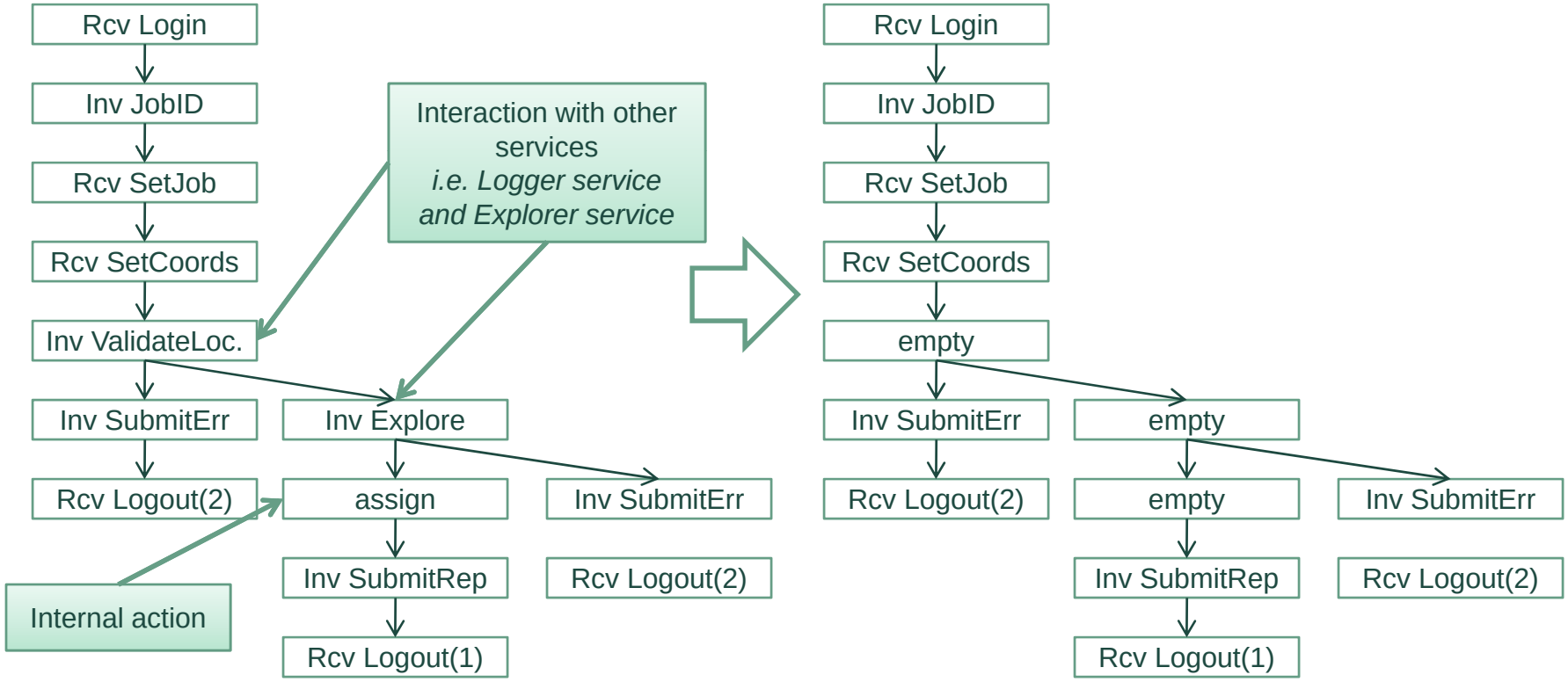


YAWL(abstracted) ***SET(Service Execution Tree)***

Example for mars explorer

Extracting SETs(2/3)

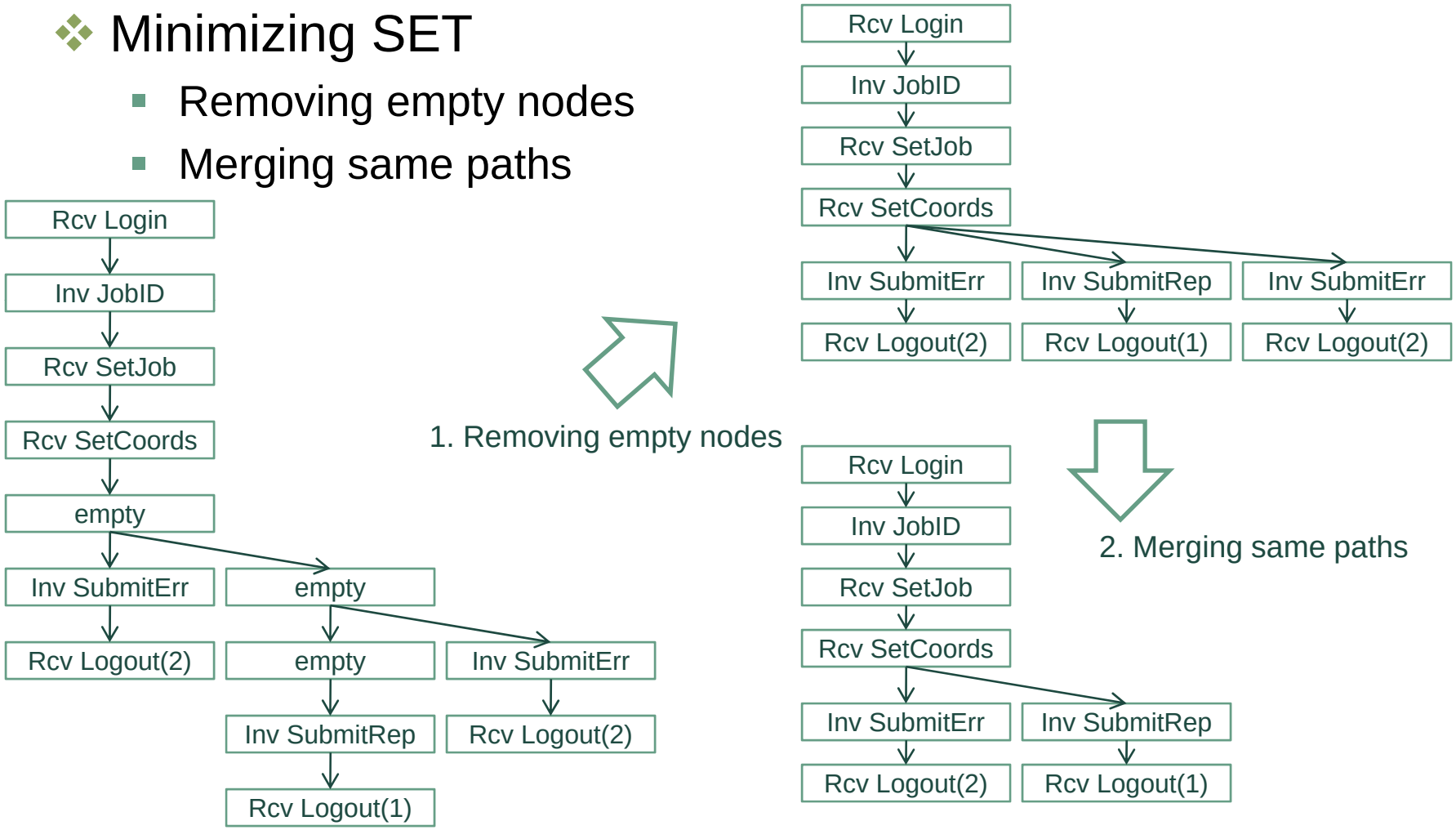
- Substituting irrelevant nodes with empty nodes
  - Remaining nodes interacting with a target service



SET(Service Execution Tree)

Extracting SETs(3/3)

- Minimizing SET
  - Removing empty nodes
  - Merging same paths

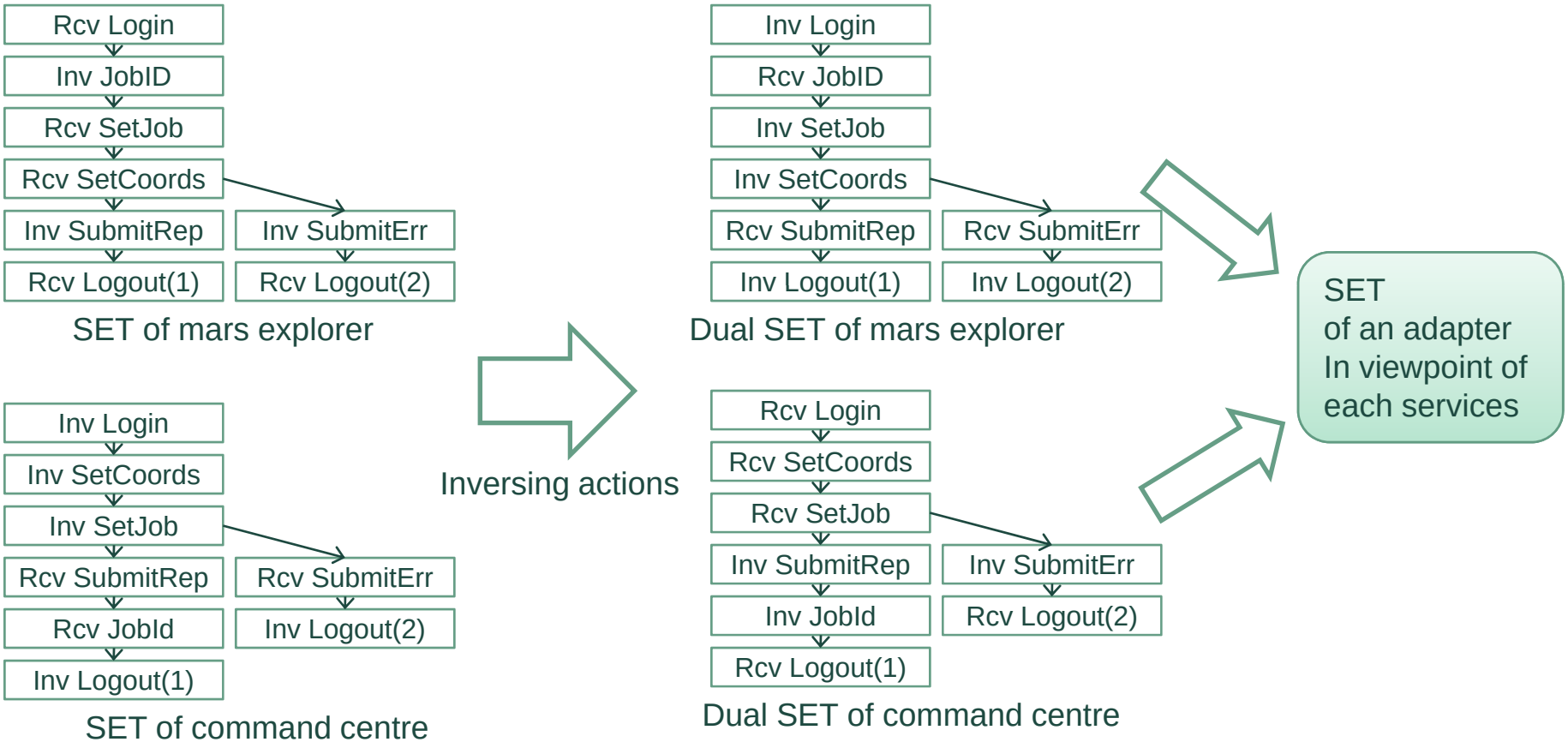


SET(Service Execution Tree)

Example for mars explorer

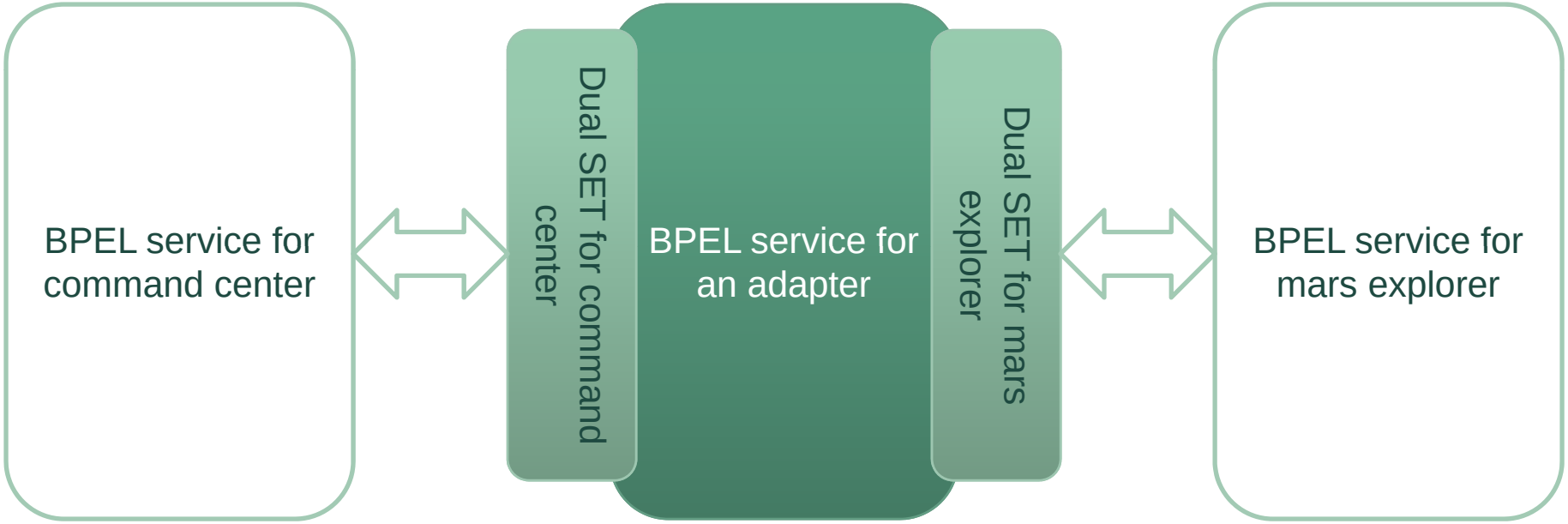
Deriving a dual SET for each SET of BPEL(1/2)

- Creating a SET of an adapter in viewpoint of each service
  - Deriving a dual SET of each services
    - Dual SET is derived from a SET by inversing each actions



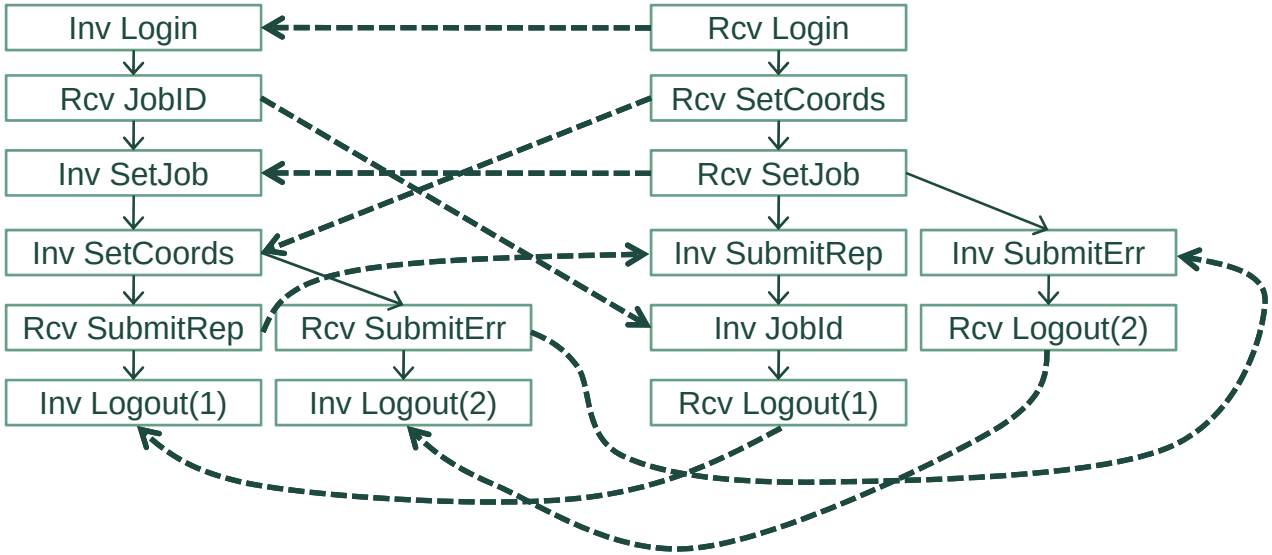
Deriving a dual SET for each SET of BPEL(2/2)

- Creating a SET of an adapter in viewpoint of each service
  - Each dual SET represents an interfaces for corresponding service



Deriving a SET of an adapter(1/2)

- Deriving data flows
  - Received message is forwarded to target service
    - Receipt of a message should occur before sending the message

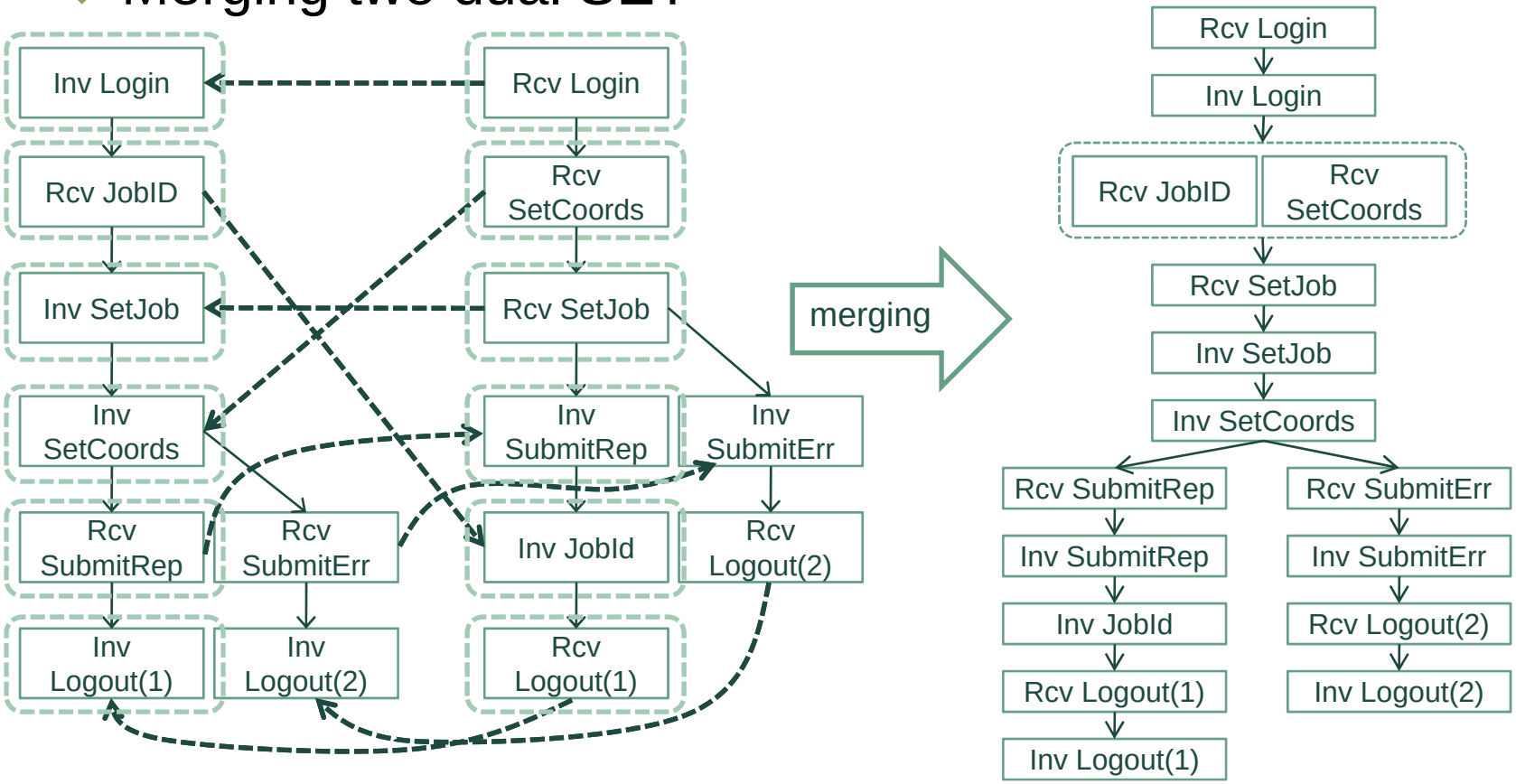


Dual SET of mars explorer

Dual SET of command centre

Deriving a SET of an adapter(2/2)

- Merging two dual SET



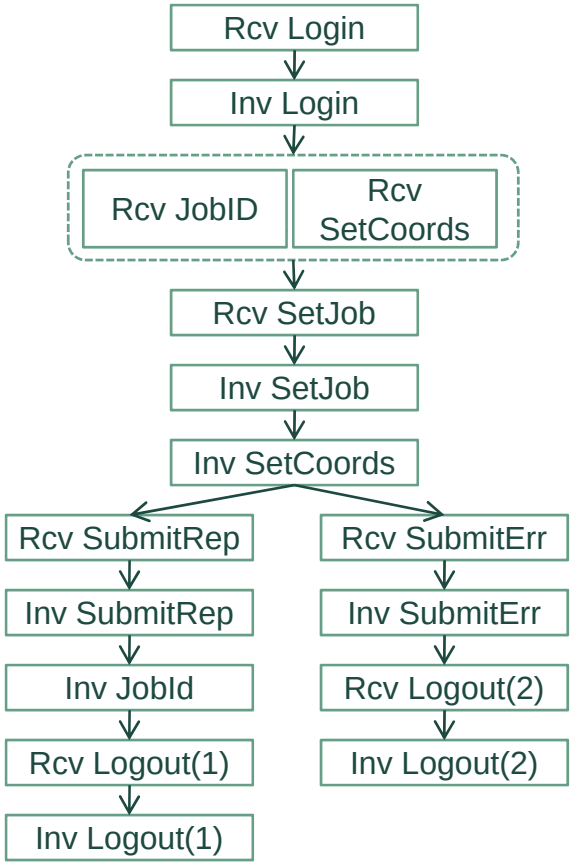
Dual SETs of mars explorer and command centre

SET of an adapter

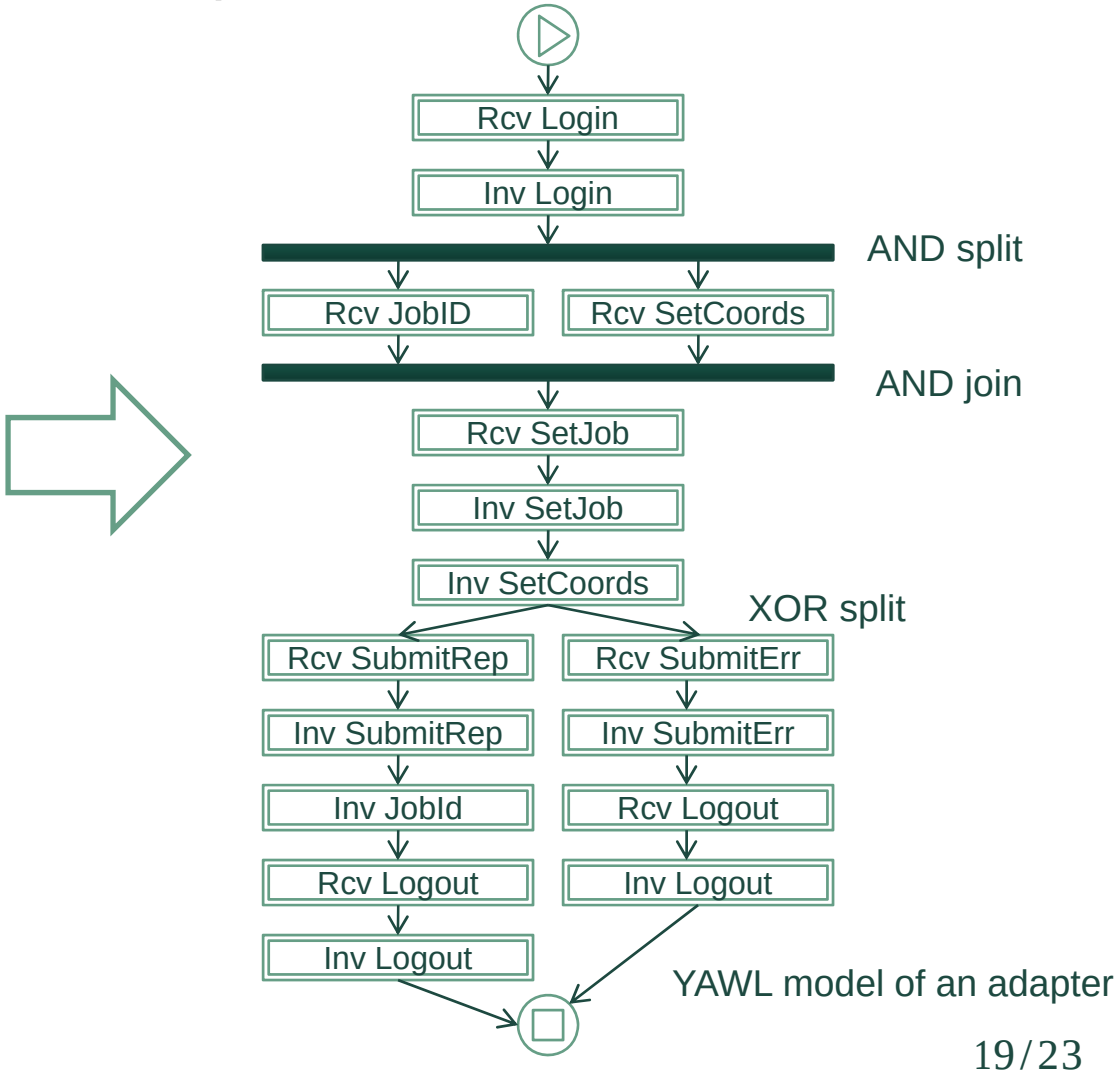
: Concurrent execution

Deriving a YAWL model for an adapter

- Translating a SET of an adapter to a YAWL model



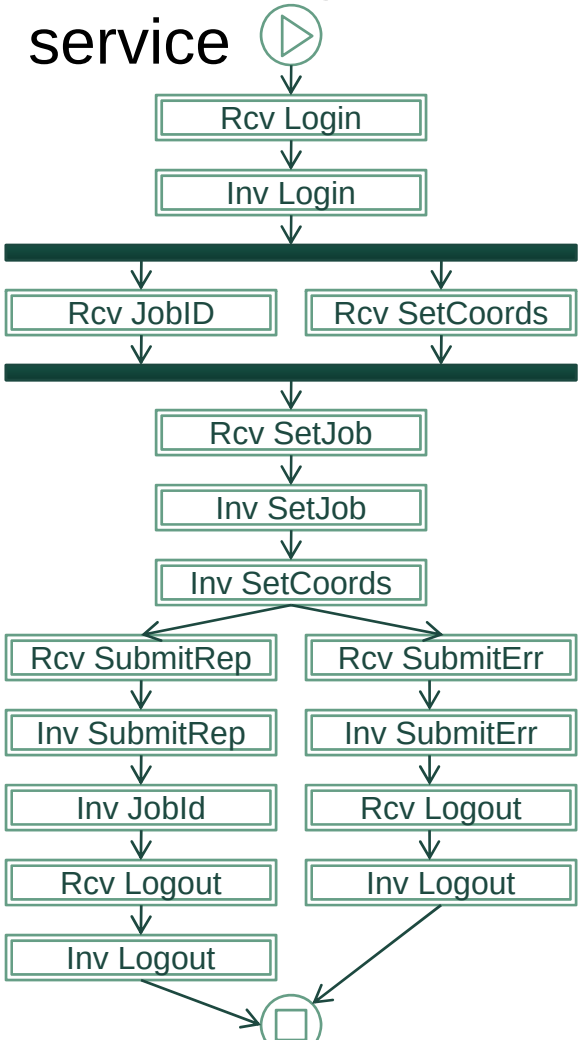
SET of an adapter



YAWL model of an adapter

Adapter deployment

- Translating a YAWL model for an adapter to a BPEL service



YAWL model of an adapter

```
<process name="Adapter for CC and ME">
  <sequence>
    <receive op="Login" from CommandCentre/>
    <invoke op="Login" of MarsExplorer/>
    <flow>
      <receive op="SetCoords" from CommandCentre/>
      <receive op="JobID" from MarsExplorer/></flow>
    <receive op="SetJob" from CommandCentre/>
    <invoke op="SetJob" of MarsExplorer/>
    <invoke op="SetCoords" of MarsExplorer/>
    <pick>
      <onMsg op="SubmitRep" from MarsExplorer >
        <sequence>
          <invoke op="SubmitRep" of CommandCentre/>
          <invoke op="JobID" of CommandCentre />
          <receive op="Logout" from CommandCentre/>
          <invoke op="Logout" of MarsExplorer/>
        </sequence></onMsg>
      <onMsg op="SubmitErr" from MarsExplorer >
        <sequence>
          <invoke op="SubmitErr" of CommandCentre/>
          <receive op="Logout" from CommandCentre/>
          <invoke op="Logout" of MarsExplorer/>
        </sequence></onMsg></pick>
  </sequence>
</process>
```

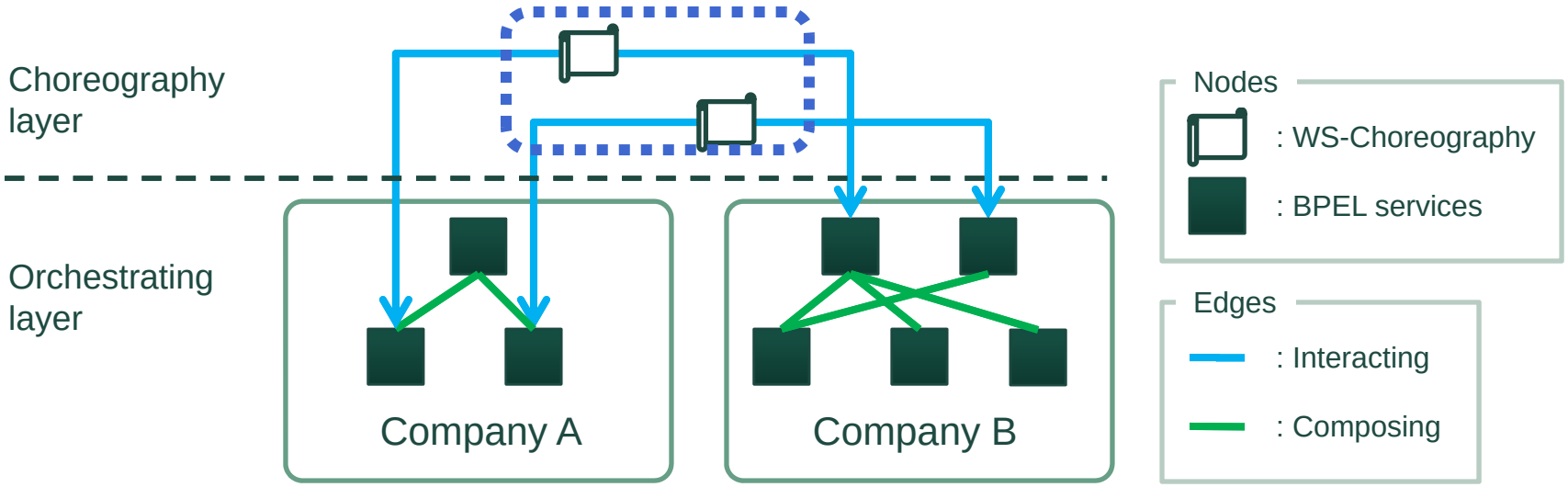
BPEL for an adapter

Conclusion

- Summary
  - Automatic synthesis of BPEL adapter
  - Generation of YAWL model for the adapter
    - YAWL allows formal analysis (dead-lock, reachability, liveness, …)
    - YAWL can be simulated with prior researches
- Future work
  - Generation of an adapter between multiple services
  - Resolving ontology mismatch(semantic mismatch)

Discussion

- Pros
  - First approach to generating an adapter between BPEL services
  - Full automation
- Cons
  - Insufficient explanation for supporting loops
  - Informal algorithms
  - Not optimized concurrency

Discussion

- My research
  - Main goal
    - Automatic generation of choreography
  - Intermediate goals
    - Resolving mismatches between BPEL services
    - Resolving dead-lock and implied scenarios