GIVUP: Automated Generation and Verification of Textual Process Descriptions

NIVON Quentin, SALAÜN Gwen, LANG Frédéric

What is BPMN?

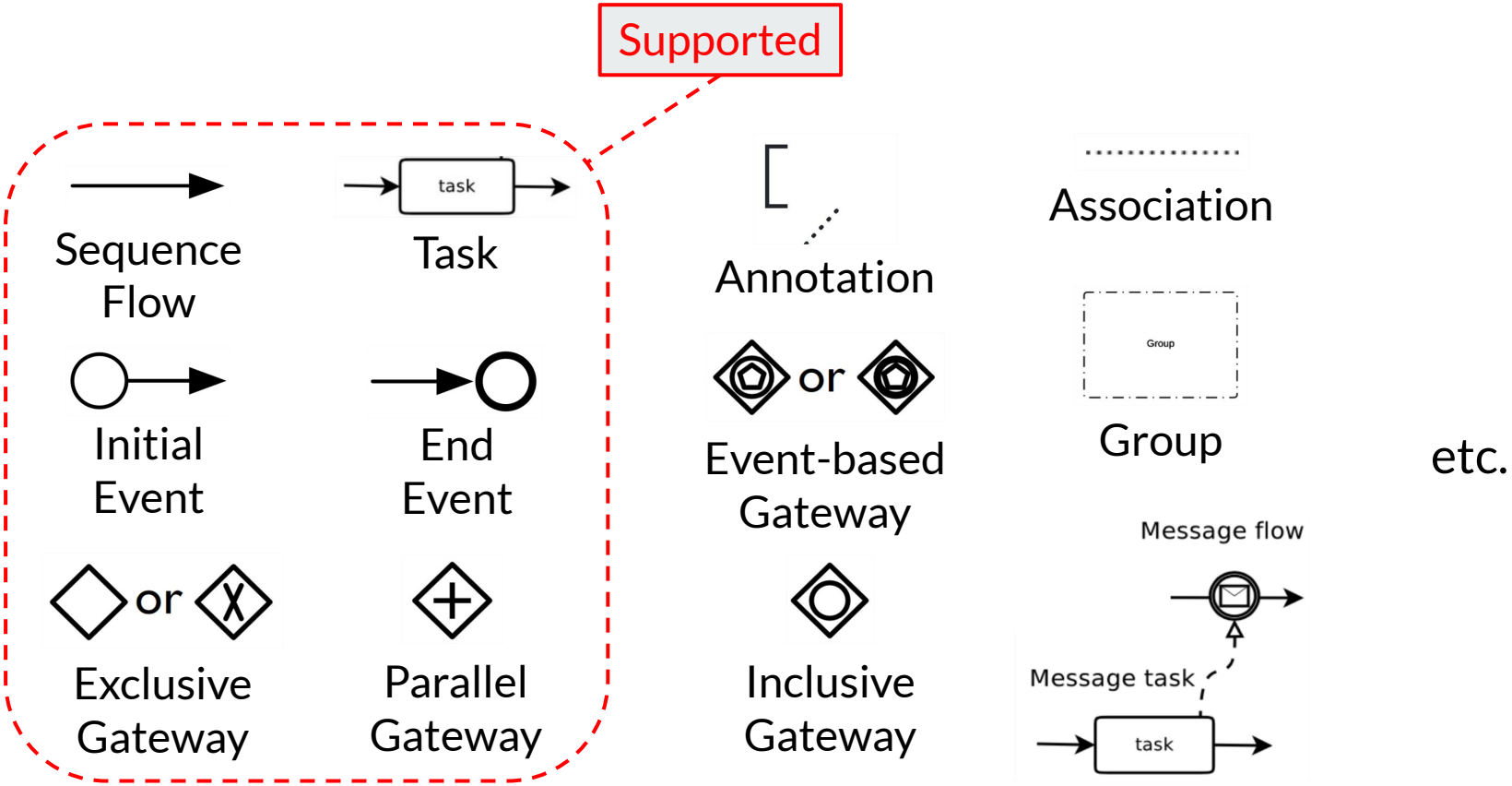
- A **workflow-based notation** created in 2004 by the Business Process Management Initiative (BPMI) and the Object Management Group (OMG).
- It aims at **representing business processes** in a way that is **understandable for both experienced and novice users.**
- An **ISO/IEC standard** since version 2.0 in 2013.

What is BPMN?

- A **workflow-based notation** created in 2004 by the Business Process Management Initiative (BPMI) and the Object Management Group (OMG).
- It aims at **representing business processes** in a way that is **understandable for both experienced and novice users.**
- An **ISO/IEC standard** since version 2.0 in 2013.

BPMN Syntax

Supported



Context

- Companies are making use of the BPMN notation to **represent their business processes.**

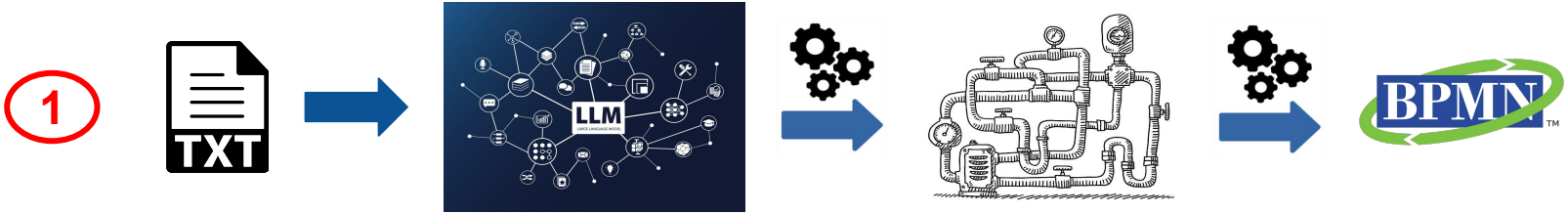
- They **hire experts** to analyse and design the **most adequate BPMN process** according to their needs.
  They also expect the experts to build a process having **the desired behaviour**.

- These processes are often **syntactically/semantically incorrect**.
  Often, they do not meet the **expected requirements**.

Research Questions

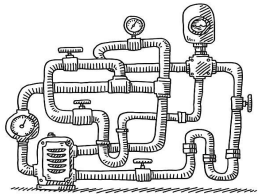
- What if you do not know **how to write BPMN**?
- What if you do not want to **spend time designing** your BPMN process graphically?
- How can you be sure that your BPMN process is **syntactically/semantically correct**?
- How can you be sure that your BPMN process has **the expected behaviour**?

General Solution



General Solution

1



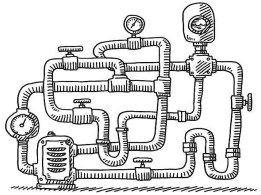
BPMN™

2

$$\Box\varphi \leftrightarrow \Box\Box\varphi$$
$$\Diamond\varphi \leftrightarrow \Diamond\Diamond\varphi$$
$$\Box\varphi \rightarrow \varphi \wedge \circ\varphi$$
$$\varphi \vee \bullet\varphi \rightarrow \Diamond\varphi$$
$$\Box(\varphi \rightarrow \bullet\varphi) \rightarrow \neg\varphi$$
$$\Box(\varphi_1 \wedge \varphi_2) \leftrightarrow \Box\varphi_1 \wedge \Box\varphi_2$$
$$\Box(\varphi_1 \rightarrow \varphi_2) \rightarrow (\Diamond\varphi_1 \rightarrow \Diamond\varphi_2)$$

General Solution

1



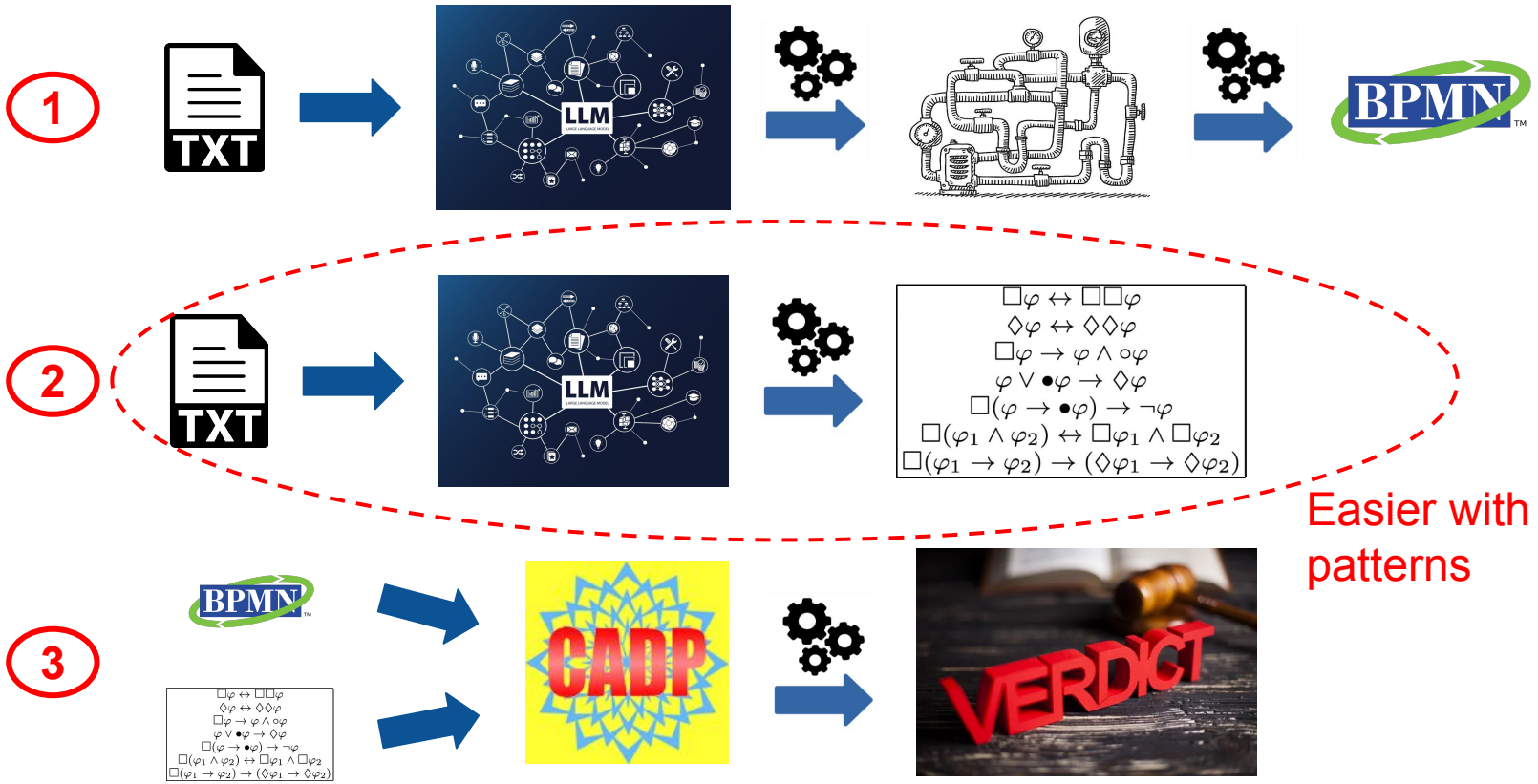
2

$$\Box\varphi \leftrightarrow \Box\Box\varphi$$
$$\Diamond\varphi \leftrightarrow \Diamond\Diamond\varphi$$
$$\Box\varphi \rightarrow \varphi \wedge \circ\varphi$$
$$\varphi \vee \bullet\varphi \rightarrow \Diamond\varphi$$
$$\Box(\varphi \rightarrow \bullet\varphi) \rightarrow \neg\varphi$$
$$\Box(\varphi_1 \wedge \varphi_2) \leftrightarrow \Box\varphi_1 \wedge \Box\varphi_2$$
$$\Box(\varphi_1 \rightarrow \varphi_2) \rightarrow (\Diamond\varphi_1 \rightarrow \Diamond\varphi_2)$$

3

$$\Box\varphi \leftrightarrow \Box\Box\varphi$$
$$\Diamond\varphi \leftrightarrow \Diamond\Diamond\varphi$$
$$\Box\varphi \rightarrow \varphi \wedge \circ\varphi$$
$$\varphi \vee \bullet\varphi \rightarrow \Diamond\varphi$$
$$\Box(\varphi \rightarrow \bullet\varphi) \rightarrow \neg\varphi$$
$$\Box(\varphi_1 \wedge \varphi_2) \leftrightarrow \Box\varphi_1 \wedge \Box\varphi_2$$
$$\Box(\varphi_1 \rightarrow \varphi_2) \rightarrow (\Diamond\varphi_1 \rightarrow \Diamond\varphi_2)$$

General Solution



Overview of our Solution for 1

First of all, an employee CollectGoods. Then, the client PayForDelivery while the employee PrepareParcel. Finally, the company can either DeliverByCar or DeliverByDrone (depending on the distance for example)

Textual Representation of the Process

Overview of our Solution for 1

First of all, an employee CollectGoods. Then, the client PayForDelivery while the employee PrepareParcel. Finally, the company can either DeliverByCar or DeliverByDrone (depending on the distance for example)

Textual Representation of the Process

Large Language Model (LLM)

Overview of our Solution for 1

First of all, an employee CollectGoods. Then, the client PayForDelivery while the employee PrepareParcel. Finally, the company can either DeliverByCar or DeliverByDrone (depending on the distance for example)

Textual Representation of the Process

Large Language Model (LLM)

- CollectGoods < (PayForDelivery, PrepareParcel)
- (PayForDelivery, PrepareParcel) < (DeliverByCar, DeliverByDrone)

$$\langle E \rangle ::= \texttt{t} \mid (\langle E \rangle) \mid \langle E_1 \rangle \langle op \rangle \langle E_2 \rangle \mid (\langle E_1 \rangle)*$$

$$\langle op \rangle ::= \text{'|'} \mid \text{'\&'} \mid \text{'<'} \mid \text{','}$$

Expressions Following an Internal Grammar

Overview of our Solution for 1

First of all, an employee CollectGoods. Then, the client PayForDelivery while the employee PrepareParcel. Finally, the company can either DeliverByCar or DeliverByDrone (depending on the distance for example)

Textual Representation of the Process

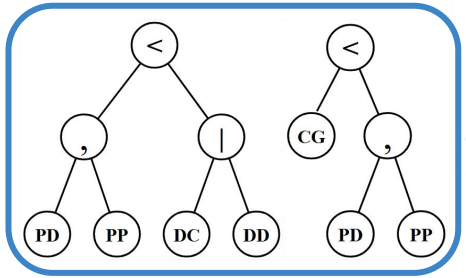
Large Language Model (LLM)

- CollectGoods < (PayForDelivery, PrepareParcel)
- (PayForDelivery, PrepareParcel) < (DeliverByCar, DeliverByDrone)

$$\langle E \rangle ::= \texttt{t} \mid (\langle E \rangle) \mid \langle E_1 \rangle \ \langle op \rangle \ \langle E_2 \rangle \mid (\langle E_1 \rangle)*$$

$$\langle op \rangle ::= \text{'|'} \mid \text{'\&'} \mid \text{'<'} \mid \text{','}$$

Expressions Following an Internal Grammar



Abstract Syntax Trees

Overview of our Solution for 1

First of all, an employee CollectGoods. Then, the client PayForDelivery while the employee PrepareParcel. Finally, the company can either DeliverByCar or DeliverByDrone (depending on the distance for example)

Textual Representation of the Process

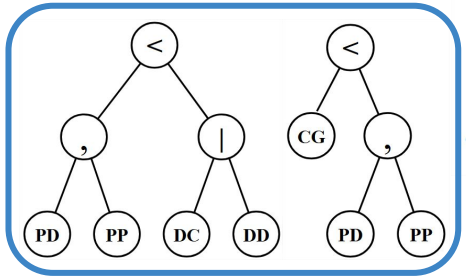
Large Language Model (LLM)

- CollectGoods < (PayForDelivery, PrepareParcel)
- (PayForDelivery, PrepareParcel) < (DeliverByCar, DeliverByDrone)

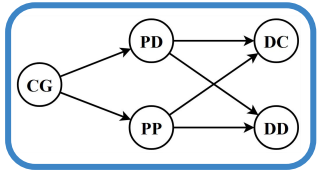
$$\langle E \rangle ::= \texttt{t} \mid (\langle E \rangle) \mid \langle E_1 \rangle \ \langle op \rangle \ \langle E_2 \rangle \mid (\langle E_1 \rangle)*$$

$$\langle op \rangle ::= \text{'|'} \mid \text{'\&'} \mid \text{'<'} \mid \text{','}$$

Expressions Following an Internal Grammar



Abstract Syntax Trees



Dependency Graph

Overview of our Solution for 1

First of all, an employee CollectGoods. Then, the client PayForDelivery while the employee PrepareParcel. Finally, the company can either DeliverByCar or DeliverByDrone (depending on the distance for example)

Textual Representation of the Process

GPT-3.5

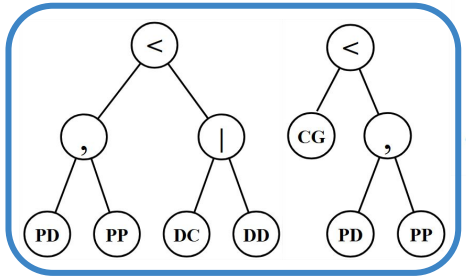
Large Language Model (LLM)

- CollectGoods < (PayForDelivery, PrepareParcel)
- (PayForDelivery, PrepareParcel) < (DeliverByCar, DeliverByDrone)

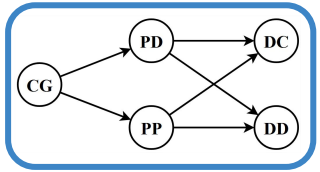
$$\langle E \rangle ::= \texttt{t} \mid (\langle E \rangle) \mid \langle E_1 \rangle \ \langle op \rangle \ \langle E_2 \rangle \mid (\langle E_1 \rangle)*$$

$$\langle op \rangle ::= \text{'|'} \mid \text{'\&'} \mid \text{'<'} \mid \text{','}$$

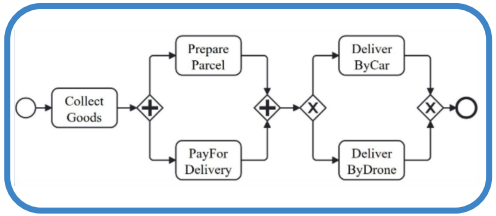
Expressions Following an Internal Grammar



Abstract Syntax Trees



Dependency Graph



BPMN Process

Detailed Solution of 1 – Step 1

The user first has to write a **textual description** of the process-to-be.

First, the banker either CreateProfile (CP) for the user, or, if it is not needed, he RetrieveCustomerProfile (RCP) which triggers the system to perform the AnalyseCustomerProfile (ACP) task.
Then, the user executes the task ReceiveSupportDocuments (RSD) so that the system can start UpdateInfoRecords (UID) and perform a BackgroundVerification (BV).
If the verification finds missing or incorrect information, the system RequestAdditionalInfo (RAI) to the user, who has to ReceiveSupportDocuments (RSD) again.
Otherwise, the process ends with CreateAccount (CA).

Detailed Solution of 1 – Step 2

The textual description is then **given to a (fine-tuned) LLM** (GPT 3.5 atm).

First, the banker either CreateProfile (CP) for the user, or, if it is not
needed, he RetrieveCustomerProfile (RCP) which triggers the system to perform the AnalyseCustomerProfile (ACP) task.
Then, the user executes the task ReceiveSupportDocuments (RSD) so that the system can start UpdateInfoRecords (UID) and perform a BackgroundVerification (BV).
If the verification finds missing or incorrect information, the system RequestAdditionalInfo (RAI) to the user, who has to ReceiveSupportDocuments (RSD) again.
Otherwise, the process ends with CreateAccount (CA).

OpenAI

GPT - 3.5

Detailed Solution of 1 – Step 2

The textual description is then **given to a (fine-tuned) LLM** (GPT 3.5 atm).

First, the banker either CreateProfile (CP) for the user, or, if it is not
needed, he RetrieveCustomerProfile (RCP) which triggers the system to perform the AnalyseCustomerProfile (ACP) task.
Then, the user executes the task ReceiveSupportDocuments (RSD) so that the system can start UpdateInfoRecords (UID) and perform a BackgroundVerification (BV).
If the verification finds missing or incorrect information, the system RequestAdditionalInfo (RAI) to the user, who has to ReceiveSupportDocuments (RSD) again.
Otherwise, the process ends with CreateAccount (CA).

OpenAI

GPT - 3.5

The LLM processes the description and returns a **set of expressions** following an **internal grammar**.

$$\langle E \rangle ::= \mathtt{t} \mid (\langle E \rangle) \mid \langle E_1 \rangle \langle op \rangle \langle E_2 \rangle \mid (\langle E_1 \rangle)*$$

$$\langle op \rangle ::= \text{‘|’} \mid \text{‘\&’} \mid \text{‘<’} \mid \text{‘;’}$$

Detailed Solution of 1 – Step 3

Given our description, the LLM returns **three expressions**:

Detailed Solution of 1 – Step 3

Given our description, the LLM returns **three expressions**:

```
(RetrieveCustomerProfile < AnalyseCustomerProfile) | CreateProfile
```

Detailed Solution of 1 – Step 3

Given our description, the LLM returns **three expressions**:

(RetrieveCustomerProfile < AnalyseCustomerProfile) | CreateProfile

(RetrieveCustomerProfile, AnalyseCustomerProfile, CreateProfile) <
(ReceiveSupportDocuments < (UpdateInfoRecords, BackgroundVerification))

Detailed Solution of 1 – Step 3

Given our description, the LLM returns **three expressions**:

(RetrieveCustomerProfile < AnalyseCustomerProfile) | CreateProfile

(RetrieveCustomerProfile, AnalyseCustomerProfile, CreateProfile) < (ReceiveSupportDocuments < (UpdateInfoRecords, BackgroundVerification))

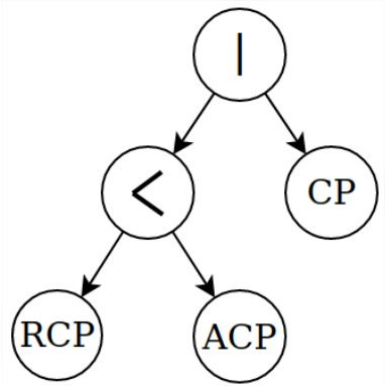
(UpdateInfoRecords, BackgroundVerification) < ((RequestAdditionalInfo < ReceiveSupportDocuments) | CreateAccount)

Detailed Solution of 1 – Step 4

These expressions are then **mapped to** their corresponding **abstract syntax trees (ASTs)**.

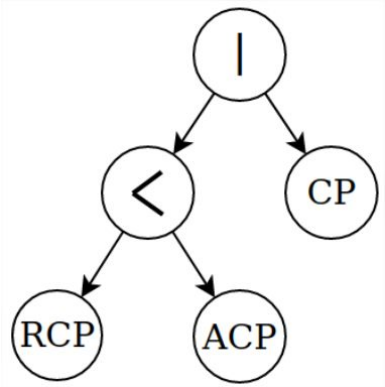
Detailed Solution of 1 – Step 4

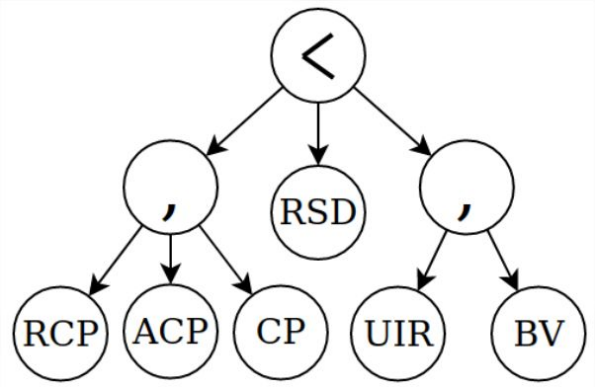
These expressions are then **mapped to** their corresponding **abstract syntax trees (ASTs)**.



Detailed Solution of 1 – Step 4

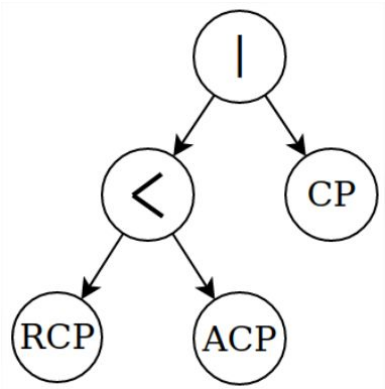
These expressions are then **mapped to** their corresponding **abstract syntax trees (ASTs)**.

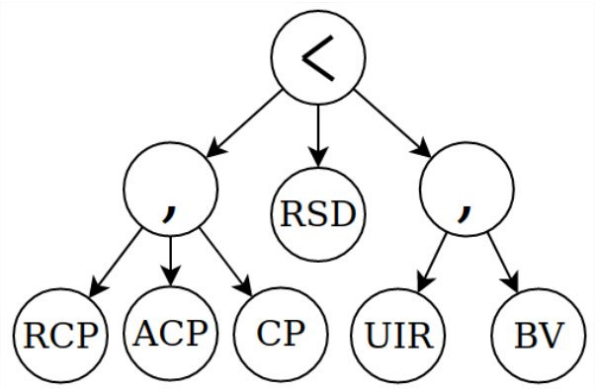


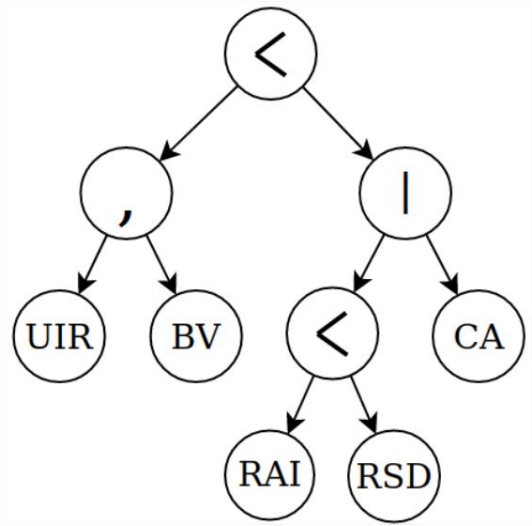


Detailed Solution of 1 – Step 4

These expressions are then **mapped to** their corresponding **abstract syntax trees (ASTs)**.

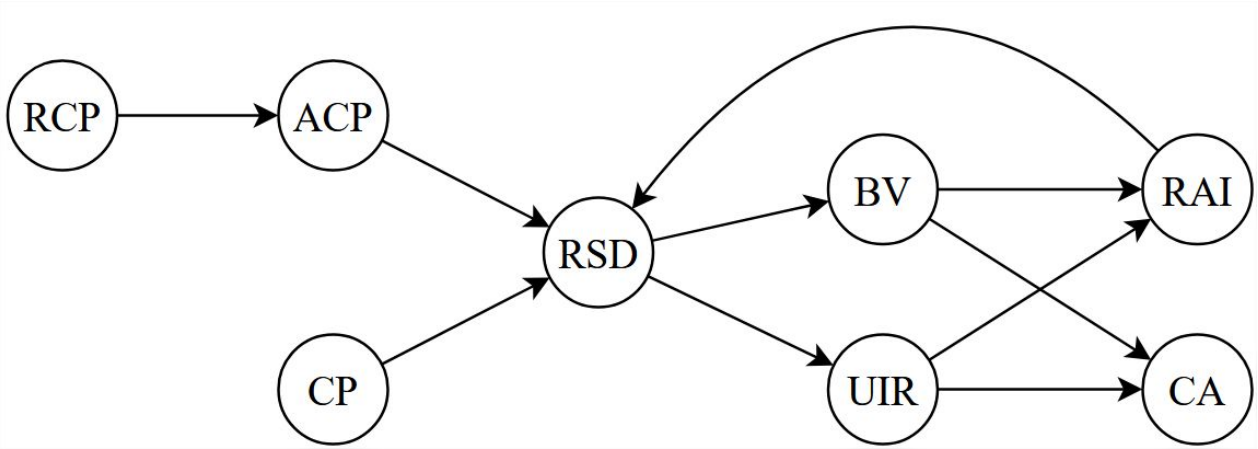






Detailed Solution of 1 – Step 5

The **sequential information** contained in the multiple ASTs is gathered to obtain a **cleaner** representation of it, called **dependency graph**.

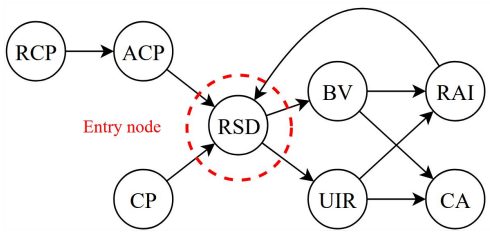


Detailed Solution of 1 – Step 6

If the dependency graph **contains loops**, they are **analysed**, and all the **information needed to reconstruct** them is extracted.

Detailed Solution of 1 – Step 6

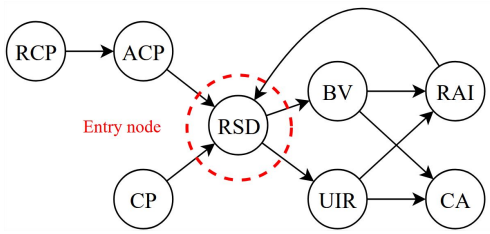
If the dependency graph **contains loops**, they are **analysed**, and all the **information needed to reconstruct** them is extracted.



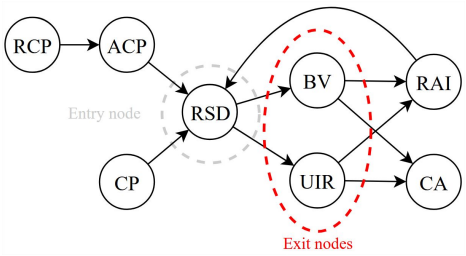
Entry node(s) computation

Detailed Solution of 1 – Step 6

If the dependency graph **contains loops**, they are **analysed**, and all the **information needed to reconstruct** them is extracted.



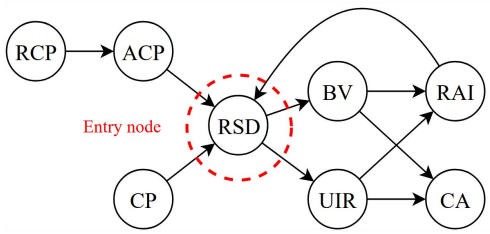
Entry node(s) computation



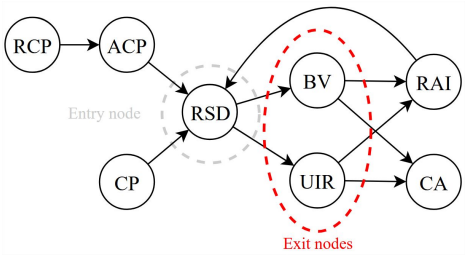
Exit node(s) computation

Detailed Solution of 1 – Step 6

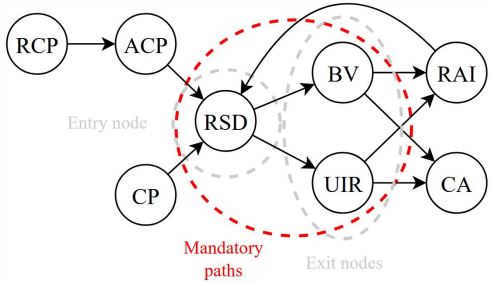
If the dependency graph **contains loops**, they are **analysed**, and all the **information needed to reconstruct** them is extracted.



Entry node(s) computation



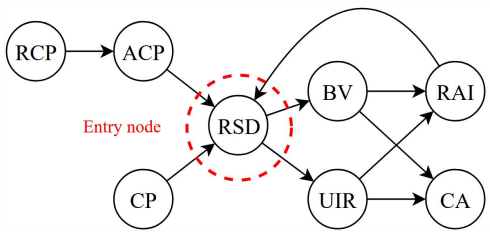
Exit node(s) computation



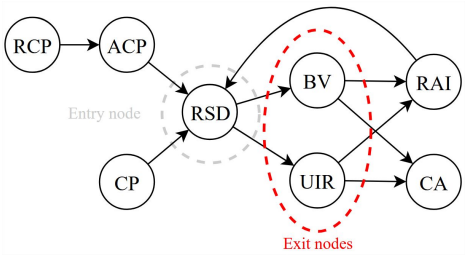
Mandatory path(s) computation

Detailed Solution of 1 – Step 6

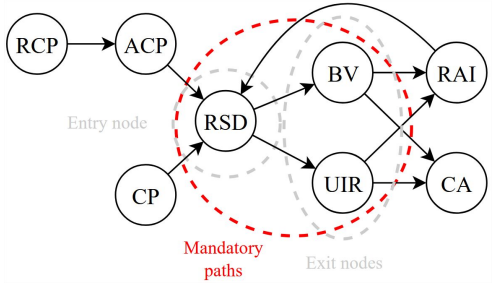
If the dependency graph **contains loops**, they are **analysed**, and all the **information needed to reconstruct** them is extracted.



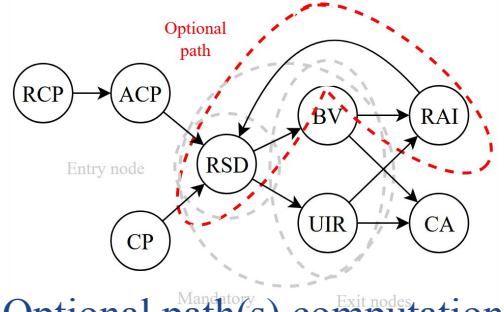
Entry node(s) computation



Exit node(s) computation



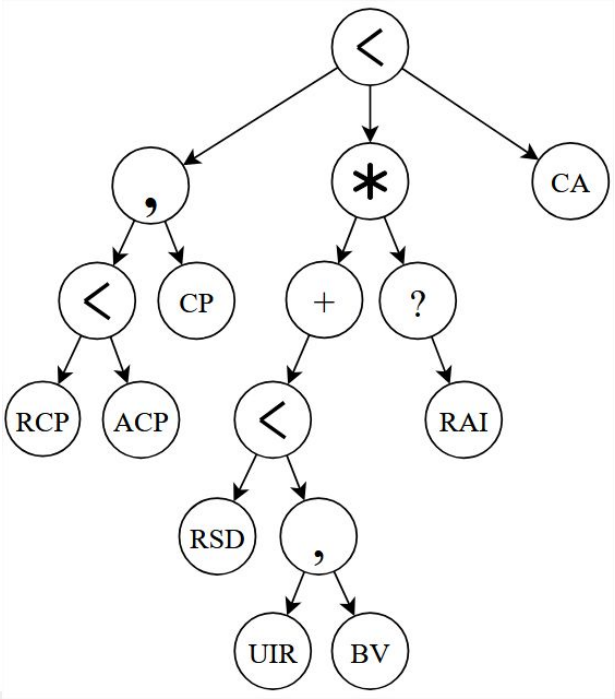
Mandatory path(s) computation



Optional path(s) computation

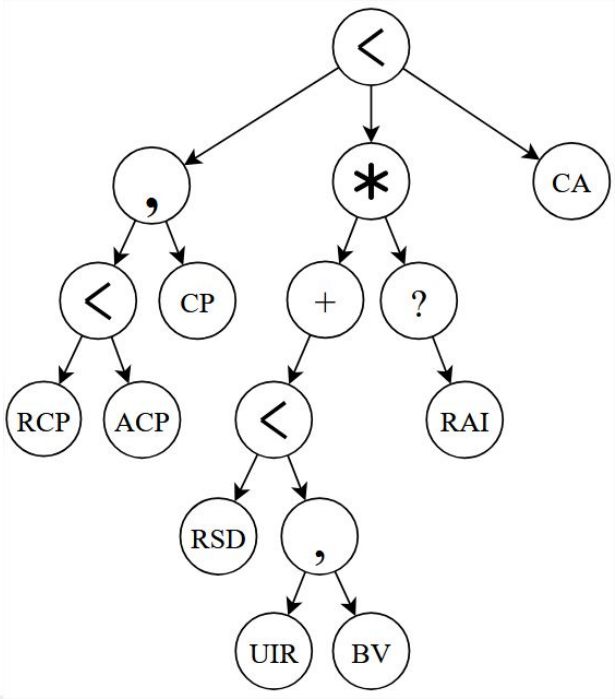
Detailed Solution of 1 – Step 7

Once the loops have been retrieved, the **AST corresponding to the dependency graph** is built from the dependency graph.



Detailed Solution of 1 – Step 8

The **ultimate step** to obtain an AST containing all the information belonging to the original expressions consists in **inserting the choices**.

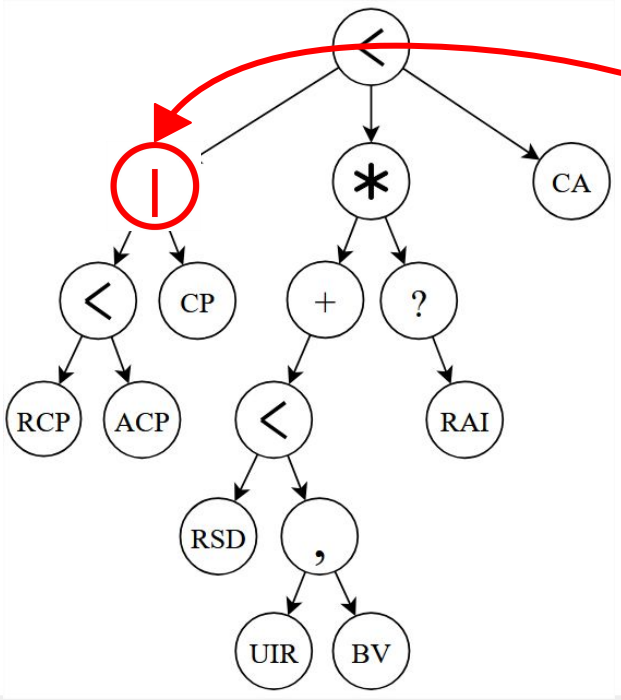


(RCP < ACP) | CP

(RAI < RSD) | CA

Detailed Solution of 1 – Step 8

The **ultimate step** to obtain an AST containing all the information belonging to the original expressions consists in **inserting the choices**.

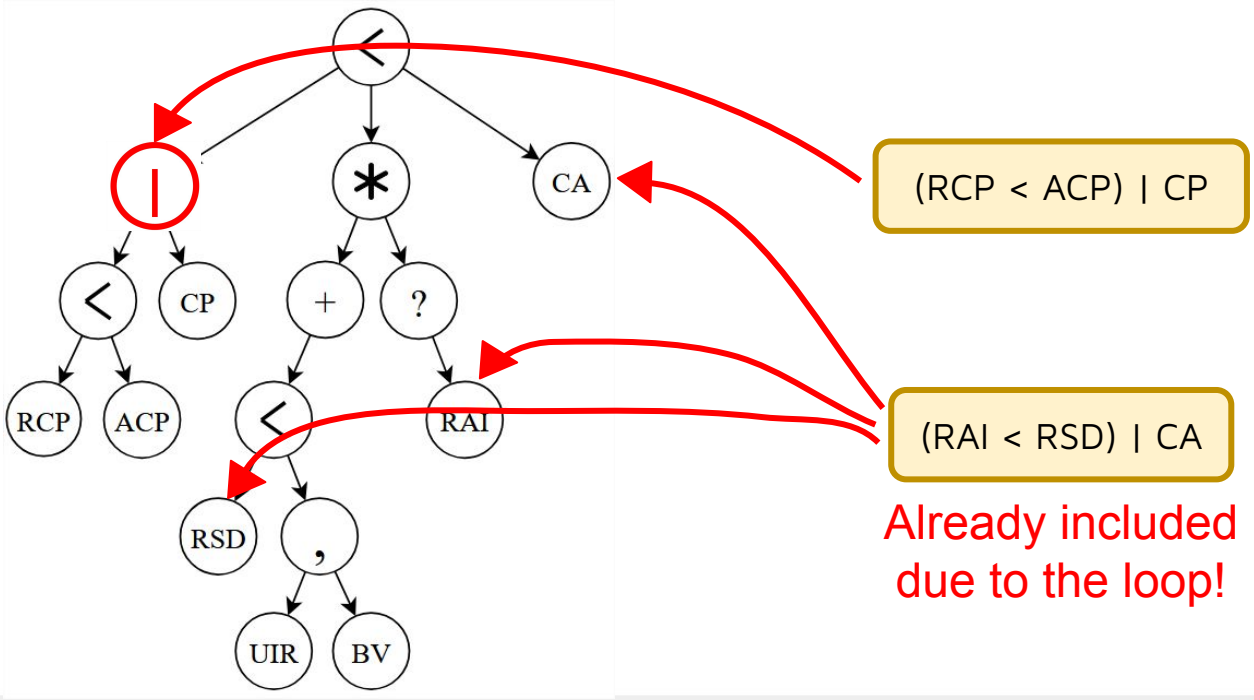


(RCP < ACP) | CP

(RAI < RSD) | CA

Detailed Solution of 1 – Step 8

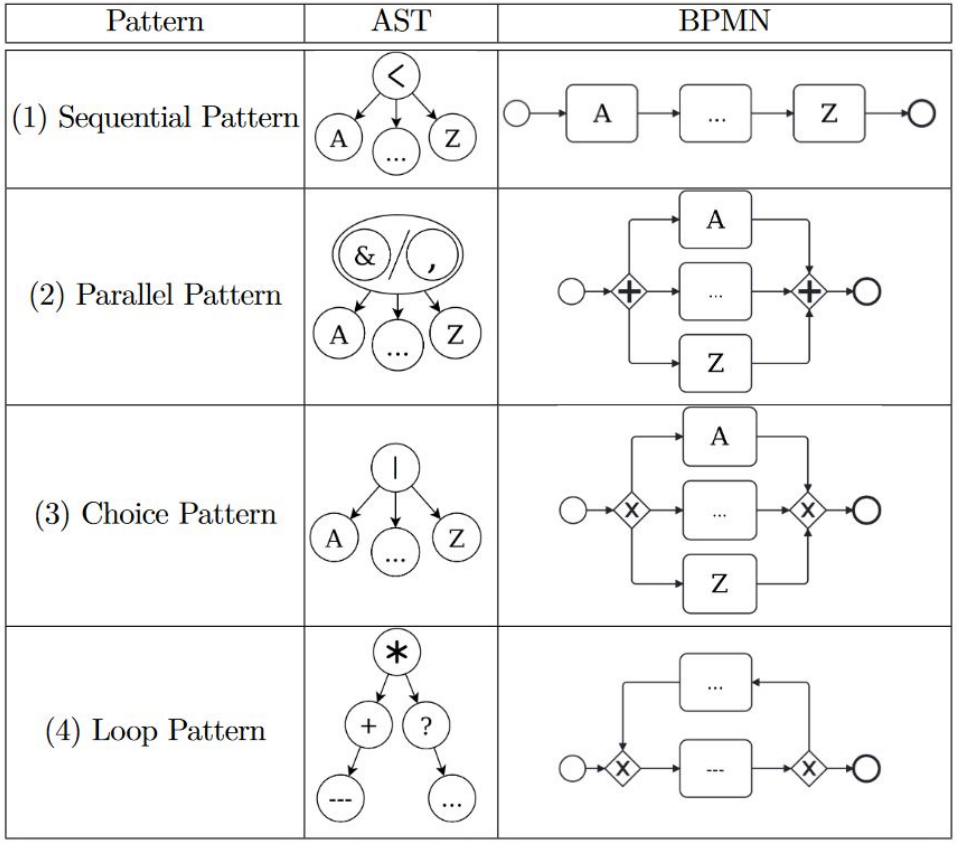
The **ultimate step** to obtain an AST containing all the information belonging to the original expressions consists in **inserting the choices**.



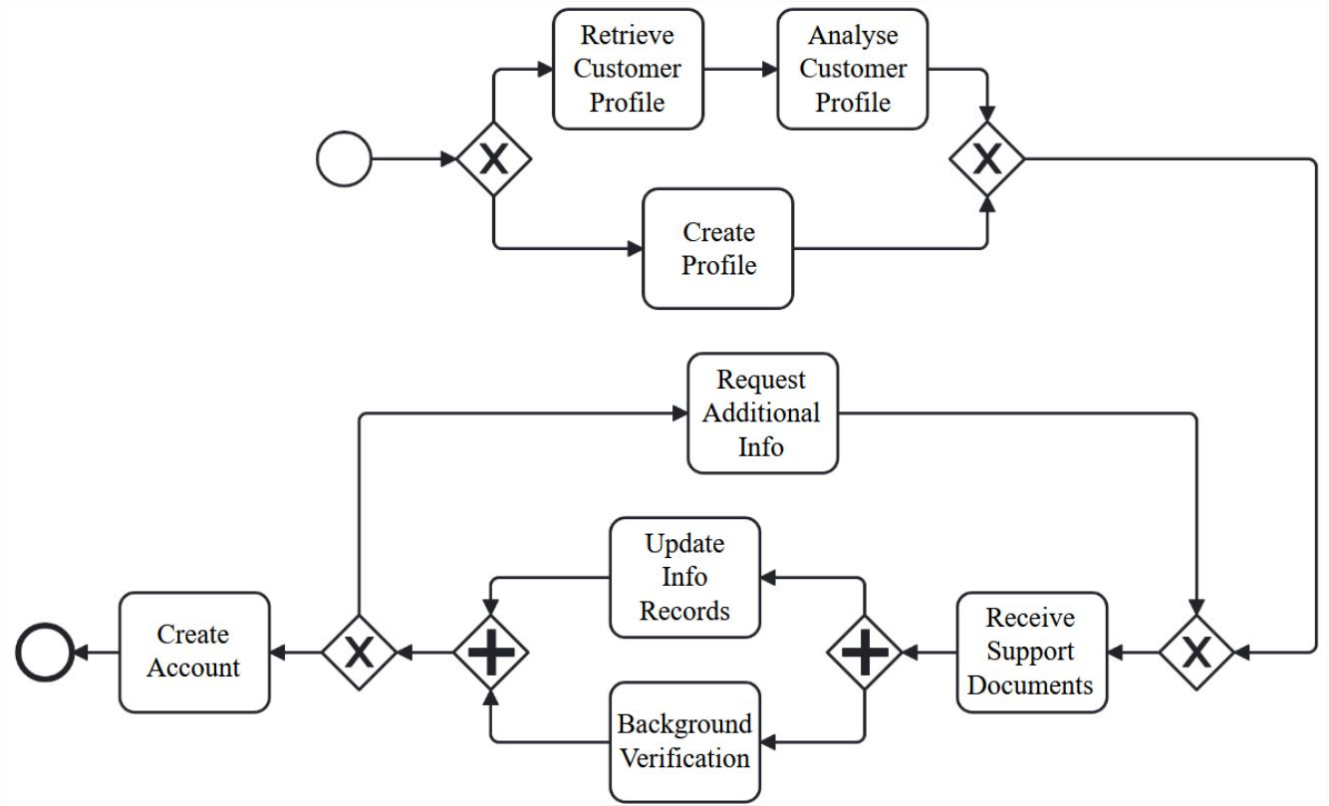
Detailed Solution of 1 – Step 9

Now that the AST contains the choices, the **BPMN process is ready to be generated**.

To do so, **patterns** are applied recursively to the merged AST, **starting from the deepest nodes** (leafs).



Detailed Solution of 1 – Result



Overview of our Solution for 3

BPMN Process

Overview of our Solution for 3

Overview of our Solution for 3

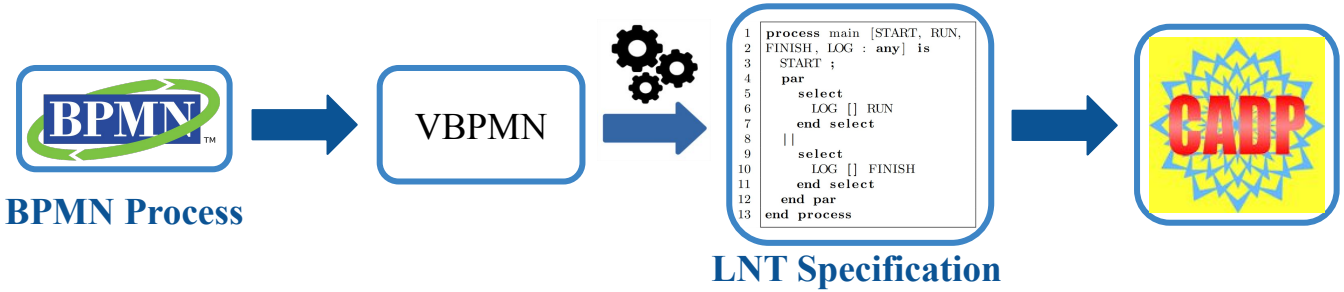
BPMN Process

VBPMN

```
process main [START, RUN,
FINISH, LOG : any] is
  START ;
  par
    select
      LOG [] RUN
    end select
  ||
    select
      LOG [] FINISH
    end select
  end par
end process
```

LNT Specification

Overview of our Solution for 3



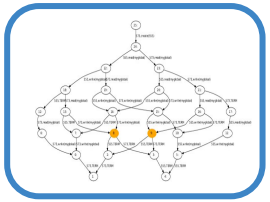
Overview of our Solution for 3

BPMN Process

VBPMN

```
process main [START, RUN,
FINISH, LOG : any] is
  START ;
  par
    select
      LOG [] RUN
    end select
  ||
    select
      LOG [] FINISH
    end select
  end par
end process
```

LNT Specification



BCG Specification

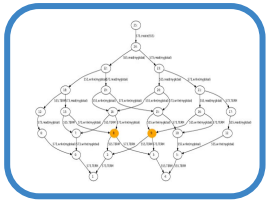
Overview of our Solution for 3

BPMN Process

VBPMN

```
process main [START, RUN,
FINISH, LOG : any] is
  START ;
  par
    select
      LOG [] RUN
    end select
  ||
    select
      LOG [] FINISH
    end select
  end par
end process
```

LNT Specification



BCG Specification

$$\Box\varphi \leftrightarrow \Box\Box\varphi$$
$$\Diamond\varphi \leftrightarrow \Diamond\Diamond\varphi$$
$$\Box\varphi \rightarrow \varphi \wedge \circ\varphi$$
$$\varphi \vee \bullet\varphi \rightarrow \Diamond\varphi$$
$$\Box(\varphi \rightarrow \bullet\varphi) \rightarrow \neg\varphi$$
$$\Box(\varphi_1 \wedge \varphi_2) \leftrightarrow \Box\varphi_1 \wedge \Box\varphi_2$$
$$\Box(\varphi_1 \rightarrow \varphi_2) \rightarrow (\Diamond\varphi_1 \rightarrow \Diamond\varphi_2)$$

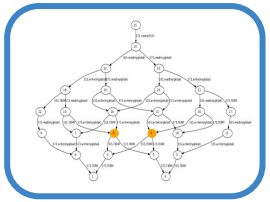
LTL Property

Overview of our Solution for 3

BPMN Process

```
process main [START, RUN,
FINISH, LOG : any] is
  START ;
  par
    select
      LOG [] RUN
    end select
  ||
    select
      LOG [] FINISH
    end select
  end par
end process
```

LNT Specification



BCG Specification

$$\begin{array}{c}
\Box\varphi \leftrightarrow \Box\Box\varphi \\
\Diamond\varphi \leftrightarrow \Diamond\Diamond\varphi \\
\Box\varphi \rightarrow \varphi \wedge \circ\varphi \\
\varphi \vee \bullet\varphi \rightarrow \Diamond\varphi \\
\Box(\varphi \rightarrow \bullet\varphi) \rightarrow \neg\varphi \\
\Box(\varphi_1 \wedge \varphi_2) \leftrightarrow \Box\varphi_1 \wedge \Box\varphi_2 \\
\Box(\varphi_1 \rightarrow \varphi_2) \rightarrow (\Diamond\varphi_1 \rightarrow \Diamond\varphi_2)
\end{array}$$

LTL Property

SPOT Toolbox

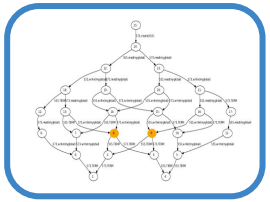
Overview of our Solution for 3

BPMN Process

VBPMN

```
process main [START, RUN,
FINISH, LOG : any] is
  START ;
  par
    select
      LOG [] RUN
    end select
  ||
    select
      LOG [] FINISH
    end select
  end par
end process
```

LNT Specification



BCG Specification

$$\square\varphi \leftrightarrow \square\square\varphi$$
$$\Diamond\varphi \leftrightarrow \Diamond\Diamond\varphi$$
$$\square\varphi \rightarrow \varphi \wedge \circ\varphi$$
$$\varphi \vee \bullet\varphi \rightarrow \Diamond\varphi$$
$$\square(\varphi \rightarrow \bullet\varphi) \rightarrow \neg\varphi$$
$$\square(\varphi_1 \wedge \varphi_2) \leftrightarrow \square\varphi_1 \wedge \square\varphi_2$$
$$\square(\varphi_1 \rightarrow \varphi_2) \rightarrow (\Diamond\varphi_1 \rightarrow \Diamond\varphi_2)$$

LTL Property

SPOT Toolbox

```
HOA: v1
name: "G!A"
States: 1
Start: 0
AP: 1 "A"
acc-name: Buchi
Acceptance: 1 Inf(0)
properties: trans-labels
properties: deterministic
--BODY--
State: 0
[!0] 0 {0}
--END--
```

Büchi Automaton
(HOA Format)

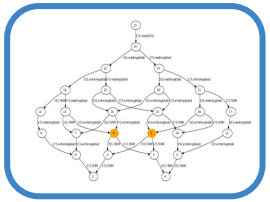
Overview of our Solution for 3

BPMN Process

VBPMN

```
process main [START, RUN,
FINISH, LOG : any] is
  START ;
  par
    select
      LOG [] RUN
    end select
  ||
    select
      LOG [] FINISH
    end select
  end par
end process
```

LNT Specification



BCG Specification

$$\begin{array}{c}
\Box\varphi \leftrightarrow \Box\Box\varphi \\
\Diamond\varphi \leftrightarrow \Diamond\Diamond\varphi \\
\Box\varphi \rightarrow \varphi \wedge \circ\varphi \\
\varphi \vee \bullet\varphi \rightarrow \Diamond\varphi \\
\Box(\varphi \rightarrow \bullet\varphi) \rightarrow \neg\varphi \\
\Box(\varphi_1 \wedge \varphi_2) \leftrightarrow \Box\varphi_1 \wedge \Box\varphi_2 \\
\Box(\varphi_1 \rightarrow \varphi_2) \rightarrow (\Diamond\varphi_1 \rightarrow \Diamond\varphi_2)
\end{array}$$

LTL Property

SPOT Toolbox

```
HOA: v1
name: "G!A"
States: 1
Start: 0
AP: 1 "A"
acc-name: Buchi
Acceptance: 1 Inf(0)
properties: trans-labels
properties: deterministic
--BODY--
State: 0
[!0] 0 {0}
--END--
```

Büchi Automaton
(HOA Format)

HOA2SVL

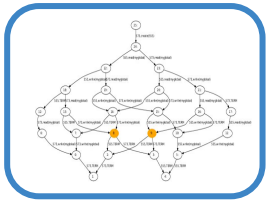
Overview of our Solution for 3

BPMN Process

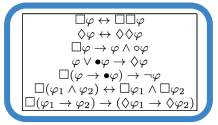
VBPMN

```
process main [START, RUN,
FINISH, LOG : any] is
  START ;
  par
    select
      LOG [] RUN
    end select
  ||
    select
      LOG [] FINISH
    end select
  end par
end process
```

LNT Specification



BCG Specification



$$\Box\varphi \leftrightarrow \Box\Box\varphi$$
$$\Diamond\varphi \leftrightarrow \Diamond\Diamond\varphi$$
$$\Box\varphi \rightarrow \varphi \wedge \circ\varphi$$
$$\varphi \vee \bullet\varphi \rightarrow \Diamond\varphi$$
$$\Box(\varphi \rightarrow \bullet\varphi) \rightarrow \neg\varphi$$
$$\Box(\varphi_1 \wedge \varphi_2) \leftrightarrow \Box\varphi_1 \wedge \Box\varphi_2$$
$$\Box(\varphi_1 \rightarrow \varphi_2) \rightarrow (\Diamond\varphi_1 \rightarrow \Diamond\varphi_2)$$

LTL Property

SPOT Toolbox

```
HOA: v1
name: "G!A"
States: 1
Start: 0
AP: 1 "A"
acc-name: Buchi
Acceptance: 1 Inf(0)
properties: trans-labels
properties: deterministic
--BODY--
State: 0
[!0] 0 {0}
--END--
```

Büchi Automaton
(HOA Format)

HOA2SVL

```
label par using
"i" || "i" -> "i",
"i" || "ACC" -> "i !ACC",
"C" || "C" -> "C",
"C" || "C !ACC" -> "C !ACC",
"B" || "B" -> "B",
"B" || "B !ACC" -> "B !ACC"
in "problem.bcg" || "buchi.bcg"
end par
```

.exp File

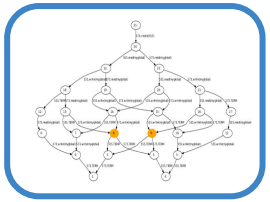
Overview of our Solution for 3

BPMN Process

VBPMN

```
process main [START, RUN,
FINISH, LOG : any] is
  START ;
  par
    select
      LOG [] RUN
    end select
  ||
    select
      LOG [] FINISH
    end select
  end par
end process
```

LNT Specification



BCG Specification

$$\square\varphi \leftrightarrow \square\square\varphi$$
$$\Diamond\varphi \leftrightarrow \Diamond\Diamond\varphi$$
$$\square\varphi \rightarrow \varphi \wedge \circ\varphi$$
$$\varphi \vee \bullet\varphi \rightarrow \Diamond\varphi$$
$$\square(\varphi \rightarrow \bullet\varphi) \rightarrow \neg\varphi$$
$$\square(\varphi_1 \wedge \varphi_2) \leftrightarrow \square\varphi_1 \wedge \square\varphi_2$$
$$\square(\varphi_1 \rightarrow \varphi_2) \rightarrow (\Diamond\varphi_1 \rightarrow \Diamond\varphi_2)$$

LTL Property

SPOT Toolbox

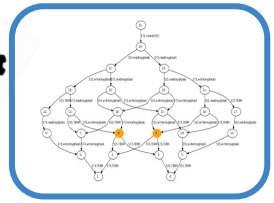
```
HOA: v1
name: "G!A"
States: 1
Start: 0
AP: 1 "A"
acc-name: Buchi
Acceptance: 1 Inf(0)
properties: trans-labels
properties: deterministic
--BODY--
State: 0
[!0] 0 {0}
--END--
```

Büchi Automaton (HOA Format)

HOA2SVL

```
label par using
"i" || "i" -> "i",
"i" || "ACC" -> "i !ACC",
"C" || "C" -> "C",
"C" || "C !ACC" -> "C !ACC",
"B" || "B" -> "B",
"B" || "B !ACC" -> "B !ACC"
in "problem.bcg" || "buchi.bcg"
end par
```

.exp File



Büchi Automaton (BCG Format)

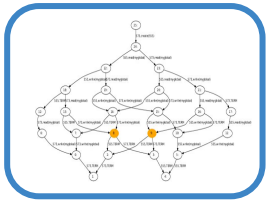
Overview of our Solution for 3

BPMN Process

VBPMN

```
process main [START, RUN,
FINISH, LOG : any] is
  START ;
  par
    select
      LOG [] RUN
    end select
  ||
    select
      LOG [] FINISH
    end select
  end par
end process
```

LNT Specification



BCG Specification

$$\square\varphi \leftrightarrow \square\square\varphi$$
$$\Diamond\varphi \leftrightarrow \Diamond\Diamond\varphi$$
$$\square\varphi \rightarrow \varphi \wedge \circ\varphi$$
$$\varphi \vee \bullet\varphi \rightarrow \Diamond\varphi$$
$$\square(\varphi \rightarrow \bullet\varphi) \rightarrow \neg\varphi$$
$$\square(\varphi_1 \wedge \varphi_2) \leftrightarrow \square\varphi_1 \wedge \square\varphi_2$$
$$\square(\varphi_1 \rightarrow \varphi_2) \rightarrow (\Diamond\varphi_1 \rightarrow \Diamond\varphi_2)$$

LTL Property

SPOT Toolbox

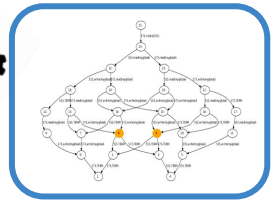
```
HOA: v1
name: "G!A"
States: 1
Start: 0
AP: 1 "A"
acc-name: Buchi
Acceptance: 1 Inf(0)
properties: trans-labels
properties: deterministic
--BODY--
State: 0
[!0] 0 {0}
--END--
```

Büchi Automaton (HOA Format)

HOA2SVL

```
label par using
"i" || "i" -> "i",
"i" || "ACC" -> "i !ACC",
"C" || "C" -> "C",
"C" || "C !ACC" -> "C !ACC",
"B" || "B" -> "B",
"B" || "B !ACC" -> "B !ACC"
in "problem.bcg" || "buchi.bcg"
end par
```

.exp File



Büchi Automaton (BCG Format)

```
% EXP_OPEN_OPTIONS="-hidden none"
"product.bcg" = generation of "task.exp";

"diag.bcg" = "product.bcg" |= [ true* . '.* !ACC' ] -|;
             result RES;
"diag.bcg" = total rename "DUMMY_LOOPY_LABEL" -> "i",
                          "DUMMY_LOOPY_LABEL !ACC" -> "i",
                          "\(.*\) !ACC" -> "\1"
             in "diag.bcg" end rename;
```

SVL Script

Overview of our Solution for 3

```
% EXP_OPEN_OPTIONS="-hidden none"
"product.bcg" = generation of "task.exp";

"diag.bcg" = "product.bcg" |= [ true* . '.* !ACC' ] -|;
             result RES;
"diag.bcg" = total rename "DUMMY_LOOPY_LABEL" -> "i",
                          "DUMMY_LOOPY_LABEL !ACC" -> "i",
                          "\(.*\) !ACC" -> "\1"
             in "diag.bcg" end rename;
```

SVL Script

Overview of our Solution for 3

```
% EXP_OPEN_OPTIONS="-hidden none"
"product.bcg" = generation of "task.exp";

"diag.bcg" = "product.bcg" |= [ true* . '.* !ACC' ] -|;
             result RES;
"diag.bcg" = total rename "DUMMY_LOOPY_LABEL" -> "i",
                          "DUMMY_LOOPY_LABEL !ACC" -> "i",
                          "\(.*\) !ACC" -> "\1"
             in "diag.bcg" end rename;
```

SVL Script

Overview of our Solution for 3

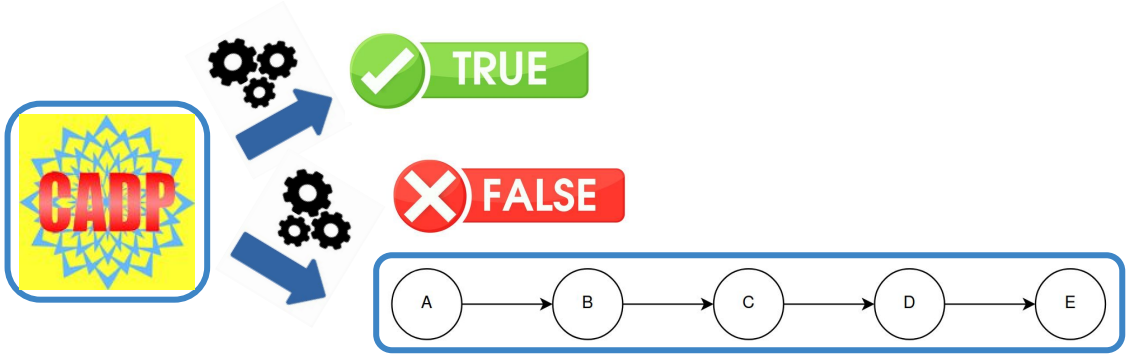
```
% EXP_OPEN_OPTIONS="-hidden none"
"product.bcg" = generation of "task.exp";

"diag.bcg" = "product.bcg" |= [ true* . '.* !ACC' ] -|;
             result RES;
"diag.bcg" = total rename "DUMMY_LOOPY_LABEL" -> "i",
                          "DUMMY_LOOPY_LABEL !ACC" -> "i",
                          "\(.*\) !ACC" -> "\1"
             in "diag.bcg" end rename;
```

SVL Script

Overview of our Solution for 3

```
% EXP_OPEN_OPTIONS="-hidden none"
"product.bcg" = generation of "task.exp";

"diag.bcg" = "product.bcg" |= [ true* . '.* !ACC' ] -|;
             result RES;
"diag.bcg" = total rename "DUMMY_LOOPY_LABEL" -> "i",
                          "DUMMY_LOOPY_LABEL !ACC" -> "i",
                          "\(.*\) !ACC" -> "\1"
             in "diag.bcg" end rename;
```

SVL Script

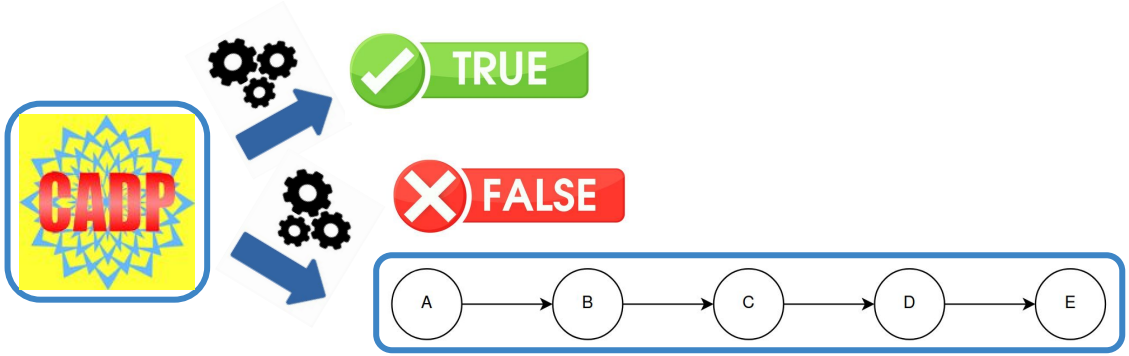


Counterexample

Overview of our Solution for 3

```
% EXP_OPEN_OPTIONS="-hidden none"
"product.bcg" = generation of "task.exp";

"diag.bcg" = "product.bcg" |= [ true* . '.* !ACC' ] -|;
             result RES;
"diag.bcg" = total rename "DUMMY_LOOPY_LABEL" -> "i",
                          "DUMMY_LOOPY_LABEL !ACC" -> "i",
                          "\(.*\) !ACC" -> "\1"
             in "diag.bcg" end rename;
```

SVL Script



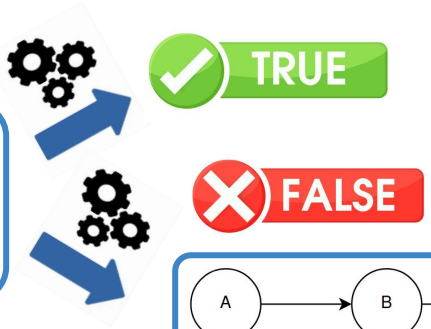
Counterexample

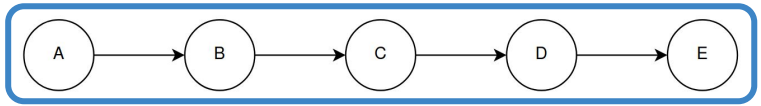
CLTS Builder

Overview of our Solution for 3

```
% EXP_OPEN_OPTIONS="-hidden none"
"product.bcg" = generation of "task.exp";

"diag.bcg" = "product.bcg" |= [ true* . '.* !ACC' ] -|;
             result RES;
"diag.bcg" = total rename "DUMMY_LOOPY_LABEL" -> "i",
                          "DUMMY_LOOPY_LABEL !ACC" -> "i",
                          "\(.*\) !ACC" -> "\1"
             in "diag.bcg" end rename;
```

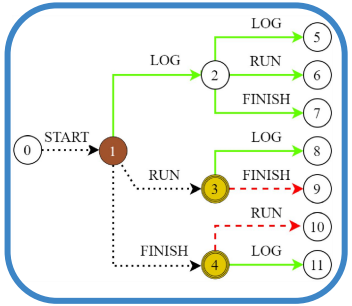
SVL Script





Counterexample

CLTS Builder

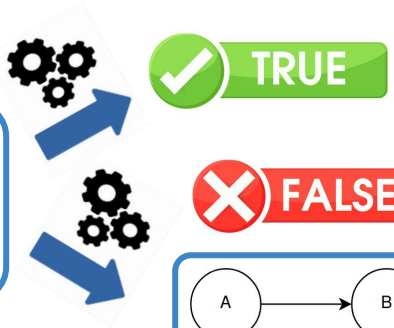


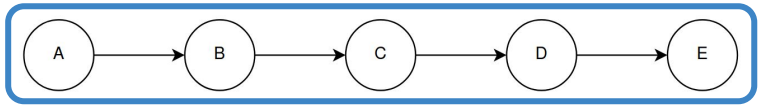
CLTS

Overview of our Solution for 3

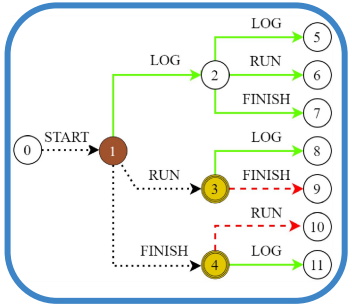
```
% EXP_OPEN_OPTIONS="-hidden none"
"product.bcg" = generation of "task.exp";

"diag.bcg" = "product.bcg" |= [ true* . '.* !ACC' ] -|;
             result RES;
"diag.bcg" = total rename "DUMMY_LOOPY_LABEL" -> "i",
                          "DUMMY_LOOPY_LABEL !ACC" -> "i",
                          "\(.*\) !ACC" -> "\1"
             in "diag.bcg" end rename;
```

SVL Script





Counterexample



CLTS

CLTS Builder

BPMN Colorator

Overview of our Solution for 3

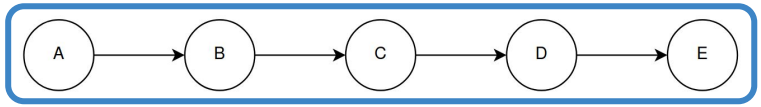
```
% EXP_OPEN_OPTIONS="-hidden none"
"product.bcg" = generation of "task.exp";

"diag.bcg" = "product.bcg" |= [ true* . '.* !ACC' ] -|;
             result RES;
"diag.bcg" = total rename "DUMMY_LOOPY_LABEL" -> "i",
                          "DUMMY_LOOPY_LABEL !ACC" -> "i",
                          "\(.*\) !ACC" -> "\1"
             in "diag.bcg" end rename;
```

SVL Script

CADP

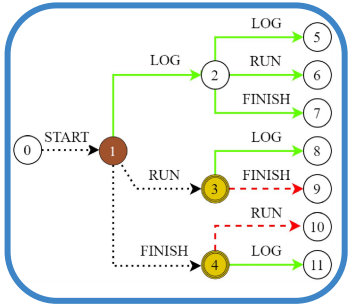
TRUE

FALSE



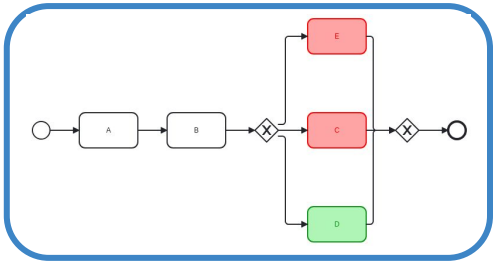
Counterexample

CLTS Builder



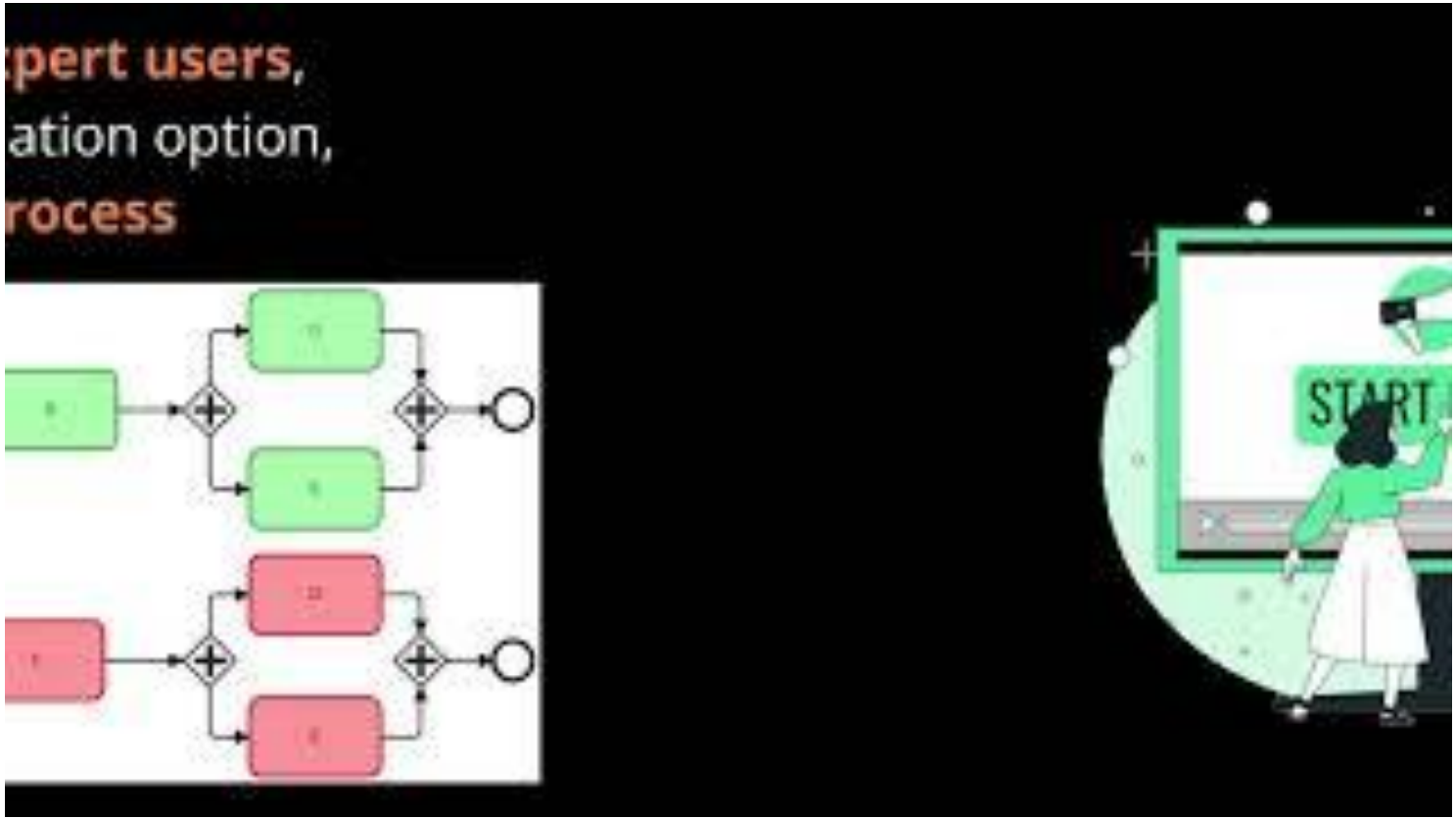
CLTS

BPMN Colorator



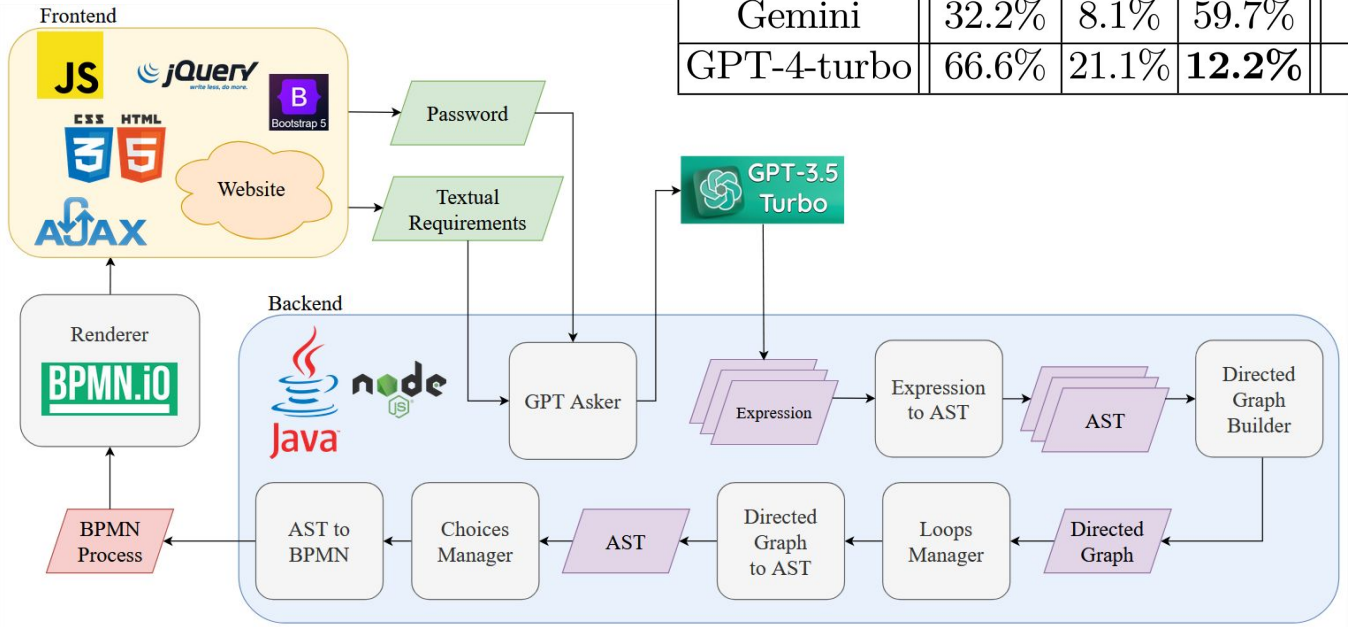
Colored BPMN Process

Tool – Presentation & Experiments



Tool – Presentation & Experiments

Tool/Model	✓	?	✗	Avg. Exec. Time (s)
Our tool	**78.5%**	**8%**	13.5%	**4.07**
ProMoAI	50%	8.7%	41.2%	24.7
Gemini	32.2%	8.1%	59.7%	8.32
GPT-4-turbo	66.6%	21.1%	**12.2%**	19.2



Tool – Presentation & Experiments

The LTL property is **correctly generated in 100%**.
Necessary otherwise the verification makes no sense.

BPMN Process	States	Trans.	BPMN Gen. Time	Prop. Gen. Time	Model Check. Time	CLTS Cons. Time	BPMN Colo. Time
Evisa App. [19]	30	31	1.67s	1.43s	4.35s	3.12s	613ms
Patient Diag. [3]	38	40	2.71s	1.14s	4.57s	3.29s	661ms
Employee Rec. [9]	39	40	1.89s	1.33s	4.57s	3.22s	830ms
Employee Hiring [6]	78	105	2.17s	2.86s	4.34s	3.22s	758ms
Perish. Goods Trans. [20]	108	150	2.22s	1.1s	4.78s	3.31s	978ms
Acc. Open. Proc. [17]	304	657	2.31s	1.23s	4.56s	4.23s	1.42s
Hard. Ret. Ship. Proc. [10]	373	819	2.56s	1.1s	4.53s	3.13s	764ms
Online Shipping [16]	375	765	2.78s	1.07s	5.12s	3.27s	1.7s
Handcrafted 1	279k	1.63m	3.25s	1.07s	8s	14.6s	17.2s
Handcrafted 2	1.67m	11m	1.79s	1.27s	23.9s	5.42m	24m
Handcrafted 3	10m	75m	1.67s	1.18s	3.05m	>1h	>1h
Handcrafted 4	60m	503m	2.08s	867ms	27.7m	>1h	>1h
Handcrafted 5	362m	3.32b	2.31s	1.85s	>1h	>1h	>1h

Tool – Presentation & Experiments

The LTL property is **correctly generated in 100%**.
Necessary otherwise the verification makes no sense.

BPMN Process	States	Trans.	BPMN Gen. Time	Prop. Gen. Time	Model Check. Time	CLTS Cons. Time	BPMN Colo. Time
Evisa App. [19]	30	31	1.67s	1.43s	4.35s	3.12s	613ms
Patient Diag. [3]	38	40	2.71s	1.14s	4.57s	3.29s	661ms
Employee Rec. [9]	39	40	1.89s	1.33s	4.57s	3.22s	830ms
Employee Hiring [6]	78	105	2.17s	2.86s	4.34s	3.22s	758ms
Perish. Goods Trans. [20]	108	150	2.22s	1.1s	4.78s	3.31s	978ms
Acc. Open. Proc. [17]	304	657	2.31s	1.23s	4.56s	4.23s	1.42s
Hard. Ret. Ship. Proc. [10]	373	819	2.56s	1.1s	4.53s	3.13s	764ms
Online Shipping [16]	375	765	2.78s	1.07s	5.12s	3.27s	1.7s
Handcrafted 1	279k	1.63m	3.25s	1.07s	8s	14.6s	17.2s
Handcrafted 2	1.67m	11m	1.79s	1.27s	23.9s	5.42m	24m
Handcrafted 3	10m	75m	1.67s	1.18s	3.05m	>1h	>1h
Handcrafted 4	60m	503m	2.08s	867ms	27.7m	>1h	>1h
Handcrafted 5	362m	3.32b	2.31s	1.85s	>1h	>1h	>1h

Starts getting (very) long on LTSs with millions of states

Conclusion

In this work, we proposed a **9 steps approach** aiming at automatically designing **syntactically and semantically correct BPMN** processes from a **textual description** of the requirements.

The main **perspectives** of this work are:

- Support unbalanced workflows
- Make use of more recent versions of GPT
- Diversify supported LTL properties
- Enrich the training dataset
- Enlarge the supported BPMN syntax