



AGH

Języki formalne i kompilatory
projekt
Generator

Informatyka Stosowana, 3 rok
WEAiE
AGH

Maciej Czekański
Krzysztof Mirek

Informacje ogólne

Program ma za zadanie generować formułę logiczną na podstawie wykrytych wzorców projektowych w schemacie BPMN oraz dostarczonych formuł logicznych odpowiadających wzorcom. Diagram jest tworzony w zewnętrznej aplikacji Paradigm Business Processes Visual Architect. Projekt jest napisany w Javie (SE7).

Funkcjonalność

Użytkownik może:

- wybrać plik wejściowy xml (otrzyma komunikat przy nieprawidłowej zawartości)
- wybrać plik wejściowy z formułami
- skonfigurować format wyjściowej formuły logicznej
- wybrać plik wyjściowy
- wywołać operację generowania

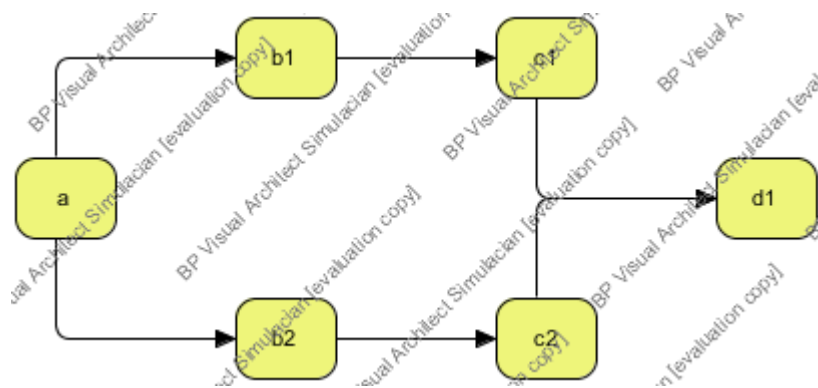
Wejście

Program wstępnie musi otrzymać na wejściu plik xml wygenerowany przez Paradigm BP Visual Architect.

```
<BPTask Compensation="false" CompletionQuantity="1" EnableIns  
  <ToSimpleRelationships>  
    <BPSequenceFlow Idref="cbQVfeyGAqGAUgPF" Name="" />  
    <BPSequenceFlow Idref="fWwVfeyGAqGAUgPJ" Name="" />  
  </ToSimpleRelationships>  
</BPTask>
```

Rozpoznawane węzły:

- **BPSequenceFlow** - przepływ
- **BPTask** - zadania
- **BPStartEvent** – początek
- **BPEndEvent** – koniec
- **BPPool** – pula
- **BPGateway** – bramka



Do parsowania pliku xml wykorzystamy technologię **JAXB** oraz pakiet **javax.xml**, ponieważ umożliwia bardzo szybkie i wydajne przetwarzanie (przykłady poniżej).

```

DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
dbf.setNamespaceAware(true);
dbf.setIgnoringComments(true);
dbf.setIgnoringElementContentWhitespace(true);

DocumentBuilder db;
try {
    db = dbf.newDocumentBuilder();
    Document dom = db.parse(filename);

    Element projElement = dom.getDocumentElement();
    // there is exactly one Model
    Element modelsElement = (Element) projElement.getElementsByTagName("Models").item(0);

    // Every task is represented by node in graph
    NodeList tasks = modelsElement.getElementsByTagName("BPTask");

    // Every sequence is connection between nodes
    XPathFactory xpathFactory = XPathFactory.newInstance();
    XPath xPath = xpathFactory.newXPath();
    XPathExpression xPathExpressionSequences = xPath.compile("//Models//ModelChildren//BPSequenceFlow");
    NodeList sequences = (NodeList) xPathExpressionSequences.evaluate(dom, XPathConstants.NODESET);
    parseSequences(sequences);
  }

```

Ilustracja 2: Parsowanie przepływu

- Ponadto należy dostarczyć plik z odpowiednimi formułami logicznymi. Przykładowy który użyliśmy.

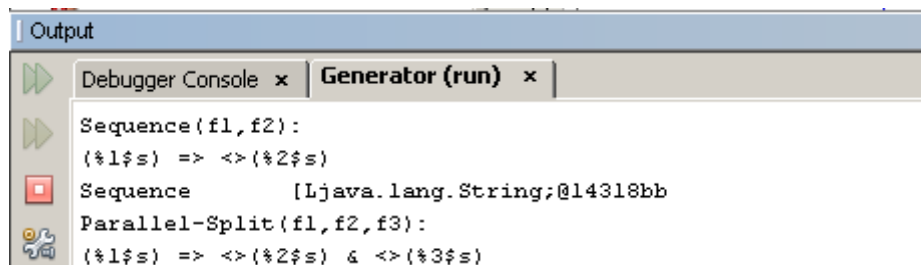
```
patterns-TL - Notatnik
Plik  Edycja  Format  Widok  Pomoc

/* version 24.07.2011
/* Basic Control Patterns
Sequence(f1,f2):
f1 => <>f2
Parallel-Split(f1,f2,f3):
f1 => <>f2 & <>f3
[]~(f1&(f2|f3))
Synchronization(f1,f2,f3):
f1 & f2 => <>f3
[]~(f3&(f1|f2))
Exclusive-Choice(f1,f2,f3):
f1 => (<>f2 & ~<>f3)|(~<>f2 & <>f3)
[]~(f2 & f3)
Simple-Merge(f1,f2,f3):
f1|f2 => <>f3
[]~(f3&(f1|f2))

/* others
Multi-Choice(f1,f2,f3):
f1 => <>f2|<>f3
Multi-Merge(f1,f2,f3,f4):
f1 => <>f2 & <>f3 & <>f4
f2 => <>f4
f3 => <>f4
Discriminator(f1,f2,f3,f4):
f1 => <>f2 & <>f3
f1|f2 => <>f4
Synchronizing-Merge(f1,f2,f3,f4):
f1 => (<>f2&~<>f3&<>f4)|(<>f2&<>f3&<>f4)
f2|f3 => <>f4
```

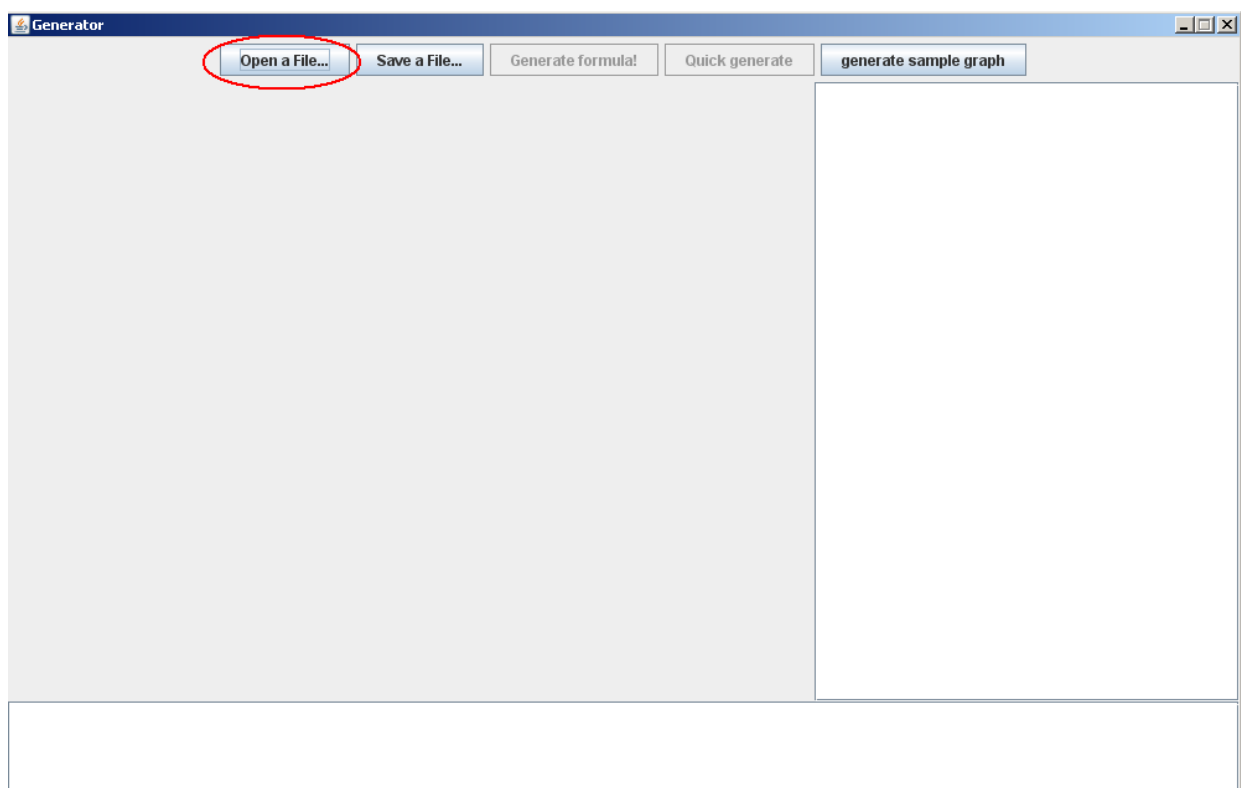
Działanie systemu

Inicjalizacja aplikacji składa się m.in. z parsowania pliku z formułami logicznymi. W konsoli jest wyświetlany postęp:

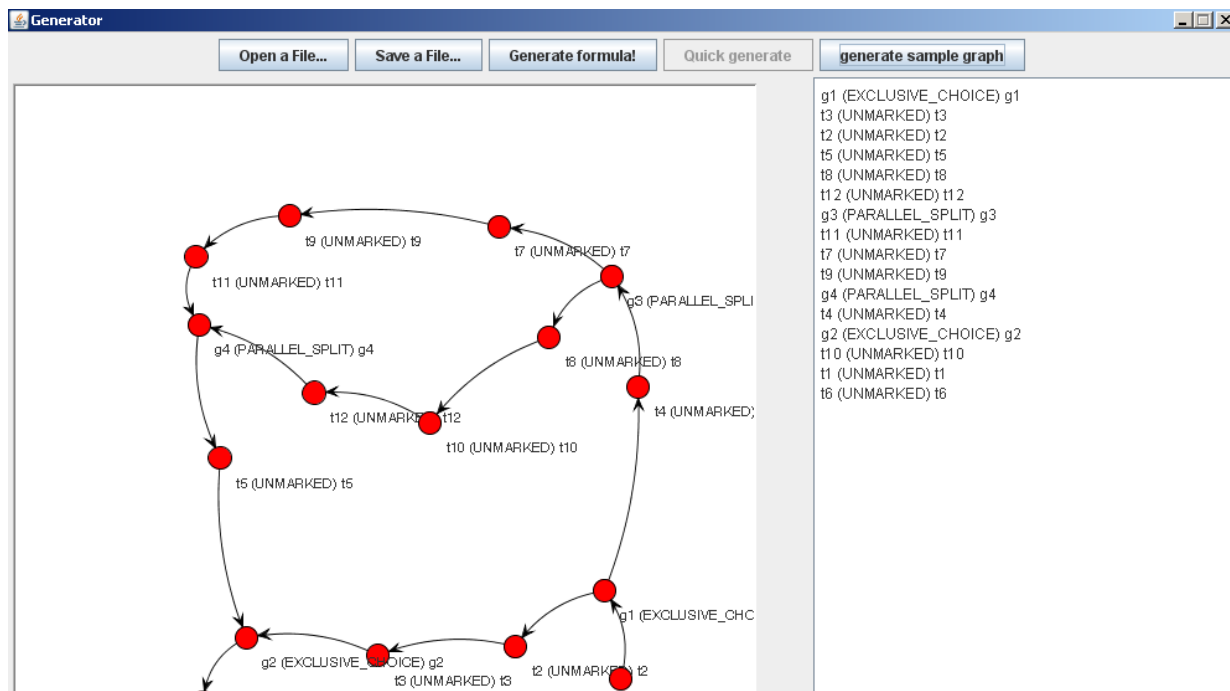


```
Output
Debugger Console x Generator (run) x
Sequence(f1, f2):
({1$s} => <>({2$s})
Sequence [Ljava.lang.String;@14318bb
Parallel-Split(f1, f2, f3):
({1$s} => <>({2$s}) & <>({3$s})
```

Następnie należy wybrać wejściowy plik .xml:

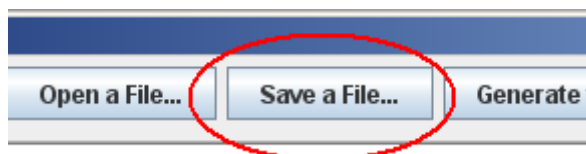


Po wybraniu pliku wejściowego na głównym ekranie pokaże się graf odwzorowujący strukturę:



W bocznym panelu jest wyświetlona lista węzłów z ich identyfikatorami i formułami, które przechowują.

Następnie należy wybrać plik wyjściowy, do którego będzie zapisany wynik:



Program udostępnia dwie opcje generowania formuły:



Generate formuła wykonuje jedną iterację algorytmu zwijania grafu. **Quick generate** zwija graf aż do pojedynczego węzła. Po odnalezieniu węzła na ekranie pozostaje jeden węzeł z formułą, a w konsoli jest wyświetlony odpowiedni komunikat:

1. (SEQUENCE) (t1) => <>((g1) => <>((t4) => <>((g3) => <>((t8) => <>((t10) => <>((t7) => <>((t9))) => <>((g4) & (t12) => <>((t11)) => <>((g2) & (t5) => <>((t3)) => <>((t6)))

2. === 0 iteracja ===
 3: 0 nowych markerow
 4. === 1 iteracja ===
 5: 0 nowych markerow
 6. === 2 iteracja ===
 7: 0 nowych markerow
 8: Odnaleziona formuła:
 9: (t1) => <>((g1) => <>((t4) => <>((g3) => <>((t8) => <>((t10) => <>((t7) => <>((t9))) => <>((g4) & (t12) => <>((t11)) => <>((g2) & (t5) => <>((t3)) => <>((t6)))

W tym samym momencie formuła jest automatycznie zapisana do wcześniej wybranego pliku:

```
output - Notatnik
Plik Edycja Format Widok Pomoc
(t1) => <>(((g1) => <>(((t4) => <>(((g3) => <>((t8) => <>((t10) => <>((t7) => <>((t9))) => <>((g4) & (t12) => <>((t11)) => <>((g2) & (t5) => <>((t3)) => <>((t6)))
```

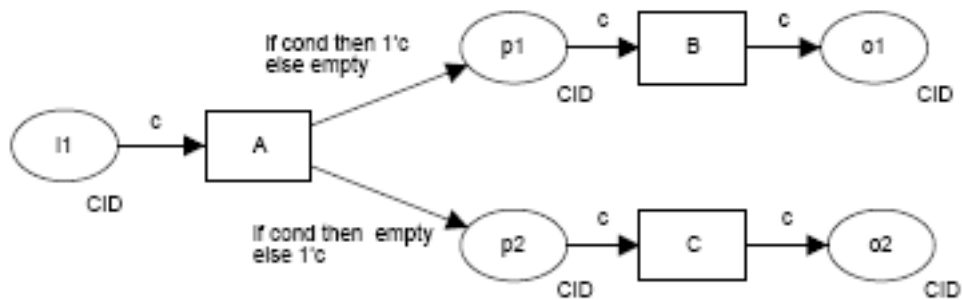
Struktura systemu

System ma rozbudowaną logikę wewnętrzną pozwalającą na odrębne wykorzystanie modułów. Utworzone moduły to:

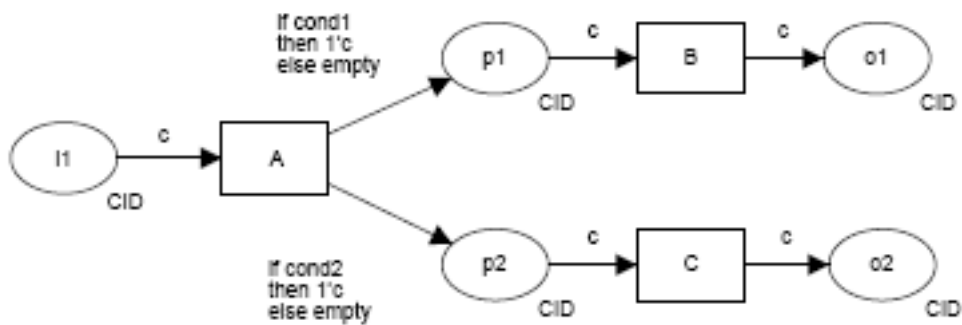
- **VisualParadigmXmlParser** – służy do parsowania plików eksportowanych przez Virtual Paradigm Architect
- **PatternFinder** – wyszukuje wzorce i odpowiednio modyfikuje graf
- **FormulaParser** – parsuje plik wejściowy z formułami odpowiadającymi wzorcom
- **Graph** – nakładka na bibliotekę JUNG (Java Universal Net-Graph Framework) służąca do obsługi grafów. Dostosowana do wymagań projektu
- **AppView** – warstwa prezentacji w technologii Java Swing
- **AppControler** – kontroler aplikacji spajający w całość wszystkie niezależnie działające moduły

Rozpoznawane wzorce

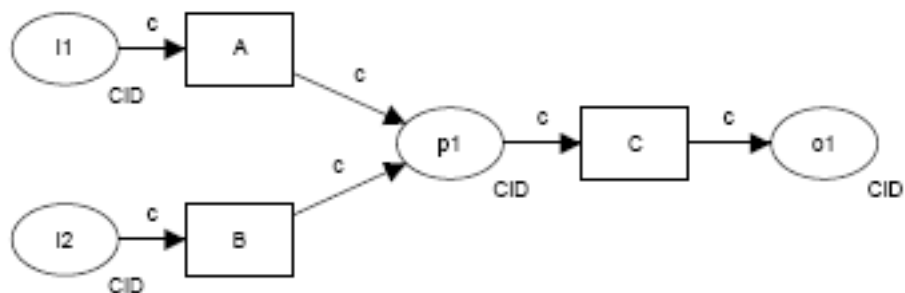
Program na podstawie pliku xml odtwarza schemat i wyszukuje w nim następujące wzorce:



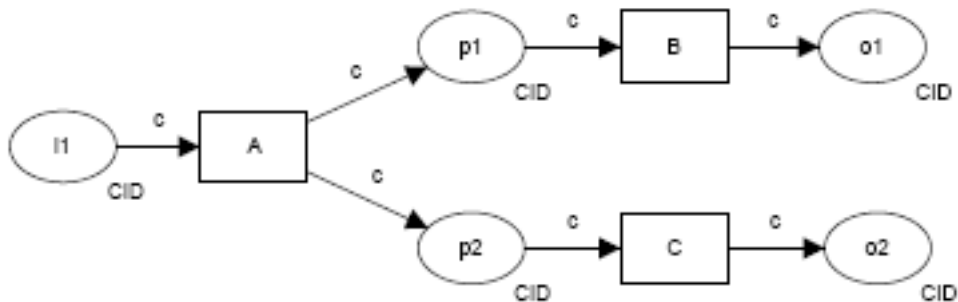
exclusive choice pattern



multi-choice pattern



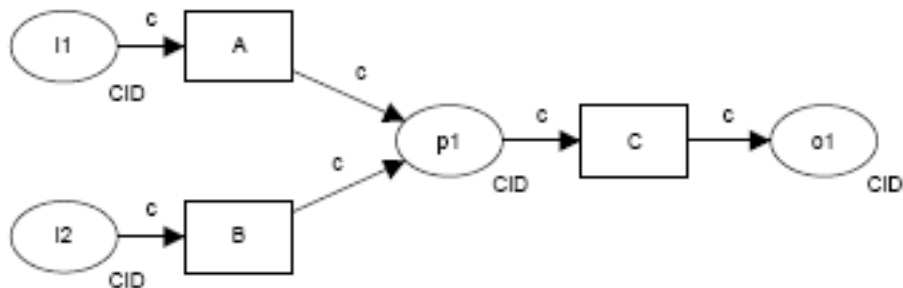
multi-merge pattern



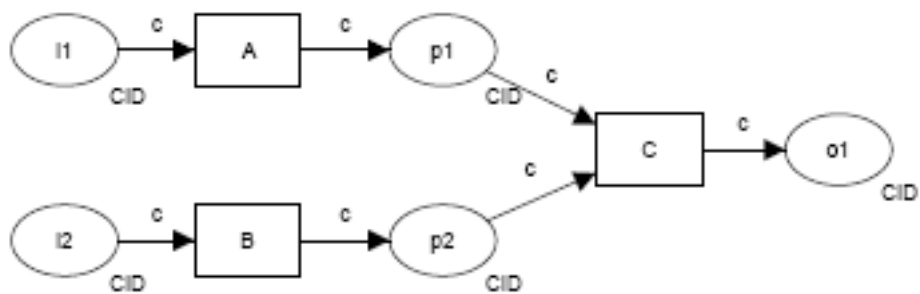
parallel split pattern



sequence pattern



simple merge pattern



synchronization pattern

Następnie zidentyfikowaniu wzorców i dostarczeniu formuły logicznej tworzony jest plik wynikowy.

Wynik

Na wyjściu program tworzy plik o podanej nazwie przez użytkownika, w którym jest zapisana wygenerowana formuła logiczna. Jest to zwykły plik tekstowy, w którym symbole logiczne są zakodowane odpowiednikami znakowymi zdefiniowanymi w pliku wejściowym. Plik może być następnie wykorzystany przez program parsujący formuły logiczne, stwierdzający ich poprawność (P)