Правительство Российской Федерации

Национальный исследовательский университет Высшая школа экономики

Отделение Программной инженерии

Кафедра Управления разработкой программного обеспечения

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

На тему: Построение BPMN-модели процесса, поведенчески эквивалентной заданной каузальной сети

Студент группы №472ПИ
_____ / Гундобин Н.А. /
« » мая 2014 г.

Руководитель ВКР
доцент, к.т.н.
_____ / Каленкова А.А. /
« » мая 2014 г.

Москва, 2014

Аннотация

Дисциплина Process Mining, изучает взаимосвязь лога событий (event log) некоторого процесса и модели этого процесса. В рамках дисциплины есть ряд задач, одной из которых является извлечение модели процесса из лога (process discovery). В результате извлечения может получиться модель с множественными процессными аномалиями, что мешает дальнейшей работе с моделью. Для решения этой проблемы используются каузальные сети (causal nets, C-nets), семантика которых игнорирует подобные аномалии. Однако модель, представленная в виде каузальной сети, может быть не очевидна для аналитиков. С другой стороны, BPMN-нотация является де-факто стандартом в области моделирования бизнес-процессов и понятна широкому кругу аналитиков. Таким образом, задача преобразования каузальной сети в BPMN-модель является актуальной.

Цель работы — разработка алгоритма построения BPMN-модели, симулирующей поведение заданной каузальной сети. Результатом работы является формальное описание алгоритма, доказательство его корректности на произвольной каузальной сети. Также алгоритм был реализован в процессно-ориентированной информационной системе ProM в рамках плагина BPMNConversions и используется исследователями.

Ключевые слова: process mining, bpmn, каузальная сеть, система переходов, ProM.

Оглавление

Введение

Process mining (извлечение моделей процессов) – относительно новая и стремительно развивающаяся дисциплина, изучающая взаимосвязь журнала (лога) событий некоторого бизнес-процесса и модели этого процесса, описанная в [4]. Лог процесса (*process log*) представляет собой последовательность сценариев (*cases*), каждый из которых является упорядоченной последовательностью действий (*activities*), происходящих в рамках некоторой деятельности. Также логи могут хранить некоторую метаинформацию: время выполнения действий, исполнителей, стоимость, и т.д. Запись лога в реальных бизнес-процессах обычно осуществляется специальной информационной системой; количество записей может быть значительным, что делает анализ «вручную» крайне неэффективным. Таким образом, *process mining* как дисциплина лежит на стыке машинного обучения и *data mining* с одной стороны и моделированием процессов – с другой.

Важно отметить, что основная идея дисциплины — это обнаружение, отслеживание и улучшение *реальных* процессов с помощью информации, находящейся в логе. В итоге, проецируя данные цели на отношения между логом и моделью процесса, можно определить три основные задачи process mining: извлечение модели процесса (discovery), проверка согласованности модели и лога (conformance checking) и улучшение модели (enhancement). Эти задачи представлены на рис. 1.

В рамках задачи извлечения разработано множество моделей и алгоритмов, работающих с этими моделями. Каждая модель обладает собственной семантикой и соответственно некоторыми особенностями и огра-

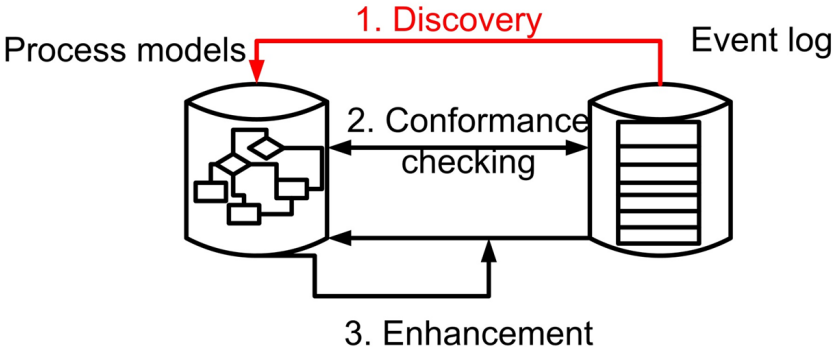

Рис. 1: Основные задачи process mining

ничениями. Подобные особенности образуют *смещение представления* (representational bias), которое значительно влияет на границы моделируемого поведения.

BPMN 2.0 (BPMN) – Business Process Model and Notation (Модель и Нотация Бизнес Процессов) [1] – является стандартом де-факто для моделирования бизнес-процессов. BPMN-модели понятны широкой аудитории аналитиков и исследователей. К тому же BPMN-модели обладают рядом положительных особенностей: модели могут быть импортированы и выполнены в любой BPMN-ориентированной программной системе; модели подходят для интеграции с точки зрения различных перспектив (от перспективы потока управления до перспективы ресурсов). Таким образом, BPMN-модель позволяет получить целостный взгляд на исследуемый процесс и подходит для человеческого анализа.

С другой стороны, каузальные сети (*causal nets, C-nets*) – одна из основных моделей, рассматриваемая в рамках этой работы и широко используемая в системе ProM [5] – основаны на декларативной семантике, обеспечивающей только *корректное* поведение, то есть свободное от различных процессных аномалий: взаимных блокировок (*deadlocks*), «зацикливаний» (*livelocks*) и других ошибок синхронизации.

Вследствие игнорирования последовательностей с ошибками синхронизации каузальные сети являются удобным представлением результата извлечения модели процесса из лога. Однако это представление может быть не очевидным для исследователей: даже простые процессы, пред-

ставленные с помощью каузальных сетей, требуют значительных усилий для анализа поведения этих процессов.

Подобная ориентированность рассмотренных выше моделей на различные цели делает актуальной задачу преобразования каузальной сети в BPMN-модель без потери поведения процесса. Стоит отметить, что в отличие от каузальных сетей BPMN допускает в моделях процессные аномалии. Из-за различия в семантике моделей существуют серьезные ограничения, при которых невозможно гарантировать отсутствие некорректного поведения в результирующей BPMN-модели.

1 Обзор используемых моделей

1.1 Сети Петри и WF-сети

В работе используются понятия *сетей Петри* и *workflow-сетей (WF-сетей)* для оценки семантики каузальных сетей и BPMN-моделей.

Сеть Петри (*Petri net*) – это тройка $PN = (P, T, F)$, где P и T – непересекающиеся множества *позиций* и *переходов*, образующие множество вершин, а $F \subseteq (P \times T) \cup (T \times P)$ – множество потоков управления.

Пусть E – конечное множество событий, τ – специальное *скрытое* событие, и $\lambda : T \to E \cup \{\tau\}$ – функция пометки. Тогда $PN = (P, T, F, \lambda)$ – *помеченная* сеть Петри.

Маркировка сети Петри – это мультимножество над множеством позиций. Маркированная сеть Петри (PN, m_0) – это сеть Петри со своей начальной маркировкой.

Графически позиции обычно представлены кружками, переходы – квадратами, а потоки из F – направленными дугами. Позиции могут содержать *фишки* (*tokens*), представленные закрашенными кружками. Текущая маркировка m обозначается количеством фишек $m(p)$ в каждой позиции $p \in P$.

Для любой вершины $n \in P \cup T$ дуга (x, n) называется *входящей дугой*, а дуга (n, x) – *исходящей дугой*; *предусловие* (*preset*) ${}^\bullet n$ и *постусловие* (*postset*) $n^\bullet$ определяются как мультимножества над $P \cup T$, такие, что ${}^\bullet n(x) = 1$, если $(x, n) \in F$, иначе ${}^\bullet n(x) = 0$, и $n^\bullet(x) = 1$ если $(n, x) \in F$, иначе $n^\bullet(x) = 0$. Заметим, что предусловия и постусловия являются множеством вершин.

Переход $t \in T$ называется *активным* (*enabled*) в маркировке m тогда и только тогда, когда ${}^\bullet t \subseteq m$. Активный переход t может *сработать* (*fire*), производя новую маркировку $m' =_{\text{def}} m - {}^\bullet t + t^\bullet$ (обозначается $m \xrightarrow{t} m'$, $m \xrightarrow{\lambda(t)} m'$, или $m \to m'$).

Маркировка m' называется *достижимой* (*reachable*) из m, если и только если существует (возможно, пустая) последовательность срабатываний $m = m_1 \to \cdots \to m_n = m'$.

$\mathcal{R}(PN, m)$ обозначает множество всех маркировок, достижимых в PN из маркировки m.

(Помеченная) Сеть Петри PN называется (помеченной) *WF-сетью* тогда и только тогда, когда:

1. Существует единственная начальная позиция (*source place*) $i \in P$ и единственная конечная позиция (*sink place*) $o \in P$, причем i не имеет входящих дуг, а o – исходящих;
2. Каждая вершина из $P \cup T$ лежит на пути из i в o;
3. Начальная маркировка в PN содержит единственную фишку в i.

1.2 Каузальные сети

Модель каузальных сетей, представленная в [3], определяется следующим образом.
Каузальная сеть (Causal net, C-net) – это кортеж $C = (A, a_i, a_o, D, I, O)$, где

- A - множество действий;
- a_i, $a_o \in A$ – начальное и конечное действия;
- $D \subseteq A \times A$ – отношение зависимостей;
- $AS = \{X \subseteq \mathcal{P}(A)^1 | X = \{\emptyset\} \vee \emptyset \notin X\}$;

[1] $\mathcal{P}(A) = \{A' \mid A' \subseteq A\}$ - показательное множество A. Таким образом, элементы AS – множества множеств действий.

– $I \in AS$ определяет множество возможных входных связей;

– $O \in AS$ – множество возможных выходных связей,

такие, что:

– $\mathrm{D} = \{(a_1, a_2) \in A \times A | a_1 \in \bigcup_{as \in I(a_2)} as,\ a_2 \in \bigcup_{as \in O(a_1)} as\}$;

– $\{a_i\} = \{a \in A | I(a) = \{\emptyset\}\}$;

– $\{a_o\} = \{a \in A | O(a) = \{\emptyset\}\}$;

– все действия в графе (A, D) лежат на пути от a_i до a_o.

Таким образом, каузальная сеть состоит из множества действий, связанных между собой каузальными зависимостями. Каузальные зависимости основаны на понятии связи. Пусть имеется сеть C, тогда *связью (activity binding)* называется $b \in B = \{(a, as^I, as^O) \in A \times \mathcal{P}(A) \times \mathcal{P}(A)\}$. *Последовательность связей* $\sigma \in B^*$ – это упорядоченное множество связей.

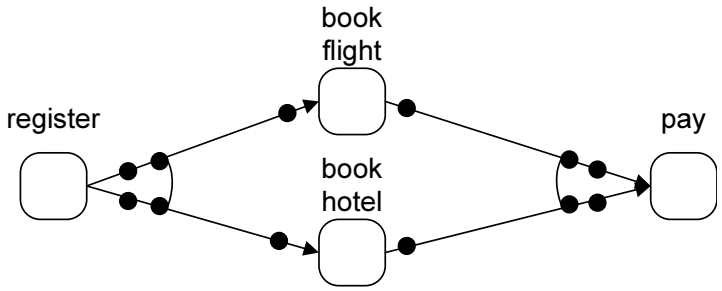

Рис. 1.1: Каузальная сеть, моделирующая процесс бронирования

Рассмотрим C-net на рис. 1.1. Данная сеть моделирует процесс бронирования билетов на самолет и/или места в отеле. Сеть содержит действия регистрации ($register$), бронирования билета на самолет ($book\ flight$), места в отеле ($book\ hotel$) и оплаты (pay). Ребра графа отражают возможность следования одного действия за другим, а черные точки с дугами определяют множество входных и выходных связей действия. Например, действие $register$ имеет 3 выходных связи: $\{book\ flight\}$, $\{book\ hotel\}$ и

$\{book\ flight, book\ hotel\}$, и ни одной входной, а действие *book hotel* имеет по одной входной и выходной связи.

Для последовательности связей каузальной сети определено понятие состояния. *Состояние* ψ задается рекурсивно на множестве последовательностей связей: $\psi(\langle\rangle) = []$, $\psi(\sigma \oplus (a, as^I, as^O)) = (\psi(\sigma) \backslash (as^I \times \{a\})) \uplus (a \times as^O)$.[2]

Состояние в каузальной сети задает множество *«задолженностей»* для каждой последовательности связей. «Задолженность» является парой действий (a, b) и означает, что для корректного завершения процесса необходимо выполнение связи вида (b, as_b^I, as_b^O), где $a \in as_b^I$.

В отличие от многих моделей процессов семантика каузальных сетей является декларативной а не операционной, то есть последовательности связей рассматриваются целиком, и не вводится понятие фишки, как, например, в сетях Петри. Более того, рассматриваемое поведение C-net ограничивается только *корректными* последовательностями связей.

Последовательность связей $\sigma = \langle (a_1, as_1^I, as_1^O), ..., (a_n, as_n^I, as_n^O) \rangle$ для каузальной сети C является корректной, если выполняются следующие условия:

- $a_1 = a_i$, $a_n = a_o$ и $a_k \in A$ a_i, a_o, $1 < k < n$;
- $\psi(\sigma) = []$;
- для любого не пустого префикса $\sigma' = \langle (a_1, as_1^I, as_1^O), ..., (a_k, as_k^I, as_k^O) \rangle$, $1 \leq k \leq n$ следует, что $(as_k^I \times a_k) \subseteq \psi(\sigma'')$, где $\sigma'' = \langle (a_1, as_1^I, as_1^O), ..., (a_{k-1}, as_{k-1}^I, as_{k-1}^O) \rangle$.

Множество корректных последовательностей связей сети C обозначается $V_{CN}(C)$.

Рассмотрим отношение между каузальными сетями и сетями Петри (WF-сетями в частности). Любая устойчивая WF-сеть может быть преобразована в эквивалентную[3] C-net, однако не каждая каузальная сеть

[2] $\oplus$ – операция присоединения элемента (или упорядоченного множества) к концу упорядоченного множества. Например, $\{a, b\} \oplus c = \{a, b, c\}$. $\uplus$ – операция объединения двух мультимножеств.

[3] Здесь имеется в виду трассовая эквивалентность.

может быть переведена в эквивалентную WF-сеть. Тем не менее может быть определено соответствие между каузальной сетью и WF-сетью, описанное в [3]. Пусть $C = (A, a_i, a_i, D, I, O)$ - некоторая каузальная сеть, и $N_C = (P, T, F)$ - соответствующая workflow-сеть. Тогда:

- $P = \{p_a^I \mid a \in A\} \cup \{p_a^O \mid a \in A\} \cup \{p_{(a_1,a_2)}^D \mid (a_1, a_2) \in D\}$,
- $T^I = \{a_X^I \mid a \in A \wedge X \in I(a) \wedge X \neq \emptyset\}$,
- $T^O = \{a_X^O \mid a \in A \wedge X \in O(a) \wedge X \neq \emptyset\}$,
- $T = A \cup T^I \cup T^O$,
- $F = \{(p_a^I, a) \mid a \in A\} \cup \{(a, p_a^O) \mid a \in A\} \cup \{(a_X^I, p_a^I) \mid a_X^I \in T^I\} \cup \{(p_a^O, a_X^O) \mid a_X^O \in T^O\} \cup \{(p_{(a_1,a)}^D, a_X^I) \mid a_X^I \in T^I \wedge a_1 \in X\} \cup \{(a_X^O, p_{(a,a_2)}^D) \mid a_X^O \in T^O \wedge a_2 \in X\}$.

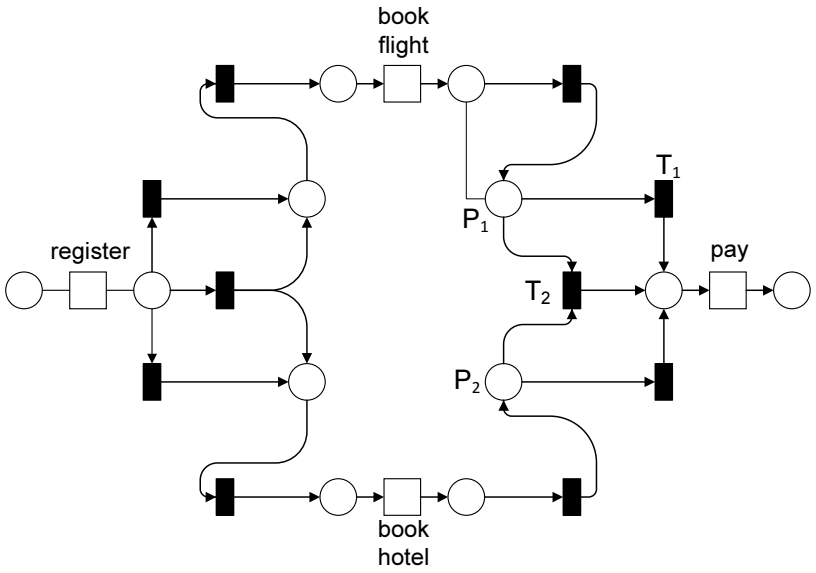

Рис. 1.2: Workflow-сеть, построенная из каузальной сети на рис. 1.1

1.3 Системы переходов

Для описания поведения каузальной сети удобно использовать системы переходов в силу простоты модели и наличия богатого математического

аппарата. Определим ([2]) *помеченную систему переходов (LTS)* как тройку $LTS = (P, Act, \rightarrow)$, где P - множество, называемое *доменом* LTS, Act - множество действий и $\rightarrow \subseteq P \times Act \times P$. $P \xrightarrow{a} Q \Rightarrow \exists(P, a, Q) \in \rightarrow$. Q называется a-производной от P.

Переход $P \Longrightarrow P'$ называется *слабым*, если для $n \geq 0$ существуют $P_1, ..., P_n$, такие, что $P' = P_n$ и $P \xrightarrow{\tau} P_1 ... \xrightarrow{\tau} P_n$, где τ — скрытое действие. Заметим, что если $\mu \in Act$, то $P \xRightarrow{\mu} P' \Leftrightarrow P \Longrightarrow P_1 \xrightarrow{\mu} P_2 \Longrightarrow P'$.

Рассмотрим две системы переходов: LTS_P и LTS_Q. Бинарное отношение $\mathcal{R}$ является *ветвящейся симуляцией* тогда и только тогда, когда для любого P', где $P \xrightarrow{\mu} P'$ либо

(a) $\mu = \tau$ и $P' \ \mathcal{R} \ Q$ или

(b) существуют Q', Q_1, Q_2, такие, что $Q \Longrightarrow Q_1 \xrightarrow{\mu} Q_2 \Longrightarrow Q'$, где $P \ \mathcal{R} \ Q_1$, $P' \ \mathcal{R} \ Q_2$ и $P' \ \mathcal{R} \ Q'$;

Если $LTS_P \ \mathcal{R} \ LTS_Q$, то LTS_Q симулирует все последовательности переходов из LTS_P.

1.4 BPMN

Нотация BPMN (Business Process Model and Notation, нотация и модель бизнес-процессов) является одним из самых популярных языков представления бизнес-процессов. BPMN имеет собственный стандарт, постоянно развиваемый и дополняемый. Стандарт имеет значительное число элементов, однако для целей данной работы будут использованы только некоторые из них.

Рассмотрим формальное определение BPMN-процесса. *BPMN-процесс* – это кортеж $BPMN_{proc} = (N, S, A, G_{XOR}, G_{AND}, e_{start}, e_{end})$, где N множество вершин, $S \subseteq N \times N$ – множество элементов потока управления. Множество вершин разделено на следующие не пересекающиеся подмножества: A - множество действий, G_{XOR}, G_{AND} - множества исключающих (XOR) и параллельных (AND) операторов (gates) и $\{e_{start}\}$ и $\{e_{end}\}$ – единичные множества, состоящие из начального и конечного события

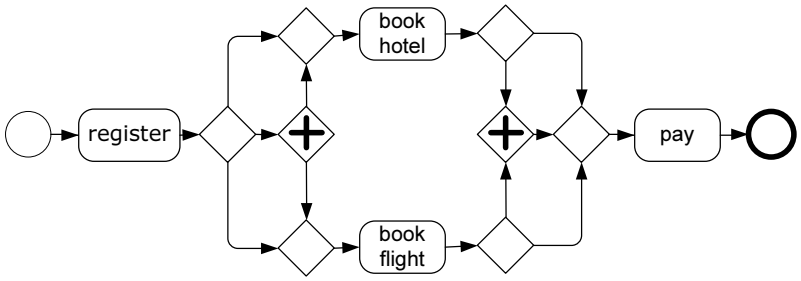

Рис. 1.3: Пример BPMN-модели процесса

соответственно. Дадим определение *согласованного (well-formed)* BPMN-процесса. Рассмотрим вершину $n \in N$, для которой *вход (preset)* ${}^\bullet n$ и *выход (postset)* $n^\bullet$ определяются как мультимножества над S, так что ${}^\bullet n(s) = 1$, если s является входным элементом потока управления для n, в противном случае ${}^\bullet n(s) = 0$, и $n^\bullet(s) = 1$ если s – выходной поток управления для n, иначе $n^\bullet(s) = 0$.

Согласованный BPMN-процесс
$BPMN_{proc} = (N, S, A, G_{XOR}, G_{AND}, e_{start}, e_{end})$ - это BPMN-процесс, который удовлетворяет следующие базовые аксиомы:

1. Начальное событие e_{start} не имеет входящих потоков управления;
2. Конечное действие e_{end} не имеет исходящих потоков управления;
3. Каждая вершина из N лежит на пути из e_{start} в e_{end}.

В отличие от сетей Петри, где фишки располагаются на позициях, *маркировка* BPMN-процесса является мультимножеством над множеством элементов потока управления, и обозначается как $m : S \to \mathbb{N}$. *Начальная маркировка* - это такая маркировка, что $m(s) = 1$, если s исходящий поток управления из e_{start}, $m(s) = 0$ иначе. Каждая вершина в BPMN-процессе может быть *активированной* (*enabled*). Рассмотрим правила срабатываний переходов в BPMN:

1. Действие $a \in A$ *активируется* в маркировке m тогда и только тогда, когда $\exists s \in S : {}^\bullet a(s) = 1$ и $m(s) \geq 1$. Другими словами, дей-

ствие a активируется в маркировке m если и только если существует хотя бы один входной поток управления, имеющий по крайней мере одну фишку. Выполняемое действие a может сработать, тем самым производя маркировку m', такую что $m'(s) = m(s) - 1$, и $m'(s') = m(s') + a^\bullet(s')$ для $s' \in S$, причем $s' \neq s$. Когда действие срабатывает, то поглощается по одной фишке на каждом входящем потоке управления, а на всех исходящих потоках управления появляется по одной новой фишке.

2. Исключающие операторы соединяют альтернативные пути: фишка на входящем потоке управления передается на один из исходящих потоков. Так же, как и действия, $g_{XOR} \in G_{XOR}$ активируются в маркировке m тогда и только тогда, когда $\exists s \in S : {}^\bullet g_{XOR}(s) = 1$, содержащий по крайней мере одну фишку в m: $m(s) \geq 1$. Шлюз производит фишку только на один из исходящих потоков: новая маркировка $m'(s) = m(s) + 1$ если $s \in S$ является одним из исходящих потоков управления, $\forall s' \in S$, таких что $s' \neq s$, $m'(s') = m(s')$.

3. Параллельный оператор $g_{AND} \in G_{AND}$ активируется в m тогда и только тогда, когда ${}^\bullet(g_{AND}) \subseteq m$, т.е. каждый входящий поток управления имеет по крайней мере одну фишку. Выполняемый оператор g_{AND} может сработать и произвести новую маркировку: m', такую, что $m' = m - {}^\bullet g_{AND} + g_{AND}^\bullet$, т.е. параллельный оператор поглощает фишку на каждом входящем потоке управления и производит фишку на каждом исходящем.

4. Стартовое событие никогда не срабатывает, так как оно не имеет входных элементов потока управления.

5. Конечное событие активируется, если в m $\exists s \in S : {}^\bullet e_{end}(s) = 1$; когда конечное действие срабатывает, то все фишки на входных элементах потока управления поглощаются, и производится новая маркировка $m' = m - {}^\bullet e_{end}$.

BPMN-процесс останавливает свое выполнение, если ни одна вершина не может сработать.

Важно отметить, что BPMN работает с *операционной семантикой*, в то время как C-net предоставляет *декларативное* описание поведения процесса. Таким образом, для сравнения поведения соответствующих BPMN-модели и каузальной сети необходим некоторый промежуточный язык. В данной работе в качестве такого языка используются системы переходов, так как и C-net, и BPMN-процессы можно привести к LTS.

2 Построение BPMN-модели процесса по каузальной сети

2.1 Ограничения преобразования

Рассмотрим WF-сеть на рис. 1.2, построенную по C-net. Заметим, что ${}^\bullet T_1 = \{P_1\}$ и ${}^\bullet T_2 = \{P_1, P_2\}$. По определению сети *свободного выбора*, сеть Петри является сетью свободного выбора тогда и только тогда, когда для любых двух переходов t_1 и t_2: ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset \Rightarrow {}^\bullet t_1 = {}^\bullet t_2$. Таким образом, рассматриваемая WF-сеть не является сетью свободного выбора.

С другой стороны BPMN-процессы относятся к классу сетей свободного выбора. Это можно легко показать, построив по произвольной BPMN-модели такую сеть Петри, что каждая вершина кроме начального и конечного событий трансформируется в переход, а на каждом элементе потока управления ставится позиция. Получается, что каузальные сети выразительнее чем BPMN в терминах потока управления.

Вследствие серьезных различий между операционной и декларативной семантикой, преобразование с полным сохранением поведения оказывается значительно затруднено. В работе *эквивалентным преобразованием* одной модели в другую считается такое, что множества всех корректных последовательностей обеих моделей совпадают. Такая оговорка обусловлена тем, что из-за разности в семантике в BPMN-процессе могут появляться некорректные последовательности действий.

2.2 Приведение C-net к LTS

Рассмотрим преобразование каузальной сети к LTS.

Определение 1 (Приведение C-net к LTS)**.** *Рассмотрим произвольную каузальную сеть* $C = (A, a_i, a_o, D, I, O)$*, которая имеет определенное пространство состояний* S*. Тогда помеченная система переходов, построенная по данной сети, определяется как* $LTS = (P, Act, \rightarrow)$*, где* P *— множество состояний каузальной сети, достижимых из начального состояния* ψ_0*,* Act *— множество действий каузальной сети, участвующих хотя бы в одной связи* $\beta = (a, as^I, as^O)$*, такой что* $\psi(\sigma), \psi(\sigma \oplus \beta) \in P$*. Переходы* $P \xrightarrow{\mu_i^I} P_1 \xrightarrow{\mu} P_2 \xrightarrow{\mu_j^O} P'$ *существуют в* LTS *тогда и только тогда, когда* $\exists \sigma, s, s', \beta = (\mu, M^I, M^O)$*, такие что* $\mu_i^I \in M^I$*,* $\mu_j^O \in M^O$*,* $\psi(\sigma) = s, \psi(\sigma \oplus \beta) = s'$ *и* $\sigma \oplus \beta$ *является частью корректной последовательности в* C*.*

Важно отметить, что если $M^I = \emptyset$ или $M^O = \emptyset$, то $P = P_1$ или $P_2 = P'$ соответственно.

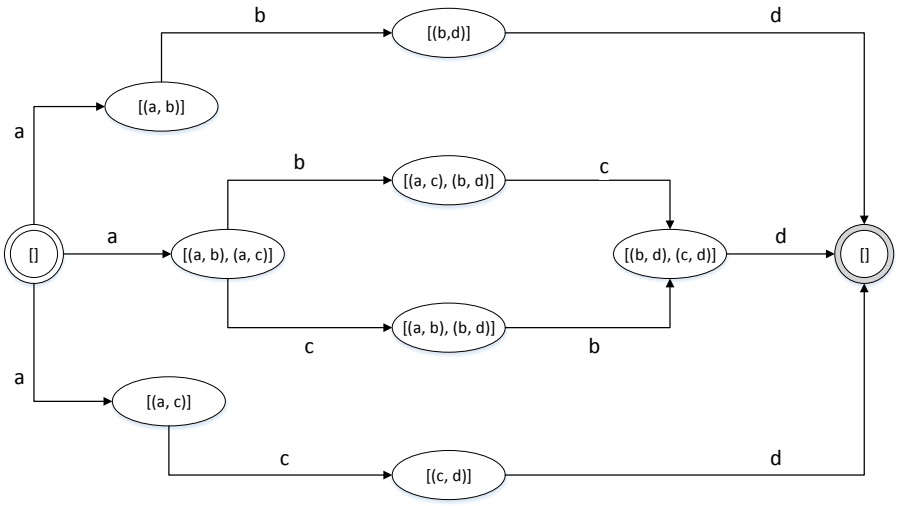

Рис. 2.1: Система переходов, построенная по каузальной сети на рис. 1.1

На рис. 2.1 представлена система переходов, моделирующая поведение каузальной сети на рис. 1.1. Домен системы состоит из конечного

пространства состояний исходной C-net. Несмотря на то, что начальное и конечное элементы домена LTS представляют одно и то же состояние каузальной сети, следует различать их, так как C-net не моделирует последовательности связи вида $\sigma = \sigma_1 \oplus ... \oplus \sigma_n$ $(n > 1)$, где σ_i — корректная последовательность связей.

Стоит отметить, что в случае неограниченной каузальной сети (*unbounded C-net*)[1] домен системы переходов может быть бесконечным множеством. Метод построения из определения 1 гарантирует, что LTS содержит все корректные последовательности исходной каузальной сети.

2.3 Соответствие каузальной сети и BPMN-модели

2.3.1 Построение BPMN-модели по каузальной сети

Алгоритм 1 (Построение BPMN-процесса по каузальной сети)**.** ***Вход:*** *Каузальная сеть* $C = (A_C, a_i, a_o, D, I, O)$.
Построим соответствующую BPMN-модель $BPMN = (N, SF, A_{BPMN}, G_{XOR}, G_{AND}, e_{start}, e_{end})$ *по следующим шагам:*

Шаг 1. Конвертация действий

(a) Для каждого $a_C \in A_C$ *построим действие* a_{BPMN} *и добавим его в* A_{BPMN}.

(b) Если $a_C = a_i$, *то добавим начальное событие* e_{start}, *соединенное с* a_{BPMN} *исходящим потоком управления.*

(c) Если $a_C = a_o$, *то добавим конечное событие* e_{end}, *соединенное входящим потоком управления с* a_{BPMN}.

Шаг 2: Конвертация входов и выходов.

Шаг 2.1: Конвертация выходов

[1]Каузальная сеть неограниченна, если она имеет бесконечное число состояний.

i. Для каждого действия $a_C \in A_C$ *установим соответствующее действие* a_{BPMN} *как* текущую вершину *(*$n_{CUR} := a_{BPMN}$*).*

ii. Рассмотрим выходы действия a_C*:* $O(a_C) = \{O_1, ..., O_k\}$*. Если* $|O(a_C)| > 1$*, добавим в BPMN-модель исключающий оператор* $g_{XOR} \in G_{XOR}$*, установим его как* текущую вершину *(*$n_{CUR} := g_{XOR}$*) и соединим входящим потоком управления с* a_{BPMN}*.*

iii. Для каждой связи $O_i, i \in \bar{1..k}$ *if* $|O_i| > 1$ *добавим параллельный оператор* $g^i_{AND} \in G_{AND}$*, соединим его входящим потоком управления с* текущей вершиной*, установим* g^i_{AND} *как* вершину связи *(*$n_i := g^i_{AND}$*), иначе установим* вершину связи *как* текущую вершину *(*$n_i := n_{CUR}$*).*

Шаг 2.2: Конвертация входов

i. Для каждого действия $a_C \in A_C$ *установим соответствующее действие* a_{BPMN} *как* текущую вершину *(*$n_{CUR} := a_{BPMN}$*).*

ii. Рассмотрим входы действия a_C*:* $I(a_C) = \{I_1, ..., I_k\}$*. Если* $|I(a_C)| > 1$*, добавим в BPMN-модель исключающий оператор* $g_{XOR} \in G_{XOR}$*, установим его как* текущую вершину *(*$n_{CUR} := g_{XOR}$*) и соединим исходящим потоком управления с* a_{BPMN}*.*

iii. Для каждой связи $I_i, i \in \bar{1..k}$ *if* $|I_i| > 1$ *добавим параллельный оператор* $g^i_{AND} \in G_{AND}$*, соединим его исходящим потоком управления с* текущей вершиной*, установим* g^i_{AND} *как* вершину связи *(*$n_i := g^i_{AND}$*), иначе установим* вершину связи *как* текущую вершину *(*$n_i := n_{CUR}$*).*

Шаг 3: Моделирование дуг.

Для каждой дуги $d = (a_C, a'_C)$ *добавим в BPMN-модель исключающий оператор* $g_d \in G_{XOR}$*, моделирующий дугу.*

Шаг 4: Связывание входов и выходов.

(a) *Для каждого действия* a_C *рассмотрим его входы* $I(a_C)$. *Для каждого входа* I_i *и каждой дуги* $d = (a'_C, a_C)$, *где* $a'_C \in I_i$, *поставим в соответствие* вершину связи n_i *и соединим ее в BPMN-модели исходящим потоком управления с вершиной* g_d.

(b) *Для каждого действия* a_C *рассмотрим его выходы* $O(a_C)$. *Для каждого выхода* O_i *и каждой дуги* $d = (a_C, a'_C)$, *где* $a'_C \in O_i$, *поставим в соответствие* вершину связи n_i *и соединим ее в BPMN-модели входящим потоком управления с вершиной* g_d.

Выход: $BPMN = (N, S, A_{BPMN}, G_{XOR}, G_{AND}, e_{start}, e_{end})$.

2.3.2 Пример построения

Каузальная сеть на рис. 1.1 соответствует BPMN-модели на рис. 2.2. Рассмотрим процесс построения более подробно. На первом шаге построения (на рис. 2.2 отмечено синим цветом) в BPMN-модель добавляются все действия из исходной каузальной сети и два события: начальное и конечное. События привязываются соответственно к начальному и конечному действию модели. В рассматриваемом примере такими действиями являются $register$ и pay.

На втором шаге создаются операторы для моделирования входов и выходов каждого действия в каузальной сети (помечены зеленым цветом). Если у действия несколько входов/выходов, то добавляется оператор исключающего «ИЛИ», связанный потоком управления с рассматриваемым действием. Таким образом, после срабатывания любого действия поток управления может пойти лишь по одному из его выходов, что соответствует поведению каузальной сети. Если какой-либо вход/выход действия a исходной C-net содержит несколько действий, то для этого выхода создается оператор «И», связанный выходным/входным потоком управления либо с a, либо с оператором «ИЛИ» (в случае, если a имеет несколько входов/выходов).

Третий шаг (помечен красным цветом) добавляет исключающие операторы для каждой дуги в каузальной сети. Данный шаг необходим для

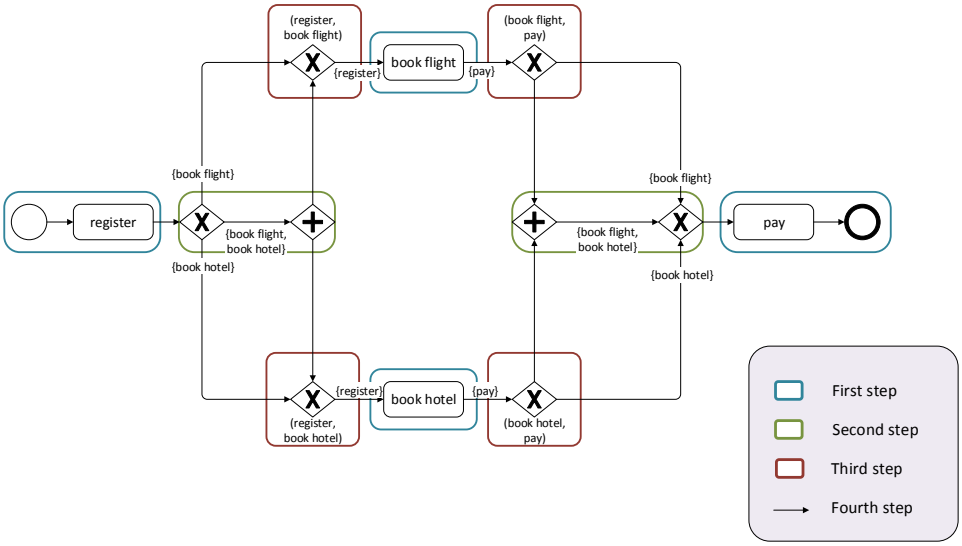

Рис. 2.2: BPMN-процесс, построенный по каузальной сети

синхронизации выходов одного действия и входов связанного действия в BPMN-модели.

Наконец, на четвертом шаге все операторы входов и выходов соединяются потоками управления с соответствующими операторами, моделирующими дуги. В результате получается BPMN-модель, симулирующая поведение исходной каузальной сети. Однако полученная модель в силу операционной семантики не избавлена от процессных аномалий.

2.3.3 Приведение BPMN-модели к LTS

Для доказательства симуляции BPMN-моделью поведения каузальной сети в работе используются системы переходов как промежуточный язык. Приведение C-net к LTS было рассмотрено в подразделе 2.2. Рассмотрим свойства системы переходов, построенной по BPMN-модели.

Определение 2 (Приведение BPMN-модели к LTS)**.** *Рассмотрим произвольную BPMN-модель* $BPMN = (N, S, A_{BPMN}, G_{XOR}, G_{AND}, e_{start}, e_{end})$*, имеющую определенное множество маркировок* $M = \{m(s) \mid s \in S\}$*. Тогда система переходов, построенная по BPMN-модели определяется как граф достижимости* $LTS = (P, A, \rightarrow)$*, где* $P = M$ *и* $A =$

$A_{BPMN} \cup \{\tau\}$, *причем переходы по* τ *соответствуют срабатываниям операторов* $g \in G_{XOR} \cup G_{AND}$.

В корректности построения системы переходов можно легко убедиться. Так как домен LTS состоит из маркировок BPMN-модели, и переходы между маркировками обеспечиваются срабатыванием действий и операторов, то в LTS такие переходы моделируются переходами по действиям и слабыми переходами (в случае срабатывания операторов).

Лемма 1. *Рассмотрим модель* $BPMN = (N, S, A_{BPMN}, G_{XOR}, G_{AND}, e_{start}, e_{end})$ *и построенная по определению 2 система переходов* $LTS = (P, A, \rightarrow)$. *Пусть в* $BPMN$ *существуют маркировка* m, *в которой активируется действие* a, *и маркировка* m', *достижимая из* m, *в которой активируются действия* $B \subset A$, *и не существует маркировка* m'', *такая, что* $m \rightarrow ...m'' \rightarrow ...m'$, *и в* m'' *активируются действия* $C \subset A$, *причем* $C \cap (B \cup \{a\}) = \emptyset$. *Тогда в* LTS *для любого* $b \in B$ *существуют переходы* $P \xrightarrow{a} P_1 \stackrel{b}{\Longrightarrow} P'$.

Доказательство. Возможность активации действия a в маркировке m означает, что имеется маркировка m_1, достижимая из m путем срабатывания a. Следовательно, LTS содержит переход $P \xrightarrow{a} P_1$.

Отсутствие маркировки m'' в переходах от m к m' говорит о том, что m' достижима из m только путем срабатывания некоторого числа (возможно, нулевого) операторов. Но срабатывания операторов в LTS отображаются как скрытые переходы, следовательно, в LTS переход в маркировку m_2 после срабатывания действия b может быть отображено как $P_1 \stackrel{b}{\Longrightarrow} P'$.

Итого, LTS содержит переходы $P \xrightarrow{a} P_1 \stackrel{b}{\Longrightarrow} P'$. □

2.3.4 Корректность построения BPMN-модели по каузальной сети

Критерием корректного построения BPMN-модели по каузальной сети является симуляция построенной моделью всех корректных последовательностей связей исходной C-net.

Лемма 2. *Пусть имеется каузальная сеть* $C = (A,\ a_i,\ a_o,\ D,\ I,\ O)$ *и построенная по ней BPMN-модель* $BPMN_{proc} = (N,\ S,\ A,\ G_{XOR},\ G_{AND},\ e_{start},\ e_{end})$. *Тогда для любой связи* $\beta = (a,\ I_i(a),\ O_j(a))$ *каузальной сети, где* β *принадлежит некоторой корректной последовательности связей* σ *справедливо следующее:*

1. *При параллельном срабатывании в BPMN-модели всех действий из* $I_i(a)$, *активируемых в маркировке* m, *существует такая маркировка* m', *достижимая из* m, *при которой возможно срабатывание действия* a.

2. *При срабатывании действия* a, *активируемого в маркировке* m, *существует такая маркировка* m', *достижимая из* m, *при которой возможно параллельное срабатывание всех действий из* $O_j(a)$.

Доказательство. Докажем первый пункт леммы. Рассмотрим некоторое действие $b \in I_i(a)$. Так как σ - корректная последовательность связей в C, то $\exists O_k(b) : a \in O_k(b)$. Если b имеет несколько выходов, то после срабатывания его в BPMN-модели в маркировке m, новая маркировка m_1 будет содержать фишку на входящем потоке управления в исключающий оператор $g^b_{XOR_O}$. Если $O_k(b)$ содержит несколько действий, то $m_2(s) = 1$, где ${}^\bullet g^b_{AND_{O_k}}(s) = 1$. Так как оператор $g^b_{AND_{O_k}}$ является разветвляющим, то $\exists s \in S, m_3$ такие, что ${}^\bullet g^{(b,a)}_{XOR}(s) = 1$ и $m_3(s) = 1$. Заметим, что при любой конфигурации выходов действия b маркировка m_3, активирующая исключающий оператор, моделирующий дугу (b, a) каузальной сети, достижима из маркировки m.

Подобный принцип можно применить и для входов действия a. Так как для любого действия $b \in I_i(a)$ существует маркировка m_l, активирующая оператор $g^{(b,a)}_{XOR}$, связанный одним или несколькими потоками управления с соответствующими входами действия a, то существует маркировка m' такая, что $m'(s) = 1$, где ${}^\bullet a(s) = 1$.

Второй пункт леммы доказывается аналогично первому. Заметим, что в случае, когда множество входов или выходов действия a является пустым (т.е. a - начальное или конечное действие), то в одном из пунктов $m = m'$. □

Таким образом, построенная BPMN модель симулирует все связи каузальной сети, входящие в *корректные* последовательности связей. Замечание про корректность является важным: алгоритм добавляет в каузальную сеть все входы и выходы, в том числе и те, которые не участвуют в корректных последовательностях связей; однако семантика каузальной сети игнорирует подобные входы и выходы, тогда как они с большой вероятностью ведут к ошибкам синхронизации в BPMN-модели.

Теорема 1 (Симуляция BPMN-моделью каузальной сети)**.** *Пусть имеется некоторая каузальная сеть* $C = (A, a_i, a_o, D, I, O)$ *и BPMN-модель* $BPMN = (N, S, A, G_{XOR}, G_{AND}, e_{start}, e_{end})$*, построенная по каузальной сети с помощью алгоритма 1. Тогда* $C \leq BPMN$*, где* $\leq$ *- отношение симуляции.*

Доказательство. Рассмотрим множество корректных последовательностей связей V_{CN} каузальной сети C. Если $V_{CN} = \emptyset$, то теорема верна.

Рассмотрим случай, когда V_{CN} имеет один или более элементов. Для доказательства симуляции будем использовать две системы переходов: $LTS_C = (P, A, \rightarrow)$, построенную по C, и $LTS_{BPMN} = (Q, A \cup \{\tau\}, \rightarrow)$, полученную из $BPMN$. Рассмотрим произвольную последовательность связей $\sigma \in V_{CN}$.

База индукции. Возьмем $\beta_0 = (a_i, \emptyset, as^O) \in \sigma$. По определению 1 LTS_C содержит переходы $P \xrightarrow{a_i} P' \xrightarrow{as^O_j} P''$ для любого действия из as^O. Из лемм 2 и 1 следует, что в LTS_{BPMN} существуют переходы $Q \xrightarrow{a_i} Q' \xRightarrow{as^O_j} Q''$ для любого действия из as^O. Таким образом, $P \mathcal{R} Q$, $P' \mathcal{R} Q'$ и $P'' \mathcal{R} Q''$, где $\mathcal{R}$ — отношение симуляции.

Шаг индукции. Пусть $\beta_i \in \sigma$ симулируется BPMN-моделью. Рассмотрим $\beta_{i+1} = (a, as^I, as^O)$. По определению 1 LTS_C содержит переходы $P \xrightarrow{as^I_i} P' \xrightarrow{a} P'' \xrightarrow{a^O_j} P'''$ для любых действий из as^I и as^O. В LTS_{BPMN} присутствуют переходы $Q \xRightarrow{as^I_i} Q' \xrightarrow{a} Q'' \xRightarrow{as^O_j} Q'''$ для любых действий из as^I и as^O. Таким образом, $P' \mathcal{R} Q'$, $P'' \mathcal{R} Q''$ и $P''' \mathcal{R} Q'''$.

Итого, рассмотрев все корректные последовательности связей подобным образом, получаем, что для любого P', такого что $P \xrightarrow{a} P'$, суще-

ствуют Q, Q_1, Q_2, Q', такие, что $Q \Longrightarrow Q_1 \xrightarrow{a} Q_2 \Longrightarrow Q'$. Таким образом, BPMN-модель симулирует поведение каузальной сети. □

Вследствие концептуальных отличий операционной семантики от декларативной обратная симуляция не может быть доказана. Если в исходной каузальной сети имеются связи, не участвующие ни в одной корректной последовательности связей, но присутствующие в итоговой BPMN-модели, то срабатывание действий этих связей с большой вероятностью приведет к ошибкам синхронизации.

Ошибки синхронизации могут возникать и в BPMN-модели, построенной по устойчивой каузальной сети. Например, если в модели на рис. 2.2 срабатывают действия $register$ и $book\ flight$, но не срабатывает действие $book\ hotel$, после срабатывания исключающего оператора $(book\ flight, pay)$ может стать активный параллельный оператор входа к действию pay.

3 Анализ результатов

3.1 Оценка сложности алгоритма преобразования

Оценим сложность алгоритма конвертации каузальной сети в BPMN-модель. Пусть в C-net имеется n действий, а общее количество входов и выходов на все действия — m.

На первом шаге алгоритма происходит прямая конвертация действий каузальной сети в действия BPMN-модели, поэтому сложность первого шага — $O(n)$. На втором шаге все входы и выходы моделируются не более двумя операторами и не более двумя потоками управления, поэтому справедливо считать сложность второго шага как $O(m)$.

Третий шаг алгоритма для каждой дуги создает исключающий оператор в BPMN-модели. Так как число дуг заведомо меньше числа связей (на одну дугу в каузальной сети приходится по крайней мере две связи: выход одного действия и вход другого), то трудоемкость этого шага заведомо меньше, чем $O(m)$.

Четвертый шаг для каждого входа, выхода и дуги, участвующей в данном входе или выходе добавляет поток управления. Таким образом, сложность четвертого шага равна $O(m)$.

Итого, сложность алгоритма может быть оценена как $O(n + m)$.

3.2 Плагин BPMNConversions для ProM

Алгоритм преобразования каузальной сети в BPMN-модель был реализован в процессно-ориентированной информационной системе ProM в рамках плагина (plug-in) BPMNConversions. Основная задача плагина — конвертация различных моделей (в том числе и каузальных сетей) в BPMN-модели.

В рамках плагина реализовано также исправление каузальной сети. Так как процесс может иметь несколько начальных или конечных действий, то результат работы «майнера» (программы, извлекающей модель процесса из лога) может представлять собой каузальную сеть с несколькими начальными или конечными действиями соответственно, что противоречит семантике C-net. В этом случае плагин непосредственно перед конвертацией добавляет единое начальное (common start) или конечное действие (common end) и связывает его со всеми начальными или конечными действиями соответственно.

Следует отдельно отметить случай, когда действие a в логе появляется в качестве как начального, так и конечного действия, но в разных последовательностях действий. Тогда в исправленной модели каузальной сети будет существовать корректная последовательность действий $\langle start, a, end \rangle$, которая может быть не отражена в логе.

Рассмотрим пример результатов работы алгоритма в рамках BPMNConversions. На рис. 3.1 представлена некоторая каузальная сеть, полученная из лога, которую необходимо конвертировать в BPMN-модель. Зеленым цветом помечены начальные действия, красным цветом — конечные. Заметим, что в визуализации модели отсутствуют черные точки, обозначающие связи. Эти связи можно посмотреть при наведении курсора мыши на действие.

На рис. 3.2 показан результат конвертации. Заметим, что в модели появились действия *start* и *end*, которые добавились в результате исправления каузальной сети.

Техническое задание на реализацию алгоритма конвертации приведено в Приложении А. Код алгоритма приведен в Приложении Б.

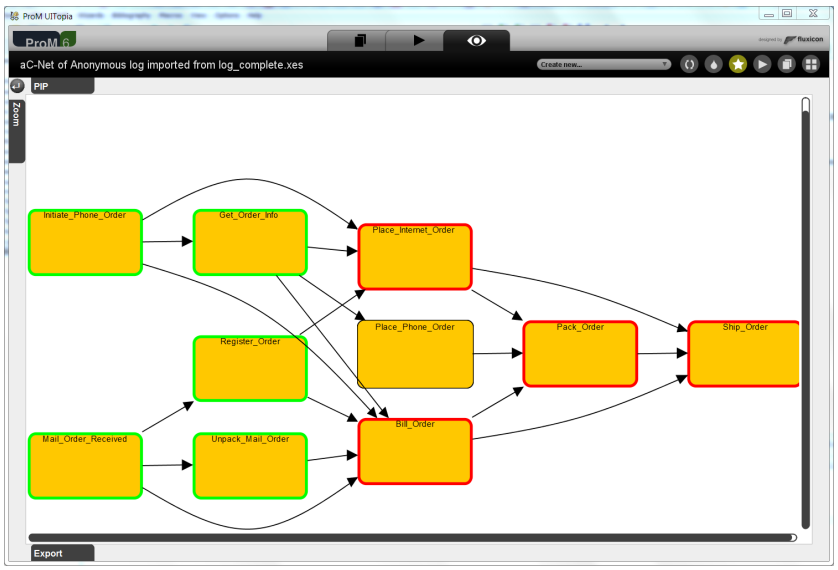

Рис. 3.1: Пример каузальной сети, извлеченной из лога процесса

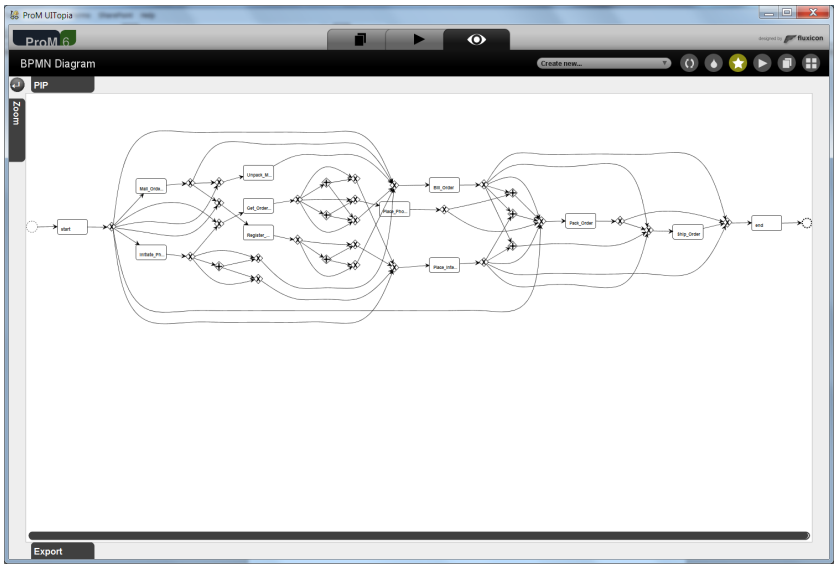

Рис. 3.2: Пример BPMN-модели, построенной по каузальной сети на рис.3.1

Заключение

В рамках работы была решена актуальная задача преобразования каузальной сети в BPMN-модель. Каузальные сети используются в рамках дисциплины Process Mining как результат извлечение модели процесса из лога этого процесса, так как семантика C-net не рассматривает последовательности действий с ошибками синхронизации. Однако представление модели процесса в виде каузальной сети может быть не очевидным для исследователей и аналитиков вследствие новизны нотации и сложности семантики.

С другой стороны, BPMN нотация знакома широкому кругу специалистов, поддерживается многими программными продуктами и является де-факто стандартом в области моделирования бизнес-процессов.

Для решения задачи был разработан алгоритм преобразования каузальной сети в BPMN-модель. Первоначальной целью работы ставилось построение BPMN-модели, эквивалентной заданной каузальной сети, однако в ходе исследования было выяснено, что в общем виде можно гарантировать только построение BPMN-модели, симулирующей поведение каузальной сети. Алгоритм был формализован, также была доказана его корректность на произвольной каузальной сети.

Алгоритм был реализован в рамках плагина BPMNConversions в системе ProM.

Литература

[1] OMG. Business Process Model and Notation (BPMN). Object Management Group, formal/2011-01-03, 2011.

[2] Davide Sangiorgi. *Introduction to Bisimulation and Coinduction.* Cambridge University Press.

[3] Wil Van Der Aalst, Arya Adriansyah, and Boudewijn Van Dongen. Causal nets: A modeling language tailored towards process discovery. In *Proceedings of the 22Nd International Conference on Concurrency Theory*, CONCUR'11, pages 28–42, Berlin, Heidelberg, 2011. Springer-Verlag.

[4] Wil M. P. van der Aalst. *Process Mining - Discovery, Conformance and Enhancement of Business Processes.* Springer, 2011.

[5] H.M.W. Verbeek, J.C.A.M. Buijs, B.F. van Dongen, and W.M.P. van der Aalst. ProM 6: The Process Mining Toolkit. In *Proc. of BPM Demonstration Track 2010*, volume 615 of *CEUR Workshop Proceedings*, pages 34–39, 2010.