УДК 519.681

А.С. ВАРОСЯН

АЛГОРИТМ ФОРМАЛЬНОЙ ВЕРИФИКАЦИИ ШАБЛОНОВ БИЗНЕС-ПРОЦЕССОВ

Ключевые слова: *бизнес-процесс, шаблон, формальная верификация.*

В настоящей статье рассматривается задача формальной проверки корректности шаблонов бизнес-процессов. Поскольку задача формальной верификации бизнес-процессов в общем случае неразрешима, возникает необходимость построения формально верифицируемых подклассов бизнес-процессов, представляющих практический интерес. Известно, что 70 % стоимости управления бизнес-процессами тратится на обслуживание существующих процессов и только 30 % — на разработку новых [1]. Известно также, что на перепроектирование существующих процессов тратится свыше 300 млн дол. в год [1, 2]. Это приводит к необходимости создания библиотек шаблонов бизнес-процессов или библиотек параметризованных описаний процессов, из которых можно получать конкретные процессы путем задания значений параметров [3]. В работе определяется формально верифицируемый подкласс шаблонов бизнес-процессов, который, как показано далее, содержит все шаблоны некоторых широко используемых библиотек шаблонов [3, 4].

Рассматриваемый подкласс шаблонов определяется заданием базовых элементов и операций над ними, с помощью которых можно построить произвольный шаблон подкласса. Показано, что задача формальной верификации шаблонов сводится к задаче формальной верификации базовых элементов. Построены алгоритмы полиномиальной сложности для верификации элементов шаблонов, на основе которых построен общий алгоритм верификации, также имеющий полиномиальную сложность.

В качестве базовой модели выбрана модель, описанная в IBM MQSeries Workflow [7]. Эта модель расширена с помощью конструкций, необходимых для рассмотрения задачи формальной верификации, которая предполагает задание пред- и постусловий процесса, т.е. условий начала и завершения процесса. Процесс называется (частично) корректным, если всякий раз при удовлетворении предусловия процесса и его успешного завершения удовлетворяется постусловие процесса. Процесс называется (полностью) корректным, если всякий раз при удовлетворении предусловию процесс завершается и удовлетворяется постусловие процесса.

В настоящей статье рассматривается проблема частичной корректности (далее — корректности) для шаблонов бизнес-процессов.

1. ФОРМАЛЬНАЯ МОДЕЛЬ БИЗНЕС-ПРОЦЕССОВ

Ниже определяется формальная модель бизнес-процессов на базе IBM MQSeries Workflow [7].

1.1. Модель процесса. Основными элементами процесса являются операторы действий (activities) (далее — операторы). Операторы имеют контейнеры входа и выхода, каждый состоит из конечного набора ячеек. Выполнение оператора преобразует состояние контейнера входа в состояние контейнера выхода. На множестве ячеек контейнеров определена сеть, по которой осуществляется пере-

дача данных. Топология сети задается изначально и не меняется в ходе функционирования процесса. Операторы связываются между собой коннекторами, каждый наделяется условием перехода. Кроме того, сходящиеся в общей точке коннекторы наделяются условием соединения, определяющим возможность прохождения потока управления через эту точку.

Существующие формальные модели процессов ациклические [7], т.е. управляющий граф процесса не содержит циклов. В то же время не практике используются бизнес-процессы, у которых управляющий граф содержит циклы. В работе построено расширение формальной модели [7] для представления также циклических процессов. Для циклического процесса определяются внутренние и внешние коннекторы. Внутренние коннекторы указывают на входную вершину цикла и принадлежат ему. Все остальные коннекторы внешние. Для внутренних коннекторов, связывающих операторы цикла с его входной вершиной, определено внутреннее условие соединения. Входная вершина цикла может быть активизирована как извне, так и при истинном значении условия внутреннего соединения.

Ниже определяются ациклические и циклические бизнес-процессы на основе W-процесса [5]. Используются следующие обозначения:

— для кортежа $x =< x_1, \ldots, x_n >$ и последовательности индексов $1 \le i_1, \ldots, i_k \le n$ $\pi_{i1, \ldots, ik}(x)$ обозначает кортеж $< x_{i1}, \ldots, x_{ik} >$.

— для множества X $\wp(X)$ обозначает множество всех подмножеств X.

Определение. W-процесс определяется как кортеж $P =< T, V, N, C, E, \Phi, i, o, \Psi, \Delta >$, в котором приняты следующие обозначения.

1. T — конечное множество типов.

Обозначение $\mathrm{DOM}(t)$ будем использовать для множества значений типа $t \in T$. Без ограничения общности для переменной v типа t отождествим $\mathrm{DOM}(v)$ с $\mathrm{DOM}(t)$. Обозначим V множество всех переменных.

2. N — конечное множество операторов, наделенных приоритетами.

3. C — конечное множество условий перехода.

4. $E \subseteq N \times N \times C$ — множество коннекторов. Если $e =< A, A', c >\in E$, то будем говорить, что коннектор e с условием перехода c соединяет оператор A с оператором A'. Предполагаем, что коннектор однозначно определяется соединяемыми операторами: $\forall e, e' \in E: \pi_{1,2}(e) = \pi_{1,2}(e') \Rightarrow \pi_3(e) = \pi_3(e')$.

5. $\Phi: N \to F$ — отображение, сопоставляющее оператору $A \in N$ условие соединения $\Phi(A)$. Считаем, что $\Phi(A) \equiv 1$, если A не имеет входных коннекторов.

6. $i: N \cup C \to V$ — отображение, сопоставляющее оператору или условию перехода контейнер входа. Последний представляет собой кортеж различных переменных, каждая из которых принадлежит некоторому типу из множества **T**.

7. $o: N \to V$ — отображение, сопоставляющее оператору контейнер выхода. Последний представляет собой кортеж различных переменных, каждая из которых принадлежит некоторому типу из множества **T**.

8. $\Psi: N \cup C \to E$ — отображение, сопоставляющее оператору или условию перехода семантику выполнения. При этом семантика выполнения оператора $A \in N$ определяется как отображение $\Psi(A)$: $\times_{v \in i(A)} \mathrm{DOM}(v) \to \times_{v \in o(A)} \mathrm{DOM}(v)$, а семантика выполнения условия $p \in C$ — как отображение $\Psi(p): \times_{v \in i(p)} \mathrm{DOM}(v) \to \{0,1\}$.

9. $\Delta: N \times (N \cup C) \to \cup_{A \in N, B \in N \cup C} \wp(o(A) \times i(B))$ — отображение, сопоставляющее каждой паре вида <оператор, оператор> или <оператор, условие перехода> бинарное отношение на множестве переменных контейнера выхода первого компонента и контейнера входа второго. Требуется выполнение следующих ограничений:

- $[A \in N, B \in N \cup C] \Rightarrow \Delta(A,B) \subseteq o(A) \times i(B)$,
- $[A \in N, B \in N \cup C, < v1, v' > \in \Delta(A,B)$,
- $< v2, v' > \in \Delta(A,B)] \Rightarrow v1 = v2$,
- $< v, v' > \in \Delta(X,Y) \Rightarrow \mathrm{DOM}(v) = \mathrm{DOM}(v')$.

Отметим, что определение W-процесса отличается от модели IBM [7] тем, что E не является ациклическим.

Определение ациклического процесса в [7] отличается от приведенного дополнительным ограничением: в п. 4 E — ациклический граф.

Графы процесса $G(N,E)$ и $G(N,\Delta)$ называют графом управления и графом передачи данных соответственно. $A^{\rightarrow} = \{e \in E | \pi_1(e) = A\}$ — множество операторов, следующих за оператором $A \in N$ в графе управления, и $A^{\leftarrow} = \{e \in E | \pi_2(e) = A\}$ — множество операторов, предшествующих оператору A в графе управления.

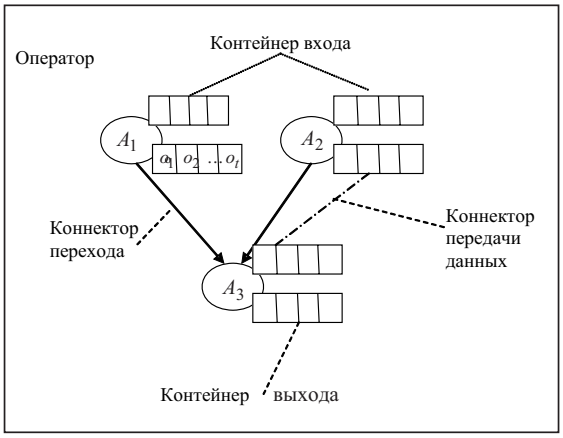

Рис. 1

Обозначим $N(P)$ множество операторов, содержащихся в P, $\mathrm{In}(P)$ и $\mathrm{Out}(P)$ — множество стартовых и конечных операторов процесса P соответственно.

Графическое представление бизнес-процессов приведено на рис. 1.

Пусть Ω — множество всех циклов, содержащихся в процессе P, и N_C — множество всех входных вершин циклов. Допустим, что $\forall A \in N$ множество $A_{\Delta}^{\rightarrow}$ определяет множество всех операторов, связанных с A с помощью коннекторов данных — $A_{\Delta}^{\rightarrow} = \{D \in N | \Delta(A,D) \neq \varnothing\}$. Обозначим K_{σ} множество всех операторов, содержащихся в цикле $\sigma \in \Omega$.

Определение (операторы пересечения циклов). Допустим, $\sigma_i \in \Omega$, $1 < i \le k \le |N_C|$, являются циклами процесса P. Операторы пересечения циклов — это множество операторов, общих для всех циклов: $\sigma_1 \cap \sigma_2 \cap \ldots \cap \sigma_k \equiv$ $\equiv K_{\sigma 1} \cap K_{\sigma 2} \cap \ldots \cap K_{\sigma k}$.

Далее определяется локализованный цикл, впервые введен Аллен при рассмотрении задачи оптимизации программ [8].

Определение (локализованный цикл). Цикл $\sigma \in \Omega$ с единственной входной вершиной a_1 называется локализованным циклом, если единственной точкой пересечения с другими циклами Ω является входная вершина цикла: $\forall \lambda \in \Omega,\ \lambda \neq \sigma$, $\lambda \cap \sigma \subseteq \{a_1\}$.

Определение (ветвящийся оператор). Оператор цикла называется ветвящимся оператором, если имеет хотя бы один выходной коннектор управления, соединенный с оператором, не принадлежащим циклу: $b \in N$ является ветвящимся оператором $\leftrightarrow \exists\ \sigma \in \Omega, e \in E, b' \in N \setminus K_{\sigma}(b \in K_{\sigma}, \pi_1(e) = b, \pi_2(e) = b')$.

Определение (согласованные операторы). Операторы A и A' назовем согласованными операторами, если они связаны коннекторами передачи данных только тогда, когда связаны также коннектором управления: $\forall A, A' \in N$, $\Delta(A,A') \neq \varnothing \Rightarrow A' \in A^{\rightarrow}$.

Обозначим $C^{C\leftarrow}(A)$ множество всех условий переходов, принадлежащих коннекторам, содержащимся в циклах с входной вершиной $A \in N_C$ и указывающим на оператор A. Обозначим $C^{*\leftarrow}(A)$ множество всех условий переходов, принадлежащих коннекторам, не содержащимся в циклах с входной вершиной $A \in N_C$ и указывающим на оператор A.

Ниже определяются сводимые ациклические и циклические процессы. Основой для их определения являются понятия структурного программирования [9].

1.2. Сводимость. Определение (сводимый ациклический процесс). Каждый W-процесс, имеющий ациклический граф управления, назовем сводимым ациклическим процессом, или A-процессом, если:

1) $\forall A \in N; |A^{\rightarrow}| > 1 \Rightarrow |A^{\leftarrow}| = 1;$

2) $\forall A \in N; |A^{\leftarrow}| > 1 \Rightarrow \forall A' \in A^{\leftarrow} (|A'^{\rightarrow}| = 1 \& \forall A'' \in A'^{\leftarrow}; |A''^{\rightarrow}| = 1);$

3) $\forall A \in N$ и $A' \in N$ — согласованные операторы.

Определение (сводимый циклический процесс). Каждый W-процесс назовем сводимым циклическим процессом, или C-процессом. При этом должны выполняться следующие условия.

1. Все циклы графа управления являются локализованными циклами.

2. $\Phi: N_C \to F$ отображение сопоставляет оператору $A \in N_C$ внешнее условие соединения $\Phi(A)$. Последнее определяется как выражение булевой алгебры, зависящее от множества переменных $C^{*\leftarrow}(A)$.

3. $\Phi_C: N_C \to F$ отображение сопоставляет оператору $A \in N_C$ внутреннее условие соединения $\Phi_C(A)$. Последнее определяется как выражение булевой алгебры, зависящее от множества переменных $C^{C\leftarrow}(A)$. Предположим, что $\forall A \notin N_C, \Phi_C(A) = 0$.

4. $\forall A \in N_C$ может быть активизирован $\Leftrightarrow (\Phi(A)(i(p_1), \ldots, i(p_{nA})) = 1) \vee$

$\vee (\Phi_C(A)(i(q_1), \ldots, i(q_{mA})) = 1)$, $C^{*\leftarrow}(A) = \{p_1, \ldots, p_{nA}\}$, $C^{C\leftarrow}(A) = \{q_1, \ldots, q_{mA}\}$.

5. $\forall A \in N$ и $A' \in N$ являются согласованными операторами.

6. $\forall A \in N; |A^{\rightarrow}| > 1 \Rightarrow |A^{\leftarrow}| = 1$.

7. $A \in N; |A^{\leftarrow}| > 1 \Rightarrow \forall A' \in A^{\leftarrow} (|A'^{\rightarrow}| = 1 \& \forall A'' \in A'^{\leftarrow}; |A''^{\rightarrow}| = 1)$.

Далее рассматриваются только сводимые бизнес-процессы.

1.3. Формальная верификация бизнес-процессов. Введены также новые понятия, необходимых для формальной верификации.

Определение (корректность бизнес-процесса). Пусть P — бизнес-процесс, α — предикат, зависящий от некоторых входных переменных операторов, а β — предикат, зависящий от некоторых выходных переменных операторов. Назовем α и β пред- и постусловием соответственно. В этом случае бизнес-процесс P корректен относительно α и β тогда и только тогда, когда $[\alpha(P) = 1 \Rightarrow \beta(MP) = 1]$, где MP — состояние выходных переменных операторов после выполнения процесса P.

Пусть заданы предусловие и постусловие процесса P.

Определение (существенная выходная переменная). Выходная переменная процесса P называется существенной, если выступает переменной выходного контейнера некоторого оператора $A \in N$, от которой зависит постусловие.

Обозначим $M(P)$ множество существенных выходных переменных бизнес-процесса P.

Добавим к каждому ветвящемуся оператору новую переменную r, называемую регистром состояния и принимающую одно из значений $\{0, 1, 2, \ldots\}$.

1. 0 означает, что ветвящийся оператор пока не выполнен.

2. $n;(n>=1)$ означает, что ветвящийся оператор выполнен n раз.

Пусть P — бизнес-процесс и $X \in N \cup C$. Тогда назовем состоянием бизнес-процесса кортеж, состоящий из переменных входных контейнеров операторов, условий перехода и существенных выходных переменных. Обозначим $B = \times_{Y \in N \cup C \cup M} \text{States}(Y)$ для множества всех состояний процесса, где $\text{States}(X) = \times_{v} \in i(X)$ и $\text{States}(M) = \times_{v \in M} \text{DOM}(v)$, $b(X)$ — состояние $X \in N \cup C \cup M$ для $b \in B$.

Пусть $b \in B$, $A \in N$. Обозначим $bA = b'$ новое состояние процесса, полученное после выполнения оператора A при состоянии процесса b. Процесс b' определяется следующим образом:

$$(b'(X))_{v'} = \begin{cases} (b(X))_{v'}, & \text{если } \forall v \in o(A)[< v, v' > \notin \Delta(A, X)], \\ (\Psi(A)(b(A)))_{v}, & \text{если} < v, v' > \in \Delta(A, X), \end{cases}$$

для всех $X \in N \cup C \cup M$.

Допустим, $b_0 \in B$ — начальное состояние процесса до его выполнения. Пусть $b = b_0 P$ — состояние процесса после его выполнения [7], а MP — множество значений существенных выходных переменных состояния b после его выполнения.

Далее покажем, что для подкласса сводимых бизнес-процессов, называемых классом шаблонов бизнес-процессов [3–6], существует базис элементарных процессов — примитивов, и соответствующие операции над этим базисом, с помощью которых можно построить произвольный шаблон бизнес-процесса.

2. ШАБЛОНЫ БИЗНЕС-ПРОЦЕССА

Введем ограничения на типы и реализации, которые соответствуют шаблонам бизнес-процессов:

1) $T(P) = \{TB, TG\}$, где $TB = \{\text{Unknown, True, False}\}$, $TG = \{\text{Unknown, Known}\}$;

2) $\forall A \in N(\text{DOM}(o(A)) = TG, \Psi(A) \equiv$ 'Known').

Такие бизнес-процессы называют унифицированными бизнес-процессами (далее — бизнес-процесс). Ниже рассматриваются три типа примитовов, используемых в построении процессов.

2.1. Примитивы. Определение (типы примитивов). Простой примитив — это простой оператор. Например, $\forall\, a \in N$ — простой примитив, $\text{In}(u) = \text{Out}(u) = N(u) = \{a\}$.

• Примитив синхронизации u — узел соединения связывающих коннекторов,

$$a_1, a_2, \ldots, a_m \in N,\ e_1 =< a_1, a_m, c_1 > \in E, \ldots, e_{m-1} =< a_{m-1}, a_m, c_{m-1} > \in E,$$

$$c_1, \ldots, c_{m-1} \in C, c_1 = c_2 = \ldots = c_{m-1} \equiv 1,$$

$$N(u) = \{a_1, a_2, \ldots, a_m\}, \text{In}(u) = \{a_1, \ldots, a_{m-1}\}, \text{Out}(u) = \{a_m\}.$$

• Примитив разделения u — ветвящийся узел выходных коннекторов,

$$a_1, a_2, \ldots, a_m \in N, e_1 =< a_1, a_2, c_1 > \in E, e_{m-1} =$$

$$=< a_1, a_m, c_{m-1} > \in E, c_1, \ldots, c_{m-1} \in C,$$

$$N(u) = \{a_1, a_2, \ldots, a_m\},\ \text{In}(u) = \{a_1\},\ \text{Out}(u) = \{a_2, \ldots, a_m\}.$$

Схемы примитивов представлены на рис. 2.

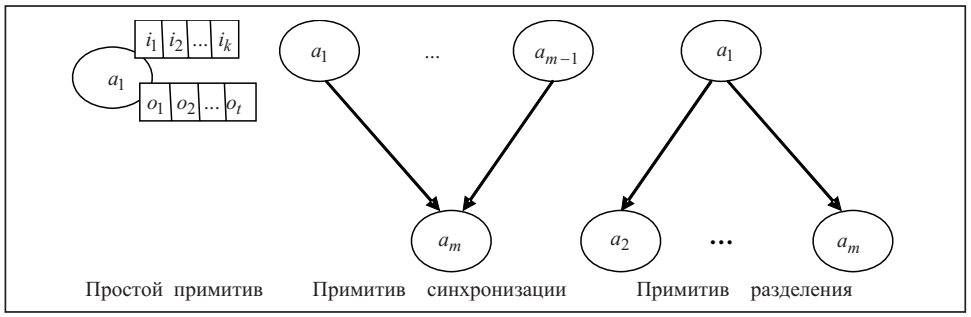

Рис. 2

Обозначим множества операторов и управляющих коннекторов, содержащихся в u, соответственно N_u и E_u. Пусть $U(P)$ — множество всех примитивов, содержавшихся в P бизнес-процессе. Обозначим также тип примитива u через $\text{type}(u) \in \{\text{простой, синхронизация, разделение}\}$. Предположим, что BU — множество всех возможных примитивов. Простые примитивы, а также примитивы разделения назовем примитивными ациклическими шаблонами.

Ниже определены две операции: конкатенации и слияния, использующиеся для рекурсивного построения новых ациклических шаблонов BPT с привлечением построенных. В качестве начальных шаблонов рассматриваются примитивные ациклические шаблоны.

2.2. Алгебра шаблонов бизнес-процессов. Определение (операция конкатенации). Конкатенация $P' = \text{Concat}(P_1, P_2, k, c)$ двух ациклических шаблонов P_1 и P_2 таких, что $\text{In}(P_1) = \{ip_1\}, \text{Out}(P_1) = \{op_1, \ldots, op_{k1}\}$, $\text{In}(P_2) = ip'_1$, $\text{Out}(P) = \{op'_1, \ldots, op'_{k2}\}$, $k1, k2 > 0$, — это ациклический шаблон такой, что:

$P' = Cn(P_1, P_2, k, c) = P_1 +_{c,k} P_2$ является бизнес-процессом с графом $G(N, E)$, где

$$N = N(P_1) \cup N(P_2),\ C = C(P_1) \cup C(P_2) \cup c,$$

$$E = E(P_1) \cup E(P_2) \cup < op_k, ip'_1, c >,\ 1 \le k \le k1,$$

$$\Delta = \Delta(P_1) \cup \Delta(P_2) \cup \Delta(op_k, ip'_1),\ M = M(P_1) \cup M(P_2).$$

Определение (операция слияния). Слияние $P' = \text{Merge}(u, P, \bar{k}, \bar{c})$ примитива синхронизации u с ациклическим шаблоном P таким, что $\text{In}(P) = \{ip_1\}$, $\text{Out}(P) = \{op_1, \ldots, op_{ko}\}$, $ko > 0$, $\text{In}(u) = \{iu_1, \ldots, iu_m\}$, $\text{Out}(u) = \{ou_1\}$, $1 \le m \le ko$, — это ациклический шаблон такой, что:

$P' = \text{Merge}(u, P, \bar{k}, \bar{c}) = P \otimes_{\bar{k}, \bar{c}}$ — бизнес-процесс с графом $G(N, E)$, где
$\bar{c} = < c_1, \ldots, c_m >, \bar{k} = < k_1, \ldots, k_m >$, где $k_i \in \text{Out}(P)$, $k_i \ne k_j$, $i \ne j$, $1 \le i, j \le m$,
$N = N(u) \cup N(P), C = C(u) \cup C(P) \cup c, M = M(u) \cup M(P)$,
$E = E(u) \cup E(P) \cup$ $\Delta = \Delta(u) \cup \Delta(P) \cup$
$\cup < op_{k_1}, iu_1, c_1 > \cup \ldots \cup < op_{k_m}, iu_m, c_m >$ $\cup \Delta(op_{k_1}, iu_1) \cup \ldots \cup \Delta(op_{k_m}, iu_m)$.

Здесь c — условие операции для конкатенации, в то время как $\bar{c}$ — условие операции для слияния. Обозначим с помощью $c(u)$ условие операции конкатенации или слияния примитива u с ациклическим шаблоном P. Будем считать, что $c(u) \equiv 1$, если примитив не комбинирован с шаблоном.

Определение (регулярный ациклический шаблон). Каждый ациклический шаблон, построенный из множества примитивов $BU' \subseteq BU$ с использованием операций конкатенации и слияния, называют регулярным ациклическим шаблоном.

Лемма 1. Для каждого регулярного ациклического шаблона существует

подмножество примитивов $BU' \subseteq BU$ такое, что шаблон можно представить в виде последовательности операций конкатенации и слияния над BU'.

Такое представление назовем разложением шаблонов на примитивы. Покажем, что каждому шаблону однозначно соответствует определенная последовательность операций над примитивами.

Алгоритм *D*. Разложение шаблона на примитивы

Вход: Ациклический шаблон P с графом $G(N,E)$.

Выход: Последовательность операций конкатенации и слияния над примитивами

Метод:

1. Для всех $A \in N$ выполнить
   - Если $|A^{\rightarrow}| = 0$, то добавить A к Front – y
2. bUnit $\leftarrow 0, N_p \leftarrow N, E_p \leftarrow E$
3. Пока !bUnit выполнить
   - 3.1. Выбрать A из Front-a
   - 3.2. $N_u \leftarrow \{A\}, E_u \leftarrow \varnothing$, $N_p \leftarrow N_p / \{A\}$, typeOP $\leftarrow ''$, $k \leftarrow \varnothing, c \leftarrow \varnothing$
   - 3.3. Если $|A^{\leftarrow}| > 1$, то:
     - a) bUnit $\leftarrow 1$, typeOP $\leftarrow$ Merge
     - b) Для всех $e \in A^{\leftarrow}, e = (A', A, p)$, выполнить
       - $N_u \leftarrow N_u \cup \{A'\}, N_p \leftarrow N_p / \{A'\}, E_u \leftarrow E_u \cup \{e\}, E_p \leftarrow E_p / \{e\}$
       - добавить A' к **k**, добавить **p** к **c**
   - 3.4. Или если $|A^{\leftarrow}| = 1, A^{\leftarrow} = \{< A', A, p >\}$, то:
     - a) typeOP $\leftarrow$ Concat, $k \leftarrow A', c \leftarrow p$
     - b) если $|A'^{\leftarrow}| = 1$, то bUnit $\leftarrow 1$
     - c) иначе если $A'^{\leftarrow} \subseteq$ Front, то $N_u \leftarrow \{A'\}$, bUnit $\leftarrow 1$
     - d) для всех $e \in A'^{\leftarrow}, e = (A', A'', p)$ выполнить
       - $N_u \leftarrow N_u \cup \{A''\}, N_p \leftarrow N_p / \{A''\}, E_u \leftarrow E_u \cup \{e\}, E_p \leftarrow E_p / \{e\}$
     - e) Иначе удалить A из Front-a
   - 3.5. Возвратить typeOP$(< N_u, E_u >, D(N_p, E_p), k, c)$.

Здесь и далее вместо выражения «регулярный ациклический шаблон» используется термин «ациклический шаблон». Следующая операция — операция цикла, используется для построения циклических шаблонов.

Обозначим path(a, a') множество операторов, принадлежащих пути между $a \in N$ и $a' \in N$.

Определение (операция цикла). Операция цикла Cycle(P, u, u', a, a', p), примененная к шаблону P, — комбинация его двух примитивов u и u' посредством соединения $a \in N_u, a' \in N'_u$, если существует последовательность примитивов $u, u_1, \ldots, u_k, u'$, связанных операцией конкатенации и слияния так, что:

- не существует обратного пути между $a_1 \in$ path(a, a') и $a_2 \in$ path$(a, a') \Rightarrow$ $\Rightarrow$!$\exists$path(a_2, a_1);
- type$(u') \neq$ {synchronization}, type$(u_i) \neq$ {synchronization}, $1 \le i \le k$;
- если type$(u'') =$ {split}, $u'' \in \{u, u_1, \ldots, u_k, u'\}$, то примитив имеет только два взаимно исключающих условия переходов.

Определение (регулярный циклический шаблон). Циклический шаблон,

построенный на ациклическом шаблоне с использованием операций цикла, называется регулярным циклическим шаблоном.

Лемма 2. Каждый регулярный циклический шаблон разлагается на последовательность операций цикла, примененных к ациклическому шаблону.

Алгоритм разложения основан на обнаружении входных вершин циклов и на удалении внутренных коннекторов [8]. Здесь и далее вместо «регулярного циклического шаблона» использован термин «циклический шаблон». Ниже представлены определения ациклических и циклических шаблонов бизнес-процессов.

Определение (множество ациклических шаблонов). $AT =< BU, \text{Concat}, \text{Merge} >$ — множество регулярных ациклических шаблнов, каждый из которых построен на примитивах BU с использованием нулевого или большего количества операций Concat и Merge.

Определение (множество циклических шаблонов). $CT =< AT, \text{Cycle} >$ — множество регулярных циклических шаблонов, каждый из которых построен на ациклических шаблонах AT с использованием нулевого или большего количества операций Cycle.

Ниже представлены взаимосвязи между шаблонами и A- и C-процессами.

Лемма 3. Каждый ациклический шаблон $t \in AT$ является A-процессом.

Лемма 4. Каждый циклический шаблон $t \in CT$ является C-процессом.

Для формальной верификации ациклических шаблонов может быть предложен алгоритм верификации A-процессов [10].

Определение (класс шаблонов бизнес-процессов — класс BPT). Классом шаблонов бизнес-процессов является множество всех возможных ациклических и циклических шаблонов.

3. ЗАДАЧА ФОРМАЛЬНОЙ ВЕРИФИКАЦИИ ШАБЛОНОВ БИЗНЕС-ПРОЦЕССОВ

Определение (предусловие инициализации примитива). Назовем $\alpha(u)$ предусловием инициализации примитива $u \in BU$, если оно определено для проверки наличия начальных значений всех переменных входа, значения которых определяются вне процесса, и используется в постусловии или предусловии процесса:

$$\alpha(u) = \bigwedge_{a \in \text{In}(u)} \bigwedge_{i'_a \in i(a),\ \exists x(<x, i'> \in \Delta(\text{OUT}, a))\}} i'_a \neq \perp .$$

Определение (постусловие выполнения примитива). Назовем $\beta(u)$ постусловием выполнения примитива $u \in BU$, если оно определено для проверки логики последовательности выполнения операторов примитивов:

$$\beta(u) = \begin{cases} b(a), \text{ где } \text{type}(u) = \text{'простая'}, \text{Act}(u) = \{a\}, \\ b(a_1) \wedge b(a_2) \wedge b(a_3), \text{ где} \\ \quad \text{type}(u) = \text{'синхронизация'}, \text{Act}(u) = \{a_1, a_2, a_3\}, \\ b(a_1) \wedge pb(a_2) \vee pb(a_3) \vee \neg(pb(a_3) \wedge pb(a_3))), \text{ где} \\ \quad \text{type}(u) = \text{'разделение'}, \text{Act}(u) = \{a_1, a_2, a_3\}. \end{cases}$$

Новые выходные переменные ($b(a_i)$ и $pb(a_i)$) добавлены для определения постусловия, $b(a_i)$ «верно» после выполнения оператора a_i, $pb(a_i)$ сформулирован для проверки выполнения оператора a_i, если удовлетворено условие его перехода. Это означает, что $pb(a_i) = \text{TRUE}$, если a_i оператор выполнен после удовлетворения поступающего условия перехода, в то время как $pb(a_i) = \text{FALSE}$, если a_i оператор не может быть выполнен, если поступающее условие перехода

«ложно».

Определение (предусловие инициализации ВРТ). Допустим, P — шаблон бизнес-процесса и α — предусловие, определенное для P с помощью логических комбинаций предусловий инициализации примитивов процесса P, связанных между собой операциями конкатенации и слияния. В этом случае α назовем предусловием инициализации процесса P:

$$\alpha = \bigwedge_{u \in U_1(P)} \alpha(u) \& c(u), \quad U_1(p) \subseteq U(P).$$

Определение (постусловие выполнения ВРТ). Допустим, P — шаблон бизнес-процесса и β — постусловие, определенное для P с помощью логических комбинаций постусловий выполнения примитивов процесса P, связанных между собой операциями конкатенации и слияния. В этом случае β назовем постусловием выполнения P:

$$\beta = \bigwedge_{u \in U_1(P)} \beta(u), \; U_1(p) \subseteq U(P).$$

Теорема 1. Пусть P — шаблон бизнес-процесса, а α и β — соответственно предусловие инициализации и постусловие выполнения, определенные для P. Задача формальной верификации шаблона P разрешима относительно α и β.

Доказательство выполняется в два шага. На первом предлагается применить алгоритм преобразования циклического шаблона в ациклический. Далее рассмотрен формальный алгоритм верификации для ациклических ВРТ.

4. ПРЕОБРАЗОВАНИЕ ЦИКЛИЧЕСКОГО ШАБЛОНА БИЗНЕС-ПРОЦЕССА В АЦИКЛИЧЕСКИЙ

Ниже представлены основные процедуры и определения, используемые алгоритмом преобразования, основанном на идее интервалов, предложенных впервые для процедуры анализа потока данных программы [8], а также соответствующие определения интервала фактор-графа.

Для узла h интервал $I(h)$ — максимальный единственный подграф входа, где h — единственный узел входа и все циклы содержат h. Уникальный узел интервала h называют входной вершиной интервала. Здесь и далее будем использовать обозначение $a \in I(h)$ вместо $a \in N(I(h))$ и обозначение $e \in I(h)$ вместо $e \in E(I(h))$.

Приведем формальные описания основных процедур, используемых в преобразовании цикла.

Процедура 1: construct_IntervalSet

Вход: Циклический ВРТ $P_E =< T, N, C, E, \Phi, \Phi_C, i, o, \Psi, \Delta >$ с графом $G(N, E)$

Выход: множество интервалов S_C

Метод:

1. Построить множество интервалов S с помощью алгоритма поиска интервалов [18].

2. $S_C = \{I(h) \in S \mid \exists (a, h, p) \in E \& (a, h, p) \in I(h)\}$.

3. Для всех $I(h) \in S_C$ выполнить

$L = \{a \in I(h) \mid a \in N, \; \text{path}(a, h) = \varnothing\}, I(h) = I(h) \setminus L;$

$I(h) = I(h) \setminus \{e \in I(h) \mid \; e \in E, \pi_1(e) \in L \;\; \text{OR} \;\; \pi_2(e) \in L\};$

4. Возвратить S_C.

Влияние на условия корректности: Нет.

Процедура 2: remove_Interval

Вход:

- Циклический ВРТ P_1 с графом $G_1(N_1, E_1)$
- Множество значений M_1 существенных выходных переменных
- $I(h)$ — интервал
- $R_{I(h)}$ — множество регистров состояния

Выход: Циклический ВРТ P_2 с графом $G_2(N_2, E_2)$. Множество значений M_2 существенных выходных переменных.

Метод:

1. $P_2 \leftarrow P_1; N_2 \leftarrow (N_1 \cup A_{I(h)}) \setminus \{a \in N_1 \mid a \in I(h)\}; E_2 \leftarrow E_1 \setminus \{e \in E_1 \mid e \in I(h)\}$; $M' \leftarrow M_1; M_2 \leftarrow \varnothing$
2. $i(A_{I(h)}) \leftarrow \varnothing; o(A_{I(h)}) \leftarrow \varnothing; o^*(A_I(h)) \leftarrow \varnothing$
3. Для всех $a \in I(h), b \in N, b \notin I(h)$ выполнить
   3.1. $\Delta_2(b, A_{I(h)}) \leftarrow \varnothing; \Delta_2(A_{I(h)}, b) \leftarrow \varnothing$
   3.2. Для всех $(w, v) \in \Delta_1(b, a)$ выполнить
   - Определить новую переменную v' так, чтобы $\mathrm{Dom}(v') = \mathrm{Dom}(v)$
   - $V_2 \leftarrow V_2 \setminus \{v\} \cup \{v'\}; i(A_{I(h)}) \leftarrow i(A_{I(h)}) \cup \{v'\}$
   - $\Delta_2(b, A_{I(h)}) \leftarrow \Delta_2(b, A_{I(h)}) \cup (w, v')$

   3.3. Для всех $(v, w) \in \Delta_1(a, b)$ выполнить
   - Определить новую переменную v' так, чтобы $\mathrm{Dom}(v') = \mathrm{Dom}(v) \cup \{\perp\}$
   - $V_2 \leftarrow V_2 \setminus \{v\} \cup \{v'\}; o(A_{I(h)}) \leftarrow o(A_{I(h)}) \cup \{v'\}$
   - $\Delta_2(A_{I(h)}, b) \leftarrow \Delta_2(A_{I(h)}, b) \cup (v', w)$
   - Если $v \in M_1$, то $M_2 \leftarrow M_2 \cup v'; M' \leftarrow M' \setminus v$
4. Для всех $a \in I(h), b \in C, b \notin I(h)$ выполнить
   4.1 Для всех $(v, w) \in \Delta_1(a, b)$ выполнить
   - Определить новую переменную v' так, чтобы $\mathrm{Dom}(v') = \mathrm{Dom}(v) \cup \{\perp\}$
   - $V_2 \leftarrow V_2 \setminus \{v\} \cup \{v'\}; o(A_{I(h)}) \leftarrow o(A_{I(h)}) \cup \{v'\}$
5. Для всех $r \in R_{I(h)}$ выполнить
   - Определить новую переменную v_r так, чтобы $\mathrm{Dom}(v_r) = \{0,1,2\}$
   - $V_2 \leftarrow V_2 \cup \{v_r\}, i(A_{I(h)}) \leftarrow i(A_{I(h))} \cup \{v_r\}$
   - $o^*(A_{I(h)}) \leftarrow o^*(A_{I(h)}) \cup v_r$
6. Для всех $v \in M'$ выполнить
   - Определить новую переменную v' так, чтобы $\mathrm{Dom}(v') = \mathrm{Dom}(v)$
   - $V_2 \leftarrow V_2 \setminus \{v\} \cup \{v'\}; o(A_{I(h)}) \leftarrow o(A_{I(h)}) \cup \{v'\}, M_2 \leftarrow M_2 \cup v'$
7. $o(A_{I(h)}) \leftarrow o(A_{I(h)}) \cup o^*(A_I(h))$
8. $\Psi(A_{I(h)}) \leftarrow < \{a \in N \mid a \in I(h)\}, \{e \in E \mid e \in I(h)\}, o^*(A_I(h)) >$;
9. Возвратить P_2, M_2.

Влияние на условия корректности:

Существенные переменные: Изменены области значений (шаг 4)

Семантика процесса, предусловие, постусловие: Нет

Процедура 3: construct_BranchActCond

Вход:

- Циклический ВРТ P_1 с графом $G_1(N_1, E_1)$
- Оператор A из интервала $I(h)$
- $I(h)$ — интервал
- $BR_{I(h)}$ — множество ветвящихся операторов
- $R_{I(h)}$ — множество регистров состояний для интервала

Выход: Условия p^c и p^0

Метод:

1. $A{\downarrow} = \{vr' \mid r' \in R_{I(h)}$ — регистр состояния $b \in BR_{I(h)}$ и $\text{path}(b, A) \neq \varnothing$ и $v_{r'} \in o^*(A_{I(h)})$ — переменная состояния для $r'\}$

2. $A{\uparrow} = \{v_{r'} \mid r' \in R_{I(h)}$ — регистр состояния $b \in BR_{I(h)}$ и $\text{path}(A, b) \neq \varnothing$ и $v_{r'} \in o^*(A_{I(h)})$ является переменной состояния для $r'\}$

3. $\text{ro}{\downarrow} = \{w \mid \exists b \in I(h), \exists d \notin I(h);\ \text{path}(b, A) \neq \varnothing, (w, v) \in \Delta(b, d)\}$

4. $\text{ro}{\uparrow} = \{w \mid \exists b \in I(h), \exists d \notin I(h);\ \text{path}(A, b) \neq \varnothing, (w, v) \in \Delta(b, d)\}$

5. Определить $p^c = \left(\underset{a \in A{\downarrow}}{\&} a = 2\right) \& \left(\underset{a \in A{\uparrow}}{\&} a = 1\right) \& \left(\underset{a \in ro{\downarrow}}{\&} v \neq \perp\right) \& \left(\underset{a \in ro{\uparrow}}{\&} v \neq \perp\right)$

6. Определить $p^0 = \left(\underset{a \in A{\downarrow}}{\&} a = 1\right) \& \left(\underset{a \in A{\uparrow}}{\&} a = 0\right) \& \left(\underset{v \in ro{\downarrow}}{\&} v \neq \perp\right) \& \left(\underset{v \in ro{\uparrow}}{\&} v = \perp\right)$

7. Возвратить p^c и p^0.

Влияние на условия корректности: Нет.

Процедура 4: construct_TransitionAndDataCon

Вход:

- Циклический ВРТ P_1 с графом $G_1(N_1, E_1)$ (до замены интервала)
- Циклический ВРТ P_2 с графом $G_2(N_2, E_2)$ (после замены интервала)
- $I(h)$ — интервал
- $BR_{I(h)}$ — множество ветвящихся операторов
- $R_{I(h)}$ — множество регистров состояний для интервала

Выход: Циклический ВРТ P_2 с графом $G_2(N_2, E_2)$

Метод:

1. $P_2 \leftarrow P_1; A_{\text{out}} = \varnothing$
2. Для всех $a \in I(h)$, $b \in N_1 \setminus I(h)$, $e =< a, b, p > \in E_1$ выполнить
   - $E_2 \leftarrow E_2 \setminus \{e\}$
   - $E^*_{I(h)} \leftarrow E^*_{I(h)} \cup \{e\}$
3. Для всех $A \in BR_{I(h)}, e \in E^*_{I(h)}$ выполнить
   - $\{p^c, p^0\} \leftarrow \text{construct\_ BranchActCond}(P_1, A, BR_{I(h)}, R_{I(h)})$
   - Определить $v^*_e = \{v' \mid \text{Dom}(v') = \text{Dom}(v), v \in o^*(A_{I(h)})\}$
   - Определить $p' = \pi_3(e) \& (p^c \vee p^0)$ так чтобы $i(p') = i(\pi_3(e)) \cup v^*_e$
   - Если $\pi_2(e) \notin A_{\text{out}}$, то $A_{\text{out}} \leftarrow A_{\text{out}} \cup \{\pi_2(e)\}; P_{\pi 2(e)} \leftarrow p'$ иначе $P_{\pi 2(e)} \leftarrow P_{\pi 2(e)} \vee p'$
   - $C_2 \leftarrow C_2 \setminus \{\pi_3(e)\}$

4. Для всех $a \in A_{\text{out}}$ выполнить

- $C_2 \leftarrow C_2 \cup \{P_a\}$; $E_2 \leftarrow E_2 \cup \{< A_{I(h)}, a, P_a >\}$; $\Delta_2(A_{I(h)}, P_a) \leftarrow \varnothing$
- Определить $\Delta_2(A_{I(h)}, P_a) = \{(w,v) | (w,v) \in \Delta(A_{I(h)}, p),\ e =< b, a, p > \in E_1,$ $b \in I(h)\} \cup \{(w,v) | w \in o^*(A_{I(h)}) \& v \in v_e^*\}$

5. Возвратить P_2.

Влияние на условия корректности: Нет.

Ниже приведен алгоритм преобразования циклического ВРТ в ациклический.

Алгоритм Т. Преобразование циклического шаблона бизнес-процесса

Вход:

- Циклический ВРТ $P_E =< T, N, C, E, \Phi, \Phi_C, i, o, \Psi, \Delta, M >$
- Множество значений M_E существенных выходных переменных

Выход:

- Ациклический ВРТ $P_W =< T', N', C', E', \Phi', i', o', \Psi', \Delta', M' >$
- Множество значений M_W существенных выходных переменных

Метод:

1. $N' \leftarrow N; C' \leftarrow C; E' \leftarrow E; M' \leftarrow M; C' \leftarrow C; \Phi' \leftarrow \Phi; P_1 \leftarrow P_E; M_1 \leftarrow M_E$
2. $S_C \leftarrow$ construct_IntervalSet (P_E) .
3. Для всех $I(h) \in S_C$ выполнить

   a) построить множество ветвящихся операторов:

$$BR_{I(h)} \leftarrow \{a \in I(h) | \exists b \notin I(h),\ \exists d \in I(h), < a, b, t_1 > \in E, < a, d, t_2 > \in E\}.$$

   b) построить множество регистров состояний:

$$R_{I(h)} \leftarrow \{r_a \mid r_a \text{ является регистром состояния } a \in BR_{I(h)}\}$$

   c) $< P_2, M_2 > \leftarrow$ remove_ Interval$(P_1, M_1, I(h), R_{I(h)})$

   d) $P_1 \leftarrow$ construct_ TransitionAndDataCon$(P_1, P_2, I(h), BR_{I(h)}, R_{I(h)})$

4. Возвратить P_1 и M_2.

Имеет место следующее утверждение.

Лемма 5. Пусть $P \in AT$ — циклический ВРТ. Тогда $Q = T(P)$ — ациклический ВРТ, так что имеет место следующее утверждение:

$$[T(P) \text{ корректен/некорректен} \Rightarrow \text{P корректен/некорректнен.}]$$

Доказательство. Пусть P — циклический ВРТ. Пусть α, β — соответственно пред- и постусловия для P, M — множество существенных выходных переменных P. Тогда P корректен относительно α и β, если $[\alpha(M) = 1 \Rightarrow \beta(MP) = 1]$.

Предположим, что применен алгоритм преобразования циклического P в ациклический ВРТ Q. Поскольку алгоритм не меняет семантики процесса с точки зрения формальной верификации, то процесс Q эквивалентен процессу P с точки зрения верификации. Пусть α_1, β_1 — соответственно пред- и постусловия для Q, а M_Q— множество существенных выходных переменных Q. Предусловие (α_1) и постусловие (β_1) процесса Q логически эквивалентны α и β соответственно, так что $\alpha(M) = 1 \Rightarrow \alpha_1(M_Q) = 1 \Rightarrow \beta_1(M_Q Q)) = 1 \Rightarrow \beta(MP) = 1$.

Утверждение доказано.

Это утверждение доказывает, что для каждого циклического ВРТ существует ациклический ВРТ, эквивалентный начальному с точки зрения формальной верификации.

5. ФОРМАЛЬНАЯ ВЕРИФИКАЦИЯ АЦИКЛИЧЕСКОГО ШАБЛОНА БИЗНЕС-ПРОЦЕССА

Ниже введено понятие трафарета для BPTs (примитивы для описания подмножеств состояний среды) [10, 11].

Разделенное множество — это кортеж, компоненты которого являются подмножествами соответствующих областей переменных. Разделенный трафарет — это пара двух разделенных множеств. Одно из них аппроксимирует множество текущих состояний процесса снизу, в то время как другое аппроксимирует сверху. Обозначим M множества всех трафаретов.

Для формальной верификации ациклических шаблонов предложен алгоритм верификации A-процессов [10].

Имеет место следующее утверждение о корректности BPT относительно α и β, основанное на сравнении разделенных множеств.

Лемма 6. Пусть P — BPT, α — предусловие и β — постусловие, определенные для P. Тогда имеют место следующие утверждения:

а) если $\pi_2(\text{finalMold}) \subseteq \pi_1(\text{mold}(\beta))$, то P корректен относительно α и β;

б) если $\neg(\pi_1(\text{finalMold}) \subseteq \pi_2(\text{mold}(\beta))$, то P не корректен относительно α и β;

в) иначе утверждение о корректности P относительно α и β не определено.

Алгоритм формальной верификации ациклических бизнес-процессов (далее — Алгоритм AV) и доказательство леммы 6 детально представлены в [10].

5.1. Формальная верификация примитивов. Утверждение 1. Пусть u — простой примитив, $\alpha(u)$ — предусловие инициализации и $\beta(u)$ — постусловие выполнения для u. Тогда алгоритм AV дает определенный ответ (корректен/некорректен) о корректности u относительно $\alpha(u)$ и $\beta(u)$.

Доказательство. Пусть u — простой примитив. Тогда $\text{In}(u) = \text{Out}(u) = Act(u) = \{a\}$.

$\text{Pre} = (in_1(1) \neq \perp \text{ AND} \ldots \text{AND} in_{k1}(1) \neq \perp)$

$\text{Post} = (mb(a_1) = \text{TRUE})$

Трафарет определяется методом добаления соотвественных входных переменных a_1 и $mb(a_1) =>$ mold $\equiv< in_1(1), \ldots, in_{k1}(1), mb(a_1) >$. После вычисления трафаретов для пред- и постусловий имеем:

mold(Pre) = = <<‘Known’ , … , ‘Known’, $\perp$> , <‘Known’ , … , ‘Known’, $\perp$ >>;

mold(Post) = = <<{‘Known’, $\perp$} , … , {‘Known’, $\perp$} , TRUE>, <{‘Known’, $\perp$}, ... … , {‘Known’, $\perp$} , TRUE>> .

После выполнения процесса получим трафарет finalMold = <<‘Known’, … ... , ‘Known’, TRUE>, <‘Known’, … , ‘Known’, TRUE>>. Условие $(\pi_2(\text{finalMold}) \subseteq \subseteq \pi_1(\text{mold}(\beta)))$ удовлетворено для процесса, поскольку <‘Known’,... … , ‘Known’, TRUE > $\subseteq$ < {‘Known’, $\perp$} , … , {‘Known’, $\perp$} , TRUE > . Это означает, что процесс корректен (лемма 6).

Утверждение доказано.

Утверждение 2. Пусть u — примитив синхронизации, $\alpha(u)$ — предусловие инициализации и $\beta(u)$ — постусловие выполнения для u. Тогда алгоритм AV дает определенный ответ (корректен/некорректен) о корректности u относительно $\alpha(u)$ и $\beta(u)$.

Утверждение 3. Пусть u — примитив разделения, $\alpha(u)$ — предусловие инициализации и $\beta(u)$ — постусловие выполнения для u. Тогда алгоритм AV дает определенный ответ (корректен/некорректен) о корректности u относительно $\alpha(u)$ и $\beta(u)$.

Лемма 7. Пусть u — примитив любого типа, $\alpha(u)$ — предусловие инициализации и $\beta(u)$ — постусловие выполнения для u. Тогда алгоритм AV дает определенный ответ (корректен/некорректен) о корректности u относительно $\alpha(u)$ и $\beta(u)$.

Доказательство леммы 7 непосредственно следует из утверждений 1–3.

Обозначим BasicUnit$V(N_u, E_u)$ процедуру проверки корректности примитива u с графом $G(N_u, E_u)$.

5.2. Формальная верификация для операций конкатенации и слияния. Лемма 8. Пусть P_1 и P_2 — такие ациклические шаблоны процессов, что $\text{In}(P_1) = \{ip_1\}$, $\text{Out}(P_1) = \{op_1, \ldots, op_{k1}\}$, $\text{In}(P_2) = \{ip'_1\}, \text{Out}(P) = \{op'_1, \ldots, op'_{k2}\}, k1, k2 > 0$.

Если P_1 корректен относительно $\alpha(P_1)$ и $\beta(P_1)$, а P_2 корректен относительно $\alpha(P_2)$ и $\beta(P_2)$, то ВРТ $P' = \text{Concat}(P_1, P_2, k, c)$ также корректен относительно предусловия $\alpha(P') = \alpha(P_1) \& \alpha(P_2) \& c$ и постусловия $\beta(P') = \beta(P_1) \& \beta(P_2)$. Если P_1 не корректен относительно $\alpha(P_1)$ и $\beta(P_1)$ или P_2 не корректен относительно $\alpha(P_2)$ и $\beta(P_2)$, то P' также не корректен относительно $\alpha(P')$ и $\beta(P')$.

Доказательство. P_1 и P_2 — такие ациклические шаблоны, что $\text{In}(P_1) = \{ip_1\}$, $\text{Out}(P_1) = \{op_1, \ldots, op_{k1}\}, \text{In}(P_2) = \{ip'_1\}, \text{Out}(P) = \{op'_1, \ldots, op'_{k2}\}$, $k1, k2 > 0$.

Тогда $P' = \text{Concat}(P_1, P_2, k, c) = P_1 +_{c,k} P_2$ — ациклический ВРТ. Пусть условие $\alpha(P') = \alpha(P_1) \& \alpha(P_2) \& c$ удовлетворяется до выполнения P'. Тогда $\alpha(P_1)$ также удовлетворяется. В результате P_1 корректен относительно $\alpha(P_1)$ и $\beta(P_1)$. Если P некорректен, то $\beta(P_1)$ не удовлетворяется. Тогда $\beta(P')$ также не удовлетворяется и соответственно P' не корректен относительно $\alpha(P')$ и $\beta(P')$. Если P_1 некорректен и c, $\alpha(P_2)$ удовлетворяются вследствие того, что удовлетворяется $\alpha(P')$, то P_2 активизируется. Если P_2 некорректен, то $\beta(P_2)$ не удовлетворяется. В этом случае $\beta(P')$ также не удовлетворяется и соответственно P' не корректен относительно $\alpha(P')$ и $\beta(P')$. Если P_2 корректен, то $\beta(P_2)$ удовлетворяется. В результате верно следующее: $\beta(P') = \beta(P_1) \& \beta(P_2)$.

Утверждение доказано.

Лемма 9. Пусть u — примитив синхронизации и P — ациклический шаблон такие, что $\text{In}(P) = \{ip_1\}, \text{Out}(P) = \{op_1, \ldots, op_{ko}\}, ko > 0, \text{Out}(u) = \{ou_1\}, \text{In}(u) = \{iu_1, \ldots, iu_m\}$. Если u корректен относительно $\alpha(u)$ и $\beta(u)$, а также P корректен относительно $\alpha(P)$ и $\beta(P)$, то $P' = \text{Merge}(u, P, \bar{k}, \bar{c})$ корректен относительно предусловия $\alpha(P') = \alpha(u) \& \ \& \alpha(P) \& c_1 \& \ldots \& c_m$ и постусловия $\beta(P') = \beta(u) \& \beta(P)$, если $\bar{c} = < c_1, \ldots, c_m >$. Если u не корректен относительно $\alpha(u)$ и $\beta(u)$ или P не корректен относительно $\alpha(P)$ и $\beta(P)$, то P' также не корректен относительно $\alpha(P')$ и $\beta(P')$.

Доказательство аналогично предыдущему доказательству.

6. АЛГОРИТМ ФОРМАЛЬНОЙ ВЕРИФИКАЦИИ ДЛЯ ШАБЛОНОВ БИЗНЕС-ПРОЦЕССОВ

Ниже представлен алгоритм формальный верификации для шаблонов относительно предусловия инициализации и постусловия выполнения.

Алгоритм V. Формальная верификация ВРТ.

Вход: ВРТ P с графом $G(N, E)$

Выход: *Корректен/Некорректен*

Метод:

1. Если $N == \varnothing$, то возвратить *Корректен*
2. Если P имеет цикл, то $P \leftarrow T(P)$
3. Для всех $A \in N$ выполнить
   - counter(A) $\leftarrow |A^{\rightarrow}|$.
   - если counter(A) = 0, то добавить A к Front-у
4. bUnit $\leftarrow 0, N_p \leftarrow N, E_p \leftarrow E$
5. Пока !bUnit выполнить
   - 5.1. выбрать A из Front-а, $N_u \leftarrow \{A\}, E_u \leftarrow \varnothing, N_p \leftarrow N_p / \{A\}$
   - 5.2. Если $|A^{\leftarrow}| > 1$, то
     - a) bUnit $\leftarrow 1$
     - b) Для всех $e \in A^{\leftarrow}, e = (A', A, p)$ выполнить

$N_u \leftarrow N_u \cup \{A'\}, N_p \leftarrow N_p / \{A'\}, E_u \leftarrow E_u \cup \{e\}, E_p \leftarrow E_p / \{e\}$

5.3. Иначе если $A^{\leftarrow}| = 1, A^{\leftarrow} = \{< A', A, p >\}$, то

a) Если $|A'^{\leftarrow}| = 1$, то bUnit $\leftarrow 1$

b) Иначе если $A'^{\leftarrow} \subseteq$ Front, то

c) $N_u \leftarrow \{A'\}$, bUnit $\leftarrow 1$

d) Для всех $e \in A'^{\leftarrow}, e = (A', A'', p)$ выполнить

$N_u \leftarrow N_u \cup \{A''\}, N_p \leftarrow N_p / \{A''\}, E_u \leftarrow E_u \cup \{e\}, E_p \leftarrow E_p / \{e\}$

e) Иначе удалить A из Front-a

6. Если BasicUnit $V(N_u, E_u) \& V(N_p, E_p)$, то возвратить *Корректен* иначе возвратить *Некорректен*.

Теорема 2. Пусть P — шаблон бизнес-процесса, α и β — соответственно предусловие инициализации и постусловие выполнения для P. Тогда алгоритм верификации (алгоритм V) дает определенный ответ (корректен/некорректен) о корректности P относительно α и β.

Доказательство теоремы 2 — прямое следствие лемм 7–9. Таким образом, доказано, что проблема формальной верификации разрешима для шаблонов бизнес-процессов.

7. ПРЕДСТАВЛЕНИЕ ITIL С ПОМОЩЬЮ ШАБЛОНОВ БИЗНЕС-ПРОЦЕССОВ

Предложенные в работе шаблоны бизнес-процессов используются для описания библиотеки IT Infrastructure Library (ITIL) — библиотеки инфраструктуры информационных технологий (www.itil-officialsite.com), содержащей множество шаблонов процессов для управления IT услугами [3–6]. Библиотека построена на основе современных методов управления и служит в качестве де-факто стандартного руководства для создания процессов IT управления [5].

IBM, HP и Microsoft поставляют собственные библиотеки процессов для управления IT (IBM PRM-IT 2007; HP ITSM 2003; 2008 MOF). Анализ этих библиотек показал, что все содержащиеся в них шаблоны бизнес-процессов построены из примитивов, предложенных в настоящей работе, с использованием операций конкатенации, слияния и цикла. Таким образом, описанный подход, помимо верификации библиотеки ITIL, позволяет решить проблему формальной верификации и для библиотек PRM-IT, ITSM, MOF.

Автор выражает благодарность академику НАН Армении, доктору физико-математических наук, профессору С.К. Шукуряну.

СПИСОК ЛИТЕРАТУРЫ

1. The infrastructure optimization journey. — Microsoft Corporation, 2008.
2. Aileen C., Wui-Gee T., Mark T. Summary of ITIL Adoption Survey Responses. Techn. Rep., itSMF Australia Conf., Toowoomba, Australia. — 2006.
3. IT Infrastructure Library, IT Service Management. — http://www.itil.co.uk/.
4. Introducing the IBM process reference model for IT, Second Edition, IBM, January 2007.
5. Turner M.S. Microsoft solutions framework essentials. — Microsoft Press, 2006.
6. Pultorak D. Microsoft operations framework (MOF): A pocket guide. — Amsterdam: Van Haren Publ., 2005.
7. Leymann F., Roller D. Production workflow: Concepts and techniques. — New York: Prentice Hall Press, 2000.
8. Allen F.E., Cocke J. A program dataflow analysis procedure // Com. of the ACM. — 1976. — **19**(3). — P. 137–147.
9. Hecht M.S., Ullman J.D. Characterizations of reducible flow graphs // J. ACM — 1974. — **21**, N 3. — P. 367–375. DOI= http://doi.acm.org/10.1145/321832.321835.
10. Kostanyan A., Varosyan A. Partial recognizing algorithm for verification of workflow processes // FUBUTEC'2008. — 2008. — P. 89–94.
11. Shoukourian S., Kostanian A., Margarian V., Ashour A. An approach for system tests design and its applications // Proc. of the 13-th IEEE VLSI Test Symposium, NJ, Princeton, USA. — 1995. — P. 448–453.

Поступила 14.07.2010