Е.Б. Самойлов,
кандидат техн. наук;
В.В. Шмелёв,
кандидат техн. наук

МЕТОД СИНТЕЗА ФУНКЦИОНАЛЬНО-ЛОГИЧЕСКИХ ПРОГРАММ ПОТОКОВЫХ ВЫЧИСЛЕНИЙ ПРИ МОНИТОРИНГЕ СОСТОЯНИЯ ТЕХНИЧЕСКИХ СИСТЕМ

Предлагается научно-методическая основа мониторинга процессов функционирования космических средств. Мониторинг представляется потоковыми вычислениями над технической и технологической информацией, формируемой объектом контроля. Классическая процедура разработки программы вычислений адаптируется к использованию аппарата формального описания алгоритмов с помощью модифицированных цветных сетей Петри и G-сетей, которые объединяются термином «вычислительные сети». В качестве исходной информации для синтеза используются таблицы хронометража и диаграммы Гантта, как наиболее распространенные способы представления циклограмм функционирования космических средств. Отличительным достоинством предложенного метода синтеза является возможность корректировки создаваемой программы с целью выявления и компенсации ошибок синтеза. Для обеспечения такой возможности предусматриваются процедуры синтаксической и семантической верификации программы. Практическая значимость метода заключается в учете при синтезе программ потоковых вычислений факторов, негативно воздействующих на процессы формирования, сбора, обработки и анализа телеметрической информации космических средств. Изложение сопровождается примером синтеза программы потоковых вычислений.

Ключевые слова: теория вычислений, функционально-логическое программирование, потоковые вычисления, обработка измерительной информации, мониторинг технических систем.

ВВЕДЕНИЕ

Системный анализ [1] причин аварий в отечественной космической отрасли позволяет сделать вывод о наличии значимого недостатка в системе мониторинга состояния технических систем (ТС) космических средств (КСр). Сегодня назрела острая необходимость изменения подхода к анализу телеметрической информации КСр. Если раньше было достаточно определения точечных значений контролируемых характеристик и сравнения их с допусковыми границами, то сегодня необходимо оценивание правильности процессов функционирования систем объекта мониторинга в целом. Для достижения этой цели предлагается проводить мониторинг КСр на основе функционально-логических программ потоковых вычислений по разнородной информации, получаемой в результате испытаний и применения КСр. Методу синтеза таких программ и посвящена статья.

В статье используется следующий ряд понятий. *Программой* называется система вычислений, которая для некоторого набора исходных данных позволяет по однозначно определенной последовательности действий, выполняемых «механически» (без участия человека) получить результат [2]. *Функционально-логической программой* называется программа, создаваемая в результате применения принципов функционального программирования с реализацией логических взаимосвязей между операторами. Под *потоковыми вычислениями* понимается обработка информации различного рода (измерительной, технологической) с целью определения состояния ТС, т. е. осуществления мониторинга ТС. Особенностью процесса обработки такой информации является обеспечение параллелизма и асинхронности выполнения совокупностей операторов [3].

ПОСТАНОВКА ЗАДАЧИ СИНТЕЗА ПРОГРАММЫ ПОТОКОВЫХ ВЫЧИСЛЕНИЙ

Исходные данные для синтеза программы

Исходная спецификация программы потоковых вычислений при мониторинге процесса R^I (кратко – программа R^I) представляется кортежем:

$$R^I = \langle S^I, L \rangle, \quad (1)$$

где S^I – это множество операций программы R^I, $S^I = \{S_k^I | \ k = 1, \ldots, card(I_s)\}$, при этом I_s – множество номеров операций;

L – множество кортежей, обуславливающих выполнение операций, $L = \{l_k | \ k = 1, \ldots, card(I_s)\}$.

Кортеж l_k определяется следующим образом:

$$l_k = \langle K_k(x), t_k, \tau_k \rangle, \quad (2)$$

где $K_k(x)$ – управляющий кортеж для операции S_k^I, $K_k(x) = \langle B_b^{(k)}(x), B_f^{(k)}(x) \rangle$, при этом $B_b^{(k)}(x)$ – предикат, обуславливающий начало выполнения операции, b – «begin»; $B_f^{(k)}(x)$ – предикат, обуславливающий окончание выполнения операции, f – «finite», x – аргумент предикатов $B_b^{(k)}(x)$ и $B_f^{(k)}(x)$, являющийся типом информации, используемой для мониторинга, $x \in X$, причем X – множество возможных типов аргумента x, в предметной области статьи – это телеметрическая, техническая, технологическая информация;

t_k – момент начала выполнения операции S_k^I;

τ_k – длительность операции S_k^I.

Величины t_k и τ_k отсчитываются в единицах измерения контролируемого процесса функционирования КСр. Такими единицами могут быть как временные метки системы единого времени, так и события при событийном характере контролируемого процесса.

Требуется найти:

1. Структуру формального описания функционально-логической программы R (кратко – программа R) потоковых вычислений.
2. Структуру формального описания унифицированной модели типовой операции (УМТО) S программы R потоковых вычислений.
3. Структуру формального описания технических ограничений Q программы R.
4. $O: R^I \to R$ – структуру оператора O преобразования программы R^I в R − оператор синтеза функционально-логической программы R.
5. $\mu: R \to R^I$ – структуру оператора ресинтеза μ программы R^I по R для проверки частичной правильности программы по идентичным пред- и постусловиям и проверки завершаемости программы.
6. $\eta = (\eta_1 \cap \eta_2 \cap \eta_3)$ – структуру предиката η проверки сохранения свойства вычислимости в синтезированной программе R, η_1 – предикат проверки непротиворечивости программы R^I, η_2 – корректности и η_3 – активности.

КОНЦЕПЦИЯ СИНТЕЗА ФУНКЦИОНАЛЬНО-ЛОГИЧЕСКОЙ ПРОГРАММЫ

В качестве модельной основы синтеза программ предлагается использовать математический аппарат сетей Петри. Это обосновывается тем, что данный абстрактный инструмент считается пригодным для моделирования именно потоковых вычислений, характеризующих-

ся асинхронностью и параллелизмом [4]. Для сетей Петри определен перечень свойств, таких, как покрываемость, достижимость, активность, ограниченность и эквивалентность, которые после интерпретации могут быть использованы для контроля свойств корректности вычислений.

Анализ известных инструментальных сред проектирования схем или моделей программ на основе сетей Петри (например [4–8]) показал, что они не обладают достаточными описательными возможностями. Кроме того, не вводятся процедуры верификации проектируемых программ, остаются только прорабатываемые в первую очередь процедуры непосредственного синтеза.

Таким образом, предлагается организовать мониторинг состояния КСр путем перевода (синтеза программ) потоковых вычислений из исходного описательного представления в прикладное с использованием модификаций сетей Петри. При этом необходимо обеспечить формальную верификацию и новые моделирующие возможности аппарата синтеза для учета потенциальных особенностей процессов функционирования КСр.

Мониторинг состояния КСр на основе функционально-логических программ потоковых вычислений организуется по схеме, состоящей из трех этапов, концептуальное содержание которых представлено на рис. 1.

Рис. 1. Схема организации мониторинга состояния ТС КСр на основе функционально-логической программы потоковых вычислений

Рассмотрим каждый этап, уделив основное внимание наиболее важному этапу II.

ИНТЕРПРЕТАЦИЯ ИСХОДНОЙ ИНФОРМАЦИИ

Процесс функционирования ТС является инвариантом процесса мониторинга состояния данной системы [11]. Это позволяет использовать в качестве исходной спецификации функционально-логической программы существующие представления процессов функционирования ТС, используемые в космической отрасли. Примем в качестве единого прототипа процесса функционирования ТС таблицу хронометража, пример которой приведен в табл. 1, а соответствующая ей диаграмма Гантта – на рис. 2.

Семантически пример спецификации описывает процесс, состоящий из трех последовательных операций O_1, O_2 и O_3, первая из которых начинается по внешнему сигналу *К*, а третья оканчивается по достижению граничного значения измеряемым параметром *П*.

Таблица 1

Пример таблицы хронометража

№	Операция	База	Время от базы	Длительность
1	O_1	*К*	0	t_1
2	O_2	Окон O_1	t_1	t_2
3	O_3	Окон O_2	t_2+t_1	*П*>1

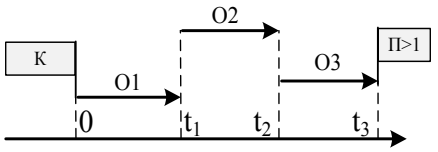

Рис. 2. Пример диаграммы Гантта

В соответствии со вторым шагом этапа I рис. 1 требуется представить исходную информацию в виде кортежа $R^I = \langle S^I, L \rangle$. Исходная спецификация – строгое представление информации о контролируемом процессе на основе модифицированной таблицы хронометража. Такое представление приведено в табл. 2.

Таблица 2

Пример исходной спецификации в виде кортежа $R^I = \langle S^I, L \rangle$ для процесса из таблицы 1

k	$S^I = \{S_k^I \mid k = 1,2,3\}$	$B_b^{(k)}$	$B_f^{(k)}$	t_k	τ_k
1	O_1	*К*	По длительности	0	t_1
2	O_2	Окон O_1	По длительности	t_1	t_2
3	O_3	Окон O_2	*П*>1	t_1+ t_2	t_3

Рассмотрим в качестве примера семантическое содержание некоторых предикатов из табл. 2. $B_b^{(1)} = K$ означает, что условие начала O_1 определяется моментом поступления команды *К*, в формальном выражении разрешением для начала O_1 является положительный результат проверки $B_b^{(1)}$, или $Pr\left(B_f^{(1)}\right) = true$. $B_f^{(1)} = По\ длительности$ означает, что условием окончания O_1 является достижение заданного значения t_1 длительностью выполнения данной операции. $B_b^{(3)} = Окон\ O_2$ означает, что условием начала O_3 является окончание выполнения O_2. $B_f^{(3)} = П > 1$ означает, что условием окончания O_3 является превышение значения 1 измеряемым параметром *П*.

МОДЕЛЬНОЕ ОБЕСПЕЧЕНИЕ СИНТЕЗА ПРОГРАММЫ ВЫЧИСЛЕНИЙ

В соответствии с постановкой задачи для синтеза программы необходимо разработать структуры программы *R*, УМТО *S* и технических ограничений *Q*. В статье данные элементы рассматриваются кратко, дополнительную информацию можно получить в [9].

Функционально-логической программой *R* потоковых вычислений называется теоретико-множественная конструкция на основе вычислительных сетей Петри следующего вида:

$$R = \langle S, J, Q \rangle \ . \qquad (3)$$

Для синтаксического представления УМТО *S* используется теоретико-множественный подход. В графологическом виде УМТО представлена на рис. 3. УМТО, по сути, является агентом или примитивом, которую можно назвать функцией разработанного «языка» контролируемых процессов. Структура УМТО описывается кортежем следующего вида:

$$S = \langle P, T, F, B, H^+, H^-, M \rangle,$$

где P – конечное непустое множество переменных модели, $P = P_{\text{вн}} \cup P_{in} \cup P_{out} = \left\{ p_i \middle| i \in I_p \right\}$ $P_{\text{вн}}$, при этом P_{in}, P_{out} – множества внутренних, входных и выходных переменных соответственно, I_p – множество номеров переменных (переменные являются модификацией позиций в сетях Петри);

T – конечное непустое множество операторов модели, $T=\{t_j \mid j \in I_T\}$, при этом I_T – множество номеров операторов (операторы являются аналогом переходов в сетях Петри);

F – входная функция инцидентности, описывающая кратность входной дуги от переменной p_i к оператору t_j и ставящая в соответствие с каждой парой $\langle p_i, t_j \rangle$ целое неотрицательное число из множества N, $F{:}P\times T\to N$;

B – входная функция инцидентности, описывающая сбрасывающую дугу [9] от переменной p_i к оператору t_j и ставящая в соответствие с каждой парой $\langle p_i, t_j \rangle$ элемент бинарного множества $N_b=\{0, 1\}$, $B{:}P\times T\to N_b$;

H^+ – выходная функция инцидентности, описывающая кратность выходной дуги от оператора t_j в переменную p_i и ставящая в соответствие с каждой парой $\langle t_j, p_i \rangle$ число из множества N, $H^+{:}T\times P\to N$;

H^- – выходная функция инцидентности, описывающая кратность выходной извлекающей дуги [9] от оператора t_j в переменную p_i и ставящая в соответствие с каждой парой $\langle t_j, p_i \rangle$ число из множества N, $H^-{:}T\times P\to N$;

M – функция, которая с каждым элементом $p_i \in P$ ставит в соответствие элемент из множества N.

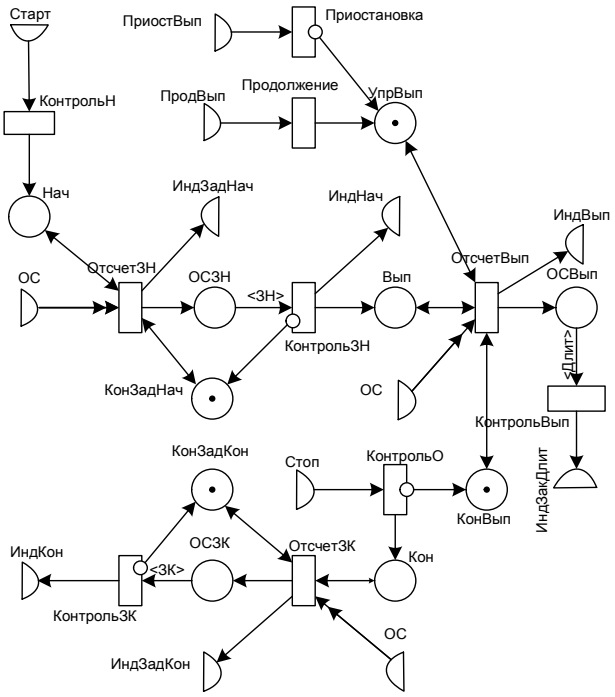

Рис. 3. Графический вид схемы УМТО

На рис. 3 входные переменные обозначены полукругом с выходящей стрелкой, внутренние – кругом, выходные – полукругом с входящей стрелкой, операторы обозначены прямоугольниками, входные дуги обозначаются простыми стрелками, сбрасывающая дуга обозначается стрелкой с двойным окончанием, выходные дуги обозначаются простыми стрелками, извлекающая дуга обозначена стрелкой с малым кружком в начале. Данная схема (за исключением входных и выходных переменных) может быть заменена специальным оператором-процедурой, представленным на рис. 4. На данном рисунке в левой части оператора-процедуры представлены входные переменные, в правой – выходные. Внутри оператора-процедуры представлены значения функции инцидентности F, определяющие целочисленные значения длительностей задержек начала и окончания выполнения операции, а также длительности непосредственного выполнения экземпляра УМТО.

Далее будем считать, что k-й экземпляр УМТО является оператором S_k функционально-логической программы R, для краткости обозначим его операцией S_k. Для проверки работы схемы УМТО использована среда CPN Tools [7, 8], позволяющая создавать и исследовать иерархические, модифицированные (цветные и расширенные) модели процессов.

Зададим $J = \{J_k \mid k=1,\ldots,card(I_s)\}$ – множество функций инцидентности между экземплярами УМТО. Его элемент можно представить как следующий кортеж:

$$J_k=\langle J_{in}^{(k)}{:}P_{out}^{(l)}\times P_{in}^{(k)}\to N,\ J_{out}^{(k)}{:}P_{out}^{(k)}\times P_{in}^{(m)}\to N\rangle, \qquad (4)$$

который описывает конкатенацию выходных переменных $P_{out}^{(l)}$ операции S_l и входных переменных $P_{in}^{(k)}$ операции S_k, а также выходных переменных $P_{out}^{(k)}$ операции S_k и входных переменных $P_{in}^{(m)}$ операции S_m, k, m, $l \in I_s$.

Выражение (4) показывает формальное задание функции инцидентности J_k, в графологическом виде конкатенация операторов-процедур представляется соединением или склеиванием соответствующих входных и выходных переменных экземпляров УМТО.

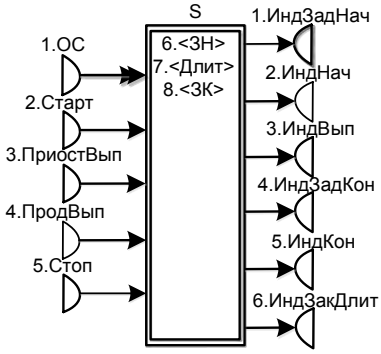

Рис. 4. Оператор-процедура

Зададим $Q = \{Q_k \mid k=1,\ldots,card(I_s)\}$ – множество отношений, определяющих варианты выполнения синтезируемой программы. $Q_k = \{q_c^{(k)} \mid c=1,\ldots,card(Q_k)\}$, где $q_c^{(k)}$ – c-е ограничение операции S_k, в математических формализмах данное ограничение является предикатом. Введем в модельное обеспечение синтеза программы потоковых вычислений z-модель (z – значение), представленную в графологическом виде на рис. 5. Z-моделью технических ограничений программы называется теоретико-множественная конструкция следующего вида:

$$q_c^{(k)} = \langle X, ZP, PT, ZF, H, ZM \rangle,$$

где X – множество возможных типов аргумента в ограничении, см. раскрытие кортежа (2);

ZP – конечное непустое множество z-переменных, $ZP = \{zp_i \mid i=1,\ldots,card(I_{ZP})\}$, при этом I_{ZP} – множество номеров z-переменных;

PT – конечное непустое множество предикатных операторов, $PT = \{pt_j \mid j=1,\ldots,card(I_{PT})\}$, при этом I_{PT} – множество номеров предикатных операторов;

ZF – входная функция инцидентности, описывающая входную, всегда однократную дугу от z-переменной zp_i к предикатному оператору pt_j, $ZF: ZP \times PT \to \{0, 1\}$;

H – выходная функция инцидентности, описывающая кратность выходной дуги от предикатного оператора pt_j в входную переменную $p_i \in P_{in}$, $H: PT \times P \to \{0, 1\}$;

ZM – функция, которая каждой z-переменной zp_i ставит в соответствие элемент d_i некоторого множества D возможных значений типа информации из множества X, $ZM: ZP \to D$.

При работе с конкретным типом $x \in X$ к указанным элементам добавляется нижний индекс (x). Тогда $d_{i(x)}$ – значение используемой при мониторинге информации типа x, например, телеметрируемого параметра x, из некоторого множества $D_{(x)}$ возможных значений типа информации $x \in X$ Таким образом, можно записать $zm_i: zp_i \to d_{i(x)}$, что означает: функция zm_i сопоставляет с z-переменной zp_i значение $d_{i(x)}$.

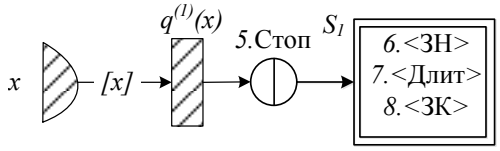

Рис. 5. Схема учета технических ограничений контролируемого процесса функционирования КСр в синтезируемой программе

На рис. 5 z-переменная обозначена заштрихованным полукругом, предикатный оператор – заштрихованным прямоугольником, дуги – стрелками.

Для учета в синтезируемой программе технических ограничений Q с предикатным оператором pt_j сопоставляется предикат $q_c^{(k)}(x)$, в котором аргументом является x. Установим, что предикатный оператор pt_j разрешен, если значение $d_{i(x)}$ его z-переменной zp_i обеспечивает истинность предикату $q_c^{(k)}(x)$. В формальном виде оператор pt_j с сопоставленным предикатом $q_c^{(k)}(x)$ и со значением $d_{i(x)}$ z-переменной zp_i разрешен, если $zm_i : d_{i(x)} \to Pr(q_c^{(k)}(d_{i(x)})) = true.$ Областью применимости $D(q_c^{(k)}(x))$ предикатного оператора pt_j называется такое множество значений $d_{i(x)}$ из $D_{(x)}$, все элементы которого обеспечивают $Pr(q_c^{(k)}(d_{i(x)})) = true$.

СИНТЕЗ ПРОГРАММЫ ПОТОКОВЫХ ВЫЧИСЛЕНИЙ

Итак, выше было рассмотрено модельное обеспечение синтеза функционально-логических программ потоковых вычислений. Теперь необходимо рассмотреть порядок его применения непосредственно при синтезе.

Этап II схемы организации мониторинга, представленной на рис. 1, является прикладной интерпретацией порядка реализации операторов O, μ и η (см. постановку задачи синтеза).

Подэтап 3 заключается в размножении экземпляров УМТО по количеству операций в процессе, что соответствует количеству строк в табл. 1 и 2. Кратность дуги <Длит> в операции S_k задается равной переменной τ_k:

$$F\left\langle \text{ОСВып,КонтрольВып}\right\rangle = \left\langle \text{Длит}\right\rangle = \tau_k .$$

Для примера исходной спецификации табл. 2 получим:

$$S = \left\{S_1, S_2, S_3\right\}. \qquad (5)$$

Для S_1 функция инцидентности $F\left\langle \text{ОСВып,КонтрольВып}\right\rangle = t_1$. Для S_2 и S_3 составляются аналогичные выражения.

Таким образом, по результатам выполнения подэтапа 3 графологическая схема синтезируемой программы наполнится тремя экземплярами УМТО. Полученная схема представлена на рис. 6. Кратности дуг <ЗН> и <ЗК> на схеме оператора-процедуры рис. 4, находящиеся внутри прямоугольника, приравниваются нулю. Кратность дуги <Длит > задается в соответствии с равенствами (5).

На подэтапах 4 и 5 формируется структура технологических особенностей вычислений, определяющих порядок выполнения операций. Необходимо определить переход от элементов кортежа (1) к входным переменным «Старт» и «Стоп» для каждого экземпляра УМТО с использованием выражения (4).

Значение переменной «Старт» определяется значением $Pr\left(B_b^{(k)}\right)$, значение переменной «Стоп» – значением $Pr\left(B_f^{(k)}\right)$

$$M\left(\text{Старт}\right) = \begin{cases} 1, \text{if } Pr\left(B_b^{(k)}\right) = \text{"true"} \\ 0, \text{if } Pr\left(B_b^{(k)}\right) = \text{"false"} \end{cases}; M\left(\text{Стоп}\right) = \begin{cases} 1, \text{if } Pr\left(B_f^{(k)}\right) = \text{"true"} \\ 0, \text{if } Pr\left(B_f^{(k)}\right) = \text{"false"} \end{cases}.$$

Для формальной интерпретации технологических ограничений установим, что значения выходных переменных операции S_l определяют условие выполнения экземпляра УМТО S_k. Четыре типа технологических ограничений, определяющих условия начала и окончания операций, задаются с помощью функций инцидентности. В качестве примера рассмотрим порядок задания ограничения типа «Окончание – начало».

Ограничение «Окончание – начало» означает, что S_k начнется после окончания S_l. Данное ограничение формализуется выражениями

$$J_{out}^{(l)}\left(\text{ИндЗакДлит,Старт}^{(k)}\right)=1 \text{ и } J_{in}^{(k)}\left(\text{ИндЗакДлит}^{(l)},\text{Старт}\right)=1.$$

Аргументами функций инцидентности являются имена переменных, между которыми формализована связь. Верхние индексы у имен переменных обозначают принадлежность к соответствующему экземпляру УМТО. Отсутствие индекса – принадлежность к экземпляру УМТО, для которого составлена функция инцидентности.

Аналогичным образом задаются ограничения «Начало – начало», «Начало – окончание» и «Окончание – окончание».

В соответствии с (4) и предикатными выражениями табл. 2 функции инцидентности для S_2 примут следующий вид:

$$J_2=\left\langle J_{in}^{(2)}\left(\text{ИндЗакДлит}^{(1)},\text{Старт}\right)=1,\ J_{out}^{(2)}\left(\text{ИндЗакДлит},\text{Старт}^{(3)}\right)=1\right\rangle.$$

Схема программы с учетом всех внесенных в нее функций инцидентности представлена на рис. 7.

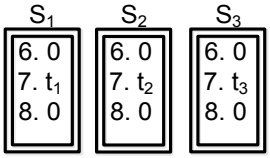

Рис. 6. Схема функционально-логической программы после подэтапа 3

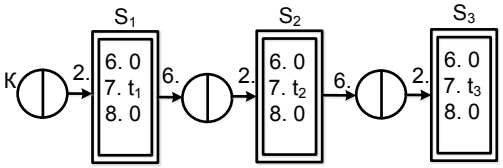

Рис. 7. Схема функционально-логической программы после подэтапа 5

На подэтапах 6 и 7 в схему программы вводятся технические ограничения контролируемого процесса с помощью *z*-моделей. Условие начала или окончания выполнения операции S_k от внешней информации задается с помощью функции инцидентности между предикатным переходом *z*-модели и входными переменными экземпляра УМТО:

$$J_{in}^{(k)}: PT\times P_{in}^{(k)}\to\{0,1\};\ P_{in}^{(k)}=\{\text{ОС},\text{Старт},\text{Стоп},\text{Приостановка},\text{Продолжение}\}.$$

Пример синтеза *z*-переменной и предикатного оператора для рассматриваемой исходной спецификации (см. табл. 2) представлен следующими выражениями:

$$ZP=\{П\};\ PT=\{q^{(3)}(П)=(П>1)\};\ ZF\langle П,\ q^{(3)}(П)\rangle=1;\ H\langle q^{(3)}(П),\text{Стоп}^{(3)}\rangle=1;\ zm(П)=0.$$

Интерпретируется функция *H* следующим образом: условием окончания операции S_3 является превышения значением *z*-переменной *П* значения «1».

Таким образом, результатом синтезирующих подэтапов 3–7 является схема функционально-логической программы, представленная на рис. 8.

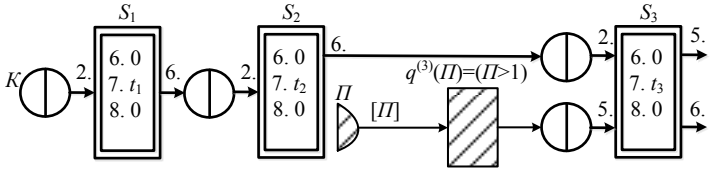

Рис. 8. Схема функционально-логической программы после подэтапа 7

ВЕРИФИКАЦИЯ ФУНКЦИОНАЛЬНО-ЛОГИЧЕСКОЙ ПРОГРАММЫ ПОТОКОВЫХ ВЫЧИСЛЕНИЙ

Подэтапы 8–11 (см. рис. 1) заключаются в верификации сформированной программы вычислений. Порядок верификации представлен на рис. 9. Полнота и очередность выявления и компенсации ошибок определяются решаемыми при проектировании задачами.

Верификация делится на две части. Первая часть «I. Синтаксическая верификация» заключается в проверке правильности интерпретации исходной спецификации.

Синтаксическая верификация заключается в выполнении стадий ресинтеза программы *R* в R^I, сопоставления результата ресинтеза и проверки завершаемости программы. Ресинтез схемы программы осуществляется путем формирования элементов кортежей (1) и (2) по содержимому кортежа (3) в три шага.

Шаг 1. Значение τ_k восстанавливается по следующей формуле:

$$\tau_k=F<\text{ОСВып}^{(k)},\ \text{КонтрольВып}^{(k)}>=<\text{Длит}^{(k)}>.$$

Шаг 2. Момент начала выполнения экземпляра УМТО S_k не контролируется, так как. данная величина является «планируемой» и зависит от конкретного варианта развития контролируемого процесса.

Шаг 3. Булевы переменные $B_b^{(k)}$ и $B_f^{(k)}$ восстанавливаются путем анализа функции инцидентности $J^{(k)}$. Порядок восстановления ограничения «Окончание – начало» может быть представлен следующей причинно-следственной связью:

$(J^{(k)}{}_{in}: P_{out}{}^{(l)} \times P_{in}{}^{(k)} \rightarrow N; P_{out}{}^{(l)} \subseteq \{\text{ИндЗакДлит}^{(l)}, \text{ИндЗак}^{(l)}\}, P_{in}{}^{(k)} \subseteq \{\text{Старт}\}) \Rightarrow B_b^{(k)} = (\text{Окон } O_l)$.

Данная связь определяет, если входная функция инцидентности $J^{(k)}{}_{in}$ устанавливает связь между выходной переменной «ИндЗакДлит$^{(l)}$» (или «ИндЗак$^{(l)}$») и входной переменной «Старт» экземпляра УМТО S_k, то условием начала S_k является окончание S_l, т. е. $B_b^{(k)}$=Окон O_l.

Аналогичные выражения составляются для остальных типов ограничений.

Сопоставление спецификаций осуществляется путем пооперационного сравнения компонентов программы с соответствующими компонентами исходной спецификации, после чего формируется невязка или ошибка синтеза программы. Такой подход является интерпретацией метода Флойда по верификации программ [4].

Вторая часть «II. Семантическая верификация» состоит в выявлении ошибок на уровне смыслового содержания операций программы вычислений. Пример экземпляра УМТО S_3 с некорректностью по входу и выходу представлен на рис. 10. Компенсация некорректности технологических ограничений заключается в добавлении оператора «ИЛИ» для компенсации некорректности по входу, умножающего оператора – для компенсации некорректности по выходу. В качестве примера на рис. 11 представлена компенсация некорректности S_3 по входу с помощью дополнительного «ИЛИ» оператора $t_\cap$.

Рис. 9. Схема верификации функционально-логической программы вычислений

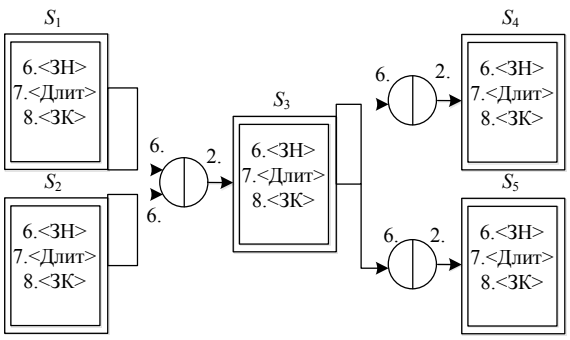

Рис. 10. Программа с некорректностью S_3 по входу и по выходу

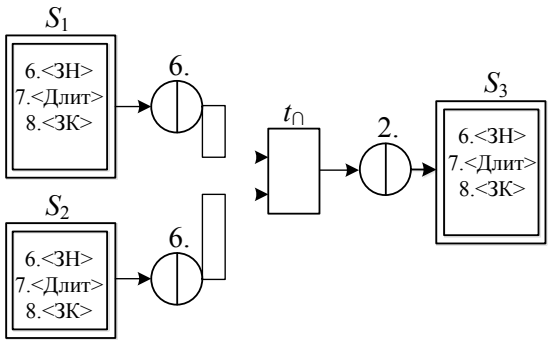

Рис. 11. Программа после компенсации некорректности S_3 по входу

Ошибка, заключающаяся в противоречивости технических ограничений, может быть допущена при задании для одного экземпляра УМТО более одной *z*-модели. Например, для двух технических ограничений с такими областями применимости, что $D(q_1^{(k)}(x)) \cap \cap D(q_2^{(k)}(x)) = \varnothing$, изменение входных переменных операции S_k будет невозможно. Компенса-

ция противоречивости технических ограничений заключается во введении в программу дополнительного предикатного оператора с областью применимости, равной объединению областей $D(q_1^{(k)}(x))$ и $D(q_2^{(k)}(x))$.

Компенсация пассивности экземпляра УМТО заключается в его удалении из схемы и установки равенства 1 входным переменным экземпляров, зависимым от удаленного по каким-либо типам технологических ограничений.

Подэтап 11 (см. рис. 1) заключается в корректировке программы с целью уменьшения количества смен значений переменных в экземплярах УМТО в единицу времени [10]. Данная задача является актуальной при мониторинге сложных процессов функционирования КСр с использованием измерительной информации, поступающей со значительной частотой, что является характерным для бортовых систем ракет-носителей.

На этапе III (см. рис. 1) проводится интерпретация исполнения программы в соответствии с положениями теории мониторинга и управления состоянием ТС [11], а также оценивание синтезированной программы по системе показателей качества [12]. Оканчивается этап III формированием информационной технологии практического применения функционально-логической программы потоковых вычислений [12].

Практическая проверка [12] представленного в статье метода показала значительное повышение оперативности получения результатов мониторинга двигательной установки ракеты-носителя «Союз-2» при сохранении выполнения заданных требований по достоверности и полноте указанных результатов.

ЗАКЛЮЧЕНИЕ

В статье рассмотрены постановка и решение задачи синтеза функционально-логических программ потоковых вычислений при мониторинге состояния КСр. Предложенный метод синтеза программ вычислений позволяет создать «язык» верхнего уровня для описания процессов обработки и анализа измерительной, технической и технологической информации, характеризующей ТС КСр.

Представленный метод отличается:

– реализацией подхода функционального программирования при синтезе программы потоковых вычислений на основе вычислительных сетей;

– разработкой процедуры синтаксической верификации синтезированной программы на основе проверки свойства ее полной (тотальной) правильности;

– формализацией процедур поиска и устранения ошибок (семантической верификацией) на основе интерпретации и проверки корректности свойств сетей Петри.

Практическая значимость и актуальность предложенного метода заключается в учете при синтезе программ вычислений факторов, искажающих процесс мониторинга состояния ТС.

Список используемых источников

1. Риски космических проектов. Анализ причин неудачных космических запусков / Г.П. Беляков и др. // Вестник СибГАУ. – 2014. – № 5 (57). – С. 208–215.

2. *Колмагоров А.Н.* К определению алгоритма // Успехи математических наук. – 1958. – Т. XIII, Вып. 4 (82). – С. 3–28.

3. *Усталов Д.А.* Коллективные потоковые вычисления: реляционные модели и алгоритмы // Моделирование и анализ информационных систем. – 2016. – Вып. 23 (2). –С. 195–210.

4. *Виноградов Р.А.*, *Кузьмин Е.В.*, *Соколов В.А.* Верификация автоматных программ средствами CPN/Tools // Моделирование и анализ информационных систем. – 2006 – Т. 13, № 2. – С. 4–15.

5. *Амбарцумян А.А.* Сетецентрическое управление на сетях Петри в структурированной дискретно-событийной системе // Управление большими системами. – 2017. – Спец. вып. 30.1 «Сетевые модели в управлении». – С. 506–535.

6. *Дьяченко Р.А.*, *Фишер А.В.*, *Богданов В.В.* Моделирование систем сбора и передачи данных с применением цветных сетей Петри // Fundamental Research. – 2013. – № 11. – P. 1122–1126.

7. *Westergaard M.*, *Kristensen L.M.* The Access/CPN Framework: A Tool for Interacting with the CPN Tools Simulator // Proc. of 30th International Conference on Applications and Theory of Petri Nets (Petri Nets 2009). Lecture Notes in Computer Science. – Berlin: Springer-Verlag, 2009. – № 5606. – P. 313–322.

8. *Jensen K.*, *Kristensen L.M.*, *Wells*. *L.* Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems // International Journal on Software Tools for Technology Transfer (STTT). – 2007. – № 9 (3–4). – P. 213–254.

9. *Шмелёв В.В.* Модели технологических процессов функционирования космических средств // Авиакосмическое приборостроение. – 2015. – № 4. – С. 78–93.

10. *Шмелёв В.В.*, *Мануйлов Ю.С.* Вычислительная ресурсоемкость сетевой модели обработки и анализа измерительной информации ракеты-носителя «Союз-2» // Информация и космос. – 2016. – № 1. – С. 155–161.

11. *Охтилев М.Ю.*, *Соколов Б.В.* Новые информационные технологии мониторинга и управления состояниями сложных технических объектов в реальном масштабе времени // Труды СПИИРАН. – 2005. –Т. 2, № 2. – С. 249–265.

12. *Шмелёв В.В.*, *Деев В.В.*, *Ткаченко В.В.* Метод организации потоковых вычислений при интеллектуальном мониторинге состояния технических систем // Наукоемкие технологии в космических исследованиях земли. – 2017. – Т. 9, № 6. – С. 24–35.