УДК 519.713: 67.02
DOI 10.21685/2072-3059-2019-2-3

В. Н. Дубинин, А. С. Войнов, И. В. Сенокосов, В. В. Вяткин

ФУНКЦИОНАЛЬНО-БЛОЧНАЯ РЕАЛИЗАЦИЯ МОДЕЛЕЙ ПЕРЕХОДОВ СОСТОЯНИЙ

Аннотация.

Актуальность и цели. В связи с широким внедрением в сфере промышленной автоматики методологий проектирования, основанных на моделях, возникает потребность трансформации моделей в исполнимый код для контроллеров. Наиболее перспективным языком программирования распределенных контроллеров в настоящее время является язык функциональных блоков (ФБ) стандарта МЭК 61499. Целью работы является разработка методов преобразования наиболее распространенных моделей переходов состояний, в числе которых конечные и магазинные автоматы, а также сети Петри, в системы ФБ.

Материалы и методы. При проведении исследования использовались положения и методы теории конечных автоматов и систем переходов, магазинных автоматов, формальных грамматик и сетей Петри, а также технологии разработки программного обеспечения на основе стандарта МЭК 61499.

Результаты. В ходе выполнения работы были получены следующие научные и практические результаты: 1) метод реализации недетерминированных конечных автоматов на основе ФБ МЭК 61499, особенностью которого является представление состояний автомата с помощью ФБ, использование механизма передачи маркеров и двухфазной схемы выполнения; 2) подход к реализации детерминированных магазинных автоматов на основе ФБ МЭК 61499, особенностью которого является использование в качестве исходной модели графового представления магазинного автомата и представление магазина в виде отдельного ФБ; 3) формальное определение селектирующих A-сетей и методика их реализации на основе ФБ, особенностью которой является представление позиций и переходов сетевой модели в виде отдельных ФБ, а также наличие специального диспетчера переходов.

Выводы. Предложенные реализационные методы могут быть использованы в проектировании промышленных киберфизических систем для мониторинга и диагностики, проверки соответствия, детектирования и выборки специфицированных последовательностей событий и параметризованных объектов из входного потока, а также для управления технологическим процессом.

Ключевые слова: недетерминированные конечные автоматы, магазинные автоматы, сети Петри, A-сети, детектирование, выборка, проверка соответствия, функциональный блок, стандарт МЭК 61499, NxtStudio.

V. N. Dubinin, A. S. Voynov, I. V. Senokosov, V. V. Vyatkin

FUNCION BLOCK-BASED IMPLEMENTATION OF STATE TRANSITION MODELS

Abstract.

Background. In connection with the widespread introduction in the field of industrial automation model-based design methodologies, there is a need to develop methods and tools for transforming the models to an executable code for controllers. The most promising programming language for distributed controllers is currently the language of function blocks (FB) of the IEC 61499 standard. The purpose of this work is to develop methods for transforming the most common state transition models, including non-deterministic finite and pushdown automata, as well as Petri nets to FB systems.

Materials and methods. This study used the statements and methods of the theory of finite automata and transition systems, pushdown automata, formal grammars and Petri nets, as well as software development methodology based on the IEC 61499 standard.

Results. In the course of the work the following scientific and practical results were obtained: 1) the implementation method of non-deterministic finite automata based on the IEC 61499 FB, the features of which are the representation of the automaton's states using FBs, the use of a token transfer mechanism and a two-phase execution scheme; 2) the approach to the implementation of deterministic pushdown automata based on the IEC 61499 FB, the peculiar properties of which are the use of the graph representation of pushdown automata as an initial model and representation of the stack in the form of a separate FB; 3) the formal definition of selective A-nets and the method of their implementation based on the IEC 61499 FB, the features of which are the presentation of places and transitions of the net model as separate FBs, as well as the presence of a special transition manager.

Conclusions. These implementation approaches can be used in the design of industrial cyber-physical systems for monitoring and diagnostics, conformance checking, detection and selection of specified sequences of events and parameterized objects from an input stream, as well as for the control of technological operations.

Keywords: non-deterministic finite automata, pushdown automata, Petri nets, A-nets, detection, selection, conformance checking**,** function block, IEC 61499 standard, NxtStudio.

Введение

В связи с широким внедрением в сфере разработки систем промышленной автоматики (ПА) методологий проектирования, основанных на моделях (например: *model-driven engineering*, *model integrated computing*, *model centered design* [1]), возникает потребность в разработке методов и средств трансформации моделей в исполнимый код программ для контроллеров. Использование формальных моделей позволит избежать двусмысленности и неопределенности, произвести верификацию и оценку производительности, что в итоге повысит качество и надежность проекта и приведет к снижению издержек проектирования. В сфере промышленной автоматики формальные модели могут использоваться в проектировании собственно управляющих алгоритмов, систем мониторинга и диагностики, систем супервизорного управления, проверки соответствия, систем обнаружения и выборки специфицированной последовательности событий и (параметризованных) объектов и т.д.

Задача детектирования и выборки специфицированной последовательности объектов из входного потока может, например, возникнуть в сборочном производстве [2]. В данном случае упорядоченность является важным свойством продукта. Кроме того, данная задача может быть адаптирована для

целей контроля правильности функционирования системы в том смысле, удовлетворяет ли поведение системы некоторому заданному шаблону. Это задача проверки соответствия (*conformance checking*), часто решаемая в рамках метода *Process mining* [3]. Детектирование специфицированных последовательностей может свидетельствовать о нормальном или, наоборот, ненормальном функционировании системы. Задача выборки специфицированной последовательности объектов является сходной с задачей распознавания языков, встречающейся, в частности, при построении трансляторов и компиляторов. В отличие от распознавателя, система выборки может явно игнорировать входные объекты, если они не инициируют в ней соответствующие изменения. Следует также отметить значительно более строгие требования по времени, предъявляемые к системам выборки, поскольку зачастую они работают в режиме реального времени. Задачи выборки не так многосторонне представлены в литературе, как задачи распознавания языков.

В ПА наиболее распространенными формальными моделями являются так называемые «модели переходов состояний» (МПС), в которых функционирование системы или процесса представляется как последовательность переходов из одного состояния в другое. Это определяется не в последнюю очередь требованиями надежности и безопасности, согласно которым в системе должны быть выделены явные состояния и их число должно быть, как правило, конечным [4]. Несмотря на общие методологические основы, данные модели довольно сильно различаются в конкретике. Примерами МПС являются конечные автоматы (КА) [4, 5], магазинные автоматы (МА) [5], сети Петри (СП), машины абстрактных состояний (МАС). Кроме «чистых» моделей, существуют их всевозможные расширения и модификации. Если в простейших моделях типа КА состояние может быть представлено одной переменной, то в сложных (например МАС) состояние представляется набором функций. Существуют значительные отличия и в сложности, и интерпретации переходов.

Основным трендом в организации систем ПА является переход от централизованных систем к распределенным. Наиболее перспективным языком программирования распределенных контроллеров в настоящее время является язык функциональных блоков стандарта МЭК 61499 [6]. Это специализированный компонентно-ориентированный язык для построения распределенных систем управления промышленными процессами.

В данной работе будут рассматриваться методы реализации моделей КА, МП и расширенных СП (А-сетей) на основе функциональных блоков (ФБ) стандарта МЭК 61499.

1. Реализация конечных автоматов

КА-модели являются наиболее популярными в промышленной автоматике. Существует большое разнообразие моделей данного класса – от классических конечных автоматов до диаграмм состояний UML [7], в которых, в частности, используется иерархическая структуризация состояний, составные переходы, исторические состояния и сторожевые условия переходов. Несмотря на это, основная идея ФБ-реализации КА-моделей остается примерно одной и той же.

Ниже в качестве примера рассматриваются подход к реализации недетерминированных конечных автоматов (НДКА).

Особенности предлагаемого подхода:

1) неявная детерминизация НДКА в режиме реального времени («на лету»). В этом случае состояния детерминированного КА получаются в процессе (параллельного) функционирования НДКА как комбинации его активных состояний. Существенным моментом при этом является синхронный характер функционирования НДКА;

2) двухфазная схема выполнения, при которой на первой фазе срабатывают разрешенные переходы в НДКА, а на второй производится публикация новых состояний автомата. Двухфазная схема была предложена в [8] для реализации НДКА на основе языков LD и FBD стандарта МЭК 61131-3;

3) механизм передачи маркеров для симуляции поведения НДКА.

В рамках предложенного подхода возможно провести его дальнейшую детализацию на основе следующих классификационных признаков:

1) семантика выполнения НДКА;

2) ориентация на переходы или состояния (в данной работе рассматривается второй подход, основы первого были рассмотрены в [2]);

3) способ обработки входных сигналов и организации второй фазы выполнения (последовательная и параллельная схемы).

Можно выделить две семантики выполнения НДКА, что касается условий сохранения (локальных) состояний НДКА:

1) состояние сохраняется, если в нем имеется петля, специфицирующая условие сохранения этого состояния [9];

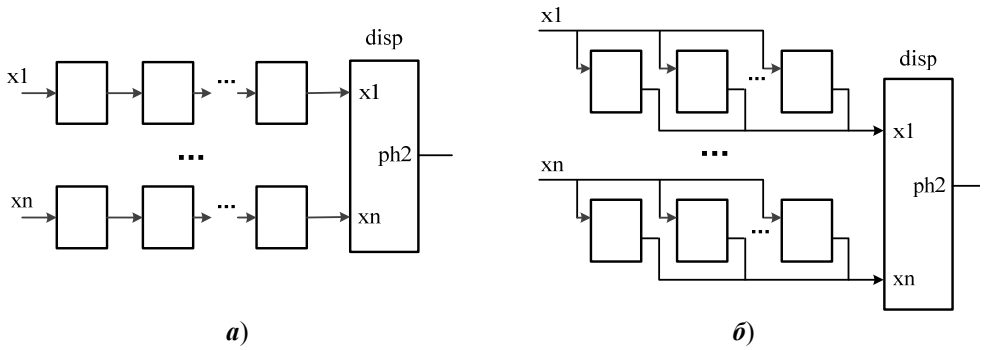
2) состояние сохраняется, если все исходящие из данного состояния переходы не являются разрешенными.

В семантике 1 явно специфицируется сохранение состояния, неявно – сброс, а в семантике 2 наоборот – явно специфицируется сброс состояния, а неявно – сохранение.

Для реализации третьего классификационного признака в случае, когда принята ориентация на состояния, разработаны соответствующие структурные шаблоны. Основными элементами шаблона являются ФБ, моделирующие состояния НДКА (называемые также ФБ-состояниями), и ФБ, представляющий диспетчер, который организует общий процесс вычислений. На рис. 1 представлены два шаблона для организации обработки входного сигнала на первой фазе выполнения – «последовательная схема» и «параллельная схема». В последовательной схеме блоки, обрабатывающие один и тот же сигнал, связаны в дейзи-цепочку. Входной сигнал x_i подается на первый блок цепочки. Выходной сигнал с последнего блока сигнализирует, что о том, что обработка сигнала x_i во всех релевантных ФБ закончена. Этот сигнал поступает в диспетчер *disp* и служит для него основанием начать вторую фазу выполнения генерацией выходного сигнала *ph2*. Один и тот же ФБ может быть задействован в разных цепочках, поскольку в нем может быть предусмотрена обработка нескольких входных сигналов.

В параллельной схеме входной сигнал x_i поступает сразу на все релевантные ФБ. Для синхронизации завершения выполнения группы ФБ используется механизм квитирования с подсчетом числа квитанций. При этом по завершении обработки входного сигнала x_i, которая заключается в возможном изменении переменной текущего состояния, каждый ФБ посылает соответствующий сигнал-квитанцию в диспетчер. Диспетчер должен собрать все

квитанции (их число определяется по числу релевантных ФБ по входному сигналу x_i), после чего инициирует вторую фазу выполнения. Преимуществом последовательной схемы является ее простота и применимость ко всем моделям выполнения ФБ вследствие своей детерминированности. Параллельная схема является более сложной и недетерминированной и применима далеко не для всех моделей выполнения ФБ. Преимуществом схемы является возможность параллельного выполнения ФБ.



а) ***б***)

Рис. 1. Последовательная (***а***) и параллельная (***б***) схемы обработки входных сигналов (на первой фазе)

Аналогично представляются шаблоны для организации второй фазы выполнения – «последовательная схема» и «параллельная схема». В последовательной схеме блоки, реализующие действия второй фазы, выполняются последовательно, один за другим.

Параллельная реализация НДКА, приведенного на рис. 2, в виде сети ФБ составного ФБ представлена на рис. 3. Следует заметить, что в данном случае принята семантика 2, а реализация производилась в инструментальной системе NxtStudio [10].

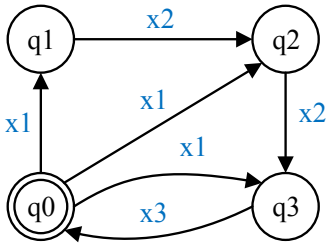


Рис. 2. Пример НДКА

На рис. 3 показаны ФБ-состояния (FB1-FB4) и ФБ-диспетчер (FB6). Через событийные входы $x1..x3$ поступает один из входных сигналов автомата. Первая фаза выполнения была подробно описана выше. На второй фаза все ФБ-состояния параллельно выдают свои внутренние состояния на выход (для других ФБ-состояний) и сообщают об этом диспетчеру. Когда диспетчер получил квитанции от всех ФБ-состояний, выдается сигнал окончания обработки входного сигнала *Out*.

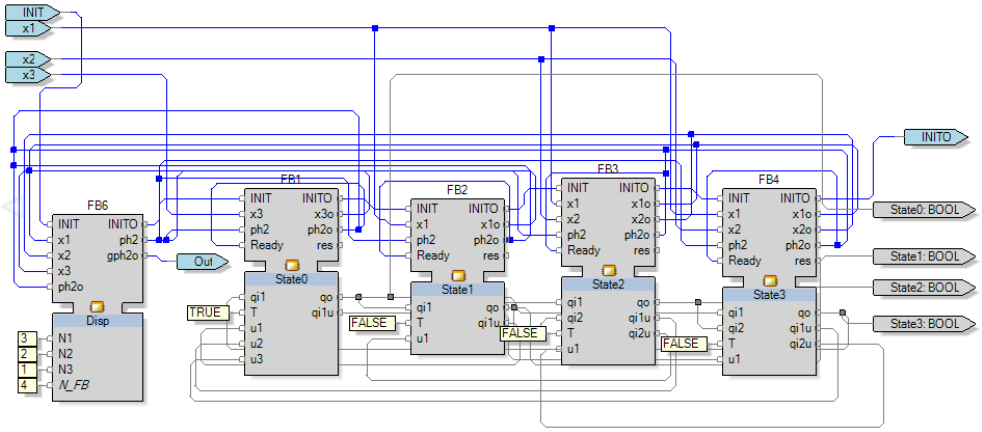


Рис. 3. Реализация НДКА в *NxtStudio*

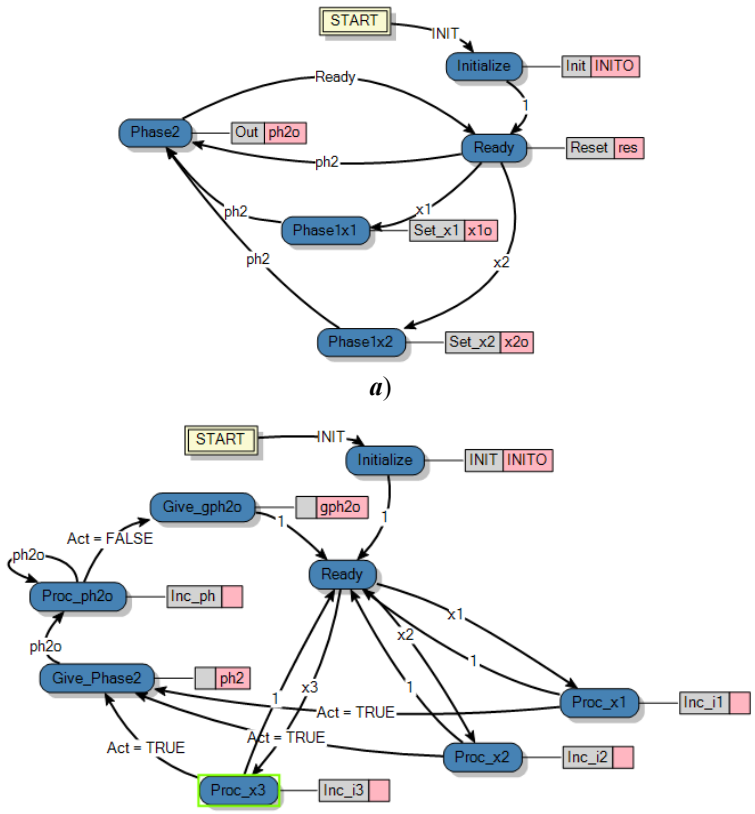
Рассмотрим более подробно базисный ФБ *FB3* типа *State2*, отвечающий за реализацию состояния $q2$. Интерфейс данного ФБ включает следующие входы-выходы. Событийные входы: *INIT* – инициализация ФБ; $x1$, $x2$ – входные сигналы НДКА; $ph2$ – запуск второй фазы выполнения; *Ready* – сигнал готовности ФБ принимать новый входной сигнал. Информационные входы: $qi1$ ($qi2$) – значение первого (второго) состояния-предшественника (истина/ложь); T – начальное значение состояния. Событийные выходы: *INITO* – инициализации ФБ окончена; $x1o$ ($x2o$) – обработка входного сигнала x_1 (x_2) окончена; $ph2o$ – вторая фаза выполнения окончена. Информационные выходы: qo – значение состояния НДКА $q2$ (истина/ложь). В ФБ также используется внутренняя переменная булевского типа q для хранения статуса состояния НДКА $q2$ (истина/ложь). На рис. 4 представлены диаграммы *ECC* для базисного ФБ State2 (***а***) и ФБ-диспетчера (***б***).

Оценим время реакции на входной сигнал для ФБ-базированной реализации НДКА. Обозначим $X = \{x_1, x_2, ..., x_n\}$ – множество входных сигналов, подаваемых на НДКА; N_S – общее число состояний НДКА; N_{x_i} – число состояний НДКА, из которых выходят дуги переходов, помеченные сигналом x_i. Тогда время первой фазы обработки входного сигнала x_i будет вычисляться как $T^I = N_{x_i}(t_1 + t_2)$, где t_1 – время обработки входного сигнала в ФБ-состоянии и выдачи соответствующей квитанции; t_2 – время обработки одной квитанции первой фазы в ФБ-диспетчере.

Время второй фазы является постоянным, независимо от входного сигнала, и определяется следующим образом: $T^{\text{II}} = N_S(t_3 + t_4) + t_5$, где t_3 – время реакции ФБ на прием сигнала начала второй фазы; t_4 – время обработки одной квитанции второй фазы в ФБ-диспетчере; t_5 – время выдачи сигнала завершения обработки входного сигнала.

Следует заметить, что возможна дальнейшая детализация формул, вплоть до времен перехода между состояниями *ECC* и выполнения алгоритмов в состояниях. Реальные измерения на системе ФБ из рис. 3, выполняемой на платформе ПК (Intel Core i5-2310 2.90GHz, ОС Windows 7), показали, что время реакции на входной сигнал лежит в пределах от 26 до 33 мкс. Это вре-

мя от момента появления входного сигнала x_i до выдачи выходного сигнала *Out* диспетчера.



а)

б)

Рис. 4. Диаграммы *ECC* для базисного ФБ State2 (***а***) и ФБ-диспетчера (***б***)

2. Реализация магазинных автоматов

Конечные автоматы являются достаточно простыми моделями и не могут описать сложные процессы, например представляемые контекстно-свободными (КС) языками. Для распознавания КС-языков используются магазинные автоматы. Формальное определение МА можно найти, например, в [5]. В отличие от конечных автоматов, в МА используется магазинная память (стек, магазин).

Ниже предлагается методика реализации детерминированных МА на основе ФБ стандарта МЭК 61499. В качестве исходного описания в методике рекомендуется использовать графовое представление МА, предложенное в работе [5]. При этом переход между состояниями автомата имеет пометку, состоящую из трех компонент: A – входной символ; B – символ вершины стека; C – символ (или последовательность символов), на который будет заменяться символ вершины стека (рис. 5). Следует заметить, что сам стек в данном представлении не фигурирует.

Краткие положения методики преобразования МА в ФБ-базированную реализацию следующие:

1) каждому переходу МА ставится в соответствие ФБ. В этом случае практически все переходы моделируются ФБ одного и того же типа, отличающимися только некоторыми параметрами при инициализации. Альтернативным подходом является подход, когда в виде ФБ представляются состояния. В этом случае каждый ФБ будет отличаться числом входных/выходных сигналов и переменных, и сложность реализации каждого отдельного блока-состояния будет довольно большой;

2) для пометки переходов, разрешенных со стороны состояний, используются маркеры. Маркер является динамическим объектом, который может быть передан из одного ФБ-перехода в другой ФБ-переход. Если ФБ имеет маркер, то он может принимать входные символы;

3) для представления стека используется отдельный ФБ, реализующий операции вталкивания и выталкивания значения в стек, а также сравнения вершины стека с заданным значением;

4) состояния МА в системе явно не представлены.

Рис. 5. Переход в магазинном автомате

На рис. 6 приведен пример фрагмента системы ФБ, задействованного в реализации некоторого перехода МА.

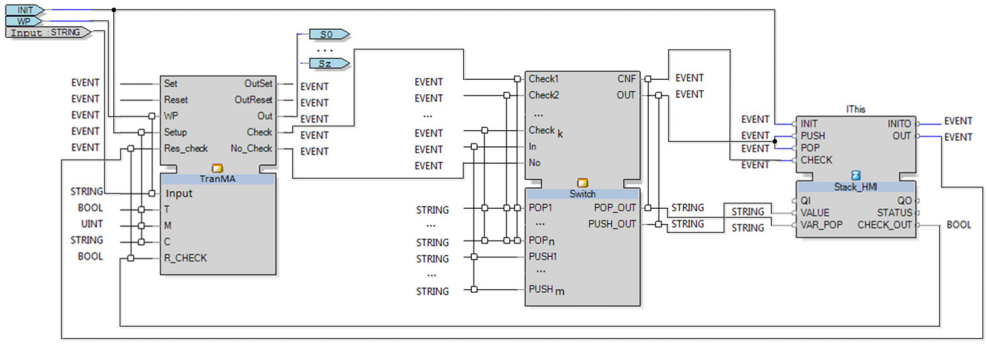


Рис. 6. Фрагмент системы ФБ для реализации перехода МА

Первый из блоков типа *TranMA* реализует собственно переход автомата. ФБ *Switch* управляет потоками данных от ФБ-переходов в ФБ стека (*Stack_HMI*).

Интерфейс базисного ФБ *TranMA* состоит из следующих групп параметров: *Set*/*Reset* – входные сигналы установки/удаления маркера в данном ФБ; *WP* – входной сигнал поступления входного символа; *Setup* – входной сигнал установки (сброса) начальных параметров ФБ; *Res_Check* – входной сигнал, информирующий об окончании проверки верхнего символа стека с магазинным символом ФБ-перехода; *OutSet*/*OutReset* – выходной сигнал, устанавливающий/сбрасывающий маркеры в других ФБ; *Out* – выходной сиг-

нал, означающий, что ФБ-переход сработал; *Check* – управляющий выходной сигнал для сравнения верхнего символа стека магазинным символом ФБ-перехода; *No_Check* – сигнал, который выдается, если верхний символ стека не совпал с магазинным символом ФБ-перехода; *Input* – входной символ; *C* – входной символ, который имеет право принять данный ФБ; *T* – флаг наличия маркера в ФБ; *R_CHECK* – результат сравнения верхнего символа магазина (истина/ложь).

Предложенная методика может быть использована в отношении предметной области работы [11], в которой предлагается использование модели селектирующего магазинного автомата для спецификации и выборки последовательностей деталей в промышленных системах сортировки, а также использование магазинного автомата-преобразователя в проектировании технологического процесса сборки изделий, управляемого потоком деталей и жетонов.

3. Реализация сетей Петри

По сравнению с автоматными моделями сетевые модели на основе сетей Петри являются более мощными. Практически отсутствуют работы по использованию СП для распознавания языков, что можно объяснить большой вычислительной сложностью процедур их анализа. Для снижения вычислительной сложности используются упрощения СП. Например, в работе [12] предлагаются структурно простые СП (S-сети), а для распознавания используется алгоритм на основе конечных автоматов, имеющий полиномиальную сложность. Если рассматривать смежные области, связанные с использованием СП для распознавания ситуаций или паттернов, то можно выделить работы [13–16]. В работе [13] СП используется для конструирования запроса к данным видеонаблюдения (в терминах событий) и далее для распознавания этого специфицированного шаблона. Авторы работы [14] предлагают раскрашенные СП для моделирования распознавания упорядоченных во времени последовательностей событий, выраженных с помощью логических и временных операторов, а также минимальных и максимальных времен задержки. В работе [15] раскрашенные СП используются для определения правильности выполнения оператором (например, пилотом) соответствующего руководства. В работе [16] для реализации проверки соответствия (*conformance checking*) авторы предлагают сети Петри с данными, позволяющими моделировать переменные, сторожевые условия и действия по вводу-выводу.

В данной работе рассматривается использование расширенных СП для детектирования и выборки параметризованных объектов вида $<a_1,a_2,\ldots,a_n>$ из входного потока. В качестве основы математического аппарата выбраны A-сети, первоначально использованные в [17] для асинхронного моделирования NCES-сетей. A-сети являются расширением СП в направлении повышения моделирующих возможностей за счет нагрузки дуг и усложнения правил разрешенности и срабатывания переходов, что позволяет эффективно представлять обработку целочисленных переменных. Кроме того, A-сети могут быть легко расширены управляющими символами и даже действиями, прикрепленными к переходам и/или позициям, что переводит их в класс сетей-преобразователей.

Селектирующая А-сеть определяется следующим кортежем:

$$(P, T, X, Y, Z, U, W_X, W_Y, W_Z, Q, A, G, m_0, T_F),$$

где P – конечное множество позиций; T – конечное множество переходов; $X \subseteq P \times T$ – множество входных дуг переходов с порогом-минимумом (дуги с проверкой на «больше»); $Y \subseteq T \times P$ – множество выходных дуг переходов; $Z \subseteq P \times T$ – множество входных дуг переходов с порогом-максимумом (дуги с проверкой на «меньше»); $U \subseteq X \times Y$ – отношение сопряженности дуг; W_X: $X \to N_0 \times \{N_0 \cup all\}$ – функция весов дуг типа X; W_Y: $Y \to \{N^+ \cup *\} \times$ $\times \{add,reset\}$ – функция весов дуг типа Y; W_Z: $Z \to N^+ \times \{N_0 \cup all\}$ – функция весов дуг типа Z; Q: $T \to N_0$ – функция приоритетов переходов. Должно выполняться ограничение $\forall t_i, t_j \in T\,[Q(t_i) \neq Q(t_j)]$. Это ограничение делает функционирование селектирующей СП *детерминированным*; A – множество параметров (атрибутов) объекта. Обозначим $\overline{A} = \bigcup_{a \in A} (a \times D^a)$ – пространство состояний параметров, D^a – домен параметра a; G – множество сторожевых функций переходов. Каждая функция $g^t \in G$ определяется как $g^t : [\overline{A}] \to \{true, false\}$, где $[\overline{A}] = \prod_{a \in A} (\{a\} \times D^a)$ – множество кортежей, которые содержат все комбинации значений переменных, тегированных именами переменных; m_0 – начальная маркировка; $T_F \subseteq T$ – множество финальных переходов. Срабатывание любого финального перехода свидетельствует об окончании выборки последовательности объектов.

Входной объект будет представляться следующим кортежем: $\overline{a} = ((a_1, v^{a_1}), (a_2, v^{a_2}), ..., (a_n, v^{a_n}))$. Переход $t \in T$ разрешен при маркировке m и входном объекте $\overline{a}$, если

$$g^t(\overline{a}) \wedge \forall p \in P[(p,t) \in X \to m(p) \geq$$

$$\geq pr_1(W_X(p,t)) \wedge (p,t) \in Z \to m(p) < pr_1(W_Z(p,t))].$$

Иными словами, переход разрешен, если истинно сторожевое условие и имеется концессия по меткам. Правила срабатывания переходов представлены в работе [17]. Следует отметить, что сочетания весов дуг образуют некоторые паттерны дуг, имеющих собственную семантику – числовую дугу, безусловную и пороговую дугу сброса, тестирующую дугу с проверкой на «больше или равно», ингибиторную дугу (дугу с проверкой на «меньше»), сопряженные дуги для безусловного копирования/добавления/передвижения меток.

Для интеграции и встраивания селектирующих А-сетей в системы управления на основе стандарта IEC 61499 ниже предлагается методика их преобразования в системы ФБ. Основная идея методики заключается в реализации элементов А-сетей – позиций и переходов в виде отдельных ФБ. ФБ-позиция хранит маркировку позиции, а также выполняет над ней элементарные операции сложения/вычитания и сброса в ноль. ФБ-переход определяет разрешенность перехода А-сети и задает изменение маркировки сосед-

них позиций при его срабатывании. Срабатыванием переходов в сетевой модели управляет ФБ-диспетчер.

Правило преобразования перехода селектирующей А-сети в ФБ приведено на рис. 7.

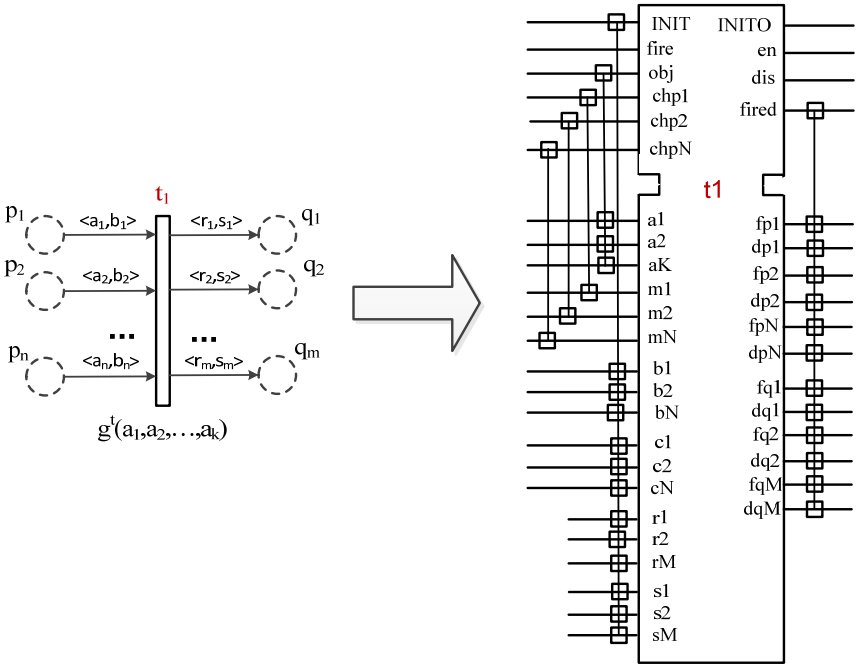


Рис. 7. Преобразование перехода селектирующей А-сети в ФБ

Описание интерфейса этого ФБ следующее. *Событийные входы*: *fire* – срабатывание разрешенного перехода; *obj* – приход нового объекта; chp_i – изменение маркировки позиции p_i. *Событийные выходы*: *en* – переход стал разрешенным; *dis* – переход деактивирован; *fired* – переход сработал. *Информационные входы*: a_k – *k*-й параметр входного объекта; m_i – (новая) маркировка позиции p_i; b_i (c_i) – первый (второй) компонент веса входной дуги из позиции p_i; r_i (s_i) – первый (второй) компонент веса выходной дуги в позицию q_i.

Примечание: входы типов *b*, *c*, *r*, *s* являются необязательными, поскольку они могут быть жестко заложены внутри ФБ (в алгоритмах обработки). Однако выведение их «наружу» позволяет строить самомодифицируемые системы (путем модификации этих параметров).

Информационные выходы: fp_i – модификатор действия над маркировкой позиции p_i; fq_i – модификатор действия над маркировкой позиции q_i; dp_i – число удаляемых меток из позиции p_i; dq_i – число добавляемых или назначаемых меток в позицию q_i. Значения модификатора действия над маркировкой: 0 – указанное число меток назначается в выходную позицию, причем она предварительно обнуляется; 1 – указанное число меток добавляется в выходную позицию в дополнение к существующим; 2 – указанное число меток удаляется из входной позиции.

В диспетчере хранится текущий список разрешенных переходов. При поступлении сигнала с входа en_i этот список пополняется переходом t_i.

При поступлении сигнала с входа dis_i из этого списка удаляется переход t_i. При приходе сигнала на вход типа *exe* диспетчер выбирает наиболее приоритетный разрешенный переход для срабатывания (активации), выдает на него сигнал запуска.

ФБ, моделирующий позицию, представлен на рис. 8.

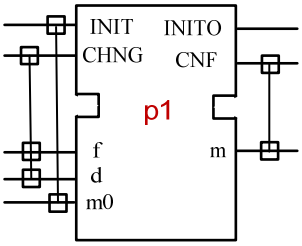


Рис. 8. ФБ-позиция

ФБ-позиция имеет следующий интерфейс. *Событийные входы*: *CHNG* – изменить маркировку позиции. *Событийные выходы*: *CNF* – маркировка позиции изменена. *Информационные входы*: *m0* – начальная маркировка позиции; *f* – модификатор действия над маркировкой позиции (установить, добавить, удалить); *d* – аргумент модификатора (сколько установить, добавить, удалить). *Информационный выход*: *m* – новое значение маркировки позиции.

Система на основе селектирующих А-сетей обрабатывает приход очередного объекта следующим образом. Заново вычисляются сторожевые условия в ФБ-переходах, где имеется концессия по меткам, статус этих переходов также обновляется. В диспетчер посылается сигнал на попытку запуска нового перехода. Следует заметить, что срабатывание какого-либо перехода может вызвать срабатывание цепочки других переходов. Для повышения быстродействия алгоритма интерпретации А-сети используется принцип локальности изменения состояний позиций и переходов. Новый переход для срабатывания может быть выбран диспетчером только после того, как закончится перевычисление: 1) маркировок входных и выходных позиций текущего сработавшего перехода; 2) статусов (разрешен / не разрешен) инцидентных им переходов.

Ниже представлен демонстрационный пример использования селектирующих А-сетей для построения линейки LEGO-конструкций (в виде «строения с прилавком внутри»). Экземпляр подобной конструкции изображен на рис. 9.

При построении LEGO-конструкций используются элементы (объекты) со следующими параметрами: 1) форма (значения I – «панель», II – «стена», III – «прямоугольный блок»); 2) цвет (значения «Green», «Yellow», «White», «Brown», «Orange», «Azure»). Предполагается, что объекты движутся по одному или нескольким конвейерам и сборочный робот, оснащенный соответствующими датчиками, производит выборку нужных объектов из потока.

Порядок выборки объектов определяется селектирующей А-сетью, представленной на рис. 10.

На рис. 10 сторожевые условия представлены в закодированном виде, с использованием значений параметров объектов LEGO-конструкции (например, IG обозначает объект типа I («панель») цвета G («зеленый»), в итоге по-

лучается сторожевое условие «ожидается зеленая панель»). В соответствии с приведенной сетевой моделью сначала последовательно выбирается «панель» зеленого или белого цвета. Далее параллельно выбираются три «стены» и два «прямоугольных блока», причем две последние «стены» должны быть одного цвета, а «прямоугольные блоки» – разного. Когда собраны все стены, может быть выбрана «зеленая панель» (в качестве «крыши»). После выборки всех требуемых объектов происходит выход из цикла, иначе цикл выборки начинается сначала. В соответствии с методикой, представленной выше, была произведена ФБ-реализация А-сети, представленной на рис. 10. Из-за ее громоздкости она в статье не приводится.

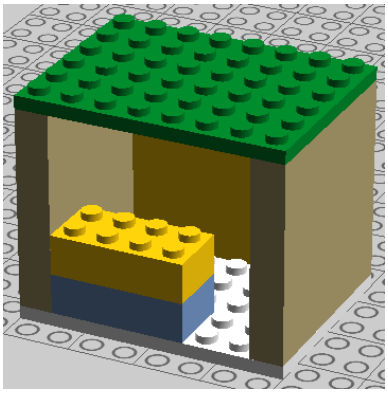


Рис. 9. Экземпляр линейки LEGO-конструкций

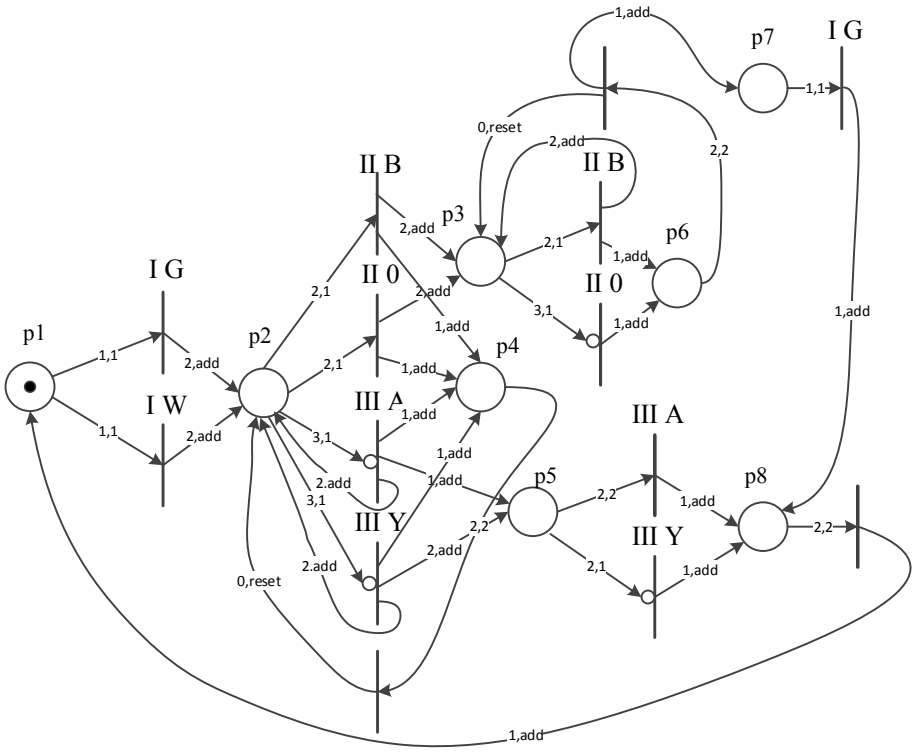


Рис. 10. Селектирующая А-сеть для построения линейки LEGO-конструкций

Заключение

В работе представлены подходы к реализации наиболее популярных в промышленной автоматике моделей переходов состояний в виде ФБ стандарта МЭК 61499. В плане определения области применения ориентация была сделана на задачи детектирования и выборки. Направлением дальнейших исследований является функционально-блочная реализация более сложных моделей переходов состояний, например, раскрашенных сетей Петри и машин абстрактных состояний, а также расширение сферы их использования в различных областях промышленной автоматики.

Библиографический список

1. **Дубинин, В. Н.** Модельно-центрированная методология проектирования распределенных компонентно-базированных информационно-управляющих систем промышленной автоматики / В. Н. Дубинин // Современные информационные технологии : сб. тр. Междунар. науч.-техн. конф. – Пенза, 2013. – Вып. 18. – С. 7–24.
2. **Dubinin, V.** Auto-Generation of Distributed Automation Software Based on Formal Product Line Specification / V. Dubinin, I. Senokosov, V. Vyatkin // Lecture Notes in Artificial Intelligence. – Cham : Springer, 2017. – Vol. 10444. – P. 80–91.
3. **Aalst, W. M. P.** Process Mining: Data Science in Action / W. M. P Aalst. – Springer Verlag, 2016. – 467 p.
4. **Поликарпова, Н. И.** Автоматное программирование / Н. И. Поликарпова, А. А. Шалыто. – Санкт-Петербург : Питер, 2009. – 176 с.
5. **Хопкрофт, Дж.** Введение в теорию автоматов, языков и вычислений / Дж. Хопкрофт, Р. Мотвани, Дж. Ульман. – Москва : Вильямс, 2002. – 528 с.
6. **Vyatkin, V.** IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design, Third Edition / V. Vyatkin. – Instrumentation Society of America (ISA), 2016. – 261 p.
7. **Drusinsky, D.** Modelling and verification using UML statecharts / D. Drusinsky. – Elsevier, 2006. – 400 p.
8. **Дубинин, В. Н.** Проектирование и реализация систем управления дискретными событийными системами на основе иерархических модульных недетерминированных автоматов (Ч. 2. Методы и средства) / В. Н. Дубинин, Д. А. Будаговский, Д. Н. Дроздов, Д. В. Артамонов // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2016. – № 2 (38). – С. 18–32.
9. **Вашкевич, Н. П.** Недетерминированные автоматы и их использование для реализации систем параллельной обработки информации : монография / Н. П. Вашкевич, Р. А. Бикташев. – Пенза : Изд-во ПГУ, 2016. – 394 с.
10. Сайт nxtStudio (nxtControl). – URL: http://www.nxtcontrol.com/
11. **Дубинин, В. Н.** Использование моделей магазинных автоматов в проектировании технологических процессов сортировки и сборки изделий / В. Н. Дубинин, И. В. Сенокосов, Л. П. Климкина, В. В. Вяткин // Современные технологии в науке и образовании (СТНО-2017) : сб. тр. Междунар. науч.-техн. и науч.-методич. конф. – Рязань, 2017. – Т. 1. – С. 10–14.
12. **Cui, T.** Recognition Algorithm Design and Complex Analysis for Languages of S-Nets / T. Cui, Q. Zeng, D. Zhang // Information Technology Journal. – 2011. – № 10. – P. 106–112.
13. **Ghanem, N.** Representation and Recognition of Events in Surveillance Video Using Petri Nets / N. Ghanem, D. DeMenthon, D. Doermann, L. Davis // Conference on Computer Vision and Pattern Recognition (CVPRW '04). – 2004. – July.

14. **Choppy, C.** Coloured Petri Nets for Chronicle Recognition / C. Choppy, O. Bertrand, P. Carle // Int. Conf. on Reliable Software Technologies – Ada-Europe 2009. Lecture Notes in Computer Science. – Springer, 2009. – Vol. 5570. – P. 266–281.
15. **Fernández, V. R.** Automatic Procedure Following Evaluation Using Petri Net-based Workflows / V. R. Fernández, A. G. Pardo, D. Camacho // IEEE Transactions on Industrial Informatics. – 2018. – Vol. 14, iss. 6. – P. 2748–2759.
16. **Leoni, M.** Decomposing Alignment-Based Conformance Checking of Data-Aware Process Models / M. Leoni, J. Munoz-Gama, J. Carmona, W. M. P. Aalst // Lecture Notes in Computer Science. – 2014. – Vol. 8841. – P. 3–20.
17. **Дубинин, В. Н.** Асинхронное моделирование NCES-сетей / В. Н. Дубинин // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2009. – № 2. – С. 3–14.

References

1. Dubinin V. N. *Sovremennye informatsionnye tekhnologii: sb. tr. Mezhdunar. nauch.-tekhn. konf.* [Modern information technologies: proceedings of an International scientific and technical conference]. Penza, 2013, iss. 18, pp. 7–24. [In Russian]
2. Dubinin V., Senokosov I., Vyatkin V. *Lecture Notes in Artificial Intelligence*. Cham: Springer, 2017, vol. 10444, pp. 80–91.
3. Aalst W. M. P. *Process Mining: Data Science in Action*. Springer Verlag, 2016, 467 p.
4. Polikarpova N. I., Shalyto A. A. *Avtomatnoe programmirovanie* [Automaton programming]. Saint-Petersburg: Piter, 2009, 176 p. [In Russian]
5. Khopkroft Dzh., Motvani R., Ul'man Dzh. *Vvedenie v teoriyu avtomatov, yazykov i vychisleniy* [Introduction into the theory of automata, languages and calculations]. Moscow: Vil'yams, 2002, 528 p. [In Russian]
6. Vyatkin V. *IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design, Third Edition*. Instrumentation Society of America (ISA), 2016, 261 p.
7. Drusinsky D. *Modelling and verification using UML statecharts*. Elsevier, 2006, 400 p.
8. Dubinin V. N., Budagovskiy D. A., Drozdov D. N., Artamonov D. V. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki* [University proceedings. Volga region. Engineering sciences]. 2016, no. 2 (38), pp. 18–32. [In Russian]
9. Vashkevich N. P., Biktashev R. A. *Nedeterminirovannye avtomaty i ikh ispol'zovanie dlya realizatsii sistem parallel'noy obrabotki informatsii: monografiya* [Nondeterministic automata and their use for implementing parallel data processing systems: monograph]. Penza: Izd-vo PGU, 2016, 394 p. [In Russian]
10. *Sayt nxtStudio (nxtControl)*. Available at: http://www.nxtcontrol.com/
11. Dubinin V. N., Senokosov I. V., Klimkina L. P., Vyatkin V. V. *Sovremennye tekhnologii v nauke i obrazovanii (STNO-2017): sb. tr. Mezhdunar. nauch.-tekhn. i nauch.-metodich. konf.* [Modern technologies in science and education: proceedings of an International scientific, technical and methodological conference]. Ryazan, 2017, vol. 1, pp. 10–14. [In Russian]
12. Cui T., Zeng Q., Zhang D. *Information Technology Journal*. 2011, no. 10, pp. 106–112.
13. Ghanem N., DeMenthon D., Doermann D., Davis L. *Conference on Computer Vision and Pattern Recognition (CVPRW '04)*, 2004, July.
14. Choppy C., Bertrand O., Carle P. *Int. Conf. on Reliable Software Technologies – Ada-Europe 2009. Lecture Notes in Computer Science*. Springer, 2009, vol. 5570, pp. 266–281.
15. Fernández V. R., Pardo A. G., Camacho D. *IEEE Transactions on Industrial Informatics*. 2018, vol. 14, iss. 6, pp. 2748–2759.
16. Leoni M., Munoz-Gama J., Carmona J., Aalst W. M. P. *Lecture Notes in Computer Science*. 2014, vol. 8841, pp. 3–20.

17. Dubinin V. N. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki* [University proceedings. Volga region. Engineering sciences]. 2009, no. 2, pp. 3–14. [In Russian]

Дубинин Виктор Николаевич
доктор технических наук, профессор, кафедра вычислительной техники, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: dubinin.victor@gmail.com

Dubinin Victor Nikolaevich
Doctor of engineering sciences, professor, sub-department of computer engineering, Penza State University (40 Krasnaya street, Penza, Russia)

Войнов Артем Сергеевич
аспирант, кафедра вычислительной техники, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: voj49@yandex.ru

Voinov Artem Sergeevich
Postgraduate student, sub-department of computer engineering, Penza State University (40 Krasnaya street, Penza, Russia)

Сенокосов Илья Владимирович
инженер, Научно-техническое предприятие «Криптософт» (Россия, г. Пенза, ул. Лермонтова, 3)

E-mail: senokosov.i@yandex.ru

Senokosov Ilya Vladimirovich
Engineer, “Cryptosoft” research and engineering enterprise (3 Lermontova street, Penza, Russia)

Вяткин Валерий Владимирович
доктор технических наук, профессор, кафедра ответственных коммуникаций и вычислений, Технический университет Лулео (Швеция, г. Лулео, ул. Регнбогсаллен, корп. А)

E-mail: valeriy.vyatkin@ltu.se

Vyatkin Valeriy Vladimirovich
Doctor of engineering sciences, professor, sub-department communication and computation systems, Lulea University of Technology (building A, Regnbagallen street, Lulea, Sweden)

Образец цитирования:

Дубинин, В. Н. Функционально-блочная реализация моделей переходов состояний / В. Н. Дубинин, А. С. Войнов, И. В. Сенокосов, В. В. Вяткин // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2019. – № 2 (50). – С. 23–38. – DOI 10.21685/2072-3059-2019-2-3.