УДК 681.5
DOI 10.21685/2072-3059-2019-1-4

Д. Н. Дроздов, В. Н. Дубинин, В. В. Вяткин

ФОРМАЛЬНАЯ СЕМАНТИКА ФУНКЦИОНАЛЬНЫХ БЛОКОВ МЭК 61499 С ВРЕМЕННЫМИ МЕТКАМИ

Аннотация.

Актуальность и цели. В настоящее время при проектировании промышленных киберфизических систем большое значение придается разработке надежных компонентно-базированных распределенных систем управления с жесткими требованиями по быстродействию. К этому «идеалу» можно приблизиться путем совместного использования международного стандарта МЭК 61499 и механизма учета временных характеристик системы в процессе ее функционирования. Целью работы является разработка формальной модели функциональных блоков стандарта МЭК 61499, расширенных временными атрибутами, для возможности ее использования на всех этапах проектирования систем управления промышленными киберфизическими системами.

Метериалы и методы. Исследования выполнены с использованием теории множеств и машин абстрактных состояний.

Результаты. Определена операционная семантика систем функциональных блоков МЭК 61499 с временными метками, отличающаяся использованием концепции развернутых одноуровневых систем, уникальными приоритетами элементов выполнения, унифицированным и независимым поведением интерфейсов функциональных блоков.

Выводы. Разработанная семантика систем функциональных блоков МЭК 61499 является формальной моделью, позволяющей производить корректную реализацию систем управления, учитывающих время, существенно упростить построение моделей для верификации и симуляции, обеспечить справедливость модели и полный детерминизм выполнения системы функциональных блоков на уровне отдельного ресурса.

Ключевые слова: формальная семантика, функциональный блок, МЭК 61499, машины абстрактных состояний, детерминизм, временные метки.

D. N. Drozdov, V. N. Dubinin, V. V. Vyatkin

FORMAL SEMANTICS OF IEC 61499 FUNCTIONAL BLOCKS WITH TEMPORARY TAGS

Abstract.

Background. Today, when designing industrial cyber-physical systems (ICPS), a great importance is devoted to development of reliable component-based distributed control systems with strict performance requirements. This “ideal” can be approached by common usage of the IEC 61499 international standard and mechanisms for accounting temporal characteristics of the system during its runtime (time-aware computations). The goal of this work is to develop a formal model of IEC 61499 function block systems, extended with temporal attributes, to enable its use in all stages of the ICPS control systems' design.

Materials and methods. The study has been performed using set theory and abstract state machines.

Results. The operational semantics of the IEC 61499 function block systems with timestamps is defined, whose distinct features are formality, usage of unfolded flat FB systems, unique priorities of the execution elements and unified and independent behaviour of the FB interfaces.

Conclusions. The developed formal semantics of IEC 61499 FB systems is a useful model that allows a correct implementation of time-aware systems, significantly simplify the construction of models for verification and simulation, ensure the fairness of the model and completely deterministic execution of FB system at the level of an individual resource.

Keywords: formal semantics, function block, IEC 61499, abstract state machines, determinism, timestamp.

Введение

В настоящее время в области промышленной автоматики происходит изменение парадигмы организации систем управления с жестко централизованных на распределенные, в которых вместо одного центрального контроллера существует множество «умных» конечных устройств, связанных друг с другом по беспроводным каналам связи и способных, в то же время, независимо исполнять определенную часть общей программы. Общепризнано, что ключевую роль в разработке программного обеспечения для таких систем управления играет международный стандарт МЭК 61499 [1]. Данный стандарт определяет общую архитектуру и графический компонентно-ориентированный язык программирования контроллеров на основе диаграмм функциональных блоков (ФБ), однако не определяет деталей семантики их выполнения, которые гарантировали бы независимость исполняемых программ от специфики платформы, детерминизм системы в целом и ее устойчивость к непредсказуемым переменам в окружающей среде, меняющим свойства вычислительных и коммуникационных элементов систем.

В работе [2] предлагается модель *pTIDES*, обеспечивающая детерминизм выполнения управляющей программы при условии, что существует известная и достаточно малая верхняя граница задержки передачи данных по сети. Однако данная модель не идеальна и может в ряде случаев привести к значительному снижению производительности системы управления. Механизм временных меток событий, использованный в *pTIDES*, является одним из способов увеличения временной информированности программных приложений, что получило развитие в принципе *Time-Aware Applications* (*TAA*) [3]. В данной статье предлагается метод вычислений с временной уверенностью (ВВУ), который развивает принцип *TAA*, опираясь на использование информации о времени порождения каждого показания датчиков. Это позволяет точно оценить общее время прохождения этого конкретного измерения через распределенную систему и соответственно скорректировать управляющее воздействие вместо ожидания максимально возможного времени задержки.

Как один из важных шагов в направлении решения данной проблемы в данной работе предлагается формальная семантика ФБ стандарта МЭК 61499 с временными метками, что позволит производить корректную реализацию систем управления, учитывающих время, существенно упростить построение моделей для верификации и симуляции, обеспечить справедливость

модели и полный детерминизм выполнения системы ФБ на уровне отдельного ресурса.

1. Неформальное описание семантики

В данной работе используются *развернутые одноуровневые системы ФБ* (РОСФБ). Как было показано в работах [4, 5], любую многоуровневую систему ФБ можно свести к одноуровневой, используя концепцию клапана данных.

В нашей интерпретации одноуровневые системы ФБ будут включать следующие элементы:

1) базисные ФБ;
2) сервисные интерфейсные ФБ (СИФБ);
3) входные интерфейсы составных ФБ (СФБ);
4) выходные интерфейсы СФБ.

Эти же элементы являются и *элементами выполнения* в системе. Субприложения в данной работе не рассматриваются, поскольку они могут быть «раскрыты» без внесения в результирующую модель дополнительных элементов. Базисный ФБ выполняется как единое целое. Пока выполнятся базисный ФБ, никакие другие действия невозможны (кроме фиксации выходных событий из СИФБ-источников). Согласно классификации выполнения СФБ [6], в нашей работе они выполняются как контейнеры (в отличие от атомарного выполнения).

Преимуществом РОСФБ является увеличение в ряде случаев скорости имитационного моделирования, выполнения и верификации, а также однородность описания системы, что упрощает ее формализацию. Из этого следуют универсальность и унифицированность описания для СФБ и СИФБ. Основным недостатком использования РОСФБ является независимость выполнения интерфейсов, что может привести к нетипичным ситуациям, таким как реентерабельность ФБ, многократное исполнение компонентных ФБ, чередование исполнения компонентных ФБ. Данных ситуаций можно избежать на стадии проектирования, используя специальные паттерны.

На рис. 1–3 представлен пример, иллюстрирующий концепцию РОСФБ. В соответствии со стандартом МЭК 61499 исходная система ФБ представляется в виде многоуровневой иерархии (рис. 1). Иерархические уровни определяются СФБ. Раскрывая СФБ и заменяя их соответствующей парой интерфейсов (входным и выходным), переходим к развернутой одноуровневой структуре, иерархия которой представлена на рис. 2. На рис. 3 приведен фрагмент РОСФБ, иерархия которой представлена на рис. 2.

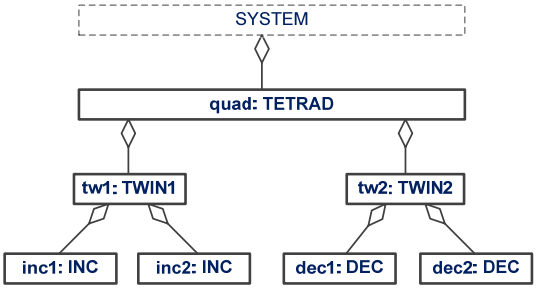

Рис. 1. Фрагмент иерархии многоуровневой системы ФБ

В многоуровневой системе ФБ однозначная идентификация элементов системы (например, событийных входов и выходов) поддерживается возможностью использования иерархических имен, включающих путь от корня к листьям дерева, с использованием той же точечной нотации (например, *quad.tw2.req*). В одноуровневой системе однозначная идентификация может поддерживаться только на основе сквозной нумерации элементов, возможно разделенных для удобства на классы. Все элементы нумеруются сквозной нумерацией в своем классе.

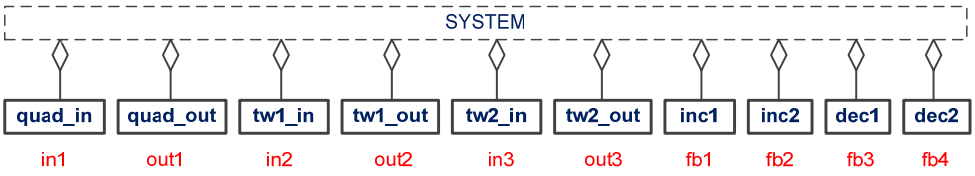

Рис. 2. Фрагмент иерархии развернутой одноуровневой системы ФБ, полученной из многоуровневой системы ФБ, представленной на рис. 1

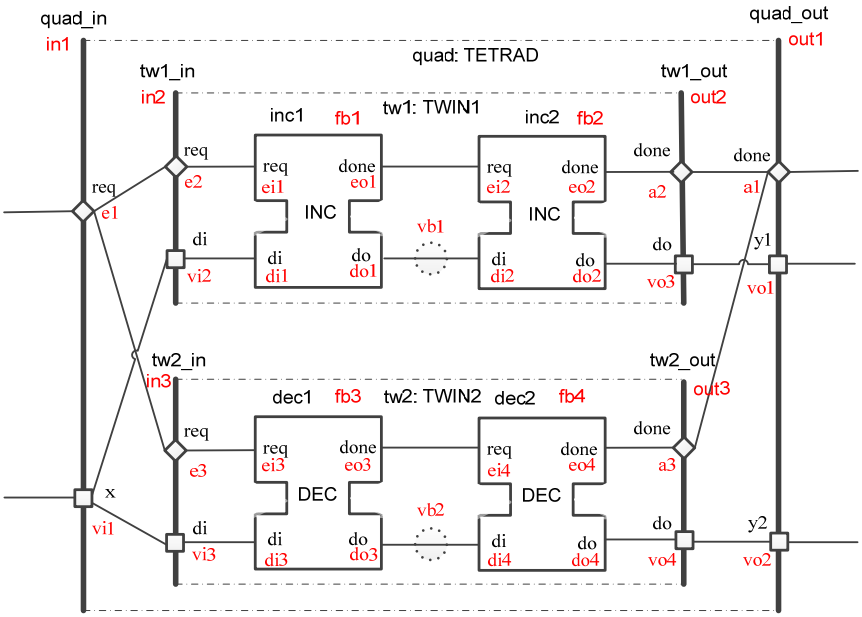

Рис. 3. Фрагмент развернутой одноуровневой системы ФБ, иерархия которой представлена на рис. 2

На рис. 3 в качестве артефактов в схему включены буферные переменные (буферы данных) – *vb1* и *vb2*. Они не являются синтаксическими элементами описания, но используются в семантической модели выполнения системы. Буферные переменные включаются в разрывы информационных связей только между ФБ. В разрывы информационных связей интерфейсов буферы не включаются, поскольку считается, что переменная, инкапсулированная в интерфейс, выполняет функции буфера данных.

Как основа для работы с временными атрибутами используется модель выполнения ФБ с временными метками, предложенная в [7]. В этой модели

элементами диспетчеризации являются входные и выходные события элементов выполнения, и именно они выстраиваются в хронологически упорядоченную очередь [7]. Использование временных меток обеспечивает справедливость, поскольку более ранние (реальные) события выполняются раньше; потенциальную возможность обеспечивает глобальная упорядоченность выполнения всей системы и поддержка концепции ВВУ.

Для обеспечения полной детерминированности выполнения системы ФБ в нашем случае, кроме временных меток, предлагается использовать принцип уникальности приоритетов любого элемента выполнения. Приоритет учитывается только в том случае, если временные метки событий равны.

Предполагается, что ресурс включает диспетчер, который управляет выполнением системы ФБ. В работе диспетчера используется очередь событий, упорядоченная в хронологическом порядке (по времени последнего изменения события). Элементами диспетчеризации в очереди событий являются событийные входы/выходы, а элементами выполнения являются ФБ и интерфейсы.

Для упрощения модели будем использовать *неявную* очередь событий, которая образуется как множество событий, находящихся на входных и выходных событийных линиях базисных и сервисных интерфейсных ФБ, а также интерфейсов СФБ. На этом множестве задано хронологическое отношение порядка. Кроме того, при описании работы диспетчера удобно использовать не очередь событий, а виртуальную *очередь элементов выполнения* (ФБ, интерфейсов), которая может быть построена на основе хронологической очереди событий однозначным образом. Упрощенный алгоритм работы (реального) диспетчера можно представить так:

```
ПОКА <очередь элементов выполнения не пуста>
ДЕЛАТЬ
   Шаг 1. Выборка элемента выполнения с минимальной
временной меткой и максимальным приоритетом
   Шаг 2. Выполнение элемента, включающее корректировку
временных меток событий и их рассылку соседним элементам
КОНЕЦ_ПОКА
```

Поскольку в системе ФБ допускаются СИФБ-источники событий, то при передвижении событий надо учитывать не только тот элемент (ФБ), который только что закончил выполнение, но и СИФБ-источники событий. Будем считать, что включение выходных событий СИФБ-источника в очередь событий будет происходить безопасно и асинхронно, параллельно с работой алгоритма работы диспетчера.

Для правильного функционирования системы предлагается оформить выполнение элементов очереди как критическую секцию. Для ее организации в модель вводится двоичный семафор *Sem*. Рассылка или передвижение событий между ФБ и интерфейсами (с корректировкой временных меток) в модели и реальной системе выполняется, как правило, самим выполняемым элементом.

Существует три основных ограничения предлагаемой модели семантики ФБ:

1) модель не учитывает ошибки синхронизации, но это без потери общего смысла; при необходимости обработка данной ошибки может быть вынесена на уровень пользовательских алгоритмов;

2) рассматривается функционирование системы только на уровне одного ресурса;

3) предполагается, что выполнение ФБ на ресурсе последовательное.

Формальное описание модели будет состоять из двух частей – схемы модели и динамики модели. Схема модели описывает статическую часть модели, включающую описание основных синтаксических конструкций, переменных, а также общих схем для функций вычисления этих переменных. Динамика модели определяется правилами изменения переменных состояния системы.

2. Формальная модель развернутой одноуровневой системы ФБ

Определение схемы модели

Модель развернутой одноуровневой системы ФБ (МРОСФБ) S может быть представлена формально следующим образом:

$$M_S = (Synt_S, Sem_S),$$

где $Synt_S$– синтаксическая часть описания (на уровне абстрактного синтаксиса); Sem_S – семантическая часть описания.

Примечание: нижний индекс S здесь и далее будет отмечать объекты, относящиеся к системе в целом.

Синтаксическая часть описания МРОСФБ определяется четверкой:

$$Synt_S = (FB_S, IN_S, OUT_S, prior_S, EvConn_S, DataConn_S),$$

где $FB_S = \{fb_1, fb_2, ..., fb_{N_{FB}}\}$ – множество компонентных ФБ; fb_i = ($FBInterface^i$, fbt^i), $i \in [1, N_{FB}]$, где $FBInterface^i = (EIX^i_{FB}, EOX^i_{FB}, VI^i_{FB}, VO^i_{FB})$ – интерфейс компонентного ФБ.

В интерфейс входят: EIX^i_{FB} и EOX^i_{FB} – множества (нагруженных) событийных входов и выходов i-го компонентного ФБ, соответственно; VI^i_{FB} и VO^i_{FB} – множества информационных входов и выходов; fbt^i – тип i-го компонентного ФБ. Как было сказано выше, компонентными ФБ могут быть только базисные и сервисные интерфейсные ФБ:

$IN_S = \{in_1, in_2, ..., in_{N_{IN}}\}$ – множество входных интерфейсов СФБ,

$in_i = (INInterface^i, cfbt^i)$, $i \in [1, N_{CFB}]$, где $INInterface^i = (EIX^i_{IN}, VI^i_{IN})$.

В данный интерфейс входят:

• EIX^i_{IN} – множество (нагруженных) событийных входов и VI^i_{IN} – множества информационных входов i-го СФБ;

• $cfbt^i$ – тип i-го составного ФБ.

• $OUT_S = \{out_1, out_2, ..., out_{N_{OUT}}\}$ – множество выходных интерфейсов СФБ, out_i = ($OUTInterface^i$, $cfbt^i$), $i \in [1, N_{CFB}]$, где $OUTInterface^i = (EOX^i_{OUT}, VO^i_{OUT})$.

В данный интерфейс входят: EOX^i_{OUT} – множество (нагруженных) событийных выходов i-го СФБ; VO^i_{OUT} – множества информационных выходов (иначе, множества выходных переменных) i-го СФБ.

Следует отметить, что интерфейсы СФБ должны быть парными. Каждому входному интерфейсу должен соответствовать ровно один выходной интерфейс, т.е. $in_i \in IN_S \leftrightarrow out_i \in OUT_S$.

Введем обозначения с нижним индексом *FB* для множеств, объединяющих все элементы всех компонентных ФБ, входящих в систему и с индексами *IN* и *OUT* для множеств, объединяющих элементы входных и выходных интерфейсов СФБ соответственно. Например: $EIX_{FB} = \bigcup_{i=1}^{N_{FB}} EIX_{FB}^{i}$ – множество (нагруженных) событийных входов всех компонентных ФБ, входящих в систему; $EIX_{IN} = \bigcup_{i=1}^{N_{IN}} EIX_{IN}^{i}$ – множество (нагруженных) событийных входов всех входных интерфейсов СФБ, входящих в систему.

Тогда множества событийных и информационных связей можно определить следующим образом:

$$EvConn_S \subseteq (EOX_{FB} \cup EOX_{OUT} \cup EIX_{IN}) \times (EIX_{FB} \cup EIX_{IN} \cup EOX_{OUT}),$$

$$DataConn_S \subseteq (VO_{FB} \cup VO_{OUT} \cup VI_{IN}) \times (VI_{FB} \cup VI_{IN} \cup VO_{OUT}).$$

Обозначим EX_S – множество всех событийных входов и выходов всех элементов системы, участвующих в выполнении, получаемое из множества $EIX_{FB} \cup EIX_{IN} \cup EOX_{FB} \cup EOX_{OUT}$ простой перенумерацией его элементов. Важно, что между этими множествами существует взаимно однозначное соответствие.

Для перехода индексации с локального уровня на глобальный, принятый в *EX*, используется функция $\widehat{\ }$ («верхняя крышка»):

$$\widehat{\ } : EIX_{FB} \cup EIX_{IN} \cup EOX_{FB} \cup EOX_{OUT} \to EX_S.$$

В этом случае, например, для элемента $eix_m^i \in EIX_{IN}^i$, определенного в локальном множестве, запись $\widehat{eix_m^i}$ означает перевод его в систему «координат» покрывающего множества EX_S.

Для обратного перехода, с глобального на локальный уровень, используется функция $\widetilde{\ }$ («верхняя волнистая черта»):

$$\widetilde{\ } : EX_S \to EIX_{FB} \cup \ \cup EIX_{IN} \cup EOX_{FB} \cup EOX_{OUT}.$$

Например, запись $\widetilde{ex_k}$ означает перевод элемента в элемент с локальной индексацией.

Обозначим VIO_S – множество всех информационных входов и выходов (т.е. входных и выходных переменных) всех элементов системы, участвующих в выполнении, получаемое из множества $VI_{FB} \cup VI_{IN} \cup VO_{FB} \cup VO_{OUT}$ простой перенумерацией его элементов, $VIO_S = \{vio_1, vio_2, ..., vio_{N_V}\}$.

Обозначим $EL_S = FB_S \cup IN_S \cup OUT_S$ – множество всех элементов выполнения, $EL_S = \{el_1, el_2, ..., el_{N_{EL}}\}$. Введем функцию, отображающую каждый элемент $ex_k^i \in EX$ в «несущий» его элемент выполнения: $el : EX_S \to EL_S$.

Введем функцию $prior_{EL} : EL_S \to N^+$, назначающую приоритеты элементам выполнения. Должно выполняться следующее ограничение, согласно которому элементы выполнения должны иметь уникальные приоритеты:

$$\forall x, y \in FB_S \cup IN_S \cup OUT_S [x \neq y \to prior_S(x) \neq prior_S(y)].$$

Семантическая часть описания МРОСФБ определяется тройкой:

$$Sem_S = (VRT_S, T_S),$$

где VRT_S – набор переменных времени выполнения МРОСФБ; T_S – набор функций переходов МРОСФБ.

Набор переменных времени выполнения определяется кортежем:

$$VRT_S = (EIX_{FB}, EOX_{FB}, EIX_{IN}, EOX_{OUT}, VI_{FB}, VO_{FB}, VI_{IN}, VO_{OUT}, VB, Sem, \tau),$$

где первые восемь элементов кортежа наследуются из синтаксического описания системы; $VB_S = \{vb_1, vb_2, ..., vb_{N_{VB}}\}$ – множество буферных переменных (буферов данных); *Sem* – двоичный семафор для сериализации выполнения элементов; τ – переменная системного времени (системный таймер), являющаяся глобальной для всей системы. Буферы данных не определяются в стандарте МЭК 61499, но, как показано в [4, 5], они необходимы в модели функционирования ФБ. В модели используется схема, когда каждой выходной переменной базисного ФБ назначается буфер данных. В этом случае $|VB_S|=|EOX_{FB}|$ и для идентификации буферов данных могут использоваться те же индексы, что и для выходных переменных ФБ, например: выходной переменной ФБ vo_m^i будет соответствовать буфер данных vb_m^i. В результате множество буферов данных может быть представлено как $VB_S = \left\{vb_{a_1}^{b_1}, vb_{a_2}^{b_2}, ..., vb_{a_n}^{b_n}\right\}$.

Определение динамики модели

Динамика модели определяется изменением значений переменных состояния в процессе функционирования системы. Сгруппируем переменные в классы (группы) и определим функции значений для каждой из групп. В дальнейшем $\mathrm{Pr}_j(X)$ будет обозначать j-ю проекцию отношения X.

Определим следующие классы переменных:

$EI_S = \mathrm{Pr}_1(EIX_{IN} \cup EIX_{FB})$ – множество (входных) событийных переменных входных интерфейсов и базисных ФБ;

$TBI_S = \mathrm{Pr}_2(EIX_{IN} \cup EIX_{FB})$ – множество переменных для хранения временных меток «Время зарождения события» для событийных входов входных интерфейсов и базисных ФБ;

$TLI_S = \mathrm{Pr}_3(EIX_{IN} \cup EIX_{FB})$ – множество переменных для хранения временных меток «Время последнего изменения события» для событийных входов входных интерфейсов и базисных ФБ;

$EO_S = \mathrm{Pr}_1(EOX_{IN} \cup EOX_{FB})$ – множество (выходных) событийных переменных выходных интерфейсов и базисных ФБ;

$TBO_S = \mathrm{Pr}_2(EOX_{IN} \cup EOX_{FB})$ – множество переменных для хранения временных меток «Время зарождения события» для событийных выходов выходных интерфейсов и базисных ФБ;

$TLO_S = \mathrm{Pr}_3(EOX_{OUT} \cup EOX_{FB})$ – множество переменных для хранения временных меток «Время последнего изменения события» для событийных выходов выходных интерфейсов и базисных ФБ;

$TB_S = TBI_S \cup TBO_S$ – множество всех временных меток (типа 1) в системе;

$TL_S = TLI_S \cup TLO_S$ – множество всех временных меток (типа 2) в системе;

$VI_S = VI_{IN} \cup VI_{FB}$ – множество входных переменных в системе;

$VO_S = VO_{IN} \cup VO_{FB}$ – множество выходных переменных в системе.

Введем обозначение Z_X: $X \to A$, для функции значений переменных, назначающей переменным из множества X значения из A. Например: функции значений семафора *Sem* и таймера τ будут выглядеть следующим образом:

$$Z_{Sem} : Sem \to \{true, false\},\ Z_\tau : \tau \to Time,$$

где $Time = \{undef, 0, 1, 2, \ldots\}$ – множество значений (дискретного) времени в системе, причем значение *undef* означает, что время не установлено или не имеет значения.

Динамика модели РОСФБ как таковая определяется функционированием элементов, ее составляющих. То есть все действия делегированы на уровень элементов выполнения. На уровне модели РОСФБ определим лишь условия выборки активного события. Для начала определим минимальное значение временной метки в системе

$$\min TL \triangleq \min_{tl \in TL_S} Z_{TL}(tl).$$

Минимальное время определяется среди значений временных меток второго типа для всех событийных входов и выходов ФБ и интерфейсов.

Определим основной предикат, определяющий выбор элемента $ex_k = (e_k, tb_k, tl_k) \in EX_S$ в качестве активного событийного входа/выхода в системе:

$$isActive(ex_k) \triangleq Z_{EI_S \cup EO_S}(e_k) \wedge (Z_{TL_S}(tl_k) = \min TL) \wedge$$

$$\wedge \bigwedge_{\substack{ex_m = (e_m, tb_m, tl_m) \in EX_S, \\ e_k \prec\prec e_m}} \overline{Z_{EI_S \cup EO_S}(e_m) \wedge (Z_{TL_S}(tl_m) = \min TL)}).$$

Для удобства введем функцию для нахождения активного событийного входа/выхода в системе: $selActive : [Z_{EI_S \cup EO_S}] \times [Z_{TL_S}] \to EX_S$. Вычисление данной функции может быть построено на основе предиката *isActive*.

3. Формальная модель входного интерфейса составного функционального блока

На рис. 4,***а*** показано синтаксическое представление входного интерфейса в графическом виде. Семантическая модель сложнее (рис. 4,***б***). Она включает все переменные, описывающие входной интерфейс, а также правила их изменения. Пунктирными кружками обозначены переменные, не входящие в определение входного интерфейса.

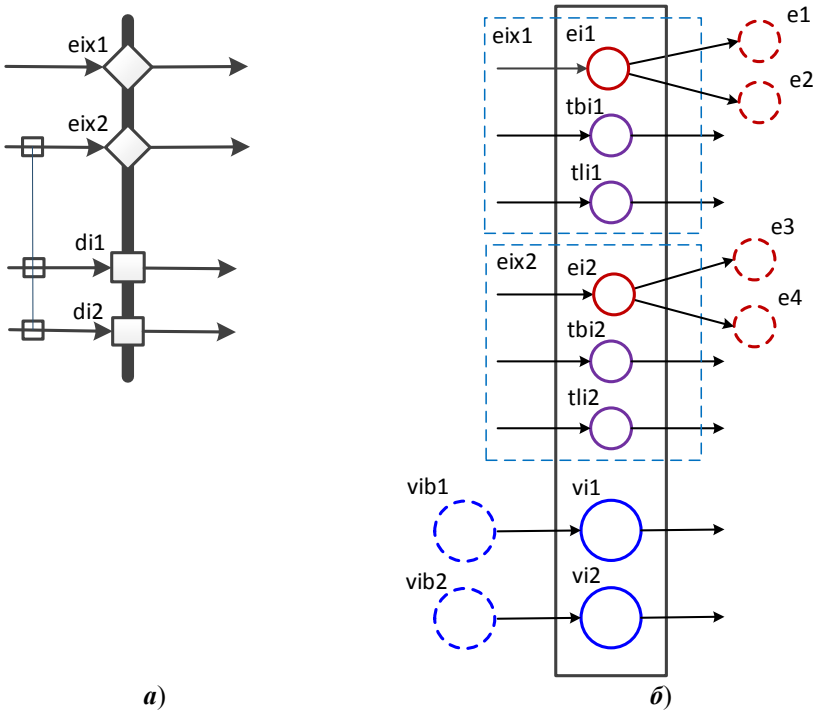

Рис. 4. Синтаксическое (***а***) и семантическое (***б***) представления входного интерфейса СФБ

Определение схемы модели

Модель *i*-го входного интерфейса (МВхИ) $in_i \in IN_S$ СФБ может быть представлена формально следующим образом:

$$M_{IN}^{i} = \left(Synt_{IN}^{i}, Sem_{IN}^{i} \right),$$

где $Synt_{IN}^{i}$ – синтаксическая часть описания (на уровне абстрактного синтаксиса); Sem_{IN}^{i} – семантическая часть описания.

Синтаксическая часть описания МВхИ определяется четверкой:

$$Synt_{IN}^{i} = \left(EIX_{IN}^{i}, VI_{IN}^{i}, IW_{IN}^{i} \right),$$

где EIX_{IN}^{i} – конечное непустое упорядоченное множество (нагруженных) событийных входов; VI_{IN}^{i} – конечное множество (входных) информацион-

ных переменных; $IW_{IN}^{i} \subseteq EIX_{IN}^{i} \times VI_{IN}^{i}$ – множество *WITH*-связей для входов.

Семантическая часть описания МВхИ определяется двойкой:

$$Sem_{IN}^{i} = \left(VRT_{IN}^{i}, T_{IN}^{i} \right),$$

где VRT_{IN}^{i} – набор переменных времени выполнения; T_{IN}^{i} – набор функций вычисления переменных, изменяемых входным интерфейсом.

Набор объектов времени выполнения определяется кортежем:

$$VRT_{IN}^{i} = \left(EIX_{IN}^{i}, VI_{IN}^{i}, \tau \right),$$

где $EIX_{IN}^{i} = \left\{ (ei_k^i, tbi_k^i, tli_k^i) \mid ei_k^i \in EI_{IN}^{i}, tbi_k^i \in TBI_{IN}^{i}, tli_k^i \in TLI_{IN}^{i} \right\}$; EI_{IN}^{i} – множество событийных переменных *i*-го входного интерфейса; TBI_{IN}^{i} – множество переменных для хранения временных меток «Время зарождения события»; TLI_{IN}^{i} – множество переменных для хранения временных меток «Время последнего изменения события»; $VI_{IN}^{i} = \{vi_1^i, vi_2^i, ..., vi_{N_{VI}}^i\}$ – множество переменных *i*-го входного интерфейса; τ – переменная системного времени (системный таймер), являющаяся глобальной для всей системы.

Определение динамики модели

Неформально функционирование входного интерфейса сводится к следующей процедуре:

1) съем данных с внешних буферов в переменные входного интерфейса, связанные WITH-связями с *активным* событийным входом;

2) установка временных меток для выходного события;

3) рассылка сформированного выходного события приемникам событий;

4) сброс активного входного события (как отработавшего).

Определим функцию для вычисления источника данных для переменной v_x входного интерфейса:

$$srcDataVI(v_x) = \{ v_y \mid (v_y, v_x) \in DataConn_S \}.$$

В «локальном» варианте, где определяется источник данных в конкретном множестве элементов выполнения, данная функция может быть переписана так:

$$srcDataVI'(v_x) = \begin{cases} vb_n^f \in VB_S, \text{ если } (\widehat{vo_n^f}, v_x) \in DataConn_S \wedge vo_n^f \in VO_{FB}, \\ vi_k^j \in VI_{IN}, \text{ если } (\widehat{vi_k^j}, v_x) \in DataConn_S \wedge vi_k^j \in VI_{IN}, \\ vo_m^i \in VO_{OUT}, \text{ если } (\widehat{vo_m^i}, v_x) \in DataConn_S \wedge vo_m^i \in VO_{OUT}. \end{cases}$$

Как видно из определения данной функции, источником данных для переменной входного интерфейса может быть буферная переменная (если

переменная v_x соединена с выходом ФБ) или переменная другого входного или выходного интерфейса.

Функционирование входного интерфейса может быть представлено семейством правил (наподобие [8]), в консеквенте которых указаны все переменные, которые изменяют свои значения:

$$\{p_{DV}^{IN}[k]: isActive(\widehat{eix_k^i}) \wedge Z_{Sem}(Sem) \Rightarrow \mathop{\Xi}_{(\widehat{eix_k^i}, ex_j) \in EvConn_S} Z_{EI_S \cup EO_S}(e_j) \leftarrow true;$$

$$\mathop{\Xi}_{(\widehat{eix_k^i}, ex_j) \in EvConn_S} Z_{TB_S}(tb_j) \leftarrow Z_{TBI_S}(\widehat{tbi_k^i});$$

$$\mathop{\Xi}_{(\widehat{eix_k^i}, ex_j) \in EvConn_S} Z_{TL_S}(tl_j) \leftarrow \tau;$$

$$\mathop{\Xi}_{(eix_k^i, vi_m^i) \in IW_{IN}^i} Z_{VI_{IN}^i}(vi_m^i) \leftarrow Z_{VIO_S \cup VB_S}(srcDataVI(\widehat{vi_m^i})); Z_{EI_{IN}^i}(ei_k^i) \leftarrow false;$$

$$Z_{TBI_{IN}^i}(tbi_k^i) \leftarrow undef; Z_{TLI_{IN}^i}(tli_k^i) \leftarrow undef; | eix_k^i \in EIX_{IN}^i, k = \overline{1, | EIX^i |}\}.$$

В соответствии с антецедентом данного правила, чтобы сработал *i*-й входной интерфейс из-за *k*-го событийного входа $eix_k^i = (ei_k^i, tbi_k^i, tli_k^i)$, необходимо, чтобы этот событийный вход был активным и семафор был открыт. Консеквент правила определяет производимые действия при срабатывании данного правила. Поскольку операция срабатывания входного интерфейса является одномоментной, то закрытие семафора не производится.

Знак «;» в правой части продукционного правила означает операцию параллельного выполнения действий. Операндами являются действия, стоящие слева и справа от этого знака. Знак Ξ используется для обозначения параллельного выполнения индексированных операций, стоящих после этого знака. При вычислении функции *min* будем считать, что неопределенные значения (*undef*) не учитываются (или их можно заменить знаком бесконечности ∞).

4. Формальная модель выходного интерфейса составного функционального блока

Функции выходного интерфейса СФБ оказываются во многом сходными с функциями входного интерфейса. В отличие от входного интерфейса, который производит съем данных из окружения в свои внутренние переменные, выходной интерфейс выдает свои внутренние данные во внешнюю среду.

В целом модель выходного интерфейса схожа с моделью входного с определением отдельного семейства правил изменения переменных:

$$\{p_{DV}^{OUT}[k]: isActive(\widehat{eox_k^i}) \wedge Z_{Sem}(Sem) \Rightarrow \mathop{\Xi}_{(\widehat{eox_k^i}, ex_j) \in EvConn_S} Z_{EI_S \cup EO_S}(e_j) \leftarrow true;$$

$$\mathop{\Xi}_{(\widehat{eox_k^i}, ex_j) \in EvConn_S} Z_{TB_S}(tb_j) \leftarrow Z_{TB_S}(\widehat{tbo_k^i});$$

$$\Xi \quad Z_{TL_S}(tl_j) \leftarrow \tau;$$
$$(\widehat{eox_k^i, ex_j}) \in EvConn_S$$

$$\Xi \quad Z_{VIO_S \cup VB_S}(v_x) \leftarrow Z_{VO}(vo_m^i); Z_{EO_{OUT}^i}(eo_k^i) \leftarrow false;$$
$$(eox_k^i, vo_m^i) \in OW_{IN}^i,$$
$$(\widehat{vo_m^i, v_x}) \in DataConn_S$$

$$Z_{TB_{OUT}^i}(tbo_k^i) \leftarrow undef; Z_{TL_{OUT}^i}(tlo_k^i) \leftarrow undef; | eox_k^i \in EOX_{OUT}^i, k = \overline{1, |EOX_{OUT}^i|}\},$$

где EOX^i_{OUT}, EO^i_{OUT}, tbo^i_k, vo^i_m и т.д. – переменные выходного интерфейса (по аналогии с переменными входного).

Заключение

В работе представлена формальная семантика ФБ стандарта МЭК 61499 с временными метками. В качестве основной нотации использован формализм машин абстрактных состояний, предложенных Ю. Гуревичем. Введение временных меток в модель позволяет описывать с ее помощью системы, использующие принцип вычислений с временной уверенностью. Разработанная формальная семантика ФБ может использоваться для верификации распределенных систем управления с учетом временных характеристик, а также служить основой для реализации соответствующей среды исполнения ФБ. Направлением дальнейших исследований является доработка транслятора стандартных XML-описаний ФБ в код SMV (fb2smv) для поддержки новой семантики [9].

Библиографический список

1. IEC 61499 Function Block. – P. 1: Architecture. – ed. 2.0, retrieved 12 October 2015.
2. **Lee, E.A.** The Past, Present and Future of Cyber-Physical Systems: A Focus on Models / E.A. Lee // Sensors. – 2015. – № 15 (3). – P. 4837–4869.
3. Time-aware applications, computers, and communication systems (TAACCS) / M. Weiss, J. Eidson, C. Barry, D. Broman, L. Goldin, B. Iannucci, K. Stanton // Technical Note (NIST TN). – 2015. – Report Number: 1867.
4. **Dubinin, V.** Towards a Formal Semantics of IEC 61499 Function Blocks / V. Dubinin, V. Vyatkin // 4th IEEE International Conference on Industrial Informatics (INDIN'2006). – Singapore, 2006. – P. 6–11.
5. **Dubinin, V.** On Definition of a Formal Model for IEC 61499 Function Blocks / V. Dubinin, V. Vyatkin // EURASIP Journal on Embedded Systems. – 2008. – Vol. 2008, article ID 426713. – 10 p. – DOI 10.1155/2008/426713.
6. **Sünder, C.** Execution Models for the IEC 61499 elements Composite Function Block and Subapplication / C. Sünder, A. Zoitl, J.H. Christensen, M. Colla, T. Strasser // 5th IEEE Int. Conference on Industrial Informatics (INDIN'07). – Vienna, Austria, 2007. – P. 1169–1175.
7. **Dai, W.** Discrete-Event-Based Deterministic Execution Semantics With Timestamps for Industrial Cyber-Physical Systems / W. Dai, C. Pang, V. Vyatkin, J. H. Christensen, X. Guan // IEEE Transactions on Systems, Man, and Cybernetics: Systems. – 2017. – Vol. PP, iss. 99. – P. 1–12.
8. **Dubinin, V.** Formal modeling and verification of IEC 61499 function blocks on the basis of transition systems / V. Dubinin, V. Vyatkin, A. Shalyto // International Siberian Conference on Control and Communications (SIBCON'2016) (Russia, Moscow, May 12–14, 2016). – Moscow, 2016. – P. 1–4.

9. **Drozdov, D.** IEC 61499 function blocks to SMV converter / D. Drozdov. – URL: https://github.com/dmitrydrozdov/fb2smv

References

1. *IEC 61499 Function Block. P. 1: Architecture. ed. 2.0, retrieved 12 October 2015*.
2. Lee E. A. *Sensors*. 2015, no. 15 (3), pp. 4837–4869.
3. Weiss M., Eidson J., Barry C., Broman D., Goldin L., Iannucci B., Stanton K. *Technical Note (NIST TN)*. 2015, Report Number: 1867.
4. Dubinin V., Vyatkin V. *4th IEEE International Conference on Industrial Informatics (INDIN'2006)*. Singapore, 2006, pp. 6–11.
5. Dubinin V., Vyatkin V. *EURASIP Journal on Embedded Systems*. 2008, vol. 2008, article ID 426713. 10 p. DOI 10.1155/2008/426713.
6. Sünder C., Zoitl A., Christensen J. H., Colla M., Strasser T. *5th IEEE Int. Conference on Industrial Informatics (INDIN'07)*. Vienna, Austria, 2007, pp. 1169–1175.
7. Dai W., Pang C., Vyatkin V., Christensen J. H., Guan X. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*. 2017, vol. PP, iss. 99, pp. 1–12.
8. Dubinin V., Vyatkin V., Shalyto A. *International Siberian Conference on Control and Communications (SIBCON'2016) (Russia, Moscow, May 12–14, 2016)*. Moscow, 2016, pp. 1–4.
9. Drozdov D. *IEC 61499 function blocks to SMV converter*. Available at: https://github.com/dmitrydrozdov/fb2smv

Дроздов Дмитрий Николаевич
аспирант, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: dmitriidrozdov9@gmail.com

Drozdov Dmitriy Nikolaevich
Postgraduate student, Penza State University (40 Krasnaya street, Penza, Russia)

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

E-mail: victor_n_dubinin@yahoo.com

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

Вяткин Валерий Владимирович
доктор технических наук, профессор, заведующий кафедрой, Технический университет Лулео (97187, Lulea, Sweden)

E-mail: valeriy.vyatkin@ltu.se

Vyatkin Valeriy Vladimirovich
Doctor of engineering sciences, professor, head of the sub-department, Lulea University of Technology (97187, Lulea, Sweden)

Образец цитирования:

Дроздов, Д. Н. Формальная семантика функциональных блоков МЭК 61499 с временными метками / Д. Н. Дроздов, В. Н. Дубинин, В. В. Вяткин // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2019. – № 1 (49). – С. 41–54. – DOI 10.21685/2072-3059-2019-1-4.