УДК 004.94 DOI: 10.21440/0536-1028-2021-2-113-122

Функциональный подход к моделированию динамики систем детерминированных конечных автоматов

Лапин Э. С.[1]*, Абдрахманов М. И.[2]
[1] Уральский государственный горный университет, г. Екатеринбург, Россия
[2] Компания «ИНГОРТЕХ», г. Екатеринбург, Россия
e-mail: marat-ab@mail.ru

Реферат

Цель работы. *Исследование функционального подхода к моделированию системы конечных автоматов, не ограниченной топологией связи между ее элементами и неоднородностью типов используемых алгоритмов.*

Актуальность. *Существенная часть технических систем, которые эксплуатируются в горной промышленности, может быть описана через модель конечного автомата. К ним относятся шахтные конвейерные системы, системы стволовой сигнализации, системы управления технологическим оборудованием и т. д. Использование такой модели позволяет сократить время на разработку управляющего программного обеспечения, эффективно выполнять работы по исследованию, отладке и тестированию алгоритмов работы. Для решения задачи моделирования динамики конечного автомата существует множество эффективных подходов и инструментов, каждый из которых имеет свои достоинства и недостатки.*

Методология. *В рамках данной статьи рассматривается методология моделирования системы конечных автоматов применительно к шахтным конвейерным системам.*

Результаты. *Разработаны модели конечных автоматов (КА) и условия для моделирования динамики систем КА применительно к шахтным конвейерным системам и комплексам.*

Выводы. *Рассматриваемый подход к моделированию с использованием функторов и аппликативных функторов для композиции модели и исследования динамики ее работы, а также возможность математического доказательства свойств модели делает его хорошей альтернативой при выборе средств для разработки моделей систем.*

Ключевые слова: *моделирование систем; моделирование динамики; конечные автоматы; функторы; аппликативные функторы; функциональное программирование.*

Введение. В горной промышленности существует довольно много систем, которые можно описать через модель конечного автомата: шахтные конвейерные системы, системы стволовой сигнализации, системы управления и блокировки работы вентиляторов и насосов и т. д. Применение такой модели упрощает исследование и разработку алгоритмов управления соответствующими техническими системами, диагностику отказов и позволяет более эффективно проводить процедуру тестирования систем.

Конечные автоматы или конечные динамические системы [1–4] (далее КА) являются универсальным способом описания и моделирования систем, которые характеризуются конечным числом состояний и детерминированным алгоритмом перехода между этими состояниями. В общем случае автомат может быть описан множеством:

$$F = \{S, R, Q, fsm, psy, q_0\}, \tag{1}$$

где S – множество входных сигналов; R – множество выходных сигналов; Q – множество состояний; fsm – функция переходов; psy – функция выходов; q_0 – начальное состояние автомата.

Существует два основных подхода к описанию процесса моделирования динамики систем КА – императивный и декларативный [5–9], которые по сути являются подходами к разработке программного обеспечения (далее ПО). В рамках императивного подхода программа (модель) представляет собой последовательность инструкций, которые необходимо выполнить для решения задачи моделирования. Такой подход является довольно распространенным и используется для моделирования процессов функционирования систем на базе различных схем (сети Петри, системы массового обслуживания и т. п.) [10]. Декларативный подход предполагает задание не последовательности действий, а спецификации решения задачи, в этом случае внимание уделяется описанию задачи, а не процессу ее решения.

В рамках данной статьи авторами предлагается использовать функциональную парадигму (далее ФП) разработки (подвид декларативного подхода) для моделирования динамики систем КА по следующим причинам:

– ФП предполагает проектирование модели системы как математического выражения, это позволяет сосредоточиться на общей схеме решения, а не на последовательности выполняемых инструкций;

– в ФП отсутствуют состояния и переменные, все функции чистые (возвращаемое значение зависит только от входных аргументов) [5–9], все это потенциально снижает количество ошибок при разработке и упрощает отладку;

– лаконичный синтаксис языков ФП-группы, мощная система типов, алгебраические типы данных и наличие необходимых структур данных с реализацией таких концепций, как функторы, монады [5–9] и т. п., позволяет гибко проектировать и тестировать решение.

Моделирование шахтной конвейерной системы. В рамках данной статьи в качестве технической системы, для которой будет построена модель, демонстрирующая функциональный подход к моделированию динамики систем КА, выбрана шахтная конвейерная система по следующим причинам:

– шахтная конвейерная система характеризуется слабой связанностью ее элементов (отдельных конвейеров) в информационном плане друг с другом;

– алгоритм управления шахтным конвейером может быть значительно упрощен без потери его ключевых особенностей;

– шахтная конвейерная система предполагает несколько вариантов соединения ее элементов друг с другом, что является важным для демонстрации рассматриваемого в статье подхода.

Для построения КА, моделирующего шахтный конвейер, сократим множество (1) до следующего набора:

$$F = \{S, Q, fsm, q_0\}.$$

Конвейер может находиться в состояниях: остановлен (*Stop*), готов к запуску (*Ready*), запущен (*Start*) и заблокирован (*Block*). Построим множество состояний Q:

$$Q = \{Stop, Ready, Start, Block\}.$$

На работу реального конвейера оказывают влияние больше двух десятков сигналов, из них выделим следующие: сигналы с датчиков контроля схода ленты (*ksl*) и экстренное ограждение (*eo*), сигнал аварии (*avs*), готов к запуску (*ps*), кнопки: стоп (*bsp*), пуск (*bst*) и авария (*bav*). Входным сигналом для КА будет кортеж, состоящий из всех перечисленных выше сигналов. В итоге множество таких кортежей составляет множество входных сигналов:

$$S = \{(ksl, eo, avs, ps, bsp, bst, bav), ksl, eo, avs, ps, bsp, bst, bav \in \{0,\ 1\}\}.$$

Сформируем композитные сигналы для удобства описание работы КА:

– сигнал *blockSignal*, для него должно быть истинно следующее логическое выражение из компонент вектора элемента *S*:

$$blockSignal = avs\ \vee\ bav;$$

– сигнал *stopSignal*, для него должно быть истинно выражение:

$$stop = (ksl \wedge eo \wedge ps \wedge bsp)\ \vee\ (\neg avs\ v\ \neg bav).$$

Диаграмма работы КА шахтного конвейера представлена на рис. 1.

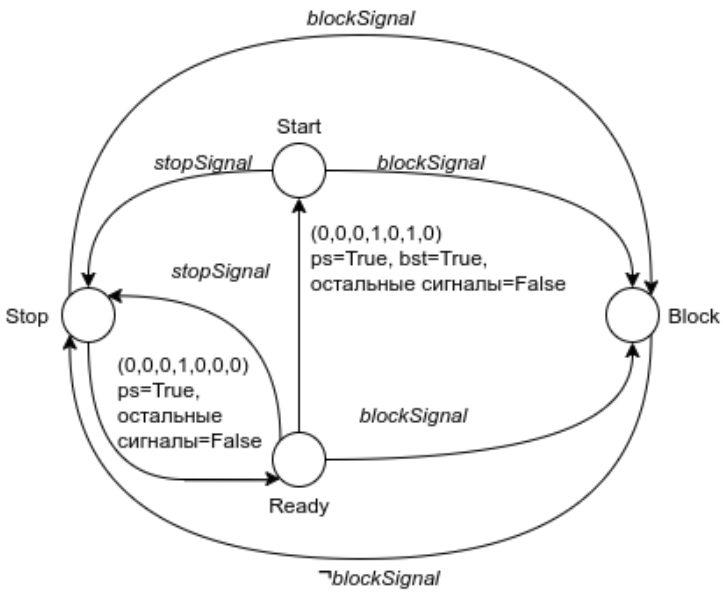

Рис. 1. Диаграмма работы КА шахтного конвейера
Fig. 1. Diagram of mine conveyor FSA operation

Приведенная диаграмма может быть реализована в виде функции *fsm*:

$$x^t = fsm\left(x^{t-1},\ s\right), \tag{2}$$

где x^{t-1} – состояние КА на момент выполнения функции перехода; s – входной сигнал; x^t – состояние КА после перехода.

Существует несколько подходов в записи работы КА, один из них – это использование лент [1]. Как правило, если сложность КА достаточно низка, как в данном примере, то реализация программной модели и непосредственно моделирование его работы не вызывает затруднений.

Часто интерес представляют не конкретные реализации КА, а системы КА. Для описания системы КА необходимо знать:

– алгоритмы работы КА, из которых состоит система;

– топологию сети связей между элементами системы.

Такой подход в полной мере применим для шахтных конвейерных систем. В них отдельные конвейеры, работа которых описывается с помощью конкретных КА, соединяются в системы, транспортирующие уголь и горную породу от места отработки (лава) до места разгрузки. Топология связи может иметь линейный вид, когда система представляет собой цепочку связанных друг с другом конвейеров (рис. 2, *а*), либо древовидный, в этом случае есть места, в которых конвейерные цепочки могут соединяться (рис. 2, *б*).

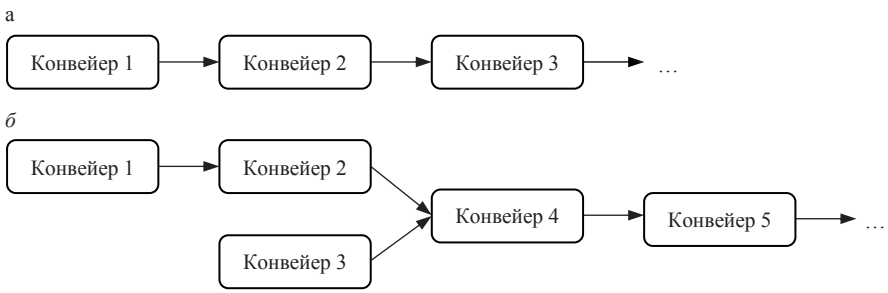

Рис. 2. Линейная топология – *а* и древовидная топология – *б* связи конвейеров
Fig. 2. Linear topology – *a* and tree topology– *б* of conveyor communication

Для моделирования топологии могут быть использованы такие структуры данных, как связный список, который может представлять линейную структуру, и дерево (либо граф), реализующее древовидную структуру. Такие структуры данных принято называть *контейнерами*, далее будет использоваться именно этот термин.

Пример линейной топологии, представленной в виде списка:

$$\left[x_1,\ x_2,\ldots,\ x_i,\ldots,\ x_N\right],\ x_i \in Q,\ N \in \mathbb{N},$$

где x_i – это состояние i-го конвейера в цепочке (всего конвейеров N).

Для моделирования динамики системы КА необходимы:

– алгоритм КА, согласно которому будет изменяться состояние элементов системы;

– наборы входных сигналов для каждого элемента системы;

– контейнер для хранения топологии связи.

Применение функторов для моделирования динамики конечных автоматов. Алгоритм работы КА, как уже было сказано ранее, может быть описан в виде функции (2). Наборы сигналов подготавливаются специалистом по моделированию для тестирования работы автомата либо могут быть взяты с реально работающей технической системы, поведение которой исследуется.

При наличии указанных составляющих основная задача моделирования динамики системы КА состоит в организации выполнения функции, реализующей логику работы КА, на уровне контейнера, не меняя структуру последнего, т. е. не нарушая топологию.

Эффективным инструментом для решения задачи, удовлетворяющим указанным требованиям, является функтор из теории категорий [9, 11]. Функтор выполняет отображение между категориями, причем помимо объектов он также отображает и морфизмы.

Если в категории C есть морфизм, который переводит a в b:

$$f\colon a \to b,$$

а F – это функтор, который отображает объекты из C в D, то F также будет отображать морфизм f в $F\,f$ (рис. 3), такой что:

$$F\,f\colon F\,a \to F\,b.$$

Эта идея может быть принята для решения задачи моделирования системы КА. Функция КА, моделирующая его динамику, при некоторой модификации выражения (2) выглядит следующим образом:

$$y = fsm(s,\ x), \tag{3}$$

где x – состояние КА на момент выполнения функции перехода; s – входной сигнал; y – состояние КА после перехода.

Для конкретного значения s функция (3) будет определяться только значением x. Для того, чтобы работать с КА, находящимися в контейнере, нам нужен функтор, который обеспечит выполнение функции (3) на уровне контейнера (рис. 4).

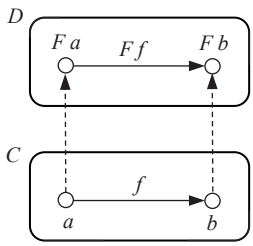

Рис. 3. Функтор в теории категорий
Fig. 3. Functor in the category theory

Эта возможность реализована во многих функциональных языках программирования. Воспользуемся для моделирования шахтной конвейерной системы языком программирования *Haskell* [9, 12]:

Реализуем КА шахтного конвейера.

Тип данных, описывающий состояние конвейера:

```
data ConvState = Stop | Ready | Run | Block
  deriving Show
```

Тип данных, описывающий входной сигнал КА:

```
data SignalState a = Sg {ksl:: a, eo :: a, avs :: a, ps :: a, bsp :: a, bst :: a, bav :: a}
  deriving Show
```

Функция, реализующая алгоритм работы КА ленточного конвейера (диаграмма на рис. 1):

```
tpConvFsm :: ConvState -> SignalState Bool -> ConvState
tpConvFsm Stop sig   | avs sig == True || bav sig == True = Block
   | ksl sig == False && eo sig == False && ps sig == True = Ready
   | otherwise = Stop
tpConvFsm Ready sig   | avs sig == True || bav sig == True = Block
    | ksl sig == True || eo sig == True || ps sig == False || bsp sig == True =
Stop
    | ksl sig == False && eo sig == False && bst sig == True = Run
    | otherwise = Ready
tpConvFsm Run sig   | avs sig == True || bav sig == True = Block
    | ksl sig == True || eo sig == True || ps sig == False || bsp sig == True =
Stop
    | otherwise = Run
tpConvFsm Block sig   | avs sig == False && bav sig == False = Stop
    | otherwise = Block
```

В качестве примера для моделирования используем конвейерную систему с линейной топологией. Контейнером для ее представления выберем связный список. В *Haskell* связный список реализован как тип данных *List*, который является представителем класса типов *Functor*, т. е. для него уже реализован весь необходимый функционал, который позволит выполнить вычисления: применить функцию *tpConvFsm* ко всем элементам контейнера, не меняя его структуру.

Создадим конвейерную цепочку, состоящую из трех конвейеров, находящихся в состоянии *Stop*:

```
convs = Stop:Stop:Stop:[]
```

Так как список является представителем класса типов *Functor*, это позволяет использовать операторы для выполнения определенных вычислений. Например, можно перевести все конвейеры принудительно в состояние *Ready,* для этого воспользуемся оператором *<$*:

```
Main> Ready <$ convs
[Ready,Ready,Ready]
```

Для моделирования источника сигналов реализуем функцию *getSignals*:

```
getSignals :: Int -> [SignalState Bool]
getSignals n = replicate n (Sg { ksl=False, eo=False, avs=False, ps=True,
bsp=False, bst=False, bav=False })
```

Она формирует список одинаковых наборов сигналов, количество которых определяется аргументом *n*. При наличии отдельных списков сигналов и состояний конвейеров их удобно объединить попарно в двухэлементные кортежи, поэтому необходимо преобразовать *tpConvFsm* в функцию одного аргумента, для этого воспользуемся функцией *uncurry*:

```
tpConvFsm' = uncurry tpConvFsm
```

Выполним моделирование системы из трех конвейеров, которые изначально находятся в состоянии *Stop*, с использованием алгоритма *tpConvFsm'* и сигналов из *getSignals* с помощью оператора *<$>*, который является аналогом *fmap* (рис. 5):

```
Main> tpConvFsm' <$> zip convs (getSignals (length convs))
[Ready,Ready,Ready]
```

Как уже было сказано ранее, алгоритм формирования набора сигналов определяется оператором, выполняющим моделирование. Сигналы могут быть подготовлены заранее либо взяты из базы данных реальной системы.

При самостоятельной реализации функтора для контейнера, хранящего топологию системы, необходимо соблюсти выполнение законов:

1. $fmap\ id \equiv id$;
2. $fmap\ (f\ .\ g) \equiv fmap\ f\ .\ fmap\ g$,

где *id* – это функция, которая возвращает переданный аргумент в неизменном виде. Запись $(f\ .\ g)$ является представлением композиции функции, эквивалентной $f(g(x))$.

Суть перечисленных законов заключается в том, чтобы при применении функции *fmap* топология системы (структура контейнера) не изменялась.

Применение аппликативных функторов для моделирования динамики конечных автоматов. С помощью функтора можно решить только ограниченный набор задач, которые возникают при моделировании динамики систем КА. Основное ограничение заключается в требовании, чтобы система была гомогенной, т. е. все ее элементы должны описываться одним и тем же КА. Довольно часто это условие не выполняется. В состав шахтной конвейерной системы могут входить конвейеры и установки, алгоритмы работы которых отличаются друг от друга.

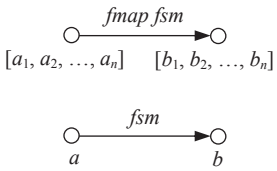

Рис. 4. Организация выполнения функции на уровне контейнера
Fig. 4. Organization of a function execution at the level of a container

Например, лавный конвейерный комплекс включает в себя скребковый конвейер, перегружатель и дробилку, а в состав конвейерной цепочки может входить питатель (рис. 5).

Также при моделировании может возникнуть задача одновременного тестирования нескольких алгоритмов работы конвейера на заданном наборе состояний и входных сигналов.

Одно из решений данной задачи – это поместить алгоритмы КА, описывающие логику работы элементов системы, в контейнер с топологией и структурой, аналогичной тому, в котором хранятся состояния элементов. Данная задача не может быть решена с помощью функтора, так как он не предполагает хранение функций в контейнере. Для этого могут быть использованы аппликативные функторы [9, 12–15], которые предоставляют возможность хранения функций внутри контейнера.

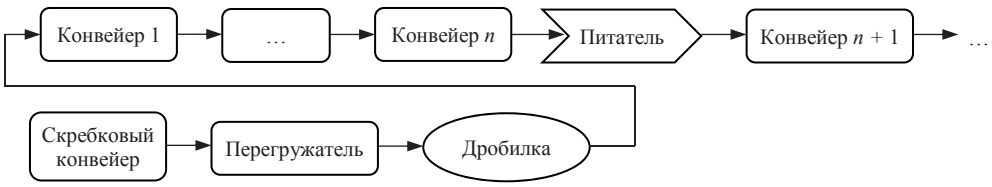

Рис. 5. Неоднородная шахтная конвейерная система
Fig. 5. Heterogeneous mine conveyor system

Для демонстрации использования аппликативных функторов в моделировании динамики систем КА воспользуемся гетерогенной конвейерной цепочкой с линейной топологией, вид которой представлен на рис. 6.

В представленной системе алгоритмы работы элементов отличаются друг от друга. Каждый из алгоритмов реализован как соответствующая функция:

lvConvFsm – алгоритм работы скребкового конвейера;
crsFsm – алгоритм работы дробилки;
ldConvFsm – алгоритм работы перегружателя;
tpConvFsm – алгоритм работы ленточного конвейера.

Выполним моделирование работы системы на языке *Haskell*, этот язык содержит нужную нам реализацию списка как представителя класса типов *Applicative*, называемый *ZipList*. При этом обычный список не может быть использован, так

как логика его работы заключается в том, что функции соответствующего списка по очереди применяются ко всем элементам из списка с данными.

Проиллюстрируем это на примере [2, 3, 6, 15]:

*> fns = [\x -> x * 3, \x -> x + 1] --список функций*
> dt = [1, 5] --набор данных для обработки
> fns <> dt --выполнение операции <*>*

ZipList предоставляет нужную реализацию. В итоге упаковка в контейнер производится с помощью конструктора класса *ZipList*, а распаковка с помощью функции *getZipList* [3, 6]:

> getZipList $ ZipList fns <> ZipList dt*

Создадим контейнер с состояниями элементов транспортной системы:
convs' = Stop:Stop:Stop:Stop:[]

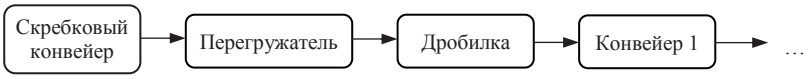

Рис. 6. Гетерогенная конвейерная цепочка с линейной топологией
Fig. 6. Heterogeneous conveyor system with linear topology

Создадим контейнер с алгоритмами работы элементов системы, для чего подготовим список функций:

fsms' = lvConvFsm:ldConvFsm:crsFsm:convFsm:[]

и список функций, выполняющих операцию, обратную каррированию (см. пример с функтором, где сделано то же самое, но для одной функции):

mods' = replicate (length fsms') uncurry

Построим контейнер с функциями подходящего вида. Здесь уже понадобятся возможности аппликативного функтора для того, чтобы применить функции из *mods'* к функциями из *fsms'*, т. е. построить их композицию, где оператором применения является <*>:

modFsms' = getZipList $ ZipList mods' <> ZipList fsms'*

Выполним один шаг моделирования:

Main> getZipList $ ZipList modFsms' <> ZipList (zip convs' (getSignals (length convs')))*
[Ready,Ready,Ready,Ready]

Для контейнера *ZipList* в языке *Haskell* выполняются законы аппликативных функторов [3]. Если возникнет задача самостоятельной реализации функционала аппликативного функтора для своего контейнера, то необходимо убедиться, что он соответствует данным правилам.

Выводы. Функциональный подход к моделированию динамики систем КА, благодаря своему удобству, является хорошим выбором при работе с моделью

в режиме цикла *чтение–вычисление–вывод* (REPL) [9, 12]. Доказуемость свойств разрабатываемой модели (программы) алгебраическими методами [9] и возможности организации вычислений с использованием идей функторов и аппликативных функторов делает данный подход хорошей альтернативой при выборе средств для разработки моделей систем.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Логика, автоматы, алгоритмы / М. А. Айзерман [и др.]. М.: Физматгиз, 1963. 556 с.
2. Хопкрофт Д., Мотвани Р., Ульман Д. Введение в теорию автоматов, языков и вычислений: пер. с англ. М.: Вильямс, 2002. 528 с.
3. Гилл А. Введение в теорию конечных автоматов. М.: Наука, 1966. 272 с.
4. Стюарт Т. Теория вычислений для программистов: пер. с англ. М.: ДМК Пресс, 2014. 384 с.
5. Роберт В. Себеста. Основные концепции языков программирования. М.: Вильямс, 2001. 672 с.
6. Абельсон Харольд, Сассман Джеральд Джей. Структура и интерпретация компьютерных программ. М.: Добросвет, 2018. 608 с.
7. Simon Peyton Jones. The Implementation of functional programming languages. Hemel Hempstead: Prentice Hall, 1987. 445 с.
8. Филд А., Харрисон П. Функциональное программирование. М.: Мир, 1993. 637 с.
9. Москвин Д. Н. Курс «Функциональное программирование». URL: https://nsk.compsciclub.ru/courses/func-prog/2020-spring/classes/ (дата обращения: 21.09.2020).
10. Советов Б. Я., Яковлев С. А. Моделирование систем. М.: Высшая школа, 2001. 343 с.
11. Bartosz Milewski. Category Theory for Programmers. URL: https://github.com/hmemcpy/milewski-ctfp-pdf/releases/download/v1.3.0/category-theory-for-programmers.pdf (дата обращения: 21.09.2020).
12. Уилл Курт. Программируй на Haskell. М.: ДМК Пресс, 2019. 648 с.
13. McBride Conor, Paterson Ross. Applicative programming with effects // J. Funct. Program. 2008. No. 18(1) P. 1–13.
14. Paterson Ross. Constructing applicative functors // Mathematics of Program Construction. Lecture Notes in Computer Science. 2012. Vol. 7342. P. 300–323.
15. Gibbons Jeremy, Oliveira Bruno C. d. S. The essence of the Iterator pattern // Journal of Functional Programming. 2009. No. 19. P. 377–402.

Поступила в редакцию 2 ноября 2020 года

Сведения об авторах:

Лапин Эдуард Самуилович – доктор технических наук, профессор, профессор кафедры автоматики и компьютерных технологий Уральского государственного горного университета. E-mail: lapin.eduard2014@yandex.ru
Абдрахманов Марат Ильдусович – кандидат технических наук, главный специалист Компании «ИНГОРТЕХ». E-mail: marat-ab@mail.ru

DOI: 10.21440/0536-1028-2021-2-113-122

Functional approach to deterministic finite-state automata systems dynamic modeling

Eduard S. Lapin[1], Marat I. Abdrakhmanov[2]
[1] Ural State Mining University, Ekaterinburg, Russia.
[2] Ingortech company, Ekaterinburg, Russia.

Abstract
Research aim *is to study the functional approach to modeling the deterministic finite-state automata system which is not confined to the elements communication topology and the heterogeneity of the algorithm types.*
Relevance. *The substantial part of engineering systems applied in the mining industry may be described through the finite-state automaton model. They include the mine conveyor systems, shaft signal systems, processing facilities control systems, etc. Such model makes it possible to shorten the time spent on control software development and carry out algorithm analysis, debug, and testing effectively. There are a lot of effective approaches and tools to solve the problem of finite-state automata dynamic modeling, each of which has its own advantages and disadvantages.*
Methodology. *In this article, the methodology of finite-state automata systems modeling is considered as applied to mine conveyor systems.*
Results. *Final-state automata (FSA) models have been developed together with the conditions for FSA systems dynamic modeling as applied to mine conveyor systems.*
Conclusions. *The considered approach to modeling, which involves functors and applicative functors*

for structure composition and its operational dynamics study, as well as the possibility to mathematically prove the model's properties, makes the approach a good alternative when choosing tools for systems models development.

Key words: *systems modeling; dynamic modeling; finite-state automata; functors; applicative functors; functional programming.*

REFERENCES

1. Aizerman M. A. et al. *Logic, automations, and algorithms*. Moscow: Fizmatgiz Publishing; 1963. (In Russ.)
2. John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation: translation from English*. Moscow: Williams Publishing; 2002. (In Russ.)
3. A. Gill. *Introduction to the theory of finite-state machines*. Moscow: Nauka Publishing; 1966. (In Russ.)
4. Tom Stuart. *Computation theory for programmers: translation from English*. Moscow: DMK Press Publishing; 2014. (In Russ.)
5. Robert W. Sebesta. *Concepts of programming languages*. Moscow: Williams Publishing; 2001. (In Russ.)
6. H. Abelson, Gerald J. Sussman. *Structure and Interpretation of Computer Programs*. Moscow: Dobrosvet Publishing; 2018. (In Russ.)
7. Simon Peyton Jones. *The Implementation of functional programming languages*. Hemel Hempstead: Prentice Hall; 1987.
8. Anthony J. Field, P. Harisson. *Functional Programming*. Moscow: Mir Publishing; 1993. (In Russ.)
9. Moskvin D. N. *Course "Functional programming"*. Available from: https://nsk.compsciclub.ru/courses/func-prog/2020-spring/classes [Accessed 21 September 2020].
10. Sovetov B. Ia., Iakovlev S. A. *Systems modeling*. Moscow: Vysshaia shkola Publishing; 2001. (In Russ.)
11. Bartosz Milewski. *Category Theory for Programmers*. Available from: https://github.com/hmemcpy/milewski-ctfp-pdf/releases/download/v1.3.0/category-theory-for-programmers.pdf [Accessed 21 September 2020].
12. W. Kurt. *Programming with Haskell.* Moscow: DMK Press Publishing; 2019. (In Russ.)
13. McBride Conor, Paterson Ross. Applicative programming with effects. *J. Funct. Program*. 2008; 18(1): 1–13.
14. Paterson Ross. Constructing applicative functors. In: *Mathematics of Program Construction. Lecture Notes in Computer Science*. 2012; 7342: 300–323.
15. Gibbons Jeremy, Oliveira Bruno C. d. S. The essence of the Iterator pattern. *Journal of Functional Programming*. 2009; 19: 377–402.

Received 2 November 2020

Information about authors:

Eduard S. Lapin – DSc (Engineering), Professor, professor of Automation and Computer Technologies Department, Ural State Mining University. E-mail: lapin.eduard2014@yandex.ru
Marat I. Abdrakhmanov – PhD (Engineering), chief specialist, Ingortech company. E-mail: marat-ab@mail.ru

Для цитирования: Лапин Э. С., Абдрахманов М. И. Функциональный подход к моделированию динамики систем детерминированных конечных автоматов // Известия вузов. Горный журнал. 2021. № 2. С. 113–122. DOI: 10.21440/0536-1028-2021-2-113-122
For citation: Lapin E. S., Abdrakhmanov M. I. Functional approach to deterministic finite-state automata systems dynamic modeling. *Izvestiya vysshikh uchebnykh zavedenii. Gornyi zhurnal = News of the Higher Institutions. Mining Journal*. 2021; 2: 113–122 (In Russ.). DOI: 10.21440/0536-1028-2021-2-113-122