УДК 517.977
ББК 32.96

СОВМЕЩЕННЫЕ СЕТИ УПРАВЛЕНИЯ И ДАННЫХ

Выхованец В. С.[1]
(*ФГБУН Институт проблем управления им. В.А. Трапезникова РАН, Москва*)
Крыжановская А. В.[2]
(*ОАО «Яндекс», Москва*)

Рассмотрено динамическое управление процессами на основе использования совмещённых сетей управления и данных. Благодаря заданию на диаграмме процесса потока данных и потока управления появляется возможность автоматической генерации управляющей программы процесса. Для спецификации типов данных в совмещенной сети управления и данных разработана теория типов, позволяющая представлять данные в виде вложенных друг в друга именованных списков. Показано, что совмещенная сеть управления и данных является универсальной алгоритмической моделью со строгой типизацией данных. Доказано, что синтез такой сети по описанию выполняемого ею преобразования типов данных является алгоритмически неразрешимой проблемой. Использование совмещенных сетей управления и данных позволяет повысить качество управления технологическими, производственными и организационными процессами на предприятиях.

Ключевые слова: моделирование процессов, динамическое управление процессами, совмещенные сети управления и данных, алгоритмические модели, теория типов, неразреши-

---

[1] *Валерий Святославович Выхованец, доктор технических наук, доцент* (*valery@vykhovanets.ru*).
[2] *Александра Валерьевна Крыжановская, руководитель проектов* (*aleksundra@gmail.com*).

мые проблемы, универсальная алгоритмическая модель со строгой типизацией данных.

1. Введение

Процессно-ориентированный подход, заключающийся в представлении объекта управления как многоуровневого набора взаимосвязанных процессов, широко используется в современных системах технологического, производственного и организационного управления.

Входами процесса обычно являются выходы других процессов. Процесс описывается диаграммой, представляющей собой сеть, состоящую из процессов (узлов) и связей управления (дуг), задающих частичный порядок выполнения подчинённых процессов [11]. Данные процессов, как правило, не отражаются на диаграммах и связываются с процессами на этапе реализации с помощью атрибутов процесса, через специальные объекты внутри процесса, путем адресной передачи и приема данных через общую для всех процессов шину данных. Такая организация работы с данными исторически наследуется от самых первых методологий анализа и обусловлена тем, что выделение потока данных на этапе моделирования сложно реализовать и методически, и технически [5]. Отсутствие явно задаваемого потока данных приводит к невозможности автоматического создания программ управления процессами, а также затрудняют актуализацию уже запущенных программ.

Целью настоящей статьи является улучшение качества управления процессами путём использования такой их модели, которая основана на явной трассировке данных и управления между процессами. Это позволяет решить проблемы автоматической генерации управляющей программы по её описанию в виде иерархически организованных диаграмм процессов, а также улучшает выразительные возможности модели.

2. Проблемы моделирования процессов

Современное моделирование процессов возникло на основе методологии структурного анализа и проектирования.

2.1. СТРУКТУРНОЕ МОДЕЛИРОВАНИЕ

Методологии структурного анализа и проектирования SADT (Structured Analysis and Design Technique) [5] и стандарт IDEF0 (Integrated Computer-Aided Manufacturing DEFinition) [13] предназначены для описания процессов в функциональном аспекте: они состоят из диаграмм, текстов и глоссария, имеющих ссылки друг на друга.

Диаграммы – главные компоненты модели, все функции и интерфейсы в них представлены как узлы (функциональные блоки) и дуги. Каждый узел может быть декомпозирован на другой диаграмме. Следует отметить, что нотации в методологии SADT и IDEF0 предназначены для описания только состава процессов, а не их последовательности.

2.2. ПОТОКИ ДАННЫХ

Для частичного устранения недостатков IDEF0 используется описание потоков данных в нотации DFD (Data Flow Diagram) [15]. Цель такого представления – продемонстрировать, как каждый процесс преобразует свои входные данные в выходные. Основными компонентами диаграмм потоков данных являются внешние и внутренние сущности, процессы, накопители данных и потоки данных. Каждый процесс может быть детализирован при помощи другой диаграммы или спецификации. Языки спецификации, как правило, не стандартизованы и могут варьироваться от естественного языка до визуальных языков моделирования.

Дальнейшее развитие методологии IDEF, а именно стандартов IDEF1 (методология моделирования информационных потоков внутри системы) и IDEF1X (методология построения реляционных структур данных), преследовало цель устранить недостатки, присущие моделям потока данных.

2.3. ДИНАМИЧЕСКИЕ МОДЕЛИ

Модели в ранее рассмотренных нотациях являются статическими по определению и не позволяют отразить функционирование моделируемых объектов во времени. Проблему создания динамических диаграмм пытались решить с помощью IDEF2 (Simulation Model Design) – методологии динамического

моделирования развития. Однако ввиду сложности использования этой методологии разработка соответствующего стандарта быстро прекратилась.

Следующим шагом в моделировании процессов стала разработка и использование стандарта IDEF3 [12], предназначенного для описания потоков работ. Модели IDEF3 применяли для детализации функциональных блоков IDEF0, не имеющих диаграмм декомпозиции. Выразительные средства стандарта IDEF3 оказались близки алгоритмическим блок-схемам.

2.4. ПОТОКИ РАБОТ

Необходимость не только описать процессы, но и полностью или частично их автоматизировать привела к появлению концепции потока работ Workflow [14], согласно которой данные и задания передаются от одного исполнителя другому для выполнения определенных действий в соответствии со сводом процедурных правил.

Данные в Workflow не перемещаются вместе с управлением, а содержатся в глобально доступных переменных и локальных переменных блоков. Последнее вызывает значительные трудности контроля и синхронизации данных.

2.5. ИНТЕГРИРОВАННЫЕ МЕТОДОЛОГИИ

Неоднозначная семантика и низкая выразительность диаграмм в стандартах IDEF и DFD привела к объединению известных методов моделирования в форме создания интегрированных методологий, одной из которых является ARIS (Architecture of Integrated Information Systems) [11]. Модель процесса в ARIS является в некотором смысле расширением IDEF3 и представляет собой поток последовательно выполняемых работ, расположенных в порядке их выполнения.

Ввиду многообразия выразительных средств в ARIS внедрен механизм методологических фильтров, позволяющих в рамках одного процесса применять только определенный набор объектов и связей. Для разработки таких фильтров требуются значительное количество времени и разработчик высокой квалификации. В конечном итоге методология ARIS дает возможность описать процессы, провести анализ полученных моделей

и определить требования к управляющей информационной системе. Однако эта методология не обеспечивает полного, корректного и однозначного описания моделируемых процессов.

2.6. ДИНАМИЧЕСКОЕ УПРАВЛЕНИЕ

Негибкость создаваемых моделей, их неспособность обеспечить оперативное реагирование на постоянные изменения в среде являются основными недостатками методологий SADT, IDEF, DFD, ARIS и др., которые стимулировали разработку концепции следующего поколения – BPM (Business Process Management) [17].

В методологии BPM на смену радикальному реинжинирингу процессов приходит динамическое управление процессами, заключающееся в возможности корректировки уже автоматизированных процессов в ответ на изменения в структуре и организации предприятия. В этом случае инструменты моделирования позволяют создавать и внедрять новые процессы «на лету», а само моделирование больше похоже на моделирование в Workflow.

Для разработки исполняемых моделей в BPM используют нотацию BPMN (Business Process Model and Notation) и соответствующие языки моделирования: язык моделирования процессов BPML (Business Process Modeling Language), язык исполнения процессов BPEL (Business Process Execution Language) и язык определения процессов XPDL (XML Process Definition Language). Однако построение моделей непосредственно на этих языках неудобно для разработчиков по причине их синтаксической и семантической сложности. По этой причине наибольшую сложность для разработчиков программных средств в методологии BPM является конвертация моделей процессов в их исполняемые формы.

До настоящего времени применение BPEL в основном заключается в автоматизированном создании шаблона для «ручной доводки» модели до исполняемого вида. В частных случаях, когда доступны исполняемые формы всех используемых процессов, а также согласованы и специфицированы механизмы передачи между ними данных, возможно автоматическое созда-

ние исполняемой модели составного процесса. Однако для массового распространения методологии BPM пока не хватает технологических возможностей, позволяющих создавать и автоматически исполнять модели автоматизируемых процессов.

2.7. ВЫВОДЫ

Таким образом, блок-схема – распространенный вид графических моделей, описывающих алгоритмы или процессы, в которых отдельные шаги или подпроцессы изображаются в виде блоков, соединенных между собой линиями, указывающими последовательность их выполнения. Однако попытка реализации процессов с помощью блок-схем не увенчалась успехом, так как известные их нотации не обладают возможностями универсальных алгоритмических моделей.

По этой причине основной проблемой современных средств моделирования процессов является наличие труднопреодолимого семантического разрыва между графической (аналитической) и исполняемой формами процесса. Семантический разрыв проявляется в том, что понятия, объекты и структуры данных, которыми оперирует аналитик, не совпадают, а порой даже не согласуются с понятиями, объектами и структурами данных, которые должен использовать разработчик исполняемой модели процессов. Для преодоления указанного семантического разрыва требуется обеспечить безусловное автоматическое преобразование исходной графической модели процесса в его исполняемую форму.

3. Совмещенная сеть управления и данных

Появление проблем, свойственных современным средствам автоматизированного управления технологическими, производственными и организационными процессами, следует связать с недостаточностью данных, отражаемых на графических моделях процессов, их нечеткой пространственно-временной структурированностью и несогласованностью типов.

Для преодоления этих проблем предлагается использовать графические модели процессов, содержащие не только связи процессов по управлению, но и источники данных и пути их

передачи. Предлагается объединить выразительные возможности двух моделей – моделей для описания потоков работ и моделей для описания потоков данных [8, 19].

3.1. ГРАФИЧЕСКАЯ НОТАЦИЯ

Графическая нотация, основанная на представлении процесса в виде совмещенной сети управления и данных, позволяет явно указать на одной диаграмме как потоки управления, так и потоки данных, связанные с моделируемым процессом (рис. 1).

Через каждый узел сети (показан в виде прямоугольника) проходят поток управления и поток данных. Поток управления, входящий в узел, называется активатором (показан в виде ромба). Получение активации означает начало работы узла. Выход потока управления из узла называется событием (показан в виде квадрата). Событие, созданное узлом, может активировать один или несколько других узлов.

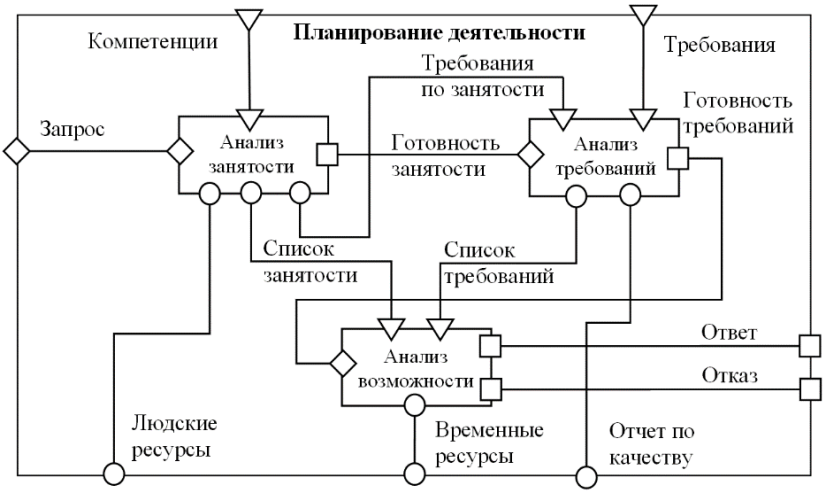

Рис. 1. Совмещенная сеть управления и данных

Потоки данных, входящие в узел (показаны в виде треугольников), передают узлу структурированные входные данные. После того как узел закончил работу, в исходящие потоки данных (показаны в виде окружностей) поступают структуриро-

ванные выходные данные, которые могут передаваться следующему узлу в качестве входных.

При срабатывании одного или нескольких активаторов узел активируется, считываются входные данные и выполняются действия, привязанные к этому узлу. Затем формируются одно или несколько событий и выходные данные.

Часть узлов, которые не имеют диаграммы, называются базовыми. Составной узел отличается от базового узла тем, что не является последним в иерархии, т.е. декомпозируется в совмещенную сеть и состоит из других узлов, каждый из которых, в свою очередь, может быть либо базовым, либо составным.

3.2. ИСПОЛНЕНИЕ СОВМЕЩЕННЫХ СЕТЕЙ

К каждому базовому узлу привязывается вычислительная семантика в виде последовательности действий (операций) на языке некоторого внешнего интерпретатора. В качестве интерпретаторов могут выступать компиляторы или интерпретаторы языков программирования, различные виртуальные машины, вызовы внешних программ с передачей им входных и получением от них выходных данных, какие-либо исполнительные устройства.

В свою очередь, составной узел строится из других узлов, а его вычислительная семантика однозначно определяется внутренними узлами и соединяющими их дугами.

Базовые узлы выполняются непосредственно тем интерпретатором, который определён во время их конструирования, а их вычислительная семантика задаётся явно. Исполнение составного узла – это исполнение всех его внутренних узлов.

При активации одного или нескольких активаторов составного узла в соответствии с его внутренними связями активизируются внутренние узлы. После окончания функционирования внутренних узлов их события по внутренним связям активизируют другие внутренние узлы. Функционирование составного узла завершается, когда все внутренние узлы перестают активироваться и заканчивают свое функционирование.

Аналогично активации происходит передача данных между внутренними узлами и формирование выходных данных составного узла.

3.3. СИНХРОНИЗАЦИЯ И РАСПАРАЛЛЕЛИВАНИЕ

Нотации многих известных методологий моделирования процессов включают в себя специальные элементы для явной (статической) организации их параллельного выполнения. Для этого, например, используется такие элементы как «логическое И», многоэкземплярные циклы, шлюз «ИЛИ» с неэксклюзивным условием и т.п. [11].

Моделирование процессов совмещёнными сетями управления и данных позволяет реализовать динамическое (неявное) распараллеливание, заключающееся в том, что в каждый конкретный момент времени и в зависимости от предыстории развития моделируемого процесса принимается решение о том, какие внутренние процессы подлежат параллельному исполнению, а какие нет.

Может оказаться так, что при одном и том же описании процесса в один момент времени параллельному исполнению подлежит одно подмножество внутренних процессов, а в другой момент времени – другое.

Таким образом, совместная и раздельная трассировка данных и управления между процессами предоставляет гибкий механизм синхронизации моделируемых процессов, позволяет реализовать естественное их распараллеливание и эффективное выполнение.

4. Типизация данных

В известных подходах к моделированию процессов структуры и типы данных возникают только на заключительном этапе реализации модели и входят в зону ответственности разработчика исполняемой программы.

Использование явной трассировки данных на диаграммах процессов требует применения достаточно развитого аппарата структуризации, типизации, объединения, сравнения, разделения и контроля данных на всех этапах работы с моделью процесса. По этой причине рассмотрим формальную теорию типов, необходимую для описания потоков данных в совмещенных сетях.

4.1. ТЕОРИЯ ТИПОВ

Зададим алфавит теории типов T, состоящий из конечного множества знаков простых типов $\Omega = \{\tau_1, \tau_2, \ldots, \tau_m\}$ и фигурных скобок. Определим перечислимое множество формул теории следующим образом:

– знак простого типа данных $\tau \in \Omega$ является типом;
– если φ – тип, то и $\{\varphi\}$ является типом;
– если φ и ψ – типы, то и $\varphi\psi$ – тип.

Выводом в теории T является порождение нового типа из имеющихся типов-посылок с помощью правил вывода вида

(1) $\varphi \Rightarrow \{\varphi\}$;

(2) $\varphi, \psi \Rightarrow \varphi\psi$,

где $\Rightarrow$ – знак вывода (подстановки).

Правило (1) задает специализацию типа, а правило (2) – агрегацию типов. При специализации из исходного типа создается новый тип, рассматриваемый как выделенный. При агрегации из двух типов создается новый тип, включающий в себя агрегируемые типы как составные части – подтипы.

Пример 1. Пусть задано множество простых типов $\Omega = \{b\}$, где b – логический тип, принимающий значения на множестве из двух элементов $\{0, 1\}$. Тогда с помощью правил вывода могут быть получены следующие типы:

$c = \underbrace{bb\ldots b}_{16} = b^{16}$ – тип символа с разрядностью 16;

$i = b^{32}$ – целочисленный тип с разрядностью 32;

$f = b^{53}\{b^{11}\}$ – тип чисел с плавающей запятой, где 53 разряда отводится под мантиссу, а 11 разрядов – под порядок числа;

$s = c^{256}$ – строковый тип с длиной строк 256 знаков;

$p = b^{1024}$ – тип битовых последовательностей данных с длиной 1024 разряда. ♦

Как видно из примера 1, для порождения множества типов достаточно одного простого типа.

Аксиоматика теории типов состоит из следующих аксиом:

(3) $\exists\tau\,(\tau = \tau)$;

(4) $\forall\varphi\,\forall\psi\,\exists\chi\,(\chi=\varphi\psi)$;

(5) $\forall\varphi\,\forall\psi\;\;(\psi\varphi=\varphi\psi)$;

(6) $\forall\varphi\,\forall\psi\;\;(\varphi\psi>\varphi\wedge\varphi<\varphi\psi)$;

(7) $\forall\varphi\,\exists\chi\,(\chi=\{\varphi\}\wedge\neg(\chi=\varphi)\wedge\neg(\chi>\varphi)\wedge\neg(\chi<\varphi))$,

где для выражения аксиом использовано исчисление предикатов с кванторами существования $\exists$ и всеобщности $\forall$, логической связкой «И» $\wedge$, логической связкой «НЕ» $\neg$, а также двуместными предикатами сравнения типов: равно $=$, больше $>$, меньше $<$. Круглые скобки использованы для выделения высказываний, на которые действуют стоящие перед ними кванторы.

Аксиома существования (3) утверждает, что существует хотя бы один тип, который равен самому себе. В свою очередь аксиома агрегации (4) утверждает существование типов-агрегатов и порождает натуральные числа вида τ, $\tau\tau$, $\tau\tau\tau$ и так далее. Аксиома равенства (5) разделяет типы на классы эквивалентности, а аксиома порядка (6) задает некоторый порядок типов. И, наконец, аксиома специализации (7) позволяет выделить (специализировать) любой тип таким образом, что он становится не равным себе и не сравнимым с собой. По своей сути аксиома специализации утверждает существование структурированных типов.

4.2. ОСНОВНЫЕ ТЕОРЕМЫ

Определение 1. Два типа φ и ψ называются несравнимыми, если и только если неверно, что они равны, неверно, что φ больше ψ, и неверно, что φ меньше ψ:

$$\varphi\#\psi\leftrightarrow\neg(\varphi=\psi)\wedge\neg(\varphi>\psi)\wedge\neg(\varphi<\psi),$$

где # – двуместный предикат несравнимости типов; $\leftrightarrow$ – логическая связка двухстороннего следования.

Теорема 1. Существуют несравнимые типы, т.е.

$$\exists\varphi\exists\psi(\neg(\varphi>\psi)\wedge\neg(\varphi<\psi)\wedge\neg(\varphi=\psi)).$$

Доказательство. Пусть $\varphi = \tau$. Существование типа φ следует из аксиомы (3). Тогда из аксиомы (7) следует существование типа $\psi = \{\tau\}$ такого, что $\neg(\varphi=\psi)$, $\neg(\varphi>\psi)$ и $\neg(\varphi<\psi)$. ♦

Теорема 2. Любая перестановка подтипов φ_i (i = 1, …, N) типа $\varphi = \varphi_1\varphi_2 \ldots \varphi_N$ является тождественным преобразовани-

ем этого типа, т.е. для любой последовательности неповторяющихся индексов $i, j, \ldots, k$ таких, что $1 \le i, j, \ldots, k \le N$ и $i \ne j \ne \ldots \ne k$ выполнимо $\varphi_1\varphi_2 \ldots \varphi_N = \varphi_i\varphi_j \ldots \varphi_k$.

Доказательство. Пусть $\varphi = \varphi_1 \ldots \psi \ldots \varphi_N$, где $\psi = \varphi_t\varphi_{t+1}$, $1 \le t < N$. Из аксиомы (5) находим, что $\psi = \varphi_{t+1}\varphi_t$. Следовательно, перестановка двух соседних подтипов является тождественным преобразованием типа. Пусть теперь $i \ne j \ne \ldots \ne k$ и $1 \le i, j, \ldots, k \le N$. Тогда $(i, j, \ldots, k)$ есть перестановка чисел от 1 до N. Известно, что любая перестановка может быть получена путем перестановки соседних элементов. Отсюда получаем утверждение теоремы. ♦

Теорема 3. Два типа φ и ψ равны тогда и только тогда, когда существует такая перестановка подтипов типа φ, которая преобразует его в тип ψ.

Доказательство. В соответствии с аксиомой (4) представим типы φ и ψ как агрегации одинакового числа подтипов: $\varphi = \varphi_1\varphi_2 \ldots \varphi_N$ и $\psi = \psi_1\psi_2 \ldots \psi_N$. Это можно сделать всегда и, возможно, многими способами.

Пусть типы φ и ψ равны. Тогда из теоремы 2 следует, что существует агрегация подтипов этих типов и перестановка $(i, j, \ldots, k)$ подтипов типа φ такие, что $\psi_1 = \varphi_i$, $\psi_2 = \varphi_j$, …, $\psi_N = \varphi_k$. Доказано прямое утверждение теоремы.

Верно и обратное утверждение. Пусть существует перестановка $(i, j, \ldots, k)$ подтипов типа φ такая, что $\psi_1 = \varphi_i$, $\psi_2 = \varphi_j$, …, $\psi_N = \varphi_k$. Тогда в соответствии с теоремой 2 типы φ и ψ равны. Что требовалось доказать. ♦

Следует обратить внимание на рекурсивный характер определения равенства типов. Если некоторый подтип φ_i является специализацией других подтипов, то соответствующий ему тип ψ_j также должен быть специализацией, а сравнение таких подтипов должно осуществляться отдельно и независимо.

Пример 2. Пусть заданы простые типы $\Omega = \{b, c, i, f, s, p\}$. Два типа $\varphi = f^2s\{cbi\}p$ и $\psi = pfs\{bic\}f$ равны, так как подтип $\{bic\}$ преобразуется в подтип $\{cbi\}$ перестановкой (2, 3, 1), а тип ψ преобразуется в тип φ перестановкой (2, 5, 3, 4, 1). ♦

Таким образом, все типы теории типов T разбивается на классы эквивалентности. В один класс эквивалентности попа-

дают типы, заданные с точностью до перестановки агрегированных в них подтипов.

Теорема 4. Тип φ меньше типа ψ тогда и только тогда, когда существует такой тип ω, что $\psi = \varphi\omega$ или $\psi = \omega\varphi$.

Доказательство. Пусть $\varphi < \psi$. Тогда из аксиомы (6) следует существование некоторого типа ω такого, что $\psi = \varphi\omega$, а из аксиомы (5) выводится $\psi = \omega\varphi$. Пусть теперь $\psi = \varphi\omega$ или $\psi = \omega\varphi$. Из аксиом (6) и (5) следует $\varphi < \varphi\omega$ или $\varphi < \omega\varphi$. Откуда получаем $\varphi < \psi$. ♦

Пример 3. Пусть задано множество простых типов из примера 2. Тогда тип $\varphi = fs\{cbi\}p$ меньше типа $\psi = pfs\{bic\}f$, так как из тождественных им типов $fsp\{cbi\}$ и $fsp\{cbi\}f$ находим, что $\psi = \varphi f$. ♦

В итоге имеем, что типы частично упорядочены и не существует наименьшего и наибольшего типа.[1] На типах может быть задана иерархия (рис. . 2), аналогичная иерархии типов в логических теориях [9] и диаграммам наследования классов в объектно-ориентированном программировании [1].

Значением типа является значение переменной, имеющей этот тип. Для получения значения типа введем операцию разыменования, которую будем обозначать квадратными скобками.

[1] *Не существование минимального типа следует из теоремы 1. Это позволяет не ограничиваться одним простым типом при построении прикладной теории типов. В свою очередь не существование наибольшего типа выводится из аксиомы агрегации и аксиомы порядка методом от противного. Это позволяет рассматривать теорию типов как конечную* (*финитную, конструктивную*) *теорию и, тем самым, избежать проблем, связанных с бесконечными построениями.*

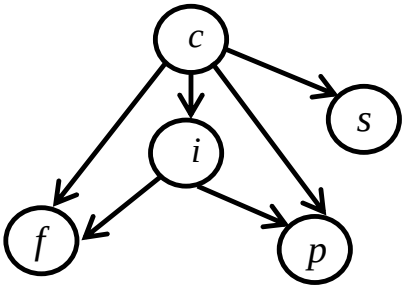

Рис. 2. Иерархия типов из примера 1

Пример 4. Пусть задан тип $c^3s\{bi\}$ над множеством простых типов из примера 2. Тогда значением $[c^3s\{bi\}]$ этого типа будет структура, состоящая из массива трех переменных с типом *c*, переменной типа *s* и структуры *bi*, состоящей, в свою очередь, из переменной типа *b* и переменной типа *i*.

Аналогично можно разыменовывать не весь тип, а некоторый его подтип, например, $c^3[s]\{bi\}$ – выделение из переменной типа $c^3s\{bi\}$ значения агрегированной в нее переменной типа *s*. ♦

4.3. РАЗРЕШИМОСТЬ ТЕОРИИ ТИПОВ

При моделировании процессов совмещенными сетями управления и данных возникает две задачи, связанные с типами: на этапе моделирования процесса – это проверка допустимости трассировки данных от одного узла сети к другому, а на этапе выполнения процесса – это проверка достаточности поступивших данных. Так как значение большего типа может быть преобразовано в значение меньшего типа, то обе эти задачи сводятся к простому сравнению типов.

Теорема 5. Сравнение типов разрешимо, т.е. для любых двух типов φ и ψ конструктивными средствами может быть установлена истинность одного и только одного из следующих высказываний: $\varphi \# \psi$, $\varphi = \psi$, $\varphi > \psi$, $\varphi < \psi$.

Доказательство. Будем рассматривать произвольный тип как конечную строку в алфавите, состоящем из знаков простых типов и фигурных скобок. Распознавание строки,

выражающей произвольный тип, как и любого скобочного выражения, выполняется магазинным автоматом [7, с. 251]. Следовательно, распознавание типов, выводимых в теории типов, разрешимо.

Пусть $\varphi = \varphi_1\varphi_2 \ldots \varphi_N$ и $\psi = \psi_1\psi_2 \ldots \psi_M$, где $N \le M$, а выделенные подтипы – это простые или специализированные подтипы, отсортированные в лексикографическом порядке. Последнее возможно ввиду того, что лексикографическая сортировка строки типа является его тождественным преобразованием (теорема 3) и задача сортировки строки разрешима, например, методом «пузырьковой» сортировки [3, с. 144], для которого может быть построена соответствующая машина Тьюринга [7, с. 328].

Тогда каждый подтип φ_i ($i = 1, \ldots, N$) будем проверять на равенство соответствующему ему подтипу ψ_i. Если подтип ψ_i не равен φ_i, то $\varphi \# \psi$. Если ψ_i равен φ_i, то переходим к следующему подтипу φ_{i+1}. Если все подтипы φ_i просмотрены, а в типе ψ еще остались непроверенные подтипы ψ_j ($j > N$), то $\varphi < \psi$. В противном случае $\varphi = \psi$.

Существование и единственность результата сравнения типов непосредственно следует из описания процедуры сравнения. Очевидно, для этой процедуры может быть построена машина Тьюринга. Следовательно, сравнение типов разрешимо. ♦

Таким образом, декомпозицию типов на подтипы и поиск перестановок подтипов, переводящих один тип в другой (условие теоремы 3), можно осуществить путем сортировки простых и специализированных подтипов в некотором лексикографическом порядке с последующим их поэлементным сравнением.

Пример 5. После сортировки подтипов типов φ и ψ из примера 2 в лексикографическом порядке, задаваемом, например, порядком перечисления простых типов во множестве Ω, имеем $\varphi = f^2ps\{bci\}$ и $\psi = f^2ps\{bci\}$. Откуда находим, что типы φ и ψ равны. ♦

Для представления именованных списков вида (Имя, Значение 1, Значение 2, …), где любым из значений может быть другой именованный список, во множество простых типов Ω включается специальный тип α. Тогда любой именованный тип

φ задается как $\{\alpha^x\varphi\}$, где $x > 1$ – индекс имени типа, а неименованный тип – как $\{\alpha\varphi\}$.

5. Теория совмещенных сетей

Для определения выразительных возможностей совмещенных сетей управления и данных при моделировании процессов рассмотрим их формальную теорию, наследуемую от ранее описанной теории типов.

5.1. ФОРМАЛИЗМ СОВМЕЩЕННЫХ СЕТЕЙ

Определение 2. Совмещенной сетью управления и данных S называется упорядоченное множество, состоящее из четырех подмножеств N, E, D и T:

(8) $S = (N, E, D, T)$,

где $N = \{n_i \mid i = 0, \ldots, p\}$ – множество внутренних узлов сети с числом элементов p, куда включен внешний интерфейсный узел n_0; $E = \{(n_i, n_j, t_k) \mid n_i, n_j \in N; t_k \in T\}$ – множество дуг управления; $D = \{(n_i, n_j, t_k) \mid n_i, n_j \in N; t_k \in T\}$ – множество дуг данных; T – множество типов данных; дуги заданы упорядоченными множествами из трех элементов $(n_i, n_j, t_k) \in N \times N \times T$, где $n_i \in N$ – начальный узел дуги, $n_j \in N$ – ее конечный узел, $t_k \in T$ – тип дуги, $\times$ – операция декартова произведения множеств.

Интерфейсный узел n_0 имеет специальное назначение (рис. 3). Дуги управления (начинаются ромбом), идущие из него, являются активаторами сети, а входящие в него – событиями. В свою очередь дуги данных (начинаются кругом), связанные с этим узлом, задают входные и выходные данные. При включении совмещенной сети в другую сеть в качестве узла она замещается своим интерфейсным узлом.

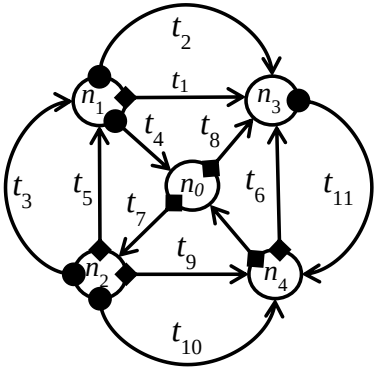

Рис. 3. Пример совмещенной сети управления и данных

Пусть задан узел n некоторой совмещенной сети, имеющий r активаторов $a_1, a_2, \ldots, a_r$ и s входных данных $i_1, i_2, \ldots, i_s$, а также генерирующий u событий $e_1, e_2, \ldots, e_u$ и v выходных данных $o_1, o_2, \ldots, o_v$. Тогда этот узел на внешнем уровне может быть описан как функция

(9) $n([\{\alpha a_1 a_2 \ldots a_r\}\{i_1 i_2 \ldots i_s\}]) \rightarrow [\{\alpha e_1 e_2 \ldots e_u\}\{o_1 o_2 \ldots o_v\}]$

с аргументами типа $\{\alpha a_1 a_2 \ldots a_r\}\{i_1 i_2 \ldots i_s\}$, которая возвращает результат типа $\{\alpha e_1 e_2 \ldots e_u\}\{o_1 o_2 \ldots o_v\}$ при каждом своем запуске (вычислении). Специальный простой тип α здесь использован для того, чтобы отличить аргументы управления от аргументов данных.

Определение 3. Выражение (9) называется интерфейсным описанием сети (узла сети).

Следует обратить внимание на то, что обычно на практике дуги управления имеют логический тип. Однако в соответствии с (8) и (9) нет никаких препятствий рассматривать эти дуги как пути передачи структурированных управляющих данных.

5.2. НЕРАЗРЕШИМЫЕ ПРОБЛЕМЫ

Несмотря на простоту формализма и доказанных ранее теорем, в рассматриваемой теории совмещенных сетей управления и данных имеются и неразрешимые проблемы.

Теорема 6. Задача построения совмещенных сетей по их интерфейсному описанию является алгоритмически неразрешимой.

Доказательство. Пусть интерфейсное описание совмещенной сети задано формулой (9). Задача построения сети по ее интерфейсному описанию в качестве элементарной операции будет содержать трассировку типизированных данных между узлами строящейся сети. Отсюда, не нарушая общности, сведем задачу построения искомой сети к построению некоторого типа выходных данных

$$\Lambda = \{e_1 e_2 \ldots e_u\}\{o_1 o_2 \ldots o_v\}$$

из подтипов, принадлежащих типу некоторых входных данных Δ:

$$\Delta = \{a_1 a_2 \ldots a_r\}\{i_1 i_2 \ldots i_s\}.$$

Представим типы Λ и Δ, их подтипы, а также подтипы специализированных подтипов и т.д., как списки строк конечной длины Y и X соответственно, отсортированные в лексикографическом порядке. Например, для типа

$$\{ba\{b^2a\{a^2b\}\}\}\{\{a^2\}b\{ba\}\}$$

строится следующий список подтипов длины 6:

$$(ab,\ ab^2,\ a^2b,\ b,\ a^2,\ ab).$$

Для выравнивания длины списков Y и X (возможно, и суммарной длины их строк), в каждый из списков добавим некоторое конечное число строк произвольных типов. При дополнении списков будем учитывать возможные преобразования входных типов данных (использовать подстроки исходных типов данных), а также типы и подтипы данных внутренней памяти конструируемой сети, если таковая имеется. Пусть длина этих списков станет равна L.

Тогда задача построения искомой сети эквивалентна комбинаторной проблеме Поста: поиску такой последовательности неповторяющихся индексов $i, j, \ldots, k$ $(1 \le i, j, \ldots, k \le L)$, для которых совпадут строки $x_1 x_2 \ldots x_L$ и $y_i y_j \ldots y_k$, где $x_1, x_2, \ldots, x_L$ и $y_i y_j \ldots y_k$ – строки списков X и Y соответственно. Известно, что комбинатор-

ная проблема Поста неразрешима [16].[1] Следовательно, не существует алгоритма, который по заданному интерфейсному описанию узла сможет согласовать аргументы внутренних и внешних узлов сети и, в конечном итоге, построить сеть, реализующую это описание. Что требовалось доказать. ♦

На содержательном уровне теорема 6 утверждает, что в формализме совмещенных сетей управления и данных не существует алгоритма синтеза программ, т.е. не существует универсального алгоритма, который по описанию преобразования данных на языке теории типов строит совмещенную сеть, реализующую это преобразование.

Также следует заметить, что построить искомую сеть не помогает и дополнение множеств исходных и результирующих типов любым конечным множеством производных типов.

Может показаться, что теорема 6 является следствием низких выразительных качеств совмещенных сетей управления и данных. Однако это не так.

5.3. УНИВЕРСАЛЬНАЯ АЛГОРИТМИЧЕСКАЯ МОДЕЛЬ

Известно несколько алгоритмических моделей, устанавливающих эквивалентность между неформальным и формальным определением алгоритма. Основной результат общей тории алгоритмов [2] – это эквивалентность универсальных алгоритмических моделей. Так, формализм частично-рекурсивных функций [4] эквивалентен другим алгоритмическим моделям: машине Тьюринга [18], нормальным алгоритмам Маркова [6], лямбда-исчислению Черча [10].

Теорема 7. Всякая частично-рекурсивная функция вычислима совмещенной сетью управления и данных.

---

[1] *Здесь использовано следствие из комбинаторной проблемы Поста, согласно которому из неразрешимости задачи построения индексов* $1 \le i, j, \ldots, k \le L$ *для обеспечения равенства строк* $x_i x_j \ldots x_k$ *и* $y_i y_j \ldots y_k$ *для любых списков X и Y достаточно большой длины следует и неразрешимость этой задачи для строк* $x_1 x_2 \ldots x_L$ *и* $y_i y_j \ldots y_k$, *так как списки X и Y предполагаются произвольными.*

Доказательство. Для доказательства теоремы необходимо показать, что константа 0, функция следования $f(x) = x + 1$ и функции тождества

$$I_m^n(x_1, x_2, \ldots, x_n) = x_m \ (m = 1, \ldots, n),$$

а также оператор суперпозиции

$$S_m^n(h, g_1, g_2, \ldots, g_m) = h(g_1, g_2, \ldots, g_m),$$

оператор примитивной рекурсии:

$$\begin{cases} R_n(x_1, x_2, \ldots, x_n, 0) = g(x_1, x_2, \ldots, x_n), \\ R_n(x_1, x_2, \ldots, x_n, y+1) = h(x_1, x_2, \ldots, x_n, y, R_n(x_1, x_2, \ldots, x_n, y)); \end{cases}$$

и неограниченный оператор минимизации

$$\mu_y(P(x_1, x_2, \ldots x_n, y)) = \min y \mid P(x_1, x_2, \ldots x_n, y)$$

реализуются совмещенными сетями управления и данных, где $y, x_1, x_2, \ldots, x_n$ – переменные; $h, g_1, g_2, \ldots, g_m$ – функции; P – предикат (функция, возвращающая логическое значение).

Реализация перечисленных выше функции и операторов совмещенными сетями приведена на рис. 4, 5, 6 и 7, где все дуги помечены типом натуральных чисел произвольной сколь угодно большой разрядности и использованы вспомогательные базовые блоки:

– мультиплексирования данных M:

$$M(a_1, a_2, x_1, x_2) = \begin{cases} x_1, \text{ если } a_1, \\ x_2, \text{ если } a_2; \end{cases}$$

– коммутации управления K ($x \neq 0$):

$$z = \begin{cases} 0, \text{ если } x = 0, \\ 1, \text{ если } x \neq 0; \end{cases} \quad \begin{cases} e_1, \text{ если } x = 0, \\ e_2, \text{ если } x \neq 0; \end{cases}$$

и один составной блок C – счетчик со сбросом:

$$y = \begin{cases} 0, \text{ если } a_1, \\ y+1, \text{ если } a_2; \end{cases}$$

определенные на рис. 8. Из приведенных рисунков непосредственно следует утверждение теоремы. ♦

Теорема 7 утверждает, что совмещенные сети управления и данных имеют выразительные возможности универсальных алгоритмических моделей, однако отличаются от последних

наличием строгой типизации данных и полностью графической формой описания.

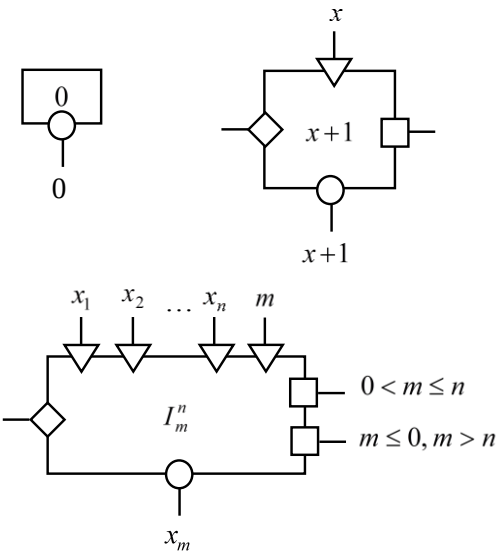

Рис. 4. Элементарные функции

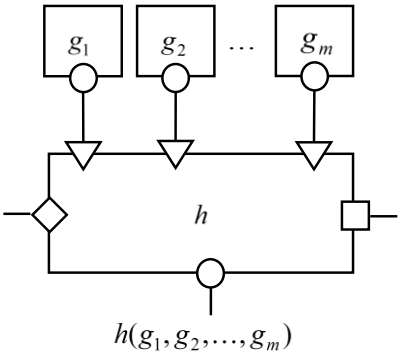

Рис. 5. Суперпозиция функций

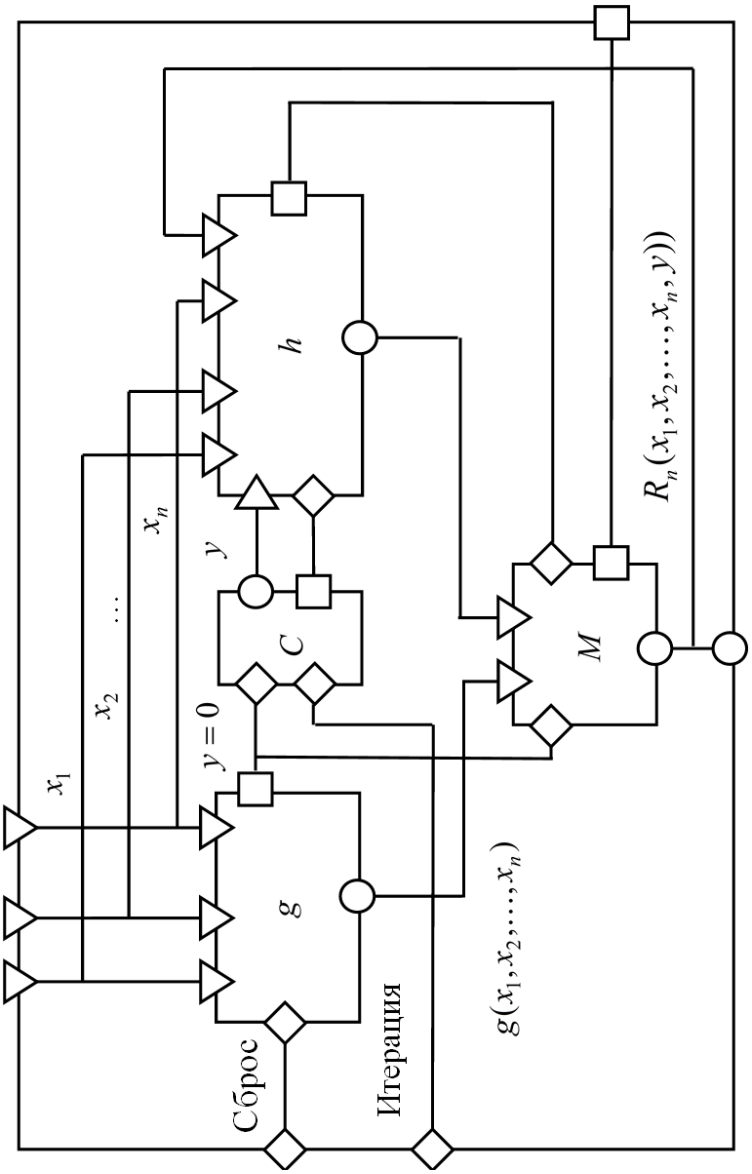

Рис. 6. Примитивная рекурсия

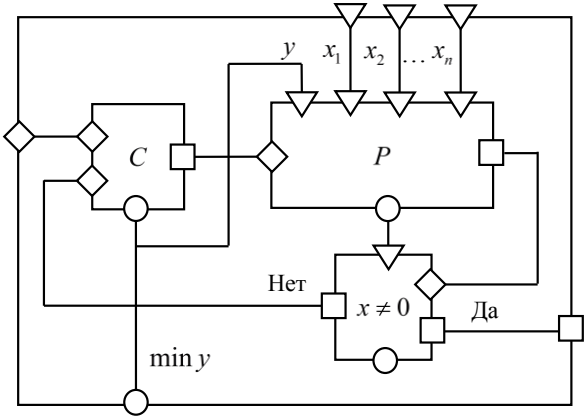

Рис. 7. Оператор минимизации

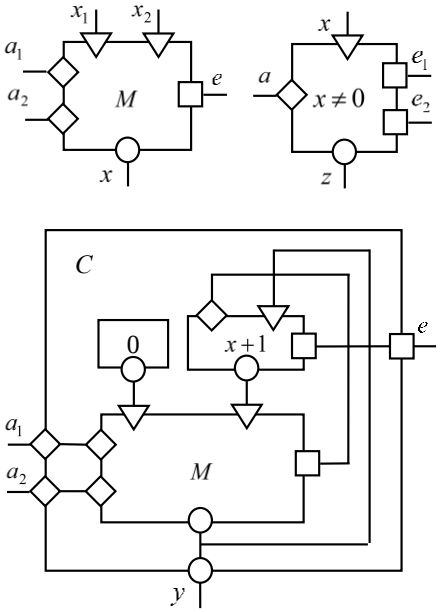

Рис. 8. Вспомогательные операторы:
мультиплексор M, коммутатор K ($x \neq 0$)*, счетчик C*

6. Заключение

Применение совмещенных сетей управления и данных позволяет преодолеть накопившиеся проблемы в области динамического управления процессами. В частности, на основе графической модели процесса в нотации совмещенных сетей управления и данных появляется возможность автоматически создавать управляющую программу процесса, а использование строгой типизации данных позволяет выполнить эффективную верификацию создаваемых моделей.

Совместная трассировка данных и управления между узлами сети предоставляет гибкий механизм синхронизации моделируемых процессов, обеспечивая естественное их распараллеливание в среде исполнения. Реализуемая при этом параллельность близка к естественному параллелизму моделируемых процессов.

С теоретической точки зрения, совмещенные сети управления и данных являются универсальной алгоритмической моделью со строгой типизацией данных. Для сравнения типов данных найдена эффективная процедура, сводимая к операциям над строками конечной длины. Для этой алгоритмической модели также доказано, что синтез программ является одной из неразрешимых проблем общей теории алгоритмов.

Как и у других алгоритмических моделей, неразрешимыми проблемами для совмещенных сетей управления и данных является распознавание любых нетривиальных свойств вычислимых функций и, в частности, распознавание по описанию процесса в виде совмещенной сети остановится ли этот процесс при заданных входных данных или не остановится (проблема остановки).

Литература

1. БУЧ Г., РАМБО Д., ДЖЕКОБСОН А. *Язык UML: Руководство пользователя*. – М., СПб.: ДМК «Пресс Питер», 2004. – 432 с.
2. КЛИНИ С.К. *Введение в метаматематику*: Пер. с англ. – М., 1957. – 526 с.
3. ЛЕВИТИН А.В. *Алгоритмы: введение в разработку и анализ*. – М.: Вильямс, 2006. – 576 с.

4. МАЛЬЦЕВ А.И. *Алгоритмы и рекурсивные функции*. – М.: Наука, 1986. – 366 с.
5. МАРКА Д., МАКГОУЭН К. *Методология структурного анализа и проектирования SADT*. – М.: Метатехнология, 1993. – 247 с.
6. МАРКОВ А.А. *Теория алгорифмов*. – М.: Изд-во АН СССР, 1954. – 376 с.
7. ХОПКРОФТ Д., МОТВАНИ Р., УЛЬМАН ДЖ. *Введение в теорию автоматов, языков и вычислений*: Пер. с англ. – М.: Издательский дом Вильямс, 2002. – 528 с.
8. ЯЦУТКО А. В., ВЫХОВАНЕЦ В.С. *Динамическое управление бизнес-процессами на основе совмещенных сетей управления и данных* // Инженерный журнал: наука и инновации. – 2013. – №2(14). – [Электронный ресурс] – URL: http://engjournal.ru/catalog/mathmodel/social/530.html. (дата обращения 20.11.2015).
9. CHURCH A. *A formulation of the simple theory of types* // The Journal of Symbolic Logic. – 1940. – Vol. 5, No.2. – P. 56–68.
10. CHURCH A. *A set of postulates for the foundation of logic* // Annals of Mathematics. – 1932. – Series 2, No. 33. – P. 346–366.
11. DAVIS R. *Business Process Modeling with ARIS: A Practical Guide*. – New York: Springer-Verlag, 2005. –531 p.
12. *Information integration for concurrent engineering* (*IICE*) / R.J. Mayer, C.P. Menzel, M.K. Painter, et al. // IDEF3 Process Description Capture Method Report. – Ohio: Air Force Material Command, 1995. – 235 p.
13. *Integration definition for function modeling* (*IDEF0*). – Washington: National Institute of Standards and Technology, 1993. – 116 p.
14. FISCHER L. *Workflow handbook 2002*. – Association with the Workflow Management Coalition (WfMC), 2002. – 428 p.
15. GANE C., SARSON T. *Structured systems analysis: Tools and techniques*. – New York: Prentice-Hall, 1979. –452 p.
16. POST E.L. *A variant of a recursively unsolvable problem* // Bulletin of American Mathematical Society. – 1946. – Vol. 52, No 4. – P. 264–268.
17. SMITH H., FINGAR P. *Business process management* (*BPM*): *The Third Wave*. – Meghan-Kiffer Press, 2002. – 312 p.

18. TURING A.M. *Computability and λ-Definability* // Journal of Symbolic Logic. – 1937. – Vol. 2, No. 4. – P. 153–163.
19. VYKHOVANETS V., YATSUTKO A. *Dynamic business process management based on the combined control and data networks* // Preprints of the 2013 IFAC Conference on Manufacturing Modelling, Management, and Control. Saint Petersburg, Russia, June 19–21, 2013. – P. 672–677.

COMBINED NETWORKS OF CONTROL AND DATA

Valeriy Vykhovanets, Institute of Control Sciences of RAS, Moscow, Dr. Sc., associative professor (valery@vykhovanets.ru).
Aleksandra Kryzhanovskaya, Yandex Company, Moscow, project manager (aleksundra@gmail.com).

Abstract: An approach is considered to dynamic process management based on the use of combined networks of control and data. Drawing a data flow and a control flow in a joint process diagram allows automatically generating the executable code for process control. A theory of types is developed to care for data types. The data are stored in the form of nested named lists. A combined network of control and data is shown to be a universal algorithmic model with strong data typing. Synthesis of a network for the given transformation of data types is proved to be an algorithmically unsolvable problem. The use of combined networks of control and data can improve the quality of control in technological, manufacturing and organizational processes.

Keywords: process modeling, dynamic control of processes, combined networks of control and data, algorithmic models, theory of types, unresolved problems, universal algorithmic model of strong typing data.

Статья представлена к публикации членом редакционной коллегии М.Ф. Караваем

Поступила в редакцию 14.02.2015.
Опубликована 30.11.2015.