*Д.С. Тарасов, Н.И. Акберова*
*Казанский государственный университет, Казань*
*Natasha.Akberova@ksu.ru*

# ВИРТУАЛЬНЫЕ МАШИНЫ ДЛЯ ИССЛЕДОВАНИЯ МОЛЕКУЛЯРНО-БИОЛОГИЧЕСКИХ ПРОЦЕССОВ

Возможность использования виртуальных машин для исследования функциональной организации живых клеток проистекает из того факта, что живая клетка может рассматриваться в качестве вычислительного устройства, и ее логическая архитектура может изучаться отдельно от физической структуры. Виртуальные машины (ВМ), повторяющие логическую организацию живой клетки, могут быть использованы для моделирования регуляции биохимических процессов, а также служить в качестве целевой архитектуры для специализированных языков моделирования клеточного метаболизма. Использование ВМ также может улучшить возможности программ автоматической аннотации генома и способствовать углублению понимания логической организации живых систем. В данной работе представлена спецификация для новой виртуальной молекулярно-биохимической машины, прототип ее реализации, а также графический язык для ее программирования. Приводится обсуждение, как представленная ВМ может быть использована для программирования молекулярно-биологических процессов.

## 1. Введение

### Что такое виртуальная машина?

Термин «Виртуальная машина» (ВМ) употребляется в двух смыслах. Первое значение: «Изолированный дубликат реально существующей вычислительной системы, в котором большинство инструкций виртуального процессора может быть выполнено на процессоре-хозяине» (Goldberg, 1975). Второе значение: «Абстрактная спецификация вычислительного устройства, которая может быть реализована различными способами аппаратно или программно». Основным различием в этих определениях является то, что во втором случае вычислительная система, которую эмулирует ВМ, может не существовать в аппаратной реализации.

В чем польза виртуальных машин для молекулярной биологии? В следующих разделах обсуждаются цели создания виртуальных машин в вычислительной технике и то, как эти цели прилагаются к молекулярной биологии.

### Виртуальные машины для обеспечения кросс-платформенной совместимости

Виртуальные машины изучаются с 1960 года, в основном, для целей достижения совместимости между различными аппаратными архитектурами.

Поскольку живые клетки могут рассматриваться в качестве вычислительного устройства (Ji, 1999), виртуальные машины могут стать важным инструментом в молекулярной биологии. Фактически программные комплексы моделирования клеточного метаболизма, основанные на моделях химической кинетики, такие как E-CELL (Tomita et al., 1999) и Cellerator (Shapiro et al., 2003) являются виртуальными машинами, созданными для того, чтобы имитировать поведение клетки на цифровом компьютере. Они предоставляют виртуализацию за счет моделирования физической (аппаратной) организации живой клетки.

Однако с самого начала исследований в области виртуальных машин стало очевидно, что аппаратная архитектура (машина с точки зрения инженера) и логическая организация (машина с точки зрения программиста) совершенно различны (Amdahl et al., 1964). Точное моделирование физического устройства системы для эмуляции поведения вычислительного устройства является избыточным и приводит к ненужным усложнениям. Таким образом, чем больше нам известно о логической организации живой клетки, тем лучше мы способны понять ее поведение без использования точных физико-химических моделей.

Логическая организация живой клетки была предметом нескольких теоретических исследований (Regev et al., 2004; Cardelli, 2005), однако работы в данном направлении находятся в зачаточном состоянии. Для того чтобы найти хорошее описание логической структуры живой клетки, необходимо поставить значительное количество вычислительных экспериментов с различными реализациями различных теоретических моделей.

### Виртуальные машины для реализации языков программирования

Для реализации языков программирования использование виртуальных машин позволяет получить ряд преимуществ, помимо кросс-платформенной совместимости. Виртульные машины, такие как SmallTalk VM (Goldberg et al., 1983) и Java VM (Lindholm et al., 1999) облегчают реализацию специфических возможностей языков, таких как динамическая компиляция и позднее связывание. Реали-

---

*Окончание статьи Г.М. Фаздаловой «Аналитический учет...»*

N – необходимое количество учетных регистров, m – количество учитываемых видов продукции, p – количество учитываемых видов затрат на выбранные виды продукции, s – количество мест возникновения учитываемых видов затрат на выбранные виды продукции.

К примеру, при расчете себестоимости трех видов изделий, состоящих из трех видов затрат, сформированных в трех местах возникновения затрат, количество регистров составит 3·3·3=27. На определенной стадии единственно целесообразной становится автоматизированная система управленческого учета затрат, поскольку количество регистров может значительно возрастать и требовать больших трудозатрат. В такой ситуации принятие решения о внедрении автоматизированной системы учета должно базироваться на сравнении стоимости ведения учета ручным способом и стоимости внедрения автоматизированной системы.

зация как минимум двух классов языков может получить преимущества от использования молекулярно-биологических виртуальных машин (МБВМ).

В первый класс входят языки для программирования моделей биологических процессов.

Ко второму классу языков можно отнести генетические языки, представленные в живой клетке в форме последовательностей ДНК. В настоящее время многие средства аннотации генома используют подходы, похожие на те, которые используются для разбора текста при создании компиляторов. Было бы интересно попробовать расширить такие средства, включив в них кодогенераторы, способные генерировать код для выполнения на МБВМ.

### Постановка задачи

Представлена спецификация молекулярно-биологической виртуальной машины, включая архитектуру и язык для ее программирования. Предложенная МБВМ предназначена для изучения возможностей моделирования вычислительных процессов в живой клетке и может быть целевой архитектурой для «ДНК-компиляторов».

## 2. Архитектура молекулярно-биологической виртуальной машины

### Типы данных и нотация, использованная в спецификации

Для спецификации, например, виртуальной машины SmallTalk используется подмножество собственно языка SmallTalk. Мы не будем использовать здесь такой подход. Вместо этого, мы определим вначале типы данных и обозначения, используемые в дальнейшем (Табл. 1). Точка "." будет использоваться для обозначения ссылки на поле объекта в форме `Object.Field`. "=" будет использоваться как оператор присваивания, "==" будет обозначать равенство, а "!=" – неравенство двух значений.

### Память объектов

В живой клетке химические клеточные компоненты содержатся в метаболических пулах, отделенных от окружающей среды мембраной. В виртуальных машинах SmallTalk или Java объекты всех видов хранятся в памяти объектов. В этом разделе описывается организация памяти объектов МБВМ. Целью является сохранить логическую организацию, существующую в реальных живых клетках.

Молекулярные и супрамолекулярные компоненты в живых клетках могут свободно плавать или быть связанными определенным способом по отношению к другим компонентам. В последнем случае можно сказать, что молекулы связаны со специфическими сайтами связывания. Сайты связывания могут быть, в свою очередь, частями других молекулярных объектов. Чтобы сохранить эту организацию мы выделили в памяти объектов две части: память сайтов и память объектов (Рис. 1а).

`LCVMObject` - это объект, который имеет список `Sites`, состоящий из сайтов `Site`, которые представляют сайты связывания, принадлежащие данному объекту. Если объект не имеет сайтов связывания, данный список является пустым. Каждый `Site` является объектом, представляющим место, где может находится объект определенного класса `Class`. Он содержит ссылку на связанный объект (`ObjectLink`). Если с сайтом не связан никакой объект, то `ObjectLink=nil`. В ходе цикла работы МБВМ объект может переместиться на другой сайт или быть освобожден в свободное плавание. Каждый сайт содержит список сайтов (`SiteLinks`), на которые объект может быть передан непосредственно.

Операция (`Operation`) – это объект, который содержит списки сайтов (`InputSites, OutputSites`), ссылку на класс (`Class`) и целое число (`OperationIndex`). Пример операции показан на рис. 1б. Класс (`Class`) – это объект, отвечающий за спецификацию того, какие операции могут производиться с определенными объектами или группами объектов (Рис. 2).

Каждый класс имеет список `Object Types`, ссылку на родительский класс `ParentClass` и на список определений методов (`MethodDe-finitions`). Определение метода (`Method Definition`) содержит списки из объектов класса `Site, Object` и `Operation`, а также целочисленные поля `MethodType` и `MethodTime`.

В живых клетках многие операции выполняются параллельно. Чтобы отразить этот факт, МБВМ должна поддерживать параллельное выполнение. Для этих целей интерпретатор хранит список потоков `Threads`. Каждый поток указывает на текущую позицию, т.е. ссылку на сайт, а также статус, который может быть 1 (активен) или 0 (отложен). Для каждого потока интерпретатор действует в соответствии со следующими правилами:

1. Если текущий сайт, на который указывает `Position` не является частью какой-либо операции (`Position.operation==nil`) и не имеет никаких ссылок на другие сайты (`Position.SiteLinks==nil`), то прекратить выполнение текущего потока и удалить его из списка потоков.

2. Если текущий сайт является частью операции (`Position. operation != nil`), и все другие входные сайты этой операции заполнены объектами, (`Position.operation.sites[n].ObjectLink != nil` для всех n), то выполнить данную операцию.

3. Если сайт не является частью операции, но имеет ссылки на другие сайты, то объект перемещается на первый свободный сайт из списка ссылок, который может содержать объекты данного типа.

Для того чтобы выполнить операцию, интерпретатор производит следующие действия: 1. В классе под номером `ClassIndex` найти описание метода, которое определяется индексом операции `OperationIndex`. 2. Если данный метод является элементарным методом, обработать его в соответствии с предопределенными правилами. 3. Если метод не является элементарным, выполнить вызов метода.

Элементарным методом называется такой метод, реализация которого встроена в ВМ.

Чтобы выполнить вызов метода, интерпретатор в общем случае копирует списки `Sites Objects` и `Operations` из `MethodDefinition` в память объектов и создает новый метод в памяти методов (Рис. 3). Затем он соединяет входные сайты операции с входными сайтами метода и выходные сайты опе-

*Табл. Типы данных, использованные для спецификации МБВМ.*

| Тип данных | Описание |
|---|---|
| Boolean | Логическое значение может быть True (правда) или False (ложь) |
| Integer | Целое число |
| Object | Ссылка на объект. Объект может содержать поля различных типов. |
| Enumeration | Перечисление, предоставляет альтернативные имена для чисел, например (active = 0, suspended = 1) |
| List | Список ссылок на объект |

рации с выходными сайтами метода. Новый поток создается для каждого входного сайта. Когда все эти потоки завершаются, метод уничтожается.

В зависимости от типа метода MethodType вызов метода обрабатывается различными способами: 1. Для `MethodType==0` новая копия метода создается при каждом вызове. 2. Для `MethodType==1` постоянно поддерживается определенное заданное число копий методов. В случае, если метод вызывается и нет доступных копий методов (все заняты), соответствующий поток приостанавливается до тех пор, пока предыдущие вызовы не будут завершены. 3. `MethodType==2` работает аналогично `MethodType==1`, но при этом возможно вхождение новых объектов в метод до того, как старые достигнут выхода.

Выполнение метода занимает несколько шагов глобального времени ВМ. Элементарные методы имеют предопределенное время выполнения. Для других методов время выполнения определяется их реализацией или заданием точного время выполнения с использованием `MethodTime`.

### Компилятор

Живые клетки хранят программы в форме ДНК-последовательностей. Эти программы должны компилироваться до того, как они будут выполнены. В МБВМ гены соответствуют строкам исходного кода, а механизмы синтеза белка соответствуют компилятору. Поскольку реализация компилятора зависит от языка, ее описание здесь опущено.

## 3. Функционирование МБВМ

Для того чтобы продемонстрировать, как МБВМ может быть использована для моделирования биологических процессов обработки информации, рассмотрим графическое представление МБВМ кода и затем покажем, как построить модель молекулярной информационной системы, используя очень простой пример из молекулярной биологии.

### Графическое представление МБВМ-кода

Для того чтобы продемонстрировать использование МБВМ, мы должны показать некоторую программу. Чтобы показать программу, необходим некоторый набор обозначений, чтобы описать ее. Для этой цели введем простые графические обозначения для МБВМ программ. Их должно быть достаточно для того, чтобы продемонстрировать работоспособность представленной конструкции МБВМ.

Представленная графическая нотация показана на рис. 4. Пустой сайт обозначается прямоугольником, сайт со связанным объектом – с помощью прямоугольника с именем класса объекта внутри. Операция обозначается кругом с именем операции. Ссылки представлены как линии.

### Модель лактозного оперона

Лактозный оперон представляет собой классический пример генетической регуляции. Рассмотрим, как представить модель лактозного оперона на МБВМ.

Рисунок 5 показывает основную идею всей системы. Клетка (в данном случае прокариотическая клетка) может получать энергию путем преобразования глюкозы. В случае, если также доступна лактоза, она может утилизироваться путем ее преобразования в глюкозу. Детали преобразования скрыты внутри метода, с названием «process», который принадлежит к классу (`LacOperon`, `Lactose`).

Реализация метода `process` показана на рис. 6.

Здесь `Gene1` – это исходная строка, представляющая

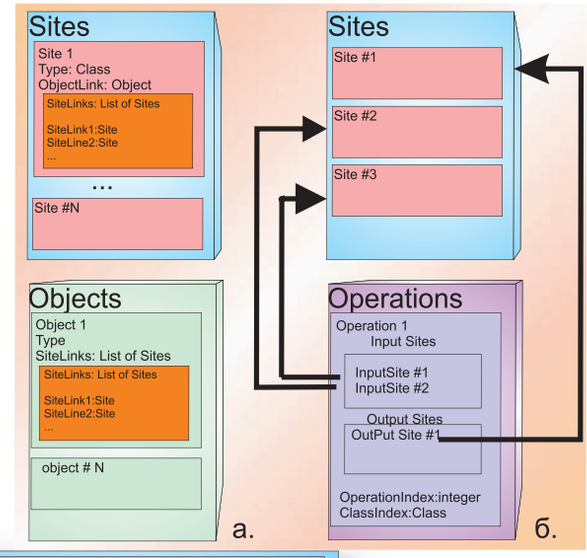


*Рис. 1. Память объектов.*

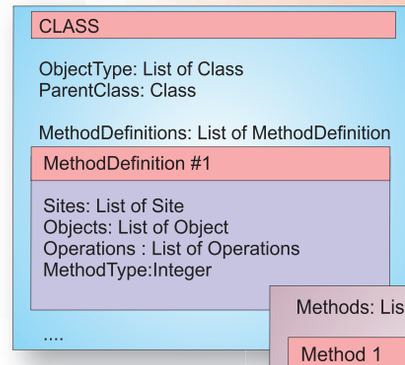


*Рис. 2. Класс МБВМ.*

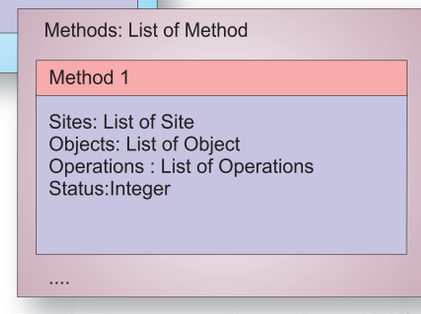


*Рис. 3. Память методов и объект Method.*

ген LacI. Его компиляция и выполнение путем операции `eval` приводит к продукции объекта `repressorpart` – субъединицы тетрамерного репрессора. Объект `repressorpart` затем перемещается через три сайта и останавливается на сайте, обозначенном `repressorpart4`. Одновременно новый `repressorpart` образуется путем выполнения `Gene1` и перемещается к `repressorpart3`.

Когда все сайты `repressorpart` заполнены, выполняется операция `combine` и образуется тетрамерный репрессор, который помещается в `repressorsite`. Если объект `allolactose` (аллолактоза) связан с сайтом `allolactose`, тогда следующая операция `combine` производит комплекс `repressor– allolactose`. В противном случае он связывается с объектом `string` (строка) из сайта `Gene2`, таким образом, делая невозможным его компиляцию и выполнение путем метода `eval`.

В случае, когда строка `Gene2` не комбинируется с `repressor`, он компилируется и выполняется. Результаты выполнения продуцирует экземпляр метода `Transport` и метода `Haloctozidase`, таким образом, увеличивая количество этих методов в памяти методов, что приводит к увеличению скорости преобразования лактозы в глюкозу (`lactose to glucose`).

### Заключение

Широкий диапазон молекулярно-биологических процессов может быть логически последовательно и единообразно представлен в виде МБВМ программ, включая ферментативные реакции и образование нековалентно связан-

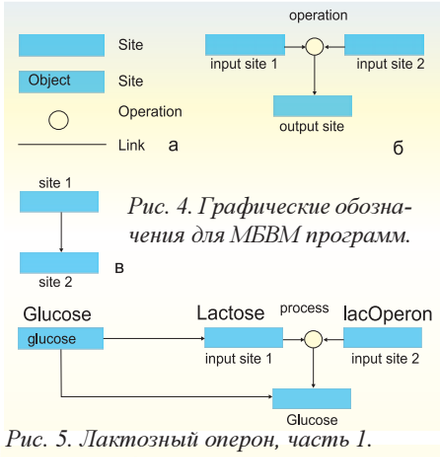


*Рис. 4. Графические обозначения для МБВМ программ.*

*Рис. 5. Лактозный оперон, часть 1.*

ных комплексов. Концепции метаболических путей могут быть абстрагированы как МБВМ операции и целые регуляторные подсистемы могут быть представлены как МБВМ объекты.

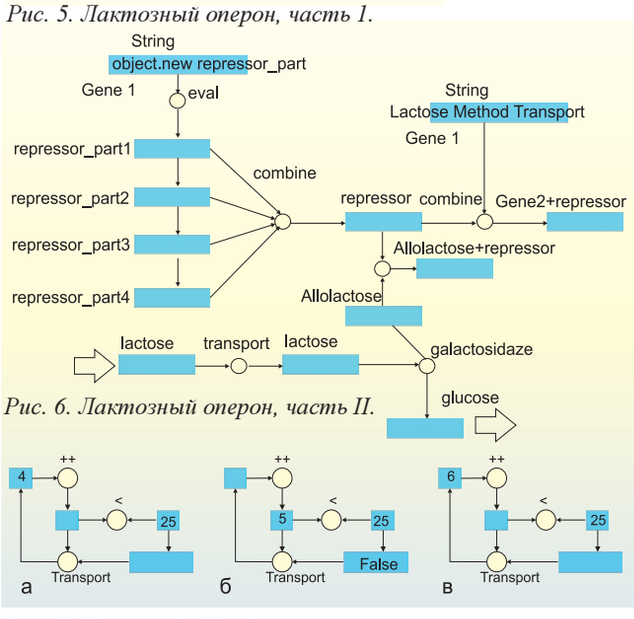


*Рис. 6. Лактозный оперон, часть II.*




*Рис. 7. Простая МБВМ программа для цикла.*

Экспрессия генов представляется в МБВМ программах как компиляция строк, и дифференциальный сплайсинг может быть представлен как система обработки строк. Таким образом, эффект мутаций может быть представлен как модификация исходных строк в определенных местах. Возможность отображения мутаций ДНК на строки исходного кода говорит о возможности создания ДНК –> МБВМ компилятора.

Более того, МБВМ может быть использована для разработки и выполнения программ, которые прямо не связаны с молекулярными системами (Рис. 7). Можно легко представить, что возможно использование одного и того же языка для моделирования биологических систем, написания расширений для программы моделирования или даже программирования баз данных для хранения биологической информации. Разумеется, любой язык общего назначения, такой как C++ или Java, может использоваться для этого, но их использование требует знаний организации операционной системы, что делает программирование сложной задачей для биологов. МБВМ этого не требует. Она основана на логической организации живой клетки, поэтому молекулярные биологи, которые научатся программировать МБВМ, обнаружат, что они уже знакомы с используемым вычислительным устройством.

## Литература


Amdahl G.M., Blaauw G.A., Brooks P.P. Architecture of the IBM System-360. *IBM J. of Research and Development.* N 8. 1964. 87-101.

Cardelli L. Abstract Machines of Systems Biology. *Transactions on Computational Systems Biology.* N 3. 2005. 145-168.

Fages F., Soliman D., Chabrier-Rivier N. Modelling and query inginteraction networks in the biochemical abstract machine BIOCHAM. *Biological Physics and Chemistry.* N 4. 2004. 64-73.

Goldberg R.P., Survey of virtual machine research. *IEEE Computer Magazine.* N 6. 1975. 34-45.

Goldberg A., Robson D., *Smalltalk-80: The Language and Its Implementation.* Addison-Wesley. 1983.

Ji S. The cell as the smallest DNA-based molecular computer. *Biosystems.* N 52. 1999. 123-133.

Lindholm T., Yellin F. *The Java(TM) Virtual Machine Specification (2nd Edition).* Addison-Wesley Professional. 1999.

Regev A., Panina E.M., Silverman W., Cardelli L., Shapiro E. BioAmbients: An Abstraction for Biological Compartments. *Theoretical Computer Science.* N 325. 2004. 141-167.

Shapiro B.E, Levchenko A., Elliot M., Wold B., Eric D.M. Cellerator: extending a computer algebra system to include biochemical arrows for signal transduction simulations. *Bioinformatics.* N 19. 2003. 677-678.

Tomita M., Hashimoto K., et al., E-CELL: software environment for whole-cell simulation. *Bioinformatics.* N 15. 1999. 72-84.



Казань: Образцовая типография, 2006. - 142 с.

**Интегральная геология**

**Р.Х. Сунгатуллин**

Монография посвящена разработке нового научного направления, системно изучающего информационно-энергетические функции различных сфер Земли.

Предложена универсальная методика компьютерно-математического изучения геологического пространства. Разработаны методические приемы формализации и ранжирования качественных характеристик отдельных сред на примере осадочного чехла Восточно-Европейской платформы. Выявлены новые геодинамические, геохимические, геофизические критерии поисков и прогнозирования полезных ископаемых. Книга предназначена для исследователей в области наук о Земле, геологов, экологов и географов широкого профиля, преподавателей и студентов естественных факультетов университетов.

Казань: Изд-во «Плутон», 2003. - 176 с.

АКАДЕМИЯ НАУК РЕСПУБЛИКИ ТАТАРСТАН

**Геофизические поиски минерального строительного сырья**

**М.Я. Боровский, Г.Е. Кузнецов, Д.И. Хасанов**

В книге показаны возможности геофизических методов поиска и разведки месторождений строительных полезных ископаемых. Рассмотрены результаты исследований на залежах карбонатного, глинистого, песчаного и песчано-гравийного сырья; представлены материалы опытно-методических наблюдений нетрадиционных видов полезных ископаемых - оникс кальцитовый и др.; приведены сведения о физических свойствах горных пород; предложена рациональная стадийность геофизических работ; определены перспективные геофизические технологии. Издание предназначено для широкого круга специалистов и студентов геофизических, геологических, экологических и строительных специальностей ВУЗов.

ISBN 5-902089-07-7