УЧЕНЫЕ ЗАПИСКИ КАЗАНСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА
Том 147, кн. 2 Естественные науки *2005*

УДК 57:007.001.57+577.2.01

МОЛЕКУЛЯРНО-БИОЛОГИЧЕСКОЕ ВЫЧИСЛИТЕЛЬНОЕ УСТРОЙСТВО: ПРИНЦИПЫ ОРГАНИЗАЦИИ

Д.С. Тарасов, Н.И. Акберова

Аннотация

Известно, что любая информационная система может быть рассмотрена с двух точек зрения. Первый подход имеет дело с физической реализацией данной системы, а второй – с ее логической организацией. Рассмотрение логической организации системы предполагает определенный уровень абстракции от ее физической структуры. Живая клетка как информационная система также может быть рассмотрена с обеих точек зрения, однако до настоящего времени большинство исследований в этой области имеет дело с физической структурой клетки, и даже в работах, исследующих процессы регуляции, не достигается необходимый уровень абстракции. В данной работе рассматривается модель логической организации живой клетки, не зависимая от ее физической структуры.

Введение

Взаимопроникновение молекулярной биологии и кибернетики в настоящее время приобретает все более выраженный характер, что во многом определяется сходством между системами биохимической регуляции и процессами управления в электронных вычислительных машинах.

Живая клетка часто рассматривается как своего рода молекулярный компьютер, управляемый программой, закодированной в ДНК [1]. Рассматривается даже особый генетический язык в форме ДНК-кода по аналогии с естественными языками общения людей [2, 3]. В других работах исследуются уже не только ДНК, но и другие биологические молекулы, и таким образом появляется термин «клеточный язык» (cellese) [1, 4]. Лингвистический подход применяется при решении проблемы автоматической аннотации секвинированных последовательностей ДНК [5–7].

К сожалению, в большинстве работ в этом направлении используется лишь формальная аналогия ДНК-кода с языком и применяются методы математической лингвистики, не давая при этом четких спецификаций клеточному или генетическому языку. В теоретических исследованиях встречаются самые разнообразные интерпретации самого понятия генетического языка и различные, часто малообоснованные аналогии с языками общения людей. В практических же приложениях, при построении программ автоматической аннотации генома, использующих лингвистические формализмы, вопросы об определении и структуре клеточного языка обычно упускаются, и к произвольному фрагменту ДНК применяются те же разрешающие процедуры, что и к предложению естественного языка.

В то же время очевидно, что клеточный язык при любом определении должен передавать информацию о процессах регуляции клеточного метаболизма и экспрессии генов. Исследования клеточного языка, направленные на извлечение и манипулирование этой информацией, имеют первостепенное значение для решения проблем автоматической аннотации геномов, генной инженерии, диагностики и лечения заболеваний.

С другой стороны, в последние годы интенсивно разрабатываются проблемы создания искусственных молекулярных компьютеров. Работающие в этой области исследователи предполагают, что подобные компьютеры могут быть более экономичными, компактными и в ряде случаев более производительными, нежели их кремневые аналоги [8]. К сожалению, несмотря на значительные успехи, достигнутые в данном направлении, современные «молекулярные компьютеры» не являются автономными, требуют участия человека при выполнении любых операций, а вывод данных в них осуществляется путем применения стандартных биохимических процедур анализа (например, хроматографии) [9].

Единственным известным на сегодняшний день автономным молекулярным компьютером является живая клетка, что определяет важность изучения клеточного языка для решения задач конструирования искусственных молекулярных компьютеров.

Клетка способна воспринимать поступающую информацию, перерабатывать ее, хранить, воспроизводить, реализовывать процессы управления и регуляции. Однако эти аспекты функционирования живой клетки изучены, как ни странно, весьма слабо. Достаточно отметить, что существующая теория вычислительных машин не способна иметь дело с молекулярно-биологическими процессами, чему имеются как чисто математические, так и экспериментальные доказательства [10].

Изложенные выше соображения определяют необходимость системного изучения проблемы клеточного языка с целью выявления принципов его построения и функционирования.

1. Подход к анализу клеточного языка и архитектуры клеточного устройства

В отличие от языка общения людей, основным назначением клеточного языка является не обеспечение обмена информацией между индивидуумами, а обеспечение эффективной регуляции клеточных процессов, т. е. клеточный язык содержит в той или иной форме указания для некоторого управляющего аппарата, напоминая этим язык программирования ЭВМ.

Программа для процессора ЭВМ представлена в виде двоичных кодов, т. е. конечной последовательности единиц и нулей. Аналогичным образом программы на клеточном языке составлены из конечной последовательности мономерных звеньев в биополимерах, основным из которых является ДНК, состоящая из четырех типов оснований и реализующая код в системе исчисления с основанием 4.

Для человека написание и чтение программ, представленных в виде машинных кодов, чрезвычайно неудобно, а в случае сложных программ вообще невозможно. В этих целей используются языки высокого уровня. Так Ассемб-

лер, будучи максимально приближен к аппаратным средствам ЭВМ, все же содержит важную особенность – программы на Ассемблере представляют собой последовательность команд процессора, которые являются одновременно синтаксическими и семантическими единицами. Каждая команда имеет смысл, соответствующий ее действию на среду исполнения программы. Ясно, что семантический анализ языковых сообщений возможен только на уровне семантических единиц. Любой практически полезный язык должен содержать определенные семантические единицы.

Развитие языков программирования шло в направлении укрупнения их семантических единиц, так что семантическая единица на языке более высокого уровня соответствовала целому набору единиц языка низкого уровня. Программы на языках высокого уровня необязательно представляют собой набор последовательных команд – примером может служить язык сверхвысокого уровня PROLOG, где семантической единицей является логический предикат.

Происходила ли эволюция клеточных языков аналогично развитию языков программирования в направлении укрупнения семантических единиц? Утверждать это наверняка невозможно, но в пользу данного предположения можно привести некоторые аргументы.

Повышение уровня языка программирования способствует ускорению процесса возникновения новых, более сложных программ, независимо от способа их написания, будь то целенаправленное программирование или метод проб и ошибок. С эволюционной точки зрения увеличение скорости возникновения новых программ ведет к повышению адаптационных возможностей живых организмов. Можно также показать, что если бы основным механизмом возникновения новых генетических программ было только мутационное изменение на самом низком уровне (мономерные звенья ДНК), то вероятность возникновения к настоящему времени высших организмов была бы близка к нулю [11]. Ответ на этот важный вопрос, проливающий свет на причины прогрессивной эволюции, должно дать углубленное изучение клеточного языка.

Исходя из изложенных выше соображений и учитывая практические потребности современной молекулярной биологии в исследовании клеточного языка, является целесообразным создание модели и полной спецификации *формального языка описания клеточных программ* (*CPDL*), отвечающей следующим требованиям:

- Рассматриваемый язык должен передавать особенности клеточного языка в такой степени, чтобы обеспечивалась возможность создания автоматических кросс-компиляторов в нуклеотидный код и трансляторов, выполняющих обратную операцию.
- Быть достаточно близким к символьным языкам программирования ЭВМ для обеспечения удобочитаемости и простоты составления программ на нем.

Можно предположить следующий подход к созданию языка, отвечающего перечисленным выше требованиям.

1. Построить описание биохимических процессов в определенной модельной системе биологической регуляции.

2. Рассмотреть только те аспекты данных процессов, которые являются инвариантными относительно переноса в ЭВМ, т. е. процессы управления.

3. Создать символьное описание процессов, выделенных в п. 2.

2. Проблема построения языка описания клеточных программ

В литературе описан ряд программ, предназначенных для моделирования клеточного метаболизма. В существующих программах применяются или уравнения химической кинетики, дающие возможность строить количественные модели, или системы правил для качественного описания происходящих процессов. Ни одна из этих программ, тем не менее, не подходит для наших целей, так как не включает возможности построения трансляторов вида:

Язык программирования высокого уровня ↔ Последовательность мономерных звеньев в биополимере.

Ясно, что создание подобных трансляторов требует глубокого понимания структуры клеточного языка. Кроме того, язык высокого уровня, используемый для построения клеточных программ, должен максимально передавать особенности клеточного языка.

О решении проблемы управления регуляцией молекулярных процессов в живой клетке, в том числе и проблемы регуляции репликации можно говорить лишь в случае, если возможно решение следующих задач:

1. трансляция управляющей информации, представленной в виде последовательности нуклеотидов ДНК, на символьный язык формального описания биологических процессов;
2. детализированное описание среды функционирования регуляторного клеточного аппарата;
3. интерпретация данных из пп. 1 и 2 для получения модели функционирования данной клеточной программы;
4. корректировка, а также создание новых клеточных программ на языке программирования высокого уровня;
5. трансляция отредактированных (новых) клеточных программ в последовательность ДНК.

Только в том случае, если разработанные математический аппарат и комплекс программных средств позволяют решать задачи 1–5, мы можем говорить о полном решении проблемы управления рассматриваемым биопроцессом. Необходимо констатировать, что в настоящее время не существует программ для комплексного решения задач 1–5.

С целью нахождения путей решения задач 1–5 необходимо произвести разработку следующих средств:

1. формального языка описания клеточных программ, рассмотренного в разделе 1;
2. интерпретатора программ данного языка для моделирования клеточных процессов средствами вычислительной техники;
3. компилятора программ CPDL в последовательность звеньев ДНК.

Создание компилятора из п. 3 требует, в свою очередь, создания эффективной базы знаний о конверсии CPDL ↔ ДНК код. В свою очередь, заполнение

такой базы требует существования модели конвертирования экспериментальных данных о структуре и функции биополимеров. Другим возможным подходом к проблеме кросс-компилятора является использование мощных вычислительных машин для моделирования функций биополимеров.

3. Проблема выбора модели

С точки зрения активности процессов управления, рассматриваемую модель можно условно разделить на три части:

1. внеклеточная среда с концентрациями, в высокой степени не зависимыми от действий управляющего аппарата клетки;
2. внутриклеточная среда, где концентрации могут сильно зависеть от действий управляющего аппарата;
3. сам управляющий аппарат клетки, представленный белками и нуклеиновыми кислотами.

Одной из основных проблем, возникающих при описании такой системы, является проблема «уровня детализации». С одной стороны, мы имеем непрерывную модель, описываемую уравнениями химической кинетики, где рассматриваются концентрации ферментов и их субстратов, а с другой стороны, во многих случаях необходимо использовать дискретную модель, детализирующую все процессы до отдельных молекул. Дискретная модель является более естественной для описания молекулярного управляющего аппарата, поскольку число регуляторных молекул определенного типа часто исчисляется сотнями или даже десятками на клетку. Непрерывная модель лучше описывает ситуацию в случае низкомолекулярных соединений. В то же время сопряжение дискретной и непрерывной частей в рамках одной и той же модели в данном случае является затруднительным.

Еще одной частью проблемы «уровня детализации» является проблема представления в модели CPDL полимерных молекул. С одной стороны, такие молекулы являются индивидуальными химическими веществами и должны рассматриваться как таковые. Однако во многих случаях факт построения данных молекул из отдельных мономерных звеньев играет первостепенную роль в процессах регуляции.

Существует также и проблема выбора вычислительной модели для гипотетической клеточной машины, использующей в качестве основного языка CPDL. Совершенно очевидно, что архитектура клеточного компьютера не имеет практически ничего общего с распространенной в современных компьютерах архитектурой машины фон Неймана, так как не содержит последовательно исполняемого набора команд и системы адресации памяти. Можно предположить, что функционирование клеточного компьютера основано на правилах и логическом выводе, а программы для него носят декларативный характер.

Известны два основных типа систем, основанных на заданных правилах и реализующих логический вывод. Это системы с обратным выводом (например, PROLOG) [12] и системы с прямым выводом (например, OPS-5). Клеточный компьютер реализует, скорее всего, систему с прямым выводом, исходящим не из конечной цели, а из имеющихся в системе фактов и правил. Как правило, практически полезные системы такого типа снабжаются некоторым механиз-

мом отбора правил, необходимых в данный момент, так как иначе они будут функционировать неэффективно.

4. Архитектура молекулярно-биологического устройства

Приведенный выше анализ наталкивает нас на возможность использования в качестве отправной точки модели «доски объявлений», использованной в компьютерной системе HEARSAY-II [13]. В системе «доски объявлений» используется механизм прямого логического вывода, основанный на правилах. При этом к поступающим фактам применяется определенный набор правил, состав которого зависит от текущего состояния системы.

Попытаемся адаптировать данную вычислительную модель для клеточного интерпретатора. В качестве верхнего уровня представления знаний в клетке используется *дисперсная база значений.* Дискретный ряд значений, элементами которого являются химические соединения, является своеобразной областью определения для «переменных», представленных в виде сайтов связывания на биологических макромолекулах.

Факты и правила в рассматриваемой системе неявно содержатся в структуре белков, а метаправилам, определяющим наборы функционирующих правил, соответствуют последовательности нуклеотидов молекулы ДНК, объединенные в единицы генетической информации.

Из приведенного описания следует интересная особенность CPDL по отношению к традиционным языкам программирования ЭВМ. В нем принципиально отсутствует оператор присваивания значения (a:=b) переводящий систему [a = y, b = z] в состояние [a = z, b = z]. Вместо него может быть рассмотрен аналогичный оператор перемещения значения с действием [a = y, b = z] → [a = z, b = <..>], где <..> представляет пустое значение.

4.1 Механизмы клеточной логики. Механизмы, реализующие клеточную логику (рис. 1), включают в себя белковый аппарат, РНК и ДНК. Все прочие молекулы описываются как составляющие внешнего состояния. Целесообразно вначале рассмотреть несколько моделей различной сложности.

4.2. Модель А. Бинарная логика. Модель А является крайним упрощением, однако ее ценность состоит в возможности выявить важные принципы организации механизмов клеточной логики (МКЛ).

МКЛ воспринимает состояние внешней среды с помощью белков-рецепторов. Таким образом, состояние белков-рецепторов в каждый момент времени отражает все важные для клетки параметры ее реального состояния. Поскольку рецепторные белки для того, чтобы оказать какое-либо действие на ДНК, должны связаться с некоторыми регуляторными сайтами, то таким образом состояние этих сайтов на ДНК также отражает все важные для клетки параметры реального состояния. Код белка-эффектора содержит информацию о том, каким образом это состояние должно быть изменено. Следовательно, ДНК содержит информацию, описывающую возможные состояния клетки, а также инструкции по последовательному изменению этих состояний. Данная информация записана на особом клеточном языке и реализуется посредством белкового аппарата.

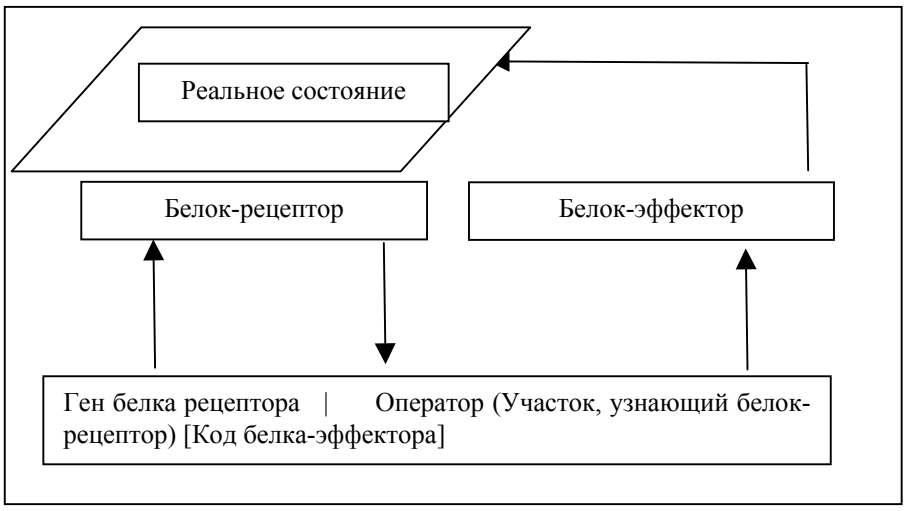

Рис. 1. Механизмы клеточной логики, модель 1

Состояние клетки описается наличием или отсутствием веществ, способных оказывать влияние на белки-рецепторы. При формальном описании действия МКЛ по модели А нет необходимости рассмотрения белков-рецепторов и белков-эффекторов как отдельных объектов МКЛ, т. к. в рамках данной модели они являются абсолютно прозрачными для информации.

4.3. Модель В. Универсальное молекулярно-биологическое вычислительное устройство (МБВУ).

Молекулярно-биологическое вычислительное устройство включает в себя рабочее пространство, знания и метазнания.

4.3.1. Рабочее пространство МБВУ. Рабочее пространство, возможно, не является собственно частью универсального клеточного интерпретатора (УКИ), однако оно является той областью, с которой фактически оперирует клеточный интерпретатор. Вообще место рабочего пространства в структуре МБВУ находится под вопросом.

Рабочее пространство содержит определенное количество элементов, называемых «слоты состояния» (state slot, s-slot). Каждый s-slot характеризуется типом образующего его химического вещества и состоянием. В модели С-1 каждый s-slot представляет одну молекулу определенного вещества, в модели С-2 – группу молекул, находящихся в одинаковом состоянии. В последнем случае s-slot также содержит число (концентрация, число молекул на клетку). Состояние – свободное вещество или связанное.

4.3.2. Знания МБВУ. С точки зрения биохимии знания МБВУ представлены белковыми молекулами. Знания являются фактами и правилами, относящимися к состоянию рабочего пространства.

Структура пространства знаний (рис. 2):

ФАКТ (Состояние Факта) ВЕЩЕСТВО А НАЛИЧИЕ.

ПРАВИЛО (Состояние Правила) [Список изм. S-Slot] :- [Список S-Slot].

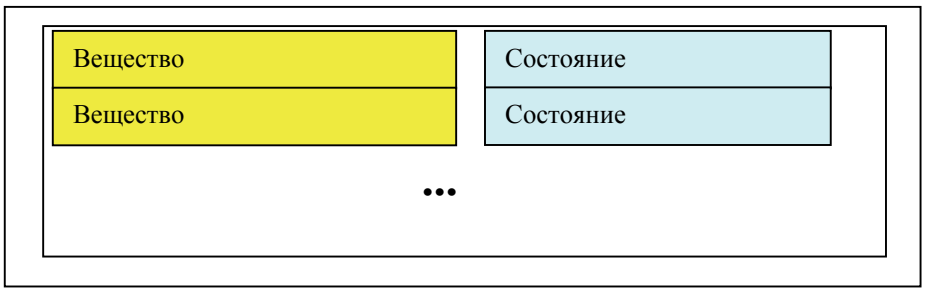

Рис. 2. Рабочее пространство универсального клеточного интерпретатора

4.3.3. Метазнания МБВУ. Метазнания МБВУ содержатся в ДНК. Они определяют правила использования знаний, т. е. какие именно знания необходимы в данный момент.

Структура метазнаний:

[Необходимые знания] :- [Имеющиеся знания]

Соотношение между предметным языком классической молекулярной биологии и языком МБВУ неоднозначно. В свете используемого формализма мы можем видеть, что молекулярная биология имеет дело с рабочим пространством МБВУ и соответственно описывает все происходящие в клетке процессы в терминах рабочего пространства. С другой стороны, *с точки зрения клетки как информационной системы*, основное значение для процессов управления имеет пространство знаний и метазнаний.

Рабочее пространство, с одной стороны, и комплекс знания-метазнания, с другой, являются в сущности двумя аспектами одного и того же явления. Знания и метазнания определенным образом закодированы в терминах состояния рабочего пространства, но при этом имеют самостоятельное значение, не совпадающие в общем случае со значением рабочего пространства.

5. Язык управления МБВУ

Язык CPDL-1 (Cell Program Description Language) разработан с учетом требований, перечисленных в предыдущем разделе, и предназначен для моделирования процессов управления и регуляции в прокариотических клетках, а также для исследований структуры и особенностей клеточного языка прокариот.

5.1. Типы данных.

Probability. Вероятностный тип. Число с плавающей точкой, являющееся константой. Никакая переменная не может иметь данный тип.

Basic type. Основной тип. Элементарная единица данных, далее не делимая.

MetaControl type. То же, что и основной тип, но используется при управлении метапроцессами.

List. Список. Список состоит из головы и хвоста. Голова может иметь основной тип, мета-контрольный тип или составной тип, хвост же всегда является списком.

State: Состояние. Описывает состояние переменной основного или метаконтрольного типа. Может иметь значения *свободно* или указывать на регистр процесса.

Compound type. Составной тип. Дерево или список.

Process. Процесс. Тип данных представляющий определенный процесс.

Для объявления раздела описания типов служит ключевое слово types. В конце каждой строки ставиться знак «;». Запись в разделе типов выглядит следующим образом:

<Название типа>: <Тип>;

Конец раздела описания типов обозначается ключевым словом end types;

5.2. Пул значений. Содержит переменные разных типов. Вариантом его служит внешний пул значений, доступ к которому осуществляется посредством директивы **external**. Объявление раздела описания пула производится следующим образом:

<Область действия> pool;

<Область действия>
internal внутриклеточный пул (используется по умолчанию).
external внеклеточный пул.
Формат описания значений в пуле:

<Число>:<Имя>=<Тип данных>

<Число>
Количество значений, помещаемых в пул изначально.
<Имя>
Идентификатор данных значений.
<Тип данных>
тип данных, объявленный в разделе типов.

Пример:

Фрагмент 1.1

```
types;
  Lac_type:  basic type
pool;
 Lac1(free)       %значение в пуле значений
 Lac2(process <pr_name> :register <name>) %значение в ре-
гистре процесса
 N:Lac(free)          % (сокр. запись)
external pool
 K:Lac(free)
```

5.3. Процессы. Процесс является автономной процедурой, время работы которой определяется метапроцессами и другими процессами. Регистры процессора могут содержать значения типов, заданных при определении процесса.

Описание процесса состоит из заголовка, тела процесса и окончания. Заголовок процесса включает в себя ключевое слово process, название данного вида

процесса, и списка регистров с указанием их типов. Регистр процесса можно представить себе как ячейку памяти, способную хранить значение определенного типа.

Process <process_name> (*Type1,Type2...TypeN*).

Тело процесса содержит факты и правила, управляющие значениями регистров процесса и их взаимодействием с пулом значений.

5.4. Факты. Факты описывают соотношения, использующиеся для описания поведения процессов. Выделяются следующие типы фактов, в каждом имеется два подтипа:

a-bound(*V1,V2,V3,Vn , Pr <Probability>*).

b-bound(*Value <Compound>*).

Определяет вероятность или устанавливает факт принятия регистрами процесса значения определенного типа. При этом данное значение извлекается из пула значений и помещается в регистр процесса. V1-Vn – последовательное описание состояний всех регистров процесса. В качестве описания состояний применяется следующие обозначения:

_ – состояние данного регистра не учитывается.

R – данный регистр свободен.

B - <Тип данных> – данный регистр принимает значение определенного типа данных. Тип данных указывается в случае, если при описании регистра процесса объявлен составной тип.

a-release/N(*Pr <Probability>*).

b-release().

Определяет вероятность или устанавливает факт принятия регистром процесса пустого значения. При этом значение, связанное до этого с регистром, возвращается в пул значений. Формат описания аналогичен формату факта bound.

a-transform(V1 <List>, V2 <List>, Pr <Probability>). Определяет вероятность перехода значения V1 в V2.

5.5. Правила используются для описания условных вероятностей.

<a-[факт]> (результат) :- <b-[факт]> (условие).

Правая часть может содержать несколько предикатов, соединенных связкой *И*.

5.6. Метапроцессы.

metaprocess <meta_name> (Pr1 <Process type1>... PrN<Process type N>, <Probability>) :- <Meta-control(<State>)>,...

Определяет вероятность добавления нового процесса meta_name при выполнении условий, перечисленных в правой части.

6. Пример: программирование задачи регуляции лактозного оперона

Данный раздел демонстрирует возможности CPDL для описания процессов генной регуляции на примере лактозного оперона *Escherichia coli.* Приводимая

здесь CPDL-программа составлена на основании самых общих данных о регуляции лактозного оперона.

При работе с конкретными биохимическими механизмами в роли типов данных выступают соответствующие химические соединения. Программе потребуются переменные представляющие лактозу, глюкозу, галактозу, аллолактозу, кофермент А, ацетилкофермент А.

Кроме того, нужен мета-контрольный тип для сайта связывания репрессора на ДНК.

Фрагмент 1.2

Types

Basic type: Lactose, Glucose_type, Galactose_type, Allolactose_type, CoA_Type, AcetylCoA_Type.

Meta-Control: Repressor_binding_site

Лактозый оперон включает четыре белка: репрессор LacI, β-галактозидазу, катализирующую гидролиз лактозы до галактозы и глюкозы, β-галактозидпермиазу, необходимую для прохождения лактозы через клеточную мембрану, и β-галактозидтрансацетилазу, катализирующую перенос ацетильного остатка от ацетилкофермента А на галактозу. Эти белки представляются четырьмя процессами.

Фрагмент 1.3

```
process  LacI(Repressor_binding_site, allolactose_type)
  bound(Repressor_binding_site,_,0.99999):-
release(_,allolactose_type)
  release(Repressor_binding_site,_,0,00001):-
release(_,allolactose_type).
  bound(Repressor_binding_site,_,0.00001) :-
bound(_,allolactose_type).
  release(Repressor_binding_site,_,0,99999) :-
bound(_,allolactose_type).
  bound(_,allolactose_type,0.9).
  release(_,allolactose_type,0.1).
end process LacI

process LacY(allolactose_type)
 external bound(allolactose_type,P).
 release(allolactose_type,P1).
end process LacY

process
LacZ(lactose_type,glucose_type,galactose_type,allolactose_type
)
bound(lactose_type,L).
release(lactose_type,L1).
transform([lactose_type],[glucose_type,galactose_type],L2).
transform([lactose_type],[allolactose_type],L3).
release(_,glucose_type,galactose_type,_,L4).
release( _,_,_,allolactose_type,L5).
end process LacZ
```

Контроль за синтезом белков LacZ, LacY, LacA осуществляется одним оператором при условии, что операторный участок не занят репрессором. Синтез репрессора в данной модели является безусловным.

Фрагмент 1.4.

```
metaprocess lac_operon([LacY,LacZ],O1) :-
Repressor_binding_site(free).
metaprocess repressor([LacI],O2).
```

Таким образом, полная программа содержит три процесса и два метапроцесса.

Программа 1.1

```
PROGRAM Lac-operon regulation
Types
Basic    type:    Lactose,    Glucose_type,    Galactose_type,
Allolactose_type.
Meta-Control:   Repressor_binding_site
Variables
Lactose         :  Lactose_type.
Glucose         :  Glucose_type.
Galactose       :  Galactose_type.
Allolactose:    :  Allolactose_type.
LacI_BS         :  Repressor_binding_site

Processes
process  LacI(Repressor_binding_site, allolactose_type)
  bound(Repressor_binding_site,_,0.99999)                    :-
release(_,allolactose_type)
  release(Repressor_binding_site,_,0,00001)                  :-
release(_,allolactose_type).
  bound(Repressor_binding_site,_,0.00001)                    :-
bound(_,allolactose_type).
  release(Repressor_binding_site,_,0,99999)                  :-
bound(_,allolactose_type).
  bound(_,allolactose_type,0.9).
  release(_,allolactose_type,0.1).
end process LacI

process LacY(allolactose_type)
 external bound(allolactose_type,P).
 release(allolactose_type,P1).
end process LacY

process
LacZ(lactose_type,glucose_type,galactose_type,allolactose_type
)
bound(lactose_type,L).
release(lactose_type,L1).
transform([lactose_type],[glucose_type,galactose_type],L2).
transform([lactose_type],[allolactose_type],L3).
```

```
release(_,glucose_type,galactose_type,_,L4).
release( _,_,_,allolactose_type,L5).
end process LacZ

Metaprocesses
metaprocess            lac_operon([LacY,LacZ],O1)            :-
Repressor_binding_site(free).
metaprocess repressor([LacI],O2).

END PROGRAM.
```

7. Язык CDPL/HL

Рассмотренный выше язык CDPL/1 является хорошим средством достаточного описания молекулярно-биологических процессов и отражением логической организации живой клетки. Однако в ряде случаев предпочтительнее иметь более приближенный к человеческой логике язык, не потеряв одновременно и преимущества CDPL как языка, имеющего структуру предметной области. Кроме того, хотелось бы повысить компактность описания и выразительность языка.

В случае описания процессов генетической регуляции целесообразно придерживается модели А, т. е. предположения о «прозрачности» белково-ферментативного аппарата для механизмов реализации клеточной логики. Это дает возможность описать язык более высокого уровня, CDPL/HL, более удобный для ряда практических применений. CDPL/HL сочетает в себе преимущества прологовской нотации и вычислительной модели, характерной для всех языков класса CDPL.

Запишем следующие выражения:

Фрагмент 2.1.

```
Promoter1:n1 :- Protein1:n2
Promoter2:n3 :- not(Protein2:n4)
Promoter3.
```

Первая строка означает, что экспрессия всех генов, начинающихся с промотора 1 (Promoter1), происходит с интенсивностью n1 в случае если присутствует белок-активатор 1 (Protein1) в концентрации, большей n2. Строка два говорит нам о том, что экспрессия всех генов, начинающихся с промотора 2 (Promoter2), происходит с интенсивностью n3 в случае, если концентрация белка-репрессора (Protein2) не превышает значения n4. Строка три говорит о том, что все гены, начинающиеся с промотора 3 (Promoter3), являются конститутивными.

Следующий фрагмент (2.2) описывает взаимосвязь генов с промоторами.

Фрагмент 2.2.

```
Declare
operon1:-protein1,protein2,protein3.
…
```

```
Genes
Protein4 :- Promoter1
case operon1/1 :- Promoter2 or  operon1 :-
Promoter1,Protein4:n5;
```

Первая строка фрагмента 2.2 содержит ключевое слово Declare. Вторая описывает оперон, состоящий из кодирующих белки 1, 2, 3 участков. Далее следует ключевое слово Genes, определяющее начало нового раздела программы. Первая строка в этом разделе означает, что белок 4 (Protein4) связан с промотором 1. Вторая строка описывает более сложную регуляцию. Оперон 1 начинается с промотора 2 и с него экспрессируется один ген, соответствующий белку 1. Белок 4 (Protein4) в концентрации большей n5 осуществляет антитерминацию, что приводит к экспрессии остальных генов оперона.

Пользуясь такими простыми выразительными средствами, можно описывать сложные механизмы генетической регуляции, в частности, программа 2.1 описывает регуляторный каскад фага лямбда.

Программа 2.1. Регуляторный каскад фага лямбда.

```
Declare
domain_1:- N,cIII,int,xis
domain_2:- Cro,cII,O,P,Q
Promoters
Pl:n1 :- not(Cro:n2),not(cI:n3). Pr:N1 :-
not(Cro:n2),not(cI:n3).
Pe:n4:- cII:n5,cIII:n6. Pm :- cI:n7,not(Cro:n8). lytic_genes
:- Q:N9.
Pr1.
Proteins
cI :- Pe;Pm.
case  domain_1/1 :- Pr or domain_1 :-Pr,N:n9.
case domain_2/1 :- Pl  or domain_2:-Pl,N:n11.
Cro :- Pr. N:- Pl.
lytic_genes :- Pr1,Q:n11.
```

Единственный недостаток рассматриваемого подхода состоит в невозможности описать средствами CDPL/HL факта расщепления репрессора cI белком RecA, т. к. он в отличие от CDPL/1 включает только ДНК-белковые взаимодействия. CPDL/HL является языком более высокого уровня, чем CDPL/1, и у нас может возникнуть необходимость включить предложения CDPL/1 в текст CDPL/HL программы подобно тому, как подпрограммы на языке ассемблера включаются в программы на языках высокого уровня, таких, как Си, Паскаль, Пролог. Данная ситуация демонстрируется фрагментом 2.3.

Фрагмент 2.3.

```
process RecA(cI,cI_part1,cI_part2)
 bound(cI,_,_).transform(cI,cI_part1,cI_part2).
 release(_,cI_part1,cI_part2).
end process
```

Включив данный фрагмент в программу 2.2, мы получим полное описание регуляторного каскада фага лямбда:

Программа 2.2. Регуляторный каскад фага лямбда с RecA.

```
Declare
domain_1:- N,cIII,int,xis
domain_2:- Cro,cII,O,P,Q
Promoters
Pl:n1 :- not(Cro:n2),not(cI:n3). Pr:N1 :-
not(Cro:n2),not(cI:n3).
Pe:n4:- cII:n5,cIII:n6. Pm :- cI:n7,not(Cro:n8). lytic_genes
:- Q:N9.
Pr1.
Genes
cI :- Pe;Pm. Cro :- Pr. N:- Pl. lytic_genes :- Pr1,Q:n11.
case domain_1/1 :- Pr or domain_1 :-Pr,N:n9.
case domain_2/1 :- Pl or domain_2:-Pl,N:n11.
CDPL/1
process RecA(cI,cI_part1,cI_part2)
 bound(cI,_,_).transform(cI,cI_part1,cI_part2).
 release(_,cI_part1,cI_part2).
end process
end CDPL/1.
```

Заключение

Таким образом, языки CDPL/1 и CDPL/HL совместно с моделью молекулярно-биологического вычислительного устройства являются хорошим средством описания логической структуры молекулярно-биологических процессов, происходящих в живой клетки. Однако их достоинства на этом не исчерпываются. Несмотря на то, что они ориентированы на решающие процедуры, реализованные в клеточном устройстве, можно создать компилятор с этих языков и для обыкновенного компьютера, что сразу открывает для нас большие возможности.

Summary

D.S. Tarasov, N.I. Akberova. Molecular biological computational device: organization principles.

Motivation: It's widely accepted that any information processing system can be viewed from two sides: "programmer view" and "engineer view". While engineer deals with physical structure of the system, programmer is interested only in its functional organization that can be relatively independent from system's physical structure. In order to understand any information processing system both points of view must be considered. In this paper we are trying to analyze living cell from programmer's view. Many authors have noticed that the living cell uses some kind of programming language in order to store and express genetic information. This assumes the existence of cell device that reads and executes programs written in the cell language. Understanding the organization of such device could be important for studying regulatory processes in living cell.

Results: based on both biochemical data and formal models of existing human-made computing devices we propose a concept of cell device architecture. This includes the

structure of cell device and a hypothetical language used for its programming. A computer simulator of cell device was designed and the compiler was implemented in order to write programs for this simulator. The functionality of proposed model was tested on the examples of well-known genetic regulation systems.

Литература

1. *Ji S.* The cell as the smallest DNA-based molecular computer // BioSystems. – 1999. – V. 52. – P. 123–133.
2. *Ратнер В.А.* Концепция молекулярно-генетических систем управления. – Новосибирск, 1993.
3. *Ратнер В.А.* Проблемы теоретического исследования генетического языка. Теоретические исследования и банки данных по молекулярной биологии и генетике. – 1988. – С. 130–131.
4. *Ji S.* The linguistics of DNA: words, sentences, grammar, phonetics and semantics // Molecular strategies in biological evolution. – 1999. – V. 870, No 18. – P. 411–417.
5. *Collado-Vides J.* Grammatical model of the regulation of gene expression // Proc. Natl. Acad. Sci. – 1992. – V. 89. – P. 9405–9409.
6. *Collado-Vides J.* A linguistic representation of the range of transcription initiation of [sigma]^{70} promoters: II. Distinctive features of promoters and their regulatory binding sites // Biosystems. – 1993. – V. 29. – P. 105–128.
7. *Claverie J.-M.* Computational methods for the identification of genes in vertebrate genomic sequences // Human Molecular Genetics. – 1997. – V. 6, No 10. – P. 1735–1744.
8. *Gibbons A., Amos M., Hodgson D.,* DNA Computing // Current Opinion in Biotechnology. – 1997. – V. 8. – P. 103–105.
9. *Adleman L.* Molecular computations of solutions to combinatorial problems // Science. – 1994. – V. 266, No 11. – P. 1021–1024.
10. *Kazic T.* After the Turing machine // Proc. Fourth DNA Computing Meeting. – 1998. – P. 131–149.
11. *Мелких А.В.* Детерминистский подход к эволюции жизни // Материалы 1-й Всеросс. научн. internet-конф. «Компьютерное и математическое моделирование в естественных науках». – 2001. – Вып. 1. – С. 11–14.
12. *Clocksin W.F.* Programming in Prolog / Ed. C.S. Mellish. – Berlin: Springer-Verlag, 1984.
13. *Уэно Х., Имидзука М.* Представление и использование знаний. – М.: Мир, 1989.

Поступила в редакцию
10.06.05

Тарасов Денис Станиславович – аспирант Казанского государственного университета.

E-mail: *dtarasov@compnera.com*

Акберова Наталья Ивановна – кандидат биологических наук, старший научный сотрудник Казанского государственного университета.

E-mail: *nakberov@ksu.ru*