ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ПРИКЛАДНАЯ ДИСКРЕТНАЯ МАТЕМАТИКА

Научный журнал

2009 **№3(5)**

Свидетельство о регистрации: ПИ №ФС 77-33762
от 16 октября 2008 г.

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Адрес редакции: 634050, г. Томск, пр. Ленина, 36
E-mail: vestnik_pdm@mail.tsu.ru

В журнале публикуются результаты фундаментальных и прикладных научных исследований отечественных и зарубежных ученых, включая студентов и аспирантов, в области дискретной математики и её приложений в криптографии, компьютерной безопасности, кибернетике, информатике, программировании, теории надежности, интеллектуальных системах.

Периодичность выхода журнала: 4 номера в год.

ООО «Издательство научно-технической литературы»
634050, Томск, пл. Ново-Соборная, 1, тел. (3822) 533-335
Редактор *Н. И. Шидловская*
Верстка *Д. А. Стефанцова*

Изд. лиц. ИД. №04000 от 12.02.2001. Подписано к печати 17.09.2009.
Формат $60 \times 84\frac{1}{8}$. Бумага офсетная. Печать офсетная. Гарнитура «Таймс».
Усл. п. л. 14,03. Уч.-изд. л. 15,71. Тираж 300 экз. Заказ №11.

Отпечатано в типографии «М-Принт», г. Томск, ул. Пролетарская, 38/1

СОДЕРЖАНИЕ

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРИКЛАДНОЙ ДИСКРЕТНОЙ МАТЕМАТИКИ

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

МАТЕМАТИЧЕСКИЕ МЕТОДЫ КРИПТОГРАФИИ

МАТЕМАТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

ПРИКЛАДНАЯ ТЕОРИЯ АВТОМАТОВ

МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ

CONTENTS

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРИКЛАДНОЙ ДИСКРЕТНОЙ МАТЕМАТИКИ

DOI 10.17223/20710410/5/1

УДК 519.7

НИЖНИЕ И ВЕРХНИЕ ОКРЕСТНОСТИ В МНОЖЕСТВЕ С ЗАМЫКАНИЕМ[1]

Н. Г. Парватов

Томский государственный университет, г. Томск, Россия

E-mail: parvatov@mail.tsu.ru

Изучаются свойства операции замыкания в пространстве (множестве с замыканием), обеспечивающие существование для его подмножеств конечных нижних и верхних окрестностей. Доказывается теорема о финитарности пространства, в котором конечно-порождаемые классы обладают конечными нижними окрестностями. Обобщаются известные теоремы А. В. Кузнецова о полноте и С. В. Яблонского о верхних окрестностях. Рассматриваемые вопросы представляют интерес в связи с проблемами полноты и выразимости, а также эффективного задания замкнутых совокупностей в пространствах дискретных функций с замыканием относительно суперпозиции.

Ключевые слова: *проблемы полноты и выразимости, теоремы А. В. Кузнецова и С. В. Яблонского.*

1. Пространство

Пусть в упорядоченной включением системе $\mathcal{B}(P)$ подмножеств множества P (далее будем говорить проще — в множестве P) задана операция замыкания $'$ [1, 2]. Иными словами, для любого подмножества $X \subseteq P$ определено его *замыкание* — подмножество $X' \subseteq P$, причём для любых множеств X и Y из $\mathcal{B}(P)$ выполняются условия

$$X \subseteq X', X' = X'', (X \subseteq Y) \Rightarrow (X' \subseteq Y').$$

В соответствии с [3] множество P с операцией замыкания $'$ в нём, то есть пару $(P, ')$, будем называть *пространством*. Множества элементов пространства, совпадающие со своими замыканиями, называются *замкнутыми множествами* или *замкнутыми классами* пространства. Множество X называется *порождающим* для замкнутого класса X'. Пространство, замкнутыми классами которого являются всевозможные подалгебры некоторой универсальной алгебры, называется *финитарным*. Финитарные пространства охарактеризованы в теореме Г. Биркгофа и О. Фринка [4], из которой, в частности, следует, что *пространство тогда и только тогда финитарно, когда в нём объединение любой направленной вверх системы замкнутых классов замкнуто*.

Пространства комбинаторных объектов (функций, графов, разбиений и др.) возникают в различных областях дискретной математики, но рассматриваемые далее вопросы наиболее важные приложения имеют в теории функциональных систем, при изучении некоторых функциональных и логических пространств.

[1]Работа выполнена в рамках реализации ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009–2013 годы.

2. Проблема выразимости и нижние окрестности

В пространстве $(P, ')$ для любого подмножества $X \subseteq P$ подмножество $Y \subseteq P$ будем называть *X-мажорирующим*, *X-минорирующим* или *X-порождающим*, если выполняется соответствующее включение:

$$X' \subseteq Y', X' \supseteq Y' \text{ или } X' = Y'.$$

Множество X элементов пространства будем называть *конечно-порождаемым*, если существует конечное X-порождающее множество.

Проблема выразимости множества X в пространстве $(P, ')$ состоит в описании (в приложениях это описание должно быть по возможности эффективным) всех X-мажорирующих подмножеств этого пространства. Эта проблема возникает при изучении дискретных функций с операциями суперпозиции. Естественным средством решения этой проблемы является указание некоторой такой системы $\mathcal{S}$ замкнутых подмножеств пространства $(P, ')$, что для любого множества $Y \in \mathcal{B}(P)$ включение $X' \subseteq Y'$ равносильно отсутствию в $\mathcal{S}$ класса, включающего Y. Систему $\mathcal{S}$, обладающую указанным свойством, будем называть *нижней окрестностью* множества X в пространстве $(P, ')$. Иными словами, нижняя окрестность множества X — это всякая система замкнутых классов пространства, не включающих X, такая, что всякий, не включающий X замкнутый класс, можно расширить до класса из этой системы.

Отметим некоторые свойства нижних окрестностей. Во-первых, ясно, что каждое множество X элементов пространства обладает нижней окрестностью, например тривиальной, состоящей из всех не включающих X замкнутых классов пространства. Далее, максимальные по включению замкнутые классы пространства среди классов, не включающих X, станем называть *X-максимальными*, а их совокупность станем обозначать через $\mathcal{S}(X)$, а также через $\mathcal{S}(x)$ в случае одноэлементного множества $X = \{x\}$. Несложно понять, что система $\mathcal{S}(X)$ включена во всякую нижнюю окрестность множества X. Нижнюю окрестность, состоящую из попарно не сравнимых по включению классов, будем называть *безызбыточной*. Иными словами, безызбыточная нижняя окрестность перестаёт быть нижней окрестностью после удаления из неё любого класса. В случае своего существования безызбыточная нижняя окрестность множества X совпадает с системой $\mathcal{S}(X)$ и является наименьшей по включению нижней окрестностью множества X. Из конечной нижней окрестности множества X, опять в случае её существования, можно выделить безызбыточную нижнюю окрестность $\mathcal{S}(X)$. Множество X, обладающее конечной нижней окрестностью, оказывается конечно-порождаемым, так как элементы X-порождающего множества можно выбрать из дополнений $X \setminus K$ для всевозможных классов $K \in \mathcal{S}(X)$.

Представляют интерес условия, при которых конечно-порождаемые подмножества пространства $(P, ')$ имеют конечные нижние окрестности. Проблема поиска таких условий в более слабой форме была поставлена и частично решена в [3]. Имеет место

Теорема 1. Пространство, в котором каждое конечно-порождаемое подмножество имеет конечную нижнюю окрестность, финитарно.

Доказательство. Рассмотрим направленную вверх систему $\mathcal{S}$ замкнутых множеств пространства $(P, ')$. Очевидно включение $\cup\mathcal{S} \subseteq (\cup\mathcal{S})'$. Докажем обратное включение. С этой целью из множества $(\cup\mathcal{S})'$ выберем произвольно элемент a. Отметим, что объединение $\cup\mathcal{S}$ является $\{a\}$-мажорирующим. Тогда если множество $\{a\}$ (очевидно, конечно-порождённое) имеет конечную нижнюю окрестность $\mathcal{S}(a)$, то система $\mathcal{S}$

обладает следующим свойством: её объединение не включено ни в один из классов системы $\mathcal{S}(a)$. В силу конечности системы $\mathcal{S}(a)$ этим свойством обладает уже некоторая конечная подсистема $\mathcal{R} \subseteq \mathcal{S}$. В силу конечности системы $\mathcal{R}$ и направленности системы $\mathcal{S}$ в системе $\mathcal{S}$ в качестве элемента содержится класс, включающий объединение $\cup\mathcal{R}$ системы $\mathcal{R}$. Этот класс, обозначим его буквой K, не включён ни в один из классов системы $\mathcal{S}(a)$, а потому является $\{a\}$-мажорирующим. Так как класс K замкнут (вместе со всеми классами системы $\mathcal{S}$), то имеют место включения $\{a\} \subseteq K' = K \subseteq \cup\mathcal{S}$. В силу выбора a доказано равенство $\cup\mathcal{S} = (\cup\mathcal{S})'$. ■

Обратная теорема не имеет места: пространство бесконечной циклической группы с замыканием относительно групповых операций финитарно, тем не менее в нём собственные подгруппы не имеют конечных нижних окрестностей. Можно утверждать лишь, что в пространстве с финитарным замыканием конечно-порождаемые подмножества имеют безызбыточные нижние окрестности [2].

В связи с теоремой 1 представляют интерес конструктивные достаточные условия, при которых в финитарном пространстве конечно-порождаемое подмножество имеет конечную нижнюю окрестность. Подобные условия обобщают теорему А. В. Кузнецова о полноте из [5]. Эта теорема неоднократно обобщалась и передоказывалась [6 – 9], в том числе самим А. В. Кузнецовым в [3]. Несмотря на это, задача поиска подобных обобщений, поставленная в [3], не утратила своей актуальности и по сей день. Она актуальна в связи с (возможным и действительным) появлением новых классов управляющих систем, приводящим к необходимости решать проблемы полноты и выразимости в новых постановках.

3. Предупорядоченное пространство

Пусть в множестве P определено (рефлексивное и транзитивное) отношение предпорядка $\leqslant$ [10]. Множество X элементов из P будем называть *наследственным классом* в предупорядоченном множестве $(P, \leqslant)$, если вместе с любым своим элементом x множество X содержит и всякий элемент $y \in P$, такой, что $y \leqslant x$. Наследственные множества комбинаторных объектов возникают естественным образом в различных областях дискретной математики, таких, как теория графов, теория дискретных функций, математическая логика и др. Наследственное множество X можно задать указанием некоторого такого подмножества $Y \subseteq X$, что X является наименьшим по включению наследственным классом среди наследственных классов, включающих Y. В этой ситуации множество Y будем называть *порождающим* для наследственного класса X, а наследственный класс X будем обозначать через $Y_{\leqslant}$. Наследственное множество X можно задать также указанием некоторого такого подмножества $Y \subseteq P \setminus X$, что X является наибольшим по включению наследственным классом среди наследственных классов, имеющих пустое пересечение с множеством Y. В этом случае X состоит из всевозможных таких элементов x из P, что для любого элемента $y \in Y$ не выполняется неравенство $y \leqslant x$. В этой ситуации множество Y будем называть *запрещающим* для X в $(P, \leqslant)$, а наследственный класс X будем обозначать через $P \setminus Y_{\geqslant}$.

Следует понимать, что дополнения наследственных классов предупорядоченного множества $(P, \leqslant)$ сами являются наследственными классами в множестве $(P, \geqslant)$ с двойственным предупорядочением. При этом запрещающее множество наследственного класса X в предупорядоченном множестве $(P, \leqslant)$ является порождающим для наследственного класса $P \setminus X$ в $(P, \geqslant)$. Этот факт согласуется с введёнными выше обозначениями.

Также полезно знать, что объединения и пересечения наследственных классов сами являются наследственными классами. Причём, если задана система наследственных классов и для каждого класса указаны некоторое порождающее и некоторое запрещающее множества, то объединение этих порождающих множеств будет порождающим множеством объединения наследственных классов системы, а объединение запрещающих множеств будет запрещающим для пересечения наследственных классов системы.

Предупорядочение $\leqslant$ и замыкание $'$ станем называть по отношению друг к другу *согласованными*, если замкнутые классы пространства $(P, ')$ являются одновременно наследственными классами предупорядоченного множества $(P, \leqslant)$. Равносильно, согласованность означает выполнение импликации

$$(x \leqslant y) \Rightarrow (\{x\}' \subseteq \{y\}')$$

для любых элементов x и y из P. При выполнении обратных импликаций предупорядочение $\leqslant$ называется *индуцированным* замыканием $'$.

Множество P, рассматриваемое вместе с определёнными в нём согласованными по отношению друг к другу замыканием $'$ и предупорядочением $\leqslant$, то есть тройку $(P, ', \leqslant)$ станем называть *предупорядоченным пространством*. При этом о замкнутых классах пространства $(P, ')$ и наследственных классах предупорядоченного множества $(P, ')$ будем говорить как о замкнутых и наследственных классах предупорядоченного пространства $(P, ', \leqslant)$. В предупорядоченном пространстве замкнутые классы являются одновременно наследственными и, следовательно, допускают задание посредством запрещающих множеств. В приложениях подобный способ задания часто оказывается эффективным (по терминологии книги [10] — «хорошим»). В связи с этим в дальнейшем наряду с прочими проблемами будем рассматривать проблему поиска условий, при которых замкнутый класс в предупорядоченном пространстве имеет конечное запрещающее множество.

4. Обобщённая теорема А. В. Кузнецова

Имеет место

Лемма 1. Пусть для некоторых множеств B_1 и B_2 и для любого множества X из $\mathcal{B}(P)$ выполняется включение

$$X' \cap B_1 \subseteq (X_{\leqslant} \cap B_2)'. \tag{1}$$

Тогда в предупорядоченном пространстве $(P, ', \leqslant)$ для любого замкнутого класса Q, такого, что $Q = (Q \cap B_1)'$, всякий класс Q_1 из $\mathcal{S}(Q)$ обладает конечным запрещающим множеством $B_2 \setminus Q_1$, то есть имеет место равенство $P \setminus Q_1 = (B_2 \setminus Q_1)_{\geqslant}$.

Доказательство. Включение $(B_2 \setminus Q_1)_{\geqslant} \subseteq P \setminus Q_1$ следует из согласованности замыкания с предупорядочением. Докажем обратное включение. Выберем произвольно элемент a в множестве $P \setminus Q_1$ и покажем, что наследственный класс $\{a\}_{\leqslant}$ имеет непустое пересечение с $B_2 \setminus Q_1$ (тогда элемент a принадлежит $(B_2 \setminus Q_1)_{\geqslant}$, и желаемое включение доказано). Предположим противное. Тогда $\{a\}_{\leqslant} \cap B_2 \subseteq Q_1$. Используя это включение, Q-максимальность множества Q_1 и включение (1), получаем последовательность включений

$$Q \cap B_1 \subseteq (Q_1 \cup \{a\})' \cap B_1 \subseteq ((Q_1 \cup \{a\})_{\leqslant} \cap B_2)' = ((Q_{1\leqslant} \cup \{a\}_{\leqslant}) \cap B_2)' =$$

$$= ((Q_{1\leqslant} \cap B_2) \cup (\{a\}_{\leqslant} \cap B_2))' \subseteq Q_1,$$

откуда $Q = (Q \cap B_1)' \subseteq Q_1$. Полученное противоречие завершает доказательство леммы. ■

В связи с леммой 1 согласованные в множестве P предупорядочение $\leqslant$ и замыкание $'$ станем называть *сильно согласованными*, если множество P является объединением направленной вверх системы $\mathcal{N}$ конечных подмножеств и существует функция $\alpha : \mathcal{N} \to \mathcal{N}$, такая, что для любых множеств $B_1 \in \mathcal{N}$, $B_2 = \alpha(B_1)$ и $X \in \mathcal{B}(P)$ выполняется включение (1). Также будем говорить в этой ситуации, что предупорядочение $\leqslant$ сильно согласовано с замыканием $'$ при помощи системы $\mathcal{N}$ и отображения α. Предупорядоченное пространство, в котором предупорядочение сильно согласовано с замыканием, будем называть *сильно предупорядоченным*.

Поскольку в финитарном пространстве конечно-порождаемые подмножества имеют безызбыточные нижние окрестности, следствием леммы 1 является

Теорема 2. В финитарном сильно предупорядоченном пространстве всякое конечно-порождаемое подмножество обладает конечной нижней окрестностью, каждый класс которой имеет конечное запрещающее множество.

Теорема 2 указывает достаточные условия существования конечных нижних окрестностей у конечно-порождаемых подмножеств пространства, чем обобщает теорему А. В. Кузнецова о полноте из [5]. Вместе с тем в теореме речь идёт не просто о существовании конечных нижних окрестностей, но о существовании нижних окрестностей, состоящих из классов с конечными множествами запретов. Как указывалось, в приложениях задание замкнутых классов конечными запрещающими множествами оказывается эффективным. Тем самым теорема 2 усиливает теорему А. В. Кузнецова.

5. Запрещающие множества и верхние окрестности

В интересующих нас приложениях существование конечного запрещающего множества у замкнутого класса предупорядоченного пространства является благоприятным, позволяя получить эффективный алгоритм распознавания принадлежности этому классу. В связи с этим представляют интерес условия, при которых замкнутый класс имеет в предупорядоченном пространстве конечное запрещающее множество. Желая найти некоторые такие условия, введём в рассмотрение (двойственное к понятию нижней окрестности) понятие верхней окрестности множества X в пространстве. Именно, станем называть так систему $\mathcal{H}$ замкнутых классов пространства, не включённых в X', если всякий не включённый в X' замкнутый класс пространства можно сузить (удаляя элементы) до класса из $\mathcal{H}$.

Отметим некоторые свойства. Ясно, что всякое множество X элементов пространства обладает верхней окрестностью, например тривиальной, состоящей из всех не включённых в X' замкнутых классов. Более того, всякое множество X обладает верхней окрестностью

$$\mathcal{H}_Y = \{\{y\}' | y \in Y\}$$

для некоторого множества $Y \subseteq P \setminus X'$. Эта верхняя окрестность состоит из замкнутых классов с одноэлементными порождающими множествами. Её можно получить из произвольной верхней окрестности $\mathcal{H}$, выбрав в множество Y по элементу из дополнений $Z \setminus X'$, где $Z \in \mathcal{H}$. Безызбыточная (состоящая из попарно не сравнимых по включению классов) верхняя окрестность в случае своего существования обязана совпадать с системой $\mathcal{H}(X)$ всех X-минимальных классов пространства. При этом X-минимальным называем минимальный по включению замкнутый класс среди подмножеств пространства, не включённых в X'. Вне зависимости от того, образуют X-минимальные классы

верхнюю окрестность множества X или нет, они обладают, очевидно, одноэлементными порождающими множествами (поскольку оказываются минимальными по включению классами в любой верхней окрестности вида $\mathcal{H}_Y$).

Из конечной нижней окрестности, опять в случае её существования, можно выделить безызбыточную.

Заметим, что система $\mathcal{H}_Y$ является верхней окрестностью множества X в предупорядоченном пространстве, если множество Y является запрещающим для замкнутого класса X'. Если же предупорядочение пространства индуцировано замыканием, то верно обратное. Именно, при индуцированном предупорядочении для любого множества $Y \subseteq P \setminus X'$ система $\mathcal{H}_Y$ тогда и только тогда является верхней окрестностью замкнутого класса X', когда для него множество Y является запрещающим. Из сказанного следует, что существование конечного запрещающего множества для замкнутого класса в предупорядоченном пространстве влечёт существование у замкнутого класса конечной верхней окрестности, а при индуцированном предупорядочении равносильно существованию конечной верхней окрестности. Представляется интересной задача выявления условий, при которых в предупорядоченном пространстве замкнутые классы с конечными верхними окрестностями обладают конечными запрещающими множествами. Подобные условия обобщают теорему С. В. Яблонского из [11] о верхней окрестности. Как будет показано далее, подобные условия выполняются в сильно предупорядоченном пространстве.

Лемма 2. Пусть $Y \subseteq P$ и система $\mathcal{H}_Y$ является верхней окрестностью замкнутого класса X в сильно предупорядоченном пространстве $(P, ', \leqslant)$. Тогда пересечение всех включающих X $\{y\}$-максимальных классов, где $y \in Y$, совпадает с X.

Доказательство. Ясно, что это пересечение включает X. Покажем, что включение не может быть строгим. Для этого выберем произвольно не принадлежащий X элемент z пространства. Поскольку множество Y является запрещающим для X при индуцированном предупорядочении, в множестве Y найдётся элемент y, такой, что $\{y\}' \subseteq \{z\}'$. По теореме 2 система $\mathcal{S}(y)$ является нижней окрестностью множества X. По этой причине ($\{y\}$-мажорирующее) множество $\{z\}$ не содержится ни в одном из классов этой системы, в отличие от (не $\{y\}$-мажорирующего) множества X, содержащегося в каком-то из классов системы. Отсюда z не содержится в рассматриваемом пересечении и лемма доказана. ■

Следствием этой леммы является

Теорема 3. В сильно предупорядоченном пространстве замкнутые классы с конечной верхней окрестностью обладают конечными запрещающими множествами.

Доказательство. Рассмотрим в сильно предупорядоченном пространстве замкнутый класс X, обладающей конечной верхней окрестностью. Тогда его конечной верхней окрестностью оказывается система $\mathcal{H}(X) = \mathcal{H}_Y$ для некоторого конечного подмножества $Y \subseteq P \setminus X$. В силу леммы 2 запрещающим множеством для X является объединение F запрещающих множеств F_K для всевозможных включающих X классов K из $\mathcal{S}(y)$, таких, что $y \in Y$. Запрещающие множества F_K можно выбрать конечными в силу теоремы 2. Тогда конечным окажется и запрещающее множество F замкнутого класса X. ■

6. Пример использования: замыкание относительно суперпозиции

Условимся через P_E обозначать множество, состоящее из всех функций $f : E^n \to E$ при всевозможных натуральных n. Это множество будем рассматривать вместе с согласованными в нём замыканием относительно суперпозиции [5–7] и *предупорядочением* $\leqslant$ *относительно отождествления* (переменных), таким, что неравенство $f \leqslant g$ означает возможность получить функцию f из функции g отождествлением переменных. Замыкание множества X функций из P_E, обозначаемое через $[X]$, состоит из всевозможных функций, вычисляемых термами, отличными от переменных, в которых функциональные символы интерпретируются в множестве X, а переменные принимают значения в множестве E. В действительности, замыкание относительно суперпозиции сильно согласовано с предупорядочением относительно отождествления. Это вытекает из следующей леммы, в которой через $P_E^{(n)}$ обозначено множество всех функций в P_E, зависящих не более чем от n переменных.

Лемма 3. Для любых натуральных n и $m = |E|^{|E|^n}$ и любого подмножества $X \subseteq P_E$ имеет место включение

$$[X] \cap P_E^{(n)} \subseteq [X_{\leqslant} \cap P_E^{(m)}].$$

Доказательство. Функция $f \in [X] \cap P_E^{(n)}$ вычисляется термом, составленным из функциональных символов функций из X и символов переменных $x_1, \ldots, x_n$. В некотором таком терме рассмотрим подтерм $g(H_1, \ldots, H_r)$, где g — символ функции из X, а $H_1, \ldots, H_r$ — некоторые термы, вычисляющие, очевидно, какие-то функции из $P_E^{(n)}$. Если какие-то термы H_i вычисляют одинаковые функции, то можно избавиться от этого, отождествив соответствующие переменные у функции g, то есть заменив её некоторой функцией из $X_{\leqslant}$ и удалив лишние подформулы. Проделав это пока возможно, получим терм для f, в образовании которого участвуют символы функций из множества $X_{\leqslant}$, зависящих не более чем от $m = |P_E^{(n)}| = |E|^{|E|^n}$ переменных. ■

Непосредственно из леммы получаем

Следствие 1. В множестве P_E замыкание относительно суперпозиции и предупорядочение относительно отождествления сильно согласованы посредством системы $\mathcal{N}$ и отображения α, где $\mathcal{N}$ состоит из множеств $P_E^{(n)}$ при натуральных n и $\alpha(P_E^{(n)}) = P_E^{(m)}$ для $m = |E|^{|E|^n}$.

Отсюда с использованием теорем 2 и 3 можно получить теорему А. В. Кузнецова о полноте из [3], а также теорему С. В. Яблонского о верхних окрестностях из [11]. Рассмотренный пример отнюдь не исчерпывает область применения теорем 2 и 3.

7. Пример использования: алгебра предикатов

Станем обозначать через Π_E множество всевозможных предикатов $p : E^n \to \{\text{И}, \text{Л}\}$, где E, как и прежде, фиксированное конечное множество, а n принимает всевозможные натуральные значения. Множество Π_E будем рассматривать с $(\exists, \wedge)$-замыканием, понимая под ним замыкание относительно операций конъюнкции и проектирования, а также операций отождествления и перестановки переменных [12]. При этом $(\exists, \wedge)$-замыкание множества X предикатов из Π_E обозначается через $[X]$. Множество $[X]$ состоит из предикатов, выражаемых формулами первого порядка, в которых предикатные символы интерпретируются в множестве X, предметные переменные интерпретируются в множестве E, а логические символы принадлежат множеству

$\{\exists, \wedge\}$. В силу возможного переобозначения связанных переменных и известного логического закона (именно следующего закона: $\exists x A \wedge B \equiv \exists x (A \wedge B)$, если x не входит в B), в предыдущей фразе слово «формулами» можно заменить фразой «предварёнными формулами». Также в множестве Π_E будем рассматривать согласованное с $(\exists, \wedge)$-замыканием *предупорядочение* $\leqslant$ *относительно отождествления* (переменных), при котором неравенство $p \leqslant q$ означает возможность получить предикат p из предиката q отождествлением переменных. На самом деле, предупорядочение сильно согласовано с замыканием в множестве Π_E в силу следующей леммы.

Лемма 4. Для любых натуральных n и $m = |E|^{|E|^n}$ и любого подмножества $X \subseteq \Pi_E$ имеет место включение

$$[X] \cap \Pi_E^{(n)} \subseteq [X_{\leqslant} \cap \Pi_E^{(m)}].$$

Здесь через $\Pi_E^{(n)}$ обозначено множество всех предикатов в Π_E, имеющих не более n аргументов.

Доказательство. Рассмотрим предикат p, принадлежащий множеству $[X] \cap \Pi_E^{(n)}$, то есть зависящий не более чем от n переменных, для определённости $x_1, \ldots, x_n$, и выражающийся в силу сделанного ранее замечания некоторой предварённой формулой вида

$$\exists x_{n+1} \ldots \exists x_{n+s} (p_1 \wedge \cdots \wedge p_r). \tag{2}$$

В ней предикаты p_i принадлежат множеству X и зависят (для определённости) от переменных из следующего списка: $x_1, \ldots, x_{n+s}$. Рассмотрим далее всевозможные наборы $ab = (a_1, \ldots, a_{n+s})$ значений этих переменных в множестве E, делающие истинной бескванторную часть $p_1 \wedge \cdots \wedge p_r$ формулы (2). Удалим часть этих наборов, оставив в рассмотрении наборы с различными и всеми возможными началами $a = (a_1, \ldots, a_n)$ из E^n. Если выполняется неравенство $n + s > m$ (где $m = |E|^{|E|^n}$), то у рассматриваемых наборов какие-то две компоненты, скажем i-я и j-я (где $1 \leqslant i < j \leqslant n + s$), совпадают. Это позволяет в формуле (2) соответствующие этим компонентам переменные x_i и x_j отождествить, заменив их одной буквой x_i (и удалив, если потребуется, ставшие бесполезными кванторные приставки). Проделаем это, пока возможно. После этого осталось заменить предикаты p_i, входящие с одинаковыми переменными, равносильными предикатами из множества $X_{\leqslant} \cap \Pi_E^{(m)}$, входящими без одинаковых переменных. Полученная в результате формула равносильна исходной и выражает всё тот же предикат p. Тем самым лемма доказана. ■

Следствие 2. В множестве Π_E $(\exists, \wedge)$-замыкание и предупорядочение относительно отождествления сильно согласованы посредством системы $\mathcal{N}$ и отображения α, где $\mathcal{N}$ состоит из множеств $\Pi_E^{(n)}$ при натуральных n и $\alpha(P_E^{(n)}) = P_E^{(m)}$ для $m = |E|^{|E|^n}$.

Рассмотренное пространство Π_E представляет интерес в связи с функциями многозначной логики в силу соответствия Галуа из [12], определяемого отношением сохранения функцией предиката. Используя это соответствие Галуа и теоремы 2 и 3, из следствия 2 можно снова получить как теоремы С. В. Яблонского о верхних окрестностях из [11], так и теорему А. В. Кузнецова о полноте из [3].

ЛИТЕРАТУРА

1. *Курош А. Г.* Лекции по общей алгебре. СПб.: Лань, 2005.
2. *Кон П.* Универсальная алгебра. М.: Мир, 1968.

3. *Кузнецов А. В.* Структуры с замыканием и критерии функциональной полноты // Успехи матем. наук. 1961. Т. XVI. № 2 (98). С. 201–202.
4. *Birkhoff G., Frink O.* Representations of lattices by sets // Transactions on American mathematical society. 1948. V. 64. P. 299–316.
5. *Яблонский С. В.* Функциональные построения в k-значной логике // Труды матем. ин-та им. Стеклова. 1958. Т. 51. С. 5–142.
6. *Мальцев А. И.* Итеративные алгебры Поста. Новосибирск: Изд-во НГУ, 1976.
7. *Мальцев А. И.* Итеративные алгебры и многообразия Поста // Алгебра и логика. 1966. Т. 5. № 2. С. 5–24.
8. *Парватов Н. Г.* Замечания о конечной порождаемости замкнутых классов // Дискрет. анализ и исслед. операций. Сер. 1. 2004. Т. 11. № 3. С. 32–47.
9. *Парватов Н. Г.* Наследственные системы дискретных функций // Там же. Сер. 2. 2007. Т. 14. № 2. С. 76–91.
10. *Дистель Р.* Теория графов. Новосибирск: Изд-во Ин-та математики, 2002.
11. *Яблонский С. В.* О строении верхней окрестности для предикатно-описуемых классов в P_k // Докл. АН СССР. 1974. Т. 218. № 2. С. 302–307.
12. *Боднарчук В. Г., Калужнин Л. А., Котов В. Н., Ромов Б. А.* Теория Галуа для алгебр Поста // Кибернетика. 1969. № 3. С. 1–10; № 5. С. 1–9.

2009 Теоретические основы прикладной дискретной математики №3(5)

DOI 10.17223/20710410/5/2

УДК 681.03

ПОЧТИ СОВЕРШЕННЫЕ НЕЛИНЕЙНЫЕ ФУНКЦИИ[1]

М. Э. Тужилин

Российский государственный гуманитарный университет, г. Москва, Россия

E-mail: mtmt@rambler.ru

Сделан обзор публикаций, посвященных почти совершенным нелинейным функциям, их свойствам и построению.

Ключевые слова: *почти совершенные нелинейные функции, почти бент-функции, криптография.*

1. PN-функции

Появление в открытой криптографии разностного метода (Differential cryptanalysis [1]) вызвало необходимость разработки подходов к построению тех классов S-боксов, которые являются оптимальными для противостояния данному методу. Одной из работ, посвященной данной проблеме и оказавшей существенное влияние на дальнейшие исследования, явилась статья K. Nyberg 1991 г. [2]. В ней было предложено два подхода к построению оптимальных S-боксов, появилось понятие совершенной нелинейной функции (Perfect Nonlinear, или, сокращенно, PN-функции).

Пусть далее натуральные m и n связаны соотношением $m \leqslant n$.

Функция $F\colon (Z/q)^n \to (Z/q)^m$ называется **совершенной нелинейной**, если для любого фиксированного ненулевого $a \in (Z/q)^n$ функция $F(x+a) - F(x)$ (ее называют производной по направлению a) принимает каждое значение из $(Z/q)^m$ ровно при q^{n-m} значениях x.

В [2, 3] приведен ряд результатов относительно PN-функций, отмечена связь их с бент-функциями, в частности приведена теорема о том, что PN-функция $F\colon (Z/q)^n \to (Z/q)$ является бент-функцией; обратное верно, если q — простое.

Однако требование к функции быть совершенной нелинейной оказалось слишком жестким, число таких функций очень невелико, а для многих важных для практики значений q и n они просто не существуют. Поэтому большее внимание уделяется введенному в [4] в 1992 г. понятию почти совершенной нелинейной функции (Almost Perfect Nonlinear или, сокращенно, APN-функции).

2. APN-функции

Всюду далее поле $\mathrm{GF}(p^n)$ отождествляется с векторным пространством $\mathrm{GF}(p)^n$.

Функция $F\colon \mathrm{GF}(p^n) \to \mathrm{GF}(p^m)$ называется **APN-функцией**, если для любых $a \neq 0$ из $\mathrm{GF}(p^n)$ и b из $\mathrm{GF}(p^m)$ уравнение $F(x+a) - F(x) = b$ имеет не более двух решений.

Строго говоря, в [4] определение APN-функции дано было для подстановок, заданных на $\mathrm{GF}(2^n)$, но в последующих работах это определение было расширено.

В [5] изложен подход к характеризации PN- и APN-функций на основе теории полуполей.

[1] Результаты работы докладывались на Международной конференции с элементами научной школы для молодёжи, г. Омск, 7–12 сентября 2009 г.

В [3] K. Nyberg ввела и более общее понятие — функции, названной ею differentially δ-uniform, для которой в приведенном выше определении APN-функции вместо двух решений уравнения рассматривается произвольное натуральное число δ. Таким образом, differentially δ-uniform функция F является PN-функцией, если $\delta = 1$, и APN-функцией, если $\delta = 2$.

Наибольший интерес исследователей прикован к случаю $p = 2$.

Как показано в [6], отношение числа APN-функций $\mathrm{GF}(2^n) \to \mathrm{GF}(2^n)$ степени d при $d \equiv 0, 3 \pmod 4$ к числу всех функций $\mathrm{GF}(2^n) \to \mathrm{GF}(2^n)$ степени d стремится к нулю с ростом n.

Многие свойства APN-функций, заданных на $\mathrm{GF}(2^n)$, описаны в обзоре [7]. Основным инструментом исследований свойств функций является их спектр, состоящий из коэффициентов Фурье. Отдельные разделы обзора посвящены квадратичным функциям (то есть функциям, чья алгебраическая степень не превышает числа 2), степенным функциям $F\colon \mathrm{GF}(p^n) \to \mathrm{GF}(p^n)$ вида $F(x) = x^d$ и APN-подстановкам — таким APN-функциям, которые являются подстановками на $\mathrm{GF}(p^n)$. Приведен список открытых проблем в области исследования APN-функций.

3. APN-подстановки

В [8] представлено три подхода к построению APN-подстановок, действующих на $\mathrm{GF}(2^n)$. Однако все они относятся к случаю нечетного n. Вопрос о существовании APN-подстановок при четном n до сих пор остается открытой проблемой. В [3] доказано, что среди квадратичных функций таких быть не может. Позднее было доказано, что подстановка F, действующая на $\mathrm{GF}(2^n)$, не может быть APN-подстановкой, если $n = 2$ или $n = 4$.

Обратная к APN-подстановке — тоже APN-подстановка.

В [9] изучаются подстановки вида $G(x) + af(x)$, где $f(x)$ — булева функция, $a \in \mathrm{GF}(2^n)$, а $G(x)$ — подстановка или линейная функция $\mathrm{GF}(2^n) \to \mathrm{GF}(2^n)$.

В силу того, что APN-подстановок при четном n до сих пор найти не удалось, актуально изучение differentially δ-uniform подстановок F с $\delta = 4$. Отметим, что именно такая подстановка используется в алгоритме AES.

4. AB-функции

Напомним, что с любой функцией $F\colon \mathrm{GF}(2^n) \to \mathrm{GF}(2^n)$ можно связать n координатных булевых функций и множество их линейных комбинаций, называемых компонентами функции F.

Понятие APN-функции тесно связано с понятием почти бент-функции (Almost Bent, или AB-функции), введенном в [10].

Функция $F\colon \mathrm{GF}(2^n) \to \mathrm{GF}(2^n)$ называется AB-функцией, если минимальное по Хеммингу расстояние между всеми компонентами функции F и всеми аффинными функциями максимально.

Это минимальное расстояние называется показателем нелинейности функции F, и максимально возможное значение его равно $2^{n-1} - 2^{\frac{n-1}{2}}$ [11]. AB-функции в отличие от APN-функций существуют только при нечетном n и являются наиболее стойкими по отношению к линейному криптоанализу. Кроме того, любая AB-функция является APN-функцией, и при нечетном n любая квадратичная функция является APN-функцией тогда и только тогда, когда она является AB-функцией [12]. Эта работа, появившаяся в 1998 г., стала важной вехой на пути исследований APN-функций. В ней,

в частности, отмечается связь APN-функций с линейными кодами длины $2^n - 1$, имеющими оптимальное кодовое расстояние.

5. Степенные функции

Относительно степенных функций, заданных на $\mathrm{GF}(2^n)$, в [12] доказано, что если h делит n и $d = k(2^h - 1) + 2^r$ при целых k и r, то степенная функция $F(x) = x^d$ не может быть APN-функцией. Отметим, что использование степенных функций в качестве S-боксов, с одной стороны, привлекательно для разработчиков криптосхем, так как это упрощает реализацию криптоалгоритмов, но, с другой стороны, может привести к построению криптоаналитических методов, основанных на «простом» виде функции.

6. Преобразования, сохраняющие свойства APN и AB

Для построения новых APN- или AB-функций из имеющихся представляют большой интерес те преобразования функций, при которых сохраняются свойства «быть APN-функцией» или «быть AB-функцией» соответственно.

Пусть F, A, A_1, A_2 суть функции $\mathrm{GF}(p^n) \to \mathrm{GF}(p^n)$. Если F есть APN-функция, A_1, A_2 — аффинные подстановки и A — аффинная функция, то $F' = A_1 \circ F \circ A_2 + A$ — тоже APN-функция. В этом случае F и F' называются расширенно аффинно эквивалентными (**EA-эквивалентными**) и аффинно эквивалентными при $A = 0$.

До недавнего времени известны были только такие APN-функции, которые EA-эквивалентны степенным функциям $F(x) = x^d$. Только в 2006 г. было доказано [13], что функции вида $x^3 + ux^{36}$, заданные на полях $\mathrm{GF}(2^{10})$ и $\mathrm{GF}(2^{12})$, при определенных элементах u из этих полей являются APN-функциями, не EA-эквивалентными степенным функциям. В табл. 1 приведены все известные на данный момент серии APN-функций, EA-эквивалентных степенным функциям, заданным на поле $\mathrm{GF}(p^n)$.

Welch and Niho APN-функции из табл. 1 являются AB-функциями. Для нечетного n AB-функциями являются функции Gold и Kasami.

В [12] было введено понятие Carlet-Charpin-Zinoviev эквивалентности (CCZ-эквивалентности), а именно: отображения F_1 и F_2 поля $\mathrm{GF}(p^n)$ на себя называются **CCZ-эквивалентными**, если графы отображений F_1 и F_2, то есть множества $\{(x, F_1(x)) \mid x \in \mathrm{GF}(p^n)\}$ и $\{(x, F_2(x)) \mid x \in \mathrm{GF}(p^n)\}$ являются аффинно эквивалентными.

Из определения следует, что отображения F_1 и F_2 CCZ-эквивалентны тогда и только тогда, когда существует аффинный автоморфизм $L = (L_1, L_2)$ декартова произведения $\mathrm{GF}(p^n) \times \mathrm{GF}(p^n)$, такой, что выполнено

$$y = F_1(x) \Leftrightarrow L_2(x, y) = F_2(L_1(x, y)).$$

CCZ-эквивалентность является более общей, чем EA-эквивалентность (любой класс EA-эквивалентности содержится в некотором классе CCZ-эквивалентности) и сохраняет свойства «быть APN-функцией» и «быть AB-функцией». Для бент-функций эти эквивалентности совпадают.

В [14] описана связь между EA-эквивалентностью и CCZ-эквивалентностью для функций $\mathrm{GF}(2^n) \to \mathrm{GF}(2^m)$. Показано, что для булевых функций (то есть при $m = 1$) эти понятия совпадают — две булевы функции CCZ-эквивалентны тогда и только тогда, когда они EA-эквивалентны.

В табл. 2 приведены серии APN-функций, CCZ-эквивалентных степенным функциям.

Таблица 1

APN-функции, EA-эквивалентные степенным функциям

Название функции	Экспонента d	Условия	Источник
Kloosterman	p^n-2	$p=2$ и n нечетно, или $p>2$ и $p\equiv 2(\mathrm{mod}\ 3)$	[3, 8]
Gold	2^i+1	$p=2$, НОД(i,n)=1	[15]
Kasami	$2^{2i}-2^i+1$	$p=2$, НОД(i,n)=1	[16]
Welch	2^t+3	$p=2$, $n=2t+1$	[17]
Niho	$2^t+2^{t/2}-1$, t — четное, $2^t+2^{\frac{3t+1}{2}}-1$, t — нечетное	$p=2$, $n=2t+1$	[18]
Инверсия	$2^{2t}-1$	$p=2$, $n=2t+1$	[3, 8]
Dobbertin	$2^{4i}+2^{3i}+2^{2i}+2^i-1$	$p=2$, $n=5i$	[19]
Helleseth, Sanberg	$\frac{p^n-1}{2}-1$	$p\equiv 3,7\ (\mathrm{mod}\ 20)$, $p^n>7$, $p^n\neq 27$, n нечетно	[20]
Dobbertin, Mills, Muller, Pott, Willems	$\frac{3^{(n+1)/2-1}}{2}$ $\frac{3^{(n+1)-1}}{8}$ $\frac{3^{(n+1)-1}}{8}+\frac{3^n-1}{2}$	$p=3$, $n\equiv 3\ (\mathrm{mod}\ 4)$ $p=3$, $n\equiv 1\ (\mathrm{mod}\ 4)$ $p=3$, $n\equiv 1\ (\mathrm{mod}\ 4)$	[21]
Felke	$\frac{3^{(n+1)/2-1}}{2}+\frac{3^n-1}{2}$	$p=3$, $n\equiv 1\ (\mathrm{mod}\ 4)$	[22]
Helleseth, Rong, Sandberg	$\frac{p^n+1}{4}+\frac{p^n-1}{2}$ $\frac{p^n+1}{4}$ $\frac{2p^n-1}{3}$ p^n-3	$p^n\equiv 3\ (\mathrm{mod}\ 8)$ $p^n\equiv 7\ (\mathrm{mod}\ 8)$ $p^n\equiv 2\ (\mathrm{mod}\ 8)$ $p=3$, $n>1$, n нечетно	[23]
Тривиальная	3	$p>3$	[20]

Таблица 2

APN-функции, CCZ-эквивалентные степенным функциям

Функция $F(x)$	Условия	Источник
$x^{2^i+1}+(x^{2^i}+x+1)\mathrm{tr}(x^{2^i+1})$	$n\geqslant 4$, n четно, НОД(i,n)=1	[24]
$[x+\mathrm{tr}_{n/3}(x^{2(2^i+1)}+x^{4(2^i+1)}+\mathrm{tr}(x)\mathrm{tr}_{n/3}(x^{2^i+1}+x^{2^{2i}(2^i+1)})]^{2^i+1}$	n четно, $n=3k$, НОД(i,n)=1	[24]

В табл. 3 приведены построенные в 2006–2007 гг. с помощью подхода C. Carlet серии APN-функций, не CCZ-эквивалентных степенным функциям.

Другой подход к построению таких функций предложен в [13]. В [25] показано, как изменить одну координатную функцию APN-функции, чтобы получить новую APN-функцию. Такой подход к построению новых APN-функций оказался весьма эффективным.

Таблица 3

APN-функции, не CCZ-эквивалентные степенным функциям

Функция $F(x)$	Условия	Источник
$x^{2^s+1}+wx^{2^{ik}+2^{mk+s}}$	$n=3k$, НОД(k,3)=НОД(s,3k)=1, $k\geqslant 4$, $i=sk \pmod 3$, $m=3i$, ord$w$=$2^{2k}+2^k+1$	[26, 27]
$x^{2^s+1}+wx^{2^{ik}+2^{mk+s}}$	$n=4k$, НОД(k,2)=НОД(s,2k)=1, $k\geqslant 3$, $i=sk \pmod 4$, $m=4i$, ord$w$=$2^{3k}+2^{2k}+2^k+1$	[28]
$x^3+\mathrm{tr}(x^9)$	$n\geqslant 7$, $n>2p$ для наименьшего $p>1$, такого, что $p\neq 3$ и НОД$(p,n)=1$	[29]
$x^{2^{2i}+2^i}+bx^{q+1}+cx^{q(2^{2i}+2^i)}$	$n=2m$, $m\geqslant 3$, $q=2^m$, $c^{q+1}=1$, $c\notin\lambda^{(2^i+1)(q-1)}, \lambda\in \mathrm{GF}(2)$, НОД$(i,m)=1$, $cb^q+b\neq 0$	[30]
$x(x^{2^i}+x^q+cx^{2^iq})+x^{2^i}(c^qx^q+sx^{2^iq})+x^{(2^i+1)q}$	$n=2m$, $m\geqslant 3$, $q=2^m$, $s\notin \mathrm{GF}(q)$, $x^{2^i+1}+cx^{2^i}+c^qx+1$ неприводим над $\mathrm{GF}(2^n)$	[30]

7. Примеры других семейств APN-функций

В [31] введено в рассмотрение семейство функций вида

$$F(x)=ux^{\frac{p^n-1}{2}}+x^{p^n-2}$$

над $\mathrm{GF}(p^n)$ ($n\geqslant 3$ и нечетно), элемент $u\in\mathrm{GF}(p^n)$ удовлетворяет условию

$$\chi(u+1)=\chi(u-1)=\chi(u),$$

где χ — мультипликативный характер поля $\mathrm{GF}(p^n)$. Доказано, что такие функции являются APN-функциями. В [32] доказано, что это семейство не CCZ-эквивалентно ни одному из известных семейств APN-функций.

В [33] построено новое семейство APN-функций

$$F(x)=bx^{2^s+1}+b^{2^k}x^{2^{k+s}+2^k}+cx2^{k+s}+2^s+\sum_{i=0}^{k-1}r_ix^{2^{i+k}+2^i},$$

где k,s — нечетные взаимно простые числа; $b,c\in\mathrm{GF}(2^{2k})$; $r_i\in\mathrm{GF}(2^k)$.

В [34] показано, что функция $\mathrm{GF}(p^{3k})\to\mathrm{GF}(p^{3k})$ вида

$$x^{p^s+1}-ux^{p^k+p^{2k+s}}$$

при выполнении условий $n=3k$, НОД$(3,k)=1$, НОД$(s,n)=t$, u — примитивный элемент поля $\mathrm{GF}(p^{3k})$, является PN-функцией, если p и n/t — нечетные, и APN-функцией, если $p=2$.

8. Классификация APN-функций при малых n

В [35] приведены результаты классификации всех APN-функций $\mathrm{GF}(2^n)\to\mathrm{GF}(2^n)$ для $n=4,5$. Вычисления производились с помощью ЭВМ, в основе их лежала

лемма о том, что функция $F\colon \mathrm{GF}(2^n) \to \mathrm{GF}(2^n)$ является APN-функцией, если и только если для всех попарно различных t, u, v, w, принадлежащих любому аффинному подпространству размерности 2 пространства $\mathrm{GF}(2^n)$, выполнено неравенство $F(t) + F(u) + F(v) + F(w) \neq 0$.

Показано, что при $n = 4$ существует один класс CCZ-эквивалентности, содержащий два класса EA-эквивалентности, причем только один из них содержит степенную функцию. Для $n = 5$ существует 3 класса CCZ-эквивалентности и 7 классов EA-эквивалентности, из них два класса не содержат степенные функции, оставшиеся пять классов содержат APN-подстановки. Подстановки из каждого класса EA-эквивалентности аффинно эквивалентны степенной функции $F(x) = x^d$, каждому классу соответствует одно значение $d \in \{3, 5, 7, 11, 15\}$.

В заключение отметим, что в Интернете доступна предварительная версия обзора по APN-функциям, написанная C. Carlet в 2008 г. [36]. В обзоре, в частности, отмечается, что все известные APN-функции обладают не самыми лучшими характеристиками по отношению к другим, отличным от разностного, криптографическим методам. Следовательно, рекомендовать их для использования в качестве S-боксов пока рано. Требуются дальнейшие исследования.

ЛИТЕРАТУРА

1. *Biham E., Shamir A.* Provable Security Against Differential Cryptography // Crypto 1990, Lecture Notes in Computer Science. 1991. V. 537. P. 2–21.
2. *Nyberg K.* Perfect nonlinear S-boxes // Eurocrypt 1991, Lecture Notes in Computer Science. 1991. V. 547. P. 378–386.
3. *Nyberg K.* Differently uniform mappings for cryptography // Eurocrypt 1993, Lecture Notes in Computer Science. 1994. V. 765. P. 55–64.
4. *Nyberg K., Knudsen L. R.* Provable Security Against Differential Cryptography // Crypto 1992, Lecture Notes in Computer Science. V. 740. P. 566–574.
5. *Bierbrauer J.* New semifields, PN and APN functions // http://www.math.mtu.edu-/jbierbra/HOMEZEUGS/PNpub06Dez08.pdf, 2008.
6. *Voloch J. F.* Symmetric Cryptography and Algebraic Curves // Proceedings of Algebraic Geometry and its Applications. 2007. P. 135–141.
7. *Berger T., Canteaut A., Charpin P., Laigle-Chapuy Y.* On Almost Perfect Nonlinear Functions Over F_{2^n} // IEEE Transactions on Information Theory. 2006. V. 52. No. 9. P. 4160–4170.
8. *Beth T., Ding C.* On almost perfect nonlinear permutations // Eurocrypt 1993, Lecture Notes in Computer Science. 1994. V. 765. P. 65–76.
9. *Charpin P., Kyureghyan G.* On a class of permutation polynomials over F_{2^n} // SETA 2008, Lecture Notes in Computer Science. 2008. V. 5203. P. 368–376.
10. *Chabaud F., Vaudenay S.* Links between differential and linear cryptanalysis // Eurocrypt 1994, Lecture Notes in Computer Science. 1994. V. 950. P. 356–365.
11. *Сидельников В. М.* О взаимной корреляции последовательностей // Проблемы кибернетики. 1971. Т. 24. С. 15–42.
12. *Carlet C., Charpin P., Zinoviev V.* Codes, bent functions and permutations suitable for DES-like cryptosystems // Designs, Codes and Cryptography. 1998. V. 15(2). P. 125–156.
13. *Edel Y., Kyureghyan G., Pott A.* A new APN function which is not equivalent to a power mapping // IEEE Transactions on Information Theory. 2006. V. 52. No. 2. P. 744–747.
14. *Budaghyan L., Carlet C.* On CCZ-equivalence and its use in secondary constructions of bent functions // Cryptology ePrint Archive 2009/042.

15. *Gold R.* Maximal recursive sequence with 3-valued recursive cross-correlation functions // IEEE Transactions on Information Theory. 1968. V. 14. No. 1. P. 154–156.

16. *Kasami T.* The weight enumerators for several classes of subcodes of the second order binary Reed-Muller codes // Information and Control. 1971. V. 18. P. 369–394.

17. *Dobbertin H.* Almost perfect nonlinear power functions over GF(2^n): The Welch case // IEEE Transactions on Information Theory. 1999. V. 45. No. 4. P. 1271–1275.

18. *Dobbertin H.* Almost perfect nonlinear power functions over GF(2^n): The Niho case // International J. of Computer and Information Sciences. 1999. V. 151. P. 57–72.

19. *Dobbertin H.* Almost perfect nonlinear power functions over GF(2^n): A new case for n divisible 5 // Proc. of Finite Fields and Applications Fq5. Springer Verlag, 2001. P. 113–121.

20. *Helleseth T., Sandberg D.* Some power mappings with low differential uniformity // Applicable Algebra in Engineering, Communication and Computing. 1997. V. 8. P. 363–370.

21. *Dobbertin H., Mills D., Muller E., Pott A., Willems W.* APN functions in odd characteristic // Discrete Mathematics. 2003. V. 267. P. 95–112.

22. *Felke P.* Computing the uniformity of power mappings: a systematic approach with the multivariate method over finite fields of odd characteristic // Ph. D. dissertation, University of Bochum, Germany. 2005.

23. *Helleseth T., Rong C., Sandberg D.* New families of almost perfect nonlinear power mappings // IEEE Transactions on Information Theory. 1999. V. 45. No. 2. P. 475–485.

24. *Budaghyan L., Carlet C., Pott A.* New Classes of Almost Bent and Almost Perfect Nonlinear Polynomials // Proceedings of the Workshop on Coding and Cryptography. 2005. P. 306–315.

25. *Edel Y., Pott A.* A new almost perfect nonlinear function which is not quadratic // Cryptology ePrint Archive 2008/313.

26. *Budaghyan L., Carlet C., Leander G.* A class of quadratic APN binomials inequivalent to power functions // Cryptology ePrint Archive 2006/445.

27. *Budaghyan L., Carlet C., Felke P., Leander G.* An infinite class of quadratic APN functions which are not equivalent to power mappings // Proceedings of the IEEE International Symposium on Information Theory. 2006.

28. *Budaghyan L., Carlet C., , Leander G.* Another class of quadratic APN binomials over F_2^n: the case n divisible by 4. // Cryptology ePrint Archive 2006/428.

29. *Budaghyan L., Carlet C., Leander G.* Constructing new APN functions from known ones // Cryptology ePrint Archive 2007/063.

30. *Budaghyan L., Carlet C.* Classes of Quadratic APN Trinomials and Hexanomials and Related Structures // Cryptology ePrint Archive 2007/098.

31. *Ness G., Helleseth T.* A new family of ternary almost perfect nonlinear mappings // IEEE Transactions on Information Theory. 2007. V. 53. No. 7. P. 2581–2586.

32. *Zeng X., Hu L., Yang Y., Jiang W.* On the inequivalence of Ness-Helleseth APN functions // Cryptology ePrint Archive 2007/379.

33. *Bracken C., Byrne E., Markin N., McGuire G.* Quadratic Almost Perfect Nonlinear Functions with Many Terms // Cryptology ePrint Archive 2007/115.

34. *Zha Z., Kyureghyan G., Wang X.* A New Family of Perfect Nonlinear Binomials // Cryptology ePrint Archive 2008/196.

35. *Brinkmann M., Leander G.* On the Classification of APN Functions up to Dimension Five // Proceedings of the Workshop on Coding and Cryptography. 2007. P. 39–48.

36. *Carlet C.* Vectorial Boolean Functions for Cryptography // www-roc.inria.fr/secret/Claude.Carlet/chap-vectorial-fcts.pdf. 2008.

2009 Теоретические основы прикладной дискретной математики №3(5)

DOI 10.17223/20710410/5/3

УДК 519.7

О СЛОЖНОСТИ МЕТОДА ФОРМАЛЬНОГО КОДИРОВАНИЯ ПРИ АНАЛИЗЕ ГЕНЕРАТОРА С ПОЛНОЦИКЛОВОЙ ФУНКЦИЕЙ ПЕРЕХОДОВ[1]

В. М. Фомичёв

Институт проблем информатики РАН, г.Москва, Россия

E-mail: fomichev@nm.ru

Исследованы генераторы гаммы (автономные автоматы), множество состояний которых есть пространство двоичных n-мерных векторов, и функция переходов реализует полноцикловую подстановку множества состояний. Оценивается сложность T_n решения системы уравнений гаммообразования (без ограничения на число уравнений) относительно неизвестного начального состояния методом формального кодирования. Оценка получена с помощью определения линейной сложности и порядка множества мономов для последовательности выходных функций генератора. Показано, что $TL(2^{n-1}) < T_n < TL(2^n)$, где $TL(m)$ — сложность решения над GF(2) системы из m линейных уравнений от m неизвестных. Данный класс генераторов порождает, в частности, нормальные рекуррентные последовательности над полем GF(2) (последовательности де Брёйна).

Ключевые слова: *моном, аннулирующий полином, линейная оболочка.*

1. Характеристики, определяющие сложность решения систем уравнений методом формального кодирования

Метод формального кодирования [1] применим для решения любой системы уравнений над конечным полем [2]. Вместе с тем трудоемкость метода меньше трудоемкости метода полного перебора только для частных классов систем уравнений. Определим характеристики системы уравнений, важные для оценки сложности ее решения методом формального кодирования.

Обозначим: M_n — множество всех мономов, зависящих от переменных $x_1, \dots, x_n$; $M_{n,d}$ — его подмножество всех мономов степени d, $0 \leqslant d \leqslant n$. Отсюда $M_n = \bigcup\limits_{d=0}^{n} M_{n,d}$, где $M_{n,0} = \{1\}$ и при $d \geqslant 1$

$$M_{n,d} = \{x_{j_1} \dots x_{j_d} : \{j_1, \dots, j_d\} \subseteq \{1, \dots, n\}, j_1 < \dots < j_d\}.$$

Определим на M_n частичный порядок: $x_{j_1} \dots x_{j_d} \leqslant x_{s_1} \dots x_{s_\ell} \Leftrightarrow \{j_1, \dots, j_d\} \subseteq \{s_1, \dots, s_\ell\}$. Заметим, что M_n — решётка, изоморфная решётке V_n булевых векторов длины n, где вектору $\delta = (\delta_1, \dots, \delta_n)$ из V_n биективно соответствует моном $x^\delta = x_1^{\delta_1} \dots x_n^{\delta_n}$ из M_n, при этом для $j = 1, \dots, n$

$$x_j^{\delta_j} = \begin{cases} x_j, & \delta_j = 1, \\ 1, & \delta_j = 0. \end{cases}$$

Единицей и нулём решётки V_n являются соответственно векторы $e_n = (1, \dots, 1)$ и $u_n = (0, \dots, 0)$, единицей и нулём решётки M_n — мономы $x^{e_n} = x_1 \dots x_n$ и $x^{u_n} = 1$.

[1] Результаты работы докладывались на Международной конференции с элементами научной школы для молодёжи, г. Омск, 7–12 сентября 2009 г.

Рассмотрим систему m уравнений от n неизвестных, заданную полиномами над GF(2):

$$\left\{\begin{array}{rcl} f_1(x_1\,,\,\ldots\,,\,x_n) & = & a_1, \\ f_2(x_1\,,\,\ldots\,,\,x_n) & = & a_2, \\ \ldots\ldots\ldots\ldots & & \\ f_m(x_1\,,\,\ldots\,,\,x_n) & = & a_m. \end{array}\right. \tag{1}$$

Левая часть системы (1) (система булевых полиномов) определяет отображение $F_{m,n}(x_1\,,\,\ldots\,,\,x_n)$: $V_n \to V_m$, правая часть — вектор $a = (a_1\,,\,\ldots\,,\,a_m) \in V_m$. Кратко систему уравнений (1) запишем как $F_{m,n} = a$.

Обозначим: $M(f(x_1\,,\,\ldots\,,\,x_n))$ (кратко $M(f)$) — множество мономов ненулевой степени полинома $f(x_1\,,\,\ldots\,,\,x_n)$; $M(F_{m,n}) = \bigcup\limits_{i=1}^{m} M(f_i(x_1\,,\,\ldots\,,\,x_n))$ — множество мономов ненулевой степени системы координатных полиномов отображения $F_{m,n}$ (системы уравнений (1)). Для полинома $f(x_1\,,\,\ldots\,,\,x_n)$ и для системы полиномов $F_{m,n}$ над GF(2) величины $|M(f)|$ и $|M(F_{m,n})|$ называются *весом полинома* $f(x_1\,,\,\ldots\,,\,x_n)$ и *весом системы полиномов* $F_{m,n}$ и обозначаются $wp(f)$ и $wp(F_{m,n})$ соответственно.

Для решения системы уравнений (1) методом формального кодирования выполняется замена переменных. Каждый ненулевой моном из $M(F_{m,n})$ заменяется новой переменной: $x_{i_1} \ldots x_{i_s} = z_j$, после чего система уравнений $F_{m,n} = a$ преобразуется в линейную систему $L_{m,\nu} = a$ от переменных $z_1\,,\,\ldots\,,\,z_\nu$, где $\nu = wp(F_{m,n})$ и $L_{m,\nu}$ — система m булевых линейных полиномов от ν переменных.

Систему линейных уравнений $L_{m,\nu} = a$ можно решать известными методами. При решении совместной системы уравнений достаточно рассматривать лишь максимальную подсистему из μ линейно независимых уравнений, где $\mu = \dim \langle F_{m,n} \rangle$ — размерность линейной оболочки системы полиномов $F_{m,n}$ или, что равносильно, число линейно независимых строк матрицы $L_{m,\nu}$. Сложность решения совместной системы уравнений полиномиально зависит от μ и ν. Например, метод Гаусса имеет сложность порядка $\max\{\mu,\nu\} \cdot (\min\{\mu,\nu\})^2$ операций поля GF(2), в частности порядка ν^3, если $\mu = \nu$. Сложность решения линейной системы другими методами также определяется величинами ν и μ.

2. Постановка задачи

Рассмотрим генератор двоичной гаммы, моделируемый автономным автоматом $A = (V_n, V_1, h, f)$, где V_n — множество состояний; V_1 — выходной алфавит; f — функция выходов; h — функция переходов, реализующая полноцикловую подстановку множества V_n.

Уравнения гаммообразования автомата A описываются системой уравнений (1), где $a \in V_m$ и $f_i(x_1\,,\,\ldots\,,\,x_n)$ есть i-я выходная функция автомата A, определенная равенством: $f_i(x_1\,,\,\ldots\,,\,x_n) = f(h^i(x_1\,,\,\ldots\,,\,x_n))$, $i = 1\,,\,\ldots\,,\,m$. Отсюда гамма и последовательность $\{f_i(x_1\,,\,\ldots\,,\,x_n)\}$ выходных функций автомата являются чисто периодическими последовательностями; длины их периодов совпадают и делят 2^n. Пусть длины их периодов равны 2^t, где $1 < t \leqslant n$.

Требуется оценить сложность определения начального состояния генератора методом формального кодирования по известному периоду гаммы, где под периодом чисто периодической последовательности с длиной периода ℓ понимается любой ее отрезок длины ℓ. Следовательно, требуется определить (оценить) $\dim \langle F_{2\tau,n} \rangle$ и $|M(F_{2\tau,n})|$, где $\tau = 2^{t-1}$.

3. Размерность линейной оболочки системы выходных функций

Пусть r — натуральное число, $W = \{w_i\}$ — чисто периодическая последовательность над V_r с длиной периода $2\tau = 2^t$, где $\tau > 1$, $m_W(\lambda)$ — минимальный многочлен (над GF(2)) последовательности W. Длину периода периодической последовательности W обозначим $\operatorname{per} W$.

Последовательность W назовём *компенсированной*, если $w_1 \oplus \cdots \oplus w_{2\tau} = u_r$, где u_r — нуль пространства V_r.

Теорема 1.

а) $m_W(\lambda) = (\lambda \oplus 1)^s$, где $\tau + 1 \leqslant s \leqslant 2\tau - \varepsilon(W)$ и

$$\varepsilon(W) = \begin{cases} 0, & \text{если } W \text{ не компенсированная,} \\ 1, & \text{если } W \text{ компенсированная;} \end{cases}$$

при этом $s = \tau + 1$, если $w_i \oplus w_{i+\tau} = \alpha$ при некотором $\alpha \neq u_r$ и при всех $i = 1, \ldots, \tau$.

б) Если многочлен $\psi(\lambda) = p_0 \oplus p_1\lambda \oplus \cdots \oplus p_k\lambda^k$ аннулирует последовательность W и $k < 2\tau$, то вес $wp(\psi(\lambda))$ чётен и выполнены равенства: $p_j = p_{j+\tau}$, $j = 0, 1, \ldots, k - \tau$; $p_j = 0$, $j = k - \tau + 1, \ldots, \tau - 1$.

Доказательство.

а) По условию длина периода последовательности W равна 2τ; следовательно, W аннулируется многочленом $\lambda^{2\tau} \oplus 1$, каноническое разложение которого на неприводимые множители есть $(\lambda \oplus 1)^{2\tau}$. Так как $m_W(\lambda)$ делит $(\lambda \oplus 1)^{2\tau}$, то $m_W(\lambda) = (\lambda \oplus 1)^s$, где $s \leqslant 2\tau$. Вместе с тем $s > \tau$, так как иначе последовательность W аннулируется многочленом $(\lambda \oplus 1)^\tau$, где $(\lambda \oplus 1)^\tau = \lambda^\tau \oplus 1$. Последнее означало бы, что длина периода последовательности W не больше τ, что противоречит условию.

Для компенсированной последовательности W верхнюю оценку линейной сложности можно уточнить: W аннулируется многочленом $1 \oplus \lambda \oplus \cdots \oplus \lambda^{2\tau-1}$, каноническое разложение которого на неприводимые множители есть $(\lambda \oplus 1)^{2\tau-1}$. Следовательно, в этом случае $m_W(\lambda) = (\lambda \oplus 1)^s$, где $s \leqslant 2\tau - 1$.

При условии $w_i \oplus w_{i+\tau} = \alpha$ нижняя оценка для s достигается, так как при всех $i = 1, \ldots, 2\tau$ выполняется

$$w_{i+\tau+1} \oplus w_{i+\tau} \oplus w_{i+1} \oplus w_i = (w_{i+\tau+1} \oplus w_{i+1}) \oplus (w_{i+\tau} \oplus w_i) = \alpha \oplus \alpha = u_r.$$

Отсюда многочлен $\lambda^{\tau+1} \oplus \lambda^\tau \oplus \lambda \oplus 1$, разложение которого на неприводимые множители есть $(\lambda \oplus 1)^{\tau+1}$, аннулирует последовательность W и является минимальным многочленом для W в соответствии с доказанной нижней оценкой для s.

б) Из утверждения *а* теоремы следует: $m_W(\lambda) = (\lambda^\tau \oplus 1)(\lambda \oplus 1)^{s-\tau}$, где $\tau + 1 \leqslant s < 2\tau$. Любой многочлен $\psi(\lambda)$, аннулирующий последовательность W, кратен многочлену $m_W(\lambda)$, то есть $\psi(\lambda) = m_W(\lambda)\varphi(\lambda)$ при некотором ненулевом многочлене $\varphi(\lambda)$. Поэтому если $\deg\psi(\lambda) = k \leqslant 2\tau - 1$, то

$$\psi(\lambda) = (\lambda^\tau \oplus 1)q(\lambda), \tag{2}$$

где $\deg q(\lambda) = k - \tau$ и $q(\lambda) = (\lambda \oplus 1)^{s-\tau}\varphi(\lambda)$. Если $q(\lambda) = q_0 \oplus q_1\lambda \oplus \cdots \oplus q_{k-\tau}\lambda^{k-\tau}$, то из (2) следует, что $p_j = p_{j+\tau} = q_j$, $j = 0, 1, \ldots, k - \tau$; $p_j = 0$, $j = k - \tau + 1, \ldots, \tau - 1$.

Из (2) следует также, что $wp(\psi(\lambda)) = 2wp(q(\lambda))$, поэтому $wp(\psi(\lambda))$ чётен. ■

Следствие. Для последовательности выходных функций автомата A выполнено

$$2\tau \geqslant \dim \langle F_{2\tau,n} \rangle \geqslant \tau + 1.$$

Доказательство. Последовательность $\{f_i(x_1,\dots,x_n)\}$ выходных функций автомата чисто периодическая, поэтому ее линейная оболочка совпадает с $\langle F_{2\tau,n}\rangle$ — с линейной оболочкой ее периода. По теореме 7.1 [3] $\dim\langle F_{2\tau,n}\rangle = \deg m_f(\lambda)$, где $m_f(\lambda)$ — минимальный многочлен последовательности $\{f_i(x_1,\dots,x_n)\}$. В соответствии с утверждением a теоремы $2\tau \geqslant \deg m_f(\lambda) \geqslant \tau+1$. Следовательно, $2\tau \geqslant \dim\langle F_{2\tau,n}\rangle \geqslant \tau+1$. ■

4. Свойства системы полиномов выходных функций автомата

Для автомата A обозначим чисто периодические последовательности: $W(h) = \{h^i(u_n)\}$ — последовательность состояний при начальном состоянии u_n; $f(W(h)) = \{f(h_i(u_n))\}$ — соответствующая ей выходная последовательность; $H = \{h^i\}$ — последовательность степеней подстановки h; $f(H) = \{f(h^i)\}$ — соответствующая ей последовательность выходных функций, $i = 0,1,2,\dots$ Заметим, что период последовательности H совпадает с циклической группой подстановок $\langle h\rangle$ порядка 2^n.

Отметим свойства этих последовательностей:

1) $\operatorname{per} W(h) = \operatorname{per} H = 2^n$;

2) $\operatorname{per} f(W(h)) = \operatorname{per} f(H) = 2\tau$, где $1 < \tau \leqslant 2^{n-1}$;

3) $m_{W(h)}(\lambda) = m_H(\lambda)$;

4) $m_{f(W(h))}(\lambda) = m_{f(H)}(\lambda)$;

5) $M(f(H)) = M(F_{2\tau,n})$.

Обозначим: $M_d(f(H)) = M_{n,d} \cap M(f(H))$ — множество всех мономов степени d, содержащихся в совокупности полиномов выходных функций из последовательности $f(H)$, $0 \leqslant d \leqslant n$. Отсюда

$$M(f(H)) = \bigcup_{d=1}^{n} M_d(f(H)).$$

На периоде последовательности $W(h)$ определим порядковый номер вектора $\gamma \in V_n$ (обозначим его $n(\gamma)$): $n(u_n) = 0$, $n(\gamma)$ — наименьшее натуральное число t, такое, что $h^t(u_n) = \gamma$.

Для монома x^δ степени $d > 0$, где $\delta \in V_n$, определим многочлен $\psi_\delta(\lambda)$ над GF(2), зависящий от функций h и f автомата A:

$$\psi_\delta(\lambda) = \bigoplus_{\gamma \leqslant \delta} \lambda^{\nu(\gamma)}, \tag{3}$$

где $\nu(\gamma)$ — наименьший неотрицательный вычет числа $n(\gamma)$ по $\bmod(2\tau)$; символом $\leqslant$ обозначено стандартное отношение частичного порядка на решётке V_n. Многочлен $\psi_\delta(\lambda)$ назовем A-многочленом монома x^δ.

Отметим некоторые свойства A-многочленов мономов:

1) $\deg\psi_\delta(\lambda) < 2\tau$;

2) $wp(\psi_\delta(\lambda)) \leqslant 2^d$ и является чётным;

3) $wp(\psi_\delta(\lambda)) = 2^d \Leftrightarrow$ в последовательности $W(h)$ номера любых двух не превосходящих δ векторов не сравнимы по $\bmod(2\tau)$;

4) $wp(\psi_\delta(\lambda)) = 0 \Leftrightarrow$ множество $\{n(\gamma) : \gamma \leqslant \delta\}$ есть совокупность 2^{d-1} пар чисел, где числа в каждой паре сравнимы по модулю 2τ.

Обозначим: $g = h^\tau$. Заметим, что подстановка g состоит из τ циклов длины $2^n/\tau$.

Лемма 1.

а) Моном x^δ степени $d > 0$ не содержится в $M(f(H)) \Leftrightarrow$ A-многочлен монома x^δ либо нулевой, либо аннулирующий для последовательности $f(H)$.

б) Если $\psi_\delta(\lambda)$ аннулирует последовательность $f(H)$ и $\gamma \leqslant \delta$ для γ, $\delta \in V_n$, то и $g(\gamma) \leqslant \delta$.

Доказательство.

а) В полиноме Жегалкина булевой функции φ коэффициент $a_\delta(\varphi)$ при мономе x^δ равен булевой сумме 2^d элементов её табличного задания [3], формула (3.16):

$$a_\delta(\varphi) = \bigoplus_{\gamma \leqslant \delta} \varphi(\gamma).$$

Преобразуем это равенство, используя порядковые номера векторов:

$$a_\delta(\varphi) = \bigoplus_{\gamma \leqslant \delta} \varphi(h^{n(\gamma)}(u_n)).$$

Отсюда имеем, в частности, равенства для коэффициентов выходных функций f_i, $i = 0, 1, 2, \ldots$:

$$a_\delta(f_i) = \bigoplus_{\gamma \leqslant \delta} f(h^i(h^{n(\gamma)}(u_n)) = \bigoplus_{\gamma \leqslant \delta} f(h^{i+n(\gamma)}(u_n)).$$

Так как $\operatorname{per} f(W(h)) = 2\tau$, то $f(h^{i+n(\gamma)}) = f(h^{i+\nu(\gamma)})$, $i = 0, 1, 2, \ldots$, следовательно,

$$a_\delta(f_i) = \bigoplus_{\gamma \leqslant \delta} f(h^{i+\nu(\gamma)}(u_n)). \tag{4}$$

По определению $x^\delta \notin M(f(H)) \Leftrightarrow a_\delta(f_i) = 0$, $i = 0, 1, 2, \ldots$ Отсюда в соответствии с (3) следует, что $x^\delta \notin Mf(H)) \Leftrightarrow A$-многочлен монома x^δ либо является нулевым, либо аннулирующим последовательность $f(H)$ (а также и $f(W(h))$, так как $m_{f(W(h))}(\lambda) = m_{f(H)}(\lambda)$).

б) В соответствии с определением подстановки g номера $n(\gamma)$ и $n(g(\gamma))$ сравнимы по модулю τ и не сравнимы по модулю 2τ, поэтому $\nu(\gamma)$ и $\nu(g(\gamma))$ также сравнимы по модулю τ. Отсюда если многочлен $\psi_\delta(\lambda)$ аннулирует последовательность $f(H)$ (а также $f(W(h))$), где $\deg \psi_\delta(\lambda) < 2\tau$, то из утверждения *б* теоремы 1 и равенства (3) вытекает, что если $\lambda^{\nu(\gamma)}$ — моном многочлена $\psi_\delta(\lambda)$, то и $\lambda^{\nu(g(\gamma))}$ также моном многочлена $\psi_\delta(\lambda)$. По построению многочлена $\psi_\delta(\lambda)$ это равносильно тому, что если $\gamma \leqslant \delta$, то и $g(\gamma) \leqslant \delta$. ■

Для монома x^δ обозначим

$$U_n(x^\delta) = \{\xi \in M_n : x^\delta \leqslant \xi\};\ \overline{U}_n(x^\delta) = M_n \setminus U_n(x^\delta).$$

При любом δ из V_n множество $\overline{U}_n(x^\delta)$ не содержит монома x^{e_n} и множество $U_n(x^\delta)$ не пусто и образует в M_n подрешётку, изоморфную решётке $M_{n-\|\delta\|}$, где $\|\delta\|$ — вес двоичного вектора δ.

Обозначим также: $M_n(\gamma, \beta) = (U_n(x^\gamma) \cap \overline{U}_n(x^\beta)) \cup (U_n(x^\beta) \cap \overline{U}_n(x^\gamma))$, где $\gamma, \beta \in V_n$.

Теорема 2.

а) Множество $M(f(H))$ содержит $M(f(x_1, \ldots, x_n))$ и все мономы степени $d > 0$ из $\bigcup\limits_{\gamma \in V_n} M_n(\gamma, g(\gamma))$ с ненулевым A-многочленом.

б) Моном $x^{e_n} \in M(f_i(x_1, \ldots, x_n))$, $i = 1, 2, \ldots \Leftrightarrow x^{e_n} \in M(f(H)) \Leftrightarrow \|f\|$ нечетен.

Доказательство.

а) Последовательность H содержит тождественную подстановку, поэтому последовательность $f(H)$ содержит функцию $f(x_1, \ldots, x_n)$. Следовательно, $M(f(x_1, \ldots, x_n)) \subseteq M(f(H))$.

При любом $\gamma \in V_n$ для векторов γ и $g(\gamma)$ не выполнены одновременно отношения $\gamma \leqslant g(\gamma)$ и $g(\gamma) \leqslant \gamma$, так как $\gamma \neq g(\gamma)$ в соответствии с определением подстановки g. Пусть, например, не выполнено отношение $g(\gamma) \leqslant \gamma$, тогда из определения множества $M_n(\gamma, g(\gamma))$ следует, что множество $U_n(x^\gamma) \cap \overline{U}_n(x^{g(\gamma)})$ содержит моном x^γ и, следовательно, не пусто. Если $x^\delta \in U_n(x^\gamma) \cap \overline{U}_n(x^{g(\gamma)})$ при некоторых $\delta, \gamma \in V_n$, то $x^\delta \in U_n(x^\gamma)$ и $x^\delta \in \overline{U}_n(x^{g(\gamma)})$. Отсюда следует, что $\gamma \leqslant \delta$ в силу определения множества $U_n(x^\gamma)$ и отношение $g(\gamma) \leqslant \delta$ не выполнено в силу определения множества $\overline{U}_n(x^{g(\gamma)})$.

Если для монома x^δ степени $d > 0$ A-многочлен $\psi_\delta(\lambda)$ ненулевой и $x^\delta \notin M(f(H))$, то по утверждению *а* леммы 1 $\psi_\delta(\lambda)$ аннулирует последовательность $f(H)$ и по утверждению *б* леммы 1 если $\gamma \leqslant \delta$, то и $g(\gamma) \leqslant \delta$, что противоречит условию. Значит, если $\psi_\delta(\lambda)$ — ненулевой многочлен, то $x^\delta \in M(f(H))$.

Включение $x^\delta \in M(f(H))$ для монома x^δ из $U_n(x^{g(\gamma)}) \cap \overline{U}_n(x^\gamma)$ (если не выполнено отношение $\gamma \leqslant g(\gamma)$) доказывается аналогично. Заметим, что рассуждения верны для произвольного вектора γ.

б) В полиноме булевой функции коэффициент при мономе x^{e_n} равен $1 \Leftrightarrow$ вес функции нечетен (теорема 3.11 [3]). Осталось заметить, что веса всех выходных функций автомата равны $\|f\|$ в силу эквивалентности этих функций относительно группы подстановок $\langle h \rangle$. ■

Замечание. Если векторы γ и $g(\gamma)$ сравнимы, например $\gamma \leqslant g(\gamma)$, то $U_n(x^{g(\gamma)}) \cap \overline{U}_n(x^\gamma) = \varnothing$ и $M_n(\gamma,g(\gamma)) = U_n(x^\gamma) \cap \overline{U}_n(x^{g(\gamma)})$.

Следствие 1. Если $g(u_n) = \beta$, то $M(f(H))$ содержит все мономы из $\overline{U}_n(x^\beta)$ с ненулевым A-многочленом; если при этом $\|\beta\| = r$ и каждый моном из $\overline{U}_n(x^\beta)$ имеет ненулевой A-многочлен, то

$$|M_d(f(H))| \geqslant \begin{cases} C_n^d, & 1 \leqslant d < r, \\ C_n^d - C_{n-r}^{d-r}, & r \leqslant d < n. \end{cases} \tag{5}$$

Доказательство. Так как $u_n \leqslant \beta$, то $M_n(u_n, \beta) = U_n(x^{u_n}) \cap \overline{U}_n(x^\beta)$ в силу замечания к теореме 2. При этом $U_n(x^{u_n}) = M_n$, значит, $M_n(u_n, \beta) = \overline{U_n}(x^\beta)$, и по утверждению *а* теоремы 2 если моном x^δ из $\overline{U}_n(x^\beta)$ имеет ненулевой A-многочлен, то $x^\delta \in M(f(H))$.

По определению $\overline{U}_n(x^\beta) = M_n \backslash U_n(x^\beta)$. Поэтому если все мономы из $\overline{U}_n(x^\beta)$ имеют ненулевые A-многочлены, то по утверждению *а* теоремы 2 $|M_d(f(H))|$ оценивается снизу числом мономов степени d, которые не больше монома x^β степени r. К таким мономам относятся все мономы степени d, если $d < r$, и если $d \geqslant r$, — все мономы степени d за исключением тех, которые превосходят фиксированный моном x^β степени r. ■

Следствие 2. Если $g(e_n) = \alpha$, то $M(f(H))$ содержит все мономы из $U_n(x^\alpha) \backslash \{x^{e_n}\}$ с ненулевым A-многочленом; если при этом $\|\alpha\| = r$ и каждый моном из $U_n(x^\alpha) \backslash \{x^{e_n}\}$ имеет ненулевой A-многочлен, то

$$|M_d(f(H))| \geqslant \begin{cases} 0, & 1 \leqslant d < r, \\ C_{n-r}^{d-r}, & r \leqslant d < n. \end{cases} \tag{6}$$

Доказательство. Так как $\alpha \leqslant e_n$, то $M_n(\alpha, e_n) = U_n(x^\alpha) \cap \overline{U}_n(x^{e_n})$ в силу замечания к теореме 2. При этом $\overline{U}_n(x^{e_n}) = M_n \backslash \{x^{e_n}\}$, значит, $M_n(\alpha, e_n) = U_n(x^\alpha) \backslash \{x^{e_n}\}$, и по утверждению *а* теоремы 2 если моном x^δ из $U_n(x^\alpha) \backslash \{x^{e_n}\}$ имеет ненулевой A-многочлен, то $x^\delta \in M(f(H))$.

Если все мономы x^δ из $U_n(x^\alpha)\backslash\{x^{e_n}\}$ имеют ненулевые A-многочлены, то по утверждению a теоремы 2 $|M_d(f(H))|$ оценивается снизу числом мономов степени d, которые не меньше монома x^α степени r, где $d \geqslant r$. ■

Из следствий 1 и 2 теоремы 2 получаем оценку множества $M(f(H))$. Обозначим для $\alpha, \beta \in V_n$:

$$M_n(\beta/u, \alpha/e) = \overline{U}_n(x^\beta) \cup U_n(x^\alpha)\backslash\{x^{e_n}\}.$$

Заметим, $M_n(\beta/u, \alpha/e) = M_n\backslash\{x^{e_n}\}$ при $\alpha \leqslant \beta$.

Следствие 3. Если $g(u_n) = \beta$, $g(e_n) = \alpha$, то $M(f(H))$ содержит все мономы из $M_n(\beta/u, \alpha/e)$ с ненулевым A-многочленом.

Теорема 3. Если $\operatorname{per} f(H) = 2^n$, то:

а) $|M(f(H))| \geqslant 2^{n-1}$;

б) при случайном равновероятном выборе h из класса всех полноцикловых подстановок

$$E|M(f(H))| \geqslant 2^n - (1{,}5)^n + (1{,}25)^n,$$

где $E\zeta$ — математическое ожидание случайной величины ζ.

Доказательство.

а) Для любой полноцикловой подстановки h подстановка g не содержит единичных циклов, поэтому $\|g(u_n)\| \geqslant 1$. Пусть $g(u_n) = \beta = (\beta_1, \ldots, \beta_n)$ и для определённости $\beta_1 = 1$.

Если $\operatorname{per} f(H) = 2^n$, то любой моном степени $d > 0$ имеет ненулевой A-многочлен, и при указанном векторе β множество $\overline{U}_n(x^\beta)$ содержит все мономы, не зависящие от x_1. Следовательно, $|M_n(\alpha/e, \beta/u)| \geqslant |\overline{U}_n(x^\beta)| \geqslant 2^{n-1}$. Отсюда $|M(f(H))| \geqslant 2^{n-1}$ в соответствии со следствием 3 теоремы 2.

б) При случайном равновероятном выборе h из класса всех полноцикловых подстановок множества V_n пара векторов (α,β), где $\alpha = g(e_n)$, $\beta = g(u_n)$, есть равновероятная бесповторная выборка из множества $(V_n\backslash\{u_n, e_n\})$. Число таких выборок равно $(2^n - 2)(2^n - 3)$.

Пусть для векторов α, β решётки V_n выполнено: $\|\alpha\| = r$, $\|\beta\| = p$, $\|\inf(\alpha, \beta)\| = k$, тогда $\|\sup(\alpha, \beta)\| = r + p - k$. Число пар таких векторов из V_n, обозначаемое $N(r, p, k)$, равно

$$N(r, p, k) = C_n^r \cdot C_r^k \cdot C_{n-r}^{p-k}.$$

Для такой пары векторов α, β выполнено

$$|\overline{U}_n(x^\beta)| = 2^n - 2^{n-p},$$

$$\left|(U_n(x^\alpha) \setminus \{x^{e_n}\}) \setminus \overline{U}_n(x^\beta)\right| = \left|U_n(x^{\sup(\alpha,\beta)}) \setminus \{x^{e_n}\}\right| = 2^{n-r-p+k} - 1,$$

$$|M_n(\alpha/e, \beta/u)| = \left|\overline{U}_n(x^\beta)\right| + \left|(U_n(x^\alpha) \setminus \{x^{e_n}\}) \setminus \overline{U}_n(x^\beta)\right| = 2^n - 2^{n-p} + 2^{n-r-p+k} - 1.$$

Следовательно, при случайном равновероятном выборе h из класса всех полноцикловых подстановок множества V_n среднее значение $|M_n(\alpha/e, \beta/u)|$ определяется формулой (используем принцип включения-исключения с учетом, что α, β, e_n, u_n суть различные векторы из V_n)

$$E\,|M_n(\alpha/e, \beta/u)| = \frac{1}{(2^n - 2)(2^n - 3)}\left[\sum_{r=0}^{n}\sum_{k=0}^{r}\sum_{p-k=0}^{n-r} N(r, p, k)(2^n - 2^{n-p} + 2^{n-r-p+k} - 1) -\right.$$

$$-\sum_{p=0}^{n} N(0,p,0)(2^n-1) - \sum_{p=0}^{n} N(n,p,p)(2^n-2^{n-p}) - \sum_{r=0}^{n} N(r,0,0)(2^{n-r}-1) -$$

$$-\sum_{r=0}^{n} N(r,n,r)(2^n-1) + (N(0,0,0)+N(0,n,0)+N(n,0,0)+N(n,n,n))(2^n-1)\Big].$$

Подсчёт сумм с использованием формулы бинома Ньютона даёт следующий результат:

$$E\ |M_n(\alpha/e,\beta/u)| = \frac{8^n-6^n+5^n-4^{n+1}+6\cdot 2^n-3}{(2^n-2)(2^n-3)}.$$

Отсюда получаем требуемую оценку. ■

Пример. Пусть функция переходов h автомата A реализуется линейным конгруэнтным генератором: $h(x) = (x+1) \bmod 2^n$. Для подстановки h множества V_n выполнено:

1) $\operatorname{per} W_j(h) = \operatorname{per} W_j(H) = 2^j$, $j = 1, \ldots, n$;

2) пусть $\operatorname{per} f(H) = 2^n$, тогда $\tau = 2^{n-1}$, $h(x) \oplus h^\tau(x) = (0, \ldots, 0, 1)$ при любом $x \in V_n$.

Последовательность $f(H)$ имеет алгебраические характеристики:

1) $m_H(\lambda) = (\lambda \oplus 1)^{\tau+1}$ — по утверждению a теоремы 1;

2) $M(H)$ содержит множество всех мономов от $x_1, \ldots, x_{n-1}$ — по утверждению a теоремы 2;

3) $\deg W_n(H) = n-1$.

Выводы

1. Последовательность выходных функций генератора гаммы, построенного на основе полноцикловой подстановки множества состояний V_n, имеет высокую линейную сложность Λ, а именно $2^{n-1}+1 \leqslant \Lambda \leqslant 2^n$.

2. Для порядка множества мономов на периоде последовательности выходных функций генератора верны оценки: $2^{n-1} \leqslant |M(f(H))| \leqslant 2^n - 1$. При $n \to \infty$ и при случайном равновероятном выборе функции переходов h из класса всех полноцикловых подстановок множества V_n математическое ожидание величины $|M(f(H))|\,/(2^n-1)$ стремится к 1.

3. Сложность T_n определения начального состояния генератора методом формального кодирования по известному периоду гаммы удовлетворяет оценкам

$$TL(2^{n-1}) < T_n < TL(2^n),$$

где $TL(m)$ — сложность решения над GF(2) системы из m линейных уравнений от m неизвестных.

ЛИТЕРАТУРА

1. *Schaumuller-Bichl.* Cryptanalysis of the Data Encryption Standard by a method of formal coding // Cryptography, Proc. Burg Feuerstein 1982. LNCS. 1983. V. 149. P. 235–255.

2. *Courtois N., Klimov A., Patarin J., Shamir A.* Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations // LNCS. 2000. V. 1807. P. 392–407.

3. *Фомичёв В. М.* Дискретная математика и криптология. М.: ДИАЛОГ–МИФИ, 2003. 400 с.

2009 Теоретические основы прикладной дискретной математики №3(5)

DOI 10.17223/20710410/5/4

УДК 65.012.810(075.8)

ПОЧТИ ВСЕ ЛАТИНСКИЕ КВАДРАТЫ ИМЕЮТ ТРИВИАЛЬНУЮ ГРУППУ АВТОСТРОФИЙ[1]

А. В. Черемушкин

Институт криптографии, связи и информатики, г. Москва, Россия

E-mail: avc238@mail.ru

Доказано, что при $n \to \infty$ почти все латинские квадраты имеют тривиальную группу автострофий. Как следствие выводится асимптотическая оценка числа главных классов эквивалентности латинских квадратов порядка n.

Ключевые слова: *латинские квадраты, квазигруппы, ортогональные массивы, автострофии.*

1. Основные определения

Пусть $I_n = \{1, 2, \ldots, n\}$. Латинским квадратом порядка n называется $n \times n$-матрица $L = (l_{ij})$, строки и столбцы которой являются подстановками множества I_n. Каждый латинский квадрат L порядка n можно рассматривать как табличное задание квазигрупповой операции «$\circ$» на множестве I_n, задаваемой равенством $i \circ j = l_{ij}$, $i, j \in I_n$, либо как ортогональный массив OA(n,3,1) вида

$$L = \{(i, j, k) \mid i, j \in I_n, k = i \circ j\}. \tag{1}$$

Изотопия — это тройка подстановок $\alpha = (r, c, s) \in S_n^3$. Здесь r переставляет строки, c — столбцы, а s осуществляет замену элементов. Действие группы изотопий на множестве латинских квадратов определяется как $\alpha : L \to L^\alpha$, где

$$L^\alpha = \{(i^r, j^c, k^s) \mid (i, j, k) \in L\}.$$

Если $L^\alpha = L$, то $\alpha \in S_n^3$ называется *автотопией*. *Изострофия* — это преобразование (α, σ), где $\alpha \in S_n^3$ — изотопия, а $\sigma \in S_3$ действует на множестве троек ортогонального массива (1) путем перестановки координат в каждой тройке $(i, j, k) \in I_n$, причем

$$L^{(\alpha,\sigma)} = \{(i^r, j^c, k^s)^\sigma \ : \ (i, j, k) \in L\}.$$

Классы эквивалентности относительно группы преобразований $\langle S_n^3, S_3 \rangle = S_n^3 \wr S_3$ называются главными классами. Наконец, (α, σ) — *автострофия*, если $L^{(\alpha,\sigma)} = L$.

2. Вспомогательная лемма

Воспользуемся следующим результатом из работы [1].

Лемма 1 [1]. Пусть L — латинский квадрат с нетривиальной группой автострофий. Тогда найдется некоторый изострофный ему латинский квадрат L', имеющий изострофию (α, σ) с одной из следующих структур:

(i) $\alpha = (r, c, s)$ для некоторого простого числа p, где r, c имеют порядок p и одинаковое число m неподвижных точек, $1 \leqslant m \leqslant n/2$;

[1]Работа выполнена при поддержке гранта Президента РФ НШ № 4.2008.10.

(ii) $\alpha = (r, c, s)$ для некоторого простого числа p, делящего n, где r, c имеют порядок p и не имеют неподвижных точек, а s имеет порядок 1 или p; более того, если $p = 2$ и $n \equiv 2 \pmod 4$, то s имеет по крайней мере две неподвижные точки;

(iii) $(\alpha, \sigma) = (1, 1, s, (RC))$, где s имеет порядок 1 или 2 и по крайней мере одну неподвижную точку;

(iv) $\sigma = (RCS)$.

Здесь подстановка σ записывается в цикловой записи, с использованием больших букв R, C и S, соответствующих номерам строк, столбцов и элементам латинского квадрата. Поэтому запись $\sigma = (RC)$ означает транспонирование квадрата относительно главной диагонали, а $\sigma = (RCS)$ — циклическую замену элементов (i, j, k) ортогонального массива (1) на (k, i, j), где $i, j, k \in I_n$.

3. Основная теорема

Основным результатом данной работы является следующая теорема, анонсированная в [2] и уточняющая теорему 4 из работы автора [3].

Теорема 1. При $n \to \infty$ почти все латинские квадраты порядка n имеют тривиальную группу автострофий.

Доказательство. Доля латинских квадратов порядка n с нетривиальной группой автотопий составляет $T(n)/L_n$, где $T(n)$ — число латинских квадратов порядка n с нетривиальной группой автотопий, а L_n — число латинских квадратов порядка n. Для оценки этой величины воспользуемся оценкой из [4] (с. 141): при $n \to \infty$ для числа латинских квадратов L_n справедлива оценка

$$L_n > \frac{n^{n^2}}{e^{n^2}}. \tag{2}$$

В силу леммы 1 число $T(n)$ можно оценить сверху выражением

$$T(n) < (R_1(n) + R_2(n) + R_3(n) + R_4(n)) \cdot (n!)^3 3!,$$

где $R_i(n)$, $1 \leqslant i \leqslant 4$, — число латинских квадратов, имеющих автострофию, вид которой указан соответственно в пп. (i)–(iv) леммы. Найдем верхние оценки каждой из величин $R_i(n)$, $1 \leqslant i \leqslant 4$.

В случае (i) для некоторого простого числа p имеем $\alpha = (r, c, s)$, где подстановки r, c имеют порядок p и одинаковое число m неподвижных точек, $1 \leqslant m \leqslant n/2$. Пусть подстановки r и c имеют i циклов длины p, $n = ip + m$, $n/(2p) \leqslant i \leqslant n/p$. Так как при действии подстановки r строки латинского квадрата, соответствующие одному циклу этой подстановки, переходят одна в другую, то достаточно задать только i строк, а остальные $ip - i$ строк определятся однозначно. Поскольку аналогичное свойство выполняется и для столбцов, то для задания оставшихся неопределенными элементов латинского квадрата, лежащих на ip столбцах, соответствующих циклам подстановки c, достаточно задать еще по $n - ip$ элементов в i столбцах. В результате осталась неопределенной только часть латинского квадрата, содержащая $(n - ip) \times$ $\times (n - ip)$ элементов.

Поэтому число латинских квадратов, имеющих автострофию указанного вида, можно оценить выражением

$$R_1(n) \leqslant \sum_{p|n, p \geqslant 2} \sum_{\frac{n}{2p} \leqslant i \leqslant \frac{n}{p}} \left(\binom{n}{ip} \frac{(ip)!}{p^i i!} \right)^2 n! n!^i ((n - ip)!)^{n - ip + i}) <$$

$$< n^2 n!^3 \max\{n!^i((n-ip)!)^{n-i(p-1)}\}.$$

Максимальное значение величины $n!^i((n-ip)!)^{n-i(p-1)}$ на интервале $n/2 \leqslant ip < n$ достигается при $i = n/4,\ p = 2$. Применяя формулу Стирлинга

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\left(1+\frac{\theta}{12n}\right)}, \quad 0 < \theta < 1,$$

имеем

$$R_1(n) \leqslant n^2 n!^3 n!^{n/4}((n/2)!)^{3n/4} \leqslant \exp\left\{\frac{5}{8}n^2 \ln n + O(n^2)\right\}.$$

В случае (ii) для некоторого простого числа p, делящего n, имеем $\alpha = (r, c, s)$, где r, c имеют порядок p и не имеют неподвижных точек, а s имеет порядок 1 или p. Поэтому величину $R_2(n)$ можно оценить с помощью аналогичных рассуждений следующим образом:

$$R_2(n) \leqslant \sum_{p|n, p\geqslant 2}\left(\frac{n!}{p^{n/p}(n/p)!}\right)^2 n!(n!)^{n/p} < n(n!)^{n/2+3} \leqslant \exp\left\{\frac{1}{2}n^2 \ln n + O(n^2)\right\}.$$

В случае (iii) в группе автострофий с точностью до сопряжения есть подстановка вида $(\alpha, \sigma) = (1, 1, s, (RC))$, причем подстановка $\sigma = (RC)$ задает симметрию латинского квадрата относительно главной диагонали с точностью до замены элементов s. Поэтому для определения всех элементов латинского квадрата достаточно задать элементы, расположенные на главной диагонали и верхнем треугольнике. Отсюда число латинских квадратов с такими автострофиями можно оценить величиной

$$R_3(n) \leqslant 3(n!)^3 n^{n(n+1)/2} \leqslant \exp\left\{\frac{1}{2}n^2 \ln n + O(n^2)\right\}.$$

В случае (iv) в группе автострофий с точностью до сопряжения есть тройной цикл вида $\sigma = (RCS)$. Поэтому тройки (i, j, k), (k, i, j) и (j, k, i) в ортогональном массиве (1) будут либо все различны, либо все одинаковы, причем они будут совпадать в том и только в том случае, когда $i = j = k$. Отсюда число латинских квадратов с такими автострофиями можно оценить величиной

$$R_4(n) \leqslant 2(n!)^3 n^{(n^2-n)/3+n} \leqslant \exp\left\{\frac{1}{3}n^2 \ln n + O(n^2)\right\}.$$

Итак, число латинских квадратов с нетривиальной группой автострофий оценивается выражением

$$T(n) = \sum_{i=1}^{4} R_i(n) \leqslant \exp\left\{\frac{5}{8}n^2 \ln n + O(n^2)\right\} \leqslant \frac{n^{n^2}}{e^{n^2}} \exp\left\{-\frac{3}{8}n^2 \ln n(1 - O(\ln^{-1} n)\right\}.$$

Окончательно, с учетом неравенства (2), получаем

$$\frac{T(n)}{L_n} \leqslant \exp\left\{-\frac{3}{8}n^2 \ln n\ (1 - O(\ln^{-1} n))\right\}.$$

Теорема доказана. ■

В частности, из данной теоремы вытекает теорема 4 из [3].

Следствие 1. При $n \to \infty$ почти все бинарные квазигруппы порядка n имеют тривиальную группу автотопий.

4. Асимптотическая оценка

Приведем теперь асимптотическую оценку числа главных классов эквивалентности латинских квадратов порядка n.

Следствие 2. При $n \to \infty$ число главных классов эквивалентности латинских квадратов порядка n асимптотически равно

$$\frac{L_n}{6n!^3}(1 + o(1)),$$

где L_n — число латинских квадратов порядка n.

Пусть $N(n)$ — число главных классов эквивалентности порядка n, а M_n — множество представителей главных классов эквивалентности латинских квадратов порядка n с нетривиальной группой автострофий. Тогда доказательство вытекает из следующего равенства работы [1]:

$$N(n) = \frac{L_n}{6n!^3} + \sum_{L \in M_n} \frac{|Par(L)| - 1}{|Par(L)|},$$

где $Par(L)$ — группа автострофий латинского квадрата L.

ЛИТЕРАТУРА

1. *McKay B. D., Meynet A., Myrvold W.* Small latin squares, quasigroups and loops // J. Combin. Designs. 2007. V. 15. No. 2. P. 98–119. http://cs.anu.edu.au/7Ebdm/papers/ls_final.pdf
2. *Черемушкин А. В.* Почти все латинские квадраты имеют тривиальную группу автострофий // Материалы IX Междунар. семинара «Дискретная математика и ее приложения», посвященного 75-летию со дня рождения академика О. Б. Лупанова (Москва, МГУ, 18–23 июня 2007 г.) / Под ред. О. М. Касим-Заде. М.: Изд-во механико-математического факультета МГУ, 2007. С. 459–460.
3. *Черемушкин А. В.* Некоторые асимптотические оценки для класса сильно зависимых функций // Вестник Томского госуниверситета. Приложение. 2006. № 17. С. 87–94.
4. *Минк М.* Перманенты. М.: Мир, 1982.

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

DOI 10.17223/20710410/5/5

УДК 621.391.1:004.7

ДИСКРЕТНОЕ МОДЕЛИРОВАНИЕ ФИЗИКО-ХИМИЧЕСКИХ ПРОЦЕССОВ[1]

О. Л. Бандман

Институт вычислительной математики и математической геофизики СО РАН, г. Новосибирск, Россия

E-mail: bandman@ssd.sscc.ru

В статье дано систематическое описание собственных результатов по исследованию дискретных моделей кинетики физико-химических процессов на микро- и наноуровнях. Модели являются расширениями классического клеточного автомата (КА) фон Неймана, отличаясь от него тем, что в них допускаются произвольные, в том числе вероятностные, функции переходов над подмножествами состояний клеток, а также асинхронные и смешанные режимы функционирования. Для математического описания моделируемых процессов используются формализмы «Алгоритма параллельных подстановок». Приводятся условия корректности и оценки эффективности параллельных реализаций для синхронных и асинхронных КА-моделей. Все представленные модели иллюстрируются результатами компьютерного моделирования.

Ключевые слова: *дискретное моделирование, мелкозернистый параллелизм, асинхронный клеточный автомат, метод Монте-Карло, поверхностная химия, кинетика наносистем, параллельные вычисления.*

1. Введение и неформальная постановка задачи

Исследование дискретных моделей пространственной динамики началось в 80-х годах прошлого столетия. Интерес к дискретному моделированию явился следствием нового взгляда на клеточный автомат (КА) фон Неймана [1], который стал рассматриваться как дискретное выражение пространственно-временно́й функции [2]. Во всех точках дискретного пространства (клетках) значения этой функции (состояния клеток) изменяются по одним и тем же правилам (правилам переходов), которые выражаются в виде булевых функций от состояний клеток из некоторой окрестности. Все клетки могут перейти в новые состояния одновременно или в любом порядке, изменив таким образом глобальное состояние КА. Если установить КА в начальное глобальное состояние, он начинает эволюционировать, переходя из одного состояния в другое, имитируя пространственно распределенный процесс, т. е. выполняя ту же задачу, что и дифференциальное уравнение в частных производных. Принципиальное различие между этими двумя моделями состоит в том, что дискретный характер правил переходов позволяет выразить присущую физико-химическим процессам нелинейность

[1]Работа поддержана Программой фундаментальных исследований Президиума РАН № 2.15 (2009), Сибирским отделением РАН, Интеграционный проект 32 (2009).

и прерывность самым непосредственным образом, отображая события, составляющие моделируемый процесс: перемещения, химические взаимодействия и фазовые превращения некоторых абстрактных или реальных частиц, а в некоторых случаях молекул и атомов [3].

Такой «новый» взгляд на КА обусловлен, главным образом, двумя факторами. С одной стороны, появлением в нашем распоряжении супермощной вычислительной техники, которая позволяет моделировать КА с миллиардами клеток, что действительно требуется в реальных задачах, поскольку для моделирования поведения микро- и наночастиц нужны очень большие клеточные пространства. С другой стороны, выявившейся слабостью традиционных математических моделей, опирающихся на дифференциальные уравнения в частных производных, не способных описывать процессы и явления, которые на современном этапе развития науки и техники выходят на первый план, а именно: химические реакции, развитие живых организмов, поведение социума. Эти явления существенно нелинейны, диссипативны, синергетичны, обладают способностью к самоорганизации, самовоспроизводству и другими экзотическими свойствами. Поиск новых подходов к моделированию таких явлений пробудил интерес к дискретным методам моделирования вообще и особый интерес к КА как модели пространственной динамики.

Очевидно, что классический фоннеймановский КА с булевым алфавитом, булевыми правилами переходов и синхронным режимом функционирования не обладает достаточно широкими моделирующими способностями. Поэтому появилось множество модификаций КА, которые допускают целочисленный и символьный алфавиты, произвольные функции переходов, всевозможные структуры дискретных пространств и режимы функционировани, однако сохраняют два следующих свойства.

1) Функции переходов для всех точек пространства одинаковы и локальны, т. е. значение следующего состояния зависит только от состояний клеток из близкого окружения.

2) Алгоритм вычисления нового глобального состояния обладает внутренним параллелизмом: функции переходов клеток либо полностью независимы по переменным (синхронные КА, или СКА), либо частично независимы (асинхронные КА, или АКА), что делает их параллельную реализацию простой и эффективной.

Эти два свойства определяют широкий класс дискретных вычислительных моделей, объединенных понятием «*мелкозернистый параллелизм*» [5]. Дискретные модели физико-химических процессов, которым посвящена статья, составляют его подкласс, называемый иногда *диффузионно-реакционными КА* [5]. При этом диффузионная составляющая включает в себя имитацию пространственных перемещений частиц, а реакционная — имитирует химические преобразования (реакцию, диссоциацию, фазовые превращения). Значительную долю диффузионно-реакционных КА составляют асинхронные КА с вероятностными правилами переходов, которые иногда называют методом Монте-Карло.

Состояния клеток могут быть либо булевыми $0, 1$, имитируя факт наличия или отсутствия частицы в клетке, либо символами `C,CO,W,WC`, если частицы характеризуются химическими свойствами, либо целыми числами $1, 2, \ldots$, указывая на число частиц в клетке. К диффузионно-реакционным КА не относятся КА-модели, называемые «решеточным газом» (Lattice-Gas Models) или КА-гидродинамикой [7], у которых состояниями служат векторы единичных скоростей. Этот класс КА-моделей наиболее развит. Он получил широкое применение с названием Lattice-Gas или Lattice-Boltzmann Models и составляет самостоятельное направление в моделировании. Исключив его

из нашего рассмотрения, мы можем называть интересующие нас КА-модели просто клеточными автоматами, или КА.

Практическое применение КА получили в научных исследованиях для имитации физико-химических процессов, таких, как химические реакции на катализаторах [8, 9], эпитаксиальный рост кристаллов [10], образование покрытий и эрозии металлов [11] и др., где они успешно заменяют методы молекулярной динамики, которые требуют слишком большого времени вычисления даже на самых мощных компьютерах. К сожалению, большинство известных применений выполнено без опоры на какую-либо теорию или методологию и носят самые разные названия: решеточная модель [12], эволюционный алгоритм [13], микроскопическая модель [14], метод Монте-Карло [9, 10] и др. Почти все эти модели имеют ограничения по времени из-за трудности параллельной реализации на кластерах и отсутствия каких-либо методов преобразований и оптимизации. А между тем понятие КА полностью покрывает их моделирующие возможности, что позволило сделать некоторые обобщения [6], применить к ним единую формальную модель [15] и на ее основе разработать условия вычислительной корректности [16] и методологию эффективного моделирования, в том числе для большеразмерных задач на суперкомпьютерах [17]. Результаты таких исследований разрознены и фрагментарны, что затрудняет их осмысливание и использование. Чтобы продвинуть изучение и, следовательно, применение КА-моделей, предпринимается попытка в рамках этой статьи объединить уже опубликованные и еще не известные результаты по теории и практике их применения.

Кроме введения и заключения, статья содержит еще три раздела. Во втором разделе даны определения понятий и формальное определение модели КА и условий его корректного функционирования. Третий раздел посвящен методологии моделирования сложных процессов, четвертый — многопроцессорной реализации КА-алгоритмов.

2. Формальное представление и свойства КА-модели

2.1. Основные понятия и определения

Для формального представления КА далее используется математическая модель, называемая «Алгоритм параллельных подстановок» (АПП) [15]. Эта модель, разработанная для описания архитектуры параллельных алгоритмов и структур на СБИС, оказалась удобной для представления КА-моделей по двум причинам: 1) в ней определены *контекстные* конфигурации, необходимые для задания вероятности и временны́х ограничений, и 2) в ней допускаются *многоклеточные* правила переходов (один оператор меняет состояния нескольких клеток).

Согласно принципам АПП, центральным понятием в КА является *клетка*, которая задается парой (a, m), где $a \in A$ — состояние клетки из алфавита A, и $m \in M$ — имя клетки из множества имен M. Значения состояний клеток обозначаются символами из первой половины латинского алфавита $a, b, \ldots$, переменные состояний — буквами из его второй половины u, v, w, x, y, z. Множество имен в общем случае счетное, но в конкретных примерах обычно оно конечно и состоит из векторов координат в D-мерном дискретном пространстве. Поскольку на данном этапе исследованы АКА при $D \leqslant 2$, то $M = \{(i, j) : i = 0, 1, \ldots, I; j = 0, 1, \ldots, J\}$. Символ m вместо (i, j) используется в общих формулах для краткости.

Конкретные значения A и M определяют класс *клеточных массивов* $\Omega(A, M) = \Omega \subseteq A \times M$, каждый представитель $\Omega(A, M)$ которого не может иметь двух клеток с одинаковыми именами. На множестве имен вводятся *именующие функции* $\varphi_k : M \to M$, $k = 1, 2, \ldots, q$ для некоторого $q \geqslant 1$. Их значения связывают с лю-

бой клеткой $m \in M$ множество *соседних* клеток, совместно с которыми она образует *локальную конфигурацию*:

$$S(m) = \{(v_0, m), (v_1, \varphi_1(m)), \ldots, (v_k, \varphi_k(m)), \ldots, (v_q, \varphi_q(m))\}. \quad (1)$$

Пронумеровав и упорядочив соседей, локальную конфигурацию (1) можно задать двумя векторами:

$$\begin{aligned} U(S(m)) &= (v_0, \ldots, v_q), \\ T(S(m)) &= (m, \ldots, \varphi_k(m), \ldots, \varphi_q(m)). \end{aligned} \quad (2)$$

Первый вектор называется *вектором состояний* для конфигурации $S(m)$, второй — ее *определяющим шаблоном.*

Три локальные конфигурации

$$\begin{aligned} S(m) &= \{(v_0, \varphi_0(m)), (v_1, \varphi_1(m)), \ldots, (v_q, \varphi_q(m))\}, \\ S'(m) &= \{(v'_0, \varphi_0(m)), (v'_1, \varphi_1(m)), \ldots, (v'_q, \varphi_q(m))\}, \\ S''(m) &= \{(u_0, \psi_0(m)), (u_1, \psi_1(m)), \ldots, (u_n, \psi_n(m))\}, \end{aligned} \quad (3)$$

где $\varphi_0(m) = \psi_0(m) = m$, записанные в виде выражения

$$\theta(m): \ S(m) \star S''(m) \to S'(m), \quad (4)$$

составляют *параллельную подстановку* (далее просто подстановку). Эпитет «параллельная» означает, что она может быть применена ко всем $m \in M$ параллельно.

Локальная конфигурация $S(m)$ в (4) называется *базовой конфигурацией* (или базой), $S''(m)$ — *контекстом*, а $S'(m)$ — *правой частью* параллельной подстановки. База и правая часть подстановки имеют один и тот же определяющий шаблон. Шаблон контекста от них отличается. Состояния v'_k в клетках правой части являются значениями *функций перехода* от значений состояний в клетках базы и контекста, т. е.

$$v'_k = f_k(v_0, v_1, \ldots, v_q, u_0, \ldots, u_n). \quad (5)$$

Локальная конфигурация $S(m)$ с переменными состояниями считается *совместимой* с клеточным массивом $\Omega(A, M)$, что обозначается как $S(m) \sqsubseteq \Omega$, если

1) для всех $(v_k, \varphi_k(m)) \in S(m)$, $k = 0, 1, \ldots, q$, и всех $(u_l, \psi_l(m)) \in S''(m)$, $l = 0, \ldots, n$, переменные v_k и u_l определены в алфавите A;

2) одноименные клетки с константными состояниями $(a, \varphi_k(m)) \in S(m)$ и $(b, \varphi_k(m)) \in \Omega$ равны, т. е. $a = b$.

Параллельная подстановка $\theta(m)$ *применима* к клеточному массиву $\Omega(A, M)$, если в нем найдется непустое подмножество клеток $M' \subseteq M$ с именами $m' \in M'$, для которых удовлетворяется условие

$$S(m') \bigcup S''(m') \sqsubseteq \Omega. \quad (6)$$

Применимая подстановка называется *локальным оператором.*

Применение локального оператора $\theta(m)$ к Ω в клетке $m' \in M'$ состоит в замене состояний в клетках $(a_k, \varphi_k(m'))$ на значения функций f_k, вычисленных по (5) (рис. 1). Выполнение локального оператора происходит за время τ, называемое *тактом.*

Применение $\theta(m)$ ко всем клеткам клеточного массива $\Omega(t)$ приводит к изменению его глобального состояния, $\Omega(t) \xrightarrow{\theta(m)} \Omega(t+1)$, и называется *итерацией.* Последовательность

$$\Sigma(\Omega) = \Omega, \Omega(1), \ldots, \Omega(t), \Omega(t+1), \ldots, \Omega(T), \quad (7)$$

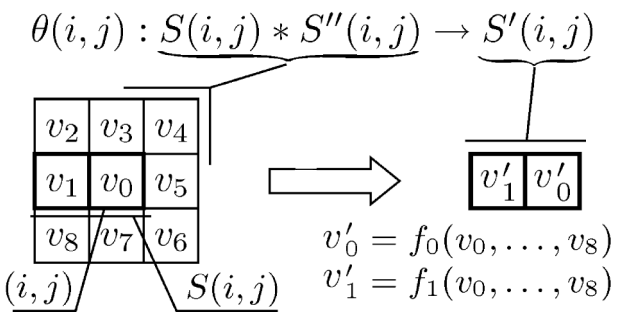

Рис. 1. Графическое представление локального оператора

полученная в результате итеративного функционирования КА, называется *эволюцией*.

Эволюция КА является дискретным представлением пространственно-временной функции. Если итеративный процесс сходится, то эволюция КА имеет завершение, т. е. существует $t = T$, такое, что $\Omega(T) = \Omega(T+1) = \Omega(T+2) = \ldots$ В иных случаях КА моделирует колебательный или хаотический процесс [18].

Левая часть локального оператора (4) разделена на базу и контекст для того, чтобы при применении его к любой конкретной клетке $m_k \in M$ отделить множества клеток, состояния которых она изменяет, от множества клеток $S''(m_k)$, состояния которых она не меняет, но использует в качестве переменных для вычисления функции переходов. При этом состояния клеток из контекста $S''(m_k)$ могут быть изменены на той же итерации применением этого же локального оператора к другой клетке m_g, если

$$S''(m_k) \cap S(m_g) \neq \varnothing. \tag{8}$$

Контексты подразделяются на контексты первого рода и контексты второго рода.

— *Контексты первого рода* содержат клетки вида $(u_k, \psi_k(m))$, в которых $\psi_k(m)$ определены в области эволюции КА и, чаще всего, находятся в непосредственной близости от базовых клеток.

— *Контексты второго рода* выполняют управляющие функции. Их локальные конфигурации содержат клетки, в которых состояния являются значениями предикатов применимости основного локального оператора, а имена не содержатся в пространстве моделирования и задаются отдельным подмножеством M'':

$$S''(m) = \{(\alpha_1, m''_1), \ldots, (\alpha_r, m''_r)\}, \tag{9}$$

где $\{\alpha_1, \ldots, \alpha_r\} = A''$ — контекстный алфавит, $\{m''_1, \ldots, m''_r\} = M''$ — множество имен контекстных клеток и $M'' \cap M = \varnothing$.

Для вычисления состояний контекстных клеток необходимы дополнительные *контекстные операторы* вида

$$\theta''_k : (x_k, m''_k) \to (\alpha_k, m''_k), \quad k = 1, 2, \ldots, r. \tag{10}$$

Состояния $\alpha_k \in A''$ являются предикатами, зависящими от внешних переменных x_k, например значений генератора случайных чисел, или значений тактирующих импульсов, или указателей области значений имен клеток и др.

В диффузионно-реакционных КА обычно используются следующие три типа управляющих контекстов, они все имеют вид (10) и различаются только функциями переходов $\alpha_k(x_k)$.

Тактирующий контекст применяется для организации n-тактного синхронного режима. В нем используется оператор с предикатом $\alpha_{_T}$, который зависит от номера шага или итерации:

$$\alpha_{_T}(t) = \begin{cases} 1, & \text{если} \quad t \bmod n = 0, \\ 0, & \text{если} \quad t \bmod n \neq 0. \end{cases} \tag{11}$$

Вероятностный контекст применяется в вероятностных КА, в его операторе θ_p (10) предикат α_p зависит от случайного числа,

$$\alpha_p(rand) = \begin{cases} 1, & \text{если} \quad rand < p, \\ 0, & \text{если} \quad rand \geqslant p, \end{cases} \tag{12}$$

где $rand$ — случайное число в интервале [0,1]; p — вероятность применимости подстановки.

Пространственный контекст применяется для выделения тех областей пространства моделирования, где соответствующий оператор применим. В его операторе θ_s предикат зависит от имени клетки $m \in M$:

$$\alpha_s(m) = \begin{cases} 1, & \text{если} \quad m \in M', \\ 0, & \text{если} \quad m \notin M', \end{cases} \tag{13}$$

где $M' \subset M$.

Составной контекст применяется, когда сразу несколько условий необходимо выполнить, чтобы основной локальный оператор был применим. В частности, для случайного выбора координат клетки $(i, j) \in M$, к которой применяется локальный оператор при асинхронном режиме функционирования, нужен *пространственно-вероятностный* контекст с предикатом вида

$$\alpha_{ps}(i,j) = \begin{cases} 1, & \text{если} \quad (i = rand_1 \cdot N) \;\&\; (j = rand_2 \cdot N), \\ 0, & \text{в иных случаях.} \end{cases} \tag{14}$$

Однако, поскольку для всех АКА $\aleph_\alpha$ контекст (14) одинаков, его можно убрать из выражений локальных операторов, оставив только индекс α при $\aleph$.

Управляющая роль контекстов второго рода распространяется также на распределение работы между несколькими основными локальными операторами $\theta_k(m), \ldots,$ $\theta_q(m)$, участвующими в описании единого физико-химического процесса. Совместно с контекстными операторами они составляют *систему локальных операторов* $\Theta = \{\theta_1(m), \ldots, \theta_q(m), \theta''_1, \ldots, \theta''_p\}$.

Пример 1. Вероятностный КА, моделирующий простую одномерную диффузию, имеет булев алфавит $A = \{0, 1\}$ и пространство моделирования, состоящее из 15 клеток $M = \{0, 1 \ldots, i, \ldots, 14\}$. Концентрация вещества моделируется количеством «единиц» в некоторой *окрестности осреднения* $Av(i)$. Пусть $Av(i) = \{i-1, i, i+1\}$, $|Av(i)| = 3$. Исходный клеточный массив $\Omega(0) = \{(1, i) : i = 1, \ldots, 4\} \cup \{(0, i) : i = 5, \ldots, 15\}$. Процесс диффузии состоит в перемешивании вещества с целью получения равномерно распределенной концентрации. КА моделирует его путем обмена состояниями между соседними клетками. Поскольку у каждой клетки два соседа, то выбор левого $(i-1)$ или правого $(i+1)$ соседа выполняется операторами $\theta_1(i)$ или $\theta_2(i)$ с вероятностью $p = 0{,}5$. Кроме того, каждая итерация разбивается на три такта t_0, t_1, t_2 таким образом, чтобы на k-м такте для каждого k операторы применялись к клеткам с именами $i = k (\bmod 3)$. Для этого нужен тактирующий контекст, указывающий

номер такта, и пространственный контекст для выделения подмножеств клеток, работающих на соответствующем такте. Таким образом, система локальных операторов $\Theta = \{\theta_1(m), \theta_2(m), \theta''_{_T}, \theta''_p, \theta''_s\}$ имеет вид

$$\begin{array}{ll} \theta_1(i): & \{(u,i),(v,i-1)\} \star \{(1,m_{_T}),(1,m_p)(1,m_s)\} \to \{(v,i),(u,i-1)\}, \\ \theta_2(i): & \{(u,i),(v,i+1)\} \star \{(1,m_{_T}),(0,m_p)(1,m_s)\} \to \{(v,i),(u,i+1)\}, \\ \theta''_{_T}: & (t,m_{_T}) \to (\alpha_{_T}(t),m_{_T}), \\ \theta''_p: & (rand_2,m_p) \to (\alpha_p(rand_2),m_p), \\ \theta''_s: & (i,m_s) \to (\alpha_s(i),m_s), \end{array} \qquad (15)$$

где $\alpha_{_T}$ вычисляется по (11) с $n = 3$, $\alpha_p(rand)$ вычисляется по (12) с $p = 0{,}5$, и $\alpha_s(k)$ — по (13) с $M' = \{i : i = k(\mathrm{mod}\,3)\}$.

Эволюция процесса (15) показана на рис. 2. Она не имеет завершения, а приходит к колебательному процессу вокруг глобального состояния при $t > 26$, в котором осредненное значение плотности $\langle u(i)\rangle \simeq 1/3$.

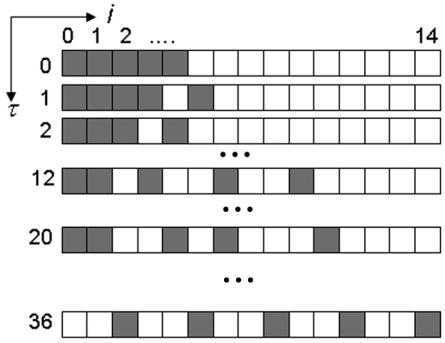

Рис. 2. Эволюция КА одномерной диффузии (15)

Если контекст второго рода может быть выражен простым предикатом, то его можно заменить надписью над стрелкой в выражении локального оператора. Например, в (15) $\theta_1(i)$ может иметь следующий вид:

$$\theta_1(i): \quad \{(u,i),(v,i-1)\} \overset{\alpha_p,\alpha_{_T},\alpha_s}{\longrightarrow} \{(v,i),(u,i-1)\}. \qquad (16)$$

2.2. Режимы функционирования и условия корректности

Применение параллельных подстановок к клеткам клеточного массива может происходить в разных режимах. Режим определяет порядок применения локальных операторов к клеткам массива за время одной итерации. Основными режимами являются: синхронный, обозначаемый символом σ, и асинхронный, обозначаемый символом α. Соответственно синхронный КА обозначается $\aleph_\sigma$, а асинхронный — $\aleph_\alpha$. Названные два режима являются двумя крайними случаями. Известны также КА со смешанными и производными режимами, например блочно-синхронный КА $\aleph_\beta$ [19].

При синхронном режиме изменение состояний происходят только после того, как все функции $f_k(v_0,\ldots,v_q)$ для всех клеток вычислены, т. е. известны новые состояния всех клеток массива. Если предположить, что все клетки реализованы аппаратурно и выполняют вычисления функций перехода одновременно, реализуя потенциальный внутренний параллелизм КА, далее называемый *клеточным параллелизмом*, то время итерации равно одному такту. В обычном последовательном компьютере синхронная параллельная смена состояний клеток имитируется путем вычисления новых

состояний всех клеток последовательно при сохранении их в другом массиве, который служит глобальным состоянием для следующей итерации. Такая параллельность называется *виртуальной*, и вычисление нового глобального состояния занимает время $\Delta t = |M| \cdot \tau$ тактов.

При асинхронном режиме ни реальной, ни виртуальной одновременности смены состояний клеток не предполагается. На каждом такте происходит применение $\theta(m)$ к одной случайным образом выбранной клетке с именем $m \in M$, которая вычисляет функцию перехода от тех состояний соседей, которые на данный момент имеют место, и сразу производит замену текущего состояния на новое. Это значит, что функции переходов f_k (5) могут иметь в качестве аргументов как текущие, так и новые состояния, например $(v_k, \varphi_k(m)) \in \Omega(t)$, а $(v_l, \varphi_l(m)) \in \Omega(t+1)$. Одна итерация выполняется за $\Delta t = |M| \cdot \tau$ тактов, т. е. за то же время, что и при виртуальном синхронном режиме. Внутренний параллелизм проявляется только в том, что смены состояний клеток происходят в произвольном порядке.

Для построения КА-моделей важным является тот факт, что два КА, отличающиеся только режимами, могут иметь совершенно разные эволюции.

Пример 2. Процесс агрегации частиц одной жидкости в смеси из двух жидкостей может быть представлен классическим КА, получившим название *КА разделения фаз* [20]. Пусть в начальном состоянии жидкость B (черная) размешана в жидкости C (белой). Концентрация B по объему равна $\rho_B(0)$. Как только размешивание прекращается, в смеси начинается агломерация частиц вещества B под действием межмолекулярных сил связи. В двумерном приближении процесс выглядит как появление черных пятен, которые постепенно растут и меняют свою форму и размеры. Рост пятен постепенно замедляется, приближаясь к одному из двух устойчивых состояний: либо «глобальное черное», либо «глобальное белое». При этом существует некоторое критическое значение начальной концентрации ϱ, такое, что если $\rho_B(0) > \varrho$, то процесс стремится к «глобальному черному», если $\rho_B(0) < \varrho$, то процесс стремится к «глобальному белому», и процесс застывает на каком-то образе при $\rho_B(0) = \varrho$. КА, моделирующий этот процесс, имеет алфавит $A_1 = \{0, 1\}$, множество имен $M = \{(i, j) : i, j = 0, \ldots, N\}$ и следующий локальный оператор

$$\theta_1(i,j) : \{(v,(i,j))\} \star S''(i,j) \to \{(u,(i,j))\}, \tag{17}$$

где $(i,j) \in M$, $S''(i,j) = \{(v_k, \varphi_k(i,j)) : \ \varphi_k(i,j) = (i+g, j+h)\}$, $g, h \in \{-2, -1, 1, 2\}$,

$$u \ = \ \begin{cases} 1, & \text{если} \quad s < 24 \text{ или } s = 25, \\ 0, & \text{если} \quad s > 25 \text{ или } s = 24, \end{cases} \qquad s = \sum_{g=-2}^{2} \sum_{h=-2}^{2} v_{i+g,j+h}.$$

Исследование КА (17) показало, что его эволюции при синхронном и асинхронном режимах сильно различаются. На рис. 3 приведены четыре состояния из эволюции синхронного КА при $\rho_B(0) = 0{,}5$. При $t = 3000$ глобальное состояние больше практически не меняется, т. е. критическая концентрация ϱ равна 0,5. При асинхронном режиме $\varrho = 0{,}2245$, и даже при вдвое меньшей начальной концентрации КА быстро эволюционирует к черному глобальному состоянию, показывая почти незаметные изменения на итерациях $t > 200$ (рис. 4).

Пример 2 служит иллюстрацией к тому факту, что КА-модель определена полностью, только если указан режим ее функционирования.

Индексом σ обозначается синхронный режим, индексом α — асинхронный и индексом β — блочно-синхронный. *Алгоритм функционирования* КА $\aleph$ полностью опреде-

лен, если, кроме этого, задан исходный клеточный массив $\Omega(0)$, т. е.

$$\aleph_{\varkappa} = \langle A, M, \Theta, \Omega(0) \rangle, \quad \varkappa \in \{\alpha, \beta, \sigma\}. \tag{18}$$

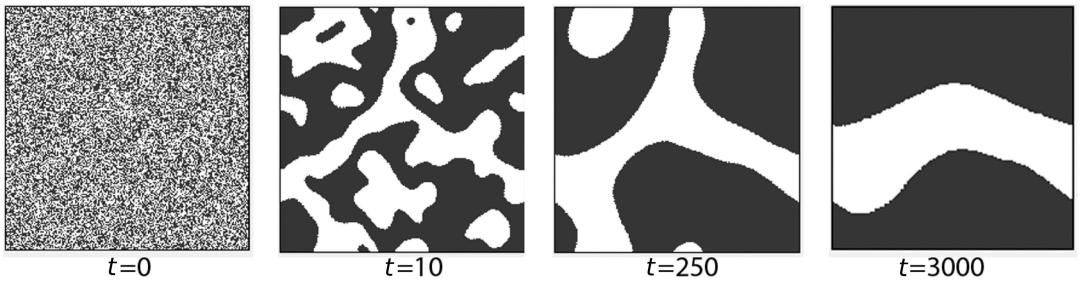

Рис. 3. Эволюция КА «разделения фаз» (18) при синхронном режиме функционирования

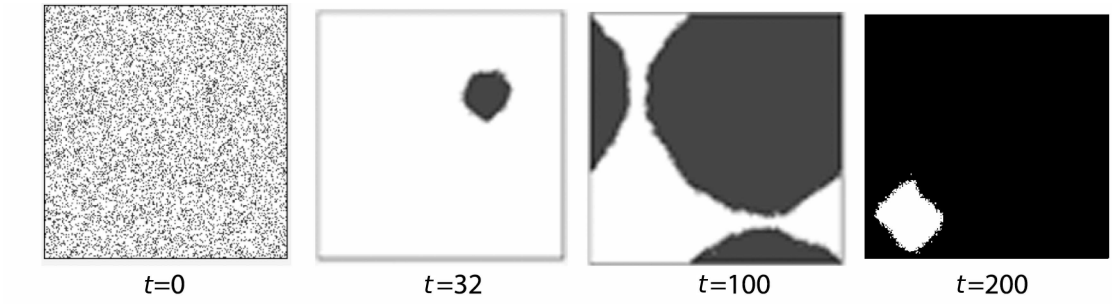

Рис. 4. Эволюция КА «разделения фаз» (18) при асинхронном режиме функционирования

2.3. Условия корректного функционирования

Как всякая математическая модель, КА должен удовлетворять условиям поведенческой и вычислительной корректности. Эти условия для КА просты и естественны, но легко могут быть нарушены при крупноблочном распараллеливании.

Условие детерминированности: *локальный оператор* $\theta(m) : S(m) \to S'(m)$ *в один и тот же момент* τ *может изменять состояние не более чем одной клетки из каждого подмножества клеток* $S'(m)$ для всех $m \in M$.

Это условие гарантирует отсутствие коллизий, т. е. таких ситуаций, когда одной и той же клетке на одном и том же такте предписано сменить свое состояние разными функциями переходов $f_k(m)$ и $f_l(\varphi_g(m))$. Это значит, что одновременность (пусть виртуальная) применения $\theta(m)$ допускается только к такому подмножеству клеток $M_k \subseteq M$, для которого выполняется условие

$$\forall m_g, m_h \in M_k \ \left(m_g \neq m_h \Rightarrow T(S'(m_g)) \bigcap T(S'(m_h)) = \varnothing \right). \tag{19}$$

Условие (19) не выполнено, например, в синхронном КА, если две клетки должны изменить состояния одновременно (рис. 5).

Очевидно, что (19) всегда выполняется для классических КА, так как их локальные операторы имеют одноклеточную базовую локальную конфигурацию, т. е. $|S'(m)| = 1$, и также в асинхронных КА, где одновременно может быть применен только один оператор и только к одной клетке.

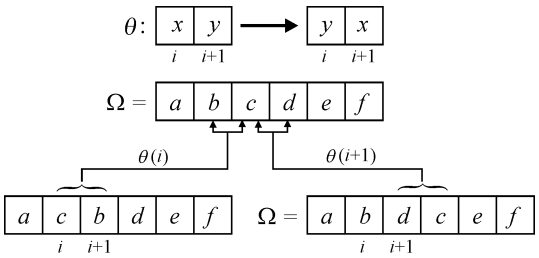

Рис. 5. Графическое представление недетерминированного поведения. Одновременное применение $\theta(i)$ и $\theta(i{+}1)$ приводит к конфликту в клетках $i, i+1, i+2$

Условие равноправия клеток: *на t-й итерации для каждого t локальный оператор $\theta(m)$ применяется ко всем клеткам $(v, m) \in \Omega(t)$, причем к каждой клетке только один раз.*

Это условие гарантирует, что все клетки участвуют в процессе эволюции в равной степени. Нарушение этого условия может быть вызвано несовершенством генератора случайных чисел, а также неправильной организацией взаимодействий между областями КА, если они расположены на разных процессорах. Очевидно, что синхронные КА всегда удовлетворяют условию равноправия, так как режим предусматривает детерминированный порядок применения локального оператора ко всем клеткам из $\Omega(t)$.

3. Моделирование физико-химических процессов

3.1. Локальные операторы физико-химических процессов

Накопленный опыт по дискретному моделированию процессов на микро- и наноуровнях связан, в основном, с изучением химических реакций на каталитических поверхностях, причем в большинстве изученных моделей используется один и тот же набор элементарных действий. Пусть $A = \{a, b, c, \varnothing\}$, где a, b, c обозначают наличие частицы определенного вещества, а $\varnothing$ — свободное место, $M = \{(i, j) : i, j = 0, \ldots, N\}$. Тогда типичные элементарные действия поверхностной химии можно отобразить в КА-моделях следующим образом.

1. *Адсорбция* моделирует попадание частицы на пустую клетку с заданной вероятностью p_a, которая вычисляется исходя из физических данных (парциальное давление газа над поверхностью, скорость струи, направленной на поверхность, и др.):

$$\theta_a : (\varnothing, (i, j)) \xrightarrow{p_a} (a, (i, j)). \tag{20}$$

Адсорбция иногда сопровождается диссоциацией, например, молекула H_2 или O_2, попадая на поверхность катализатора, разлагается на два атома. Тогда ей требуются две соседние пустые клетки, которые могут быть смежными по оси i:

$$\theta_{2a} : \{(\varnothing, (i, j)), (\varnothing, (i+1, j))\} \xrightarrow{p_b} \{(b, (i, j)), (b, (i+1, j))\}, \tag{21}$$

или по оси j:

$$\theta'_{2a} : \{(\varnothing, (i, j)), (\varnothing, (i, j+1))\} \xrightarrow{p_b} \{(b, (i, j)), (b, (i, j+1))\}. \tag{22}$$

Десорбция, или сублимация, — действие, обратное адсорбции, обозначающее освобождение клетки. Поскольку простое исчезновение противоречит природе явлений, то обычно десорбция указывает на перемещение частицы из области моделирования во

внешнюю среду. Локальный оператор имеет вид (20), в котором левая и правая части поменялись местами, т. е.

$$\theta_{ds} : (a,(i,j)) \xrightarrow{p_a} (\varnothing,(i,j)). \tag{23}$$

Химическая реакция — действие, в котором участвует несколько разных веществ. Здесь возможно несколько вариантов.

1) Если две частицы, вступающие в реакцию $a + b \to c$, расположены в смежных клетках и продукт реакции остается в области моделирования, то он может занять любую из двух освобождающихся клеток равновероятно, что должно быть представлено четырьмя локальными операторами, например:

$$\begin{array}{ll}
\theta_{1r}: & \{(a,(i,j)),(b,(i+1,j))\} \xrightarrow{p_r} \{(\varnothing,(i,j))(c,(i+1,j))\},\\
\theta_{2r}: & \{(a,(i,j)),(b,(i+1,j))\} \xrightarrow{1-p_r} \{(c,(i,j))(\varnothing,(i+1,j))\},\\
\theta_{3r}: & \{(b,(i,j)),(a,(i+1,j))\} \xrightarrow{p_r} \{(\varnothing,(i,j))(c,(i+1,j))\},\\
\theta_{4r}: & \{(b,(i,j)),(a,(i+1,j))\} \xrightarrow{1-p_r} \{(c,(i,j))(\varnothing,(i+1,j))\},
\end{array} \tag{24}$$

где $p_r = 0{,}5$.

Если реакция сопровождается сублимацией, то для ее описания достаточно одного оператора (24), у которого в правой части обе клетки пусты.

Диффузия. Процесс диффузии есть блуждание частиц в стремлении выровнять концентрацию вещества в пространстве. Наиболее естественным образом этот процесс отображается вероятностным АКА, впервые введенным в [19] и названным там «наивной диффузией». АКА наивной диффузии имеет булев алфавит, дополненный множеством контекстных символов $A'' = \{0,1,\ldots,N\}$, и множество имен $M = \{(i,j) : i,j = 0,1,\ldots,N\}$, дополненное контекстным множеством $M'' = \{m_s, m_i, m_j\}$. Система локальных операторов содержит один основной локальный оператор

$$\begin{array}{ll}
\theta_d: & \{(u,(i,j)),(u_k,\varphi_k(i,j))\} \star \{(i,m_i),(j,m_j),(k,m_k)\} \to\\
& \to \{(u_k,(i,j)),(u,\varphi_k(i,j))\},
\end{array} \tag{25}$$

один вероятностный контекстный для случайного выбора соседа $\varphi_k(i,j)$

$$\theta''_k : (x,m_k) \to (k,m_k), \quad k = \begin{cases}
1, & \text{если} \quad 0 & < rand_1 & \leqslant 0{,}25,\\
2, & \text{если} \quad 0{,}25 & < rand_1 & \leqslant 0{,}5,\\
3, & \text{если} \quad 0{,}5 & < rand_1 & \leqslant 0{,}75,\\
4, & \text{если} \quad 0{,}75 & < rand_1 & \leqslant 1
\end{cases} \tag{26}$$

и два контекстных оператора для случайного выбора клетки

$$\begin{array}{lll}
\theta''_i: & (x,m_i) \to (i,m_i), & i = rand_1 \cdot N,\\
\theta j'': & (x,m_j) \to (j,m_j), & j = rand_2 \cdot N.
\end{array} \tag{27}$$

Если диффузия анизотропна, то интервалы вероятностей в (26) не одинаковы, а распределены в соответствии со значениями коэффициентов диффузии вдоль соответствующих направлений.

На рис. 6 показаны три глобальных состояния эволюции КА двумерной наивной КА-диффузии. Исходным состоянием служил заполненный нулями клеточный массив размерами 200×200, в центральной части которого имеется квадратная область Ω' высокой концентрации частиц:

$$\Omega' = \{(1,(i,j)) : i,j = 70,\ldots,130\}.$$

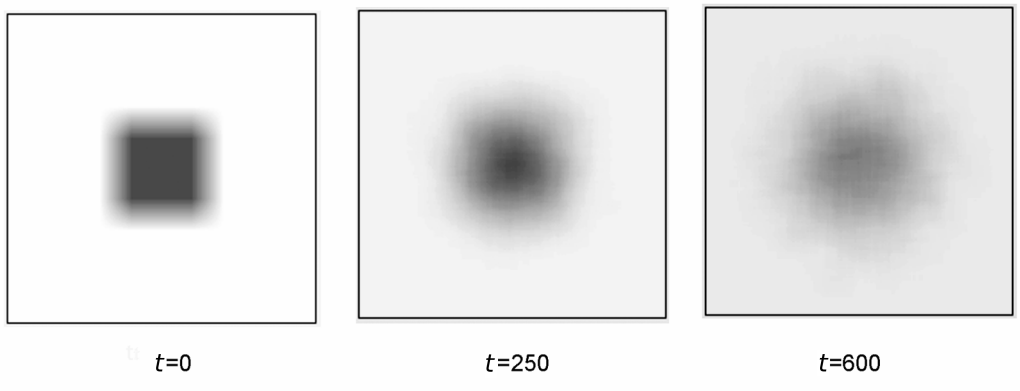

Рис. 6. Наивная асинхронная КА-диффузия: глобальные состояния клеточного массива при $t = 0, 250, 600$

Из рис. 6 видно, что процесс диффузии постепенно превращает плотный квадрат в круг с уменьшающейся от центра концентрацией. Для получения модельного коэффициента диффузии производилось осреднение концентрации в каждой точке по окрестности с радиусом $r = 8$ клеток. Сопоставление осредненных клеточных массивов с решениями дифференциального уравнения Лапласа позволило вычислить модельный коэффициент диффузии C_{mod}, который должен быть равен безразмерному коэффициенту в уравнении Лапласа $C = \dfrac{d \cdot \tau}{h^2}$, где d (м^2/с) — физический коэффициент диффузии, τ (с) — шаг по времени, h (м) — шаг по длине в дифференциальном уравнении [21]. Соотношение коэффициентов C и C_{mod} позволяет вычислить масштабы, связывающие дискретные модельные параметры с реальными физическими значениями.

3.2. Композиции локальных операторов

Построение КА-моделей сложных процессов требует использования в одном алгоритме нескольких взаимодействующих локальных операторов, которые составляют композиции разного типа [22]. При моделировании физико-химических процессов используются чаще всего *суперпозиция* локальных операторов и *вероятностная последовательная* композиция.

Суперпозиция нескольких локальных операторов — это один локальный оператор вида

$$\theta(m) = \theta_l(\theta_{l-1}(\dots(\theta_1(m)))), \tag{28}$$

который может применяться как в синхронном, так и в асинхронном режиме, причем компонентные операторы всегда применяются к клетке в заданном в (28) порядке и не могут быть разделены применением другого локального оператора к другой клетке.

Вероятностная последовательная композиция множества операторов

$$\Theta(m) = \{\theta_1(m), \theta_2(m), \dots \theta_l(m)\} \tag{29}$$

предполагает применение компонентных операторов в случайном порядке к случайно выбранной клетке, причем в течение одной итерации каждый $\theta_k(m) \in \Theta(m)$ применяется к каждой клетке $m \in M$ по одному разу. Такой режим функционирования называется вероятностным асинхронным, или в [8, 9] методом Монте-Карло.

Пример 3. Упрощенная АКА-модель химической реакции окисления монооки-си углерода на катализаторе, называемая иногда ZBG-реакцией по инициалам авторов [23], подробно исследована в [8, 9]. Над поверхностью катализатора имеется смесь

двух газов: кислорода и монооксиси углерода с парциальными давлениями p_b и p_a соответственно. В АКА-модели поверхность катализатора, на котором происходит реакция, представлена клеточным массивом $\Omega(A, M)$, в котором $M = \{(i, j) : i, j = 0, 1, \ldots, N\}$, $A = \{a, b, \varnothing\}$, причем $(a, (i, j))$, $(b, (i, j))$ и $(\varnothing, (i, j))$ обозначают клетки, в которых находятся молекулы CO, O, или клетка пуста соответственно. В алгоритме используется вероятностная последовательная композиция множества $\{\theta_1(i, j), \theta_2(i, j), \theta_3(i, j), \theta_4(i, j)\}$ локальных операторов, моделирующих следующие элементарные действия.

1) Адсорбция CO из газа — $\theta_1(i, j)$: если клетка (i, j) пуста, ее занимает молекула CO с вероятностью p_a.

2) Адсорбция O_2 из газа — $\theta_2(i, j)$: если клетка (i, j) пуста и имеет пустого соседа, то с вероятностью p_b в каждой их них оказывается молекула кислорода. Один из четырех соседей клетки (i, j) выбирается с вероятностью $p_n = 0{,}25$.

3) Реакция окисления CO ($CO+O \to CO_2$) — $\theta_3(i, j)$: если в клетке (i, j) находится молекула CO и в соседней клетке находится молекула кислорода, то между ними происходит реакция. В результате получается молекула CO_2, которая переходит в газ. Обе клетки опустошаются. Соседняя клетка выбирается с вероятностью $p_n = 0{,}25$.

4) Та же самая реакция окисления CO ($O+CO \to CO_2$) происходит, если в клетке (i, j) находится молекула кислорода, а в соседней клетке имеется молекула CO — $\theta_4(i, j)$:

$$
\begin{array}{ll}
\theta_1(i,j): & \{(\varnothing,(i,j))\} \xrightarrow{p_a} \{(a,(i,j))\}, \\
\theta_2(i,j): & \{(\varnothing,(i,j))(0,\varphi_{k_2}(i,j))\} \star (k_2, m_1) \xrightarrow{p_b} \{(b,(i,j)),(b,(\varphi_{k_2}(i,j))\}, \\
\theta_3(i,j): & \{(a,(i,j))(b,\varphi_{k_3}(i,j))\} \star (k_3, m_2) \xrightarrow{p_n} \{(\varnothing,(i,j)),(\varnothing,(\varphi_{k_3}(i,j))\}, \\
\theta_4(i,j): & \{(b,(i,j))(a,\varphi_{k_4}(i,j))\} \star (k_4, m_3) \xrightarrow{p_n} \{(\varnothing,(i,j)),(\varnothing,(\varphi_{k_4}(i,j))\}.
\end{array}
\tag{30}
$$

В $\theta_2(i, j), \theta_3(i, j), \theta_4(i, j)$ состояния $k_2, k_3, k_4 \in \{1, 2, 3, 4\}$ указывают на одного из четырех соседей клетки (i, j), а именно: $\varphi_1(i, j) = (i, j+1)$, $\varphi_2(i, j) = (i+1, j)$, $\varphi_3(i, j) = (i, j-1)$, $\varphi_4(i, j) = (i-1, j)$. Значения k_2, k_3, k_4 вычисляются соответствующими вероятностными контекстными операторами $\theta_2'', \theta_3'', \theta_4''$ вида (26).

На рис. 7 показаны три глобальных состояния, полученных при моделировании реакции на клеточном массиве размером 200×200 с периодическими граничными условиями.

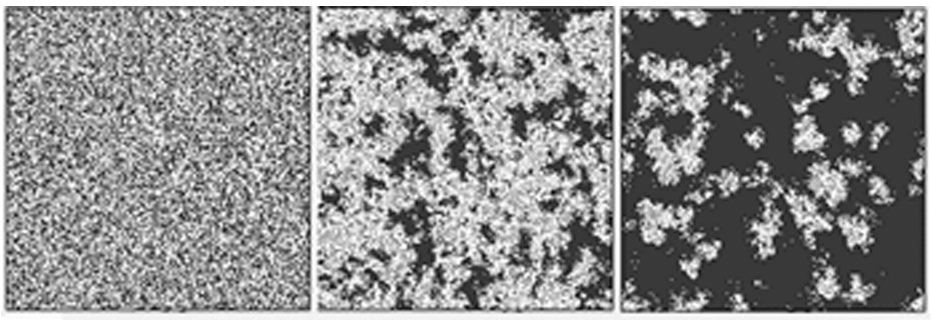

Рис. 7. Три глобальных состояния эволюции АКА $\aleph_\alpha$. Черные пикселы обозначают CO, серые — O, белые — пустые места

4. Многопроцессорные реализации КА-алгоритмов

Свойства мелкозернистости КА-алгоритмов предопределяют способ их многопроцессорной реализации с использованием декомпозиции области моделирования [17].

Пусть клеточный массив $\Omega(A, M)$ имеет структуру прямоугольника, который разделен на n равных по количеству клеток *доменов* размерами $|\mathrm{Dom}| = |M|/n$ клеток. Эффективность распараллеливания выражается формулой

$$Q(n) = \frac{t_\Omega}{n \cdot t_{\mathrm{dom}} + L \cdot t_{ex}}, \tag{31}$$

где $t_\Omega, t_{\mathrm{dom}}$ — времена, необходимые для выполнения одной итерации на всем КА и на одном домене соответственно, т. е. $t_\Omega = \tau \cdot |M|$; $t_{\mathrm{dom}} = \tau \cdot |\mathrm{Dom}|$; t_{ex} — время, затрачиваемое на один межпроцессорный обмен; L — число обменов за одну итерацию. Очевидно, что для получения высокой эффективности необходимо, чтобы

$$\tau \cdot |\mathrm{Dom}| >> L \cdot t_{ex}. \tag{32}$$

В синхронных КА обмен данными производится один раз за итерацию, т. е. $L = 1$. Если принять $t_{ex}/\tau \simeq 10^3$ (что соответствует реальности), то условие (32) легко выполняется при $|\mathrm{Dom}| \geqslant 10^4$. Иными словами, высокая эффективность параллельных реализаций достигается без труда.

Для асинхронных КА межпроцессорный обмен должен выполняться всякий раз, когда изменяется состояние хотя бы одной клетки, находящейся на границе домена. Если это условие не выполняется, то не выполняются условия корректности (19), поскольку при стохастическом выборе клеток невозможно гарантировать, что новое состояние пограничной клетки (v, m_g), $m_g \in \mathrm{Dom}_k$, не понадобится на следующем же такте для вычисления нового состояния пограничной клетки соседнего домена (u, m_l), если $m_l \in \mathrm{Dom}_l$, $m_l = \varphi_k^{-1}(m_g)$, $\varphi_k(m_g) \in T_S$. Отсюда следует, что для асинхронных КА за одну итерацию надо выполнить $L = P$ пересылок, где P — периметр домена. Легко проверить, что при соотношении $t_{ex}/\tau \simeq 10^3$ приемлемая эффективность ($>70\,\%$) достигается при $|\mathrm{Dom}| \geqslant 10^7$, что делает распараллеливание нецелесообразным.

Поскольку препятствием для эффективного распараллеливания является асинхронизм, то возникает соблазн ввести частичную синхронизацию режима функционирования, при этом

а) по возможности, сохранить эволюцию КА;

б) не нарушить условий корректности.

Это можно сделать, трансформировав асинхронный КА $\aleph_\alpha = \langle A, M, \Theta \rangle$ в *блочно-синхронный* КА $\aleph_\beta = \langle A, M, \Theta \rangle$ следующим образом [16].

1. На множестве имен M для каждого $m \in M$ определяется шаблон

$$B(m) = \{\varphi_0(m), \varphi_1(m), \ldots, \varphi_q(m), \varphi_{q+1}(m), \ldots, \varphi_r(m)\},$$

называемый *блоком*, который включает в себя определяющий шаблон $T(S(m))$ базовой локальной конфигурации $S(m)$:

$$T(S(m)) \subseteq B(m). \tag{33}$$

2. Строится разбиение M на *независимые подмножества* $M_1, \ldots, M_r$, для которых выполняются следующие соотношения:

$$|M_k| = \frac{|M|}{r}, \qquad k = 1, \ldots, r; \tag{34}$$

$$\bigcup_{k=1}^{r} M_k = M, \qquad \forall k, l \in \{1, \ldots, r\} \left(k \neq l \Rightarrow M_k \bigcap M_l = \varnothing \right); \tag{35}$$

$$\bigcup_{m\in M_k} B(m) = M, \qquad \forall m_g, m_l \in M_k \left(m_g \neq m_l \Rightarrow B(m_g) \bigcap B(m_l) = \varnothing \right). \tag{36}$$

Иными словами, вводится r разбиений множества имен M блоками одного размера, причем каждый блок $B(m)$ включает в себя базовый шаблон $T(S(m))$.

3. Смена глобального состояния $\Omega(t) \xrightarrow{\Theta(M)} \Omega(t+1)$ производится за r стадий. На каждой стадии производится применение всех локальных операторов ко всем клеткам из одного случайно выбранного независимого подмножества M_j, $j = 1, 2, \ldots, r$. Режим функционирования в пределах стадии — асинхронный, т. е. порядок выбора клеток в независимом подмножестве произвольный. Это не нарушает условия (19), так как функции переходов не пересекаются по переменным, что гарантируется условием (33). Следовательно, результат глобального перехода к следующей стадии — единственный и одинаков как для синхронного, так и для асинхронного режимов.

Тот факт, что при блочно-синхронном режиме на каждой стадии функции переходов разделены по переменным, позволяет производить обмен данными между доменами после завершения вычислений на каждой стадии. При этом ни одно условие корректности не нарушается, следовательно, допускается формирование пакета, в котором содержатся новые состояния клеток из обрабатываемого на этой стадии независимого подмножества. Объем V пакета равен $c \cdot P_{\mathrm{dom}}/r$, где P_{dom} — периметр домена, $r = |B(m)|$. С учетом того, что стадии обрабатываются последовательно, условие эффективности распараллеливания (32) принимает вид

$$\tau \cdot |\mathrm{Dom}| > r \cdot t_{ex}. \tag{37}$$

Из (37) следует, что чем больше размер блока r, тем больше время обменов. Следовательно, блок надо выбирать минимального размера, при котором удовлетворяются условия (19) и (33).

Пример 4. Процесс преобразования химической энергии в электрическую в водородном топливном элементе происходит следующим образом. Водород поступает на пористый графитовый анод, активированный катализатором (платиной). Попадая на платину, атомы водорода разлагаются на электроны и протоны:

$$\mathtt{H}_2 \rightarrow 2\mathtt{H}^+ + 2e^-. \tag{38}$$

Протоны направляются к катоду, где, соединяясь с кислородом, образуют воду. Электроны поступают в электрическую цепь, создавая ток.

Диссоциация водорода на аноде определяет производительность топливного элемента, поэтому изучается особенно подробно. Моделирование его позволяет определить зависимость получаемого тока от парциального давления поступающего водорода и от структуры платинового покрытия в порах графитового анода.

Процесс моделируется АКА $\aleph_a = \langle A, M, \Theta \rangle$. В нём алфавит состояний есть $A = \{\varnothing, H, e\}$, где $\varnothing$ означает, что клетка свободна, H — клетка содержит атом водорода, e — клетка содержит e электронов, $e \in \{0, 1, \ldots\}$. Множество имен клеток есть $M = \{(i,j) : i,j \in 0, 1 \ldots, N\} \cup \{E, m_s, m_p\}$ и соответствует клеткам на поверхности пор анода; E — счетчик электронов; m_s, m_p — контекстные клетки, предикаты в которых соответствуют вероятности p_s попадания атомов водорода на платину, равной доле активированной поверхности, и вероятности выполнения диссоциации, равной парциальному давлению водорода p_{H}. Множество Θ содержит пять основных локальных операторов $\{\theta_e, \theta_H(k) : k = 1, 2, 3, 4\}$ и два контекстных вероятностных оператора вида (10):

$$\begin{aligned} &\theta_e : && \{(H,(i,j)),(e,E)\} \rightarrow \{(\varnothing,(i,j)),(e+1,E)\}, \\ &\theta_H(k) : && \{(\varnothing,(i,j)),(\varnothing,\varphi_k(i,j))\} \xrightarrow{p_{\mathrm{H}},p_s} \{(H,(i,j)),(H,\varphi_k(i,j))\}, \quad k = 1,2,3,4, \end{aligned} \tag{39}$$

где $\varphi_k(i,j) \in T(S(i,j))$ — k-я клетка, смежная с (i,j); p_{H} — парциальное давление водорода; p_s — доля площади пор, активированной платиной.

Моделирование проводилось на клеточном массиве с размерами $|M| = 10000 \times 10000$ клеток, что соответствует площади поверхности анода $S_a = 10^{-8}$ мм^2 при доле платинового покрытия $p_s = 0{,}5$. Время выполнения 100 итераций составляло $t = 64{,}4$ с. Для того чтобы получить данные для площади диссоциации $S_a \simeq 1$ мкм^2, было выполнено моделирование на 36 процессорах с такими же размерами доменов. Для параллельной реализации АКА был преобразован в блочно-синхронный с блоками размера $|B(m)| = 3 \times 3$. Таким образом, итерация состояла из 9 асинхронных стадий с синхронной передачей данных (рис. 8).

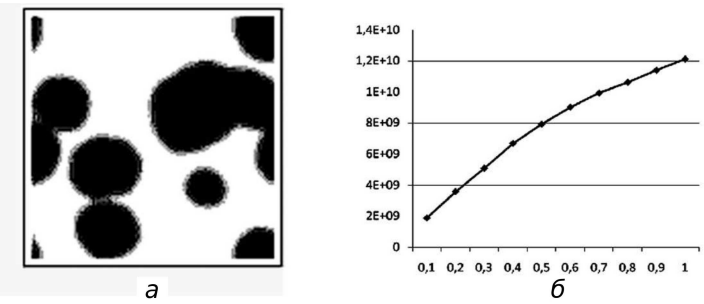

Рис. 8. Моделирование процесса диссоциации водорода на аноде топливного водородного элемента: *а* — платиновое покрытие поверхности поры при $p_s = 0{,}5$; *б* — зависимость количества полученных электронов от парциального давления поступающего водорода

Вычисления проводились на кластере MVS-1000 в ССКЦ СО РАН. Время вычисления 1000 итераций составило 1080 с, что соответствует эффективности распараллеливания $Q = 0{,}6$ и позволяет надеяться, что на 1000 процессорах за приемлемое время (несколько часов) можно моделировать получение тока с площадей около 1 см^2.

Заключение

Представленные в статье дискретные методы компьютерного моделирования позволяют исследовать физико-химические процессы «изнутри», наблюдая химические и фазовые преобразования, а также перемещения частиц на микро- и наноуровне. Методы основаны на клеточно-автоматной модели природных явлений, которая является альтернативой дифференциальным уравнениям, и позволяют существенно расширить возможности математического и компьютерного моделирования. Поскольку в современных технологиях наблюдается стремление создавать материалы, полученные на основе превращений микро- и наночастиц, и разработка современной технологии невозможна без компьютерного моделирования, то можно надеяться, что КА-моделирование станет востребованным не только в науке, но и в промышленности.

ЛИТЕРАТУРА

1. *Toffolli T.* Cellular Automata as an Alternative to (rather than Approximation of) Differential Equations in Modeling Physics // Physica D.1984. V. 10. P. 117–127.
2. *Wolfram S.* Statistical mechanics of Cellular automata // Rev. Mod. Phys. 1993. V. 55. P. 607–640.
3. *Boon J. P., Dab D., Kapral R., Lawniczak A.* Lattice-Gas Automata for Reactive Systems // Phys. Rep. 1996. V. 273. P. 55–147.

4. *Бандман О. Л.* Мелкозернистый параллелизм в вычислительной математике // Программирование. 2001. № 4. С. 1–18.
5. *Бандман О. Л.* Клеточно-автоматные модели пространственной динамики // Системная информатика: Сб. научн. тр. Новосибирск: Изд – во СО РАН. 2006. Вып. 10: Методы и модели современного программирования. С. 59–111.
6. *Bandman O.* Synchronous versus asynchronous cellular automata for simulating nano-systems kinetics // Bulletin of the Novosibirsk Computer Center. Series: Computer Science. Issue 27. 2006. P. 1–12.
7. *Rothman B. H., Zaleski S.* Lattice-Gas Cellular Automata. Simple Models of Complex Hydrodynamics. London: Cambridge Univ. Press, 1997. 320 p.
8. *Makeev A. G.* Coarse bifurcation analysis of kinetic Monte Carlo simulations: a lattice-gas model with lateral interactions // J. Chem. Phys. 2002. V. 117. No. 18. P. 8229–8240.
9. *Elokhin V. I., Latkin E. I., Matveev A. V., Gorodetskii V. V.* Application of Statistical Lattice Models to the Analysis of Oscillatory and Autowave Processes on the Reaction of Carbon Monoxide Oxidation over Platinum and Palladium Surfaces // Kinet. Catal. 2003. V. 44. No. 5. P. 672–700.
10. *Neizvestny I. G., Shwartz N. L., Yanovitskaya Z. Sh., and Zverev A. V.* 3D-model of epitaxial growth on porous {111} and {100} Si surfaces // Comp. Phys. Commun. 2002. V. 147. P. 272–275.
11. *Betz G., Husinsky W.* Surface erosion and film growth studied by a combined molecular dynamics and kinetic Monte Carlo code // Izvestia of Russian Academy of Sciences. Phys. Ser. 2002. V. 66. No. 4. P. 585–587.
12. *Kovalev E. V., Resnyanskii E. D., Elokhin V. I., et al.* Novel statistical lattice model for the supported nanoparticle. Features of the reaction performance influenced by the dynamically changed shape and surfaces morphology of the supported active particle // Phys.Chem.Chem.Phys. 2003. V. 5. P. 784–790.
13. *Alba E., Troya J. M.* Cellular Evolutionary Algorithms: evaluating the influence of ratio // Lect. Not. Comp. Sci. 2000. V. 197. P. 29–38.
14. *Malvanets A., Kapral R.* Microscopic model for Fitz-Nagumo dynamics // Phys. Rev. E. 1997. V. 55. No. 5. P. 5657–5670.
15. *Achasova S., Bandman O., Markova V., Piskunov S.*. Parallel Substitution Algorithm. Theory and Application. Singapore: World Scientific, 1994. 198 p.
16. *Bandman O.* Parallel Simulation of Asynchronous Cellular Automata Evolution // Lect. Not. Comp. Sci. V. 4173. 2006. P. 41–48.
17. *Бандман О. Л.* Параллельная реализация клеточно-автоматных алгоритмов моделирования пространственной динамики // Сибирский журн. вычислительной математики. 2007. № 4. С. 45–361.
18. *Wolfram S.* A new kind of science Champaign, Ill., USA: Wolfram Media Inc, 2002. 2000 p.
19. *Toffolli T., Margolus N.* Cellular Automata Machine. USA: MIT Press. 1987, 284 p.
20. *Vichniac G.* Simulating Physics by Cellular Automata // Phys. D. 1984. V. 10. P. 86–112.
21. *Bandman O.* Comparative Study of Cellular automata Diffusion Models // Lect. Not. Comp. Sci. 1999. V. 1662. P. 395–399.
22. *Бандман О. Л.* Методы композиции клеточных автоматов для моделирования пространственной динамики // Вестник Томского госуниверситета. 2002. № 9(1). С. 188–192.
23. *Ziff R. M., Gulari E., Bershad Y.* Kinetic phase transitions in irreversible surface-reaction model // Phys. Rev. Lett. 1986. V. 56. P. 553–2558.

DOI 10.17223/20710410/5/6

УДК 004.007.519-7

КЛЕТОЧНО-АВТОМАТНАЯ МОДЕЛЬ ФОРМИРОВАНИЯ ПОРОШКОВОЙ СТРУИ[1]

Ю. Г. Медведев

Институт вычислительной математики и математической геофизики СО РАН, г. Новосибирск, Россия

E-mail: medvedev@ssd.sscc.ru

При взрыве зарядов возникают течения, в которых происходят структурные превращения входящих в облицовку мелкодисперсных порошков. Получившиеся в результате продукты синтеза в последнее время активно изучаются. Их экспериментальное исследование достаточно дорого и требует сложных установок, тщательного подбора взрывчатого вещества и состава порошков. Поэтому численное моделирование этих процессов является актуальной задачей. Первой и самой важной частью модели является струя, несущая в себе участвующие в синтезе порошки. В работе предпринимается попытка промоделировать формирование такой струи клеточным автоматом. То, что эта модель дискретна, обуславливает эффективность ее программной реализации как в последовательном, так и в параллельном вариантах и позволяет минимизировать использование машинного времени. В работе представлена новая модель газового потока, несущего порошковые составляющие. Описаны результаты компьютерного моделирования с однокомпонентной облицовкой. Произведено сравнение со струей без порошка.

Ключевые слова: *клеточный автомат, поток газа, порошковая струя.*

Введение

В большинстве классических клеточно-автоматных моделей потоков используется один тип частиц — частицы газа [1]. Двумерная модель, ставшая основой для множества новых моделей, называется FHP (Frish, Hasslacher, Pomeau) [2]. Каждая ее клетка имеет форму шестиугольника и, в силу идемпотентности отношения соседства, имеет семь соседей. В качестве внутренних состояний клеток модели используются булевы векторы. Состояние клетки — это набор модельных частиц газа с единичной массой. Частицы дискретно перемещаются из клетки в клетку через равные интервалы времени. Каждая частица либо имеет вектор скорости с единичным модулем, направленный в сторону одной из шести нетождественных соседних клеток, либо имеет нулевую скорость. В клетке одновременно не могут находиться две или более частицы с одинаковыми скоростями. Каждый из семи разрядов вектора состояния клетки указывает на наличие или отсутствие в этой клетке частицы с соответствующей ему скоростью. Следовательно, одновременно в клетке могут находиться от нуля до семи частиц. Для расширения возможностей модели FHP была создана многочастичная модель FHP-MP (multi-particle) [3]. Она допускает одновременное присутствие в клетке более одной частицы с одинаковой скоростью. Эта модель позволяет работать в более широком диапазоне давлений, так как максимальное количество частиц в клетке ограничено только программной реализацией. Кроме того, она более пригодна для создания композиций

[1]Работа выполнена при поддержке интеграционного проекта СО РАН №32, 2009 г. и проекта 2.6 программы Президиума РАН, 2009 г.

с другими клеточно-автоматными моделями [4]. Построению и исследованию одной из таких композиций и посвящена эта работа. Композиция построена для моделирования газовой струи, несущей в себе порошок, путем добавления в модель FHP-MP частиц порошка и задания новых правил поведения. Новая композиционная модель с двумя типами частиц — газ и порошок — названа FHP-GP (gas-powder).

В статье описана новая клеточно-автоматная модель порошковой струи FHP-GP, приведены результаты вычислительных экспериментов по моделированию порожденной взрывом струи газа, несущей порошок, произведено сравнение с результатами моделирования струи газа без порошка, полученными в результате вычислительных экспериментов с использованием модели FHP-MP.

1. Описание модели FHP-GP

Под клеточным автоматом модели FHP-GP будем понимать тройку объектов $\langle \mathrm{K}, N, \Theta \rangle$, где $\mathrm{K} = \{c_1, c_2, \ldots, c_i, \ldots\}$ — множество клеток, заданное их координатами в некотором дискретном пространстве. Каждая клетка $c \in \mathrm{K}$ характеризуется состоянием $s(c)$ и координатами $x(c)$ и $y(c)$ на Декартовой плоскости. Состояние $s(c)$ зависит от дискретного времени t, а координаты остаются неизменными. Следовательно, между любыми двумя клетками $c_1 \in \mathrm{K}$ и $c_2 \in \mathrm{K}$ с легкостью можно подсчитать расстояние $d(c_1, c_2)$. В модели FHP-GP состояние $s(c)$ — это не один вектор, как у моделей-предшественников, а два: для газовых частиц целочисленный, как в FHP-MP, и для частиц порошка булев, как в FHP. Такая композиция является новой клеточно-автоматной моделью.

Для каждой клетки $c \in \mathrm{K}$ определено упорядоченное множество из семи элементов $N(c) = \{n_i(c) : n_0(c) = c, n_i(c) \in \mathrm{K} \;\&\; d(c, n_i(c)) = 1, i = 1, 2, \ldots, 6\}$. Входящие в него клетки $n_i(c)$ находятся в отношении соседства с клеткой c и называются ее соседними клетками или соседями. Таким образом, структура множества клеток K клеточного автомата представляется неориентированным графом, в котором вершинами являются клетки, а ребрами — отношение соседства. Этот граф имеет регулярную структуру и степени вершин, равные семи.

Состояние s клетки $c \in \mathrm{K}$ есть множество $s\,(c) = \left\{\mathbf{s}^G\,(c)\,, \mathbf{s}^P\,(c)\right\}$, состоящее из двух векторов. Первый из них $\mathbf{s}^G\,(c)$ имеет целочисленные компоненты $s_i^G\,(c)\,, i = 0, 1, \ldots, 6$, указывающие на количество частиц газа в клетке c с единичной массой и единичным вектором скорости $\mathbf{e}_i\,(c)$, направленным в сторону соседа $n_i(c)$. Эти частицы несут импульс $\mathbf{p}_i^G\,(c) = s_i^G\,(c)\,\mathbf{e}_i\,(c)$. Второй вектор $\mathbf{s}^P\,(c) = \left(s_0^P\,(c)\,, s_1^P\,(c)\,, \ldots, s_6^P\,(c)\right)$ является булевым, его компоненты определяют наличие или отсутствие в клетке c частиц порошка с вектором скорости $\mathbf{e}_i\,(c)$ и импульсом $\mathbf{p}_i^P\,(c) = M_P s_i^P\,(c)\,\mathbf{e}_i\,(c)$, где $i = 0, 1, \ldots, 6$, а M_P — масса одной частицы порошка. В направлении каждого из соседей $n_i(c)$ может быть ориентировано не более одного вектора скорости. Множество состояний $s\,(c)$ всех клеток $c \in \mathrm{K}$ в один и тот же момент времени t называется глобальным состоянием $\Omega(t) = \{s(c_1), s(c_2), \ldots, s(c_i), \ldots\}$ клеточного автомата.

На рис. 1 изображена клетка c, единичные векторы $\mathbf{e}_i\,(c)$ находящихся в ней частиц газа и порошка и ее соседи $n_i(c), i = 0, 1, \ldots, 6$. Вектор скорости $\mathbf{e}_i\,(c)$ либо направлен в сторону одной из соседних клеток $n_i(c)$ (при $i = 1, \ldots, 6$), либо равен нулю (при $i = 0$). Масса частиц, движущихся в направлении $\mathbf{e}_i\,(c)$, равна $m_i\,(c) = s_i^G\,(c) + M_P s_i^P\,(c)$.

Компоненты $s_0^G\,(c)\,, s_1^G\,(c)\,, \ldots, s_6^G\,(c)$ газовой составляющей вектора состояния s^G клетки c принимают целочисленные значения. Суммарная масса частиц газа в клетке c

равна

$$m^G(c) = \sum_{i=0}^{6} s_i^G(c), \tag{1}$$

где s_i^G — i-й компонент вектора состояний s^G. Масса частицы порошка равна M_P, поэтому масса всех частиц порошка в клетке c равна

$$m^P(c) = M_P \sum_{i=0}^{6} s_i^P(c). \tag{2}$$

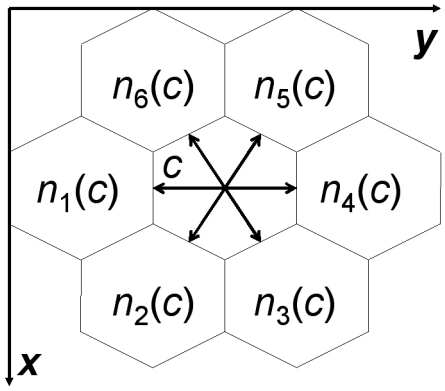

Рис. 1. Векторы скорости частиц, нумерация соседей

Общая масса частиц газа и порошка в клетке c равна

$$m(c) = m^G(c) + m^P(c). \tag{3}$$

Импульс частиц, направленных к соседу $n_i(c)$, $\mathbf{p}_i(c) = \mathbf{p}_i^G(c) + \mathbf{p}_i^P(c)$ в клетке $c \in \mathrm{K}$, есть сумма импульсов газа $\mathbf{p}_i^G(c)$ и порошка $\mathbf{p}_i^P(c)$. Суммарный по всем направлениям импульс равен

$$\mathbf{p} = \sum_{i=1}^{6} \mathbf{p}_i. \tag{4}$$

Из (4), учитывая рис. 1, легко подсчитать проекции p_x и p_y импульса $\mathbf{p}(c)$ на Декартовы оси Ox и Oy:

$$p_x = \frac{\sqrt{3}}{2}\left(|\mathbf{p}_2| + |\mathbf{p}_3| - |\mathbf{p}_5| - |\mathbf{p}_6|\right), \tag{5}$$

$$p_y = |\mathbf{p}_4| - |\mathbf{p}_1| + \frac{1}{2}\left(|\mathbf{p}_3| + |\mathbf{p}_5| - |\mathbf{p}_2| - |\mathbf{p}_6|\right). \tag{6}$$

Определено разбиение клеток $c \in \mathrm{K}$ на типы. Клетками среды $c_{\text{ср}} \in \mathrm{K}_{\text{ср}}$ называются клетки, в которых выполняются законы сохранения массы и импульса. Клетки стенок $c_{\text{ст}} \in \mathrm{K}_{\text{ст}}$ — это клетки, в которых выполняется закон сохранения массы, но может нарушаться закон сохранения импульса. И наконец, источники $c_{\text{ист}} \in \mathrm{K}_{\text{ист}}$ — клетки, в которых могут нарушаться как закон сохранения массы, так и закон сохранения импульса. Множества клеток среды $\mathrm{K}_{\text{ср}}$, стенок $\mathrm{K}_{\text{ст}}$ и источников $\mathrm{K}_{\text{ист}}$ попарно

не пересекаются ($\mathrm{K_{ср}} \cap \mathrm{K_{ст}} = \varnothing, \mathrm{K_{ср}} \cap \mathrm{K_{ист}} = \varnothing, \mathrm{K_{ст}} \cap \mathrm{K_{ист}} = \varnothing$). Объединение этих множеств совпадает с множеством всех клеток автомата ($\mathrm{K_{ср}} \cup \mathrm{K_{ст}} \cup \mathrm{K_{ист}} = \mathrm{K}$). Поведение стенок и источников задает граничные условия клеточного автомата.

В модели FHP-GP используется клеточный автомат с синхронным режимом функционирования. На каждой итерации происходит смена состояний $s\,(t)$ всех клеток $c \in \mathrm{K}$ на состояния $s\,(t+1) = \theta\,(s\,(t))$, где $\theta\,(s\,(t)) \in \Theta$ — соответствующая функция переходов клетки c. Клеточный автомат при этом переходит из глобального состояния $\Omega(t)$ в новое глобальное состояние $\Omega(t+1)$. Для того чтобы модель адекватно отображала физический процесс, функция θ в клетках среды $c_{\mathrm{ср}} \in \mathrm{K_{ср}}$ должна удовлетворять законам сохранения массы газа

$$\sum_{c\in\mathrm{K}} \sum_{i=0}^{6} \theta\left(s_i^G\,(c_{\mathrm{ср}})\right) = \sum_{c\in\mathrm{K}} \sum_{i=0}^{6} s_i^G\,(c_{\mathrm{ср}}), \tag{7}$$

массы порошка

$$\sum_{c\in\mathrm{K}} \sum_{i=0}^{6} \theta\left(s_i^P\,(c_{\mathrm{ср}})\right) = \sum_{c\in\mathrm{K}} \sum_{i=0}^{6} s_i^P\,(c_{\mathrm{ср}}) \tag{8}$$

и общего импульса

$$\sum_{c\in\mathrm{K}} \sum_{i=1}^{6} \theta\,(\mathbf{p}_i\,(c_{\mathrm{ср}})) = \sum_{c\in\mathrm{K}} \sum_{i=1}^{6} \mathbf{p}_i\,(c_{\mathrm{ср}}). \tag{9}$$

Каждая итерация выполняется в две фазы: сдвиг и столкновение. Функция переходов θ состоит, таким образом, из суперпозиции функций θ_1 (сдвиг) и θ_2 (столкновение):

$$\theta\,(s) = \theta_2\,(\theta_1\,(s))\,. \tag{10}$$

В фазе сдвига в каждой клетке $c \in \mathrm{K}$ каждая частица, учтенная в компонентах $s_i^G\,(c)$ и $s_i^P\,(c)$, $i = 1, \ldots, 6$, векторов состояния $\mathbf{s}^G\,(c)$ и $\mathbf{s}^P\,(c)$ перемещается в соседнюю клетку $n_i(c)$, соответствующую ее вектору скорости $\mathbf{e}_i\,(c)$. Частицы, соответствующие компонентам $s_0^G\,(c)$ и $s_0^P\,(c)$, остаются в клетке c. Таким образом, i-е компоненты $s_i^G\,(c)$ и $s_i^P\,(c)$ векторов состояния $\mathbf{s}^G\,(c)$ и $\mathbf{s}^P\,(c)$ клетки c после сдвига принимают значения

$$\theta_1\left(s_i^G\,(c)\right) = \begin{cases} s_i^G\left(N_{((i+2) \bmod 6)+1}\,(c)\right) & \text{для } i = 1, 2, \ldots, 6, \\ s_i^G\,(c) & \text{для } i = 0; \end{cases} \tag{11}$$

$$\theta_1\left(s_i^P\,(c)\right) = \begin{cases} s_i^P\left(N_{((i+2) \bmod 6)+1}\,(c)\right) & \text{для } i = 1, 2, \ldots, 6, \\ s_i^P\,(c) & \text{для } i = 0. \end{cases} \tag{12}$$

Несмотря на то, что при сдвиге масса и импульс частиц в клетке изменяются, в пределах всего клеточного автомата они сохраняются, т. е. условия (7), (8) и (9) выполняются.

В фазе столкновения θ_2 происходит изменение направления движения частиц согласно некоторым правилам столкновения, не зависящим от состояний соседних клеток. В модели FHP-GP функция θ_2 вероятностная. Ниже описаны правила столкновения для клеток разных типов: среды, стенок и источников.

В клетках среды $c_{\mathrm{ср}} \in \mathrm{K_{ср}}$ результатом функции θ_2 равновероятно выбирается одно из возможных значений, при которых сохраняются массы газа и порошка и общий импульс:

$$\sum_{i=0}^{6} \theta_2 \left(s_i^G \left(c_{\text{ср}} \right) \right) = \sum_{i=0}^{6} s_i^G \left(c_{\text{ср}} \right), \; c_{\text{ср}} \in \text{К}; \tag{13}$$

$$\sum_{i=0}^{6} \theta_2 \left(s_i^P \left(c_{\text{ср}} \right) \right) = \sum_{i=0}^{6} s_i^P \left(c_{\text{ср}} \right), \; c_{\text{ср}} \in \text{К}; \tag{14}$$

$$\sum_{i=1}^{6} \theta_2 \left(\mathbf{p}_i \left(c_{\text{ср}} \right) \right) = \sum_{i=1}^{6} \mathbf{p}_i \left(c_{\text{ср}} \right), \; c_{\text{ср}} \in \text{К}. \tag{15}$$

При выполнении условий (13), (14) и (15) условия (7), (8) и (9) тем более выполняются.

В клетках $c_{\text{ст}} \in \text{К}_{\text{ст}}$, являющихся стенками, частицы «отражаются» в обратном направлении, нарушая при этом закон сохранения импульса:

$$\theta_2 \left(s_i^P \left(c_{\text{ст}} \right) \right) = \begin{cases} s^P_{((i+2) \bmod 6)+1} \left(c_{\text{ст}} \right) \text{ для } i = 1, 2, \ldots, 6, \\ s_i^P \left(c_{\text{ст}} \right) \text{ для } i = 0. \end{cases} \tag{16}$$

Из-за того, что количество частиц в клетке не меняется, условия (13), (14), а следовательно, и (7), (8) выполняются. Условие (15) может нарушаться, так как меняются направления векторов скорости частиц, что соответствует изменению импульса частиц в газовой среде при столкновении с твердыми стенками. Такое поведение частиц в клетках-стенках моделирует условие нулевой скорости потока на границах препятствий.

Каждая клетка-источник $c_{\text{ист}} \in \text{К}_{\text{ист}}$ по алгоритму, определяемому граничными условиями, генерирует частицы со всевозможными направлениями вектора скорости. Из клеток-источников можно создавать разные объекты, задавая различные способы введения газа в моделируемый объем. Например, установив такие клетки в пространстве в одну линию (как правило, у границы клеточного массива), можно получить источник равномерного потока частиц заданной плотности. Отдельно установленный источник будет моделировать форсунку. Естественно, при генерации новых частиц ни масса $m^G \left(c_{\text{ист}} \right)$, ни импульс $\mathbf{p}^G \left(c_{\text{ист}} \right)$ частиц газа не сохраняются.

При моделировании практический интерес представляют осредненные значения скорости $\langle \mathbf{u} \rangle$ и плотности газа $\left\langle \rho^G \right\rangle$ и порошка $\left\langle \rho^P \right\rangle$ по некоторой окрестности $Av(c_0)$, которая включает все клетки $c \in \text{К}$, удаленные от клетки c_0 не больше чем на некоторую величину r, называемую радиусом осреднения. Осредненная скорость газа вычисляется как сумма скоростей всех частиц в клетках c, попадающих в окрестность осреднения $Av(c_0)$, деленная на количество этих частиц, или, что то же самое, суммарный импульс, деленный на суммарную массу:

$$\langle \mathbf{u} \rangle \left(c_0 \right) = \frac{\sum\limits_{c \in Av(c_0)} \sum\limits_{i=0}^{6} \mathbf{p}_i \left(c \right)}{\sum\limits_{c \in Av(c_0)} \sum\limits_{i=0}^{6} m_i \left(c \right)}. \tag{17}$$

Осредненная плотность частиц $\left\langle \rho^G \right\rangle$ и $\left\langle \rho^P \right\rangle$ подсчитывается в той же окрестности $Av(c_0)$ следующим образом:

$$\left\langle \rho^G \right\rangle (c_0) = \frac{1}{|Av\,(c_0)|} \sum_{c \in Av(c_0)} \sum_{i=0}^{6} s_i^G\,(c); \tag{18}$$

$$\left\langle \rho^P \right\rangle (c_0) = \frac{1}{|Av\,(c_0)|} \sum_{c \in Av(c_0)} \sum_{i=0}^{6} s_i^P\,(c), \tag{19}$$

где $|Av(c_0)|$ — количество клеток, попадающих в окрестность осреднения $Av(c_0)$.

Осредненные значения скорости $\langle \mathbf{u} \rangle$ являются модельными значениями скорости газового потока, осредненные значения плотности $\left\langle \rho^P \right\rangle$ являются модельными значениями плотности порошка, а осредненные значения плотности $\left\langle \rho^G \right\rangle$ являются модельными значениями давления газа.

Следует отметить, что осредненные значения модельных скорости, плотности и давления соответствуют их физическим аналогам только в том случае, когда окрестность осреднения $Av(c_0)$ состоит исключительно из клеток среды $c \in \mathrm{K}$. В противном случае, для предотвращения искажения результата осреднения значения $\langle \mathbf{u} \rangle, \left\langle \rho^G \right\rangle$ и $\left\langle \rho^P \right\rangle$ полагают неопределенными для заданного радиуса осреднения r. Это условие не позволяет определять осредненные значения на расстоянии ближе, чем радиус осреднения r от границ (стенок и источников), в том числе и находящихся внутри моделируемого объекта. Чем ближе к стенке нужно подсчитать осредненные значения, тем меньше нужно выбирать радиус осреднения r, но это приводит к увеличению автоматного шума.

2. Результаты тестовых исследований модели

Для проверки свойств предложенной модели была создана ее программная реализация, позволяющая проводить вычислительные эксперименты, задавая различные начальные условия и параметры эксперимента. Код программ написан на языке Си.

Хранение состояния автомата осуществляется в двумерном массиве. Из-за синхронного режима функционирования приходится использовать два таких массива. Функция сдвига θ_1 использует в качестве аргумента значения из одного из них, а результат помещает в другой. Для записи результата массивы используются поочередно: один на четных, а другой на нечетных итерациях. Функция столкновения θ_2 записывает свой результат в тот же массив, в котором находятся ее аргументы, и применяется к каждому из двух массивов поочередно после того, как функция θ_1 запишет в него свой результат.

Программным ограничением является максимальное количество частиц в одной клетке, имеющих одинаковый вектор скорости $\mathbf{e}_i\,(c)$, равное 255. Это означает, что каждый из семи компонентов $s_i^G\,(c)$ вектора состояния хранится в одном байте памяти. Вектор состояния порошкового компонента $\mathbf{s}^P\,(c)$ целиком занимает один байт, интерпретируемый как булев вектор, семь из восьми разрядов которого отвечают за шесть направлений и одну частицу покоя, а восьмой разряд не используется. Еще один байт занимает информация о типе клетки. Таким образом, состояние каждой клетки занимает 18 байт памяти с учетом дублирования во втором массиве. К примеру, клеточный автомат, эксперименты с которым описаны ниже, имеющий размер 200×400 клеток, требует около полутора мегабайт памяти.

Реализация функции сдвига θ_1 тривиальна. Она использует второй массив для записи своих значений, поэтому при последовательном обходе клеток автомата исходные значения, хранящиеся в первом массиве, не портятся. После того как эта функ-

ция отработала, указатели на массивы обмениваются своими значениями, выполняется функция θ_2, а затем следующая итерация.

Функция θ_2 последовательно перебирает все клетки автомата по следующим правилам. Если клетка имеет тип «стенка», то векторы скорости всех частиц в ней меняются на антипараллельные. Клетки-источники генерируют новые частицы газа с заданной вероятностью. Клетки среды производят столкновение по следующему алгоритму. Вначале вычисляются концентрация частиц в клетке и проекции их импульсов на декартовы оси. Затем полным перебором подсчитывается количество состояний, сохраняющих массу газа, массу порошка и общий импульс, обусловленные выражениями (13), (14) и (15). После этого при помощи генератора случайных чисел, описанного в [5], равновероятно выбирается одно из этих состояний; его и полагают результатом функции θ_2. Из-за полного перебора столкновение является самой затратной вычислительной функцией, несмотря на ряд оптимизаций, дающих выигрыш в скорости на несколько десятичных порядков.

Ниже описаны вычислительные эксперименты, проведенные с моделью FHP-GP. В первом из них используется только газовая составляющая модели и не используются частицы порошка. Такие начальные условия редуцируют модель до FHP-MP. Во втором эксперименте используются частицы как газа, так и порошка. Размеры автомата, топология стенок и другие параметры в обоих экспериментах одинаковы, что дает возможность сравнить их результаты.

Клеточный автомат, использующийся в этих вычислительных экспериментах, имеет размеры 200×400 клеток (вдоль декартовых осей Ox и Oy соответственно). Периметр с координатами в интервалах [(1, 1), (1, 200)], [(1, 1), (400, 1)], [(1, 200), (400, 200)] и [(400, 1), (400, 200)] выстроен из клеток-стенок. Также из стенок с координатами [(100, 1), (100, 80)] и [(100, 121), (100, 200)] построено сопло. Остальные — клетки среды. Так как моделируемый объем замкнут, клетки-источники не используются.

Результаты моделирования приведены на рис. 2. В первом эксперименте это поля давления газа и скорости потока, а во втором — поля давления газа и концентрации порошка. Начальное состояние автомата дано в верхней строке, в последующих строках — после некоторого количества итераций. Стенки изображены черными линиями. Так как клетки шестиугольные, вертикальные стенки имеют форму прямых линий, а горизонтальные — зигзагообразных ломаных. Для получения полей давления и концентрации произведено осреднение плотности газа $\langle \rho^G \rangle$ и порошка $\langle \rho^P \rangle$ с радиусом $r = 1$. Более темным цветом изображены участки с бóльшими давлением газа и концентрацией частиц. Для получения поля скорости произведено осреднение скорости потока $\langle \mathbf{u} \rangle$ с радиусом $r = 3$. Длина векторов пропорциональна скорости, а их направление совпадает с направлением потока.

В начале обоих экспериментов в клетках среды находятся частицы газа со средней плотностью $\langle \rho_i^G \rangle = 3$ частицы на каждое направление, или 21 частица в клетке. Для моделирования взрыва в левой части камеры расположены две области газа с плотностью $\langle \rho_0^G \rangle = 60$ частиц покоя в клетке. Они имеют форму наклонных полос толщиной по 40 клеток и изображены в столбцах «Давление газа» черным цветом. Кроме того, во втором эксперименте внутри полос газа высокого давления в столбце «Концентрация порошка» имеются две наклонные полосы порошка с плотностью $\langle \rho_0^P \rangle = 1$ частица покоя в клетке. Эти полосы имеют толщину по 10 клеток. Масса частицы порошка $M_P = 20$.

Результаты первого эксперимента после 70, 220 и 390 итераций показывают, что в левой части камеры от участков высокого давления навстречу друг другу идут

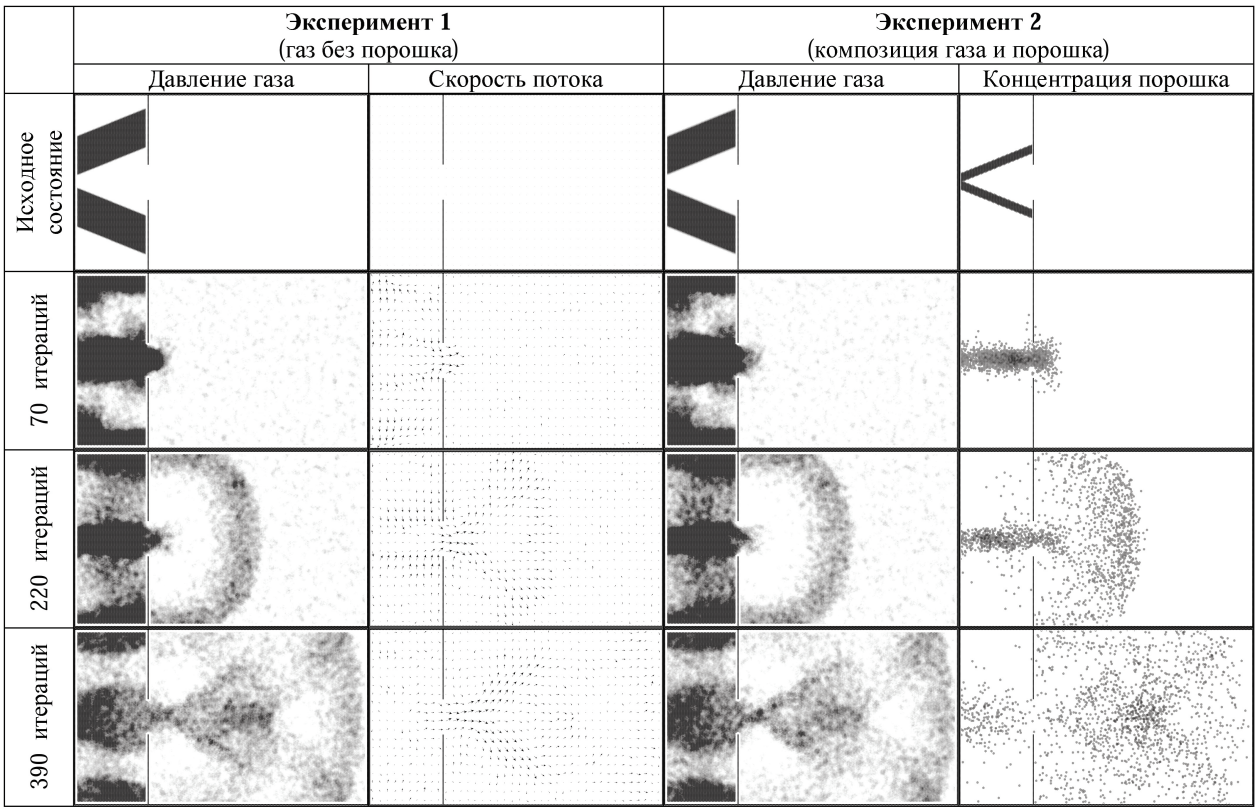

Рис. 2. Формирование струи. Результаты вычислительных экспериментов

ударные волны, которые сталкиваются, и газовый поток со сферическим фронтом распространяется от сопла в правую часть камеры. После того как в левой части камеры другие два фронта, удалявшиеся друг от друга, отразились от верхней и нижней стенок и столкнулись, из сопла вправо вылетел еще один фронт, который смешался с отраженным от верхней и нижней стенок первым фронтом. Далее происходит отражение первого фронта от правой стенки и перемешивание газа, что не представляет интереса и не отражено на рисунке.

Во втором эксперименте поведение газа качественно не отличается от первого, так как импульс частиц газа в полосах высокого давления на порядок превосходит импульс частиц порошка. В исходном состоянии порошок покоится. Затем он, вытолкнутый взрывом, летит вправо, образуя облако со сферическим фронтом. В модели реализовано отталкивание частиц порошка от стенок, которые в проведенном эксперименте располагались довольно близко от формирующейся струи, поэтому в конце процесса наблюдается перемешивание порошка. Но, несмотря на это, форма струи качественно совпадает с зафиксированной в натурном эксперименте при определенных условиях [6].

Заключение

Проведенные эксперименты показали, что новая клеточно-автоматная модель FHP-GP способна описать процессы формирования порошковой струи. Конечно же, в статье показан только принцип применения модели. Для ее практического использования следует производить привязку модельных величин к физическим. Это необходимо делать каждый раз при настройке модели на новые размеры камеры, плот-

ность и степень намола порошка, температуру газа, мощность взрывчатки и другие параметры. Кроме того, в модели не учитывается гравитация и ряд других физических явлений, но при необходимости их довольно легко учесть. Также легко перейти к трехмерному варианту предложенной модели, используя клеточный автомат с пространственной структурой в форме ромбического додекаэдра, подобно тому, как был осуществлен переход от двумерной модели FHP к трехмерной RD [7]. Конечно, получившаяся модель RD-GP будет требовать очень много вычислительных ресурсов, но если закон Мура будет продолжать работать, то в скором будущем суперкомпьютеры смогут покрыть эти потребности.

ЛИТЕРАТУРА

1. *Бандман О. Л.* Клеточно-автоматные модели пространственной динамики // Системная информатика. 2006. № 10. С. 59–113.
2. *Frisch U., Hasslacher B., and Pomeau Y.* Lattice-Gas automata for Navier-Stokes equations // Phys. Rev. Lett. 1986. No. 56. P. 1505.
3. *Медведев Ю. Г.* Расширение клеточно-автоматной модели потока FHP-I до многочастичного варианта FHP-MP // Новые информационные технологии в исследовании сложных структур. Томск, 2008. С. 73.
4. *Bandman O. L.* Cellular Automata composition technuques for spatial dynamics simulation // Bull. Nov. Comp. Center, Comp. Science. 2008. V. 27. P. 1–39.
5. *Kalgin K. V.* Acceleration of linear congruental generators // Bull. Nov. Comp. Center, Comp. Science. 2008. V. 27. P. 51–53.
6. *Кабулашвили В. Г., Тришин Ю. А.* Струйные течения при взрывном обжатии пористых облицовок и их применение // Взрыв, удар, защита: Сб. научных трудов. Новосибирск, 1988. № 18. С. 149.
7. *Медведев Ю. Г.* Трехмерная клеточно-автоматная модель потока вязкой жидкости // Автометрия. 2003. Т. 39. № 3. С. 43–50.

МАТЕМАТИЧЕСКИЕ МЕТОДЫ КРИПТОГРАФИИ

DOI 10.17223/20710410/5/7

УДК 519.7

О ПОТОЧНЫХ И АВТОМАТНЫХ ШИФРСИСТЕМАХ С СИММЕТРИЧНЫМ КЛЮЧОМ[1]

И. В. Панкратов

Томский государственный университет, г. Томск, Россия

E-mail: ivan.pankr@gmail.com

Рассматриваются симметричные поточные и автоматные шифрсистемы. Показывается неотличимость поточных шифрсистем, у которых неотличимы генераторы ключевого потока; определяются понятия последовательностного шифра, автоматной и самосинхронизирующейся с задержкой автоматной шифрсистем; показывается инъективность функции выходов автомата шифрования в автоматной шифрсистеме при любой фиксации состояния автомата и ключа шифрсистемы; устанавливается функциональная эквивалентность классов поточных и автоматных шифрсистем, а именно: для каждой системы любого из этих классов существует система в другом классе, которая задаёт то же семейство последовательностных шифров, что и первая; как альтернатива конструктивному определению понятия поточной самосинхронизирующейся шифрсистемы даётся дескриптивное определение этого понятия и устанавливается равносильность обоих определений; показывается, что регистровыми шифрсистемами исчерпываются все автоматные самосинхронизирующиеся системы с сильносвязными проекциями автомата шифрования.

Ключевые слова: *поточная шифрсистема, автоматная шифрсистема, последовательностный шифр, самосинхронизирующаяся шифрсистема, генератор ключевого потока, регистровая шифрсистема.*

Введение

В данной работе все рассматриваемые шифрсистемы и шифры предполагаются симметричными, а автоматы — конечными. Продолжаются исследования, начатые автором в [1], где были определены понятия поточной шифрсистемы и самосинхронизирующихся с задержкой поточной и регистровой шифрсистем и было показано, что последними исчерпываются все поточные самосинхронизирующиеся системы, у которых проекции генератора ключевого потока являются сильносвязными автоматами. Показывается неотличимость поточных шифрсистем, имеющих неотличимые генераторы ключевого потока (теорема 1); определяются понятия последовательностного шифра, автоматной и самосинхронизирующейся с задержкой автоматной шифрсистем; доказывается, что функция выходов автомата шифрования в автоматной шифрсистеме инъективна при любой фиксации состояния автомата и ключа шифрсистемы (теорема 2); устанавливается функциональная эквивалентность классов поточных и автоматных шифрсистем (теорема 3), а именно: для каждой системы любого из этих

[1]Работа выполнена в рамках реализации ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009–2013 годы. Результаты работы докладывались на Международной конференции с элементами научной школы для молодёжи, г. Омск, 7–12 сентября 2009 г.

классов существует система в другом классе, которая задаёт то же семейство последовательностных шифров, что и первая; как альтернатива конструктивному определению в [1] понятия поточной самосинхронизирующейся шифрсистемы (в терминах самосинхронизирующихся проекций генератора ключевого потока) здесь даётся ещё и дескриптивное определение этого понятия (в терминах задаваемых самосинхронизирующихся последовательностных шифров) и показывается равносильность обоих определений (теорема 4); наконец, устанавливается, что регистровыми шифрсистемами исчерпываются также и автоматные самосинхронизирующиеся системы с сильносвязными проекциями автомата шифрования (теорема 6). Формулировку теорем 3, 4, 6 (без доказательства) можно найти в тезисах работы [2].

Все теоретико-автоматные понятия и обозначения, используемые в статье, заимствуются из [3]. Конечный автомат (Мили) с входным алфавитом X, выходным алфавитом Y, множеством состояний Q и функциями переходов и выходов $\psi\colon Q \times X \to Q$ и $\varphi\colon Q \times X \to Y$ соответственно обозначается набором $A = <X, Q, Y, \psi, \varphi>$. В нём функции ψ и φ, определённые на $Q \times X$, считаются определёнными (индукцией по длине входного слова) и на $Q \times X^*$. Последовательность выходных символов, которую автомат вырабатывает из начального состояния $q \in Q$ под действием входного слова α, обозначается $\bar{\varphi}(q, \alpha)$ или $\bar{\varphi}_q(\alpha)$. В автомате Мура A, где функция $\varphi(q, x)$ зависит от аргумента x фиктивно и является фактически функцией из Q в Y, иногда вместо $\varphi(q, x)$ пишем $\varphi(q)$.

В автомате $A = <X \times K, Q, Y, \psi, \varphi>$ с двумя входами X называется алфавитом сообщений, K — множеством, или пространством, ключей и X^* — множеством сообщений. В нём для произвольного ключа $k \in K$ вводятся функции $\psi_k(q, x) = \psi(q, (x, k))$ и $\varphi_k(q, x) = \varphi(q, (x, k))$ соответственно переходов и выходов на ключе k и автомат $A_k = <X, Q, Y, \psi_k, \varphi_k>$, называемый *проекцией автомата A на вход X при данном ключе $k \in K$*. Наконец, понятие последовательностного шифра определяется следующим образом.

Определение 1. *Последовательностным шифром* называется набор из пяти объектов $S = <X, Y, K, f, g>$, где X, Y и K — конечные множества, называемые соответственно *входным алфавитом*, *выходным алфавитом* и *ключевым пространством*, f и g — функции, $f\colon X^* \times K \to Y^*$, $g\colon Y^* \times K \to X^*$, называемые функциями соответственно *шифрования* и *расшифрования* и связанные отношением обратимости

$$\forall \alpha \in X^* \forall \beta \in Y^* \forall k \in K [f(\alpha, k) = \beta \Rightarrow g(\beta, k) = \alpha].$$

Последовательностные шифры с общей функцией шифрования считаются равными.

1. Поточные шифрсистемы

Определение 2 [1]. Назовём *поточной шифрсистемой* набор из шести объектов $\Sigma_S = <X, Y, K, G, e, d>$, где $G = <Y \times K, Q, \Gamma, \psi, \varphi>$ — конечный автомат, называемый *генератором ключевого потока* (*ГКП*), такой, что любая его проекция G_k есть автомат Мура, $e\colon X \times \Gamma \to Y$, $d\colon Y \times \Gamma \to X$ — правила шифрования и расшифрования одного символа открытого текста (сообщения) с помощью одного символа ключевого потока (гаммы), связанные отношением обратимости

$$\forall \gamma \in \Gamma \forall x \in X \forall y \in Y [e(x, \gamma) = y \Rightarrow d(y, \gamma) = x].$$

Введём для Σ_S функции $\bar{e}\colon Q \times X^* \times K \to Y^*$ и $\bar{d}\colon Q \times Y^* \times K \to X^*$, называемые *алгоритмами* соответственно *шифрования* и *расшифрования сообщений*, которые по

ключу k и начальному состоянию q_0 определяются по следующим формулам:

$$\bar{e}(q_0, \Lambda, k) = \Lambda,$$

$$\bar{e}(q_0, x_0x_1 \dots x_{n-1}, k) = y_0y_1 \dots y_{n-1}, \text{ где } \begin{cases} \gamma_t = \varphi_k(q_t), \\ y_t = e(x_t, \gamma_t), \\ q_{t+1} = \psi_k(q_t, y_t), \end{cases} \quad t = 0, 1, \dots, n-1,$$

$$\bar{d}(q_0, \Lambda, k) = \Lambda,$$

$$\bar{d}(q_0, y_0y_1 \dots y_{n-1}, k) = x_0x_1 \dots x_{n-1}, \text{ где } \begin{cases} \gamma_t = \varphi_k(q_t), \\ x_t = d(y_t, \gamma_t), \\ q_{t+1} = \psi_k(q_t, y_t), \end{cases} \quad t = 0, 1, \dots, n-1.$$

Эти уравнения проиллюстрированы схемами на рис. 1 и 2, где D обозначает элемент задержки на единицу дискретного времени. В них последовательности $q_0q_1 \dots$ и $\gamma_0\gamma_1 \dots$ называются соответственно последовательностью состояний и ключевым потоком, или гаммой, вырабатываемыми в ГКП G из начального состояния q_0 при заданном ключе k под воздействием входного слова $y_0y_1 \dots$

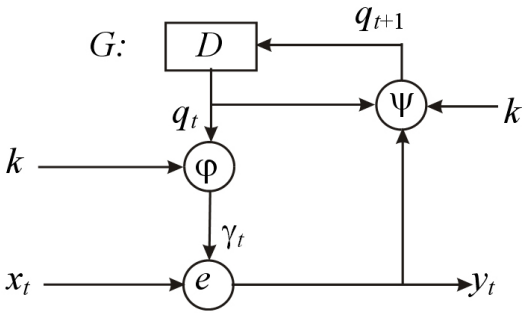

Рис. 1. Схема алгоритма шифрования в поточной шифрсистеме

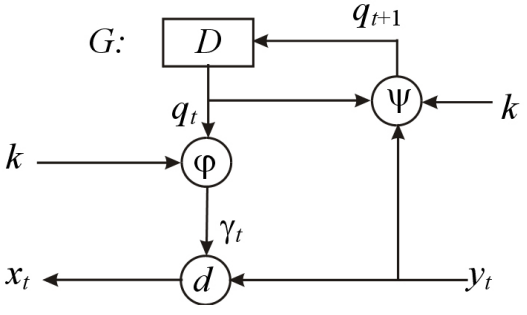

Рис. 2. Схема алгоритма расшифрования в поточной шифрсистеме

По определению функций $\bar{e}$ и $\bar{d}$ легко проверить, что при любых G, $q \in Q$, e и d, где e и d связаны отношением обратимости, пятерка $S_q(\Sigma_S) = <X, Y, K, \bar{e}_q, \bar{d}_q>$, где $\bar{e}_q \colon X^* \times K \to Y^*$, $\bar{d}_q \colon Y^* \times K \to X^*$ и $\bar{e}_q(\alpha, k) = e(q, \alpha, k)$, $\bar{d}_q(\beta, k) = d(q, \beta, k)$ для всех $k \in K$, $\alpha \in X^*$ и $\beta \in Y^*$, будет последовательностным шифром. Множество

$S(\Sigma_S) = \{S_q(\Sigma_S) \mid q \in Q\}$ всех таких шифров называется далее *семейством последовательностных шифров, задаваемых шифрсистемой* Σ_S.

Возможны различные варианты использования поточной шифрсистемы в зависимости от выбора начального состояния q_0 её ГКП: 1) q_0 фиксировано и заранее известно; 2) q_0 вычисляется открытым способом по ключу; 3) q_0 вычисляется как некая случайная величина и каким-то образом (открыто или закрыто) передается корреспонденту. В [1] показан вариант вероятностной самосинхронизирующейся поточной шифрсистемы, в котором начальное состояние генерируется случайно и в открытом виде передается корреспонденту вместе с криптограммой. В общем случае начальное состояние является вероятностной функцией от ключа.

Поточную шифрсистему $\Sigma_S = <X, Y, K, G, e, d>$ будем называть *неотличимой* [1] от поточной шифрсистемы $\Sigma'_S = <X, Y, K, G', e', d'>$, где $G = <Y \times K, Q, \Gamma, \psi, \varphi>$ и $G' = <Y \times K, Q', \Gamma', \psi', \varphi'>$, если выполняется следующее соотношение:

$$\forall k \in K \forall q \in Q \exists q' \in Q' \forall \alpha \in X^* [\bar{e}(q, \alpha, k) = \bar{e}'(q', \alpha, k)].$$

Отметим, что неотличимость двух поточных шифрсистем друг от друга не влечёт равенства семейств задаваемых ими последовательностных шифров. Причина в порядке кванторов. Содержательно неотличимость гласит, что для любого ключа и для любого начального состояния первой шифрсистемы найдется такое начальное состояние второй шифрсистемы, что соответствующие алгоритмы шифрования будут совпадать. Однако при этом возможно, что при разных ключах одному и тому же состоянию первой шифрсистемы будут соответствовать разные состояния второй шифрсистемы, то есть может вообще не быть таких начальных состояний разных шифрсистем, в которых шифрсистемы задают одинаковые последовательностные шифры.

Будем говорить, что ГКП $G = <Y \times K, Q, \Gamma, \psi, \varphi>$ *неотличим* от ГКП $G' = <Y \times K, Q', \Gamma, \psi', \varphi'>$, если для любого ключа $k \in K$ его проекция G_k на вход Y сильно неотличима [3] от аналогичной проекции G'_k, т. е. каждое состояние в G_k неотличимо от некоторого состояния в G'_k.

Теорема 1. Пусть $\Sigma_S = <X, Y, K, G, e, d>$ и $\Sigma'_S = <X, Y, K, G', e, d>$ — две шифрсистемы с разными ГКП G и G'. Тогда если G неотличим от G', то и Σ_S неотличима от Σ'_S.

Доказательство. Пусть $G = <Y \times K, Q, \Gamma, \psi, \varphi>$, $G' = <Y \times K, Q', \Gamma, \psi', \varphi'>$. Возьмём произвольные $k \in K$, $q_0 \in Q$, $\alpha = x_0 x_1 \dots x_{n-1} \in X^*$ и состояние $q'_0 \in Q'$, неотличимое от $q_0 \in Q$.

Будем иметь

$$\bar{e}(q_0, x_0 x_1 \dots x_{n-1}, k) = y_0 y_1 \dots y_{n-1}, \text{ где } \begin{cases} \gamma_t = \varphi_k(q_t), \\ y_t = e(x_t, \gamma_t), \\ q_{t+1} = \psi_k(q_t, y_t), \end{cases} \quad t = 0, 1, \dots, n-1,$$

$$\bar{e}'(q'_0, x_0 x_1 \dots x_{n-1}, k) = y'_0 y'_1 \dots y'_{n-1}, \text{ где } \begin{cases} \gamma'_t = \varphi'_k(q'_t), \\ y'_t = e(x_t, \gamma'_t), \\ q'_{t+1} = \psi'_k(q'_t, y'_t), \end{cases} \quad t = 0, 1, \dots, n-1.$$

Индукцией по t убедимся в равенстве $y_t = y'_t$ для каждого $t = 0, 1, \dots, n-1$.

База индукции: $t = 0$. Имеем: $\gamma_0 = \varphi_k(q_0) = \varphi'_k(q'_0) = \gamma'_0$, $y_0 = e(x_0, \gamma_0) = e(x_0, \gamma'_0) = y'_0$.

Предположение индукции. Предположим, что $y_0y_1\dots y_{t-1} = y'_0y'_1\dots y'_{t-1}$ для некоторого $t \geqslant 1$.

Шаг индукции. Ввиду неотличимости состояний $\psi_k(q_0, y_0y_1\dots y_{t-1})$ и $\psi'_k(q'_0, y_0y_1\dots y_{t-1})$, следующей из неотличимости q_0 и q'_0, можно записать $\gamma_t = \varphi_k(q_t) = \varphi_k(\psi_k(q_{t-1}, y_{t-1})) = \varphi_k(\psi_k(q_0, y_0y_1\dots y_{t-1})) = \varphi'_k(\psi'_k(q'_0, y_0y_1\dots y_{t-1})) = \varphi'_k(\psi'_k(q'_0, y'_0y'_1\dots y'_{t-1})) = \varphi'_k(\psi'_k(q'_{t-1}, y'_{t-1})) = \varphi'_k(q'_t) = \gamma'_t$ и $y_t = e(x_t, \gamma_t) = e(x_t, \gamma'_t) = y'_t$.

Индуктивное заключение: $y_0y_1\dots y_{n-1} = y'_0y'_1\dots y'_{n-1}$.

Следовательно, $\bar{e}(q_0, \alpha, k) = \bar{e}'(q'_0, \alpha, k)$.

Ввиду произвольности $k \in K$, $q_0 \in Q$ и $\alpha \in X^*$ теорема доказана. ■

Таким образом, установлено, что при замене в шифрсистеме одного ГКП на другой, неотличимый от него, получится шифрсистема, неотличимая от исходной.

2. Автоматные шифрсистемы

Определение 3. Назовём *автоматной шифрсистемой* пятерку объектов $\Sigma_A = <X, Y, K, E, D>$, где $E = <X \times K, Q, Y, \psi, \varphi>$ и $D = <Y \times K, Q^*, X, \psi^*, \varphi^*>$ суть конечные автоматы, называемые *автоматами шифрования* и *расшифрования* соответственно, если для любого состояния $q \in Q$ найдется такое состояние $q^* \in Q^*$, что пятерка $S_{qq^*}(\Sigma_A) = <X, Y, K, \bar{\varphi}_q, \bar{\varphi}^*_{q^*}>$ будет последовательностным шифром, то есть

$$\forall q \in Q \exists q^* \in Q^* \forall k \in K \forall \alpha \in X^* \forall \beta \in Y^* [\bar{\varphi}_q(\alpha, k) = \beta \Rightarrow \bar{\varphi}^*_{q^*}(\beta, k) = \alpha].$$

Определение 4. Множество $S(\Sigma_A) = \{S_{qq^*}(\Sigma_A) \mid q \in Q\}$ называется далее *семейством последовательностных шифров, задаваемых шифрсистемой* Σ_A.

Теорема 2. Функция выходов автомата шифрования автоматной шифрсистемы при фиксированных значениях состояния и ключа инъективна как функция одного аргумента — символа сообщений.

Доказательство. Пусть $\Sigma_A = <X, Y, K, E, D>$ есть автоматная шифрсистема с автоматами шифрования $E = <X \times K, Q, Y, \psi, \varphi>$ и расшифрования $D = <Y \times K, Q^*, X, \psi^*, \varphi^*>$. Докажем, что при любых фиксированных значениях состояния $q \in Q$ и ключа $k \in K$ функция $\varphi_{qk}(x) = \varphi(q, (x, k))$ инъективна.

Предположим противное: пусть для некоторых состояния $q \in Q$ и ключа $k \in K$

$$\exists x_1, x_2 \in X[(x_1 \neq x_2) \,\&\, (\varphi_{qk}(x_1) = \varphi_{qk}(x_2))].$$

Тогда по определению автоматной шифрсистемы Σ_A найдётся такое $q^* \in Q^*$, что для любых y_1 и y_2 в Y

$$\bar{\varphi}_q(x_1, k) = y_1 \Rightarrow \bar{\varphi}^*_{q^*}(y_1, k) = x_1,\ \bar{\varphi}_q(x_2, k) = y_2 \Rightarrow \bar{\varphi}^*_{q^*}(y_2, k) = x_2.$$

Здесь $\bar{\varphi}_q(x_1, k) = \varphi_{qk}(x_1)$ и $\bar{\varphi}_q(x_2, k) = \varphi_{qk}(x_2)$, поэтому $y_1 = y_2$ и, следовательно, $x_1 = x_2$, что не так. ■

Интересно отметить, что ещё 50 лет назад А. Д. Закревский [4] предложил использовать в качестве алгоритмов шифрования конечные автоматы с функцией выходов, инъективной в каждом состоянии. Теорема 2 говорит о том, что других автоматов, которые можно было бы использовать в этом качестве, не бывает.

3. Равносильность поточных и автоматных шифрсистем

Определение 5. Две шифрсистемы (поточные или автоматные) называются *эквивалентными*, если они задают одно и то же семейство последовательностных шифров (в предположении о равенстве шифров, отличающихся только функцией расшифрования). Два класса шифрсистем (поточных или автоматных) *равносильны*, если каждая шифрсистема любого из них эквивалентна некоторой шифрсистеме другого.

Теорема 3. Классы поточных и автоматных шифрсистем равносильны.

Доказательство. Докажем сначала, что каждая поточная шифрсистема эквивалентна некоторой автоматной шифрсистеме. Для этого возьмем произвольную поточную шифрсистему $\Sigma_S = <X, Y, K, G, e, d>$, где $G = <Y \times K, Q', \Gamma, \psi', \varphi'>$, и построим автоматную шифрсистему $\Sigma_A = <X, Y, K, E, D>$, где автоматы $E = <X \times K, Q, Y, \psi, \varphi>$ и $D = <Y \times K, Q^*, X, \psi^*, \varphi^*>$ определяются соотношениями

$$Q = Q^* = Q',$$

$$\psi(q,(x,k)) = \psi'(q, e(x, \varphi'(q,k)), k),\ \varphi(q,(x,k)) = e(x, \varphi'(q,k)),$$

$$\psi^*(q^*,(y,k)) = \psi'(q,y,k),\ \varphi^*(q^*,(y,k)) = d(y, \varphi'(q,k)).$$

Непосредственно проверяется, что

$$\forall q \in Q[(\bar{e}_q = \bar{\varphi}_q)\ \&\ (\bar{d}_q = \bar{\varphi}^*_q)].$$

Следовательно, $S_q(\Sigma_S) = S_{qq^*}(\Sigma_A)$ и $S(\Sigma_S) = S(\Sigma_A)$. Этим доказано, что поточная шифрсистема Σ_S эквивалентна автоматной шифрсистеме Σ_A.

Теперь докажем обратное, а именно: любая автоматная шифрсистема эквивалентна некоторой поточной шифрсистеме. Для этого возьмем произвольную автоматную шифрсистему $\Sigma_A = <X, Y, K, E, D>$, где $E = <X \times K, Q, Y, \psi, \varphi>$ и $D = <Y \times K, Q^*, X, \psi^*, \varphi^*>$, и построим поточную шифрсистему $\Sigma_S = <X, Y, K, G, e, d>$, где операции e, d и автомат $G = <Y \times K, Q', \Gamma, \psi', \varphi'>$ определяются соотношениями

$$Q' = Q,\ \Gamma = Q \times K,$$

$$e(x,\gamma) = e(x,(q,k)) = \varphi(q,(x,k)),\ d(y,\gamma) = d(y,(q,k)) = \varphi^{-1}_{qk}(y),$$

$$\psi'(q,(y,k)) = \psi(q,(d(y,(q,k)),k)) = \psi(q,(\varphi^{-1}_{qk}(y),k)),\ \varphi'(q,(y,k)) = (q,k),$$

где функция $\varphi^{-1}_{qk}(y)$ является обратной к функции $\varphi_{qk}(x)$. Она существует в силу инъективности последней по теореме 2.

Непосредственно проверяется, что

$$\forall \gamma \in \Gamma \forall x \in X \forall y \in Y[e(x,\gamma) = y \Rightarrow d(y,\gamma) = x],$$

$$\forall q \in Q[\bar{\varphi}_q = \bar{e}_q\ \&\ \bar{\varphi}^*_q = \bar{d}_q].$$

Следовательно, $S_{qq^*}(\Sigma_A) = S_q(\Sigma_S)$ и $S(\Sigma_A) = S(\Sigma_S)$. Этим доказано, что автоматная шифрсистема Σ_A эквивалентна поточной шифрсистеме Σ_S. ■

Основная польза от этой теоремы заключается в том, что она позволяет к шифрсистемам одного класса применять результаты, полученные для шифрсистем другого класса, либо сконструировать шифрсистему по любой из этих двух схем и рассматривать её с обеих точек зрения.

4. Самосинхронизирующиеся поточные шифрсистемы

Определение 6 [1]. Назовём автомат $A = <X, Q, Y, \psi, \varphi>$ *самосинхронизирующимся с задержкой* τ, если выполняется условие

$$\forall q \in Q \forall \alpha, \alpha' \in X^* \forall \tilde{\alpha} \in X^\tau \forall \xi \in X^* [\bar{\varphi}(\psi(q, \alpha\tilde{\alpha}), \xi) = \bar{\varphi}(\psi(q, \alpha'\tilde{\alpha}), \xi)].$$

Содержательно это определение можно объяснить так: «В каком бы состоянии q автомат A ни находился, какую бы последовательность (α или α') входных символов в него ни подали, его выходная последовательность после подачи в него любого слова $\tilde{\alpha}$ длиной τ будет определяться только этим словом, последующей строкой входных символов ξ и, может быть, начальным состоянием q». Иными словами, происходит следующее: автомат строками α или α' переводится из состояния q в состояние $\psi(q, \alpha)$ или $\psi(q, \alpha')$ соответственно; затем строкой $\tilde{\alpha}$ длиной τ в состояние $\psi(\psi(q, \alpha), \tilde{\alpha}) = \psi(q, \alpha\tilde{\alpha})$ или $\psi(q, \alpha'\tilde{\alpha})$ соответственно; после этого он будет вести себя одинаково в обоих случаях (выдавая на входную последовательность ξ одну и ту же выходную), т. е. синхронизируется строкой $\tilde{\alpha}$.

Определение 7. Последовательностный шифр $S = <X, Y, K, f, g>$ называется *самосинхронизирующимся с задержкой* τ, если выполняются следующие условия:

1) $\forall \beta \in Y^* \forall k \in K\, |g(\beta, k)| = |\beta|$,
2) для любых β, ζ, β' в Y^*, $\tilde{\beta}$ в Y^τ, $\alpha, \xi, \alpha', \xi'$ в X^*, $\tilde{\alpha}, \tilde{\alpha'}$ в X^τ и $k \in K$, таких, что $|\xi| = |\xi'| = |\zeta|$, $g(\beta\tilde{\beta}\zeta, k) = \alpha\tilde{\alpha}\xi$ и $g(\beta'\tilde{\beta}\zeta, k) = \alpha'\tilde{\alpha'}\xi'$, имеет место $\xi = \xi'$.

Содержательный смысл этого понятия аналогичен предыдущему. Если произвольным образом исказить часть исходного шифртекста $\beta\tilde{\beta}\zeta$ (заменив β на β'), то при расшифровании искажения распространятся не далее чем на τ символов от последнего искаженного символа.

Определение 8 (конструктивное) [1]. Назовём поточную шифрсистему $\Sigma_S = <X, K, Y, G, e, d>$ *самосинхронизирующейся с задержкой* τ, если при любом ключе $k \in K$ проекция G_k автомата G является самосинхронизирующимся автоматом с задержкой τ.

Определение 9 (дескриптивное). Назовём поточную шифрсистему *самосинхронизирующейся с задержкой* τ, если все задаваемые ею последовательностные шифры являются самосинхронизирующимися с задержкой τ.

Будем предполагать далее, что в множестве Γ любой рассматриваемой поточной шифрсистемы Σ_S нет различных эквивалентных символов, т. е. таких γ и γ', что $\gamma \neq \gamma'$, но $\forall x \in X[e(x, \gamma) = e(x, \gamma')]$ и $\forall y \in Y[d(y, \gamma) = d(y, \gamma')]$. Ввиду возможности замены всех попарно эквивалентных символов гаммы одним из них без изменения результатов шифрования и расшифрования сообщений в системе данное предположение не приводит к потере общности рассмотрения.

Теорема 4. Дескриптивное и конструктивное определения самосинхронизирующейся поточной шифрсистемы (без эквивалентных символов гаммы) равносильны — определяют одно и то же множество шифрсистем.

Доказательство. Рассмотрим произвольную поточную шифрсистему $\Sigma_S = <X, Y, K, G, e, d>$ и её ГКП $G = <Y \times K, Q, \Gamma, \psi, \varphi>$. Требуется доказать равносильность следующих двух высказываний:

(Д) = «все последовательностные шифры, задаваемые системой Σ_S, являются самосинхронизирующимися с задержкой τ»;

(К) = «при любом ключе $k \in K$ проекция G_k автомата G является самосинхронизирующимся автоматом с задержкой τ».

Используем следующие обозначения:

$$\beta = y_{-n}y_{-n+1}\dots y_{-1};\ \beta' = y'_{-m}y'_{-m+1}\dots y'_{-1};\ \tilde{\beta} = y_0y_1\dots y_{\tau-1};\ \zeta = y_\tau y_{\tau+1}\dots y_{l-1};$$

$$\alpha = x_{-n}x_{-n+1}\dots x_{-1};\ \tilde{\alpha} = x_0x_1\dots x_{\tau-1};\ \alpha' = x'_{-m}x'_{-m+1}\dots x'_{-1};\tilde{\alpha'} = x'_0x'_1\dots x'_{\tau-1};$$

$$\xi = x_\tau x_{\tau+1}\dots x_{l-1};\ \xi' = x'_\tau x'_{\tau+1}\dots x'_{l-1},$$

где $y_i, y'_i \in Y$, $x_i, x'_i \in X$.

Докажем сначала, что (К) $\Rightarrow$ (Д). Для этого нужно доказать следующие два предложения:

1) $\forall q \in Q \forall \beta \in Y^* \forall k \in K \left|\bar{d}_q(\beta, k)\right| = |\beta|$;
2) $\forall q \in Q \forall \beta, \beta', \zeta \in Y^* \forall \tilde{\beta} \in Y^\tau \forall k \in K[(\bar{d}_q(\beta\tilde{\beta}\zeta, k) = \alpha\tilde{\alpha}\xi)\ \&\ (\bar{d}_q(\beta'\tilde{\beta}\zeta, k) = \alpha'\tilde{\alpha'}\xi') \Rightarrow (\xi = \xi')]$.

Первое предложение следует из определения функции $\bar{d}$. Для доказательства второго возьмем произвольные $q \in Q$, $\beta, \beta', \zeta \in Y^*$, $\tilde{\beta} \in Y^\tau$, $k \in K$. Пусть для некоторых $\alpha, \xi, \alpha', \xi'$ в X^*, $\tilde{\alpha}, \tilde{\alpha'}$ в X^τ, таких, что $|\xi| = |\xi'| = |\zeta|$, будет $\bar{d}_q(\beta\tilde{\beta}\zeta, k) = \alpha\tilde{\alpha}\xi$ и $\bar{d}_q(\beta'\tilde{\beta}\zeta, k) = \alpha'\tilde{\alpha'}\xi'$. Вычислим $q_\tau = \psi_k(q, \beta\tilde{\beta})$, $q'_\tau = \psi_k(q, \beta'\tilde{\beta})$, $\gamma = \gamma_\tau\gamma_{\tau+1}\dots\gamma_{l-1} = \bar{\varphi}_k(q_\tau, \zeta) = \bar{\varphi}_k(\psi_k(q, \beta\tilde{\beta}), \zeta)$ и $\gamma' = \gamma'_\tau\gamma'_{\tau+1}\dots\gamma'_{l-1} = \bar{\varphi}_k(q'_\tau, \zeta) = \bar{\varphi}_k(\psi_k(q, \beta'\tilde{\beta}), \zeta)$. В силу (К) имеет место $\bar{\varphi}_k(\psi_k(q, \beta\tilde{\beta}), \zeta) = \bar{\varphi}_k(\psi_k(q, \beta'\tilde{\beta}), \zeta)$, поэтому $\gamma = \gamma'$ и $x_i = d(y_i, \gamma_i) = d(y_i, \gamma'_i) = x'_i$ для $i = \tau, \tau+1, \dots, l-1$, т. е. $\xi = \xi'$. Таким образом, действительно (К) $\Rightarrow$ (Д).

Обратное следование (Д) $\Rightarrow$ (К) докажем от противного. Предположим, что не верно (К), то есть для некоторого ключа $k \in K$ проекция G_k ГКП G не является самосинхронизирующимся автоматом с задержкой τ и, следовательно,

$$\bar{\varphi}_k(\psi_k(q, \beta\tilde{\beta}), \zeta) \neq \bar{\varphi}_k(\psi_k(q, \beta'\tilde{\beta}), \zeta)$$

для некоторых $q \in Q, \beta, \beta', \zeta \in Y^*, \tilde{\beta} \in Y^\tau$. Пусть $\bar{\varphi}_k(\psi_k(q, \beta\tilde{\beta}), \zeta) = \gamma_\tau\gamma_{\tau+1}\dots\gamma_{l-1}$ и $\bar{\varphi}_k(\psi_k(q, y\beta' y\tilde{\beta}), \zeta) = \gamma'_\tau\gamma'_{\tau+1}\dots\gamma'_{l-1}$. Найдем такое t, что $\tau \leqslant t \leqslant l-1$ и $\gamma_\tau = \gamma'_\tau, \gamma_{\tau+1} = \gamma'_{\tau+1}, \dots, \gamma_{t-1} = \gamma'_{t-1}, \gamma_t \neq \gamma'_t$. Положим $\beta_0 = y_{-n}y_{-n+1}\dots y_{-1+t-\tau}$, $\tilde{\beta}_0 = y_{t-\tau}y_{1+t-\tau}\dots y_{\tau-1+t-\tau}$, $\zeta_0 = y_{\tau+t-\tau}y_{\tau+1+t-\tau}\dots y_{l-1}$ и $\beta'_0 = y_{-m}y_{-m+1}\dots y_{-1+t-\tau}$. Имеем: $\beta_0\tilde{\beta}_0\zeta_0 = \beta\tilde{\beta}\zeta$, $\beta'_0\tilde{\beta}_0\zeta_0 = \beta'\tilde{\beta}\zeta$, $\bar{\varphi}_k(\psi_k(q, \beta_0\tilde{\beta}_0), \zeta_0) = \gamma_t\gamma_{t+1}\dots\gamma_{l-1}$, $\bar{\varphi}_k(\psi_k(q, \beta'_0\tilde{\beta}_0), \zeta_0) = \gamma'_t\gamma'_{t+1}\dots\gamma'_{l-1}$ и $\gamma_t \neq \gamma'_t$.

Тем самым показано, что существуют такие $k \in K$, $q \in Q, \beta_0, \beta'_0 \in Y^*, \tilde{\beta}_0 \in Y^\tau$ и $y_t \in Y$, что $\gamma_t = \varphi_k(\psi_k(q, \beta_0\tilde{\beta}_0), y_t) \neq \varphi_k(\psi_k(q, \beta'_0\tilde{\beta}_0), y_t) = \gamma'_t$, или с учётом того, что G_k является автоматом Мура, $\gamma_t = \varphi_k(q_t) \neq \varphi_k(q'_t) = \gamma'_t$ для $q_t = \psi_k(q, \beta_0\tilde{\beta}_0)$ и $q'_t = \psi_k(q, \beta'_0\tilde{\beta}_0)$.

Пусть для любого $y \in Y$ и некоторых $\alpha, \alpha' \in X^*, \tilde{\alpha}, \tilde{\alpha'} \in X^\tau, x, x' \in X$

$$\bar{d}_q(\beta_0\tilde{\beta}_0 y, k) = \alpha\tilde{\alpha}x,\ \bar{d}_q(\beta'_0\tilde{\beta}_0 y, k) = \alpha'\tilde{\alpha'}x'.$$

Тогда по определению $\bar{d}$ будет $x = d(y, \varphi_k(q_t)) = d(y, \gamma_t)$ и $x' = d(y, \varphi_k(q'_t)) = d(y, \gamma'_t)$, а в силу (Д) — $x = x'$. Таким образом, $\forall y \in Y(d(y, \gamma_t) = d(y, \gamma'_t))$, откуда ввиду взаимной обратимости d и e следует $\forall x \in X(e(x, \gamma_t) = e(x, \gamma'_t))$, что противоречит отсутствию в Г эквивалентных символов. ■

5. Самосинхронизирующиеся автоматные шифрсистемы

Определение 10. Назовём автомат Мура $R = <Y, Y^\tau, \Gamma, \sigma, \varphi>$ *регистром сдвига длиной* τ, если его функция переходов σ есть функция сдвига, определяемая по формуле

$$\sigma(y_{t-\tau}y_{t-\tau+1}\dots y_{t-1}, y_t) = y_{t-\tau+1}y_{t-\tau+2}\dots y_t.$$

Определение 11. Назовём *регистром сдвига длиной* τ *с ключом* такой автомат Мили с двумя входами $R = <Y \times K, Y^\tau, \Gamma, \sigma', \varphi>$, что для любого значения $k \in K$ его проекция $R_k = <Y, Y^\tau, \Gamma, \sigma'_k, \varphi_k>$ на вход Y есть регистр сдвига длиной τ.

Определение 12. Назовем поточную шифрсистему $\Sigma_R = <X, K, Y, R, e, d>$ *регистровой*, если её ГКП R есть регистр сдвига с ключом. Длина регистра R называется размерностью шифрсистемы.

Для регистровых шифрсистем функцию шифрования $\bar{e}$ можно переписать в более простой форме:

$$\bar{e}(q_0, x_0x_1\dots x_{n-1}, k) = y_0y_1\dots y_{n-1}, \text{ где } \begin{cases} \gamma_t = \varphi_k(y_{t-\tau}y_{t-\tau+1}\dots y_{t-1}), \\ y_t = e(x_t, \gamma_t), \end{cases} \quad t = 0, 1, \dots, n-1.$$

Здесь $q_0 = y_{-\tau}y_{-\tau+1}\dots y_{-1}$.

Эти уравнения проиллюстрированы схемой рис. 3.

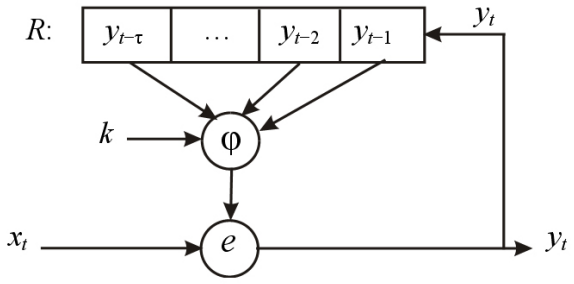

Рис. 3. Схема регистровой шифрсистемы

Регистровая шифрсистема размерностью τ является самосинхронизирующейся с задержкой τ по конструктивному определению. Как видно из следующей теоремы, в некотором смысле верно и обратное.

Теорема 5 [1]. Любая самосинхронизирующаяся с задержкой τ поточная шифрсистема с обратной связью $\Sigma_S = <X, Y, K, G, e, d>$, у которой все проекции G_k ГКП G суть сильно связные автоматы, неотличима от некоторой регистровой шифрсистемы $\Sigma_R = <X, Y, K, R, e, d>$ размерности τ.

Заметим, что в [5] под самосинхронизирующимися поточными шифрами понимаются именно регистровые шифрсистемы. Теорема 5 служит по существу теоретическим обоснованием такого понимания.

Определение 13. Назовём автоматную шифрсистему $\Sigma_A = <X, Y, K, E, D>$ *самосинхронизирующейся с задержкой* τ, если для её автомата расшифрования $D = <Y \times K, Q^*, X, \psi^*, \varphi^*>$ выполняется следующее условие:

$$\forall q^* \in Q^* \forall \beta, \beta', \zeta \in Y^* \forall \tilde{\beta} \in Y^\tau [\bar{\varphi}^*_k(\psi^*_k(q^*, \beta\tilde{\beta}), \zeta) = \bar{\varphi}^*_k(\psi^*_k(q^*, \beta'\tilde{\beta}), \zeta)].$$

Из определений 4, 7 и 13 непосредственно следует, что автоматная шифрсистема является самосинхронизирующейся с задержкой τ, если и только если этим свойством обладают все задаваемые ею последовательностные шифры.

Теорема 6. Любая самосинхронизирующаяся с задержкой τ автоматная шифрсистема $\Sigma_A = <X, K, Y, E, D>$, у которой все проекции E_k шифрующего автомата E суть сильно связные автоматы, неотличима от некоторой регистровой шифрсистемы $\Sigma_R = <X, K, Y, R, e, d>$ размерности τ.

Доказательство. Следуя доказательству теоремы 3, построим поточную шифрсистему $\Sigma_S = <X, Y, K, G, e, d>$, эквивалентную шифрсистеме $\Sigma_A = <X, Y, K, E, D>$. Поскольку шифрсистемы эквивалентны и шифрсистема Σ_A является самосинхронизирующейся с задержкой τ, то и шифрсистема Σ_S также является самосинхронизирующейся с задержкой τ.

Зафиксируем любое значение ключа $k \in K$ и рассмотрим в Σ_S проекцию G_k ГКП G на вход Y. По построению в доказательстве теоремы 3 функция переходов ГКП G определяется как

$$\psi'(q, (y, k)) = \psi(q, (\varphi_{qk}^{-1}(y), k)).$$

Соответственно этому функция переходов его проекции G_k определится как

$$\psi'_k(q, y) = \psi_k(q, \varphi_{qk}^{-1}(y)).$$

Поскольку по теореме 2 функция $\varphi_{qk}(x)$ инъективна, то обратная ей функция $\varphi_{qk}^{-1}(y)$ существует и сюръективна. В этом случае для любых $q_1, q_2 \in Q(= Q')$, $\alpha \in X^*$, $\beta \in Y^*$ если $\psi_k(q_1, \alpha) = q_2$ и $\bar{\varphi}_k(q_1, \alpha) = \beta$, то $\bar{\varphi}_k^{-1}(q_1, \beta) = \alpha$ и $\psi'_k(q_1, \beta) = \psi_k(q_1, \bar{\varphi}_{qk}^{-1}(\beta)) = \psi_k(q_1, \alpha) = q_2$, и из сильной связности автомата E_k следует сильная связность автомата G_k. Теперь по теореме 5 шифрсистема Σ_S, а вместе с ней и шифрсистема Σ_A, неотличима от некоторой регистровой шифрсистемы Σ_R размерности τ. ■

ЛИТЕРАТУРА

1. *Панкратов И. В.* К определению понятия самосинхронизирующегося поточного шифра // Вестник Томского госуниверситета. Приложение. 2007. № 23. С. 114–117.
2. *Панкратов И. В.* О поточных и автоматных шифрсистемах // Прикладная дискретная математика. Приложение. 2009. № 1. С. 21–24.
3. *Агибалов Г. П., Оранов А. М.* Лекции по теории конечных автоматов. Томск: Изд-во Том. ун-та, 1984. 185 с.
4. *Закревский А. Д.* Метод автоматической шифрации сообщений // Прикладная дискретная математика. 2009. № 2. С. 127–137.
5. *Menezes A., van Oorshot P., Vanstone S.* Handbook of Applied Cryptography. CRC Press, 1996. 661 p.

МАТЕМАТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

DOI 10.17223/20710410/5/8

УДК 004.94

АНАЛИЗ УСЛОВИЙ ПОЛУЧЕНИЯ ДОСТУПА ВЛАДЕНИЯ В РАМКАХ БАЗОВОЙ РОЛЕВОЙ ДП-МОДЕЛИ БЕЗ ИНФОРМАЦИОННЫХ ПОТОКОВ ПО ПАМЯТИ[1]

П. Н. Девянин

Институт криптографии, связи и информатики, г. Москва, Россия

E-mail: peter_devyanin@hotmail.com

В рамках базовой ролевой ДП-модели анализируются условия получения недоверенными субъект-сессиями доступа владения к доверенным субъект-сессиям. При этом рассматривается случай, когда взаимодействует произвольное число субъект-сессий, и они не получают доступа владения друг к другу с использованием информационных потоков по памяти к функционально ассоциированным с субъект-сессиями сущностям.

Ключевые слова: *компьютерная безопасность, ролевая модель, ДП-модели.*

1. Основные элементы базовой ролевой ДП-модели

На основе семейства ролевых моделей $RBAC$ [1 – 3] и семейства ДП-моделей компьютерных систем (КС) с дискреционным или мандатным управлением доступом [4] построена базовая ролевая ДП-модель (БР ДП-модель) [5, 6]. Эта модель ориентирована на анализ в КС с ролевым управлением доступом условий передачи прав доступа ролей и реализации информационных потоков по памяти и по времени.

В настоящее время в рамках БР ДП-модели не удалось завершить исследования КС с ролевым управлением доступом, на условия функционирования которых не наложено ограничений. В связи с этим в [5, 6] с применением БР ДП-модели выполнен анализ необходимых и достаточных условий передачи прав доступа для случая, когда в системе существуют только две субъект-сессии двух пользователей.

Рассмотрим БР ДП-модель, в которой взаимодействует произвольное число субъект-сессий, и они не получают доступа владения друг к другу с использованием информационных потоков по памяти к функционально ассоциированным с субъект-сессиями сущностям.

Основными элементами БР ДП-модели являются:

$E = O \cup C$ — множество сущностей, где O — множество объектов, C — множество контейнеров и $O \cap C = \varnothing$;

U — множество пользователей;

L_U — множество доверенных пользователей;

N_U — множество недоверенных пользователей;

[1]Работа выполнена в рамках реализации ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009–2013 годы. Результаты работы докладывались на Международной конференции с элементами научной школы для молодёжи, г. Омск, 7–12 сентября 2009 г.

$S \subseteq E$ — множество субъект-сессий пользователей;

L_S — множество доверенных субъект-сессий;

N_S — множество недоверенных субъект-сессий;

R — множество ролей;

AR — множество административных ролей;

$R_r = \{read_r, write_r, append_r, execute_r, own_r\}$ — множество видов прав доступа;

$R_a = \{read_a, write_a, append_a, own_a\}$ — множество видов доступа;

$R_f = \{write_m, write_t\}$ — множество видов информационных потоков;

$A \subseteq S \times E \times R_a$ — множество доступов субъект-сессий к сущностям;

$F \subseteq E \times E \times R_f$ — множество информационных потоков между сущностями;

$P \subset E \times R_r$ — множество прав доступа к сущностям;

$UA : U \to 2^R$ — функция авторизованных ролей пользователей;

$AUA : U \to 2^{AR}$ — функция авторизованных административных ролей пользователей;

$PA : R \to 2^P$ — функция прав доступа ролей;

$user : S \to U$ — функция принадлежности субъект-сессии пользователю;

$roles : S \to 2^R \cup 2^{AR}$ — функция текущих ролей субъект-сессий;

$can\_manage\_rights : AR \to 2^R$ — функция администрирования прав доступа ролей;

$H_E : E \to 2^E$ — функция иерархии сущностей;

$H_R : R \to 2^R$ — функция иерархии ролей;

$H_{AR} : AR \to 2^{AR}$ — функция иерархии административных ролей;

$G = (PA, user, roles, A, F, H_E)$ — состояние системы;

$\Sigma(G^*, OP)$ — система, при этом G^* — множество всех возможных состояний, OP — множество правил преобразования состояний;

$G \vdash_{op} G'$ — переход системы $\Sigma(G^*, OP)$ из состояния G в состояние G' с использованием правила преобразования состояний $op \in OP$;

$\Sigma(G^*, OP, G_0)$ — система $\Sigma(G^*, OP)$ с начальным состоянием G_0;

$[s] \subset E \cup U$ — множество сущностей, функционально ассоциированных с субъект-сессией s (при этом по определению выполняется условие $s \in [s]$), и пользователей, каждый из которых может создать субъект-сессию, являющуюся функционально ассоциированной сущностью с субъект-сессией s;

$fa : U \times E \to 2^E \cup 2^U$ — функция, задающая множества сущностей, функционально ассоциированных с субъект-сессией, при ее создании пользователем (или от имени пользователя другой субъект-сессией) из сущности. При этом если пользователь $u \in U$ или субъект-сессия от имени пользователя u не может создать из сущности $e \in E$ новую субъект-сессию, то по определению $fa(u, e) = \varnothing$. Кроме того, если для пользователя $u \in U$ и сущности $e \in E$ существует пользователь $x \in U$, такой, что выполняется условие $x \in fa(u, e)$, то по определению будем считать, что пользователь x может создать субъект-сессию, которая будет являться функционально ассоциированной сущностью с субъект-сессией, создаваемой пользователем u из сущности e. По определению выполняется условие: для каждой субъект-сессии $s \in S$ существует единственная сущность $e_s \in E$, такая, что справедливо равенство $fa(user(s), e_s) = [s]$;

$y(E) \subset L_S \times E$ — множество пар вида (доверенная субъект-сессия, сущность), относительно которых корректна доверенная субъект-сессия y;

$de\_facto\_roles : S \to 2^{R \cup AR}$ — функция фактических текущих ролей субъект-сессий, при этом по определению в каждом состоянии системы $G = (PA, user, roles, A, F, H_E)$ для каждой субъект-сессии $s_1 \in S$ выполняется равенство:

$de\_facto\_roles(s_1) = roles(s_1) \cup \{r \in R \cup AR : \exists s_2 \in S\ [(s_1, s_2, own_a) \in A\ \&\ \&\ r \in roles(s_2)]\}$;

$de\_facto\_rights : S \to 2^P$ — функция фактических текущих прав доступа субъект-сессий, при этом по определению в каждом состоянии системы $G = (PA, user, roles, A, F, H_E)$ для каждой субъект-сессии $s \in S$ выполняется равенство:

$de\_facto\_rights(s) = \{p \in P : \exists r \in de\_facto\_roles(s)\ [p \in PA(r)]\}$;

$de\_facto\_actions : S \to 2^P \times 2^R$ — функция фактических возможных действий субъект-сессий, при этом по определению в каждом состоянии системы $G = (PA, user, roles, A, F, H_E)$ для каждой субъект-сессии $s_1 \in S$ выполняется равенство:

$de\_facto\_actions(s_1) = (PA(roles(s_1)) \times can\_manage\_rights(roles(s_1) \cap AR)) \cup \cup\{(p, r) \in P \times R : \exists s_2 \in S\ \exists (s_1, s_2, own_a) \in A\ [r \in can\_manage\_rights(roles(s_2) \cap AR)\ \&\ \&\ p \in PA(roles(s_2))]\}$.

В БР ДП-модели определены следующие правила преобразования состояний:

— монотонные: $take\_role(x, r)$, $grant\_right(x, r, (y, \alpha_r))$, $create\_entity(x, r, y, z)$, $create\_first\_session(u, r, y, z)$, $create\_session(x, w, r, y, z)$, $rename\_entity(x, y, z)$, $control(x, y, z)$, $access\_own(x, y)$, $take\_access\_own(x, y, z)$, $access\_read(x, y)$, $access\_write(x, y)$, $access\_append(x, y)$, $flow(x, y, y', z)$, $find(x, y, z)$, $post(x, y, z)$, $pass(x, y, z)$, $take\_flow(x, y)$;

— и немонотонные: $remove\_role(x, r)$, $remove\_right(x, r, (y, \alpha_r))$, $delete\_entity(x, y, z)$.

По аналогии с базовой ДП-моделью может быть доказано, что в рамках БР ДП-модели при анализе условий передачи прав доступа, реализации информационных потоков по памяти или по времени можно обойтись применением только монотонных правил преобразования состояний.

2. Условия получения права доступа владения

2.1. Вспомогательные предложения

С целью описания в рамках БР ДП-модели условий получения недоверенной субъект-сессией доступа владения к доверенной субъект-сессии рассмотрим случай, когда взаимодействует произвольное число субъект-сессий, и они не получают доступа владения друг к другу с использованием информационных потоков по памяти к функционально ассоциированным с субъект-сессиями сущностям.

Используем следующие определение и предположение БР ДП-модели.

Определение 1. Назовем траекторию функционирования системы $\Sigma(G^*, OP)$ траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, если при ее реализации используются только монотонные правила преобразования состояний, и доверенные субъект-сессии:

— не берут роли в множество текущих ролей;

— не дают другим ролям права доступа к сущностям;

— не получают доступа владения к субъект-сессиям.

Предположение 1. Каждые пользователь или субъект-сессия системы $\Sigma(G^*, OP)$ вне зависимости от имеющихся у них авторизованных ролей являются либо доверенными, либо недоверенными. Доверенные пользователи или субъект-сессии не создают новых субъект-сессий. Каждые недоверенный пользователь или субъект-сессия могут создать только недоверенную субъект-сессию.

Дадим определения.

Определение 2. Траекторию без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_N} G_N$, где $N \geqslant 0$, назовем простой, если при ее реализации для $0 \leqslant i \leqslant N$ каждое правило op_i не является правилом вида $control(x, y, z)$, использующим для получения доступа владения субъект-сессией x к субъект-сессии y информационный поток по памяти $(x, z, write_m)$, где $z \in [y]$.

Определение 3. Пусть $G_0 = (PA_0, user_0, roles_0, A_0, F_0, H_{E_0})$ — состояние системы $\Sigma(G^*, OP)$, в котором существуют недоверенный пользователь $x \in N_U$ и субъект-сессия или недоверенный пользователь $y \in N_U \cup S_0$, такие, что $x \neq y$. Определим предикат $simple\_can\_access\_own(x, y, G_0)$, который будет истинным тогда и только тогда, когда существуют состояния $G_1, \ldots, G_N$ и правила преобразования состояний $op_1, \ldots, op_N$, такие, что $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_N} G_N$, где $N \geqslant 0$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, и существуют субъект-сессии s_x, $s_y \in S_N$, такие, что $user_N(s_x) = x$, или $s_y = y$, или $user_N(s_y) = y$ и выполняется условие $(s_x, s_y, own_a) \in A_N$.

Для упрощения записи алгоритмически проверяемых необходимых и достаточных условий истинности предиката $simple\_can\_access\_own(x, y, G_0)$ используем следующие определения.

Определение 4. Пусть $G = (PA, user, roles, A, F, H_E)$ — состояние системы $\Sigma(G^*, OP)$, в котором существуют субъект-сессии или недоверенные пользователи $x, y \in N_U \cup S$. Определим предикат $simple\_directly\_access\_own(x, y, G)$, который будет истинным тогда и только тогда, когда или $x = y$, или выполняется одно из следующих условий:

1. Если $y \in N_U$ и $x \in N_U$, то существуют сущность $e_y \in E$ и роль $r_y \in R$, такие, что $(e_y, execute_r) \in PA(UA(y))$, $r_y \in can\_manage\_rights(AUA(y))$ и выполняется одно из условий:

— $r_y \in UA(x)$;
— $x \in fa(y, e_y)$.

2. Если $y \in N_U$ и $x \in N_S \cap S$, то существуют сущность $e_y \in E$ и роль $r_y \in R$, такие, что $(e_y, execute_r) \in PA(UA(y))$, $r_y \in can\_manage\_rights(AUA(y))$ и выполняется одно из условий:

— $r_y \in UA(user(x))$;
— $x \in fa(y, e_y)$.

3. Если $y \in N_U$ и $x \in L_S \cap S$, то существуют сущность $e_y \in E$ и роль $r_y \in R$, такие, что $(e_y, execute_r) \in PA(UA(y))$, $r_y \in can\_manage\_rights(AUA(y))$ и выполняется одно из условий:

— $r_y \in roles(x)$;
— $x \in fa(y, e_y)$.

4. Если $y \in S$ и $x \in N_U$, то выполняется одно из условий:

— $(y, own_r) \in PA(UA(x))$;
— $x \in [y]$.

5. Если $y \in S$ и $x \in N_S \cap S$, то выполняется одно из условий:

— $(y, own_r) \in PA(UA(user(x)))$;
— $x \in [y]$;
— $(x, y, own_a) \in A$.

6. Если $y \in S$ и $x \in L_S \cap S$, то выполняется одно из условий:
— $(y, own_r) \in PA(roles(x))$;
— $x \in [y]$;
— $(x, y, own_a) \in A$.

Определение 5. Пусть $G = (PA, user, roles, A, F, H_E)$ — состояние системы $\Sigma(G^*, OP)$, в котором существует субъект-сессия или недоверенный пользователь $x \in N_U \cup S$. Назовем множество $X \subset N_U \cup S$ островом субъект-сессии или недоверенного пользователя x, если $X = \{x\} \cup \{y \in (N_U \cup S) \setminus \{x\}$: существует последовательность $s_1 s_2 \dots s_m$, где $s_1 = x$, $s_2, \dots, s_m \in (N_U \cup S) \setminus \{x\}$, $s_m = y$ и $m \geqslant 2$, такая, что для каждого i, $1 \leqslant i < m$, истинен предикат $simple\_directly\_access\_own(s_i, s_{i+1}, G)\}$ и $s_{i+1} \neq x$. Определим функцию $island : N_U \cup S \to 2^{N_u \cup S}$, задающую для каждых субъект-сессии или недоверенного пользователя соответствующий им остров.

В отличие от островов в классической модели *Take-Grant* острова в БР ДП-модели задаются для каждых субъект-сессии или недоверенного пользователя в отдельности и могут пересекаться.

Из определения 5 следует, что если существуют субъект-сессия или недоверенные пользователи x и y, такие, что $y \in island(x)$, то $island(y) \subseteq island(x)$.

Утверждение 1. Пусть $G_0 = (PA_0, user_0, roles_0, A_0, F_0, H_{E_0})$ — состояние системы $\Sigma(G^*, OP)$, в котором существуют недоверенный пользователь или недоверенная субъект-сессия $x \in N_U \cup (N_S \cap S_0)$ и субъект-сессия или недоверенный пользователь $y \in island(x) \setminus \{x\}$. Тогда справедливо одно из предложений:
— если $x \in N_U$, то истинен предикат $simple\_can\_access\_own(x, y, G_0)$;
— если $x \in N_S \cap S_0$, то истинен предикат $simple\_can\_access\_own(user_0(x), y, G_0)$.

Доказательство. Пусть выполнены условия утверждения. Тогда по определению 5 в $N_U \cup S$ существует последовательность $s_1 s_2 \dots s_m$, где $m \geqslant 2$, $s_1 = x$, $s_m = y$ и для каждого $i = 1, \dots, m-1$ истинен предикат $simple\_directly\_access\_own(s_i, s_{i+1}, G_0)$ и $s_{i+1} \neq x$. Докажем утверждение индукцией по длине m этой последовательности.

Пусть $m = 2$. Тогда справедливо неравенство $x \neq y$, истинен предикат $simple\_directly\_access\_own(x, y, G_0)$ и по определению 4 истинен предикат $directly\_access\_own(x, y, G_0)$ [6]. Следовательно (см. утверждение 2 и определение 13 в [6]), существуют состояния $G_1, \dots, G_N$ и правила преобразования состояний $op_1, \dots, op_N$, такие, что $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \dots \vdash_{op_N} G_N$, где $N \geqslant 0$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, и существуют субъект-сессии $s_x, s_y \in S_N$, такие, что $(s_x, s_y, own_a) \in A_N$ и выполняется одно из условий:
— $user_N(s_x) = x, user_N(s_y) = y$;
— $user_N(s_x) = x$, $s_y = y$;
— $s_x = x$, $user_N(s_y) = y$;
— $s_x = x$, $s_y = y$.

Тем самым в случае $m = 2$ утверждение доказано.

Предположим (предположение индукции), что таким образом утверждение доказано всегда, когда $2 \leqslant m \leqslant l$ для некоторого $l \geqslant 2$, и докажем его (шаг индукции), когда $m = l + 1$.

Итак, пусть $m = l + 1$ и $z = s_{m-1}$. Тогда $x \in N_U \cup (N_S \cap S_0)$ и $z \in (N_U \cup S_0) \setminus \{x\}$. Рассмотрим случай, когда выполняются условия $x \in N_S \cap S_0$, $z \in S_0$. Иные случаи

рассматриваются аналогично. Так как по определению 5 $z \in island(x) \setminus \{x\}$, то по предположению индукции существуют состояния $G_1, \ldots, G_N$ и правила преобразования состояний $op_1, \ldots, op_N$, такие, что $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_N} G_N$, где $N \geqslant 0$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа и $(x, z, own_a) \in A_N$. Если $z = y$, то шаг индукции обоснован. Пусть поэтому $z \neq y$. Возможны два случая.

П е р в ы й с л у ч а й: $z \in N_S \cap S_0$. Тогда по определению 5 $y \in island(z) \setminus \{z\}$, истинен предикат $simple\_directly\_access\_own(z, y, G_0)$ и по предположению индукции существуют состояния $G_{N+1}, \ldots, G_K$ и правила преобразования состояний $op_{N+1}, \ldots, op_K$, такие, что $G_N \vdash_{op_{N+1}} G_{N+1} \vdash_{op_{N+2}} \ldots \vdash_{op_K} G_K$, где $K \geqslant N$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа и $(z, y, own_a) \in A_K$. Положим $op_{K+1} = take\_access\_own(x, z, y)$. Тогда $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_{K+1}} G_{K+1}$ является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, выполняется условие $(x, y, own_a) \in A_{K+1}$, и по определению 3 шаг индукции обоснован.

В т о р о й с л у ч а й: $z \in L_S \cap S_0$. Тогда истинен предикат $simple\_directly\_access\_own(z, y, G_0)$ и выполняется условие 6 определения 4.

Если $(y, own_r) \in PA(roles_0(z))$, то $(y, own_r) \in de\_facto\_rights(x)$ и тогда пусть $op_{N+1} = access\_own(x, y)$. Если $z \in [y]$, то положим $op_{N+1} = control(x, y, z)$. Наконец, если $(z, y, own_a) \in A_0$, то пусть $op_{N+1} = take\_access\_own(x, z, y)$.

Таким образом, $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_{N+1}} G_{N+1}$ является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, выполняется условие $(x, y, own_a) \in A_{N+1}$, и по определению 3 шаг индукции обоснован.

Утверждение доказано. ■

Введём обозначения:

$island\_roles : N_U \cup (N_S \cap S) \to 2^R \cup 2^{AR}$ — функция, задающая для каждых недоверенного пользователя или недоверенной субъект-сессии $x \in N_U \cup (N_S \cap S)$ роли, которыми обладают все субъект-сессии или недоверенные пользователи, принадлежащие острову x. При этом по определению справедливо равенство $island\_roles(x) = \{r \in R \cup AR : \exists y \in island(x)\ [(y \in N_U \,\&\, r \in UA(y) \cup AUA(y)) \vee (y \in N_S \cap S \,\&\, r \in UA(user(y)) \cup AUA(user(y))) \vee (y \in L_S \cap S \,\&\, r \in roles(y))]\}$;

$island\_rights : N_U \cup (N_S \cap S) \to 2^P$ — функция, задающая для каждых недоверенного пользователя или недоверенной субъект-сессии $x \in N_U \cup (N_S \cap S)$ права доступа, которыми обладают все субъект-сессии или недоверенные пользователи, принадлежащие острову x. При этом по определению справедливо равенство $island\_rights(x) = \{p \in P : \exists r \in island\_roles(x)\ [p \in PA(r)]\}$;

$island\_actions : N_U \cup (N_S \cap S) \to 2^P \times 2^R$ — функция, задающая для каждых недоверенного пользователя или недоверенной субъект-сессии $x \in N_U \cup \cup (N_S \cap S)$ возможные действия, которыми обладают все субъект-сессии или недоверенные пользователи, принадлежащие острову x. При этом по определению справедливо равенство $island\_actions(x) = \{(p, r) \in P \times R : \exists y \in island(x)\ [(y \in N_U \,\&\, (p, r) \in PA(UA(y)) \times can\_manage\_rights(AUA(y))) \vee (y \in N_S \cap S \,\&\, (p, r) \in PA(UA(user(y))) \times can\_manage\_rights(AUA(user(y)))) \vee (y \in L_S \cap S \,\&\, (p, r) \in PA(roles(y)) \times can\_manage\_rights(roles(y) \cap AR))]\}$.

Следствие 1. Пусть $G_0 = (PA_0, user_0, roles_0, A_0, F_0, H_{E_0})$ — состояние системы $\Sigma(G^*, OP)$, в котором существуют недоверенный пользователь или недоверенная

субъект-сессия $x \in N_U \cup (N_S \cap S_0)$. Тогда существуют состояния $G_1, \ldots, G_N$ и правила преобразования состояний $op_1, \ldots, op_N$, такие, что $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_N} G_N$, где $N \geqslant 0$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, и существует недоверенная субъект-сессия $s_x \in N_S \cap S_N$, такая, что либо $user_N(s_x) = x$, либо $s_x = x$ и выполняются условия:

— $island\_roles(x) = island\_roles(s_x) \subset de\_facto\_roles_N(s_x)$ (множество фактических ролей субъект-сессии s_x включает все роли субъект-сессий или пользователей, принадлежащих ее острову);

— $island\_rights(x) = island\_rights(s_x) \subset de\_facto\_rights_N(s_x)$ (множество фактических прав доступа субъект-сессии s_x включает все права доступа субъект-сессий или пользователей, принадлежащих ее острову);

— $island\_actions(x) = island\_actions(s_x) \subset de\_facto\_actions_N(s_x)$ (множество фактических возможных действий субъект-сессии s_x включает все возможные действия субъект-сессий или пользователей, принадлежащих ее острову).

Доказательство. Если $x \in N_U$, то по предположению 1 недоверенный пользователь x может создать недоверенную субъект-сессию s_x, такую, что $user(s_x) = x$. При этом из определений функций *island_ roles*, *island_ rights*, *island_ actions* следует, что справедливы равенства $island\_roles(x) = island\_roles(s_x)$, $island\_rights(x) = = island\_rights(s_x)$, $island\_actions(x) = island\_actions(s_x)$. Если $x \in N_S \cap S_0$, то положим $s_x = x$.

Из утверждения 1 следует, что существуют состояния $G_1, \ldots, G_N$ и правила преобразования состояний $op_1, \ldots, op_N$, такие, что $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_N} G_N$, где $N \geqslant 0$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа. При этом выполняются условия:

— если $y \in (island(x) \backslash \{x\}) \cap N_U$, то существует субъект-сессия $s_y \in S_N$, такая, что $user_N(s_y) = y$ и $(s_x, s_y, own_a) \in A_N$;

— если $y \in (island(x) \backslash \{x\}) \cap S_0$, то $(s_x, s_y, own_a) \in A_N$.

Таким образом, в состоянии G_N выполняются условия:

— $island\_roles(x) = island\_roles(s_x) \subset de\_facto\_roles_N(s_x)$;

— $island\_rights(x) = island\_rights(s_x) \subset de\_facto\_rights_N(s_x)$;

— $island\_actions(x) = island\_actions(s_x) \subset de\_facto\_actions_N(s_x)$.

Утверждение следствия доказано. ■

2.2. Основная теорема

Дадим необходимые определения.

Определение 6. Пусть $G = (PA, user, roles, A, F, H_E)$ — состояние системы $\Sigma(G^*, OP)$, в котором существуют недоверенная субъект-сессия или недоверенный пользователь $x \in N_U \cup (N_S \cap S)$ и субъект-сессии или недоверенные пользователи $y, z \in N_U \cup S$. Будем говорить, что субъект-сессия или недоверенный пользователь y соединяется простым мостом с субъект-сессией или недоверенным пользователем z через недоверенную субъект-сессию или недоверенного пользователя x, если $z \in island(x)$ и существует роль $r_y \in R$, такая, что выполняются следующие два условия.

1. Или $y \in N_U$ и $r_y \in UA(y)$, или $y \in N_S \cap S$ и $r_y \in UA(user(y))$, или $y \in L_S \cap S$ и $r_y \in roles(y)$.

2. Или $z \in N_U$ и $r_y \in can\_manage\_rights(AUA(z))$, или $z \in N_S \cap S$ и $r_y \in can\_manage\_rights(AUA(user(z)))$, или $z \in L_S \cap S$ и $r_y \in can\_manage\_rights(roles(z) \cap AR)$.

Определение 7. Пусть $G = (PA, user, roles, A, F, H_E)$ — состояние системы $\Sigma(G^*, OP)$, в котором существуют недоверенная субъект-сессия или недоверенный пользователь $x \in N_U \cup (N_S \cap S)$ и субъект-сессии или недоверенные пользователи $y, z \in N_U \cup S$. Будем говорить, что субъект-сессия или недоверенный пользователь y соединяется мостом с субъект-сессией или недоверенным пользователем z через недоверенную субъект-сессию или недоверенного пользователя x, когда существуют субъект-сессии или недоверенные пользователи $v, w \in N_U \cup S$ и роли $r_v, r_y \in R$, такие, что $v, w, z \in island(x)$, $w, z \in island(v)$, $z \in island(w)$ и выполняются следующие условия.

1. Или $y \in N_U$ и $r_y \in UA(y)$, или $y \in N_S \cap S$ и $r_y \in UA(user(y))$, или $y \in L_S \cap S$ и $r_y \in roles(y)$.

2. Или $v \in N_U$ и $r_v \in UA(v)$, $r_y \in can\_manage\_rights(AUA(v))$, или $v \in N_S \cap S$ и $r_v \in UA(user(v))$, $r_y \in can\_manage\_rights(AUA(user(v)))$, или $v \in L_S \cap S$ и $r_v \in roles(v)$, $r_y \in can\_manage\_rights(roles(v) \cap AR)$.

3. Или $w \in N_U$ и $r_v \in can\_manage\_rights(AUA(w))$, или $w \in S$ и $(w, own_r) \in \in PA(r_v)$.

Введём обозначения:

$is\_simple\_bridge : (N_U \cup (N_S \cap S)) \times (N_U \cup S) \times (N_U \cup S) \to \{true, false\}$ — функция, для которой по определению справедливо равенство $is\_simple\_bridge(x, y, z) = true$ тогда и только тогда, когда y соединен простым мостом с z через x, где $x \in N_U \cup \cup (N_S \cap S)$, $y, z \in N_U \cup S$;

$is\_bridge : (N_U \cup (N_S \cap S)) \times (N_U \cup S) \times (N_U \cup S) \to \{true, false\}$ — функция, для которой по определению справедливо равенство $is\_bridge(x, y, z) = true$ тогда и только тогда, когда y соединен мостом с z через x, где $x \in N_U \cup (N_S \cap S)$, $y, z \in N_U \cup S$.

В отличие от мостов в классической модели *Take-Grant* мосты в БР ДП-модели являются ориентированными. То есть в БР ДП-модели, если субъект-сессия или недоверенный пользователь y соединяется мостом или простым мостом с субъект-сессией или недоверенным пользователем z, то не обязательно z соединяется мостом с y.

Пусть $G = (PA, user, roles, A, F, H_E)$ — состояние системы $\Sigma(G^*, OP)$. В дополнение к обозначениям, введенным при определении графа доступов в базовой ДП-модели, используем следующие обозначения (см. рис. 1):

- вершины из множества $U \cup S$ (соответствующие пользователям и субъект-сессиям) в графе доступов будут обозначаться «•»;
- вершины из множества $R \cup AR$ (соответствующие ролям и административным ролям) в графе доступов будут обозначаться «∗»;
- если для роли $r \in R \cup AR$ и права доступа $(e, \alpha) \in P$ выполняется условие $(e, \alpha) \in \in PA(r)$, то в графе доступов r соединена с e ребром вида рис. 1, *а*, помеченным α;
- если для недоверенного пользователя $x \in N_U$ и роли $r \in R \cup AR$ выполняется условие $r \in UA(x)$ или $r \in AUA(x)$, то в графе доступов x соединен с r ребром вида рис. 1, *б*, помеченным UA или AUA соответственно;
- если для недоверенной субъект-сессии $x \in N_S \cap S$ и роли $r \in R \cup AR$ выполняется условие $r \in UA(user(x))$ или $r \in AUA(user(x))$, то в графе доступов x соединен с r ребром вида рис. 1, *в*, помеченным UA или AUA соответственно;
- если для доверенной субъект-сессии $x \in L_S \cap S$ и роли $r \in R \cup AR$ выполняется условие $r \in roles(x)$, то в графе доступов x соединен с r ребром вида рис. 1, *г*, помеченным *roles*;

— если для административной роли $ar \in AR$ и роли $r \in R$ выполняется условие $r \in can\_manage\_rights(ar)$, то в графе доступов ar соединена с r ребром вида рис. 1, *д*, помеченным cmr;
— если для субъект-сессий или недоверенных пользователей $x, y \in N_U \cup S$ выполняется условие $y \in island(x)$, то в графе доступов x соединен с y ребром вида рис. 1, *е*, помеченным *island*.

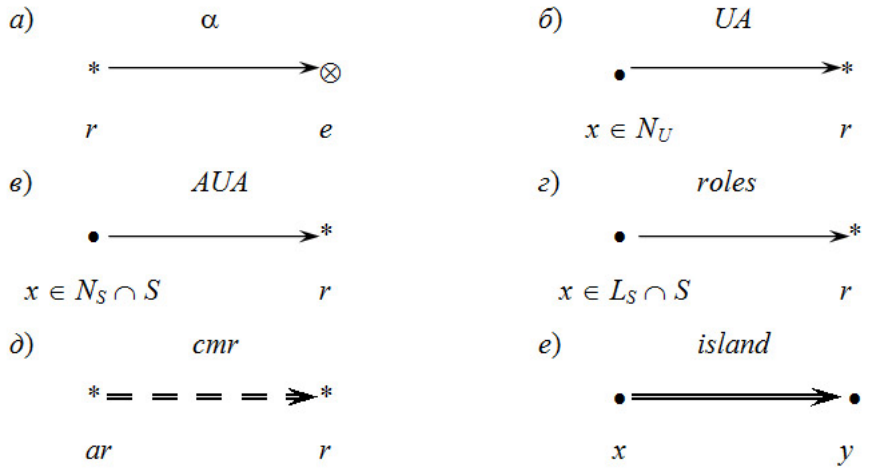

Рис. 1. Обозначения ребер в графе доступов

Пример 1. Пусть $G = (PA, user, roles, A, F, H_E)$ — состояние системы $\Sigma(G^*, OP)$, в котором существуют недоверенная субъект-сессия или недоверенный пользователь $x \in N_U \cup (N_S \cap S)$ и субъект-сессии или недоверенные пользователи $y, z \in N_U \cup S$. С использованием введенных обозначений приведены примеры простого моста (рис. 2, *а*) и моста (рис. 2, *б*), соединяющих y с z через x.

При этом в состоянии, представленном на рис. 2, *а*, выполняется условие $is\_simple\_bridge(x, y, z) = true$, а в состоянии, представленном на рис. 2, *б*, — условие $is\_bridge(x, y, z) = true$.

Сформулируем и обоснуем алгоритмически проверяемые необходимые и достаточные условия истинности предиката $simple\_can\_access\_own(x, y, G_0)$.

Теорема 1. Пусть $G_0 = (PA_0, user_0, roles_0, A_0, F_0, H_{E_0})$ — состояние системы $\Sigma(G^*, OP)$, в котором существуют недоверенный пользователь $x \in N_U$ и субъект-сессия или недоверенный пользователь $y \in N_U \cup S_0$, такие, что $x \neq y$. Предикат $simple\_can\_access\_own(x, y, G_0)$ является истинным тогда и только тогда, когда существуют последовательности недоверенных субъект-сессий или недоверенных пользователей $x_1, \ldots, x_m \in N_U \cup (N_S \cap S_0)$ и субъект-сессий или недоверенных пользователей $y_1, \ldots, y_m \in N_U \cup S_0$, где $m \geqslant 1$, таких, что $x_1 = x$, $y_m = y$, $y_i \in island(x_i)$ для $1 \leqslant i \leqslant m$ и выполняются следующие условия.

1. Если $m \geqslant 2$, то справедливо равенство $is\_bridge(x_m, y_{m-1}, y) = true$.

2. Если $m \geqslant 3$, то для каждого i, $2 \leqslant i < m$, или $is\_bridge(x_i, y_{i-1}, y_i) = true$, или $is\_simple\_bridge(x_i, y_{i-1}, y_i) = true$.

Доказательство. Докажем достаточность выполнения условий теоремы для истинности предиката $simple\_can\_access\_own(x, y, G_0)$. Применим индукцию по длине m последовательностей субъект-сессий или недоверенных пользователей.

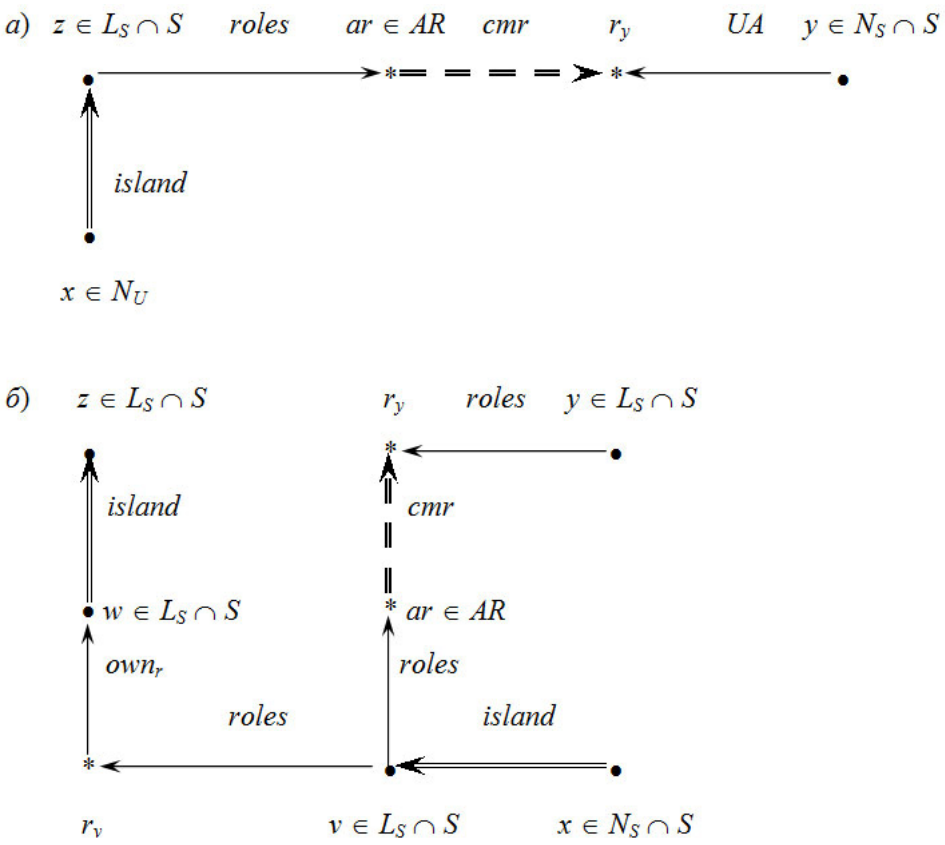

Рис. 2. Примеры мостов: простой мост ($а$); мост ($б$)

Пусть $m = 1$, тогда по условию $y \in island(x)$. Следовательно, по утверждению 1 предикат $simple\_can\_access\_own(x, y, G_0)$ является истинным.

Пусть $m = 2$, тогда по условию $is\_bridge(x_2, y_1, y) = true$, $y_1 \in island(x)$, $y \in island(x_2)$. Значит, по определению 7 существуют субъект-сессии или недоверенные пользователи $v, w \in N_U \cup S_0$ и роли $r_v, r_{y_1} \in R$, такие, что $v, w \in island(x_2)$, $w, y \in island(v)$, $y \in island(w)$ и выполняются следующие условия:

— или $y_1 \in N_U$ и $r_{y_1} \in UA_0(y_1)$, или $y_1 \in N_S \cap S_0$ и $r_{y_1} \in UA_0(user_0(y_1))$, или $y_1 \in L_S \cap S_0$ и $r_{y_1} \in roles_0(y_1)$;
— или $v \in N_U$ и $r_v \in UA_0(v)$, $r_{y_1} \in can\_manage\_rights(AUA_0(v))$, или $v \in N_S \cap S_0$ и $r_v \in UA_0(user_0(v))$, $r_{y_1} \in can\_manage\_rights(AUA_0(user_0(v)))$, или $v \in L_S \cap S_0$ и $r_v \in roles_0(v)$, $r_{y_1} \in can\_manage\_rights(roles_0(v) \cap AR)$;
— или $w \in N_U$ и $r_v \in can\_manage\_rights(AUA_0(w))$, или $w \in S_0$ и $(w, own_r) \in PA_0(r_v)$.

Пример выполнения условий теоремы для случая $m = 2$ приведен на рис. 3.

Если $w \in N_U$ и $r_v \in can\_manage\_rights(AUA_0(w))$, то по предположению 1 существует сущность $e_w \in E_0$, такая, что $(e_w, execute_r) \in PA_0(UA_0(w))$. Положим в этом случае $op_1 = create\_first\_session(w, r_v, e_w, s_w)$ и $M = 1$. Если же $w \in S_0$, то положим $s_w = w$ и $M = 0$.

Таким образом, выполняются условия $(s_w, own_r) \in PA_M(r_v)$, $s_w \in island(x_2)$, $s_w \in island(v)$, $y \in island(s_w)$ и $((s_w, own_r), r_{y_1}) \in island\_actions(x_2)$.

По утверждению следствия 1 существуют состояния $G_{M+1}, \ldots, G_L$ и правила преобразования состояний $op_{M+1}, \ldots, op_L$, такие, что $G_M \vdash_{op_{M+1}} G_{M+1} \vdash_{op_{M+2}} \ldots \vdash_{op_L} G_L$, где $L - M \geqslant 0$, является простой траекторией без кооперации доверенных и недоверенных

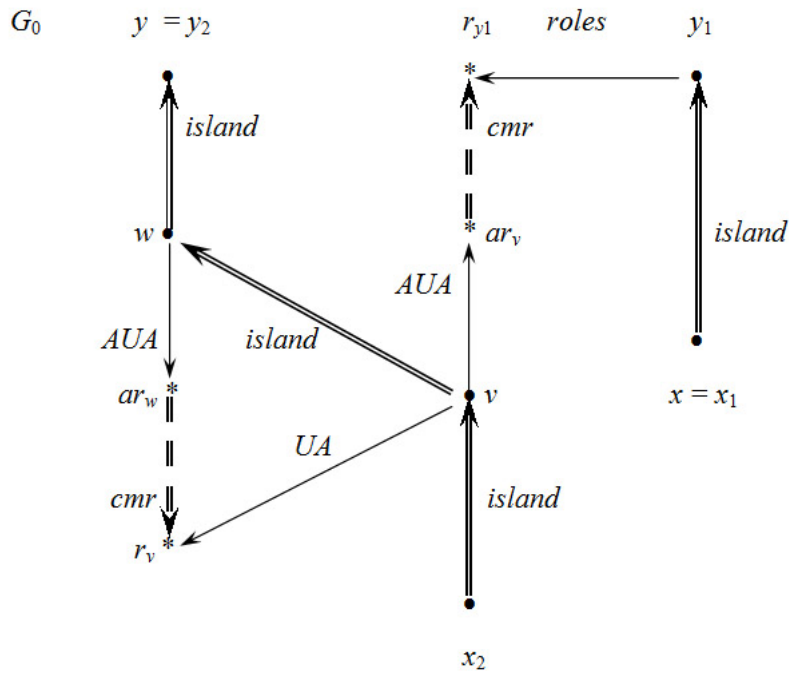

Рис. 3. Пример выполнения условий теоремы для случая $m = 2$

субъект-сессий для передачи прав доступа, и существует недоверенная субъект-сессия $s_{x_2} \in N_S \cap S_L$, такая, что либо $user_L(s_{x_2}) = x_2$, либо $s_{x_2} = x_2$, и в состоянии G_L выполняется условие $island\_actions(x_2) = island\_actions(s_{x_2}) \subset de\_facto\_actions_L(s_{x_2})$. Следовательно, выполняется условие $((s_w, own_r), r_{y_1}) \in de\_facto\_actions_L(s_{x_2})$. Положим $op_{L+1} = grant\_right(s_{x_2}, r_{y_1}, (s_w, own_r))$.

Значит, в состоянии G_{L+1}, таком, что $G_L \vdash_{op_{L+1}} G_{L+1}$, выполняются условия $(s_w, own_r) \in PA(r_{y_1})$ и $y \in island(x)$. Таким образом, по утверждению 1 истинен предикат $simple\_can\_access\_own(x, y, G_{L+1})$. Так как траектория $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_{L+1}} G_{L+1}$ является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, то по определению 3 является истинным предикат $simple\_can\_access\_own(x, y, G_0)$.

Пусть $m > 2$ и утверждение верно для всех последовательностей длины $l < m$. Докажем достаточность условий теоремы при длине последовательности, равной m.

По условию теоремы существуют последовательности недоверенных субъект-сессий или недоверенных пользователей $x_1, \ldots, x_m \in N_U \cup (N_S \cap S_0)$ и субъект-сессий или недоверенных пользователей $y_1, \ldots, y_m \in N_U \cup S_0$, таких, что $x_1 = x$, $y_m = y$, $y_i \in island(x_i)$, где $1 \leqslant i \leqslant m$, и

— справедливо равенство $is\_bridge(x_m, y_{m-1}, y) = true$;
— для каждого i, $2 \leqslant i < m$, или $is\_bridge(x_i, y_{i-1}, y_i) = true$, или $is\_simple\_bridge(x_i, y_{i-1}, y_i) = true$.

По предположению индукции истинен предикат $simple\_can\_access\_own(x_2, y, G_0)$. Используя технику доказательства, примененную для случая $m = 2$, получаем, что существуют состояния $G_1, \ldots, G_M$ и правила преобразования состояний $op_1, \ldots, op_M$, такие, что $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_M} G_M$, где $M \geqslant 0$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, и существует роль $r_{y_2} \in R$, такая, что выполняются следующие условия:

— или $y_2 \in N_U$ и $r_{y_2} \in UA_0(y_2)$, или $y_2 \in N_S \cap S_0$ и $r_{y_2} \in UA_0(user_0(y_2))$, или $y_2 \in L_S \cap S_0$ и $r_{y_2} \in roles_0(y_2)$;
— $(y, own_r) \in PA_M(r_{y_2})$.

Если выполняется условие $is\_bridge(x_2, y_1, y_2) = true$, то по аналогии с рассмотренным случаем для $m = 2$ получаем, что является истинным предикат $simple\_can\_access\_own(x, y, G_0)$.

Пусть выполняется условие $is\_simple\_bridge(x_2, y_1, y_2) = true$ (рис. 4).

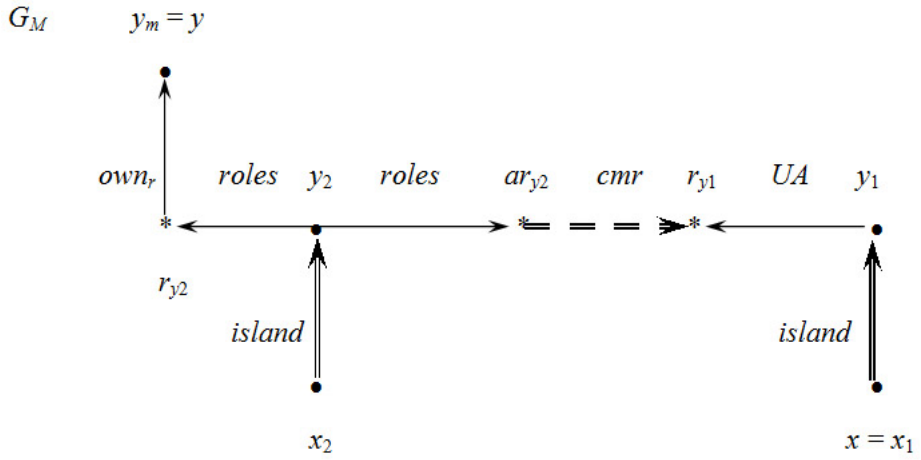

Рис. 4. Пример выполнения условия $is\_simple\_bridge(x_2, y_1, y_2) = true$

Так как $(y, own_r) \in PA_M(r_{y_2})$, $y_2 \in island(x_2)$ и $((y, own_r), r_{y_1}) \in island\_actions(x_2)$, то по утверждению следствия 1 существуют состояния $G_{M+1}, \ldots, G_L$ и правила преобразования состояний $op_{M+1}, \ldots, op_L$, такие, что $G_M \vdash_{op_{M+1}} G_{M+1} \vdash_{op_{M+2}} \ldots \vdash_{op_L} G_L$, где $L - M \geqslant 0$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, и существует недоверенная субъект-сессия $s_{x_2} \in N_S \cap S_L$, такая, что либо $user_L(s_{x_2}) = x_2$, либо $s_{x_2} = x_2$ и в состоянии G_L выполняется условие $island\_actions(x_2) = island\_actions(s_{x_2}) \subset \subset de\_facto\_actions_L(s_{x_2})$. Следовательно, $((y, own_r), r_{y_1}) \in de\_facto\_actions_L(s_{x_2})$. Положим $op_{L+1} = grant\_right(s_{x_2}, r_{y_1}, (y, own_r))$.

Значит, в состоянии G_{L+1}, таком, что $G_L \vdash_{op_{L+1}} G_{L+1}$, выполняются условия $(y, own_r) \in PA(r_{y_1})$ и $y \in island(x)$. Таким образом, по утверждению 1 истинен предикат $simple\_can\_access\_own(x, y, G_{L+1})$. Так как траектория $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_{L+1}} G_{L+1}$ является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, то по определению 3 является истинным предикат $simple\_can\_access\_own(x, y, G_0)$. Индуктивный шаг доказан.

Обоснована достаточность выполнения условий теоремы для истинности предиката $simple\_can\_access\_own(x, y, G_0)$.

Докажем необходимость выполнения условий теоремы для истинности предиката $simple\_can\_access\_own(x, y, G_0)$. По определению 3 существуют состояния $G_1, \ldots, G_N$ и правила преобразования состояний $op_1, \ldots, op_N$, такие, что $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_N} G_N$, где $N \geqslant 0$, является простой траекторией без кооперации доверенных и недоверенных субъект-сессий для передачи прав доступа, и существуют субъект-сессии $s_x, s_y \in S_N$, такие, что $user_N(s_x) = x$, или $s_y = y$, или $user_N(s_y) = y$ и выполняется условие $(s_x, s_y, own_a) \in A_N$.

Среди всех траекторий выберем ту, у которой длина N является минимальной. Проведем доказательство индукцией по длине траекторий N.

Пусть $N = 0$, тогда $(s_x, s_y, own_a) \in A_0$. По определению 4 истинен предикат $simple\_directly\_access\_own(x, y, G_0)$ и $y \in island(x)$. Положим $m = 1$. Следовательно, выполнены условия теоремы.

Пусть $N = 1$, тогда из минимальности N следует, что $(s_x, s_y, own_a) \notin A_0$. Возможны три случая.

П е р в ы й с л у ч а й: $op_1 = control(s_x, s_y, s_z)$, где $s_z \in E_0$ и $s_z \in [s_y]$. При этом либо $s_x = s_z$, либо $s_z \in S_0$ и $(s_x, s_z, own_a) \in A_0$.

В т о р о й с л у ч а й: $op_1 = take\_access\_own(s_x, s_z, s_y)$ и $\{(s_x, s_z, own_a), (s_z, s_y, own_a)\} \subset A_0$.

Т р е т и й с л у ч а й: $op_1 = access\_own(s_x, s_y)$ и $(s_y, own_r) \in de\_facto\_rights_0(s_x)$. Значит, существует $s_z \in S_0$, такая, что $(s_x, s_z, own_a) \in A_0$ и $(s_y, own_r) \in PA_0(roles_0(s_z))$.

Таким образом, в каждом из трех случаев по определению 5 выполняется условие $y \in island(x)$. Положим $m = 1$. Следовательно, выполнены условия теоремы.

Пусть $N > 1$ и утверждение теоремы верно для всех траекторий длины $l < N$. Докажем, что при длине траектории N, если истинен предикат $simple\_can\_access\_own(x, y, G_0)$, то выполняются условия теоремы.

Из минимальности N следует, что $(s_x, s_y, own_a) \notin A_{N-1}$. Возможны три случая.

П е р в ы й с л у ч а й: $op_N = control(s_x, s_y, s_z)$, где $s_z \in E_{N-1}$ и $s_z \in [s_y]$. При этом либо $s_x = s_z$, либо $s_z \in S_{N-1}$ и $(s_x, s_z, own_a) \in A_{N-1}$. Если $s_x = s_z$, то по определению 5 выполняется условие $y \in island(x)$. Положим $m = 1$. Следовательно, выполнены условия теоремы. Пусть $s_x \neq s_z$, тогда положим $z = user_{N-1}(s_z)$. Возможны две ситуации.

Первая ситуация: $s_z \notin S_0$. Тогда по предположению 1 $z \in N_U$, по определению 5 истинен предикат $simple\_directly\_access\_own(z, y, G_0)$ и истинен предикат $simple\_can\_access\_own(x, z, G_0)$ с длиной траектории меньше N. Следовательно, по предположению индукции существуют последовательности недоверенных субъект-сессий или недоверенных пользователей $x_1, \ldots, x_m \in N_U \cup (N_S \cap S_0)$ и субъект-сессий или недоверенных пользователей $y_1, \ldots, y_m \in N_U \cup S_0$, где $m \geqslant 1$, таких, что $x_1 = x$, $y_m = z$, $y_i \in island(x_i)$, где $1 \leqslant i \leqslant m$, и

— если $m \geqslant 2$, то справедливо равенство $is\_bridge(x_m, y_{m-1}, z) = true$;
— если $m \geqslant 3$, то для каждого i, $2 \leqslant i < m$, или $is\_bridge(x_i, y_{i-1}, y_i) = true$, или $is\_simple\_bridge(x_i, y_{i-1}, y_i) = true$.

Если $m = 1$, то по определению 5 выполняется условие $y \in island(x)$. Если $m \geqslant 2$, то по определениям 5 и 7 справедливо равенство $is\_bridge(x_m, y_{m-1}, y) = true$. Следовательно, условия теоремы выполнены.

Вторая ситуация: $s_z \in S_0$. По определению 5 истинен предикат $simple\_directly\_access\_own(s_z, y, G_0)$ и истинен предикат $simple\_can\_access\_own(x, s_z, G_0)$ с длиной траектории меньше N. Далее аналогично обоснованию в первой ситуации получаем, что условия теоремы выполнены.

В т о р о й с л у ч а й: $op_N = take\_access\_own(s_x, s_z, s_y)$ и $\{(s_x, s_z, own_a), (s_z, s_y, own_a)\} \subset A_{N-1}$. Положим $z = user_{N-1}(s_z)$. Возможны две ситуации.

Первая ситуация: $s_z \notin S_0$ или $s_z \in N_S \cap S_0$. Тогда по предположению 1 выполняется условие $z \in N_U$ и истинны предикаты $simple\_can\_access\_own(x, z, G_0)$ и $simple\_can\_access\_own(z, y, G_0)$ с длиной траекторий меньше N. Следовательно, по предположению индукции существуют последовательности недоверенных субъект-сессий или недоверенных пользователей $x'_1, \ldots, x'_n, x''_1, \ldots, x''_k \in N_U \cup (N_S \cap S_0)$ и субъект-

сессий или недоверенных пользователей $y'_1, \ldots, y'_n, y''_1, \ldots, y''_k \in N_U \cup S_0$, где $1 \leqslant n$, $1 \leqslant k$, таких, что $x'_1 = x$, $x''_1 = y'_n = z$, $y''_k = y$, $y'_i \in island(x'_i)$, где $1 \leqslant i \leqslant n$, $y''_i \in island(x''_i)$, где $1 \leqslant i \leqslant k$, и

— если $n \geqslant 2$, то справедливо равенство $is\_bridge(x'_n, y'_{n-1}, z) = true$;
— если $k \geqslant 2$, то справедливо равенство $is\_bridge(x''_k, y''_{k-1}, y) = true$;
— если $n \geqslant 3$, то для каждого i, $2 \leqslant i < n$, или $is\_bridge(x'_i, y'_{i-1}, y'_i) = true$, или $is\_simple\_bridge(x'_i, y'_{i-1}, y'_i) = true$;
— если $k \geqslant 3$, то для каждого i, $2 \leqslant i < k$, или $is\_bridge(x''_i, y''_{i-1}, y''_i) = true$, или $is\_simple\_bridge(x''_i, y''_{i-1}, y''_i) = true$.

Положим: $m = n + k - 1$; $x_i = x'_i$, где $1 \leqslant i \leqslant n$; $x_{n+i-1} = x''_i$, где $2 \leqslant i \leqslant k$; $y_i = y'_i$, где $1 \leqslant i < n$; $y_{n+i-1} = y''_i$, где $1 \leqslant i \leqslant k$. Тогда условия теоремы выполнены.

Вторая ситуация: $s_z \in L_S \cap S_0$. Тогда по определению 3 выполняется условие $(s_z, s_y, own_a) \in A_0$ и истинен предикат $simple\_can\_access\_own(x, s_z, G_0)$ с длиной траектории меньше N. Следовательно, по определению 5 истинен предикат $simple\_directly\_access\_own(s_z, y, G_0)$, и аналогично первому случаю получаем, что условия теоремы выполнены.

Третий случай: $op_N = access\_own(s_x, s_y)$ и выполняется условие $(s_y, own_r) \in$ $\in de\_facto\_rights_{N-1}(s_x)$. Значит, существует $s_z \in S_{N-1}$, такая, что либо $s_x = s_z$ и $(s_y, own_r) \in PA_{N-1}(roles_{N-1}(s_x))$, либо $s_x \neq s_z$, $(s_x, s_z, own_a) \in A_{N-1}$ и $(s_y, own_r) \in$ $\in PA_{N-1}(roles_{N-1}(s_z))$.

Если $s_x = s_z$ и $(s_y, own_r) \in PA_{N-1}(roles_{N-1}(s_x))$, то по определению 5 выполняется условие $y \in island(x)$. Положим $m = 1$. Следовательно, выполнены условия теоремы.

Пусть $s_x \neq s_z$. Тогда $(s_x, s_z, own_a) \in A_{N-1}$ и истинен предикат $simple\_can\_access\_$-$own(x, s_z, G_0)$ с длиной траектории меньше N. Следовательно, по предположению индукции существуют последовательности недоверенных субъект-сессий или недоверенных пользователей $x_1, \ldots, x_n \in N_U \cup (N_S \cap S_0)$ и субъект-сессий или недоверенных пользователей $y_1, \ldots, y_n \in N_U \cup S_0$, где $n \geqslant 1$, таких, что $x_1 = x$, $y_n = s_z$, $y_i \in island(x_i)$, где $1 \leqslant i \leqslant n$, и

— если $n \geqslant 2$, то справедливо равенство $is\_bridge(x_n, y_{n-1}, s_z) = true$;
— если $n \geqslant 3$, то для каждого $2 \leqslant i < n$ справедливо равенство или $is\_bridge(x_i, y_{i-1}, y_i) = true$, или $is\_simple\_bridge(x_i, y_{i-1}, y_i) = true$.

Если либо $s_z \in L_S \cap S_0$ и $(s_y, own_r) \in PA_0(roles_0(s_z))$, либо $z \in N_U$ и $(s_y, own_r) \in$ $\in PA_0(UA_0(z))$, то по определению 7 $is\_bridge(x_n, y_{n-1}, y) = true$. Положим $m = n$. Следовательно, выполнены условия теоремы.

Пусть либо $s_z \in L_S \cap S_0$ и $(s_y, own_r) \notin PA_0(roles_0(s_z))$, либо $z \in N_U$ и $(s_y, own_r) \notin$ $PA_0(UA_0(z))$. Тогда если $s_z \in L_S \cap S_0$, то по определениям 1 и 2 имеет место $r_y \in roles_0(s_z)$. Если $z \in N_U$, то $r_y \in UA_0(z)$. Тогда существует такое M, что $1 \leqslant M < N$ и выполняется одно из следующих трех условий.

Первое условие: $y \in N_U$ и $op_M = create\_first\_session(y, r_y, e_y, s_y)$, где $e_y \in E_{M-1}$ и $r_y \in can\_manage\_rights(AUA_0(y))$. Таким образом, по определению 7 $is\_bridge(x_n, y_{n-1}, y) = true$. Положим $m = n$. Следовательно, выполнены условия теоремы.

Второе условие: $op_M = create\_session(s'_x, s'_y, r_y, e_y, s_y)$, где по предположению 1, определениям 1 и 2 существуют недоверенные субъект-сессии $s'_x, s'_y \in N_S \cap S_{M-1}$, такие, что $y = user_M(s_y) = user_M(s'_y)$, $r_y \in can\_manage\_rights(roles_{M-1}(s'_y) \cap AR)$, $e_y \in E_{M-1}$ и либо $s'_x = s'_y$, либо $(s'_x, s'_y, own_a) \in A_{M-1}$. Следовательно,

$r_y \in can\_manage\_rights(AUA_0(y))$. Таким образом, по определению 7 $is\_bridge(x_n, y_{n-1}, y) = true$. Положим $m = n$. Следовательно, выполнены условия теоремы.

Третье условие: $op_M = grant\_right(s'_x, r_y, (s_y, own_r))$, где по определениям 1 и 2 существует недоверенная субъект-сессия $s'_x \in N_S \cap S_{M-1}$, такая, что $((s_y, own_r), r_y) \in \in de\_facto\_actions_{M-1}(s'_x)$. Следовательно, возможны две ситуации.

Первая ситуация: выполняются условия $(s_y, own_r) \in PA_0(UA_0(user_{M-1}(s'_x)))$ и $r_y \in can\_manage\_rights(AUA_0(user_{M-1}(s'_x)))$, тогда положим $m = n + 1$, $x_m = user_{M-1}(s'_x)$, $y_m = y$. При этом по определению 7 $is\_bridge(x_m, s_z, y) = true$. Следовательно, выполнены условия теоремы.

Вторая ситуация: существует субъект-сессия $s'_y \in S_{M-1}$, такая, что $(s_y, own_r) \in \in PA_{M-1}(roles_{M-1}(s'_y))$, $r_y \in can\_manage\_rights(roles_{M-1}(s'_y) \cap AR)$ и $(s'_x, s'_y, own_a) \in \in A_{M-1}$. Положим $x' = user_{M-1}(s'_x)$. Если $s'_y \in L_S \cap S_0$, то положим $y' = = s'_y$, при этом выполняется условие $r_y \in can\_manage\_rights(roles_0(y') \cap AR)$. Если $s'_y \in N_S \cap S_{M-1}$, то положим $y' = user_{M-1}(s'_y)$, при этом выполняется условие $r_y \in can\_manage\_rights(AUA_0(y'))$. Следовательно, истинен предикат $simple\_can\_access\_own(x', y', G_0)$ с длиной траектории меньше $M - 1$. По предположению индукции существуют последовательности недоверенных субъект-сессий или недоверенных пользователей $x_{n+1}, \ldots, x_{n+k} \in N_U \cup (N_S \cap S_0)$ и субъект-сессий или недоверенных пользователей $y_{n+1}, \ldots, y_{n+k} \in N_U \cup S_0$, где $k \geqslant 1$, таких, что $x_{n+1} = x'$, $y_{n+k} = y'$, $y_i \in island(x_i)$, где $n + 1 \leqslant i \leqslant n + k$, и

— если $k \geqslant 2$, то справедливо равенство $is\_bridge(x_{n+k}, y_{n+k-1}, y') = true$;
— если $k \geqslant 3$, то для каждого i, $n + 2 \leqslant i < n + k$, или $is\_bridge(x_i, y_{i-1}, y_i) = true$, или $is\_simple\_bridge(x_i, y_{i-1}, y_i) = true$;
— выполняется условие $is\_simple\_bridge(x_{n+1}, y_n, y_{n+1}) = true$, где $y_n = s_z$.

Если либо $y' \in L_S \cap S_0$ и $(s_y, own_r) \in PA_0(roles_0(y'))$, либо $y' \in N_U$ и $(s_y, own_r) \in \in PA_0(UA_0(y'))$, то $is\_bridge(x_{n+k}, y_{n+k-1}, y) = true$. Положим $m = n + k$. Следовательно, выполнены условия теоремы.

Пусть ни $y' \in L_S \cap S_0$ и $(s_y, own_r) \in PA_0(roles_0(y'))$, ни $y' \in N_U$ и $(s_y, own_r) \in \in PA_0(UA_0(y'))$. Тогда так как $(s_y, own_r) \in PA_{M-1}(roles_{M-1}(y'))$, то, повторяя (если потребуется) многократно рассуждения, приведённые для третьего случая, получаем, что условия теоремы выполнены.

Следовательно, доказан шаг индукции при длине траектории, равной N. Доказательство необходимости условий теоремы для истинности предиката $simple\_can\_access\_own(x, y, G_0)$ закончено.

Теорема доказана. ■

Таким образом, в рамках БР ДП-модели обосновываются необходимые и достаточные условия получения субъект-сессией, функционирующей от имени недоверенного пользователя, доступа владения к другой субъект-сессии для случая, когда в системе взаимодействует произвольное число субъект-сессий, и они не используют информационные потоки по памяти.

ЛИТЕРАТУРА

1. *Девянин П. Н.* Модели безопасности компьютерных систем: Учеб. пособие для студ. высш. учеб. заведений. М.: Издательский центр «Академия», 2005. 144 с.
2. *Bishop M.* Computer Security: art and science. ISBN 0-201-44099-7, 2002. 1084 p.
3. *Sandhu R.* Role-Based Access Control // Advanced in Computers. Academic Press, 1998. V. 46.

4. *Девянин П. Н.* Анализ безопасности управления доступом и информационными потоками в компьютерных системах. М.: Радио и связь, 2006. 176 с.

5. *Девянин П. Н.* О разработке моделей безопасности информационных потоков в компьютерных системах с ролевым управлением доступом // Материалы Третьей Междунар. научн. конф. по проблемам безопасности и противодействия терроризму. МГУ им. Ломоносова. 25–27 октября 2007 г. М.: МЦНМО, 2008. С. 261–265.

6. *Девянин П. Н.* Базовая ролевая ДП-модель // Прикладная дискретная математика. 2008. № 1(1). С. 64-70.

DOI 10.17223/20710410/5/9
УДК 004.94

ДП-МОДЕЛЬ КОМПЬЮТЕРНОЙ СТЕГАНОГРАФИЧЕСКОЙ СИСТЕМЫ

Е. В. Девянина

ФГНУ «ГНТЦ „Наука“», г. Москва, Россия

E-mail: k_vitalievna@mail.ru

В статье реализуется попытка на основе семейства дискреционных ДП-моделей применить теорию формального моделирования управления доступом и информационными потоками для анализа условий безопасного функционирования компьютерных стеганографических систем.

Ключевые слова: *компьютерная безопасность, стеганография, ДП-модели.*

Большинство существующих подходов, используемых в компьютерной стеганографии [1 – 3], ориентированы либо на применение различных математических (в первую очередь вероятностных) приемов для разработки или анализа стойкости стеганографических преобразований, либо на исследование форматов файлов-контейнеров (изображений, видео, текстовых документов), используемых для сокрытия информации. В то же время стеганографические системы, как правило, реализуются программным обеспечением (ПО), являющимся частью компьютерных систем (КС). Таким образом, на безопасность стеганографической системы может оказывать влияние среда КС, в которой она функционирует.

Важнейшим компонентом обеспечения безопасности большинства существующих КС является система логического управления доступом и информационными потоками, для исследования которой используются формальные модели [4 – 6]. Следовательно, целесообразно рассмотреть подходы к применению таких моделей для анализа безопасности компьютерных стеганографических систем (КСС).

Семейство формальных моделей логического управления доступом и информационными потоками (ДП-моделей) [6, 7] предоставляет механизмы, позволяющие анализировать безопасность современных КС. Как правило, КСС функционируют в КС с дискреционным управлением доступом. Значит, в качестве основы построения формальной модели КСС (далее, сокращенно, КСС ДП-модели) следует выбрать дискреционные ДП-модели.

Заметим, что на безопасность КСС оказывают влияние следующие три фактора, которые будут взяты за основу при построении КСС ДП-модели:

— **целостность** ПО, реализующего КСС. То есть целостность сущностей, функционально ассоциированных с субъектами, реализующими стеганографические преобразования;
— **недоступность** для нарушителя параметров стеганографических преобразований. То есть недоступность данных, содержащихся в сущностях, параметрически ассоциированных с субъектами, реализующими стеганографические преобразования (сущностей, получение данных которых позволит нарушителю однозначно идентифицировать стеганографическое преобразование и осуществить обратное стеганографическое преобразование);

— **зависимость** безопасности ПО, реализующего стеганографические преобразования, от безопасности среды КСС, в которой оно функционирует. Например, зависимость от безопасности используемых в КСС операционных систем (ОС), в том числе от применяемых в ОС технологий администрирования.

Таким образом, при построении КСС ДП-модели целесообразно взять за основу ДП-модель с функционально ассоциированными с субъектами сущностями (ФАС ДП-модель), ДП-модель для политики безопасного администрирования (ПБА ДП-модель) [6] и ДП-модель с функционально и параметрически ассоциированными с субъектами сущностями (ФПАС ДП-модель) [7].

При описании КСС ДП-модели используем следующие предположения и определения.

Предположение 1. Будем считать, что КСС состоит из двух компьютеров (компьютера-источника и компьютера-приемника), удовлетворяющих следующим условиям:

— на компьютере-источнике и компьютере-приемнике функционируют доверенные субъекты «операционная система», реализующие среду для ПО, выполняющего стеганографические преобразования;
— на компьютере-источнике функционирует доверенный субъект, осуществляющий стеганографические преобразования данных из сущности-источника;
— на компьютере-приемнике функционирует доверенный субъект, осуществляющий обратное стеганографическое преобразование данных и их запись в сущность-приемник;
— передача данных между доверенными субъектами компьютера-источника и компьютера-приемника осуществляется через сущность «канал связи» (например, сеть Интернет);
— на компьютере-приемнике функционирует доверенный субъект, осуществляющий непосредственную обработку стеганографических контейнеров (таким образом, стеганографические контейнеры — видео, аудио или текстовые файлы — обрабатываются в соответствии с их содержанием, например, для просмотра).

Определение 1. *Нарушителем* в КСС является недоверенный субъект, имеющий право доступа на чтение к сущности «канал связи» (пассивный нарушитель) или имеющий права доступа на чтение и запись к сущности «канал связи» (активный нарушитель).

Определение 2. *Нарушением безопасности* КСС является реализация субъектом-нарушителем направленного к себе информационного потока по памяти от сущности-источника или от сущности-приемника.

Определение 3. Доверенные субъекты компьютера-источника и компьютера-приемника называются *стеганографически корректными*, когда они не участвуют в реализации информационных потоков по памяти от сущности-источника или сущности-приемника к сущности «канал связи».

Определение 4. Доверенные субъекты компьютера-источника и компьютера-приемника называются *параметрически корректными*, когда они не участвуют в реализации информационных потоков по памяти от параметрически ассоциированных с ними сущностей к сущности «канал связи».

Существует ПО [1], сохраняющее в стеганографических контейнерах данные, позволяющие однозначно идентифицировать стеганографическое преобразование и осуществить обратное преобразование. Значит, не всегда доверенные субъекты, реализующие стеганографические преобразования, являются параметрически корректными. В то же время такие доверенные субъекты, как правило, не осуществляют сохранение в контейнерах или передачу по каналам связи скрываемых данных в явном, непреобразованном виде. Следовательно, стеганографическая корректность доверенных субъектов, реализующих стеганографические преобразования, является естественной для существующих КСС. Поэтому в дальнейшем будем использовать следующее предположение.

Предположение 2. Будем считать, что в рамках КСС ДП-модели все доверенные субъекты являются стеганографически корректными.

Из предположения 2 следует, что доверенные субъекты КСС не осуществляют непосредственную передачу данных, подлежащих стеганографичекому сокрытию, через сущность «канал связи». Кроме того, в соответствии с предположениями ФАС, ФПАС и ПБА ДП-моделей доверенные субъекты не реализуют информационные потоки по времени и имеют права доступа владения к сущностям, размещенным на компьютере, на котором функционирует соответствующий доверенный субъект.

В рамках предположений 1 и 2, используя элементы ФАС, ФПАС и ПБА ДП-моделей, опишем элементы КСС ДП-модели. Используем следующие обозначения:

s_{os_1}, s_{os_2} — доверенные субъекты «операционная система» компьютера-источника и компьютера-приемника соответственно;

$e_{os_1} \in [s_{os_1}]$, $e_{os_2} \in [s_{os_2}]$ — сущности, функционально ассоциированные с субъектами s_{os_1} и s_{os_2} соответственно;

e_s, e_d — сущность-источник и сущность-приемник соответственно;

s_1 — доверенный субъект, реализующий стеганографическое преобразование на компьютере-источнике, при этом по определению выполняются условия $s_1 < s_{os1}$ (субъект s_1 подчинен в иерархии сущностей субъекту s_{os_1}), $s_1 \in [s_{os_1}]$;

s_2 — доверенный субъект, реализующий обратное стеганографическое преобразование на компьютере-приемнике, при этом по определению выполняются условия $s_2 < s_{os_2}$, $s_2 \in [s_{os_2}]$;

s_3 — доверенный субъект, осуществляющий на компьютере-приемнике непосредственную обработку стеганографических контейнеров, при этом по определению выполняются условия $s_3 < s_{os_2}$, $s_3 \in [s_{os_2}]$;

$e_1 \in [s_1]$, $e_2 \in [s_2]$ — сущности, функционально ассоциированные с субъектами s_1 и s_2 соответственно (содержащиеся в сущностях данные задают стеганографическое преобразование, реализуемое субъектами);

$e_3 \in [s_3]$ — сущность, функционально ассоциированная с субъектом s_3;

$e'_1 \in]s_1[$, $e'_2 \in]s_2[$ — сущности, параметрически ассоциированные с субъектами s_1 и s_2 соответственно (содержащиеся в сущностях данные идентифицируют стеганографическое преобразование, реализуемое субъектами);

e_c — сущность «канал связи»;

s_a — недоверенный субъект-нарушитель.

Начальное состояние $G_0 = (S_0, E_0, R_0 \cup A_0 \cup F_0, H_0)$ системы $\Sigma(G^*, OP, G_0)$ зададим в соответствии с рис. 1 (показан случай активного нарушителя, в случае пассивного нарушителя субъект s_a не имеет права доступа $write_r$ к сущности e_c).

Утверждение 1. Пусть в рамках КСС ДП-модели задана система $\Sigma(G^*, OP, G_0)$ и выполнены предположения 1 и 2. Предикат $can\_write\_memory(e_s, s_a, G_0)$ или пре-

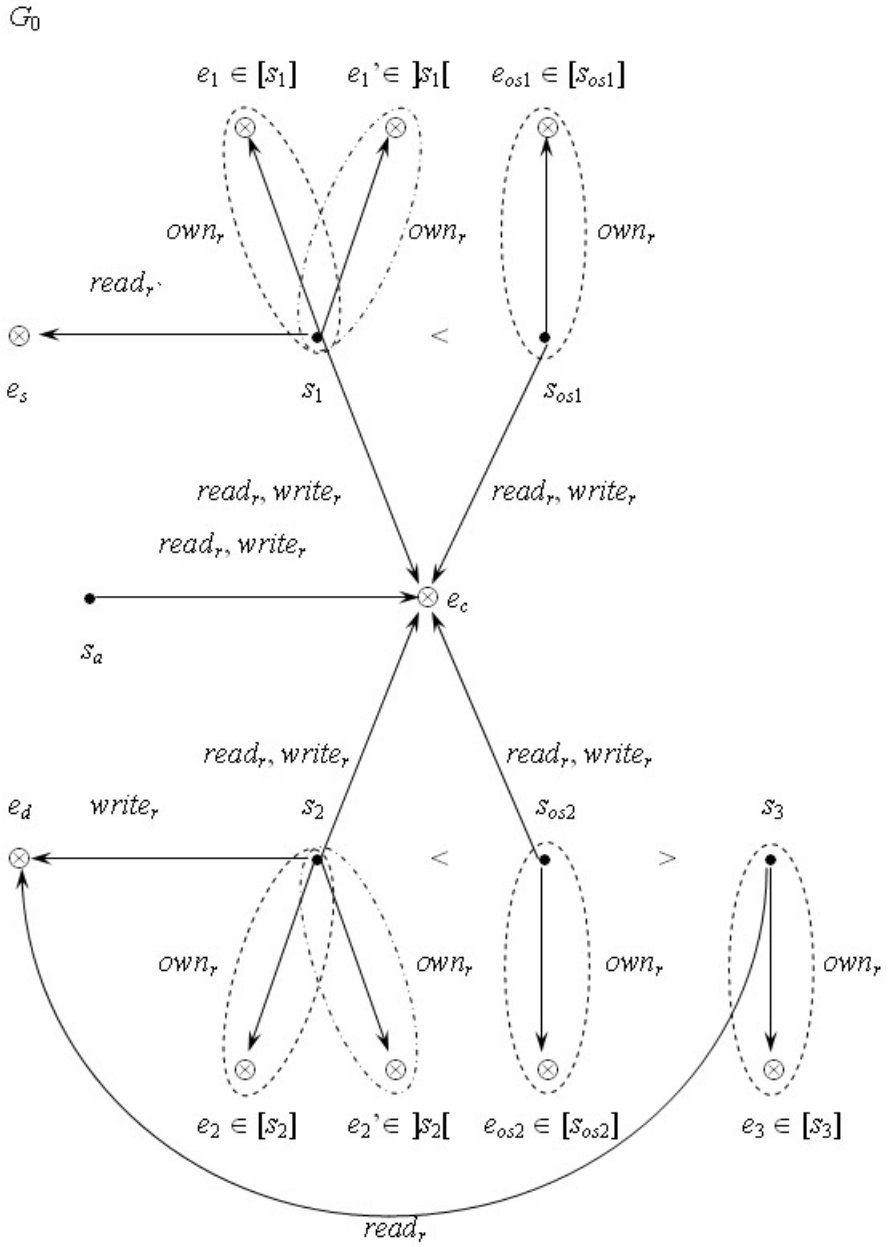

Рис. 1. Начальное состояние системы в рамках КСС ДП-модели

дикат $can\_write\_memory(e_d, s_a, G_0)$ является истинным (происходит нарушение безопасности КСС) тогда и только тогда, когда выполняется одно из следующих трех условий:

Условие 1. Субъект-нарушитель s_a является пассивным или активным, и доверенный субъект s_1 или s_2 является параметрически некорректным.

Условие 2. Субъект-нарушитель s_a является активным, и хотя бы один из доверенных субъектов s_1, s_2, s_{os_1} и s_{os_2} является некорректным (в соответствии с определением, заданным в рамках ФАС ДП-модели) относительно сущности e_c.

Условие 3. Субъект-нарушитель s_a является активным, и доверенный субъект s_3 является некорректным относительно сущности e_d.

Доказательство. Доказательство необходимости выполнения условий утверждения является достаточно трудоемким и проводится аналогично обоснованию необ-

ходимых условий истинности предиката $can\_write\_memory(x, y, G_0)$, выполненному в рамках ФПАС ДП-модели [7].

Доказательство достаточности выполнения условий утверждения является конструктивным. В качестве примера приведем обоснование истинности предиката $can\_write\_memory(e_s, s_a, G_0)$ при выполнении условия 1 утверждения. Пусть доверенный субъект s_1 является параметрически некорректным. Тогда положим:

$op_1 = post(s_1, e_c, s_a)$ — субъект s_1 через сущность e_c реализует информационный поток по памяти к субъекту-нарушителю s_a;

$op_2 = own\_take(read_r, s_1, e'_1)$ — субъект s_1, пользуясь правом доступа владения к сущности e'_1, получает к ней право доступа на чтение;

$op_3 = pass(e'_1, s_1, s_a)$ — субъект s_1, являясь параметрически некорректным, реализует информационный поток по памяти от сущности e'_1 к субъекту-нарушителю s_a;

$op_4 = know(s_a, s_1)$ — субъект-нарушитель s_a, пользуясь информационным потоком по памяти от сущности e'_1, параметрически ассоциированной с субъектом s_1, получает к нему право доступа владения;

$op_5 = take\_right(read_r, s_a, s_1, e_s)$ — субъект-нарушитель s_a берет у субъекта s_1 право доступа на чтение к сущности-источнику e_s;

$op_6 = access\_read(s_a, e_s)$ — субъект-нарушитель s_a получает доступ на чтение к сущности-источнику e_s и реализует от нее к себе информационный поток по памяти.

Следовательно, существует траектория $G_0 \vdash_{op_1} \ldots \vdash_{op_6} G_6 = (S_6, E_6, R_6 \cup A_6 \cup \cup F_6, H_6)$, такая, что выполняется условие $(e_s, s_a, write_m) \in F_6$. Значит, предикат $can\_write\_memory(e_s, s_a, G_0)$ является истинным. ■

Поясним смысл приведенных выше условий.

Условие 1 соответствует случаю, когда субъекты, реализующие стеганографические преобразования, передают в канал связи данные, позволяющие однозначно идентифицировать стеганографические преобразования.

При выполнении условия 2 доверенные субъекты КСС содержат уязвимости реализации, позволяющие активному субъекту-нарушителю, модифицировав данные, передаваемые по каналу связи, изменить содержание сущностей, функционально ассоциированных с доверенными субъектами, и получить над ними контроль.

В случае, когда выполняется условие 3, доверенный субъект, осуществляющий на компьютере-приемнике непосредственную обработку стеганографических контейнеров, содержит уязвимость реализации, позволяющую активному субъекту-нарушителю, модифицировав данные, передаваемые по каналу связи, изменить содержание сущностей, функционально ассоциированных с доверенным субъектом, и получить контроль над доверенными субъектами компьютера-приемника.

Доказательство утверждения показывает, что в рамках КСС ДП-модели возможен анализ безопасности КС, реализующих стеганографические преобразования данных и их передачу по каналам связи. Кроме того, возможно дальнейшее развитие КСС ДП-модели с целью более детального описания свойств и условий функционирования ПО, реализующего стеганографические преобразования, правил управления доступом, параметров администрирования и конфигурирования ОС, используемых в КСС.

ЛИТЕРАТУРА

1. *Аграновский А. В., Девянин П. Н., Хади Р. А., Черемушкин А. В.* Основы компьютерной стеганографии: Учеб. пособие для вузов. М.: Радио и связь, 2003. 152 с.
2. *Грибунин В. Г., Оков И. Н., Туринцев И. В.* Цифровая стеганография. М.: СОЛОН-Пресс, 2002. 272 с.

3. *Коханович Г. Ф., Пузыренко А. Ю.* Компьютерная стеганография. Теория и практика. Киев: МК-Пресс, 2002. 288 с.

4. *Bishop M.* Computer Security: art and science. ISBN 0-201-44099-7, 2002. 1084 p.

5. *Девянин П. Н.* Модели безопасности компьютерных систем: Учеб. пособие для студ. высш. учеб. заведений. М.: Издательский центр «Академия», 2005. 144 с.

6. *Девянин П. Н.* Анализ безопасности управления доступом и информационными потоками в компьютерных системах. М.: Радио и связь, 2006. 176 с.

7. *Колегов Д. Н.* ДП-модель компьютерной системы с функционально и параметрически ассоциированными с субъектами сущностями // Вестник Сибирского государственного аэрокосмического университета им. акад. М. Ф. Решетнева. 2009. Вып. 1(22). Часть 1. С. 49–54.

DOI 10.17223/20710410/5/10

УДК 681.322

МОДЕЛИРОВАНИЕ СЕТЕВЫХ КОМПЬЮТЕРНЫХ СИСТЕМ С УЯЗВИМОСТЯМИ[1]

Д. Н. Колегов

Томский государственный университет, г. Томск, Россия

E-mail: d.n.kolegov@gmail.com

Предлагается дискреционная ДП-модель сетевых компьютерных систем (КС), построенная на основе ДП-модели с функционально и параметрически ассоциированными с субъектами сущностями (ФПАС ДП-модели) и отражающая специфичные условия функционирования сетевых КС с уязвимостями. Показывается возможность применения этой модели для построения всех путей нарушения безопасности в рассматриваемых КС, погружения в неё других моделей, построенных с этой целью, и для проверки условий получения недоверенным субъектом права доступа владения к доверенному субъекту.

Ключевые слова: *сетевые компьютерные системы, тестирование безопасности, граф атак, математические модели безопасности, ДП-модели.*

Введение

Будем рассматривать КС, управляющие доступом и информационными потоками в компьютерных сетях. Будем называть их сетевыми КС. Одним из механизмов обеспечения безопасности таких КС является их тестирование. Среди многих задач тестирования безопасности КС важное место занимает задача обнаружения возможных путей её нарушения, понимаемого как получение недоверенным субъектом права доступа владения к доверенному субъекту КС. Исчерпывающее решение этой задачи предполагает разработку подходящей математической модели безопасности КС — модели, ориентированной именно на построение всех возможных путей нарушения безопасности КС. Моделями с такой ориентацией являются модели формального описания и верификации политик безопасности КС и модели анализа графов атак. Такие модели реализуются подсистемами компьютерного моделирования безопасности в системах автоматизированного тестирования возможности проникновения, управления уязвимостями и предотвращения атак КС.

В известных моделях [1 – 5], ориентированных на построение возможных путей нарушения безопасности КС, обычно строится алгоритм, который из начального состояния КС путем последовательного применения всех определенных в модели правил преобразования состояний, удовлетворящих текущему состоянию КС, получает все возможные траектории ее функционирования. Каждая траектория проверяется на принадлежность к множеству запрещенных траекторий функционирования КС в соответствии с требованиями априорно заданной политики управления доступом и информационными потоками. Основное отличие разных моделей друг от друга заключается в математическом аппарате (формальные грамматики, конечные автоматы, методы верификации, логическое программирование), применяемом для получения всех

[1]Работа выполнена в рамках реализации ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009–2013 годы. Результаты работы докладывались на Международной конференции с элементами научной школы для молодёжи, г. Омск, 7–12 сентября 2009 г.

возможных траекторий функционирования КС и построения по ним возможных путей нарушения безопасности. В каждой из этих моделей используются свои, часто не встречающиеся в других моделях определения элементов и механизмов защиты КС, а реализуемый математический аппарат часто недостаточен для определения условий получения недоверенным субъектом права доступа владения к доверенным субъектам. Всё это затрудняет построение и обоснование правил управления доступом, реализация которых в КС позволяет предотвратить нарушение безопасности последних.

В данной работе в основу построения математической модели безопасности сетевых КС положена одна из дискреционных ДП-моделей [6] — ФПАС ДП-модель [7, 8], наиболее полно отражающая особенности современных КС и обеспечивающая возможность алгоритмической проверки условий получения недоверенными субъектами права доступа владения к доверенным субъектам и построения всех возможных путей нарушения безопасности КС. На её основе строится СДУ ДП-модель, представляющая собой ДП-модель сетевой дискреционной КС с уязвимостями, связанными с ошибками в программном обеспечении (ПО), с доверием субъектов, с доступом к паролям субъектов и раскрытием параметров КС, и ориентированная на применение в системах компьютерного моделирования безопасности КС. В ней ФПАС ДП-модель уточняется с учетом существенных особенностей реализации механизмов защиты и их нарушений в современных сетевых КС. После определения СДУ ДП-модели в статье показывается, как такой моделью может быть описана любая реальная сетевая КС и каким образом с её помощью могут быть построены всевозможные пути нарушения безопасности в КС. На примере модели VTG [2] демонстрируется возможность погружения в СДУ ДП-модель других моделей тестирования безопасности КС.

В изложении материала широко используются определения, предположения и обозначения из [6 – 8]. Кроме того, в соответствии с [9, 10] под уязвимостью КС понимается некоторое свойство КС, которое может быть использовано для нарушения ее безопасности.

1. ДП-модель безопасности сетевой дискреционной КС с уязвимостями

ФПАС ДП-модель, на базе которой строится СДУ ДП-модель, предназначена для адекватного моделирования КС широкого класса, а именно всех таких КС управления доступом и информационными потоками, в которых сущности могут ассоциироваться с субъектами функционально или параметрически. Ввиду своей общности ФПАС ДП-модель, естественно, может не отражать всех существенных деталей КС из более узкого класса, в частности конкретных КС, и тем самым не обеспечивать возможной точности результата моделирования последних. Для ориентации общей модели на подкласс требуется прописать в ней все элементы КС в подклассе и все условия получения в них недоверенными субъектами прав доступа владения к доверенным субъектам. Именно так по ФПАС ДП-модели построена в [11] ДП-модель файловых систем КС с дискреционным управлением доступом, и именно так определяется здесь СДУ ДП-модель — путём определения в ФПАС ДП-модели, во-первых, тех элементов, которые присущи сетевым дискреционным КС с уязвимостями, и, во-вторых, тех условий применения в ней правила $know()$ для получения права доступа владения к субъекту, которые адекватны этому процессу в рассматриваемых КС.

1.1. Основные элементы СДУ ДП-модели

Определение 1. Под состоянием сетевой КС $\Sigma(G^*, OP)$ понимается граф доступов $G = (S, E, R \cup A \cup F, H)$, где

E — множество сущностей, в состав которого входят: $EC \subset E$ — подмножество сущностей-контейнеров (узлов, или компьютеров КС) и $IC \subset E$ — подмножество сущностей-интерфейсов (субъектов или объектов), в которые записывают данные одни субъекты, и из которых считывают данные другие субъекты, осуществляющие взаимодействие;

S — множество субъектов КС;

R — множество ребер графа-состояния G, соответствующих правам доступа пользователей к сущностям;

A — множество ребер графа-состояния G, соответствующих доступам пользователей к сущностям;

F — множество ребер графа-состояния G, соответствующих информационным потокам между сущностями;

H: $E \to 2^E$ — функция иерархии сущностей, ее значения на множестве сущностей соответствуют распределению сущностей по узлам КС, и каждая сущность $e \in E$ либо является узлом КС ($e \in EC$), либо размещена на некотором единственном для каждой сущности узле (для сущности e существует единственный узел $c \in EC$, такой, что $e < c$).

Для нарушения безопасности КС субъекты-нарушители используют, как правило, следующие уязвимости [12 – 15], приводящие к получению ими прав доступа владения к доверенным субъектам:

- пропущенные в системных функциях операционных систем (ОС), применяемых в узлах КС, механизмы проверки на наличие прав использования административных полномочий;
- ошибки в ПО (например, [16, 17]), реализующем прикладные и системные процессы ОС;
- отсутствие аутентификации и авторизации пользователей при доступах к ресурсам узлов КС (доверие);
- ошибки в реализации, конфигурировании и использовании КС, приводящие к реализации информационных потоков по памяти между нарушителем и субъектами КС;
- наличие информационных потоков по памяти от сущностей-паролей к субъектам-нарушителям;
- возможность получения или изменения некоторых параметров КС, позволяющих идентифицировать используемое на узлах системное и прикладное ПО, установленные обновления ОС и зарегистрированных в них пользователей.

Таким образом, будем исходить из следующего предположения.

Предположение. В каждом состоянии $G = (S, E, R \cup A \cup F, H)$ компьютерной системы $\Sigma(G^*, OP)$ выполняются следующие условия.

1. Для каждого узла $c \in EC$ определены доверенные пользователи, обладающие правом доступа владения к каждой сущности, размещенной на данном узле. Если пользователь $os_c \in S$ является доверенным пользователем узла $c \in EC$, то он обладает правом доступа владения к каждой сущности, размещенной на данном узле.

2. Множества I и EC не пересекаются. Множества E, S и функция H не изменяются на всех траекториях функционирования КС. Сущности-интерфейсы из IC используются как для передачи данных между субъектами одного узла, так и между субъектами разных узлов КС. В последнем случае управление доступом к таким интерфейсам может реализовываться КС межсетевого экранирования (МЭ). Дан-

ные сущности ассоциированы с субъектами, которые осуществляют к ним доступ на чтение.

3. Если субъект-пользователь $u \in S$ активизировал процесс $p \in S$, то последний наследует все права пользователя u и является функционально ассоциированным с ним. Если известно, что в некотором процессе $p \in S$ имеются ошибки, приводящие к утечке права доступа, то существует сущность-ошибка, функционально ассоциированная с данным процессом, т. е. существует сущность f в E, такая, что $f \in [p]$ и $(p, f, write_r) \in R$.

4. Существуют подмножества $U_L, S_L \subset S$, такие, что субъекты из U_L могут иметь права доступа к сущностям только на чтение, и если $s \in S_L$, то существуют один или несколько субъектов $u \in U_L$, таких, что $s \in]u[$.

5. Если сущность $p \in E$ является паролем, на основе которого субъекту-пользователю $u_1 \in S$, передавшему пароль через интерфейс $l \in IC$ процессу $r \in S$, последний предоставляет права доступа субъекта $u_2 \in S$ к другим сущностям КС, то $p \in [u_2]$, $p \in]u_2[$, $l \in (r)$, $(u_1, l, write_r) \in R$. Кроме того, если получение прав доступа пользователя u_1 позволяет получить права пользователя u_2, то $(u_1, u_2, own_r) \in R$.

Замечание 1. Из условия 3 следует, что в множество S при моделировании сетевой КС достаточно включить только субъектов-пользователей ее узлов. Для каждого субъекта-пользователя могут существовать несколько сущностей-интерфейсов из IC, в которые записывают и из которых считывают эти субъекты данные.

Замечание 2. Примерами субъектов из S_L в условии 4 являются сетевые службы NetBIOS, позволяющие получать доступ к чувствительной информации КС с правами анонимного пользователя $u \in U_L$ через нулевые сессии в ОС семейства Windows 2000/XP/2003/Vista [13].

Замечание 3. Примерами выполнения условия 5 являются возможность сквозной аутентификации пользователей на Citrix-сервере (аутентифицированный в домене пользователь при доступе к Citrix-серверу домена автоматически аутентифицируется на нем) и механизмы аутентификации служб rlogin, rsh и rexec в ОС семейства UNIX [13].

1.2. Правило *know*() в СДУ ДП-модели

По определению в ФПАС ДП-модели если сущность $z \in E$ параметрически ассоциирована с субъектом $y \in S$ в состоянии G, то чтение данных из z субъектом $x \in S$ позволяет ему получить право доступа владения к субъекту y в этом или последующих состояниях КС с помощью правила $know()$. В ДП-модели файловых систем [11] недоверенный субъект $x \in N_S$ может, применив правило преобразования состояний $know()$, получить право доступа владения к субъекту $y \in S$ тогда и только тогда, когда в КС реализованы информационные потоки по памяти к субъекту x от всех сущностей, параметрически ассоциированных с субъектом y.

В сетевых КС для получения субъектом-нарушителем x права владения к субъекту y ему, как правило, требуется получить доступ не только к сущности z, но и доступ на запись к некоторой сущности $l \in E$, являющейся интерфейсом или портом некоторого субъекта-процесса $r \in S$, осуществляющего предоставление x прав доступа y на основе данных в сущности l. При этом сущности z и l являются ассоциированными с субъектом r; сущности l и r, как правило, размещены на одном узле КС, а сущности z и y могут быть размещены на разных узлах КС. На рис. 1 изображен пример получе-

ния прав доступа субъекта y субъектом-нарушителем x, получившим доступ к паролю z и к интерфейсу l субъекта r. Здесь сущности z, y, r, l размещены на одном узле КС.

Например, для защиты от записи в сущность-интерфейс l субъекта r могут использоваться следующие механизмы защиты: группы доступа (ОС FreeBSD), фильтрация по IP-адресам узлов КС, с которых осуществляется доступ (механизм ACL, выделенные сети управления), и именам пользователей (служба SSH).

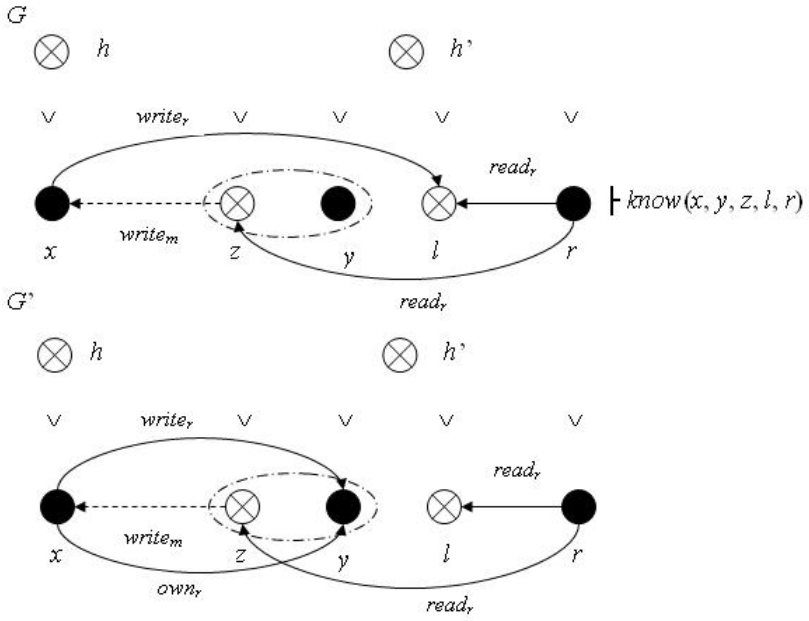

Рис. 1. Правило преобразования состояний $know(x, y, z, l, r)$ СДУ ДП-модели

Соответственно этому в ДП-модели сетевых КС правило $know()$ определяется формально, как показано в табл. 1.

Таблица 1

Условия и результаты применения правила *know(x, y, z, l, r)*

Правило	Исходное состояние $G = (S, E, R \cup A \cup F, H)$	Результирующее состояние $G' = (S', E', R' \cup A' \cup F', H')$
$know(x, y, z, l, r)$	$x, y, r \in S$, $l, z \in E$, $l \in (r)$, $(x, l, write_r) \in R$, $z \in]y[$ и или $x = z$, или $(z, x, write_m) \in F$	$S' = S$, $E' = E$, $A' = A$, $H' = H$, $F' = F$, $R' = R \cup (x, y, own_r)$

1.3. Определение СДУ ДП-модели

Определение 2. ДП-модель компьютерной системы $\Sigma(G^*, OP)$ с множеством правил преобразования состояний ФАС ДП-модели, заданных без учета использования информационных потоков по времени, и правилом преобразования $know()$, определенным в табл. 1, называется сетевой ДП-моделью с уязвимостями, или СДУ ДП-моделью, если её состояния подчиняются определению 1 и удовлетворяют условиям предположения в п. 1.1.

Таким образом, для моделирования безопасности сетевых КС с уязвимостями требуется модифицировать, как показано в табл. 1, условия применения правила преобразования $know()$ ФПАС ДП-модели и построить соответствующее уточнение последней по известной в теории ДП-моделей методике, а именно: в рамках модели сформулировать и обосновать условия возможности получения недоверенным субъектом права доступа владения к доверенному субъекту, скорректировать метод предотвращения получения недоверенным субъектом права доступа владения к доверенному субъекту, разработать алгоритмы построения замыкания графа доступов, графов распространения прав доступов и информационных потоков и сформулировать алгоритмы нахождения всех возможных путей нарушения безопасности. Это построение может быть осуществлено аналогично [7, 8, 18] и выходит за пределы данной статьи.

2. Построение СДУ ДП-модели для реальных КС

Для использования средств анализа безопасности сетевых КС с уязвимостями в рамках их ДП-моделей на практике необходимо представлять реальные КС их СДУ ДП-моделями. Эта задача решается следующим методом.

Метод. Пусть в КС идентифицированы узлы, открытые порты, ОС, сетевые службы, ПО, реализующее последние, и его уязвимости, а также пользователи. Пусть задана политика управления доступом и информационными потоками, реализованная в сетевой подсистеме КС.

Шаг 1. Описать множество EC узлов КС и определить функцию иерархии сущностей H. Для каждого узла КС определить одного доверенного субъекта-пользователя и множество недоверенных пользователей, от имени которых функционируют недоверенные процессы ОС этого узла. Включить пользователей в множество S.

Шаг 2. С помощью систем управления уязвимостями (сканеров безопасности) определить известные уязвимости процессов, запущенных на узлах КС, и наличие сетевых служб, использующих доверие и предоставляющих доступ к параметрам КС. В соответствии с этим определить функционально (найденные известные уязвимости) или параметрически (чувствительная информация) ассоциированные с субъектами-пользователями сущности.

Шаг 3. Описать множество файлов, в которых содержатся данные аутентификации субъектов-пользователей. Определить параметрически ассоциированные сущности с субъектами-пользователями, а также службы регистрации пользователей, удаленного управления и доступа и их интерфейсы в IC.

Шаг 4. Определить сущности-интерфейсы обмена данными между процессами узлов КС и права доступа к ним субъектов узлов КС, исходя из политики управления доступом и информационными потоками (права доступа, таблицы маршрутизации, списки ACL, правила МЭ и систем предотвращения атак). Включить такие сущности в множество IC и построить множество R.

Применение данного метода позволяет получить граф доступов начального состояния G_0 СДУ ДП-модели сетевой КС $\Sigma(G^*, OP)$ с уязвимостями.

3. Применение СДУ ДП-модели

СДУ ДП-модель может быть использована в задачах построения и анализа возможных путей нарушения безопасности КС, а также для представлении известных моделей формального описания и верификации политик безопасности КС и моделей анализа графов атак с целью алгоритмической проверки условий нарушения безопасности в них.

3.1. Построение путей нарушения безопасности КС

Граф доступов G', полученный из графа доступов G ДП-модели КС применением правил преобразования состояний этой ДП-модели, называется замыканием графа доступов G, если применение к графу G' данных правил не приводит к появлению в нем новых ребер.

Пусть требуется построить все пути нарушения безопасности, т. е. все последовательности, состоящие из ребер графа начального состояния и правил преобразования состояний, описывающие переход КС из начального состояний G_0 в некоторое состояние G, в котором нарушена безопасность КС. Для этого следует аналогично [18] построить замыкание графа доступов СДУ ДП-модели, в котором по определению содержатся все запрещенные доступы или информационные потоки, а затем по нему построить граф распространения прав доступов и информационных потоков, что позволит найти все возможные пути данного нарушения безопасности КС в рамках ее модели.

3.2. Разработка и обоснование условий и метода предотвращения возможности получения права доступа владения

В рамках построенной СДУ ДП-модели аналогично [7, 8] могут быть установлены и обоснованы необходимые и достаточные условия возможности получения недоверенным субъектом права доступа владения к доверенному субъекту КС. Затем на основе этих условий можно аналогично методу 4.1 из [6] разработать и теоретически обосновать метод предотвращения в КС такой возможности. Результатом применения этого метода к некоторой КС является КС $\Sigma(G^*, OP, G_0)$, в которой для каждого недоверенного субъекта $x \in N_S \cap S_0$ и доверенного субъекта $y \in L_S \cap S_0$ предикат $can\_share\_own(x, y, G_0, L_S)$ является ложным, что как раз и означает невозможность получения в КС недоверенным субъектом права доступа владения к доверенному субъекту.

3.3. Погружение других моделей в СДУ ДП-модель

Как правило, известные модели КС, ориентированные на поиск возможных путей нарушения их безопасности, не являются субъектно-объектными, поэтому их правила преобразования состояний описываются на языке с высоким уровнем абстракции (например, язык модели абстрактных классов [3]), что требует большого числа правил описания уязвимостей КС. ДП-модели относятся к классу субъектно-объектных моделей. В них все вопросы безопасности (в соответствии с основной аксиомой теории компьютерной безопасности) описываются доступами субъектов к объектам КС, что позволяет моделировать известные уязвимости КС для анализа их безопасности с помощью небольшого числа правил. Основные известные модели, используемые на практике для построения возможных путей нарушения безопасности КС (например, модель REM [1] и расширение [2] модели Take-Grant для КС с уязвимостями (VTG-модель)), могут быть представлены в рамках СДУ ДП-модели.

Покажем возможность представления VTG-модели в рамках разработанной СДУ ДП-модели. Опишем основные положения модели VTG:

- каждому узлу КС соответствует некоторое множество известных уязвимостей;
- дополнительно к стандартным видам прав доступа модели Take-Grant добавлены следующие виды прав доступа: h — принадлежности сущности к узлу, x — выполнения, o — владения;
- наличие уязвимостей в сущностях обозначается с помощью метки последних;

— рассматриваются три класса уязвимостей: переполнение буфера в сегменте стека, доступ к паролям, отношение доверия между сущностями КС;
— возможность использования уязвимостей отражается в новых правилах преобразования состояний $bof\_rule()$, $pc\_rule()$ и $rt\_rule()$ соответственно;
— для проверки возможности получения субъектом x права доступа к сущности y вводится предикат $can\_access(\alpha, x, y, VTG_0)$, истинность которого проверяется путем построения замыкания графа доступов.

Для представления VTG-модели в рамках СДУ ДП-модели следует:

— по уязвимостям КС в первой из них определить ассоциированные, функционально или параметрически ассоциированные с субъектами сущности и права доступа к ним, как в условиях 3 и 5 предположения;
— пометить ребра, используемые в отношении доверия между сущностями, правом own_r;
— вместо прав доступа r, w и x использовать права доступа $read_r$, $write_r$, $execute_r$ соответственно;
— вместо видов прав доступа t, g и o использовать право доступа own_r;
— вместо права h определить функцию H (дуге графа доступов VTG-модели из вершины x в вершину y, помеченной правом h, соответствует отношение $y < x$ в ДП-модели);
— вместо правил $bof\_rule()$ и $pc\_rule()$ использовать правила преобразования $control()$ и $know()$ ФПАС ДП-модели.

Тем самым VTG-модель может быть погружена в СДУ ДП-модель. Возможность представления модели REM [1], реализованной в сканере безопасности NetSPA, в рамках СДУ ДП-модели показывается аналогично в [18].

Таким образом, представление моделей безопасности КС, ориентированных на построение возможных путей нарушения их безопасности, действительно возможно в рамках семейства ДП-моделей с дискреционным управлением доступом.

Заключение

Данное исследование позволяет сделать следующий вывод: семейство ДП-моделей может быть использовано для моделирования сетевых КС с уязвимостями, связанными с ошибками в ПО, с доверием субъектов, с доступом к паролям субъектов и раскрытием параметров КС, для построения всех возможных путей нарушения их безопасности, алгоритмической проверки условий получения недоверенным субъектом прав доступа доверенного субъекта и предотвращения возможности этого как непосредственно в рамках предложенной СДУ ДП-модели, так и посредством представления в ней других моделей безопасности сетевых КС с уязвимостями (например, REM и VTG).

ЛИТЕРАТУРА

1. *Artz M.* NETspa, A Network Security Planning Architecture: M.S. Thesis; Cambridge: Massachusetts Institute of Technology, 2002.
2. *Shahriary H. R., et al.* Network Vulnerability Analysis Thorough Vulnerability Take-Grant Model // Proc. of 7th International Conference on Information and Communications Security. 2005.
3. *Danforth M.* Models for Threat Assessment in Networks: Ph.D. dissertation; Davis: University of California, 2006.

4. *Jajodia S.* Topological Analysis of Network Attack Vulnerability // Managing Cyber Threats: Issues, Approaches and Challenges, 2003.
5. *Sheyner O.* Scenario Graphs and Attack Graphs: Ph.D. dissertation; Carnegie Mellon University. Pittsburgh, 2004.
6. *Девянин П. Н.* Анализ безопасности управления доступом и информационными потоками в компьютерных системах. М.: Радио и связь, 2006. 176 с.
7. *Колегов Д. Н.* ДП-модель компьютерной системы с функционально и параметрически ассоциированными с субъектами сущностями // Вестник Сибирского государственного аэрокосмического университета имени академика М. Ф. Решетнева. 2009. Вып. 1(22). Часть 1. С. 49–54.
8. *Колегов Д. Н.* Анализ безопасности информационных потоков по памяти в компьютерных системах с функционально и параметрически ассоциированными сущностями // Прикладная дискретная математика. 2009. № 1(3). С. 117–126.
9. *Souppaya M. et al.* Technical Guide to Information Security Testing and Assessment. Recommendations of the National Institute of Standards and Technology. 2008.
10. ФСТЭК России. Руководящий документ. Безопасность информационных технологий. Концепция оценки соответствия автоматизированных систем требованиям безопасности информации. М., 2004.
11. *Буренин П. В.* Подходы к построению ДП-модели файловых систем // Прикладная дискретная математика. 2009. № 1(3). С. 93–113.
12. *Касперский К.* Техника сетевых атак. Т. 1: Приемы противодействия. М.: СОЛОН-пресс, 2001. 400 с.
13. *Козиол Дж., Личфилд Д., Эйтэл Д. и др.* Искусство взлома и защиты систем. СПб.: Питер, 2006. 416 с.
14. *Проскурин В. Г., Крутов С. В., Мацкевич И. В.* Защита в операционных системах: Учеб. пособие для вузов. М.: Радио и связь, 2000. 168 с.
15. *McNab C.* Network Security Assessment, second edition. ISBN-10:0-596-51030-6, 2007. 478 p.
16. База уязвимостей SecurityFocus [Электронный ресурс]. Режим доступа: http://www.securityfocus.com/bid.
17. База уязвимостей NVD [Электронный ресурс]. Режим доступа: http://nvd.nist.gov.
18. *Колегов Д. Н.* Применение ДП-моделей для анализа защищенности сетей // Прикладная дискретная математика. 2008. № 1. С. 71–88.

DOI 10.17223/20710410/5/11

УДК 004.056

РЕАЛИЗАЦИЯ МЕТОДА ЗАЩИТЫ УЧЁТНЫХ ДАННЫХ ДЛЯ ДОСТУПА К СУБД В ВЕБ-ПРИЛОЖЕНИЯХ[1]

П. А. Паутов

Томский государственный университет, г. Томск, Россия

E-mail: __Pavel__@mail.ru

В статье рассматриваются особенности аутентификации в веб-приложениях и возникающие при этом проблемы безопасности. Предлагается метод защиты аутентификационных данных для доступа к СУБД. Рассматриваются подходы к реализации метода на языке программирования PHP: с использованием только стандартных средств языка и специального модуля для работы с СУБД, позволяющего получать доступ к установленному соединению с СУБД без имени пользователя и пароля.

Ключевые слова: *многоуровневые приложения, веб-приложения, безопасность веб-приложений, аутентификация в многоуровневом приложении, защита учётных данных для доступа к СУБД, язык программирования PHP.*

Введение

В многоуровневых приложениях при аутентификации одного уровня приложения перед другим возникает проблема защиты хранимых аутентификационных данных каждого уровня, начиная со второго. В работе автора [1] для решения этой проблемы предложено использовать схему с использованием асимметричного шифрования. В данной работе рассматривается реализация этой схемы в трёхуровневом приложении на языке программирования PHP. Результатом реализации является программная библиотека, которую разработчики веб-приложений могут использовать для внедрения рассматриваемой схемы в свои проекты. Рассматриваются два варианта реализации: с использованием 1) только стандартных средств языка PHP и 2) специального модуля для работы с СУБД, позволяющего получать доступ к установленному соединению с СУБД без имени пользователя и пароля. В первом случае учётные данные для доступа к СУБД открываются на прикладном уровне при обработке каждого запроса пользователя, а во втором — только в начале сеанса работы пользователя.

Тезисное изложение результатов этой работы приведено в [2].

1. Многоуровневое приложение

Многоуровневое (или N-уровневое) приложение является приложением, разделённым на N самостоятельных уровней, каждый из которых может выполняться на отдельной платформе. Типичным примером такого подхода является трёхуровневая архитектура, которая подразумевает разделение приложения на следующие уровни:

1) уровень представления — отвечает за представление данных для пользователя;
2) уровень приложения (логики приложения) — отвечает за обработку данных в соответствии с логикой приложения;

[1]Работа выполнена в рамках реализации ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009–2013 годы. Результаты работы докладывались на Международной конференции с элементами научной школы для молодёжи, г. Омск, 7–12 сентября 2009 г.

3) уровень данных — отвечает за хранение данных.

Трёхуровневая архитектура используется в веб-приложениях, где уровни распределены следующим образом:

1) уровень представления — браузер;
2) уровень приложения — программа, исполняемая на веб-сервере;
3) уровень данных — СУБД.

Распределение приложения на несколько изолированных уровней позволяет достичь большей гибкости при масштабировании приложения или смене используемых технологий на каком-либо уровне [3].

2. Аутентификация в трёхуровневом приложении

Применение трёхуровневой архитектуры приводит к появлению нескольких звеньев аутентификации. В классическом двухуровневом приложении клиент аутентифицируется перед сервером. В трёхуровневом приложении имеются два звена аутентификации: клиент аутентифицируется перед прикладным уровнем, а прикладной уровень аутентифицируется перед СУБД. На рис. 1 представлен данный механизм с применением парольной аутентификации.

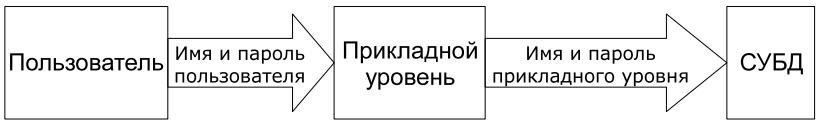

Рис. 1. Два звена аутентификации в веб-приложении

Можно выделить два частных случая (подхода) в организации трёхуровнего приложения:

1) прикладной уровень имеет фиксированный набор пользователей СУБД, от имени которых он может работать;
2) каждому пользователю приложения соответствует пользователь СУБД.

Далее будем рассматривать только первый случай с применением парольной аутентификации. Такой способ организации системы широко распространён в современных веб-приложениях. Чаще всего для взаимодействия с базой данных используется всего один пользователь. Например, так устроены приложения «1С-Битрикс» и «PHP-Nuke» [4, 5]. Самые доступные тарифы хостинг-провайдеров ограничивают количество пользователей для доступа к СУБД [6, 7]. А так как количество пользователей приложения, как правило, превышает заданный лимит пользователей СУБД, то остаётся возможность использовать только первый подход. В этом случае каждому пользователю приложения ставится в соответствие пользователь СУБД с минимально необходимыми привилегиями.

Таким образом, возникает необходимость хранения аутентификационных данных (имён и паролей) пользователей СУБД на прикладном уровне. Во многих современных веб-приложениях эти данные хранятся в открытом виде в некотором файле конфигурации на веб-сервере. При компрометации прикладного уровня злоумышленник может получить доступ к этим данным. В работе [1] рассмотрено несколько схем защиты паролей пользователей СУБД, хранимых на прикладном уровне. Предпочтительной для использования является схема с использованием асимметричного шифра.

3. Схема аутентификации с использованием асимметричного шифра

Каждому пользователю системы ставится в соответствие пара — открытый и закрытый ключи. На прикладном уровне предлагается хранить таблицу следующего вида:

1) имя пользователя приложения;
2) учётные данные для его доступа к СУБД — имя и пароль соответствующего пользователя СУБД, зашифрованные с помощью открытого ключа пользователя приложения;
3) открытый ключ пользователя приложения;
4) закрытый ключ пользователя приложения, зашифрованный с помощью пароля.

Запись, соответствующая администратору приложения, должна содержать имена и пароли всех пользователей СУБД.

Алгоритм аутентификации пользователя перед приложением будет выглядеть следующим образом:

1. Пользователь приложения передаёт свои имя и пароль прикладному уровню.
2. Прикладной уровень находит в описанной выше таблице имя пользователя и с помощью пароля расшифровывает закрытый ключ.
3. Прикладной уровень с помощью закрытого ключа пользователя расшифровывает учётные данные, необходимые для доступа к СУБД.
4. Прикладной уровень устанавливает соединение с СУБД от имени учётной записи, полученной на предыдущем шаге.
5. В случае успешного соединения считается, что пользователь прошёл аутентификацию.

Данная схема позволяет хранить данные для доступа к СУБД в закрытом виде, но, как видно из алгоритма аутентификации, имя и пароль пользователя СУБД на некоторое время появляются в открытом виде в памяти прикладного уровня. Поэтому если злоумышленник имеет возможность перехватывать данные из памяти прикладного уровня, то данная схема только на некоторое время от момента взлома злоумышленником прикладного уровня может отсрочить получение им доступа к СУБД, а именно до того, как очередной пользователь выполнит вход в систему. Если злоумышленник получит доступ, например, только к файлам конфигурации приложения, то он не сможет получить доступ к СУБД. Использование данного метода позволяет сократить время, в течение которого учётные данные для доступа к СУБД будут находиться в открытом виде.

При использовании рассматриваемой схемы усложняются операции по управлению пользователями. Ниже приведены алгоритмы выполнения данных операций.

Смена пароля пользователем.

1. Пользователь присылает свой старый и новый пароли.
2. Прикладной уровень расшифровывает закрытый ключ пользователя с помощью пароля.
3. Прикладной уровень расшифровывает учётные данные СУБД с помощью закрытого ключа.
4. Прикладной уровень устанавливает соединение с СУБД для проверки правильности старого пароля.
5. В случае успеха предыдущего шага прикладной уровень зашифровывает закрытый ключ пользователя на новом пароле.

Создание нового пользователя. Создать нового пользователя приложения может только администратор.

1. Администратор присылает свой пароль и пароль нового пользователя.
2. Прикладной уровень с помощью пароля администратора получает учётные данные СУБД.
3. Прикладной уровень генерирует открытый и закрытый ключи нового пользователя.
4. Прикладной уровень шифрует соответствующие пользователю учётные данные СУБД на закрытом ключе пользователя.
5. Прикладной уровень шифрует закрытый ключ нового пользователя с помощью пароля.

Смена учётной записи СУБД. При смене учётной записи СУБД необходимо с помощью открытых ключей соответствующих пользователей зашифровать новые учётные данные.

Изменение пользователя СУБД для данного пользователя приложения.

1. С помощью пароля администратора расшифровывается закрытый ключ администратора, а с помощью последнего расшифровываются учётные данные СУБД.
2. Учётные данные СУБД шифруются с помощью открытого ключа пользователя.

4. Реализация схемы аутентификации с использованием асимметричного шифра

4.1. Результат реализации

Для реализации данной схемы был выбран язык программирования PHP, так как он являются одним из самых распространённых средств для разработки веб-приложений. Результатом является программная библиотека, которую разработчики веб-приложений могут использовать для внедрения рассматриваемой схемы в свои проекты. Задачи, решаемые программной библиотекой, следующие:

1) реализация рассмотренных алгоритмов управления пользователями;
2) управление сеансами работы пользователей и подключениями к СУБД.

Первая задача была решена с использованием библиотеки OpenSSL и базы данных SQLite. OpenSSL предоставляет необходимые криптографические функции, а модуль SQLite позволяет работать с локальным файлом как с SQL-совместимой базой данных. Так было организовано хранилище учётных данных, описанное в п. 3.

Рассмотрим вторую задачу подробнее. Для этого опишем работу типичного веб-приложения, разработанного с использованием языка программирования PHP.

4.2. Схема веб-приложения

На рис. 2 изображены основные компоненты такого приложения.

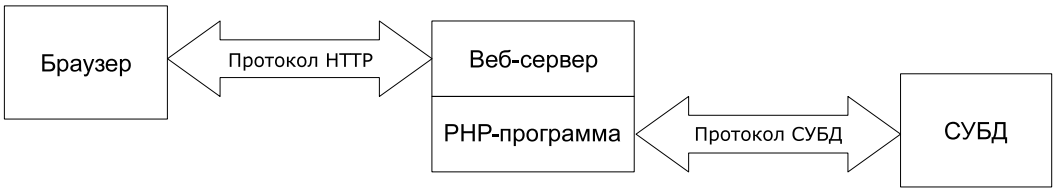

Рис. 2. Компоненты веб-приложения

Браузер посылает веб-серверу HTTP-запрос. Веб-сервер запускает соответствующую PHP-программу. Программа *устанавливает соединение с СУБД*, получает необходимые данные, *закрывает соединение*, оформляет полученные данные в виде HTML-страницы и передаёт последнюю веб-серверу. Веб-сервер отправляет страницу браузеру как ответ на HTTP-запрос.

Протокол HTTP не имеет состояния, т. е. веб-сервер не хранит информации о клиентах между запросами. Однако для того чтобы веб-приложение могло контролировать доступ пользователя и выдавать различное содержимое страниц в зависимости от пользователя, необходимо отслеживать сеанс работы пользователя. Для решения данной задачи применяется механизм сессий. Во время первого обращения пользователя к приложению генерируется некоторый случайный идентификатор (идентификатор сессии), который отсылается браузеру в теле HTTP-ответа. Далее, в теле каждого HTTP-запроса браузер посылает этот идентификатор веб-серверу. PHP-программа ведёт базу данных сессий, в которой для каждого идентификатора хранятся различные данные о пользователе: имя, уровень доступа и т. п.

4.3. Использование только стандартных средств PHP

Для работы описанной схемы веб-приложения прикладному уровню необходимы имя и пароль для доступа к СУБД для обработки каждого запроса пользователя. Так как в рассматриваемой схеме учётные данные СУБД зашифрованы открытым ключом пользователя, а закрытый ключ пользователя зашифрован с помощью пароля пользователя, то для обработки каждого запроса в сеансе прикладному уровню понадобится пароль пользователя. Пароль же предъявляется пользователем только в начале сеанса работы и не должен храниться в течение сеанса. Если же на каждый запрос требовать от пользователя предъявления пароля, то использование операций асимметричной криптографии при обработке каждого запроса может (при большом количестве пользователей) негативно сказаться на производительности системы. Для решения этих проблем после аутентификации пользователя перед приложением выполняются следующие действия:

1. На прикладном уровне генерируется случайный ключ K симметричного шифрования. Этот ключ записывается в сессию текущего пользователя.
2. С помощью ключа K шифруются учётные данные СУБД.
3. Зашифрованные учётные данные СУБД отправляются браузеру в теле HTTP-ответа.

Далее с каждым запросом браузер посылает веб-серверу идентификатор сессии и зашифрованные на ключе K учётные данные для доступа к СУБД. Таким образом, пароль пользователя требуется только в начале сеанса работы и используется для аутентификации пользователя с применением асимметричного шифрования, как описано выше, а во время обработки запросов для получения учётных данных СУБД используется симметричный шифр.

4.4. Использование постоянных соединений с СУБД

Классическая схема работы PHP-программы, когда соединение с СУБД устанавливается и разрывается при обработке каждого запроса пользователя, не всегда является оптимальной. Если установка соединения с СУБД сопровождается большими накладными расходами, то выгоднее поддерживать постоянное соединение и использовать это соединение для обработки запросов. Поэтому в PHP существует механизм работы с постоянными соединениями с СУБД [8]. В стандартном программном интерфейсе PHP для работы с СУБД имя пользователя и пароль необходимы как для того, чтобы

установить новое соединение, так и для того, чтобы получить доступ к уже установленному соединению. Фактически же для работы с уже установленным соединением имя и пароль для СУБД не нужны.

На основе стандартного модуля PHP для СУБД MySQL был реализован модуль, позволяющий получать доступ к уже установленному соединению без имени пользователя и пароля. В этом модуле функция установки соединения с СУБД возвращает идентификатор, который можно сохранить в сессии и использовать для получения соединения с СУБД при обработке последующих запросов.

Заключение

При использовании парольной аутентификации прикладному уровню необходимы имя и пароль пользователя СУБД в открытом виде для установления соединения с СУБД. В большинстве веб-приложений эти данные хранятся на прикладном уровне в открытом виде постоянно. Реализованная схема аутентификации позволяет хранить учётные данные для доступа к СУБД в закрытом виде и получать их в открытом только при необходимости.

Реализация схемы с помощью только стандартных средств языка PHP не требует специального модуля для работы с СУБД и легче переносима на разные платформы. Однако при использовании такой реализации учётные данные для доступа к СУБД открываются на прикладном уровне при обработке каждого запроса пользователя. Подход же с применением специального модуля для работы с СУБД предоставляет более высокий уровень защиты, так как учётные данные для доступа к СУБД открываются только в начале сеанса работы пользователя.

ЛИТЕРАТУРА

1. *Паутов П. А.* Проблема аутентификации в многоуровневых приложениях // Прикладная дискретная математика. 2008. № 2. С. 87–90.
2. *Паутов П. А.* Реализация метода защиты аутентификационных данных в многоуровневых приложениях // Прикладная дискретная математика. Приложение. 2009. № 1. С. 50–51.
3. `www.wikipedia.org` — Википедия. 2009.
4. Руководство по инсталляции «1С-Битрикс: Управление сайтом 6.xx». 2007. `www.1c-bitrix.ru`
5. *Karakas C., Erba C.* PHP-Nuke: Management and Programming. 2005. `www.karakas-online.de/EN-Book`
6. `www.masterhost.ru` — 2009.
7. `www.peterhost.ru` — 2009.
8. *Olson P.* PHP Manual. 2009. `www.php.net/manual`

ПРИКЛАДНАЯ ТЕОРИЯ АВТОМАТОВ

DOI 10.17223/20710410/5/12

УДК 519.713

К ПОСТРОЕНИЮ ПРОВЕРЯЮЩИХ ТЕСТОВ ОТНОСИТЕЛЬНО НЕРАЗДЕЛИМОСТИ ДЛЯ НЕДЕТЕРМИНИРОВАННЫХ АВТОМАТОВ[1]

Е. А. Акеньшина, Н. В. Шабалдина

Томский государственный университет, г. Томск, Россия

E-mail: katarinka_87@list.ru, snv@kitidis.tsu.ru

Предлагается способ улучшения метода построения тестов для недетерминированных автоматов относительно неразделимости. Показывается, что тест, построенный согласно модифицированному методу, будет по-прежнему полным, но при этом менее избыточным.

Ключевые слова: *недетерминированный автомат, отношение неразделимости, полный проверяющий тест.*

Введение

Поведение многих дискретных систем можно описать моделью с конечным числом переходов, например моделью конечного автомата. Для детерминированных автоматов методы построения проверяющих тестов достаточно хорошо развиты. Для недетерминированных автоматов, в которых одной входной последовательности может сопоставляться несколько выходных последовательностей, методы построения тестов активно развиваются, но в основном при тестировании используется предположение «о всех погодных условиях», т. е. предполагается, что есть возможность подавать входную последовательность, пока не пронаблюдаем все выходные реакции на нее. Такое предположение перестает быть реалистичным, если при тестировании нет возможности полностью контролировать проверяемый автомат, что имеет место, например, при удаленном тестировании реализаций телекоммуникационных протоколов. В данной работе предлагается модификация метода построения тестов для недетерминированных автоматов относительно неразделимости для модели «черного ящика», описанного в [1], в которой не используется ограничение «все погодные условия».

1. Основные определения и обозначения

Недетерминированным автоматом называется пятерка $A = (S, I, O, h, s_1)$, где S — множество состояний с выделенным начальным состоянием s_1; I и O — соответственно входной и выходной алфавиты; $h \subseteq S \times I \times S \times O$ — отношение переходов – выходов. Обозначим $out(s, \alpha) = \{\beta : \exists\ s' \in S[(s, \alpha, s', \beta) \in h]$, т. е. $out(s, \alpha)$ есть множество выходных реакций автомата в состоянии s на входную последовательность α.

Состояние s' называется *i-преемником* состояния s, если существует такой выходной символ $o \in O$, что четверка (s, i, s', o) содержится в h. Множество состояний

[1]Работа выполнена при поддержке ФЦП «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007–2012 годы», ГК № 02.514.12.4002.

$M' \subseteq S$ называется *i-преемником* множества состояний $M \subseteq S$, если M' есть множество всех *i-преемников* всех состояний множества M.

При тестировании проверяются различные отношения соответствия между эталонным и проверяемым автоматами.

Пусть A и B — полностью определенные автоматы. Состояние t автомата B называется *редукцией* состояния s автомата A (обозначение: $t \leqslant s$), если $\forall \alpha \in I^*[out(t,\alpha) \subseteq out(s,\alpha)]$. Если $t_1 \leqslant s_1$, то автомат B называется *редукцией* автомата A.

Состояние s автомата A и состояние t автомата B *неразделимы* (обозначение: $s \sim t$), если $\forall \alpha \in I^*[out(s,\alpha) \cap out(t,\alpha) \neq \varnothing]$. Если $\exists \alpha \in I^*\ [out(s,\alpha) \cap out(t,\alpha) = \varnothing]$, то состояния s и t *разделимы по* α (обозначение: $s \not\sim_\alpha t$), или просто *разделимы* (обозначение: $s \not\sim t$). Автоматы A и B *неразделимы*, если $s_1 \sim t_1$. Если $s_1 \not\sim_\alpha t_1$, то автоматы A и B *разделимы по* α (обозначение: $A \not\sim_\alpha B$), или просто *разделимы* (обозначение: $A \not\sim B$); последовательность α называется *разделяющей последовательностью* для автоматов A и B. Разделяющая последовательность $\alpha \in I^*$ называется *кратчайшей*, если любая другая входная последовательность, разделяющая автоматы A и B, не короче α.

2. Построение проверяющих тестов относительно неразделимости

В работе [2] был предложен алгоритм построения кратчайшей разделяющей последовательности для автоматов A и B и показано, что если данные автоматы имеют соответственно не более n и m состояний, то длина кратчайшей разделяющей последовательности будет не более чем 2^{nm-1}, и данная экспоненциальная оценка является достижимой.

Для построения качественных тестов необходима не только формальная модель описания эталонной и проверяемой систем, но и формальное задание модели неисправности. В работе [1] был предложен метод построения полного проверяющего теста относительно неразделимости для модели «черного ящика». Этот метод позволяет строить тесты относительно модели неисправности $\langle A, \sim, \mathfrak{R}_m(A)\rangle$, где область неисправности $\mathfrak{R}_m(A)$ есть множество всех полностью определенных автоматов с теми же входным и выходным алфавитами, что и у A, и числом состояний не более m, разделимых с эталоном или являющихся редукциями эталона, где m — целое положительное число. Отношение конформности $\sim$ предполагает, что полный проверяющий тест обнаружит всякий автомат из области неисправности $\mathfrak{R}_m(A)$, разделимый с эталоном.

В основу рассматриваемого метода положен метод построения разделяющей последовательности для двух автоматов, и при формулировке условия усечения дерева использовалась экспоненциальная оценка длины разделяющей последовательности.

Алгоритм 1. Построение полного проверяющего теста относительно модели неисправности $\langle A, \sim, \mathfrak{R}_m(A)\rangle$.

Вход: Полностью определенный автомат A и верхняя граница m числа состояний автоматов из области неисправности $\mathfrak{R}_m(A)$.

Выход: Полный проверяющий тест TS относительно модели неисправности $\langle A, \sim, \mathfrak{R}_m(A)\rangle$.

Шаг 1. Построим усеченное дерево преемников автомата спецификации A. Корнем дерева на нулевом уровне является начальное состояние s_1 автомата A; вершины дерева помечены подмножествами состояний автомата A. Пусть уже построены j уровней дерева, $j \geqslant 0$. Для заданной нетерминальной вершины j-го уровня, помеченной подмножеством состояний K, и для заданного входного символа i в дереве есть ребра, помеченные символом i, в вершину, помеченную i-преемниками подмножества K.

Текущая вершина $Current$ на k-м уровне, $k > 0$, помеченная подмножеством K состояний из множества S, объявляется листом дерева, если путь из корня в эту вершину содержит $2^{|K|\cdot m}$ вершин, помеченных подмножествами множества K, и начальное состояние s_1 не содержится в K. Если начальное состояние принадлежит K, то вершина $Current$ объявляется листом, если путь из корня в эту вершину покрывает $(2^{|K|\cdot m-1}+1)$ вершин, помеченных подмножествами множества K.

Шаг 2. Включаем в TS каждую входную последовательность, которая помечает путь из корня к листу в усеченном дереве.

В качестве примера рассмотрим спецификацию A, представленную на рис. 1, и построим полный проверяющий тест относительно модели неисправности $\langle A, \sim, \mathfrak{R}_2(A)\rangle$.

A	a	b
x	$a/0, 1, 2, 3$	$a/1, 2$
y	$b/1, 2$	$a/0$ $b/3$

Рис. 1. Автомат A

Полученное по данному алгоритму усеченное дерево преемников представлено на рис. 2. Суммарная длина полного проверяющего теста составляет 277 символов.

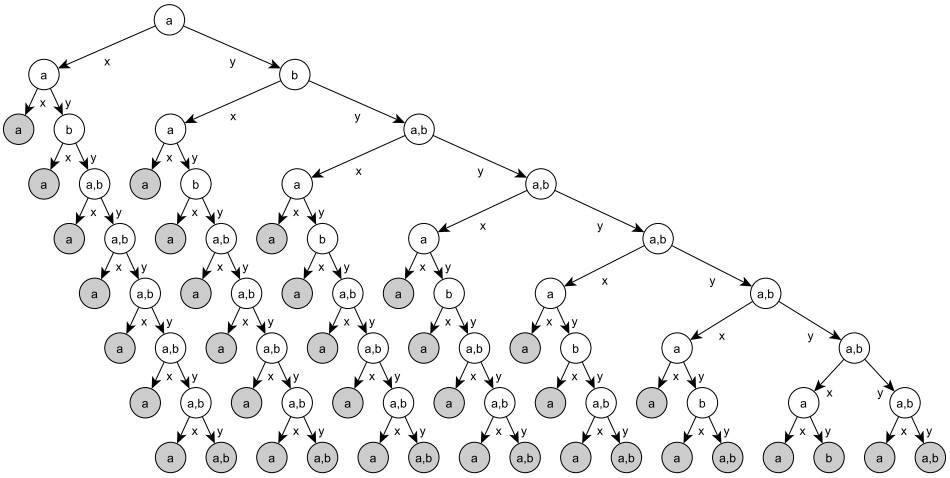

Рис. 2. Усеченное дерево преемников, построенное по алгоритму 1

3. Модификация метода построения проверяющих тестов относительно неразделимости

Алгоритм 1 не доставляет кратчайшего теста. Для иллюстрации этого факта рассмотрим тестовую последовательность $xyyyyyyy$ из предыдущего примера, которой в усеченном дереве преемников (рис. 2) соответствует путь $a_x\ a_y\ b_y\ \{a,b\}_y\ \{a,b\}_y\ \{a,b\}_y\ \{a,b\}_y\ \{a,b\}_y\ \{a,b\}$. Прямым перебором можно убедиться, что если автомат-реализация имеет состояния 1 и 2, то соответствующий путь в усеченном дереве $Tree_{A\cap B}$, построенном по пересечению эталонного автомата и реализации, будет уже усечен по-

сле $\{a1\}_x\{a2\}_y\{b1,b2\}_y\{b1\}_y\{b2\}_y\{a,b\}$, так как для подмножества a были перебраны все варианты и из последующих подмножеств его можно исключить. Таким образом, при уточнении условий усечения дерева данную тестовую последовательность можно сократить на три символа.

Сократить тестовую последовательность можно также и в более общем случае, когда на рассматриваемом пути дерева перебраны все возможные варианты для состояний некоторого множества P, являющегося подмножеством множества K, или для нескольких подмножеств P_i множества K, в том числе и в случае, когда эти подмножества пересекаются.

Таким образом, можно модифицировать метод построения полного проверяющего теста относительно модели неисправности $\langle A, \sim, \mathfrak{R}_m(A)\rangle$, уточнив условия усечения дерева преемников.

Алгоритм 2. Построение полного проверяющего теста относительно модели неисправности $\langle A, \sim, \mathfrak{R}_m(A)\rangle$.

Вход: Полностью определенный автомат A и верхняя граница m числа состояний автоматов из области неисправности $\mathfrak{R}_m(A)$.

Выход: Полный проверяющий тест TS относительно модели неисправности $\langle A, \sim, \mathfrak{R}_m(A)\rangle$.

Шаг 1. Построим усеченное дерево преемников автомата спецификации A. Корнем дерева на нулевом уровне является начальное состояние s_1 автомата A; вершины дерева помечены подмножествами состояний автомата A. Пусть уже построены j уровней дерева, $j \geqslant 0$. Для заданной нетерминальной вершины j-го уровня, помеченной подмножеством состояний K, и для заданного входного символа i в дереве есть ребра, помеченные символом i, в вершину, помеченную i-преемниками подмножества K. Текущая вершина $Current$ на k-м уровне, $k > 0$, помеченная подмножеством K состояний из множества S, объявляется листом дерева, если выполняется любое из перечисленных ниже условий усечения, т. е. если путь из корня в вершину $Current$ содержит:

1) $2^{|K|\cdot m}$ вершин, помеченных подмножествами множества K, если $s_1 \notin K$;
2) $2^{|K|\cdot m-1}+1$ вершин, помеченных подмножествами множества K, если $s_1 \in K$;
3) $2^{(|K|-|P|)\cdot m}+n$ вершин, помеченных подмножествами множества K, если $s_1 \notin K$ или $s_1 \in K$ и $s_1 \in P$;
4) $2^{(|K|-|P|)\cdot m-1}+n+1$ вершин, помеченных подмножествами множества K, если $s_1 \in K$ и $s_1 \notin P$.

Здесь P — это такое подмножество состояний из множества K, что до некоторого l-го уровня ($l < k$) перебраны все возможные подмножества P, а n — это количество вершин на данном пути, помеченных подмножествами K, содержащими подмножества P, которые находятся на уровнях не ниже l-го уровня дерева (если $P = \varnothing$, то $n = 0$). Множество P строится итеративно:

1) $P = \varnothing$;
2) $P = P \cup P_j$ для каждого подмножества P_j множества K, такого, что $P_j \cap P = Q$ и путь из корня в вершину $Current$ содержит $(2^{(|P_j|-|Q|)\cdot m}-1)$ вершин, помеченных подмножествами P_j, если $s_1 \notin P_j$ или $s_1 \in Q$ (для $Q \neq \varnothing$), либо $(2^{(|P_j|-|Q|)\cdot m-1})$ вершин в случае $s_1 \in P_j$.

Шаг 2. Включаем в TS каждую входную последовательность, которая помечает путь из корня к листу в усеченном дереве.

Теорема 1. Для заданного эталонного автомата A и целого числа m алгоритм 2 доставляет полный проверяющий тест относительно модели неисправности $\langle A, \sim, \mathfrak{R}_m(A)\rangle$.

Доказательство. Рассмотрим самый общий случай — подмножество K состояний из множества S и начальное состояние $s_1 \notin K$. Согласно алгоритму 1, который доставляет полный проверяющий тест относительно модели неисправности $\langle A, \sim, \mathfrak{R}_m(A)\rangle$, вершина усеченного дерева преемников $Tree_A$, помеченная подмножеством K, объявляется листом дерева, если путь из корня в эту вершину содержит $2^{|K|\cdot m}$ вершин, помеченных подмножествами множества K. Это соответствует перебору всех возможных подмножеств K' в усеченном дереве преемников $Tree_{A\cap B}$, построенному по пересечению эталонного автомата A и некоторой реализации B, где K' — это подмножество состояний пересечения $A \cap B$, таких, что первый символ каждой пары из K' содержится в K.

Если вершина, помеченная подмножеством K, не будет объявлена листом согласно условию 1 усечения дерева преемников, то, возможно, для данной вершины выполняется условие 3, т. е. существует такое подмножество состояний P из множества K, что до некоторого l-го уровня дерева на пути из корня в вершину $Current$ перебраны все возможные подмножества P. Множество P итеративно объединяется с каждым подмножеством P_j множества K, таким, что $P_j \cap P = Q$ и путь из корня в вершину $Current$ содержит $(2^{(|P_j|-|Q|)\cdot m} - 1)$ вершин, помеченных подмножествами P_j. Добавление каждого такого P_j в множество P справедливо, так как если путь содержит указанное количество повторов, то тем самым перебраны все возможные варианты подмножеств P_j.

Для построенного таким образом P (если $P \neq \varnothing$) на соответствующем пути в дереве $Tree_{A\cap B}$ будут перебраны все возможные подмножества P' (P' — это подмножество состояний $A \cap B$, таких, что первый символ каждой пары из P' содержится в P). Значит, далее на данном пути в дереве $Tree_{A\cap B}$ из рассмотрения можно исключить вершины, помеченные подмножествами K', которые содержат подмножества P'; поэтому рассматриваемых вершин, помеченных подмножествами K, не содержащих подмножеств P, будет $2^{(|K|-|P|)\cdot m}$. Но также необходимо учесть все n вершин, помеченных подмножествами K, содержащими подмножества P, которые встретились на рассматриваемом пути в дереве $Tree_A$ выше, чем l-й уровень, так как из данных вершин подмножества P исключать не можем. Следовательно, количество вершин, помеченных подмножествами K, для усечения дерева в таком случае составляет $2^{(|K|-|P|)\cdot m} + n$ вершин.

Далее рассмотрим случай, когда $s_1 \in K$. По алгоритму 1 вершина, помеченная подмножеством K, объявляется листом, если путь из корня в данную вершину содержит $(2^{|K|\cdot m-1} + 1)$ вершин, помеченных подмножествами множества K, что соответствует условию усечения 2 алгоритма 2. Если вершина, помеченная подмножеством K, не будет объявлена листом согласно данному условию, то для случая $s_1 \in K$ и $s_1 \notin P$ может иметь место выполнение условия усечения 4. То есть если на данном пути для каждого $P_j \subseteq P$ встретилось, как и в предыдущем случае, необходимое число вершин, помеченных подмножествами P_j, то листовой будет являться вершина, путь из корня в которую содержит $2^{(|K|-|P|)\cdot m-1} + n + 1$ вершин, помеченных подмножествами K.

Если $s_1 \in K$ и $s_1 \in P$, т. е. s_1 принадлежит одному из подмножеств $P_j \subseteq P$, то необходимо, чтобы на рассматриваемом пути дерева встретилось $(2^{(|P_j|-|Q|)\cdot m-1})$ вершин, помеченных подмножествами P_j. Для остальных подмножеств P_j из P требуется

встретить такое же количество вершин, как и в первом случае. Тогда при подсчете вершин, помеченных подмножествами K, можно исключить из рассмотрения подмножества P начиная с l-го уровня, и вершина, помеченная подмножеством K, объявляется листом, если путь из корня в данную вершину содержит $2^{(|K|-|P|)\cdot m} + n$ вершин, помеченных подмножествами множества K (условие усечения 3). ■

На рис. 3 представлено усеченное дерево преемников для спецификации, изображенной на рис. 1, построенное согласно алгоритму 2. Суммарная длина полного проверяющего теста в этом случае составляет 194 символа, что на 83 символа меньше, чем для алгоритма 1.

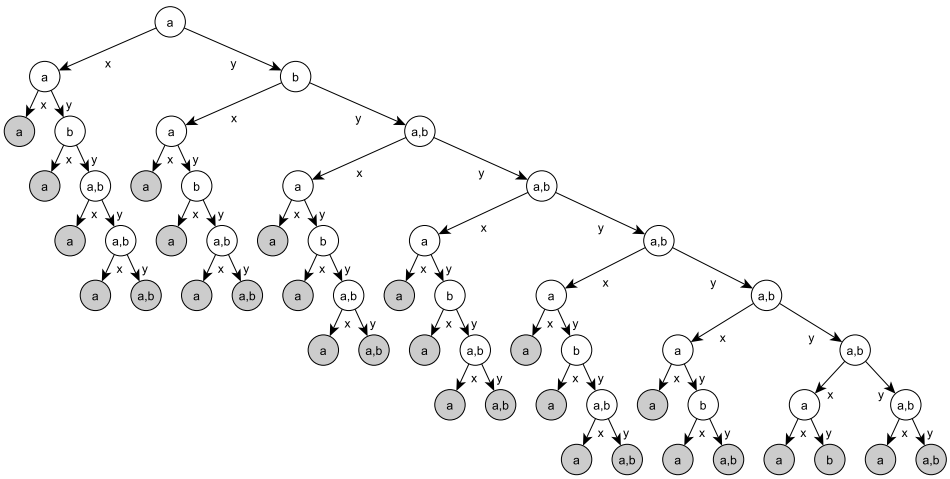

Рис. 3. Усеченное дерево преемников, построенное по алгоритму 2

Заключение

В данной работе предложена модификация метода построения тестов для недетерминированных автоматов относительно неразделимости, описанного в работе [1]. Эта модификация, в отличие от других методов синтеза тестов для недетерминированных автоматов, не ориентирована на выполнение предположения «о всех погодных условиях». Показано, что тест, построенный согласно модифицированному методу, будет по-прежнему полным, но при этом в большинстве случаев менее избыточным.

ЛИТЕРАТУРА

1. *Shabaldina N., El-Fakih K. and Yevtushenko N.* Testing Nondeterministic Finite State Machines With Respect to the Separability Relation // Lect. Not. Comp. Sci. 2007. V. 4581. P. 305–318.
2. *Евтушенко Н. В., Спицына Н. В.* О верхней оценке длины разделяющей последовательности // Вестник Томского госуниверситета. Приложение. Сер. Математика. Кибернетика. Информатика. 2006. № 18. С. 54–58.

МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ

DOI 10.17223/20710410/5/13

УДК 519.682

АНАЛИТИЧЕСКИЙ ПОДХОД В ТЕОРИИ КОНТЕКСТНО-СВОБОДНЫХ ЯЗЫКОВ В НОРМАЛЬНОЙ ФОРМЕ ГРЕЙБАХ[1]

О. И. Егорушкин*, К. В. Сафонов**

*Красноярский государственный аграрный университет,
**Сибирский государственный аэрокосмический университет
имени академика М. Ф. Решетнева, г. Красноярск, Россия

E-mail: safonovkv@rambler.ru

Контекстно-свободные языки рассматриваются как формальные степенные ряды, являющиеся решением системы полиномиальных уравнений с некоммутативными относительно умножения переменными. Предложено изучать эти системы в нормальной форме Грейбах, что позволит более эффективно использовать аналитические методы. Рассматриваются коммутативные образы контекстно-свободных языков и определяющих их систем уравнений в комплексной области.

Ключевые слова: *контекстно-свободные языки, нормальная форма Грейбах.*

Формальным языком L называют множество цепочек в алфавите $\{x_1, \ldots, x_n\}$, выделенных с помощью конечного набора правил. Такие цепочки, принадлежащие свободной полугруппе $\{x_1, \ldots, x_n\}^*$, называются грамматически правильными предложениями (над словарем), а конечное множество правил, с помощью которых выделяются эти цепочки, называют грамматикой. Таким образом, формальный язык определяется совокупностью соответствующих правил и способом выделения цепочек с помощью этих правил.

Практически важный класс формальных языков образуют контекстно-свободные языки (кс-языки), являющиеся мощным средством моделирования естественных языков и языков программирования [1–3]. Обозначим $W = \{x_1, \ldots, x_n\} \cup \{z_1, \ldots, z_m\}$ свободную полугруппу с операцией умножения над «расширенным» алфавитом $\{x_1, \ldots, x_n, z_1, \ldots, z_m\}$.

Словарем языка является конечное множество $X = \{x_1, \ldots, x_n\}$ слов языка, называемое терминальным множеством, а $Z = \{z_1, \ldots, z_m\}$ — множество вспомогательных символов z_j, необходимых для задания грамматических правил, называемое нетерминальным множеством; $W^* = (X \cup Z)^*$ — соответствующая свободная полугруппа относительно операции конкатенации. Дополним её операцией формального сложения «+» мономов из множества W^* (вместо суммы можно взять объединение «∪», следуя [1]), а также коммутативной операцией умножения мономов на (целые) числа. Таким образом, можно рассматривать не только многочлены, но и формальные степенные ряды с числовыми (как правило, целыми) коэффициентами от некоммутативных переменных.

[1]Результаты работы докладывались на Международной конференции с элементами научной школы для молодёжи, г. Омск, 7–12 сентября 2009 г.

Кс-грамматика есть совокупность правил подстановки, которые каждому нетерминальному символу z_j ставят в соответствие некоторые мономы от терминальных и нетерминальных символов:

$$z_j \to f_{j1}(x,z), \ldots, z_j \to f_{jq_j}(x,z),$$

причем z_1 — особый символ, играющий роль начала предложения в языке. Применение к нему произвольного числа подстановок порождает всевозможные правильные предложения языка.

Важной задачей теоретической информатики и ее приложений в теоретическом программировании является изучение структуры кс-языков и ее связи со структурой порождающих грамматик.

Правилам подстановки ставится в соответствие [3] система полиномиальных уравнений: каждому вспомогательному символу z_j, содержащемуся в левой части подстановки, сопоставляется уравнение $z_j = p_j(x,z)$, где $p_j(x,z) = f_{j1}(x,z) + \ldots + f_{jq_j}(x,z)$. Таким образом, кс-грамматике соответствует система полиномиальных уравнений

$$z_j = p_j(x,z), j = 1, \ldots, m, \tag{1}$$

называемая системой уравнений Хомского — Щютценберже, а кс-языком называется [3, 4] первая компонента z_1 ее решения $(z_1(x), \ldots, z_m(x))$, получаемого методом последовательных приближений

$$z_j^{(k+1)} = p_j(x, z^{(k)}), z^{(0)} = (0, \ldots, 0), j = 1, \ldots, m,$$

где $z^{(k)} = (z_1^{(k)}, \ldots, z_m^{(k)})$, 0 — нулевой моном, такой, что $0 \cdot u = u \cdot 0 = 0$ для любого монома u.

В результате итераций компоненты z_j выражаются формальными степенными рядами, причем кс-язык есть тот из них, который представляет выделенный символ z_1:

$$z_1 = \sum_i < z_1, w_i > w_i. \tag{2}$$

Здесь $< z_1, w_i >$ — числовой коэффициент при мономе w_i от некоммутативных переменных $x_1, \ldots, x_n$ (формальная сумма всех мономов и образует кс-язык). Условие того, что язык является контекстно-свободным, состоит также и в том, что многочлены $p_j(x,z)$ не содержат мономов z_j и e, где e — пустая цепочка. Таким образом, существование и единственность решения системы (1) обеспечивается её специфической структурой и методом последовательных приближений.

Система уравнений Хомского — Щютценберже впервые изучалась аналитическими методами в работах К. В. Сафонова (напр., [3]). Предложенный им подход состоял в изучении систем полиномиальных уравнений произвольной степени от многих переменных в комплексной области, что, естественно, сопряжено со значительными трудностями. В настоящей работе предлагается изучать вместо системы уравнений Хомского — Щютценберже эквивалентную ей систему в нормальной форме Грейбах [5, с. 127], в которой все многочлены имеют вторую степень по переменной z; переход к нормальной форме Грейбах всегда возможен за счет увеличения, быть может, числа переменных. Для уравнений второй степени аналитические методы исследования представляются более перспективными: их эффективность в большей мере зависит от степени уравнений, чем от числа переменных.

Более точно, нормальная форма Грейбах системы (1) подразумевает, что все входящие в нее многочлены имеют вид $p_j(x,z) = f_j(z) + g_j(x,z) + h_j(x)$, где $f_j(z)$ — квадратичная форма от переменных $z_1, \ldots, z_m$, многочлен $g_j(x,z)$ линеен по $z_1, \ldots, z_m$, а многочлен $h_j(x)$ от них не зависит [5, с. 127]. А значит, можно считать, что система (1) имеет вид

$$z_j = f_j(z) + g_j(x,z) + h_j(x) = 0, j = 1, \ldots, m. \qquad (1^*)$$

Нам понадобится понятие коммутативного образа формального ряда (многочлена) [2, 3]. А именно, поставим в соответствие формальному степенному ряду (многочлену) ряд (многочлен) с комплексными переменными, задав отображение терминальных x_i и нетерминальных z_i символов из множества $X \cup Z$ в множество комплексных переменных, оставив за ними прежние обозначения, тогда далее $(x,z) \in C^{m+n}_{x,z}$. Таким образом, получаем фиксированный гомоморфизм, который ставит в соответствие формальному ряду r его *коммутативный образ* — сходящийся в окрестности нуля степенной ряд от комплексных переменных

$$ci(r) = \sum_k a_k x^k, \qquad (3)$$

где

$$a_k x^k = a_{k_1,\ldots,k_n} x_1^{k_1} \ldots x_n^{k_n},$$

$$a_k = \sum_{\sharp x_1(w_i)=k_1,\ldots,\sharp x_n(w_i)=k_n} < r, w_i >, \qquad (4)$$

символ $\sharp c(d)$ означает число вхождений символа c в моном d.

Итак, рассмотрим систему уравнений (1*). Если в ней $f_j(z) \equiv 0$ при всех j, то эта система линейная и порождаемый ею язык принадлежит к простейшему классу линейных языков [1, 2]. Однако важно, что существует связь между кс- и линейными языками, точнее, между их коммутативными образами.

Рассмотрим коммутативный образ кс-языка z_1

$$ci(z_1) = \sum a_k x^k = \sum a_{k_1,\ldots,k_n} x_1^{k_1} \ldots x_n^{k_n}, \qquad (5)$$

который является алгебраической функцией, а также коммутативный образ

$$ci(L) = \sum_{k_0 \geqslant 0, k \geqslant 0} L_{k_0,k} x_0^{k_0} x^k \qquad (6)$$

некоторого линейного языка над терминальными символами $x_0, x_1, \ldots, x_n$, который является рациональной функцией.

Будем называть ряд (5) *диагональю* ряда (6), если при всех $k_1, \ldots, k_n$ выполнено условие

$$a_{k_1,\ldots,k_n} = L_{k_1,k_1,k_2,\ldots,k_n},$$

и в следующей теореме установим, для каких именно кс-языков коммутативные образы являются диагоналями линейных языков.

Теорема 1. Если однородные многочлены второй степени $f_2(z), \ldots, f_m(z)$, входящие в систему (1*), не зависят от переменной z_1 и система уравнений

$$f_j(z) = 0, j = 2, \ldots, m, \qquad (7)$$

имеет единственный нуль $z = 0$, то коммутативный образ кс-языка, порожденного системой (1*), является диагональю некоторого линейного языка.

Учитывая относительную простоту структуры линейных языков и конструкции диагонали, можно сделать вывод о том, что структура широкого класса кс-языков, описанных в теореме 1, является достаточно простой.

Доказательство. Рассмотрим корни системы (1*) следующим образом: будем считать переменные x и z_1 комплексными параметрами, а остальные корни $z[1] = (z_2, \ldots, z_m) = z[1](x, z_1)$ зависящими от этих параметров. Условие теоремы 1 означает, что ни при каких значениях параметров корни системы (1*) не стремятся к бесконечности (как это бывает, например, когда старший коэффициент многочлена при некотором значении параметра обращается в нуль). Следовательно, при всех значениях параметров x и z_1 число корней $z^{(k)}[1](x, z_1)$ подсистемы $p_j(x, z) = 0, j = 2, \ldots, m$, с учетом кратностей μ_k одно и то же, а потому возможно корректно определить результант этой подсистемы относительно уравнения $p_1(x, z) = 0$:

$$P(x, z_1) = \prod_k p_1^{\mu_k}(x, z_1, z^{(k)}[1](x, z_1)),$$

где в произведение входят все корни уравнения. В силу симметричной зависимости этой функции от корней и алгебраичности самих корней результант также является симметричной алгебраической функцией, т. е. многочленом.

Согласно теореме о результанте системы [6, с. 238], кратность корня $(0, 0)$ системы (1*), равная 1, совпадает с кратностью корня $z_1 = 0$ многочлена $P(x, z_1)$ при $x = 0$. А значит, для многочлена $P(x, z_1)$ выполняется условие

$$P(0, 0) = 0, \frac{\partial P(0, 0)}{\partial z_1} \neq 0.$$

Отсюда следует, что алгебраическая функция z_1, определяемая многочленом P, является диагональю некоторой рациональной функции от переменных $x_0, x_1, \ldots, x_n$ (см. [7]). Поскольку каждая голоморфная в нуле рациональная функция является коммутативным образом соответствующего линейного языка, теорема 1 доказана. ■

Порождающая кс-язык система уравнений (1*) в нормальной форме Грейбах позволяет установить и другие свойства кс-языков, например, является данный формальный степенной ряд контекстно-свободным языком или нет. Сгруппируем коммутативный образ формального языка в ряд по однородным многочленам:

$$ci(z_1) = \sum_j (a)_j(x), \tag{8}$$

где $(a)_j(x)$ — однородный многочлен степени j. Достаточно установить, что ряд (8) удовлетворяет уравнению степени 2, поскольку такую степень имеет система (1*). Возводя ряд (8) в квадрат, получим ряд $(ci(z_1))^2 = \sum_j (a)_j^{(2)}(x)$. Тот факт, что оба ряда, будучи умноженными на многочлены, дают в сумме тождественный нуль, означает, что достаточно длинные отрезки этих рядов линейно зависимы. Таким образом, получается следующая теорема.

Теорема 2. Для того чтобы ряд по однородным многочленам (8) удовлетворял полиномиальному уравнению степени 2, необходимо и достаточно, чтобы при всех

$j \geqslant j_0$, $l \geqslant l_0$ выполнялось равенство

$$\begin{vmatrix} (a)_j(x) & \dots & (a)_{j+l}(x) & (a)^{(2)}_j(x) & \dots & (a)^{(2)}_{j+l}(x) \\ (a)_{j+1}(x) & \dots & (a)_{j+l+1}(x) & (a)^{(2)}_{j+1}(x) & \dots & (a)^{(2)}_{j+l+1}(x) \\ \dots & \dots & \dots & \dots & \dots & \dots \\ (a)_{j+q}(x) & \dots & (a)_{j+l+q}(x) & (a)^{(2)}_{j+q}(x) & \dots & (a)^{(2)}_{j+l+q}(x) \end{vmatrix} \equiv 0,$$

где $q = 2l + 1$.

Таким образом, можно сделать вывод о том, что нормальная форма Грейбах систем уравнений, порождающих кс-языки, имеет ряд преимуществ, связанных с тем, что эти уравнения имеют степень не выше второй. Так, нормальная форма Грейбах позволяет установить важную связь кс-языков с линейными языками (теорема 1), а также получить фундаментальное условие, характеризующее кс-языки (теорема 2).

ЛИТЕРАТУРА

1. *Глушков В. М., Цейтлин Г. Е., Ющенко Е. Л.* Алгебра, языки, программирование. Киев: Наук. думка, 1974. 328 с.
2. *Семенов А. Л.* Алгоритмические проблемы для степенных рядов и контекстно-свободных грамматик // Докл. АН СССР. 1973. Т. 212. С. 50–52.
3. *Сафонов К. В.* О возможности вычислительного распознавания контекстно-свободных языков // Вычислительные технологии. 2005. Т. 10. № 4. С. 91–98.
4. *Сафонов К. В., Егорушкин О. И.* О синтаксическом анализе и проблеме В. М. Глушкова распознавания контекстно-свободных языков Хомского // Вестник Томского госуниверситета. Приложение. 2006. № 17. С. 63–66.
5. *Salomaa A., Soitolla M.* Automata-Theoretic Aspects of Formal Power Series. N.Y.: Springer Verlag, 1978. 176 p.
6. *Айзенберг Л. А., Южаков А. П.* Интегральные представления и вычеты в многомерном комплексном анализе. Новосибирск: Наука, 1979. 366 с.
7. *Safonov K. V.* On Power Series of Algebraic and Rational Functions in C^n // J. Mathematical Analysis and Applications. 2000. V. 243. P. 261–277.

2009 Математические основы информатики и программирования №3(5)

DOI 10.17223/20710410/5/14

УДК 004.94

МЕТОД АВТОМАТИЗИРОВАННОГО ПОИСКА ПРОГРАММНЫХ ОШИБОК В АЛГОРИТМАХ ОБРАБОТКИ СЛОЖНОСТРУКТУРИРОВАННЫХ ДАННЫХ[1]

А. Н. Макаров

ФГНУ «ГНТЦ „Наука“», г. Москва, Россия

E-mail: byalex@mail.ru

Рассмотрен разработанный автором метод поиска программных ошибок в программном обеспечении при отсутствии исходных текстов. Метод основан на стрессовом тестировании совместно с автоматической трассировкой программного обеспечения. При этом наборы тестовых данных могут формироваться статически или динамически на основе результатов трассировки.

Ключевые слова: *компьютерная безопасность, тестирование, программная ошибка.*

Введение

При разработке программных комплексов часто решается задача по интеграции в них программных модулей, для которых отсутствуют исходные коды и сопроводительная техническая документация. Для обеспечения надежности функционирования комплекса в целом может потребоваться выполнить анализ бинарного кода используемых модулей.

В данном случае под анализом бинарного кода подразумевается проверка корректности работы программного обеспечения (ПО) и отсутствия программных ошибок. Существуют различные классификации ошибок, встречающихся в ПО [1, 2]. Далее считаем, что *программная ошибка* — это ошибка реализации ПО, допущенная разработчиками на этапе кодирования. Проявлением программной ошибки является аварийное завершение процесса, связанного с соответствующим ПО.

Применяют несколько основных подходов, которым отдается предпочтение при решении задачи тестирования бинарного кода [3, 4]:

— обратная инженерия (*reverse engineering*) — применяется с целью получения ПО на языке ассемблера или на языке высокого уровня;
— анализ двоичного кода — предполагает наличие анализирующего приложения, которое читает собранное ПО и просматривает его с применением некоторых эвристических правил;
— тестирование нагрузкой или стрессовое тестирование — используется набор файлов сценариев, которые посылают ПО разнообразные входные данные различного размера и структуры.

В последнее время получил распространение один из видов стрессового тестирования — фазинг (*fuzzing*) [1, 5, 6]. Его преимущество — возможность быстрого получения результатов. Выделяют два вида фазинга — мутационный (*mutation fuzzing*) и генерационный (*generation fuzzing*). В мутационном фазинге предполагается внесение

[1] Результаты работы докладывались на Международной конференции с элементами научной школы для молодёжи, г. Омск, 7–12 сентября 2009 г.

искажений в обрабатываемые входные данные, тогда как генерационный фазинг предполагает создание входных данных на основе определенных правил.

На основе мутационного фазинга автором был разработан метод автоматизированного поиска программных ошибок (АППО), обладающий следующими свойствами:

— метод ориентирован на применение в условиях отсутствия исходного кода исследуемого ПО;
— метод предназначен для выявления ошибок кодирования («программистских» ошибок) ПО, которые, как правило, трудно выявить, основываясь на формальных методах верификации ПО;
— метод является динамическим (анализ исследуемого ПО проводится во время его работы, т. е. исследованию подвергается не само ПО, а процесс, порожденный ПО);
— метод применим для тестирования широкого спектра ПО, например ПО или программных компонент, обрабатывающих файлы с нетривиальной внутренней структурой;
— метод не ориентирован на применение в рамках какой-либо конкретной аппаратно-программной платформы.

Предлагаемый метод представляет сложноструктурированные данные в виде приведенной ниже модели структур сущностей-данных.

1. Модель структур сущностей-данных

Опишем модель структур сущностей-данных, которую используем при описании метода АППО. Дадим определение.

Определение 1. ПО, для которого проводится анализ корректности реализации с помощью метода АППО, будем называть исследуемым ПО.

Сущности-данные, обрабатываемые исследуемым ПО, как правило, имеют внутреннюю структуру (структуру данных). При этом даже в случае, когда ПО обрабатывает явно неструктурированные данные (поток данных), можно считать, что данные имеют структуру вида последовательности байт (бит или слов). С использованием [7] дадим определения и сделаем предположение.

Определение 2. *Структура* — иерархическая организация взаимосвязанных данных, обладающая свойствами, которые позволяют выделить структуру среди прочих и отнести ее к определенному типу. Структуры, относящиеся к одному типу, по определению обладают одинаковой организацией взаимосвязи данных.

Определение 3. *Элементарная структура* — один байт. По определению каждая структура, кроме элементарной, в соответствии с ее иерархической организацией состоит из вложенных в нее структур. Значение каждой структуры представляется последовательностью (строкой) байт. При этом длину данной последовательности назовем размером структуры.

Определение 4. *Тип структуры* — атрибут структуры, однозначно задающий множество значений, которые могут принимать структуры данного типа, и набор операций, выполнимых над ними.

Определение 5. *Обработчик структуры* — совокупность кода исследуемого ПО, осуществляющего обработку структуры. Обработчик однозначно идентифицирует

структуру. Структуру назовем *известной* в случае, когда известен ее обработчик (полностью описаны алгоритмы, реализованные в исследуемом ПО для обработки данной структуры).

Замечание 1. Если у двух структур одинаковый обработчик, то эти структуры имеют одинаковый тип. Две однотипные структуры могут иметь различные обработчики, и, следовательно, структуры следует считать различными.

Предположение 1. Каждой сущности-данным, обрабатываемой исследуемым ПО, соответствует некоторая структура.

Используем следующие обозначения:

Ω — множество структур, обрабатываемых исследуемым ПО;

Ω^* — множество всех конечных последовательностей структур из множества Ω;

$\Omega_{\text{know}} \subset \Omega$ — множество известных структур;

T — множество типов структур;

$byte \in T$ — тип элементарной структуры;

$\Lambda \notin T$ — специальный символ, означающий, что тип структуры не может быть определен (исследуемое ПО может обрабатывать сущности-данные, содержащие структуры, определение типов которых затруднено);

$\Omega_{\text{type}} \subset \Omega$ — множество структур, тип которых определен;

Θ — множество обработчиков структур в исследуемом ПО (точное описание элементов данного множества не является существенным; потенциально оно может быть получено с использованием методов обратной инженерии);

$\Delta \notin \Theta$ — специальный символ, означающий, что обработчик структуры неизвестен (отсутствует полное описание алгоритмов, реализованных в исследуемом ПО для обработки некоторой структуры);

$B = \{0, \ldots, 255\}$ — множество значений элементарной структуры;

$B^n = \{(b_1, \ldots, b_n)\}$ — множество всех последовательностей длины $n \geqslant 1$, где $b_i \in B$;

$B^* = \bigcup_{n \geqslant 1} B^n$ — множество всех последовательностей конечной длины из элементов в B.

Определение 6. Определим $\lambda : \Omega \to \mathbb{N}$ — функцию, задающую размер структур, где $\mathbb{N}$ — множество натуральных чисел, и $\upsilon : \Omega \to B^*$ — функцию, задающую значение структур. При этом по определению каждое значение структуры $\alpha \in \Omega$ задается последовательностью из множества $B^{\lambda(\alpha)}$.

Определение 7. Определим $\tau : \Omega \to T \cup \{\Lambda\}$ — функцию, задающую типы структур, $\omega : \Omega \to \Theta \cup \{\Delta\}$ — функцию, задающую для каждой структуры ее обработчик; $\sigma : \Theta \to \mathbb{N}$ — функцию, задающую для каждого обработчика объем кода (в тысячах строк кода, ТСК). По определению выполняются следующие условия:

— для двух структур $\alpha_1, \alpha_2 \in \Omega$ справедливо равенство $\alpha_1 = \alpha_2$ тогда и только тогда, когда справедливо равенство $\omega(\alpha_1) = \omega(\alpha_2)$;
— структура $\alpha \in \Omega$ является известной ($\alpha \in \Omega_{\text{know}}$) тогда и только тогда, когда выполняется условие $\omega(\alpha) \neq \Delta$;
— если у структуры $\alpha \in \Omega$ не определен тип ($\tau(\alpha) = \Lambda$), то она является неизвестной ($\alpha \notin \Omega_{\text{know}}$, $\omega(\alpha) = \Delta$);
— если у структуры $\alpha \in \Omega$ определен тип ($\alpha \in \Omega_{\text{type}}$), то выполняется условие $\tau(\alpha) \neq \Lambda$;
— у известных структур определен их тип; таким образом, выполняется условие $\Omega_{\text{know}} \subset \Omega_{\text{type}}$.

Определение 8. Зададим на множестве типов структур T отношение частичного порядка «$\leqslant$» (отношение «быть подтипом»). При этом тип $t_1 \in T$ является подтипом типа $t_2 \in T$ ($t_1 \leqslant t_2$) тогда и только тогда, когда структура типа t_2 содержит структуру типа t_1. Если $t_1 \leqslant t_2$ и $t_1 \neq t_2$, то будем записывать $t_1 < t_2$.

Определение 9. *Множество базовых типов* $T_0 \subset T$ — подмножество множества типов T, такое, что выполняется условие $byte \in T_0$ и для любого $t \in T_0$ не существует типа $s \in T \backslash \{byte\}$, такого, что справедливо неравенство $s < t$. Зададим *множество производных типов* $T_1 = T \backslash T_0$.

Замечание 2. Примерами базовых типов могут являться тип «строка» или тип «числовая переменная».

Определение 10. Будем говорить, что структура $\alpha \in \Omega$ является структурой с *базовым типом*, если выполняется условие $\tau(\alpha) \in T_0$. Будем говорить, что структура α является структурой с *производным типом*, если выполняется условие $\tau(\alpha) \in T_1$.

Определение 11. Определим $H_T : T \to 2^T$ — функцию иерархии типов, сопоставляющую каждому типу $t \in T \backslash \{byte\}$ множество типов $H_T(t) \subset T$ и удовлетворяющую условию: если $r \in H_T(t)$, то $r < t$ и не существует типа $s \in T$, такого, что $r < s < t$. Положим $H_T(byte) = \varnothing$.

Определение 12. Зададим на множестве структур Ω отношение частичного порядка «$\leqslant$» (отношение «быть подструктурой»). При этом структура $\alpha \in \Omega$ является подструктурой $\beta \in \Omega$ ($\alpha \leqslant \beta$) тогда и только тогда, когда структура β содержит структуру α. Если $\alpha \leqslant \beta$ и $\alpha \neq \beta$, то будем записывать $\alpha < \beta$. По определению выполняется условие: если $\alpha \leqslant \beta$, то $\tau(\alpha) \leqslant \tau(\beta)$ и $\lambda(\alpha) \leqslant \lambda(\beta)$.

Замечание 3. Возможны случаи, когда две различные структуры $\alpha_1, \alpha_2 \in \Omega$, $\alpha_1 \neq \alpha_2$, имеют одинаковый тип ($\tau(\alpha_1) = \tau(\alpha_2)$), но при этом их обработчики различны ($\omega(\alpha_1) \neq \omega(\alpha_2)$). Например, это возможно, когда структура — строка α_1, входящая в состав структуры «массив строк», имеет обработчик, отличный от обработчика структуры — строки α_2, входящей в состав структуры «список строк». Таким образом, для рассмотренных структур выполняются условия $\tau(\alpha_1) = \tau(\alpha_2)$ и $\omega(\alpha_1) \neq \omega(\alpha_2)$. При этом любая из этих структур может являться либо известной, либо неизвестной.

Определение 13. Определим $H_\Omega : \Omega \to \Omega^*$ — функцию иерархии структур, сопоставляющую каждой структуре $\alpha \in \Omega$, такой, что $\tau(\alpha) \neq byte$, конечную последовательность структур $H_\Omega(\alpha) = (\alpha_1, \ldots, \alpha_m) \in \Omega^m$, $m \geqslant 1$, удовлетворяющих условиям: $\alpha_i < \alpha$ и не существует структуры $\beta \in \Omega$, такой, что $\alpha_i < \beta < \alpha$ ($i = 1, \ldots, m$). Если $\tau(\alpha) = byte$, то $H_\Omega(\alpha)$ — пустая последовательность (длины 0).

Замечание 4. Если для структуры $\alpha \in \Omega$ справедливо равенство $H_\Omega(\alpha) = (\alpha_1, \ldots, \alpha_m)$, то из определений 8, 11, 12 и 13 следует, что справедливо равенство $H_T(\alpha) = \{\tau(\alpha_1), \ldots, \tau(\alpha_m)\}$.

Замечание 5. Возможен случай, когда типы и значения структуры $\alpha \in \Omega$ и входящих в нее структур не определены. Тогда можно считать, что $\tau(\alpha) = \Lambda$ и $H_\Omega(\alpha) = (\alpha_1, \ldots, \alpha_{\lambda(\alpha)})$, где для $1 \leqslant i \leqslant \lambda(\alpha)$ справедливо равенство $\tau(\alpha_i) = byte$ (структура α состоит из последовательности элементарных структур, длина которой равна размеру структуры α). В то же время возможен случай, когда тип структуры α

не определен ($\tau(\alpha) = \Lambda$) и она состоит из последовательности структур известных типов, не совпадающих с элементарным. Например, структура α может состоять только из структур — строк и структур — числовых констант. При этом в совокупности назначение или взаимосвязь входящих в структуру α структур будут оставаться неизвестными и, следовательно, тип структуры α будет не определен. Другой случай: тип структуры α определен ($\alpha \in \Omega_{\text{type}}$), а множество $H_\Omega(\alpha)$ неизвестно. Например, структура α представляет собой *doc*-файл, тогда как внутренние структуры, составляющие *doc*-файл, неизвестны.

Замечание 6. Так как на множестве структур и на множестве типов структур заданы отношения частичного порядка, то для наглядности представления иерархии структур или иерархии типов структур можно использовать ориентированный граф. При этом вершинами графа являются структуры или их типы, а ребра соответствуют отношению частичного порядка между вершинами.

Часто при исследовании структур данных не анализируются форматы и порядок их представления непосредственно в сущностях-данных (файлах, потоках данных), обрабатываемых исследуемым ПО. Например, как правило, предполагается, что данные структур (последовательности соответствующих им байт) не могут перемешиваться. В рассматриваемом методе АППО осуществляется поиск программных ошибок исследуемого ПО, являющихся следствием, в том числе некорректной обработки ПО структур в сущностях-данных. Значит, целесообразно осуществить анализ структур с учетом возможного их представления в сущностях-данных. Дадим определение.

Определение 14. Пусть структуры $\alpha, \beta \in \Omega$ такие, что $\alpha < \beta$ (по определению 12 выполняется условие $\lambda(\alpha) \leqslant \lambda(\beta)$). Представлением структуры α в структуре β назовем подпоследовательность последовательности $\upsilon(\beta)$, соответствующую значению структуры $\upsilon(\alpha)$. При этом по определению задана инъективная функция $\psi : \{1, \ldots, \lambda(\alpha)\} \to \{1, \ldots, \lambda(\beta)\}$ (которую назовем функцией отображения структуры α в структуру β), такая, что для любых значений структур $\upsilon(\alpha) = (a_1, \ldots, a_{\lambda(\alpha)})$ и $\upsilon(\beta) = (b_1, \ldots, b_{\lambda(\beta)})$ справедливы равенства $a_i = b_{\psi(i)}$, где $1 \leqslant i \leqslant \lambda(\alpha)$.

Замечание 7. В соответствии с определением 14 для структур $\alpha, \beta \in \Omega$, таких, что $\alpha < \beta$, элементы последовательности $\upsilon(\beta)$, соответствующие представлению структуры α в структуре β, могут следовать в произвольном порядке и между ними могут содержаться элементы, не принадлежащие значению $\upsilon(\alpha)$ структуры α.

В ПО, как правило, структуры базовых типов являются наиболее распространенными. В связи с этим производители ПО стремятся упростить представление этих структур в сущностях-данных, что упрощает разработку обработчиков структур базовых типов. Таким образом, целесообразно использовать следующее предположение.

Предположение 2. Пусть структуры $\alpha, \beta \in \Omega$ такие, что $\alpha < \beta$ и структура α имеет базовый тип ($\tau(\alpha) \in T_0$). Тогда функция ψ отображения структуры α в структуру β обладает следующим свойством: для $0 \leqslant i < \lambda(\alpha)$ выполняется условие $\psi(i+1) = \psi(i) + 1 = \psi(1) + i$.

Таким образом, описана модель структур, содержащихся в сущностях-данных, обрабатываемых исследуемым ПО.

2. Модель процесса исследуемого ПО

Рассматриваемый метод АППО является динамическим [8], то есть он ориентирован на анализ программных ошибок в исследуемом ПО, когда оно является активным

(выполняется как процесс ОС). В связи с этим опишем модель процесса ОС, которую используем при описании метода АППО. Дадим определения.

Определение 15. Процессом ОС, реализующим исследуемое ПО (процессом исследуемого ПО), назовем автомат $P{<}S, I, p, s_0{>}$, где S — множество всех состояний процесса; $I \subset \Omega^*$ — множество входных данных; $p : I \times S \to S$ — функция переходов процесса из состояния в состояние; $s_0 \in S$ — начальное состояние процесса.

Замечание 8. Функция перехода p задается последовательностью исполняемых инструкций кода исследуемого ПО, которое реализует процесс P.

Определение 16. Для каждого процесса исследуемого ПО $P{<}S, I, p, s_0{>}$ на множестве состояний S зададим множество *аварийных состояний* $E \subset S \backslash \{s_0\}$. При этом по определению для каждого аварийного состояния $e \in E$ процесса P выполняется условие: для любых входных данных $i \in I$ справедливо равенство $p(i, e) = e$ (процесс, перейдя в аварийное состояние, не может перейти ни в какое другое состояние).

Если процесс попадает в аварийное состояние, то будем говорить, что процесс P *завершился аварийно*.

Замечание 9. Как правило, для процесса любого исследуемого ПО множество аварийных состояний E можно считать всегда известным, так как оно задается средой ОС, в которой функционирует процесс. Процесс аварийно завершается тогда и только тогда, когда выполнение очередной машинной инструкции «физически» невозможно (например, нет доступа к памяти или процессор не может декодировать машинную команду) [9].

Определение 17. *Эталонные входные данные* — входные данные, которые, как правило, созданы процессом исследуемого ПО, и их обработка по определению не приводит к его аварийному завершению. Если эталонные входные данные являются файлом, который подается на вход процессу исследуемого ПО, то назовем его *эталонным входным файлом*.

Определение 18. Пусть $P{<}S, I, p, s_0{>}$ — процесс исследуемого ПО. Будем говорить, что в процессе P содержится программная ошибка, если существуют входные данные $i \in I$, такие, что справедливо условие $p(i, s_0) \in E$. *Программная ошибка* — это ошибка в коде исследуемого ПО (как правило, допущенная его разработчиками), которая приводит к аварийному завершению процесса исследуемого ПО на соответствующих входных данных.

Используем обозначения:

$I_S \subset I$ — множество эталонных входных данных для процесса $P{<}S, I, p, s_0{>}$, при этом по определению 17 для любых эталонных входных данных $i \in I_S$ и любого состояния $s \in S \backslash E$ выполняется условие $p(i, s) \notin E$;

$I_F \subset I$ — множество входных данных, при обработке которых происходит аварийное завершение процесса $P{<}S, I, p, s_0{>}$, при этом по определениям 17 и 18 справедливо равенство $I_F \cap I_S = \varnothing$;

$(e, i_1, \ldots, i_n)$ — траектория программной ошибки процесса $P{<}S, I, p, s_0{>}$ исследуемого ПО; при этом $e \in E$ и $(i_1, \ldots, i_n)$ — последовательность входных данных длины $n \geqslant 1$, подаваемых на вход процесса $P{<}S, I, p, s_0{>}$, где для $1 \leqslant k \leqslant n$ выполняется условие $i_k \in I$ и справедливо равенство $p(i_n, p(i_{n-1}, \ldots p(i_2, p(i_1, s_0)) \ldots)) = e$;

F — множество всех траекторий программных ошибок процесса $P{<}S, I, p, s_0{>}$ исследуемого ПО.

В дальнейшем целесообразно использовать следующее предположение.

Предположение 3. В любом исследуемом ПО существуют программные ошибки.

В рамках предположения 3 для процесса $P{<}S, I, p, s_0{>}$ любого исследуемого ПО существуют входные данные $(i_1, \ldots, i_n) \in I$, где $n \geqslant 1$, такие, что справедливо равенство $p((i_1, \ldots, i_n), s_0) \in E$ (при обработке входных данных $(i_1, \ldots, i_n)$ процесс $P{<}S, I, p, s_0{>}$ завершается аварийно). При этом $(p((i_1, \ldots, i_n), s_0), i_1, \ldots, i_n) \in F$ является траекторией программной ошибки процесса $P{<}S, I, p, s_0{>}$.

Таким образом, описана модель процесса исследуемого ПО.

3. Метод автоматизированного поиска программных ошибок

Для применения метода АППО разработано программное средство (ПС), реализующее следующие функции:

— формирование входных данных для исследуемого ПО;
— проведение тестов (активизация исследуемого ПО — запуски процесса исследуемого ПО и передача ему подготовленных входных данных);
— регистрация программных ошибок, возникших в результате обработки входных данных процессом исследуемого ПО.

Опишем метод АППО.

Метод АППО. Пусть дано исследуемое ПО. Применение метода АППО состоит в выполнении следующих шести этапов.

Этап 1. Проводится анализ доступной информации об исследуемом ПО, в том числе осуществляется анализ:

— доступной документации;
— доступных спецификаций структур данных, обрабатываемых исследуемым ПО;
— доступных фрагментов исходных кодов модулей исследуемого ПО или ОС (т. е. в среде, где активизируется исследуемое ПО), которые реализуют обработчики структур, содержащихся во входных данных исследуемого ПО.

В результате выполнения этапа 1 для исследуемого ПО:

— задаются множества структур Ω, известных структур Ω_{know}, типов структур T, структур известных типов Ω_{type} и обработчиков структур Θ, функции λ, υ, τ, ω, H_T и H_Ω;
— описывается, как в среде ПС исследуемое ПО может быть активизировано, т. е. запущен процесс $P{<}S, I, p, s_0{>}$ исследуемого ПО, и как этому процессу могут передаваться входные данные (множество $I \subset \Omega^*$); описывается множество входных данных I и их вид (файл, сетевые пакеты или их последовательность, последовательность событий нажатия клавиш на клавиатуре).

Этап 2. На основе результатов выполнения этапа 1 выбираются структуры, для которых является *целесообразным* восстановление и анализ кода их обработчиков. Выбор структур может быть осуществлен с учетом следующих факторов:

— наличия ресурсов для проведения детального анализа (дизассемблирование, отладка, тестирование, эксперименты на макете) кода обработчиков структур в исследуемом ПО;
— предположений о наличии программных ошибок в обработчиках выбранных структур;
— предположений о возможности восстановления кода обработчиков структур за доступное время.

Для выбранных структур осуществляется восстановление и анализ кода их обработчиков, в результате по определению 5 данные структуры становятся известными. Таким образом, уточняются значения множеств Ω, Ω_{know}, T, Ω_{type} и Θ, функций λ, υ, τ, ω, H_T и H_Ω.

Этап 3. На основе результатов выполнения этапов 1 и 2 выбираются неизвестные структуры (структуры $\alpha \notin \Omega_{\text{know}}$), для которых определен тип (структуры из множества Ω_{type}) или определение их типа является целесообразным. Выбор структур может быть осуществлен с учетом следующих факторов:

— наличия ресурсов для проведения детального анализа и описания структур;
— предположений о наличии программных ошибок в обработчиках структур выбранных типов;
— предположений о возможности описания структур за доступное время.

Для выбранных структур осуществляется определение их типа. Таким образом, уточняются значения множеств T и Ω_{type}, функций λ, υ, τ, H_T и H_Ω.

Этап 4. ПС осуществляет автоматизированный поиск программных ошибок. Поиск заключается в том, что в среде ПС запускается процесс $P{<}S, I, p, s_0{>}$ исследуемого ПО (каждый запуск процесса назовем тестом). При выполнении одного теста процессу $P{<}S, I, p, s_0{>}$ передаются входные данные, сформированные разработанными процедурами формирования входных данных (ПФВД; более подробное описание ПФВД рассмотрено далее) из эталонных входных данных путем модификации значения структур из множества Ω_{type} (множества структур, тип которых определен). Если в результате обработки входных данных процесс $P{<}S, I, p, s_0{>}$ завершился аварийно (найдена программная ошибка), то управление возвращается ПС, которое фиксирует состояние процесса $P{<}S, I, p, s_0{>}$ и входные данные, обработка которых стала причиной его аварийного завершения. При этом ПС восстанавливает корректность процесса $P{<}S, I, p, s_0{>}$ для дальнейшего тестирования.

Этап 5. На основе результатов выполнения этапов 1, 2 и 3 выбираются структуры, тип которых не определен и которые целесообразно считать элементарными (считать, что тип таких структур равен *byte*). Выбор структур может быть осуществлен с учетом следующих факторов:

— предположений о наличии программных ошибок в обработчиках данных структур;
— наличия ресурсов для проведения за доступное время тестов исследуемого ПО при обработке входных данных, в которых модифицируются значения данных структур.

Задается Ω_{byte} — множество структур, выбранных на этапе 5.

Этап 6. ПС осуществляет тестирование исследуемого ПО. При выполнении одного теста процессу $P{<}S, I, p, s_0{>}$ передаются входные данные, сформированные ПФВД из эталонных входных данных путем модификации значения структур из множества Ω_{byte} (множества структур, которые целесообразно считать элементарными). Если в результате обработки входных данных процесс $P{<}S, I, p, s_0{>}$ завершился аварийно (найдена программная ошибка), то управление возвращается ПС, которое фиксирует состояние процесса $P{<}S, I, p, s_0{>}$ и входные данные, обработка которых стала причиной его аварийного завершения. При этом ПС восстанавливает корректность процесса $P{<}S, I, p, s_0{>}$ для дальнейшего тестирования.

Прокомментируем метод АППО.

Замечание 10. Этап 1 метода АППО состоит в предварительном исследовании ПО, и время, требуемое для его выполнения, целесообразно не учитывать при оценке

результативности метода АППО. Кроме того, выбор структур, которые целесообразно считать элементарными, на этапе 5 не требует затрат времени, так как может быть осуществлен, например, случайно из неизвестных структур входных данных, тип которых не определен.

Замечание 11. Этап 2 предполагает поиск программных ошибок в сложноструктурированных данных на основе «классического» подхода (дизассемблирование и отладка). При выполнении второго этапа следует соблюдать компромисс. С одной стороны, знание всех исходных кодов обработчиков структур позволит решить задачу более эффективно (например, на основе средств анализа исходных кодов). С другой стороны, временные затраты, требуемые для восстановления исходных кодов, могут оказаться неприемлемыми. Отметим, что этап 2 может проводиться параллельно с этапами 3–6.

Можно также предложить, что оценка результативности метода АППО должна учитывать следующие параметры:

— среднее время обнаружения программной ошибки;
— средний объем кода обработчиков структур;
— среднее число программных ошибок, выявленных на этапах 2, 4 и 6;
— среднее время проведения одного теста на этапах 4 и 6;
— относительное число аварийных завершений, являющееся усредненным отношением числа аварийно завершившихся тестов к общему числу проведенных тестов на этапах 4 и 6.

В настоящий момент для эмпирической оценки результативности метода используется относительное число аварийных завершений (обозначим его через K). Опыт применения описанного метода показал, что при значении K около 0,1% (на 1000 тестов один тест завершается аварийно) и выше результативность метода для данного ПО можно признать удовлетворительной. Поскольку время проведения одного теста, как правило, занимает не более 10 с (на типовом компьютере), то при значении K порядка 0,1% и выше программные ошибки будут найдены за приемлемое время. Эксперименты показали, что существует ПО, для которого значение K достигает 5–6%.

Замечание 12. После того как программная ошибка будет найдена, наиболее разумно связаться с разработчиками программных модулей и сообщить об ошибке. К сожалению, это не всегда возможно. Также следует учесть, что если информация о программной ошибке будет принята разработчиками к сведению, то до внесения соответствующих исправлений может пройти достаточно большой срок. Если проявления программной ошибки не допустимы, то одним из возможных решений является организация дополнительной фильтрации входных данных. Как минимум, дополнительная фильтрация будет блокировать входные данные, для которых уже были выявлены программные ошибки в результате применения метода АППО.

В заключение опишем назначение ПФВД в алгоритме АППО.

4. Процедуры формирования входных данных

Применение метода выявило существенную зависимость между параметром K и применяемыми при тестировании на 4 и 6 этапах ПФВД. Были разработаны общие ПФВД (насколько это возможно), применимые к большинству исследуемого ПО. Тем не менее для повышения значения параметра K при тестировании конкретного ПО требуется разработка (или модификация существующих) индивидуальных ПФВД.

ПФВД должны сформировать входные данные, которые процессом исследуемого ПО будут приняты к обработке.

На этапах 4 и 6 метода АППО генерируются наборы входных данных, на которых запускается процесс исследуемого ПО $P{<}S, I, p, s_0{>}$. Входные данные генерируются ПФВД из эталонных входных данных путем модификации значения структур из множеств Ω_{type} и Ω_{byte}.

Используются ПФВД двух видов:

— статические ПФВД, основанные на различных эвристических правилах, которые применяются для преобразования структур без учета особенностей их обработки процессом исследуемого ПО;
— динамические ПФВД, преобразующие эталонные входные данные по правилам, которые формируются на основе трассировки процесса ПО.

Для статических ПФВД задаются различные эвристические правила преобразования структур из множеств Ω_{type} и Ω_{byte} (этапы 4 и 6). Если структуры, содержащиеся в множествах Ω_{type} и Ω_{byte}, будут найдены во входных эталонных данных, то они будут преобразованы ПФВД.

Для повышения результативности метода было решено сочетать стрессовое тестирование и динамический анализ на основе трассировки процесса исследуемого ПО. Непосредственно перед выполнением этапов 4 и 6 для каждого эталонного файла выполняются следующие действия:

— собирается трасса процесса исследуемого ПО на эталонном входном файле;
— уточняются значения множеств T, Ω_{type} и Ω_{byte}, функций λ, υ, τ, H_T и H_Ω;
— выбирается статическая ПФВД, наиболее подходящая для проведения очередного теста.

На основе трассы процесса исследуемого ПО в ПС, реализующем метод, введены вспомогательные механизмы — это механизм оценки покрытия программного кода, задействованного при обработке входных данных, и механизм определения обработчиков структур на основе сравнения трасс.

Заключение

Разработанный автором метод применяется для анализа надежности ПО в ОС семейства *Windows* 2003/*XP*/*Vista*, а также частично опробован в ОС *Windows CE* и ОС *Linux*. Эксперименты показали хорошие результаты для ПО, обрабатывающего сложноструктурированные форматы файлов.

Реализованное на основе метода ПС работает в автоматическом режиме, не требует существенных усилий со стороны оператора ПС. В качестве результатов работы ПС предоставляет множество сигнатур для организации дополнительной фильтрации входных данных, вызывающих аварийное завершение в ПО, доступ к исходным кодам которого отсутствует. Таким образом, может быть повышена надежность функционирования разрабатываемых программных комплексов.

В настоящий момент ведутся работы по исследованию и оценке параметров результативности метода с целью анализа его возможностей и расширения границ применения.

ЛИТЕРАТУРА

1. *Козиол Д., Личфилд Д., Эйтэл Д. и др.* Искусство взлома и защиты системы. СПб.: Питер, 2006. 416 с.

2. *Ховард М., Лебланк Д.* Защищенный код. 2-е изд. М.: Издательско-торговый дом «Русская Редакция», 2005. 704 с.
3. *Хогланд Г., Мак-Гроу Г.* Взлом программного обеспечения: анализ и использование кода. М.: Издательский дом «Вильямс», 2005. 400 с.
4. *Eilam E.* Reversing: Secrets of Reverse Engineering. Wiley Publishing, 2005. 589 p.
5. *Miller C., Peterson Z. N. J.* Analysis of Mutation and Generation-Based Fuzzing — `securityevaluators.com/files/papers/analysisfuzzing.pdf` — 2007.
6. *Neystadt J.* Automated Penetration Testing with White-Box Fuzzing — `msdn.microsoft.com/en-us/library/cc162782.aspx` — Microsoft Corporation, 2008.
7. *Левитин А. В.* Алгоритмы: введение в разработку и анализ. М.: Издательский дом «Вильямс», 2006. 576 с.
8. *Калбертсон Р., Браун К., Кобб Г.* Быстрое тестирование. М.: Издательский дом «Вильямс», 2002. 384 с.
9. Intel Architecture Software Developer's Manual. V. 3: System Programming. Intel, 1999.

СВЕДЕНИЯ ОБ АВТОРАХ

АКЕНЬШИНА Екатерина Алексеевна — студентка Томского государственного университета, г. Томск. E-mail: **katarinka_87@list.ru**

БАНДМАН Ольга Леонидовна — профессор, доктор технических наук, главный научный сотрудник Института вычислительной математики и математической геофизики СО РАН, г. Новосибирск. E-mail: **bandman@ssd.sscc.ru**

ДЕВЯНИН Петр Николаевич — доктор технических наук, доцент, заместитель заведующего кафедрой Института криптографии, связи и информатики, г. Москва. E-mail: **peter_devyanin@hotmail.com**

ДЕВЯНИНА Екатерина Витальевна — инженер ФГНУ «ГНТЦ „Наука“», г. Москва. E-mail: **k_vitalievna@mail.ru**

ЕГОРУШКИН Олег Игоревич — старший преподаватель кафедры математического моделирования и информатики Красноярского государственного аграрного университета, г. Красноярск. E-mail: **oiegorush@mail.ru**

КОЛЕГОВ Денис Николаевич — аспирант кафедры защиты информации и криптографии Томского государственного университета, г. Томск. E-mail: **d.n.kolegov@gmail.com**

МАКАРОВ Алексей Николаевич — научный сотрудник ФГНУ «ГНТЦ „Наука“», г. Москва. E-mail: **byalex@mail.ru**

МЕДВЕДЕВ Юрий Геннадьевич — кандидат технических наук, научный сотрудник Института вычислительной математики и математической геофизики СО РАН, г. Новосибирск. E-mail: **medvedev@ssd.sscc.ru**

ПАНКРАТОВ Иван Владимирович — студент Томского государственного университета. E-mail: **ivan.pankr@gmail.com**

ПАРВАТОВ Николай Георгиевич — кандидат физико-математических наук, доцент кафедры защиты информации и криптографии Томского государственного университета, г. Томск. E-mail: **parvatov@mail.tsu.ru**

ПАУТОВ Павел Александрович — аспирант кафедры защиты информации и криптографии Томского государственного университета, г. Томск. E-mail: **__Pavel__@mail.ru**

САФОНОВ Константин Владимирович — доктор физико-математических наук, заведующий кафедрой прикладной математики Сибирского государственно-

го аэрокосмического университета им. М. Ф. Решетнева, г. Красноярск. E-mail: **safonovkv@rambler.ru**

ТУЖИЛИН Михаил Эльевич — старший научный сотрудник, кандидат физико-математических наук, доцент кафедры фундаментальной и прикладной математики Российского государственного гуманитарного университета, г. Москва. E-mail: **mtmt@rambler.ru**

ФОМИЧЁВ Владимир Михайлович — старший научный сотрудник, доцент, доктор физико-математических наук, ведущий научный сотрудник Учреждения Российской академии наук «Институт проблем информатики РАН», г. Москва. E-mail: **fomichev@nm.ru**

ЧЕРЕМУШКИН Александр Васильевич — доктор физико-математических наук, заведующий кафедрой Института криптографии, связи и информатики, г. Москва. E-mail: **avc238@mail.ru**

ШАБАЛДИНА Наталия Владимировна — кандидат технических наук, доцент кафедры ИТИДиС Томского государственного университета, г. Томск. E-mail: **svn@kitidis.tsu.ru**

АННОТАЦИИ СТАТЕЙ НА АНГЛИЙСКОМ ЯЗЫКЕ

Parvatov N. G. **LOWER AND UPPER NEIGHBORHOODS IN SET WITH CLOSURE PROPERTY.** Lower and upper neighborhoods in set with closure property are studied.

Tuzhilin M. E. **APN-FUNCTIONS.** This survey contains basic results about properties, equivalence and methods of construction of APN-functions.

Fomichev V. M. **ON COMPLEXITY OF FORMAL CODING METHOD FOR ANALYSIS OF GENERATOR WITH MONOCYCLE SUBSTITUTIONAL TRANSITION FUNCTION.** Here we investigate autonomous automata with automaton states being binary n-dimensional vectors and transition function being a monocycle substitution. The complexity T_n of solving gamma generating equations system by formal coding method is estimated asuming the number of equations is not constrained. Bounds of T_n are obtained by estimating line complexity and the order monomial sets for the output functions sequence. It is stated that $TL(2^{n-1}) < T_n < TL(2^n)$ where $TL(m)$ is a complexity of solving linear equations system of size $m \times m$ over field GF(2).

Cheremushkin A. V. **ALMOST ALL LATIN SQUARES HAVE TRIVIAL AUTOPARATOPY GROUP.** The main result is: almost all Latin squares of order n have trivial autoparatopy group as $n \to \infty$. As a consequence we obtain the asymptotic number of main classes of Latin squares of order n:

$$\frac{L_n}{6n!^3}(1 + o(1)),$$

where L_n — number of Latin squares of order n.

Bandman O. L. **DISCRETE MODELS OF PHYSICAL-CHEMICAL PROCESSES.** The results of the investigation of discrete models of physical-chemical kinetic processes are presented in a systematic form. The models are extensions of the classical fon-Neumann Cellular Automaton (CA), differing from it in two points: 1) transition functions are allowed to be probabilistic, and 2) not only synchronous, but asynchronous and composed modes of functioning may be used. Mathematical background of the models is based on the formalisms of the "Parallel Substitution Algorithm". Validity conditions and parallel implementation efficiency for synchronous and asynchronous CA-models are studied. All models are illustrated by the results of computer simulation of physical-chemical kinetics on micro- and nano- levels.

Medvedev Yu. G. **CELLULAR-AUTOMATON MODEL FOR EXPLOSION SIMULATION.** A new cellular-automaton model for gas-powder flow simulating is proposed. Computing experiments have been carried out with this model; they demonstrate a correlation of the new model with the physical laws.

Pankratov I. V. **SYMMETRIC STREAM AND FINITE AUTOMATON CIPHERSYSTEMS.** The following statements are proved in the paper: 1) stream ciphersystems with the indistinguishable key stream generators are indistinguishable themselves; 2) the output function of the encryption automaton in any finite automaton ciphersystem

is injective one for any its state and key being fixed; 3) the classes of the stream and of the finite automaton ciphersystems are functionally equivalent; 4) every selfsynchronizing with a delay τ finite automaton ciphersystem with the strongly connected projections of the encryption automaton is indistinguishable from a ciphersystem built on the base of a shiftregister of the length τ. Besides, a descriptive and a constructive definitions of the selfsynchronizing stream ciphersystem are introduced, and the equivalence between them are stated.

Devyanin P. N. **ANALYSIS OF CONDITIONS FOR RECEPTION OF ACCESS OWNING WITHIN THE BASE ROLE DP-MODEL OF COMPUTER SYSTEMS WITHOUT MEMORY INFORMATION FLOWS.** Necessary and sufficient conditions for the reception by untrusted user sessions of access owning to trusted user sessions are stated within the base role DP-model of computer systems with the role management of access. It is supposed in the consideration that any number of user sessions cooperate, and they do not get access owning to each other with the use of memory information flows to entities functionally associated with the user sessions.

Devyanina E. V. **DP-MODEL OF COMPUTER STEGANOGRAPHY SYSTEM.** In article attempt on the basis of family of discretionary DP-models is realised to apply the theory of formal modelling of management of access and information streams to the analysis of conditions of safe functioning computer steganography systems.

Kolegov D. N. **MODELING NETWORK COMPUTER SYSTEMS WITH VULNERABILITIES.** In the paper, a model for discretionary network computer systems with vulnerabilities is proposed. The model is based on the DP-model with functionally or parametrically associated with subjects entities. It is shown how to apply this model for finding all the possible paths to violate the security policy, for representating other models designed for the same purposes by its means, and for developing it in order to verify if an untrusted entity could take the rights of a trusted entity.

Pautov P. A. **THE IMPLEMENTATION OF THE DBMS CREDENTIALS PROTECTION METHOD IN THE WEB APPLICATIONS.** The paper considers the peculiarities of authentication in multi-tier environment and corresponding security problems. The protection method for DBMS credentials is provided. The ways for the implementation of the method using PHP programming language are considered.

Akenshina E. A., Shabaldina N. V. **TEST SUITES DERIVATION FOR NONDETERMINISTIC FINITE STATE MASHINES WITH RESPECT TO THE SEPARABILITY RELATION.** In this paper we improve test suite derivation method for nondeterministic FSMs with respect to the nonseparability relation. The nonseparability relation can be checked when "all weather conditions" assumption does not hold. Our modification is based on the refinement of tree truncation conditions. It is shown that test suites constructed according to the method with our improvements are (in most cases) shorter and still complete.

Egorushkin O. I., Safonov K. V. **AN ANALITIC APPROACH IN THE THEORY OF CONTEXT-FREE LANGUAGES GREIBACH NORMAL FORM.** Context-free languages are consider as formal power series, which are solutions of the polynomial equations systems with noncommutative variables respectively multiplication. It is suggested to investigate these systems in Greibach normal form, that allows to research it

more effectively. Commutative images of languages and defining systems are considered in complex domain.

Makarov A. N. **METHOD FOR AUTOMATED PROGRAM ERRORS SEARCH IN COMPLEX DATA PROCESSING ALGORITHMS.** In the paper, an original method developed by the author is discussed. The method permits to search program errors in software without source code. The method is based on stress testing and automated software trace. In this case, the test data can be formed statically or dynamically on the base of tracing results.