# ПАМЯТЬ ЭВМ КАК АБСТРАКТНЫЙ АВТОМАТ

А. Г. Сурков

МЦ СПбГУ
199178, Санкт-Петербург, 14-я линия ВО, д. 29

<ASurkov@mail.ru>

УДК 681.3
*А. Г. Сурков.* **Память ЭВМ как абстрактный автомат** // Труды СПИИРАН, Вып. 2, т. 2. — СПб.: Наука, 2005.
**Аннотация.** *Представлен формализм построения абстрактного автомата, реализующего функциональность оперативной памяти ЭВМ, построено метрическое множество таких автоматов, определены операции сложения и умножения. Рассмотрены вопросы использования полученных формальных решений в практике распределенных вычислений.* — Библ. 7 назв.

UDC 681.3
A. G. Surkov. **Computer memory in abstract automat approach** // SPIIRAS Proceedings. Issue 2, vol. 2. — SPb.: Nauka, 2005.
**Abstract.** *Construction of the abstract computer memory automat is provided, which realizes functionality of the computer's memory, additive and multiplicative operation are defined. Discussed some questions of utilization achieved formal results in practice of distributed computing.* — Bibl. 7 items.

## 1. Построение абстрактного автомата

Рассмотрим память ЭВМ как абстрактный аппарат, воспринимающий обращение к памяти как входной символ, а выдаваемый результат — как выходной символ.

Пусть CTL — множество поддерживаемых памятью типов операций, состоящее из операций записи и чтения: $CTL = \{RD;WR\}$ ; ADR — множество адресов, поддерживаемых памятью; D — алфавит симовлов информации, хранимых в памяти. Предположим, для простоты, что все запросы к памяти удовлетворяются и какой либо диагностики не требуется. Случай, поддерживающий отказы памяти и авторизацию запросов, получается расширением входного и выходного алфавитов на необходимый набор символов. Также предполагаем, что $|D|>1$.

Представим память как неинициальный автомат $(A,Q,B,\varphi,\psi)$. Входным алфавитом A автомата памяти будет $A = D * ADR * CTL$, а выходным алфавитом B будет алфавит символов информации D. Обработку входного символа, содержащего RD, будем называть операцией чтения, а содержащего WR — операцией записи. В операции чтения информационный символ во входном символе игнорируется, а в операции записи выходной символ совпадает с записываемым. Введем также псевдооперации: $Adr : A \mapsto ADR$, которая выделяет из входного символа адресную часть, $D : A \mapsto D$ информационный символ, $Ctl : A \mapsto CTL$ — управляющую информацию. Функция $\alpha : D * ADR * CTL \mapsto A$ формирует символ из его компонент.

Сформулируем основные свойства, которыми обладает память ЭВМ в терминах преобразования входных слов в выходные.



Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Для удобства изложения введем функцию $\psi\_ : A^* \mapsto B$, которая отображает последнный выданный памятью символ от входного слова. Эта функция более наглядна, чем выходной символ, так как значение выходного символа автомата памяти может зависеть от предшествующих входных символов.

*Свойство 1*. На операцию чтения не влияет результат исполнения предшествующей ей операции чтения.

$$c = RD, \forall((k_1,k_2) \in ADR^2,(x_1,x_2) \in D^2, q \in Q, a \in A^*, i \in \{1;2\}),$$
$$\alpha_i = \alpha(x_i,k_i,c) : \psi\_(q,a\alpha_2) = \psi\_(q,a\alpha_1\alpha_2). \quad (1)$$

*Свойство 2*. На операцию чтения не влияет результат исполнения предшествующей ей операции записи по другому адресу.

$$c_1 = WR; c_2 = RD; \forall((k_1,k_2) \in ADR^2, k_1 \neq k_2; (x_1,x_2) \in D^2; q \in Q; a \in A^*,$$
$$i \in \{1;2\}), \alpha_i = \alpha(x_i,k_i,c_i) : \psi\_(q,a\alpha_2) = \psi\_(q,a\alpha_1\alpha_2). \quad (2)$$

*Свойство 3*. На операцию записи не влияет результат исполнения предшествующей ей любой операции.

$$c_1 \in CTL, c_2 = WR, \forall((k_1,k_2) \in ADR^2,(x_1,x_2) \in D^2, q \in Q, a \in A^*, i \in \{1;2\}),$$
$$\alpha_i = \alpha(x_i,k_i,c_i) : \psi\_(q,a\alpha_2) = \psi\_(q,a\alpha_1\alpha_2). \quad (3)$$

*Свойство 4*. Операция чтения, идущая сразу же за операцией записи по тому же адресу, дает в результате записанный предыдущей операцией символ вне зависимости от предшествующих записи операций.

$$c_1 = WR, c_2 = RD, \forall(k \in ADR,(x_1,x_2) \in D^2, q \in Q, a \in A^*, i \in \{1;2\}),$$
$$\alpha_i = \alpha(x_i,k,c_i) : \psi\_(q,a\alpha_1\alpha_2) = x_1. \quad (4)$$

Обозначим как `MEM(D, ADR, CTL)` класс автоматов памяти, имеющих алфавит хранимых символов `D`, множество адресов `ADR` и множество управляющих символов `CTL`.

*Лемма 1*. Последняя во входном слове операция чтения по некоторому фиксированному адресу будет выдавать значение последнего записанного по этому адресу символа, а при отсутствии такой операции записи, значение операции $\psi\_$ не определено.

Рассмотрим входную цепочку $a'\alpha''$, где $\alpha''$ — операция чтения по указанному адресу. Рассмотрим разбиение цепочки $a'$ как $a' = a_1\alpha' a_2$, где $a_1$, $a_2$ — произвольные цепочки из `A*`, а $\alpha'$ — операция записи символа `d` по указанному адресу. Варианты:

1. Такое разбиение не существует ввиду отсутствия $\alpha'$ во входной цепочке. Тогда значение операции $\psi\_$ не определено.

2. Цепочка $a_1$ нам далее не интересна, так как согласно (3) она не влияет на вычисление функции $\psi\_$. Аналогично расмотрим разбиение подцепочки $a_2$ на $a_1'\alpha' a_2'$, при этом $a_1'$ не пусто. Варианты:


Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

2.1 Такое разбиение осуществимо. Тогда производим переобозначение цепочек снятием штриха и возвращаемся к шагу 2.

2.2 Такое разбиение неосуществимо. Тогда $\alpha'$ соответствует последней предшествующей операции записи по данному адресу. То есть в цепочке $a_2$ отсутствует операция записи по указанному адресу. Тогда согласно (1) и (2)

$$\psi\_(a_1\alpha' a_2\alpha'') = \psi\_(a_1\alpha'\alpha'')$$

и, возвращаясь к исходной цепочке:

$$\psi\_(a'\alpha'') = \psi\_(a_1\alpha'\alpha'') = d$$

по (4), что и требовалось доказать.

*Следствие*. При наличии в цепочке некоторого количества операций записи по одному фиксированному адресу, все эти операции, кроме последней, могут быть проигнорированы.

Непосредственно следует из действий по доказательству леммы.

*Лемма 2.* Для автомата памяти состояния, соответствующие разным адресам, различимы.

Так как $|D|>1$, то $\exists\{d_1,d_2\}\subset D, d_1\neq d_2; \{k_1,k_2\}\subset ADR, k_1\neq k_2$. Пусть $a\in A^*$: $a$ содержит операции записи символа $d_1$ для всех адресов памяти. Пусть $a_1=\alpha(d_2,k_2,RD)$. И $a_2=\alpha(d_2,k_2,WR)$. Тогда цепочки $aa_1$ и $aa_2$ различают эти состояния.

*Лемма 3*. Для автомата памяти состояния, соответствующие различной хранимой информации по указанному адресу, различимы.

Эти состояния различаются цепочкой, оканчивающейся операцией чтения по указанному адресу.

*Лемма 4*. Операции чтения не меняют состояния неприводимого автомата памяти.

Действительно, если при исполнении единичной операции чтения состояние автомата изменяется, то должна существовать цепочка, различающие эти состояния, что противоречит свойствам 1 и 3.

Так как множества $D$ и $ADR$ конечны, то их можно полностью упорядочить, введя в этих множествах некоторое отношение порядка. Например, закодировать символы и адреса и сравнивать их в соответствии со значением их двоичной кодировки.

*Лемма 5*. Свойства (1–4) определяют передаточную функцию абстрактного инициального автомата памяти.

Определим инициальный автомат памяти $V^0_{mem}=(A,Q,B,\varphi,\psi,q_0)$. Начальное состояние $q_0$ будет создаваться входным словом $a_0$, содержащим операции записи по всем адресам.

Тогда по лемме 1 $\psi\_(a^0a')$ будет давать начальное значение, если слово a' не содержит записи по адресу, указанному в последнем символе цепочки, или последний записанный цепочкой a' символ. Тогда рассматривая входные



Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

слова a'α' для $V^{\circ}_{mem}$ как $a^0a'\alpha'$ для первоначального абстрактного автомата памяти, где $\alpha'$ — очередной входной символ во входной цепочке для $V^{\circ}_{mem}$ и соответствующий символ $\psi\_(a_0a'\alpha')$ — как выходной, получим для $V^{\circ}_{mem}$ передаточную функцию $f : A^* \mapsto B^*$.

Требования (1–4) с заданными начальными значениями однозначно определяют передаточную функцию для $V^{\circ}_{mem}$. Очевидно, что любые два абстрактных аппарата памяти после приема такого входного слова $a_0$ будут находиться в неотличимых состояниях.

*Теорема 1.* В изучаемом неприводимом аппарате базисом достижимости будет являться набор слов, составленных из символов - операций записи каждого информационного символа по всем адресам автомата.

Для того, чтобы это было справедливо, покажем, что:

1) $\{\varphi(q, \bigcup_{d\in D,\, k\in ADR}\alpha(d,k,WR))\} = Q$. Очевидно, что цепочкой, состоящей из операций записи нужной информации по соответствующим адресам, формируется некоторое состояние автомата. Пусть $\{\varphi(q, \bigcup_{d\in D,\, k\in ADR}\alpha(d,k,WR))\} = QQ \subset Q$. Предположим, что $\exists q' \in Q \setminus QQ$. Рассмотрим набор информационных символов $(d_1, \ldots, d_{Adr})$, полученный в результате исполнения операции чтения по всем адресам автомата. Построим состояние q"исполнением операций записи полученного набора $(d_1,...,d_{Adr})$ по всем адресам. По построению $q' \in Q \setminus QQ$, $q'' \in QQ$, то есть $q' \neq q''$, но $q'$ неотличимо от $q''$, что противоречит предположению о неприведенности аппарата.

2) Если $d_1 \neq d_2$, то $\varphi(q,\alpha(d_1k,WR)) \neq \varphi(q,\alpha(d_2k,WR))$ по лемме 3 и, если $k_1 \neq k_2$, то $\varphi(q,\alpha(d_1k_1,WR)) \neq \varphi(q,\alpha(d_2k_2,WR))$ по лемме 2.

3) Если известно состояние автомата, то корректирующие действия могут быть произведены не полным набором символов записи.

В полученном аппарате базисом отличимости будет являться набор слов, составленных из символов — операций чтения какого либо информационного символа по всем адресам автомата. Действительно:

1) Если состояния $q$ и $q'$ не совпадают, то это означает, что по некоторому адресу записаны различные символы (лемма 3), следовательно, операция чтения по этому адресу выдаст различные выходные символы.

2) Базис состоит из односимвольных слов.

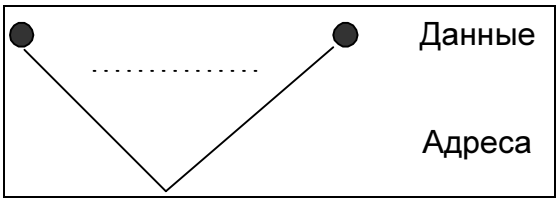
3) Собственное подмножество данного множество не будет содержать операций чтения по какому либо адресу, а значит, не будет различать состояния, отличающиеся информацией, записанной по такому адресу.

*Теорема 2.* Автомат памяти приведенного вида содержит $|D|^{|ADR|}$ состояний.

Учитывая леммы 2 и 3, построим неинициальный аппарат памяти такой, что переменная его состояния $q$ описывается адресным деревом. Ветки этого дерева соответствуют адресам из $ADR$. Каждый узел адреса маркируется хранимым информационным символом из $D$.



Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru




При наличии кодировки (упорядочивания) адресов $ADR : ADR \mapsto [1, |ADR|] \subset N$ общее состояние автомата q определяется упорядоченным по адресу набором данных $q = (d_1, d_2, ..., d_{Adr})$, так как по теореме 1 набор соответствующих операций записи образуют базис достижимости. Переменную q при наличии кодирования символов данных $D : D \mapsto [1, |D|] \subset N$ можно также записать как полином:

$$q = (d_0 - 1)^* |D|^0 + (d_1 - 1)^* |D|^1 + ... + (d_{Adr} - 1)^* |D|^{Adr-1}, \quad (5)$$

где $q \in [0, |D|^{|ADR|} - 1]$; $d_i \in [1, |D|] \subset N$.

Очевидно, что $|Q| = |D|^{|ADR|}$. Эта часть теоремы доказана.

Множество состояний Q построено, входные алфавиты A и B определены в начале статьи, построим функции перехода φ и отображения выходного символа ψ.

Определим функцию изменения состояния $q = q(Adr, d)$ как выбор из корня указанной Adr ветки адреса и перемаркировку в узле адреса, сооветствующую заданному значению символа информации d. Тогда:

$q' = \varphi(q, \alpha(d, adr, c))$. Если $c = RD$, то $q' = q$. Если $c = WR$, то $q' = q(adr, d)$.

$b = \psi(q, \alpha(d, adr, c))$. Если $c = WR$, то $b = d$. Если $c = RD$, то по adr находим узел дерева и записанный в нем символ и этот символ выдается как значение функции ψ.

Проверим, что построеный аппарат удовлетворяет свойствам (1–4).

Свойство 1. Так как $c = RD$, то $q' = q$, сколько раз бы операция не производилась.

Свойство 2. Предшествующая операция записи не трогает маркировку символа в читаемом узле.

Свойство 3. Так как изменение состояния и выдача выходного символа строится исключительно по входному символу, то значение выходного символа не зависит от предшествующих входных символов и состояния автомата.

Свойство 4. Очевидно.

Согласно лемме 5 выполнение свойств (1–4) однозначно определяет передаточную функцию f. Построенный аппарат отвечает (1–4), и, следовательно, произвольный абстрактный аппарат памяти после его инициализации будет неотличим от построенного.

Полученный автомат неприводим. Доказательство аналогично приведенному в лемме 2.



Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Согласно теореме 2.2 [1] множество неотличимых автоматов от данного автомата приведенного вида имеет мощность 1 с точностью до изоморфизмов. Таким образом, построен аппарат, моделирующий работу памяти ЭВМ.

*Замечание 1*. Если для двух информационных алфавитов $D_1$, $D_2$ автоматов $V_1 \in MEM(D_1, ADR_1, CTL), V_2 \in MEM(D_2, ADR_2, CTL)$ существует гомоморфизм $\mu : D_1^n \mapsto D_2$, факторизующий адресное множества $ADR_1$, то возможно построение «упаковывающего» преобразования, размещающего $n$ символов автомата $V_1$ в одном символе автомата $V_2$. Образ этого преобразования занимает $|ADR_1|/n$ адресного пространства автомата $V_2$. Если $|ADR_1|/n \lhd |ADR_2|$, то автомат $V_1$ можно *разместить* в автомате $V_2$. Это свойство лежит в современных операционных системах в основе метода построения виртуальной памяти, отображаемой на жесткий диск.

*Замечание 2*. Определим функцию проектирования $Prj$: $ADR * A \mapsto A$. Результатом действия этой функции будет символ, повторяющий входящий, если он содержит операцию, применяющуюся к указанному адресу, иначе – пустой символ.

Если $Adr(a) = adr$, то $Pr\,j(adr, a) = a$; иначе $\Lambda$. (6)

Аналогично, для цепочек данная операция формирует подцепочки, составленные только из символов, применимых для указанного адреса. $Prj$: $ADR * A^* \mapsto A^*$.

Очевидно, что

$$\forall (c = CTL, k \in ADR, x \in D, q \in Q, a \in A^*),$$
$$\alpha = \alpha(x,k,c) : \psi\_(q, Pr\,j(k, a\alpha)) = \psi\_(q, a\alpha). \quad (7)$$

Операция проектирования может быть применена к некоторой группе адресов из $ADR$, оставляя в силе (7), где уже $k \subset ADR$.

Из (7) следует, что для аппарата, получаемого сужением множества ADR также выполняются операции (1–4), что позволяет его также рассматривать как аппарат памяти.

*Следствие:* для моделирования (эмуляции) памяти ЭВМ той же разрядности (того же алфавита множества $D$), хост-машине необходимо иметь множество адресов $ADR$, превосходящее множество адресов эмулируемой ЭВМ.

Из возможности моделирования $ADR^{хост-машины} > ADR^{эмулируемая}$ следует $\exists V' \subset V_{mem}^{хост} : V' \xrightarrow{гомоморф} V_{mem}^{эмулируемая}$.

Учитывая замечание 1, данное неравенство может быть справедливо для ЭВМ с меньшим обхемом физической памяти, но в которых аппарат подкачки (swapping) операционной системы или написанный программно дает необходимое количество виртуальной памяти в адресное пространство задачи.

*Замечание 3.* Как множество автоматов памяти можно рассматривать объекты хранения информации ОС: файлы, диски, ленты и т. д. Здесь информационным символом является байт в общем случае, два байта для Unicode текстов, 512 байт для диска и т. д. Адресом в этом случае может быть соответственно относительная позиция символа в файле, комбинация <цилиндр, дорожка, сектор>, отображаемая в $N$ (или так называемый $LBA$ адрес блока) для дис-


Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

ка и т.д. Описанные в данной статье методы будут работать для данных объектов.

## 2. Операции с автоматами памяти

*Определим операцию сложения автоматов памяти по адресам* $V = V_1 \overset{a}{+} V_2$.

Пусть $V = (A, Q, B, \varphi, \psi)$, $V_1 = (A_1, Q_1, B, \varphi_1, \psi_1) \in MEM(D_1, ADR, CTL)$ и $V_2 = (A_2, Q_2, B, \varphi_2, \psi_2) \in MEM(D_2, ADR, CTL)$. Предположим, $ADR_1 \cap ADR_2 = 0$. Тогда определим $A = D * ADR * CTL$. Следовательно, $A_1 \cap A_2 = 0$, а $A = D * ADR * CTL = A_1 \cup A_2$; $|ADR| = |ADR_1| + |ADR_2|$. Будем считать, что при поступлении на вход символа из $\mathrm{A}_1$ работает аппарат $\mathrm{V}_1$, а при поступлении на вход символа из $\mathrm{A}_2$ работает аппарат $\mathrm{V}_2$ ($Q = Q_1 \otimes Q_2$):

$\varphi(q_1 \otimes q_2, \alpha) = \varphi_1(q_1, \Pr j(A_1, \alpha)) \otimes \varphi_2(q_2, \Pr j(A_2, \alpha))$. Переход в новое состояние в аппаратах $\mathrm{V}_1$ и $\mathrm{V}_2$ при поступлении пустого входного символа не производится.

$\psi(q_1 \otimes q_2, \alpha) = \psi_1(q_1, \Pr j(A_1, \alpha))\, \psi_2(q_2, \Pr j(A_2, \alpha))$. В ответ на пустой символ, автомат формально дает тоже пустой символ. Следовательно, один из двух выходных символов пуст, так что их конкатенация дает символ из $\mathrm{B}$.

Покажем, что $\mathrm{V}$ удовлетворяет свойствам (1–4).

Свойство 1. Рассмотрим цепочку $\mathrm{a}\alpha'\alpha$.

Пусть $\{\alpha, \alpha'\} \subset A_1$ и представляют собой сиволы операций чтения. Тогда согласно (7) $\psi\_(q, a\alpha'\alpha) = \psi\_(q, \Pr j(A_1, a\alpha'\alpha))$. Но на множестве $\mathrm{A}_1$ $\psi\_(q, a\alpha'\alpha) = \psi^1\_(q, a\alpha'\alpha)$, а для автомата $\mathrm{V}_1$ формула (1) дает $\psi^1\_(q, a\alpha'\alpha) = \psi^1\_(q, a\alpha)$. Таким образом, $\psi\_(q, a\alpha'\alpha) = \psi\_(q, (A_1, a\alpha))$ и (1) для $\mathrm{V}$ в этом случае верно.

Пусть $\alpha' \in A_2, \alpha \in A_1$. Тогда $\Pr j(A_1, a\alpha'\alpha) = \Pr j(A_1, a\alpha)$. И аналогично получается: $\psi\_(q, a\alpha'\alpha) = \psi\_(q, \Pr j(A_1, a\alpha'\alpha)) = \psi\_(q, \Pr j(A_1, a\alpha)) = \psi\_(q, a\alpha)$.

Случаи $\alpha \in A_2, \alpha' \in A$ рассматриваются совершенно аналогично.

Доказательство применимости свойств (2–4) производится аналогично.

Таким образом, $V \in MEM(D, ADR, CTL)$, $|Q| = |Q_1| * |Q_2| = |D|^{|ADR_1| + |ADR_2|}$.

*Определим операцию умножения автоматов памяти по данным* $V = V_1 \overset{d}{*} V_2$.

Пусть $V = (A, Q, B, \varphi, \psi)$, $V_1 = (A_1, Q_1, B, \varphi_1, \psi_1) \in MEM(D_1, ADR, CTL)$ и $V_2 = (A_2, Q_2, B, \varphi_2, \psi_2) \in MEM(D_2, ADR, CTL)$. Предположим, что $D_1 \cap D_2 = 0$. Тогда определим $D = D_1 \cup D_2; Q = Q_1 \otimes Q_2$. Следовательно, $A_1 \cap A_2 = 0$, $A = D * ADR * CTL = A_1 \cup A_2; |D| = |D_1| * |D_2|$. И будем считать, что при поступлении на вход символа из $\mathrm{A}_1$ работает аппарат $\mathrm{V}_1$, а при поступлении на вход символа из $\mathrm{A}_2$ работает аппарат $\mathrm{V}_2$, а результаты их работы соединяются, чтоб получить выходной символ из $\mathrm{D}$.



Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

$\varphi(q,\otimes q_2,\alpha) = \varphi_1(q_1,\Pr j(A_1,\alpha)) \otimes \varphi_2(q_2,\Pr j(A_2,\alpha))$,

$\psi(q,\otimes q_2,\alpha) = \psi_1(q_1,\Pr j(A_1,\alpha)) \otimes \psi_2(q_2,\Pr j(A_2,\alpha))$.

Проверка свойств (1–4) основываются на том факте, что операции проектирования и умножения коммутируют друг с другом.

Таким образом,

$$V \in MEM(D,ADR,CTL),$$

$$|Q| = |D_1|^{|ADR|} * |D_2|^{|ADR|} = |Q_1| * |Q_2|.$$

*Замечание 1.* Для создания законченной картины аналогично можно определить операции умножения по адресу и сложения по данным. Можно доказать, что полученный результат также будет лежать в классе автоматов с соответствующими параметрами. Но исследование этих операций на настоящий момент автору представляется не интереным. Поэтому далее введенные операции сложения по адресам и умножения по данным будем называть просто сложением и умножением соответственно.

Рассмотрим множество автоматов `MEM(D,CTL)` с фиксированными `D` и `CTL` и произвольным `ADR`: $MEM(D,CTL) = \bigcup_{ADR} MEM(D,ADR,CTL)$ Назовем метрикой $d : MEM(D,CTL)^2 \mapsto N$ мощность множества различий адресов автоматов. То есть

$$d(V_1,V_2) = |Adr(V_1) \setminus Adr(V_2)| + |Adr(V_2) \setminus Adr(V_1)|. \quad (8)$$

Проверим для этого следующее [2]:

1. $d(V_1,V_2) = 0 \Leftrightarrow V_1 = V_2$. Если $V_1 = V_2$, то оба слагаемых в правой части (8) равны нулю. Если $d(V_1,V_2) = 0$, то $\{V_1,V_2\} \subset MEM(D,ADR,CTL)$, который согласно т. 2.2 [1] содержит единственный с точностью до изоморфизма элемент. Следовательно $V_1 = V_2$.

2. $d(V_1,V_2) \le d(V_1,V_3) + d(V_2,V_3)$. Обозначим $A \Delta B = (A \setminus B) \cup (B \setminus A)$ Пусть для определенности $x \in V_1, x \notin V_2$. Обратный случай ($x \in V_2, x \notin V_1$) рассматривается совершенно аналогично.

Пусть $x \notin V_3$, тогда $x \in V_1 \Delta V_3$. Пусть $x \in V_3$, тогда $x \in V_2 \Delta V_3$. То есть $V_1 \Delta V_2 \subseteq (V_1 \Delta V_3) \cup (V_2 \Delta V_3) \Rightarrow |V_1 \Delta V_2| \le |(V_1 \Delta V_3) \cup (V_2 \Delta V_3)| \le |(V_1 \Delta V_3)| + |(V_2 \Delta V_3)|$.

Таким образом, `MEM(D,CTL)` — метрическое пространство и можно использовать понятие сходимости.

## 3. Использование построенных автоматов и операций.

Операции сложения по адресу интересны для создания объектов хранения информации супер большого объема, которые невозможно или не целесообразно размещать на одной ЭВМ. Для решения этой задачи на каждом узле операцией проектирования (6) создается описанный выше подавтомат памяти,



Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

которые затем собираются в масштабах кластера определенной выше операцией сложения по адресам в единый аппарат $V_{mem} = \sum_{i \in \text{узлы сети}}^{a} V_i$ .

Стандарт MPI [3] программно сопровождает операции работы с разделяемой памятью, поддерживаемые аппаратно в технологиях SCI (Scalable Coherent Interface), memory channel, InfiniBand (прямое использование RDMA - удаленного доступа в память процесса) [4]. MPI определяет операции RMA (remote memory access — удаленный доступ к памяти) по созданию окон в памяти процесса (`MPI_ALLOC_MEM`, `MPI_WIN_CREATE` и т. п.), которые можно разделять с другими процессами, состоящими в коммуникаторе параллельного вычисления, операциями `PUT`, `GET` и другими. Эти окна и являются базовыми «кирпичиками» построения распределения аппарата памяти. Для переадресации запроса к памяти в нужный узел, создается функция, которая преобразует адрес запроса в пару `<имя узла, адрес в окне памяти узла>` $\eta : ADR \mapsto < node, ADR_{node} >$.

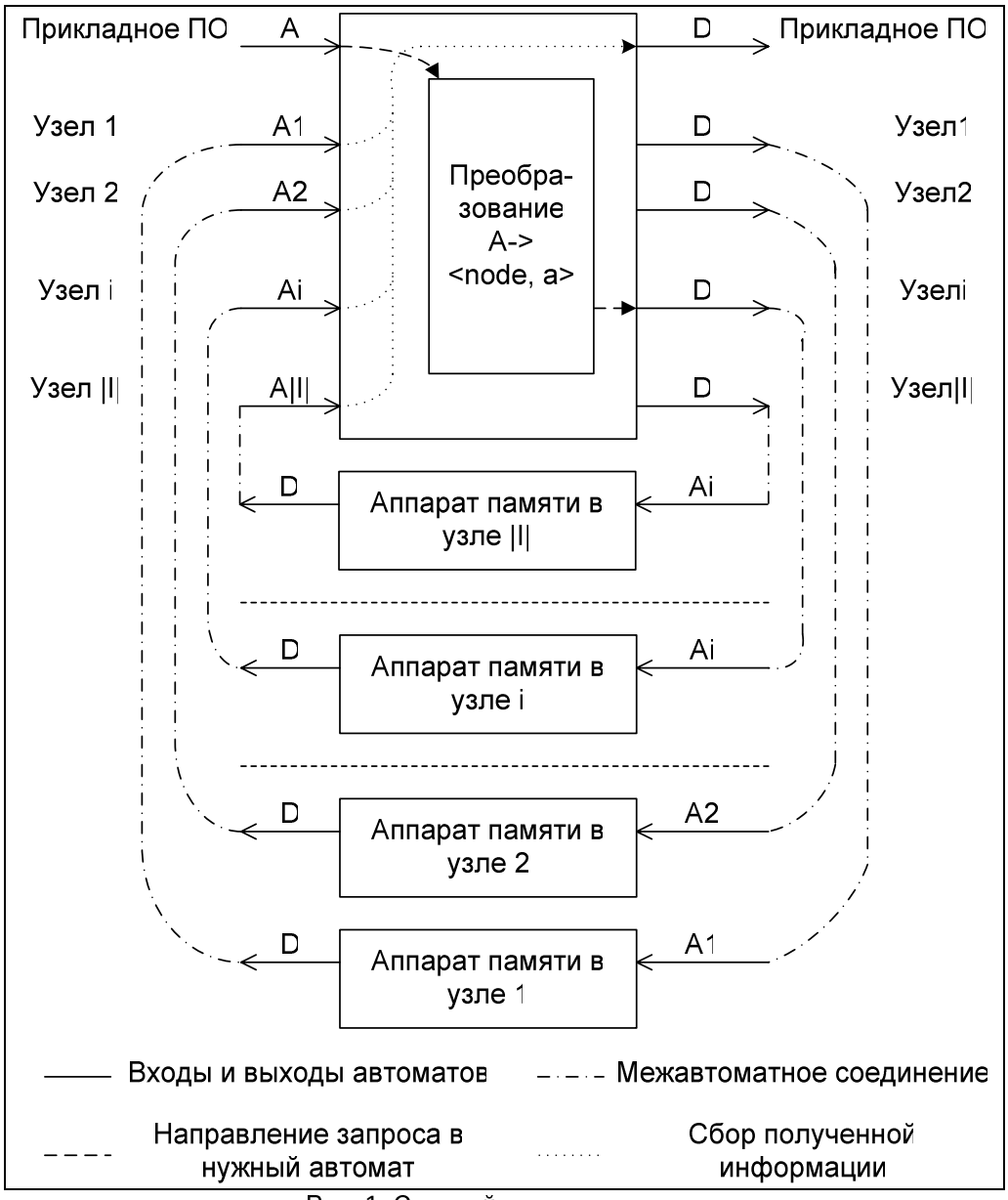



Рис. 1. Сетевой аппарат памяти.



Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Поддержка функции η производится в каждом узле, поддерживающим контент данной задачи. Управление работой этой функции может быть централизовано, например, размещением рабочих таблиц программных реализаций этих функции в разделяемой области памяти, доступной всем по чтению и только управляющему процессу на запись. Политика кэширования этой области может быть определена администратором задачи (кластера) или иным образом.

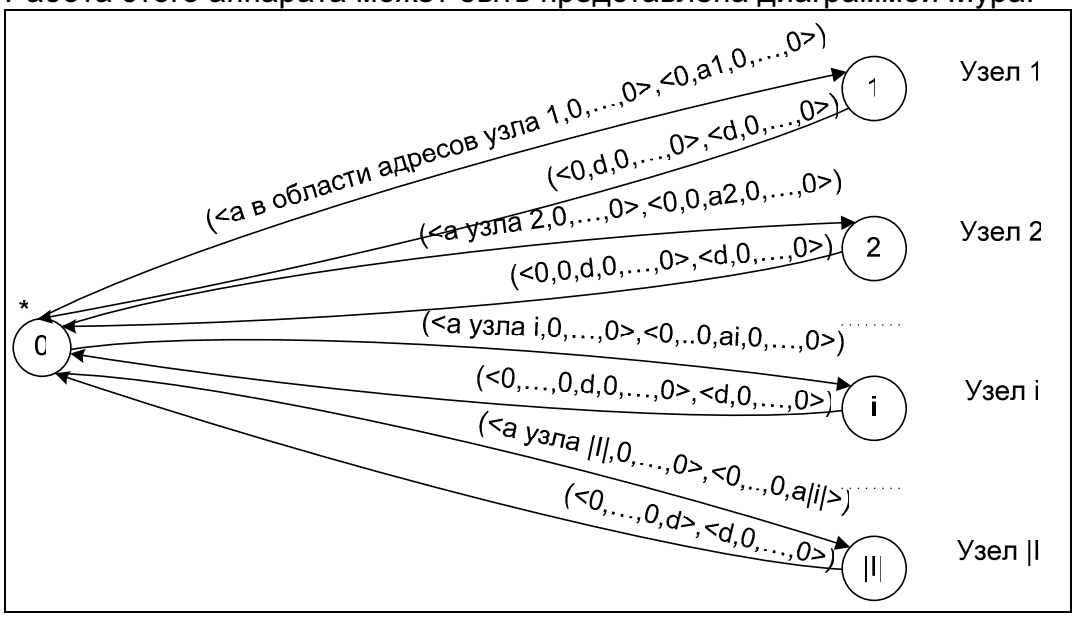
Пусть I множество индексов узлов кластера. Тогда аппарат, реализующий функции сетевого аппарата памяти для прикладного программного обеспечения $V = (A', Q', B', \varphi', \psi')$, может выглядеть следующим образом (см. рис. 1).

Пусть:

$A' = A \otimes \prod_{i \in I} A_i$, где A — входной символ запроса к собранному аппарату памяти, $A_i$ — алфавит входных символов аппарата i-го узла.

$B' = D \otimes \prod_{i \in I} D$ — предполагаем, что информация во всех автоматах организована одинаково.

Работа этого аппарата может быть представлена диаграммой Мура:




0 – начальное состояние этого автомата.

Модификация операторов PUT и GET MPI или введение собственных операторов для работы с этим автоматом предоставит необходимую программную поддержку.

Расширением задачи построения сетевого автомата памяти является сборка его в распределенной среде обработки информации типа ГРИД [5,6]. Сложность этой задачи заключается в том, что межузловые каналы связи в распределенной сети обычно имеют меньшую пропускную способность, чем в кластере, но зато таких узлов может быть собрано гораздо большее количество. Также распределенные сети могут предоставлять узлы низкой степени доступности, что означает относительно предыдущего случая большее время задержки исполнения обращения к памяти или даже возможную потерю данных. Одним из путей решения этих вопросов является использование репликации данных DataGrid [7]. В этом случае репликацию данных и нахождение узла, где



Труды СПИИРАН. 2005. Вып. 2. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2005. Issue 2. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

производится размещение информации, обеспечивает системное ПО Грид, но эффективность доступа при этом сильно снижается. Можно использовать ручное создание последовательностей аппаратов памяти, объединяемых вручную, пока он не сойдутся по определенной выше метрике к нужному аппарату. Этот «аппроксимирующий» процесс будет представлять собой динамическую задачу отслеживания состояния собранного аппарата памяти и дособирания его по мере необходимости. Временную динамику построения и сопровождения такого автомата удобно рассматривать как временную последовательность аппаратов памяти $V_t$, сходящихся по метрике (8) к требуемому. Каждый такой автомат $V_t$ представляется в виде суммы узловых автоматов, его составляющих $V_t = \sum_{i \in I}^{a} V_i$, где I — индекс подключенных узлов. Изменения в данном случае претерпевает индекс, который указывает на различные узлы в разное время.

В соответствии с физиологией ГРИД [6], узловые автоматы памяти могут быть реализованы как сервисы ГРИД. Процесс нахождения свободных аппаратов и их объем может быть сделан средствами штатного сервиса регистрации ГРИД. При проектировании сервиса аппаратов памяти, представляется целесообразным указание в реестре сервисов объема памяти и качества предоставляемых услуг, например, в виде указания времени отклика на запрос. Эти данные можно формировать динамически на экспериментальной основе. Выбр оптимального сервиса по запросу <объем памяти, время отклика> из доступных в первом приближении также можно сделать средствами сервиса регистрации ГРИД.

## Литература


[1] *Кудрявцев В. Б., Алешин С. В., Подколзин А. С.* Введение в теорию автоматов М.: Наука, 1985. — 320 с.

[2] *Корн Г., Корн Т.* Справочник по математике (для научных работников и инженеров), 4-е изд. М.: Наука, 1977. — 832 с.

[3] The Message Passing Interface Standard <http://www-unix.mcs.anl.gov/mpi/index.html>.

[4] Новости мира высокопроизводительных вычислений. Лаборатория Параллельных Информационных Технологий, НИВЦ МГУ <www.parallel.ru>.

[5] *Foster I., Kesselman C., Tuecke S.*, The Anatomy of the Grid: Enabling Scalable Virtual Organizations // Supercomputer Applications N 15(3), 2001. — 25 с.

[6] *Foster I., Kesselman C., Nick J., Tuecke S.*, The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure <http://www.globus.org/research/papers.html/> рабочая группа, Global Grid Forum, 2002. — 31 с.

[7] Globus: Data GRID architecture <http://www.globus.org/datagrid/architecture.html/>.