Министерство образования Российской Федерации
Ярославский государственный университет
имени П.Г.Демидова

# МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ

Том 8 №1 2001

Основан в 1999 г.
Выходит 2 раза в год

*Свидетельство о регистрации №019209 от 16.08.99*
*Государственного Комитета Российской Федерации по печати*



**Адрес редакции:** 150000, Ярославль, ул.Советская, 14
**E-mail:** mais@uniyar.ac.ru

Научные статьи в журнал принимаются на кафедре ТИ. Статья должна содержать УДК, аннотацию и сопровождаться набором текста в редакторе LaTeX.

# СОДЕРЖАНИЕ

*Моделирование и анализ информационных систем. Т.8, №1. 2001*



Лицензия ЛР №020319 от 30.12.96. Корректор А.А.Аладьева
Подписано в печать 30.09.2001. Формат 60x88$^1/_8$. Печать офсетная.
Усл.печ.л. 7,61. Уч.-изд.л. 4,5. Тираж 100 экз. 266

Отпечатано на ризографе. Ярославский государственный университет имени П.Г.Демидова, 150 000, Ярославль, ул.Советская, 14

*Модел. и анализ информ. систем.* Т.**8**, №**1** (2001) 3–8

УДК 519.248.254

# Об одной "мере центральности" элементов конечного множества в $R^n$

Иванов А.В.
*Ярославский государственный университет*
*150 000, Ярославль, Советская, 14*

В работе исследуется один аффинно-инвариантный подход к определению "меры центральности" элементов конечного множества в $R^n$. Данный подход находит применение, в частности, в робастной статистике. В работе введено определение глубины точки и множества в $R^n$. Большая часть работы посвящена множествам точек одинаковой глубины — их исследованию и построению. Данный подход имеет определенные преимущества по сравнению с подходом, предложенным Тьюки.

## Введение

1. Данная работа посвящена одной "мере центральности" точек конечных множеств в $R^n$. Сама эта тема возникла в связи с некоторыми аспектами многомерной робастности.

Как известно, качество классических статистических выводов заметно снижается при засорении выборки (и тому подобных отклонениях от исходных предпосылок). В связи с этим за последние десятилетия интенсивно развивалась робастная статистика [1], [2], процедуры которой устойчивы в таких ситуациях.

В одномерном случае распространенным приемом является предварительное исключение по $k$ ($k < n/2$) крайних членов вариационного ряда с каждой стороны, после чего можно применять обычные схемы обработки. Здесь не возникает вопроса, какие именно элементы выборки считать крайними. В многомерном случае ситуация меняется. Здесь существуют различные подходы. Наибольший интерес представляют аффинно-инвариантные методы.

Исходя из предположения о непрерывности выборки, можно считать, что все элементы выборки различны и находятся в общем положении, то есть никакие $n+1$ элементов не лежат в одной $(n-1)$-плоскости (тем самым мы пренебрегаем событием нулевой вероятности).

Один из аффинно-инвариантных методов определения "крайних" точек, состоящий в процедуре, названной "лущением", был предложен Тьюки (см. [1], [3]). Пусть $A = \{p_1, p_2, \ldots, p_N\}$ – исходная выборка в $R^n$. Построим выпуклую оболочку точек $p_1, p_2, \ldots, p_N$, и тем, которые попадут на границу этой оболочки, присвоим "глубину" 1. Далее, все точки глубины 1 удаляются из $A$, и к оставшейся части $A_1$ выборки $A$, если она непуста, применяется та же процедура (находим выпуклую оболочку $A_1$ и т. д.). До исчерпания $A$ все элементы $p_i$ выборки $A$ получат свои "глубины по Тьюки" $t(p_i)$. Точки с меньшей глубиной рассматриваются как более "крайние" (то есть менее надежные).

2. А. Ю. Левин заметил, что данная изящная процедура не вполне адекватна с чисто статистической точки зрения, и предложил другой подход к определению "надежности" точки, а именно, точка $p_i \in A$ имеет глубину $T(p_i) = k$, если любая гиперплоскость, проходящая через $p_i$, отсекает от $A$ не менее $k$ точек.

Недостаток процедуры Тьюки состоит в следующем. Для любого элемента $p_i \in A$ с $t(p_i) = 1$ всегда найдется такой линейный функционал $f(x) \not\equiv 0$, что $f(p_i)$ будет являться первым членом вариационного ряда чисел $f(p_1), f(p_2), \ldots, f(p_N)$ (обозначим этот ряд $\mathrm{Var} f(A)$), то есть, здесь $t(p_i)$ вполне соответствует, так сказать, скалярной глубине. Но оказывается, что при $t(p_i) > 1$, это, вообще говоря, не так. На рисунке 1 (слева) приведен пример выборки, для которой образ $f(p_i)$ любой точки $p_i$ с $t(p_i) = 2$ не будет принадлежать к двум первым членам $\mathrm{Var} f(A)$, каков бы ни был линейный функционал $f(x) \not\equiv 0$. (Вариационный ряд содержит все $N$ чисел $f(p_1), f(p_2), \ldots, f(p_N)$, расположенных в порядке неубывания). Итак, выделение точек с $t(p) = 1$ вполне оправдано, чего нельзя сказать об итеративном повторении процедуры.

Конструкция, предложенная А. Ю. Левиным, свободна от упомянутого дефекта. В самом деле, пусть существует такой линейный функционал $f(x) \not\equiv 0$, что $f(p_i)$ является $k$-м членом $\mathrm{Var} f(A)$. Тогда гиперплоскость $f(x) = f(p_i)$ отсекает как минимум $k$ точек от $A$ (включая точки из $A$, лежащие в указанной гиперплоскости). И наоборот, из существования гиперплоскости $f(x) = f(p_i)$, отсекающей $k$ точек от $A$,

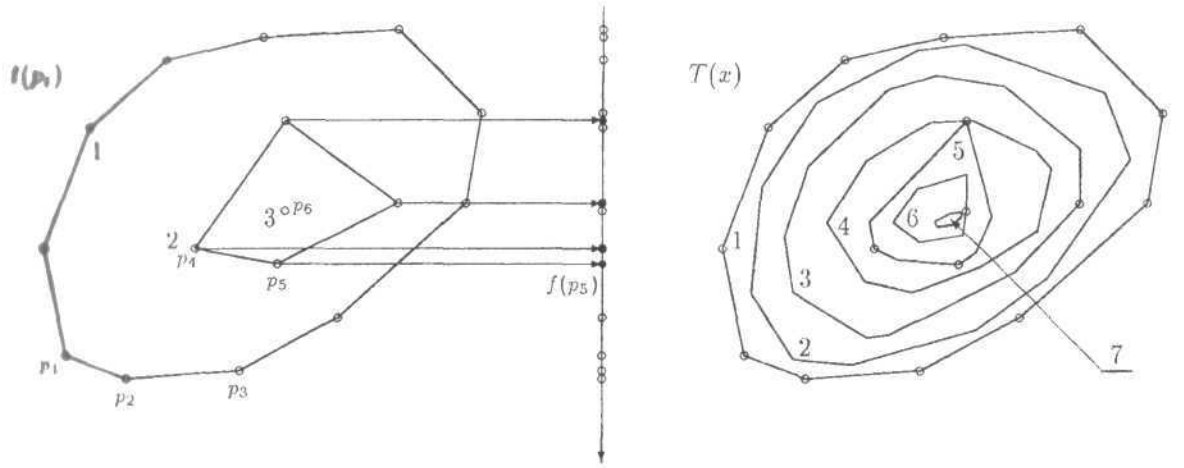


Рис. 1. Слева: глубина $t(p_i)$ точек некоторой выборки по Тьюки. Справа: множества $M_1 \supseteq M_2 \supseteq \ldots \supseteq M_r$ точек одинаковой глубины $T(x)$ для той же выборки.

следует существование линейного функционала ($f(x)$ или $-f(x)$), такого, что $k$-м членом $\mathrm{Var} f(A)$ будет являться значение $f(p_i)$.

Функцию глубины естественно также распространить на все $R^n$ — это позволит шире применять соображения выпуклости.

3. Исходя из данных соображений, для любой выборки $A$ в $R^n$ можно построить множества $M_0 \supseteq M_1 \supseteq \supseteq M_r$, удовлетворяющие условию: любая гиперплоскость, проведенная через любую точку $x \in M_k$, отсекает от множества $A$ не менее $k$ точек. См. рис. 1 (справа). Точки выборки, лежащие в $M_k$, могут рассматриваться как "более надежные", чем те, что лежат в $M_{k-1} \setminus M_k$.

Ясно, что процедура вычисления глубины $T(p)$ является более сложной, чем процедура "лущения", и для больших выборок в пространствах большой размерности может оказаться весьма трудоемкой и практически неприменимой. Впрочем, это относится, хотя и в меньшей степени, и к процедуре Тьюки, так как построение выпуклой оболочки в пространствах большой размерности также является трудоемкой задачей.

В данной статье исследуются геометрические и алгоритмические аспекты, связанные с множествами $M_k$. Предварительно уточним используемые определения и обозначения.

## Определения и обозначения

**Определение 1.** Точками, *отсеченными от множества* $A$ *гиперплоскостью* $h$, будем считать точки, лежащие в том из двух замкнутых полупространств, разделяемых гиперплоскостью $h$, где точек множества $A$ меньше. В том случае, если по обе стороны от гиперплоскости $h$ лежит одинаковое количество точек из $A$, то выбор одного из двух указанных полупространств может быть произвольным.

**Определение 2.** *Глубиной* $T(x)$ *точки* $x \in R^n$ во множестве $A$ будем называть такое число $k$, что $x \in M_k \setminus M_{k+1}$.

**Определение 3.** *Глубиной* $T(A)$ *множества* $A$ назовем такое число $k$, что $M_k \neq \emptyset$ и $M_{k+1} = \emptyset$.

Заметим, что $M_{T(A)}$ является в некотором смысле "центром" множества $A$. Ясно также, что $M_0 = R^n$.

Определение глубины $T(x)$ связано с определением глубины Тьюки следующим образом. Пусть $p_i \in A$ и $t(p_i) = k$. Тогда любая гиперплоскость, проходящая через $p_i$, отсекает от $A$ как минимум $k$ точек из $A$: по крайней мере по одной вершине каждой из $k$ выпуклых оболочек, построенных процедурой "лущения" и содержащих $p_i$. Значит, $T(p_i) \geq k$, то есть $T(p_i) \geq t(p_i)$ для любой точки $p_i \in A$. Кроме того, очевидно, что $M_1$ совпадает с выпуклой оболочкой множества $A$, а $M_2$ не содержит вершин этой выпуклой оболочки. Таким образом, если $p_i \in A$ и $t(p_i) = 1$, то $T(p_i) = 1$.

В данной работе мы будем использовать следующие обозначения.

Пусть $h$ — некоторая гиперплоскость в $R^n$. Тогда положим:

$H(h)$ — то из двух открытых полупространств, разделяемых гиперплоскостью $h$, в котором содержится меньше точек из $A$ (если оба указанных полупространства содержат одинаковое количество точек

из $A$, то в качестве $B(h)$ можно взять любое них);

$U(h) = R^n \setminus B(h)$;

$A_B(h) = A \cap B(h)$ — множество точек из $A$, отсеченных гиперплоскостью $h$ от $A$ и не лежащих в $h$;

$S(h) = |A_B(h)|$ — число точек множества $A$, лежащих в $B(h)$;

$C(A)$ — выпуклая оболочка множества $A$; $C(\{p_1, p_2, \ldots, p_n\})$ — выпуклая оболочка точек $p_1, p_2, \ldots, p_n$.

И, наконец, последние два обозначения.

Пусть заданы произвольная гиперплоскость $h$, $(n-2)$-плоскость $l \subset h$ и выбрано одно из двух возможных направлений "вращения" $h$ вокруг $l$. Для наглядности мы используем терминологию, применяемую в пространствах малых размерностей: в $R^3$, например, можно говорить о вращении плоскости вокруг прямой как оси этого вращения — так как смысл этой терминологии очевиден: дело сводится к углу поворота нормали к $h$ в соответствующей 2-плоскости. Тогда положим:

$h(\phi, l)$ — гиперплоскость, возникающая при повороте $h$ вокруг $l$ на угол $\phi$ $(0 \le \phi < \pi)$ в заданном направлении;

$X(h, l, \phi_1, \phi_2) = \bigcup\limits_{\phi_1 < \phi < \phi_2} h(\phi, l) \setminus l$.

## Конструктивное описание множеств $M_k$

**Постановка задачи.** *Дано конечное множество точек общего положения $A \subset R^n$, $|A| = N, 1 \le n < N$. Построить $M_k \subset R^n$ — множество точек, удовлетворяющих условию: любая гиперплоскость, проведенная через любую точку $x \in M_k$, отсекает от $A$ не менее $k$ точек, где $k$ — натуральное число.*

Докажем сначала одно простое свойство $M_k$.

**Утверждение 1.** *Множества $M_k$ являются выпуклыми.*

**Доказательство.** Предположим противное: множество $M_k$ не выпуклое. Тогда существуют такие три точки $a$, $b$ и $c$, что $a \in M_k$, $b \in M_k$, $c \in (a, b)$ и $c \notin M_k$. Так как $c \notin M_k$, то существует проходящая через точку $c$ гиперплоскость $h$, отсекающая от $A$ менее $k$ точек.

Точки $a$ и $b$ не могут лежать в $h$, так как в этом случае они не принадлежали бы множеству $M_k$ (так как $h$ отсекает менее $k$ точек от $A$). Следовательно, точки $a$ и $b$ лежат в разных открытых полупространствах, разделяемых гиперплоскостью $h$. Пусть, для определенности, $a \in B(h)$. Тогда гиперплоскость $h'$, параллельная $h$ и проходящая через $a$, отсекает от $A$ менее $k$ точек, то есть $a \notin M_k$. Получили противоречие. Утверждение 1 доказано.

Далее будет дана более глубокая характеристика множеств $M_k$, в частности, будет показано, что $M_k$ являются многогранниками (если $M_k \ne \emptyset$ и $M_k \ne R^n$), и дан способ построения данных множеств.

Основным результатом данной статьи является следующее утверждение.

**Теорема.**

*1. Если $k > \lfloor (N-n)/2 + 1 \rfloor$, то $M_k = \emptyset$.*

*2. Для любого $k$, такого, что $0 < k \le \lfloor (N-n+1)/2 \rfloor$, множество $M_k$ является пересечением всех замкнутых полупространств, обладающих следующими двумя свойствами:*

*полупространство содержит $N - (k-1)$ точку множества $A$;*

*граничная гиперплоскость данного полупространства содержит $n$ точек множества $A$.*

**Доказательство.**

1. Покажем, что $M_k = \emptyset$ при $k > \lfloor (N-n)/2 + 1 \rfloor$. Существует гиперплоскость $h$, проходящая через некоторые $n$ точек $p_1, p_2, \ldots, p_n \in A$, и разделяющая два открытых полупространства, которые при четном $(N-n)$ содержат по $(N-n)/2$ точек из $A$, а при нечетном $(N-n)$ — соответственно $(N-n-1)/2$ и $(N-n+1)/2$ точек из $A$. Ясно, что любая гиперплоскость, параллельная $h$, не может отсекать от $A$ более $(N-n+1)/2 \le \lfloor (N-n)/2 + 1 \rfloor$ точек, то есть $M_{\lfloor (N-n)/2+1 \rfloor + 1} \setminus h = \emptyset$.

Рассмотрим гиперплоскость $h$. Через любую точку $x \in h$ можно провести $(n-2)$-плоскость $l \subset h$, которая разобьет $h$ на замкнутую $(n-1)$-полуплоскость $h_1$ и открытую $(n-1)$-полуплоскость $h_2$ таким образом, что $h_1$ будет содержать не более одной точки из множества $\{p_1, p_2, \ldots, p_n\} \subset A$. Очевидно, что можно подобрать такой достаточно малый угол $\phi$, что $A \cap (X(h, l, 0, \phi) \cup (h(\phi, l) \setminus l)) = \emptyset$, то есть гиперплоскость $h$ в процессе вращения от первоначального положения до совпадения с $h(\phi, l)$ не пройдет ни через одну точку из $A$, не лежащую в $l$. Направление вращения при нечетном $(N-n)$ выбирается таким образом, чтобы точки из $A \cap h_2$ оказались в $U(h(\phi, l))$, а при четном $(N-n)$ направление вращения не имеет значения. Гиперплоскость $h(\phi, l)$ отсекает от $A$ не более $(N-n)/2+1$ точки при четном $(N-n)$, и не более $(N-n-1)/2+1$ точки при нечетном $(N-n)$. Следовательно, $M_{\lfloor (N-n)/2+1 \rfloor + 1} \cap h = \emptyset$.

Учитывая, что $M_{k+1} \subseteq M_k$, получаем, что если $k > \lfloor (N-n)/2 + 1 \rfloor$, то $M_k = \emptyset$.

2. Для доказательства теоремы для случая $0 < k \le \lfloor (N-n+1)/2 \rfloor$ приведем несколько вспомогательных лемм.

**Лемма 1.** *Пусть $x \in R^n$, $h$ — произвольная гиперплоскость, проходящая через $x$, и $S(h) \le \lfloor (N-n)/2 \rfloor$. Тогда существует гиперплоскость $h'$, проходящая через $x$ и некоторые точки $p_1, p_2, \ldots, p_{n-1} \in A$, такие, что $x, p_1, p_2, \ldots, p_{n-1}$ не лежат в одной $(n-2)$-плоскости и $A_B(h') \subseteq A_B(h)$.*

**Доказательство.** Сначала докажем индукцией по $j$ утверждение, что для каждого $j$ $(0 \le j \le n-1)$ существует гиперплоскость $h_j$, проходящая через $x$ и некоторые точки $p_1, p_2, \ldots, p_j \in A$, такие, что $x, p_1, p_2, \ldots, p_j$ не лежат в одной $(j-1)$-плоскости, и $A_B(h_j) \subseteq A_B(h)$, а затем положим $h' = h_{n-1}$.

При $j = 0$ утверждение верно, так как можно положить $h_0 = h$ (очевидно).

Пусть существование гиперплоскости $h_{j-1}$ $(1 \le j \le n-1)$ уже доказано. Тогда, если существует такая точка $p_j \in A$, что $p_j \in h_{j-1}$, и точки $x, p_1, p_2, \ldots, p_{j-1}, p_j$ не лежат в одной $(j-1)$-плоскости, то положим $h_j = h_{j-1}$. В противном случае рассмотрим произвольную $(n-2)$-плоскость $l \subset h_{j-1}$, проходящую через $x, p_1, p_2, \ldots, p_{j-1}$. Очевидно, $l$ существует.

*Идея дальнейшего построения интуитивно проста и состоит в следующем.* Будем поворачивать $h_{j-1}$ вокруг $l$ до тех пор, пока она впервые не пройдет через точку из $A \setminus h_{j-1}$. Полученную гиперплоскость и выберем в качестве $h_j$. Приведем строгое доказательство этого факта.

Выберем произвольное направление вращения $h_{j-1}$ вокруг $l$. Для каждой точки $p^i \in A \setminus h_{j-1}$ найдется $h_{j-1}(\phi^i, l)$, проходящая через $p^i$. Обозначим $\phi_{min} = \min\{\phi^i : 1 \le i \le |A \setminus h_{j-1}|\}$. Ясно, что $X(h_{j-1}, l, 0, \phi_{min}) \cap A = \emptyset$, так как в противном случае угол $\phi_{min}$ не был бы минимальным среди всех $\phi^i$. Так как $p_1, p_2, \ldots, p_{j-1} \in l \subset h_{j-1}(\phi_{min}, l)$, $S(h_{j-1}) \le S(h) \le \lfloor (N-n)/2 \rfloor$ и $X(h_{j-1}, l, 0, \phi_{min}) \cap A = \emptyset$, то $A_B(h_{j-1}(\phi_{min}, l)) \subseteq A_B(h_{j-1}) \subseteq A_B(h)$. Следовательно, можно положить $h_j = h_{j-1}(\phi_{min}, l)$. Лемма 1 доказана.

**Лемма 2.** *Пусть $h$ — гиперплоскость, проходящая через $p_1, p_2, \ldots, p_n \in A$, и $S(h) < \lfloor (N-n)/2 \rfloor$. Тогда для любого $m$, такого, что $S(h) < m \le \lfloor (N-n)/2 \rfloor$, существует гиперплоскость $h'(m)$, проходящая через $p_1, p_2, \ldots, p_{n-1}$ и некоторую точку $p_{i(m)} \in A$ $(n+1 \le i(m) \le N)$, такая, что $S(h'(m)) = m$ и $p_n \in B(h'(m))$.*

**Доказательство.** Пусть $l$ — $(n-2)$-плоскость, проходящая через $p_1, p_2, \ldots, p_{n-1}$. Для каждой точки $p_{n+t} \in A \setminus h$ $(1 \le t \le N-n)$ найдется гиперплоскость $h(\phi_t, l)$, проходящая через $p_{n+t}$. Пусть точки из $A \setminus h$ занумерованы таким образом, что $\phi_1 < \phi_2 < \ldots < \phi_{N-n}$ при таком направлении вращения $h$ вокруг $l$, что $p_n \in B(h(\phi_1, l))$. Возможность выбора такого направления вращения достаточно очевидна, но для полноты доказательства приведем обоснование этого факта.

Пусть $r$ — расстояние от $p_n$ до $l$; $v$ — вектор нормали к $h$, направленный в $U(h)$; $p_n'$ — точка, получающаяся в результате переноса $p_n$ на расстояние $r$ в направлении $v$. Из двух возможных направлений вращения $h$ вокруг $l$ выберем то, при котором гиперплоскость $h(\pi/4, l)$ проходит через $p_n'$. Покажем, что при этом направлении вращения $p_n \in B(h(\phi_1, l))$. Рассмотрим два случая.

1) $S(h(\phi_1, l)) = (N-n)/2$. Тогда в качестве $B(h(\phi_1, l))$ можно взять любое из открытых полупространств, разделяемых гиперплоскостью $h(\phi_1, l)$, в частности, то, которое содержит $p_n$.

2) $S(h(\phi_1, l)) < (N-n)/2$. Так как $\phi_1 = \min\{\phi_1, \phi_2, \ldots, \phi_{N-n}\}$, то $X(h, l, 0, \phi_1) \cap A = \emptyset$. Следовательно, если $p_n \notin B(h(\phi_1, l))$, то $A \subseteq A_B(h) \cup A_B(h(\phi_1, l)) \cup \{p_1, p_2, \ldots, p_{n+1}\}$ и $|A| \le S(h) + S(h(\phi_1, l)) + n + 1$. Если $N-n$ четно, то $S(h) \le (N-n)/2 - 1$ и $S(h(\phi_1, l)) \le (N-n)/2 - 1$, а если $N-n$ нечетно, то $S(h) \le (N-n-1)/2 - 1$ и $S(h(\phi_1, l)) \le (N-n-1)/2$, откуда следует, что $|A| \le N-1$. Но это невозможно, так как $|A| = N$. Таким образом, $p_n \in B(h(\phi_1, l))$.

Пусть при некотором $t$ $(0 \le t \le N-n)$ выполняется неравенство $S(h(\phi_t, l)) < \lfloor (N-n)/2 \rfloor$. Для единообразия записи положим $h(\phi_0, l) = h(\phi_{N-n+1}, l) = h$, $\phi_0 = 0$ и $\phi_{N-n+1} = \pi$. Так как $X(h, l, \phi_t, \phi_{t+1}) \cap A = \emptyset$, то выполняются включения: $A_B(h(\phi_t, l)) \setminus p_{n+t+1} \subseteq A_B(h(\phi_{t+1}, l)) \subseteq A_B(h(\phi_t, l)) \cup p_{n+t}$, откуда следует, что $|S(h(\phi_{t+1}, l)) - S(h(\phi_t, l))| \le 1$. Из приведенного здесь неравенства также следует, что если при некотором $t$ $(1 \le t \le N-n-1)$ выполняется неравенство $S(h(\phi_t, l)) < \lfloor (N-n)/2 \rfloor$ и $p_n \in B(h(\phi_t, l))$, то $p_n \in B(h(\phi_{t+1}, l))$. Таким образом доказано, что утверждение леммы верно при всех $m$, таких, что $S(h) < m \le \max\{S(h(\phi_t, l)) : 1 \le t \le N-n\}$. (Ясно, что $\max S(h(\phi_t, l)) \le \lfloor (N-n)/2 \rfloor$.) Гиперплоскостью $h'(m)$ является одна из гиперплоскостей $h(\phi_t, l)$.

Покажем, что $\max S(h(\phi_t, l)) = \lfloor (N-n)/2 \rfloor$. Предположим противное: $\max S(h(\phi_t, l)) < \lfloor (N-n)/2 \rfloor$. Тогда, как показано выше, $p_n \in B(h(\phi_1, l)), p_n \in B(h(\phi_2, l)), \ldots, p_n \in B(h(\phi_{N-n}, l))$. Но тогда $A \subseteq A_B(h) \cup A_B(h(\phi_{N-n}, l)) \cup \{p_1, p_2, \ldots, p_{n-1}\} \cup p_N$, откуда при четном $N-n$ имеем: $S(h) \le (N-n)/2 - 1$, $S(h(\phi_{N-n}, l)) \le (N-n)/2 - 1$ и $|A| \le N-2$, а если $N-n$ нечетно, то $S(h) \le (N-n-1)/2 - 1$, $S(h(\phi_{N-n}, l)) \le (N-n-1)/2 - 1$ и $|A| \le N-3$, что невозможно, так как $|A| = N$. Следовательно, $\max S(h(\phi_t, l)) = \lfloor (N-n)/2 \rfloor$. Лемма 2 доказана.

Ниже приведены три варианта леммы 2, возникающие, когда вместо точки $p_n \in A$ рассматривается точка $x \notin A$. В зависимости от взаимного положения $x$ и точек из $A$, данные утверждения нуждаются в соответствующих изменениях.

**Лемма 2'.** *Пусть $h$ — гиперплоскость, проходящая через точки $x \notin A$ и $p_1, p_2, \ldots, p_{n-1} \in A$, $S(h) \le \lfloor (N-n)/2 \rfloor$, $x, p_1, p_2, \ldots, p_{n-1}$ не лежат в одной $(n-2)$-плоскости и $h$ содержит ровно $n-1$ точку из $A$. Тогда для любого $m$, такого, что $S(h) \le m \le \lfloor (N-n)/2 \rfloor$, существует гиперплоскость $h'(m)$, проходящая через $p_1, p_2, \ldots, p_{n-1}$ и некоторую точку $p_{i(m)} \in A$ $(n \le i(m) \le N)$, такая, что $S(h'(m)) = m$ и $x \in B(h'(m))$.*

**Доказательство.** Пусть, как и в лемме 2, $l$ — $(n-2)$-плоскость, проходящая через $p_1, p_2, \ldots, p_{n-1}$; $h(\phi_t, l)$ $(0 \le t \le N-n)$ — гиперплоскости, проходящие через соответствующие точки $p_{n+t} \in A \setminus h$ и $\phi_0 < \phi_1 < \ldots < \phi_{N-n}$ при таком направлении вращения $h$ вокруг $l$, что $x \in B(h(\phi_0, l))$. Возможность выбора такого направления вращения доказывается так же, как в лемме 2.

Так как, по построению, либо $A_B(h(\phi_0, l)) = A_B(h) \setminus p_n$, либо $A_B(h(\phi_0, l)) = A_B(h)$, то $S(h(\phi_0, l)) \le S(h)$. Если $S(h(\phi_0, l)) = S(h) = m$, то $h'(m) = h(\phi_0, l)$. В противном случае одно из неравенств $S(h(\phi_0, l)) \le S(h) \le m$ является строгим, следовательно, $S(h(\phi_0, l)) < m \le \lfloor (N-n)/2 \rfloor$. Так же, как в лемме 2, доказывается, что если $0 \le t \le N-n-1$, $S(h(\phi_t, l)) < \lfloor (N-n)/2 \rfloor$ и $x \in B(h(\phi_t, l))$, то $|S(h(\phi_{t+1}, l)) - S(h(\phi_t, l))| \le 1$ и $x \in B(h(\phi_{t+1}, l))$, то есть утверждение леммы верно и при всех $m$, таких, что $S(h(\phi_0, l)) < m \le \max\{S(h(\phi_t, l)) : 0 \le t \le N-n\} \le \lfloor (N-n)/2 \rfloor$.

Осталось показать, что $\max S(h(\phi_t, l)) = \lfloor (N-n)/2 \rfloor$. Если $\max S(h(\phi_t, l)) < \lfloor (N-n)/2 \rfloor$, то $x \in B(h(\phi_{N-n}, l))$. Тогда $A \subseteq A_B(h) \cup A_B(h(\phi_{N-n}, l)) \cup \{p_1, p_2, \ldots, p_{n-1}\} \cup p_N$. Если $N-n$ четно, то $S(h) \le (N-n)/2$ и $S(h(\phi_{N-n}, l)) \le (N-n)/2 - 1$. Если $N-n$ нечетно, то $S(h) \le (N-n-1)/2$ и $S(h(\phi_{N-n}, l)) \le (N-n-1)/2 - 1$. $|A| \le S(h) + S(h(\phi_{N-n}, l)) + n < N$ в обоих случаях, что невозможно, так как $|A| = N$. Следовательно, $\max S(h(\phi_t, l)) = \lfloor (N-n)/2 \rfloor$. Лемма 2' доказана.

**Лемма 2''.** *Пусть $h$ — гиперплоскость, проходящая через $p_1, p_2, \ldots, p_n \in A$, $S(h) < \lfloor (N-n)/2 \rfloor$, $x \in h$, $x \notin A$ и точки $x$ и $p_n$ лежат в одной $(n-1)$-полуплоскости относительно $(n-2)$-плоскости, проходящей через $p_1, p_2, \ldots, p_{n-1}$. Тогда для любого $m$, такого, что $S(h) < m \le \lfloor (N-n)/2 \rfloor$, существует гиперплоскость $h'(m)$, проходящая через $p_1, p_2, \ldots, p_{n-1}$ и некоторую точку $p_{i(m)} \in A$ $(n+1 \le i(m) \le N)$, такая, что $S(h'(m)) = m$ и $x \in B(h'(m))$.*

**Доказательство.** Искомая гиперплоскость совпадает с гиперплоскостью, существование которой доказано в лемме 2, так как $x$ и $p_n$ лежат в одном и том же полупространстве относительно любой гиперплоскости $h(\phi_t, l)$, проходящей через $p_1, p_2, \ldots, p_{n-1}, p_{n+t} \in A$ $(1 \le t \le N-n)$. Лемма 2'' доказана.

**Лемма 2'''.** *Пусть $h$ — гиперплоскость, проходящая через $p_1, p_2, \ldots, p_n \in A$, $S(h) \le \lfloor (N-n-1)/2 \rfloor$, $x \in h$, $x \notin A$ и точки $x$ и $p_n$ лежат в разных $(n-1)$-полуплоскостях относительно $(n-2)$-плоскости, проходящей через $p_1, p_2, \ldots, p_{n-1}$. Тогда для любого $m$, такого, что $S(h) \le m \le \lfloor (N-n)/2 \rfloor$, существует гиперплоскость $h'(m)$, проходящая через $p_1, p_2, \ldots, p_{n-1}$ и некоторую точку $p_{i(m)} \in A$ $(n+1 \le i(m) \le N)$, такая, что $S(h'(m)) = m$ и $x \in B(h'(m))$.*

**Доказательство** данной леммы аналогично доказательству леммы 2'.

Гиперплоскость $h'(k-1)$ содержит $n$ точек из $A$, а замкнутое полупространство $U(h'(k-1))$ содержит $N-(k-1)$ точку из $A$. Следовательно, $U(h'(k-1))$ является одним из полупространств, описанных в условии теоремы. Таким образом, если выполнены условия одной из лемм 2, 2', 2'', 2''', и число $k-1$ принадлежит указанному в их формулировках диапазону значений $m$, то точка $p_n$ (соответственно $x$) не принадлежит пересечению описанных в условии теоремы полупространств. Дальнейшее доказательство в основном базируется на данном факте.

**Доказательство** теоремы для случая $0 < k \le \lfloor (N-n+1)/2 \rfloor$.

Обозначим $P_k$ — пересечение замкнутых полупространств, описанных в условии теоремы.

Покажем, что $P_k = M_k$.

Пусть $x \notin P_k$. Тогда по крайней мере одно из описанных в условии теоремы замкнутых полупространств не содержит $x$. Другими словами, существует гиперплоскость $h$, содержащая $n$ точек из $A$, такая, что $x \in B(h)$ и $S(h) = k-1$. Значит, гиперплоскость $h'$, параллельная $h$ и проходящая через $x$, отсекает от $A$ не более $k-1$ точки. Следовательно, $x \notin M_k$. Мы показали, что $M_k \subseteq P_k$.

Пусть $x \notin M_k$. Тогда существует гиперплоскость $h$, проходящая через $x$ и отсекающая от $A$ менее $k$ точек. Так как $k \le \lfloor (N-n+1)/2 \rfloor$, то $S(h) \le k-1 \le \lfloor (N-n+1)/2 \rfloor - 1 \le \lfloor (N-n)/2 \rfloor$, и, следовательно, по лемме 1 существует гиперплоскость $h'$, проходящая через $x$ и некоторые точки $p_1, p_2, \ldots, p_{n-1} \in A$, такие, что $x, p_1, p_2, \ldots, p_{n-1}$ не лежат в одной $(n-2)$-плоскости, и $A_B(h') \subseteq A_B(h)$. $S(h') \le S(h)$.

Если в гиперплоскости $h$ лежит хотя бы одна точка из $A$, то $S(h) < k-1$, и, следовательно, $S(h') \le S(h) < k-1 \le \lfloor (N-n+1)/2 \rfloor - 1 \le \lfloor (N-n)/2 \rfloor$. Таким образом, выполнены условия одной из лемм 2, $2'$. $2''$ или $2'''$. Следовательно, $x \notin P_k$.

Пусть $h$ не содержит точек из $A$. Тогда возможны два случая.

1. Гиперплоскость $h'$ содержит ровно $n-1$ точку из $A$. Тогда, так как $S(h') \le k-1 \le \lfloor (N-n)/2 \rfloor$, то $x \notin P_k$ согласно лемме $2'$.

2. Гиперплоскость $h'$ содержит $n$ точек $p_1, p_2, \ldots, p_n$ из $A$.

Если $x \notin C(p_1, p_2, \ldots, p_n)$, то найдется по крайней мере одна точка $p_j$ $(1 \le j \le n)$, лежащая с точкой $x$ в разных $(n-1)$-полуплоскостях относительно $(n-2)$-плоскости, проходящей через $p_1, p_2, \ldots, p_{j-1}, p_{j+1}, \ldots, p_n$. Так как $S(h') \le k-1 \le \lfloor (N-n+1)/2 \rfloor - 1 = \lfloor (N-n-1)/2 \rfloor$ и $k-1 \le \lfloor (N-n)/2 \rfloor$, то выполняются условия леммы $2'''$, и, следовательно, $x \notin P_k$.

Пусть $x \in C(p_1, p_2, \ldots, p_n)$. Гиперплоскости $h$ и $h'$ пересекаются по некоторой $(n-2)$-плоскости $l$. Так как $x \in C(p_1, p_2, \ldots, p_n)$, $x \in l$ и $p_1 \notin l, p_2 \notin l, \ldots, p_n \notin l$ (так как $h$ не содержит точек из $A$), то по крайней мере одна из точек $p_1, p_2, \ldots, p_n$ лежит в $B(h)$. Следовательно, $S(h') < S(h) \le k-1 \le \lfloor (N-n)/2 \rfloor$, и $x \notin P_k$ согласно лемме $2''$. Итак, $M_k \supseteq P_k$.

Из $M_k \subseteq P_k$ и $M_k \supseteq P_k$ следует, что $P_k = M_k$. Теорема доказана.

Таким образом, мы получили описание множеств $M_k$ для всех $k$, кроме $k = (N-n)/2+1$ при четном $N-n$. В последнем случае $M_k$, по-видимому, также совпадает с пересечением описанных в теореме замкнутых полупространств, однако строгого доказательства данного факта пока не найдено.

**Утверждение 2.** *Если $0 < k \le \lfloor (N-n+1)/2 \rfloor$ и $M_k$ не пусто, то $M_k$ является замкнутым ограниченным выпуклым многогранником в пространстве $R^m$ $(1 \le m \le n)$.*

**Доказательство.** Предположим, что множество $M_k$ не ограничено. Тогда существует точка $x \in M_k$, такая, что $x \notin C(A)$. Так как $x \notin C(A)$, то существует гиперплоскость $h$, проходящая через $x$ и не пересекающая $C(A)$. $S(h) = 0$, следовательно, по лемме 1, существует гиперплоскость $h'$, проходящая через не лежащие в одной $(n-2)$-плоскости точки $x$ и $p_1, p_2, \ldots, p_{n-1} \in A$, такая, что $S(h') = 0$.

Если $h'$ содержит ровно $n-1$ точку из $A$, то $x \notin M_k$ согласно теореме и лемме $2'$. Если же $h'$ содержит $n$ точек из $A$ (обозначим их $p_1, p_2, \ldots, p_n$), то $x \notin C(\{p_1, p_2, \ldots, p_n\})$, так как $x \notin C(A)$ и $C(\{p_1, p_2, \ldots, p_n\}) \subset C(A)$. Следовательно, существует по крайней мере одна точка $p_j$ $(1 \le j \le n)$, лежащая с точкой $x$ в разных $(n-1)$-полуплоскостях относительно $(n-2)$-плоскости, проходящей через $p_1, p_2, \ldots, p_{j-1}, p_{j+1}, \ldots, p_n$. Тогда $x \notin M_k$ согласно теореме и лемме $2'''$. Получили противоречие. Следовательно, множество $M_k$ ограничено.

Так как $M_k$ является пересечением конечного числа замкнутых полупространств и $M_k$ ограничено, то либо $M_k$ — замкнутый выпуклый ограниченный многогранник в $R^m$ $(1 \le m \le n)$, либо $M_k = \emptyset$. Утверждение 2 доказано.

Отметим, что не аффинно-инвариантные конструкции использованы лишь при доказательстве утверждений и не участвуют в их формулировках.

Приведенная в работе теорема дает переборный алгоритм для построения множеств точек заданной глубины для произвольной выборки в пространстве $R^n$. Программа, работающая по указанному алгоритму, позволяет построить все не пустые $M_k$ (задавая их множеством граней) на плоскости для выборки из 50 точек за 0,12 секунды, а для выборки из 100 точек за 1,3 секунды (на компьютере Pentium-III 550MHz, RAM-128MB). В трехмерном пространстве тот же эксперимент (на том же компьютере) требует 8,2 секунды для выборки из 30 точек, 21 секунду для выборки из 40 точек, 57 секунд для выборки из 50 точек. С ростом размерности пространства временные затраты растут экспоненциально. Впрочем, трудоемкость известных алгоритмов построения выпуклых оболочек, необходимых для выполнения процедуры "лущения", также зависит экспоненциально от размерности пространства.

Вопрос о более экономных алгоритмах решения задачи построения $M_k$ представляет несомненный интерес, но выходит за рамки данной статьи.

## Литература

[1] Huber P. J. Robust statistics: A review //Ann. Math. Stat. 1972. 43(3). P.1041-1067.

[2] Хьюбер П. Робастность в статистике. М.: Мир, 1984.

[3] Препарата Ф., Шеймос М. Вычислительная геометрия: Введение. М. Мир, 1989.

*Модел. и анализ информ. систем.* Т.8, №1 (2001) 9–11

УДК 517.9

# О возможности сведения вопроса о существовании и устойчивости стационарных режимов одной нелинейной краевой задачи к исследованию двумерного отображения

Куликов А.Н.

*Ярославский государственный университет*
*150 000, Ярославль, Советская, 14*

В данной работе рассматривается возможность сведения нелинейной краевой задачи для системы двух дифференциальных уравнений гиперболического типа к исследованию нелинейного двумерного отображения. При этом использован классический метод, часто называемый методом отражений. Для построенного двумерного отображения и его второй итерации исследован вопрос о существовании и устойчивости неподвижных точек.

**1. Постановка задачи.**

На отрезке [0;1] рассмотрим краевую задачу

$$u_t = v_x - a_1 u, \quad v_t = u_x - a_2 v, \tag{1}$$

$$v(t,0) = 0, \quad \alpha v(t,1) + u(t,1) + (1+\beta)u(t,0) - u^3(t,0) = 0. \tag{2}$$

Здесь $u = u(t,x)$, $v = v(t,x)$ – переменные составляющие напряжения и силы тока в распределенной RCLG – линии; $a_1 = G\sqrt{\frac{L}{C}}$, $a_2 = R\sqrt{\frac{C}{L}}$, где R, C, L, G – распределенные сопротивление, емкость, индуктивность, проводимость, параметры $\alpha \in [0;1]$, $\beta \geq 0$. Система уравнений (1) хорошо известна как система телеграфных уравнений. В данном случае она рассматривается вместе с нелинейными краевыми условиями и моделирует RCLG – автогенератор. Более подробное обсуждение физических аспектов постановки задачи можно найти в работах [1 – 3]. В работах [2 – 3] рассматривается случай, когда R и G малы, т. е. малы $a_1$, $a_2$, а также мал коэффициент $\alpha$, фигурирующий в краевом условии (2). Более того в этих работах его полагают настолько пренебрежимо малым, что считается целесообразным положить его равным 0. С другой стороны, в работах [3 – 4] отмечалось, что свойства решений краевой задачи при $\alpha = 0$ и при $\alpha > 0$ значительно различаются.

В данной работе рассмотрим иную предельную ситуацию: $a_1$, $a_2$, пренебрежимо малы по сравнению с $\alpha$ и $\beta$ и поэтому для первоначального изучения динамики краевой задачи положим $a_1 = a_2 = 0$.

**2. Сведение краевой задачи к исследованию нелинейного отображения.**

В работе [5] показано, что задача Коши для краевой задачи (1), (2) корректно разрешима, если

$$u(0,x) = f(x), \quad v(o,x) = g(x), \tag{3}$$

где $f(x)$, $g(x) \in C^1[0;1]$ $(W_2^1(0;1))$ и эти функции удовлетворяют краевым условиям (2).

Итак рассмотрим систему

$$u_t = v_x, \quad v_t = u_x \tag{4}$$

вместе с краевыми условиями (2) и начальными условиями (3). Проинтегрируем задачу (4), (2), (3), следуя методу, предложенному в работе [5]. Для этого предположим, что функции $f(x)$, $g(x)$ продолжены каким – либо образом с сохранением гладкости на всю числовую ось. Тогда решение системы (4), удовлетворяющее начальным условиям (5), может быть найдено по известной формуле Даламбера:

$$u(t,x) = \frac{f(x+t) + f(x-t)}{2} + \frac{g(x+t) - g(x-t)}{2},$$

$$v(t,x) = \frac{f(x+t) - f(x-t)}{2} + \frac{g(x+t) + g(x-t)}{2}.$$

Подставляя полученные $u(t,x)$, $v(t,x)$ в краевые условия (2), получаем тот способ продолжения $f(x)$ и $g(x)$, при котором функции $u(t,x)$, $v(t,x)$ будут удовлетворять и краевым условиям (2). Принципиальным моментом следует, очевидно, считать первый шаг при предложенном способе продолжения: продолжение $f(x)$ и $g(x)$ на отрезки [-1;0] и [1;2]. Анализируя первое из краевых условий (2), легко выводится тот факт, что $f(x)$ четная, а $g(x)$ нечетная функции. Осталось доопределить $f(x)$ и $g(x)$ на [1,2].

Пусть $s \in [0;1]$. Подставляя $u(t,x)$ во второе из краевых условий (2) при $t = s$, а затем при $t = -s$, после элементарных алгебраических преобразований получаем

$$f(1+s) = -\frac{(1+\alpha^2)}{(1-\alpha^2)}f(1-s) - \frac{2\alpha}{(1-\alpha^2)}g(1-s) - 2\frac{(1+\beta)}{(1-\alpha^2)}(f(s)-\alpha g(s)) + \\ + \frac{1}{(1+\alpha)}(f(s)+g(s))^3 + \frac{1}{(1-\alpha)}(f(s)-g(s))^3. \tag{5}$$

$$g(1+s) = \frac{2\alpha}{(1-\alpha^2)}f(1-s) + \frac{(1+\alpha^2)}{(1-\alpha^2)}g(1-s) - 2\frac{(1+\beta)}{(1-\alpha^2)}(g(s)-\alpha f(s)) + \\ + \frac{1}{(1+\alpha)}(f(s)+g(s))^3 - \frac{1}{(1-\alpha)}(f(s)-g(s))^3. \tag{6}$$

Положим теперь $f_1(x) = u(1,x)$, $g_1(x) = v(1,x)$. Из формулы Даламбера вытекает, что

$$f_1(x) = \frac{f(x+1)+f(x-1)}{2} + \frac{g(x+1)-g(x-1)}{2},$$

$$g_1(x) = \frac{f(x+1)-f(x-1)}{2} + \frac{g(x+1)+g(x-1)}{2}.$$

Используя четность $f(x)$, нечетность $g(x)$ и формулы (5), (6), окончательно получаем, что при $t = 1$ решение смешанной задачи (4), (2), (3) порождает отображение

$$f_1(x) = \frac{1}{1+\alpha}\{-(1+\beta)(f(x)+g(x)) + \alpha\cdot f(1-x) + g(1-x) + (f(x)+g(x))^3\}, \tag{7}$$

$$g_1(x) = \frac{1}{1+\alpha}\{-(1+\beta)(f(x)+g(x)) - f(1-x) - \alpha\cdot g(1-x) + (f(x)+g(x))^3\}, \tag{8}$$

из $C_m(W_0)$ в $C_m(W_m)$. Здесь нелинейное многообразие $C_m \subset C^1[0;1]\times C^1[0;1]$ ($W_m \subset W_2^1[0;1]\times W_2^1[0;1]$) и состоит из тех пар функций $f(x)$, $g(x)$, которые удовлетворяют краевым условиям (2).

Изучение отображения (7), (8) представляет безусловно самостоятельный интерес, но в дальнейшем представляется целесообразным задачу упростить, опираясь при этом на соображения физического характера. Как отмечалось, например, в монографии [3], колебания, реализуемые в генераторе, обычно снимаются с выхода усилителя, т.е. при $x = 1$ в рассматриваемой здесь краевой задаче. Поэтому, по крайней мере, для первоначального изучения представляет интерес изучение отображения (7), (8) при $x = 1$, а также $x = 0$. Последнее означает, что мы рассматриваем при исследовании только значения $u(t,x)$ и $v(t,x)$ только на входе и выходе линии, что в значительной мере оправдано и с физической точки зрения.

Положим $z_1 = f(0)$, $z_2 = g(1)$, $w_1 = f_1(0)$, $w_2 = g_1(1)$ и отметим при этом, что в силу краевых условий (2) $g(0) = 0$, $g_1(0) = 0$, а также

$$f(1) = -\alpha\cdot z_2 - (1+\beta)z_1 + z_1^3,$$

$$f_1(1) = -\alpha\cdot w_2 - (1+\beta)w_1 + w_1^3.$$

Из (7), (8) выводим, что краевая задача (4), (2), (3) порождает двумерное отображение

$$w_1 = -(1+\beta)z_1 + (1-\alpha)z_2 + z_1^3. \tag{9}$$

$$w_2 = \frac{1}{1+\alpha}\{(2\beta+\beta^2)z_1 - (1+\beta)(1-\alpha)z_2 - (1+\beta)z_1^3 + (-(1+\beta)z_1 + (1-\alpha)z_2 + z_1^3)^3\}. \tag{10}$$

Отображение (9), (10) определено при любых значениях переменных $z_1$, $z_2$ и любых значениях параметра $\beta \geq 0$ и при $\alpha \in [0;1)$. Это вытекает из предложенного метода интегрирования краевой задачи, в частности из вида формул (5), (6).

Замечание. Можно рассмотреть случай, когда $\beta \in (-1;0)$, но этот случай мало интересен с физической точки зрения (см. [1 - 3]).

**3. Двумерное отображение.**

Равенства (9), (10) порождают двумерное отображение $w = T(\alpha,\beta) \circ z$, $z = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$, $w = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$.

Очевидно, что каждому периодическому решению краевой задачи (4), (2) с периодом $T = n$, $n \in N$ соответствуют неподвижные точки отображения $T(\alpha,\beta)$ или его итераций с наследованием свойств устойчивости. Обратно, разумеется, не обязательно верно, но тем не менее изучение свойств отображения $T(\alpha,\beta)$ во многом проясняет динамику краевой задачи (4), (2).

**Теорема 1.** *Двумерное отображение $T^2(\alpha,\beta)$ имеет неподвижные точки $z_a = \begin{pmatrix} \pm\sqrt{\beta} \\ 0 \end{pmatrix}$. Эти две неподвижные точки асимптотически устойчивы при $\alpha \in (0;1)$ и $\beta \in [0;1)$ и неустойчивы при $\beta > 1$. Само отображение $T(\alpha,\beta)$ имеет две неподвижные точки $z_b = \begin{pmatrix} \pm\sqrt{2+\beta} \\ 0 \end{pmatrix}$, которые неустойчивы при всех рассматриваемых значениях параметров $\alpha$ и $\beta$.*

Доказательство теоремы достаточно элементарно. Для доказательства того факта, что $z_a$ является неподвижной точкой $T^2(\alpha,\beta)$, достаточно показать, что $T(\alpha,\beta) \circ z_a = -z_a$. Последнее равенство и также то, что $T(\alpha,\beta) \circ z_b = z_b$ проверяется непосредственной подстановкой в (9), (10).

Для проверки условий устойчивости можно использовать теорему Ляпунова об устойчивости по первому приближению для отображений: отображение $T(\alpha,\beta)$ линеаризуются на $z_a(z_b)$ и для полученной матрицы Якоби исследуются ее собственные значения (мультипликаторы отображения (9), (10)). При этом получаем характеристическое уравнение:

$$\lambda^2 + p\lambda + q = 0,$$

где $p = 2\dfrac{(1-2\beta)}{(1+\alpha)}$; $q = \dfrac{(1-\alpha)}{(1+\alpha)}$ в первом случае и $p = -2\dfrac{(2\beta+5)}{(1+\alpha)}$, $q = \dfrac{(1-\alpha)}{(1+\alpha)}$ во втором. После чего проверка устойчивости неподвижных точек сводится к выявлению условий, при которых выполнены неравенства $|\lambda_k| < 1$ $(k = 1,2)$.

## Литература


[1] Колесов Ю. С. Математическая теория RC – генераторов с распределенными параметрами в цепи обратной связи // Дифференциальные уравнения и их применения. Вильнюс: Ин-т. физики и математики АН Лит. ССР, 1971. Вып. 2. 68 с.

[2] Камбулов В. Ф., Колесов А. Ю. О явлении буферности в одной резонансной гиперболической краевой задаче из радиофизики // Мат. сб. 1995. Т.186, №7. С. 77-96.

[3] Колесов А. Ю., Мищенко Е. Ф., Розов Н. К. Асимптотические методы исследования периодических решений нелинейных гиперболических уравнений. М.: Наука, 1998. 191 с.

[4] Колесов Ю. С. Куликов А. Н. Бифуркация автоколебаний в классической системе телеграфных уравнений при одном неклассическом нелинейном граничном условии // Мат. заметки, 1999. Т.66, №6. С. 948 - 949.

[5] Куликов А. Н. Исследование одной краевой задачи, возникающей в радиофизике // Исследования по устойчивости и теории колебаний. Ярославль: ЯрГУ, 1976. С. 67 - 85.

*Модел. и анализ информ. систем.* Т.8, №1 (2001) 12–18

УДК 519.673

# Нейросетевая модель для планирования путей роботов в динамически изменяющейся среде

Лебедев Д.В.
*Ярославский государственный университет*
*150 000, Ярославль, Советская, 14*

Представлена новая нейросетевая модель для планирования путей точечного (по размеру) объекта в динамически изменяющейся среде. Для процесса планирования *не требуется* предварительной информации о состоянии среды. Робот перемещается по регулярной решетке (рабочему полю), в узлах которой могут находиться стационарные препятствия. Рассматривается случай, когда робот и препятствия движутся с одинаковой, постоянной скоростью. Планирование путей осуществляется с помощью нейронной сети, реализующей волновую динамику. Каждый нейрон сети имеет лишь локальные связи. На каждом такте сеть генерирует числовое поле потенциалов, несущее новую информацию о расположении препятствий, траектории которых неизвестны объекту. Такое изменение поля потенциалов может интерпретироваться как *обновление сенсорных данных робота либо данных камеры*, наблюдающей за рабочим полем. Эффективность предложенного подхода подтверждена компьютерной симуляцией.

## 1. Введение

Способность самостоятельно планировать маршрут является критической для автономных объектов. Часто таким объектам приходится функционировать в среде (напр., аэропорты, вокзалы, строительные участки), состояние которой изменяется с течением времени. Такие изменения связаны, как правило, с незапланированным появлением в среде посторонних объектов.

Задачам планирования маршрутов в нестационарных средах посвящено большое количество работ. Фундаментальной и покрывающей различные аспекты, связанные с планированием движения в нестационарных средах, является работа [1]. Как показывает анализ ряда публикаций, планирование движения в динамически меняющихся средах на порядок сложнее, чем в стационарных (см. для сравн. [2]-[4]). В литературе описан ряд методик для решения задачи о нахождении маршрута в динамической среде. Так, если траектории подвижных препятствий заданы изначально, известным является подход к планированию пути в пространстве *конфигурация-время* (напр., [5], [6]). В этом пространстве подвижные препятствия являются стационарными, поэтому при планировании маршрута могут быть использованы стандартные методики планирования путей в стационарной среде [7]. В этом случае необходимо, однако, наблюдать за монотонностью пути во времени. Автором [8] введена концепция *транзитных препятствий*, которые существуют в среде лишь определённое время, т.е. могут появляться и исчезать.

В случае, когда траектории движения препятствий неизвестны и существует ограничение на скорость движения объекта, обычно применяют методику *разложения на путь/скорость*. Суть ее состоит в планировании маршрута в два этапа: а) планировании пути, обходящего стационарные препятствия; и б) коррекции профиля скорости, соответствующего спланированному пути, для обхода подвижных препятствий (см., напр., [9]). В работе [10] приведена попытка статистически выделить *паттерны движения* препятствий из данных, получаемых от камеры, наблюдающей за рабочим полем. Траектории движения затем определяются в терминах вероятности столкновения с препятствием, а также согласно ожидаемому времени достижения цели.

Существует, ряд нейросетевых подходов к решению задачи планирования путей. Все из нижеприведенных, однако, работают только в неизменяющейся среде. Кроме того, оптимальность пути часто остается вне рассмотрения. В работе [11] авторы используют самоорганизующуюся сеть Кохонена, содержащую узлы двух типов. Работа [12] содержит описание нейронной сети с осциллирующим поведением, находящей решение задачи динамического программирования и осуществляющей тем самым планирование пути для объекта с двумя степенями свободы. Алгоритм, предложенный в [13], использует для планирования маршрута набор промежуточных точек, связанных эластичными струнами. Градиентные силы потенциального поля, сгенерированного многослойной нейронной сетью, минимизируют длину эластичных струн, заставляя их одновременно с этим огибать области препятствий. Нейронная сеть, описание

Уравнения состояния нейрона ((1)-(3)) содержат правила, обеспечивающие изменение уровня активности $i$-го нейрона и ассоциированных с ним весов связей. Так, нейроны, соответствующие стационарным препятствиям, получают единичное значение на внешнем тормозящем входе $\alpha_i$ и не изменяют значение уровня активности (3). Из той же формулы, активность нейрона, соответствующего целевой позиции и получающего единичное значение на внешнем возбуждающем входе $\beta_i$, инкрементируется на каждом такте в процессе эволюции сети. Эволюция $i$-го нейрона поля сети в момент времени $t$ зависит от наличия в окрестности этого нейрона активного соседа, изменившего уровень активности к рассматриваемому моменту времени. Если $i$-ый нейрон к этому моменту времени уже активен, функция $D$ гарантирует изменение веса лишь от нейрона-соседа с активностью не большей, чем у $i$-го нейрона. Функция $P_k(.)$ определяет приоритет среди соседей из окрестности $i$-го нейрона и выбирает единственного соседа с указанными характеристиками. Приоритет каждому из соседей $i$-го нейрона может быть присвоен, например, с помощью возрастающей числовой метки, согласно которой и будет осуществляться обход соседей (см. рис. 1Б)).

Эволюция нейронов сети (не соответствующих препятствиям и целевой позиции) в каждый дискретный момент времени жестко детерминирована и обусловлена наличием в их окрестности нейрона с отмеченными выше свойствами.

## 3.2. Планирование путей и движение робота

В начальный момент времени значения активностей всех нейронов сети и весовых коэффициентов нулевые. Активность сети инициируется на целевом нейроне (нейроне, соответствующем целевой позиции в $C$), уровень активности которого становится равным единице. Возбуждение целевого нейрона приводит к порождению волнового фронта в поле сети. В момент времени 2 активность целевого нейрона вновь инкрементируется, и его $n = 2d$ неактивных нетормозящих (не соответствующих препятствиям) соседей изменяют значения уровней активности и весовых коэффициентов согласно уравнениям (1)-(3). Процесс продолжается по мере того, как волны нейронной активности распространяются и достигают каждый нетормозящий нейрон в поле сети. Увеличение активности целевого нейрона, таким образом, порождает на каждом такте работы сети новый волновой фронт. Положительное значение (равное единице) на тормозящем внешнем входе $\gamma_j$ нейрона $j$ в момент времени $t_k$ означает присутствие в ассоциированной с нейроном $j$ позиции подвижного препятствия. Согласно (4), вес связи от этого нейрона (вес $w_{ji}$) примет нулевое значение. После того, как препятствие покинет рассматриваемую позицию, значение веса будет вновь изменяться в соответствии с (4).

В этот момент активность нейрона становится нулевой. После того, как препятствие переместится из рассматриваемой позиции, нейрон восстановит уровень активности в соответствии с вновь пришедшим волновым фронтом.

В каждый момент времени активность всех нейронов сети формирует числовое поле потенциалов над дискретным представлением $C$. Глобальный минимум сгенерированного поля соответствует нейрону, ассоциированному с целевой позицией. Поле потенциалов представляет собой множество достижимости целевой позиции.

Динамика сети, таким образом, реализует процедуру сложения волновых фронтов. Так, согласно уравнениям (1), в момент времени 2 уровень активности целевого нейрона станет равным двум, а уровни активностей его нетормозящих соседей примут значение, равное трем. На каждом такте вновь порожденный волновой фронт несет обновленную информацию о расположении препятствий в поле роботов. Функция $P_k(.)$ отвечает за обновление веса лишь от единственного нейрона-соседа, активность которого изменилась вследствие прихода новой волны.

В момент, когда волновой фронт достигает нейрона, соответствующего начальной позиции, робот начинает движение. На каждом последующем такте робот перемещается в направлении положительного антиградиента поля потенциалов в позицию, ассоциированную с нейроном, вес от которого к текущей позиции имеет единичное значение. Перемещения робота определяют цепочку нейронов, соответствующую пути от начальной до целевой позиции. Так, если $\tau(t_s) = S$ - стартовая позиция робота, а $t_s$ - время, когда волновой фронт достигает начальную позицию, следующими позициями робота будут $\tau(t_s + n) = \{p_j : x_j > 0, w_{ji} > 0, j \in \nu_i, t_s + n \leq t_g\}$, $n = \overline{1..t_g - t_s - 1}$, где $p_j$ - позиция в $C$, ассоциированная с нейроном $j$, $t_g$ - время достижения целевой позиции.

Путь робота, сформированный таким образом, будет безопасным, т.е. обходящим стационарные и динамические препятствия в рабочем поле.

## 3.3. Симуляция на ЭВМ

Предложенная модель была реализована в объектно-ориентированной среде условного программирования. Для удобства и максимальной гибкости были описаны классы *PointRobot* и *Obstacle*. При наличии этих абстракций код легко может модифицироваться без нарушения общей логики программы.

### 3.3.1. Планирование пути в динамической среде

На рис. 2 представлена модельная ситуация планирования пути робота в среде, содержащей стационарные и динамические препятствия.

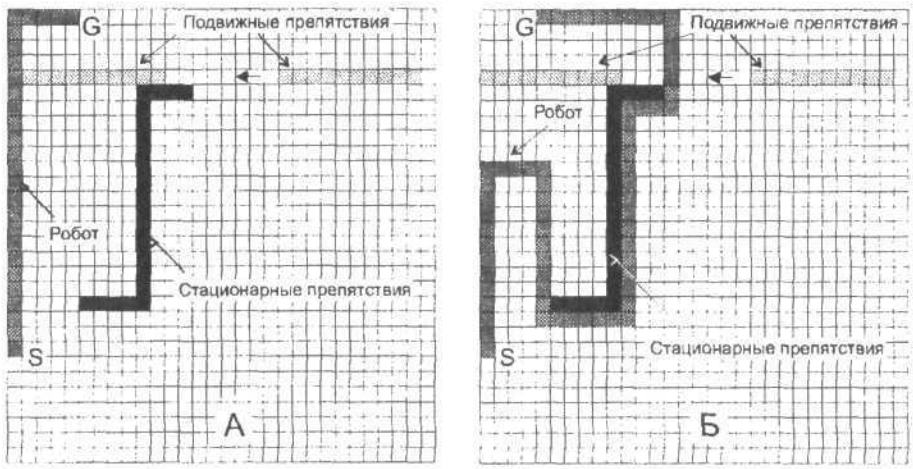


Рис. 2. Планирование пути в динамической среде.

Начальная и целевая позиции для робота обозначены на рисунке как $S$ и $G$. Нейронная сеть состоит из $30 \times 30$ нейронов. Подвижные препятствия появляются в среде после 20-го такта работы сети и начинают движение в направлении, указанном стрелкой. Робот начинает перемещаться в момент, когда первый волновой фронт достигает его начальной позиции.

На рис. 2А) изображена ситуация, когда подвижные препятствия, остановившись, не оказывают влияния на путь робота. Робот достигает цели, следуя вдоль пути, который эквивалентен маршруту в стационарной среде с аналогичным (как на рис. 2А)) набором препятствий.

В ситуации, показанной на рис. 2Б), процесс планирования стартовал с теми же начальными условиями, как в случае 2А). Подвижные препятствия, однако, перемещаясь справа налево, закрыли выход из туннеля. Робот, отреагировав на динамические изменения в среде, выбрал другой маршрут для достижения цели.

### 3.3.2. Преследование цели

Примеры, иллюстрирующие моделирование задачи о преследовании цели, показаны на рис. 3. Здесь положение целевой позиции изменяется с течением времени. Задача робота - поймать цель до момента, когда она выйдет за пределы поля. Робот и цель начинают двигаться синхронно после того, как первый волновой фронт достигает позиции робота. Поскольку робот и цель перемещаются с одинаковой скоростью, то для захвата роботу необходимо выбирать более короткий путь, чем цель.

На рис. 3А) продемонстрирован случай преследования в среде без препятствий. Цель петляет по рабочему полю и становится настигнутой в отмеченной позиции.

Ситуация, проиллюстрированная на рис. 3Б), имеет аналогичные начальные условия, но здесь рабочее поле содержит запрещенные позиции. Робот следует за целью, обходя препятствия, и настигает ее в указанной точке.

# 4. Заключение

В работе представлена нейросетевая архитектура для планирования путей роботов в средах, изменяющих свое состояние с течением времени. На каждом такте изменение уровня активности целевого ней-

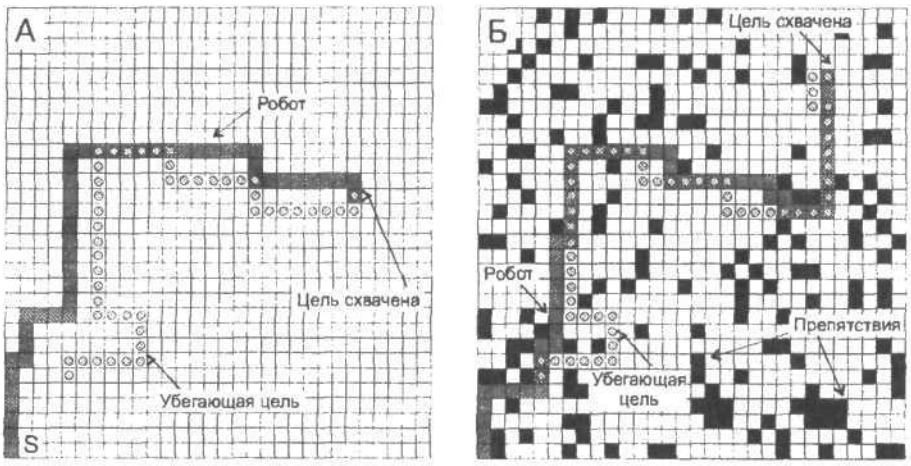


Рис. 3. Преследование цели.

рона порождает распространение нового волнового фронта. Нейронная активность распространяется по нейронам сети, минуя нейроны, соответствующие препятствиям. Активности всех нейронов сети формируют числовое поле потенциалов над дискретным представлением среды робота. Глобальный минимум поля достигается на целевом нейроне. Робот начинает движение в момент, когда волновой фронт достигает нейрона, ассоциированного с начальной позицией. Следующим в цепочке, формирующей путь, будет активный нейрон, изменивший значение уровня активности вследствие прихода новой волны активности, вес от которого к нейрону, связанному с текущей позицией, равен единице.

Необходимо отметить следующие положительные моменты представленной модели:

- модель эффективна с точки зрения вычислений и легко реализуется посредством условного программирования; возможно планирование в реальном времени (в смысле скорости распространения нейронной активности и обновления информации о среде);
- сложность вычисления линейно зависит от числа нейронов сети;
- планирование реализуется эффективно в средах, где могут происходить непредвиденные изменения;
- итоговый путь является глобально-оптимальным в смысле длины;

  **Утверждение.** Длина пути оптимальна в метрике $L_1$ (манхэттенская метрика, определенная в $\Re^2$ как $\rho(x_1, y_1, x_2, y_2) = |x_1 - x_2| + |y_1 - y_2|$).

  **Доказательство.** Действительно, путь на каждом шаге складывается из последовательности смежных точек в $C$, ассоциированных с набором активных нейронов сети. Количество нейронов в этом наборе всегда минимально, поскольку пути с большей длиной отсекаются функцией приоритета (4). Расстояние между нейронами-соседями в этом наборе есть расстояние в метрике $L_1$ между точками в $C$, соответствующими нейронам цепочки. Путь, таким образом, есть сумма таких расстояний между нейронами и, следовательно, является оптимальным в метрике $L_1$. Робот, отсюда, на каждом такте переходит в позицию, принадлежащую кратчайшему пути, который соответствует текущему такту и текущим изменениям в среде.

- сеть является локально связанной; не требуется предварительного обучения сети.

Волновая сеть, таким образом, представляет достаточно мощный механизм для решения различных задач, связанных с планированием путей как в стационарных, так и в динамических средах. Поскольку сеть оперирует напрямую дискретной информацией о среде, она может интегрироваться с различными системами визуального наблюдения или контроля, например, с сенсорными системами роботов или с камерой, фиксирующей изменения в рабочей среде.

## Литература

[1] Fujimura Kikuo. Motion Planning in Dynamic Environments. 1992. Springer-Verlag.

[2] Canny J.F. The Complexity of Robot Motion Planning. 1988. MIT Press.

[3] Schwartz J.T., Sharir M. Algorithmic Motion Planning in Robotics. In: Handbook of theoretical computer science (ed. Jan van Leeuwen). 1990. Elsevier.

[4] Reif J, Sharir M. Motion Planning in the Presence of Moving Obstacles // Journal of the Association for Computing Machinery. 1994. Vol. 41(4). P.764-790.

[5] Fujimura Kikuo. Motion Planning using Transient Pixel Representations // Proc. of the Int. Conf. on Robotics and Automation. 1993. P.34-39.

[6] Baginski B. The $Z^3$-Method for Fast Path Planning in Dynamic Environments // Proc. of IASTED Conference Applications of Control and Robotics. 1996. P.47-52.

[7] Latombe J.-C. Motion Planning in the Presence of Moving Obstacles. In: The Robotics Review 2 (eds. Oussama Khatib, John J. Craig, Tomas Lozano-Perez. 1992. MIT Press.

[8] Fujimura Kikuo. Motion Planning Amid Transient Obstacles // The Int. Journal of Robotics Research. 1994. Vol. 13(5). P.395-407.

[9] Fraichard T., Demazeau Y. Motion Planning in a Multi-Agent World. In: Decentralized A.I. (eds. Yves Demazeau, Jean-Pierre Mueller). 1990. Elsevier.

[10] Kruse E., Wahl F.M. Camera-based Observation of Obstacle Motions to Derive Statistical Data for Mobile Robot Motion Planning // Proc. of the Int. Conf. on Robotics and Automation. 1998. P.662-667.

[11] Vleugels J., Kok J.N., Overmars M. Motion Planning with Complete Knowledge Using A Colored SOM // Int. Journal of Neural Systems. 1997. Vol. 8(5-6). P.613-628.

[12] Lemmon M. 2-Degree-of-freedom Robot Path Planning using Cooperative Neural Fields // Neural Computation. 1991. Vol. 3. P.350-362.

[13] Lee S., Kardaras G. Collision-free Path Planning with Neural Networks // Proc. of the Int. Conf. on Robotics and Automation. 1997. P.3565-3570.

[14] Kindermann T, Cruse H, Dautenhahn K. A fast, three-layer neural network for path finding // Network: Computation in Neural Systems. 1996. Vol.7. P.423-436.

[15] Агеев Д.А, Истратов А.Ю. Нейросетевая реализация задачи поиска оптимального пути // Известия Академии наук. Теория и системы управления. 1998ю Вып.1. С.124-132.

[16] Yang S.X., Meng M. An efficient neural network approach to dynamic robot motion planning // Neural Networks. 2000. Vol.13. P. 143-148.

[17] Kassim A.A, Kumar V. Path Planning for Autonomous Robots Using Neural Networks // Journal of Intelligent Systems. 1997. Vol.7(1-2). P. 33-56.

[18] Лебедев Д.В. Применение нейронной сети волнового распространения для планирования безопасных путей плоских роботов // Информационные технологии. 2000. №10. С.19-26.

[19] Лебедев Д.В. Планирование оптимальных путей плоских роботов на основе использования волновой нейросетевой архитектуры // Моделирование и анализ информационных систем. 2000. Том 7. №1. С.30-38.

*Модел. и анализ информ. систем.* Т.**8**, №**1** (2001) 19–24

УДК 519.872

# Приоритетное множество для системы массового обслуживания $M_2|G_2|1$

Черменский П.П.
*Ярославский государственный университет*
*150 000, Ярославль, Советская, 14*

## 1. Введение

В работе рассматривается система массового обслуживания, состоящая из одного прибора и бесконечной очереди, на вход которой поступают требования двух типов, разрешается прерывание обслуживания. Для этой системы найдено параметрическое описание приоритетного множества - множества всех значений средних длин очередей в системе. Задача решалась путём сведения исходной системы к системе с ветвящимися потоками вторичных требований. Решение основано на результатах работ [1]-[4].

## 2. Постановка задачи

Опишем исходную систему. Система $M_2|G_2|1$ состоит из одного прибора и бесконечной очереди. На вход системы поступают два пуассоновских потока требований. Интенсивность первого потока равна $\lambda_1 = \lambda\alpha_1$, интенсивность второго потока равна $\lambda_2 = \lambda\alpha_2$ ($\alpha_1, \alpha_2 > 0$, $\alpha_1 + \alpha_2 = 1$). Длительность обслуживания требований из первого потока постоянна и равна $a$, длительность обслуживания требований из второго потока с вероятностью $1-p$ равна $b_1$, а с вероятностью $p$ равна $b_1 + b_2$. Будем считать, что выполняется условие существования стационарного режима: $\lambda_1 a + \lambda_2(b_1 + pb_2) < 1$. Допускается прерывание обслуживания требований обоих типов с последующим дообслуживанием. Правило, выбирающее порядок обслуживания требований, назовем дисциплиной обслуживания.

Через $L_1^*$ обозначим среднюю длину очереди требований первого типа, через $L_2^*$ – среднюю длину очереди требований второго типа.

Определим приоритетное множество:

$$M = \{(L_1^*, L_2^*),\ \text{по всем дисциплинам обслуживания}\}. \tag{1}$$

Требуется описать приоритетное множество $M$.

## 3. Результат

Приоритетное множество $M$ является выпуклой оболочкой точек на плоскости $(L_1^*, L_2^*)$, принадлежащих шести следующим множествам:

$$\begin{cases} L_1^* = \left\{\rho_1^2 + \lambda_1\lambda_2(b_1 - y_2)^2 + p\lambda_1\lambda_2(b_2 - z_1)^2\right\}\dfrac{1}{2(1-\rho_1)} \\ L_2^* = \left\{\dfrac{\lambda_2^2(b_1^2 - 2b_1y_2)}{2(1-\rho_1)} + \dfrac{p\lambda_2^2(b_2^2 - 2b_2z_1)}{2(1-\rho_1)} + \dfrac{p\lambda_2^2b_2}{1-\rho_1}z_1 + \dfrac{\lambda_2^2(b_1 + pb_2\rho_1)}{1-\rho_1}y_1 + \dfrac{\lambda_2a\rho_1}{1-\rho_1} + \right. \\ \qquad \left. + \rho_2\rho_3\right\}\dfrac{1}{1-\rho} + \dfrac{\lambda_2\rho_1}{1-\rho_1}y_2 + \dfrac{p\lambda_2\rho_1}{1-\rho_1}z_1 + \dfrac{p\lambda_2\rho_1(b_1 - y_2)}{1-\rho_1}\left(\dfrac{\lambda_2b_2}{1-\rho} + 1\right) \end{cases} \tag{2}$$

$$\begin{cases} L_1^* = \left\{\lambda_1\lambda_2(b_1 - y_1)^2 + p\lambda_1\lambda_2(b_2 - z_2)^2 + \rho_1^2\right\}\dfrac{1}{2(1-\rho_1)} \\ L_2^* = \left\{\dfrac{p\lambda_2^2}{1-\rho_1}\left(\dfrac{z_1^2}{2} - b_2z_1\right) + \dfrac{\rho_1^2 + \rho_2^2 + \lambda_2b_2\rho_3}{2(1-\rho_1)} + \lambda_1\lambda_2a^2/2\right\}\dfrac{1}{1-(\rho_1+\rho_2)}\left(\dfrac{p(\rho_2+\rho_3)}{1-\rho} + 1\right) + \\ \quad + \left\{\dfrac{p\lambda_2(\rho-1)\rho_1}{1-\rho_1}y_1 + \dfrac{p\lambda_2^2}{1-\rho_1}\left(b_2z_1 - \dfrac{z_1^2}{2}\right) + p\rho_2\rho\right\}\dfrac{1}{1-\rho} + \dfrac{\lambda_2\rho_1}{1-\rho_1}y_1 + \dfrac{p\lambda_2\rho_1}{1-\rho_1}(z_2 - z_1) \end{cases} \tag{3}$$

$$
\begin{cases}
L_1^* = \left\{\dfrac{\lambda_1 b_1 \rho_2 + \rho_1^2 + \lambda_1 b_2 \rho_3}{2(1-\rho_2)} + \dfrac{p\lambda_1\lambda_2}{1-\rho_2}\left(b_2 z_1 + \dfrac{z_1^2}{2}\right)\right\}\dfrac{1}{1-(\rho_1+\rho_2)} + \dfrac{\lambda_1\rho_2}{1-\rho}x_1 \\
L_2^* = \left\{\dfrac{\lambda_1 b_1 \rho_2 + \rho_1^2 + \lambda_1 b_2 \rho_3}{2(1-\rho_2)} + \dfrac{p\lambda_1\lambda_2}{1-\rho_2}\left(b_2 z_1 + \dfrac{z_1^2}{2}\right)\right\}\dfrac{1}{(1-(\rho_1+\rho_2))}\dfrac{p\lambda_2 a}{(1-\rho)} + \\
\quad + \left\{\lambda_1\lambda_2(a-x_1)^2 + p\lambda_2^2(b_2-z_2)^2 + \rho_2^2\right\}\dfrac{1}{2(1-\rho_2)}\left(\dfrac{p\rho}{1-\rho}+1\right) + \\
\quad + \left\{\dfrac{p\lambda_1\lambda_2}{1-\rho_2}x_1(a-x_1/2) + \dfrac{p\lambda_2^2}{1-(\rho_2+\rho_3)}z_1(b_2-z_1/2) + \dfrac{p^2\lambda_2^2}{1-\rho_2}(z_2-z_1)\left(b_2-\dfrac{z_2+z_1}{2}\right) + \right. \\
\quad \left. + p\rho_1\rho_2 + p(\rho_2+\rho_3)\right\}\dfrac{1}{1-\rho} + \dfrac{p\lambda_2(\rho_1+\rho_2)}{1-(\rho_1+\rho_2)}z_1 + \dfrac{p\lambda_2\rho_2}{1-\rho_2}(z_2-z_1)
\end{cases} \tag{4}
$$

$$
\begin{cases}
L_1^* = \left\{\dfrac{\lambda_1(\rho_2+\rho_3)}{1-(\rho_2+\rho_3)}\left[\dfrac{x_1^2}{2} + (a-(x_2-x_1))(x_2-x_1)\right] + \rho_1^2/2\right\}\dfrac{1}{1-\rho} \\
L_2^* = \left\{\lambda_1\lambda_2(a-x_2)^2 + p\lambda_2^2(b_2-z_1)^2 + \rho_2^2\right\}\dfrac{1}{2(1-\rho_2)}\left[\dfrac{p(\rho_2+\rho_3)}{1-(\rho_2+\rho_3)}+1\right] + \dfrac{p\lambda_2\rho_2}{1-\rho_2}z_1 + \\
\quad + \left\{\dfrac{p\lambda_1\lambda_2}{1-\rho_2}(x_2-x_1)\left(a-\dfrac{x_2+x_1}{2}\right) + \dfrac{p^2\lambda_2^2}{1-\rho_2}z_1(b_2-\dfrac{z_1}{2}) + p\rho_2(\rho_2+\rho_3)\right\}\dfrac{1}{1-(\rho_2+\rho_3)}
\end{cases} \tag{5}
$$

$$
\begin{cases}
L_1^* = \dfrac{\lambda_1\rho_3(2b_1+b_2) + \rho_1^2 + \lambda_1\lambda_2 b_1^2}{2(1-\rho)(1-(\rho_2+\rho_3))} + \dfrac{\lambda_1(\rho_2+\rho_3)}{1-(\rho_2+\rho_3)}x_1 \\
L_2^* = \dfrac{\lambda_1\lambda_2(a-x_1)^2 + \rho_2^2 + \lambda_2\rho_3(2b_1+b_2)}{2(1-(\rho_2+\rho_3))}
\end{cases} \tag{6}
$$

$$
\begin{cases}
L_1^* = \dfrac{\lambda_1\lambda_2(b_1-y_1)^2 + 2\lambda_1\rho_3(b_1-y_1) + \rho_1^2 + \lambda_1 b_2\rho_3}{2(1-\rho_1)} \\
L_2^* = \left\{\dfrac{(\rho_2 - \lambda_a 2\rho_3)\rho_1}{1-\rho_1}y_1 + \dfrac{\rho_2^2 + \lambda_2 a\rho_1 + \lambda_2 b_2\rho_3 + 2\rho_2\rho_3}{2(1-\rho_1)}\right\}\dfrac{1}{1-\rho} + \dfrac{\lambda_2\rho_1}{1-\rho_1}y_1
\end{cases} \tag{7}
$$

**Константы** $\rho_1, \rho_2, \rho_3, \rho$ равны: $\rho_1 = \lambda_1 a$, $\rho_2 = \lambda_2 b_1$, $\rho_3 = p\lambda_2 b_2$, $\rho = \rho_1 + \rho_2 + \rho_3$. Произвольные параметры $x_1, x_2, y_1, y_2, z_1, z_2$ изменяются в следующих пределах: $0 \leqslant x_1 < x_2 \leqslant a$, $0 \leqslant y_1 < y_2 \leqslant b_1$, $0 \leqslant z_1 < z_2 \leqslant b_2$.

**Пример** множества $M$ приведён на рисунке 1:

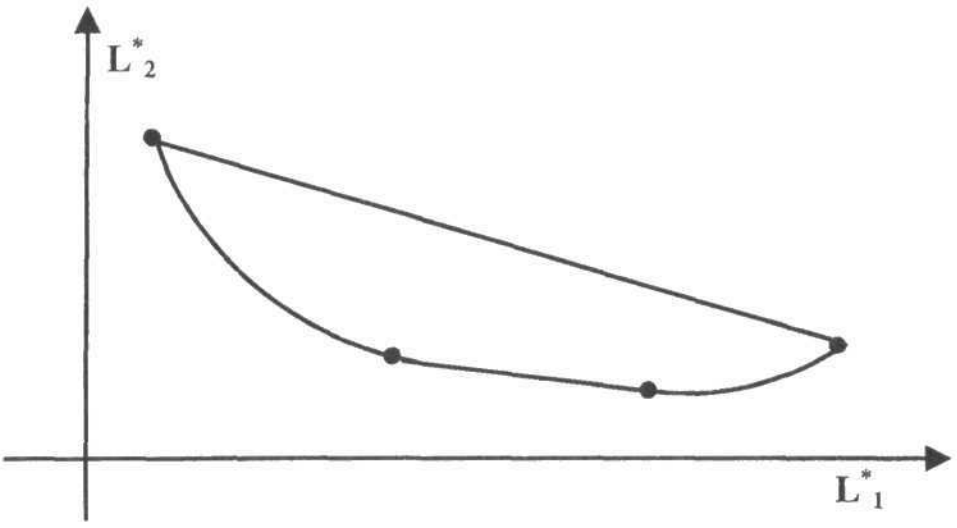


Рис. 1. Пример множества $M$.

## 4. Схема нахождения приоритетного множества

**Опишем** основные этапы нахождения приоритетного множества $M$.

1) Выберем достаточно малый параметр $\theta$ и рассмотрим те дисциплины обслуживания, которые допускают прерывание обслуживания только в моменты времени, кратные $\theta$. Для таких дисциплин

обслуживания, рассматриваемая система сводится к системе с ветвящимися потоками вторичных требований (СВПВТ), изученной в работах [2] и [1]. Обозначим через $L_i$, $i = \overline{1, m+n}$ средние длины очередей в этой системе.

2) В работе [4] показано, что приоритетное множество для СВПВТ является многогранником в $R^{n+m}$, обозначим его $P(\theta)$.

3) Пусть $M_P(\theta)$ - проекция $P(\theta)$ на плоскость $\{(L_1^*, L_2^*)\}$. При этой проекции точка с координатами $(L_1, \ldots, L_{m+n})$ переходит в точку $(L_1^* = \sum\limits_{1}^{m} L_i, L_2^* = \sum\limits_{m+1}^{m+n} L_i)$. Проекции вершин многогранника $P(\theta)$, в которых достигается минимум линейной функции

$$F = C_1 \sum_{1}^{m} L_i + C_2 \sum_{m+1}^{m+n} L_i, \tag{8}$$

и будут вершинами многоугольника $M_P(\theta)$. Таким образом, для нахождения многоугольника $M_P(\theta)$, нужно найти все те вершины многогранника $P(\theta)$, в которых достигается минимум линейной функции (8) при всевозможных значениях $C_1$ и $C_2$.

4) Приоритетное множество $M$ находится как $\lim\limits_{\theta \to 0} M_P(\theta)$.

## 5. Сведение к СВПВТ

Опишем подробнее первый этап решения. СВПВТ строится следующим образом. Выберем $0 < \theta \ll 1$, так чтобы $a = m\theta$, $b_1 = k\theta$, $b_2 = s\theta$, для простоты будем считать $m, s, k$ натуральными. Будем рассматривать только те дисциплины обслуживания, которые допускают прерывания обслуживания в моменты времени, кратные $\theta$. В результате получаем систему, имеющую $m + k + s$, $(k + s = n)$ новых типов требований, которая определяется следующим образом:

1) система состоит из одного прибора и $m + (k + s)$ типов требований;

2) на вход системы поступает пуассоновский поток заявок с интенсивностью $\lambda$; Поступившее требование с вероятностью $\alpha_1$ считается требованием 1 типа, с вероятностью $\alpha_2 = 1 - \alpha_1$ требованием $(m + 1)$ типа;

3) каждое из $(m + n)$ типов требований имеет постоянное время обслуживания $\theta$;

4) в каждый момент времени обслуживается только одно требование, прерывание обслуживания не допускается;

5) требования одного типа обслуживаются в порядке поступления;

6) после завершения обслуживания требований $i$-ого типа, с вероятностью $q_{ij}$ появляется требование $j$-ого типа, которое ставится в очередь на обслуживание, где:

$$q_{ij} = \begin{cases} 1, & j = i + 1,\ i \neq m, i \neq m + k, i \neq m + n. \\ p, & 0 \leqslant p \leqslant 1,\ i = m + k, j = m + k + 1. \\ 0, & \text{в остальных случаях.} \end{cases} \tag{9}$$

Таким образом вышеописанная СВПВТ совпадает с системой, изученной в [1].

## 6. Дисциплины обслуживания

Для нахождения минимума функции (8) нам необходимо описать те дисциплины обслуживания, при которых он достигается (различным значениям констант $C_1$ и $C_2$ будут соответствовать различные дисциплины обслуживания). Эта задача решалась в работах [3] и [4]. Приведем необходимые нам результаты этих работ.

Каждая дисциплина обслуживания, при которой достигается минимум произвольной линейной функции от средних длин очередей $L_i$ в СВПВТ, задается перестановкой $\sigma = (\sigma_1, \ldots, \sigma_{m+n})$ множества индексов $\{1, \ldots, m+n\}$ следующим образом: требования, имеющие тип $\sigma_i$, будут ставиться на обслуживание раньше требований типа $\sigma_j$ если $i < j$. Опишем вспомогательные величины, которые необходимы для вычисления $\sigma$.

Введём величины $\gamma_i$ - математическое ожидание суммарной длительности обслуживания (без учета ожидания) требований типа $i$ и всех требований типа $i+1, \ldots, m$ (если $i \leqslant m$), или $i+1, \ldots, m+n$ (если $i > m$). Ясно, что:

$$\gamma_i = \begin{cases} (m-i+1)\theta, & 1 \leqslant i \leqslant m. \\ (m+k-i+1+ps)\theta, & m+1 \leqslant i \leqslant m+k. \\ (m+n-i+1)\theta, & m+k+1 \leqslant i \leqslant m+n. \end{cases} \tag{10}$$

Введём величины $R_i$, которые будем называть приоритетными индексами требований $i$-ого типа (константы $C_1$ и $C_2$ входят в уравнение функции (8)).

$$R_i = \begin{cases} \dfrac{C_1}{\gamma_i}, & 1 \leqslant i \leqslant m, \text{ и } C_1 > 0. \\ \dfrac{C_2}{\gamma_i}, & m+k+1 \leqslant i \leqslant m+n, \text{ и } C_2 > 0. \\ \max\left\{\dfrac{C_2}{\gamma_i}, \dfrac{C_2(1-p)}{(m+k-i+1)\theta}\right\}, & m+1 \leqslant i \leqslant m+k, \text{ и } C_2 > 0. \\ 0, & \forall i, \text{ при } C_1 < 0, \text{ или } C_2 < 0. \end{cases}$$

В работе [3] доказано, что дисциплина обслуживания, при которой достигается минимум функции (8), заключается в следующем: если $R_i > R_j$, то требование типа $i$ должно ставиться на обслуживание раньше требования типа $j$; если же два требования имеют одинаковый приоритетный индекс, то порядок их обслуживания может быть выбран произвольно.

Из этих результатов следует, что перестановки $\sigma$ для СВПВТ, определяющие дисциплины обслуживания, при которой для заданных значений $C_1$ и $C_2$ достигается минимум функции (8), будут следующими:

1) для $C_1 > 0$, $C_2 > 0$:

$$\sigma = (*, 1, v_2, v_2 - 1, \ldots, v_1 + 1, w_1, w_1 - 1, \ldots, m+k+1, v_1, v_1 - 1, \ldots, m+1) \tag{11a}$$
$$\sigma = (*, 1, w_2, w_2 - 1, \ldots, w_1 + 1, v_1, v_1 - 1, \ldots, m+1, w_1, w_1 - 1, \ldots, m+k+1) \tag{11b}$$
$$\sigma = (*, m+1, w_2, w_2 - 1, \ldots, w_1 + 1, u_1, u_1 - 1, \ldots, 1, w_1, w_1 - 1, \ldots, m+k+1) \tag{11c}$$
$$\sigma = (*, m+1, u_2, u_2 - 1, \ldots, u_1 + 1, w_1, w_1 - 1, \ldots, m+k+1, u_1, u_1 - 1, \ldots, 1) \tag{11d}$$
$$\sigma = (*, m+1, u_1, u_1 - 1, \ldots, 1) \tag{11e}$$
$$\sigma = (*, 1, v_1, v_1 - 1, \ldots, m+1) \tag{11f}$$

2) для $C_1 < 0$, $C_2 > 0$:

$$\sigma = (*, m+1, w_1, w_1 - 1, \ldots, m+k+1, m, \ldots, 2, 1) \tag{12a}$$
$$\sigma = (*, m+1, m, \ldots, 2, 1) \tag{12b}$$

3) для $C_1 > 0$, $C_2 < 0$:

$$\sigma = (*, 1, m+n, m+n-1, \ldots, m+k+1, m+k, \ldots, m+2, m+1) \tag{13}$$

4) для $C_1 < 0$, $C_2 < 0$:

$$\sigma = (*, 1, m+1) \tag{14a}$$
$$\sigma = (*, m+1, 1) \tag{14b}$$

Здесь произвольные параметры $u_1, u_2, v_1, v_2, w_1, w_2$ являются натуральными и изменяются в следующих пределах: $1 < u_1 < u_2 \leqslant m$, $m+1 < v_1 < v_2 \leqslant m+k$, $m+k+1 < w_1 < w_2 \leqslant m+n$. При этом '$*$' - означает любую перестановку оставшихся индексов из множества $\{1, \ldots, m+n\}$.

Из описания СВПВТ очевидно следует, что для любых требований (за исключением требований 1, $m+1$ и $m+k+1$ типов), порядок обслуживания между ними можно менять произвольным образом, если выполняется следующее условие: приоритетные индексы $R_i$ всех этих требований должны удовлетворять одному и тому же из неравенств:

$$R_1 \geqslant R_i \geqslant R_{m+1}; \qquad R_{m+1} \geqslant R_i \geqslant R_1;$$
$$R_1 \geqslant R_i \geqslant R_{m+k+1} \geqslant R_{m+1}; \qquad R_{m+1} \geqslant R_i \geqslant R_{m+k+1} \geqslant R_1;$$

Данное свойство было использовано при вычислении перестановок (11a)-(11d).

Перестановки (12a)-(14b) очевидно являются частными случаями перестановок (11a)-(11f), поэтому их можно не рассматривать.

## 7. Сведения, необходимые для вычисления вершин $M_P(\theta)$

Для нахождения вершин многогранника $M_P(\theta)$ необходимо вычислить величины $L_i$ (средние длины очередей требований в СВПВТ) для заданных дисциплин обслуживания, задаваемых перестановками, при которых реализуется минимум функции (8). Все необходимые нам перестановки описаны в уравнениях (11a)-(11f). Приведем необходимые нам результаты работ [1] и [4].

Введём величины $x_{ij}$, являющиеся решениями следующей системы линейных уравнений:

$$\begin{aligned} &\sum_{i=1}^{m+n} a_{ij}x_{ir} + \sum_{i=1}^{m+n} a_{ir}x_{ij} = G_{jr}, \quad 1 \leqslant j \leqslant r \leqslant m+n. \\ &x_{\sigma_i\sigma_j} = 0, \quad \text{если } i > j. \end{aligned} \tag{15}$$

где константы $a_{ij}$ и $G_{ij}$ определяются как:

$$\begin{array}{lll} a_{ii} = 1, & i \neq 1, i \neq m+1; & a_{11} = 1 - \lambda_1\theta; \\ a_{ii+1} = -1, & i \neq m, i \neq m+k; & a_{m+1m+1} = 1 - \lambda_2\theta; \\ a_{i1} = -\lambda_1\theta, & i \neq 1; & a_{m+km+k+1} = -p; \\ a_{im+1} = -\lambda_2\theta, & i \neq m+1; & a_{ij} = 0, \text{ в остальных случаях;} \end{array} \tag{16}$$

$$\begin{aligned} &G_{ij} = G_{ji}, \quad \forall i, j; && G_{ij} = 0, \quad j \neq 1, j \neq m+1, i \neq 1, i \neq m+1; \\ &G_{11} = 2\lambda_1^2\theta^2(\lambda_1 m + \lambda_2(k+ps))/2; && G_{m+1m+1} = 2\lambda_2^2\theta^2(\lambda_1 m + \lambda_2(k+ps))/2; \\ &G_{1m+1} = 2\lambda_1\lambda_2\theta^2(\lambda_1 m + \lambda_2(k+ps))/2; \\ &G_{1j} = \begin{cases} \lambda_1^2\theta, & 1 \leqslant j \leqslant m. \\ \lambda_1\lambda_2\theta, & m+1 \leqslant j \leqslant m+k. \\ p\lambda_1\lambda_2\theta, & m+k < j \leqslant m+n. \end{cases} && G_{m+1j} = \begin{cases} \lambda_1\lambda_2\theta, & 1 \leqslant j \leqslant m. \\ \lambda_2^2\theta, & m+1 \leqslant j \leqslant m+k. \\ p\lambda_2^2\theta, & m+k < j \leqslant m+n. \end{cases} \end{aligned} \tag{17}$$

Тогда, как показано в работе [1], средние длины очередей для СВПВТ находятся следующим образом.

$$\begin{aligned} L_1 &= \sum_{i=1}^{m+n} x_{i1}\theta + \lambda_1\theta^2(\lambda_1 m + \lambda_2(k+ps))/2. \\ L_{m+1} &= \sum_{i=1}^{m+n} x_{i1}\theta + \lambda_2\theta^2(\lambda_1 m + \lambda_2(k+ps))/2. \\ L_i &= \sum_{i=1}^{m+n} x_{i1}\theta, \quad i \neq 1, i \neq m+1. \end{aligned} \tag{18}$$

Для решения уравнений (15) и вычисления $L_i$ будет полезным следующее утверждение:

**Утверждение 1.** *Если приоритетный индекс $R_j$ удовлетворяет неравенству: $R_j \leqslant \max\{R_1, R_{m+1}\}$, тогда $L_j = 0$ и для любого $i$: $x_{ij} = 0$.*

*Доказательство.* Из описания дисциплины обслуживания для СВПВТ задаваемой перестановкой $\sigma$, следует, что требования типа $j$ имеют больший приоритет, чем требования 1 и $m+1$ типов. Следовательно, они будут ставиться на обслуживание сразу после появления в системе. Тогда длина соответствующей им очереди будет равна нулю. Величины $x_{ij} = 0$, $\forall i$ будут удовлетворять уравнениям (15), (18), и поэтому являются их решениями. □

## 8. Построение приоритетного множества $M$

Чтобы найти приоритетное множество $M$, необходимо вычислить величины $L_1^*$, $L_2^*$. Тогда для каждой перестановки $\sigma$ (11a)-(11f) из уравнений (15)-(18) найдем величины $L_i$ - которые определяют вершины многоугольника $M_P(\theta)$. Далее надо вычислить пределы $\lim\limits_{\theta\to 0}\sum\limits_{1}^{m} L_i, \lim\limits_{\theta\to 0}\sum\limits_{m+1}^{m+n} L_i$ – они и будут искомыми величинами $L_1^*$, $L_2^*$. Эти громоздкие вычисления занимают несколько страниц, но являются достаточно стандартными, поэтому не приводятся. При этом параметры $u_1, u_2, v_1, v_2, w_1, w_2$ и $x_1, x_2, y_1, y_2, z_1, z_2$ будут связаны следующим образом:

$$x_1 = \lim_{\theta\to 0} u_1; \quad y_1 = \lim_{\theta\to 0} v_1; \quad z_1 = \lim_{\theta\to 0} w_1; \quad x_2 = \lim_{\theta\to 0} u_2; \quad y_2 = \lim_{\theta\to 0} v_2; \quad z_2 = \lim_{\theta\to 0} w_2;$$

В результате получаются формулы (2)-(7), задающие соответствующие множества на плоскости $(L_1^*, L_2^*)$, выпуклой оболочкой которых является приоритетное множество $M$.

## Литература

[1] *Китаев М.Ю., Рыков В.В.* Система обслуживания с ветвящимися потоками вторичных требований // АиТ. 1980. 9. С. 52-61.

[2] *Климов Г. П.* Системы обслуживания с разделением времени. I // ТВП. 1974. Вып. 3. Т. XIX. С. 558-576.

[3] *Климов Г. П.* Системы обслуживания с разделением времени. II // ТВП. 1978. Вып. 2. Т. XXIII. С. 331-339.

[4] *Тимофеев Е.А.* Оптимизация средних длин очередей в системе обслуживания с ветвящимися потоками вторичных требований // АиТ. 1995. С. 60-67.

*Модел. и анализ информ. систем.* Т.8, №1 (2001) 25–37

УДК 681.326

# Особенности построения функционально-информационной структуры современных систем промышленной автоматизации

Асеев Д.И., Воронина Т.В., Зафиевский А.В., Курчидис В.А.,
Лунев М.Ю., Назанский А.С., Рыльков С.А.
*Ярославский государственный университет*
*150 000, Ярославль, Советская, 14*

Проводится анализ состояния и разнообразия информационных технологий и средств, используемых при построении современных СПА. Одновременно с проведением этого анализа обсуждаются некоторые особенности, определяемые различными факторами, которые затрагивают основные аспекты построения систем рассматриваемого класса. Приводится описание принципов построения функциональной и информационной структуры распределенных СПА, которые ориентированы на использование методов и средств компонентных программных технологий.

В общем виде структура распределенной системы промышленной автоматизации представляется как совокупность устройств (узлов), взаимодействующих через коммуникационные сети и обменивающихся информацией в режиме реального времени под управлением программных средств, функционирующих в соответствии с конкретным проектом.

Предлагаемое в работе описание информационно-функциональной структуры СПА сочетает элементы объектно-ориентированного проектирования и компонентных технологий и рассматривается как основа для одного из системных решений соответствующей задачи, которое позволяет в достаточной степени учитывать влияние основных факторов, определяющих особенности построения СПА.

## 1. Введение

Современный этап развития систем промышленной автоматизации (СПА) характеризуется практически полным переходом от аналоговых к цифровым технологиям, основанным на широком использовании вычислительной техники, сетевых средств, информационного и программного обеспечения благодаря таким преимуществам цифровых систем, как гибкость и экономичность решений, информативность, масштабируемость, надежность, безопасность и другие. Структура СПА приобретает распределенный характер, вследствие чего обобществляется доступ к данным реального времени, получаемым в ходе контроля объектов и процессов автоматизации.

Построение современных СПА — сложная организационно-техническая задача, успешное решение которой во многом определяется наличием развитых аппаратных и программных средств, обеспечивающих организацию единого информационного пространства и функционирование в нем элементов системы. При построении такой функционально-информационной структуры необходимо учитывать целый ряд факторов, определяемых особенностями как самих систем, так и средств аппаратно-программной поддержки.

Наиболее важными факторами, влияющими на процесс построения такой структуры в системах промышленной автоматизации, являются следующие:

- широкое внедрение сетевых технологий, обусловленное распределенным характером систем промышленной автоматизации;
- использование развитых программных средств на основе операционных систем реального времени, связанное с необходимостью функционирования систем автоматизации в режиме реального времени;
- применение современных информационных технологий, позволяющее обеспечить высокую степень их интеграции в комплексных системах управления и информационных системах различных уровней;
- использование разнообразных инструментальных средств, новых подходов к технологии программирования, вызванное необходимостью применения широкого спектра приложений и быстрой их разработки.

В данной работе проводится анализ состояния и разнообразия информационных технологий и средств, используемых при построении современных СПА. Одновременно с проведением этого анализа обсуждаются некоторые особенности, определяемые перечисленными выше факторами, которые затрагивают основные аспекты построения систем рассматриваемого класса. Приводится описание принципов построения функциональной и информационной структуры распределенных СПА, которые ориентированы на использование методов и средств компонентных программных технологий и которые в достаточной степени учитывают влияние всех перечисленных выше факторов.

# 2. Основные факторы, определяющие особенности построения функционально-информационной структуры СПА

## 2.1. Использование сетевых технологий

Сеть является той интегрирующей физической средой, на основе которой осуществляется взаимодействие между рассредоточенными элементами распределенной системы автоматизации. В качестве таких элементов могут быть задействованы датчики, исполнительные механизмы, рабочие места операторов, серверы баз данных, сетевая аппаратура (мосты, коммутаторы, шлюзы и т.д.) и многое другое.

Важной особенностью сетей является иерархическая структура их построения. Сети нижнего уровня, или промышленные сети, обслуживающие реальные процессы, поддерживаются целым рядом промышленных сетевых протоколов: CAN, LON, PROFIBUS, Interbus-S, FIP, FF, Device NET, SDS, ASI, HART и другими [1, 2]. Выход на более высокие уровни в системы визуализации данных, в комплексные автоматизированные системы управления технологическими процессами (АСУТП) и производством (АСУП), коммерческие и административные системы организуется через стандартные офисные LAN-сети типа Ethernet, Token Ring, FDDI и другие с использованием стеков протоколов TCP/IP, IPX/SPX, NetBIOS/SMB, а также на основе базовых технологий корпоративных и глобальных сетей Intranet/Internet.

Для сетей нижнего уровня, работающих в режиме реального времени, актуальны проблемы выбора протокола и совместимости устройств, порождаемые многообразием протоколов, особенностями каждой промышленной сети, отсутствием единых международных стандартов на промышленные сети. Аналогичные проблемы возникают и для сетей верхних уровней, но они не столь актуальны, так как решаются на основе существующих стандартов.

Выбор конкретных решений осуществляется исходя из экономических соображений, функционального назначения и обеспечения совместимости с используемым оборудованием и программными средствами, учитывая открытость сетевых технологий и уровень их поддержки в стране.

Так, например, решение проблемы согласования различных промышленных сетей возможно на основе использования готовых аппаратно-программных шлюзов либо на построении коммуникационных серверов. Не составляет особой трудности найти межпротокольный шлюз, например ASI-PROFIBUS, сложнее организовать коммуникационные серверы с функциями обработки и архивации данных с последующей передачей их на верхние уровни. Эти задачи могут быть решены на различных процессорных архитектурах (VME, Compact PCI, Open PLC, PCI, ISA и т.п.) и под управлением широкого класса операционных систем VxWorks, OS9, QNX, Windows NT, Windows CE и других [1].

Таким образом, при построении информационно-программных комплексов для систем промышленной автоматизации основные проблемы сосредоточены в области организации взаимодействия сетей различных уровней. Функционирование системы в режиме реального времени предъявляет высокие требования к временным параметрам сетей и средствам их программной поддержки, реализация которых возможна на основе операционных систем реального времени.

## 2.2. Применение операционных систем реального времени

Операционная система, являясь платформой для построения программного комплекса, во многом определяет общую структуру информационно-программной среды, в которой функционирует система автоматизации. Поэтому весьма актуальна проблема выбора приемлемого варианта из множества известных операционных систем реального времени (ОСРВ) [3, 4, 5, 6]. Известны экономические и технические критерии выбора, наиболее важные параметры которых отражают интегральную стоимость программного продукта, а также время задержки реакции системы [3].

Коммерческие ОСРВ жесткого реального времени (Vx Works, OS9, QNX и другие), не допускающие задержек реакции системы, имеют, как правило, высокую стоимость (около 10-20 тыс. долларов),

содержат набор готовых драйверов, комфортные специализированные средства быстрой отладки, что резко сокращает продолжительность разработки, особенно при построении сложных систем реального времени.

Расширения реального времени RTAI, RT Linux, RTX для операционных систем Linux, Windows NT, Windows CE, поддерживающие работу в режиме мягкого реального времени, имеют стоимость примерно на порядок ниже, чем коммерческие ОСРВ, предоставляют широкий выбор средств отладки и разработки, содержат обширный набор драйверов.

Большинство современных ведущих ОСРВ поддерживают целый спектр аппаратных архитектур (Intel, Motorola, RISC, MIPS, Power PC и др.). Набор аппаратных средств, являясь частью комплекса реального времени, должен адекватно отражать специфику решаемой задачи.

Таким образом, при построении сравнительно несложных систем промышленной автоматизации, в которых время реакции некритично и составляет не более 1 мс, можно остановиться на Windows NT как на программной платформе. Наличие развитых инструментальных средств, большого набора драйверов делает Windows NT весьма привлекательной не только при использовании в системах реального времени, но и при разработке прикладного программного обеспечения (ППО) для распределенных систем промышленной автоматизации, т.к. процесс создании ППО является достаточно длительным и дорогостоящим. Одной из важнейших задач, решаемых при разработке прикладных средств, является обеспечение информационного взаимодействия различных уровней системы промышленной автоматизации.

## 2.3. Интеграция информационных технологий

Использование высоких информационных технологий обеспечивает оперативный доступ к обширной корпоративной информации различных уровней, что позволяет осуществлять эффективное управление процессом, принятие тактических решений на основе информации, полученной от различных датчиков, программируемых контроллеров, организовать планирование производства, управление предприятием в целом. Проблема организации такого «сквозного» сетевого доступа решается на основе применения инструментальных программных средств типа SCADA-систем. Наиболее популярны SCADA-системы: Factory Link, InTouch, Genesis, Real Flex, Sitex, FIX, Trace Mode, IGSS, Image, RSView и другие [7, 8].

Основу большинства SCADA-систем составляют несколько программных компонентов (база данных реального времени, ввода-вывода, предыстории, аварийных ситуаций) и администраторов (доступа, управления, сообщений).

Большинство SCADA-систем реализовано на платформах MS Windows, однако в таких SCADA-системах, как Real Flex и Sitex, основу программной платформы составляет операционная система реального времени QNX.

В комплексе программных компонентов «сквозной» автоматизации производства SCADA-системы в основном поддерживают цеховой уровень автоматизации, связанный с получением и визуализацией информации от контроллеров, распределенных систем управления, т.е. информации, поступающей с уровня промышленных сетей. Важным достоинством подобных программных инструментов является наличие средств доставки информации со SCADA-уровня наверх, на уровень управления (АСУТП, АСУП) или планирования производства. Ряд фирм предлагают продукты, представляющие собой системы управления производством [7].

Огромное значение имеет возможность взаимодействия инструментальных средств АСУТП с распространенными офисными программными продуктами такими, как Microsoft Back Office Suite. Например, фирма Wonderware предлагает программные средства, которые интегрируются с такими продуктами, как MS SQL Server, Windows NT Server, System Management Server, SNA Server, Mail Server. Этой же фирмой на базе MS SQL Server построен Industrial SQL Server, позволяющий в реальном масштабе времени регистрировать данные, поступающие с InTouch-сервера ввода-вывода.

Актуальным становится требование передачи статической и динамической информации на Web-узлы. В некоторых браузерах объекты ActiveX позволяют передавать данные из SCADA-системы на Web-страницы. В настоящее время уже разработаны и существуют многофункциональные компоненты типа Scout фирмы Wonderware, обеспечивающие возможность доступа к системам автоматизации на базе InTouch через Internet/Intranet и позволяющие удаленному пользователю взаимодействовать с прикладной задачей автоматизации, как с простой Web-страницей [7].

Интеграция используемых информационных технологий позволяет объединить разнородные средства автоматизации в единый информационный комплекс — единое информационное пространство, в котором обеспечивается предоставление данных реального времени всем уровням систем контроля и управления: АСУТП/SCADA-систем, АСУП/ERP-систем.

## 2.4. Использование технологий программирования

Применение SCADA-систем неизбежно приводит к проблеме разработки приложений, решение которой возможно с использованием инструментов самих SCADA-систем, традиционных языков программирования или компонентных технологий. Объектно-ориентированное проектирование, сочетающееся с компонентной технологией, позволяет радикально изменить принципы организации и построения систем [9, 10].

Опираясь на стандарты, компонентные технологии позволяют заменить сегодняшние уникальные решения стройной архитектурой взаимозаменяемых программных модулей и приложений. Применение новых технологий позволяет обеспечить совместимость устройств, систем управления и информационных систем.

Новые компонентные архитектуры нашли отражение в таких технологиях для среды Windows, как COM/DCOM, OLE, ActiveX. Основой многих компонентных технологий является модель COM (Component Object Model) — компонентная объектная модель, предложенная фирмой Microsoft [9]. Модель COM обеспечивает связь между разными процессами на одной и той же машине.

Распределенная модель COM — DCOM (Distributed COM) отвечает за взаимодействие между процессами на разных машинах. Компонент подобен мини-приложению, он поставляется как двоичный код, готовый к использованию. Из накапливаемого набора компонентов в библиотеках можно быстро собирать требуемые приложения. Модификация или расширение приложения сводится к замене одного из составляющих его компонентов новой версией. COM — это спецификация, указывающая, как создавать динамически взаимозаменяемые компоненты. Компоненты COM состоят из исполняемого кода, распространяемого в виде динамически компонуемых библиотек DLL или EXE-файлов Win32.

Информационная система автоматизации базируется на приложениях, компоненты которых ответственны за управление технологическим процессом в реальном времени, визуализацию, выборку информации. Многие распределенные приложения надо интегрировать в уже существующие на производстве сетевые инфраструктуры. Модель COM поддерживает ряд распространенных транспортных протоколов, включая TCP/IP,UDP, IPX/SPX, NetBIOS, поэтому программные средства комплексной автоматизации, использующие технологию DCOM, нечувствительны к сетевым протоколам и позволяют с небольшими затратами наращивать существующие системы.

Сложные распределенные приложения часто работают в неоднородной среде. Использование стандартной модели DCOM позволяет интегрировать компоненты на различных языковых платформах в одно распределенное приложение.

Объекты ActiveX базируются на технологии COM, устанавливающей общую схему взаимодействия программных компонентов, что обеспечивает стандартную инфраструктуру, позволяющую объектам обмениваться данными и функциями между приложениями. Любые ActiveX-объекты могут загружаться в систему разработки большинства SCADA-систем и использоваться при создании прикладных программ. Управление ActiveX-объектами осуществляется с помощью данных, методов и событийных функций, свойственных выбранному объекту.

Технология OLE (Object Linking and Embedding - включение и встраивание объектов) предложена компанией Microsoft как более эффективное и надежное средство передачи данных между процессами взамен DDE (Dinamic Data Exchange — стандартный динамический обмен данными). Механизм OLE поддерживается рядом SCADA-систем RSView, FIX, InTouch, Factory Link и др. На базе OLE создан новый стандарт — OPC (OLE for Process Control — OLE для процессов управления), ориентированный на рынок промышленной автоматизации. Стандарт OPC позволяет объединять на уровне объектов различные системы управления и контроля, функционирующие в распределенной гетерогенной среде, устраняет необходимость использования различного нестандартного оборудования и соответствующих программных драйверов.

Серверы OPC обеспечивают стандартный способ передачи информации в программу визуализации, базы данных и другие приложения. Появление OPC-серверов означает разработку программных стандартов обмена с технологическими устройствами. OPC-интерфейс допускает различные варианты обмена: получение данных с устройств, из распределенной системы управления или из любого приложения.

# 3. Построение функционально-информационной структуры распределенных СПА на основе компонентных технологий

## 3.1. Общая структура системы

В соответствии с принципами системотехнического подхода [11] процесс построения сложной программно информационной системы можно разбить на три этапа.

1) Постановка задачи. На этом этапе вводятся базовые понятия, определяется набор функций, типичный для большинства систем данного класса, создаются достаточно общие структурные модели, описываются сценарии работы.

2) Анализ системы. Основной задачей анализа системы является разделение целого на части, его декомпозиция. Система делится на подсистемы, при необходимости этот процесс повторяется, что приводит к иерархическим древовидным структурам. На этом этапе в системе выделяются компоненты, которые могут быть затем разбиты на подкомпоненты и т.д., определяются функции компонентов и интерфейсы между ними, описываются протоколы взаимодействия. Удовлетворяя принципам полноты и простоты, создается исчерпывающее описание системы в терминах компонентного проектирования.

3) Апробация и реализация. Создание сложных новых систем помимо адекватного описания системы требует исчерпывающего тестирования на соответствие этому описанию. Стандартизация позволяет создавать новые системы путем конструирования из существующих компонентов, прошедших апробацию в разнообразных условиях применения. Этап реализации характеризуется тем, что системы, разработанные на втором этапе и прошедшие апробацию, становятся стандартом для систем данного типа. Новые системы создаются на основе этих стандартов.

В данном разделе обсуждаются вопросы, относящиеся, главным образом, к первому этапу и связанные в большей степени с построением функционально-информационной структурной модели распределенной системы промышленной автоматизации. Изложение ведется с позиций объектно-ориентированного проектирования [12] и ориентировано на создание соответствующего информационно-программного комплекса на основе компонентных технологий [9].

В самом общем случае распределенная система промышленной автоматизации может быть представлена как система обработки данных, которую в свою очередь принято рассматривать как совокупность технических и программных средств.

Технические средства составляют основу системы и представляют собой оборудование для ввода, хранения, преобразования и вывода информации. Состав технических средств определяется структурой системы, т.е. тем, из каких элементов состоит система и как они связаны между собой. Основные технические элементы системы — устройства: датчики, контроллеры, процессоры, устройства сопряжения, запоминающие устройства, устройства ввода-вывода и т.д. Устройства связаны между собой с помощью интерфейсов, включающих в себя совокупность каналов передачи данных и программных средств их поддержки.

Программные средства, поддерживающие требуемый набор функций системы, строятся по многоуровневому иерархическому принципу. Проблема выбора операционной системы и прикладного программного обеспечения обсуждалась выше.

Функционирование системы определяется взаимодействием программных и технических средств, в результате чего свойства системы проявляются как совокупные свойства этих средств. Функционирование системы представляется в виде процессов — динамических объектов, реализующих целенаправленный акт обработки данных. Функционирование системы возможно в нескольких режимах, которые проявляются в порядке инициирования задач, распределения приоритетов при использовании ресурсов, в организации ввода данных, хранения программы в оперативной памяти, вывода данных и др. Режимы обработки данных порождают соответствующие режимы функционирования системы. С позиций режима обработки данных рассматриваемые системы относятся к системам реального времени, т.е. к информационно-управляющим системам, обеспечивающим передачу и обработку информации со скоростью, соответствующей скорости протекания управляющего процесса.

Таким образом, в общем случае структура распределенной системы промышленной автоматизации может быть представлена как совокупность устройств (узлов), взаимодействующих через коммуникационные сети и обменивающихся информацией в режиме реального времени под управлением программных средств, функционирующих в соответствии с конкретным проектом.

В соответствии с этим описанием структура системы промышленной автоматизации может быть представлена в виде схемы, изображенной на рис. 1. Эта структура, основу которой составляют программно-технические средства, определяет модель и общесистемные требования к функционированию реализуемой системы.

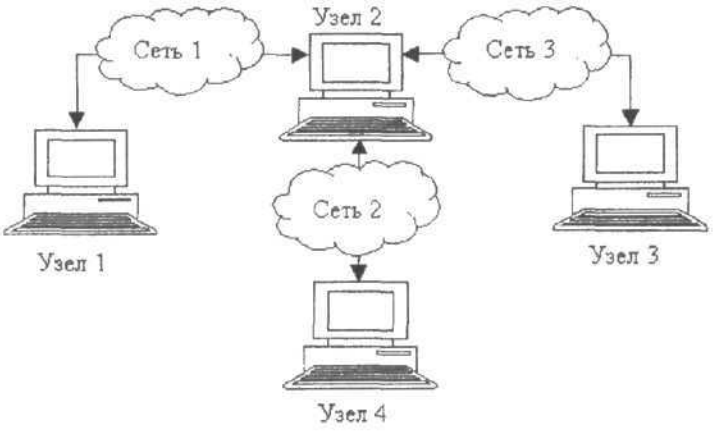


Рис. 1.

Следует отметить, что в целом идея подобного подхода к построению информационных систем не является принципиально новой, но ее использование применительно к реализации распределенных информационно-программных средств СПА упирается в проблемы комплексного характера, до настоящего времени не нашедшие своего законченного решения. Предлагаемое в работе описание функционально-информационной структуры СПА, сочетающее элементы объектно-ориентированного проектирования и компонентных технологий, можно рассматривать как основу для одного из решений соответствующей задачи, которое позволяет в достаточной степени учитывать влияние основных факторов, определяющих особенности построения СПА.

Прежде, чем перейти к более детальному обсуждению структуры и принципов функционирования системы, целесообразно определить некоторые понятия, активно используемые в дальнейшем изложении.

**Узел** — это оборудование (программно-аппаратный комплекс: компьютер, контроллер и т.п.), на котором функционирует один или несколько программных модулей.

**Конфигуратор** системы — набор правил, определяющих связи между каналами в системе, на основе которых осуществляется конфигурирование системы. Конфигуратор может быть статическим, например, если он используется при инсталляции системы. Динамический конфигуратор позволяет производить изменение конфигурации в процессе функционирования системы.

**Канал** — это некоторая уникальная метка в конфигураторе, однозначно связанная с источником данных — контроллером, сервером математической обработки, архивом, базой данных и т.д.

**Модуль** — это основная структурная единица системы, представляющая собой независимый двоичный объект (exe- или dll-файл), способный выполнять некоторый набор функций, сохранять и восстанавливать свое состояние, а также обмениваться данными с другими модулями по заранее определенным правилам.

Модули обмениваются между собой данными, которые могут поступать из различных источников. Для идентификации поступающих данных используется понятие канала. Модули посылают друг другу данные в виде сообщений, содержащих собственно данные (полезную информацию) и некоторую служебную информацию.

**Интерфейс** — это определенный набор описаний процедур и функций модуля, для которого имеются средства вызова извне. На основе интерфейса осуществляется взаимодействие модулей.

## 3.2. Структура и функционирование информационно-программной среды системы

### 3.2.1. Общая схема взаимодействия модулей в системе

В системе могут присутствовать три типа модулей: модули обработки, ядро, модули связи.

1) Модули обработки - это основной тип модулей в системе. Они обеспечивают доступ к реальным источникам данных (контроллерам, серверам математической обработки, архивам, базам данных и

т.п.). Модули обработки могут записывать и считывать информацию из этих источников, обмениваться данными между собой при помощи других типов модулей, а также выполнять комплексную либо специальную обработку данных. Функциональное назначение таких модулей может быть различным: архивация, визуализация, предварительная обработка и т.п.

2) Модули связи осуществляют передачу данных по сети между различными узлами системы. Для каждого протокола, используемого для передачи данных, создается отдельный модуль связи.

3) Ядро осуществляет функции диспетчера-маршрутизатора сообщений с отслеживанием механизма реального времени. Сообщения от модулей поступают в ядро, после чего ядро, в соответствии с конфигуратором, определяет адресата каждого сообщения и отсылает его соответствующему модулю.

Общая схема взаимодействия модулей в распределенной среде системы представлена на рис. 2.

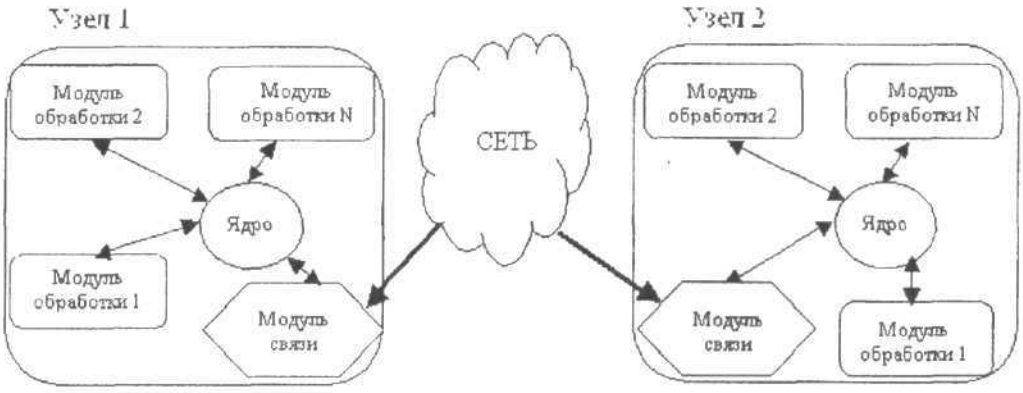


Рис. 2.

Каждый тип модулей имеет свой интерфейс, изображенный на схеме в виде стрелки, соединяющей эти модули. На каждом узле системы должно работать ровно одно ядро, а также некоторый набор модулей обработки и модулей связи. При запуске системы ядро читает файл конфигуратора и в соответствии с ним загружает в память нужные модули и передает им данные, необходимые для инициализации. Также из файла конфигуратора в ядро загружается таблица маршрутизации для данного узла, в которой описаны связи между каналами различных модулей этого узла. Если этап инициализации прошел успешно, то система начинает работу.

В самом общем виде процесс работы системы можно описать следующим образом: модули обработки получают данные из внешних источников (от контроллеров, из архива и т.п.), а модули связи — по сети от других узлов и передают их в ядро в виде сообщений. В ядре имеется очередь сообщений, куда добавляются данные от модулей в соответствии с приоритетами. Сообщения из этой очереди выбираются по порядку, и по таблице маршрутизации определяется адресат данного сообщения. Если сообщение предназначено модулю данного узла, то оно передается соответствующему модулю с указанием нужного канала. Если сообщение адресовано на другой узел системы, то оно передается ядром модулю связи, который, в свою очередь, посылает его по сети соответствующему узлу. На узле адресата должен работать такой же модуль связи, который получит сообщение и передаст его в ядро своего узла, а ядро отправит сообщение в нужный модуль.

Важным вопросом является способ передачи данных в ядро. Возможны два основных варианта:

- модуль передает данные по собственной инициативе, вызывая соответствующую процедуру ядра;
- модуль передает данные по запросу ядра, которое для этого вызывает процедуру модуля.

Достоинством первого подхода является то, что данные могут быть переданы в ядро без задержки, т.е. сразу же, как только они получены модулем. Но такой подход имеет и ряд недостатков, главным из которых является непредсказуемость системы: трудно прогнозировать, какова будет длина очереди сообщений в ядре и сколько времени потребуется на прохождение того или иного сообщения. Кроме того, каждый модуль должен будет иметь ссылку на ядро и таким образом получается слишком сильная связь между ядром и модулем, что противоречит принципам компонентных технологий.

Более разумным представляется использование второго подхода, позволяющего разбить работу системы на четко определенные циклы, каждый из которых состоит из определенной последовательности шагов. Цикл системы может выглядеть, например, так: ядро запрашивает данные у каждого модуля и размещает их в очереди сообщений в соответствии с приоритетом; затем из очереди выбираются по порядку сообщения и передаются модулям, которые их обрабатывают; после этого цикл повторяется сначала. Несомненным достоинством такого подхода является то, что модули становятся более независимыми. Они ничего не должны знать о ядре и поэтому могут предоставить данные любому приложению, которое вызовет соответствующую процедуру модуля.

С функциональной точки зрения можно считать, что модули обработки предоставляют на узле системы доступ к определенным видам сервиса, а ядро выступает в роли диспетчера, регламентирующего правила этого доступа. С точки зрения компонентных технологий модули обработки выступают в качестве серверов, а ядро является клиентом.

Таким образом, абстрактная структура системы представляется довольно лаконичной. Система представляет собой сеть, в которой основным видом сервиса является доступ к данным любого вида (в том числе реального времени). Этот сервис реализуется системой взаимосвязанных серверов через клиентов-посредников. Такая идеология хорошо согласуется с возможностями COM/DCOM-технологии программирования.

Исходя из вышесказанного, можно определить набор функций, который должен реализовывать каждый тип модуля для выполнения возложенных на него задач.

1) Ядро должно реализовывать следующие функции:

   - загрузка данных: при выполнении этой функции должны загружаться данные из файла конфигуратора, создаваться ссылки на нужные модули и т.п., то есть должна происходить инициализация системы;
   - старт: эта функция должна запускать работу системы;
   - стоп: остановка системы, освобождение ресурсов, выгрузка модулей и т.п.

2) Модули обработки должны реализовывать те же функции, что и ядро:

   - загрузка данных: загрузка специфических для данного модуля данных;
   - старт: после вызова этой функции модуль должен быть готов к работе, т.е. к приему, передаче и обработке сообщений;
   - стоп: освобождение ресурсов, готовность к выгрузке.

   Дополнительные функции модулей обработки:

   - получение данных: эта функция должна получать данные в виде набора сообщений и обрабатывать их;
   - посылка данных: при вызове этой функции модуль должен передавать данные клиенту, также в виде набора сообщений.

3) Модули связи должны реализовывать все те же функции, что и модули обработки. Единственное отличие состоит в том, что при получении данных от ядра, кроме набора сообщений, модуль связи должен получать также и адреса узлов, на которые нужно направить соответствующее сообщение.

Основная функция системы — это обеспечение необходимого обмена данными между модулями. Каждый модуль в общем случае может являться независимым приложением и поддерживать некоторый набор протоколов обмена. Взаимодействие между модулями может происходить как локально на одном узле, так и удаленно через некоторую, возможно разнородную, сеть.

### 3.2.2. Локальное взаимодействие модулей

Локальное взаимодействие происходит, когда оба модуля находятся на одном и том же узле (рис. 3).

Например, от модуля A необходимо передать некоторые данные в модуль B. Для этого модуль A формирует сообщение с этими данными и передаёт его в ядро. Затем ядро определяет адресата сообщения и если он расположен локально, то сообщение напрямую ядром отсылается модулю B. Несмотря на то, что схема локального взаимодействия предельно проста, возможны несколько вариантов ее организации в зависимости от того, какую конкретно систему необходимо построить. Рассмотрим два основных варианта организации локального взаимодействия модулей.

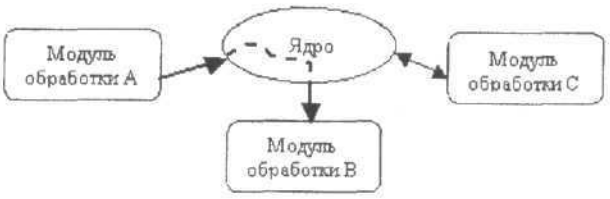


Рис. 3.

1) Если система строится лишь как среда для обеспечения обмена данными между независимыми приложениями, то имеет смысл адресат и источник содержать внутри сообщения. Схема локального взаимодействия при этом выглядит так: модуль А формирует сообщение и отдаёт его ядру ( при этом модуль А должен обладать всей необходимой информацией о модуле В, т.е. иметь его адрес в системе и тип протокола); ядро разбирает сообщение и, основываясь на адресате, передаёт его модулю В. Роль ядра минимальна и сводится к функции управления приоритетами. Такая система является наиболее гибкой, поскольку приложения общаются практически напрямую. Но когда система строится на основе некоторого конфигуратора, в котором жестко описаны все связи между приложениями, такой подход неудобен, поскольку информацию конфигуратора приходится распределять между модулями. Во-первых, это трудно реализуемо, а во-вторых, это может перегружать функции модулей, поскольку в системах с большим количеством модулей данных (возможно, от разных разработчиков), невозможно проконтролировать правильность маршрутизации сообщения модулем, а ошибка здесь может привести к неправильной работе всей системы.

2) Второй вариант предполагает, что строится система со строго определёнными типами приложений и связей между ними. В этом случае удобно равномерно распределить функциональную нагрузку между модулями, а информацию конфигуратора хранить централизованно в ядре. Схема локального взаимодействия в этом случае такова: модуль А формирует сообщение в формате «источник-данные» и передаёт его ядру; ядро получает сообщение, определяет его адресата, например, при помощи таблицы маршрутизации, созданной на основе файла конфигуратора, и передаёт его на обработку модулю В в виде «адресат-данные».

Для систем автоматизации типа SCADA наиболее подходит второй вариант реализации локального взаимодействия.

### 3.2.3. Удалённое взаимодействие между модулями

Общая схема удалённого взаимодействия между двумя узлами напрямую показана на рис. 4.

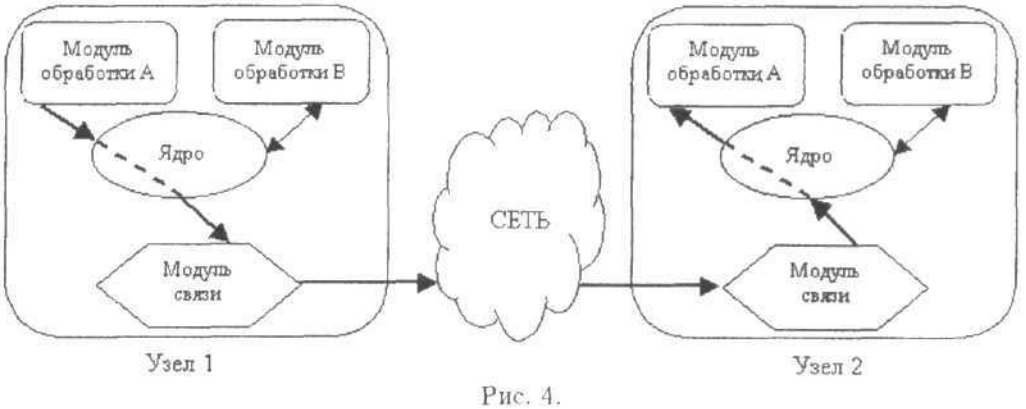


Рис. 4.

На узле 1 модуль А формирует сообщение и передаёт его ядру. Ядро определяет, что сообщение предназначено для модуля другого узла системы и передаёт его модулю связи. Модуль связи устанавливает соединение с аналогичным модулем связи на узле 2 и пересылает ему сообщение. Модуль связи узла 2 передаёт сообщение ядру, а ядро — адресату, модулю А узла 2.

Более сложный случай возникает, когда нельзя напрямую установить связь и сообщение должно пройти несколько узлов, прежде чем оно достигнет адресата (рис. 5).

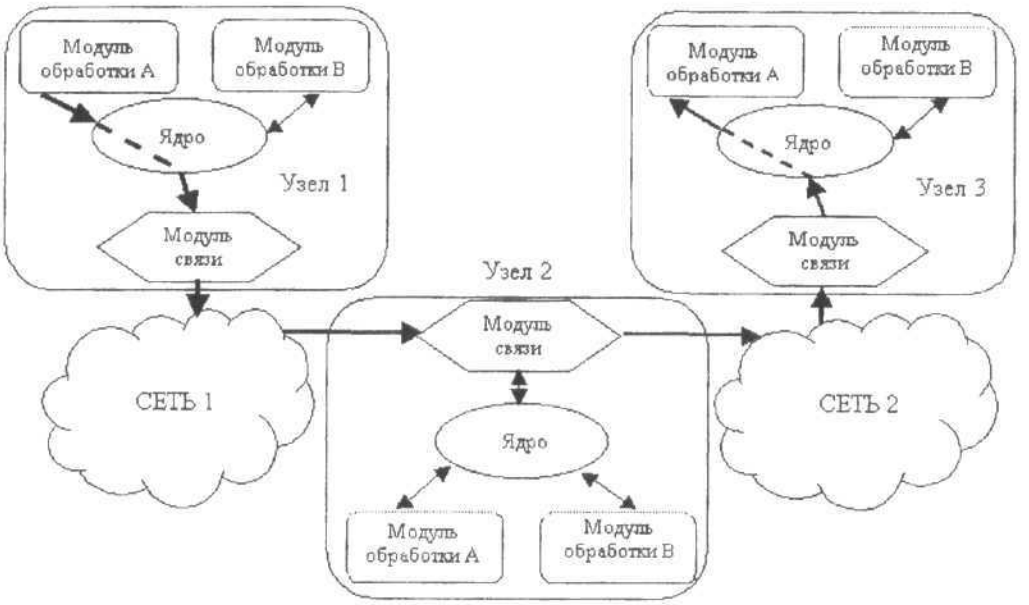


Рис. 5.

Пусть сообщение модуля А узла 1 должно быть передано модулю А узла 3. Для этого оно должно сначала пройти через сеть 1 и попасть в модуль связи узла 2, а затем через сеть 2, которая может отличаться от сети 1 как физически, так и логически, чтобы попасть в модуль связи узла 3. Главная проблема здесь возникает при организации маршрута следования сообщений, поскольку возможно несколько путей от узла 1 к узлу 3, проходящих через различные сети. Рассматривалось несколько решений, но наиболее интересным представляется то, при котором задача прокладки маршрута возлагается на модули связи соответствующих узлов.

Для примера, изображенного на рис. 5, это может происходить следующим образом. Ядро узла 1 передаёт сообщение в свой модуль связи. Модуль связи, исходя из информации о прямых связях между узлами, описанных в конфигураторе, определяет кратчайший маршрут и узел на этом маршруте, которому надо передать сообщение, – это будет узел 2. Модуль связи узла 2 проделывает ту же процедуру и передаёт сообщение модулю связи узла 3. Преимущества такого подхода заключаются в следующем:

- нет необходимости описывать в конфигураторе все возможные пути сообщений, которых может быть достаточно много;
- модули связи могут оптимизировать маршруты;
- можно легко обеспечить подключение к любому узлу системы в гетерогенной сети (для этого существует стек протоколов TCP/IP, но в некоторых случаях использование его может быть невозможно или неэффективно).

Следует заметить, что при описании системы формально или неформально употреблялись термины, которые используются для описания сетей передачи данных (сообщение, маршрутизация, связь и т.д.). В связи с этим следует еще раз обратить внимание на то, что рассматриваемая система представляет собой программную сеть определенной структуры, наложенную на сеть передачи данных. В этом смысле можно считать, что, например, ядро или модуль связи выполняет функции маршрутизатора сети 2-го уровня, т.е. маршрутизатора не сети передачи данных, а сети программных приложений, что и приводит к необходимости принятия определенных альтернативных решений на узлах при функционировании системы.

## 3.3. Средства реализации дополнительных возможностей в системе

Ниже в качестве демонстрации эффективности использования описанных принципов рассматривается реализация некоторых дополнительных возможностей СПА, которые в настоящее время широко распространены при построении SCADA-систем.

### 3.3.1. Поддержка стандартов OPC

Стандарт OPC (OLE for Process Control) разрабатывается с целью предоставить стандартный механизм для связи с различными устройствами, которые применяются в СПА. OPC базируется на технологии COM и представляет собой набор интерфейсов, которые должны поддерживаться OPC сервером. Сейчас большинство производителей промышленных контроллеров разрабатывают для своих устройств OPC серверы, которые представляют собой программные модули, осуществляющие связь с устройством и предоставляющие данные клиенту по определенным интерфейсам (рис. 6).

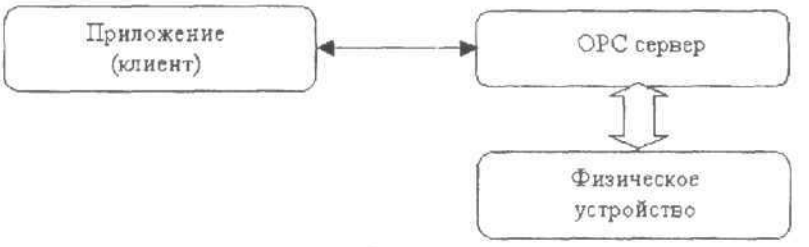


Рис. 6.

При использовании предлагаемых принципов построения систем OPC-серверы могут быть интегрированы в структуру системы, по крайней мере, двумя способами.

1) Как модули обработки (рис. 7). При таком подходе данные с OPC серверов поступают непосредственно в ядро, которое должно будет затем передавать их в другие модули. При этом ядро должно работать не только с интерфейсами системы, но также с OPC-интерфейсами, осуществляя перевод соответствующих данных в формат системы.


Рис. 7.

2) Как источники данных (рис. 8). В этом случае для таких источников необходимо разработать модуль обработки, который выступает как OPC-клиент. Этот модуль должен будет получать данные от одного или нескольких OPC-серверов по OPC-интерфейсам, а затем передавать их в ядро по интерфейсам системы. Такой подход представляется более рациональным по следующим причинам. Во-первых, если в проекте нет необходимости в использовании OPC-серверов, то заказчик может не приобретать этот модуль. Во-вторых, это лучше согласуется с принципами компонентного проектирования, т.к. каждый модуль предоставляет четко определенный сервис.

### 3.3.2. Информационное взаимодействие через Internet

В данном случае под взаимодействием (удаленным доступом) через Internet понимается возможность обращаться средствами Internet к информационным сервисам, предоставляемым системой автоматизации. Такая возможность позволяет удаленным пользователям (Web-клиентам) получать данные о ходе объектовых процессов (текущие значения контролируемых параметров и результатов их обработки), используя стандартное клиентское программное обеспечение, например, такое, как Internet Explorer или Netscape Navigator.

Очевидно, что для ответа на запросы Web-клиентов в систему автоматизации должен быть интегрирован Web-сервер. На сервере должно функционировать приложение, взаимодействующее с системой и с сервером, поддерживающее стандарт CGI или ISAPI/NSAPI. Кроме того, в системе должен быть

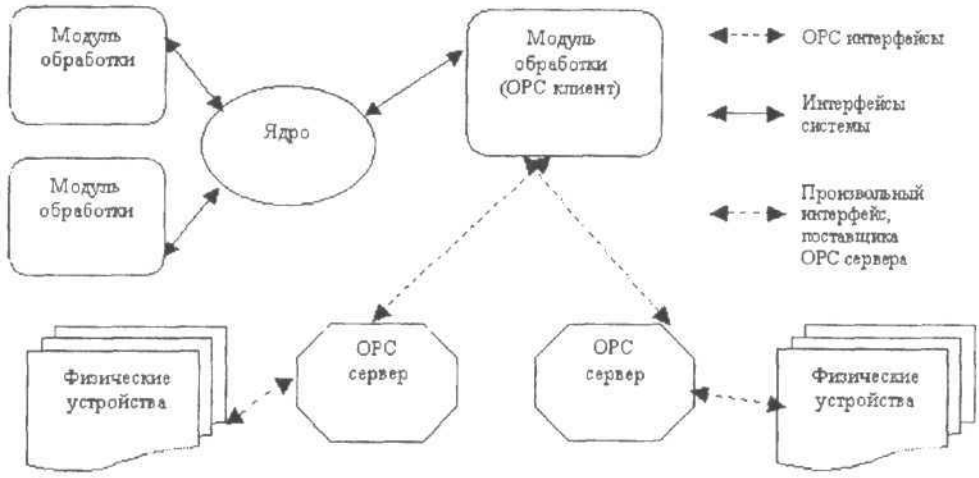


Рис. 8.

установлен модуль обработки (ниже он назван Internet-модулем), от которого это приложение будет получать данные. В Internet-модуль от ядра системы должны поступать все данные, которые планируется публиковать в Internet.

Таким образом, когда Web-клиент обращается к Web-серверу с запросом, тот переадресует этот запрос работающему на сервере приложению. Приложение связывается с Internet-модулем и получает от него запрошенные данные. На основе этих данных формируется HTML-страница, которая возвращается Web-серверу в качестве ответа на запрос, а Web-сервер пересылает ее клиенту, запросившему данные (рис. 9). Таким образом, наблюдение за ходом процесса может осуществляться через Internet без установки на компьютере удаленного Web-клиента какого-либо дополнительного программного обеспечения.

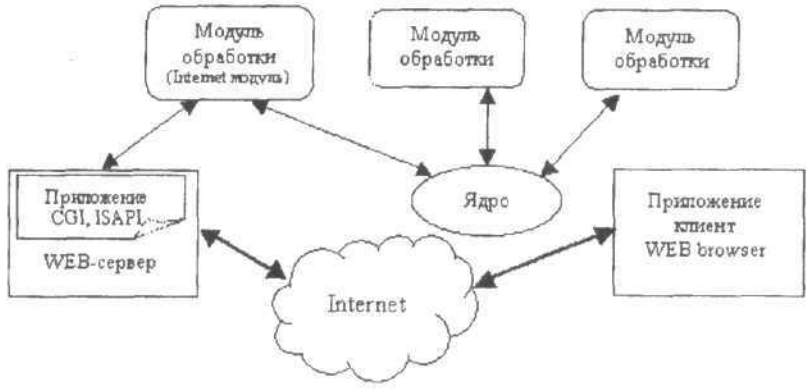


Рис. 9.

## 4. Заключение

В работе проведен анализ состояния и разнообразия информационных технологий и средств, используемых при построении современных СПА. Одновременно с проведением этого анализа рассмотрены некоторые особенности, определяемые различными факторами, которые затрагивают основные аспекты построения систем рассматриваемого класса и фактически характеризуют общие требования к функционально-информационной структуре СПА.

Описаны принципы построения функциональной и информационной структуры распределенных СПА, которые ориентированы на использование методов и средств компонентных программных технологий. В общем виде структура распределенной системы промышленной автоматизации представляется как совокупность устройств (узлов), взаимодействующих через коммуникационные сети и обменивающихся информацией в режиме реального времени под управлением программных средств, функционирующих в соответствии с конкретным проектом.

Предлагаемое в работе описание информационно-функциональной структуры СПА сочетает элементы объектно-ориентированного проектирования и компонентных технологий и рассматривается как основа для одного из системных решений соответствующей задачи, которое позволяет в достаточной степени учитывать влияние основных факторов, определяющих особенности построения СПА.

## Литература

[1] Любашин А.Н. Промышленные сети // Мир компьютерной автоматизации. 1999. N1. С. 38-44.

[2] Карпенко Е. Возможности CAN-протокола // Современные технологии автоматизации. 1998. N4. С. 16-20.

[3] Жданов А., Латыев А. Выбор ОС при построении систем реального времени. // PC WEEK. 2001. N1. С. 22-23.

[4] Жданов А. Операционные системы реального времени // PC WEEK. 1999. N8. С. 17-18,24.

[5] Прожогин С. Neutrino: быстрее, выше и ... меньше // Современные технологии автоматизации. 2000. N1. С. 32-37.

[6] Калядин А. DCX – управление и обмен данными в реальном времени в Windows // PC WEEK. 2000. N21. С. 19-20.

[7] Куцевич Н.А. SCADA-системы и муки выбора // Мир компьютерной автоматизации. 1999. N1. С. 72-78.

[8] Бунин В. и др. SCADA-системы: проблемы выбора // Современные технологии автоматизации. 1999. N4. С. 6-23.

[9] Куцевич Н.А Компонентные технологии в системах промышленной автоматизации // Открытые системы. 1999. N4. С. 31-36.

[10] Воас Д. Сопровождение компонентных систем // Открытые системы. 1998. N6. С. 17-21.

[11] Перегудов Ф.И., Тарасенко Ф.П. Введение в системный анализ. – М.: Высш.шк., 1989. 367 с.

[12] Буч Г. Объектно-ориентированный анализ и проектирование (с примерами приложений на C++) М.: Издательство Бином, СПб: Невский диалект, 1998. 560 с.

*Модел. и анализ информ. систем.* Т.8, №1 (2001) 38–43

УДК 681.3

# Верификация графов потоков данных с использованием символьной проверки модели для CTL

Кузьмин Е.В.
*Ярославский государственный университет*
*150 000, Ярославль, Советская, 14*

Работа посвящена верификации программ потоков данных на основе их абстрактной модели. В качестве метода верификации рассматривается символьная проверка модели для логики CTL.

## 1. Введение

В данной работе рассматривается задача верификации графов потоков данных, представляющих собой абстрактную модель программ потоков данных. Для верификации используется метод символьной проверки модели (symbolic model checking – *MC*), в котором требуемые свойства поведения системы записываются на языке темпоральной логики CTL, а само поведение системы задаётся конечным автоматом или системой переходов. Таким образом, задача верификации сводится к проверке выполнимости некоторой логической формулы для некоторой системы переходов, т.е. к проверке является ли система переходов моделью логической формулы.

Процесс верификации строится на основе символьного подхода к решению задачи *MC* для формул $\mu$-исчисления. Формулы темпоральной логики CTL адекватно задаются формулами $\mu$-исчисления, что позволяет свести решение задачи проверки выполнимости формулы CTL к оценке формулы $\mu$-исчисления на той же модели. Использование $\mu$-исчисления даёт возможность применять для задания логических и автоматных конструкций символьные структуры – ориентированные бинарные диаграммы решений (OBDD), являющиеся достаточно экономным средством символьного представления булевых функций и формул.

## 2. Графы потоков данных

Граф потоков данных $PR = (S, \Delta, s^0)$ представляет собой *модель программы потоков данных* [1], где

- $S$ – множество состояний программы (конфигураций графа), $S = (I \times G_i) \times (U \times G_u)$, $I \subset P$, $U \subset P$, $I \cap U = 0$, $I \cup U = P$, $G_i = \{0, 1\}$, $G_u = \{0, 11, 10\}$, $I$ – множество информационных портов, $U$ – множество управляющих портов;
- $\Delta : S \to 2^S$, определяющая множество возможных состояний, в которые программа может перейти из текущего состояния;
- $s^0$ – начальное состояние, $s^0 \in S$.

Поведение программы определяется функцией переходов $\Delta$. В соответствии с *асинхронной концепцией поведения*, если в некоторый момент существует несколько допустимых переходов, то только один (любой из них) срабатывает в следующий момент.

Допустимая последовательность состояний программы (конфигураций графа потоков данных), определяемая функцией переходов $\Delta$, называется *процессом*. В общем случае процесс может быть бесконечным. Множество всех процессов из некоторого начального состояния называется *поведением* программы (графа потоков данных).

## 3. Синтаксис и семантика CTL

Темпоральная логика CTL (computation tree logic) [2] является достаточно выразительным средством для описания поведения систем параллельных вычислений и позволяет учитывать наиболее важные свойства

– справедливость, безопасность, живость – обеспечивающие корректность и надёжность их функционирования. Логика CTL определяется на темпоральных структурах Крипке.

**Определение.** *Темпоральной структурой Крипке* называется система $(AP, W, R, w^0, L)$, где AP – множество элементарных высказываний, W – непустое множество состояний, $R \subseteq W \times W$ – тотальное отношение непосредственной достижимости между состояниями (т.е. для каждого $w \in W$ существует $w' \in W$ такой, что $(w, w') \in R$), $w^0 \in W$ – начальное состояние, и $L : W \to 2^{AP}$

Путем из состояния $w_0$ в структуре Крипке называется бесконечная последовательность состояний $\pi = w_0, w_1, \ldots$ таких, что для любого $i \geq 0$ выполняется $(w_i, w_{i+1}) \in R$.

Введём отношение выполнимости формулы CTL на заданной структуре Крипке. Для состояния $w$ структуры Крипке $K$ и формулы CTL $\varphi$ запись $K, w \models \varphi$ означает, что $\varphi$ истинна в состоянии $w$ структуры $K$. Для данного состояния $w$ структуры $K$ и формулы $\varphi$ отношение $\models$ индуктивно определяется следующим образом.

- $K, w \models p$ для $p \in AP \iff p \in L(w)$; $K, w \models \mathbf{true}$ и $K, w \not\models \mathbf{false}$ для любого $w \in W$;
- $K, w \models \neg\varphi \iff K, w \not\models \varphi$;
- $K, w \models \varphi \wedge \psi \iff K, w \models \varphi$ и $K, w \models \psi$;
- $K, w \models \varphi \vee \psi \iff K, w \models \varphi$ или $K, w \models \psi$;
- $K, w \models AX\varphi \iff K, w' \models \varphi$ для всех состояний $w'$: $(w, w') \in R$;
- $K, w \models EX\varphi \iff K, w' \models \varphi$ для некоторого состояния $w'$: $(w, w') \in R$;
- $K, w \models A(\varphi U \psi) \iff$ для каждого пути $w_0, w_1, \ldots$, выходящего из состояния $w_0 = w$, найдётся $j \geq 0$, такое что $K, w_j \models \psi$ и при этом для всех $i$, $0 \leq i < j$, $K, w_i \models \varphi$;
- $K, w \models E(\varphi U \psi) \iff$ для некоторого пути $w_0, w_1, \ldots$, выходящего из состояния $w_0 = w$, найдётся $j \geq 0$, такое что $K, w_j \models \psi$ и при этом для всех $i$, $0 \leq i < j$, $K, w_i \models \varphi$;
- $K, w \models A(\varphi V \psi) \iff$ для каждого пути $w_0, w_1, \ldots$, выходящего из состояния $w_0 = w$, и любого состояния $w_j$, $j \geq 0$, для которого $K, w_j \models \neg\psi$, найдётся состояние $w_i$, $0 \leq i < j$, что $K, w_i \models \varphi$;
- $K, w \models E(\varphi V \psi) \iff$ для некоторого пути $w_0, w_1, \ldots$, выходящего из состояния $w_0 = w$, и любого состояния $w_j$, $j \geq 0$, для которого $K, w_j \models \neg\psi$, найдётся состояние $w_i$, $0 \leq i < j$, что $K, w_i \models \varphi$;

Помимо введённых выше логических связок и темпоральных операторов для удобства употребляются следующие традиционные логические связки и операторы: $\varphi \to \psi = \neg\varphi \vee \psi$, $F\varphi = \mathbf{true} U \varphi$ ("Eventually" – когда-нибудь, в конечном счёте), $G\varphi = \mathbf{false} V \varphi$ ("Always" – всегда).

Формула логики CTL $\varphi$ считается выполнимой (или истинной) на структуре $K = (AP, W, R, w^0, L)$, если $K, w^0 \models \varphi$. При этом структура $K$ называется моделью формулы $\varphi$. Задача проверки выполнимости формулы логики CTL на модели заключается в распознавании выполнимости заданной формулы $\varphi$ на заданной конечной модели $K$.

## 4. Пропозициональное $\mu$-исчисление

*Построение формул $\mu$-исчисления производится из элементарных высказываний* $AP = \{p_1, p_2, \ldots\}$, пропозициональных переменных $VAR = \{R_1, R_2, \ldots\}$, логических связок $\neg, \wedge, \vee$, модальных операторов $\langle a \rangle$ и $[a]$, где $a$ – действие из множества $Act = \{a_1, a_2, \ldots\}$, операторов наибольшей и наименьшей неподвижных точек $\mu R_i.(\ldots)$ и $\nu R_i.(\ldots)$, где пропозициональная переменная $R_i$ в выражения, ограниченные операторами неподвижных точек, должна входить только позитивно. Вхождение переменной $R$ в формулу $\varphi$ называется *позитивным*, если $R$ находится в области действия четного числа отрицаний.

Формулы $\mu$-исчисления определяются относительно системы переходов $M = (\mathsf{T}, T, L)$, где $\mathsf{T}$– непустое конечное множество состояний, $L : AP \to 2^{\mathsf{T}}$ – отображение каждого элементарного высказывания на некоторое подмножество состояний, для которых это утверждение истинно, $T : Act \to 2^{\mathsf{T} \times \mathsf{T}}$– отображение задающее систему перехода между состояниями по некоторому действию.

Формально формула $\varphi$ интерпретируется как множество состояний, в которых $\varphi$ является истинной. Обозначим это множество состояний как $[\![\varphi]\!]_M e$, где $M$ – это система переходов, а $e : VAR \to 2^{\mathsf{T}}$ – функция, сопоставляющая пропозициональной переменной некоторое подмножество состояний. Обозначим за

$e[R \leftarrow S]$ новую функцию, которая будет такой же, что и $e$ за исключением того, что $e[R \leftarrow S](R) = S$. Множество $[\![\varphi]\!]_M e$ определяется рекурсивно следующим образом.

- $[\![p]\!]_M e = L(p)$; $[\![R]\!]_M e = e(R)$;
- $[\![\neg\varphi]\!]_M e = \top - [\![\varphi]\!]_M e$;
- $[\![\varphi \wedge \psi]\!]_M e = [\![\varphi]\!]_M e \cap [\![\psi]\!]_M e$;
- $[\![\varphi \vee \psi]\!]_M e = [\![\varphi]\!]_M e \cup [\![\psi]\!]_M e$;
- $[\![\langle a\rangle\varphi]\!]_M e = \{s \mid \exists t\ [s \xrightarrow{a} t \text{ и } t \in [\![\varphi]\!]_M e]\}$;
- $[\![[a]\varphi]\!]_M e = \{s \mid \forall t\ [s \xrightarrow{a} t \text{ следует } t \in [\![\varphi]\!]_M e]\}$;
- $[\![\mu R.\varphi]\!]_M e$ – это наименьшая неподвижная точка функции $\tau : 2^\top \to 2^\top$: $\tau(S) = [\![\varphi]\!]_M e[R \leftarrow S]$;
- интерпретация $\nu R.\varphi$ аналогична, только берется наибольшая неподвижная точка.

**Лемма 1.** [3] *Если $R$ входит в $\varphi$ позитивно и $\varphi$ не содержит конструкций неподвижных точек, то функция $\tau(S)$ является монотонной относительно включения множеств, т.е. из $S_1 \subseteq S_2$ следует, что $\tau(S_1) \subseteq \tau(S_2)$.*

**Утверждение 1.** [3] *Пусть на конечном множестве $A$ задан частичный порядок $\succ$ и существуют наименьший $\bot$ и наибольший $\top$ элементы. Пусть функция $f : A \to A$ монотонно неубывающая, т.е. из $x \succ y$ следует $f(x) \succ f(y)$. Тогда существуют наименьшая $\mu x.f(x)$ и наибольшая $\nu x.f(x)$ неподвижные точки $f$, и $\mu x.f(x) = f^n(\bot)$, $\nu x.f(x) = f^n(\top)$ для некоторого $n \leq |A|$.*

Так как функция $\tau(S)$ является монотонной на множестве $2^\top$ по включению $\subseteq$, то данное утверждение применимо для этой функции. Таким образом, получаем итеративные алгоритмы нахождения наименьшей и наибольшей неподвижных точек ($\tau^i(S)$ вычисляется как $\tau^0(S) = S$ и $\tau^{i+1}(S) = \tau(\tau^i(S))$):

$$[\![\mu R.\varphi]\!]_M e = \bigcup_i \tau^i(\bot) \qquad \text{и} \qquad [\![\nu R.\varphi]\!]_M e = \bigcap_i \tau^i(\top).$$

## 5. Перевод CTL в $\mu$-исчисление

Алгоритм $Tr$ имеет на входе формулу CTL, а результатом работы этого алгоритма является эквивалентная формула $\mu$-исчисления с одним действием $a$.

- $Tr(p) = p$;
- $Tr(\neg\varphi) = \neg Tr(\varphi)$;
- $Tr(\varphi \wedge \psi) = Tr(\varphi) \wedge Tr(\psi)$;
- $Tr(\varphi \vee \psi) = Tr(\varphi) \vee Tr(\psi)$;
- $Tr(EX\varphi) = \langle a\rangle Tr(\varphi)$;
- $Tr(AX\varphi) = [a]Tr(\varphi)$;
- $Tr(E(\varphi U \psi)) = \mu Z.(Tr(\psi) \vee (Tr(\varphi) \wedge \langle a\rangle Z))$;
- $Tr(A(\varphi U \psi)) = \mu Z.(Tr(\psi) \vee (Tr(\varphi) \wedge [a] Z))$;
- $Tr(E(\varphi V \psi)) = \nu Z.(Tr(\psi) \wedge (Tr(\varphi) \vee \langle a\rangle Z))$;
- $Tr(A(\varphi V \psi)) = \nu Z.(Tr(\psi) \wedge (Tr(\varphi) \vee [a] Z))$;

**Теорема 1.** [3] *Пусть $K = (AP, W, R, w^0, L)$ – структура Крипке, $\varphi$ – формула CTL. Пусть интерпретацией действия $a$ будет $R$. Элементарное высказывание $p$ в $Tr(\varphi)$ интерпретируется как $L(p)$. Множество состояний $\top$ – это $W$. Тогда для всех $w \in W$: $w \models \varphi \Leftrightarrow w \in [\![Tr(\varphi)]\!]_M e$.*

Дальше предполагается рассмотрение только лишь тех формул $\mu$-исчисления, которые соответствуют формулам CTL.

## 6. Перевод $\mu$-исчисления в OBDD

Рассмотрим представление системы переходов $M = (\top, T, L)$ в виде OBDD [4]. Множество состояний системы переходов $\top$ кодируется множеством значений $n$ булевых переменных $x_1, \ldots, x_n$, т.е. $\top$ - множество 0-1 векторов длины $n$.

Будем строить диаграммы OBDD, соответствующие формулам $\mu$-исчисления следующим образом.

- Каждое элементарное высказывание $p$ имеет ассоциированную с ним диаграмму OBDD. Будем обозначать такую диаграмму OBDD как $OBDD_p(\vec{x})$. Некоторый вектор $\vec{x} \in \{0,1\}^n$ удовлетворяет $OBDD_p$, если $\vec{x} \in L(p)$.

- Каждый символ действия $a$ имеет ассоциированную с ним диаграмму $OBDD_a(\vec{x}, \vec{x}')$. 0-1 вектор $(\vec{y}, \vec{z}) \in \{0,1\}^{2n}$ удовлетворяет $OBDD_a$, если $(\vec{y}, \vec{z}) \in T(a)$.

Обозначим за $A\langle R \leftarrow B_R\rangle$ ассоциацию пропозициональной переменной $R$ с OBDD диаграммой $B_R$. Приведенная ниже программная функция $B$ имеет на входе формулу $\mu$-исчисления $\varphi$, и результатом ее работы является диаграмма OBDD, соответствующая этой формуле.

- $B(p, A) = OBDD_p(\vec{x})$;
- $B(R, A) = A[R]$;
- $B(\neg\varphi, A) = \neg B(\varphi, A)$;
- $B(\varphi \wedge \psi, A) = B(\varphi, A) \wedge B(\psi, A)$;
- $B(\varphi \vee \psi, A) = B(\varphi, A) \vee B(\psi, A)$;
- $B(\langle a\rangle\varphi, A) = \exists\vec{x}'(OBDD_a(\vec{x}, \vec{x}') \wedge B(\varphi, A)(\vec{x}'))$;
- $B([a]\varphi, A) = B(\neg\langle a\rangle\neg\varphi, A)$;
- $B(\mu R.\varphi, A) = FIX(\varphi, A, FALSE\text{-}BDD)$;
- $B(\nu R.\varphi, A) = FIX(\varphi, A, TRUE\text{-}BDD)$;

Диаграммы OBDD для булевых функций **false** и **true** обозначаются *FALSE-BDD* и *TRUE-BDD* соответственно.

**function** $FIX(\varphi, A, B_R)$
**begin**
  result-bdd := $B_R$;
  **do**
    old-bdd := result-bdd;
    result-bdd := $B(\varphi, A\langle R \leftarrow$ old-bdd$\rangle)$;
  **while**(not-equal(old-bdd, result-bdd));
  **return**(result-bdd);
**end**;

Нужно отметить, что диаграммы для операторов неподвижных точек строятся всего один раз, и в случае вложенности одного оператора в другой не пересчитываются, т.к. вложенный оператор неподвижной точки не будет содержать связанной переменной внешнего оператора (по допущению, сделанному выше).

# 7. Символьная верификация графов потоков данных

Выбор метода Symbolic Model Checking для верификации графов потоков данных был обусловлен следующим. Графы потоков данных представляют собой параллельную систему с *конечным* числом состояний. При верификации не требуется дополнительного ручного конвертирования графов потоков данных в структуру Крипке. Состояния графа потоков данных легко и естественно представляются в виде бинарных векторов фиксированной длины, что позволяет эффективно применять символьную структуру – ориентированную двоичную диаграмму решений (OBDD) – для задания множества состояний, удовлетворяющих некоторой булевой функции.

Для реализации символьного метода поверки модели (для графов потоков данных) необходимо:

- построить символьную структуру для представления графа (дерева) достижимых конфигураций (т.е. построить систему переходов по состояниям графа потоков данных);
- некоторым образом задавать элементарные высказывания.

## 7.1. Граф достижимости

Как было показано выше, для применения символьного метода проверки модели необходимо представить множество достижимых конфигураций графов потоков данных набором двоичных векторов $\vec{x} \in \{0,1\}^n$, а затем построить $OBDD(\vec{x}, \vec{x}')$ для задания системы переходов по этим конфигурациям (состояниям).

Конфигурация абстрактной модели графа потоков данных, по сути, является вектором состояний портов, которые за исключением управляющих портов содержат либо 0, либо 1. Поэтому каждому информационному порту можно сопоставить двоичную переменную. А для каждого управляющего порта, который может принимать три значения 0, 11, 10, сопоставим две двоичные переменные.

Построение $OBDD(\vec{x}, \vec{x}')$ предлагается реализовывать на стадии формирования множества достижимых конфигураций. Процесс накопления множества достижимых конфигураций из начальной конфигурации заключается в том, что на каждом шаге определяется множество непосредственных последователей

текущего множества конфигураций графов потоков данных. Процесс останавливается, если не будет перехода в некоторую новую конфигурацию.

Рассмотрим булеву функцию $T(\vec{x}, \vec{x}')$, которая будет истинной, если существует непосредственный переход от одного состояния графа потоков данных $\vec{x}$ в другое $\vec{x}'$.

Будем считать, что функция $T(\vec{x}, \vec{x}')$ представлена в совершенной дизъюнктивной нормальной форме. Если непосредственный переход из одного состояния в другое существует, тогда будет истинна только одна конъюнкция $x_1^\sigma \wedge \ldots x_n^\sigma \wedge x'^\sigma_1 \wedge \ldots x'^\sigma_n$ ($\sigma$ – обозначает возможное отрицание), соответствующая этим состояниям. Построение диаграммы OBDD для такой конъюнкции осуществляется за время $O(n)$.

Таким образом, мы имеем возможность строить диаграмму $OBDD(\vec{x}, \vec{x}')$ для функции $T(\vec{x}, \vec{x}')$ динамически. Т.е. при появлении нового перехода между состояниями производится построение диаграммы OBDD для истинной в этих состояниях конъюнкции, и осуществляется дизъюнкция между этой диаграммой и диаграммой OBDD, соответствующей функции $T(\vec{x}, \vec{x}')$ на предыдущем шаге.

Трудоемкость процедуры построения $OBDD(\vec{x}, \vec{x}')$ для функции $T(\vec{x}, \vec{x}')$ ограничена $O(|OBDD(\vec{x}, \vec{x}')| \cdot n^2 \cdot k)$, где $k$ - мощность множества достижимых конфигураций графов потоков данных, а $n$ - длина вектора $\vec{x}$ , т.к. нужно будет произвести $k$ построений диаграмм достижимых конфигураций, и $k$ дизъюнкций между диаграммой для нового перехода и диаграммой, соответствующей $T(\vec{x}, \vec{x}')$ на предыдущем шаге.

Диаграмма $OBDD_{Rch}(\vec{x})$ для множества достижимых конфигураций строится аналогичным образом, за исключением того, что функция, распознающая достижимую конфигурацию, будет зависеть только от одного вектора $\vec{x}$.

## 7.2. Представление элементарных высказываний

Для представления элементарных высказываний $q \in AP$ предлагается использовать булевы формулы над переменными, задающими конфигурации графов потоков данных. Т.е. высказывание $q$ будет являться булевой функцией $q = q(p_1^t, p_2^t, \ldots, p_k^t) = q(x_1, x_2, \ldots, x_n)$, где $k$ – количество портов, $n$ - количество переменных, кодирующих конфигурации, а $t$ принимает значение либо $i$, если порт информационный, либо $u$, если порт управляющий; $p_j^i = p_j^i(x_j)$ – булева функция, соответствующая информационному порту $p_j$, заданная формулой от одной переменной $x_j$; $p_j^u = p_j^u(x_{j_1}, x_{j_2})$ – булева функция, соответствующая управляющему порту $p_j$, заданная формулой от двух переменных $x_{j_1}$ и $x_{j_2}$;

Таким образом, $OBDD_q(\vec{x})$ для множества конфигураций, для которых истинно элементарное высказывание $q$, можно построить на основе формулы, задающей функцию $q(\vec{x})$.

## 7.3. Символьная верификация графов потоков данных

Таким образом, в процессе верификации графов потоков данных можно выделить следующую последовательность шагов:

- Представление множества достижимых состояний и системы переходов по состояниям в символьном виде с помощью ориентированных бинарных диаграмм решений OBDD;
- Представление каждого элементарного высказывания в виде диаграммы OBDD;
- Представление верифицируемых свойств на языке логики CTL;
- Перевод формулы логики CTL в формулу $\mu$-исчисления;
- Оценка формулы $\mu$-исчисления, т.е. построение множества состояний, удовлетворяющих формуле;

Для графов потоков данных указанный выше алгоритм оценки формулы $\mu$-исчисления будет иметь изменения в следующих пунктах:

- $B(\neg\varphi, A) = \top \backslash B(\varphi, A)$, где $\top$ – множество всех возможных конфигураций графа потоков данных;
- $B(\langle a \rangle \varphi, A) = \exists \vec{x}' \in OBDD_{Rch}(\vec{x})\ (OBDD_a(\vec{x}, \vec{x}') \wedge B(\varphi, A)(\vec{x}'))$;

## 8. Заключение

Таким образом, в работе был рассмотрен процесс верификации графов потоков данных с использованием символьной проверки модели для логики CTL. Данный метод удобен для автоматической верификации, что позволило реализовать программный модуль верификации графов потоков данных для инструментальной системы построения и анализа программ потоков данных.

Автор выражает благодарность проф. Соколову В.А. за замечания и полезные советы.

## Литература

[1] Sokolov V.A., Roubtsova E.E., Roubtsov S.A. On technology of design and analysis of dataflow programs. Lecture Notes in Computer Science, N 1277, Springer-Verlag, $4^{th}$ International Conference "Parallel Computing Technologies 97" (PACT - 97). 1997. P.115-120.

[2] Emerson E.A. Temporal and Modal Logic. //Handbook of Theoretical Computer Science: Volume B, Formal Models and Semantics. North-Holland Pub. Co. MIT Press. 1990. P. 995–1072.

[3] Berezin S., Clarke E. M., Jha S., Marrero W. Model Checking Algorithms for the $\mu$-Calculus. CMU-CS-96-180, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1996.

[4] Anderson H. R. An Introduction to Binary Decision Diagrams. Lecture notes for 49285 Advanced Algorithms E97, Technical University of Denmark, 1997.

*Модел. и анализ информ. систем.* Т.**8**, №**1** (2001) 44–46

УДК 517.3

# Аттракторы распределенной модели реакции Белоусова

Харьков А. Е.
*Ярославский государственный университет*
*150 000, Ярославль, Советская, 14*

Выполнено численное исследование свойств автоволновых процессов одной системы типа реакция-диффузия, моделирующей динамику реакции Белоусова в единичном квадрате.

## 1. Постановка задачи

В соответствии с [1] предметом анализа является краевая задача

$$\dot{N}_1 = D_1 \Delta N_1 + r_1 \Big[1 + a(1 - N_3) - N_1\Big] N_1, \tag{1}$$

$$\dot{N}_2 = D_2 \Delta N_2 + r_2 \Big[N_1 - N_2\Big] N_2, \quad \dot{N}_3 = D_3 \Delta N_3 + r_3 \Big[N_2 - N_3\Big] N_3, \tag{2}$$

$$\left.\frac{\partial N_1}{\partial n}\right|_{\Gamma} = \left.\frac{\partial N_2}{\partial n}\right|_{\Gamma} = \left.\frac{\partial N_3}{\partial n}\right|_{\Gamma} = 0. \tag{3}$$

Здесь $N_1$ , $N_2$ , $N_3$ - концентрации соответственно бромата, церия и бромида; $r_1$ , $r_2$ ,$r_3$ - коэффициенты линейного роста; $D_1$ , $D_2$ , $D_3$ - коэффициенты диффузии; $a$ - коэффициент "давления" бромида на бромат; $\Gamma$ - граница единичного квадрата; $n$ - нормаль к границе.

Стартовыми для нас являются следующие значения параметров:

$$r_1 = 1, r_2 = 1, r_3 = 3, a = 40, D_1 = 0.001. D_2 = 0.005, D_3 = 0.01.$$

Отметим, что конкретное задание первых четырех параметров мало влияет на динамические свойства краевой задачи (1)-(3). Роль коэффициентов диффузии более существенна. Нами они выбраны так, чтобы сохранялась устойчивость однородного цикла, причем пространственно неоднородные колебания также имеют место. Последний феномен наиболее ярко проявляется при уменьшении коэффициентов диффузии. Ниже они уменьшаются пропорционально.

Для численного анализа краевой задачи (1)-(3) используем разностную схему с "экспонентой", предложенную в [2].

Покроем единичный квадрат равномерной сеткой с шагом $h$. Значение концентрации $N_1, N_2, N_3$ в узлах обозначим $N_{lkm}, l \in \{1, 2, 3\}$. Рассматриваемая разностная схема для $N_1$ будет иметь следующий вид:

$$N_{1km}(t+\tau) = [N_{1km}(t) + \tau D_1 \Delta_u N_{1km}(t)] \times \exp\Big[\tau r_1 \Big(1 + a(1 - N_{3km}(t)) - N_{1km}(t)\Big)\Big].$$

$$(k, m = 1, ..., M-1, \quad t = 0, \tau, 2\tau, ...);$$

$$N_{l0k} = N_{l1k};\ N_{lMk} = N_{l(M-1)k};\ N_{lm0} = N_{lm1};\ N_{lmM} = N_{lm(M-1)},\ (k = 0, ..., M; m = 0, ..., M; l = 1, 2, 3),$$

где $M = 1/h$, $\Delta_u$ - обычная разностная аппроксимация оператора Лапласа, $\tau$ - шаг по времени. Считаем, что $\tau = 0.001$, $h = 0.02$. Разностные схемы для $N_2, N_3$ строятся аналогично.

На приводимых в дальнейшем рисунках в целях наглядности (полная пространственная картина сложным образом меняется с течением времени) показываются пространственные распределения концентрации $N_2$, причем черным (белым) отмечаются области, где $N_2 > 2 (N_2 < 2)$.

## 2. Результаты численного исследования

При выбранных параметрах системы область притяжения однородного цикла состоит из достаточно гладких функций. Однако имеется одна интересная характерная особенность - долго длится время установления.

Если же начальное распределение сосредоточено в какой-то части квадрата, то возникает пульсирующий режим. На рис. 1 на трех разделенных во времени сериях показана динамика пространственного

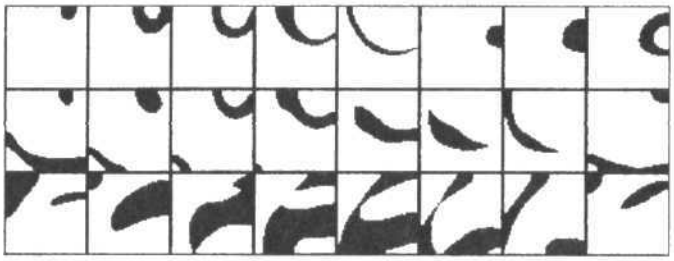

Рис. 1

изменения $N_2$. На каждой серии (например, верхний ряд квадратов) интервал между срезами равен 0.2. Интервал между сериями равен 20.

При уменьшении на порядок коэффициентов диффузии однородный цикл теряет устойчивость и происходит пространственное усложнение установившихся режимов, что хорошо показано на рис. 2, где, как и на рис. 1, приведены три разделенных во времени серии с интервалом 20 между ними. При этом промежуток времени между срезами на каждой серии равен 0.2.

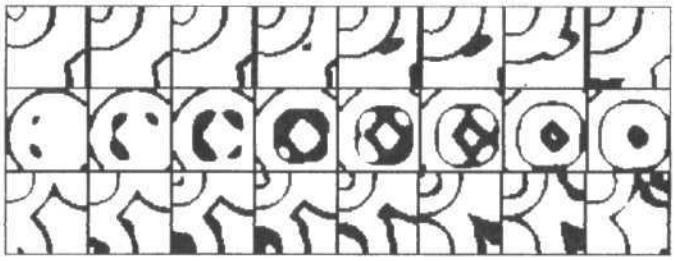

Рис. 2 $D_1 = 0.0001, \quad D_2 = 0.0005, \quad D_1 = 0.001.$

При еще одном уменьшении на порядок коэффициентов диффузии наступает так называемый диффузионный хаос. На рис. 3 приведены характерные "картинки", достаточно произвольно выбранные из временного интервала.

Рис. 3 $D_1 = 0.00001, \quad D_2 = 0.00005, \quad D_1 = 0.0001.$

На наглядном уровне диффузионный хаос проявляется следующим образом. Поделим квадрат на несколько мелких частей, в каждой из них колебания достаточно интенсивны, однако они сильно рассогласованы по фазе. Тем не менее усредненные по пространственным переменным колебания

$$M_k = \int_0^1 \int_0^1 N_k(t, x, y) dx dy, \quad k = 1, 2, 3,$$

заметно менее интенсивны, чем те колебания, которые доставляет однородный цикл. Это видно из показанного на рис. 4, где в левой части этого рисунка приведен график однородных колебаний, а в правой - усредненные колебания при указанных на рис. 3 значениях коэффициентов диффузии.

## 3. Заключение

Хотя явление диффузионного хаоса изучалось численными методами многими авторами, в нашей задаче он обладает специфическими чертами, и все же главным результатом статьи считаем факт наличия пульсирующих режимов, сосуществующих с устойчивым однородным циклом. Отметим также, что установленные факты хорошо согласуются с известными экспериментальными результатами. Вероятно, следует отметить, что с помощью математической модели впервые дано теоритическое объяснение феномену возникновения автоволновых процессов в ставшей уже давно классической реакции Белоусова.

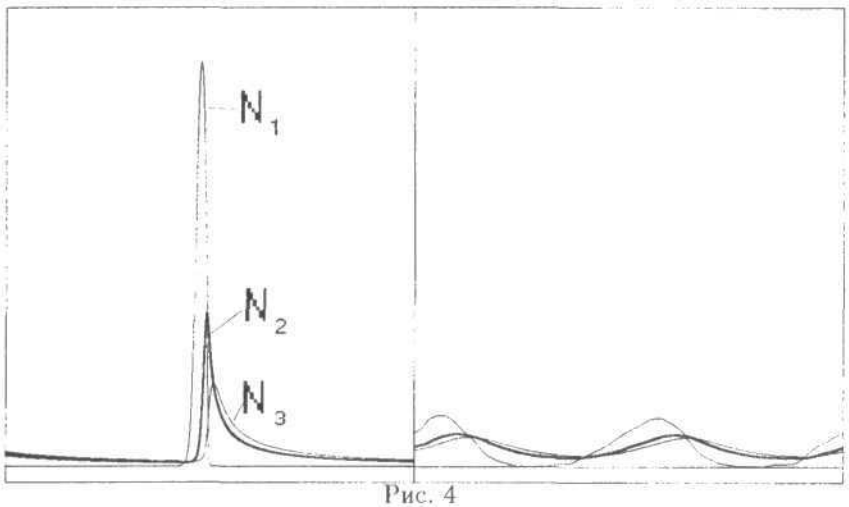


Рис. 4

Другие известные модели (их немного) таким свойством не обладают.

Автор выражает признательность Колесову Ю.С. за постановку задачи и помощь.

## Литература

[1] Колесов Ю.С., Колесов А.Ю., Майоров В.В. Рекция Белоусова: математическая модель и экспериментальные факты. // Динамика биологических популяций. Горький, 1987, с.43-51.

[2] Колесов Ю.С.,Майоров В.В.. Пространственная и временная самоорганизация в одновидовом биоценозе. // Динамика Биологических популяций. Горький, 1986. С. 3-13.

*Модел. и анализ информ. систем.* Т.**8**, №**1** (2001) 47-49

УДК 681.3

# Корректные отображения систем с переходами

Белов Ю.А.
*Ярославский государственный университет*
*150 000, Ярославль, Советская, 14*

Для систем с помеченными переходами определяется понятие корректных отображений. Используя это понятие, некоторые результаты о корректных редукциях сетей Петри переносятся на системы с переходами.

## Введение

При моделировании различных информационных систем сетями Петри важной задачей является минимизация полученной сети с сохранением заданного поведения. Имеется несколько различных определений поведенческой эквивалентности сетей Петри, наиболее тонкой из которых является, видимо, бисимуляционная эквивалентность [2, 3]. На ее основе ставится задача корректной редукции, то есть построения новой сети бисимулярной исходной, но меньшей размерности. В отмеченных работах получены условия корректности для двух видов редукции сетей, из этих условий выведен ряд свойств корректных редукций.

В данной работе эти условия корректности переносятся на более общую схему - системы с помеченными переходами [1], при этом некоторым аналогом корректных редукций становятся корректные отображения систем.

## 1. Определения

**Система с помеченными переходами** (labeled transitions systems LTS) - это тройка $D = \langle S, L, T\rangle$, где $S$ - произвольное абстрактное множество, называемое множеством состояний, $L$ - множество меток (имен) переходов, $T \subseteq S \times L \times S$ - множество переходов. Элементы из $T$ записываются в следующем виде: $s \xrightarrow{l} s'$, если $(s, l, s') \subseteq T$ и читаются так: система $D$ из состояния $s$ под действием перехода с именем $l$ перешла в состояние $s'$. Понятие бисимуляции ( то есть возможности взаимного моделирования поведения двух систем) определяется следующим образом. Пусть имеются две системы $D = \langle S_1, L, T_1\rangle$ и $H = \langle S_2, L, T_2\rangle$ с одинаковым множеством $L$ имен переходов. Бинарное отношение $R \subseteq S_1 \times S_2$ является **отношением бисимуляции**, если для любой пары $(s, q) \in R$ из того, что $s \xrightarrow{l} s'$ в $D$ следует, что в $H$ существует переход с той же меткой $l$ вида $q \xrightarrow{l} q'$, при котором $(s', q') \in R$. Аналогично, если $q \xrightarrow{l} q'$ в $H$, то в $D$ найдется переход $s \xrightarrow{l} s'$ с той же меткой $l$, при котором $(s', q') \in R$.

При исследовании поведения системы $D$ с переходами обычно задается некоторое начальное состояние $s_0 \in S$ и изучается пара $(D, s0)$. Две системы $D = \langle S_1, L, T_1\rangle$ и $H = \langle S_2, L, T_2\rangle$ $D$ и $H$ с одинаковыми множествами переходов называются **бисимулярными при начальных состояниях** $s_{01} \in S_1$ и $s_{02} \in S_2$, если существует такое отношение бисимуляции $R$, что $(s_{01}, s_{02}) \in R$, что будет обозначаться так: $(D, s_{01}) \cong (H, s_{02})$. Состояния $s_1$ и $s_2$ одной системы $D$ будем называть **бисимулярными**, если $(D, s1) \cong (D, s2)$. Короче это будем обозначать $s_1 \cong s_2$, если ясно, о какой системе идет речь. Легко проверить, что отношение бисимулярности на множестве состояний одной системы является отношением эквивалентности.

Дальнейшие определения описывают свойства отображений состояний одной системы в состояния другой системы. Для сетей Петри состояние сети определяется имеющейся на данный момент маркировкой - распределением фишек по позициям. При редукции сети каждой маркировке исходной сети можно естественным способом однозначно сопоставить некоторую маркировку редуцированной сети, что фактически и приводит к отображению множества состояний исходной сети в множество состояний редуцированной сети.

Итак, пусть заданы две системы с переходами - $D = \langle S_1, L, T_1\rangle$ и $H = \langle S_2, L, T_2\rangle$. Будем рассматривать однозначное отображение $f : S_1 \to S_2$. Говорим,что $f$ имеет **прямое свойство переноса**, если из того, что в $D$ существует переход $s \xrightarrow{l} s'$ следует, что в $H$ существует переход $f(s) \xrightarrow{l} f(s')$ с той же меткой $l$. Говорим, что $f$ имеет **обратное свойство переноса**, если из того, что в $H$ имеется переход $f(s) \xrightarrow{l} f(s')$

*для некоторых* $s, s' \in S_1$ *следует, что существуют такие состояния* $\tilde{s}, \tilde{s}'$, *что:* $f(\tilde{s}) = f(s)$, $f(\tilde{s}') = f(s')$ и существует переход $\tilde{s} \xrightarrow{l} \tilde{s}'$ в $D$.

Корректность отображения $f$ устанавливается по отношению к некоторому множеству "существенных" состояний $A \subseteq S_1$. Принцип выделения существенных маркировок (то есть состояний) для сетей Петри был обсужден и предложен в [2]. Там же предложено считать, что множество существенных состояний $A$ должно обладать такими свойствами:

- $A$ - *замкнуто относительно последователей*, то есть из того, что $s \in A$ и $s \xrightarrow{l} s'$ следует, что $s' \in A$;
- $A$ — $f$-замкнуто, то есть, если $s \in A$ и $f(s) = f(\tilde{s})$, то и $\tilde{s} \in A$.

Отображение $f$ называется **корректным относительно множества существенных состояний** $A$ или $A$-**корректным**, если для любого $s \in A$ $(D, s) \cong (H, f(s))$.

## 2. Критерий корректности отображения

В отмеченной работе [3] доказательство корректности редукции сети опирается на некоторое техническое утверждение, которое здесь переносится в более общую ситуацию систем с переходами и формулируется в терминах отображений, а не бинарных отношений, как в [3].

**Теорема 1.** *Пусть* $D$ *и* $H$ *- две системы с помеченными переходами. Пусть* $f$ *- отображение* $S_1$ *на* $S_2$, $A \subseteq S_1$ *- множество существенных состояний, замкнутое относительно последователей и* $f$*-замкнутое. Тогда если* $f$ *имеет оба свойства переноса и прообразы* $f$ *состоят из бисимулярных состояний (то есть для любых* $s, s_1$, *если* $f(s) = f(s_1)$, *то* $s_1 \cong s$*), то* $f$ *- A-корректно.*

*Доказательство.* Требуется доказать, что при указанных условиях для любого $s \in A$ $(D, s) \cong (H, f(s))$, или более кратко, $s \cong f(s)$, так как $D$ и $H$ фиксированы: $D = \langle S, L, T \rangle$ и $H = \langle S_1, L, T_1 \rangle$.

Определим некоторое отношение $R \subseteq A \times S_1 \subseteq S \times S_1$ и докажем, что оно будет отношением бисимуляции между $(D, s)$ и $(H, f(s))$:

$$R = \{\forall s \in A | (s, s_1) : \exists s' \in A : s \cong s', f(s') \cong s_1\}.$$

То есть пара $(s, s_1) \in R$ тогда и только тогда, когда существует такое состояние $s' \in A$, что $s' \cong s$ и $f(s') \cong s_1$.

Докажем, что $R$ является отношением бисимуляции. Пусть $(s, s_1) \in R$ и $s \xrightarrow{l} q$ , $s \in A$, $q \in S$. Докажем, что существует $q_1 \in S_1$ такой, что $s_1 \xrightarrow{l} q_1$ и $(q, q_1) \in R$. Рассмотрим диаграмму:

$$\begin{array}{ccccccc} s & \cong & s' & \xrightarrow{f} & f(s') & \cong & s_1 \\ {\scriptstyle l}\downarrow & & {\scriptstyle l}\downarrow & & {\scriptstyle l}\downarrow & & {\scriptstyle l}\downarrow \\ q & \cong & q' & \xrightarrow{f} & f(q') & \cong & q_1 \end{array}$$

и обоснуем существование всех ее объектов и отображений. $q \in A$ в силу $s \in A$ и замкнутости $A$ относительно последователей. $s' \in A$ существует по определению $R$ и $q'$ – по определению бисимулярности $s \cong s'$. $q' \in A$ - также в силу замкнутости $A$ относительно последователей. Переход $f(s') \xrightarrow{l} f(q')$ существует в силу существования перехода $s' \xrightarrow{l} q'$ и прямого свойства переноса $f$. Далее: $f(s') \cong s_1$ по определению $R$ и $q_1$ существует по определению бисимуляции в силу существования перехода $f(s') \xrightarrow{l} f(q')$. При этом пара $(q, q_1) \in R$, так как удовлетворяет определению отношения.

Обратно. Пусть $(s, s_1) \in R$ и имеется переход $s_1 \xrightarrow{l} q_1$. Докажем, что существует такое $q \in A$, что существует переход $s \xrightarrow{l} q$ и $(q, q_1) \in R$.

Рассмотрим диаграмму

$$\begin{array}{ccccccccc} s & \cong & s' & \cong & \tilde{s}' & \longrightarrow & f(\tilde{s}') & = & f(s') & \cong & s_1 \\ {\scriptstyle l}\downarrow & & {\scriptstyle l}\downarrow & & {\scriptstyle l}\downarrow & & {\scriptstyle l}\downarrow & & {\scriptstyle l}\downarrow & & {\scriptstyle l}\downarrow \\ q & \cong & q' & \cong & \tilde{q}' & \longrightarrow & f(\tilde{q}') & = & q_2 & \cong & q_1 \end{array}$$

и аналогично объясним ее. $s' \in A$ существует и $f(s') \cong s_1$ по определению $R$. $q_2$ и переход $f(s') \xrightarrow{l}$ существуют в силу бисимулярности $f(s') \cong s_1$ и существования перехода $s_1 \xrightarrow{l} q_1$. В силу того, $f$ — сюръекция (отображение на $S_1$), существует $\overline{q}'$ такое, что $f(\overline{q}') = q_2$. В силу обратного свойс переноса существует такая пара $\tilde{s}', \tilde{q}' \in S$, что $f(\tilde{s}') = f(s')$ и $f(\tilde{q}') = f(\overline{q}') = q_2$ и $\tilde{s}' \xrightarrow{l} \tilde{q}'$. Так $f(\tilde{s}') = f(s')$, $s' \in A$ и $A$ - $f$-замкнуто, получаем $\tilde{s}' \in A$, а тогда $\tilde{q}' \in A$ в силу замкнутости $A$ относител последователей. $s' \cong \tilde{s}'$ , так как $f(\tilde{s}') = f(s')$ и прообразы $f$ состоят из бисимулярных состояний тогда существует переход $s' \xrightarrow{l} q'$, где $q' \in A$ и $q' \cong \tilde{q}'$ в силу бисимулярности $\tilde{s}'$ и $s'$. Наконец, существ переход $s \xrightarrow{l} q$, где $q \cong q'$ в силу бисимулярности $s$ и $s'$ и $q \in A$ (т.к. $s \in A$). При этом получим

$$q \cong q' \cong \tilde{q}', \qquad f(\tilde{q}') = q_2 \cong q_1, \qquad q' \in A.$$

Это означает, что $(q, q_1) \in R$ – по определению.

Таким образом, доказано, что $R$ – бисимуляция. Для завершения доказательства отметим, что $\forall s$ очевидна принадлежность $(s, f(s)) \in R$, так как $s \cong s$ и в качестве $s'$ можно взять $s$. Тогда, в силу т что $R$ – бисимуляция и $(s, f(s)) \in R$, имеем $(D, s) \cong (H, f(s))$ для любого $s \in A$.

В заключение отметим, что свойство бисимулярности прообразов из $A$ обязательно имеется, ес $A$-корректно.

Действительно, если $f(s) = f(s')$, где $s, s' \in A$, то $s \cong f(s)$ и $s' \cong f(s')$ в силу корректности, а тог $s \cong s'$. Более того, если $f(s) \cong f(s')$, то $s \cong s'$.

## Литература


[1] Finkel A. Reduction and covering of infinite reachibility trees. Information and Computation. V.82 1990. P.144-179.

[2] Сидорова Н.С. Бисимуляционно-эквивалентные преобразования сетей Петри. Ярославль, Яр 1998. Препринт №1. 52 с.

[3] Schnoeblin Ph., Sidorova N. Bisimulation and reduction of Petri nets. Proc. $21^{th}$ Int. Conf. Appl. Theory of Petri Nets. Aarhus, Denmark, June 2000.