НАУЧНОЕ ИЗДАНИЕ МГТУ ИМ. Н. Э. БАУМАНА

НАУКА и ОБРАЗОВАНИЕ

Эл № ФС77 - 48211. Государственная регистрация №0421200025. ISSN 1994-0408

электронный научно-технический журнал

Временные оценки и гомоморфизмы асинхронных систем

 # 01, январь 2014

DOI: 10.7463/0114.0695993

Кудряшова Е. С., Хусаинов А. А.

УДК 519.7

Россия, МГТУ им. Н.Э. Баумана
ekatt@inbox.ru
husainov51@yandex.ru

Введение

В работе [1] были введены асинхронные системы для моделирования параллельных вычислительных систем. В [1, 2] рассматривались категории асинхронных систем. Морфизмы асинхронных систем $(S, s_0, E, I, Tran) \to (S', s'_0, E', I', Tran')$ определялись как пары отображений (η, σ), в которых $\eta\colon E \rightharpoonup E'$ — некоторое частичное отображение множеств событий, а $\sigma\colon S \to S'$ — отображение множеств состояний, удовлетворяющее дополнительным условиям. В работе [3] исследовались морфизмы, в которых η сопоставляет каждому событию произведение попарно независимых событий.

В данной работе мы изучаем морфизмы, введенные в работе [4] и примененные в [5] для оценки производительности параллельных вычислительных систем. Эти морфизмы задаются как пары (η, σ), в которых η сопоставляет каждому событию некоторую трассу и может интерпретироваться как разложение события в композицию операций. Это позволяет нам моделировать разложение каждой операции в композицию операций. В случае, когда операция разлагается в последовательное выполнение более мелких операций, мы приходим к алгоритму нахождения минимального времени выполнения процесса, заданного с помощью трассы.

1. Асинхронные системы и пунктированные полигоны

Введем необходимые определения моноида трасс, пространства состояний и асинхронной системы. Пространство состояний можно рассматривать как пунктированное множество с действием моноида трасс. Это позволяет определить гомоморфизмы пространств состояний как морфизмы пунктированных множеств с действием моноидов трасс. Гомоморфизмы асинхронных систем определяются как гомоморфизмы пространств состояний переводящие начальное состояние в начальное.

Моноиды трасс. Пусть E — множество. *Отношением независимости* на E называется подмножество $I \subseteq E \times E$, удовлетворяющее следующим условиям:

- для каждого $a \in E$ имеет место $(a, a) \notin I$ (антирефлексивность),
- для всех $a,\ b \in E$ верна импликация $(a, b) \in I \Rightarrow (b, a) \in I$ (симметричность).

Элементы $a,\ b \in E$ называются *независимыми*, если $(a, b) \in I$.

Пусть E^* обозначает моноид слов алфавита E, включающий пустое слово 1. Для произвольного отношения независимости I на E определен моноид $M(E, I)$, равный фактор-моноиду моноида слов E^*, полученного отождествлением слов, отличающихся перестановкой рядом стоящих независимых букв. Этот моноид называется *свободным частично коммутативным* или *моноидом трасс*. Он был введен и исследован в работе П. Картье и Д. Фоаты [6]. Имеет многочисленные приложения в параллельном программировании [7, 8, 9].

Пространства состояний. *Пространство состояний* $(S, E, I, Tran)$ состоит из произвольных множеств S, E, симметричного антирефлексивного отношения $I \subseteq E \times E$ и множества $Tran \subseteq S \times E \times S$, обладающих свойствами:

1) если $(s, a, s') \in Tran\ \&\ (s, a, s'') \in Tran$, то $s' = s''$;

2) если $(a, b) \in I\ \&\ (s, a, s') \in Tran\ \&\ (s', b, s'') \in Tran$, то существует такой $s_1 \in S$, что $(s, b, s_1) \in Tran\ \&\ (s_1, a, s'') \in Tran$ (рис. 1). **рис.1**

Элементы из S называются *состояниями*, из E — *событиями*, $I \subseteq E \times E$ — *отношением независимости*, элементы $(s, a, s') \in Tran$ — переходами.

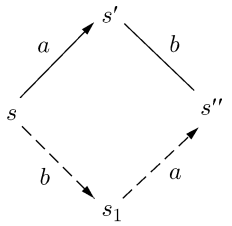

Рис. 1. Свойство 2

Пространство состояний можно рассматривать как множество S с частичным действием моноида трасс $M(E, I)$.

Хорошо известно [2], что частичное отображение $f\colon A \rightharpoonup B$ можно рассматривать как тотальное $\widehat{f}\colon A \cup \{*\} \to B \cup \{*\}$, добавляя к каждому множеству «бесконечно удаленную» точку $*$ и полагая для $x \in A$

$$\widehat{f}(x) = \begin{cases} f(x), & \text{если } f(x) \text{ определено;} \\ *, & \text{в других случаях.} \end{cases}$$

Для $x = *$ положим $\widehat{f}(x) = *$. Положим $A_* = A \cup \{*\}$. Множество A_* будет *пунктированным* в том смысле, что в нем выделена точка $*$. Отображение между пунктированными множествами, переводящее выделенную точку в выделенную называется *пунктированным*.

Определение 1. *Пунктированным полигоном* $(S, M(E,I), \cdot)$ *над моноидом трасс* называется тройка, состоящая из множества S, моноида трасс $M(E,I)$ и отображения $\cdot\colon S_* \times M(E,I) \to S_*$, $(s,\mu) \mapsto s \cdot \mu$, удовлетворяющего соотношениям

- $(\forall s \in S_*)(\forall \mu_1 \in M(E,I))(\forall \mu_2 \in M(E,I))\ (s \cdot \mu_1) \cdot \mu_2 = s \cdot (\mu_1 \mu_2)$;
- $(\forall s \in S_*)\ s \cdot 1 = s$;
- $(\forall \mu \in M(E,I))\ * \cdot \mu = *$.

В частности, каждое правое $M(E,I)$-множество можно рассматривать как пунктированный полигон над моноидом трасс, добавляя к множеству состояний состояние $*$ и определяя на нем действие как $* \cdot \mu = *$.

Соотношения показывают, что действия $(s,a) \mapsto s \cdot a$ достаточно определить для всех $s \in S_*$ и $a \in E$.

Предложение 1. [10, Теорема 2.3.5]. *Пусть* $(S_*, M(E,I), \cdot)$ *— пунктированный полигон над моноидом трасс. Положим*

$$Tran = \{(s,a,s') | s \in S, s' \in S, s \cdot a = s'\}.$$

Тогда четверка $(S,E,I,Tran)$ *будет пространством состояний. И наоборот, всякому пространству состояний* $(S,E,I,Tran)$ *будет соответствовать пунктированный полигон* $(S_*, M(E,I), \cdot)$ *с действием* $s \cdot a$*, определенным для всех* $s \in S_*$ *и* $e \in E$ *по формуле*

$$s \cdot a = \begin{cases} s', & \text{если существует такой } s' \in S, \text{ что } (s,a,s') \in Tran; \\ *, & \text{в других случаях.} \end{cases}$$

Гомоморфизмы пространств состояний. *Гомоморфизмом пространств состояний* называется морфизм соответствующих полигонов. Он задается с помощью гомоморфизма $f\colon M(E,I) \to M(E',I')$ и отображения $\sigma\colon S_* \to S'_*$, таких, что $\sigma(*) = *$ и $\sigma(s \cdot \mu) = \sigma(s) \cdot f(\mu)$ для всех $\mu \in M(E,I)$ и $s \in S_*$.

Асинхронной системой $A = (S, s_0, E, I, Tran)$ называется пространство состояний $(S,E,I,Tran)$ с выделенным элементом $s_0 \in S$, который называется *начальным состоянием*.

Предложение 1 показывает, что асинхронную систему можно рассматривать как пунктированный полигон $(S_*, M(E,I), \cdot)$ вместе с выделенным элементом $s_0 \in S$. Мы будем называть этот элемент *начальным состоянием*. Ниже мы будем обозначать переходы $(s,a,s') \in Tran$ через $s \xrightarrow{a} s'$.

Гомоморфизмом асинхронных систем называется гомоморфизм пространств состояний, переводящий начальное состояние в начальное.

2. Асинхронная система с функцией времени

Для распараллеливания вычислительного процесса этот процесс сначала разлагают в композицию конечной последовательности операторов, затем находят пары независимых операторов и объединяют независимые операторы в блоки-ярусы параллельной формы

[11]. Мы будем использовать эту идею для вычисления времени выполнения. Пусть A — асинхронная система. Мы будем интерпретировать события $a \in E$ как выполняющиеся во времени машинные команды, инструкции. Обозначим через $\mathbb{N}$ множество неотрицательных целых чисел. Пусть $\mathbb{N}_+$ — множество положительных целых чисел. Пусть для каждой инструкции задано время выполнения в тактах $\tau(a) \in \mathbb{N}_+$. Наша задача — найти алгоритм вычисления минимального времени, необходимого для выполнения конечной последовательности инструкций.

Параллельная форма. Мы рассмотрим сначала случай, когда $\tau(a) = 1$ для всех $a \in E$. *Параллельным выполнением* трассы $\mu = a_1 \cdots a_n \in M(E, I)$ называется ее разложение в произведение блоков $[B_1] \cdots [B_n]$, где каждый блок состоит из попарно независимых инструкций. Инструкции каждого блока будут выполняться одновременно. Поскольку время выполнения каждой инструкции $a \in E$ равно 1, то временем этого параллельного выполнения будет число блоков данного разложения. Рассматривая различные разложения, мы получим различные времена выполнения. Из этих соображений определим *минимальное время выполнения трассы* $\mu \in M(E, I)$ как наименьшее число блоков разложения этой трассы.

Определение 2. Для $[w] \in M(E, I)$ *разложением Фоаты* называется произведение классов $[w] = [w_1][w_2] \cdots [w_n]$, в котором:

1) для каждого $i = 1, \ldots, n$ слово w_i состоит из попарно независимых, и, стало быть, различных букв $a \in E$;

2) для каждого $i = 2, \ldots, n$ и для всякой буквы a из w_i существует такая буква b из w_{i-1}, что $(a, b) \notin I$.

Пусть E — линейно упорядоченное множество. В этом случае в каждом ярусе разложения Фоаты можно переставить буквы так, чтобы они располагались в возрастающем порядке. Получим нормальную форму в смысле определения 2. Она называется *нормальной формой Фоаты*, ее единственность мы установим ниже (следствие 1).

Например, для класса $[abba]$ в порожденном двумя элементами $\{a, b\}$ моноиде трасс нормальная форма Фоаты будет равна $abab$.

Предложение 2. [6]. *Для любого $[w] \in M(E, I)$ существует единственная последовательность слов $w_1, \ldots, w_n$, каждое из которых состоит из попарно независимых букв, таких, что $[w] = [w_1] \cdots [w_n]$ будет разложением Фоаты.*

◄ Докажем с помощью индукции по длине слова w. Пусть $w = w'a$. Тогда длина слова w' меньше длины слова w. Стало быть, существует разложение $[w] = [w_1] \cdots [w_n][a]$. Возможны следующие случаи.

1. Буква a зависима с некоторой буквой b из слова w_n, в этом случае это разложение будет разложением Фоаты.

2. Буква a независима со всеми буквами слова w_n, в этом случае сравниваем a со всеми буквами из слов $w_{n-1}, w_{n-2}, \cdots$, пока не встретим k, при котором w_k имеет букву, зависимую

с буквой a. В этом случае разложение Фоаты будет равно

$$[w_1]\cdots[w_k][w_{k+1}a][w_{k+2}]\cdots[w_n]\,.$$

3. Если a независима со всеми буквами из слов $w_1, \ldots, w_n$, то разложение Фоаты равно $[w_1a][w_2]\cdots[w_n]$.

Слова w_1, w_2, $\ldots$, w_n, будут единственными с точностью до перестановок. Если потребовать, чтобы буквы слов $w_1 \ldots, w_n$ были расположены в порядке возрастания, то слово $w_1\cdots w_n$ будет единственным. ►

Следствие 1. Пусть множество E линейно упорядочено. Тогда для всякого класса $[w] \in M(E, I)$ его нормальная форма Фоаты $w_1\cdots w_n \in [w]$ является единственной.

Для любой трассы ее разложение Фоаты имеет наименьшее число ярусов.

Следствие 2. Пусть $[w] = [w_1]\cdots[w_n]$ — разложение в произведение трасс, каждая из которых w_i состоит из попарно независимых букв. Тогда n не меньше, чем число ярусов разложения Фоаты элемента $[w]$.

◄ Доказательство вытекает из построения разложения Фоаты. Если некоторая буква из w_i не зависима со всеми буквами из w_{i-1}, то при построении разложения Фоаты она перейдет в один из блоков w_j, где $j < i$. Количество блоков от этого перемещения не увеличится. Поэтому число ярусов в разложении Фоаты не меньше, чем n. ►

Предложение 3. [7]. *Если время выполнения каждого события равно* 1*, то минимальное время выполнения трассы равно высоте ее нормальной формы Фоаты.*

Приведение трассы к нормальной форме Фоаты аналогично построению канонической ярусно-параллельной нормальной формы [11, 12].

Построение математической модели для расчета времени. Рассмотрим теперь общий случай. Пусть задана асинхронная система A с функцией времени, принимающей произвольные значения $\tau(a) \in \mathbb{N}_+$.

Попытаемся разложить переход $s \xrightarrow{a} s'$ в композицию переходов, имеющих зависимые инструкции e_1, e_2, $\ldots$, e_k, где $k = \tau(a)$. Каждая из инструкций e_i выполняется за один такт. Для того, чтобы это разложение стало возможным, добавим к множеству S «промежуточные» состояния и обозначим их через se_1, se_1e_2, $se_1e_2e_3$, $\ldots$, $se_1e_2\cdots e_{k-1}$. Положим $se_1e_2\cdots e_k = s'$, где $k = \tau(a)$. Получим разложение перехода $s \xrightarrow{a} s'$ в композицию $s \xrightarrow{e_1} s_1 \xrightarrow{e_2} s_2 \to \cdots \to s_{k-1} \xrightarrow{e_k} s'$, где $s_i = sa_1\cdots a_i$ для всех $1 \le i \le k-1$. Для расчета времени достаточно взять разложение перехода, в котором инструкции e_i равны между собой. Их общее значение будет соответствовать элементу a, и мы его обозначим через $\breve{a}$. Будет иметь место $e_i = \breve{a}$ для всех $1 \le i \le \tau(a) - 1$.

Если переходы $s \xrightarrow{a} s' \xrightarrow{b} s''$ имеют независимые инструкции, то $\breve{a}$ и $\breve{b}$ могут выполняться одновременно. Это приводит к необходимости ввести новые промежуточные состояния $s\breve{a}^i\breve{b}^j$, где $1 \le i \le \tau(a) - 1$, $1 \le j \le \tau(b) - 1$.

Определим новую асинхронную систему A_τ следующим образом. Введем на E некоторое произвольное отношение линейного порядка. Положим $E_\tau = \{\breve{a} \mid a \in E\}$. (Между элементами множеств E_τ и E существует биекция $a \mapsto \breve{a}$.) Отношение независимости I_τ будет состоять из пар $(\breve{a}, \breve{b})$, для которых $(a, b) \in I$. Введем множество состояний $S_\tau = \{(s, \breve{a}_1^{i_1}\breve{a}_2^{i_2}\cdots\breve{a}_m^{i_m}) \mid s \in S \;\&\; s \cdot a_1 \cdots a_m \in S \;\&\; a_1 < a_2 < \cdots < a_m \;\&\; (a_i, a_j) \in I$ для всех $1 \le i < j \le m \;\&\; 1 \le i_1 < \tau(a_1),\ 1 \le i_2 < \tau(a_2), \ldots, 1 \le i_m < \tau(a_m)\}$.

Мы будем рассматривать также состояния, у которых имеет место $i_q = 0$ или $i_q = \tau(a_q)$ для некоторого $q \in \{1, 2, \ldots, m\}$. Они будут отождествляться с некоторыми состояниями из S_τ, определенными по формулам

$$(s, \breve{a}_1^{i_1}\breve{a}_2^{i_2}\cdots\breve{a}_{q-1}^{i_{q-1}}\breve{a}^0\breve{a}_{q+1}^{i_{q+1}}\cdots\breve{a}_m^{i_m}) = (s, \breve{a}_1^{i_1}\breve{a}_2^{i_2}\cdots\breve{a}_{q-1}^{i_{q-1}}\breve{a}_{q+1}^{i_{q+1}}\cdots\breve{a}_m^{i_m}),$$
$$(s, \breve{a}_1^{i_1}\breve{a}_2^{i_2}\cdots\breve{a}_{q-1}^{i_{q-1}}\breve{a}_q^{\tau(a_q)}\breve{a}_{q+1}^{i_{q+1}}\cdots\breve{a}_m^{i_m}) = (s \cdot a_q, \breve{a}_1^{i_1}\breve{a}_2^{i_2}\cdots\breve{a}_{q-1}^{i_{q-1}}\breve{a}_{q+1}^{i_{q+1}}\cdots\breve{a}_m^{i_m}).$$

Определим частичное действие моноида $M(E_\tau, I_\tau)$ на S_τ. С этой целью для элемента $\breve{a} \in E_\tau$ и состояния $\sigma = (s, \breve{a}_1^{i_1}\cdots\breve{a}_m^{i_m}) \in S_\tau$ положим

$$\sigma \cdot \breve{a} = (s, \breve{a}_1^{i_1}\breve{a}_2^{i_2}\cdots\breve{a}_{q-1}^{i_{q-1}}\breve{a}_q^{i_q+1}\breve{a}_{q+1}^{i_{q+1}}\cdots\breve{a}_m^{i_m}),$$

если $a = a_q$ для некоторого $q \in \{1, \cdots, m\}$. Если же $(a, a_i) \in I$ для всех $i \in \{1, 2, \cdots, m\}$, то вставим элемент a в последовательность так, чтобы были верны неравенства $a_1 < \cdots <$ $< a_{q-1} < a < a_q < \cdots < a_m$, и положим $\sigma \cdot \breve{a} = (s, \breve{a}_1^{i_1}\breve{a}_2^{i_2}\cdots\breve{a}_{q-1}^{i_{q-1}}\breve{a}\breve{a}_q^{i_q}\cdots\breve{a}_m^{i_m})$.

Во всех остальных случаях полагаем $\sigma \cdot \breve{a} = *$. В этих случаях действие не определено.

Пусть A — асинхронная система с функцией времени $\tau : E \to \mathbb{N}_+$. *Минимальным временем выполнения трассы* $\mu = a_1 \cdots a_n \in M(E, I)$, такой, что $s \cdot \mu = s'$, для некоторых $s, s' \in S$ называется минимальное время выполнения трассы $\breve{a}_1^{\tau(a_1)} \cdots \breve{a}_n^{\tau(a_n)}$ асинхронной системы A_τ. Из предложения 3 получаем

Предложение 4. *Минимальное время выполнения трассы $a_1 \cdots a_n$, для которой существуют такие $s, s' \in S$, что $s \cdot a_1 \cdots a_n = s'$, равно высоте нормальной формы Фоаты трассы $\breve{a}_1^{\tau(a_1)}\breve{a}_2^{\tau(a_2)} \cdots \breve{a}_n^{\tau(a_n)}$.*

Вычисление времени с помощью разложения в композицию попарно независимых операций. Две трассы $\mu, \nu \in M(E, I)$ называются *независимыми*, если для любых букв a из μ и b из ν верно $(a, b) \in I$. В противном случае они *зависимы*. Будем называть трассы μ, ν *тотально зависимыми*, если для каждой буквы a из μ и для каждой буквы b из ν имеет место $(a, b) \notin I$.

Обобщим предложение 4 следующим образом. Вместо отображения $a \mapsto \breve{a}^{\tau(a)}$ будем рассматривать отображение ξ, сопоставляющее каждому $a \in E$ некоторую трассу $\xi(a) =$ $= \breve{a}_1 \cdots \breve{a}_{\tau(a)}$.

Докажем сначала вспомогательную лемму. Пусть $M(E, I)$ и $M(E', I')$ — моноиды трасс. Будем называть слова $a_1 \cdots a_n \in E^*$ и $a'_1 \cdots a'_n \in E'^*$ *подобными*, если для всех $1 \le i < j \le n$ имеет место

$$(a_i, a_j) \in I \;\Leftrightarrow\; (a'_i, a'_j) \in I'.$$

Лемма 1. Пусть $M(E, I)$ и $M(E', I')$ – моноиды трасс. Если слова $a_1 \cdots a_n \in E^*$ и $a'_1 \cdots a'_n \in E'^*$ подобны, то их высоты нормальных форм трасс $[a_1 \cdots a_n]$ и $[a'_1 \cdots a'_n]$ равны.

◄ Введем произвольное отношение линейного порядка на E. Перестановка букв (a_i, a_{i+1}) слова $a_1 \cdots a_n$ возможна тогда и только тогда, когда возможна перестановка (a'_i, a'_{i+1}). Причем такие перестановки подобных слов будут приводить к подобным словам. Отсюда последовательность перестановок, приводящая трассу $[a_1 \cdots a_n]$ к нормальной форме $[B_1] \cdots [B_m]$, приведет трассу $[a'_1 \cdots a'_n]$ к произведению некоторых блоков $B'_1 \cdots B'_m$, каждый из которых состоит из попарно независимых букв. Вводя отношение линейного порядка на E' и производя сортировку букв внутри каждого блока B'_i, получим нормальную форму трассы $[a'_1 \cdots a'_n]$. Отсюда вытекает доказываемое утверждение. Трассы имеют равные высоты. Более того, длины блоков B_i и B'_i будут равны для всех $1 \leq i \leq m$. ►

Предложение 5. *Пусть A — асинхронная система с функцией времени τ: $E \to \mathbb{N}_+$. Для произвольного гомоморфизма ξ: $M(E, I) \to M(E', I')$, переводящего независимые трассы в независимые, а тотально зависимые — в тотально зависимые, такие, что для каждого $a \in E$ длина трассы $\xi(a)$ равна $\tau(a)$, время выполнения каждой трассы $[a_1 \cdots a_n]$ будет равно высоте нормальной формы трассы $[\xi(a_1) \cdots \xi(a_n)]$.*

◄ В этом случае слово $\xi(a_1) \cdots \xi(a_n)$ будет подобно слову $a_1^{\tau(a_1)} \cdots a_n^{\tau(a_n)}$. Из леммы 1 вытекает, что высоты соответствующих трасс равны. С помощью предложения 4 получаем желаемое. ►

Гомоморфизмы, сохраняющие минимальное время. Пусть (A, τ) и (A', τ') — асинхронные системы с функциями времени. Для произвольной трассы $\mu \in M(E, I)$, допускающей состояния $s_1, s_2 \in S$, такие, что $s_1 \cdot \mu = s_2$, обозначим через $\vartheta(\mu)$ ее минимальное время выполнения.

Определение 3. Гомоморфизм асинхронных систем $(f, \sigma) : A \to A'$ называется *сохраняющим минимальное время*, если для произвольной трассы $\mu \in M(E, I)$, допускающей состояния $s_1,\ s_2 \in S$, такие, что $s_1 \cdot \mu = s_2$, имеет место $\vartheta(\mu) = \vartheta'(f(\mu))$.

Теорема 1. Пусть (f, σ): $A \to A'$ — гомоморфизм асинхронных систем, и пусть τ и τ' — временные функции на этих асинхронных системах. Предположим, что гомоморфизм обладает следующим свойствами.

1. Для каждого $e \in E$ трасса $f(e) \in M(E', I')$ является произведением попарно зависимых элементов $f(e) = e'_1 \cdots e'_n$, где $e'_1, \ldots, e'_n \in E'$. Причем для этих элементов имеет место равенство $\tau(e) = \tau'(e'_1) + \cdots + \tau'(e'_n)$.

2. Для любой пары $(a, b) \in I$ трассы $f(a)$ и $f(b)$ независимы.

3. Если a и b зависимы, то $f(a)$ и $f(b)$ тотально зависимы.

Тогда (f, σ) сохраняет минимальное время.

◄ Рассмотрим произвольную трассу $\mu \in M(E, I)$ и докажем, что минимальное время ее выполнения равно минимальному времени выполнения $\vartheta'(f(\mu))$ трассы $f(\mu)$. Пусть $\mu =$

$[e_1 \cdots e_m]$. Время $\vartheta(\mu)$ равно высоте нормальной формы Фоаты для трассы $\breve{e}_1^{\tau(e_1)} \cdots \breve{e}_m^{\tau(e_m)}$. Согласно предложению 5, время $\vartheta(\mu)$ будет равно высоте нормальной формы трассы $\xi(e_1) \cdots \xi(e_m)$, для произвольной функции ξ, сопоставляющей каждой букве класс произведения $\tau(e)$ попарно зависимых букв и переводящей независимые буквы в независимые трассы, а зависимые буквы в тотально зависимые.

По предположению, для каждого $e \in E$ существует единственное слово $e'_1 \cdots e'_n$, такое, что $f(e) = e'_1 \cdots e'_n$. Это позволяет нам определить функцию ξ: $E \to M(E'_{\tau'}, I'_{\tau'})$ по формуле

$$\xi(e) = \breve{e}_1'^{\,\tau'(e'_1)} \cdots \breve{e}_n'^{\,\tau(e'_n)}.$$

Трасса $\xi(e)$ состоит из $\tau'(e'_1) + \cdots \tau'(e'_n) = \tau(e)$ попарно зависимых букв, и функция ξ удовлетворяет условиям предложения 5, откуда минимальное время выполнения трассы μ будет равно высоте нормальной формы Фоаты трассы $\xi(e_1) \cdots \xi(e_m)$.

Поскольку $\xi(e)$ получается из $f(e) = e'_1 \cdots e'_n$ с помощью замены букв e'_i элементами $\breve{e}_i'^{\,\tau'(e'_i)}$, то $\xi(e_1) \cdots \xi(e_m)$ получается из $f(\mu) = f(e_1) \cdots f(e_m)$ заменой букв e'_i элементами $\breve{e}_i'^{\,\tau'(e'_i)}$. Отсюда следует, что минимальной время работы трассы $f(\mu)$ равно высоте нормальной формы Фоаты трассы $\xi(e_1) \cdots \xi(e_m)$.

Следовательно, $\vartheta(\mu) = \vartheta'(f(\mu))$. ►

3. Вычисление минимального времени выполнения

В данном пункте мы опишем алгоритм вычисления минимального времени выполнения и эксперимент, частично подтверждающий этот алгоритм. Эксперимент будет проведен для процесса, который можно определить следующим образом.

Определение 4. Псевдо-конвейером называется цикл, состоящий из конечной последовательности действий $a_1 a_2 \cdots a_n$, в которой события a_i и a_{i+1} зависимы для всех $1 \leq i \leq$ $\leq n-1$, а все остальные события независимы. Этот цикл выполняется некоторое конечное число раз.

Псевдо-конвейер можно представить с помощью трассы $(a_1 a_2 \cdots a_n)^m$, для некоторого $m \geq 1$. Например, этому определению удовлетворяет конечная последовательность, состоящая из n неделимых операций присваивания $x_i = f_i(x_{i-1})$, $1 \leq i \leq n$, составляющих цикл, выполняющийся m раз.

Алгоритм нахождения минимального времени. Согласно приведенным выше утверждениям, для нахождения минимального времени выполнения трассы $a_1 \cdots a_n$, нужно выписать трассу

$$\breve{a}_1^{\tau(a_1)} \breve{a}_2^{\tau(a_2)} \cdots \breve{a}_n^{\tau(a_n)}$$

и затем построить ее нормальную форму Фоаты. Количество блоков этой нормальной формы будет равно искомому минимальному времени.

Мы будем рассматривать процессы, заданные как последовательности срабатываний переходов сети Петри. Напомним терминологию.

Сеть Петри определяется как пятерка $(P, T, pre, post, M_0)$, состоящая из конечных множеств P и T, функций $M_0 : P \to \mathbb{N}$, pre: $T \to \mathbb{N}^P$, $post$: $T \to \mathbb{N}^P$. Здесь $\mathbb{N}^P$ обозначает множество всех функций $P \to \mathbb{N}$. Элементы $p \in P$ называются *местами*, $t \in T$ — *переходами*, $M \in \mathbb{N}^P$ — *маркировками*, а M_0 — *начальной маркировкой*. Определим отношение порядка на $\mathbb{N}^P$, полагая $M \leq M'$, если для всех $p \in P$ верно $M(p) \leq M'(p)$. Сумму и разность функций определим как $(M \pm M')(p) = M(p) \pm M'(p)$. Для M, $M' \in \mathbb{N}^P$ и $t \in T$ запись $M \xrightarrow{t} M'$ будет означать, что выполнены следующие два условия:

1) $M \geq pre(t)$;

2) $M' = M - pre(t) + post(t)$.

В этом случае, мы будем говорить, что маркировка M' получена из M с помощью срабатывания перехода t.

Пусть $(P, T, pre, post, M_0)$ — сеть Петри. Обозначим ${}^\bullet t = \{p \in P : pre(t)(p) \neq 0\}$, $t^\bullet = \{p \in P : post(t)(p) \neq 0\}$.

Определим отношение

$$I = \{(t_1, t_2) \in T \times T : ({}^\bullet t_1 \cup t_1{}^\bullet) \cap ({}^\bullet t_2 \cup t_2{}^\bullet) = \emptyset\}.$$

Легко видеть, что I будет отношением независимости на множестве T. Пусть $\mathbb{N}^P$ — множество всех маркировок, а $Tran$ состоит из троек (M_1, t, M_2), для которых $M \xrightarrow{t} M'$. Тогда пятерка $(\mathbb{N}^P, M_0, T, I, Tran)$ будет асинхронной системой.

Предположим, что каждому переходу $a \in T$ сопоставлено положительное целое число $\tau(a)$ — время выполнения перехода. Покажем, как находить минимальное время выполнения трассы, состоящей из переходов, связывающих маркировки.

Пример 1. Рассмотрим псевдо-конвейер, состоящий из трех операций, применяемых к очереди, содержащей m элементов (рис. 2). Каждая из операций считается неделимой. Она читает данные из области памяти, соответствующей входному месту, вычисляет значение некоторой функции и записывает в область памяти выходного места. Таким образом, соседние операции зависимы. **рис.2**

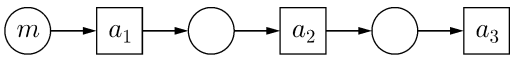

Рис. 2. Сеть Петри псевдо-конвейера

Трасса выполнения процесса будет равна $t = [(a_1 a_2 a_3)^m]$. Пусть $\tau(a_1) = 2$, $\tau(a_2) = 1$, $\tau(a_3) = 2$. Тогда временная трасса, соответствующая трассе $[(a_1 a_2 a_3)^m]$ будет равна $[(a_1^2 a_2 a_3^2)^m]$. После приведения к нормальной форме Фоаты получаем $[(a_1^2 a_2 a_3^2)^m] = [a_1][a_1]([a_2][a_1 a_3][a_1 a_3])^{m-1}[a_2][a_3][a_3]$. Отсюда $\vartheta(t) = 1+1+3(m-1)+1+1+1 = 3m+2$. Если выполнять действия последовательно, то время выполнения будет равно $5m$.

Описание эксперимента. Для постановки эксперимента с целью подтверждения или опровержения полученных результатов мы разработали программное обеспечение, вычисляющее время работы сети Петри псевдо-конвейера или более общей системы — синхронизационного графа. Опишем алгоритм работы программы.

Программа организована таким образом, что в процессе работы для каждого перехода сети создается отдельный поток. Взаимодействие между потоками обеспечивает выполнение условия зависимости соседних операций.

Для того чтобы потоки выполнялись одновременно, за операцию в данной программе будем принимать задержку выполнения на заданное пользователем время. В этом заключается основная идея, делающая возможными эксперименты по измерению времени работы параллельно работающих процессов.

Работа программы начинается с создания глобального таймера, который заканчивает отсчет времени после выполнения последней операции. С помощью него мы получаем на выходе результирующее время работы псевдо-конвейера.

В качестве объекта синхронизации работы потоков в данной программе выступает *событие* (event). Объект событие может находиться в двух состояниях: *занят* (есть событие) и *свободен* (нет события). Состояние, при котором объект событие занят, называется *сигнальным*, в противном случае *невыделенным*. Функция потока представляет собой ожидание сигналов о завершении смежных потоков, выполнение операции и последующую установку события, сигнализирующего об окончании работы.

Эксперимент для псевдо-конвейера. Рассмотрим сеть Петри псевдо-конвейера, изображенного на рис. 3. **рис.3**

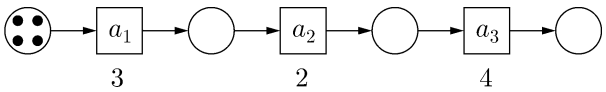

Рис. 3. Пример псевдо-конвейера

Пусть $\tau(a_1) = 3$, $\tau(a_2) = 2$, $\tau(a_3) = 4$. Используя алгоритм расчета минимального времени выполнения трассы, описанный выше, получим

$$[(a_1^3 a_2^2 a_3^4)^m] = [a_1][a_1][a_1]([a_2][a_2][a_1a_3][a_1a_3][a_1a_3][a_3])^{m-1}[a_2][a_2][a_3][a_3][a_3][a_3].$$

Отсюда $\vartheta(t) = 3 + 6(m-1) + 2 + 4 = 6m + 3$. При $m = 4$ имеем $\vartheta(t) = 27$.

Теперь рассчитаем время выполнения трассы экспериментальным путем.

После запуска программы в главном потоке создается три нити — по одной на каждую операцию. Для синхронизации работы потоков создается три события. Начальное состояние событий — сигнальное. Первый поток a_1 ожидает установки события $event_2$. Поскольку все события установлены, поток a_1 начинает выполнять операцию, при этом меняя состояние события $event_1$ на невыделенное. По окончанию работы поток a_1 освобождает $event_2$ и устанавливает $event_1$. Поток a_2 постоянно находится в ожидании событий $event_1$ и $event_3$. Они сигнализируют ему о том, что потоки a_1 и a_3 завершили свою работу. Дождавшись сигнала $event_1$ и $event_3$ поток a_2 осуществляет задержку, освобождает эти события и меняет состояние $event_2$ на сигнальное. Сигнал $event_2$ позволяет a_1 и a_3 приступить к работе. Поток a_1 заканчивает свою работу через время $\tau(a_1) = 3$, и снова освобождает $event_2$ и

устанавливает $event_1$. При этом поток a_3 работает на 1 секунду дольше и по завершению освобождает $event_2$ и устанавливает $event_3$. Далее управление переходит к потоку a_2. Передача управления между потоками продолжается до тех пор, пока количество элементов $m > 0$. По окончанию работы программы мы получаем минимальное время выполнения трассы, которое совпадает с теоретическими расчетами.

На рис. 4 представлен график зависимости времени выполнения трассы от количества m элементов очереди для примера рис. 3. Легко видеть, что при малых m, экспериментальные расчеты (пунктирная линия) совпадают с теоретическими, тогда как при увеличении m наблюдается небольшая погрешность вычислений. **рис.4**

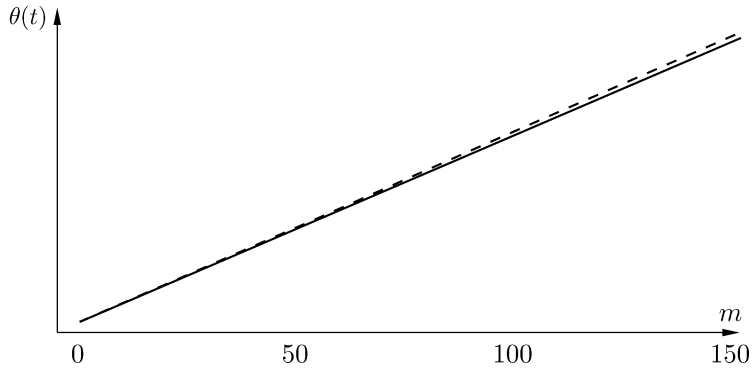

Рис. 4. Зависимость времени обработки m элементов очереди

Более общие структуры. В случае изменения структуры конвейерной обработки, работа программы осуществляется аналогичным образом. Это можно применить, например, к *волновым системам* [13]. Для иллюстрации этого утверждения рассмотрим сеть, представленную на рис. 5. **рис.5**

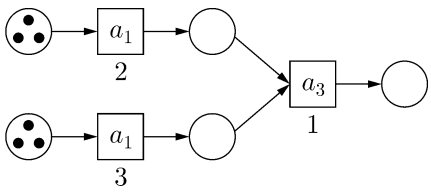

Рис. 5. Пример обработки конвейерного типа

Выполнив построение нормальной формы Фоаты, получим выражение для нахождения минимального времени выполнения:

$$[(a_1^2 a_2^3 a_3^1)^m] = [a_1 a_2][a_1 a_2][a_2]([a_3][a_1 a_2][a_1 a_2][a_2])^{m-1}[a_3].$$

Отсюда $\vartheta(t) = 3 + 4(m-1) + 1 = 4m$. При $m = 3$, $\vartheta(t) = 12$.

Поскольку данный пример во многом схож с примером рис. 3, остановимся исключительно на синхронизации работы потоков. Потоки a_1 и a_2 находятся в ожидании события $event_2$ и выполняются параллельно. После завершения операций, оба потока освобождают $event_2$ и устанавливают $event_1$ и $event_3$ соответственно. Поток a_2 получает управление

после перехода $event_1$ и $event_3$ в сигнальное состояние. Результат выполнения программы также совпадает с теоретическими расчетами.

Вычислительный эксперимент показал справедливость теоретических расчетов для некоторых частных случаев трасс. Однако стоит учесть, что успех эксперимента напрямую зависит от мощности вычислительных машин, на которых он проводится. В процессе тестирования было установлено, что совпадение теоретических и экспериментальных значений минимального времени выполнения наблюдается при относительно больших величинах $\tau(a)$ (порядка 1 с). При уменьшении времени выполнения операций появляется погрешность вычислений, в некоторых случаях превосходящая результирующее время. Наличие погрешности можно связать с тем, что при малых $\tau(a)$ временные затраты на установку событий и передачу управления между потоками превосходят затраты на собственно вычисления. Мы не учитываем этих погрешностей. Этой проблеме посвящена, например, работа [14], но в ней рассматриваются только конвейерные системы. Наш способ позволяет вычислять производительность любой асинхронной системы, для которой заданы времена выполнения событий.

Данный вопрос требует дополнительных исследований, поэтому проведенный эксперимент можно считать условно успешным.

Заключение

Применение рассмотренных асинхронных систем к сетям Петри, у которых независимы переходы, не имеющие общих мест, приводит к способу нахождения минимального времени выполнения для параллельных процессов, моделируемых с помощью сетей Петри. Эксперимент частично подтверждает правильность выбора модели для расчета времени выполнения.

В перспективе исследование оценок времени для разложения операций в композицию операций, которые могут быть независимы. Для изучения этого вопроса было бы хорошо установить связь между асинхронными системами с функцией времени и временными многомерными автоматами, введенными Губо [15].

Работа выполнена в рамках программы стратегического развития государственных образовательных учреждений высшего профессионального образования, № 2011-ПР-054.

Список литературы

1. Bednarczyk M. Categories of Asynchronous Systems. Brighton: University of Sussex, 1987. 230 p.
2. Winskel G., Nielsen M. Models for Concurrency // Handbook of Logic in Computer Science. Vol. 4. Semantic Modelling / Abramsky S., Gabbay D.M., Maibaum T.S.E. (eds.). Oxford University Press, 1995. P. 1–148.
3. Bednarczyk M.A., Bernardinello L., Caillaud B., Pawlowski W., Pomello L. Modular System Development with Pullbacks // Applications and Theory of Petri Nets 2003. Berlin: Springer-

Verlag, 2003. P. 140–160. DOI: 10.1007/3-540-44919-1_12 (Ser. Lecture Notes in Computer Science; vol. 2679).

4. Husainov A. On the homology of small categories and asynchronous transition systems // Homology, Homotopy and Applications. 2004. Vol. 6, no. 1. P. 439–471. Available at: http://projecteuclid.org/euclid.hha/1139839561 (accessed 01.12.2013).
5. Хусаинов А.А., Кудряшова Е.С. Модель для временных оценок параллельных вычислительных систем // Информационные технологии XXI века: материалы международной научной конференции (Хабаровск, 20-24 мая 2013 г.). Хабаровск: Изд-во Тихоокеан. гос. ун-та, 2013. С. 203–207.
6. Cartier P., Foata D. Problèmes combinatories de commutation et rearrangements. Berlin: Springer-Verlag, 1969. 88 p. (Ser. Lecture Notes in Math.; vol. 85).
7. Diekert V. Combinatorics on Traces. Berlin: Springer-Verlag, 1990. 169 p. DOI: 10.1007/3-540-53031-2 (Ser. Lecture Notes in Computer Science; vol. 454).
8. Diekert V., Métivier Y. Partial Commutation and Traces // Handbook of Formal Languages. Vol. 3. Beyond Words. New York: Springer-Verlag, 1997. P. 457–533. DOI: 10.1007/978-3-642-59126-6_8.
9. Mazurkiewicz A. Trace theory // Petri Nets: Applications and Relationships to Other Models of Concurrency. Berlin: Springer-Verlag, 1987. P. 278–324. DOI: 10.1007/3-540-17906-2_30 (Ser. Lecture Notes in Computer Science; vol. 255).
10. Хусаинов А. Исследование параллельных систем методами теории категорий. Саарбрюкен: LAP LAMBERT Academic Publishing, 2012. 152 с.
11. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. Санкт-Петербург: БХВ-Петербург, 2002. 602 с.
12. Карпов В.Е. Введение в распараллеливание алгоритмов и программ // Компьютерные исследования и моделирование. 2010. Т. 2, № 3. С. 231–272. Режим доступа: http://crm.ics.org.ru/journal/article/1725/ (дата обращения 01.12.2013).
13. Трещев И.А. Методы построения систем автоматизированного распараллеливания приложений для архитектур с симметрично-адресуемой памятью // Ученые записки Комсомольского-на-Амуре государственного технического университета. Сер. «Науки о природе и технике». 2011. Т. 5, № I-1. С. 29–32. Режим доступа: http://www.uzknastu.ru/files/pdf/I-1(5)2011/29-32.pdf (дата обращения 01.12.2013).
14. Герценбергер К.В., Чепин Е.В. Аналитическая модель оценки производительности многопроцессорной обработки данных для набора параллельных алгоритмических структур // Бизнес-информатика. 2011. № 4. С. 24–30. Режим доступа: http://bijournal.hse.ru/2011–4%20(18)/44322998.html (дата обращения 01.12.2013).
15. Goubault E. Durations for truly-concurrent transitions // Programming Languages and Systems — ESOP '96. Berlin: Springer-Verlag, 1996. P. 173–187. DOI: 10.1007/3-540-61055-3_36 (Ser. Lecture Notes in Computer Science; vol. 1058).

SCIENTIFIC PERIODICAL OF THE BAUMAN MSTU

SCIENCE and EDUCATION

EL № FS77 - 48211. №0421200025. ISSN 1994-0408

electronic scientific and technical journal

Timed estimates and homomorphisms of asynchronous systems

 # 01, January 2014

DOI: 10.7463/0114.0695993

Kudryashova E. S., Husainov A. A.

Bauman Moscow State Technical University
105005, Moscow, Russian Federation
ekatt@inbox.ru
husainov51@yandex.ru

We study asynchronous systems used for mathematical modeling of the parallel computer system. It is considered as a set with partially trace monoid action. This allowed us to introduce homomorphisms of asynchronous systems as corresponding polygonal morphisms. Asynchronous systems with time function are studied using these homomorphisms. An algorithm for computing the minimal execution time of parallel processes in asynchronous systems is constructed. The conditions of homomorphisms of asynchronous systems minimal execution time keeping are found. The algorithm is used for compute of execution time of parallel processes, which consists of Petri net transitions. There are examples of computing minimal execution time for pseudo-pipeline and wave system. An experiment realized with multithreaded application that is built on a given Petri net is described. The experiment confirms the method of estimating the minimum execution time of the parallel process.

Publications with keywords: commutative monoid, trace monoid, asynchronous transition system, timed systems

Publications with words: commutative monoid, trace monoid, asynchronous transition system, timed systems

References

1. Bednarczyk M. *Categories of Asynchronous Systems*. Brighton, University of Sussex, 1987. 230 p.
2. Winskel G., Nielsen M. Models for Concurrency. In: Abramsky S., Gabbay D.M., Maibaum T.S.E. (eds.). Handbook of Logic in Computer Science. Vol. 4. Semantic Modelling. Oxford University Press, 1995, pp. 1–148.

3. Bednarczyk M.A., Bernardinello L., Caillaud B., Pawlowski W., Pomello L. Modular System Development with Pullbacks. In: Applications and Theory of Petri Nets 2003. Berlin, Springer-Verlag, 2003, pp. 140–160. DOI: 10.1007/3-540-44919-1_12 (Ser. Lecture Notes in Computer Science; vol. 2679).

4. Husainov A. On the homology of small categories and asynchronous transition systems. Homology, Homotopy and Applications, 2004, vol. 6, no. 1, pp. 439–471. Available at: http://projecteuclid.org/euclid.hha/1139839561, accessed 01.12.2013.

5. Khusainov A.A., Kudryashova E.S. Model' dlya vremennykh otsenok parallel'nykh vychislitel'nykh sistem [Model for time estimate of parallel computing systems]. Informatsionnye tekhnologii 21 veka: materialy mezhdunarodnoy nauchnoy konferentsii [Information Technologies of 21 Century: Proceedings of the International Scientific Conference], Khabarovsk, 20-24 May 2013. Khabarovsk, Pacific National University Publ., 2013, pp. 203–207.

6. Cartier P., Foata D. Problèmes combinatories de commutation et réarrangements. Berlin, Springer-Verlag, 1969. 88 p. (Ser. Lecture Notes in Math.; vol. 85).

7. Diekert V. Combinatorics on Traces. Berlin, Springer-Verlag, 1990. 169 p. DOI: 10.1007/3-540-53031-2 (Ser. Lecture Notes in Computer Science; vol. 454).

8. Diekert V., Métivier Y. Partial Commutation and Traces. In: Handbook of Formal Languages. Vol. 3. Beyond Words. New York, Springer-Verlag, 1997, pp. 457–533. DOI: 10.1007/978-3-642-59126-6_8.

9. Mazurkiewicz A. Trace theory. In: Petri Nets: Applications and Relationships to Other Models of Concurrency. Berlin, Springer-Verlag, 1987, pp. 278–324. DOI: 10.1007/3-540-17906-2_30 (Ser. Lecture Notes in Computer Science; vol. 255).

10. Khusainov A. Issledovanie parallel'nykh sistem metodami teorii kategoriy [Parallel systems using the theory of categories analysis]. LAP LAMBERT Academic Publishing, Saarbruken, 2012. 152 p. (in Russian)

11. Voevodin V.V., Voevodin Vl.V. Parallel'nye vychisleniya [Concurrent calculatios]. St. Petersburg, BHV-Petersburg, 2002. 602 p.

12. Karpov V.E. Vvedenie v rasparallelivanie algoritmov i programm [Introduction to the parallelization of algorithms and programs]. Komp'yuternye issledovaniya i modelirovanie [Computer Research and Modeling], 2010, vol. 2, no. 3, pp. 231–272. Available at: http://crm.ics.org.ru/journal/article/1725/, accessed 01.12.2013.

13. Treshchev I.A. Metody postroeniya sistem avtomatizirovannogo rasparallelivaniya prilozheniy dlya arkhitektur s simmetrichno-adresuemoy pamyat'yu [Methods for building systems of automated parallelization programming for architectures with symmetrically-addressable memory]. Uchenye zapiski Komsomol'skogo-na-Amure gosudarstvennogo tekhnicheskogo universiteta. Ser. "Nauki o prirode i tekhnike" [Scholarly Notes of Komsomolsk-na-Amure

State Technical University. Ser. “Engineering and Natural Sciences”], 2011, vol. 5, no. I-1, pp. 29–32. Available at: http://www.uzknastu.ru/files/pdf/I-1(5)2011/29-32.pdf, accessed 01.12.2013.

14. Gertsenberger K.V., Chepin E.V. Analiticheskaya model’ otsenki proizvoditel’nosti mnogoprotsessornoy obrabotki dannykh dlya nabora parallel’nykh algoritmicheskikh struktur [Analytical model for evaluating performance multiprocessing for a set of parallel algorithmic structure]. Biznes-informatika [Business Informatics], 2011, no. 4, pp. 24–30. Available at: http://bijournal.hse.ru/2011–4%20(18)/44322998.html, accessed 01.12.2013.
15. Goubault E. Durations for truly-concurrent transitions. In: Programming Languages and Systems — ESOP ’96. Berlin, Springer-Verlag, 1996, pp. 173–187. DOI: 10.1007/3-540-61055-3_36 (Ser. Lecture Notes in Computer Science; vol. 1058).