Программное обеспечение для декомпозиции двудольных орграфов

Зайцев Д.А.

Одесская национальная академия связи им. А.С.Попова
Ул. Кузнечная 1, Одесса 65029, Украина
www.geocities.com/zsoftua

Abstract

Zaitsev D.A. Software for decomposition of bipartite directed graphs. *Software for decomposition of bipartite directed graphs (Petri nets) into functional subnets was represented. Decomposition is aimed to speed-up of large-scale Petri net models analysis. Basic operations of decomposition algorithm were defined, the optimization of data structures for effective implementation of basic operations was executed, detailed algorithm of decomposition was represented. Variants of decomposition software module integration into automated systems of Petri net models analysis and synthesis were studied. Results of decomposition a host of nets allow the conclusion about good enough partibility of real-life objects' models.*

Keywords: *bipartite digraph, Petri net, decomposition, functional subnet*

Введение

Известные методы нахождения инвариантов сетей Петри имеют экспоненциальную вычислительную сложность [1-3], что делает их практически неприменимыми для исследования сетей большой размерности. В работах [4,5] предложено использовать декомпозицию двудольных ориентированных графов на функциональные подсети [6,7] для ускорения инвариантного анализа моделей Петри. Кроме того, декомпозиция успешно применена для ускорения решения фундаментального уравнения [8] сетей Петри. Полученные ускорения вычислений оцениваются экспоненциальной функцией от размерности сети [4,5,8].

Представленный в [4,5] обобщённый алгоритм декомпозиции требует дополнительной детализации в целях обеспечения его эффективной программной реализации. Возникает проблема, связанная с разработкой структур данных и алгоритмических конструкций, обеспечивающих наименьшую временную и емкостную сложность. Как правило, для моделей Петри реальных объектов характерна относительно невысокая плотность дуг. Таким образом, работа с упакованным представлением графов является одним из основных требований к программному обеспечению.

Целью настоящей работы является создание эффективного программного обеспечения для декомпозиции моделей Петри большой размерности на функциональные подсети.

Следует отметить, что декомпозиция на минимальные функциональные сети [6,7] является вспомогательным методом, предназначенным для ускорения инвариантного анализа и процесса решения фундаментального уравнения сетей Петри [4,5,8]. Поэтому в качестве основного программного продукта представлен универсальный встраиваемый модуль DECO декомпозиции сетей Петри, реализованный на языке программирования ANSI C.

На основе модуля DECO разработана программа DEBORAH с интерфейсом командной строки для декомпозиции сетей, заданных текстовыми файлами, которая реализована в среде ОС Unix и Windows. Кроме того, выполнено встраивание модуля декомпозиции DECO в известную моделирующую систему Tina [9]; при этом обеспечена поддержка как абстрактного, так и графического форматов представления сетей, принятых в системе Tina.

Основные понятия и определения

Сеть Петри – это тройка $N=(P,T,F)$, где $P=\{p\}$ – конечное множество вершин, называемых позициями, $T=\{t\}$ – конечное множество вершин, называемых переходами, отношение смежности вершин $F\subseteq P\times T\bigcup T\times P$ – задаёт множество дуг, соединяющих позиции и переходы. Таким образом, сеть Петри представляет собой двудольный ориентированный граф [10,11], одну долю вершин которого составляют позиции, а другую – переходы.

Введём специальные обозначения множеств входных, выходных и смежных вершин для позиций и переходов сети:

$$^\bullet p=\{t\mid\exists(t,p)\in F\},\ \ p^\bullet=\{t\mid\exists(p,t\}\in F\},\ \ ^\bullet p^\bullet={}^\bullet p\cup p^\bullet,$$
$$^\bullet t=\{p\mid\exists\{p,t\}\in F\},\ \ t^\bullet=\{p\mid\exists(t,p)\in F\},\ \ ^\bullet t^\bullet={}^\bullet t\cup t^\bullet.$$

Аналогичным образом можно определить множества входных, выходных и смежных вершин для произвольного подмножества позиций либо переходов; такие обозначения будут использованы в дальнейшем изложении.

Сетью с входными и выходными позициями будем называть тройку $Z=(N,X,Y)$, где N – сеть Петри, $X\subseteq P$ – её входные позиции, $Y\subseteq P$ – её выходные позиции, причём множества входных и выходных позиций не пересекаются: $X\cap Y=\varnothing$. Входные и выходные позиции $C=X\cup Y$ называют *контактными*, а позиции $Q=P\setminus(X\cup Y)$ – *внутренними*.

Функциональной сетью будем называть сеть с входными и выходными позициями $Z=(N,X,Y)$, в которой входные позиции не имеют входящих дуг, а выходные позиции – исходящих: $\forall p\in X:{}^\bullet p=\varnothing$, $\forall p\in Y: p^\bullet=\varnothing$. В со-

ответствии с терминологией теории графов [11] входные позиции являются источниками, а выходные стоками; таким образом, произвольную сеть Петри можно рассматривать как функциональную. Функциональную сеть также будем обозначать $Z = (X, Q, Y, T, F)$, указывая структурные элементы сети Петри *N*.

Сеть Петри $N' = (P', T', F')$ является *подсетью* сети *N*, если

$$P' \subseteq P, T' \subseteq T, F' \subseteq F(P', T').$$

Подсетью, порождённой указанным множеством вершин $B(P', T')$ будем называть подсеть $N' = (P', T', F')$, где F' содержит все дуги, соединяющие вершины P', T' в исходной сети:

$$F' = \{(p,t) \mid p \in P', t \in T', (p,t) \in F\} \cup \{(t,p) \mid p \in P', t \in T', (t,p) \in F\}.$$

Подсетью, порождённой указанным множеством переходов $B(T')$ будем называть подсеть $B(P', T')$, где $P' = \{p \mid p \in P, \exists t \in T' : (t,p) \in F \vee (p,t) \in F\}$. Иными словами вместе с переходами T' сеть содержит все смежные им позиции и порождается этими вершинами.

Для краткости в обозначениях отношение смежности будем далее опускать, подразумевая исходное отношение смежности *F* вершин сети.

Подсеть $Z = B(R) = (X, Q, Y, R)$ сети Петри *N* будем называть *полной* в *N*, если в N выполняется $X^{\bullet} \subseteq R$, ${}^{\bullet}Y \subseteq R$, ${}^{\bullet}Q^{\bullet} \subseteq R$.

Функциональную сеть $Z = (N', X', Y')$ будем называть *функциональной подсетью* сети N и обозначать $Z \succ N$, если N' является подсетью N, и, кроме того, Z связана с оставшейся частью сети только посредством дуг, инцидентных контактным позициям таким образом, что входные позиции могут иметь только входящие дуги, а выходные – только исходящие:

$$\forall p \in X' : \{(p,t) \mid t \in T \setminus T'\} = \varnothing, \quad \forall p \in Y' : \{(t,p) \mid t \in T \setminus T'\} = \varnothing,$$

$$\forall p \in Q' : \{(p,t) \mid t \in T \setminus T'\} = \varnothing \wedge \{(t,p) \mid t \in T \setminus T'\} = \varnothing.$$

Сеть, являющуюся *разностью* исходной сети Петри и её функциональной подсети $Z'' = N - Z'$, определим следующим образом:

$$Z'' = (Y', P \setminus (X' \cup Y' \cup Q'), X', T \setminus T').$$

Функциональная подсеть $Z' \succ N$ является *минимальной* тогда и только тогда, когда она не содержит другую функциональную подсеть исходной сети Петри N.

Свойства функциональных подсетей и алгоритм декомпозиции

Перечислим основные *свойства функциональных подсетей*, изученные в [4-8]:

1. Функциональная подсеть порождается множеством своих переходов.
2. Множество минимальных функциональных подсетей определяет разбиение множества переходов на непересекающиеся подмножества.

3. Подсеть, порождённая множеством переходов полна в сети Петри, тогда и только тогда, когда она является функциональной подсетью.
4. Функциональная сеть является суммой (объединением) конечного числа минимальных функциональных подсетей.
5. Каждая контактная позиция в декомпозиции сети Петри имеет не более одной входящей минимальной функциональной подсети и не более одной исходящей минимальной функциональной подсети.
6. Сеть Петри инвариантна тогда и только тогда, когда инвариантны всё её минимальные функциональные подсети и существует общий ненулевой инвариант контактных позиций.
7. Фундаментальное уравнение сети Петри разрешимо тогда и только тогда, когда оно разрешимо для всех её минимальных функциональных подсетей и существует общее решение для контактных позиций.

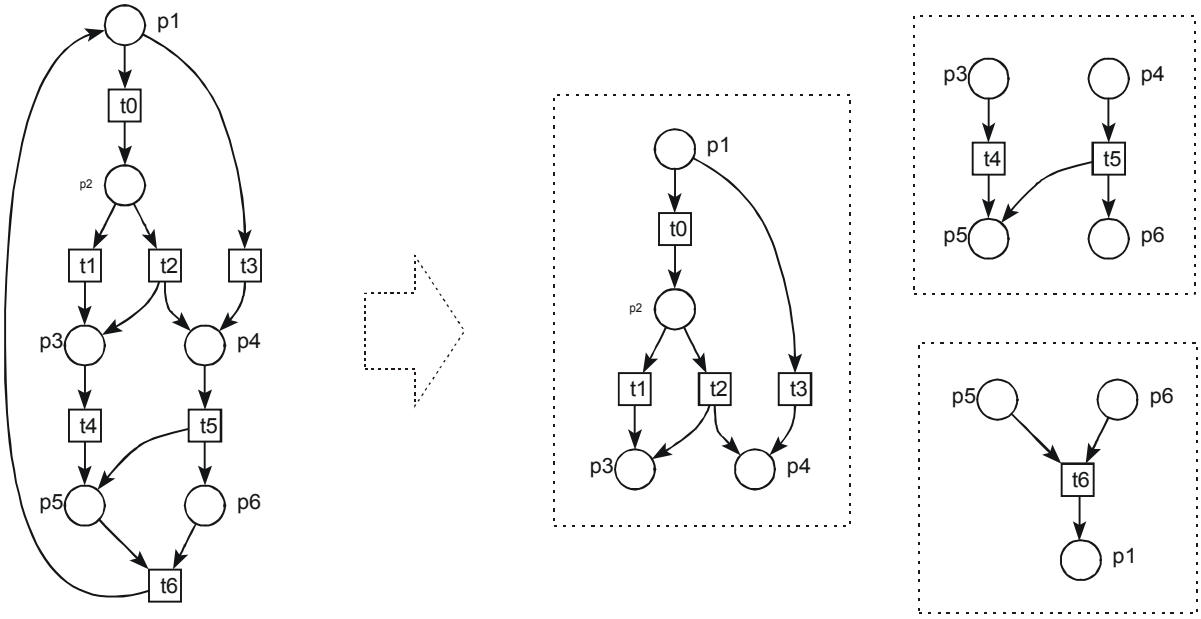

Рис. 1. Пример декомпозиции на функциональные подсети

Алгоритм декомпозиции заданной сети Петри на минимальные функциональные подсети представлен в [6,7] и состоит в выполнении следующих шагов:

Шаг 0. Выберем произвольный переход $t \in T$ сети N и включим его во множество избранных переходов $R := \{t\}$.

Шаг 1. Построим подсеть Z, порождённую множеством R:
$Z = B(R) = (X, Q, Y, R)$.

Шаг 2. Если Z полная в N, то Z искомая подсеть. Останов.

Шаг 3. Формируем множество поглощаемых переходов:

$$S = \{t \mid t \in X^{\bullet} \wedge t \notin R \vee t \in {}^{\bullet}Y \wedge t \notin R \vee t \in {}^{\bullet}Q^{\bullet} \wedge t \notin R\}.$$

Шаг 4. Полагаем $R := R \cup S$ и переходим на шаг 1.

В [7] доказано, что подсеть Z построенная описанным алгоритмом является минимальной функциональной подсетью сети Петри N; кроме того, алгоритм имеет полиномиальную сложность не превышающую $o(n^3)$, где n – количество вершин сети. Пример декомпозиции на функциональные сети представлен на Рис. 1.

Программный модуль декомпозиции

Программная реализация алгоритма декомпозиции требует выбора структур данных, обеспечивающих эффективную обработку информации. Традиционным является представление сети Петри с помощью матрицы инцидентности, либо, для сетей с петлями – парой матриц, задающими отдельно входящие и исходящие дуги переходов. Декомпозиция предназначена для обработки сетей большой размерности. Как правило, матрицы инцидентности таких сетей является разрежёнными. Стандартное представление разрежённых матриц использует массив (список) ненулевых элементов, соответствующий множеству дуг сети Петри. Например, возможно описание сети двумя массивами с элементами вида:

$$(np,\ nt),\ (nt,\ np),$$

где np – номер позиции, а nt – номер перехода; значение ненулевого элемента не указывается, так как для сетей без кратных дуг оно равно единице.

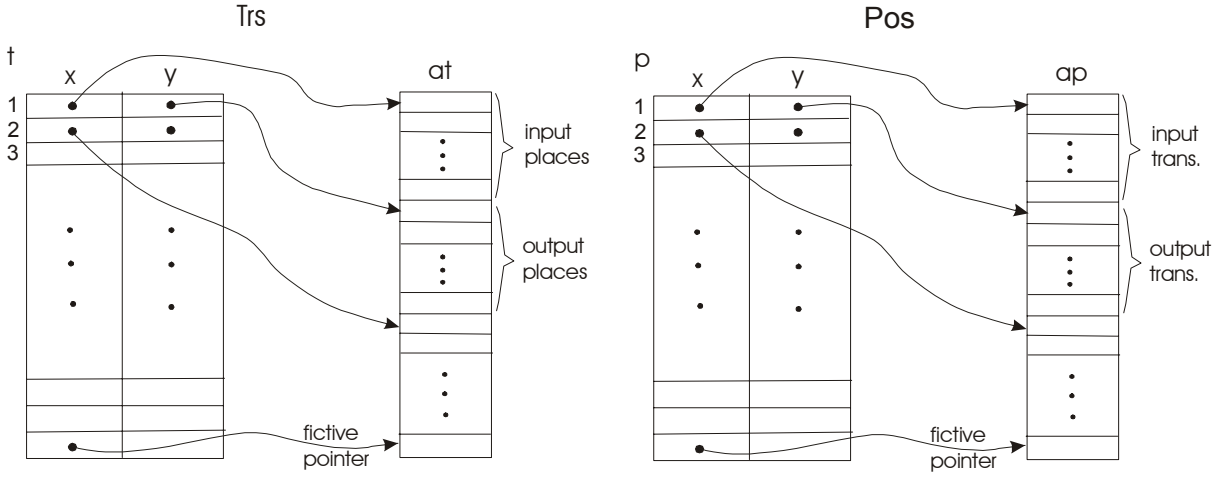

Рис. 2. Основные структуры данных

Рассмотрим операции, наиболее часто выполняемые алгоритмом декомпозиции. При построении порождённой подсети $B(R)$ на шаге 1 – это по-

иск входящих и исходящих позиций для переходов множества R: $X=\{p\,|\,p\in{}^{\bullet}R\wedge p\notin R^{\bullet}\}$, $Y=\{p\,|\,p\notin{}^{\bullet}R\wedge p\in R^{\bullet}\}$, $Q=\{p\,|\,p\in{}^{\bullet}R\wedge p\in R^{\bullet}\}$. При формировании множества поглощаемых переходов на шаге 2 – это поиск исходящих, входящих и смежных переходов для множеств X,Y,Q соответственно: $X^{\bullet}$, ${}^{\bullet}Y$, ${}^{\bullet}Q^{\bullet}$. Таким образом, основными операциями алгоритма являются операции нахождения множеств ${}^{\bullet}t$, $t^{\bullet}$, ${}^{\bullet}t^{\bullet}$, ${}^{\bullet}p$, $p^{\bullet}$, ${}^{\bullet}p^{\bullet}$.

Применение упорядоченных массивов входящих и исходящих дуг требует дополнительные массивы указателей, для быстрого обращения к элементам, описывающим дуги, инцидентные выбранной вершине. Поэтому, для представления сети Петри выбраны структуры данных, представленные на Рис. 2. Каждый из двух типов `Trs` и `Pos` является полным представлением сети Петри, но тип `Trs` обеспечивает быстрый доступ к дугам, инцидентным переходам, а тип `Pos` – к дугам, инцидентным позициям. Такое незначительное дублирование информации оправдано, так как позволяет избежать выполнения поиска информации при реализации основных операций алгоритма.

Рассмотрим более подробно тип данных `Trs`. Массив `t` индексируется номером перехода и содержит указатели на входные `x` и выходные `y` позиции. Инцидентные позиции для всех переходов располагаются в массиве `at`. При этом позиции следуют в таком порядке: сначала входные позиции первого перехода, затем выходные позиции первого перехода, затем входные позиции второго перехода и выходные позиции второго перехода и так далее. Таким образом, следующие операторы циклов

$p\in{}^{\bullet}t_i$ `: for( j=t[i].x; j<t[i].y; ++j ) {p=at[j];}`

$p\in t_i^{\bullet}$ `:for( j=t[i].y; j<t[i+1].x; ++j ) {p=at[j];}`

$p\in{}^{\bullet}t_i^{\bullet}$ `:for( j=t[i].x; j<t[i+1].x; ++j ) {p=at[j];}`

обеспечивают обработку входных, выходных и инцидентных позиций перехода с номером `i`. Аналогичным образом организован тип данных `Pos`. Замети, что для обеспечения описанных способов использования типов `Trs` и `Pos` необходима одна фиктивная последняя запись указателя `x`.

Выберем структуры данных для представления минимальных функциональных подсетей. Поскольку в соответствии со свойством 2 множество минимальных функциональных подсетей задаёт разбиение множества переходов на непересекающиеся подмножества, в качестве основного результата работы алгоритма будем рассматривать индикатор подсети, представленный массивом `SubNet` имеющим размерность множества переходов. Массив задаёт принадлежность переходов минимальным функцио-

нальным подсетям таким образом, что значение элемента SubNet[i] равняется номеру подсети, к которой принадлежит переход t_i. Подсети нумеруются начиная с единицы. В программной реализации дополнительная информация о декомпозиции представлена массивами SX и SY, являющимися индикаторами входных и выходных позиций соответственно.

```
SubNet=0; iSubNet=1;
цикл (построение подсетей)
    для (t∈T : SubNet[t]=0) положить { R={t}; SubNet[t]=iSubNet }
    цикл (поглощение переходов)
        Построение порождённой подсети B(R):
        X=Q=Y=∅
        для всех позиций p∈P:
            pinp=pout=false;
            для всех переходов t∈R:
                если p∈•t то pinp=true;
                если p∈t• то pout=true;
            если pinp=true & pout=true то Q=Q∪{p}
            если pinp=true то X=X∪{p}
            если pout=true то Y=Y∪{p}
        Построение множества поглощаемых переходов S:
        S=∅
        Проверка выходных переходов X:
        для всех позиций p∈X
            для всех переходов t∈p•
                если t∉R∧t∉S то { S=S∪{t}; SubNet[t]=iSubNet }
        Проверка входных переходов Y:
        для всех позиций p∈Y
            для всех переходов t∈•p
                если t∉R∧t∉S то { S=S∪{t}; SubNet[t]=iSubNet }
        Проверка инцидентных переходов Q:
            для всех позиций p∈Q
                для всех переходов t∈•p•
                    если t∉R∧t∉S то { S=S∪{t}; SubNet[t]=iSubNet }
        R=R∪S
    повторять пока S≠∅
    iSubNet++
повторять пока существует переход t∈T такой что iSubNet[t]=0
```

Рис. 3. Детализированный алгоритм декомпозиции

Кроме того, на текущем проходе алгоритма формируемая минимальная функциональная подсеть, представлена массивами `R, X, Q, Y`, соответствующими элементам определения функциональной подсети. А множество поглощаемых переходов – массивом `S`.

Детализированный алгоритм декомпозиции, реализуемый программным модулем DECO, представлен на Рис. 3. Модуль написан на языке ANSI C и насчитывает около 200 строк исходного текста. Проверена работоспособность модуля в средах Unix и Windows.

Интеграция модуля декомпозиции

Так как декомпозиция предназначена для ускорения анализа моделей большой размерности, основная программная реализация DEBORAH (www.geocities.com/zsoftua/softr.htm) использует интерфейс командной строки и текстовые форматы входных и выходных файлов:

```
>DEBORAH InpFile OutFile
```

где файл `InpFile` содержит исходное описание сети Петри, а в файл `OutFile` выводятся результаты декомпозиции.

Рис. 4 Алгоритм работы программы DEBORAH

Обобщённый алгоритм работы программы DEBORAH представлен на Рис. 4. Использование динамически распределяемой памяти обеспечивается двухпроходным сканированием исходного файла описания сети. На первом проходе определяется размер сети, затем выделяется память для данных, а на втором проходе осуществляется фактический ввод сети.

Формат исходного файла достаточно прост. Строки, начинающиеся с точки с запятой, рассматриваются как комментарии. Каждая строка данных перечисляет входные и выходные позиции очередного перехода, причём номера входных позиций указываются со знаком минус. Переходы нумеруются в порядке поступления строк.

Такой достаточно простой формат представления сети позволяет легко интегрировать программу DEBORAH в известные моделирующие системы, такие как Tina [9], CPN Tools [12]. При этом достаточно написать конвертор, преобразующий формат представления сетей моделирующей системы в формат DEBORAH. Описанный подход позволяет использовать графический редактор моделирующей системы для визуального представления сетей.

Конвертор для форматов сетей моделирующей системы Tina [9] включён в состав последней версии программы DEBORAH, таким образом, что добавления пункта `Decomposition` в меню `Tools` и его настройка на вызов программы DEBORAH позволяет интегрировать декомпозицию в систему Tina. Заметим, что Tina использует два формата представления сетей Петри: абстрактный (.net) и графический (.ndr). Абстрактный формат основан на представлении сети последовательностью описаний переходов; основное отличие состоит в использовании имён элементов сети. Графический формат дополнительно содержит координаты элементов сети на плоскости.

Стандартный текстовый файл результатов декомпозиции содержит индикаторы подсети для переходов, а также индикаторы входных и выходных позиций. Расширенный формат предполагает отдельный вывод подмножеств `X,Q,Y,R` для каждой подсети, и, кроме того, построение и вывод графа декомпозиции [7]. Дополнительной опцией является вывод подсетей в отдельные файлы. Использование такой опции совместно с графическим форматом позволяет получить визуальное представление подсетей.

Анализ опыта декомпозиция моделей реальных объектов

При отладке и тестировании программы DEBORAH выполнялась декомпозиция сравнительно небольших сетей Петри, описанных в литературе [13,14], насчитывающих десятки элементов. Пример декомпозиции представлен на Рис. 1.

Характеристики программы при работе с сетями большой размерности исследованы на специально генерированных случайных двудольных орграфах. Кроме того, выполнена декомпозиция моделей Петри таких реальных объектов как: телекоммуникационные протоколы [15,16] ECMA, TCP; химические реакции [17] апоптоза, метаболизма, гликолиза; процессы обработки транзакций в системе Transit [18] регистрации движения ценностей в странах Европейского союза и Европейского региона свободной торговли (EFTA).

Следует отметить, что случайные двудольные орграфы близки к неразделимым [7], в то время, как модели реальных объектов содержат хорошо локализованные связи и позволяют получить декомпозицию на полсети со средней размерностью около десятка вершин. Таким образом, полученная декомпозиция позволяет затем значительно ускорить вычисление инвариантов и решение фундаментального уравнения [4,5,8].

Заключение

Выполнена программная реализация алгоритма декомпозиции двудольных орграфов (сетей Петри) на функциональные подсети. Программа декомпо-

зиции DEBORAH может применяться как самостоятельно, так и совместно с известными моделирующими системами, такими как Tina, CPN Tools и предназначена для ускорения инвариантного анализа и процессов решения фундаментального уравнения сетей Петри.

Результаты декомпозиции множества сетей подтверждают предположение о хорошей разделимости моделей Петри реальных объктов.

Литература

1. Крывый С.Л. О некоторых методах решения и критериях совместимости систем линейных диофантовых уравнений в области натуральных чисел // Кибернетика и системный анализ, 1999, № 4, с. 12-36.
2. Зайцев Д.А. Теоретическое обоснование метода Тудика // Научные труды Донецкого государственного технического университета, серия "Информатика, кибернетика и вычислительная техника", Вып. 74, 2004, с. 286-293.
3. Зайцев Д.А. К вопросу о вычислительной сложности метода Тудика // Искусственный интеллект.- 2004, №1, с. 29-37.
4. Зайцев Д.А. Инварианты функциональных подсетей // Труды Одесской национальной академии связи им. А.С.Попова, №4, 2003, с. 57-63.
5. Zaitsev D.A. Decomposition-based calculation of Petri net invariants // Proceedings of Token based computing Workshop of the 25-th International conference on application and theory of Petri nets, Bologna, Italy, June 21-25, 2004, pp. 79-83.
6. Zaitsev D.A. Subnets with Input and Output Places // Petri Net Newsletter, Vol. 64, April 2003, pp. 3-6. Cover Picture Story.
7. Зайцев Д.А. Декомпозиция сетей Петри // Кибернетика и системный анализ, №5, 2004, с. 131-140.
8. Zaitsev D.A. Solving the fundamental equation of Petri net using the decomposition into functional subnets // 11th Workshop on Algorithms and Tools for Petri Nets, September 30 - October 1, 2004, University of Paderborn, Germany, p. 75-81.
9. Berthomieu B., Ribet O.-P., Vernadat F. The tool TINA - construction of abstract state space for Petri nets and Time Petri nets // International Journal of Production Research, Vol. 42, no. 4, 2004 (http://www.laas.fr/tina).
10. Мурата Т. Сети Петри: Свойства, анализ, приложения // ТИИЭР, т. 77, №4, 1989, с. 41-85.
11. Татт У. Теория графов. М.: Мир, 1988, 424с.
12. Beaudouin-Lafon M., Mackay W.E., Jensen M. et al. CPN Tools: A Tool for Editing and Simulating Coloured Petri Nets // LNCS 2031: Tools and Algorithms for the Construction and Analysis of Systems, 2001, 574-580 (http://www.daimi.au.dk/CPNTools).

13. Girault C., Volk R. Petri nets for systems engineering – A guide to modelling, verification and applications, Springer-Verlag, 2003.
14. Cortadella J., Kishinevsky M., Kondratyev A., Lavagno L., Yakovlev A. Logic synthesis of asynchronous controllers and interfaces, Springer-Verlag, 2002.
15. Zaitsev D.A. Verification of Protocol ECMA with Decomposition of Petri Net Model // Proceedings of The International Conference on Cybernetics and Information Technologies, Systems and Applications, Orlando, Florida, USA, July 21-25, 2004.
16. Zaitsev D.A. Verification of protocol TCP via decomposition of Petri net model into functional subnets // Proceedings of the Poster session of 12th Annual Meeting of the IEEE / ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, October 5-7, 2004, Volendam, Netherlands, p. 73-75.
17. Heiner M., Koch I. Petri net based model validation in systems biology // Proceedings of the 25-th International conference on application and theory of Petri nets, Bologna, Italy, June 21-25, 2004, p. 216-237.
18. Verbeek E., Van der Torn R. Transit case study // Proceedings of the 25-th International conference on application and theory of Petri nets, Bologna, Italy, June 21-25, 2004, p. 392-410.

Опубликовано: Научные труды Донецкого государственного технического университета, серия "Информатика, кибернетика и вычислительная техника", Вып. 93, 2005, с. 60-70.