Российская академия наук
Сибирское отделение
Институт систем информатики
им. А. П. Ершова

В.Е. Козюра

РЕАЛИЗАЦИЯ СИСТЕМЫ ПРОВЕРКИ МОДЕЛЕЙ РАСКРАШЕННЫХ СЕТЕЙ ПЕТРИ С ИСПОЛЬЗОВАНИЕМ РАЗВЕРТОК

Препринт
94

Новосибирск 2002

В работе описывается реализация метода проверки моделей раскрашенных сетей Петри с применением разверток. Описанная система проверки моделей является прототипной реализацией и требует выполнения некоторых операций вручную. Основной задачей описанной в работе реализации было показать принципиальную возможность использования данного метода на практике и дать несколько примеров верификации моделей распределенных систем. В качестве примеров были рассмотрены: РСП, представляющая задачу об обедающих философах, РСП, описывающая модель однобитового коммуникационного протокола, и РСП, представляющая модель коммуникационного протокола “Отправитель—Получатель”. Данные примеры были верифицированы для свойств “прогресса” и “безопасности”.

Siberian Division of the Russian Academy of Sciences
A. P. Ershov Institute of Informatics Systems

Kozura V.E.

IMPLEMENTATION OF THE MODEL CHECKING ALGORITHM FOR COLOURED PETRI NETS BASED ON NET UNFOLDINGS

Preprint
94

Novosibirsk 2002

The present paper describes the implementation of the model checking method for coloured Petri nets (CPN) based on net unfoldings. The considered system is a prototype implementation and contains some steps made manually. The main purpose of this work was to show the possibility of applying the model checking procedure based on net unfoldings for CPN to some practical models of distributed systems. As the examples, the CPN representing the dining philosophers problem, an Alternating Bit Protocol and the "Producer—Consumer" system have been considered. These examples are verified using the progress and safety properties as the logical specifications.

1. ВВЕДЕНИЕ

Верификация распределенных систем является важным направлением теории современного программирования. Естественный подход к проблеме верификации состоит в моделировании распределенных систем конечными автоматами или сетями Петри и в верификации полученных моделей. Среди наиболее важных подходов к верификации сетей Петри можно выделить симуляцию работы сети и анализ ее пространства состояний. При симуляции работа сети изучается пошаговым методом в ручном или полуавтоматическом режимах. Хотя многие ошибки в работе сети могут быть обнаружены на этапе симуляции, доказательная проверка корректности работы сети может быть получена для данного подхода только при полном моделировании работы сети и проведении так называемой проверки моделей. Проверка моделей заключается в описании требуемых свойств системы на языках логических спецификаций и доказательстве истинности или ложности данных спецификаций на построенном пространстве состояний системы.

К сожалению, при полном моделировании работы сети мы сталкиваемся с так называемой “проблемой взрыва числа состояний” (state explosion problem). Эта проблема состоит в том, что при росте размеров рассматриваемой сети ее полная модель достаточно быстро становится необозримо большой. Это не позволяет надеяться на построение полной модели для реальных протоколов. Отдельным и, как это следует из вышесказанного, достаточно важным направлением верификации распределенных систем является разработка методов, направленных на решение проблемы взрыва числа состояний. Среди таких методов можно выделить следующие: метод упрямых множеств (stubborn set method), использование двоичных разрешающих диаграмм (symbolic binary decision diagrams (BDD)), методы, основанные на частичном порядке (partial order methods), а также использование симметрии и эквивалентности на рассматриваемых моделях [37].

Рассмотренный в данной работе подход относится к методам, основанным на использовании частичного порядка. В качестве модели при проверке логических спецификаций вместо полного графа достижимости используется так называемая развертка сети Петри. Развертки сетей Петри являются относительно новым и интенсивно развивающимся направлением в области верификации сетей Петри. Данный метод позволяет во многих случаях существенно уменьшить размер модели, не теряя при этом свойств рассматриваемой сети. Библиография этого подхода постоянно расширяет-

ся, добавляются все новые возможности использования разверток при верификации сетей Петри.

Среди различных расширений стандартных сетей Петри выделяются сети Петри высокого уровня — раскрашенные сети Петри (РСП) [20, 21], для которых развит теоретический аппарат, накоплен значительный опыт использования и реализована система симуляции и анализа Design/CPN [21]. В РСП вместо обычных фишек используются типизированные знаковые элементы. Также есть возможность задания функций на дугах и переходах. Это расширение позволяет выражать в достаточно компактном виде сложные системы, не теряя при этом возможностей применять методы, разработанные для стандартных сетей Петри.

Данная работа посвящена применению метода проверки моделей (с использованием в качестве моделей разверток) к раскрашенным сетям Петри.

Использование разверток для анализа сетей Петри было рекомендовано МакМилланом в работе [32]. Предлагается использовать конечный префикс максимального ветвящегося процесса вместо полного графа достижимости. Размер развертки является экспоненциальным в общем случае, и в последующих работах были предложены некоторые улучшения определений и алгоритмов построения развертки [12, 30]. Так в работе [12] рекомендуется рассматривать обобщенное понятие порядка на конфигурациях; в результате был предложен критерий финитизации, являющийся оптимальным для одно-безопасных сетей Петри. В работе [30] было показано, что для n-безопасных сетей Петри данный критерий не является оптимальным и было предложено существенное улучшение критерия для n-безопасных сетей. Алгоритм МакМиллана построения развертки является в общем случае экспоненциальным по размеру получаемой развертки. В работе [30] также был предложен алгоритм построения развертки, линейно зависящий от произведения числа мест и переходов развертки.

Первоначально МакМиллан предложил использовать развертки для анализа достижимости и обнаружения тупиков. Эти методы были улучшены в последующих работах [18, 34]. В работе [18] предложено улучшение метода определения достижимости данного состояния сети. В работе [34] дается метод обнаружения тупиковых состояний с использованием системы неравенств, описывающих данную сеть Петри. Дж. Эспарца в работе [11] предложил метод проверки моделей для одно-безопасных сетей Петри и логики S_4 с использованием развертки. В работе [7] для сетей Петри с интервальным временем была построена развертка и применен алгоритм проверки моделей Эспарцы. В работе [40] Ф. Валнер предложил алгоритм проверки моделей для LTL логики, основанный на развертке сети Петри. В его рабо-

те известный теоретико-автоматный подход к верификации LTL формул [39] был перенесен на одно-безопасные сети Петри без тупиковых состояний. Этот подход был развит в последующих работах [9, 14, 15, 19]. На последнем этапе алгоритма Валнера строится дополнительный граф, состоящий из точек сечения рассматриваемой развертки. В работах [14, 15] была показана возможность модификации алгоритма, позволяющая избежать построения данного дополнительного графа, и описана реализация данного подхода с приведением экспериментальных результатов. Работы [9, 19] посвящены теоретическим аспектам методов проверки моделей с использованием разверток. В работе [19] были доказаны некоторые сложностные оценки алгоритмов проверки моделей с использованием разверток. Работа [6] посвящена рассмотрению возможностей применения методов развертки к неограниченным сетям Петри. Стоит также отметить работу [13], в которой метод развертки был применен к синхронизированному произведению двух сетей Петри.

Использование развертки РСП без применения критериев финитизации в общем случае для метода упрямых множеств было рассмотрено в работе [38]. В данной работе помимо отсутствия финитизации также использовалось старое описание раскрашенных сетей Петри. Все это заставило автора сделать вывод о неэффективности использования разверток РСП, и в дальнейших работах он описал метод упрямых множеств для РСП без построения развертки. Развертки РСП не рассматривались более в литературе. В то время как для стандартных сетей Петри были развиты критерии финитизации, эффективные алгоритмы построения, техника работы со свойствами достижимости и тупиками и алгоритмы проверки моделей для логики LTL, возможность применения данной техники к РСП оставалась открытым вопросом.

В работах [23, 24, 25] методы разверток были применены к раскрашенным сетям Петри, описанным в работах [20, 21]. В этих работах приведены алгоритмы построения разверток РСП и оценки сложностей данных алгоритмов. Показано, как применять методы обнаружения тупиков, описанные в [34], к РСП. Кроме того, методы работы с развертками интервально-временных СП [7] формально адаптированы к интервально-временным РСП в работах [26, 27]. Следующим шагом была адаптация методов проверки моделей, развитых для стандартных сетей Петри, к РСП. В работе [28] описана основанная на состояниях семантика LTL для РСП и алгоритм проверки моделей для логики LTL и РСП. Корректность полученного алгоритма формально доказана. Также показана возможность применения данного алгоритма к РСП с интервальным временем.

Данная работа описывает реализацию метода проверки моделей раскрашенных сетей Петри. Описанная система проверки моделей с использованием разверток является прототипной реализацией и требует выполнения некоторых операций вручную. Целью проведенных экспериментов не являлось сравнение данной реализации с классическими системами проверки моделей. Основной задачей данной работы было показать принципиальную возможность использования данного метода на практике и дать несколько примеров верификации распределенных систем. В качестве примеров были рассмотрены: РСП, представляющая задачу об обедающих философах, сеть, описывающая модель однобитового коммуникационного протокола и сеть, представляющая модель коммуникационного протокола "Отправитель—Получатель". Данные примеры были верифицированы для свойств "прогресса" и "безопасности".

2. РАЗВЕРТКИ СЕТЕЙ ПЕТРИ

2.1. Ветвящиеся процессы

Цель данной главы — определить понятие развертки стандартной сети Петри (СП), являющейся символическим представлением графа достижимости сети Петри. Развертка определяется как конечный префикс максимального ветвящегося процесса СП. Для определения максимального ветвящегося процесса нам необходимо определение О-сети, являющейся ограниченным вариантом стандартной СП. Мы будем считать известными определения сети Петри, разметки СП, а также механизмы функционирования стандартной СП.

Определение 2.1. Разметка M сети Петри $\boldsymbol{N} = (P, T, N)$ называется *n-безопасной*, если $M(p) \le n \quad \forall p \in P$. Сеть Петри $\boldsymbol{N}$ называется *n-безопасной*, если все ее достижимые разметки являются n-безопасными. Однобезопасные сети Петри также называются *безопасными*.

Определение 2.2. *Пре-множеством* вершины $x \in P \cup T$, обозначаемым ${}^\bullet x$, называется множество

$${}^\bullet x = \{y \in P \cup T \mid N(y, x) = 1\}.$$

Пост-множеством вершины $x \in P \cup T$, обозначаемым $x^\bullet$, называется множество

$$x^\bullet = \{y \in P \cup T \mid N(x, y) = 1\}.$$

Определение 2.3. Вершины x_1 и x_2 находятся в конфликте, обозначаемом $x_1 \# x_2$, если существуют переходы t_1 и t_2 такие, что ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \varnothing$, и (t_1, x_1) и (t_2, x_2) принадлежат транзитивному замыканию N (которое мы будем обозначать $\mathbf{R_t}$). Вершина x находится в самоконфликте, если x#x.

Мы будем писать $x_1 \leq x_2$, если $(x_1, x_2) \in \mathbf{R_t}$ и $x_1 < x_2$, если $x_1 \leq x_2$ и $x_1 \neq x_2$.

Определение 2.4. Будем обозначать x *co* y или x || y, если неверно ни одно из следующих условий x < y, x > y, x#y.

Определение 2.5. *О-сетью* называется сеть Петри $\boldsymbol{N}$ = (P, T, N), где функция инцидентности $N \subseteq P \times T \cup T \times P$ удовлетворяет следующим условиям:

(a) $\forall p \in P \quad |{}^\bullet p| \leq 1$;

(b) N — ациклично, т.е. иррефлексивное и транзитивное замыкание N является частичным порядком;

(c) $\forall x \in P \cup T$ множество $\{y \in P \cup T \mid y \leq x\}$ является конечным, что дает нам существование Min(*N*) (множества минимальных элементов *N* по отношению к R_t). Min(*N*) считается состоящим только из мест сети *N*;

(d) ни один из переходов не находится в самоконфликте.

Начальная разметка О-сети M_0 определяется следующим образом: $M_0(p) = 1$, если $p \in \mathrm{Min}(\boldsymbol{N})$, и $M_0(p) = 0$ иначе.

Утверждение 2.1. О-сеть является одно-безопасной сетью Петри.

Определение 2.8. Пусть даны две сети Петри $\boldsymbol{N_1} = (P_1, T_1, N_1)$ и $\boldsymbol{N_2} = (P_2, T_2, N_2)$. *Гомоморфизм* h из $\boldsymbol{N_2}$ в $\boldsymbol{N_1}$ — это отображение

$$h: P_2 \cup T_2 \to P_1 \cup T_1$$

такое, что

(a) $h(P_2) \subseteq P_1$ и $h(T_2) \subseteq T_1$;

(b) $\forall t \in T_2 \; h|_{{}^\bullet t} = {}^\bullet t \to {}^\bullet h(t)$, $\forall t \in T_2 \; h|_{t^\bullet} = t^\bullet \to h(t)^\bullet$.

Определение 2.9. *Ветвящийся процесс* сети Петри $\boldsymbol{N}$ = (P, T, N) — это пара $\beta = (\boldsymbol{N'}, h)$, где $\boldsymbol{N'} = (P', T', N')$ является сетью-процессом, и h — это гомоморфизм из $\boldsymbol{N}$ в $\boldsymbol{N'}$ такой, что ограничение h на Min($\boldsymbol{N'}$) является биекцией между Min($\boldsymbol{N'}$) и M_0, и $\forall t_1, t_2 \in T'$ из ${}^\bullet t_1 = {}^\bullet t_2$ и $h(t_1) = h(t_2)$ следует $t_1 = t_2$.

В работе [10] было показано, что для каждой сети Петри существует единственный максимальный ветвящийся процесс с точностью до гомоморфизма. Максимальный ветвящийся процесс сети $\boldsymbol{N}$ будем обозначать МВР($\boldsymbol{N}$).

2.2. Определения разверток сетей Петри

В данном разделе вводятся понятия конфигурации, локальной конфигурации, точек сечения, а также понятие развертки сети Петри, являющейся конечным префиксом максимального ветвящегося процесса, который в общем случае может быть бесконечным. Такое понятие развертки было введено МакМилланом в работе [32], и там же был предложен первый критерий финитизации.

Определение 2.10. *Конфигурацией* С О-сети $\boldsymbol{N} = (P, T, N)$ называется множество переходов, удовлетворяющих следующим условиям:

(1) $t \in C \Rightarrow \forall\, t_0 \le t :\ t_0 \in C$

(2) $\forall t_1, t_2 \in C : \neg(t_1 \# t_2)$.

Определение 2.11. Множество $X_0 \subseteq X$ вершин называется *ко-множеством*, если $\forall t_1, t_2 \in X_0$: $(t_1 \text{ co } t_2)$.

Определение 2.12. Множество $X_0 \subseteq X$ вершин называется *сечением*, если оно является максимальным ко-множеством по отношению к операции включения множеств.

Конечные конфигурации О-сети и сечения тесно связаны. Пусть С является конечной конфигурацией. Тогда $\mathrm{Cut}(C) = (\mathrm{Min}(\boldsymbol{N}) \cup C^\bullet) \setminus {}^\bullet C$ является сечением.

Определение 2.13. Пусть дана СП $\boldsymbol{N_1} = (P_1, T_1, N_1)$ и МВР($\boldsymbol{N_1}$) = ($\boldsymbol{N_2}$, h), где $\boldsymbol{N_2} = (P_2, T_2, N_2)$, является максимальным ветвящимся процессом сети $\boldsymbol{N_1}$. Пусть С — это конфигурация сети $\boldsymbol{N_2}$. Мы определим разметку *Mark(C)* = Cut(C) сети $\boldsymbol{N_1}$ следующим образом: $\mathrm{Mark}(C) = h((\mathrm{Min}(\boldsymbol{N_2}) \cup C^\bullet) \setminus {}^\bullet C)$.

Определение 2.14. Пусть $\boldsymbol{N}$ является О-сетью. Для перехода $t \in T$ конфигурация $[t] = \{t' \in T \mid t' \le t\}$ называется *локальной конфигурацией*. Тот факт, что [t] является конфигурацией, можно легко проверить.

Определение 2.15. Переход $t \in T$ из СП называется *GT_0-точкой сечения*, если существует $t_0 \in T$ такой, что $\mathrm{Mark}([t]) = \mathrm{Mark}([t_0])$ и $[t_0] \subset [t]$.

Определение 2.16. Переход $t \in T$ из СП называется *GT-точкой сечения*, если существует $t_0 \in T$ такой, что $\mathrm{Mark}([t]) = \mathrm{Mark}([t_0])$ и $|[t_0]| < |[t]|$.

Определение 2.17. Переход $t \in T$ из СП называется *EQ-точкой сечения*, если существует $t_0 \in T$ такой, что

(1) $Mark([t]) = Mark([t_0])$,

(2) $|[t_0]| = |[t]|$,

(3) $\neg(t \parallel t_0)$,

(4) не существует EQ-точки сечения среди t' таких, что $t' \parallel t_0$ и $|[t']| \leq |[t_0]|$.

Определение 2.18. Для СП N *развертка* получается из максимального ветвящегося процесса после удаления всех переходов t' таких, что существует точка сечения t и $t < t'$, а также всех мест $p \in t'^{\bullet}$. При использовании GT_0(GT)-точек сечения полученную развертку будем называть *GT_0(GT)-развертка* или разверткой МакМиллана. Если в качестве точек сечения возьмем объединение GT-точки сечения $\cup$ EQ-точки сечения, то полученная развертка называется *EQ-развертка*.

Относительно данных видов разверток были доказаны свойства конечности, безопасности и полноты [12, 32]. Первое означает, что все полученные развертки для n-безопасных сетей Петри являются конечными. Свойство безопасности наследуется от свойства безопасности максимального ветвящегося процесса и означает отсутствие в развертке конфигураций, не соответствующих никакой достижимой разметке исходной сети. Свойство полноты означает, что для каждой достижимой разметки исходной сети существует конфигурация в развертке, соответствующая данной разметке. Таким образом, мы можем использовать развертку сети Петри для проверки свойства достижимости разметок данной сети Петри.

Было показано [12], что в некоторых случаях развертки МакМиллана неэффективны, и тогда лучше применять другие критерии финитизации. В случае развертки МакМиллана полученный префикс может иметь экспоненциальный рост, в то время как минимальный конечный префикс растёт только линейно.

В данной работе мы не будем рассматривать алгоритмы построения разверток для стандартных сетей Петри. Обзор имеющихся алгоритмов можно найти, например, в [30]. В работе [23] можно найти подробное описание алгоритмов построения разверток РСП, а также описание сложностей построения данных разверток.

3. МЕТОДЫ ПРОВЕРКИ МОДЕЛЕЙ

Среди методов верификации в последние десятилетия одно из ведущих мест занимают методы проверки модели. Грубо говоря, проверка моделей является процедурой, решающей для заданной модели M и логической спецификации φ, верна ли данная спецификация в рассматриваемой модели или нет. Истинность данной спецификации обозначается как $M \models \varphi$. Модель M является, как правило, автомато-подобной системой, моделирующей требуемые свойства системы, а логическая спецификация φ — формулой временной модальной логики. Методы проверки моделей делятся на два типа.

1. Локальная проверка моделей. По данной формуле φ и данному участку (состоянию) s системы необходимо определить, является ли формула φ истинной на участке s или нет.
2. Глобальная проверка моделей. По данной формуле φ и описанию всей системы M необходимо определить те участки (состояния) системы M, в которых формула φ является истинной.

Методы проверки моделей предоставляют полностью автоматический способ решения данных задач. В совокупности с известными, достаточно эффективными алгоритмами для логик с сильной выразительной силой, методы проверки моделей дают нам эффективный подход к верификации распределенных систем.

3.1. Логика линейного времени

В этом разделе мы дадим основные определения логики линейного времени, используемой в работе, а именно LTL-логики. Пусть некоторое множество **R** считается множеством *атомарных высказываний*.

Определение 3.1. Множество LTL-формул индуктивно определяется следующим образом:

- если $\varphi = a \in \mathbf{R}$, то φ — формула;
- если φ и ψ — формулы, то $\varphi \vee \psi$, $\varphi \wedge \psi$, $\neg\varphi$, $X\varphi$, $\varphi U \psi$ также являются формулами.

Другие пропозициональные операторы, такие как "$\rightarrow$" и "$\leftrightarrow$", определяются стандартным образом. Например $\varphi \rightarrow \psi \equiv \varphi \wedge \neg \psi$. Дополнительно можно определить еще два оператора: $\Diamond\varphi \equiv \text{true } U\ \varphi$, $\Box\varphi \equiv \neg\Diamond\neg\varphi$.

Множество атомарных высказываний, содержащихся в формуле φ, обозначается ⟨φ⟩.

Формулы LTL интерпретируются на *w-словах* над алфавитом $2^{\mathbf{R}}$.

Определение 3.2. *W-словом* γ над алфавитом $2^{\mathbf{R}}$ называется бесконечная последовательность $\gamma = a_0a_1a_2...$, где для всех i $a_i \in 2^{\mathbf{R}}$.

Высказывание $a \in \mathbf{R}$ верно в a_i, если $a \in a_i$. Обозначим через $\gamma \models \varphi$ тот факт, что w-слово $\gamma = a_0a_1a_2...$ удовлетворяет φ. Индуктивно это можно определить следующим образом:

$\gamma \models \varphi$, если φ — атомарное высказывание и $\varphi \in \gamma(0)$;
$\gamma \models \neg\varphi$, если не верно, что $\gamma \models \varphi$;
$\gamma \models \varphi \vee \psi$, если $\gamma \models \varphi$ или $\gamma \models \psi$;
$\gamma \models X\varphi$, если $\gamma(1) \models \varphi$;
$\gamma \models \varphi U \psi$, если $\exists i \geq 0$ такое, что $\gamma(i) \models \psi$ и $\gamma(j) \models \varphi \ \forall j<i$.

Множество w-слов, удовлетворяющих φ, обозначим L_φ.

Ниже будет описан алгоритм проверки моделей для LTL формул для стандартных одно-безопасных сетей Петри с использованием разверток в качестве моделей. Чтобы проводить такую проверку, нам необходимо иметь семантику LTL формулы на сетях Петри, которая будет определена ниже. Пусть дана СП $\boldsymbol{N}$ = (P, T, F). Мы будем рассматривать семантику, основанную на состояниях, в которой множество атомарных высказываний ассоциируется с множеством мест сети P. Высказывание $p \in P$ верно в состоянии M, если M(p) = 1. Последовательность $M_0 \rightarrow^{t0} M_1 \rightarrow^{t1} M_2...$ удовлетворяет φ, если w-слово $M_0M_1M_2...$ принадлежит L_φ. Разметка M удовлетворяет φ, если любая последовательность срабатываний $M \rightarrow^{t0} M_1 \rightarrow^{t1} M_2...$ удовлетворяет φ. В этом случае будем писать $M \models \varphi$. РСП $\boldsymbol{N}$ удовлетворяет φ, если M_0 удовлетворяет φ. В этом случае будем писать $\boldsymbol{N} \models \varphi$.

В дальнейшем в соответствии с работами [15, 22, 40] мы будем использовать фрагмент LTL′ логики LTL, в котором нет оператора X. В этом фрагменте не возможно различить слова $\xi = x_0x_1x_2...$ и слово ξ′, полученное из ξ с помощью повторения конечное число раз некоторых x_i. Такие два слова будут названы *статтерно-эквивалентными*.

Лемма 3.1 [31]**.** Если φ является LTL′-формулой и ξ и ξ′ — два статтерно-эквивалентных слова, тогда $\xi \models \varphi \Leftrightarrow \xi' \models \varphi$.

3.2. Алгоритм проверки моделей для LTL логики

В данной части работы будет приведен алгоритм проверки моделей для логики LTL, основанный на представлении как модели, так и формулы в виде конечного автомата (так называемый теоретико-автоматный подход). Согласно теоретико-автоматному подходу модель представляется в виде так называемой помеченной системы переходов, а в качестве представления LTL-формулы рассматриваются автоматы Бюхи. Ниже мы приводим соответствующие определения и основную идею алгоритма проверки моделей. Более подробное описание может быть найдено в [40]. В работе [40] также было предложено использовать данный подход при верификации стандартных сетей Петри на их представлении в виде развертки. Этот подход будет также описан в данной части работы. Вначале опишем теоретико-автоматный подход к проверке формул LTL логики.

Определение 3.3. *Помеченной системой переходов (labeled transition system (LTS))* называется тройка $(S, Act, \rightarrow)$, где S — конечное множество состояний, Act — множество действий и $\rightarrow \subseteq S \times Act \times S$ — функция инцидентности. Если $(s, a, s') \in \rightarrow$ (обозначим это как $s \rightarrow^a s'$), то мы будем говорить, что автомат может перейти из состояния s в состояние s′ посредством срабатывания действия a.

Определение 3.4. *Автоматом Бюхи* над алфавитом 2^R называется четверка $A = (Q, q_0, \delta, F)$, где Q — это конечное множество состояний, q_0 — начальное состояние, $\delta \subseteq Q \times 2^R \times Q$ — функция переходов, и $F \subseteq Q$ — множество *допустимых состояний.*

Определение 3.5. *Последовательностью срабатываний* автомата A на w-слове ξ называется бесконечная последовательность $\sigma = qq_1q_2...$ такая, что $\forall i \leq 0 \ (q_i, \xi(i), q_{i+1}) \in \delta$.

Последовательность срабатываний называется *допустимой*, если *допустимые состояния* встречаются в σ бесконечно часто. Автомат Бюхи A *допускает* слово ξ, если существует допустимая последовательность срабатываний A на w-слове ξ.

Пусть L(A) обозначает множество w-слов, допускаемых A.

Теорема 3.1 [39]. Пусть φ является LTL-формулой. Существует автомат Бюхи A_φ такой, что $L(A_\varphi) = L_\varphi$.

Существуют эффективные методы построения автомата A_φ по данной формуле φ [16].

Если дана формула φ и система, заданная автоматом A над алфавитом 2^R, согласно теоретико-автоматному подходу строится произведение A и автоматной системы $A_{\neg\varphi}$. Основная идея состоит в том, что переход

$$(\langle a, b\rangle, \langle a', b'\rangle) \Leftrightarrow (a, T, a') \in \delta(A) \text{ и } (b, T, b') \in \delta(A_{\neg\varphi}).$$

Для этой конструкции верно следующее утверждение: отсутствие в произведении циклов, содержащих допускающее состояние, эквивалентно истинности φ для автомата A. Подробное описание данного подхода можно найти в работе [16].

Ф. Валнер [40] предложил использовать теоретико-автоматный подход для проверки формул LTL логики на одно-безопасных сетях Петри с использованием их разверток в качестве модели. Семантика LTL формулы для СП была описана выше. Автомат $A_{\neg\varphi}$ рассматривается в качестве одно-безопасной сети Петри следующим образом: множество состояний автомата будем рассматривать как множество мест. Каждую дугу автомата будем рассматривать как снабженную переходом сети. В начальной разметке A_0 единственная точка находится в месте q_0. Каждый переход рассматривается, как помеченный одно-безопасной разметкой M, включающей только места из $\langle\varphi\rangle$. Аналогично теоретико-автоматному подходу строится произведение системной СП и полученной автоматной сети.

Пусть задана одно-безопасная СП $\boldsymbol{N} = (P, T, F)$ и автоматная сеть $A_{\neg\varphi} = (Q, q_0, \delta, F)$. Произведение строится таким образом, чтобы срабатывания системной и автоматной сетей происходили поочередно. Для каждого места $p \in P$ такого, что $p \in \langle\varphi\rangle$, добавим *комплиментарное* место p'. Это делается для контроля разметок системной сети (см [40] для более подробного объяснения). Для p' будем считать $M(p') = 1 - M(p)$. Множество комплиментарных мест обозначим P'. Множество $Obs(\varphi) = \{p, p' \mid p \in \langle\varphi\rangle\}$ называется множеством *обозреваемых (обозримых)* мест. Добавим дополнительные места s_f и s_s для организации поочередного срабатывания сетей **N** и $A_{\neg\varphi}$.

Полученное множество мест: $P \cup P' \cup Q \cup \{s_f, s_s\}$.

Множество переходов: $T \cup \delta$.

Кроме имеющихся в сетях дуг, отношение F в полученном произведении расширяется с помощью следующих конструкций:

1) добавляются дуги (s_s, t), (t, s_f) $\forall t \in T$ и (s_f, d), (d, s_s) $\forall d \in \delta$;
2) добавляются дуги (q, d), (d, q') $\forall\ d = (q, P', q') \in \delta$, (p, d), (d, p) $\forall p \in P'$ и (r', d), (d, r') $\forall r \in \langle \varphi \rangle \setminus P'$.

Для данного произведения строится развертка и ищутся циклы, содержащие допустимые места в автоматной сети. В работе [40] было доказано, что наличие такого цикла соответствует пути в системной сети, удовлетворяющего $\neg\varphi$.

Проблема нахождения циклов решается в два этапа.

Во-первых, строится ориентированный граф G = (V, Edg), где V = Off является множеством точек сечения и Edg множеством дуг. Наличие в графе G дуги $e \to e'$ означает достижимость состояния [e′] из состояния [e]. Некоторые дуги в графе помечаются символом “a”. Это означает, что на пути из [e] в [e′] встречается допустимое место. Вторым шагом применяем стандартный алгоритм на G по поиску сильно связанных компонент или циклов, содержащих хотя бы одну дугу, помеченную символом “a”.

Как было замечено в [40], строго последовательное поведение полученного произведения сводит на нет все преимущества использования методов частичного порядка, в частности, методов развертки. Там же было предложено ограничиться рассмотрением только *статтерно-инвариантных* свойств, т.е. свойств, описываемых в логике LTL′. В данном случае достаточно рассматривать только *видимые* переходы, т.е. переходы, имеющие смежные с обозреваемыми местами дуги. При построении произведения системной и автоматной СП это означает, что мы будем проводить синхронизацию только видимых переходов системной СП с переходами автоматной СП. При таком построении существует возможность того, что некоторые последовательности срабатываний системной СП удовлетворяют $\neg\varphi$, но соответствующие пути в произведении не являются допускаемыми. Это происходит в том случае, когда рассматриваемая последовательность содержит только конечное число видимых переходов. Тем не менее, в работе [40] был предложен алгоритм для проверки формул LTL' логики. Алгоритм состоит из двух шагов.

Шаг 1. Имея произведение системной СП и автоматной сети $A_{\neg\varphi}$, мы строим развертку и граф T = (Off, Edge), как это было описано выше. Дополнительно некоторые дуги помечаются символом “b”. $e \to^b e'$ означает, что на пути из [e] в [e′] сработал некоторый переход из $A_{\neg\varphi}$. Т. е., дуги графа могут быть помечены символами “a”, “b” или “a” и “b” одновременно.

Теперь мы можем применять стандартные алгоритмы нахождения максимальных сильно связанных компонентов (scc). Если мы нашли хотя бы одну scc, содержащую переход помеченный символом “a”, мы имеем допустимую последовательность срабатываний в автоматной сети и можем восстановить последовательность срабатываний системной РСП, удовлетворяющую $\neg\varphi$.

Шаг 2. Удаляем из графа все сильно связные компоненты, содержащие дуги, помеченные символами “a” или “b”. В оставшихся циклах для каждой точки сечения e можем определить последнее достижимое состояние автоматной сети $q_e = \mathrm{Mark}([e]) \cap Q$ и последнюю достижимую обозреваемую подразметку сети $P_e = \mathrm{Mark}([e]) \cap \mathrm{Obs}(\varphi)$. Далее для каждой точки сечения e мы можем, используя только автоматную сеть, определить, допускает ли разметка P_e в состоянии q_e цикл, содержащий допустимое место. Если такой цикл найден, мы можем реконструировать последовательность срабатываний системной сети, удовлетворяющую $\neg\varphi$.

4. РАСКРАШЕННЫЕ СЕТИ ПЕТРИ

В этом разделе приведены основные определения раскрашенных сетей Петри (РСП). Более детальное описание можно найти в [8,9].

Определение 4.1. *Мультимножеством* называется функция $m: S \to \mathrm{Nat}$, где S — это произвольное множество, а Nat является множеством натуральных чисел.

Над мультимножествами естественным образом определяются операции сложения $m_1 + m_2$, умножения на скаляр $n \cdot m$, сравнения $m_1 \le m_2$, $m_1 < m_2$, вычитания и модуль (размер) мультимножества $|m|$.

Мощность множества m определяется следующим образом: $|m| = \sum_{s \in S} m(s)$.

Пусть Var(expr) обозначает множество переменных в выражении expr, и Type(expr) обозначает тип выражения expr.

Определение 4.2. *Раскрашенной сетью Петри* называется сеть

$$\boldsymbol{N} = (S, P, T, A, N, C, G, E, I),$$

где S, P, T, A — это множества *цветов, мест, переходов* и *дуг* таких, что

– $P \cap T = P \cap A = T \cap A = \varnothing$;

– N является отображением $N: A \to P \times T \cup T \times P$;
– C является *функцией раскраски* $C: P \to S$;
– G является *страж-функцией* (guard function), определенной на переходах, такой, что $\forall t \in T$ Type(G(t))=bool и $\text{Type}(\text{Var}(G(t))) \subseteq S$;
– E является функцией на дугах такой, что $\text{Type}(E(a)) = C(p)_{MS}$, где p — это место из N(a) и $\text{Type}(\text{Var}(E(a))) \subseteq S$;
– I является *инициализирующей функцией,* определенной на местах, такой, что $\forall p \in P$ $\text{Type}(I(p)) = C(p)_{MS}$.

Естественным образом вводятся следующие множества:

$A(x) = \{a \in A \mid \exists x_1 \in X: [N(a) = (x, x_1) \vee N(a) = (x_1, x)]\}$,

$\forall t \in T: \text{Var}(t) = \{v \in \text{Var}(G(t)) \vee \exists a \in A(t): v \in \text{Var}(E(a))\}$,

$A(x_1, x_2) = \{a \in A \mid N(a) = (x_1, x_2)\}$,

$E(x_1, x_2) = \sum E(a)$, где суммирование проводится по всем $a \in A(x_1, x_2)$.

Определение 4.3. *Означивание* b — это функция из Var(t) такая, что

$$b(v) \in \text{Type}(v) \text{ и } G(t)\langle b\rangle.$$

Множество означиваний для перехода t обозначается B(t).

Для любого $v \in \text{Var}(t)$ $b(v) \in \text{Type}(v)$ при условии $G(t)\langle b\rangle$, где $G(t)\langle b\rangle$ означает результат вычисления G(t) при означивании b.

Определение 4.4. *Знаковым элементом (token element)* называется пара (p, c), где $p \in P$ и $c \in C(p)$.

Множество знаковых элементов обозначается TE.

Определение 4.5. *Означивающим элементом (binding element)* назовем пару (t, b) , где $t \in T$, а $b \in B(t)$ (множества всех означиваний t). Множество всех означивающих элементов будем обозначать BE.

Определение 4.6. *Разметкой* M называется мультимножество над TE.

Определение 4.7. *Начальной разметкой* M_0 назовем разметку, удовлетворяющую следующей формуле: $\forall (p,c) \in TE : M_0(p, c) = (I(p))(c)$.

Определение 4.8. *Шагом* Y называется мультимножество над BE.

Определение 4.9. Шаг Y возможен в разметке M, если

$$\forall p \in P \sum_{(t,b)\in Y} E(p,t)\langle b\rangle \le M(p)$$

и полученная после срабатывания Y разметка дается формулой:

$$M_1(p) = M(p) - \sum_{(t,b)\in Y} E(p,t)\langle b\rangle + \sum_{(t,b)\in Y} E(t,p)\langle b\rangle.$$

В этом случае, если разметка M_2 непосредственно достижима из разметки M_1, мы будем писать $M_1[Y\rangle M_2$. Множество разметок, достижимых из M, обозначается $[M_0\rangle$. Множество достижимых разметок сети равно $[M_0\rangle$.

В качестве примера рассмотрим РСП, представляющую задачу об обедающих философах, в которой философы берут обе палочки одновременно (рис. 1).

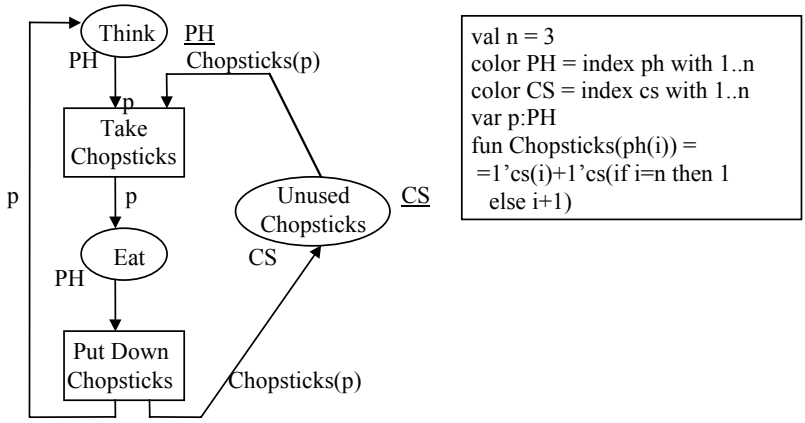

Рис. 1. Обедающие философы

Для проведения проверки моделей на раскрашенных сетях Петри в качестве системы переходов берется граф достижимости. Граф достижимости дает нам полную модель поведения раскрашенной сети Петри.

Определение 4.10. Графом достижимости называется граф (V,A,N), где

$V = [M_0\rangle$ — множество вершин;

A — множества троек $(M_1, b, M_2) \in V \times BE \times V$ таких, что $M_1[b\rangle M_2$ — множество дуг;

N — функция смежности такая, что $a = (M_1, b, M_2) \in A \Rightarrow N(a) = (M_1, M_2)$.

Следующий алгоритм строит граф достижимости по данной РСП [20, 21]:

```
Waiting:=∅;
Node(M0);
repeat
    select a node M1∈Waiting;
    for all (b,M2)∈Next(M1) do
    begin
        Node(M2);
        Arc(M1,b,M2);
    end;
    Waiting:=Waiting-{M1};
until Waiting ≠ ∅
```

$Next(M_1)$ определяется следующим образом:

$$Next(M_1) = \{(b, M_2) \in BE \times M \mid M_1[b\rangle M_2\}.$$

Функция Node добавляет M_2 в множество Waiting.

5. РАЗВЕРТКИ РАСКРАШЕННЫХ СЕТЕЙ ПЕТРИ

5.1. Ветвящийся процесс для раскрашенных сетей Петри

В данной главе описывается применение методов развертки к раскрашенным сетям Петри. Развертка определяется как конечный префикс максимального ветвящегося процесса РСП. В данном пункте дано определение ветвящегося процесса РСП и доказано существование максимального ветвящегося процесса для заданной РСП. В данной работе на рассматриваемые РСП накладывается ряд ограничений. Мы начнем с описания рассматриваемого подкласса РСП. Данный подкласс является достаточно большим для описания многих интересных систем и тем не менее позволяет нам построить конечный префикс для максимального ветвящегося процесса сети из данного класса. В описании мы следуем нотации CPN ML, описанной, например, в работе [20].

При описании цветов основная идея состоит в рассмотрении только конечных цветовых областей и функций, имеющих конечные области определения и значений.

Множество основных цветовых областей определяется на основе четырех основных типов языка Standard ML (SML) следующим образом:

color A = int with m..n // m < n
color B = bool
color C = unit
color D1 = string with "x".."y" and m..n // x < y and m < n
color D2 = string with s1|s2|...sn // the explicit enumeration.

Также допускается непосредственное задание конечных цветовых областей. Например:

color E = with X1|X2|…|Xn
color F = index expr with m..n,

Допускается переименование имеющихся цветов

color G = bool with (yes, no)
color H = unit with e.

и построение новых цветовых областей из имеющихся следующим образом:

color I = product A1×A2×A3×…×An
color J = record i:A1, j:A2, … k:An
color K = list A with m..n
color L = color A

Все функции, описанные в нотации SML и имеющие областями определения и значений вышеуказанные цветовые области, допускаются в рассматриваемом подклассе. Это же относится к переменным, констан-

там, операциям и выражениям в сети. Ниже приведены два примера функций, допустимых в рассматриваемом подклассе РСП.

Fun F1(n:A) = if n>2 then 1 else 2
Fun F2(x:E) = case x of p ⇒ 2'e | q ⇒ e

Определение 5.1. РСП, удовлетворяющая описанным выше требованиям, будем называть *S-конечной*.

Понятие n-безопасности для РСП имеет то же значение, что и для стандартных сетей Петри.

Определение 5.2. Разметка M РСП называется *n-безопасной,* если $|M(p)| \leq n$ для любого $p \in P$. РСП называется *n-безопасной*, если все ее достижимые разметки являются n-безопасными.

В данной работе мы будем рассматривать РСП, удовлетворяющие следующим свойствам:

1) число мест и переходов в сети конечно;
2) рассматриваемые РСП являются n-безопасными;
3) рассматриваемые РСП являются S-конечными.

Если не оговорено противное, то под термином РСП мы будем понимать РСП, удовлетворяющую данным требованиям.

Определим понятия пре- и пост-множества для РСП.

Определение 5.3. *Пре-множеством* вершины $x \in P \cup T$, обозначаемым $^\bullet x$, называется множество $^\bullet x = \{y \in P \cup T \mid \exists a: N(a) = (y, x)\}$. *Пост-множеством* вершины $x \in P \cup T$, обозначаемым $^\bullet x$, называется множество $x^\bullet = \{y \in P \cup T \mid \exists a: N(a) = (x,y)\}$.

Теперь мы готовы дать основное определение данной части — определение ветвящегося процесса РСП.

Определение 5.4. *Ветвящимся процессом* РСП $\boldsymbol{N_1} = (S_1, P_1, T_1, A_1, N_1, C_1, G_1, E_1, I_1)$ является четвёрка $(\boldsymbol{N_2}, h, \varphi, \eta)$, где $\boldsymbol{N_2} = (P_2, T_2, N_2)$ — это О-

сеть, h — это гомоморфизм из $\boldsymbol{N_2}$ в $\boldsymbol{N_1}$, φ и η — это функции из P_2 и T_2 соответственно такие, что

a) $\varphi(p) \in C(h(p))$;
b) $\eta(t) \in B(h(t))$.

Остальные условия приведены ниже.

c) $\mathrm{Min}(\boldsymbol{N_2}) == M_0$.

Здесь и далее оператор двойного равенства будет означать равенство двух мультимножеств знаковых элементов. Данное равенство также может быть записано в виде $\forall p_1 \in P_1 \ \Sigma_{(\ p \in \mathrm{Min}(\boldsymbol{N2}) \mid h(p) = p1)} \varphi(p) = M_0(p_1)$.

d) $G(h(t)) < \eta(t) > \ \forall t \in T_2$;

e) $\forall t' \in T_2 \mid (\exists a \in A_1: N_1(a) = (p, t) \text{ и } h(t') = t) \Rightarrow E(a) < \eta(t') >= \sum_{(p' \in I)} \varphi(p')$,
где $I = \{p' \in {}^\bullet t' \mid h(p') = p\}$;
$\forall t' \in T_2 \mid (\exists a \in A_1: N_1(a) = (t,p) \text{ и } h(t')=t) \Rightarrow E(a) < \eta(t) >= \sum_{(p' \in I)} \varphi(p')$,
где $I = \{p' \in (t,b)^\bullet \mid h(p') = p\}$;

f) если $(h(t_1) = h(t_2))$ и $(\eta(t_1) = \eta(t_2))$ и $({}^\bullet t_1 = {}^\bullet t_2)$, то $t_1 = t_2$.

Замечание. Используя первые два свойства, мы можем связать знаковый элемент (p, c) сети $\boldsymbol{N_1}$ с каждым местом сети $\boldsymbol{N_2}$ и каждый означивающий элемент (t, b) сети $\boldsymbol{N_1}$ с каждым переходом сети $\boldsymbol{N_2}$**.** В дальнейшем мы иногда будем рассматривать места $\boldsymbol{N_2}$ как знаковые элементы сети $\boldsymbol{N_1}$, а переходы сети $\boldsymbol{N_2}$ как означивающие элементы сети $\boldsymbol{N_1}$. Например, $h((t, b)) = t$ будет означать $h(t') = t$ и $\eta(t') = b$ для некоторого $t' \in T_2$ или $p \in {}^\bullet(t, b)$ будет означать $p \in {}^\bullet t'$ и $h(t') = t$ и $\eta(t') = b$. Аналогично мы будем рассматривать $(p, c) \in P_2$ как $p' \in P_2$ и $h(p') = p$ и $\varphi(p) = c$. То же верно для $h(p, c) = p$ и $h(t, b) = t$.

Можно показать, что любая конечная РСП имеет максимальный ветвящийся процесс [23]. Мы можем доказать существование максимального ветвящегося процесса, рассмотрев алгоритм его построения. В качестве такого алгоритма мы адаптируем алгоритм, предложенный МакМилланом в работе [32] для раскрашенных сетей Петри.

Максимальный ветвящийся процесс сети $\boldsymbol{N}$ будем обозначать МВР($\boldsymbol{N}$). Полученный МВР($\boldsymbol{N}$) часто является бесконечным даже для конечной РСП $\boldsymbol{N}$. Поскольку для МВР($\boldsymbol{N}$)= ($\boldsymbol{N_2}$, h, φ, η) сеть $\boldsymbol{N_2}$ является стандартной (нераскрашенной) сетью-процессом, определения конфигурации, локальной конфигурации, сечения, конфликта и другие определения, связанные с О-сетью, можно применять и к максимальному ветвящемуся процессу РСП. В следующей части работы будут определены критерии финитизации макси-

мального ветвящегося процесса и построены различные типы разверток РСП.

5.2. Определения и основные свойства развертки раскрашенных сетей Петри

В данной части работы мы покажем, как адаптировать технику финитизации максимального ветвящегося процесса к РСП. Будут построены различные типы разверток РСП и доказаны важные свойства полученных разверток.

Дадим предварительно следующее определение.

Определение 5.5. Пусть дана РСП $\boldsymbol{N_1} = (S_1, P_1, T_1, A_1, N_1, C_1, G_1, E_1, I_1)$ и МВР($\boldsymbol{N_1}$) = ($\boldsymbol{N_2}$, h, φ, η), где $\boldsymbol{N_2} = (P_2, T_2, N_2)$ является максимальным ветвящимся процессом сети $\boldsymbol{N_1}$. Пусть C — это конфигурация сети $\boldsymbol{N_2}$. Мы определим разметку *Mark(C)* по сечению Cut(C) сети $\boldsymbol{N_1}$ следующим образом:

$$\mathrm{Mark}(C)(p) = \Sigma_{(p' \in \mathrm{Cut}(C) \mid h(p') = p)} M_2(p').$$

Рассмотрим максимальный ветвящийся процесс для данной РСП. Легко заметить, что МВР($\boldsymbol{N}$) удовлетворяет свойству полноты, т.е. для любой достижимой разметки M сети $\boldsymbol{N}$ существует конфигурация C МВР($\boldsymbol{N}$) (т.е. C является конфигурацией соответствующей О-сети) такая, что Mark(C) = M. Иначе необходимая последовательность срабатываний может быть добавлена к МВР($\boldsymbol{N}$), не нарушая требований определения, и мы получим противоречие с максимальностью МВР(N). Формальное доказательство можно найти в работе [23].

Определенные для стандартных СП три типа финитизации также могут быть применены к РСП.

Определение 5.6. Переход $t \in T$ из РСП называется *GT_0-точкой сечения*, если существует $t_0 \in T$ такой, что Mark([t]) = Mark([t_0]) и [t_0] ⊂ [t].

Определение 5.7. Переход $t \in T$ из РСП называется *GT-точкой сечения*, если существует $t_0 \in T$ такой, что Mark([t]) = Mark([t_0]) и |[t_0]| < |[t]|.

Определение 5.8. Переход $t \in T$ из РСП называется *EQ-точкой сечения*, если существует $t_0 \in T$ такой, что

1) Mark([t]) = Mark([t_0]);

2) $|[t_0]| = |[t]|$;
3) $\neg(t \parallel t_0)$;
4) не существует EQ-точки сечения среди t' таких, что $t' \parallel t_0$ и $|[t']| \leq |[t_0]|$.

Определение 5.9. Для РСП N развертка получается из максимального ветвящегося процесса после удаления всех переходов t' таких, что существует точка сечения t и $t < t'$, а также всех мест $p \in t'^{\bullet}$. При использовании GT_0(GT)-точек сечения полученную развертку будем называть *GT_0(GT)-развертка* или разверткой МакМиллана. Если в качестве точек сечения возьмем объединение GT-точки сечения $\cup$ EQ-точки сечения, то полученная развертка называется *EQ-развертка*.

Утверждение 5.1. Размер EQ-развертки $\leq$ размер GT-развертки $\leq$ размер GT_0-развертки.

Теорема 5.1. Пусть N_1 является РСП. Тогда для её разверток верны следующие утверждения:

1) EQ-развертка, GT-развертка и GT_0-развертка конечны;
2) EQ-развертка, GT-развертка и GT_0-развертка безопасны, т.е. если C и C′ — конфигурации, тогда $C \subseteq C' \Rightarrow \text{Mark}(C') \in [\text{Mark}(C)\rangle$;
3) EQ-развертка, GT-развертка и GT_0-развертка являются полными, т.е. $M \in [M_0\rangle \Rightarrow$ существует конфигурация C такая, что Mark(C) = M.

6. МЕТОД ПРОВЕРКИ МОДЕЛЕЙ ДЛЯ РАСКРАШЕННЫХ СЕТЕЙ ПЕТРИ

В данной главе мы применим метод проверки моделей для однобезопасных сетей Петри, описанный выше, к раскрашенным сетям Петри и их разверткам. Мы будем рассматривать автоматную сеть $A_{\neg\varphi}$ как РСП и построим РСП, являющуюся произведением системной РСП и данной автоматной сети. Далее для полученной РСП согласно методам, описанным выше, строится развертка и ищутся пути, удовлетворяющие $\neg\varphi$.

Вначале определим LTL-семантику для РСП, основанную на состояниях. Напомним, что мы рассматриваем только n-безопасные РСП. Множество атомарных высказываний отождествляется с множеством $n \cdot TE = TE \times TE \times ... \times TE$, где n является числом из условия n-безопасности и TE является множеством знаковых элементов. Утверждение $m'(p, c)$ верно в состоянии M, если $m'(p, c) \in M(p)$. Последовательность $M_0[t_0, b_0\rangle M_1[t_1, b_1\rangle M_2...$ удовлетворяет φ, если w-слово $M_0M_1M_2...$ принадлежит L_φ. Разметка M удовлетворяет φ, если любая последовательность срабатываний $M[t, b\rangle M_1[t_1, b_1\rangle M_2...$

удовлетворяет φ. В этом случае будем писать $M \models \varphi$. РСП $\boldsymbol{N}$ удовлетворяет φ, если M_0 удовлетворяет φ. В этом случае будем писать $\boldsymbol{N} \models \varphi$. Пусть $p \in P$ является местом РСП **N**. Если $(p, c) \in \langle\varphi\rangle$ для некоторого $c \in C(p)$, то иногда для удобства мы будем писать $p \in \langle\varphi\rangle$.

Используемый алгоритм проверки моделей включает в себя поиск циклов, содержащих допускаемые места. Для поиска таких циклов мы будем использовать алгоритм, предложенный в [40]. Мы также будем считать рассматриваемую РСП $\boldsymbol{N}$ сетью без тупиковых состояний (deadlock free). В работе [23] были описаны методы обнаружения тупиков в стандартной СП и РСП с применением разверток. Если в сети обнаружено тупиковое состояние, имеющее какое-либо системное значение, например, конечное состояние получения всех сообщений в коммуникационных протоколах, то мы можем добавить дополнительные переходы для элиминации данного тупикового состояния.

Пусть мы имеем РСП $\boldsymbol{N} = (S, P, T, A, N, C, G, E, I)$ и автоматную сеть $A_{\neg\varphi} = (Q, q_0, \delta, F)$, описанную в пункте 2.3.4.

Рассмотрим автоматную сеть $A_{\neg\varphi}$ как раскрашенную. Множество состояний автомата будем рассматривать как множество мест. Множество цветов состоит из одного цветового элемента /color H=unit with e/. Каждую дугу автомата будем рассматривать как переход без страж-функции. В начальной разметке A_0 единственный знаковый элемент — точка, находящаяся в месте q_0. Каждый переход рассматривается, как помеченный n-безопасной разметкой M, включающей только места из $\langle\varphi\rangle$. Построим произведение системной РСП и полученной раскрашенной автоматной сети.

Определение 6.1. Пусть даны РСП $\boldsymbol{N} = (S, P, T, A, N, C, G, E, I)$ и раскрашенная автоматная сеть $A_{\neg\varphi}$. Произведение *Prod* строится таким образом, что сети **N** и $A_{\neg\varphi}$ срабатывают поочередно.

1. Для каждого места $p \in P$ такого, что $(p, c) \in \langle\varphi\rangle$ для некоторого $c \in C(p)$, добавим *комплиментарное* место p'. Это делается для проверки подразметки M при срабатывании автоматного перехода, помеченного разметкой M.

Для p' будем считать $M(p') = \sum_{\{c \mid (p,c)\in\langle\varphi\rangle\}}(n'(p, c) \setminus M(p))$. Множество комплиментарных мест обозначим P'.

Множество $Obs(\varphi)=\{p, p' \mid (p, c) \in \langle\varphi\rangle$ для некоторого $c\}$ называется множеством *обозреваемых (обозримых)* мест.

2. Добавим дополнительные места s_f и s_s для организации поочередного срабатывания сетей **N** и $A_{\neg\varphi}$. Для этих мест положим $C(s_f) = C(s_s) =$ /color H=unit with e/. Это моделирует простые, не раскрашенные места.

Полученное множество мест в сети ***Prod***: $P \cup P' \cup Q \cup \{s_f, s_s\}$.

3. Для каждого перехода t автоматной сети, помеченного разметкой M, добавим две дуги $a_{p,t}$ и $a_{t,p}$:

$$N(a_{p,t}) = (p, t)\ \forall p \in Obs(\varphi),$$
$$N(a_{t,p}) = (t, p)\ \forall p \in Obs(\varphi),$$
$$E(a_{p,t}) = E(a_{t,p}) = m'(p, c),$$
$$\forall p \in \langle\varphi\rangle, \text{ где m является индексом (p, c) в M.}$$

Для комплиментарных мест

$$E(a_{p',t}) = E(a_{t,p'}) = (n - m)'(p, c)\ \forall p' \in Obs(phi) \setminus \langle\varphi\rangle.$$

4. Для каждого перехода t автоматной сети мы добавляем дуги $a_{t,s}$ и $a_{t,f}$:

$$N(a_{t,s}) = (t, s_s),\ N(a_{t,f}) = (s_f, t),\ E(a_{t,s}) = E(a_{t,f}) = e.$$

Для каждого перехода T системой РСП добавляем дуги $a_{T,s}$ и $a_{T,f}$:

$$N(a_{T,s}) = (s_s, T),\ N(a_{T,f}) = (T, s_f),\ E(a_{T,s}) = E(a_{T,f}) = e.$$

5. Для каждого перехода T системной РСП и каждого места $p \in \langle\varphi\rangle$ таких, что $p \in {}^\bullet T$ или $p \in T^\bullet$, добавляется дуга $a_{T,p}$.

$$N(a_{T,p}) = \begin{cases} (T, p), & \text{åñëè } p \in {}^\bullet T, \\ (p', T) & \text{иначе.} \end{cases}$$

Если $p \in {}^\bullet T$ и $p \in T^\bullet$, добавим обе дуги. Ниже даны выражения на дугах.

$$E(a_{T,p}) = \begin{cases} E(p, T), & \text{åñëè } N(a_{T,p}) = (T, p), \\ E(T, p) & \text{иначе.} \end{cases}$$

6. Начальная M_{Prod_0} разметка задается следующим образом:

$$M_{Prod\_0} = M_0 \cup M'_0 \cup \{q_0\} \cup \{s_f\},$$

что означает, что мы помещаем по одному знаковому элементу в места q_0 и s_f, и множество M'_0 задаётся следующим образом:

$$M'_0(p') = \sum_{\{c \mid (p,c) \in \langle\varphi\rangle\}} (n'(p, c) \setminus M_0(p)).$$

Пусть $\gamma = M_0[t_0, b_0\rangle M_1 [t_1, b_1\rangle...$ является последовательностью срабатываний ***Prod***. Пусть для каждой разметки M_j из γ $P_j = M_j \cap P$ будет ограничением M_j на системные места. Проекция γ на системные места определяется следующим образом:

$$Proj(\gamma) = P_{i0}[t_{i0},b_{i0}\rangle P_{i1}[t_{i1},b_{i1}\rangle...$$

Последовательность $P_{i0} P_{i1} P_{i2}$...обозначим $Proj_M(\gamma)$.

Утверждение 6.1. Пусть дано произведение ***Prod*** РСП ***N*** и автоматной сети $A_{\neg\varphi}$. $N \models \varphi \Leftrightarrow$ сеть ***Prod*** не содержит циклической последовательности срабатываний, включающей места из $A_{\neg\varphi}$, соответствующие допустимым состояниям автомата $A_{\neg\varphi}$.

Как было замечено в [40], строго последовательное поведение полученного произведения сводит на нет все преимущества использования методов частичного порядка. Там же было предложено ограничиться рассмотрением только статтерно-эквивалентных свойств. В данном случае достаточно рассматривать только видимые переходы, т.е. переходы, имеющие смежные с обозреваемыми местами дуги. В данной работе мы также ограничимся статтерно-эквивалентными свойствами. При построении произведения системной и автоматной РСП это означает, что мы будем проводить синхронизацию только видимых переходов системной РСП с переходами автоматной РСП. При таком построении существует возможность того, что некоторые последовательности срабатываний системной РСП удовлетворяют $\neg\varphi$, но соответствующие пути в произведении не допускаются. Это происходит в том случае, когда рассматриваемая последовательность содержит только конечное число видимых переходов. Следующее утверждение дает нам возможность применять двухшаговый алгоритм проверки LTL′ формулы, изложенный выше, к произведению РСП и автоматной сети $A_{\neg\varphi}$.

Утверждение 6.2. Если γ является последовательностью срабатываний произведения Prod системной РСП и автоматной сети $A_{\neg\varphi}$, содержащей бесконечно много видимых переходов, тогда $Proj(\gamma) \models \neg\varphi \Leftrightarrow \gamma$ является допустимой последовательностью.

В качестве примера рассмотрим задачу об обедающих философах, описанную выше. Будем проверять формулу $\varphi \equiv \Diamond(Eat, 1)$, означающую, что первый философ при любой последовательности срабатываний рано или поздно окажется в состоянии Eat. Рассмотрим отрицание данной формулы

и соответствующую автоматную РСП: $\neg\varphi \equiv \neg\Diamond(\text{Eat}, 1)$. Автоматная РСП для данной формулы приведена на рис. 2.

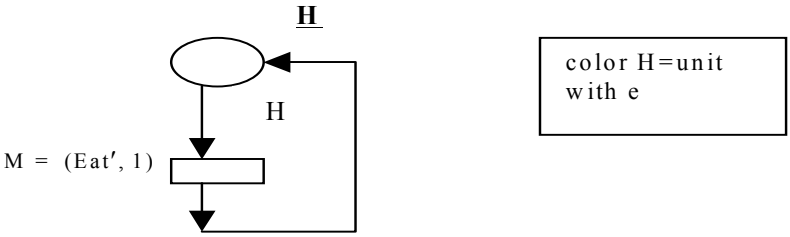

Рис. 2. Автоматная РСП $A_{\neg\varphi}$ для $\neg\varphi \equiv \neg\Diamond(\text{Eat}, 1)$

На рис. 3 приведено частично построенное произведение ***Prod*** РСП, представленных на рис. 1 и 2.

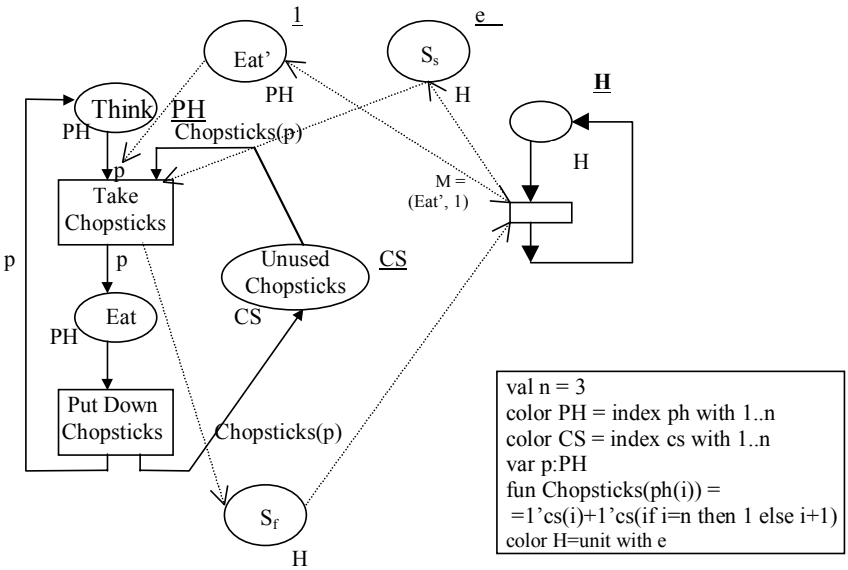

Рис. 3. Частично построенное произведение

Согласно рассматриваемому алгоритму необходимо построить развертку сети ***Prod*** и построить граф, состоящий из точек сечения. Для данного графа применяется двухшаговый алгоритм, описанный в части 3.2. На рис. 4 приведена развертка произведения РСП, представляющей задачу об обедающих философах (рис. 1) для числа философов n = 2 и автоматной РСП для формулы $\neg \varphi \equiv \neg \Diamond($ Eat, 1). Для данного примера все три вида развертки являются идентичными. Рис. 5 представляет граф, состоящий из точек сечения для данного примера.

Как видно из рисунка, данный граф содержит цикл и, следовательно, формула $\varphi \equiv \Diamond$(Eat, 1), означающая, что первый философ при любой последовательности срабатываний рано или поздно окажется в состоянии Eat, не верна для РСП, представляющей задачу об обедающих философах. Эта формула не верна на последовательности срабатываний сети, в которой только второй философ берет палочки и переходит в состояние Eat, кладет палочки и переходит в состояние Think и т.д.

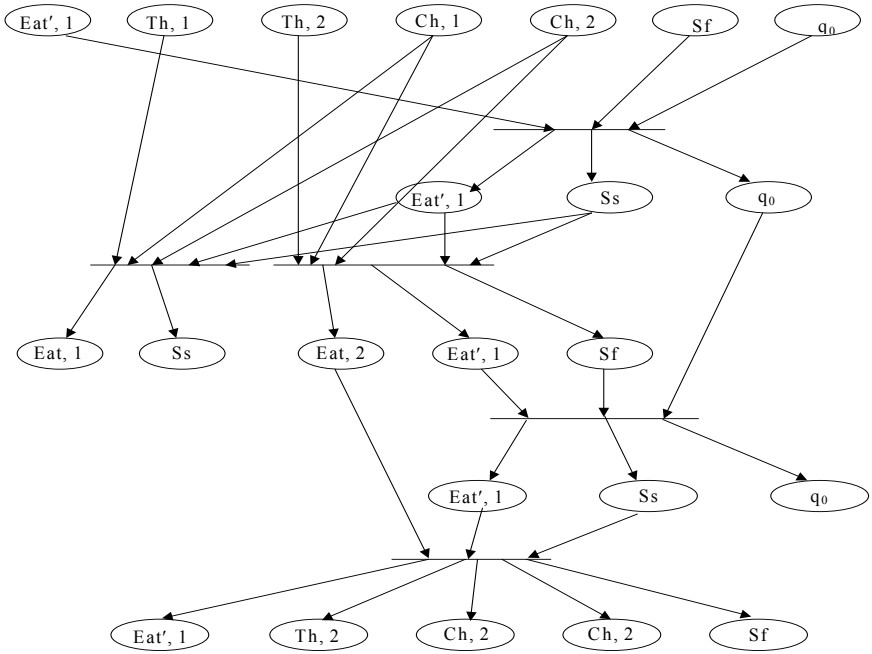

Рис. 4. Развертка РСП (рис. 3)

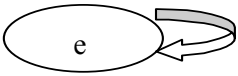

Рис. 5. Граф состоящий из единственной точки сечения

Прототипная реализация данного алгоритма для РСП представлена в следующей части работы. Там же приведены результаты экспериментов, проведенных с использованием данной системы.

7. РЕАЛИЗАЦИЯ МЕТОДА ПРОВЕРКИ МОДЕЛЕЙ С ИСПОЛЬЗОВАНИЕМ РАЗВЕРТОК

7.1 Описание реализации

В данной части работы будет описана прототипная реализация метода проверки моделей с использованием разверток, описанного в пятой главе. При реализации данной системы не ставилось цели проведения серьезных экспериментов и сравнения их результатов с результатами аналогичных экспериментов, проведенными другими методами верификации систем. Целью данной реализации было показать принципиальную возможность использования данного метода на практике и привести несколько примеров верификации систем.

Данная система не производит полной автоматизации верификации процесса. Схема работы системы представлена на рис. 6 и, как можно видеть, система не содержит стандартного входного языка. Во внутреннее представление системы записывается построенное “вручную” произведение двух РСП: исходной и автоматной для формулы $\neg\varphi$, также построенной “вручную”. Далее система осуществляет построение разверток для различных критериев финитизации и строит граф, состоящий из точек сечения разверток. Последний блок системы ищет сильно связные компоненты данного графа. Если данная проверка оказывается неудачной, тогда второй шаг алгоритма проверки моделей также делается вручную. Этот шаг требует работы только с автоматной РСП для $\neg\varphi$ и не является для рассмотренных примеров сложной задачей.

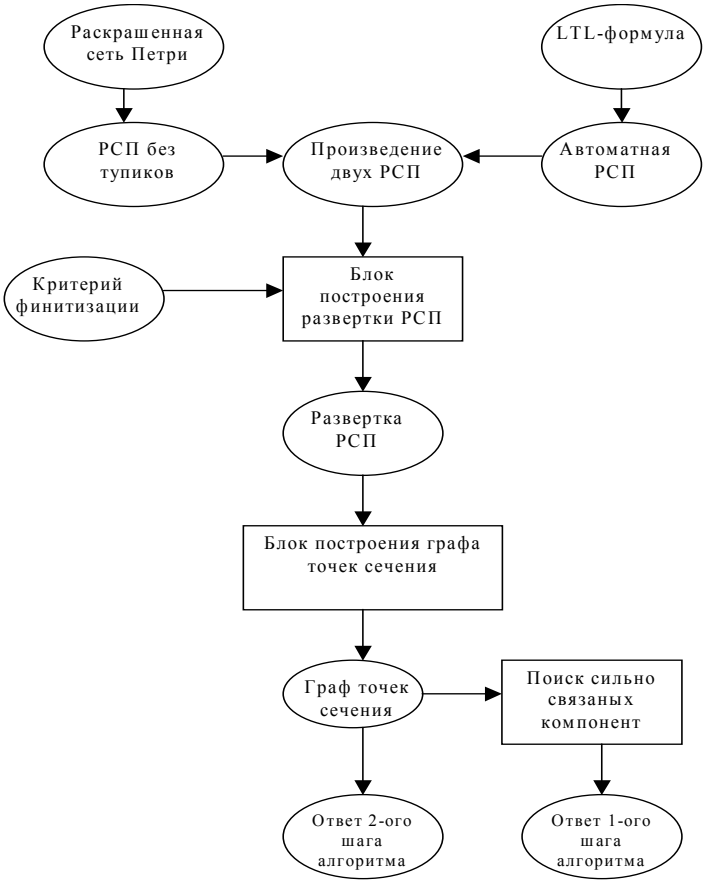

Рис. 6. Система проверки моделей с использованием разверток

Все блоки системы реализованы на языке C++ для платформы Windows 2000. В системе отсутствуют входной и выходные языки. Произведение системной и автоматной РСП записывается непосредственно во внутреннее представление системы. Развертка полученного произведения и граф, представляющий точки сечения, записываются в выходные файлы также непосредственно во внутреннем представлении. Система поиска сильно связных компонентов является внешней по отношению к системе и требует предварительной загрузки полученного графа.

7.2. Экспериментальные результаты

Обедающие философы

В качестве первого примера для верификации рассмотрим РСП (рис.7), представляющую задачу об обедающих философах. Философы сидят за круглым столом и берут вначале левую, затем правую палочку. Взяв обе палочки, философ переходит в состояние еды. Сеть была промоделирована для n=2..5.

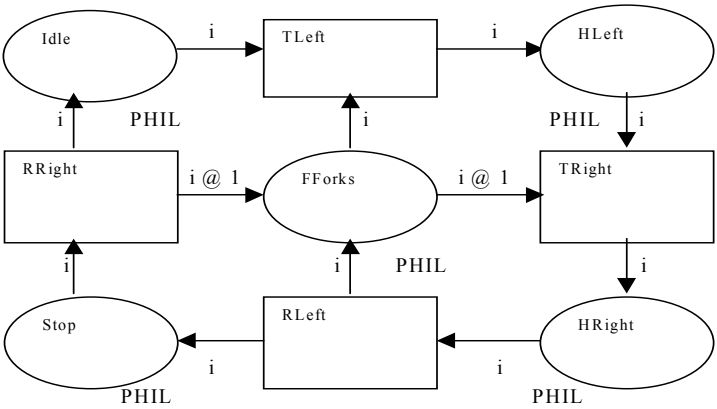

Рис. 7. Обедающие философы

Для данной сети проверялись свойства прогресса, формально описанные ниже. Предварительно нам необходимо модифицировать сеть так, чтобы она не содержала тупиковых состояний. Как было показано в работе [23], используя теорию разверток раскрашенных сетей Петри, мы можем эффективно обнаруживать тупиковые состояния. Если обнаруженное тупиковое состояние системы имеет какой-либо "системный смысл", мы естественным образом можем модифицировать данную РСП в РСП, не содержащую тупиковых состояний. Для сети, представляющей задачу об обедающих философах, были внесены следующие изменения.

1. Был добавлен цвет CS = index cs with 1..n+1 для описания места Free Chopsticks. Философ p брал для еды палочки p и p + 1. Моделируется ситуация: философы за прямоугольным столом.

2. Мы можем преобразовать данную РСП в РСП, работающую аналогично РСП, представленной на рис. 1. В полученной сети философы берут обе палочки одновременно.

Для рассмотренной немодифицированной сети возникает тупиковая ситуация, когда все философы взяли по одной палочке. Для двух предложенных модификаций сети было показано отсутствие тупиковых состояний. Для данных РСП с помощью системы проверки моделей с использованием разверток были проверены следующие свойства формулы, описывающие свойства прогресса.

Свойства прогресса

1. В качестве первой формулы рассмотрим формулу

$$\Diamond\,(\text{Hright}, 1),$$

описывающую условие получения первым философом обеих палочек. Автоматная РСП была описана для данной формулы в предыдущей главе данной работы. Формула оказывается ложной для данной РСП, так как существует возможность циклической работы сети без получения первым философом палочек для еды.

2. Мы можем изменить формулу (1) так, чтобы она стала верной. Для этого мы включим в нее остальных философов. Получим следующую формулу:

$$\Diamond\,(\,(\text{Hright}, 1) \vee (\text{Hright}, 2) \vee \ldots \vee (\text{Hright}, n)\,),$$

где n — это число рассматриваемых философов. Данная формула оказывается истинной после ее проверки с помощью системы.

3. Следующие формулы были рассмотрены для описания того факта, что система возвращается в начальное состояние:

$(\text{Hright}, 1) \rightarrow \Diamond(\text{Idle}, 1)$ — формула верна только для n = 2 и n = 3;

(Hright, 1) $\rightarrow \Diamond$ ((Idle, 1) $\wedge$ (Idle, 2) $\wedge$ ... $\wedge$ (Idle, n)) — формула также верна только для n = 2 и n = 3;

(Hright, 1) $\rightarrow \Diamond$ ((Idle, 1) $\vee$ (Idle, 2) $\vee$ ... $\vee$ (Idle, n)) — формула является истинной;

((Hright, 1) $\vee$ (Hright, 2) $\vee$ ... $\vee$ (Hright, n)) $\rightarrow \Diamond$ ((Idle, 1) $\vee$ (Idle, 2) $\vee$ ... $\vee$ (Idle, n)) — формула также является истинной для данной сети.

Эти результаты верны как для первой, так и для второй модификации данной сети.

Битовый протокол

Мы рассматриваем битовый протокол (ABP), описанный в [20, 21]. Сеть Петри, соответствующая этому протоколу, представлена на рис. 8. Протокол моделирует в простейшем виде передачу и прием сообщений через нестабильную среду. Пользователь-передатчик передает набор сообщений в заданном порядке пользователю-приемнику. Пользователь-приемник, получив сообщение, отправляет обратно подтверждение приема данного сообщения. Получив подтверждение, пользователь-приемник начинает отправку следующего сообщения. Сообщение может быть отправлено несколько раз, так как среда может терять отправленные сообщения (как и отправленные подтверждения приема). Вносить какие-либо искажения в сообщения среда не может.

Сеть была промоделирована для n=1..4.

Перед тем, как описать эксперименты, проведенные с помощью системы проверки моделей с использованием разверток, заметим, что битовый протокол содержит тупиковое состояние. Данное тупиковое состояние является “естественным”, так как оно соответствует состоянию получения всех отправленных сообщений, а также получения всех подтверждений. Мы можем естественным образом модифицировать рассматриваемую РСП и получить сеть без тупиков с помощью добавления дополнительного перехода, проверяющего получение последнего подтверждения и организующего циклическую дальнейшую работу сети (рис. 9).

Для полученной сети проверялись следующие формулы, описывающие свойства прогресса и безопасности.

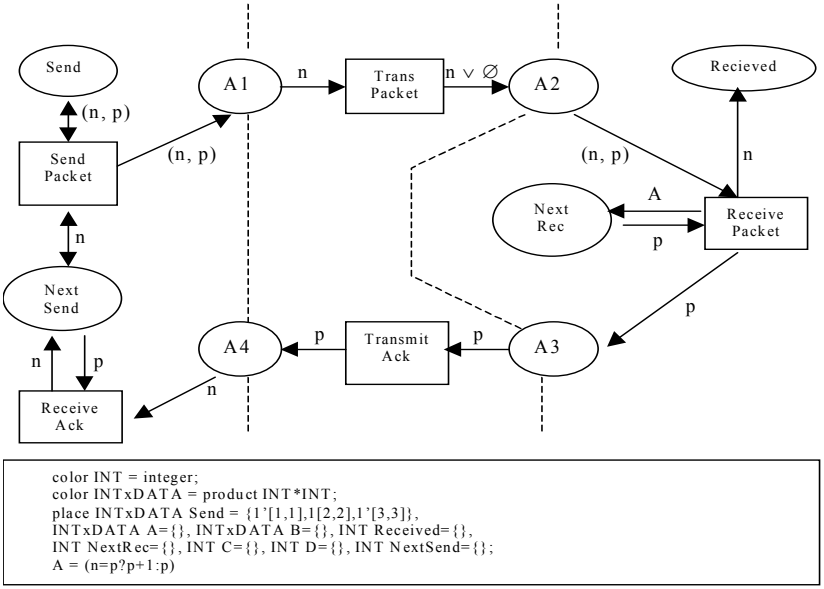

Рис. 8. Протокол АВР

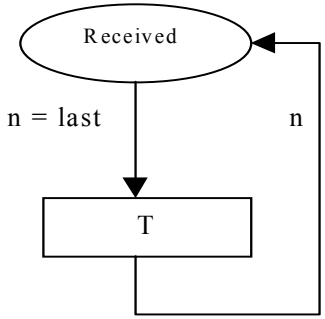

Рис. 9. Модификация битового протокола

Свойства прогресса. Как было описано выше, начав функционировать, протокол может вернуться в начальное состояние, если на этапе передачи сообщения произойдет потеря данных. Следующая формула проверяет возможность выхода из полученного “ливлока” для n = 3:

$$(A1, 1'[1, 1]) \rightarrow \Diamond ((Send, 1'[1, 1]) \wedge (Send, 1[2, 2]) \wedge (Send, 1'[3, 3]))$$

Данная формула оказывается ложной, что означает возможность выхода из “ливлока” и продолжения дальнейшей работы протокола.

Также проверялись следующие формулы, описывающие свойства прогресса:

$(A2, 1'[1, 1]) \rightarrow \Diamond$(Received, 1′[1, 1]) — формула верна для данного протокола;

$(A1, 1'[1, 1]) \rightarrow \Diamond$(Received, 1′[1, 1]) — формула неверна из-за описанного выше “ливлока”.

Свойства безопасности. В качестве формул, описывающих свойства безопасности данного протокола, будем рассматривать следующие формулы:

1. Формула

$$(Received, 1'[1,1]) \rightarrow \Box(Received, 1'[1,1])$$

означает, что после принятия первого сообщения оно уже не может быть потеряно. Данная формула была верна во всех проверенных случаях.

2. Следующая формула дает пример формулы, описывающей порядок получения сообщений. Она означает, что, получив подтверждение принятия (n – 1)-ого сообщения, протокол не будет отправлять его в дальнейшем.

$$((Next\ Send, n) \wedge (A1, n - 1)) \rightarrow \neg \Box(A1, n - 1).$$

Данная формула верна во всех проверенных случаях.

3. Последнюю рассматриваемую формулу можно отнести как к описанию свойств прогресса, так и к описанию безопасности рассматриваемого протокола.

$$(Received, 1'[1,1]) \rightarrow \Diamond (Received, 1'[3,3]).$$

Формула означает, что после получения первого сообщения при любом функционировании сети наступит момент, в котором будет получено третье сообщение. Данная формула неверна. После получения первого сообщения среда может начать работать полностью нестабильно. Чтобы проверить,

что получение третьего сообщения все же возможно нужно рассмотреть отрицание данной формулы

$$\neg((\text{Received}, 1'[1,1]) \rightarrow \Diamond\, (\text{Received}, 1'[3,3])).$$

Данная формула также оказывается ложной, что означает, что третье сообщение будет получено при нормальной работе среды передачи.

Протокол “Отправитель—получатель”

Протокол “Отправитель—получатель” (рис. 10), описанный в работе [21], был верифицирован в системе проверки моделей с использованием разверток и является основным примером для верификации в данной системе. В работах [23, 24, 25] развертка данной сети была рассмотрена теоретически со значением nb = 1, и было показано, что для данного примера развертка является очень эффективным методом моделирования с точки зрения борьбы с проблемой взрыва числа состояний. В данной части будут описаны эксперименты по проверке свойств при размере буфера nb = 1,2.

Предварительно необходимо модифицировать данную сеть с целью избавления от тупиковых состояний. Тупик возникает в данной системе только в случае получения “Получателем” всех отправленных сообщений. Необходимая модификация сети показана на рис. 11.

Так как поведение данного протокола является сильно недетерминированным, существует множество вариантов построения формул, описывающих свойства прогресса.

Для данного протокола проверялись следующие формулы, описывающие свойства прогресса и безопасности.

Свойства прогресса. Обозначим места “Отправителя” с цветом Prod × Cons A_1 и A_2. Места с тем же цветом в части “Получатель” обозначим B_1 и B_2. Тогда первый вариант формулы, описывающей свойства прогресса, будет следующим:

$$((A_1, (m, n)) \wedge (A_2, (m, n))) \rightarrow \Diamond((B_1, (m, n)) \wedge (B_2, (m, n))).$$

Формула означает, что если “Отправитель” m послал “Получателю” n сообщение, то оно дойдет по крайней мере до мест B_1 и B_2. Данная формула после проведения проверки с помощью системы оказалась верной для рассматриваемых значений размеров сети.

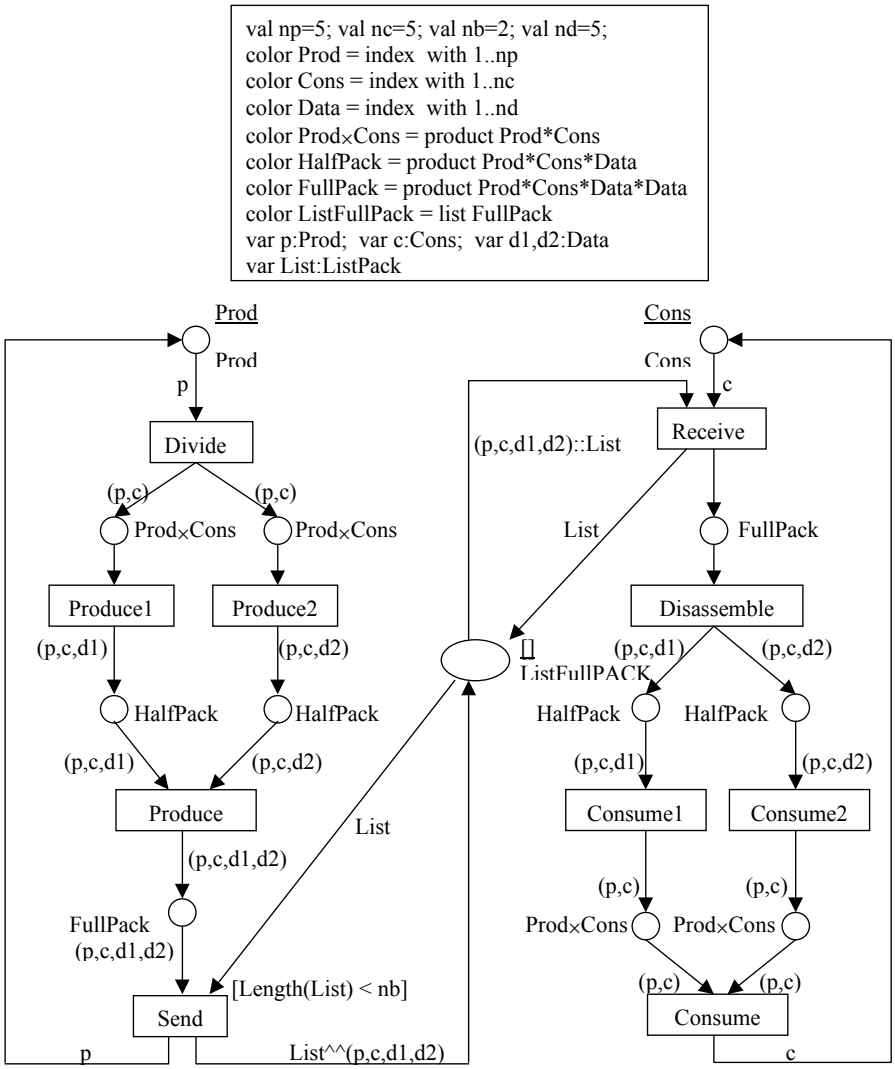

Рис.10. Пример “Отправитель—Получатель”

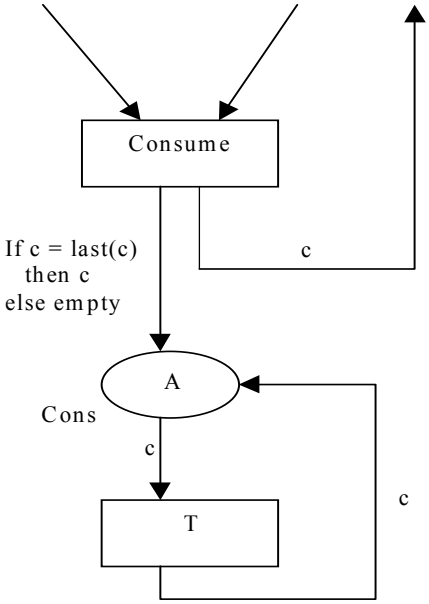

Рис. 11. Модификация протокола “Отправитель—Получатель”

Места с цветами Prod и Cons будем обозначать Prod и Cons соответственно. Следующие формулы, также выражающие свойства прогресса системы, являются верными для рассматриваемой сети.

$$(\text{Prod}, m) \rightarrow \Diamond \vee_{(n=1..nc)} ((B_1, (m, n)) \wedge (B_2, (m, n))),$$
$$((A_1, (m, n)) \wedge (A_2, (m, n))) \rightarrow \Diamond \neg (\text{Cons}, n).$$

Первая формула задает условие, согласно которому, если у нас есть “Отправитель” с номером m, тогда для некоторого “Получателя” n в местах B_1 и B_2 окажутся фишки с цветами (m, n). Данная формула оказывается неверной, т.к. “Отправитель” m может не отправить ни одного сообщения на некоторой бесконечной последовательности срабатываний рассматриваемой сети.

Вторая формула означает, что если “Отправитель” с номером m отправил сообщение “Получателю” с номером n, то наступит момент, когда данный “Получатель” начнет обрабатывать полученное сообщение. Данная формула верна для рассматриваемого протокола.

Свойства безопасности. В рассматриваемом протоколе отсутствует порядок получения сообщений. Таким образом, нет необходимости, как в случае битового протокола, проверять правильность порядка прихода сообщений. В качестве формул, описывающих свойства безопасности данного протокола, будем рассматривать следующие формулы.

Во-первых, формулы, означающие, что отправленное сообщение не может быть потеряно. Для этого мы можем использовать формулы, описывающие свойства прогресса. Во-вторых, формулы, означающие, что сообщения не могут быть заменены на другие в процессе передачи. Это может быть выражено, например, формулой

$$(\neg(A_1, (k, l))\wedge\neg(A_2, (k, l)) \wedge \neg(C_1, (k, l, i)) \wedge \neg(C_2, (k, l, i))) \mathbf{U} ((C_1, (k, l, i)) \wedge (C_2, (k, l, i))),$$

где места C_1 и C_2 обозначают места “Отправителя”, имеющие цвет HalfPack. Данная формула описывает ситуацию, в которой некоторое сообщение, не проходя через места A_1 и A_2, попадает в места C_1 и C_2. При подстановке некоторых конкретных значений данная формула оказывается ложной.

В следующей таблице приведены некоторые численные результаты экспериментов, проведенных с протоколом “Отправитель—получатель”. В качестве формул рассмотрим описанные выше формулы свойств прогресса:

$$\Phi_1 \equiv ((A_1, (m, n)) \wedge (A_2, (m, n))) \rightarrow \Diamond((B_1, (m, n)) \wedge (B_2, (m, n))),$$
$$\Phi_2 \equiv ((A_1, (m, n)) \wedge (A_2, (m, n))) \rightarrow \Diamond\neg(\text{Cons}, n).$$

Экспериментальные результаты

np	nc	nd	nb	Формула	Размер развертки произведения	Размер графа точек сечения
1	1	1	1	Φ_1	36	3
1	1	1	1	Φ_2	42	4
2	2	1	2	Φ_1	617	11

ЗАКЛЮЧЕНИЕ

В данной работе описан метод проверки моделей для раскрашенных сетей Петри с использованием разверток в качестве моделей. Для данного метода приведены основные теоретические результаты и описана прототипная реализация системы проверки моделей, базирующаяся на развертках РСП. Данная реализация работает в полуавтоматическом режиме и позволяет проводить верификацию моделей распределенных систем, описанных на языке раскрашенных сетей Петри.

Проверка моделей заключается в описании требуемых свойств системы на языках логических спецификаций и доказательстве истинности или ложности данных спецификаций на построенном пространстве состояний системы. Использование развертки в качестве модели позволяет во многих случаях существенно уменьшить рассматриваемое пространство состояний. Таким образом, данный подход способствует повышению эффективности методов проверки моделей и является существенным вкладом в борьбу с известной проблемой “взрыва числа состояний”.

В работе описана структура системы проверки моделей и особенности ее реализации. С помощью данной системы был проведен ряд экспериментов, демонстрирующих возможность использования метода проверки моделей с применением разверток РСП на практике и эффективность данного подхода. В качестве примеров были рассмотрены РСП, представляющая задачу об обедающих философах, сеть, описывающая модель однобитового коммуникационного протокола, и сеть, представляющая модель коммуникационного протокола “Отправитель—Получатель”. Метод использует логику линейного времени LTL для задания логических спецификаций. Данные примеры были верифицированы для свойств “прогресса” и “безопасности”.

Благодарности. Я хочу поблагодарить Валерия Александровича Непомнящего за привлечение моего внимания к проблеме разверток раскрашенных сетей Петри и множество плодотворных дискуссий. Также я хочу поблагодарить Елену Боженкову за ценные замечания по теме работы.

СПИСОК ЛИТЕРАТУРЫ

- **Алексеев Г.И. и др.** Использование сетей Петри для верификации распределенных систем, представленных на языке Estelle // Известия РАН. Сер.: Теория и системы управления. — 1999. — № 5. — с.105-116.

- **Непомнящий В.А. и др.** Верификация Estelle-спецификаций распределенных систем посредством раскрашенных сетей Петри. — Новосибирск: Институт систем информатики СО РАН, 1998. — 140 стр.
- **Козюра В.Е., Непомнящий В.А., Новиков Р.М.** Верификация раскрашенных сетей Петри методом проверки моделей. — Новосибирск, 2001. — 24 стр. — (Препр. / СО РАН. ИСИ; № 89).
- **Козюра В.Е., Новиков Р.М.** Использование метода проверки моделей для верификации коммуникационных протоколов, представленных раскрашенными сетями Петри // Тез. докл. IV сибирского конгресса по прикладной и индустриальной математике (ИНПРИМ-2000). — Часть V. — Стр. 44.
- **Козюра В.Е., Новиков Р.М.** Верификация коммуникационных протоколов с использованием системы PNO // Тр. конф. Молодых ученых, ИВТ СО РАН, Новосибирск, 2001. — Т. 1. — Стр. 35—38.
- **Abdulla A.P., Purushothaman I., Nylen A.** Unfoldings of Unbounded Petri Nets // Proc. CAV 2000. — Berlin a.o.: Springer-Verlag, 2000. — P. 495—507. — (Lect. Notes Comput. Sci.; Vol. 1855).
- **Bieber B., Fleischhack H.** Model Checking of Time Petri Nets Based on Partial Order Semantics // Proc. CONCUR'99. — Berlin a.o.: Springer-Verlag, 1999. — P. 210—225. — (Lect. Notes Comput. Sci.; Vol. 1664).
- **Cheng A., Christensen S., Mortensen K. H.** Model Checking Coloured Petri Nets Exploiting Strongly Connected Components // DAIMI PB — 519, March 1997.
- **Couvreur J.-M., Grivet S., Poitrenaud D.** Designing an LTL Model-Checker Based on Unfolding Graphs // Lect. Notes Comput. Sci. — 2000. — Vol. 1825. — P. 123—145.
- **Engelfriet J.** Branching Processes of Petri Nets // Acta Informatica. — 1991. — Vol. 28. — P. 575—591.
- **Esparsa J.** Model-Checking Using Net Unfoldings // Lect. Notes Comput. Sci. — 1993. — Vol. 668. — P. 613—628.
- **Esparsa J., Römer S., Volger W.** An Improvement of McMillan's Unfolding Algorithm // Proc. TACAS'96. — Berlin a.o.: Springer-Verlag, 1997. — P. 87—106. — (Lect. Notes Comput. Sci.; Vol. 1055).
- **Esparsa J., Römer S.** An unfolding algorithm for synchronous product of transition systems // Proc. of CONCUR'99, — Berlin a.o.: Springer-Verlag, 1999. — P. 2—20. — (Lect. Notes Comput. Sci.; Vol. 1664).
- **Esparza J., Heljanko K.** A New Unfolding Approach to LTL Model-Checking // Lect. Notes Comput. Sci. — 2000. — Vol. 1853. — P. 475—486.

15. **Esparza J., Heljanko K.** Implementing LTL Model Checking with Net Unfoldings // Lect. Notes Comput. Sci. — 2001. — Vol. 2057. — P. 37—56.
16. **Gerth R., Peled D., Vardi M., Wolper P.** Simple On-the-fly Automatic Verification of Lineal Temporal Logic // Protocol Specification, Testing, and Verification, PSTV'95, 1995. — P. 3—18.

17. **Godefroid P., Wolper P.** Using partial orders for the efficient verification of deadlock freedom and safety properties // Formal Methods in System Design. — 1993— Vol. 2, N 2. — P. 149—164.
18. **Graves B.** Computing reachability properties hidden in finite net unfoldings. // Lect. Notes Comput. Sci. — 1997. — Vol. 1346. — P. 327—341.
19. **Heljanko K.** Model Checking with Finite Complete Prefixes is PSPACE-Complete // Proc. CONCUR'2000. — Berlin a.o.: Springer-Verlag, 2000. — P. 108—122. — (Lect. Notes Comput. Sci.; Vol. 1877).
20. **Jensen K.** Coloured Petri Nets. Vol. 1. — Berlin a.o.: Springer, 1995.
21. **Jensen K.** Coloured Petri Nets. Vol. 2. — Berlin a.o.: Springer, 1995.
22. **Kaivola R.** Using compositional preorders in the verification of sliding window protocol // Proc. CAV'97. — Berlin a.o.: Springer-Verlag, 1997. — P. 48—59. — (Lect. Notes Comput. Sci.; Vol. 1254).
23. **Kozura V.E.** Unfoldings of Coloured Petri Nets. — Novosibirsk, 2000. — 34 p. — (Prepr. /SB RAS. IIS; N 80).
24. **Kozura V.E.** Unfoldings of Coloured Petri Nets // 4th Intern. Conf. "Perspectives of System Informatics", PSI'01 / Preliminary Proc. — Novosibirsk, 2001. — P. 147 — 152.
25. **Kozura V.E.** Unfoldings of Coloured Petri Nets // Proc. PSI'01. — Berlin a.o.: Springer-Verlag, 2001. — P. 270—280. — (Lect. Notes Comput. Sci.; Vol. 2244).
26. **Kozura V.E.** Unfoldings of Timed Coloured Petri Nets. — Novosibirsk, 2001. — 33 p. —(Prepr. /SB RAS. IIS; N 82).
27. **Kozura V.E.** Unfoldings of Timed Coloured Petri Nets // Proc. Workshop CS&P'2001, Warsaw, October 3—5, 2001. — P. 128—139.
28. **Kozura V.E.** LTL model checking of coloured Petri nets based on net unfoldings // Joint Bull. of NCC&IIS. Ser.: Comp. Sci. — 2001. — № 14. — P. 83—101.
29. **Bodin E., Kozura V., Shilov N.** Experiments with model checking for μ-calculus in specification and verification project REAL // Proc. 5th New Zealand Formal Program Development Colloqium. — 1999. — P. 1—18. — (IIMS Tech. Rep. / New Zealand; No 99-1).
30. **Kondratyev A., Kishinevsky M., Taubin A., Ten S.** A Structural Approach for the Analysis of Petri Nets by Reduced Unfoldings // 17th Intern. Conf. on Application and Theory of Petri Nets, Osaka, June 1996. — Berlin a.o.: Springer-Verlag, 1996 — P. 346—365. — (Lect. Notes Comput. Sci.; Vol.1091).
31. **Lamport L.** What good is temporal logic? // Information Processing 83: Proc. / IFIP 9th World Computer Congr., Paris, France, September 19—23, 1983. — North Holland, 1983. — P. 657—668.
32. **McMillan K.L.** Using Unfolding to Avoid the State Explosion Problem in the Verification of Asynchronous Circuits // Lect. Notes Comput. Sci. — 1992. — Vol. 663. — P. 164—174.
33. **McMillan K.L.** A technique of a state space search based on unfoldings // Formal Methods in System Design. — 1995. — Vol. 6, N 1. — P. 45—65.

34. **Melzer S., Römer S.** Deadlock Checking Using Net Unfoldings // Proc. Conf. on Computer Aided Verification (CAV'97), Haifa, 1997. — Berlin a.o.: Springer-Verlag, 1997. — P. 352—363. — (Lect. Notes Comput. Sci.; Vol. 1254).
35. **Mueller-Olm M., Schmidt D., Stefen B.** Model-Checking. A Tutorial Introduction // Lect. Notes Comput. Sci. — 1999. — Vol. 1694. — P. 330—354.
36. **Schmidt K.** How to calculate symmetries of Petri nets // Acta Informatica. — 2000. — Vol. 36. — P. 545—590.
37. **Valmari A.** The State Explosion Problem // Lect. Notes Comput. Sci. — 1998. — Vol.1491. — P. 429—528.
38. **Valmari A.** Stubborn Sets of Coloured Petri Nets // Proc. 12th Intern. Conf. on Application and Theory of Petri Nets. — Gjern, 1991. — P. 102—121.
39. **Vardy M.Y., Wolper P.** An automata-theoretic approach to automatic program verification // Proc. 1st IEEE Symp. on Logic in Computer Science. — 1986. — P. 322—331.
40. **Wallner F.** Model-Checking LTL Using Net Unfoldings // Proc. Conf. on Computer Aided Verification (CAV'95), Vancouver, 1995. — Berlin a.o.: Springer-Verlag, 1998.— P. 207—218. — (Lect. Notes Comput. Sci.; Vol. 1427).
41. **R.Cleaveland, M.Klein, B.Steffen.** Faster Model Cheking for Modal Mu-Calculus // Proc Conf. on Computer Aided Verification (CAV'92), Montreal, Canada. — Berlin a.o.: Springer-Verlag, 1998.— P. 410—422 — (Lect. Notes Comput. Sci.; Vol. 663).
42. **Clarke E. M., Grumberg O., Peled D. A.** Model Checking. — The MIT Press Cambridge, Massachusetts London, England, 1999.
43. **Cohen R., Segall A.** An efficient reliable ring protocol // IEEE Transact. Communs. — 1991. — Vol.39, N.11. — P.1616—1624.
44. **Holzmann G. I.** Design and validation of computer protocols. — Englewood Cliffs, NJ: Prentice Hall, 1991.

В.Е. Козюра

РЕАЛИЗАЦИЯ СИСТЕМЫ ПРОВЕРКИ МОДЕЛЕЙ РАСКРАШЕННЫХ СЕТЕЙ ПЕТРИ С ИСПОЛЬЗОВАНИЕМ РАЗВЕРТОК

Препринт
94

Рукопись поступила в редакцию 17.01.02
Рецензент Ф. А. Мурзин
Редактор З. В. Скок

Подписано в печать 03.04.02
Формат бумаги 60 × 84 1/16 Объем 2.5 уч.-изд.л., 2.8 п.л.
Тираж 50 экз.

НФ ООО ИПО “Эмари” РИЦ, 630090, г. Новосибирск, пр. Акад. Лаврентьева, 6