ГОСКОМИТЕТ РОССИИ ПО ДЕЛАМ
НАУКИ И ВЫСШЕЙ ШКОЛЫ
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ

МЕХАНИКО-МАТЕМАТИЧЕСКИЙ
ФАКУЛЬТЕТ
Кафедра вычислительных систем

Тарасюк Игорь Валерьевич

Эквивалентности на сетях Петри

Научный руководитель
к.ф.-м.н., доцент
Вирбицкайте И.Б.

Новосибирск 1994

Содержание

Глава 1

Введение

В литературе, посвященной сетям Петри, был введен широкий класс семантических эквивалентностей. Основные из них можно изобразить точками на координатной плоскости, представленной на рисунке 1.1. Движению вправо по оси X будет соответствовать возрастание степени детализации процессов сетей. При движении вверх по оси Y будет возрастать степень взаимного моделирования.

Известны следующие точки на оси X.

Последовательная семантика. Другое название этой семантики — интерливинговая. Выполнение процесса представляется последовательностями срабатывания действий.

Шаговая семантика. Выполнение процесса моделируется последовательностями срабатывания мультимножеств действий.

Семантика частичных слов. Выполнению процесса соответствует срабатывание частично упорядоченного мультимножества действий (ЧУММ). В этой семантике один процесс может моделировать другой, если отношение причинной зависимости между действиями в его ЧУММ менее строгое или такое же, как в ЧУММ второго процесса.

Семантика ЧУММ. Аналогична семантике частичных слов, за исключением того, что при моделировании должно сохраняться отношение зависимости на действиях в ЧУММ.

Процессно-сетевая семантика. Выполнение процесса представляется срабатыванием ациклической бесконфликтной сети (так называемой C-сети). Для краткости будем называть эквивалентности, связанные с этой семантикой, "процессными".

По оси Y откладываются такие точки.

Обычные эквивалентности. Процесс определяется множеством возможных выполнений, и недетерминизм не учитывается (то есть не берется в расчет место, где происходит недетерминированный выбор между несколькими расширениями процесса).

Бисимуляционные эквивалентности. Принимается во внимание недетерминизм.

ST-бисимуляционные эквивалентности. Предполагается, что действия имеют некоторую внутреннюю структуру (или выполняются не мгновенно, а за определенный промежуток времени).

Сохраняющие историю бисимуляционные эквивалентности. Эти эквивалентности учитывают "прошлое" процессов, то есть то *как* новый, расширяющий процесс соединяется с тем, выполнение которого привело в данное состояние.

Посмотрим, какие упомянутые в литературе эквивалентности заполняют эту координатную плоскость.

- Рассмотрим обычные эквивалентности.
  - Самая простая эквивалентность, лежащая в начале системы координат — это языковая (последовательная, интерливинговая). Она обозначается символом $\equiv_i$. Две сети последовательно эквивалентны, если они имеют одни и те же языки. Эквивалентность определяется в [15] на структурах событий.
  - Шаговая эквивалентность (обозначение $\equiv_s$) связывает сети, имеющие одинаковые множества шаговых следов (то есть последовательностей мультимножеств действий). Определение для структур событий можно найти в [15].
  - Эквивалентность на ЧУММ (обозначается $\equiv_{pom}$) связывает сети, процессы которых имеют одинаковые ЧУММ действий. Эта эквивалентность определена в [16] для сетей Петри и в [15] для структур событий.
- Среди бисимуляционных эквивалентностей известны следующие.
  - Последовательная бисимуляционная эквивалентность (обозначение $\underline{\leftrightarrow}_i$) была введена в [23] на автоматах. Определение можно найти также в [1, 2, 4, 5, 16] для сетей Петри и в [14, 15, 26, 27] для структур событий.

- Шаговая бисимуляционная эквивалентность (обозначение $\leftrightarrows_s$) введена в [22]. Определение на сетях Петри имеется в [1, 2, 16], а на структурах событий — в [14, 15, 26, 27].
- Бисимуляционная эквивалентность на частичных словах (обозначение $\leftrightarrows_{pw}$) введена в [26] на структурах событий. Определение для сетей Петри можно найти в [2], а для структур событий — также в [27].
- Бисимуляционная эквивалентность на ЧУММ (обозначение $\leftrightarrows_{pom}$) была введена в [4, 16]. Определения для сетей Петри даны в [1, 2, 5], а для структур событий — в [14, 15, 26, 27].
- Процессная бисимуляционная эквивалентность (обозначение $\leftrightarrows_{pr}$) введена в [2] на сетях Петри.

• Известны следующие ST-бисимуляционные эквивалентности.

- Последовательная ST-бисимуляционная эквивалентность (обозначение$\leftrightarrows_{iST}$) была введена на сетях Петри в [16]. Определение для сетей Петри можно также найти в [4], а для структур событий — в [15, 26, 27]. Заметим, что последовательная ST-бисимуляционная эквивалентность совпадает с соответствующей шаговой эквивалентностью.
- ST-бисимуляционная эквивалентность на частичных словах (обозначение $\leftrightarrows_{pwST}$) введена в [26] на структурах событий. Это определение можно также найти в [27].
- ST-бисимуляционная эквивалентность на ЧУММ (обозначение $\leftrightarrows_{pomST}$) также введена в [26] на структурах событий. Определение есть также в [27].

• В литературе была рассмотрена сохраняющая историю бисимуляционная эквивалентность (обозначение $\leftrightarrows_{pomh}$). Она введена в [25] на поведенческих структурах под именем "бисимуляционная эквивалентность поведенческих структур". В [14] эта эквивалентность была определена на структурах событий и названа " сохраняющая историю бисимуляционная эквивалентность". В [5] определение было введено на сетях Петри. В этой работе эквивалентности было дано имя "полностью параллельная бисимуляционная эквивалентность". Определение для сетей Петри можно также найти в [4, 13], а для структур событий — в [14, 15, 26, 27].

В данной работе был введен ряд новых эквивалентностей, дополняющих уже известные. Эти эквивалентности также изображены

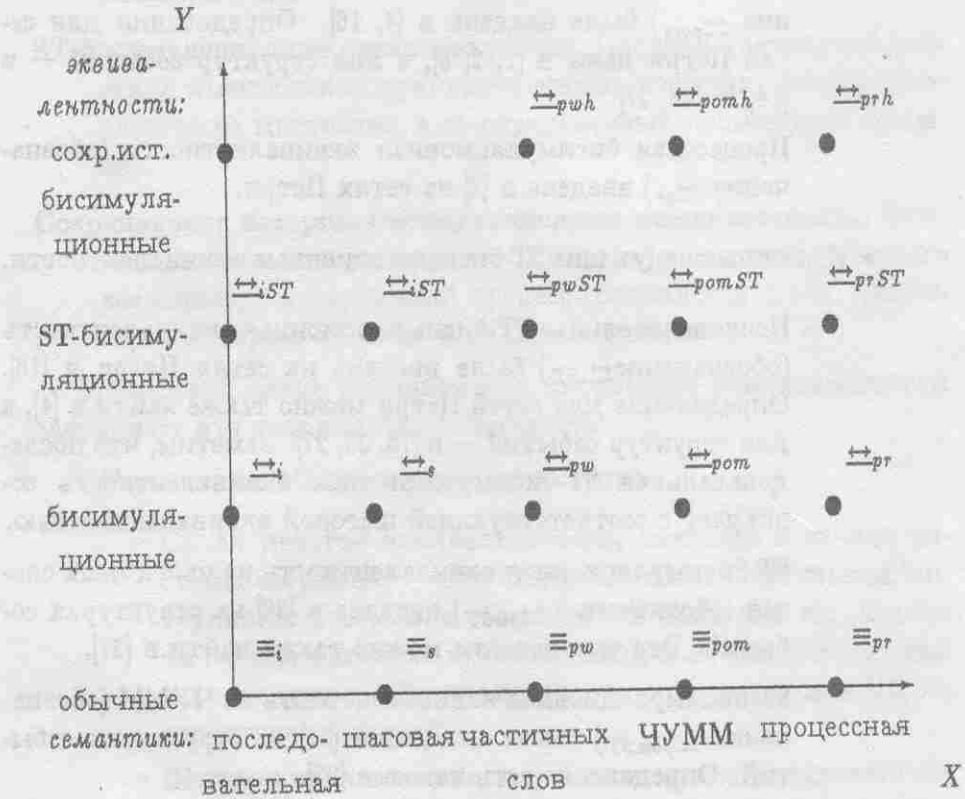


Рисунок 1.1: Классификация эквивалентностей

на рисунке 1.1. В работе рассмотренна взаимосвязь между всеми упомянутыми эквивалентностями и построена решетка связей.

Кроме того, введенные эквивалентности были рассмотрены на трех классах сетей Петри: на последовательных и С-сетях, а также на сетях со строгой пометкой. Показано, что процессные эквивалентности хорошо различают последовательные сети, в отличие от остальных эквивалентностей, большинство из которых сливается на этом классе сетей.

В работе рассмотрено, как связаны эквивалентности на формулах алгебры параллельных недетерминированных процессов AFP_2 с эквивалентностями на сетях и предложен способ автоматической проверки эквивалентности формул этой алгебры.

Сделаем краткий обзор содержания работы. Глава 2 посвящена исследованию сетевых эквивалентностей. В пункте 1 этой главы даются основные определения, используемые в дальнейшем. Пункт 2 рассказывает об обычных эквивалентностях на сетях, а пункт 3 — о бисимуляционных эквивалентностях. В пункте 4 доказывается теорема, устанавливающая взаимосвязь между всеми введенными эквивалентностями. В пункте 5 эквивалентности исследуются на различных классах сетей. Глава 3 посвящена эквивалентностям на формулах процессов. В пункте 1 дается краткий обзор алгебры AFP_2. В пункте 2 рассматриваются эквивалентности на формулах и устанавливается их взаимосвязь с сетевыми эквивалентностями. Пункт 3 рассказывает о системе правил переписывания RWS_2, используемой для автоматизации проверки формул на эквивалентность. Заключительная глава 4 содержит идеи по развитию данной работы. В приложении А описывается структура и принципы работы программы CANON, а в приложении В приводятся примеры преобразования формул AFP_2 к каноническому виду с помощью этой программы.

Глава 2

Исследование сетевых эквивалентностей

2.1 Основные определения

2.1.1 Мультимножества

Пусть X — некоторое множество. *Мультимножество* M над X — отображение $M : X \to \mathcal{N}$, где $\mathcal{N}$ — множество натуральных чисел. Для $x \in X$ $M(x)$ — *кратность* x в M. Пишем $x \in M$, если $M(x) > 0$.

Когда $\forall x \in X\ M(x) \leq 1$, M — обычное множество. M *конечно*, если $M(x) = 0$ для всех $x \in X$, за исключением, быть может, конечного их числа. *Мощность* мультимножества M определяется так: $|M| = \sum_{x \in X} M(x)$. В дальнейшем будем рассматривать только конечные мультимножества. Обозначим через $\mathcal{M}(X)$ *множество всех конечных мультимножеств* над X.

Понятия теории множеств стандартно расширяются на конечные мультимножества. Если $M, M' \in \mathcal{M}(X)$, определим $M+M'$ так: $(M+M')(x) = M(x)+M'(x)$. Пишем $M \subseteq M'$, если $\forall x \in X\ M(x) \leq M'(x)$. Когда $M' \subseteq M$, определяем $M - M'$ так: $(M - M')(x) = M(x) - M'(x)$. Запись $M + x - y$ используется вместо $M + \{x\} - \{y\}$. Символ $\emptyset$ обозначает пустое мультимножество.

2.1.2 Маркированные сети

Помеченная сеть — четверка $N = \langle P_N, T_N, F_N, l_N \rangle$, где :

- $P_N = \{p, q, \ldots\}$ — множество мест;
- $T_N = \{u, v, \ldots\}$ — множество переходов;

- $F_N : (P_N \times T_N) \cup (T_N \times P_N) \to \mathcal{N}$ — отношение инцидентности с весами;
- $l_N : T_N \to Act$ — пометка переходов именами действий.

Предполагается, что $P_N \cap T_N = \emptyset$.

Пусть N — помеченная сеть. *Маркировка* N есть мультимножество $M \in \mathcal{M}(P_N)$.

Маркированная сеть — пятерка $N = \langle P_N, T_N, F_N, l_N, M_N \rangle$ такая, что $\langle P_N, T_N, F_N, l_N \rangle$ — помеченная сеть, а $M_N \in \mathcal{M}(P_N)$ — начальная маркировка. Будем писать "сеть" вместо "маркированная сеть". Пусть дана сеть N и некоторый переход $u \in T_N$. *Предусловие* и *постусловие* u, обозначаемые соответственно ${}^\bullet u$ и $u^\bullet$, это мультимножества, которые определяются следующим образом: $({}^\bullet u)(p) = F_N(p,u)$ и $(u^\bullet)(p) = F_N(u,p)$. Аналогичные понятия введем для мест: $({}^\bullet p)(u) = F_N(u,p)$ и $(p^\bullet)(u) = F_N(p,u)$. Переход u *нестабильный*, если ${}^\bullet u = \emptyset$. Сеть *стабильна*, если в ней нет нестабильных переходов. Сеть *конечна*, если $P_N \cup T_N$ — конечное множество.

Пусть $M \in \mathcal{M}(P_N)$ — маркировка сети N. Переход $u \in T_N$ *допустим* в M, если ${}^\bullet u \subseteq M$. Если u допустим в M, то срабатывание этого перехода изменяет маркировку M на $M' = M - {}^\bullet u + u^\bullet$, запись $M \xrightarrow{u} M'$ или $M \xrightarrow{a} M'$, если $l_N(u) = a$. Пишем $M \to M'$, если $M \xrightarrow{u} M'$ для некоторого u.

2.1.3 Процессы

Помеченная C-сеть — четверка $C = \langle P_C, T_C, F_C, l_C \rangle$, где:

1. $\forall v \in T_C$ ${}^\bullet v$ и $v^\bullet$ — обычные множества;
2. места не разветвлены, то есть $\forall p \in P_C$ $|{}^\bullet p| \leq 1$ и $|p^\bullet| \leq 1$;
3. F_C хорошо сформировано, то есть в F_C не существует бесконечных в обратную сторону цепочек следующего вида: $\cdots (p_n, v_n)(v_n, p_{n-1}) \cdots (p_1, v_1)(v_1, p_0)$.

Введем следующие обозначения: ${}^\circ C = \{p \in P_C | {}^\bullet p = \emptyset\}$ — множество *начальных* мест C и $C^\circ = \{p \in P_C | p^\bullet = \emptyset\}$ — множество *конечных* мест C. Известно фундаментальное свойство C-сетей: для C-сети C существует последовательность срабатываний переходов ${}^\circ C = L_0 \xrightarrow{v_1} \cdots \xrightarrow{v_n} L_n = C^\circ$ такая, что $L_i \subseteq P_C$ $(0 \leq i \leq n)$, $P_C = \cup_{i=0}^n L_i$ и $T_C = \{v_1, \ldots, v_n\}$. Такую последовательность назовем *полным выполнением* C.

Даны сеть N и помеченная C-сеть C. Отображение $f : P_C \cup T_C \to P_N \cup T_N$ — *встраивание* C в N, обозначение $f : C \to N$, если:

1. $f(P_C) \in \mathcal{M}(P_N)$ и $f(T_C) \in \mathcal{M}(T_N)$;

2. $\forall v \in T_C\ l_C(v) = l_N(f(v))$;

3. $\forall v \in T_C\ {}^\bullet f(v) = f({}^\bullet v)$ и $f(v)^\bullet = f(v^\bullet)$.

Условие 3 говорит, что встраивание учитывает отношение инцидентности. Следовательно, если ${}^\circ C \xrightarrow{v_1} \dots \xrightarrow{v_n} C^\circ$ — полное выполнение C, то $M = f({}^\circ C) \xrightarrow{f(v_1)} \dots \xrightarrow{f(v_n)} f(C^\circ) = M'$ — последовательность срабатываний переходов в N, *соответствующая* этому полному выполнению, запись $M \xrightarrow{C,f} M'$. Обратно, для любой последовательности срабатываний $M \xrightarrow{u_1} \dots \xrightarrow{u_n} M'$ сети N существуют помеченная C-сеть C и встраивание $f : C \to N$ такие, что $M = f({}^\circ C)$, $M' = f(C^\circ)$, $u_i = f(v_i)$ $(0 \le i \le n)$ и ${}^\circ C \xrightarrow{v_1} \dots \xrightarrow{v_n} C^\circ$ — полное выполнение C.

Процесс, допустимый в маркировке M сети N — пара $\pi = (C, f)$, где C — помеченная C-сеть (в этом случае будем для краткости писать просто "C-сеть"), а $f : C \to N$ — встраивание такое, что $M = f({}^\circ C)$. Процесс, допустимый в M_N , — *процесс* N. Обозначим *множество всех процессов* N, *допустимых в* M как $\Pi(N, M)$, а *множество процессов* N через $\Pi(N)$. В дальнейшем будем рассматривать только стабильные сети с *конечными* процессами, то есть с процессами, имеющими конечные C-сети.

Если $\pi \in \Pi(N, M)$, то срабатывание этого процесса меняет маркировку M на $M' = M - f({}^\circ C) + f(C^\circ) = f(C^\circ)$, обозначение $M \xrightarrow{\pi} M'$. C-сеть задает порядок на переходах (отношение *предшествования, причинной зависимости*) $\prec_C$, определяемый так: $\prec_C = F_C^+ \lceil_{T_C \times T_C}$, где F_C^+ — транзитивное замыкание F_C. *Начальный* процесс сети N — это $\pi_N = (C_N, f_N) \in \Pi(N)$, где $T_{C_N} = \emptyset$. Пусть $\pi = (C, f)$, $\tilde{\pi} = (\tilde{C}, \tilde{f}) \in \Pi(N)$, $\hat{\pi} = (\hat{C}, \hat{f}) \in \Pi(N, f(C^\circ))$, $C = \langle P_C, T_C, F_C, l_C \rangle$, $\tilde{C} = \langle P_{\tilde{C}}, T_{\tilde{C}}, F_{\tilde{C}}, l_{\tilde{C}} \rangle$, $\hat{C} = \langle P_{\hat{C}}, T_{\hat{C}}, F_{\hat{C}}, l_{\hat{C}} \rangle$.

Пишем $\pi \xrightarrow{\hat{\pi}} \tilde{\pi}$, если:

1. $P_C \cup P_{\hat{C}} = P_{\tilde{C}}$; $T_C \cup T_{\hat{C}} = T_{\tilde{C}}$; $F_C \cup F_{\hat{C}} = F_{\tilde{C}}$; $l_C \cup l_{\hat{C}} = l_{\tilde{C}}$;

2. $f \cup \hat{f} = \tilde{f}$.

В этом случае говорят, что $\tilde{\pi}$ — расширение π *на процесс* $\hat{\pi}$, а $\hat{\pi}$ — *расширяющий* процесс для π. Заметим, что для любого $\pi \in \Pi(N)$ $\pi_N \xrightarrow{\pi} \pi$. Будем писать $\pi \to \tilde{\pi}$, если $\pi \xrightarrow{\hat{\pi}} \tilde{\pi}$ для некоторого расширяющего процесса $\hat{\pi}$.

$\tilde{\pi}$ — расширение π *на одно действие*, если $\pi \to \tilde{\pi}$ и $|T_{\tilde{C}} \setminus T_C| = 1$. В этом случае пишем $\pi \xrightarrow{v} \tilde{\pi}$ или $\pi \xrightarrow{a} \tilde{\pi}$, если $T_{\tilde{C}} \setminus T_C = \{v\}$ и $l_{\tilde{C}}(v) = a$.

Таким образом, полное выполнение C в "контексте" сети N можно записать с помощью процессов как $\pi_0 \xrightarrow{v_1} \dots \xrightarrow{v_n} \pi_n$, где

$\pi_i = (C_i, f_i)$, $C_i^\circ = L_i$, $(0 \le i \le n)$ и ${}^\circ C = L_0 \xrightarrow{v_1} \cdots \xrightarrow{v_n} L_n = C^\circ$ — полное выполнение C.

$\tilde{\pi}$ — расширение π *на мультимножество действий*, или *шаг*, если $\pi \xrightarrow{\hat{\pi}} \tilde{\pi}$ и $\prec_{\hat{C}} = \emptyset$. В этом случае пишем $\pi \xrightarrow{A} \tilde{\pi}$, когда $l_{\hat{C}}(T_{\hat{C}}) = A$, $A \in \mathcal{M}(Act)$.

2.1.4 Отображения

Даны сети $N = \langle P_N, T_N, F_N, l_N, M_N \rangle$ и $N' = \langle P_{N'}, T_{N'}, F_{N'}, l_{N'}, M_{N'} \rangle$. Назовем β *отображением* N *в* N', запись $\beta : N \to N'$, если $\beta : P_N \cup T_N \to P_{N'} \cup T_{N'}$, $\beta(P_N) \subseteq P_{N'}$ и $\beta(T_N) \subseteq T_{N'}$. Пишем $\beta(N) = N'$, когда $\beta(P_N) = P_{N'}$ и $\beta(T_N) = T_{N'}$.

Отображение $\beta : N \to N'$ — *сохраняющая пометку биекция* между N и N', обозначение $\beta : N \approx N'$, если:

1. β — биекция и $\beta(N) = N'$;
2. $\forall u \in T_N \; l_N(u) = l_{N'}(\beta(u))$.

Пишем $N \approx N'$, если существует сохраняющая пометку биекция $\beta : N \approx N'$.

Отображение $\beta : N \to N'$ — *изоморфизм* между N и N', обозначение $\beta : N \simeq N'$, если:

1. $\beta : N \approx N'$;
2. $\forall u \in T_N \; {}^\bullet\beta(u) = \beta({}^\bullet u)$ и $\beta(u)^\bullet = \beta(u^\bullet)$.

Сети N и N' *изоморфны*, обозначение $N \simeq N'$, если существует изоморфизм $\beta : N \simeq N'$.

Имеются две помеченные С-сети $C = \langle P_C, T_C, F_C, l_C \rangle$ и $C' = \langle P_{C'}, T_{C'}, F_{C'}, l_{C'} \rangle$.

Отображение $\beta : T_C \to T_{C'}$ — *сохраняющая пометку биекция* между T_C и $T_{C'}$, обозначение $\beta : T_C \approx T_{C'}$, если:

1. β — биекция и $\beta(T_C) = T_{C'}$;
2. $\forall v \in T_C \; l_C(v) = l_{C'}(\beta(v))$.

Пишем $T_C \approx T_{C'}$, если существует сохраняющая пометку биекция $\beta : T_C \approx T_{C'}$.

Отображение $\beta : T_C \to T_{C'}$ — *гомоморфизм* между T_C и $T_{C'}$, обозначение $\beta : T_C \sqsubseteq T_{C'}$, если:

1. $\beta : T_C \approx T_{C'}$;
2. $\forall v, w \in T_C \; v \prec_C w \Rightarrow \beta(v) \prec_{C'} \beta(w)$.

Пишем $T_C \sqsubseteq T_{C'}$, если существует гомоморфизм $\beta : T_C \sqsubseteq T_{C'}$.

Отображение $\beta : T_C \to T_{C'}$ — *изоморфизм* между T_C и $T_{C'}$, обозначение $\beta : T_C \simeq T_{C'}$, если $\beta : T_C \sqsubseteq T_{C'}$ и $\beta^{-1} : T_{C'} \sqsubseteq T_C$. Пишем $T_C \simeq T_{C'}$, если существует изоморфизм $\beta : T_C \simeq T_{C'}$.

2.2 Обычные эквивалентности на сетях

Последовательный след сети N — это последовательность $a_1 \cdots a_n \in Act^*$ такая, что $\pi_N \stackrel{a_1}{\to} \pi_1 \stackrel{a_2}{\to} \ldots \stackrel{a_n}{\to} \pi_n$, где $\pi_i \in \Pi(N)$ $(1 \leq i \leq n)$, а π_N — начальный процесс N. Обозначим *множество всех последовательных следов* N $SeqTraces(N)$. Две сети N и N' *последовательно эквивалентны*, обозначение $N \equiv_i N'$, если $SeqTraces(N) = SeqTraces(N')$.

Шаговый след сети N — это последовательность $A_1 \cdots A_n \in (\mathcal{M}(Act))^*$ такая, что $\pi_N \stackrel{A_1}{\to} \pi_1 \stackrel{A_2}{\to} \ldots \stackrel{A_n}{\to} \pi_n$, где $\pi_i \in \Pi(N)$ $(0 \leq i \leq n)$, а π_N — начальный процесс N. Обозначим *множество всех шаговых следов* N $StepTraces(N)$. Две сети N и N' *шагово эквивалентны*, обозначение $N \equiv_s N'$, если $StepTraces(N) = StepTraces(N')$.

ЧУММ-след сети N — это ЧУММ ρ — класс изоморфизма T_C для $\pi = (C, f) \in \Pi(N)$, где $C = (P_C, T_C, F_C, l_C)$. Пишем $\rho \sqsubseteq \rho'$, если $T_C \sqsubseteq T_{C'}$ для $T_C \in \rho$ и $T_{C'} \in \rho'$. В этом случае говорим, что ЧУММ ρ *менее последователен* или *более параллелен*, чем ρ'. Обозначим через $Pomsets(N)$ *множество всех следов ЧУММ* N. Две сети N и N' *эквивалентны на частичных словах*, запись $N \equiv_{pw} N'$, если $Pomsets(N) \sqsubseteq Pomsets(N')$ и $Pomsets(N') \sqsubseteq Pomsets(N)$, то есть для любого $\rho' \in Pomsets(N')$ существует $\rho \in Pomsets(N)$ такой, что $\rho \sqsubseteq \rho'$ и наоборот. Две сети N и N' *эквивалентны на ЧУММ*, обозначение $N \equiv_{pom} N'$,если $Pomsets(N) = Pomsets(N')$.

Процессный след сети N — это класс изоморфизма C для $\pi = (C, f) \in \Pi(N)$. $ProcessNets(N)$ обозначает *множество всех процессных следов* N. Две сети N и N' *процессно эквивалентны*, запись $N \equiv_{pr} N'$, если $ProcessNets(N) = ProcessNets(N')$.

2.3 Бисимуляционные эквивалентности

Бисимуляция — это фундаментальная поведенческая эквивалентность. Чтобы две сети были бисимуляционно эквивалентны, должно существовать некоторое отношение R (бисимуляция) на их состояниях такое, что:

- Начальные состояния обеих сетей связаны отношением R.
- Если сети находятся в состояниях, связанных отношением R, и одна из них перешла в новое состояние, то другая сеть

может смоделировать поведение первой сети, также перейдя в новое состояние. При этом новые состояния сетей должны быть тоже связаны отношением R.

Состояниями сетей могут быть, например , маркировки или процессы. Существуют и другие виды состояний (например, ST-процессы, которые мы рассмотрим позже).

Запись $R : N \underline{\leftrightarrow}_{\alpha} N'$ будет означать, что R — бисимуляция типа α между сетями N и N'. Сети N и N' назовем α-бисимуляционно эквивалентными, обозначение $N \underline{\leftrightarrow}_{\alpha} N'$, если $R : N \underline{\leftrightarrow}_{\alpha} N'$ для некоторой α-бисимуляции R.

2.3.1 Обычные бисимуляции

Пусть $R \subseteq \Pi(N) \times \Pi(N')$. В следующих определениях $\hat{\pi} = (\hat{C}, \hat{f})$, $\hat{\pi}' = (\hat{C}', \hat{f}')$.

R — *последовательная бисимуляция* между N и N', запись $R : N \underline{\leftrightarrow}_{;} N'$, если:

1. $(\pi_N, \pi_{N'}) \in R$;
2. $(\pi, \pi') \in R$, $\pi \xrightarrow{a} \tilde{\pi}$, $a \in Act \Rightarrow \exists \tilde{\pi}' : \pi' \xrightarrow{a} \tilde{\pi}'$ и $(\tilde{\pi}, \tilde{\pi}') \in R$;
3. Как предыдущий пункт, но роли N и N' меняются.

R — *шаговая бисимуляция* между N и N', запись $R : N \underline{\leftrightarrow}_{s} N'$, если:

1. $(\pi_N, \pi_{N'}) \in R$;
2. $(\pi, \pi') \in R$, $\pi \xrightarrow{A} \tilde{\pi}$, $a \in \mathcal{M}(Act) \Rightarrow \exists \tilde{\pi}' : \pi' \xrightarrow{A} \tilde{\pi}'$ и $(\tilde{\pi}, \tilde{\pi}') \in R$;
3. Как предыдущий пункт, но роли N и N' меняются.

R — *бисимуляция на частичных словах* между N и N', запись $R : N \underline{\leftrightarrow}_{pw} N'$, если:

1. $(\pi_N, \pi_{N'}) \in R$;
2. $(\pi, \pi') \in R$, $\pi \xrightarrow{\hat{\pi}} \tilde{\pi}$, $\Rightarrow \exists \tilde{\pi}' : \pi' \xrightarrow{\hat{\pi}'} \tilde{\pi}'$, $T_{\hat{C}'} \sqsubseteq T_{\hat{C}}$ и $(\tilde{\pi}, \tilde{\pi}') \in R$;
3. Как предыдущий пункт, но роли N и N' меняются.

R — *бисимуляция на ЧУММ* между N и N', запись $R : N \underline{\leftrightarrow}_{pom} N'$, если:

1. $(\pi_N, \pi_{N'}) \in R$;
2. $(\pi, \pi') \in R$, $\pi \xrightarrow{\hat{\pi}} \tilde{\pi}$, $\Rightarrow \exists \tilde{\pi}' : \pi' \xrightarrow{\hat{\pi}'} \tilde{\pi}'$, $T_{\hat{C}} \simeq T_{\hat{C}'}$ и $(\tilde{\pi}, \tilde{\pi}') \in R$;

3. Как предыдущий пункт, но роли N и N' меняются.

R — *процессная бисимуляция* между N и N', запись $R : N \underline{\leftrightarrow}_{pr} N'$, если:

1. $(\pi_N, \pi_{N'}) \in R$;

2. $(\pi, \pi') \in R$, $\pi \xrightarrow{\hat{\pi}} \tilde{\pi}$, $\Rightarrow$ $\exists \tilde{\pi}' : \pi' \xrightarrow{\hat{\pi}'} \tilde{\pi}'$, $\hat{C} \simeq \hat{C}'$ и $(\tilde{\pi}, \tilde{\pi}') \in R$;

3. Как предыдущий пункт, но роли N и N' меняются.

2.3.2 ST-процессы

ST-процессы вводятся для описания состояний временных сетей, в которых каждый переход (и помечающее его действие) срабатывает не мгновенно, а, начав работать, работает некоторое время, а затем завершается.

ST-процесс сети N — пара (π_E, π_P) такая, что $\pi_E, \pi_P \in \Pi(N)$, $\pi_P \xrightarrow{\pi_W} \pi_E$ и $\forall v, w \in T_{C_E}\ v \prec_{C_E} w \Rightarrow v \in T_{C_P}$ В этом случае π_E — процесс, начавший работать, то есть все действия π_E начали выполняться. Процесс π_P соответствует той части π_E, которая уже завершилась, а π_W — той части, которая еще работает. Очевидно, что $\prec_{C_W} = \emptyset$. $ST - \Pi(N)$ означает *множество всех ST-процессов* N.

(π_N, π_N) будет *начальным ST-процессом* N. Пусть (π_E, π_P), $(\tilde{\pi}_E, \tilde{\pi}_P) \in ST - \Pi(N)$. Пишем $(\pi_E, \pi_P) \to (\tilde{\pi}_E, \tilde{\pi}_P)$, если $\pi_E \to \tilde{\pi}_E$ и $\pi_P \to \tilde{\pi}_P$.

2.3.3 ST-бисимуляции

Пусть $R \subseteq ST - \Pi(N) \times ST - \Pi(N') \times \mathcal{B}$, где $\mathcal{B} = \{\beta | \beta : T_C \to T_{C'},\ \pi = (C, f) \in \Pi(N),\ \pi' = (C', f') \in \Pi(N')\}$, а в определении процессной ST-бисимуляции $\mathcal{B} = \{\beta | \beta : C \to C',\ \pi = (C, f) \in \Pi(N),\ \pi' = (C', f') \in \Pi(N')\}$.

В следующих определениях $\pi_E = (C_E, f_E)$, $\pi_P = (C_P, f_P)$, $\pi'_E = (C'_E, f'_E)$, $\pi'_P = (C'_P, f'_P)$, $\pi = (C, f)$, $\pi' = (C', f')$.

R — *последовательная ST-бисимуляция* между N и N', запись $R : N \underline{\leftrightarrow}_{ST} N'$, если:

1. $((\pi_N, \pi_N), (\pi_{N'}, \pi_{N'}), \emptyset) \in R$;

2. $((\pi_E, \pi_P), (\pi'_E, \pi'_P), \beta) \in R \Rightarrow \beta : T_{C_E} \approx T_{C'_E}$ и $\beta(T_{C_P}) = T_{C'_P}$;

3. $((\pi_E, \pi_P), (\pi'_E, \pi'_P), \beta) \in R$, $(\pi_E, \pi_P) \to (\tilde{\pi}_E, \tilde{\pi}_P) \Rightarrow \exists \tilde{\beta}$, $(\tilde{\pi}'_E, \tilde{\pi}'_P) :$ $(\pi'_E, \pi'_P) \to (\tilde{\pi}'_E, \tilde{\pi}'_P)$, $\tilde{\beta}\lceil_{T_{C_E}} = \beta$ и $((\tilde{\pi}_E, \tilde{\pi}_P), (\tilde{\pi}'_E, \tilde{\pi}'_P), \tilde{\beta}) \in R$;

4. Как предыдущий пункт, но роли N и N' меняются.

R — *ST-бисимуляция на частичных словах* между N и N', запись $R : N \underline{\leftrightarrow}_{pwST} N'$, если:

1. $((\pi_N, \pi_N), (\pi_{N'}, \pi_{N'}), \emptyset) \in R$;
2. $((\pi_E, \pi_P), (\pi'_E, \pi'_P), \beta) \in R \Rightarrow \beta : T_{C_E} \approx T_{C'_E}$ и $\beta(T_{C_P}) = T_{C'_P}$;
3. $((\pi_E, \pi_P), (\pi'_E, \pi'_P), \beta) \in R$, $(\pi_E, \pi_P) \to (\tilde{\pi}_E, \tilde{\pi}_P) \Rightarrow \exists \tilde{\beta}$, $(\tilde{\pi}'_E, \tilde{\pi}'_P)$: $(\pi'_E, \pi'_P) \to (\tilde{\pi}'_E, \tilde{\pi}'_P)$, $\tilde{\beta}\lceil_{T_{C_E}} = \beta$, $\tilde{\beta}^{-1} : T_{C'} \sqsubseteq T_C$, где $\pi_P \xrightarrow{\pi} \tilde{\pi}_E$, $\pi'_P \xrightarrow{\pi'} \tilde{\pi}'_E$ и $((\tilde{\pi}_E, \tilde{\pi}_P), (\tilde{\pi}'_E, \tilde{\pi}'_P), \tilde{\beta}) \in R$;
4. Как предыдущий пункт, но роли N и N' меняются.

R — *ST-бисимуляция на ЧУММ* между N и N', запись $R : N \underline{\leftrightarrow}_{pomST} N'$, если:

1. $((\pi_N, \pi_N), (\pi_{N'}, \pi_{N'}), \emptyset) \in R$;
2. $((\pi_E, \pi_P), (\pi'_E, \pi'_P), \beta) \in R \Rightarrow \beta : T_{C_E} \approx T_{C'_E}$ и $\beta(T_{C_P}) = T_{C'_P}$;
3. $((\pi_E, \pi_P), (\pi'_E, \pi'_P), \beta) \in R$, $(\pi_E, \pi_P) \to (\tilde{\pi}_E, \tilde{\pi}_P) \Rightarrow \exists \tilde{\beta}$, $(\tilde{\pi}'_E, \tilde{\pi}'_P)$: $(\pi'_E, \pi'_P) \to (\tilde{\pi}'_E, \tilde{\pi}'_P)$, $\tilde{\beta}\lceil_{T_{C_E}} = \beta$, $\tilde{\beta} : T_C \simeq T_{C'}$, где $\pi_P \xrightarrow{\pi} \tilde{\pi}_E$, $\pi'_P \xrightarrow{\pi'} \tilde{\pi}'_E$ и $((\tilde{\pi}_E, \tilde{\pi}_P), (\tilde{\pi}'_E, \tilde{\pi}'_P), \tilde{\beta}) \in R$;
4. Как предыдущий пункт, но роли N и N' меняются.

R — *процессная ST-бисимуляция* между N и N', запись $R : N \underline{\leftrightarrow}_{prST} N'$, если:

1. $((\pi_N, \pi_N), (\pi_{N'}, \pi_{N'}), \emptyset) \in R$;
2. $((\pi_E, \pi_P), (\pi'_E, \pi'_P), \beta) \in R \Rightarrow \beta : C_E \approx C'_E$ и $\beta(C_P) = C'_P$;
3. $((\pi_E, \pi_P), (\pi'_E, \pi'_P), \beta) \in R$, $(\pi_E, \pi_P) \to (\tilde{\pi}_E, \tilde{\pi}_P) \Rightarrow \exists \tilde{\beta}$, $(\tilde{\pi}'_E, \tilde{\pi}'_P)$: $(\pi'_E, \pi'_P) \to (\tilde{\pi}'_E, \tilde{\pi}'_P)$, $\tilde{\beta}\lceil_{P_{C_E} \cup T_{C_E}} = \beta$, $\tilde{\beta} : C \simeq C'$, где $\pi_P \xrightarrow{\pi} \tilde{\pi}_E$, $\pi'_P \xrightarrow{\pi'} \tilde{\pi}'_E$ и $((\tilde{\pi}_E, \tilde{\pi}_P), (\tilde{\pi}'_E, \tilde{\pi}'_P), \tilde{\beta}) \in R$;
4. Как предыдущий пункт, но роли N и N' меняются.

2.3.4 Сохраняющие историю бисимуляции

Пусть $R \subseteq \Pi(N) \times \Pi(N') \times \mathcal{B}$, где $\mathcal{B} = \{\beta | \beta : T_C \to T_{C'}, \pi = (C, f) \in \Pi(N), \pi' = (C', f') \in \Pi(N')\}$, а в определении процессной сохраняющей историю бисимуляции $\mathcal{B} = \{\beta | \beta : C \to C', \pi = (C, f) \in \Pi(N), \pi' = (C', f') \in \Pi(N')\}$.

В следующих определениях $\pi = (C, f), \tilde{\pi} = (\tilde{C}, \tilde{f})$, $\pi' = (C', f'), \tilde{\pi}' = (\tilde{C}', \tilde{f}')$.

R — *сохраняющая историю бисимуляция на частичных словах* между N и N', запись $N \underline{\leftrightarrow}_{pwh} N'$, если:

1. $(\pi_N, \pi_{N'}, \emptyset) \in R$;

2. $(\pi, \pi', \beta) \in R \Rightarrow \beta : T_C \approx T_{C'}$;

3. $(\pi, \pi', \beta) \in R$, $\pi \to \tilde{\pi} \Rightarrow \exists \tilde{\beta}$, $\tilde{\pi}' : \pi' \to \tilde{\pi}'$, $\tilde{\beta}\lceil_{T_C} = \beta$, $\tilde{\beta}^{-1} : T_{\tilde{C}'} \sqsubseteq T_{\tilde{C}}$ и $(\tilde{\pi}, \tilde{\pi}', \tilde{\beta}) \in R$;

4. Как предыдущий пункт, но роли N и N' меняются.

R — *сохраняющая историю бисимуляция на* **ЧУММ** между N и N', запись $N \underline{\leftrightarrow}_{pomh} N'$, если:

1. $(\pi_N, \pi_{N'}, \emptyset) \in R$;

2. $(\pi, \pi', \beta) \in R \Rightarrow \beta : T_C \simeq T_{C'}$;

3. $(\pi, \pi', \beta) \in R$, $\pi \to \tilde{\pi} \Rightarrow \exists \tilde{\beta}$, $\tilde{\pi}' : \pi' \to \tilde{\pi}'$, $\tilde{\beta}\lceil_{T_C} = \beta$ и $(\tilde{\pi}, \tilde{\pi}', \tilde{\beta}) \in R$;

4. Как предыдущий пункт, но роли N и N' меняются.

R — *процессная сохраняющая историю бисимуляция* между N и N', запись $N \underline{\leftrightarrow}_{prh} N'$, если:

1. $(\pi_N, \pi_{N'}, \emptyset) \in R$;

2. $(\pi, \pi', \beta) \in R \Rightarrow \beta : C \simeq C'$;

3. $(\pi, \pi', \beta) \in R$, $\pi \to \tilde{\pi} \Rightarrow \exists \tilde{\beta}$, $\tilde{\pi}' : \pi' \to \tilde{\pi}'$, $\tilde{\beta}\lceil_{P_C \cup T_C} = \beta$ и $(\tilde{\pi}, \tilde{\pi}', \tilde{\beta}) \in R$;

4. Как предыдущий пункт, но роли N и N' меняются.

Очевидно, что можно упростить эти определения, рассматривая расширения процессов только на одно действие. Итерация таких расширений дает расширение на процесс.

2.4 Сравнение сетевых эквивалентностей

В этой главе доказывается теорема, устанавливающая взаимосвязи между всеми введенными ранее эквивалентностями.

Теорема 1 *Пусть* $\sim \in \{\equiv, \underline{\leftrightarrow}\}$
и $\alpha, \beta \in \{i,\ s,\ pw,\ pom,\ pr,\ iST,\ pwST,\ pomST,\ prST,\ pwh,\ pomh,\ prh\}$. *Для сетей* N *и* N' $N \sim_\alpha N' \Rightarrow N \sim_\beta N'$ *тогда и только тогда, когда в графе на рисунке 2.1 существует направленный путь* $\sim_\alpha \to \cdots \to \sim_\beta$.

Доказательство.

$\Leftarrow$ Проверим, что все импликации на рисунке 2.1 действительны.

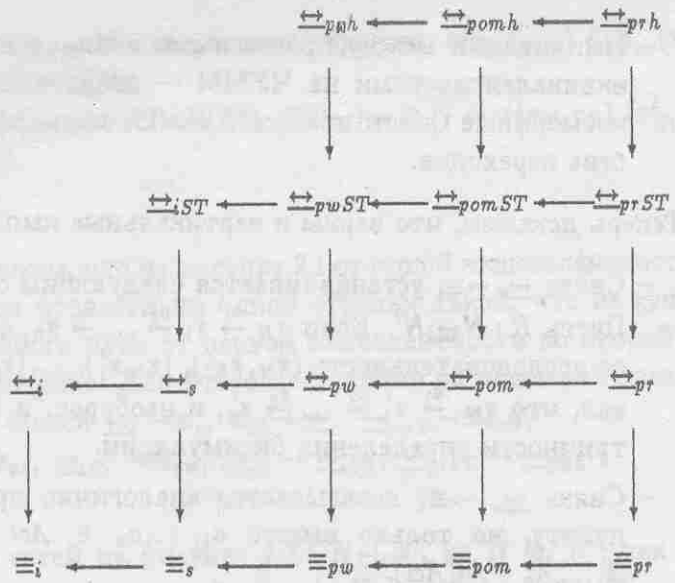


Рисунок 2.1: Взаимосвязь эквивалентностей

- Сначала проверим горизонтальные импликации.
  - Связь $\equiv_s \rightarrow \equiv_i$ следует из того, что последовательный след $a_1 \cdots a_n \in Act^*$ есть шаговый след $A_1 \cdots A_n \in (\mathcal{M}(Act))^*$ такой, что $A_1 = \{a_1\}, \ldots, A_n = \{a_n\}$.
  - Связь $\equiv_{pw} \rightarrow \equiv_s$ следует из того факта, что следу $A_1 \cdots A_n \in (\mathcal{M}(Act))^*$, где $\pi_N \stackrel{A_1}{\rightarrow} \pi_1 \stackrel{A_2}{\rightarrow} \ldots \stackrel{A_n}{\rightarrow} \pi_n$, соответствует ЧУММ ρ — класс изоморфизма T_{C_n}.
  - Связь $\leftrightarrows_s \rightarrow \leftrightarrows_i$ легко доказать, если в определении шаговой бисимуляции рассматривать расширения процессов на одноэлементные мультимножества.
  - Связь $\leftrightarrows_{pw} \rightarrow \leftrightarrows_s$ становится очевидной, если в определении бисимуляции на частичных словах использовать расширяющие процессы, переходы С-сетей которых не упорядочены по отношению предшествования.
  - Связь $\leftrightarrows_{pwST} \rightarrow \leftrightarrows_{iST}$ устанавливается, исходя из того, что гомоморфизм на переходах С-сетей процессов является сохраняющей пометку биекцией.
  - Импликации между эквивалентностями на ЧУММ и на частичных словах — следствия того, что эквивалентность сильнее сохраняющей пометку биекции на переходах С-сетей процессов.

– Импликации между процессными эквивалентностями и эквивалентностями на ЧУММ — следствия факта, что изоморфные С-сети процессов имеют изоморфные множества переходов.

• Теперь докажем, что верны и вертикальные импликации.

– Связь $\leftrightarrows_i \rightarrow \equiv_i$ устанавливается следующим образом. Пусть $R : N \underline{\leftrightarrows}_i N'$. Если $\pi_N \xrightarrow{a_1} \pi_1 \xrightarrow{a_2} \ldots \xrightarrow{a_n} \pi_n$, то существует последовательность $(\pi_N, \pi_{N'}), (\pi_1, \pi'_1), \ldots, (\pi_n, \pi'_n) \in R$ такая, что $\pi_{N'} \xrightarrow{a_1} \pi'_1 \xrightarrow{a_2} \ldots \xrightarrow{a_n} \pi'_n$, и наоборот, в силу симметричности определения бисимуляции.

– Связь $\leftrightarrows_s \rightarrow \equiv_s$ доказывается аналогично предыдущему пункту, но только вместо $a_1, \ldots, a_n \in Act$ используем $A_1, .., A_n \in \mathcal{M}(Act)$.

– Связь $\leftrightarrows_{pw} \rightarrow \equiv_{pw}$ доказывается так. Пусть $R : N \underline{\leftrightarrows}_{pw} N'$ и ρ — класс изоморфизма T_C для $\pi = (C, f) \in \Pi(N)$. Так как для любой сети N, ее начального процесса π_N и процесса π справедливо $\pi_N \xrightarrow{\pi} \pi$, существует пара $(\pi, \pi') \in R$ такая, что $\pi' = (C', f')$ и $T_{C'} \sqsubseteq T_C$. Если ρ' — класс изоморфизма $T_{C'}$, то $\rho' \sqsubseteq \rho$. Значит, $Pomsets(N') \sqsubseteq Pomsets(N)$. Тот факт, что $Pomsets(N) \sqsubseteq Pomsets(N')$, доказывается аналогично с использованием симметричности определения бисимуляции.

– Связь $\leftrightarrows_{pom} \rightarrow \equiv_{pom}$ доказывается так же, как в предыдущем пункте, только с использованием изоморфизма вместо гомоморфизма на переходах С-сетей процессов.

– Связь $\leftrightarrows_{pr} \rightarrow \equiv_{pr}$ устанавливается аналогично предыдущему пункту с использованием процессных следов вместо ЧУММ-следов и изоморфизма на С-сетях процессов вместо изоморфизма на их переходах.

– Импликации $\leftrightarrows_{\alpha ST} \rightarrow \leftrightarrows_\alpha$, $\alpha \in \{pw, pom, pr\}$, доказываются с помощью построения на основе отношения $R : N \underline{\leftrightarrows}_{\alpha ST} N'$ отношения $S : N \underline{\leftrightarrows}_\alpha N'$, которое определяется так: $\exists \beta \, ((\pi_E, \pi_E), (\pi'_E, \pi'_E), \beta) \in R \Leftrightarrow (\pi_E, \pi'_E) \in S$.

– Связь $\leftrightarrows_{iST} \rightarrow \leftrightarrows_s$ проверяется так же, как в предыдущем пункте, с учетом того, что шагу $\pi \xrightarrow{A} \tilde{\pi}$, где $A = \{a_1, \ldots, a_n\} \in \mathcal{M}(Act)$, соответствует последовательность ST-процессов $(\pi_E, \pi_P), \ldots, (\tilde{\pi}_E, \pi_P), \ldots, (\tilde{\pi}_E, \tilde{\pi}_P)$ такая, что $\pi_E \xrightarrow{a_1} \ldots \xrightarrow{a_n} \tilde{\pi}_E$, $\pi_P \xrightarrow{a_1} \ldots \xrightarrow{a_n} \tilde{\pi}_P$ и $\pi_E = \pi_P$, $\tilde{\pi}_E = \tilde{\pi}_P$.

– Импликации $\leftrightarrows_{\alpha h} \rightarrow \leftrightarrows_{\alpha ST}$, $\alpha \in \{pw, pom, pr\}$, доказываются построением на основе $R : N \underline{\leftrightarrows}_{\alpha h} N'$ отношения S :

$N \underline{\leftrightarrow}_{aST} N'$ следующим образом: $(\pi_E, \pi'_E, \beta) \in R$, $(\pi_E, \pi_P) \in ST - \Pi(N)$,
$(\pi'_E, \pi'_P) \in ST - \Pi(N')$, $\beta(T_{C_P}) = T_{C'_P} \Leftrightarrow ((\pi_E, \pi_P), (\pi'_E, \pi'_P), \beta) \in S$.

$\Rightarrow$ Докажем, что на рисунке 2.1 от одной эквивалентности к другой нельзя провести ни одной стрелки такой, что не существует направленного пути от первой экивалентности ко второй в графе на этом рисунке. Для этого достаточно на примерах доказать отсутствие связей $\underline{\leftrightarrow}_i \rightarrow \equiv_s$, $\equiv_{pr} \rightarrow \underline{\leftrightarrow}_i$, $\underline{\leftrightarrow}_{pwh} \rightarrow \equiv_{pom}$, $\underline{\leftrightarrow}_{pomh} \rightarrow \equiv_{pr}$, $\underline{\leftrightarrow}_{iST} \rightarrow \equiv_{pw}$, $\underline{\leftrightarrow}_{pr} \rightarrow \underline{\leftrightarrow}_{iST}$, $\underline{\leftrightarrow}_{prST} \rightarrow \underline{\leftrightarrow}_{pwh}$.

- Для сетей на рисунке 2.2.1 $N \underline{\leftrightarrow}_i N'$, но $N \not\equiv_s N'$, так как в N существует шаговый след $\{a, b\}$, которого нет в N'.

- Для сетей на рисунке 2.2.2 $N \equiv_{pr} N'$; но $N \not\underline{\leftrightarrow}_i N'$, так как только в N можно выполнить действие a так, что после него нельзя выполнить b.

- Для сетей на рисунке 2.2.3 $N \underline{\leftrightarrow}_{pwh} N'$, но $N \not\equiv_{pom} N'$, так как в N b может зависеть от a.

- Для сетей на рисунке 2.2.4 $N \underline{\leftrightarrow}_{pomh} N'$, но $N \not\equiv_{pr} N'$, так как N — С-сеть, не изоморфная С-сети N'.

- Для сетей на рисунке 2.2.5 $N \underline{\leftrightarrow}_{iST} N'$, но $N \not\equiv_{pw} N'$, так как сети N соответствует ЧУММ такое, что даже менее последовательное ЧУММ не может быть выполнено в N'.

- Для сетей на рисунке 2.2.6 $N \underline{\leftrightarrow}_{pr} N'$, но $N \not\underline{\leftrightarrow}_{iST} N'$, так как в N' действие a может так начать работать, что никакое b уже не может стартовать.

- Для сетей на рисунке 2.2.7 $N \underline{\leftrightarrow}_{prST} N'$, но $N \not\underline{\leftrightarrow}_{pwh} N'$, так как только N' может выполнить a и b так, чтобы следующее действие, c, обязательно зависело от a. $\square$

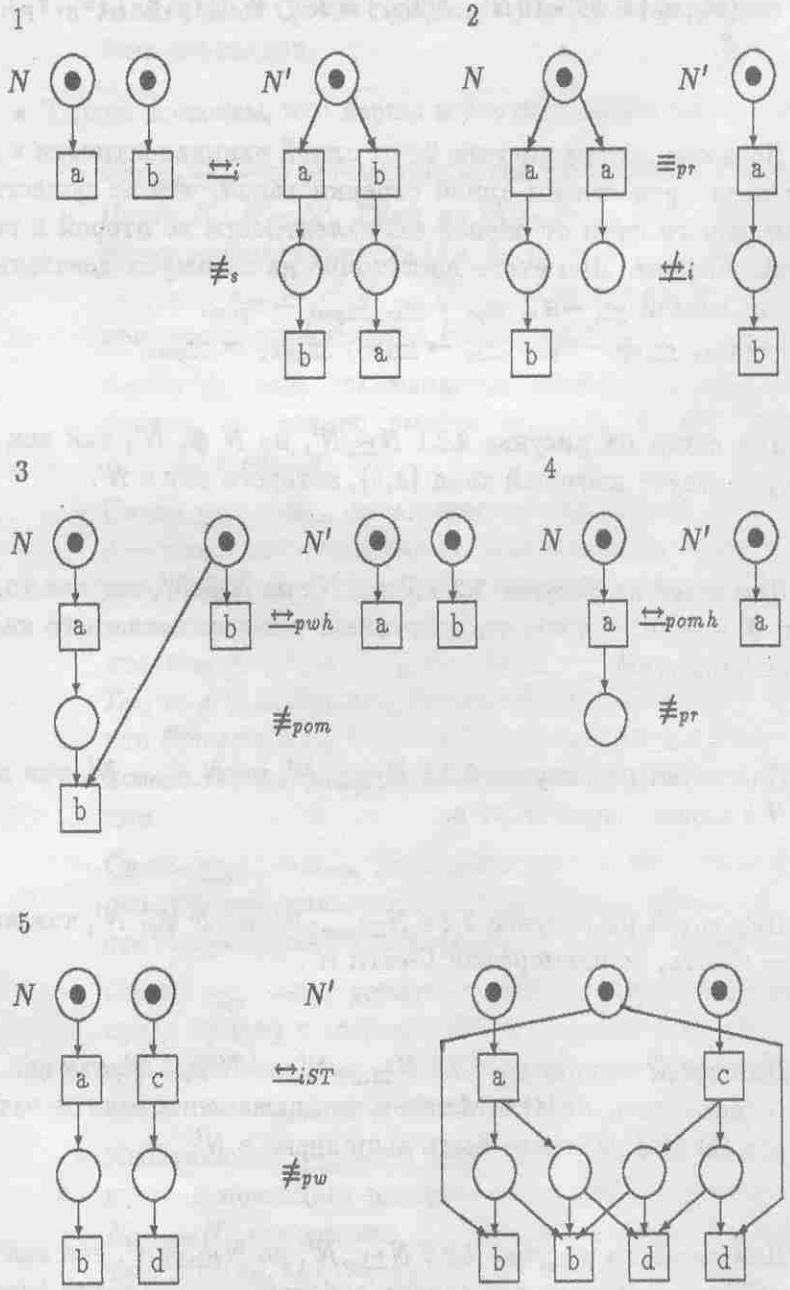
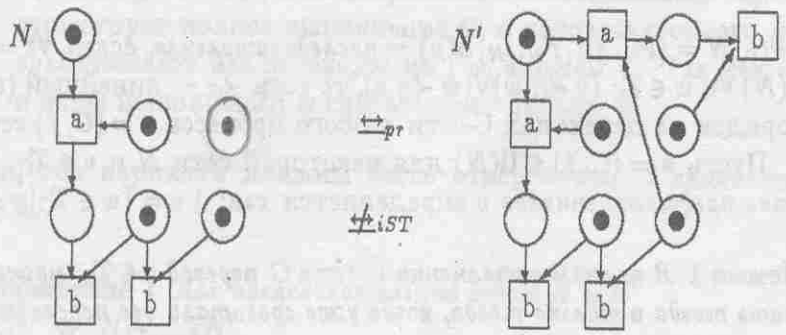


Рисунок 2.2: Примеры на сетях

6



7

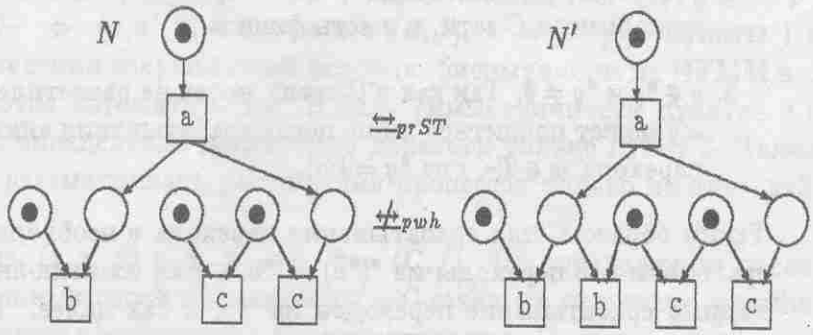


Рисунок 2.2: Примеры на сетях (продолжение)

2.5 Эквивалентности на различных классах сетей Петри

В литературе был введен ряд классов сетей Петри посредством введения некоторых ограничений на их определение или наоборот, с помощью расширения понятия сетей. При этом рассмотренные ранее эквивалентности могут слиться на сетях первого вида или стать менее зависимыми друг от друга на втором виде сетей. Эта глава посвящена исследованию того, как ведут себя эквивалентности на трех классах сетей первого вида. Мы рассмотрим последовательные сети, в которых невозможно параллельное выполнение переходов, а также C-сети и сети со строгой пометкой.

2.5.1 Эквивалентности на последовательных сетях

Сеть $N = (P_N, T_N, F_N, l_N, M_N)$ —*последовательная*, если: $\forall \pi = (C, f) \in \Pi(N)\, \forall v, w \in T_C\ (v \prec_C w) \vee (w \prec_C v)$, то есть $\prec_C$ — линейный (полный) порядок на переходах C-сети любого процесса $\pi = (C, f)$ сети N.

Пусть $\pi = (C, f) \in \Pi(N)$ для некоторой сети N и $v \in T_C$. *Множество предшественников* v определяется так: $\downarrow v = \{w \in T_C | w \prec_C v\}$.

Лемма 1 *В полном выполнении C-сети C переход $v \in T_C$ может сработать тогда и только тогда, когда уже сработали все переходы из $\downarrow v$.*

Доказательство.

$\Rightarrow$ Если переход $v \in T_C$ может сработать, в местах из ${}^\bullet v$ есть фишки. Эти места можно разделить на две группы.

1. $q \in {}^\bullet v$ и ${}^\bullet q = \emptyset$. Тогда $q \in {}^\circ C$ и, по определению полного выполнения C-сети, в q есть фишка.

2. $q \in {}^\bullet v$ и ${}^\bullet q \neq \emptyset$. Так как в C-сетях места не разветвлены, фишка в q может появиться лишь после срабатывания единственного перехода $w \in T_C$, где ${}^\bullet q = \{w\}$.

Таким образом, для срабатывания перехода v необходимо, чтобы сработали все переходы из ${}^\bullet({}^\bullet v) = {}^{2\bullet} v$, а для их выполнения необходимо срабатывание переходов из ${}^{4\bullet} v$, и так далее. Так как, по определению C-сетей, в F_C нет бесконечных в обратную сторону цепочек, существует $n \in \mathcal{N}$ такое, что ${}^{2n\bullet} v = \emptyset$. Следовательно, для срабатывания v необходимо, чтобы выполнились все переходы из $\bigcup_{i=1}^{n-1} {}^{2i\bullet} v = \downarrow v$.

$\Leftarrow$ Если в полном выполнении C уже сработали все переходы из $\downarrow v$ для $v \in T_C$, то выполнились и переходы из ${}^{2\bullet} v$. Поэтому все места из ${}^\bullet v$ имеют фишки, и v может сработать. □

Лемма 2 *Пусть C — C-сеть и $v, w \in T_C$. В этом случае $v \prec_C w$ тогда и только тогда, когда в любом полном выполнении C v срабатывает раньше w.*

Доказательство.

$\Rightarrow$ По лемме 1 переход $w \in T_C$ может сработать только тогда, когда уже сработали все переходы из $\downarrow w$, а $v \in \downarrow w$.

$\Leftarrow$ Докажем от противного. Пусть $\neg(v \prec_C w)$. Возможны два варианта.

1. $w \prec_C v$. Тогда, по доказанному ранее, в любом полном выполнении C w срабатывает перед v, что противоречит исходному предположению.

2. $\neg(w \prec_C v)$, то есть v и w не сравнимы по отношению $\prec_C$. Тогда существует полное выполнение C, в котором с самого начала срабатывают все переходы из $\downarrow w$, а затем w. Так как $v \notin \downarrow w$, в этом выполнении w срабатывает раньше v.

Итак, оба варианта должны быть отвергнуты. Следовательно, $v \prec_C w$. □

Утверждение 1 *Для последовательных сетей N и N'* $N \underline{\leftrightarrow}_i N' \Leftrightarrow N \underline{\leftrightarrow}_{pomh} N'$.

Доказательство.

$\Leftarrow$ Следует из теоремы 1.

$\Rightarrow$ Пусть $R : N \underline{\leftrightarrow}_i N'$ для последовательных сетей N и N'. Докажем, что $S : N \underline{\leftrightarrow}_{pomh} N'$, где S определяется так: $(\pi, \pi') \in R$, $\beta : T_C \simeq T_{C'} \Leftrightarrow (\pi, \pi', \beta) \in S$, где $\pi = (C, f)$, $\pi' = (C', f')$. Пункты 1 и 2 определения сохраняющей историю бисимуляции на ЧУММ выполняются автоматически. В силу симметричности пунктов 3 и 4 этого определения достаточно доказать только пункт 3. Также можно рассматривать расширения процессов только на одно действие.

Пусть $(\pi, \pi', \beta) \in S$, $\pi \xrightarrow{v} \tilde{\pi}$, $\tilde{\pi} = (\tilde{C}, \tilde{f})$. По определению последовательных сетей все переходы в С-сетях их процессов линейно упорядочены отношением предшествования. Поэтому, по лемме 2 полные выполнения таких сетей единственны. Рассмотрим полное выполнение $\tilde{C}$ в N $\pi_N = \pi_0 \xrightarrow{v_1} \ldots \xrightarrow{v_n} \pi_n = \pi \xrightarrow{v} \tilde{\pi}$. По определению последовательной бисимуляции существуют $\tilde{\pi}'$, v' такие, что $\pi_{N'} = \pi'_0 \xrightarrow{\beta(v_1)} \ldots \xrightarrow{\beta(v_n)} \pi'_n = \pi' \xrightarrow{v'} \tilde{\pi}'$, $\tilde{\pi}' = (\tilde{C}', \tilde{f}')$, $l_{\tilde{C}}(v) = l_{\tilde{C}'}(v')$, $(\tilde{\pi}, \tilde{\pi}') \in R$. Это — единственное полное выполнение $\tilde{C}'$ в N'.

Рассмотрим отображение $\tilde{\beta}$, где $\tilde{\beta}\lceil_{T_C} = \beta$ и $\tilde{\beta}(v) = v'$. Очевидно, что $\tilde{\beta} : T_{\tilde{C}} \approx T_{\tilde{C}'}$. Докажем, что $\tilde{\beta} : T_{\tilde{C}} \simeq T_{\tilde{C}'}$. Для этого достаточно показать следующее: $\forall w \in T_C$ $w \prec_{\tilde{C}} v \Leftrightarrow \tilde{\beta}(w) \prec_{\tilde{C}'} \tilde{\beta}(v)$. Пусть $w \prec_{\tilde{C}} v$. Тогда в любом полном выполнении $\tilde{C}$, а, следовательно, и в единственном полном выполнении этой сети, w срабатывает раньше v. По определению $\tilde{\beta}$, в единственном полном выполнении $\tilde{C}'$ $\tilde{\beta}(w)$ срабатывает также раньше, чем $\tilde{\beta}(v)$. Таким образом, можно утверждать, что в любом полном выполнении $\tilde{C}'$ $\tilde{\beta}(w)$ срабатывает перед $\tilde{\beta}(v)$. Тогда по лемме 2 $\tilde{\beta}(w) \prec_{\tilde{C}'} \tilde{\beta}(v)$. В обратную сторону сохранение отношения причинной зависимости доказывается аналогично. □

Утверждение 2 *Для последовательных сетей N и N'* $N \equiv_i N' \Leftrightarrow N \equiv_{pom} N'$.

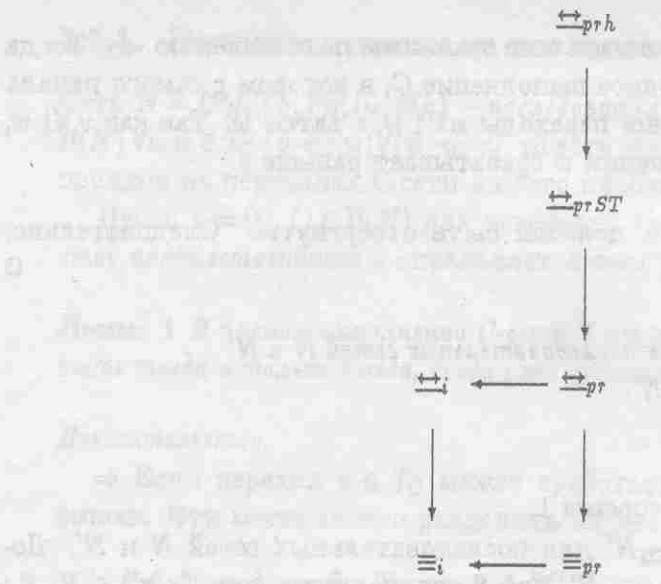


Рисунок 2.3: Эквивалентности на последовательных сетях

Доказательство.

$\Leftarrow$ Следствие теоремы 1.

$\Rightarrow$ Пусть $N \equiv_i N'$. Докажем, что $N \equiv_{pom} N'$, то есть что $Pomsets(N) = Pomsets(N')$. Пусть $\pi = (C, f) \in \Pi(N)$. Так как N — последовательная сеть, то существует единственное полное выполнение C в N $\pi_N = \pi_0 \xrightarrow{v_1} \ldots \xrightarrow{v_n} \pi_n = \pi$. Так как $N \equiv_i N'$, то существуют π'_i, v'_i $(1 \leq i \leq n)$ такие, что $\pi_{N'} = \pi'_0 \xrightarrow{v'_1} \ldots \xrightarrow{v'_n} \pi'_n = \pi'$ $\pi' = (C', f')$, $l_C(v_i) = l_{C'}(v'_i)$. Это — единственное полное выполнение C' в N'. Определим отображение β так: $\beta(v_i) = v'_i$. Очевидно, что $\beta : T_C \approx T_{C'}$. По лемме 2 в силу единственности полных выполнений C и C' последовательность, в которой срабатывают переходы этих сетей в их полных выполнениях, задает и полный порядок на этих переходах. Поэтому $\forall v, w \in T_C$ $v \prec_C w \Leftrightarrow \beta(v) \prec_{C'} \beta(w)$, то есть $\beta : T_C \simeq T_{C'}$. Следовательно, $Pomsets(N) \subseteq Pomsets(N')$. Тот факт, что $Pomsets(N') \subseteq Pomsets(N)$, доказывается аналогично. □

Теорема 2 *Пусть* $\sim \in \{\equiv, \leftrightarrows\}$, $\alpha, \beta \in \{i, pr, prST, prh\}$. *Для последовательных сетей* N *и* N' $N \sim_\alpha N' \Rightarrow N \sim_\beta N'$ *тогда и только тогда, когда в графе на рисунке 2.3 существует направленный путь* $\sim_\alpha \rightarrow \cdots \rightarrow \sim_\beta$.

Доказательство.

$\Leftarrow$ Следствие теоремы 1.

$\Rightarrow$ Проверим, что на рисунке 2.3 от одной эквивалентности к другой нельзя провести ни одной стрелки такой, что не существу-

ет направленнного пути от первой эквивалентности ко второй. Для этого достаточно на примерах доказать отсутствие связей $\leftrightarrows_i \not\to \equiv_{pr}$, $\equiv_{pr} \not\to \leftrightarrows_i$, $\leftrightarrows_{pr} \not\to \leftrightarrows_{prST}$, $\leftrightarrows_{prST} \not\to \leftrightarrows_{prh}$.

- Для сетей на рисунке 2.2.4 $N \leftrightarrows_i N'$, но $N \not\equiv_{pr} N'$, так как N — С-сеть, не изоморфная N'.
- Для сетей на рисунке 2.2.2 $N \equiv_{pr} N'$, но $N \not\leftrightarrows_i N'$, так как только в N можно выполнить действие a так, что после него нельзя выполнить b.
- Для сетей на рисунке 2.4.1 $N \leftrightarrows_{pr} N'$, но $N \not\leftrightarrows_{prST} N'$, так как только в N' можно начать выполнять процесс с действием a так, чтобы его можно было расширить на действие b только одним способом (то есть так, чтобы расширенный процесс был только один).
- Для сетей на рисунке 2.4.2 $N \leftrightarrows_{prST} N'$, но $N \not\leftrightarrows_{prh} N'$, так как только в N' может выполниться процесс, в котором срабатывают последовательно действия a и b, так, чтобы следующее действие, c, могло расширить этот процесс только одним способом (то есть С-сеть с действием c, расширяя С-сеть, соответствующую последовательности ab, сцепляется с ее подсетью, содержащей a, только одним способом).

□

2.5.2 Эквивалентности на С-сетях

С-сеть — это маркированная помеченная С-сеть. Пусть N – некоторая сеть. Действие $a \in Act$ *автопараллельно* в N, если $\exists \pi = (C, f) \in \Pi(N)\ \exists v, w \in T_C\ \neg(v \prec_C w) \& \neg(w \prec_C v)$ и $l_C(v) = l_C(w) = a$. То есть в некотором процессе сети N два действия a не зависят друг от друга. Сеть N *без автопараллелизма*, если ни одно из действий не автопараллельно в N.

Рассмотрим, как ведут себя ранее введенные эквивалентности на С-сетях без автопараллелизма.

Утверждение 3 *Для С-сетей без автопараллелизма N и N' $N \equiv_i N' \Leftrightarrow N \leftrightarrows_i N'$.*

Доказательство.

$\Leftarrow$ Следствие теоремы 1.

$\Rightarrow$ Пусть $N \equiv_i N'$, где N и N' — сети без автопараллелизма. Определим отношение R следующим образом: $\pi_N = \pi_0 \xrightarrow{a_1} \dots \xrightarrow{a_n} \pi_n = \pi$, $\pi_{N'} = \pi'_0 \xrightarrow{a_1} \dots \xrightarrow{a_n} \pi'_n = \pi'$, $a_i \in Act\ (1 \le i \le n) \Leftrightarrow (\pi, \pi') \in R$.

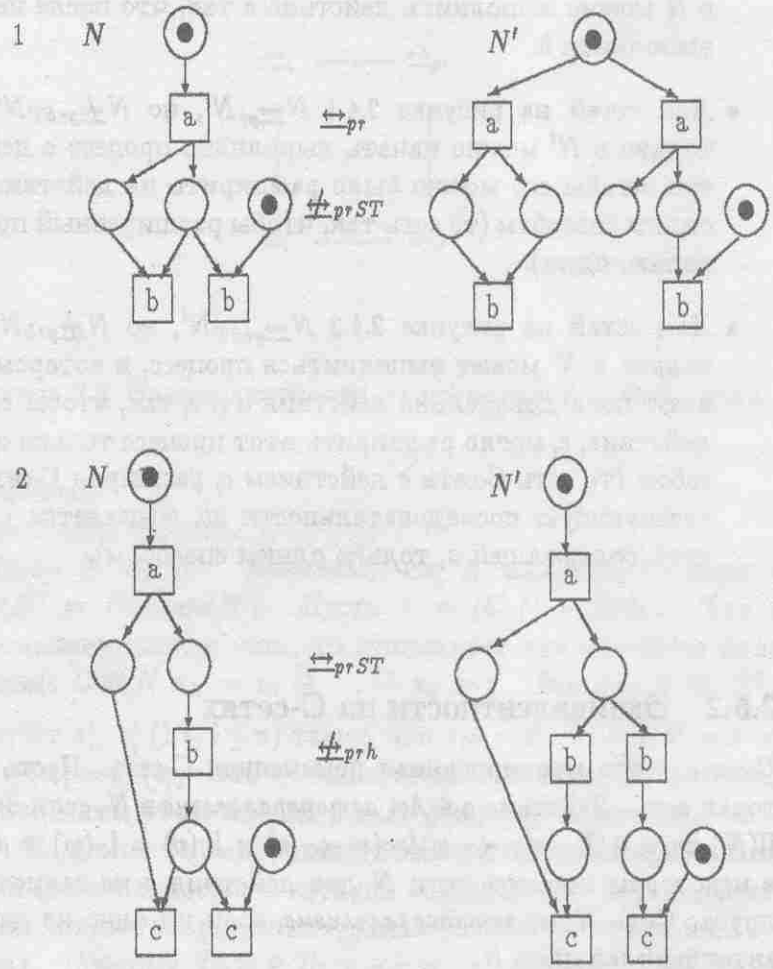


Рисунок 2.4: Примеры на последовательных сетях

Докажем, что $R : N \underline{\leftrightarrow}_i N'$. Пункт 1 определения последовательной бисимуляции выполняется автоматически. В силу симметричности пунктов 2 и 3 этого определения достаточно доказать только пункт 2. Проведем доказательство от противного.

Пусть $(\pi, \pi') \in R$, $\pi \xrightarrow{a} \tilde{\pi}$, $a \in Act$, но π' невозможно расширить на действие a. По определению отношения R имеем последовательности расширений процессов $\pi_N = \pi_0 \xrightarrow{a_1} \cdots \xrightarrow{a_n} \pi_n = \pi$ и $\pi_{N'} = \pi'_0 \xrightarrow{a_1} \cdots \xrightarrow{a_n} \pi'_n = \pi'$. Так как $N \equiv_i N'$, то существуют $\bar{\pi}'_i \in \Pi(N')$ $(0 \le i \le n)$ такие, что $\pi_{N'} = \bar{\pi}'_0 \xrightarrow{a_1} \cdots \xrightarrow{a_n} \bar{\pi}'_n = \bar{\pi}'$ и $\bar{\pi}'$ можно расширить на действие a. Пусть $M'_i = f'_i(C'^{\circ}_i)$, $\bar{M}'_i = \bar{f}'_i(\bar{C}'^{\circ}_i)$, где $\pi'_i = (C'_i, f'_i)$, $\bar{\pi}'_i = (\bar{C}'_i, \bar{f}'_i)$ $(0 \le i \le n)$ и $l_{N'}(u'_i) = l_{N'}(\bar{u}'_i) = a_i$ $(1 \le i \le n)$. Тогда $M_{N'} = M'_0 \xrightarrow{u'_1} \cdots \xrightarrow{u'_n} M'_n = M'$ и $M_{N'} = \bar{M}'_0 \xrightarrow{\bar{u}'_1} \cdots \xrightarrow{\bar{u}'_n} \bar{M}'_n = \bar{M}'$ — последовательности срабатываний переходов в N'. Так как в M', в отличие от $\bar{M}'$, не допустим ни один a-помеченный переход, то $M' \neq \bar{M}'$. Выберем наименьшее i $(1 \le i \le n)$ такое, что $M'_{i-1} = \bar{M}'_{i-1}$, но $M'_i \neq \bar{M}'_i$. Имеем $M'_{i-1} \xrightarrow{u'_i} M'_i$ и $M'_{i-1} \xrightarrow{\bar{u}'_i} \bar{M}'_i$. Очевидно, что $u'_i \neq \bar{u}'_i$ и ${}^\bullet u'_i \subseteq M'_{i-1}$, ${}^\bullet \bar{u}'_i \subseteq M'_{i-1}$. Имеем два случая.

1. ${}^\bullet u'_i +^\bullet \bar{u}'_i \subseteq M'_{i-1}$. Так как $l_{N'}(u'_i) = l_{N'}(\bar{u}'_i) = a_i$, то действие a_i автопараллельно в N', что противоречит предположению об отсутствии автопараллелизма.

2. ${}^\bullet u'_i +^\bullet \bar{u}'_i \not\subseteq M'_{i-1}$. Тогда ${}^\bullet u'_i \cap {}^\bullet \bar{u}'_i \neq \emptyset$, что противоречит условию неразветвленности мест в определении С-сети.

Таким образом, оба варианта должны быть отвергнуты и $R : N \underline{\leftrightarrow}_i N'$. □

Заметим, что условие отсутствия автопараллелизма существенно в только что доказанном утверждении. На рисунке 2.5.1 в сети N действие a автопараллельно, $N \equiv_i N'$, но $N \not\underline{\leftrightarrow}_i N'$, так как в N действие a может сработать так, что действие b еще нельзя будет выполнить.

Утверждение 4 *Для С-сетей без автопараллелизма N и N' $N \underline{\leftrightarrow}_i N' \Leftrightarrow N \underline{\leftrightarrow}_s N'$.*

Доказательство.

$\Leftarrow$ Следствие теоремы 1.

$\Rightarrow$ Пусть $N \underline{\leftrightarrow}_i N'$, где N и N' — С-сети без автопараллелизма. Докажем, что $N \underline{\leftrightarrow}_s N'$. Пункт 1 определения шаговой бисимуляции выполняется автоматически. В силу симметричности пунктов 2 и 3 этого определения достсточно доказать только пункт 2. Проведем доказательство от противного.

Пусть $(\pi, \pi') \in R$, $\pi \xrightarrow{A} \tilde{\pi}$, $A \in \mathcal{M}(Act)$ и π' невозможно расширить на A. Тогда существуют a, $b \in A$ такие, что π' невозможно

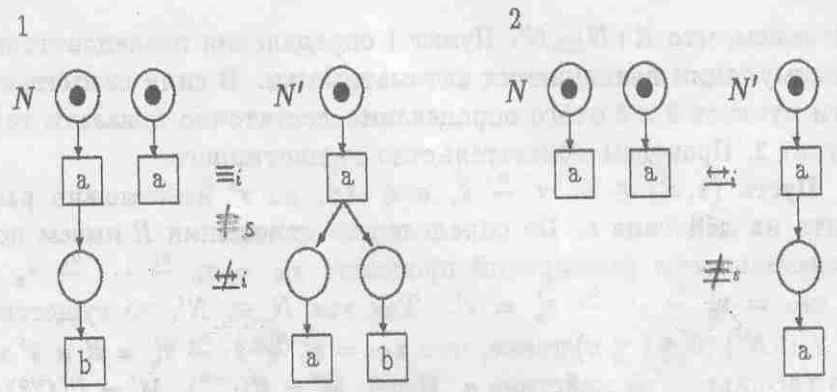


Рисунок 2.5: Примеры на C-сетях

расширить на на множество $\{a, b\}$. Так как N — сеть без автопараллелизма, то $a \neq b$. По определению последовательной бисимуляции π' можно расширить на действия a и b в отдельности. Пусть $M' = f'(C'^{\circ})$, где $\pi' = (C', f')$. Тогда $\exists u'$, $\bar{u}' \in T_{N'}$ $l_{N'}(u') = a$, $l_{N'}(\bar{u}') = b$, ${}^\bullet u' \subseteq M'$, ${}^\bullet \bar{u}' \subseteq M'$, но ${}^\bullet u' +^\bullet \bar{u}' \not\subseteq M'$. Следовательно, ${}^\bullet u' \cap^\bullet \bar{u}' \neq \emptyset$, что противоречит условию неразветвленности мест в определении C-сети. □

Условие отсутствия автопараллелизма важно и в этом утверждении. На рисунке 2.5.2 в сети N действие a автопараллельно, $N \leftrightarrows_i N'$, но $N \not\equiv_s N'$, так как только в N действие a может сработать параллельно с самим собой.

Таким образом, на C-сетях без автопараллелизма последовательная эквивалентность совпадает с шаговой бисимуляционной эквивалентностью.

2.5.3 Эквивалентности на сетях со строгой пометкой

Сеть $N = \langle P_N, T_N, F_N, l_N, M_N \rangle$ *со строгой пометкой*, если ее помечающая функция l_N биективна, то есть $\forall u, \bar{u} \in T_N$ $u \neq \bar{u} \Rightarrow l_N(u) \neq l_N(\bar{u})$.

Рассмотрим ранее введенные эквивалентности на сетях со строгой пометкой.

Утверждение 5 *Для строго помеченных сетей N и N' $N \equiv_i N' \Leftrightarrow N \leftrightarrows_i N'$.*

Доказательство.

$\Leftarrow$ Следствие теоремы 1.

$\Rightarrow$ Пусть $N \equiv_i N'$, где N и N' — сети со строгой пометкой. Определим отношение R следующим образом: $\pi_N = \pi_0 \xrightarrow{a_1} \cdots \xrightarrow{a_n} \pi_n =$

π, $\pi_{N'} = \pi'_0 \xrightarrow{a_1} \dots \xrightarrow{a_n} \pi'_n = \pi'$, $a_i \in Act$ $(1 \le i \le n)$ $\Leftrightarrow$ $(\pi, \pi') \in R$. Докажем, что $R : N \underline{\leftrightarrow}_i N'$. Пункт 1 определения последовательной бисимуляции выполняется автоматически. В силу симметричности пунктов 2 и 3 этого определения достаточно доказать только пункт 2. Проведем доказательство от противного.

Пусть $(\pi, \pi') \in R$, $\pi \xrightarrow{a} \tilde{\pi}$, $a \in Act$, но π' невозможно расширить на действие a. По определению отношения R имеем последовательности расширений процессов $\pi_N = \pi_0 \xrightarrow{a_1} \dots \xrightarrow{a_n} \pi_n = \pi$ и $\pi_{N'} = \pi'_0 \xrightarrow{a_1} \dots \xrightarrow{a_n} \pi'_n = \pi'$. Так как $N \equiv_i N'$, то существуют $\bar{\pi}'_i \in \Pi(N')$ $(0 \le i \le n)$ такие, что $\pi_{N'} = \bar{\pi}'_0 \xrightarrow{a_1} \dots \xrightarrow{a_n} \bar{\pi}'_n = \bar{\pi}'$ и $\bar{\pi}'$ можно расширить на действие a. Пусть $M'_i = f'_i(C'^{\circ}_i)$, $\bar{M}'_i = \bar{f}'_i(\bar{C}'^{\circ}_i)$, где $\pi'_i = (C'_i, f'_i)$, $\bar{\pi}'_i = (\bar{C}'_i, \bar{f}'_i)$ $(0 \le i \le n)$ и $l_{N'}(u'_i) = l_{N'}(\bar{u}'_i) = a_i$ $(1 \le i \le n)$. Тогда $M_{N'} = M'_0 \xrightarrow{u'_1} \dots \xrightarrow{u'_n} M'_n = M'$ и $M_{N'} = \bar{M}'_0 \xrightarrow{\bar{u}'_1} \dots \xrightarrow{\bar{u}'_n} \bar{M}'_n = \bar{M}'$ — последовательности срабатываний переходов в N'. Так как в M', в отличие от $\bar{M}'$, не допустим ни один a-помеченный переход, то $M' \ne \bar{M}'$. Так как N' — строго помеченная сеть, то $u'_i = \bar{u}'_i$ $(1 \le i \le n)$. Тогда после срабатывания одной и той же последовательности переходов $u'_1, \dots, u'_n \in T_{N'}$ из начальной маркировки мы получаем разные маркировки M' и $\bar{M}'$. Приходим к противоречию. Следовательно, $R : N \underline{\leftrightarrow}_i N'$. □

Для строго помеченных сетей в графе на рисунке 2.1 нельзя провести ни одной стрелки от последовательным к шаговым эквивалентностям, от эквивалентностей на частичных словах к эквивалентностям на ЧУММ, а также от эквивалентностей на ЧУММ к процессным. Это доказывается следующими примерами на строго помеченных сетях.

1. На рисунке 2.6.1 $N \underline{\leftrightarrow}_i N'$, но $N \not\equiv_s N'$, так как только в сети N действия a и b могут выполниться параллельно.

2. На рисунке 2.6.2 $N \underline{\leftrightarrow}_{pwh} N'$, но $N \not\equiv_{pom} N'$, так как только в N' действие b может зависеть от a.

3. На рисунке 2.2.4 $N \underline{\leftrightarrow}_{pomh} N'$, но $N \not\equiv_{pr} N'$, так как N — С-сеть, не изоморфная С-сети N'.

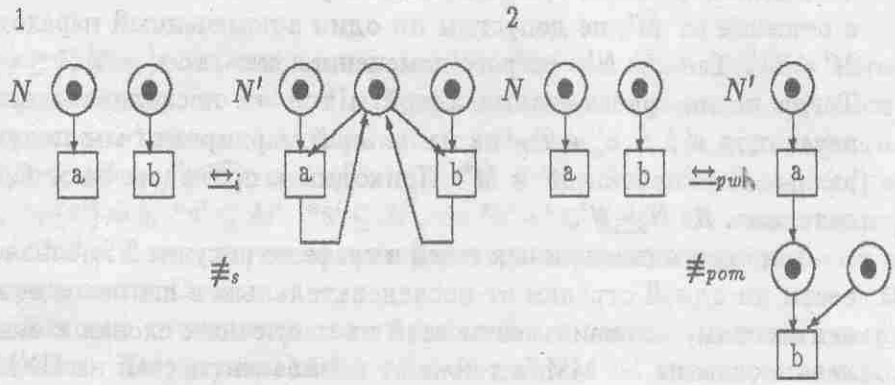


Рисунок 2.6: Примеры на сетях со строгой пометкой

Глава 3

Эквивалентности и формульные представления процессов

Для описания параллельных систем и процессов и исследования их поведенческих свойств был предложен ряд моделей, различающихся представлением параллельности. Среди таких моделей алгебраические исчисления занимают особое место. В этих исчислениях процесс описывается алгебраической формулой, и проверка свойств процесса выполняется посредством эквивалентностей, аксиом и правил вывода. Все исчисления, рассматриваемые ниже, имеют общее ядро. Они конструируют процесс из атомарных действий (некоторые из которых могут взаимодействовать) с использованием операций последовательного выполнения, параллелизма и недетерминизма. В зависимости от семантики операции параллелизма эти алгебраические исчисления могут быть разделены на три группы.

1. CCS [20], TCSP [6], PA [3] — наиболее известные алгебраические исчисления для описания параллельного процесса на основе последовательной (интерливинговой) семантики, в которой параллельность моделируется последовательным недетерминизмом.

2. SCCS [21], ACP [7] были предложены для аксиоматизации группы параллельных моделей, основанных на шаговой семантике, в которой параллельное выполнение двух процессов моделируется чередованием мультимножеств их атомарных действий. Множество операций было расширено (по сравнению

с CCS и PA) введением нового оператора, описывающего одновременное выполнение двух действий.

3. Алгебраические исчисления для группы моделей, описывающих *истинный параллелизм* и основанных на семантике ЧУММ. Среди таких моделей наиболее известны сети Петри, структуры событий, ЧУММ, О-сети (Occurrence Nets) [24] и А-сети [17]. Отношение причинной зависимости действий в модели задает на них отношение предшествования, порождающее частичный порядок. Соответствеено, два действия параллельны, если они причинно независимы.

Алгебры, предлагаемые для спецификации процесса, могут быть разделены на *"описательные"* и *"аналитические"*. В описателных алгебрах спецификация процесса обеспечивает описание структурных свойств разрабатываемых параллельных систем. Аналитические алгебры содержат механизмы для проверки поведенческих свойств процессов. Алгебры $AFP_1, AFP_2, SAFP_2$, рассмотренные в [8, 9, 10, 11, 12], относятся к третьей группе алгебраических исчислений и объединяют механизмы как для описания параллельных недетерминированных процессов, так и для исследования их свойств.

В этой главе мы рассмотрим одно из таких алгебраических исчислений, алгебру AFP_2.

3.1 Алгебра AFP_2

3.1.1 Синтаксис

Пусть $\alpha = \{a, b, c, \ldots\}$ — конечный алфавит символов *действий* (базис действий процесса). Действия комбинируются в процесс с помощью операций ; (*предшествование*), ∇ (*исключение* или *альтернатива*) и $\|$ (*параллельность*). В процессе $(a; b)$ сначала выполняется действие a и только после этого — b. Процесс $(a \nabla b)$ описывает два возможных поведения: если выбирается выполнение действия a, то b не случается и наоборот. Формула $(a \| b)$ представляет процесс, в котором действия a и b выполняются параллельно.

Предполагается, что каждое действие имеет свое уникальное имя. Например, формула $(a; c) \| (b; c)$ описывает процесс, в котором действия a и b выполняются параллельно, и только затем выполняется c. Таким образом, выполнения действия c в подпроцессах $(a; c)$ и $(b; c)$ синхронизированы.

Алгебра AFP_2 содержит механизмы для описания как самих процессов, так и их свойств. Для выражения и проверки различ-

ных свойств процессов к базису действий процесса добавляются другие множества. $\bar{\alpha} = \{\bar{a}, \bar{b}, \bar{c}, \ldots\}$ — двойной к α алфавит символов *не-действий*, которые выражают факт, что соответствующие действия не выполняются во время работы процесса из-за выполнения некоторых альтернативных им действий. $\Delta_\alpha = \{\delta_a, \delta_b, \delta_c, \ldots\}$ — алфавит *тупиковых* действий, описывающий действия, которые не могут выполниться из-за некоторого противоречия или ошибки в описании процесса.

Вводятся также дополнительные операции: $\vee$ (*дизъюнкция* или *объединение*), $\overline{\|}$ (*"не случится"*), $\tilde{\overline{\|}}$ (*"не случится ошибочно"*).

Формула $(P \vee Q)$ определяет процесс, в котором выполняется либо подпроцесс P, либо Q. В этом случае множество возможных поведений процесса — объединение множеств поведений подпроцессов P и Q.

Операция $\overline{\|}$ — модифицированное отрицание. $\overline{\|}P$ означает, что процесс P не случится, то есть выполняются не-действия из P.

Операция $\tilde{\overline{\|}}$ — другой тип отрицания. $\tilde{\overline{\|}}P$ означает, что процесс P не случается в результате некоторой ошибки, то есть любое действие из P не случается во время функционирования процесса в результате некоторых противоречивых требований в описании процесса.

Формула AFP_2 в базисе $\alpha \cup \bar{\alpha} \cup \delta_\alpha$ определяется так:

1. a, $\bar{a}$, δ_a, где $a \in \alpha$, $\bar{a} \in \bar{\alpha}$, $\delta_a \in \Delta_\alpha$ — *элементарные формулы*;
2. Если P и Q — формулы, тогда $P \| Q$, $P \bigtriangledown Q$, $P; Q$, $P \vee Q$, $\overline{\|}P$, $\tilde{\overline{\|}}P$ — формулы.

3.1.2 Денотационная семантика

Семантика процесса, который описывается формулой AFP_2, есть множество частичных порядков, то есть совокупность частично упорядоченных множеств (ЧУМ), рассмотренных в [8, 9] с порядком по предшествованию действий при выполнении данного процесса.

Частично упорядоченное множество — пара $\rho = (X, \prec)$, где

- X — множество вершин, моделирующих действия, не-действия и тупиковые действия процесса, то есть $X \subseteq \alpha \cup \bar{\alpha} \cup \Delta_\alpha$;
- $\prec$ — частичный порядок над X, где $a \prec b$ интерпретируется так: действие a обязательно предшествует b в процессе.

Обозначим через $X^+ = \{x \in X | x \in \alpha\}$ *подмножество действий* X, $X^- = \{x \in X | x \in \bar{\alpha}\}$ *подмножество не-действий* X, $\Delta_X = \{x \in$

$X|x \in \Delta_\alpha\}$ *подмножество тупиковых действий* X, то есть $X = X^+ \cup X^- \cup \Delta_X$. Мы будем рассматривать ЧУМ только двух типов: либо $X = X^+ \cup X^-$, либо $X = X^+ \cup \Delta_X$, то есть если в ЧУМ существует некоторое тупиковое действие, мы не будем различать не-действия и тупиковые действия и будем рассматривать все не-действия в "отрицательной" части процесса как тупиковые.

Над частично упорядоченными множествами вводятся операции $\|$, ∇, $;$, $\overline{\|}$, $\tilde{\overline{\|}}$ и операция *модифицированного объединения* $\tilde{\cup}$, определяемые в [8, 9].

Пусть $\mathcal{D}_2[P]$ — совокупность частично упорядоченных множеств, связанных с процессом P. *Денотационная семантика* AFP_2 определяется так:

1. $\mathcal{D}_2[a] = (\{a\}, \emptyset)$, $\mathcal{D}_2[\bar{a}] = (\{\bar{a}\}, \emptyset)$, $\mathcal{D}_2[\delta_a] = (\{\delta_a\}, \emptyset)$;
2. $\mathcal{D}_2[P\|Q] = \mathcal{D}_2[P]\|\mathcal{D}_2[Q]$;
3. $\mathcal{D}_2[P;Q] = \mathcal{D}_2[P];\mathcal{D}_2[Q]$;
4. $\mathcal{D}_2[P \nabla Q] = \mathcal{D}_2[P] \nabla \mathcal{D}_2[Q]$;
5. $\mathcal{D}_2[P \vee Q] = \mathcal{D}_2[P]\tilde{\cup}\mathcal{D}_2[Q]$;
6. $\mathcal{D}_2[\overline{\|} P] = \overline{\|}\mathcal{D}_2[P]$;
7. $\mathcal{D}_2[\tilde{\overline{\|}} P] = \tilde{\overline{\|}}\mathcal{D}_2[P]$.

Рассмотрим процесс, описываемый формулой $P = (a \nabla b)\|(b \nabla c)$. Тогда $\mathcal{D}_2[P] = \{(\{b, \bar{a}, \bar{c}\}, \emptyset), (\{a, c, \bar{b}\}, \emptyset)\}$ состоит из двух частично упорядоченных множеств.

3.1.3 Аксиоматизация

Рассмотренная денотационная семантика для процессов задает эквивалентность. Два процесса P и Q *эквивалентны*, обозначение $P \approx_e Q$, если $\mathcal{D}_2[P] = \mathcal{D}_2[Q]$. В соответствии с этим отношением эквивалентности вводится система аксиом Θ_2.

В следующих равенствах P, Q, R обозначают формулы AFP_2, а $a \in \alpha$, $\bar{a} \in \bar{\alpha}$, $\delta_a \in \Delta_\alpha$.

1. Ассоциативность

   1.1 $P\|(Q\|R) = (P\|Q)\|R$

   1.2 $P \nabla (Q \nabla R) = (P \nabla Q) \nabla R$

   1.3 $P \vee (Q \vee R) = (P \vee Q) \vee R$

   1.4 $P;(Q;R) = (P;Q);R$

2. Коммутативность

2.1 $P\|Q = Q\|P$

2.2 $P \bigtriangledown Q = Q \bigtriangledown P$

2.3 $P \vee Q = Q \vee P$

3. Дистрибутивность

3.1 $(P\|Q);R = (P;R)\|(Q;R)$

3.2 $P;(Q\|R) = (P;Q)\|(P;R)$

3.3 $(P \vee Q);R = (P;R) \vee (Q;R)$

3.4 $P;(Q \vee R) = (P;Q) \vee (P;R)$

3.5 $(P \vee Q)\|R = (P\|R) \vee (Q\|R)$

3.6 $P \bigtriangledown (Q\|R) = (P \bigtriangledown Q)\|(P \bigtriangledown R)$

4. Аксиомы для $\bigtriangledown$ и $\overline{\|}$

4.1 $P \bigtriangledown Q = (P\|(\overline{\|}Q)) \vee ((\overline{\|}P)\|Q)$

4.2 $\overline{\|}(P\|Q) = (\overline{\|}P)\|(\overline{\|}Q)$

4.3 $\overline{\|}(P \vee Q) = (\overline{\|}P) \vee (\overline{\|}Q)$

4.4 $\overline{\|}(P;Q) = (\overline{\|}P)\|(\overline{\|}Q)$

4.5 $\overline{\|}a = \bar{a}$

4.6 $\overline{\|}\bar{a} = \bar{a}$

4.7 $\overline{\|}\delta_a = \bar{a}$

5. Структурные свойства

5.1 $\bar{a};P = \bar{a}\|P$

5.2 $P;\bar{a} = P\|\bar{a}$

5.3 $P\|(P;Q) = (P;Q)$

5.4 $Q\|(P;Q) = (P;Q)$

5.5 $P;Q;R = (P;Q)\|(Q;R)$

5.6 $(P;Q)\|(Q;R) = (P;Q)\|(Q;R)\|(P;R)$

5.7 $P\|P = P$

5.8 $P \vee P = P$

5.9 $P \vee Q = P$, если $Q = pref(P)$ (понятие префикса будет определено дальше)

6. Аксиомы для тупиковых действий и $\tilde{\overline{\|}}$

6.1 $a\|\bar{a} = \delta_a$

6.2 $a;a = \delta_a$

6.3 $a\|\delta_a = \delta_a$

6.4 $\delta_a;P = \delta_a\|(\tilde{\Pi}P)$

6.5 $P;\delta_a = P\|\delta_a$

6.6 $\delta_a\|(\Pi P) = \delta_a\|(\tilde{\Pi}P)$

6.7 $\tilde{\Pi}a = \delta_a$

6.8 $\tilde{\Pi}\bar{a} = \delta_a$

6.9 $\tilde{\Pi}\delta_a = \delta_a$

6.10 $\tilde{\Pi}(P\|Q) = (\tilde{\Pi}P)\|(\tilde{\Pi}Q)$

6.11 $\tilde{\Pi}(P;Q) = (\tilde{\Pi}P)\|(\tilde{\Pi}Q)$

6.12 $\tilde{\Pi}(P \vee Q) = (\tilde{\Pi}P) \vee (\tilde{\Pi}Q)$

Для доказательства полноты системы аксиом Θ_2 вводится понятие канонической формы формулы AFP_2.

3.1.4 Каноническая форма формул

Введем ряд понятий, необходимых для построения правил переписывания.

Алфавит формулы P, обозначение $\alpha(P)$, — это подмножество алфавита действий α, которое определяется следующим образом.

1. $\alpha(a) = \alpha(\bar{a}) = \alpha(\delta_a) = a$;
2. $\alpha(\Pi P) = \alpha(\tilde{\Pi}P) = \alpha(P)$;
3. $\alpha(P \circ Q) = \alpha(P) \cup \alpha(Q)$, $\circ \in \{\|, \triangledown, ;, \vee\}$.

Содержимое формулы P, обозначение $cont(P)$, — множество символов из $\alpha(P) \cup \bar{\alpha}(P) \cup \Delta_\alpha(P)$ следующего вида.

1. $cont(a) = \{a\}$, $cont(\bar{a}) = \{\bar{a}\}$, $cont(\delta_a) = \{\delta_a\}$;
2. $cont(\Pi P) = cont(\tilde{\Pi}P) = cont(P)$;
3. $cont(P \circ Q) = cont(P) \cup cont(Q)$, $\circ \in \{\|, \triangledown, ;, \vee\}$.

Как следует из определения, $cont(P)$ — множество элементарных подформул формулы P. Определим "положительную" часть содержимого формулы P следующим образом: $cont^+(P) = cont(P) \cap \alpha$.

Введем понятия, связанные со структурой формулы.

Предшествование — формула вида $P_1;\ldots;P_n = ;_{i=1}^{n} P_i$, $P_i \in \alpha \cup \bar{\alpha} \cup \Delta_\alpha$ $(1 \le i \le n)$;

Конъюнкция — формула вида $P_1\|\ldots\|P_n = \|_{i=1}^{n} P_i$, где P_i — предшествования $(1 \le i \le n)$.

Нормальная конъюнкция — конъюнкция, для которой истинны следующие утверждения.

1. Каждая формула P_i $(1 \le i \le n)$ имеет одну из следующих форм:
   (a) элементарная формула a $(a \in \alpha)$, $\bar{a}(\bar{a} \in \bar{\alpha})$, δ_a $(\delta_a \in \Delta_\alpha)$;
   (b) *элементарное предшествование* $(a;b)$, где $a, b \in \alpha$ и $a \ne b$;
2. Если имеется формула P_i $(1 \le i \le n)$ в форме δ_a $(\delta_a \in \Delta_\alpha)$, тогда нет другой формулы P_j $(1 \le j \le n)$ такой, что $P_j = \bar{b}$ $(\bar{b} \in \bar{\alpha})$;
3. Для любых формул P_i и P_j $(1 \le i \ne j \le n)$ таких, что $\alpha(P_i) \cap \alpha(P_j) \ne \emptyset$ P_i и P_j должны иметь форму различных элементарных предшествований;
4. Для любой пары $P_i = (a;b)$ и $P_j = (b;c)$ $(1 \le i \ne j \le n)$ существует терм $P_k = (a;c)$ $(1 \le k \le n)$, описывающий транзитивное замыкание отношения предшествования для действий a, b и c.

Назовем 1 (соответственно 2, 3, 4)-*конъюнкцией* конъюнкцию, удовлетворяющую условию 1 (соответственно 2, 3, 4) из определения нормальной конъюнкции. Аналогично введем определение, например 1,2,3-конъюнкции. В соответствии с этими определениями нормальная конъюнкция есть 1,2,3,4-конъюнкция.

Пусть P и Q — нормальные конъюнкции. Формула P — *префикс* формулы Q, обозначение $P = pref(Q)$, если выполняются следующие условия.

1. $cont^+(P) \subset cont^+(Q)$;
2. элементарное предшествование $(a;b)$ является конъюнктивным членом Q и $b \in cont^+(P)$ тогда и только тогда, когда $(a;b)$ — конъюнктивный член P;

Например, в формуле $(a\|c\|\bar{b}\|\bar{d}\|\bar{e}) \vee (c\|\delta_a\|\delta_b\|\delta_d\|\delta_e) \vee (a\|\delta_b\|\delta_c\|\delta_d\|\delta_e) \vee ((b;d)\|(b;e)\|\bar{a}\|\bar{c})$ вторая и третья конъюнкции — префиксы первой.

Дизъюнкция — формула вида $P = P_1 \vee \ldots \vee P_n = \vee_{i=1}^{n} P_i$, где P_i $(1 \le i \le n)$ — конъюнкции.

Формула P имеет *каноническую форму*, если выполняются следующие условия: $P = \bigvee_{i=1}^{n} P_i$ — дизъюнкция и

1. P_i $(1 \le i \le n)$ — нормальная конъюнкция;

2. любые P_i и P_j $(1 \leq i \neq j \leq n)$ различны;

3. никакая из формул P_i и P_j $(1 \leq i \neq j \leq n)$ не является префиксом другой.

Заметим, что каждый дизъюнктивный член канонической формы формулы характеризует одно из возможных альтернативных поведений процесса и является представлением частичного порядка этого процесса.

Например, формула $(a||c||\bar{b}||\bar{d}||\bar{e}) \vee ((b;d)||(b;e)||\bar{a}||\bar{c})$ имеет каноническую форму.

Аналогично определениям для конъюнкций вводим определения 1 (2, 3 соответственно)-*дизъюнкций* и другие. Таким образом, каноническая форма — 1,2,3-дизъюнкция.

Конъюнкция (дизъюнкция) *максимальна*, если она не является конъюнктивным (дизъюнктивным) членом никакой другой конъюнкции (дизъюнкции).

Например, в формуле $((a||\bar{b}||c) \vee d)||e$ конъюнкции $a||b$ и $b||c$ не максимальные, так как они — конъюнктивные члены максимальной конъюнкции $a||\bar{b}||c$. Вся формула — не конъюнкция, так как ее первый конъюнктивный член — дизъюнкция.

Запись $P =_{\Theta_2} Q$ означает, что равенство формул P и Q может быть доказано с использованием системы аксиом Θ_2.

Формулы P и Q *изоморфны*, обозначение $P \simeq Q$, тогда и только тогда, когда P может быть сведена к Q (и наоборот) с использованием аксиом коммутативности для операций $||$, ∇, $\vee$ и ассоциативности для $||$, ∇, $\vee$, $;$.

Например, формулы $(a||\bar{b}||\bar{c}) \vee (c||\bar{a}||\bar{b})$ и $(\bar{a}||\bar{b}||c) \vee (b||a||\bar{c})$ изоморфны.

Следующие теоремы были доказаны в [8, 9].

Теорема 3 *Любая формула AFP_2 может быть сведена к единственной до изоморфизма канонической форме.*

Теорема 4 *Для любых формул P и Q алгебры AFP_2 истинно $P \approx_e Q \Leftrightarrow P =_{\Theta_2} Q$.*

Таким образом, мы можем выяснить, эквивалентны ли любые две формулы P и Q алгебры AFP_2. Для этого достаточно привести эти формулы к их каноническим формам P' и Q' и проверить их на изоморфизм.

3.2 Взаимосвязь эквивалентностей алгебры AFP_2 и ранее введенных

В пункте, посвященном аксиоматизации AFP_2, была введена обычная эквивалентность, связывающая формулы процессов, которые имеют одинаковые ЧУМ.

Пусть $\rho = (X, \prec)$ —ЧУМ. Тогда $\rho^+ = (X^+, \prec)$ — это *действительная, "наблюдаемая" часть* ρ над множеством действий. Если процесс P характеризуется множеством ЧУМ $\{\rho_i\}_{i=1}^n$, то есть $\mathcal{D}_2[P] = \{\rho_i\}_{i=1}^n$, тогда $\mathcal{D}_2^+[P] = \{\rho_i^+\}_{i=1}^n$ обозначает его *"наблюдаемую" часть* над множеством действий.

Два процесса P и Q *наблюдаемо эквивалентны*, запись $P \approx_+ Q$, если $\mathcal{D}_2^+[P] = \mathcal{D}_2^+[Q]$. Таким образом, наблюдаемая эквивалентность — ограничение обычной эквивалентности. Эта эквивалентность учитывает только ЧУМ над множеством действий и не берет в расчет не-действия и тупиковые действия. Она была введена в работах, посвященных алгебрам AFP_1 и AFP_2 на основе одноименной эквивалентности алгебры CCS, рассмотренной в [19, 20].

В [8] рассмотрена алгебра AFP_0, введенная в [18], в которой формулам соответствуют конечные A-сети. В этой же работе построено отображение Ψ из множества формул AFP_0 в соответствующее множество AFP_2 такое, что совокупность ЧУМ сети, описываемой формулой P в AFP_0, совпадает с множеством ЧУМ недетерминированного процесса, описываемого формулой $\Psi(P)$ в AFP_2. Это позволяет нам не различать формулы P и $\Psi(P)$ в семантике ЧУММ. Таким образом, если у нас есть сеть, определяемая формулой P в AFP_0, мы можем анализировать ее свойства и поведение посредством той же самой формулы P в AFP_2.

В таком случае очевидно, что понятия эквивалентностей в AFP_2 можно применить и к сетям, если рассматривать формулы AFP_2 над алфавитом α, которые содержат только операции $\|$, ∇, ;, как формулы AFP_0, определяющие некоторые сети. Тогда наблюдаемая эквивалентность совпадает с обычной ЧУММ-эквивалентностью, введенной ранее, так как в эквивалентности на ЧУММ рассматриваются только "наблюдаемые" действия. Обычная же эквивалентность алгебры AFP_2, рассматриваемая на формулах AFP_0, более строгая, чем наблюдаемая эквивалентность, так как она учитывает "негативную" информацию, которая дается не-действиями и тупиковыми действиями.

Рассмотрим сети, определяемые формулами $P = a\|(a \nabla b)\|b$ и $Q = (a\|(a \nabla b)\|b); c$. Очевидно, что $P \leftrightarrows_{prh} Q$, так как действие c в сети, определяемой формулой Q, никогда не сработает. Тем не менее, $P \not\approx_e Q$, так как $\mathcal{D}_2[P] = \{(\{a, \delta_b\}, \emptyset), (\{b, \delta_a\}, \emptyset)\}$, а $\mathcal{D}_2[Q] =$

$\{(\{a, \delta_b, \delta_c\}, \emptyset), (\{b, \delta_a, \delta_c\}, \emptyset)\}$, то есть тупиковое действие δ_c оказывает влияние на эту эквивалентность. Таким образом, обычная эквивалентность алгебры AFP_2 не является следствием никакой ранее введенной эквивалентности на сетях.

Кроме того, никакая из процессных эквивалентностей не является следствием обычной эквивалентности алгебры AFP_2, что демонстрирует следующий пример. Пусть $P = a; b$ и $Q = (a; b)\|a$. Тогда $\mathcal{D}_2[P] = \mathcal{D}_2[Q] = \{(\{a, b\}, \prec)\}$, где $a \prec b$. Следовательно, $P \approx_e Q$, но $P \not\equiv_{pr} Q$, так как в в сети, определяемой формулой Q, переход с пометкой a имеет дополнительное выходное место.

3.3 Автоматизация проверки эквивалентности формул

Процесс приведения формулы AFP_2 к каноническому виду с помощью аксиом системы Θ_2 довольно трудоемок, особенно если эта формула имеет большую длину и сложную структуру, что часто случается при формульном представлении реальных процессов. Кроме того, в процессе приведения аксиомы из Θ_2 приходится применять в ту и другую сторону.

Для того чтобы автоматически привести исходную формулу к одному из изоморфных между собой канонических видов, нужно создать систему правил переписывания, при применении которых формула приводится к нужному виду.

В соответствии с этими требованиями строится система правил переписывания RWS_2.

3.3.1 Система правил переписывания RWS_2

Перед описанием системы RWS_2 введем необходимое определение.

Замена в формуле P подформулы P_i на Q, запись $[P]^{P_i}_{Q}$, есть формула $P_1 \circ \ldots P_{i-1} \circ Q \circ P_{i+1} \circ \ldots \circ P_n$, где $P = P_1 \circ \ldots \circ P_{i-1} \circ P_i \circ P_{i+1} \circ \ldots \circ P_n$, $\circ \in \{\|, \triangledown, \vee, ;\}$.

В следующих правилах RWS_2 P, Q, R обозначают формулы AFP_2, a, b, $c \in \alpha$, $\bar{a}$, $\bar{b} \in \bar{\alpha}$, δ_a, $\delta_b \in \Delta_\alpha$, а цифры в скобках — номера равенств системы Θ_2, которые использовались при построении соответствующих правил.

1.1. $\circ \in \{\|, ;, \vee\} \Rightarrow$
$P \circ (Q \circ R) \to (P \circ Q) \circ R$
(1.1, 1.3, 1.4);

2.1. $(\bullet, \circ) \in \{(\|, ;), (\vee, ;), (\vee, \|)\} \Rightarrow$

$(P \circ Q) \bullet R \to (P \bullet R) \circ (Q \bullet R)$
(3.1, 3.3, 3.5);

2.2. $(\bullet, \circ) \in \{(\|, ;), (\vee, ;), (\vee, \|)\} \Rightarrow$
$P \bullet (Q \circ R) \to (P \bullet Q) \circ (P \bullet R)$
(2.1, 3.2, 3.4, 3.5);

3.1 $P \bigtriangledown Q \to (P \| (\sqcap Q)) \vee ((\sqcap P) \| Q)$
(4.1);

4.1. $\circ \in \{\|, ;\}, \neg \in \{\sqcap, \tilde{\sqcap}\} \Rightarrow$
$\neg(P \circ Q) \to (\neg P) \| (\neg Q)$
(4.2, 4.4, 6.10, 6.11);

4.2. $\neg \in \{\sqcap, \tilde{\sqcap}\} \Rightarrow$
$\neg(P \vee Q) \to (\neg P) \vee (\neg Q)$
(4.3, 6.12);

4.3. $P = a$ или $P = \bar{a}$ или $P = \delta_a \Rightarrow$
$\sqcap P \to \bar{a}$
(4.5, 4.6, 4.7);

4.4. $P = a$ или $P = \bar{a}$ или $P = \delta_a \Rightarrow$
$\tilde{\sqcap} P \to \delta_a$
(6.7, 6.8, 6.9);

5.1. $P, Q, R \in \alpha \cup \bar{\alpha} \cup \Delta_\alpha \Rightarrow$
$(P; Q); R \to ((P; Q) \| (Q; R)) \| (P; R)$
(5.5, 5.6);

5.2. $Q \in \alpha \cup \bar{\alpha} \cup \Delta_\alpha \Rightarrow$
$\bar{a}; Q \to a \| Q$
(5.1);

5.3. $P \in \alpha \cup \bar{\alpha} \cup \Delta_\alpha \Rightarrow$
$P; \bar{a} \to P \| \bar{a}$
(5.2);

5.4. $a; a \to \delta_a$
(6.2);

5.5. $Q = b$ или $Q = \bar{b}$ или $Q = \delta_b \Rightarrow$
$\delta_a; Q \to \delta_a \| \delta_b$
(6.4, 6.7, 6.8, 6.9);

5.6. $P \in \alpha \cup \bar{\alpha} \cup \Delta_\alpha \Rightarrow$
$P; \delta_a \to P \| \delta_a$
(6.5);

6.1. P — 1-конъюнкция, $P' = \delta_a$ — конъюнктивный член P $\Rightarrow$
$P\|\bar{b} \to P\|\delta_b$
(1.1, 2.1, 4.5, 6.6, 6.7);

6.2. P — 1-конъюнкция, $P' = \bar{b}$ — конъюнктивный член P $\Rightarrow$
$P\|\delta_a \to [P]^{P'}_{\delta_b}\|\delta_a$
(1.1, 2.1, 4.5, 6.6, 6.7);

7.1. P — 1,2-конъюнкция, P' — конъюнктивный член P, $P' = a$ или $P' = b$ $\Rightarrow$
$P\|(a;b) \to [P]^{P'}_{(a;b)}$
(1.1, 2.1, 5.3, 5.4);

7.2. P — 1,2-конъюнкция, P' — конъюнктивный член P, $P' = (a;b)$ или $P' = (b;a)$ $\Rightarrow$
$P\|a \to P$
(1.1, 2.1, 5.3, 5.4);

7.3. P — 1,2-конъюнкция, $P' = a$ — конъюнктивный член P, $\Diamond \in \{^-, \delta\}$ $\Rightarrow$
$P\|\Diamond a \to [P]^{P'}_{\delta_a}$
(1.1, 2.1, 6.1, 6.3);

7.4. P — 1,2-конъюнкция, P' — конъюнктивный член P, $P' = \bar{a}$ или $P' = \delta_a$ $\Rightarrow$
$P\|a \to [P]^{P'}_{\delta_a}$
(1.1, 2.1, 6.1, 6.3);

7.5. P — 1,2-конъюнкция, $P' = (a;b)$ — конъюнктивный член P, $\Diamond \in \{^-, \delta\}$ $\Rightarrow$
$P\|\Diamond a \to [P]^{P'}_{\delta_b}\|\delta_a$
(1.1, 1.4, 2.1, 5.1, 6.1, 6.3, 6.4, 6.7);

7.6. P — 1,2-конъюнкция, $P' = (b;a)$ — конъюнктивный член P, $\Diamond \in \{^-, \delta\}$ $\Rightarrow$
$P\|\Diamond a \to [P]^{P'}_{b}\|\delta_a$
(1.1, 2.1, 5.2, 6.1, 6.3, 6.5);

7.7. P — 1,2-конъюнкция, P' — конъюнктивный член P, $P' = \bar{a}$ или $P' = \delta_a$ $\Rightarrow$
$P\|(a;b) \to [P]^{P'}_{\delta_a}\|\delta_b$
(1.1, 1.4, 2.1, 5.1, 6.1, 6.3, 6.4, 6.7);

7.8. P — 1,2-конъюнкция, P' — конъюнктивный член P, $P' = a$ или $P' = \delta_a$ $\Rightarrow$
$P\|(b;a) \to [P]^{P'}_{\delta_a}\|b$
(1.1, 2.1, 5.2, 6.1, 6.3, 6.5);

7.9. P — 1,2-конъюнкция, $P' = Q$ — конъюнктивный член P $\Rightarrow$
$P||Q \to P$
(1.1, 2.1, 5.7);

8.1. P — 1,2,3-конъюнкция, $P' = (a;b)$ — конъюнктивный член P, в максимальной 1,2,3-конъюнкции, содержащей P в качестве конъюнктивного члена, нет члена $P'' = (a;c)$ $\Rightarrow$
$P||(b;c) \to (P||(b;c))||(a;c)$
(1.1, 2.1, 5.6);

8.2. P — 1;2,3-конъюнкция, $P' = (c;a)$ — конъюнктивный член P, в максимальной 1,2,3-конъюнкции, содержащей P в качестве конъюнктивного члена, нет члена $P'' = (b;a)$ $\Rightarrow$
$P||(b;c) \to (P||(b;c))||(b;a)$
(1.1, 2.1, 5.6);

9.1. P — 1-дизъюнкция, P' — дизъюнктивный член P, $P' \simeq Q$ $\Rightarrow$
$P \vee Q \to P$
(1.1, 1.3, 2.1, 2.3, 5.8);

10.1. P — 1,2-дизъюнкция, Q — нормальная конъюнкция, P' — дизъюнктивный член P, $Q = pref(P')$ $\Rightarrow$
$P \vee Q \to P$
(1.3, 2.3, 5.9);

10.2. P — 1,2-дизъюнкция, Q — нормальная конъюнкция, P' — дизъюнктивный член P, $P' = pref(Q)$ $\Rightarrow$
$P \vee Q \to [P]_Q^{P'}$
(1.3, 2.3, 5.9).

Сделаем краткий обзор системы правил переписывания.

Для избежания бесконечных цепочек вывода вида $P \circ (Q \circ R) \to (P \circ Q) \circ R \to P \circ (Q \circ R) \to \ldots$, $\circ \in \{||, \vee, ;\}$ вводится правило левой ассоциативности 1.1.

В систему RWS_2 нельзя включать правила коммутативности, применение которых может привести к бесконечным цепочкам вида $P \circ Q \to Q \circ P \to P \circ Q \to \ldots$, $\circ \in \{||, \vee\}$. Поэтому дополнительно вводятся симметричные правила, необходимые при отсутствии правил коммутативности. На этой идее основаны правила дистрибутивности группы 2.

Например, в системе Θ_2 нет аксиомы, симметричной аксиоме 3.5 $(P \vee Q)||R = (P||R) \vee (Q||R)$. Поэтому, если мы не включим в нашу систему нового симметричного правила, основанного на аксиоме $P||(Q \vee R) = (P||Q) \vee (P||R)$, мы не сможем преобразовать формулу $a||(b \vee c)$ к виду $(a||b) \vee (a||c)$.

Правило 3.1 позволяет избавиться от символа ∇, а правила группы 4 — от символов $\|$ и $\tilde{\|}$.

Правила группы 5 используются для удовлетворения свойству 1 из определения нормальной конъюнкции.

В связи с отсутствием правил коммутативности возникают и трудности другого рода, связанные с удалением друг от друга конъюнктивных (дизъюнктивных) членов формулы, к которым можно применить некоторую аксиому. Тогда в дополнение к подходу, основанному на введении симметричных правил, применяется другой метод. Рассмотрим формулу $P\|Q$ (или $P \vee Q$). В подформуле P, являющейся конъюнкцией (дизъюнкцией), ищется подформула P' такая, что к $P'\|Q$ (к $P' \vee Q$) применима некоторая аксиома из Θ_2.

На этой идее основаны правила группы 6, предназначенные для удовлетворения свойству 2 нормальной конъюнкции, группы 7, необходимые для выполнения свойства 3, правила группы 8 для удовлетворения свойству 4, а также правила групп 9 и 10 для выполнения свойств 2 и 3 канонической формы соответственно.

Например, аксиома 6.1 $a\|\bar{a} = \delta_a$ не применима при отсутствии правил коммутативности к формулам $a\|b\|\bar{a}$ и $\bar{a}\|b\|a$. Поэтому вводятся симметричные правила 7.3 и 7.4. В первом случае, при применении правила 7.3 ($P = a\|b$, $P' = a$) получаем формулу $\delta_a\|b$. Во втором случае эту же формулу получаем по правилу 7.4 ($P = \bar{a}\|b$, $P' = \bar{a}$).

Для избежания бесконечных цепочек вывода вида $(a;b)\|(b;c) \to ((a;b)\|(b;c))\|(a;c) \to (((a;b)\|(b;c))\|(a;c))\|(a;c) \to \ldots$ в правилах группы 8 конъюнктивный член — транзитивное замыкание отношения предшествования — ищется в максимальной 1,2,3-конъюнкции, содержащей данную. Правило применяется только тогда, когда такого члена в максимальной 1,2,3-конъюнкции нет. Этот прием предохраняет от бесконечного увеличения длины формулы в процессе ее приведения.

Необходимо отметить,что правила 6.1, 6.2, 7.5–7.8 основаны на новых аксиомах, полученных из аксиом Θ_2, доказательство которых приведено ниже. Цифры над знаками равенства — номера применяемых аксиом. Звездочка означает, что аксиома применяется справа налево. Цифры в скобках — номера новых, только что доказанных здесь аксиом.

1. $\bar{a}\|(a;b) \overset{5.1*}{=} \bar{a};(a;b) \overset{1.4}{=} (\bar{a};a);b \overset{5.1}{=} (\bar{a}\|a);b \overset{2.1}{=} (a\|\bar{a});b \overset{6.1}{=} \delta_a;b \overset{6.4}{=} \delta_a\|(\tilde{\|}b) \overset{6.7}{=} \delta_a\|\delta_b$;

2. $\delta_a\|(a;b) \overset{6.1*}{=} (a\|\bar{a})\|(a;b) \overset{1.1*}{=} a\|(\bar{a}\|(a;b)) \overset{(1)}{=} a\|(\delta_a\|\delta_b) \overset{1.1}{=} (a\|\delta_a)\|\delta_b \overset{6.3}{=} \delta_a\|\delta_b$;

3. $\bar{a}\|(b;a) \overset{2.1}{=} (b;a)\|\bar{a} \overset{5.2*}{=} (b;a);\bar{a} \overset{1.1*}{=} b;(a;\bar{a}) \overset{5.2}{=} b;(a\|\bar{a}) \overset{6.1}{=} b;\delta_a \overset{6.5}{=} b\|\delta_a \overset{2.1}{=} \delta_a\|b;$

4. $\delta_a\|(b;a) \overset{6.1*}{=} (a\|\bar{a})\|(b;a) \overset{1.1*}{=} a\|(\bar{a}\|(b;a)) \overset{(3)}{=} a\|(\delta_a\|b) \overset{1.1}{=} (a\|\delta_a)\|b \overset{6.3}{=} \delta_a\|b;$

5. $\delta_a\|\bar{b} \overset{4.5*}{=} \delta_a\|(\overline{\|}b) \overset{6.6}{=} \delta_a\|(\tilde{\|}b) \overset{6.7}{=} \delta_a\|\delta_b.$

3.3.2 Доказательство сходимости RWS_2

Докажем серию утверждений о системе RWS_2, гаранирующих, что по окончании процесса приведения формулы мы получим одну из изоморфных между собой канонических форм.

Утверждение 6 *К формуле AFP_2 не применимо ни одно из правил групп 1–5 тогда и только тогда, когда она является дизъюнкцией 1-конъюнкций.*

Доказательство.

$\Rightarrow$ Заметим, что, если к формуле не применимо правило 1.1, то все скобки, объединяющие подформулы, соединенные одинаковыми знаками операций, смещены влево. В дальнейшем будем предполагать, что все формулы уже обладают этим свойством.

Если к формуле не применимы правила 1.1 и 3.1, то в этой формуле нет операции ∇.

Очевидно также, что с помощью правил группы 4 мы избавляемся от $\|$ и $\tilde{\|}$. Если в формуле есть эти символы, стоящие перед сложными подформулами, то к ней обязательно применимы правила группы 4, смещающие знаки операций $\|$ и $\tilde{\|}$ к элементарным подформулам видов a, $\bar{a}$, δ_a. Затем, при применении правил 4.3 и 4.4, знаки $\|$ и $\tilde{\|}$ исчезают. Итак, если к формуле не применимы правила групп 1, 3, 4, эта формула без символов ∇, $\|$ и $\tilde{\|}$.

Если, в дополнение к этому, к формуле не применимы правила группы 2, приводящие формулу к виду $P = \vee_{i=1}^{m}\|_{j=1}^{n_i} P_{ij}$, то формула является дизъюнкцией конъюнкций с членами P_{ij}, являющимися предшествованиями или элементарными формулами.

Если к тому же к формуле не применимо правило 5.1, то можно заключить, что она — дизъюнкция конъюнкций с членами вида $P_{ij} = (Q_{ij};R_{ij})$ или $P_{ij} = Q_{ij}$, где Q_{ij}, $R_{ij} \in \alpha \cup \bar{\alpha} \cup \Delta_\alpha$.

Правила 5.2–5.6 позволяют избавиться от символов из $\bar{\alpha} \cup \Delta_\alpha$ и одинаковых символов в двучленных предшествованиях. Таким образом, если к формуле не применимы правила групп 1–5, она является дизъюнкцией конъюнкций с членами вида a, $\bar{a}$, δ_a или $(a;b)$, $a \neq b$, то есть эта формула — дизъюнкция 1-конъюнкций.

$\Leftarrow$ Так как формула — дизъюнкция 1-конъюнкций, в ней все скобки, объединяющие подформулы, соединенные одинаковыми

знаками операций, смещены влево. Значит, к ней не применимо правило 1.1. Кроме того, формула имеет вид $P = \vee_{i=1}^{m} \|_{j=1}^{n_i} P_{ij}$, где P_{ij} — элементарная формула или элементарное предшествование, то есть к формуле не применимы правила группы 2. Так как она не содержит символов ∇, $\overline{\|}$, $\tilde{\|}$, то к ней не применимы правила групп 3, 4. Из элементарности конъюнктов-предшествований следует неприменимость правил группы 5. □

Утверждение 7 *К формуле AFP_2 не применимо ни одно из правил групп 1–6 тогда и только тогда, когда она является дизъюнкцией 1,2-конъюнкций.*

Доказательство.

⇒ По предыдущему утверждению, наша формула — дизъюнкция 1-конъюнкций. Пусть для формулы не выполняется свойство 2 нормальной конъюнкции. Значит, в формуле есть дизъюнкт, в котором имеются конъюнктивные члены видов $\bar{b}$ и δ_a. Так как формула — дизъюнкция 1-конъюнкций, то этот дизъюнкт — 1-конъюнкция. Таким образом, к ней применимо правило 6.1 или 6.2, что противоречит исходному утверждению.

⇐ По предыдущему утверждению, к формуле не применимы правила групп 1–5. Правила группы 6 не применимы, так как любой ее дизъюнктивный член не содержит в качестве подформул одновременно элементы из $\bar{a}$ и δ_α. □

Утверждение 8 *К формуле AFP_2 не применимо ни одно из правил групп 1–7 тогда и только тогда, когда она является дизъюнкцией 1,2,3-конъюнкций.*

Доказательство.

⇒ По предыдущему утверждению, формула — дизъюнкция 1,2-конъюнкций. Достаточно доказать, что из неприменимости правил группы 7 следует выполнение условия 3 нормальной конъюнкции для всех дизъюнктивных членов формулы. Рассмотрим ситуации, когда для дизъюнктивного члена формулы, 1,2-конъюнкции $P_i = \|_{j=1}^{n_i} P_{ij}$, выполняется условие $\alpha(P_{ik}) \cap \alpha(P_{il}) \neq \emptyset$ $(1 \leq k \neq l \leq n)$ и P_{ik}, P_{il} не являются различными элементарными предшествованиями, то есть не выполняется условие 3. Мы покажем, что во всех этих ситуациях к формуле применимы правила, уменьшающие число таких пар дизъюнктов.

1. Конъюнкты видов $\bar{a}$ и δ_a. Для нашей формулы этот случай не имеет места из-за того, что для нее выполняется свойство 2 нормальной конъюнкции.

2. Конъюнктивные члены имеют вид a и $(a;b)$ или b и $(a;b)$. Тогда к конъюнкции, содержащей эти члены, применимо правило 7.1 или 7.2.

3. Конъюнктивные члены имеют вид a и $\bar{a}$ или a и δ_a или $\bar{a}$ и δ_a. Тогда к соответствующей конъюнкции применимо одно из правил 7.3, 7.4.

4. Конъюнктивные члены имеют вид $\bar{a}$ и $(a;b)$ или $\bar{b}$ и $(a;b)$, или δ_a и $(a;b)$, или δ_b и $(a;b)$. Тогда к конъюнкции применимо одно из правил 7.5–7.8.

5. Конъюнктивные члены имеют вид a и a или $\bar{a}$ и $\bar{a}$, или δ_a и δ_a, или $(a;b)$ и $(a;b)$, то есть совпадают. Тогда к конъюнкции применимо правило 7.9, уничтожающее лишний конъюнктивный член.

Связи между конъюнктивными членами наглядно представляются схемой на рисунке 3.1, в которой линии с номерами пунктов доказательства этого утверждения соединяют формулы, которые могут быть на месте P_{ik} и P_{il}. Эта схема показывает, что в утверждении рассмотрены все случаи невыполнимости условия 3 нормальной конъюнкции.

Значит, если для формулы не выполняется условие 3, к ней обязательно применимо какое-либо из правил группы 7, что противоречит исходному утверждению. Итак, при неприменимости к формуле правил групп 1–7 можно утверждать, что она — дизъюнкция 1,2,3-конъюнкций.

$\Leftarrow$ По предыдущему утверждению, к формуле не применимы правила групп 1–6. Правила группы 7 не применимы, так как в дизъюнктах 1,2,3-дизъюнкции нет конъюнктивных членов — не элементарных предшествований, содержащих одинаковые символы. □

Утверждение 9 *К формуле AFP_2 не применимо ни одно из правил групп 1–8, тогда и только тогда, когда она является 1-дизъюнкцией.*

Доказательство.

$\Rightarrow$ По предыдущему утверждению, формула — дизъюнкция 1,2,3-конъюнкций. Покажем, что из неприменимости правил группы 8 следует выполнение условия 4 нормальной конъюнкции для всех ее дизъюнктивных членов. Если правила 8.1 и 8.2 не применимы к формуле, то в каждом ее дизъюнктивном члене содержатся все транзитивные замыкания отношения предшествования, то есть для любого дизъюнкта выполняется условие 4 нормальной конъюнкции. В заключение заметим, что 1,2,3,4-конъюнкция

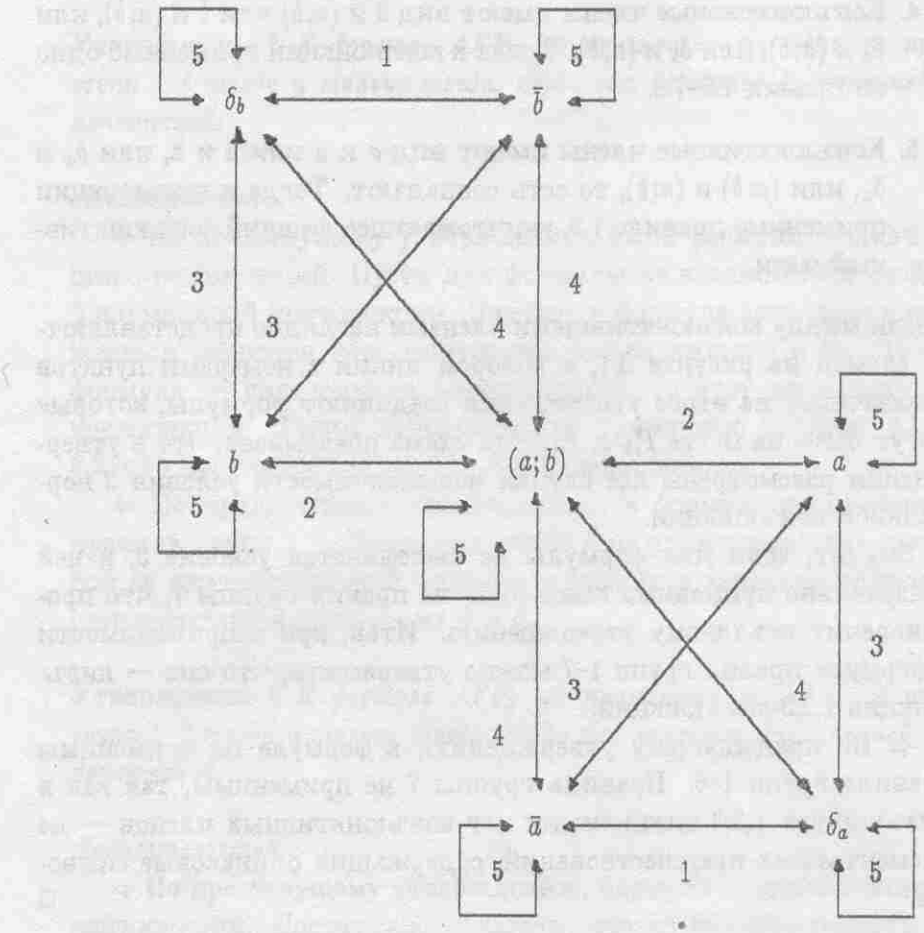


Рисунок 3.1: Конъюнкты с пересекающимися алфавитами

— нормальная конъюнкция, а дизъюнкция нормальных конъюнкций — 1-дизъюнкция.

⇐ По предыдущему утверждению, к формуле не применимы правила групп 1–7. Так как формула — 1-дизъюнкция, в любом ее дизъюнктивном члене содержатся все транзитивные замыкания отношения предшествования, то есть правила группы 8 не применимы. □

Утверждение 10 *К формуле AFP_2 не применимо ни одно из правил групп 1–9 тогда и только тогда, когда она — 1,2-дизъюнкция.*

Доказательство.

⇒ По предыдущему утверждению, формула — 1-дизъюнкция. Если правило 9.1 не применимо, то все дизъюнктивные члены различны, то есть формула — 1,2-дизъюнкция.

⇐ По предыдущему утверждению, к формуле не применимы правила групп 1–8. Правила группы 9 не применимы из-за того, что по свойству 2 канонической формы в ней все дизъюнктивные члены различны. □

Утверждение 11 *К формуле AFP_2 не применимо ни одно из правил RWS_2 тогда и только тогда, когда она находится в канонической форме.*

Доказательство.

⇒ По предыдущему утверждению, формула — 1,2-дизъюнкция. Если правила группы 10 не применимы к формуле, то в ней нет дизъюнктов, один из которых — префикс другого, то есть выполняется свойство 3 канонической формы, а 1,2,3-дизъюнкция — это каноническая форма формулы.

⇐ По предыдущему утверждению, к формуле не применимы правила групп 1–9. Правила группы 10 не применимы, так как в канонической форме нет ни одной пары префиксных один другому дизъюнктов. □

Таким образом, для проверки двух формул на эквивалентность достаточно, используя систему RWS_2, автоматически привести их к каноническим формам, а затем проверить эти канонические формы на изоморфизм. Автором написана программа CANON, основанная на утверждениях данного пункта, выполняющая автоматическое преобразование произвольной формулы AFP_2 к каноническому виду. Описание и примеры работы этой программы можно найти в приложениях.

Глава 4

Заключение

В данной работе была введена достаточно обширная группа эквивалентностей на сетях Петри. Были даны определения как обычных, так и бисимуляционных (обычных, ST- и сохраняющих историю) эквивалентностей в различных семантиках и выяснена взаимосвязь между данными эквивалентностями на сетях с конечными процессами без λ-действий. Также было проверено, как ведут себя эти эквивалентности на различных классах сетей. Кроме того, была рассмотрена алгебра AFP_2 и показано, как эквивалентности на формулах процессов, определенные в этой алгебре, связаны с сетевыми эквивалентностями. Также разработана система правил переписывания RWS_2, на основе которой была написана программа CANON, позволяющая автоматически проверять формулы алгебры AFP_2 на эквивалентность.

Дальнейшим развитием работы могло бы послужить выяснение полной картины взаимосвязей между введенными в главе 2 эквивалентностями на C-сетях, на сетях со строгой пометкой, а также исследование эквивалентностей на конечных A-сетях, которые описываются формулами алгебры AFP_0.

Еще одним направлением развития данной темы могло бы служить рассмотрение введенных эквивалентностей на более широком классе сетей, а именно, на сетях с λ-действиями. Возможно, на этих сетях некоторые эквивалентности перестали бы быть связанными между собой. В работах [26, 27] рассмотрен пример на структурах событий с λ-действиями, показывающий, что ST-бисимуляционные эквивалентности и сохраняющая историю бисимуляционная эквивалентность независимы на структурах событий этого типа.

Кроме того, хотелось бы понять, можно ли ввести бисимуляционную эквивалентность на формулах алгебры AFP_2, или же для

введения этой эквивалентности необходимо каким-то образом расширить алгебру, чтобы в ней можно было бы описывать процессы с несколькими одноименными действиями.

Изучение того, как связаны ST- и сохраняющие историю бисимуляционные эквивалентности с бисимуляционными эквивалентностями на местах, введенными в [1, 2], также является одним из направлений дальнейшего исследования.

Приложение А

Описание программы CANON

Программа CANON объемом около 3000 строк на языке Си предназначена для преобразования произвольных формул алгебры AFP_2 к каноническому виду. Эта программа основана на утверждениях главы 3.

Тело функции main имеет следующий вид.

```
вывод предварительной информации о предназначении
программы и формате вводимой формулы;
представление формулы в виде двусвязного списка;
выдача сообщения "формула считана";
преобразование списка в двусвязное дерево;
уничтожение списка;
печать формулы;
step=1; /*номер шага*/
do
{
 вывод step;
 nar=0; /*число применений правил на шаге
          с номером step*/
 применение правил;
 вывод nar;
 step++; /*следующий шаг*/
}
while(nar!=0);
вывод канонической формы формулы;
```

Формула должна вводиться по определенному формату. Симво-

Исходное обозначение символа	∇	$\vee$	$\|\|$	;	$\overline{\prod}$	$\tilde{\prod}$	$^-$	δ
Имя символьной константы	ALT	DSJ	CNC	PRC	NOC	MNO	NOT	DLT
Изображение символа	#	+	|	;	'	"	-	*

Таблица А.1: Представление символов в программе CANON

лы ∇, $\vee$, $||$, ;, $\overline{\prod}$, $\tilde{\prod}$, $^-$, δ заменяются на имеющиеся на клавиатуре в соответствии с таблицей А.1.

Кроме того,формула может иметь один из следующих видов.

1. a;
2. -a , *a ;
3. 'a , "a ;
4. '(P) , "(P) ;
5. a#b , a+b , a|b , a;b ;
6. a#(P) , a+(P) , a|(P) , a;(P) ;
7. (P)#a , (P)+a , (P)|a , (P);a ;
8. (P)#(Q) , (P)+(Q) , (P)|(Q) , (P);(Q) .

Здесь P и Q — формулы видов 2–8, a и b — символы элементарных действий. Как видно из описания формата формулы, она не должна иметь внешних скобок.

В процессе ввода формула представляется в виде двусвязного списка с элементами, которые содержат указатели на левого и правого соседей, а также один символ. Затем по этому списку строится представление формулы в виде двусвязного дерева с вершинами, содержащими указатели на отца, 1 и 2 сыновей, а также два символа, которые могут представлять элементарную формулу, то есть формулу видов 1 и 2. Таким образом, формула преобразуется в дерево с промежуточными вершинами, которые соответствуют функциональным символам и листьями, представляющими элементарные формулы. В этом представлении подформулам соответствуют поддеревья.

После этого, для экономии памяти, список уничтожается и формула, представленная в виде дерева, печатается (проверка правильности преобразования в дерево).

Потом начинается процесс применения к формуле правил. На каждом шаге с номером step просматриваются все правила, и, если они применимы, применяются, а число прменений правил на данном шаге, kar, увеличивается. В конце шага проверяется, равно ли число примененных на нем правил нулю. Если это так, то ни одно из правил не применимо к формуле, а это, по утверждению главы 3 означает, что она имеет канонический вид. В этом случае циклическое применение правил заканчивается, формула выводится на экран и программа заканчивает работу. В противном случае номер шага увеличивается на единицу, и к формуле опять применяются правила.

Назовем правило *непосредственно применимым* к дереву, если оно применимо ко всему этому дереву (то есть формула, соответствующая дереву, интерпретируется как левая часть некоторого правила переписывания). Правило *косвенно применимо* к дереву, если оно непосредственно применимо к к некоторому его поддереву, за исключением самого этого дерева. Правило *применимо* к дереву, если оно непосредственно или косвенно применимо к нему.

Например, правило 1.1 косвенно применимо к дереву, соответствующему формуле $a||(b;(c;d))$, а именно, к правому его поддереву, корневая вершина которого содержит первый символ предшествования.

Тела правил имеют следующий вид.

```
if(root!=NULL)
{
 if(правило непосредственно применимо к
    дереву с указателем на корень root)
 {
  установка указателей на поддеревья,
  соответствующие подформулам в
  правиле вывода;
  вывод информации о применяемом
  правиле и печать поддеревьев;
  преобразование дерева в соответствии
  с применяемым правилом;
  (*addrnar)++; /*увеличение на единицу счетчика числа
                  примененных на данном шаге правил*/
  вывод новой формулы;
 }
```

```
else
{
 применение правила к левому поддереву;
 применение правила к правому поддереву;
}
}
```

Таким образом, при применении правила к дереву просматриваются все его поддеревья, пока указатель на корень текущего поддерева, root, не станет равным NULL. То есть, если правило применимо к дереву, оно применяется.

Приложение В

Примеры работы программы CANON

В этой главе демонстрируется работа программы CANON по приведению формул алгебры AFP_2 к каноническому виду.

Формула $(a;(b \bigtriangledown c))|||(a \bigtriangledown b)$.

```
this program writed by Tarasyuk I.V.
program CANON transforms AFP2-formula
to canonical form
AFP2-formula may be in one of the next forms
1) a
2) -a  *a
3) 'a  "a
4) '(P)  "(P)
5) a#b  a+b  a|b  a;b
6) a#(P)  a+(P)  a|(P)  a;(P)
7) (P)#a  (P)+a  (P)|a  (P);a
8) (P)#(Q)  (P)+(Q)  (P)|(Q)  (P);(Q)
where a and b are symbols of elementary actions,
P and Q are formulas types 2-8
input formula
sign of end is EOF
formula has been read
your formula is:
(a;(b#c))|(a#b)
step 1
rule 3.1 is applyed
```

```
P=b
Q=c
new formula is:
(a;((b|('c))+(('b)|c)))|(a#b)
rule 3.1 is applyed
P=a
Q=b
new formula is:
(a;((b|('c))+(('b)|c)))|((a|('b))+(('a)|b))
rule 4.3 is applyed
P=c
new formula is:
(a;((b|(-c))+(('b)|c)))|((a|('b))+(('a)|b))
rule 4.3 is applyed
P=b
new formula is:
(a;((b|(-c))+((-b)|c)))|((a|('b))+(('a)|b))
rule 4.3 is applyed
P=b
new formula is:
(a;((b|(-c))+((-b)|c)))|((a|(-b))+(('a)|b))
rule 4.3 is applyed
P=a
new formula is:
(a;((b|(-c))+((-b)|c)))|((a|(-b))+((-a)|b))
number of applyed rules on step 1 is 6
step 2
rule 2.2 is applyed
P=(a;((b|(-c))+((-b)|c)))
Q=(a|(-b))
R=((-a)|b)
new formula is:
((a;((b|(-c))+((-b)|c)))|(a|(-b)))+
((a;((b|(-c))+((-b)|c)))|((-a)|b))
number of applyed rules on step 2 is 1
step 3
rule 1.1 is applyed
P=(a;((b|(-c))+((-b)|c)))
Q=a
R=(-b)
new formula is:
(((a;((b|(-c))+((-b)|c)))|a)|(-b))+
((a;((b|(-c))+((-b)|c)))|((-a)|b))
rule 1.1 is applyed
```

```
P=(a;((b|(-c))+((-b)|c)))
Q=(-a)
R=b
new formula is:
(((a;((b|(-c))+((-b)|c)))|a)|(-b))+
(((a;((b|(-c))+((-b)|c)))|(-a))|b)
rule 2.2 is applyed
P=a
Q=(b|(-c))
R=((-b)|c)
new formula is:
((((a;(b|(-c)))+(a;((-b)|c)))|a)|(-b))+
(((a;((b|(-c))+((-b)|c)))|(-a))|b)
rule 2.2 is applyed
P=a
Q=(b|(-c))
R=((-b)|c)
new formula is:
((((a;(b|(-c)))+(a;((-b)|c)))|a)|(-b))+
((((a;(b|(-c)))+(a;((-b)|c)))|(-a))|b)
number of applyed rules on step 3 is 4
step 4
rule 2.1 is applyed
P=(a;(b|(-c)))
Q=(a;((-b)|c))
R=a
new formula is:
((((a;(b|(-c)))|a)+((a;((-b)|c))|a))|(-b))+
((((a;(b|(-c)))+(a;((-b)|c)))|(-a))|b)
rule 2.1 is applyed
P=(a;(b|(-c)))
Q=(a;((-b)|c))
R=(-a)
new formula is:
((((a;(b|(-c)))|a)+((a;((-b)|c))|a))|(-b))+
((((a;(b|(-c)))|(-a))+((a;((-b)|c))|(-a)))|b)
rule 2.2 is applyed
P=a
Q=b
R=(-c)
new formula is:
(((((a;b)|(a;(-c)))|a)+((a;((-b)|c))|a))|(-b))+
((((a;(b|(-c)))|(-a))+((a;((-b)|c))|(-a)))|b)
rule 2.2 is applyed
```

```
P=a
Q=(-b)
R=c
new formula is:
(((((a;b)|(a;(-c)))|a)+(((a;(-b))|(a;c))|a))|(-b))+
((((a;(b|(-c)))|(-a))+((a;((-b)|c))|(-a)))|b)
rule 2.2 is applyed
P=a
Q=b
R=(-c)
new formula is:
(((((a;b)|(a;(-c)))|a)+(((a;(-b))|(a;c))|a))|(-b))+
(((((a;b)|(a;(-c)))|(-a))+((a;((-b)|c))|(-a)))|b)
rule 2.2 is applyed
P=a
Q=(-b)
R=c
new formula is:
(((((a;b)|(a;(-c)))|a)+(((a;(-b))|(a;c))|a))|(-b))+
(((((a;b)|(a;(-c)))|(-a))+(((a;(-b))|(a;c))|(-a)))|b)
rule 5.3 is applyed
P=a
Q=(-c)
new formula is:
(((((a;b)|(a|(-c)))|a)+(((a;(-b))|(a;c))|a))|(-b))+
(((((a;b)|(a;(-c)))|(-a))+(((a;(-b))|(a;c))|(-a)))|b)
rule 5.3 is applyed
P=a
Q=(-b)
new formula is:
(((((a;b)|(a|(-c)))|a)+(((a|(-b))|(a;c))|a))|(-b))+
(((((a;b)|(a;(-c)))|(-a))+(((a;(-b))|(a;c))|(-a)))|b)
rule 5.3 is applyed
P=a
Q=(-c)
new formula is:
(((((a;b)|(a|(-c)))|a)+(((a|(-b))|(a;c))|a))|(-b))+
(((((a;b)|(a|(-c)))|(-a))+(((a;(-b))|(a;c))|(-a)))|b)
rule 5.3 is applyed
P=a
Q=(-b)
new formula is:
(((((a;b)|(a|(-c)))|a)+(((a|(-b))|(a;c))|a))|(-b))+
(((((a;b)|(a|(-c)))|(-a))+(((a|(-b))|(a;c))|(-a)))|b)
```

```
number of applyed rules on step 4 is 10
step 5
rule 1.1 is applyed
P=(a;b)
Q=a
R=(-c)
new formula is:
((((((a;b)|a)|(-c))|a)+(((a|(-b))|(a;c))|a))|(-b))+
(((((a;b)|(a|(-c)))|(-a))+(((a|(-b))|(a;c))|(-a)))|b)
rule 1.1 is applyed
P=(a;b)
Q=a
R=(-c)
new formula is:
((((((a;b)|a)|(-c))|a)+(((a|(-b))|(a;c))|a))|(-b))+
((((((a;b)|a)|(-c))|(-a))+(((a|(-b))|(a;c))|(-a)))|b)
rule 2.1 is applyed
P=((((a;b)|a)|(-c))|a)
Q=(((a|(-b))|(a;c))|a)
R=(-b)
new formula is:
((((((a;b)|a)|(-c))|a)|(-b))+((((a|(-b))|(a;c))|a)|(-b)))+
((((((a;b)|a)|(-c))|(-a))+(((a|(-b))|(a;c))|(-a)))|b)
rule 2.1 is applyed
P=((((a;b)|a)|(-c))|(-a))
Q=(((a|(-b))|(a;c))|(-a))
R=b
new formula is:
((((((a;b)|a)|(-c))|a)|(-b))+((((a|(-b))|(a;c))|a)|(-b)))+
((((((a;b)|a)|(-c))|(-a))|b)+((((a|(-b))|(a;c))|(-a))|b))
number of applyed rules on step 5 is 4
step 6
rule 1.1 is applyed
P=((((((a;b)|a)|(-c))|a)|(-b))+
((((a|(-b))|(a;c))|a)|(-b)))
Q=(((((a;b)|a)|(-c))|(-a))|b)
R=((((a|(-b))|(a;c))|(-a))|b)
new formula is:
(((((((a;b)|a)|(-c))|a)|(-b))+((((a|(-b))|(a;c))|a)|(-b)))+
(((((a;b)|a)|(-c))|(-a))|b))+((((a|(-b))|(a;c))|(-a))|b)
number of applyed rules on step 6 is 1
step 7
rule 7.1 is applyed
P=(a|(-b))
```

```
P'=a
Q=(a;c)
new formula is:
(((((((a;b)|a)|(-c))|a)|(-b))+((((a;c)|(-b))|a)|(-b)))+
(((((a;b)|a)|(-c))|(-a))|b))+((((a|(-b))|(a;c))|(-a))|b)
rule 7.1 is applyed
P=(a|(-b))
P'=a
Q=(a;c)
new formula is:
(((((((a;b)|a)|(-c))|a)|(-b))+((((a;c)|(-b))|a)|(-b)))+
(((((a;b)|a)|(-c))|(-a))|b))+((((a;c)|(-b))|(-a))|b)
rule 7.2 is applyed
P=(((a;b)|a)|(-c))
P'=(a;b)
Q=a
new formula is:
((((((a;b)|a)|(-c))|(-b))+((((a;c)|(-b))|a)|(-b)))+
(((((a;b)|a)|(-c))|(-a))|b))+((((a;c)|(-b))|(-a))|b)
rule 7.2 is applyed
P=((a;c)|(-b))
P'=(a;c)
Q=a
new formula is:
((((((a;b)|a)|(-c))|(-b))+(((a;c)|(-b))|(-b)))+
(((((a;b)|a)|(-c))|(-a))|b))+((((a;c)|(-b))|(-a))|b)
rule 7.2 is applyed
P=((((a;b)|a)|(-c))|(-a))
P'=(a;b)
Q=b
new formula is:
((((((a;b)|a)|(-c))|(-b))+(((a;c)|(-b))|(-b)))+
((((a;b)|a)|(-c))|(-a)))+((((a;c)|(-b))|(-a))|b)
rule 7.3 is applyed
P=(((a;b)|a)|(-c))
P'=a
Q=(-a)
new formula is:
((((((a;b)|a)|(-c))|(-b))+(((a;c)|(-b))|(-b)))+
(((a;b)|(*a))|(-c)))+((((a;c)|(-b))|(-a))|b)
rule 7.4 is applyed
P=(((a;c)|(-b))|(-a))
P'=(-b)
Q=b
```

```
new formula is:
((((((a;b)|a)|(-c))|(-b))+(((a;c)|(-b))|(-b)))+
(((a;b)|(*a))|(-c)))+(((a;c)|(*b))|(-a))
rule 7.5 is applyed
P=(a;b)
P'=(a;b)
Q=(*a)
new formula is:
((((((a;b)|a)|(-c))|(-b))+(((a;c)|(-b))|(-b)))+
(((*b)|(*a))|(-c)))+(((a;c)|(*b))|(-a))
rule 7.5 is applyed
P=((a;c)|(*b))
P'=(a;c)
Q=(-a)
new formula is:
((((((a;b)|a)|(-c))|(-b))+(((a;c)|(-b))|(-b)))+
(((*b)|(*a))|(-c)))+(((*c)|(*b))|(*a))
rule 7.6 is applyed
P=(((a;b)|a)|(-c))
P'=(a;b)
Q=(-b)
new formula is:
(((((a|a)|(-c))|(*b))+(((a;c)|(-b))|(-b)))+
(((*b)|(*a))|(-c)))+(((*c)|(*b))|(*a))
rule 7.9 is applyed
P=a
P'=a
Q=a
new formula is:
((((a|(-c))|(*b))+(((a;c)|(-b))|(-b)))+
(((*b)|(*a))|(-c)))+(((*c)|(*b))|(*a))
rule 7.9 is applyed
P=((a;c)|(-b))
P'=(-b)
Q=(-b)
new formula is:
((((a|(-c))|(*b))+((a;c)|(-b)))+
(((*b)|(*a))|(-c)))+(((*c)|(*b))|(*a))
number of applyed rules on step 7 is 12
step 8
rule 6.1 is applyed
P=((*b)|(*a))
P'=(*a)
Q=(-c)
```

```
new formula is:
((((a|(-c))|(*b))+((a;c)|(-b)))+
(((*b)|(*a))|(*c)))+(((*c)|(*b))|(*a))
rule 6.2 is applyed
P=(a|(-c))
P'=(-c)
Q=(*b)
new formula is:
((((a|(*c))|(*b))+((a;c)|(-b)))+
(((*b)|(*a))|(*c)))+(((*c)|(*b))|(*a))
number of applyed rules on step 8 is 2
step 9
rule 9.1 is applyed
P=((((a|(*c))|(*b))+((a;c)|(-b)))+
(((*b)|(*a))|(*c)))
P'=(((*b)|(*a))|(*c))
Q=(((*c)|(*b))|(*a))
new formula is:
(((a|(*c))|(*b))+((a;c)|(-b)))+(((*b)|(*a))|(*c))
number of applyed rules on step 9 is 1
step 10
rule 10.1 is applyed
P=(((a|(*c))|(*b))+((a;c)|(-b)))
P'=((a;c)|(-b))
Q=(((*b)|(*a))|(*c))
new formula is:
((a|(*c))|(*b))+((a;c)|(-b))
rule 10.2 is applyed
P=((a|(*c))|(*b))
P'=((a|(*c))|(*b))
Q=((a;c)|(-b))
new formula is:
(a;c)|(-b)
number of applyed rules on step 10 is 2
step 11
number of applyed rules on step 11 is 0
out form is:
(a;c)|(-b)
```

Каноническая форма формулы $(a;c)|||\bar{b}$.

Формула $(a \bigtriangledown (b;e))|||(d \bigtriangledown (c;e))$.

```
this program writed by Tarasyuk I.V.
program CANON transforms AFP2-formula
to canonical form
AFP2-formula may be in one of the next forms
1) a
2) -a  *a
3) 'a  ~a
4) '(P)  ~(P)
5) a#b  a+b  a|b  a;b
6) a#(P)  a+(P)  a|(P)  a;(P)
7) (P)#a  (P)+a  (P)|a  (P);a
8) (P)#(Q)  (P)+(Q)  (P)|(Q)  (P);(Q)
where a and b are symbols of elementary actions,
P and Q are formulas types 2-8
input formula
sign of end is EOF
formula has been read
your formula is:
(a#(b;e))|(d#(c;e))
step 1
rule 3.1 is applyed
P=a
Q=(b;e)
new formula is:
((a|('(b;e)))+(('a)|(b;e)))|(d#(c;e))
rule 3.1 is applyed
P=d
Q=(c;e)
new formula is:
((a|('(b;e)))+(('a)|(b;e)))|((d|('(c;e)))+(('d)|(c;e)))
rule 4.1 is applyed
P=b
Q=e
new formula is:
((a|(('b)|('e)))+(('a)|(b;e)))|((d|('(c;e)))+(('d)|(c;e)))
rule 4.1 is applyed
P=c
Q=e
new formula is:
((a|(('b)|('e)))+(('a)|(b;e)))|((d|(('c)|('e)))+(('d)|(c;e)))
rule 4.3 is applyed
P=b
new formula is:
((a|((-b)|('e)))+(('a)|(b;e)))|((d|(('c)|('e)))+(('d)|(c;e)))
```

```
rule 4.3 is applyed
P=e
new formula is:
((a|((-b)|(-e)))+((('a)|(b;e)))|((d|((('c)|('e)))+((('d)|(c;e)))
rule 4.3 is applyed
P=a
new formula is:
((a|((-b)|(-e)))+((-a)|(b;e)))|((d|((('c)|('e)))+((('d)|(c;e)))
rule 4.3 is applyed
P=c
new formula is:
((a|((-b)|(-e)))+((-a)|(b;e)))|((d|((-c)|('e)))+((('d)|(c;e)))
rule 4.3 is applyed
P=e
new formula is:
((a|((-b)|(-e)))+((-a)|(b;e)))|((d|((-c)|(-e)))+((('d)|(c;e)))
rule 4.3 is applyed
P=d
new formula is:
((a|((-b)|(-e)))+((-a)|(b;e)))|((d|((-c)|(-e)))+((-d)|(c;e)))
number of applyed rules on step 1 is 10
step 2
rule 1.1 is applyed
P=a
Q=(-b)
R=(-e)
new formula is:
(((a|(-b))|(-e))+((-a)|(b;e)))|((d|((-c)|(-e)))+((-d)|(c;e)))
rule 1.1 is applyed
P=d
Q=(-c)
R=(-e)
new formula is:
(((a|(-b))|(-e))+((-a)|(b;e)))|(((d|(-c))|(-e))+((-d)|(c;e)))
rule 2.1 is applyed
P=((a|(-b))|(-e))
Q=((-a)|(b;e))
R=(((d|(-c))|(-e))+((-d)|(c;e)))
new formula is:
(((a|(-b))|(-e))|(((d|(-c))|(-e))+((-d)|(c;e))))+
(((-a)|(b;e))|(((d|(-c))|(-e))+((-d)|(c;e))))
rule 2.2 is applyed
P=((a|(-b))|(-e))
Q=((d|(-c))|(-e))
```

```
R=((-d)|(c;e))
new formula is:
((((a|(-b))|(-e))|((d|(-c))|(-e)))+
(((a|(-b))|(-e))|((-d)|(c;e))))+
(((-a)|(b;e))|(((d|(-c))|(-e))+((-d)|(c;e))))
rule 2.2 is applyed
P=((-a)|(b;e))
Q=((d|(-c))|(-e))
R=((-d)|(c;e))
new formula is:
((((a|(-b))|(-e))|((d|(-c))|(-e)))+
(((a|(-b))|(-e))|((-d)|(c;e))))+
((((-a)|(b;e))|((d|(-c))|(-e)))+
(((-a)|(b;e))|((-d)|(c;e))))
number of applyed rules on step 2 is 5
step 3
rule 1.1 is applyed
P=((((a|(-b))|(-e))|((d|(-c))|(-e)))+
(((a|(-b))|(-e))|((-d)|(c;e))))
Q=(((-a)|(b;e))|((d|(-c))|(-e)))
R=(((-a)|(b;e))|((-d)|(c;e)))
new formula is:
(((((a|(-b))|(-e))|((d|(-c))|(-e)))+
(((a|(-b))|(-e))|((-d)|(c;e))))+
(((-a)|(b;e))|((d|(-c))|(-e))))+
(((-a)|(b;e))|((-d)|(c;e)))
number of applyed rules on step 3 is 1
step 4
rule 1.1 is applyed
P=((a|(-b))|(-e))
Q=(d|(-c))
R=(-e)
new formula is:
((((((a|(-b))|(-e))|(d|(-c)))|(-e))+
(((a|(-b))|(-e))|((-d)|(c;e))))+
(((-a)|(b;e))|((d|(-c))|(-e))))+
(((-a)|(b;e))|((-d)|(c;e)))
rule 1.1 is applyed
P=((a|(-b))|(-e))
Q=(-d)
R=(c;e)
new formula is:
((((((a|(-b))|(-e))|(d|(-c)))|(-e))+
((((a|(-b))|(-e))|(-d))|(c;e)))+
```

```
(((-a)|(b;e))|((d|(-c))|(-e))))+
(((-a)|(b;e))|((-d)|(c;e)))
rule 1.1 is applyed
P=((-a)|(b;e))
Q=(d|(-c))
R=(-e)
new formula is:
((((((a|(-b))|(-e))|(d|(-c)))|(-e))+
((((a|(-b))|(-e))|(-d))|(c;e)))+
((((-a)|(b;e))|(d|(-c)))|(-e)))+
(((-a)|(b;e))|((-d)|(c;e)))
rule 1.1 is applyed
P=((-a)|(b;e))
Q=(-d)
R=(c;e)
new formula is:
((((((a|(-b))|(-e))|(d|(-c)))|(-e))+
((((a|(-b))|(-e))|(-d))|(c;e)))+
((((-a)|(b;e))|(d|(-c)))|(-e)))+
((((-a)|(b;e))|(-d))|(c;e))
number of applyed rules on step 4 is 4
step 5
rule 1.1 is applyed
P=((a|(-b))|(-e))
Q=d
R=(-c)
new formula is:
(((((((a|(-b))|(-e))|d)|(-c))|(-e))+
((((a|(-b))|(-e))|(-d))|(c;e)))+
((((-a)|(b;e))|(d|(-c)))|(-e)))+
((((-a)|(b;e))|(-d))|(c;e))
rule 1.1 is applyed
P=((-a)|(b;e))
Q=d
R=(-c)
new formula is:
(((((((a|(-b))|(-e))|d)|(-c))|(-e))+
((((a|(-b))|(-e))|(-d))|(c;e)))+
(((((-a)|(b;e))|d)|(-c))|(-e)))+
((((-a)|(b;e))|(-d))|(c;e))
number of applyed rules on step 5 is 2
step 6
rule 7.6 is applyed
P=((((-a)|(b;e))|d)|(-c))
```

```
P'=(b;e)
Q=(-e)
new formula is:
(((((((a|(-b))|(-e))|d)|(-c))|(-e))+
((((a|(-b))|(-e))|(-d))|(c;e)))+
(((((-a)|b)|d)|(-c))|(*e)))+
((((-a)|(b;e))|(-d))|(c;e))
rule 7.8 is applyed
P=(((a|(-b))|(-e))|(-d))
P'=(-e)
Q=(c;e)
new formula is:
(((((((a|(-b))|(-e))|d)|(-c))|(-e))+
((((a|(-b))|(*e))|(-d))|c))+
(((((-a)|b)|d)|(-c))|(*e)))+
((((-a)|(b;e))|(-d))|(c;e))
rule 7.9 is applyed
P=((((a|(-b))|(-e))|d)|(-c))
P'=(-e)
Q=(-e)
new formula is:
((((((a|(-b))|(-e))|d)|(-c))+
((((a|(-b))|(*e))|(-d))|c))+
(((((-a)|b)|d)|(-c))|(*e)))+
((((-a)|(b;e))|(-d))|(c;e))
number of applyed rules on step 6 is 3
step 7
rule 6.1 is applyed
P=((a|(-b))|(*e))
P'=(*e)
Q=(-d)
new formula is:
((((((a|(-b))|(-e))|d)|(-c))+
((((a|(-b))|(*e))|(*d))|c))+
(((((-a)|b)|d)|(-c))|(*e)))+
((((-a)|(b;e))|(-d))|(c;e))
rule 6.2 is applyed
P=((a|(-b))|(*e))
P'=(-b)
Q=(*d)
new formula is:
((((((a|(-b))|(-e))|d)|(-c))+
((((a|(*b))|(*e))|(*d))|c))+
(((((-a)|b)|d)|(-c))|(*e)))+
```

```
((((-a)|(b;e))|(-d))|(c;e))
rule 6.2 is applyed
P=((((-a)|b)|d)|(-c))
P'=(-c)
Q=(*e)
new formula is:
((((((a|(-b))|(-e))|d)|(-c))+
((((a|(*b))|(*e))|(*d))|c))+
(((((-a)|b)|d)|(*c))|(*e)))+
((((-a)|(b;e))|(-d))|(c;e))
number of applyed rules on step 7 is 3
step 8
rule 6.2 is applyed
P=((((-a)|b)|d)|(*c))
P'=(-a)
Q=(*e)
new formula is:
((((((a|(-b))|(-e))|d)|(-c))+
((((a|(*b))|(*e))|(*d))|c))+
(((((*a)|b)|d)|(*c))|(*e)))+
((((-a)|(b;e))|(-d))|(c;e))
number of applyed rules on step 8 is 1
step 9
number of applyed rules on step 9 is 0
out form is:
((((((a|(-b))|(-e))|d)|(-c))+
((((a|(*b))|(*e))|(*d))|c))+
(((((*a)|b)|d)|(*c))|(*e)))+
((((-a)|(b;e))|(-d))|(c;e))
```

Каноническая форма формулы $(a||d||\bar{b}||\bar{c}||\bar{e}) \vee (a||c||\delta_b||\delta_d||\delta_e) \vee$
$(b||\bar{d}||\delta_a||\delta_c||\delta_e) \vee ((b;e)||(c;e)||\bar{a}||\bar{d})$.

Литература

[1] C.Autant, Z.Belmesk, Ph.Schnoebelen: *Strong bisimularity on nets revisited.* Research Report 847-I, LIFIA-IMAG, Grenoble, France, 28 p., March 1991.

[2] C.Autant, Ph.Schnoebelen: *Place bisimulations in Petri nets.* LNCS 616, p.45–61, June 1992.

[3] J.Baeten, J.Bergstra, J.Klop: *An operational semanticsfor process algebra.* Report CS-R 8522, Centrum voor Wiskunde en Informatica, 1985.

[4] Luca Bernardinello, Fiorella de Cindio: *A survey of basic net models and modular net classes.* Design methods based on nets, Esprit basic research action 3148, June 1989–June 1990.

[5] Eike Best, Raymond Devillers, Astrid Kiehn, Lucia Pomello: *Concurrent bisimulations in Petri nets.* Springer Verlag, Acta Informatica 28, p.231–264, 1991.

[6] S.D.Brookes, C.A.R.Hoare, A.D.Roscoe: *A theory of communicating sequential processes.* Journal of ACM, Vol.31, No.3, p.560–599, 1984.

[7] J.Bergstra, G.Klop: *Process algebra for synchronous communication.* Information and Control 60, p.109–137, North Holland, Amsterdam, 1984.

[8] Ludmila Cherkasova: *Posets with non-actions: A model for concurrent nondeterministic processes.* Arbeitspapiere der GMD 403, 68p., July 1989.

[9] Ludmila Cherkasova: *Algebra AFP_2 for concurrent nondeterministic processes: Fully abstract model and complete axiomatization.* Reihe Informatic 72, 28p., July 1990.

[10] Ludmila Cherkasova: *A fully abstract model for concurrent nondeterministic processes based on posets with non-actions.* Computer Science, Department of Software Technology, Report CS-R 9031, Center for Mathematics and Computer Science, Amsterdam, Netherlands, 42p., July 1990.

[11] Ludmila Cherkasova, Vadim Kotov: *Concurrent nondeterministic processes: Adequacy of structure and behaviour.* LNCS 379, p.67–87, 1989.

[12] Ludmila Cherkasova, Vadim Kotov: *Descriptive and analitical process algebras.* LNCS 424, p.77–104, 1989.

[13] Raymond Devillers: *Maximality preserving bisimulation.* Theoretical Computer Science 102, p.165–184, 1992.

[14] Rob van Glabbeek, Ursula Goltz: *Equivalence notions for concurrent systems and refinement of actions.* LNCS 379, p.237–248, 1989.

[15] Rob van Glabbeek: *The refinement theorem for ST-bisimulation semantics.* In: M.Broy and C.B.Jones, editors. Proc. IFIP Working Conference of Programming Concepts and Methods, Sea of Galilee, Israel, 1990, to appear.

[16] Rob van Glabbeek, Frits Vaandrager: *Petri net models for algebraic theories of concurrency.* LNCS 259, p.224–242, 1987.

[17] V.E.Kotov, L.A.Cherkasova: *On structural properties of generalized processes.* LNCS 188, p.288–306, 1984.

[18] Vadim Kotov: *An algebra for parallelism based on Petri nets.* LNCS 64, p.39–55, 1978.

[19] R.Milner: *Algebras for communicating systems.* Report CSR-25-78/ Computer Science Department University Edinburgh, Edinburgh, 18 p., 1978.

[20] R.Milner: *A calculus of communicating systems.* LNCS 92, p.172–180, 1980.

[21] R.Milner: *Calculi for synchrony and asynchrony.* Theoretical Computer Science 25, p.267–310, 1983.

[22] M.Nielsen, P.S.Thiagarajan: *Degrees of non-determinizm and concurrency: A Petri net view.* LNCS 181, p.89–117, December 1984.

[23] D.Park: *Concurrency and automata on infinite sequences.* LNCS 104, p.167–183, March 1981.

[24] C.A.Petri: *Non-sequential processes.* GMD-ISF, Rep.77-05, 1977.

[25] A.Rabinovitch, B.A.Trakhtenbrot: *Behaviour structures and nets.* Fundamenta Informaticae XI, p.357–404, 1988.

[26] Walter Vogler: *Bisimulation and action refinement.* LNCS 480, p.309–321, 1991.

[27] Walter Vogler: *Modular construction and partial order semantics of Petri nets.* Springer Verlag, LNCS 625, 252p., 1992.