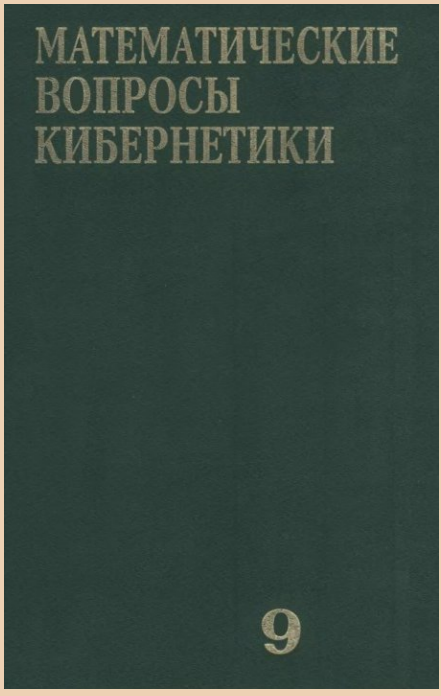
ИПМ им. М.В. Келдыша РАН • Электронная библиотека
Математические вопросы кибернетики • Выпуск 9



**В. А. Захаров,
И. В. Незнанов**

Операторные модели взаимодействующих процессов

Рекомендуемая форма библиографической ссылки: Захаров В. А., Незнанов И. В. Операторные модели взаимодействующих процессов // Математические вопросы кибернетики. Вып. 9. — М.: Физматлит, 2000. — С. 127–160. URL: http://library.keldysh.ru/mvk.asp?id=2000-127

ОПЕРАТОРНЫЕ МОДЕЛИ ВЗАИМОДЕЙСТВУЮЩИХ ПРОЦЕССОВ

В. А. ЗАХАРОВ, И. В. НЕЗНАНОВ

(МОСКВА)

§ 1. Введение

Цель данной работы — ввести понятие схемы программ для систем параллельно выполняющихся процессов на основе известных подходов к моделированию обычных последовательных программ схемами и исследовать некоторые их свойства.

Принимается следующая модель параллелизма: все процессы системы работают с разделенной памятью (память каждого процесса изолирована) и могут обмениваться сообщениями по асинхронным каналам емкости 1. Каждый такой канал в любой момент времени может содержать не более одного сообщения; если канал сообщений не содержит, то он свободен, если же канал содержит какое-либо сообщение, то он занят. Запись процессом сообщения в свободный канал проходит успешно, канал становится занятым; считывание процессом сообщения из занятого канала также проходит успешно, канал становится свободным. Запись процессом сообщения в занятый канал или считывание сообщения из пустого канала приводят к приостановке выполнения данного процесса до тех пор, пока какой-то другой процесс не освободит или, соответственно, не запишет данные в этот канал. Каждый канал соединяет два различных процесса, один может только писать в канал, другой — только читать, остальные процессы ни писать, ни читать из канала не могут.

Предложены два подхода к построению схем систем взаимодействующих процессов. Первый подход (интерпретационный) основан на стандартных схемах последовательных программ, описанных, например, в [2, 7]. В рамках этого подхода схемы строятся над множеством операторов присваивания, ветвления, посылки и приема сообщений. Каждый процесс имеет свою память (некоторое конечное множество переменных), выполнение схемы происходит при заданных интерпретации и начальном состоянии памяти. Интерпретация задает предметную область значений переменных, и семантику операторов присваивания и ветвления. При рассмотрении взаимодействующих процессов сообщениями, которыми обмениваются процессы, являются элементы из предметной области. Специфичные для систем взаимодействующих процессов операторы записи в канал имеют вид $\mathbf{send}(i, t)$ где i — номер канала, в который происходит запись, а t — некоторый терм. При выполнении такого оператора происходит запись значения терма t, вычисленного при заданной интерпретации и текущем состоянии памяти, в i-й канал. Операторы чтения из канала

имеют вид **receive**(i, x) где i — номер канала, из которого происходит чтение, а x — некоторая переменная, в которую помещается считанное из канала значение.

Второй подход (пропозициональный) развит из модели Янова эквивалентности схем программ [1, 3, 4, 5], в которой схемы программ строятся над алфавитами операторных символов и логических условий. В этом случае схемы систем взаимодействующих процессов строятся над алфавитами символов операторов преобразования данных (или просто операторных символов), логических условий, символов операторов приема и посылки сообщений. Выполнение схемы происходит на функции разметки, задающей способ означивания логических условий, т. е. указывающей способ выбора дальнейшей ветви вычисления при прохождении ветвлений схемы. Здесь выполнение операторов приводит к построению историй вычисления процессов — деревьев специального вида, отражающих «протокол» работы системы процессов, для которой была построена схема. Для каждого оператора приема и посылки сообщения известно, к какому каналу и процессу они относятся и, соответственно, с каким другим процессом связан этот канал. Это дает возможность наращивать историю вычисления процесса пошагово из уже имеющихся историй обоих процессов. Полностью процедура этого построения описана в § 3.

Оба предложенных подхода к моделированию параллельных программ схемами существенно отличаются от ранее известных направлений исследований в этой области [6, 9].

Процедуры построения вычислений схемы системы взаимодействующих процессов (ССВП) в обоих подходах имитируют выполнение соответствующей программной системы вычислительным устройством, работающим в режиме мультипрограммирования. В этом режиме все процессы выполняются одним центральным процессором и на каждом такте работы вычислительного устройства выполняется оператор лишь одного процесса, причем выбор процесса для выполнения среди активных на данный момент процессов недетерминирован. Это приводит к тому, что процедура построения вычисления также недетерминирована и результат вычисления может зависеть от способа выбора процесса для выполнения на каждом шаге. Одна из задач этой работы — доказать, что в данной модели параллелизма (при разделенной памяти процессов и взаимодействии их по синхронным каналам емкости 1) этот недетерминизм несуществен: если система завершает свою работу, то результат ее функционирования не зависит от выбора процесса для выполнения на каждом шаге. Обоснование этого факта позволяет корректно определить понятие завершения схемы и результата ее выполнения независимо от выбора активного процесса на каждом шаге вычисления и, тем самым, корректно ввести понятие эквивалентности схем.

На основе результатов вычисления ССВП в каждом из подходов можно сравнивать различные схемы на эквивалентность. При интерпретационном подходе две схемы эквивалентны, если при любой интерпретации I и любом начальном состоянии памяти W_0 либо обе они завершают вычисления с одинаковым результатом — набором значений выходных переменных, либо вычисления обеих схем никогда не завершаются. При пропозициональном подходе две схемы эквивалентны, если на любой функции разметки они одновременно либо завершают, либо не завершают свое выполнение и если завершают, то результаты их — наборы деревьев-историй всех процессов схемы — эквивалентны. Другая задача данной работы — исследовать взаимосвязь между предложенными подходами к построению схем систем взаимодействующих процессов.

Доказано, что из пропозициональной эквивалентности двух схем следует их интерпретационная эквивалентность.

§ 2. Интерпретационная модель систем взаимодействующих процессов

2.1. Базовые конструкции. Схемы систем взаимодействующих процессов (ССВП) в интерпретационной модели строятся над базисом $(X, F, P, Spec)$, включающим в себя множество $X = \{x_1, \ldots, x_n\}$ переменных, которое будем называть памятью процессов, множество $F = \{f_1, \ldots, f_n\}$ функциональных символов и множество $P = \{p_1, \ldots, p_n\}$ предикатных символов, а также множество специальных символов

$$Spec = \{\textbf{start}, \textbf{stop}, (,), :=, \textbf{send}, \textbf{receive}\}.$$

Каждому функциональному символу f_k сопоставлена его местность (арность) i_k, каждому предикатному символу p_k сопоставлена его местность j_k. Базис $(X, F, P, Spec)$ будем обозначать B.

Будем использовать обычные индуктивные определения интерпретации базиса и терма в сигнатуре базиса. Множество всех термов обозначим через **Term**. *Оценкой*, или *состоянием памяти процессов* назовем отображение $W\colon X \to D$, где D — предметная область интерпретации.

Стандартным образом индуктивно определяются понятия атомарной формулы, или атома, в сигнатуре базиса, а также значений терма и атома при фиксированной оценке (состоянии памяти процессов) и интерпретации.

2.2. Схемы систем взаимодействующих процессов. Рассмотрим систему из k взаимодействующих процессов, которые обмениваются сообщениями по l асинхронным каналам емкости 1. *Схемой* такой *системы* назовем ориентированный граф G, состоящий из k компонент связности $\Gamma_1, \ldots, \Gamma_k$. Каждый из подграфов Γ_i называется *процессом* и обладает следующими свойствами (см. рис. 1):

1) обязательно содержит единственную вершину, помеченную символом **start**, в такую вершину не входят дуги и выходит единственная дуга;

2) обязательно содержит хотя бы одну вершину, помеченную выражением $\textbf{stop}(x_{i_1}, \ldots, x_{i_{m_i}})$ (которое назовем *оператором завершения*), где $x_{i_1}, \ldots, x_{i_{m_i}} \in X$, из такой вершины ни одной дуги не выходит;

3) может содержать вершину, помеченную выражением $x := t$ (которое назовем *оператором присваивания*), где $x \in X$ и $t \in \textbf{Term}$, из нее выходит единственная дуга;

4) может содержать вершину, помеченную атомарной формулой $p^{(n)}(t_1, \ldots, t_n)$, из такой вершины выходят две дуги, одна из которых помечена символом 0, другая — символом 1;

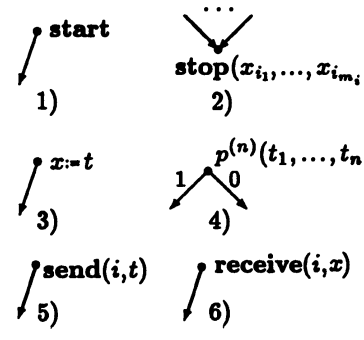


Рис. 1. Типы вершин стандартных ССВП

5) может содержать вершину, помеченную выражением $\textbf{send}(i, t)$ (которое назовем *оператором отправления*), где $t \in \textbf{Term}$ и $i \in \{1, \ldots, l\}$, из нее выходит единственная дуга;

6) может содержать вершину, помеченную выражением $\textbf{receive}(i, x)$ (которое назовем *оператором приема*), где $x \in X$, $i \in \{1, \ldots, l\}$, из нее также выходит единственная дуга.

При этом если не оговорено особо, то считается, что в вершину может входить произвольное число дуг. Вершину типа 1) будем называть *начальной*, типа 2) — *заключительной*, типа 3) — *вершиной-преобразователем*, типа 4) — *вершиной-распознавателем*, типа 5) — *вершиной-источником*, типа 6) — *вершиной-приемником*.

Пусть v — рассматриваемая вершина. В случаях 1), 3), 5), 6) из v выходит единственная дуга, ведущая в некоторую вершину w, которую мы будем обозначать через $\mathbf{succ}(v)$; в случае 4) вершину, достижимую из v по дуге с пометкой 0, будем обозначать $\mathbf{succ}_0(v)$, а вершину, достижимую из v по дуге с пометкой 1 — через $\mathbf{succ}_1(v)$; в случае 2) положим $\mathbf{succ}(v)=v$.

Обозначим через $\mathbf{Op}(G)$ множество всех операторов преобразования, отправления и приема сообщений, встречающихся в схеме, а через $\mathbf{Op}(\Gamma_i)$ — множество операторов преобразования, отправления и приема сообщений, встречающихся в i-м процессе, $\mathbf{Op}(G)=\bigcup \mathbf{Op}(G_i)$.

На процессы $\Gamma_1, \ldots, \Gamma_k$ наложим некоторые ограничения.

Введем множества $\mathbf{Term}_i = \{t \in \mathbf{Term} \mid$ в Γ_i содержится вершина одного из типов: вершина-преобразователь, помеченная оператором $x := t$, вершина-источник, помеченная словом $\mathbf{send}(i, t)$, вершина-распознаватель, помеченная атомарной формулой $p^{(n)}(t_1, \ldots, t_n)$, причем $t \in \{t_1, \ldots, t_n\}\}$.

Тогда подграфы $\Gamma_1, \ldots, \Gamma_k$ должны быть таковы, что при некотором разбиении множества переменных $X = \bigcup_{i=1}^{k} X_i$ на попарно непересекающиеся подмножества X_i выполнены следующие условия:

— любой терм t из $\mathbf{Term}_i$ содержит вхождения только переменных из X_i;

— если Γ_i содержит вершину второго, третьего или шестого типов, то все переменные, встречающиеся в этих пометках, принадлежат X_i;

— если процесс Γ_i содержит источник с оператором $\mathbf{send}(m, t)$ (приемник с оператором $\mathbf{receive}(m, x)$), а процесс Γ_j содержит источник с оператором $\mathbf{send}(n, t')$ (приемник с оператором $\mathbf{receive}(n, x')$), и $i \neq j$, то $m \neq n$;

— если в Γ_i есть вершина с оператором $\mathbf{send}(m, t)$, а в Γ_j есть вершина с оператором $\mathbf{receive}(m, x')$, то $i \neq j$.

Определенную таким образом схему G будем называть *интерпретационной или стандартной схемой систем взаимодействующих процессов* (ССВП).

Для заданной ССВП $G = (\Gamma_1, \ldots, \Gamma_k)$ каждый *канал* связи процессов Ch_i представляется парой $\langle S_i, R_i \rangle$, где $S_i = \{$множество всех источников G с операторами отправления вида $\mathbf{send}(i, t)\}$, $R_i = \{$множество всех приемников G с операторами приема вида $\mathbf{receive}(i, x)\}$. Налагая перечисленные выше ограничения на структуру подграфов $\Gamma_1, \ldots, \Gamma_k$, мы подразумеваем, что

— все процессы имеют разделенную память;

— операторы, непосредственно изменяющие состояние памяти (присваивания и чтения из канала) не могут изменять значения переменных, принадлежащих другим процессам;

— все термы, встречающиеся в операторах и распознавателях процесса, построены над переменными только из этого процесса;

— каналы допускают передачу только в одном направлении;

— в каждый канал запись может осуществлять только один процесс, и считывание из каждого канала может осуществлять только один процесс;

— каждый канал соединяет два различных процесса.

Если некоторая заключительная вершина имеет список переменных $x_{i_1}, \ldots, x_{i_s}$ и достигается в описанном ниже процессе построения вычисления ССВП, то значения этих переменных на момент достижения данной вершины являются результатом работы i-го процесса.

2.3. Состояния стандартной ССВП. Пусть дана схема G системы k взаимодействующих по l каналам процессов, $G = (\Gamma_1, \ldots, \Gamma_k)$. Занумеруем вершины G: в каждом подграфе Γ_i нумеруем вершины последовательными натуральными числами так, чтобы начальная вершина имела номер 1,

а в остальном — произвольно. Пусть вершины каждого процесса Γ_i занумерованы числами $1, \ldots, n_i$. Фиксируем интерпретацию базиса B (введя тем самым в рассмотрение область интерпретации D).

Определение 1. *Состояние памяти процессов* G — это произвольное отображение $W: X \to D$. Множество $\{W: X \to D\}$ всех состояний памяти обозначим *Val*. Для каждого процесса Γ_i и состояния W памяти схемы обозначим через $W|_i$ сужение W на множество переменных X_i данного процесса.

Определение 2. Вектор $r = (r_1, \ldots, r_k)$, где $r_i \in \{1, \ldots, n_i\}$ для всех $i \in \{1, \ldots, k\}$, назовем *состоянием управления*.

Определение 3. Пусть $W \in Val$, $r = (r_1, \ldots, r_k)$ — состояние управления. Тогда *состоянием процесса* Γ_i назовем пару $(r_i, W|_i)$. *Состоянием всех процессов* назовем пару (r, W).

Для построения вычисления схемы еще потребуются следующие объекты:

— *вектор индикаторов занятости каналов* $z = (z_1, \ldots, z_l)$, где z_i принимает значение 0 или 1 для каждого i, $1 \leqslant i \leqslant l$;

— дополнительное множество переменных $M = \{m_1, \ldots, m_l\}$, отличных от переменных из X, которые мы будем называть *памятью каналов*.

Аналогично состоянию памяти процессов введем *состояние памяти каналов* как отображение $U: M \to D$. Множество $\{f: M \to D\}$ всех состояний памяти каналов обозначим *ChVal*.

Введение дополнительного множества переменных M связано с тем, что ССВП использует для хранения данных как внутренние переменные процессов, так и каналы. Посылка сообщения в канал приводит к записи в него некоторого значения, и для введения конфигурации, или состояния ССВП как объекта, полностью описывающего состояние схемы на отдельном шаге функционирования требуется учитывать значения, записанные в каналы схемы.

Определение 4. *Состоянием каналов* назовем пару (z, U), где $U \in ChVal$, а z — вектор индикаторов занятости каналов.

Содержательно $z_i = 1$ будет обозначать факт занятости i-го канала (т. е. ситуацию, в которой выполнился оператор **send**(i, t), но еще не сработал оператор **receive**(i, x)), соответственно, $z_i = 0$ — факт его свободы. Если $z_i = 1$, то $U(m_i)$ будет обозначать конкретное значение из D, записанное в i-й канал. Если $z_i = 0$, то $U(m_i)$ не определено.

Процесс выполнения ССВП заключается в последовательном обходе по шагам вершин каждого процесса, начиная с начальной. Шаг выполнения состоит в том, что процесс выполняет действие, приписанное достигнутой на данный момент вершине, и переходит в некоторую следующую вершину. Если при обходе вершин некоторого процесса достигнута заключительная вершина, то процесс считается завершившим свое выполнение, иначе — незавершившимся. В процессе работы ССВП на каждом шаге незавершившийся процесс либо может беспрепятственно выполнить свое очередное действие, либо должен быть приостановлен, если он ожидает данные из канала, который на этом шаге свободен или передает данные в канал, который в данное время занят; его выполнение может быть возобновлено, если другой процесс, соответственно, записал данные в канал, либо освободил его. В первом случае скажем, что процесс активен, во втором — приостановлен. Введем в рассмотрение множества *Act* и *Stop*, содержащие в качестве своих элементов номера процессов ССВП. Множество *Act* будет содержать номера активных на данном шаге процессов, а множество *Stop* — номера процессов, которые к данному моменту уже завершили свое выполнение. Формально процедура изменения множеств *Act* и *Stop* и их начальные значения будут описаны ниже в алгоритме построения вычисления схемы.

Определение 5. *Состоянием* ССВП G назовем набор $(r, W, z, U, Act, Stop)$, где $W \in Val$ и $U \in ChVal$, r — вектор управления, z — вектор занятости каналов, множества Act и $Stop$ — подмножества множества $\{1, \ldots, k\}$.

2.4. Вычисления ССВП. Рассмотрим ССВП $G = (\Gamma_1, \ldots, \Gamma_k)$, образованную k процессами, взаимодействующими по l каналам связи. Пусть задана интерпретация I базиса B и выделено некоторое состояние W_0 памяти ССВП, которое мы назовем *начальным состоянием памяти.* Для заданной пары (I, W_0) опишем процедуру построения вычисления стандартной ССВП G.

Процедура построения вычисления ССВП G инициирует последовательный обход по шагам вершин графов процессов, причем обход вершин каждого процесса начинается с начальной вершины процесса (которая имеет номер 1). Поэтому для каждого процесса можно говорить о текущей вершине как о последней вершине графа процесса, достигнутой на данном шаге при обходе вершин ССВП.

Начальным состоянием схемы G назовем состояние

$$A_0 = (r, W_0, z, U, Act, Stop),$$

где $r = (1, \ldots, 1)$ (текущими вершинами каждого процесса являются начальные вершины), $z = (0, \ldots, 0)$ (в начальном состоянии системы все каналы свободны для записи), $U \in ChVal$ (начальным состоянием памяти каналов является произвольное состояние), $Act = \{1, \ldots, k\}$ и $Stop = \varnothing$ (все процессы активны и ни один из них не завершился).

На множестве состояний ССВП G введем отношение *непосредственного преобразования схемой G состояний ССВП G.*

Пусть $A = (r, W, z, U, Act, Stop)$ — некоторое состояние стандартной ССВП $G = (\Gamma_1, \ldots, \Gamma_k)$, где $r = (r_1, \ldots, r_k)$ — состояние управления, W — состояние памяти процессов, $z = (z_1, \ldots, z_l)$ — вектор индикаторов занятости каналов, $U = (u_1, \ldots, u_l)$ — вектор памяти каналов. Рассмотрим произвольное $j \in Act$ и процесс Γ_j. Выделенный процесс Γ_j назовем *активизированным.* Преобразование состояния системы осуществляется в два этапа.

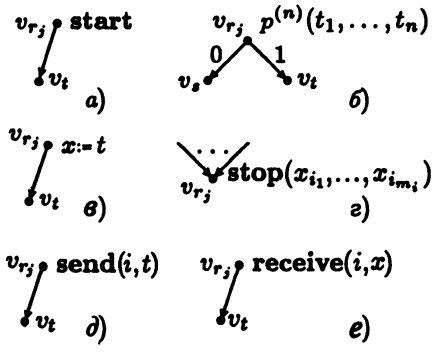


Рис. 2. Построение вычисления

На первом этапе в зависимости от типа текущей вершины v_{r_j} в Γ_j сформируем новое состояние A' системы G, руководствуясь следующими правилами (см. рис. 2):

1) v_{r_j} — начальная и $\mathbf{succ}(v_{r_j}) = v_t$, тогда $A' = (r', W, z, U, Act, Stop)$, где (см. рис. 2,*а*)

$$r'_i = \begin{cases} r_i, & i \neq j, \\ t, & i = j; \end{cases}$$

2) v_{r_j} — вершина-распознаватель с приписанной ей атомарной формулой $p(t_1, \ldots, t_n)$, при этом $\mathbf{succ}_0(v_{r_j}) = v_t$ и $\mathbf{succ}_1(v_{r_j}) = v_s$, тогда $A' = (r', W, z, U, Act, Stop)$, где (см. рис. 2,*б*)

$$r'_i = \begin{cases} r_i, & i \neq j, \\ s, & i = j \text{ и } p_I(W) = true, \\ t, & i = j \text{ и } p_I(W) = false \end{cases}$$

(здесь $p_I(W)$ — значение атомарной формулы p в интерпретации I на состоянии памяти W);

3) v_{r_j} — вершина-преобразователь, помеченная оператором $x := t$ и $\mathbf{succ}(v_{r_j}) = v_t$, тогда $A' = (r', W', z, U, Act, Stop)$, где (см. рис. 2,*в*)

$$W'(y) = \begin{cases} W(y), & y \in X \setminus \{x\}, \\ t_I(W), & y = x, \end{cases} \qquad r'_i = \begin{cases} r_i, & i \neq j, \\ t, & i = j; \end{cases}$$

4) v_{r_j} — заключительная вершина, тогда $A' = (r, W, z, U, Act', Stop')$, где (см. рис. 2,*г*) $Act' = Act \setminus \{j\}$ и $Stop' = Stop \cup \{j\}$ (здесь вычисляются значения переменных из списка аргументов оператора **stop** и этот список значений объявляется результатом вычисления j-го процесса);

5) v_{r_j} — вершина-источник с оператором $\mathbf{send}(i, t)$ и $\mathbf{succ}(v_{r_j}) = v_t$, тогда $A' = (r', W, z', U', Act', Stop)$, где (см. рис. 2,*д*)

если $z_i = 0$, то $Act' = Act$,

$$u'_m = \begin{cases} u_m, & m \neq i, \\ t_I(W), & m = i, \end{cases} \qquad z'_m = \begin{cases} z_m, & m \neq i, \\ 1, & m = i, \end{cases} \qquad r'_m = \begin{cases} r_m, & m \neq j, \\ t, & m = j, \end{cases}$$

а если $z_i = 1$, то $r' = r$, $z' = z$, $U' = U$ и $Act' = Act \setminus \{j\}$;

6) v_{r_j} — вершина-приемник с оператором $\mathbf{receive}(i, x)$ и $\mathbf{succ}(v_{r_j}) = v_t$, тогда $A' = (r', W, z', U', Act', Stop)$, где (см. рис. 2,*е*)

если $z_i = 1$, то $Act' = Act$,

$$W'(y) = \begin{cases} W(y), & y \in X \setminus \{x\}, \\ u_i, & y = x, \end{cases} \qquad z'_m = \begin{cases} z_m, & m \neq i, \\ 0, & m = i, \end{cases} \qquad r'_m = \begin{cases} r_m, & m \neq j, \\ t, & m = j, \end{cases}$$

а если $z_i = 0$, то $r' = r$, $z' = z$, $W' = W$ и $Act' = Act \setminus \{j\}$.

На втором этапе выполняется процедура послешаговой коррекции состояний активности процессов.

Пусть $A' = (r', W', z', U', Act', Stop')$ — состояние системы, образованное по окончании первого этапа. Рассмотрим все номера заблокированных процессов $j \in \{1, \ldots, k\} \setminus (Act' \cup Stop')$. Тогда

а) если $v_{r'_j}$ — вершина-источник с пометкой $\mathbf{send}(i, t)$ и $z'_i = 0$, то модифицируем A', полагая $A' = (r', W', z', U', Act' \cup \{j\}, Stop')$;

б) если $v_{r'_j}$ — вершина-приемник с пометкой $\mathbf{receive}(x, i)$ и $z'_i = 1$, то модифицируем A', полагая $A' = (r', W', z', U', Act' \cup \{j\}, Stop')$.

Полученное по окончании второго этапа модифицированное состояние A' считается *результатом непосредственного преобразования состояния A схемой G*, а само отношение непосредственного преобразования обозначается при этом

$$A \xrightarrow{j} A'.$$

Отметим, что в случае $Act = \varnothing$ состояние A не допускает ни одного непосредственного преобразования. Если при этом $Stop = \{1, \ldots, k\}$, то такое состояние назовем *заключительным состоянием*. В том случае если $Act = \varnothing$ и $j \notin Stop$ для некоторого процесса Γ_j, состояние A называется *тупиковым состоянием*.

Определение 6. *Вычислением стандартной ССВП G* назовем такую последовательность состояний $A_0, A_1, \ldots$ (возможно, бесконечную), что A_0 — начальное состояние схемы, а для каждого члена A_i, $i > 0$, этой последовательности выполняется отношение

$$A_{i-1} \xrightarrow{n_i} A_i.$$

При этом последовательность $n_1 n_2 \ldots$ номеров процессов, активизированных на каждом шаге вычисления, будем называть *планом данного вычисления.*

Фактически, план — это стратегия выбора процесса из числа активных для исполнения на каждом шаге вычисления схемы. Само вычисление представляет собой последовательный обход графов процессов заданной ССВП, сопровождаемый изменением текущего состояния системы. Последовательность вершин, по которым совершается подобный обход в процессе Γ_i по плану H для начального состояния W_0 в интерпретации I назовем *путем вычисления процесса* Γ_i *ССВП* G *по плану* H и обозначим $\pi(G, \Gamma_i, H)$ или $\pi_i(G, H)$ или $\pi_i(H)$, если из контекста ясно, о какой ССВП идет речь. Совокупность путей вычисления $(\pi_1(H), \ldots, \pi_k(H))$ всех процессов G назовем *путем вычисления* G *по плану* H. Поскольку вычисление схемы на заданном начальном состоянии однозначно определяется соответствующим планом, в дальнейшем мы будем говорить, что вычисление схемы G

$$A_0 \xrightarrow{n_1} A_1 \xrightarrow{n_2} A_2 \xrightarrow{n_3} \ldots$$

выполняется по плану $H = n_1 n_2 n_3 \ldots$

Определение 7. Пустым планом выполнения схемы G назовем план $H = e$, где e — пустая цепочка символов в алфавите $\{1, \ldots, k\}$.

Пустой план схемы G порождает вычисление, состоящее из единственной начальной конфигурации A_0.

Определение 8. План H выполнения схемы $G = (\Gamma_1, \ldots, \Gamma_k)$ (или план схемы G) называется *завершающим* (*тупиковым*), если H — конечная последовательность.

Вычисление схемы, выполняющейся по плану H, завершается заключительным (тупиковым) состоянием.

В заключительном состоянии определены результаты вычисления всех процессов системы. Список этих результатов будем называть *результатом вычисления стандартной ССВП* G *на плане* H. Если план вычисления схемы является тупиковым, или вычисление схемы на плане H оказывается бесконечным, то мы будем говорить, что *результат такого вычисления не определен.*

Это означает, что процедура построения вычисления на завершающем плане доводит вычисление всех процессов до заключительных вершин, а процедура построения вычисления на тупиковом плане приходит к состоянию, в котором все незавершившиеся процессы не активны (тупиковая ситуация).

Будем говорить, что конечный план H_1 схемы G является *префиксом* плана H_2 схемы G, если $H_1 = n_1 \ldots n_q$, $H_2 = m_1 \ldots m_q m_{q+1} \ldots$ и $n_i = m_i$ для всех $i \in \{1, \ldots, q\}$.

Замечание. Если H — план выполнения схемы G, то любой его префикс — план выполнения схемы G.

Замечание. Никакой план не может иметь строгого префикса, являющегося тупиковым планом.

Следует заметить также, что для одного и того же начального состояния ССВП различные планы порождают различные вычисления, которые могут оказаться завершающими, бесконечными или тупиковыми. Поэтому для введения понятий результата выполнения схемы на заданных интерпретации и начальном состоянии памяти и эквивалентности схем необходимо предварительно убедиться в том, что результаты вычислений схемы при заданном начальном состоянии памяти инвариантны относительно планов этих вычислений.

§ 3. Пропозициональная модель систем взаимодействующих процессов

3.1. Пропозициональные ССВП. В данной модели схемы строятся над алфавитами $A = \bigcup\limits_{i=1}^{k} A_i$ операторных символов, алфавитом $P = \bigcup\limits_{i=1}^{k} P_i$ символов логических условий, алфавитом $S = \bigcup\limits_{j=1}^{l} S_j$ символов операторов записи в каналы и алфавитом $R = \bigcup\limits_{j=1}^{l} R_j$ символов операторов чтения из каналов. При этом разбиения $A = \bigcup A_i$, $P = \bigcup P_i$, $S = \bigcup S_j$, $R = \bigcup R_j$ образованы попарно непересекающимися множествами символов.

Рассмотрим *систему* из k процессов, взаимодействующих по l каналам. Тогда *схемой* такой системы называется ориентированный граф G, состоящий из k компонент связности $\Gamma_1, \ldots, \Gamma_k$. Каждый подграф Γ_i называется *процессом* и имеет следующую структуру (см. рис. 3):

1) обязательно содержит единственную вершину, в которую не входят дуги и выходит единственная дуга;

2) обязательно содержит хотя бы одну вершину, из которой ни одной дуги не выходит;

3) может содержать вершины, помеченные символом a из A, из каждой такой вершины выходит единственная дуга;

4) может содержать вершины, которым приписано логическое условие p из P, из таких вершин выходят две дуги, одна из которых помечена символом 0, другая — символом 1;

5) может содержать вершины, помеченные символом s из S, из которых выходит ровно одна дуга;

6) может содержать вершины, помеченные символом r из R, из которых также выходит ровно одна дуга.

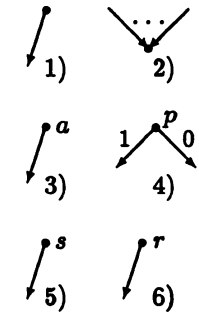


Рис. 3. Типы вершин пропозициональных ССВП

Если не оговорено особо, то в вершину может входить произвольное число дуг. Вершину типа 1) будем называть *начальной*, типа 2) — *заключительной*, типа 3) — *вершиной-преобразователем*, типа 4) — *вершиной-распознавателем*, типа 5) — *вершиной-источником*, типа 6) — *вершиной-приемником*.

Обозначим через $\mathbf{Op}(G) = A \cup S \cup R$ множество символов операторов, встречающихся в схеме G, через $\mathbf{Op}(\Gamma_i)$ — множество символов операторов, встречающихся в i-м процессе.

Как и для стандартных ССВП, в случаях 1), 3), 5), 6) единственную вершину, достижимую из рассматриваемой вершины v, обозначим через $\mathbf{succ}(v)$; в случае 4) вершину, достижимую из v по дуге с пометкой 0 (соответственно, 1) обозначим через $\mathbf{succ}_0(v)$ (через $\mathbf{succ}_1(v)$); в случае 2) положим $\mathbf{succ}(v) = v$.

При этом разметка вершин графов Γ_i должна удовлетворять следующим условиям:

— для любого $i \in \{1, \ldots, k\}$ преобразователи графа Γ_i помечены только символами из множества A_i, распознаватели графа Γ_i — только символами из P_i;

— для каждого $i \in \{1, \ldots, l\}$ существует не более одного подграфа Γ_j, содержащего источники, помеченные символами из S_i;

— для каждого $i \in \{1, \ldots, l\}$ существует не более одного подграфа Γ_j, содержащего приемники, помеченные символами из R_i;

— если подграф Γ_i содержит источники, помеченные символами из S_m, а подграф Γ_j содержит приемники, помеченные символами из R_m, то $i \neq j$.

Каждая пара множеств (S_j, R_j) обозначает все операторы записи и чтения для j-го канала. Указанные выше требования аналогичны требованиям к стандартным схемам взаимодействующих процессов.

Поскольку все источники (и приемники) для каждого канала принадлежат лишь одному процессу, для символов из алфавитов S и R примем обозначения s^i_{jk} и r^i_{jk} соответственно, где i — номер процесса, которому принадлежат вершины, помеченные этим символом, j — номер канала, запись в который обозначает этот символ, k — порядковый номер данного символа среди всех символов записи в j-й канал. Аналогичен смысл индексов и для символов r^i_{jk}. Операторные символы из A будем обозначать a^i_k, где i обозначает номер процесса, преобразователи которого помечены этим символом, k будет обозначать порядковый номер этого символа среди всех символов A_i. Логические условия будем обозначать p^i_k, где смысл индексов тот же что и для A.

3.2. Деревья вычислений.

Определение 9. *Корневым реберно-помеченным деревом* назовем четверку (V, G, v, f), где (V, G) — дерево, $v \in V$ — выделенная вершина, называемая *корнем*, f — *функция разметки*, помечающая каждое ребро символом из $A \cup S \cup R$.

В дальнейшем потребуется рассматривать не произвольные реберно-помеченные деревья, а их подкласс — те деревья, которые являются историями вычислений некоторых процессов.

Определение 10. *Дерево вычисления системы из k взаимодействующих по l каналам процессов* — это корневое реберно-помеченное дерево, определяемое индуктивно (см. рис. 4) следующим образом.

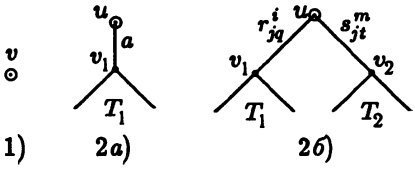


Рис. 4. Деревья вычислений

1. $T = (\{v\}, G, v, f)$, где $G = \varnothing$ и f — вырожденная (определенная на пустом множестве) функция пометки ребер, есть *дерево вычисления*; это вырожденное дерево без ребер, состоящее из одной вершины, являющейся корнем. Назовем такое дерево *начальным деревом*.

2. Пусть $T_1 = (V_1, G_1, v_1, f_1)$ и $T_2 = (V_2, G_2, v_2, f_2)$ — деревья вычислений. Тогда *деревом вычисления* является также и любое дерево T вида а) или б):

а) $T = (V_1 \cup \{u\}, G_1 \cup \{(u, v_1)\}, u, f)$, где u — новая вершина, являющаяся корнем T; при этом $f(e) = f_1(e)$ для любого ребра e из G_1, и $f((u, v_1)) = a$ для ребра (u, v_1), где a — символ из $A \cup S$ (в этом случае дерево вычисления T будем обозначать $T = T_1 * a$);

б) $T = (V_1 \cup V_2 \cup \{u\}, G_1 \cup G_2 \cup \{(u, v_1), (u, v_2)\}, u, f)$, где u — новая вершина — корень T и $f(e) = f_1(e)$ для любого e из G_1, $f(e) = f_2(e)$ для любого e из G_2, $f((u, v_1)) = r^i_{jq}$ для ребра (u, v_1), $f((u, v_2)) = s^m_{jt}$ для ребра (u, v_2), где s^m_{jt} и r^i_{jq} — символы операторов чтения и записи, относящиеся к одному и тому же каналу, но к различным процессам, $i \neq m$ (в этом случае дерево вычисления T будем обозначать $T = T_1 * r^i_{jq} + T_2 * s^m_{jt}$).

Замечание. На самом деле класс деревьев, являющихся историями вычисления процессов, уже класса всех деревьев вычислений. Из изложенного ниже алгоритма построения истории вычисления схемы будет следовать, что получающиеся в нем деревья будут деревьями вычислений в смысле определения 10.

3.3. Функции разметки. Пусть $Tr = \{T\}$ — множество всех реберно-помеченных деревьев. Пусть $(T_1, \ldots, T_k)$ — набор из k корневых реберно-помеченных деревьев и $Tr^k = \{(T_1, \ldots, T_k)\}$ — множество всех таких наборов.

Для построения вычисления схемы потребуется понятие функции разметки.

Определение 11. *Функцией разметки* назовем функцию

$$\mu\colon\ Tr^k \times P \to \{true, false\},$$

которая каждому набору деревьев из Tr и каждому логическому условию p из P ставит в соответствие значение истинности *true* или *false*, и при этом удовлетворяющую следующему условию:

— если есть два набора $(T_1, \ldots, T_i, \ldots, T_k)$ и $(T_1, \ldots, T_i', \ldots, T_k)$, различающиеся только в i-й компоненте, и T_i' получено из T_i с помощью правила $2a$) из определения дерева вычисления, то для любого логического условия $p_q^j \in P \setminus P_i$ выполнено

$$\mu((T_1, \ldots, T_i, \ldots, T_k), p_q^j) = \mu((T_1, \ldots, T_i', \ldots, T_k), p_q^j);$$

— если T_i' получено из T_i и T_q с помощью правила $2б$, так что T_i — левое поддерево T_i', то для любого логического условия $p_q^j \in P \setminus P_i$ выполнено

$$\mu((T_1, \ldots, T_i, \ldots, T_k), p_q^j) = \mu((T_1, \ldots, T_i', \ldots, T_k), p_q^j).$$

Условие, налагаемое на функции разметки, отражает тот факт, что в принятой модели параллельных вычислений логические условия каждого процесса могут быть изменены только операторами этого процесса, но не других.

Таким образом, функцию разметки μ можно представить в виде

$$\mu = \mu_1 \times \mu_2 \times \ldots \times \mu_k,$$

где $\mu_i\colon\ Tr \times P_i \to \{true, false\}$ — функции, задающие значения логических условий из P_i в зависимости от истории выполнения i-го процесса, а произведение $\mu = \mu_1 \times \ldots \times \mu_k$ задается таким образом:

$$\mu((T_1, \ldots, T_k), p) = \mu_i(T_i, p)$$

для всякого $p \in P_i$, $1 \leqslant i \leqslant k$.

3.4. Состояния пропозициональной ССВП. Рассмотрим схему $G = (\Gamma_1, \ldots, \Gamma_k)$. В каждом подграфе Γ_i занумеруем вершины последовательными натуральными числами так, чтобы начальная вершина получила номер 1, а в остальном — произвольно. Пусть для любого i вершины Γ_i занумерованы числами $1, \ldots, n_i$.

Определение 12. Вектор $r = (r_1, \ldots, r_k)$, где $r_i \in \{1, \ldots, n_i\}$, назовем *состоянием управления*.

Как и для стандартных ССВП, i-я компонента вектора управления обозначает номер текущей вершины процесса Γ_i на данном шаге вычисления.

Аналогично случаю стандартных ССВП, введем вектор занятости каналов $z = (z_1, \ldots, z_l)$, $z_i \in \{0, 1\}$ для любого i, $1 \leqslant i \leqslant l$. Как и раньше, $z_i = 1$ означает, что i-й канал занят, а $z_i = 0$ — что i-й канал свободен.

Определение 13. *Состоянием процессов* назовем пару (r, D), где r — состояние управления, $D = \{D_1, \ldots, D_k\}$ — набор из k бинарных реберно-помеченных деревьев. Набор D будем называть *состоянием памяти процессов*.

Определение 14. *Состоянием каналов* назовем пару (z, C), где z — состояние занятости каналов, $C = (C_1, \ldots, C_l)$ — набор из l бинарных реберно-помеченных деревьев.

Введем также множества *Act* и *Stop* — подмножества $\{1, \ldots, k\}$. Их смысл аналогичен смыслу множеств *Act* и *Stop* для стандартных схем.

Определение 15. *Состоянием системы* будем называть набор $B = (r, D, z, C, Act, Stop)$, где (r, D) — состояние процессов, (z, C) — состояние каналов.

Состояние $B_0 = (r, D_0, z, C, Act, Stop)$ назовем *начальным состоянием системы*, если $r = (1, \ldots, 1)$ (текущими вершинами каждого процесса являются начальные вершины), $D_0 = (V_0, \ldots, V_0)$, где V_0 — начальное дерево, $z = (0, \ldots, 0)$ (в данном состоянии системы все каналы свободны для записи), $C = (C_1, \ldots, C_l)$ — произвольный набор из l бинарных реберно-помеченных деревьев — произвольное состояние памяти каналов, $Act = \{1, \ldots, k\}$, $Stop = \varnothing$ (т. е. все процессы активны и ни один не завершился).

На множестве состояний ССВП G введем отношение *непосредственного преобразования состояний схемой* G по функции разметки μ.

Пусть $B = (r, D, z, C, Act, Stop)$ — некоторое состояние пропозициональной ССВП $G = (\Gamma_1, \ldots, \Gamma_k)$, где $r = (r_1, \ldots, r_k)$ — состояние управления, $z = (z_1, \ldots, z_l)$ — вектор индикаторов занятости каналов, $C = (C_1, \ldots, C_l)$ — вектор памяти каналов. Рассмотрим произвольное $j \in Act$ и процесс Γ_j. Выделенный процесс Γ_j назовем *активизированным*. Преобразование состояния системы осуществляется в два этапа.

На первом этапе в зависимости от типа текущей вершины v_{r_j} в Γ_j сформируем новое состояние B' системы G, руководствуясь следующими правилами (рис. 5):

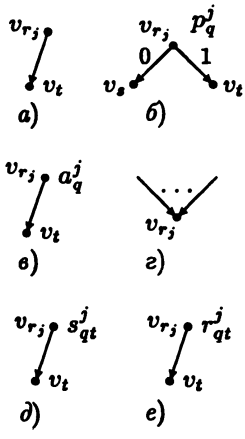


Рис. 5. Построение вычисления

1) v_{r_j} — начальная и $\mathbf{succ}(v_{r_j}) = v_t$, тогда $B' = (r', D, z, C, Act, Stop)$, где (рис. 5,*а*)

$$r_i' = \begin{cases} r_i, & i \neq j, \\ t, & i = j; \end{cases}$$

2) v_{r_j} — вершина-распознаватель с приписанным ей логическим условием p_q^j, при этом $\mathbf{succ}_0(v_{r_j}) = v_t$ и $\mathbf{succ}_1(v_{r_j}) = v_s$, тогда $B' = (r', D, z, C, Act, Stop)$, где (рис. 5,*б*)

$$r_i' = \begin{cases} r_i, & i \neq j, \\ s, & i = j \text{ и } \mu(D)(p_q^j) = true, \\ t, & i = j \text{ и } \mu(D)(p_q^j) = false \end{cases}$$

(здесь $\mu(D)$ — значение функции разметки на текущем состоянии памяти процессов D);

3) v_{r_j} — вершина-преобразователь, помеченная операторным символом a_q^j и $\mathbf{succ}(v_{r_j}) = v_t$, тогда $B' = (r', D', z, C, Act, Stop)$, где (рис. 5,*в*)

$$D_i' = \begin{cases} D_i, & i \neq j, \\ D_j * a_q^j, & i = j, \end{cases} \qquad r_i' = \begin{cases} r_i, & i \neq j, \\ t, & i = j, \end{cases}$$

новое состояние памяти показано на рис. 6;

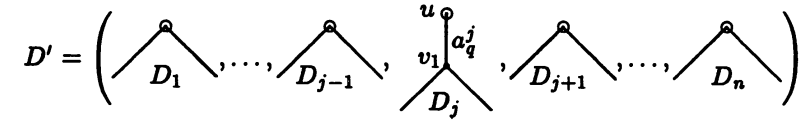


Рис. 6. Новое состояние памяти

4) v_{r_j} — заключительная (рис. 5,*г*), тогда $B'=(r, D, z, C, Act', Stop')$, где $Act'=Act\setminus\{j\}$ и $Stop'=Stop\cup\{j\}$;

5) v_{r_j} — вершина-источник, помеченная символом оператора отправления s_{qt}^j и $\mathbf{succ}(v_{r_j})=v_t$, тогда $B'=(r', D, z', C', Act', Stop)$, где (рис. 5,*д*)

если $z_i=0$, то

$$C'_m=\begin{cases}C_m, & m\neq q,\\ D_j * s_{qt}^j, & m=q,\end{cases}\qquad z'_m=\begin{cases}z_m, & m\neq q,\\ 1, & m=q,\end{cases}\qquad r'_m=\begin{cases}r_m, & m\neq j,\\ t, & m=j,\end{cases}$$

$Act'=Act$, новое состояние каналов показано на рис. 7,

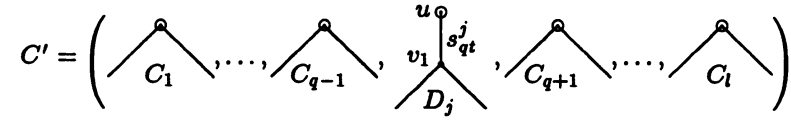


Рис. 7. Новое состояние каналов

а если $z_i=1$, то $r'=r$, $z'=z$, $C'=C$ и $Act'=Act\setminus\{j\}$;

6) v_{r_j} — вершина-приемник, помеченная символом оператора приема r_{qt}^j и $\mathbf{succ}(v_{r_j})=v_t$, тогда $B'=(r', D', z', C', Act', Stop)$, где (рис. 5,*е*)

если $z_i=1$, то

$$D'_m=\begin{cases}D_m, & m\neq j,\\ D_j * r_{qt}^j + C_q, & m=j,\end{cases}\qquad z'_m=\begin{cases}z_m, & m\neq q,\\ 0, & m=q,\end{cases}\qquad r'_m=\begin{cases}r_m, & m\neq j,\\ t, & m=j,\end{cases}$$

$Act'=Act$, новое состояние памяти процессов показано на рис. 8 (при этом $C_q=T_2 * s_{qp}^m$, где T_2 — некоторое дерево),

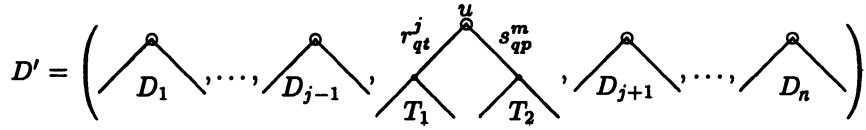


Рис. 8. Новое состояние памяти

а если $z_i=0$, то $r'=r$, $z'=z$, $D'=D$ и $Act'=Act\setminus\{j\}$.

Затем выполняется процедура пошаговой коррекции состояний активности процессов.

Пусть $B'=(r', D', z', C', Act', Stop')$ — состояние системы после некоторого шага. Рассмотрим последовательно все $j\in\{1,\ldots,k\}\setminus(Act'\cup Stop')$ и для каждого процесса Γ_j выполним следующую модификацию состояния B':

а) если $v_{r_j'}$ — вершина-источник с пометкой s_{qt}^j и $z'_q=0$, то модифицируем B', полагая $B'=(r', D', z', C', Act'\cup\{j\}, Stop')$;

б) если $v_{r_j'}$ — вершина-приемник с пометкой r_{qt}^j и $z'_q=1$, то модифицируем B', полагая $B'=(r', D', z', C', Act'\cup\{j\}, Stop')$.

Полученное по окончании второго этапа модифицированное состояние B' считается *результатом непосредственного преобразования состояния B схемой G*, а само отношение непосредственного преобразования обозначается при этом $B\xrightarrow{j}B'$.

Как и в случае стандартных ССВП $Act=\varnothing$ означает, что состояние B не допускает ни одного непосредственного преобразования. Если при этом $Stop=\{1,\ldots,k\}$, то такое состояние назовем *заключительным состоянием*. В том случае, если $Act=\varnothing$ и $j\notin Stop$ для некоторого процесса Γ_j, состояние A называется *тупиковым состоянием*.

Определение 16. *Вычислением пропозициональной ССВП* G назовем такую последовательность состояний $B_0, B_1, \ldots$ (возможно, бесконечную), что

B_0 — начальное состояние схемы;

для каждого члена B_i, $i > 0$, этой последовательности выполняется отношение

$$B_{i-1} \xrightarrow{n_i} B_i.$$

При этом последовательность $n_1 n_2 \ldots$ номеров процессов, активизированных на каждом шаге вычисления, назовем *планом данного вычисления.*

Так же как и в случае стандартных ССВП, мы будем говорить, что *вычисление* пропозициональной ССВП G

$$B_0 \xrightarrow{n_1} B_1 \xrightarrow{n_2} B_2 \xrightarrow{n_3} \ldots$$

выполняется по плану $H = n_1 n_2 n_3 \ldots$.

Совершенно аналогично случаю стандартных ССВП мы введем понятие завершающихся и тупиковых планов и вычислений пропозициональных ССВП.

Определение 17. Если на некотором плане H пропозициональная схема G завершает вычисление по функции разметки μ в заключительном состоянии $A = (r, D, z, C, Act, Stop)$, то набор деревьев D назовем *результатом* (*историей выполнения*) схемы G. Дерево D_i при этом назовем *историей выполнения* i*-го процесса.* В том случае, когда вычисление схемы является тупиковым или бесконечным, результат такого вычисления считается неопределенным.

Как и в случае стандартных ССВП, пропозициональная ССВП на одной и той же функции разметки допускает несколько различных вычислений в зависимости от выбранного плана, и поэтому для корректного определения понятий результата вычисления схемы на заданной функции разметки и эквивалентности схем необходимо убедиться в том, что результат функционирования пропозициональной ССВП не зависит от выбранного плана вычисления.

§ 4. Интерливинги

4.1. Трансляция стандартных ССВП в пропозициональные. Пусть есть стандартная ССВП $G = (\Gamma_1, \ldots, \Gamma_k)$. Перейдем к соответствующей ей пропозициональной схеме G_{pr} следующим образом. Для каждого процесса Γ_i

— все различные операторы присваивания $x_{i_1} := t_{i_1}, \ldots, x_{i_q} := t_{i_q}$ в Γ_i обозначим попарно различными символами $a_{i_1}, \ldots, a_{i_q}$;

— все различные операторы записи $\mathbf{send}(j, t_1), \ldots, \mathbf{send}(j, t_q)$ в j-й канал в процессе Γ_i обозначим попарно различными символами $s^i_{j1}, \ldots, s^i_{jq}$;

— все различные операторы $\mathbf{receive}(j, t_1), \ldots, \mathbf{receive}(j, t_v)$ чтения из j-го канала в процессе Γ_i обозначим попарно различными символами $r^i_{j1}, \ldots, r^i_{jv}$;

— все различные атомарные формулы

$$\pi_1(x^j_{11}, \ldots, x^j_{1k_1}), \ldots, \pi_w(x^j_{w1}, \ldots, x^j_{wk_w})$$

в процессе Γ_i обозначим попарно различными символами $p^i_1, \ldots, p^i_w$.

— удалим операторы **start** и **stop**.

Пропозициональной схемой $G_{\text{пр}}=(\Gamma_{\text{пр}_1},\ldots,\Gamma_{\text{пр}_k})$, соответствующей стандартной ССВП $G=(\Gamma_1,\ldots,\Gamma_k)$, назовем пропозициональную схему, полученную из G заменой меток вершин-операторов и преобразователей на их пропозициональные обозначения, введенные выше, и снятием меток с начальных и конечных вершин. Поскольку G — стандартная ССВП, процессы $\Gamma_1,\ldots,\Gamma_k$ удовлетворяют требованиям к стандартной ССВП, поэтому $G_{\text{пр}}$ удовлетворяет требованиям к пропозициональным схемам программ. Пусть $\mathbf{Op}(G_{\text{пр}})$ и $P(G_{\text{пр}})$ соответственно множество операторных символов $\{a_j^i, s_{jk}^i, r_{jk}^i\}$ и логических условий $p_1^i,\ldots,p_q^i$ пропозициональной схемы $G_{\text{пр}}$.

4.2. Интерливинги. Пусть задан некоторый алфавит, состоящий из множеств $A=\bigcup\limits_{i=1}^{k} A_i$ операторных символов присваивания, $P=\bigcup\limits_{i=1}^{k} P_i$ символов логических условий, $S=\bigcup\limits_{j=1}^{l} S_i$ символов операторов записи в канал, $R=\bigcup\limits_{j=1}^{l} R_i$ символов операторов чтения из каналов для пропозициональных ССВП из k процессоров с l каналами взаимосвязи. Обозначим через $\mathbf{Op}=A\cup S\cup R$ множество всех операторных символов. Для каждой цепочки (конечной или бесконечной) h из $\mathbf{Op}^*\cup\mathbf{Op}^\omega$ операторных символов и для каждого канала связи m, $1\leqslant m\leqslant l$, обозначим $h|_m$ подпоследовательность h, образованную всеми содержащимися в h операторами записи и считывания, относящимися к m-му каналу связи.

Определение 18. *Интерливингом (чередованием) в алфавите* $\mathbf{Op}$ назовем всякую цепочку h из $\mathbf{Op}^*\cup\mathbf{Op}^\omega$, удовлетворяющую следующему условию: для любого канала m, $1\leqslant m\leqslant l$, и для любого конечного префикса h' цепочки $h|_m$ количество операторов считывания в цепочке h' равно или на единицу меньше количества операторов записи.

Рассмотрим произвольную стандартную ССВП G, соответствующую ей пропозициональную ССВП $G_{\text{пр}}$ и некоторый план $H=n_1\ldots n_k\ldots$ вычисления схемы G на начальном состоянии памяти W_0 системы в некоторой интерпретации I. Пусть

$$Comp=A_0\xrightarrow{n_1}A_1\xrightarrow{n_2}A_2\xrightarrow{n_3}\ldots$$

есть вычисление стандартной схемы G по плану H.

Для заданного плана H построим последовательность операторных символов $T=t_1t_2\ldots$ по следующим правилам:

1) вначале полагаем $T=e$, где e — пустая цепочка, построенная в соответствии с пустым префиксом e плана H;

2) пусть построена цепочка $T=t_1\ldots t_{k-1}$ в соответствии с префиксом $\widehat{H}=n_1\ldots n_{q-1}$ плана H. Рассмотрим q-й член n_q последовательности H и q-й член $A=(r, W, z, U, Act, Stop)$ вычисления $Comp$. Если в процессе Γ_{n_q} пропозициональной схемы $G_{\text{пр}}$ текущая вершина v с номером r_{n_q} помечена операторным символом y, причем либо $y=a_j^{n_q}$ — оператор присваивания, либо $y=s_{jm}^{n_q}$ — оператор отправления, и $z_j=0$ (канал открыт для записи), либо $y=r_{jm}^{n_q}$ — оператор приема, и $z_j=1$ (канал открыт для прочтения), то приписываем к операторной цепочке T этот оператор, полагая $T=t_1\ldots\ldots t_{k-1}y$. В противном случае последовательность T не изменяется.

Очевидно, построенная цепочка T является некоторым интерливингом.

Определение 19. Последовательность операторных символов T, построенную указанным выше способом, назовем *интерливингом стандартной ССВП* G, *порожденным планом* H при начальном состоянии памяти W_0 в интерпретации I, и обозначим $T(H, W_0, I)$. Множество интерливингов, порождаемых различными планами при начальном состоянии W_0 в интерпретации I обозначим $Intr(W_0, I)$.

Фактически, интерливинг, порождаемый планом H при работе схемы G на (I, W_0), — это последовательность пропозициональных обозначений выполненных операторов схемы при ее работе на (I, W_0) в соответствии с планом H. Символы p_j^i, **start** и **stop** в интерливинг не попадают. Операторы из **Op** попадают в интерливинг только после своего фактического исполнения, т. е. если попытка выполнить оператор отправления или приема приводит к блокированию процесса, то данный оператор на этом этапе в интерливинг не попадает.

Определение 20. Интерливинг схемы G называется *завершающим*, если он порождается завершающим планом.

Определение 21. Пусть конечный интерливинг $T(H, W_0, I)$ порождается планом $H = n_1 n_2 \ldots n_m$. Пусть

$$Comp = A_0 \xrightarrow{n_1} A_1 \xrightarrow{n_2} A_2 \xrightarrow{n_3} \ldots \xrightarrow{n_m} A_{n_m}$$

есть вычисление стандартной схемы G на плане H и заключительная конфигурация есть $A_{n_m} = (r, W, z, U, Act, Stop)$. Тогда *результатом интерливинга* $T(H, W_0, I)$ назовем состояние памяти W из состояния системы A_{n_m} и обозначим его **res**(T, G, I, W_0).

Замечание. Два разных плана могут порождать один и тот же интерливинг.

Утверждение 1. *Пусть два плана H_1 и H_2 порождают в одной и той же ССВП G интерливинги $T(H_1, W_0, I)$ и $T(H_2, W_0, I)$, причем $T(H_1, W_0, I) = T(H_2, W_0, I)$. Тогда результаты этих интерливингов совпадают.*

Доказательство следует из алгоритма построения вычисления. Так как начальное состояние памяти фиксировано и впоследствии изменяется лишь при выполнении операторов из множества **Op** $= \{a_j^i, s_{jk}^i, r_{jk}^i\}$, а планы порождают один и тот же интерливинг, то результаты работы планов получаются из начального состояния применением одних и тех же операторов в одной и той же последовательности. Это означает, что результаты работы планов совпадают.

Следствие 1. *Определение* 21 *корректно, т. е. результат интерливинга не зависит от выбора плана, порождающего данный интерливинг.*

Следствие 2. *Пусть H_1 и H_2 — два плана, порождающие один и тот же интерливинг. Тогда значения любой атомарной формулы на состояниях памяти системы, полученных при работе G на планах H_1 и H_2 после выполнения последнего из операторов интерливинга, совпадают.*

Эти следствия прямо вытекают из утверждения 1.

Для проведения ряда дальнейших доказательств нам потребуется ввести понятия минимального плана, порождающего заданный интерливинг H, и пути выполнения интерливинга. Следствие 3 позволяет ввести эти понятия корректно.

Следствие 3. *Пусть два плана H_1 и H_2 порождают один и тот же интерливинг C. Пусть для каждого процесса Γ_i последовательности π_i и χ_i являются путями вычисления Γ_i на планах H_1 и H_2 соответственно. Тогда для каждого $i \in \{1, \ldots, k\}$ либо π_i является префиксом χ_i, либо χ_i является префиксом π_i (в этом случае будем говорить, что пути π_i и χ_i согласуются). При этом если $\pi_i = \chi_i \psi_i$, то путь ψ_i проходит только через вершины-распознаватели, заканчиваясь, возможно, в заключительной вершине или вершине с оператором обмена по каналу, приводящим к блокированию Γ_i, и наоборот, если $\chi_i = \pi_i \varphi_i$, то φ_i проходит через вершины тех же типов.*

Доказательство. Предположим, что для некоторого i ни π_i не является начальным отрезком χ_i, ни χ_i не является начальным отрезком π_i. Более того, пусть Γ_i — первый из процессов, в котором возникает это несоответствие, т. е. на момент его появления построенные пути в других процессах согласуются. Это значит, что при прокладке путей π_i и χ_i был встречен как минимум один распознаватель, при прохождении через который на планах H_1 и H_2 были выбраны разные ветви продолжения вычисления. Пусть p_j^i — первый из таких распознавателей в Γ_i (это значит, что до прохождения p_j^i пути π_i и χ_i совпадают). Тогда выполнение процесса Γ_i схемы по планам H_1 и H_2 доходит до распознавателя p_j^i, при этом к моменту его прохождения на плане H_1 был порожден некоторый интерливинг C_1, на плане H_2 — некоторый интерливинг C_2 (C_1 и C_2 — префиксы C).

Пусть $C_1 = u_1 \ldots u_{m_1} \ldots u_{n_1}$ и $C_2 = u_1 \ldots u_{m_2} \ldots u_{n_2}$, где u_{m_k} — последний оператор процесса Γ_i, выполненный по плану H_k, $k = 1, 2$, перед прохождением распознавателя p_j^i. Планы H_1 и H_2 порождают один и тот же интерливинг C, поэтому либо C_1 — префикс C_2, либо C_2 — префикс C_1. Пути выполнения схемы G на планах H_1 и H_2 до момента прохождения p_j^i в процессе Γ_i совпадали, поэтому $m_1 = m_2$ и $C_0 = u_1 \ldots u_{m_1}$ — общий префикс C_1 и C_2. Поэтому значения истинности атомарной формулы, приписанной p_j^i, на результатах работы C_1 и C_2 совпадают. Это противоречит тому, что при прохождении через распознаватель p_j^i на планах H_1 и H_2 выбираются разные ветви продолжения вычисления. Таким образом, либо π_i является префиксом χ_i, либо наоборот.

Пусть π_i — префикс χ_i, $\chi_i = \pi_i \theta_i$. Тогда θ_i проходит лишь по вершинам-распознавателям, заканчиваясь, быть может, в вершинах-распознавателях, вершинах с операторами завершения либо обмена данными, в которых происходит блокирование процесса, поскольку прохождение θ_i через вершины-преобразователи или через вершины с операторами обмена без блокировки Γ_i привело бы к занесению этих операторов в интерливинг, порождаемый планом H_2, но не в интерливинг, порождаемый H_1, в результате чего H_1 и H_2 не порождали бы один и тот же интерливинг.

Определение 22. План $H = n_1 \ldots n_q \ldots$ называется *минимальным порождающим (м.п.) планом для интерливинга* T, если H порождает интерливинг T, но никакая собственная подпоследовательность H' плана H не порождает T.

Это означает, что м.п. план H для конечного интерливинга T конечен и никакой процесс Γ_i, выполнив все свои операторы, входящие в интерливинг T, не «уходит» по распознавателям дальше, пока «отставшие» процессы выполняют свои операторы из T. Путь выполнения схемы на м.п. плане H для интерливинга T, таким образом, в каждом процессе завершается на вершине с оператором, метка которого последней была добавлена в интерливинг.

Определение 23. *Путем выполнения интерливинга* T назовем набор путей $\pi_1, \ldots, \pi_k$ выполнения процессов $\Gamma_1, \ldots, \Gamma_k$ схемы G на минимальном порождающем плане H для интерливинга T.

Очевидно, что у каждого интерливинга есть свой м.п. план (берем любой конечный план, порождающий интерливинг, и прореживаем его до тех пор, пока он не перестанет порождать этот интерливинг). Если H_1 и H_2 — м.п. планы, порождающие один и тот же интерливинг T, то по следствию 3 пути вычисления каждого процесса Γ_i на планах H_1 и H_2 совпадают. Таким образом, определение пути выполнения интерливинга корректно, т. е. не зависит от выбора м.п. плана.

Утверждение 2. *Пусть T — интерливинг вида $T = T_0 v_1 v_2 \widehat{T}_0$ схемы G, где $v_1 \in \mathbf{Op}(\Gamma_i)$, $v_2 \in \mathbf{Op}(\Gamma_j)$. Тогда существует план выполнения G, имеющий вид $H = H_0 ij \widehat{H}_0$, порождающий T, причем операторы v_1, v_2 выполняются на шагах $|H_0|+1$ и $|H_0|+2$.*

Доказательство. Так как T — интерливинг, то он порождается некоторым планом H'. Пусть H' имеет вид

$$H_0' n_1 \dots n_s i m_1 \dots m_r j \widehat{H}_0',$$

где H_0' порождает интерливинг T_0, на шагах $n_1, \dots, n_s$ и $m_1, \dots, m_r$ интерливинг не изменяется, на шаге i в интерливинг добавляется оператор v_1, на шаге j — оператор v_2, а затем выполнение G и построение T продолжаются в соответствии с планом $\widehat{H}_0'$.

Пусть $n_1 = |H_0'| + s + 1$ — номер шага, на котором в интерливинг был добавлен оператор v_1 и $n_2 = |H_0'| + s + r + 2$ — номер шага, на котором был добавлен оператор v_2. Перед шагом n_1 работы алгоритма построения вычисления схемы G на H' текущая вершина в Γ_i помечена операторным символом v_1, а текущая вершина в Γ_j — это либо вершина, помеченная операторным символом v_2 (если $m_l \neq j$ для любого $l \in \{1, \dots, r\}$), либо это распознаватель p_l^j, причем если q чисел среди $m_1, \dots, m_r$ равны j, то путь от этого распознавателя, определяемый значениями истинности атомарных формул процесса Γ_i на состоянии памяти, полученном после шага n_1 (которое совпадает с состоянием памяти перед шагом n_2), проходит через q распознавателей в вершину, помеченную операторным символом v_2.

Пусть $t_1, \dots, t_{r-q}$ — все числа из $m_1, \dots, m_r$, не равные j, взятые в том же порядке. Тогда рассмотрим план

$$H = H_0' n_1 \dots n_s jjj \dots jijt_1 \dots t_{r-q} \widehat{H}_0'.$$

В силу того, что выполнение операторов Γ_i не влияет на состояние памяти процесса Γ_j, состояние памяти Γ_j перед шагом n_2 при выполнении на H совпадает с состоянием памяти Γ_j после прохождения плана H_0', поэтому на шагах $n_1, \dots, n_1 + q$ процесс Γ_j проходит тот же путь по распознавателям, приходя в вершину, помеченную v_2, что и при прохождении плана H' на шагах от n_1 до n_2. В процессе Γ_i на этот момент текущей является вершина с оператором v_1, она проходится на следующем шаге, затем в Γ_j проходится текущая вершина, помеченная v_2, затем все остальные процессы проходят те же пути, чтобы попасть в то же состояние, что и после шага n_2 при выполнении по плану H', и затем выполнение происходит по общему суффиксу $\widehat{H}_0'$ планов H и H'. Таким образом, план H также порождает интерливинг T.

4.3. Преобразования интерливингов. Докажем корректность интерпретационной модели ССВП, т. е. независимость результата функционирования схемы от выбора плана исполнения, а именно покажем, что либо на всех планах схема G останавливается с одним и тем же результатом, либо на всех планах схема G не останавливается. Для этого покажем, что любой план вычисления может быть замещен планом особого вида — каноническим планом. Обоснование этого факта достигается путем введения системы S преобразований интерливингов. Как будет показано далее, интерливинг, порождаемый произвольным планом, преобразованиями из этой системы приводится к каноническому интерливингу, порождаемому каноническим планом. Поскольку преобразования системы S не изменяют результата интерливинга, отсюда будет следовать требуемый результат.

Рассмотрим следующую систему тождеств на множестве цепочек операторных символов:

1) $a_q^i y \equiv y a_q^i$, где y — любой из операторов вида a_t^j, s_{mt}^j, r_{mt}^j, $i \neq j$;

2) $s_{qs}^i y \equiv y s_{qs}^i$, где y — оператор вида a_t^j, s_{mt}^j, $i \neq j$, или оператор вида r_{mt}^j, $i \neq j$, $q \neq m$;

3) $r_{qs}^i y \equiv y r_{qs}^i$, где y — оператор вида a_t^j, r_{mt}^j, $i \neq j$ или оператор вида s_{mt}^j, $i \neq j$, $q \neq m$.

На множестве интерливингов введем отношение транспозиции *Trn*, полагая для любой пары интерливингов $Trn(T_1, T_2)$, если T_2 получен из T_1 перестановкой пары соседних операторных символов $y_1 y_2$, для которой справедливо одно из тождеств 1)–3).

Утверждение 3. *Транзитивное замыкание Trn* отношения транспозиции Trn является отношением эквивалентности на множестве интерливингов.*

Утверждение 4. *Пусть T_1 и T_2 — два конечных эквивалентных интерливинга вычисления стандартной ССВП G для одного и того же начального состояния памяти W_0 в интерпретации I. Тогда результаты этих интерливингов совпадают.*

Класс эквивалентности интерливингов по отношению *Trn** назовем *трассой* [8].

Введем на операторах схемы отношение приоритета: скажем, что оператор v_1 имеет *более высокий приоритет*, чем v_2, если

1) $v_1 = r_{qw}^i$, $v_2 = r_{tm}^j$ и $i < j$,

2) $v_1 = r_{qw}^i$, v_2 — оператор s_{tm}^j или a_t^j при любых $i \neq j$,

3) $v_1 = s_{qw}^i$, $v_2 = s_{tm}^j$ и $i < j$,

4) $v_1 = s_{qw}^i$, v_2 — оператор a_t^j при любых $i \neq j$,

5) $v_1 = a_q^i$, $v_2 = a_t^j$ и $i < j$.

Помимо этого мы будем считать, что всякое логическое условие имеет более высокий приоритет, чем любой оператор-преобразователь или оператор обмена, из двух логических условий p_m^j и p_n^i большим приоритетом обладает условие, относящееся к процессу с меньшим номером, любой оператор **start** приоритетнее любого логического условия и их приоритет между собой также находится в обратной зависимости от номера процесса, которому принадлежит оператор.

Опираясь на введенное таким образом понятие приоритета операторов, образуем систему правил переписывания интерливингов

$$\mathrm{RW} = \{(y_1 y_2) \mapsto (y_2 y_1) : y_1 y_2 \equiv y_2 y_1, \text{ приоритет у } y_2 \text{ выше, чем у } y_1\}.$$

Правила переписывания системы RW позволяют осуществлять эквивалентные преобразования интерливингов сообразно отношению приоритета путем замены всякого вхождения левой части правила в интерливинг на соответствующую правую часть того же правила. Применение переписывающего правила к интерливингу T будем обозначать $T \mapsto T'$.

Напомним некоторые основные понятия, относящиеся к теории переписывания.

Интерливинг T назовем *нормальным* относительно RW, если ни одно правило переписывания из RW не применимо к T. Система правил переписывания называется

— *нётеровой*, если для любого интерливинга T не существует бесконечной последовательности $T \mapsto T' \mapsto T'' \mapsto \ldots$;

— *направленной*, если для любой тройки интерливингов T, T', T'' такой, что $T \mapsto T'$, $T \mapsto T''$, существует интерливинг T''', обладающий свойством $T' \mapsto T'''$, $T'' \mapsto T'''$.

Выполнимость свойств нётеровости и направленности означает, что любой интерливинг можно привести к нормальной форме, применяя в произвольном порядке правила переписывания, причем эта нормальная форма определяется исходным интерливингом однозначно (*свойство Чёрча–Россера*).

Утверждение 5. *Система переписывания RW обладает свойствами нётеровости и направленности, и, следовательно, свойством Чёрча–Россера.*

Следствие 4. *Для любой пары эквивалентных интерливингов T_1 и T_2 существует единственная нормальная форма T_0, к которой интерливинги T_1 и T_2 могут быть приведены путем применения правил системы RW.*

Опираясь на введенное таким образом понятие приоритета операторов, введем определение канонического плана выполнения схемы G.

Определение 24. Пусть $G = (\Gamma_1, \ldots, \Gamma_k)$ — стандартная схема k процессов. Пусть план $H = n_1 n_2 n_3 \ldots$ порождает вычисление

$$Comp = A_0 \xrightarrow{n_1} A_1 \xrightarrow{n_2} A_2 \xrightarrow{n_3} \ldots \xrightarrow{n_m} A_{n_m}.$$

Тогда план H назовем *каноническим* в том случае, когда для каждого $i = 0, 1, 2, \ldots$ выполнено следующее условие: если конфигурация A_i имеет вид

$$A_i = (r, Mem, z, ChMem, Act, Stop),$$

где $Act = \{j_1, \ldots, j_q\}$, $r = (r_1, \ldots, r_k)$, а $y_1, \ldots, y_q$ — символы, помечающие вершины процессов $\Gamma_{j_1}, \ldots, \Gamma_{j_q}$ с номерами $r_{j_1}, \ldots, r_{j_q}$ соответственно в пропозициональной схеме G_{pr}, и при этом символ y_m имеет наивысший приоритет среди $y_1, \ldots, y_q$, то $n_i = m$ (т. е. на каждом шаге вычисления выполняется действие, имеющее наивысший приоритет среди всех допустимых для исполнения на данном шаге действий вычисления).

Утверждение 6. *Для заданной стандартной ССВП G интерпретации I и начального состояния W_0 канонический план H вычисления G определяется однозначно.*

Поэтому в дальнейшем канонический план схемы G для начального состояния W_0 в интерпретации I условимся обозначать $H(G, W_0, I)$.

Определение 25. Интерливинг T схемы G называется *каноническим*, если он порождается каноническим планом.

Утверждение 7. *Пусть T — завершающий интерливинг схемы G, порожденный планом H для начального состояния W_0 в интерпретации I, и $\hat{T}$ получен из T применением правил системы RW. Тогда $\hat{T}$ — интерливинг схемы G, т. е. он порождается некоторым планом $\hat{H}$, и результаты T и $\hat{T}$ совпадают.*

Доказательство. Достаточно рассмотреть случай, когда $\hat{T}$ получен из T применением лишь одного правила системы RW. Для доказательства проведем индукцию по длине интерливингов. Так как T — конечный и $\hat{T}$ получен из T одной из перестановок системы S, то $|T| = |\hat{T}|$.

Если $T = e$ (пустая цепочка), то $\hat{T} = e$. Тогда $\hat{T}$, так же как и T, порождается пустым планом. Результаты их работы совпадают с начальным состоянием памяти системы W_0.

Пусть теперь утверждение верно для интерливингов длины, не превосходящей $n-1$. Пусть $T = u_1 \ldots u_{n-2} v_1 v_2 u_{n+1} \ldots$ — интерливинг G и цепочка $\hat{T} = u_1 \ldots u_{n-2} v_2 v_1 u_{n+1} \ldots$ получается из T перестановкой двух операторов v_1 и v_2 правилом из системы S.

Рассмотрим случаи, возникающие в зависимости от конкретного вида операторов v_1 и v_2.

Случай $v_1 = a_q^i$, $v_2 = a_t^j$ и $i \neq j$. По утверждению 2 существует план $H = H_0 ij \widehat{H}_0$, порождающий T. Пусть $H = n_1 \ldots n_s ijn_{s+3} \ldots$. Обозначим $A = (r_k, W_k, z_k, U_k, Act_k, Stop_k)$ — состояние системы перед, а $A' = (r'_k, W'_k, z'_k, U'_k, Act'_k, Stop'_k)$ — после k-го шага работы алгоритма, $k = 1, 2, \ldots$. Тогда $\{i, j\} \subseteq Act'_s$, так как из $i \notin Act'_s$ следует, что H не отвечает определению плана, поскольку на $(s+1)$-м шаге построения вычисления для исполнения берется неактивный процесс Γ_i. Если же $j \notin Act'_s$, то $j \notin Act'_{s+1}$, поскольку выполнение на $(s+1)$-м шаге оператора a_j^i из $\mathbf{Op}(\Gamma_i)$ не влияет на активность процесса Γ_j, и H также не соответствует определению плана. Но если $\{i, j\} \subseteq Act'_s$, то последовательность $\widehat{H} = n_1 \ldots n_s jin_{n+3} \ldots$ — план работы схемы G и $\widehat{H}$ порождает $\widehat{T}$.

Совпадение результатов их выполнения следует из того, что состояния системы после s-го шага работы G на H и $\widehat{H}$ совпадают, выполнение оператора a_q^i не изменяет состояния памяти Γ_j, а выполнение оператора a_t^j не изменяет состояния памяти Γ_i, следовательно, преобразования, выполняемые операторами a_q^i и a_t^j в памяти процессов Γ_i и Γ_j, и состояние управления системы после выполнения обоих операторов не зависят от порядка их выполнения, а остальные параметры системы при этом не меняются. Поэтому последующие состояния системы при работе G на общем суффиксе $\widehat{H}_0$ планов H и $\widehat{H}$ совпадают и, как следствие, результаты работы G на H и $\widehat{H}$ совпадают.

Случаи $v_1 = a_q^i$, $v_2 = s_{tm}^j$ и $v_1 = a_q^i$, $v_2 = r_{tm}^j$ при $i \neq j$ рассматриваются аналогично. Совпадение состояний системы после построения обоих интерливингов следует из того, что выполнение a_q^i не влияет на состояние каналов системы и состояние памяти процесса Γ_j, а выполнение v_2 не влияет на состояние памяти процесса Γ_i.

Случай $v_1 = s_{qk}^i$, $v_2 = s_{tl}^j$ при $i \neq j$ (а, значит, $q \neq t$). По утверждению 2 существует план $H = H_0 ij \widehat{H}_0 = n_1 \ldots n_s ijn_{s+3} \ldots$, порождающий интерливинг T. В введенных выше обозначениях необходимо, чтобы $\{i, j\} \subseteq Act'_s$, $z'_{s_q} = 0$ и $z'_{s_t} = 0$, поскольку если $i \notin Act'_s$, то H не отвечает определению плана; если $j \notin Act'_s$, то $j \notin Act'_{s+1}$ так как выполнение на $(s+1)$-м шаге оператора s_{qk}^i не влияет на активность j-го процесса и H опять не отвечает определению плана. Если $z'_{s_q} = 1$, то на $(s+1)$-м шаге попытка выполнить оператор s_{qk}^i приводит к блокировке процесса Γ_i. Активность Γ_i после этого может быть восстановлена лишь после выполнения некоторого оператора r_{qv}^m, но это приведет к занесению r_{qv}^m в интерливинг T после u_{n-2} и перед $v_1 = s_{qk}^i$, а значит H порождает интерливинг, отличный от T. Если $z'_{s_t} = 1$, то $z'_{(s+1)_t} = 1$, поскольку выполнение оператора s_{qk}^i записи в q-й канал на $(s+1)$-м шаге не изменяет состояния t-го канала. Отсюда следует, что на $(s+2)$-м шаге Γ_j потеряет активность, которая может быть восстановлена лишь выполнением некоторого оператора r_{tu}^m, что приведет к занесению этого оператора в интерливинг между операторами v_1 и v_2. Но тогда H опять порождает интерливинг, отличный от T.

В этой ситуации последовательность $\widehat{H} = n_1 \ldots n_s jin_{s+3} \ldots$ есть корректный план выполнения схемы G: до s-го шага он совпадает с H; далее, поскольку $j \in Act'_s$ и $z'_{s_t} = 0$, то Γ_j может быть выбран для исполнения (как это предписывает план $\widehat{H}$), в Γ_j выполняется оператор s_{tl}^j (так как

t-й канал свободен для записи). Он заносится в интерливинг, его выполнение не влияет на активность Γ_i и состояние q-го канала, следовательно, $i \in Act'_{s+1}$ и на $(s+2)$-м шаге для исполнения может быть выбран активный процесс Γ_i. Так как $z'_{(s+1)_q} = 0$, в нем будет выполнен и занесен в интерливинг оператор s^i_{qk}. Указанный план $\widehat{H}$ порождает интерливинг $\widehat{T}$.

Совпадение результатов работы планов H и $\widehat{H}$ следует из семантики операторов отправления: они не изменяют состояние памяти процессов, поэтому состояния q-го и t-го каналов и состояние управления после выполнения обоих операторов в разных порядках совпадают, остальные параметры системы при этом не меняются. Кроме того, планы H и $\widehat{H}$ имеют общий префикс H_0 и общий суффикс $\widehat{H}_0$, поэтому результат T совпадает с результатом $\widehat{T}$.

Случай $v_1 = s^i_{jk}$, $v_2 = a^j_t$ рассматривается аналогично.

Случай $v_1 = s^i_{qk}$, $v_2 = r^j_{tl}$, где $i \neq j$ и $q \neq t$. По утверждению 2 существует план вида $H = H_0 ij \widehat{H}_0$, порождающий интерливинг T. В этом случае необходимо, чтобы выполнялось $\{i, j\} \subseteq Act'_s$, $z'_{s_q} = 0$ и $z'_{s_t} = 1$. Если $i \notin Act'_s$, то H не отвечает определению плана; если $j \notin Act'_s$, то выполнение на $(s+1)$-м шаге оператора s^i_{qk} не влияет на активность процесса Γ_j, следовательно, $j \notin Act'_{s+1}$ и H опять не отвечает определению плана. Если $z'_{s_q} = 1$, то на s-м шаге не может выполниться оператор s^i_{qk}, активизация процесса Γ_i возможна лишь после выполнения некоторого оператора r^m_{qv} чтения из q-го канала после операторов $u_1, \ldots, u_{n-2}$, значит, H порождает интерливинг, отличный от T. Если $z'_{s_t} = 0$, то так как выполнение s^i_{qk} на $(s+1)$-м шаге не изменяет состояния t-го канала, получаем, что на $(s+2)$-м шаге оператор r^j_{tl} не может быть выполнен, для этого после выполнения $u_1, \ldots, u_{n-2}$, s^i_{qk} и перед $r^j_{t_l}$ должен выполниться некоторый оператор s^m_{tu} записи в q-й канал, т. е. H опять порождает отличный от T интерливинг.

Из того, что $\{i, j\} \subseteq Act'_s$, $z'_{s_q} = 0$ и $z'_{s_t} = 1$, получаем, что последовательность $\widehat{H} = H_0 ji \widehat{H}_0$ — корректный план выполнения схемы G и он порождает интерливинг $\widehat{T}$.

Совпадение результатов работы T и $\widehat{T}$ следует из того, что изменения, вносимые операторами s^i_{qk} и r^j_{tl} в состояние памяти Γ_j, состояния каналов q и t и состояния управления системы по семантике операторов отправления и приема и при $q \neq t$, не зависят от порядка выполнения s^i_{qk} и r^j_{tl}, а все остальные параметры системы этими операторами не изменяются. Это значит, что состояния системы после $(s+2)$-го шага при работе G на планах $H_0 ij$ и $H_0 ji$ совпадают, а потому и дальнейшие состояния системы при работе G на общем суффиксе $\widehat{H}_0$ планов H и $\widehat{H}$ будут совпадать. Поэтому и результаты работы интерливингов T и $\widehat{T}$ совпадают.

Случаи $v_1 = r^i_{qk}$, $v_2 = a^j_t$; $v_1 = r^i_{qk}$, $v_2 = s^j_{tm}$; $v_1 = r^i_{qk}$, $v_2 = r^j_{tm}$ рассматриваются аналогично.

Утверждение 8. *Пусть $T_1 = Tv_1v_2\widehat{T}$ и $T_2 = Tv_2v_1\widehat{T}$ — два интерливинга, получающиеся друг из друга одной из перестановок системы S. Тогда пути выполнения интерливингов T_1 и T_2 совпадают.*

Доказательство. Как следует из утверждения 2 и доказательства утверждения 7, существуют планы H_1 и H_2, порождающие T_1 и T_2, такие что $H_1 = Hij\widehat{H}$ и $H_2 = Hji\widehat{H}$, причем после выполнения плана H текущей в процессе Γ_i является вершина w_1, помеченная символом v_1, текущей в процессе Γ_2 является вершина w_2, помеченная символом v_2. Затем

на H_1 (H_2) выполняются операторы v_1, v_2 (соответственно, v_2, v_1) и работа схемы продолжается на плане $\widehat{H}$. Таким образом, пройти через любой распознаватель p_i^q схема может, работая на общем префиксе H планов H_1 и H_2 либо на их общем суффиксе $\widehat{H}$. В первом случае выбор ветви дальнейшего вычисления при прохождении через p_i^q делается одинаковый, так как H — общий префикс H_1 и H_2. Во втором случае выбор также одинаков поскольку результаты интерливингов Tv_1v_2 и Tv_2v_1 совпадают, схема G на планах Hij и Hji приходит в одно и то же состояние управления, в котором текущая вершина Γ_i есть $\mathbf{succ}(w_1)$ и текущая вершина Γ_j есть $\mathbf{succ}(w_2)$, затем выполнение происходит по общему суффиксу $\widehat{H}$ планов H_1 и H_2. Поэтому при прохождении через любой распознаватель p_i^q делается один и тот же выбор ветви выполнения, следовательно, пути выполнения T_1 и T_2 совпадают.

Утверждение 7 позволяет в любой последовательности применять правила системы RW к интерливингу без влияния на его результат. Основываясь на этом, укажем алгоритм приведения некоторого интерливинга к каноническому путем применения правил системы RW.

Теорема 1. *Пусть G — стандартная ССВП, I — интерпретация, W_0 — начальное состояние, H — некоторый план вычисления. Тогда вычисление G на плане H завершается с результатом W в том и только том случае, когда вычисление G на каноническом плане $H(G, W_0, I)$ завершается с тем же результатом W.*

Доказательство. Пусть $T = u_1 \dots u_n$ — завершающий интерливинг схемы G. Применим к нему следующий алгоритм приведения к каноническому виду. Для каждого $i \in \{1, \dots, n\}$ рассмотрим оператор u_i. Пусть $u_i \in \mathbf{Op}(\Gamma_l)$, где l — некоторый номер. Найдем максимальное k такое, что $k < i$ и $u_k \in \mathbf{Op}(\Gamma_l)$, либо положим $k = 0$, если u_i — первый оператор процесса Γ_l в интерливинге. Фактически, k — позиция последнего перед u_i оператора процесса Γ_l. Таким образом, $T = u_1 \dots u_k \dots u_i \dots$, где между u_k и u_i нет операторов из Γ_l. Если $u_i = s_{qt}^l$ (или r_{qt}^l), то на участке $u_{k+1} \dots u_{i-1}$ может быть не более одного оператора чтения r_{qw}^v из q-го канала (соответственно, оператора записи s_{qw}^v в q-й канал), поскольку иначе в участке $u_{k+1} \dots u_{i-1}$ будет содержаться участок $r_{qw}^l \dots r_{qv}^l$ (соответственно, $s_{qw}^l \dots s_{qv}^l$), на котором нет операторов процесса Γ_l, а, значит, и операторов отправления в q-й канал (соответственно, приема из q-го канала), что противоречит тому, что $u_1 \dots u_i$ — интерливинг.

В случае, если u_i — оператор обмена по q-му каналу и на участке $u_{k+1} \dots u_{i-1}$ есть единственный оператор обмена по q-му каналу u_m из другого процесса, обозначим через m позицию этого парного u_i оператора. В случае же отсутствия такого парного оператора или если u_i — оператор преобразования a_j^i, то полагаем $m = k$. Таким образом, получен участок интерливинга $u_{m+1}, \dots, u_{i-1}$, на котором нет операторов из процесса Γ_l и, если u_i — оператор обмена по q-му каналу, то нет и операторов обмена по q-му каналу из других процессов.

Рассмотрим числа $m+1, \dots, i-1$. Пусть r — наименьшее из них, такое, что u_r — оператор из группы меньшей важности, чем u_i, либо оператор той же группы важности из процесса с номером большим чем l, т. е. u_r обладает меньшим приоритетом, чем u_i (см. п. 4.3). Тогда удаляем u_i с позиции с номером i и вставляем его в интерливинг перед оператором u_r. Если таких r вообще нет, т. е. u_r — оператор с большим, чем у u_i, приоритетом для любого $r \in \{m+1, \dots, i-1\}$, то интерливинг оставляем без изменения. Очевидно, что такое преобразование интерливинга может быть выполнено применением конечного числа перестановок из S: так как среди

$u_r, \ldots, u_{i-1}$ нет операторов из Γ_l и операторов обмена по тому же каналу, что и u_i, то все они согласно системе S перестановочны с u_i, и его можно переставить на место перед u_r, последовательно меняя местами с предшествующими.

Применяем такое преобразование для всех $i \in \{1, \ldots, n\}$, беря за исходный для первого шага начальный интерливинг T, для последующих шагов — интерливинг, полученный на предыдущем шаге. Алгоритм завершает работу, если ни один из операторов интерливинга, полученного на некотором шаге, не может быть перенесен ближе к началу. Поскольку при работе алгоритма операторы выстраиваются в порядке убывания приоритета, число шагов алгоритма на конечном интерливинге T конечно. Пусть в результате получен интерливинг T_0. Покажем, что T_0 — канонический.

Допустим противное. Предположим, что интерливинг T_0 получен описанным выше алгоритмом, интерливинг $\widehat{T}$ — канонический (т. е. он порожден каноническим планом) и $T_0 \neq \widehat{T}$. Пусть $\widehat{T} = u_1 \ldots u_k v_1 \ldots$, а $T_0 = u_1 \ldots u_k v_2 \ldots$ и $v_1 \neq v_2$.

В зависимости от конкретного вида v_1 рассмотрим следующие случаи.

а) $v_1 = r_{qt}^i$. Интерливинг $\widehat{T}$ порождается каноническим планом, следовательно, на некотором шаге работы G по каноническому плану после выполнения операторов $u_1, \ldots, u_k$ (при этом u_k — последний выполненный оператор) текущая вершина процесса Γ_i есть оператор чтения r_{qt}^i, среди текущих вершин в каждом из активных после выполнения операторов $u_1, \ldots, u_k$ процессов нет распознавателей и среди номеров всех активных процессов, в которых текущая вершина — вершина-приемник, i — минимальный.

Поскольку первые k операторов у T_0 и $\widehat{T}$ совпадают, можно считать, что до момента выполнения оператора u_k включительно планы, порождающие T_0 и $\widehat{T}$, совпадают. Тогда и при работе G на плане, порождающем T_0, на некотором шаге алгоритма построения вычисления после выполнения операторов $u_1, \ldots, u_k$ и до выполнения v_2 текущая вершина процесса Γ_i помечена r_{qt}^i. Поскольку T_0 получен из завершающего интерливинга T применением перестановок из системы S, то T_0 — также завершающий интерливинг схемы G, а, значит, процесс Γ_i на плане, порождающем T_0, доходит до завершающей вершины. Поэтому после выполнения операторов $u_1, \ldots, u_k$ процесс Γ_i проходит через вершину r_{qt}^i, что означает, что T_0 имеет вид $T_0 = u_1 \ldots u_k v_2 \ldots u_m r_{qt}^i \ldots$, причем на участке $v_2 \ldots r_{qt}^i$ нет операторов из Γ_i, так как после выполнения $u_1, \ldots, u_k$ процесс Γ_i приходит в вершину r_{qt}^i лишь по распознавателям или непосредственно. Так как $\widehat{T}$ имеет вид $\widehat{T} = u_1 \ldots u_k r_{qt}^i \ldots$, то r_{qt}^i был выполнен $(k+1)$-м, поэтому q-й канал после выполнения $u_1, \ldots, u_k$ был занят. Отсюда следует, что q-й канал при построении интерливинга T_0 после выполнения $u_1, \ldots, u_k$ был занят. Но это значит, что в T_0 на участке $v_2 \ldots u_m$ нет операторов записи в q-й канал. Кроме того, поскольку $v_1 \neq v_2$, то v_2 — оператор с меньшим приоритетом, чем v_1. Но поскольку на $v_2 \ldots u_m$ нет операторов из Γ_i и операторов взаимодействия через q-й канал и v_2 — оператор более низкого приоритета, чем v_1, то, согласно алгоритму приведения к каноническому виду, на i-м шаге надо поставить оператор v_1 перед v_2. Таким образом, работа алгоритма на i-м шаге не доведена до конца, и его работу можно продолжить. Это противоречит тому, что T_0 — окончательный результат работы алгоритма.

б) $v_1 = s_{qt}^i$. Так как $\widehat{T}$ порождается каноническим планом, то среди процессов, активных после выполнения операторов $u_1, \ldots, u_k$, нет таких, в которых текущая вершина — вершина-приемник, и среди номеров всех активных после $u_1, \ldots, u_k$ процессов, в которых текущая вершина — вер-

шина-приемник, i — минимальный. Это означает, что на некотором шаге работы G на каноническом плане после выполнения операторов $u_1, \ldots, u_k$ текущая вершина процесса Γ_i есть оператор записи s^i_{qt}.

Аналогично предыдущему случаю показывается, что интерливинг T_0 имеет вид $T_0 = u_1 \ldots u_k v_2 \ldots u_m s^i_{qt} \ldots$, причем на участке $v_2 \ldots u_m$ нет операторов из Γ_i. Так как $\widehat{T}$ имеет вид $\widehat{T} = u_1 \ldots u_k s^i_{qt} \ldots$, то s^i_{qt} был выполнен $(k+1)$-м, поэтому q-й канал после выполнения $u_1, \ldots, u_k$ был свободен. Отсюда следует, что q-й канал при построении интерливинга T_0 после выполнения $u_1, \ldots, u_k$ был свободен. Но это значит, что в T_0 на участке $v_2 \ldots u_m$ нет операторов приема из q-го канала (они не могут выполниться пока в канал не будет записано сообщение одним из операторов s^i_{qw}, которые принадлежат процессу Γ_i и отсутствуют среди $v_2, \ldots, u_m$). Кроме того, поскольку $v_1 \neq v_2$, то v_2 — оператор с меньшим приоритетом, чем v_1. Отсюда, так как на $v_2, \ldots, u_m$ нет операторов из Γ_i и операторов взаимодействия через q-й канал, согласно алгоритму, на i-м шаге требуется поставить v_1 перед v_2. Таким образом, работа алгоритма на i-м шаге до конца не доведена и ее можно продолжить.

в) $v_1 = a^i_j$. Так как $\widehat{T}$ порождается каноническим планом, то среди процессов, активных после выполнения операторов $u_1, \ldots, u_k$, нет таких, в которых текущая вершина — вершина-приемник или источник и среди номеров всех активных после выполнения $u_1, \ldots, u_k$ процессов, в которых текущая вершина — вершина-преобразователь, i — минимальный.

Аналогично предыдущему случаю показывается, что интерливинг T_0 имеет вид $T_0 = u_1 \ldots u_k v_2 \ldots u_m a^i_j \ldots$, причем на участке $v_2 \ldots a^i_q$ нет операторов из Γ_i. Так как $\widehat{T}$ имеет вид $\widehat{T} = u_1 \ldots u_k a^i_j \ldots$, то a^i_j был выполнен $(k+1)$-м, поэтому после выполнения $u_1, \ldots, u_k$ процесс Γ_i был активен. Выполнение каждого из операторов $u_1, \ldots, u_k$ не влияет на активность процесса Γ_i, и поскольку $v_1 \neq v_2$, то v_2 — оператор преобразования с большим, чем i, номером процесса, т. е. оператор с меньшим приоритетом, чем u_i. Отсюда, так как на $v_2 \ldots u_m$ нет операторов из Γ_i, согласно алгоритму, на i-м шаге требуется поставить v_1 перед v_2. Таким образом, работа алгоритма на i-м шаге до конца не доведена и ее можно продолжить.

В любом случае получаем, что работа алгоритма построения T_0 из интерливинга T на $(k+1)$-м шаге не доведена до конца. Но это значит, что если $T_0 = u_1 \ldots u_k v_1 \ldots$, а $\widehat{T} = u_1 \ldots u_k v_2 \ldots$, и v_1, v_2 — разные операторы, то T_0 не может быть получен из завершающего интерливинга T схемы G вышеописанным алгоритмом.

Таким образом, если T_0 — результат работы алгоритма, то либо T_0 — собственный префикс $\widehat{T}$, либо $\widehat{T}$ — собственный префикс T_0, либо $T_0 = \widehat{T}$. Так как T_0 — завершающий интерливинг, то T_0 не может быть собственным префиксом $\widehat{T}$. С другой стороны, пусть $\widehat{T}$ — собственный префикс T_0, $\widehat{T} = u_1 \ldots u_n$ и $T_0 = u_1 \ldots u_n u_{n+1} \ldots$. Это значит, что после выполнения оператора u_n не все процессы завершились или заблокированы и выполнение схемы G на каноническом плане (и построение интерливинга $\widehat{T}$) может быть продолжено. Таким образом, $\widehat{T} = T_0$.

Все вышеприведенные результаты не зависят от той части семантики операторов, которая отражает способ фиксации результата, а от свойств интерливингов, планов, и той части семантики операторов, которая определяет активность/неактивность процессов и возможность или невозможность оператора выполниться, являющиеся общими для обеих введенных разновидностей схем, поэтому все вышеперечисленные результаты верны и для пропозициональных схем.

В частности, верна следующая теорема.

Теорема 2. *Пусть H_1 и H_2 — корректные планы выполнения схемы G на функции разметки μ и на H_1 схема G завершает свою работу. Тогда G и на H_2 завершает работу, результаты вычисления (наборы деревьев) G на H_1 и H_2 эквивалентны и порождаемые при этом интерливинги получаются друг из друга перестановками из системы S.*

На основании приведенных теорем мы получаем возможность дать корректное определение результата вычисления ССВП и эквивалентности пары ССВП в интерпретационной и пропозициональной моделях.

Определение 26. *Результатом вычисления стандартной (пропозициональной) ССВП G для начального состояния W_0 в интерпретации I* (соответственно, *на функции разметки μ*) назовем результат вычисления G на каноническом плане $H(G, W_0, I)$ (соответственно, $H(G, \mu)$).

Определение 27. Две стандартные ССВП G_1 и G_2 *эквивалентны*, если для любой интерпретации I и любого начального состояния W_0 результаты вычисления этих схем совпадают, т. е. вычисление G_1 на каноническом плане завершается тогда и только тогда, когда вычисление G_2 на соответствующем этой схеме каноническом плане также завершается, и при этом результаты этих вычислений совпадают.

Определение 28. Два дерева вычислений $T_1 = (V_1, G_1, v_1, f_1)$ и $T_2 = (V_2, G_2, v_2, f_2)$ назовем *эквивалентными*, если они изоморфны как корневые помеченные деревья, т. е. существует отображение

$$\varphi\colon (V_1, G_1) \to (V_2, G_2)$$

со свойствами $\varphi(v_1) = v_2$ и $f_1(e) = f_2(\varphi(e))$ для любого $e \in G_1$.

Определение 29. Два набора деревьев вычислений $(T_1, \ldots, T_k)$ и $(S_1, \ldots, S_k)$ *эквивалентны*, если существует перестановка $\pi = (i_1, \ldots, i_k)$ такая, что для каждого $j \in \{1, \ldots, k\}$ деревья T_j и S_{i_j} эквивалентны.

Определение 30. Две схемы G_1 и G_2 *пропозиционально эквивалентны*, если для любой функции разметки μ схема G_1 останавливается на μ тогда и только тогда, когда схема G_2 останавливается на μ, и при этом их результаты эквивалентны.

§ 5. Корректность пропозициональной эквивалентности ССВП относительно интерпретационной эквивалентности

Пусть дана стандартная ССВП G. В начале § 3 была описана процедура перехода от схемы G к соответствующей ей пропозициональной схеме путем введения пропозициональных обозначений для операторов G. В этом параграфе будет исследована взаимосвязь между введенными выше способами определения эквивалентности ССВП и доказан следующий факт: если G_1, G_2 — стандартные ССВП и $\widehat{G}_1$, $\widehat{G}_2$ — соответствующие им пропозициональные схемы, то из пропозициональной эквивалентности $\widehat{G}_1$ и $\widehat{G}_2$ следует интерпретационная эквивалентность G_1 и G_2.

Пусть дана стандартная ССВП G. Способ перехода от нее к соответствующей ей пропозициональной схеме подразумевает формирование алфавита операторных символов **Op** — пропозициональных обозначений всех различных операторов и атомарных формул, встречающихся в G.

Пусть U — операторная цепочка в алфавите **Op**, $U = u_1 u_2 \ldots u_n$. Рассмотрим $U_i = U|\,\mathbf{Op}(\Gamma_i)$ для всех $i \in \{1, \ldots, k\}$ — проекции U на все множества $\mathbf{Op}(\Gamma_i)$ (т. е. $U|\,\mathbf{Op}(\Gamma_i)$ — последовательность операторов i-го процесса в U, взятых в том же порядке). Пусть $U_i = u_1^i \ldots u_{m_i}^i$ для всех $i \in \{1, \ldots, k\}$.

Рассмотрим схему G_U из k процессов, показанную на рис. 9. Применим к ней уже известный алгоритм построения вычисления результата пропозициональной схемы G_U на произвольной функции разметки. Поскольку в схеме G_U нет ветвлений, она выполняется одинаково на всех функциях разметки.

Если работа G_U завершается, то результат ее работы обозначим D. Тогда скажем, что система деревьев D *порождена операторной цепочкой* U, и обозначим это через $D = \mathbf{Tree}(U)$. Если же G_U не завершает работу, то неизбежно некоторые ее процессы блокируются; зацикливание процессов невозможно из-за отсутствия циклов в графах процессов G_U. Тогда скажем, что U *не порождает никакой системы деревьев.*

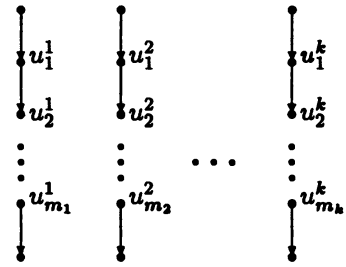


Рис. 9. Схема для построения системы деревьев по операторной цепочке

Пусть дана некоторая стандартная схема G и фиксирована некоторая интерпретация I. Пусть Tr — множество всех деревьев вычислений, помеченных операторными символами из алфавита $\mathbf{Op}(G)$, а Val — множество всех состояний памяти $X = \{x_1, \ldots, x_k\}$, где предметная область зафиксирована при задании интерпретации I.

Определение 31. Определим функцию ψ: $Tr \times Val \to Val$ индуктивно следующим образом ($W \in Val$):

— если T — начальное дерево вычисления, то $\psi(V, W) = W$;

— если $T = T_0 * a_j^i$ и a_j^i соответствует оператору $x := t$, то

$$\psi(T, W)(y) = \begin{cases} \psi(T_0, W)(y), & y \in X,\ y \neq x, \\ t_I(\psi(T_0, W)), & y = x; \end{cases}$$

— если $T = T_0 * s_j^i$, то $\psi(T, W) = \psi(T_0, W)$;

— если $T = T_0 * r_{qw}^i + T_1 * s_{qm}^j$, где r_{qw}^i соответствует оператору $\mathbf{receive}(q, x)$, а s_{qm}^j — оператору $\mathbf{send}(q, t)$, то

$$\psi(T, W)(y) = \begin{cases} \psi(T_0, W)(y), & y \in X,\ y \neq x, \\ t_I(\psi(T_1, W)), & y = x. \end{cases}$$

Функция ψ по начальному состоянию памяти и дереву дает результат — конечное состояние памяти, получаемое из начального состояния преобразованиями в соответствии со структурой дерева.

Продолжим ψ на наборы деревьев: пусть $(T_1, \ldots, T_k)$ — набор из k деревьев и $X = \bigcup X_i$, $i \in \{1, \ldots, k\}$, — разбиение памяти системы на память отдельных процессов. Тогда $\widehat{\psi}(T_1, \ldots, T_k, W)$ — состояние памяти X, определяемое следующим образом:

$$\widehat{\psi}(T_1, \ldots, T_k, W)(y) = \begin{cases} \psi(T_1, W)(y), & y \in X_1, \\ \psi(T_2, W)(y), & y \in X_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ \psi(T_k, W)(y), & y \in X_k. \end{cases}$$

Таким образом,

$$\widehat{\psi}(T_1, \ldots, T_k, W) = \psi(T_1, W) \times \psi(T_2, W) \times \ldots \times \psi(T_k, W),$$

где произведение $\times$ было введено в конце п. 3.3.

Рассмотрим схему G при некоторой интерпретации I и начальном состоянии памяти W_0. Пусть C — некоторый (не обязательно завершающий) интерливинг схемы G на (I, W_0), т. е. C порождается при работе G на (I, W_0) по некоторому плану H. *Результатом работы интерливинга* C (если он завершается), как и раньше, назовем состояние памяти G, полученное после завершения C и обозначаемое $\mathbf{res}(C, G, I, W_0)$.

Утверждение 9. *Пусть интерливинг C схемы G порождает набор деревьев* $\mathbf{Tree}(C) = (T_1, \ldots, T_k)$. *Тогда при фиксированной интерпретации I и начальном состоянии памяти W_0*

$$\widehat{\psi}(T_1, \ldots, T_k, W_0) = \mathbf{res}(C, G, I, W_0).$$

Доказательство. Проведем индукцию по длине интерливинга.

Если $C = e$ — пустой интерливинг, то он, в частности, порождается пустым планом и порождает систему деревьев $(V, \ldots, V)$, где V — начальное дерево вычисления (см. рис. 4, 1). Результат выполнения схемы на пустом плане есть начальное состояние памяти и по определению

$$\widehat{\psi}(V, \ldots, V, W_0) = W_0.$$

Предположим, что утверждение выполнено для всех интерливингов схемы G длины, не превосходящей $n-1$. Пусть $C = u_1 \ldots u_{n-1} u_n$ — интерливинг схемы G длины n, тогда $C_0 = u_1 \ldots u_{n-1}$ — также интерливинг схемы G. Пусть C порождает систему деревьев $(T_1, \ldots, T_k)$. Интерливинг C_0 также порождает некоторую систему деревьев, так как завершающий план выполнения G_{C_0} может быть получен из завершающего плана схемы G_C выбрасыванием последнего вхождения номера процесса, которому принадлежит u_n. Пусть C_0 порождает систему деревьев $(T_1^0, \ldots, T_k^0)$. Рассмотрим следующие случаи.

а) $u_n = a_j^i$, здесь a_j^i обозначает оператор $x := t$, где $x \in X_i$, $t \in \mathbf{Term}(\Gamma_i)$ (см. рис. 4, 2,*а*). Тогда из алгоритма построения $\mathbf{Tree}(C)$ и $\mathbf{Tree}(C_0)$ получаем

$$T_m = \begin{cases} T_m^0, & m \neq i, \\ T_i^0 * a_j^i, & m = i. \end{cases} \tag{1}$$

Пусть $H = n_1 \ldots n_m$ — м. п. план интерливинга C в схеме G. Так как $C = u_1 \ldots u_n$ и H — м. п. план для C, то после $(m-1)$-го шага работы алгоритма построения вычисления G на (I, W_0) в соответствии с $H_0 = n_1 \ldots n_{m-1}$ состояние схемы имеет вид $A = (r, W, z, C, Act, Stop)$, где r_i указывает на вершину в Γ_i, помеченную операторным символом a_j^i.

План H_0 порождает интерливинг $C_0 = u_1 \ldots u_{n-1}$ длины $n-1$, следовательно, по предположению индукции,

$$\widehat{\psi}(T_1^0, \ldots, T_k^0, W_0) = \mathbf{res}(C_0, G, I, W_0) = W.$$

При этом H_0 уже может не быть м. п. планом для C_0.

По алгоритму вычисления G на (I, W_0) в соответствии с H на m-м шаге получается состояние $A' = (r', W', z', C', Act', Stop')$, где $W' = \mathbf{res}(C, G, I, W_0)$ и для любого $y \in X$ выполнено

$$W'(y) = \begin{cases} W(y), & y \neq x, \\ t_I(W), & y = x. \end{cases} \tag{2}$$

Так как $T_j = T_j^0$ для всех $j \neq i$, то из (2) при всяком $y \in X_j$ выполнено

$$\widehat{\psi}(T_1, \ldots, T_k, W_0)(y) = \psi(T_j, W_0) = \psi(T_j^0, W_0)(y) = \\ = \mathbf{res}(C_0, G, I, W_0)(y) = W(y) = W'(y) = \mathbf{res}(C, G, I, W_0);$$

при всяком $y \in X_i$, $y \neq x$ из (1), (2) и определения функций $\widehat{\psi}$, ψ следует, что

$$\widehat{\psi}(T_1, \ldots, T_k, W_0)(y) = \psi(T_i, W_0)(y) = \psi(T_i^0 * a_j^i, W_0)(y) = \psi(T_i^0, W_0)(y) = \\ = \mathbf{res}(C_0, G, I, W_0)(y) = W(y) = W'(y) = \mathbf{res}(C, G, I, W_0);$$

при $y=x$ из определения функций $\widehat{\psi}$, ψ и (1), (2) следует, что

$$\widehat{\psi}(T_1, \ldots, T_k, W_0)(x) = \psi(T_i, W_0)(x) = \psi(T_i^0 * a_j^i, W_0)(x) =$$
$$= t_I(\psi(T_i^0, W_0)) = t_I(\widehat{\psi}(T_1^0, \ldots, T_k^0, W_0)) = t_I(\mathbf{res}(C_0, G, I, W_0)) =$$
$$= t_I(W) = W'(x) = \mathbf{res}(C, G, I, W_0)(x).$$

Таким образом, при всех $y \in X$ выполнено

$$\widehat{\psi}(T_1, \ldots, T_k, W_0)(y) = \mathbf{res}(C, G, I, W_0)(y).$$

Отсюда $\widehat{\psi}(T_1, \ldots, T_k, W_0) = \mathbf{res}(C, G, I, W_0)$.

б) $u_n = s_{jk}^i$, где s_{jk}^i обозначает некоторый оператор отправления сообщения в j-й канал, $s_j^i \in \mathbf{Op}(\Gamma_i)$ (см. рис. 4, 2,*а*). Тогда

$$T_m = \begin{cases} T_m^0, & m \neq i, \\ T_i^0 * s_j^i, & m = i. \end{cases} \tag{3}$$

Пусть $H = n_1 \ldots n_m$ — минимальный порождающий план для интерливинга C в схеме G. Так как C имеет вид $C = u_1 \ldots u_n$ и H — м. п. план, то после $(m-1)$-го шага работы алгоритма построения вычисления G на (I, W_0) в соответствии с планом H состоянием системы будет $(r, W, z, C, Act, Stop)$, где r_i — метка вершины, помеченная оператором s_{jk}^i. Следовательно, план $H_0 = n_1 \ldots n_{m-1}$ порождает $C_0 = u_1 \ldots u_{n-1}$ и по предположению индукции

$$\widehat{\psi}(T_1^0, \ldots, T_i^0, \ldots, T_k^0, W_0) = \mathbf{res}(C_0, G, I, W_0) = W.$$

По алгоритму построения вычисления G на (I, W_0) по плану H на m-м шаге из A получается состояние $A' = (r', W', z', C', Act', Stop')$, где $W' = \mathbf{res}(C, G, I, W_0)$. По определению результата работы интерливинга и, поскольку семантика оператора **send** такова, что его выполнение не приводит к изменению состояния памяти, $W'(x) = W(x)$ при всех $x \in X$.

Отсюда по предположению индукции, а также из определения функций $\widehat{\psi}$, ψ и (3) получаем, что для любых $j \in \{1, \ldots, k\}$ и $y \in X_j$

$$\widehat{\psi}(T_1, \ldots, T_k, W_0)(y) = \psi(T_j, W_0)(y) = \psi(T_j^0, W_0)(y) =$$
$$= \mathbf{res}(C_0, G, I, W_0) = W(y) = W'(y) = \mathbf{res}(C, G, I, W_0)(y).$$

Таким образом, $\widehat{\psi}(T_1, \ldots, T_k, W_0) = \mathbf{res}(C, G, I, W_0)$.

в) $u_n = r_{jk}^i$, где r_{jk}^i обозначает некоторый оператор **receive**(j, x) приема сообщения из j-го канала, $r_j^i \in \mathbf{Op}(\Gamma_i)$ (см. рис. 4, 2,*б*).

Пусть $H = n_1 \ldots n_m$ — м. п. план для C. Тогда, поскольку на m-м шаге был выполнен оператор r_{jk}^i, то перед m-м шагом в j-й канал было записано некоторое значение. Пусть $C = u_1 \ldots u_t \ldots u_n$, где u_t — оператор s_{jd}^l записи в j-й канал и между u_t и u_n нет операторов отправления сообщений в j-й канал. Тогда существует номер g, такой, что при работе алгоритма построения вычисления G на (I, W_0) в соответствии с планом H на g-м шаге выполняется оператор u_t, и план $H_1 = n_1 \ldots n_g$ — порождающий для интерливинга $C_1 = u_1 \ldots u_t$. Пусть $H_0 = n_1 \ldots n_{m-1}$ — план, порождающий интерливинг $C_0 = u_1 \ldots u_{n-1}$. Пусть на g-м шаге работы алгоритма построения вычисления G на (I, W_0) в соответствии с планом H из состояния $A_1 = (r_1, W_1, z_1, C_1, Act_1, Stop_1)$ получено состояние $A_1' = (r_1', W_1', z_1', C_1', Act_1', Stop_1')$ выполнением оператора u_t, а после $(m-1)$-го шага работы G на H было получено состояние $A_2 = (r_2, W_2, z_2, C_2, Act_2, Stop_2)$.

По предположению индукции если интерливинг C_1 порождает систему деревьев $(T_1^1, \ldots, T_k^1)$, а C_0 порождает систему деревьев $(T_1^0, \ldots, T_k^0)$, то

$$\widehat{\psi}(T_1^1, \ldots, T_k^1, W_0) = \mathbf{res}(C_1, G, I, W_0),$$
$$\widehat{\psi}(T_1^0, \ldots, T_k^0, W_0) = \mathbf{res}(C_0, G, I, W_0).$$

Рассмотрим m-й шаг построения вычисления G на (I, W_0) в соответствии с H. На нем из состояния A_2 получено новое состояние

$$A_2' = (r_2', W_2', z_2', C_2', Act'_2, Stop'_2).$$

В соответствии с семантикой оператора **receive**

$$W_2'(y) = \begin{cases} W_2(y), & y \neq x, \\ t_I(W_1'), & y = x. \end{cases} \tag{4}$$

В соответствии с семантикой оператора **send**

$$W_1'(x) = W_1(x) \text{ при всех } x \in X, \tag{5}$$

$$T_m = \begin{cases} T_m^0, & m \neq i, \\ T_i^0 * r_j^i + T_l^1 * s_{jd}^l, & m = i. \end{cases} \tag{6}$$

Тогда по предположению индукции, определению функций ψ, $\widehat{\psi}$, а также из (4), (5), (6) для всех $y \in X$, $y \neq x$ выполнено

$$\widehat{\psi}(T_1, \ldots, T_k, W_0)(y) = \psi(T_j, W_0)(y) = \psi(T_j^0, W_0)(y) =$$
$$= \mathbf{res}(C_0, G, I, W_0)(y) = W_2(y) = W_2'(y) = \mathbf{res}(C, G, I, W_0);$$

для $y = x$ выполнено

$$\widehat{\psi}(T_1, \ldots, T_k, W_0)(x) = \psi(T_i, W_0)(x) = t_I(\psi(T_l^1, W_0)) =$$
$$= t_I(\widehat{\psi}(T_1^1, \ldots, T_l^1, \ldots, T_k^1, W_0)) = t_I(\mathbf{res}(C_1, G, I, W_0)) =$$
$$= t_I(W_1') = W_2'(x) = \mathbf{res}(C, G, I, W_0).$$

Таким образом, для всех $x \in X$ выполнено

$$\widehat{\psi}(T_1, \ldots, T_k, W_0)(x) = \mathbf{res}(C, G, I, W_0)(x).$$

Проведенный разбор вариантов показывает, что каким бы ни был последний оператор u_m в интерливинге C, всегда

$$\widehat{\psi}(T_1, \ldots, T_k, W_0) = \mathbf{res}(C, G, I, W_0).$$

Утверждение 10. *Пусть G_1 и G_2 — две стандартные схемы в одном базисе B и зафиксирована некоторая интерпретация I базиса B. Пусть C_1 — интерливинг схемы G_1, а C_2 — интерливинг схемы G_2. Пусть* $\mathbf{Tree}(C_1) = \mathbf{Tree}(C_2)$. *Тогда для любого начального состояния W_0 выполнено*

$$\mathbf{res}(C_1, G_1, I, W_0) = \mathbf{res}(C_2, G_2, I, W_0).$$

Доказательство. По утверждению 9 выполняется

$$\mathbf{res}(C_1, G_1, I, W_0) = \widehat{\psi}(\mathbf{Tree}(C_1), W_0),$$
$$\mathbf{res}(C_2, G_2, I, W_0) = \widehat{\psi}(\mathbf{Tree}(C_2), W_0),$$

но $\mathbf{Tree}(C_1) = \mathbf{Tree}(C_2)$, следовательно, $\widehat{\psi}(\mathbf{Tree}(C_1), W_0) = \widehat{\psi}(\mathbf{Tree}(C_2), W_0)$ и $\mathbf{res}(C_1, G_1, I, W_0) = \mathbf{res}(C_2, G_2, I, W_0)$.

Определение 32. Набор деревьев $(T_1, \ldots, T_k)$ назовем *порождаемым*, если существует интерливинг C такой, что $(T_1, \ldots, T_k) = \mathbf{Tree}(C)$.

Очевидно, что состояния процессов пропозициональной ССВП, возникающие в процессе функционирования схемы, являются порождаемыми наборами деревьев.

Пусть $P_i = \{p_j^i\}$, где $j \in \{1, \ldots, m_i\}$ — все логические условия i-го процесса для всех $i \in \{1, \ldots, k\}$. Каждое логическое условие p_j^i процесса Γ_i схемы $\widehat{G}$ обозначает некоторую атомарную формулу $\pi_q^i(x_1, \ldots, x_n)$ — пометку распознавателя из процесса Γ_i в схеме G, причем в списке аргументов присутствуют лишь переменные из X_i.

Пусть фиксирована некоторая интерпретация I базиса B и некоторое начальное состояние W_0: $X \to D$. Обозначим через $p_j^i(W)$ значение логического условия p_j^i на состоянии памяти W: $X \to D$ и интерпретации I, равное $\pi_{q_j}^i(W(x_1), \ldots, W(x_n))$; здесь $W(x_k)$ — значение переменной x_k на состоянии памяти W, а $\pi_{q_j}^i$: $D^n \to \{0, 1\}$ — отношение на D^n, сопоставленное предикатному символу π_q^i.

Обозначим через $P(W)$ набор из $\{0, 1\}^{m_1 + m_2 + \ldots + m_k}$ значений истинности всех логических условий процесса на (I, W), равный

$$\{p_1^1(W), \ldots, p_{m_1}^1(W), p_1^2(W), \ldots, p_{m_2}^2(W), \ldots, p_1^k(W), \ldots, p_{m_k}^k(W)\}.$$

Определим функцию разметки $\widehat{\mu}(I, W_0)$ на порождаемых наборах деревьев следующим образом: если $(T_1, \ldots, T_k)$ — порождаемый интерливингом C набор деревьев, то положим $\widehat{\mu}(T_1, \ldots, T_k) = P(\widehat{\psi}(\mathbf{Tree}(C), W_0))$; на остальных наборах деревьев определим $\widehat{\mu}$ произвольным образом, следя за соблюдением ограничений из определения функции разметки. Поскольку необходимо вычислять значения функции разметки лишь на наборах деревьев, возникающих при построении вычисления ССВП, которые являются порождаемыми, конкретные значения функции разметки на непорождаемых наборах деревьев не важны.

Определение $\widehat{\mu}(I, W_0)$ корректно: если C_1 и C_2 — два разных интерливинга, порождающих одну систему деревьев $(T_1, \ldots, T_k)$, то $\mathbf{Tree}(C_1) = \mathbf{Tree}(C_2) = (T_1, \ldots, T_k)$ и $\widehat{\psi}(\mathbf{Tree}(C_1), W_0) = \widehat{\psi}(\mathbf{Tree}(C_2), W_0)$. Таким образом, определение $\widehat{\mu}$ на порождаемом наборе деревьев не зависит от выбора интерливинга, порождающего эту систему деревьев.

Пусть выполнение пропозициональной ССВП G по некоторому плану H на функции разметки μ приводит к построению интерливинга C. Как и раньше, состояние процессов, которое получено при работе G на μ в соответствии с некоторым планом, порождающим C после шага, на котором к интерливингу был приписан последний операторный символ, будем называть *результатом интерливинга* C *при работе* G *на* μ и обозначать $\mathbf{res}(C, G, \mu)$.

Утверждение 11. *Пусть* $\widehat{G}$ *— пропозициональная схема,* μ *— произвольная функция разметки. Пусть* $\widehat{G}$ *при работе на* μ *по некоторому плану* H *завершает свое выполнение с результатом* $(T_1, \ldots, T_k) = \widehat{G}(\mu)$, *порождая при этом интерливинг* C. *Тогда*

$$\mathbf{Tree}(C) = \widehat{G}(\mu).$$

Доказательство. Пусть H — некоторый завершающий план схемы G на функции разметки μ. Доказательство проводится индукцией по длине интерливинга.

Если $\widehat{G}$ такова, что на μ и плане H схема $\widehat{G}$ завершает работу и порождает интерливинг $C = e$, то $\mathbf{Tree}(C) = (V, \ldots, V)$, где V — начальное дерево вычисления.

Результатом вычисления $\widehat{G}$ на μ в соответствии с планом H является набор деревьев $\widehat{G}(\mu) = (V, \ldots, V)$, так как не было выполнено ни одного оператора, приводящего к модификации начального состояния процессов. Таким образом, $\mathbf{Tree}(C) = \widehat{G}(\mu)$.

Пусть известно, что для любого (не обязательно завершающего) плана $\widehat{H}$ выполнения $\widehat{G}$ на μ, порождающего интерливинг $\widehat{C} = u_1 \ldots u_m$ длины m, где $m \leqslant n - 1$, выполнено $\mathbf{Tree}(\widehat{C}) = \mathbf{res}(\widehat{C}, \widehat{G}, \mu)$. Докажем, что и для любого интерливинга C схемы G длины n выполнено $\mathbf{Tree}(C) = \mathbf{res}(C, \widehat{G}, \mu)$.

Пусть H — план выполнения схемы $\widehat{G}$ на μ, порождающий интерливинг $C = u_1 \ldots u_n$ длины n. Пусть $H = n_1 \ldots n_m \ldots n_t$, где на шаге с номером n_m был выполнен последний оператор u_n из C. Тогда план $H_0 = n_1 \ldots n_{m-1}$ порождает интерливинг $C_0 = u_1 \ldots u_{n-1}$ и, по предположению индукции, $\mathbf{Tree}(C_0) = \mathbf{res}(C_0, \widehat{G}, \mu)$.

Интерливингу C_0 соответствует схема G_{C_0}, а интерливингу C_1 — схема G_{C_1} (как в алгоритме построения системы деревьев по интерливингу).

При этом, поскольку C_0 и C — интерливинги схемы $\widehat{G}$, выполнение схем G_{C_0} и G_C доходит до конца и порождаются системы деревьев $(T_1^0, \ldots, T_k^0)$ и $(T_1, \ldots, T_k)$. Действительно, если H — план выполнения $\widehat{G}$ на μ, при котором порождается интерливинг C, то план, получающийся из H выбрасыванием шагов, на которых выбранный в $\widehat{G}$ процесс проходит через распознаватели, — завершающий план для G_C, значит, выполнение схемы G_C доходит до конца и порождает некоторый набор деревьев.

Равенство $\mathbf{Tree}(C) = \mathbf{res}(C, \widehat{G}, \mu)$ следует из предположения индукции и того, что алгоритм построения $\mathbf{Tree}(C)$ и алгоритм функционирования $\widehat{G}$ на μ, в результате работы которого формируется $\mathbf{res}(C, \widehat{G}, \mu)$, модифицируют соответственно $\mathbf{Tree}(C_0)$ и $\mathbf{res}(C_0, \widehat{G}, \mu)$ одинаковым образом в зависимости от вида оператора u_m. Таким образом, $\mathbf{Tree}(C) = \mathbf{res}(C, \widehat{G}, \mu)$.

Но если C — завершающий интерливинг схемы $\widehat{G}$, то поскольку результат функционирования пропозициональной ССВП на μ не зависит от выбора плана выполнения (и порождаемого им интерливинга C), $\mathbf{res}(C, \widehat{G}, \mu) = \widehat{G}(\mu)$, следовательно, $\mathbf{Tree}(C) = \widehat{G}(\mu)$.

Утверждение 12. *Пусть G — стандартная схема, заданы некоторая интерпретация базиса I, начальное состояние W_0, и G заканчивает работу на (I, W_0) по некоторому плану H, порождая при этом интерливинг C. Тогда соответствующая G пропозициональная схема $\widehat{G}$, выполняясь на функции разметки $\widehat{\mu}(I, W_0)$ по этому же плану H, заканчивает работу, порождая тот же интерливинг C, и наоборот, если $\widehat{G}$, выполняясь на $\widehat{\mu}(I, W_0)$ по плану H, заканчивает работу, порождая интерливинг C, то схема G, выполняясь на (I, W_0) по плану H, заканчивает работу, порождая тот же интерливинг C.*

Доказательство. Пусть стандартная схема G завершает работу на (I, W_0). Пусть H — завершающий план выполнения G на (I, W_0). Покажем, что H — завершающий план выполнения $\widehat{G}$ на $\widehat{\mu}(I, W_0)$. Так как $\widehat{G}$ получена из G просто заменой меток вершин с операторов на операторные символы и поскольку правила выполнения и занесения операторов в интерливинг для стандартных и пропозициональных схем одинаковы, то достаточно показать, что пути выполнения G на (I, W_0) и $\widehat{G}$ на $\widehat{\mu}(I, W_0)$ совпадают, т. е. что при прохождении через распознаватели в обоих случаях выбирается одна и та же ветвь дальнейшего исполнения.

Пусть на плане H на шаге с номером n_k в схеме G проходится распознаватель, которому приписана атомарная формула $\pi_j^i(x_1, \ldots, x_n)$. Это значит, что в схеме $\widehat{G}$ на плане H на шаге с номером n_k проходится распознаватель, помеченный символом p_q^i, где p_q^i соответствует $\pi_j^i(x_1, \ldots, x_n)$. Тогда если перед шагом n_k в G и $\widehat{G}$ был построен один и тот же интерливинг C_0, то, с одной стороны, $\mathbf{Tree}(C_0) = \mathbf{res}(C_0, \widehat{G}, \mu(I, W_0))$ — состояние процессов, которое они имеют перед шагом n_k, значит, выбор ветви выполнения $\widehat{G}$ при прохождении p_q^i зависит от значения $\widehat{\mu}(I, W_0)(T_1, \ldots, T_k)(p_q^i)$, где $(T_1, \ldots, T_k)$ — набор деревьев, порожденных при построении интерливинга C_0, т. е. $(T_1, \ldots, T_k) = \mathbf{Tree}(C_0)$. Поскольку $(T_1, \ldots, T_k)$ порождается интерливингом C_0, то $\widehat{\mu}(I, W_0)(T_1, \ldots, T_k)$ определена. С другой стороны, известно, что $\widehat{\psi}$ строит результат выполнения интерливинга C_0 на паре (I, W_0) по дереву $\mathbf{Tree}(C_0)$, т. е.

$$\widehat{\psi}(\mathbf{Tree}(C_0), W_0) = \mathbf{res}(C_0, G, I, W_0). \tag{7}$$

По алгоритму построения вычисления G на (I, W_0), выбор ветви для продолжения определяется значением

$$\pi_{jI}^i(\mathbf{res}(C_0, G, I, W_0)(x_1), \mathbf{res}(C_0, G, I, W_0)(x_2), \ldots, \mathbf{res}(C_0, G, I, W_0)(x_n)),$$

но из (7) и определения $\widehat{\mu}(I, W_0)$ следует, что

$$\widehat{\mu}(\mathbf{Tree}(C_0), W_0)(p_q^i) = P(\widehat{\psi}(\mathbf{Tree}(C_0), W_0))(p_q^i) = p_q^i(\widehat{\psi}(\mathbf{Tree}(C_0), W_0)) =$$
$$= \pi_{jI}^i(\mathbf{res}(C_0, G, I, W_0)(x_1), \mathbf{res}(C_0, G, I, W_0)(x_2), \ldots, \mathbf{res}(C_0, G, I, W_0)(x_n)).$$

Таким образом, в любом случае для продолжения берется одна и та же ветвь. Это значит, что пути выполнения G на (I, W_0) и $\widehat{G}$ на $\widehat{\mu}(I, W_0)$ в соответствии с планом H совпадают, и так как H — завершающий план для G, то отсюда следует, что интерливинги, порождаемые H при работе G на (I, W_0) и при работе $\widehat{G}$ на $\widehat{\mu}(I, W_0)$ совпадают и H — завершающий интерливинг схемы $\widehat{G}$.

Доказательство в обратную сторону проводится аналогично.

Утверждение 13. *Для произвольных интерпретации I, начального состояния памяти W_0 и стандартной ССВП G если $\widehat{G}$ — соответствующая ей пропозициональная схема, G заканчивает работу на (I, W_0) и при этом C — некоторый ее завершающий интерливинг, то* $\mathbf{Tree}(C) = \widehat{G}(\widehat{\mu}(I, W_0))$.

Доказательство. Так как G останавливается на (I, W_0), то существует завершающий план H выполнения G на (I, W_0), порождающий интерливинг C. Но по утверждению 12 план H — завершающий план выполнения $\widehat{G}$ на $\widehat{\mu}(I, W_0)$ и при этом $\widehat{G}$ на $\widehat{\mu}(I, W_0)$, работая в соответствии с планом H, порождает тот же интерливинг C. Из утверждения 11 следует, что $\mathbf{Tree}(C) = \widehat{G}(\widehat{\mu}(I, W_0))$.

Теорема 3. *Пусть G_1 и G_2 — две стандартные схемы в общем базисе B, $\widehat{G}_1$ и $\widehat{G}_2$ — соответствующие им пропозициональные схемы. Тогда если $\widehat{G}_1$ и $\widehat{G}_2$ пропозиционально эквивалентны, то G_1 и G_2 эквивалентны в стандартном смысле.*

Доказательство. Пусть заданы произвольные интерпретация I базиса B и начальное состояние W_0 памяти схемы. Пусть схема G_1 останавливается на (I, W_0) и C_1 — один из завершающих интерливингов G_1.

Пусть функция разметки $\widehat{\mu}$ построена по (I, W_0) как описано выше. Тогда, по утверждению 12, $\widehat{G}_1$ останавливается на $\widehat{\mu}$, а, значит, и $\widehat{G}_2$ останавливается на $\widehat{\mu}$, порождая при этом некоторый интерливинг C_2. По утверждению 12 схема G_2 останавливается на (I, W_0) и C_2 — один из ее завершающих интерливингов. Из пропозициональной эквивалентности $\widehat{G}_1$ и $\widehat{G}_2$ следует, что $\widehat{G}_1(\widehat{\mu}) = \widehat{G}_2(\widehat{\mu})$. Тогда, по утверждению 13, $\mathbf{Tree}(C_1) = \widehat{G}_1(\widehat{\mu})$, $\mathbf{Tree}(C_2) = \widehat{G}_2(\widehat{\mu})$, а, значит, $\mathbf{Tree}(C_1) = \mathbf{Tree}(C_2)$. Но тогда, по утверждению 10, $\mathbf{res}(C_1, G_1, I, W_0) = \mathbf{res}(C_2, G_2, I, W_0)$. Таким образом, G_2 на (I, W_0) также останавливается и результаты работы G_1 и G_2 на интерпретации I и начальном состоянии W_0 совпадают. Теорема доказана.

СПИСОК ЛИТЕРАТУРЫ

1. Ершов А. П. Об операторных схемах Янова // Проблемы кибернетики. Вып. 20. — М.: Наука, 1968. — С. 181–200.
2. Котов В. Е., Сабельфельд В. К. Теория схем программ. — М.: Наука, 1991.
3. Подловченко Р. И. Полугрупповые модели программ // Программирование. — 1981. — № 4. — С. 12–23.
4. Подловченко Р. И. Моделирование программ схемами и построение систем преобразований схем // Кибернетика. — 1982. — № 6. — С. 28–35.
5. Янов Ю. И. О логических схемах алгоритмов // Проблемы кибернетики. Вып. 1. — М.: Физматгиз, 1958. — С. 75–127.
6. Karp R. M., Miller R. E. Parallel program schemata // J. Computer and System Sci. — 1969. — V. 3, № 3. — P. 361–385.
7. Luckhem D. C., Park D. M., Paterson M. S. On formalized computer programs // J. Computer and System Sci. — 1970. — V. 4, № 3. — P. 220–249.
8. Mazurkiewicz A. Trace theory // Lect. Notes Comp. Sci. — 1986. — V. 255. — P. 279–324.
9. Owicki S., Gries D. Verifying properties of parallel programs: an axiomatic approach // Comm. ACM. — 1976. — V. 6, № 4. — P. 279–285.

Поступило в редакцию 9 IX 1999