Стандартные схемы программ

Пусть задано бесконечное множество предметных переменных

$Var = \{x_1, x_2, \ldots, x_n, \ldots\}$,

а также сигнатура $\sigma = (Const, Func, Pred)$ первого порядка, состоящая из

- множество констант $Const = \{c_1, c_2, \ldots, \}$,
- множество функциональных символов $Func = \{f_1^{(r_1)}, f_2^{(r_2)}, \ldots\}$,
- множество предикатных символов $Pred = \{P_1^{(s_1)}, P_2^{(s_2)}, \ldots\}$.

Над множеством переменных Var и сигнатурой σ строятся

- множество термов $Terms(Var, \sigma) = \{t_1, t_2, \ldots\}$ и
- множество атомарных формул (атомов) $Atoms(Var, \sigma) = \{A_1, A_2, \ldots\}$.

Стандартные схемы программ

Терм — это всякое выражение, которое может быть построено по следующим правилам:

- всякая переменная x из множества Var является термом;
- всякая константа c из множества $Const$ является термом;
- если $t_1, t_2, \ldots, t_n$ — это набор термов, а $f^{(n)}$ — это n-местный функциональный символ из множества $Func$, то выражение $f(t_1, t_2, \ldots, t_n)$ является термом.

Атомарной формулой (атомом) называется всякое выражение вида $P(t_1, t_2, \ldots, t_m)$, где $P^{(m)}$ — это m-местный предикатный символ из множества $Pred$, а $t_1, t_2, \ldots, t_m$ — это набор термов.

Стандартные схемы программ

Стандартные схемы программ строятся из операторов следующих пяти типов:

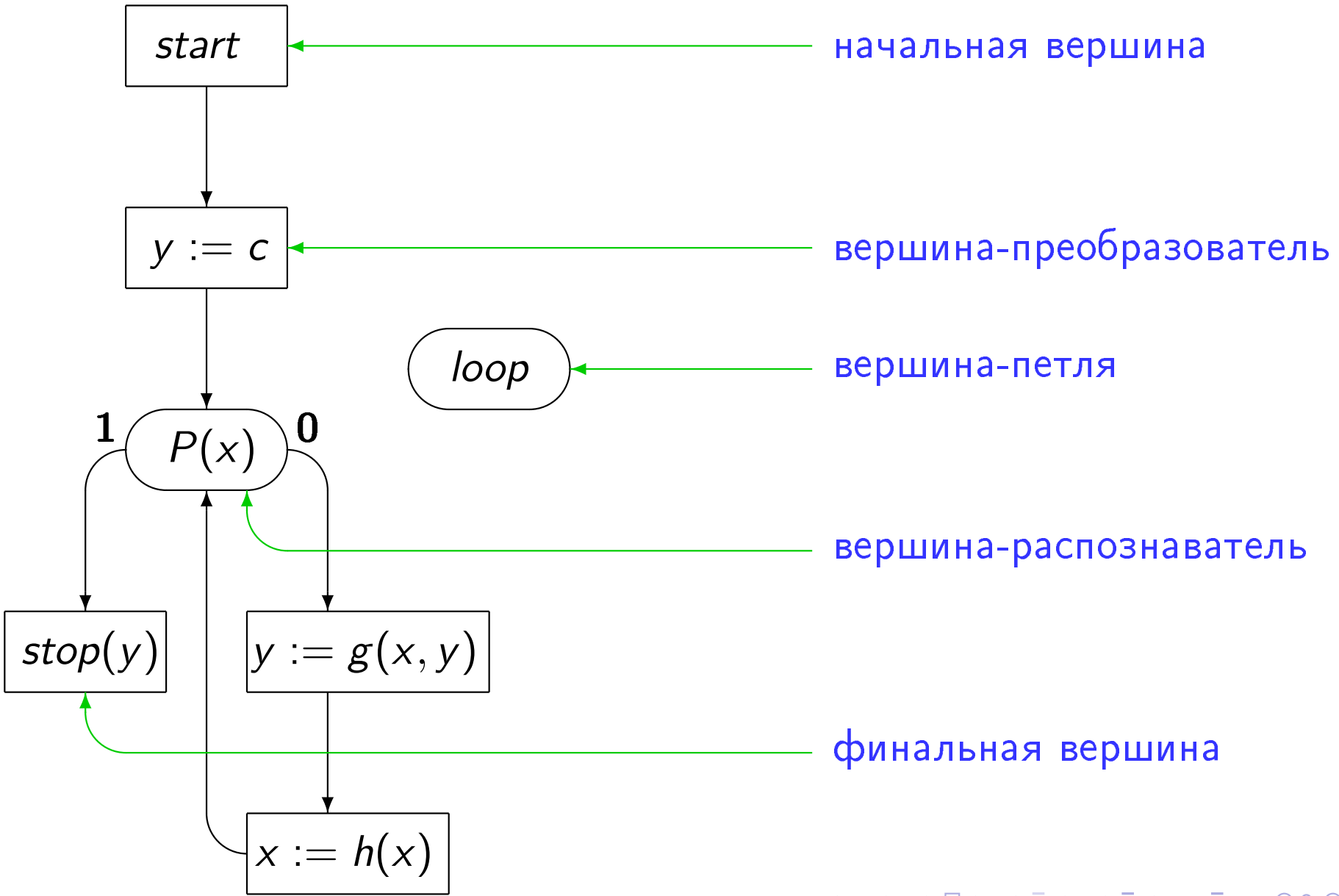
1. начальный оператор — выражение вида $start$;
2. финальный оператор — выражение вида $stop(t_{j_1}, \ldots, t_{j_m})$, где $t_{j_1}, \ldots, t_{j_m}$ — термы из множества $Terms(Var, \sigma)$;
3. оператор присваивания — выражение вида $x := t$, где $x \in Var$ и $t \in Terms(Var, \sigma)$;
4. тест (или, условие) — атомарная формула из множества $Atoms(Var, \sigma)$;
5. оператор петли — выражение $loop$.

Стандартные схемы программ

Стандартная схема программ — это конечный размеченный ориентированный граф, вершины которого разделены на пять классов:

1. Начальная вершина . В схеме имеется в точности одна начальная вершина. Она помечена начальным оператором. Из нее исходит одна дуга. Она не имеет входящих дуг.
2. Финальная вершина . В схеме может быть несколько финальных вершин, и каждая из них помечена финальным оператором и не имеет ни одной исходящей дуги.
3. Вершина-преобразователь . Одна помечена оператором присваивания, и из нее исходит в точности одна дуга.
4. Вершина-распознаватель . Она помечена тестом; из нее исходят две дуги, одна из которых помечана символом 0 (0-дуга), а другая — символом 1 (1-дуга).
5. Вершина-петля . Она помечена оператором петли и не имеет исходящих дуг.

Стандартные схемы программ



Стандартные схемы программ

Вычисления и эквивалентность стандартных схем программ, подобно отношению выполнимости для формул логики предикатов, определяются в интерпретациях первого порядка.

Интерпретация сигнатуры σ — это алгебраическая система $I = (D, \overline{Const}, \overline{Func}, \overline{Pred})$, в которой

- D — область интерпретации (область значений переменных и констант, область определения функций и отношений);
- $\overline{Const} : Const \to D$ — оценка констант;
- $\overline{Func} : Func \to (D^n \to D)$ — оценка функциональных символов;
- $\overline{Pred} : Pred \to (D^m \to \{\mathbf{0}, \mathbf{1}\})$ — оценка предикатных символов.

Интерпретацию I можно воспринимать как сменную библиотеку встроенных функций (алгебраических операций) и отношений заданной сигнатуры σ , в которой строятся программы.

Стандартные схемы программ

Оценкой переменных в интерпретации I называется всякое отображение $\xi : Var \to D$.

Значения термов $t(x_1, \ldots, x_k)[\xi]_I$ и атомарных формул $A(x_1, \ldots, x_k)[\xi]_I$ в интерпретации I на заданной оценке переменных ξ определются обычным образом индукцией по структуре термов и атомов:

- если $t(x_1, \ldots, x_k) = x_i$, то $t(x_1, \ldots, x_k)[\xi]_I = \xi(x_i)$;
- если $t(x_1, \ldots, x_k) = c$, то $t(x_1, \ldots, x_k)[\xi]_I = \bar{c}$;
- если $t(x_1, \ldots, x_k) = f(t_1, \ldots, t_n)$, то $t(x_1, \ldots, x_k)[\xi]_I = \bar{f}(t_1[\xi]_I, \ldots, t_n[\xi]_I)$;
- если $P(t_1, t_2, \ldots, t_m)$ — атомарная формула, то $P(t_1, t_2, \ldots, t_m)[\xi]_I = \bar{P}(t_1[\xi]_I, \ldots, t_m[\xi]_I)$.

Стандартные схемы программ

Переменные в стандартных схемах программ — это простейшие структуры данных, и оценки переменнных — это состояния памяти в вычислениях схем программ.

Для произвольных оценки переменных $\xi : Var \to D$, переменной $y, y \in Var$, и элемента $d, d \in D$, обозначим записью $\xi[y \leftarrow d]$ оценку переменных ξ' , которая определяется соотношениями

$$\xi'(x) = \begin{cases} \xi(x), & \text{если } x \neq y, \\ d, & \text{если } x = y. \end{cases}$$

Иными словами, $\xi[y \leftarrow d]$ — это состояние памяти, которое образуется из состояния памяти ξ в том случае, когда переменная y принимает значение d .

Стандартные схемы программ

Вычисления стандартных схем программ определяются в интерпретациях на оценках переменных.

Пусть задана стандартная схема программ π с множеством вершин V , интерпретация I и оценка переменных ξ в этой интерпретации.

Тогда вычислением схемы программ π в интерпретации I на начальном состоянии памяти ξ называется максимальная последовательность пар

$$comp(\pi, I, \xi) = (v_0, \xi_0), (v_1, \xi_1), \ldots, (v_i, \xi_i), (v_{i+1}, \xi_{i+1}), \ldots$$

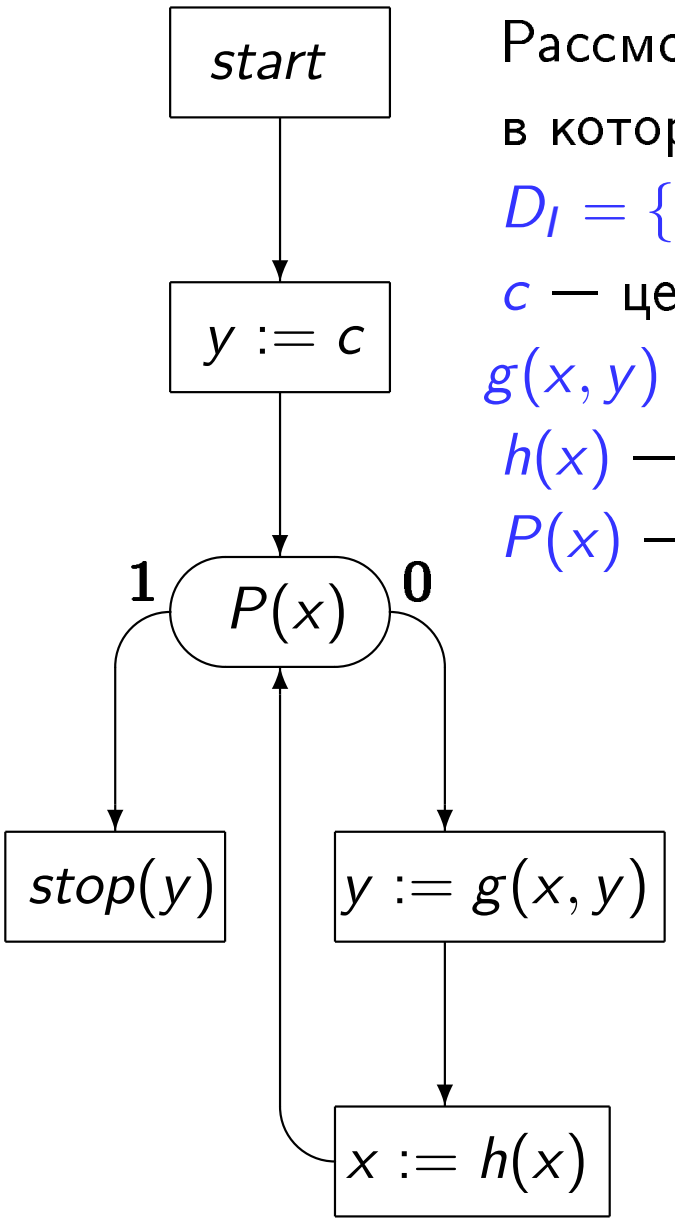
удовлетворяющая следующим требованиям:

1. v_0 — начальная вершина, $\xi_0 = \xi$;
2. для каждого $i, i \geq 0$, в зависимости от типа вершины v_i справедливы соотношения

Стандартные схемы программ

- если v_i — начальная вершина, то v_{i+1} — это вершина-преемник вершины v_i , а $\xi_{i+1} = \xi_i$;
- если v_i — финальная вершина, которой приписан оператор $stop(t_1, \ldots, t_m)$, то вычисление $comp(\pi, I, \xi)$ завершается парой (v_i, ξ_i) , и набор $val(\pi, I, \xi) = (t_1[\xi_i]_I, \ldots, t_m[\xi_i]_I)$ считается результатом вычисления ;
- если v_i — вершина-преобразователь, которой приписан оператор присваивания $x := t(x_1, \ldots, x_k)$, то v_{i+1} — это преемник вершины v_i , а $\xi_{i+1} = \xi_i[x \leftarrow t[\xi_i(x_1), \ldots, \xi_i(x_k)]_I]$;
- если v_i — вершина-распознаватель, которой приписан тест $A(x_1, \ldots, x_k)$, то v_{i+1} — это δ -преемник вершины v_i , где $\delta = A[\xi_i(x_1), \ldots, \xi_i(x_k)]_I$, а $\xi_{i+1} = \xi_i$;
- если v_i — вершина-петля, то $(v_{i+1}, \xi_{i+1}) = (v_i, \xi_i)$ (т.е. вычисление зацикливается).

Стандартные схемы программ



Рассмотрим арифметическую интерпретацию,

в которой

$D_I = \{0, 1, 2, \dots\}$

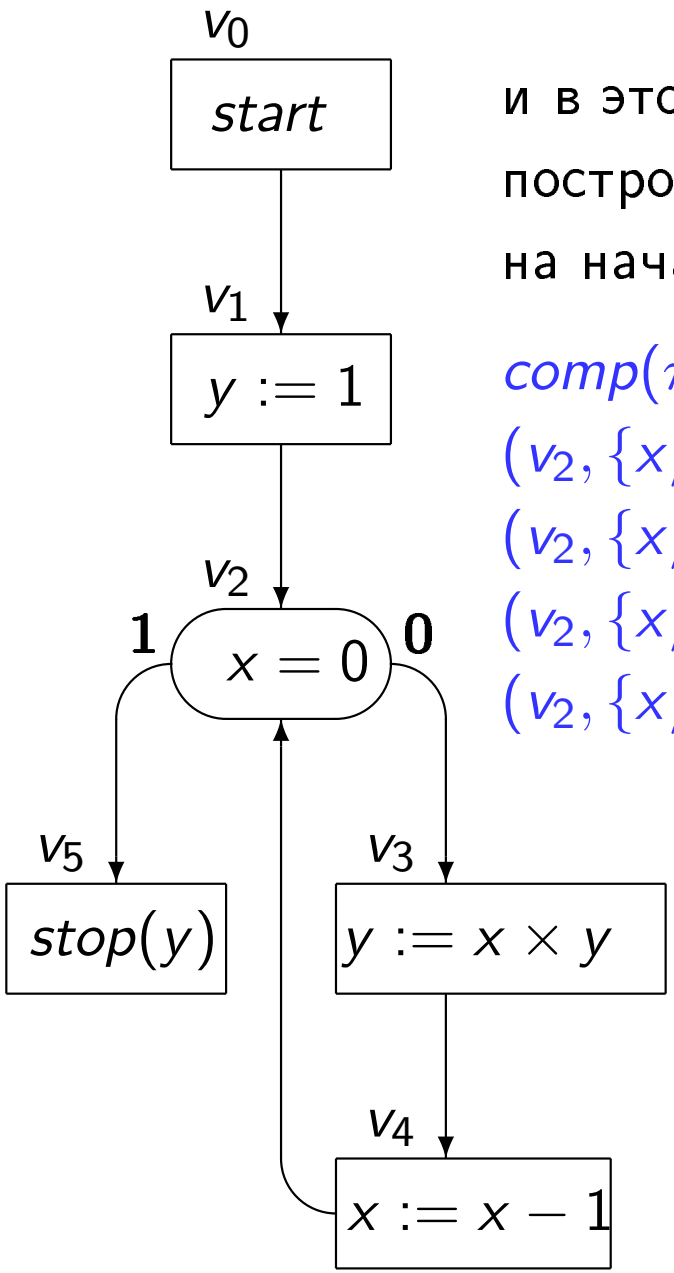
c — целое число 1 ,

$g(x, y)$ — функция умножения целых чисел: $x \times y$,

$h(x)$ — функция вычитания 1: $x - 1$,

$P(x)$ — предикат сравнения с нулем: $x = 0$.

Стандартные схемы программ



и в этой интерпретации

построим вычисление программы

на начальном состоянии памяти $\xi = \{x/3, y/7\}$:

$comp(\pi, I, \xi) = (v_0, \{x/3, y/7\}), (v_1, \{x/3, y/7\}),$
$(v_2, \{x/3, y/1\}), (v_3, \{x/3, y/1\}), (v_4, \{x/3, y/3\}),$
$(v_2, \{x/2, y/3\}), (v_3, \{x/2, y/3\}), (v_4, \{x/2, y/6\}),$
$(v_2, \{x/1, y/6\}), (v_3, \{x/1, y/6\}), (v_4, \{x/1, y/6\}),$
$(v_2, \{x/0, y/6\}), (v_5, \{x/1, y/6\}).$

$val(\pi, I, \xi) = \{y/6\}$.

LARGE Стандартные схемы программ

Всякое вычисление стандартной схемы программ соответствует некоторому ориентированному маршруту (конечному или бесконечному) в схеме программ.

Например, в рассмотренной ранее схеме программ вычислению

$$comp(\pi, I, \xi) = \begin{array}{l}(v_0, \xi),(v_1, \xi),(v_2, \xi_1),(v_3, \xi_1),(v_4, \xi_2),(v_2, \xi_3),(v_3, \xi_3),\\ (v_4, \xi_4),(v_2, \xi_5),(v_3, \xi_5),(v_4, \xi_6),(v_2, \xi_7),(v_3, \xi_7)\end{array}$$

соответствует ориентированный маршрут

$$trace = v_0 \to v_1 \to v_2 \to v_3 \to v_4 \to v_2 \to v_3 \to v_4 \to v_2 \to v_3 \to v_4 \to v_2 \to v_5$$

Будем говорить, что маршрут $trace$ реализуется вычислением $comp(\pi, I, \xi)$ стандартной схемы программ π .

Схема программ называется свободной , если каждый маршрут, ведущий из начальной вершины схемы в ее финальную вершину, реализуется некоторым вычислением этой схемы

Проблема эквивалентности для стандартных схем программ

Стандартные схемы программ π_1 и π_2 называются функционально эквивалентными (обозначается $\pi_1 \sim \pi_2$), если для любой интерпретации I и для любого начального состояния памяти ξ верно одно из двух:

- оба вычисления $comp(\pi_1, I, \xi)$ и $comp(\pi_2, I, \xi)$ бесконечны (зацикливаются);
- оба вычисления $comp(\pi_1, I, \xi)$ и $comp(\pi_2, I, \xi)$ завершаются, и результаты этих вычислений одинаковы $val(\pi_1, I, \xi) = val(\pi_2, I, \xi)$.

Проблема функциональной эквивалентности для стандартных схем программ состоит в том, чтобы для произвольной заданной пары стандартных схем программ π_1 и π_2 выяснить, являются ли эти схемы программ функционально эквивалентными $\pi_1 \sim \pi_2$?

Описание стандартных схем программ при помощи подстановок

Вначале мы немного изменим облик стандартной схемы программ, сделав ее более похожей на размеченную систему переходов, используемую для описания конечных автоматов.

Состояниями такой размеченной системы переходов будут вершины-распознаватели, а также начальная и финальная вершины; дуги (переходы) системы будут помечены подстановками, выражающими вычислительный эффект линейных участков схемы программ, т.е. конечных последовательностей операторов присваивания.

Установив такую взаимосвязь схем программ и автоматов, мы сможем воспользоваться методами и результатами теории автоматов для изучения проблемы эквивалентности стандартных схем программ.

Описание стандартных схем программ при помощи подстановок

Подстановка — это всякое отображение $\theta : Var \to Terms$, сопоставляющее каждой переменной некоторый терм.

Множество $Dom_\theta = \{x : \theta(x) \neq x\}$ называется областью подстановки. Если область подстановки — это конечное множество переменных, то подстановка называется конечной. Множество конечных подстановок обозначим Subst.

Если $\theta \in Subst$ и $Dom_\theta = \{x_1, x_2, \ldots, x_n\}$, то подстановка θ однозначно определяется множеством пар

$$\{x_1/\theta(x_1), x_2/\theta(x_2), \ldots, x_n/\theta(x_n)\}.$$

Каждая пара $x_i/\theta(x_i)$ называется связкой.

Если $Dom_\theta = \emptyset$, то такая подстановка θ называется тождественной (или пустой) подстановкой и обозначается буквой ε.

Описание стандартных схем программ при помощи подстановок

Для подстановок определены две основные операции.

Применение подстановки

Пусть $\theta \in Subst$ и $E \in Terms \cup Atoms$. Тогда запись $E\theta$ обозначает результат применения подстановки θ к E , который определется так:

- если $E = x,\ x \in Var$, то $E\theta = \theta(x)$;
- если $E = c,\ c \in Const$, то $E\theta = c$;
- если $E = F(t_1, t_2, \ldots, t_k)$, то $E\theta = F(t_1\theta, t_2\theta, \ldots, t_n\theta)$.

Композиция подстановок

Пусть $\theta, \eta \in Subst$. Композиция подстановок $\theta\eta$ — это подстановка μ , которая определяется следующим соотношением: $\mu(x) = (x\theta)\eta$ для любой $x \in Var$.

Описание стандартных схем программ при помощи подстановок

Кроме того, применение подстановок можно распространить на оценки переменных — состояния памяти схем программ.

Применение подстановок к оценкам переменных

Пусть $\theta \in Subst$ и $\xi : Var \to D$ — оценка переменных в интерпретации I . Тогда результатом применения подстановки θ к оценке переменных ξ является оценка переменных $\xi' = \theta[\xi]_I$, которая определяется соотношением $\xi'(x) = \theta(x)[\xi]_I$ для каждой переменной $x, x \in Var$.

Иными словам, если $\theta = \{x_1/t_1, x_2/t_2, \ldots, x_n/t_n\}$, то

$$\theta[\xi]_I(x) = \begin{cases} t_i[\xi]_I, & \text{если } x = x_i,\ 1 \le i \le n, \\ \xi(x), & \text{иначе.} \end{cases}$$

Описание стандартных схем программ при помощи подстановок

Вычислительный эффект оператора присваивания может быть выражен при помощи операций над подстановками.

Утверждение 1.

Для любого оператора присваивания $x := t(x_1, \ldots, x_k)$, интерпретации I и оценки переменных $\xi : Var \to D$ верно равенство

$$\xi[x \leftarrow t[\xi_i(x_1), \ldots, \xi_i(x_k)]_I] = \{x/t(x_1, \ldots, x_k)\}[\xi]_I$$

Доказательство.

Следует непосредственно из определений.

Описание стандартных схем программ при помощи подстановок

Операции композиции и применения подстановок к оценкам переменных взаимосвязаны простым соотношением.

Утверждение 2.

Для любых подстановок $\theta_1, \theta_2 \in \mathit{Subst}$, интерпретации I и оценки переменных $\xi : \mathit{Var} \to D$ верно равенство

$$\theta_1\theta_2[\xi]_I = \theta_1[\theta_2[\xi]_I]_I.$$

Доказательство.

Следует непосредственно из определений.

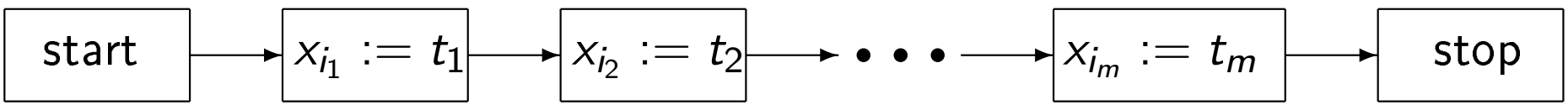
Описание стандартных схем программ при помощи подстановок

На основании утверждений 1 и 2 вычислительный эффект последовательности операторов присваивания может быть выражен при помощи операций композиции и применения подстановок к оценкам переменных следующим образом.

Утверждение 3.

Для любых интерпретации I и оценки переменных $\xi_0 : Var \to D$ результат ξ вычисления $comp(\pi, \xi_0, I)$ стандартной схемы программ

π



удовлетворяет равенству $\xi = (\{x_{i_m}/t_m\} \ldots \{x_{i_2}/t_2\}\{x_{i_1}/t_1\})[\xi]_I$.

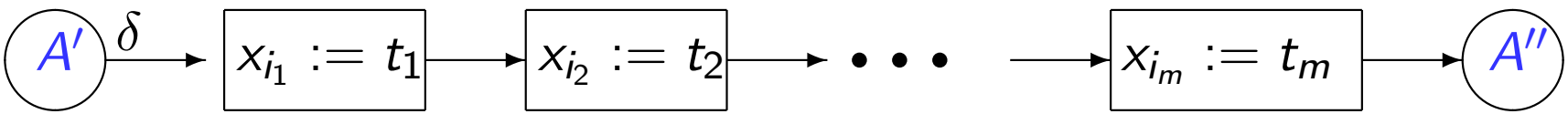
Доказательство.

Индукцией по m с применением утверждений 1 и 2.

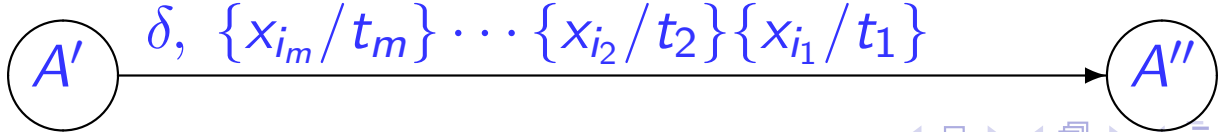
Описание стандартных схем программ при помощи подстановок

Каждой стандартной схеме программ π сопоставим размеченную систему переходов $LTS(\pi)$, вершинами которой являются начальная и финальная вершины схемы , а также все вершины-распознаватели схемы π . Все эти вершины сохраняют приписанные им операторы и тесты.

Все переходы между вершинами системы помечены подстановками; кроме того, каждый переход, исходящий из вершины-распознавателя помечен 0 или 1 . Разметка переходов проводится по следующему правилу: всякому маршруту в схеме π между вершинами-преобразователями, имеющему вид

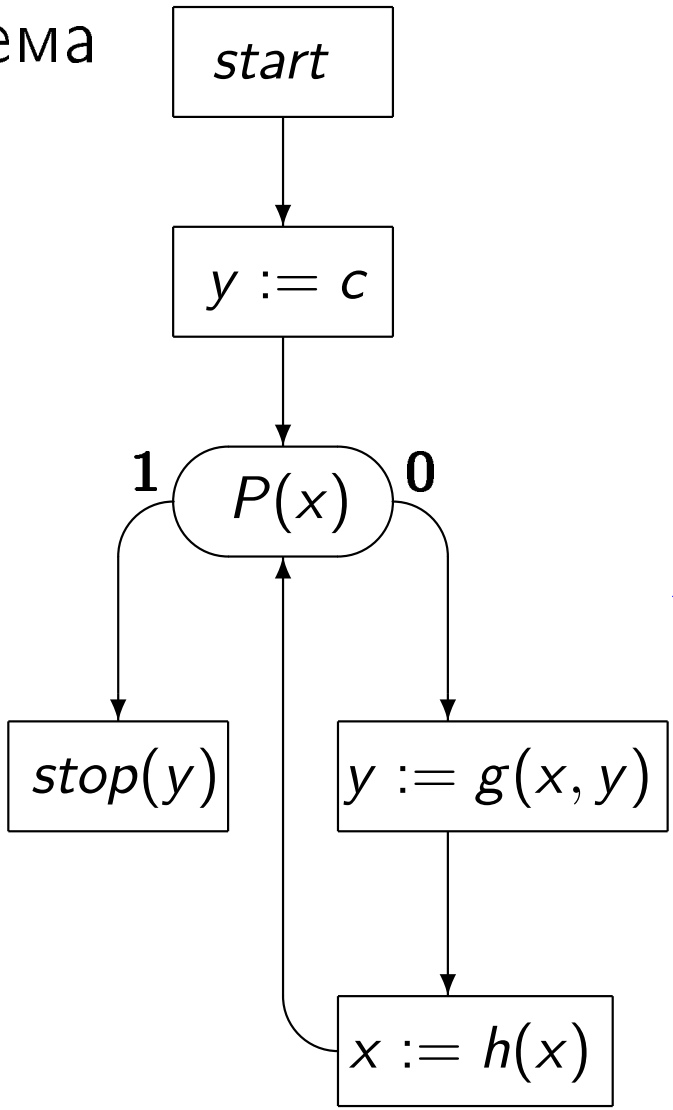


в системе $LTS(\pi)$ соответствует переход



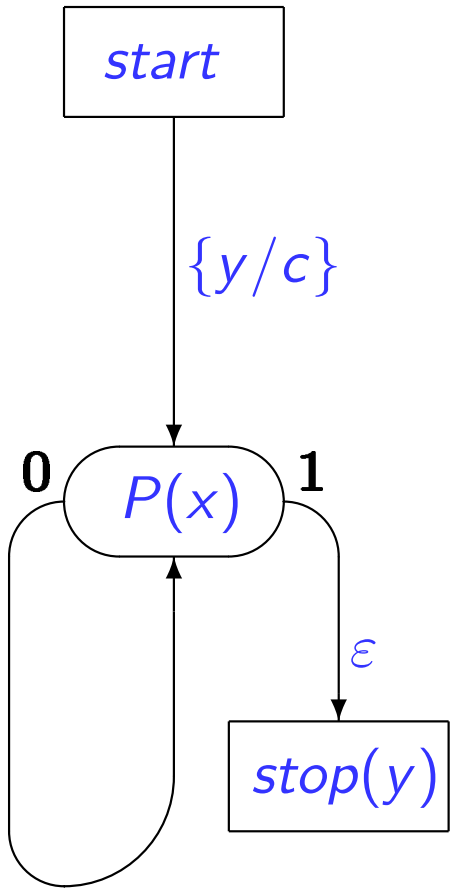
Описание стандартных схем программ при помощи подстановок

Схема π



Система переходов $LTS(\pi)$

$\{x/h(x), y/g(x, y)\}$



Описание стандартных схем программ при помощи подстановок

Вычислением системы переходов $LTS(\pi))$ в интерпретации I на начальном состоянии памяти ξ называется максимальная последовательность пар

$$comp(LTS(\pi), I, \xi) = (v_0, \xi_0), (v_1, \xi_1), \ldots, (v_i, \xi_i), (v_{i+1}, \xi_{i+1}), \ldots$$

удовлетворяющая следующим требованиям:

1. v_0 — начальная вершина, $\xi_0 = \xi$;
2. для каждого $i, i \geq 0$
   - v_{i+1} — это вершина-преемник вершины v_i ;
   - если v_i — вершина-распознаватель, которой приписан тест $A(x_1, \ldots, x_k)$, то v_{i+1} — это δ -преемник вершины v_i , где $\delta = A[\xi_i(x_1), \ldots, \xi_i(x_k)]_I$;
   - если вершины v_i и v_{i+1} связаны переходом, который помечен подстановкой θ , то $\xi_{i+1} = \theta[\xi_i]_I$;
   - если v_i — финальная вершина, которой приписан оператор $stop(y_1, \ldots, y_m)$, то вычисление $comp(LTS(\pi), I, \xi)$ завершается парой (v_i, ξ_i) , и набор $val(\pi, I, \xi) = (\xi_i(y_1), \ldots, \xi_i(y_m))$ считается результатом

Описание стандартных схем программ при помощи подстановок

Из определения вычислений для схем программ и соответствующих им систем переходов, а также из утверждения 3 следует

Утверждение 4.

Для любых стандартной схемы программ π , интерпретации I и оценки переменных ξ последовательность пар

$$(v_0, \xi_0), (v_1, \xi_1), \ldots, (v_i, \xi_i), (v_{i+1}, \xi_{i+1}), \ldots$$

является вычислением $comp(LTS(\pi), I, \xi)$ системы переходов $LTS(\pi)$, соответствующей схеме π , в том и только том случае, если эта последовательность является проекцией вычисления $comp(\pi, I, \xi)$ схемы π на множество контрольных вершин схемы (т.е. удалением из $comp(\pi, I, \xi)$ всех таких пар (v_i, ξ_i) , у которых v_i — вершина-преобразователь).

Описание стандартных схем программ при помощи подстановок

Системы размеченных переходов $LTS(\pi_1)$ и $LTS(\pi_2)$ называются функционально эквивалентными , если для любой интерпретации I и для любого начального состояния памяти ξ верно одно из двух:

- оба вычисления $comp(LTS(\pi_1), I, \xi)$ и $comp(LTS(\pi_2), I, \xi)$ бесконечны (зацикливаются);
- оба вычисления $comp(LTS(\pi_1), I, \xi)$ и $comp(LTS(\pi_2), I, \xi)$ завершаются, и результаты этих вычислений одинаковы $val(LTS(\pi_1), I, \xi) = val(LTS(\pi_2), I, \xi)$.

Из утверждения 4 следует

Теорема 1.

Каковы бы ни были схемы программ π_1 и π_2 , они функционально эквивалентны тогда и только тогда, когда функционально эквивалентны соответствующие им размеченные системы переходов $LTS(\pi_1)$ и $LTS(\pi_2)$.

Описание стандартных схем программ при помощи подстановок

Теорема 1 позволяет при изучении проблемы эквивалентности стандартных схем программ рассматривать не сами схемы, а соответствующие им системы переходов, размеченные подстановками.

Синтаксис стандартных схем программ удобен, когда нужно установить взаимосвязь этой модели вычислений с реальными программами.

Но для математического анализа стандартных схем программ гораздо более подходят размеченные системы переходов.

Поэтому в дальнейшем, не оговаривая этого особо, мы будем называть «схемой программ» размеченную систему переходов, соответствующую стандартной схеме программ.

Эрбрановские интерпретации для стандартных схем программ

Проблема эквивалентности стандартных схем программ сходна с проблемой общезначимости формул логики предикатов — и в том и в другом случае нужно проверять поведение информационных конструкций (формул, представляющих знания, или программ, описывающих алгоритмы) на всех возможных алгебраических интерпретациях базисных элементов, из которых созданы эти конструкции.

При этом бесконечно варьируется не только множество интерпретаций, но и поведение конструкции в одной отдельной интерпретации — в случае формул логики предикатов такое разнообразие поведения обусловлено использованием кванторов, а в случае схем программ — сколь угодно длинными их вычислениями.

Эрбрановские интерпретации для стандартных схем программ

Для сокращения такого «тройного комбинаторного взрыва» при проверке выполнимости формул логики предикатов Жак Эрбран предложил ограничиться только интерпретациями специального вида — свободными алгебраическими системами, которые впоследствии были названы эрбрановскими интерпртациями.

Попробуем посмотреть, можно ли ограничиться только эрбрановскими интерпретациями при анализе поведения стандартных схем программ, и, в частности, при исследовании проблемы функциональной эквивалентности.

Эрбрановские интерпретации для стандартных схем программ

Далее будем рассматривать стандартные схемы программ, в которых используются предметные переменные из множества $Var = \{x_1, x_2, \ldots, x_n, \ldots\}$, а также термы и атомы сигнатуры $\sigma = (Const, Func, Pred)$.

Для каждой предметной переменной $x, x \in Var$, введем специальную новую константу **x** , не принадлежащую сигнатуре σ . Множество введенных таким образом констант обозначим записью C_{Var} .

Эрбрановским универсумом будем называть множество всевозможных основных (т.е. не содержащих переменных) термов, которые могут быть построены из множества констант $Const \cup C_{Var}$ и множества функциональных символов $Func$.

Определенный таким образом эрбрановский универсум обозначим записью $H_{Var,\sigma}$ (индексы в этой записи иногда будум опускать).

Эрбрановские интерпретации

Эрбрановской интерпретацией (ее так же называют свободной интерпретацией) называется всякая интерпретация $I = (D, \overline{Const}, \overline{Func}, \overline{Pred})$ сигнатуры σ , которая удовлетворяет следующим требованиям:

1. областью интерпретации служит эрбрановский универсум $H_{Var,\sigma}$;
2. оценкой $\overline{c}$ каждой константы c из множества $Const \cup C_{Var}$ является терм c из множества $H_{Var,\sigma}$;
3. значением каждой функции $\overline{f}^{(n)}$ на наборе основных термов $t_1, t_2, \ldots, t_n$ из множества $H_{Var,\sigma}$ является основной терм $f^{(n)}(t_1, t_2, \ldots, t_n)$;
4. оценка предикатных символов может быть произвольной.

Как видно из определения, эрбрановские интерпретации заданной сигнатуры имеют только одну степень свободы — произвольный выбор оценки предикатных символов.

Эрбрановские интерпретации

Вычисления стандартных схем программ на свободных интерпретациях начинаются из одного и того же начального состояния памяти — фиксированной начальной оценки переменных $\widehat{\xi_0}$, удовлетворяющей равенству $\widehat{\xi_0}(x) = \mathbf{x}$ для каждой предметной переменной $x, x \in Var$.

Из определения эрбрановской интерпретации и начальной оценки переменных следует, что справедливо

Утверждение 5.

Для любых подстановок θ_1, θ_2 и эрбрановской интерпретации I верно соотношение

$$\theta_1[\widehat{\xi_0}]_I = \theta_2[\widehat{\xi_0}]_I \Leftrightarrow \theta_1 = \theta_2.$$

А это означает, что множество состояний данных, вычислимых схемами программ в эрбрановских интерпретациях посредством операции применения подстановок, изоморфно множеству подстановок с операцией композиции.

Эрбрановские интерпретации

Таким образом, при вычислении схем программ в эрбрановских интерпретациях в качестве состояний данных можно использовать подстановки из множества *Subst* .

Теорема 2.

Каковы бы ни были схемы программ π_1, π_2 они функционально эквивалентны тогда и только тогда, когда они эквивалентны на множестве эрбрановских интерпретаций

Доказательство.

($\Rightarrow$) Очевидно.

($\Leftarrow$) Рассмотрим произвольную интерпретацию I и начальную оценку переменных ξ_0 в этой интерпретации. Определим эрбрановскую интерпретацию I_H , задав следующую оценку атомарных формул:

для каждой атомарной формулы A и подстановки θ положим

$$A[\theta]_{I_H} = A[\theta[\xi_0]]_I$$

Эрбрановские интерпретации

Тогда для каждой из схем программ $\pi_j, j = 1, 2$, имеют место соотношения

$$comp(\pi_j, I, \xi_0) = (v_{0j}, \xi_0), (v_{1j}, \xi_{1j}), \ldots, (v_{ij}, \xi_{ij}), \ldots, (v_{n_j j}, \xi_{n_j j})$$

$$\Longleftrightarrow$$

существует последовательность подстановок $\theta_{1j}, \ldots, \theta_{ij}, \ldots, \theta_{nj}$ (каждая θ_{ij} — это композиция подстановок, приписанных дугам маршрута, который реализует схема π_j в интерпретации I), для которой

$$comp(\pi_j, I, \xi_0) = (v_{0j}, \xi_0), (v_{1j}, \theta_{1j}[\xi_0]_I), \ldots, (v_{ij}, \theta_{ij}[\xi_0]_I), \ldots, (v_{n_j j}, \theta_{n_j j}[\xi_0]_I)$$

$$\Longleftrightarrow$$

согласно определению оценок атомарных формул в эрбрановской интерпретации I_H

$$comp(\pi_j, I_H, \varepsilon) = (v_{0j}, \varepsilon), (v_{1j}, \theta_{1j}), \ldots, (v_{ij}, \theta_{ij}), \ldots, (v_{n_j j}, \theta_{n_j j}).$$

Т.к. схемы эквивалентны в H-интерпретациях, имеет место равенство $\theta_{n_1 1} = \theta_{n_2 2}$. Поэтому

$$res(\pi_1, I, \xi_0) = \xi_{n_1 1} = \theta_{n_1 1}[\xi_0] = \theta_{n_2 2}[\xi_0] = \xi_{n_2 2} = res(\pi_2, I, \xi_0)$$,

т.е. схемы π_1, π_2 вычисляют одинаковые результаты в (произвольной) интерпретации I , и, значит, они эквивалентны.

Неразрешимость проблем анализа поведения стандартных схем программ

Имея в виду, что свойства вычислений стандартных схем программ полностью проявляются на эрбрановских интерпретациях, покажем, что все эти свойства (завершаемость, тотальность, функциональной эквивалентность и др.) алгоритмически неразрешимы.

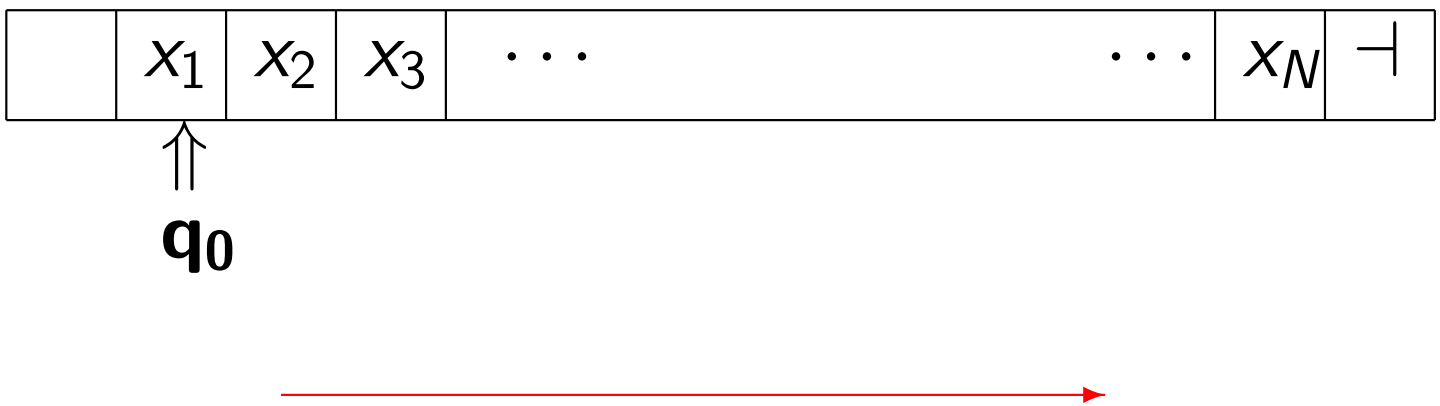
Это можно сделать, продемонстрировав, что стандартные схемы программ моделируют на эрбрановских интерпретациях вычисления многоголовочных конечных автоматов.

А, как известно, проблемы анализа поведения многоголовочных автоматов алгоритмически неразрешимы.

Неразрешимость проблем анализа

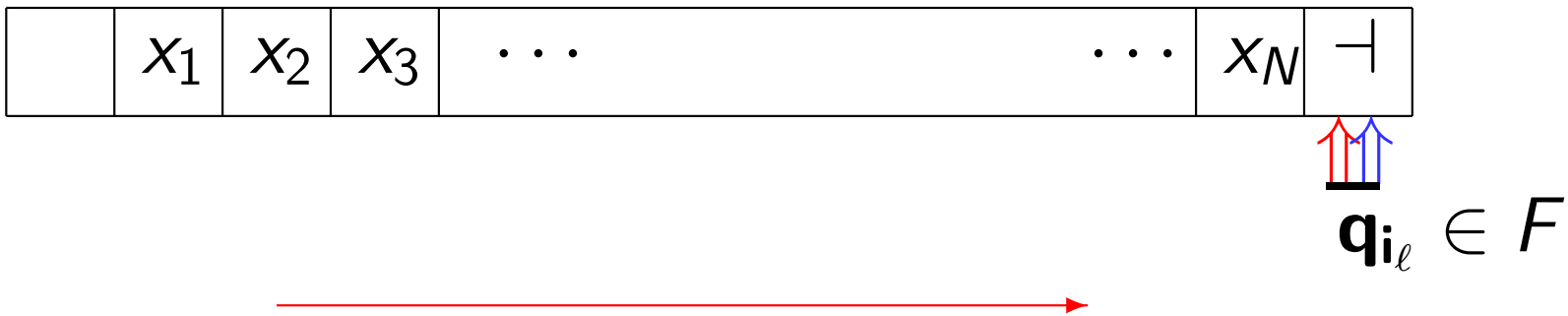
Конечный автомат — это устройство для чтения слов, записанных на ленте.

Обычный конечный автомат имеет только одну считывающую головку, которая движется по ленте в одну сторону.



Неразрешимость проблем анализа

А что будет, если автомат будет иметь **ДВЕ** считывающие головки, которые могут перемещаться только в одну сторону, но при этом независимо друг от друга?



Оказывается, что вычислительные возможности такого автомата усиливаются настолько, что он способен моделировать машины Тьюринга!

Неразрешимость проблем анализа

Мы ограничимся рассмотрением двухголовочных автоматов, работающих над двоичным алфавитом.

Двухголовочный детерминированный конечный автомат — это система $\mathcal{D} = (S_1, S_2, s_0, F, T)$, где

- S_1 и S_2 — конечные множества состояний, в которых автомат управляет соответственно первой и второй головками.
- s_0 — начальное состояние из множества $S_1 \cup S_2$,
- F — подмножество финальных состояний, $F \cap (S_1 \cup S_2) = \emptyset$,
- $T : S_1 \cup S_2 \times \{0, 1\} \to S_1 \cup S_2 \cup F$ — функция переходов.

Если $s \in S_j, j \in \{1, 2\}$, то равенство $T(s, x) = s'$ означает, что при считывании j -й головкой автомата, пребывающего в состоянии s , символа x автомат сдвигает эту головку на одну позицию вправо и переходит в состояние s' . Автомат прекращает вычисление при достижении финального состояния.

Неразрешимость проблем анализа

Известно, что проблема пустоты для многоленточных автоматов алгоритмически неразрешима.

Неразрешимость этой проблемы можно доказать построением процедуры, транслирующей каждую машину Тьюринга M в двухголовочный автомат $\mathcal{D}_M$, который достигает финального состояния в том и только том случае, если входное слово представляет собой конечную последовательность конфигураций машины Тьюринга M , образующую завершающееся вычисление этой машины на пустой ленте.

Поскольку проблема останова машин Тьюринга на пустой ленте неразрешима, также неразрешима и проблема пустоты для двухголовочных автоматов.

А теперь покажем, как проблема пустоты для двухголовочных автоматов сводима к аналогичной проблеме для стандартных схем программ, работающих в эрбрановских интерпретациях.

Неразрешимость проблем анализа

Теорема 3.

Для любого бинарного двухголовочного конечного автомата $\mathcal{D} = (S_1, S_2, s_0, F, T)$ существует такая схема программ $\pi_{\mathcal{D}}$, которая имеет хотя бы одно конечное завершающееся вычисление в том и только том случае, когда автомат $\mathcal{D}$ допускает хотя бы одно входное слово.

Доказательство.

Будем строить схемы в сигнатуре $Const = \emptyset$, $Func = \{f^{(1)}\}$, $Pred = \{P^{(1)}\}$; в них будут использоваться только две переменные x_1, x_2, соотвтетствующие двум головкам автомата. Для каждого $i, i \geq 0$, терм $\underbrace{f(\ldots f}_{i \text{ раз}}(\mathbf{x})\ldots)$ условимся обозначать записью $t^i(\mathbf{x})$.

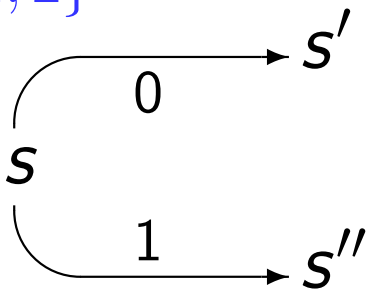
Трансляция двухголовочного конечного автомата $\mathcal{D}$ в схему программ $\pi_{\mathcal{D}}$ проводится по следующим трем правилам

Неразрешимость проблем анализа

Автомат $\mathcal{D}$

$s \in S_j$

$j \in \{1, 2\}$



s_0 — начальное состояние

s — финальное состояние

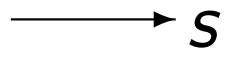
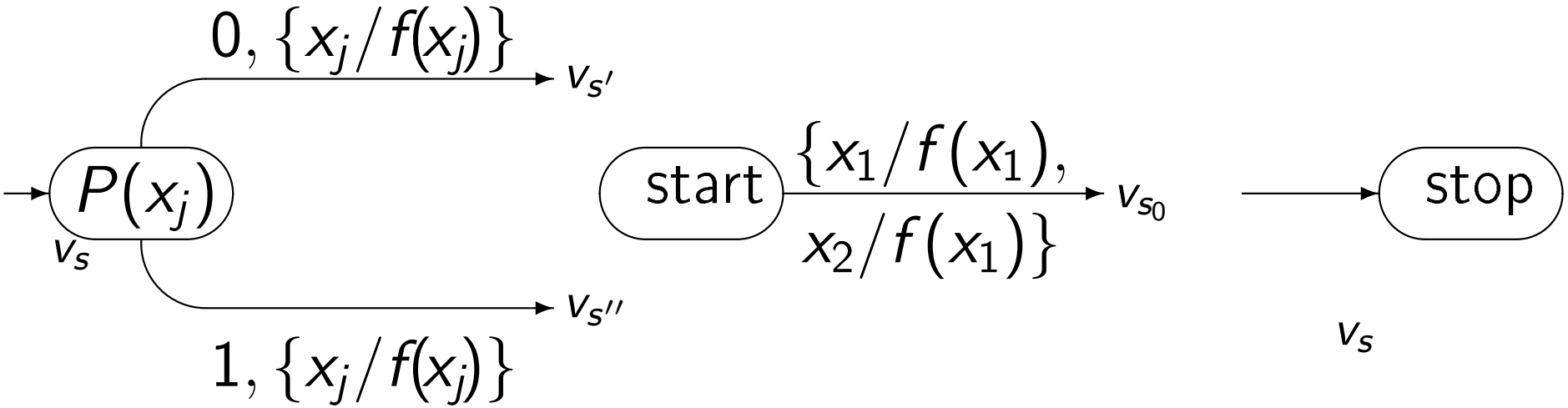


Схема программ $\pi_{\mathcal{D}}$



Неразрешимость проблем анализа

Взаимосвязь между вычислениями двухголовочного автомата $\mathcal{D}$ и схемы программ $\pi_{\mathcal{D}}$ основывается на соответствии между входными словами автомата и эрбрановскими интерпретациями, на которых проводятся вычисления схкмы:

1) каждому двоичному входному слову $w = \delta_1\delta_2, \ldots, \delta_n$ соответствует эрбрановская интерпретация I_w , в которой оценка предикатного символа P определяется равенствами $\overline{P}[t^i(\mathbf{x}_i)] = \delta_i$ для каждого $i, 1 \leq i \leq n$.

2) каждой эрбрановской интерпретации I и натуральному числу n соответствует слово $w_{I,n} = \delta_1\delta_2, \ldots, \delta_n$, в котором каждый двоичный символ $\delta_i, 1 \leq i \leq n,$ определяется равенством $\delta_i = \overline{P}[t^i(\mathbf{x}_i)]$.

Неразрешимость проблем анализа

Принимая во внимание тот способ трансляции многоголовочных автоматов в схемы программ, который был описан ранее, индукцией по n легко показать справедливость следующих двух утверждений:

1) Двухголовочного автомата $\mathcal{D}$ при вычислении на слове $w = \delta_1\delta_2, \ldots, \delta_n$ достигает состояния s и при этом его считывающие головки обозревают буквы слова w в позициях k_1 и k_2 тогда и только тогда, когда вычисление $comp(\pi_{\mathcal{D}}, I_w, \varepsilon)$ схемы программ $\pi_{\mathcal{D}}$ содержит пару $(v_s, \{x_1/t^{k_1}(x_1), x_1/t^{k_2}(x_1)\})$.

2) Схема программ $\pi_{\mathcal{D}}$ в эрбрановской интерпретации I имеет вычисление $comp(\pi_{\mathcal{D}}, I, \varepsilon)$, содержащее пару $(v_s, \{x_1/t_1, x_2/t_2\})$ тогда и только тогда, когда $t_1 = t^{k_1}(x_1), t_2 = t^{k_2}(x_1)$ для некоторых натуральных k_1, k_2 , и при этом двухголовочный автомат $\mathcal{D}$ при вычислении на слове w_I достигает состояния s , в котором его считывающие головки обозревают буквы слова w в позициях k_1 и k_2 .

Неразрешимость проблем анализа

Из приведенных утверждений следует, что двухголовочный автомат $\mathcal{D}$ допускает некоторое бинарное слово тогда и только тогда, когда схема программ $\pi_{\mathcal{D}}$ завершает вычисление в некоторой эрбрановской интерпретации.

Учитывая теорему об эрбрановских интерпретациях, приходим к выводу о том, что проблема пустоты для двухголовочных конечных автоматов сводится к проблеме пустоты для стандартных схем программ. □

Следствие 1.

Проблема останова для стандартных схем программ алгоритмически неразрешима.

Следствие 2.

Проблемы тотальности и функциональной эквивалентности для стандартных схем программ алгоритмически неразрешимы.

Логико-термальная эквивалентность стандартных схем программ

Функциональная эквивалентность стандартных схем программ, требующая от эквивалентных схем программ вычислять одинаковый результат в каждой интерпретации, оказалась алгоритмически неразрешимой.

Разрешимое отношение эквивалентности можно получить, усилив требования, предъявляемые к эквивалентным программам. Например, пожелать, чтобы эквивалентные программы в каждой интерпретации выполняли одинаковые последовательности операторов или проверяли одну и ту же последовательность логических условий.

Однако в 1973 г. был установлен результат, который значительно усугубил негативный эффект алгоритмической неразрешимости проблемы функциональной эквивалентности.

Логико-термальная эквивалентность стандартных схем программ

Отношение эквивалентности $\approx$ на множестве стандартных схем программ называется

- интерпретационным , если $\pi_1 \not\approx \pi_2$ обязательно подразумевает, что для некоторой интерпретации I и начальной оценки переменных ξ_0 либо $val(\pi_1, I, \xi_0) \neq val(\pi_2, I, \xi_0)$, либо вычисление одной из схем бесконечно, а другой конечно;
- невырожденным , если из $\pi_1 \approx \pi_2$ обязательно следует, что в любой эрбрановской интерпретации H термы набора $val(\pi_1, H, \xi_0)$ состоят из тех же самых символов, что и термы набора $val(\pi_2, H, \xi_0)$.

Теорема [В.Э. Иткин, З. Звиногродский]

Любое отношение интерпретационной невырожденной эквивалентности стандартных схем программ алгоритмически неразрешимо.

Логико-термальная эквивалентность стандартных схем программ

Эффект этой теоремы похож на действие теоремы Райса: никакая «разумная» эквивалентность схем программ, определяемая на основании интерпретаций, неразрешима.

Поэтому для эффективной проверки функциональной эквивалентности программ нужно придумать такое отношение эквивалентности $\approx$ схем программ, которое

1. аппроксимирует функциональную эквивалентность, т.е. удовлетворяет соотношению $\pi_1 \approx \pi_2 \;\Rightarrow\; \pi_1 \sim \pi_2$ для любых схем программ π_1, π_2 ;
2. определяется без привлечения понятия «интерпретация».

Наиболее интересное отношение эквивалентности такого рода — логико-термальную эквивалентность схем программ — предложил в 1972 г. Владимир Эммануилович Иткин (Новосибирск).

Логико-термальная эквивалентность стандартных схем программ

Рассмотрим стандартную схему программ, представленную в виде размеченной системы переходов $LTS(\pi)$, в которой

- множество вершин V включает начальную вершину v_0 , множество финальных вершин V_{out} , и множество внутренних вершин V' : $V = \{v_0\} \cup V_{out} \cup V'$;
- начальной вершине v_0 приписан оператор $B(v_0) =$старт ;
- каждой финальной вершине $u, u \in V_{out}$, приписан оператор выхода $B(u) = \text{стоп}(x_{i_1}, \ldots, x_{i_k})$;
- каждой внутренней вершине $v, v \in V'$, приписана атомарная формула $B(v) = P(t_1, \ldots, t_m)$;
- из начальной вершины v_0 исходит единственная дуга $v_0 \xrightarrow{\theta_0} v$, помеченная подстановкой $\theta_0 \in Subst$;
- из каждой внутренней вершины v исходят две дуги $v \xrightarrow{0,\theta'} v'$ и $v \xrightarrow{1,\theta''} v''$, помеченные битами 0 , 1 , а также подстановками $\theta', \theta'' \in Subst$.

Логико-термальная эквивалентность

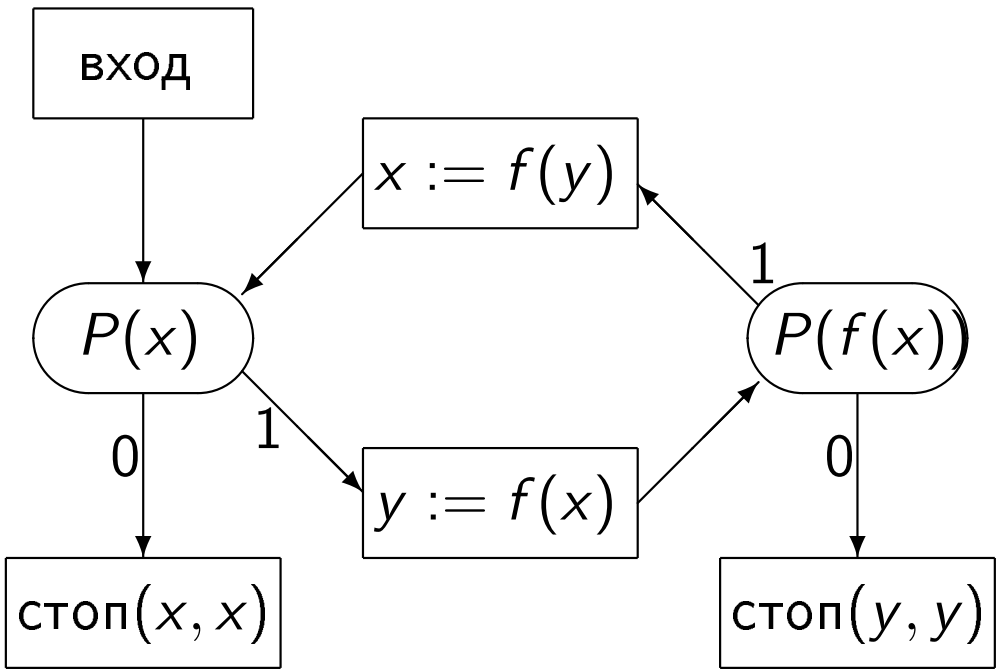
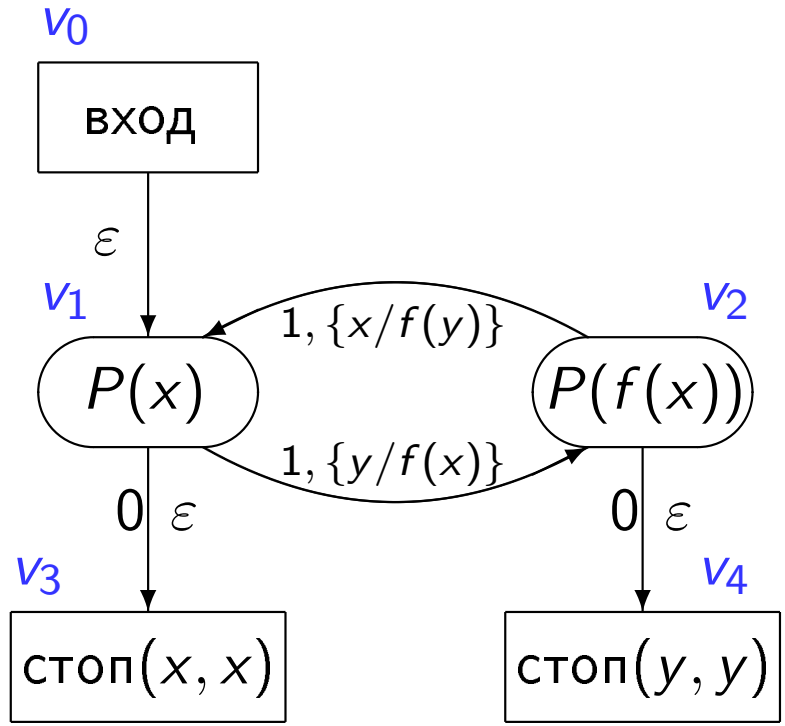


Схема π



Система переходов π

Логико-термальная эквивалентность

Рассмотрим некоторый путь в системе переходов $LTS(\pi)$, ведущий из начальной вершины v_0 в некоторую финальную вершину: $\alpha = v_0 \xrightarrow{\theta_0} v_1 \xrightarrow{\delta_1,\theta_1} v_2 \xrightarrow{\delta_2,\theta_2} \cdots \xrightarrow{\delta_n,\theta_n} v_{n+1}$

Логико-термальной историей пути α будем называть чередующуюся последовательность

$$lth(\alpha) = B(v_1)\theta_0, \delta_1, B(v_2)\theta_1\theta_0, \delta_2, \ldots, \delta_n, B(v_{n+1})\theta_n \cdots \theta_2\theta_1\theta_0$$

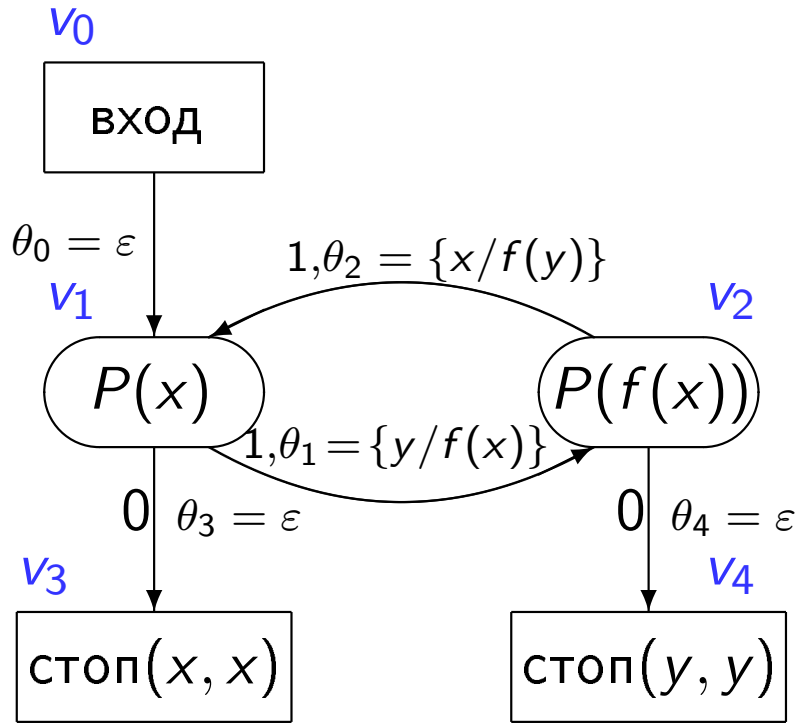
состоящую из

- примеров $B(v_i)\theta_{i-1} \cdots \theta_1\theta_0$ тех атомов, которые приписаны вершинам пути $v_i, 1 \leq i \leq n+1$, и специализированы композицией подстановок $\theta_{i-1} \cdots \theta_1\theta_0$, помечающих дуги пути α ,
- битов (булевых значений) δ_i , которые также помечают дуги пути α .

Логико-термальная эквивалентность

$$\alpha = v_0 \xrightarrow{\theta_0} v_1 \xrightarrow{1,\theta_1} v_2 \xrightarrow{1,\theta_2} v_1 \xrightarrow{1,\theta_1} v_2 \xrightarrow{0,\theta_4} v_4$$

$$\begin{aligned} lth(\alpha) = {} & P(x), 1, \\ & P(f(x)), 1, \\ & P(f(f(x))), 1, \\ & P(f(f(f(x)))), 0, \\ & \text{стоп}(f(f(f(x))), f(f(f(x)))) \end{aligned}$$



Система переходов π

Логико-термальная эквивалентность

Фактически, л-т история пути α в схеме π представляет собой описание условий, при которых этот путь может быть реализован в некоторой интерпретации, а также описание результата вычисления схемы π в этой интерпретации.

Утверждение 1.

Путь α в схеме $LTS(\pi)$, имеющий логико-термальную историю

$$lth(\alpha) = B_1, \delta_1, B_2, \delta_2, \ldots, B_i, \delta_i, \ldots, B_n, \delta_n, \text{стоп}(t_1, \ldots, t_k),$$

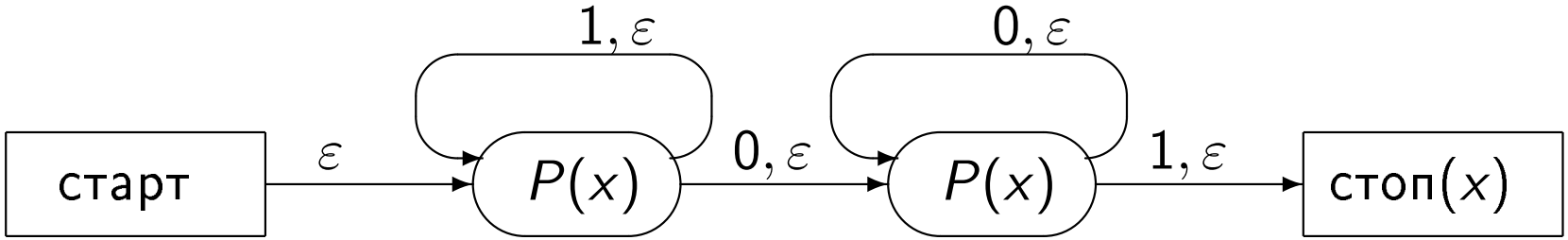
реализуем в интерпретации I при начальной оценке переменных ξ_0 тогда и только тогда, когда для каждого $i, 1 \leq i \leq n$, справедливо равенство $B_i[\xi_0]_I = \delta_i$. При этом результатом вычисления $comp(\pi, I, \xi_0)$ будет набор $(t_1[\xi_0]_I, \ldots, t_k[\xi_0]_I)$.

Доказательство.

Следует непосредственно из определения вычисления схемы и л-т истории.

Логико-термальная эквивалентность

Заметим, что не все пути, имеющие логико-термальные истории, могут быть реализованы.



Например, в этой схеме ни один путь из начальной вершины в финальную не может быть реализован ни в одной интерпретации, т.к. логико-термальные истории этих путей предъявляют к интерпретациям несовместные требования:

$$\begin{cases} P(x)[\xi]_I = 1 \\ P(x)[\xi]_I = 0 \end{cases}$$

Однако эта схема имеет бесконечно много л-т историй вида

$$P(x), 1, \ldots, P(x), 1, P(x), 0, P(x), 0, \ldots, P(x), 0, P(x), 1, \text{стоп}(x)$$

Логико-термальная эквивалентность

Логико-термальной характеристикой (детерминантом) стандартной схемы π назовем множество всех ее логико-термальных историй

$$Det(\pi) = \{lth(\alpha) : \quad \alpha \text{ — терминальный путь в системе переходов для схемы } \pi\}$$

Стандартные схемы программ π_1 и π_2 назовем логико-термально эквивалентными (л-т эквивалентными), обозначая это отношение записью $\pi_1 \overset{lt}{\sim} \pi_2$, если выполняется равенство $Det(\pi_1) = Det(\pi_2)$.

Из утверждения 1 и определения л-т эквивалентности следует

Теорема 1.

Для любых схем программ π_1 и π_2 верно соотношение

$$\pi_1 \overset{lt}{\sim} \pi_2 \;\Rightarrow\; \pi_1 \sim \pi_2.$$

Логико-термальная эквивалентность

Схемы π_1 и π_2 л-т эквивалентны (это еще надо доказать!).
Поэтому они также функционально эквивалентны.

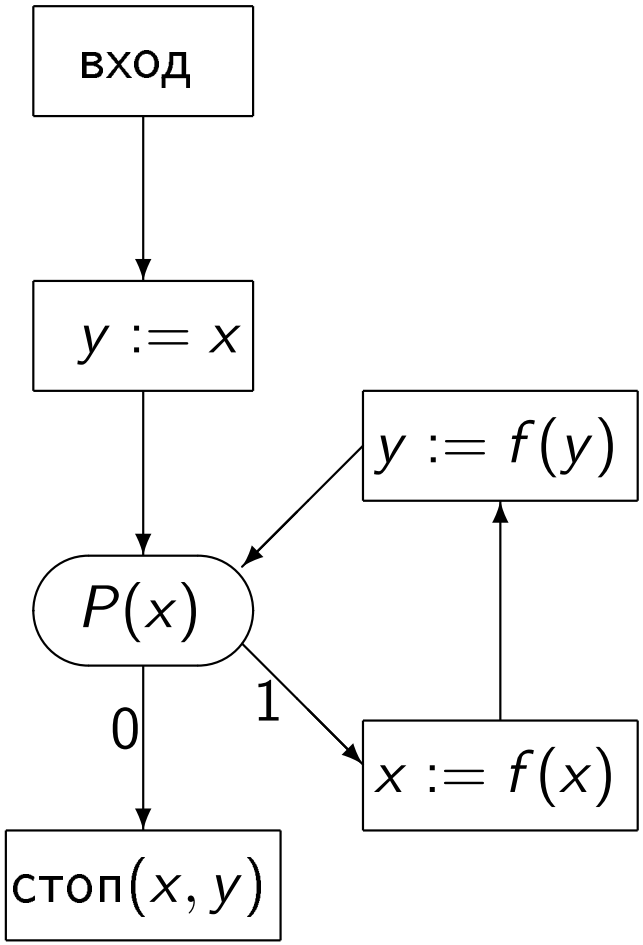


Схема π_1

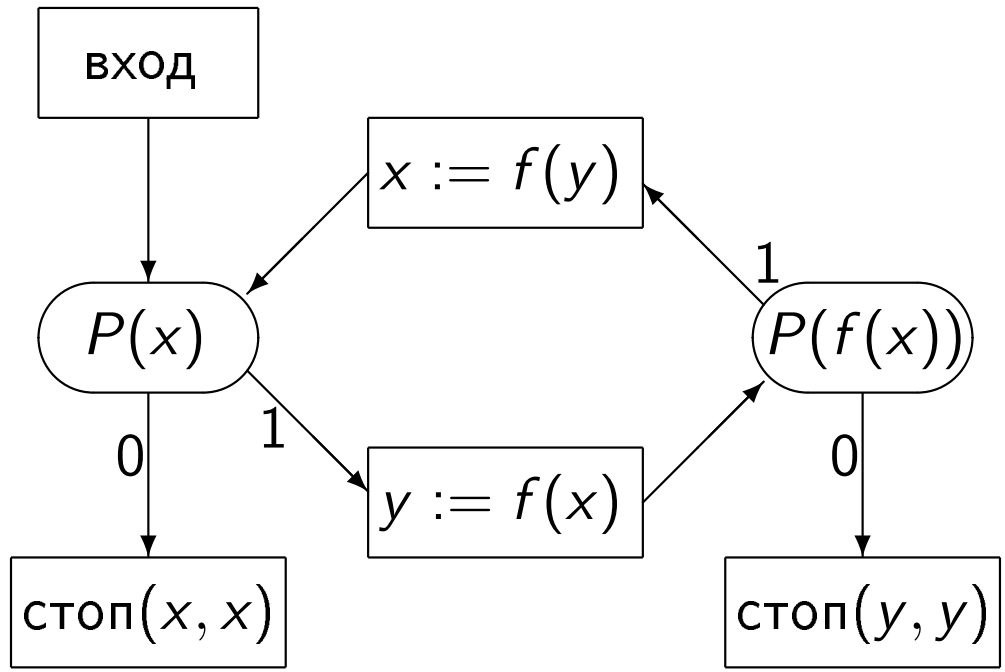
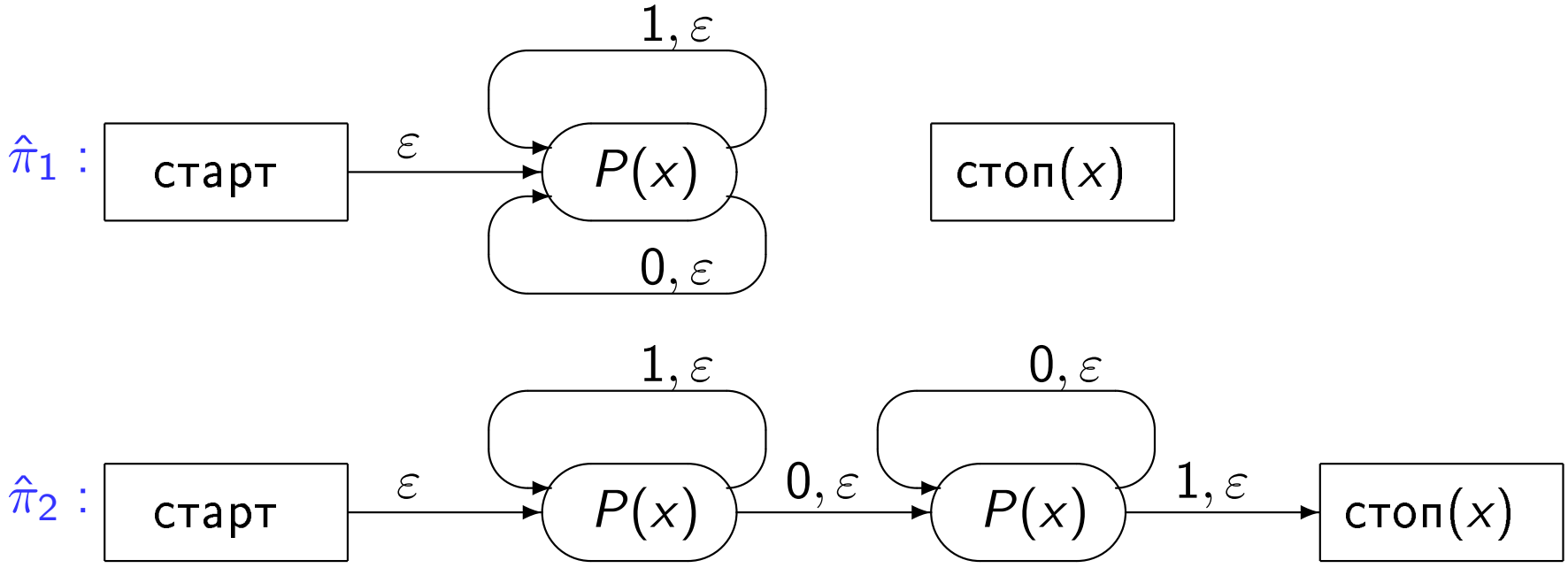


Схема π_2

Логико-термальная эквивалентность

Схемы $\hat{\pi}_1$ и $\hat{\pi}_2$ не являются л-т эквивалентными, т.к. $Det(\hat{\pi}_1) = \emptyset$, $Det(\hat{\pi}_2) \neq \emptyset$.

Однако эти схемы функционально эквивалентны.



Логико-термальная эквивалентность

Итак, мы выяснили, что

1. Логико-термальная эквивалентность схем программ **аппроксимирует** функциональную эквивалентность. Поэтому успешная проверка логико-термальной эквивалентности позволяет доказать функциональную эквивалентность схем программ.

2. Логико-термальная эквивалентность схем программ является **не-интерпретационной эквивалентностью**, и поэтому не подпадает под действие теоремы Иткина-Звиногродского о неразрешимости проблемы интерпретационной эквивалентности стандартных схем программ.

Логико-термальная эквивалентность

По сути дела, детерминант схемы программы $Det(\pi)$ — это просто глобальная синтаксическая характеристика схемы π (наподобие ее размера, класса изоморфизма и др.).

Эта характеристика принимает во внимание только те особенности синтаксического устройства схемы, которые оказывают влияние на реализуемость путей и результат реализующего вычисления.

Таким образом, проблема проверки л-т эквивалентности схем программ — это задача синтаксического анализа программ.

А, как нам известно, многие задачи синтаксического анализа программ имеют успешное и эффективное решение.

Попробуем построить алгоритм проверки л-т эквивалентности стандартных схем программ.

Граф совместных вычислений схем программ

А как решают подобную задачу в других моделях вычислений? С чего разумно начать?

Для проверки эквивалентности конечных автоматов $\mathcal{A}_1$ и $\mathcal{A}_2$ можно построить их декартово произведение $\mathcal{A}_1 \times \mathcal{A}_2$ — пути в системе переходов $\mathcal{A}_1 \times \mathcal{A}_2$ представляют совместные вычисления автоматов $\mathcal{A}_1$ и $\mathcal{A}_2$ на одних и тех же входных словах.

Различие в поведении неэквивалентных автоматов проявляется в том, что они по-разному реагируют на некоторое входное слово, и это будет заметно в одном из совместных вычислений этих автоматов. Значит, неэквивалентность автоматов $\mathcal{A}_1$ и $\mathcal{A}_2$ может быть обнаружена в системе переходов $\mathcal{A}_1 \times \mathcal{A}_2$.

Попробуем ввести подходящим образом граф, представляющий совместные вычисления стандартных схем программ, и использовать его для обнаружения различий в их логико-термальных характеристиках.

Граф совместных вычислений схем программ

Пути в системах переходов $LTS(\pi_1)$ и $LTS(\pi_2)$

$$
\begin{aligned}
tr_1 &= v'_0 \xrightarrow{\theta'_0} v'_1 \xrightarrow{\delta'_1,\theta'_1} v'_2 \xrightarrow{\delta'_2,\theta'_2} \cdots \xrightarrow{\delta'_n,\theta'_n} v'_{n+1} \\
tr_2 &= v''_0 \xrightarrow{\theta''_0} v''_1 \xrightarrow{\delta''_1,\theta''_1} v''_2 \xrightarrow{\delta''_2,\theta''_2} \cdots \xrightarrow{\delta''_n,\theta''_n} v'_{n+1}
\end{aligned}
$$

будем называть совместными , если для каждого $i, 1 \leq i \leq n$, справедливо равенство $\delta'_i = \delta''_i$.

Чтобы проверить л-т эквивалентность схем программ π_1 и π_2 , нужно сравнить л-т истории всех пар совместных путей в системах переходов для этих схем. Поэтому все пары совместных путей целесообразно собрать в одной конечной структуре — графе совместных вычислений.

Граф совместных вычислений

Чтобы построить граф совместных вычислений для схем программ $LTS(\pi_1)$ и $LTS(\pi_2)$ нужно вначале сделать эти схемы независимыми друг от друга по памяти.

Предварительная подготовка схем программ

Пусть имеется пара схем программ над множеством переменных $X = \{x_1, \ldots, x_n\}$, и эти схемы представлены размеченными системами переходов $LTS(\pi_1)$ и $LTS(\pi_2)$.

Переименуем все вхождения переменных $x_1, \ldots, x_n$ в первой схеме программ π_1 в $x'_1, \ldots, x'_n$, а все вхождения этих же переменных во второй схеме программ π_2 в $x''_1, \ldots, x''_n$. Полученные в результате проведенных переименований системы переходов обозначим $LTS(\pi')$ и $LTS(\pi'')$.

Чтобы иметь возможность сравнивать результаты вычислений схем программ, будем использовать подстановку

$$\eta_0 = \{x'_1/x_1, \ldots, x'_n/x_n,\ x''_1/x_1, \ldots, x''_n/x_n\} ,$$

обратную по отношению к проведенным переименованиям.

Граф совместных вычислений

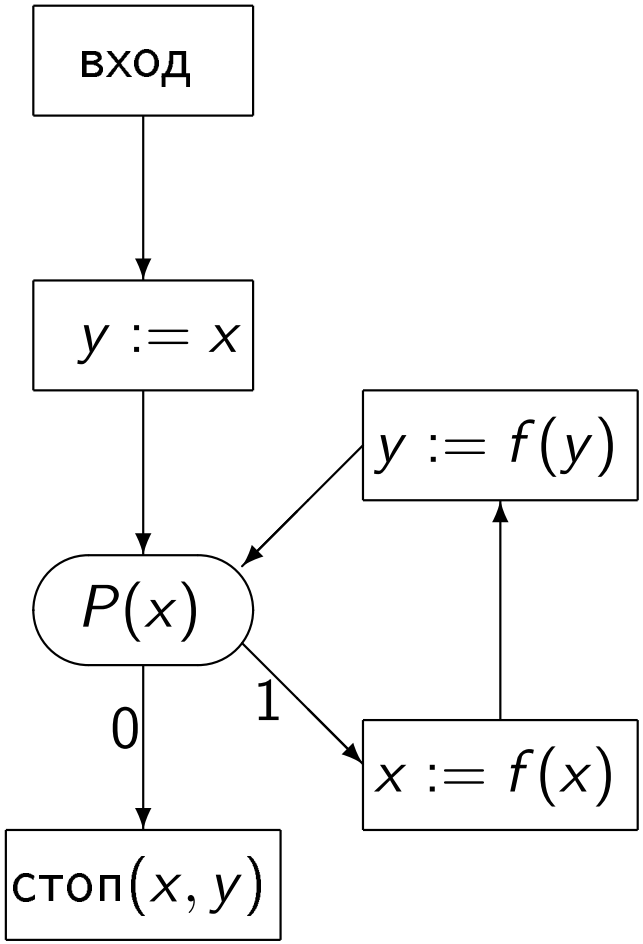


Схема π_1

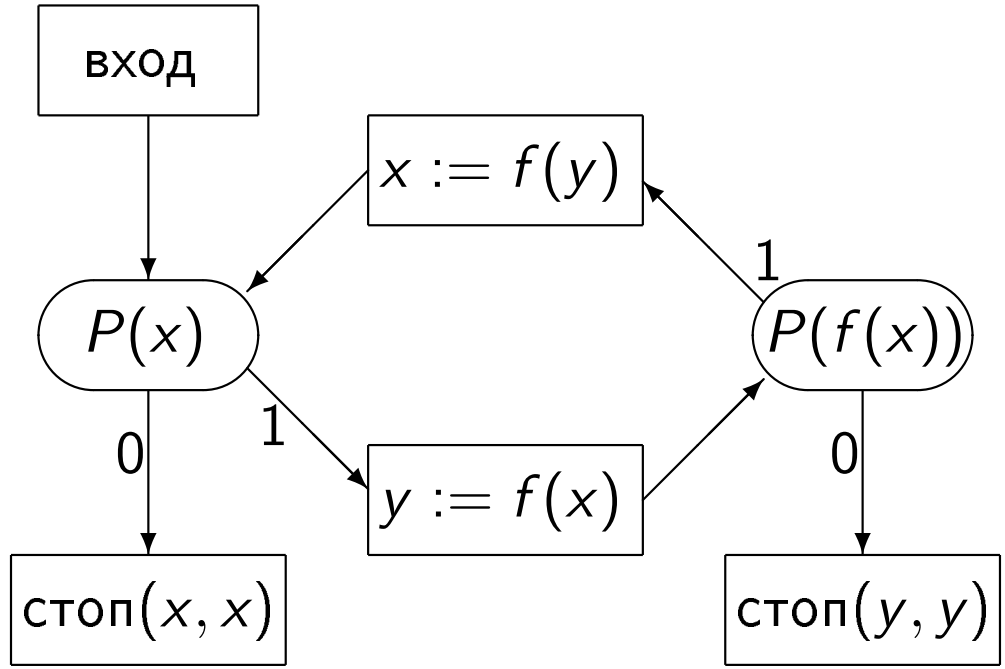
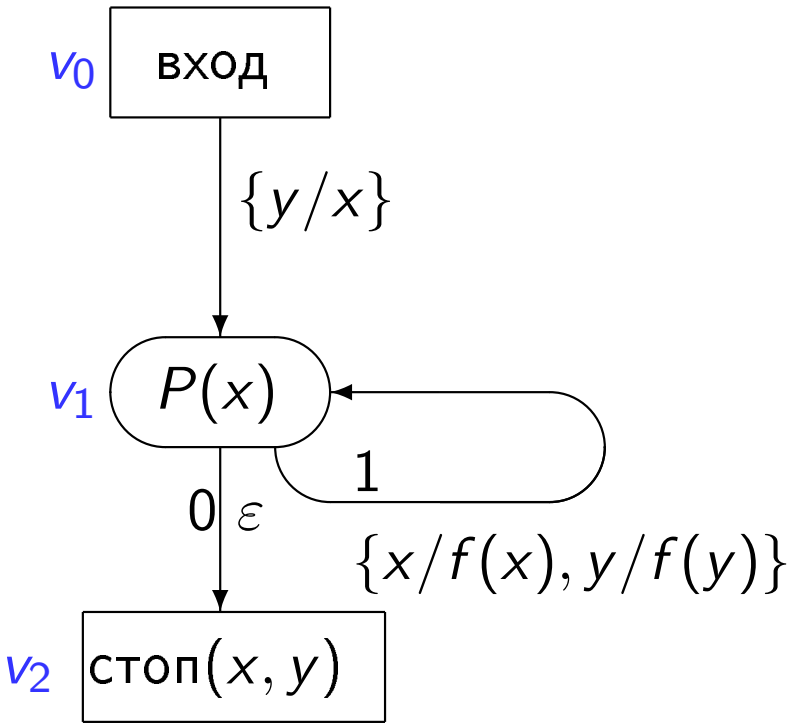
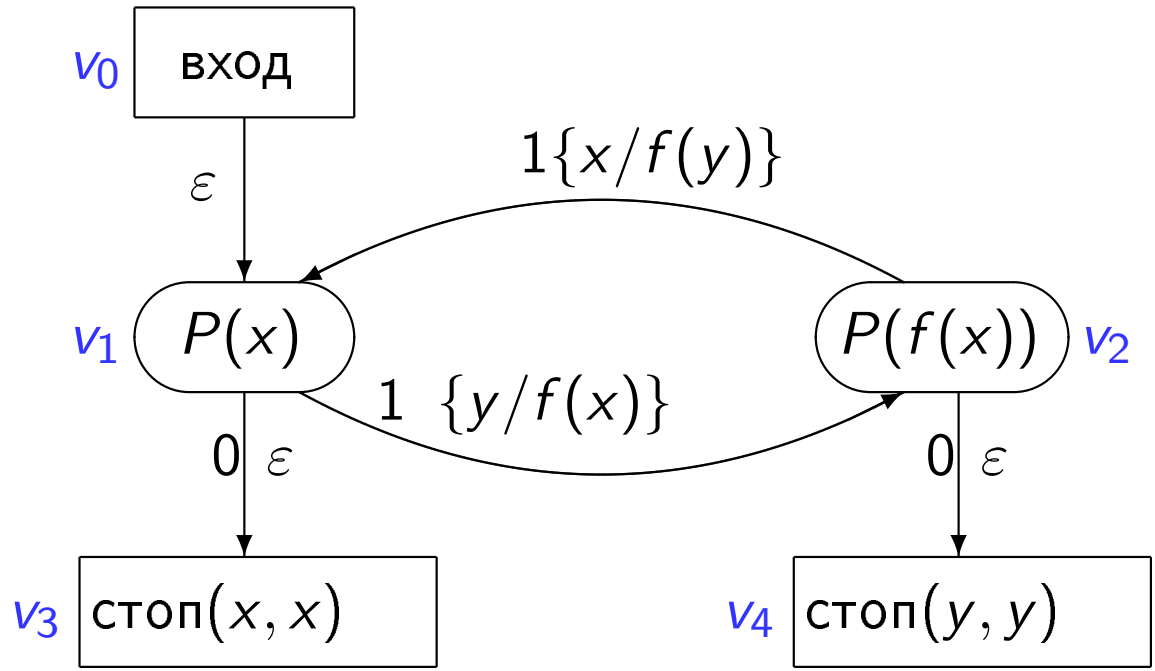


Схема π_2

Граф совместных вычислений



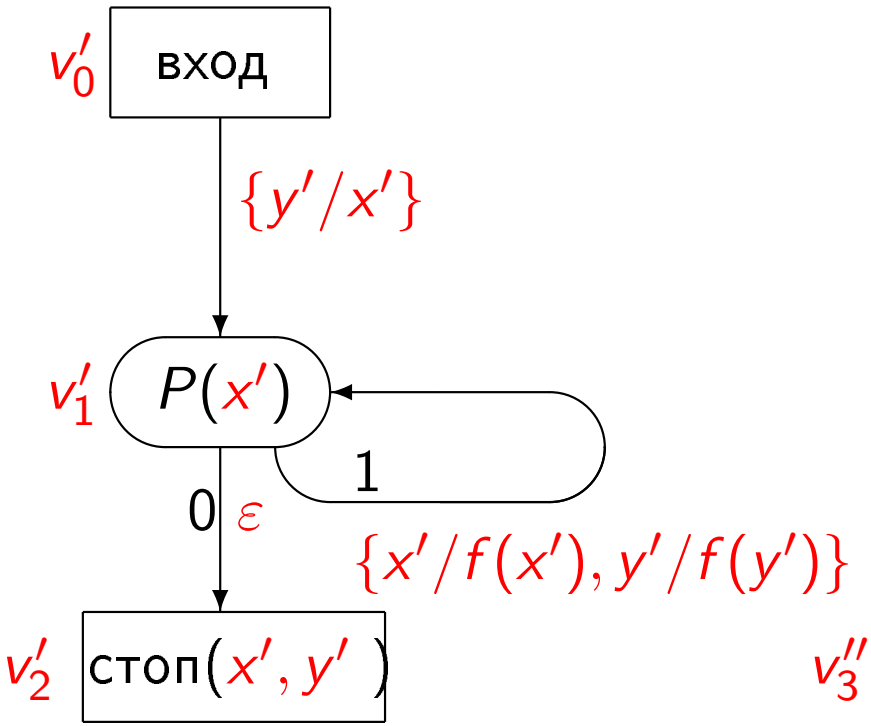
Система переходов $LTS(\pi_1)$



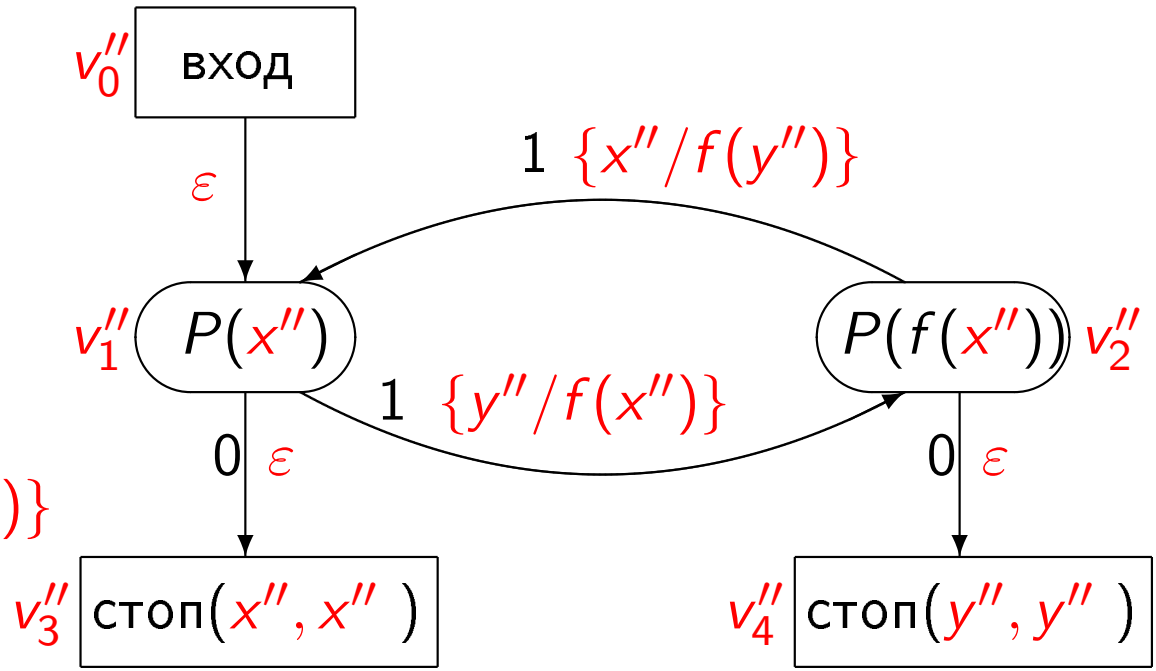
Система переходов $LTS(\pi_2)$

Граф совместных вычислений

$$\eta_0 = \{x'/x, y'/y,\ x''/x, y''/y\}$$



Система переходов $LTS(\pi')$



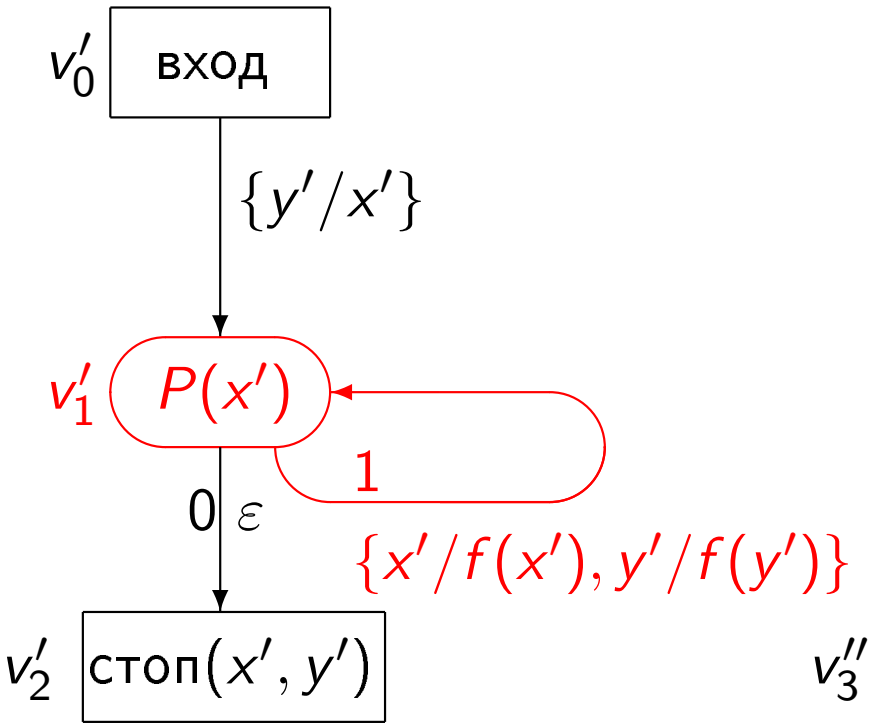
Система переходов $LTS(\pi'')$

Граф совместных вычислений

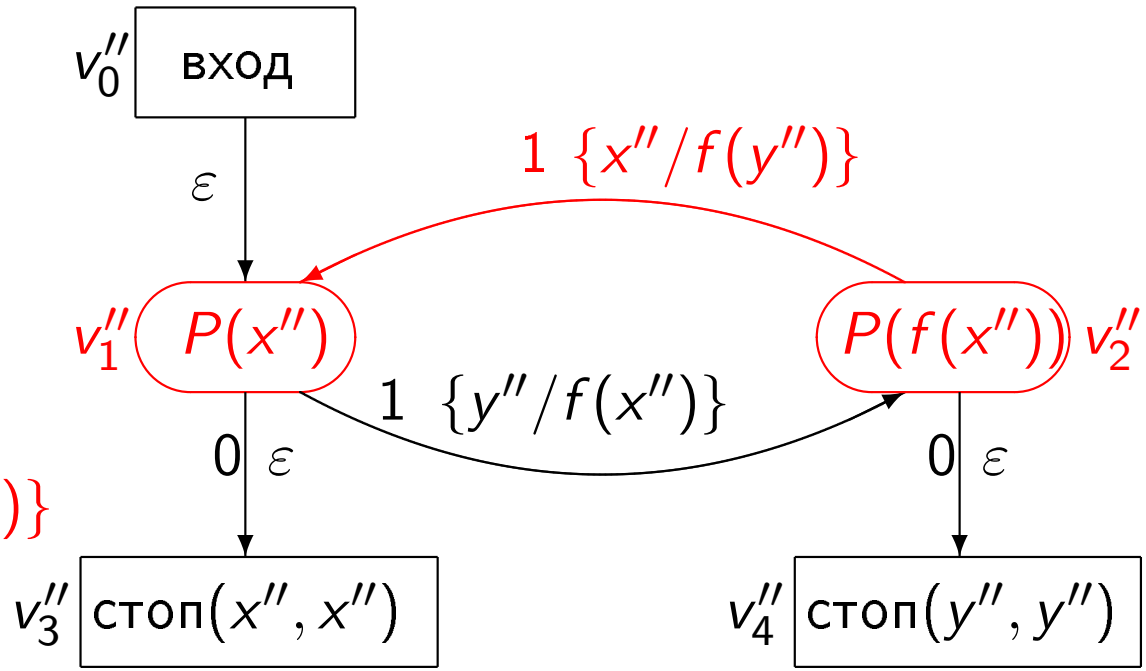
Граф совместных вычислений $\Gamma[\pi_1, \pi_2]$ схем программ, представленных размеченными системами переходов $LTS(\pi')$ и $LTS(\pi'')$ с переименованными переменными, — это наименьший размеченный граф, удовлетворяющий следующим условиям:

1. граф $\Gamma[\pi', \pi'']$ содержит вершину $u_0 = (v'_0, v''_0)$, где v'_0, v''_0 — начальные вершины схем π' и π'' ;
2. если в схемах π' и π'' из вершин v'_0 и v''_0 исходят дуги $v'_0 \xrightarrow{\theta'_0} v'_1$ и $v''_0 \xrightarrow{\theta''_0} v''_1$, то в графе $\Gamma[\pi_1, \pi_2]$ из вершины $u_0 = (v'_0, v''_0)$ в вершину $u_1 = (v'_0, v''_0)$ исходит дуга, помеченная подстановкой $\theta'_0 \cup \theta''_0$;
3. если граф $\Gamma[\pi_1, \pi_2]$ содержит вершину $u_i = (v'_i, v''_i)$, и в схемах π' и π'' из вершин v'_i и v''_i исходят дуги $v'_i \xrightarrow{\delta, \theta'_i} v'_j$ и $v''_i \xrightarrow{\delta, \theta''_i} v''_j$ для некоторого $\delta, \delta \in \{0, 1\}$, то в графе $\Gamma[\pi_1, \pi_2]$ из вершины $u_i = (v'_i, v''_i)$ в вершину $u_j = (v'_j, v''_j)$ исходит дуга, помеченная битом δ и подстановкой $\theta'_i \cup \theta''_i$.

Граф совместных вычислений

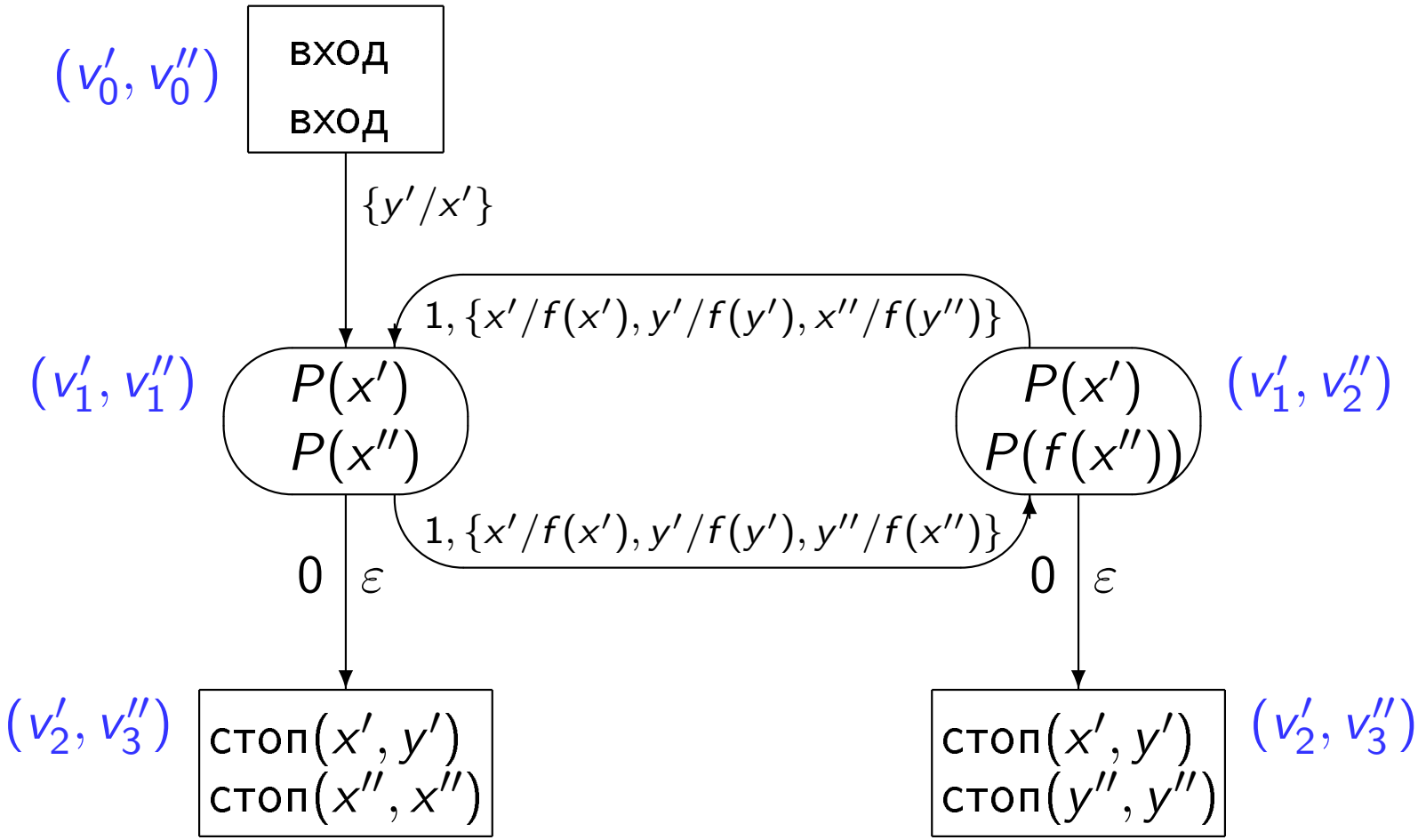


Система переходов $LTS(\pi')$



Система переходов $LTS(\pi'')$

Граф совместных вычислений



Граф совместных вычислений $\Gamma[\pi_1, \pi_2]$

Граф совместных вычислений

Вершину $u_0 = (v'_0, v''_0)$, с которой начинается построение графа $\Gamma[\pi_1, \pi_2]$, будем называть корневой вершиной графа.
Как видно из описания правил построения графа $\Gamma[\pi_1, \pi_2]$, каждый корневой маршрут в графе совместных вычислений

$$\alpha = (v'_0, v''_0) \xrightarrow{\theta_0} (v'_1, v''_1) \xrightarrow{\delta_1,\theta_1} (v'_2, v''_2) \xrightarrow{\delta_2,\theta_2} \cdots \xrightarrow{\delta_n,\theta_n} (v'_{n+1}, v''_{n+1})$$

взаимно однозначно соответствует паре совместных путей в системах переходов $LTS(\pi')$ и $LTS(\pi'')$

$$\begin{aligned} tr_1 &= v'_0 \xrightarrow{\theta'_0} v'_1 \xrightarrow{\delta_1,\theta'_1} v'_2 \xrightarrow{\delta_2,\theta'_2} \cdots \xrightarrow{\delta_n,\theta'_n} v'_{n+1} \\ tr_2 &= v''_0 \xrightarrow{\theta''_0} v''_1 \xrightarrow{\delta_1,\theta''_1} v''_2 \xrightarrow{\delta_2,\theta''_2} \cdots \xrightarrow{\delta_n,\theta''_n} v'_{n+1} \end{aligned}$$

для котрых $\theta_i = \theta'_i \cup \theta''_i$ для всех $i, 0 \leq i \leq n$.

Композицию $\theta_n \cdots \theta_1 \theta_0 \eta_0$ подстановки η_0 , восстанавливающей имена переменных $x_1, \ldots, x_n$ в схемах π_1, π_2 , и подстановок $\theta_0, \theta_1, \cdots \theta_n$, приписанных дугам коревого маршрута α в графе совместных вычислений, обозначим записью $\theta[\alpha]$.

Граф совместных вычислений

Предположим, что в схемах программ π_1, π_2 все вершины существенны, т.е. каждая из них лежит на некотором пути из начальной вершины в финальную.

Тогда справедлива следующая

Теорема 2.

Схемы программ π_1 и π_2 л-т эквивалентны тогда и только тогда, когда для любого корневого маршрута

$$\alpha = (v_0', v_0'') \xrightarrow{\theta_0} (v_1', v_1'') \xrightarrow{\delta_1,\theta_1} (v_2', v_2'') \xrightarrow{\delta_2,\theta_2} \cdots \xrightarrow{\delta_n,\theta_n} (v_{n+1}', v_{n+1}'')$$

в графе совместных вычислений $\Gamma[\pi_1, \pi_2]$ верно равенство

$$B'(v_{n+1}')\theta(\alpha) = B''(v_{n+1}'')\theta(\alpha),$$

где $B'(v_{n+1}')$ и $B''(v_{n+1}'')$ — атомарные формулы, приписанные вершинам v_{n+1}' и v_{n+1}'' в схемах π' и π'' соответственно.

Граф совместных вычислений

Доказательство.

Если схемы программ π_1 и π_2 не содержат несущественных вершин, то $\pi_1 \overset{lt}{\sim} \pi_2$

$$\Longleftrightarrow$$

в $LTS(\pi'), LTS(\pi'')$ для любой пары совместных путей

$$\begin{aligned} tr_1 &= v'_0 \xrightarrow{\theta'_0} v'_1 \xrightarrow{\delta_1,\theta'_1} v'_2 \xrightarrow{\delta_2,\theta'_2} \cdots \xrightarrow{\delta_n,\theta'_n} v'_{n+1} \\ tr_2 &= v''_0 \xrightarrow{\theta''_0} v''_1 \xrightarrow{\delta_1,\theta''_1} v''_2 \xrightarrow{\delta_2,\theta''_2} \cdots \xrightarrow{\delta_n,\theta''_n} v'_{n+1} \end{aligned}$$

верно равенство $B'(v'_{n+1})\theta'_n \cdots \theta'_1\theta'_0\eta_0 = B'(v''_{n+1})\theta''_n \cdots \theta''_1\theta''_0\eta_0$

$$\Longleftrightarrow$$

для любого корневого маршрута

$$\alpha = (v'_0, v''_0) \xrightarrow{\theta_0} (v'_1, v''_1) \xrightarrow{\delta_1,\theta_1} (v'_2, v''_2) \xrightarrow{\delta_2,\theta_2} \cdots \xrightarrow{\delta_n,\theta_n} (v'_{n+1}, v''_{n+1})$$

в графе совместных вычислений $\Gamma[\pi_1, \pi_2]$ верно равенство $B'(v'_{n+1})\theta(\alpha) = B''(v''_{n+1})\theta(\alpha)$

QED

Граф совместных вычислений

Теорема 2 сводит проверку л-т эквивалентности схем программ π_1, π_2 к проверке равенства атомарных формул

$$B'(v')\theta(\alpha) = B''(v'')\theta(\alpha)$$

для каждой вершины $u = (v', v'')$ и каждого корневого маршрута α , ведущего в эту вершину в графе совместных вычислений $\Gamma[\pi_1, \pi_2]$.

Однако в графе $\Gamma[\pi_1, \pi_2]$ может быть бесконечно много корневых маршрутов. Как проверить это равенство для бесконечно большого числа подстановок $\theta(\alpha)$?

А как проверить, что все слова бесконечного языка обладают некоторым нужным свойством?

Можно построить общий шаблон для всех слов этого языка (например, в форме регулярного выражения), и затем показать, что этот шаблон удовлетворяет требуемому свойству. Попробуем применить этот прием к подстановкам.

Антиунификация подстановок

На множестве подстановок *Subst* определим отношение сравнения $\preceq$:

$$\theta_1 \preceq \theta_2 \Leftrightarrow \exists\, \rho \ : \theta_2 = \theta_1 \rho$$

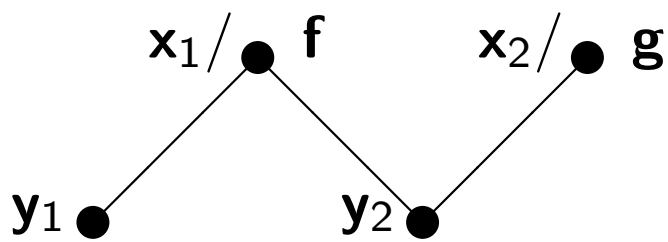
Если $\theta_1 \preceq \theta_2$, то подстановка θ_2 называется примером подстановки θ_1 , а подстановка θ_1 — шаблоном подстановки θ_1 .

Например, если

$\theta_1 = \{x_1/f(y_1, y_2), x_2/g(y_2)\}$,
$\theta_2 = \{x_1/f(c, g(z)), x_2/g(g(z))\}$,

то $\theta_1 \preceq \theta_2$, поскольку $\theta_2 = \theta_1\{y_1/c, y_2/g(z)\}$.

θ_1 :



Антиунификация подстановок

На множестве подстановок *Subst* определим отношение сравнения $\preceq$:

$$\theta_1 \preceq \theta_2 \Leftrightarrow \exists\, \rho \ : \theta_2 = \theta_1 \rho$$

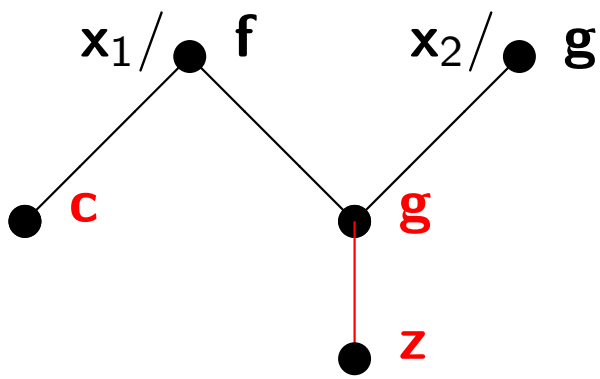
Если $\theta_1 \preceq \theta_2$, то подстановка θ_2 называется примером подстановки θ_1 , а подстановка θ_1 — шаблоном подстановки θ_1 .

Например, если

$\theta_1 = \{x_1/f(y_1, y_2), x_2/g(y_2)\}$,
$\theta_2 = \{x_1/f(c, g(z)), x_2/g(g(z))\}$,

то $\theta_1 \preceq \theta_2$, поскольку $\theta_2 = \theta_1\{y_1/c, y_2/g(z)\}$.

θ_2 :



Антиунификация подстановок

На множестве подстановок *Subst* определим отношение сравнения $\preceq$:

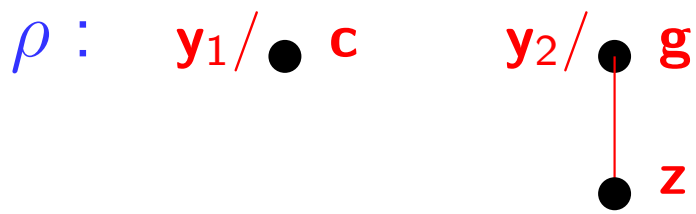
$$\theta_1 \preceq \theta_2 \;\Leftrightarrow\; \exists\, \rho \;: \theta_2 = \theta_1 \rho$$

Если $\theta_1 \preceq \theta_2$, то подстановка θ_2 называется примером подстановки θ_1 , а подстановка θ_1 — шаблоном подстановки θ_1 .

Например, если

$\theta_1 = \{x_1/f(y_1, y_2), x_2/g(y_2)\}$,
$\theta_2 = \{x_1/f(c, g(z)), x_2/g(g(z))\}$,

то $\theta_1 \preceq \theta_2$, поскольку $\theta_2 = \theta_1\{y_1/c, y_2/g(z)\}$.



Антиунификация подстановок

На множестве подстановок *Subst* определим отношение сравнения $\preceq$:

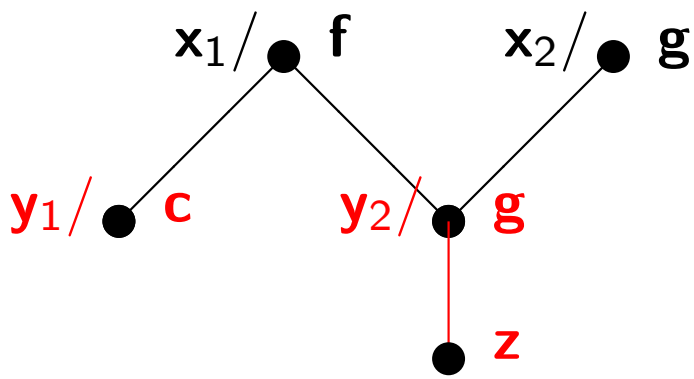
$$\theta_1 \preceq \theta_2 \Leftrightarrow \exists\, \rho \ : \theta_2 = \theta_1 \rho$$

Если $\theta_1 \preceq \theta_2$, то подстановка θ_2 называется примером подстановки θ_1 , а подстановка θ_1 — шаблоном подстановки θ_1 .

Например, если

$\theta_1 = \{x_1/f(y_1, y_2), x_2/g(y_2)\}$,
$\theta_2 = \{x_1/f(c, g(z)), x_2/g(g(z))\}$,

то $\theta_1 \preceq \theta_2$, поскольку $\theta_2 = \theta_1\{y_1/c, y_2/g(z)\}$.



θ_2 — пример θ_1

θ_1 — шаблон θ_2

Антиунификация подстановок

Наименьшим элементом в множестве подстановок $Subst$ по отношению сравнения $\preceq$ является тождественная подстановка $\varepsilon = \{x_1/x_1, x_2/x_2, \ldots, x_n/x_n\}$, поскольку для любой подстановки θ верно равенство $\theta = \varepsilon\theta$.

А вот наибольшей подстановки в множестве $Subst$ нет: некоторые пары подстановок (например, $\eta' = \{x/c\}$ и $\eta'' = \{x/d\}$) могут не иметь общего примера.

Во избежание «неравноправия» добавим к множеству $Subst$ новую мнимую подстановку τ , удовлетворяющую равенствам

$$\theta\tau = \tau\theta = \tau \quad \text{и} \quad E'\tau = E''\tau$$

для любой подстановки θ и выражений E', E'' .

Расширенное таким образом множество подстановок обозначим записью $Subst^\tau$.

Антиунификация подстановок

Подстановки θ' и θ'' назовем подобными , если существует такая подстановка-переименование ρ , для которой справедливо равенство $\theta'' = \theta'\rho$.

Отношение подобия обозначим записью $\theta' \approx \theta''$.

Утверждение 2.

Для любой пары подстановок θ', θ'' верно

$$\theta' \preceq \theta'' \ \wedge \ \theta'' \preceq \theta' \ \Rightarrow \ \theta' \approx \theta''.$$

Доказательство.

Очевидно следует из определения отношений $\preceq$ и $\approx$.

Антиунификация подстановок

Упорядоченное множество подстановок $(Subst^{\tau}, \preceq)$ обладает свойством обрыва убывающих цепей.

Утверждение 3.

Операция композиции подстановок монотонна относительно сравнения $\preceq$:

$$\theta_1 \preceq \theta_2 \;\Rightarrow\; \eta\theta_1 \preceq \eta\theta_2$$

Доказательство.

Очевидно следует из определения операции композиции подстановок и отношения $\preceq$.

Антиунификация подстановок

Упорядоченное множество подстановок $(Subst^{\tau}, \preceq)$ обладает свойством обрыва убывающих цепей.

Утверждение 4.

Не существует бесконечной убывающей последовательности попарно неподобных друг другу подстановок

$$\theta_1 \succeq \theta_2 \succeq \theta_3 \succeq \cdots$$

Доказательство.

Каждую подстановку $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ охарактеризуем натуральным числом $h(\theta) = h_1(\theta) + h_2(\theta)$, где

- $h_1(\theta)$ — это суммарное число вхождений функциональных символов и констант в термы $t_1, \ldots, t_n$,
- $h_2(\theta) = \sum\limits_{y \in Var(t_1, \ldots, t_n)} ((n(y, t_1) + \cdots + n(y, t_n) - 1)$, где $n(y, t)$ — кратность вхождения переменной y в терм t .

Остается лишь заметить, что для любой пары подстановок θ', θ'' верно $\theta' \preceq \theta'' \ \wedge \ \theta' \not\approx \theta'' \ \Rightarrow \ h(\theta') < h(\theta'')$ QED

Антиунификация подстановок

На упорядоченном множестве подстановок $(Subst^{\tau}, \preceq)$, помимо операции композиции, можно ввести еще две операции:

1. операцию вычисления точной верхней грани $\theta_1 \wedge \theta_2$, т.е. такого наиболее простого общего примера $\eta : \theta_1 \preceq \eta,\ \theta_2 \preceq \eta$, который не превосходит других общих примеров этой пары подстановок;
2. операцию вычисления точной нижней грани $\theta_1 \vee \theta_2$, т.е. такого наиболее специального общего шаблона $\lambda : \lambda \preceq \theta_1,\ \lambda \preceq \theta_2$, который не уступает другим общим шаблонам этой пары подстановок.

Операцию вычисления точной верхней грани также называют унификацией . Это та самая унификация, которая используется в резолютивном выводе и в логическом программировании. Для выполнения этой операции можно применять любой известный алгоритм унификации (например, алгоритм Мартелли-Монтанари).

Антиунификация подстановок

Пусть заданы две подстановки $\theta' = \{x_1/t'_1, \ldots, x_n/t'_n\}$ и $\theta'' = \{x_1/t''_1, \ldots, x_n/t''_n\}$ из множества $Subst$, и пусть Z — множество вспомогательных переменных, не входящих в состав термов из области значений подстановок θ' и θ'' .

Алгоритм атиунификации строит неубывающую последовательность подстановок $\eta_0, \eta_1, \ldots$, проводя преобразование системы аннотированных уравнений, порожденных подстановками θ' и θ'' . Каждое уравнение системы помечается некоторой вспомогательной переменной из множества Z .

В начале работы алгоритма система аннотированных уравнений имеет вид

$$\begin{array}{lccc} z_1: & t'_1 & = & t''_1 \\ & \ldots & \ldots & \ldots \\ z_n: & t'_n & = & t''_n \end{array}$$

а $\eta_0 = \{x_1/z_1, \ldots, x_n/z_n\}$.

Антиунификация подстановок

На каждом i -ом шаге работы алгоритма к системе уравнений, до тех пор пока это возможно, в произвольном порядке применяются следующие 2 правила переписывания уравнений и вычисления последовательности подстановок $\eta_0, \eta_1, \ldots$.

1. Аннотированное уравнение $z: \; f(s'_1, \ldots, s'_k) = f(s''_1, \ldots, s''_k)$ замещается системой уравнений

$$\begin{array}{lccc} z_{N+1}: & s'_1 & = & s''_1 \\ & \ldots & \ldots & \ldots \\ z_{N+k}: & s'_k & = & s''_k \end{array}$$

где $z_{N+1}, \ldots, z_{N+k}$ — переменные из множества Z , ранее не использованные для разметки других уравнений; при этом $\eta_i = \eta_{i-1}\{z/f(z_{N+1}, \ldots, z_{N+k})\}|_{x_1,\ldots,x_n}$.

2. Если в системе есть пара одинаковых аннотированных уравнений $z_i : s' = s''$ и $z_j : s' = s''$, то одно из них (например, $z_j : s' = s''$) удаляется; при этом $\eta_i = \eta_{i-1}\{z_j/z_i\}|_{x_1,\ldots,x_n}$.

Антиунификация подстановок

Как только будет построена такая система аннотированных уравнений, к которой не применимо ни одно из указанных двух правил, алгоритм прекращает работу. Результатом его работы является последняя из построенных подстановок $\eta_N = \theta' \vee \theta''$.

Вычисление $\theta' \vee \theta''$, где
$\theta' = \{x_1/f(y_1, g(y_2)), x_2/g(y_2)\}$
$\theta'' = \{x_1/f(g(y_2), f(y_1, y_2)), x_2/f(y_1, y_2)\}$.

$$\mathcal{E}_0 : \begin{cases} z_1 : & f(y_1, g(y_2)) = f(g(y_2), f(y_1, y_2)) \\ z_2 : & g(y_2) = f(y_1, y_2) \end{cases}$$

$\eta_0 = \{x_1/z_1, x_2/z_2\}$

$$\Downarrow (1)$$

$$\mathcal{E}_1 : \begin{cases} z_3 : & y_1 = g(y_2) \\ z_4 : & g(y_2) = f(y_1, y_2) \\ z_2 : & g(y_2) = f(y_1, y_2) \end{cases}$$

$\eta_1 = \eta_0\{z_1/f(z_3, z_4)\}|_{x_1,x_2} = \{x_1/f(z_3, z_4), x_2/z_2\}$

Антиунификация подстановок

$$\mathcal{E}_1 : \begin{cases} z_3 : & y_1 = g(y_2) \\ z_4 : & g(y_2) = f(y_1, y_2) \\ z_2 : & g(y_2) = f(y_1, y_2) \end{cases}$$

$\eta_1 = \{x_1/f(z_3, z_4), x_2/z_2\}$

$\Downarrow (2)$

$$\mathcal{E}_2 : \begin{cases} z_3 : & y_1 = g(y_2) \\ z_4 : & g(y_2) = f(y_1, y_2) \end{cases}$$

$\eta_2 = \eta_1\{z_2/z_4\}|_{x_1,x_2} = \{x_1/f(z_3, z_4), x_2/z_4\}$

КОНЕЦ: $\theta' \vee \theta'' = \{x_1/f(z_3, z_4), x_2/z_4\}$

Антиунификация подстановок

Утверждение 4.

Предложенный алгоритм антиунификации, получив на входе произвольную пару подстановок θ' и θ'' , всегда завершает свое выполнение и вычисляет подстановку $\eta = \theta' \vee \theta''$.

Доказательство.

Самостоятельно, опираясь на определение наиболее специального шаблона пары подстановок и следуя плану доказательства завершаемости и корректности алгоритма унификации Мартелли-Монтанари. QED

Антиунификация подстановок

Операция антиунификации обладает хорошо известными алгебраическими свойствами

Утверждение 5.

Для любых подстановок $\theta_1, \theta_2, \theta_3$ из множества $Subst^\tau$ выполняются законы

коммутативности: $\theta_1 \vee \theta_2 \approx \theta_2 \vee \theta_1$,

ассоциативности: $\theta_1 \vee (\theta_2 \vee \theta_3) \approx (\theta_1 \vee \theta_2) \vee \theta_3$,

идемпотентности: $\theta_1 \vee \theta_1 \approx \theta_1$,

левой дистрибутивности: $\theta_1(\theta_2 \vee \theta_3) \approx \theta_1\theta_2 \vee \theta_1\theta_3$.

Доказательство.

Самостоятельно, опираясь на определение операции антиунификации и комопозиции подстановок.

Антиунификация подстановок

Операцию антиунификации можно использовать для проверки равенства выражений.

Утверждение 6.

Для любых выражений E_1, E_2 и подстановок θ', θ'' справедливо соотношение

$$E_1\theta' = E_2\theta' \ \wedge \ E_1\theta'' = E_2\theta'' \ \Leftrightarrow \ E_1(\theta' \vee \theta'') = E_2(\theta' \vee \theta'').$$

Доказательство.

($\Rightarrow$) Если $E_1\theta' = E_2\theta'$ и $E_1\theta'' = E_2\theta''$, то атомы E_1, E_2 унифицируемы и имеют наиболее общий унификатор μ, для которого верны неравенства $\mu \preceq \theta'$ и $\mu \preceq \theta''$.

Операция антиунификации вычисляет точную нижнюю грань $\theta' \vee \theta''$ подстановок θ', θ'', и поэтому $\mu \preceq \theta' \vee \theta''$.

Следовательно, подстановка $\theta' \vee \theta''$ унифицирует атомы E_1, E_2, т. е. верно равенство $E_1(\theta' \vee \theta'') = E_2(\theta' \vee \theta'')$.

($\Leftarrow$) Очевидно в силу определения точной нижней грани подстановок.

QED

Алгоритм проверки логико-термальной эквивалентности стандартных схем программ

Теорема 2 позволила свести задачу проверки л-т эквивалентности пары стандартных схем программ π' и π'' к задаче проверке свойств композиций подстановок для каждого пути α в графе совместных вычислений $\Gamma[\pi', \pi'']$ этих схем программ:

$$\pi' \overset{lt}{\sim} \pi'' \Leftrightarrow \forall (v', v'') \in V_{\pi', \pi''} \; \forall \alpha \in Path(v', v'') : B(v')\theta(\alpha) = B(v'')\theta(\alpha)$$

Утверждение 6 позволяет упростить анализ композиций подстановок в бесконечном множестве путей $Path(v', v'')$ для каждой вершины $w = (v', v'')$ в графе $\Gamma[\pi', \pi'']$:

$$\forall \alpha \in Path(v', v'') \; : \; B(v')\theta(\alpha) = B(v'')\theta(\alpha) \Leftrightarrow B(v')\theta_w = B(v'')\theta_w,$$

где $\theta_w = \bigvee\limits_{\alpha \in Path(v', v'')} \theta(\alpha)$.

Алгоритм проверки эквивалентности

Таким образом, для эффективной проверки логико-термальной эквивалентности схем программ $\pi' \overset{lt}{\sim} \pi''$ достаточно научиться эффективно вычислять наиболее специальный шаблон

$$\theta_w = \bigvee_{\alpha \in Path(v', v'')} \theta(\alpha)$$

подстановок $\theta(\alpha)$, ассоциированных со всеми возможными путями $\alpha \in Path(v', v'')$, ведущими в заданную вершину $w = (v', v'')$ графа совместных вычислений $\Gamma[\pi', \pi'']$.

Покажем, как можно итеративно вычислить все подстановки-шаблоны θ_w , опираясь на алгоритм антиунификации подстановок и закон левой дистрибутивности композиции подстановок относительно операции антиунификации:

$$\theta_1(\theta_2 \vee \theta_3) \approx \theta_1\theta_2 \vee \theta_1\theta_3.$$

Алгоритм проверки эквивалентности

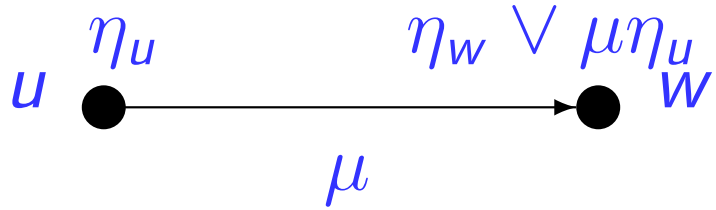
В процедуре глобальной разметки графа $\Gamma[\pi', \pi'']$ каждой вершине $w = (v', v'')$ этого графа приписывается подстановка η_w , которая приближает сверху искомую точную нижнюю грань $\theta_w = \bigvee\limits_{\alpha \in Path(v', v'')} \theta(\alpha)$.

Это приближение уточняется по ходу работы алгоритма. В начале работы процедуры корневой вершине $w_0 = (\text{вход}, \text{вход})$ приписывается подстановка $\{x'_1/z_1, \ldots, x'_n/z_n, x''_1/z_1, \ldots, x''_n/z_n\}$, а всем остальным вершинам $w, w \neq w_0$, приписывается наибольшая в решетке подстановок мнимая подстановка τ .

Алгоритм проверки эквивалентности

Далее, до тех пор пока это возможно, применяется следующее правило переписывания подстановок η_w , которыми помечены вершины графа:

> если в графе $\Gamma[\pi', \pi'']$ существует дуга $u \xrightarrow{\mu} w$, для которой не выполняется неравенство $\eta_w \preceq \theta\eta_u$, то вершине w вместо подстановки η_w приписывается подстановка $\eta_w \vee \mu\eta_u$.



Процедура переписывания завершает работу в том случае, если для всех дуг $u \overset{\mu}{\Rightarrow} w$ графа $\Gamma[\pi', \pi'']$ выполняется неравенство $\eta_w \preceq \mu\eta_u$.

Алгоритм проверки эквивалентности

Теорема 4.

Каковы бы ни были программы π' и π'' процедура глобальной разметки графа совместных вычислений $\Gamma[\pi', \pi'']$ завершает свою работу и вычисляет в каждой вершине $w = (v', v'')$ подстановку $\theta_w = \bigvee_{\alpha \in Path(v', v'')} \theta(\alpha)$.

Доказательство.

1. Применение правила переписывания к разметке графа $\Gamma[\pi', \pi'']$ приводит к строгому убыванию пометок в одной из вершин графа:
новая подстановка $\eta_w \vee \mu\eta_u$, которая будет приписана одной из вершин w , строго меньшее подстановки η_w , которая была приписана этой вершине до применения правила.

Таким образом, свойство обрыва убывающих цепей (Утверждение 3) гарантирует завершаемость работы процедуры переписывания.

Алгоритм проверки эквивалентности

Доказательство.

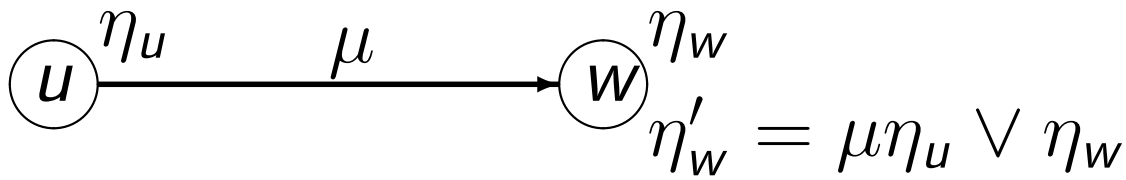
2. Воспользовавшись индукцией по числу шагов процедуры и свойством левой дистрибутивности композиции подстановок относительно операции антиунификации, можно показать, что на каждом шаге работы процедуры неравенство

$$\bigvee_{\alpha \in Path(v', v'')} \theta(\alpha) \preceq \eta_w$$

выполняется для любой вершины $w = (v', v'')$ графа $\Gamma[\pi', \pi'']$.

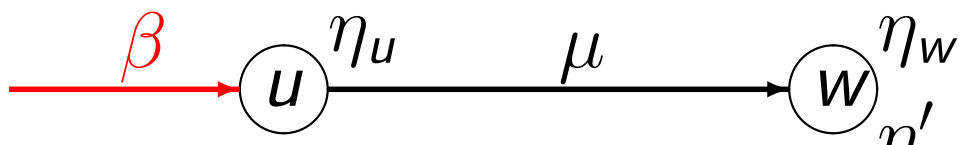
Алгоритм проверки эквивалентности

2. Действительно, если правило переписывания разметок было применено к дуге $u \xrightarrow{\mu} w$, то



Алгоритм проверки эквивалентности

2. Действительно, если правило переписывания разметок было применено к дуге $u \xrightarrow{\mu} w$, то

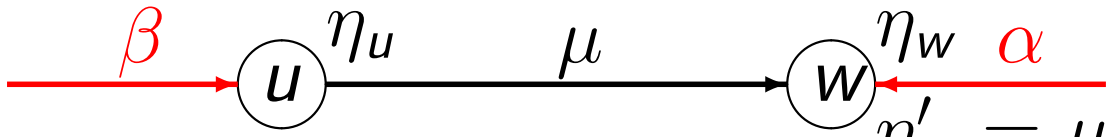


$$\bigvee_{\beta \in Path(u)} \theta(\beta) \preceq \eta_u \qquad (1) \qquad \eta'_w = \mu\eta_u \vee \eta_w$$

Индуктивная гипотеза для вершины u

Алгоритм проверки эквивалентности

2. Действительно, если правило переписывания разметок было применено к дуге $u \xrightarrow{\mu} w$, то



$$\eta'_w = \mu\eta_u \vee \eta_w$$

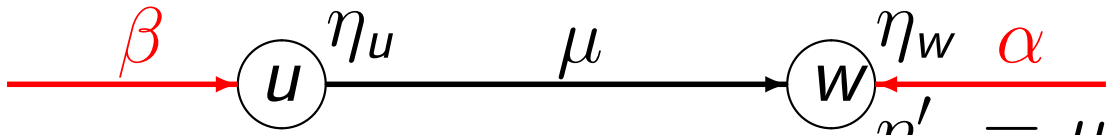
$$\bigvee_{\beta \in Path(u)} \theta(\beta) \preceq \eta_u \qquad (1)$$

$$\bigvee_{\alpha \in Path(w)} \theta(\alpha) \preceq \eta_w \qquad (2)$$

Индуктивная гипотеза для вершины w

Алгоритм проверки эквивалентности

2. Действительно, если правило переписывания разметок было применено к дуге $u \xrightarrow{\mu} w$, то



$$\eta'_w = \mu\eta_u \vee \eta_w$$

$$\bigvee_{\beta \in Path(u)} \theta(\beta) \preceq \eta_u \qquad (1)$$

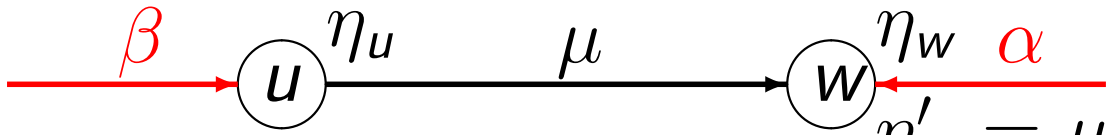
$$\bigvee_{\alpha \in Path(w)} \theta(\alpha) \preceq \eta_w \qquad (2)$$

$$\bigvee_{\alpha \in Path(w)} \theta(\alpha) \preceq \bigvee_{\beta \in Path(u)} \mu\theta(\beta) \qquad (3)$$

Если $\beta \in Path(u)$, то $\beta, (u, w) \in Path(w)$

Алгоритм проверки эквивалентности

2. Действительно, если правило переписывания разметок было применено к дуге $u \xrightarrow{\mu} w$, то



$$\bigvee_{\beta \in Path(u)} \theta(\beta) \preceq \eta_u \qquad (1)$$

$$\bigvee_{\alpha \in Path(w)} \theta(\alpha) \preceq \eta_w \qquad (2)$$

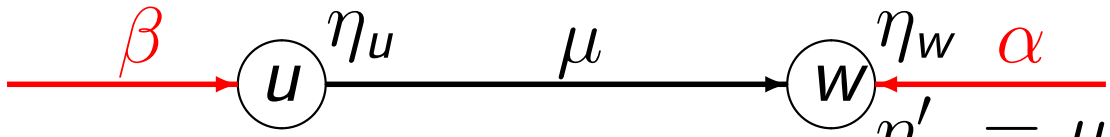
$$\bigvee_{\alpha \in Path(w)} \theta(\alpha) \preceq \bigvee_{\beta \in Path(u)} \mu\theta(\beta) \qquad (3)$$

$$\bigvee_{\beta \in Path(u)} \mu\theta(\beta) = \mu \bigvee_{\beta \in Path(u)} \theta(\beta) \qquad (4)$$

По закону дистрибутивности композиции относительно антиунификации

Алгоритм проверки эквивалентности

2. Действительно, если правило переписывания разметок было применено к дуге $u \xrightarrow{\mu} w$, то



$$\eta'_w = \mu\eta_u \vee \eta_w$$

$$\bigvee_{\beta \in Path(u)} \theta(\beta) \preceq \eta_u \qquad (1)$$

$$\bigvee_{\alpha \in Path(w)} \theta(\alpha) \preceq \eta_w \qquad (2)$$

$$\bigvee_{\alpha \in Path(w)} \theta(\alpha) \preceq \bigvee_{\beta \in Path(u)} \mu\theta(\beta) \qquad (3)$$

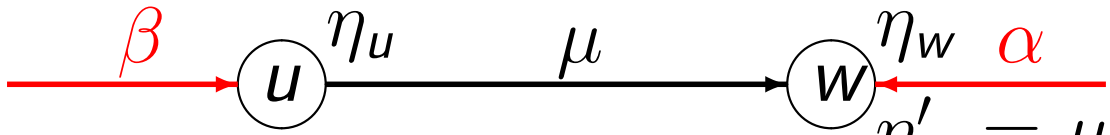
$$\bigvee_{\beta \in Path(u)} \mu\theta(\beta) = \mu \bigvee_{\beta \in Path(u)} \theta(\beta) \qquad (4)$$

$$\mu \bigvee_{\beta \in Path(u)} \theta(\beta) \preceq \mu\eta_u \qquad (5)$$

Из (1) по закону монотонности композиции

Алгоритм проверки эквивалентности

2. Действительно, если правило переписывания разметок было применено к дуге $u \xrightarrow{\mu} w$, то



$$\eta'_w = \mu\eta_u \vee \eta_w$$

$$\bigvee_{\beta \in Path(u)} \theta(\beta) \preceq \eta_u \qquad (1)$$

$$\bigvee_{\alpha \in Path(w)} \theta(\alpha) \preceq \eta_w \qquad (2)$$

$$\bigvee_{\alpha \in Path(w)} \theta(\alpha) \preceq \bigvee_{\beta \in Path(u)} \mu\theta(\beta) \qquad (3)$$

$$\bigvee_{\beta \in Path(u)} \mu\theta(\beta) = \mu \bigvee_{\beta \in Path(u)} \theta(\beta) \qquad (4)$$

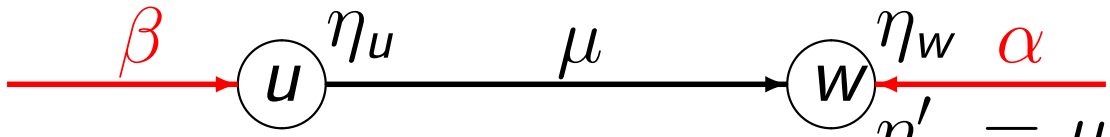
$$\mu \bigvee_{\beta \in Path(u)} \theta(\beta) \preceq \mu\eta_u \qquad (5)$$

$$\bigvee_{\alpha \in Path(w)} \theta(\alpha) \preceq \mu\eta_u \qquad (6)$$

Из (3), (4), (5) по транзитивности отношения $\preceq$

Алгоритм проверки эквивалентности

2. Действительно, если правило переписывания разметок было применено к дуге $u \xrightarrow{\mu} w$, то



$$\eta'_w = \mu\eta_u \vee \eta_w$$

$$\bigvee_{\beta \in Path(u)} \theta(\beta) \preceq \eta_u \qquad (1)$$

$$\bigvee_{\alpha \in Path(w)} \theta(\alpha) \preceq \eta_w \qquad (2)$$

$$\bigvee_{\alpha \in Path(w)} \theta(\alpha) \preceq \bigvee_{\beta \in Path(u)} \mu\theta(\beta) \qquad (3)$$

$$\bigvee_{\beta \in Path(u)} \mu\theta(\beta) = \mu \bigvee_{\beta \in Path(u)} \theta(\beta) \qquad (4)$$

$$\mu \bigvee_{\beta \in Path(u)} \theta(\beta) \preceq \mu\eta_u \qquad (5)$$

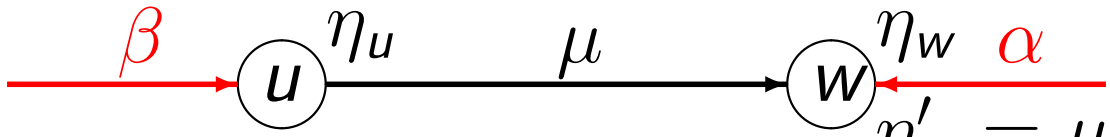
$$\bigvee_{\alpha \in Path(w)} \theta(\alpha) \preceq \mu\eta_u \qquad (6)$$

$$\bigvee_{\alpha \in Path(w)} \theta(\alpha) \preceq \mu\eta_u \vee \eta_w \qquad (7)$$

Из (2) и (6) по определению антиунификации

Алгоритм проверки эквивалентности

2. Действительно, если правило переписывания разметок было применено к дуге $u \xrightarrow{\mu} w$, то



$$\eta'_w = \mu\eta_u \vee \eta_w$$

$$\bigvee_{\beta \in Path(u)} \theta(\beta) \preceq \eta_u \qquad (1)$$

$$\bigvee_{\alpha \in Path(w)} \theta(\alpha) \preceq \eta_w \qquad (2)$$

$$\bigvee_{\alpha \in Path(w)} \theta(\alpha) \preceq \bigvee_{\beta \in Path(u)} \mu\theta(\beta) \qquad (3)$$

$$\bigvee_{\beta \in Path(u)} \mu\theta(\beta) = \mu \bigvee_{\beta \in Path(u)} \theta(\beta) \qquad (4)$$

$$\mu \bigvee_{\beta \in Path(u)} \theta(\beta) \preceq \mu\eta_u \qquad (5)$$

$$\bigvee_{\alpha \in Path(w)} \theta(\alpha) \preceq \mu\eta_u \qquad (6)$$

$$\bigvee_{\alpha \in Path(w)} \theta(\alpha) \preceq \mu\eta_u \vee \eta_w \qquad (7)$$

$$\bigvee_{\alpha \in Path(w)} \theta(\alpha) \preceq \eta'_w$$

Из (7) по определению правила переписывания разметок

Алгоритм проверки эквивалентности

Доказательство.

3. Воспользовавшись индукцией по длине маршрута и применяя свойство монотонности композиции подстановок (Утверждение 3), можно показать, что для каждой вершины $w = (v', v'')$ графа $\Gamma[\pi', \pi'']$ и для каждого маршрута α , ведущего в эту вершину, по окончании работы процедуры переписывания выполняется неравенство $\eta_w \preceq \theta(\alpha)$.

Проведите индуктивное обоснование самостоятельно, подобно тому, как это было сделано в п. 2 .

Отсюда следует, что для каждой вершины $w = (v', v'')$ вычисленная в конце работы алгоритма разметки подстановка η_w удовлетворяет неравенству

$$\eta_w \preceq \bigvee_{\alpha \in Path(v', v'')} \theta(\alpha).$$

Алгоритм проверки эквивалентности

Таким образом, для проверки л.-т. эквивалентности программ $\pi' \overset{lt}{\sim} \pi''$ достаточно

1. построить граф $\Gamma[\pi', \pi'']$ совместных вычислений схем программ π' и π'' ,
2. применить описанный алгоритм разметки вершин w графа подстановками η_w ,
3. проверить для каждой вершины $w = (v', v'')$ в графе $\Gamma[\pi', \pi'']$ выполнимость равенства $B'[v']\eta_w = B''[v'']\eta_w$, где η_w — это подстановка, вычисленная алгоритмом разметки для вершины w .

Теоремы 2-4 гарантируют завершаемость и корректность предложенного алгоритма проверки л.-т. эквивалентности стандартных схем программ.

Исследования показали, что проверка л.-т. эквивалентности стандартных схем программ осуществима за время $O(n^6)$.

Алгоритм проверки эквивалентности

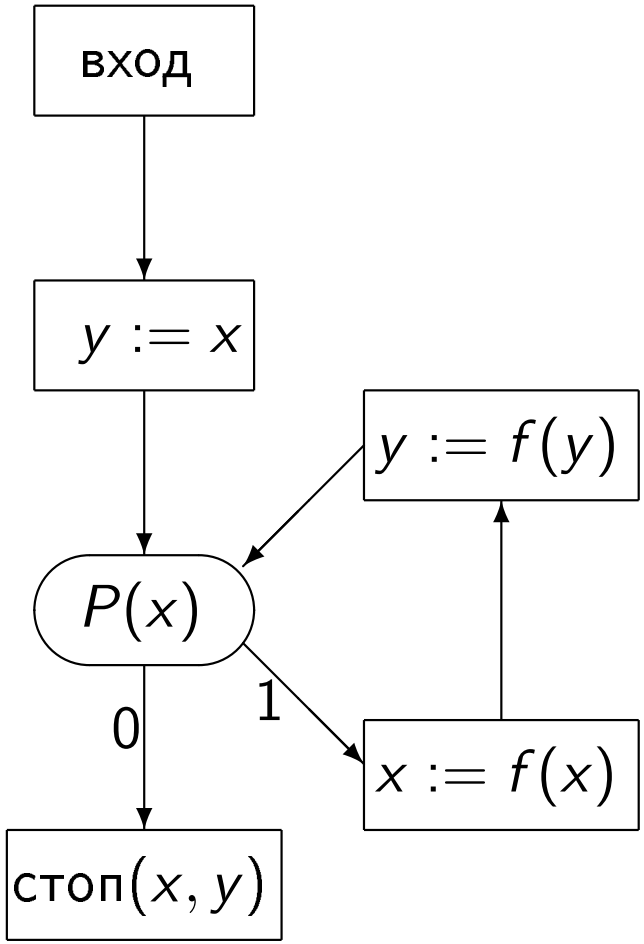


Схема π'

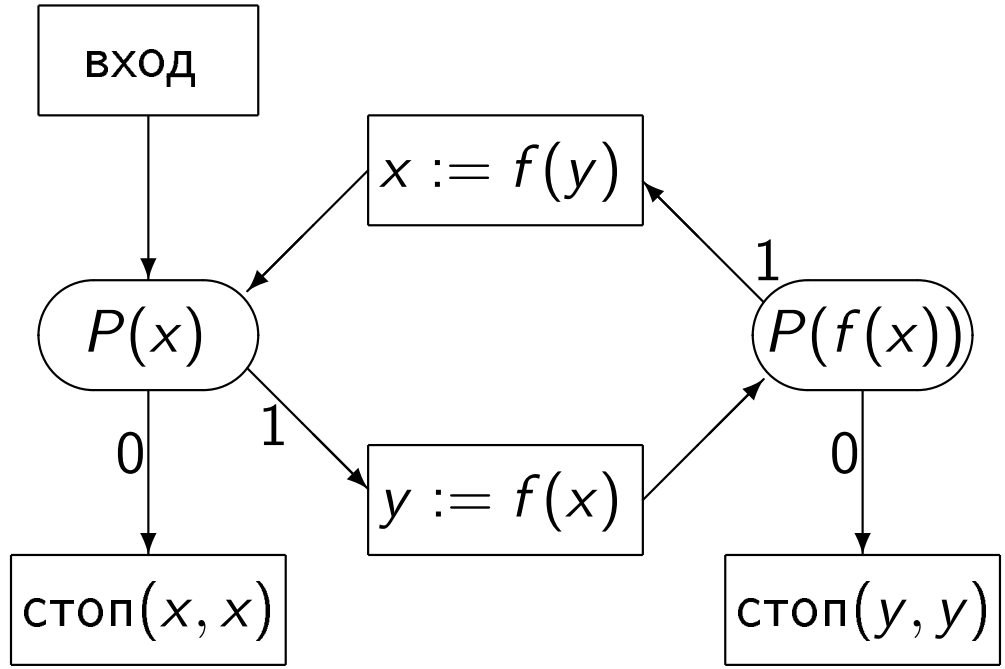
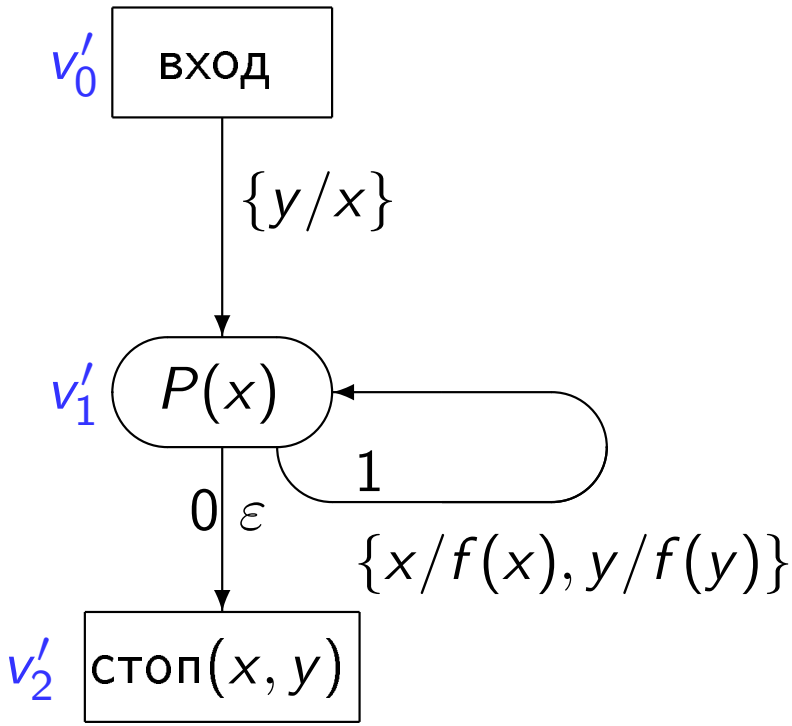
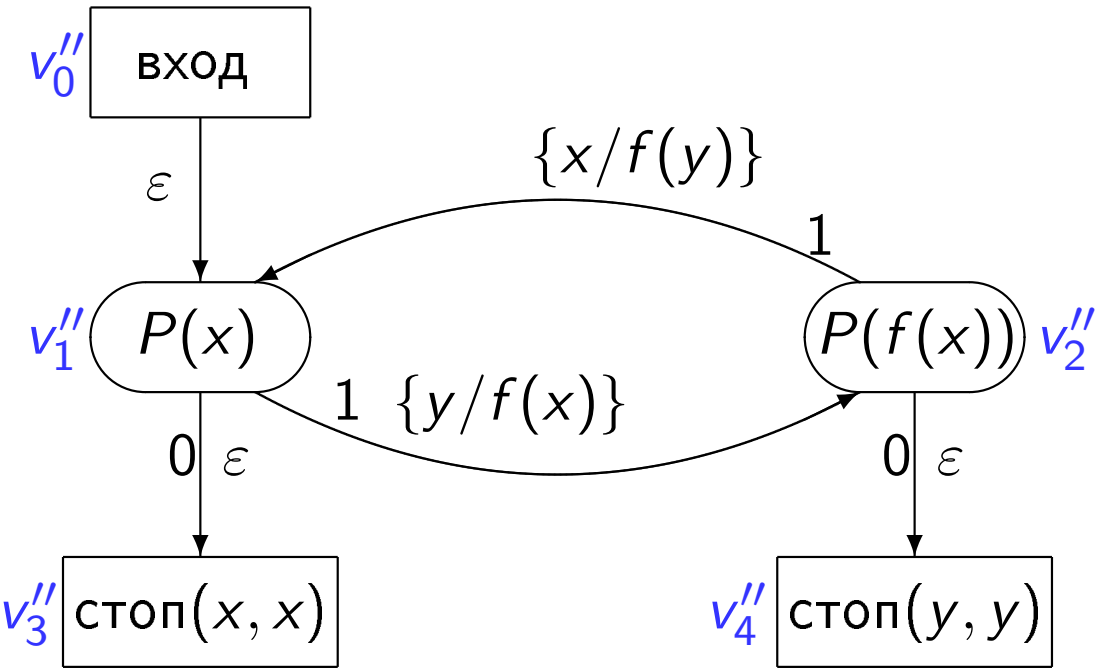


Схема π''

Алгоритм проверки эквивалентности

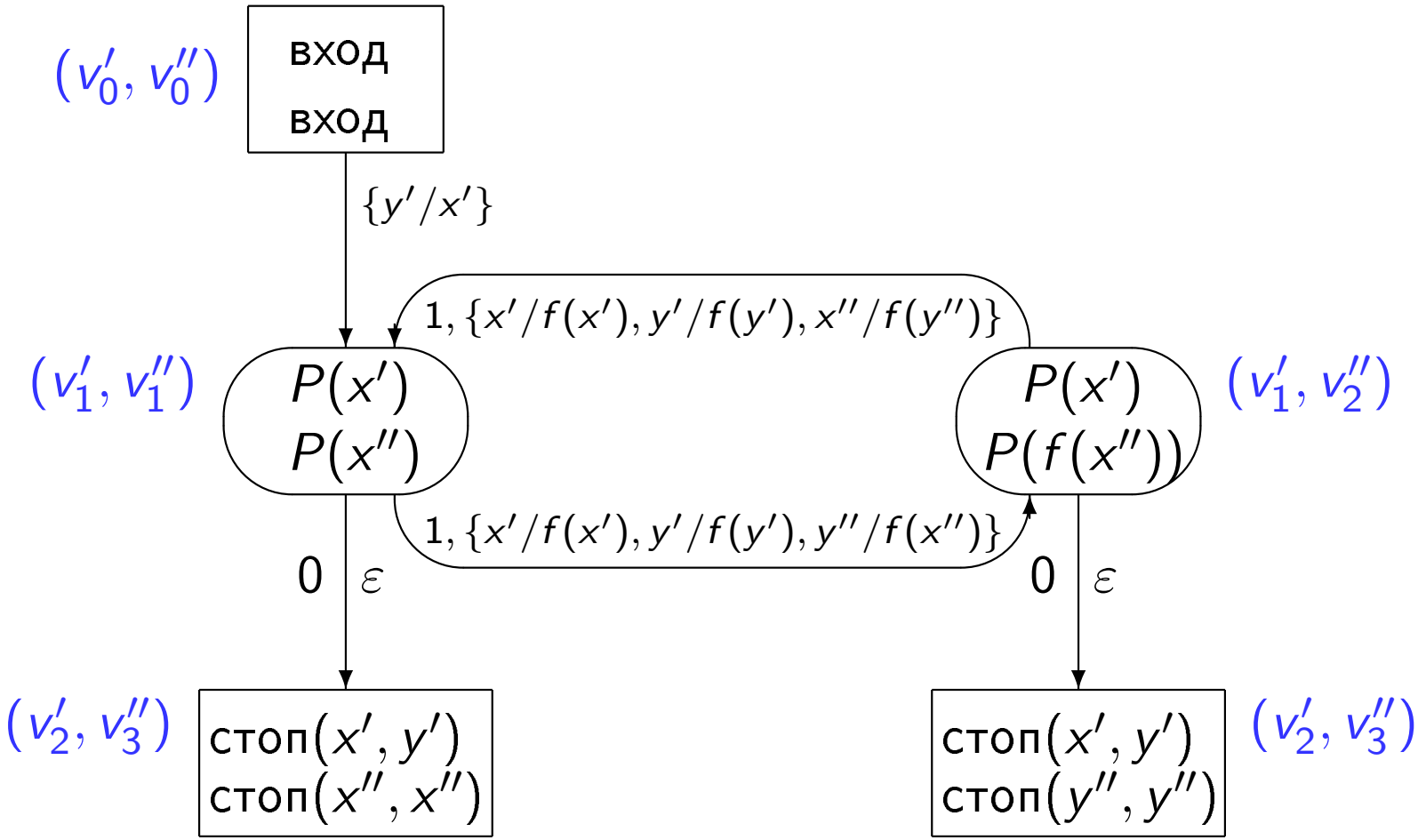


Система переходов π'



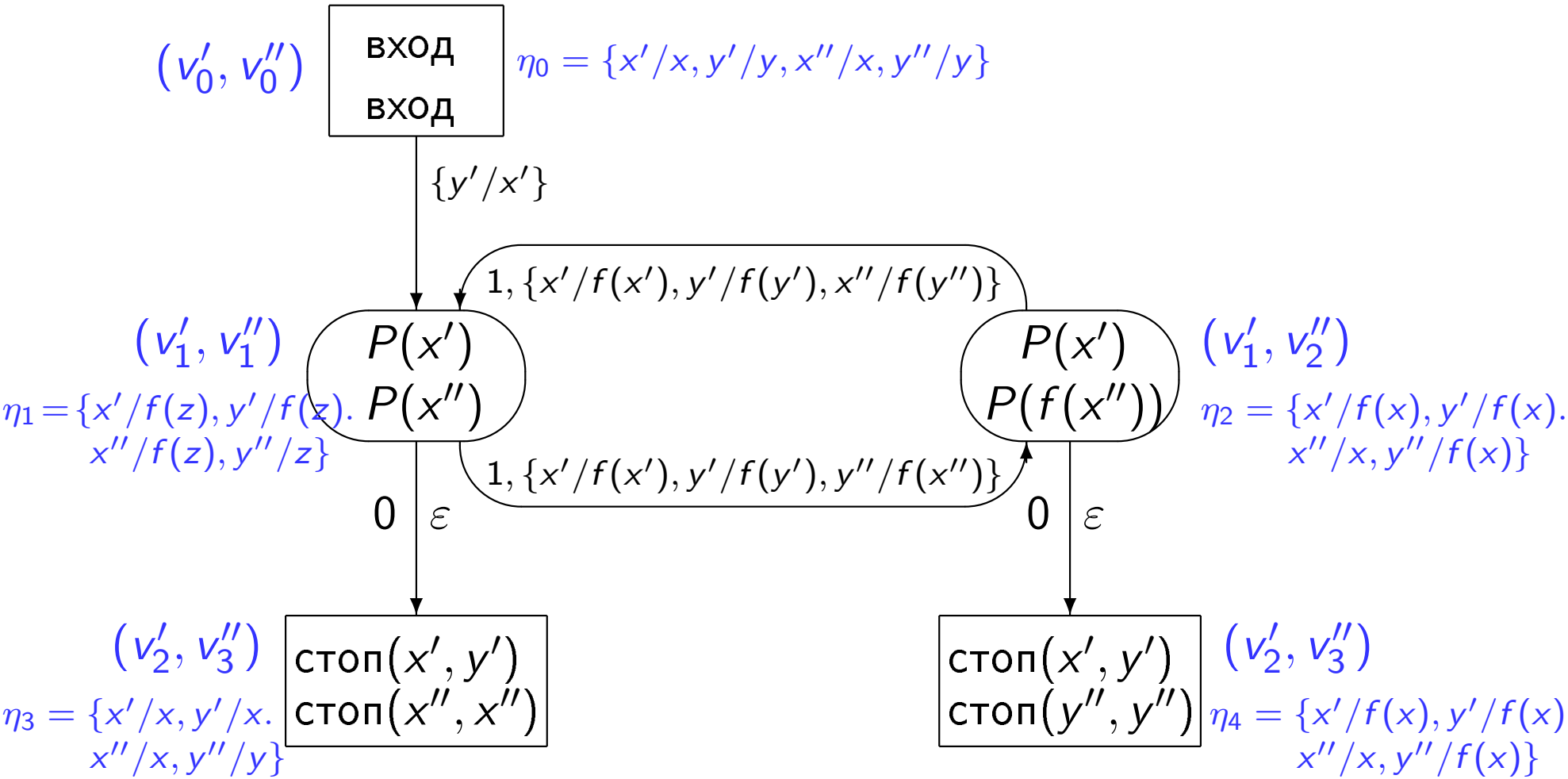
Система переходов π''

Алгоритм проверки эквивалентности



Граф совместных вычислений $\Gamma[\pi', \pi'']$

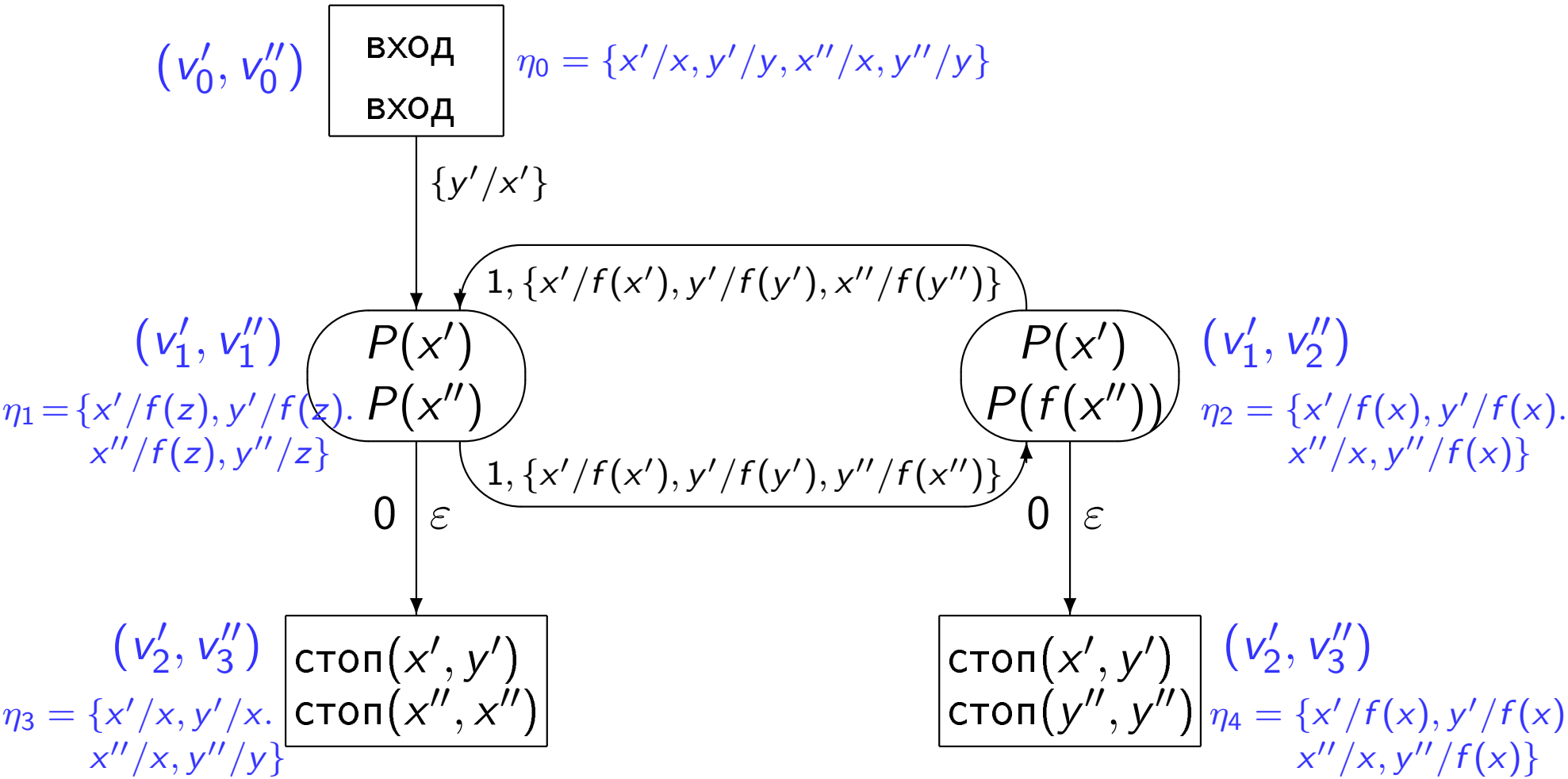
Алгоритм проверки эквивалентности



Окончательная разметка

Граф совместных вычислений $\Gamma[\pi', \pi'']$

Алгоритм проверки эквивалентности



Таким образом, $\pi' \overset{lt}{\sim} \pi''$

Граф совместных вычислений $\Gamma[\pi', \pi'']$