# Межфакультетский курс «Методы искусственного интеллекта в задачах анализа данных и верификации программ» Лекция 1 Верификация алгоритмов вычисления суммы, корня и возведения в степень

А.М.Миронов

Методы искусственного интеллекта в задачах верификации программ заключаются в выполнении интеллектуального анализа правильности программ. Результатом интеллектуального анализа является построение формального доказательства правильности анализируемой программы.

## 1 Проблема верификации программ

Проблема верификации (т.е. доказательства правильности) программ занимает центральное положение в теории и практике разработки программного обеспечения. Под правильностью программ понимается их соответствие различным условиям корректности, безопасности, устойчивости в случае непредусмотренного поведения окружения, эффективности использования ресурсов времени и памяти, оптимальности реализованных в программе алгоритмов, и т.п.

Как правило, для обоснования правильности программы её тестируют, т.е анализируют её поведение на некоторых входных данных. Однако тестирование обладает очевидным недостатком: если его возможно провести не для всех допустимых входных данных, а только лишь для их небольшой части (что имеет место почти всегда), то оно не может служить гарантированным обоснованием того, что тестируемая программа

обладает проверяемыми свойствами. Как отметил один из основоположников программирования Э. В. Дейкстра, «тестирование может лишь помочь выявить некоторые ошибки, но отнюдь не доказать их отсутствие».

Ошибки в программах могут быть весьма тонкими, но во многих программах наличие даже незначительных ошибок категорически недопустимо. Например, наличие ошибок в таких программах, как

- программы управления атомными электростанциями,
- программы, управляющие работой медицинских устройств,
- программы в бортовых системах управления самолетов и космических аппаратов,
- программы в системах управления секретными базами данных, системах электронной коммерции и т.п.

может привести к существенному ущербу для экономики и жизни людей.

Приведем один пример, иллюстрирующий наличие ошибок даже в очень простых программах, правильность которых на первый взгляд не вызывает никакого сомнения. Рассмотрим программу $P$, задача которой заключается в зачислении денег на счет клиента банка. Количество денег на счету этого клиента хранится в базе данных банка в переменной $x$. Когда $P$ выполняет действия по зачислению суммы $s$ на этот счет, она выполняет следующие действия:

- копирует в свою внутреннюю память значение переменной $x$,
- вычисляет новое значение, которое должна иметь переменная $x$, оно равно сумме текущего значения $x$ и зачисляемой суммы $s$, и
- заносит в переменную $x$ это новое значение.

Даже если программа $P$ выполняет все свои действия правильно, это не гарантирует корректности обслуживания клиента в том случае, когда состояние его счета может изменяться несколькими такими программами. Рассмотрим ситуацию, когда состояние счета клиента изменяют две программы $P_1$ и $P_2$ описанного выше типа. Возможен следующий вариант совместного выполнения этих программ:

- сначала программа $P_1$ выполняет свое первое действие, оно начинается в момент времени $t_1$, а заключительное действие $P_1$ (обновление значения $x$) происходит в момент времени $t_2$, и

- в момент времени, лежащий в интервале между $t_1$ и $t_2$, программа $P_2$ начинает свою операцию зачисления денег на счет клиента, причем выполнение первого действия программы $P_2$ (копирование значения переменной $x$) производится до выполнения заключительного действия программы $P_1$.

После завершения работы обоих программ те деньги, которые зачислила на счет клиента одна из программ $P_1$, $P_2$, просто пропадут.

Ошибку описанного выше типа можно не обнаружить путем тестирования, т.к. операция зачисления денег на счет клиента выполняется практически мгновенно, и поэтому среди тестов, которыми можно анализировать программы подобного типа, с большой вероятностью могут отсутствовать такие тесты, в которых две различные программы, обслуживающие счета клиентов, почти одновременно обращаются к одному и тому же ресурсу памяти.

Если же в результате какого-либо тестирования указанная выше ошибка обнаруживается и для её исправления конструируются специальные программные механизмы (семафоры и т.п.) с целью задания правильной дисциплины обращения программ к одному и тому же ресурсу памяти, то встает вопрос о том, насколько эти механизмы соответствуют своему предназначению (в частности, защищены ли семафоры от непредусмотренного и неавторизованного изменения их значений). Это тоже может анализироваться путем тестирования, и опять может получиться так, что среди тестов, которыми анализируется поведение указанных выше специальных программных механизмов, с большой вероятностью будут отсутствовать такие тесты, в которых проявляется некорректное поведение этих механизмов. Например, две программы $P_1$ и $P_2$, работающие с совместно используемым ресурсом $R$, могут использовать в качестве семафора общую переменную $b$, принимающую 2 значения: 1 (доступ к $R$ открыт) и 0 (доступ к $R$ закрыт). Если какая-либо из этих программ, например $P_1$, хочет работать с $R$, она проверяет значение $b$:

- если $b = 1$, то $P_1$ изменяет значение $b$ на 0 (запрещая тем самым доступ к $R$ программе $P_2$ на время своей работы с $R$) и работает с $R$, после чего изменяет значение $b$ обратно на 1,

- а если $b = 0$, то $P_1$ ждет, пока значение $b$ не станет равным 1.

Однако если программа $P_2$ тоже хочет работать с $R$, и она проверила значение $b$ после того момента когда его проверила $P_1$, но до того момента как $P_1$ изменила значение $b$ с 1 на 0, то может возникнуть нарушение дисциплины доступа к $R$.

Гарантированное обоснование правильности программ может быть получено только при помощи альтернативного подхода, принципиально отличного от тестирования. Данный подход называется **верификацией**. В самом общем виде верификация программы может пониматься как построение математического доказательства утверждения о том, что верифицируемая программа соответствует своему предназначению. Предназначение программы может быть выражено, например, путем описания функции, которую должна вычислять эта программа, или правил взаимодействия этой программы с другими программами, т.е. реакции, которую эта программа должна обеспечивать в ответ на получение сигналов или сообщений от других программ.

Формальное описание предназначения программы (или некоторых свойств, которыми она должна обладать) в виде математического утверждения называется **спецификацией** этой программы. Спецификация может представлять собой формальное описание самых разнообразных свойств программы, например:

- её входные и выходные данные находятся в заданном соотношении,
- программа всегда завершает свою работу,
- во время работы программы не происходит сбоев и ненормальных ситуаций (например, деления на 0, извлечения квадратного корня из отрицательного числа, выхода индекса за границы массива, неавторизованных утечек информации и т.п.),
- программа решает свою задачу за установленное время и использует не более установленного объема памяти.

Для верификации программы $P$ необходимо определить

- математический смысл всех конструкций, используемых в $P$, называемый **формальной семантикой** (или просто **семантикой**) этих конструкций, и
- спецификацию $Spec$ этой программы, выражающую то свойство программы $P$, которое необходимо верифицировать,

после чего можно ставить вопрос о верификации $P$ относительно $Spec$, т.е. о построении математического доказательства утверждения о том, что $P$ удовлетворяет $Spec$.

# 2 Необходимые математические понятия

## 2.1 Термы и связанные с ними понятия

Мы будем предполагать, что заданы следующие множества.

- Множество $Types$, элементы которого называются **типами**. Мы будем понимать типы так же, как понимаются типы данных в языках программирования. Каждому типу $\tau \in Types$ сопоставлено множество $D_\tau$ **значений** типа $\tau$, называемое **доменом** типа $\tau$.

  Символ $\mathcal{D}$ обозначает множество всех значений всех типов.

- Множество $Var$, элементы которого называются **переменными**. Каждой переменной $x \in Var$ сопоставлен тип $\tau(x) \in Types$. Каждая переменная $x \in Var$ может принимать **значения** в домене $D_{\tau(x)}$, т.е. в различные моменты времени переменная $x$ может быть связана с различными элементами домена $D_{\tau(x)}$.

- Множество $Con$, элементы которого называются **константами**.

  Каждой константе $c \in Con$ сопоставлены тип $\tau(c) \in Types$ и значение из $D_{\tau(c)}$, обозначаемое тем же символом $c$ и называемое **интерпретацией** константы $c$. Будем считать, что $\forall\, \tau \in Types$ любой элемент $d \in D_\tau$ является константой типа $\tau$, которой соответствует сам элемент $d$.

- Множество $Fun$, элементы которого называются **функциональными символами (ФС)**. Каждому ФС $f \in Fun$ сопоставлены

  - **функциональный тип (ФТ)** $\tau(f)$, который представляет собой запись вида
    $$(\tau_1, \ldots, \tau_n) \to \tau, \tag{1}$$
    где $\tau_1, \ldots, \tau_n, \tau \in Types$, и
  - частичная функция вида $D_{\tau_1} \times \ldots \times D_{\tau_n} \to D_\tau$, где $\tau(f)$ имеет вид (1), данная функция обозначается тем же символом $f$ и называется **интерпретацией** ФС $f$.

    (Напомним, что функция $f : A \to B$ называется **частичной**, если $\forall\, a \in A$ значение $f(a)$ м.б. не определено.)

**Функциональной переменной** называется переменная, тип которой является функциональным типом. Множество всех функциональных переменных обозначается записью $FVar$.

**Термы** строятся из переменных, констант и ФС. Множество всех термов обозначается символом $Tm$. Каждый терм $e$ имеет тип $\tau(e) \in Types$, определяемый структурой терма $e$.

Правила построения термов имеют следующий вид:

- каждая переменная и константа является термом того типа, который сопоставлен этой переменной или константе, и
- если $e_1, \ldots, e_n \in Tm$, $f \in Fun \cup FVar$, и $\tau(f)$ имеет вид (1), где $\tau_1 = \tau(e_1)$, ..., $\tau_n = \tau(e_n)$, то запись $f(e_1, \ldots, e_n)$ – терм типа $\tau$.

Терм $e \in Tm$ называется **подтермом** терма $e' \in Tm$, если либо $e = e'$, либо $e' = f(e_1, \ldots, e_n)$, где $f \in Fun \cup FVar$, и $\exists\, i \in \{1, \ldots, n\}$: $e$ – подтерм терма $e_i$. Запись $e \subseteq e'$, где $e, e' \in Tm$, означает, что $e$ – подтерм $e'$. Запись $e \subset e'$, где $e, e' \in Tm$, означает, что $e \subseteq e'$ и $e \neq e'$.

Индукцией по структуре терма $e \in Tm$ нетрудно доказать, что

$$\text{если } e_1 \text{ и } e_2 \text{ – различные подтермы терма } e, \text{ то либо } e_1 \subset e_2, \text{ либо } e_2 \subset e_1, \text{ либо } e_1 \text{ и } e_2 \text{ не имеют общих компонентов.} \tag{2}$$

Запись $x \in e$, где $x \in Var, e \in Tm$, означает, что $x$ входит в $e$.

Будем использовать следующие обозначения и соглашения:

- $\forall\, e \in Tm\ Var(e)$ обозначает множество $\{x \in Var \mid x \in e\}$,
- $\forall\, e_1, \ldots, e_n \in Tm\ Var(e_1, \ldots, e_n) = Var(e_1) \cup \ldots \cup Var(e_n)$,
- $\forall\, X \subseteq Var\ Tm(X)$ обозначает множество $\{e \in Tm \mid Var(e) \subseteq X\}$,
- $FVar(e)$ обозначает множество $Var(e) \cap FVar$,
- $\forall\, \tau \in Types$ запись $Tm_\tau$ обозначает множество $\{e \in Tm \mid \tau(e) = \tau\}$, $\forall\, E \subseteq Tm$ запись $E_\tau$ обозначает множество $E \cap Tm_\tau$,
- в терме вида $f(e)$ скобки слева и справа от $e$ могут опускаться, если $\tau(f)$ имеет вид $(\tau) \to \tau'$,
- для каждой рассматриваемой функции $f : E \to E'$, где $E, E' \subseteq Tm$, будем предполагать, что $\forall\, e \in E\ \ \tau(e) = \tau(f(e))$.

Терм $e \in Tm$ **замкнут**, если $Var(e) = \emptyset$. Каждому замкнутому терму $e$ соответствует объект $eval(e)$, называемый **значением** данного терма, и либо является элементом $D_{\tau(e)}$, либо не определён. Если $e$ – константа, то $eval(e)$ – интерпретация этой константы, и если $e$ имеет вид $f(e_1, \ldots, e_n)$, то $eval(e)$ определён, только если определено значение функции $f$ на кортеже $(eval(e_1), \ldots, eval(e_n))$, и в этом случае $eval(e)$ равен этому значению. Для каждого замкнутого терма $e$ будем обозначать объект $eval(e)$ той же записью, что и сам терм (т.е. $e$).

## 2.2 Примеры типов и функциональных символов

### 2.2.1 Арифметические типы и функциональные символы

$Con$ содержит типы **N** и **I**, значениями которых являются натуральные $(0, 1, \ldots)$ и целые числа соответственно.

$Fun$ содержит ФС $+, -, \cdot, div, mod$ типа $(\mathbf{I}, \mathbf{I}) \to \mathbf{I}$, и

- функции $+$, $-$, и $\cdot$ представляют собой соответствующие арифметические операции,
- $div$ – частичная функция (определённая, только когда второй аргумент отличен от 0), $mod$ – частичная функция (определённая, только когда второй аргумент больше 0), $div$ и $mod$ вычисляют частное и остаток соответственно от деления первого аргумента на второй.

Термы $+(e_1, e_2)$, $-(e_1, e_2)$, $\cdot(e_1, e_2)$, $div(e_1, e_2)$, $mod(e_1, e_2)$ будут записываться в виде $e_1 + e_2$, $e_1 - e_2$, $e_1 e_2$, $e_1/e_2$ и $e_1 \% e_2$ соответственно.

### 2.2.2 Логические типы и функциональные символы

$Types$ содержит тип **B**, и $D_{\mathbf{B}} = \{0, 1\}$. Термы типа **B** называются **формулами**. Множество всех формул обозначается $Fm$. $\forall X \subseteq Var$ запись $Fm(X)$ обозначает множество $Tm(X) \cap Fm$. При построении формул могут использоваться обычные булевы ФС ($\neg$, $\wedge$, $\vee$, $\to$ и т.д.), которым соответствуют функции отрицания, конъюнкции, дизъюнкции и т.д. Символ 1 обозначает тождественно истинную формулу, а символ 0 – тождественно ложную формулу. Формулы вида $\wedge(e_1, e_2)$, $\vee(e_1, e_2)$ и т.п. мы будем записывать в более привычном виде $e_1 \wedge e_2$, $e_1 \vee e_2$ и т.д. Формулы вида $e_1 \wedge \ldots \wedge e_n$ и $e_1 \vee \ldots \vee e_n$ могут также записываться в виде $\left\{ \begin{smallmatrix} e_1 \\ \ldots \\ e_n \end{smallmatrix} \right\}$ и $\left[ \begin{smallmatrix} e_1 \\ \ldots \\ e_n \end{smallmatrix} \right]$ соответственно, а также в виде $\bigwedge_{i \in \{1,\ldots,n\}} e_i$ и $\bigvee_{i \in \{1,\ldots,n\}} e_i$ соответственно, и также в виде $\{e_1, \ldots, e_n\}$ и $[e_1, \ldots, e_n]$ соответственно. Формулы вида $\neg e$ могут обозначаться $\bar{e}$.

Разные ФС могут иметь одинаковое обозначение. Ниже мы приводим примеры таких ФС. Для каждого типа $\tau$

- $Fun$ содержит ФС $eq$ типа $(\tau, \tau) \to \mathbf{B}$, которому соответствует функция, отображающая пару $(d_1, d_2) \in D_\tau \times D_\tau$ в элемент 1, если $d_1 = d_2$, и 0, если $d_1 \neq d_2$;
- если на $D_\tau$ задано отношение частичного порядка, то $Fun$ содержит ФС $<, \leq, >, \geq$ типа $(\tau, \tau) \to \mathbf{B}$, каждому из этих ФС соответствует функция, отображающая каждую пару $(d_1, d_2) \in D_\tau \times D_\tau$ в элемент

– 1, если $d_1 < d_2$, $d_1 \leq d_2$, $d_1 > d_2$, $d_1 \geq d_2$ соответственно, и

– 0, если соответствующее соотношение неверно;

- *Fun* содержит ФС $if\_then\_else$ типа $(\mathbf{B}, \tau, \tau) \to \tau$, соответствующая функция отображает тройку вида $(1, d_1, d_2)$ в $d_1$ и тройку вида $(0, d_1, d_2)$ в $d_2$.

Термы $eq(e_1, e_2)$, $<(e_1, e_2)$ и т.д. будут записываться в виде $e_1 = e_2$, $e_1 < e_2$ и т.д. соответственно. Термы $if\_then\_else(e, e_1, e_2)$ будут записываться в виде **if** $e$ **then** $e_1$ **else** $e_2$, или $e?e_1 : e_2$.

### 2.2.3 Строковые типы и функциональные символы

*Types* содержит типы **L** и **S**, значения которых называются **символами** и **символьными строками** (или просто **строками**) соответственно. Каждая строка представляет собой последовательность символов $a_1 \dots a_n$, где $n \geq 0$. При $n = 0$ эта последовательность пустая и обозначается $\varepsilon$.

*Fun* содержит

- ФС *head* и *tail* типа $\mathbf{S} \to \mathbf{L}$ и $\mathbf{S} \to \mathbf{S}$ соответственно, которым соответствуют частичные функции, определенные только для непустых строк, данные функции отображают строку $a_1 \dots a_n$ в символ $a_1$ и строку $a_2 \dots a_n$ соответственно (называемые **головой** и **хвостом** строки $a_1 \dots a_n$ соответственно);

- ФС *conc* типа $(\mathbf{L}, \mathbf{S}) \to \mathbf{S}$, которому соответствует функция, отображающая пару $(a, a_1 \dots a_n)$ в строку $aa_1 \dots a_n$.

Термы $conc(e, e')$, $head(e)$, $tail(e)$ будем записывать в сокращенном виде $ee', e_h$ и $e_t$ соответственно.

### 2.2.4 Кортежные типы и функциональные символы

Для каждого списка типов $\tau_1, \dots, \tau_n$ (некоторые компоненты этого списка могут совпадать)

- *Types* содержит тип, обозначаемый записью $(\tau_1, \dots, \tau_n)$, и

$$D_{(\tau_1,\dots,\tau_n)} = D_{\tau_1} \times \dots \times D_{\tau_n},$$

- *Fun* содержит ФС *tuple* типа $(\tau_1, \dots, \tau_n) \to (\tau_1, \dots, \tau_n)$, ему соответствует тождественная функция, термы вида $tuple(e_1, \dots, e_n)$ будут обозначаться записью $(e_1, \dots, e_n)$.

## 2.3 Подстановки

**Подстановкой** называется функция $\theta : Var \to Tm$. Будем говорить, что подстановка $\theta$ заменяет переменную $x \in Var$ на терм $\theta(x)$.

Будем использовать следующие обозначения:

- множество всех подстановок обозначается символом $\Theta$;
- $\forall\, \theta \in \Theta$ запись $Var(\theta)$ обозначает множество $\{x \in Var \mid \theta(x) \neq x\}$;
- $\forall\, X \subseteq Var \quad \Theta(X) = \{\theta \in \Theta \mid Var(\theta) \subseteq X\}$;
- подстановка $\theta \in \Theta$ может обозначаться записями

$$x \mapsto \theta(x) \quad \text{или} \quad (\theta(x_1)/x_1, \ldots, \theta(x_n)/x_n), \tag{3}$$

  вторая запись в (3) используется, когда $Var(\theta) = \{x_1, \ldots, x_n\}$;
- $\forall\, \theta \in \Theta,\ \forall\, e \in Tm$ запись $e^\theta$ обозначает терм, получаемый из $e$ заменой $\forall\, x \in Var(e)$ каждого вхождения $x$ в $e$ на терм $\theta(x)$;
- $\forall\, \theta, \theta' \in \Theta$ запись $\theta\theta'$ обозначает подстановку $x \mapsto (x^\theta)^{\theta'}$.

Подстановка $\theta$ **замкнута**, если $\forall\, x \in Var(\theta)\ Var(x^\theta) = \emptyset$. Множество всех замкнутых подстановок из $\Theta(X)$ обозначается $X^\bullet$.

Пусть заданы терм $e \in Tm$ и список $\vec{x} = (x_1, \ldots, x_n)$ различных переменных, причём $Var(e) \subseteq \{x_1, \ldots, x_n\}$. Будем использовать обозначения:

- $D_{\tau(\vec{x})}$ обозначает множество $D_{\tau(x_1)} \times \ldots \times D_{\tau(x_n)}$;
- $e(\vec{x})$ обозначает функцию вида $D_{\tau(\vec{x})} \to D_{\tau(e)}$, такую, что

$$\forall\, \vec{d} = (d_1, \ldots, d_n) \in D_{\tau(\vec{x})} \quad e(\vec{x}) : \vec{d} \mapsto e^{(d_1/x_1, \ldots, d_n/x_n)}. \tag{4}$$

## 2.4 Массивы

В программах могут использоваться структуры данных, называемые **массивами**. Массивы обозначаются записями вида $a_{m..n}$, где $a$ – имя массива, $m$ и $n$ – термы типа **I**, обозначающие нижнюю и верхнюю границы массива соответственно. Массив $a_{m..n}$ может обозначаться более коротко путем указания лишь его имени, без указания нижней и верхней границ. Если значения $m, n$ нижней и верхней границ массива $a$ таковы, что $m \leq n$, то массив $a$ непуст, в противном случае он является пустым. Для каждого массива $a_{m..n}$ и каждого $i \in \{m, \ldots, n\}$ определен объект, обозначаемый записью $a_i$ и называемый **компонентой** массива

$a$ с индексом $i$. Компоненты массива $a$ рассматриваются как переменные одинакового типа. Если $a$ и $b$ – массивы вида $a_{m..n}$ и $b_{m..n}$, то $b = perm(a)$ означает, что $b$ – перестановка $a$, т.е. существует биекция $f$ на $\{m, \ldots, n\}$, такая, что $\forall\, i \in \{m, \ldots, n\}\ \ b_i = a_{f(i)}$.

Пусть задан массив $a_{m..n}$ и $i, j, i', j' \in \{m, \ldots, n\}$. Будем обозначать записями $ord(a_{i..j})$, $a_{i..j} \le a_{i'..j'}$, $a_{i..j} \le a_{i'}$ формулы соответственно:

$$\bigwedge\nolimits_{i \le k < j}(a_k \le a_{k+1}), \quad \bigwedge_{\substack{i \le k \le j \\ i' \le k' \le j'}} (a_k \le a_{k'}), \quad \bigwedge\nolimits_{i \le k \le j}(a_k \le a_{i'}).$$

(Напомним, что если множество конъюнктивных членов пусто, то их конъюнкция равна 1.)

### 2.5 Истинностные значения утверждений

Будем использовать следующее обозначение: если $A$ – произвольное утверждение (выраженное на естественном или формальном языке), то запись $[\![A]\!]$ обозначает значение 1 если $A$ истинно и 0 если $A$ ложно.

## 3 Программы, представленные в виде блок-схем

В этой части рассматривается задача верификации последовательных и распределенных программ, представленных в виде блок-схем, в операторной форме и в виде процессов. В качестве метода их верификации используется один из наиболее широко распространённых методов верификации программ, известный под названием **метод инвариантов**.

Одним из языков описания последовательных нерекурсивных программ является язык блок-схем. На данном языке программа представляется в графовой форме. Графовая форма представления программ в последнее время завоевывает все бóльшую популярность по причине того, что такая форма представления программ облегчает их понимание и упрощает их анализ.

**Блок-схема (БС)** представляет собой конечный ориентированный граф, каждая вершина которого имеет один из следующих видов:

- **начальная** вершина, она обозначается символом $\odot$, в каждой БС имеется только одна начальная вершина, из неё выходит одно ребро, и в неё не входит ни одного ребра;

- **присваивание**, вершина такого вида обозначается записью вида

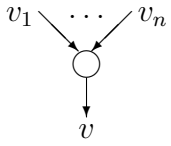
$$\boxed{x := e}\ ,$$

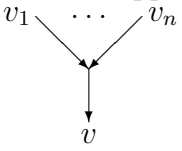
где $x \in Var$, $e \in Tm$, $\tau(x) = \tau(e)$, из каждого присваивания выходит только одно ребро;

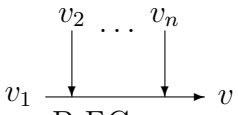
- **условный переход**, вершина такого вида обозначается записью вида $\textcircled{\beta}$, где $\beta \in Fm$, из каждого условного перехода выходит два ребра с метками 1 и 0 соответственно;

- **пустой оператор**, такая вершина обозначается символом $\bigcirc$, из неё выходит только одно ребро;

- **терминальная** вершина, она обозначается символом $\bigotimes$, из каждой терминальной вершины не выходит ни одного ребра.

Пустой оператор в БС, как правило, не изображают, т.е. фрагмент БС вида 

может изображаться в виде 

, или 

, или другим аналогичным образом.

В БС могут использоваться следующие обозначения:

- последовательность вершин вида $\boxed{x_1 := e_1} \to \cdots \to \boxed{x_n := e_n}$, где в каждую вершину (кроме, возможно, первой) входит только одно ребро, может обозначаться прямоугольником $\boxed{\begin{array}{c} x_1 := e_1 \\ \cdots \\ x_n := e_n \end{array}}$;

- запись

$$\boxed{(x_1, \ldots, x_n) := (e_1, \ldots, e_n)} \tag{5}$$

обозначает последовательность присваиваний вида

$$\boxed{y_1 := e_1} \to \cdots \to \boxed{y_n := e_n} \to \boxed{x_1 := y_1} \to \cdots \to \boxed{x_n := y_n}\ , \tag{6}$$

где переменные $y_1, \ldots, y_n$ – новые, т.е. если какая-либо БС содержит фрагмент вида (5), то он рассматривается как последовательность (6), где переменные $y_1, \ldots, y_n$ присутствуют только в присваиваниях из (6) и не присутствуют в этой БС вне (6);

- запись вида $\boxed{x \rightleftharpoons y}$, где $x, y \in Var$, имеет тот же смысл, что и запись $\boxed{(x, y) := (y, x)}$.

С каждой БС $P$ связана некоторая формула $Pre(P)$, называемая **предусловием** БС $P$.

Если $P$ – БС, то $V(P)$ обозначает совокупность всех вершин $P$ и $Var(P)$ обозначает множество всех переменных, входящих в $P$.

# 4 Выполнение блок-схемы

**Выполнение** БС $P$ – это обход её вершин, начиная с начальной вершины (т.е. последовательность переходов по рёбрам от одной вершины БС $P$ к другой), с выполнением действий, сопоставленных проходимым вершинам. После выполнения действия, соответствующего текущей вершине, происходит переход по выходящему из неё ребру к следующей вершине. На каждом шаге $i \geq 0$ выполнения БС $P$ определена подстановка $\theta_i \in Var(P)^\bullet$. Подстановка $\theta_0$ должна удовлетворять предусловию. Действия в вершинах выполняются в зависимости от вида вершин:

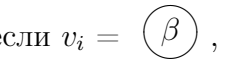
- $\odot$ или $\bigcirc$: никаких действий не происходит;
- $\boxed{x := e}$ : $x$ становится связанной со значением терма $e$;
- $\textcircled{\beta}$ : для перехода к следующей вершине выбирается ребро с меткой, равной значению формулы $\beta$;
- $\otimes$: выполнение БС завершается.

Формальное описание выполнения БС $P$ имеет следующий вид: это последовательность шагов с номерами $0, 1, \ldots$, с каждым шагом $i \geq 0$ выполнения $P$ связаны вершина $v_i \in V(P)$ и подстановка $\theta_i \in Var(P)^\bullet$ (называемые **текущей вершиной** и **текущей подстановкой** на шаге $i$), причем $v_0 = \odot$, $Pre(P)^{\theta_0} = 1$ и для каждого $i \geq 0$ выполнены следующие условия:

- если $v_i$ – начальная вершина или пустой оператор, то $v_{i+1}$ – конец ребра, выходящего из $v_i$; $\theta_{i+1} = \theta_i$,
- если $v_i = \boxed{x := e}$, то $v_{i+1}$ – конец ребра, выходящего из $v_i$, и

$$\theta_{i+1}(x) \stackrel{\text{def}}{=} e^{\theta_i}, \quad \forall\, y \in Var(P) \setminus \{x\} \quad \theta_{i+1}(y) \stackrel{\text{def}}{=} \theta_i(y)$$

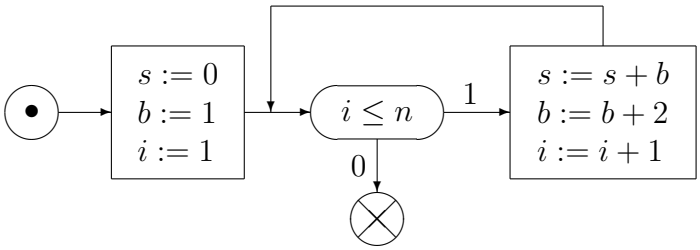
(данное действие заключается в обновлении значения переменной $x$: после исполнения этого действия значение $x$ становится равным значению терма $e$ на текущих значениях переменных БС $P$, значения остальных переменных не изменяются);

- если $v_i =$  , то
  - если $\beta^{\theta_i} = 1$ или 0, то $v_{i+1}$ – конец ребра, выходящего из $v_i$, и имеющего метку 1 или 0 соответственно, и $\theta_{i+1} = \theta_i$;
  - если $\beta^{\theta_i}$ не определено, то выполнение БС завершается;
- если $v_i = \bigotimes$, то выполнение БС завершается.

Выполнение действий, соответствующих фрагменту  , можно рассматривать как перемену местами содержимого переменных $x$ и $y$.

# 5 Примеры блок-схемы

БС, вычисляющая сумму $1 + 3 + 5 \ldots + (2n - 1)$, где $n \geq 1$ – входное значение, результат должен быть занесён в переменную $s$.




(7)

Все переменные в данной БС имеют тип **I**.

# 6 Задача верификации блок-схем

Пусть задана БС $P$, и её спецификация представляет собой пару формул $(Pre, Post)$ с переменными из $Var(P)$, имеющих следующий смысл:

- формула $Pre$ называется **предусловием** и выражает условие, которому должны удовлетворять значения переменных БС $P$ в момент начала её выполнения;

- формула $Post$ называется **постусловием**, и выражает условие, которому должны удовлетворять значения переменных БС $P$ после завершения её выполнения.

Пусть $P$ – БС и $\theta \in Var(P)^\bullet$. Мы будем говорить, что $P$ **выполняется с начальной подстановкой** $\theta$, если в момент начала выполнения $P$ значение каждой переменной $x \in Var(P)$ было равно $\theta(x)$.

Запись $\theta \xrightarrow{P} \otimes$ обозначает утверждение: если $P$ выполняется с начальной подстановкой $\theta$, то выполнение $P$ когда-либо завершится.

Если верно $\theta \xrightarrow{P} \otimes$, то запись $P\theta$ обозначает подстановку из $Var(P)^\bullet$, определяемую следующим образом: пусть $P$ выполняется с начальной подстановкой $\theta$, тогда $\forall\, x \in Var(P)$ значение $x^{P\theta}$ равно тому значению, которое будет иметь переменная $x$ после завершения выполнения $P$.

Задача **верификации** БС $P$ относительно спецификации $(Pre, Post)$ заключается в доказательстве следующего утверждения:

$$\forall\, \theta \in Var(P)^\bullet, \quad \text{если } \ Pre^\theta = 1, \ \text{ то } \ \theta \xrightarrow{P} \otimes \ \text{ и } \ Post^{P\theta} = 1. \qquad (8)$$

Данное утверждение обозначается записью $Pre \xrightarrow{P} Post$.

# 7 Метод инвариантов для верификации блок-схем

В этом параграфе излагается метод доказательства утверждений вида $Pre \xrightarrow{P} Post$, называемый **методом инвариантов**. Для его формулировки введём понятия базового множества и базового пути.

# 8 Базовые множества и базовые пути

Пусть задана БС $P$. **Путь** в $P$ – это последовательность $\pi = \alpha_1 \dots \alpha_k$ рёбер $P$, такая, что $\forall\, i = 1, \dots, k-1$ конец ребра $\alpha_i$ совпадает с началом ребра $\alpha_{i+1}$. Путь $\pi = \alpha_1 \dots \alpha_k$ называется **циклом** в $P$, если $\alpha_1 = \alpha_k$.

Множество $M$ точек на некоторых ребрах БС $P$ называется **базовым** для $P$, если каждый цикл в $P$ содержит ребро, на котором имеется точка из $M$. Если $M$ – базовое множество для $P$, то путь $\pi$ в $P$ называется **базовым** относительно $M$, если он непуст, на первом и последнем ребрах этого пути имеются точки из $M$, и на других рёбрах этого пути точек из $M$ нет. Множество всех базовых относительно $M$ путей в $P$ обозначается записью $\Pi_{P,M}$.

Докажем, что множество $\Pi_{P,M}$ конечно. Если $\Pi_{P,M}$ бесконечно, то оно содержит пути сколь угодно большой длины. Тогда в $\Pi_{P,M}$ есть путь $\pi$ вида $\alpha_1 \dots \alpha_i \dots \alpha_j \dots \alpha_k$, где $\alpha_i = \alpha_j$, $1 < i < j < k$. Последовательность $\alpha_i \dots \alpha_j$ является циклом, и, согласно определению базового множества, одно из рёбер этого цикла содержит точку из $M$, что противоречит предположению о том, что путь $\pi$ – базовый относительно $M$.

$\forall\, \pi \in \Pi_{P,M}$ будем обозначать записями $start(\pi)$ и $end(\pi)$ точки из $M$, которые лежат на первом и последнем ребре $\pi$ соответственно.

Базовое множество $M$ для БС $P$ называется **полным**, если

- на ребре, выходящем из начальной вершины $P$, есть точка из $M$ (такая точка называется **начальной**), и
- на каждом ребре, входящем в какую-либо терминальную вершину $P$, есть точка из $M$ (такие точки называются **терминальными**).

Пусть $M$ – полное базовое множество для БС $P$. Каждому выполнению $Exec$ БС $P$ соответствует последовательность $\pi = \alpha_1\alpha_2 \dots$ рёбер $P$, в которой каждое ребро $\alpha_i$ является тем ребром, по которому происходит перемещение на шаге $i$ этого выполнения от текущей вершины $v_i$ к вершине $v_{i+1}$. Обозначим записью $i_1 i_2 \dots$ последовательность номеров тех рёбер из $\pi$, на которых есть базовые точки. Согласно определению понятий полного базового множества и выполнения БС, $i_1 = 1$, и для каждой пары $(i_j, i_{j+1})$ соседних индексов в $i_1 i_2 \dots$ последовательность $\pi_j = \alpha_{i_j} \dots \alpha_{i_{j+1}}$ является базовым относительно $M$ путём. Если путь $\pi$ конечен, то его последнее ребро входит в терминальную вершину и, следовательно, содержит точку из $M$. Таким образом, можно представить $\pi$ в виде последовательности $\pi_1\pi_2 \dots$, где $\pi_1$, $\pi_2$, $\dots$ – базовые относительно $M$ пути. Каждый член $\pi_j$ этой последовательности мы будем называть **компонентой** выполнения $Exec$.

$\forall\, \pi \in \Pi_{P,M}, \forall\, \theta, \theta' \in Var(P)^\bullet$ запись $\theta \xrightarrow{\pi} \theta'$ означает, что $\pi$ – компонента некоторого выполнения БС $P$, и $\theta, \theta'$ – текущие подстановки в моменты прохода через точки $start(\pi)$ и $end(\pi)$ соответственно при движении по пути $\pi$ во время этого выполнения.

# 9 Описание метода инвариантов

Пусть заданы БС $P$ и её спецификация, выражаемая предусловием $Pre$ и постусловием $Post$. Доказательство утверждения $Pre \xrightarrow{P} Post$ **методом инвариантов** имеет следующий вид.

1. Выбирается полное базовое множество точек $M$ для $P$, и с каждой точкой $m \in M$ связывается формула $\varphi_m \in Fm$, называемая **инвариантом** в точке $m$, причем выполнены следующие условия:

   - если $m$ – начальная точка, то $\varphi_m = Pre$,
   - если $m$ – терминальная точка, то $\varphi_m = Post$,
   - для любого пути $\pi \in \Pi_{P,M}$ и любых подстановок $\theta, \theta' \in Var(P)^\bullet$, таких, что $\theta \xrightarrow{\pi} \theta'$, верна импликация

   $$\varphi^{\theta}_{start(\pi)} = 1 \quad \Rightarrow \quad \varphi^{\theta'}_{end(\pi)} = 1. \tag{9}$$

   Ниже, при анализе импликаций вида (9), $\forall\, x \in Var(P)$ будем обозначать значения $x^\theta$ и $x^{\theta'}$ записями $x$ и $x'$ соответственно.

2. Свойство завершаемости $P$ ($\forall\, \theta \in Var(P)^\bullet \;\; Pre^\theta = 1 \Rightarrow \theta \xrightarrow{P} \otimes$) обосновывается путем указания

   - базового множества $N$ для $P$,
   - частично упорядоченного множества $L$, являющегося **фундированным**, т.е. такого, что не существует бесконечной строго убывающей последовательности $l_0, l_1, \ldots$ элементов $L$, и
   - множества термов $\{u_n \in Tm(Var(P)) \mid n \in N\}$,

   таких, что

   - при каждом выполнении $P$, $\forall\, n \in N$ при каждом проходе через $n$ текущая подстановка $\theta$ удовлетворяет условию $u^\theta_n \in L$, и
   - для любого пути $\pi \in \Pi_{P,N}$ и любых подстановок $\theta, \theta' \in Var(P)^\bullet$, таких, что $\theta \xrightarrow{\pi} \theta'$, верно неравенство $u^{\theta}_{start(\pi)} > u^{\theta'}_{end(\pi)}$.

Изложенный в этом пункте метод инвариантов открыт Р. Флойдом .

# 10 Обоснование метода инвариантов

Докажем, что если выполнены условия в пунктах 1 и 2 параграфа 9, то $Pre \xrightarrow{P} Post$, т.е. $\forall\, \theta \in Var(P)^\bullet$ из $Pre^\theta = 1$ следует $\theta \xrightarrow{P} \otimes$ и $Post^{P\theta} = 1$.

Пусть $Exec$ – произвольное выполнение БС $P$ с начальной подстановкой $\theta$, удовлетворящей условию $Pre^\theta = 1$. Этому выполнению соответствует последовательность $\pi = \alpha_1\alpha_2 \ldots$ рёбер $P$, в которой каждое ребро $\alpha_i$ является тем ребром, по которому происходит перемещение на шаге $i$ этого выполнения от текущей вершины $v_i$ к вершине $v_{i+1}$.

Если бы выполнение $Exec$ было бесконечным, то $\pi$ тоже была бы бесконечной, и некоторое ребро встречалось бы в ней бесконечно много раз. Пусть $i_1 i_2 \ldots$ – бесконечная последовательность номеров рёбер из $\pi$, таких, что $\alpha_{i_1} = \alpha_{i_2} = \ldots$. Подпоследовательности $\pi_{i_j} = \alpha_{i_j} \ldots \alpha_{i_{j+1}}$ $(j \geq 1)$ последовательности $\pi$ являются циклами, и т.к. $N$ – базовое множество, то $\forall\, j \geq 1 \quad \pi_{i_j}$ содержит ребро, на котором есть точка из $N$. Таким образом, точки из $N$ присутствуют на бесконечном числе членов последовательности $\pi$. Обозначим номера таких членов записями $j_1, j_2, \ldots$, а точки из $N$ на них – записями $n_1$, $n_2$, .... Пути $\alpha_{j_k} \ldots \alpha_{j_{k+1}}$ $(k \geq 1)$ являются базовыми относительно $N$, поэтому, согласно пункту 2 параграфа 9, текущие подстановки $\theta_{j_1}, \theta_{j_2}, \ldots$ удовлетворяют неравенствам

$$u_{n_1}^{\theta_{j_1}} > u_{n_2}^{\theta_{j_2}} > \ldots, \tag{10}$$

т.е. в $L$ есть бесконечная строго убывающая последовательность (10), что противоречит предположению о фундированности $L$.

Таким образом, выполнение $Exec$ является конечным. В соответствии со сказанным в конце пункта 8 можно представить $\pi$ в виде конечной последовательности $\pi_1 \ldots \pi_k$ путей из $\Pi_{P,M}$.

Согласно определениям выполнения и полного базового множества, а также условиям в пункте 1 параграфа 9:

- $m_0 \stackrel{\text{def}}{=} start(\pi_1)$ – начальная точка, поэтому $\varphi_{m_0} = Pre$,
- $m_k \stackrel{\text{def}}{=} end(\pi_k)$ – терминальная точка, поэтому $\varphi_{m_k} = Post$, и
- $\forall\, i = 1, \ldots, k-1 \quad m_i \stackrel{\text{def}}{=} end(\pi_{i-1}) = start(\pi_i) \in M$.

Кроме того, $\forall\, i = 1, \ldots, k$ текущие подстановки $\theta_{i-1}$ и $\theta_i$ вычисления $Exec$ до и после прохождения компоненты $\pi_i$ последовательности $\pi_1 \ldots \pi_k$ соответственно удовлетворяют условию $\theta_{i-1} \stackrel{\pi_i}{\to} \theta_i$, поэтому, согласно (9),

$$\forall\, i = 1, \ldots, k \quad \varphi_{m_{i-1}}^{\theta_{i-1}} = 1 \;\Rightarrow\; \varphi_{m_i}^{\theta_i} = 1.$$

Суммируя все вышесказанное, получаем цепочку импликаций:

$$(Pre^{\theta} = 1) \Rightarrow (\varphi_{m_0}^{\theta} = 1) \Rightarrow (\varphi_{m_1}^{\theta_1} = 1) \Rightarrow \ldots$$
$$\ldots \Rightarrow (\varphi_{m_k}^{\theta_k} = 1) \Rightarrow (Post^{P\theta} = 1). \blacksquare$$

# 11 Примеры фундированных множеств

В качестве фундированных множеств, требуемых для обоснования завершаемости, можно брать, например, следующие множества:

- множество натуральных чисел $\mathbf{N} = \{0, 1, \ldots\}$,
- множество $L^k$ кортежей длины $k$ элементов произвольного фундированного множества $L$, с лексикографическим порядком, который определяется следующим образом: для любой пары кортежей $a = (a_1, \ldots, a_k)$, $b = (b_1, \ldots, b_k)$ из $L^k$

  $a \leq b \Leftrightarrow a = b$ или $\exists\, i \in \{1, \ldots, k\} : a_i < b_i, \forall\, j \in \{1, \ldots, i-1\}\ a_j = b_j$.

  Обоснуем фундированность этого множества. Пусть существует бесконечная строго убывающая последовательность

  $$(a_1^1, \ldots, a_k^1) > (a_1^2, \ldots, a_k^2) > \ldots \tag{11}$$

  элементов множества $L^k$. Из (11) и из определения лексикографического порядка следует, что $a_1^1 \geq a_1^2 \geq \ldots$. Поскольку $L$ фундировано, то в этой последовательности неравенств не может быть бесконечного количества строгих неравенств, т.е.

  $$\exists\, i_1 :\ a_1^{i_1} = a_1^{i_1+1} = \ldots. \tag{12}$$

  Из последнего соотношения и из (11) следует, что

  $$(a_2^{i_1}, \ldots, a_k^{i_1}) > (a_2^{i_1+1}, \ldots, a_k^{i_1+1}) > \ldots \tag{13}$$

  Применяя изложенные выше рассуждения к (13), получаем, что

  $$\exists\, i_2 \geq i_1 :\ a_2^{i_2} = a_2^{i_2+1} = \ldots, \tag{14}$$

  откуда на основании (13) следует, что

  $$(a_3^{i_2}, \ldots, a_k^{i_2}) > (a_3^{i_2+1}, \ldots, a_k^{i_2+1}) > \ldots.$$

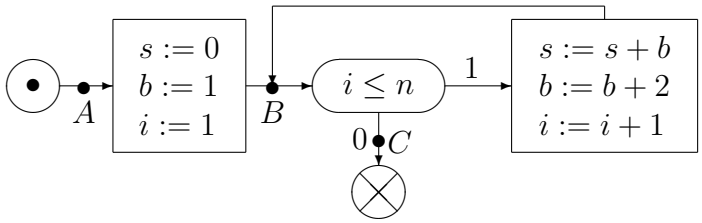
  Продолжая так и дальше, в конце концов получим, что

  $$\exists\, i_k \geq i_{k-1} :\ a_k^{i_k} = a_k^{i_k+1} = \ldots. \tag{15}$$

  Из (12), (14), (15) следует, что кортежи в (11), начиная с кортежа с номером $i_k$, совпадают, что противоречит предположению. ■

# 12 Верификация блок-схемы вычисления суммы

Верифицируем БС вычисления суммы относительно предусловия $n \geq 1$ и постусловия $s = n^2$. В качестве множества $M$ точек, необходимого для верификации этой БС методом инвариантов, возьмем множество $\{A, B, C\}$ точек, изображенных ниже:




Инварианты в точках $A, B, C$ имеют следующий вид:

$$\varphi_A = (n \geq 1) \quad \text{(предусловие)},$$
$$\varphi_B = (s = (i-1)^2) \wedge (b = 2i - 1) \wedge (i \leq n + 1),$$
$$\varphi_C = (s = n^2) \quad \text{(постусловие)}.$$

Для доказательства того, что инвариант $\varphi_B$ определен правильно, необходимо исследовать все этапы выполнения этой БС, которые начинаются и заканчиваются проходом через точки из $M$.

1. На первом этапе выполнения этой БС происходит переход от $A$ к $B$, с выполнением присваиваний в блоке между $A$ и $B$ (т.е $s := 0$, $b := 1$ и $i := 1$), и (9) имеет вид

$$n \geq 1 \;\Rightarrow\; \left\{ \begin{array}{l} s' = (i'-1)^2 \\ b' = 2i' - 1 \\ i' \leq n' + 1 \end{array} \right\}. \tag{16}$$

   Поскольку $s' = 0$, $b' = 1$, $i' = 1$ и $n' = n$, то (16) переписывается в виде

$$n \geq 1 \;\Rightarrow\; \left\{ \begin{array}{l} 0 = 0^2 \\ 1 = 2 \cdot 1 - 1 \\ 1 \leq n + 1 \end{array} \right\}, \tag{17}$$

   что, очевидно, верно.

2. Рассмотрим произвольный этап выполнения этой БС, на котором происходит переход от $B$ к $B$ по циклу, здесь

   - выполняются присваивания $s := s + b$, $b := b + 2$ и $i := i + 1$, и

- верно условие, на основании которого возможно движение по этому циклу ($i \le n$).

Импликация (9) с учетом этого условия имеет вид

$$\left\{ \begin{array}{l} s = (i-1)^2 \\ b = 2i - 1 \\ i \le n+1 \\ i \le n \end{array} \right\} \Rightarrow \left\{ \begin{array}{l} s' = (i'-1)^2 \\ b' = 2i' - 1 \\ i' \le n'+1 \end{array} \right\}. \qquad (18)$$

Учитывая соотношения $s' = s + b$, $b' = b + 2$ $i' = i + 1$, $n' = n$, заключаем, что импликация (18) верна.

3. Рассмотрим этап выполнения БС, на котором происходит переход от $B$ к $C$. Он возможен только при условии $i > n$. Импликация (9) с учетом этого условия имеет вид

$$\left\{ \begin{array}{l} s = (i-1)^2 \\ b = 2i - 1 \\ i \le n+1 \\ i > n \end{array} \right\} \Rightarrow s' = (n')^2. \qquad (19)$$

Учитывая равенства $s' = s$, $n' = n$, а также то, что $i$ имеет тип **I**, и, следовательно, из $i \le n+1$ и $i > n$ следует, что $i = n+1$, заключаем, что импликация (19) верна.

Для доказательства завершаемости положим

$$N \stackrel{\text{def}}{=} \{B\},\ L \stackrel{\text{def}}{=} \mathbf{N},\ u_B \stackrel{\text{def}}{=} n + 1 - i.$$

Из доказанного выше соотношения $\varphi_B$ следует, что при каждом проходе через $B$ текущая подстановка $\theta$ удовлетворяет условию $u_B^\theta \in \mathbf{N}$, и при произвольном переходе от $B$ к $B$ по циклу

$$u_B^\theta = (n + 1 - i) > (n + 1 - (i + 1)) = (n' + 1 - i') = u_B^{\theta'},$$

где $\theta$ и $\theta'$ – текущие подстановки до и после прохода по циклу.

# Межфакультетский курс «Методы искусственного интеллекта в задачах анализа данных и верификации программ» Лекция 2 Примеры верификации алгоритмов

А.М.Миронов

В этом пункте мы рассмотрим применение метода инвариантов для верификации различных БС.

Напомним, что задача верификации БС заключается в доказательстве утверждения $Pre \xrightarrow{P} Post$ **методом инвариантов**, который имеет следующий вид.

1. Выбирается полное базовое множество точек $M$ для $P$, и с каждой точкой $m \in M$ связывается формула $\varphi_m \in Fm$, называемая **инвариантом** в точке $m$, причем выполнены следующие условия:

   - если $m$ – начальная точка, то $\varphi_m = Pre$,
   - если $m$ – терминальная точка, то $\varphi_m = Post$,
   - для любого пути $\pi \in \Pi_{P,M}$ и любых подстановок $\theta, \theta' \in Var(P)^\bullet$, таких, что $\theta \xrightarrow{\pi} \theta'$, верна импликация

   $$\varphi^{\theta}_{start(\pi)} = 1 \quad \Rightarrow \quad \varphi^{\theta'}_{end(\pi)} = 1. \tag{1}$$

   Ниже, при анализе импликаций вида (1), $\forall\, x \in Var(P)$ будем обозначать значения $x^\theta$ и $x^{\theta'}$ записями $x$ и $x'$ соответственно.

2. Свойство завершаемости $P$ ($\forall\, \theta \in Var(P)^\bullet \;\; Pre^\theta = 1 \Rightarrow \theta \xrightarrow{P} \otimes$) обосновывается путем указания

   - базового множества $N$ для $P$,

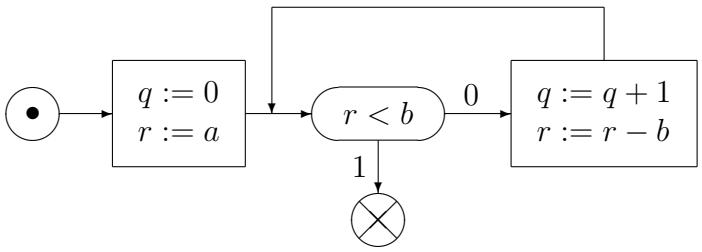
- частично упорядоченного множества $L$, являющегося **фундированным**, т.е. такого, что не существует бесконечной строго убывающей последовательности $l_0, l_1, \ldots$ элементов $L$, и
- множества термов $\{u_n \in Tm(Var(P)) \mid n \in N\}$,

таких, что

- при каждом выполнении $P$, $\forall\, n \in N$ при каждом проходе через $n$ текущая подстановка $\theta$ удовлетворяет условию $u_n^\theta \in L$, и
- для любого пути $\pi \in \Pi_{P,N}$ и любых подстановок $\theta, \theta' \in Var(P)^\bullet$, таких, что $\theta \xrightarrow{\pi} \theta'$, верно неравенство $u^\theta_{start(\pi)} > u^{\theta'}_{end(\pi)}$.

# 1 Примеры блок-схем

1. БС, вычисляющая частное и остаток от целочисленного деления $a$ на $b$, где $a$ на $b$ – положительные целые числа. В результате выполнения данной программы в переменные $q$ и $r$ должны быть занесены частное и остаток соответственно от деления $a$ на $b$, т.е. должны быть выполнены соотношения $a = bq + r$ и $0 \le r < b$.



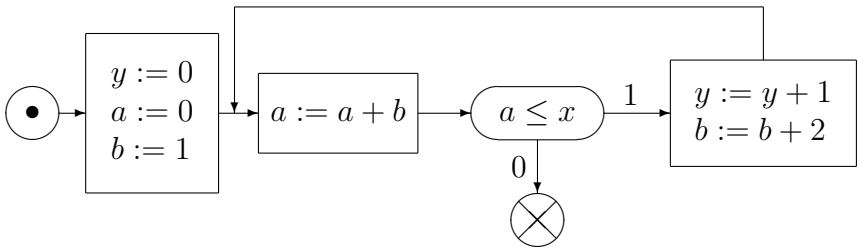

(2)

   Все переменные в данной БС имеют тип **I**.

   Алгоритм, реализованный в данной БС, заключается в вычитании $b$ из $a$ до тех пор, пока результат не станет меньше $b$. Число этих вычитаний равно частному, а результат вычитаний – остатку от целочисленного деления $a$ на $b$.

2. БС, вычисляющая целую часть числа $\sqrt{x}$, где $x$ – неотрицательное

целое число. Результат должен быть занесён в переменную $y$.

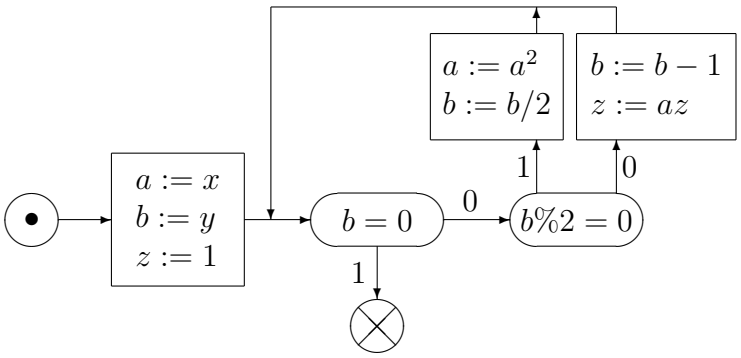



(3)

Все переменные в данной БС имеют тип **I**.

Алгоритм, реализованный в данной БС, заключается в использовании тождества $1+3+5+\ldots+(2n-1) = n^2$. Переменная $b$ принимает последовательно значения $1, 3, 5, \ldots$, эти значения суммируются, и результат заносится в переменную $a$, а число слагаемых в этой сумме заносится в переменную $y$. Когда значение $a$ станет превосходить $x$, из приведенного выше тождества следует, что $y$ содержит требуемое значение $\lfloor \sqrt{x} \rfloor$.

3. БС, реализующая быстрый алгоритм возведения в степень. В результате работы данной БС вычисляется значение $x^y$, где $y \geq 0$. Результат должен быть занесен в переменную $z$.




(4)

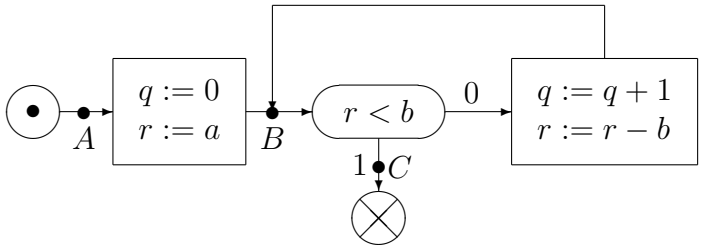
Все переменные в данной БС имеют тип **I**. Исходное значение $y$ предполагается неотрицательным. Вычисление $x^y$ производится по следующему принципу:

- если $y = 0$, то $x^y = 1$,
- если $y$ – положительное четное, то $x^y = (x^2)^{y/2}$, и
- если $y$ – нечетное, то $x^y = x^{y-1}x$.

Данный принцип позволяет понизить число умножений при вычислении $x^y$ с $y - 1$ до $O(\log_2 y)$.

# 2 Верификация блок-схемы деления с остатком

Верифицируем БС (2) относительно предусловия $(a \geq 0) \wedge (b \geq 1)$ и постусловия $(a = bq + r) \wedge (0 \leq r < b)$. Выберем точки $A$, $B$, $C$, как показано на рисунке:




Инвариантами $\varphi_A$ и $\varphi_C$ являются предусловие и постусловие соответственно, а инвариант $\varphi_B$ имеет вид $(a = bq + r) \wedge (r \geq 0)$.

Докажем, что $\varphi_B$ определен правильно.

1. При первом входе в $B$ $\varphi_B$ имеет вид $(a = b \cdot 0 + a) \wedge (a \geq 0)$, истинность данного соотношения следует из $\varphi_A$.

2. При переходе от $B$ к $B$ по циклу верно дополнительное условие $r \geq b$, с учетом которого импликация (1) имеет вид

$$\left\{ \begin{array}{l} a = bq + r \\ r \geq 0 \\ r \geq b \end{array} \right\} \Rightarrow \left\{ \begin{array}{l} a' = b'q' + r' \\ r' \geq 0 \end{array} \right\}. \qquad (5)$$

   Учитывая соотношения $a' = a$, $b' = b$, $q' = q + 1$, $r' = r - b$, заключаем, что импликация (5) верна.

3. При переходе от $B$ к $C$ верно дополнительное условие $r < b$, что в сочетании с $\varphi_B$ влечёт $\varphi_C$.

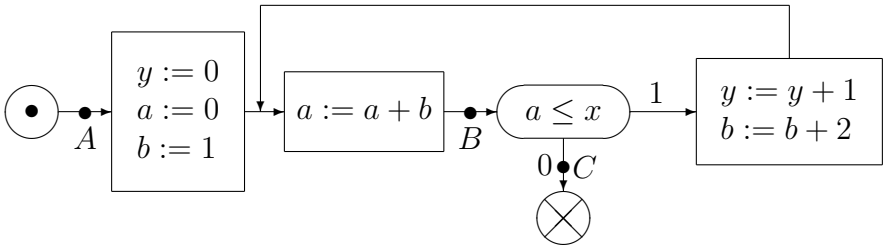
Для доказательства завершаемости положим $N \stackrel{\text{def}}{=} \{B\}$, $L \stackrel{\text{def}}{=} \mathbf{N}$, $u_B \stackrel{\text{def}}{=} r$. Из доказанного выше соотношения $\varphi_B$ следует, что при каждом проходе через $B$ текущая подстановка $\theta$ удовлетворяет условию $u_B^\theta \in \mathbf{N}$, и с учетом дополнительного соотношения $b \geq 1$ (истинность которого в точке $B$ следует из $\varphi_A$ и из того, что значение переменной $b$ не изменяется в процессе выполнения БС) заключаем, что при произвольном переходе от $B$ к $B$ по циклу

$$u_B^\theta = r > r - b = r' = u_B^{\theta'},$$

где $\theta$ и $\theta'$ – текущие подстановки до и после прохода по циклу.

## 2.1 Верификация блок-схемы извлечения корня

Верифицируем БС (3) относительно предусловия $x \geq 0$ и постусловия $y = \lfloor \sqrt{x} \rfloor$. Выберем точки $A$, $B$, $C$, как показано на рисунке:




Определим инварианты в выбранных точках следующим образом:

$$\varphi_A \stackrel{\text{def}}{=} (x \geq 0),\ \varphi_B \stackrel{\text{def}}{=} \left\{ \begin{array}{l} y \geq 0 \\ y^2 \leq x \\ a = (y+1)^2 \\ b = 2y+1 \end{array} \right\},\ \varphi_C \stackrel{\text{def}}{=} \left\{ \begin{array}{l} y \geq 0 \\ y^2 \leq x < (y+1)^2 \end{array} \right\}$$

(нетрудно видеть, что $\varphi_C$ эквивалентна постусловию).

Докажем, что $\varphi_B$ определен правильно.

1. При первом входе в $B$ $\varphi_B$ имеет вид $\left\{ \begin{array}{l} 0 \geq 0 \\ 0^2 \leq x \\ 1 = (0+1)^2 \\ 1 = 2 \cdot 0 + 1 \end{array} \right\}$, истинность данного соотношения следует из $\varphi_A$.

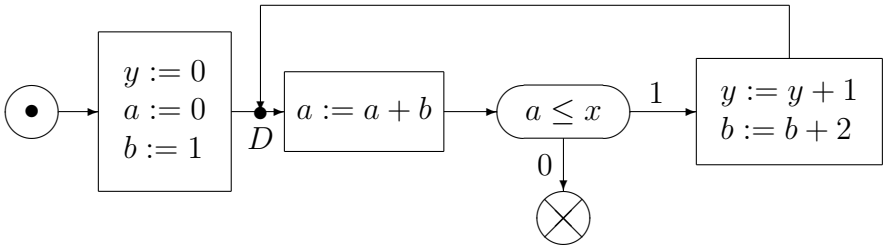
2. При переходе от $B$ к $B$ по циклу верно дополнительное условие $a \leq x$, с учетом которого импликация (1) имеет вид

$$\left\{ \begin{array}{l} y \geq 0 \\ y^2 \leq x \\ a = (y+1)^2 \\ b = 2y+1 \\ a \leq x \end{array} \right\} \Rightarrow \left\{ \begin{array}{l} y' \geq 0 \\ (y')^2 \leq x' \\ a' = (y'+1)^2 \\ b' = 2y'+1 \end{array} \right\}. \qquad (6)$$

   Учитывая соотношения $y' = y+1$, $b' = b+2$, $a' = a+b+2$, $x' = x$, нетрудно установить, что импликация (6) верна.

3. При переходе от $B$ к $C$ верно дополнительное условие $x < a$, что в сочетании с $\varphi_B$ влечёт $\varphi_C$.

Докажем завершаемость данной БС. Положим $N \stackrel{\text{def}}{=} \{D\}$, где точка $D$ выбрана так, как показано на рисунке:




и, кроме того, $L \stackrel{\text{def}}{=} \mathbf{N}$, $u_D \stackrel{\text{def}}{=} x - a$. Для обоснования того, что

- при каждом проходе через $D$ текущая подстановка $\theta$ удовлетворяет условию $u_D^\theta \in \mathbf{N}$ и
- при переходе от $D$ к $D$ по циклу $u_D^\theta = x - a > x' - a' = u_D^{\theta'}$, где $\theta$ и $\theta'$ – текущие подстановки до и после прохода по циклу,

определим вспомогательный инвариант в $D$: $\varphi_D \stackrel{\text{def}}{=} (x \geq a) \wedge (b \geq 1)$.

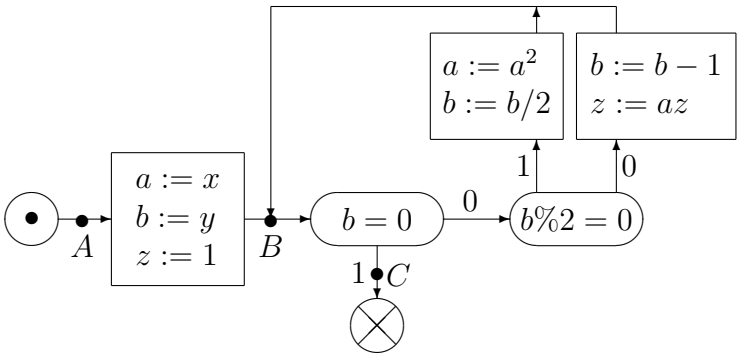
Истинность $\varphi_D$ при каждом проходе через $D$ следует из того, что

- $\varphi_D$ верно при первом проходе через $D$ (это следует из $\varphi_A$) и
- при произвольном переходе от $D$ к $D$ по циклу верно соотношение, написанное в условном операторе БС ($a \leq x$).

Из истинности $\varphi_D$ при каждом проходе через $D$ следуют вышеупомянутые свойства, связанные с точкой $D$.

## 2.2 Верификация блок-схемы возведения в степень

Верифицируем БС (4) относительно предусловия $y \geq 0$ и постусловия $z = x^y$. Выберем точки $A$, $B$, $C$, как показано на рисунке:



Определим инварианты в выбранных точках следующим образом:

$$\varphi_A \stackrel{\text{def}}{=} (y \geq 0),\ \varphi_B \stackrel{\text{def}}{=} \left\{ \begin{array}{l} b \geq 0 \\ y \geq 0 \\ za^b = x^y \end{array} \right\},\ \varphi_C \stackrel{\text{def}}{=} (z = x^y).$$

Нетрудно доказать, что $\varphi_B$ определен правильно.

Для доказательства завершаемости данной БС можно взять $N \stackrel{\text{def}}{=} \{B\}$, $L \stackrel{\text{def}}{=} \mathbf{N}$, $u_B \stackrel{\text{def}}{=} b$.

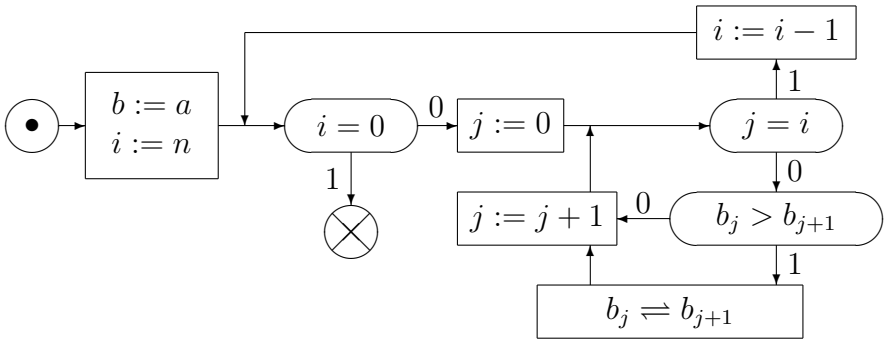
# Межфакультетский курс «Методы искусственного интеллекта в задачах анализа данных и верификации программ» Лекция 3 Верификация алгоритма сортировки массива

А.М.Миронов

## 1 Блок-схема сортировки массива

БС, реализующая сортировку массива методом пузырька, определяется следующим образом.

Сортируемый массив имеет вид $a_{0..n}$, где $n \geq 0$. Результат должен быть записан в массив $b_{0..n}$. Требуется, чтобы после завершения выполнения этой БС были верны утверждения $b = perm(a)$ и $ord(b)$.




(1)

В данном примере и ниже используется следующее обозначение: если $a$ и $b$ – массивы с одинаковыми типами компонентов и одинаковыми нижними и верхними индексами (т.е. имеют вид $a_{m..n}$ и $b_{m..n}$), то $a := b$ обозначает операцию копирования компонентов массива $b$ в соответствующие

компоненты массива $a$, т.е. последовательность присваиваний $a_m := b_m$, $a_{m+1} := b_{m+1}$, ..., $a_n := b_n$.

Алгоритм, реализованный в БС (1), заключается в выполнении $n$ прогонок по массиву $b$, каждая из которых заключается в просмотре слева направо участка от 0-й до $i$-й позиции сортируемого массива и перемене местами его соседних элементов, если они нарушают порядок. Сначала производится прогонка по всему массиву, в результате этой прогонки в последнюю позицию массива помещается его максимальный элемент, в результате следующей прогонки в предпоследнюю позицию помещается следующий по величине элемент и т.д.

# 2 Верификация блок-схемы сортировки

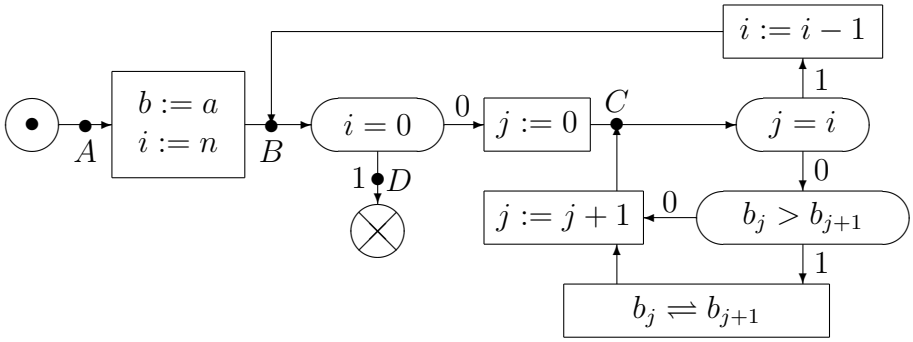
Верифицируем БС (1) относительно предусловия $n \geq 0$ и постусловия $(b = perm(a)) \wedge ord(b)$.

Конъюнктивные члены в постусловии обосновываются раздельно.

Обоснование утверждения $b = perm(a)$ заключается в том, что

- оно является верным после выполнения первого присваивания и
- все остальные действия в этой БС сохраняют его истинность, т.к. единственное действие в БС, которое изменяет массив $b$, это перестановка его соседних компонентов ($b_j \rightleftharpoons b_{j+1}$), однако эта перестановка не изменяет содержимое массива $b$.

Для обоснования утверждения $ord(b)$ выберем точки $A$, $B$, $C$, $D$, как показано на рисунке:




Инвариантами $\varphi_A$ и $\varphi_D$ являются формулы ($n \geq 0$) и $ord(b_{0..n})$ соответ-

ственно. Инварианты $\varphi_B$ и $\varphi_C$ имеют следующий вид:

$$\varphi_B \stackrel{\text{def}}{=} \left\{ \begin{array}{l} 0 \le i \le n \\ ord(b_{i..n}) \\ b_{0..i} \le b_{i+1..n} \end{array} \right\}, \quad \varphi_C \stackrel{\text{def}}{=} \left\{ \begin{array}{l} 1 \le i \le n \\ 0 \le j \le i \\ ord(b_{i..n}) \\ b_{0..i} \le b_{i+1..n} \\ b_{0..j-1} \le b_j \end{array} \right\}.$$

Докажем, что инварианты $\varphi_B$ и $\varphi_C$ определены правильно.

1. При первом входе в $B$ $i' = n$ и $\varphi_B^{\theta'} = \left\{ \begin{array}{l} 0 \le n \le n \\ ord(b_{n..n}) \\ b_{0..n} \le b_{n+1..n} \end{array} \right\}$. Истинность соотношения

   $$\varphi^{\theta}_{start(\pi)} = 1 \quad \Rightarrow \quad \varphi^{\theta'}_{end(\pi)} = 1$$

   следует из $\varphi_A$.

2. При переходе от $B$ к $C$ верно дополнительное условие $i \neq 0$. Кроме того, меняет свое значение только переменная $j$ ($j' = 0$). С учетом этого условие корректности

   $$\varphi^{\theta}_{start(\pi)} = 1 \quad \Rightarrow \quad \varphi^{\theta'}_{end(\pi)} = 1$$

   имеет вид

   $$\left\{ \begin{array}{l} 0 \le i \le n \\ ord(b_{i..n}) \\ b_{0..i} \le b_{i+1..n} \\ i \neq 0 \end{array} \right\} \Rightarrow \left\{ \begin{array}{l} 1 \le i \le n \\ 0 \le 0 \le i \\ ord(b_{i..n}) \\ b_{0..i} \le b_{i+1..n} \\ b_{0..0-1} \le b_0 \end{array} \right\}. \qquad (2)$$

   Нетрудно видеть, что импликация (2) верна.

3. При переходе от $C$ к $B$ верно дополнительное условие $j = i$. Кроме того, меняет свое значение только $i$. С учетом этого доказываемая импликация

   $$\varphi^{\theta}_{start(\pi)} = 1 \quad \Rightarrow \quad \varphi^{\theta'}_{end(\pi)} = 1$$

   имеет вид

   $$\left\{ \begin{array}{l} 1 \le i \le n \\ 0 \le j \le i \\ ord(b_{i..n}) \\ b_{0..i} \le b_{i+1..n} \\ b_{0..j-1} \le b_j \\ j = i \end{array} \right\} \Rightarrow \left\{ \begin{array}{l} 0 \le i' \le n \\ ord(b_{i'..n}) \\ b_{0..i'} \le b_{i'+1..n} \end{array} \right\}.$$

Поскольку $i' = i - 1$, то последнюю импликацию можно переписать в виде

$$\left\{ \begin{array}{l} 1 \le i \le n \\ ord(b_{i..n}) \\ b_{0..i} \le b_{i+1..n} \\ b_{0..i-1} \le b_i \end{array} \right\} \Rightarrow \left\{ \begin{array}{l} 0 \le i - 1 \le n \\ ord(b_{i-1..n}) \\ b_{0..i-1} \le b_{i..n} \end{array} \right\}. \qquad (3)$$

Нетрудно видеть, что импликация (3) верна.

4. При переходе от $C$ к $C$ по короткому циклу верны дополнительные условия $j \neq i$ и $b_j \le b_{j+1}$. Кроме того, меняет свое значение только переменная $j$ ($j' = j+1$). С учетом этого доказываемая импликация

$$\varphi^{\theta}_{start(\pi)} = 1 \quad \Rightarrow \quad \varphi^{\theta'}_{end(\pi)} = 1$$

имеет вид

$$\left\{ \begin{array}{l} 1 \le i \le n \\ 0 \le j < i \\ ord(b_{i..n}) \\ b_{0..i} \le b_{i+1..n} \\ b_{0..j-1} \le b_j \\ b_j \le b_{j+1} \end{array} \right\} \Rightarrow \left\{ \begin{array}{l} 1 \le i \le n \\ 0 \le j + 1 \le i \\ ord(b_{i..n}) \\ b_{0..i} \le b_{i+1..n} \\ b_{0..j} \le b_{j+1} \end{array} \right\}. \qquad (4)$$

Нетрудно видеть, что импликация (4) верна.

5. При переходе от $C$ к $C$ по длинному циклу верны дополнительные условия $j \neq i$ и $b_j > b_{j+1}$. Кроме того, меняют свое значение переменная $j$ и массив $b$ ($j' = j + 1$, $b'_j = b_{j+1}$, $b'_{j+1} = b_j$, $\forall\, k \in \{0, \ldots, n\} \setminus \{j, j+1\} \quad b'_k = b_k$). С учетом этого, соотношение

$$\varphi^{\theta}_{start(\pi)} = 1 \quad \Rightarrow \quad \varphi^{\theta'}_{end(\pi)} = 1$$

имеет вид

$$\left\{ \begin{array}{l} 1 \le i \le n \\ 0 \le j < i \\ ord(b_{i..n}) \\ b_{0..i} \le b_{i+1..n} \\ b_{0..j-1} \le b_j \\ b_{j+1} < b_j \end{array} \right\} \Rightarrow \left\{ \begin{array}{l} 1 \le i \le n \\ 0 \le j + 1 \le i \\ ord(b'_{i..n}) \\ b'_{0..i} \le b'_{i+1..n} \\ b'_{0..j} \le b'_{j+1} \end{array} \right\}. \qquad (5)$$

Отдельно рассмотрим случаи $j + 1 < i$ и $j + 1 = i$.

- Если $j+1<i$, то

$$\left\{\begin{array}{l} b'_0=b_0,\ldots,b'_{j-1}=b_{j-1},\\ b'_j=b_{j+1},b'_{j+1}=b_j,\\ b'_{j+2}=b_{j+2},\ldots,b'_i=b_i,\ldots,b'_n=b_n,\end{array}\right.$$

откуда нетрудно обосновать (5).

- Если $j+1=i$, то (5) следует из импликации

$$\left\{\begin{array}{l} 1\le i\le n\\ ord(b_{i..n})\\ b_{0..i}\le b_{i+1..n}\\ b_{0..i-2}\le b_{i-1}\\ b_i<b_{i-1}\end{array}\right\}\Rightarrow\left\{\begin{array}{l} ord(b'_{i..n})\\ b'_{0..i}\le b'_{i+1..n}\\ b'_{0..i-1}\le b'_i\end{array}\right\}\qquad(6)$$

при условии $\left\{\begin{array}{l} b'_0=b_0,\ldots,b'_{i-2}=b_{i-2},\\ b'_{i-1}=b_i,b'_i=b_{i-1},\\ b'_{i+1}=b_{i+1},\ldots,b'_n=b_n.\end{array}\right.$

6. При переходе от $B$ к $D$ верно дополнительное условие $i=0$, что в сочетании с $\varphi_B$ влечёт $\varphi_D$.

Докажем завершаемость БС (1). Положим $N\stackrel{\text{def}}{=}\{C\}$, $L\stackrel{\text{def}}{=}\mathbf{N}^2$ (лексикографический порядок), $u_C\stackrel{\text{def}}{=}(i,i-j)$. Из $\varphi_C$ следует, что при проходе через $C$ текущая подстановка $\theta$ удовлетворяет условию $u_C^\theta\in\mathbf{N}^2$ и при переходе от $C$ к $C$ по верхнему циклу $i'=i-1$, поэтому

$$u_C^\theta=(i,i-j)>(i-1,\ldots)=u_C^{\theta'},$$

а при переходе от $C$ к $C$ по нижним циклам $i'=i$, $j'=j+1$, поэтому

$$u_C^\theta=(i,i-j)>(i,i-(j+1))=(i',i'-j')=u_C^{\theta'},$$

где $\theta$ и $\theta'$ – текущие подстановки до и после прохода по циклам.

# Межфакультетский курс «Методы искусственного интеллекта в задачах анализа данных и верификации программ» Лекция 4 Верификация функциональной программы сортировки строки

А.М.Миронов

## 1 Понятие функциональной программы

Для неформальной иллюстрации понятия функциональной программы мы рассмотрим некоторые примеры функциональных программ, описывающих функции на символьных строках (которые мы будем называть просто строками).

**Функциональная программа (ФП)** представляет собой систему функциональных уравнений. Функция, которую определяет ФП, является первой компонентой решения этой системы функциональных уравнений.

### 1.1 Конкатенация строк

Приводимая ниже ФП определяет функцию на строках

$$append : D_{\mathbf{S}} \times D_{\mathbf{S}} \to D_{\mathbf{S}},$$

которая преобразует пару строк $(u, v)$ в их **конкатенацию**, т.е.

- в строку $a_1 \dots a_n\, b_1 \dots b_m$, если $u$ и $v$ имеют вид соответственно

$$a_1 \dots a_n \quad \text{и} \quad b_1 \dots b_m \quad (n, m > 0),$$

- в строку $v$, если $u = \varepsilon$, и в строку $u$, если $v = \varepsilon$.

ФП, описывающая функцию *append*, имеет вид

$$\varphi(u, v) = \quad (u = \varepsilon)?v : u_h\varphi(u_t, v). \tag{1}$$

ФП (1) состоит из одного функционального уравнения, неизвестная величина в котором – функциональная переменная $\varphi$. Левая часть уравнения (1) состоит из функциональной переменной $\varphi$, соответствующей описываемой функции, и списка формальных параметров этой функции ($u$ и $v$). Правая часть уравнения (1) представляет собой выражение, описывающее связь значения описываемой функции на паре аргументов $(u, v)$ с её значениями на других аргументах. Уравнение (1) можно рассматривать как рекурсивный алгоритм вычисления функции *append*.

Нетрудно доказать, что функция *append* является единственным решением функционального уравнения (1).

Ниже будем обозначать терм $append(u, v)$ записью $u * v$ (или $uv$).

## 1.2 Инвертирование строки

Другим примером является ФП, описывающая функцию

$$reverse : D_{\mathbf{S}} \to D_{\mathbf{S}},$$

которая преобразует каждую строку $u$ в строку, получаемую из $u$ её записью в обратном порядке, т.е. $reverse(a_1 \dots a_n) = a_n \dots a_1$.

ФП, описывающая функцию $reverse$, имеет вид

$$\varphi(u) = (u = \varepsilon)?\varepsilon : \varphi(u_t) * (u_h\varepsilon).$$

В этой ФП используется функция *append*, которую мы описали в виде ФП в предыдущем параграфе.

## 1.3 Поиск подстроки

Третьим примером является ФП, описывающая функцию

$$find : D_{\mathbf{S}} \times D_{\mathbf{S}} \to D_{\mathbf{B}}.$$

Значением данной функции на паре строк $u, v$ является 1, если $u$ является подстрокой строки $v$, т.е. если $v$ является значением выражения $x * (u * y)$ для некоторых $x, y$, и 0 иначе.

ФП, описывающая функцию $find$, имеет вид

$$\begin{cases} \varphi_1(u,v) = \varphi_2(u,v)?1 : \Big((v=\varepsilon)?0 : \varphi_1(u,v_t)\Big) \\ \varphi_2(u,v) = (u=\varepsilon)?1 : \Big((v=\varepsilon)?0 : ((u_h=v_h)?\varphi_2(u_t,v_t) : 0)\Big) \end{cases}.$$

Данная ФП представляет собой систему из двух функциональных уравнений. Эта система имеет единственное решение, представляющее собой пару функций

$$(find,\ prefix)$$

($find$ соответствует $\varphi_1$, а *prefix* – $\varphi_2$), где функция $find$ является той функцией, для описания которой предназначена данная ФП, и функция *prefix* является вспомогательной функцией, она имеет вид

$$prefix : D_{\mathbf{S}} \times D_{\mathbf{S}} \to D_{\mathbf{B}};$$

её значением на паре строк $u, v$ является 1, если $u$ – префикс $v$, т.е. $v$ имеет вид $u * x$ для некоторой $x$, и 0 иначе.

# 2 Верификация функциональной программы сортировки строки

Требуется доказать, что $\forall\, x \in D_{\mathbf{S}}$

$$\mathbf{ord}(\mathbf{sort}(x)) = 1, \tag{2}$$

где

- **sort** – функция сортировки строк, определяемая ФП:

$$\begin{cases} \mathbf{sort}(x) = (x=\varepsilon)?\ \varepsilon : \mathbf{insert}(x_h, \mathbf{sort}(x_t)) \\ \mathbf{insert}(a,y) = (y=\varepsilon) \quad ?\ a\varepsilon \\ \qquad\qquad\qquad\quad : (a \le y_h) \quad ?\ ay : y_h\, \mathbf{insert}(a, y_t) \end{cases}$$

- **ord** – функция проверки упорядоченности строки:

$$\begin{cases} \mathbf{ord}(x) = (x=\varepsilon) \quad ?\ 1 \\ \qquad\qquad\quad : (x_t=\varepsilon) \quad ?\ 1 \\ \qquad\qquad\qquad\quad : (x_h \le (x_t)_h) \quad ?\ \mathbf{ord}(x_t) : 0 \end{cases}$$

Ниже мы будем обозначать терм вида $\mathbf{insert}(a, y)$ записью $a \to y$.
Если $x = \varepsilon$, то $\mathbf{sort}(x) = \varepsilon$ и

$$\mathbf{ord}(\mathbf{sort}(x)) = \mathbf{ord}(\varepsilon) = 1.$$

Если $x \neq \varepsilon$, то доказываемое равенство (2) переписывается в виде

$$\mathbf{ord}(x_h \to \mathbf{sort}(x_t)) = 1. \tag{3}$$

По индуктивному предположению, верно равенство

$$\mathbf{ord}(\mathbf{sort}(x_t)) = 1,$$

из которого следует (3) по нижеследующей лемме.

**Лемма**
Имеет место импликация

$$\mathbf{ord}(y) = 1 \quad \Rightarrow \quad \mathbf{ord}(a \to y) = 1. \tag{4}$$

**Доказательство**.
Доказываем лемму индукцией по $y$.
Если $y = \varepsilon$, то правая часть в (4) имеет вид

$$\mathbf{ord}(a\varepsilon) = 1,$$

что верно по определению **ord**.
Пусть $y \neq \varepsilon$ и для каждого $z < y$ верна импликация

$$\mathbf{ord}(z) = 1 \quad \Rightarrow \quad \mathbf{ord}(a \to z) = 1. \tag{5}$$

Обозначим $c \stackrel{\text{def}}{=} y_h$, $d \stackrel{\text{def}}{=} y_t$.
(4) имеет вид

$$\mathbf{ord}(cd) = 1 \quad \Rightarrow \quad \mathbf{ord}(a \to cd) = 1. \tag{6}$$

Для доказательства импликации (6) нужно доказать, что при условии $\mathbf{ord}(cd) = 1$ верны импликации

(a) $a \leq c \quad \Rightarrow \quad \mathbf{ord}(a(cd)) = 1,$

(b) $c < a \quad \Rightarrow \quad \mathbf{ord}(c(a \to d)) = 1.$

(a) верно потому, что из $a \leq c$ следует

$$\mathbf{ord}(a(cd)) = \mathbf{ord}(cd) = 1.$$

Докажем (b).

- $d = \varepsilon$. В этом случае правая часть в (b) имеет вид

$$\mathbf{ord}(c(a\varepsilon)) = 1. \tag{7}$$

(7) следует из $c < a$.

- $d \neq \varepsilon$. Обозначим $p \stackrel{\text{def}}{=} d_h$, $q \stackrel{\text{def}}{=} d_t$.

В этом случае надо доказать, что при $c < a$

$$\mathbf{ord}(c(a \to pq)) = 1. \tag{8}$$

1. Если $a \leq p$, то (8) имеет вид

$$\mathbf{ord}(c(a(pq))) = 1. \tag{9}$$

Поскольку $c < a \leq p$, то (9) следует из равенств

$$\begin{array}{l}\mathbf{ord}(c(a(pq))) = \mathbf{ord}(a(pq)) = \mathbf{ord}(pq) = \\ = \mathbf{ord}(c(pq)) = \mathbf{ord}(cd) = 1.\end{array}$$

2. Если $p < a$, то (8) имеет вид

$$\mathbf{ord}(cp(a \to q)) = 1. \tag{10}$$

Поскольку по предположению

$$\mathbf{ord}(cd) = \mathbf{ord}(cpq) = 1,$$

то $c \leq p$, и поэтому (10) можно переписать в виде

$$\mathbf{ord}(p(a \to q)) = 1. \tag{11}$$

При $p < a$

$$a \to d = a \to pq = p(a \to q),$$

поэтому (11) можно переписать в виде

$$\mathbf{ord}(a \to d) = 1. \tag{12}$$

(12) следует по индуктивному предположению для леммы (т.е. из импликации (5), в которой $z \stackrel{\text{def}}{=} d$) из равенства

$$\mathbf{ord}(d) = 1,$$

которое обосновывается цепочкой равенств

$$\begin{array}{ll}1 = \mathbf{ord}(cd) = \mathbf{ord}(cpq) = & \text{(т.к. } c \leq p\text{)} \\ = \mathbf{ord}(pq) = \mathbf{ord}(d). & \end{array}$$

# Межфакультетский курс «Методы искусственного интеллекта в задачах анализа данных и верификации программ» Лекция 5 Теория процессов

А.М.Миронов

## 1 Неформальное понятие процесса и примеры процессов

Прежде чем сформулировать формальное понятие процесса, мы опишем данное понятие неформально и рассмотрим простейшие примеры процессов.

### 1.1 Неформальное понятие процесса

Мы понимаем под процессом модель поведения динамической системы на некотором уровне абстракции.

**Процесс** можно представлять себе как граф $P$, компоненты которого имеют следующий смысл.

- Вершины графа $P$ называются **состояниями** и изображают ситуации (или классы ситуаций), в которых может находиться моделируемая система в различные моменты своего функционирования.

  Одно из состояний является выделенным, оно называется **начальным состоянием** процесса $P$.

- Рёбра графа $P$ имеют метки, изображающие **действия**, которые может исполнять моделируемая система.

- Функционирование процесса $P$ описывается переходами по рёбрам графа $P$ от одного состояния к другому. Функционирование начинается из начального состояния.

  Метка каждого ребра изображает действие процесса, исполняемое при переходе от состояния в начале ребра к состоянию в его конце.

## 1.2 Пример процесса

В качестве первого примера рассмотрим процесс, представляющий собой простейшую модель поведения некоторого торгового автомата.

Мы будем представлять себе этот автомат как машину, которая имеет монетоприемник, кнопку и лоток для выдачи товара. Когда покупатель хочет приобрести товар, он опускает монету в монетоприемник, нажимает на кнопку, и после этого в лотке появляется товар.

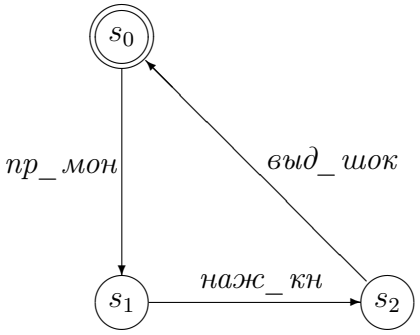
Предположим, что наш автомат торгует шоколадками по цене 1 монета за штуку. Опишем действия такого автомата.

- По инициативе покупателя в автомате могут происходить следующие действия: попадание в щель монеты и нажатие кнопки.
- В ответ автомат может осуществлять реакцию: выдавать в лоток шоколадку.

Обозначим действия короткими именами:

- прием монеты мы обозначим записью *пр_мон*,
- нажатие кнопки – записью *наж_кн*, и
- выдачу шоколадки – записью *выд_шок*.

Процесс нашего торгового автомата выглядит следующим образом:




(1)

Диаграмма (1) объясняет, как функционирует торговый автомат:

- вначале автомат находится в состоянии $s_0$, в этом состоянии он ожидает появления в приемнике монеты (то, что состояние $s_0$ является начальным, изображается на диаграмме двойным кружочком вокруг идентификатора этого состояния),
- когда монета появляется, автомат переходит в состояние $s_1$ и ждет нажатия на кнопку,
- после нажатия кнопки автомат переходит в состояние $s_2$, выдает шоколадку и возвращается в состояние $s_0$.

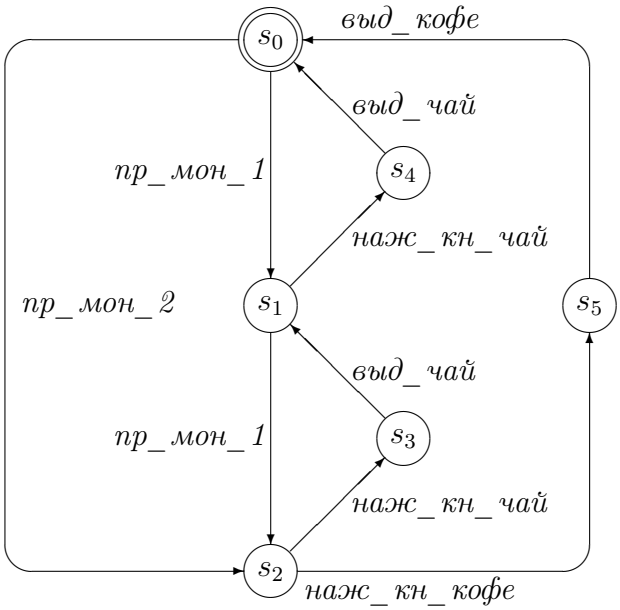
## 1.3 Другой пример процесса

Рассмотрим более сложный пример торгового автомата, который отличается от предыдущего тем, что продаёт два вида товаров: чай и кофе, причём стоимость чая – 1 рубль, а стоимость кофе – 2 рубля.

Автомат имеет две кнопки: одну – для чая, другую – для кофе.

Покупатель может платить монетами достоинством в 1 рубль и 2 рубля. Данные монеты будут обозначаться знакосочетаниями *мон_1* и *мон_2* соответственно. Если покупатель опустил в монетоприемник монету *мон_1*, он может купить только чай. Если же он опустил монету *мон_2*, он может купить кофе или два чая. Кофе можно купить также, опустив в монетоприёмник две монеты *мон_1*.

Процесс такого торгового автомата выглядит следующим образом:



Для формального определения понятия процесса мы должны уточнить понятие действия. Это уточнение излагается в параграфе 2.

# 2 Действия

Для задания процесса $P$, представляющего собой модель поведения некоторой динамической системы, должно быть указано множество $Act(P)$ **действий**, которые может выполнять процесс $P$. Будем предполагать, что действия всех процессов являются элементами некоторого универсального множества $Act$ всех возможных действий, которые может выполнить какой-либо процесс, т.е. для любого процесса $P$ $Act(P) \subseteq Act$.

Выбор множества $Act(P)$ действий процесса $P$ зависит от целей моделирования. В разных ситуациях для представления модели анализируемой системы в виде некоторого процесса могут выбираться разные множества действий.

Будем предполагать, что $Act$ делится на 3 следующих класса.

1. **Входные действия**, которые изображаются записями вида $?\alpha$. Действие вида $?\alpha$ понимается как прием объекта с именем $\alpha$.

2. **Выходные действия**, которые изображаются записями вида $!\alpha$. Действие вида $!\alpha$ понимается как посылка объекта с именем $\alpha$.

3. **Внутреннее** (или **невидимое**) действие, которое обозначается символом $\tau$. Внутренним мы называем такое действие процесса $P$, которое не связано с его взаимодействием с **окружающей средой** (т.е. с процессами, являющимися внешними по отношению к процессу $P$ и с которыми он может взаимодействовать). Например, внутреннее действие может быть связано со взаимодействием компонентов процесса $P$.

   В действительности внутренние действия могут быть самыми разнообразными, но для обозначения всех внутренних действий мы будем использовать один и тот же символ $\tau$. Это отражает наше желание не различать все внутренние действия, т.к. они не являются «наблюдаемыми» извне процесса $P$.

Обозначим знакосочетанием $Names$ совокупность имён объектов, которые могут приниматься или посылаться процессами. Множество $Names$ предполагается бесконечным.

Множество $Act$ представляет собой дизъюнкное объединение

$$Act = \{?\alpha \mid \alpha \in Names\} \sqcup \{!\alpha \mid \alpha \in Names\} \sqcup \{\tau\}.$$

Отметим, что объекты, которые принимаются и посылаются процессами, могут иметь самую различную природу (как материальную, так и нематериальную). Например, ими могут быть материальные ресурсы, люди, деньги, информация, энергия и т.д.

Кроме того, сами понятия приема и посылки могут иметь виртуальный характер, т.е. слова «прием» и «посылка» могут использоваться лишь как метафоры, а в действительности никакого приема или посылки какого-либо реального объекта может и не происходить.

Для каждого имени $\alpha \in Names$ действия $?\alpha$ и $!\alpha$ называются **комплементарными**. Будем использовать следующие обозначения:

- для каждого действия $a \in Act \setminus \{\tau\}$
  - $\bar{a}$ обозначает действие, комплементарное к $a$: $\overline{?\alpha} = !\alpha$, $\overline{!\alpha} = ?\alpha$,
  - $name(a)$ обозначает имя в $a$, т.е. $name(?\alpha) = name(!\alpha) = \alpha$,
- $\forall\, L \subseteq Act \setminus \{\tau\}\ \ \overline{L} \stackrel{\text{def}}{=} \{\overline{a} \mid a \in L\}$, $names(L) \stackrel{\text{def}}{=} \{name(a) \mid a \in L\}$.

# 3 Определение понятия процесса

**Процессом** (над множеством действий $Act$) называется тройка $P$ вида $(S, s^0, R)$, компоненты которой имеют следующий смысл:

- $S$ – множество, элементы которого называются **состояниями**,
- $s^0 \in S$ – состояние, называемое **начальным состоянием**,
- $R \subseteq S \times Act \times S$, элементы множества $R$ называются **переходами**, если переход из $R$ имеет вид $(s_1, a, s_2)$, то
  - будем говорить, что этот переход является переходом из состояния $s_1$ в состояние $s_2$ с выполнением действия $a$,
  - состояния $s_1$ и $s_2$ называются **началом** и **концом** этого перехода, а действие $a$ называется **меткой** этого перехода, и
  - будем обозначать данный переход записью $s_1 \stackrel{a}{\rightarrow} s_2$.

Совокупность всех процессов обозначается записью $Proc$.

Процесс $(S, s^0, R)$ можно представлять себе как граф с множеством вершин $S$, выделенной вершиной $s^0$, рёбра которого соответствуют переходам из $R$: если переход имеет вид $s_1 \stackrel{a}{\rightarrow} s_2$, то ему соответствует ребро с началом $s_1$, концом $s_2$ и меткой $a$.

**Функционирование** процесса $P = (S, s^0, R)$ заключается в порождении последовательности переходов вида $s^0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \dots$ и выполнении действий $a_0, a_1, a_2 \dots$, соответствующих этим переходам. Более подробно: на каждом шаге функционирования $i \geq 0$

- процесс находится в некотором состоянии $s_i$ $(s_0 = s^0)$,
- если есть хотя бы один переход из $R$ с началом в $s_i$, то $P$
  - недетерминированно выбирает переход с началом в $s_i$, помеченный таким действием $a_i$, которое можно выполнить в текущий момент времени (если таких переходов нет, то процесс временно приостанавливает свою работу до того момента, когда появится хотя бы один такой переход),
  - выполняет действие $a_i$, после чего переходит в состоние $s_{i+1}$, которое является концом выбранного перехода,
- если в $R$ нет переходов с началом в $s_i$, то $P$ заканчивает работу.

Ниже для каждого процесса $P$ запись $Act(P)$ будет обозначать множество невнутренних действий процесса $P$:

$$Act(P) = \{a \in Act \setminus \{\tau\} \mid \exists\, s \xrightarrow{a} s' \in R\}.$$

Процесс $(S, s^0, R)$ называется **конечным**, если $S$ – конечное множество. Конечный процесс можно изображать диаграммой, в которой

- каждому состоянию соответствует кружочек на плоскости, в котором м.б. написан идентификатор (имя этого состояния),
- каждому переходу соответствует стрелка из начала этого перехода в его конец, на которой написана метка этого перехода,
- начальное состояние обозначается двойным кружочком.

Состояние $s$ процесса $P = (S, s^0, R)$ называется

- **достижимым**, если $s = s^0$ или существует последовательность переходов в $P$, имеющая вид $s^0 \xrightarrow{a_1} \dots \xrightarrow{a_n} s$,
- **недостижимым**, если оно не является достижимым,
- **терминальным**, если не существует переходов с началом в $s$.

Будем считать процессы $P_1 = (S_1, s_1^0, R_1)$ и $P_2 = (S_2, s_2^0, R_2)$ равными, если они изоморфны как графы, т.е. существует биективное отображение $f : S_1 \to S_2$, такое, что $f(s_1^0) = s_2^0$, и для каждой пары вершин $s, s' \in S_1$ и каждого $a \in Act$ существует ребро $s \xrightarrow{a} s' \in R_1$ тогда и только тогда, когда существует ребро $f(s) \xrightarrow{a} f(s') \in R_2$.

Кроме того, будем считать процессы $P_1 = (S_1, s_1^0, R_1)$ и $P_2 = (S_2, s_2^0, R_2)$ равными, если $P_2$ получается из $P_1$ удалением недостижимых состояний и связанных с ними рёбер.

Процесс $(S, s^0, R)$ называется **детерминированным**, если $\forall\, s \in S$, $\forall\, a \in Act$ существует не более одного $s' \in S$, такого, что $s \xrightarrow{a} s' \in R$.

# 4 Операции на процессах

В этом пункте мы определим операции на процессах, при помощи которых из одних процессов можно строить другие, более сложные процессы.

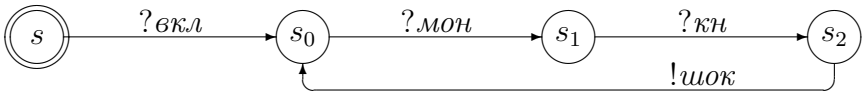
## 4.1 Префиксное действие

Первая такая операция – префиксное действие.

Пусть заданы процесс $P = (S, s^0, R)$ и действие $a \in Act$. Применением операции **префиксного действия** к паре $(a, P)$ является процесс, обозначаемый записью $a; P$ и имеющий вид $(S \sqcup \{s\}, s, R \sqcup \{s \xrightarrow{a} s^0\})$, т.е.

- множество состояний процесса $a; P$ получается добавлением к $S$ нового состояния $s$, которое будет начальным в $a; P$, и
- множество переходов процесса $a; P$ получается добавлением к $R$ нового перехода $s \xrightarrow{a} s^0$.

Проиллюстрируем действие данной операции на примере торгового автомата из параграфа 1.2. Обозначим процесс, представляющий поведение этого автомата, записью $P_{\text{та}}$. Расширим множество действий данного автомата новым действием $?вкл$, которое будет означать включение этого автомата в сеть. Процесс $?вкл; P_{\text{та}}$ представляет поведение нового торгового автомата, который в начальном состоянии не может ни принимать монет, ни воспринимать нажатия на кнопку, ни выдавать шоколадок. Единственное, что он может, – это стать включенным, после этого его поведение ничем не будет отличаться от поведения исходного автомата. Графовое представление процесса $?вкл; P_{\text{та}}$ имеет вид

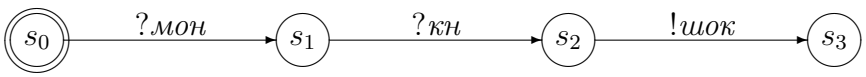


## 4.2 Пустой процесс

Среди всех процессов существует один наиболее простой. Этот процесс имеет всего одно состояние и не имеет переходов. Для обозначения такого процесса используется константа (т.е. нульарная операция) **0**.

Возвращаясь к примерам с торговыми автоматами, можно сказать, что процесс **0** представляет поведение сломанного автомата, то есть такого автомата, который вообще не может выполнять никаких действий.

Путем применения операций префиксного действия к процессу **0** можно определять поведение более сложных автоматов. Рассмотрим, например, процесс $P = ?мон; ?кн; !шок; \mathbf{0}$. Графовое представление этого процесса имеет вид




Этот процесс задает поведение автомата, который обслуживает ровно одного покупателя и после этого ломается.

## 4.3 Альтернативная композиция

Следующая операция на процессах – альтернативная композиция. Данная операция используется в том случае, когда по паре процессов $P_1, P_2$ требуется построить процесс $P$, который будет функционировать либо как процесс $P_1$, либо как процесс $P_2$, причём выбор процесса, в соответствии с которым $P$ будет функционировать, может определяться как самим $P$, так и окружающей средой, в которой функционирует $P$.

Например, если $P_1$ и $P_2$ имеют вид $P_1 = ?\alpha; P_1'$, $P_2 = ?\beta; P_2'$ и в начальный момент времени окружающая среда может предложить $P$ ввести объект $\alpha$, но не может предложить $P$ ввести объект $\beta$, то $P$ должен выбрать то поведение, которое является единственно возможным в данной ситуации, т.е. работать так же, как процесс $P_1$.

Отметим, что в данном случае выбирается такой процесс, первое действие в котором может быть выполнено в текущий момент времени. Выбрав $P_1$ и выполнив действие $?\alpha$, процесс $P$ обязан продолжать работу в соответствии со своим выбором, т.е. в соответствии с процессом $P_1'$.

Если же в начальный момент времени окружающая среда может предложить $P$ ввести как объект $\alpha$, так и объект $\beta$, то $P$ недетерминированно (т.е. произвольно) выбирает процесс, в соответствии с которым он будет работать, или же этот выбор производится с учётом дополнительных факторов.

Операция альтернативной композиции определяется следующим образом. Пусть процессы $P_1, P_2$ имеют вид $P_i = (S_i, s_i^0, R_i)$ $(i = 1, 2)$, причём множества состояний $S_1$ и $S_2$ не имеют общих элементов (если $S_1$ и $S_2$ имеют общие элементы, то для определения процесса $P_1 + P_2$ в качестве второго слагаемого надо взять не сам $P_2$, а его изоморфную копию, которая дизъюнктна с $P_1$).

**Альтернативная композиция** процессов $P_1$ и $P_2$ – это процесс

$$P_1 + P_2 = (S_1 \sqcup S_2 \sqcup \{s^0\}, s^0, R),$$

где $R = R_1 \sqcup R_2 \sqcup \{s^0 \xrightarrow{a} s \mid s_i^0 \xrightarrow{a} s \in R_i, i = 1, 2\}$ (т.е. $R$ получается добавлением к $R_1 \sqcup R_2$ переходов из начального состояния, дублирующих переходы из начальных состояний процессов-слагаемых).

Рассмотрим в качестве примера торговый автомат, который продаёт газированную воду, причём

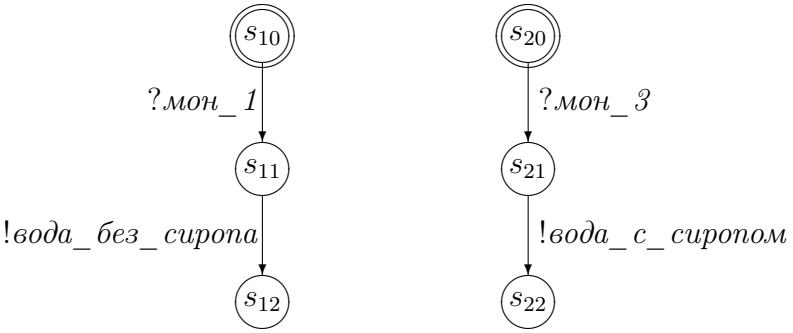
- если покупатель опускает в него монету *мон_ 1* достоинством в 1 копейку, то автомат выдаёт стакан воды без сиропа,
- а если покупатель опускает в него монету *мон_ 3* достоинством в 3 копейки, то автомат выдаёт стакан воды с сиропом

и сразу после продажи одного стакана воды автомат ломается.

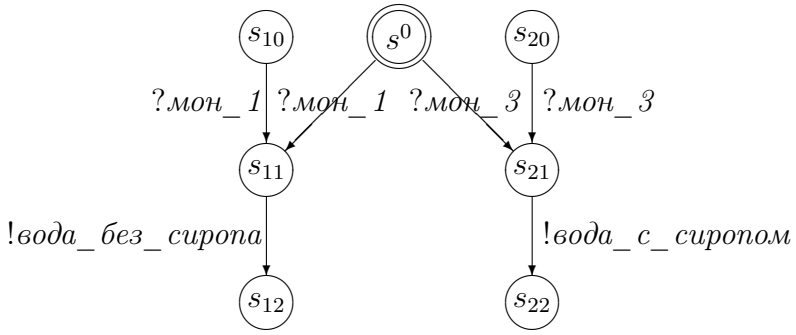
Поведение данного автомата описывается следующим процессом:

$$\begin{array}{rcl} P_{\text{газ\_вода}} & = & ?\textit{мон\_ 1}; !\textit{вода\_ без\_ сиропа}; \mathbf{0} \;\; + \\ & + & ?\textit{мон\_ 3}; !\textit{вода\_ с\_ сиропом}; \mathbf{0} \end{array} \qquad (2)$$

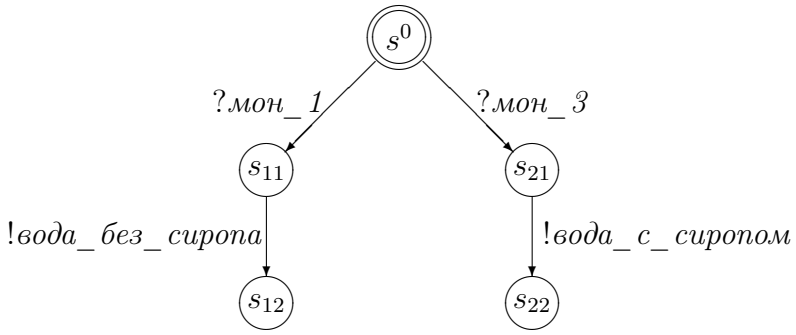
Графовые представления слагаемых в сумме (2) имеют вид




Согласно определению альтернативной композиции, графовое представление процесса (2) получается добавлением к предыдущей диаграмме нового состояния и переходов из этого состояния, дублирующих переходы из начальных состояний слагаемых, в результате чего получается следующая диаграмма:




Поскольку состояния $s_{10}$ и $s_{20}$ недостижимы, то их и связанные с ними переходы можно удалить, в результате чего получится диаграмма




которая и является искомым графовым представлением процесса (2).

Рассмотрим другой пример. Опишем разменный автомат, который может принимать купюры достоинством в 1000 рублей. Автомат должен выдать либо 2 купюры по 500 рублей, либо 10 купюр по 100 рублей, причем выбор способа размена осуществляется независимо от желания клиента. Сразу после одного сеанса размена автомат ломается.

$$P_{\text{размен}} = ?1\_no\_1000; (!2\_no\_500; \mathbf{0} + !10\_no\_100; \mathbf{0})$$

На этих двух примерах видно, что альтернативная композиция может использоваться для описания как минимум двух принципиально различных ситуаций.

1. Во-первых, она может выражать зависимость поведения системы от поведения её окружения.

   Например, в случае автомата $P_{\text{газ\_вода}}$ реакция автомата определяется действием покупателя, а именно достоинством монеты, которую он ввел в автомат. В данном случае процесс, представляющий

поведение моделируемого торгового автомата, является **детерминированным**, то есть его функционирование однозначно определяется входными действиями.

2. Во-вторых, на примере автомата $P_{\text{размен}}$ мы видим, что при одних и тех же входных действиях возможна различная реакция автомата. Это – пример **недетерминизма**, то есть неопределенности поведения системы. Неопределённость в поведении систем может происходить по крайней мере по двум причинам.

   (a) Во-первых, поведение систем может зависеть от **случайных факторов**. Такими факторами могут быть, например, сбои в аппаратуре, коллизии в компьютерной сети, отсутствие купюр необходимого достоинства в банкомате или что-либо еще.

   (b) Во-вторых, модель всегда есть некоторая абстракция или упрощение реальной системы. А при этом некоторые факторы, которые влияют на поведение этой системы, могут быть просто исключены из рассмотрения.

   В частности, на примере процесса $P_{\text{размен}}$ мы видим, что реальная причина выбора варианта поведения автомата может не учитываться в процессе, представляющем модель поведения этого автомата.

## 4.4 Параллельная композиция

Операция параллельной композиции используется для построения моделей поведения динамических систем, состоящих из взаимодействующих компонентов.

Если система состоит из двух компонентов, поведение которых описывается процессами $P_1$ и $P_2$, и функционирование системы заключается в совместном функционировании её компонентов, то поведение этой системы описывается процессом, который называется **параллельной композицией** процессов $P_1$ и $P_2$ и определяется следующим образом.

Пусть процессы $P_1$ и $P_2$ имеют вид $P_i = (S_i, s_i^0, R_i)$ $(i = 1, 2)$. **Параллельной композицией** процессов $P_1$ и $P_2$ называется процесс

$$P_1 \,|\, P_2 = (S_1 \times S_2, (s_1^0, s_2^0), R),$$

где $R$ содержит следующие переходы:

- переход $(s_1, s_2) \xrightarrow{a} (s_1', s_2)$, если $s_1 \xrightarrow{a} s_1' \in R_1$, $s_2 \in S_2$,
- переход $(s_1, s_2) \xrightarrow{a} (s_1, s_2')$, если $s_1 \in S_1$, $s_2 \xrightarrow{a} s_2' \in R_2$,

- переход $(s_1, s_2) \xrightarrow{\tau} (s_1', s_2')$, если $s_1 \xrightarrow{a} s_1' \in R_1$, $s_2 \xrightarrow{\overline{a}} s_2' \in R_2$.

Таким образом, каждое выполнение действия процессом $P_1|P_2$ представляет собой

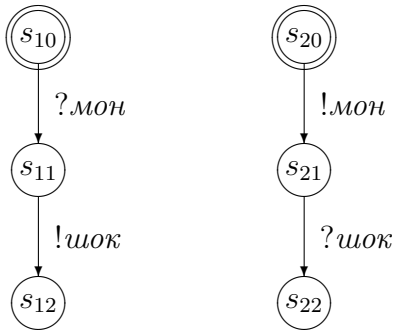
- либо одиночное выполнение действия одним из процессов $P_1, P_2$, во время выполнения которого другой из этих процессов приостанавливает свою работу,
- либо одновременное совместное выполнение действий обоими процессами $P_1, P_2$, которое заключается в том, что
  - один из этих процессов ($P_i$) передаёт другому процессу ($P_j$) некоторый объект и
  - процесс $P_j$ в тот же самый момент времени принимает от процесса $P_i$ этот объект,

  такой вид совместного выполнения называется **синхронным взаимодействием**.

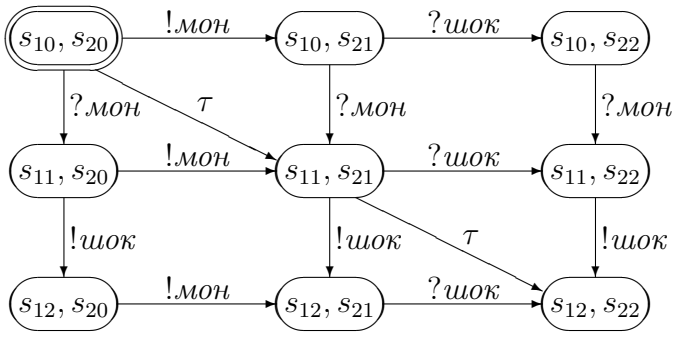
Приведём в качестве примера процесс $P_1 \,|\, P_2$, где процессы $P_1$ и $P_2$ представляют поведение торгового автомата и покупателя:

- $P_1 = ?\mathit{мон}; !\mathit{шок}; \mathbf{0}$, т.е. $P_1$ получает монету, выдаёт шоколадку и после этого ломается,
- $P_2 = !\mathit{мон}; ?\mathit{шок}; \mathbf{0}$, т.е. $P_2$ опускает монету, получает шоколадку и после этого завершает свою работу.

Графовые представления $P_1$ и $P_2$ имеют вид



Графовое представление процесса $P_1|P_2$ имеет вид




В данной диаграмме представлены все возможные варианты совместного функционирования $P_1$ и $P_2$. Каждое действие процесса $P_1|P_2$ м.б.

- синхронным взаимодействием: первое выполнение действия процессом $P_1|P_2$ м.б. результатом одновременного выполнения действий автоматом и покупателем (переход $(s_{10}, s_{20}) \xrightarrow{\tau} (s_{11}, s_{21})$: покупатель опускает в автомат монету, и автомат её принимает), или
- одиночным выполнением действия каким-либо из процессов $P_1, P_2$, например
  - переход $(s_{10}, s_{20}) \xrightarrow{!мон} (s_{10}, s_{21})$ интерпретируется следующим образом: покупатель опускает в автомат монету, но эта монета по каким-либо причинам не принимается автоматом, или
  - переход $(s_{11}, s_{21}) \xrightarrow{!шок} (s_{12}, s_{21})$ интерпретируется следующим образом: автомат выдал шоколадку, но покупатель по каким-либо причинам её не получил (например, к автомату подошёл вор и взял эту шоколадку, прежде чем её смог взять покупатель).

## 4.5 Ограничение

Пусть заданы процесс $P = (S, s^0, R)$ и подмножество $L \subseteq Names$.

**Ограничением** $P$ по $L$ называется процесс

$$P \setminus L = (S, s^0, \{s \xrightarrow{a} s' \mid a = \tau, \text{ или } name(a) \notin L\}),$$

т.е. $P \setminus L$ получается из $P$ удалением переходов, метки которых содержат имена из $L$.

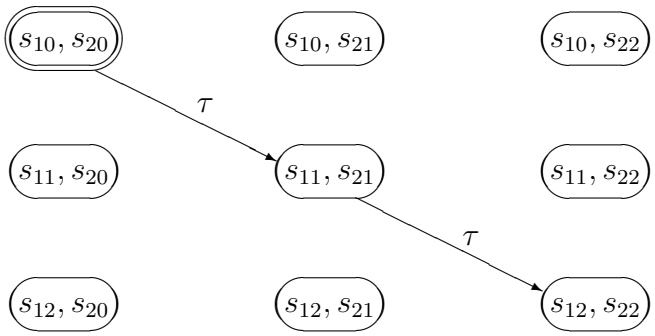
Операция ограничения используется, как правило, совместно с операцией параллельной композиции, для представления процессов, состоящих из нескольких компонентов, взаимодействие между которыми должно удовлетворять ограничениям. Эти ограничения заключаются в невозможности одиночного выполнения некоторых действий.

Например, пусть $P_1$ и $P_2$ – торговый автомат и покупатель, которые рассматривались в предыдущем параграфе. Мы хотели бы описать процесс, являющийся моделью такого параллельного функционирования процессов $P_1$ и $P_2$, при котором эти процессы могут совершать действия, связанные с покупкой-продажей шоколадки, только совместно.

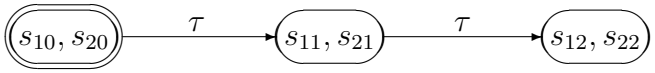
Искомый процесс имеет вид

$$P \stackrel{\text{def}}{=} (P_1|P_2) \setminus \{\textit{мон}, \textit{шок}\} \tag{3}$$

Графовое представление процесса (3) имеет вид



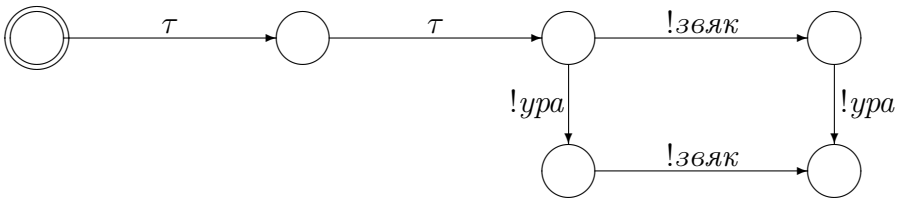

После удаления недостижимых состояний получится процесс, имеющий следующее графовое представление:




Рассмотрим другой пример. Немного изменим определения торгового автомата и покупателя: пусть они еще и сигнализируют об успешном выполнении своей работы, т.е. их процессы имеют следующий вид:

$$P_1 \stackrel{\text{def}}{=} ?\textit{мон}; !\textit{шок}; !\textit{звяк}; \mathbf{0}, \quad P_2 \stackrel{\text{def}}{=} !\textit{мон}; ?\textit{шок}; !\textit{ура}; \mathbf{0}$$

В этом случае графовое представление процесса (3) после удаления недостижимых состояний имеет вид




Данный процесс допускает одиночное выполнение тех невнутренних действий, которые не связаны с покупкой и продажей шоколадки.

### 4.6 Переименование

Пусть заданы процесс $P = (S, s^0, R)$ и функция $\theta : Names \to Names$.

**Переименованием** $P$ относительно $\theta$ называется процесс

$$P^\theta = (S, s^0, \{s \xrightarrow{a^\theta} s' \mid s \xrightarrow{a} s' \in R\}),$$

где $(!\alpha)^\theta = !\theta(\alpha)$, $(?\alpha)^\theta = ?\theta(\alpha)$, $\tau^\theta = \tau$, т.е. $P^\theta$ получается из $P$ заменой имён в метках переходов: каждое имя $\alpha$ заменяется на $\theta(\alpha)$.

Операция переименования позволяет многократно использовать один и тот же процесс $P$ в качестве компоненты при построении более сложного процесса $P'$. Эта операция используется для предотвращения «конфликтов» между именами действий, используемых в различных вхождениях $P$ в $P'$.

Если $\theta$ действует нетождественно лишь на имена $\alpha_1, \ldots, \alpha_n$ и отображает их в имена $\beta_1, \ldots, \beta_n$ соответственно, то для $P^\theta$ используется обозначение $P(\beta_1/\alpha_1, \ldots, \beta_n/\alpha_n)$.

## 5 Свойства операций на процессах

В этом параграфе мы приводим некоторые свойства определённых выше операций на процессах. В излагаемых ниже свойствах символы $P, L$ и $\theta$ (возможно, с индексами) изображают произвольные процессы, множества имен и переименования соответственно.

1. $(P_1 + P_2) + P_3 = P_1 + (P_2 + P_3)$.

   Из данного свойства следует, что допустимы выражения вида

$$P_1 + \ldots + P_n, \tag{4}$$

так как при любой расстановке скобок в выражении (4) получится один и тот же процесс. Данный процесс можно описать явно следущим образом. Пусть процессы $P_i$ $(i = 1, \ldots, n)$ имеют вид

$$P_i = (S_i, s_i^0, R_i) \quad (i = 1, \ldots, n), \tag{5}$$

причём множества $S_1, \ldots, S_n$ дизъюнктны. Тогда (4) имеет вид

$$(S_1 \sqcup \ldots \sqcup S_n \sqcup \{s^0\}, s^0, R),$$

где $\forall\, i = 1, \ldots, n$ $R$ содержит все переходы из $R_i$ и $\forall\, s_i^0 \xrightarrow{a} s \in R_i$ $R$ содержит переход $s^0 \xrightarrow{a} s$.

2. $(P_1 \mid P_2) \mid P_3 = P_1 \mid (P_2 \mid P_3)$.

   Из данного свойства следует, что допустимы выражения вида

   $$P_1 \mid \ldots \mid P_n, \tag{6}$$

   так как при любой расстановке скобок в выражении (6) получится один и тот же процесс. Данный процесс можно описать явно следущим образом. Пусть процессы $P_i$ $(i = 1, \ldots, n)$ имеют вид (5), тогда процесс (6) имеет вид

   $$P = (S_1 \times \ldots \times S_n, (s_1^0, \ldots, s_n^0), R),$$

   где $R$ содержит следующие переходы:

   - $(s_1, \ldots, s_n) \xrightarrow{a} (s_1, \ldots, s_{i-1}, s_i', s_{i+1}, \ldots, s_n)$, где $s_i \xrightarrow{a} s_i' \in R_i$,
   - $(s_1, \ldots, s_n) \xrightarrow{\tau} (s_1, \ldots, s_{i-1}, s_i', s_{i+1}, \ldots, s_{j-1}, s_j', s_{j+1}, \ldots, s_n)$, где для некоторого $a \in Act \setminus \{\tau\}$ $s_i \xrightarrow{a} s_i' \in R_i$, $s_j \xrightarrow{\bar{a}} s_j' \in R_j$.

3. $P_1 + P_2 = P_2 + P_1$.

4. $P_1 \mid P_2 = P_2 \mid P_1$.

5. $P \mid \mathbf{0} = P$.

6. $\mathbf{0} \setminus L = \mathbf{0}$.

7. $\mathbf{0}^\theta = \mathbf{0}$.

8. $P \setminus L = P$, если $L \cap names(Act(P)) = \emptyset$.

9. $(a; P) \setminus L = \begin{cases} \mathbf{0}, & \text{если } a \neq \tau \text{ и } name(a) \in L, \\ a; (P \setminus L) & \text{иначе.} \end{cases}$

10. $(P_1 + P_2) \setminus L = (P_1 \setminus L) + (P_2 \setminus L)$.

11. $(P_1 \mid P_2) \setminus L = (P_1 \setminus L) \mid (P_2 \setminus L)$, если

$$L \cap names(Act(P_1) \cap \overline{Act(P_2)}) = \emptyset.$$

12. $(P \setminus L_1) \setminus L_2 = P \setminus (L_1 \cup L_2)$.

13. $P^\theta \setminus L = (P \setminus \theta^{-1}(L))^\theta$.

14. $P^{\theta_{id}} = P$, где $\theta_{id}$ – тождественная функция.

15. $P^\theta = P^{\theta'}$, если $\forall\, \alpha \in names(Act(P))\ \ \theta(\alpha) = \theta'(\alpha)$.

16. $(a; P)^\theta = a^\theta; (P^\theta)$.

17. $(P_1 + P_2)^\theta = P_1^\theta + P_2^\theta$.

18. $(P_1 \mid P_2)^\theta = P_1^\theta \mid P_2^\theta$, если сужение $\theta$ на $names(Act(P_1) \cup Act(P_2))$ является инъективной функцией.

19. $(P \setminus L)^\theta = P^\theta \setminus \theta(L)$, если функция $\theta$ инъективна.

20. $P^{\theta\theta'} = P^(\theta' \circ \theta)$.

# 6 Эквивалентности процессов

В этой главе мы излагаем несколько определений понятия эквивалентности процессов. Выбор того или иного варианта определения эквивалентности процессов для его применения в конкретной ситуации должен определяться тем, как именно в данной ситуации понимается одинаковость процессов. В параграфах 6.1 и 6.2 вводятся понятия простой и сильной эквивалентности процессов. Данные понятия используются в тех ситуациях, когда все действия, выполняющиеся в процессах, имеют одинаковый статус. В параграфах **??** и **??** предлагаются другие варианты понятия эквивалентности процессов: наблюдаемая эквивалентность и наблюдаемая конгруэнция. Данные понятия используются в тех ситуациях, когда действие $\tau$ рассматривается как ненаблюдаемое.

С каждым из вариантов понятия эквивалентности процессов связаны естественные задачи: распознавание для двух заданных процессов, являются ли они эквивалентными и построение по заданному процессу $P$ процесса $P'$, имеющего минимальное число состояний среди всех процессов, которые эквивалентны $P$.

## 6.1 Простая эквивалентность

### 6.1.1 Поведение процесса

Будем использовать следующие обозначения: $\forall\, P = (S, s^0, R) \in Proc$

- $\forall\, s \in S$ запись $P^s$ обозначает процесс $(S, s, R)$ (т.е. $P^s$ отличается от $P$ только начальным состоянием),
- $\forall\, P' \in Proc$, $\forall\, a \in Act$ запись $P \xrightarrow{a} P'$ обозначает утверждение: в $P$ имеется ребро вида $s^0 \xrightarrow{a} s$ и $P' = P^s$.

Утверждение $P \xrightarrow{a} P'$ может интерпретироваться следующим образом: $P$ может выполнить действие $a$, после чего вести себя как $P'$.

Запись $P \xrightarrow{a} P'$ называется **переходом** из $P$.

**Поведением** процесса $P$ называется дерево $Beh(P)$,

- каждая вершина $v$ которого помечена некоторым процессом $P^v$,
- каждое ребро которого помечено некоторым действием из $Act$ и
- для каждой вершины $v \in Beh(P)$ множество ребер, выходящих из $v$, находится во взаимно однозначном соответствии с множеством переходов из $P^v$: ребру $v \xrightarrow{a} v'$ соответствует переход вида $P^v \xrightarrow{a} P^{v'}$.

### 6.1.2 Понятие простой эквивалентности

Будем говорить, что процессы $P, P' \in Proc$ находятся в отношении **простой эквивалентности**, если деревья $Beh(P)$ и $Beh(P')$ изоморфны, т.е. существует биективное отображение $f$ множества вершин дерева $Beh(P)$ на множество вершин дерева $Beh(P')$, такое, что $\forall\, v, v' \in Beh(P)$, $\forall\, a \in Act$ существует ребро $v \xrightarrow{a} v'$ тогда и только тогда, когда существует ребро $f(v) \xrightarrow{a} f(v')$.

$\forall\, P, P' \in Proc$ запись $P \equiv P'$ обозначает, что процессы $P$ и $P'$ находятся в отношении простой эквивалентности.

### 6.1.3 Примеры процессов, находящихся в отношении простой эквивалентности

В этом пункте мы приводим несколько примеров процессов, находящихся в отношении простой эквивалентности. Обоснование утверждений о простой эквивалентности этих процессов является несложным.

1. $P \equiv P'$, где $P$ и $P'$ имеют следующий вид:


2. Если графы процессов $P$ и $P'$ изоморфны или $P'$ получается из $P$ удалением недостижимых состояний, то $P \equiv P'$.

3. $\forall\, P \in Proc \;\; P + \mathbf{0} \equiv P$.

4. Если $P = (S, s^0, R)$ и список всех ребер, выходящих из $s^0$, имеет вид $\{s^0 \xrightarrow{a_i} s_i \mid i = 1, \ldots, n\}$, то $P \equiv a_1; P^{s_1} + \ldots + a_n; P^{s_n}$.

5. Пусть процесс $P$ имеет вид $P_1| \ldots |P_n$, где

$$\forall\, i = 1, \ldots, n \quad P_i \equiv a_{i1}; P_{i1} + \ldots + a_{in_i}; P_{in_i}. \tag{7}$$

   Тогда $P \equiv P'$, где $P'$ является суммой

   - всех процессов вида

$$a_{ij}; (P_1| \ldots |P_{i-1}|P_{ij}|P_{i+1}| \ldots |P_n),$$

     где $i = 1, \ldots, n,\; j = 1, \ldots, n_i$, и
   - всех процессов вида

$$\tau; (P_1| \ldots |P_{i-1}|P_{ik}|P_{i+1}| \ldots |P_{j-1}|P_{jl}|P_{j+1}| \ldots |P_n),$$

     где $1 \le i < j \le n, \quad a_{ik}, a_{jl} \neq \tau$, и $a_{ik} = \overline{a_{jl}}$.

6. **Теорема о разложении**: пусть $P$ имеет вид $(P_1| \ldots |P_n)\backslash L$, причем верно (7). Тогда $P \equiv P'$, где $P'$ является суммой

   - всех процессов вида

$$a_{ij}; \Big((P_1| \ldots |P_{i-1}|P_{ij}|P_{i+1}| \ldots |P_n) \setminus L\Big),$$

     где $i = 1, \ldots, n,\; j = 1, \ldots, n_i$, $a_{ij} = \tau$ или $name(a_{ij}) \notin L$, и
   - всех процессов вида

$$\tau; \Big((P_1| \ldots |P_{i-1}|P_{ik}|P_{i+1}| \ldots |P_{j-1}|P_{jl}|P_{j+1}| \ldots |P_n) \setminus L\Big),$$

     где $1 \le i < j \le n, \quad a_{ik}, a_{jl} \neq \tau$, и $a_{ik} = \overline{a_{jl}}$.

## 6.2 Сильная эквивалентность

### 6.2.1 Понятие сильной эквивалентности

Другим вариантом понятия эквивалентности процессов является **сильная эквивалентность**. Данное понятие основано на следующем понимании эквивалентности процессов: если мы рассматриваем процессы $P_1$ и $P_2$ как эквивалентные, то должно быть выполнено условие: если один из этих процессов $(P_i)$ может выполнить некоторое действие $a \in Act$ и после этого вести себя как некоторый процесс $P_i'$ (т.е. $P_i \xrightarrow{a} P_i'$), то другой процесс $(P_j)$ должен обладать способностью выполнить то же самое действие $a$, после чего вести себя как процесс $P_j'$ (т.е. $P_j \xrightarrow{a} P_j'$), который эквивалентен $P_i'$.

Обозначим символом $\mathcal{M}$ совокупность всех отношений $\mu$ на множестве $Proc$, удовлетворяющих условию: $\forall\,(P_1, P_2) \in \mu$

$$\begin{array}{l} \forall\, a \in Act, \forall\, P_1' : P_1 \xrightarrow{a} P_1' \;\; \exists\, P_2' : P_2 \xrightarrow{a} P_2' \text{ и } (P_1', P_2') \in \mu, \\ \forall\, a \in Act, \forall\, P_2' : P_2 \xrightarrow{a} P_2' \;\; \exists\, P_1' : P_1 \xrightarrow{a} P_1' \text{ и } (P_1', P_2') \in \mu. \end{array} \tag{8}$$

Естественно считать процессы $P_1, P_2$ эквивалентными, если существует хотя бы одно отношение $\mu \in \mathcal{M}$, содержащее пару $(P_1, P_2)$. Данное свойство можно выразить соотношением $(P_1, P_2) \in \bigcup_{\mu\in\mathcal{M}} \mu$.

Определим $\sim$ как отношение $\bigcup_{\mu\in\mathcal{M}} \mu$. Данное отношение называется **сильной эквивалентностью**. Нетрудно видеть, что $\sim \in \mathcal{M}$.

**Теорема 1**
$\sim$ является отношением эквивалентности.

**Доказательство**.

- $\sim$ рефлексивно, т.к. $Id_{Proc} = \{(P, P) \mid P \in Proc\} \in \mathcal{M}$,
- $\sim$ симметрично, т.к. из того, что $\forall\, \mu \in \mathcal{M} \;\; \mu^{-1} \in \mathcal{M}$, следует, что

$$\sim^{-1} = (\bigcup_{\mu\in\mathcal{M}} \mu)^{-1} = \bigcup_{\mu\in\mathcal{M}} (\mu^{-1}) \subseteq \bigcup_{\mu\in\mathcal{M}} \mu = \sim,$$

- $\sim$ транзитивно, т.к. если $\mu_1, \mu_2 \in \mathcal{M}$, то $\mu_1 \circ \mu_2 \in \mathcal{M}$, поэтому

$$\sim \circ \sim = (\bigcup_{\mu_1\in\mathcal{M}} \mu_1) \circ (\bigcup_{\mu_2\in\mathcal{M}} \mu_2) = \bigcup_{\mu_1,\mu_2\in\mathcal{M}} (\mu_1 \circ \mu_2) \subseteq \bigcup_{\mu\in\mathcal{M}} \mu = \sim . \;\blacksquare$$

Процессы $P_1, P_2 \in Proc$ называются **сильно эквивалентными**, если $(P_1, P_2) \in \sim$. Если процессы $P_1$ и $P_2$ сильно эквивалентны, то этот факт обозначается записью $P_1 \sim P_2$.

Нетрудно доказать, что

$$\text{если } P \equiv P', \text{ то } P \sim P'. \tag{9}$$

### 6.2.2 Критерий сильной эквивалентности, основанный на понятии бимоделирования

Пусть заданы два процесса: $P_i = (S_i, s_i^0, R_i)$ $(i = 1, 2)$. **Бимоделированием (БМ)** между $P_1$ и $P_2$ называется отношение $\rho \subseteq S_1 \times S_2$, содержащее пару $(s_1^0, s_2^0)$ и удовлетворяющее условию: $\forall\, (s_1, s_2) \in \rho$

$$\begin{array}{l} \forall\, a \in Act, \forall\, s_1' : s_1 \xrightarrow{a} s_1' \;\; \exists\, s_2' : s_2 \xrightarrow{a} s_2' \text{ и } (s_1', s_2') \in \rho, \\ \forall\, a \in Act, \forall\, s_2' : s_2 \xrightarrow{a} s_2' \;\; \exists\, s_1' : s_1 \xrightarrow{a} s_1' \text{ и } (s_1', s_2') \in \rho. \end{array} \tag{10}$$

**Теорема 2**
$P_1 \sim P_2$ тогда и только тогда, когда существует БМ между $P_1$ и $P_2$.

**Доказательство**.

1. Пусть $P_1 \sim P_2$. Определим отношение $\rho \subseteq S_1 \times S_2$:

$$\rho \stackrel{\text{def}}{=} \{(s_1, s_2) \in S_1 \times S_2 \mid P_1^{s_1} \sim P_2^{s_2}\}.$$

Докажем что $\rho$ является БМ между $P_1$ и $P_2$.

Так как $P_1 = P_1^{s_1^0}$ и $P_2 = P_2^{s_2^0}$, то из $P_1 \sim P_2$ следует, что $(s_1^0, s_2^0) \in \rho$.

Докажем первое свойство в (10): из $(s_1, s_2) \in \rho$ и $s_1 \xrightarrow{a} s_1'$ следует

$$\exists\, s_2' : s_2 \xrightarrow{a} s_2', (s_1', s_2') \in \rho. \tag{11}$$

Из $(s_1, s_2) \in \rho$ следует $P_1^{s_1} \sim P_2^{s_2}$. Из $s_1 \xrightarrow{a} s_1'$ следует $P_1^{s_1} \xrightarrow{a} P_1^{s_1'}$.

Так как $\sim \in \mathcal{M}$, то $\exists\, P_2' : P_2^{s_2} \xrightarrow{a} P_2'$ и $P_1^{s_1'} \sim P_2'$.

Согласно определению понятия перехода из процесса, из $P_2^{s_2} \xrightarrow{a} P_2'$ следует, что $\exists\, s_2' : s_2 \xrightarrow{a} s_2'$ и $P_2' = P_2^{s_2'}$.

Из $P_1^{s_1'} \sim P_2^{s_2'}$ следует, что $(s_1', s_2') \in \rho$. Таким образом, верно (11).

Аналогично доказывается второе свойство в (10).

2. Пусть существует БМ $\rho$ между $P_1$ и $P_2$. Докажем, что $P_1 \sim P_2$ путем построения отношения $\mu \in \mathcal{M}$, содержащего пару $(P_1, P_2)$.

   Определим $\mu \stackrel{\text{def}}{=} \{(P_1^{s_1}, P_2^{s_2}) \mid (s_1, s_2) \in \rho\}$.

   Так как $(s_1^0, s_2^0) \in \rho$, то $(P_1, P_2) = (P_1^{s_1^0}, P_2^{s_2^0}) \in \mu$.

   Для доказательства соотношения $\mu \in \mathcal{M}$ докажем первое свойство в (8): из соотношений $(P_1^{s_1}, P_2^{s_2}) \in \mu$ и $P_1^{s_1} \stackrel{a}{\to} P_1'$ следует, что

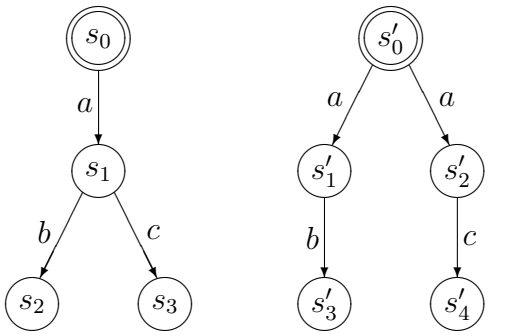
$$\exists\, P_2' : P_2^{s_2} \stackrel{a}{\to} P_2' \text{ и } (P_1', P_2') \in \mu. \tag{12}$$

   Так как $(s_1, s_2) \in \rho$ и из $P_1^{s_1} \stackrel{a}{\to} P_1'$ следует, что $\exists\, s_1' : s_1 \stackrel{a}{\to} s_1'$ и $P_1' = P_1^{s_1'}$, то по определению БМ $\exists\, s_2' : s_2 \stackrel{a}{\to} s_2'$, $(s_1', s_2') \in \rho$, поэтому в качестве $P_2'$ в (12) можно взять $P_2^{s_2'}$.

   Аналогично доказывается второе свойство в (8). ■

Используя теорему 2, можно обосновать следующие утверждения.

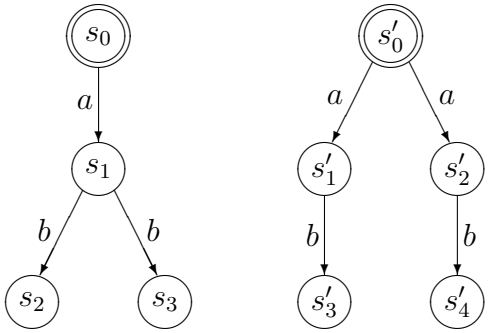
1. Процессы




(13)

   не являются сильно эквивалентными. Действительно, если они сильно эквивалентны, то по теореме 2 существует БМ $\rho$, содержащее пару $(s_0, s_0')$. Из $(s_0, s_0') \in \rho$ и $s_0' \stackrel{a}{\to} s_1'$ следует, что $\exists\, s : s_0 \stackrel{a}{\to} s$ и $(s, s_1') \in \rho$. Так как из $s_0$ выходит единственное ребро, то $s = s_1$. Из $(s_1, s_1') \in \rho$ и $s_1 \stackrel{c}{\to} s_3$ следует, что $\exists\, s : s_1' \stackrel{c}{\to} s$ и $(s_3, s) \in \rho$. Однако ребра вида $s_1' \stackrel{c}{\to} s$ не существует.

2. Процессы




сильно эквивалентны. БМ, обосновывающее это утверждение, имеет вид $\{(s_0, s'_0), (s_1, s'_1), (s_1, s'_2), (s_2, s'_3), (s_2, s'_4), (s_3, s'_3), (s_3, s'_4)\}$.

# Межфакультетский курс «Методы искусственного интеллекта в задачах анализа данных и верификации программ» Лекция 6 Теория процессов для конфиденциального обмена информацией

А.М.Миронов

## 1 Процессы для конфиденциального обмена информацией

Важными примерами процессов являются **протоколы для конфиденциального обмена информацией** (называемые также просто **протоколами**). В этом параграфе рассматриваются некоторые протоколы для конфиденциального обмена информацией.

Каждый протокол представляет собой параллельную композицию процессов, выполняющих формирование, отправление, приём и обработку сообщений. Такие процессы называются **агентами** протокола, а сообщения, пересылаемые от одного агента другому, называются **кадрами** (**frame**). Для обеспечения конфиденциальности информация в кадрах представляется в зашифрованном виде. Агенты пересылают свои сообщения друг другу через каналы. Кадры в каналах могут искажаться или пропадать (например, в результате воздействия радиопомех). Поэтому каждый кадр должен содержать не только ту информацию, которую один агент желает передать другому, но также и средства, позволяющие получателю этого кадра выяснить, был ли этот кадр искажён в процессе передачи.

## 1.1 Однонаправленный протокол с чередующимися битами

Первый из рассматриваемых нами протоколов передачи даннных состоит из двух агентов: **отправителя** (*Sender*) и **получателя** (*Receiver*). Задачей протокола является организация надёжной доставки кадров от отправителя к получателю через ненадёжный канал связи (который может искажать и терять передаваемые кадры). Данный протокол называется **однонаправленным протоколом с чередующимися битами**, в англоязычной литературе он называется **one-way Alternating Bit Protocol**, или, сокращённо, **one-way ABP**.

Работа протокола происходит следующим образом.

1. **Отправитель** получает сообщения от своего источника данных, эти сообщения называются **пакетами**. Задача отправителя заключается в периодическом выполнении следующих действий:

   - получить с порта $in$ очередной пакет от своего источника данных, сформировать из этого пакета кадр, послать этот кадр в канал $c$ и включить таймер действием $!start$,
   - если таймер пришлёт сигнал *timeout* (который означает, что время ожидания подтверждения посланного кадра закончилось и, видимо, этот кадр до получателя не дошёл), то послать этот кадр в канал $c$ ещё раз,
   - если придёт подтверждение от получателя (через канал $c'$), то это означает, что текущий кадр дошёл до него успешно, и нужно выключить таймер действием $!stop$, получить следующий пакет от своего источника данных и т.д.

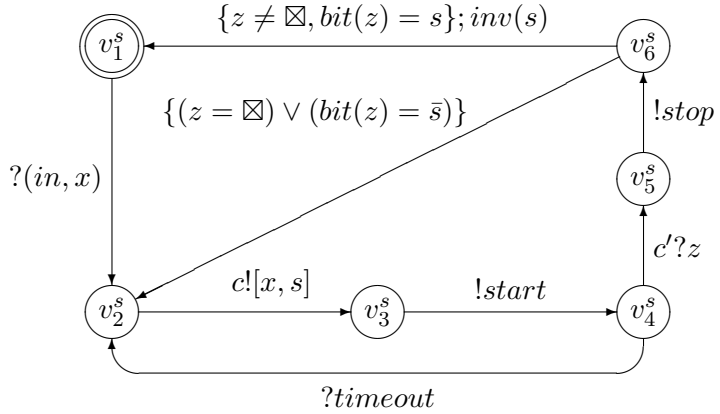
   Механизм, с помощью которого получатель может отличить новый кадр от кадра, переданного повторно, реализован в данном протоколе следующим образом: среди переменных отправителя и получателя присутствуют булевы переменные $s$ и $r$ соответственно, где $s$ – чётность номера очередного кадра, которого пытается послать отправитель, и $r$ – чётность номера очередного кадра, которого ожидает получатель. В начальный момент $s = r = 0$.

   Будем обозначать кадр, соответствующий пакету $x$, с которым связан бит $s$, записью $[x, s]$. Мы предполагаем, что информация в кадре $[x, s]$ является зашифрованной, но не указываем функцию шифрования явно. Для извлечения пакета и бита из кадра используются

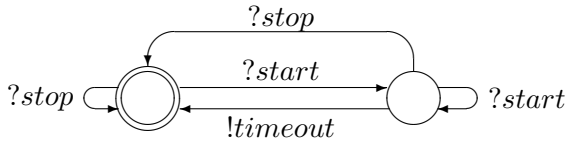
функции *info* и *bit* соответственно, обладающие свойствами:

$$info([x,s]) = x, \quad bit([x,s]) = s.$$

Работа отправителя представляется нижеследующим процессом $P$, в котором запись $inv(s)$ обозначает присваивание $s := 1-s$ и запись $\bar{s}$ обозначает терм $1-s$. Ниже записи вида $inv(x)$ и $\bar{x}$, где $x$ – булева переменная, имеют тот же смысл.




Поведение таймера, с которым взаимодействует $P$, представляется процессом $Timer$, он имеет вид

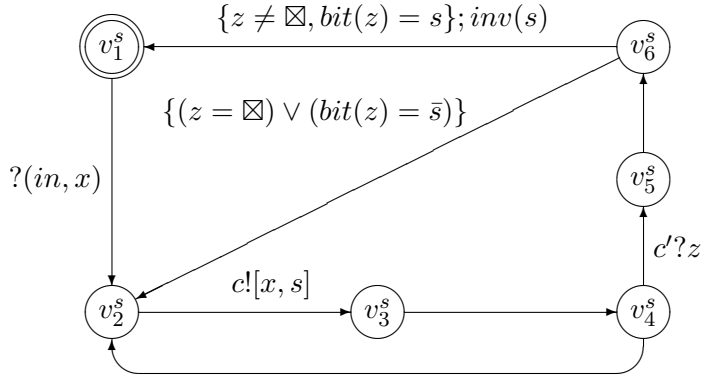



(1)

**Процесс отправителя** $Sender$ имеет вид

$$(P|Timer) \setminus \{start, stop, timeout\}.$$

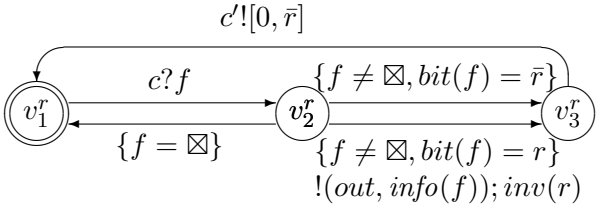
Нетрудно доказать, что $Sender$ наблюдаемо эквивалентен процессу



2. **Получатель** периодически выполняет следующие действия:

   - получение из канала $c$ очередного кадра $f$ и проверка наличия искажений в полученном кадре $f$,
   - если кадр $f$ искажён, то получатель его игнорирует (предполагая, что отправитель, не дождавшись подтверждения, пошлёт это кадр ещё раз),
   - если кадр $f$ не искажён, то извлечение из него бита $bit(f)$, и
     - если $bit(f)$ совпадает с ожидаемым битом $r$, то извлечение из этого кадра пакета $info(f)$ и передача этого пакета с порта *out* потребителю данных и инвертирование бита $r$,
     - посылка отправителю через канал $c'$ подтверждения полученного кадра (которое имеет вид $[0, \bar{r}]$).

Описанный выше алгоритм представляется процессом *Receiver*:

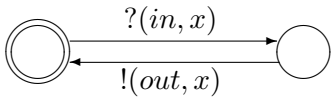



(2)

Поведение всего протокола описывается процессом

$$\begin{aligned} &Protocol \stackrel{\text{def}}{=} (Sender \mid Receiver), \\ &Init(Protocol) = \{s = r = 0, c = c' = \varepsilon\}. \end{aligned} \tag{3}$$

Спецификация процесса (3) заключается в том, что данный процесс наблюдаемо эквивалентен следующему процессу *Buffer*:




(4)

## 1.2 Двунаправленный протокол передачи сообщений с чередующимися битами

В большинстве ситуаций пересылки данных между двумя агентами требуется **двунаправленная передача**, т.е. передача кадров в обоих направлениях, где каждый из агентов выступает как в роли отправителя, так и в роли получателя. Работа агентов в таких ситуациях выглядит следующим образом: если агент $B$ успешно принял кадр $f$ от агента $A$,

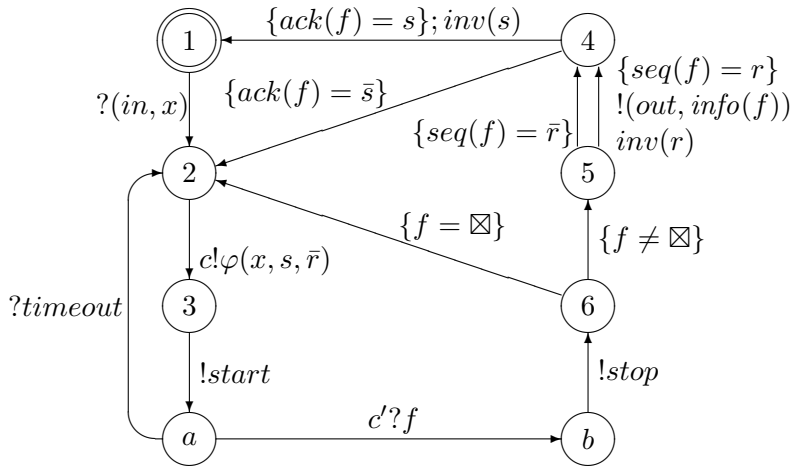
то он посылает подтверждение получения кадра $f$ в составе своего кадра, содержащего пакет для $A$.

Простейшим протоколом двунаправленной надежной передачи является **двунаправленный протокол с чередующимися битами** (two-way ABP). Поведение каждого из участников этого протокола описывается одним и тем же процессом $P$, совмещающим в себе функции отправителя и получателя. У каждого из участников имеется свой источник данных (от которого он получает пакеты, пересылаемые в составе кадров другому участнику) и свой потребитель данных (которому он передает пакеты, получаемые из кадров от другого участника).

Каждый пересылаемый кадр $f$ имеет вид $[x, s, r]$, где $x$ – пакет, $s$ – бит, сопоставленный пакету $x$, $r$ – бит подтверждения последнего полученного неискажённого кадра. Для извлечения пакетов и битов из кадров используются функции *info*, *seq* и *ack*, обладающие свойствами:

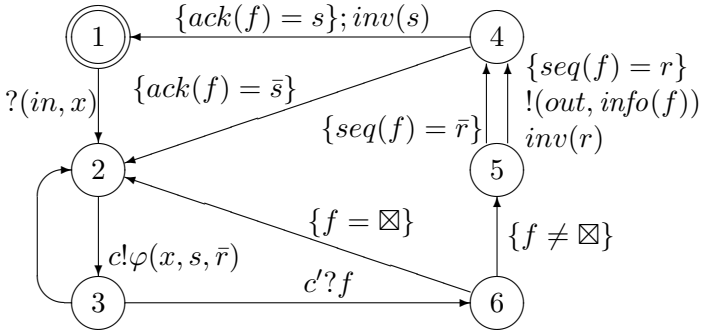
$$info([x, s, r]) = x, \quad seq([x, s, r]) = s, \quad ack([x, s, r]) = r.$$

Алгоритм работы участников данного протокола является объединением алгоритмов работы отправителя и получателя в протоколе из предыдущего пункта и представляется следующим процессом $P$:




Процесс *Agent*, описывающий поведение агента данного протокола, имеет вид $(P|Timer) \setminus \{start, stop, timeout\}$, где $Timer$ – процесс (1).

Нетрудно доказать, что *Agent* наблюдаемо эквивалентен процессу




(5)

Поведение всего протокола описывается процессом

$$\begin{aligned} &Protocol \stackrel{\text{def}}{=} \ (Agent_1 \,|\, Agent_2), \\ &Init(Protocol) = \{s_1 = 0, r_1 = 0, s_2 = 0, r_2 = 0, c_{12} = \varepsilon, c_{21} = \varepsilon\}, \end{aligned} \tag{6}$$

где $Agent_1$ и $Agent_2$ получаются из (5) приписыванием индекса 1 или 2 соответственно к каждому имени и к каждой переменной, не являющейся каналом, заменой в $Agent_1$ каналов $c$ и $c'$ на $c_{12}$ и $c_{21}$ соответственно и заменой в $Agent_2$ каналов $c$ и $c'$ на $c_{21}$ и $c_{12}$ соответственно.

# Межфакультетский курс «Методы искусственного интеллекта в задачах анализа данных и верификации программ» Лекция 7 Шифрование электронных платежей и model checking

А.М.Миронов

## 1 Введение

При обмене сообщениями в небезопасной среде используются методы шифрования. Шифрование сообщения $m$ – это преобразование $m$ в такое сообщение $k(m)$, где $k$ – ключ шифрования, по которому невозможно восстановить исходное сообщение $m$, не зная ключа дешифрования, соответствующего ключу шифрования $k$.

Алгоритмы обмена сообщениями с использованием шифрования называются **криптографическими протоколами (КП)**.

Мы будем рассматривать в этой лекции алгоритмы шифрования с **открытым ключом (ОК)**. Каждый открытый ключ связан с некоторым участником КП $A$, обозначается $A^+$ и используется как ключ для шифрования сообщений. Шифрованные сообщения $A^+(m)$ обозначаются более коротко как $A(m)$. Соответствующий ключ дешифрования обозначается $A^-$. Мы будем называть участников КП **агентами**.

## 2 Криптографические протоколы

**Криптографический протокол** (**КП**), называемый также просто **протоколом** – это распределённый алгоритм, определяющий порядок обмена сообщениями между несколькими **агентами**, в качестве которых

могут выступать, например, люди, компьютерные программы, вычислительные комплексы, базы данных, сети связи, банковские карточки, и т.д. Агенты, принимающие участие в работе КП, называются **участниками** этого КП.

Действия, выполняемые каждым из участников КП, могут иметь следующий вид:

- **посылка сообщения** другому участнику этого КП (или группе участников),
- **приём сообщения** от другого участника,
- **внутренние действия**, к числу которых относятся
  - выполнение участником некоторых вычислений,
  - проверка логических условий, и
  - обновление значений переменных.

КП предназначены для обеспечения безопасности передачи, обработки и хранения информации в небезопасной среде. Свойства безопасности, которые должен обеспечивать КП, могут иметь, например, следующий вид:

- **целостность** передаваемых сообщений, которая заключается в том, что всякое изменение сообщений в процессе их передачи будет обнаружено в ходе выполнения КП,
- **секретность** передаваемых сообщений, которая заключается в отсутствии неавторизованной утечки информации в процессе работы КП.

Некоторые из сообщений, пересылаемых участниками КП, могут быть зашифрованными. Шифрование сообщений делается для того, чтобы противник, которому станут доступны пересылаемые сообщения, не смог ознакомиться с их содержанием.

Шифрование сообщения $m$ представляет собой применение некоторого алгоритма $Encr$ (называемого **алгоритмом шифрования**) к паре $(k, m)$, где

- $k$ – битовая строка, называемая **ключом шифрования** (или просто **ключом**), и
- $m$ – шифруемое сообщение, называемое в данном случае **открытым текстом (ОТ)**.

Мы будем обозначать результат применения алгоритма $Encr$ к паре $(k, m)$ записью $k(m)$ и называть её **шифртекстом (ШТ)** сообщения $m$ на ключе $k$.

Для того, чтобы извлечь из ШТ $k(m)$ исходное сообщение $m$, должен быть задан алгоритм $Decr$, называемый **алгоритмом дешифрования**, и обладающий следующими свойствами:

- алгоритм $Decr$ получает на вход пару вида $(d, u)$, где
  - $d$ – битовая строка, называемая **ключом дешифрования**, и
  - $u$ – сообщение,

  результат применения алгоритма $Decr$ к паре $(d, u)$ будем обозначать записью $d(u)$,
- каждому ключу шифрования $k$ должен соответствовать ключ дешифрования $d_k$, такой, что для каждого сообщения $m$ верно равенство

$$d_k(k(m)) = m. \tag{1}$$

# 3 Системы шифрования

**Система шифрования (СШ)** представляет собой набор следующих данных:

- пара алгоритмов $Encr$ и $Decr$ шифрования и дешифрования соответственно, и
- набор ограничений, которым должны удовлетворять ключи шифрования и дешифрования, а также шифруемые сообщения.

Системы шифрования принято подразделять на два класса: симметричные и асимметричные.

1. В **симметричных** СШ **(ССШ)** каждый ключ шифрования $k$ совпадает с соответствующим ему ключом дешифрования $d_k$. Как правило, алгоритмы шифрования и дешифрования в ССШ тоже совпадают.
2. В **асимметричных** СШ **(АСШ)** ключи шифрования могут отличаться от соответствующих им ключей дешифрования, и

- ключи шифрования и дешифрования в АСШ, как правило, связаны с конкретными агентами, мы будем обозначать эти ключи записями $A^+$ и $A^-$ соответственно, где $A$ – идентификатор агента, с которым связаны эти ключи, и
- ключ $A^+$ является **открытым** (т.е. известен всем агентам), в то время как ключ $A^-$ **закрыт**: он не должен быть известен никому кроме агента $A$ (и, возможно, некоторым доверенным агентам).

Некоторые АСШ обладают следующим свойством: для каждого сообщения $m$ верно равенство

$$A^+A^-(m) = m. \tag{2}$$

Такие АСШ можно использовать для **аутентификации** агентов: если какой-либо агент $A$, использующий эту АСШ, хочет доказать свою подлинность (т.е. доказать, что он действительно является тем, за кого себя выдаёт), то он может сделать это путем предъявления пары $(m, m')$, где $m$ – произвольное сообщение, и $m' = A^-(m)$. Доказательство подлинности $A$ будет принято, если будет выполнено условие

$$A^+(m') = m. \tag{3}$$

Данное условие обосновывается предположением о том, что

- без знания закрытого ключа $A^-$ создать сообщение $m'$, удовлетворяющее условию (3), невозможно, и
- никто, кроме агента $A$, не должен знать ключ $A^-$.

# 4 Электронная подпись

Электронная подпись является одним из средств, предназначенных для доказательства подлинности передаваемых сообщений и их отправителей.

Протокол электронной подписи – это алгоритм, преобразующий пару $(m, A)$, где $m$ – сообщение и $A$ – имя агента, в строку,

- обозначаемую записью $\langle m \rangle_A^s$, и
- называемую **электронной подписью (ЭП)** сообщения $m$, созданной агентом $A$.

При вычислении $\langle m\rangle_A^s$ используется **закрытый ключ** $A^-$, который должен быть известен только агенту $A$.

Тройку $(m, a, \langle m\rangle_A^s)$ мы будем обозначать записью $\langle m\rangle_A$.

ЭП должна обладать следующими свойствами:

- **возможность проверки подлинности ЭП**: существует открытый алгоритм позволяющий по тройке $(m, A, u)$ проверить истинность равенства $u = \langle m\rangle_A^s$,
- **невозможность подделки ЭП**: задача вычисления $\langle m\rangle_A^s$ без знания $A^-$ является труднорешаемой,
- **невозможность подмены**: задача нахождения по $\langle m\rangle_A^s$ такого $m' \neq m$, что $\langle m'\rangle_A^s = \langle m\rangle_A^s$, является труднорешаемой.

В том случае, когда подписывающий агент $A$ и проверяющий агент $B$ могут использовать одну и ту же АШС, $\langle m\rangle_A^s$ может иметь один из следующих видов: $A^-(m)$, $B^+A^-(m)$, $A^-(a, A^-(m), t)$, где $T$ – метка времени, и т.п.

# 5 Хэш-функции

**Хэш-функция (ХФ)** - это функция $h$ вида

$$h : D \to \{0,1\}^n, \quad \text{где } D \subseteq \{0,1\}^*$$

(где $n$ – заданное число, $\{0,1\}^n$ – множество битовых последовательностей длины $n$, $\{0,1\}^*$ – множество конечных битовых последовательностей), алгоритм вычисления которой должен быть общеизвестен, и которая обладает следующими свойствами:

- $h$ – **односторонняя функция**, т.е. не существует быстрого алгоритма нахождения по заданному $y \in \{0,1\}^n$ такого $x \in D$, что $y = h(x)$,
- $h$ **устойчива к коллизиям**, т.е. сложно найти различные $x_1, x_2 \in D$, такие, что $h(x_1) = h(x_2)$.

ХФ применяются для контроля целостности данных, аутентификации источников данных, и многих других целей.

# 6 Формальные описания и примеры протоколов

## 6.1 Понятие формального описания протокола

Одним из простейших видов формального описания КП является список $P$ записей вида

$$A \to B : [\![\varphi]\!]\, m, \tag{4}$$

$$A \to \{B_1, \ldots, B_n\} : [\![\varphi]\!]\, m, \tag{5}$$

или

$$A : [\![\varphi]\!] \text{ внутреннее действие}, \tag{6}$$

где $A$, $B$, $B_1, \ldots, B_n$ – имена агентов, $\varphi$ – условие, $m$ – выражение, значением которого является сообщение (**сообщением** мы называем любой объект, который один из участников КП передаёт другому в процессе функционирования КП, этот объект может быть как символьной строкой, так и набором банкнот, товаром, и т.п.). Записи (4), (5) и (6) изображают действия, которые должны выполнять агенты в процессе функционирования КП.

Действия, входящие в список $P$, выполняются последовательно, их выполнение происходит следующим образом:

- (4) и (5) выполняется путем проверки $\varphi$, и
  - если $\varphi$ выполнено, то $A$ посылает сообщение $m$ агенту $B$ (в случае (4)), или агентам $B_1, \ldots, B_n$ (в случае (5)),
  - если же $\varphi$ не выполнено, то данное действие пропускается, и выполняется следующее по списку действие,
- (6) выполняется аналогично, только в данном случае происходит не посылка сообщения от $A$, а вычисления и изменение значений переменных агента $A$.

Если $\varphi$ выполнено всегда, то компонента $[\![\varphi]\!]$ в записи действий опускается.

Если текущее исполняемое действие не относится к какому-либо из участников КП, то этот участник не функционирует в момент исполнения этого действия.

# 7 Уязвимости протоколов

## 7.1 Понятие уязвимости протокола

Нарушения свойств безопасности в процессе работы КП могут происходить по причине противодействия со стороны агентов, называемых **противниками**.

Противники подразделяются на следующие два класса:

- **пассивные противники**, они могут перехватывать сообщения, пересылаемые участниками КП, и анализировать их,
- **активные противники**, они могут делать то же, что и пассивные противники, а также
  - модифицировать или удалять перехваченные сообщения,
  - генерировать новые сообщения и посылать их участникам КП,
  - выдавать себя за участников КП.

Кроме того, нарушения свойств безопасности КП возможны из-за действий участников КП, которые могут нарушать (умышленно или неумышленно) предписанные протоколом правила взаимодействия с другими участниками этого КП.

Возможность нарушения свойств безопасности в процессе работы КП называют **уязвимостями** этого КП.

При решении задач анализа уязвимостей КП используются следующие предположения о противнике:

- противник полностью знает КП,
- противнику доступны все сообщения, пересылаемые между участниками КП, он может их модифицировать, удалять, и заменять своими сообщениями,
- противник не может извлечь из перехваченных ШТ соответствующие ОТ, если он не знает необходимых ключей.

# 8 Пример уязвимости протокола

В этом пункте мы рассмотрим пример уязвимости КП с подтверждением приёма.

## 8.1 Протокол с подтверждением приёма

Рассмотрим следующую ситуацию: агенты $A$ и $B$ используют общую АСШ для секретного обмена сообщениями, и для контроля правильности передачи каждое получаемое сообщение отсылается назад отправителю (чтобы отправитель был уверен, что сообщение получено в неискажённом виде), согласно следующему протоколу:

$$\begin{array}{l} A \to B : B^+A^-(m) \\ B \to A : A^+B^-(m) \end{array} \tag{7}$$

(из полученного ШТ извлекается сообщение $m$ и возвращается отправителю в зашифрованном виде в качестве подтверждения приёма).

К сожалению, данный КП уязвим к следующей атаке: активный противник $M$ может перехватить первое сообщение и послать его $B$ (от своего имени), в результате чего будут выполнены следующие действия:

$$\begin{array}{l} M \to B : B^+A^-(m) \\ B \to M : M^+B^-M^+A^-(m) \end{array}$$

т.к. при взаимодействии с $M$ агент $B$ действует согласно протоколу (7): получив сообщение $m'$ $(= B^+A^-(m))$, он должен послать в ответ сообщение

$$M^+B^-M^+B^-(m') \ (= M^+B^-M^+A^-(m)). \tag{8}$$

Получив (8), $M$ может извлечь из него $m$.

Вывод: нужно проверять осмысленность получаемых сообщений.

# 9 Протокол электронных платежей

## 9.1 Понятие протокола электронных платежей

В протоколах электронных платежей, называемых также протоколами **электронной коммерции (ПЭК)** некоторые взаимодействия между участниками имеют коммерческий характер. Сообщения, передаваемые в таких взаимодействиях, являются электронными аналогами бумажных денег и называются **электронными банкнотами (ЭБ)**.

Свойства, которыми могут обладать ЭБ, могут иметь следующий вид.

1. По ЭБ невозможно вычислить тех участников, которые использовали её в своих платежах (анонимность).

2. ЭБ можно передавать другим участникам, которые могут использовать её по своему усмотрению.

3. ЭБ невозможно использовать более одного раза путём изготовления дубликата.

## 9.2 Пример ПЭК

В излагаемом ниже ПЭК $A$ – покупатель, $B$ – банк, $T, T'$ – продавцы.

Сначала $A$ получает от $B$ подписанную банком электронную банкноту (ЭБ) вида $(e, r, h(\alpha_1), h(\alpha_2))_B$, где $e$ – номинал банкноты, $\alpha_1$ и $\alpha_2$ – доли секрета: $id_A = \alpha_1 \oplus \alpha_2$, $h$ – ХФ.

Банк $B$ имеет базу данных $\mathcal{R}$, в которой он хранит вторые компоненты (которые уникальны) всех полученных им для депонирования ЭБ.

Протокол электронного платежа имеет следующий вид:

1. $A \to T : \langle e, r, h(\alpha_1), h(\alpha_2)\rangle_B$ (ЭБ для платежа)
2. $T \to A : b := (b_1, \ldots, b_k) \underset{r}{\in} \{1, 2\}^k$
3. $A \to T : \alpha_b := (\alpha_{1b_1}, \ldots, \alpha_{kb_k})$
   ($T$ принимает платёж, если среди $\{h(\alpha_{ib_i}) \mid i = 1, \ldots, k\}$ есть третья или четвёртая компонента полученной ЭБ)
4. $T \to B : (\langle e, r, h(\alpha_1), h(\alpha_2)\rangle_B, \alpha_b)$ (депонирование)
5. $B$ проверяет: $r \in \mathcal{R}$ ?
   - Если $r \notin \mathcal{R}$, то $\mathcal{R} := \mathcal{R} \cup \{r\}$, $B \to T$ : деньги
   - Если $r \in \mathcal{R}$, то $B$ сравнивает $\alpha_b$ в ЭБ и в своей БД.
     - Если они совпадают, то ЭБ скопировал $T$.
     - Если они не совпадают, то ЭБ скопировал $A$.
       Т.к. она уже использовалась при покупке у другого продавца $T'$, который послал $A$ список $b' \neq b$, то $\exists\, i$: одному из продавцов $T, T'$ $A$ послал $\alpha_{i1}$, а другому – $\alpha_{i2}$.
       $B$ получает $id_A = \alpha_{i1} \oplus \alpha_{i2}$.

# 10 Model checking

Протоколы могут анализироваться путем построения их математических моделей и описания их свойств на языке темпоральной логики.

Ниже в этой лекции мы будем называть протоколы **распределенными программами (РП)**.

## 10.1 Математические модели РП

При верификации РП методом model checking **математическая модель РП** (называемая ниже **системой переходов**) представляет собой граф,

- вершины которого называются **состояниями** и изображают ситуации (или классы ситуаций), в которых может находиться эта РП в различные моменты времени, и
- рёбра которого соответствуют переходам, по которым могут происходить изменения состояний во время выполнения этой РП.

## 10.2 Спецификация

**Спецификация РП** – это описание анализируемых свойств этой РП в виде формального текста, выраженное, например, в виде формулы математической логики.

Если свойство РП изначально было выражено на естественном языке, то при переводе его в спецификацию важно обеспечить соответствие между естественно-языковым описанием этого свойства и его спецификацией, т.к. в случае несоблюдения этого условия результаты верификации не будут иметь смысла.

Одним из инструментов формального описания свойств РП является **темпоральная логика**. Свойства РП выражаются в темпоральной логике **темпоральными формулами**.

Примеры свойств, выражаемых темпоральными формулами:

- РП при любом варианте своего функционирования не будет находиться ни в одном из состояний из заданного класса;
- РП при некотором функционировании когда-нибудь попадёт в некоторое состояние из заданного класса.

Как правило, при проведении рассуждений, связанных с анализом свойств РП, описываемых темпоральными формулами, рассматриваются не всевозможные темпоральные формулы, а только темпоральные формулы из некоторого ограниченного класса. Классы темпоральных формул принято называть **темпоральными логиками**, т.е. словосочетание «темпоральная логика» имеет два значения: в первом значении это язык, на котором можно выражать спецификации в виде темпоральных формул, а во втором – некоторый класс темпоральных формул.

В этом тексте будут рассмотрены следующие темпоральные логики: CTL (Computational Tree Logic), LTL (Linear Temporal Logic), PCTL (Probabilistic Computational Tree Logic).

## 10.3 Построение формальных доказательств

Если анализируемая модель РП удовлетворяет своей спецификации, то доказательство этого свойства методом model checking может быть построено либо путем рассмотрения всех возможных вариантов функционирования анализируемой РП, либо на основе дедуктивных рассуждений. Если же анализируемая модель РП не удовлетворяет своей спецификации, то для обоснования этого факта, как правило, достаточно построить хотя бы один вариант функционирования анализируемой модели (называемый **контрпримером**), при котором нарушается свойство, выражаемое спецификацией.

Главным достоинством метода model checking является возможность полностью автоматического анализа модели РП. Одним из главных недостатков model checking является высокая вычислительная сложность процедуры верификации на основе этого метода.

# 11 Системы переходов

Ниже предполагается, что задано множество $AP$, элементы которого называются **атомарными утверждениями** (**atomic propositions**).

## 11.1 Понятие системы переходов

**Система переходов (СП)** (которая также называется **моделью Крипке**) – это пятёрка $\Sigma = (S, R, L, S^0, \mathcal{F})$, компоненты которой имеют следующий смысл:

- $S$ – конечное множество **состояний**,
- $R \subseteq S \times S$ – **отношение перехода**,
- $L$ – функция (называемая **оценкой**) вида $L : S \times AP_\Sigma \to \{1, 0\}$, где $AP_\Sigma \subseteq AP$,
- $S^0 \subseteq S$ – множество **начальных состояний**,
- $\mathcal{F}$ – некоторая совокупность подмножеств множества $S$, называемых **условиями справедливости**.

Оценка $L$ имеет следующий смысл: $\forall\, s \in S,\ \forall\, p \in AP_\Sigma$ утверждение $p$ считается **истинным** в $s$, если $L(s, p) = 1$, и **ложным** в $s$ иначе.

Выражение $L(s, p)$ будем записывать более компактно в виде $p^s$.

СП $\Sigma = (S, R, L, S^0, \mathcal{F})$ удобно рассматривать как граф (обозначаемый тем же символом $\Sigma$) с множеством вершин $S$. $\forall\,(s, s') \in R$ данный граф содержит ребро из $s$ в $s'$.

Будем использовать следующие обозначения и соглашения:

- $\forall\, s \in S \;\; R(s) = \{s' \in S \mid (s, s') \in R\}, \;\; R^{-1}(s) = \{s' \in S \mid (s', s) \in R\}$;
- если $\Sigma$ – СП, компоненты которой не указаны, то будем обозначать её компоненты записями $S_\Sigma$, $R_\Sigma$, $L_\Sigma$, $S^0_\Sigma$, $\mathcal{F}_\Sigma$ соответственно;
- если СП $\Sigma$ задана в виде тройки $(S, R, L)$, то $S^0_\Sigma = S$, $\mathcal{F}_\Sigma = \{S\}$;
- если СП $\Sigma$ задана в виде четверки $(S, R, L, S^0)$, то $\mathcal{F}_\Sigma = \{S\}$.

## 11.2 Пути в системах переходов

**Путь** в СП $\Sigma$ – это последовательность $\pi = (s_0, s_1, \ldots)$ состояний из $S_\Sigma$, такая, что $\forall\, i \geq 0 \quad s_{i+1} \in R_\Sigma(s_i)$. Если данная последовательность бесконечна, то путь $\pi$ называется **бесконечным**, иначе путь $\pi$ называется **конечным**. Если путь $\pi$ конечный и имеет вид $(s_0, \ldots, s_n)$, то будем говорить, что $\pi$ – путь из $s_0$ в $s_n$. Этот путь называется **циклом**, если $s_n = s_0$, и **пустым**, если $n = 0$.

Если $\pi = (s_0, \ldots)$ и $S \subseteq S_\Sigma$, то $\pi \subseteq S$ означает, что $\forall\, i \geq 0\; s_i \in S$.

Если пути $\pi$ и $\pi'$ имеют вид $(s_0, \ldots, s_n)$ и $(s_n, s_{n+1}, \ldots)$ соответственно, то запись $\pi\pi'$ обозначает путь $(s_0, \ldots, s_n, s_{n+1}, \ldots)$, называемый **конкатенацией** $\pi$ и $\pi'$.

Если $\pi$ – цикл, то $\pi^\infty$ обозначает бесконечную конкатенацию $\pi\pi\pi\ldots$

Если $\pi$ имеет вид $(s_0, s_1, \ldots)$, то будем говорить, что $\pi$ – **путь из** $s_0$.

По умолчанию под путями будем подразумевать бесконечные пути, а если какой-либо из рассматриваемых путей является конечным, то это будет специально оговариваться.

Для каждого пути $\pi$ в СП $\Sigma$ запись $\inf(\pi)$ обозначает множество

$$\{s \in S_\Sigma \mid s \text{ имеет бесконечно много вхождений в } \pi\}.$$

Путь $\pi$ в СП $\Sigma$ называется **справедливым**, если

$$\forall\, F \in \mathcal{F}_\Sigma \;\; \inf(\pi) \cap F \neq \emptyset.$$

$\forall\, s \in S_\Sigma \;\; \Pi_s$ обозначает совокупность всех справедливых путей из $s$ в $\Sigma$, и $\Pi_\Sigma$ обозначает совокупность всех справедливых путей в $\Sigma$.

Запись $\Pi^0_\Sigma$ обозначает множество $\bigcup_{s \in S^0_\Sigma} \Pi_s$.

## 11.3 Построение системы переходов, соответствующей распределенной программе

**Распределённой программой (РП)** будем называть конечную совокупность процессов $P = (P_1, \ldots, P_k)$, где каждый процесс $P_i$ - это граф с множеством вершин $V_i$, ребра которого помечены присваиваниями $x := e$ или проверками условия $\{\beta\}$, где $\beta$ – терм, принимающий значение 1 или 0. Считаем, что $\forall\, i = 1, \ldots, k$ $P_i$ имеет начальное состояние $P_i^0$, множество рёбер $Edges_i$, и предусловие $Pre_i$. Будем использовать следующие обозначения:

- $\forall\, i = 1, \ldots, k$ $X_i$ – множество, состоящее из всех переменных, входящих в $P_i$, а также дополнительной переменной $at_i$, принимающей значения в множестве вершин процесса $P_i$;
- $X = \bigcup_{i=1}^{k} X_i$, $X' = \{x' \mid x \in X\} \subseteq Var$, где $X \cap X' = \emptyset$;
- для каждого множества переменных $X$ запись $X^\bullet$ обозначает множество всех означиваний переменных из $X$, т.е. множество всех функций вида $\theta : X \to D$, где $D$ – множество значений,
если $X = \{x_1, \ldots, x_n\}$, и $\theta$ сопоставляет каждому $x_i$ значение $d_i$, то $\theta$ может обозначаться записью
$$\begin{pmatrix} x_1 = d_1 \\ \ldots \\ x_n = d_n \end{pmatrix}$$
- $\forall\, (\theta, \theta') \in X^\bullet \times X^\bullet \quad \theta \sqcup \theta'$ – подстановка из $(X \sqcup X')^\bullet$, определяемая следующим образом:
$$\forall\, x \in X \;\; (\theta \sqcup \theta')(x) = \theta(x), \;\; (\theta \sqcup \theta')(x') = \theta'(x');$$
- $\forall\, Y \subseteq X$ запись $same(Y)$ обозначает формулу $\bigwedge_{x \in Y}(x' = x)$;
- $Edges = \bigcup_{i=1}^{k} Edges_i$;
- $\forall\, \gamma \in Edges$ запись $R_\gamma$ обозначает формулу, определяемую следующим образом: пусть $n$ и $n'$ – начало и конец $\gamma$ соответственно, тогда
  - если метка ребра $\gamma$ имеет вид $x := e$, то
$$R_\gamma = (at_i = n) \wedge (at_i' = n') \wedge (x' = e) \wedge same(X \setminus \{x, at_i\}),$$

– если метка ребра $\gamma$ имеет вид $\{\beta\}$, то

$$R_\gamma = (at_i = n) \wedge (at'_i = n') \wedge \beta \wedge same(X \setminus \{at_i\});$$

- символ $R$ обозначает формулу $\bigvee_{\gamma \in Edges} R_\gamma$;
- $Pre$ обозначает формулу $\bigwedge_{i=1}^{k} \Big( Pre_i \wedge (at_i = P_i^0) \Big)$.

Для любых $\theta, \theta' \in X^\bullet$:

- $Pre^\theta = 1$ тогда и только тогда, когда $\theta$ является одной из возможных подстановок из $X^\bullet$ в начальный момент выполнения $P$ и
- $R^{\theta \sqcup \theta'} = 1$ тогда и только тогда, когда верно утверждение:
  - если перед выполнением какого-либо действия РП $P$ каждая переменная $x \in X$ имела значение $\theta(x)$,
  - то после выполнения этого действия каждая переменная $x \in X$ будет иметь значение $\theta'(x)$.

СП $\Sigma$, соответствущая РП $P$, имеет следующие компоненты.

- $S_\Sigma \stackrel{\text{def}}{=} X^\bullet$, $S_\Sigma^0 \stackrel{\text{def}}{=} \{\theta \in X^\bullet \mid Pre^\theta = 1\}$.
- $R_\Sigma \stackrel{\text{def}}{=} \{(\theta, \theta') \in X^\bullet \times X^\bullet \mid R^{\theta \sqcup \theta'} = 1\}$.

  Из определения $R_\Sigma$ и приведенного выше замечания следует, что каждому выполнению $P$ соответствует некоторый путь в $\Sigma$.
- $AP_\Sigma \stackrel{\text{def}}{=} Fm(X)$, $\forall\, \theta \in S_\Sigma$, $\forall\, \beta \in Fm(X)$ $\;L_\Sigma(\theta, \beta) \stackrel{\text{def}}{=} \beta^\theta$.
- Выбор множества $\mathcal{F}_\Sigma$ условий справедливости индивидуален для каждой РП. На реальные выполнения РП P могут быть наложены дополнительные условия, и $\mathcal{F}_\Sigma$ должно быть выбрано с таким расчетом, чтобы множество $\Pi_\Sigma^0$ не содержало путей, не соответствующих никакому реальному выполнению (т.е. не удовлетворяющему этим дополнительным условиям) РП P.

  Такие дополнительные условия могут иметь следующий вид.
  - Не должно быть таких выполнений $P$, в которых один из входящих в неё процессов не совершает никаких действий после некоторого момента времени.

– Один из процессов в $P$ (обозначим его $P_i$) может обращаться с запросами к другому процессу (обозначим его $P_j$), и процесс $P_j$ может посылать процессу $P_i$ ответы на эти запросы.

  Не должно быть таких бесконечных выполнений $P$, в которых

  * одно из действий представляет собой запрос от $P_i$ к $P_j$ и
  * все последующие действия $P_j$ не являются посылкой ответа от $P_j$ к $P_i$ на этот запрос.

– Один из процессов в $P$ может посылать сообщения другому процессу, причём сообщения при пересылке могут пропадать.

  Не должно быть таких выполнений $P$, в которых один из процессов в $P$ бесконечно много раз посылает сообщения другому процессу в $P$ и все эти сообщения пропадают.

# 12 Model checking на основе CTL

В этом пункте изучается темпоральная логика CTL (Computational Tree Logic), используемая для формальной спецификации свойств РП.

Основными объектами всех темпоральных логик являются **темпоральные формулы** (называемые также просто **формулами**).

Будем считать, что для каждой из рассматриваемых в этом тексте темпоральных логик TL верно следующее:

- каждое утверждение из множества $AP$ атомарных утверждений, введённого в пункте 11, является формулой логики TL и
- TL замкнута относительно **булевых комбинаций**, т.е. среди формул логики TL есть константы $\top$ и $\bot$, и если TL содержит формулы $B$ и $C$, то TL также содержит формулы $\bar{B},\ B \wedge C,\ B \vee C$.

Запись вида $B \to C$, где $B, C \in$ TL, обозначает формулу $\bar{B} \vee C$.

# 13 Темпоральная логика CTL

## 13.1 Формулы темпоральной логики CTL

Совокупность **формул** темпоральной логики CTL, называемых также **CTL-формулами**, обозначается записью $Fm_{CTL}$. Как было отмечено выше, $Fm_{CTL}$ содержит $AP$ и замкнуто относительно булевых комбинаций. Кроме того, если $Fm_{CTL}$ содержит формулы $B$ и $C$, то $Fm_{CTL}$

содержит также формулы

$$\mathbf{AX}B,\ \ \mathbf{EX}B,\ \ \mathbf{AF}B,\ \ \mathbf{EF}B,\ \ \mathbf{AG}B,\ \ \mathbf{EG}B,\ \ \mathbf{AU}(B,C),\ \ \mathbf{EU}(B,C).$$

Жирные символы (**AX** и т.д.) в данных формулах называются **темпоральными операторами**.

## 13.2 Значения CTL-формул

Пусть задана СП $\Sigma$.

$\forall\, s \in S_\Sigma, \forall\, B \in Fm_{CTL}$ **значение** $B^s \in \{1,0\}$ формулы $B$ в состоянии $s$ определяется индуктивно:

- если $B = p \in AP$, то $B^s = \begin{cases} L_\Sigma(s,p), \text{ если } p \in AP_\Sigma, \\ 0, \text{ если } p \notin AP_\Sigma, \end{cases}$
- значения булевых комбинаций определяются стандартно:

$$\top^s = 1,\ \ \bot^s = 0,$$
$$(\bar{B})^s = \overline{B^s},\ \ (B \wedge C)^s = B^s \wedge C^s \text{ и т.д.}$$

- значения CTL-формул, начинающихся с темпорального оператора, определяются следующим образом:
  - $(\mathbf{AX}B)^s = [\![\, \forall\, s' \in R(s)\ \ B^{s'} = 1 \,]\!]$,
  - $(\mathbf{EX}B)^s = [\![\, \exists\, s' \in R(s) : B^{s'} = 1 \,]\!]$,
  - $(\mathbf{AF}B)^s = [\![\, \forall\, \pi = (s_0, \ldots) \in \Pi_s\ \exists\, i \geq 0 : B^{s_i} = 1 \,]\!]$,
  - $(\mathbf{EF}B)^s = [\![\, \exists\, \pi = (s_0, \ldots) \in \Pi_s,\ \exists\, i \geq 0 : B^{s_i} = 1 \,]\!]$,
  - $(\mathbf{AG}B)^s = [\![\, \forall\, \pi = (s_0, \ldots) \in \Pi_s\ \forall\, i \geq 0 : B^{s_i} = 1 \,]\!]$,
  - $(\mathbf{EG}B)^s = [\![\, \exists\, \pi = (s_0, \ldots) \in \Pi_s :\ \forall\, i \geq 0 : B^{s_i} = 1 \,]\!]$,
  - $(\mathbf{AU}(B,C))^s = [\![\, \forall\, \pi = (s_0, \ldots) \in \Pi_s\ \exists\, i \geq 0 : C^{s_i} = 1, \forall\, j < i \quad B^{s_j} = 1 \,]\!]$,
  - $(\mathbf{EU}(B,C))^s = [\![\, \exists\, \pi = (s_0, \ldots) \in \Pi_s, \exists\, i \geq 0 : C^{s_i} = 1, \forall\, j < i \quad B^{s_j} = 1 \,]\!]$.

**Значением** CTL-формулы $B$ в СП $\Sigma$ называется множество

$$B^{S_\Sigma} \stackrel{\text{def}}{=} \{s \in S_\Sigma \mid B^s = 1\}.$$

# Межфакультетский курс «Методы искусственного интеллекта в задачах анализа данных и верификации программ» Лекция 8 Прогнозирование в больших данных

А.М.Миронов

## 1 Задача прогнозирования временных рядов

В этой главе изучается задача прогнозирования временны́х рядов. Под **временны́м рядом** понимается последовательность $y = (y_1, y_2, \ldots)$ элементов некоторого множества $Y$, называемых **исходами**. В этой главе будем рассматривать случай, когда $Y = \{0, 1\}$ или $[0, 1]$ (вместо 0 м.б. $-1$).

Прогнозируемый временной ряд $y$ может быть бесконечным или иметь конечную длину, которую мы будем обозначать символом $T$, т.е. во втором случае $y = (y_1, y_2, \ldots, y_T)$.

Задача прогнозирования временного ряда $y$ заключается в построении **прогнозирующего алгоритма (ПА)** $A$, который на каждом шаге прогнозирования $t = 1, 2, \ldots$ выдаёт значение $\gamma_t \in Y$, называемое **прогнозом** временного ряда $y$ в момент $t$. После того, как $A$ выдал значение $\gamma_t$, становится известным значение $y_t \in Y$ **исхода** в момент $t$.

На каждом шаге прогнозирования $t$ определена **потеря** $l_t \in [0, 1]$ ПА $A$, связанная с несовпадением прогноза $\gamma_t$ и исхода $y_t$. Если $\gamma_t = y_t$, то $l_t = 0$. Как правило, потеря $l_t$ является значением некоторой **функции потерь** $\lambda$ на паре $(\gamma_t, y_t)$, например $l_t = [\![\gamma_t \neq y_t]\!]$, где для каждого логического утверждения $\varphi$ запись $[\![\varphi]\!]$ обозначает число 1, если $\varphi$ истинно, и 0, если $\varphi$ ложно.

Если $y = (y_1, y_2, \ldots, y_T)$, то будем называть **кумулятивной потерей** ПА $A$ величину $L_T \stackrel{\text{def}}{=} \sum_{t=1}^{T} l_t$.

## 2 Смешивающее прогнозирование

Один из методов построения прогнозирующих алгоритмов заключается в следующем. Пусть имеется несколько ПА $A_1$, ..., $A_N$. Для каждого $i = 1, \ldots, N$ в каждый момент времени $t = 1, \ldots, T$ ПА $A_i$ выдаёт прогноз $\gamma_t^i$. Будем обозначать записью $L_T^i$ кумулятивную потерю ПА $A_i$. Используя алгоритмы $A_1, \ldots, A_N$, можно построить ПА $A$, называемый **смешивающим** ПА. На каждом шаге прогнозирования $t$ прогноз $\gamma_t$, выдаваемый алгоритмом $A$, определяется как некоторая функция от прогнозов $\gamma_t^1, \ldots, \gamma_t^N$, называемая **функцией смешивания**.

ПА $A_1, \ldots, A_N$, участвующие в построении смешивающего ПА $A$, будем называть **экспертами**. Экспертов можно сравнивать по качеству их прогнозов: эксперт $A_i$ лучше эксперта $A_j$, если $L_T^i < L_T^j$. Будем обозначать экспертов просто их номерами $1, \ldots, N$, и множество экспертов $\{1, \ldots, N\}$ будем обозначать символом $I$.

Величина $R_T = L_T - \min_{i \in I} L_T^i$ называется **регретом** смешивающего ПА $A$. Данная величина выражает собой отличие кумулятивной потери ПА $A$ от кумулятивной потери наилучшего эксперта. При построении смешивающих ПА функция смешивания должна выбираться так, чтобы регрет смешивающего ПА был как можно меньше.

Ниже рассматриваются некоторые смешивающие ПА.

## 3 Алгоритм большинства

В этом пункте излагается смешивающий ПА, в предположении что среди экспертов $1, \ldots, N$ существует эксперт $i_0$, в каждый момент времени выдающий правильный прогноз, т.е. такой, что $\forall\, t \geq 1 \;\; \gamma_t^{i_0} = y_t$. Этот алгоритм называется **алгоритмом большинства (Majority Algorithm, MA)**. Прогнозы данного ПА определяются следующим образом:

$$\forall\, t \geq 1 \quad \gamma_t := [\![ |\{i \in B_t \mid \gamma_t^i = 1\}| \geq \tfrac{|B_t|}{2} ]\!], \tag{1}$$

где $\forall\, t \geq 1$ множество $B_t \subseteq \{1, \ldots, N\}$ определяется следующим образом:

$$B_t = \{i \in \{1, \ldots, N\} \mid \forall\, t' = 1, \ldots, t-1 \;\; \gamma_{t'}^i = y_{t'}\}. \tag{2}$$

Будем говорить, что ПА $A$ **делает ошибку** на шаге $t$, если $\gamma_t \neq y_t$.

**Теорема 4.1**
MA делает не более $\log_2 N$ ошибок.

**Доказательство**.

Нетрудно видеть, что последовательность множеств (2) обладает свойством $B_1 \supseteq B_2 \supseteq \ldots$, и каждое из множеств $B_t$ непусто, т.к. $i_0 \in B_t$.

Если МА делает ошибку на шаге $t$, т.е. $\gamma_t \neq y_t$, то

- либо $\gamma_t = 1$ и $y_t = 0$, в этом случае, согласно (1),

$$|\{i \in B_t \mid \gamma_t^i = 1\}| \geq \tfrac{|B_t|}{2}, \tag{3}$$

  согласно (2),

$$B_{t+1} = \{i \in B_t \mid \gamma_t^i = 0\}, \tag{4}$$

  и из (3) и (4) следует, что $|B_{t+1}| \leq \frac{|B_t|}{2}$,

- либо $\gamma_t = 0$ и $y_t = 1$, в этом случае, согласно (1),

$$|\{i \in B_t \mid \gamma_t^i = 1\}| < \tfrac{|B_t|}{2}, \tag{5}$$

  согласно (2),

$$B_{t+1} = \{i \in B_t \mid \gamma_t^i = 1\}, \tag{6}$$

  и из (5) и (6) следует, что $|B_{t+1}| < \frac{|B_t|}{2}$.

В обоих случаях верно неравенство $|B_{t+1}| \leq \frac{|B_t|}{2}$.

Пусть МА делает $k$ ошибок, и $t$ – момент, в который делается $k$–я ошибка, тогда, по установленному выше,

$$N \geq |B_1| \geq 2^k |B_{t+1}|. \tag{7}$$

Учитывая неравенство $|B_{t+1}| \geq 1$, из (7) получаем неравенство $N \geq 2^k$, или $k \leq \log_2 N$. ■

# 4 Алгоритм взвешенного большинства

В этом пункте излагается смешивающий алгоритм, называемый **алгоритмом взвешенного большинства (Weighted Majority Algorithm, WMA)**. Данный алгоритм может использоваться в том случае, когда среди экспертов $1, \ldots, N$ может не быть эксперта, выдающего в каждый момент времени правильный прогноз.

В каждый момент времени $t$ данный алгоритм сопоставляет каждому эксперту $i$ некоторое число $w_t^i \in [0, 1]$, называемое **весом** этого эксперта. В начальный момент $t = 1$ вес каждого эксперта равен 1. Считаем, что потери имеют вид $l_t = |\gamma_t - y_t|$, $l_t^i = |\gamma_t^i - y_t|$.

Прогнозы данного ПА и изменения весов определяются следующим образом: выбирается параметр $\varepsilon \in (0,1)$, и

$$\forall\, t = 1, \ldots, T \left\{ \begin{array}{l} \gamma_t := [\![ \sum_{i:\gamma_t^i=0} w_t^i \le \sum_{i:\gamma_t^i=1} w_t^i ]\!] \\ w_{t+1}^i := w_t^i (1 - \varepsilon l_t^i) \end{array} \right. \tag{8}$$

**Теорема 4.2**
Для кумулятивных потерь WMA верно неравенство

$$L_T \le \tfrac{2}{1-\varepsilon} \min_{i \in I} L_T^i + \tfrac{2}{\varepsilon} \ln N \tag{9}$$

(т.е. WMA ошибается примерно не более чем в $\frac{2}{1-\varepsilon}$ раз, чем наилучший эксперт).

**Доказательство**.
Будем использовать следующие обозначения:

$$M = L_T, \;\; m = \min_{i \in I} L_T^i, \;\; |\vec{w}_t| = \sum_{i \in I} w_t^i.$$

Пусть $i$ – номер наилучшего эксперта.
$w_t^i$ корректируется $\le m$ раз, поэтому

$$|\vec{w}_T| \ge w_T^i \ge (1-\varepsilon)^m. \tag{10}$$

Нетрудно проверить (это делается так же, как в доказательстве предыдущей теоремы), что если WMA делает ошибку на шаге $t$, то

$$\sum_{i:\gamma_t^i \ne y_t} w_t^i \ge \sum_{i:\gamma_t^i = y_t} w_t^i. \tag{11}$$

Прибавив к обеим частям (11) $\sum_{i:\gamma_t^i \ne y_t} w_t^i$, получаем

$$\sum_{i:\gamma_t^i \ne y_t} w_t^i \ge \tfrac{|\vec{w}_t|}{2}. \tag{12}$$

Из (12) и из определения $w_{t+1}^i$ в (8) следует, что если WMA делает ошибку на шаге $t$, то

$$\begin{array}{l} |\vec{w}_{t+1}| = \sum_{i \in I} w_t^i (1 - \varepsilon l_t^i) = \\ = |\vec{w}_t| - \varepsilon \sum_{i:\gamma_t^i \ne y_t} w_t^i \le |\vec{w}_t| (1 - \frac{\varepsilon}{2}), \end{array}$$

т.е. если WMA делает ошибку на шаге $t$, то $\frac{|\vec{w}_{t+1}|}{|\vec{w}_t|} \le \frac{\varepsilon}{2}$.

Из определения весов в (8) следует, что для каждого $t = 1, \ldots, T-1$ $\frac{|\vec{w}_{t+1}|}{|\vec{w}_t|} \le 1$. Следовательно,

$$\tfrac{|\vec{w}_T|}{|\vec{w}_1|} = \prod_{t=1}^{T-1} \tfrac{|\vec{w}_{t+1}|}{|\vec{w}_t|} \le (1 - \tfrac{\varepsilon}{2})^M,$$

откуда, учитывая (10) и равенство $|\vec{w}_1| = N$, получаем:

$$\tfrac{(1-\varepsilon)^m}{N} \le (1 - \tfrac{\varepsilon}{2})^M,$$

логарифмируя которое, и учитывая неравенство

$$\ln(1+x) \le x \quad \text{при } x \in (-1, 1) \tag{13}$$

получаем:

$$m \ln(1-\varepsilon) - \ln N \le M \ln(1 - \tfrac{\varepsilon}{2}) \le -\tfrac{\varepsilon}{2} M$$

откуда следует неравенство

$$\tfrac{\varepsilon}{2} M \le m \ln \tfrac{1}{1-\varepsilon} + \ln N. \tag{14}$$

Применяя (13) для $x = \frac{\varepsilon}{1-\varepsilon}$, получаем соотношения

$$\ln \tfrac{1}{1-\varepsilon} = \ln(1 + \tfrac{\varepsilon}{1-\varepsilon}) \le \tfrac{\varepsilon}{1-\varepsilon} \tag{15}$$

Из (14) и (15) следует неравенство

$$\tfrac{\varepsilon}{2} M \le m \tfrac{\varepsilon}{1-\varepsilon} + \ln N,$$

которое эквивалентно доказываемому неравенству (9). ■

# Межфакультетский курс «Методы искусственного интеллекта в задачах анализа данных и верификации программ» Лекция 9 Верификация алгоритмов с бинарными и численными потерями

А.М.Миронов

## 1 Алгоритм машинного обучения с бинарными потерями

В этом параграфе рассматривается задача построения алгоритма машинного обучения, в которой

- множества объектов и ответов имеют вид $X = \mathbf{R}^n$, $Y = \{-1, 1\}$, и
- задача обучения заключается в нахождении по выборке $S$ классификатора $a_S$ вида

$$a_S(x) = sign\Big(\sum_{i=1}^{n} x^i w_i - w_0\Big), \quad \text{где } w_1, \ldots, w_n, w_0 \in \mathbf{R}, \qquad (1)$$

  которая минимизирует риск $Q(a_S)$.

Потери данного алгоритма имеют бинарный характер.

### 1.1 Описание алгоритма обучения Розенблатта

Пусть выборка $S$ строго линейно разделима, т.е. $\exists\, w_1, \ldots, w_n, w_0 \in \mathbf{R}$:

$$\forall\, (x, y) \in S \quad \Big(\sum_{i=1}^{n} x^i w_i - w_0\Big) y > 0, \quad \text{где } (x^1, \ldots, x^n) = x. \qquad (2)$$

Нетрудно видеть, что неравенство в (2) равносильно неравенству

$$\langle (x^1, \ldots, x^n, -1), (w_1, \ldots, w_n, w_0) \rangle y > 0,$$

поэтому задача нахождения для строго разделимой выборки $S$ такого вектора $(w_1, \ldots, w_n, w_0)$, который удовлетворяет условию (2), сводится к задаче нахождения вектора $w \in \mathbf{R}^{n+1}$, такого, что

$$\forall\, (x, y) \in S \quad \langle (x, -1), w \rangle y > 0,$$

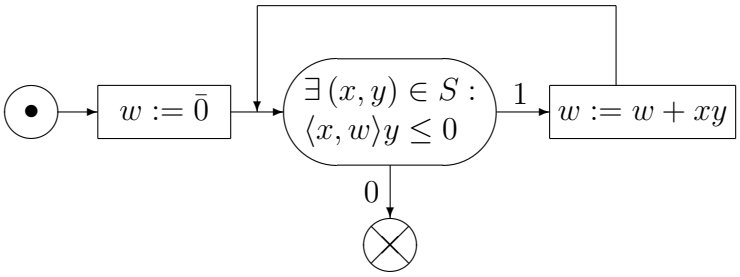
где $(x, -1) \in \mathbf{R}^{n+1}$ получается из $x$ добавлением компоненты, равной $-1$.

Таким образом, задача нахождения для строго разделимой выборки классификатора $a_S$ со свойством $Q(a_S) = 0$ сводится к следующей задаче: пусть выборка $S$ такова, что существует вектор $w$, обладающий свойством

$$\forall\, (x, y) \in S \quad \langle x, w \rangle y > 0. \tag{3}$$

Требуется найти хотя бы один вектор $w$, обладающий свойством (3).

Для поиска такого вектора $w$ можно использовать излагаемый ниже алгоритм, называемый **алгоритмом Розенблатта**. Данный алгоритм можно представить в виде следующей блок-схемы:




(4)

где $\bar{0}$ – вектор, компоненты которого равны 0, и $x, y$ в правом прямоугольнике – компоненты какой-либо пары $(x, y) \in S$, такой, что $\langle x, w \rangle y \leq 0$.

Нетрудно видеть, что после завершения работы данного алгоритма значение переменной $w$ будет обладать свойством (3).

## 1.2 Завершаемость алгоритма Розенблатта

Завершаемость алгоритма Розенблатта обосновывается нижеследующей теоремой, которая называется **теоремой Новикова**.

**Теорема 1 (A.Novikoff, 1962)** [?], [?].

Пусть конечная выборка $S \subset \mathbf{R}^n \times \{-1, 1\}$ такова, что свойство (3) выполняется для некоторого вектора $\hat{w}$, т.е.

$$\forall\, (x, y) \in S \quad \langle x, \hat{w} \rangle y > 0.$$

Тогда алгоритм Розенблатта, применяемый к выборке $S$, завершает свою работу после не более чем $(\sigma/\rho)^2$ циклов, где

$$\sigma = \max_{(x,y)\in S} \|x\|, \quad \rho = \frac{1}{\|\hat{w}\|} \min_{(x,y)\in S} \langle x, \hat{w}\rangle y.$$

**Доказательство**.

В доказательстве данной теоремы используется **неравенство Коши-Буняковского**, которое верно для произвольной пары $a, b$ векторов из линейного пространства со скалярным произведением $\langle\ \rangle$, имеет вид

$$\langle a, b\rangle \le \|a\|\,\|b\| \tag{5}$$

и доказывается следующим образом: $\forall\, t \in \mathbf{R}$

$$\begin{aligned} &0 \le \|at + b\|^2 = \\ &= \langle at + b, at + b\rangle = \\ &= \langle a, a\rangle t^2 + 2\langle a, b\rangle t + \langle b, b\rangle = \\ &= \|a\|^2 t^2 + 2\langle a, b\rangle t + \|b\|^2. \end{aligned}$$

Соотношение

$$\forall\, t \in \mathbf{R} \quad \|a\|^2 t^2 + 2\langle a, b\rangle t + \|b\|^2 \ge 0 \tag{6}$$

равносильно тому, что дискриминант

$$(2\langle a, b\rangle)^2 - 4\|a\|^2\|b\|^2 \tag{7}$$

квадратного трехчлена в (6) неположителен, т.е.

$$\langle a, b\rangle^2 - \|a\|^2\|b\|^2 \le 0,$$

откуда следует (5).

Обозначим записями $w^{(k-1)}$ и $w^{(k)}$ значения переменной $w$ до и после $k$–го выполнения цикла в (4). Т.к. $w^{(k)} = w^{(k-1)} + xy$, то

$$\begin{aligned} &\langle w^{(k)}, \hat{w}\rangle = \langle w^{(k-1)} + xy, \hat{w}\rangle = \langle w^{(k-1)}, \hat{w}\rangle + \langle x, \hat{w}\rangle y \ge \\ &\ge \langle w^{(k-1)}, \hat{w}\rangle + \|\hat{w}\|\rho. \end{aligned} \tag{8}$$

Применяя (8) $k$ раз, и учитывая $w^{(0)} = \bar{0}$, получаем:

$$\langle w^{(k)}, \hat{w}\rangle \ge \langle w^{(0)}, \hat{w}\rangle + k\|\hat{w}\|\rho = k\|\hat{w}\|\rho. \tag{9}$$

Согласно неравенству Коши-Буняковского, $\langle w^{(k)}, \hat{w}\rangle \le \|w^{(k)}\|\,\|\hat{w}\|$, поэтому из (9) следует, что

$$k\|\hat{w}\|\rho \le \langle w^{(k)}, \hat{w}\rangle \le \|w^{(k)}\|\,\|\hat{w}\|,$$

откуда следует неравенство $k\rho \le \|w^{(k)}\|$, или

$$k^2\rho^2 \le \|w^{(k)}\|^2. \tag{10}$$

С другой стороны,

$$\|w^{(k)}\|^2 = \langle w^{(k-1)} + xy, w^{(k-1)} + xy\rangle = \|w^{(k-1)}\|^2 + 2\langle x, w^{(k-1)}\rangle y + \|x\|^2, \tag{11}$$

и поскольку $\langle x, w^{(k-1)}\rangle y \le 0$ и $\|x\| \le \sigma$, то из (11) следует, что

$$\|w^{(k)}\|^2 \le \|w^{(k-1)}\|^2 + \|x\|^2 \le \|w^{(k-1)}\|^2 + \sigma^2. \tag{12}$$

Применяя (12) $k$ раз, и учитывая $w^{(0)} = \bar{0}$, получаем:

$$\|w^{(k)}\|^2 \le k\sigma^2. \tag{13}$$

Объединяя (10) и (13) получаем:

$$k^2\rho^2 \le \|w^{(k)}\|^2 \le k\sigma^2,$$

откуда следует неравенство $k^2\rho^2 \le k\sigma^2$, или $k\rho^2 \le \sigma^2$, или $k \le (\sigma/\rho)^2$.

Таким образом, число выполнений цикла в блок-схеме (4) не превосходит $(\sigma/\rho)^2$. ■

## 1.3 Модификации алгоритма Розенблатта

Для ускорения процесса нахождения искомого вектора $w$, удовлетворяющего условию (3), приведенный выше алгоритм можно модифицировать:

- в качестве начального значения $w$ в (4) можно брать не $\bar{0}$, а любой вектор, рекомендуется брать в качестве начального вектор

  $$\left(\frac{\langle x^1, y\rangle}{\|x^1\|^2}, \ldots, \frac{\langle x^n, y\rangle}{\|x^n\|^2}\right),$$

  где вектора $x^1, \ldots, x^n, y$, определяются следующим образом: если $S$ имеет вид $\{(x_i, y_{x_i}) \mid i = 1, \ldots, l\}$, и $\forall\, i = 1, \ldots, l \;\; x_i = (x_i^1, \ldots, x_i^n)$, то

  $$\forall\, j = 1, \ldots, n \;\; x^j = (x_1^j, \ldots, x_l^j), \quad y = (y_{x_1}, \ldots, y_{x_l})$$

  (это оптимальный выбор, если $\langle x^i, x^j\rangle$ близко к 0 при $i \ne j$),

- действие $w := w + xy$ можно заменить на $w := w + xy\eta$, где $\eta$ – фиксированное положительное число.

# 2 Алгоритмы прогнозирования с численными потерями

В этом пункте рассматривается следующая постановка задачи построения смешивающего прогнозирующего алгоритма (ПА): имеется несколько экспертов $1, \ldots, N$, но для каждого шага прогнозирования $t$ вместо прогнозов экспертов $\gamma_t^i$ известны лишь численные потери $l_t^i \in [0,1]$, которые несут эксперты на шаге $t$. Требуется построить ПА, кумулятивные потери которого были бы как можно ближе к кумулятивным потерям наилучшего из имеющихся экспертов (т.е. к $\min_{i \in I} L_T^i$).

Будем использовать понятие **вероятностного распределения (ВР)** на множестве $I = \{1, \ldots, N\}$, которое представляет собой произвольный вектор $\vec{p} = (p^1, \ldots, p^N)$ неотрицательных действительных чисел, удовлетворяющих условию $\sum_{i \in I} p^i = 1$. Множество всех ВР на $I$ будем обозначать записью $I^\triangle$. Вектор из $I^\triangle$, все компоненты которого совпадают (т.е. равны $\frac{1}{N}$) будем называть **равномерно распределенным (р.р.)**.

Также будем использовать следующее обозначение – если $\vec{w} = (w^1, \ldots, w^N)$ – вектор неотрицательных действительных чисел, то $norm(\vec{w})$ – это ВР $(p^1, \ldots, p^N)$, где

$$\forall\, i = 1, \ldots, N \;\; p^i = \frac{w^i}{|\vec{w}|}, \quad |\vec{w}| = \sum_{i=1}^N w^i.$$

Предлагается следующее решение описанной выше задачи: на каждом шаге прогнозирования $t$ определяется ВР $\vec{p}_t = (p_t^1, \ldots, p_t^N) \in I^\triangle$, и прогноз искомого алгоритма $A$ в момент $t$ полагается равным прогнозу эксперта $i$, номер которого выбран из $I$ случайным образом, в соответствии с распределением $\vec{p}_t$ (т.е. с вероятностью $p_t^1$ выбран эксперт 1, с вероятностью $p_t^2$ выбран эксперт 2, и т.д.). Потери ПА $A$ в момент $t$ совпадают с потерями выбранного эксперта $i$ в момент $t$.

Нетрудно видеть, что математическое ожидание $l_t$ потери ПА $A$ в момент $t$ совпадает со **скалярным произведением** $\langle \vec{p}_t, \vec{l}_t \rangle = \sum_{i \in I} p_t^i l_t^i$, где $\vec{l}_t = (l_t^1, \ldots, l_t^N)$.

$\forall\, t = 1, \ldots, T$ определяем $\vec{p}_t$ как $norm(\vec{w}_t)$, где

$$\begin{array}{l} \vec{w}_1 \text{ – р.р.} \\ \forall\, t = 1, \ldots, T-1, \forall\, i \in I \;\; w_{t+1}^i := w_t^i \beta^{l_t^i} \end{array} \tag{14}$$

где $\beta \in (0,1)$ – параметр.

Описанный выше алгоритм называется **алгоритмом оптимального распределения потерь**, и обозначается записью $Hedge(\beta)$.

**Лемма 4.1**

Средняя кумулятивная потеря $L_T = \sum_{t=1}^{T} l_t$ данного алгоритма удовлетворяет неравенству

$$\ln |\vec{w}_{T+1}| \leq -(1-\beta)L_T. \tag{15}$$

**Доказательство**.

Докажем эквивалентное неравенство:

$$|\vec{w}_{T+1}| \leq e^{-(1-\beta)L_T}. \tag{16}$$

Согласно определению (14), $\forall\, t = 1, \ldots, T$

$$\begin{array}{l} |\vec{w}_{t+1}| = \sum_{i\in I} w_{t+1}^i = \sum_{i\in I} w_t^i \beta^{l_t^i} \leq \\ \leq \sum_{i\in I} w_t^i (1-(1-\beta)l_t^i) \end{array} \tag{17}$$

(в (17) используем неравенство

$$\beta^{l_t^i} \leq 1-(1-\beta)l_t^i, \tag{18}$$

которое следует из выпуклости функции $y = \beta^x$).

Правая часть (17) равна

$$|\vec{w}_t| - (1-\beta)\sum_{i\in I} w_t^i l_t^i = |\vec{w}_t|(1-(1-\beta)l_t) \tag{19}$$

Из неравенства $1 + x \leq e^x$ ($\forall\, x \in \mathbf{R}$) следует неравенство

$$1-(1-\beta)l_t \leq e^{-(1-\beta)l_t} \tag{20}$$

Из (17), (19) и (20) следует, что $\forall\, t = 1, \ldots, T$

$$|\vec{w}_{t+1}| \leq |\vec{w}_t| e^{-(1-\beta)l_t} \tag{21}$$

Перемножая неравенства (21) для $t = 1, \ldots, T$, производя сокращения, и учитывая $|\vec{w}_1| = 1$, получаем искомое неравенство (16). ■

Перепишем (15) в виде

$$L_T \leq -\tfrac{1}{1-\beta} \ln |\vec{w}_{T+1}|. \tag{22}$$

Пусть $i$ – произвольный элемент множества $\{1, \ldots, N\}$. Из неравенства $w_{T+1}^i \leq |\vec{w}_{T+1}|$ следует неравенство

$$-\tfrac{1}{1-\beta} \ln |\vec{w}_{T+1}| \leq -\tfrac{1}{1-\beta} \ln w_{T+1}^i \tag{23}$$

Из (14) следует, что

$$w_{T+1}^i = w_1^i \beta^{L_T^i} = \tfrac{1}{N} \beta^{L_T^i}. \tag{24}$$

Из (22), (23) и (24) следует, что

$$L_T \le -\tfrac{1}{1-\beta}(\ln \tfrac{1}{N} + L_T^i \ln \beta). \tag{25}$$

Поскольку (25) верно для каждого $i \in \{1, \ldots, N\}$, то получаем соотношение

$$L_T \le \tfrac{1}{1-\beta} \ln \tfrac{1}{\beta} \min_{i \in I} L_T^i + \tfrac{\ln N}{1-\beta}. \tag{26}$$

Неравенство (26) означает, что средние кумулятивные потери ПА $Hedge(\beta)$ не превосходят кумулятивных потерь наилучшего эксперта, умноженных на константу $\frac{1}{1-\beta} \ln \frac{1}{\beta}$, к которым добавлен регрет (т.е. ошибка обучения) $\frac{\ln N}{1-\beta}$.

**Теорема**.

Если в ПА $Hedge(\beta)$ в качестве параметра $\beta \in (0,1)$ выбирается число $\frac{1}{1+\sqrt{\frac{2}{T/\ln N}}}$, то верна оценка

$$L_T \le \min_{i \in I} L_T^i + \sqrt{2T \ln N} + \ln N. \tag{27}$$

**Доказательство**.

Сначала докажем вспомогательное утверждение: если действительные числа $L, L', R, R'$ удовлетворяют неравенствам $L' > L \ge 0$, $R' \ge R > 0$, и $\beta = \frac{1}{1+\sqrt{\frac{2}{L'/R'}}}$, то

$$\tfrac{1}{1-\beta} \ln \tfrac{1}{\beta} L + \tfrac{1}{1-\beta} R \le L + \sqrt{2L'R'} + R \tag{28}$$

Обозначим $\sigma = \sqrt{2R'/L'}$.

Имеет место неравенство

$$\ln \tfrac{1}{\beta} \le \tfrac{1-\beta^2}{2\beta} \tag{29}$$

т.к. производная функции $f(\beta) = \frac{1-\beta^2}{2\beta} - \ln \frac{1}{\beta}$ равна $-\frac{(\beta-1)^2}{2\beta^2}$, и поскольку $f(1) = 0$, то если бы (29) было бы неверно, т.е. $f(\beta) < 0$, то, по теореме Лагранжа, $\exists\, \beta' \in (0,1) : f'(\beta') > 0$, что невозможно (производная функции $f$ отрицательна во всех точках интервала $(0,1)$).

Из (29) следует, что $\frac{1}{1-\beta}\ln\frac{1}{\beta} \le \frac{1+\beta}{2\beta}$, поэтому

$$\begin{aligned}
&\text{левая часть (28)} \ \le \tfrac{1+\beta}{2\beta}L + \tfrac{1}{1-\beta}R = \\
&= \tfrac{1}{2}(1+\tfrac{1}{\beta})L + \tfrac{1}{1-\beta}R = L + \tfrac{1}{2}L\sigma + \tfrac{1}{1-\frac{1}{1+\sigma}}R \le \\
&\le L + \sqrt{\tfrac{L'R'}{2}} + R + \tfrac{R}{\sigma} \le \text{правая часть (28)} \ .
\end{aligned}$$

(мы используем равенство $\frac{1}{1-\frac{1}{1+\sigma}} = 1 + \frac{1}{\sigma}$).

Рассматривая (28) для $L = \min_{i\in I} L_T^i$, $L' = T$, $R = R' = \ln N$, и учитывая (26), получаем (27). ■

# Межфакультетский курс «Методы искусственного интеллекта в задачах анализа данных и верификации программ» Лекция 10 Усиление алгоритмов машинного обучения

А.М.Миронов

В этой лекции рассматривается задача усиления алгоритмов машинного обучения, т.е. преобразование слабых алгоритмов машинного обучения в сильные алгоритмы машинного обучения.

Для описания этой задачи приведем необходимые определения.

Пусть заданы множества $X$ и $Y$, элементы которых называются **объектами** и **ответами** соответственно, как правило, $Y = \{0, 1\}$. **Обучающей выборкой (ОВ)** будем называть произвольную совокупность $S$ вида

$$S = \{(x^i, y^i, p^i) \mid i \in I\} \tag{1}$$

где $I = \{1, \ldots, N\}$, $\forall\, i \in I \;\; x^i \in X$, $y^i \in Y$, $(p^1, \ldots, p^N) \in I^{\triangle}$. Запись $|S|$ обозначает число компонентов в $S$ (т.е. $N$).

Каждая тройка $(x, y, p)$ из ОВ $S$ интерпретируется как утверждение о том, что объекту $x$ соответствует ответ $y$ с мерой уверенности $p$.

Под **алгоритмом машинного обучения (АМО)** понимаем алгоритм, получающий на вход ОВ $S$, и выдающий функцию $h : X \to Y$, называемую **классификатором**. **Ошибка** классификатора $h$ на ОВ $S$ – это число

$$Err(h, S) = \sum_{i \in I} p^i [\![h(x^i) \neq y^i]\!].$$

АМО называется

- **сильным**, если для каждой ОВ $S$ и $\forall\, \varepsilon, \delta \in (0, 1)$ он выдает с вероятностью $> 1 - \delta$ за время, полиномиально зависящее от $\frac{1}{\varepsilon}, \frac{1}{\delta}, |S|$, классификатор $h$, такой, что $Err(h, S) \leq \varepsilon$,

- **слабым**, если для каждой ОВ $S$ $\exists \varepsilon \in (0, \frac{1}{2})$: $\forall \delta \in (0,1)$ верно то же, что и для сильного АМО.

Ниже решается следующая задача: пусть имеется слабый АМО, требуется на базе него построить сильный АМО. Метод преобразования слабого АМО в сильный АМО называется **бустингом**. Излагаемый ниже бустинг называется **AdaBoost** (**адаптивное усиление**). Говоря нефомально, в основе данного бустинга лежит выделение таких элементов ОВ $S$, на которых классификатор $h$, получаемый при помощи слабого АМО делает наибольшую ошибку, и коррекция $h$ на именно этих элементах.

Входными данными для алгоритма AdaBoost являются ОВ (1) и слабый АМО $WeakLearn$.

Алгоритм AdaBoost имеет следующий вид. Выбирается натуральное число $T$, и выполняется нижеследующая последовательность из $T$ шагов. На каждом шаге $t$ этой последовательности определяются следующие объекты:

- вектора $\vec{w}_t = (w_t^1, \ldots, w_t^N)$ и $\vec{p}_t = (p_t^1, \ldots, p_t^N)$, где

$$\vec{p}_t = norm(\vec{w}_t),$$
$$\vec{w}_1 = (p^1, \ldots, p^N)\ (p^i \text{ – компоненты исходной ОВ } S)$$

- классификатор $h_t$, получаемый применением исходного слабого АМО $WeakLearn$ к ОВ $S(\vec{p}_t)$, где $S(\vec{p}_t)$ получается из $S$ заменой в каждой входящей в неё тройке $(x^i, y^i, p^i)$ компоненты $p^i$ на $p_t^i$,

- $\varepsilon_t := Err(h_t, S(\vec{p}_t))\ (< \frac{1}{2})$, $\beta_t := \frac{\varepsilon_t}{1-\varepsilon_t}$,

- $w_{t+1}^i := w_t^i \beta_t^{l_t^i}$, где $l_t^i = [\![h_t(x^i) = y^i]\!]$.

После выполнения этих действий определяется искомый классификатор $h$:

$$h(x) = [\![\langle \vec{q}, \vec{h}(x) \rangle \geq \tfrac{1}{2}]\!] \tag{2}$$

где $\vec{q} = norm(\ln \frac{1}{\beta_1}, \ldots, \ln \frac{1}{\beta_T}), \vec{h}(x) = (h_1(x), \ldots, h_T(x))$.

**Теорема 4.5**

Ошибка результирующего классификатора (2) удовлетворяет неравенству

$$Err(h, S) \leq 2^T \prod_{t=1}^{T} \sqrt{\varepsilon_t(1-\varepsilon_t)}. \tag{3}$$

**Доказательство**.

Согласно определениям, $\forall\, t = 1, \ldots, T$ верны равенства

$$\varepsilon_t = \sum_{i \in I} p_t^i (1 - l_t^i) = 1 - \sum_{i \in I} p_t^i l_t^i = 1 - \frac{1}{|\vec{w}_t|} \sum_{i \in I} w_t^i l_t^i$$

Следовательно, $\sum_{i \in I} w_t^i l_t^i = |\vec{w}_t|(1 - \varepsilon_t)$, откуда, учитывая неравенство из предыдущей лекции для $\beta = \beta_t$, получаем:

$$\begin{array}{l} |\vec{w}_{t+1}| = \sum_{i \in I} w_t^i \beta_t^{l_t^i} \le \sum_{i \in I} w_t^i (1 - (1 - \beta_t) l_t^i) = \\ = |\vec{w}_t| - (1 - \beta_t) \sum_{i \in I} w_t^i l_t^i = \\ = |\vec{w}_t| - (1 - \beta_t) |\vec{w}_t| \sum_{i \in I} p_t^i l_t^i = \\ = |\vec{w}_t| (1 - (1 - \beta_t)(1 - \varepsilon_t)) = |\vec{w}_t| 2\varepsilon_t. \end{array} \tag{4}$$

Таким образом, $\forall\, t = 1, \ldots, T$

$$|\vec{w}_{t+1}| \le |\vec{w}_t| 2\varepsilon_t. \tag{5}$$

Перемножая неравенства (5) для $t = 1, \ldots, T$, и учитывая $|\vec{w}_1| = 1$, получаем:

$$|\vec{w}_{T+1}| \le 2^T \prod_{t=1}^T \varepsilon_t. \tag{6}$$

Отметим, что $\forall\, i \in I$ из $h(x_i) \ne y_i$ следует, что

$$\prod_{t=1}^T \beta_t^{l_t^i} \ge \left(\prod_{t=1}^T \beta_t\right)^{1/2} \tag{7}$$

Действительно, $l_t^i = 1 - |h_t(x_i) - y_i|$, и

- если $y_i = 0$ и $h(x_i) = 1$, то $\forall\, t = 1, \ldots, T$

  $$\begin{array}{l} \beta^{l_t^i} = \beta_t^{1 - |h_t(x_i) - y_i|} = \beta_t^{1 - h_t(x_i)} \\ \sum_{t=1}^T \ln \frac{1}{\beta_t} h_t(x_i) \ge \frac{1}{2} \sum_{t=1}^T \ln \frac{1}{\beta_t} \end{array}$$

  откуда следует (7) для данного случая, и

- если $y_i = 1$ и $h(x_i) = 0$, то $\forall\, t = 1, \ldots, T$

  $$\begin{array}{l} \beta^{l_t^i} = \beta_t^{1 - |h_t(x_i) - y_i|} = \beta_t^{h_t(x_i)} \\ \sum_{t=1}^T \ln \frac{1}{\beta_t} h_t(x_i) < \frac{1}{2} \sum_{t=1}^T \ln \frac{1}{\beta_t} \end{array}$$

  откуда следует (7) для данного случая.

Учитывая (7), получаем:

$$\begin{array}{l} |\vec{w}_{T+1}| \ge \sum_{i : h(x_i) \ne y_i} w_{T+1}^i = \sum_{i : h(x_i) \ne y_i} p^i \prod_{t=1}^T \beta_t^{l_t^i} \ge \\ \ge \left(\sum_{i : h(x_i) \ne y_i} p^i\right) \left(\prod_{t=1}^T \beta_t\right)^{1/2} = Err(h, S) \left(\prod_{t=1}^T \beta_t\right)^{1/2}, \end{array}$$

откуда, учитывая (6), получаем (3). ■

**Следствие 4.1**

Пусть $\forall\, t = 1, \ldots, T$ ошибка $\varepsilon_t$ классификатора $h_t$ из алгоритма AdaBoost удовлетворяет условию $\varepsilon_t \leq \frac{1}{2} - \gamma_t$, где $\gamma_t > 0$. Тогда ошибка результирующего классификатора (2) удовлетворяет условию

$$Err(h, S) \leq e^{-2\sum_{t=1}^{T} \gamma_t^2}. \tag{8}$$

**Доказательство**.

В данном случае правая часть (3) равна

$$\prod_{t=1}^{T} \sqrt{1 - 4\gamma_t^2} = e^{\sum_{t=1}^{T} \frac{1}{2} \ln(1-4\gamma_t^2)} \ ,$$

откуда, учитывая неравенство $\ln(1 - 4\gamma_t^2) \leq -4\gamma_t^2$, получаем неравенство (8). ■

В частности, если $\forall\, t = 1, \ldots, T\ \gamma_t = \gamma$, то (8) будет иметь вид

$$Err(h, S) \leq e^{-2T\gamma^2}$$

откуда получаем оценку на число итераций AdaBoost, достаточных для выполнения условия $Err(h, S) < \varepsilon$:

$$T > \frac{1}{2\gamma^2} \ln \frac{1}{\varepsilon} \ .$$

# Межфакультетский курс «Методы искусственного интеллекта в задачах анализа данных и верификации программ»
# Лекция 11
# Алгоритмы следования за лидером

А.М.Миронов

В этой лекции рассматривается другой подход к решению задачи прогнозирования. Прогнозирующий алгоритм (ПА), построенный в соответствии с данным подходом, называется **алгоритмом следования за возмущённым лидером (Follow the Perturbed Leader, FPL)**. Данный алгоритм является вероятностной модификацией обычного ПА $A$ следования за лидером, который имеет следующий вид: на каждом шаге прогнозирования $t$ определяется лидер, т.е. такой эксперт $i$, кумулятивные потери $L^i_{t-1}$ которого минимальны, и на шаге $t$ прогноз $A$ полагается равным прогнозу лидера $i$. Потери ПА $A$ в момент $t$ совпадают с потерями эксперта $i$ в момент $t$.

К сожалению, данный ПА может привести к потерям, существенно превышающим потери каждого из экспертов. Например, пусть число экспертов равно двум, и последовательности их потерь на шагах $1, \ldots, 7$ имеют вид

$$l^1_{1,\ldots,7} = (0.5,\ 0,\ 1,\ 0,\ 1,\ 0,\ 1),$$
$$l^2_{1,\ldots,7} = \quad (0,\ 1,\ 0,\ 1,\ 0,\ 1,\ 0).$$

Последовательности соответствующих кумулятивных потерь имеют вид

$$L^1_{1,\ldots,7} = (0.5,\ 0.5,\ 1.5,\ 1.5,\ 2.5,\ 2.5,\ 3.5),$$
$$L^2_{1,\ldots,7} = \quad (0,\quad 1,\quad 1,\quad 2,\quad 2,\quad 3,\quad 3).$$

Нетрудно видеть, что в данном случае лидерами на шагах $2, \ldots, 7$ являются соответственно 2, 1, 2, 1, 2, 1, и каждый раз, следуя за лидером

на текущем шаге, ПА следования за лидером будет нести потерю 1, и его кумулятивные потери на шагах $2, \ldots, 7$ будет иметь вид

$$L_{2,\ldots,7} = \ (1,\ 2,\ 3,\ 4,\ 5,\ 6)$$

т.е. его кумулятивные потери на каждом шаге примерно вдвое больше кумулятивных потерь каждого из экспертов.

Излагаемый ниже ПА FPL отличается от ПА следования за лидером лишь в изменении понятия лидера: на каждом шаге $t$ лидером среди экспертов $1, \ldots, N$ является тот эксперт $i$ (называемый **возмущённым лидером**), у которого минимальной является величина

$$L^i_{t-1} - \tfrac{1}{\varepsilon_t}\xi^i,$$

где $\varepsilon_t$ – параметр, и $\xi^1, \ldots, \xi^N$ – независимо одинаково распределенные **случайные величины (СВ)**, с экспоненциальным законом распределения, т.е. их плотность имеет вид $p(x) = e^{-x}$ $(x \geq 0)$.

$\forall\, t = 1, \ldots, T$ обозначим

$$\begin{array}{l} i_t = \ \text{СВ}\ \ \arg\min_{i\in I}(L^i_{t-1} - \frac{1}{\varepsilon_t}\xi^i) \\ l_t = \mathbf{E}l^{i_t}_t = \sum_{i\in I} l^i_t \mathbf{P}\{i_t = i\},\ \ L_T = \sum_{t=1}^T l_t, \end{array}$$

где $\mathbf{P}\{\varphi\}$ обозначает вероятность события $\varphi$, а $\mathbf{E}\xi$ обозначает математическое ожидание СВ $\xi$.

**Теорема 4.6**

Если параметр $\varepsilon_t$ из ПА FPL имеет вид $\sqrt{\frac{2\ln N}{t}}$, то

$$L_T \leq \min_{i\in I} L^i_T + 3\sqrt{2T\ln N}. \tag{1}$$

**Доказательство**.

$\forall\, t = 1, \ldots, T$ обозначим

$$\begin{array}{l} i'_t = \ \text{СВ}\ \ \arg\min_{i\in I}(L^i_t - \frac{1}{\varepsilon_t}\xi^i) \\ l'_t = \mathbf{E}l^{i'_t}_t = \sum_{i\in I} l^i_t \mathbf{P}\{i'_t = i\},\ \ L'_T = \sum_{t=1}^T l'_t. \end{array}$$

Неравенство (1) следует из доказываемых ниже соотношений (2) и (12).

1. Докажем неравенства

$$L_T \leq L'_T + \textstyle\sum_{t=1}^T \varepsilon_t \leq L'_T + 2\sqrt{2T\ln N}. \tag{2}$$

Второе неравенство следует из определения $\varepsilon_t$ и свойства

$$\sum_{t=1}^{T} \frac{1}{\sqrt{t}} \leq 1 + \int_1^T \frac{dt}{\sqrt{t}} < 2\sqrt{T}, \tag{3}$$

а первое неравенство следует из свойства

$$\forall\, t = 1, \ldots, T \quad l_t - l'_t \leq \varepsilon_t l_t \ \ (\leq \varepsilon_t, \text{ т.к. } l_t \in [0,1]). \tag{4}$$

(4) следует из неравенств

$$l'_t \geq e^{-\varepsilon_t} l_t \geq (1 - \varepsilon_t) l_t. \tag{5}$$

Второе неравенство в (5) следует из свойства

$$\forall\, x \in \mathbf{R} \ \ e^{-x} \geq 1 - x,$$

а первое неравенство в (5) можно переписать в виде

$$\sum_{i \in I} l_t^i \mathbf{P}\{i_t = i\} \leq e^{\varepsilon_t} \sum_{i \in I} l_t^i \mathbf{P}\{i'_t = i\} \tag{6}$$

(6) следует из свойства $\forall\, i \in I$

$$\mathbf{P}\{i_t = i\} \leq e^{\varepsilon_t} \mathbf{P}\{i'_t = i\}. \tag{7}$$

(7) следует из соответствующих неравенств для условных вероятностей: $\forall\, c_1, \ldots, c_N \geq 0$

$$\begin{array}{c} \mathbf{P}\{i_t = i \mid \forall\, j \neq i \ \ \xi^j = c_j\} \leq \\ \leq e^{\varepsilon_t} \mathbf{P}\{i'_t = i \mid \forall\, j \neq i \ \ \xi^j = c_j\} \end{array} \tag{8}$$

Докажем (8). Обозначим условие $\forall\, j \neq i \ \ \xi^j = c_j$ символом $\varphi$, и определим

$$\begin{array}{l} m_i = \min_{j \neq i}(L_{t-1}^j - \frac{1}{\varepsilon_t} c_j), \\ m'_i = \min_{j \neq i}(L_t^j - \frac{1}{\varepsilon_t} c_j) = \min_{j \neq i}(L_{t-1}^j + l_t^j - \frac{1}{\varepsilon_t} c_j). \end{array}$$

Нетрудно видеть, что $m_i \leq m'_i$.

Используя введенные выше обозначения, неравенство (8) можно переписать следующим образом:

$$\begin{array}{c} \mathbf{P}\{L_{t-1}^i - \frac{1}{\varepsilon_t} \xi^i \leq m_i \mid \varphi\} \leq \\ \leq e^{\varepsilon_t} \mathbf{P}\{L_{t-1}^i + l_t^i - \frac{1}{\varepsilon_t} \xi^i \leq m'_i \mid \varphi\}. \end{array} \tag{9}$$

Если в неравенстве в правой части (9) заменить $l_t^i$ на 1, а $m_i'$ на $m_i$, то данное неравенство усилится, поэтому (9) следует из неравенства

$$\begin{array}{c}\mathbf{P}\{L_{t-1}^i - \frac{1}{\varepsilon_t}\xi^i \le m_i \mid \varphi\} \le \\ \le e^{\varepsilon_t}\mathbf{P}\{L_{t-1}^i + 1 - \frac{1}{\varepsilon_t}\xi^i \le m_i \mid \varphi\}.\end{array} \tag{10}$$

которое эквивалентно неравенству

$$\begin{array}{c}\mathbf{P}\{\xi^i \ge \varepsilon_t(L_{t-1}^i - m_i) \mid \varphi\} \le \\ \le e^{\varepsilon_t}\mathbf{P}\{\xi^i \ge \varepsilon_t(L_{t-1}^i - m_i + 1) \mid \varphi\}.\end{array} \tag{11}$$

(11) обосновывается следующими свойствами экспоненциально распределенной СВ $\xi$: $\forall\, a, b \ge 0$

$$\begin{array}{l}\mathbf{P}\{\xi \ge a\} = e^{-a}, \\ \mathbf{P}\{\xi \ge a + b\} = e^{-b}\mathbf{P}\{\xi \ge a\}.\end{array}$$

2. Докажем, что

$$L_T' \le \min_{i\in I} L_T^i + \frac{\ln N}{\varepsilon_T}. \tag{12}$$

Будем использовать следующие обозначения:

$$\begin{array}{l}\vec{l_t} := (l_t^1, \ldots, l_t^N),\ \vec{L_t} := (L_t^1, \ldots, L_t^N), \\ \vec{\xi} := (\xi^1, \ldots, \xi^N), \\ \tilde{l_t} = \vec{l_t} - \vec{\xi}(\frac{1}{\varepsilon_t} - \frac{1}{\varepsilon_{t-1}}),\ \tilde{L_t} = \vec{L_t} - \vec{\xi}\frac{1}{\varepsilon_t}\end{array} \tag{13}$$

(полагаем $\varepsilon_0 = 1$).

Нетрудно доказать, что $\tilde{L}_T = \tilde{L}_{T-1} + \tilde{l}_T$.

Пусть $E = \{\vec{e}_1, \ldots, \vec{e}_N\} \subseteq \mathbf{R}^N$, где $\forall\, i = 1, \ldots, N$ $\vec{e}_i$ имеет вид $(0, \ldots, 1, \ldots 0)$ (единица – на $i$-м месте).

$\forall\, \vec{l} = (l^1, \ldots, l^N) \in \mathbf{R}$ обозначим

$$M(\vec{l}) = \arg\min_{\vec{e}_i \in E} \langle \vec{e}_i, \vec{l} \rangle,$$

Нетрудно видеть, что

$$\begin{array}{l}\langle M(\vec{l}), \vec{l}\rangle = \min_{i\in I} l^i, \\ l_t^{i_t'} = \langle M(\tilde{L}_t), \vec{l_t}\rangle,\ \ L_T' = \mathbf{E}\sum_{t=1}^T \langle M(\tilde{L}_t), \vec{l_t}\rangle, \\ \langle M(\tilde{L}_{T-1}), \tilde{L}_{T-1}\rangle \le \langle M(\tilde{L}_T), \tilde{L}_{T-1}\rangle\end{array} \tag{14}$$

(неравенство в третьей строчке (14) следует из того, что его левая часть – минимальная компонента $\tilde{L}_{T-1}$, а правая – некоторая компонента $\tilde{L}_{T-1}$).

Индукцией по $T$ докажем неравенство

$$\sum_{t=1}^{T}\langle M(\tilde{L}_t),\tilde{l}_t\rangle \le \langle M(\tilde{L}_T),\tilde{L}_T\rangle. \tag{15}$$

При $T=1$ (15) имеет вид $\langle M(\tilde{l}_1),\tilde{l}_1\rangle \le \langle M(\tilde{l}_1),\tilde{l}_1\rangle$.

Индуктивный переход (от $T-1$ к $T$): используя индуктивное предположение, и учитывая неравенство в третьей строчке (14), получаем:

$$\begin{array}{l}\text{левая часть (15)} \le \\ \le \langle M(\tilde{L}_{T-1}),\tilde{L}_{T-1}\rangle + \langle M(\tilde{L}_T),\tilde{l}_T\rangle \le \\ \le \langle M(\tilde{L}_T),\tilde{L}_{T-1}\rangle + \langle M(\tilde{L}_T),\tilde{l}_T\rangle = \langle M(\tilde{L}_T),\tilde{L}_T\rangle = \\ = \text{ правая часть (15)}.\end{array}$$

Используя определение $\tilde{l}_t$ (см. третью строчку в (13)), (15) можно переписать так:

$$\begin{array}{l}\sum_{t=1}^{T}\langle M(\tilde{L}_t),\vec{l}_t\rangle \le \\ \le \langle M(\tilde{L}_T),\tilde{L}_T\rangle + \sum_{t=1}^{T}\langle M(\tilde{L}_t),\vec{\xi}\rangle(\frac{1}{\varepsilon_t}-\frac{1}{\varepsilon_{t-1}}).\end{array} \tag{16}$$

Из определения $\tilde{L}_T$, следует неравенство

$$\begin{array}{l}\langle M(\tilde{L}_T),\tilde{L}_T\rangle \le \langle M(\vec{L}_T),\vec{L}_T-\vec{\xi}\frac{1}{\varepsilon_T}\rangle = \\ = \min_{i\in I} L_T^i - \langle M(\vec{L}_T),\xi\rangle\frac{1}{\varepsilon_T}\end{array} \tag{17}$$

Т.к. $\langle M(\vec{L}_T),\xi\rangle = \xi^k$ для некоторого $k$, и $\mathbf{E}\xi^k = 1$, то

$$\mathbf{E}\langle M(\vec{L}_T),\xi\rangle\frac{1}{\varepsilon_T} = \frac{1}{\varepsilon_T}\mathbf{E}\xi^k = \frac{1}{\varepsilon_T}. \tag{18}$$

Оценим второй член в (16):

$$\begin{array}{l}\sum_{t=1}^{T}\langle M(\tilde{L}_t),\vec{\xi}\rangle(\frac{1}{\varepsilon_t}-\frac{1}{\varepsilon_{t-1}}) \le \\ \le \sum_{t=1}^{T}\max_{i\in I}\xi^i(\frac{1}{\varepsilon_t}-\frac{1}{\varepsilon_{t-1}}) \le \frac{1}{\varepsilon_T}\max_{i\in I}\xi^i\end{array}$$

Нетрудно доказать, что

$$\mathbf{E}\max_{i=1,\dots,N}\xi^i \le 1+\ln N. \tag{19}$$

Действительно, поскольку СВ $\xi^1,\dots,\xi^N$ независимы, и функция распределения показательно распределённой СВ имеет вид $1-e^{-x}$,

то функция распределения СВ $\max_{i=1,\ldots,N}\xi^i$ имеет вид $(1-e^{-x})^N$, поэтому её плотность равна $N(1-e^{-x})^{N-1}$, и следовательно её мат. ожидание равно

$$N\int_0^\infty (1-e^{-x})^{N-1}e^{-x}xdx. \tag{20}$$

Обозначим (20) записью $a_N$. Поскольку

$$\begin{array}{l} a_N = N\int_0^\infty (1-e^{-x})^{N-1}e^{-x}xdx = \\ = N\int_0^\infty (1-e^{-x})(1-e^{-x})^{N-2}e^{-x}xdx = \\ = \frac{N}{N-1}a_{N-1} - N\int_0^\infty e^{-x}(1-e^{-x})^{N-2}e^{-x}xdx = \\ = \frac{N}{N-1}a_{N-1} - \frac{N}{N-1}\int_0^\infty e^{-x}xd(1-e^{-x})^{N-1} = \\ \text{(применяем интегрирование по частям)} \\ = \frac{N}{N-1}a_{N-1} + \frac{N}{N-1}\int_0^\infty (1-e^{-x})^{N-1}de^{-x}x = \\ = \frac{N}{N-1}a_{N-1} + \frac{N}{N-1}\int_0^\infty (1-e^{-x})^{N-1}e^{-x}(1-x)dx = \\ = \frac{N}{N-1}a_{N-1} + \frac{1}{N-1}(1-a_N), \end{array}$$

откуда получаем: $a_N = a_{N-1} + \frac{1}{N}$, следовательно

$$\mathbf{E}\max_{i=1,\ldots,N}\xi^i = a_N = 1 + \tfrac{1}{2} + \ldots + \tfrac{1}{N}, \tag{21}$$

откуда следует (19), поэтому

$$\begin{array}{l} \mathbf{E}\sum_{t=1}^T \langle M(\tilde{L}_t), \vec{\xi}\rangle(\frac{1}{\varepsilon_t} - \frac{1}{\varepsilon_{t-1}}) \leq \\ \leq \frac{\mathbf{E}\max_{i=1,\ldots,N}\xi^i}{\varepsilon_T} \leq \frac{1+\ln N}{\varepsilon_T} \end{array} \tag{22}$$

Таким образом, согласно второй строчке в (14), а также (16), (17), (18) и (22)

$$\begin{array}{l} L'_T = \mathbf{E}\sum_{t=1}^T \langle M(\tilde{L}_t), \vec{l}_t\rangle \leq \\ \leq \min_{i\in I} L^i_T - \frac{1}{\varepsilon_T} + \frac{1+\ln N}{\varepsilon_T} \end{array}$$

откуда следует (12). ■

# Межфакультетский курс «Методы искусственного интеллекта в задачах анализа данных и верификации программ» Лекция 12 Метод опорных векторов

А.М.Миронов

## 1 Метод опорных векторов

В этой лекции излагается наиболее популярный метод машинного обучения – **метод опорных векторов (Support Vector Machines, SVM)**, который был создан в 70-е годы прошлого века сотрудниками Института проблем управления АН СССР В. Н. Вапником и А. Я. Червоненкисом.

Данный метод предназначен для решения задач классификации и регрессионного анализа.

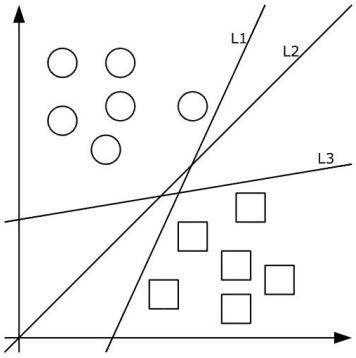
### 1.1 Оптимальность аппроксимирующих функций

В предыдущих лекциях рассматривалась задача нахождения по строго линейно разделимой выборке $S \subseteq \mathbf{R}^n \times \{-1, 1\}$ АФ $a_S$ вида

$$a_S(x) = sign\Big(\sum_{i=1}^{n} x^i w_i - w_0\Big)$$

такой, что $Q(a_S) = 0$. Как было отмечено выше, функция $a_S$ данного вида обладает свойством $Q(a_S) = 0$ тогда и только тогда, когда гиперплоскость $P$, определяемая уравнением $\sum\limits_{i=1}^{n} x^i w_i - w_0 = 0$, разделяет множества $S^+$ и $S^-$, т.е. $S^+$ и $S^-$ содержатся в разных полупространствах, на которые $P$ делит $\mathbf{R}^n$.

Можно доказать, что задача построения разделяющей гиперплоскости для строго линейно разделимой выборки $S$ имеет бесконечно много решений. Например, несколько различных решений данной задачи изображено на нижеследующем рисунке (в данном случае $n = 2$):




где кружочки обозначают элементы $S^+$, а квадратики - элементы $S^-$.

Встает вопрос о том, можно ли ввести какие-либо меры оптимальности решений данной задачи.

В качестве одной из мер оптимальности функции $a_S$ указанного выше вида можно рассматривать, например, расстояние

$$\rho(S^+ \cup S^-, P) \tag{1}$$

между $S^+ \cup S^-$ и $P$. Напомним, что $\forall\, A, B \subseteq \mathbf{R}^n$ расстояние $\rho(A, B)$ между $A$ и $B$ определяется как $\inf\limits_{a \in A, b \in B} \|a - b\|$.

Назовем **полосой, разделяющей** $S^+$ **и** $S^-$, и определяемой гиперплоскостью $P$, часть пространства $\mathbf{R}^n$, заключенную между гиперплоскостями $P_{S^+}$ и $P_{S^-}$, которые получаются параллельным переносом гиперплоскости $P$ вдоль вектора нормали к ней

- по направлению к $S^+$ на расстояние $\rho(P, S^+)$, и
- по направлению к $S^-$ на расстояние $\rho(P, S^-)$,

соответственно. Будем обозначать эту полосу записью $[P_{S^+}, P_{S^-}]$. Нетрудно видеть, что во внутренней части полосы $[P_{S^+}, P_{S^-}]$ точек из $S^+$ и $S^-$ нет.

Расстояние $\rho(P_{S^+}, P_{S^-})$ назовем **шириной** полосы $[P_{S^+}, P_{S^-}]$. Можно доказать, что (1) достигает максимального значения, когда ширина полосы $[P_{S^+}, P_{S^-}]$ равна $\rho(S^+, S^-)$, и $P$ находится посередине этой полосы.

Назовем гиперплоскость $P$, находящуюся посередине полосы $[P_{S^+}, P_{S^-}]$ с шириной $\rho(S^+, S^-)$, **оптимальной гиперплоскостью**, разделяющей выборку $S$. Ниже излагается метод построения такой гиперплоскости.

Кроме того, ниже вводится еще одна мера оптимальности АФ $a_S$, и излагается алгоритм построения функции $a_S$, оптимальной относительно этой меры.

## 1.2 Построение оптимальной разделяющей гиперплоскости для строго линейно разделимой выборки

### 1.2.1 Описание задачи

В этом пункте мы предполагаем, что задана строго линейно разделимая выборка $S$, и $P$ – какая-либо гиперплоскость, разделяющая $S^+$ и $S^-$.

По определению, определенные выше гиперплоскости $P_{S^+}$ и $P_{S^-}$ параллельны, поэтому можно считать, что их уравнения различаются лишь в свободном члене и имеют вид

$$\langle x, v\rangle - a = 0, \quad \langle x, v\rangle - b = 0, \quad \text{где } v \in \mathbf{R}^n,\ a, b \in \mathbf{R},\ a \neq b. \tag{2}$$

$\forall\, \lambda \in \mathbf{R} \setminus \{0\}$ уравнения

$$\langle x, \lambda v\rangle - \lambda a = 0, \quad \langle x, \lambda v\rangle - \lambda b = 0 \tag{3}$$

равносильны соответствующим уравнениям в (2), т.е. определяют те же самые гиперплоскости $P_{S^+}$ и $P_{S^-}$. Нетрудно видеть, что если в качестве $\lambda$ взять число $\frac{2}{a-b}$, то уравнения (3) будут равносильны соответствующим уравнениям

$$\langle x, w\rangle - w_0 = 1, \quad \langle x, w\rangle - w_0 = -1, \tag{4}$$

где $w = \lambda v$, $w_0 = \frac{a+b}{a-b}$. Таким образом, можно считать, что

- гиперплоскости $P_{S^+}$ и $P_{S^-}$ определяются уравнениями (4), и
- точки из $S^+$ и $S^-$ находятся в непересекающихся полупространствах, определяемых $P_{S^+}$ и $P_{S^-}$, т.е. удовлетворяют соотношениям

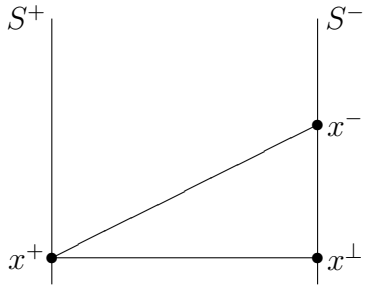
  $$\langle x, w\rangle - w_0 \geq 1 \quad \text{и} \quad \langle x, w\rangle - w_0 \leq -1$$

  соответственно.

Вычислим ширину $\rho$ полосы $[P_{S^+}, P_{S^-}]$.

Выберем на $P_{S^+}$ и $P_{S^-}$ точки $x^+$ и $x^-$ соответственно. Пусть $x^\perp$ – основание перпендикуляра, опущенного из точки $x^+$ на гиперплоскость $P_{S^-}$.

Искомая ширина $\rho$ равна длине катета $[x^+, x^\perp]$ прямоугольного треугольника с вершинами в точках $x^+, x^-, x^\perp$:




Эту длину можно вычислить как произведение

- длины гипотенузы $[x^+, x^-]$, т.е. $\|x^+ - x^-\|$, и
- косинуса угла $\varphi = \widehat{x^- x^+ x^\perp}$, который выражается через скалярное произведение: $\cos\varphi = \frac{\langle x^+ - x^-, x^+ - x^\perp\rangle}{\|x^+ - x^-\|\,\|x^+ - x^\perp\|}$,

т.е. $\rho = \frac{\langle x^+ - x^-, x^+ - x^\perp\rangle}{\|x^+ - x^\perp\|}$. Поскольку вектор $x^+ - x^\perp$ ортогонален к $P$, то он имеет вид $\mu w$ для некоторого числа $\mu$, поэтому

$$\rho = \frac{\langle x^+ - x^-, \mu w\rangle}{\|\mu w\|} = \frac{\mu}{|\mu|}\frac{\langle x^+ - x^-, w\rangle}{\|w\|} = \sigma\frac{\langle x^+ - x^-, w\rangle}{\|w\|}, \tag{5}$$

где $\sigma = 1$ или $-1$. Поскольку $x^+ \in P_{S^+}$ и $x^- \in P_{S^-}$, то

$$\langle x^+, w\rangle - w_0 = 1, \quad \langle x^-, w\rangle - w_0 = -1,$$

откуда следует, что

$$\langle x^+ - x^-, w\rangle = \langle x^+, w\rangle - \langle x^-, w\rangle = (w_0 + 1) - (w_0 - 1) = 2, \tag{6}$$

Из (5) и (6) следует, что $\sigma = 1$, и $\rho = \frac{2}{\|w\|}$.

Таким образом, задача поиска оптимальной разделяющей гиперплоскости для $S$, т.е. такой гиперплоскости $P$, которая определяет полосу $[S_P^+, S_P^-]$ максимальной ширины, сводится к следующей задаче: найти такие $w \in \mathbf{R}^n$ и $w_0 \in \mathbf{R}$, чтобы

- значение $\|w\|$ было минимально возможным, и
- были выполнены условия $\begin{cases} \forall\, x \in S^+ & \langle x, w\rangle - w_0 \geq 1, \\ \forall\, x \in S^- & \langle x, w\rangle - w_0 \leq -1, \end{cases}$ которые можно переписать в виде

$$\forall\, x \in X_S \quad y_x(\langle x, w\rangle - w_0) - 1 \geq 0 \tag{7}$$

где $X_S \stackrel{\text{def}}{=} \{x \in \mathbf{R}^n \mid (x, y_x) \in X_S\}$.

Если решение $(w, w_0)$ этой задачи найдено, то оптимальная разделяющая гиперплоскость $P$ определяется уравнением $\langle x, w\rangle - w_0 = 0$.

Заметим, что решение данной задачи совпадает с решением задачи

$$\frac{\|w\|^2}{2} \to \min \tag{8}$$

при условиях (7). Таким образом, мы свели задачу построения оптимальной разделяющей гиперплоскости для строго линейно разделимой выборки к оптимизационной задаче (8) при условиях (7).

### 1.2.2 Метод решения оптимизационной задачи

В этом пункте мы рассмотрим подход к решению общей оптимизационной задачи, частным случаем которой является оптимизационная задача (8) при условиях (7).

Данная общая задача имеет следующий вид: заданы

- функция $f : \mathbf{R}^n \to \mathbf{R}$ (называемая **целевой** функцией), которая является **выпуклой**, т.е.

$$\forall\, x, x' \in \mathbf{R}^n,\ \forall\, \alpha \in [0,1] \quad f(\alpha x + (1-\alpha)x') \le \alpha f(x) + (1-\alpha) f(x'),$$

- множество **условий** вида $g_1(x) \le 0, \ldots, g_m(x) \le 0$, где $g_1, \ldots, g_m$ – выпуклые функции вида $\mathbf{R}^n \to \mathbf{R}$.

Обозначим записью $D_{g_1,\ldots,g_m}$ множество

$$D_{g_1,\ldots,g_m} \stackrel{\text{def}}{=} \{x \in \mathbf{R}^n \mid \forall\, i = 1, \ldots, m \;\; g_i(x) \le 0\}. \tag{9}$$

Требуется найти аргумент $\hat{x} \in D_{g_1,\ldots,g_m}$, на котором достигается минимальное значение функции $f$, в предположении что минимальность рассматривается относительно множества $D_{g_1,\ldots,g_m}$, т.е. требуется, чтобы

$$f(\hat{x}) = \min_{x \in D_{g_1,\ldots,g_m}} f(x).$$

Данную задачу можно решать с помощью **теоремы Каруша-Куна-Таккера**, которую называют основной теоремой выпуклой нелинейной оптимизации. Ниже мы приводим частный случай этой теоремы.

**Теорема 1 (W.Karush, 1942, H.Kuhn and A.Tucker, 1951).**
Пусть заданы выпуклые функции $f$, $g_1, \ldots, g_m$ вида $\mathbf{R}^n \to \mathbf{R}$, причем

$$\exists\, x \in \mathbf{R}^n : \forall\, i = 1, \ldots, m \quad g_i(x) < 0.$$

Обозначим символом $L$ функцию (называемую **функцией Лагранжа**)

$$L = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x), \tag{10}$$

где $\lambda_1, \ldots, \lambda_m$ – переменные, называемые **множителями Лагранжа**.

$\forall\, \hat{x} \in D_{g_1,\ldots,g_m}$ следующие утверждения эквивалентны:

$$f(\hat{x}) = \min_{x \in D_{g_1,\ldots,g_m}} f(x), \tag{11}$$

$$\exists\, \hat{\lambda}_1, \ldots, \hat{\lambda}_m \in \mathbf{R} : \begin{cases} \forall\, i = 1 \ldots, m \quad \hat{\lambda}_i \geq 0, \\ \forall\, i = 1 \ldots, m \quad \hat{\lambda}_i g_i(\hat{x}) = 0, \\ L(\hat{x}, \hat{\lambda}_1 \ldots, \hat{\lambda}_m) = \min\limits_{x \in \mathbf{R}^n} L(x, \hat{\lambda}_1 \ldots, \hat{\lambda}_m). \end{cases} \tag{12}$$

Если функции $f, g_1, \ldots, g_m$ дифференцируемые, то функция Лагранжа $L$ тоже будет дифференцируемой, и этом случае третье соотношение в (12) равносильно соотношению

$$\forall\, i = 1, \ldots, n \quad \frac{\partial L}{\partial x_i}(\hat{x}, \hat{\lambda}_1, \ldots, \hat{\lambda}_m) = 0. \tag{13}$$

**Доказательство**.

Сначала докажем, что если для точки $\hat{x} \in D_{g_1,\ldots,g_m}$ верно утверждение (12), то для нее будет верно утверждение (11).

Т.к. $\forall\, i = 1, \ldots, m$, $\forall\, x \in D_{g_1,\ldots,g_m}$ $\hat{\lambda}_i g_i(x) \leq 0$, то

$$\forall\, x \in D_{g_1,\ldots,g_m} \quad L(x, \hat{\lambda}_1, \ldots, \hat{\lambda}_m) = f(x) + \sum_{i=1}^{m} \hat{\lambda}_i g_i(x) \leq f(x). \tag{14}$$

Из второго соотношения в (12) следует, что

$$L(\hat{x}, \hat{\lambda}_1, \ldots, \hat{\lambda}_m) = f(\hat{x}) + \sum_{i=1}^{m} \hat{\lambda}_i g_i(\hat{x}) = f(\hat{x}) + \sum_{i=1}^{m} 0 = f(\hat{x}). \tag{15}$$

Из третьего соотношения в (12) следует, что

$$\forall\, x \in \mathbf{R}^n \quad L(\hat{x}, \hat{\lambda}_1, \ldots, \hat{\lambda}_m) \leq L(x, \hat{\lambda}_1, \ldots, \hat{\lambda}_m). \tag{16}$$

Из (14) (15) и (16) следует, что для $\hat{x}$ верно (11):

$$\forall\, x \in D_{g_1,\ldots,g_m} \quad f(\hat{x}) = L(\hat{x}, \hat{\lambda}_1, \ldots, \hat{\lambda}_m) \leq L(x, \hat{\lambda}_1, \ldots, \hat{\lambda}_m) \leq f(x).$$

Теперь докажем, что если для точки $\hat{x} \in D_{g_1,\ldots,g_m}$ верно утверждение (11), то для нее будет верно утверждение (12).

Обозначим символом $\Lambda$ множество всех векторов $(\lambda_0, \ldots, \lambda_m) \in \mathbf{R}^{m+1}$, каждый из которых удовлетворяет следующему условию: $\exists x \in \mathbf{R}^n$:

$$\begin{cases} \lambda_0 > f(x) - f(\hat{x}), \\ \forall\, i = 1, \ldots, m \;\; \lambda_i \geq g_i(x). \end{cases}$$

Очевидно что множество $\Lambda$ непусто.

Нулевой вектор $\bar{0} = (0, \ldots, 0)$ не входит в $\Lambda$, т.к. иначе $\exists\, x \in \mathbf{R}^n$:

$$\begin{cases} 0 > f(x) - f(\hat{x}) \quad (\Rightarrow \quad f(x) < f(\hat{x})), \\ \forall\, i = 1, \ldots, m \;\; 0 \geq g_i(x) \quad (\Rightarrow \quad x \in D_{g_1,\ldots,g_m}), \end{cases}$$

т.е. $\min\limits_{x \in D_{g_1,\ldots,g_m}} f(x) < f(\hat{x})$, что противоречит предположению (11).

Докажем, что множество $\Lambda$ выпукло, т.е. $\forall\, \lambda, \lambda' \in \Lambda$, $\forall\, \alpha \in [0, 1]$

$$\lambda^{(\alpha)} \stackrel{\text{def}}{=} \alpha\lambda + (1 - \alpha)\lambda' \in \Lambda. \tag{17}$$

Действительно, если $\lambda$ и $\lambda'$ имеют вид $(\lambda_0, \ldots, \lambda_m)$ и $(\lambda'_0, \ldots, \lambda'_m)$ соответственно, и вектора $x, x'$ таковы, что

$$\begin{array}{l} \lambda_0 > f(x) - f(\hat{x}), \;\; \forall\, i = 1, \ldots, m \;\; \lambda_i \geq g_i(x) \\ \lambda'_0 > f(x') - f(\hat{x}), \;\; \forall\, i = 1, \ldots, m \;\; \lambda'_i \geq g_i(x') \end{array}$$

то нетрудно проверить (используя выпуклость $f$, $g_1$, …, $g_m$), что вектор $x^{(\alpha)} \stackrel{\text{def}}{=} \alpha x + (1 - \alpha)x'$ обладает свойством

$$\begin{cases} \lambda_0^{(\alpha)} > f(x^{(\alpha)}) - f(\hat{x}), \\ \forall\, i = 1, \ldots, m \;\; \lambda_i^{(\alpha)} \geq g_i(x^{(\alpha)}). \end{cases} \tag{18}$$

Распишем (18) во всех деталях:

$$\begin{cases} \lambda_0^{(\alpha)} = \alpha\lambda_0 + (1 - \alpha)\lambda'_0 > \\ \qquad > \alpha f(x) - \alpha f(\hat{x}) + (1 - \alpha) f(x') - (1 - \alpha) f(\hat{x}) \geq \\ \qquad \geq f(\alpha x + (1 - \alpha)x') - f(\hat{x}) = f(x^{(\alpha)}) - f(\hat{x}), \\ \forall\, i = 1, \ldots, m \;\; \lambda_i^{(\alpha)} = \alpha\lambda_i + (1 - \alpha)\lambda'_i \geq \\ \qquad \geq \alpha g_i(x) + (1 - \alpha) g_i(x') \geq g_i(\alpha x + (1 - \alpha)x') = g_i(x^{(\alpha)}). \end{cases}$$

Таким образом, (17) верно.

**Лемма об отделимости.**

Множество $\Lambda$ обладает **свойством отделимости**: $\exists\, \lambda^* \in \mathbf{R}^{m+1}$:

$$\lambda^* \neq \bar{0},\ \forall\, \lambda \in \Lambda\ \ \langle \lambda^*, \lambda \rangle \geq 0 \tag{19}$$

(т.е. $\Lambda$ содержится в одном из полупространств, на которые делит пространство $\mathbf{R}^{m+1}$ гиперплоскость, определяемая уравнением $\langle \lambda^*, x \rangle = 0$).

**Доказательство.**

Если размерность $Lin(\Lambda)$ линейной оболочки множества $\Lambda$ (т.е. минимального линейного подпространства пространства $\mathbf{R}^{m+1}$, содержащего $\Lambda$) меньше чем $m+1$, то в качестве искомого вектора $\lambda^*$ можно взять любой ненулевой вектор из ортогонального дополнения $Lin(\Lambda)$. В этом случае (19) верно по причине того, что

$$\forall\, \lambda \in \Lambda \quad \langle \lambda^*, \lambda \rangle = 0.$$

Пусть размерность $Lin(\Lambda)$ равна $m+1$. В этом случае $\Lambda$ содержит линейно независимое множество из $m+1$ векторов. Обозначим векторы, входящие в это множество, записями $v_1, \ldots, v_{m+1}$.

Определим $\vec{\Lambda} \stackrel{\text{def}}{=} \{\xi\lambda \mid \xi > 0, \lambda \in \Lambda\}$. Отметим, что $\Lambda \subseteq \vec{\Lambda}$, и $\bar{0} \notin \vec{\Lambda}$.

Множество $\vec{\Lambda}$ выпуклое, т.к. $\forall\, \xi, \xi' > 0$, $\forall\, \lambda, \lambda' \in \vec{\Lambda}$, $\forall\, \alpha \in [0,1]$

$$\alpha(\xi\lambda) + (1-\alpha)(\xi'\lambda') \in \vec{\Lambda}. \tag{20}$$

Действительно, вектор в (20) равен $\xi''\lambda''$, где

- $\xi'' = \alpha\xi + (1-\alpha)\xi' > 0$, и
- $\lambda'' = \frac{\alpha\xi}{\xi''}\lambda + \frac{(1-\alpha)\xi'}{\xi''}\lambda' \in [\lambda, \lambda']$, поэтому $\lambda'' \in \Lambda$.

Докажем, что $v = \sum\limits_{i=1}^{m+1} v_i$ – внутренняя точка $\vec{\Lambda}$, т.е. $\exists\, \varepsilon > 0$: $\varepsilon$–окрестность $U_v^\varepsilon \stackrel{\text{def}}{=} \{v' \in \mathbf{R}^{m+1} \mid \|v' - v\| < \varepsilon\}$ вектора $v$ лежит в $\vec{\Lambda}$.

Обозначим символом $V$ множество, состоящее из всех векторов вида $\sum\limits_{i=1}^{m+1} \alpha_i v_i$, где $\forall\, i = 1, \ldots, m\ \ |\alpha_i| \leq \frac{1}{2}$. Для каждого вектора $x$ из единичной сферы $S \stackrel{\text{def}}{=} \{x \in \mathbf{R}^{m+1} \mid \|x\| = 1\}$ обозначим записью $\varepsilon_x$ максимальное число, такое, что $\varepsilon_x x \in V$ (такое число существует, т.к. множество $V$ является ограниченным). Поскольку единичная

сфера в $\mathbf{R}^{m+1}$ – компакт, то непрерывная функция $x \mapsto \varepsilon_x$ на $S$ достигает наименьшего значения $\varepsilon > 0$ на некотором элементе $S$.

Итак, $\forall\, x \in S \;\; \varepsilon x \in V$, откуда следует, что $U_{\bar{0}}^{\varepsilon} \subseteq V$, поэтому

$$U_v^{\varepsilon} = v + U_{\bar{0}}^{\varepsilon} \subseteq v + V = \{ \sum_{i=1}^{m+1} \alpha_i v_i \mid \tfrac{1}{2} \leq \alpha_i \leq \tfrac{3}{2} \} \subseteq \vec{\Lambda}.$$

Следовательно, каждый элемент $U_{-v}^{\varepsilon} = -v + U_{\bar{0}}^{\varepsilon}$ не лежит в $\vec{\Lambda}$ (иначе некоторый отрезок из $\vec{\Lambda}$ с концами в $U_{-v}^{\varepsilon}$ и $U_v^{\varepsilon}$ содержал бы $\bar{0}$).

Обозначим записью $\vec{\Lambda}^c$ **замыкание** множества $\vec{\Lambda}$, т.е. множество, получаемое добавлением к $\vec{\Lambda}$ всех его предельных точек. Нетрудно доказать, что замыкание любого выпуклого множества является выпуклым множеством. В частности, множество $\vec{\Lambda}^c$ выпукло.

Поскольку ни одна точка из $U_{-v}^{\varepsilon}$ не является предельной для $\vec{\Lambda}$, то, в частности, $-v \notin \vec{\Lambda}^c$.

Определим $u_0$ как такой вектор, из $\vec{\Lambda}^c$, что

$$\rho(-v, u_0) = \min\{\rho(-v, u) \mid u \in \vec{\Lambda}^c\}.$$

где $\rho$ – евклидово расстояние между векторами. Такой вектор существует. Действительно, обозначим

- символом $R$ расстояние от $-v$ до какого-либо элемента $\vec{\Lambda}^c$, и
- записью $B_{-v}^R$ замкнутый шар с центром в $-v$ радиуса $R$.

Поскольку множество $B_{-v}^R$ замкнуто и ограничено, то непустое множество $\vec{\Lambda}^c \cap B_{-v}^R$ тоже замкнуто и ограничено, т.е. оно компактно. Нетрудно видеть, что

$$\min\{\rho(-v, u) \mid u \in \vec{\Lambda}^c\} = \min\{\rho(-v, u) \mid u \in \vec{\Lambda}^c \cap B_{-v}^R\}.$$

Существование точки $u_0$, на которой функция $u \mapsto \rho(-v, u)$ на компактном множестве $\vec{\Lambda}^c \cap B_{-v}^R$ принимает наименьшее значение, обосновывается теми же методами, которыми обосновывается аналогичное утверждение в теореме из параграфа **??**.

Также аналогичными методами обосновывается следующее утверждение: гиперплоскость $P$, проходящая через точку $u_0$ и ортогональная отрезку $[-v, u_0]$, делит пространство $\mathbf{R}^{m+1}$ на два полупространства, одно из которых (обозначим его $L$) имеет вид

$$L = \{x \in \mathbf{R}^{m+1} \mid x = u_0 \text{ или } \widehat{-vu_0x} \geq \tfrac{\pi}{2}\}$$

и $\vec{\Lambda}^c \subseteq L$.

Докажем, что $\bar{0} \in P$. Пусть $\bar{0} \notin P$. Тогда $\bar{0} \neq u_0 \in P$. Т.к. $\bar{0} \in \vec{\Lambda}^c$ ($\bar{0}$ – предельная точка $\vec{\Lambda}$), то $\widehat{-vu_0\bar{0}} \geq \frac{\pi}{2}$. Поскольку $u_1 \stackrel{\text{def}}{=} 2u_0 \in \vec{\Lambda}^c$, то $\widehat{-vu_0u_1} \geq \frac{\pi}{2}$. Однако углы $(-vu_0\bar{0})$ и $(-vu_0u_1)$ являются смежными, сумма их величин равна $\pi$, что возможно только если $\widehat{-vu_0\bar{0}} = \frac{\pi}{2}$, поэтому $\bar{0} \in P$.

Вектор, определяемый отрезком $[-v, u_0]$, т.е. $u_0 - (-v) = u_0 + v$, ортогонален $P$ и принадлежит $L$, поэтому

$$L = \{x \in \mathbf{R}^{m+1} \mid \langle u_0 + v, x\rangle \geq 0\}$$

и в качестве искомого вектора $\lambda^*$ можно взять $u_0 + v$. ■

Продолжим доказательство теоремы Каруша-Куна-Таккера.
Пусть вектор $\lambda^*$, обладающий свойством (19), имеет вид $(\lambda_0^*, \ldots, \lambda_m^*)$.

1. Докажем, что $\forall\, i = 0, \ldots, m \;\; \lambda_i^* \geq 0$.

   Поскольку $\Lambda$ содержит вектора

   $$\lambda^{(0)} = (1, 0, \ldots, 0) \quad \text{и} \quad \lambda^{(i)} = (\varepsilon, 0 \ldots, 0, 1, 0, \ldots, 0) \quad (\forall\, i = 1, \ldots, m)$$

   где $\varepsilon > 0$, и единица в $\lambda^{(i)}$ стоит в позиции номер $i$ (мы считаем, что первая позиция в $\lambda^{(i)}$ имеет номер 0), то из (19) следует, что

   $$\begin{array}{l} \langle \lambda^*, \lambda^{(0)}\rangle = \lambda_0^* \geq 0 \\ \forall\, i = 1, \ldots, m \;\; \langle \lambda^*, \lambda^{(i)}\rangle = \lambda_0^*\varepsilon + \lambda_i^* \geq 0 \end{array}$$

   откуда, ввиду произвольности $\varepsilon$, заключаем, что $\lambda_i^* \geq 0$.

2. Докажем, что

   $$\forall\, i \in \{1, \ldots, m\} \quad \lambda_i^* g_i(\hat{x}) = 0. \tag{21}$$

   Т.к. $\forall\, \varepsilon > 0$ множество $\Lambda$ содержит вектор

   $$\lambda \stackrel{\text{def}}{=} (\varepsilon, 0, \ldots, 0, g_i(\hat{x}), 0, \ldots, 0),$$

   то из (19) следует, что $\langle \lambda^*, \lambda\rangle = \lambda_0^*\varepsilon + \lambda_i^* g_i(\hat{x}) \geq 0$, откуда, ввиду произвольности $\varepsilon$, заключаем, что $\lambda_i^* g_i(\hat{x}) \geq 0$.

   Таким образом, либо $\lambda_i^* g_i(\hat{x}) = 0$, либо $\lambda_i^* g_i(\hat{x}) > 0$. Однако если бы было верно неравенство $\lambda_i^* g_i(\hat{x}) > 0$, то, учитывая доказанное в предыдущем пункте неравенство $\lambda_i^* \geq 0$, заключаем, что $g_i(\hat{x}) > 0$, что противоречит условию $g_i(\hat{x}) \leq 0$.

3. Докажем, что $\forall\, x \in \mathbf{R}^n$

$$\lambda_0^* f(\hat{x}) + \sum_{i=1}^{m} \lambda_i^* g_i(\hat{x}) \le \lambda_0^* f(x) + \sum_{i=1}^{m} \lambda_i^* g_i(x). \tag{22}$$

Из (21) следует, что левая часть (22) равна $\lambda_0^* f(\hat{x})$.

Т.к. $\forall\, \varepsilon > 0$ множество $\Lambda$ содержит вектор

$$\lambda \stackrel{\text{def}}{=} (f(x) - f(\hat{x}) + \varepsilon, g_1(x), \ldots, g_m(x)),$$

то из (19) следует, что

$$\langle \lambda^*, \lambda \rangle = \lambda_0^*(f(x) - f(\hat{x}) + \varepsilon) + \sum_{i=1}^{m} \lambda_i^* g_i(x) \ge 0,$$

откуда, ввиду произвольности $\varepsilon$, следует (22).

Докажем, что $\lambda_0^* \neq 0$.

По условию, $\exists\, x$: $\forall\, i = 1, \ldots, m \;\; g_i(x) < 0$.

Если $\lambda_0^* = 0$, то из $\lambda^* \neq \bar{0}$ следует, что $\exists\, i \in \{1, \ldots, m\} : \lambda_i^* > 0$, поэтому в данном случае левая часть (22) равна 0, а правая часть(22) меньше 0, что невозможно.

Искомые значения $\hat{\lambda}_1, \ldots, \hat{\lambda}_m$ определяются как $\frac{\lambda_1^*}{\lambda_0^*}, \ldots, \frac{\lambda_m^*}{\lambda_0^*}$.

Истинность каждого из трех соотношений в (12) следует из соответствующих пунктов предыдущего рассуждения.

Докажем, что если функции $f, g_1, \ldots, g_m$ дифференцируемые, то третье соотношение в (12) равносильно соотношению (13), т.е.

$$L(\hat{x}, \hat{\lambda}) = \min_{x \in \mathbf{R}^n} L(x, \hat{\lambda}) \quad \Leftrightarrow \quad \forall\, i = 1, \ldots, n \quad \frac{\partial L}{\partial x_i}(\hat{x}, \hat{\lambda}) = 0, \tag{23}$$

где $\hat{\lambda} = (\hat{\lambda}_1 \ldots, \hat{\lambda}_m)$.

Импликация «$\Rightarrow$» в (23) верна потому, что ее правая часть является необходимым условием минимума дифференцируемой функции.

Обоснуем импликацию «$\Leftarrow$» в (23). Из выпуклости $f, g_1, \ldots, g_m$ и неотрицательности $\hat{\lambda}_1$, …, $\hat{\lambda}_m$ следует выпуклость функции $x \mapsto L(x, \hat{\lambda})$:

$$\begin{aligned}
&\forall\, x, x' \in \mathbf{R}^n,\ \forall\, \alpha \in [0,1] \quad L(\alpha x + (1-\alpha)x', \hat{\lambda}) = \\
&= f(\alpha x + (1-\alpha)x') + \sum_{i=1}^{m} \hat{\lambda}_i g_i(\alpha x + (1-\alpha)x') \le \\
&\le \alpha f(x) + (1-\alpha) f(x') + \sum_{i=1}^{m} \hat{\lambda}_i (\alpha g_i(x) + (1-\alpha) g_i(x')) = \\
&= \alpha L(x, \hat{\lambda}) + (1-\alpha) L(x', \hat{\lambda}),
\end{aligned}$$

откуда следует, что значение аргумента $\hat{x}$, удовлетворяющее правой части (23), является глобальным минимумом этой функции. ■

### 1.2.3 Применение метода

Применим изложенный выше метод к оптимизационной задаче (8) при условиях (7). В данном случае

- целевая функция $f$ имеет вид $\frac{\|w\|^2}{2}$, и
- условия являются линейными неравенствами

$$y_x(\langle x, w\rangle - w_0) - 1 \geq 0, \quad \text{где } x \in X_S. \tag{24}$$

Докажем, что целевая функция выпукла. Данная функция является суперпозицией трех функций: функции $w \mapsto \|w\|$, функции $x \mapsto x^2$, и функции $x \mapsto \frac{1}{2}x$. Эти функции выпуклы, т.к.

- выпуклость функции $w \mapsto \|w\|$ следует из неравенства треугольника ($\|a + b\| \leq \|a\| + \|b\|$) для нормы в произвольном векторном пространстве: $\forall\, w, w' \in \mathbf{R}^n,\ \forall\, \alpha \in [0, 1]$

$$\|\alpha w + (1 - \alpha)w'\| \leq \|\alpha w\| + \|(1 - \alpha)w'\| = \alpha\|w\| + (1 - \alpha)\|w'\|,$$

- выпуклость функции $x \mapsto x^2$ обосновывается следующим образом: $\forall\, x, x' \in \mathbf{R},\ \forall\, \alpha \in [0, 1]$ требуемое неравенство

$$(\alpha x + (1 - \alpha)x')^2 \leq \alpha x^2 + (1 - \alpha)(x')^2$$

  после раскрытия скобок, перегруппировки слагаемых и приведения подобных членов преобразуется в эквивалентное неравенство

$$\alpha^2(x - x')^2 \leq \alpha(x - x')^2$$

  которое верно потому, что $\alpha \in [0, 1]$,

- функция $x \mapsto \frac{1}{2}x$ выпукла потому, что любая линейная функция является выпуклой.

Нетрудно доказать, что если функции $f : \mathbf{R}^n \to \mathbf{R}$ и $g : \mathbf{R} \to \mathbf{R}$ выпуклы и, кроме того, $g$ монотонно неубывающая, то их суперпозиция $(g \circ f)$ тоже выпукла. Действительно, $\forall\, x, x' \in \mathbf{R}^n,\ \forall\, \alpha \in [0, 1]$

$$\begin{aligned}
&(g \circ f)(\alpha x + (1 - \alpha)x') = g(f(\alpha x + (1 - \alpha)x')) \leq \\
&\leq g(\alpha f(x) + (1 - \alpha)f(x')) \leq \alpha g(f(x)) + (1 - \alpha)g(f(x')) = \\
&= \alpha(g \circ f)(x) + (1 - \alpha)(g \circ f)(x').
\end{aligned}$$

Поскольку функции $x \mapsto x^2$ и $x \mapsto \frac{1}{2}x : \mathbf{R} \to \mathbf{R}$ – выпуклы и монотонно неубывающие, то, следовательно их суперпозиция с выпуклой функцией $w \mapsto \|w\|$, т.е. функция $w \mapsto \frac{1}{2}\|w\|^2$ тоже выпукла. ■

Функция Лагранжа для данной задачи имеет вид

$$L = \frac{\|w\|^2}{2} - \sum_{x \in X_S} \lambda_x (y_x(\langle x, w \rangle - w_0) - 1), \tag{25}$$

и соотношение (13) имеет следующий вид:

$$\forall\, i = 1, \ldots, n \quad \hat{w}_i - \sum_{x \in X_S} \hat{\lambda}_x y_x x_i = 0, \quad 0 - \sum_{x \in X_S} \hat{\lambda}_x y_x (-1) = 0,$$

что можно переписать в виде

$$\hat{w} = \sum_{x \in X_S} \hat{\lambda}_x y_x x, \quad \sum_{x \in X_S} \hat{\lambda}_x y_x = 0. \tag{26}$$

Из теоремы 1 следует, что исходная задача сводится к задаче поиска вектора $\hat{w}$, числа $\hat{w}_0$, и набора чисел $\hat{\lambda} = \{\hat{\lambda}_x \geq 0 \mid x \in X_S\}$, удовлетворяющих соотношениям в (26) и условию

$$\forall\, x \in X_S \quad \begin{cases} y_x(\langle x, \hat{w} \rangle - \hat{w}_0) - 1 \geq 0 \\ \hat{\lambda}_x (y_x(\langle x, \hat{w} \rangle - \hat{w}_0) - 1) = 0. \end{cases} \tag{27}$$

**Теорема 2**.

Задача нахождения объектов $\hat{w}$, $\hat{w}_0$, и $\hat{\lambda} = \{\hat{\lambda}_x \geq 0 \mid x \in X_S\}$, удовлетворяющих соотношениям (26) и (27), сводится к задаче нахождения объектов $\hat{w}$, $\hat{w}_0$, и $\hat{\lambda}$, минимизирующих значения выражения

$$\sum_{x \in X_S} \hat{\lambda}_x (y_x(\langle x, \hat{w} \rangle - \hat{w}_0) - 1) \tag{28}$$

при условиях

$$\begin{cases} \hat{w} = \sum\limits_{x \in X_S} \hat{\lambda}_x y_x x, \quad \sum\limits_{x \in X_S} \hat{\lambda}_x y_x = 0, \\ \forall\, x \in X_S \quad \begin{cases} \hat{\lambda}_x \geq 0 \\ y_x(\langle x, \hat{w} \rangle - \hat{w}_0) - 1 \geq 0. \end{cases} \end{cases} \tag{29}$$

**Доказательство**.

1. Пусть объекты $\hat{w}$, $\hat{w}_0$, и $\hat{\lambda} = \{\hat{\lambda}_x \geq 0 \mid x \in X_S\}$ удовлетворяют соотношениям (26) и (27). Тогда при их подстановке вместо соответствующих объектов в (28) и (29) получаем, что

- значение суммы (28) будет равно 0 (т.к., согласно второму равенству в (27), каждое слагаемое в этой сумме равно 0), и
- соотношения в (29) верны, это следует из (26) и (27).

С другой стороны, сумма (28) при условиях (29), не может быть меньше 0, т.к., согласно этим условиям, каждое ее слагаемое является произведением неотрицательных чисел.

Таким образом, объекты $\hat{w}$, $\hat{w}_0$, и $\hat{\lambda} = \{\hat{\lambda}_x \geq 0 \mid x \in X_S\}$ – решение задачи минимизации суммы (28) при условиях (29).

2. Согласно условиям (29), каждое слагаемое в сумме (28) при этих условиях неотрицательно, т.е. сумма (28) неотрицательна, и
   - если минимальное значение этой суммы равно 0, то каждое слагаемое в этой сумме равно 0, т.е. объекты $\hat{w}$, $\hat{w}_0$, и $\hat{\lambda}$, решающие задачу минимизации (28) при условиях (29), удовлетворяют соотношениям (26) и (27), и
   - если минимальное значение этой суммы больше 0, то тогда решение задачи нахождения $\hat{w}$, $\hat{w}_0$, и $\hat{\lambda} = \{\hat{\lambda}_x \geq 0 \mid x \in X_S\}$, удовлетворяющих соотношениям (26) и (27), не существует (что по предположению невозможно). ■

Перепишем сумму (28) путем раскрытия скобок, перегруппировки слагаемых и использования линейности скалярного произведения:

$$\begin{aligned}
&\sum_{x\in X_S} \hat{\lambda}_x y_x \langle x, \hat{w}\rangle - \sum_{x\in X_S} \hat{\lambda}_x y_x \hat{w}_0 - \sum_{x\in X_S} \hat{\lambda}_x = \\
&= \sum_{x\in X_S} \langle \hat{\lambda}_x y_x x, \hat{w}\rangle - (\sum_{x\in X_S} \hat{\lambda}_x y_x)\hat{w}_0 - \sum_{x\in X_S} \hat{\lambda}_x = \\
&= \langle \sum_{x\in X_S} \hat{\lambda}_x y_x x, \hat{w}\rangle - (\sum_{x\in X_S} \hat{\lambda}_x y_x)\hat{w}_0 - \sum_{x\in X_S} \hat{\lambda}_x.
\end{aligned} \tag{30}$$

Из условий (29) следует, что (30) можно переписать в виде

$$\langle \hat{w}, \hat{w}\rangle - \sum_{x\in X_S} \hat{\lambda}_x = \|\hat{w}\|^2 - \sum_{x\in X_S} \hat{\lambda}_x. \tag{31}$$

Выражение (31) можно переписать, используя лишь переменные $\hat{\lambda}_x$:

$$\sum_{x,x'\in X_S} \hat{\lambda}_x \hat{\lambda}_{x'} y_x y_{x'} \langle x, x'\rangle - \sum_{x\in X_S} \hat{\lambda}_x. \tag{32}$$

Таким образом, исходная задача свелась к задаче нахождения набора

$$\hat{\lambda} = \{\hat{\lambda}_x \mid x \in X_S\}$$

минимизирующего значение выражения (32), при условиях

$$\forall\, x \in X_S \quad \hat{\lambda}_x \geq 0, \quad \sum_{x \in X_S} \hat{\lambda}_x y_x = 0.$$

Такая задача называется **задачей квадратичного программирования (ЗКП)**. Существует много алгоритмов решения этой задачи.

Искомый вектор $\hat{w}$ вычисляется по найденному решению $\hat{\lambda}$ данной ЗКП согласно первому равенству в (29). Для вычисления искомого $\hat{w}_0$ выбирается такая пара $x \in X_S$, что $\hat{\lambda}_x \neq 0$, в этом случае, согласно второму равенству в (27), должно быть верно равенство

$$y_x(\langle x, \hat{w}\rangle - \hat{w}_0) - 1 = 0,$$

из которого следует, что

$$\hat{w}_0 = \langle x, \hat{w}\rangle - y_x.$$

Если данная ЗКП имеет не единственное решение, то среди всех этих решений выбирается такое, что число $\hat{w}_0$, вычисленное по этому решению, удовлетворяет последнему неравенству в (29).

Обоснуем, почему $\exists\, x \in X_S : \hat{\lambda}_x \neq 0$. Если бы все числа $\hat{\lambda}_x$ были равны 0, то $\hat{w}$ – нулевой вектор, и из последнего неравенства в (29) следует, что $\forall\, x \in X_S \quad -y_x\hat{w}_0 - 1 \geq 0$, или $-y_x\hat{w}_0 \geq 1$. Выборка $S$ предполагается нетривиальной, т.е. $y_x$ м.б. равно как 1, так и $-1$, откуда следует, что $\hat{w}_0 \geq 1$ и $-\hat{w}_0 \geq 1$, что невозможно. ■

# Межфакультетский курс «Методы искусственного интеллекта в задачах анализа данных и верификации программ» Лекция 13 Агрегирующий алгоритм и теоретико-игровые методы

А.М.Миронов

## 1 Агрегирующий алгоритм

### 1.1 Смешиваемые функции потерь

Будем использовать следующие понятия и обозначения (некоторые из них были введены ранее):

- $Y = \{0, 1\}$ – множество **исходов**,
- $\Gamma = [0, 1]$, или $[-1, 1]$, или $Y^{\triangle}$ – множество **прогнозов**,
- $\eta > 0$ – **параметр обучения**, $\beta = e^{-\eta}$,
- $\lambda : Y \times \Gamma \to \mathbf{R}_{\geq 0}$ – **функция потерь (ФП)**, непрерывна по второму аргументу,
- $I = \{1, \ldots, N\}$ – множество **экспертов**,
- $\forall\, t \geq 1,\ \forall\, i \in I$
  - $\gamma_t^i$ – **прогноз** эксперта $i$ на шаге $t$,
  - $y_t$ – **исход** на шаге $t$,
  - $l_t^i = \lambda(y_t, \gamma_t^i)$ – **потери** эксперта $i$ на шаге $t$,

– $L_t^i = \sum_{t'=1}^{t} l_{t'}^i$ **кумулятивные потери** эксперта $i$ на шаге $t$,

- **алгоритм обучения**: это построение последовательностей $\vec{w}_0, \vec{w}_1, \ldots$ и $\vec{p}_0, \vec{p}_1, \ldots$ векторов из $\mathbf{R}^I$, где
  - $\vec{w}_0 = \vec{p}_0 \in I^\triangle$,
  - $\forall\, t \geq 1, \forall\, i \in I \;\; w_t^i = \beta^{l_t^i} w_{t-1}^i = \beta^{L_t^i} p_0^i$,
  - $\forall\, t \geq 1 \;\; \vec{p}_t = norm(\vec{w}_t) \in I^\triangle$,
- $g_t = \log_\beta \sum_{i \in I} \beta^{l_t^i} p_{t-1}^i$ – **средние потери** на шаге $t$,
- $L_t = \sum_{t'=1}^{t} g_{t'}$ – **средние кумулятивные потери** на шаге $t$.

Отметим, что $L_t = \log_\beta \sum_{i \in I} \beta^{L_t^i} p_0^i$. Действительно,

$$\begin{aligned} g_t &= \log_\beta \textstyle\sum_{i \in I} \beta^{l_t^i} p_{t-1}^i = \log_\beta \sum_{i \in I} \beta^{l_t^i} \frac{w_{t-1}^i}{\sum_{j \in I} w_{t-1}^j} = \\ &= \log_\beta \frac{\sum_{i \in I} \beta^{l_t^i} w_{t-1}^i}{\sum_{j \in I} w_{t-1}^j} = \log_\beta \frac{\sum_{i \in I} \beta^{L_t^i} p_0^i}{\sum_{j \in I} \beta^{L_{t-1}^j} p_0^i} = \\ &= \log_\beta \textstyle\sum_{i \in I} \beta^{L_t^i} p_0^i - \log_\beta \sum_{i \in I} \beta^{L_{t-1}^i} p_0^i, \end{aligned}$$

откуда непосредственно следует доказываемое равенство.

Примеры ФП:

$$\lambda(y, \gamma) = \begin{cases} -\frac{1}{\eta} \ln |\bar{y} - \gamma| \;\; \text{(логарифмическая, } \bar{0} = 1, \bar{1} = 0), \\ c(y - \gamma)^2 \;\; \text{(квадратичная, } c \text{ – константа)}, \\ c|y - \gamma| \;\; \text{(абсолютная, } c \text{ – константа)}, \\ [\![ y \neq \gamma ]\!] \;\; \text{(простая)}, \\ -\ln \gamma(y) \;\; \text{(вероятностная, } \Gamma = Y^\triangle). \end{cases}$$

ФП $\lambda$ называется **смешиваемой ФП (СФП)**, если

$$\mathcal{U} = \textstyle\bigcup_{\gamma \in \Gamma} [0, \beta^{\lambda(0,\gamma)}] \times [0, \beta^{\lambda(1,\gamma)}]$$

является выпуклым подмножеством $\mathbf{R}^2$ (это будет, например, когда ФП $\lambda$ – логарифмическая или квадратичная).

## 1.2 Описание агрегирующего алгоритма прогнозирования

В настоящей лекции рассматривается задача вычисления более качественных прогнозов по сравнению с теми прогнозами, которые вычислялись в предыдущих главах, в том случае, когда ФП $\lambda$ является СФП.

Агрегирующий алгоритм - это функция, сопоставляющая каждому моменту времени $t$ прогноз $\gamma_t^* \in \Gamma$, определяемый в соответствии с нижеследующим Предложением 5.2.

**Предложение 5.2**

Если $\lambda$ – СФП, то $\forall\, t \geq 1\ \exists\, \gamma_t^* \in \Gamma$ : каким бы ни было значение исхода $y_t \in Y$, будет верно неравенство

$$\lambda(y_t, \gamma_t^*) \leq g_t. \tag{1}$$

**Доказательство**.

Совокупность точек

$$\{(\beta^{\lambda(0,\gamma_t^i)}, \beta^{\lambda(1,\gamma_t^i)}) \mid i \in I\} \tag{2}$$

принадлежит выпуклому множеству $\mathcal{U}$.

Согласно определению средних потерь на шаге $t$, величина $g_t$ является выпуклой комбинацией вида

$$g_t = \log_\beta \textstyle\sum_{i \in I} \beta^{\lambda(y_t,\gamma_t^i)} p_{t-1}^i.$$

Т.к. $\mathcal{U}$ выпукло, то выпуклая комбинация

$$\textstyle\sum_{i \in I} (\beta^{\lambda(0,\gamma_t^i)}, \beta^{\lambda(1,\gamma_t^i)}) p_{t-1}^i \tag{3}$$

точек из множества (2) тоже принадлежит $\mathcal{U}$.

Построим луч с началом в $\mathbf{0} = (0, 0)$, проходящий через точку (3). Этот луч пересекает границу

$$\{(\beta^{\lambda(0,\gamma)}, \beta^{\lambda(1,\gamma)}) \mid \gamma \in \Gamma\}$$

множества $\mathcal{U}$ в некоторой точке $\mathbf{P} = (\beta^{\lambda(0,\gamma_t^*)}, \beta^{\lambda(1,\gamma_t^*)})$.

Поскольку точка (3) принадлежит отрезку $[\mathbf{0}, \mathbf{P}]$, то её абсцисса и ордината не превосходят абсциссы и ординаты соответственно точки $\mathbf{P}$, т.е.

$$\textstyle\sum_{i \in I} \beta^{\lambda(0,\gamma_t^i)} p_{t-1}^i \leq \beta^{\lambda(0,\gamma_t^*)},$$
$$\textstyle\sum_{i \in I} \beta^{\lambda(1,\gamma_t^i)} p_{t-1}^i \leq \beta^{\lambda(1,\gamma_t^*)}.$$

т.е. $\forall y \in Y$ верно неравенство

$$\textstyle\sum_{i \in I} \beta^{\lambda(y,\gamma_t^i)} p_{t-1}^i \leq \beta^{\lambda(y,\gamma_t^*)},$$

логарифмируя которое, и заменяя $y$ на $y_t$, получаем (1). ■

Если $\lambda$ – СФП, то будем обозначать записью $L_t^{AA}$ кумулятивную потерю $\sum_{t'=1}^{t} \lambda(y_{t'}, \gamma_{t'}^*)$, где $\gamma_{t'}^*$ – прогнозы, определяемые в доказательстве

предложения 5.2 ($AA$ является аббревиатурой словосочетания «агрегирующий алгоритм»).

Из предложения 5.2 следует свойство

$$\forall\, t \geq 1 \quad L_t^{AA} \leq L_t.$$

Отметим, что если $\vec{p}_0$ – р.р., то

$$L_t^{AA} \leq L_t = \log_\beta(\textstyle\sum_{i\in I} \beta^{L_t^i} \frac{1}{N}) \leq \log_\beta(\beta^{L_t^i} \frac{1}{N}) \quad (\forall\, i \in I),$$

поэтому

$$L_t^{AA} \leq \min_{i\in I} L_t^i + \tfrac{\ln N}{\eta}.$$

Таким образом, регрет агрегирующего алгоритма равен $\frac{\ln N}{\eta}$.

# 2 Теория игр в прогнозировании

## 2.1 Понятие матричной игры

В **матричной игре** принимают участие два игрока, имеющие множества **стратегий** $X$ и $Y$ соответственно. Предполагается заданной **функция выигрыша** $f : X \times Y \to \mathbf{R}$. Если $X$ и $Y$ конечны, то функцию $f$ можно представить в виде матрицы с коэффициентами $f(x, y)$. Сеанс игры заключается в том что каждый из участников выбирает стратегию $x \in X$ и $y \in Y$ соответственно (выбор стратегий производится независимо), и после выполнения этого сеанса выигрыш первого участника совпадает с потерями второго участника и равен $f(x, y)$. Описанная выше игра обозначается записью $(X, Y, f)$.

**Нижняя цена** и **верхняя цена** игры $(X, Y, f)$ определяются как величины

$$\underline{v} = \sup_{x\in X} \inf_{y\in Y} f(x, y) \quad \text{и} \quad \bar{v} = \inf_{y\in Y} \sup_{x\in X} f(x, y)$$

соответственно, они интерпретируются как **гарантированный выигрыш** первого участника, и **гарантированная потеря** второго участника, соответственно.

Докажем, что $\underline{v} \leq \bar{v}$. Поскольку $\forall\, x \in X, \forall\, y \in Y$

$$\inf_{y\in Y} f(x, y) \leq f(x, y) \leq \sup_{x\in X} f(x, y) \tag{4}$$

то $\inf_{y\in Y} f(x, y) \leq \sup_{x\in X} f(x, y)$, откуда следует $\underline{v} \leq \bar{v}$.

Пример $\underline{v} \neq \bar{v}$: орлянка $\begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix}$, $\underline{v} = -1, \bar{v} = 1$.

**Седловая точка (СТ)** игры $(X, Y, f)$– это пара $(x_0, y_0)$ из $X \times Y$, такая, что

$$\forall\, x \in X, \forall\, y \in Y \quad f(x, y_0) \le f(x_0, y_0) \le f(x_0, y). \tag{5}$$

СТ существует не для каждой игры.

Если игра имеет СТ $(x_0, y_0)$, то $\underline{v} = \bar{v} = f(x_0, y_0)$. Действительно, из (5) следует, что

$$\bar{v} = \inf_{y\in Y} \sup_{x\in X} f(x, y) \le \sup_{x\in X} f(x, y_0) = f(x_0, y_0) = \\ = \inf_{y\in Y} f(x_0, y) \le \sup_{x\in X} \inf_{y\in Y} f(x, y) = \underline{v} \le \bar{v}.$$

Пусть заданы две игры вида $(X, Y, f)$ и $(X, Y, f')$, где

- либо $\exists\, a \in \mathbf{R}$: $\forall\, x \in X, \forall\, y \in Y \quad f'(x, y) = a + f(x, y)$,
- либо $\exists\, a > 0$: $\forall\, x \in X, \forall\, y \in Y \quad f'(x, y) = af(x, y)$.

Нетрудно доказать, что в этом случае $\forall\, x \in X, \forall\, y \in Y$

$$(x, y) \text{ – СТ игры } (X, Y, f) \;\Leftrightarrow\; (x, y) \text{ – СТ игры } (X, Y, f').$$

## 2.2 Смешанные расширения матричных игр

Матричная игра $(X, Y, f)$ называется **конечной**, если $X$ и $Y$ конечны: $X = \{1, \dots, N\}$, $Y = \{1, \dots, M\}$.

**Смешанным расширением** конечной игры $(X, Y, f)$ называется игра $(X^\triangle, Y^\triangle, f^\triangle)$, где функция $f^\triangle : X^\triangle \times Y^\triangle \to \mathbf{R}$ определяется следующим образом: $\forall\, p \in X^\triangle,\ \forall\, q \in Y^\triangle$

$$f^\triangle(p, q) = \textstyle\sum_{x\in X, y\in Y} f(x, y) p_x q_y. \tag{6}$$

где $p = (p_1 \dots p_N)$, $q = (q_1 \dots q_M)$.

Элементы множеств $X^\triangle$ и $Y^\triangle$ будем называть **смешанными стратегиями** участников. Смешанные стратегии вида $(0 \dots 1 \dots 0)$ (одна единица, и остальные – нули) будем называть **чистыми стратегиями**. Совокупность чистых стратегий из $X^\triangle$ будем обозначать записью $X^\bullet$.

Функцию $f^\triangle$ можно интерпретировать следующим образом: если участники выбирают свои стратегии случайным образом, первый – в соответствии с распределением $p$ а второй – $q$, то математическое ожидание выигрыша первого участника и потерь второго участника равно (6).

Ниже мы будем обозначать функцию выигрыша смешанного расширения конечной игры тем же символом, что и функцию выигрыша основной игры (т.е. без индекса $\triangle$).

**Теорема 7.3** (фон Нейман)
Для любой конечной игры её смешанное расширение имеет СТ. ■

$(p^*, q^*)$ – СТ iff

$$q^* \in \{q \mid \min_p f(p,q) \to \max\}$$
$$p^* \in \{p \mid \max_q f(p,q) \to \min\}$$

Будем обозначать СТ смешанного расширения конечной игры записью $(p^*, q^*)$, а её цену – $v^*$.

**Теорема 7.4**
Для любой конечной игры $(X, Y, f)$ и любых распределений $p \in X^\triangle$, $q \in Y^\triangle$ следующие условия эквивалентны:

$$(p,q) \text{ – СТ смешанного расширения } (X^\triangle, Y^\triangle, f), \tag{7}$$

$$\forall\, e \in X^\bullet, \forall\, e' \in Y^\bullet \quad f(e,q) \le f(p,q) \le f(p,e'). \tag{8}$$

**Доказательство**.
Нетрудно проверить, что

$$\forall\, q \in Y^\triangle \;\; \max_{p \in X^\triangle} f(p,q) = \max_{e \in X^\bullet} f(e,q)$$
$$\forall\, p \in X^\triangle \;\; \min_{q \in Y^\triangle} f(p,q) = \min_{e' \in Y^\bullet} f(p,e')$$

$$v = \max_p \min_j f(p,e_j) = \min_q \max_i f(e_i,q). \;\blacksquare \tag{9}$$

## 2.3 Нахождение решения смешанного расширения

Считаем все $a_{ij} = f(x,y) > 0$ (иначе прибавим одно и то же число ко всем компонентам матрицы игры)

- Нахождение $p^*$:
  (9) $\Rightarrow \exists\, p = (p_1 \dots p_N) : \forall\, j \;\; f(p,e_j) \ge v$, т.е.

  $$\textstyle\sum_i a_{ij} p_i \ge v \;\; (\forall\, j)$$

  Обозначим $x_i = \frac{p_i}{v}$, т.е. $\sum_i x_i = \frac{1}{v}$ при условиях

  $$\begin{array}{l} \sum_{i=1}^N a_{ij} x_i \ge 1 \;\; (\forall\, j), \\ x_i \ge 0 \;\; (\forall\, i) \end{array} \tag{10}$$

  Поиск решения = ЗЛП: найти $\vec{x} : \sum_{i=1}^N x_i \to \min$ при условиях (10).

- Нахождение $\vec{q}^*$:

  (9) $\Rightarrow \exists\, \vec{q} = (q_1 \dots q_M) : \forall\, i\ f(e_i, \vec{q}) \le v$, т.е.

  $$\textstyle\sum_j a_{ij} q_j \le v \ \ (\forall\, i)$$

  Обозначим $x'_j = \frac{q_j}{v}$, т.е. $\sum_j x'_j = \frac{1}{v}$ при условиях

  $$\begin{array}{l} \sum_{j=1}^{M} a_{ij} x'_j \le 1 \ \ (\forall\, i), \\ x'_j \ge 0 \ \ (\forall\, j) \end{array} \tag{11}$$

  Поиск решения = ЗЛП: найти $\vec{x}'$ : $\sum_{j=1}^{M} x'_j \to \max$ при условиях (11).