Логические основы программирования

Лекции Г.Э. Яхъяевой для 2 курса ФИТ НГУ

1 семестр, 2019 г.

Содержание

Если нашли очепятки, то пишите мне, я поправлю :)

1. Автоматизация рассуждений

В прошлом семестре мы с вами сталкивались с какими нибудь большущими-пребольшущими формулами логики предикатов, с кванторами всякими, замороченными... и мы хотели понять, являются ли они тождественно истинными или нет. Чтобы выяснить это, мы рассуждали логически. Но единого алгоритма проверки истинности для формул логики предикатов не существует. Если мы интуитивно понимали, что формула истинная, то мы доказывали от противного и так приходили к противоречию. Если же наоборот, то искали контраргумент. Но это все хорошо, когда эти задачки на сообразительность кажутся занимательными и мы их решали.

Гораздо сложнее становится, когда мы хотим автоматизировать, когда хотим придумать программу, которая сама будет решать задачи. Это сделать достаточно сложно... Тем более существует еще одна проблема: теоремы Гёделя и Чёрча о неразрешимости. Что значит неразрешимо? Т.е. нет такого рекурсивного алгоритма, проверяющего, т.и. формула или т.л.? Есть у нас рекурсивно-перечислимые алгоритмы, т.е. проблема перечислима, но не разрешима. Это значит, что не существует алгоритма, на вход которому мы подали формулу, а на выходе выдавался ответ на истинность.

Но опять же, проблема перечислима. Мы можем на вход подать формулу и, если она т.и., то за конечное число шагов мы вычислим это. В противном случае алгоритм будет работать бесконечно и никогда не остановится. На практике строятся алгоритмы, которые, грубо говоря, приближённые. Они считают какую-то вероятность ответа: т.е. если все же программа выяснила, что формула т.и., то нам повезло и на выходе получаем истину. В противном же случае после какого то N-го шага, какого-то этапа, программа завершает работу и показывает некоторую вероятность того, что формула т.л.. Тут

проблема состоит в том, на каком этапе обрывать проверку формулы, но это решают различные алгоритмы...

Допустим, такой алгоритм(что решает формулы) у нас есть. Как мы доказывали т.и. формулы? Через построение дерева вывода. Можно ли это запрограммировать? Да. Но у нас правил вывода достаточно много, и мы сидим и думаем, каким способом построить дерево, в какой момент нужно дерево строить подобным образом, а когда наоборот... Т.е. здесь все равно нужна человеческая сообразительность. Перебором построить дерево практически невозможно. Поэтому процедуру построения дерева надо как то оптимизировать. Оптимизация заключается в том, что строится *правило резолюции*(чуть позже разберём) и у нас все доказывается с использованием только одного правила. Вот это и есть основная идея нашего автомата.

§23. Скулемовские стандартные формы

Определение 23.1.

Литерой мы будем называть атомарную формулу или её отрицание.

Атомарная формула(вспомним) - это:

$P(\overline{x}),\ \neg P(\overline{x}),\ t_1 = t_2,\ t_1 \neq t_2.$

Дизъюнктом называют дизъюнкцию литер. Аналогия - элементарная дизъюнкция.

$\square$ обозначают пустой дизъюнкт(когда нечего дизъюнктить).

Пусть у нас есть сигнатура логики предикатов σ, $\varphi \in S(\sigma)$. Пусть φ в **ПНФ** имеет вид $\varphi = Q_1 x_1 \dots Q_n x_n\ \psi(\overline{x}),\ Q_i \in \{\exists, \forall\}$ и ψ-бесквантворная.

Обозначим через r номер первого $\exists$ в φ:

1) $r = 1,\ \varphi = \exists x_1 Q_2 x_2 \dots Q_n x_n\ \psi(\overline{x})$;

Тогда возъмём новую константу $c \notin \sigma$ и возьмём $\sigma' = \sigma \cup \{c\}$. Построим формулу $\varphi' = Q_2 x_2 \dots Q_n x_n \ \psi(c, x_2, \dots, x_n)$. Так мы избавились от квантора существования, добавив новую константу.

2) $r > 1$, т.е. $\varphi = \forall_1 x_1 \dots \forall_{r-1} x_{r-1} \exists_r x_r Q_{r+1} x_{r+1} \dots Q_n x_n \ \psi(x_1, \dots, x_n)$. Обогощаем сигнатуру функцией $f^{(r-1)} \notin \sigma$, $\sigma' = \sigma \cup \{f^{(r-1)}\}$. Построим $\varphi' = \forall_1 x_1 \dots \forall_{r-1} x_{r-1} Q_{r+1} x_{r+1} \dots Q_n x_n \ \psi(x_1, \dots, x_{r-1}, f(x_1, \dots, x_{r-1}), x_{r+1}, \dots, x_n)$. Так мы избавились от квантора существования, добавив новую функцию.

После этого мы посмотрели на нашу формулу φ': если она имеет еще кванторы существования, то таким же образом от них избавляемся. И когда мы дошли до конца, т.е. избавились от всех $\exists$, то такая формула обозначается $\underline{\varphi_{sko}}$ - **скулемовская стандартная форма**.

Пример:

Есть высказывание "Каждый студент прочитал хотя бы одну книгу", т.е. некоторая $\varphi = \forall x \exists y \ P(x, y)$. У Иванова прочтена одна книга, у Петрова другая книга. Т.е. у

Иванова - прочтена книга a,

Петрова - прочтена книга b.

Т.е. мы можем сделать функцию, что в зависимости от фамилии человека дает книгу, что тот прочел, т.е. $\varphi = \forall x \ P(x, f(x))$.

Приведем еще одно высказывание "Данную книгу прочли все студенты", т.е. $\varphi = \exists x \forall y \ P(x, y)$. Заменили книгу на константу, как бы у нас одна "константная книга", которую все прочли, т.е $\varphi' = \forall y \ P(c, y)$.

Сигнатура для **ссф** $\sigma(\varphi_{sko}) = \sigma \cup \{c_1, \dots, c_l\} \cup \{f_1, \dots, f_k\}$.

$l + k$ - число $\exists$ в φ.

Теорема 23.2.

Пусть $\varphi \in S(\sigma)$, φ - тожд.л. $\Leftrightarrow \varphi_{sko}$ - тожд.л.

Доказательство:

Рассмотрим 2 случая:

Случай 1: $\varphi = \exists_1 x_1 Q_2 x_2 \dots Q_n x_n \ \psi(\overline{x})$.

Тогда $\varphi' = Q_2 x_2 \dots Q_n x_n \ \psi(c, x_2, \dots, x_n)$.

Покажем, что φ - т.л. $\Leftrightarrow \varphi'$ - т.л.

$\boxed{(\Rightarrow)}$ Пусть φ - т.л. и φ' - выполнима $\Rightarrow \exists \mathfrak{A}' \in K(\sigma \cup \{c\}) \colon \mathfrak{A}' \vDash \varphi'$.

Тогда есть некоторая интерпретация константы, т.е. $c^{\mathfrak{A}'} = a \in |\mathfrak{A}'|$.

Распишем $\mathfrak{A}' \vDash \varphi' \Leftrightarrow \mathfrak{A}' \vDash Q_2 x_2, \dots, Q_n x_n \ \psi(c^{\mathfrak{A}'}, x_1, \dots, x_n) \Leftrightarrow$

$\Leftrightarrow \exists a \in |\mathfrak{A}'| \colon \mathfrak{A}' \vDash Q_2 x_2, \dots, Q_n x_n \ \psi(a, x_2, \dots, x_n)$.

Возьмём $\mathfrak{A} = \mathfrak{A}' \restriction \sigma$. Тогда $\exists a \in |\mathfrak{A}| \colon \mathfrak{A} \vDash Q_2 x_2, \dots, Q_n x_n \ \psi(a, x_2, \dots, x_n) \Rightarrow$

$\Rightarrow \mathfrak{A} \vDash \exists x_1 Q_2 x_2, \dots, Q_n x_n \ \psi(x_1, \dots, x_n) \Rightarrow \mathfrak{A} \vDash \varphi$. Получили противоречие с тем, что φ - т.л.

$\boxed{(\Leftarrow)}$ Пусть φ' - т.л. и φ - выполнима $\Rightarrow \exists \mathfrak{A} \in K(\sigma) \colon \mathfrak{A} \vDash \varphi$.

Т.е. $\mathfrak{A} \vDash \exists x_1 Q_2 x_2, \dots, Q_n x_n \ \psi(x_1, \dots, x_n) \Leftrightarrow$

$\Leftrightarrow \exists a \in |\mathfrak{A}| \colon \mathfrak{A} \vDash Q_2 x_2, \dots, Q_n x_n \ \psi(a, x_2, \dots, x_n)$.

Теперь обогатим сигнатуру константой $\sigma' = \sigma \cup \{c\}$ и расширим модель $\mathfrak{A}' = \mathfrak{A} \downharpoonright \sigma'$.

Тогда, по определению, $c^{\mathfrak{A}'} = a \Rightarrow \mathfrak{A}' \vDash Q_2 x_2, \dots, Q_n x_n \ \psi(c^{\mathfrak{A}'}, x_2, \dots, x_n) \Rightarrow$

$\Rightarrow \mathfrak{A}' \vDash \varphi'$. Получили противоречие с тем, что φ' - т.л.

Случай 2:

$\varphi = \forall_1 x_1, \dots, \forall_{r-1} x_{r-1} \exists_r x_r Q_{r+1} x_{r+1}, \dots, Q_n x_n \ \psi(x_1, \dots, x_n)$.

Тогда $\varphi' = \forall_1 x_1, \dots, \forall_{r-1} x_{r-1} Q_{r+1} x_{r+1}, \dots, Q_n x_n \ \psi(x_1, \dots, x_{r-1}, f(x_1, \dots, x_{r-1}), x_{r+1}, \dots, x_n)$.

Покажем, что φ - т.л. $\Leftrightarrow \varphi'$ - т.л.

$\boxed{(\Rightarrow)}$ Пусть φ - т.л. и φ' - выполнима.

Тогда $\exists \mathfrak{A}' \in K(\sigma \cup \{f\}) \colon \mathfrak{A}' \vDash \varphi'$.

Т.е. $\mathfrak{A}' \vDash Q_1 x_1, \ldots, Q_{r-1} x_{r-1} Q_{r+1} x_{r+1}, \ldots, Q_n x_n \; \psi(x_1, \ldots, x_{r-1}, f(x_1, \ldots, x_{r-1}), x_{r+1}, \ldots, x_n) \Rightarrow \forall a_1, \ldots, a_{r-1} \in |\mathfrak{A}'|$:

$: \mathfrak{A}' \vDash Q_{r+1} x_{r+1}, \ldots, Q_n x_n \; \psi(a_1, \ldots, a_{r-1}, f(a_1, \ldots, a_{r-1}), x_{r+1}, \ldots, x_n)$.

Пусть $f(a_1, \ldots, a_{r-1}) := c$, т.к. функция уже определена, то она для каждого набора аргументов выдает значение, здесь же для $a_1, \ldots, a_{r-1}$ выдается c.

Тогда для $\forall a_1, \ldots, a_{r-1} \in |\mathfrak{A}'| \; \exists c \in |\mathfrak{A}'|$:

$: \mathfrak{A}' \vDash Q_{r+1} x_{r+1}, \ldots, Q_n x_n \; \psi(a_1, \ldots, a_{r-1}, c, x_{r+1}, \ldots, x_n)$.

Возьмём $\mathfrak{A} = \mathfrak{A}' \restriction \sigma$. Тогда $\forall a_1, \ldots, a_{r-1} \in |\mathfrak{A}| \; \exists c \in |\mathfrak{A}|$:

$: \mathfrak{A} \vDash Q_{r+1} x_{r+1}, \ldots, Q_n x_n \; \psi(a_1, \ldots, a_{r-1}, c, x_{r+1}, \ldots, x_n) \Rightarrow \mathfrak{A} \vDash \varphi$.

Получили противоречие с тем, что φ - т.л.

$\boxed{(\Leftarrow)}$ Пусть φ' - т.л. и φ - выполнима $\Rightarrow \exists \mathfrak{A} \in K(\sigma) \colon \mathfrak{A} \vDash \varphi$.

Тогда $\mathfrak{A} \vDash \forall_1 x_1, \ldots, \forall_{r-1} x_{r-1} \exists_r x_r Q_{r+1} x_{r+1}, \ldots, Q_n x_n \; \psi(x_1, \ldots, x_n) \Rightarrow$

$\Rightarrow \forall a_1, \ldots, a_{r-1} \in |\mathfrak{A}| \; \exists c \in |\mathfrak{A}| \colon \mathfrak{A} \vDash Q_{r+1} x_{r+1}, \ldots, Q_n x_n \; \psi(a_1, \ldots, a_{r-1}, c, x_{r+1}, \ldots, x_n)$.

Зададим новую функцию, тогда $\sigma' = \sigma \cup \{f^{(r-1)}\}$, $\mathfrak{A}' = \mathfrak{A} \downharpoonright \sigma'$. Зададим интерпретацию для функции на модели $f^{\mathfrak{A}'}(a_1, \ldots, a_{r-1}) := c$. Тогда

$\forall a_1, \ldots, a_{r-1} \in |\mathfrak{A}'| \colon \mathfrak{A}' \vDash Q_{r+1} x_{r+1}, \ldots, Q_n x_n \psi(a_1, \ldots, a_{r-1}, f(a_1, \ldots, a_{r-1}), x_{r+1}, \ldots, x_n) \Rightarrow \mathfrak{A}' \vDash \forall x_1, \ldots, x_{r-1} Q_{r+1} x_{r+1}, \ldots, Q_n x_n \; \psi(x_1, \ldots, x_{r-1}, f(x_1, \ldots, x_{r-1}), x_{r+1}, \ldots, x_n) \Rightarrow \mathfrak{A}' \vDash \varphi'$. Получили противоречие с тем, что φ' - т.л.

Из случаев 1) и 2) получаем то, что за один шаг преобразования мы сохранили т.л. формулы. В итоге за конечное число шагов преобразования мы получаем то, что φ_{sko} так же остается т.л. Теорема доказана.

Следствие 23.3.

Предложение φ - т.и. $\Leftrightarrow (\neg\varphi)_{sko}$ - т.л.

Возьмём формулу φ и её **ссф** φ_{sko}. Нас интересует, является ли она т.л. или нет, какой у неё алгоритм определения т.л..

Пусть $\varphi_{sko} = \forall_1 x_1, \ldots, \forall_n x_n \; \psi(x_1, \ldots, x_n)$.

Пусть $\psi(x_1, \ldots, x_n)$ - конъюнкция дизъюнктов(аналогия **КНФ**, но это не КНФ).

Т.е. $\psi(x_1, \ldots, x_n) = \varphi_1(x_1, \ldots, x_m) \,\&\, \varphi_2(x_1, \ldots, x_m) \,\&\, \ldots \,\&\, \varphi_k(x_1, \ldots, x_m)$.

Здесь $\varphi_i(x_1, \ldots, x_m)$ - это дизъюнкт, $m \leqslant n$(т.к. могут быть разные формулы и в них могут быть фиктивные переменные).

Обозначим $S = \{\varphi_1(x_1, \ldots, x_m), \varphi_2(x_1, \ldots, x_m), \ldots, \varphi_k(x_1, \ldots, x_m)\}$.

Мы будем говорить, что **множество дизъюнктов S представляет φ_{sko}**.

Через S мы будем обозначать ψ и далее все рассуждения будут над множеством дизъюнктов. Нам нужно показать, что S - т.л.

Определение 23.4.

1) S - т.л. $\Leftrightarrow \varphi_{sko}$ - т.л.

2) S - выполнимо $\Leftrightarrow \varphi_{sko}$ - выполнима.

3) S - т.л. $\Leftrightarrow \forall \mathfrak{A} \; \exists a_1, \ldots, a_n \in |\mathfrak{A}| \; \exists i \colon \mathfrak{A} \nvDash \varphi_i(a_1, \ldots, a_n)$.

4) φ_{sko} выполнима $\Leftrightarrow \exists \mathfrak{A} \; \forall a_1, \ldots, a_n \in |\mathfrak{A}| \; \forall i \colon \mathfrak{A} \vDash \varphi_i(a_1, \ldots, a_n)$.

§24. Эрбрановский универсум

Мы говорили, что S - т.л., когда на любой модели существует набор всяких и т.д.. Но что значит "на любой модели"? Это значит, что у нас зафиксирована сигнатура, но могут быть разные универсумы у моделей. Пусть вот сигнатура у нас состоит из двух функций, например, + и -, а универсум может быть из натуральных, вещественных, комплексных чисел, а то и вообще каких нибудь яблок, и мы еще умудряемся их складывать...

И тогда проверять т.л. у **ссф** очень тяжело, потому что у нас необозримое множество моделей, на которых нужно будет все проверять. Будет в разы проще, если мы будем рассматривать не любые модели, а модели, у которых *некоторый фиксированный универсум*. Данное фиксированное множество называется **эрбрановским универсумом**.

Определение 24.1.

Пусть дано S - множество дизъюнктов, и имеется сигнатура множества $\sigma(S)$.

Определим $\sigma_S = \begin{cases} \sigma(S) \cup \{c\}, \text{если не имеет констант} \\ \sigma(S), \text{если имеются константы} \end{cases}$

Определение 24.2.

Пусть $S = \{\varphi_1(x_1, \ldots, x_m), \varphi_2(x_1, \ldots, x_m), \ldots, \varphi_k(x_1, \ldots, x_m)\}$ - множество дизъюнктов.

Тогда:

1) $H_S = \{t \in T(\sigma_S) \mid FV(t) = \varnothing (t \text{ - замкнутый терм})\}$ - **эрбранов универсум для** S;

2) Множество всех атомарных предложений

$A_S = \{P(h_1, \ldots, h_n) \mid P \in \sigma_S,\ h_1, \ldots, h_n \in H_S\}$ - **эрбранов базис для** S;

3) $\varphi_i(h_1, \ldots, h_n)$, $\varphi_i \in S$, $h_1, \ldots, h_n \in H_S$ - **основной пример дизъюнкта** $\varphi_i(x_1, \ldots, x_n)$;

4) $\mathfrak{A}_H = \langle H_S; \sigma_S \rangle$ - ***H*-интерпретация** S, если выполняются:

а) $\forall c \in \sigma_S : c^{\mathfrak{A}_H} = c$;

б) $\forall f^n \in \sigma_S \; \forall h_1, \ldots, h_n \in H_S$ выполняется:

$$f^{\mathfrak{A}_H}(h_1, \ldots, h_n) = f(h_1, \ldots, h_n) \in H_S.$$

Все данные модели будем называть H-интерпретацией. Обратим внимание на то, что значения истинности предикатов здесь не задаются, т.е. предикаты мы можем задавать любым образом. Таким образом у нас получается много различных H-интерпретаций. Константы и функции вписали стандартные, а в зависимости от того, как у нас истинности предикатов распределяются, у нас образовываются разные модели.

Пример:

Пусть $S = \{\underbrace{P(x_1, x_2) \vee Q(f(x_1))}_{\varphi_1(x1,x2)}, \underbrace{\neg Q(x_2) \vee R(x_1, x_3)}_{\varphi_2(x1,x2,x3)}\}$.

Тогда $\sigma(S) = \{P^{(2)}, Q^{(1)}, R^{(2)}, f^{(1)}\}$ и $\sigma_S = \{c, P^{(2)}, Q^{(1)}, R^{(2)}, f^{(1)}\}$.

Из σ_S нам нужны все замкнутые термы для H_S.

Тогда H_S примет вид $H_S = \{c, f(c), f(f(c)), \ldots\}$, т.е. будет бесконечным множеством. Если бы функции $f^{(1)}$ не было, то у нас была бы одна константа в универсуме(только в том случае, если $f(x) \neq x$!).

Эрбранов базис выглядит тогда так $A_S = \{P(c, c), P(c, f(c)), P(f(c), c), \ldots$ $\ldots, Q(c), Q(f(c)), \ldots, R(c, c), \ldots\}$, т.е. мы перебираем все возможные наборы аргументов для всех предикатов в σ_S.

Основным примером для дизъюнкта $P(x_1, x_2) \vee Q(f(x_1))$ будет $P(c, f(c)) \vee Q(f(c))$. Т.е. у нас основных примеров достаточно много.

Все зависит от наличия функции в S. Если её нет, то эрбранов универсум

состоит из одной константы, а A_S конечен.

И теперь о H-интерпретации. Мы строим модель, основным множеством которого будет H_S и σ_S. Затем задаём истинность предикатов, строя эрбрановский базис.

Так строятся эти определения.

Определение 24.3.

Пусть S - множество дизъюнктов, $\mathfrak{A} \in K(\sigma(S))$. Говорят, что $\mathfrak{A}_H$ **соответствует** $\mathfrak{A}$, если $\forall P \in \sigma(S)\ \forall h_1, \ldots, h_n \in H_S$ выполняется:

$$\mathfrak{A}' \models P(h_1^{\mathfrak{A}'}, \ldots, h_n^{\mathfrak{A}'}) \Leftrightarrow \mathfrak{A}_H \models P(h_1, \ldots, h_n),$$

где $\mathfrak{A}' = \mathfrak{A} \upharpoonright \sigma_S$.

Замечание 24.4.

Модели $\mathfrak{A}$ может соответсвовать более одной $\mathfrak{A}_H$.

Лемма 24.5.

Пусть S - множество дизъюнктов, $\mathfrak{A} \in K(\sigma(S))$, $\mathfrak{A}_H$ соответствует $\mathfrak{A}$. Тогда S выполнима на $\mathfrak{A}(\mathfrak{A} \models S[\gamma]) \Leftrightarrow \mathfrak{A}_H \models S$.

Доказательство:

($\Rightarrow$) Пусть $\mathfrak{A} \models S$ и $\mathfrak{A}_H \not\models S \Leftrightarrow \exists \varphi \in S\ \exists h_1, \ldots, h_n \in H_S$:
: $\mathfrak{A}_H \not\models \varphi(h_1, \ldots, h_n)$.

Т.к. $\varphi(\overline{h}) = P_1(\overline{h}) \vee \cdots \vee P_k(\overline{h}) \vee \neg P_{k+1}(\overline{h}) \vee \cdots \vee \neg P_{k+l}(\overline{h}) \Leftrightarrow$

$$\left.\begin{array}{ll} \exists i \in \{1, k\}: & \mathfrak{A}_H \not\models P_i(\overline{h}) \Leftrightarrow \mathfrak{A}' \models P_i(\overline{h}) \\ \text{или} & \\ \exists j \in \{k+1, k+l\}: & \mathfrak{A}_H \models P_j(\overline{h}) \Leftrightarrow \mathfrak{A}' \not\models P_j(\overline{h}) \end{array}\right\} \Rightarrow$$

$\Rightarrow \exists a \in |\mathfrak{A}|: \mathfrak{A} \not\models [P_i(\overline{h})]_a^{c \in \sigma_S \setminus \sigma(S)}$ или $\mathfrak{A} \models [P_j(\overline{h})]_a^{c \in \sigma_S \setminus \sigma(S)} \Rightarrow \mathfrak{A} \not\models S$.
Пришли к противоречию.

$(\Leftarrow)$ Пусть $\mathfrak{A}_H \vDash S$ и $\mathfrak{A} \nvDash S \Leftrightarrow \exists \varphi \in S\ \exists h_1, \ldots, h_n \in H_S$:
: $\mathfrak{A}_H \nvDash \varphi(h_1, \ldots, h_n)$.

Т.к. $\varphi(\overline{h}) = P_1(\overline{h}) \vee \cdots \vee P_k(\overline{h}) \vee \neg P_{k+1}(\overline{h}) \vee \cdots \vee \neg P_{k+l}(\overline{h}) \Leftrightarrow$

$$\left.\begin{array}{ll} \forall i \in \{1, k\}: & \mathfrak{A}_H \nvDash P_i(\overline{h}) \Leftrightarrow \mathfrak{A}' \vDash P_i(\overline{h}) \\ \text{или} & \\ \forall j \in \{k+1, k+l\}: & \mathfrak{A}_H \vDash P_j(\overline{h}) \Leftrightarrow \mathfrak{A}' \nvDash P_j(\overline{h}) \end{array}\right\} \Rightarrow$$

$\Rightarrow \forall a \in |\mathfrak{A}| : \mathfrak{A} \vDash [P_i(\overline{h})]_a^{c \in \sigma_S \setminus \sigma(S)}$ или $\mathfrak{A} \nvDash [P_j(\overline{h})]_a^{c \in \sigma_S \setminus \sigma(S)} \Rightarrow S$ выполнима на $\mathfrak{A}$. Пришли к противоречию.

Лемма доказана. (Подлежит редактированию)

Теорема 24.6.

S - т.л. $\Leftrightarrow S$ - ложно на всех $\mathfrak{A}_H$.

Доказательство:

$(\Rightarrow)$ Если S т.л., то оно ложно на всех моделях.

$(\Leftarrow)$ Пусть S - ложно на всех $\mathfrak{A}_H$, S выполнима $\Rightarrow \exists \mathfrak{A} \in K(\sigma(S))$:
: $\mathfrak{A} \vDash S \Rightarrow \exists \mathfrak{A}_H$: $\mathfrak{A}_H \vDash S$. Пришли к противоречию.

Теорема доказана.

Таким образом мы можем доказывать т.л. S, не рассматривая истинность на абсолютно всех моделях, а только на H-интерпретациях. Тем не менее, хоть мы и фиксируем универсум, у нас моделей все так же много, а в некоторых случаях их количество равно континууму. Следующая глава объяснит, как с этими моделями работать.

§25. Семантические деревья

Определение 25.1.

Пусть φ - предложение. Тогда **контрарной парой** называют пару $\{\varphi, \neg\varphi\}$.

Определение 25.2.

Семантическим деревом $D(S)$, построенным над S, будем называть бинарным деревом, в котором каждому ребру приписывается $\varphi \in A_S$, либо его отрицание следующим образом:

1) Ребрам, выходящим из одной вершины, приписываются элементы контра--рной пары;

2) Для каждой вершины N строится множество атомарных предложений $I(N)$, приписанных всем рёбрам ветвей, соединяющих N с корнем дерева.

Тогда в $I(N)$ не должны содержаться контрарные пары и все элементы в ветви перечисляются только один раз.

Пример:

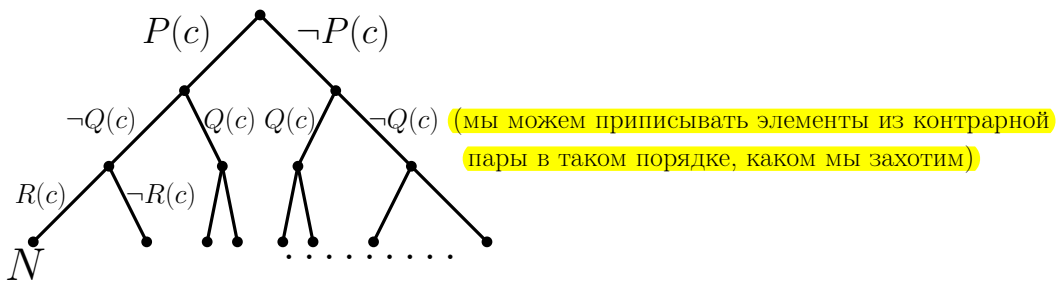
1) Пусть $S = \{P(x) \vee Q(x), R(x)\}$. Тогда:

$\sigma(S) = \{P^{(1)}, Q^{(1)}, R^{(1)}\},\ \sigma_S = \{c, P^{(1)}, Q^{(1)}, R^{(1)}\};$

$H_S = \{c\};$

$A_S = \{P(c), Q(c), R(c)\}.$

Тогда $D(S)$ будет выглядеть так (один из возможных видов):



(мы можем приписывать элементы из контрарной пары в таком порядке, каком мы захотим)

Здесь $I(N) = \{R(c), \neg Q(c), P(c)\}$.

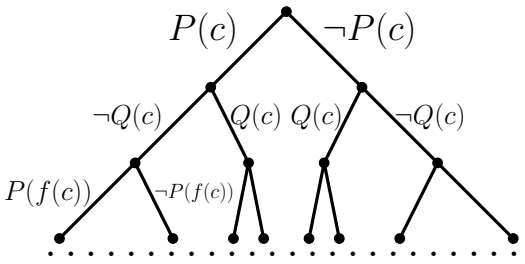
2) Пусть $S = \{P(x), Q(f(x))\}$. Тогда:

$\sigma_S = \{c, P, Q, f\}$;

$H_S = \{c, f(c), f(f(c)), f(f(f(c))), \dots\}$;

$A_S = \{P(c), Q(c), P(f(c)), Q(f((c)), \dots\}$;

$D(S)$:



Данное дерево будет бесконечным из-за наличия в S функции в дизъюнкте.

Напоминает ли ветвь из бесконечного дерева что нибудь нам? Если мы рассмотрим какую нибудь H-интерпретацию и построим её элементарную диаграмму, то можем заметить, что каждая ветвь у дерева как раз задает ту саммую диаграмму для модели, т.е. каждая бесконечная ветка интерпретирует какую-то одну H-интерпретацию.

Таким образом, у нас может быть континуум H-интерпретаций, которые полностью легли в одно семантическое дерево.

Определение 25.3.

Будем говорить, что вершина N **ставится в соответствие** некоторой $\mathfrak{A}_H$: $\mathfrak{A}_H \vDash I(N)$, т.е. одной вершине соответствует сразу несколько H-интерпретаций.

Классом моделей, соответствующих N, является

$$\mathfrak{A}_{I(N)} = \{\mathfrak{A}_H \mid \mathfrak{A}_H \vDash I(N)\}.$$

Определение 25.4.

1) Пусть $D(S)$ - конечное. Тогда $D(S)$ является **полным**, если для любой тупиковой вершины N выполняется $I(N) = \{\varphi \in A_S$ или $\neg\varphi \in A_S\}$;

2) Пусть $D(S)$ - бесконечное. Тогда $D(S)$ является **полным**, если каждая его бесконечная ветка содержит все элементы Эрбрановского базиса, взятые с отрицанием или без.

Определение 25.5.

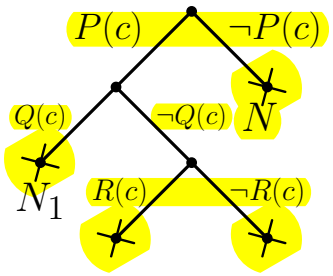
Пусть $D(S)$ - семанитическое дерево. Говорят, что вершина N называется **опровергающей вершиной**,

если $\exists\varphi(\overline{x}) \in S\ \exists h_1, \ldots, h_n \in H_S\ \forall\mathfrak{A}_H \in \mathfrak{A}_{I(N)}\colon \mathfrak{A}_H \nvDash \varphi(\overline{h})$.

Конечное поддерево полного семантического дерева называется **замкнутым**, если каждая его ветвь оканчивается опровергающей вершиной.

Пример:

1) Пусть $S = \{P(x), Q(x) \vee R(x), \neg P(x) \vee \neg Q(x), \neg P(x) \vee \neg R(x)\}$. Тогда $D(S)$:



Возьмём $\mathfrak{A}_H \in \mathfrak{A}_{I(N)}$.

$\mathfrak{A}_H \vDash \neg P(c)$.

$\mathfrak{A}_H \nvDash P(c)$(на модели ложен дизъюнкт $P(c) \in S$) $\Rightarrow$

$\Rightarrow N$ - опровергающая вершина.

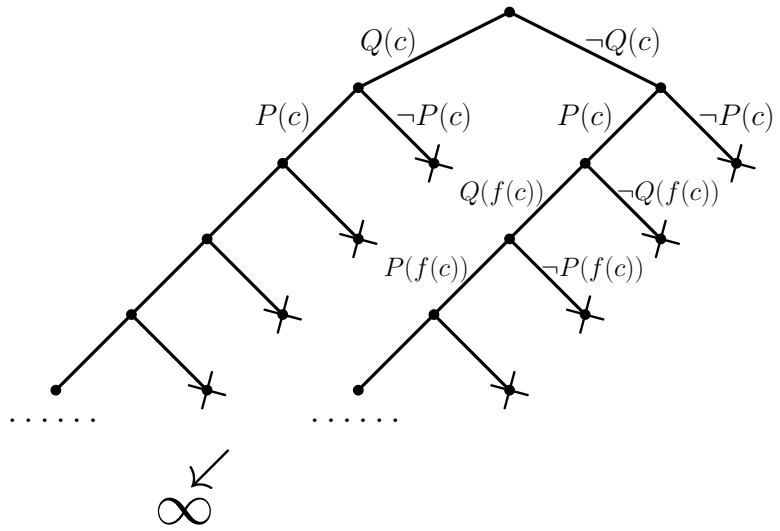
Возьмём $\mathfrak{A}'_H \in \mathfrak{A}_{I(N_1)}$.

$\mathfrak{A}'_H \vDash P(c)$ и $\mathfrak{A}'_H \vDash Q(c)$.

но $\mathfrak{A}'_H \nvDash \neg P(c) \vee \neg Q(c) \Rightarrow N_1$ - опровергающая вершина.

Рассмотрев все вершины, мы поймём, что дерево замкнутое.

2) Пусть $S = \{P(x), \neg P(x) \vee Q(f(x))\}$. Тогда $D(S)$:



В конечном итоге данное дерево является незамкнутым, но стоит добавить в S предикат $\neg Q(f(x))$, то дерево замкнется.

Чтобы понять, для чего нужны семантические деревья, нужно ввести теорему Эрбрана.

Теорема 25.6.(Эрбрана).

S - т.л. $\Leftrightarrow \forall D(S)$ существует замкнутое поддерево.

Доказательство:

$\boxed{(\Rightarrow)}$ Пусть S - т.л.. Построим полное семантическое дерево $D(S)$. Рассмотрим бесконечную ветку B. Тогда $I(B)$ - множество атомарных предложений, лежащее на данной ветви. Т.к. B бесконечная $\Rightarrow$ B не замкнутая $\Rightarrow \exists \mathfrak{A}_H$: $\mathfrak{A}_H \vDash I(B)$. Получаем противоречие, т.к. S - т.л..

$\boxed{(\Leftarrow)}$ Пусть существует замкнутое поддерево $D(S)$. Допустим, что S выполнима $\Rightarrow \exists \mathfrak{A}_H$ $\forall \varphi(\overline{x}) \in S$ $\forall h_1, \ldots, h_n \in H_S$: $\mathfrak{A}_H \vDash \varphi(\overline{h})$ $\Rightarrow$ В $D(S)$ существует бесконечная ветка $\Rightarrow$ получаем противоречие с тем, что у нас дерево замкнуто.

Теорема доказана.

Данная теорема является знаковой, т.к. она существенно упрощает задачу доказательства истинности формулы. Если мы найдём замкнутое конечное поддерево, то мы сможем обойти его за конечное число шагов и решить задачу. Но проблема все равно остается не рекурсивной, а лишь перечислимой, т.к. дерево может быть незамкнутым, а бесконечным, и мы будем по нему идти соответственно тоже бесконечно.

Но тем не менее, все алгоритмы, что ищут ответ на вопрос "т.л. формула или нет?", по сути сводятся к обходу семантического дерева.

Позже на практике мы разберем, как более оптимально нам строить данные деревья, т.к. можем его начать с абсолютно любого предиката.

Самым популярным алгоритмом для построения семантического дерева в программах поиска истинности формулы является *правило резолюции*, что рассмотрим чуть позже.

Далее мы рассмотрим применение теоремы Эрбрана.

§26. Правила Дэвиса-Патнема

Ниже приведены правила, которые упрощают доказательство т.л. S:

Предложение 26.1.(правило тавтологий).

Пусть S' получено из множества S вычёркиванием всех тавтологий, т.е. вычеркиванием всех т.и. дизъюнктов.

Тогда S - т.л. $\Leftrightarrow$ S' - т.л.

Доказательство:

Очевидно.

Предложение 26.2.(правило однолитерных дизъюнктов).

Пусть есть однолитерный дизъюнкт $L(x_1, \ldots, x_n) \in S$.

Строим множество $S' = \{\varphi \in S \mid L \notin \varphi\}$. Тогда:

1) Если $S' = \varnothing$, то S - выполнима;

2) Если $S' \neq \varnothing$, то строим $S'' = \{[\varphi]^{L}_{\neg\varnothing} \mid \varphi \in S'\}$.

Тогда S - т.л. $\Leftrightarrow$ S'' - т.л..

Доказательство:

Рассмотрим два случая:

$\boxed{S' = \varnothing} \Rightarrow \forall \varphi \in S \colon \varphi(\overline{x}) = \varphi'(\overline{x}) \vee L(\overline{x})$.

Возьмём $\mathfrak{A}_H$ и для $\forall h_1, \ldots, h_n \in H_S \colon \mathfrak{A}_H \vDash L(h_1, \ldots, h_n) \Rightarrow$

$\Rightarrow \mathfrak{A}_H \vDash \varphi \Rightarrow S$ - выполнима.

$\boxed{S' \neq \varnothing}$ Тогда строим S'' и доказываем от противного:

$\boxed{(\Rightarrow)}$ Пусть S - т.л. и S'' - выполнимо $\Rightarrow \exists \mathfrak{A}'_H \in K(\sigma_S \backslash \{L\}) \colon \mathfrak{A}'_H \vDash S''$.

Это значит, что $\forall \varphi(\overline{x}) \in S'' \; \forall h_1, \ldots, h_n \in H_{S''} \colon \mathfrak{A}'_H \vDash \varphi(h_1, \ldots, h_n)$.

Определим $\mathfrak{A}_H = \mathfrak{A}'_H \restriction \sigma_S$(доопределим L). Тогда полагаем, что

$\forall h_1, \ldots, h_s \in H_S \colon \mathfrak{A}_H \vDash L(h_1, \ldots, h_s)$.

Возьмём $\psi \in S$. Тогда рассмотрим следующие случаи:

1) $\psi = \psi' \vee L \Rightarrow \mathfrak{A}_H \vDash \psi$;

2) $\psi = \psi' \vee \neg L \Rightarrow$ т.к. $\psi' \in S'' \Rightarrow \mathfrak{A}'_H \vDash \psi' \Rightarrow \mathfrak{A}_H \vDash \psi' \Rightarrow \mathfrak{A}_H \vDash \psi$;

3) $L \notin \psi \Rightarrow$ т.к. $\psi \in S''$, выходит $\mathfrak{A}_H \vDash \psi$.

В трёх случаях вышло, что $\mathfrak{A}_H \vDash \psi \Rightarrow S$ - выполнима $\Rightarrow$ получаем противоречие с начальными условиями.

$\boxed{(\Leftarrow)}$ Пусть S'' - т.л. и S - выполнима $\Rightarrow \exists \mathfrak{A}_H \in K(\sigma_S) \colon \mathfrak{A}_H \vDash S \Rightarrow$

$\Rightarrow \forall h_1, \ldots, h_n \in H_S \colon \mathfrak{A}_H \vDash L(\overline{h}) \Rightarrow \forall \varphi \in S \colon \varphi = \varphi' \vee L$ и $\mathfrak{A}_H \vDash \varphi$.

Если $\varphi = \varphi' \vee \neg L$ и имеем, что $\mathfrak{A}_H \vDash \varphi$ и $\mathfrak{A}_H \nvDash \neg L(\overline{h}) \Rightarrow \mathfrak{A}_H \vDash \varphi' \Rightarrow$

$\Rightarrow L \notin \varphi \colon \mathfrak{A}_H \vDash \varphi$.

Теперь возьмём $\mathfrak{A}''_H = \mathfrak{A}_H \restriction \sigma''_S$, где $\sigma''_S = \sigma_S \backslash \{L\}$.

Тогда $\forall \psi \in S'' \colon \mathfrak{A}_H \vDash \psi \Rightarrow \mathfrak{A}''_H \vDash \psi \Rightarrow S''$ - выполнима $\Rightarrow$ получаем противоречие с начальными условиями.

Предложение доказано.

Пример:

Пусть $S = \{P(x), P(x) \vee Q(x), \neg P(x) \vee R(x)\}$.

Тогда

$S' = \{\neg P(x) \vee R(x)\}$,

$S'' = \{R(x)\}$.

И мы доказываем т.л. S'', если $S' \neq \varnothing$.

Предложение 26.3.(правило чистых литер).

Пусть даны $L(\overline{x})$ и множество дизъюнктов S. Пусть для $\forall \varphi \in S \colon \neg L \notin \varphi$.

Литера L - *чистая*, т.е. нигде нет её отрицания.

Тогда S - т.л. $\Leftrightarrow S' = \{\varphi \in S \mid L \notin \varphi\}$ - т.л..

Доказательство:

$\boxed{(\Rightarrow)}$ Пусть S - т.л.. Допустим, что S' - выполнима $\Rightarrow$

$\Rightarrow \exists \mathfrak{A}'_H \in K(\sigma_S \backslash \{L\}) \colon \mathfrak{A}'_H \vDash S'$.

Определим $\mathfrak{A}_H = \mathfrak{A}'_H \restriction \sigma_S$ такую, что $\forall h_1, \ldots, h_n \in H_S \colon \mathfrak{A}_H \vDash L(\overline{h})$.

Тогда мы получим, что для $\forall \varphi \in S$:

1) $\varphi = \varphi_1 \vee L \Rightarrow \forall h_1, \ldots, h_n \in H_S \colon \mathfrak{A}_H \vDash \varphi(\overline{h})$;

2) $L \notin \varphi \Rightarrow \mathfrak{A}'_H \vDash \varphi \Rightarrow \mathfrak{A}_H \vDash \varphi$.

Отсюда вышло, что $\mathfrak{A}_H \vDash \varphi \Rightarrow S$ - выполнима. Пришли к противоречию.

$\boxed{(\Leftarrow)}$ Пусть S' - т.л. и S - выполнима $\Rightarrow \exists \mathfrak{A}_H \in K(\sigma_S) \colon \mathfrak{A}_H \vDash S$.

Т.к. $S' \subseteq S \Rightarrow \mathfrak{A}_H \vDash S'$. Тогда вводим $\mathfrak{A}'_H = \mathfrak{A}_H \restriction \sigma_S \backslash \{L\}$ такую, что $\mathfrak{A}'_H \vDash S' \Rightarrow S'$ - выполнима. Пришли к противоречию.

Предложение доказано.

Предложение 26.4.(правило расщепления).

Пусть $S = \{\varphi_1 \vee L, \ldots, \varphi_n \vee L, \psi_1 \vee \neg L, \ldots, \psi_m \vee \neg L, \xi_1, \ldots, \xi_k\}$, где φ_i, ψ_j, ξ_l не содержат L.

Пусть $S_1 = \{\varphi_1, \ldots, \varphi_n, \xi_1, \ldots, \xi_k\}$ и $S_2 = \{\psi_1, \ldots, \psi_m, \xi_1, \ldots, \xi_k\}$.

Тогда S - т.л. $\Leftrightarrow$ S_1 - т.л. и S_2 - т.л..

Доказательство:

(⇒) Пусть S - т.л.. Предположим, что одно из S_1 и S_2 не является т.л.. Допустим, S_1 - выполнимо(случай для S_2 аналогичен, но не забываем про отрицание L) $\Rightarrow \exists \mathfrak{A}'_H \in K(\sigma_S \backslash \{L\}) \colon \mathfrak{A}'_H \models S_1$. Расширим модель, получим $\mathfrak{A}_H = \mathfrak{A}'_H \restriction \sigma_S$ такую, что $\forall h_1, \ldots, h_n \in H_S \colon \mathfrak{A}_H \models \neg L(\overline{h})$. Тогда получается для:

$\forall i = \overline{1,n} \colon \mathfrak{A}'_H \models \varphi_i \Rightarrow \mathfrak{A}_H \models \varphi_i \Rightarrow \mathfrak{A}_H \models \varphi_i \vee L$.

$\forall j = \overline{1,m} \colon \mathfrak{A}_H \models \neg L \Rightarrow \mathfrak{A}_H \models \psi_j \vee \neg L$.

$\forall l = \overline{1,k} \colon \mathfrak{A}'_H \models \xi_l \Rightarrow \mathfrak{A}_H \models \xi_l$.

Отсюда следует, $\mathfrak{A}_H \models S \Rightarrow S$ - выполнима. Пришли к противоречию.

(⇐) Пусть S_1 - т.л., S_2 - т.л. и S - выполнимо.

S - выполнимо $\Rightarrow \exists \mathfrak{A}_H \in K(\sigma_S) \colon \mathfrak{A}_H \models S$.

Т.к. L входит в S, то у нас есть два случая:

Сл.1. $\mathfrak{A}_H \models L(\overline{h})$;

Сл.2. $\mathfrak{A}_H \models \neg L(\overline{h})$.

Рассмотрим их подробнее.

Возьмём $\mathfrak{A}'_H = \mathfrak{A}_H \restriction \sigma_S \backslash \{L\}$. Тогда:

Сл.1.

$\forall j = \overline{1,m} \colon \mathfrak{A}_H \models \psi_j \vee \neg L$ и $\mathfrak{A}_H \models L \Rightarrow \mathfrak{A}_H \models \psi_j \Rightarrow \mathfrak{A}'_H \models \psi_j$;

$\forall l = \overline{1,k} \colon \mathfrak{A}_H \models \xi_l \Rightarrow \mathfrak{A}'_H \models \xi_l$.

Отсюда следует, что $\mathfrak{A}'_H \models S_2 \Rightarrow S_2$ - выполнимо. Пришли к противоречию.

Сл.2.

$\forall i = \overline{1,n}\colon \mathfrak{A}_H \vDash \varphi_i \vee L$ и $\mathfrak{A}_H \vDash \neg L \Rightarrow \mathfrak{A}_H \vDash \varphi_i \Rightarrow \mathfrak{A}'_H \vDash \varphi_i$;

$\forall l = \overline{1,k}\colon \mathfrak{A}_H \vDash \xi_l \Rightarrow \mathfrak{A}'_H \vDash \xi_l$.

Отсюда следует, что $\mathfrak{A}'_H \vDash S_1 \Rightarrow S_1$ - выполнимо. Пришли к противоречию.

Предложение доказано.

Следствие 26.5.

Если $S = \{\varphi \vee L, \psi \vee \neg L\}$, то S - т.л. $\Leftrightarrow$ множество $\{\varphi \vee \psi\}$ - т.л..

Это следствие и является **правилом резолюции**.

Более общее определение правила резолюции:

$$\frac{\varphi \vee L, \psi \vee \neg L}{\varphi \vee \psi}$$

Правилами Дэвиса-Патнема мы можем упрощать доказательство т.л. или же пробовать применять правило резолюции. Далее наша задача состоит в том, чтобы понять, что *правила резолюции* нам хватит для того, чтобы доказывать т.л. формул, т.е. нам более не потребуются все правила вывода с прошлого семестра. Тогда автоматизировать доказательство становится уже более просто.

Но есть еще одна проблема.

§27. Подстановка и унификация

Ну и какая же проблема у нас возникает? Рассмотрим пример:

$S = \{P(x) \vee Q(x), \neg P(f(y)) \vee R(y)\}$, x, y - свободные переменные.

Вроде бы есть P и его отрицание, так и хочется применить ПРАВИЛО РЕЗОЛЮЦИИ, но они различаются входными термами.

Тогда пусть $[P(x) \vee Q(x)]^x_{f(a)} = P(f(a)) \vee Q(f(a))$.

А в другом $[\neg P(f(y)) \vee R(y)]^y_a = \neg P(f(a)) \vee R(a)$.

Теперь *правило резолюции* можно применить.

Но можно пойти и иным путем:

$[P(x) \vee Q(x)]^x_{f(x)} = P(f(x)) \vee Q(f(x))$ и

$[\neg P(f(y)) \vee R(y)]^y_x = \neg P(f(x)) \vee R(x)$.

Подобных примеров можно придумать много и к ним применить ПРАВИЛО РЕЗОЛЮЦИИ.

Все замены называются **подстановками**, процесс приведения к одинаковым термам называется **унификацией**.

Далее мы узнаем, как это делать, и можно ли это всё *унифицировать*.

Определение 27.1.

Матрицу вида $\begin{pmatrix} x_1, \ldots, x_n \\ t_1, \ldots, t_n \end{pmatrix}$, где переменные $x_i \neq x_j$, а t_i - терм, называется **подстановкой**.

Если для $\forall i = \overline{1,n}$: t_i - замкнутый терм, то матрица называется **основной подстановкой**.

Обозначим (основную)подстановку θ, E - некоторое выражение,
а $[E]^\theta$ - **(основной)пример выражения** E.

Определение 27.2.

Пусть даны подстановки $\theta = \begin{pmatrix} x_1, \ldots, x_n \\ t_1, \ldots, t_n \end{pmatrix}$, $\lambda = \begin{pmatrix} y_1, \ldots, y_m \\ u_1, \ldots, u_m \end{pmatrix}$. Тогда как построить композицию подстановок?

Строим матрицу $\theta \circ \lambda = \begin{pmatrix} x_1, \ldots, x_n & y_1, \ldots, y_m \\ [t_1]^\lambda \ldots [t_n]^\lambda & u_1, \ldots, u_m \end{pmatrix}$, затем:

1) $\forall i = \overline{1,n}$: если $x_i = [t_i]^\lambda$, то вычеркиваем $\begin{pmatrix} x_i \\ [t_i]^\lambda \end{pmatrix}$ столбец.

2) $\forall j = \overline{1,m}$: если $\exists x_i$: $x_i = y_j$, то мы должны вычеркнуть из матрицы $\begin{pmatrix} y_j \\ u_j \end{pmatrix}$ столбец.

Данные действия проделывать в строгом порядке!

Пример:

Пусть $\theta = \begin{pmatrix} x & y \\ f(y) & z \end{pmatrix}$, $\lambda = \begin{pmatrix} x & y & z \\ a & b & y \end{pmatrix}$ Тогда

$$\theta \circ \lambda = \begin{pmatrix} x & y\ x\ y\ z \\ f(b)\ y\ a\ b\ y \end{pmatrix};$$

Применяем шаг 1:

$$\theta \circ \lambda = \begin{pmatrix} x & y\ x\ y\ z \\ f(b)\ y\ a\ b\ y \end{pmatrix};$$

Применяем шаг 2:

$$\theta \circ \lambda = \begin{pmatrix} x & x\ y\ z \\ f(b)\ a\ b\ y \end{pmatrix};$$

В конце получаем:

$$\theta \circ \lambda = \begin{pmatrix} x & y\ z \\ f(b)\ b\ y \end{pmatrix}.$$

Определение 27.3.

Пусть дано множество выражений $W = \{E_1, \ldots, E_n\}$. Говорят, что подстановка θ - **унификатор** для множества W,
если $[E_1]^\theta = [E_2]^\theta = \cdots = [E_n]^\theta$.

Пример:

$$\begin{matrix} P(x) \\ \neg P(f(y)) \end{matrix} \to \left[\theta = \begin{pmatrix} x & y \\ f(a) & a \end{pmatrix}\right] \to \begin{matrix} P(f(a)) \\ \neg P(f(a)) \end{matrix}$$

Здесь θ - унификатор.

Определение 27.4.

Унификатор σ называют **наиболее общим унификатором** для множества выражений $W = \{E_1, \ldots, E_n\}$, если для любого θ для W существует подстановка λ такая, что $\theta = \sigma \circ \lambda$, т.е. стоит применить к σ какую то подстановку, то мы получаем другой унификатор.

Теперь введём *алгоритм унификации*. Но прежде, чем его вводить, определим несколько понятий.

Определение 27.5.

Пусть дано множество выражений $W = \{E_1, \ldots, E_n\}$. Тогда **множество рассогласований** D **для** W получается появлением первой слева позиции, на которой не для всех выражений из W стоит один и тот же символ, а затем выписыванием из каждого выражения в W подвыражения, которое начинается с символа, занимающего ту же позицию.

Пример:

Пусть $W = \{P(x, f(x,y)), P(x,y), P(x, g(h(k(x))))\}$.

Тогда $D = \{f(x,y), y, g(h(k(x)))\}$, т.е. это множество, что содержит символы, с которых начались различия в выражениях из W. Это множество надо как раз *унифицировать* и найти **НОУ**.

Алгоритм унификации:

На вход подается $W = \{E_1, \ldots, E_n\}$.

Шаг 1: $k := 0$; $W_k = W$; $\sigma_k = \varnothing$;

Шаг 2: Если $||W_k|| = 1$, тогда {

σ_k - **НОУ**;

конец алгоритма;

} Иначе мы строим D_k;

Шаг 3: Если $\exists x_k \in D_k \; \exists$ терм $t_k \in D_k : x_k \notin t_k$, тогда {

$$\sigma_{k+1} := \sigma_k \circ \begin{pmatrix} x_k \\ t_k \end{pmatrix};$$

$W_{k+1} := [W_k]^{x_k}_{t_k}$;

$k := k + 1$;

goto Шаг 2;

} Иначе **НОУ** не существует и конец алгоритма;

Пример:

Пусть $W = \{P(a, x, f(g(y))), P(z, f(z), f(u))\}$.

Запустим алгоритм:

Шаг 1.

Шаг 2. Строим $D_0 = \{a, z\}$;

Шаг 3.

$$\sigma_1 = \begin{pmatrix} z \\ a \end{pmatrix};$$

$W_1 = \{P(a, x, f(g(y))), P(a, f(a), f(u))\}$;

Шаг 2. $D_1 = \{x, f(a)\}$;

Шаг 3.

$$\sigma_2 = \begin{pmatrix} z & x \\ a & f(a) \end{pmatrix};$$

$W_2 = \{P(a, f(a), f(g(y))), P(a, f(a), f(u))\}$;

Шаг 2. $D_2 = \{g(y), u\}$;

Шаг 3. Делаем последнюю замену и получаем W_3, состоящий из 1 дизъюнкта $\Rightarrow W-$унифицируем $\Rightarrow$ конец алгоритма(освободи память потом;)).

Здесь **НОУ** будет $\sigma_3 = \begin{pmatrix} z & x & u \\ a & f(a) & g(y) \end{pmatrix}$.

Важно отметить, что мы можем заменить переменную на переменную, переменную на терм, но никак не имеем права менять терм на терм или переменную!

По соглашению, переменными в этих записях принято считать x, y, z, u, v, w, а константами a, b, c и все отличные от переменных буквы.

§28. Правило резолюций

Определение 28.1.

Пусть $\varphi = L_1 \vee L_2 \vee \varphi'$, где L_1, L_2 - литеры.

Тогда если существует **НОУ** σ такой, что $[L_1]^\sigma = [L_2]^\sigma$, то $[\varphi]^\sigma$ - **склейка** дизъюнкта φ.

Пример:

Пусть $\varphi = P(x) \vee P(f(y)) \vee Q(x, y)$.

Литеры $P(x)$ и $P(f(y))$ можно склеить.

Тогда **НОУ** $\sigma = \begin{pmatrix} x \\ f(y) \end{pmatrix}$ и склейкой будет $[\varphi]^\sigma = P(f(y)) \vee Q(f(y), y)$.

Определение 28.2.(правило резолюций).

Пусть даны $\varphi(x_1, \ldots, x_n)$ и $\psi(y_1, \ldots, y_n)$, не имеющих общих переменных, две литеры $L_1 \in \varphi$ и $L_2 \in \psi$ и **НОУ** σ такое, что $[L_1]^\sigma = [\neg L_2]^\sigma$.

Тогда **резольвентой** $Res(\varphi, \psi) = [\varphi']^\sigma \backslash [L_1]^\sigma \vee [\psi']^\sigma \backslash [L_2]^\sigma$, где $\varphi' = \varphi$ или склейка φ, $\psi' = \psi$ или склейка ψ. Склейки образованы с помощью **НОУ** σ.

Пример:

Пусть $\varphi = P(x) \vee \neg Q(y) \vee P(f(y)),\ \psi = Q(a) \vee R(x, f(y))$.

Переименуем переменные. Возьмём, например, ψ, и получим

$\psi_1 = Q(a) \vee R(u, f(y))$.

Возьмём в качестве **НОУ** $\sigma = \begin{pmatrix} x & y \\ f(y) & a \end{pmatrix}$, тогда

$[\varphi]^\sigma = P(f(y)) \vee \neg Q(a)$,

$[\psi_1]^\sigma = Q(a) \vee R(u, f(v))$.

Тогда $Res(\varphi, \psi_1) = P(f(y)) \vee R(u, f(v))$.

Теорема 28.3.(о логическом следствии).

Пусть для φ и ψ существует $Res(\varphi, \psi)$. Говорят, что φ, ψ - выполнимы на $\mathfrak{A} \Rightarrow Res(\varphi, \psi)$ - выполнима на $\mathfrak{A}$, где $\mathfrak{A} \in K(\sigma(\{\varphi, \psi\}))$.

Доказательство:

Допустим, что $Res(\varphi, \psi)$ - не выполнима на $\mathfrak{A}$. Резольвента - некоторый дизъюнкт $\Rightarrow Res(\varphi, \psi) = \xi(\overline{x})$. Раз $Res(\varphi, \psi)$ не выполнима $\Rightarrow$

$\Rightarrow \forall a_1, \ldots, a_n \in |\mathfrak{A}| : \mathfrak{A} \nvDash \xi(\overline{a})$.

В свою очередь, $\varphi = \varphi' \vee L, \psi = \psi' \vee \neg L$. Тогда $\xi = \varphi' \vee \psi'$.

Получается 2 случая: $\forall a_1, \ldots, a_n : \mathfrak{A} \vDash L(\overline{a})$, либо $\mathfrak{A} \vDash \neg L(\overline{a})$:

Сл.1. $\mathfrak{A} \vDash L(\overline{a}) \Rightarrow \mathfrak{A} \nvDash \psi(\overline{a})$, т.к. $\mathfrak{A} \nvDash \varphi'(\overline{a}) \vee \psi'(\overline{a}) \Rightarrow \mathfrak{A} \nvDash \varphi'(\overline{a})$ и $\mathfrak{A} \nvDash \psi'(\overline{a}) \Rightarrow$ получили противоречие(ψ выполнима).

Сл.2. $\mathfrak{A} \vDash \neg L(\overline{a}) \Rightarrow \mathfrak{A} \nvDash \varphi(\overline{a})$, т.к. $\mathfrak{A} \nvDash \varphi'(\overline{a}) \vee \psi'(\overline{a}) \Rightarrow \mathfrak{A} \nvDash \varphi'(\overline{a})$ и $\mathfrak{A} \nvDash \psi'(\overline{a}) \Rightarrow$ получили противоречие(φ выполнима).

Из противоречий получаем, что $Res(\varphi, \psi)$ выполнима. Теорема доказана.

§29. Полнота метода резолюций

Определение 29.1.

Пусть S - множество дизъюнктов. Говорят, что последовательность $\varphi_1, \ldots, \varphi_n = \varphi$ называется **результативным выводом** дизъюнкта φ из S, где каждая $\varphi_i \in S$ или $\varphi_i = Res(\varphi_j, \varphi_k),\ j < i, k < i, j \neq k$.

Лемма 29.2.(подъёма).

Пусть даны дизъюнкты φ, ψ. Тогда если существуют основные примеры дизъюнктов $\varphi' = [\varphi]^{x_i}_{h_i \in H_S}$ и $\psi' = [\psi]^{y_j}_{p_j \in H_S}$ такие, что существует $Res(\varphi', \psi')$, то существует и $Res(\varphi, \psi)$.

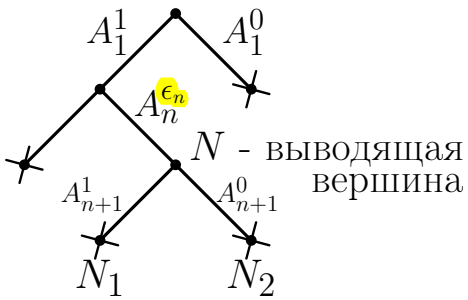
Без доказательства.

Теорема 29.3.(о полноте).

S - т.л. $\Leftrightarrow \exists$ результативный вывод $\square$.

Доказательство:

$\boxed{(\Rightarrow)}$ Т.к. S - т.л. $\Rightarrow$ $\exists$ замкнутое семантическое дерево $D(S)$. Если $D(S)$ вырождено и состоит из одной вершины, то теорема выполняется. В ином случае(если есть более одной вершины и разветвляется) рассмотрим дерево:



У вершины N есть путь $I(N) = \{A_1^{\epsilon_1}, A_2^{\epsilon_2}, \ldots, A_n^{\epsilon_n}\}$, где $\epsilon_i = \{0, 1\}$ и $A_i^0 = \neg A_i,\ A_i^1 = A_i$.

Тогда $I(N_1) = \{A_1^{\epsilon_1}, A_2^{\epsilon_2}, \ldots, A_n^{\epsilon_n}, A_{n+1}^1\}$,

$I(N_2) = \{A_1^{\epsilon_1}, A_2^{\epsilon_2}, \ldots, A_n^{\epsilon_n}, A_{n+1}^0\}$.

N_1- опровергающий $\Rightarrow \exists\varphi(\overline{x}) \in S\ \exists h_1, \ldots, h_k \in H_S \colon \mathfrak{A}_{I(N_1)} \nvDash \varphi(h_1, \ldots, h_k)$.
N_2- опровергающий $\Rightarrow \exists\psi(\overline{x}) \in S\ \exists p_1, \ldots, p_k \in H_S \colon \mathfrak{A}_{I(N_2)} \nvDash \psi(p_1, \ldots, p_k)$.

Но на вершине N выполняется $\mathfrak{A}_{I(N)} \vDash \varphi(h_1, \ldots, h_k)$ и

$\mathfrak{A}_{I(N)} \vDash \psi(p_1, \ldots, p_k)$.

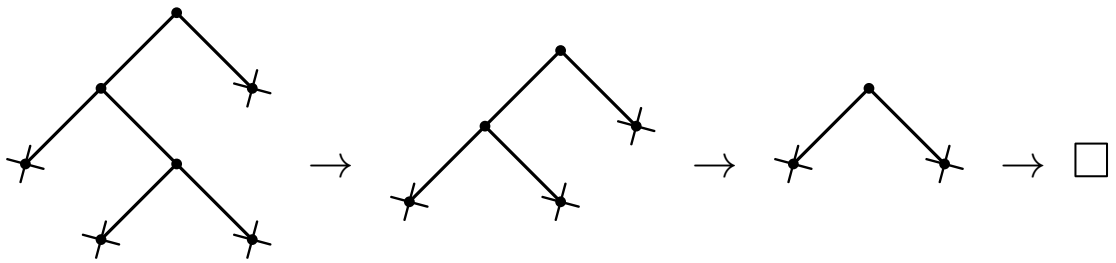
Стоит добавить в путь еще одно A_n, так дизъюнкты φ и ψ опровергаются сразу. Значит они имеют вид:

1) $\varphi = \varphi'(\overline{h}) \vee \neg A_{n+1}$ и $\mathfrak{A}_{I(N)} \nvDash \varphi'$;

2) $\psi = \psi'(\overline{p}) \vee A_{n+1}$ и $\mathfrak{A}_{I(N)} \nvDash \psi'$.

Из 1) и 2) следует, что $\mathfrak{A}_{I(N)} \nvDash \varphi'(\overline{h}) \vee \psi'(\overline{p}) \Rightarrow \mathfrak{A}_{I(N)} \nvDash Res(\varphi(\overline{h}), \psi(\overline{p})) \Rightarrow$
$\Rightarrow$ по лемме 29.2. существует $Res(\varphi, \psi)$, где переменные не из H_S.

Тогда добавим новую резольвенту в S, тогда $S' = S \cup \{Res(\varphi, \psi)\}$, т.е. добавим еще один дизъюнкт. S' стала больше, но в дереве вершина N стала опровергающей как раз из-за того, что добавили резольвенту $\Rightarrow$ мы можем выкинуть ветви, что исходили из N. Далее мы берём следующие две вершины и повторяем конечное число раз этот процесс сокращения, и в конце дерево сократится до пустого дизъюнкта, т.е. для нашего случая:



А всё множество S' как раз и является результативным выводом, где в итоге всё сводится к пустому дизъюнкту, что означает т.л. S.

$\boxed{(\Leftarrow)}$ Пусть S - выполнимо $\Rightarrow \exists\mathfrak{A}\colon \mathfrak{A} \vDash S$. Раз на модели выполняются все дизъюнкты из S, то по теореме 28.3. $\Rightarrow \forall\varphi, \psi \in S\colon \mathfrak{A} \vDash Res(\varphi, \psi)$. Но т.к. $\varphi_1, \ldots, \varphi_n = \square$ - результативный вывод $\Rightarrow \mathfrak{A} \vDash \square \Rightarrow$ получаем полный бред, т.к. $\square$ не может выполняться. Теорема доказана.

Эта теорема как раз подтверждает то, что если S - т.л., то с помощью только одного правила резолюции мы можем доказать это. Вопрос состоит только в том, когда процесс доказательства завершится.

Остаётся только вопрос стратегии доказательства.

§30. Стратегии и виды резолюции

В ходе доказательства у нас строится семантическое дерево и мы его обходим, выясняя, является ли оно замкнутым или нет. Если замкнуто, то рано или поздно с помощью метода резолюции мы это выясним и докажем т.л.. Но проблема заключается, вновь вспомним, в том, что дерево может быть бесконечным, и мы уйдем в бесконечный перебор. Здесь как раз встаёт вопрос, как искать нам по дереву: идти в глубину или ширину, используя одно или другое и т.п.. Поэтому нужны различные стратегии и алгоритмы.

У нас есть метод резолюции, как мы его будем программировать? На вход подаются дизъюнкты, мы же перебором ищем резольвенты. Рассмотрим самый простой вариант:

Метод насыщения уровня.

На вход подаётся S.

В начале ставим $S_0 = S$ и далее строим:

$$S_k = \{Res(\varphi_1, \varphi_2) \mid \varphi_1 \in S_0 \cup \cdots \cup S_{k-1},\ \varphi_2 \in S_{k-1}\}$$

И так строим, пока не наткнёмся на множество, содержащее пустой дизъюнкт.

Т.е. в начале мы выводим все возможные резольвенты в S_0, составляя S_1.

Затем строим S_2, используя резольвенты из S_1 и S_0. И так до самого конца..

Пример:

S_0		
(1)	$P \vee Q$	
(2)	$\neg P \vee Q$	
(3)	$P \vee \neg Q$	
(4)	$\neg P \vee \neg Q$	
S_1		
(5)	Q	(1)+(2)
(6)	P	(1)+(3)
(7)	$Q \vee \neg Q$	(1)+(4)
(8)	$P \vee \neg P$	(1)+(4)
(9)	$Q \vee \neg Q$	(2)+(3)
(10)	$P \vee \neg P$	(2)+(3)
(11)	$\neg P$	(2)+(4)
(12)	$\neg Q$	(3)+(4)
S_2		
(13)	$P \vee Q$	(1)+(7)
(14)	$P \vee Q$	(1)+(8)
(15)	$P \vee Q$	(1)+(9)
(16)	$P \vee Q$	(1)+(10)
(17)	Q	(1)+(11)
(18)	P	(1)+(12)
(19)	Q	(2)+(6)

(20)	$\neg P \vee Q$	(2)+(7)
(21)	$\neg P \vee Q$	(2)+(8)
(22)	$\neg P \vee Q$	(2)+(9)
(23)	$\neg P \vee Q$	(2)+(10)
(24)	$\neg P$	(2)+(12)
(25)	P	(3)+(5)
(26)	$P \vee \neg Q$	(3)+(7)
(27)	$P \vee \neg Q$	(3)+(8)
(28)	$P \vee \neg Q$	(3)+(9)
(29)	$P \vee \neg Q$	(3)+(10)
(30)	$\neg Q$	(3)+(11)
(31)	$\neg P$	(4)+(5)
(32)	$\neg Q$	(4)+(6)
(33)	$\neg P \vee \neg Q$	(4)+(7)
(34)	$\neg P \vee \neg Q$	(4)+(8)
(35)	$\neg P \vee \neg Q$	(4)+(9)
(36)	$\neg P \vee \neg Q$	(4)+(10)
(37)	Q	(5)+(7)
(38)	Q	(5)+(9)
(39)	$\square$	(5)+(12)

Если мы бы делали это сами, то заметили пустой дизъюнкт гораздо раньше, чем машина. Не трудно догадаться, что этот метод достаточно громоздкий. Здесь достаточно простое S на входе, но в итоге мы тратим очень много времени на вычисления. Что же тогда предлагается? Мы выкидываем т.и. дизъюнкты, т.е., например, $Q \vee \neg Q$ и т.п., и потом не используем их в дальнейших вычислениях.

Определение 30.1.

Дизъюнкт φ **поглощает** дизъюнкт ψ, если существует такая подстановка σ, что $[\varphi]^\sigma \subseteq \psi$.

Пример:

Пусть $\varphi(x) = P(x)$ и $\psi(x) = P(a) \vee Q(x)$.

Тогда берём $\sigma = \begin{pmatrix} x \\ a \end{pmatrix}$ и получаем $[\varphi(x)]^\sigma \subseteq \psi$, т.е. φ поглощает ψ.

Стратегия вычеркивания.

Стратегия вычеркивания заключается в вычеркивании из множества S_k дизъюнктов(из метода насыщения), которые либо являются т.и., либо поглощаются ранее выведенными дизъюнктами.

Для примера возьмём предыдущее S:

S_0		
(1)	$P \vee Q$	
(2)	$\neg P \vee Q$	
(3)	$P \vee \neg Q$	
(4)	$\neg P \vee \neg Q$	
S_1		
(5)	Q	(1)+(2)
(6)	P	(1)+(3)
(7)	$\neg P$	(2)+(4)
(8)	$\neg Q$	(3)+(4)
S_2		
(9)	$\square$	(5)+(8)

Здесь построили S_1 и сразу вычеркнули всё, что было т.и. или поглощалось, и более не используем в следующем шаге. После данной оптимизации на S_2 сразу выходит пустой дизъюнкт. Но минусом данного алгоритма будет то, что мы затрачиваем время на нахождение поглощаемых и т.и. дизъюнктов. Поэтому данная стратегия тоже не сильно пригодна.

Стратегия семантической резолюции.

Начнем с примера:

Пусть $S = \{\neg P(x) \vee \neg Q(x) \vee R(x), P(x) \vee R(x), Q(x) \vee R(x), \neg R(x)\}$.

Применим стратегию вычеркивания:

S_0		
(1)	$\neg P \vee \neg Q \vee R$	
(2)	$P \vee R$	
(3)	$Q \vee R$	
(4)	$\neg R$	
S_1		
(5)	$\neg Q \vee R$	(1)+(2)
(6)	$\neg P \vee R$	(1)+(3)
(7)	$\neg P \vee \neg Q$	(1)+(4)
(8)	P	(2)+(4)
(9)	Q	(3)+(4)
S_2		
(10)	R	(2)+(6)
(11)	$\neg Q$	(4)+(5)
(12)	$\neg P$	(4)+(6)
и т.д...		
S_3		
(13)	$\square$	(4)+(10)

Стратегию вычеркивания тоже можно улучшить. Заметим, что пустой дизъюнкт вышел из (4) и (10), (10) из (2) и (6), т.е. здесь (5), (7), (8) и (9) были "холостыми" ходами. Было бы идеально, если выполнились только ходы (10) и (6) и, соответственно, (13), т.е. выполнился только кратчайший путь.

Какая идея у семантической резолюции? Предполагается построить модель, на которой некоторые(или все) дизъюнкты из S будут истинными.

По теореме о логическом следовании если на модели выполняются дизъюнкты, то будут выполняться и их резольвенты. Если всё будет выполняться, то мы к пустому дизъюнкту никогда не придём, т.е. множество, все дизъюнкты которого выполняются на найденной модели, является выполнимым. Поэтому мы берём некоторую модель и если на ней вдруг все выполняется, то алгоритм заканчивается. Иначе же на этой модели часть дизъюнктов выполняется, а другая наоборот не выполняется.

Возьмём, например, модель, которую описывает диаграмма:

$D(\mathfrak{A}_H) = \{\neg P(a), \neg Q(a), \neg R(a), \neg P(f(a)), \dots\}$.

У нас получается, что дизъюнкты $\neg P(x) \vee \neg Q(x) \vee R(x)$ и $\neg R(x)$ будут выполняться на модели, а $P(x) \vee R(x)$ и $Q(x) \vee R(x)$ не будут. Тогда зачем нам использовать резольвенту (1) и (4), если она тоже выполнима? Если будем брать резольвенты выполнимых дизъюнктов, мы никогда не придём к пустому дизъюнкту. Тогда эту резольвенту сразу убираем. Идея семантической резолюции следующая:

S_0		
(1)	$\neg P \vee \neg Q \vee R$	
(2)	$P \vee R$	
(3)	$Q \vee R$	
(4)	$\neg R$	
S_1		
(5)	$\neg Q \vee R$	(1)+(2)
(6)	$\neg P \vee R$	(1)+(3)
(7)	$\neg P \vee \neg Q$	(1)+(4)
(8)	P	(2)+(4)
(9)	Q	(3)+(4)
S_2		
(10)	R	(2)+(6)
(11)	$\neg Q$	(4)+(5)
(12)	$\neg P$	(4)+(6)
и т.д...		
S_3		
(13)	$\square$	(4)+(10)

Мы разбиваем S на:

$S^f = \{P(x) \vee R(x), Q(x) \vee R(x)\}$;

$S^t = \{\neg P(x) \vee \neg Q(x) \vee R(x), \neg R(x)\}$ - множества невыполнимых и выполнимых дизъюнктов соответсвенно(здесь выполнимые дизъюнкты помечены красным, а ложные синим). Затем строим резольвенты, беря по одному дизъюнкты из S^f и S^t. Все резольвенты, что образованы только выполнимыми дизъюнктами, мы убираем, т.е. исключаются (7), (11) и (12) из вычислений. Но у нас остались "холостые" (8) и (9). Что с ними делать? Берём наши предикаты P, Q, R и упорядочиваем их любым способом. Упорядочим их, например, следующим образом: первый встретившийся предикат в S^t будет самым старшим, и так по убыванию, т.е. $P > Q > R$. Тогда говорим,

что брать резольвенту можем с тем истинным дизъюнктом, у которого есть старший предикат. Тогда мы можем вычеркнуть операции (8) и (9), т.к. в них применялся выполнимый дизъюнкт (4), в котором не было предиката P, а был лишь R.

Теперь мы вычеркнули все "холостые" операции и получили вполне хорошую оптимизацию. У нас вышла лишней операция (5), но она только одна..

Давайте проанализируем (5) и (6). Чтобы получить (5), мы резольвируем (1) и (2), оттуда применяем (3) и получаем (10). Чтобы получить (6), мы резольвируем (1) и (3), оттуда применяем (2) и получаем (10). Выходит, что у нас 2 одинаковые, по сути, дороги к получению (10), т.е. мы в одном случае сначала резольвируем (1) и (2), затем резольвируем с (3), а в другом случае (1) и (3), затем с (2). А что нам мешает без всяких (5) и (6) сказать, что нам хватит только (1), (2) и (3), чтобы получить (10)? То есть, если мы поймём, что нам нужны только (1), (2) и (3), то мы можем убрать (5) и (6) операции.

У нас выполняется только дизъюнкт (1), а (2) и (3) - ложные на $\mathfrak{A}_H$, что задали в самом начале. Введём следующее определение:

Определение 30.2.

Пусть даны $\mathfrak{A}_H$ и упорядочение предикатов $\mathbb{P}$. Конечное множество дизъюнктов $\{\varphi_1, \varphi_2, \ldots, \varphi_q, \psi\} (q \geqslant 1)$ называется $\mathbb{P}\mathfrak{A}_H$**-клашем**, если:

1) $\mathfrak{A}_H \not\models \varphi_1, \ldots, \mathfrak{A}_H \not\models \varphi_q$;

2) $\mathfrak{A}_H \not\models R_{q+1}$, где:

$R_1 := \psi$ и затем $\forall i \in \{1, \ldots, q\}$: $R_{i+1} := Res(R_i, \varphi_i)$ при условии, что сокращается литера, содержащая наибольший предикат в φ_i.

Дизъюнкты $\varphi_1, \varphi_2, \ldots, \varphi_q$ называются **электронами клаша**, дизъюнкт ψ - **ядро клаша**, дизъюнкт R_{q+1} - $\mathbb{P}\mathfrak{A}_H$**-резольвента клаша**.

Теорема 30.3.(о полноте семантической резолюции).

Если множество дизъюнктов S - т.л., то для любой $\mathfrak{A}_H$ и любого линейного упорядочивания предикатов $\mathbb{P}$ множества дизъюнктов S существует $\mathbb{P}\mathfrak{A}_H$-вывод пустого дизъюнкта множества S.

О чём говорит эта теорема? Мы можем выбрать любую $\mathfrak{A}_H$ - интерпретацию, любое упорядочивание, и если S - т.л., то в рано или поздно мы выведем пустой дизъюнкт. Зависит только от того, какую модель выберем. В одном случае быстрее выведется, в другом случае чуть дольше будет, но принципиально мы всегда найдём решение.

Семантическая резолюция считается хорошей стратегией, но проблема с поиском $\mathbb{P}\mathfrak{A}_H$-клашей всё равно имеется. Здесь с данной стратегией мы в чем то выигрываем, а в чём то проигрываем.

Рассмотрим следующую стратегию:

ЛОК-Резолюция.

Суть данной стратегии заключается в том, что мы индексируем не просто предикаты, что имеются в S, а индексируем все вхождения литер из S. Если происходит склейка, получившейся после склейки литере приписывается наименьший индекс литеры, что склеивалась с другой с бОльшим индексом.

Определение 30.4.

Пусть $\varphi = L_1(x_1, \ldots, x_n)_{i_1} \vee \cdots \vee L_k(x_1, \ldots, x_n)_{i_k} \vee \varphi'$. Если существует такой **НОУ**-λ такая, что:

$$[L_1(x_1, \ldots, x_n)_{i_1}]^\lambda = \cdots = [L_k(x_1, \ldots, x_n)_{i_k}]^\lambda,$$

то дизъюнктом, образовавшегося после склейки, является $[L_1(x_1, \ldots, x_n)_{\mathbb{K}}]^\lambda \vee [\varphi']^\lambda$, где $\mathbb{K} = min\{i_1, \ldots, i_k\}$, является **ЛОК-склейкой**

дизъюнкта φ.

Основная идея ЛОК-резолюции заключается в том, что мы резольвенту можем искать только по литерам, которые имеют наименьший индекс в дизъюнкте, в котором они находятся.

Определение 30.5.

Рассмотрим дизъюнкты $\varphi(x_1, \ldots, x_n)$ и $\psi(x_1, \ldots, x_n)$. Пусть литера L_1 имеет в φ наименьший индекс и литера L_2 в ψ имеет наименьший индекс. Тогда если существует такой **НОУ** σ, что $[L_1]^\sigma = [\neg L_2]^\sigma$, то **ЛОК-резольвентой** называется:

$$LRes(\varphi, \psi) = [\varphi']^\sigma \backslash [L_1]^\sigma \cup [\psi']^\sigma \backslash [L_2]^\sigma,$$

где либо $\varphi' = \varphi$ или лок-склейка дизъюнкта φ, и $\psi' = \psi$ или лок-склейка дизъюнкта ψ.

Пример:

S_0		
(1)	$P_1 \vee Q_2$	
(2)	$P_3 \vee \neg Q_4$	
(3)	$\neg P_6 \vee Q_5$	
(4)	$\neg P_8 \vee \neg Q_7$	
S_1		
(5)	$\neg P_6$	(3)+(4)
S_2		
(6)	Q_2	(1)+(5)
(7)	$\neg Q_4$	(2)+(5)
S_3		
(8)	$\square$	(6)+(7)

Возьмём те же самые 4 дизъюнкта. Мы их проиндексируем. Здесь они хитро проиндексированы, т.е. нужно тоже рассматривать отдельные стратегии индексации, т.к. от этого тоже зависит время, затраченное на доказательство. Смотрим, (1) и (2) резольвировать не можем, т.к. у Q наибольший индекс. Могут резольвироваться только (3) и (4). Далее резольвируем (1) и (5), (2) и (5), затем (6) и (7) и получаем пустой дизъюнкт. Здесь ЛОК-резолюция показала себя с лучшей стороны.

Так же доказывается теорема о полноте ЛОК-резолюции, т.е. данный метод тоже не с потолка взят. А то, как накладывать индексы на литеры - это чуть ли не целая наука.

ЛОК-резолюция очень хорошо срабатывает, если мы хорошо расстанавливаем индексы. Это одновременно плюс и минус данной стратегии. Программирование ЛОК-резолюции достаточно трудное, поэтому более интересной с точки зрения алгоритмов программирования является линейная резолюция.

Линейная резолюция.

Суть стратегии заключается в том, что мы объявляем некоторый *входной дизъюнкт* и к нему как бы "по линии" прибавляем другие дизъюнкты, с которыми можно вывести резольвенту. Затем к резольвенте ищем другой дизъюнкт и так идём до самого пустого дизъюнкта.

Определение 30.6.

Пусть дано множество дизъюнктов S с выделенным дизъюнктом $\varphi_0 \in S$. Последовательность дизъюнктов $\varphi_1, \ldots, \varphi_n$ называется **линейным выводом** дизъюнкта φ_n из множества S с верхним дизъюнктом $\varphi_0 \in S$, если для $\forall i \in \{0, \ldots, n-1\}$ имеем:

$$\varphi_{i+1} = Res(\varphi_i, \psi_i),$$

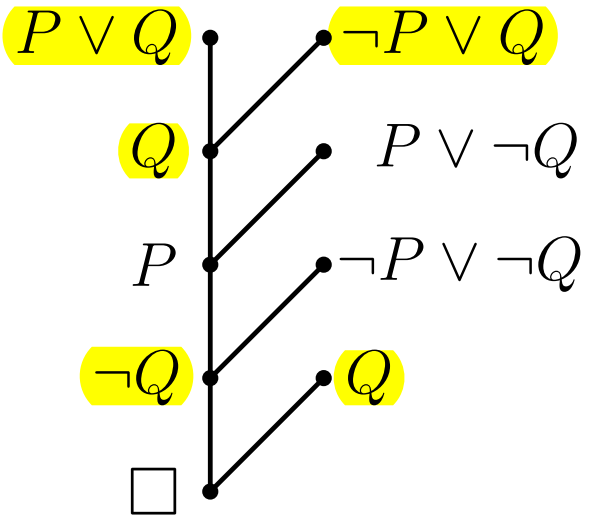
где либо $\psi_i \in S$, либо $\psi_i = \varphi_j$ для некоторого $j < i$.

Дизъюнкт φ_0 будем называть **верхним дизъюнктом**, дизъюнкты $\varphi_1, \ldots, \varphi_n$ будем называть **центральными дизъюнктами**, а дизъюнкты $\psi_1, \ldots, \psi_n$ - **боковыми дизъюнктами**.

Пример:

Пусть $S = \{P(x) \vee Q(x), \neg P(x) \vee Q(x), P(x) \vee \neg Q(x), \neg P(x) \vee \neg Q(x)\}$.

Тогда линейная резолюция строится следующим образом:



Здесь верхним дизъюнктом мы объявили $P \vee Q$ и далее ищем для него дизъюнкты, чтобы получить резольвенту. Затем к полученной резольвенте подбираем другой дизъюнкт и так до самого конца. Стоит обратить внимание, что мы на N-м шаге уже подбираем не только дизъюнкты из S для резольвирования, но и ранее образовавшиеся резольвенты.

Теорема 30.7.(о полноте линейной резолюции).

Если множество дизъюнктов S противоречиво, то *существует* линейный вывод пустого дизъюнкта из множества S.

Проблема данной стратегии заключается в том, какой дизъюнкт взять верхним. При удачном выборе мы достаточно быстро приходим к нашему ответу. В противном случае, если нам сильно не повезёт при выборе, то мы вовсе можем не получить пустой дизъюнт. Фишка теоремы о полноте линейной резолюции заключается в том, что такая ветвь существует, но

не для всех возможных верхних вершин. В теоремах о полноте предыдущих стратегий говорилось, что для любого чего-то существует вывод. Но данная стратегия немного отличается от предыдущих.

На сегодняшний день существует около 50-60 различных стратегий и все они, как правило, комбинируются в современных *резолверах*, но ЛОК-резолюция, семантическая резолюция и линейная резолюция являются являются "столпами" всех этих стратегий, и все трое были придуманы достаточно давно и на их основе строятся новые стратегии. Сейчас обычно комбинация стратегий при доказательстве зависит от самих входных данных, т.е. проверяют всякие вхождения литер и т.д.. Но всё равно все стратегии сводятся к различным методам обхода семантических деревьев.

Вся загвоздка логического программирования как раз и заключается в том, что все эти обходы неоптимальные и достатоно тяжёлые в разных случаях. Говорят, что язык Prolog сейчас зашёл в тупик как раз из-за того, что все эти стратегии и алгоритмы всё равно являются несовершенными.

§31. Язык программирования Prolog

Определение 31.1.

Пусть дан дизъюнкт $\varphi = \neg A_1(\overline{x}) \vee \cdots \vee \neg A_n(\overline{x}) \vee B_1(\overline{x}) \vee \cdots \vee B_m(\overline{x})$.

Тогда:

1) Если $n = 0, m = 1$, то φ называется **фактом**;

2) Если $n \neq 0, m = 1$, то φ называется **правилом**.

Рассмотрим подробней правило:

Пусть правило $\varphi = \neg A_1(\overline{x}) \vee \cdots \vee \neg A_n(\overline{x}) \vee B(\overline{x})$.

Мы можем вынести отрицание у $A_i(\overline{x})$ наружу, получим:

$\varphi = \neg(A_1(\overline{x})\ \&\ \ldots\ \&\ A_n(\overline{x})) \vee B(\overline{x})$.

Заменим на дизъюнкцию:

$\varphi = (A_1(\overline{x})\ \&\ \ldots\ \&\ A_n(\overline{x})) \to B(\overline{x})$.

На языке Prolog данная импликация запишется так:

```
1 B(x̄) :-A_1, ..., A_n .  /*это правило*/
2 B(x̄) .  /*это факт*/
```

Определение 31.2.

Логической программой называют конечный набор фактов и правил.

Определение 31.3.

Пусть есть логическая программа $\rho = \{C_1, \ldots, C_n\}$,

Тогда множество положительных литер $G_1(\overline{x}), \ldots, G_k(\overline{x})$ называется **запросом к логической программе** ρ.

Конъюнкция тех литер $G(\overline{x}) = G_1(\overline{x})\ \&\ \ldots\ \&\ G_k(\overline{x})$ называется **целью**, где $G_i(\overline{x})$ - **подцель**.

Что значит здесь "ответить на запрос"?

Чтобы ответить на вопрос, мы должны выяснить, выполняется ли $\rho \models \exists\overline{x}\ G(\overline{x})$. Ответом на запрос будет "выполняется ли это условие?":

1) Если это условие истинно, то надо найти такую подстановку $\theta = \begin{pmatrix} \overline{x} \\ \overline{t} \end{pmatrix}$, чтобы выполнялось $\rho \models [G(\overline{x})]^\theta$. Тогда ответом на запрос будет "да";

2) Если выясняется, что условие ложно, то получаем ответ "нет".

Теперь осталось понять, как ищется ответ на данный вопрос. Это выпол-

няется как раз с помощью резолюции.

Что значит запись $\rho \models \exists\overline{x}\ G(\overline{x})$?

Распишем подробнее:

$\{C_1, \ldots, C_n\} \models\ \exists\overline{x}\ (G_1(\overline{x})\ \&\ \ldots\ \&\ G_k(\overline{x})) \Rightarrow$

$\Rightarrow (C_1\ \&\ \ldots\ \&\ C_n) \to \exists\overline{x}\ (G_1(\overline{x})\ \&\ \ldots\ \&\ G_k(\overline{x}))$.

Если утверждение $\rho \models \exists\overline{x}\ G(\overline{x})$ верно, то формула выше является т.и.. Тогда её отрицание является т.л., т.е.

$\neg((C_1\ \&\ \ldots\ \&\ C_n) \to \exists\overline{x}\ (G_1(\overline{x})\ \&\ \ldots\ \&\ G_k(\overline{x}))) \sim$

$\sim\quad C_1\ \&\ \ldots\ \&\ C_n\ \&\ \forall\overline{x}\ (\neg G_1(\overline{x}) \vee \cdots \vee \neg G_k(\overline{x}))$.

Проверяем т.л. формулы путём проведения **скулемизации**(приведению к φ_{sko}) и построения множества дизъюнктов

$S = \{C_1', \ldots, C_n',\ \neg G_1(\overline{x}) \vee \cdots \vee \neg G_k(\overline{x})\}$, где $C_i' = C_i$ с переименованным переменными, т.к. мы выносили $\forall\overline{x}$ наружу и надо было избежать коллизий имён.

Если S - т.л., то утверждение верное и мы получаем на выходе программы "да". В ходе доказательства в классическом Prolog используется стратегия линейной резолюции.

Пример 1: (на псевдокоде)

```
Блок фактов:
Мама("Наташа","Даша"). /*здесь говорится о том, что Наташа - мама Даши*/
Мама("Даша","Маша").

Блок запросов:
Мама("Наташа", "Даша"). - "да" /*здесь мы спросили, является ли Наташа мамой Даши?*/
Мама("Наташа", "Маша"). - "нет"
Мама(x, "Даша"). - "Наташа" /*здесь мы выясняем, кто является мамой Даши*/
Мама("Наташа", y). - "Даша" /*и наоборот*/
Мама(x, y). - "Наташа - мама Даши и Даша - мама Маши" /*делаем перебор фактов*/
Мама(x, _). - "Наташа, Даша" /*выясняем, кто является мамой*/
Мама(_, y). - "Даша, Маша" /*и наоборот*/
```

Код на Prolog для примера 1:

Можете проверить работу программы здесь:)

```
mother('Natalya', 'Darya').
mother('Darya', 'Maria').

?-mother('Natalya', 'Darya'),write("1st goal is true"),nl.
?-mother('Darya', 'Natalya'),write("2st goal is true"),nl.
?-mother(X, 'Darya'),write(X),nl,fail.
?-mother('Darya', Y),write(Y),nl,fail.
?-mother(X, Y),write(X),write(" is the mother "), write(Y),nl,fail.
?-mother(X, _),write(X),nl,fail.
?-mother(_, Y),write(Y),nl,fail.
```

Пример 2:

Давайте немного усложним программу:

Введем новое правило. Давайте для фактов из 1го примера обозначим бабушку, тогда правило будет выглядеть так:

Баб(x,y) :- Мама(x,z),Мама(z,y).

И теперь мы можем спрашивать, кто является бабушкой из приведенных людей, а так же, кто является внучкой.

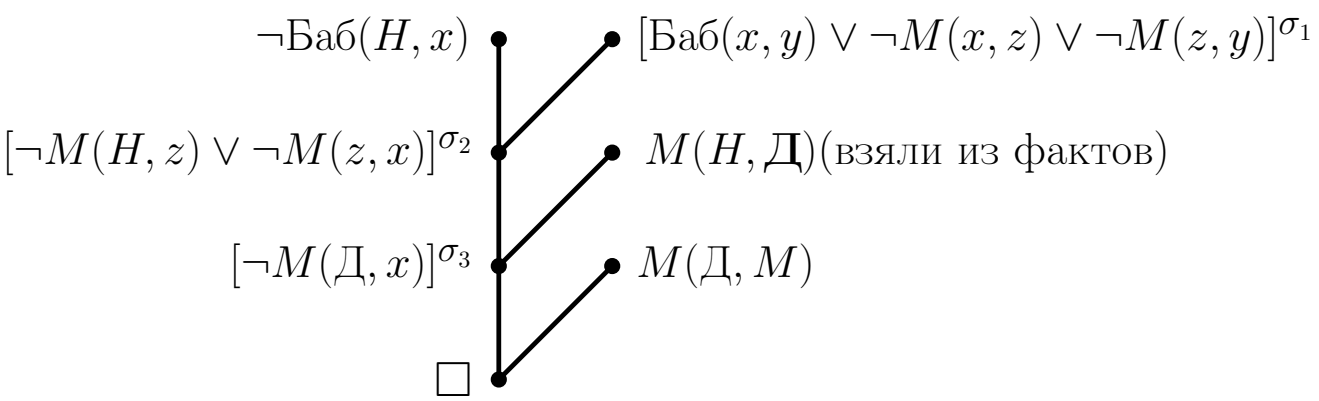
Давайте сделаем запрос:

```
?-bab('Natalya', X),write('granddaughter is '),write(X),nl,fail.
```

Что же выполняется под капотом? Здесь верхним дизъюнктом Prolog берёт ¬Баб(H, x) и далее выполняет:

Для краткой записи обозначим: Н - Наташа, Д - Даша, М - Маша и М(x, y) - Мама(x, y).

Тогда дерево будет выглядеть так:



где:

$$\sigma_1 = \begin{pmatrix} x & y \\ H & x \end{pmatrix},\ \sigma_2 = \begin{pmatrix} z \\ \text{Д} \end{pmatrix},\ \sigma_3 = \begin{pmatrix} x \\ M \end{pmatrix}.$$

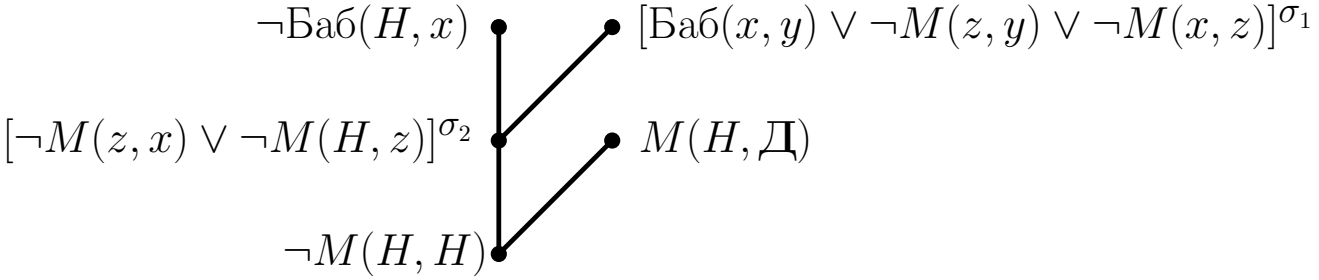
Как мы увидели, ответ существует, т.к. мы пришли к пустому дизъюнкту. Ответом на наш вопрос будет следующим:

Мы хотели узнать x, так смотрим его замены x: $x \to$ Маша(см. подстановки). Выходит, ответом на запрос будет ($x =$"Маша").

Но стоит переписать правило бабушки подобным образом:

Баб(x,y) :- Мама(z,y),Мама(x,z).

Дизъюнкт в S в итоге остаётся тем же самым, но когда исполняется программа, то расчёты изменятся(из-за алгоритма поиска **НОУ**):

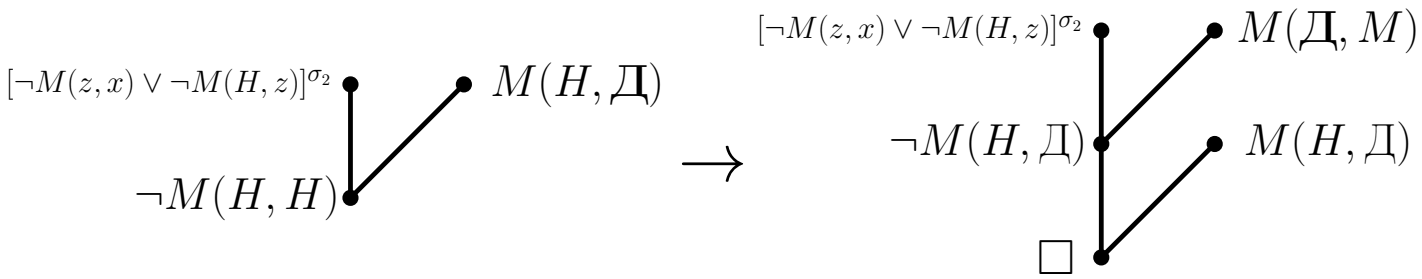


тупик...

где:

$$\sigma_1 = \begin{pmatrix} x & y \\ H & x \end{pmatrix},\ \sigma_2 = \begin{pmatrix} z & x \\ H & \text{Д} \end{pmatrix}.$$

Тупиковый вывод произошёл из-за такой подстановки - это проблема алгоритма **НОУ**. В этом случае Prolog делает откат доказательства на шаг назад и ищет уже другие варианты боковых дизъюнктов, чтобы избежать тупикового вывода, т.е.:



Здесь σ_2 изменилась на $\begin{pmatrix} z & x \\ Д & M \end{pmatrix}$. Ответ остался прежним ($x =$"Маша").

Код примера 2:

```
mother('Natalya', 'Darya').
mother('Darya', 'Maria').

bab(_X,_Y):-mother(_X,_Z),mother(_Z,_Y).

?-bab('Natalya', X),write('granddaughter is '),write(X),nl,fail.
```

В следующем примере мы рассмотрим логическое ветвление программы.

Пример 3:

Давайте попробуем посравнивать числа. Реализуем процедуру поиска максимального из 2х двух элементов.

Код поиска $max(x, y)$ на Prolog:

```
max(X,Y,X):- X>Y,!.
max(X,Y,Y).

?-max(1,5,Max),write(Max).
```

Прежде чем объяснять поведение данной программы, следует определить следующее:

$$\overbrace{max(x,y,x)}^{\text{это голова правила}} \quad :- \quad \overbrace{x>y}^{\text{это тело правила}}$$

Давайте рассмотрим строки кода 1 и 2:

Допустим, мы ввели запрос:

```
?-max(1,5,Max),write(Max).
```

Тогда: Если $x > y$(выполняется строка 1), то прерываем выполнение следующих команд(из-за оператора !) и записываем $Max = x$(см. запрос). Иначе $Max = y$. Затем печатаем после выполнения условий Max в консоль. Если бы в третий аргумент в 1й или 2й строке мы написали что-то другое, то Max приравнялся бы к ним, т.е.:

```
max(X,Y,3):- X>Y,!./*Max=3, если X>Y, и не выплняем 2ю строку*/
max(X,Y,9)./*Max=9, иначе*/

?-max(1,5,Max),write(Max)./*Если X>Y, выводим 3, иначе 9*/
```

Если бы "!"(**оператор отсечения)** не стоял после сравнения, то мы бы продолжили после 1й строки выполнять 2ю.

Но откуда 2я строка знает, какие переменные мы передавали в 1ю строку? Введём определение:

Определение 31.4.

Набор правил, у которых одинаковая голова, называется **процедурой**. Процедура пишется в одном месте(т.е. все команды стоят вместе в одной части программы) и никак иначе. Переменные с одинаковыми именами в процедуре являются общими(дикая *отсебятина*, но на практике так вышло).

Т.е. в нашем листинге строки 1 и 2 вместе образуют *процедуру*. Если не

выполнился предикат первой строки, то будет выполняться 2й в нашем случае. В итоге мы все равно получим нужный ответ.

Если бы мы хотели сравнивать не числа, а слова, то тогда бы нам пришлось вводить факты, гласящие о том, какое слово больше другого, а знак ">" в листинге примера заменили на предикат, проверяющий истинность от X и Y(см. пример 1).

Примерно вот так работает Prolog - на языке фактов, правил и запросов. Стоит отметить, что в Prolog нет циклов - есть только рекурсии.

Приятный эпилог данной главы.

Яхъяева Г.Э.:

Почему Prolog связывают с ИИ, когда речь о нём заходит? Думаю, вам известен тест Тьюринга. Суть теста заключается в том, что если машина будет общаться с нами и мы того не заметим, то машина прошла тест. Но уже данный тест машины смогли пройти.

Первой машиной, что пыталась имитировать человеческий диалог, была система Элиза. Она была изобретена в 66м году. Система имитировала диалог психотерапевта путём методики *отзеркаливания*, т.е. машине что-то писал человек, а машина задавала ему вопросы его же словами. В итоге человек сам с собой говорит-говорит, и рано или поздно он сам себя успокаивает(такой метод используется в психологии). В системе написано огромное количество правил на построение вопросов или ответов к "пациенту", так же система запоминала факты, сказанные ранее человеком, и применяла их в будущих вопросах. Так система и работала успешно. Люди тогда были огромном восторге от этой системы.

Дальнейшей ветвью развития были экспертные системы - системы, что

имитировали экспертов определённой области. Классический пример - медицинская экспертная система. Там по симптомам как раз строился диагноз. В те времена был *большой бум* логического программирования, ведь люди делали столь удивительные вещи, задавая лишь факты и правила. Prolog тогда был на взлёте и многие были уверенны, что Prolog скоро станет обыденностью. Но сейчас это всё пришло в упадок из-за того, что модель, созданная на Prolog, не может сама обучаться и приспосабливаться к новым вещам своей предметной среды. Но это могут делать нейросети, из-за чего они сейчас пользуются огромной популярностью.

Prolog нынче не сильно востребован, но до сих пор встречается где-то и, возможно, он когда нибудь воскреснет и будет его дальнейшее активное развитие. Всем же известна система $Wolfram$? Некоторая часть её кода написана на Prolog. Возможно, после *бума* нейросетей вновь вернется логическое программирование. Чем плохи нейросети? Их знания крайне тяжело интерпретировать, т.е. обучили мы модель и не можем понять, как именно она поняла освоенный набор данных для обучения. Если нейросеть будет врачём, то никто не поймёт, почему она рекомендовала такое лекарство, для нас он - черный ящик. Это не проблема, но тоже доставляет трудности при тестировании.

И если логическое программирование когда нибудь вернётся, то вы сможете в нём продолжить разбираться далее. Поэтому данные знания и данная глава так нужны в этом курсе:)

2. Лямбда-исчисление

Лямбда-исчисление важно в функциональном программировании. Давайте рассмотрим его.

Определение 32.1.

Множеством лямбда-термов Λ является:

1) $x_i \in \Lambda$ - переменная;

2) $c \in \Lambda$ - константа;

3) s, t - λ-термы, то их **комбинация** st тоже λ-терм;

4) s - λ-терм, то $\lambda x.s$ - λ-терм.

Здесь λ перед x - **абстрактор**, а процесс *навешивания* λ на x - **абстракция**.

Пример 1:

Пусть x - переменная. Тогда:

1) xx - терм;

2) $\lambda x.xx$ - терм.

Так же принято считать, что кортеж $\overline{x} = x_1, \ldots, x_n$.

Тогда $\lambda\overline{x}.s = \lambda x_1 \ldots x_n.s = \underbrace{\lambda x_1.(\lambda x_2.(\lambda x_3 \ldots (\lambda x_n.s) \ldots)}_{n \text{ раз}}$ - так же является термом.

Обозначим $\overline{s} = s_1 \ldots s_n$.

Тогда $t\overline{s} = ts_1 \ldots s_n = \underbrace{(\ldots(ts_1)s_2) \ldots s_{n-1})s_n}_{n \text{ раз}}$ - терм.

Пример 2:

Пусть дан λ-терм $s = \lambda xy.yx(\lambda z.z)$.

Здесь подразумевается, что действие лямбды λxy **распространено** на

всю правую часть выражения.

Так же s можно переписать так: $s = \lambda x \lambda y.yx(\lambda z.z) = \lambda x.(\lambda y.(yx)(\lambda z.z))$.

А если мы расставим по другому скобки?

Рассмотрим $s_1 = (\lambda xy.yx)\lambda z.z = (\lambda x \lambda y.yx)\lambda z.z = (\lambda x.(\lambda y.yx))\lambda z.z$.

Здесь уже действие лямбды λxy ограничено скобками, в которых она лежит.

Если заметить, то здесь действие абстрактора во многом похоже на действие кванторов из логики предикатов.

Откуда вообще пошло λ-исчисление? Буква λ была взята без особого смысла, просто так исторически сложилось. По факту это **исчисление функций**. Здесь у нас понятия формулы как таковой не существует, существуют только функции и термы.

Изначально формулу с аргументами мы писали как $t(x)$. Алонзо Чёрч же предложил такую запись $\hat{x}.t$, что обозначало вычисление t при подстановке переменных x. Когда начали печатать книги, они не смогли напечать $\hat{x}$ и просто печатали $\wedge x$. Позже это медленно перешло в λx. Поэтому эти исчисления назвали λ-исчислениями.

В чем основная идея λ-исчисления? Идея заключается в том, чтобы свести все выражения к *функтам*, т.е. сводить все функции в примитивную запись в строку(в чём-то аналогия с польской обратной нотацией, вспоминай калькулятор). С точки зрения программирования, этот синтаксис легко обрабатывается и считается компьютером. Можно почитать про это еще здесь, хорошо дополняет понимание этого чудо-аппарата.

Определение 32.2.

Переменная x **входит свободно** в t, если не находится в области действия абстрактора λx.

Переменная x **входит связанно** в t, если находится в области действия абстрактора λx.

Так же определяются **свободная** и **связанные** переменные.

Определение 32.3.

Множеством свободных переменных терма s $FV(s)$ обозначают:

1) $s = x$: $FV(x) = \{x\}$;

2) $s = c$: $FV(c) = \varnothing$;

3) $s = t_1 t_2$: $FV(s) = FV(t_1) \cup FV(t_2)$;

4) $s = \lambda x.t$: $FV(\lambda x.t) = FV(t) \setminus \{x\}$.

Определение 32.4.

Множеством связанных переменных терма s $BV(s)$ обозначают:

1) $s = x$: $BV(x) = \varnothing$;

2) $s = c$: $BV(c) = \varnothing$;

3) $s = t_1 t_2$: $BV(s) = BV(t_1) \cup BV(t_2)$;

4) $s = \lambda x.t$: $BV(\lambda x.t) = BV(t) \cup \{x\}$.

Определение 32.5.

λ-терм, не имеющий свободных переменных, называется **замкнутым термом** или **комбинатором**.

Замечание 32.6.

Для $\forall\lambda$-терма t: $BV(t)$ и $FV(t)$ - конечны.

Пример:

Пусть $s = (\lambda xy.x)(\lambda x.zx)$, тогда $FV(s) = \{z\},\ BV(s) = \{x, y\}$.

§33. Подстановки

Определение 33.1.

Запись вида $s[x := t]$ будем называть **подстановкой** терма t вместо x в терме s.

Правила подстановки следующие:

1) $x[x := t] = t$;

2) $y[x := t] = y$, если $y \neq x$, т.е. тут пытаемся поменять то, чего нет в терме;

3) $c[x := t] = c$;

4) $s_1 s_2[x := t] = s_1[x := t]s_2[x := t]$;

5) $\lambda x.s[x := t] = \lambda x.s$, т.е. нельзя заменить связную переменную;

6) $(\lambda y.s)[x := t] = \lambda y.(s[x := t])$, если $x \neq y$ и $(x \notin FV(s)$ или $y \notin FV(t))$.

В случае 6, если бы мы не поставили такое громоздкое условие, могли бы образоваться захваты переменных, т.е., например, $\lambda y.xy[x := y] = \lambda y.yy$. Тогда такая замена могла *сломать* выражение. Поэтому ставится условие непринадлежности к свободным переменным терма. Грубо говоря, это правило подразумевает, что если мы и хотим заменить на переменную, чьё имя уже занято в терме, то её стоит переименовать.

Или же здесь, чтобы избежать захвата, можно переименовать переменную в терме, на которую хотим заменить другую переменную, т.е.

$$(\lambda y.s)[x := t] = \lambda z.((s[y := z])[x := t]),\ z \notin FV(s) \cup FV(t).$$

По факту, вся эта *суматоха* с переменными только из-за того, чтобы избежать их *коллизий*.

Так же важно понимать, что мы можем заменять только переменные на термы.

§34. Редукция лямбда-исчисления

Над термами мы можем проводить преобразования, что во многом похожи на *эквивалентные преобразования* в логике высказываний.

Определение 34.1.

Всего существует 3 главные преобразования(редукции):

1) α-**преобразование**: $\lambda x.s \underset{\alpha \text{ преоб.}}{\longrightarrow} \lambda y.s[x := y],\ y \notin FV(s),\ x \sim y$;

2) β-**преобразование**: $(\lambda x.s)t \underset{\beta \text{ преоб.}}{\longrightarrow} s[x := t]$;

3) η-**преобразование**: $\lambda x.sx \underset{\eta \text{ преоб.}}{\longrightarrow} s$.

Чтобы лучше понять 2е преобразование, можно привести следующий пример: пусть есть некоторое исчисление, где есть сложение элементов и некоторые константы. Тогда например:

$$(\lambda x.x + x)2 \underset{\beta}{\longrightarrow} 2 + 2 = 4$$

Пример:

$\lambda x.(\lambda z.zx) \underset{\alpha}{\longrightarrow} \lambda y.(\lambda z.zy)$;

$\lambda x.zyx \underset{\eta}{\longrightarrow} zy$.

Приведём несколько примеров β-редукции:

1) $(\lambda x.xy)(\lambda z.zx) \underset{\beta}{\longrightarrow} (\lambda z.zx)y$;

2) $(\lambda x.\underbrace{xxx}_{s})\underbrace{(\lambda x.xxx)}_{t} \underset{\beta}{\longrightarrow} (\lambda x.\underbrace{xxx}_{s})\underbrace{(\lambda x.xxx)}_{t}(\lambda x.xxx) \underset{\beta}{\longrightarrow}$

$\underset{\beta}{\longrightarrow} (\lambda x.xxx)(\lambda x.xxx)(\lambda x.xxx)(\lambda x.xxx)$;

И так этот терм можно редуцировать до бесконечности.

3) $(\lambda x.x)a \underset{\beta}{\longrightarrow} a$;

4) $(\lambda x.y)a \underset{\beta}{\longrightarrow} y$;

5) $(\lambda x.\underbrace{xx}_{s})\underbrace{(\lambda y.y)}_{t}z \underset{\beta}{\longrightarrow} (\lambda y.y)(\lambda y.y)z \underset{\beta}{\longrightarrow} (\lambda y.y)z \underset{\beta}{\longrightarrow} z.$

Теорема 34.2.

$(\lambda\overline{x}.s)\overline{x} \underset{\beta}{\longrightarrow} s$, где $\overline{x} = x_1 \dots x_n$.

Доказательство:

Индукция по n:

$\underline{n=1}$: $(\lambda x.s)x \underset{\beta}{\longrightarrow} s[x := x] = s$;

$\underline{n=2}$: $(\lambda x_1 x_2.s)x_1x_2 = ((\lambda x_1.(\lambda x_2.s))x_1)x_2 \underset{\beta}{\longrightarrow} (\lambda x_2.s[x_1 := x_1])x_2 =$

$= (\lambda x_2.s)x_2 \underset{\beta}{\longrightarrow} s[x_2 := x_2] = s$;

$\underline{< n \to n}$: $(\lambda\overline{x}_{n+1}.s)\overline{x}_{n+1} = \underbrace{(\dots(\lambda x_1.(\dots(\lambda x_n.(\lambda x_{n+1}.s))x_1)\dots)x_{n+1}}_{n+1 \text{ раз}} \underset{\beta}{\longrightarrow}$

$\underset{\beta}{\longrightarrow} \underbrace{(\dots(\lambda x_2.(\dots(\lambda x_n.(\lambda x_{n+1}.s[x_1 := x_1]))x_2)\dots)x_{n+1}}_{n \text{ раз}} \xrightarrow[n\ \beta\text{-преобр-й}]{}$

$\xrightarrow[n\ \beta\text{-преобр-й}]{} s[x_{n+1} := x_{n+1}] = s.$

Теорема доказана.

Таким образом, в программировании самым важным является β-преобразование, т.к. α- и η-преобразования являются по сути техническими и используются, как правило, только в доказательствах.

Определение 34.3.(отношение редукции).

Говорят, что из терма s **редуцируется** терм t $(s \to t) \Leftrightarrow$

$\Leftrightarrow \exists u_1, \dots, u_n\colon\ s \underset{\epsilon_1}{\longrightarrow} u_1 \underset{\epsilon_2}{\longrightarrow} u_2 \underset{\epsilon_3}{\longrightarrow} \dots \underset{\epsilon_n}{\longrightarrow} u_n \underset{\epsilon_{n+1}}{\longrightarrow} t$, где $\epsilon_i \in \{\alpha, \beta, \eta\}$.

Предложение 34.4.(свойства редукции).

Для $\forall s, t, u \in \Lambda$:

1) $t \to t$;

2) ($s \to t$ и $t \to u$) $\Rightarrow s \to u$;

3) $s \to t \Rightarrow su \to tu$;

4) $s \to t \Rightarrow us \to ut$;

5) $s \to t \Rightarrow \lambda x.s \to \lambda x.t$.

Определение 34.5.

Говорят, что терм s **подобен** терму t $(s \sim t) \Leftrightarrow s \to t$ или $t \to s$.

Причём подобие может задаваться и так:

$$\begin{matrix} s_1 \to t \\ s_2 \to t \end{matrix} \Rightarrow \begin{matrix} s_1 \sim t \\ s_2 \sim t \end{matrix} \Rightarrow s_1 \sim s_2.$$

Определение 34.6.

λ-терм t имеет **нормальную форму**, если к нему применимо либо только α-преобразование, либо вообще ничего нельзя применить.

Основная цель наших преобразований состоит в том, чтобы привести терм к **н.ф.**.

Пример:

Пусть дан терм $s = (\lambda x.y)((\lambda x.xxx)(\lambda x.xxx))$.

Здесь всё зависит от того, как мы будем преобразовывать. Если мы станем преобразовывать внутренние правые скобки в s, то терм только расширится и мы не сможем так получить **н.ф.**.

Но можем сделать преобразование следующим образом:

$(\lambda x.\underbrace{y}_{s})\underbrace{((\lambda x.xxx)(\lambda x.xxx))}_{t} \underset{\beta}{\longrightarrow} y.$

Тогда мы получим **н.ф.**, т.к y мы уже ничего сделать не можем.

Наша задача - преобразовывать терм так, чтобы получать **н.ф.**, т.к. применять β-редукцию мы можем разными способами.

Теорема 34.7.

Если $s \to t$ и t имеет **н.ф.**, то последовательность редукций, что начинается с терма s и состоит в применении правила редукции внешнему(самому левому) редексу, всегда завершается и приводит терм s к **н.ф.**.

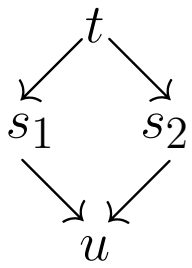
Без доказательства.

Теорема 34.8.(Чёрча-Россера или правило ромба).

Если $t \to s_1$ и $t \to s_2 \Rightarrow \exists u\colon s_1 \to u$ и $s_2 \to u$.

Без доказательства.

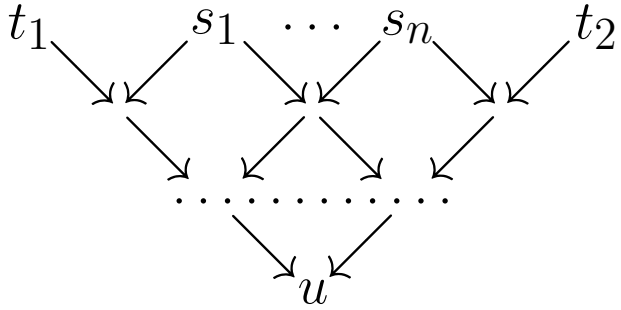
Если показать графически эту теорему, то она будет выглядеть так:



Следствие 34.9.

$t_1 \sim t_2 \Rightarrow \exists u\colon t_1 \to u$ и $t_2 \to u$.

Графически это следствие будет выглядеть так:



Давайте вспомним, как строится подобие термов? Либо из t_1 редуцируется t_2, либо наоборот. Что значит редуцируется? Значит, что есть набор термов, что являются *шагами* редукции одного t к другому. Этим набором как раз является $s_1, \ldots, s_n$. Применяем к $t_1, t_2, s_1, \ldots, s_n$ конечное число раз *правило ромба* и получаем такую пирамидку вывода.

Следствие 34.10.

Если $t \sim t_1$ и $t \sim t_2$, t_1 и t_2 имеют **н.ф.**, то t_1 и t_2 **равны с точностью до α-преобразования**, т.е. с точностью до переименования переменных.

Доказательство:

$(t \sim t_1$ и $t \sim t_2) \to t_1 \sim t_2 \Rightarrow \exists u\colon t_1 \underset{\alpha}{\longrightarrow} u$ и $t_2 \underset{\alpha}{\longrightarrow} u$, поскольку t_1 и t_2 имеют **н.ф.**, т.е. их можно только переименовать $\Rightarrow$ t_1 и t_2 равны с точностью до α-преобразования.

Следствие доказано.

О чём эти следствия говорят? О том, что если у терма и есть **н.ф.**, то она единственная.

§35. Комбинаторы

Еще раз вспомним, что **комбинатором** называется терм без свободных переменных, т.е. все переменные связаны абстракциями. С логической точки зрения, для программирования вполне достаточно 3х комбинаторов, т.е. из них можно всё вывести. Остальные же комбинаторы используются для каких-либо иных целей.

Определение 35.1.

3 основными комбинаторами являются:

1) $\mathcal{I} := \lambda x.x$;

2) $\mathcal{K} := \lambda xy.x$;

3) $\mathcal{S} := \lambda xyz.(xz)(yz)$.

Предложение 35.2.

$\forall s, t, u \in \Lambda$:

1) $\mathcal{I}s \sim s$;

2) $\mathcal{K}st \sim s$;

3) $\mathcal{S}stu \sim (su)(tu)$.

Доказательство:

Применяем β-редукцию и успешно сокращаем.

Предложение доказано.

Теорема 35.3.

Для $\forall\lambda$-терма t, не содержащего λ-абстракции, существует λ-терм u, который так же не содержит λ-абстракции и представляется собой композицию комбинаторов $\mathcal{I}, \mathcal{K}, \mathcal{S}$ и переменных и $u \sim \lambda x.t$.

Без доказательства.

Следствие 35.4.

Для $\forall\lambda$-терма t $\exists t'$: t' - композиция $\mathcal{I}, \mathcal{K}$ и переменных такая, что $FV(t') = FV(t)$ и $t' \sim t$.

Без доказательства.

В чём идея следствия? Пусть, допустим, у нас существует огромный терм,

но мы его можем преобразовать в такой вид, что в нём будут составляющие только того вида, что обозначены в следствии. На этом обзор λ-исчисления окончен.

3. Модальная логика

Историки считают, что модальностью еще начинал заниматься Аристотель, однако строго модальная логика возникла только в прошлом столетии и получила довольно широкое приложение.

Например, у нас есть логика высказываний, а там есть высказывания, предложения, в которых можем утверждать "*истинно ли что-то или ложно?*".

Возьмём высказывание "*Александр Македонский завтра выиграет сражение*". Истинно ли это? Когда завтра наступит, мы узнаем, но пока нам неизвестен исход. Но мы можем предположить два исхода: "*он выиграет*" или "*он проиграет*". Мы можем рассмотреть все возможные миры, где как раз получаем ответ на данный вопрос: Макендонский победит или проиграет. И мы говорим, что "возможно" он победит - это значит, что существует тот мир, где данное событие произошло, или наоборот. А можем сказать, что он "необходимо" победит, т.е. во всех возможных мирах он побеждает в данном событии, или наоборот. Т.е. здесь следующий день рассматриваем как некоторое множество миров, где может что-то произойти. В данном ключе мы будем рассуждать всю следующую тему.

"Необходимое событие" или "возможное событие" - это и является **модальностью**.

Сначала мы будем рассматривать пропозициональную модальную логику, где нет никаких кванторов.

Определение 36.1.

Обозначим набор пропозициональных переменных как множество

$$\sigma = \{A_1, A_2, \dots\}.$$

Данное множество может быть конечным или бесконечным.

Так же обозначим набор модальностей

$$MOD = \{m_1, m_2, \dots\}.$$

Существуют разные модальности, и пара (σ, MOD) задаёт **модальную логику**.

Дополнительное объяснение модальной логики находится здесь. Далее введём понятие формулы.

Определение 36.2.

Формулой модальной логики (σ, MOD) являются:

1) $\forall A_i \in \sigma$ - формула;

2) φ, ψ - формулы $\Rightarrow (\varphi \,\&\, \psi),\ (\varphi \vee \psi),\ (\varphi \to \psi),\ \neg\varphi,\ \neg\psi$ - формулы;

3) φ - формула и m - модальность, то $\langle m\rangle\varphi, [m]\varphi$ - формулы;

4) Других формул нет.

Определение 36.3.

Если $||MOD|| = 1$, то (σ, MOD) является **унимодальной логикой**, иначе **полимодальной логикой**.

Мы будем рассматривать только унимодальную логику. Т.к. у нас только одна модальность, то то формулы с модальностями будем писать так:

$\langle m\rangle\varphi \Rightarrow \Diamond\varphi$(возможно φ);

$[m]\varphi \Rightarrow \Box\varphi$(необходимо φ).

В области искуственного интеллекта используется логика описания(description logic), что тоже является полимодальной логикой. Т.е. это всё же имеет смысл изучать.

Определение 36.4.

Подмножество пропозициональных переменных $w \subseteq \sigma$ является **миром**. Всего всевозможных миров в унимодальной логике равно $2^{||\sigma||}$.

Определение 36.5.

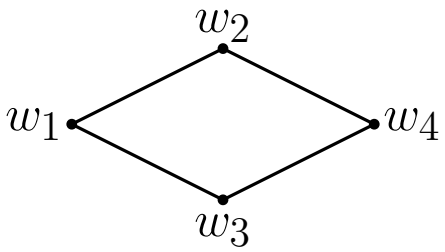
Структурой Крипке является пара $\langle W, R \rangle$, где:

W - множество миров;

Бинарное отношение $R \subseteq W^2$ является **отношением достижимости**.

Пример:

Покажем пример структуры Крипке на примере с Македонским.



Здесь мы находимся в мире w_1 и рассуждаем, что произойдёт завтра. И у нас есть два исхода: победа(в w_2) или поражение(в w_3). А потом уже, если мы в какой-то мир из w_2 или w_3 в итоге попали, то дальше, предположим, эти исходы слились и на следующий день он уже гарантированно победил. А может наоборот случиться и события ещё сильнее разветвлятся, т.е. появится еще больше различных миров. Здесь отношение R - как раз соединения миров.

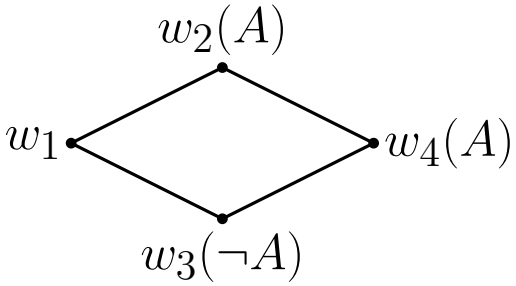
Определение 36.6.

Моделью Крипке будем обозначать набор $\mathfrak{M} = \langle W, R, V \rangle$, где $\langle W, R \rangle$ - структура Крипке и V: $\sigma \to W$ - означивание, где

$$\forall A_i \in \sigma\colon\ V(A_i) = \{w \in W \mid A_i \in w\}.$$

Здесь уже, помимо информации о том, какие есть миры и как они связаны, так же появляется некоторая информация об этих мирах, где какие истинные переменные.

Еще раз рассмотрим пример с Македонским:



Здесь A - победа Македонского и $V(A) = \{w_2, w_4\}$. Допустим, он вновь победил в w_4.

Определение 36.7.

Пусть есть модель Крипке $\mathfrak{M} = \langle W, R, V \rangle$ и $w \in W$. Тогда пара $(\mathfrak{M}, w)$ называется **структурой мира** w в $\mathfrak{M}$.

Определение 36.8.

Пусть дана модель Крипке $\mathfrak{M} = \langle W, R, V \rangle$ и $w \in W$.

Определим **истинность формулы на модели Крипке**:

1) $\mathfrak{M}, w \vDash A_i \Leftrightarrow w \in V(A_i)$;

2) $\mathfrak{M}, w \vDash \varphi_1 \,\&\, \varphi_2 \Leftrightarrow \mathfrak{M}, w \vDash \varphi_1$ и $\mathfrak{M}, w \vDash \varphi_2$;

3) $\mathfrak{M}, w \vDash \varphi_1 \vee \varphi_2 \Leftrightarrow \mathfrak{M}, w \vDash \varphi_1$ или $\mathfrak{M}, w \vDash \varphi_2$;

4) $\mathfrak{M}, w \vDash (\varphi_1 \to \varphi_2) \Leftrightarrow \mathfrak{M}, w \nvDash \varphi_1$ или $\mathfrak{M}, w \vDash \varphi_2$;

5) $\mathfrak{M}, w \vDash \neg\varphi \Leftrightarrow \mathfrak{M}, w \nvDash \varphi$;

6) $\mathfrak{M}, w \vDash \Diamond\varphi \Leftrightarrow \exists v \in W$: $((w, v) \in R$ и $\mathfrak{M}, v \vDash \varphi)$;

7) $\mathfrak{M}, w \vDash \Box\varphi \Leftrightarrow \forall v \in W\colon ((w,v) \in R \Rightarrow \mathfrak{M}, v \vDash \varphi)$.

В 7м пункте важно учесть, что "$\Rightarrow$" сопоставима с импликацией.

Определение 36.9.

$\mathfrak{M} \vDash \varphi \Leftrightarrow \varphi$ истинно в каждом мире из модели $\mathfrak{M}$.

Формула φ истинна на структуре Крипке, если она истинна на любой модели с данной структурой.

Предложение 36.10.

φ - т.и. в логике высказываний $\Leftrightarrow \varphi$ - т.и. в модальной логике.

Предложение 36.11.

Следующие формулы т.и. в модальной логике:

1) $\Box(A \to B) \to (\Box A \to \Box B)$;

2) $\Box(A \,\&\, B) \leftrightarrow (\Box A \,\&\, \Box B)$;

3) $\Diamond(A \vee B) \leftrightarrow (\Diamond A \vee \Diamond B)$;

4) $(A \to \Diamond(B \to C)) \leftrightarrow \Diamond(B \to (A \to \Diamond C))$.

Доказательство:

1) Докажем от противного.

Пусть $\exists\mathfrak{M}\ \exists w\colon\ \mathfrak{M}, w \nvDash (\Box(A \to B) \to (\Box A \to \Box B)) \Leftrightarrow$

$\Leftrightarrow \mathfrak{M}, w \vDash \Box(A \to B)$ и $\mathfrak{M}, w \vDash \neg(\Box A \to \Box B) \Leftrightarrow$

$\Leftrightarrow \mathfrak{M}, w \vDash \Box(A \to B)$ и $\mathfrak{M}, w \vDash (\Box A \,\&\, \neg\Box B) \Leftrightarrow$

$\Leftrightarrow \mathfrak{M}, w \vDash \Box(A \to B)$ и $\mathfrak{M}, w \vDash \Box A$ и $\mathfrak{M}, w \vDash \neg\Box B \Leftrightarrow$

$\Leftrightarrow \forall v \in W\colon (w,v) \in R \Rightarrow \mathfrak{M}, v \vDash A \to B \Leftrightarrow$

$\Leftrightarrow \forall v \in W\colon (w,v) \in R \Rightarrow \mathfrak{M}, v \vDash A,\ \mathfrak{M}, w \vDash \neg\Box B \Rightarrow$

$\Rightarrow \forall v \in W\colon (w,v) \in R \Rightarrow \mathfrak{M}, v \vDash B$ и $\mathfrak{M}, w \vDash \neg\Box B \Rightarrow$

$\Rightarrow \mathfrak{M}, w \vDash \Box B$ и $\mathfrak{M}, w \vDash \neg\Box B$. Получаем противоречие $\Rightarrow$ 1я формула т.и.

Остальное по похожей схеме...

Предложение доказано.

Определение 36.12.

φ - выполнима на $\mathfrak{M} \Leftrightarrow \exists w \in W\colon\ \mathfrak{M}, w \vDash \varphi$.

Предложение 36.13.

Следующие формулы выполнимы:

1) $\Box A \to A$;

2) $\Box A \to \Box\Box A$;

3) $\Box(A \vee B) \to (\Box A \to \Box B)$.

Определение 36.14.

$\varphi \sim \psi$(формулы φ и ψ эквивалентны) $\Leftrightarrow \forall \mathfrak{M} = \langle W, R, V\rangle,\ \forall w \in W$: : $\mathfrak{M}, w \vDash \varphi \Leftrightarrow \mathfrak{M}, w \vDash \psi$.

Предложение 36.15.

Следующие формулы эквивалентны:

1) $\Diamond\varphi \sim \neg\Box\neg\varphi$;

2) $\Box\varphi \sim \neg\Diamond\neg\varphi$.

§37. Бисимуляция

Определение 37.1.

Пусть есть модели $\mathfrak{M} = \langle W, R, V\rangle$ и $\mathfrak{M}' = \langle W', R', V'\rangle$, $w \in W$, $w' \in W'$. Тогда:

1) $w \equiv w'$(**атомарно эквивалентны**) $\Leftrightarrow$

$\Leftrightarrow \forall A \in \sigma\colon\ \mathfrak{M}, w \vDash A \Leftrightarrow \mathfrak{M}', w' \vDash A$;

2) $w \approx w'$(**модально эквивалентны**) $\Leftrightarrow$

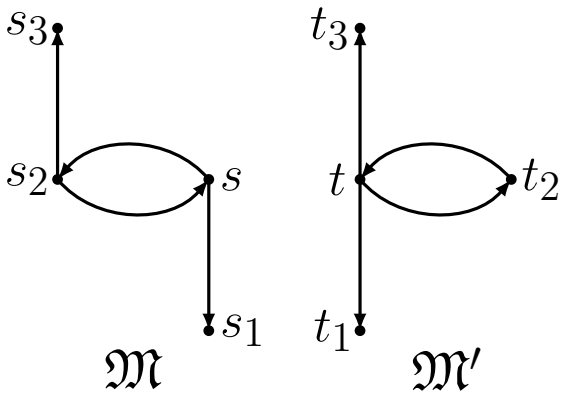
$\Leftrightarrow \forall \varphi \in F(\sigma)\colon\ \mathfrak{M}, w \vDash \varphi \Leftrightarrow \mathfrak{M}', w' \vDash \varphi$.

Стоит отметить, что из атомарной эквивалентности модальная эквивалентность не следует.

Если у нас есть одинаковые миры, то это еще не значит, что на них сработает любая формула.

Пример 1:

Рассмотрим миры s и t в моделях $\mathfrak{M}, \mathfrak{M}'$ соответственно:



Пусть $s \equiv t$. Рассмотрим формулу $\varphi = \Box(\Box 0 \vee \Diamond\Box 0)$, где 0 - т.л. формула.

Если $\mathfrak{M}, s \vDash \varphi \Rightarrow \mathfrak{M}, s_1 \vDash \Box 0 \vee \Diamond\Box 0$ и $\mathfrak{M}, s_2 \vDash \Box 0 \vee \Diamond\Box 0 \Leftrightarrow$

$$\Leftrightarrow \begin{cases} \mathfrak{M}, s_1 \vDash \Box 0 \text{ или } \mathfrak{M}, s_1 \vDash \Diamond\Box 0 \\ \mathfrak{M}, s_2 \vDash \Box 0 \text{ или } \mathfrak{M}, s_2 \vDash \Diamond\Box 0 \end{cases}$$

Давайте рассмотрим это подробней. У нас получается, что в системе в мире s_1 выполнится только формула $\Box 0$, поскольку нет выходящих из s_1 миров, тогда импликация для истинности необходимости выполняется(см. истинности модальностей). Для s_2 выполнится $\Diamond\Box 0 \Rightarrow$ на s истинно φ.

Рассмотрим для t:

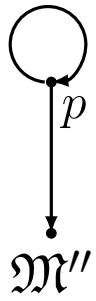
$$\mathfrak{M}', t \vDash \varphi \Leftrightarrow \begin{cases} \mathfrak{M}', t_1 \vDash \Box 0 \text{ или } \mathfrak{M}', t_1 \vDash \Diamond\Box 0 \\ \mathfrak{M}', t_2 \vDash \Box 0 \text{ или } \mathfrak{M}', t_2 \vDash \Diamond\Box 0 \\ \mathfrak{M}', t_3 \vDash \Box 0 \text{ или } \mathfrak{M}', t_3 \vDash \Diamond\Box 0 \end{cases}$$

В t_1 и t_3 всё выполнится, но в t_2 - нет $\Rightarrow$ внешняя необходимость не выполнится для мира $t \Rightarrow s \not\approx t$.

Т.е. нам и не потребовалось знать, какие формулы истинны на s или t, мы просто взяли т.л. формулу и через неё показали, что из атомарной эквивалетности не следует модальная эквивалентность.

Пример 2:

Построим модель с миром p:



Здесь $s \equiv p$ и $\mathfrak{M}'', p \vDash \varphi$. Выходит, несмотря на различные модели $\mathfrak{M}$ и $\mathfrak{M}''$, миры модально и атомарно эквивалентны. Так же мы можем сказать, что модели бисимулируют, т.е. у них есть *эквивалентные* миры.

Осталось записать определение *бисимуляции*.

Определение 37.2.

Пусть даны модели $\mathfrak{M} = \langle W, R, V\rangle$ и $\mathfrak{M}' = \langle W', R', V'\rangle$. Отношение $E \subseteq W \times W'$ называется **бисимуляцией между моделями $\mathfrak{M}$ и $\mathfrak{M}'$**, $E \neq \varnothing$, если для $\forall w \in W\ \forall w' \in W'$: $(w, w') \in E$ выполняется только тогда, когда выполняются:

1) $w \equiv w'$ - *атомарная гармония*;

2) "*Зиг*":

$\forall v \in W$: $((w, v) \in R \Rightarrow \exists v' \in W'$: $((w', v') \in R'$ и $(v, v') \in E))$;

3) "*Заг*":

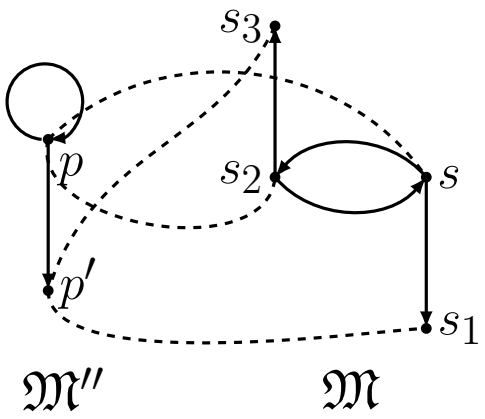
$\forall v' \in W'$: $((w', v') \in R' \Rightarrow \exists v \in W$: $((w, v) \in R$ и $(v, v') \in E))$.

Здесь так же важно учесть, что "$\Rightarrow$" сопоставима с импликацией.

Т.е. что эти условия значат? Если для каждого мира w из W найдётся такой мир w' из W', с которым он атомарно эквивалентен(*атомарная гармония*) и при этом для всех миров, выходящих из w, найдутся выходящие миры из w', с которыми они будут совпадать(*Зиг*), а так же для всех миров, выходящих из w', найдутся выходящие миры из w, с которыми они будут совпадать(*Заг*).

Пример:

Давайте рассмотрим бисимуляцию на нашем предыдущем примере:



Здесь пунктиром как раз выступает E. Рассмотрим миры p, s_2 и s. О чём говорит бисимуляция? Если из p выходит "тупиковый" мир, то и из $s2$ и s тоже должны выходить тупиковые. Выходит, что p' "совпадает" с s_1 и s_3. Сами же миры p, s и s_2 совпадают, т.к. s и s_2 "схлопываются" в один мир, поскольку существуют отношения (p, s_2) и (p, s). Так же s_3 и s_1 "схлопываются" в p'. Отсюда выходит, что модели $\mathfrak{M}$ и $\mathfrak{M}''$ бисимулируют.

Определение 37.3.

Будем говорить, что модели **бисимулируют**, т.е. $\mathfrak{M} \rightleftarrows \mathfrak{M}' \Leftrightarrow \exists$ бисимуляция E.

Структура мира w бисимулирует со структурой мира w', т.е.
$(\mathfrak{M}, w) \rightleftarrows (\mathfrak{M}', w') \Leftrightarrow \exists$ бисимуляция $E\colon (w, w') \in E$.

Теорема 37.4.

Пусть есть модели $\mathfrak{M} = \langle W, R, V\rangle$ и $\mathfrak{M}' = \langle W', R', V'\rangle$, $w \in W$, $w' \in W'$. Тогда $(\mathfrak{M}, w) \rightleftarrows (\mathfrak{M}', w') \Rightarrow (\mathfrak{M}, w) \approx (\mathfrak{M}', w')$.

Доказательство:

Докажем индукцией по длине формулы φ:

1) $\varphi = A_i \in \sigma$: $(\mathfrak{M}, w) \rightleftarrows (\mathfrak{M}', w') \Rightarrow w \equiv w' \Rightarrow \mathfrak{M}, w \vDash A_i \Leftrightarrow$
$\Leftrightarrow \mathfrak{M}', w' \vDash A_i \Rightarrow (\mathfrak{M}, w) \approx (\mathfrak{M}', w')$;

2) $\varphi = \varphi_1 \,\&\, \varphi_2$: По инд. предположению $\left.\begin{array}{l} \mathfrak{M}, w \vDash \varphi_1 \Leftrightarrow \mathfrak{M}', w' \vDash \varphi_1 \\ \mathfrak{M}, w \vDash \varphi_2 \Leftrightarrow \mathfrak{M}', w' \vDash \varphi_2 \end{array}\right\} \Rightarrow$

$\Rightarrow \mathfrak{M}, w \vDash \varphi_1 \,\&\, \varphi_2 \Leftrightarrow \mathfrak{M}', w' \vDash \varphi_1 \,\&\, \varphi_2 \Rightarrow (\mathfrak{M}, w) \approx (\mathfrak{M}', w')$;

Таким же образом доказывается и для $\vee, \rightarrow, \neg$...

6) $\varphi = \Diamond\varphi_1$:

Пусть $\mathfrak{M}, w \vDash \Diamond\varphi_1 \Leftrightarrow \exists v \in W$: $((w, v) \in R$ и $\mathfrak{M}, v \vDash \varphi_1) \Rightarrow$
$\Rightarrow \exists v' \in W'$: $((w', v') \in R'$ и $(v, v') \in E) \xRightarrow{\text{инд.пр.}} \exists v' \in W'$: $((w', v') \in R$ и $\mathfrak{M}', v' \vDash \varphi_1 \Leftrightarrow \mathfrak{M}', w' \vDash \Diamond\varphi_1 \Rightarrow (\mathfrak{M}, w) \approx (\mathfrak{M}', w')$;

7) $\varphi = \Box\varphi_1$:

Пусть $\mathfrak{M}, w \vDash \Box\varphi_1 \Leftrightarrow \mathfrak{M}, w \vDash \neg\Diamond\neg\varphi_1 \Leftrightarrow \mathfrak{M}, w \nvDash \Diamond\neg\varphi_1 \xLeftrightarrow{\text{6) пункт}} \mathfrak{M}', w' \nvDash$
$\nvDash \Diamond\neg\varphi_1 \Leftrightarrow \mathfrak{M}', w' \vDash \neg\Diamond\neg\varphi_1 \Leftrightarrow \mathfrak{M}', w' \vDash \Box\varphi_1 \Rightarrow (\mathfrak{M}, w) \approx (\mathfrak{M}', w')$.

Теорема доказана.

Теорема 37.5.

Пусть модели $\mathfrak{M} = \langle W, R, V\rangle$ и $\mathfrak{M}' = \langle W', R', V'\rangle$ - конечные, и $w \in W$, $w' \in W'$.

Тогда $(\mathfrak{M}, w) \rightleftarrows (\mathfrak{M}', w') \Leftrightarrow (\mathfrak{M}, w) \approx (\mathfrak{M}', w')$.

Доказательство:

$\boxed{(\Rightarrow)}$ Предыдущая теорема.

$(\Leftarrow)$ Пусть $(\mathfrak{M}, w) \approx (\mathfrak{M}', w')$. Зададим бисимуляцию $E \subseteq W \times W'$.

Покажем, что $(w, w') \in E \Leftarrow (\mathfrak{M}, w) \approx (\mathfrak{M}', w')$.

Проверим выполнимость условий:

1) Атомарная гармония:

Тогда $\mathfrak{M}, w \vDash A_i \Leftrightarrow \mathfrak{M}', w' \vDash A_i$.

2) "*Зиг*" :

Рассмотрим два случая:

Сл.1. Пусть w - тупиковый мир, т.е. $\neg\exists v \in W$: $(w, v) \in R$. Импликация в условии "*Зиг*" выполняется $\Rightarrow$ "*Зиг*" выполняется.

Сл.2. Пусть w - не тупиковый и $\exists v \in W$: $(w, v) \in R$. Докажем от противного и предположим, что "*Зиг*" не выполняется.

Тогда выходит по условию, что $\forall v' \in W'$: $(w', v') \in R'$: $(v, v') \notin E$.

Рассмотрим множество $S' = \{v' \in W' \mid (w', v') \in R'\}$. Покажем, что данное множество не пустое.

Поскольку мы взяли нетупиковый мир w, то у него точно есть выход в другой мир, и будет выполняться $\mathfrak{M}, w \vDash \Diamond 1$. А т.к. по условию $(\mathfrak{M}, w) \approx (\mathfrak{M}', w') \Rightarrow \mathfrak{M}', w' \vDash \Diamond 1 \Rightarrow w'$ - не тупиковый $\Rightarrow S' \neq \varnothing$.

С другой стороны, S' - конечно. Тогда $S' = \{v'_1, v'_2, \ldots, v'_n\}$. Поскольку "*Зиг*" не выполняется $\Rightarrow \forall v'_i \in S'$: $(v, v'_i) \notin E \Rightarrow (\mathfrak{M}, v) \not\approx (\mathfrak{M}', v'_i) \Rightarrow$
$\Rightarrow \exists \varphi_i : \mathfrak{M}, v \vDash \varphi_i$ и $\mathfrak{M}', v'_i \nvDash \varphi_i$.

Т.е. у нас для каждого такого мира есть своя формула, что ломает модальную эквивалентность. Тогда давайте рассмотрим формулу $\varphi = \varphi_1 \,\&\, \ldots \,\&\, \varphi_n$. Тогда $\mathfrak{M}, w \vDash \Diamond\varphi$ и $\mathfrak{M}', w' \nvDash \Diamond\varphi \Rightarrow (\mathfrak{M}, w) \not\approx (\mathfrak{M}', w')$. Получили противоречие с начальными условиями $\Rightarrow$ "*Зиг*" выполняется.

3) "*Заг*" : Аналогичное доказательство с "*Зиг*".

Теорема доказана.

Стоит отметить, что утверждение теоремы верно и для бесконечных моделей, если в этих моделях для каждого содержащегося в них мира не больше конечного числа последующих из них миров.

§38. Бисимулятивное сокращение

Определение 38.1.

Пусть $\mathfrak{M} = \langle W, R, V \rangle, w, v \in W$. Тогда миры **эквивалентны**, т.е. $w \sim v \Leftrightarrow (\mathfrak{M}, w) \approx (\mathfrak{M}, v)$. Здесь $\sim$ - отношение эквивалентности.

Определение 38.2.

Определим **множество классов эквивалентности**

$$W_\sim = \{[w]_\sim \mid w \in W\}.$$

Модель $\mathfrak{M}_\sim = \langle W_\sim, R_\sim, V_\sim \rangle$ называется **бисимулятивным сокращением модели** $\mathfrak{M}$, если:

1) $([w]_\sim, [v]_\sim) \in R_\sim \Leftrightarrow \exists w' \in [w]_\sim \, \exists v' \in [v]_\sim \colon (w', v') \in R$;

2) $V_\sim(A) = \{[w]_\sim \in W_\sim \mid w \in V(A)\}$.

Теорема 38.3.

Пусть $\mathfrak{M} = \langle W, R, V \rangle$ - конечная.

Тогда $\mathfrak{M} \rightleftarrows \mathfrak{M}_\sim$.

Доказательство:

Зададим бисимуляцию $E \subseteq W \times W_\sim \colon \forall w \in W \colon (w, [w]_\sim) \in E$.

Проверим выполнимость условий бисимуляции:

1) Атомарная гармония:

$\mathfrak{M}, w \vDash A \Leftrightarrow w \in V(A) \Leftrightarrow [w]_\sim \in V_\sim(A) \Leftrightarrow \mathfrak{M}_\sim, [w]_\sim \vDash A$.

2) "*Зиг*" :

Рассмотрим два случая:

Сл.1. Пусть w - тупиковый мир. Очевидно.

Сл.2. Пусть w - не тупиковый. Тогда $\exists v \in W$: $(w, v) \in R$. Но $w \in [w]_\sim$ и $v \in [v]_\sim \Rightarrow ([w]_\sim, [v]_\sim) \in R_\sim$.

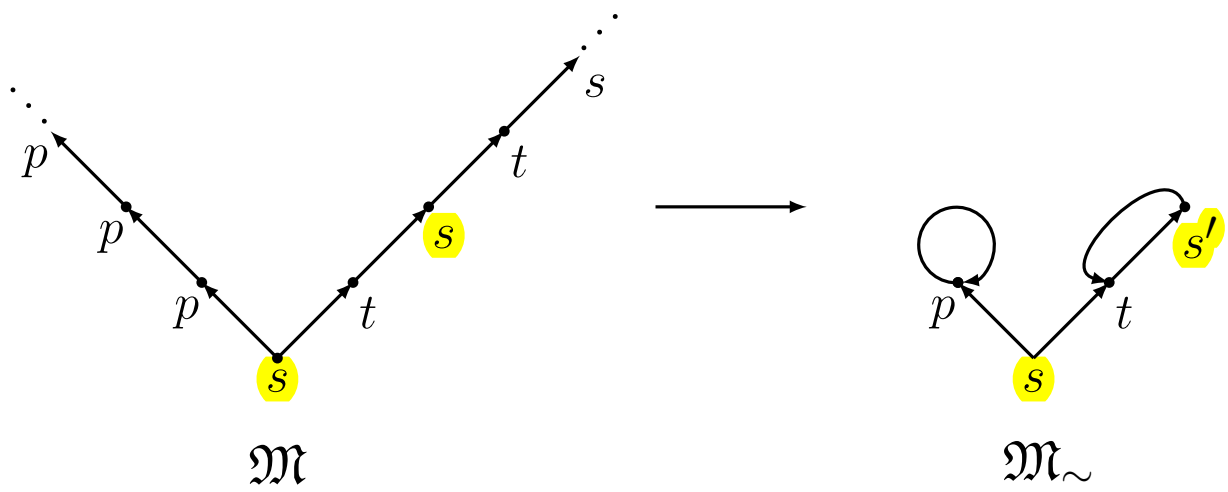
По определению отношения E выполняется $(v, [v]_\sim) \Rightarrow$ "*Зиг*" выполняется.

3) "*Заг*" : Аналогичное доказательство с "*Зиг*".

Теорема доказана.

Для чего нам нужно бисимулятивное сокращение?

Пример:



Рассмотрим модель $\mathfrak{M}$. Мы видим, что миры s и t чередуются и каждый мир s модально эквивалентен следующему s. Это же будет выполняться и с мирами p и t.

Здесь видно, что мы можем сократить нашу модель до модели $\mathfrak{M}_\sim$ - нашего бисимулятивного сокращения. Здесь из t мы не можем вернуться s, т.к. s' и s не являются модально эквивалентными, т.к. из s можно попасть в p, а из s' нельзя. Здесь классов эквивалентностей будет 4: класс эквивалентности

p, класс s, класс s' и класс t.

Данное сокращение упрощает работу с моделью, что как раз показал нам пример.

§39. Модельные конструкции

Определение 39.1.

Пусть даны модели $\mathfrak{M} = \langle W, R, V\rangle$ и $\mathfrak{M}' = \langle W', R', V'\rangle$.
Тогда $\mathfrak{M}, \mathfrak{M}'$ - **непересекающиеся**, если $W \cap W' = \varnothing$.

Определение 39.2.

Пусть есть множество моделей $\mathfrak{M}_i = \langle W_i, R_i, V_i\rangle,\ i \in \mathcal{I}$.
Тогда модель $\mathfrak{M} = \langle W, R, V\rangle$ называется **дизъюнктным объединением**, если $\mathfrak{M} = \biguplus\limits_{i\in\mathcal{I}} \mathfrak{M}_i$, где:

1) $W = \bigvee\limits_{i\in\mathcal{I}} W_i$;

2) $R = \bigvee\limits_{i\in\mathcal{I}} R_i$;

3) $V(A) = \bigvee\limits_{i\in\mathcal{I}} V_i(A),\ \forall A \in \sigma$.

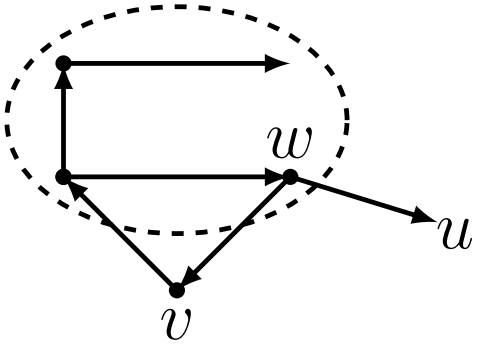
Определение 39.3.

Пусть дана модель $\mathfrak{M} = \langle W, R, V\rangle$ и есть подмножество $W' \subseteq W$. Тогда модель $\mathfrak{M}' = \langle W', R', V'\rangle$ называется **сужением модели $\mathfrak{M}$**, где:

1) $R' = R \cap (W' \times W')$;

2) $V'(A) = V(A) \cap W'$.

Пример:



Рассмотрим нашу модель и обведём наше подмножество W'. И получается, что модель внутри нашего кружка и будет являться *сужением*, миры v и u не войдут в новую модель.

Если нам дано n миров, то 2^n различных сужений мы сможем построить.

Определение 39.4.

Сужение модели является **сгенерированной подмоделью**, если оно замкнуто относительно отношения достижимости, т.е. если R' замкнуто.

Замкнутость говорит о том, что если какой-то мир принадлежит сужению, то и из него последующий мир так же должен быть в сужении.

Т.е. наш пример сужения выше не является сгенерированной подмоделью, ибо замкнутость нарушает мир w.

Определение 39.5.

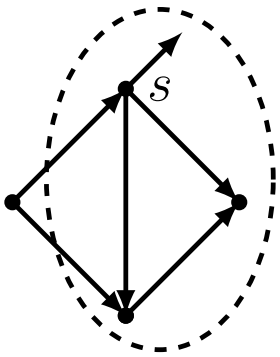
Пусть $\mathfrak{M}'$ - сгенерированная подмодель $\mathfrak{M}$. Рассмотрим подмножество $S \subseteq W'$, удолетворяющее следующим условиям:

$\forall w' \in W' \; \exists s \in S \; \exists v_1, \ldots, v_n \in W'$: $(s, v_1) \in R', (v_1, v_2) \in R', \ldots\ldots$
$\ldots, (v_{n-1}, v_n) \in R', (v_n, w') \in R'$.

Если S выполняет это условие, то оно называется **порождающим** и модель $\mathfrak{M}'$ **порождается** множеством миров S.

Т.е. S - это множество миров, из которых мы можем попасть в любой другой мир.

Пример:



Здесь $\mathfrak{M}'$ в круге, а *порождающим множеством* будет мир s.

Определение 39.6.

Пусть даны модели $\mathfrak{M} = \langle W, R, V\rangle$ и $\mathfrak{M}' = \langle W', R', V'\rangle$. Тогда отображение $f\colon W \to W'$ называется **ограниченным морфизмом**, если выполняются следующие условия:

1) *Атомарная гармония*: $w \in V(A) \Leftrightarrow f(w) \in V'(A)$;

2) *Морфизм*:

$$\forall w, v \in W\colon ((w, v) \in R \Rightarrow (f(w), f(v)) \in R');$$

3) "*Заг*":

$\forall w \in W\ \forall v' \in W'\colon (f(w), v') \in R' \Rightarrow \exists v \in W\colon (f(v) = v'$ и $(w, v) \in R)$.

Условие "*Зиг*" просто перешёл в "*Морфизм*". Здесь так же важно учесть, что "$\Rightarrow$" сопоставима с импликацией.

Данное выше определение очень сильно похоже на бисимуляцию. В чём различие? Если в бисимуляции могло так сложиться, что один мир переходит в два других мира на иной модели, а те два мира с другой модели в один мир(см. пример после 37.2.), то ограниченный морфизм не позволит из

одного мира попасть в два, только в один мир. Т.е. получается, что ограниченный морфизм - это бисимуляция с *навешенной* функциональностью и для любого мира существует только единственный образ в иной модели. Т.е. f - отношение бисимуляции, обладающее функциональностью.

Определение 39.7.

Если f является *сюръекцией*, то $\mathfrak{M}'$ - **ОМ-образ модели** $\mathfrak{M}$ (модель $\mathfrak{M}'$ из прошлого определения).

Теорема 39.8.

Для $\forall\varphi$ выполняется:

1) Пусть $\mathfrak{M} = \biguplus\limits_{i\in\mathcal{I}} \mathfrak{M}_i$. Тогда для $\forall w \in W_i\ \forall i \in \mathcal{I}$: $\mathfrak{M}, w \vDash \varphi \Leftrightarrow \mathfrak{M}_i, w \vDash \varphi$;

2) Пусть $\mathfrak{M}'$ - сгенерированная подмодель $\mathfrak{M}$.

Тогда для $\forall w \in W'$: $\mathfrak{M}, w \vDash \varphi \Leftrightarrow \mathfrak{M}', w \vDash \varphi$;

3) Если $\mathfrak{M}'$ - **ОМ**-образ $\mathfrak{M}$. Тогда для $\forall w \in W'$: $\mathfrak{M}, w \vDash \varphi \Leftrightarrow \mathfrak{M}', w \vDash \varphi$.

§40. Разрешимость классов структур Крипке

Вспомним, что структура Крипке - это

$F = (W, R)$ - пара из множества миров и отношения достижимости.

Если $F \vDash \varphi$, то φ истинная в любой модели с данной структурой.

В данном параграфе мы будем рассматривать $K = \{F_i \mid i \in \mathcal{I}\}$ - **класс структур Крипке**.

Предложение 40.1.

$K \vDash \varphi \Leftrightarrow \forall F_i \in K\colon F_i \vDash \varphi$.

Определение 40.2.

Класс структур K является **разрешимым**, если

$$\exists\varphi\forall F\colon F \vDash \varphi \Leftrightarrow F \in K.$$

Теорема 40.3.

Класс всех структур Крипке K_{frame} разрешим.

Доказательство:

Возьмём формулу $\xi = \Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$. Данная формула является т.и. $\Rightarrow$ она истинна везде $\Rightarrow$ класс всех структур Крипке разрешим.

Теорема доказана.

Далее мы будем искать классы, на которых истинна некоторая известная нам формула.

Теорема 40.4.

Пусть некоторая формула φ - выполнима и опровержима. Тогда:

1) $\Box\varphi \to \varphi$ **определяет класс** рефлексивных структур

$$K_1 = \{F = (W, R) \mid \forall w \in W, (w, w) \in R\}$$

2) $\varphi \to \Box\varphi$ определяет класс структур изолированных рефлексивных миров $K_2 = \{F \mid \forall w\forall v\colon (w, v) \in R \Rightarrow w = v\}$

3) $\Box 0$ определяет класс структур изолированных иррефлексивных миров

$$K_3 = \{F \mid \forall w\forall v\colon (w, v) \notin R\}$$

4) $\varphi \to \Box\Diamond\varphi$ определяет класс симметричных структур

$$K_4 = \{F \mid \forall w\forall v\colon (w, v) \in R \Rightarrow (v, w) \in R\}$$

5) $\Box\varphi \to \Box\Box\varphi$ определяет класс транзитивных структур

$$K_5 = \{F \mid \forall w \forall v \forall y \colon (w, v) \in R, (v, y) \in R \Rightarrow (w, y) \in R\}$$

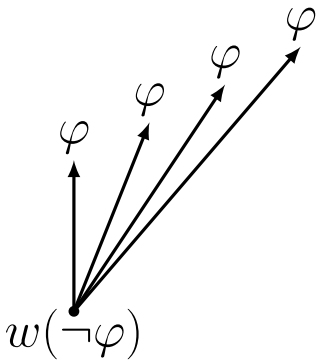
6) $\Diamond\varphi \to \Box\Diamond\varphi$ определяет класс структур, у которых отношение достижимости R - отношение эквивалентности.

Доказательство:

1) Пусть $F \vDash (\Box\varphi \to \varphi) \Leftrightarrow F \in K, F = (W, R)$.

$\boxed{(\Rightarrow)}$ Докажем от противного. Пусть $F \vDash (\Box\varphi \to \varphi)$ и $F \notin K_1 \Rightarrow$
$\Rightarrow \exists w \in W \colon (w, w) \notin R$.

Зададим модель $\mathfrak{M} = (W, R, V)$ следующим образом: $\mathfrak{M}, w \nvDash \varphi$ и $\mathfrak{M}, w \vDash \Box\varphi \Rightarrow \mathfrak{M}, w \nvDash (\Box\varphi \to \varphi)$.



Объясним этот шаг: здесь w иррефлексивен и будет выглядеть следующим образом. Поэтому у нас получается, что в мире w ложна формула $\Box\varphi \to \varphi$, откуда мы получаем противоречие с начальными условиями.

$\boxed{(\Leftarrow)}$ Докажем вновь от противного.

Пусть $F \in K_1$ и $F \nvDash (\Box\varphi \to \varphi) \Rightarrow \forall w \in W \colon F, w \vDash \Box\varphi$ и $F, w \nvDash \varphi$.

Из того, что $F, w \vDash \Box\varphi$ и $F \in K_1 \Rightarrow (w, w) \in R \Rightarrow F, w \vDash \varphi$. Получаем противоречие.

Другие пункты будут разобраны на семинаре...

Теорема доказана.

§41. Неразрешимость классов структур Крипке

Не все классы разрешимые. В данном параграфе как раз разберём их.

Определение 41.1.

1) Пусть $\{F_i \mid i \in \mathcal{I}\}$ - семейство непересекающихся структур Крипке. Тогда $\biguplus_{i\in\mathcal{I}} F_i$ - **дизъюнктное объединение**;

Данное объединение дано для структур, в прошлый раз давали объединение именно для моделей.

2) Пусть даны структуры $F = \langle W, R\rangle$, $F' = \langle W', R'\rangle$. Будем говорить, что F' - **подструктура** F, т.е. $F' \sqsubseteq F$, если каждая модель структуры F' - сгенерированная подмодель некоторой модели структуры F;

3) Пусть даны структуры $F = \langle W, R\rangle$, $F' = \langle W', R'\rangle$. Будем говорить, что F' - **OM-образ структуры** F, если каждая модель структуры F' - **OM**-образ некоторой модели структуры F.

Предложение 41.2.

Для любой формулы φ справедливы следующие условия:

1) Пусть $\{F_i \mid i \in \mathcal{I}\}$ - семейство непересекающихся структур Крипке. Тогда $\forall i \in \mathcal{I}$: $F_i \vDash \varphi \Rightarrow \biguplus_{i\in\mathcal{I}} F_i \vDash \varphi$;

2) Пусть $F' \sqsubseteq F$. Тогда $F' \vDash \varphi \Rightarrow F \vDash \varphi$;

3) Пусть F' - **OM**-образ F. Тогда $F' \vDash \varphi \Rightarrow F \vDash \varphi$.

Следствие 41.3.

Если класс структур Крипке не замкнут относительно дизъюнктного объединения или выделения подструктуры или формирования **OM**-образа, то он **неразрешим**.

Это следствие как раз нам и нужно для определения неразрешимости. Что значит неразрешимость? Это значит, что не существует формулы, что определяла бы эти классы.

Следствие 41.4.

Следующие классы структур Крипке неразрешимы:

1) Класс линейных структур;

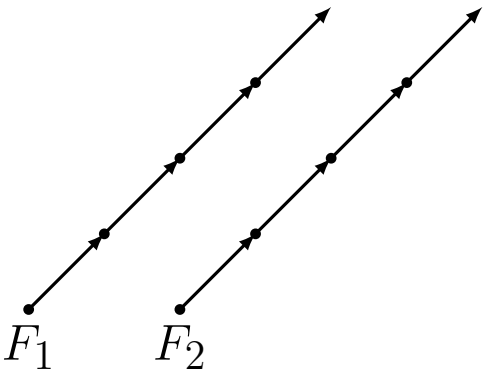
2) Класс антисимметричных структур;

3) Класс иррефлексивных структур;

4) Класс не рефлексивных и не иррефлексивных структур.

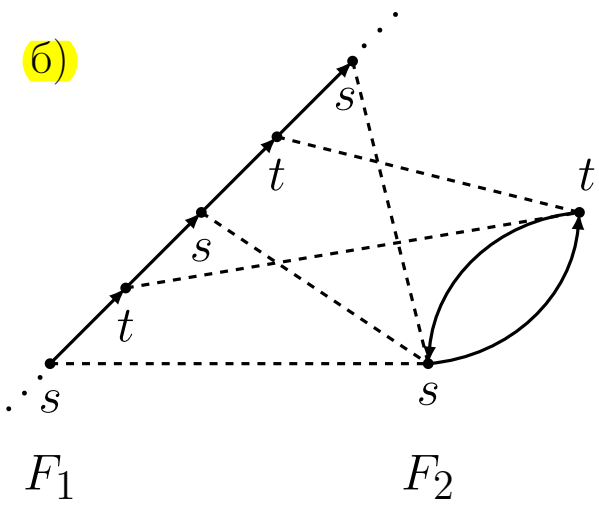
Доказательство:

а)



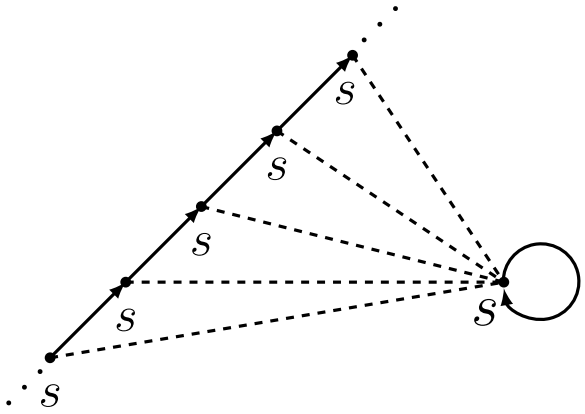
Возьмём их дизъюнктное объединение $F_1 \uplus F_2$. Получается ли в итоге линейная структура? Нет. Класс линейных структур не замкнут относительно дизъюнктного объединения. У нас говорится, что если формула истинна на некоторой структуре, то она истинна и на дизъюнктном объединении. Но из-за этого свойства нарушается и линейность, т.е. дизъюнктное объединение нелинейное.

б)



Здесь мы видим, что F_2 есть **OM**-образ F_1, но она уже не является антисимметричной структурой $\Rightarrow$ класс антисимметричных структур не замкнут относительно формирования **OM**-образа.

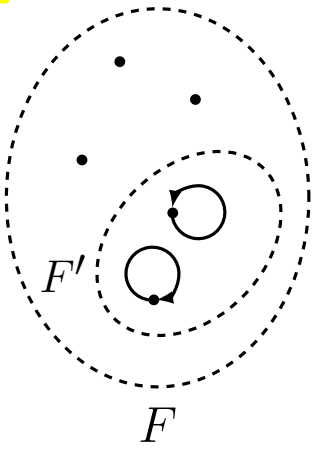
в)



F_1 F_2

Здесь та же история с **OM**-образом. F_2 есть **OM**-образ F_1, но она уже не является иррефлексивной структурой.

г)



В этом классе есть структуры с иррефлексивными и рефлексивными мирами. Рассмотрим такую структуру F. Так давайте выделим подструктуру F', что содержит только рефлексивные миры.

Тогда $F' \sqsubseteq F$, но F не рефлексивная $\Rightarrow$ класс структур не рефлексивных и не иррефлексивных структур не замкнут относительно выделения подструктуры.

Следствие доказано.

Следствие 41.5.

Каждый разрешимый класс структур либо обладает свойством рефлексивности, либо содержится в классе изолированных иррефлексивных миров.

§42. Стандартная трансляция модальной логики в Л.П.

В самом начале главы, когда вводили понятие модальной логики, мы обозначили множество пропозициональных переменных

$$\sigma = \{A_1, A_2, \ldots, A_n\}.$$

В этой главе мы научимся транслировать модальную логику в логику предикатов. Для этого введём новую сигнатуру

$$\sigma^* = \{P_1^{(1)}, \ldots, P_n^{(1)}, R^{*(2)}\}.$$

Определение 42.1.

Будем называть отображение $st_x\colon F(\sigma) \to F(\sigma^*)$ **стандартной трансляцией в логику предикатов**, если выполняются следующие условия:

1) $\forall A_i \in \sigma\colon st_x(A_i) = P_i(x)$;

2) $st_x(1) = 1,\ st_x(0) = 0$;

3) $st_x(\varphi\ \&\ \psi) = st_x(\varphi)\ \&\ st_x(\psi)$;

4) $st_x(\varphi \vee \psi) = st_x(\varphi) \vee st_x(\psi)$;

5) $st_x(\varphi \to \psi) = st_x(\varphi) \to st_x(\psi)$;

6) $st_x(\neg\varphi) = \neg st_x(\varphi)$;

7) $st_x(\Box\varphi) = \forall y(R^*(x, y) \to st_y(\varphi))$;

8) $st_x(\Diamond\varphi) = \exists y(R^*(x, y)\ \&\ st_y(\varphi))$.

Здесь переменные x, y являются как раз теми мирами, которыми мы пользовались в модальной логике.

Пример:

Пусть $\varphi = A_1\ \&\ \Box(A_2 \to \Diamond A_3)$. Тогда

$st_x(\varphi) = st_x(A_1\ \&\ \Box(A_2 \to \Diamond A_3)) \sim$

$\sim st_x(A_1)\ \&\ st_x(\Box(A_2 \to \Diamond A_3)) \sim$

$\sim P_1(x)\ \&\ \forall y(R^*(x,y) \to st_y(A_2 \to \Diamond A_3)) \sim$

$\sim P_1(x)\ \&\ \forall y(R^*(x,y) \to (st_y(A_2) \to st_y(\Diamond A_3))) \sim$

$\sim P_1(x)\ \&\ \forall y(R^*(x,y) \to (P_2(y) \to \exists z(R^*(y,z)\ \&\ P_3(z)))).$

Стоит отметить, что единственной свободной переменной в полученной формуле предикатов будет только x, т.е. тот мир, с которого мы транслировали формулу. И что бы мы не делали, мы всё равно получим после трансляции формулу логики предикатов только с одной свободной переменной. Так же отметим, что не всякой произвольной формуле логики предикатов мы сможем найти эквивалентную формулу модальной логики.

Мы отображаем наши формулы модальной логики в логику предикатов, однако как нам транслировать саму модель Крипке в логику предикатов? Или как её там интерпретировать? Введём следующее определение.

Определение 42.2.

Модель логики предикатов, **интерпретирующая модель Крипке** $\mathfrak{M} = \langle W, R, V\rangle$, является $\mathfrak{M}^* = \langle W, \sigma^*\rangle$, удолетворяющая следующим условиям:

1) $\mathfrak{M}^* \vDash R^*(w,v) \Leftrightarrow (w,v) \in R$, где $R^* \in \sigma^*$;

2) $\mathfrak{M}^* \vDash P_i(w) \Leftrightarrow w \in V(A_i)$, где $A_i \in \sigma,\ P_i \in \sigma^*$.

В конце можно заметить, что модальная логика по сути является сужением логики предикатов. Модальная логика *шире*, чем логика высказываний(поскольку в ЛВ нет модальностей), но она *уже*, чем логика предикатов. Но логика предикатов неразрешима.. Но мы можем выделить малую часть предикатов - модальную логику, где всё разрешимо, но при этом она меньше, но там лучше реализовываются некоторые вещи из предикатов.

Мы рассмотрели только самую простую модальную логику с одной модальностью. Но конечно же можно рассмотреть более сложные логики! Обычно стараются сделать более *объёмные* модальные логики, но при этом разрешимыми.

§43. Модальная логика первого порядка

Данная логика редко где используется, однако не повредит знать этого *дикого зверя*.

Когда мы рассматривали модальную логику высказываний(прошлые параграфы), там сигнатура состояла только из пропозициональных переменных. В логике первого порядка будет самая обычная сигнатура, а в узлах структур будут лежать обычные модели логики предикатов. У всех моделей будет одна и та же сигнатура. Получается более богатая формализация модальной логики.

Мы не будем рассматривать сигнатуры, в которых есть функциональные символы, иначе опять будут пляски с бесконечными множествами, как во 2й главе. Будем рассматривать только предикатную сигнатуру

$$\sigma = \{P_1^{(n_1)}, P_2^{(n_2)}, \dots\}.$$

Так же обозначим счётное множество переменных

$$X = \{x_1, x_2, \dots\}.$$

Поскольку у нас здесь нет констант и функциональных символов, то $X = T(\sigma)$, т.е. X равно множеству термов σ.

Определение 43.1.

Определение **формулы модальной логики первого порядка**:

1) $\forall x_i, x_j \in X$: $(x_i = x_j)$ - формула;

2) $\forall P^{(n)} \in \sigma \; \forall x_1, \ldots, x_n \in X$: $P(x_1, \ldots, x_n)$ - формула;

3) Пусть φ, ψ - формулы, тогда:

$$(\varphi \,\&\, \psi), (\varphi \vee \psi), (\varphi \to \psi), \neg\varphi, \forall x\varphi, \exists x\varphi, \Box\varphi, \Diamond\varphi \text{ - формулы;}$$

4) Других формул нет.

Что же у нас тогда будет моделью в этой логике? Введём следующее определение.

Определение 43.2.

Моделью Крипке постоянной предметной области мы будем называть $\mathfrak{M} = \langle W, R, D, \{V_w\}_{w \in W} \rangle$, где:

1) W - множество миров;

2) $R \subseteq W^2$ - отношение достижимости;

3) $D \neq \varnothing$ - **область квантификации**;

4) $V_w(P^{(n)}) \subseteq D^n$ - множество означивания предиката $P^{(n)}$ в мире w.

Зачем нам вообще это всё? Если в модальной логике высказываний на узлах в мирах задавалась истинность переменных, то теперь здесь задаётся истинность предикатов. Как именно? Через означивание $V_w(P^{(n)})$. D - некоторая фиксированная область, с которой мы работаем при определении истинности формул, отсюда понятие модели *с постоянной предметной областью*.

В начале главы мы говорили, что на узлах теперь будут модели логики предикатов. Это будут как раз модели $\mathfrak{B} = \langle D, \sigma \rangle$, где истинность предика-

тов на модели задаётся через $V_w(P^{(n)})$.

Давайте рассмотрим более подробно истинность формулы на модели модальной логики первого порядка.

Определение 43.3.

Пусть дана модель $\mathfrak{M} = \langle W, R, D, \{V_w\}_{w \in W} \rangle$.

Определим **истинность формулы на модели Крипке постоянной предметной области**. Пусть нам дано означивание $\gamma \colon X \to D$. Тогда:

1) $\mathfrak{M}, w \vDash (x_i = x_j)[\gamma] \Leftrightarrow \gamma(x_i) = \gamma(x_j)$;

2) $\mathfrak{M}, w \vDash P(x_1, \ldots, x_n)[\gamma] \Leftrightarrow \langle \gamma(x_1), \ldots, \gamma(x_n) \rangle \in V_w(P^{(n)})$;

3) $\mathfrak{M}, w \vDash (\varphi \,\&\, \psi)[\gamma] \Leftrightarrow \mathfrak{M}, w \vDash \varphi[\gamma]$ и $\mathfrak{M}, w \vDash \psi[\gamma]$;

4) $\mathfrak{M}, w \vDash (\varphi \vee \psi)[\gamma] \Leftrightarrow \mathfrak{M}, w \vDash \varphi[\gamma]$ или $\mathfrak{M}, w \vDash \psi[\gamma]$;

5) $\mathfrak{M}, w \vDash (\varphi \to \psi)[\gamma] \Leftrightarrow \mathfrak{M}, w \nvDash \varphi[\gamma]$ или $\mathfrak{M}, w \vDash \psi[\gamma]$;

6) $\mathfrak{M}, w \vDash \neg\varphi[\gamma] \Leftrightarrow \mathfrak{M}, w \nvDash \varphi[\gamma]$;

7) $\mathfrak{M}, w \vDash \forall x \varphi(x)[\gamma] \Leftrightarrow \forall a \in D \colon \mathfrak{M}, w \vDash \varphi(a)[\gamma]$;

8) $\mathfrak{M}, w \vDash \exists x \varphi(x)[\gamma] \Leftrightarrow \exists a \in D \colon \mathfrak{M}, w \vDash \varphi(a)[\gamma]$;

9) $\mathfrak{M}, w \vDash \Box\varphi[\gamma] \Leftrightarrow \forall v \in W \colon ((w, v) \in R \Rightarrow \mathfrak{M}, v \vDash \varphi[\gamma])$;

10) $\mathfrak{M}, w \vDash \Diamond\varphi[\gamma] \Leftrightarrow \exists v \in W \colon ((w, v) \in R$ и $\mathfrak{M}, v \vDash \varphi[\gamma])$.

Так же определим следующее:

1) $\mathfrak{M} \vDash \varphi \Leftrightarrow \forall w \in W \colon \mathfrak{M}, w \vDash \varphi$;

2) $\mathfrak{M}, w \vDash \varphi \Leftrightarrow \forall \gamma \colon \mathfrak{M}, w \vDash \varphi[\gamma]$.

Данная логика называется на иностранном **QK**. Давайте рассмотрим её аксиоматику.

Определение 43.4.

Исчисление QK$=K+ND+B$.

Её аксиоматика состоит из следующих аксиом:

1) Аксиомы Гильбертовского исчисления высказываний, то есть:

1. $(\varphi \to (\psi \to \varphi))$;
2. $((\varphi \to \psi) \to ((\varphi \to (\psi \to \xi)) \to (\varphi \to \xi)))$;
3. $((\varphi \,\&\, \psi) \to \varphi)$;
4. $((\varphi \,\&\, \psi) \to \psi)$;
5. $((\varphi \to \psi) \to ((\varphi \to \xi) \to (\varphi \to (\psi \,\&\, \xi))))$;
6. $(\varphi \to (\varphi \vee \psi))$;
7. $(\psi \to (\varphi \vee \psi))$;
8. $((\varphi \to \xi) \to ((\psi \to \xi) \to ((\varphi \vee \psi) \to \xi)))$;
9. $((\varphi \to \psi) \to ((\varphi \to \neg\psi) \to \neg\varphi))$;
10. $(\neg\neg\varphi \to \varphi)$.

2) (K) $\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$;

3) (R) $x = x$;

4) (S) $(x = y \,\&\, [\varphi]_x^z) \to [\varphi]_y^z$;

5) (A) $\forall x\varphi \to [\varphi]_y^x$;

6) (ND) $(x \neq y) \to \Box(x \neq y)$;

7) (B) $\forall x\Box\varphi \to \Box\forall x\varphi$ (Баркан).

Правила вывода:

$\dfrac{\varphi, \varphi \to \psi}{\psi}(modus\ ponens)$, $\qquad \dfrac{\varphi \to [\psi]_y^x}{\varphi \to \forall x\varphi}$, $\qquad \dfrac{\varphi}{\Box\varphi}$.

Этих аксиом и правил вывода достаточно, чтобы полностью задать модальную логику первого порядка.

Важно понимать, что аксиомы K, ND, B задают *модальность* логике, а все остальные аксиомы, взятые из логик высказывания и предикатов, зада-

ют иные свойства обозреваемой логики.

Предложение 43.5.

В логике **QK** доказуема обратная формула Баркан $\Box\forall x\varphi \to \forall x\Box\varphi$ (CB).

Доказательство:

Строим дерево секвенций:

$$\dfrac{\dfrac{\forall x\varphi \to \varphi}{\Box(\forall x\varphi \to \varphi),} \qquad \Box(\forall x\varphi \to \varphi) \to (\Box\forall x\varphi \to \Box\varphi)}{\dfrac{\Box\forall x\varphi \to \Box\varphi}{\Box\forall x\varphi \to \forall x\Box\varphi}}$$

Отсюда следует, что $\Box\forall x\varphi \to \forall x\Box\varphi$ доказуема.

Предложение доказано.

Мы рассмотрели модель Крипке с фиксированной предметной областью. В этой модели во всех узлах будут модели логики предикатов с одинаковым универсумом D и одинаковой сигнатурой σ. Но а если нам потребуется, чтобы универсумы на разных моделях в узлах были соответственно разными, т.е. модели с переменной предметной областью? Для этого введём следующее определение.

Определение 43.6.

Моделью Крипке переменной предметной области мы будем называть $\mathfrak{M} = \langle W, R, D, \{\delta_w\}_{w\in W}, \{V_w\}_{w\in W}\rangle$, где:

1) W - множество миров;

2) $R \subseteq W^2$ - отношение достижимости;

3) $D \neq \varnothing$ - область квантификации;

4) $\delta_w \subseteq D$ - **область квантификации в мире** w;

5) $V_w(P^{(n)}) \subseteq (\delta_w)^n$ - множество означивания предиката $P^{(n)}$ в мире w.

Здесь ничего почти не изменилось за исключением того, что теперь все модели в узлах могут быть с разными универсумами δ_w и разными истинностями предикатов $V_w(P^{(n)})$.

Определение 43.7.

Определим **истинность формулы на модели Крипке переменной предметной области**. Пусть нам дано означивание $\gamma\colon X \to D$. Тогда:

1) $\mathfrak{M}, w \vDash (x_i = x_j)[\gamma] \Leftrightarrow \gamma(x_i) = \gamma(x_j)$;

2) $\mathfrak{M}, w \vDash P(x_1, \ldots, x_n)[\gamma] \Leftrightarrow \langle \gamma(x_1), \ldots, \gamma(x_n)\rangle \in V_w(P^{(n)})$;

3) $\mathfrak{M}, w \vDash (\varphi \,\&\, \psi)[\gamma] \Leftrightarrow \mathfrak{M}, w \vDash \varphi[\gamma]$ и $\mathfrak{M}, w \vDash \psi[\gamma]$;

4) $\mathfrak{M}, w \vDash (\varphi \vee \psi)[\gamma] \Leftrightarrow \mathfrak{M}, w \vDash \varphi[\gamma]$ или $\mathfrak{M}, w \vDash \psi[\gamma]$;

5) $\mathfrak{M}, w \vDash (\varphi \to \psi)[\gamma] \Leftrightarrow \mathfrak{M}, w \nvDash \varphi[\gamma]$ или $\mathfrak{M}, w \vDash \psi[\gamma]$;

6) $\mathfrak{M}, w \vDash \neg\varphi[\gamma] \Leftrightarrow \mathfrak{M}, w \nvDash \varphi[\gamma]$;

7) $\mathfrak{M}, w \vDash \forall x\varphi(x)[\gamma] \Leftrightarrow \forall a \in \delta_w\colon \mathfrak{M}, w \vDash \varphi(a)[\gamma]$;

8) $\mathfrak{M}, w \vDash \exists x\varphi(x)[\gamma] \Leftrightarrow \exists a \in \delta_w\colon \mathfrak{M}, w \vDash \varphi(a)[\gamma]$;

9) $\mathfrak{M}, w \vDash \Box\varphi[\gamma] \Leftrightarrow \forall v \in W\colon ((w, v) \in R \Rightarrow \mathfrak{M}, v \vDash \varphi[\gamma])$;

10) $\mathfrak{M}, w \vDash \Diamond\varphi[\gamma] \Leftrightarrow \exists v \in W\colon ((w, v) \in R$ и $\mathfrak{M}, v \vDash \varphi[\gamma])$.

Так же сохраняются истинности формулы в мире и на модели.

Рассмотрим так же истинность формулы Баркан $\forall x\Box\varphi(x) \to \Box\forall x\varphi(x)$ на модели переменной п.о.. Будет ли формула истинной?

Подберём формулу для Баркан $\varphi(x) =$"*Студент x хорошо учится*". Проверим истинность получившейся формулы.

Если сейчас в мире w это выполняется, то в следующем мире v уже другая

предметная область. Тогда формула в следующем мире может быть ложной, поскольку мог поменяться универсум.

Если в нашей формуле Баркан говорилось, что:

Любой сегодняшний студент потом будет хорошо учиться

СЛЕДУЕТ

в будущем все студенты будут хорошо учиться

то не факт, что все будущие студенты будут хорошо учиться, так как мог прийти троечник из академа и начать понижать статистику потоку. У нас могли хорошо учиться текущие студенты, но насчёт будущих мы ничего сказать не можем. Поэтому формула Баркан не является истинной на модели Крипке переменной предметной области.

А что насчёт обратной формулы Баркан $\Box\forall x\varphi(x) \to \forall x\Box\varphi(x)$?

Она так же не будет выполняться, поскольку если в будущем все студенты будут хорошо учиться, то нет никакого гаранта, что все сегодняшние будут хорошо учиться, ибо кого-то из сегодняшних могут отчислить. Поэтому обратный Баркан так же невыполним на модели Крипке переменной п.о..

Т.е. если область у модели Крипке переменная, то аксиома Баркан ломается. Но мы будем рассматривать только два крайних случая:

1) Область у последующих миров не увеличивается;

2) Область у последующих миров не сужается.

В случае 1) будет выполняться простая формула Баркан, а в случае 2) - обратная формула Баркан.

Отсюда можно сделать интересные выводы:

1) Если на модели Крипке выполняется формула обратного Баркан, то области последующих миров в модели не сужаются;

2) Если на модели Крипке выполняется формула Баркан, то области по-

следующих миров в модели не увеличиваются;

3) Если на модели Крипке выполняется обратная и простая формулы Баркан, то модель с постоянной п.о.;

4) В ином случае, когда никакая из формул Баркан не выполняется, мы ничего сказать не можем, т.к. предметные области у миров переменные.

Запишем это в более строгом виде.

Определение 43.8.

Пусть дана модель Крипке переменной п.о. $\mathfrak{M} = \langle W, R, D, \{\delta_w\}_{w \in W}, \{V_w\}_{w \in W} \rangle$, тогда:

1) $\mathfrak{M}$ - **модель расширяющейся предметной области**, если

$$\forall w, v \in W\colon (w, v) \in R \Rightarrow \delta_w \subseteq \delta_v;$$

2) $\mathfrak{M}$ - **модель сужающейся предметной области**, если

$$\forall w, v \in W\colon (w, v) \in R \Rightarrow \delta_v \subseteq \delta_w.$$

Предложение 43.9.

$\mathfrak{M}$ - модель расширяющейся п.о. $\Leftrightarrow \mathfrak{M} \vDash CB$ (обратный Баркан).

$\mathfrak{M}$ - модель сужающейся п.о. $\Leftrightarrow \mathfrak{M} \vDash B$ (Баркан).

Замечание 43.10.

$\mathfrak{M} \vDash B$ и $\mathfrak{M} \vDash CB \Leftrightarrow \mathfrak{M}$ - модель Крипке постоянной п.о..

Если не выполняются B и CB, то модель с переменной п.о..

Помимо аксиомы Баркан, если продолжаем работать с переменной областью, у нас так же есть проблемы и с аксиомой A - $\forall x \varphi \to [\varphi]^x_y$. Мы могли заменить x на y, которого просто нет в области квантификации мира. Тогда могут выйти случаи, когда после замены изменится истинность формулы,

что очень нехорошо. Для этого придумали одну хитрость.

Предложение 43.11.

Зададим формулу $E(y) = \exists z(z = y)$.

Тогда $\gamma(y) \in \delta_w \Leftrightarrow \mathfrak{M}, w \vDash E(y)[\gamma]$.

И чтобы в будущем избежать проблем с аксиомой A, мы её заменим на аксиому AE - $(\forall x\varphi \;\&\; E(y)) \to [\varphi]^x_y$.

На основе того, что мы сейчас выяснили насчёт аксиом, мы можем определить новые логики:

1) $K + ND + E + B$ - модальная логика первого порядка для моделей Крипке сужающейся п.о.;

2) $K + ND + E + CB$ - модальная логика первого порядка для моделей Крипке расширяющейся п.о.;

3) $K+ND+E$ - модальная логика первого порядка для того случая, когда нам не нужно фиксировать сужение или увеличение предметной области у моделей Крипке.

На этой ноте мы заканчиваем данный параграф. Краем глаза заметил в какой-то статье, что модальную логику первого порядка где-то используют в автоматизации доказательств, как во 2й главе.

Есть еще достаточно великое количество подвидов модальной логики, и один из этих подвидов будет рассматриваться в следующем параграфе.

§44. Пропозиционально временная логика

Иногда её именуют **темпоральной логикой**.

Временная логика - частный случай модальной логики, когда цепь миров интерпретируется как линия времени и событий. Используется для описания последовательностей явлений и их взаимосвязи по временной шкале. Данная логика широко используется в интеллектуальных системах реального времени, системах верификации программного кода или отдельных различных систем.

Определение 44.1.

Пара $\langle T, < \rangle$ называется **потоком времени**, если:

1) $T \neq \varnothing$;

2) Отношение "$<$" $\subseteq T^2$ иррефлексивно и транзитивно.

Определение 44.2.

Поток времени $\langle T, < \rangle$ называется **линейным потоком времени**, если:

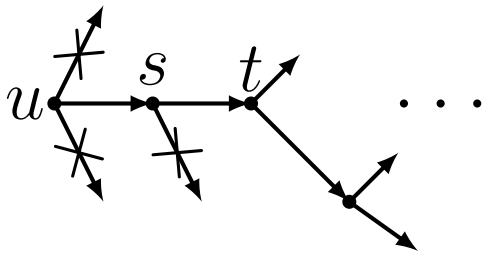
$$\forall t, s \in T : (t = s) \vee (t < s) \vee (s < t).$$

Пример:

Линейными потоками могут быть: $\langle \mathbb{N}, < \rangle, \langle \mathbb{Q}, < \rangle, \langle \mathbb{R}, < \rangle$.

Определение 44.3.

Поток времени $\langle T, < \rangle$ называется **древовидным потоком времени**, если $\forall t, s, u \in T : ((s < t) \,\&\, (u < t)) \to ((u = s) \vee (s < u) \vee (u < s))$.



О чём говорится в последнем определении? Здесь речь о том, что прошлое обязательно сравнимо между собой, а будущее еще не определено, следовательно, оно может быть древовидным. Обычно за мир t обозначают сегодняшний день, значит до него все события уже свершились и никаких других ответвлений до t уже быть не может. Но t - сегодняшний день, и мы не знаем, что будет впереди, следовательно возможны некоторые варианты дальнейших ответвлений. После выбора прошлое фиксируется и другие ответвления пропадают. Таким образом задаётся древовидная структура потока.

Исследуется огромное количество различных потоков, но древовидный и линейный потоки пользуются большой поопулярностью по очевидным причинам. Исследовали еще цикличные потоки, когда временной промежуток повторяется раз за разом. Рассмотрим её.

Определение 44.4.

Поток времени $\langle T, < \rangle$ называется **цикличным потоком времени**, если выполняются следующие условия:

1) $(t = s) \vee (t < s) \vee (s < t)$ - линейность;

2) $\neg((t < s) \,\&\, (s < t))$ - антисимметричность;

3) $((t < s) \,\&\, (t < u) \,\&\, (t < v) \,\&\, (s < u) \,\&\, (u < v)) \to (s < v)$ - транзитивность будущего;

4) $((s < t) \,\&\, (u < t) \,\&\, (v < t) \,\&\, (s < u) \,\&\, (u < v)) \to (s < v)$ - транзитивность прошлого;

5) $\exists a, b, c \in T\colon (a < b) \,\&\, (b < c) \,\&\, (c < a)$.

Зададим понятие модели Крипке временной логики.

Определение 44.5.

Моделью временной логики называется $\mathfrak{M} = \langle T, <, h \rangle$, где $h \colon \sigma \to \rho(T)$ - означивание пропозициональных переменных.

В зависимости от того, какие будем рассматривать модальности, у нас будут получаться различные формулы. Рассмотрим самые классические модальности.

Определение 44.6.

Модальностями Приота называют модальности:

1) $\langle F \rangle$ - "возможно в будущем", т.е.

$$\mathfrak{M}, t \vDash F\varphi \Leftrightarrow \exists s \in T \colon (t < s \;\&\; \mathfrak{M}, s \vDash \varphi);$$

2) $\langle P \rangle$ - "возможно в прошлом", т.е.

$$\mathfrak{M}, t \vDash P\varphi \Leftrightarrow \exists s \in T \colon (s < t \;\&\; \mathfrak{M}, s \vDash \varphi);$$

3) $[G]$ - "необходимо в будущем", т.е.

$$\mathfrak{M}, t \vDash G\varphi \Leftrightarrow \forall s \in T \colon (t < s \Rightarrow \mathfrak{M}, s \vDash \varphi);$$

4) $[H]$ - "необходимо в прошлом", т.е.

$$\mathfrak{M}, t \vDash H\varphi \Leftrightarrow \forall s \in T \colon (s < t \Rightarrow \mathfrak{M}, s \vDash \varphi).$$

Следующие модальности дополняют первый набор. Это набор модальностей относительно сегодняшнего или вчерашнего дня.

Определение 44.7.

Введём модальности относительно дня:

1) X - "возможно завтра", т.е.

$\mathfrak{M}, t \vDash X\varphi \Leftrightarrow \exists s \in T\colon (t < s \;\&\; \neg\exists u(t < u \;\&\; u < s) \;\&\; \mathfrak{M}, s \vDash \varphi)$;

2) $[X]$ - "необходимо завтра", т.е.

$\mathfrak{M}, t \vDash [X]\varphi \Leftrightarrow \forall s \in T\colon ((t < s \;\&\; \neg\exists u(t < u \;\&\; u < s)) \Rightarrow \mathfrak{M}, s \vDash \varphi)$;

3) Y - "возможно вчера", т.е.

$\mathfrak{M}, t \vDash Y\varphi \Leftrightarrow \exists s \in T\colon (t > s \;\&\; \neg\exists u(t > u \;\&\; u > s) \;\&\; \mathfrak{M}, s \vDash \varphi)$;

4) $[Y]$ - "необходимо вчера", т.е.

$\mathfrak{M}, t \vDash [Y]\varphi \Leftrightarrow \forall s \in T\colon ((t > s \;\&\; \neg\exists u(t > u \;\&\; u > s)) \Rightarrow \mathfrak{M}, s \vDash \varphi)$.

Учтём, что если поток времени линейный, то "возможно завтра" и "необходимо завтра" совпадают, так же и для вчерашнего дня.

Если у нас поток *непрерывен*, т.е. $T = \mathbb{R}$, тогда понятия вчерашнего или завтрашнего дня нет по понятным причинам. Тогда модальности относительно дней не работают. Они работают, когда поток *дискретный*, т.е. $T = \mathbb{N}$ или $T = \mathbb{Z}$ например.

Существует еще третий вид модальностей, который так же часто используют - это "*Until*" и "*Since*".

Определение 44.8.

Модальности "*Until*" и "*Since*":

1) $U(\varphi, \psi)$ ="До тех пор, пока не выполнится φ, будет выполняться и ψ", т.е.

$\mathfrak{M}, t \vDash U(\varphi, \psi) \Leftrightarrow \exists s\colon (t < s \;\&\; \mathfrak{M}, s \vDash \varphi \;\&\; \forall u((t < u \;\&\; u < s) \Rightarrow$
$\Rightarrow \mathfrak{M}, u \vDash \psi))$;

2) $S(\varphi, \psi)$ ="С тех пор, как выполняется φ, выполняется и ψ", т.е.

$\mathfrak{M}, t \vDash S(\varphi, \psi) \Leftrightarrow \exists s$: $(t > s \;\&\; \mathfrak{M}, s \vDash \varphi \;\&\; \forall u((t > u \;\&\; u > s) \Rightarrow$
$\Rightarrow \mathfrak{M}, u \vDash \psi))$.

В данном параграфе рассматривались только самые ходовые модальности. Важно понимать, что мы сами можем придумать разные модальности. Если открыть какую нибудь книгу по временным логикам, то там для описания реальности придумывают собственные модальности, т.е. записывают шаблонные предложения, затем описание их истинности на модели. На этом закончим обзор временной логики и рассмотрим её трансляцию в логику предикатов.

§45. Стандартная трансляция временной логики в Л.П.

Идея данной трансляции не сильно отличается от предыдущей, но меняется из-за специфики временной логики.

Вспомним сигнатуру, которую ввели в начале модальной логики:

$$\sigma = \{A_1, A_2, \dots\}.$$

Введём сигнатуру, которую будем использовать при трансляции:

$$\sigma^* = \{P_1^{(1)}, P_2^{(1)}, \dots, <\}.$$

Определение 45.1.

Будем называть отображение st_x: $F(\sigma) \to F(\sigma^*)$ **стандартной трансляцией в логику предикатов**, если выполняются следующие условия:

1) $\forall A_i \in \sigma$: $st_x(A_i) = P_i(x)$;

2) $st_x(1) = 1,\ st_x(0) = 0$;

3) $st_x(\varphi \;\&\; \psi) = st_x(\varphi) \;\&\; st_x(\psi)$;

4) $st_x(\varphi \vee \psi) = st_x(\varphi) \vee st_x(\psi)$;

5) $st_x(\varphi \to \psi) = st_x(\varphi) \to st_x(\psi)$;

6) $st_x(\neg\varphi) = \neg st_x(\varphi)$;

7) $st_x(F\varphi) = \exists y(x < y \;\&\; st_y(\varphi))$;

8) $st_x(G\varphi) = \forall y(x < y \to st_y(\varphi))$;

9) $st_x(P\varphi) = \exists y(x > y \;\&\; st_y(\varphi))$;

10) $st_x(H\varphi) = \forall y(x > y \to st_y(\varphi))$;

11) $st_x(X\varphi) = \exists y(x < y \;\&\; \neg\exists z(x < z \;\&\; z < y) \;\&\; st_y(\varphi))$;

12) $st_x([X]\varphi) = \forall y((x < y \;\&\; \neg\exists z(x < z \;\&\; z < y)) \to st_y(\varphi))$;

13) $st_x(Y\varphi) = \exists y(x > y \;\&\; \neg\exists z(x > z \;\&\; z > y) \;\&\; st_y(\varphi))$;

14) $st_x([Y]\varphi) = \forall y((x > y \;\&\; \neg\exists z(x > z \;\&\; z > y)) \to st_y(\varphi))$;

15) $st_x(U(\varphi, \psi)) = \exists y(x < y \;\&\; st_y(\varphi) \;\&\; \forall z((x < z \;\&\; z < y) \to st_z(\psi))$;

16) $st_x(S(\varphi, \psi)) = \exists y(x > y \;\&\; st_y(\varphi) \;\&\; \forall z((x > z \;\&\; z > y) \to st_z(\psi))$.

Определение 45.2.

Пусть дана модель временной логики $\mathfrak{M} = \langle T, <, h\rangle$. Тогда будем говорить, что $\mathfrak{M}^* = \langle T, \sigma^*\rangle$ **интерпретирует** $\mathfrak{M}$, если $\forall t, s \in T$ выполняется:

1) $\mathfrak{M}^* \vDash (t < s) \Leftrightarrow t < s$;

2) $\forall A_i \in \sigma\ \forall P_i \in \sigma^*$: $\mathfrak{M}^* \vDash P_j(t) \Leftrightarrow t \in h(A_i)$.

В дальнейшем нам надо разобрать, как работать с формулами в потоках времени. Для этого выделяются классы потоков с некоторыми одинаковыми свойствами.

§46. Классы потоков времени

Определение 46.1.

Классом потоков времени мы будем называть множество

$$C = \{\langle T_i, <_i \rangle \mid i \in \mathcal{I}\}.$$

Определение 46.2.

Пусть даны $\varphi(x), \psi(x) \in F(\sigma^*)$. Тогда $\varphi(x)$ и $\psi(x)$ **эквивалентны над классом потоков** C, т.е. $\varphi(x) \sim_c \psi(x)$, если выполняется:

$$\forall \langle T, < \rangle \in C \; \forall \mathfrak{M} = \langle T, <, h \rangle \; \forall t \in T\colon \; \mathfrak{M} \vDash (\varphi(t) \leftrightarrow \psi(t)),$$

где σ^* из прошлого параграфа.

Определение 46.3.

Пусть дан класс $C = \{\langle T_i, <_i \rangle \mid i \in \mathcal{I}\}$. Пусть $\alpha \in F(\sigma), \varphi(x) \in F(\sigma*)$. Тогда $\alpha \sim_c \varphi(x) \Leftrightarrow \forall t \in T\colon st_t(\alpha) \sim_c \varphi(t)$.

Определение 46.4.

Пусть дана модальная логика $ML = (\sigma, \{m_1, \ldots, m_n\})$ и класс $C = \{\langle T_i, <_i \rangle \mid i \in \mathcal{I}\}$. Тогда ML **выразительно полна над классом потоков** C, если $\forall \varphi(x) \in F(\sigma^*) \; \exists \alpha \in F(\sigma)\colon \alpha \sim_c \varphi(x)$.

Здесь мы сами задаём модальную логику, задавая набор модальностей. Зачем? Читай продолжение.

Теорема 46.5.

Не существует временной логики с конечным набором модальностей, выразительно полной над классом всех потоков времени, т.е. класс

$C = \{\langle T_i, \alpha_i\rangle \mid i \in \mathcal{I}\}$, где α - некоторое отношение между мирами, то есть, например, отношение для древовидных потоков, линейных и всяких других экзотических потоков.

Без доказательства.

Определение 46.6.

Пусть дан класс $C = \{\langle T_i, <_i\rangle \mid i \in \mathcal{I}\}$ и $\alpha \in F(\sigma)$. Тогда:

1) α **описывает прошлое над классом** C, если выполняется условие:

$$\forall\langle T_i, <_i\rangle \in C\ \forall\mathfrak{M}_1 = \langle T_i, <_i, h_1\rangle\ \forall\mathfrak{M}_2 = \langle T_i, <_i, h_2\rangle\ \forall t \in T:$$
$$:\forall u(u < t\ \&\ (\mathfrak{M}_1, u) \approx (\mathfrak{M}_2, u)) \Rightarrow (\mathfrak{M}_1, t \vDash \alpha \Leftrightarrow \mathfrak{M}_2, t \vDash \alpha).$$

2) α **описывает будущее над классом** C, если выполняется условие:

$$\forall\langle T_i, <_i\rangle \in C\ \forall\mathfrak{M}_1 = \langle T_i, <_i, h_1\rangle\ \forall\mathfrak{M}_2 = \langle T_i, <_i, h_2\rangle\ \forall t \in T:$$
$$:\forall u(t < u\ \&\ (\mathfrak{M}_1, u) \approx (\mathfrak{M}_2, u)) \Rightarrow (\mathfrak{M}_1, t \vDash \alpha \Leftrightarrow \mathfrak{M}_2, t \vDash \alpha).$$

3) α **описывает настоящее над классом** C, если выполняется условие:

$$\forall\langle T_i, <_i\rangle \in C\ \forall\mathfrak{M}_1 = \langle T_i, <_i, h_1\rangle\ \forall\mathfrak{M}_2 = \langle T_i, <_i, h_2\rangle\ \forall t \in T:$$
$$:\mathfrak{M}_1, t \vDash \alpha \Leftrightarrow \mathfrak{M}_2, t \vDash \alpha.$$

4) α является **разделяемой над классом** C, если она является булевой комбинацией формул($\&$, $\vee, \neg, \rightarrow$), описывающих либо прошлое, либо будущее, либо настоящее.

5) Логика ML **разделима над классом** C, если для каждой её формулы найдётся эквивалентная ей разделяемая формула над C, т.е.

$$\forall\beta \in F(\sigma)\ \exists\alpha \in F(\sigma)\colon \beta \sim_c \alpha,\ \alpha \text{ - разделяемая.}$$

Пример:

$S(A_1, A_2)\ \&\ FA_1$ - разделима, но $S(A_1, A_2)$ - нет.

Теорема 46.7.

Пусть C - класс линейных потоков. Пусть ML содержит модальности Приота(опр. 44.6.), тогда ML - разделима над $C \Leftrightarrow ML$ - выразительно полна.

Теорема 46.8.

Пусть ML содержит модальности Приота и модальности "$Until$" и "$Since$". Тогда ML - разделима над $\langle \mathbb{N}, < \rangle$.

Определение 46.9.

Модальность $Stavi(\varphi, \psi)$="ψ *истинно в настоящее время, вплоть до некоторого времени, после которого через некоторое время* ψ *становится ложной и* φ *становится истинной*", т.е.

$$Stavi(P_1, P_2) = \exists s \Big(t < s \ \& \ \exists u \big(t < u < s \ \& \ \neg P_2(u) \big) \ \&$$
$$\& \ \exists u \big(t < u < s \ \& \ \forall v (t < v < u \to P_2(v)) \big) \ \&$$
$$\& \ \forall u \big(t < u < s \to \big[\exists v (u < v \ \& \ \forall w (t < w < v \to P_2(w))) \big] \vee$$
$$\vee \ \big[\forall v (u < v < s \to P_1(v)) \ \& \ \exists v (t < v < u \ \& \ \neg P_2(v)) \big] \big) \Big).$$

Теорема 46.10.

Логика ML, содержащая модальности Приота, "$Until$", "$Since$" и "$Stavi$" разделима над классом всех линейных потоков.

Т.е. зачем нам нужны данные свойства модальной логики? Важно понимать, что здесь под модальной логикой имеется ввиду некоторая временная логика. И чтобы описать, например, все линейные потоки в логике, нам достаточно, например, иметь в ML модальности Приота, "$Until$", "$Since$" и "$Stavi$". Описав эту логику с линейными потоками, там мы дальше можем исследовать некоторые свойства формул и линейных порядков.

4. Нечёткая логика

§47. Многозначные логики

Многозначная логика — тип формальной логики, в которой допускается более двух истинностных значений для высказываний. В настоящее время существует очень много других систем многозначной логики, которые в свою очередь могут быть сгруппированы по классам. Важнейшими из таких классов являются частичные логики и нечёткие логики.

Первым, кто разработал многозначную логику высказываний, был Лукасевич в 1920 году.

Определение 47.1.

Трёхзначная логика Лукасевича задаётся истинностями:

1 - "*True*",

0 - "*False*",

$1/2$ - "*Unknown*".

Таблицы истинности в данной логике задаются следующим образом:

OR	**F**	**T**	**U**
F	F	T	U
T	T	T	T
U	U	T	U

AND	**F**	**T**	**U**
F	F	F	F
T	F	T	U
U	F	U	U

NOT	
F	T
T	F
U	U

Так же важно понимать, что $A \vee \neg A \neq 1$ и $A \,\&\, \neg A \neq 0$.

Определение 47.2.

Логика Бочвара задаётся истинностями:

1 - "*Истина*",

0 - "*Ложь*",

1/2 - "*Бессмысленно(Senselessly)*".

Таблицы истинности в данной логике задаются следующим образом. Здесь столбцы и колонки по другому поставлены! Это для лучшего запоминания:

OR	**T**	**S**	**F**
T	T	S	T
S	S	S	S
F	T	S	F

AND	**T**	**S**	**F**
T	T	S	F
S	S	S	S
F	F	S	F

NOT	
T	F
S	S
F	T

В 1938 г. Д.А. Бочваром была построена первая в мире логика бессмысленности в связи с проблемой разрешения логических антиномий, в первую очередь *парадокса Рассела*. В данной системе третье истинностное значение 1/2 предлагается интерпретировать не столько как промежуточное между истиной 1 и ложью 0, сколько как парадоксальное значение или даже как «бессмысленность».

§48. Нечёткая логика

Чуть позже, в 1965, Лотфи Заде придумывает **нечёткую логику** - обобщение логики и теории множеств, где основные операции производятся над над функцией принадлежности элемента к **нечёткому** множеству, принимающей любые значения в интервале $[0, 1]$, а не только 0 или 1.

Здесь основная работа с функцией принадлежности. Самая обычная функция - характеристическая для чётких множеств, выглядит так

$$\mu_A(x) = \begin{cases} 1, \ x \in A \\ 0, \ x \notin A \end{cases}.$$

А вот с функцией принадлежности элемента к нечёткому множеству всё будет поинтересней:

Пример:

Рассмотрим самый простой пример.

Формализуем неточное определение "*горячий чай*". В качестве x будет выступать шкала температуры в градусах Цельсия. Она будет изменяется от 0 до 100 градусов. Нечёткое множество для понятия "*горячий чай*" может выглядеть следующим образом:

$$C = \{(0,0); (0,10); (0,20); (0.15,30); (0.3,40); (0.6,50); (0.8,60); (0.9,70); (1,80); (1,90); (1,100)\}.$$

т.е. C - множество упорядоченных пар вида $(\mu_C(x), x)$.

Так, чай с температурой $60°C$ принадлежит к множеству "*Горячий*" со степенью принадлежности 0.8. Для одного человека чай при температуре $60°C$ может оказаться горячим, для другого – не слишком горячим. Именно в этом и проявляется нечеткость задания соответствующего множества.

Для нечетких множеств, как и для обычных, определены основные логические операции. Существуют различные виды нечётких логик.

Определение 48.1.(Операции над нечёткими множествами).

1) **Операции Заде:**

$\mu_{A\cup B}(x) = max\{\mu_A(x), \mu_B(x)\}$,

$\mu_{A\cap B}(x) = min\{\mu_A(x), \mu_B(x)\}$,

$\mu_{\neg A}(x) = 1 - \mu_A(x)$.

Пример:

Пусть $\mu_A(x) = 0,7$. Тогда $\mu_{\neg A}(x) = 0.3$.

$\mu_{A \cup \neg A}(x) = max\{0.7, 0.3\} = 0.7 \neq 1$.

$\mu_{A \cap \neg A}(x) = min\{0.7, 0.3\} = 0.3 \neq 0$.

2) **Логика Лукасевича:**

$\mu_{A \cup B}(x) = min\{\mu_A(x) + \mu_B(x), 1\}$,

$\mu_{A \cap B}(x) = max\{\mu_A(x) + \mu_B(x) - 1, 0\}$,

$\mu_{\neg A}(x) = 1 - \mu_A(x)$.

3) **Вероятностная логика:**

$\mu_{A \cup B}(x) = \mu_A(x) + \mu_B(x) - \mu_{A \cap B}(x)$,

$\mu_{A \cap B}(x) = \mu_A(x) \cdot \mu_B(x)$,

$\mu_{\neg A}(x) = 1 - \mu_A(x)$.

4) **Драстическая логика:**

$$\mu_{A \cup B}(x) = \begin{cases} \mu_A(x), & \mu_B(x) = 0 \\ \mu_B(x), & \mu_A(x) = 0 \\ 1, & \text{иначе} \end{cases}$$

$$\mu_{A \cap B}(x) = \begin{cases} \mu_A(x), & \mu_B(x) = 1 \\ \mu_B(x), & \mu_A(x) = 1 \\ 0, & \text{иначе} \end{cases}$$

Определение 48.2.

Двухместная функция $f\colon [0,1]^2 \to [0,1]$ называется **t-нормой**, если выполняются следующие условия:

1) $f(x,1) = x$ и $f(x,0) = 0$;

2) $f(x,y) = f(y,x)$;

3) $f(x, f(y,z)) = f(f(x,y), z)$;

4) $x_1 \leqslant x_2 \Rightarrow f(x_1, y) \leqslant f(x_2, y)$.

Определение 48.3.

Двухместная функция $g\colon [0,1]^2 \to [0,1]$ называется **t-конормой**, если выполняются следующие условия:

1) $g(x,1) = 1$ и $g(x,0) = x$;

2) $g(x,y) = g(y,x)$;

3) $g(x, g(y,z)) = g(g(x,y), z)$;

4) $x_1 \leqslant x_2 \Rightarrow g(x_1, y) \leqslant g(x_2, y)$.

В настоящее время в нечеткой логике в качестве операций конъюнкции и дизъюнкции широко используются t-нормы и t-конормы соответственно, пришедшие в нечёткую логику из теории вероятностных метрических пространств. Эти операции достаточно хорошо изучены и лежат в основе многих формальных построений нечеткой логики.

Определение 48.4.

Одноместная функция $N\colon [0,1] \to [0,1]$ называется **нечётным отрицанием**, если выполняются следующие условия:

1) $N(N(x)) = x$;

2) $N(0) = 1$ и $N(1) = 0$;

3) $x_1 \leqslant x_2 \Rightarrow N(x_2) < N(x_1)$.

Определение 48.5.

Законы де Моргана в нечёткой логике выглядят следующим образом:

$N(f(x,y)) = g(N(x), N(y))$,

$N(g(x,y)) = f(N(x), N(y))$.

Предложение 48.6.

1) $\forall f$ - t-норм $\forall x, y \in [0,1]$: $f(x,y) \leqslant min\{x,y\}$;

2) $\forall g$ - t-конорм $\forall x, y \in [0,1]$: $g(x,y) \geqslant max\{x,y\}$.

Доказательство:

1) $\left.\begin{array}{l} f(x,y) \leqslant f(x,1) = x \\ f(x,y) \leqslant f(1,y) = y \end{array}\right\} \Rightarrow f(x,y) \leqslant min\{x,y\}$.

2) Аналогично 1-му.

Предложение доказано.

Предложение 48.7.

1) $\forall f$ - t-норм $f(x,y) \geqslant f_d(x,y)$, где $f_d(x,y)$ - драстическая t-норма;

2) $\forall g$ - t-конорм $g(x,y) \leqslant g_d(x,y)$, где $g_d(x,y)$ - драстическая t-конорма.

Драстические t-конормы и t-нормы описаны в 48.1. соответственно.

Доказательство:

Рассмотрим 3 возможных случая:

1) Пусть $x \neq 1, y \neq 1$, тогда $f(x,y) \geqslant f_d(x,y) = 0$;

2) Пусть $x = 1$, тогда $f(1,y) = y = f_d(1,y)$;

3) Пусть $y = 1$, тогда $f(x,1) = x = f_d(x,1)$.

Для t-конорм аналогично.

Предложение доказано.

Предложение 48.8.

Нормы Заде - единственные **идемпотентные нормы**, т.е. $f(x,x) = x$.

Доказательство:

Докажем от противного. Допустим, что f-идемпотентна, а норма Заде f_z - нет. Тогда $\forall x \in [0,1]: f(x,x) = x$.

Пусть $f \neq f_z \Rightarrow \exists x, y: x < y \;\&\; f(x,y) \neq f_z(x,y) \Rightarrow f(x,y) < f_z(x,y) \Rightarrow$
$\Rightarrow f(x,x) \leqslant f(x,y) < f_z(x,y) = x \Rightarrow f(x,x) < x \Rightarrow$ получаем противоречие с начальными условиями $\Rightarrow f_z$ - единственные идемпотентные.

Предложение доказано.

Идемпотентность соблюдается так же только у t-конорм Заде.

К нечётной логике сначала научное сообщество отнеслось скептически, поскольку нечётная логика противоречила логике самого Аристотеля, которой люди руководствовались на протяжении многих веков. Мотивацией создания подобной логики стал "*принцип несовместимости*" Заде, который гласил "*чем сложнее система, тем сложнее дать точные и в то же время имеющие практическое значение суждения об её поведении*". Т.е. потребовался инструмент, что не нуждался в абсолютной строгости ответа и с помощью которого можно было бы обрабатывать неточные данные.

Нечёткая логика эту возможность и дала. Но не смотря на то, что она была придумана в 1965 году, полное признание она получила лишь 20 лет спустя после доказательства Бартоломеем Коско знаменитой теоремы FAT(Fuzzy Approximation Theorem). В бизнесе и финансах нечеткая логика получила признание после того, как в 1988 году экспертная система на основе нечетких правил для прогнозирования финансовых индикаторов единственная предсказала биржевой крах. И количество успешных нечётких-применений в на-

стоящее время исчисляется тысячами.

На данный момент широкое применение нечёткая логика имеет в нейронных сетях, базах данных, *мягких вычислениях*, гибридных интеллектуальных системах и т.д..

Но несмотря на такие огромные прорывы благодаря нечёткой логике, она так же имеет весомые недостатки и свои парадоксы. И она по сей день имеет противников. Но тем не менее, благодаря достижениям нечёткой логики в обработке поисковых запросов, я смог Вам склепать столь прекрасное творение под названием "*методичка по ЛОПу*" :)

Мы, мужчины, не умеем сказать «прощай», оставаясь при этом джентльменами. Может, Кэри Грант или Дэвид Нивен знали, как это делается, но я сомневаюсь, что им было под силу сказать: «Прощай, математическая логика», — сохраняя при этом обаяние.

Большим М.А., 2020