Глава 4

Разрешимость логики $\mathcal{ALC}$

Здесь мы установим разрешимость всех логических проблем, которые были перечислены в предыдущем разделе, для логики $\mathcal{ALC}$. Для этого мы опишем так называемый *табло-алгоритм* (tableau algorithm), который проверяет выполнимость баз знаний в этой логике. Чтобы общее описание алгоритма было легче воспринимать, мы сначала посмотрим, как он работает на конкретном примере.

Пример 4.1. Пусть мы хотим проверить, верно ли вложение концептов:

$$\exists R.A \sqcap \neg\forall R.(\neg B \sqcap A) \;\sqsubseteq\; \exists R.B.$$

Согласно лемме 2.2, для этого мы должны проверить на выполнимость концепт

$$C \;:=\; \exists R.A \sqcap \neg\forall R.(\neg B \sqcap A) \sqcap \neg\exists R.B.$$

Если ответ на этот вопрос будет «да», то ответ на исходный вопрос будет «нет» и наоборот.

Предварительный шаг: *нормализуем* концепт C, то есть «пронесем» все отрицания внутрь с тем, чтобы получить эквивалентный концепт, в котором все отрицания стоят лишь перед атомарными концептами. Это всегда можно сделать, пользуясь законами де-Моргана ($\neg(C \sqcap D) \equiv (\neg C \sqcup \neg D)$ и т.п.), законами двойственности ($\neg\exists R.D \equiv \forall R.\neg D$ и т.п.) и законом снятия двойного отрицания ($\neg\neg C \equiv C$). В нашем случае мы получим нормализованный концепт:

$$C_0 \;=\; \exists R.A \sqcap \exists R.(B \sqcup \neg A) \sqcap \forall R.\neg B.$$

Будем пытаться строить модель, в которой этот концепт выполним (непуст). Создадим начальную точку x с условием **(0)** $x\colon C_0$. Далее из этого условия мы будем выводить новые условия на строящуюся модель — и записывать их в «протокол», имеющий вид обычного ABox. Итак, изначально имеем ABox $\mathcal{A} := \{x\colon C_0\}$. Поскольку C_0 есть конъюнкция трех концептов, то x должен принадлежать всем трем, поэтому дописываем в $\mathcal{A}$ следующие факты: **(1)** $x\colon \exists R.A$, **(2)** $x\colon \exists R.(B \sqcup \neg A)$, **(3)** $x\colon \forall R.\neg B$.

Ввиду условия **(1)** должна существовать точка y, связанная с x отношением R, в которой выполнен концепт A. Поэтому мы добавляем в $\mathcal{A}$ следующие факты: **(4)** xRy, **(5)** $y\colon A$.

Аналогично, из условия **(2)** следует существование точки z, связанной с x отношением R, в которой выполнен концепт $B \sqcup \neg A$. Поэтому добавляем в $\mathcal{A}$ следующие факты: **(6)** xRz, **(7)** $z\colon B \sqcup \neg A$.

Далее, ввиду условия **(3)** во всех точках, связанных с x отношением R (в нашем случае это точки y и z), должен быть выполнен концепт $\neg B$. Поэтому добавляем в $\mathcal{A}$ следующие факты: **(8)** $y\colon \neg B$, **(9)** $z\colon \neg B$.

Остается разобрать условие **(7)**. Так как точка z принадлежит дизъюнкции (объединению) концептов, то она принадлежит одному или другому концепту. Поэтому нужно рассмотреть два случая. Первый случай: **(7′)** $z\colon B$ — немедленно приводит к противоречию с условием **(9)**, согласно которому $z\colon \neg B$. Второй же случай: **(7″)** $z\colon \neg A$ — никаких явных противоречий не создает.

Итак, мы довели все условия до элементарных — и пришли к протоколу (ABox $\mathcal{A}$), в котором никаких явных противоречий нет. Можно показать, что этот ABox $\mathcal{A}$, а вместе с ним и исходный концепт C_0, имеет модель. Фактически сам ABox и представляет модель: нужно все созданные в процессе работы алгоритма точки считать элементами области интерпретации, а присутствующие в $\mathcal{A}$ элементарные факты (вида xRy или $y\colon A$) считать заданием интерпретации атомарных концептов и ролей:

$$\mathcal{I} = (\Delta, \cdot^{\mathcal{I}}),\ \text{где } \Delta = \{x, y, z\},\ \ A^{\mathcal{I}} = \{y\},\ \ B^{\mathcal{I}} = \varnothing,\ \ R^{\mathcal{I}} = \{\langle x, y\rangle, \langle x, z\rangle\}.$$

В такой модели все факты из ABox будут верными. Следовательно, концепт C выполним, а исходное включение концептов — неверно.

Замечание 4.1. При создании точки z мы, строго говоря, должны были рассмотреть случаи, когда она совпадает с какой-либо ранее построенной (в нашем случае — с точкой x или y) и когда она является новой точкой. Однако на самом деле это излишне, так как последний случай покрывает предыдущие. Действительно, если при совпадении точки z с какой-то ранее введенной мы сможем построить модель, то в ситуации, когда z — новая точка, мы тем более сможем построить модель, поскольку во этом случае на точку z (а тем самым и на модель) накладывается меньше ограничений. Это соображение будет использоваться при описании табло-алгоритма в общем случае.

4.1 Понятие разрешающего алгоритма

Прежде чем описывать алгоритм и доказывать, что он «решает» нужную проблему, дадим этому понятию формальное определение. Обычно в математической логике, после того, как задан язык и его семантика, пытаются построить исчисление, выводящее те и только те высказывания в этом языке, которые являются «верными» в этой семантике. Если «верность» (например, общезначимость) формулы φ обозначать как $\models \varphi$, а выводимость ее в исчислении обозначать как $\vdash \varphi$, то вводятся два понятия:

- исчисление называется *корректным*, если для любой формулы φ из $\vdash \varphi$ следует $\models \varphi$;
- исчисление называется *полным*, если для любой формулы φ из $\models \varphi$ следует $\vdash \varphi$.

Другими словами, корректное исчисление выводит *только* «верные» формулы; полное — выводит *все* «верные» формулы.

В дескрипционной логике традиционно отсутствуют исчисления как таковые. Вместо этого для заданной логики пытаются построить алгоритм, проверяющий, например, истинность аксиом или выполнимость концептов. Пусть зафиксирована ДЛ $\mathcal{L}$, и требуется построить алгоритм $\mathfrak{A}$, который бы проверял вложение заданных концептов $C \sqsubseteq D$ относительно заданной терминологии $\mathcal{T}$, где $C, D, \mathcal{T}$ формулируются в языке $\mathcal{L}$. Напомним, что это отношение обозначается как $\mathcal{T} \models C \sqsubseteq D$. Ответ алгоритма $\mathfrak{A}$ на входных данных $(C, D, \mathcal{T})$ будем записывать как $\mathfrak{A}(C, D, \mathcal{T})$ и будем считать, что алгоритм на любом входе работает лишь конечное время и выдает только ответы 0 и 1. Тогда, по аналогии с приведенным выше определением, мы будем говорить, что алгоритм $\mathfrak{A}$ проверки вложения концептов:

- является *корректным*, если для любых $C, D, \mathcal{T}$ из $\mathfrak{A}(C, D, \mathcal{T}) = 1$ следует $\mathcal{T} \models C \sqsubseteq D$;
- является *полным*, если для любых $C, D, \mathcal{T}$ из $\mathcal{T} \models C \sqsubseteq D$ следует $\mathfrak{A}(C, D, \mathcal{T}) = 1$.

Однако, как уже говорилось ранее, в ДЛ традиционно разрабатывают алгоритмы для проверки *выполнимости* концептов. Вспомним, что вложенность концептов $C \sqsubseteq D$ эквивалентна *не*выполнимости концепта C (лемма 2.2); соответственно, и ответы алгоритма будут противоположны тем, что давались при проверке вложенности концептов. Поэтому для проблемы выполнимости вышеприведенные понятия «меняются местами», и мы приходим к следующему определению.

Определение 4.1. Алгоритм $\mathfrak{A}$ является *разрешающей процедурой* для проблемы выполнимости концептов относительно терминологий для ДЛ $\mathcal{L}$, если выполнены следующие три условия:

Завершаемость: для любых $(C, \mathcal{T})$ алгоритм $\mathfrak{A}$ выдает ответ $\mathfrak{A}(C, \mathcal{T})$ через конечное время;

Корректность: для любых $(C, \mathcal{T})$ если концепт C выполним относительно $\mathcal{T}$, то $\mathfrak{A}(C, \mathcal{T}) = 1$;

Полнота: для любых $(C, \mathcal{T})$ если $\mathfrak{A}(C, \mathcal{T}) = 1$, то концепт C выполним относительно $\mathcal{T}$.

4.2 Табло-алгоритм для логики $\mathcal{ALC}$ без терминологий

Пусть дан концепт C_0 логики $\mathcal{ALC}$, выполнимость которого требуется выяснить. Без ограничения общности можно считать, что C_0 уже нормализован, т.е. все встречающиеся в нем символы отрицания стоят перед атомарными концептами. Строим начальный ABox $\mathcal{A}_0 = \{x_0 : C_0\}$. Далее на каждом шаге к текущему ABox $\mathcal{A}$ применяется правило — и получается один или два новых ABox. Список правил приведен в Таблице 4.1. Порядок применения правил произволен[1] (и в этом большой простор для выбора стратегий с целью оптимизации алгоритма). Обратите внимание, что правила для $\sqcap$, $\exists$, $\forall$ из текущего ABox $\mathcal{A}$ создают один новый ABox $\mathcal{A}'$, тогда как $\sqcup$-правило из ABox $\mathcal{A}$ создает два новых ABox $\mathcal{A}'$ и $\mathcal{A}''$ (и далее правила применяются к каждому из них «независимо»).

[1] В случае логики $\mathcal{ALC}$ правильность работы алгоритма не зависит от порядка применения правил. Однако это уже не так для многих других ДЛ, где требуется выбрать определенную стратегию для обеспечения завершаемости, корректности или полноты табло-алгоритма.

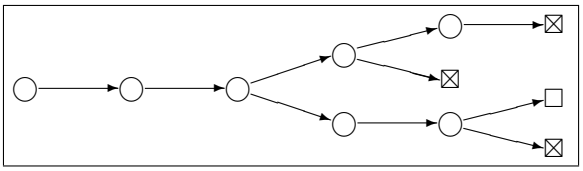
Правило	Условия применения	Действие
$\sqcap$-правило	если 1. $x\colon (C \sqcap D) \in \mathcal{A}$ 2. $x\colon C \notin \mathcal{A}$ или $x\colon D \notin \mathcal{A}$	то $\mathcal{A}' := \mathcal{A} \cup \{x\colon C,\ x\colon D\}$
$\sqcup$-правило	если 1. $x\colon (C \sqcup D) \in \mathcal{A}$ 2. $x\colon C \notin \mathcal{A}$ и $x\colon D \notin \mathcal{A}$	то $\mathcal{A}' := \mathcal{A} \cup \{x\colon C\}$ $\mathcal{A}'' := \mathcal{A} \cup \{x\colon D\}$
$\exists$-правило	если 1. $x\colon \exists R.C \in \mathcal{A}$ 2. нет такого y, что $xRy \in \mathcal{A}$ и $y\colon C \in \mathcal{A}$	то создать новую точку y и $\mathcal{A}' := \mathcal{A} \cup \{xRy,\ y\colon C\}$
$\forall$-правило	если 1. $x\colon \forall R.C \in \mathcal{A}$ 2. есть такой y, что $xRy \in \mathcal{A}$ и $y\colon C \notin \mathcal{A}$	то $\mathcal{A}' := \mathcal{A} \cup \{y\colon C\}$

Таблица 4.1: Правила табло-алгоритма для логики $\mathcal{ALC}$.

Таким образом, из исходного ABox $\mathcal{A}_0$ путем применения описанных правил будет построено дерево (назовем его *дерево поиска*), у которого в корне находится $\mathcal{A}_0$ и у каждого ABox есть 0, 1 или 2 последователя. Применение правил прекращается, если к очередному ABox $\mathcal{A}$ не применимо ни одно из правил, либо если в $\mathcal{A}$ содержится явное противоречие (см. ниже), так что применять дальнейшие правила в надежде построить модель не имеет смысла. Эти два вида ABox будут листьями дерева поиска (назовем их *листовыми* ABox). Введем соответствующие термины:

- ABox $\mathcal{A}$ называется *противоречивым*, если он содержит факты $x\colon A$ и $x\colon \neg A$ для некоторого индивида x и атомарного концепта A, либо он содержит факт $x\colon \bot$ для некоторого индивида x;
- ABox $\mathcal{A}$ называется *полным непротиворечивым*, если он не является противоречивым, но ни одно из правил табло-алгоритма к нему не применимо.

Дерево поиска можно схематично изобразить следующим образом, где мы обозначили символом ◯ — произвольный ABox, ⊠ — противоречивый ABox, □ — полный непротиворечивый ABox:



Если алгоритм встречает полный непротиворечивый ABox, то он выдает ответ 1 (мы докажем, что в этом случае существует модель) и оставшиеся ветви дерева поиска уже не обходит. Напротив, если алгоритм встречает противоречивый ABox, то это лишь означает, что данная ветвь дерева поиска не привела к модели; тогда алгоритм продолжает строить остальные ветви дерева поиска. Наконец, когда алгоритм обошел всё дерево поиска и оказалось, что все листовые ABox противоречивы, алгоритм выдает ответ 0 (мы докажем, что в этом случае моделей нет). Итак, по построению табло-алгоритм возаращает значение 1 $\Longleftrightarrow$ хотя бы один из листовых ABox является полным непротиворечивым (и возвращает 0 в противном случае).

Формально табло-алгоритм можно описать в виде рекурсивной процедуры $\mathsf{SAT}(\mathcal{A})$,[2] описанной в Таблице 4.2. Она принимает на вход произвольный ABox $\mathcal{A}$ (с условием, что все концепты в нем нормализованы — а исходный $\mathcal{A}_0$ как раз такой) и выдает ответ 0 или 1.

Теперь наша задача показать, что данный алгоритм действительно решает проблему выполнимости концептов логики $\mathcal{ALC}$. Предварительно заметим, что каждый ABox $\mathcal{A}$ можно рассматривать как размеченный граф (G, L). Вершинами графа G являются индивиды x, встречающиеся в $\mathcal{A}$; ребра соответствуют имеющимся в $\mathcal{A}$ фактам вида xRy; меткой $L(x)$ вершины x является множество таких концептов C, что в $\mathcal{A}$ имеется факт $x\colon C$, то есть $L(x) = \{C \mid x\colon C \in \mathcal{A}\}$. Ниже мы увидим, что для полного непротиворечивого ABox этот граф фактически представляет собой искомую модель, для поиска которой и предназначен табло-алгоритм.

Лемма 4.1 (Завершаемость)**.** *Не существует бесконечной цепочки $\mathcal{A}_0, \mathcal{A}_1, \ldots$, в которой каждый ABox $\mathcal{A}_{i+1}$ получен из $\mathcal{A}_i$ по какому-либо правилу табло-алгоритма.*

Доказательство. Данное утверждение следует из того, что в процессе применения правил табло-алгоритма структура (G, L), соответствующая ABox $\mathcal{A}$, «монотонно возрастает» и «ограничена сверху» (тем самым,

[2] Уточним, что используемый в тексте процедуры оператор return d возвращает значение d в качестве результата и прекращает выполнение процедуры (т.е. написанные после него операторы не выполняются). В частности, выполнение приведенной процедуры дойдет до последнего оператора return 0 только в том случае, если ни один из предыдущих операторов return не выполнился.

```
Функция SAT(A)
{ если в A есть x:⊥ для некоторого x, то return 0;
  если в A есть x:A и x:¬A для некоторых x и A, то return 0;
  если к A не применимо ни одно из правил, то return 1;
  если применимо правило ⊓, ∃ или ∀, то применить любое, получив A', и return SAT(A');
  если применимо ⊔-правило, то применить его, получив A' и A'', и далее:
   { если SAT(A') = 1, то return 1;
     если SAT(A'') = 1, то return 1; }
  return 0;
}
```

Таблица 4.2: Табло-алгоритм для логики $\mathcal{ALC}$.

процесс роста может длиться лишь конечное число шагов). Монотонность правил очевидна — они лишь добавляют что-либо в ABox, но ничего не удаляют. Остается установить ограниченность. Она следует из ряда утверждений:

- граф G является деревом с корнем[3] x_0. Действительно, изначально граф состоит из одной вершины x_0, а новые вершины порождаются лишь $\exists$-правилом, которое соединяет вновь созданную вершину ровно с одной из уже существующих вершин.
  Как следствие, из корня x_0 в любую вершину $x \neq x_0$ ведет единственная цепочка ребер. Длину этой цепочки (число ребер в ней) будем называть *уровнем* вершины x.
- Ширина[4] дерева G ограничена (количеством кванторов $\exists$ в исходном концепте; обозначим его N). Действительно, ребра, исходящие из какой-либо вершины, создаются $\exists$-правилами, которые, в свою очередь, соответствуют концептам вида $\exists R.C$, встречающимся в C_0.
- Глубина[5] дерева G ограничена (кванторной сложностью[6] концепта C_0; обозначим ее $M := d(C_0)$). Это следует из того, что кванторная сложность концептов в метках вершин уменьшается с увеличением уровня вершины. Формально, если $x{:}\,C \in \mathcal{A}$ и x находится на уровне ℓ, то $d(C) \leqslant M - \ell$. Данное утверждение доказывается индукцией по ℓ. База индукции очевидна; шаг индукции следует из того, как сформулированы $\exists$- и $\forall$-правила: если на каком-то уровне были концепты вида $\exists R.C$ и $\forall R.C$, то эти правила на следующий уровень переносят лишь концепт C (сложности на 1 меньшей). Таким образом, общее число вершин в графе G не превышает $1 + N + N^2 + \cdots + N^M$.
- Метка каждой вершины ограничена: $L(x) \subseteq \mathsf{Sb}(C_0)$, где $\mathsf{Sb}(C_0)$ есть множество подконцептов[7] концепта C_0. Действительно, любое правило добавляет в метку любой вершины лишь подконцепты уже имевшихся там концептов.

Таким образом, размер структуры (G, L) ограничен сверху, а значит, таких структур конечное число. ◀

Лемма 4.2 (Корректность)**.** *Справедливы следующие утверждения:*

(a) *Концепт C_0 выполним $\iff$ ABox $\mathcal{A}_0 = \{x_0{:}\,C_0\}$ выполним.*

(b) *Пусть $\mathcal{A}'$ получен из $\mathcal{A}$ по одному из правил: $\sqcap, \exists, \forall$. Тогда если $\mathcal{A}$ выполним, то $\mathcal{A}'$ выполним.*

(c) *Пусть $\mathcal{A}'$ и $\mathcal{A}''$ получены из $\mathcal{A}$ по $\sqcup$-правилу. Тогда если $\mathcal{A}$ выполним, то $\mathcal{A}'$ или $\mathcal{A}''$ выполним.*

Доказательство. Пункт (a) очевиден. Утверждения пунктов (b) и (c), касающиеся правил для $\sqcap$ и $\sqcup$, доказываются легко, поэтому мы остановимся на утверждениях, касающихся правил для $\exists$ и $\forall$.

Пусть $\mathcal{A}'$ получен из $\mathcal{A}$ по $\exists$-правилу. Если ABox $\mathcal{A}$ выполним, то существует его модель $\mathcal{I}$. Ввиду 1-й предпосылки $\exists$-правила, имеем $x^{\mathcal{I}} \in (\exists R.C)^{\mathcal{I}}$. По определению семантики это означает, что существует $d \in \Delta$ такой, что $(x^{\mathcal{I}}, d) \in R^{\mathcal{I}}$ и $d \in C^{\mathcal{I}}$. Чтобы построить модель для $\mathcal{A}'$, расширим $\mathcal{I}$, положив $y^{\mathcal{I}} := d$. Тогда $(x^{\mathcal{I}}, y^{\mathcal{I}}) \in R^{\mathcal{I}}$ и $y^{\mathcal{I}} \in C^{\mathcal{I}}$. Тем самым $\mathcal{I} \models xRy$ и $\mathcal{I} \models y{:}\,C$, то есть $\mathcal{I}$ является моделью $\mathcal{A}'$.

Пусть $\mathcal{A}'$ получен из $\mathcal{A}$ по $\forall$-правилу. Если ABox $\mathcal{A}$ выполним, то существует его модель $\mathcal{I}$. Ввиду 1-й и 2-й предпосылок $\forall$-правила, имеем $x^{\mathcal{I}} \in (\forall R.C)^{\mathcal{I}}$ и $(x^{\mathcal{I}}, y^{\mathcal{I}}) \in R^{\mathcal{I}}$. Из первого следует, что для любого

[3] Напомним, что *деревом* называется связанный ориентированный граф, в котором каждая вершина, кроме одной (называемой корнем), имеет ровно одного непосредственного предшественника.

[4] *Шириной* дерева называется максимум по всем вершинам x количества ребер, выходящих из x.

[5] *Глубиной* дерева называется максимальный уровень его вершин, т.е. наибольшая длина цепи, ведущей из корня в лист.

[6] *Кванторная сложность* $d(C)$ концепта C определяется как максимальная глубина вложенности кванторов в нем. Формально, она определяется индуктивно: $d(\bot) = d(\top) = d(A) = 0$, где A — атомарный концепт; $d(\neg C) = d(C)$, $d(C \sqcap D) = d(C \sqcup D) = \max\{d(C), d(D)\}$, $d(\exists R.C) = d(\forall R.C) = 1 + d(C)$.

[7] *Подконцептами* концепта C называются все концепты, из которых он был построен. Формально, множество $\mathsf{Sb}(C)$ всех подконцептов концепта C определяется индуктивно: $\mathsf{Sb}(\top) = \{\top\}$, $\mathsf{Sb}(\bot) = \{\bot\}$, $\mathsf{Sb}(A) = \{A\}$ для атомарных концептов A, $\mathsf{Sb}(\neg C) = \{\neg C\} \cup \mathsf{Sb}(C)$, $\mathsf{Sb}(C \sqcap D) = \{C \sqcap D\} \cup \mathsf{Sb}(C) \cup \mathsf{Sb}(D)$, $\mathsf{Sb}(\exists R.C) = \{\exists R.C\} \cup \mathsf{Sb}(C)$, аналогично для $\sqcup$ и $\forall$.

элемента $d \in \Delta$, такого что $(x^{\mathcal{I}}, d) \in R^{\mathcal{I}}$, имеет место $d \in C^{\mathcal{I}}$. В частности, это верно для $d = y^{\mathcal{I}}$, значит, $y^{\mathcal{I}} \in C^{\mathcal{I}}$. Тем самым $\mathcal{I} \models y{:}\, C$, то есть $\mathcal{I}$ является моделью $\mathcal{A}'$. ◀

Лемма 4.3 (Полнота)**.** *Всякий полный непротиворечивый ABox выполним.*

Доказательство. Пусть ABox $\mathcal{A}$ полный и непротиворечивый. Построим его *каноническую* модель $\mathcal{I}$:

$$\begin{aligned}\Delta &:= \{x \mid \text{индивид } x \text{ встречается в } \mathcal{A}\},\\ A^{\mathcal{I}} &:= \{x \in \Delta \mid x{:}\, A \in \mathcal{A}\} \text{ для каждого атомарного концепта } A,\\ R^{\mathcal{I}} &:= \{(x,y) \in \Delta \times \Delta \mid xRy \in \mathcal{A}\} \text{ для каждой атомарной роли } R.\end{aligned}$$

Докажем, что $\mathcal{I} \models \mathcal{A}$, т.е. $\mathcal{I} \models \alpha$ для каждого факта $\alpha \in \mathcal{A}$. Для фактов вида xRy это следует из построения $R^{\mathcal{I}}$. Для фактов же вида $x{:}\, C$ индукцией по построению концепта C докажем требуемую импликацию:

$$\text{для любого } x \in \Delta\text{:} \qquad \boxed{\text{если } x{:}\, C \in \mathcal{A}\text{, то } x \in C^{\mathcal{I}}} \qquad (\star)$$

Все концепты в ABox $\mathcal{A}$ нормализованы, поэтому всякий концепт построен из концептов вида $\top, \bot, A, \neg A$ (где A атомарный) с помощью связок $\sqcap, \sqcup, \exists, \forall$. База индукции:

- Очевидно, для $\bot$ посылка импликации $(\star)$ ложна, а для $\top$ заключение импликации $(\star)$ истинно.
- Если $x{:}\, A \in \mathcal{A}$, то по построению $A^{\mathcal{I}}$ имеем $x \in A^{\mathcal{I}}$.
- Если $x{:}\, \neg A \in \mathcal{A}$, то $x \in (\neg A)^{\mathcal{I}}$, поскольку иначе $x \in A^{\mathcal{I}}$, откуда по определению $A^{\mathcal{I}}$ следовало бы $x{:}\, A \in \mathcal{A}$, что вместе с $x{:}\, \neg A \in \mathcal{A}$ означало бы, что $\mathcal{A}$ противоречив, вопреки условию леммы.

Шаг индукции. Случаи связок $\sqcap$ и $\sqcup$ разбираются тривиально; остановимся на кванторах.

- Случай $C = \exists R.D$. Пусть $x{:}\, C \in \mathcal{A}$, то есть $x{:}\, \exists R.D \in \mathcal{A}$, тем самым выполнена 1-я предпосылка $\exists$-правила. Так как ABox $\mathcal{A}$ полный, то $\exists$-правило к нему не применимо, значит не выполнена его 2-я предпосылка, то есть существует такой $y \in \Delta$, что $xRy \in \mathcal{A}$ и $y{:}\, D \in \mathcal{A}$. Отсюда следует $(x,y) \in R^{\mathcal{I}}$ по построению $R^{\mathcal{I}}$, а также $y \in D^{\mathcal{I}}$ по предположению индукции для D. Тем самым $x \in (\exists R.D)^{\mathcal{I}}$ по определению семантики квантора $\exists$.
- Случай $C = \forall R.D$. Пусть $x{:}\, C \in \mathcal{A}$, то есть $x{:}\, \forall R.D \in \mathcal{A}$, тем самым выполнена 1-я предпосылка $\forall$-правила. Чтобы доказать, что $x \in (\forall R.D)^{\mathcal{I}}$, возьмем произвольный $y \in \Delta$ такой, что $(x,y) \in R^{\mathcal{I}}$, и покажем, что $y \in D^{\mathcal{I}}$. По предположению индукции для R мы имеем $xRy \in \mathcal{A}$. Поскольку ABox $\mathcal{A}$ полный, то $\forall$-правило к нему не применимо, значит не выполнена его 2-я предпосылка, то есть имеет место $y{:}\, D \in \mathcal{A}$. Отсюда по предположению индукции для D вытекает $y \in D^{\mathcal{I}}$, что и требовалось.

Итак, мы построили модель $\mathcal{I}$ ABox $\mathcal{A}$, откуда следует, что $\mathcal{A}$ выполним. ◀

Теперь мы готовы доказать основную теорему.

Теорема 4.4 (Разрешимость $\mathcal{ALC}$)**.** *Проблема выполнимости концептов логики $\mathcal{ALC}$ разрешима. А именно, табло-алгоритм является разрешающей процедурой для этой проблемы.*

Доказательство. Требуется доказать завершаемость, корректность и полноту табло-алгоритма.

Завершаемость: Из леммы 4.1 следует, что дерево поиска не имеет бесконечных цепей, а так как степень его ветвления ограничена (не более 2), то дерево поиска конечно.[8] Как следствие, для любых входных данных табло-алгоритм обойдет это дерево за конечное время и выдаст ответ.

Остается доказать корректность и полноту, т.е. эквивалентность: $\mathcal{A}_0$ выполним $\iff \mathsf{SAT}(\mathcal{A}_0) = 1$.

Корректность: Если $\mathcal{A}_0$ выполним, то по лемме 4.2(b,c) хотя бы один из листовых ABox $\mathcal{A}$ выполним. Он не может быть противоречивым, поскольку ни $\{x{:}\, A, x{:}\, \neg A\}$, ни $\{x{:}\, \bot\}$ не является выполнимым. Значит, $\mathcal{A}$ есть полный непротиворечивый. По построению, табло-алгоритм в этом случае выдает ответ 1.

Полнота: Пусть табло-алгоритм на входе $\mathcal{A}_0$ выдал ответ 1. Значит, он нашел полный непротиворечивый ABox $\mathcal{A}$. По лемме 4.3 ABox $\mathcal{A}$ выполним. Ввиду того, что $\mathcal{A}_0 \subseteq \mathcal{A}$ (ибо правила табло-алгоритма лишь добавляют факты в ABox, ничего не удаляя), мы заключаем, что $\mathcal{A}_0$ выполним. ◀

Теорема 4.5. *Проблема выполнимости ABox в логике $\mathcal{ALC}$ разрешима.*

Доказательство. Достаточно применить тот же табло-алгоритм, но не к ABox вида $\{x_0{:}\, C_0\}$, а к произвольному ABox $\mathcal{A}_0$. Единственное, о чем нужно позаботиться — нормализовать все концепты в ABox. Все доказательства сохраняют силу, за исключением леммы 4.1, где вид графа G будет чуть сложнее: вместо дерева мы будем иметь произвольный граф (изначальный ABox $\mathcal{A}_0$), из вершин которого «растут» деревья, строимые табло-алгоритмом. В оценках размера графа G и его меток нужно теперь учитывать не единственный концепт C_0, а всю совокупность концептов, встречающихся в ABox $\mathcal{A}_0$. ◀

[8] На самом деле, это частный случай общей **Леммы Кёнига**: если в ориентированом дереве число вершин бесконечно, а степень ветвления каждой вершины конечна, то существует бесконечная ориентированная цепь.

4.3 Табло-алгоритм для логики $\mathcal{ALC}$ с терминологиями

Теперь мы опишем алгоритм для решения более сложной проблемы: дан концепт C_0 и терминология $\mathcal{T}$; требуется проверить, выполним ли C_0 относительно $\mathcal{T}$. Без ограничения общности, концепт C_0 нормализован (то есть в нем все отрицания стоят лишь перед атомарными концептами), а также $\mathcal{T}$ состоит из единственной аксиомы $\top \sqsubseteq E$, где концепт E тоже нормализован.

Итак, даны концепты C_0 и E и требуется выяснить, существует ли интерпретация $\mathcal{I}$, в которой $E^{\mathcal{I}} = \Delta$ и $C_0^{\mathcal{I}} \neq \varnothing$. Как и раньше, берем начальный ABox $\mathcal{A}_0 := \{x_0 \colon C_0\}$. На первый взгляд может показаться, что достаточно лишь добавить к описанному ранее табло-алгоритму дополнительное правило — помещающее концепт E в каждую точку x в ABox. Однако оказывается, что при этом мы можем потерять условие завершаемости. Это и следовало заподозрить, поскольку завершаемость прежнего табло-алгоритма опиралась на то, что кванторная сложность концептов в метках вершин уменьшается по мере увеличения глубины вершины; здесь же мы в каждую точку помещаем концепт E одной и той же кванторной сложности. Подозрения действительно оправдываются, как показывает следующий контрпример.

Пример 4.2. Проверим таким способом выполнимость концепта $A \sqcap B$ относительно терминологии $\mathcal{T} = \{\top \sqsubseteq \exists R.A\}$. Стартовый ABox $\mathcal{A}_0 = \{x_0 \colon (A \sqcap B)\}$. Раскрывая конъюнкцию, мы добавляем в ABox факты $x_0 \colon A$ и $x_0 \colon B$. Добавим также факт $x_0 \colon \exists R.A$.

Применим теперь $\exists$-правило к точке x_0, в результате чего создадим точку x_1 и добавим в ABox факты $x_0 R x_1$ и $x_1 \colon A$. Добавим также факт $x_1 \colon \exists R.A$. Аналогично будет создана точка x_2, такая что $x_1 R x_2$, с метками $x_2 \colon A$ и $x_2 \colon \exists R.A$; затем будет создана точка x_3 и т.д. Как видим, процесс «зациклился» — он создает точки с одинаковыми метками.

Чтобы этого не происходило, нужно отслеживать подобные циклы и прерывать их. В нашем случае метка точки x_1 содержится в метке точки x_0, а именно $L(x_1) = \{A, \exists R.A\} \subseteq \{A, B, \exists R.A\} = L(x_0)$. Это является признаком того, что дальше эту ветвь продолжать не нужно. В такой ситуации мы будем говорить, что точка x_1 *блокирована* (точкой x_0).

Заметим, что условие блокировки можно было, в принципе, сформулировать не как включение, а как равенство меток; такой алгоритм тоже будет работать правильно и всегда завершаться. Однако, в этом случае он зачастую будет делать лишнюю работу, что мы и наблюдаем в нашем примере: прежде чем прервать цикл, пришлось бы построить точку x_2, метка которой в точности совпадёт с меткой x_1.

Таким образом, прежде чем формулировать табло-алгоритм, нам требуется ввести новые понятия. Пусть на очередном шаге алгоритма имеется ABox $\mathcal{A}$. Меткой точки x в ABox $\mathcal{A}$ будем называть множество $L(x) = \{C \mid x \colon C \in \mathcal{A}\}$. Напомним также, что в ориентированном графе вершина x называется *предком* вершины y, а вершина y — *потомком* вершины x, если из x в y ведет ориентированный путь.

Определение 4.2 (Блокировка)**.** Точка x *блокирует* точку y, если x есть предок y и $L(x) \supseteq L(y)$.

Точку y будем называть *блокированной*,[9] если она блокирована некоторой точкой x.

Блокированные точки и их потомков будем называть *неактивными* точками, остальные — *активными*.

Теперь правила табло-алгоритма сформулировать легко — нужно к каждому правилу из Таблицы 4.1 добавить дополнительное предусловие «точка x — активная», а также ввести новое правило, добавляющее концепт E в каждую точку. Получающаяся система правил приведена в Таблице 4.3.

Сам же табло-алгоритм остается прежним (см. Таблицу 4.2), за исключением того, что в строчку, где фигурируют правила $\sqcap, \exists, \forall$, необходимо добавить упоминание $\mathcal{T}$-правила, которое, как и те правила, из ABox $\mathcal{A}$ создает один новый ABox $\mathcal{A}'$. Теперь остается доказать, что построенный таким образом табло-алгоритм является разрешающей процедурой для проблемы выполнимости концептов относительно терминологий. Корректность (лемма 4.2) переносится почти автоматически, так как новые правила (за исключением $\mathcal{T}$-правила) совпадают с прежними, лишь применяются реже из-за дополнительного предусловия («x активная»). Несколько сложнее обстоит дело с завершаемостью (лемма 4.1), поскольку старая оценка на размер структуры, создаваемой табло-алгоритмом, теперь неверна — как мы уже говорили, кванторная сложность концептов в метках вершин теперь не уменьшается по мере увеличения глубины вершины. Наконец, основные изменения произойдут в доказательстве полноты алгоритма (лемма 4.3): интуитивно, мы должны показать, что блокируя точки и исключая их из дальнейших построений алгоритма, мы не упустили никаких «скрытых» противоречий в ABox $\mathcal{A}$ и у $\mathcal{A}$ действительно существует модель. Перейдем к строгим формулировкам и доказательствам. Напомним, что выражение «ABox $\mathcal{A}$ выполним относительно TBox $\mathcal{T}$» означает то же самое, что и «база знаний $(\mathcal{T}, \mathcal{A})$ выполнима».

[9] Вообще говоря, условие блокированности является «временным»: в какой-то момент работы алгоритма точка y может быть блокирована точкой x, но при применении очередного правила табло-алгоритма в точку y могут добавиться новые концепты — и она может перестать быть блокированной. Однако, наш алгоритм будет устроен таким образом, что блокированные точки «выпадают» из процесса и в их метки ничего не добавляется; соответственно, однажды став блокированной, точка остается ею до конца.

Правило	Условия применения	Действие
$\sqcap$-правило	если 0. точка x — активная 1. $x\colon (C \sqcap D) \in \mathcal{A}$ 2. $x\colon C \notin \mathcal{A}$ или $x\colon D \notin \mathcal{A}$	то $\mathcal{A}' := \mathcal{A} \cup \{x\colon C,\ x\colon D\}$
$\sqcup$-правило	если 0. точка x — активная 1. $x\colon (C \sqcup D) \in \mathcal{A}$ 2. $x\colon C \notin \mathcal{A}$ и $x\colon D \notin \mathcal{A}$	то $\mathcal{A}' := \mathcal{A} \cup \{x\colon C\}$ $\mathcal{A}'' := \mathcal{A} \cup \{x\colon D\}$
$\exists$-правило	если 0. точка x — активная 1. $x\colon \exists R.C \in \mathcal{A}$ 2. нет такого y, что $xRy \in \mathcal{A}$ и $y\colon C \in \mathcal{A}$	то создать новую точку y и $\mathcal{A}' := \mathcal{A} \cup \{xRy,\ y\colon C\}$
$\forall$-правило	если 0. точка x — активная 1. $x\colon \forall R.C \in \mathcal{A}$ 2. есть такой y, что $xRy \in \mathcal{A}$ и $y\colon C \notin \mathcal{A}$	то $\mathcal{A}' := \mathcal{A} \cup \{y\colon C\}$
$\mathcal{T}$-правило	если 0. точка x — активная 1. $x\colon E \notin \mathcal{A}$	то $\mathcal{A}' := \mathcal{A} \cup \{x\colon E\}$

Таблица 4.3: Правила табло-алгоритма для логики $\mathcal{ALC}$+TBox.

Лемма 4.6 (Завершаемость)**.** *Не существует бесконечной цепочки $\mathcal{A}_0, \mathcal{A}_1, \ldots$, в которой каждый последующий ABox получен из предыдущего по какому-либо правилу табло-алгоритма.*

Доказательство. Дословно повторяет доказательство леммы 4.1 за исключением оценки глубины дерева G, которая здесь будет иной. Напомним, что $\mathsf{Sb}(C)$ обозначает множество подконцептов концепта C; очевидно, что его мощность зависит линейно от длины концепта $|C|$. Метка $L(x)$ любой точки x является подмножеством объединения $\mathsf{Sb}(C_0) \cup \mathsf{Sb}(E)$; мощность последнего обозначим через n. Тогда, число всевозможных меток $L(x)$ не превышает $K := 2^n - 1$ (минус один, поскольку пустых меток не бывает по построению табло-алгоритма). Отсюда вытекает, что в дереве G, которое соответствует ABox $\mathcal{A}$, длина любой цепи не может превышать $K + 1$. Действительно, любая цепь длины $K + 1$ должна содержать две вершины с одинаковыми метками, а значит, одна из этих вершин будет блокированной, и потому цепь далее расти не может. Итак, глубина дерева G ограничена (экспонентой от длины исходного концепта и терминологии). ◀

Лемма 4.7 (Корректность)**.** *Справедливы следующие утверждения:*
(a) *Концепт C_0 выполним относительно $\mathcal{T} \iff$ ABox $\mathcal{A}_0 = \{x_0\colon C_0\}$ выполним относительно $\mathcal{T}$.*
(b) *Пусть $\mathcal{A}'$ получен из $\mathcal{A}$ по одному из правил: $\sqcap, \exists, \forall, \mathcal{T}$. Если $(\mathcal{T}, \mathcal{A})$ выполнима, то $(\mathcal{T}, \mathcal{A}')$ выполнима.*
(c) *Пусть $\mathcal{A}', \mathcal{A}''$ получены из $\mathcal{A}$ по $\sqcup$-правилу. Если $(\mathcal{T}, \mathcal{A})$ выполнима, то $(\mathcal{T}, \mathcal{A}')$ или $(\mathcal{T}, \mathcal{A}'')$ выполнима.*

Доказательство. Аналогично лемме 4.2 для всех правил, кроме нового $\mathcal{T}$-правила — для него имеем: если $(\mathcal{T}, \mathcal{A})$ выполнима, то существует такая интерпретация $\mathcal{I}$, что $\mathcal{I} \models \mathcal{T}, \mathcal{A}$, а так как $\mathcal{T} = \{\top \sqsubseteq E\}$, то $E^{\mathcal{I}} = \Delta$. В частности, $x^{\mathcal{I}} \in E^{\mathcal{I}}$, то есть $\mathcal{I} \models x\colon E$, и значит $\mathcal{I} \models \mathcal{A}'$, тем самым $(\mathcal{T}, \mathcal{A}')$ выполнима. ◀

Лемма 4.8 (Полнота)**.** *Всякий полный непротиворечивый ABox содержит выполнимое относительно $\mathcal{T}$ подмножество, содержащее ABox $\mathcal{A}_0$.*

Доказательство. Пусть $\mathcal{A}$ — полный и непротиворечивый ABox. Нам требуется построить ABox $\widehat{\mathcal{A}}$, удовлетворяющий условию $\mathcal{A}_0 \subseteq \widehat{\mathcal{A}} \subseteq \mathcal{A}$, и являющийся выполнимым относительно $\mathcal{T}$. Возьмем в качестве $\widehat{\mathcal{A}}$ ограничение ABox-а $\mathcal{A}$ на множество активных точек:

$$\widehat{\mathcal{A}} := \{\ x\colon C \in \mathcal{A}\ \mid\ x \text{ активная }\}\ \cup\ \{\ xRy \in \mathcal{A}\ \mid\ x \text{ и } y \text{ активные }\}.$$

Ясно, что $\mathcal{A}_0 \subseteq \widehat{\mathcal{A}}$, так как точка x_0 не имеет предков и потому не может быть блокированной. Остается доказать, что $\widehat{\mathcal{A}}$ выполним относительно $\mathcal{T}$. Построим аналог канонической модели. По существу, модель будет «считана» с ABox $\widehat{\mathcal{A}}$ с добавлением новых стрелок: всякая стрелка, которая в $\mathcal{A}$ вела в какую-либо блокированную точку y, будет «перенаправлена» в точку z, которая блокирует точку y. Формально, модель $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ строится следующим образом:

$$\begin{aligned}
\Delta &:= \{x \mid \text{ индивид } x \text{ встречается в } \widehat{\mathcal{A}}, \text{ т.е. является активной точкой}\},\\
A^{\mathcal{I}} &:= \{x \in \Delta \mid x\colon A \in \mathcal{A}\},\\
R^{\mathcal{I}} &:= \{(x, y) \in \Delta \times \Delta \mid xRy \in \mathcal{A}\} \cup\\
&\quad \cup \{(x, z) \in \Delta \times \Delta \mid \exists \text{ точка } y, \text{ блокированная точкой } z \text{ и такая, что } xRy \in \mathcal{A}\}.
\end{aligned}$$

Требуется доказать, что $\mathcal{I} \models \top \sqsubseteq E$ и $\mathcal{I} \models \widehat{\mathcal{A}}$; последнее означает $\mathcal{I} \models \alpha$ для всех фактов $\alpha \in \widehat{\mathcal{A}}$. Для фактов вида $xRy \in \widehat{\mathcal{A}}$ это очевидно: поскольку $xRy \in \mathcal{A}$ и точки x, y — активные, то $(x, y) \in R^{\mathcal{I}}$ по первой строчке построения $R^{\mathcal{I}}$, а значит $\mathcal{I} \models xRy$. Для фактов же вида $x\colon C$ индукцией по построению концептаЁC докажем требуемую импликацию:

$$\text{для любого } x \in \Delta\colon \qquad \boxed{\text{если } x\colon C \in \mathcal{A}, \text{ то } x \in C^{\mathcal{I}}} \qquad (\star)$$

Напомним, что ввиду нормализованности, концепт C построен из концептов вида $\top, \bot, A, \neg A$ (где A атомарный) с помощью связок $\sqcap, \sqcup, \exists, \forall$.

База индукции — дословно как в лемме 4.8. Шаги $\sqcap$ и $\sqcup$ тривиальны. Остановимся на кванторах.

- Случай $C = \exists R.D$. Пусть $x\colon \exists R.D \in \widehat{\mathcal{A}} \subseteq \mathcal{A}$. Надо доказать, что $x \in (\exists R.D)^{\mathcal{I}}$. Точка x — активная, так как она из $\widehat{\mathcal{A}}$. Таким образом, 0-я и 1-я предпосылки $\exists$-правила выполнены. Поскольку АВох $\mathcal{A}$ полон, то к нему $\exists$-правило не применимо. Значит, не выполнена 2-я предпосылка, то есть существует такая точка y, что $xRy, y\colon D \in \mathcal{A}$. Ввиду того, что непосредственный предок точки y (точка x) является активной, возможны два случая:
  - Точка y — активная, то есть $y \in \Delta$, и значит $xRy, y\colon D \in \widehat{\mathcal{A}}$. Отсюда по предположению индукции $\langle x, y\rangle \in R^{\mathcal{I}}$ и $y \in D^{\mathcal{I}}$, тем самым $x \in (\exists R.D)^{\mathcal{I}}$ по определению семантики квантора $\exists$.
  - Точка y блокирована. Тогда y блокирована некоторой *активной*[10] точкой $z \in \Delta$. Тогда $\langle x, z\rangle \in R^{\mathcal{I}}$ по второй строчке построения $R^{\mathcal{I}}$. Кроме того, блокировка означает, что z есть предок y и $L(y) \subseteq L(z)$. Из $y\colon D \in \mathcal{A}$ следует $D \in L(y) \subseteq L(z)$, тем самым $z\colon D \in \mathcal{A}$. Более того, $z\colon D \in \widehat{\mathcal{A}}$, поскольку z — активная. Отсюда по предположению индукции имеем $z \in D^{\mathcal{I}}$, что вместе $\langle x, z\rangle \in R^{\mathcal{I}}$ влечет $x \in (\exists R.D)^{\mathcal{I}}$ по определению семантики квантора $\exists$.
- Случай $C = \forall R.D$. Пусть $x\colon \forall R.D \in \widehat{\mathcal{A}} \subseteq \mathcal{A}$. Точка x — активная, так как она из $\widehat{\mathcal{A}}$. Таким образом, 0-я и 1-я предпосылки $\forall$-правила выполнены. Поскольку АВох $\mathcal{A}$ полон, то к нему $\forall$-правило не применимо, значит, не выполнена 2-я предпосылка $\forall$-правила (для любого y). Нам требуется доказать, что $x \in (\forall R.D)^{\mathcal{I}}$. Для этого возьмем любой $z \in \Delta$, такой что $(x, z) \in R^{\mathcal{I}}$ и покажем, что $z \in D^{\mathcal{I}}$. Возможны два случая (отвечающие первой и второй строчке построения $R^{\mathcal{I}}$):
  - $xRz \in \mathcal{A}$. Так как 2-я предпосылка $\forall$-правила для z не выполнена, то $z\colon D \in \mathcal{A}$. Более того, $z\colon D \in \widehat{\mathcal{A}}$, поскольку z — активная точка. Отсюда $z \in D^{\mathcal{I}}$ по предположению индукции.
  - существует точка y, блокированная точкой z и такая, что $xRy \in \mathcal{A}$. Это означает, что z есть предок y и $L(y) \subseteq L(z)$. Так как 2-я предпосылка $\forall$-правила для z не выполнена, то $z\colon D \in \mathcal{A}$. Отсюда $D \in L(y) \subseteq L(z)$, тем самым $z\colon D \in \mathcal{A}$. Более того, $z\colon D \in \widehat{\mathcal{A}}$, поскольку z — активная точка. Отсюда по предположению индукции заключаем $z \in D^{\mathcal{I}}$.

Таким образом, $\mathcal{I} \models \widehat{\mathcal{A}}$. Кроме того, $\mathcal{A}$ полон, значит, к нему $\mathcal{T}$-правило не применимо. Поэтому для любой точки $x \in \Delta$ (т.е. активной точки) факт $x\colon E$ лежит в $\mathcal{A}$, откуда по $(\star)$ имеет место $x \in E^{\mathcal{I}}$. Следовательно, $\mathcal{I} \models E$. Тем самым АВох $\widehat{\mathcal{A}}$ выполним относительно $\mathcal{T}$. ◀

Теорема 4.9 (Разрешимость $\mathcal{ALC}$ с терминологиями)**.** *Проблема выполнимости концептов относительно терминологий в логике $\mathcal{ALC}$ разрешима. А именно, описанный табло-алгоритм является разрешающей процедурой для этой проблемы.*

Доказательство. Завершаемость и корректность табло-алгоритма доказываются так же, как в теореме 4.4. Слегка изменится доказательство полноты:

Полнота: Пусть табло-алгоритм на входе $\mathcal{A}_0$ выдал ответ 1. Значит, он построил полный непротиворечивый АВох $\mathcal{A}$. По лемме 4.8 существует АВох $\widehat{\mathcal{A}}$, выполнимый относительно $\mathcal{T}$ и такой, что $\mathcal{A}_0 \subseteq \widehat{\mathcal{A}} \subseteq \mathcal{A}$. Отсюда заключаем, что $(\mathcal{T}, \mathcal{A}_0)$ совместна, и по лемме 4.7(a) концепт C_0 выполним относительно $\mathcal{T}$. ◀

Теорема 4.10. *Проблема совместности баз знаний в логике $\mathcal{ALC}$ разрешима.*

Доказательство. Достаточно применить тот же табло-алгоритм, но не к АВох вида $\{x_0\colon C_0\}$, а к произвольному АВох $\mathcal{A}_0$. Все доказательства модифицируются точно также, как описано в теореме 4.5. ◀

[10] По двум разным причинам, которые могут пригодиться в других логиках, поэтому приведем их обе. Первая: если y блокирована некоторой точкой z_1, которая в свою очередь блокирована точкой z_2, то y блокирована точкой z_2; поскольку имеется лишь конечное число предков точки y, то, продолжая эту цепь, мы в итоге найдем точку z, блокирующую y и являющуюся активной. Вторая: множество предков точки y состоит из ее непосредственного предшественника (точки x — активной) и множества предков точки x, которое содержит лишь из активные точки.

Замечание 4.2. Что же такое «табло»? По сути, это синтаксическая структура, имитирующая некоторую модель и содержащая в том или ином виде исходные данные табло-алгоритма (концепт, который нужно проверить на выполнимость, терминологию, базу знаний и т.д.), тем самым гарантируя их выполнение в данной модели. Как мы видели выше, в случае логики $\mathcal{ALC}$ табло есть просто текущий ABox, который пополняется алгоритмом на каждом шаге. В конце работы алгоритма по табло можно непосредственно считать модель. В общем же случае, в табло-алгоритмах для других ДЛ возможны вариации:

- табло может не иметь вид ABox (например, в табло могут добавляться утверждения вида $x \neq y$);
- табло может не представлять модель буквально (например, роли в табло могут не быть транзитивными, тогда как в модели они должны были бы быть транзитивными, чтобы выполнялась аксиома транзитивности);
- табло может быть бесконечным, и в этом случае табло-алгоритм строит не само табло, а некоторую конечную «сокращенную структуру», из которой табло получается каким-либо способом (например, разверткой путей, ведущих из корня).