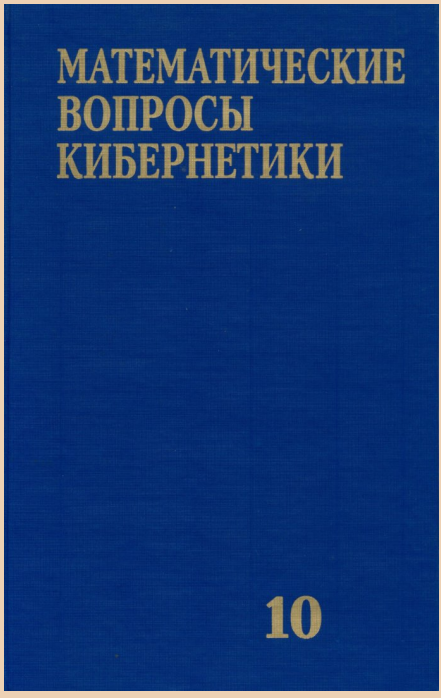
ИПМ им. М.В. Келдыша РАН • Электронная библиотека
Математические вопросы кибернетики • Выпуск 10



С. В. Попов

Логическое моделирование

Рекомендуемая форма библиографической ссылки: Попов С. В. Логическое моделирование // Математические вопросы кибернетики. Вып. 10. — М.: Физматлит, 2001. — С. 167–214. URL: http://library.keldysh.ru/mvk.asp?id=2001-167

ЛОГИЧЕСКОЕ МОДЕЛИРОВАНИЕ *)

С. В. ПОПОВ
(МОСКВА)

Введение

Под логическим моделированием будем понимать конструирование логического исчисления, сигнатура которого отображает понятия исследуемой предметной области, а аксиоматика — зависимости между ними. При этом логические модели исчисления должны удовлетворять закономерностям предметной области. Исчисление будет тем более адекватным, чем больше существенных закономерностей предметной области учитывает модель.

Примеры подобных задач легко найти в теории управления, искусственном интеллекте, в технических областях, в экономике и т. д. Кроме того, накопившаяся в базах данных информация зачастую требует обобщения. Это также задача логического моделирования, решение которой дает возможность получить дополнительные сведения из уже существующих данных. Следующий пример иллюстрирует тип задач и проверяемых критериев, которые возникают при анализе реальных баз данных.

Пример 1. В уставе ЗАО «Профит» сформулированы следующие правила, которым должны удовлетворять члены управляющих органов:

1) члены финансового комитета должны избираться лишь среди членов общей дирекции;

2) нельзя быть одновременно членом общей дирекции и членом ревизионной комиссии, не будучи членом финансового комитета;

3) ни один член ревизионной комиссии не может быть членом финансового комитета.

Пусть информация об управляющих органах Профита представлена в виде трех одноместных отношений: *Финансовый комитет*: Иванов, Светлов, Измайлов, Севостьянов; *Общая дирекция* : Дубовицкий, Смирнов, Иванов, Светлов, Севостьянов, Куприянов; *Ревизионная комиссия* : Измайлов, Сушенков, Храмов, Борисов, Куприянов.

Требуется установить, удовлетворяют ли эти отношения сформулированному критерию. Анализ сформулированного условия показывает, что финансовый комитет ЗАО должен образовывать подмножество общей дирекции, а членом ревизионной комиссии не может быть член общей дирекции. Теперь нетрудно увидеть, что управляющие органы ЗАО выбраны с нарушением устава, так как Измайлов входит в финансовый комитет, не являясь членом общей дирекции, а Куприянов является членом одновременно ревизионной комиссии и общей дирекции.

---

*) Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (проекты 00-01-00199 и 01-01-00930).

В статье предлагается метод построения логических моделей по спецификации предметной области. Демонстрируется, что при определенных условиях логическая формула сама является программой, которая конструирует ее модель. В итоге исходная спецификация, как логическое исчисление, автоматически порождает собственную модель.

Трудность реализации подобных систем состоит в том, что типы данных современных средств программирования мало приспособлены для описания логических моделей. Проблема здесь в том, что обычные средства не разделяют процессы порождения элементов пространства поиска и управления им. Это вызывает существенные трудности из-за необходимости одновременного порождения новых элементов пространства поиска и управления этим процессом. Поэтому естественным представляется принципиально иной подход. Он состоит в разделении на уровне языков программирования механизмов порождения моделей (синтаксическая часть спецификации) и управления порождением (семантическая часть).

В частности, успешно зарекомендовал себя т. н. двухуровневый язык логического моделирования Покос: его нижний уровень основывается на механизме построения логических моделей и предназначен для порождения элементов поискового пространства, а верхний — представляет собой язык управления потоками данных. Разделяя процессы порождения и управления, удается существенно снизить трудозатраты при разработке систем поиска решения. Основная идея двухуровневого программирования заключается в следующем: всякая задача представляется в виде определенной системы понятий, а нахождение решения сводится к построению логической модели, отношения которой определяются введенными понятиями.

Описание реальной содержательной области представляет собой достаточно трудную проблему в силу большого числа понятий и соотношений между ними, которые описывают их вложенность и иные отношения, например временные. Поэтому решение подобных задач требует инструмента, позволяющего достаточно эффективно вводить необходимую систему понятий и легко с ней работать, трансформируя по мере появления новых или переопределения старых. Тем самым уточнение спецификации сводится к уточнению понятийного базиса предметной области.

Этот инструмент должен базироваться, во-первых, на строгом логическом формализме, который позволяет создавать непротиворечивые описания предметной области, и во-вторых, на механизме наполнения понятий, исходя из вида спецификации и требуемого решения. Таким образом, легко усматривается наличие двух механизмов — первого для описания предметной области, и второго собственно для поиска решения.

В Покосе сочетаются две формы программирования: логического — в виде механизма построения логических моделей, и операторного — в виде языка операторного типа, реализующего описание стратегий управления, которые используются при построении моделей. Второй уровень служит метаязыком по отношению к логическому языку, его конструкции содержат прямые ссылки на конструкции первого уровня. Тем самым Покос — это двухуровневый язык — его нижний (логический) уровень служит для описания элементов проблемной области, которые суть элементы искомых моделей, и верхний (операционный) — для задания метода порождения необходимого фрагмента логической модели.

Вычисление программы сводится к нахождению неподвижной точки в классе нормальных моделей. Найденные неподвижные точки представляют собой логические модели, которые согласуются с содержательной спецификацией, так как отношения и функции языка соответствуют неформальной спецификации предметной области, а построение новых элементов модели

напоминает обычные рассуждения. При этом единицами моделей являются не отдельные элементы, а классы эквивалентности.

Программа на Покосе может обладать несколькими неподвижными точками, что, естественно, ставит вопрос об их интерпретации. В языке Покос множественность неподвижных точек рассматривается как множественность решений задачи при различных стратегиях решения. Это согласуется с семантикой, принятой в неклассических логиках и определенной в терминах возможных миров. При описании решений в терминах Покоса также можно говорить о возможных состояниях в пространстве поиска.

Как представляется, Покос, с одной стороны, является шагом в развитии традиционного логического программирования и, с другой, — обобщением т. н. концептуального программирования, когда объекты предметной области и средства их порождения отделены от управления порождением. Основное назначение Покоса состоит в создании инструмента для моделирования систем управления. Как показывает опыт, использование Покоса при разработке логических моделей реальных процессов и систем управления позволяет существенно снизить трудозатраты по сравнению с программированием традиционными средствами.

§ 1. Логическое моделирование. Пропозициональный случай

1.1. Пропозициональные задачи. Декларированная выше цель работы — описание предмета и методов логического моделирования. Поэтому основное внимание будет уделено построению различных интерпретаций логических формул. Известно, что имеется большое число нетрадиционных интерпретаций. Ниже будет определена так называемая *операционная*, которая является теоретической основой логического моделирования. Вначале операционная семантика определяется для пропозициональных формул.

Следующим образом определим понятие задачи: *задачей* называется четверка $Z = (C, F, c_0, C_{\text{fin}})$, где C есть множество состояний; $F = \{f: C \to C\}$ — совокупность отображений (действий), $c_0 \in C$ — исходное состояние и C_{fin} — множество финальных (или заключительных) состояний. Применение действия f к состоянию c обозначим $f(c)$. Решить задачу — значит найти последовательность действий, которая переводит исходное состояние в одно из финальных.

Состояния задач могут представлять собой векторы, множества определенного вида или таблицы, реализующие отношения. Финальные состояния, как правило, задаются фиксированием определенных отношений между элементами множества C: равенство элементов (например, равенство отрезков в задачах геометрии), включение элемента в множество (при решении комбинаторных задач) или непустота интерпретации (как при построении логических моделей).

В множестве C можно выделить некоторое подмножество так называемых *достижимых состояний*, каждое из которых есть результат применения некоторой последовательности действий к исходному состоянию. Совокупность всех достижимых состояний задачи Z представляется в виде развертки R_Z — дерева с единственным корнем, каждый узел которого помечен некоторым состоянием. Корень помечен исходным состоянием, а непосредственные потомки всякого узла развертки образуются в результате применений к этому узлу всех действий, каждое ребро помечено соответствующим действием. Развертка задает отношение частичного порядка $<$ на множестве состояний: если состояние d есть потомок состояния c, то полагаем, что выполняется отношение $c < d$ (c *выше* d).

Траекторией для задачи Z называется любой простой путь развертки, при этом верхний узел пути называется *началом* траектории, а нижний —

концом. Траекторию, содержащую последовательность $f_1, f_2, \ldots, f_r$ действий, обозначим $f_1 \to f_2 \to \ldots \to f_r$.

Состояние c задачи Z называется ее *неподвижной точкой*, если для любого действия f из F справедливо равенство $f(c) = c$. Полагаем, что неподвижная точка есть висячий узел развертки. В общем случае задача может обладать несколькими неподвижными точками. Легко заметить, что если c и d суть разные неподвижные точки, то они не сравнимы относительно порядка $<$.

В этом разделе будут представлены так называемые пропозициональные задачи. Каждая такая задача характеризуется n пропозициональными переменными $x_1, x_2, \ldots, x_n$, участвующими в описании ее действий. Ее пространством состояний служит единичный n-мерный куб E^n.

Если $c = (\sigma_1 \sigma_2 \ldots \sigma_n)$ есть состояние задачи и $\sigma_i = 0$ или $\sigma_i = 1$, то полагаем, что в этом состоянии значение переменной x_i определено и равно соответственно 0 (*ложь*) или 1 (*истина*), $i = 1, 2, \ldots, n$. Поэтому для пропозициональной формулы $L(x_{i_1}, x_{i_2}, \ldots, x_{i_m})$, зависящей от переменных $x_{i_1}, x_{i_2}, \ldots, x_{i_m}$, где $\{x_{i_1}, x_{i_2}, \ldots, x_{i_m}\} \subseteq \{x_1, x_2, \ldots, x_n\}$, можно говорить о ее логическом значении в состоянии c, которое вычисляется обычным способом для соответствующих значений ее переменных.

Всякая n-мерная пропозициональная задача содержит совокупность $F = \{f_i\colon L_i \Rightarrow \beta_1^i \,\&\, \beta_2^i \,\&\, \ldots \,\&\, \beta_{k_i}^i\}$ действий, где L_i — формула в базисе $(\&, \neg)$, зависящая от переменных $x_{j_1}, x_{j_2}, \ldots, x_{j_{m_i}}$, принадлежащих множеству $\{x_1, x_2, \ldots, x_n\}$, $\Rightarrow$ — новый символ и β_h^i, $h = 1, 2, \ldots, k_i$, суть литеры, среди которых отсутствуют контрарные пары, $i = 1, 2, \ldots, q$.

Применение действия f_i к состоянию $c \in E^n$ осуществляется следующим образом. Пусть $c = (\sigma_1 \sigma_2 \ldots \sigma_n)$ и формула L_i истинна в c. Если конъюнкт $\beta_1^i \,\&\, \beta_2^i \,\&\, \ldots \,\&\, \beta_{k_i}^i$ также истинен в c, то результат применения f_i к c совпадает с c (обозначается $f_i(c) = c$). Если конъюнкт $\beta_1^i \,\&\, \beta_2^i \,\&\, \ldots \,\&\, \beta_{k_i}^i$ ложен в c, то результат применения f_i к c есть вектор $d = (s_1 s_2 \ldots s_n)$, где $s_j = \sigma_j$, если в $\beta_1^i \,\&\, \beta_2^i \,\&\, \ldots \,\&\, \beta_{k_i}^i$ отсутствует литера $x_j^{1-\sigma_j}$, в противном случае $s_j = 1 - \sigma_j$, $j = 1, 2, \ldots, n$ (обозначается $f_i(c) = d$). Если формула L_i ложна в c, то $f_i(c) = c$.

Пример 2. Пусть состояние c есть вектор 0110 и действие f имеет вид $\bar{x}_1 \,\&\, x_3 \Rightarrow x_4$. Тогда $f(c) = d$ и состояние d есть вектор 0111. С другой стороны, если f выглядит как $\bar{x}_2 \& x_3 \Rightarrow x_4$, то $f(c) = c$. Пусть состояние c есть вектор 000 и действие f имеет вид $x \,\&\, \bar{y} \Rightarrow z$. Тогда $f(c) = d$ и состояние d есть вектор 001. Легко увидеть, что состояние 001 является неподвижной точкой для f.

Таким образом, каждое действие задачи представляет собой отображение $2^{E^n} \to 2^{E^n}$, т. е. обладает операционной семантикой. Если же трактовать символ $\Rightarrow$ как импликацию, то действия являются логическими формулами в базисе $(\&, \neg, \Rightarrow)$, зависящими от n переменных, и поэтому обладают логической семантикой. Поэтому, говоря в дальнейшем о логических значениях действий, будем полагать, что они рассматриваются как логические формулы в этом базисе. Рассмотрим как связаны эти два типа семантик.

Достаточно просто доказываются следующие утверждения.

Теорема 1. *Пусть c и d состояния задачи, f — ее действие, $f(c) = d$ и вектор c принадлежит единичному покрытию N_f формулы f. Тогда векторы c и d совпадают.*

В обратную сторону утверждение также справедливо.

Теорема 2. *Пусть для состояния c и действия f имеет место равенство $f(c) = c$. Тогда вектор c принадлежит единичному покрытию формулы f.*

Доказательство. Если $f(c)=c$, то возможны лишь два случая: левая часть формулы f может быть либо ложна либо истинна. В первом случае вся импликация истинна в силу ложности ее посылки. Во втором случае при означивании переменных, определяемом вектором c, заключение формулы f также истинно. Следовательно, формула f истинна в c.

Теорема доказана.

Теорема 3. *Если для действия f и состояния c верно, что $f(c)=d$, и векторы c и d различны, то f ложна в c и истинна в d.*

Доказательство. Из того, что $f(c)=d$, и векторы c и d различны, следует, что в состоянии c посылка формулы f истинна, а заключение — ложно. Поэтому вся импликация ложна. С другой стороны, для вектора d заключение импликации f истинно. Следовательно, и вся импликация истинна.

Теорема доказана.

Теперь понятно, что если задача обладает неподвижной точкой, то при соответствующем ей означивании логические значения всех действий суть *истина*. С другой стороны, любое состояние, которое не является неподвижной точкой, опровергает хотя бы одно действие.

Пример 3. Легко показать, что не всякая пропозициональная задача имеет неподвижную точку. Действительно, пусть действия задачи суть следующие: f_1: $x\,\&\,\bar{y}\,\&\,\bar{z}\Rightarrow\bar{x}\,\&\,\bar{y}$; f_2: $\bar{x}\,\&\,y\,\&\,\bar{z}\Rightarrow\bar{y}\,\&\,z$; f_3: $\bar{x}\,\&\,\bar{y}\,\&\,z\Rightarrow x\,\&\,\bar{z}$ и начальное состояние есть вектор 100. Развертка этой задачи представляет собой бесконечную ветвь.

Отметим, что если развертка задачи содержит бесконечную ветвь, ребра которой помечены действиями $f_1, f_2, \ldots, f_s$, то в каждом состоянии, которыми помечены ее узлы, формула $f_1\&f_2\&\ldots\&f_s$ опровержима.

Будем говорить, что задача Z обладает *единственной неподвижной точкой*, если в каждой ее развертке, независимо от начального состояния, все неподвижные точки представляют собой совпадающие векторы. Легко увидеть, что совокупность F порождает однозначное отображение, если образом начального состояния считать неподвижную точку.

Будем говорить, что задача является *монотонной*, если в каждой ее развертке, независимо от начального состояния, в любой траектории один компонент вектора может измениться лишь один раз.

Пример 4. Следующее множество действий определяет монотонную задачу и генерирует единственную неподвижную точку: $x\,\&\,y\Rightarrow u$; $x\,\&\,y\,\&\,u\Rightarrow v$; $x\,\&\,y\,\&\,u\,\&\,v\Rightarrow z$.

Легко увидеть, что если действия обладают положительными конъюнктами слева и справа, то они определяют монотонную задачу, обладающую единственной неподвижной точкой. С другой стороны, если все действия обладают положительными конъюнктами лишь в правой части, то они определяют монотонную задачу, но, в общем случае, могут генерировать несколько неподвижных точек.

Изложенная методика нахождения неподвижных точек пропозициональных задач напоминает работу устройства управления, отрабатывающего регулирующее воздействие всякий раз, когда нарушается один из параметров, по которому происходит управление. В данном случае управляющими критериями выступают действия задач, и если в некотором состоянии действие f интерпретируется как ложная формула, то система (здесь: задача) переводится в новое состояние, в котором f принимает значение *истина*. Система занимает устойчивое состояние (здесь: задача достигает неподвижной точки), когда отклонения по всем ее критериям (здесь: действиям) нулевые (здесь: все действия интерпретируются как принимающие значение *истина*).

Исследуемым свойством в терминах операционной семантики будет выполнимость формул. Покажем, что каждая пропозициональная формула G

от n переменных определяет пропозициональную задачу Z_G также от n переменных, неподвижные точки которой представляют собой модели формулы G.

Пусть G есть к.н.ф. и $\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_m$ — это ее дизъюнкт. Представим его в виде совокупности m действий:

$$\bar{\alpha}_1 \,\&\, \bar{\alpha}_2 \,\&\, \ldots \bar{\alpha}_{i-1} \,\&\, \bar{\alpha}_{i+1} \,\&\, \ldots \,\&\, \bar{\alpha}_m \Rightarrow \alpha_i,\ i = 1, 2, \ldots, m.$$

Проделав такие преобразования со всеми дизъюнктами формулы G, мы построим совокупность F действий. Теперь, если принять в качестве множества состояний задачи Z_G совокупность всех вершин n-мерного единичного куба и выделить начальное состояние c_0, то задача Z_G определена полностью.

Таким образом, формула G, с одной стороны, описывает логические закономерности и обладает соответствующей логической семантикой, а с другой естественно определяет задачу и поэтому обладает операционной семантикой.

Из ранее доказанного получаем, что справедливо следующее утверждение.

Теорема 4. *Неподвижные точки задачи Z_G определяют означивания переменных формулы G, при которых она истинна.*

Пример 5. Пусть $G = (x \vee \bar{y} \vee z) \& (x \vee \bar{y} \vee \bar{z}) \& (\bar{x} \vee \bar{y} \vee z) \& (x \vee y \vee \bar{z})$. Она определяет следующие действия:

$$f_{11}\colon \bar{x} \,\&\, y \Rightarrow z;\ f_{21}\colon \bar{x} \,\&\, y \Rightarrow \bar{z};\ f_{31}\colon x \,\&\, y \Rightarrow z;\ f_{41}\colon \bar{x} \,\&\, \bar{y} \Rightarrow \bar{z};$$

$$f_{12}\colon \bar{x} \,\&\, \bar{z} \Rightarrow \bar{y};\ f_{22}\colon \bar{x} \,\&\, z \Rightarrow \bar{y};\ f_{32}\colon x \,\&\, \bar{z} \Rightarrow \bar{y};\ f_{42}\colon \bar{x} \,\&\, z \Rightarrow y;$$

$$f_{13}\colon y \,\&\, \bar{z} \Rightarrow x;\ f_{23}\colon y \,\&\, z \Rightarrow x;\ f_{43}\colon \bar{y} \,\&\, z \Rightarrow x;\ f_{41}\colon \bar{x} \,\&\, \bar{y} \Rightarrow \bar{z}.$$

Формула G принимает истинное значение на наборах 000, 100, 111, 101, которые суть неподвижные точки задачи Z_G. Легко увидеть, что выбрав произвольное начальное состояние из множества 010, 011, 110, 001, мы получим развертку с этими неподвижными точками.

На этом примере легко убедиться в немонотонности переходов между состояниями. Действительно, логические значения, которые приписываются переменным, не фиксируются для всех состояний развертки, они меняются при переходе от состояния к состоянию, в одних состояниях это *истина*, в других *ложь* и эти значения обуславливаются лишь используемыми действиями задачи.

Для дальнейшего изложения потребуется такое соглашение: будем рассматривать к.н.ф. G от n переменных, $G = D_1 \,\&\, D_2 \,\&\, \ldots \,\&\, D_m$, и полагать, что каждый дизъюнкт D_i определяет совокупность действий задачи Z_G, которую обозначим также D_i, $i = 1, 2, \ldots, m$.

Теорема 5. *Если действие f из D_i применимо к состоянию c и $f(c) = c$, то все остальные действия из D_i также не изменяют этого состояния.*

С другой стороны, имеет место следующее утверждение.

Теорема 6. *Если действие f из D_i применимо к состоянию c, $f(c) = d$ и векторы c и d различны, то результаты применения всех действий из D_i к c попарно различны.*

Доказательство. Пусть $f = \Delta_0 \Rightarrow l_1$, где l_1 — литера. По условию, конъюнкция Δ_0 истинна в c, а так как $c \neq d$, то литера l_1 в c ложна. Всякое действие $g \in D_i$, отличное от f, имеет вид $\bar{l}_1 \,\&\, \Delta_1 \Rightarrow l_2$, где $\Delta_0 = \bar{l}_2 \,\&\, \Delta_1$ и, следовательно, литера l_2 истинна в c, а l_2 — ложна. Но так как l_1 ложна в c, то $\bar{l}_1$ — истинна. Поэтому конъюнкция $\bar{l}_1 \,\&\, \Delta_1$ истинна в c, а литера

l_2 — ложна. Следовательно, действие g применимо к состоянию c, $g(c) = e$ и $e \neq d$.

Теорема доказана.

1.2. Геометрическая интерпретация. Приведем графическую интерпретацию конструкций из предыдущего раздела.

Пусть к.н.ф. A от n переменных имеет вид $D_1 \& D_2 \& \ldots \& D_m$ и действие $f = \Gamma \Rightarrow x^\sigma$, где $\sigma \in \{0, 1\}$, принадлежит задаче Z_A. Переменная x не встречается в левой части этого действия, следовательно, конъюнкт Γ состоит из $q \geqslant n - 1$ литер и поэтому он определяет интервал N_Γ размерности $n - q$. Последний разбивается на две непересекающиеся части N^0 и N^1, отличающиеся лишь значениями компонента, соответствующего переменной x: в N^0 этот компонент равен нулю, в N^1 — единице. Для каждой точки $\sigma_0 \in N^0$ существует единственная точка $\sigma_1 \in N^1$, которая отличается от σ_0 лишь компонентом, соответствующим переменной x. Назовем такие пары точек *соответствующими*. Так как всякая пара соответствующих точек отличается лишь одним компонентом, то в кубе E^n они соединены ребром.

Действие f всякую точку из $N^{1-\sigma}$ преобразует в соответствующую точку из N^σ, т. е. $f: N^{1-\sigma} \to N^\sigma$. Обозначим области $N^{1-\sigma}$ и N^σ соответственно $\mathrm{Def}(f)$ и $\mathrm{Im}(f)$. Графически представим это отображение в виде дуг, соединяющих соответствующие точки и начинающихся в $\mathrm{Def}(f)$. Тогда *графом, определяемым задачей* Z_A, называется ориентированный граф G_A, узлы которого суть все вершины n-мерного единичного куба, а дуги определяются действиями задачи.

Пример 6. Пусть формула $A(x, y, z)$ выглядит следующим образом: $x \,\&\, \bar{y} \,\&\, (x \vee z)$. Тогда действия задачи Z_A суть $f_1 = x$; $f_2 = \bar{y}$; $f_3 = \bar{x} \Rightarrow z$; $f_4 = \bar{z} \Rightarrow x$. Нетрудно увидеть, что $\mathrm{Def}(f_1) = \{000, 001, 010, 011\}$, $\mathrm{Im}(f_1) = \{100, 101, 110, 111\}$, $\mathrm{Def}(f_2) = \{010, 110, 111, 011\}$, $\mathrm{Im}(f_2) = \{000, 001, 101, 100\}$, $\mathrm{Def}(f_3) = \{000, 010\}$, $\mathrm{Im}(f_3) = \{001, 011\}$, $\mathrm{Def}(f_4) = \{000, 010\}$, $\mathrm{Im}(f_4) = \{100, 110\}$ и дуги, определяемые действиями задачи Z_A, покрывают все узлы куба.

Из геометрической интерпретации задачи легко вытекает утверждение.

Теорема 7. *Всякая точка единичного покрытия формулы A есть либо тупик в графе G_A, либо его изолированная вершина.*

Из определения неподвижной точки следует, что тупики графа G_A и его изолированные точки суть неподвижные точки задачи Z_A.

Пусть конъюнкт $K = x_1^{s_1} \,\&\, x_2^{s_2} \,\&\, \ldots \,\&\, x_q^{s_q}$ определяет интервал N_K. Назовем все точки единичного куба, отличающиеся от N_K в точности в одном компоненте, соответствующем одной из переменных $x_1, x_2, \ldots, x_q$, *гиперсферой* S_K, определяемой конъюнктом K, а интервал N_K — *центром гиперсферы* S_K. Если $q = n$, то гиперсфера S_K превращается в единичную сферу с центром в точке $(s_1 s_2 \ldots s_n)$.

Пример 7. В трехмерном кубе конъюнкт $K = \bar{x} \,\&\, y$ определяет интервал $N_K = \{010, 011\}$. Определяемая им гиперсфера есть множество $\{000, 110, 001, 111\}$.

Имеет место следующее утверждение.

Теорема 8. *Пусть к.н.ф. A от n переменных имеет вид $D_1 \& \ldots \& D_m$. Тогда для любого дизъюнкта $D_i = x_1^{s_1} \vee x_2^{s_2} \vee \ldots \vee x_q^{s_q}$, $i = 1, 2, \ldots, m$, и всякого действия f из D_i верно, что $\mathrm{Def}(f) = N_{K_i}$, где K_i есть двойственный к D_i конъюнкт.*

Доказательство. Дизъюнкт D_i эквивалентно представим в виде импликации $K_i \supset \mathit{false}$, где *false* — тождественно ложная функция. Тогда каждое действие f_r из D_i, $r = 1, 2, \ldots, q$, получается из этой импликации

переносом из левой части в правую в точности одной литеры $x_r^{1-s_r}$ с одновременным инвертированием ее знака. Легко увидеть, что это действие результативно применимо только к точкам интервала N_{K_i}, инвертируя в них компонент $1-s_r$, соответствующий переменной x_r.

Теорема доказана.

Следствие 1. *Область значений* $\mathrm{Im}(f)$ *всякого действия* f *из* D_i, $i=1,2,\ldots,m$, *представляет собой фрагмент гиперсферы с центром* N_K. *Области значений всех действий, порожденных дизъюнктом* D_i, *покрывают всю гиперсферу* S_K.

Таким образом, всякое действие f из D_i представимо в виде совокупности дуг, начинающихся в центре N_{K_i} и оканчивающихся в точках S_{K_i} гиперсферы. В этом случае будем говорить, что гиперсфера S_{K_i} и ее центр N_{K_i} определяются дизъюнктом D_i.

Следствие 2. *Никакая неподвижная точка задачи* Z_A *не принадлежит центру ни одной гиперсферы, порожденной дизъюнктами к.н.ф.* A.

Иными словами всякая неподвижная точка либо лежит на одной из гиперсфер, и не принадлежит центру никакой другой гиперсферы, либо лежит вне всех гиперсфер, т. е. расположена на расстоянии большем двух от центра всякой гиперсферы.

Из предыдущих построений и рассуждений вытекает справедливость следующей теоремы.

Теорема 9. *Пусть* N_A *есть единичное покрытие формулы* A. *Тогда вектор* σ *принадлежит* N_A *тогда и только тогда, когда в графе* G_A *вершина* σ *либо тупиковая, либо изолированная.*

Следствие. *Операционная семантика полна в том смысле, что любой вектор единичного покрытия формулы* A *есть неподвижная точка задачи* Z_A, *и наоборот, всякая неподвижная точка* Z_A *принадлежит единичному покрытию формулы* A.

Чтобы получить удобную процедуру нахождения неподвижных точек, расширим язык описания пространства состояний задач, вводя в рассмотрение множество $\{0,1,\_\}^n$ тернарных векторов. Нулевые и единичные компоненты этих векторов будем называть *значимыми*, а компоненты $\_$ — *незначимыми*. Каждый такой вектор σ определяет в единичном n-мерном кубе интервал, содержащий совокупность всех бинарных векторов, получающихся из σ всевозможными подстановками нулей и единиц вместо всех его незначимых компонентов. Теперь можно полагать, что всякая гиперсфера определяется единственным вектором из $\{0,1,\_\}^n$.

Продолжение примера 1. Пусть требуется установить в какие руководящие органы ЗАО "Профит" может входит один человек. Для этого введем следующие обозначения: обозначим P членство в финансовом комитете, Q — членство в общей дирекции и R — в ревизионной комиссии. Тогда все три условия можно представить в виде формулы $A=(P\supset Q)\,\&\,(\neg(Q\,\&\,R)\vee P)\,\&\,\neg(R\,\&\,P)$, к.н.ф. которой эквивалентна формуле: $(Q\supset\bar{R})\,\&\,(P\supset Q)$. Пространство состояний задачи Z_A описывается трехмерным единичным кубом с упорядочением переменных P, Q, R. Формула A определяет две гиперсферы $S_1=\{\_10,\_01\}$ и $S_2=\{00\_,11\_\}$ с центрами соответственно $N_1=\{\_11\}$ и $N_2=\{10\_\}$. Имеем следующие соотношения:

$$N_1\cap S_2=\{\_11\}\cap\{00\_,11\_\}=\{111\};$$
$$N_2\cap S_1=\{10\_\}\cap\{\_10,\_01\}=\{101\}.$$

Теперь точки гиперсфер S_1 и S_2, не принадлежащие ни одному центру N_1 или N_2, образуют множество

$$S_1 \cup S_2 - \{111, 101\} = \{\_10, \_01, 00\_, 11\_\} - \{111, 101\} = \{\_10, 00\_\}.$$

Как видно, задача Z_A обладает совокупностью неподвижных точек, определяемых векторами $\{\_10, 00\_\}$, которые образуют единичное покрытие исходной формулы. В этом нетрудно убедиться простой проверкой. Решения исходной логической задачи, если отбросить тривиальное, когда управляющие органы ЗАО пусты, суть следующие: один человек может быть членом общей дирекции, но не входить в ревизионную комиссию, при этом он может как входить в финансовый комитет, так и не входить, или являться только членом ревизионной комиссии, не участвуя в других органах управления ЗАО.

Приведем процедуру построения всех неподвижных точек задачи Z_A, определяемой к.н.ф. A, зависящей от n переменных, $A = D_1 \& D_2 \& \ldots \& D_m$. С этой целью введем следующие определения.

Два вектора называются *ортогональными*, если в них на соответствующих местах стоят дополнительные значения 0 и 1.

Определим операцию *вычитания* (-) *векторов* σ_1 и σ_2 над множеством $\{0, 1, \_\}$.

1. Если σ_1 и σ_2 ортогональны, то $\sigma_1 - \sigma_2 = \sigma_1$.

2. Если каждый значимый компонент σ_2 совпадает с соответствующим компонентом σ_1, то $\sigma_1 - \sigma_2 = \varnothing$.

3. В противном случае для простоты представим вектор σ_1 в виде $(s_1^1 s_2^1 \ldots s_q^1 \_ \ldots \_)$, переставив вперед все его значимые компоненты, и пусть при такой же перестановке $\sigma_2 = (s_1^2 s_2^2 \ldots s_n^2)$. Тогда $\sigma_1 \cdot \sigma_2 = \{(s_1^1 s_2^1 \ldots s_q^1 1 - s_{q+1}^2 \ldots \_), \ldots, (s_1^1 s_2^1 \ldots s_q^1 \_ \ldots 1 - s_n^2)\}$.

Операция вычитания обобщается на случай, когда первый аргумент есть множество векторов, следующим образом: $\{\sigma_1, \sigma_2, \ldots, \sigma_m\} \cdot \sigma = \sigma_1 - \sigma \cup \ldots \cup \sigma_m - \sigma$.

Следующая процедура позволяет легко находить все неподвижные точки задачи Z_A.

Процедура нахождения всех неподвижных точек

Пусть дизъюнкты D_i, $i = 1, 2, \ldots, m$ определяют гиперсферы S_i с центрами N_i. Полагаем, что каждый центр N_i определяется вектором $\sigma_i \in \{0, 1, \_\}^n$. Тогда все неподвижные точки задачи Z_A и только они принадлежат разности $\Lambda - \sigma_1 - \sigma_2 - \ldots - \sigma_m$.

Очевидно следующее утверждение.

Теорема 10. *Если формула A выполнима, то разность* $\Lambda - \sigma_1 - \sigma_2 - \ldots - \sigma_m$ *совпадает с ее единичным покрытием.*

Пример 8. Пусть к.н.ф. A выглядит следующим образом: $(x \vee y \vee z) \& (\bar{x} \vee y \vee \bar{z}) \& (\bar{x} \vee \bar{y} \vee \bar{z}) \& (\bar{x} \vee \bar{y} \vee z)$. Легко увидеть, что центры всех гиперсфер, выглядят следующим образом: 000, 101, 111, 110. Вычитая их последовательно из Λ получаем такие множества: $\Lambda - 000 = \{001, 010, 100\}$; $\{001, 010, 100\} - 101 = \{001, 010, 100\}$; $\{001, 010, 100\} - 111 = \{001, 010, 100\}$; $\{001, 010, 100\} - 110 = \{001, 010, 100\}$. Проверкой убеждаемся, что все наборы 001, 010, 100 суть модели для формулы A.

Пример 9. Рассмотрим формулу: $x(x \vee u)(\bar{x} \vee y)(y \vee \bar{z})(y \vee u)(\bar{x} \vee z \vee \bar{u})(\bar{x} \vee \bar{z} \vee \bar{u})(\bar{x} \vee \bar{y} \vee z)(\bar{x} \vee z \vee u)(\bar{y} \vee u)$. Полагаем, что переменные упорядочены следующим образом: x, y, z, u. Для этой формулы центры гиперсфер определяются векторами: $\sigma_1 = 0\_\_$, $\sigma_2 = 0\_0$, $\sigma_3 = 10\_\_$, $\sigma_4 = \_01\_$, $\sigma_5 = \_0\_0$, $\sigma_6 = 1\_01$, $\sigma_7 = 1\_11$, $\sigma_8 = 110\_$, $\sigma_9 = 1\_00$, $\sigma_{10} = \_1\_0$.

Последовательно вычитая эти векторы из Λ, получаем следующие множества: $\Lambda - 0\_\_ = \{1\_\_\}$; $\{1\_\_\} - 0\_0 = \{1\_\_\}$; $\{1\_\_\} - 10\_ = \{11\_\}$;

$\{11\_\} - \_01\_ = \{11\_\}$; $\{11\_\} - \_0\_0 = \{11\_\}$; $\{11\_\} - 1\_01 = \{111\_, 11\_0\}$; $\{111\_, 11\_0\} - 1\_11 = \{1110, 11\_0\}$; $\{1110, 11\_0\} - 110\_ = \{1110\}$; $\{1110\} - 1\_00 = \{1110\}$; $\{1110\} - \_1\_0 = \varnothing$. Таким образом, исходная формула тождественно ложна.

Пример 10. Пусть требуется найти решение следующей системы линейных булевских уравнений: $x+y=0$; $x+z=1$; $x+y+u=0$; $y+z+u=1$. Нетрудно увидеть, что каждому из этих уравнений соответствует совокупность гиперсфер, центры которых характеризуются либо четным либо нечетным распределением единиц в зависимости от значения в правой части уравнения, и содержат все компоненты, соответствующие переменным уравнения. Поэтому все точки гиперсфер, порождаемых одним уравнением, расположены попарно друг от друга на расстоянии два. Следовательно, центры и гиперсферы, порождаемые одним уравнением, покрывают весь единичный куб, определяемый переменными этого уравнения. Легко увидеть, что все гиперсферы, относящиеся к одному уравнению, лишь соприкасаются, так как их центры расположены на расстоянии два. Используя методику перехода от центра к гиперсфере и начиная с точки 00__, которая лежит на гиперсферах, определяемых первым уравнением, получаем следующую последовательность точек, принадлежащих различным гиперсферам: 00__, 001_, 0010. Последняя точка этой последовательности принадлежит всем гиперсферам, данной системы. Если поиск начинается с точки 11__, то последовательность выглядит так: 11__, 110_, 1100. Она также завершается точкой, лежащей на всех гиперсферах данной системы. С другой стороны, последовательность, начинающаяся с точки 00__, может выглядеть таким образом: 00__, 0001 (здесь применяется переход от центра _000 к точке _001, определяемый последним уравнением). Но точка 0001 принадлежит центру гиперсферы, определяемой третьим уравнением, и поэтому решением не является.

Таким образом, если представить поиск решения в виде развертки, определяемой системой уравнений, то максимальная длина траектории, которая заканчивается неподвижной точкой, или точкой, уже встречающейся в траектории, не превосходит числа уравнений системы.

1.3. Свойства неподвижных точек. Исследуем свойства задач, которые вытекают из их графической интерпретации. Легко усмотреть, что задачей Z_A единичный куб разбивается на три непересекающиеся части:

множество K_A, представляющее собой объединение центров гиперсфер, определяемых дизъюнктами к.н.ф. A;

множество S_A, которое есть совокупность точек, принадлежащих гиперсферам, но не принадлежащих центрам гиперсфер;

множество N_A изолированных вершин.

Установим некоторые свойства всех этих множеств. Для этого нам потребуется следующее определение.

Будем говорить, что переменная x_i формулы $A(x_1, x_2, \ldots, x_n)$ *несущественна на бинарном наборе* $(\sigma_1 \ldots \sigma_i \ldots \sigma_n)$, $i \in \{1, 2, \ldots, n\}$, если $A(\sigma_1 \ldots 0 \ldots \sigma_n) = A(\sigma_1 \ldots 1 \ldots \sigma_n)$.

Из определения гиперсферы видно, что *во всякой точке из множества S_A по меньшей мере одна переменная формулы A существенна.* Действительно, изменение некоторого одного компонента неподвижной точки на двойственный переводит вектор в центр гиперсферы, в котором формула ложна.

Для элементов множества N_A имеет место следующее утверждение.

Теорема 11. *Точка единичного куба принадлежит множеству N_A тогда и только тогда, когда на этом наборе всякая переменная формулы A несущественна, а сама формула истинна.*

Следствие. *Если в точке единичного куба каждая переменная формулы A несущественна, а сама формула истинна, то эта точка принадлежит множеству N_A.*

Пусть к.н.ф. A имеет вид $D_1 \& D_2 \& \ldots \& D_i \& D_{i+1} \& \ldots \& D_m$, $Var(D_1 \& D_2 \& \ldots \& D_i) = X_0 \cup X_1$, $Var(D_{i+1} \& \ldots \& D_m) = X_0 \cup X_2$, $X_1 \cap X_2 = \emptyset$. Допустим, что формулы $B = D_1 \& D_2 \& \ldots \& D_i$ и $C = D_{i+1} \& \ldots \& D_m$ определяют задачи соответственно Z_B и Z_C. Исследуем, как относится множество неподвижных точек задачи Z_A и задач Z_B и Z_C. Для этого нам потребуется такое определение.

Пусть $\sigma_1 = \sigma_1\sigma_2 \ldots \sigma_n$, $\sigma_2 = \sigma_1'\sigma_2' \ldots \sigma_n'$, суть векторы из $\{0, 1, \_\}^n$. Тогда *произведение* $\sigma_1 \times \sigma_2$ этих векторов вычисляется следующим образом. Если векторы σ_1 и σ_2 ортогональны, то их произведение пусто. В противном случае $\sigma_1 \times \sigma_2$ есть вектор $\sigma_1''\sigma_2'' \ldots \sigma_n''$, где компоненты σ_i'', $i = 1, 2, \ldots, n$, вычисляются следующим образом: если $\sigma_i = \_$, то $\sigma_i'' = \sigma_i'$; если $\sigma_i' = \_$, то $\sigma_i'' = \sigma_i$; если $\sigma_i = \sigma_i'$, то $\sigma_i'' = \sigma_i$.

Пример 11. Пусть $\sigma_1 = 10\_1\_$, $\sigma_2 = \_01\_0$. Тогда $\sigma_1 \times \sigma_2 = 10110$.

Это определение расширяется естественным образом, когда второй аргумент представляет собой множество $M = \{\sigma_1, \sigma_2, \ldots, \sigma_q\}$ векторов: $\sigma \times M = \{\sigma \times \sigma_1, \sigma \times \sigma_2, \ldots, \sigma \times \sigma_q\}$.

Справедливы следующие утверждения.

Лемма 1. *Пусть $\sigma_1 = \sigma_{11}\sigma_{12}\Lambda_m$ и $\sigma_2 = \Lambda_h\sigma_{21}\sigma_{22}$ суть не ортогональные векторы, $\sigma_{11} \in \{0, 1\}^h$, $\sigma_{22} \in \{0, 1\}^m$. Тогда $\sigma_1 \times \sigma_2 = \sigma_{11}(\sigma_{12} \times \sigma_{21})\sigma_{22}$.*

Лемма 2. *Пусть σ_1 и σ_2 — произвольные векторы. Тогда $\sigma_1 - \sigma_2 = \sigma_1 \times (\Lambda - \sigma_2)$.*

Приступим теперь к описанию неподвижных точек задач Z_B и Z_C. Пусть задача Z_B обладает неподвижной точкой σ, и центры гиперсфер, определяемые задачей Z_C, образуют множество $\{\delta_1, \ldots, \delta_m\}$. Всякий вектор, принадлежащий множеству $\sigma - \{\delta_1, \ldots, \delta_m\}$, будет неподвижной точкой для задачи Z_A. По доказанному $\sigma - \{\delta_1, \ldots, \delta_m\} = \sigma \times (\Lambda - \{\delta_1, \ldots, \delta_m\})$. Разность $\Lambda - \{\delta_1, \ldots, \delta_m\}$ представляет собой множество всех неподвижных точек задачи Z_C. Таким образом, вычисление неподвижных точек задачи Z_A свелось к вычислению произведений каждой неподвижной точки задачи Z_B на каждую неподвижную точку задачи Z_C.

Исследуем теперь взаимное расположение множеств S_B, S_C и S_A. Можно показать, что верно такое утверждение.

Теорема 12. *Пусть $\sigma_0, \sigma_1, \sigma_2$ суть бинарные векторы, означивающие переменные множеств соответственно X_0, X_1 и X_2. Тогда, если $\sigma_1 \times \sigma_0 \in S_B$, а $\sigma_2 \times \sigma_0 \in S_C$, то $\sigma_1 \times \sigma_0 \times \sigma_2$ принадлежит S_A.*

Доказательство. В точке $\sigma_1 \times \sigma_0 \times \sigma_2$ формула A истинна потому, что обе формулы B и C истинны. По условию, $\sigma_1 \times \sigma_0 \in S_B$, а $\sigma_2 \times \sigma_0 \in S_C$. Поэтому в точке $\sigma_1 \times \sigma_0 \times \sigma_2$ каждая переменная формулы A — существенна. Следовательно, точка $\sigma_1 \times \sigma_0 \times \sigma_2$ принадлежит S_A.

Теорема доказана.

Следствие. *Множество S_A включено в множество, которое получается склейкой множеств S_B и S_C, как описано в теореме.*

В обратную сторону подобное свойство может не выполняться. В частности, для некоторых задач Z_A, Z_B и Z_C справедливо утверждение.

Теорема 13. *Существуют такие означивания σ_0, σ_1 и σ_2 переменных X_0, X_1 и X_2, соответственно, для которых имеет место включение $\sigma_1 \times \sigma_0 \times \sigma_2 \in S_A$, но неверно, что $\sigma_1 \times \sigma_0 \in S_B$ и $\sigma_2 \times \sigma_0 \in S_C$.*

Рассмотрим некоторые отношения между формулой A и определяемой ею задачей Z_A.

Справедлива следующие утверждения.

Теорема 14. *Для выполнимой формулы A в единичном кубе не может существовать область, в каждой точке которой A ложна, и такая, что ни одно действие задачи Z_A не выводит из него.*

Следствие 1. *Всякая формула A определяет разбиение единичного куба на множества K_A, S_A и N_A единственным образом.*

Следствие 2. *Если формула A невыполнима, то существует простая траектория, определяемая задачей Z_A, которая содержит в точности 2^n узлов единичного куба.*

1.4. Выполнимость и доказуемость. В этом разделе исследуем некоторые свойства операционной семантики в ее связи с доказуемостью формул. В частности, выявим соответствие между выполнимостью формулы, как наличием неподвижных точек соответствующей задачи, и выводимостью.

Пусть имеется некоторое правило логического следования, которое позволяет из некоторого множества $\{A_1, A_2, \ldots, A_n\}$ посылок получать заключение B. Обозначим это в виде $A_1, A_2, \ldots, A_n \vdash B$.

Пример 12. Правило резолюции, которое мы будем в дальнейшем исследовать, выглядит следующим образом: $A \vee B, \bar{B} \vee C, \vdash A \vee C$.

Пусть формулы $A_1, A_2, \ldots, A_n, B$ определены над одним множеством X переменных. Если имеет место логическое следование $A_1, A_2, \ldots$ $\ldots, A_n \vdash B$, то всякое означивание переменных из X, для которых все формулы $\{A_1, A_2, \ldots, A_n\}$ истинны, превращает в истину и формулу B. Тем самым, если представить формулу B как совокупность S определяемых ею гиперсфер, то их центры покрывают некоторое множество точек единичного куба, которые принадлежат центрам гиперсфер, определяемых конъюнкцией $A_1 \& A_2 \& \ldots \& A_n$.

Действительно, из того, что $A_1, A_2, \ldots, A_n \vdash B$ есть логическое следование, вытекает, что в каждой точке, в которой опровержима формула B, опровержима конъюнкция $A_1 \& A_2 \& \ldots \& A_n$. Но произвольная формула опровержима только в центрах определяемых ею гиперсфер.

Следовательно, новые следствия из $\{A_1, A_2, \ldots, A_n\}$ не расширяют множества точек, принадлежащих центрам гиперсфер, определяемых формулами $A_1, A_2, \ldots, A_n$.

Пример 13. Правило резолюции по двум центрам гиперсфер, определяемым дизъюнктами $x \vee y$, $\bar{y} \vee z$ (они представлены интервалами соответственно 00_ и _10), строит новую гиперсферу, центр 0_0 которой определяется дизъюнктом $x \vee z$. Этот новый центр включает точки куба 010 и 000, которые уже принадлежат имеющимся центрам.

Таким образом, если имеется некоторое множество M дизъюнктов, то вопрос о его невыполнимости эквивалентен вопросу покрытия центрами гиперсфер всего единичного куба. В терминах правила резолюций это выглядит как возможность вывода из множества M контрарной пары x и $\bar{x}$ для любой переменной x. Как видно, эти два условия эквивалентны, так как центры двух гиперсфер, определяемых литерами x и $\bar{x}$, покрывают весь куб.

Если используется другое правило логического вывода, то в случае его полноты возникает аналогичное необходимое и достаточное условие невыполнимости исходного множества формул.

Пример 14. Рассмотрим к.н.ф. $F(x_1, x_2, \ldots, x_n)$ от n переменных, каждый дизъюнкт которой содержит не более, чем две литеры. Упорядочим все дизъюнкты следующим образом: расположим вначале дизъюнкты, содержащие литеру $\bar{x}_1$, за ними располагаются дизъюнкты, содержащие литеру x_1; из оставшихся аналогично вначале располагаем все дизъюнкты, содержащие литеру $\bar{x}_2$, а затем — дизъюнкты, содержащие литеру x_2, и т. д. Назовем так выделенные группы соответственно $M_{10}, M_{11}, M_{20}, M_{21}, \ldots,$

здесь M_{i0} — совокупность дизъюнктов, в которых содержится литера $\bar{x}_i$, и не содержатся литеры, образованные из переменных с меньшими номерами, M_{i1} — совокупность дизъюнктов, в которых содержится литера x_i, и не содержатся литеры, образованные из переменных с меньшими номерами, $i = 1, 2, \ldots, n$.

Если в одном множестве $M_{i\sigma}$, $\sigma \in \{0, 1\}$, содержится пара дизъюнктов $x_i^\sigma \vee y$, $x_i^\sigma \vee \bar{y}$, то заменяем ее на одну литеру x_i^σ, $i = 1, 2, \ldots, n$. Если в итоге для некоторой переменной x_i будет получена контрарная пара x_i, $\bar{x}_i$, то исходное множество дизъюнктов невыполнимо. В противном случае, используя правило резолюции для пары множеств $M_{10} \cup M_{20} \cup M_{21}$ и $M_{11} \cup M_{20} \cup M_{21}$, аннигилируем в каждом из них только переменную x_2. Очевидно, что когда к дизъюнктам $x_1 \vee x_2$ и $\bar{x}_2 \vee x_q$ применяется правило резолюции, то возникает новый центр, определяемый дизъюнктом $x_1 \vee x_q$. Все возможные применения правила резолюции к этой паре множеств порождают все центры, обладающие следующим свойством.

Если в такой центр входит точка $\sigma_1\sigma_2\sigma_3 \ldots \sigma_n$, то входит и точка $\sigma_1(1 - \sigma_2)\sigma_3 \ldots \sigma_n$, расположенная на расстоянии 1. Следовательно, этот центр включает ребро $\sigma_1\_\sigma_3 \ldots \sigma_n$, определяемое вектором, в котором второй компонент не существенный. Нетрудно увидеть, что в результате подобных преобразований из последующих рассуждений переменная x_2 может быть исключена.

Как и прежде, если в одном из результирующих множеств появляется пара дизъюнктов $x_1^\sigma \vee y$, $x_1^\sigma \vee \bar{y}$, то заменяем ее на одну литеру x_1^σ. Если в результате таких действий возникнет контрарная пара x_1, $\bar{x}_1$, то исходное множество невыполнимо. В противном случае к результирующей паре применяем в точности такие же рассуждения, используя пару множеств M_{30} и M_{31}, и т. д.

Нетрудно понять, что исходная формула будет невыполнима в точности в том случае, когда эта процедура получит контрарную пару x_1, $\bar{x}_1$.

Рассмотрим теперь двойственную задачу нахождения единичного покрытия для множества M дизъюнктов. Очевидно, что невыполнимость этого множества можно доказать, показав, что его единичное покрытие пусто. Так как единичное покрытие для M есть дополнение к множеству определяемых M центров гиперсфер, то эти две задачи эквивалентны. Таким образом, эти два способа установления невыполнимости двойственные. Первый, использующий получение логических следствий, состоит в консервативном расширении множества центров гиперсфер, второй — в анализе покрытия, которое этими гиперсферами определяется.

Представим множество M центрами $\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2, \ldots, \boldsymbol{\sigma}_m$ определяемых им гиперсфер. Тогда единичное покрытие для M совпадает с множеством $\Lambda - \{\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2, \ldots, \boldsymbol{\sigma}_m\}$. Или, если разбить множество M на подмножества $M_1, M_2, \ldots, M_q$, то единичное покрытие для M вычисляется, как пересечение единичных покрытий для каждого из этих множеств.

Пусть множество M_{10} имеет вид: $\{\bar{x}_1 \vee l_1, \bar{x}_1 \vee l_2, \ldots, \bar{x}_1 \vee l_q\}$, где $l_1, l_2, \ldots, l_q$ суть некоторые литеры, и центры, которые определяют эти дизъюнкты, составляют множество $\{1\boldsymbol{\sigma}_1, 1\boldsymbol{\sigma}_2, \ldots, 1\boldsymbol{\sigma}_q\}$. Разность $\Lambda - \{1\boldsymbol{\sigma}_1, 1\boldsymbol{\sigma}_2, \ldots, 1\boldsymbol{\sigma}_q\}$ представлена двумя векторами $0\_\ldots\_$ и $\_\ldots\_1 - \boldsymbol{\sigma}_1\_\ldots\_1 - \boldsymbol{\sigma}_2\_\ldots\_1 - \boldsymbol{\sigma}_q\_\ldots\_$. Аналогично, если множество M_{11} имеет вид: $\{x_1 \vee h_1, x_1 \vee h_2, \ldots, x_1 \vee h_r\}$, где $h_1, h_2, \ldots, h_r$ — литеры, и множество центров, которое определяют эти дизъюнкты, имеет вид $0\boldsymbol{\sigma}'_1, 0\boldsymbol{\sigma}'_2, \ldots, 0\boldsymbol{\sigma}'_r$, то разность $\Lambda - \{0\boldsymbol{\sigma}'_1, 0\boldsymbol{\sigma}'_2, \ldots, 0\boldsymbol{\sigma}'_q\}$ представлена двумя векторами $1\_\ldots\_$ и $\_\ldots\_1 - \boldsymbol{\sigma}'_1\_\ldots\_1 - \boldsymbol{\sigma}'_2\_\ldots\_1 - \boldsymbol{\sigma}'_r\_\ldots\_$. Отсюда получаем, что единичное покрытие для $M_{10} \cup M_{11}$ представляет собой пару $0\_\ldots\_\boldsymbol{\sigma}'_1\_\ldots\_\boldsymbol{\sigma}'_2\_\ldots\_\boldsymbol{\sigma}'_r\_\ldots\_$ и $1\_\ldots\_\boldsymbol{\sigma}_1\_\ldots\_\boldsymbol{\sigma}_2\_\ldots\_\boldsymbol{\sigma}_q\_\ldots\_$.

Аналогично вычисляются единичные покрытия для остальных множеств $M_{i0} \cup M_{i1}$, $i = 1, 2, \ldots, n$.

В итоге получаем, что исходная формула невыполнима тогда и только тогда, когда пересечение построенных таким образом единичных покрытий пусто.

Пусть $G(x_1, x_2, \ldots, x_n) = \&_{q=1,n}(x_q l_1, \ldots, l_{r_q}) \vee (\bar{x}_q h_1, \ldots, h_{s_q})$, где литеры $l_1, \ldots, l_{r_q}, h_1, \ldots, h_{s_q}$ получены из переменных с номерами, большими q, $q = 1, 2, \ldots, n-1$. Единичным покрытием для дизъюнкции $(x_q l_1, \ldots, l_{r_q}) \vee \vee (\bar{x}_q h_1, \ldots, h_{s_q})$ служит объединение интервалов $0\_\ldots\_\sigma'_1\_\ldots\_\sigma'_2\_\ldots\_\sigma'_r\_\ldots\_$ и $1\_\ldots\_\sigma_1\_\ldots\_\sigma_2\_\ldots\_\sigma_q\_\ldots\_$, где значения существенных компонентов определяются показателями у литер $l_1 \ldots l_{r_q}, h_1, \ldots, h_{s_q}$, $q = 1, 2, \ldots, n-1$. Но в точности такое же единичное покрытие определяется формулой $F(x_1, x_2, \ldots, x_n)$ при соответствующим распределении показателей степеней у переменных в дизъюнктах, имеющих длину не более, чем 2. Отсюда получаем, что формула $G(x_1, x_2, \ldots, x_n)$ невыполнима в точности в том случае, когда невыполнима соответствующая к.н.ф., дизъюнкты которой имеют длину не более, чем 2.

Таким образом, задачу установления пустоты совокупностей моделей иногда удобно свести к задаче определения вида всех центров гиперсфер.

§ 2. Задачи первого порядка

2.1. Сведение формул первого порядка к задачам. В этом разделе мы расширим пропозициональный формализм до языка первого порядка, который будем использовать для формулирования задач и описания их операционной семантики. Затем покажем, что всякая формула первого порядка определяет (в общем случае не одну) задачу, неподвижные точки которой суть ее логические модели.

Введем следующие определения.

Пусть $\boldsymbol{x} = (x_1, \ldots, x_n)$, $\boldsymbol{t} = (t_1, \ldots, t_n)$, $n > 0$, $Q = (q_1, \ldots, q_s)$, $s \geqslant 0$ — последовательности соответственно попарно различных переменных, термов, не содержащих переменных из $\boldsymbol{x}$, и произвольных выражений. *Подстановкой* σ из $\boldsymbol{t}$ на места переменных из $\boldsymbol{x}$ называется совокупность $\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$ так называемых *подстановочных компонентов*, а результат σQ ее умножения на Q (называется σ-примером над $\boldsymbol{t}$) получается заменой в Q всякой переменной x_i на терм t_i, $i = 1, 2, \ldots, n$. Когда конкретный вид переменных и термов в последовательностях, соответственно $\boldsymbol{x}$ и $\boldsymbol{t}$ не имеет значения, то подстановку σ будем изображать в сокращенном виде $\{\boldsymbol{x} \leftarrow \boldsymbol{t}\}$. Если σ и θ суть подстановки, то $\sigma\theta Q$ есть результат умножения Q вначале на σ, а результата — на θ. Очевидно, что если переменные подстановок σ и θ образуют непересекающиеся множества, то они коммутативны, т. е. $\sigma\theta Q = \theta\sigma Q$.

Сужение подстановки σ на переменные множества $\boldsymbol{y} = \{y_1, \ldots, y_r\}$, $\boldsymbol{y} \subseteq \boldsymbol{x}$, есть результат выбрасывания из σ подстановочных компонентов $x_i \leftarrow t_i$, таких, что $x_i \notin \boldsymbol{y}$, $i = 1, 2, \ldots, n$. Обратным к сужению является *расширение* подстановки.

Для простоты будем рассматривать формулы первого порядка, находящиеся в предваренной нормальной форме. Пусть формула G имеет вид $Q(D_1 \& \ldots \& D_h)$, где Q есть кванторная приставка, ее матрица находится в к.н.ф., $\mathcal{A}$ есть ее интерпретация с носителем M и $|M| = m$. Полагаем, что отношения и отображения из $\mathcal{A}$, интерпретирующие предикатные, функциональные символы и символы констант из G, обозначены теми же символами.

Образуем по формуле G пропозициональную формулу G^*, заменив всякий квантор $\forall x$ в ее приставке на конъюнкцию $\bigcap\limits_{x\in M}$, а всякий квантор $\exists y$ на дизъюнкцию $\bigcup\limits_{y\in M}$. Если формула G выполнима в области M, то конкретная интерпретация ее предикатных и функциональных символов и констант, при которой она истинна, может быть получена по формуле G^*. Действительно, означивание атомарных формул, при котором истинна формула G^*, позволяет определить соответствующие интерпретации всех предикатных и функциональных символов, а также констант, при которых G истинна. Опишем метод построения такой интерпретации, если она существует.

Следующим образом построим по G^* совокупность T_G множеств бескванторных формул.

Базисный шаг. $T_G=\{G^*\}$.

Индукционный шаг. Пусть уже построено некоторое семейство T_G и некоторому множеству Ω из T_G принадлежит формула X, т. е. $\Omega=\{X\}\cup\Omega_1$. Возможны следующие случаи:

а) $X=\bigcap\limits_{x\in M} D$. В этом случае заменим множество Ω новым $\{D^1,\ldots,D^m\}\cup\Omega_1$, где $D^1,\ldots,D^m$ получаются из D последовательной заменой переменной x каждым элементом из M;

б) $X=\bigcup\limits_{y\in M} D$. В этом случае заменим Ω на m множеств $\{D^1\}\cup\Omega_1,\ldots,\{D^m\}\cup\Omega_1$, где $D^1,\ldots,D^m$ получаются из D последовательной заменой переменной y каждым элементом из M;

г) если все построенные множества содержат формулы, не начинающиеся с символов $\bigcap\limits_{x\in M}$ или $\bigcup\limits_{y\in M}$, то в каждом из них символ конъюнкции заменим на запятую.

В силу конечности кванторной приставки исходной формулы и множества M и того, что всякий раз на шаге а) или б) исключается в точности один символ $\bigcap\limits_{x\in M}$ или $\bigcup\limits_{y\in M}$, порождая конечное множество формул, процедура оканчивает работу после конечного числа шагов. В итоге получим семейство $\Omega_1,\Omega_1,\ldots,\Omega_l$ множеств, где $\Omega_i=\{C_1^i,\ldots,C_{m_i}^i\}$, $i=1,2,\ldots,l$, и C_j^i получаются из $\{D_1,\ldots,D_h\}$ в результате подстановок элементов из M вместо индивидных переменных.

Пример 15. Пусть формула имеет вид $\forall x\forall yF(x,y)$, и множество M состоит из двух элементов $\{a,b\}$. Построенное по этой формуле семейство состоит из четырех множеств: $\{F(a,a),F(b,a)\}$, $\{F(a,b),F(b,a)\}$, $\{F(a,a),F(b,b)\}$, $\{F(a,b),F(b,b)\}$. Для того, чтобы формула $\forall x\forall yF(x,y)$ была истина в интерпретации с носителем M необходимо, чтобы по меньшей мере одно множество из T_G было выполнимым.

Представим каждое из множеств Ω_i, $i=1,2,\ldots,q$, построенной на каком-либо шаге совокупности T_G как конъюнкцию содержащихся в нем формул, а совокупность T_G всех множеств как дизъюнкцию этих конъюнкций. Тогда устранение символов $\bigcap\limits_{x\in M}$ и $\bigcup\limits_{y\in M}$ представляет собой эквивалентные преобразования в силу эквивалентности $A\,\&\,(B\vee C)=(A\,\&\,B)\vee(A\,\&\,C)$. Из построения совокупности T_G вытекает, что множества $\Omega_1,\Omega_2,\ldots,\Omega_l$ отличаются друг от друга лишь подстановками на места переменных, связанных кванторами существования. Все формулы одного множества Ω_i, $i=1,2,\ldots,l$, характеризуются всевозможными подстановками элементов из M на места переменных, связанных кванторами общности, и определенными кортежами элементов из M, подставленных на места переменных, связанных кванторами существования.

В итоге каждое множество Ω_i, $i = 1, 2, \ldots, l$, можно рассматривать как пропозициональную к.н.ф. Полагаем, что все эти формулы зависят от n пропозициональных переменных $p_1, p_2, \ldots, p_n$, и каждая из них определяет задачу, соответственно $Z_1, Z_2, \ldots, Z_l$. Последние характеризуются одним исходным состоянием и различными множествами действий, соответственно $F_1, F_2, \ldots, F_l$.

Что касается состояний задач $Z_1, Z_2, \ldots, Z_l$, то будем использовать несколько иное их представление. В пропозициональном случае формула от n переменных определяет задачу, пространство состояний которой есть единичный n-мерный куб, и каждый компонент n-местного бинарного вектора служит для означивания переменных. Для первопорядковых задач мы рассмотрим следующее представление состояний. Пусть c есть состояние задачи, т. е. бинарный n-местный вектор, который следующим образом разделяет совокупность всех n атомарных отношений на положительные и отрицательные: если его компонент $\sigma_i = 1$, то соответствующий ему атом входит положительно, если $\sigma_i = 0$, то — отрицательно, $i = 1, 2, \ldots, n$. Обозначим совокупность всех положительных атомов Pol, а отрицательных — Neg и будем ассоциировать с состоянием c его *описание* o_c, представляющее собой только множество положительных атомов. В последующем, говоря о применении действия f к состоянию c, мы будем пользоваться прежним обозначением $f(c)$, но при этом рассматривать не бинарный вектор c, а описание o_c этого состояния, выделяя лишь положительные атомы и подразумевая, что остальные входят отрицательно.

При этом применение действия $f = \mathrm{Det}(\boldsymbol{a}) \Rightarrow L(\boldsymbol{a})$ к состоянию c обозначает, что каждая положительная литера из левой части $\mathrm{Det}(\boldsymbol{a})$ содержится в o_c, а отрицательная — не содержится. Тогда $f(c) = d$ и описание o_d состояния d получается добавлением атома $L(\boldsymbol{a})$, если $L(\boldsymbol{a})$ — атом, и вычеркиванием $\bar{L}(\boldsymbol{a})$ из o_c, если $L(\boldsymbol{a})$ — отрицательная литера и ее отрицание содержится в o_c. Нетрудно увидеть, что такое применение действия f полностью аналогично применению, если оставить прежнее представление состояний в виде бинарных векторов.

Задачи, построенные по формуле первого порядка отличаются от задач, порождаемых пропозициональными формулами. Основное отличие состоит в том, что формула первого порядка может определять несколько задач, в то время, как пропозициональная формула определяет в точности одну. Отличия касаются не только количества задач, но и их представлений.

Все действия задач $Z_1, Z_2, \ldots, Z_l$ могут быть получены из дизъюнктов $D_1, \ldots, D_h$ при соответствующей подстановке элементов носителя. Покажем, что по формуле G можно построить единственную задачу Z_G, такую, что ее действия получаются по исходной формуле, а всякая задача из совокупности $Z_1, Z_2, \ldots, Z_l$ есть некоторый пример Z_G над носителем M. Иными словами, непосредственно по исходной формуле G мы определяем ее операционную семантику. В итоге, развертка каждой задачи Z_i, $i = 1, 2, \ldots, l$, может быть получена с помощью действий только одной задачи Z_G. С этой целью рассмотрим два случая.

1. В приставке Q отсутствуют кванторы существования. В этом случае при порождении совокупности T_G не происходит увеличения числа множеств, поэтому $l = 1$ и Ω_1 представляет собой множество дизъюнктов. В соответствии с предыдущими рассуждениями полагаем, что порождена единственная n-мерная пропозициональная задача Z_1. Из ее построения следует, что *всякая неподвижная точка задачи* Z_1 *является моделью формулы* G.

Рассмотрим развертку R_{Z_1} задачи Z_1, и пусть ее действие $f = \mathrm{Det}(\boldsymbol{a}) \Rightarrow$ $\Rightarrow L(\boldsymbol{a})$, где $\mathrm{Det}(\boldsymbol{a})$ есть конъюнкция фундаментальных литер, $L(\boldsymbol{a})$ — литера и $\boldsymbol{a}$ — константы из M. Поставим в соответствие действию f действие

$f^* = \mathrm{Det}(\boldsymbol{x}) \Rightarrow L(\boldsymbol{x})$, где $\boldsymbol{x}$ — это индивидные переменные исходной формулы в том порядке, в котором они перечислены в кванторной приставке, и $\boldsymbol{a} = \sigma\boldsymbol{x}$ для некоторой подстановки σ из M. Здесь $\mathrm{Det}(\boldsymbol{x})$ есть конъюнкция литер со свободными переменными естественным образом связанная с $\mathrm{Det}(\boldsymbol{a})$. Таким образом, по задаче Z_1 построена новая задача Z_G, множество состояний которой совпадает с множеством состояний Z_1, а действия получаются непосредственно из дизъюнктов $D_1, \ldots, D_h$ исходной формулы аналогично тому, как это было сделано в пропозициональном случае. Опишем теперь порядок применения действий задачи Z_G.

Покажем, во-первых, что всякое применение действия $f(c)$ задачи Z_1 можно заменить применением действия $f^*(c)$ задачи Z_G, если использовать соответствующую подстановку на места переменных $\boldsymbol{x}$. С этой целью полагаем, что действие

$$f^* = \mathrm{Det}(\boldsymbol{x}) \Rightarrow L(\boldsymbol{x})$$

применимо к состоянию c, если при некоторой подстановке σ из M на места переменных $\boldsymbol{x}$ конъюнкция $\mathrm{Det}(\sigma\boldsymbol{x})$ включена в o_c. В результате получаем, что $f^*(c) = d$, где состояние d характеризуется описанием o_d, в котором, по сравнению с o_c, добавлен положительный атом $L(\sigma\boldsymbol{x})$, если $L(\boldsymbol{x})$ — положительная литера, или вычеркнута литера $\bar{L}(\sigma\boldsymbol{x})$, если $L(\boldsymbol{x})$ — отрицательная.

В результате такой замены каждого действия f задачи Z_1 на соответствующее действие f^* задачи Z_G при определенной подстановке σ из M на места переменных $\boldsymbol{x}$, развертка R_{Z_1} превращается в развертку R_{Z_G} задачи Z_G, действия которой определены непосредственно по матрице формулы G. Развертка R_{Z_G} отличается от R_{Z_1} тем, что метки ее дуг отличаются из-за присутствия в действиях задачи индивидных переменных. (В общем случае их будет меньше, так как несколько действий задачи Z_1 получаются из одного действия задачи Z_G в результате разных подстановок на места индивидных переменных). Отсюда вытекает, что *всякая неподвижная точка пропозициональной задачи Z_1 является также неподвижной точкой задачи Z_G.*

Теперь в обратную сторону.

Пусть c есть состояние задачи Z_G, в которое ведет траектория

$$c_0 \to f_1 \to f_2 \to \ldots \to f_q \to c.$$

Для каждого действия $f_i(\boldsymbol{x})$, $i = 1, 2, \ldots, q$, этой траектории, если $f_i(e) = d$, то это обозначает, что существует такая подстановка σ из M вместо переменных $\boldsymbol{x}$, что левая часть формулы $f_i(\sigma\boldsymbol{x})$ содержит все положительные атомы описания o_e и содержит отрицания атомов, не включенных в o_e. В результате описание состояния d отличается от описания o_e добавлением или исключением одного атома, что определяется знаком левой части действия f_i.

Но в множестве Ω_1 перечислены результаты всех подстановок на места индивидных переменных исходной формулы. В результате по указанной траектории задачи Z_G можно построить соответствующую траекторию задачи Z_1 из начального состояния в состояние c. Эти рассуждения позволяют сделать вывод о том, что *состояние c есть неподвижная точка задачи Z_1 тогда и только тогда, когда оно есть неподвижная точка задачи Z_G.*

Пример 16. Пусть $G = \forall x \forall y (P(x, y) \supset P(y, x))$. Рассмотрим формулу $\bigcap\limits_{x=a,b,c} \bigcap\limits_{y=a,b,c} (P(x, y) \supset P(y, x))$, которая порождает следующие действия:

$$f_{11} = P(a, b) \Rightarrow P(b, a); f_{12} = \bar{P}(b, a) \Rightarrow \bar{P}(a, b);$$
$$f_{21} = P(a, c) \Rightarrow P(c, a); f_{22} = \bar{P}(c, a) \Rightarrow \bar{P}(a, c);$$
$$f_{31} = P(b, a) \Rightarrow P(a, b); f_{32} = \bar{P}(a, b) \Rightarrow \bar{P}(b, a);$$
$$f_{41} = P(b, c) \Rightarrow P(c, b); f_{42} = \bar{P}(c, b) \Rightarrow \bar{P}(b, c);$$
$$f_{51} = P(c, a) \Rightarrow P(a, c); f_{52} = \bar{P}(a, c) \Rightarrow \bar{P}(c, a);$$
$$f_{61} = P(c, b) \Rightarrow P(b, c); f_{62} = \bar{P}(b, c) \Rightarrow \bar{P}(c, b).$$

Пусть исходное состояние задачи характеризуется описанием $\{P(a, b), P(c, a)\}$. Эта задача обладает четырьмя неподвижными точками:

$$\{P(a, b), P(b, a), P(c, a), P(a, c)\}, \{P(a, b), P(b, a)\}, \{P(c, a), P(a, c)\}, \varnothing.$$

Как легко заметить, они суть модели исходной формулы.

Рассмотрим теперь задачу Z_G, которая определяется правилами $f_1 = P(x, y) \Rightarrow P(y, x)$ и $f_2 = \bar{P}(y, x) \Rightarrow \bar{P}(x, y)$. Легко увидеть, что при том же начальном состоянии ее развертка совпадает с разверткой исходной задачи при замене действий $f_{11}, f_{21}, \ldots, f_{61}$ на f_1 и $f_{12}, f_{22}, \ldots, f_{62}$ на f_2.

Продолжение примера 1. Рассматриваемая интерпретация будет моделью для указанного условия, если ни для одного из двух конъюнктов $F(x) \& \bar{D}(x)$ и $D(x) \& R(x)$ не найдется подстановки, которая делает хотя бы один из них истинным. Но первый из них принимает значение *истина* при подстановке $\{x \leftarrow$ Измайлов$\}$, а второй при подстановке $\{x \leftarrow$ Куприянов$\}$. Следовательно, эта интерпретация не является неподвижной точкой задачи и поэтому не является моделью исходной формулы, т. е. сформулированное условие не выполняется.

Рассмотрим как можно изменить интерпретации так, чтобы в результате получилась модель для сформулированного условия.

Нетрудно увидеть, что после эквивалентных преобразований результирующая задача характеризуется следующими действиями: $f_{11} = \bar{F}(x) \Rightarrow D(x)$, $f_{12} = \bar{D}(x) \Rightarrow \bar{F}(x)$, $f_{21} = D(x) \Rightarrow \bar{R}(x)$ и $f_{22} = R(x) \Rightarrow \bar{D}(x)$. В соответствии с действием f_{11} к отношению D (Общая дирекция) следует добавить Измайлова, а в соответствии с действием f_{21} из отношения R (Ревизионная комиссия) следует вычеркнуть Куприянова. В результате получаются такие отношения: Финансовый комитет (F): Иванов, Светлов, Измайлов, Севостьянов; Общая дирекция (D): Дубовицкий, Смирнов, Иванов, Светлов, Севостьянов, Куприянов, Измайлов; Ревизионная комиссия (R): Измайлов, Алексеев, Сушенков, Храмов, Борисов.

Эти отношения также не представляют собой модель для сформулированного критерия, так как применимы действия $f_{21} = D(x) \Rightarrow \bar{R}(x)$ и $f_{22} = R(x) \Rightarrow \bar{D}(x)$ при подстановке $\{x \leftarrow$ Измайлов$\}$. В соответствии с действием f_{21} вычеркнем Измайлова из ревизионной комиссии и получим новые отношения: Финансовый комитет (F): Иванов, Светлов, Измайлов, Севостьянов; Общая дирекция (D): Дубовицкий, Смирнов, Иванов, Светлов, Севостьянов, Куприянов, Измайлов; Ревизионная комиссия (R): Алексеев, Сушенков, Храмов, Борисов.

Результирующие отношения образуют искомую модель, так как при любой подстановке обе формулы $F(x) \& \bar{D}(x)$ и $D(x) \& R(x)$ истины.

Пример 17. Построим модель с носителем $\{a, b\}$ для формулы $\forall x \forall y (P(x, y) \supset (P(y, x) \vee Q(x))$. Эта формула порождает действия:

$$f_1 = \bar{P}(y, x) \,\&\, \bar{Q}(x) \Rightarrow \bar{P}(x, y); \quad f_2 = P(x, y) \,\&\, \bar{Q}(x) \Rightarrow P(y, x);$$
$$f_3 = P(x, y) \,\&\, \bar{P}(y, x) \Rightarrow Q(x).$$

Пусть начальное состояние c задачи характеризуется описанием $o_c = \{P(a, b)\}$. Применение действия f_1 к состоянию c порождает состояние d_1, не содержащее положительных литер, а применение действия f_3 порождает состояние d_2 с описанием $\{P(a, b), Q(a)\}$. Состояние d_1 есть неподвижная точка, так как действия f_2, f_3 не применимы, а f_1 не добавляет положительных компонент. Состояние d_2 также неподвижная точка, так как действие f_2 не применимо, а действия f_1, f_3 не добавляют положительных компонентов.

В силу эквивалентности $\bigcap\limits_{x \in M} \bigcap\limits_{y \in M} (P(x, y) \supset (P(y, x) \vee Q(x)) = (P(a, a) \supset$ $\supset (P(a, a) \vee Q(a)) \& (P(a, b) \supset (P(b, a) \vee Q(a))) \& (P(b, a) \supset (P(a, b) \vee Q(b))) \&$ $\& (P(b, b) \supset (P(b, b) \vee Q(b)))$, оба состояния суть искомые модели.

2. Рассмотрим теперь общий случай, когда в кванторной приставке формулы G присутствуют кванторы существования. Преобразование такой формулы, состоящее в замене кванторов логическими операциями, генерирует несколько пропозициональных задач. Если допустить для всех этих задач одно начальное состояние, то они будут различаться лишь действиями.

Покажем, что общий случай можно свести к предыдущему случаю универсальной формулы.

Легко увидеть соответствие между сколемовской формой формулы G и порождаемым по ней семейством T_G. Действительно, в каждом множестве $\Omega_i \in T_G$, $i = 1, 2, \ldots, l$, на места всех переменных, связанных кванторами общности, подставляются всевозможные кортежи из элементов носителя интерпретации. В то время, как на места всех переменных, связанных кванторами существования, во всех формулах из одного множества Ω_i подставлены лишь определенные наборы элементов носителя. Следовательно, каждое множество $\Omega_i \in T_G$, $i = 1, 2, \ldots, l$, описывает один из возможных способов интерпретации сколемовских функций, а вся совокупность T_G задает все возможные интерпретации сколемовских функций.

Исходя из этих рассуждений, можно ограничиться рассмотрением лишь универсальных формул, расширив сигнатуру за счет сколемовских функций. Но сколемовские функции исходно не интерпретируются, поэтому их интерпретации должны определяться дополнительно.

Пример 18. Рассмотрим формулу $\exists x \forall y \exists z ((G(x) \supset H(x)) \supset$ $\supset (G(y) \supset H(z))$. Ее сколемовская форма выглядит следующим образом: $\forall y ((G(c) \supset H(c)) \supset (G(y) \supset H(f_z(y)))$. После приведения матрицы к к.н.ф. и отбрасывания кванторов общности, получается формула: $(G(c) \vee \bar{G}(y) \vee H(f_z(y))) \& (\bar{H}(c) \vee \bar{G}(y) \vee H(f_z(y)))$.

Эта формула порождает следующие действия: $f_{11} = \bar{G}(c) \,\&\, G(y) \Rightarrow$ $\Rightarrow H(f_z(y))$; $f_{12} = \bar{G}(c) \,\&\, \bar{H}(f_z(y)) \Rightarrow \bar{G}(y)$; f_{13}: $G(y) \,\&\, \bar{H}(f_z(y)) \Rightarrow G(c)$; $f_{21} = G(y) \,\&\, H(c) \Rightarrow H(f_z(y))$; $f_{22} = G(y) \,\&\, \bar{H}(f_z(y)) \Rightarrow \bar{H}(c)$; $f_{23} = H(c) \,\&$ $\& \bar{H}(f_z(y)) \Rightarrow \bar{G}(y)$.

Пусть множество $M = \{a, b\}$ есть носитель интерпретации и начальное состояние задачи имеет вид $\{G(a), H(a)\}$. Полагая, что сколемовская константа c интерпретируется как элемент b носителя, получаем развертку с тремя неподвижными точками: $\{G(a), H(a), H(f_z(a))\}$, $\{H(a)\}$ и $\{G(a), G(b), H(a)\}$.

Следующий пример хорошо иллюстрирует, что при различных интерпретациях сколемовских формул получаются разные задачи.

Пример 19. Пусть формула G имеет вид $\forall x \exists y (P(x, y) \supset Q(x))$. Для наглядности зафиксируем различные области значений переменных x и y, соответственно $M_x = \{A, B\}$ и $M_y = \{1, 2, 3\}$. Пропозициональная формула G^* имеет вид: $\bigcap\limits_{x \in M} \bigcup\limits_{y \in M} (P(x, y) \supset Q(x))$. По ней однозначно восстанавливается множество T_G:

$$\{\bar{P}(A,1) \vee Q(A), \bar{P}(B,1) \vee Q(B)\}, \quad \{\bar{P}(A,2) \vee Q(A), \bar{P}(B,1) \vee Q(B)\},$$

$$\{\bar{P}(A,3) \vee Q(A), \bar{P}(B,1) \vee Q(B)\}, \quad \{\bar{P}(A,1) \vee Q(A), \bar{P}(B,2) \vee Q(B)\},$$

$$\{\bar{P}(A,2) \vee Q(A), \bar{P}(B,2) \vee Q(B)\}, \quad \{\bar{P}(A,3) \vee Q(A), \bar{P}(B,2) \vee Q(B)\},$$

$$\{\bar{P}(A,1) \vee Q(A), \bar{P}(B,3) \vee Q(B)\}, \quad \{\bar{P}(A,2) \vee Q(A), \bar{P}(B,3) \vee Q(B)\},$$

$$\{\bar{P}(A,3) \vee Q(A), \bar{P}(B,3) \vee Q(B)\}.$$

Каждая задача, которая может быть построена по исходной формуле, получается из дизъюнкта $\bar{P}(x, \varphi(x)) \vee Q(x)$ при определенной интерпретации сколемовской функции φ. Для задачи, которая определяется множеством $\{\bar{P}(A, 1) \vee Q(A), \bar{P}(\mathrm{B}, 2) \vee Q(B)\}$, получаем действия: $f_{11} = P(A, 1) \Rightarrow Q(A)$; $f_{12} = \bar{Q}(A) \Rightarrow \bar{P}(A, 1)$; $f_{21} = P(B, 2) \Rightarrow Q(B)$; $f_{22} = \bar{Q}(B) \Rightarrow \bar{P}(B, 2)$.

Легко увидеть, что состояние c с описанием $\{P(A, 1), P(B, 2)\}$, не является неподвижной точкой, так как применения действий f_{11} и f_{21} приводит к новому состоянию с описанием $\{P(A, 1), P(B, 2), Q(A), Q(B)\}$. Последнее является неподвижной точкой задачи и моделью исходной формулы. Другой неподвижной точкой этой задачи будет модель с пустой положительной частью, которая получается в результате применения действий f_{12} и f_{22}.

Если выбрать другую интерпретацию сколемовской функции, определяемую, например, множеством: $\{\bar{P}(A, 2) \vee Q(A), \bar{P}(B, 3) \vee Q(B)\}$, то состояние c будет неподвижной точкой, и следовательно, моделью исходной формулы.

Таким образом всякая формула первого порядка порождает целый класс задач, отличающихся друг от друга различными интерпретациями сколемовских функций.

При таком определении задач по формулам первого порядка имеется как сходство с пропозициональными задачами, так и различие. Как и в пропозициональном случае, при построении развертки наблюдается немонотонность интерпретаций, определяемых разными состояниями. Разные состояния характеризуются разными интерпретациями.

Сходство также указывается следующими теоремами.

Теорема 15. *Пусть f есть действие задачи Z_G, c, d — суть состояния такие, что $f(c) = d$ и $o_c \neq o_d$. Тогда при интерпретации предикатных и функциональных символов, определяемых состоянием c, формула f ложна.*

Доказательство. Пусть $f(\boldsymbol{x}) = \Gamma(\boldsymbol{x}) \Rightarrow L(\boldsymbol{x})$. Его применимость в c, в результате которой получается новое состояние d, обозначает наличие подстановки $\sigma = \{\boldsymbol{x} \leftarrow \boldsymbol{a}\}$ такой, что $\Gamma(\boldsymbol{a}) \subseteq o_c$ и $L(\boldsymbol{a}) \notin o_c$. Но тогда формула $f(\boldsymbol{a})$ ложна в o_c, если трактовать $\Rightarrow$ как импликацию.

Теорема доказана.

Теорема 16. *Пусть f и g суть действия задачи Z_G, полученные из одного дизъюнкта формулы G, и состояние c есть неподвижная точка для f. Тогда c есть неподвижная точка и для g.*

Доказательство. Действительно, o_c есть модель для f. Но формулы f и g логически эквивалентны. Следовательно, o_c есть модель для g. Следовательно, c есть неподвижная точка для g.

Теорема доказана.

Теорема 17. *Пусть f есть действие задачи Z_G с заданной интерпретацией сколемовских функций, c, d суть состояния и $f(c)=d$. Тогда формула $\&o_c \& f \supset \&o_d$ общезначима. (Здесь $\&o_c$ и $\&o_d$ обозначает конъюнкцию всех атомов соответственно в o_c и o_d.)*

Теорема 18. *Пусть действие f задачи Z_G, получено из дизъюнкта D, c, d — суть состояния такие, что $f(c)=d$ и $o_c \neq o_d$. Тогда все остальные действия, полученные из того же дизъюнкта, также применимы к c, порождая состояния с попарно различными описаниями.*

Доказательство. Пусть действие $f(\boldsymbol{x}) = \Gamma(\mathbf{x}) \Rightarrow L(\mathbf{x})$. Его применимость в состоянии c в результате которой получается состояние d, обозначает наличие подстановки, $\{\mathbf{x} \leftarrow \mathbf{a}\}$ такой, что все положительные атомы $\Gamma(\mathbf{a})$ содержатся в o_c, отрицания отрицательных литер — не содержатся, $L(\mathbf{a}) \notin o_c$, если $L(\mathbf{x})$ — это положительная литера и $\bar{L}(\mathbf{a}) \notin o_c$, если $L(\mathbf{x})$ — отрицательная. Но тогда всякое действие, полученное из того же дизъюнкта, имеет вид: $\bar{L}(\mathbf{x})\&\Phi(\mathbf{x}) \Rightarrow H(\mathbf{x})$. Видно, что оно также применимо к c, порождая состояние, описание которого отлично от писания o_c.

Теорема доказана.

Такой случай демонстрируется следующим примером.

Пример 20. Пусть действия f и f^* получены из одного дизъюнкта $\bar{P}(x) \vee Q(x)$, имеют вид, соответственно: $P(x) \Rightarrow Q(x)$ и $\bar{Q}(x) \Rightarrow \bar{P}(x)$, и состояние c обладает описанием $o_c = \{P(a)\}$. Тогда $f(c)=d$ и $o_d = \{P(a), Q(a)\}$, $f^*(c)=e$ и $o_e = \varnothing$. Состояния d и e имеют разные описания.

Различие в первопорядковых и пропозициональных задачах заключается в том, что при выполнении условий теоремы 15 формула f может оказаться ложной в o_d. Проиллюстрируем это примером.

Пример 21. Пусть действие f имеет вид: $P(x) \Rightarrow P(h(x))$, состояние c имеет описание $\{P(a)\}$ и h есть циклический сдвиг: $h(a)=b$, $h(b)=e$, $h(e)=a$. Тогда $f(c)=d$ и $o_d = \{P(a), P(b)\}$. Формула f ложна в состоянии d, так как ее посылка истина, а заключение — ложно. Следовательно, d не является неподвижной точкой и o_d не является моделью для формулы f.

Естественно возникает вопрос о возможности иной интерпретации действий задач. Это сводится к расширению языка действий, когда действия могут представлять собой не только универсальные формулы. Рассмотрим следующий пример.

Продолжение примера 19. Преобразуем формулу G к эквивалентному виду: $\forall x(\forall y P(x, y) \supset Q(x))$ и определим ее операционную семантику следующим образом: действие $f = \forall y P(x, y) \Rightarrow Q(x)$ применимо в состоянии c, если в описании o_c отношение, интерпретирующее предикат P, включает все пары $\{a\} \times M_Y$, где $a \in M_X$. В этом случае порождается новое состояние путем добавления $Q(a)$ для элемента a. Содержательно это обозначает, что к описанию добавляется новый элемент, когда посылка правила истинна для всех элементов множества M_Y при фиксированном элементе из M_X.

Очевидно, что в этом случае состояние c будет неподвижной точкой действия f.

Нетрудно увидеть, что подобные рассуждения имеют место, когда мы вводим новые понятия. Например, включение множеств определяется следующим образом: $\forall X \forall Y(\forall x(x \in X \supset x \in Y) \supset X \subseteq Y)$.

В случае применения действий задач, определенного в этом разделе, (назовем его *нормальным*) каждая неподвижная точка есть логическая модель исходной формулы. Соответствие здесь имеет место в обе стороны: если найдена неподвижная точка, то она есть модель, и, с другой стороны, всякая модель формулы есть неподвижная точка задачи.

Любое ограничение нормального применения влечет сужение класса получаемых моделей, так как мы накладываются более жесткие ограничения на применение действий. Поэтому в общем случае множества неподвижных точек одной задачи при различных применениях действий различаются.

Пример 22. Пусть действие f имеет вид: $P(x)\,\&\,\bar{R}(x, y) \Rightarrow Q(x)$, и состояние c имеет описание $o_c = \{P(a), R(a, b)\}$. Тогда при нормальном применении $f(c) = d$, где $o_d = \{P(a), R(b, a), Q(a)\}$, т. е. c не является неподвижной точкой.

Допустим теперь, что то же действие применяется следующим образом: левая часть принимает значение *истина* в состоянии c, если при любой подстановке θ над M на место переменной y отношение $R(\sigma x, \theta y)$ не встречается в описании o_c. Легко увидеть, что такое применение выражается формулой $P(x)\&\forall y\bar{R}(x, y) \Rightarrow Q(x)$. В итоге получаем, что $f(c) = c$, т. е. c есть неподвижная точка для $P(x)\&\forall y\bar{R}(x, y) \Rightarrow Q(x)$.

§ 3. Структурированные задачи

Приведенное выше определение задачи достаточно широкое и строить на его базе прикладную теорию затруднительно. Поэтому введем некоторое уточнение этого понятия, а именно определим так называемые *структурированные задачи*. Для этого уточним вид состояний и действий.

Пусть $\boldsymbol{O}$ есть множество (объектов), $\boldsymbol{R} = \{R_1^{n_1}, \ldots, R_s^{n_s}\}$ — совокупность $R_1^{n_1} \subseteq \boldsymbol{O}^{n_1}, \ldots, R_s^{n_s} \subseteq \boldsymbol{O}^{n_s}$ (*базисных*) отношений.

Каждое состояние c характеризуется своим описанием o_c, которое представимо в виде совокупности $\{R_1^c, \ldots, R_s^c\}$ отношений, где $\{R_1^c \subseteq R_1^{n_1}, \ldots, R_s^c \subseteq R_s^{n_s}\}$. Таким образом, описание всякого состояния представляет собой совокупность отношений, каждое из которых есть часть соответствующего базисного отношения. Множество всех состояний обозначим C.

Исходное состояние c_0 характеризуется описанием $o_{c_0} = \{R_1^0, \ldots, R_s^0\}$.

Задачей Z, как и прежде, называется четверка $(C, F, c_0, C_{\text{fin}})$, где F есть совокупность $\{f\colon C \to C\}$действий, c_0 — исходное состояние и C_{fin}, $C_{\text{fin}} \subseteq C$ — множество финальных состояний. В реальных задачах финальные состояния задаются определенными отношениями между элементами множества $\boldsymbol{O}$.

В дальнейшем мы будем иметь дело с формулами первого порядка, в которых явно выделяются два непересекающихся множества предикатных символов: базисные, они интерпретируются отношениями из $\boldsymbol{R}$, и вспомогательные, они интерпретируются иначе. Чтобы не вводить сложного формализма, обозначим предикатные символы первого типа теми же символами, которыми обозначены базисные отношения.

Аргументами всех предикатов будут наборы термов, построенные из функциональных символов фиксированного словаря $\boldsymbol{\Phi}$ и индивидных переменных. Когда конкретный вид термов не важен, будем изображать их в сокращенном виде $\boldsymbol{t}_1(\boldsymbol{x}_1), \ldots, \boldsymbol{t}_r(\boldsymbol{x}_r)$.

Пусть $\mathcal{T} = (\boldsymbol{P}, \boldsymbol{\Phi}, \boldsymbol{O}, \vdash)$ есть прикладная теория первого порядка, где $\boldsymbol{P}$ — множество предикатных символов, отличных от базисных, $\boldsymbol{\Phi}$ — мно-

жество функциональных символов и $\vdash$ — отношение следования. Назовем сигнатуру $(\boldsymbol{P}, \boldsymbol{\Phi}, \boldsymbol{O})$ *вспомогательной* в отличие от базисной $(\boldsymbol{R}, \boldsymbol{\Phi}, \boldsymbol{O})$.

3.1. Детерминативы и следование в пространстве состояний. В этом разделе мы опишем формулы определенного вида — так называемые *детерминативы*, которые играют существенную роль в последующем, и отношение выполнимости базисных формул, зависящее от состояний задачи. Затем выделим класс подстановок, назовем их *ситуации*, которые превращают детерминатив в истинную формулу в данном состоянии. Определяемые таким образом подстановки используются при получении новых состояний.

Назовем *детерминативом* формулу

$$R_{i_1}(\psi_1(\boldsymbol{x})) \,\&\, \ldots \,\&\, R_{i_m}(\psi_m(\boldsymbol{x})) \,\&\, \forall \boldsymbol{y}(R_{j_1}^{\sigma_1}(\zeta_1(x, y)) \,\&\, \ldots \,\&\, R_{j_n}^{\sigma_n}(\zeta_n(\boldsymbol{x}, \boldsymbol{y})))$$

в базовой сигнатуре, где $m > 0$, $n \geqslant 0$ и $\sigma_1, \ldots, \sigma_n \in \{0, 1\}$.

Подформулы

$$R_{i_1}(\psi_1(\boldsymbol{x})) \,\&\, \ldots \,\&\, R_{i_m}(\psi_m(\boldsymbol{x}))$$

и

$$\forall \boldsymbol{y}(R_{j_1}^{\sigma_1}(\zeta_1(\boldsymbol{x}, \boldsymbol{y})) \,\&\, \ldots \,\&\, R_{j_n}^{\sigma_n}(\zeta_n(\boldsymbol{x}, \boldsymbol{y})))$$

детерминатива называются соответственно *определяющей* и *уточняющей*, а переменные $\boldsymbol{x}$ — *определенными*. Детерминатив без уточняющей части называется *положительным*. Если нас не интересует детальный вид детерминатива, то будем обозначать его $\mathrm{DET}(\boldsymbol{x})$, явно указывая его определенные переменные.

Определим отношение выполнимости формул базисной сигнатуры в зависимости от состояния задачи.

Пусть R есть n-местный базисный предикатный символ, c - состояние, R^c — часть описания o_c и $(a_1, \ldots, a_n) \in \boldsymbol{O}^n$. Тогда, если $(a_1, \ldots, a_n) \in R^c$, то будем говорить, что предикат R истинен в состоянии c для аргументов $a_1, \ldots, a_n$, и обозначать это $\vdash^c R(a_1, \ldots, a_n)$, если же $(a_1, \ldots, a_n) \notin R^c$, то будем говорить, что предикат R ложен в состоянии c для аргументов $a_1, \ldots, a_n$, и обозначать $\not\vdash^c R(a_1, \ldots, a_n)$. Таким образом, истинность или ложность базисных предикатов зависят не только от их аргументов, но и от состояния задачи. Один и тот же предикат может быть истинным в одном состоянии, но ложным в другом.

Истинность и ложность базисных формул в состоянии c определяются обычным образом. Для произвольных формул A и B имеют место следующие соотношения: $\vdash^c A \,\&\, B \Leftrightarrow \vdash^c A$ и $\vdash^c B$; $\vdash^c A \vee B \Leftrightarrow \vdash^c A$ или $\vdash^c B$; $\vdash^c A \supset B \Leftrightarrow$ из $\vdash^c A$ вытекает $\vdash^c B$; $\vdash^c \bar{A} \Leftrightarrow \not\vdash^c A$. $\vdash^c \forall x A(x) \Leftrightarrow$ для всякого элемента $a \in \boldsymbol{O}$ имеет место отношение $\vdash^c A(a)$; $\vdash^c \exists x A(x) \Leftrightarrow$ найдется некоторый элемент $a \in \boldsymbol{O}$ такой, что имеет место отношение $\vdash^c A(a)$.

Будем говорить, что из совокупности формул $A_1, \ldots, A_n$ в состоянии c *следует* формула B, обозначается $A_1, \ldots, A_n \vdash^c B$, если из отношений $\vdash^c A_1, \ldots, \vdash^c A_n$ вытекает отношение $\vdash^c B$.

Пусть $\mathrm{DET}(\boldsymbol{x})$ есть детерминатив, c — состояние задачи. Тогда *ситуацией*, определяемой этим детерминативом в состоянии c, называется множество DET^c всех подстановок σ вида $\{x_1 \leftarrow a_1, \ldots, x_n \leftarrow a_n\}$, где $\boldsymbol{x} = \{x_1, \ldots, x_n\}$, $a_1, \ldots, a_n \in \boldsymbol{O}$ таких, что выполняется отношение $\vdash^c \mathrm{DET}(\sigma \boldsymbol{x})$.

Пример 23. Пусть $\mathrm{DET}(x) = P(x) \,\&\, \forall y \bar{R}(x, y)$, $\boldsymbol{O} = \{a, b\}$, $o_c = \{P(a)\}$. Тогда DET^c включает подстановку $\{x \leftarrow a\}$, так как ни один из атомов $\{R(a, a), R(a, b)\}$ не принадлежит o_c и поэтому выполняется отношение $\vdash^c P(a) \,\&\, \forall y \neg R(a, y)$. С другой стороны, DET^c не включает подстановку $\{x \leftarrow b\}$, так как $P(b)$ не входит в o_c.

Поясним определение ситуации следующим образом.

Из определений истинности формулы в состоянии и ситуации вытекает, что если $\sigma \in \mathrm{DET}^c$, в определяющую часть детерминатива $\mathrm{DET}(\boldsymbol{x})$ входит атом $R(\boldsymbol{\psi}(\boldsymbol{x}))$, то $\sigma\boldsymbol{\psi}(\boldsymbol{x}) \in R^c$; если в его уточняющую часть входит атом $R(\boldsymbol{\zeta}(\boldsymbol{z}))$, то $\sigma\theta\boldsymbol{\zeta}(\boldsymbol{z}) \in R^c$ при любой подстановке θ из $\boldsymbol{O}$ на места не определенных переменных; если в его уточняющую часть входит литера $\bar{R}(\boldsymbol{\zeta}(\boldsymbol{z}))$, то $\sigma\theta\boldsymbol{\zeta}(\boldsymbol{z}) \notin R^c$ при любой подстановке θ из $\boldsymbol{O}$ на места не определенных переменных.

Всякое действие задачи представимо в виде

$$\mathrm{DET}(\bar{x}) \,\&\, L(\boldsymbol{x}) \Rightarrow R^{\sigma_1}_{k_1}(\boldsymbol{\varphi}_1(\boldsymbol{x})) \,\&\, \ldots \,\&\, R^{\sigma_t}_{k_t}(\boldsymbol{\varphi}_t(\boldsymbol{x})),$$

где $L(\boldsymbol{x})$ и $\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_t$ — соответственно формула и термы вспомогательной сигнатуры; $\sigma_1, \ldots, \sigma_t \in \{0, 1\}$, и в правой части нет одноименных предикатов, входящих одновременно положительно и отрицательно.

Его выполнение происходит следующим образом. Пусть c есть некоторое состояние, f — действие и $f(c) = d$. Тогда $o_d = \{R^c_1 + \Delta R^c_1, \ldots, R^c_s + \Delta R^c_s\}$, где $\Delta R^c_1, \ldots, \Delta R^c_s$ суть изменения отношений соответственно $R^c_1, \ldots, R^c_s$: если $\sigma_i = 1$, то изменение ΔR^c_i — положительное, если $\sigma_i = 0$, то ΔR^c_i — отрицательное; $R^c_i + \Delta R^c_i$ обозначает, что фрагмент ΔR^c_i добавляется к отношению R^c_i, если изменение положительное, и исключается из него, если оно отрицательное, $i = k_1, \ldots, k_t$.

Всякое изменение содержит все кортежи, получаемые в результате выполнения следующих действий.

Для всякой подстановки $\sigma \in \mathrm{DET}^c$, если формула $L(\sigma\boldsymbol{x})$ принадлежит теории $\mathcal{T}$, то ΔR^c_i включает кортеж, образованный из $\boldsymbol{\varphi}_i(\sigma\boldsymbol{x})$, приведением к некоторому фиксированному (назовем его *нормальным*) виду каждого его элемента, в противном случае не включает, $i = k_1, \ldots, k_t$. Под нормальным понимается заранее оговоренный представитель класса эквивалентности. Например, для терма $\mathbf{2} + \mathbf{3}$ таким представителем будет $\mathbf{5}$ при стандартной интерпретации сложения над областью целых чисел.

Нетрудно увидеть, что для всякого состояния и любого действия результат его применения в этом состоянии определяется однозначно.

3.2. Некоторые свойства структурированных задач. Исследуем свойства структурированных задач, связанные с их логической интерпретацией. С этой целью обозначим $\boldsymbol{Z}$ прикладную теорию первого порядка, получающуюся добавлением к теории $\mathcal{T}$ в качестве аксиом всех действий задачи Z, трактуя каждое действие как универсальную формулу и знак $\Rightarrow$ как импликацию. Результирующая теория обладает сигнатурой, объединяющей базовую и вспомогательную. Расширим отношение $\vdash^c$ на формулы объединенной сигнатуры путем следующего добавления.

Если P есть n-арный вспомогательный предикатный символ, то полагаем, что он истинен для аргументов $a_1, \ldots, a_n$ в состоянии c, обозначим это $\vdash^c P(a_1, \ldots, a_n)$, тогда и только тогда, когда атом $P(a_1, \ldots, a_n)$ принадлежит теории $\mathcal{T}$. Тем самым, истинность или ложность вспомогательных предикатов не зависит от состояния задачи, а определяется лишь их аргументами.

Таким образом, отношение $\vdash^c$ определено для всякой формулы исчисления $\boldsymbol{Z}$.

Покажем, что применение произвольного действия в любом состоянии равносильно получению логического следования в этом состоянии.

Справедливо следующее утверждение.

Теорема 19. *Пусть действие f имеет вид*

$$POL(\boldsymbol{x}) \,\&\, \forall \boldsymbol{y} NEG(\boldsymbol{x}, \boldsymbol{y}) \,\&\, L(\boldsymbol{x}) \Rightarrow R^{\sigma_1}_{k_1}(\boldsymbol{\varphi}_1(\sigma\boldsymbol{x})) \,\&\, \ldots \,\&\, R^{\sigma_t}_{k_t}(\boldsymbol{\varphi}_t(\sigma\boldsymbol{x})),$$

для состояний c и d выполняется равенство $f(c)=d$, и $o_c \neq o_d$. Тогда справедливо отношение: $\not\vdash^c \forall \boldsymbol{x} f$.

Доказательство. Пусть σ есть произвольная подстановка из $\boldsymbol{O}$ на места переменных $\boldsymbol{x}$. Из определения логического следования в состоянии c вытекает, что достаточно ограничиться рассмотрением лишь подстановок из DET^c, так как при остальных подстановках формула $POL(\boldsymbol{x})$ ложна в состоянии c. Формула σf имеет вид:

$$POL(\sigma\boldsymbol{x}) \,\&\, \forall \boldsymbol{y} NEG(\sigma\boldsymbol{x}, \boldsymbol{y}) \,\&\, L(\sigma\boldsymbol{x}) \Rightarrow R_{k_1}^{\sigma_1}(\varphi_1(\sigma\boldsymbol{x})) \,\&\, \ldots \,\&\, R_{k_t}^{\sigma_t}(\varphi_t(\sigma\boldsymbol{x})).$$

Каждое отношение из $POL(\sigma\boldsymbol{x})$, встречается в o_c, и поэтому имеет место отношение $\vdash^c POL(\sigma\boldsymbol{x})$. Следовательно, можно перейти к рассмотрению формулы:

$$\forall \boldsymbol{y} NEG(\sigma\boldsymbol{x}, \boldsymbol{y}) \,\&\, L(\sigma\boldsymbol{x}) \Rightarrow R_{k_1}^{\sigma_1}(\varphi_1(\sigma\boldsymbol{x})) \,\&\, \ldots \,\&\, R_{k_t}^{\sigma_t}(\varphi_t(\sigma\boldsymbol{x})).$$

При любой подстановке θ на места переменных $\boldsymbol{y}$ все положительные базисные отношения из $NEG(\sigma\boldsymbol{x}, \theta\boldsymbol{y})$ содержится в o_c, а отрицательные — не содержатся. Следовательно, имеет место отношение $\vdash^c NEG(\sigma\boldsymbol{x}, \theta\boldsymbol{y})$.

Из принадлежности формулы $L(\sigma\boldsymbol{x})$ теории $\mathcal{T}$ следует, что каждый положительный атом $R_{k_j}(\varphi_j(\sigma\boldsymbol{x}))$ из заключения формулы σf принадлежит описанию o_d. Если же в заключение входит отрицательная литера $\bar{R}_{k_j}(\varphi_j(\sigma\boldsymbol{x}))$, то кортеж $\varphi_j(\sigma\boldsymbol{x})$ термов не принадлежит отношению $R_{k_j}^d$, $j=1, 2, \ldots, t$.

По условию $o_c \neq o_d$. Следовательно, при получении состояния d произошли изменения по сравнению с состоянием c, т. е. добавились или исключились некоторые фрагменты базисных отношений. Но это значит, что для некоторой подстановки $\lambda \in \mathrm{DET}^c$ левая часть импликации

$$POL(\lambda\boldsymbol{x}) \,\&\, \forall \boldsymbol{y} NEG(\lambda\boldsymbol{x}, \boldsymbol{y}) \,\&\, L(\lambda\boldsymbol{x}) \Rightarrow R_{k_1}^{\sigma_1}(\varphi_1(\lambda\boldsymbol{x})) \,\&\, \ldots \,\&\, R_{k_t}^{\sigma_t}(\varphi_t(\lambda\boldsymbol{x}))$$

истинна в c, а правая — ложна. Следовательно, имеет место отношение $\not\vdash^c \lambda f$. Но так как $\forall \boldsymbol{x} f$ есть универсальная формула, то верно $\not\vdash^c \forall \boldsymbol{x} f$.

Теорема доказана.

Следствие 1. *Пусть для состояний c, d и действия f выполняется равенство $f(c)=d$ и $o_c=o_d$. Тогда верно, что $\vdash^c \forall \boldsymbol{x} f$.*

Следствие 2. *Пусть c есть неподвижная точка задачи. Тогда для любого действия $f(x_1, \ldots, x_n)$ этой задачи имеет место отношение $\vdash^c \forall x_1 \ldots \forall x_n f(x_1, \ldots, x_n)$.*

Замечание. Если для состояний c, d и действия f выполняется равенство $f(c)=d$, и $o_c \neq o_d$, то из этого еще не вытекает отношение $\vdash^d \forall \boldsymbol{x} f$. Действительно, возможен случай, когда $f(d)=e$, и $o_d \neq o_e$, что имеет место, когда в правой и левой частях формулы f встречаются одинаковые базовые предикаты. Тогда выполняется условие леммы и поэтому $\not\vdash^d \forall \boldsymbol{x} f$.

Проиллюстрируем это замечание следующим примером.

Пример 24. Пусть действие $f = R(x) \Rightarrow R(x+1)$ применяется в состоянии c, описание которого состоит из отношения $\{R(0)\}$. Тогда $f(c)= = d$, и $o_d = \{R(0), R(1)\} \neq o_c$. Но o_d не есть модель для формулы $\forall x f$, и поэтому имеет место отношение $\not\vdash^d \forall x f$.

Применение действия к произвольному состоянию можно представить как отработку управляющего воздействия некоторой регулирующей системой. Целевая функция системы состоит в том, чтобы в итоге для каждого действия f выполнялось отношение $\vdash^c \forall \boldsymbol{x} f$. Если отношение $\vdash^c \forall \boldsymbol{x} f$ для

действия f выполняется, то оно не применимо, если же нарушается, то применимо. В результате система переходит в новое состояние, в котором действие, как логическая формула, может принять истинное значение.

Теорема 20. *Пусть действие f имеет вид*

$$POL(\boldsymbol{x}) \,\&\, \forall \boldsymbol{y} NEG(\boldsymbol{x}, \boldsymbol{y}) \,\&\, L(\boldsymbol{x}) \Rightarrow R_{k_1}^{\sigma_1}(\varphi_1(\sigma \boldsymbol{x})) \,\&\, \ldots \,\&\, R_{k_t}^{\sigma_t}(\varphi_t(\sigma \boldsymbol{x})),$$

для состояний c, d выполняется равенство $f(c) = d$. Тогда имеет место отношение: $o_c, \forall \boldsymbol{x} f \vdash^c \&o_d$, где $\&o_d$ есть конъюнкция всех атомов описания o_d.

Доказательство. Если $o_c = o_d$, то доказательство элементарно.

Рассмотрим теперь случай, когда $o_c \neq o_d$. Так как $o_c \neq o_d$, то существуют в точности $m > 0$ таких подстановок $\sigma_1, \sigma_2, \ldots, \sigma_m$ из DET^c, что имеет место следование $o_c, \sigma_1 f, \sigma_2 f, \ldots, \sigma_m f \vdash^c o_d$. Это суть все подстановки, для которых левые части выражений $\sigma_1 f, \sigma_2 f, \ldots, \sigma_m f$ истинны в c, а некоторые литеры из их правых частей ложны. В соответствии с описанным выполнением действие $\sigma_i f$ изменяет описание o_c, добавляя нечто или устраняя, $i = 1, 2, \ldots, m$.

Но тогда, используя логическое следование $(A(a) \supset B) \supset \forall x A \supset B$, получаем искомое следование $o_c, \forall \boldsymbol{x} f \vdash^c o_d$.

Теорема доказана.

Следствие. *Для всякого состояния d задачи и ее исходного состояния выполняется отношение $o_{c_0}, Z \vdash^{c_0} \&o_d$.*

Таким образом, описания состояний задачи представляют собой логические следствия из описания исходного состояния и аксиоматики исчисления, которое определяется задачей. Отношение логического следования в этом случае рассматривается как следование из начального состояния.

Подводя итог, можно сказать, что всякая задача, с одной стороны, определяет развертку и поэтому можно говорить о вычислениях задачи, как о достижении некоторых состояний. С другой стороны, задача определяет логическое исчисление с семантикой, отличной от обычной. Ее характерной чертой является немонотонность: истинность формулы в одном состоянии не влечет истинности в другом. Как будет показано, введенные операционная и логическая семантики, в определенном смысле, эквивалентны.

3.3. Неподвижные точки задач. Исследуем свойства неподвижных точек. Последние играют важную роль, так как решения многих содержательных задач сводятся к нахождению неподвижных точек.

Справедливо следующее утверждение.

Теорема 21. *Пусть M_Z есть модель исчисления Z такая, что совокупность ее положительных атомов является описанием некоторого состояния c. Тогда c есть неподвижная точка задачи Z.*

Доказательство. Пусть f есть действие задачи Z. Так как $\boldsymbol{M}_Z$ есть модель для $\boldsymbol{Z}$ и одновременно описание состояния c, то для любых элементов $a_1, \ldots, a_n \in \boldsymbol{O}^n$ из того, что $\vdash^c \mathrm{DET}(a_1, \ldots, a_n)$ и $\vdash^c L(a_1, \ldots, a_n)$, следует $\vdash^c R_{k_1}^{\sigma_1}(\varphi_1(a_1, \ldots, a_n)) \& \ldots \& R_{k_t}^{\sigma_t}(\varphi_t(a_1, \ldots, a_n))$, т. е. $\vdash^c R_{k_i}(\varphi_i(a_1, \ldots, a_n))$, если $\sigma_i = 1$, и $\vdash^c \bar{R}_{k_i}(\varphi_i(a_1, \ldots, a_n))$, если $\sigma_i = 0$, $i = 1, 2, \ldots, t$.

В соответствии с определением выполнимости формулы в состоянии c, это означает, что $\varphi_i(a_1, \ldots, a_n) \in R_{k_i}^c$, если $\sigma_i = 1$, и $\varphi_i(a_1, \ldots, a_n) \notin R_{k_i}^c$, если $\sigma_i = 0$, $i = 1, 2, \ldots, t$, т. е. $f(c) = c$.

Теорема доказана.

Пример 25. Одна задача может обладать несколькими различными неподвижными точками. Действительно, рассмотрим задачу Z, действия которой суть следующие: $f_1 = R(a) \Rightarrow P(a) \,\&\, \bar{Q}(a)$ и $f_2 = Q(a) \Rightarrow P(b) \,\&\, \bar{R}(a)$, и начальное состояние c_0 обладает описанием $\{R(a), Q(a)\}$. Если

состояния c и d таковы, что $f_1(c_0)=c$ и $f_2(c_0)=d$, то $o_c=\{R(a), P(a)\}$ и $o_d=\{P(b), Q(a)\}$. Легко проверить, что действие f_2 не применимо к c и, если $f_1(c)=c_1$, то $o_c=o_{c_1}$; действие f_1 не применимо к d и, если $f_2(d)=d_1$, то $o_d=o_{d_1}$. Отсюда следует что c и d суть две различные неподвижные точки задачи. Проверкой убеждаемся, что $\{R(a), P(a)\}$ и $\{P(b), Q(a)\}$ суть модели исчисления $\boldsymbol{Z}$.

Пример 26. Не всякая модель задачи является ее описанием состояния, т. е. ее неподвижной точкой. Действительно, пусть задача задана своими начальным состоянием $\{P(a)\}$ и действием $P(a)\Rightarrow Q(a)$. Очевидно, что ее единственной неподвижной точкой будет $\{P(a), Q(a)\}$. Но формула $P(a)\supset Q(a)$ также обладает моделями $\{\bar{P}(a), Q(a)\}$ и $\{\bar{P}(a), \bar{Q}(a)\}$, ни одна из которых не представляет собой описание состояния этой задачи.

Замечание. В данном формализме отрицание некоторого события, представленного базисным отношением, при интерпретации действий представляется как отсутствие (вхождение отрицания слева) или исключение (вхождение отрицания справа) соответствующего кортежа термов. С другой стороны, отрицание используется в формулах исчисления $\mathcal{T}$ обычным образом.

Пример 27. Рассмотрим систему $\mathcal{S}$, которая может находиться в любом из конечного числа N состояний. Поставим в соответствие этим состояниям N вершин ориентированного графа, перенумеровав их от 1 до N. С течением времени система меняет свое состояние, выполняя определенные преобразования. В тех случаях, когда система находится в i-м состоянии, может быть выработано управляющее воздействие, в результате которого она переходит в определенное новое состояние.

Допустим, что каждая дуга, соединяющая узлы i и j, помечена величиной t_{ij}, которая обозначает время, необходимое для перехода по этой дуге системы из i-го состояния в j-е.

Обозначим u_i время перехода системы из i-го состояния в N-е (желаемое состояние) оптимальным способом, $i=1, 2, \ldots, N$. Принцип оптимальности Беллмана приводит к следующей системе нелинейных уравнений:

$$u_i=\min(t_{ij}+u_j),\ i=1, 2, \ldots, N-1,\ u_N=0.$$

Здесь минимум берется по всем парам (i, j) при $i\neq j$.

Для нахождения решения этой системы применим метод последовательных приближений. Для этого в качестве исходного приближения u_i^0 выберем решение, преобразующее систему непосредственно из состояния i в состояние N, $u_i^0=t_{iN}$, $i=1, 2, \ldots, N$. Если прямой переход из состояния i непосредственно в состояние N не существует, предполагаем, что затраченное время достаточно велико. Приближения более высокого порядка получаются обычным способом:

$$u_i^{k+1}=\min(t_{ij}+u_j^k),\ \ i=1, 2, \ldots, N-1,\quad u_N^{k+1}=0$$

при $k=0, 1, 3, \ldots$. Здесь минимум берется также по всем парам (i, j) при $i\neq j$.

Опишем систему $\mathcal{S}$ отношением $T(t, i, j)$, первый элемент которого обозначает время, необходимое, чтобы перевести систему из i-го состояния в j-е. Введем еще одно вспомогательное отношение $U(u, i)$, которое будет фиксировать вычисляемые приближения. Исходно оно содержит пары (u_i^0, i), где $u_i^0=t_{iN}$, $i=1, 2, \ldots, N$.

Пусть действие f_1 имеет вид:

$$T(t, i, j)\,\&\,U(u_j, j)\,\&\,U(u_i, i)\,\&\,u_i>t+u_j\Rightarrow U(t+u_j, i).$$

Нетрудно видеть, что оно задает описанную выше схему приближений. В соответствии с заданной операционной семантикой, после выполнения этого действия отношение U дополняется новыми парами $(t + u_j, i)$, если для существующих в нем пар (u_i, i) и (u_j, j) выполняется неравенство $u_i > t + u_j$. После того, как действие проработает, отношение U будет включать некоторое количество новых кортежей $(a_1, i), (a_2, i), \ldots, (a_n, i)$, где $N > 0$ и $i = 1, 2, \ldots, N - 1$, причем для них выполняется порядок $a_{j_1} \leqslant a_{j_2} \leqslant \ldots \leqslant a_{j_n}$. В соответствии со схемой приближения необходимо оставить лишь единственный кортеж, именно (a_{j_1}, i), чтобы сработало требование выделения минимального значения в схеме приближения. Очевидно, что это достигается в результате выполнения действия

$$f_2 = U(u_1, i) \,\&\, U(u_2, i) \,\&\, u_1 < u_2 \Rightarrow \bar{U}(u_2, i).$$

В результате для каждого значения $i = 1, 2, \ldots, N - 1$ останутся лишь те кортежи из множества $(a_1, i), (a_2, i), \ldots, (a_n, i)$, которые обладают наименьшим значением первого компонента.

Многократное применение действий f_1 и f_2 приводит к неподвижной точке, которая есть модель заданного исчисления.

Проиллюстрируем процесс сходимости к неподвижной точке на примере системы, граф которой задан отношением $T(t, i, j)$, представляющее собой набор троек:

$$\{(1, 2, 5), (1, 5, 18), (2, 3, 12), (2, 6, 14), (3, 4, 10), (3, 7, 3),$$
$$(4, 8, 14), (5, 6, 17), (5, 9, 7), (6, 7, 3), (6, 10, 6), (7, 8, 21),$$
$$(7, 11, 12), (8, 12, 9), (9, 10, 14), (10, 11, 17), (11, 12, 6)\}.$$

Начальные данные для вычислительной схемы заданы бинарным отношением U^0:

$$\{(1, 100), (2, 100), (3, 100), (4, 100), (5, 100), (6, 100),$$
$$(7, 100), (8, 9), (9, 100), (10, 100), (11, 6), (12, 0)\}.$$

После первого применения действий f_1 и f_2 отношение U^1 выглядит следующим образом:

$$\{(1, 100), (2, 100), (3, 100), (4, 23), (5, 100), (6, 100),$$
$$(7, 18), (8, 9), (9, 100), (10, 23), (11, 6), (12, 0)\}.$$

После второго их применения отношение U^2 приобретает вид:

$$\{(1, 100), (2, 100), (3, 21), (4, 23), (5, 100), (6, 21),$$
$$(7, 18), (8, 9), (9, 37), (10, 23), (11, 6), (12, 0)\}.$$

После третьего применения действий отношение U^3 выглядит следующим образом:

$$\{(1, 100), (2, 34), (3, 21), (4, 23), (5, 38), (6, 21), (7, 18),$$
$$(8, 9), (9, 37), (10, 23), (11, 6), (12, 0)\}.$$

После четвертого применения действий f_1 и f_2 отношение U^4 приобретает искомый вид:

$$\{(1, 39), (2, 34), (3, 21), (4, 23), (5, 38), (6, 21),$$
$$(7, 18), (8, 9), (9, 37), (10, 23), (11, 6), (12, 0)\}.$$

Это отношение для каждого узла исходного графа задает вес оптимального пути, соединяющего его с узлом N. Оно представляет собой неподвижную точку отношения U, которое вместе с отношением T есть модель для исчисления, задаваемого задачей.

Решение можно сделать более информативным, если вместо двухместного предиката U использовать трехместный $U(u, i, j)$, причем интерпретация первых двух аргументов остается прежней, а третий аргумент j интерпретируется как узел, к которому необходимо перейти из узла i, чтобы получить путь из i в N с весом u. В этом случае действия приобретают следующий вид:

$$f_1 = T(t, i, j) \,\&\, U(u_j, j, s) \,\&\, U(u_i, i, k) \,\&\, u_i > t + u_j \Rightarrow U(t + u_j, i, j);$$

$$f_2 = U(u_1, i, j) \,\&\, U(u_2, i, k) \,\&\, u_1 < u_2 \Rightarrow \bar{U}(u_2, i, k).$$

В результате неподвижная точка задачи в отношении U для каждого узла $i = 1, 2, \ldots, N-1$ содержит не только вес оптимального пути, но и ту дугу, по которой надо перемещаться из i, чтобы этот вес был достижим. Для рассмотренной выше задачи, оптимальный путь выглядит следующим образом: 1, 2, 3, 7, 11, 12.

3.4. Исследование типов задач. Введенная операционная семантика определяет результат вычислений в зависимости от порядка применений действий. В общем случае применения не коммутативны, в силу наличия отрицаний перед базисными предикатами. Поэтому представляет интерес исследование задач в зависимости от наличия отрицаний в их действиях.

Введем следующие определения.

Траектория называется *монотонной*, если для любых ее состояний c и d, таких что $c < d$ имеет место включение $o_c \subseteq o_d$. *Неподвижная точка*, располагающаяся в конце монотонной траектории, также называется *монотонной*. Наконец, *задача монотонна*, если все траектории ее развертки монотонные.

Замечание. Из монотонности задачи не вытекает, что применимость действия f в состоянии c обуславливает его применимость в состоянии d таком, что $c < d$. Действительно, пусть в описание o_c не входит отношение $R(a)$, а в o_d — входит и в левой части действия f содержится отрицательная литера $\bar{R}(x)$. Тогда f может быть применимо в c, но не применимо в d.

Будем говорить, что задача *обладает единственной неподвижной точкой*, если все ее неподвижные точки имеют одинаковые описания.

Нетрудно доказать следующее утверждение.

Теорема 22. *Пусть Z есть задача, в действиях которой отрицание не встречается перед базисными предикатными символами. Тогда она монотонная и обладает единственной неподвижной точкой.*

Доказательство. Задача Z монотонная, так как всякое ее действие f, примененное к произвольному состоянию c, не производит вычеркиваний из описания o_c. Таким образом, если $f(c) = d$, то $o_c \subseteq o_d$.

Допустим теперь, что в развертке R_Z имеются две различные неподвижные точки c и d, такие что $c \neq d$, и t_1, t_2 — траектории, соединяющие корень развертки с узлами соответственно c и d. Применим последовательность t_2 действий к состоянию c, что возможно, так как $o_{c_0} \subseteq o_c$. В силу несовпадения описаний o_c и o_d, получим новое состояние e такое, что $o_c \subset o_e$. Получили противоречие, так как c есть неподвижная точка.

Теорема доказана.

Обозначим L^+ язык действий, в которых отсутствуют отрицания перед базисными предикатами. Любое конечное подмножество формул этого языка задает класс задач, которые характеризуются различными начальными

и финальными состояниями и, соответственно, описаниями состояний. Как показано, всякая такая задача монотонна и обладает единственной неподвижной точкой.

Ослабим ограничения на вид действий, разрешив использование отрицаний в их левых частях. Будем полагать, что все такие действия образуют язык L_0^-. Справедливо следующее свойство этого языка.

Теорема 23. *Если все действия задачи принадлежат языку* L_0^-, *то она монотонная.*

Доказательство легко вытекает из того, что всякое действие задачи не вычеркивает элементов отношений из описаний состояний.

Пример 28. Задачи, действия которых принадлежат языку L_0^-, могут не обладать единственной неподвижной точкой. Действительно, рассмотрим задачу со следующими действиями: $f_1 = P(a) \,\&\, \bar{R}(a) \Rightarrow Q(a)$; $f_2 = P(a) \,\&\, \bar{Q}(a) \Rightarrow R(a)$, и начальным состоянием $\{P(a)\}$. Ее развертка, помимо исходного состояния c_0, содержит еще два состояния c и d, $f_1(c_0) = d$, $f_2(c_0) = c$, где $o_c = \{P(a), R(a)\}$, $o_d = \{P(a), Q(a)\}$, которые, как легко проверить, суть неподвижные точки.

Интерпретация множественности неподвижных точек достаточно очевидна — различные неподвижные точки представляют собой различные решения, которые могут удовлетворять различным требованиям к финальным состояниям.

Расширим язык L^+ до языка L_1^-, включающего действия, в которых вхождения отрицаний разрешены лишь в правых частях. Этот язык не обладает ни свойством монотонности, ни свойством уникальности неподвижной точки (см. пример 25 из предыдущего раздела). Однако он обладает одним интересным свойством, которое позволяет говорить о сводимости некоторых задач из L_1^- к задачам из L_0^-. Чтобы показать это, введем ряд обозначений и определений.

Пусть действие f задачи Z имеет вид $\mathrm{DET}(\boldsymbol{x}) \,\&\, L(\boldsymbol{x}) \Rightarrow \bar{R}(\varphi_1(\boldsymbol{x})) \,\&\, A$, где R есть n-арный предикат. Введем новый n-арный предикатный символ R^*, и заменим f на действие $f^* = \mathrm{DET}(\boldsymbol{x}) \& L(\boldsymbol{x}) \Rightarrow R^*(\varphi_1(\boldsymbol{x})) \& A$.

Произведем подобные преобразования со всеми действиями задачи Z, в правых частях которых встречается литера $\bar{R}(\boldsymbol{t})$ для некоторого кортежа $\boldsymbol{t}$ термов. Предикатный символ R^* назовем *ассоциированным с* R. Таким образом, после указанного преобразования число базисных предикатных символов увеличивается на единицу и одновременно уменьшается на единицу число отрицательных предикатов в правых частях действий.

Преобразуем все действия задачи введением ассоциированных предикатных символов, чтобы в новых действиях отрицательные литеры справа не встречались. Допустим, что в результате преобразований базисная сигнатура пополнилась за счет новых предикатных символов $Q_1, Q_2, \ldots, Q_n$, и Q_i ассоциирован с R_i, $i = 1, 2, \ldots, n$.

Рассмотрим произвольное действие g, в котором базисный символ R встречается слева в определяющей части детерминатива, и с ним ассоциирован символ R^*, т. е. $g = R(\psi(\boldsymbol{x})) \& \mathrm{DET}(\boldsymbol{x}) \& L_1(\boldsymbol{x}) \Rightarrow B$.

Заменим действие g на

$$g^* = R(\psi(\boldsymbol{x})) \,\&\, \bar{R}^*(\psi(\boldsymbol{x})) \,\&\, \mathrm{DET}(\boldsymbol{x}) \,\&\, L_1(\boldsymbol{x}) \Rightarrow B.$$

Если базисный предикат R встречается в уточняющей части действия g, то воспользовавшись эквивалентностью $\forall y(A \,\&\, B) = \forall y A \,\&\, \forall y B$, переходим к действию

$$g = \forall \boldsymbol{y} R(\psi(\boldsymbol{x}, \boldsymbol{y})) \,\&\, \mathrm{DET}(\boldsymbol{x}) \,\&\, L_1(\boldsymbol{x}) \Rightarrow B.$$

Теперь действие g^* приобретает вид

$$\forall y(R(\psi(x, y)) \,\&\, \bar{R}^*(\psi(x, y))) \,\&\, \mathrm{DET}(x) \,\&\, L_1(x) \Rightarrow B.$$

Произведем подобные преобразования со всеми действиями, в которых слева встречаются предикаты, для которых определены ассоциированные предикаты.

Обозначим результирующую задачу Z^* и будем говорить, что исходная задача преобразована в Z^* за счет введения предикатных символов $Q_1, Q_2, \ldots, Q_n$, ассоциированных соответственно с $R_1, R_2, \ldots, R_n$.

Задачи Z и Z^* связаны определенным образом. В частности, имеет место следующая теорема.

Теорема 24. *Всякая модель M_{Z^*} исчисления Z^* превращается в модель M_Z исчисления Z вычеркиванием из каждого отношения R_i ассоциированного отношения Q_i, $i = 1, 2, \ldots, n$.*

Доказательство. Рассмотрим, какими логическими значениями обладают действия задач Z^* и Z при одном и том же кортеже a аргументов. Приведем следующие рассуждения для каждой пары R_i и Q_i предикатных символов, $i = 1, 2, \ldots, n$.

Пусть отношение R принадлежит модели M_{Z^*}, отношение R^* в этой модели интерпретирует предикат R^* и отношение $\mathcal{R} = R - R^*$ интерпретирует предикат R в M_Z. Возможны следующие случаи.

Имеет место включение $a \in R - R^*$. Это значит, что в модели M_{Z^*} значения $R(a)$ есть *истина*, $R^*(a)$ — *ложь* и $R(a) \,\&\, \bar{R}^*(a)$ — *истина*. Но тогда $\mathcal{R}(a) =$ *истина*, $\bar{\mathcal{R}}(a) =$ *ложь*. Поэтому логические значения формул f и f^*, а также g и g^* совпадают, так как $R^*(a) = \bar{\mathcal{R}}(a) =$ *ложь* и $R(a) \,\&\, \bar{R}^*(a) = \mathcal{R}(a) =$ *истина*. Таким образом, в этом случае для аргументов a формулы f и f^*, а также g и g^* принимают одинаковые логические значения.

Случаи когда $a \in R \cap R^*$ или $a \in R^* - R$ разбираются подобно.

Случай, когда $a \notin R \cup R^*$ несколько отличается от трех предыдущих, так как непосредственно нельзя заключить, что для рассматриваемого набора a логические значения соответствующих действий задач Z^* и Z совпадают. Для того, чтобы убедиться в верности утверждения, приведем следующие рассуждения.

В модели M_{Z^*} имеют место равенства: $R(a) = R^*(a) =$ *ложь*, $R(a) \,\&\, \bar{R}^*(a) =$ *ложь*. Но тогда $\mathcal{R}(a) =$ *ложь*, $\bar{\mathcal{R}}(a) =$ *истина*. Отношение $\bar{\mathcal{R}}(a)$ встречается лишь в заключении действий на месте отношения $R^*(a)$ (= *ложь*). В данном случае для набора a произвольное действие задачи Z^*, в заключение которого входит предикат R^*, принимают значение *истина*, что возможно лишь в случае, когда посылка действия ложна. Но посылка этого действия будет ложна и тогда, когда мы перейдем к соответствующему действию задачи Z, в силу того, что логические значения $R(a) \,\&\, \bar{R}^*(a)$ и $\mathcal{R}(a)$ есть *ложь*. Следовательно, и в этом случае действия f и f^* принимают значение *истина*.

Что касается действий g и g^*, то их логические значения также совпадают в силу равенств $R(a) \,\&\, \bar{R}^*(a) = \mathcal{R}(a) =$ *ложь*.

Доказательство достаточно прозрачно, когда базисный предикат R встречается в определяющей части действия. Когда же он входит в уточняющую часть в виде $\forall y(R(\psi(x, y)))$, то доказательство также справедливо, так как мы рассматриваем значения предикатов $R(a)$ и $\bar{R}^*(a)$ вне зависимости от того, на место каких переменных, свободных или связанных подставлены элементы кортежа a.

Разобраны все случаи. Теорема доказана.

Таким образом, всякая модель исчисления Z^* после указанных преобразований превращается в модель исчисления Z, а так как, всякая непо-

движная точка задачи есть модель соответствующего исчисления, то имеет место следующее следствие.

Следствие. *Всякая неподвижная точка задачи Z^* преобразуется в неподвижную точку задачи Z в результате вычеркивания из всякого отношения R_i ассоциированного отношения Q_i, $i=1, 2, \ldots, n$.*

Обратное утверждение справедливо лишь для частного случая неподвижных точек задачи Z.

Назовем *траекторию* задачи *квазимонотонной*, если она либо монотонна, либо для всякого базисного отношения верно, что любой кортеж не может появиться вновь как элемент этого отношения после вычеркивания его из этого отношения на начальном участке траектории.

Неподвижная точка задачи называется *квазимонотонной*, если она расположена в конце квазимонотонной траектории.

Теорема 25. *Описание всякой квазимонотонной неподвижной точки задачи Z может быть получено из описания некоторой неподвижной точки задачи Z^* путем вычеркивания из всякого отношения R_i ассоциированного отношения Q_i, $i=1, 2, \ldots, n$.*

Доказательство. Построим по квазимонотонной траектории t_Z задачи Z соответствующую траекторию t_{Z^*} задачи Z^*. Для всякого действия f, у которого в заключении имеется отрицательная литера, соответствующим действием служит f^*, аналогично для действия g соответствующим служит g^*. В случае, если действие не менялось, то соответствующее действие совпадает с исходным.

Применение действия f^* влечет лишь пополнение ассоциированного отношении. Поэтому отношение R^* содержит все кортежи отношения R исходной задачи, которые в ней элиминировались в результате применения f.

Применение действия g^* в состоянии c возможно лишь в случае, когда для кортежа $\boldsymbol{a}$ имеют место отношения $\boldsymbol{a} \in R^c$, $\boldsymbol{a} \notin R^{*c}$, т. е. $\boldsymbol{a} \in R^c - R^{*c}$. Но это соответствует тому, что действие g применимо, если кортеж $\boldsymbol{a}$ содержится в отношении R^c задачи Z, т. е. не был вычеркнут ранее.

Если действие g^* получается из g преобразованием в уточняющей части, то применение g^* возможно для кортежа $\boldsymbol{a}$ в случае, когда для всевозможных кортежей $\boldsymbol{b}$, подставленных на место связанных переменных, имеют место отношения $\psi(\boldsymbol{a}, \boldsymbol{b}) \in R^c$, $\psi(\boldsymbol{a}, \boldsymbol{b}) \notin R^{*c}$, т. е. $\psi(\boldsymbol{a}, \boldsymbol{b}) \in R^c - R*c$. Но это соответствует тому, что действие g применимо, если кортеж $\psi(\boldsymbol{a}, \boldsymbol{b})$ содержится в отношении R^c задачи Z, т. е. не был вычеркнут ранее.

Отсюда вытекает, что описание o_c состояния c задачи Z отличаются от описания o_c^* задачи Z^* тем, что отношение $R^c \in o_c$ равно $R^c - R^{*c}$, где $R^c, R^{*c} \in o_c^*$.

Полагаем, что соответствующие узлы траектории t_Z и построенной по ней траектории t_{Z^*} поименованы одинаково, хотя описания этих состояний могут различаться. Рассмотрим произвольное состояние $c \in t_Z$. В силу квазимонотонности исходной траектории, для всякой пары ассоциированных отношений R^c, R^{*c} задачи Z^*, и отношения $\mathscr{R}^c$, интерпретирующего в задаче Z отношение R, выполняется равенство $\mathscr{R}^c = R^c - R^{*c}$. Для базисных предикатов задачи Z, не обладающих ассоциированными предикатами, выполняется простое равенство для состояний t_Z и соответствующих состояний t_{Z^*}.

Таким образом, если c есть квазимонотонная неподвижная точка задачи Z, то в t_{Z^*} ей соответствует состояние, описание которого связано с описанием o_c, как указано выше.

Допустим, что это состояние не есть неподвижная точка, т. е. ниже него возможно появление состояния, описание которого отлично от o_c. Это может быть лишь в случае применения действий типа f^* или g^*. Но тогда число таких применений конечно, так как ассоциированные предикаты

входят в левые части действий лишь отрицательно. Поэтому через конечное число шагов будет достигнута неподвижная точка c^*, которая связана с состоянием c указанным выше образом.

Теорема доказана.

Итак, всякая задача, описанная на языке L_1^-, может быть переформулирована в виде задачи на языке L_0^-, так, что неподвижные точки последней преобразуются в неподвижные точки первой. И таким образом могут быть получены все квазимонотонные неподвижные точки исходной задачи. Иными словами, язык L_1^- определенным образом сводится к L_0^-. Такое сведение вполне осмысленно, исходя из практических соображений, так как язык L_0^- обладает свойством монотонности, которое используется при восстановлении траектории, ведущей к найденному решению.

Пример 29. Рассмотрим действия $f_1 = P(a) \Rightarrow \bar{R}(a)$ и $f_2 = R(a) \Rightarrow Q(a)$ и задачи Z_1 и Z_2, каждая из которых обладает этими действиями, но различаются начальными состояниями. Задача Z_1 имеет начальное состояние $\{P(a)\}$, а Z_2 — начальное состояние $\{P(a), R(a)\}$. Тогда задача Z_1 обладает неподвижной точкой $\{P(a)\}$, а Z_2 — неподвижными точками $\{P(a)\}$ и $\{P(a), Q(a)\}$.

Введем ассоциированный предикат R^* и заменим f_1 на $g_1 = P(a) \Rightarrow R^*(a)$, а f_2 на $g_2 = \bar{R}^*(a) \& R(a) \Rightarrow Q(a)$. Рассмотрим развертки задач Z_1^* и Z_2^*, которые получаются присоединением начальных состояний соответственно $\{P(a)\}$ и $\{P(a), R(a)\}$. Неподвижные точки для этих задач суть $\{P(a), R^*(a)\}$ и $\{P(a), R(a), R^*(a), Q(a)\}$. Теперь неподвижные точки для исходных задач получаются, если интерпретировать предикат R отношением $R(a) - R^*(a)$.

Пример 30. Покажем, что для немонотонных траекторий сведение, описанное в теореме, не имеет места. Действительно, пусть задача обладает следующими действиями: $f = P \Rightarrow \bar{R}$, $g = S \Rightarrow R$, $h = R \Rightarrow Q$. После введения ассоциированного предиката R^* и соответствующего преобразования действий получим задачу: $f^* = P \Rightarrow R^*$, $g^* = S \Rightarrow R$, $h^* = R \& \bar{R}^* \Rightarrow Q$.

Рассмотрим траекторию $f \to g \to h$ первой задачи из исходного состояния $PQRS = 1011$. Заключительным состоянием этой траектории будет 1111. Если теперь преобразовать эту траекторию заменой действий, как описано в теореме, то действие h^* не применяется и состояние 1111 получено не будет.

Наконец, самым общий язык L_2^- описания действий задач не имеет ограничений на вхождения отрицаний в действиях. Этот язык также сводится к L_0^-, если рассматривать только квазимонотонные траектории, хотя сведение носит более громоздкий характер.

Пусть действие f задачи Z выглядит следующим образом: $\mathrm{DET}(\boldsymbol{x}) \& \& L(\boldsymbol{x}) \Rightarrow \bar{R}(\varphi_1(\boldsymbol{x})) \& A$, где R есть базовый n-местный предикатный символ. Введем новый n-местный предикатный символ R^*, ассоциированный с R, и заменим f на действие $f^* = \mathrm{DET}(\boldsymbol{x}) \& L(\boldsymbol{x}) \Rightarrow R^*(\varphi_1(\boldsymbol{x})) \& A$. Произведем подобные преобразования для всех действий, в правых частях которых встречается отрицательный предикат R.

Преобразуем все действия исходной задачи путем введения ассоциированных предикатных символов таким образом, что в новых действиях не будут входить отрицательные предикатные символы справа. Допустим, что в результате преобразований список базовых предикатных символов пополнился за счет ассоциированных — $Q_1, Q_2, \ldots, Q_n$, где предикатный символ Q_i ассоциирован с R_i, $i = 1, 2, \ldots, n$.

Рассмотрим действие g, в котором базовый предикат R встречается в определяющей части и с ним ассоциирован предикат R^*, т. е. $g = R(\psi(\boldsymbol{x})) \&$

$\& \mathrm{DET}(x) \& L_1(x) \Rightarrow B$. Заменим действие g на $g^* = R(\psi(x)) \& \bar{R}^*(\psi(x)) \& \& \mathrm{DET}(x) \& L_1(x) \Rightarrow B$.

Если базисный предикат R встречается положительно в уточняющей части действия g, то переходим к действию

$$g^* = \forall y(R(\psi(x, y)) \& \bar{R}^*(\psi(x, x))) \& \mathrm{DET}(x) \& L_1(x) \Rightarrow B.$$

Произведем подобные преобразования со всеми действиями, в которых слева в определяющих частях встречаются предикатные символы, для которых введены ассоциированные предикаты.

Рассмотрим теперь действие h, в котором предикат R встречается слева отрицательно в уточняющей части, т. е.

$$h = \forall y \bar{R}(\zeta(x, y)) \& \mathrm{DET}(x) \& L_2(x) \Rightarrow C.$$

Введем новый предикатный символ R^{**}, который по аналогии с R^* назовем (вторым) ассоциированным с R. Заменим действие h на

$$h^* = \forall y R^{**}(\zeta(x, y)) \& \mathrm{DET}(x) \& L_2(x) \Rightarrow C.$$

Отметим, что в полученных действиях предикат R^{**} не встречается справа. Дополнительно введем три действия:

$$h_1 = \bar{R}(z) \Rightarrow R^{**}(z); \quad h_2 = R^*(z) \Rightarrow R^{**}(z); \quad h_3 = R^{**}(z) \& R(z) \Rightarrow R^*(z).$$

Теперь, если интерпретировать символ $\Rightarrow$ как импликацию, то конъюнкция $h_1 \& h_2 \& h_3$ эквивалентна тому, что в любой модели с носителем O формулы $R^{**}(z)$ и $\bar{R}(z) \vee R^*(z)$ принимают одно и то же значение. При этом $h_1 \& h_2$ задает для отношения R^{**} ограничение снизу (объединение $\bar{R} \cup R^*$ включено в R^{**}), а h_3 — ограничение сверху (R^{**} включено в объединение $\bar{R} \cup R^*$).

Произведем подобные преобразования со всеми действиями, в которых слева в уточняющих частях встречаются отрицательные предикаты, для которых введены ассоциированные. Обозначим Z^* результирующую задачу, полученную введением новых предикатных символов $Q_1, Q_2, \ldots, Q_n, H_1, H_2, \ldots, H_n$, ассоциированных соответственно с $R_1, R_2, \ldots, R_n$, и с начальным состоянием, которое обладает дополнительно пустыми отношениями, интерпретирующими эти предикаты.

Задачи Z и Z^* связаны определенным образом. В частности, имеет место следующее утверждение.

Теорема 26. *Всякая модель M_Z. исчисления Z^* превращается в модель M_Z исчисления Z за счет вычеркивания из отношения R_i ассоциированного отношения Q_i, $i = 1, 2, \ldots, n$.*

Всякая неподвижная точка задачи есть ее модель, поэтому можно утверждать, что имеет место следствие.

Следствие. *Всякая неподвижная точка задачи Z^* преобразуется в неподвижную точку задачи Z при интерпретации каждого предиката R_i отношением $R_i - Q_i$, где Q_i есть первое ассоциированное с R_i отношение, $i = 1, 2, \ldots, n$.*

Справедливо и, в определенном смысле, обратное утверждение.

Теорема 27. *Описание всякой квазимонотонной неподвижной точки задачи Z может быть получено из описания некоторой неподвижной точки задачи Z^* путем вычеркивания из отношений R_i элементов ассоциированных отношений Q_i, $i = 1, 2, \ldots, n$.*

Таким образом, язык L_2^- также сводится к языку L_0^-. С другой стороны, не монотонные траектории задач, формулируемых в L_2^-, могут не преобразовываться в траектории задач в языке L_0^-.

Пример 31. Рассмотрим задачи Z_1 и Z_2, каждая из которых обладает действиями $f_1 = P(a) \Rightarrow \bar{R}(a)$, $f_2 = R(a) \Rightarrow Q(a)$ и $f_3 = \bar{R}(a) \Rightarrow S(a)$, но различаются начальными состояниями, Z_1 имеет начальное состояние $\{P(a)\}$ и $Z_2 - \{P(a), R(a)\}$. Множества $\{P(a), S(a)\}$ и $\{P(a), Q(a), S(a)\}$ суть модели формулы $f_1 \& f_2 \& f_3$.

Введем новый предикатный символ R^* и заменим действия f_1 на $g_1 = P(a) \Rightarrow R^*(a)$, f_2 на $g_2 = \bar{R}^*(a) \& R(a) \Rightarrow Q(a)$, f_3 на $g_3 = R^{**}(a) \Rightarrow S(a)$ и добавим еще два действия $h_1 = \bar{R}(a) \Rightarrow R^{**}(a)$; $h_2 = R^*(a) \Rightarrow R^{**}(a)$.

Построим развертки задач Z_1^* и Z_2^*, которые получаются присоединением начальных состояний соответственно $\{P(a)\}$ и $\{P(a), R(a)\}$. Неподвижные точки для них суть $\{P(a), R^*(a), R^{**}(a), S(a)\}$ и $\{P(a), R(a), R^*(a), Q(a)\}$. Заменив пару $R(a), R^*(a)$ на $R(a) - R^*(a)$ и исключив $R^{**}(a)$, получим, что порожденные интерпретации совпадают с соответствующими моделями для формулы $f_1 \& f_2 \& f_3$.

Описанное выше сведение имеет практическое значение. Действительно, зачастую при решении даже простых задач объяснить решение бывает не проще, чем его найти. Нарушение монотонности приводит к тому, что часть информации из описаний состояний удаляется в ходе поиска решения. Но такая информация может быть важной для восстановления хода рассуждений, который приводит к найденному решению.

Представьте, что при решении геометрической задачи вам удалось доказать равенство двух фигур, но некоторые шаги в доказательстве были пропущены. Если это не существенные шаги, то восстановить доказательство просто, но если они достаточно оригинальные, то восстановление всего доказательства оказывается ничуть не проще, чем его построение с самого начала. В точности так же, в вычислительных задачах, мы не можем выпускать промежуточные шаги из опасения потерять важную информацию. Используя сводимость задач к монотонным, мы избегаем такой потери, одновременно не теряя решений.

3.5. Задачи с эквивалентностями. В этом разделе мы покажем, как расширить определение задачи, включив в него правило подстановки равного вместо равного, и как следствие, осуществить переход к построению нормальных моделей. Тем самым выразительные возможности языка задач расширяются, так как появляется возможность рассуждать не о конкретных объектах той или иной области, а о целых классах.

Пусть базисная сигнатура задачи Z включает двухместный предикатный символ E, понимаемый как эквивалентность, т. е. для него выполняются законы рефлексивности, коммутативности и транзитивности. Поэтому задача Z включает действия ref: $\Rightarrow E(x, x)$; com: $E(x, y) \Rightarrow E(y, x)$; tr: $E(x, y) \& E(y, z) \Rightarrow E(x, z)$.

Это влечет, разбиение множества $\boldsymbol{O}$ объектов в каждом состоянии c на классы эквивалентности. Описание o_c представимо в так называемом *нормальном виде* e_c, когда к одному классу отнесены все попарно эквивалентные элементы. Очевидно, что для различных состояний такие разбиения могут различаться. Каждый класс эквивалентности с представителем a обозначим $[a]$. В общем случае пусть $[(a_1, \ldots, a_n)]$ обозначает кортеж $([a_1], \ldots, [a_n])$. В векторной записи это выглядит так: если $\boldsymbol{a} = (a_1, \ldots, a_n)$, то $[\boldsymbol{a}] = ([a_1], \ldots, [a_n])$.

Пусть $\boldsymbol{x} = (x_1, \ldots, x_n)$, $\boldsymbol{a} = (a_1, \ldots, a_n)$. Определим подстановку $\{\boldsymbol{x} \leftarrow [\boldsymbol{a}]\}$ как множество $\{\sigma\colon \sigma = \{x_1 \leftarrow b_1, \ldots, x_n \leftarrow b_n\}, (b_1, \ldots, b_n) \in [a_1] \times \ldots \times [a_n]\}$ подстановок. Тем самым, $\{\boldsymbol{x} \leftarrow [\boldsymbol{a}]\}$ обозначает все подстановки $\{\boldsymbol{x} \leftarrow \boldsymbol{b}\}$ для всякого вектора $\boldsymbol{b} \in [a_1] \times \ldots \times [a_n]$.

Расширим множество действий исходной задачи за счет действий:

$$\text{subst}_P: \bigcap_{i=1,n} E(x_i, y_i) \,\&\, R(x_1, x_2, \ldots, x_n) \Rightarrow R(y_1, y_2, \ldots, y_n);$$

$$\text{subst}_\varphi: \bigcap_{i=1,n} E(x_i, y_i) \Rightarrow E(\varphi(x_1, x_2, \ldots, x_n), \varphi(y_1, y_2, \ldots, y_n))$$

для всех предикатных и функциональных символов, сохраняющих эквивалентность E.

Естественно, что предикат E может встречаться в других действиях задачи как справа, так и слева. Для наших целей наложим ограничение на вид используемых действий. Потребуем, чтобы предикат E не встречался в их правых частях отрицательно. Тем самым, действия являются монотонными относительно эквивалентности в том смысле, что применение любого из них может лишь увеличить размеры классов эквивалентности.

В связи с введенными действиями уместно допустить, что описание e_c состояния c всегда является неподвижной точкой для каждого из действий ref, com, tr. Это достигается применением указанных действий до тех пор, пока для них не будет достигнута неподвижная точка. В итоге мы переходим от обычного представления состояний, когда классы эквивалентности не выделены, к их явному описанию.

Возникает вопрос: в каком случае можно ограничиться рассмотрением не описания o_c, а его нормальной интерпретации, когда элементы одного класса не разделяются? И всегда ли в этом случае операционная семантика будет совпадать с логической? Чтобы ответить на эти вопросы, уточним ряд определений.

В частности, несколько изменяется понятие ситуации, определяемой детерминативом $\text{DET}(\boldsymbol{x})$ в состоянии c. А именно, *нормальной ситуацией*, определяемой детерминативом $\text{DET}(\boldsymbol{x})$ в состоянии c, называется множество $\text{DET}^{nc} = \{\sigma: \{\boldsymbol{x} \leftarrow [\boldsymbol{a}]\}$, где $\boldsymbol{x} = (x_1, \ldots, x_n)$, $\boldsymbol{a} = (a_1, \ldots, a_n)$, $a_1, \ldots, a_n \in \boldsymbol{O}$, таких, что

если в определяющую часть $\text{DET}(\boldsymbol{x})$ входит атом $R(\psi(x))$, то $\sigma\psi(\boldsymbol{x}) \in R^c$;

если в уточняющую часть входит атом $R(\zeta(z))$, то $\sigma\theta\zeta(z) \in R^c$ при любой подстановке θ из $\boldsymbol{O}$ на места не определенных переменных;

если в уточняющую часть входит литера $\bar{R}(\zeta(z))$, то $\sigma\theta\zeta(z) \notin R^c$ при любой подстановке θ из $\boldsymbol{O}$ на места не определенных переменных $\}$.

Из определения нормальной ситуации вытекает, что $\text{DET}^{nc} = \text{DET}^c$.

Действительно, если $\{\boldsymbol{x} \leftarrow [\boldsymbol{a}]\} \subseteq \text{DET}^{nc}$, где $\boldsymbol{x} = (x_1, \ldots, x_n)$, $\boldsymbol{a} = (a_1, \ldots, a_n)$, то в DET^c присутствуют все подстановки $\{\boldsymbol{x} \leftarrow [\boldsymbol{b}]\}$, где $\boldsymbol{b} \in [a_1] \times \ldots \times [a_n]$, так как состояние c является неподвижной точкой для действий ref, com, tr.

С другой стороны, легко показать, что всякая подстановка $\{\boldsymbol{x} \leftarrow \boldsymbol{a}\}$ из DET^c присутствует в DET^{nc}.

Таким образом отличие ситуаций DET^c и DET^{nc} заключается лишь в том, что в DET^{nc} подстановки перечисляются с учетом имеющихся классов эквивалентности, а в DET^c те же самые подстановки перечислены, без учета разбиения на классы эквивалентности.

Определим так называемое *нормальное применение* $f^n(c)$ действия f в состоянии c. Обычное применение действия отличается от нормального тем, что в последнем мы имеем дело не с отдельными кортежами базисных отношений, которые представляют собой составные части описаний состояний, а с множествами кортежей, представляющих собой элементы декартовых произведений классов эквивалентности.

Пусть действие f выглядит следующим образом:

$$\text{DET}(\boldsymbol{x}) \,\&\, L(\boldsymbol{x}) \Rightarrow R_{k_1}^{\sigma_1}(\varphi_1(\boldsymbol{x})) \,\&\, \ldots \,\&\, R_{k_t}^{\sigma_t}(\varphi_t(\boldsymbol{x})),$$

где $L(\boldsymbol{x})$ — формула вспомогательной сигнатуры, $\varphi_1, \ldots, \varphi_t$ — наборы термов и $\sigma_1, \ldots, \sigma_t \in \{0, 1\}$.

Нормальное применение этого действия в состоянии c описывается следующим образом. Пусть $f^n(c) = d$. Тогда отличные от E базисные отношения в состоянии d выглядят так: $\{R_1^c + \Delta R_1^c, \ldots, R_s^c + \Delta R_s^c\}$, где $\Delta R_1^c, \ldots, \Delta R_s^c$ суть изменения отношений соответственно $R_1^c, \ldots, R_s^c$, зависящие от показателей соответственно $\sigma_1, \ldots, \sigma_t$; $R_i^c + \Delta R_i^c$ обозначает, что фрагмент ΔR_i^c добавляется к R_i^c, если атом R_i входит в f положительно, и вычеркивается из R_i^c, если входит отрицательно, $i = k_1, \ldots, k_t$.

Всякое изменение $\Delta R_{k_i}^c$ содержит в точности все те кортежи, которые удовлетворяют следующим условиям. Если для каждой подстановки σ из $\{\boldsymbol{x} \leftarrow [\boldsymbol{a}]\} \subseteq \mathrm{DET}^{nc}$ имеет место отношение $\vdash^c L(\sigma\boldsymbol{x})$, то $\Delta R_{k_i}^c$ включает все кортежи из $\varphi_i(\sigma\boldsymbol{x})$, $i = 1, 2, \ldots, t$.

Если же в заключение действия входит атом $E(\varphi_1(\boldsymbol{x}), \varphi_2(\boldsymbol{x}))$, то для каждой подстановки σ из $\{\boldsymbol{x} \leftarrow [\boldsymbol{a}]\} \subseteq \mathrm{DET}^{nc}$ такой, что $\vdash^c L(\sigma\boldsymbol{x})$, объединяются классы эквивалентности, включающие термы $\varphi_1(\sigma\boldsymbol{x})$ и $\varphi_2(\sigma\boldsymbol{x})$.

Таким образом, для всякого состояния и любого действия результат его нормального применения определяется однозначно. Как видно из определения, нормальное применение существенно отличается от обычного применения всякого действия.

Замечание. Если задача не удовлетворяет сформулированным выше ограничениям и в правой части действия встречается отрицательная литера $\bar{E}$, то возможно, что в описании некоторого состояния встречается класс эквивалентности, например $\{a, b, c\}$, и в результате применения действия генерируется изменение $\bar{E}(a, b)$. Но тогда необходимо каким-то образом трансформировать класс $\{a, b, c\}$, что может оказаться невозможно сделать, так как $E(b, c)$ и $E(a, c)$ влечет $E(a, b)$. Возникает противоречие, которое невозможно устранить, если ограничится лишь нормальными применениями.

Обозначим Z^n задачу, которая получается из Z отбрасыванием действий ref, com, tr и использованием лишь нормальных вычислений. Теперь легко заметить, что если t^n — траектория из Z^n такая, что $t^n(c_0) = d$, то в Z существует траектория t такая, что $t(c_0) = d$. Но это не дает возможности утверждать, что если состояние c есть неподвижная точка задачи Z^n, то c является неподвижной точкой задачи Z.

С другой стороны, если состояние c есть неподвижная точка задачи Z, то c есть неподвижная точка задачи Z^n.

Действительно, всякое нормальное применение представимо в виде некоторой траектории, состоящей лишь из действий задачи Z. Поэтому, если бы состояние c не было неподвижной точкой для Z^n, то можно было бы построить траекторию в Z, которая изменяла бы это состояние.

Покажем теперь, что исчисление $\boldsymbol{Z}$ обладает так называемыми *нормальными* моделями, элементами которых являются классы эквивалентности. И эти нормальные модели порождаются задачей Z^n.

Определим *нормальную* интерпретацию базисных формул.

Пусть R есть n-местный базисный предикатный символ и c — состояние. Тогда, если всякий кортеж $(b_1, \ldots, b_n) \in [a_1] \times \ldots \times [a_n]$ принадлежит R^c, то будем говорить, что предикат R истинен для аргументов $([a_1], \ldots, [a_n])$ в состоянии c, и обозначать это $\vdash^{nc} R([a_1], \ldots, [a_n])$. Тем самым логические значения базисных предикатов определены для классов эквивалентности, которые выступают в качестве новых элементов по отношению к носителю $\boldsymbol{O}$. Обозначим $\boldsymbol{O}^{nc}$ фактор-множество, определяемое эквивалентностью E и состоянием c. Построенная интерпретация, носителем которой является это фактор-множество, называется *нормальной*.

Из этого определения следует, что $\vdash^{nc} R([a_1], \ldots, [a_n]) \Leftrightarrow \vdash^{c} R(b_1, \ldots, b_n)$ для всех кортежей $(b_1, \ldots, b_n) \in [a_1] \times \ldots \times [a_n]$.

Отношения истинности и ложности формул в нормальной интерпретации распространяются на произвольные формулы обычным образом.

Теперь уже нетрудно определить связь нормальных операционных и логических семантик. С этой целью для задачи Z^n носителем интерпретации в состоянии c берется фактор-множество $\boldsymbol{O}^{nc}$ и осуществляются лишь нормальные применения действий. Отличие от прежнего определения задачи Z^n состоит в том, что

во-первых, нормальные ситуации трактуются как подстановки из фактор-множества $\boldsymbol{O}^{nc}$,

во-вторых, изменение $\Delta R^c_{k_i}$ содержит в точности все те кортежи, которые удовлетворяют следующим условиям. Для каждой подстановки $\sigma = \{\boldsymbol{x} \leftarrow [\boldsymbol{a}]\} \in \mathrm{DET}^{nc}$ такой, что $\vdash^{nc} L(\sigma \boldsymbol{x})$, $\Delta R^c_{k_i}$ включает кортеж $\varphi_i(\sigma \boldsymbol{x})$, $i = 1, 2, \ldots, t$.

Заметим, что здесь $\{\boldsymbol{x} \leftarrow [\boldsymbol{a}]\}$ понимается как подстановка из фактор-множества, а не как совокупность подстановок кортежей декартова произведения классов эквивалентности.

Теперь из доказанного ранее вытекает утверждение.

Теорема 28. *Если c есть неподвижная точка задачи Z^n, то имеет место отношение $\vdash^{nc} \boldsymbol{Z}$.*

Таким образом, неподвижная точка задачи Z^n определяет нормальную модель исчисления $\boldsymbol{Z}$.

С другой стороны, справедлива и обратная теорема

Теорема 29. *Пусть имеет место отношения $\vdash^{nc} \boldsymbol{Z}$ и o_c есть описание состояния задачи Z^n. Тогда c есть неподвижная точка задачи Z^n.*

Тем самым мы показали как перейти от обычной задачи к ее нормальной формулировке, т. е. задаче, область определения которой представляет собой совокупность классов эквивалентности. Задача может содержать несколько эквивалентностей и включать предикаты, которые не сохраняют те или иные эквивалентности. В этом случае в формулировку нормальной задачи вводятся соответствующие действия subst_P и subst_φ для тех предикатов и функциональных символов, которые сохраняют соответствующие эквивалентности.

Переход от обычных моделей к нормальным позволяют существенно повысить возможности логического моделирования при описании реальных систем, так как спецификации в терминах фактор-множеств в большинстве случаев более естественны.

§ 4. Практические аспекты логического моделирования

В предыдущих разделах был сформулирован теоретический базис, который положен в основу реальной системы логического моделирования. Ниже подводится итог этим теоретическим исследованиям описанием двухуровневого языка логического моделирования Покос. Его первый (логический) уровень представляет собой описание предметной области, которое вводит все понятия, необходимые для решения, и определяют на них частичный порядок. Это описание представляет собой совокупность аксиом прикладного исчисления первого порядка, для которого в процессе вычисления строится модель.

Все аксиомы имеют вид так называемых *продукций*:

ЕСЛИ ⟨условие-1⟩ ТО ⟨действие-1 ⟩И
ЕСЛИ ⟨условие-2⟩ ТО ⟨действие-2⟩ И
. . . . . .
ЕСЛИ ⟨условие-n⟩ ТО ⟨действие-n⟩.

Они представляют собой формулы первого порядка в определенной сигнатуре, которая определяется отношениями и функциями предметной области, а также набором встроенных функций и отношений языка Покос. Поэтому задача формулируется в терминах соответствующей проблемной области, и набор ее базисных отношений и функций близок к лексике соответствующего специалиста.

Вычисление продукции состоит в проверке условий из левых частей импликаций и, в случае их выполнения, реализации соответствующих действий, которые, как правило, сводятся к определенным изменениям пространства поиска. Уже образованные элементы пространства поиска хранятся в так называемой *динамической базе данных* (ДБД) — аналоге баз данных реляционного типа. Ее содержание меняется в процессе поиска решения.

Поиск решения на Покосе представляет собой построение логической модели аксиом предметной области. Ее конструирование происходит в соответствии со стратегиями, определяемыми механизмом управления применения продукций, который позволяет манипулировать как аксиомами, так и элементами пространства поиска. В качестве такого средства используется язык управления решением, который в качестве базисных конструкций содержит операторы манипулирования аксиомами и элементами ДБД.

Решение представляет собой фрагмент предметной области, на котором выполняются определенные условия. Последние формулируются в исходной спецификации. Решение управляется стратегиями, задаваемыми пользователем, исходя из практических критериев, связанных с имеющимися в распоряжении вычислительными ресурсами.

Тем самым, решение задачи как поиск модели прикладного исчисления, определяется двумя компонентами: *декларативной* — описаниями понятий предметной области, и *процедурной* — заданием стратегии поиска. Одна и та же содержательная задача может формулироваться над различными понятийными системами. Выбирая наиболее адекватную, удается получить достаточно компактные программы.

Предлагаемый подход не требует априорного знания всего решения, достаточно иметь представление лишь о некотором его начальном фрагменте. По мере накопления знаний об исходной проблеме происходит уточнение как аксиоматики, так и фрагмента пространства поиска, который предположительно содержит решение. Одновременно происходит уточнение стратегий поиска, из-за чего перебор приобретает приемлемый характер, в результате сведения его к совокупности локальных действий.

4.1. Язык описания предметной области. *Язык описания предметной области* (ЯОП) представляет собой многосортный язык первого порядка. Для более наглядного описания ЯОП выделим в нем иерархию из трех уровней его конструкций, конструкции более высокого уровня строятся на основе конструкций низших. Идея такой многоуровневости вытекает из следующих соображений.

Во-первых, при формализации содержательных спецификаций, приходится оперировать с обычными списками символов для образования из них различных имен: констант, переменных, функций, отношений, сортов и т. д. Во-вторых, синтаксис формул первого порядка налагает собственные требования на конструкции языка, что диктуется выбранной концепцией поиска решения.

Конструкции нулевого уровня — *имена* суть обычные идентификаторы.

Наибольшую долю в языке предметной области занимают конструкции первого уровня: константы, переменные, функции.

Второй уровень образуют слова и множества из индивидных термов.

Программа на ЯОП состоит из нескольких разделов, в каждом из которых описываются ее составляющие, необходимые для последующего вычисления. Основные конструкции ЯОП (продукции) аналогичны формулам многосортного исчисления первого порядка.

Первым разделом программы является раздел *DECLARATIONS*, в котором описываются все базисные конструкции языка. Это суть сорта переменных, констант и функций; константы и функции, которые используются для описания предметной области; а также отношения, среди которых особо выделяются эквивалентности.

Сорт есть слово нулевого уровня из фиксированного для данной проблемной области множества. На сортах определен частичный порядок $<$ таким образом, что они образуют решетку с одним наибольшим и одним наименьшим сортами (соответственно, $TERM$ и NUL).

Пример 32. Определение типов треугольников для программы решения задач по планиметрии может выглядеть следующим образом: $TERM <$ треугольник$<$ прямоугольный $< NUL$; $TERM <$ треугольник$<$ равнобедренный $<$ равносторонний $< NUL$.

Таким образом, сорт *равнобедренный треугольник* является частным случаем сорта *треугольник*, а *равносторонний треугольник* — частным случаем *равнобедренного*.

В Покосе имеются сорта с фиксированным значением. Это, в первую очередь, $TERM$, NUL, INT, RAT, STR.

$TERM$ — самый общий сорт. Любой другой сорт есть его частный случай.

NUL — пустой сорт. Является подсортом любого другого сорта.

INT — сорт целых чисел. Константа с этим сортом записывается как $INT.n$, где n — целое число.

RAT — рациональная дробь. Константа с этим сортом записывается как $RAT.n/m$, где n, m — целые числа, n — числитель, m — знаменатель.

STR — строка символов.

Константы служат для обозначения объектов предметной области. Каждая константа характеризуется сортом и именем. Объявление константы производится либо ее записью в разделе *CONSTANTS* либо определением в продукциях. Поэтому в программе могут использоваться необъявленные константы. Записываются константы следующим образом: *имя-сорта.имя-константы*.

Каждая константа однозначно определяется своим именем, поэтому у двух различных констант не может быть одного имени. Но разные вхождения одной и той же константы могут иметь разные сорта. В этом случае сорта должны быть попарно сравнимы относительно частичного порядка сортов данной области.

Пример 33. Если в программе сорта задают типы треугольников, то возможно следующее описание констант: *треугольник.ABC*; *равнобедренный.EFG*; *прямоугольный.ABC*. В последующем константа *треугольник.ABC* может входить в таблицы или в логические формулы как *треугольник.ABC*, *равносторонний.ABC*, *прямоугольный.ABC* или *равнобедренный.ABC*. В процессе работы программы могут возникнуть другие константы с определенными выше сортами, например, *треугольник.DEF* или *прямоугольный.PQR*.

Переменные — как и обычные индивидные переменные в логических формулах, служат для обозначения элементов предметной области. Наподобие констант, они определяются сортом и именем. Переменные используют-

ся только в продукциях, каждая переменная является локальной для отдельной продукции. Переменные с одинаковыми именами в разных продукциях рассматриваются как независимые. Переменные имеют вид *&сорт. имя.*

Пример 34. В задаче о миссионерах и людоедах, продукция, описывающая перевозку одного миссионера и одного людоеда с одного берега на другой, может выглядеть так:

NAME: переезд 1 миссионера и 1 людоеда на правый берег
KEYS: $P(\&INT.x, \&INT.y, INT.1)$
COND: $(\&INT.x \geqslant INT.1)AND(\&INT.y \geqslant INT.1)AND$
$((\&INT.x = INT.1)OR(\&INT.x = \&INT.y))$
CONC: $P(\&INT.x - INT.1, \&INT.y - INT.1, INT.0)$
COM:
END

Здесь переменные $\&INT.x$ и $\&INT.y$ определяются в ключах продукции. Затем, они используются в условии (раздел *COND*) и в заключении (раздел *CONC*) продукции. Здесь же используются две константы $INT.0$ и $INT.1$.

Предикаты в ЯОП предусмотрены двух типов — *вычисляемые* и *символические.* Первые служат для установления некоторых, обычно традиционных, отношений: равенства, порядка, включения и т. п. между объектами предметной области. Вторые фиксируют отношения, которые характерны для данной предметной области. В каждом состоянии задачи эти предикаты интерпретируются одноименными отношениями, всякое отношение задается единственной таблицей и имя таблицы совпадает с именем предиката. Поиск решения сводится к доопределению этих таблиц с учетом условий, сформулированных в продукциях.

Предикаты определяются простым перечислением их имен и следующих за ними аргументов, заключенных в скобки.

Пример 35. Описание предикатов для задачи о миссионерах и людоедах может выглядеть следующим образом: $P(INT.3, INT.3, INT.1)$. Запись $P(INT.3, INT.3, INT.1)$ определяет исходную ситуацию: три миссионера — $INT.3$ и три людоеда — $INT.3$ находятся на левом берегу — $INT.1$. В задаче используется единственный сорт INT.

Вычисляемые предикаты используются в условиях продукции, и каждому из них соответствует некоторая процедура, вычисляющая значения *истина* или *ложь* в зависимости от значений аргументов. Например, равенство записывается стандартным образом: *терм*1 = *терм*2. Этому отношению соответствует процедура, результатом которой является *истина*, когда *терм*1 совпадает с *терм*2, и *ложь* в противном случае. Основной набор таких предикатов включает — равенство, сравнения и неравенство. Для образования условий используются обычные логические операторы AND, OR, XOR, NOT.

Пример 36. Термы $INT.2$ и $RAT.4/2$ — равны, т. е. предикат $INT.2 = RAT.4/2$ после вычисления принимает значение *истина.* Термы $SORT.2$ и $SORT.4/2$ — не равны, т. е. предикат $SORT.2 = SORT.4/2$ после вычисления принимает значение *ложь.*

Пример 37. В условной части продукции может располагаться такая запись, состоящая из равенств, сравнений и логических связок.

COND:
$(\&INT.x \geqslant INT.1)AND(\&INT.y \geqslant INT.1)AND$
$((\&INT.x = INT.1)OR(\&INT.x = \&INT.y))$.

Функции в ЯОП так же как и предикаты бывают вычисляемые и символические. Назначение символических функций заключается в единообразном представлении сложных объектов предметной области — стеков, деревьев, схем и т. п., поэтому они не обладают иным значением кроме их

собственного вида. Символические функции определяются следующим образом: *сорт функции*: *имя функции* (*список сортов аргументов*).

Вычисляемые функции реализуются некоторыми встроенными процедурами. С помощью встроенных функций реализованы математические операции — сложение, вычитание, деление и умножение. Функции-аналоги математических операций записываются в принятой форме. Функции, реализующие нестандартные математические операции, и другие процедуры записываются как имя функции, за которым следует список аргументов, заключенный в скобки. Некоторые функции обладают не фиксированным числом аргументов.

Пример 38. Символическая функция *set* от нефиксированного количества аргументов, определяет упорядоченное множество своих аргументов. При помощи функции *cycle* можно циклически переставлять элементы множества, а при помощи функции *firstfromset* — получить первый элемент упорядоченного множества. Результат вычисления функции *firstfromset*(*set*(*SORT.A*, *SORT.B*, *SORT.C*)) равен *SORT.A*.

Каждое отношение *эквивалентности* задается идентификатором, и определяет соответствующие классы эквивалентности. Класс эквивалентности содержит один или несколько термов. По свойству рефлексивности, любой терм эквивалентен самому себе и поэтому исходно уже образует класс эквивалентности, состоящий из одного элемента.

Понятие *терма* в Покосе используется в традиционном математическом смысле. Чтобы терм был правильным должно выполняться требование соответствия сортов подставляемых констант и переменных сортам аргументов функций.

Пример 39. Термы *INT*.1, *INT*.1 + *RAT*.12/5 — правильные. Терм *INT*.1+ *треугольник.ABC* — неправильный.

Продукции — это логические формулы, выступающие как операторы для порождения новых строк таблиц, интерпретирующих предикаты. Каждая продукция, представленная в традиционном логическом формализме, выглядит следующим образом:

$$\Gamma \Rightarrow (F_1 \Rightarrow A_1) \,\&\, \ldots \,\&\, (F_n \Rightarrow A_n).$$

Здесь $\Gamma = L_1 \,\&\, \ldots \,\&\, L_m$, где L_i, $i = 1, 2, \ldots, m$, суть литеры, образованные из символических предикатов (они называются ключами продукции), аргументами которых служат переменные, константы или термы, образованные с помощью символических функций. Эта конъюнкция есть посылка всей продукции. Переменные из положительных литер продукции, входящих в Γ, называются *определяемыми*.

Бескванторные логические формулы F_i, $i = 1, 2, \ldots, m$, образованы из вычисляемых и символических предикатов и отношений эквивалентности. Все их индивидные переменные должны быть определены. Каждая из формул F_i представляет собой условие, которое проверяется прежде, чем будет осуществлено действие, описанное в части A_i.

Вычисление условия происходит следующим образом. Вычисляемые предикаты по каждому набору аргументов выдают значение *истина* или *ложь*. Для символических предикатов проверяется присутствие строки аргументов в одноименной таблице. Если такой кортеж в таблице имеется, то этот предикат также принимает значение *истина*, в противном случае — *ложь*. Если аргументом условия служит эквивалентность, то проверяется присутствие двух термов в одном классе эквивалентности. Присутствие их в одном классе влечет значение *истина*, отсутствие — *ложь*. В результате этих действий условие принимает в точности одно значение — *истина* или *ложь*.

A_i, $i = 1, 2, \ldots, m$, суть литеры, образованные из символических предикатов, чьи индивидные переменные определены в Γ. Они представляют

собой заключения продукции, каждое заключение выполняется в том случае, когда соответствующее условие принимает значение *истина*.

Заключение представляет собой набор литер, образованных из символических предикатов, на аргументных местах которых стоят произвольные термы, зависящие от переменных, которые определены в ключах продукции. Вычисление продукции состоит в том, что в случае выполнения условия в одноименную таблицу добавляется эта строка термов с соответствующими подстановками на места переменных (случай положительного вхождения) или исключается строка (случай отрицательного вхождения).

Например, в заключительной части *CONC*: $P(\&INT.x - INT.1, \&INT.y - INT.1, INT.0)$ продукции происходит изменение таблицы P. При этом добавляемая строка определяется результирующей строкой аргументов. Для всякой негативной литеры **not**R заключения из таблицы R вычеркивается порожденный кортеж аргументов, если он в ней присутствует.

4.2. Команды и операторы языка управления решением. *Язык управления решением* (ЯУР) служит для описания стратегий поиска и является метаязыком по отношению к ЯОП, так как данными для него выступают, в том числе, и конструкции ЯОП. Каждая программа на Покосе состоит из двух фрагментов, представленных на языках описания предметной области и управления решением. Первый служит для описания аксиоматики прикладного исчисления, а второй — для управления построением логической модели. Поэтому ЯУР напоминает обычный операторный язык программирования, так как в нем имеются стандартные управляющие конструкции: условные операторы, операторы цикла и перехода по метке и т. п. С другой стороны, он снабжен набором оригинальных операторов, предназначенных для манипулирования содержимым таблицами и выполнения продукций.

Ниже перечислены некоторые наиболее значимые конструкции, содержащиеся в ЯУР. При описании языка придерживаемся следующего соглашения: все выделенные жирным слова суть термины, принадлежащие языку, нежирные слова представляют собой имена отношений, таблиц, условий, операторов, разделителей и т. п.

Блоки операторов ограничиваются ключевыми словами **begin** и **end**, имеются операторы альтернативы **if-then-else**, цикла **while**, перехода по метке **goto**.

Условие — это логическое выражение в базисе **and**, **or**, **not**, **xor** над переменными и вычисляемыми логическими функциями. Последние являются встроенными средствами языка. По сравнению с обычными отношениями равенства, порядка и т. п., в ЯУР встречаются характерные лишь для него логические функции. Логическая функция **fixedpoint**(⟨имя продукции⟩) принимает значение *истина*, если после очередного вычисления указанной продукции не было зафиксировано изменений в таблицах, соответствующих ее заключению, **empty**(⟨имя предиката⟩) истинна, если одноименная таблица пуста, **have**(⟨имя предиката⟩, ⟨строка аргументов⟩) истинна, если соответствующая таблица содержит указанную строку аргументов.

Для работы с таблицами используются *директивные* операторы, предназначенные для изменения их содержимого. Они позволяют изменять таблицы, а также копировать и перемещать строки из одних таблиц в другие.

В зависимости от выполняемого действия, директивный оператор записывается в одном из следующих форматов:

(1) ⟨действие-1⟩ ⟨место действия-1⟩ **in** ⟨предикат-1⟩ **to** ⟨место действия-2⟩ **in** ⟨предикат-2⟩;

(2) ⟨действие-2⟩ ⟨место действия-1⟩ **in** ⟨предикат-1⟩

При этом действие-1 ::= **move** | **copy**; действие-2 ::= **erase** | **cut** | **paste** | **add**. Они определяют, соответственно, перемещение, копирование, чистку, вырезание в буфер, вставку из буфера и добавление строк.

Место-действия ::= **firstline** | **lastline** | **all** | **from** ⟨разделитель⟩ **upto** ⟨разделитель⟩ | **from** ⟨разделитель⟩ **downto** ⟨разделитель⟩ | **match** ⟨строка аргументов⟩ | **not match** ⟨строка аргументов⟩ | **before** ⟨разделитель⟩ | **after** ⟨разделитель⟩ | **before** ⟨строка аргументов⟩.

Этот фрагмент оператора определяет строки, над которыми выполняется соответствующее действие. Для разных действий множество допустимых мест, где они могут выполняться, различно. Так конструкции **firstline**, **lastline**, **all**, **match** ⟨строка аргументов⟩ и **before** ⟨разделитель⟩ по отношению к таблице-источнику понимаются соответственно, как первая строка, последняя строка, все строки, строки, совпадающие с маской, и все строки до разделителя. Те же конструкции по отношению к таблице-приемнику понимаются соответственно, как «вместо первой строки», «вместо последней строки», «вместо всех строк», «вместо строк, совпадающих с маской» и «до разделителя».

Указатель **all** выделяет содержимое всей таблицы. В рассматриваемой ниже задаче о восьми ферзях директива **erase in** *bit*1 **all** чистит таблицу, описывающую битые поля. Оператор с выделенным местом действия **from** *a*1 **to** *a*2 выделяет фрагмент таблицы между разделителями *a*1 и *a*2. Вместо *a*1 можно поставить **top**, то есть 1, вместо *a*2 — **bottom**, т. е. номер ее последней записи.

Действия и способы указания места в директивных операторах позволяют вырезать, копировать и перемещать строки из таблицы в таблицу и из таблицы в буфер и обратно. Также возможно выполнение директивного оператора для части таблицы, удовлетворяющей определенным условиям.

Строка аргументов в директивном операторе состоит из констант и символа подчеркивания, который определяет немаскируемые места в маске.

Приведем примеры действий, осуществляемых директивными операторами. Конструкция **add** ⟨строка⟩ добавляет новую строку в таблицу. Например, оператор **add** (*s*1.*name*1, *s*2.*name*2) **before-start in** *pred*1 добавляет строку *s*1.*name*1, *s*2.*name*2 в начало таблицы *pred*1. Оператор, содержащий фрагмент **erase**, удаляет указываемый фрагмент таблицы. Например, оператор **erase all in** *bit* удалит все строки из таблицы *bit*. Фрагмент **cut** позволяет перемещать фрагмент таблицы в буфер.

Пример 40. Пусть изначально таблица *pred* имела следующий вид:

$$(S.A, S.B, S.C)(S.A, S.C, S.C)(S.A, S.D, S.C)(S.A, S.E, S.A)(S.A, S.F, S.A).$$

Применение оператора **erase match** *pred*(_, *S.D*, _) **in** *pred* изменит эту таблицу на

$$(S.A, S.B, S.C)(S.A, S.C, S.C)(S.A, S.E, S.A)(S.A, S.F, S.A)$$

т. е. будет удалена третья строка.

Директивный оператор **erase match** *pred*(*S.A*, _, _) **in** *pred* удалит все строки таблицы, оператор **erase in** *pred* **on** *pred*(_, *S.D*, _) **up** удалит все строки таблицы сначала до строки со вторым аргументом *S.D*, т. е. изменит ее содержимое на такое: (*S.A*, *S.D*, *S.C*)(*S.A*, *S.E*, *S.A*)(*S.A*, *S.F*, *S.A*).

С каждой таблицей ассоциирован свой список разделителей. Разделители весьма облегчают описание разнообразных стратегий. Каждый разделитель имеет уникальное имя, в одной таблице не может быть двух одинаковых разделителей. Для манипулирования разделителями используются директивные операторы **mark** и **unmark**. Установка разделителей производится при помощи оператора следующего вида **mark** *таблица разделитель*

способ-разделения, где *таблица* задает таблицу, *разделитель* указывает название разделителя, *способ-разделения* — это одна из следующих директив: **at-begin**, **at-end**, **before** ⟨условие⟩ или **after** ⟨условие⟩.

Оператор **unmark** *таблица разделитель* удаляет разделитель из таблицы.

Помимо директивных, в Покосе присутствуют т. н. *продуктивные* операторы, осуществляющие выполнение продукций. Они построены по следующему шаблону: **run** ⟨имя продукции⟩ ⟨тип выполнения⟩ ⟨место выполнения⟩.

Тип выполнения определяет, как будет выполняться продукция — один раз или несколько. Некоторые типы выполнения приведены ниже. Оператор с конструкцией **until-fix** ⟨предикат⟩ выполняется до достижения неподвижной точки указанного предиката. Фрагмент **once** оператора разрешает выполнение продукции лишь один раз. По умолчанию, если после имени продукции не стоит **once**, она выполняется один раз. Оператор, содержащий конструкцию **for** *a*1 выполнится лишь один раз для строки, на которую указывает разделитель *a*1

Место выполнения продуктивного оператора указывает тот фрагмент таблицы, строки которого унифицируются с переменными соответствующего ключа продукции. Эта часть оператора имеет такой же вид, как соответствующая часть для директивного оператора.

4.3. Примеры программ и приемов программирования. В качестве демонстрации возможностей Покоса приведем несколько программ, реализующих общеизвестные стратегии перебора. Для простоты предположим, что пространство поиска упорядочено в виде иерархического дерева с единственным корнем и с фиксированной выходной степенью вершин. Каждая вершина характеризуется единственным кортежем отношения *Tree*, которое в начальный момент содержит лишь описание исходного состояния, а все узлы, расположенные непосредственно ниже узла N, порождаются кортежем, приписанным N. Чтобы не загромождать изложения, предположим, что продукция *Gen* – *Tree* по одному узлу порождает описания всех непосредственно ниже лежащих.

Пример 41. Стратегия перебора в ширину с удалением всякий раз просмотренного фрагмента дерева выглядит следующим образом. Здесь *Cond* есть условие порождения пространства поиска задачи.

while *Cond*
begin
mark *Tree*@*A* **at end**
{* в самый низ таблицы *Tree* помещается разделитель @*A* для разделения имеющихся элементов от вновь помещаемых *}
move from top to @*A* **in** *Tree* **to all in** *Gen*–*Tree*–*Input*
run *Gen* – *Tree*
move all in *Gen*–*Tree*–*Output* **to after** @*A* **in** *Tree*
{* выделяем в таблице *Tree* фрагмент сверху до разделителя @*A* и перемещаем в таблицу *Gen* – *Tree* – *Input*. Затем с помощью продукции *Gen*–*Tree*–*Input* по таблице *Gen*–*Tree*–*Input* порождаем новые данные, которые помещаем в таблицу *Gen* – *Tree* – *Input*. После этого данные из *Gen*–*Tree*–*Input* перемещаются в таблицу *Tree**}
unmark *Tree*@*A*
{* разделитель @*A* удаляется из таблицы *Tree*. Теперь в *Tree* хранится лишь вновь полученный уровень *}
end

Пример 42. Еще проще выглядит программа перебора в глубину с удалением просмотренных узлов:

while *Cond*

begin
move last line in *Tree* **to all in** *Gen – Tree – Input*
run *Gen – Tree*
move all in *Gen – Tree – Output* **to after end in** *Tree*
{* выделяем последнюю строку таблицы *Tree* и по ней порождаем новые данные, которые помещаем в самый низ таблицы *Tree*. Теперь в *Tree* самый нижний фрагмент есть вновь порожденный уровень пространства поиска, который порожден единственной вершиной *}
end

Остановка программ в обоих примерах происходит либо при нарушении условия, либо в случае, если порождаемая таблица оказывается пустой.

Пример 43. Всякие операции, связанные с анализом весов узлов дерева, реализуются продукциями. Так при реализации $\alpha - \beta$-процедуры в таблицу *Tree* помещается описание лишь тех узлов, которые приводят к соответствующему пересмотру веса порождаемого их узла (в зависимости от того, является ли она $\alpha-$ или $\beta-$ узлом). Для формирования очереди надо помещать новые записи вниз таблицы, а брать для порождения новых — сверху. Это легко реализуется, например, такой программой:

while *Cond*
begin
move first line in *Tree* **to all in** *Gen – Tree – Input*
run *Gen – Tree*
move all in *Gen – Tree – Output* **to after end in** *Tree*
{* выделяем первую строку таблицы *Tree* и по ней порождаем новые данные, которые помещаем в самый низ таблицы *Tree**}
end

Пример 44. В качестве примера программы на Покосе рассмотрим задачу о восьми ферзях. В ней используются следующие предикаты: *Ferz*(*INT.n*, *INT.x*) задает текущую конфигурацию на поле, n — номер ферзя и координата по вертикали, x — координата по горизонтали. Координаты шахматного поля обозначаются парой (y, x), где y — координата по вертикали, x — по горизонтали. Предикат *Bit*(*INT.y*, *INT.x*) обозначает, что поле (y, x) — битое. *Проверяемый* ферзь задается предикатом *Try*(*INT.n*), где n — номер ферзя. Если таблица *Try* не пуста, то необходимо выполнить проверку (шаг 3 алгоритма). *Выставляемый* ферзь задается предикатом *Place*(*INT.n*), *передвигаемый* задается предикатом *Move*(*INT.n*).

Алгоритм работы программы выглядит следующим образом.

1. Инициализация — устанавливается первая позиция (например 1,1). Выставляется 2-й ферзь, переход к п. 2.

2. Генерация битых полей всеми выставленными ферзями.

3. Проверка выставляемого ферзя. Если это 9-й ферзь — выход. Иначе на доску выставляется проверяемый ферзь и осуществляется переход к п. 4.

4. Проверка ферзя. Если он стоит на битом поле, то этот ферзь — передвигаемый и осуществляется переход к п. 5. В противном случае выставляется следующий ферзь и осуществляется переход к п. 2.

5. Если передвигаемый ферзь стоит на 8-й позиции, то он убирается, а предыдущий ферзь передвигается. После этого осуществляется переход к п 3. В противном случае этот ферзь проверяемый, и осуществляется переход к п. 4.

PREDICATES
{**Ferz* — выставленные на доске ферзи *}
Ferz(*INT*.1, *INT*.1)
{**Place*(*n*) обозначает передвижение ферзя с номером n. Ферзи нумеруются по вертикали, первый — это самый нижний, он располагается на первой горизонтали *}

Place(*INT*.2)
{∗*Iks* — вспомогательный предикат ∗}

Iks(*INT*.1)*Iks*(*INT*.2)*Iks*(*INT*.3)*Iks*(*INT*.4)
Iks(*INT*.5)*Iks*(*INT*.6)*Iks*(*INT*.7)*Iks*(*INT*.8)

{∗*Move*(*INT*.*m*) — обозначает, что следующий шаг состоит в передвижении ферзя с номером n, *ShowResult*(1) — что требуется показать построенный результат, *Try*(n) — требуется проверить новую позицию ферзя с номером n, *Bit*1(*INT*.*n*, *INT*.*y*, *INT*.*x*) — обозначает, что поле с координатами (y, x) бьется ферзем номер n, *Bit*(*INT*.*y*, *INT*.*x*) — что поле с координатами (y, x) бьется каким-то ферзем ∗}
PRODUCTIONS
{∗ *genBit* — осуществляет построение множества битых полей ∗}
NAME: *GenBit*1
KEY: *Ferz*(&*INT*.*f*, &*INT*.*x*0)*Iks*(&*INT*.*x*)*Iks*(&*INT*.*y*)
{∗ построение битых полей по диагоналям в соответствии с уравнением $y - x = f - x0$, где f — номер ферзя и координата по вертикали, $x0$ — координата по горизонтали ∗}
COND: ((&*INT*.*y*)=(&*INT*.*x*+&*INT*.*f*−&*INT*.*x*0))
CONC: *Bit*1(&*INT*.*f*, &*INT*.*y*, &*INT*.*x*)
{∗ решается уравнение $y - x = -(f - x0)$, где f — номер ферзя и координата по вертикали, $x0$ — координата по горизонтали ∗}
COND: ((&*INT*.*y*)=(&*INT*.*f*+&*INT*.*x*0−&*INT*.*x*))
CONC: *Bit*1(&*INT*.*f*, &*INT*.*y*, &*INT*.*x*)
{∗ построение битых полей по вертикалям и горизонталям. Пустой раздел *COND* равносилен тождественно истинному условию ∗}
COND:
CONC: *Bit*1(&*INT*.*f*, &*INT*.*x*, &*INT*.*x*0)*Bit*1(&*INT*.*f*, &*INT*.*f*, &*INT*.*x*)
END
{∗*GenBit*2 — образует таблицу *Bit* из таблицы *Bit*1∗}
NAME: *GenBit*2
KEY: *Bit*1(&*INT*.*f*, &*INT*.*y*, &*INT*.*x*)
COND:
CONC: *Bit*(&*INT*.*y*, &*INT*.*x*)
END
{∗*PlaceTest* — проверка и выставление ферзя ∗}
NAME: *PlaceTest*
KEYS: *Place*(&*INT*.*n*)
{∗ Проверка на выход за границу поля ∗}
COND: (&*INT*.*n* = *INT*.9)
CONC: *ShowResult*(*INT*.1) **not** *Place*(&*INT*.*n*)
{∗ Установка ферзя ∗}
COND: (&*INT*.*n* < *INT*.9)
CONC: *Ferz*(&*INT*.*n*, *INT*.1)*Try*(&*INT*.*n*) **not** *Place*(&*INT*.*n*)
END
{∗*Try* — проверка не находится ли ферзь на битом поле и передвижение проверяемого ферзя или выставление нового ∗}
NAME: *Try*
KEYS: *Try*(&*INT*.*n*)*Ferz*(&*INT*.*n*, &*INT*.*x*)
COND: *Bit*(&*INT*.*n*, &*INT*.*x*)
CONC: *Move*(&*INT*.*n*) **not** *Try*(&*INT*.*n*)
COND: **not** *Bit*(&*INT*.*n*, &*INT*.*x*)
CONC: *Place*(&*INT*.*n*+*INT*.1) **not** *Try*(&*INT*.*n*)
END

{*Move — передвижение ферзя на одну позицию; если это невозможно, то достигнут конец доски. Поэтому следует передвинуть предыдущего ферзя и стереть битые им поля *}

NAME: *Move*
KEYS: *Move*(&*INT.n*)*Ferz*(&*INT.n*, &*INT.x*)
COND: (&*INT.x* = *INT.*8)
CONC: *Move*(&*INT.n* − *INT.*1)**not** *Move*(&*INT.n*)
Clear(&*INT.n*)**not** *Ferz*(&*INT.n*, &*INT.x*)*Clear*(&*INT.n* − *INT.*1)
COND: (&*INT.x* < *INT.*8) **and** (&*INT.x* > *INT.*0)
CONC: *Ferz*(&*INT.n*, &*INT.x* + *INT.*1) **not** *Ferz*(&*INT.n*, &*INT.x*)
Clear(&*INT.n*) **not** *Move*(&*INT.n*)*Try*(&*INT.n*)
END
NAME: *Clear*1
KEYS: *Clear*(&*INT.n*)*Iks*(&*INT.y*)*Iks*(&*INT.x*)
COND: *Bit*1(&*INT.n*, &*INT.y*, &*INT.x*)
CONC: **not** *Bit*1(&*INT.n*, &*INT.y*, &*INT.x*)
END
NAME: *Clear*2
KEYS: *Iks*(&*INT.y*)*Iks*(&*INT.x*)
COND: *Bit*(&*INT.y*, &*INT.x*)
CONC: **not** *Bit*(&*INT.y*, &*INT.x*)
COND: *Clear*(&*INT.y*)
CONC: **not** *Clear*(&*INT.y*)
END
begin
P:
if empty(*Place*) **then goto** *T*
print "—- *RUNNINGGenBit*1, 2"

{* Генерация очередного «недоступного» пространства шахматного поля *}

run *GenBit*1
run *GenBit*2
print "—− *RUNNINGPlaceTest*"
run *PlaceTest*
if empty(*ShowResult*) **then goto** *T* **else goto***E*
T:
if empty(*Try*) **then goto** *M*
print "—−*RUNNINGTry*"
run *Try*
goto *P*
M:
if empty (*Move*) **then goto** *P*
print "—−*RUNNINGMove*"
run *Move*
print "—−*RUNNINGClear* & *GenBit*2"
run *Clear*1
run *Clear*2
run *GenBit*2
goto *P*
E:
print *Ferz*
end.

Найденное программой решение выглядит так:

(*INT.*1, *INT.*1), (*INT.*2, *INT.*5), (*INT.*3, *INT.*8), (*INT.*4, *INT.*6), (*INT.*5, *INT.*3), (*INT.*6, *INT.*7), (*INT.*7, *INT.*2), (*INT.*8, *INT.*4).

Поступило в редакцию 15 V 2001