Свойства стабильного поведения динамических дедуктивных баз данных.
Часть I. $\exists\exists-$стабильность и перспективность. [1]

Дехтярь Михаил Иосифович
170000 Тверь
ул. Желябова, 3
ТвГУ, кафедра информатики
dekhtyar@tsu.tversu.ac.ru

Диковский Александр Яковлевич
125047 Москва
Миусская пл. 4
ИПМ им. Келдыша РАН
dikovsky@applmat.msk.su

Резюме
В работе определены некоторые свойства устойчивого поведения дискретных динамических систем (в частности, стабильность и гомеостатичность) в терминах логических программ с модификациями над конечными базами данных с ограниченями целостности. Формальная модель, используемая нами, основана на новом понятии - ограниченном воздействии внешней среды. Основные результаты работы состоят в оценках вычислительной сложности рассматриваемых свойств при ограничениях, представляющих интерес для проиложений.

1 Введение

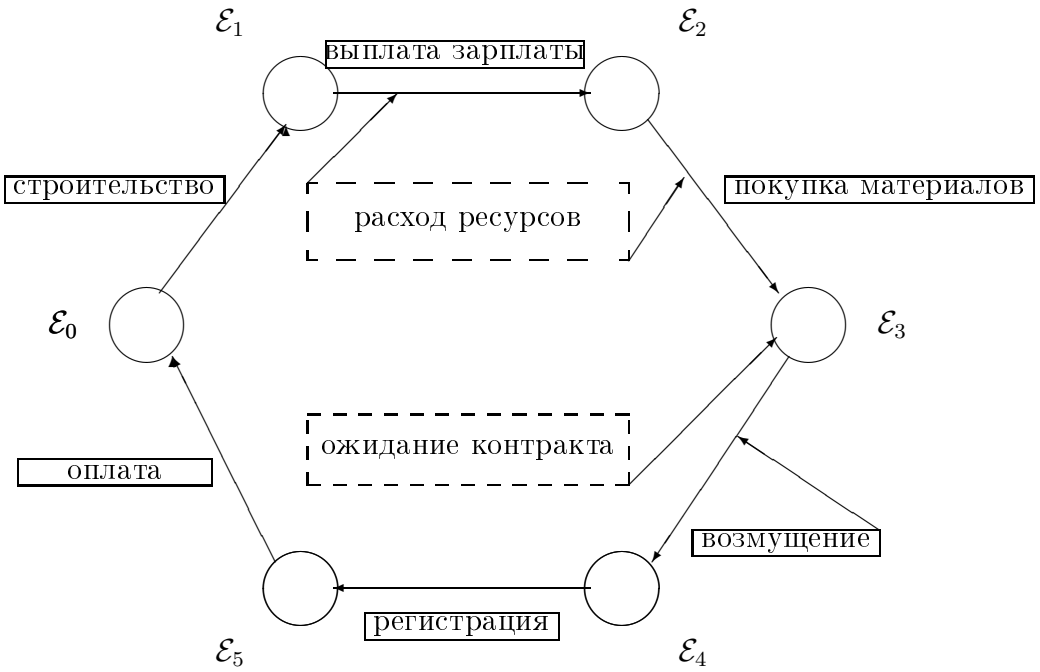
В данной работе развивается подход к анализу поведения дискретных динамических систем, предложенный в наших предыдущих публикациях [8, 9, 10]. В них дискретные динамические системы рассматриваются как недетерминированные преобразования состояний, представляемых конечными моделями (реляционными базами данных). В качестве языка описания свойств состояний и их преобразований естественно использовать логический язык 1-го порядка **L**. Мы ограничиваемся подмножеством Хорновских формул этого языка и синтаксисчески различаем описания состояний и их теорий. Далее мы, как это обычно делается, см. например, [20], выделяем *экстенсиональную* сигнатуру реляционных баз данных, а теории представляем, как логические программы $\mathcal{I}$ в некотором расширении этой сигнатуры (*интенсиональной* сигнатуре). При этом для всякой экстенсиональной базы данных $\mathcal{E}$ из $\mathcal{I} \cup \mathcal{E}$ не должны выводиться никакие экстенсиональные факты кроме $\mathcal{E}$. Динамическая природа

[1]Эта работа была поддержана грантом INTAS (Grant 94-2412) и Российским Фондом фундаментальных исследований (грант 93-012-627).

систем представляется с помощью простых динамических операторов *insert*, и *delete*, которые непосредственно изменяют базы данных (являются *элементарными модификациями*). При этом реляционные базы данных становятся изменяемыми состояниями баз данных (*состояниями БД*), а логические программы соответствуют попадают в традиционные рамки дедуктивных баз данных(*ДБД*) [16]. Их цели $:- G$ задают сложные модификации состояний БД. Операционная семантика программ основана на методе резолюций. Каждая цель $:- G$ индуцирует недетерминированный оператор на состояниях БД (*модификацию*) $\overset{G}{\vdash_{\mathcal{I}}}$ (или просто $\overset{G}{\vdash}$, если $\mathcal{I}$ подразумевается) определяемый естественным образом: $\mathcal{E} \overset{G}{\vdash_{\mathcal{I}}} \mathcal{E}'$ тогда и только тогда, когда существует успешное опровержение $\mathcal{I} \cup \mathcal{E} \cup \{ :- G\} \Rightarrow \square$, завершающееся в состоянии $\mathcal{E}'$. Этот оператор, в общем случае не является, чисто логическим, поскольку он немонотонен: некоторые факты, присутствующие в $\mathcal{E}$, могут отсутствовать в $\mathcal{E}'$. Состояния динамических систем естественным образом разбиваются на *допустимые* и *недопустимые*. Допустимые состояния традиционно задаются с помощью условий, называемых *ограничениями целостности* (IC). Обычно IC являются монотонными или, по крайней мере, их можно считать *стационарными,* т.е. такими условиями, проверка которых не изменяет состояний БД. В предыдущих работах авторов [8, 9, 10] IC задавались посредством стационарных логических программ. В данной работе мы задавать эти условия замкнутыми формулами логики 1-го порядка в экстенсиональной сигнатуре. Как правило, модификации, нарушающие IC, рассматривались как недопустимые (см. [16, 20, 22]). В этом состоит одно из главных отличий предлагаемого подхода от традиционного. При этом мы имеем в ввиду важный специальный класс приложений ДБД, в которых они рассматриваются как модели дискретных динамических систем, функционирующих в активной внешней среде, и чье поведение (как и реакции среды) зависит от некоторых ресурсов. В этих приложениях объемы доступных ресурсов ограничены, и эта ограниченность может быть задана как с помощью теории поведения динамической системы, так и посредством соответствующих IC. Действия динамических систем представляются как модификации состояний БД, задавемые программами. Возможные реакции внешней среды на эти действия представляются как непосредственные изменения состояний БД. И те, и другие могут влиять на объем доступных ресурсов. Отсюда получается основное отличие в подходах к трактовке ограничений целостности. Мы различаем два источника изменений состояний БД: модификации, вызванные действиями самой ДБД, и внешние *возмущения* среды, в которой эта ДБД действует. При таком взаимодействии действия одной стороны могут вызывать нарушения IC, а другой - восстанавливать выполнение этих условий. Рассмотрим следующий содержательный пример.

Предположим, что некоторая фирма заключает контракты и занимается строительством дачных домиков. Финансирование ее деятельности полностью

зависит от платежей заказчиков. Полученные от них суммы зачисляяются на текущий счет и тратятся на материалы и на выплату ежемесячной зарплаты служащим (см. Рис. 1). В состоянии $\mathcal{E}_0$ фирма располагает контрактом и строит дом. В следующем состоянии $\mathcal{E}_1$ она платит зарплату сотрудникам. Таким образом, переход в состояние $\mathcal{E}_2$ связан с расходом некоторых ресурсов. Будучи в состоянии $\mathcal{E}_2$, фирма приобретает необходимые материалы, снова затрачивая финансовые ресурсы, и оказывается в недопустимом состоянии $\mathcal{E}_3$. В этом состоянии финансовые ресурсы исчерпаны и нельзя совершить никакое действие до тех пор, пока не будет заключен и оплачен новый контракт. Таким образом, $\mathcal{E}_3$ - это состояние ожидания контракта на строительном рынке. Мы подчеркиваем, что появление контракта на рынке не регулируется правилами функционирования фирмы. Это событие является **внешним** по отношению к деятельности фирмы и представляет "возмущение"среды, в которой она действует. После появления требуемого контракта, переводящего систему в допустимое состояние $\mathcal{E}_4$, он регистрируется (переход в состояние $\mathcal{E}_5$), деньги помещаются на счет и система возвращается в начальное состояние $\mathcal{E}_0$. Следовательно, фирма может продолжать свою деятельность при условии, что на рынке будет иметься достаточное число контрактов.



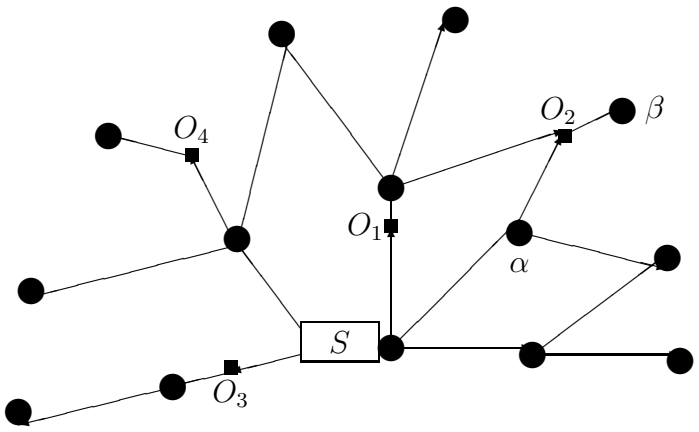
.1

Приведенный пример показывает, что в такого рода приложениях центральной задачей является поиск стратегии поведения, гарантирующей теоретически бесконечный процесс жизни системы (*устойчивое поведение*). Мы сосре-

доточимся на главном обстоятельстве, препятствующем устойчивому поведению динамической системы, - нарушению IC. Для простоты мы предположим, что процесс взаимодействия между системой и внешней средой всегда **начинается в допустимом состоянии,** и что первый шаг всегда выполняется той стороной, **которая может нарушить IC.** Другая сторона должна стараться **восстановить IC.** Таким образом, мы приходим к двум дуальным режимам устойчивого поведения. Режим, в котором первый шаг делается системой (как в нашем примере), мы называем *стабильным.* Стабильность означает, что нарушения IC, вызванные действиями динамической системы, могут быть исправлены за счет возмущений (коррекций) ее внешней среды. Т.е. система может функционировать бесконечно **за счет этих возмущений.**

Двойственный режим устойчивого поведения, при котором первый шаг выполняет среда, мы называем *гомеостатичным.* Гомеостатичность означает, что нарушения IC, вызванные возмущениями внешней среды, могуть быть исправлены за счет действий динамической системы. Другими словами, несмотря на разрушающие возмущения среды, система может функционировать бесконечно. Рассмотрим следующий пример.

Предположим, что некоторая другая фирма занимается доставкой товаров со склада по заказам. На рис. 2 показана схема дорог, заправочные станции отмеченны кружками, прямоугольник S обозначает местоположение склада. Квадратики, обозначенные O_i, обозначают местоположения возможных заказчиков.



.2

Предполагается, что:

- двигаясь в любом направлении с полностью заправленным баком, можно добраться до следующей заправочной станции;

- заказ на доставку принимается, если путь к заказчику не требует более двух заправок;

- грузовик с товаром стартует с полным баком и суммой денег, достаточной для одной дополнительной заправки;

- выполнение одного заказа обеспечивает сумму денег, достаточную для четырех заправок.

Действием этой системы является доставка по одному заказу. Появление же очередного заказа рассматривается как возмущение внешней среды. Как и в предыдущем примере, появление заказов не зависит от самой системы. Пусть условие IC означает отсутствие невыполненных заказов. Тогда приведенная система гомеостатична в том смысле, что если у фирмы есть достаточно грузовиков для развозки максимального числа заказов, то она может функционировать бесконечно. Например, если появился заказ на доставку товаров в O_2, то грузовик может проехать в эту точку через станцию α. Затем после доставки груза он должен добраться до ближайшей станции β для новой заправки. На обратном пути он должен заправиться еще раз и после возвращения в S снова наполнить бак и быть готовым к выполнению следующего заказа. Из указанных выше предположений следует, что для выполнения такого маршрута у водителя хватит денег.

До этого момента мы не давали точных определений устойчивого поведения . Без таких уточнений всегда можно восстановить выполнение IC для обеих моделей , по крайней мере теоретически. Для этого после всякого нарушения IC противоположной стороне достаточно просто восстановливать предыдущее допустимое состояние. Разумеется, в приложениях с ограниченными ресурсами такое восстановление возможно не всеегда. По этой причине разумно ввести некоторую меру обема возмущений и рассматривать только униформно ограниченные возмущения. Точнее, будем считать *траекторию* (т.е. последовательность состояний системы преобразуемых по очереди действиями системы и внешней среды) $\delta-$ограниченной, если размер каждого возмущения на этой траектории не превосходит δ. Простейшим способом измерения состояний является теоретико-множественный: состояния и границы на возмущения являются конечными множествами экстенсиональных атомов, а порядок на них задается отношением включения на множествах. Кроме этого способа мы также рассматриваем несколько более общий, основанный на *информационных эквивалентностях* на состояниях. Мы будем рассматривать два вида траекторий: *стабильные* и *гомеостатичные.* В первом случае действия системы применяются к допустимым состояниям, а возмущения (коррекции) восстанавливают IC, во втором - наоборот. Свойства $\delta-$стабильного и $\delta-$гомеостатичного поведения определяются существованием бесконечных стабильных или, соответственно, гомеостатичных $\delta-$ограниченных траекторий. В результате мы получаем различные релятивизированные модели устойчивого поведения. Например, $\delta-$ограниченная $\exists\exists-$стабильность в заданном состоянии (проиллюстрированная первым примером) означает непустоту множества бесконечных $\delta-$ограниченных стабильных траекторий, начинающихся

в этом состоянии. Второй пример иллюстрирует свойство $\delta-$ограниченного $\forall\exists-$гомеостатичного поведения в данном состоянии, это означает, что для каждого $\delta-$ограниченного возмущения, изменяющего это состояние, существует некоторое действие системы, приводящее к допустимому состоянию. Другими словами, это означает, что система может исправить все $\delta-$ограниченные деструктивные возмущения. В приложениях встречаются и некоторые другие виды неограниченно устойчивого поведения (например, $\delta-$ограниченная $\forall\exists-$стабильность или $\delta-$ограниченная $\forall\forall-$гомеостатичность). Эти свойства мы рассмотрим во второй части работы.

Понятия $\delta-$ограниченной стабильности и гомеостатичности характеризуют устойчивое поведение динамических систем на допустимых состояниях. Мы будем также рассматривать свойство $\delta-$ограниченной *перспективности*, которое характеризует поведение системы в недопустимых состояниях. Данное состояние $\mathcal{E}$ перспективно, если из него можно достичь посредством конечной $\delta-$ограниченной траектории некоторое допустимое состояние. Таким образом, пространство состояний разбивается на два класса: состояния из которых можно достичь допустимое состояние - ($\delta-$*перспективные*) состояния, и те, из которых нельзя - ($\delta-$*неперспективные* состояния). На самом деле, свойство $\delta-$ограниченной перспективности, которое дальше рассматривается, является $\delta-$ограниченной $\exists\exists-$перспективностью. Как и для стабильности, можно определить естественным образом и другие виды $\delta-$ограниченной перспективности, используя различные комбинации кванторов. Они будут исследованы в других работах.

a) b)

.3

Рассмотрим, например, конечную часть плоскости, разбитую на ячейки, с симметричным отношением соседства ν на них таким, что $C_1\nu C_2 \Rightarrow \neg\exists C$ $(C_1\nu C \ \&\ C\nu C_2)$ (следовательно, ν антирефлексивно). Предположим, что в ячейках могут находиться организмы Υ, поведение которых подчиняется следующим двум правилам:

- каждый организм может породить потомка в одной пустой соседней ячейке и

- организм погибает, если у него нет пустой соседней ячейки.

Действие этой системы - это одновременное применение этих правил (без конфликтов) ко всем организмам. Назовем состояние допустимым, если в нем жив хотя бы один организм и у каждого организма есть пустая соседняя ячейка. Тогда состояние a) на рис.3 перспективно, а состояние b) нет.

Понятно, что все свойства поведения систем и состояний, которые здесь рассматриваются, неразрешимы в общем случае, если не налагать никаких ограничений на языки теорий, задающих поведение, и на IC. Простейшие динамические языки - это языки, предложения которых имеют вид
$\Rightarrow$,
где - это конъюнкция $Con_1\&...\&Con_k$ экстенсиональных атомов или их отрицаний, а - это последовательность элементарных модификаций $Act_1, \ldots, Act_m$. Такого рода языки рассматривались в [1] и других работах тех же авторов(например, таким является язык транзакций DL), хотя и под несколько другим углом зрения. Языки этого вида широко используются в различных областях приложений, наиболее широко в искусственном интеллекте (в экспертных системах и т.п.), где такие предложения называются "продукциями". В виде клауз продукции представляются как

$$q :- Con_1, \ldots, Con_k, Act_1, \ldots, Act_m.$$

Поскольку q - интенсиональный символ, то продукции не рекурсивны.

Другой тип языков клауз, который мы рассматриваем, естественно возникает из требования модульности рекурсии: модули динамической логической программы образуют уровни, согласованные с элементарными модификациями аналогично тому, как отрицания согласуются со слоями стратифицированных логических программ [3]. Поэтому мы будем называть такие программы *динамически (d-) стратифицированными.*

Что же касается ограничений целостности, то мы рассмотрим в качестве IC некоторые подклассы формул 1-го порядка, которые с одной стороны покрывают многие классы приложений, а с другой - являются просто проверяемыми. Наболее сильное ограничени на IC, которое мы наукладываем - это *базисность* (отсутствие переменных) и *монотонность* (отсутствие отрицаний) формул. В нашей классификации это ограничение называется IC0-*критериям.* Следующий рассматриваемый нами класс состоит из базисных формул, это ограничение мы называем IC1-*критериям.* Еще более широким является класс IC, задаваемых $\exists$−формулами. Мы будем говорить, что ограничения из этого класса удовлетворяют IC2-*критерию.*

В этой работе мы изучаем вычислительную сложность свойств перспективности и $\exists\exists$-стабильности поведения дискретных динамических систем, описываемых продукционными или d-стратифицированными теориями и ограничени-

ями целостности, удовлетворяющими одному из указанных IC-критериев. Эти два свойства, очевидно, не являются эквивалентными в классе всех динамических дедуктивных баз данных. Первое из них эквивалентно проблеме остановки, а ко второму легко можно свести проблему всюду определенности частично рекурсивных функций. Тем не менее, их сложность во многих рассматриваемых нами случаях одинакова. Например, в случае продукционных плоских (т.е. без сложных термов) ДДБ обе проблемы являются $SPACE(2^{poly})$-полными. Для ДДБ с базисными продукциями обе проблемы являются $PSPACE$-полными. В некоторых более узких классах ДДБ сложности этих проблем различны. Например, для класса базисных продукционных систем, не удаляющих факты из состояний, с ограничениями целостности, удовлетворяющими IC2-критерию ограниченная $\exists\exists$-стабильность является NP-полной, а ограниченная перспективность является $PSPACE$-полной. Однако в еще более узком классе базисных и позитивных (без отрицаний) продукционных систем обе проблемы являются NP-полными. При IC0-критерии перспективность распознаваема в этом классе ДДБ за линейное время, а $\exists\exists$-стабильность становится тривиальной. Интересно, что для d-стратифицируемых базисных плоских ДДБ сложность рассматриваемых проблем совпадает с их сложностью для соответствующих продукционных систем. А именно, обе проблемы $PSPACE$-полны для базисных d-стратифицированных ДДБ и являются $SPACE(2^{poly})$-полными для d-стратифицированных плоских ДДБ.

Эта статья организована следующим образом. В разделе 2 обсуждаются работы других авторов, связанные с данной. В разделе 3 собраны необходимые предварительные сведения. В разделе 4 вводятся основные понятия, связанные с устойчивым поведением, и определяется классификация динамических дедуктивных баз данных. Раздел 5 включает результаты, характеризующие вычислительную сложность проблемы ограниченной $\exists\exists$-перспективности для рассматриваемых классов ДДБ. Раздел 5 связан с соответствующими результатами о сложности ограниченной $\exists\exists$-стабильности.

2 Обзор литературы

В последнее десятилетие в области дедуктивных баз данных с модификациями проводились интенсивные исследования. В этой работе мы адаптируем традиционный подход дедуктивных баз данных (представленный, например, в [16]) к задаче описания свойств поведения дискретных динамических систем. Основное внимание уделялось языкам модификаций и анализу их выразимости и сложности (см. [1, 21, 20, 4]), а также декларативной и операционной семантикам модификаций (см. [22, 13, 5, 24]). Последняя тема тесно связана с анализом пересмотров и модификаций для пропозициональных баз (cf. [18, 14]). Центральные задачи в этой области связаны с определением результатов модификаций и про-

веркой их корректности относительно IC. Модификации, нарушающие IC, как правило рассматривались как недопустимые. Специфика приложений, рассматривавшихся нами, заставила изменить эту традиционную точку зрения на IC. Мы рассматриваем ситуации, когда модификации систем могут нарушать допустимость состояний БД, а возмущения внешней среды ДДБ, которые следуют за модификациями, восстанавливают IC (или обратные ситуации). Такая интерактивная точка зрения приводит естественно к определению свойств поведения систем в терминах траекторий. В нескольких работах поведение активных баз данных уже описывалось в терминах траекторий (трасс) (см. [5, 24, 7]), но в них шаги траекторий соответствовали только модификациям систем. Следует отметить, что в работе [24] было введено понятие "сюрпризов", призванное отразить влияние некоторых временных внешних событий на поведение ДДБ. В работе [7] рассмотрена сложность задачи планирования для класса базисных продукционных систем типа STRIPS. Технически эта работа весьма близка к той части нашей работы, в которой рассматривается проблема ограниченной перспективности.

3 Обозначения и предварительные определения

Мы рассматриваем логические программы с модификациями в языке 1-го порядка $\mathbf{L}$ как недетерминированные преобразования состояний баз данных (БД-состояний) Сигнатура языка $\mathbf{L}$ в общем случае содержит как предикатные символы $\mathbf{P}$, так и множество имен функций $\mathbf{F}$. Для получения естественной семантики модификаций мы, как и в работе [20], разбиваем $\mathbf{P}$ на две непересекающихся части: $\mathbf{P}^e$ (*экстенсиональные* предикаты) и $\mathbf{P}^i$ (*интенсиональные* предикаты). Эрбранов базис $\mathbf{B}$ языка $\mathbf{L}$ соответственно разбивается на $\mathbf{B}^e = \{p(t_1, ..., t_k) \mid t_i \in \mathbf{H}, p \in \mathbf{P}^e\}$ и $\mathbf{B}^i = \{q(u_1, ..., u_l) \mid u_i \in \mathbf{H}, q \in \mathbf{P}^i\}$, где $\mathbf{H}$ - это Эрбранов универсум языка $\mathbf{L}$. Состояния базы данных (БД) - это конечные подмножества $\mathbf{B}^e$. Головы клауз в логических программах всегда являются интенсиональными атомами. Тела клауз могут включать элементарные модификации баз данных двух видов: $insert(A), delete(A)$, где A - это экстенсиональный атом. В телах предложений допускаются также *экстенсиональные атомы с отрицаниями*. Для уменьшения размера программ в их телах мы допускаем использование дизъюнкции ";"и оператора модификации *change(A,B)* вместо комбинации операторов *delete(A), insert(B).*

Операционная семантика логических программ основана на традиционном SLD-опровержении, но имеет следующую специфику. Во-первых, используется прологовское правило выбора самого левой подцели с безопасными вызовами элементарных модификаций (т.е. все переменные в аргументах модификаций $insert(A)$ и $delete(A)$ в момент вызова должны быть базисными термами). Отрицание трактуется как конечная опровержимость, которая в нашем весьма

специальном случае означает, что отрицаение атома успешно, если его нельзя унифицировать ни с одним фактом текущего состояния БД. Успешное опровержение имеет побочный эффект, состоящий в преобразовании исходного состояния БД в некоторое новое состояние. С каждой логической программой с модификациями можно естественно связать следующее отношение $\mathcal{E} \vdash \mathcal{E}'$ на БД-состояниях.

Определение 1. *Пусть* $\mathcal{I}$ *- логическая программа,* $\mathcal{E}, \mathcal{E}'$ *- два состояния БД и* $:- G$ *- некоторая цель. Опровержение* $\mathcal{I} \cup \mathcal{E} \cup \{ :- G\}$ *- это конечная последовательность* $S = s_1, s_2, ..., s_n$ *троек (состояний вычисления) вида* $s_i = (\mathcal{E}_i, G_i, \sigma_i)$, *в которой* $\mathcal{E}_i$ *- состояние БД,* G_i *является целью и* σ_i *- подстановка,* $\mathcal{E}_1 = \mathcal{E}$, $G_1 = G$, σ_1 *- тождественная подстановка* ε, G_n *- пустая цель* $\Box$, *и каждый шаг* $(s_i, s_{i+1}), i < n$, *имеет один из следующих видов. Пусть* $G_i = L_1, L_2, \ldots, L_k$.

(1) Если L_1 *- атом, то для некоторого варианта клаузы (возможно, с пустым телом)* $H :- B_1, \ldots, B_r$ $(r \geq 0)$ *и для наиболее общего унификатора* θ *атомов* $L_1 \circ \sigma_i$ *и* H *и, как обычно, имеют место равенства* $G_{i+1} = B_1, \ldots, B_r, L_2, \ldots, L_k$, $\sigma_{i+1} = \sigma_i \circ \theta$ *и* $\mathcal{E}_{i+1} = \mathcal{E}_i$.

(2) Если $L_1 = \neg q(\bar{t})$ *для некоторого* $q \in \mathcal{P}^e$ *и* $q(\bar{t}) \circ \sigma_i$ *нельзя унифицировать ни с одним фактом* $\mathcal{E}_i$, *тогда имеют место равенства* $G_{i+1} = L_2, \ldots, L_k, \mathcal{E}_{i+1} = \mathcal{E}_i$ *и* $\sigma_{i+1} = \sigma_i$.

(3) Если $L_1 = insert(A)$ *и* $A \circ \sigma_i \in \mathbf{B}^e$ *то имеют место равенства* $G_{i+1} = L_2, \ldots, L_k$, $\mathcal{E}_{i+1} = \mathcal{E}_i \cup \{ A \circ \sigma_i\}$ *и* $\sigma_{i+1} = \sigma_i$.

(4) Если $L_1 = delete(A)$ *и* $A \circ \sigma_i \in \mathbf{B}^e$ *то имеют место равенства* $G_{i+1} = L_2, \ldots, L_k$, $\mathcal{E}_{i+1} = \mathcal{E}_i \setminus \{A \circ \sigma_i\}$ *и* $\sigma_{i+1} = \sigma_i$.

Если такое опровержение существует, то мы, как обычно, пишем $\mathcal{I} \cup \mathcal{E} \cup \{ :- G\} \Longrightarrow \Box$.
Если, кроме того, $\mathcal{E}_n = \mathcal{E}'$, *то мы скажем, что программа* $\mathcal{I}$ *с целью* $\{ :- G\}$ *преобразует/* $\mathcal{E}$ *в* $\mathcal{E}'$, *и будем обозначать это как* $\mathcal{E} \overset{G}{\vdash_{\mathcal{I}}} \mathcal{E}'$. *Посредством* $\mathcal{E} \overset{p}{\vdash_{\mathcal{I}}} \mathcal{E}'$ *мы будем обозначать шаг опровержения, на котором применяется клауза* p .

Из этого определения видно, что мы рассматриваем каждую цель логической программы как предопределенное недетерминированное преобразование состояний БД. В отличие от [5] мы не расширяем язык логических программ с тем, чтобы он отражал явно внутренние состояния БД в ходе опровержения. Тем не менее, наша семантика позволяет естественным образом налагать ограничения на эти состояния. Например, успешное применение клаузы

$$h :- insert(son(jack, tom)), \neg ancestor(jack, tom), etc, ...$$

возможно только в том случае, если добавление факта $son(jack, tom)$ к текущему состоянию БД не приведет к зацикливанию отношения "ancestor".

В реальных приложениях даннве в базах обычно можно разбить на классы

эквивалентности, поскольку некоторые атрибуты отношений играют чисто информационную роль, т.е. их изменение не влияет на существенные свойства баз данных. Данные, различающиеся только такими "несущественными"чертами, можно считать эквивалентными. Такое разбиение данных на классы иногда позволяет уменьшить потенциально бесконечную необозримую прикладную область до конечной или более обозримой. Некоторые варианты таких свойств уже рассматривались в теории. Например, так называемая C-обобщенность, используемая в ряде работ [17, 2, 1, 6] определяет на состояниях БД эквивалентность относительно переименования констант вне C. Мы вводим для отражения этой ситуации следующее понятие.

Определение 2. *Пусть* $\equiv$ *- некоторая эквивалентность на* $\mathbf{H}$. *Ее можно естественным образом распространить на* $\mathbf{B}^e$: $g(t_1, ..., t_n) \equiv g(t'_1, ..., t'_n)$ *тогда и только тогда, когда* $t_i \equiv t'_i$ *для всех* $1 \leq i \leq n$. *Для любых* $D_1, D_2 \subseteq \mathbf{B}^e$ *мы полагаем* $D_1 \Leftarrow D_2$, *если* $D_1/_{\equiv} \subseteq D_2/_{\equiv}$. $D_1 \Leftrightarrow D_2$ *если* $D_1 \Leftarrow D_2$ *и* $D_2 \Leftarrow D_1$.

Логическая программа $\mathcal{I} \cup \{ :- G\}$ *является* согласованной с эквивалентностью $\equiv$, *если для любых трех состояний* $\mathcal{E}_1, \mathcal{E}'_1, \mathcal{E}_2 \subseteq \mathbf{B}^e$, *таких что* $\mathcal{E}_1 \Leftrightarrow \mathcal{E}_2$ *и* $\mathcal{E}_1 \overset{G}{\vdash}_{\mathcal{I}} \mathcal{E}'_1$ *существует такое состояние* $\mathcal{E}'_2$, *что* $\mathcal{E}'_1 \Leftrightarrow \mathcal{E}'_2$ *и* $\mathcal{E}_2 \overset{G}{\vdash}_{\mathcal{I}} \mathcal{E}'_2$.

Формула Φ *является* согласованной с эквивалентностью, $\equiv$ *если для любых состояний* $\mathcal{E}_1, \mathcal{E}_2 \subseteq \mathbf{B}^e$ *из* $\mathcal{E}_1 \Leftrightarrow \mathcal{E}_2$ *следует, что* $\mathcal{E}_1 \models \Phi$ *эквивалентно* $\mathcal{E}_2 \models \Phi$.

Согласованность с информационной эквивалентностью является вполне разумным ограничением, соответствующим сути дела во многих приложениях. Скажем, в нашем примере со строительной фирмой база данных контрактов реализуется посредством предиката $contract(Number, Customer, Sum)$. При этом естественно считать эквивалентными все факты вида $contract(n_i, c_i, s_i)$, отличающиеся только именами заказчиков c_i, поскольку эти имена никак не влияют на финансовое состояние и поведение фирмы.

Клас всех логических программ по вычислительной силе эквивалентен классу машин Тьюринга. Поэтому большинство естественных алгоритмических проблем, связанных с преобразованиями состояний БД неразрешимо. Тем не менее, можно наложить на логические программы некоторые ограничения, часто выполняющиеся в приложениях, которые обеспечат разрешимость ряда проблем. Одим из наиболее реалистических ограничений этого вида является свойство стратифицируемости, аналогичное в некотором смысле [3].

Определение 3. *Пусть* $\mathcal{P}$ *- логическая программа. Скажем, что предикат* p ссылается на *предикат* q, *если в* $\mathcal{P}$ *есть клауза* p *с вызовом* q *в ее теле. Пусть отношение* "зависит от" *является рефлексивным и транзитивным замыканием лтношения "ссылается на". Максимальые сисльно связные*

компоненты графа отношения "зависит от"назовем кликами. $\mathcal{P}$ *называется* динамически стратифицированной (d-стратифицированной), *если для всякой ее клаузы*

$$p(\bar{t}) :- p_1(\bar{t_1}), \ldots, p_i(\bar{t_i}), q(\bar{u}), p_{i+1}(\bar{t}_{i+1}), \ldots, p_r(\bar{t_r}),$$

в которой q *входит в клику* p, *все предикатыs* $p_1, \ldots, p_i, p_{i+1}, \ldots, p_r$ *являются* стационарными, *т.е. не зависят от элементарных модификаций.*

Смысл этого определения состоит в том, что модификации БД могут проводится лишь на тех шагах, на которых меняется клика вызываемого предиката. Следующее определение задает некоторую классификацию логических программ по виду их клауз.

Определение 4. *Логическая программа* $\mathcal{P}$ *является* стационарной, *если все ее предикаты стационарны.* $\mathcal{P}$ *является* позитивной, *если в ней нет отрицаний. Она называется* базисной, *если все ее клаузы базисные,т.е. не содержат переменных. Она является* плоской, *если все термы в ее клаузах являются переменными или константами. Мы назовем* $\mathcal{P}$ расширяющей, *если в ее клаузах не используется модификация delete/1. Программу* $\mathcal{P}$ *назовем* продукционной, *если она задает единственный интенсиональный предикат* $q/0$ *и все ее клаузы являются* продукциями, *т.е. имеют вид* $q :- Con_1, \ldots, Con_k, Act_1, \ldots, Act_m$, *где каждое* Con_i *является экстенсиональным литералом, а каждое* Act_j *является элементарной модификацией.*

Термин "продукция"позаимствовани из ИИ. Наша клауза $q :- Con_1, \ldots, Con_k, Act_1, \ldots, Act_m$ соответсвует там условному правилу преобразования вида $CONDITION \Rightarrow ACTION$, где $CONDITION$ - это конъюнкция литералов Con_i, а $ACTION$ - это последовательность модификаций $Act_1, \ldots, Act_m$. Правила такого вида использвались, в частности, в языке транзакций DL в [1], хотя и в программах с несколько иной семантикой.

4 Поведение динамических дедуктивных БД

Дедуктивные базы данных (ДБД), которые мы рассматриваем, соответствуют их определению в работе [16]. Они включают интенсиональную логическую программу с модификациями, заранее определнный набор целей, реализующих преобразования состояний БД и запросы, ограничения целостности, заданные логической формулой, и некоторое отношение эквивалентности на данных. Для описания поведения динамических ДБД во внешней среде, влияющей на состояния БД, мы вводим новое понятие *траектории* и определяем свойства траекторий, отражающие устойчивое поведение.

Определение 5. Динамическая дедуктивная база данных(динамическая ДБД) *в языке* **L** - *это система*

$$\mathcal{B} =< \mathcal{I} \cup \{ :- G_1, \ldots, :- G_n \}, \ \Phi, \ \equiv_i >$$

где:

- $\mathcal{I}$ - *логтческая программа с модификациями в* $\mathbf{L}$,

- *все цели* G_i, $i = 1, \ldots, n$, *являются либо базисными, либо стационарными,*

- Φ - *некоторая формула 1-го порядка в экстенсиональной сигнатуре, задающая* ограничения целостности (IC),

- $\equiv_i$ - *эквивалентность на* $\mathbf{B}^e$ *(мы называем ее* эквивалентностью данных), *и логические программы* $\mathcal{I} \cup \{ :- G_i \}$, $i = 1, \ldots, n$, *и IC* Φ *согласованы с* $\equiv_i$.

Базисные цели назовем модификациями, *а стационарные* - запросами.

Теперь приведем основное определение этой работы. Его ключевым понятием является понятие траектории с возмущениями, которое отражает реактивное поведение динамической ДДБ во взаимодействии с ее средой.

Определение 6. *Для любых двух непересекающихся множеств* $\mathcal{D}^+, \mathcal{D}^- \subseteq \mathbf{B}^e$, *определим* $(\mathcal{D}^+, \mathcal{D}^-)$-возмущение *как отношение* $\mathcal{E}_1 \stackrel{\mathcal{D}^+, \mathcal{D}^-}{\longrightarrow} \mathcal{E}_2$ *на состояниях БД такое, что* $\mathcal{E}_1, \mathcal{E}_2 \subseteq \mathbf{B}^e$ *и* $\mathcal{E}_2 = (\mathcal{E}_1 \cup \mathcal{D}^+) \setminus \mathcal{D}^-$.

Пусть $\mathcal{B} =< \mathcal{I} \cup \{ :- G_1, \ldots, :- G_n \}, \ \Phi, \ \equiv_i >$ - *динамическая ДДБ. Мы определим* траекторию $\mathcal{B}$ *как конечную или бесконечную последовательность вида*

$$\omega : \ \mathcal{E}_0 \stackrel{G_{i_1}}{\vdash_{\mathcal{I}}} \mathcal{E}_1^* \stackrel{\mathcal{D}_1^+, \mathcal{D}_1^-}{\longrightarrow} \mathcal{E}_1 \stackrel{G_{i_2}}{\vdash_{\mathcal{I}}} \mathcal{E}_2^* \stackrel{\mathcal{D}_2^+, \mathcal{D}_2^-}{\longrightarrow} \mathcal{E}_2 \ \ldots \ .$$

Мы называем такие траектории (начинающиеся с модификаций) м-траекториями. *Двойственные траектории, начинающиеся с возмущений, мы называем* в-траекториями. *Последовательность пар* $d = (\mathcal{D}_1^+, \mathcal{D}_1^-), (\mathcal{D}_2^+, \mathcal{D}_2^-), \ldots$ *называется* возмущением *траектории* ω. *Пусть* $\delta =< \mathcal{D}^+, \mathcal{D}^- >$ *с конечными* $\mathcal{D}^+, \mathcal{D}^- \subseteq \mathbf{B}^e$. *Тогда* ω *является* $\delta-$ограниченной (строго $\delta-$ограниченной,) *если для всех* $k \geq 1$ $\mathcal{D}_k^+ \Lessdot_i \mathcal{D}^+$ $\mathcal{D}_k^- \Lessdot_i \mathcal{D}^-$ *(соответственно,* $\mathcal{D}_k^+ \subseteq \mathcal{D}^+$ *and* $\mathcal{D}_k^- \subseteq \mathcal{D}^-$*). M-траектория (в-траектория) является* стабильной *(соответственно,* гомеостатичной,*) если каждое ее состояние* $\mathcal{E}_i$, $i = 0, 1, 2, \ldots$ *удовлетворяет IC* Φ, *т.е.* $\mathcal{E}_i \models \Phi$. *Две траектории* эквивалентны, *если их соотвественные (т.е. имеющие одинаковые номера) состояния БД являются эквивалентными.*

Стабильность траектории означает, что все нарушения IC, вызванные модификациями ДДБ, компенсируются на ней соответствующими возмущениями. Двойственное понятие гомеостатичности означает, что все нарушения IC, вызванные возмущениями, компенсируются соответствующими модификациями ДДБ. Заметим, что всегда можно было бы компенсировать нарушения IC за счет достаточно больших возмущений или, в двойственном случае, за счет достаточно сложных модификаций. Поэтому, чтобы получить адекватное определение устойчивого поведения, требуется разумно ограничить компенсирующие

или компенсируемые возмущения. Приведенное ниже определение содержит примеры понятий релятивизированного устойчивого поведения, основанные на таких ограничениях.

Определение 7. *Пусть даны* $\mathcal{B}$ - *ДДБ и* $\delta =< \mathcal{D}^+, \mathcal{D}^- >$ *с конечными* $\mathcal{D}^+, \mathcal{D}^- \subseteq \mathbf{B}^e$.

$\mathcal{B}$ *является* $\delta - \exists\exists$-стабильной в состоянии БД $\mathcal{E}$, *если у нее есть бесконечная* δ-*ограниченная стабильная траектория, начинающаяся в* $\mathcal{E}$.

$\mathcal{B}$ *является* $\delta - \forall\exists$-гомеостатичной в состоянии БД $\mathcal{E}$, *если в дереве всех ее* δ-*ограниченных в-траекторий, начинающихся в* $\mathcal{E}$ *имеется* $\forall\exists$-*поддерево, в котором все ветви являются бесконечными гомеостатичными траекториями.*

Состояние БД $\mathcal{E}$ *назовем* $\delta-$перспективным *для* $\mathcal{B}$, *если существует конечная* δ-*ограниченная траектория, начинающаяся в* $\mathcal{E}$ *и заканчивающаяся в некотором состоянии БД, удовлетворяющем IC* Ф. *Мы назовем состояние* $\mathcal{E}$ перспективным, *если оно является* δ-*перспективным для пустого* δ.

Мы не приводим здесь других вариантов стабильности и гомеостатичности. Даже определение $\delta - \forall\exists$-гомеостатичности включено в эту работу только для иллюстрации основных применяемых нами подходов. Представленные в ней результаты относятся к перспективности и $\exists\exists$-стабильности.

Пример 1. Рассмотрим игрушечную ДДБ $\mathcal{B}_1$, описывающую состояние здоровья с помощью трех пропозициональных экстенсиональных предикатов $healthy, ill$ и $medicine$ (последний задает доступность необходимого лекарства). Интенсиональная логическая программа состоит из двух модификаций:

$upd_1 :- ill, medicine, delete(ill), delete(medicine), insert(healthy)$.

$upd_2 :- healthy, change(healthy, ill)$.

The IC ϕ_1 describes satisfactory states:

$healthy \vee (ill \;\&\; medicine)$.

Эта ДДБ является $(\{medicine\}, \emptyset) - \exists\exists$-стабтльной в любом состоянии БД, удовлетворяющим IC. Действительно, если состояние БД содержит $healthy$, то можно применить модификацию upd_2 . Это может нарушить IC, если в состоянии нет факта $medicine$. Однако, IC можно восстановить, применив непустое возмущение. Если же в состоянии имеются ill и $medicine$, то применима первая модификация upd_1 и возникнет состояние, содержащее $healthy$. Отметим также, что ни одно из состояний БД, в котором нарушеноIC не является перспективным для $\mathcal{B}_1$. Однако, если заменить ограничение целостности ϕ_1 на следующее ϕ_2 :

$healthy \vee \neg ill$.

то состояние $\{ill, medicine\}$ не удовлетворяет новому IC ϕ_2, хотя оно перспективно.

Пример 2. Расширим предыдущий пример, включив две различные болезни (ds_1, ds_2) и соответствующие им лекарства (mdc_1, mdc_2) и пусть в результате

лечения этих болезней возникает иммунитет (imm_1, imm_2). Новая ДДБ $\mathcal{B}_2$ имеет следующую интенсиональную логическую программу

$upd_{1i} :- ds_i, mdc_i, delete(ds_i), delete(mdc_i), insert(imm_i).$ $(i = 1, 2)$,

$upd_2 :- \neg imm_1, \neg imm_2, \neg ds_1, \neg ds_2, insert(ds_1), insert(ds_2).$

$upd_{3i} :- imm_i, delete(imm_i).$ $(i = 1, 2)$

и условие допустимости IC:

$(\neg ds_1 \vee mdc_1)$ & $(\neg ds_2 \vee mdc_2)$.

ДДБ $\mathcal{B}_2$ является $\delta - \exists\exists$-стабильной для $\delta = (\{mdc_1, mdc_2\}, \emptyset)$ во всех состояниях, удовлетворяющих IC. Это достаточно очевидно, потому что в каждом состоянии БД, удовлетворяющем IC, применима хотя бы одна модификация и, кроме того, максимальное возмущение переводит любое состояние в состояние, удовлетворяющее IC. Однако, для $\delta_1 = (\{mdc_1\}, \emptyset)$ $\mathcal{B}_2$ не является $\delta_1 - \exists\exists$-стабильной ни в каком состоянии. Причина этого в том, что всякая бесконечная траектория должна содержать применение модификации upd_2, которая приводит к состоянию, включающему обе болезни. Тогда для восстановления IC необходим факт mdc_2 , которого нет в δ_1.

Алгоритмические проблемы, которые мы рассматриваем в этой работе включают задачу выполнимости IC на состоянии, т.е. задачу проверки истинности формулы 1-го порядка на конечной модели. Эта задача, очевидно, разрешима в полиномиальной памяти. С другой стороны, имеются классы приложений с весьма протыми ограничениями целостности (экспертные системы, планирование и др.). Поэтому мы будем рассматривать ниже классы ДБД, чьи ограничения целостности Φ удовлетворяют одному из следующих условий:

(IC0) Φ - базисная формула, не содержащая отрицаний (монотонная);

(IC1) Φ - базисная;

(IC2) Φ - экзистенциально замкнутая формула в пренксной форме .

Сложность проверки истинности IC при этих ограничениях хорошо известна:

Предложение

(1) Задачу проверки выполнимости IC на состоянии БД $\mathcal{E}$ для IC1-ограничений (IC0-ограничений) Φ можно решить за время, линейное от $|\Phi| + |\mathcal{E}|$;

(2) Задачу проверки выполнимости IC на состоянии БД $\mathcal{E}$ для IC2-ограничений Φ является NP-полной;

(3) Задачу проверки выполнимости IC на состоянии БД $\mathcal{E}$ для произвольных ограничений целостности Φ является $PSPACE$-полной.

После того, как мы расклассифицировали ограничения целостности, мы наложим некоторые работающие условия на интенсиональные части ДБД и проанализируем их влияние на вычислительную сложность свойств их поведения. Через $PROD$ обозначим множество всех ДБД с продукционными интенсиональными программами $\mathcal{I}$. Внутри этого класса выделим несколько подклас-

сов:

- $PROF$ - подкласс всех ДБД из $PROD$ с плоскими интенсиональными частями $\mathcal{I}$;

- $PROG$ - подкласс всех ДБД из $PROD$ с базисными интенсиональными частями;

- $PROG^-$ - подкласс всех ДБД из $PROG$ с расширяющими интенсиональными частями;

- $PROG^+$ - подкласс всех ДБД из $PROG^-$ с позитивными интенсиональными частями (т.е. интенсиональные части программ из $PROG^+$ не содержат отрицаний и удалений $delete/1$).

ДБД $\mathcal{B}_1$ в примере 1 принадлежит классу $PROG^-$, а $\mathcal{B}_2$ в примере 2 входит в $PROG$. Во всех указанных классах можно найти много важных приложений. Например, электронные таблицы можно непосредственно представить как ДБД из класса $PROG$ с весьма простыми IC1-ограничениями. Так называемые продукционные экспертные системы попадают в класс $PROF$, а некоторые из них даже в класс $PROG^-$.

Кроме подклассов продукционных ДБД, мы рассмотрим два класса ДБД с д-стратифицированной рекурсией:

- **GDS** - класс всех ДБД с д-стратифицированными и базисными интенсиональными частями;

- **FDS** - класс всех ДБД с д-стратифицированными и плоскими интенсиональными частями.

Из этих определений непосредственно следует, что $PROG \subset GDS$ и $PROF \subset FDS$.

Следующая техническая лемма позволяет во многих случаях вместо пространства всех δ-ограниченных траекторий рассматривать подпространство строго δ-ограниченных траекторий.

Лемма 1.
Пусть $\mathcal{B}$ - ДБД с базисной (плоской) интенсиональной частью, $\mathcal{E}$ - некоторое состояние БД и $\delta =< \mathcal{D}^+, \mathcal{D}^- >$, где $\mathcal{D}^+, \mathcal{D}^- \subseteq \mathbf{B}^e$ конечные множества. Тогда существует конечное $\tilde{\delta} =< \tilde{\mathcal{D}}^+, \tilde{\mathcal{D}}^- >$, такие что для каждой $\delta-$ограниченной траектории ω ДБД $\mathcal{B}$, начинающейся в $\mathcal{E}$, существует строго $\tilde{\delta}$-ограниченная эквивалентная траектория $\tilde{\omega}$ ДБД $\mathcal{B}$, начинающаяся в $\mathcal{E}$. При этом, если $\mathcal{B}$ базисная, то $\mid \tilde{\delta} \mid = O(\mid \mathcal{B} \mid + \mid \mathcal{E} \mid + \mid \delta \mid)$, а если $\mathcal{B}$ плоская, то $\mid \tilde{\delta} \mid \leq 2^{pol(|\mathcal{B}|+|\mathcal{E}|+|\delta|)}$.

Доказательство. Предположим $\mathcal{I}$ базисная. Тогда положим $\tilde{\mathcal{D}}^+ = \mathcal{D}^+$ и $\tilde{\mathcal{D}}^- = \{a \in \mathbf{B}^e \mid a\ \mathcal{I}, \delta\ \mathcal{E},\ a \not\leqslant_i \mathcal{D}^-\}$. Рассмотрим некоторую δ-ограниченную траекторию

$$\omega:\ \mathcal{E}_0 \overset{G_{i_1}}{\vdash_{\mathcal{I}}} \mathcal{E}_1^* \overset{\mathcal{D}_1^+,\mathcal{D}_1^-}{\longrightarrow} \mathcal{E}_1 \ldots \mathcal{E}_{n-1} \overset{G_{i_n}}{\vdash_{\mathcal{I}}} \mathcal{E}_n^* \overset{\mathcal{D}_n^+,\mathcal{D}_n^-}{\longrightarrow} \mathcal{E}_n \ \ldots\ ,$$

где $\mathcal{E}_0 = \mathcal{E}$. Мы построим $\tilde{\omega}$ индукцией пол n. Первое состояние БД в $\tilde{\omega}$ $\tilde{\mathcal{E}}_0 = \mathcal{E}$. Предположим, что $\mathcal{E}_{n-1} \Leftrightarrows_i \tilde{\mathcal{E}}_{n-1}$. Так как $\mathcal{I} \cup \{ :- G_{i_n}\}$ согласована с $\equiv_i$, то существует такое состояние $\tilde{\mathcal{E}}_n$, что $\mathcal{E}_n \Leftrightarrows_i \tilde{\mathcal{E}}_n$ и $\tilde{\mathcal{E}}_{n-1} \overset{G_{i_n}}{\vdash_{\mathcal{I}}} \tilde{\mathcal{E}}_n$. Следовательно мы можем продолжить $\tilde{\omega}$ с помощью этой модификации. Предположим, что $\mathcal{E}_n^* \Leftrightarrows \tilde{\mathcal{E}}_n^*$. Положим тогда $\tilde{\mathcal{D}}_n^+ = \{a \in \mathcal{D}^+ \mid \exists b(b \in \mathcal{D}_n^+ \ \& \ a \equiv_i b)\}$ и $\tilde{\mathcal{D}}_n^- = \{a \in \tilde{\mathcal{E}}_n^* \mid \neg\exists c \in \mathcal{E}_n \ (\ c \equiv_i a\ )\}$. Другими словами, мы устраняем те атомы из $\tilde{\mathcal{E}}_n^*$, чьи классы эквивалентности удаляются из $\mathcal{E}_n^*$ в результате вычитания $\mathcal{D}_n^-$. Ясно, что $\mathcal{E}_n \Leftrightarrows_i \tilde{\mathcal{E}}_n$ и $\tilde{\mathcal{D}}_n^+ \subseteq \tilde{\mathcal{D}}^+$. Покажем, что это возмущение строгое, т.е. $\tilde{\mathcal{D}}_n^- \subseteq \tilde{\mathcal{D}}^-$. Let $a \in \tilde{\mathcal{D}}_n^-$. Тогда $a \in \tilde{\mathcal{E}}_n^*$. Поскольку $\mathcal{I}$ базисная, то только атомы из $\mathcal{I}$, $\tilde{\mathcal{D}}^+$ или $\mathcal{E}_0$ могут появляться в состояниях строго $\tilde{\delta}$-ограниченной траектории $\tilde{\omega}$. Следовательно, a входит в $\mathcal{I}$, $\tilde{\mathcal{D}}^+$ или $\mathcal{E}_0$. С другой стороны из эквивалентности состояний БД $\tilde{\mathcal{E}}_n^*$ и $\mathcal{E}_n^*$ следует, что существует атом $b \in \mathcal{D}_n^- \cap \mathcal{E}_n^*$, эквивалентный атому a. Тогда существует такое $c \in \mathcal{D}^-$, что $c \equiv_i b \equiv_i a$. Следовательно, из определения $\tilde{\mathcal{D}}^-$ мы получаем $a \in \mathcal{D}^-$.

В случае плоской $\mathcal{I}$ положим $\tilde{\mathcal{D}}^+ = \mathcal{D}^+$ как и выше и $\tilde{\mathcal{D}}^- = \{p(c_1, ..., c_m) \in \mathbf{B}^e \mid c_1, ..., c_m \ occur \ in \ \mathcal{I}, \ \delta \ \ \mathcal{E}, \ \ p(c_1, ..., c_m) \Leftarrow_i \mathcal{D}^-\}$. Доказательство корректности этого выбора аналогично доказательству для базисного случая.

Обе верхние оценки для $\tilde{\delta}$ следуют непосредственно из определений. □

Лемма 1 будет далее использована во всех доказательствах верхних оценок. В них мы определяются некоторые недетерминированные алгоритмы, угадывающие δ-ограниченные траектории с определенными свойствами (например, перспективности, стабильности, гомеостатичности и др.). Применение этой леммы гарантирует, что вместо произвольных траекторий достаточно находить $\tilde{\delta}$-ограниченные траектории с теми же свойствами. При этом входная тройка $\mathcal{B}, \delta, \mathcal{E}$ конструктивно заменяется новым входом $\mathcal{B}, \tilde{\delta}, \mathcal{E}$. Однако, как показывает доказательство леммы 1, $\tilde{\delta}$ строится за полиномиальное время в базисном случае и за экспоненциальное время в плоском случае. Таким образом, благодаря этой лемме, не ограничивая общности, мы можем описывать недетерминированные алгоритмы, угадывающие и проверяющие лишь строго $\delta-$ограниченные траектории.

5 Сложность перспективности

Ограниченная перспективность гарантирует достижимость некоторого состояния, удовлетворяющего IC, из данного состояния посредством некоторой конечной $\delta-$ограниченной траектории. Пусть $\mathbf{P}$ - некоторый класс ДБД. Проблема перспективности для этого класса - это проблема вхождения троек вида $(\mathcal{B}, \delta, \mathcal{E})$ во множество

$$PROMISE(\mathbf{P}) = \{(\mathcal{B}, \delta, \mathcal{E}) \mid \mathcal{E} \ -\delta- \ \ \mathcal{B} \in \mathbf{P}\}.$$

Следующая теорема показывает, что сложность проблемы перспективности для классов продукционных ДБД непосредственно зависит от наличия различных источников немонотонности в их интенсиональных частях.

Теорема 1.

(1) Проблема $PROMISE(PROG^+)$ *с IC0-ограничениями решается за линейное время, а с IC1-ограничениями является* NP*-полной.*

(2) Проблема $PROMISE(PROG^-)$ *с IC2-ограничениями принадлежит* NP*, если* $\mathcal{D}^- = \emptyset$*, и является* $NP-$*трудной в этом классе с IC0-ограничениями.*

(3) Проблема $PROMISE(PROG^-)$ *принадлежит* $PSPACE$ *и является* $PSPACE-$*трудной в этом классе с IC0-ограничениями.*

(4) Проблема $PROMISE(PROF)$ *принадлежит* $SPACE(2^{poly})$ *и является* $SPACE(2^{poly})$*-трудной в этом классе с IC0-ограничениями.*

(5) Проблема $PROMISE(PROD)$ *неразрешима.*

Доказательство. (1) Поскольку ни в условиях продукций, ни в IC, удовлетворяющих IC0-критерию, нет отрицаний, то $(\mathcal{B}, < D^+, D^- >, \mathcal{E}) \in PROMISE(PROG^+)$ тогда и только тогда, когда $(\mathcal{B}, < \emptyset, \emptyset >, \mathcal{E} \cup D^+) \in PROMISE(PROG^+)$. Кроме того, чтобы достичь допустимое состояние достаточно применять каждую продукцию, как только ее условие станет выполнимо, и использовать ее лишь однажды. Наконец, поскольку $\mathcal{B}$ - базисная, то всякий атом встречающийся в состояниях траектории обязан присутствовать во входных данных. Следовательно, обычный алгоритм прямого поиска обеспечивает в этом случае квадратичную верхнюю оценку времени. В [11] описан метод реализации этого алгоритма за линейное время.

Верхняя оценка. Предположим теперь, что IC является базисной, но не монотонной. По лемме 1 мы можем ограничиться поиском строго $\delta-$ограниченных траекторий. Заметим далее, что если такая успешная траектория существует, то существует такая же траектория длины не больше, чем $2 \times (\ )$. Дело в том, что всегда достаточно рассматривать траектории

$$\omega: \ \mathcal{E}_0 \ \overset{G_{i_1}}{\vdash_{\mathcal{I}}} \ \mathcal{E}_1^* \ \overset{\mathcal{D}_1^+, \mathcal{D}_1^-}{\longrightarrow} \ \mathcal{E}_1, ..., \ \overset{G_{i_n}}{\vdash_{\mathcal{I}}} \ \mathcal{E}_2^* \ \overset{\mathcal{D}_n^+, \mathcal{D}_n^-}{\longrightarrow} \ \mathcal{E}_n,$$

в которых $D_j^- = \emptyset$ для всех $j < n$. А среди таких траекторий достаточно ограничиться траекториями, в которых ни одна продукция не применяется дважды. Это сразу же дает недетерминированный полиномиальный алгоритм.

Нижняя оценка. Пусть IC - это произвольная пропозициональная формула φ с пропозициональными переменными $p_1, ..., p_n$. Пусть интенсиональная часть $\mathcal{I}$ ДБД $\mathcal{B}$ состоит из базисных продукций $q :- insert(p_i)$ для всех $1 \leq i \leq n$. Тогда легко видеть, что $(\mathcal{B}, < \emptyset, \emptyset >, \emptyset) \in PROMISE(PROG^+)$ тогда и только тогда, когда $\varphi \in SAT$. Таким образом $SAT \leq_{pol} PROMISE(PROG^+)$.

(2) *Верхняяя оценка.* При условии, что $\mathcal{D}^- = \emptyset$ как и в предыдущем утверждении для случая IC1-ограничений, можно доказать, что достаточно рассматривать при поиске траектории, в которых каждая продукция применяется не

более одного раза.

Нижняя оценка следует из оценки утверждения (1).

(3) *Верхняя оценка.* Недетерминированный алгоритм, работающий в полиномиальной памяти, получается непосредственно. Тогда требуемый результат следует из известной теоремы Сэвича о том, что $NPSPACE = PSPACE$ [23].

Нижняя оценка. Мы покажем, что произвольный язык, распознаваемый некоторой машиной Тьюринга $\mathcal{M}$ в памяти, ограниченной полиномом p сводится за полиномиальное время к $PROMISE(PROG^-)$. Пусть q_j $(0 \le j \le k)$ - состояния $\mathcal{M}$ (q_1 - начальное состояние, q_0 - заключительное допускающее состояние). Пусть a_j $(0 \le j \le l)$ - символы алфавита ленты $\mathcal{M}$ (a_0 - пустой символ). Пусть $x = a_{i_1}, ..., a_{i_n}$ - входное слово и $N = p(n)$ - граница памяти. Экстенсиональная сигнатура $\mathcal{B}$ состоит из следующих 0-арных предикатов:

- q_j, q'_j $(0 \le j \le k)$;
- $a_{sj}, a'_{sj} (1 \le s \le N, 0 \le j \le l)$: в ячейке s записан символ a_j;
- h_s, h'_s $(1 \le s \le N)$: головка $\mathcal{M}$ находится в ячейке s.

Начальное состояние БД $\mathcal{E}_0 = \{q_1, h_1, a_{1i_1}, ..., a_{ni_n}, a_{(n+1)0}, ..., a_{N0}\}$ кодирует начальную конфигурацию $\mathcal{M}$.

Для каждой команды $q_j a_m \to q_u a_v S$, где $S \in \{-1, 0, 1\}$, - интенсиональная логическая программа $\mathcal{I}$ содержит N продукций вида:

$$(i) \quad g \ :- \ q_j, h_s, a_{sm}, \bigwedge_{\alpha \ne j} \neg q_\alpha, \bigwedge_{\beta \ne s} \neg h_\beta, \bigwedge_{\gamma \ne m} \neg a_{s\gamma}, \bigwedge_{\lambda} \neg q'_\lambda, \bigwedge_{\mu} \neg h'_\mu, \bigwedge_{s'} \bigwedge_{\nu} \neg a'_{s'\nu},$$
$$insert(q'_u), insert(h'_{s+S}), insert(a'_{sv}). \qquad (1 \le s \le N).$$

Кроме того, $\mathcal{I}$ содержит следующие $(k+1)(l+1)N^2$ продукций:

$$(ii) \quad g \ :- \ q'_j, h'_t, a'_{sm}, \bigwedge_{\alpha \ne j} \neg q'_\alpha, \bigwedge_{\beta \ne t} \neg h'_\beta, \bigwedge_{\gamma} \neg a_{s\gamma}, \bigwedge_{\lambda} \neg q_\lambda, \bigwedge_{\mu} \neg h_\mu, \bigwedge_{\tau \ne m} \neg a'_{s\tau},$$
$$\bigwedge_{s' \ne s} \bigwedge_{\nu} \neg a'_{s'\nu}, insert(q_j), insert(h_t), insert(a_{sm}). \qquad (1 \le s, t \le N).$$

В качестве ограничения для возмущений возьмем

$$\delta \ = \ < \emptyset, \{ \ q_j, q'_j \ \mid \ 0 \le j \le k\} \cup \{ \ h_s, h'_s \ \mid \ 1 \le s \le N\}$$
$$\cup \{ \ a_{sj}, a'_{sj} \ \mid \ 1 \le s \le N, \ 0 \le j \le l\} >$$

(т.е. возмущения ничего не могут добавлять к состояниям, но могут удалить произвольное подмножество фактов).

Наконец, пусть IC $\Phi = q_0$ и $\mathcal{B} =< \mathcal{I} \cup \{ :- \ g\}, \Phi, = >$.

Тогда теорема следует из

Предложения. *$\mathcal{M}$ допускает x в памяти N* $\Leftrightarrow$ $(\mathcal{B}, \delta, \mathcal{E}_0) \in PROMISE(PROG^-)$.

Пусть K - конфигурация $\mathcal{M}$, а $\mathcal{E}$ - состояние БД. Скажем, что $\mathcal{E}$ *соответствует* K, если

- $\mathcal{E}$ содержит один факт q_j и q_j - это состояние $\mathcal{M}$ в конфигурации K;
- $\mathcal{E}$ содержит один факт h_s и s - это номер ячейки, в которой находится головка машины в конфигурации K ;
- для каждого факта $a_{tm} \in \mathcal{E}$ ячейка t в конфигурации K содержит символ a_m;
- в $\mathcal{E}$ нет фактов со штрихами.

Пусть состояние БД $\mathcal{E}$ соответствует конфигурации K, в которой головка расположена в ячейке s. Предположим, что команда $q_j a_m \to q_u a_v S$ переводит K в K'. Этот шаг вычисления моделируется траекторией из 4-х шагов

$$\mathcal{E} \overset{(i)}{\vdash_{\mathcal{I}}} \mathcal{E}_1^* \overset{\emptyset,\mathcal{D}_1^-}{\longrightarrow} \mathcal{E}_1 \overset{(ii)}{\vdash_{\mathcal{I}}} \mathcal{E}_2^* \overset{\emptyset,\mathcal{D}_2^-}{\longrightarrow} \mathcal{E}_2,$$

где (i) и (ii) - это определенные выше продукции, соответствующие применяемой команде, $\mathcal{D}_1^- = \{q_j, h_s, a_{sm},\}$ $\mathcal{D}_2^- = \{q'_u, h'_{s+S}, a'_{sv}.\}$ Тогда легко проверить, что $\mathcal{E}_2$ соответствует K'. Отсюда следует первая половина предложения. Обратно, если некотороя траектория указанного вида из 4-х шагов с некоторыми $\mathcal{D}_1^-, \mathcal{D}_2^-$ начинается в состоянии $\mathcal{E}$, соответствующем конфигурации K, то команда, однозначно определяемая продукцией вида (i), применима к K и переводит ее в конфигурацию, которой соответсвует $\mathcal{E}_2$. Тогда утверждение предложения следует из того, что без ограничения общности можно рассматривать только траектории, разбиваемые на последовательные 4-х шаговые подтраектории указанного вида.

(4) **Верхняя граница.** Пусть $\mathcal{B}$ - это ДБД из $PROF$, $\mathcal{E}$ - состояние БД и $\delta =< \mathcal{D}^+, \mathcal{D}^- >$. Обозначим через p_e число экстенсиональных предикатов, встречающихся в $\mathcal{B}, \delta, \mathcal{E}$, через c число различных констант в этих множествах, и через a_e максимальную "арность"экстенсиональных предикатов. Тогда нетрудно заметить, что не больше чем $N = p_e c^{a_e}$ различных экстенсиональных атомов могут встретиться в состояниях $\delta-$ограниченных траекторий, начинающихся в $\mathcal{E}$. Поэтому N является верхней границей размера состояний на таких траекториях. Искомый недетерминированный алгоритм начинает работать на входе $(\mathcal{B}, \delta, \mathcal{E})$ и угадывает пошагово строго $\delta-$ограниченную траекторию $\mathcal{B}$ длины не превосходящей 2^{N+1}, начинающуюся в $\mathcal{E}$ и заканчивающуюся в некотором состоянии БД, удовлетворяющем IC. По лемме 1 этот алгоритм правильно устанавливает $\delta-$перспективность $\mathcal{E}$. Очевидно, что существует такой полином pol, что описанный алгоритм работает в памяти, не превосходящей $2^{pol(|(\mathcal{B},\delta,\mathcal{E})|)}$.

Нижняя оценка. Зафиксируем произвольную машину Тьюринга $\mathcal{M}$, работающую в памяти, ограниченной $2^{p(n)}$ для некоторого полинома $p(n)$, где n - длина входного слова x , и пусть $N = p(n)+1$. Пусть $\mathcal{M}$ имеет алфавит ленты A , алфавит состояний Q и программу P как в доказательстве нижней

оценки в утверждении (3). Пусть $x = a_{i_1}a_{i_2}...a_{i_n}$. Экстенсиональная сигнатура $\mathcal{P}^e$ будет включать два предиката $a/N+1$ and $h/N+1$. Содержательно, $a(s_1,\ldots,s_N,i)$ означает, что ячейка с двоичным номером $s_1\ldots s_N$ $(s_i \in \{0,1\})$ содержит символ a_i. $h(s_1,\ldots,s_N,k)$ означает, что головка $\mathcal{M}$ находится в ячейке с двоичным номером $s_1\ldots s_N$ и текущим состоянием $\mathcal{M}$ является q_k . Для каждой команды $q_ja_m \to q_ua_vS$ из P интенсиональная логическая программа $\mathcal{I}$ содержит продукции, моделирующие ее применение в любой ячейке от 0-й до 2^N-1-ой. В случае $S=1$ эти продукции имеют вид:

g :- $h(S_1,...,S_{N-1},0,j)$, $a(S_1,...,S_{N-1},0,m)$, $a(S_1,...,S_{N-1},1,X)$,
$change(h(S_1,...,S_{N-1},0,j),h(S_1,...,S_{N-1},1,u))$,
$change(a(S_1,...,S_{N-1},0,m),a(S_1,...,S_{N-1},0,v))$
. . . .
g :- $h(S_1,...,S_j,0,1,...,1,j)$, $a(S_1,...,S_j,0,1,...,1,m)$, $a(S_1,...,S_j,1,0,...,0,X)$,
$change(h(S_1,...,S_r,0,1,...,1,j),h(S_1,...,S_r,1,0,...,0,u))$,
$change(a(S_1,...,S_r,0,1,...,1,m),a(S_1,...,S_r,0,1,...,1,v))$
. . . .
g :- $h(0,1,...,1,j)$, $a(0,1,...,1,m)$, $a(1,0,...,0,X)$,
$change(h(0,1,...,1,j),h(1,0,...,0,u))$,
$change(a(0,1,...,1,m),a(0,1,...,1,v))$

и кроме этих еще продукции вида:

g :- $h(S_1,...,S_{N-1},0,j)$, $a(S_1,...,S_{N-1},0,m)$, $\neg a(S_1,...,S_{N-1},1,X)$,
$change(h(S_1,...,S_{N-1},0,j),h(S_1,...,S_{N-1},1,u))$,
$change(a(S_1,...,S_{N-1},0,m),a(S_1,...,S_{N-1},0,v))$
$insert(a(S_1,...,S_{N-1},1,0))$
. . . .
g :- $h(S_1,...,S_r,0,1,...,1,j)$, $a(S_1,...,S_r,0,1,...,1,m)$, $\neg a(S_1,...,S_r,1,0,...,0,X)$,
$change(h(S_1,...,S_r,0,1,...,1,j),h(S_1,...,S_r,1,0,...,0,u))$,
$change(a(S_1,...,S_r,0,1,...,1,m),a(S_1,...,S_r,0,1,...,1,v))$
$insert(a(S_1,...,S_r,1,0,...,0,0))$
. . . .
g :- $h(0,1,...,1,j)$, $a(0,1,...,1,m)$, $\neg a(1,0,...,0,X)$,
$change(h(0,1,...,1,j),h(1,0,...,0,u))$,
$change(a(0,1,...,1,m),a(0,1,...,1,v))$
$insert(a(1,0,...,0,0))$

для моделирования правого сдвига головки в новую ячейку с пустым символом a_0. Аналогичные продукции включаются и для команд с двумя другими сдвигами. Положим t $\mathcal{B} = <\mathcal{I} \cup \{ :- g \}, h(S_1,\ldots,S_N,1), =>, \delta =<\emptyset,\emptyset>,$ и $\mathcal{E}_x = \{ h(0,\ldots,0,1,0), a(0,\ldots,0,1,i_1),\ldots, a(j_1,\ldots,j_N,i_n) \},$ где $j_1,\ldots,j_N$ - это двоичное представление n . Легко подсчитать, что $|\mathcal{B}| + |\mathcal{E}_x| = O(|$

$\mathcal{M} \mid N^2$). Кроме того, нетрудно проверить, что $\mathcal{M}$ достигает заключительную конфигурацию K_{fin} из начальной конфигурации с x на ленте тогда и только тогда, когда существует допустимое состояние $\mathcal{E}_{K_{fin}}$, содержащее факт вида $h(S_1, \ldots, S_N, 1)$, которое можно достичь из $\mathcal{E}_x$ с помощью применения продукций из $\mathcal{I}$. Это утверждение можно установить индукцией по числу шагов вычисления $\mathcal{M}$ на входе x.

(5) Неразрешимость $PROMISE(PROD)$ доказывается непосредственным сведением к этой проблеме проблемы остановки для машин Минского с двумя счетчиками. □

Следствие *Проблема $PROMISE(PROG)$ принадлежит $PSPACE$, и является $PSPACE-$трудной в этом классе с IC0-ограничениями.*

Отметим, что нижние границы сложности в утверждениях (1),(2) и (4) теоремы 2 имеют место для подпроблем с $\delta =< \emptyset, \emptyset >$. Более того, можно показать, что это справедливо и для приведенного выше следствия. Случай $\delta =< \emptyset, \emptyset >$ интересен сам по себе, поскольку для продукционных ДБД рассматриваемая проблема эквивалентна известной в ИИ задаче планирования для систем типа STRIPS (см. [15, 19, 7]). В [7] имеется детальный сложностной анализ этой задачи.

Классы GDS и FDS существенно шире, чем классы $PROG$ and $PROF$, соответственно. Следующая теорема показывает, что, тем не менее, для них проблема перспективности имеет ту же сложность.

Теорема 2.
(1) Проблема $PROMISE(GDS)$ является $PSPACE$-полной.
(2) Проблема $PROMISE(FDS)$ является $SPACE(2^{poly})$-полной.

Доказателъство. Верхние оценки требуют некоторых дополнительных понятий и лемм. Вычисления логических программ естественным образом представляются их деревьями вычислений. Мы модифицируем ниже определение дерева вычислений, использованное в работе [12] для задания вычислений пролог программ.

Определение 8. *Пусть $\mathcal{I}$ - логическая программа, $R = A_1, ..., A_n$, и c - вычисление этой программы $\mathcal{E}_1 \overset{R}{\vdash}_{\mathcal{I}} \mathcal{E}_2$. Тогда дерево этого вычисления $t = t(c)$ определяется следующим образом. Вершинами t являются пары вида (G, U), где G - одна из подцелей, разрешаемых в c, а U - это подстановка, выбираемая на этом шаге резолюции. Корнем t является пара $v_0 = (R, \varepsilon)$, а ее сыновъями $(A_1, U_1), ..., (A_n, U_n)$. Предположим, вершина $v = (G_1, U)$ дерева t соответствует подцели G_1, разрешаемой в c в состоянии вычисления $(\mathcal{E}, (G_1, ..., G_k), \sigma)$. Если G_1 является элементарной модификацией или атомом с отрицанием, то v - это лист и $U = \varepsilon$. Если G_1 - это*

экстенсиональный или интенсиональный атом, разрешаемый с помощью единичной клаузы (факта) B *из* $\mathcal{I} \cup \mathcal{E}$, *то* v *- это лист и* U *является наиболее общим унификатором* $G_1 \circ \sigma$ *и* B. *И наконец, если* G_1 *- интенсиональный атом, разрешаемый с помощью клаузы* $H :- B_1, \ldots, B_r (r \geq 1)$, *и* θ *- это наиболее общий унификатор* $G_1 \circ \sigma$ *и* H, *то* $U = \theta$ *и у вершины* v *в* t *имеется* r *сыновей:* $(B_1, U_1), ..., (B_r, U_r)$.

С каждой вершиной v *дерева* t *мы связываем ее входное и выходное состояние БД* $\mathcal{E}^{in}(v)$ *и* $\mathcal{E}^{out}(v)$ *и их глобальные контексты* $\sigma^{in}(v), \sigma^{out}(v)$ *следующим образом. Для корня* v_0 *полагаем* $\mathcal{E}^{in}(v_0) = \mathcal{E}_1$, $\mathcal{E}^{out}(v_0) = \mathcal{E}_2$ *и* $\sigma^{in}(v) = \varepsilon$. *Для листьев* v *с отрицательными подцелями -* $\mathcal{E}^{in}(v) = \mathcal{E}^{out}(v)$ *и* $\sigma^{in}(v) = \sigma^{out}(v)$. *Для листьев* v *с элементарными модификациями* $\sigma^{in}(v) = \sigma^{out}(v)$ *и* $\mathcal{E}^{out}(v)$ *получается из* $\mathcal{E}^{in}(v)$ *посредством соответствующей элементарной модификации (вставки или удаления некоторого факта). Для остальных листьев* $v = (G, U)$, $\mathcal{E}^{in}(v) = \mathcal{E}^{out}(v)$ *и* $\sigma^{out}(v) = \sigma^{in}(v) \circ U$. *Для внутренней вершины* $v = (G, U)$ *с* r *сыновьями* $v_1 = (B_1, U_1), ..., v_r = (B_r, U_r)$ *полагаем* $\mathcal{E}^{in}(v_1) = \mathcal{E}^{in}(v)$, $\mathcal{E}^{out}(v) = \mathcal{E}^{out}(v_r)$, $\mathcal{E}^{out}(v_i) = \mathcal{E}^{in}(v_{i+1})$ *для всех* $1 \leq i < r$, $\sigma^{in}(v_1) = \sigma^{in}(v) \circ U$, $\sigma^{out}(v) = \sigma^{out}(v_r)$ *и* $\sigma^{out}(v_i) = \sigma^{in}(v_{i+1})$ *для всех* $1 \leq i < r$. *Для вершины* $v = (G, U)$, *мы назовем литералы* $inst^{in}(v) = G \circ \sigma^{in}(v)$ *и* $inst^{out}(v) = G \circ \sigma^{out}(v)$ in-вариантом *и* out-вариантом of *цели* G в v.

Нетрудно проверить, что приведенные определения входных и выходных состояний БД и глобальных контнстов корректны. Из этго определения, в частности, следует, что $\sigma^{out}(v_0)$, ограниченная переменными цели R является результирующей подстановкой - ответом, вычисляемым c.

Для иллюстрации этих понятий рассмотрим логическую программу $\mathcal{I}$ с клаузами

$s :- e_1(X), a(X)$

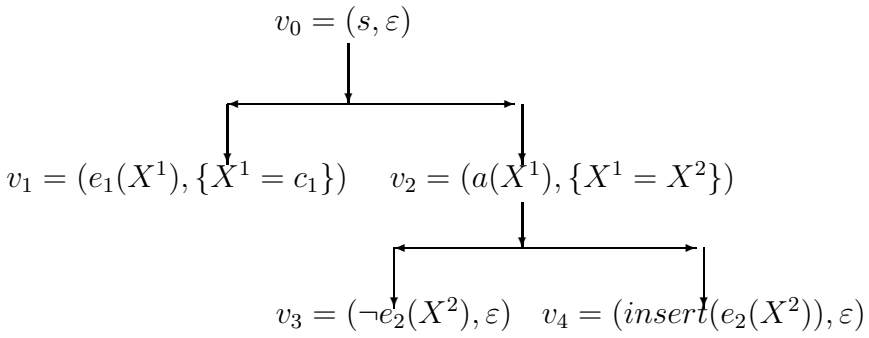
$a(X) :- \neg e_2(X), insert(e2(X))$

$a(X) :- e_2(X), delete(e2(X))$

и с целью $:- s$. Пусть $\mathcal{E}_0 = \{e_1(c_1), e_2(c_2)\}$ и $\mathcal{E}_1 = \{e_1(c_1), e_2(c_2), e_2(c_1)\}$ - состояния БД. Дерево вычисления $\mathcal{E}_0 \overset{s}{\vdash}_{\mathcal{I}} \mathcal{E}_1$ показано на рис. 4.

Определение 9. *Пусть* t *- дерево вычисления, и* $v_1 = (A_1, U_1)$ *and* $v_2 = (A_2, U_2)$ *- вершины* t, *такие что* v_1 *является предком* v_2. *Мы назовем такую пару вершин* сокращаемой, *если* $\mathcal{E}^{in}(v_1) = \mathcal{E}^{in}(v_2)$, $\mathcal{E}^{out}(v_1) = \mathcal{E}^{out}(v_2)$ *и существует взимно однозначное переименование перемеенных* θ, *такое что* $A_1 = A_2 \circ \theta$ *и для каждой переменной* X *in* A_1 $\sigma^{in}(v_1)(X) = \sigma^{in}(v_2)(X \circ \theta)$ *и* $\sigma^{out}(v_1)(X) = \sigma^{out}(v_2)(X \circ \theta)$. *Дерево* t *называется* сокращенным, *если в нем нет ни одной пары сокращаемых вершин.*

Два дерева вычислений t_1, t_2 *с корнями* v_{01} *и* v_{02}, *соответственно,* эквивалентны, *если* $inst^{in}(v_{01}) = inst^{in}(v_{02})$, $inst^{out}(v_{01}) = inst^{out}(v_{02})$, $\mathcal{E}^{in}(v_{01}) = \mathcal{E}^{in}(v_{02})$ *и* $\mathcal{E}^{out}(v_{01}) = \mathcal{E}^{out}(v_{02})$.



.4

Например, в дереве вычислений на рис. 4 $inst^{in}(v_1) = e_1(X^1)$, $inst^{out}(v_1) = e_1(c^1)$, $\mathcal{E}^{in}(v_4) = \mathcal{E}_0$, $\mathcal{E}^{out}(v_4) = \mathcal{E}_1$, $inst^{in}(v_3) = \neg e_2(c_1)$ и $\sigma^{out}(v_4) = \{X^1 = c_1, X^2 = X^1\}$.

Лемма 2. *(Лемма о сокращении). Для всякого дерева вычисления t_1 существует эквивалентное сокращенное дерево вычисления t_2.*

Доказательство проводится индукцией по числу сокращаемых пар в t_1. Пусть v_1, v_2 составляют сокращаемую пару в t_1. Пусть $L(v_1) = A_1 = A_1[\bar{U}_1]$, $L(v_2) = A_2 = A_2[\bar{U}_2]$, $\mathcal{E}^{in}(v_1) = \mathcal{E}^{in}(v_2)$, $\mathcal{E}^{out}(v_1) = \mathcal{E}^{out}(v_2)$, а θ - взаимно однозначное переименование переменных такое, что $A_1 = A_2 \circ \theta$, $(\sigma^{in}(v_2)\lceil\bar{U}_2) \circ \theta = (\sigma^{in}(v_1)\lceil\bar{U}_1)$ и $(\sigma^{out}(v_2)\lceil\bar{U}_2) \circ \theta = (\sigma^{out}(v_1)\lceil\bar{U}_1)$. Из определения сокращаемой пары следует, что $inst^{in}(v_2) \circ \theta = inst^{in}(v_1)$ и $inst^{out}(v_2) \circ \theta = inst^{out}(v_1)$. Для каждой вершины v дерева t обозначим через $t(v)$ поддерево t с корнем v. Тогда дерево t_2', получающееся из t_1 заменой его поддерева $t_1(v_1)$ на дерево $t_1(v_2) \circ \theta$, также является деревом вычисления. Действительно, всякое изменение поддерева $t_1(v_1)$ может повлиять на вершины вне $t_1(v_1)$ только через $\sigma^{out}(v_1)$ или через $\mathcal{E}^{out}(v_1)$, а изменения в вершинах вне поддерева $t_1(v_2)$ могут повлиять на вершины поддерева $t_1(v_2)$ только через $\sigma^{in}(v_2)$ или через $\mathcal{E}^{in}(v_2)$. Но все они совпадают на множестве их общих переменных. Более того, деревья вычислений t_1 и t_2' эквивалентны, потому что по той же причине замена не влияет ни на $inst^{in}(v_0)$ и $inst^{out}(v_0)$, ни на $\mathcal{E}^{in}(v_0)$ и $\mathcal{E}^{out}(v_0)$ для корня v_0 дерева t_1. В результате, t_2' содержит на одну сокращаемую пару меньше чем дерево t_1. Повторяя эту процедуру столько раз, сколько имеется сокращаемых пар в t_1, мы получим требуемое дерево вычисления t_2. □

Лемма 3. *Пусть t - дерево вычисления $\mathcal{E}_1 \overset{G}{\vdash}_{\mathcal{I}} \mathcal{E}_2$, $\mathbf{b}_{\mathcal{I}}$ - максимальная длина тела клаузы в программе $\mathcal{I}$, $\mathbf{l}(t) = max\{\mid inst^{in}(v)\mid, \mid inst^{out}(v)\mid \mid v \in t\}$, и $\mathbf{db}(t) = max\{\mid \mathcal{E}^{in}(v)\mid, \mid \mathcal{E}^{out}(v)\mid \mid v \in t\}$. Тогда вычисление, задаваемое t, можно промоделировать на машине Тьюринга, работающей в памяти $O(max\{\mathbf{db}(t), depth(t) \times \mathbf{b}_{\mathcal{I}} \times \mathbf{l}(t)\})$.*

Доказательство. Машина $\mathcal{M}$ строится по программе $\mathcal{I}$ обычном образом. В каждый момент состояние текущей цели записано на стеке $\mathcal{M}$ (с текущей подцелью на вершине стека). Очередной шаг либо удаляет эту подцель со стека, либо заменяет ее на последовательность подцелей из тела используемой на этом шаге клаузы. Таким образом, в каждый момент длина стека не превышает длины текущей цели. Используемое правило вычисления гарантирует, что длина целей никогда не превзойдет длины пути в дереве t из корня в лист, умноженной на $\mathbf{b}_{\mathcal{I}}$. Следовательно, размер стека не превосходит это значение, умноженное на $\mathbf{l}_{\mathcal{I}}(t)$. А размер ленты, на которой записано состояние БД никогда не превышает $\mathbf{db}_{\mathcal{I}}(t)$, так как все опреации на этой ленте выполняются в памяти, занятой текущим состоянием БД. □

Возвращаясь к доказательству верхних оценок, отметим, что на каждой ветви сокращенного дерева вычисления все вершины попарно не эквивалентны. Следовательно, по лемме о сокращении, глубину дерева вычисления для **GDS** можно ограничить величиной $|\mathcal{I}|^2$. Для **FDS** эту глубину можно ограничить величиной $|\mathcal{I}|pc^a$, где p - число интенсиональных предикатов в $\mathcal{I}$, c - число ее констант и a - максимальная арность ее интенсиональных предикатов. Следовательно, по лемме 3 получаются требуемые верхние оценки сложности для отдельных шагов-модификаций. Память же, требуемая для моделирования шагов-возмущений, не превосходит размеров состояний БД. Таким образом, недетерминированный алгоритм, угадывающий строго $\delta-$ограниченную траекторию, заканчивающуюся в допустимом состоянии, удовлетворяет условиям теоремы.

Нижние оценки следуют из утверждений (3) и (4) теоремы 1. □

6 Сложность стабильности

В этом разделе мы изучим сложность проблемы $\delta-\exists\exists$–стабильности. Для каждого рассматриваемого класса ДБД $\mathbf{P}$ будет установлена сложность следующей проблемы:

$$STABLE(\mathbf{P}) = \{(\mathcal{B}, \delta, \mathcal{E}) \mid \mathcal{B} \in \mathbf{P} \quad \delta-\exists\exists- \quad \mathcal{E}\}$$

Следующая теорема содержит оценки сложности для продукционных систем.

Теорема 3.

(1) Проблема $STABLE(PROG^+)$ для IC1-ограничений является NP-полной.

(2) Проблема $STABLE(PROG^-)$ для IC2-ограничений принадлежит NP, а для IC0-ограничений является NP-трудной.

(3) Проблема $STABLE(PROG)$ *является* $PSPACE$*-полной.*
(4) Проблема $STABLE(PROF)$ *является* $SPACE(2^{poly})$*-полной.*
(5) Проблема $STABLE(PROD)$ *неразрешима.*

Доказательство. (1) *Нижняя оценка.* Пусть φ - произвольная пропозициональная формула с пропозициональными переменными $p_1, ..., p_n$. Рассмотрим ДБД $\mathcal{B}$ с интенсиональной частью $\mathcal{I}$, состоящей из базисной клаузы $q :- insert(p_0)$. В качестве IC возьмем формулу $\Phi = \varphi \vee \neg p_0$. Тогда легко видеть, чтот $(\mathcal{B}, < \{p_1, ..., p_n\}, \emptyset >, \emptyset) \in STABLE(PROG^+) \leftrightarrow \varphi \in SAT$. Следовательно, $SAT \leq_{pol} STABLE(PROG^+)$.

(2)*Верхняя оценка* для $STABLE(PROG^-)$ основана на следующем утверждении.

Лемма 4. *Для любой ДБД* $\mathcal{B} \in PROG^-$ *и любого состояния БД* $\mathcal{E}$ *следующие утверждения эквивалентны:*

(i) $\mathcal{B}$ *является* $\delta-\exists\exists$*-стабильной в* $\mathcal{E}$;

(ii) существует строго δ*-ограниченная траектория, начинающаяся в* $\mathcal{E}$, *в которой некоторая продукция применяется дважды.*

Для доказательства леммы рассмотрим стабильную строго $(\mathcal{D}^+, \mathcal{D}^-)$-ограниченную траекторию с двумя применениями некоторой продукции π :

$$\omega : \ \mathcal{E}_0 ... \ \mathcal{E}_i \ \overset{\pi}{\vdash_{\mathcal{I}}} \ \mathcal{E}_i^* \ \overset{\mathcal{D}_i^+, \mathcal{D}_i^-}{\longrightarrow} \ \mathcal{E}_{i+1} ... \ \mathcal{E}_j \ \overset{\pi}{\vdash_{\mathcal{I}}} \ \mathcal{E}_j^* \ \overset{\mathcal{D}_j^+, \mathcal{D}_j^-}{\longrightarrow} \ \mathcal{E}_{j+1} \ ... \ .$$

Обозначим через $add^-(\pi)$ множество тех атомов, добавляемых π, которые имеются в $\mathcal{D}^-$, и положим $\hat{\mathcal{D}}_j^+ = \emptyset$, $\hat{\mathcal{D}}_j^- = add^-(\pi) \setminus \mathcal{E}_j$. Тогда, если мы подставим в ω возмущение $\mathcal{E}_j^* \overset{\hat{\mathcal{D}}_j^+, \hat{\mathcal{D}}_j^-}{\longrightarrow} \mathcal{E}'_{j+1}$ вместо $\mathcal{E}_j^* \overset{\mathcal{D}_j^+, \mathcal{D}_j^-}{\longrightarrow} \mathcal{E}_{j+1}$, то окажется, что $\mathcal{E}'_{j+1} = \mathcal{E}_j$, т.е. мы получили требуемую δ-стабильную траекторию.

Из этой леммы и леммы 1 следует, что для проверки стабильности достаточно рассматривать только строго δ-ограниченные траектории длины, не превосходящей $2 \mid \mathcal{B} \mid$.

Верхние оценки в утверждениях (3),(4) следуют из соответствующих верхних оценок теоремы 4 ниже.

(3) *Нижняя оценка.* Мы покажем, что язык, распознаваемы произвольной машиной Тьюринга $\mathcal{M}$ в памяти, ограниченной полиномом p, сводится за полиномиальное время к $STABLE(PROG)$. Пусть $\{q_j \mid 1 \leq j \leq k\} \cup \{q_s, q_f\}$ - множество состояний $\mathcal{M}$ (q_1 - начальное состояние, q_s - заключительное допускающее состояние, а q_f - заключительное отвергающее состояние). Пусть a_j $(0 \leq j \leq l)$ - символы алфавита ленты $\mathcal{M}$ (a_0 - пустой символ). Пусть $x = a_{i_1}, ..., a_{i_n}$ - входное слово и $N = p(n)$ - граница памяти. Экзистенциальная сигнатура $\mathcal{B}$ состоит из следующих 0-арных предикатов:

- q_j $(1 \leq j \leq k)$;
- a_{tj} $(1 \leq t \leq N, 0 \leq j \leq l)$: ячейка t содержит символ a_j;

- h_t $(1 \le t \le N)$: головка $\mathcal{M}$ находится в ячейке t.

Начальное состояние БД $\mathcal{E}_x = \{q_1, h_1, a_{1i_1}, ..., a_{ni_n}, a_{(n+1)0}, ..., a_{N0}\}$ кодирует начальную конфигурацию $\mathcal{M}$.

Для каждой команды $q_j a_m \to q_u a_v S$, где $S \in \{-1, 0, 1\}$, интенсиональная программа $\mathcal{I}$ содержит N продуцкций:

$$g :- q_j, h_t, a_{tm}, change(q_j, q_u), change(h_t, h_{t+S}), change(a_{tm}, a_{tv}) \quad (1 \le t \le N).$$

Кроме этого, $\mathcal{I}$ включает одну "заключительную"продукцию $g :- q_s$. Положим $\delta =< \emptyset, \emptyset >$ и $\Phi = \bigvee_{j=1}^{k} q_j \vee q_s$. Нетрудно проверить, что $\mathcal{M}$ допускает x в памяти $N \Leftrightarrow (\mathcal{B}, \delta, \mathcal{E}_x) \in STABLE(PROG)$.

(4) *Нижняя оценка* для этого случая получается с помощью конструкции теоремы 1 (4). Единственное отличие состоит в добавлении указанной выше "заключительной"продукции к $\mathcal{I}$ и использовании формулы Φ из предыдущего утверждения в качестве IC.

(5) $STABLE(PROD)$ неразрешима по той же причине, что и проблема перспективности для $PROD$. □

Теорема 4.

(1) Проблема $STABLE(GDS)$ является $PSPACE$-полной.

(2) Проблема $STABLE(FDS)$ является $SPACE(2^{poly})$-полной.

Доказателъство. Нижние оценки следуют непосредственно из нижних оценок утверждений (3),(4) теоремы 3.

Из леммы 1 следует,что существование δ-стабильной траектории $\mathcal{B}$, начинающейся в данном состоянии БД $\mathcal{E}$ эквивалентно существованию строго $\tilde{\delta}$-ограниченной стабильной траектории. Далее, поскольку число различных состояний БД, которые могут встретиться на строго $\tilde{\delta}$-ограниченных траекториях, начинающихся в $\mathcal{E}$ конечно, то на любой такой стабильной траектории хотя бы одно допустимое состояние $\mathcal{E}_i$ должно повториться. Следовательно, достаточно проверять лишь начальные отрезки строго $\tilde{\delta}$-ограниченных стабильных траекторий, завершающиеся такими допустимыми $\mathcal{E}_j$, что для некоторого $i < j$ $\mathcal{E}_i = \mathcal{E}_j$. По лемме 3 каждый шаг-модификация на такой траектории можно угадать и проверить в полиномиальной памяти для базисного случая и в экспоненциальной памяти для плоского случая. По лемме 1 мы получим те же границы памяти для моделирования шагов-возмущений. Тогда существует недетерминированный алгоритм, угадывающий и проверяющий начальные отрезки строго $\tilde{\delta}$-ограниченных стабильных траекторий в памяти, ограниченной полиномо от размера входа в базисном случае и экспонентой от размера входа - в плоском случае. □

7 Заключение

В работе предложена новая модель интерактивного поведения дедуктивных баз данных и определены некоторые характеристики его устойчивости по отношению к возмущениям внешней среды. Даже весьма упрощенные примеры, приведенные нами, показывают, что различного вида динамические системы, взаимодействующие со внешней средой, могут моделироваться ДБД, а свойства устойчивости их поведения можно формулировать и исследовать в предложенных терминах. Мы рассмотрели два самых слабых свойства устойчивого поведения - перспективность и $\exists\exists$-стабильностьhow и показали, что они разрешимы в ряде классов ДБД, интересных для приложений. Следует отметить, что соответствующие алгоритмы в общем случае либо работают за полиномиальное время при весьма строгих, но достаточно разумных ограничениях, либо являются простыми алгоритмами вида "породи-проверь", требующими экспоненциального времени или памяти. Тем не менее, в конкретных областях, возможно, такие недетерминированные алгоритмы могут быть реализованы более эффективно. Поэтому мы предполагаем в дальнейшем разработать, используя предложенный подход, интерактивное программное окружение для моделирования и анализ поведения сложных динамических систем.

Вторая часть этой работы будет посвящена некоторым другим свойствам устойчивого поведения, включая другие виды стабильности и гомеостатичность.

Благодарности

Авторы очень признательны за полезное обсуждение этой работы Марсу Валиеву и участникам семинаров в университете Париж-12, Московском государственном университете и Тверском государственном университете.

Список литературы

[1] Abiteboul, S., Vianu, V., *Datalog extensions for database queries and updates,* I.N.R.I.A. Technical Report No. 900, 1988.

[2] Aho, A.V., Ullman, J.D., *Universality of data retrieval languages, Proc. 6th ACM Symp. on Principles of Prog. Languages,* San Antonio, Texas, 110-117, 1979.

[3] Apt, K.R., Blair, H. and Walker A., *Towards a theory of declarative knowledge.* in: J. Minker (ed.) *Foundations of deductive databases and logic programming.* Morgan Kaufman Pub., Los Altos, 89-148, 1988.

[4] Bonner, A.J., *Hypothetical Datalog: complexity and expressibility.* Theoretical Computer Science, 76, 3-51, 1990.

[5] Bonner,A.G., Kifer, M., *Transaction logic programing,* In *Proc. of the Tenth Intern. Conf. on Logic Programming*. The MIT Press, 257-279, 1993.

[6] Chandra, A.K., Harel, D., *Computable queries for relational databases.* Journal of Computer and System Sciences, vol. 21, n.2, 156-178, 1980.

[7] Bylander, T., *The Computational Complexity of Propositional STRIPS Planning.* Artificial Intelligence, vol. 69, 165-204, 1994.

[8] Dekhtyar, M.I., Dikovsky, A.Ja., *On Stable Behavior of Dynamic Deductive Data Bases* In *Proc.of the 10th International Symp. on Logic Programming,* Ithaca, USA. The MIT Press, 677, 1994.

[9] Дехтярь М.И., Диковский А.Я., *Динамические дедуктивные базы данных.* Известия РАН, Техническая кибернетика N 5, 1994, 55-66.

[10] Dekhtyar, M.I., Dikovsky, A.Ja., *Dynamic Deductive Data Bases with Steady Behavior.* In *Proc. of the 12th International Conf. on Logic Programming,* (Ed. L. Sterling), The MIT Press, 183-197, 1995.

[11] Dikovsky, A.Ja., *Linear time solutions to the problems related to automatic synthesis of acyclic programw.* Programming, 3, 1985.

[12] Dikovsky, A.Ja., *On computational complexity of Prolog programs.* Theoretical Computer Science, 119, 63-102, 1993.

[13] Dung, P.M., Representing actions in logic programming and its application in database updates. In *Proc. of the Tenth Intern. Conf. on Logic Programming.* The MIT Press, 222-238, 1993.

[14] Eiter, T., Gottlob, G., *On the complexity of propositional knowledge base revision, updates, and counterfactuals.* Artificial Intelligence, vol. 57, 227-270, 1992.

[15] Fikes, R.E., Nilsson, N.J., *STRIPS: a new approach to the application of the theorem proving to the problem solving.* Artif. Intell. vol. 2 (3-4), 189-208, 1971.

[16] Gallaire, H., Minker, J., Nicolas, J.-M., *Logic and databases: a deductive approach.* ACM Computing Surveys, vol. 16, n.2, 153-185, 1984.

[17] Hull, R., *Relative information capacity of simple relational database schemata.* SIAM J. of Computing, vol. 15, n.3 856-886, 1986.

[18] Katsuno, H., Mendelzon, A. O., *Propositional knowledge base revision and minimal change.* Artificial Intelligence, vol. 52, 253-294, 1991.

[19] Lifschitz, V., *On the semantics of STRIPS.* In *Reasoning about Actions and Plans: Proc. of the 1986 Workshop*, Timberline, OR, 1-9, 1987.

[20] Manchanda, S., Warren, D.S., *A logic-based language for database updates.* In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming,* Morgan-Kaufmann, Los Altos, CA, 363-394, 1988.

[21] Naqvi, S., Krishnamurthy, R., *Database updates in logic programming.* In *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 251-262, 1988.

[22] Sadri, F., Kowalski, R., *A theorem proving approach to database integrity.* In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming,* Morgan-Kaufmann, Los Altos, CA, 313-362, 1988.

[23] Savitch, W.J., *Relationship between nondeterministic and deterministic tape complexities.* J. Comput. Syst. Sci., vol. 4, 177-192, 1970.

[24] Subrahmanian, V.S., Zaniolo, C., *Relating Stable Models and AI Planning Domains.* In *Proc. of the 12th International Conf. on Logic Programming,* (Ed. L. Sterling), The MIT Press, 233-247, 1995.