Санкт-Петербургский государственный университет

ШЛЁНСКИХ Алексей Анатольевич

Выпускная квалификационная работа

Анализ, обобщение и поиск зависимостей, использующих разностную семантику

Уровень образования: бакалавриат

Направление *02.03.03 «Математическое обеспечение и администрирование информационных систем»*

Основная образовательная программа *СВ.5162.2021 «Технологии программирования»*

Научный руководитель:
доцент кафедры информационно-аналитических систем, к. ф.-м. н., Михайлова Е. Г.

Рецензент:
Программист-разработчик, ООО «В Контакте», Слесарев А. Г.
Консультант:
ассистент кафедры ИАС, Чернышев Г. А.

Санкт-Петербург
2025

Saint Petersburg State University

Alexey Shlenskikh

Bachelor's Thesis

Analyzing, extending, and discovering dependencies that use difference semantics

Education level: bachelor

Speciality *02.03.03 "Software and Administration of Information Systems"*

Programme *CB.5162.2021 "Programming Technologies"*

Scientific supervisor:
C.Sc, Associate Professor E.G. Mikhailova

Reviewer:
Programmer-developer, LLC "V Kontakte" A.G. Slesarev
Consultant:
Assistant G.A. Chernishev

Saint Petersburg
2025

Оглавление

Введение

Профилирование данных является процессом сбора информации о данных. Примером такой информации могут являться некоторые простые показатели, такие как количество строк в таблице или среднее значение среди значений в некотором столбце. Кроме таких простых показателей, на данных можно искать и некоторые закономерности [1].

Одна из наиболее известных таких закономерностей — функциональная зависимость (Functional Dependency, FD) [10]. Она определена для табличных данных, и её смысл в том, что по значениям некоторых атрибутов записи можно точно определить значение некоторого другого атрибута.

У FD существуют различные расширения, такие как метрические функциональные зависимости (Metric Functional Dependency, MFD) [9], зависимости соседства (Neighborhood Dependency, NED) [2], разностные (или дифференциальные) зависимости (Differential Dependency, DD) [11] и сопоставляющие зависимости (Matching Dependency, MD) [6]. Расширениями они являются в том смысле, что любую FD можно выразить как закономерность одного из этих типов. Эти расширения лучше подходят для некоторых задач, таких как обнаружение дубликатов, поддержка целостности данных.

Для этих типов закономерностей (зависимостей) интуитивно имеет смысл задача поиска всех соблюдающихся зависимостей. Для решения этой задачи для MD существует алгоритм HyMD [5]. Также существует множество алгоритмов, решающих эту задачу для FD [3, 10].

В этой работе приводятся некоторые теоретические соображения, обосновывающие общий вид таких задач. С использованием этих идей предлагаются подобные задачи для остальных рассматриваемых типов зависимостей и доказывается, что имеет место вложение этих задач друг в друга. Это даёт возможность использовать уже существующий алгоритм для задач, алгоритмов решения которых нет в открытом доступе. На практике, однако, такое использование неудобно.

При этом автором в ходе работы по дополнительной оптимизации

упомянутого алгоритма [8] было замечено, что при некоторой модификации он позволяет получать решения подобной задачи для типа зависимости формально более общей формы, чем MD. И уже к задаче поиска для такого типа остальные задачи поиска сводить значительно проще. В работе этот новый тип зависимостей определяется и представляется модификация алгоритма HyMD, позволяющая решать указанную задачу.

1. Постановка задачи

Целью работы является обобщение знаний о зависимостях, использующих разностную семантику (difference semantics). Для её выполнения были поставлены следующие задачи:

1. Показать вложение задач поиска MFD, DD, NED, MD друг в друга;
2. Предложить тип зависимости, в задачу поиска которого вкладываются эти задачи поиска;
3. Разработать алгоритм для решения этой задачи поиска.

2. Обзор

При подготовке этого отчёта было просмотрено несколько работ, определяющих различные типы зависимостей на данных. Все они определялись по-разному: в разных стилях и с разными степенями формальности.

Словосочетание “разностная семантика” используется только в [11], но в данной работе зависимостями с разностной семантикой называются вообще все зависимости, которые задействуют понятие различия значений.

Зависимости могут соблюдаться (hold) или не соблюдаться. Также может использоваться слово “удерживаться”.

В этой работе используются определения из статьи [5]. Отношение — набор атрибутов (в иных работах называется схемой), экземпляр отношения — набор записей, имеющих эти атрибуты (в иных работах используется слово “отношение”), запись — функция, действующая из множества атрибутов в множество их значений.

В литературе (например, в книге [12]) определения наборов записей допускают бесконечные множества. Упоминаемые здесь алгоритмы работают только с конечными множествами записей.

Однако взгляд с бесконечными множествами записей имеет практический смысл для задачи проверки целостности данных. В ней мы, имея некоторую таблицу, можем сделать предположение об условиях, которым данные обязаны соответствовать. Эти условия можно проверять для поступающих затем данных. Например, мы можем сделать предположение, что фамилия, имя и отчество человека определяют серию и номер его паспорта. Если встречается запись, в которой ФИО совпадает с какой-то другой, а паспорта отличаются, то мы можем либо понять, что ошибочна запись, либо понять, что неверно условие.

Исследование зависимостей, соблюдающихся для тех конечных данных, которые имеются изначально, даёт некоторое начальное предположение — набор зависимостей, которые могут соблюдаться и, напротив, зависимости, которые точно не соблюдаются. Это предположение мо-

жет использоваться для проверки новых данных и уточняться.

Далее следуют описания рассматриваемых зависимостей. Все из них определены на одном или двух отношениях, для каждого из которых рассматриваются экземпляры. В основном в определениях рассматривается единственный экземпляр. Далее в отчёте будут использоваться следующие обозначения:

- R — первое отношение;
- r — экземпляр R;
- S — второе отношение (если зависимость определена для двух отношений);
- s — экземпляр S;
- $X \in R$ — X является атрибутом R;
- $X \in S$ — X является атрибутом S;
- $X \subseteq R$ — X является множеством атрибутов R;
- если A — атрибут, то $a[A]$ — значение A в записи a;
- если X — множество атрибутов, то $a[X]$ — проекция записи a на атрибуты X;

2.1. FD

В работе [10] функциональная зависимость определяется как утверждение вида $X \to A$, где $X \subseteq R$, $A \in R$. FD определяется на одном отношении.

Соблюдение определяется следующим образом:

Определение 1. FD $X \to A$ *соблюдается* на или *действительна* (valid) для $r \iff$

$$\forall (r_1, r_2) \in (r \times r) \quad (\forall B \in X \; r_1[B] = r_2[B] \implies r_1[A] = r_2[A]).$$

То есть значения атрибутов из X точно определяют значения A.

X называется *левой частью* (left hand side, LHS), A — *правой частью* (right hand side, RHS).

Определение 2. FD $X \to A$ называется *обобщением* (generalization) FD $Y \to A$, если $X \subseteq Y$, *специализацией* (specialization), если $X \supseteq Y$.

Определение 3. FD $X \to A$ *нетривиальна* (non-trivial), если $A \notin X$.

Определение 4. FD $X \to A$ *минимальна* (minimal), если $\nexists B \in R \;\; (X \backslash B) \to A$ — действительная зависимость на r.

Чтобы найти все соблюдающиеся на r FD достаточно найти все минимальные нетривиальные FD, соблюдение иных зависимостей можно проверить с помощью этого множества и логического вывода, не обращаясь к данным.

FD также может определяться с множеством атрибутов в правой части:

Определение 5. $X \subseteq R$, $Y \subseteq R$, FD $X \to Y$ соблюдается $\iff$

$$\nexists (p, q) \in (r \times r) \quad (\forall A \in X \;\; p[A] = q[A] \wedge \exists B \in Y \;\; p[B] \neq q[B]).$$

Здесь и далее обозначаем $(n \dots m) = \{x \in \mathbb{Z} \,|\, n \leqslant x \leqslant m\}$.

Теорема 1 (Декомпозиция FD)**.** $X \subseteq R$, $A = \{A_1, A_2, \dots, A_n\} \subseteq R \Rightarrow$

$$X \to A \textit{ соблюдается} \iff \forall i \in (1 \dots n) \quad X \to A_i \textit{ соблюдается.}$$

Эти определение и теорема представлены в [12].

Для поиска FD на данных можно использовать алгоритм HyFD [10]. Этот алгоритм выполняет поиск среди всех возможных FD на заданной таблице.

2.2. MFD

В данной работе используется определение метрической функциональной зависимости из работы [9]. MFD определяется на одном отношении.

В указанной работе через $\mathrm{dom}\,(X)$ обозначается домен атрибута $X \in R$. Если $X = A_1 A_2 \ldots A_k$, то принимается $\mathrm{dom}\,(X) = \mathrm{dom}\,(A_1) \times \mathrm{dom}\,(A_2) \times \ldots \times \mathrm{dom}\,(A_k)$.

Через $\pi_X(r)$ обозначается разбиение r по значениям атрибутов из X. $\pi_X(r) \coloneqq \{[a]_X(r) \mid a \in r\}$, где $[a]_X(r) \coloneqq \{b \in r \mid b[X] = a[X]\}$ — класс эквивалентности записи a по отношению равенства значений атрибутов из X.

Если $T \subseteq r$ — множество записей, $Y \subseteq R$, то $T[Y] \coloneqq \{t[Y] | t \in T\}$ — множество проекций записей из T на атрибуты Y.

В определении участвуют $X \subseteq R$, $Y \subseteq R$, $d : \mathrm{dom}\,(Y) \times \mathrm{dom}\,(Y) \to \mathbb{R}$ — метрика — выполняются симметричность, неравенство треугольника, тождество аргументов при нулевом результате, — $\delta \geqslant 0$ — параметр.

Пусть $\Delta_d(M) \coloneqq \max_{\{p,q\} \subseteq M} d(p, q)$ — диаметр множества M согласно метрике d.

Определение 6. MFD $X \xrightarrow{\delta} Y$ с метрикой d соблюдается на $r \iff$

$$\max_{T \in \pi_X(r)} \Delta_d(T[Y]) \leqslant \delta.$$

Интуитивно, это то же самое, что FD, но для значений Y не требуется точного равенства.

Факт 1. *FD $X \to Y$ выражается как MFD $X \xrightarrow{\delta} Y$, если d — дискретная метрика, $\delta = 0$.*

Этот факт приведён в [9], где дискретная метрика названа *exact metric*.

2.3. NED

Тип зависимости был представлен в работе [2], где был назван ND. Здесь используется NED, чтобы отличить его от Numerical Dependency [7]. NED определяется на одном отношении.

Авторы исходной статьи не определяют данную зависимость формально, но приводят её пример. В следующем разделе настоящей работы будет предложено её формальное определение.

Каждый атрибут $A \in R$ снабжается функцией близости (closeness function) $\theta_A(\cdot, \cdot)$, возвращающей значение между 0 и 1. Чем ближе её значение к 1, тем ближе значения.

Предикат соседства (neighborhood predicate, NP) сопоставляет атрибутам *пороговые значения* (threshold). Примером предиката соседства является

$$FSize^{0.5} Income^{0.8},$$

где атрибуту $FSize \in R$ сопоставляется пороговое значение 0.5, а атрибуту $Income \in R$ — 0.8. Две записи $t_1 \in r$ и $t_2 \in r$ являются соседями (neighbors) согласно ему, если $\theta_{FSize}(t_1[FSize], t_2[FSize]) \geqslant 0.5 \wedge \theta_{Income}(t_1[Income], t_2[Income]) \geqslant 0.8$.

Примером NED является

$$FSize^{0.5} Income^{0.8} \rightarrow Monovol^{1.0}.$$

Она описывает гипотезу, что соседи согласно $FSize^{0.5} Income^{0.8}$ также являются соседями согласно $Monovol^{1.0}$.

2.4. DD

Этот тип зависимости определяется в работе [11]. DD определяется на одном отношении.

Каждому атрибуту $B \in R$ сопоставляется метрика — выполняются неотрицательность, симметричность, тождество при нулевом результате — $d_B : \mathrm{dom}\,(B) \times \mathrm{dom}\,(B) \rightarrow \mathbb{D}$ и множество значений расстояний метрики $\mathbb{D}$. Условие неравенства треугольника для d_B не требуется.

Определение 7. $\varphi[B]$ $(B \in R)$ — *дифференциальная функция* (differential function) определяет ограничение (constraint) на значения расстояний $\mathbb{D}$, замеренными с помощью d_B по значениям B. Ограничения задаются операторами $=, <, >, \leqslant, \geqslant$.

Определение 8. Пусть $t_1 \in r$, $t_2 \in r$, $B \in R$. Дифференциальная функция $\varphi[B]$ возвращает `true`, если значение разницы в атрибуте B в них *согласовано* (agree) относительно ограничения в $\varphi[B]$. Это обозначается как $(t_1, t_2) \asymp \varphi[B]$.

Например, если имеется дифференциальная функция $\varphi[\text{name}] = [\text{name}(\leqslant 6)]$ и $d_{\text{name}}(t_1[\text{name}], t_2[\text{name}]) = 5$, что соответствует $\leqslant 6$, тогда говорим, что записи t_1, t_2 согласованы с дифференциальной функцией $\varphi[\text{name}]$, что обозначается как $(t_1, t_2) \asymp [\text{name}(\leqslant 6)]$.

При $B \in R$, $\text{dom}(B)$ считается конечным, поэтому для каждого атрибута набор расстояний $\mathbb{D}$ тоже конечен. То есть атрибуту B соответствует конечное количество дифференциальных функций относительно $\mathbb{D}$.

В этой работе это было интерпретировано следующим образом:

$$\mathbb{D} \coloneqq \{d_B(a, b) | (a, b) \in (\text{dom}(B) \times \text{dom}(B))\}.$$

Такое определение $\mathbb{D}$ используется в разделе 3.2.4.

Определение 9. Если $Z \subseteq R$, то дифференциальная функция $\varphi[Z]$ является конъюнкцией ограничений на атрибутах Z:

$$\varphi[Z] = \bigwedge_{B_i \in Z} \varphi[B_i].$$

Определение 10. Дифференциальное ограничение назовём *неограниченным* (unlimited), если любая пара записей согласована относительно него.

Определение 11. *Мощность* (cardinality) дифференциальной функции есть количество атрибутов, указанных в ней.

Определение 12. Пусть $\varphi_1[X]$ — дифференциальная функция, $X \subseteq R$. Пусть $Z \subseteq X$, тогда $\varphi_1[Z]$ — *проекция* $\varphi_1[X]$ на атрибуты Z.

Определение 13. Дифференциальная зависимость имеет форму

$$\varphi_L[X] \rightarrow \varphi_R[Y],$$

где $X \subseteq R, Y \subseteq R$, φ_L и φ_R — дифференциальные функции на атрибутах X и Y соответственно.

Определение 14. Экземпляр r *удовлетворяет* DD $\varphi_L[X] \to \varphi_R[Y]$ $\iff$

$$\forall\{t_1, t_2\} \subseteq r \quad ((t_1, t_2) \asymp \varphi_L[X] \implies (t_1, t_2) \asymp \varphi_R[Y]).$$

Это обозначается как $r \models \varphi_L[X] \to \varphi_R[Y]$.

Определение 15 (Соблюдение набора DD)**.** Пусть Σ — множество DD. $\Sigma \models r$ (соблюдается на r) $\iff$

$$\forall \varphi_L[X] \to \varphi_R[Y] \in \Sigma \quad r \models \varphi_L[X] \to \varphi_R[Y].$$

Дифференциальные функции также можно пересекать естественным образом. Например, $[\text{name}(\leqslant 5) \wedge \text{address}(\leqslant 12)] \wedge [\text{address}(\leqslant 10)] = [\text{name}(\leqslant 5) \wedge \text{address}(\leqslant 10)]$ или, аналогично, $[\text{name}(\leqslant 8) \wedge \text{name}(\geqslant 4)] = [\text{name}(\geqslant 4, \leqslant 8)]$.

Определение 16. Пусть $\varphi_1[Z]$, $\varphi_2[Z]$ — дифференциальные функции ($Z \subseteq R$). $\varphi_1[Z] \geq \varphi_2[Z]$ $\iff$

$$\forall (t_1, t_2) \quad ((t_1, t_2) \asymp \varphi_2[Z] \implies (t_1, t_2) \asymp \varphi_1[Z]).$$

Говорим, что $\varphi_1[Z]$ *включает* $\varphi_2[Z]$.

Определение 17. Пусть Σ — набор DD. Σ *согласованный* (consistent), если $\exists I (I$ — экземпляр отношения $\wedge\ I \neq \varnothing \wedge I \models \Sigma)$.

Определение 18. Пусть Σ_1, Σ_2 — наборы DD. $\Sigma_1 \models \Sigma_2$ $\iff$

$$\forall I \quad (I \models \Sigma_1 \implies I \models \Sigma_2),$$

где I — (произвольный) экземпляр отношения. Говорим, что Σ_2 следует из Σ_1 (Σ_1 implies Σ_2).

Определение 19. $\Sigma_1 \equiv \Sigma_2$ $\iff$

$$\Sigma_1 \models \Sigma_2 \wedge \Sigma_2 \models \Sigma_1.$$

Говорим, что Σ_1 и Σ_2 эквивалентны (equivalent).

Определение 20. Пусть Σ — набор DD. Набор DD Σ_1 — *покрытие* (cover) $\Sigma \iff$

$$\Sigma_1 \equiv \Sigma.$$

Определение 21. Набор DD Σ_c — *минимальное покрытие* (minimal cover) $\Sigma \iff$

1. *покрытие*, $\Sigma_c \equiv \Sigma$;
2. *редуцированный слева* (left-reduced),
   $\forall \varphi_L[X] \to \varphi_R[Y] \in \Sigma_c \quad \nexists(\varphi_1, W)(W \subseteq X \quad \wedge \quad \varphi_1[W] \geq \varphi_L[W] \quad \wedge$
   $\Sigma_c \models \varphi_1[W] \to \varphi_R[Y])$;
3. *включённый справа* (right-subsumed),
   $\forall \varphi_L[X] \to \varphi_R[Y] \in \Sigma_c \quad \nexists(\varphi_1, W)(Y \subseteq W \quad \wedge \quad \varphi_1[Y] \preceq \varphi_R[Y] \quad \wedge$
   $\Sigma_c \models \varphi_L[X] \to \varphi_1[W]))$;
4. *неизбыточный* (nonredundant), $\nexists \Sigma'(\Sigma' \equiv \Sigma \wedge \Sigma' \subsetneq \Sigma_c)$.

Определение 22. DD *в стандартной форме* (standard form) — DD вида $\varphi_L[X] \to \varphi_R[B]$, где $B \in R$ (не $B \subseteq R$).

Для каждого атрибута $B_i \in R$ изучается конечное множество дифференциальных функций — пространство поиска (search space). В пример приводится пространство поиска дифференциальных функций с ограничениями в виде интервалов — $\Phi(B_i) = \{B_i(\geqslant u, \leqslant v) | 0 \leqslant u \wedge u \leqslant v \wedge v \leqslant D\}$, где D — максимальное расстояние.

Естественным образом пространство поиска для m атрибутов в множестве $X = \{B_1, \ldots, B_m\} \subseteq R$ определяется как $\Phi(X) = \Phi(B_1) \times \ldots \times \Phi(B_m)$.

Определение 23. Дифференциальная функция называется *неограниченной* (unlimited), если любой экземпляр отношения R удовлетворяет ей.

Определение 24. Дифференциальная функция называется *невыполнимой* (infeasible), если не существует непустого экземпляра отношения R, который удовлетворял бы ей.

Проблема поиска DD определяется как проблема поиска минимального покрытия всех соблюдающихся на r DD из $\Phi(R)$.

В [4] представлен алгоритм поиска DD FastDD, который находится в открытом доступе. Этот алгоритм находит множество соблюдающихся DD, принимая на вход конечное множество дифференциальных функций для каждого атрибута.

2.5. MD

Приведено определение из работы [5]. MD определяется на двух отношениях.

Два значения классифицируются как похожие с помощью *меры сходства* (similarity measure) и *границы решения* (decision boundary). Вместе они формируют *классификатор сходства* (similarity classifier). Классификатор с границей решения 0.0 классифицирует любые два значения как похожие.

Определение 25. *Мера сходства* (similarity measure) — функция, определяющая сходство двух значений, возвращающая действительное число из $[0.0, 1.0]$.

MD определяются на наборе мер сходства $\approx$ и сопоставлений столбцов (column match) $C = \{C_1, C_2, \ldots, C_m\}$, где $C_i = (A_i, B_i, \approx_i) \in R \times S \times \approx$.

Определение 26. Пусть $\approx = \{\approx_1, \approx_2, \ldots, \approx_m\}$ — набор мер сходства, $A_i \in R$, $B_i \in S$ — i-тые атрибуты R и S соответственно, $C = \{C_1, C_2, \ldots, C_m\} = \{(A_1, B_1, \approx_1), (A_2, B_2, \approx_2), \ldots, (A_m, B_m, \approx_m)\} \subseteq R \times S \times \approx$ — набор сопоставлений столбцов. MD определена как

$$\left(\bigwedge_{i=1}^{m} R[A_i] \approx_{i,\lambda_i} S[B_i] \right) \rightarrow R[A_j] \approx_{j,\rho_j} S[B_j].$$

Левая часть (LHS) является конъюнкцией m классификаторов сходства. Каждый из них представлен сопоставлением столбцов C_i и границей решения $\lambda_i \in [0.0, 1.0]$. Набор границ решения левой части обозначается $\lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$. Правая часть (RHS) является одним классификатором сходства, $j \in (1 \ldots m)$. Она состоит из сопоставления столбцов C_j и границы решения $\rho_j \in [0.0, 1.0]$.

Определение 27. Для MD с обозначениями выше пара записей (r_k, s_l) *соответствует* (matches) LHS X MD $((r_k, s_l) \models X) \iff$

$$\bigwedge_{i=1}^{m} (r_k[A_i] \approx_i s_l[B_i]) \geqslant \lambda_i.$$

Определение 28. Аналогично для её RHS пара записей (r_k, s_l) *соответствует* (matches) RHS Y MD $((r_k, s_l) \models Y) \iff$

$$(r_k[A_j] \approx_j s_l[B_j]) \geqslant \rho_j.$$

Определение 29. MD *соблюдается* на $r, s \iff$

$$\forall (r_k, s_l) \in (r \times s) \quad ((r_k, s_l) \models X \implies (r_k, s_l) \models Y).$$

Если пара записей соответствует LHS MD, но не соответствует RHS MD, то говорим, что пара *нарушает* (violates) MD.

Определение 30. MD $\phi(\lambda, \rho_j)$ *тривиальная* (trivial) $\iff \lambda_j \geqslant \rho_j$.

Определение 31. Для сопоставления столбцов $C_i = (A_i, B_i, \approx_i)$ назовём границу решения λ_i *естественной*, если $\exists (p, q) \in (r \times s) \quad (p[A_i] \approx_i q[B_i]) = \lambda_i$. Определение из работы [8]. В работе [5] использовалось фактически то же самое определение, но оно было описано словами.

Тривиальные зависимости соблюдаются вне зависимости от данных, поэтому их можно не искать.

По действительным (valid), то есть соблюдающимся, MD можно определить другие действительные MD. Определяется частичный порядок таким образом, чтобы действительные MD определяли следующие

за ними элементы как действительные. Сначала упорядочиваются LHS по их наборам границ решения:

Определение 32. Пусть λ, λ' — наборы границ решения. $\lambda \preceq \lambda' \iff$

$$\forall i \in (1 \dots m) \quad \lambda_i \leqslant \lambda'_i.$$

При этом говорим, что λ *включает* (subsumes) λ'. λ называется *обобщением* (generalization) λ', а λ' называется *специализацией* (specialization) λ.

Далее этот порядок используется для введения порядка на всех MD:

Определение 33. Пусть $\phi(\lambda, \rho_j)$, $\phi'(\lambda', \rho'_j)$ — MD. $\phi(\lambda, \rho_j) \preceq \phi'(\lambda', \rho'_j) \iff$

$$\lambda \preceq \lambda' \wedge \rho_j \geqslant \rho'_j.$$

Точно также говорим, что ϕ *включает* (subsumes) ϕ'. ϕ называется *обобщением* (generalization) ϕ', а ϕ' называется *специализацией* (specialization) ϕ.

Если MD соблюдается, то и все следующие элементы согласно $\preceq$ также соблюдаются.

Поскольку не все MD представляют практический интерес, авторы предлагают также критерии интересности, согласно которым MD будут исключаться. Критерии интересности нужны для уменьшения пространства поиска.

Предлагаемые критерии интересности:

1. Мощность (cardinality) — количество классификаторов столбцов с ненулевой границей решения в LHS, исключать зависимости, если она слишком велика;

2. Поддержка (support) — количество пар, соответствующих LHS, исключать MD, у которых оно слишком низкое;

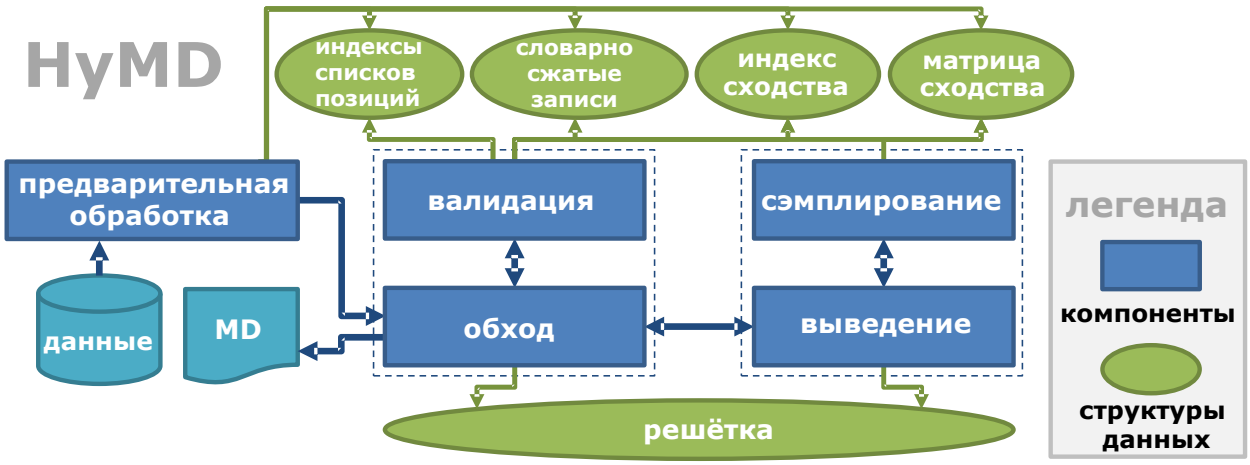


Рис. 1: Источник: [5]

3. Отсутствие пересечения (disjointness) — граница решения у классификатора столбцов в левой части с тем же индексом, что и у классификатора в правой части, нулевая, исключать зависимости, не обладающие этим свойством;

4. Значение границы решения (decision boundaries (value)) — достаточно высокое значение границы решения в классификаторе столбцов, предлагается исключать зависимости, если она слишком низкая;

5. Количество границ решения (decision boundaries (number)) — точность границы решения в LHS, предлагается исключать зависимости с некоторыми границами решения в LHS.

2.6. HyMD

В работе [5] описывается алгоритм поиска MD, называющийся HyMD. Здесь приведено его краткое описание. На рисунке 1 изображено устройство этого алгоритма.

Границы решения, по которым производится поиск (естественные), выводятся из данных на этапе предварительной обработки (preprocessing). На этом же этапе формируются четыре структуры данных:

- индексы списков позиций (position list indexes, PLIs) — для каждого столбца таблиц сопоставляет значение списку записей (кластеру), которые содержат это значение в этом столбце;

- словарно сжатые записи (dictionary compressed records) — список записей таблиц, где в каждой из них значения заменены на их идентификаторы, которые являются номерами кластеров в соответствующих PLI;

- матрица сходства (similarity matrix) — для сопоставления столбцов сопоставляет каждой паре значений из указанных в нём столбцов их сходство;

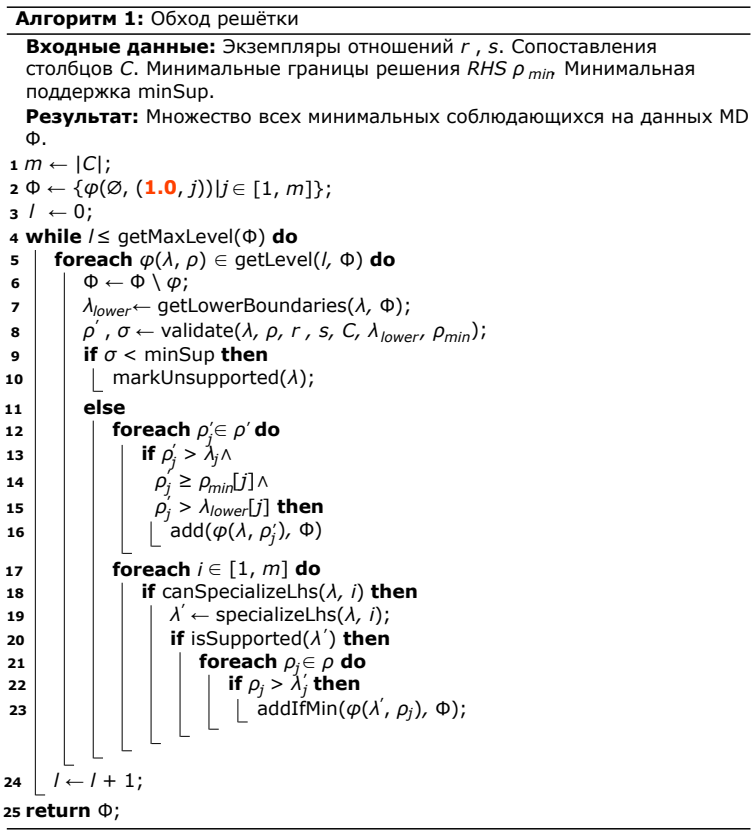
- индекс сходства (similarity index) — для сопоставления столбцов указывает для каждого значения первого столбца на сходства, а каждое сходство указывает на список идентификаторов записей s, в которых значение второго столбца имеет сходство с первым значением не меньше его.

Смысл матриц сходства: сходства между значениями нужно вычислять очень много раз для различных возможных MD, вычисление сходств — дорогая операция, поэтому предлагается хранить все нужные сходства в них.

Вычисление сходств является дорогой операцией, но каждое вычисление независимо от всех остальных, поэтому эти вычисления выполняются параллельно.

Зависимости логически выстраиваются в решётку согласно частичному порядку $\preceq$, здесь она именуется “решётка порядка”.

Множество MD, которые предполагаются соблюдающимися в некоторый момент исполнения, (множество кандидатов) хранится в структуре, называемой решёткой (lattice), здесь “решётка кандидатов”. Изначально предполагается, что соблюдаются все зависимости. Структура предоставляет несколько операций, которые позволяют исполнять алгоритм. Она реализуется как префиксное дерево, где LHS зависимостей представлены как пути до узла, а в каждом узле хранятся границы



Алгоритм 1: Обход решётки

Входные данные: Экземпляры отношений r , s. Сопоставления столбцов C. Минимальные границы решения RHS ρ_{min}, Минимальная поддержка minSup.

Результат: Множество всех минимальных соблюдающихся на данных MD Φ.

```
1  m ← |C|;
2  Φ ← {φ(∅, (1.0, j))|j ∈ [1, m]};
3  l ← 0;
4  while l ≤ getMaxLevel(Φ) do
5      foreach φ(λ, ρ) ∈ getLevel(l, Φ) do
6          Φ ← Φ \ φ;
7          λ_lower ← getLowerBoundaries(λ, Φ);
8          ρ', σ ← validate(λ, ρ, r , s, C, λ_lower, ρ_min);
9          if σ < minSup then
10             markUnsupported(λ);
11         else
12             foreach ρ'_j ∈ ρ' do
13                 if ρ'_j > λ_j ∧
14                    ρ'_j ≥ ρ_min[j] ∧
15                    ρ'_j > λ_lower[j] then
16                     add(φ(λ, ρ'_j), Φ)
17             foreach i ∈ [1, m] do
18                 if canSpecializeLhs(λ, i) then
19                     λ' ← specializeLhs(λ, i);
20                     if isSupported(λ') then
21                         foreach ρ_j ∈ ρ do
22                             if ρ_j > λ'_j then
23                                 addIfMin(φ(λ', ρ_j), Φ);
24     l ← l + 1;
25 return Φ;
```

Рис. 2: Источник: [5]. Красным выделена изначальная граница решения, равная 1.0.

решения правых частей всех кандидатов с этой LHS. Для структуры сохраняется инвариант минимальности — все MD в ней должны оставаться минимальными относительно $\leq$ друг среди друга или, иными словами, все MD в ней должны быть попарно несравнимы относительно $\leq$. MD также не добавляются, если они являются тривиальными.

Алгоритм комбинирует два подхода поиска зависимостей: обход решётки (приведён на рис. 2) и выведение из пар записей.

В первом решётка порядка делится на множества MD (уровни). Затем для каждой зависимости уровня параллельно происходит валидация (validation) — определение границы решения в правой части. Зависимости, для которых валидация уже была проведена, исключаются. Механизм валидации зависит от мощности MD. Для мощности больше чем один требуется пересекать множества элементов списков из индексов сходства. В процессе валидации для ненулевой мощности просматриваются элементы разбиения r на множества записей, в которых

совпадают значения столбцов из R, которые находятся в классификаторах столбцов из левой части с ненулевой границей решения, и явно обнаруживаются пары записей, нарушающие MD. Они сохраняются и могут быть использованы вторым подходом. Для получения множеств записей, соответствующих LHS, используются индексы сходства, для проверки на соответствие правой части используются матрицы сходства.

После валидации последовательно для каждой MD границы решения устанавливаются в полученные значения — если не нарушается обозначенный выше инвариант решётки кандидатов, иначе они "удаляются"[1], — и в решётку кандидатов добавляются специализации всех MD, у которых граница решения была понижена, если это не нарушит минимальность. Специализации имеют один вид: граница решения в левой части для каждого из возможных сопоставлений столбцов повышается до более высокой границы решения из тех, что ищутся, остальные границы решения остаются такими же, как раньше. Повышение происходит до наименьшей из бóльших границ, если такая существует, иначе специализация не добавляется.

В [8] алгоритм меняет подход на выведение из пар записей, как только обход решётки становится неэффективным согласно некоторому показателю, в [5] это происходит сразу после обработки одного уровня.

Выведение из пар записей также использует множество кандидатов. По некоторому принципу выбираются пары записей (это называется "сэмплирование", sampling), затем нужные значения в них сравниваются, то есть из матриц сходств получаются нужные сходства. Затем с помощью решётки кандидатов определяются зависимости, нарушаемые этой парой.

После получения набора нарушенных MD происходит процесс, сходный с предыдущим подходом. Последовательно для каждой нарушенной зависимости устанавливаются пониженные границы решения, если не нарушается инвариант решётки кандидатов, иначе зависимость удаляется. Затем в решётку кандидатов добавляются, если при этом со-

[1] в реализации это определяется во время валидации

храняется инвариант, специализации такого же вида, как и для обхода решётки, но повышение границы происходит до наименьшего из таких значений (которые ищутся), чтобы пара записей больше не сопоставлялась получающейся LHS.

В HyMD выведение из пар записей сначала выбирает пары записей, которые были обнаружены в процессе валидации.

Пары выбираются до тех пор, пока выведение из пар записей не становится неэффективным согласно некоторому критерию.

От подхода к сэмплированию время работы алгоритма может зависеть очень сильно, в некоторых случаях времена работы могут различаться на два порядка [8].

HyMD начинает поиск зависимостей с подхода выведения из пар записей, переключается между двумя подходами. Алгоритм завершает работу после выполнения обхода решётки, как только в решётке кандидатов не остаётся зависимостей, для которых нужно проводить валидацию.

3. Анализ зависимостей

3.1. Общие идеи

В этой части указываются утверждения для зависимостей, которые помогут в формулировке общего вида задачи поиска.

FD и MD определяют тривиальные зависимости, которые искать не нужно. Обобщим их определения.

Определение 34. Зависимость **тривиальна**, если она соблюдается на любых данных.

3.1.1. Минимальное покрытие

Все указанные определения зависимостей явно или неявно используют идею сравнения записей в некоторых парах. Для FD это пары из $(r \times r)$, для MD это пары из $(r \times s)$. Будем обозначать множество таких пар как t.

Определение 35. Обозначим через $\text{holds}[\text{dep}, t]$ утверждение, что зависимость dep соблюдается на табличных данных, пары записей которых обозначены как t.

В работе [5] явно определяется частичный порядок для MD. В работе [10] порядок явно не определяется, но говорится, что зависимости для нахождения всех соблюдающихся FD достаточно набора минимальных FD, из которых затем можно вывести остальные. Дадим название порядку, который здесь подразумевается.

Определение 36. Назовём $\preceq$ **порядком следования**, если

$$\forall \text{dep} \forall \text{dep}'(\text{dep} \preceq \text{dep}' \implies \forall t(\text{holds}[\text{dep}, t] \implies \text{holds}[\text{dep}', t])).$$

Теперь с помощью этого порядка можно представлять большие множества зависимостей небольшим набором:

Определение 37. Назовём множество зависимостей φ **покрытием** множества Φ, если

$$\Phi = \{\psi | \exists \psi' (\psi' \in \varphi \wedge \psi' \preceq \psi)\}.$$

Назовём покрытие φ **минимальным**, если его элементы являются минимальными относительно порядка следования в Φ:

$$\varphi = \{\psi | \nexists \psi' (\psi' \in \Phi \wedge \psi' \preceq \psi \wedge \psi' \neq \psi)\}].$$

Пользуясь этими определениями, можно сказать, что задача, которую решают алгоритмы HyFD и HyMD, — это задача поиска минимального покрытия множества соблюдающихся зависимостей. То есть присутствует некоторое множество нетривиальных зависимостей, на котором ведётся поиск, и из него выбирается подмножество соблюдающихся на данных зависимостей.

Для случая функциональных зависимостей понятно, что это за множество:

Факт 2. *Множество, в котором выполняется поиск функциональных зависимостей — это множество всех возможных нетривиальных функциональных зависимостей на отношении того экземпляра, который является входным параметром.*

Это множество конечно, хотя и велико.

Для поиска MD с помощью HyMD предлагается задавать набор сопоставлений столбцов и искать зависимости, классификаторы столбцов которых содержат одно из них и какое-то значение, а именно естественную границу решения. Меры сходства по определению имеют бесконечное множество значений, поэтому для алгоритма имеет смысл выбрать некоторое конечное подмножество. Обоснований выбора именно естественных границ решения в статье [5] не приводится, эта деталь будет разобрана далее в работе.

Но идея распространяется и на другие зависимости — для задачи поиска логично задавать атрибуты, значения которых будут измеряться,

и функции, которые будут выполнять измерение. Зависимости в множестве, где ведётся поиск покрытия, которое будет определяться как часть решения задачи, должны использовать эти функции и некоторый конечный набор расстояний. Обоснование для каких-то конкретных вариантов определения этого множества будет приведено в разделе 3.3.

3.1.2. Общий вид зависимостей

Утверждение, что зависимость соблюдается, будем называть **условием соблюдения**.

Все зависимости задействуют в своих определениях некоторые условия для каждой пары. Для FD и MD явно обозначаются левая часть (LHS) и правая часть (RHS).

Посмотрим на определение условия соблюдения для FD:

$$\forall (r_1, r_2) \in (r \times r) \quad (\forall B \in X \; r_1[B] = r_2[B] \implies r_1[A] = r_2[A]).$$

Если обобщать, то в нём утверждается, что для каждой пары, для которой выполняются все условия, указанные в LHS, условие, указанное в RHS, также выполняется.

Далее будет показано, что для условия соблюдения любой из рассматриваемых пяти зависимостей можно привести эквивалентное логическое выражение, имеющее следующую форму:

Форма выражения 1.

$$\forall u \in t \quad (\forall e \in \text{LHS} \; \text{ElRst}[e, u] \implies \forall e' \in \text{RHS} \; \text{ElRst}[e', u]),$$

где $\text{ElRst}[e, u]$ *— логическое выражение с двумя свободными переменными, множество LHS является функцией. Множество, подставляемое на место e будем называть* ***элементом*** *зависимости. Способ составления множеств* LHS *и* RHS *зависит от рассматриваемого типа зависимости и будет указан.*

Если $\text{ElRst}[e, u]$ *истинно для некоторой пары записей u, то говорим, что записи в этой паре* ***сопоставлены*** *элементом e.*

Элемент зависимости можно рассматривать как обобщение классификатора сходства, определённого для MD.

Вспомним, что утверждение, что достаточно найти набор минимальных (соблюдающихся) зависимостей, чтобы узнать набор всех соблюдающихся зависимостей, для FD обосновывалось тем, что из минимальных зависимостей можно логически вывести все остальные соблюдающиеся. Применим здесь эту идею.

Теорема 2.

$$\forall u \in t\ (\forall e \in \text{LHS ElRst}[e, u] \implies \forall e' \in \text{RHS ElRst}[e', u])$$
$$\iff \forall e' \in \text{RHS} \quad \forall u \in t\ (\forall e \in \text{LHS ElRst}[e, u] \implies \text{ElRst}[e', u]).$$

Теорема доказывается средствами математической логики, доказательство опущено для краткости.

Она означает, что для задачи поиска достаточно рассматривать только зависимости, у которых в правой части один элемент, ведь те, у которых в правой части несколько элементов, можно вывести логически.

Если обозначить этот единственный элемент как RHS, то достаточно рассматривать ещё более простую форму выражения:

Форма выражения 2.

$$\forall u \in t\ (\forall e \in \text{LHS ElRst}[e, u] \implies \text{ElRst}[\text{RHS}, u]).$$

Приведение условий соблюдения зависимостей к этой форме выполняется для целей обоснования задания задачи поиска.

3.1.3. Вид элементарного ограничения

Уточним форму условия $\text{ElRst}[e, u]$. В MD пара записей сопоставляется правой частью, если результат меры сходства не меньше некоторого значения. В случае DD дифференциальная функция возвращает `true`, если записи согласованы относительно ограничений, которые задаются с помощью операторов порядка ($\leqslant$, $\geqslant$, $<$, $>$) и равенства. В

MFD диаметр множества — который определяется через сравнение пар значений — сравнивается с некоторым числом.

В этих зависимостях результат применения какой-то функции к значениям атрибутов записей сравнивается с каким-то значением относительно некоторого линейного порядка.

В таких терминах можно переформулировать и условия, использующиеся в определении соблюдения FD. Пусть `eq` — это функция, которая возвращает `true`, если значения совпадают, и `false` иначе. Пусть линейный порядок R_{bool} — это естественный порядок на булевых значениях, где `false` предшествует true: $R_{\text{bool}} = \{(\texttt{false}, \texttt{false}), (\texttt{false}, \texttt{true}), (\texttt{true}, \texttt{true})\}$. Условие $r_1[A] = r_2[A]$ тогда представляется как $\texttt{true}\ R_{\text{bool}}\ \texttt{eq}(r_1[A], r_2[A])$.

Теорема 3. *Выражение* $\text{ElRst}[e, u]$ *из форм выражения 1 и 2 может иметь вид*

$$l \overset{\mathbf{L}}{\leq} F(T(p), V(q)),$$

где $e = ((T, V, F, \overset{\mathbf{L}}{\leq}), l)$, $T : r \to X$, $V : s \to Y$, $F : X \times Y \to \mathbf{L}$, $\overset{\mathbf{L}}{\leq}$ — *нестрогий линейный порядок на* $\mathbf{L}$, $u = (p, q)$.

Это выражение можно рассматривать как обобщённый вид выражения из определения 28. Первый элемент e здесь является аналогом сопоставления столбцов, назовём его **сопоставлением записей**, второй — аналогом границы решения, будем называть его так же.

Эта теорема будет доказываться для каждого типа зависимостей. В случае рассматриваемых типов функции T, V возвращают значения атрибутов записей.

3.2. Переформулировка соблюдения зависимостей

В этом разделе доказывается теорема 3 для всех рассматриваемых типов зависимостей, то есть приводятся к одной из форм 2 или 1. Если задана одна из них, то другая тоже считается заданной (см. теорему 2). После того, как LHS и RHS задаются явным образом, доказательства

эквивалентности проводятся только с помощью замен определений и простых логических выводов, поэтому эти шаги опущены.

Также будет указано на способы задания пространства поиска. Сами задачи поиска будут сформулированы после указания общих принципов в разделе 3.3.

Через $[]_A$ обозначим функцию, возвращающую атрибут записи A, через $[]'_X$ — проекцию записи на атрибуты X. Через flip обозначим функцию, которая сопоставляет отношению $\leqslant$ отношение $\geqslant$, отношению $\geqslant$ — отношение $\leqslant$.

3.2.1. FD

В определении FD задействованы атрибуты $X \subseteq R$, $Y \in R$. Нужные понятия уже были определены в разделе 3.1.3. Для формы выражения 2 $\text{LHS} = \{(([]_A, []_A, \text{eq}, R_{\text{bool}}), \texttt{true}) | A \in X\}$, $\text{RHS} = (([]_Y, []_Y, \text{eq}, R_{\text{bool}}), \texttt{true})$.

Множество, на котором выполняется поиск FD, уже известно — это все возможные на отношении подаваемого на вход экземпляра зависимости.

3.2.2. MFD

Теорема 4 (Равносильное условие соблюдения MFD)**.** *В обозначениях из раздела 2.2 верно следующее:*

$$\max_{T \in \pi_X(r)} \Delta_d(T[Y]) \leqslant \delta$$
$$\iff \forall (p, q) \in (r \times r) \quad (\forall B \in X \ p[B] = q[B] \implies d(p[Y], q[Y]) \leqslant \delta).$$

Лемма 1.

$$\forall T \in \pi_X(r) \quad \forall (p, q) \in (T \times T) \quad d(p[Y], q[Y]) \leqslant \delta$$
$$\iff \forall (p, q) \in (r \times r) \quad (p[X] = q[X] \implies d(p[Y], q[Y]) \leqslant \delta).$$

Доказательство. $\implies$:

Возьмём произвольную пару записей $(p, q) \in (r \times r)$. $p[X] = q[X] \implies$

$\exists T \in \pi_X(r) \quad (p \in T \wedge q \in T)$. Очевидно, что $(p, q) \in (T \times T)$. Тогда по предположению $d(p[Y], q[Y]) \leqslant \delta$.

$\Longleftarrow$:

Рассмотрим произвольный класс эквивалентности $T \in \pi_X(r)$, произвольную пару записей $(p, q) \in (T \times T)$. По определению класса эквивалентности $p[X] = q[X]$. Очевидно $(p, q) \in (T \times T) \implies (p, q) \in (r \times r)$. Тогда по предположению $d(p[Y], q[Y]) \leqslant \delta$. Лемма доказана. □

Теперь докажем теорему.

Доказательство теоремы. Сначала проведём преобразования по определению.

$$\max_{T \in \pi_X(r)} \Delta_d(T[Y]) \leqslant \delta \iff \max_{T \in \pi_X(r)} \max_{\{p,q\} \subseteq T[Y]} d(p, q) \leqslant \delta$$
$$\iff \max_{T \in \pi_X(r)} \max_{\{p,q\} \subseteq T} d(p[Y], q[Y]) \leqslant \delta.$$

Далее, очевидно, что $\max_{\{p,q\} \subseteq T} d(p[Y], q[Y]) = \max_{(p,q) \in (T \times T)} d(p[Y], q[Y])$.

$$\max_{T \in \pi_X(r)} \max_{\{p,q\} \subseteq T} d(p[Y], q[Y]) \leqslant \delta \iff \max_{T \in \pi_X(r)} \max_{(p,q) \in (T \times T)} d(p[Y], q[Y]) \leqslant \delta$$
$$\iff \forall T \in \pi_X(r) \quad \forall (p, q) \in (T \times T) \quad d(p[Y], q[Y]) \leqslant \delta.$$

По лемме 1:

$$\forall T \in \pi_X(r) \quad \forall (p, q) \in (T \times T) \quad d(p[Y], q[Y]) \leqslant \delta$$
$$\iff \forall (p, q) \in (r \times r) \quad (p[X] = q[X] \implies d(p[Y], q[Y]) \leqslant \delta).$$

Тривиальное преобразование даёт искомый результат:

$$\forall (p, q) \in (r \times r) \quad (p[X] = q[X] \implies d(p[Y], q[Y]) \leqslant \delta)$$
$$\iff \forall (p, q) \in (r \times r) \quad (\forall B \in X \; p[B] = q[B] \implies d(p[Y], q[Y]) \leqslant \delta).$$

□

Теперь определим для формы выражения 2 LHS =

$\{(([]_A, []_A, \text{eq}, R_{\text{bool}}), \texttt{true}) | A \in X\}$, RHS $= (([]'_Y, []'_Y, d, \text{flip}(\leqslant)), \delta)$.

Множество зависимостей для задачи поиска может определяться с помощью задания соответствия между некоторыми наборами атрибутов и функцией-метрикой.

3.2.3. DD

Используются обозначения из раздела 2.4.

Ясно, что форма, написанная в [11] отличается только в обозначениях от следующего вида, ведь условия $\{t_1, t_2\} \subseteq r$ и $(t_1, t_2) \in (r \times r)$ равносильны.

Факт 3.

$$\forall \{t_1, t_2\} \subseteq r \quad ((t_1, t_2) \asymp \varphi_L[X] \implies (t_1, t_2) \asymp \varphi_R[Y])$$
$$\iff \forall (t_1, t_2) \in (r \times r) \quad ((t_1, t_2) \asymp \varphi_L[X] \implies (t_1, t_2) \asymp \varphi_R[Y]).$$

В DD для каждого атрибута может быть определено несколько ограничений с помощью операторов, например price($\geqslant 100, \leqslant 900$). Очевидно, что каждое такое ограничение можно переформулировать в конъюнкцию нескольких: price($\geqslant 100$) $\wedge$ price($\leqslant 900$).

Тогда дифференциальные функции, составляющие дифференциальную зависимость будут выглядеть как конъюнкция дифференциальных функций, в которых указан один атрибут, один оператор и одно значение. Представим это как множество троек (атрибут, знак, значение) $G = \{(A_1, \text{ord}_1, v_1), (A_2, \text{ord}_2, v_2), \ldots, (A_n, \text{ord}_n, v_n)\}$ для $\varphi_L[X]$, аналогично H для $\varphi_R[Y]$.

Ограничения в дифференциальных функциях специфицируются операторами $=$, $<$, $>$, $\leqslant$, $\geqslant$. Пока что будем считать, что дифференциальные функции содержат только последние два оператора. В следующем разделе покажем, что ограничения с использованием остальных операторов можно переформулировать, используя только их.

Приведя DD в такой вид, зададим для формы выражения 1 LHS и RHS. LHS $= \{(([]_A, []_A, d_A, \text{flip}(\text{ord})), v) | (A, \text{ord}, v) \in G\}$, RHS $= \{(([]_A, []_A, d_A, \text{flip}(\text{ord})), v) | (A, \text{ord}, v) \in H\}$.

Множество зависимостей для задачи поиска может определяться с помощью задания соответствия между атрибутом и функцией-метрикой для него, то есть отображения d.

3.2.4. $DD(\leqslant, \geqslant) \equiv DD(=, \leqslant, \geqslant, <, >)$

Далее говорим, что дифференциальные функции $\varphi_1[X]$, $\varphi_2[X]$ равносильны, $\varphi_1[X] \equiv \varphi_2[X] \iff$

$$\forall (p, q) \in (r \times r) \quad ((p, q) \asymp \varphi_1[X] \iff (p, q) \asymp \varphi_2[X]).$$

Факт 4. *Все неограниченные дифференциальные функции равносильны. То есть, если $\varphi_1[X]$, $\varphi_2[X]$ — неограниченные дифференциальные функции, то $\varphi_1[X] \equiv \varphi_2[X]$.*

Следствие 1. *$\varphi_1[X]$, $\varphi_2[X]$ — неограниченные дифференциальные функции $\Longrightarrow$*

$$\forall \varphi_R[Y] \quad \{\varphi_1[X] \to \varphi_R[Y]\} \equiv \{\varphi_2[X] \to \varphi_R[Y]\}.$$

Аналогичные утверждения верны для невыполнимых дифференциальных функций.

Обратим внимание, что если рассматриваемый экземпляр отношения пуст, то условие соблюдения выполняется вне зависимости от указываемых LHS и RHS, поэтому этот случай не рассматриваем.

Теорема 5.

$$0 \in \mathbb{D}.$$

Доказательство. По определению DD:

$$\forall B(d_B : \mathrm{dom}(B) \times \mathrm{dom}(B) \to \mathbb{D} \wedge d_B \text{ — метрика}).$$

$\mathrm{dom}(B) \neq \varnothing \Rightarrow (\mathrm{dom}(B) \times \mathrm{dom}(B)) \neq \varnothing \Rightarrow$ $\exists a \in \mathrm{dom}(B) \quad (a, a) \in (\mathrm{dom}(B) \times \mathrm{dom}(B))$. По определению метрики $\forall a \in \mathrm{dom}(B) \quad d_B(a, a) = 0 \Rightarrow 0 \in \mathbb{D}$. □

Теперь покажем, что нестрогих неравенств достаточно для приведения условия соблюдения DD в форму 1. Рассмотрим ограничения, использующие операторы $=$, $<$, $>$.

Ограничения вида $\mathrm{attr}(= a)$ очевидно выражаются как $\mathrm{attr}(\leqslant a) \wedge \mathrm{attr}(\geqslant a)$.

Для строгих неравенств воспользуемся конечностью $\mathbb{D}$ для каждого атрибута.

Если рассматриваются ограничения с операторами $>$, $<$, то для атрибута $B \in R$, $\mathrm{dom}(B) \neq \varnothing$, есть два случая: $|\mathrm{dom}(B)| = 1$ и $|\mathrm{dom}(B)| > 1$. Первый из них не имеет большого значения для практического применения, поэтому здесь он рассматривается кратко.

Если $|\mathrm{dom}(B)| = 1$, то по теореме 5 и поскольку $\mathbb{D}$ — множество значений расстояний метрики, $\mathbb{D} = \{0\}$, то есть для этого атрибута возможны только следующие ограничения: $B(= 0)$, $B(> 0)$, $B(< 0)$, $B(\leqslant 0)$, $B(\geqslant 0)$. Очевидно, что $[B(= 0)] \equiv [B(\leqslant 0)] \equiv [B(\geqslant 0)]$, все три ограничения являются неограниченными. Тогда в форме выражения 1 соответствующие им тройки можно не включать во множества LHS и RHS.

Если дифференциальная функция содержит ограничения $B(> 0)$, $B(< 0)$, то она является невыполнимой, $[B(> 0)] \equiv [B(< 0)]$. Если $\forall B \in R \quad |\mathrm{dom}(B)| = 1$, то дифференциальная зависимость содержит невыполнимую функцию слева, а поэтому точно выполняется. Множество LHS для такой дифференциальной зависимости можно просто оставить пустым. Если же последнее условие не выполняется, то в множество G или H можно включить вместо тройки, соответствующей этой функции, любую другую невыполнимую (см. ниже).

Теперь рассмотрим случай $|\mathrm{dom}(B)| > 1$. Для такого атрибута, поскольку атрибутам сопоставляются метрики, в определении которых содержится условие тождества аргументов при нулевом результате, $\exists a \ (a \neq 0 \wedge a \in \mathbb{D})$.

Поскольку $\forall B \in R \quad (\mathbb{D} \neq \varnothing \wedge \mathbb{D} \in \mathrm{Fin})$, существуют минимумы и максимумы любого $\mathbb{D}$, обозначим их здесь как $m \coloneqq \min(\mathbb{D})$, $n \coloneqq \max(\mathbb{D})$, при этом в случае $|\mathrm{dom}(B)| > 1$ верно $m \neq n$, то есть $m < n$.

Если ограничение имеет вид $\mathrm{attr}(< a)$, $a \in \mathbb{D}$, то есть два случая. $a = m$, тогда дифференциальная функция, включающая его, является невыполнимой и ей равносильна дифференциальная функция $[\mathrm{attr}(\leqslant m) \wedge \mathrm{attr}(\geqslant n)]$. $a \neq m$, тогда этому ограничению равносильно

ограничение $\text{attr}(\leqslant \max(\{x \in \mathbb{D} | x < a\}))$.

Для $\text{attr}(> a)$, $a \in \mathbb{D}$ есть аналогичные случаи. Если $a = n$, тогда дифференциальная функция, включающая его, является невыполнимой и ей равносильна дифференциальная функция $[\text{attr}(\leqslant m) \wedge \text{attr}(\geqslant n)]$. Если $a \neq n$, тогда этому ограничению равносильно ограничение $\text{attr}(\geqslant \min(\{x \in \mathbb{D} | x > a\}))$.

3.2.5. NED

В приведённой статье определение было дано неформально, формализуем.

Если обозначить предикат соседства в части NED слева от стрелки как LHS^{NED}, а справа, соответственно, RHS^{NED}, утверждение "p и q являются соседями относительно предиката NP" записать как предикат $\text{neighbors}_{\text{NP}}(p, q)$, то определение соблюдения NED на r выглядит так:

$$\forall (p, q) \in (r \times r) \quad (\text{neighbors}_{\text{LHS}^{\text{NED}}}(p, q) \implies \text{neighbors}_{\text{RHS}^{\text{NED}}}(p, q)).$$

NP сопоставляет атрибут пороговому значению, то есть является функцией, то есть множеством пар. Выражением θ_A обозначается функция, сопоставленная атрибуту A.

Тогда $\text{neighbors}_{\text{NP}}(p, q) \iff \forall (A, b) \in \text{NP} \quad \theta_A(p[A], q[A]) \geqslant b$. Запишем формальное определение NED.

Определение 38. NED соблюдается на $r \iff$

$$\forall (p, q) \in (r \times r) \quad (\forall (A, b) \in \text{LHS}^{\text{NED}} \quad \theta_A(p[A], q[A]) \geqslant b \implies \forall (A, b) \in \text{RHS}^{\text{NED}} \quad \theta_A(p[A], q[A]) \geqslant b).$$

Тогда для формы 1 зададим $\text{LHS} = \{(([]_A, []_A, \theta_A, \text{flip}(\geqslant)), b) | (A, b) \in \text{LHS}^{\text{NED}}\}$, $\text{RHS} = \{(([]_A, []_A, \theta_A, \text{flip}(\geqslant)), b) | (A, b) \in \text{RHS}^{\text{NED}}\}$

Множество зависимостей для задачи поиска может определяться с помощью задания соответствия между атрибутом и функцией для него, то есть отображения θ.

3.2.6. MD

Условие соблюдения для этого типа зависимости почти полностью соответствует предлагаемой форме, поэтому сразу укажем для формы 2 $\mathrm{LHS} = \{((\llbracket]_{A_i}, \llbracket]_{B_i}, \approx_i, \mathrm{flip}(\geqslant)), \lambda_i) | i \in (1 \dots m)\}$, $\mathrm{RHS} = ((\llbracket]_{A_j}, \llbracket]_{B_j}, \approx_j, \mathrm{flip}(\geqslant)), \lambda_j)$.

Способ задания множества поиска уже имеется — с помощью задания сопоставлений столбцов.

3.3. Полное пространство поиска

Теперь можно определить общий вид множества зависимостей, поиск минимального покрытия подмножества которого нужно выполнить.

В статье [5] это множество задаётся с помощью сопоставлений столбцов — тройки из меры сходства и двух столбцов. Выполняется поиск среди MD, которые имеют классификаторы столбцов, состоящими из этих сопоставлений и естественной границы решения (см. определение 31). Авторы статьи [5] не приводят обоснований для этого, поэтому приведём аргумент в пользу такого решения ниже.

Распространим понятие естественной границы решения на общий случай.

Определение 39. Для четвёрки $(T, V, F, \overset{\mathbf{L}}{\leq})$ и набора пар записей t назовём значение l **естественной границей решения**, если

$$\exists (p, q) \in t \quad F(T(p), V(q)) = l.$$

Определение 40. Для элемента $e = ((T, V, F, \overset{\mathbf{L}}{\leq}), l)$ обозначим через U_e следующее подмножество пар записей из рассматриваемых данных

$$U_e \coloneqq \{u | u \in t \wedge \mathrm{ElRst}[e, u]\} = \{(p, q) | (p, q) \in t \wedge l \overset{\mathbf{L}}{\leq} F(T(p), V(q))\}.$$

Назовём его множеством пар записей, выбираемых элементом e.

Рассмотрим форму выражения 2. Простыми преобразованиями с по-

мощью свойств подмножеств это выражение можно представить как $\bigcap_{e \in \text{LHS}} U_e \subseteq U_{\text{RHS}}$.

Факт 5. *Пусть a и b, $a \neq b$ — естественные границы решения на t для $H = (T, V, F, \overset{\mathbf{L}}{\leq})$, причём $\nexists c (a \overset{\mathbf{L}}{\leq} c \overset{\mathbf{L}}{\leq} b \wedge c$ — естественная граница решения).*

Обозначим $(a, b)_{\overset{\mathbf{L}}{\leq}} \coloneqq \{d | d \neq a \wedge d \neq b \wedge a \overset{\mathbf{L}}{\leq} d \overset{\mathbf{L}}{\leq} b\}$.

Тогда $\forall d \in (a, b)_{\overset{\mathbf{L}}{\leq}} \quad U_{(H,d)} = U_{(H,b)}$.

То есть между естественными границами решения нет таких значений, с помощью которых можно было бы выбрать пары записей “точнее”, чем задав значение в элементе зависимости как естественную границу решения.

Но пусть $\bar{l}$ — наибольшая естественная граница решения и пусть есть элемент, больший $\bar{l}$ — $\bar{l} \overset{\mathbf{L}}{\leq} \mathbf{1}$. Тогда имеется ещё одно множество, которое можно выбрать элементом зависимости e_1 с границей $\mathbf{1}$ — $U_{e_1} = \varnothing$.

Учитывая этот факт, обратим внимание на алгоритм HyMD (см. рис. 2). В нём поиск начинает с предположения, что все границы решения в правых частях зависимостей минимального покрытия равны 1.0, вне зависимости от того, является 1.0 естественной границей решения или нет. В ходе работы алгоритма это значение будет продвигаться дальше по решётке и окажется в ответе, если все критерии интересности отключены. То есть алгоритм решает не ту задачу, которая заявляется — в зависимости из ответа будет присутствовать граница решения, не являющаяся естественной.

На практике, однако, такие зависимости нужны нечасто, ведь они соблюдаются исключительно из-за отсутствия контр-примеров. Тем не менее, в случае поддержки целостности данных, который описан в обзоре, они могут быть полезны.

После этого замечания можно определять множество зависимостей, в котором будет вестись поиск.

Определение 41. Если $U_e = t$, то называем элемент зависимости e **тотальным**.

Общую форму задачи поиска сделаем подобной задаче для MD, то есть входным параметром являются наборы сопоставлений записей.

Определение 42. Пусть имеются наборы сопоставлений записей C_{LHS}, C_{RHS} и множество пар записей t.

Называем элемент зависимости из LHS корректным, если его первым элементом является сопоставление записей из набора C_{LHS}.

Называем элемент зависимости в RHS корректным, если его первым элементом является сопоставление записей из набора C_{RHS}.

Зависимость, у которой условие соблюдения приведено в форму 2, называем корректной, если её множество LHS является множеством корректных элементов и RHS является корректным элементом.

Назовём элементы зависимостей идентичными, если равны их сопоставления записей и множества пар записей, выбираемые этими элементами.

Назовём зависимость идентичной другой зависимости, если возможно представить их условия соблюдения в форме 2 так, что их RHS были идентичны и между нетотальными элементами их LHS существовала биекция, сопоставляющая идентичные элементы.

Назовём зависимость сокращённой, если в её LHS отсутствуют тотальные элементы.

Множество сокращённых нетривиальных зависимостей называем **полным пространством поиска**, если для любой нетривиальной корректной зависимости в этом пространстве поиска существует единственная идентичная ей зависимость.

Через $\overline{\text{dom}}(F)$ обозначим область определения функции F.

На полном пространстве поиска можно привести обобщённую форму порядка следования, использовав идею соответствующего порядка, определённого для MD:

Определение 43 (Общая форма порядка следования)**.** Пусть LHS, $\text{RHS} = (\text{RHS}^1, \text{RHS}_l)$ — значения переменных из условия соблюдения зависимости φ в форме 2, $\preceq_{\text{RHS}}$ — линейный порядок из RHS^1, LHS',

$\mathrm{RHS}' = (\mathrm{RHS}'^1, \mathrm{RHS}'_l)$ — аналогичные множества для зависимости φ'.
LHS *включает* (subsumes) LHS' — $\mathrm{LHS} \preceq \mathrm{LHS}' \iff$

$$\overline{\mathrm{dom}}(\mathrm{LHS}) \subseteq \overline{\mathrm{dom}}(\mathrm{LHS}') \wedge \forall((T, V, F, \overset{\mathbf{L}}{\leq}), l) \in \mathrm{LHS} \quad l \overset{\mathbf{L}}{\leq} \mathrm{LHS}'((T, V, F, \overset{\mathbf{L}}{\leq})).$$

Называем LHS *обобщением* LHS', LHS' — *специализацией* LHS.
φ *включает* φ' — $\varphi \preceq \varphi' \iff$

$$\mathrm{LHS} \preceq \mathrm{LHS}' \wedge \mathrm{RHS}^1 = \mathrm{RHS}'^1 \wedge \mathrm{RHS}'_l \preceq_{\mathrm{RHS}} \mathrm{RHS}_l.$$

Называем φ *обобщением* φ', φ' — *специализацией* φ.

Теперь определим общую форму задачу поиска.

Определение 44. Для некоторого типа зависимости задача поиска является задачей нахождения минимального покрытия подмножества соблюдающихся на данных зависимостей в полном пространстве поиска.

Факт 6. *Пусть имеется набор сопоставлений записей C и множество пар записей t.*

Назовём множеством дополненных естественных границ решения для некоторого сопоставления записей из C множество его естественных границ решения, дополненное элементом, большим его наибольшей естественной границы решения, если такой существует, иначе не дополняем множество им.

Множество нетривиальных зависимостей с условием соблюдения в форме 2, в которых LHS является функцией, сопоставляющей сопоставление записей из C одному из элементов его множества дополненных естественных границ решения, а в RHS вторым элементом является элемент такого множества для сопоставления записей в RHS, не имеющее идентичных зависимостей, является полным.

То есть в построении множества, по которому будет вестись поиск, естественные границы решения предлагается дополнить также границей решения, большей, чем наибольшая естественная, если такая существует.

3.4. Задачи поиска

В этом разделе определяются параметры задач поиска для рассматриваемых типов зависимостей. Имея их, задачи поиска можно определить согласно общим соображениям, приведённым в предыдущих разделах. Задача поиска для FD уже определена (см. Факт 2).

Также показываются вложения задач поиска. Вложение задач поиска для типов зависимости обозначим через $\hookrightarrow$.

Через $((p \in r) \mapsto Q(p))$ обозначаем функцию, которая применяется к p — записи r и возвращает $Q(p)$.

Через $(f \circ g)$ обозначаем композицию функций f и g.

3.4.1. FD $\hookrightarrow$ MFD

Параметры задачи поиска MFD:

1. Экземпляр отношения;
2. Сопоставление наборов атрибутов метрикам (θ).

Для этой зависимости в общей форме задачи поиска $C_{\text{LHS}} = \{([]_A, []_A, \text{eq}, R_{\text{bool}}) | A \in R\}$, $C_{\text{RHS}} = \{([]'_X, []'_X, d, \text{flip}(\leqslant)) | (X, d) \in \theta\}$.

Через $\rho_{\mathcal{D}}$ обозначим дискретную метрику.

Выражение FD через MFD уже приводится в [9] и указано в обзоре как Факт 1. Задача поиска FD сводится к задаче поиска MFD путём сопоставления каждого атрибута входного экземпляра дискретной метрике, то есть $\theta = \{(\{A\}, \rho_{\mathcal{D}}) | A \in R\}$.

3.4.2. MFD $\hookrightarrow$ DD

Параметры задачи поиска DD:

1. Экземпляр отношения;
2. Сопоставление атрибутов метрикам (d).

Для этой зависимости в общей форме задачи поиска $C_{\text{LHS}} = \{([]_A, []_A, d_A, \text{ord})|A \in R \wedge \text{ord} \in \{\leqslant, \geqslant\}\}$, $C_{\text{RHS}} = \{([]_A, []_A, d_A, \text{ord})|A \in R \wedge \text{ord} \in \{\leqslant, \geqslant\}\}$.

Для сведения задачи поиска MFD к задаче поиска DD нужно выполнить преобразование входных параметров. Для каждого набора атрибутов параметра 2 в MFD к таблице в параметре 1 нужно добавить столбец, содержащий все значения этих атрибутов, на вход задачи для DD нужно подавать эту изменённую таблицу. В d этим новым столбцам нужно сопоставить функцию, которая передаст их метрике из θ. Остальным столбцам нужно сопоставить дискретную метрику. Затем из результата нужно убрать зависимости, у которых в правой части атрибут, которому сопоставлена дискретная метрика.

То есть $d = \{(A, \rho_D)|A \in R\} \cup \{(\text{singleattr}(X), \text{fromsingle}(f))|(X, f) \in \theta\}$, где singleattr сопоставляет набору атрибутов некоторый новый атрибут, а fromsingle передаёт значение атрибута как если бы это было несколько значений. К каждой записи изначального r (для MFD) применяется makerecord $= (p \in r \mapsto p \cup \{(\text{singleattr}(X), \text{tosinglevalue}(p[X]))|(X, f) \in \theta\})$, где tosinglevalue создаёт из проекции записи новое значение.

3.4.3. DD $\hookrightarrow$ NED

Параметры задачи поиска NED:

1. Экземпляр отношения;

2. Сопоставление атрибутов функциям (θ).

Для этой зависимости в общей форме задачи поиска $C_{\text{LHS}} = \{([]_A, []_A, \theta_A, \text{flip}(\geqslant)|A \in R\}$, $C_{\text{RHS}} = \{([]_A, []_A, \theta_A, \text{flip}(\geqslant)|A \in R\}$.

Для DD значения расстояния между значениями каждого атрибута сравниваются с помощью двух линейных порядков, но в этой задаче каждому атрибуту можно сопоставлять только одну функцию. Поэтому у каждой записи экземпляра, который подаётся на вход задаче для DD, нужно скопировать столбцы.

А именно положим, newattr(A) — это функция, которая возвращает атрибут, отличный от атрибута A, результаты применения этой функции к разным атрибутам не равны ни друг другу, ни какому-либо атрибуту таблицы. Тогда к каждой записи подаваемой на вход задачи для DD таблицы применяем newrecord = $(p \in r \mapsto p \cup \{(\text{newattr}(A), p[A]) | A \in R\})$

В сведении DD к NED будет использоваться тот факт, что DD требует, чтобы функции, которые сопоставлены атрибутам, являлись метриками. Для NED такого требования нет.

Пусть $f_B : [0, +\infty) \to [0.0, 1.0]$ — некоторая обратимая функция, для которой выполнено $\forall a \in [0, +\infty) \quad \forall (p, q) \in (r \times r) \quad (d_B(p, q) \geqslant a \iff f_B(d_B(p, q)) \geqslant f_B(a))$, $g_B : [0, +\infty) \to [0.0, 1.0]$ — некоторая обратимая функция, для которой выполнено $\forall a \in [0, +\infty) \quad \forall (p, q) \in (r \times r) \quad (d_B(p, q) \leqslant a \iff g_B(d_B(p, q)) \geqslant g_B(a))$. Примерами $f_B(x)$ и $g_B(x)$ являются $1 - \frac{1}{x+1}$ и $\frac{1}{x+1}$ соответственно.

Тогда зададим $\theta = \{(B, f_B \circ h) | (B, h) \in d\} \cup \{(\text{newattr}(B), g_B \circ h) | (B, h) \in d\}$. Значения метрик, которые сопоставлялись атрибутам исходно, можно получить, применив соответствующие обратные к f_B и g_B функции.

3.4.4. NED ↪ MD

В общей постановке параметры задачи поиска MD:

1. Пара экземпляров отношений;
2. Сопоставления столбцов $C = \{(A_1, B_1, \approx_1), \ldots, (A_m, B_m, \approx_m)\}$ как для HyMD.

Для этой зависимости в общей форме задачи поиска $C_{\text{LHS}} = \{([]_A, []_B, \approx, \text{flip}(\geqslant) | (A, B, \approx) \in C\}$, $C_{\text{RHS}} = \{([]_A, []_B, \approx, \text{flip}(\geqslant) | (A, B, \approx) \in C\}$.

Этот случай очень простой, NED — это частный случай MD для одной таблицы и с одинаковыми атрибутами. Поэтому сведение тоже простое. Во-первых, на вход подаётся пара одинаковых экземпляров — оба из которых являются входным параметром 1 задачи по-

иска для NED. Во-вторых сопоставления столбцов определяются как $C = \{(A, A, \theta_A) | A \in R\}$.

4. MDE

Такое сведение зависимостей друг к другу на практике неудобно. Ранее в процессе оптимизации алгоритма HyMD [8] была реализована новая версия этого алгоритма. Некоторые её элементы (например, использование индексов границ решения вместо их самих или процесс валидации зависимости) указывают на возможность небольших модификаций в коде алгоритма, которые позволили бы решать задачу поиска в общем виде, а значит решать её для всех рассмотренных выше типов зависимостей.

Если эти модификации включить в реализацию, то получится алгоритм, который ведёт поиск формально более широкого класса зависимостей. Формы выражения 2 и 1 в некотором смысле сами задают некоторый тип зависимостей. В этом разделе данный тип зависимости формализуется.

Понять мотивацию определения можно, посмотрев на соответствующие определения из [5]. Для следующих определений будут указываться соответствующие определения для MD, если такие есть.

4.1. Основы

MDE (Matching Dependency Extended, расширенные сопоставляющие зависимости) — обобщение MD. Определяется на двух отношениях, обозначения как и выше: R и S — первое и второе отношения, r и s — соответствующие экземпляры. Поиск MD есть подзадача поиска MDE и любую MDE можно извлечь с помощью немодифицированного HyMD, изменяя таблицу и подбирая правильные сопоставления столбцов.

Определение 45. *Функция разбиения* (partitioning function) — $T : \mathrm{dom}(J) \to X$, где $\mathrm{dom}(J)$ — записи некоторого отношения ($J \in \{R, S\}$), X — множество результатов.

Данное определение является обобщением получения значения атрибута в MD. Автором выбрано такое название, поскольку в процессе валидации зависимости в HyMD записи разбиваются на множества,

каждое из которых соответствует значениям некоторых атрибутов. Для этой зависимости такое разбиение может зависеть не от значений в записи напрямую, а, например, от какой-то их комбинации.

Определение 46. *Функция сравнения* (comparison function) — $F : X \times Y \to \mathbf{L}$, X и Y — значения некоторых функций, $\mathbf{L}$ — некоторое множество.

В определении MD функции сравнения соответствует мера сходства, $\mathbf{L} = [0.0, 1.0]$.

Определение 47. *Сопоставление записей* (record match) — четвёрка $(T, V, F, \overset{\mathbf{L}}{\leq})$ из двух функций разбиения $T : \mathrm{dom}(R) \to X$ и $V : \mathrm{dom}(S) \to Y$, функции сравнения $F : X \times Y \to \mathbf{L}$ и линейного порядка $\overset{\mathbf{L}}{\leq}$ на $\mathbf{L}$.

Это расширение понятия сопоставления столбцов для MD. В MD порядок $\overset{\mathbf{L}}{\leq} = \leqslant$.

Считаем, что в каждом $\mathbf{L}$ есть наименьший и наибольший элементы относительно $\overset{\mathbf{L}}{\leq}$. Обозначаем их как $\underline{l_{\mathbf{L}}}$, $\overline{l_{\mathbf{L}}}$. Если такие отсутствует, множество и порядок можно дополнить. Для MD $\underline{l_{\mathbf{L}}} = 0.0$, $\overline{l_{\mathbf{L}}} = 1.0$.

Определение 48. *Сопоставляющий классификатор* (matching classifier) — пара из сопоставления записей и возможного значения $l \in \mathbf{L}$ функции из него, называемого границей решения (decision boundary).

Эти определения аналогичны определениям сопоставления столбцов и классификатора сходства для MD соответственно.

Определение 49. Граница решения l называется *естественной* для сопоставления записей $(T, V, F, \overset{\mathbf{L}}{\leq})$ (natural decision boundary), если $\exists (p, q) \in (r \times s) \quad F(T(p), V(q)) = l$.

Эта идея не представлена в качестве логической формулы в статье [5] про HyMD, но в ней ведётся работа только с такими границами решения, соответствующая формула явно указана в работе [8] об оптимизации этого алгоритма.

Определение 50. Записи $(p, q) \in (r \times s)$ *сопоставлены* (matched) классификатором $((T, V, F, \overset{\mathbf{L}}{\leq}), l)$, $(l \in \mathbf{L})$, если $l \overset{\mathbf{L}}{\leq} F(T(p), V(q))$.

Это понятие аналогично одноимённому понятию для MD.

Определение 51. Сопоставляющий классификатор $((T, V, F, \overset{\mathbf{L}}{\leq}), l)$ назовём *всеобщим* (universal classifier), если его значение является наименьшим в $\mathbf{L}$ $(l = \underline{l_{\mathbf{L}}})$.

Определение 52. Сопоставляющий классификатор $((T, V, F, \overset{\mathbf{L}}{\leq}), l)$ назовём *тотальным* (total classifier), если он сопоставляет все пары записей из $(r \times s)$: $\forall (p, q) \in (r \times s) \quad l \overset{\mathbf{L}}{\leq} F(T(p), V(q))$.

Определение 53. Сопоставляющий классификатор $((T, V, F, \overset{\mathbf{L}}{\leq}), l)$ назовём *наиболее ограничивающим* (most restrictive classifier), если его значение является наибольшим в $\mathbf{L}$ $(l = \overline{l_{\mathbf{L}}})$.

В MD аналогом всеобщего классификатора является классификатор с границей решения, равной 0.0, наиболее ограничивающего — с границей, равной 1.0.

Тотальные классификаторы появляются на этапе предварительной обработки, когда определяются естественные границы решения. Если такая граница всего одна для некоторого сопоставления столбцов, его можно обрабатывать отдельно от остальных[2].

Границу решения, которая для некоторого сопоставления записей составит тотальный сопоставляющий классификатор, назовём *тотальной*. Аналогично определим *универсальную* границу решения.

Понятие всеобщего классификатора является более сильным, чем тотального. Для любой пары таблиц всеобщий классификатор является тотальным, обратное неверно.

Алгоритм поиска работает с классификаторами, имеющими естественные границы решения, и наиболее ограничивающим классификатором[3].

[2] См. реализацию HyMD в https://github.com/Desbordante/desbordante-core

[3] См. реализацию в https://github.com/Desbordante/desbordante-core

Определение 54. Пусть X — конечное множество сопоставляющих классификаторов, являющееся функцией. Записи (p, q) *сопоставлены* X ($(p, q) \models X$), если они сопоставлены каждым из классификаторов в нём.

Условие функциональности — формальность, нужная для упрощения определения, чтобы одному сопоставлению записей не соответствовало несколько границ решения. В контексте этой зависимости всё равно будет иметь значение только наибольшая граница. Условие конечности продиктовано практическими соображениями.

Определение 55. MDE φ определяется как пара из конечного множества сопоставляющих классификаторов X, являющегося функцией, (левой части, LHS) и ещё одного сопоставляющего классификатора $Y = ((T_Y, V_Y, F_Y, \overset{\mathbf{L}}{\leq_Y}), l_Y)$ (правой части, RHS).

MDE φ *соблюдается* на (r, s) $\iff$

$$\forall (p, q) \in (r \times s) \quad (\forall ((T, V, F, \overset{\mathbf{L}}{\leq}), l) \in X \quad l \overset{\mathbf{L}}{\leq} F(T(p), V(q)) \implies l_Y \overset{\mathbf{L}}{\leq_Y} F_Y(T_Y(p), V_Y(q)).$$

Определение 56. Называем левую часть X невыполнимой (infeasible), если $\nexists (p, q)$ $((p, q) \models X)$.

Это определение — аналог невыполнимой дифференциальной функции.

Дальше в тексте MDE с левой частью X и правой частью Y может обозначаться как $\varphi(X, Y)$, Y_l — граница решения правой части для неё, Y^1 — сопоставление записей в её правой части, $\leq_Y$ — линейный порядок из Y^1.

Зададим частичный порядок на MDE.

Определение 57. Пусть $\varphi(X, Y)$, $\varphi'(X', Y')$ — MDE. X *включает* (subsumes) X' — $X \preceq X'$ $\iff$

$$\overline{\mathrm{dom}}(X) \subseteq \overline{\mathrm{dom}}(X') \wedge \forall ((T, V, F, \overset{\mathbf{L}}{\leq}), l) \in X \quad l \overset{\mathbf{L}}{\leq} X'((T, V, F, \overset{\mathbf{L}}{\leq})).$$

Называем X *обобщением* X', X' — *специализацией* X.
φ *включает* φ' — $\varphi(X, Y) \preceq \varphi'(X', Y') \iff$

$$X \preceq X' \wedge Y^1 = Y'^1 \wedge Y'_l \preceq_Y Y_l.$$

Называем φ *обобщением* φ', φ' — *специализацией* φ.

Эти определения аналогичны соответствующим определениям для MD.

4.2. Интересность

Аналогично ранее описанным типам зависимостей, не все зависимости, определённые таким образом, дают информацию, которая может быть полезна. Далее определяются критерии, согласно которым некоторые MDE могут отсекаться из пространства поиска, последствием чего станет уменьшение его и, соответственно, времени работы алгоритма, который выполняет поиск. Критерии унаследованы от критериев, определённых для задачи поиска MD в работе [5].

Сначала исключаем зависимости, которые не дают вообще никакой информации.

Определение 58. MDE *тривиальна*, если она соблюдается вне зависимости от данных.

В частности, MDE $\varphi(X, Y)$ тривиальна, если в $\overline{\text{dom}}(X)$ присутствует сопоставляющий классификатор с тем же сопоставлением записей, что в правой части, и граница решения в Y в правой части предшествует границе решения в X, то есть если $Y = (M, l)$, то $M \in \overline{\text{dom}}(X) \wedge l \overset{\mathbf{L}}{\preceq} X(M)$. Это аналогично тривиальности для MD.

При этом можно заметить, что под определение выше попадают и другие случаи. Например, если значения равны, то точно известно, что расстояние Левенштейна между ними равно нулю.

Зависимость, у которой классификатор в RHS универсальный, очевидно, тривиальна. Это аналогично MD с нулевой границей решения в правой части.

Определение 59. *Мощность* (cardinality) MDE — количество классификаторов в левой части, т.е. $|X|$.

Определение 60. *Поддержка* (support) σ MDE $\varphi(X, Y)$ — количество пар записей, сопоставляемых LHS MD.

$$\sigma(\varphi) \coloneqq |\{(p, q) \in (r \times s) | (p, q) \models X\}|$$

Далее следуют формулировки критериев интересности из MD (2.5) для MDE:

1. Мощность — предлагается не рассматривать зависимости со слишком большой мощностью, так как они могут быть сложны для понимания;

2. Поддержка — предлагается не рассматривать зависимости со слишком маленькой поддержкой, так как такие зависимости с меньшей вероятностью "имеют смысл" и могут соблюдаться только из-за отсутствия контрпримеров;

3. Отсутствие пересечения — для некоторых случаев использования зависимости, у которых сопоставление записей из правой части присутствует в левой ($Y = (M, l) \wedge M \in \overline{\mathrm{dom}}(X)$), не нужны, поэтому они могут исключаться;

4. Величина границы решения — чем меньше граница решения относительно соответствующего линейного порядка, тем больше записей сопоставляется классификатором. Для случая, когда маленькие границы решения не нужны, можно исключать зависимости с такими границами решения из пространства поиска;

5. Точность границ решения в левой части — пространство поиска можно значительно уменьшить, если исключить зависимости с некоторыми границами решения в левой части из него.

Задача поиска для этого типа зависимости буквально совпадает с общей формой задачи поиска.

5. HyMDE

В этом разделе описывается алгоритм решения задачи поиска MDE на паре таблиц. MDE и MD похожи и, как было упомянуто выше, для поиска этой зависимости достаточно модифицировать HyMD. Поэтому здесь описываются только его модификации в общих чертах.

Для удобства работы с различными типами значений функций сравнения на этапе предварительной обработки сначала выявляются дополненные естественные границы решения, которые сортируются и записываются в список. Далее в алгоритме используются используются только индексы границ решения.

5.1. Структуры данных

Структуры данных, используемые в алгоритмах, похожи. В каждом индексе списков позиций задаётся разбиение записей таблицы по значениям на множества (кластеры) в записи соответствующего столбца. Для поиска MDE используется подобная структура, но разбиение записей таблицы задаётся по значениям функций разбиения. Название соответствует смыслу, поэтому оно оставлено таким же.

В структуре “словарно сжатые записи” для каждой записи хранятся идентификаторы её значений. В HyMD они используются для группировки записей при валидации и как индексы в следующие две структуры. По смыслу каждый идентификатор указывает на кластер в предыдущей структуре. Для поиска MDE используется аналогичная структура, где записи сопоставлен её номер кластера в первой структуре для каждого сопоставления записей. Называем её “кластеры записей” (record clusters).

Структура “матрица сходства” сопоставляет паре значений результат меры сходства. Здесь вместо идентификаторов значений в качестве ключей нужно использовать, опять же, идентификатор кластера в индексе списков позиций. Так как в описании этой зависимости сходства больше не упоминаются, используем более общее название — “матрица значений” (value matrix).

Структура “индекс сходства” сопоставляет значению и границе решения множество записей, содержащих достаточно похожее значение в правильном атрибуте. “Достаточно похожее” здесь означает большее или равное некоторому сходству. В более общих терминах это означает следующее или предшествующее согласно некоторому частичному порядку. Для этой зависимости множество будет выглядеть как $\{q|a \overset{\mathbf{L}}{\leq} F(T(p), V(q))\}$. Это напоминает понятие верхнего множества и при определённом задании порядка то множество, которое возвращает эта структура данных, буквально является верхним множеством. Поэтому используем название “индекс верхних множеств” (upper set index).

5.2. Изменения в HyMD

В этапе предварительной обработки произошли следующие изменения:

- границы решения, по которым ведётся поиск, возвращаются явно;
- разбиение таблицы происходит не по значениям в столбцах, а по значениям функций.

Обход решётки и выведение из пар записей остались без значительных изменений, как и валидация и сэмплирование.

6. Эксперименты

Были проведены тесты на корректность путём сравнения результатов с уже реализованным алгоритмом HyMD и задания сопоставлений записей так, чтобы симулировать поиск MD. В таблице 1 указаны их параметры.

Стандартная конфигурация:

- Каждое сопоставление записей состоит из:
  - Функций разбиения, которые возвращают значение атрибута;
  - Нормализованного расстояния Левенштейна;
  - Порядка $\leqslant$;
- Рассматриваются только зависимости с поддержкой большей, чем количество записей;
- Включён критерий отсутствия пересечения;
- Не рассматриваются зависимости, у которых величина границы решения в классификаторе записей меньше 0.7.

Полная конфигурация:

- Каждое сопоставление записей состоит из:
  - Функций разбиения, которые возвращают значение атрибута;
  - Нормализованного расстояния Левенштейна;
  - Порядка $\leqslant$;
- Критерии интересности отключены.

После исполнения алгоритмов с конфигурациями, указанными в таблице 1, их результаты совпали.

Таблица 1: Параметры конфигураций тестов

Набор данных	Строки	Конфигурация	Зависимости
animals_beverages	4	Стандартная	2
animals_beverages	4	Полная	25
adult	32561	Стандартная	111
CORA	1879	Стандартная	2151

Заключение

В ходе работы были исследованы типы зависимостей, использующие разностную семантику.

- Исследованы типы зависимостей MFD, DD, NED, MD, указаны способы сведения задач поиска минимального покрытия друг к другу;
- Определён тип зависимостей MDE;
- Разработан алгоритм HyMDE, решающий задачу поиска MDE.

Код алгоритма HyMDE доступен по ссылке: https://github.com/Desbordante/desbordante-core/pull/567.

Список литературы

[1] Abedjan Ziawasch, Golab Lukasz, Naumann Felix. Profiling relational data: a survey // The VLDB Journal. — 2015. — Aug.. — Vol. 24, no. 4. — P. 557–581. — URL: https://doi.org/10.1007/s00778-015-0389-y.

[2] Bassée Renaud, Wijsen Jef. Neighborhood Dependencies for Prediction // Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining. — PAKDD '01. — Berlin, Heidelberg : Springer-Verlag, 2001. — P. 562–567.

[3] Discovering Functional Dependencies through Hitting Set Enumeration / Tobias Bleifuß, Thorsten Papenbrock, Thomas Bläsius et al. // Proc. ACM Manag. Data. — 2024. — Mar.. — Vol. 2, no. 1. — 24 p. — URL: https://doi.org/10.1145/3639298.

[4] Efficient Differential Dependency Discovery / Shulei Kuang, Honghui Yang, Zijing Tan, Shuai Ma // Proc. VLDB Endow. — 2024. — May. — Vol. 17, no. 7. — P. 1552–1564. — URL: https://doi.org/10.14778/3654621.3654624.

[5] Efficient Discovery of Matching Dependencies / Philipp Schirmer, Thorsten Papenbrock, Ioannis Koumarelas, Felix Naumann // ACM Trans. Database Syst. — 2020. — Aug.. — Vol. 45, no. 3. — 33 p. — URL: https://doi.org/10.1145/3392778.

[6] Fan Wenfei. Dependencies revisited for improving data quality // Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. — 2008. — URL: https://api.semanticscholar.org/CorpusID:17134608.

[7] Grant J., Minker J. Numerical dependencies // Advances in Database Theory / Ed. by H. Gallaire, J. Minker, J.-M. Nicolas. — Plenum Publishing Co., 1984. — Vol. 2.

[8] Lightning Fast Matching Dependency Discovery with Desbordante / Alexey Shlyonskikh, Michael Sinelnikov, Daniil Nikolaev et al. // 2024 36th Conference of Open Innovations Association (FRUCT). — 2024. — P. 729–740.

[9] Metric Functional Dependencies / Nick Koudas, Avishek Saha, Divesh Srivastava, Suresh Venkatasubramanian. — 2009. — 03. — P. 1275–1278.

[10] Papenbrock Thorsten, Naumann Felix. A Hybrid Approach to Functional Dependency Discovery // Proceedings of the 2016 International Conference on Management of Data. — SIGMOD '16. — New York, NY, USA : Association for Computing Machinery, 2016. — P. 821–833. — URL: https://doi.org/10.1145/2882903.2915203.

[11] Song Shaoxu, Chen Lei. Differential dependencies: Reasoning and discovery // ACM Trans. Database Syst. — 2011. — Aug.. — Vol. 36, no. 3. — 41 p. — URL: https://doi.org/10.1145/2000824.2000826.

[12] Ullman Jeffrey D. Principles of database and knowledge-base systems, Vol. I. — USA : Computer Science Press, Inc., 1988. — ISBN: 088175188X.