Сибирский математический журнал
Май—июнь, 2026. Том 67, № 3

УДК 519.682.1+519.683+519.7+519.1

ОБЪЕКТНО–ОРИЕНТИРОВАННЫЕ ДАННЫЕ И ПРЕФИКСНАЯ ПЕРЕЗАПИСЬ. ЧАСТЬ I: ОБЗОР И ОСНОВНЫЕ РЕЗУЛЬТАТЫ

А. Е. Гутман

Аннотация. Детерминированная префиксная перезаписывающая система представляет собой систему перезаписи строк, не содержащую пар правил вида $X \to Y$, $X \to Z$, где $Y \neq Z$, в которой перезаписи подлежат только самые длинные префиксы. Для таких систем определяются и исследуются аналоги понятий, характерных для объектно-ориентированных систем данных: наследование классов и объектов, экземпляры классов, атрибуты классов и экземпляров, концептуальная зависимость и непротиворечивость, концептуальная схема, типы, подтипы и др. Особое внимание уделяется алгоритмической проверке различных свойств рассматриваемых перезаписывающих систем.

DOI 10.33048/smzh.2026.67.302

Ключевые слова: префиксная перезапись, полутуэвская система, информационная система, объектно-ориентированная система данных, проверка непротиворечивости, онтология модели данных.

Введение

Классический объектно-ориентированный подход к описанию структурированных данных задействует два фундаментальных отношения: *has* («иметь») и *is* («быть»).

Отношение *has* связывает объекты и классы с их атрибутами. Соотношение «`X` *has a* `Y`» («`X` *имеет* `Y`») означает, что объект или класс `X` обладает атрибутом с именем `Y`, и тем самым можно говорить об «`Y` объекта `X`» как о свойстве объекта `X`, обычно обозначаемом через `X.Y`. Например, если веб-страница имеет кнопку отправки, стиль которой предполагает наличие рамки определенной ширины, то можно говорить о «ширине рамки стиля кнопки страницы», имея в виду объект `page.submitButton.style.border.width`.

Отношение *is* может использоваться для (1) создания экземпляров классов, (2) наследования классов от классов и (3) присваивания значений атрибутам. Соотношение «`40` *is an* `Integer`» («`40` — *это* `Integer`») связывает объект `40` с классом `Integer` и подразумевает, что число `40` является экземпляром класса `Integer`. Фраза «`Integer` *is* `Number`» означает, что класс `Integer` наследуется от класса `Number`. Утверждение «`Bob.age` *is* `40`» присваивает значение `40` атрибуту `age` объекта `Bob`.

Работа выполнена в рамках государственного задания ИМ СО РАН (проект № FWNF–2026–0022).

© **2026 Гутман А. Е.**

Как видно из приведенных выше примеров, широкая интерпретация отношения *is* позволяет устранить различие между объектами и классами. Единая система данных может объединять декларацию классов («метаданные») и создание объектов-экземпляров вместе с их инициализацией («данные»). Это не означает, что данные и метаданные лучше объединять, чем разделять, но такой подход позволяет унифицировать анализ данных и, в частности, открывает возможность для разработки общего инструмента проверки концептуальной и семантической непротиворечивости.

Отношения *is* и *has* естественно связаны между собой. Интерпретация отношения *is* как наследования или создания экземпляра класса предполагает, что в случае «`X` *is* `Y`» все атрибуты `Y` наследуются классом или объектом `X`. В частности, если «`X` *is* `Y`» и «`Y` *has a* `Z`», то «`X` *has a* `Z`». Более того, если «`X` *is* `Y`» — единственная явная информация об `X`, то можно заключить, что «`X.Z` *is* `Y.Z`». (Под явной информацией понимаются *is*-правила, образующие рассматриваемую систему данных.) При этом можно вывести *неявную* информацию об `X` и сказать, что «`X.Z` *is* `Y.Z` неявно». Таким образом, при означивании свойства `X.Z` производится перезапись, заменяющая в нем префикс `X` на `Y` согласно явному правилу «`X` *is* `Y`». Этот же принцип применим к конструкциям произвольной длины. Например, если явно известно, что «`Bob.Address` *is* `Alice.Work.Address`», то `Bob.Address.City.Code` получает неявную перезапись в виде `Alice.Work.Address.City.Code`.

Ясно, что явные правила имеют приоритет над любыми неявными выводами. Поэтому, если «`X` *is* `Y`», «`Y` *has a* `Z`» и система данных содержит явное правило «`X.Z` *is* `A`», то последнее правило имеет приоритет над неявным «`X.Z` *is* `Y.Z`». Возможен также конфликт иного рода — когда одновременно применимы несколько явных правил. В качестве примера рассмотрим следующий фрагмент системы данных:

```
block.style.color  is block
header.style.color is red
button             is block
button.style       is header.style
```

Попробуем выяснить, чему равен цвет в стиле кнопки, т. е. раскрыть значение `button.style.color`. Поскольку `button` — это `block`, было бы естественно ожидать, что `button.style.color` — это `block.style.color`, т. е. `blue`. Но, с другой стороны, `button.style` — это `header.style`, а следовательно, `button.style.color` — это `header.style.color`, т. е. `red`. Интуитивно последнее означивание должно иметь приоритет, поскольку правило «`button.style` *is* `header.style`», по-видимому, имеет преимущество перед «`button` *is* `block`». Причина состоит не в том, что одно из правил расположено после другого. (Система данных считается неупорядоченным множеством *is*-правил.) Ключевой момент здесь в том, что правило «`button.style` *is* `header.style`» более *конкретно*, так как оно означивает более длинный объект — `button.style`, а не `button`. Следовательно, при означивании следует *переписывать самый длинный префикс*, т. е. задействовать самое конкретное из применимых правил.

Теперь остановимся на *непротиворечивости данных*. Очевидно, при построении системы определений следует избегать концептуальных циклов. Фраза «`man` *is* `man`» ничего не определяет, поскольку означивание термина `man` приводит к бесконечному циклу. Вместе с тем концептуальная непротиворечивость не запрещает рекурсию. Например, правило «`man.son` *is* `man`» вполне

допустимо. С другой стороны, правило «`man` *is* `man.son`» выглядит некорректным: оно по-прежнему не проясняет смысл термина `man`, пока не определен атрибут `son` объекта `man`, а последний лишен смысла до определения `man`. Правила «`man` *is* `Adam`» и «`Adam.rib` *is* `man.rib`» тоже образуют противоречивую пару, поскольку `Adam.rib` есть `man.rib`, а последнее неявно переписывается в `Adam.rib`. Подобные примеры обосновывают необходимость формального определения концептуальной непротиворечивости и поиска соответствующей эффективной проверки. (Эта задача аналогична анализу онтологии системы данных как множества концептуальных определений.)

Всякое систематическое определение понятий, классов или объектов неизбежно подразумевает наличие хотя бы одного понятия, не требующего определения. Вообще говоря, первичных понятий может быть несколько, но теоретически достаточно иметь лишь один «универсальный объект». Обозначим его через ω. Слово X вида $\texttt{entity}\,.\,\texttt{attr}_1\,.\,\texttt{attr}_2\,.\cdots.\,\texttt{attr}_n$ в рамках рассматриваемой системы данных перезаписывается с применением наиболее конкретного *is*-правила, в результате чего возникает новое слово и процесс перезаписи самых длинных префиксов последующих слов продолжается до тех пор, пока не будет достигнуто слово ω. В этом случае можно заключить, что исходное слово X является *объектом* (или *понятием*). В противном случае, если процесс перезаписи либо (1) заканчивается неперезаписываемым словом, отличным от ω, либо (2) никогда не завершается, слово X считается лишенным смысла. Возможность (2) делает анализ нетривиальным и обосновывает поиск алгоритмической проверки осмысленности того или иного слова. (Эта задача аналогична анализу онтологии понятия в рамках системы данных.)

Рассматривая какой-либо объект X и последовательность атрибутов D вида $\texttt{attr}_1\,.\,\texttt{attr}_2\,.\cdots.\,\texttt{attr}_n$, будем говорить, что D является *деталью* объекта X, если слово $X.D$ имеет смысл. Множество $\|X\|$ всех деталей объекта X можно рассматривать как *тип* объекта X. (Такой подход известен как «утиная типизация».) Всякий раз, когда алгоритмическая процедура предполагает формальный аргумент A, тело процедуры обычно содержит вхождения слова A вместе с некоторыми словами вида $A.D_i$. Для корректной работы процедуры при подстановке объекта X вместо A необходимо (и, вероятно, достаточно), чтобы все слова $X.D_i$ имели смысл. Иначе говоря, объект X должен иметь подходящий тип. В этой связи было бы полезно иметь алгоритм сравнения типов объектов: для данных объектов X и Y необходимо научиться эффективно сравнивать типы X и Y, т. е. определять, какие из соотношений $\|X\| = \|Y\|$, $\|X\| \subseteq \|Y\|$, $\|X\| \supseteq \|Y\|$ имеют место. Эта задача нетривиальна хотя бы потому, что тип объекта может быть бесконечным. (Например, при наличии правила «`man.son` *is* `man`» тип объекта `man` содержит все слова `son`, `son.son`, `son.son.son`, ...)

Ниже будут приведены формальные определения рассматриваемых понятий, сформулированы основные результаты и описаны алгоритмы для всех упомянутых задач. Чтобы обозначения были менее громоздкими, будем считать имена сущностей и атрибутов отдельными символами (буквами) некоторого алфавита $\mathbb{A}$ и условимся записывать последовательности атрибутов $\alpha_1\,.\,\alpha_2\,.\cdots.\,\alpha_n$ в виде $\alpha_1\alpha_2\ldots\alpha_n$, превращая их тем самым в слова над $\mathbb{A}$. Явные правила «X *is* Y» будем записывать в виде $X \to Y$.

Поскольку материал данной работы опирается на вполне стандартные сведения из классических областей, в дальнейшем по ходу изложения приводят-

ся лишь явно используемые ключевые определения и факты. Более подробные сведения о системах префиксной перезаписи и регулярных системах читатель может найти в основополагающей работе Ю. Р. Бюхи о регулярных канонических системах [1], исследовании Д. Кокаля регулярной структуры префиксной перезаписи [2], монографии Р. В. Бука и Ф. Отто о системах перезаписи строк [3] и характеризации регулярных языков посредством префиксных грамматик, принадлежащей М. Фрейзеру и Д. Пейджу [4]. Что касается объектно-ориентированной парадигмы, полезными классическими источниками служат обзор Л. Карделли и В. Вегнера [5], манифест систем объектно-ориентированных баз данных М. Аткинсона и др. [6] и обзор К. Р. Диттриха понятий объектно-ориентированной модели данных [7]. Сведения, относящиеся к теории графов, можно почерпнуть из классической монографии Ф. Харари [8].

Данная статья расширяет набор определений и результатов, анонсированных в [9], и является первой в запланированной серии публикаций, посвященных представлению объектно-ориентированных данных в виде префиксных перезаписывающих систем. Статья имеет характер предваряющего обзора и не содержит доказательств утверждений и обоснований алгоритмов. Все детали, включая различные примеры, будут опубликованы в последующих частях серии.

Материал статьи сгруппирован в шести параграфах. В §1 приведены основные определения, касающиеся рассматриваемых перезаписывающих систем и интерпретации в них таких классических объектно-ориентированных понятий, как наследование, декларация члена класса, создание объекта-экземпляра класса и означивание свойства. Параграф 2 посвящен рассмотрению перезаписывающих систем как систем формальных определений и исследованию концептуальной непротиворечивости таких систем. Параграф 3 содержит интерпретацию понятий перекрывающего, добавленного и неявного атрибута, а также понятия декларирующего класса (источника) атрибута. В §4 вводятся и детально исследуются понятия типа и подтипа в их связи с наследованием классов и означиванием атрибутов. Параграф 5 посвящен эффективному анализу перезаписи и включает алгоритмы, проверяющие такие свойства системы, как отсутствие бесконечно перезаписываемых слов и концептуальная непротиворечивость. Параграф 6, в свою очередь, посвящен анализу структуры типов и содержит, в том числе, описание алгоритмов, позволяющих сравнивать типы объектов и находить конкретные детали, отличающие один тип от другого.

1. Детерминированная префиксная перезаписывающая система

1.1. На протяжении всей статьи $\mathbb{A}$ — конечный алфавит, $\mathbb{A}^*$ (соответственно, $\mathbb{A}^+$) — множество всех (соответственно, всех непустых) слов над $\mathbb{A}$. Для краткости будем говорить «*слово*» вместо «непустое слово над $\mathbb{A}$». Элементы $\mathbb{A}$ называются *буквами*. Буквы традиционно отождествляются с соответствующими однобуквенными словами. Длина слова X обозначается через $|X|$. Символ $\mathbb{N}$ обозначает множество натуральных чисел $\{1, 2, \dots\}$.

1.2. Слово X будем называть *префиксом* (соответственно, *собственным префиксом*) слова $Y \in \mathbb{A}^+$ и писать $X \sqsubseteq Y$ или $Y \sqsupseteq X$ (соответственно, $X \sqsubset Y$ или $Y \sqsupset X$), если $X \in \mathbb{A}^+$ и $Y = XS$ для некоторого $S \in \mathbb{A}^*$ (соответственно, $S \in \mathbb{A}^+$).

Если $n \in \mathbb{N}$, $X \in \mathbb{A}^+$ и $|X| \geqslant n$, то символом $X\restriction_n$ условимся обозначать префикс слова X, имеющий длину n. Таким образом, слово $X\restriction_n \in \mathbb{A}^+$ определяется соотношениями $X\restriction_n \sqsubseteq X$ и $\left|X\restriction_n\right| = n$.

1.3. Если символ $\rightsquigarrow$ обозначает какое-либо бинарное отношение, то через $\overset{*}{\rightsquigarrow}$ будем обозначать рефлексивное транзитивное замыкание отношения $\rightsquigarrow$, а через $\overset{+}{\rightsquigarrow}$ — его транзитивное замыкание. Таким образом, запись $x \overset{*}{\rightsquigarrow} y$ (соответственно, $x \overset{+}{\rightsquigarrow} y$) означает, что

$$x = z_0 \rightsquigarrow z_1 \rightsquigarrow \cdots \rightsquigarrow z_n = y$$

для некоторых $z_1, \ldots, z_n$, где $n \geqslant 0$ (соответственно, $n \geqslant 1$). В частности, соотношение $x \overset{*}{\rightsquigarrow} y$ справедливо тогда и только тогда, когда $x = y$ или $x \overset{+}{\rightsquigarrow} y$.

1.4. Рассмотрим произвольное бинарное отношение $\to$ на $\mathbb{A}^+$ (т. е. подмножество декартова квадрата $\mathbb{A}^+ \times \mathbb{A}^+$) и букву $\omega \in \mathbb{A}$. Пару $\langle \to, \omega \rangle$ назовем *детерминированной префиксной перезаписывающей системой*, или для краткости *системой*, если множество $\to$ конечно, непусто и удовлетворяет следующим условиям:

(a) из $X \to Y$ и $X \to Z$ следует $Y = Z$;

(b) не существует таких $S, Y \in \mathbb{A}^*$, что $\omega S \to Y$.

В дальнейшем, давая определения и формулируя утверждения, по умолчанию считаем фиксированной произвольную детерминированную префиксную перезаписывающую систему $\langle \to, \omega \rangle$, относительно которой интерпретируются вводимые понятия, а также используемые термины и обозначения.

1.5. Положим

$$\mathbb{E} := \{X : X \to Y \text{ для некоторого } Y\}$$

и условимся называть элементы множества $\mathbb{E}$ *явными словами*. Символом $\mathbb{A}_{\mathbb{E}}$ обозначается *явный алфавит* — множество всех букв, встречающихся в явных словах:

$$\mathbb{A}_{\mathbb{E}} := \min\{A \subseteq \mathbb{A} : \mathbb{E} \subseteq A^+\}.$$

Кроме того, определим число μ как длину самого длинного явного слова:

$$\mu := \max\{|E| : E \in \mathbb{E}\}.$$

1.6. Будем говорить, что слово E является *явным префиксом* слова X, если $E \in \mathbb{E}$ и $E \sqsubseteq X$. Слово X назовем *перезаписываемым*, если X имеет хотя бы один явный префикс.

Как легко видеть, условие 1.4(a) означает, что для каждого $E \in \mathbb{E}$ существует единственное слово E', удовлетворяющее соотношению

$$E \to E',$$

а условие 1.4(b) равносильно тому, что слова вида ωS, где $S \in \mathbb{A}^*$, не являются перезаписываемыми.

1.7. Для перезаписываемого слова X рассмотрим его самый длинный явный префикс E, определим соответствующий суффикс $S \in \mathbb{A}^*$, для которого $X = ES$, и положим

$$X' := E'S,$$

где $E \to E'$ (см. п. 1.6). Слово X' будем называть *результатом перезаписи* слова X. Введем бинарное отношение $\Rightarrow$ на $\mathbb{A}^+$, полагая

$$X \Rightarrow Y \text{ тогда и только тогда, когда } X \text{ перезаписываемо и } X' = Y.$$

1.8. Рекурсивно определим

$$\begin{aligned}
&\mathbb{W}_0 := \mathbb{A}^+,\\
&X^{(0)} := X \ \text{ для } X \in \mathbb{W}_0;\\
&\mathbb{W}_n := \{X \in \mathbb{W}_{n-1} : X^{(n-1)} \text{ перезаписываемо}\},\ n \geqslant 1,\\
&X^{(n)} := (X^{(n-1)})' \ \text{ для } X \in \mathbb{W}_n,\ n \geqslant 1.
\end{aligned}$$

Слово $X^{(n)}$ назовем *n-м результатом перезаписи* слова X. Таким образом, для каждого $X \in \mathbb{W}_n$ справедливы соотношения

$$\begin{aligned}
&X = X^{(0)} \Rightarrow X^{(1)} \Rightarrow \cdots \Rightarrow X^{(n)};\\
&X \stackrel{*}{\Rightarrow} X^{(n)} \text{ при } n \geqslant 0,\\
&X \stackrel{+}{\Rightarrow} X^{(n)} \text{ при } n > 0.
\end{aligned}$$

Элементы множества $\bigcap_{n=1}^{\infty} \mathbb{W}_n$ называются *бесконечно перезаписываемыми словами*. Остальные слова называются *конечно перезаписываемыми*. Для данного слова X максимальную (конечную или бесконечную) последовательность вида

$$\langle X^{(0)}, X^{(1)}, X^{(2)}, \ldots \rangle$$

будем называть *последовательностью перезаписи* слова X. Таким образом, слово X конечно (бесконечно) перезаписываемо тогда и только тогда, когда последовательность перезаписи слова X конечна (бесконечна).

1.9. Будем говорить, что слово $X \in \mathbb{A}^+$ является *объектом*, если $X \stackrel{*}{\Rightarrow} \omega$. Символом $\mathbb{O}$ обозначим множество всех объектов:

$$\mathbb{O} := \{X \in \mathbb{A}^+ : X \stackrel{*}{\Rightarrow} \omega\}.$$

Как легко видеть, последовательность перезаписи всякого объекта $X \in \mathbb{O}\backslash\{\omega\}$ имеет вид

$$X = X^{(0)} \Rightarrow \cdots \Rightarrow X^{(n)} \to \omega, \quad n \geqslant 0.$$

Согласно условию 1.4(b) универсальный объект ω не является перезаписываемым, причем ω является единственным неперезаписываемым объектом.

1.10. Из приведенных выше обозначений и определений ясно, что система $\langle \to, \omega \rangle$ рассматривается как префиксная перезаписывающая система, причем перезаписи подлежат только самые длинные префиксы слов. При этом система интерпретируется как распознающее устройство, для которого множество $\mathbb{O}$ служит распознаваемым языком (см. [10]). Такая система названа «детерминированной», поскольку всякое перезаписываемое слово имеет единственный результат перезаписи.

Введение понятия объекта можно рассматривать как отделение «понятий» от «бессмысленных слов». Объект — это слово X, имеющее «смысл», а именно результат перезаписи $X' = X^{(1)}$, который тоже имеет смысл — $(X')' = X^{(2)}$, и так далее вплоть до последнего результата перезаписи — «универсального объекта» ω, смысл которого предполагается заданным заранее. Отношение $\to$ тем самым трактуется как концептуальное определение, а правило $X \to Y$ рассматривается как определение понятия X: «X — это Y». Правило $X\alpha \to Z$ является определением свойства: «свойство α объекта X — это Z», а правило $X\alpha\beta \to Z$ служит определением «свойство β свойства α объекта X — это Z» и т.д.

В этом смысле условие 1.4(a), налагаемое на отношение $\to$, означает концептуальную однозначность (понятие не может иметь несколько различных смыслов). Условие 1.4(b) означает, что универсальный объект ω не имеет осмысленных атрибутов.

Отношение $\to$ можно также трактовать как объектно-ориентированное наследование или создание экземпляра и рассматривать правило $X \to Y$ как явное указание на тот факт, что «класс X непосредственно наследуется от класса Y» или «объект X является экземпляром класса Y». Далее, правило $X\alpha \to Z$ можно рассматривать как декларацию атрибута или означивание свойства: «класс X содержит атрибут α, значения которого принадлежат классу Z», или «свойство $X\alpha$ имеет значение Z», или «свойство $X\alpha$ является экземпляром класса Z». В этом смысле условие 1.4(a), налагаемое на отношение $\to$, запрещает множественное наследование (и, следовательно, ни один объект не может принадлежать нескольким несравнимым классам). Кроме того, согласно условию 1.4(b) универсальный объект ω не обладает никакими декларируемыми свойствами.

1.11. Пусть $\langle \mathbb{O}, \Leftarrow \rangle$ — ориентированный граф, вершинами которого являются всевозможные объекты, а дугами служат пары $\langle X, Y \rangle$ объектов, удовлетворяющих соотношению $Y \Rightarrow X$.

Теорема. *Граф $\langle \mathbb{O}, \Leftarrow \rangle$ является деревом с конечными уровнями.*

Дерево $\langle \mathbb{O}, \Leftarrow \rangle$ будем называть *деревом наследования*. Корнем этого дерева служит универсальный объект ω. Отметим, что дерево наследования может быть бесконечным и даже не иметь листьев.

1.12. Введем бинарное отношение $\Rightarrow_{\mathrm{w}}$ на $\mathbb{A}^+$, полагая

$$X \Rightarrow_{\mathrm{w}} Y \text{ тогда и только тогда, когда } X = ES \text{ и } Y = E'S \text{ для некоторых } E \in \mathbb{E} \text{ и } S \in \mathbb{A}^*.$$

Таким образом, отношение $\Rightarrow_{\mathrm{w}}$ представляет собой перезапись, соответствующую системе $\langle \to, \omega \rangle$, рассматриваемой как обычная префиксная перезаписывающая система, а не как система с перезаписью самых длинных префиксов. Ясно, что из $X \to Y$ следует $X \Rightarrow Y$, а из $X \Rightarrow Y$ следует $X \Rightarrow_{\mathrm{w}} Y$. Поэтому формулы $X \overset{+}{\Rightarrow} Y$ и $X \overset{+}{\Rightarrow}_{\mathrm{w}} Y$ можно прочитывать как «X *перезаписывается в* Y» и «X *слабо перезаписывается в* Y». Формулу $X \to Y$ можно снабдить прочтением «X *явно перезаписывается в* Y».

1.13. Введем бинарное отношение $\rightarrowtail$ на $\mathbb{A}^+$, полагая

$$X \rightarrowtail Y \text{ тогда и только тогда, когда } X \sqsupset Y \text{ или } X \Rightarrow_{\mathrm{w}} Y.$$

Как легко видеть, транзитивное замыкание $\overset{+}{\rightarrowtail}$ является наименьшим транзитивным отношением на $\mathbb{A}^+$, обладающим следующими тремя свойствами для всех $X, Y, S \in \mathbb{A}^+$:

(a) если $X \to Y$, то $X \overset{+}{\rightarrowtail} Y$;

(b) если $X \to Y$, то $XS \overset{+}{\rightarrowtail} YS$;

(c) $XS \overset{+}{\rightarrowtail} X$.

В случае $X \overset{+}{\rightarrowtail} Y$ будем говорить, что X *концептуально зависит от* Y. Слово X назовем *корректно определенным*, если X концептуально не зависит от X.

2. Концептуальная непротиворечивость

2.1. Будем говорить, что рассматриваемая система $\langle \to, \omega \rangle$ *концептуально непротиворечива*, если все слова корректно определены, т. е. ни одно слово концептуально не зависит от самого себя. Для краткости дадим имя этому условию:

Система концептуально непротиворечива. (**Con**)

Приведенная выше терминология оправдывается неформальной трактовкой правила перезаписи $X \to Y$ как определения X через Y («X — это Y») и трактовкой правила $XS \to Z$ как определения детали («деталь S объекта X — это Z»). Поэтому неформально отношение $X \overset{+}{\rightarrowtail} Y$ можно понимать следующим образом: определение X явно или неявно задействует Y, а значит, при последовательном описании концептуальной схемы понятие Y должно быть введено раньше X, иначе X окажется некорректно определенным.

2.2. Слово X назовем *рекуррентным* (соответственно, *слабо рекуррентным*), если $X \overset{+}{\Rightarrow} XS$ (соответственно, $X \overset{+}{\Rightarrow}_{\mathrm{w}} XS$) для некоторого $S \in \mathbb{A}^*$.

2.3. Предложение. *Для любых* $X, Y \in \mathbb{A}^+$

$X \overset{+}{\rightarrowtail} Y$ *тогда и только тогда, когда* $X \sqsupset Y$ *или* $X \overset{+}{\Rightarrow}_{\mathrm{w}} YS$ *для некоторого* $S \in \mathbb{A}^*$.

2.4. Следствие. *Слово корректно определено тогда и только тогда, когда оно не является слабо рекуррентным.*

2.5. Для произвольного символа $\rightsquigarrow$, обозначающего какое-либо бинарное отношение на $\mathbb{A}^+$, положим

$$|\rightsquigarrow| := \{X \in \mathbb{A}^+ : E \overset{*}{\rightsquigarrow} X \text{ для некоторого } E \in \mathbb{E}\}$$

и обозначим через $[\rightsquigarrow]$ ориентированный граф, вершинами которого являются слова из $|\rightsquigarrow|$, а дугами служат пары $\langle X, Y \rangle$, где $X, Y \in |\rightsquigarrow|$ и $X \rightsquigarrow Y$.

Условимся называть граф $[\rightarrowtail]$ *концептуальной схемой*, а граф $[\Rightarrow_{\mathrm{w}}]$ — *схемой слабой перезаписи* рассматриваемой системы. Поскольку из соотношения $X \Rightarrow_{\mathrm{w}} Y$ следует $X \rightarrowtail Y$, схема слабой перезаписи $[\Rightarrow_{\mathrm{w}}]$ является подграфом концептуальной схемы $[\rightarrowtail]$.

2.6. Теорема. *Следующие свойства системы равносильны:*
(1) *все слова корректно определены, т. е. выполнено условие* (Con);
(2) *каждое явное слово корректно определено;*
(3) *не существует слабо рекуррентных явных слов;*
(4) *не существует слабо рекуррентных слов;*
(5) *концептуальная схема ациклична;*
(6) *концептуальная схема ациклична и конечна;*
(7) *схема слабой перезаписи ациклична и конечна.*

2.7. Введем следующее именованное условие:

Не существует рекуррентных слов. (**Rec**)

2.8. Предложение. *Из* (Con) *следует* (Rec).

2.9. Теорема. *Если все слова* $X \in \mathbb{A}^+_{\mathbb{E}}$ *длины* $|X| \leqslant \mu$ *не являются рекуррентными, то все слова не являются рекуррентными, т. е. выполнено* (Rec).

2.10. Пусть $B(\mathscr{S})$ — множество слов, определенное по системе $\mathscr{S}$ некоторым условием. Будем говорить, что $B(\mathscr{S})$ является *базисом рекуррентности*, если для произвольной системы $\mathscr{S}$ из нерекуррентности всех слов, принадлежащих $B(\mathscr{S})$, следует нерекуррентность всех слов. Теорема 2.9 утверждает, что множество $\{X \in \mathbb{A}^+_{\mathbb{E}} : |X| \leqslant \mu\}$ является базисом рекуррентности. Несмотря на свою конечность, это множество может быть довольно объемным. Автору неизвестны условия, которые определяли бы существенно меньшие базисы рекуррентности. (Имеются примеры, показывающие, что ни множество $\mathbb{E}$ явных слов, ни множество всех префиксов явных слов не могут служить базисом рекуррентности.)

2.11. Введем следующее именованное условие:

Все слова конечно перезаписываемы. (**Fin**)

2.12. Теорема. *Если все явные слова конечно перезаписываемы, то все слова конечно перезаписываемы, т. е. выполнено* (Fin).

2.13. Теорема. *Из* (Rec) *следует* (Fin).

Таким образом, согласно предложению 2.8 и теореме 2.13 из (Con) следует (Rec), а из (Rec) следует (Fin). Как показывают примеры, обратные импликации в общем случае неверны.

2.14. Введем следующие два именованных условия:

Все явные слова являются объектами, т. е. $\mathbb{E} \subseteq \mathbb{O}$. (**Obj**)

Если $X \in \mathbb{A}^+$, $\alpha \in \mathbb{A}$ и $X\alpha \in \mathbb{E}$, то $X \in \mathbb{O}$. (**PreObj**)

2.15. Теорема. *Предположим, что выполнено* (Obj). *Пусть* $X, Y \in \mathbb{A}^+$, $X \to Y$, *и пусть буква* $\alpha \in \mathbb{A}\backslash\{\omega\}$ *встречается в* Y. *Тогда* $\alpha \in \mathbb{A}_{\mathbb{E}}$.

Таким образом, при выполнении (Obj) все буквы, отличные от ω и встречающиеся в правилах системы, принадлежат явному алфавиту.

2.16. Предложение.
(1) *Пусть* $X, Y \in \mathbb{A}^+$ *и* $X \overset{*}{\Rightarrow} Y$. *Тогда включение* $X \in \mathbb{O}$ *равносильно* $Y \in \mathbb{O}$.
(2) *Пусть выполнено* (Obj). *Для данного* $X \in \mathbb{A}^+$ *включение* $X \in \mathbb{O}\backslash\{\omega\}$ *имеет место тогда и только тогда, когда* $X \overset{*}{\Rightarrow} E$ *для некоторого* $E \in \mathbb{E}$.
(3) *Пусть выполнено* (PreObj). *Если* $X, S \in \mathbb{A}^+$ *и* $XS \in \mathbb{O}$, *то* $X \in \mathbb{O}$.

3. Атрибуты

3.1. Пусть $X \in \mathbb{O}$ и $\alpha \in \mathbb{A}$. Будем говорить, что α является *атрибутом* объекта X, если $X\alpha \in \mathbb{O}$. Объект $X\alpha$ в этом случае называется *свойством* (точнее, *свойством* α или α-*свойством*) объекта X. Множество всех атрибутов X обозначим через $\|X\|_1$. Из условия 1.4(b) ясно, что $\|\omega\|_1 = \varnothing$.

Букву α назовем *явным атрибутом* (соответственно, *неявным атрибутом*) объекта X, если $X\alpha \in \mathbb{O} \cap \mathbb{E}$ (соответственно, $X\alpha \in \mathbb{O}\backslash\mathbb{E}$).

Назовем α *перекрывающим атрибутом* (соответственно, *добавленным атрибутом*) объекта X, если $X\alpha \in \mathbb{O} \cap \mathbb{E}$ и, кроме того, X перезаписываемо и $X'\alpha \in \mathbb{O}$ (соответственно, $X'\alpha \notin \mathbb{O}$). Если α является перекрывающим атрибутом X, то будем говорить, что свойство $X\alpha$ *перекрывает* свойство $X'\alpha$.

Таким образом, всякий атрибут является либо явным, либо неявным, а всякий явный атрибут является либо перекрывающим, либо добавленным.

3.2. Предложение. *Для всех $X \in \mathbb{O}$ и $\alpha \in \mathbb{A}$ справедливы следующие утверждения:*

(1) *α является неявным атрибутом объекта X тогда и только тогда, когда X перезаписываемо, $X\alpha \notin \mathbb{E}$ и $X'\alpha \in \mathbb{O}$;*

(2) *α является добавленным атрибутом объекта X тогда и только тогда, когда X перезаписываемо, $X\alpha \in \mathbb{O}$ и $X'\alpha \notin \mathbb{O}$.*

3.3. Предложение. *Пусть выполнено* (Obj). *Если $X, Y \in \mathbb{A}^+$, $\alpha \in \mathbb{A}$, $X \overset{*}{\Rightarrow} Y$ и $Y\alpha \in \mathbb{O}$, то $X\alpha \in \mathbb{O}$.*

3.4. Предложение. *Пусть выполнено* (Obj). *Рассмотрим последовательность перезаписи*

$$X = X^{(0)} \Rightarrow \cdots \Rightarrow X^{(n)} = \omega$$

объекта X. Если $\alpha \in \|X\|_1$, то существует число $0 \leqslant i \leqslant n$ такое, что

$$X^{(0)}\alpha, \ldots, X^{(i)}\alpha \in \mathbb{O}, \quad X^{(i+1)}\alpha, \ldots, X^{(n)}\alpha \notin \mathbb{O}$$

и α является добавленным атрибутом объекта $X^{(i)}$.

3.5. Следствие. *Пусть выполнено* (Obj). *Буква α является атрибутом объекта X тогда и только тогда, когда $X^{(n)}\alpha \in \mathbb{E}$ для некоторого $n \geqslant 0$.*

3.6. Культура программирования обычно включает рекомендацию давать декларируемым сущностям как можно более содержательные имена в разумных пределах, чтобы каждый идентификатор помогал читателю догадываться о роли соответствующего объекта в данном фрагменте кода. В частности, не рекомендуется создавать омонимию внутри локального контекста — например, декларировать переменную или класс с тем же именем, что и у какого-либо атрибута (свойства или метода), но с иной семантикой, или наоборот — декларировать атрибуты с именами, занятыми системными объектами, классами или переменными, находящимися в том же локальном контексте и имеющими иной содержательный смысл.

Букву ω, а также буквы, образующие явные однобуквенные слова, условимся называть *автономными буквами*. В рамках рассматриваемого формализма именно автономные буквы играют роль системных и декларируемых имен, а именами атрибутов служат буквы, занимающие в словах вторую или более дальнюю позицию. С учетом этой аналогии упомянутое выше отсутствие омонимии можно сформулировать в виде (несколько более сильного) условия о том, что автономная буква может занимать в явном слове только первую позицию:

если $\alpha \in \mathbb{A}$, $X \in \mathbb{A}^+$, $Y \in \mathbb{A}^*$ и $X\alpha Y \in \mathbb{E}$,
то $\alpha \neq \omega$ и слово α не является явным. (**NoClash**)

3.7. Теорема. *Пусть выполнено* (NoClash). *Тогда автономная буква может занимать в объекте только первую позицию:*

если $\alpha \in \mathbb{A}$, $X \in \mathbb{A}^+$, $Y \in \mathbb{A}^*$ и $X\alpha Y \in \mathbb{O}$,
то $\alpha \neq \omega$ и слово α не является явным.

3.8. Предложение.

(1) *Всякий однобуквенный объект образован автономной буквой.*

(2) *Если выполнено условие* (Obj), *то автономные буквы — это в точности буквы, образующие однобуквенные объекты.*

(3) *Если выполнено условие* (PreObj), *то первая буква любого объекта является автономной.*

3.9. Пусть $X \in \mathbb{O}$ и $\alpha \in \mathbb{A}$. Будем называть X *источником атрибута* α, если α является добавленным атрибутом объекта X, т. е. если $X\alpha \in \mathbb{O} \cap \mathbb{E}$ и $X'\alpha \notin \mathbb{O}$ или, что то же самое, $X\alpha \in \mathbb{O}$ и $X'\alpha \notin \mathbb{O}$ (см. пп. 3.1 и 3.2).

Понятие источника служит аналогом понятия декларирующего класса для данного члена (свойства, метода и т. п.) — максимально абстрактного среди классов, обладающих этим членом, или, точнее, максимально далекого предка среди таких классов в какой-либо из цепей наследования.

Отметим, что наличие у какого-либо атрибута нескольких различных источников (несравнимых относительно наследования) вполне допустимо и не противоречит практике. Например, в языках .NET источниками атрибута `Length` служат классы `System.Array`, `System.String`, `System.IO.Stream` и некоторые другие классы.

3.10. Теорема.

(1) *Если* $X, Y \in \mathbb{O}$, $X \Rightarrow Y$ *и* $\|Y\|_1 = \varnothing$, *то объект* X *является источником всех своих атрибутов.* (*Например, это так в случае* $X \to \omega$.)

(2) *Если выполняется* (NoClash), *то автономная буква не может быть атрибутом какого-либо объекта и, в частности, не имеет источника.*

(3) *Если выполняются* (Obj) *и* (PreObj), *то всякая неавтономная буква явного алфавита имеет хотя бы один источник.*

(4) *Если выполняются* (Obj), (PreObj) *и* (NoClash), *то для любой неавтономной буквы* $\alpha \in \mathbb{A}_{\mathbb{E}}$ *следующие утверждения равносильны:*
   (a) *для любых явных* α*-свойств* $X\alpha$ *и* $Y\alpha$ *существует такое явное* α*-свойство* $Z\alpha$, *что* $X \overset{*}{\Rightarrow} Z$ *и* $Y \overset{*}{\Rightarrow} Z$;
   (b) *для любых* α*-свойств* $X\alpha$ *и* $Y\alpha$ *существует такое явное* α*-свойство* $Z\alpha$, *что* $X \overset{*}{\Rightarrow} Z$ *и* $Y \overset{*}{\Rightarrow} Z$;
   (c) α *обладает единственным источником.*

(5) *Любой атрибут любого объекта имеет источник. Более того, если* α — *атрибут объекта* X, *то* $X \overset{*}{\Rightarrow} Y$ *для некоторого источника* Y *атрибута* α.

(6) *Пусть выполнено* (Obj). *Буква* α *является атрибутом объекта* X *тогда и только тогда, когда* $X \overset{*}{\Rightarrow} Y$ *для некоторого источника* Y *атрибута* α.

4. Типы объектов

4.1. Будем говорить, что слово $D \in \mathbb{A}^+$ является *деталью* объекта X, если $XD \in \mathbb{O}$. Через $\|X\|$ обозначим множество всех деталей X и назовем $\|X\|$ *типом* объекта X. (Из условия 1.4(b) ясно, что $\|\omega\| = \varnothing$.)

Отметим, что множество $\mathbb{O}$ всех объектов может быть бесконечным и, более того, могут существовать объекты X с бесконечным типом $\|X\|$. Тем не менее множество $\{\|X\| : X \in \mathbb{O}\}$ всех типов объектов всегда конечно (см. теорему 6.2).

4.2. Будем говорить, что $\|X\|$ является *подтипом* $\|Y\|$, если $\|X\| \supseteq \|Y\|$. Отметим, что в приведенном определении фигурирует именно обратное включение множеств. Этим подчеркивается то обстоятельство, что тип $\|X\|$, будучи подтипом $\|Y\|$, является более конкретным и более богатым, а тип $\|Y\|$ — более абстрактным и более бедным.

4.3. Предложение. *Для любых объектов* X *и* Y *справедливы следующие утверждения:*

(1) *если* $\|X\| = \|Y\|$, *то* $\|XD\| = \|YD\|$ *для всех* $D \in \|X\|$;

(2) *если* $\|X\| \subseteq \|Y\|$, *то* $\|XD\| \subseteq \|YD\|$ *для всех* $D \in \|X\|$.

При дополнительном предположении (PreObj) *имеют место следующие утверждения:*

(3) $\|X\| = \|Y\|$ *тогда и только тогда, когда* $\|Y\|_1 = \|X\|_1$ *и* $\|X\alpha\| = \|Y\alpha\|$ *для всех* $\alpha \in \|X\|_1$*;*

(4) $\|X\| \subseteq \|Y\|$ *тогда и только тогда, когда* $\|X\|_1 \subseteq \|Y\|_1$ *и* $\|X\alpha\| \subseteq \|Y\alpha\|$ *для всех* $\alpha \in \|X\|_1$*.*

4.4. Предложение. *Если* $X, Y \in \mathbb{O}$*,* $X \Rightarrow Y$ *и* $XS \notin \mathbb{E}$ *для всех* $S \in \mathbb{A}^+$*, то* $\|X\| = \|Y\|$*.*

4.5. Если неформально интерпретировать отношение $X \stackrel{+}{\Rightarrow} Y$ как «X наследуется от Y» (или «X является частным случаем Y» или «X — это Y») и трактовать объекты $X\alpha$ как свойства X, то в случае $X \stackrel{+}{\Rightarrow} Y$ объект X должен в некотором смысле наследовать свойства Y и, возможно, делать их более конкретными и расширять их совокупность. Формально это требование сводится к следующему:

$$\text{если } X, Y \in \mathbb{O} \text{ и } X \stackrel{+}{\Rightarrow} Y, \text{ то } \|X\| \supseteq \|Y\|. \qquad (\textbf{CoInh+})$$

Приведенные ниже факты уточняют условие (CoInh+).

4.6. Введем следующее именованное условие:

$$\begin{array}{l}\text{Если } X, Y \in \mathbb{A}^+,\ \alpha \in \mathbb{A},\ X\alpha \in \mathbb{E},\ Y\alpha \in \mathbb{O} \text{ и } X \Rightarrow Y,\\ \text{то } X\alpha \in \mathbb{O} \text{ и } \|X\alpha\| \supseteq \|Y\alpha\|.\end{array} \qquad (\textbf{CoInh})$$

4.7. Теорема. *Пусть выполнено* (PreObj)*. Следующие утверждения равносильны:*

(1) *система удовлетворяет* (CoInh)*;*

(2) *если* $X, Y \in \mathbb{O}\backslash\{\omega\}$ *и* $X \Rightarrow Y$*, то* $\|X\| \supseteq \|Y\|$*;*

(3) *если* $X, Y \in \mathbb{O}$ *и* $X \stackrel{*}{\Rightarrow} Y$*, то* $\|X\| \supseteq \|Y\|$*.*

Следовательно, при выполнении (PreObj) условия (CoInh+) и (CoInh) равносильны.

4.8. Следствие. *Пусть выполнены* (PreObj) *и* (CoInh)*.*

(1) *Если* $Y \in \mathbb{O}$ *и* $X \stackrel{*}{\Rightarrow} Y$*, то* $\|X\|_1 \supseteq \|Y\|_1$ *и* $\|X\alpha\| \supseteq \|Y\alpha\|$ *для всех* $\alpha \in \|Y\|_1$*.*

(2) *Если* $X, Y, D \in \mathbb{A}^+$*,* $X \stackrel{*}{\Rightarrow} Y$ *и* $YD \in \mathbb{O}$*, то* $XD \in \mathbb{O}$ *и* $\|XD\| \supseteq \|YD\|$*.*

(3) *Если* $X, Y \in \mathbb{A}^+$*,* $X \stackrel{+}{\rightarrowtail} Y$ *и* $Y \in \mathbb{O}$*, то* $X \in \mathbb{O}$ *и* $\|X\| \supseteq \|Y\|$*.*

4.9. Объектные системы с типизацией значений атрибутов обычно удовлетворяют следующим (или аналогичным) требованиям. Предположим, что класс Y имеет декларированный атрибут α с типом значения τ. Если в этом случае объект x является экземпляром класса Y, то x имеет атрибут α, тип значения которого совпадает с τ или является более конкретным, чем τ. Аналогично, если класс X наследуется от класса Y, имеющего атрибут α типа τ, то X имеет унаследованный атрибут α, тип значения которого совпадает с τ или является более конкретным, чем τ. Если интерпретировать отношение $X \Rightarrow Y$ как «объект X является экземпляром класса Y» или «класс X непосредственно наследуется от класса Y» и трактовать отношение $X\alpha \to V$ как «V является явным значением свойства $X\alpha$» или «V является декларированным классом значения атрибута α в классе X», то приведенные выше требования можно формализовать в виде следующего именованного условия:

$$\begin{array}{l}\text{Если } X, Y, V \in \mathbb{A}^+,\ \alpha \in \mathbb{A},\ Y\alpha \in \mathbb{O},\ X\alpha \to V \text{ и } X \Rightarrow Y,\\ \text{то } V \in \mathbb{O} \text{ и } \|V\| \supseteq \|Y\alpha\|.\end{array} \qquad (\textbf{CoVal})$$

4.10. Теорема. *Совокупность условий* (PreObj) *и* (CoVal) *влечет* (CoInh).

4.11. Как показывает следующее утверждение, при выполнении (PreObj) условие (CoVal) влечет свой существенно более сильный вариант.

Предложение. *Пусть выполнены* (PreObj) *и* (CoVal).

Если $X, Y, V \in \mathbb{A}^+$, $\alpha \in \mathbb{A}$, $Y\alpha \in \mathbb{O}$, $X\alpha \Rightarrow V$, $X \overset{*}{\Rightarrow} Y$ *и* $X \neq Y$, *то* $V \in \mathbb{O}$ *и* $\|V\| \supseteq \|Y\alpha\|$.

4.12. *Графом типов объектов* будем называть граф, вершинами которого являются всевозможные объекты, а дугами служат пары $\langle X, Y\rangle$, где $X, Y \in \mathbb{O}$ и $\|X\| \subseteq \|Y\|$.

Для любой системы граф наследования $\langle \mathbb{O}, \Leftarrow\rangle$ является деревом (см. теорему 1.11), причем при выполнении (PreObj) и (CoInh) дерево наследования вкладывается в граф типов объектов (см. теорему 4.7). Тем не менее граф типов объектов не всегда представляет собой дерево. Более того, имеются примеры систем, удовлетворяющих всем рассматриваемым в данной работе именованным условиям, в которых существует четверка объектов A, B, C, D, образующих поддерево в графе наследования и «ромб» в графе типов:

$$D \Rightarrow B \Rightarrow A \Leftarrow C,$$

$$\|B\| \supset \|A\| \subset \|C\|, \quad \|B\| \subset \|D\| \supset \|C\|, \quad \|B\| \text{ и } \|C\| \text{ несравнимы.}$$

4.13. Концептуальная зависимость была введена в п. 1.13 как отношение на множестве всех слов. Поскольку слова, не являющиеся объектами, считаются «бессмысленными», было бы естественно описать зависимость между объектами с привлечением только объектов: если понятие X концептуально зависит от понятия Y, то должна существовать цепь понятий (а не произвольных слов), связывающая X с Y. Этот принцип обосновывается следующей теоремой (см. также следствие 4.15).

4.14. Теорема. *Пусть выполнены* (Obj), (PreObj) *и* (CoInh). *Объект* X *концептуально зависит от объекта* Y *тогда и только тогда, когда существуют* $X_1, \ldots, X_n \in \mathbb{O}$, $n \geqslant 1$, *такие, что* $X \rightarrowtail X_1 \rightarrowtail \cdots \rightarrowtail X_n = Y$.

4.15. Следствие. *Пусть выполнены* (Obj), (PreObj) *и* (CoInh). *Ограничение отношения* $\overset{+}{\rightarrowtail}$ *на* $\mathbb{O}$ *является наименьшим транзитивным отношением на* $\mathbb{O}$, *обладающим следующими тремя свойствами для всех* $X, Y \in \mathbb{O}$ *и* $D \in \mathbb{A}^+$:

(a) *если* $X \to Y$, *то* $X \overset{+}{\rightarrowtail} Y$;

(b) *если* $X \to Y$ *и* $D \in \|X\| \cap \|Y\|$, *то* $XD \overset{+}{\rightarrowtail} YD$;

(c) *если* $D \in \|X\|$, *то* $XD \overset{+}{\rightarrowtail} X$.

Приведенное выше утверждение показывает, что при выполнении (Obj), (PreObj) и (CoInh) концептуальная зависимость между объектами может быть описана в полном соответствии с исходным определением этого отношения на множестве всех слов. Единственное отличие состоит в том, что последнее описание не выходит за пределы множества объектов.

4.16. В теореме 4.14 и следствии 4.15 ни одно из условий (Obj), (PreObj) или (CoInh) нельзя опустить. Если отказаться хотя бы от одного из этих трех условий, возникают примеры систем, в которых отношение концептуальной зависимости между объектами не может быть охарактеризовано в рамках множества объектов: в этих системах имеются такие объекты X и Y, что X концептуально зависит от Y, но любая цепь, концептуально связывающая X с Y, непременно задействует бессмысленные слова.

4.17. Предположим, что для некоторого слова (понятия, объекта или класса) X потребовалось узнать «историю» его определения. История X должна содержать само слово X и историю его родителя X', а если X имеет вид $Y\alpha$, где α — перекрывающий атрибут объекта Y, то в историю X уместно включить историю перекрываемого слова $Y'\alpha$. Кроме того, отражая связь X с X', полезно разместить в истории X какой-либо признак того, имеет ли X какие-либо явные отличия от X', т. е. указать, есть ли у X детали, отсутствующие у X', и имеются ли объекты XD, значения которых перекрывают $X'D$.

С формальной точки зрения историю будем считать ориентированным графом, каждая вершина которого — объект, а каждая дуга, выходящая из $A \in \mathbb{O}$ и входящая в $B \in \mathbb{O}$, является тройкой вида $\langle A, r, B\rangle$, где $r \in \{0, 1, 2\}$. Всякий такой граф условимся называть *h-графом*. *Историей объекта* X назовем h-граф Γ, удовлетворяющий следующим условиям:

(a) X является вершиной Γ;
(b) к любой вершине Γ, отличной от X, ведет путь от X;
(c) Γ содержит дугу $\langle A, 0, B\rangle$ (соответственно, $\langle A, 1, B\rangle$) тогда и только тогда, когда $A \Rightarrow B$ и A не имеет явных деталей (соответственно, имеет хотя бы одну явную деталь);
(d) Γ содержит дугу $\langle A, 2, B\rangle$ тогда и только тогда, когда существуют объект C и его перекрывающий атрибут α такие, что $A = C\alpha$ и $B = C'\alpha$.

Теорема.
(1) *Любой объект имеет единственную историю.*
(2) *История любого объекта конечна.*
(3) *Если система концептуально непротиворечива, то история любого объекта ациклична.*

5. Эффективный анализ перезаписи

Оставшаяся часть статьи посвящена алгоритмической проверке различных свойств рассматриваемых перезаписывающих систем, и следующая теорема является основным шагом в этом направлении.

5.1. Теорема. *Для данной системы положим* $\mu := \max\{|E| : E \in \mathbb{E}\}$. *Слово* X *является бесконечно перезаписываемым тогда и только тогда, когда выполняется одно из следующих двух (взаимоисключающих) условий:*

(a) *существуют целые числа* $n \geqslant 0$ *и* $r > 0$ *такие, что* $X^{(n)} = X^{(n+r)}$;
(b) *существуют целые числа* $n \geqslant 0$ *и* $r > 0$ *такие, что*

$$\mu \leqslant \left|X^{(n)}\right| \leqslant \left|X^{(n+1)}\right|, \ldots, \left|X^{(n+r)}\right|,$$
$$X^{(n)} \neq X^{(n+r)}, \quad X^{(n)}\restriction_\mu = X^{(n+r)}\restriction_\mu.$$

В случае (a) *последовательность перезаписи слова* X *содержит цикл:*

$$X \overset{*}{\Rightarrow} \underbrace{X^{(n)} \Rightarrow \cdots \Rightarrow X^{(n+r-1)}}_{\text{период}} \Rightarrow X^{(n)} \Rightarrow \cdots.$$

В случае (b) *положим* $Y = X^{(n)}\restriction_\mu$ *и определим соответствующий суффикс* $S \in \mathbb{A}^*$, *для которого* $X^{(n)} = YS$. *Тогда существует слово* $R \in \mathbb{A}^+$ *такое, что* $Y^{(r)} = YR$, *и последовательность перезаписи* $\langle X^{(0)}, X^{(1)}, \ldots\rangle$ *содержит подпоследовательность, образованную словами* $X^{(n+mr)} = YR^mS$, $m \geqslant 0$, *каждое из которых начинает регулярный «период роста» длины* r:

$$\begin{aligned} X \overset{*}{\Rightarrow} X^{(n)} &= YS &&\Rightarrow Y^{(1)}S &&\Rightarrow \cdots \Rightarrow Y^{(r-1)}S \\ \Rightarrow X^{(n+r)} &= YRS &&\Rightarrow Y^{(1)}RS &&\Rightarrow \cdots \Rightarrow Y^{(r-1)}RS \\ \Rightarrow X^{(n+2r)} &= YR^2S &&\Rightarrow Y^{(1)}R^2S &&\Rightarrow \cdots \Rightarrow Y^{(r-1)}R^2S \\ \Rightarrow \cdots & \\ \Rightarrow X^{(n+mr)} &= YR^mS &&\Rightarrow Y^{(1)}R^mS &&\Rightarrow \cdots \Rightarrow Y^{(r-1)}R^mS \\ \Rightarrow \cdots. & \end{aligned}$$

В частности, $\{X^{(n)}, X^{(n+1)}, \dots\} = \{Y^{(j)}R^mS : 0 \leqslant j < r,\ m \geqslant 0\}$.

5.2. Пусть $\mathscr{P}$ — какое-либо множество «конструктивных объектов» (т. е. множество, чьи элементы могут использоваться как входные данные алгоритмов), и пусть $C(Y,p)$ — условие, которое может быть наложено на слова Y с дополнительными параметрами $p \in \mathscr{P}$. Формально можно считать, что C является подмножеством произведения $\mathbb{A}^+ \times \mathscr{P}$ и для всех $Y \in \mathbb{A}^+$ и $p \in \mathscr{P}$ выражение $C(Y,p)$ означает включение $\langle Y,p\rangle \in C$.

Для такого условия $C(Y,p)$ определим условие $C'(Y,R,S,p)$, где $Y,R \in \mathbb{A}^+$, $S \in \mathbb{A}^*$, $p \in \mathscr{P}$, следующим образом:

$C'(Y,R,S,p)$ тогда и только тогда,
когда существует $m \geqslant 1$ такое, что $C(YR^mS,p)$.

Будем говорить, что условие $C(Y,p)$ *циклически разрешимо*, если справедливы следующие два утверждения:

(a) существует алгоритм, проверяющий условие $C(Y,p)$ для $Y \in \mathbb{A}^+$, $p \in \mathscr{P}$;
(b) существует алгоритм, проверяющий условие $C'(Y,R,S,p)$ для $Y,R \in \mathbb{A}^+$, $S \in \mathbb{A}^*$, $p \in \mathscr{P}$.

Отметим, что условие (a) в общем случае не влечет (b). Этот факт можно вывести из существования рекурсивного множества $C \subseteq \mathbb{N}^2$, для которого множество $\{n \in \mathbb{N} : (\exists\, m \in \mathbb{N})\ \langle m,n\rangle \in C\}$ не является рекурсивным (см., например, [11, гл. 1, §6]).

5.3. Пусть $C(Y,p)$ — циклически разрешимое условие. Теорема 5.1 обосновывает следующий простой алгоритм, который по заданным системе, слову $X \in \mathbb{A}^+$ и параметру p проверяет существование слова $Y \in \mathbb{A}^+$, удовлетворяющего условиям $X \overset{*}{\Rightarrow} Y$ и $C(Y,p)$.

Алгоритм. Существует ли такое слово Y, что $X \overset{*}{\Rightarrow} Y$ и $C(Y,p)$?

- Если $C(X,p)$, возвращаем **Да**.
  Если слово X не является перезаписываемым, возвращаем **Нет**.
- В противном случае последовательно вычисляем результаты перезаписи $X^{(1)},\dots,X^{(i)},\dots$ и на каждом шаге $i \geqslant 1$ анализируем фрагменты $\langle X^{(n)}, X^{(n+1)}, \dots, X^{(n+r)}\rangle$, где $0 \leqslant n < n+r = i$, следующим образом:
  - Если $C\big(X^{(n+r)},p\big)$, возвращаем **Да**.
  - Если $X^{(n)} = X^{(n+r)}$, возвращаем **Нет**.
  - Если $\langle X^{(n)}, \dots, X^{(n+r)}\rangle$ удовлетворяет условию (b) теоремы 5.1,
    полагаем $Y := X^{(n)}{\restriction_\mu}$;
    определяем суффикс $S \in \mathbb{A}^*$, для которого $X^{(n)} = YS$;
    выбираем $R \in \mathbb{A}^+$ так, что $Y^{(r)} = YR$
    (такое слово R существует по теореме 5.1);
    если $C'(Y,R,S,p)$, возвращаем **Да**; иначе возвращаем **Нет**.
  - Если $X^{(n+r)}$ не является перезаписываемым, возвращаем **Нет**.
    В противном случае переходим к следующему шагу, $i+1$.

5.4. Как легко видеть, для данной системы $\mathscr{S}$ условие

$$C(Y, \mathscr{S}) = \text{«}Y \text{ не является перезаписываемым в } \mathscr{S}\text{»}$$

циклически разрешимо. Поэтому алгоритм 5.3, специализированный для этого условия, проверяет конечную перезаписываемость данного слова в данной системе. Ниже приведена упрощенная версия этой процедуры.

Алгоритм. Является ли слово X конечно перезаписываемым?

- Начинаем вычисление последовательности перезаписи $X^{(0)}$, $X^{(1)}$, ...
- Если возникает неперезаписываемое слово $X^{(i)}$, возвращаем **Да**.
- В противном случае согласно теореме 5.1 в последовательности встретится фрагмент $\langle X^{(n)}, X^{(n+1)}, \ldots, X^{(n+r)}\rangle$, удовлетворяющий условию (a) или условию (b) теоремы 5.1. В этом случае возвращаем **Нет**.

Поскольку множество $\mathbb{E}$ явных слов конечно, из теоремы 2.12 следует, что *условие* (Fin) *эффективно проверяемо*: достаточно ко всем явным словам применить алгоритм 5.4.

5.5. Специализация алгоритма 5.3 для условия

$$C(Y) = \text{«}Y = \omega\text{»}$$

проверяет, *является ли данное слово объектом*. (Эту же проверку можно осуществить посредством незначительной модификации алгоритма 5.4.) Теперь ясно, что *условия* (Obj) *и* (PreObj) *эффективно проверяемы*.

5.6. Отметим, что при выполнении условия (Fin) включение $X \in \mathbb{O}$ проверяется тривиально: достаточно проверить, оканчивается ли словом ω (конечная) последовательность перезаписи X. Кроме того, если выполнено (Obj), можно остановить вычисление последовательности перезаписи X, если на некотором шаге $n \geqslant 0$ возникнет явное слово $X^{(n)} \in \mathbb{E}$. Более того, если выполнены и (Obj), и (PreObj), вычисление можно завершить, если очередной результат перезаписи $X^{(n)}$ окажется префиксом какого-либо явного слова (см. предложение 2.16).

5.7. Как легко видеть, условие

$$C(Y, X) = \text{«}Y \sqsupseteq X\text{»}$$

циклически разрешимо. Поэтому соответствующим образом специализированная версия алгоритма 5.3 проверяет, *является ли данное слово X рекуррентным*. По теореме 2.9 заключаем, что (Rec) *эффективно проверяемо*: чтобы проверить отсутствие рекуррентных слов, достаточно применить этот алгоритм ко всем словам над $\mathbb{A}_{\mathbb{E}}$ длины не более μ. (Тем не менее получающаяся проверка имеет экспоненциальное время работы; см. замечание 2.10.)

Другой подход к проверке (Rec) может быть основан на теореме 2.13, согласно которой (Rec) влечет (Fin). Сначала проверим (Fin). Если это условие не выполняется, то (Rec) также не выполняется. Если же (Fin) выполняется, то условие (Rec) можно проверить, обработав все слова X над $\mathbb{A}_{\mathbb{E}}$ длины $|X| \leqslant \mu$ и вернув **Нет**, если среди них встретится такое слово X, что $X \sqsubseteq X^{(n)}$ для некоторого $n \geqslant 1$.

5.8. По теореме 2.6 условие (Con) можно эффективно проверить с помощью построения концептуальной схемы и проверки ацикличности этой схемы в ходе ее построения. Описанный ниже алгоритм проверяет, является ли данная система концептуально непротиворечивой. Если это так, алгоритм возвращает концептуальную схему системы, а в противном случае возвращает пример цикла в концептуальной схеме. В алгоритме используются переменный ориентированный граф $\Gamma = \langle \Gamma_{\mathrm{V}}, \Gamma_{\mathrm{A}} \rangle$ (с вершинами Γ_{V} и дугами Γ_{A}) и переменная $\mathscr{A}$, значениями которой являются конечные подмножества декартова квадрата $\mathbb{A}^+ \times \mathbb{A}^+$.

Алгоритм. Концептуально непротиворечива ли данная система?
(1) Полагаем $\Gamma_{\mathrm{V}} := \mathbb{E}$, $\Gamma_{\mathrm{A}} := \varnothing$.
(2) Полагаем $\mathscr{A} := \{\langle X, Y\rangle : X$ — сток графа Γ и $X \rightarrowtail Y\}$.
(3) Если $\mathscr{A} = \varnothing$, констатируем **концептуальную непротиворечивость** и возвращаем Γ в качестве **концептуальной схемы**.
(4) В противном случае для каждой пары $\langle X, Y\rangle \in \mathscr{A}$ делаем следующее:
если $X = Y$ или Γ содержит путь из Y в X,
утверждаем, что система **концептуально противоречива**,
и возвращаем **цикл**: $X \rightarrowtail X$ или $Y \rightarrowtail \cdots \rightarrowtail X \rightarrowtail Y$;
в противном случае полагаем $\Gamma_{\mathrm{V}} := \Gamma_{\mathrm{V}} \cup \{Y\}$, $\Gamma_{\mathrm{A}} := \Gamma_{\mathrm{A}} \cup \{\langle X, Y\rangle\}$.
(5) Переходим к (2).

5.9. Если алгоритм 5.8 применить к концептуально противоречивой системе, он вернет пример цикла

$$X_0 \rightarrowtail X_1 \rightarrowtail \cdots \rightarrowtail X_m = X_0$$

в концептуальной схеме, но не гарантирует, что все слова X_i в этом цикле являются объектами (даже в случае, когда X_0 — объект). С другой стороны, из теоремы 4.14 следует, что при выполнении условий (Obj), (PreObj) и (CoInh) любой путь

$$X_0 \rightarrowtail X_1 \rightarrowtail \cdots \rightarrowtail X_m$$

между объектами X_0 и X_m можно преобразовать в путь из *объектов*

$$Y_0 \rightarrowtail Y_1 \rightarrowtail \cdots \rightarrowtail Y_n, \quad \text{где } Y_0 = X_0 \text{ и } Y_n = X_m.$$

Отметим, что такое преобразование может быть выполнено *эффективно*.

5.10. Небольшим изменением алгоритма 5.8 можно получить процедуру построения схемы слабой перезаписи вместо концептуальной схемы. Приведенный ниже алгоритм проверяет для произвольной системы, существуют ли слабо рекуррентные слова, и либо возвращает пример такого слова, либо строит схему слабой перезаписи системы.

Алгоритм. Существует ли слабо рекуррентное слово?
(1) Полагаем $\Gamma_{\mathrm{V}} := \mathbb{E}$, $\Gamma_{\mathrm{A}} := \varnothing$.
(2) Полагаем $\mathscr{A} := \{\langle X, Y\rangle : X$ — сток графа Γ и $X \Rightarrow_{\mathrm{w}} Y\}$.
(3) Если $\mathscr{A} = \varnothing$, утверждаем, что **слабо рекуррентных слов нет**, и возвращаем Γ в качестве **схемы слабой перезаписи**.
(4) В противном случае для каждой пары $\langle X, Y\rangle \in \mathscr{A}$ делаем следующее:
если $X \sqsubseteq Y$ или Γ содержит путь из префикса $Y_0 \sqsubseteq Y$ в X,
возвращаем **слабо рекуррентное слово**:
$X \Rightarrow_{\mathrm{w}} Y \sqsupseteq X$ или $Y_0 \Rightarrow_{\mathrm{w}} \cdots \Rightarrow_{\mathrm{w}} X \Rightarrow_{\mathrm{w}} Y \sqsupseteq Y_0$;
в противном случае полагаем $\Gamma_{\mathrm{V}} := \Gamma_{\mathrm{V}} \cup \{Y\}$, $\Gamma_{\mathrm{A}} := \Gamma_{\mathrm{A}} \cup \{\langle X, Y\rangle\}$.
(5) Переходим к (2).

5.11. Согласно теореме 2.6 схема слабой перезаписи концептуально противоречивой системы содержит путь вида $X_0 \overset{+}{\Rightarrow}_{\mathrm{w}} X_n \sqsupseteq X_0$, где $X_0 \in \mathbb{E}$. Будучи примененным к противоречивой системе, алгоритм 5.10 возвращает пример пути $Y_0 \overset{+}{\Rightarrow}_{\mathrm{w}} Y_n \sqsupseteq Y_0$, но начальное слово Y_0 не обязано быть явным. В этой связи стоит отметить, что всякий путь вида $Y_0 \overset{+}{\Rightarrow}_{\mathrm{w}} Y_n \sqsupseteq Y_0$ может быть *эффективно* преобразован в путь вида $X_0 \overset{+}{\Rightarrow}_{\mathrm{w}} X_n \sqsupseteq X_0$ (той же длины), где $X_0 \in \mathbb{E}$.

5.12. Наиболее убедительным свидетельством концептуальной противоречивости представляется пример цикла $X_0 \rightarrowtail \cdots \rightarrowtail X_n = X_0$, состоящего из *объектов* и начинающегося с *явного* слова X_0. Отметим, что если противоречивая система удовлетворяет условиям (Obj), (PreObj) и (CoInh), то цикл такого вида может быть найден *эффективно*.

6. Эффективный анализ типов объектов

6.1. Объект C будем называть *характером*, если либо $C = \omega$, либо $C \sqsubset E$ для некоторого слова $E \in \mathbb{E}$. Множество всех характеров обозначим через $\mathbb{C}$. Ясно, что это множество конечно.

Для данного объекта X рассмотрим последовательность перезаписи

$$X^{(0)} \Rightarrow \cdots \Rightarrow X^{(n)} = \omega$$

и обозначим через $\mathrm{ch}(X)$ первый характер, встречающийся в этой последовательности. Назовем $\mathrm{ch}(X)$ *характером* объекта X.

Поскольку $\mathrm{ch}(C) = C$ для всех $C \in \mathbb{C}$, справедливо равенство

$$\mathbb{C} = \{\mathrm{ch}(X) : X \in \mathbb{O}\}.$$

6.2. Теорема. *Для всех $X \in \mathbb{O}$ имеет место равенство $\|X\| = \|\mathrm{ch}(X)\|$. В частности,*

$$\{\|X\| : X \in \mathbb{O}\} = \{\|C\| : C \in \mathbb{C}\},$$

и множество $\{\|X\| : X \in \mathbb{O}\}$ конечно.

6.3. Под *задачей сравнения типов* понимается поиск алгоритма, обеспечивающего сравнение типов объектов, т. е. определяющего, какие из отношений

$$\|X\| = \|Y\|, \quad \|X\| \subseteq \|Y\|, \quad \|X\| \supseteq \|Y\|$$

выполняются для данных объектов X и Y. Ясно, что по данному объекту можно эффективно определить его характер. Следовательно, согласно теореме 6.2 задача сравнения типов сводится к эффективному сравнению типов характеров.

6.4. Для произвольной функции $\tau : \mathbb{C} \to \mathbb{N}$ определим следующее отношение эквивалентности на $\mathbb{C}$:

$X \underset{\tau}{\sim} Y$ тогда и только тогда,
когда $\|X\|_1 = \|Y\|_1$ и $\tau\big(\mathrm{ch}(X\alpha)\big) = \tau\big(\mathrm{ch}(Y\alpha)\big)$ для всех $\alpha \in \|X\|_1$.

Функцию $\tau : \mathbb{C} \to \mathbb{N}$ назовем *типизацией*, если для любых $X, Y \in \mathbb{C}$

$$\text{из } \tau(X) = \tau(Y) \text{ следует } X \underset{\tau}{\sim} Y.$$

Простейшим примером типизации служит произвольная инъекция $\tau : \mathbb{C} \to \mathbb{N}$. Типизацию τ назовем *минимальной*, если ее образ $\mathrm{im}\,\tau := \tau[\mathbb{C}]$ имеет наименьшее число элементов $|\mathrm{im}\,\tau|$ среди всевозможных типизаций:

$$|\mathrm{im}\,\tau| \leqslant |\mathrm{im}\,\sigma| \text{ для любой типизации } \sigma : \mathbb{C} \to \mathbb{N}.$$

Функцию $\tau : \mathbb{C} \to \mathbb{N}$ назовем *точной типизацией*, если для любых $X, Y \in \mathbb{C}$

$$\tau(X) = \tau(Y) \text{ равносильно } \|X\| = \|Y\|.$$

Утверждение 6.5(2) приведенной ниже теоремы оправдывает употребление термина «типизация» в последнем определении.

6.5. Теорема. *Пусть* $\tau : \mathbb{C} \to \mathbb{N}$.
(1) *Если для любых* $X, Y \in \mathbb{C}$ *из* $\|X\| = \|Y\|$ *следует* $\tau(X) = \tau(Y)$, *то для любых* $X, Y \in \mathbb{C}$ *из* $\|X\| = \|Y\|$ *следует* $X \underset{\tau}{\sim} Y$.
(2) *Всякая точная типизация является типизацией.*
(3) *Пусть выполнено* (PreObj). *Если* τ *является типизацией, то для всех* $X, Y \in \mathbb{C}$ *из* $\tau(X) = \tau(Y)$ *следует* $\|X\| = \|Y\|$.
(4) *Утверждение, обратное к* (3), *не имеет места: существуют система, удовлетворяющая условию* (PreObj), *и функция* $\tau : \mathbb{C} \to \mathbb{N}$ *такие, что для всех* $X, Y \in \mathbb{C}$ *из* $\tau(X) = \tau(Y)$ *следует* $\|X\| = \|Y\|$, *но функция* τ *не является типизацией.*
(5) *Утверждение* (3) *не сохраняет силу без требования* (PreObj).

6.6. Теорема. *Пусть выполнено* (PreObj). *Функция* $\tau : \mathbb{C} \to \mathbb{N}$ *является точной типизацией тогда и только тогда, когда* τ *— минимальная типизация.*

6.7. Следующий алгоритм использует две переменные функции $\tau, \sigma : \mathbb{C} \to \mathbb{N}$ (каждая из которых может быть закодирована в виде массива натуральных чисел, индексированных характерами).

Алгоритм. Вычисление точной типизации.
(1) Определяем τ, полагая $\tau(X) := 1$ для всех $X \in \mathbb{C}$.
(2) Если для всех $X, Y \in \mathbb{C}$ из $\tau(X) = \tau(Y)$ следует $X \underset{\tau}{\sim} Y$, **возвращаем** τ.
(3) Присваиваем σ копию τ.
(4) Для каждого $k \in \big\{\tau(X) : X \in \mathbb{C}$ и $(\exists Y \in \mathbb{C})\big(X \underset{\tau}{\not\sim} Y$ и $\tau(X) = \tau(Y)\big)\big\}$ выполняем следующее:
произвольно нумеруем множество $\{X \in \mathbb{C} : \tau(X) = k\}$,
превращая его в последовательность $\langle X_1, \dots, X_n\rangle$, $n \geqslant 2$,
и для каждого $i = 2, \dots, n$ выполняем следующее:
если $X_i \underset{\tau}{\sim} X_j$ для некоторого $1 \leqslant j < i$, полагаем $\sigma(X_i) := \sigma(X_j)$,
а в противном случае полагаем $\sigma(X_i) := \max\{\sigma(X) : X \in \mathbb{C}\} + 1$.
(5) Полагаем $\tau := \sigma$ и переходим к (2).

6.8. Теорема. *Алгоритм* 6.7 *завершается для любой системы, и если система удовлетворяет условию* (PreObj), *то возвращаемая функция* $\tau : \mathbb{C} \to \mathbb{N}$ *является точной типизацией:* $\tau(X) = \tau(Y)$ *тогда и только тогда, когда* $\|X\| = \|Y\|$.

Алгоритм 6.7 аналогичен алгоритму В. Г. Визинга, осуществляющему разбиение множества вершин графа на классы подобных вершин (см. [12]). Автор благодарен С. В. Августиновичу за обнаружение этой аналогии.

6.9. Для произвольного бинарного отношения $\lhd$ на $\mathbb{C}$ (т. е. подмножества $\lhd \subseteq \mathbb{C}^2$) определим бинарное отношение $\widetilde{\lhd}$ на $\mathbb{C}$ следующим образом:

$X \mathrel{\widetilde{\lhd}} Y$ тогда и только тогда,
когда $\|X\|_1 \subseteq \|Y\|_1$ и $\mathrm{ch}(X\alpha) \lhd \mathrm{ch}(Y\alpha)$ для всех $\alpha \in \|X\|_1$.

Отношение $\lhd \subseteq \mathbb{C}^2$ назовем *подтипизацией*, если $\lhd \subseteq \widetilde{\lhd}$, т. е. для всех $X, Y \in \mathbb{C}$

$$\text{из } X \lhd Y \text{ следует } X \mathrel{\widetilde{\lhd}} Y.$$

Простейшим примером подтипизации служит отношение равенства на $\mathbb{C}$. *Точной подтипизацией* назовем отношение $\{\langle X, Y\rangle \in \mathbb{C}^2 : \|X\| \subseteq \|Y\|\}$, т. е. такое отношение $\lhd \subseteq \mathbb{C}^2$, что для любых $X, Y \in \mathbb{C}$

$$X \lhd Y \text{ равносильно } \|X\| \subseteq \|Y\|.$$

Утверждение 6.10(2) приведенной ниже теоремы оправдывает употребление термина «подтипизация» в последнем определении.

6.10. Теорема. *Пусть* $\lhd$ *— произвольное бинарное отношение на* $\mathbb{C}$.
(1) *Если* $\lhd$ *включает точную подтипизацию, то и* $\widetilde{\lhd}$ *включает точную подтипизацию. Иными словами, если для любых* $X, Y \in \mathbb{C}$ *из* $\|X\| \subseteq \|Y\|$ *следует* $X \lhd Y$*, то для любых* $X, Y \in \mathbb{C}$ *из* $\|X\| \subseteq \|Y\|$ *следует* $X \mathrel{\widetilde{\lhd}} Y$.
(2) *Точная подтипизация является подтипизацией.*
(3) *Пусть выполнено* (PreObj). *Тогда всякая подтипизация включается в точную подтипизацию: если* $\lhd$ *— подтипизация,* $X, Y \in \mathbb{C}$ *и* $X \lhd Y$*, то* $\|X\| \subseteq \|Y\|$.

6.11. Описанный ниже алгоритм использует два переменных множества $\lhd, \lhd_0 \subseteq \mathbb{C}^2$.

Алгоритм. Вычисление точной подтипизации.
(1) Полагаем $\lhd := \mathbb{C}^2$.
(2) Полагаем $\lhd_0 := \lhd \cap \widetilde{\lhd}$.
(3) Если $\lhd_0 = \lhd$, **возвращаем** $\lhd$.
В противном случае полагаем $\lhd := \lhd_0$ и переходим к (2).

6.12. Теорема. *Для любой системы алгоритм* 6.11 *завершается, и если система удовлетворяет условию* (PreObj)*, то возвращаемое отношение* $\lhd$ *является точной подтипизацией: для всех* $X, Y \in \mathbb{C}$ *соотношение* $X \lhd Y$ *равносильно включению* $\|X\| \subseteq \|Y\|$.

6.13. Таким образом, при выполнении (PreObj) соотношение $\|X\| \subseteq \|Y\|$ для $X, Y \in \mathbb{O}$ поддается эффективной проверке. (Как легко видеть, отсюда следует, что условия (CoInh) *и* (CoVal) *эффективно проверяемы*.) Тем не менее одного только утверждения «$\|X\| \not\subseteq \|Y\|$» не всегда достаточно, и для более тонкого анализа может потребоваться конкретная причина нарушения включения $\|X\| \subseteq \|Y\|$. В громоздкой системе поиск конкретной детали $D \in \|X\| \backslash \|Y\|$ может оказаться нетривиальным, и соответствующий алгоритм был бы полезным инструментом диагностики. Напомним, что согласно теореме 6.2 решение этой задачи достаточно автоматизировать для характеров.

6.14. *Диагностикой подтипизации* назовем функцию $\Delta : \mathscr{D} \to \mathbb{A}^+$, определенную на множестве

$$\mathscr{D} = \{\langle X, Y\rangle \in \mathbb{C}^2 : \|X\| \not\subseteq \|Y\|\}$$

и сопоставляющую каждой паре $\langle X, Y\rangle \in \mathscr{D}$ какую-нибудь деталь

$$\Delta(X, Y) \in \|X\| \backslash \|Y\|.$$

6.15. Следующий алгоритм использует переменное множество $\mathscr{D} \subseteq \mathbb{C}^2$ и переменную функцию $\Delta : \mathscr{D} \to \mathbb{A}^+$.

Алгоритм. Вычисление диагностики подтипизации.
(1) Полагаем $\mathscr{D} := \varnothing$.
(2) Для всех попарно различных $X, Y \in \mathbb{C}$ выполняем следующее:
если существует $\alpha \in \|X\|_1 \backslash \|Y\|_1$,
то добавляем пару $\langle X, Y\rangle$ к $\mathscr{D}$ и присваиваем $\Delta(X, Y) := \alpha$.
(3) Для всех попарно различных $X, Y \in \mathbb{C}$ таких, что $\langle X, Y\rangle \notin \mathscr{D}$, выполняем следующее:
если имеется атрибут $\alpha \in \|X\|_1$, для которого $\big\langle \mathrm{ch}(X\alpha), \mathrm{ch}(Y\alpha)\big\rangle \in \mathscr{D}$,
то добавляем пару $\langle X, Y\rangle$ к $\mathscr{D}$
и присваиваем $\Delta(X, Y) := \alpha\, \Delta\big(\mathrm{ch}(X\alpha), \mathrm{ch}(Y\alpha)\big)$.
(4) Если на шаге (3) не было присваиваний, то **возвращаем** Δ.
В противном случае переходим к (3).

6.16. Теорема. *Для любой системы алгоритм* 6.15 *завершается, и если система удовлетворяет условию* (PreObj)*, то возвращаемая функция* $\Delta : \mathscr{D} \to \mathbb{A}^+$ *является диагностикой подтипизации:*

$$\mathscr{D} = \{\langle X, Y\rangle \in \mathbb{C}^2 : \|X\| \not\subseteq \|Y\|\},$$
$$\Delta(X, Y) \in \|X\|\backslash\|Y\| \text{ для всех } \langle X, Y\rangle \in \mathscr{D}.$$

6.17. Диагностику подтипизации $\Delta : \mathscr{D} \to \mathbb{A}^+$ назовем *лаконичной*, если для всех $\langle X, Y\rangle \in \mathscr{D}$

$$|\Delta(X, Y)| = \min\{|D| : D \in \|X\|\backslash\|Y\|\},$$

т. е. деталь $\Delta(X, Y) \in \|X\|\backslash\|Y\|$ имеет наименьшую длину среди всевозможных слов $D \in \|X\|\backslash\|Y\|$. В этом смысле $\Delta(X, Y)$ дает самый короткий ответ на вопрос о том, почему не выполняется включение $\|X\| \subseteq \|Y\|$.

Диагностика подтипизации, возвращаемая алгоритмом 6.15, не всегда лаконична. Более того, имеются примеры систем, удовлетворяющих всем рассматриваемым в данной работе именованным условиям, для которых этот алгоритм дает верные, но не самые короткие ответы.

6.18. Описанный ниже алгоритм является модификацией алгоритма 6.15. Основное отличие заключается в использовании вспомогательной целочисленной переменной n, значение которой увеличивается на единицу после каждого прохождения этапа (3).

Алгоритм. Вычисление лаконичной диагностики подтипизации.
(1) Полагаем $\mathscr{D} := \varnothing$ и $n := 1$.
(2) Для всех попарно различных $X, Y \in \mathbb{C}$ выполняем следующее:
    если существует $\alpha \in \|X\|_1\backslash\|Y\|_1$,
    то добавляем пару $\langle X, Y\rangle$ к $\mathscr{D}$ и присваиваем $\Delta(X, Y) := \alpha$.
(3) Для всех попарно различных $X, Y \in \mathbb{C}$ таких, что $\langle X, Y\rangle \notin \mathscr{D}$, выполняем следующее:
    если существует атрибут $\alpha \in \|X\|_1$,
    для которого $\big\langle \mathrm{ch}(X\alpha), \mathrm{ch}(Y\alpha)\big\rangle \in \mathscr{D}$ и $\big|\Delta\big(\mathrm{ch}(X\alpha), \mathrm{ch}(Y\alpha)\big)\big| = n$,
    то добавляем пару $\langle X, Y\rangle$ к $\mathscr{D}$
    и присваиваем $\Delta(X, Y) := \alpha\,\Delta\big(\mathrm{ch}(X\alpha), \mathrm{ch}(Y\alpha)\big)$.
(4) Если на шаге (3) не было присваиваний, то **возвращаем** Δ.
    В противном случае полагаем $n := n + 1$ и переходим к (3).

6.19. Теорема. *Для любой системы алгоритм* 6.18 *завершается, и если система удовлетворяет* (PreObj)*, то возвращаемая функция* $\Delta : \mathscr{D} \to \mathbb{A}^+$ *является лаконичной диагностикой подтипизации:*

$$\mathscr{D} = \{\langle X, Y\rangle \in \mathbb{C}^2 : \|X\| \not\subseteq \|Y\|\},$$
$$\Delta(X, Y) \in \|X\|\backslash\|Y\|,$$
$$|\Delta(X, Y)| = \min\{|D| : D \in \|X\|\backslash\|Y\|\}$$

для всех $\langle X, Y\rangle \in \mathscr{D}$.

Благодарность. Автор признателен Сергею Владимировичу Августиновичу за плодотворные обсуждения.

ЛИТЕРАТУРА

1. *Büchi J. R.* Regular canonical systems // Arch. Math. Logik Grundlag. 1964. V. 6. P. 91–111.
2. *Caucal D.* On the regular structure of prefix rewriting // Theoret. Comput. Sci. 1992. V. 106, N 1. P. 61–86.
3. *Book R. V., Otto F.* String-rewriting systems. New York: Springer, 1993.
4. *Frazier M., Page C. D.* Prefix grammars: An alternative characterization of the regular languages // Inform. Process. Lett. 1994. V. 51, N 2. P. 67–71.
5. *Cardelli L., Wegner P.* On understanding types, data abstraction, and polymorphism // ACM Comput. Surv. 1985. V. 17, N 4. P. 471–523.
6. *Аткинсон М., Бансилон Ф., ДеВитт Д., Диттрих К., Майер Д., Здоник С.* Манифест систем объектно-ориентированных баз данных // Системы управления базами данных. 1995. № 4.
7. *Dittrich K. R.* Object-oriented data model concepts // Advances in Object-Oriented Database Systems. Berlin: Springer, 1994. P. 29–45.
8. *Харари Ф.* Теория графов. Москва: Мир, 1973.
9. *Gutman A. E.* Object-oriented data as prefix rewriting systems // Vladikavkaz. Mat. Zh. 2015. V. 17, N 3. P. 23–35.
10. *Саломаа А.* Жемчужины теории формальных языков. Москва: Мир, 1986.
11. *Барвайс Дж.* Справочная книга по математической логике. Часть III: Теория рекурсии. Москва: Наука, 1982.
12. *Визинг В. Г.* Дистрибутивная раскраска вершин графа // Дискрет. анализ и исслед. операций. 1995. Т. 2, № 4. С. 3–12.

Поступила в редакцию 15 января 2026 г.
После доработки 15 января 2026 г.
Принята к публикации 7 апреля 2026 г.

Гутман Александр Ефимович (ORCID 0000-0003-2030-7459)
Институт математики им. С. Л. Соболева СО РАН,
пр. Академика Коптюга, 4, Новосибирск 630090
gutman@math.nsc.ru