tanulmányok 134/1982

MTA Számitástechnikai és Automatizálási Kutató Intézet Budapest

MAGYAR TUDOMÁNYOS AKADÉMIA
SZÁMITÁSTECHNIKAI ÉS AUTOMATIZÁLÁSI KUTATÓ INTÉZETE

ОСНОВЫ ЯЗЫКА ЗАПРОСА БАЗЫ ДАННЫХ

SDLA/SET

Az SDLA/SET adatbázis lekérdező nyelv alapjai

Irta:

KNUTH ELŐD
RONYAI LAJOS

Tanulmányok 134/1982

A kiadásért felelős:

DR VÁMOS TIBOR

ISBN 963 311 142 0
ISSN 0324-2951

1. ОБ'ЕКТНЫЙ ЯЗЫК SDLA

Ниже приводятся некоторые основные свойства системы SDLA. Они понадобятся в дальнейшем. Другие частные вопросы могут быть рассмотрены в работах [4] и [5].

1.1. Описание на двух уровнях

Описание какой-либо системы при помощи SDLA осуществляется главным образом по дедуктивному методу: описание расчленяется на два уровня. Переход от обобщенного, абстрактного к специальному выражается отношением друг к другу этих двух уровней.

Базовая схема системной модели задается на т.н. уровне определения. На этом уровне определяются характеризующие систему основные понятия, представляющие собой в интерпретации SDLA и тип отношения. На этом же уровне может быть описано отношение между понятиями /суб- и суперординация, целостность, принуждения/. Язык пользователя модели SDLA задается созданием форм, относящихся к понятиям. Создаваемое таким образом метаописание контролируется системой с различных аспектов. Относительно критериев контроля см. [5], [6].

Второй, конкретный уровень описания осуществляется посредством т.н. описательного диалога. С помощью сформированного языка могут создаваться конкретные экземпляры /инстанции/ определенных ранее понятий. Тем самым как бы заполняем заданные, но еще пустые отношения, на уровне определения. Созданные таким образом инстанции хранятся в базе данных, через которую они доступны для целей анализа системы, контроля и сгенерирования документов.

1.2. Определение языка пользователя

Как об этом в предыдущем пункте уже было сказано, язык пользователя формируется на первом уровне, в фазе определений. Опре-

деление понятия, по-существу, осуществляется следующим образом:

(1) concept имя-1 is имя-2 /sel-1:attr-1,..., sel-n:attr-n/;

где

- имя-1: имя задаваемого понятия;
- имя-2: имя непосредственно вышестоящего понятия;
- sel-i: i-тое имя селектора;
- attr-i: i-тое имя атрибута, /являющегося также понятием/.

Здесь в отдельных случаях имена селекторов могут отсутствовать; их роль в таком случае берут на себя имена соответствующих атрибутов.

Одним из основных принципов реляционного взгляда /см. подробнее [4]/ является то, что понятие может рассматриваться с позиции любого из его атрибутов. Поэтому любое из предложений /высказываний, относящихся к инстанциям/ создаваемого языка произносится с определенной точки зрения. Каждое понятие может рассматриваться с некоторой абсолютной точки зрения, с которой оно является таким, как в действительности. Взгляды с позиций отдельных атрибутов называются реляционными. Их описание осуществляется следующим образом: под определением (1) следует записать предложения вида

(2) from sel - i: < имена остальных селекторов, разделенные включаемым текстом > ;

С точки зрения любого селектора может быть записано произвольное число форм и после двоеточия необязательно должны следовать все остальные имена селекторов /неполная форма/. С предложениями вида (2) естественным образом ассоциируются два понятия. Понятие, создаваемое предложением (1) представляет собой тип формы, а i-тый атрибут является точкой зрения формы.

Описание абсолютных форм имеет вид:

(3) **form absolute**: < селекторы, разделенные включаемым текстом >;

Здесь точка зрения формы является абсолютной /значит не понятие/, а тип ее - понятие (1).

Заданием форм вида (2) и (3) описывается "включаемый текст", как форма описания, применяемая с актуальной точки зрения, представляющая собой в заданной среде тип формы, как понятие.

С помощью форм (2) и (3), по-существу, задается язык пользователя **SDLA**. Во всех фазах работы с моделью **SDLA** /описание данных, анализ системы, запрос/ используется этот язык, дополненный некоторыми, независящими от конкретной модели синтактическими элементами. Таковыми являются, в частности, синтагмы **set**, **equ**, и т.д. Нам кажется, что используемость системы, удобство пользователя в большой мере возрастает вследствие того, что пользователь применяет во всех фазах, по сути дела, тот же самый язык. Само собой разумеется, что при этом семантика изменяется в соответствии с состоянием **SDLA**.

Такая возможность создается засчет особенностей понятийного мышления. Широкий класс языков, включающий как естественные языки, так и, например, языки математики, характеризуется своеобразной функциальной многогранностью. Для прояснения приведем пример. Рассмотрим следующее предложение:

(4) **Prolepsis** представляет собой такой языковый оборот, в котором кому- или чему-нибудь присваиваются такие свойства, которыми он /она, оно/ будет владеть только позже.

В функциональном отношении такое предложение может рассматриваться как определение, т.е. как описание понятия. В нем за-

даются характеристики "prolepsis"-а, как языкового оборота. Это представляет собой обыкновенную форму определения, заданного с помощью "genus proximum" и "differentia specifica". Однако вышеприведенное предложение может рассматриваться с функциональной точки зрения как и алгоритм. Допустим, имеется следующее высказывание:

(5) Два брата и их жертва проскакали мимо красивой Флоренции.

С помощью предложения (4), как описания алгоритма, можно решить, что утверждение (5) содержит ли prolepsis или нет. /В данном случае ответ: да, поскольку мертвые не скачут на лошади./ Разумеется, что в случае языков, характеризующихся большей силой выражения, являющихся шире чем (4), можно найти и менее тривиальные примеры. Так, например, язык первого порядка частных упорядочений является весьма простым. Он включает в себя единственный реляционный знак с двумя переменными: "≤", и для него выполняются аксиомы рефлексивности, антисимметрии и транзитивности. Легко могут быть заданы те частно-упорядоченные множества P, в которых любой из трех элементов имеет минимальное верхнее предельное значение. P является таковым, если $P = \varphi$, где

$$(5)\quad \varphi = \forall x \forall y \forall z \exists u \forall v (\Psi(x,y,z,u) \wedge (\Psi(x,y,z,v) \rightarrow u \leq v))$$

где

$$\Psi(a,b,c,d) = a \leq d \wedge b \leq d \wedge c \leq d \ .$$

Однако для контроля свойства φ целесообразно использовать φ, как алгоритм. Для этого целесообразнее использовать другую, сформулируемую также на этом языке формулу:

$$(6)\quad \varphi' = \forall x \forall y \exists u \forall v (\Psi'(x,y,u) \wedge (\Psi'(x,y,v) \rightarrow u \leq v))$$

где

$$\Psi'(a,b,c) = a \leq c \wedge b \leq c.$$

Очевидно, что для любого "poset" P $\varphi \longleftrightarrow \varphi'$, выражая тем самым тот факт, что достаточно доказать только для пар элементов существование минимального верхнего предельного значения.

Эти примеры были приведены только для иллюстрации того, что сформированный подходящим образом, но все же еще простой язык может быть пригоден для выполнения разных функций.

1.3. Синтаксис языка пользователя

Пользователь может сформулировать предложения данных, реляционные и абсолютные предложения, соответствующие формам (2) и (3).

Сначала займемся формированием реляционного предложения. Рассмотрим формулу (2).

(7) имя : < последующие имена в местах имен селекторов в (2) >

Имеющиеся здесь имена представляют собой имена инстанций, соответствующих типам селекторов в (2). /Здесь также как и в дальнейшем имеется ввиду общее правило согласования типов. См. [5], [6]./ Допускается также и следующий, более короткий вариант:

(8) < имена в местах селекторов в (2) >;

Это позволяет создать безимянные инстанции. Подобная необходимость возникает особенно при описании явлений типа связей.

Абсолютные предложения могут также просто сформулированы на основе форм типа (1) и (3). Общий вид их следующий:

(9) имя : имя 1 < последующие имена в местах селекторов в (3)>

Здесь имя 1 представляет собой имя понятия (1), а остальные имена - имена инстанций соответствующих типов. Весьма часто бывает, что часть в скобках отсутствует. В таких случаях допускаются два равноценных решения:

(10) имя : имя 1;

имя 1 имя;

Соответствующая этому случаю форма не должна быть отдельно описана: она считается автоматически существующей.

На основе сказанного в пп. 1.2. можно говорить также и о типах и взглядах предложений данных. Тип предложения данных совпадает с типом соответствующей формы. В случае взгляда подобное описание должно быть дополнено. Взгляды реляционного предложения представляют собой взгляд формы и подчиненные ему понятия. Такая конвенция является естественной, поскольку и специальное понятие может рассматриваться со всех таких точек зрения, которые являются разумными для более общего понятия.

1.4. Закон взглядов

В предыдущем пункте была представлена базисная единица объектного языка: предложение данных. В процессе коммуникации с системой SDLA из этих единиц могут быть построены сложные описания, запросы и т.д. Правило правильного составления описания, по-существу, заключается в том, чтобы взгляд любого из раскрываемых предложений данных стал верным /на основе предыдущих предложений/, предложение было конкретно-правильным. Следовательно ниже две вещи будем определять параллельно и рекурсивно, а именно, ряд верных взглядов H_n и то, что после конкретно-правильного ряда предложений данных

(11) $t_0, t_1, \ldots, t_{n-1}$ $\quad n \geq 1$

описание

(12) $s = t_0, t_1, \ldots, t_{n-1}, t_n$

когда является конкретно-правильным.

i/ если t абсолютное предложение, тогда ряд $t_0 = t$ является конкретно-правильным и $H_0 = \{$ тип предложения t $\}$

ii/ допустим, что ряд (11) является контекстно-правильным и H_{n-1} представляет собой ряд верных взглядов; тогда ряд (12) является конкретно-правильным, если

а/ t_n - абсолютное предложение.

Пусть в этом случае $H_n = \{$ тип предложения $t_n \}$.

б/ t_n - реляционное предложение и в ряде верных взглядов $H_{n-1} = W_0 \ldots W_n$ имеется такой тип W_i, который является взглядом t_n.

Пусть $j = \max \{ i,$ взгляд $W_i\ t_n \}$. В этом случае $H_n = W_0 \ldots W_j, W_{j+1}$, где W_{j+1} представляет собой тип t_n.

Правило, изложенное в пп. i/ - ii/ называется законом взгляда. В пп. ii/б естественным путем был выбран взгляд для актуального предложения. В определенных случаях целесообразнее иначе осуществить выбор возможных взглядов. Язык владеет такими дополнительными средствами, при помощи которых может быть модифицировано определение H_n, если это необходимо. С их помощью могут быть удалены другие конечные сегменты ряда H_{n-1}, и таким образом описания становятся более простыми и короткими.

Роль и правило формирования ряда H_n могут быть хорошо проиллюстрированы при помощи т.н. "ямы взглядов". Верные взгляды находятся в яме взглядов таким образом, что внизу находится всегда абсолютный взгляд. При интерпретации нового предложения осуществляется выбор из ямы до тех пор, пока не будет найден

один из взглядов предложения. Если такого нет, то предложение является неправильным в данном контексте. Если взгляд найден, то предложение принимается и его тип закладывается /сверху/ в яму. Представленное здесь правило обращения с ямой соответствует ii/б.

Если имеется правильное, удовлетворяющее закону взгляда описание, тогда ко всем предложениям t_i могут быть соотнесены по одной из секций /блоков/. Секция предложения t_i состоит из предложений

(13) $t_i, t_{i+1}, \ldots, t_j,$

если тип t_i после произнесения t_ℓ $(i \leq \ell \geq j)$ еще находится в яме взгляда, но при произнесении t_{j+1} стирается.

Применение синонима "блок" в нашем случае неслучайное. Определенная здесь структура блока, как синтактическая категория, удовлетворяет тому же правилу, как соответствующие категории определенных языков программирования /например, АЛГОЛ 68, СИМУЛА 67/.

(14) В этом случае блок также представляет собой ряд следующих /синтактически/ друг за другом "предложений", и если два блока содержат общее "предложение", то один из них содержит в себе другой.

Нетрудно увидеть, что если задать произвольную, удовлетворяющую этому правилу иерархию блоков, то она может быть реализована при помощи программы на языке АЛГОЛ 68, возможно, присоединением некоторого наиболее крайнего блока. Покажем, что секции объектных описаний SDLA, удовлетворяющие закону взгляда, удовлетворяют и правилу (14), т.е. они образуют блочную структуру. Точнее говоря, справедливо следующее:

1.1. Утверждение: Пусть будет дано описание SDLA вида (12),

удовлетворяющее закону взгляда, двумя секциями которого являются S_1 и S_2 так, что предложение $t_k (0 \leq k \leq n)$ содержится как в секции S_1, так и в секции S_2. Тогда $S_1 \geq S_2$ $S_2 \geq S_1$. /Отметим, что здесь S, S_1, S_2 математически следует считать либо последовательностью, либо подпоследовательность!/

Доказательство: Пусть будет t_i начальным предложением S_1, а t_j - начальным предложением S_2; пусть будет сначала $i \leq j$. Ввиду того, что t_k содержится в обоих секциях, k - тое предложение было произнесено позже, нежели i - тое и j - тое и после его произнесения в яме взглядов находятся еще и типы t_i и t_j. Поэтому вследствие $i \leq j \leq k$ по произнесении t_j в яме находится также /еще/ и тип t_i, следовательно тип t_i не находится выше, чем тип t_j. Вследствие особенности ямы тип t_i не может быть удален раньше типа t_j, следовательно согласно определению секции $S_1 \geq S_2$. Аналогичным рассуждением может быть доказано, что если $i < j$, то $S_2 \geq S_1$. □

С каждой из секций может ассоциироваться один из типов, а именно тип ее начального предложения. /Как любопытный факт отметим, что эта абстракция берет свое начало из жаргона применения SDLA. В частности, секция, начинающаяся предложением типа "file", мы называли просто файловой секцией./ Знание системы секций и типов секций вместе взято уже, означает собой релевантную информацию. С их помощью в любой точке описания можно предсказать актуальное содержание ямы взглядов и решить относительно следующего предложения: будет ли оно правильным в заданном контексте. Для этого, разуемеется, достаточно знать начатые до этого блоки и их типы. Тем самым представляется также возможной и перефразировка закона взгляда:

1.2. Утверждение: Пусть

(15) $t_1, \ldots, t_i$

представляет собой описание, удовлетворяющее закону взгладов.

После произнесения t_i в яме взгляда внизу находится абсолютный взгляд, а над ним подряд имеются типы тех секций, которые не заканчивались перед t_i. Тип более широкой секции находится глубже, чем тип более узкой секции. (15) можно продолжить предложением t_{i+1}, если t_{i+1} является абсолютным предложением, либо (15) имеет такую секцию, которая может быть расширена предложением t_{i+1}.

Доказательство: Из определения секции вытекает, что упомянутые взгляды находятся в яме, и в яме ничего другого нет. Рассматриваемые блоки все содержат t_i; следовательно более широкий из них непременно начинается раньше, и так его тип раньше попадает в яму. Последнее утверждение является непосредственным последствием ранее сказанного. □

И так, закон взгляда может рассматриваться также как и закономерность, существующая между секциями и их типами. Структуры секций, систему включений можно хорошо продемонстрировать позиционированием предложений внутри строки. Разумеется, что от пользователя нельзя требовать осуществления позиционирования, поэтому SDLA автоматически перестраивает предложения описания в такую форму. Это облегчает также и быстрое обнаружение ошибок.

Контроль закона взгляда /context checking/ представляет собой один из наиболее важных контролей SDLA, осуществляемый при любой коммуникации с системой, использующей объектный язык. При таком контроле, по-существу, проверяется наличие гипо- и паратаксисов, описанных на мета-уровне, с помощью конкретных испытаний. При описании модели посредством SDLA можно осуществить контроль различных свойств в рамках context checking. Разумеется, что для этого необходимо задать систему форм (2) и (3) подходящим образом. При этом возникают различные интересные методические проблемы. Однако их выяснение и решение еще требуют сбора дальнейших опытов и проведения исследований.

1.5. Пример

Ниже приводится простой пример применения описанных выше средств на основе 5 . Задается понятийная схема вместе с соответствующими формами. При помощи созданного таким образом языка сформулируется объектное описание и в связи с этим представляется также и функционирование ямы взгляда. Наконец дается представление о семантике описания при условии, что она интерпретируется в фазе описания. Рассмотрим следующую понятийную схему:

(16)
```
concept system component;
concept process;
concept data;
concept entity;
concept condition;
concept information;

concept belong (process, system component);
form process: belongs to system component;

concept use (process, data);
form process: uses data;
form data: is used by process;

concept produce (process, data):
form process: produces data;
form data: produced by process;

concept purpose (use);

concept derivation is purpose (entity);
form use: to derive entity;

concept update is purpose (information);
form use: to update information;

concept derivation condition (derivation, condition);
form derivation: under condition;
```

В описании не были использованы селекторы. Вместо них могут быть использованы соответствующие имена атрибутов, поскольку это в данном случае не приводит к недоразумению. Например, форма задания "use" в (16) можно рассматривать в качестве сокращения предложения

(17) concept use (process: process, data: data);

С помощью созданного таким образом объектного языка можно сформулировать следующее:

```
(18) process X;
        belongs to S;
        uses D1, D2;
           to derive E;
              under C;
        uses D3;
           to update I;
        uses D4, D5;
        produces D6;
     data D7;
        is used by Y;
        produced by Z;
        и т.д.
```

В описании (18) предложения

```
(19) process X; и
     data D7;
```

представляют собой абсолютные, а остальные - реляционные предложения. Строки описания позиционированы по секциям или же иначе, по глубине ямы взглядов. Нам кажется: по приведенным примерам видно также и то, что посредством использования возможностей фазы определения может быть создан целевой язык более высокого порядка, стоящий близко к человеческому мышлению.

На основе знания секций легко высказаться относительно содержания ямы взгляда в любой точке описания. Для этой цели необходимо использовать результат 1.2 A' . Например, после произнесения

(20) under C;

содержание ямы взгляда следующее:

(21)

derivation condition
derivation
use
process
abszolut

Последующее предложение будет верно только с точки зрения "process", следовательно после его произнесения яма будет иметь вид:

(22)

use
process
abszolut

Рассмотрим, что описание 18 какой смысл имеет в фазе описания данных! Инстанции, как правило, генерируются таким образом, что каждым предложением создается инстанция, сочетающаяся с типом предложения, если еще не имеется таковой с таким же именем. Если один из атрибутов, фигурирующихся в предложении, еще не был сгенерирован, то и это произойдет. Референции созданных инстанций определяются в соответствии с контекстом. Вместо подробной интеритерации этого ниже задаются инстанции, соответствующие (18), в логической схеме.

(23)
```
process X;
system component S;
data D1;
data D2;
data D3;
data D4;
data D5;
data D6;
entity E;
condition C;
information I;
belong (X, S);
use A1 (X,D1);
use A2 (X,D2);
derivation A3 (A1,E);
derivation A4 (A2,E);

derivation condition (A3,C);
derivation condition (A4,C);

use A5 (X, D3);
update (A5,I);

use (X,D4);
use (X,D5);

produce (X,D6);


data  D7;
process Y;
process Z;

use (Y,D7);
produce (Z,D7);
```

Идентификаторы, начинающиеся здесь с буквы "A", представляют собой "внутренние индикаторы", нефигурирующиеся в первоначальном тексте.

2. ПОД'ЯЗЫК SET

В предыдущем разделе были представлены основные синтактические аспекты объектного языка SDLA и было показано также на одно из его применений, в частности на то, что с его помощью каким образом могут быть заполнены реляции, созданные на уровне определений. В настоящем разделе объектные языки будут рассмотрены, как средства запросов.

Среди запросов, относящихся к созданной модели SDLA, одним из наиболее важных является т.н. запрос set. При помощи запросов set могут быть созданы однородные множества инстанций, т.е. такие множества, к которым может быть соотнесен независимый от актуального состояния базы данных /значит, вычисляемый/ тип. Разумеется, что тип элементов множеств должен быть подтипом типа множества. Вышеприведенное неявно означает и то, что под действием операций set не возникают новые, ранее несуществующие инстанции.

Наша концепция здесь резко расходится с принципами реляционного управления данными коммерческого назначения. Нами здесь не разрешаются такие операции, которые приводят к возникновению реляций, неимеющих типов /например, прямое произведение, join/. Из проекций могут осуществляться только проекции специального вида. Следовательно, подъязык set обладает меньшей силой выражения, чем реляционный язык запросов Codd-а. Наше решение подтверждается следующими:

1. На основе осуществленных до настоящего времени приложений /на предприятиях INORGA, VOLÁN/ становится ясно, что пользователи считают весьма важным то, чтобы объекты, произведенные в процессе анализа и запросы, были простыми, т.е. чтобы не было необходимости углубляться в информационно-логических проблемах конкретного представления.

2. На наш взгляд, в практических приложениях структурные свойства систем в большинстве случаев могут быть выражены и с

точки зрения отдельных составляющих ее элементов.

3. Мы считаем, что язык запросов, а значит и его эффективность и силу описания целесообразно разработать в зависимости от моделируемого явления /системы/.

4. Исключенные формально из системы операции /например, join/ при тщательной подготовке на мета-уровне в конкретных случаях могут быть реализованы.

Ниже сначала представляется синтактика и семантика описаний set, а далее приводятся некоторые проблемы логического характера. Анализируются сила описания рассматриваемого аппарата, а также структура запрашиваемых формул.

2.1. Спецификация set

Сначала определяем понятие обозначения множества. Допустимыми обозначениями множеств /в фазе запросов/ являются следующие:

1. any

   Это означает собой совокупность объектов, ассоциированных с местом появления типов /немаркированная переменная/.

2. any - имя переменной

   Означает собой тоже самое, что и 1. Представляет собой маркированную величину.

3. имя переменной

   Оно действительно согласно принципу локальности. Означает собой всегда точно тот самый объект, к которому конкретно относится соответствующее обозначение вида 2.

4. имя объекта

   Означает собой /одноэлементное/ множество, состоящее из именованного объекта.

5. имя set

   Означает собой множество объектов, определенное описанием set.

6. имя coset

   Означает собой дополнение множества, определенного описанием set, относящееся к множеству всех инстанций заданного типа.

7. имя set

   Оно действует в течение всего процесса запросов при условии, что оно было сформировано ранее в виде 5 или 6. Означает собой то же множество объектов, которое уже существует под этим именем.

Выше были представлены все возможности обозначения множеств, которыми можно пользоваться при запросах. Обозначения 1.,.2., 3., 4., 7 в форме объектного языка могут быть записаны всюду, куда раньше могли быть записаны имена инстанций. На основе обозначения множества вида 4 можно утверждать, что оно означает собой действительное расширение имеющегося ранее синтаксиса. Ввиду того, что к любому из допустимых множеств может быть соотнесен производный /на мета-уровне/ тип, в случае даже таких общих предложений данных имеет смысл говорить о типе или же возможных взглядах предложения.

Как это видно по 1 и 4, имеются и такие множества, которые как бы априори имеются в нашем распоряжении. В частности:

а/ Множества, состоящие из одной инстанции. Их типом является тип инстанций.

б/ Совокупность инстанций, относящихся к одному заданному типу.

Для определения других множеств могут использоваться спецификации set. Их вид может быть следующий:

(24) set имя set ;
<описание set>
equ;

или

(25) coset имя set;
<описание set>
equ;

Фигурирующееся выше "описание set" представляет собой такое описание, сформулированное на расширенном объектном языке, удовлетворяющее закону взгляда, в котором все абсолютные предложения имеют один и тот же тип. Общий для всех имеющихся в описании абсолютных предложений тип и будет типом создаваемого множества. Все элементы определяемого множества в общем смысле согласовываются с этим типом. Следует отметить, что тип обязательно существует, поскольку все описания, удовлетворяющие закону взгляда, начинаются с абсолютного предложения. Интерпретацию начнем с редукционного шага. Допустим, что задано некоторое множество следующей схемой спецификации:

(26) set S;
B_1
.
.
.
B_k
equ;

где $B_1, \ldots, B_k$ соответственно - абсолютные секции описания. Пусть S_i $(1 \leq i \leq k)$ представляет собой следующее множество:

(27) set S_i
B_i
equ;

Ясно, что если (26) представляет собой правильное описание set, тогда это справедливо и для (27). В этом случае согласно определению $S = \bigcup_{1}^{k} s_i$. Аналогично можно рассуждать и в случае комплементарных множеств. Если в спецификациях (26) и (27) вместо set поставить coset, то определение примет следующий вид: $S = s_i$.

Таким образом интерпретация была сведена к тому случаю, когда описание включает в себя только одну абсолютную секцию. Теперь определим, что следует подразумевать под тем, что некоторое обыкновенное, т.е. построенное исключительно при помощи одноэлементных множеств /инстанций/ объектное описание согласовано с моделью SDLA. Описание прежде всего должно быть контекстно-правильным. Дальнейшие ограничения интерпретируются индукцией, соответствующей длине описания:

i/ Если t_0 - абсолютное предложение, и /возможное/ имя инстанции, относящейся к его типу - P, а атрибуты, имеющиеся в предложении, - $Q_1, \ldots, Q_\ell$, то в модели должна быть инстанция P соответствующего типа, для которой атрибутами, относящимися к селекторам, являются инстанции $Q_1, \ldots, Q_\ell$.

ii/ Если описание

(28) $t_0, \ldots, t_{n-1}$

согласовывается с моделью SDLA, тогда это справедливо и для описания, дополненного предложением t_n, если

а/ t_n - абсолютное предложение и для него справедливо i/;

б/ t_n представляет собой реляционное предложение, непосредсвенно подчиненное предложению t_j,

/возможное/ имя инстанции, относящейся к t_n - P, инстанция, относящаяся к t_j - R, а атрибуты в t_n - Q_1, ..., Q_ℓ; в этом случае в модели должны существовать объекты P соответствующего типа, для которого атрибутом, относящимся к актуальному взгляду, является R, а атрибутами, относящимися к селекторам - Q_1, ..., Q_ℓ.

В случае неполных предложений условия наши могут соблюдаться, разумеется, только для имен /инстанций/, фигурирующихся в этих предложениях. Следовательно, некоторое описание согласовано с моделью SDLA в том случае, если соответствующие инстанции существуют, и соблюдаются между ними реляции, определенные в описании.

Рассмотрим ниже спецификацию set вида (2б) при условии k = 1. В этом случае тип первой строки спецификации будет представлять собой тип определяемого множества. Допустим, что в первой строке обозначение множества, соответствующее этому типу - R, а ряд дальнейших имен set, встречающихся в описании в порядке чтения - s_1, ..., s_m. Присутствующие здесь все имена представляют собой обязательно имена существующих уже ранее множеств. Множество, соответствующее маркированной переменной, вставится в ряд только при появлениях 2-ого типа.

Тогда

(29) $S = \{y \in R \mid \exists\, x_i \in s_i \ ; \ * \}$

* : = если вставить y вместо R, а x_i - вместо s_i, возникающее таким образом описание согласовано с моделью SDLA.

Иначе говоря: в новое множество включаются те объекты из R,

для которых могут быть найдены объекты $x_i \in s_i$ так, чтобы утверждение, имеющееся в описании, оказалось справедливым для совокупности $y, x_1, \ldots, x_m$. При выборе coset следует поступать точно также с той разницей, что при этом необходимо взять относящееся к типу дополнение множества, выбранного согласно правилу (29).

После интерпретации спецификации set программой запросов создается, "вычисляется" множество, которое в дальнейшем может быть использовано для создания других множеств, а также для формирования избирательных и структурных документов. Для этих целей множества могут быть доступны посредством их имен.

Говоря о средствах запросов, модно их классифицировать по методу выражения свойств. В своей основополагающей работе [3] **Gallaire, Minker** и **Nicolas** различают три основных класса:

1. Языки, основывающиеся на реляционной алгебре.
2. Языки, ориентированные на предикаты.
3. Языки, ориентированные на отображения.

На наш взгляд, система запросов **SDLA** может быть соотнесена скорее всего к третьей группе. Типичным примером подобных языков является **SEQUEL** [8].

Общий вид отображений на языке **SEQUEL** следующий:

(30)
```
SELECT A
FROM R
WHERE B = value
```

где **A** и **B** - атрибуты, а **R** - реляция. В системе запросов **SDLA** основные отображения формируются с помощью set.

Ниже приводятся несколько примеров для спецификаций set и coset. Наряду с определением (16) рассмотрим следующее описа-

ние множества.

```
(31)  set  процессы производства брака;
      process any;
         uses any;
           to derive брак;
      equ;
```

Это множество состоит из тех процессов, которые используются для создания элементов множества типа **entity** с именем брак. Теперь пусть часть определения имеет следующий вид:

```
(32)  concept woman;
      concept man /wife: woman/;
      form absolute: husband of wife;
      concept origin  /of: man, from: woman/;
      form of: son of from;
```

С помощью маркированной переменной лекго могут быть выбраны мужчины типа Эдина:

```
(33)  set Эдин;
      any: husband of any X;
         son of X;
      equ;
```

Это множество состоит из тех мужчин, которые одновременно являются мужьями и детьми какой-либо женщины. На основе (32) легко можем сказать, что кроме, например, Клеопатры и Савитри какие женщины еще могут быть найдены в базе данных:

```
(34)  coset  остальные;
      woman  Клеопатра;
      woman  Савитри;
      equ;
```

2.2. Выразительность подъязыка set

Ниже займемся некоторыми свойствами логического характера подъязыка **set**. Одним из важных свойств моделей **SDLA** является разнородность, что наблюдается на двух уровнях. С одной стороны, не различаются резко вещи, имеющие характер сущности /**entity**/ и связи; объекты обоих видов появляются, как элементы каких-либо реляций. С другой стороны, эти сущности и связи и сами бывают многообразными. Для описания модели, характеризующейся такой обширной разноцветностью, вместо логики первого порядка более пригодна т.н. многосортная логика. В ее рамках свойства "разнородных" структур могут быть более тонко рассмотрены.

Сначала кратко рассмотрим основные понятия многосортной логики. При этом, в первую очередь, укажем на различия от односортной, обыкновенной логики первого порядка.

Основными элементами некоторого многосортного языка α являются следующие:

(35) i/ множество $I \neq \emptyset$ - множество сортов;

ii/ $v^i_1, \ldots, v^i_n$ - символы переменных сорта $i \in I$;

iii/ множество символов функций n-переменных /s - сортные символы функции/ для всех $n \geq 0$ и $s = (i_1, \ldots, i_{n+1}) \in I^{n+1}$;

iv/ множество символов отношений n-переменных /r - сортные символы отношения/ для всех $n \geq 0$ и $r = (i_1, \ldots, i_n) \in I^n$;

v/ принятые символы высказываний;

iv/ кванторы $\exists_i$ и $\forall_i$ для всех $i \in I$.

Как и в случае односортных языков, и здесь можно говорить о термах, формулах, модели, истинности, теории и т.п. Здесь рас-

сматриваются такие языки, которые включают в себя только функции 0-переменных /значит, константы/, следовательно правилами формирования термов заниматься не будем. При построении формул необходимо придерживаться нового правила: с помощью квантора $\exists_i$ либо $\forall_i$ можно обозначить только переменную i-того сорта. В моделях языка α к каждому из сортов $i \in I$ можно соотнести подуниверсум и переменные i-того сорта могут принимать значения только в пределах его. В соответствии с этим изменяется и смысл квантификации.

С помощью вышеприведенных средств уже хорошо могут быть описаны основные особенности SDLA. При этом следует отметить, что любая многосортная логика может укладываться в рамках подходящей односортной логики /см. например, Enderton [1]/, следовательно оба формализма характеризуются приблизительно одинаковой выразительностью. И все-таки, в нашем случае рассмотренный выше подход является более естественным и удобным.

Совокупность понятий /concept/, сформулированных в части определений, пусть представляет собой I, множество сортов. Язык α будет включать в себя исключительно только символы реляций. Если f представляет собой понятие, атрибутами которого являются $f_1, \ldots, f_n$, тогда можно определить реляционный символ F, сортом которого является $(f, f_1, \ldots, f_n)$. F будет соответствовать реляции, определенной f. Все понятия определяют свои атрибуты, поэтому для модели SDLA, /которая одновременно является также и моделью языка α,/ истинны следующие аксиомы:

(36)

$$\forall_f x \forall_{f_1} x_1 \ldots \forall_{f_n} x_n \forall_{f_1} y_1 \ldots \forall_{f_n} y_n ((F(x,x_1,\ldots,x_n) \wedge F(x,y_1,\ldots,y_n)) \rightarrow$$

$$\rightarrow \left(\bigwedge_{i=1}^{n} x_i = y_i \right)) \quad \text{для всех понятий f.}$$

В фазе определений могут быть определены целостности /см. [5]/. Они выражают то, что некоторая система атрибутов представляет

собой ключ. Подобные аксиомы могут быть сформулированы аналогично (36), /где первый аргумент F является ключем/.

Возможность интенсионального определения представляет собой импликации /см. [5]/. Импликации могут быть выражены аксиомами следующего вида:

$$(37)\quad \forall_f x \forall_{f_1} x_1 \ldots \forall_{f_n} x_n \exists_g y \; (F(x,x_1,\ldots,x_n) \rightarrow G(y,x_{i_1},\ldots,x_{i_k}))$$

где f - определенное выше понятие, а g - атрибуты $f_{i_1},\ldots,k_{i_k}$.
Таким образом в фазе определений SDLA задаются многосортные языки α и теория Th, которые формулами (36) и возможными целостностями и импликациями аксиоматизируются. При вводе данных и генерации инстанций задается некоторая модель этой теории. Заданием типа инстанций определяются подуниверсумы, относящиеся к сортам, а заданием атрибутов определяются конкретные значения реляций в α. При этом контролируется соблюдение аксиом (36) и целостностей и, если необходимо, автоматической генерацией объекта обеспечивается истинность формул (37).

Пример: Рассмотрим, что при таком подходе что же собой означает инстанция A3 в (23). Ее сортом является direvation, которому в модели соответствует реляция трех переменных с именем DERIVATION и при этом соблюдается

(38) DERIVATION (A3, A1, E).

Для исследования запросов расширим язык α символами констант. Для каждой актуальной инстанции, а значит для каждого из элементов модели, вводится новый символ константы, интерпретацией которого является сама инстанция. /К подобному принято прибегать при исследовании диаграмм, а также элементарных диаграмм моделей. Символы констант следует рассматривать как символы функций 0-переменных./

Рассмотрим теперь, что означает собой следующее: некоторое объектное описание, содержащее в себе единственное абсолютное предложение, согласовывается с моделью SDLA. Покажем, что это равноценно следующему: некоторая формула $\varphi(x_1, \ldots, x_m)$ истина для подстановки $x_i = A_i$ $(1 \leq i \leq m)$, где A_i представляет собой необязательно инстанции, встречающиеся в описании, а φ - конъюнкцию простых формул, т.е.

(39) $\varphi = F_1 \wedge \ldots \wedge F_n$

Здесь, не различая имя и элемент, опираемся на определение и обозначения, приведенные в пп. 2.1. Если f является типом t_0, тогда значение 2.1.i/ равноценно тому, что при соответствующих инстанциях $R_1, \ldots, R_k$ выполняется

(40) $F(P, \vec{Q}, \vec{R})$

Записью векторов $\vec{Q}$ и $\vec{R}$ иллюстрируется то, что инстанции Q_i и R_i были записаны в места соответствующих аргументов реляционного символа F. Дальше можно воспользоваться индукцией. Допустим, что первая строка n-1 описания характеризуется формулой φ_1 вида (39). Пусть будет g типом t_n. Разумеется, что t_n - реляционное предложение. Смысл 2.1. ii/ равноценен тому, что

(41) $G(P, R, \vec{Q}, \vec{R})$

где $\vec{R}$ снова является вектором, состоящим из дополнительных атрибутов. Описание в целом согласовывается с моделью, если при наличии надлежащей подстановки формула $\varphi = \varphi_1 \wedge G$ является истинной и φ действительно имеет вид (39). Легко заметить, что формула φ зависить только от форм, применяемых в объектном описании. Когда в нее записываются имена инстанций, тогда осуществляется частичная оценка формулы. Описание может считаться согласованным с моделью тогда, когда частичная оценка может быть превращена в полную так, что при этом формула остается истинной.

Сформулированные выше определения множеств могут быть также записаны при помощи расширенного языка. Если A является инстанцией, а соответствующий ей символ константы - C_A, то одноэлементное множество, состоящее из A, может быть записано так:

(42) $\{y \mid y = C_A\}$.

Если необходимо выразить все инстанции типа f, то может быть записано следующее:

(43) $\{y \mid \exists_{f_1} x_1 \ldots \exists_{f_n} x_n \, F(y, x_1, \ldots, x_n)\}$.

Определение (29) примет следующий вид.

(44)

$$S=\{y \mid (y \in R) \wedge (\exists_{f_1} x_1 \ldots \exists_{f_m} x_m \exists_{g_1} y_1 \ldots \exists_{g_k} y_k ((\bigwedge_{i=1}^{m} x_i \in S_i) \wedge \varphi(y, \vec{x}, \vec{y})))\}$$

где φ - формула вида (39), соответствующая описателю set, а $\vec{y}$ - вектор, относящийся к дополнительным атрибутам. В случае дополнения необходимо взять отрицание формулы.

Последняя формула была составлена не полностью на языке α, поскольку в ней фигурируются также и подформулы $x_i \in S_i$, $y \in R$.

Однако здесь можно помочь:

2.1. Утверждение: Любое из множеств, определяемых с помощью подъяыка set , может быть представлено в виде:

(45) $\{y \mid \Psi(y)\}$,

где Ψ - формула расширенного языка α. Единственной свободной переменной Ψ является y.

Доказательство. Применяется индукция, соответствующая сложнос-

ти определения множества. Выражения (42) и (43) удовлетворяют требованиям к . Возьмем теперь множество S из (26). Если множества Ψ , соответствующие (27), были определены формулами $\Psi_i(y)$, тогда $\Psi = \Psi_1 v \ldots v \Psi_k$ пригодна для задания S и утверждение остается в силе и для Ψ. В случае определений coset подходящей формулой является $\Psi = \Psi_1 \wedge \ldots \wedge \Psi_k$. Наконец, если согласно (44) R задается формулой φ', а S_i - φ_i, тогда

(46)

$$S=\{y \mid \varphi'(y) \wedge (\exists_{f_1} x_1 \ldots \exists_{f_m} x_m \exists_{g_1} y_1 \ldots \exists_{g_k} y_k ((\bigwedge_{i=1}^{m} \varphi_i\ x_i)) \wedge \varphi(y, \vec{x}, \vec{y})))\}$$

и утверждение остается в силе и для этой формулы. В случае дополняющего множества необходимо взять отрицание фрмулы.

С помощью формул (42)-(44) легко может быть определена Ψ и для конкретных случаев. Например, для определения множества (33).

Можно воспользоваться следующим выражением:

(47) $\Psi(y) = \exists_w x \exists_o z(\mathrm{MAN}(y,x) \wedge \mathrm{ORIGIN}(z,y,x))$

Таким образом все множества могут быть заданы при помощи формул расширенного языка α. Однако, инверсия утверждения 1 неверна, система запросов set неполна. Это означает то, что с ее помощью невозможно задать все множества вида (45). Например, как правило, нельзя описать множества H, имеющие вид:

(48) $H = \{y \mid \forall x P(y, x)\}$,

где P представляет собой реляционный символ языка. /Рассматриваются те процессы, которыми используются все процессоры./ Это затруднение известно для разработчиков языков запросов как проблема отрицательной информации /см. [7]/. В структурах данных, заполненных преимущественно экстенсионально, в процессе ввода возникают исключительно положительные факты, утверж-

дения: такая информация, что некоторые вещи находятся друг от друга в реляции относительно какой-либо основной реляции. Ввод отрицаний подобных утверждений, т.е. отрицательной информации, как правило, так не можем осуществить. Количество отрицательной информации чаще всего многократно превышает количество верной положительной информации. Таким образом, вследствие постулирования отрицательной информации модели, в большинстве случаев, стали бы "тяжеловатыми" либо неиспользуемыми.

В заключении покажем, что формулы запросов $\Psi(y)$ являются закрытыми по отношению к комбинациям булевой алгебры.

2.2. Утверждение: Пусть $\Psi(y)$, $\Psi_1(y)$, $\Psi_2(y)$ являются формулами расширенного языка α, запрашиваемые с помощью подъязыка **set**. Тогда

а/ $\neg\Psi(y)$

б/ $\Psi_1(y) \wedge \Psi_2(y)$

в/ $\Psi_1(y) \vee \Psi_2(y)$

также характеризуются этим свойством.

Доказательство:

а/ Необходимо изменить первое слово в определении множества, соответствующим Ψ: вместо **set** следует записать **coset**, а вместо **coset** - **set**.

б/ Согласно правилу **De Morgan** достаточно, если а/ и в/ верны.

в/ Пусть имя типа, соответствующего y, будет: **name**. Тогда

(49)
```
set s;
name s1;
name s2;
equ;
```

является подходящей спецификацией, где s_1 и s_2 представляют собой имена множеств, соответствующих формулам Ψ_1 и Ψ_2 . □

Этот результат можно еще сформулировать так, что множества заданного типа являются закрытыми по отношению операций теории множеств.

Л И Т Е Р А Т У Р А

1. Enderton: A Mathematical Introduction to Logic; Academic Press, 1972.

2. Logic and Databases; Ed. Gallaire, Minker; Plenum Press, 1978.

3. Gallaire, Minker, Nicolas: An Overview and Introduction to Logic and Databases; in [2], 3-30.

4. Knuth, Radó, Rónyai: Relational Query Language in Reference Environment; MTA SZTAKI WP II/9, 1980.

5. Knuth, Radó, Tóth: Az SDLA elõzetes ismertetáse; MTA SZTAKI Tanulmányok, 104, 1980.

6. Knuth, Rónyai: Closed Convex Reference Schemes; Proc. VII. Symp. on Math. Programming, Varna, 1981.

7. Reiter: On Closed World Data Bases; in [2] 55-76.

8. Tsichritzis, Lochovsky: Data Base Manugement Systems; Akademic Press, 1977.

A TANULMÁNYSOROZATBAN 1981-BEN MEGJELENTEK:

116/1981 Siegler András: Egy 6 szabadságfokú antropomorf manipulátor kinematikája számítógépes vezérlése

117/1981 Knuth Előd–Radó Péter: Principles of Computer Aided System Description

118/1981 Demetrovics János–Gyepesi György: Általános függőségek és lekérdezéssel kapcsolatos algoritmusok relációs adatmodellekben

119/1981 Sztanó Tamás: REAL–TIME programrendszerek eseményvezérelt szervezése

120/1981 Szentgyörgyi Zsuzsa: A számítástechnika műszaki fejlődése és társadalmi hatásai

121/1981 Vicsek Tamásné (Strehó Mária): Vizsgálatok a kezdeti érték problémák numerikus megoldásával kapcsolatban

122/1981 Andó Györgyi–Lipcsey Zsolt: Sztochasztikus Ljapunov módszerek és alkalmazásaik

123/1981 Márkusz Zsuzsanna: Intelligens interaktiv rendszerek elvi problémái

124/1981 Márkusz Zsuzsanna: Logikai alapú programozási módszerek és alkalmazásaik számítógéppel segített építészeti tervezési feladatok megoldásához

125/1981 Fabók Julianna: Software impelementációs nyelvek

126/1981 Várszegi Sándor: Multimikroszámítógépes-rendszerek

127/1981 Lipcsey Zsolt: N-személyes minőségi differenciáljátékok késleltetéssel és késleltetés nélkül

128/1981 Böszörményi László: Multa-task rendszerek fejlesztése magasszintű nyelven

129/1981 Tóth János: A formális reakciókinetika globális determinisztikus és sztochasztikus modelljéről és néhány alkalmazásáról

A TANULMÁNYSOROZATBAN 1982-BEN MEGJELENTEK:

130/1982 Barabás Miklós – Tőkés Szabolcs: A lézer printer képalkotás hibái és optikai korrekciójuk

131/1982 RG-II/KNVVT "Szisztemü upravlenija bazami dannüh i informacionnüe szisztemü" Szbornik naucsno-iszszledovatel'szkih rabot rabocsej gruppü RG-II KNVVT, Budapest, 1979. Tom I.

132/1982 RG-II KNVVT Tom II.

133/1982 RG-II KNVVT Tom III.

Készült Békéscsabán, a MAHIR Nyomdaüzemében – 82.4237