УДК 004.8

А.Ф. Кургаев, С.Н. Григорьев

ОПРЕДЕЛЕНИЕ ФОРМАЛЬНЫХ ЯЗЫКОВ В МЕТАЯЗЫКЕ НОРМАЛЬНЫХ ФОРМ ЗНАНИЙ

Исследованы выразительные возможности метаязыка нормальных форм знаний по отношению к формальным языкам разного уровня. Даны формальные описания множества лингвистических примеров. Разработано графическое метаописание интерпретатора универсальной машины Тьюринга, эквивалентное приведенному текстовому описанию. Представленное определение интерпретатора универсальной машины Тьюринга может быть применено для решения задачи моделирования поведения любой Т-машины (решающей задачу преобразования данных на ее ленте памяти). Дано формальное определение транслятора детерминированной машины Тьюринга. Обоснована и определена возможность описания в метаязыке нормальных форм знаний интерпретирующе-транслирующего и транслирующе-транслирующего процессов постановки и решения произвольной задачи, имеющей решение.

Ключевые слова: метаязык, регулярные языки, контекстно-свободные языки, контекстные языки, 0-языки, машины Тьюринга, текстовое и графическое описания языка.

Введение

Для определения языков используют формальные грамматики, которые задают четверкой

$$G = <\mathbf{V_T}, \mathbf{V_N}, S, \mathbf{P}>,$$

где $\mathbf{V_T}$ и $\mathbf{V_N}$ – непересекающиеся терминальный и нетерминальный алфавиты; S – аксиома, начальный символ; **P** – конечная система правил подстановки. Каждой грамматике G однозначно соответствует анализируемый (или порождаемый) язык $L(G)$.

В зависимости от выразительных возможностей различают четыре уровня грамматик (и языков) с номерами от 3 до 0 в порядке возрастания их общности: автоматные (или регулярные), контекстно-свободные (КС), далее – контекстные (или НС-грамматики) и на вершине иерархии – уровня 0 и соответствующие им машины Тьюринга.

Множество **G** грамматик $G \in \mathbf{G}$ с единообразным набором структур правил множества **P** оформляют в некоторый метаязык для определения синтаксиса объектных (описываемых) языков использованием некоторого количества правил. Среди известных метаязыков наиболее популярны BNF (Backus-Naur Form) и Extended (расширенный) BNF [1].

Однако, в качестве языка представления знаний метаязык EBNF (и все другие известные метаязыки) имеет определенные недостатки. В частности, созданный первично для узкоспециальных целей и такой, что хорошо их обеспечивает, он не является функционально полным и потому не пригоден для представления произвольных знаний.

В работе [2] предложен и в [3, 4] описан метаязык нормальных форм знаний (НФЗ), развивающий выразительные средства EBNF. Для достижения функциональной полноты в метаязык EBNF введены:

- отношение отрицания, применимое к любому понятию в описании (обозначается знаком ^, предшествующим имени отрицаемого понятия);
- три элементарных операции над информационными структурами: распознавание, распознавание со следом и порождение, обозначаются соответственно знаками ?, # и !, присоединяемыми к имени понятия, вводящему некоторую информационную структуру.

Покажем, что для формальных языков каждого уровня в метаязыке НФЗ есть адекватные выразительные средства.

1. База знаний в метаязыке НФЗ

Любое описание предметной области в метаязыке НФЗ составляется из двух частей. Первая часть – это информационная структура множества определений понятий (нетерминалов), связанных базо-

выми отношениями метаязыка НФЗ. Вторая часть состоит из двух подмножеств терминалов – множества элементарных процедур и множества элементарных структур данных, которые реализуют на одном из языков программирования в форме единой библиотечной системы. Каждый из терминалов имеет собственное имя и в описании информационной структуры используется наравне с нетерминальными понятиями.

Определение 1. База знаний – это любое замкнутое на предметы предметной области ***D*** определение понятия в форме информационной структуры, элементами которой являются понятия, связанные между собой системой отношений метаязыка НФЗ, выполнимых на предметной области ***D***.

Определение 2. Данные – это предметные значения понятий.

2. Структура области данных

Общую предметную область ***D*** значений понятий представим двумя независимыми массивами – входным (INP) и выходным (OUT), каждый из них удовлетворяет аксиомам отношения предшествования [5]:

- никакой предмет не предшествует самому себе;
- если A предшествует B, а B предшествует C, то A предшествует C;
- если A и B – произвольные различные предметы, то либо A предшествует B, либо B предшествует A;
- в любом непустом классе предметов имеется первый предмет, то есть предмет, предшествующий всем остальным предметам класса.

С массивами INP и OUT свяжем переменные m и n, принимающие из множества натуральных чисел значения текущих координат соответствующих массивов, и две библиотеки пар процедур одноименных предикатов – библиотеку анализа над данными INP и библиотеку порождения над данными OUT. Переменные m и n будем считать независимыми, неявно заданными в квантификации всеобщности, аргументами (указателями на предметные значения) именованных понятий базы знаний.

Для каждой процедуры из библиотеки анализа есть одноименная процедура в библиотеке порождения и наоборот. Каждая процедура – это реализация операционального определения некоторого предиката в форме алгоритма обработки данных. Как правило, – это структурно простые действия типа сравнения с константой, записи, чтения, арифметические и другие вычисления.

Каждая процедура библиотеки анализа согласно своему содержанию:

- выполняет некоторое преобразование данных, взятых из массива INP, начиная с координаты, определяемой значением переменной m;
- формирует значение истинности, зависящее от успеха преобразования;
- при успехе (истинность – "истина") модифицирует значение переменной m, переводя указатель на начало следующего фрагмента массива INP.

Каждая процедура библиотеки порождения:

- порождает некоторый фрагмент данных в массив OUT, начиная с координаты, определяемой значением переменной n;
- формирует значение истинности "истина" и модифицирует значение переменной n, переводя указатель на начало следующего свободного фрагмента массива OUT.

Введем также набор системных процедур, управляющих этими элементами пространства данных, в том числе:

RB – всегда истинная процедура переключения библиотек. После ее выполнения вызов процедуры из библиотеки порождения заменяется вызовом процедуры из библиотеки анализа и наоборот;

RIO – всегда истинная процедура переключения массивов данных. После ее выполнения процедуры из библиотеки анализа обрабатывают данные из OUT, а процедуры из библиотеки порождения – данные из INP;

UIO – всегда истинная процедура объединения / разделения массивов данных. После однократного ее выполнения INP обрабатывается как два массива (INP и OUT) с всегда равными координатами анализа и порождения. Повторный ее вызов разделяет массивы.

Процедура RIO переключения массивов объединяет в единый процесс преобразование данных одного из массивов с преобразованием данных другого массива. То, что порождено в OUT, становится доступным для анализа и наоборот, то, что проанализировано из INP, может быть заменено порожденным.

3. Соотношения между языками для основных форм определения понятий в метаязыке НФЗ

Формальным языком, заданным на базовом множестве **U**, называют всякое подмножество свободной полугруппы **U***, т. е. множество цепочек из элементов **U** (подмножество множества **U***), вследствие чего на формальные языки распространяются теоретико-множественные операции объединения, умножения, дополнения, пересечения, возведения в степень [6–8]. Согласно теории понятий [9] любому понятию (наравне с содержанием, смыслом) присуще множество предметных значений, составляющее объем понятия. Основываясь на одинаковой (теоретико-множественной) природе объема понятия и формального языка в [10] обоснован ряд утверждений, принятых здесь в качестве исходных свойств метаязыка НФЗ.

Утверждение 1. Определение в форме альтернативы задает язык определяемого понятия как *объединение* языков понятий определяющей части.

Утверждение 2. Определение понятия в форме последовательности задает его язык как *произведение* языков понятий определяющей части.

Утверждение 3. Определение понятия в форме последовательности двух понятий, одно из которых взято с отрицанием, задает язык определяемого как *дополнение* языка отрицательного до языка положительного понятия.

Утверждение 4.

а) *отрицательное* определение любого понятия в форме *альтернативы отрицательных* понятий задает язык как *пересечение* языков понятий, входящих в альтернативу.

б) определение любого понятия последовательностью двух понятий, одно из которых взято *с отрицанием* и задано *альтернативой отрицательных* понятий, а другое – положительное, задает язык понятия как *пересечение* языков понятий, входящих в определяющую часть альтернативы.

Утверждение 5. Определение любого понятия в форме *итерации* другого понятия задает язык понятия как *итерацию* (операция Клини) языка понятия, входящего в определяющую часть.

Так, если понятие с именем A определено как итерация понятия с именем B:

$$A = (B);$$

то язык L_A определен итерацией языка L_B, т. е. произвольной *степенью* (в том числе *нулевой*) языка L_B.

Утверждение 6. Язык всякого понятия (с именем A), являющийся *ненулевой степенью* языка другого понятия (с именем B), задается рекурсивным определением формы:

$$A = B\,A/B;$$

4. Определение регулярных языков

Язык является регулярным, если он задан грамматикой, правила которой имеют форму:

$A \to x$ и/или $A \to y\,B$, допуская

$A \to x\,A$ ($x, y \in \mathbf{V_T^+}$; $A, B \in \mathbf{V_N}$).

Поскольку любая конечная последовательность слов выразима конечным произведением и/или конечной степенью языков, включающих лишь один символ

алфавита $\mathbf{V_T}$, то в силу утверждений 1, 5 и 6 справедливо

Утверждение 7. Любой регулярный язык может быть определен в метаязыке НФЗ.

Пример 1. Язык [7]:

$$L = \{a^n \mid n \geq 0\} \ ,$$

определяется итерацией

$$L = (\ 'a'\ ).$$

Пример 2. Язык [7]:

$$L = \{a^n \cup b^m \mid n, m \geq 1\},$$

определяется в форме

$$L = \ 'a'\ \ ('a'\ ) \ / \ 'b'\ ('b').$$

В силу замкнутости класса регулярных языков относительно умножения, объединения, итерации, дополнения, пересечения [6–8] (и, добавим, – ненулевой степени), справедливо

Утверждение 8. Язык любого понятия, определение которого дано с использованием лишь форм, указанных в формулировках утверждений 1–6, является регулярным.

5. Определение контекстно-свободных языков

Язык является контекстно-свободным, если задан КС-грамматикой, правила которой имеют форму:

$A \to \varphi$, где $A \in \mathbf{V_N}$, $\varphi \in (\mathbf{V_N} \cup \mathbf{V_T})^*$.

Класс КС-грамматик включает класс регулярных грамматик в качестве собственного подкласса и отличается от него допущением правил с самовставлением нетерминальных символов (теорема 11 [8]).

В самоопределении метаязыка НФЗ [4] нет ограничений на самовставление определяемого понятия, следовательно, справедливо

Утверждение 9. Любой КС-язык можно определить в метаязыке НФЗ.

Пример 3. Язык [6, 12]:

$$L = \{a^n b^n \mid n > 0\},$$

определяется предложением метаязыка НФЗ в форме альтернативы последовательностей:

$$L = \ 'a'\ \ L\ 'b'\ /\ \ 'a'\ \ 'b'\ .$$

Пример 4. Язык [6, 8]:

$$L = \{a^n b^n a^m \mid n, m > 0\}$$

определяется в форме:

1) $L = S_1\, S_2$;
2) $S_1 = \ 'a'\ \ S_1\ 'b'\ /\ 'a'\ 'b'$;
3) $S_2 = 'a'\ \ S_2\ /\ 'a'\ .$

Утверждение 10. Язык любого понятия, определение которого дано с использованием форм, указанных в утверждениях 1–6, и форм с самовставлением, является КС-языком.

6. Определение контекстных языков

Язык является контекстным (НС-языком), если задан НС-грамматикой, правила которой имеют форму:

$\varphi_1\, A\, \varphi_2 \to \varphi_1\, \omega\, \varphi_2$, где $A \in \mathbf{V_N}$; φ_1, φ_2, $\omega \in (\mathbf{V_N} \cup \mathbf{V_T})^*$ и ω не пусто.

КС-грамматики составляют подкласс НС-грамматик при ограничении правилами с пустым контекстом φ_1 и φ_2.

Вследствие незамкнутости КС-языков относительно операций дополнения и пересечения (теорема 21 [8]), в класс контекстных включаются языки, заданные с использованием этих операций над КС-языками. В силу утверждений 3 – 4 (формулировка которых не ограничивает сложность языков, над которыми определены эти операции) справедливо

Утверждение 11. Существуют НС-языки, для определения которых достаточно теоретико-множественных операций и форм с самовставлением.

Пример 5. Язык [6–8, 11–13]:

$$L = \{a^m b^m a^m \mid m \geq 1\}$$

формально определяется описанием метаязыка НФЗ, имеющим следующий смысл. Если первые две (из трех) подцепочки

равны по числу символов (контекстное условие), то первая подцепочка произвольна по длине, а вторая – равна третьей:

1) $L = \wedge S_1\ ('a')\ S_2$;

2) $\wedge S_1 = 'a'\ \wedge S_1\ 'b' / \ 'a'\ \ 'b'$;

3) $S_2 = 'b'\ S_2\ 'a' / \ 'b'\ \ 'a'$.

НС-грамматики занимают промежуточное положение между контекстно-свободными и самыми общими (0-грамматиками): в структуре их правил вместо одного нетерминала слева (как в КС-грамматиках), хотя и в ограниченном виде, но уже есть цепочка, поэтому они, не решая всех проблем формализации языков, приобретают проблемы реализации 0-грамматик.

Кроме теоретико-множественных операций, в теории формальных грамматик для определения языков используется дерево составляющих [6] (структура составляющих, С-маркер [12]) конкретного предложения. С-маркер строится конечным преобразователем (на его ленте памяти) в процессе вывода предложения по заданной грамматике языка. Известно [6, 8], что С-маркер можно построить для любого предложения любого КС-языка.

В описанной [3, 4] реализации метаязыка НФЗ операция построения С-маркера предложения языка некоторого понятия именуется символом "#" (в текстовом описании присоединяется к имени понятия) и выполняется процедурой интерпретации с записью показателей структуры в память следа. С-маркер используется процедурой интерпретации в режиме порождения – как повторение конечного процесса вывода предложения. Операция построения С-маркера является аналогом логического оператора *неопределенной* дескрипции ("эпсилон-оператора" [9, 14]), выделяющего конкретный предмет из объема понятия, к которому он применен.

Множество форм метаязыка НФЗ не включает *определенной* дескрипции, называемой также "йота-оператором" [9] (рассмотренного в [15] определения понятия в форме равенства предметных значений), однако она нужна для ассоциативного поиска, выражения контекстных условий и для других целей. Семантику идентичности значения некоторого понятия (с именем N) текущему значению из массива INP будем выражать (с использованием процедуры RB переключения библиотек) в следующей общей форме:

A = RB N RB! / ^RB.

Ее смысл состоит в следующем: при интерпретации понятия с именем A после выполнения первого терминала RB вызов любого терминала из одной библиотеки заменяется вызовом одноименного терминала из другой. Поэтому при порождении предложения (связанного с понятием N) выполняется анализ из INP в точности такого же предложения. Если этот анализ успешен, то выполнение второго терминала RB восстанавливает исходный порядок вызова терминалов из библиотек и интерпретация определения понятия A оканчивается со значением истинности "истина", иначе – "ложь".

Пример 6. Язык [13]:

$$L = \{a^n b^m a^n b^m c^3 \mid n, m \geq 1\},$$

на примере которого доказано существование НС-языков, не являющихся КС-языками, *формально определяется* описанием в метаязыке НФЗ:

1) $L = \ S_1\ S_2 \#\ \ S_3\ 'c'\ \ 'c'\ \ 'c'\ \ \wedge 'c'$;

2) $S_1 = 'a'\ \ S_1 / \ 'a'$;

3) $S_2 = 'b'\ \ S_2 / 'b'$;

4) S_3 = RB S_1 S_2 RB! / ^RB.

Пример 7. Язык [12]:

L = {$x\ x$ | x – не пустая цепочка из букв "a", "b"} *формально определяется* описанием в метаязыке НФЗ:

1) $L = S_1 \text{RIO}\ S_2 \#\ \text{RIO}\ \ S_1 !\ S_3 \wedge S_1 / S_1\ L$;

2) $S_1 = 'a' / 'b'$;

3) $S_2 = (\ S_1)$;

4) S_3 = RB S_2 S_1 RB! / ^RB;

Исходная ситуация – в INP не пустая цепочка из букв "a", "b", OUT – пуст; **результат** – "истина", если в INP – xx, дополнительно – в OUT порождается x.

При интерпретации понятия L с понятием S_1 связывается значение первого символа цепочки из INP, с понятием S_2 – (возможно, пустая) цепочка символов из OUT, далее значение S_1 порождается в OUT, а интерпретацией S_3 проверяется совпадение конкатенации значений S_2 и S_1 с текущим значением цепочки из INP:

- если совпадает (значение S_3 – истина), то интерпретацией $\wedge S_1$ выполняется проверка окончания цепочки и, если цепочка исчерпана, то успешное завершение интерпретации понятия L, иначе выполняется вторая альтернатива определения L – пропускается первая буква текущего значения цепочки и рекурсивное повторение интерпретации понятия L;
- если не совпадает, то формируется значение S_3 – ложь, вследствие чего выполняется вторая альтернатива определения L с пропуском первой буквы текущего значения цепочки и рекурсивным повторением интерпретации понятия L. Если же выполнение второй альтернативы определения L не может быть продолжено из-за исчерпания цепочки букв, то формируется результирующее значение истинности ложь.

Рассмотренные общесистемные операционные средства, определения примеров известных НС-языков и отсутствие контрпримеров *обосновывают*

Утверждение 12. Всякий НС-язык можно определить в метаязыке НФЗ.

7. Определение 0-языков

Язык принадлежит уровню 0 (является 0-языком), если он задан 0-грамматикой с использованием правил "$\varphi \rightarrow \psi$", ограниченных лишь тем, что φ и ψ (цепочки из $\mathbf{V_N \cup V_T}$) не могут быть пустыми.

Общие системы подстановок универсальны: если существует формальная процедура задания языка, то его можно определить 0-грамматикой [6]. Множество правил 0-грамматики включает КС-грамматику, с каждой интерпретацией которой можно связать С-маркер (структуру составляющих [16] или, в другой терминологии, дерево составляющих [6]), множество трансформаций, каждая из которых *преобразует* набор (k, $k \geq 1$) С-маркеров в новый С-маркер, и множество правил, определяющих порядок применения трансформаций. Среди разновидностей трансформаций различают присоединение, перестановки, устранение (или добавление) элементов собственного анализа, любую из которых можно применять только к полностью проанализированной цепочке [16].

8. Универсальная машина Тьюринга

Универсальная машина Тьюринга – это U-машина с фиксированной структурой, имитирующая *поведение* любой машины Тьюринга (Т-машины).

Для Т-машины исходный результат состоит в том, что если язык L определен 0-грамматикой, то по ней можно построить Т-машину, которая *допускает* язык L, и наоборот [16, 17]. Тоесть утверждается однозначное соответствие между 0-языком и Т-машиной. Следовательно, для множества 0-языков *можно* построить множество Т-машин.

U-машина отличается тем, что на ней одной *можно* моделировать поведение всего множества Т-машин, каждая из которых *допускает* некоторый 0-язык, и тем самым *решить произвольную задачу* уровня 0, для которой есть решение.

В теории алгоритмов интерпретатор U-машины исследовался столь детально потому, что составляет отдельную задачу, сложность которой (по определению) не ниже уровня 0, так как в конечном итоге, именно интерпретатор решает произвольную задачу уровня 0 (интерпретируя модель любой Т-машины, решающей задачу уровня 0). В силу этого реализация в некотором языке интерпретатора U-машины гарантирует *достаточность* его выразительных возможностей для постановки и решения произвольной задачи (для которой есть решение) и, тем самым, *универсальность* машины, реализующей этот язык. А. Тьюринг предложил и показал способ построения

U-машины, М. Минский [18] разработал форму представления любой Т-машины на ленте U-машины и описал интерпретатор U-машины в форме диаграммы состояний.

Используя эти результаты, для формализации примем два *допущения* [19].

1. Пусть полубесконечная вправо лента U-машины отождествлена с совмещенными (после выполнения процедуры UIO) массивами INP и OUT пространства данных метаязыка НФЗ и разделена на зоны, расположенные слева-направо в следующем порядке:

- начальное состояние q_t некоторой произвольной Т-машины;
- первый, обозреваемый читающе-записывающей головкой Т-машины, символ a_t;
- конечное описание Т-машины (последовательность взаимосвязанных продукций), представленное пятерками q_i, a_i, q_j, a_j, d_i – текущее состояние Т-машины, считываемый символ, новое состояние Т-машины, записываемый символ и значение сдвига (0 – вправо, 1 – влево) читающе-записывающей головки Т-машины соответственно. Каждая из продукций имеет смысл выражения: "*Если* Т-машина находится в состоянии q_i и на ее ленте обозревается символ a_i, *то* заменить его на символ a_j и перейти в состояние q_j, в котором выполнить преобразование соседнего (слева или справа от a_i) символа ленты Т-машины". Каждое состояние q_i, q_j, q_t из множества Q пусть именуется одним символом латиницы или цифры. Входные символы a_i, a_j, a_t пусть принадлежат алфавиту Σ, составленному из букв латиницы и цифр, а также включают нулевой код;
- разделяющий выделенный символ Код0;
- полубесконечная вправо псевдолента Т-машины содержит конечную цепочку символов (за ней – нулевые коды), одно из ее знакомест занято маркером (тот же выделенный символ Код0), указывающим положение читающе-записывающей головки Т-машины.

2. Положение читающе-записывающей головки U-машины однозначно определяется значением переменной m (координаты INP).

В начальный момент: читающе-записывающая головка U-машины расположена в крайнем левом положении и обозревает символ, именующий начальное состояние q_t Т-машины; читающе-записывающая головка Т-машины установлена также в крайнем левом положении ее ленты (маркер Код0 записан в первой клетке псевдоленты Т-машины).

Семантику интерпретатора U-машины *формально определим* в метаязыке НФЗ:

1) интерпретация = состояние символ# (^Код0 ^правило ^преобразовать);

2) ^правило = та_пятерка? состояние символ сдвиг# / пятерка ^Код0 ^правило;

3) та_пятерка = RB состояние символ RB! / ^RB;

4) пятерка = состояние символ состояние символ сдвиг;

5) ^преобразовать = (^Код0 пятерка) Код0 (символ ^Код0) ^зап_сдвиг заменить ^чтение? Код0!;

6) ^зап_сдвиг = символ? сдвиг! / сдвиг!;

7) сдвиг = '0' / '1';

8) заменить = символ ^если_0? символ! /^если_0? символ! /^записать_символ;

9) ^если_0 = '0';

10) ^записать_символ = символ? символ! / символ!;

11) ^чтение = символ#;

База знаний состоит из 16 понятий, в том числе 11 нетерминалов определяют состояния U-машины (это число можно уменьшить до 10, заменив понятие "пятерка" его определяющей частью) и пяти терминалов: RB – переключения библиотек; "символ", "состояние" – значения из множеств Σ и Q соответственно; "сдвиг" – значения из двузначного множества {'0', '1'} и "Код0" – разделяющий символ или маркер. В ряду известных определений U-машины,

составленных в свое время Икено, Ватанабе и Минским, данное является наиболее *естественным* и *простейшим* по критерию К. Шеннона [20].

Графическое метаописание интерпретатора U-машины показано на рисунке. Очевидно, что оно эквивалентно приведенному текстовому описанию.

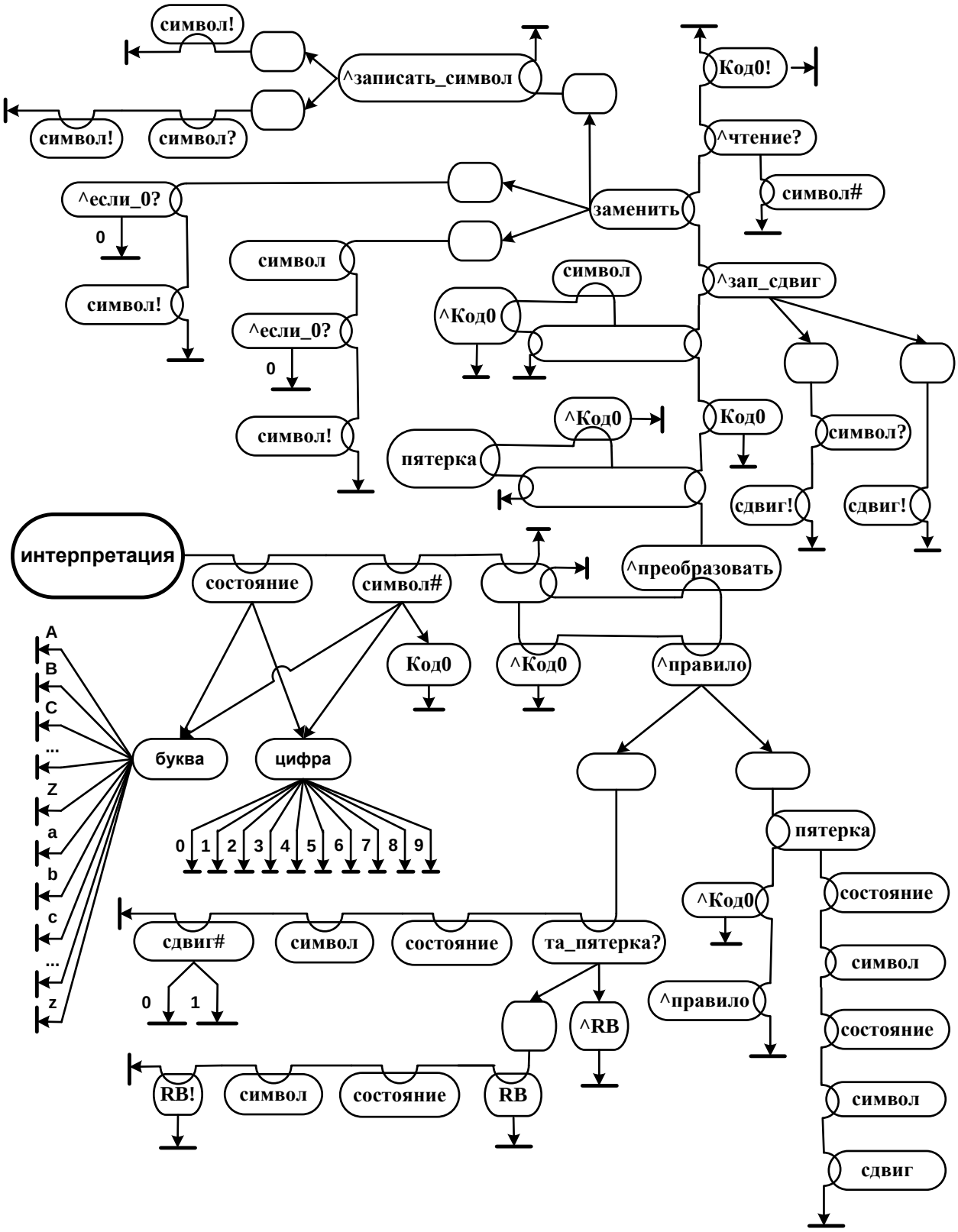

Рисунок. Граф метаописания интерпретатора U-машины в метаязыке НФЗ

Представленное определение интерпретатора U-машины может быть применено для решения задачи моделирования поведения любой Т-машины (решающей задачу преобразования данных на ее ленте памяти), которая формулируется обычным образом – как доказательство общего суждения: "Любое значение данных из INP (начало которых определено значением переменной *m*) суть интерпретация".Вывод этого суждения идет в следующей последовательности. При анализе со следом с понятиями "состояние" и "символ" связываются: начальное состояние Т-машины и первый символ на ее ленте. Затем в итерационном цикле интерпретируется последовательность понятий "^поиск_правила" и "^преобразовать", первое из них определяет ассоциативный поиск пятерки, описывающей очередную смену состояния моделируемой Т-машины, второе – замену на ленте Т-машины ранее взятого символа новым (из соответствующей пятерки), читает очередной символ (согласно указателю о сдвиге головки Т-машины) и записывает на его место маркер Т-машины. Поиск подходящего правила Т-машины выполняется просмотром пятерок описания Т-машины отождествлением текущего и связанного значений. Если проанализированные со следом состояние или символ не совпадают со значениями элементов очередной пятерки, то пятерка пропускается, иначе – анализируются со следом новое состояние, заменяющий символ и значение сдвига головки Т-машины. Если все описание Т-машины уже просмотрено безуспешно и на очереди разделяющий Код0, то формируется значение истинности "ложь", рекурсивный и затем итерационный цикл прерываются и интерпретатор заканчивает свою работу с заключительным значением истинности "истина". Иначе поиск пятерки рекурсивно повторяется до нужной пятерки или пока не выявится, что запомненное состояние – заключительное. Если пятерка найдена, – после последовательной интерпретации четного числа отрицаний головка U-машины возвращается к началу описания Т-машины. Интерпретация понятия "^преобразовать" начинается с пропуска описания Т-машины, затем пропуск разделяющего символа и далее поиск маркера. После того, как положение головки Т-машины найдено, вместо маркера записывается значение сдвига, затем анализ значения сдвига, запись на его место заменяющего символа, чтение очередного символа (слева или справа от маркера) и запись вместо него маркера.

Рассмотренная интерпретация представленного в метаязыке НФЗ определения U-машины подтверждает его семантическую эквивалентность диаграмме состояний, составленной М. Минским [18].

Эта задача – **пример** задачи *поведения* с фиксированной *стратегией*. Ее смысл состоит в преобразовании *данных* (представленных на ленте *любой* Т-машины) согласно *знаниям* (системе продукций в форме диаграммы состояний *конкретной* Т-машины) о преобразовании. Результат (в форме данных на ленте Т-машины) можно использовать как конечный результат или как исходные данные для следующей задачи. Работу U-машины можно возобновить после установки головок в начальное положение и смены данных и/или программы Т-машины.

Утверждение 13. Метаязык НФЗ включает семантические возможности U-машины, следовательно, любой 0-язык в форме интерпретатора может быть определен в метаязыке НФЗ.

Используя этот результат, *интерпретирующе-транслирующий процесс* постановки и решения произвольной задачи представляется следующей последовательностью преобразований:

- на метаязыке описать интерпретатор объектного языка и ввести в машину в качестве данных – разместить в массиве INP;
- выполнить ввод (трансляцию в машинное представление) определения интерпретатора объектного языка;
- на объектном языке описать задачу и ввести в машину в качестве данных – разместить в массиве INP;

- дать задание машине вывести общее суждение в форме "Любое значение данных из INP (чье начало определено значением переменной m) суть интерпретация объектного языка".

Результат: значение истинности последнего суждения и преобразованные данные.

9. Формализация в метаязыке НФЗ транслятора Т-языка

Любая детерминированная Т-машина описывается [7] конечным множеством продукций: $q_i \ a_i \rightarrow q_j \ a_j \ d_i$, форма которых совместно с определением множеств **Q** ($q_i, q_j \in$ **Q**), **Σ** ($a_i \ , \ a_j \in$ **Σ**) и **D** ($d_i \in$ **D**) задает язык Т-машин (Т-язык).

Имя каждого состояния ($q_i \ , \ q_j$) из конечного множества **Q** пусть принадлежит объему понятия "имя_понятия", алфавит **Σ** пусть составлен как объединение объемов понятий "буква", "цифра" и "знак" (их определение дано в [4]), и **D** – пусть две выделенных константы ('0', '1'), имеющих смысл сдвига читающе-записывающей головки вправо и влево соответственно.

Синтаксис этого языка *формально определим* таким образом:

1) Т-язык = продукция (продукция);

2) продукция = состояние1 пробел символ1 фон '→' фон состояние2 пробел символ2 пробел сдвиг;

3) сдвиг = '0' / '1';

4) состояние1 = имя_понятия;

5) состояние2 = имя_понятия;

6) символ1 = буква / цифра / ^мета_знак знак ;

7) символ2 = буква / цифра / ^мета_знак знак.

Семантика любой продукции состоит в смене одного состояния Т-машины на другое, если текущий символ совпадает с символом посылки; в случае применимости продукции – замена текущего символа символом заключения и сдвиг читающе-записывающей головки.

Семантику Т-языка *формально определим* таким образом:

1) семантика_Т-языка = Т_определения анализ_символов запись_символов;

2) Т_определения = Т_определение (Т_определение);

3) Т_определение = состояние1 Т_определяющее ';' ;

4) Т_определяющее = ' = ' Т_элемент ('/' Т_элемент / '/' 'истина');

5) Т_элемент = '^если_' символ1 пробел заключение;

6) заключение = '^записать_' символ2 пробел определение_сдвига пробел состояние2 ';' пробел;

7) определение_сдвига = символ2 / 'Сдвиг_влево' ;

8) анализ_символов = анализ_символа (анализ_символа);

9) запись_символов = записать_символ (записать_символ);

10) анализ_символа = '^если_' символ1 ' $=_{df}$ ' символ1 ';' пробел;

11) записать_символ = '^записать_' символ2 ' = ' символ2 '!' ';' пробел.

Согласно этому определению семантика любой конкретной Т-машины задается конечным (не пустым) множеством определений семантики всех ее состояний. Любое состояние Т-машины – это альтернатива допустимых вариантов замены текущего символа, сдвига читающе-записывающей головки (вправо, пропуская текущий символ, или влево, выполнив терминал "Сдвиг_влево") и перехода в одно из следующих состояний (не обязательно другое) в зависимости от значения текущего символа. Здесь "Сдвиг_влево" – всегда истинный терминал, определенный в форме семантической процедуры, выполняющей декремент значения m. На множестве состояний любой конкретной Т-машины считается известным (для использования) ее начальное состояние q_i, заключительное – определяется как

успешное завершение работы Т-машины, которое достигается из произвольного состояния в результате выяснения неприменимости преобразований текущего символа, предусмотренных определением состояния. Анализ и запись значений символов (исходно указанных в продукциях Т-языка) представлены как отдельные понятия, объединенные в соответствующие группы, расположенные (в синтаксическом представлении в метаязыке) после определений состояний Т-машины.

Семантику трансляции определения любой Т-машины из Т-языка в метаязык *формально определим* следующим образом:

1) Т-язык_в_метаязык = ^преобр_прод RIO ^определ_вперед ^анализы_вперед ^нормализация RIO;

2) ^преобр_прод = преобр_1 (преобр_1);

3) преобр_1 = состояние1 пробел символ1 фон '→' фон состояние2 пробел символ2 пробел# посылка '^записать_' символ2 пробел! преобр_сдвиг? пробел состояние2 ';' пробел анализ_символа записать_символ!;

4) посылка = состояние1 '$=_{df}$ ' '^если_' символ1 '?' пробел;

5) преобр_сдвиг = '0'? символ2! / '1'? 'Сдвиг_влево'!;

6) ^определ_вперед = Т_определение (^вперед_определение Т_определение);

7) ^вперед_определение = анализ_запись Т_определение# RIO Т_определение анализ_запись! RIO;

8) анализ_запись = (анализ_символа / записать_символ);

9) ^анализы_вперед = Т_определения (анализ_символов ^вначале_анализ);

10) ^вначале_анализ = записать_символ анализ_символа# RIO анализ_символа записать_символ! RIO;

11) ^нормализация = (склеить_определения) (удалить_дубли_анализа) (удалить_дубли_записи);

12) склеить_определения = состояние1 Т_определяющее# состояние1 Т_определяющее! ';' (^поиск_состояния? '/' Т_элемент!)? '/' 'истина' ';'!;

13) ^поиск_состояния = ^взять_определение? RIO хвост RIO! /Т_определение ^поиск_состояния;

14) ^взять_определение = то_же_состояние? '$=_{df}$ ' Т_элемент ';' хвост#;

15) то_же_состояние = RB состояние1 RB! / ^RB;

16) хвост = (Т_определение) (анализ_символа) (записать_символ) Код0;

17) удалить_дубли_анализа = анализ_символа# анализ_символа! (^тожд_анализ);

18) ^тожд_анализ = ^исключить_анализ? RIO хвост RIO! / анализ_символа ^тожд_анализ;

19) ^исключить_анализ = тот_же_анализ? хвост#;

20) тот_же_анализ = RB анализ_символа RB! / ^RB;

21) удалить_дубли_записи = записать_символ# записать_символ! (^тожд_запись);

22) ^тожд_запись = ^исключить_запись? RIO хвост RIO! / записать_символ ^тожд_запись;

23) ^исключить_запись = та_же_запись? хвост#;

24) та_же_запись = RB записать_символ RB! / ^RB.

Это определение, как и любое другое, становится выполнимым (интерпретируемым) после его ввода (трансляции в машинное представление).

Результат его интерпретации: представление в метаязыке НФЗ семантики любой Т-машины размещается с начала массива INP, замещая исходное представление в Т-языке. Логический результат: значение истинности – "истина", значения переменных m и n – исходные.

Согласно данному определению трансляция определения любой Т-машины из Т-языка в метаязык НФЗ состоит в следующей последовательности преобразований.

При интерпретации понятия "^преобр_прод" любая последовательность продукций преобразуется из исходного представления (в Т-языке) в представление в метаязыке, причем каждая продукция преобразуется независимо от других и каждой из них ставится в соответствие три определения взаимосвязанных понятий – состояния (посылки), анализа (символа посылки) и записи (символа заключения). Результат преобразования – в массиве OUT. По общему правилу интерпретации инверсии: логический результат – "истина", значения m и n – исходные.

Недостаток полученного представления (тот же, что и исходного) – при его интерпретации необходим ассоциативный поиск нужного определения текущего состояния из-за возможного наличия в описании Т-машины нескольких, расположенных в произвольном порядке, независимых разных определений всякого состояния. Последующие преобразования направлены на устранение этого недостатка и составляют *упорядочение и нормализацию* определения любой Т-машины.

При интерпретации понятий "^определения_вперед" и "^анализы_вперед" выполняется *упорядочение* в массиве OUT преобразованного определения Т-машины: определения всех состояний перемещаются в начало, анализы символов размещаются вслед за определениями состояний, а записи символов остаются в конце определения Т-машины.

В процессе интерпретации итерации понятия "склеить_определения" выполняется объединение множества различных определений любого состояния в одно – определение состояния в форме *альтернативы* допустимых в данном состоянии преобразований текущего символа и переходов в следующее состояние. В процессе интерпретации итерации понятия "удалить_дубли_анализа" исключаются тождественные анализы символов, а итерации понятия "удалить_дубли_записи" – тождественные записи символов. Результат этих преобразований порождается в массив INP, замещая исходное представление в Т-языке.

Полученное определение (в метаязыке) любой конкретной Т-машины после ввода и присоединения к имеющейся базе знаний может быть применено для решения соответствующей задачи обработки данных (введенных в массив INP), которая формулируется обычным образом: как доказательство общего категорического суждения формы "Любое конкретное значение данных из INP (начало которых определено значением переменной m) суть q_i", где q_i – имя начального состояния Т-машины, определение которой введено в базу знаний.

Представленная формализация транслятора Т-языка обосновывает

Утверждение 14. Любой 0-язык в форме транслятора может быть определен в метаязыке НФЗ.

Используя этот результат, **транслирующе-транслирующий процесс** постановки и решения произвольной задачи некоторой предметной области определим следующей последовательностью преобразований [10]:

- на метаязыке НФЗ описать транслятор языка предметной области и ввести в машину в качестве данных – разместить в массиве INP;
- выполнить ввод (трансляцию в машинное представление) определения транслятора языка предметной области и присоединить к имеющейся базе знаний;
- на языке предметной области описать задачу и ввести в машину в качестве данных – разместить в массиве INP;
- дать задание машине вывести общее суждение в форме "Любое значение данных из INP (начало которых определено значением переменной m) суть транслятор языка предметной области", результат – значение истинности суждения и определение на метаязыке НФЗ конкретной задачи предметной области;
- выполнить ввод (трансляцию в машинное представление) определения

(результат предыдущего пункта) этой конкретной задачи и присоединить к имеющейся базе знаний;

- исходные данные задачи разместить в массиве INP;
- дать задание машине вывести общее суждение в форме "Любое значение данных из INP (начало которых определено значением переменной m) суть начало программы конкретной задачи".

Результат: значение истинности последнего суждения и преобразованные данные, смысл которых определяется семантикой задачи.

Выводы

Исследованы выразительные возможности метаязыка НФЗ по отношению к формальным языкам разного уровня. Показано, что для формального языка каждого уровня в метаязыке НФЗ есть адекватные выразительные средства. Представленные результаты исследования, иллюстрированные множеством примеров, обосновывают вывод об универсальных выразительных возможностях метаязыка НФЗ, о его пригодности в качестве базового инструмента для реализации языков и систем произвольной сложности и назначения.

1. International Standard ISO/IEC 14977 : 1996(E). Электронный ресурс. Режим доступа: http://www.cl.cam.ac.uk/~mgk25/iso-14977.pdf
2. Спосіб представлення і використання знань / О.П. Кургаєв, С.М. Григор'єв / Патент на корисну модель **UA 92484 U**, 2014р., Бюл. №16.
3. Kurgaev A., Grygoryev S. The normal forms of knowledge. Dopov. NAN Ukraine, 2015, № 11. P. 36–43.
4. Кургаев А.Ф., Григорьев С.Н. Метаязык нормальных форм знаний. *Кибернетика и системный анализ*. 2016. Том. 52. № 6. С. 11–20.
5. Ершов А.П. Предварительные соображения о лексиконе программирования. *Кибернетика и вычислительная техника*. Под ред. В.А. Мельникова. 1985. Вып. 1. С. 199–210.
6. Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л. Алгебра. Языки. Программирование. К.: Наук. думка, 1978. 320 с.
7. Гросс М., Лантен А. Теория формальных грамматик. М.: Мир, 1971. 294 с.
8. Хомский Н. Формальные свойства грамматик. *Кибернетический сборник*. Новая серия. М.: Мир, 1966. Вып. 2. С. 122–230.
9. Войшвилло Е.К. Понятие как форма мышления: логико-гносеологический анализ. М.: Изд. МГУ, 1989. 239 с.
10. Кургаев А.Ф. Проблемная ориентация архитектуры компьютерных систем. Киев: Сталь, 2008. 540 с.
11. Тейз А., Грибомон П., Луи Ж. и др. Логический подход к искусственному интеллекту: от классической логики к логическому программированию. Пер. с фр.; Под ред. Г.П. Гаврилова. М.: Мир, 1990. 432 с.
12. Хомский Н. и Миллер Дж. Введение в формальный анализ естественных языков. *Кибернетический сборник*. Новая серия. М.: Мир, 1965. Вып. 1. С. 229–290.
13. Хомский Н. О некоторых формальных свойствах грамматик. *Кибернетический сборник.* М.: Изд-во иностр. лит., 1962. Вып. 5. С. 279–311.
14. Гильберт Д., Бернайс П. Основания математики: Теория доказательств. М.: Наука, 1982. 652 с.
15. Кургаев А.Ф. Логические формы определения понятия. *УСиМ.* 1998. № 2. С. 3–12.
16. Chomsky, N.: Aspects of the Theory of Syntax, MIT Press, Cambridge, MA, 1965.
17. Fu K.S. Syntactic Pattern Recognition and Applications. New Jersey: Prentice-Hall, Inc., Englewood Cliffs, 1982. 596 p.
18. Minsky M.L. Computation: Finite and Infinite Machines. NY: Prentice-Hall, Englewood Cliffs, 1967.
19. Kurgaev A., Grygoryev S. The Universal Turing Machine Interpreter. Dopov. NAN Ukraine, 2016, 10. С. 28–34. https://doi.org/10.15407/dopovidi2016.10.028
20. Shannon C.E. A universal Turing machine with two internal states. Princeton: Automata Studies, 1956.

References

1. International Standard ISO/IEC 14977 : 1996(E). [Electronic resourse]. – Mode of

access: http://www.cl.cam.ac.uk/~mgk25/iso-14977.pdf
2. Kurgaev A., Grygoryev S. Utility model patent UA 92484 U, 2014, Bulletin № 16 (in Ukrainian).
3. Kurgaev, A. The normal forms of knowledge / A.Kurgaev, S.Grygoryev. – Dopov. NAN Ukraine, 2015, № 11. – P. 36-43 (in Russian).
4. Kurgaev, A. Metalanguage of Normal Forms of Knowledge / A.Kurgaev, S.Grygoryev. – Cybernetics and Systems Analysis. – November 2016, 52(6). – P. 11-20. (in Russian).
5. Ershov, A.P. Preliminary considerations on the lexicon of programming // Cybernetics and computer technology / Ed. V.A. Melnikov. - 1985. - Vyp. 1. – P. 199–210. (in Russian).
6. Glushkov, V. Algebra. Languages. Programming / V.Glushkov, G.Zeitlin, E.Yushchenko. – K.: Nauk. Dumka, 1978. – 320 p. (in Russian).
7. Gross M. Theory of formal grammars / M.Gross, A.Lunten. – M.: Mir, 1971. - 294 p. (in Russian).
8. Chomsky, N. Formal properties of grammars. / Cybernetic collection. New episode. – Moscow: Mir, 1966. – Vyp. 2. – P. 122–230. (in Russian).
9. Voishvillo E. Concept as a form of thinking: logical-gnoseological analysis / E. Voishvillo. – Moscow: Izd. MSU, 1989. – 239 p. (in Russian).
10. Kurgaev, A.F. Problem orientation of the architecture of computer systems / A.F. Kurgayev. – Kiev: Steel, 2008. – 540 p. (in Russian).
11. Teiz, A. Logical approach to artificial intelligence: from classical logic to logical programming / A. Teiz, P. Gribomon, J. Louis et al. – Transl. With fr .; Ed. G.P. Gavrilov. – Moscow: Mir, 1990. – 432 p. (in Russian).
12. Chomsky, N. Introduction to the formal analysis of natural languages / N. Chomsky, J. Miller. – Cybernetic collection. New episode. – Moscow: Mir, 1965. – Vyp. 1. – P. 229–290. (in Russian).
13. Chomsky, N. On some formal properties of grammars / Cybernetic collection. – Moscow: Izd-vo inostr. Lit., 1962. - Vyp. 5. – P. 279–311. (in Russian).
14. Gilbert, D. Foundations of Mathematics: Theory of Evidence / D.Gilbert, P.Bernays. – M oscow: Nauka, 1982. – 652 p. (in Russian).
15. Kurgaev A. Logical forms of definition of the concept // Control Systems and Computers. – 1998. – № 2. – P. 3–12. (in Russian).
16. Chomsky, N. Aspects of the Theory of Syntax, MIT Press, Cambridge, MA, 1965.
17. Fu, K.S. Syntactic Pattern Recognition and Applications. – New Jersey: Prentice-Hall, Inc., Englewood Cliffs, 1982. – 596 p.
18. Minsky, M.L. Computation: Finite and Infinite Machines. – NY: Prentice-Hall, Englewood Cliffs, 1967.
19. Kurgaev, A. The Universal Turing Machine Interpreter / A.Kurgaev, S.Grygoryev. – Dopov. NAN Ukraine, 2016, 10. – P. 28-34. https://doi.org/10.15407/dopovidi2016.10.028 (in Russian)
20. Shannon, C.E. A universal Turing machine with two internal states. – Princeton: Automata Studies, 1956.

Получено 23.05.2017

Об авторах:

Кургаев Александр Филиппович,
доктор технических наук,
профессор, ведущий научный сотрудник.
Количество публикаций в
украинских изданиях – более 200.
Количество публикаций в
зарубежных индексированных
изданиях – около 10,
H-індекс (Google Scholar): 4
http://orcid.org/0000-0001-5348-2734,

Григорьев Сергей Николаевич,
соискатель.
Количество публикаций в
украинских изданиях – более 20.
http://orcid.org/0000-0002-7583-3088.

Место работы авторов:

Институт кибернетики
имени В.М. Глушкова НАН Украины.
03187, Киев-187,
проспект Академика Глушкова, 40.
Тел.: 050 881 6218,
050 505 05 75.
E-mail: afkurgaev@ukr.net,
Sergey@Grigoriev.kiev.ua