ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ

УДК 004.89
doi: 10.18101/2304-5728-2016-4-3-12

© *В. А. Кравченко, Д. Ш. Ширапов, С. И. Олзоева*

Использование функциональных грамматик в концептуальном программировании

Статья посвящена вопросам синтеза программ. Вместо широко распространенного метода семантических сетей на основе атрибутных грамматик предлагается использовать аппарат функциональных грамматик. На примере показаны универсальность и преимущества такого подхода.

Ключевые слова: инженерия знаний, решатель задач, синтез программ, функциональное программирование.

© *V. A. Kravchenko, D. Sh. Shirapov, S. I. Olzoeva*

The functional grammars in the conceptual programming

Article is devoted to the synthesis programs. The functional grammars are used instead of popular method the semantic networks based on the attribute grammars. The example shows the versatility and benefits of this method.

Keywords: knowledge engineering, problem solver, synthesis programs, functional programming.

Введение

Целью статьи является демонстрация возможностей аппарата функциональных грамматик для создания систем автоматизации программирования. Актуальной является задача выработки универсального метода синтеза программ, который был бы применим для любых языков программирования. Существующие методы концептуального программирования построены на использовании семантических сетей, описываемых атрибутными грамматиками [1]. Такой подход является эффективным, но не дает возможности получения универсальной программы в общем виде. Генерируемая программа является частной и написана на том же языке, что и сама система концептуального программирования. Использование функциональных грамматик позволяет получить программу в виде суперпозиции функций. Такой результат является более общим и из него могут быть получены программы на разных языках программирования.

1. Концептуальное программирование

Концептуальное программирование – это способ решения задач на ЭВМ, заключающийся в описании понятий для выражения смысла задачи. В терминах этих понятий описывается задача, строится программа решения и выполняются вычисления [1].

Свойства концептуального программирования:

1) программирование в терминах предметной области решаемых задач;

2) использование ЭВМ уже на этапе постановки задачи;

3) автоматический синтез программы решения каждой задачи;

4) использование семантической памяти для накопления знаний о решаемых задачах.

Согласно Лаврову С.С., знания подразделяются на три категории (рисунок 1): фактографические, алгоритмические и концептуальные [2].

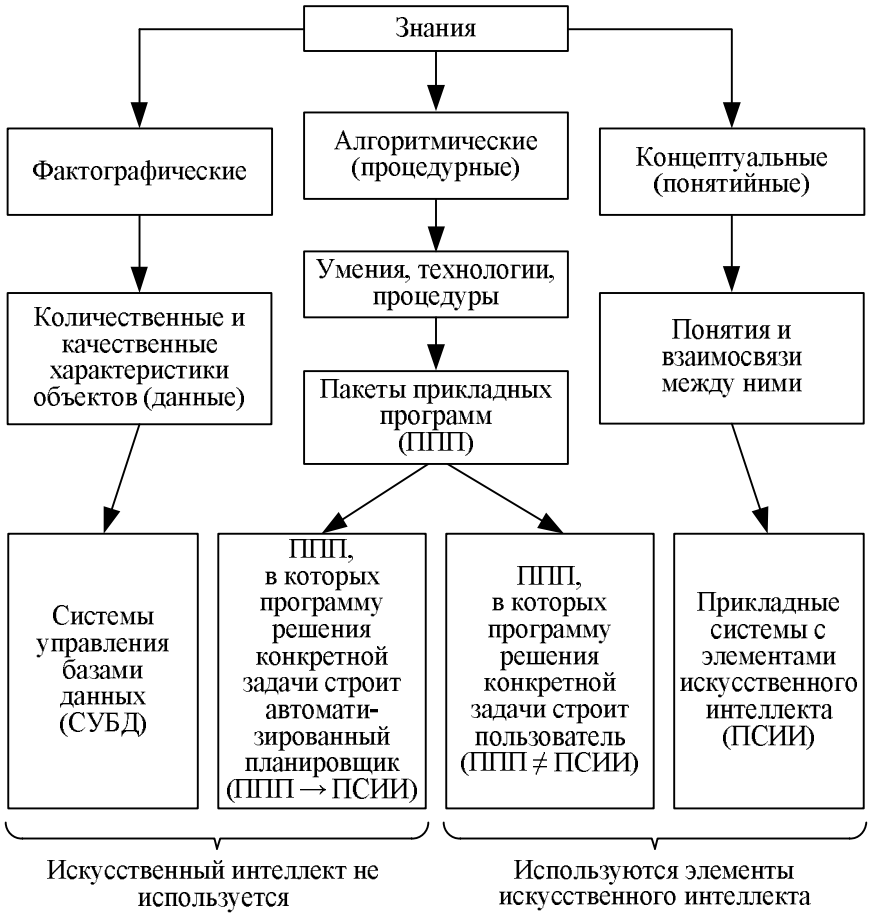

Рис. 1

Задачи, решаемые на основе фактографических знаний, вообще не требуют программирования, для них достаточно иметь уже готовую систему управления базами данных (СУБД). Задачи на основе алгоритмических

знаний требуют программирования для создания пакета прикладных программ (ППП) решения. Задачи на основе концептуальных знаний требуют создания прикладных систем с элементами искусственного интеллекта (ПСИИ).

Так как при концептуальном программировании решение задач строится на использовании терминов (понятий) рассматриваемой предметной области, т.е. концептуальных знаний (отсюда и название), то результат такого программирования всегда является системой с элементами искусственного интеллекта (ПСИИ). В отличие от этого, системы обычного (не концептуального) программирования могут иметь отношение к искусственному интеллекту только при определенных условиях (рисунок 1).

Таким образом, концептуальное программирование – самый актуальный вид программирования, всегда использующий методы искусственного интеллекта.

Важное значение в системе концептуального программирования, как и в любой интеллектуальной системе, занимает способ представления знаний. Он определяет алгоритм поиска решения и алгоритм синтеза программ. Это, в свою очередь, влияет на эффективность системы в целом.

2. Применение аппарата функциональных грамматик

В классической работе по концептуальному программированию [1], в системах ПРИЗ, СПОРА и в современных коммерческих проектах применяется представление знаний в виде семантических сетей. Данные сети описываются атрибутными грамматиками.

Атрибутная грамматика $AG = (G,\ A_S,\ A_I,\ R)$ представляет собой контекстно-свободную грамматику $G = (V_T,\ V_N,\ P,\ S)$, в которой с каждым терминальным и нетерминальным символом связано конечное множество наследуемых A_I и синтезируемых A_S атрибутов. Атрибуты связаны между собой конечным множеством семантических правил R.

Как указано, например, в [3] формализм атрибутных грамматик является удобным средством для описания семантики языков программирования, но реализация вычислителей на атрибутных грамматиках сталкивается с большими трудностями. К таким трудностям можно отнести несоответствие чисто функциональной природы атрибутного вычислителя и связанной с ней неупорядоченностью процесса вычисления атрибутов и упорядоченностью элементов программы.

Авторы предлагают в качестве модели представления знаний в системах концептуального программирования вместо атрибутных грамматик использовать функциональные грамматики. Функциональные грамматики впервые были введены профессором Тузовым В. А. [4].

Функциональная грамматика $FG = (V_T,\ V_N\ ,P,\ F,\ S)$ представляет собой контекстно-свободную грамматику, каждой продукции которой поставлена в соответствие одна функция из множества F. Продукции из множества P имеют вид: $A_i \to \varphi_i\{f_i\}$, где $A_i \in V_N$, $\varphi_i \in (V_N \cup V_T)^*$, $f_i \in F$.

Формально функциональная грамматика является атрибутной грамматикой с единственным синтезируемым атрибутом. Но главное принципиальное отличие заключается в том, что функция f_i не является атрибутом ни нетерминала A_i, ни цепочки φ_i. Функция f_i выражает сущность правила $A_i \to \varphi_i\{f_i\}$. Наследуемые атрибуты в функциональной грамматике отсутствуют.

В [4] приводится сравнительный анализ разных методов описания систем программирования. Анализируются атрибутные (АГ) и функциональные (ФГ) грамматики, а также венский метод (ВМ), грамматики ван Вейнгаардена (ГВ), продукционные системы (ПС) и аксиоматический подход (АП). Результаты анализа представлены в таблице 1. Знаком «+» указана принадлежность элемента методу, знаком «–» – отсутствие принадлежности. Символ «0» указывает на возможность моделирования элемента.

Таблица 1

№	Элемент	ВМ	АГ	ГВ	ПС	АП	ФГ
1	Описание синтаксиса	+	+	+	+	–	+
2	Операция *eval*	–	–	–	–	–	+
3	Описание множеств N_i	0	–	+	–	–	+
4	Операция отождествления	–	–	+	–	–	+
5	Память с варьируемой структурой	–	–	–	–	–	+
6	Память с постоянной структурой	+	0	0	+	–	+
7	Рекурсия	+	0	0	+	–	+
8	Операция *A/B*	+	–	–	+	–	+
9	Управляющая память	–	–	–	–	–	+
10	Система приведений	–	–	–	–	0	+
11	Многоуровневость	–	–	–	–	–	+

Согласно указанному анализу, аппарат функциональных грамматик дает самое полное описание систем программирования. Вообще, все виды грамматик изоморфно вкладываются в функциональную грамматику, т.е. функциональные грамматики являются обобщением всех методов, указанных выше, включая атрибутные грамматики, применяемые в концептуальном программировании. Таким образом, использование функциональных грамматик позволяет получить более гибкую систему, чем при использовании атрибутных грамматик.

Второе преимущество функциональных грамматик заключается в универсальности результата решения задачи.

Результат работы любой системы знаний относится к одному из четырёх видов ответов [2]:

1) логический ответ – система подтверждает или опровергает какую-либо информацию об объекте;

2) фактографический ответ – система выдает решение задачи в виде определенного факта об объекте;

3) процедурный ответ – система синтезирует программу для ЭВМ, способную решать поставленную задачу;

4) понятийный ответ – система выдает определенный закон, по которому можно решить задачу.

При использовании аппарата функциональных грамматик результат решения задачи представляет собой суперпозицию функций $\sigma = \sigma\{f_i\}$. Из неё можно получить любой из перечисленных ответов. Согласно [4], суперпозиция функций может иметь один из трёх типов:

1) тип «знач» – выполненная суперпозиция функций, представляющая собой определенное значение;

2) тип «функ» – невыполненная, но активная суперпозиция функций, являющая функциональной программой;

3) тип «текст» – невыполненная суперпозиция функций, представляющая собой обычный текст.

Изначально, построенная суперпозиция имеет тип «текст», например:

f1(f2(a, f3(b, c), d), e, f4(g, h))

Такой результат работы системы соответствует понятийному ответу. Его суть – закон решения рассматриваемой задачи.

Если вместо аргументов a, b, c, d, e, g, h подать исходные данные задачи, то за счет выполнения функций f1, f2, f3, f4, описанных в базе знаний, будет получено значение функции. Тогда суперпозиция будет иметь тип «знач». Это соответствует фактографическому и логическому ответам системы знаний. Конкретный вид ответа зависит от типа использованных функций.

Если же функции базы знаний f1, f2, f3, f4 транслировать в функции на интересующем языке программирования, то суперпозиция получит тип «функ». Система знаний в этом случае будет выдавать процедурный ответ, т.е. реализовывать автоматический синтез программ.

Таким образом, для реализации системы концептуального программирования, осуществляющей синтез программ на основе функциональных грамматик, необходимо:

1) построить решатель задач, состоящий из базы знаний и механизма решения, моделированных аппаратом функциональных грамматик;

2) дополнить базу знаний функциями на необходимых языках программирования;

3) в механизм вывода решения встроить специальный модуль, осуществляющий переход от суперпозиции функций к программе, т.е. переход от типа «текст» к типу «функ».

Любой современный язык программирования содержит такой инструмент, как функции, поэтому задача построения модуля перехода не представляется очень сложной. Причем подобный модуль нужно создать один

раз для выбранного языка программирования.

Более сложной задачей является дополнение базы знаний программными функциями. Тем более, что каждая новая база знаний для определенной предметной области должна содержать свой набор таких функций. Однако подобная задача является неизбежной и относится к проблеме извлечения знаний.

Таким образом, при описании знаний в концептуальном программировании можно перейти от атрибутных грамматик к функциональным. Это, во-первых, позволяет более адекватно описать и эффективнее реализовать базу знаний и механизм вывода решателя, благодаря наличию свойств, описанных в таблице 1. Во-вторых, результат работы системы концептуального программирования представляет собой не просто код программы на одном языке программирования, а универсальный ответ в виде $\sigma = \sigma\{f_i\}$, который можно либо напрямую вычислить, либо перевести в любой заранее выбранный язык программирования.

3. Пример

Рассмотрим отличие результата работы системы концептуального программирования на основе функциональных грамматик от классической системы, использующей атрибутные грамматики.

В качестве примера была выбрана элементарная задача, описанная в классической работе по концептуальному программированию [1].

Предметная область задана соотношениями:

$$X + Y = 2,5; \qquad Z = X - Y;$$

$$W = Z + X \cdot Y; \qquad X = 0,5.$$

Необходимо построить программу, вычисляющую W.

В [1] на языке УТОПИСТ условие задачи имеет вид:

```
программа ВТОРАЯ;
пусть
    X, Y, Z, W: вещ;
    урав X+Y=2.5;
    урав Z=X-Y;
    урав W=Z+X*Y;
    урав X=0.5;
действия
    на ВТОРАЯ вычислить W;
конец
```

Результат, выдаваемой системой, представляет синтезированный код на языке УТОПИСТ:

```
X:=0.5;
Y:=2.5-X;
Z:=X-Y;
W:=Z+X*Y.
```

Теперь рассмотрим, как выглядит поиск решения и результат в системе, основанной на функциональных грамматиках.

База знаний «ВТОРАЯ» в виде функциональной грамматики:

$FG = (V_T,\ V_N\ , P,\ F,\ S);$

$V_T = \{X\};$

$V_N = \{Y,\ Z,\ W\};$

$$
\begin{aligned}
P = \{ \quad & X \to Y \ \{f_1\}; \\
& X \to ZY \ \{f_2\}; \\
& X \to WZY \ \{f_6\}; \\
& Y \to X \ \{f_1\}; \\
& Y \to ZX \ \{f_3\}; \\
& Y \to WZX \ \{f_6\}; \\
& Z \to XY \ \{f_3\}; \\
& Z \to WXY \ \{f_5\}; \\
& W \to ZXY \ \{f_4\}; \quad \};
\end{aligned}
$$

$$
\begin{aligned}
F = \{ \quad & f_1(x) = 2,5 - x; \\
& f_2(x,y) = x + y; \\
& f_3(x,y) = x - y; \\
& f_4(x,y,z) = x + y \cdot z; \\
& f_5(x,y,z) = x - y \cdot z; \\
& f_6(x,y,z) = (x - y)/z; \quad \};
\end{aligned}
$$

$S = \{W\}.$

Поиск решения осуществляется в разработанном ранее решателе на языке ЛИСП [5]. Для этого база знаний *FG* формируется и загружается в решатель по правилам, указанным в [6].

Команда (Решение ‘(W X) ВТОРАЯ) запускает процесс решения задачи. На рисунке 2 представлен алгоритм поиска решения в виде дерева.

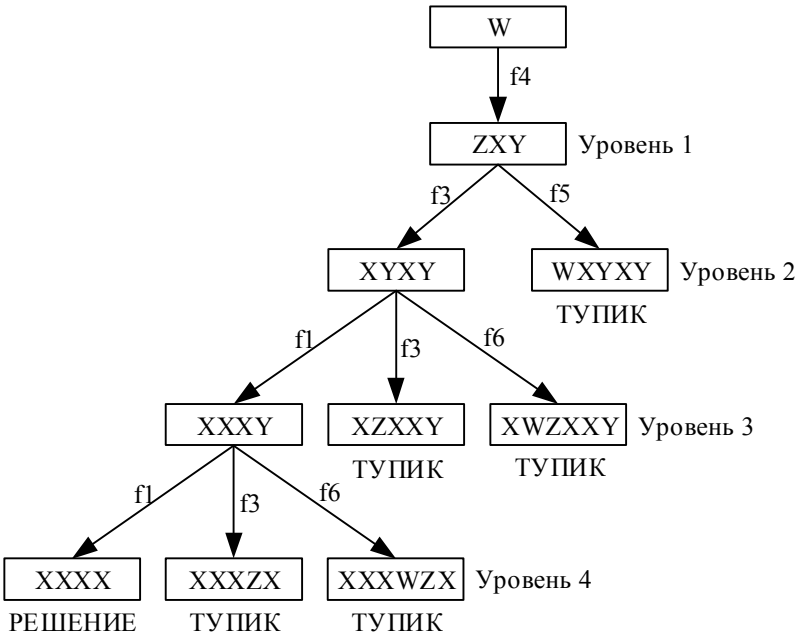

Рис. 2

Таким образом, решение представляет ветвь:

$$W \xrightarrow{f_4} ZXY \xrightarrow{f_3} XYXY \xrightarrow{f_1} XXXY \xrightarrow{f_1} XXXX$$

Данному результату соответствует суперпозиция функций:

$$W = f_4(\ \ (f_3(X,\ f_1(X)),\ X,\ f_1(X)\ \ ).$$

Такое решение является более предпочтительным, чем код, выдаваемый на языке УТОПИСТ. Суперпозиция функций может принять форму кода на любом языке программирования, в котором существует аппарат функций.

Так, например, если необходима программа на языке Си, то нужно закодировать функции множества F следующим образом:

```
float f1(float x) {
    return (2.5-x);
};
float f3(float x, float y) {
    return (x-y);
};
float f4(float x, float y, float z) {
    return (x+y*z);
};
```

Тогда программа на языке Си будет иметь вид:

```
int main {
    float W;
    W = f4(  (f3 (0.5,  f1(0.5) ),  0.5,  f1(0.5)  )
```

```
        return(0);
}
```

Аналогичным образом можно закодировать функции на другом языке программирования и записать программу в виде суперпозиции этих функций.

Заключение

Использование функциональных грамматик в концептуальном программировании является более универсальным, чем использование атрибутных грамматик.

Во-первых, функциональные грамматики более адекватно описывают базу знаний и модель вывода решения, что при правильном использовании может увеличить эффективность поиска решения задачи.

Во-вторых, результат концептуального программирования в виде суперпозиции функций позволяет находить решение задачи как непосредственно (логический или фактографический ответы, тип суперпозиции «знач»), так и в виде синтезированной программы для ЭВМ на любом выбранном языке программирования (процедурный ответ, тип «функ»). Кроме того, при добавлении лингвистического модуля решение задачи может быть представлено в виде текстовых рекомендаций для пользователя (понятийный ответ, тип суперпозиции «текст»).

Литература

1. Тыугу Э. Х. Концептуальное программирование. — М.: Наука. Главная редакция физико-математической литературы, 1984. — 256 с.

2. Новиков Ф. А. Искусственный интеллект: представление знаний и методы поиска решения. — СПб.: Изд-во Политехн. ун-та, 2010. — 240 с.

3. Серебряков В. А., Галочкин М. П. Основы конструирования компиляторов. — М.: Изд-во URSS, 2001. — 174 с.

4. Тузов В. А. Математическая модель языка. — Л.: Изд-во Ленингр. ун-та, 1984. — 176 с.

5. Кравченко В. А., Могнонов П. Б., Чимитов Д. Н. Программная реализация решателя задач на основе метода функциональных грамматик // Вестник ВСГУТУ. — Улан-Удэ: Изд-во ВСГУТУ, 2013. — №6. — С. 36 – 42.

6. Кравченко В. А., Ширапов Д. Ш. построение баз знаний для решения задач методом функциональных грамматик // Вестник Бурятского государственного университета. — Улан-Удэ: Изд-во Бурятского госуниверситета, 2015. — №9. — С. 96 – 102.

References

1. Tyugu Je. H. Konceptual'noe programmirovanie. — M.: Nauka. Glavnaja redakcija fiziko-matematicheskoj literatury, 1984. — 256 s.

2. Novikov F. A. Iskusstvennyj intellekt: predstavlenie znanij i metody

poiska reshenija. — SPb.: Izd-vo Politehn. un-ta, 2010. — 240 s.

3. Serebrjakov V. A., Galochkin M. P. Osnovy konstruirovanija kompiljatorov. — M.: Izd-vo URSS, 2001. — 174 s.

4. Tuzov V. A. Matematicheskaja model' jazyka. — L.: Izd-vo Leningr. un-ta, 1984. — 176 s.

5. Kravchenko V. A., Mognonov P. B., Chimitov D. N. Programmnaja realizacija reshatelja zadach na osnove metoda funkcional'nyh grammatik // Vestnik VSGUTU. — Ulan-Udje: Izd-vo VSGUTU, 2013. — №6. — S. 36 –42.

6. Kravchenko V. A., Shirapov D. Sh. postroenie baz znanij dlja reshenija zadach metodom funkcional'nyh grammatik // Vestnik Burjatskogo gosudarstvennogo universiteta. — Ulan-Udje: Izd-vo Burjatskogo gosuniversiteta, 2015. — №9. — S. 96 – 102.

Кравченко Вячеслав Александрович, старший преподаватель кафедры «Электронные вычислительные системы» Восточно-Сибирского государственного университета технологий и управления», e-mail: krawyach@mail.ru.

Ширапов Дашадондок Шагдарович, доктор физико-математических наук, профессор, заведующий кафедрой «Электронные вычислительные системы» Восточно-Сибирского государственного университета технологий и управления», e-mail: shirapov.dashadondok@yandex.ru.

Олзоева Сэсэг Ивановна, доктор технических наук, профессор кафедры «Электронные вычислительные системы» Восточно-Сибирского государственного университета технологий и управления», e-mail: sseseg@yandex.ru.

Kravchenko Vyacheslav Aleksandrovich, senior lecturer, Department «Electronic systems» of the East Siberian State University of Technology and Management.

Shirapov Dashadondok Shagdarovich, DSc, professor, head of Department «Electronic systems» of the East Siberian State University of Technology and Management.

Olzoeva Seseg Ivanovna, DSc, professor, Department «Electronic systems» of the East Siberian State University of Technology and Management.