УДК 681.3.06

Т. В. Грищук, к. т. н., доц.; М. М. Биков, к. т. н., проф.

МОДЕЛЮВАННЯ ДИСКУРСУ В СИСТЕМАХ З ГОЛОСОВИМИ ІНТЕРФЕЙСАМИ

Представлено модифікований синтаксис запису КВ-граматик, показано можливість застосування цього синтаксису для отримання інтерпретованого коду вхідних команд.

***Ключові слова**: КВ-граматики, дерева виводу, розпізнавання мови, голосовий інтерфейс.*

Для людини мова є природною формою спілкування. Тому реалізація людино-машинного інтерфейсу в сучасних технічних системах на основі мовного вводу-виводу – перспективний напрямок розвитку інтелектуальних систем керування.

Обробка природної мови – це формулювання та дослідження комп'ютерно-ефективних механізмів для забезпечення комунікації з ЕОМ природною мовою. Об'єктами дослідження є:

- власне природні мови;
- використання природної мови як в комунікації між людьми, так і в комунікації людини з ЕОМ.

Завдання досліджень – створення комп'ютерно-ефективних моделей комунікації людини з ЕОМ. Саме така постановка задачі відрізняє обробку природної мови (Natural Language Processing, NLP) від задач традиційної лінгвістики та інших дисциплін, що вивчають природні мови, і дозволяє віднести її до області штучного інтелекту [1].

Розрізняють загальну і прикладну NLP. Завдання загальної NLP є розробка моделей використання мови людиною. Прикладна NLP займається звичайно не моделюванням, а безпосередньо можливістю комунікації людини з ЕОМ природною мовою.

Більшість систем розпізнавання мови (СРМ) призначені для вирішення конкретних завдань керування (приладами, процесами тощо), тому їхній граматичний рівень передбачає формалізацію природної мови. В загальному випадку достатнім є формальний опис дискурсу за допомогою контекстно-вільних (КВ) граматик. Найпоширенішим синтаксисом запису КВ-граматик є форми Бекуса-Наура (БНФ) [2].

Загальноприйнятим методом представлення виводу деякого непорожнього ланцюжка вважається дерево виводу (іноді його називають деревом синтаксичного розбору). Розглянемо просте правило для виводу одного речення: *Main (GoCommand (GoVerb ("Go"), HomeDir ("home")))*. Дерево виводу для цього правила представлено на рис.1.

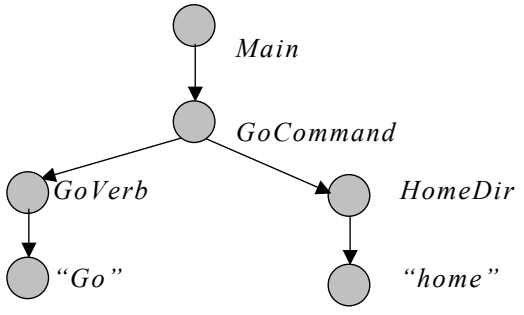

Рис. 1. Приклад дерева виводу

Алгоритмічне та математичне забезпечення програмного модуля голосового інтерфейсу повинно розв'язувати дві основні задачі. По-перше, необхідно забезпечити таку організацію

фраз вхідної граматики, яка дозволила б відмовитися від збереження самих правил. По-друге, синтаксис КВ-граматики й математична модель організації фраз КВ-граматики повинні забезпечувати можливість перетворення будь-якої вхідної фрази в інтерпретований програмний код з можливістю застосування структурованого або об'єктно орієнтованого підходів.

У цій статті представлено новий спосіб запису КВ-граматик, на основі якого можна побудувати граф граматики. Вивід фраз граматики відбувається шляхом обходу графу в ширину, що дає можливість паралельного виводу всіх фраз граматики для збільшення швидкості розпізнавання [3].

Вершинами цього графу є термінали, а дуги – це правила граматики. Основним недоліком такого підходу є те, що ми втрачаємо інформацію про розміщення нетермінальних символів, а це потрібно для контролю процесу розпізнавання при обході усіх граматичних ланцюгів на представленому графі. Нетермінальні символи є точками розгалуження. Оскільки один і той же термінал може виступати в граматиці як окремо, так і в якості частини нетерміналу, то під час генерування ланцюгів на графі виникає потреба в складному граматичному контролі. Покажемо це на прикладі випадку дискурсу, що описує процес форматування речення.

У цьому прикладі ми описуємо ситуацію, коли користувач може змінити стиль написаного ним речення на "напівжирний" (*bold*) або на "курсив" (*italic*).

Запишемо граматику в БНФ:

Grammar "Format"

< main > : make < object > < style > |

bold it;

< object > : it | sentence;

< style > : bold | italic;

Ця граматика описується через 5 термінальних символів (*make, it, sentence, bold, italic*) і 2 нетермінальні (*object*, *style*).

Дискурс у цьому випадку описується таким графом:

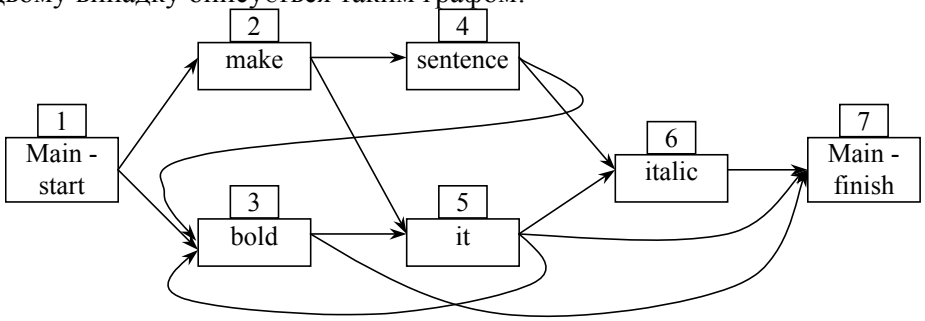

Рис. 2. Граф граматики "Format"

У цьому випадку виникає необхідність виконувати граматичний контроль граматичного ланцюжка після виходу з кожного термінала. Так, виходячи з рис. 2, для представленого графа можливими є циклічні повторення фрази "*bold it*", що, звичайно, заборонено граматикою.

Наявні методи граматичного виводу характеризуються також складністю за різними типами невизначеності, головною з яких є синтаксична невизначеність, коли один і той же символ входить одночасно в декілька правил граматики. Для усунення такої невизначеності необхідно вводити додаткові операції контролю синтаксису.

Для того, щоб спростити процедуру граматичного контролю, в роботі пропонується змінити вигляд графа граматики шляхом внесення додаткових вершин, що будуть позначати входи і виходи з нетермінальних символів. Так, кожне правило типу $< S >= a < b >$ буде переведено у форму $< s\, a < b\, b > s >$.

На рис. 3 показано модифікований граф граматики. Нескладний візуальній аналіз цього графа засвідчує, що зараз немає необхідності в додаткових посиланнях на правила граматики.

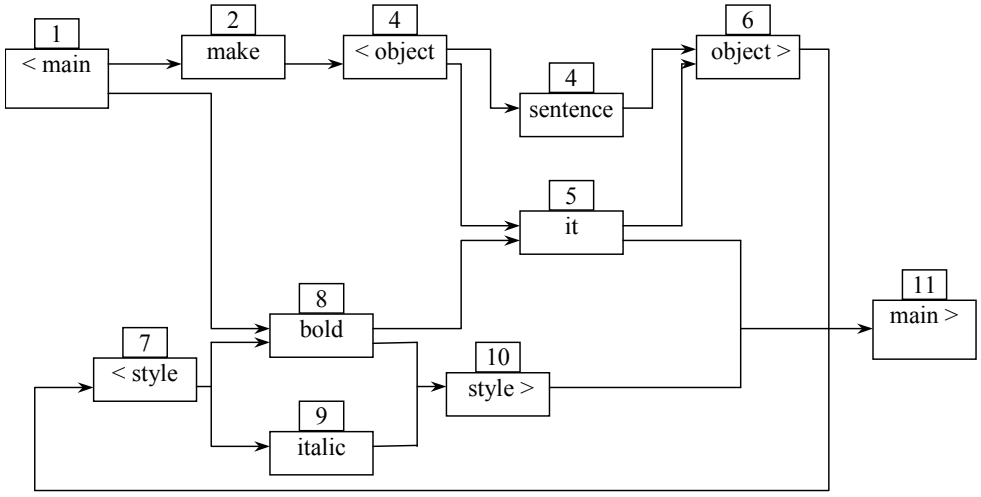

Рис. 3. Граф граматики в модифікованому синтаксисі

Головний принцип дії голосового інтерфейсу – це перетворення вхідної команди користувача, що була виконана природною мовою, у програмний код мовою інтерпретатора, який одразу (в інтерактивному режимі) буде виконано. Як можна бачити з наведеного вище прикладу, класична форма запису дозволяє просто описати вхідні команди користувача, але потребує використання додаткових засобів для перетворення вхідної команди у вихідний програмний код. Для виконання потрібного перетворення необхідно на основі початкового дерева виводу отримати таке дерево виводу, на гілках якого знаходилися б термінальні символи, які фактично були б частиною програмного коду, що може бути інтерпретованим.

Класичний підхід має такі переваги:

– опис вхідної частини відокремлено від частини побудови вихідної граматики;
– покрокове формування вихідного дерева дозволяє відслідковувати процес формування, що допомагає під час редагування граматики;
– наявність можливості знаходження некоректних або небажаних команд.

Недоліками підходу є:

– під час опису правил перетворення необхідно чітко уявляти собі структуру дерева та усвідомлювати вплив кожного кроку на результуюче дерево;
– складність редагування проявляється в тому, що внесення змін на одному рівні викликає потребу змінювати наступні рівні правил.

Другою перевагою запропонованого дужкового способу запису граматик є те, що на його основі можна будувати графи атрибутивних граматик. У цьому випадку з вершинами графу асоціюються елементи вихідної мови (скрипта, який в кінцевому рахунку виконає команду). Такий спосіб запису дає можливість реалізувати в процесі розробки складних граматичних конструкцій принцип структурного програмування. Так, на верхніх рівнях правил граматики можна задавати загальну структуру виконуючого стрипта, а на нижчих рівнях граматики – безпосередньо отримувати фактичні параметри.

Розглянемо приклад атрибутивної граматики. Ця граматика описує чотири фрази, кожна з яких транслюється в деякий умовний скрипт.

```
<main(script: my_color, my_object)>:
        draw <color(my_color)> <object(my_object)>
                {script = "Draw (" + my_object + "," + my_color ");";};
        <color(color)>:
                red     {color = "red";} |
```

```
        green {color = “green”;};
<object(object)>:
        rectangle {object = “rectangle”;} |
        circle    {object = “circle”;}
```

У дужковому синтаксисі ця граматика має вигляд:

<main draw <color color> <object object> {α_1} main>
<color red {α_2} color>
<color green {α_3} color>
<object rectangle {α_4} object>
<object circle {α_5} object>,

де:

{α_1} = {script = “Draw (” + my_object + “,” + my_color “);”;}
{α_2} = {color = “red”;}
{α_3} = {color = “green”;}
{α_4} = {object = “rectangle”;}
{α_5} = {object = “circle”;}.

На рис. 4 наведено граф граматики, яка розглядається.

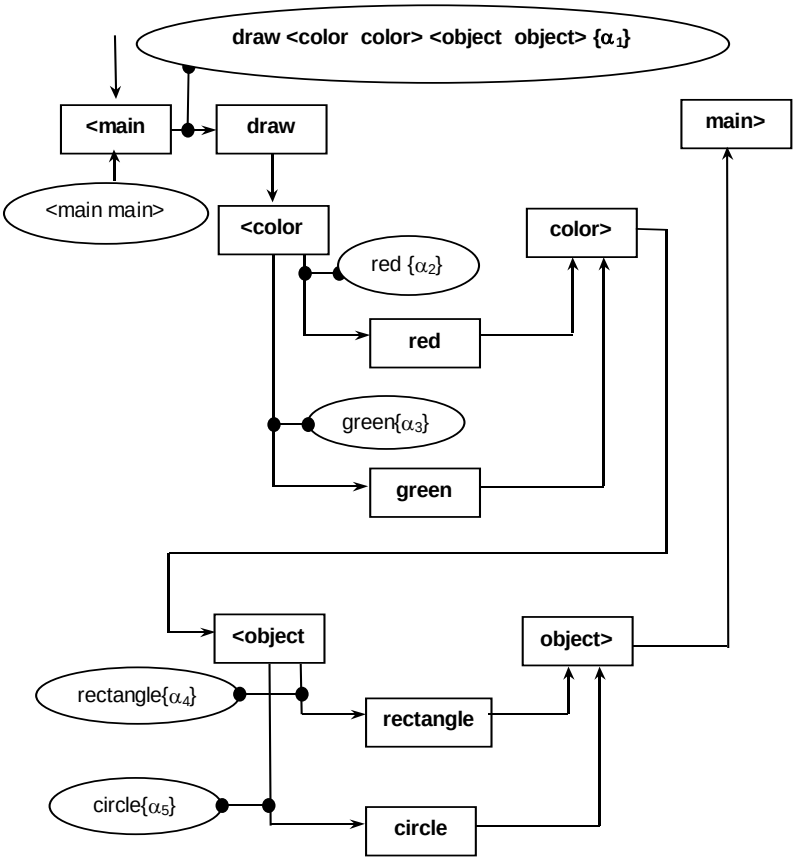

Рис. 4. Приклад графу для атрибутивної граматики

У результаті генерації фрази на граматичній мережі існує можливість отримати не тільки саму фразу, але й її вивід у дужковій формі. Наприклад, для фрази

draw red rectangle

отримуємо вивід:

<main draw <color red {α_2} color> <object rectangle {α_4} object> {α_1} main>.

Інтерпретація дужкового запису виводу призводить до генерації коду внутрішньою мовою

(Intermediate Translation Language – ITL). Для фрази, що розглядається, отримаємо ITL код, який на високому рівні деталізації еквівалентний такому:

```
Entry point()
{
        Declare translation;
        Main(translation);
}
Main(script)
{
        Declare my_color, my_object;
        Color(my_color);
        Object(my_object);
        script = “Draw (” + my_object + “,” + my_color “);”
}
Color(color)
{
        color = “red”;
}
Object(object)
{
oject = “rectangle”;
}
```

Інтерпретація ITL коду, який згенеровано на граматичній мережі, дає такий результат трансляції: Draw (rectangle, red);.

Висновки

Отже, розглянутий спосіб запису КВ-граматик дає можливість не лише отримати наочний вигляд граматики у вигляді орієнтованого навантаженого графу, але й виконувати паралельний вивід фраз граматики без додаткового граматичного контролю для усунення граматичної невизначеності. Ще одною перевагою цього підходу є можливість отримувати інтерпретований програмний код на етапі виводу вхідної команди, використовуючи основні засади структурного програмування.

СПИСОК ЛІТЕРАТУРИ

1. Encyclopaedia of Artificial Intelligence. Entry Natural Language Understanding. – P. 660 – 677. – Режим доступу: http://sabia.tic.udc.es/encyclopediaAI/.

2. Льюис Ф. Теоретические основы проектирования компиляторов / Льюис Ф., Розенкранц Д., Стирнз Р. – М.: Мир, 1979. – 654 с.

3. Грищук Т. В. Розпізнавання природної мови на граматичних марковських мережах / Т.В. Грищук // Наукові праці Донецького національного технічного університету. Серія: “Обчислювальна техніка та автоматика”. – Донецьк: ДонНТУ, 2005. – С. 181 – 187.

Грищук Тетяна Вікторівна **–** к. т. н., доцент кафедри комп'ютерних систем управління, тел.: (0432)-598222, thryshuk@mail.ru.

Биков Микола Максимович **–** к. т. н., професор кафедри комп'ютерних систем управління.
Вінницький національний технічний університет.