

الجلسة الثالثة: المعرب

الهدف من الجلسة

التعريف بأداة المسح yacc من مكتبة ply.

بناء برنامج يحاكي عمل المعرب.

أداة yacc

تقوم هذه الأداة بتنفيذ التحليل القواعدي للبرنامج المصدري حيث تولد سلسلة القواعد التي تتبعها اللغة التي يقوم المترجم بترجمتها.

عند إجراء عملية الإعراب، يُعبّر عن النص بسلسلة من الأجزاء المنتهية terminal والأجزاء غير المنتهية non-terminal، مثال:

لإعراب عبارة رياضية، يمكن كتابة القواعد التالية:

```
expression : expression + term
            | expression - term
            | term

term       : term * factor
            | term / factor
            | factor

factor    : NUMBER
            | ( expression )
```

رسم توضيحي 1 قواعد تعبر عن عمليات بسيطة

الاجزاء المنتهية هي:

NUMBER - + * / وهي تطابق للرموز tokens نحصل عليها من الدخل المصدري.

الاجزاء غير المنتهية هي:

expression term factor

هذه الرموز مكونة من مجموعة من اجزاء منتهية و قواعد أخرى.

الترجمة المباشرة للنص syntax directed translation:

الطريقة المعنوية لوصف كود النص، فكل قاعدة يعبر عنها مجموعة من الخصائص حيث ترتبط كل خاصية برمز ما و action معين.

عند التعرف على قاعدة ما، يتم تنفيذ ال action الخاص به.

مثال آلة حاسبة بسيطة:

Grammar	Action
expression0 : expression1 + term expression1 - term term	expression0.val = expression1.val + term.val expression0.val = expression1.val - term.val expression0.val = term.val
term0 : term1 * factor term1 / factor factor	term0.val = term1.val * factor.val term0.val = term1.val / factor.val term0.val = factor.val
factor : NUMBER (expression)	factor.val = int(NUMBER.lexval) factor.val = expression.val

رسم توضيحي 2 الأفعال الخاصة بكل قاعدة

تتبع أداة yacc طريقة LALR في الإعراب أي الـ bottom-up حيث يحاول التعرف على الجانب الأيمن من قواعد النحو المختلفة. عند العثور على جانب أيمن صحيح في المُدخلات، يُفَعَّل رمز الإجراء المناسب، ويستبدل رموز النحو بالرمز الموجود على الجانب الأيسر.

Step	Symbol	Stack	Input Tokens	Action
1			3 + 5 * (10 - 20)\$	Shift 3
2	3		+ 5 * (10 - 20)\$	Reduce factor : NUMBER
3	factor		+ 5 * (10 - 20)\$	Reduce term : factor
4	term		+ 5 * (10 - 20)\$	Reduce expr : term
5	expr		+ 5 * (10 - 20)\$	Shift +
6	expr +		5 * (10 - 20)\$	Shift 5
7	expr + 5		* (10 - 20)\$	Reduce factor : NUMBER
8	expr + factor		* (10 - 20)\$	Reduce term : factor
9	expr + term		* (10 - 20)\$	Shift *
10	expr + term *		(10 - 20)\$	Shift (
11	expr + term * (10 - 20)\$	Shift 10
12	expr + term * (10		- 20)\$	Reduce factor : NUMBER
13	expr + term * (factor		- 20)\$	Reduce term : factor
14	expr + term * (term		- 20)\$	Reduce expr : term
15	expr + term * (expr		- 20)\$	Shift -
16	expr + term * (expr -		20)\$	Shift 20
17	expr + term * (expr - 20)\$	Reduce factor : NUMBER
18	expr + term * (expr - factor)\$	Reduce term : factor
19	expr + term * (expr - term)\$	Reduce expr : expr - term
20	expr + term * (expr)\$	Shift)
21	expr + term * (expr)		\$	Reduce factor : (expr)
22	expr + term * factor		\$	Reduce term : term * factor
23	expr + term		\$	Reduce expr : expr + term
24	expr		\$	Reduce expr
25			\$	Success!

رسم توضيحي 3 عمليات shift-reduce

مثال متكامل معرب يعبر عن آلة حاسبة بسيطة مع العمليات الأساسية:

يوجد في الآلة الحاسبة العديد من العمليات الحسابية ولكل عملية قاعدة خاصة بها، فمثلاً القاعدة التي تعبر عن عملية الجمع هي:

expression : expression PLUS expression

تابع البايثون في ملف المعرب الذي يعبر عن عملية الجمع هو:

```
def p_expression_plus(p):  
    'expression : expression PLUS expression'  
    p[0] = p[1] + p[3]  
    print("PLUS Op.")
```

رسم توضيحي 4 التابع الذي يعبر عن قاعدة الجمع

القواعد التي تعبر عن آلة حاسبة بسيطة :

```
Rule 0    S' -> input  
Rule 1    input -> input line  
Rule 2    input -> line  
Rule 3    line -> expression EQUAL  
Rule 4    expression -> NUMBER  
Rule 5    expression -> expression PLUS expression  
Rule 6    expression -> expression MINUS expression  
Rule 7    expression -> expression MULT expression  
Rule 8    expression -> expression DIVS expression  
Rule 9    expression -> expression POWER expression  
Rule 10   expression -> MINUS expression  
Rule 11   expression -> PARENTHESISL expression PARENTHESISR
```

رسم توضيحي 5 القواعد التي تعبر عن آلة حاسبة

```

precedence = [
    ('left', 'PLUS', 'MINUS'),
    ('left', 'MULT', 'DIVS'),
    ('left', 'NEG'),
    ('right', 'POWER'),
]

def p_input(p):
    '''input : input line
    |         | line'''
    pass

def p_line(p):
    'line : expression EQUAL'
    print(f"The result is = {p[1]}")

def p_expression_number(p):
    'expression : NUMBER'
    p[0] = p[1]
    print(f"Number {p[1]}")

def p_expression_plus(p):
    'expression : expression PLUS expression'
    p[0] = p[1] + p[3]
    print("PLUS Op.")

def p_expression_minus(p):
    'expression : expression MINUS expression'
    p[0] = p[1] - p[3]
    print("MINUS Op.")

def p_expression_mult(p):
    'expression : expression MULT expression'
    p[0] = p[1] * p[3]
    print("MULT Op.")

def p_expression_div(p):
    'expression : expression DIVS expression'
    if p[3] == 0:
        print("Error: Division by zero")
        p[0] = 0
    else:
        p[0] = p[1] / p[3]
        print("DIV Op.")

def p_expression_power(p):
    'expression : expression POWER expression'
    p[0] = math.pow(p[1], p[3])
    print("Power Op.")

def p_expression_negative(p):
    'expression : MINUS expression %prec NEG'
    p[0] = -p[2]
    print("Negative Op.")

```

رسم توضيحي 6 كود المعرب الموافق للقواعد السابقة

ملاحظة:

1. تقدم الأداة yacc من مكتبة ply القواعد التي تعبر عن كود المعرب إضافة لكل حالة (shift+reduce) يمر بها المعرب أثناء عمله.
2. عند تنفيذ ملف المعرب بشكل صحيح يظهر ملف بلاهة out. وهو يمثل هذه القواعد.

تمارين:

- عدل كود المترجم ليقوم بإعراب قاعدة تعبر عن باقي القسمة.
- عدل كود المترجم ليقوم بإعراب قاعدة تعبر عن باعملية الـ AND المنطقية مع مراعاة المقسوم عليه.

انتهت الجلسة

إعداد: م. ريم جبيلي