Compilers Techniques
Lecture 7
المحاضرة السابعة

Syntax Analyser – Top-down parser -

المحلل القواعدي – الإعراب من الأعلى للأسفل -

Reference: Aho2 Sections 3.6-3.7; Aho1 pp. 113-125; Grune 2.1.6.1-2.1.6.6 (different style); Hunter 3.3 (very condensed); Cooper1 2.4-2.4.3

السنة الرابعة – المستوى السابع - الهندسة المعلوماتية

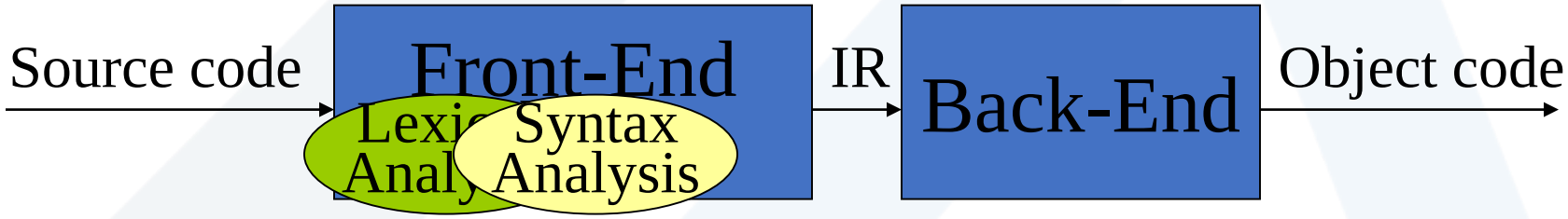
Lectures Topics

الإعراب Parsing:

- يعبر عن الإعراب حر السياق بالقواعد حرة السياق.
- وهي عملية إكتشاف الاشتقاق لبعض الجمل.

محاضرة اليوم:

الإعراب من الأعلى إلى الأسفل Top-down parsing



المحلل القواعدي Syntax Analysis

- **تعريف:** نقول عن القاعدة أنها عودية يسارية إذا كانت تحتوي على رمز غير تحديدي *A*، *اشتقاقه من الشكل* $A \Rightarrow Aa$، *لسلسلة ما* a .
- يمكن أن تسبب القواعد العودية اليسارية دخول المعرب التراجعي التدرجي في حلقة لانهائية.
- قد يكفي استبدال $A \rightarrow Aa \mid b$
  بـ $A \rightarrow bA'$ و $A' \rightarrow aA' \mid \varepsilon$
  للتخلص من العودية اليسارية.
- **مثال:**

$Sum \rightarrow Sum+number \mid number$

would become: $Sum \rightarrow number\ Sum'$

$Sum' \rightarrow +number\ Sum' \mid \varepsilon$

إلغاء العودية اليسارية

بتطبيق تحويل القواعد في المثال في الشريحة الثانية نحصل على:

1.	*Goal* → *Expr*	5. *Term* → *Term* * *Factor*	
2.	*Expr* → *Expr* + *Term*	6.	| *Term* / *Factor*
3.	| *Expr* – *Term*	7.	| *Factor*
4.	| *Term*	8. *Factor* → *number*	
		9.	| *id*

Expr → *Term Expr'*
Expr' → *+Term Expr'* | – *Term Expr'* | ε
Term → *Factor Term'*
Term' → **Factor Term'* | */ Factor Term'* | ε
(*Goal* → *Expr* and *Factor* → *number* | *id* remain unchanged)

الخوارزمية العامة: تعمل من أجل القواعد غير الدورية وبدون ε-productions.

1-ترتيب الرموز غير التحديدية: $A_1, A_2, A_3, \ldots, A_n$

2. For i=1 to n do
for j=1 to i-1 do
I) replace each production of the form $A_i \rightarrow A_j\gamma$ *with*
the productions $A_i \rightarrow \delta_1\gamma \mid \delta_2\gamma \mid \ldots \mid \delta_k\gamma$
where $A_j \rightarrow \delta_1 \mid \delta_2 \mid \ldots \mid \delta_k$ *are all the current* A_j *productions*
II) eliminate the immediate left recursion among the A_i

مشاكل الإعراب Top-Down حتى الآن

- نستطيع توليد معرب من الأعلى إلى الأدنى. إلا أننا إذا وقعنا على قاعدة خاطئة يتوجب علينا التراجع عكسياً backtrack.

- **الفكرة الأساسية** هي التوقع في إدخال واستخدام السياق من أجل الانتقاء الصحيح.
- ماهو حجم التوقع المطلوب؟
  - عموماً كمية كبيرة عشوائية.
  - لحسن الحظ فإن معظم لغات البرمجة تقوم على بناء صفوف فرعية من القواعد حرة السياق يمكن إعرابها بتوقع محدود.

الإعراب التنبؤي Predictive Parsing

- الفكرة هي بناء معرب بدون الحاجة لعملية Backtrack (فكرة المعرب التنبؤي Predictive Parser) من خلال التنبؤ برمز واحد من النحو.
- **الفكرة الأساسية Basic idea:**
  - لكل production نتاج $A \rightarrow a \mid b$ يجب أن يكون هناك طريقة وحيدة لاختيار النتاج الصحيح الذي سنتوسع منه.
- *1- FIRST* sets:
  - من أجل كل رمز (A) فإن المجموعة FIRST(A) تمثل مجموعة الرموز التحديدية Terrminals التي تظهر كرمز بداية في سلسلة واحدة أو أكثر من السلاسل المشتقة من A.

  مثال: النحو في الشريحة رقم 4:

  $$FIRST(Expr')=\{+,-,\varepsilon\},\ FIRST(Term')=\{*,/,\varepsilon\},\ FIRST(Factor)=\{number,\ id\}$$

- 2- The LL(1) property **(L - left-to-right, L – leftmost, 1 – lookahead)**
  - إذا ظهر كلا النتاجين $A \rightarrow a$ وَ $A \rightarrow b$ في *النحو*، يجب أن يتحقق الشرط:
  - $FIRST(a) \cap FIRST(b) = \varnothing$ والذي يسمح للمعرب أن يختار الاشتقاق الصحيح بعد التنبؤ برمز واحد فقط من النحو.
  - ***النحو في الشريحة رقم 4 يحقق هذه الخاصية!***

الإعراب التنبؤي Predictive Parsing

- الفكرة هي بناء معرب بدون الحاجة لعملية Backtrack (فكرة المعرب التنبؤي Predictive Parser) من خلال التنبؤ برمز واحد من النحو.
- **الفكرة الأساسية Basic idea:**
  - لكل production نتاج $A \rightarrow a \mid b$ يجب أن يكون هناك طريقة وحيدة لاختيار النتاج الصحيح الذي سنتوسع منه.
- *1- FIRST* sets:
  - من أجل كل رمز (A) فإن المجموعة FIRST(A) تمثل مجموعة الرموز التحديدية Terrminals التي تظهر كرمز بداية في سلسلة واحدة أو أكثر من السلاسل المشتقة من A.

  مثال: النحو في الشريحة رقم 4:

  $$FIRST(Expr')=\{+,-,\varepsilon\},\ FIRST(Term')=\{*,/,\varepsilon\},\ FIRST(Factor)=\{number,\ id\}$$

- 2- The LL(1) property **(L - left-to-right, L – leftmost, 1 – lookahead)**
  - إذا ظهر كلا النتاجين $A \rightarrow a$ وَ $A \rightarrow b$ في *النحو*، يجب أن يتحقق الشرط:
  - $FIRST(a) \cap FIRST(b) = \varnothing$ والذي يسمح للمعرب أن يختار الاشتقاق الصحيح بعد التنبؤ برمز واحد فقط من النحو.
  - ***النحو في الشريحة رقم 4 يحقق هذه الخاصية!***

الإعراب التنبؤي التراجعي التدرجي
Recursive Descent Predictive Parsing


```
Main()
  token=next_token();
  if (Expr()==false)
    then <next_compilation_step>
  else return false;

Expr()
  if (Term()==false)
    then result=false
  else if (EPrime()==false)
    then result=false
  else result=true
  return result

EPrime()
  if (token=='+' or '-') then
    token=next_token()
    if (Term()==false)
      then result=false
    elseif (EPrime()==false)
      then result=false
    else result=true
  else result=true /* ε */
  return result

Term()
  if (Factor()==false)
    then result=false
  else if (TPrime()==false)
    then result=false
  else result=true
  return result
```

```
TPrime()
  if (token=='*' or '/') then
    token=next_token()
    if (Factor()==false)
      then result=false
    else if (TPrime()==false)
      then result=false
    else result=true
  else result=true
  return result

Factor()
  if (token=='number' or 'id')then
    token=next_token()
    result=true
  else
    report syntax_error
    result=false
  return result
```

لا داعي للعودة للخلف من أجل التصحيح!

check :-)

القواعد التي تم تصميم المعرب التنبؤي التراجعي المجاور لأجلها

Goal → *Expr*

Expr → *Term Expr'*

Expr' → *+Term Expr'* | *– Term Expr'* | *ε*

Term → *Factor Term'*

Term' → **Factor Term'* | */ Factor Term'* | *ε*

Factor → *number* | *id* remain unchanged)

تعدد النتاج من جهة اليسار
Left Factoring

ماذا لو أن النحو Grammar الخاص بي لا يتضمن خاصية LL(1)؟
في بعض الأحيان يمكن تحويل النحو ليتضمن هذه الخاصية.

الخوارزمية:

1- من أجل كل A («غير تحديدية» *non-terminal*)، أوجد اللاحقة *prefix* الأطول، لتكن a، المشتركة مع اثنين أو أكثر من البدائل.

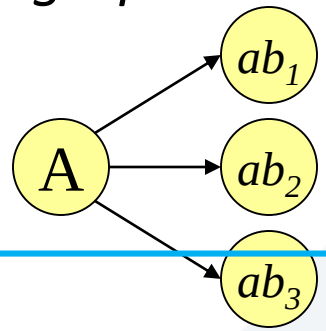
2- *إذا كانت* $a \neq \varepsilon$، *عندها استبدل كل نتاجات* A *من الشكل* $A \rightarrow ab_1 | ab_2 | ab_3 | ... | ab_n | \gamma$ (*حيث أن* γ *هي أي شيء لا يبدأ بالرمز* a) *بالشكل التالي:*

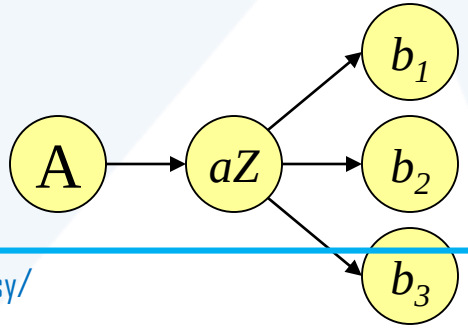
$$A \rightarrow aZ \mid \gamma \quad \text{و} \quad Z \rightarrow b_1 | b_2 | b_3 | ... | b_n$$

كرر ما سبق حتى لا يعود هناك أي لواحق Prefix مشتركة.

Example: $A \rightarrow ab_1 \mid ab_2 \mid ab_3$ would become $A \rightarrow aZ$ and $Z \rightarrow b_1 | b_2 | b_3$

Note the graphical representation:





مثال عن Left Factoring

هذا النحو هو نسخة مختلفة عن النحو الموجود في الشريحة رقم 4.

Goal → *Expr*
Expr* → *Term* + *Expr
| *Term* – *Expr*
| *Term*

Term* → *Factor* * *Term
| *Factor* / *Term*
| *Factor*
Factor → *number*
| *id*

لدينا مشكلة مع القواعد المختلفة لغير التحديدية Expr وكذلك بالنسبة لغير التحديدية Term. في كلتا الحالتين، الرمز الموجود على الطرف اليميني هو ذاته (*Term في حالة* Expr و *Factor في حالة Term*).

الحل: بدايةً

FIRST(Term)=FIRST(Term)∩FIRST(Term)={number, id}.
FIRST(Factor)=FIRST(Factor)∩FIRST(Factor)={number, id}.

النحو لا يحقق الخاصية LL(1) !

Applying left factoring:

Expr → *Term Expr´*
Expr´ → *+ Expr | – Expr | ε*

FIRST(+)={+}; FIRST(–)={–}; FIRST(ε)={ε};
FIRST(–)∩ FIRST(+) ∩ FIRST(ε)= =∅

Term → *Factor Term´*
Term´ → ** Term | / Term | ε*

FIRST(*)={}; FIRST(/)={/}; FIRST(ε)={ε};*
FIRST(*)∩ FIRST(/) ∩ FIRST(ε)= =∅

Conclusion
الخلاصة

- **الإعراب من الأعلى إلى الأسفل:**
  - التكرار مع التراجع العكسي إلى الوراء في حال فشل الإعراب من أجل التصحيح (قليل الاستخدام عملياً).
  - التكرار مع التنبؤ برمز واحد من الدخل دون الحاجة للعودة للتصحيح.
- **الإعراب التنبؤي غير التكراري ممكن أيضاً:** الحفاظ على المكدس بشكل صريح وليس بشكل ضمني من خلال العودية وتحديد النتاج الممكن تطبيقه من خلال جدول. (كتاب آهو الصفحات 186-190).
- إذا كان لدينا نحو سياق حر لا يحقق خاصية LL(1)، لا يمكن عندها أن نقرر فيما إذا كان النحو المحقق لخاصية LL(1) موجود بالفعل أم لا.

مثال عن Left Factoring

1. *Goal* → *Expr*
2. *Expr* → *Term Expr´*
3. *Expr´* → + *Expr*
4. | - *Expr*
5. | ε
6. *Term* → *Factor Term´*
7. *Term´* → * *Term*
8. | / *Term*
9. | ε
10. *Factor* → *number*
11. | *id*

يحدد الرمز التالي كل خيار بشكل صحيح ولا نحتاج للعودة للخلف مرة أخرى للتصحيح

Rule	Sentential Form	Input
-	*Goal*	| x – 2*y
1	*Expr*	| x – 2*y
2	*Term Expr´*	| x – 2*y
6	*Factor Term´ Expr´*	| x – 2*y
11	*id Term´ Expr´*	| x – 2*y
Match	*id Term´ Expr´*	x | – 2*y
9	*id ε Expr´*	x | – 2*y
4	*id – Expr*	x | – 2*y
Match	*id – Expr*	x – | 2*y
2	*id – Term Expr´*	x – | 2*y
6	*id – Factor Term´ Expr´*	x – | 2*y
10	*id – num Term´ Expr´*	x – | 2*y
Match	*id – num Term´ Expr´*	x – 2 | *y
7	*id – num * Term Expr´*	x – 2 | *y
Match	*id – num * Term Expr´*	x – 2* | y
6	*id – num * Factor Term´ Expr´*	x – 2* | y
11	*id – num * id Term´ Expr´*	x – 2* | y
Match	*id – num * id Term´ Expr´*	x – 2*y |
9	*id – num * id Expr´*	x – 2*y |
5	*id – num * id*	x – 2*y |