Compilers Techniques
Lecture 8
المحاضرة الثامنة

Syntax Analyser – Top-down parser -

المحلل القواعدي – الإعراب (1)LR-

Reference: Aho Pages 209-277;

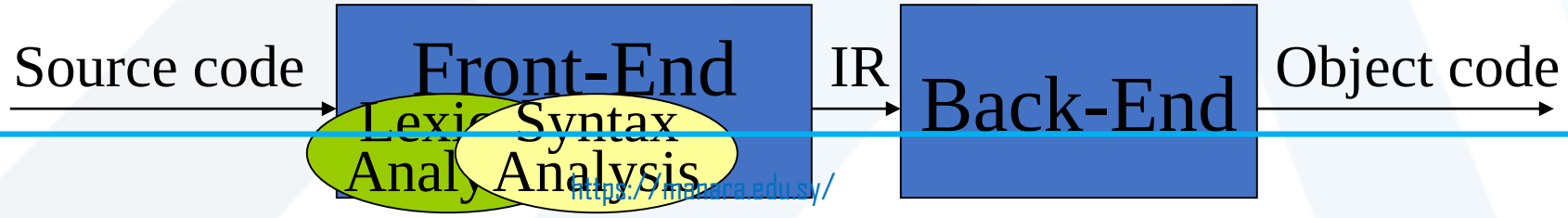
السنة الرابعة – المستوى السابع - الهندسة المعلوماتية

Lectures Topics

من المحاضرة الأخيرة الإعراب من أعلى إلى أدنى :Top-Down Parsing

- نبدأ من جذر الشجرة ونتابع باتجاه الأوراق.
- نعتمد قاعدة إنتاج ونسعى لمطابقة الدخل.
- قد نحتاج للتقصي العكسي إذا أخذنا اختياراً خاطئاً.
- بعض القواعد لا تحتاج للتقصي العكسي (الإعراب التنبؤيpredictive parsing).

محاضرة اليوم: الإعراب من العمق إلى السطح Bottom-Up parsing



ماهي مسألة الإعراب من العمق إلى السطح Bottom-Up

المحلل القواعدي Syntax Analysis

الهدف Goal: ليكن لدينا القواعد، G ، المطلوب بناء شجرة إعراب لسلسلة (جملة) انطلاقاً من الأوراق صعوداً إلى الجذر (بالاستقراء من جملة الدخل إلى رمز الدخل S).

بالعودة إلى عملية الإعراب من أجل الحصول على الاشتقاق والتي يعبر عنها بـ:

$$S \Rightarrow \delta_0 \Rightarrow \delta_1 \Rightarrow \delta_2 \Rightarrow \ldots \Rightarrow \delta_{n-1} \Rightarrow sentence$$

لكي نشتق δ_{i-1} من δ_i، نطابق بعض القواعد من الجهة اليمينية *rhs b* في δ_i، ومن ثم نستبدل *b* بمقابلها من *lhs*، *A* . *وهذا ما يسمى الاختصار (يفترض أن A→b).*

شجرة الإعراب **parse tree** هي ناتج العلاّمات **tokens** والاختصار **reductions**.

مثال: ليكن لدينا القواعد وسلاسل الدخل التالية **abbcde**:

1. Goal→aABe
2. A→Abc
3. |b
4. B→d

Sentential Form	Production	Position
abbcde	3	2
a A bcde	2	4
a A de	4	3
a A B e	1	4
Goal	-	-

إيجاد اختصار Reductions

- **ما الذي نحاول إيجاده؟**
  - سلسلة فرعية b تطابق الجانب الأيمن من قاعدة الإنتاج والتي تشكل إحدى خطوات الاشتقاق من اليمين. ونطلق على هذه السلسلة الفرعية بشكل غير رمزي اسم المقبض **handle.**
- الشكل الرمزي يعرّف مقبض الجمل المنظور إليها من اليمين δ على أنه الزوج
- $<A \rightarrow b, k>$
- حيث $A \rightarrow b \in P$ و
- k هي الموضع في الشكل الجملي δ للرمز اليميني في b.

(صيغة الجمل المنظور إليها من اليمين: هو صيغة جملة نرصد وقوعها في اشتقاق ما من اليمين.)

- ولأن δهي صيغة جملية منظور إليها من اليمين، فإن السلسلة الفرعية إلى يمين المقبض تحتوي فقط على رموز تحديدية. ولذلك فإن المعرب لايحتاج لمسح هذا الجزء ثانيةً.
- وإذا كانت القواعد غير غامضة unambiguous، فإن هنالك مقبض وحيد لكل صيغة جملية منظور إليها من اليمين.
- (هيكلية البرهان وفقاً للتعريف: إذا لم يكن هنالك غموض فالاشتقاق من اليمين وحيد لذلك يكون لدينا قاعدة إنتاج وحيدة عند كل خطوة من خطوات توليد الشكل الرمزي الجملي، وبالتالي هنالك موضع وحيد تطبق عنده القاعدة وبالتالي هنالك فقط مقبض وحيد (unique handle

فإذا استطعنا إيجاد هذه المقابض فبإمكاننا بناء الاشتقاق!

إيجاد اختصار Reductions

مثال: ليكن لدينا القواعد لجملة منظور إليها من اليسار، المطلوب إيجاد الاشتقاق من اليمين للصيغة $x - 2*y$. (انطلاقاً من الهدف لدينا مقبض وحيد لأن القواعد غير غامضة). ويمكننا في كل خطوة تعريف المقبض كما يلي:

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

Production	Sentential Form	Handle
-	*Goal*	-
1	*Expr*	1,1
3	*Expr – Term*	3,3

والمسألة هي: إيجاد المقابض التي تسمح بإعراب الجملة $x - 2*y$.

إيجاد اختصار Reductions

مثال: ليكن لدينا القواعد لجملة منظور إليها من اليسار، المطلوب إيجاد الاشتقاق من اليمين للصيغة $x - 2*y$. (انطلاقاً من الهدف لدينا مقبض وحيد لأن القواعد غير غامضة). ويمكننا في كل خطوة تعريف المقبض كما يلي:

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

Production	Sentential Form	Handle
-	*Goal*	-
1	*Expr*	1,1
3	*Expr – Term*	3,3
4	*Expr – Term*Factor*	5,5
5	*Expr – Term*id*	9,5
6	*Expr – Factor*id*	7,3
7	*Expr - number*id*	8,3
8	*Term - number*id*	4,1
9	*Factor - number*id*	7,1
10	*id-number*id*	9,1

والمسألة هي: إيجاد المقابض التي تسمح بإعراب الجملة $x - 2*y$.

المحلل القواعدي Syntax Analysis

- نسمي عملية اكشاف المقبض بالتشذيب المقبضي handle pruning.
- نطبق الخوارزمية التالية لكي نجري الاشتقاق من اليمين إلى اليسار:

for *i=n* to *1*, step *-1*
  find the handle $<A \rightarrow b,k>_i$ in δ_i
  replace *b* with *A* to generate δ_{i-1}

(نحتاج إلى *2n مرحلة، حيث n هي طول الاشتقاق*).

- تستند إحدى التطبيقات إلى استخدام الكدسة stack للاحتفاظ برموز القواعد وعازل دخل input buffer للاحتفاظ بالسلسلة التي يجب إعرابها. نطبق أربع عمليات هي:
  - **الإزاحة shift:** يجري إزاحة (دفع) الدخل في قمة المكدس.
  - **الاختصار reduce:**
  - دفع طرف المقبض الأيمن إلى قمة المكدس
  - توضيع الطرف الأيسر من المقبض ضمن الكدسة
  - جلب المقبض من الكدسة
  - دفع الرمز غير التحديدي الأيسر إلى الكدسة.
  - **القبول accept:** تنهي الإعراب وتعلن نجاح العملية.
  - **الخطأ error:** تستدعي برنامج كشف الأخطاء error recovery routine.

تنفيذ معرب shift-reduce parser


```
push $ onto the stack
token = next_token()
repeat
   if the top of the stack is a handle A→b
          then /* reduce b to A */
                  pop the symbols of b off the stack
                  push A onto the stack
          elseif (token != eof) /* eof: end-of-file = end-of-input */
                  then /* shift */
                          push token
                          token=next_token()
                  else /* error */
                          call error_handling()
until (top_of_stack == Goal && token==eof)
```

تبين الأخطاء Errors: a) فشلنا في الحصول على مقبض أو b) عندما نصل إلى EOF ونحتاج إلى الإزاحة shift. وهكذا فإن المعرب يحتاج إلى تمييز الأخطاء الإعرابية syntax errors.

Syntax Analysis المحلل القواعدي

تنفيذ معرب shift-reduce parser
مثال: x–2*y

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

1-أزح حتى تصبح قمة المكدس هي النهاية اليمنى للمقبض.
2-أوجد النهاية اليسرى للمقبض واختصر.

(5 shifts, 9 reduces, 1 accept)

Stack	Input	Handle	Action
$	id – num * id	None	Shift
$ id	– num * id	9,1	Reduce 9
$ Factor	– num * id	7,1	Reduce 7
$ Term	– num * id	4,1	Reduce 4
$ Expr	– num * id	None	Shift
$ Expr –	num * id	None	Shift
$ Expr – num	* id	8,3	Reduce 8
$ Expr – Factor	* id	7,3	Reduce 7
$ Expr – Term	* id	None	Shift
$ Expr – Term *	id	None	Shift
$ Expr – Term * id		9,5	Reduce 9
$ Expr – Term * Factor		5,5	Reduce 5
$ Expr – Term		3,3	Reduce 3
$ Expr		1,1	Reduce 1
$ Goal		none	Accept

ما الذي يمكن أن يجري بشكل خاطئ؟

- **تعارض Shift/reduce:** لايستطيع المعرب أن يقرر فيما إذا كان عليه الإزاحة أو الاختصار.

مثال: القواعد المعلقة المتعلقة بـelse، تنتج عن غموض القواعد.

الحل أ)هو تعديل القواعد، ب)الحل من خلال مرحلة الإزاحة shift.

- **تعارض الاختصار/الاختصار Reduce/reduce conflicts:** لايستطيع المعرب تقرير أي الاختصارات العديدة الممكنة سيقوم بتنفيذه.

مثال: **`id(id,id)`؛ يرتبط الاختصار بـ: فيما إذا كانت `id` الأولى تشير إلى** array أو إلى function.

قد يكون ذلك صعب التحديد.

ومفتاح إعراب العمق- السطح bottom-up parsing: هو آلية إيجاد المقبض.

قواعد LR(1)
(مثال مناسب على تطبيق النظرية لحل المسائل المعقدة عملياً)

L تعبر عن مسح الدخل من اليسار إلى اليمين left-to-right؛
و R تعبر عن بناء اشتقاق من اليمين إلى اليسار بشكل عكسي
1 تقابل عدد رموز الدخل التي نتوقعها.

نسمي القواعد LR(1) إذا استطعنا -في حال الاشتقاق بالرمز أقصى اليمين-

أ)أن نعزل المقبض لكل صيغة اشتقاق للجملة من اليمين

و ب)تحديد قاعدة الإنتاج التي يمكننا من خلالها إجراء الاختصار.

وذلك من خلال مسح صيغة الجملة من اليسار إلى اليمين، آخذين بعين الاعتبار على الأكثر رمز وحيد مابعد النهاية اليمنى للمقبض.

- **قواعد LR(1) مستخدمة بشكل كبير للبناء الآلي للمعربات المرنة الفعالة:**
  - يمكننا افتراضياً التعبير عن جميع لغات القواعد حرّة السياق بصيغة LR(1).
  - قواعد LR أكثر القواعد عمومية والتي يمكن إعرابها بمعرب shift-reduce دون تقصي عكسي (قواعد حرة السياق تحديدية)
  - يمكننا تنفيذ المعربات بزمن متناسب مع العلاّمات+الاختصارات tokens+reductions.
  - معربات LR تكشف الخطأ بأقرب وقت ممكن من خلال مسح الدخل من اليسار إلى اليمين.

أساس LR Parsing

L تعبر عن مسح الدخل من اليسار إلى اليمين left-to-right؛
و R تعبر عن بناء اشتقاق من اليمين إلى اليسار بشكل عكسي
1 تقابل عدد رموز الدخل التي نتوقعها.

- قراءة العلامات tokens من عازل الدخل (بنفس طريقة معرب shift-reduce).
- إضافة معلومات حالة إضافية بعد كل رمز في الكدسة. تلخص الحالة المضافة المعلومات المتضمنة في الكدسة والواقعة تحت هذه الحالة في الكدسة. وستبدو الكدسة كما يلي:

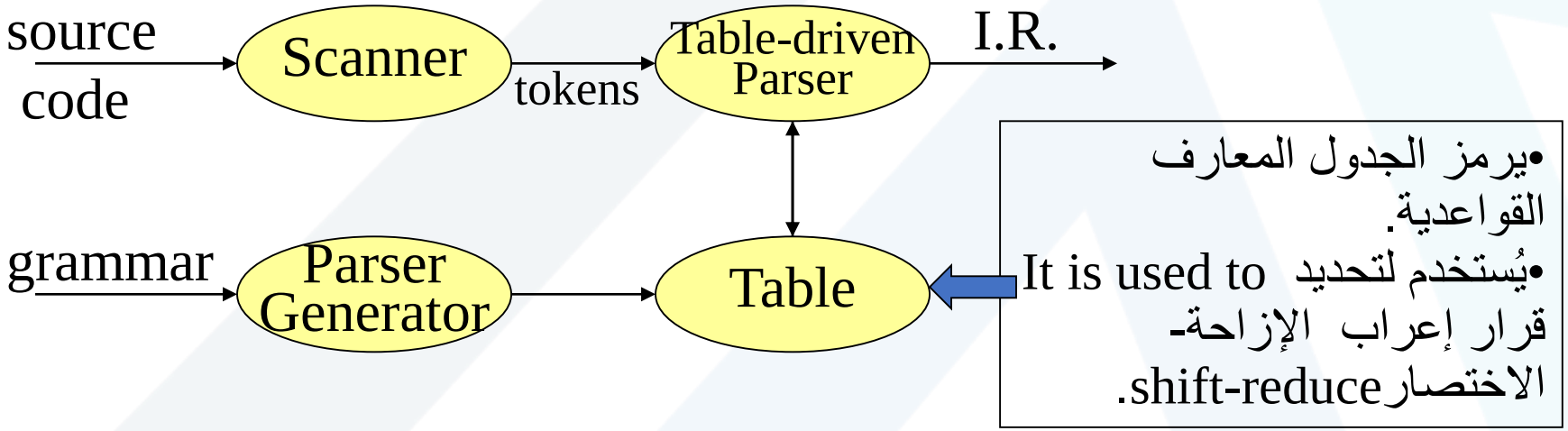
$$\$ S_0 \; Expr \; S_1 \; - \; S_2 \; num \; S_3$$

- باستخدام جدول مؤلف من جزئين:
  - **الفعل** **action**[state_on_top_of_stack, input_symbol]: والذي يعيد إما shift s (push a symbol and a state)؛ أو reduce by a rule؛ أو accept؛ أو error.
  - **الانتقال** **goto**[state_on_top_of_stack,non_terminal_symbol]: والذي يعيد حالة جديدة لدفعها إلى الكدسة بعد الاختصار.

```
Push $ onto the stack
push s0
token=next_token()
repeat
  s=top_of_the_stack  /* not pop! */
  if ACTION[s,token]==‘reduce A→b’
      then  pop 2*(symbols_of_b) off the stack
            s=top_of_the_stack  /* not pop! */
            push A; push GOTO[s,A]
      elseif ACTION[s,token]==‘shift sx’
                  then push token; push sx
                       token=next_token()
                  elseif ACTION[s,token]==‘accept’
                       then break
                       else report_error
end repeat
report_success
```

أساس LR Parsing

- معربات LR(1) مقادة بالجداول، وهي معربات إزاحة-اختصار shift-reduce تستخدم سياق يميني من أجل معالجة التمييز.
- يمكننا بناؤها يدوياً؛ وتناسب البناء الآلي.
- وباختصار: هذه المعربات أكثر قوة.



مثال

لنأخذ بعين الاعتبار القواعد والجداول التالية:

1. *Goal* → *CatNoise*
2. *CatNoise* → *CatNoise miau*
3. | *miau*

STATE	ACTION		GOTO
	eof	miau	CatNoise
0	-	Shift 2	1
1	accept	Shift 3	
2	Reduce 3	Reduce 3	
3	Reduce 2	Reduce 2	

Example 1: (input string miau)

Stack	Input	Action
$ s0	miau eof	Shift 2
$ s0 miau s2	eof	Reduce 3
$ s0 CatNoise s1	eof	Accept

Example 2: (input string miau miau)

Stack	Input	Action
$ s0	miau miau eof	Shift 2
$ s0 miau s2	miau eof	Reduce 3
$ s0 CatNoise s1	miau eof	Shift 3
$ s0 CatNoise s1 miau s3	eof	Reduce 2
$ s0 CatNoise s1	eof	accept

لاحظ أنه لايمكن أن يكون هنالك خطأ قواعدي syntax error لـ CatNoise، لأن لها رمز إنهاء وحيد.
أما مثلاً: "miau woof" هي مسألة معجمية، وليست خطأً قواعدياً.

eof هي رمز تقليدي يعبر عن end-of-file (=end of input)

Syntax Analysis المحلل القواعدي

مثال2
x–2*y

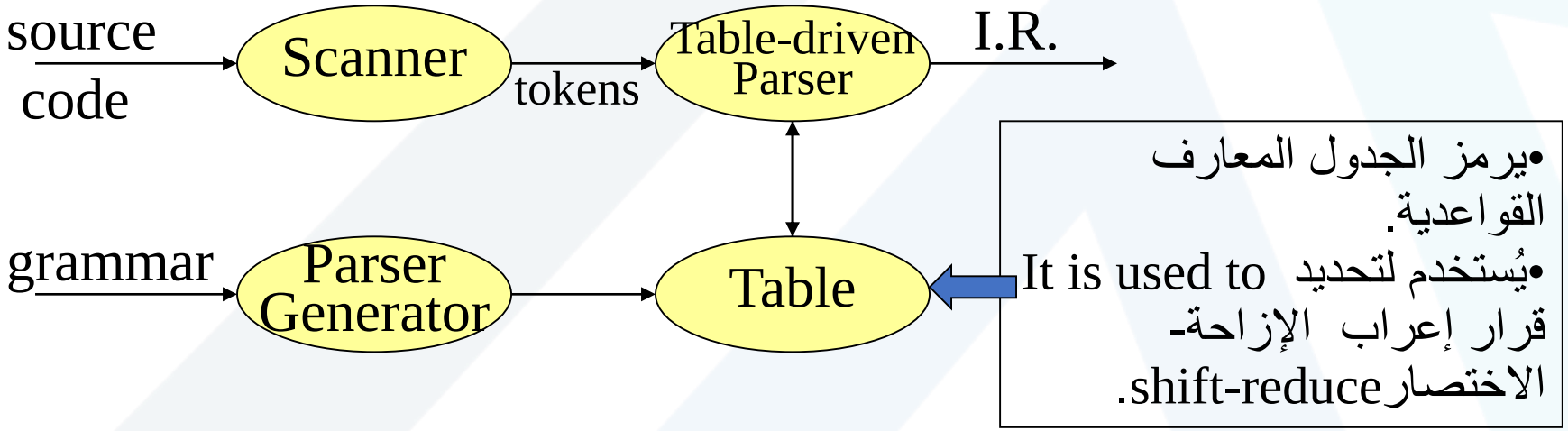
1. *Goal → Expr*
2. *Expr → Expr + Term*
3. *| Expr – Term*
4. *| Term*
5. *Term → Term * Factor*
6. *| Term / Factor*
7. *| Factor*
8. *Factor → number*
9. *| id*

Stack	Input	Action
$ s0	id-num*id eof	Shift 5
$ s0 id s5	-num*id eof	Reduce 9
$ s0 Factor s3	-num*id eof	Reduce 7
$ s0 Term S2	-num*id eof	Reduce 4
$ s0 Expr S1	-num*id eof	Shift 7
$ s0 Expr – S7	num*id eof	Shift 4
$ s0 Expr – S7 num S4	*id eof	Reduce 8
$ s0 Expr – S7 Factor S3	*id eof	Reduce 7
$ s0 Expr – S7 Term S11	*id eof	Shift 8
$ s0 Expr – S7 Term S11* S8	id eof	Shift 5
$ s0 Expr – S7 Term S11* S8 id S5	eof	Reduce 9
$ s0 Expr – S7 Term S11* S8 Factor S12	eof	Reduce 5
$ s0 Expr – S7 Term S11	eof	Reduce 3
$ s0 Expr S1	eof	Accept

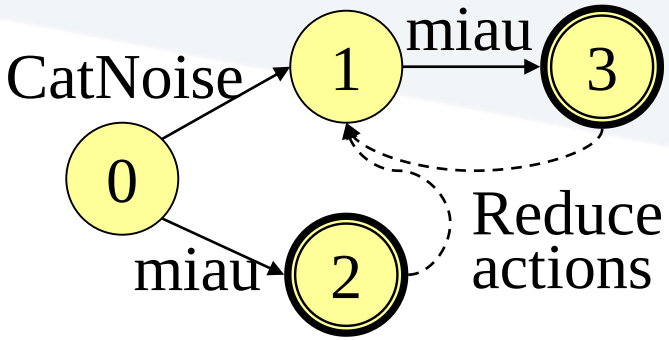
STATE	ACTION							GOTO		
	eof	+	–	*	/	num	id	Expr	Term	Factor
0						S 4	S 5	1	2	3
1	Acc	S 6	S 7							
2	R 4	R 4	R 4	S 8	S 9					
3	R 7	R 7	R 7	R 7	R 7					
4	R 8	R 8	R 8	R 8	R 8					
5	R 9	R 9	R 9	R 9	R 9					
6						S 4	S 5		10	3
7						S 4	S 5		11	3
8						S 4	S 5			12
9						S 4	S 5			13
10	R 2	R 2	R 2	S 8	S 9					
11	R 3	R 3	R 3	S 8	S 9					
12	R 5	R 5	R 5	R 5	R 5					
13	R 6	R 6	R 6	R 6	R 6					

خوارزمية بناء جدول LR(1)

- عملية الإعراب سهلة في حال وجود جدول LR(1)
- لكن كيف يتم بناء هذا الجدول؟



LR Parsers:
كيف يعمل؟



- المفتاح: لغة المقابض نظامية.
  - بناء DFA خاص بتمييز المقابض
  - جداول الفعاليات Action والانتقال Goto ترمز الـDFA.
- كيف نولد جداول الفعاليات والانتقالات (Action and Goto tables)؟
  - نستخدم القواعد لبناء نموذج الـDFA.
  - استخدام نموذج بناء جداول الفعاليات والانتقالات (Action and Goto tables)
  - إذا نجح البناء فإن القواعد هي LR(1).
- ثلاثة خوارزميات عامة تستخدم لبناء الجداول:
  - LR(1): مجموعة كاملة من قواعد LR(1)، جداول كبيرة وبطء وبناء كبير.
  - SLR(1): حجم أصغري لصفوف القواعد؛ حجم أصغري للجداول، بناء بسيط وسريع.
  - LALR(1): حجم متوسط للقواعد؛ حجم أصغري للجداول؛ عام جداً.

سندرس أول خوارزمية فقط

عناصر LR(1)

- عنصر LR(1) هو زوج [A,B]، حيث:
  - A هي قاعدة إنتاج $\alpha \rightarrow \beta\gamma\delta$ مع • a عند موضع معين من الـ*rhs*.
  - B هي رمز تنبؤي lookahead symbol.
- رمز • يشير إلى موضع قمة الكدسة.
  - $[\alpha \rightarrow \beta\gamma \bullet \delta, a]$: يكون الدخل الذي هو ضمن مجال الرؤية (أي الموجود في الكدسة) متناسب مع استخدام $\alpha \rightarrow \beta\gamma\delta$، وأن المعرب قد ميز التسلسل $\beta\gamma$.
  - $[\alpha \rightarrow \beta\gamma\delta \bullet, a]$: رأى المعرب $\beta\gamma\delta$، وأن الرمز التنبؤي لـ a متناسب مع الاختصار إلى α.
- قاعدة الإنتاج $\alpha \rightarrow \beta\gamma\delta$ مع التنبؤ a، تولد:
- $[\alpha \rightarrow \bullet\beta\gamma\delta, a]$, $[\alpha \rightarrow \beta \bullet \gamma\delta, a]$, $[\alpha \rightarrow \beta\gamma \bullet \delta, a]$, $[\alpha \rightarrow \beta\gamma\delta \bullet, a]$
- ***ومجموعة عناصر LR(1) منتهية.***
  - *مجموعات عناصر LR(1) تمثل حالات معرب LR(1).*

خوارزمية بناء الجدول LR(1)

- بناء الجدول Table construction:
  - **1-بناء المجموعة القانونية من مجموعات** عناصر LR(1)، S
    - أ)نبدأ بـ S_0 مع النظر إلى [Goal→•α, eof] وإيجاد جميع العناصر المكافئة كـ**closure**(S_0).
    - نحسب تكرارياً، من أجل كل S_k وكل رمز α (العناصر التحديدية وغير التحديدية)، والانتقال **goto**(S_k,α). فإذا لم تكن المجموعة في الاختيار نقوم بإضافتها. وهذا يوصلنا في حقيقة الأمر إلى نقطة ثابتة.
  - **2-نملأ الجدول من تشكيلة من مجموعات عناصر** LR(1).
  - ترمز المجموعة القانونية كلياً مخطط الانتقال لـDFA الموجد للمقابض.
  - التنبؤ هو المفتاح لاختيار الفعالية action:

خوارزمية بناء الجدول LR(1)
أولاً): Closure(state)

Closure(s) // s is the state
  while (s is still changing)
    for each item $[\alpha \rightarrow \beta \bullet \gamma \delta, a]$ in s
      for each production $\gamma \rightarrow \tau$
        for each terminal b in FIRST(δa)
          if $[\gamma \rightarrow \bullet \tau, b]$ is not in s, then add it.

Goto(s,x)
  new=∅
  for each item [α→β•xδ,a] in s
    add [α→βx•δ,a] to new
  return closure(new)

مثال 2

بناء جدول الإعراب LR (1) Parse Table

1. G → L
2. L → L P
3. L → P
4. P → (P)
5. P → ( )

لحساب المجموعة S0

1- نكتب قاعدة الهدف بتوقع رمز محرف نهاية الملف eof (القاعدة الأولى)

ومنها ننطلق لبقية الحالات:

2- نقسم القاعدة لجزأين وفق $\alpha \rightarrow \beta \bullet \gamma \delta, a$:

الأول هو الرمز اليساري γ

الثاني هو قيمة First لما تبقى من النتاج مع الرمز التحديدي أي First(δa)

3- نضيف توقع لكل نتاج $\gamma \rightarrow \tau$ بدلالة الرمز First(δa) أي $[\gamma \rightarrow \bullet \tau, b]$

في حال وجود التوقع سابقاً لا نقوم بإضافته للمجموعة.

S0: {[G→•L,eof], [L→•LP,eof], [L→•P,eof], [P→•(P),eof], [P→•(),eof]
[L→•LP, ( ], [L→•P, ( ], [P→•(P), ( ], [P→•(), ( ]}

For the first rule: [G→•L,eof], γ = L, δ=eof → FIRST(eof)=eof → study all production of L with predictive symbol eof

For the second rule: [L→•LP,eof], **γ = L, δ=P → FIRST(P eof)= (→ study all production of L with predictive symbol (**

بناء جدول الإعراب LR (1) Parse Table للنحو الأصلي

1. G → L
2. L → L P
3. L → P
4. P → (P)
5. P → ( )

S0: {[G→•L,eof], [L→•LP,eof], [L→•P,eof], [P→•(P),eof], [P→•(),eof]
[L→•LP, ( ], [L→•P, ( ], [P→•(P), ( ], [P→•(), ( ]}

L

P

(

S1: {[G→L•,eof],
[L→L•P,eof],
[P→•(P),eof],
[P→•(),eof]
[L→L•P, ( ],
[P→•(P), ( ],
[P→•(), ( ]}

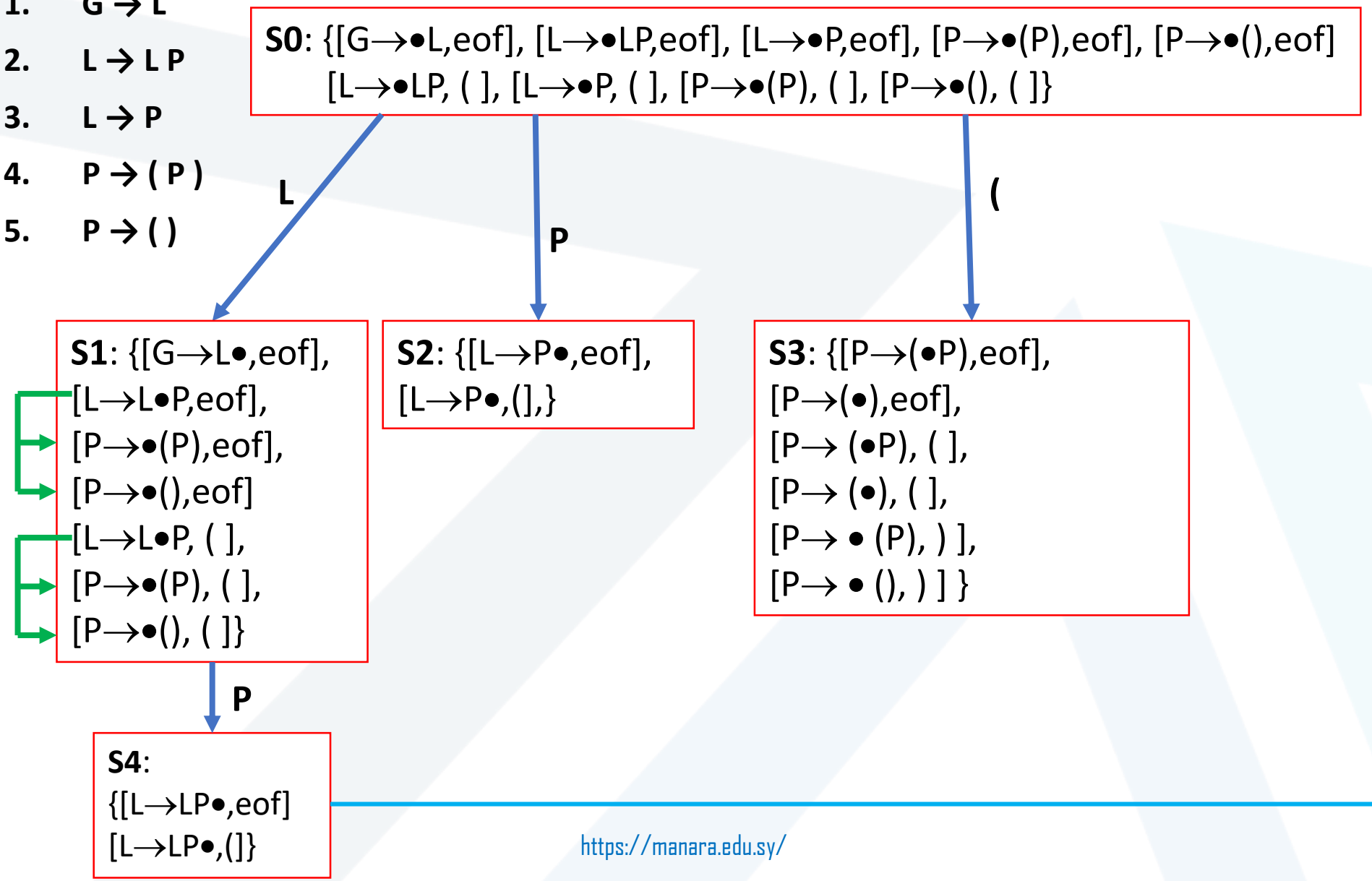
S2: {[L→P•,eof],
[L→P•,(],}

S3: {[P→(•P),eof],
[P→(•),eof],
[P→ (•P), ( ],
[P→ (•), ( ]
[P→ • (P),)],
[P→ • (),)] }

[P→(• P),(]
FIRST( )( )={) }
هنا ندرس كل نتاجات P بتوقع الرمز)

بناء جدول الإعراب LR (1) Parse Table للنحو الأصلي

1. G → L
2. L → L P
3. L → P
4. P → (P)
5. P → ( )



بناء جدول الإعراب LR (1) Parse Table للنحو الأصلي

1. G → L
2. L → L P
3. L → P
4. P → (P)
5. P → ( )

S0: {[G→•L,eof], [L→•LP,eof], [L→•P,eof], [P→•(P),eof], [P→•(),eof]
[L→•LP, ( ], [L→•P, ( ], [P→•(P), ( ], [P→•(), ( ]}

L

P

(

S1: {[G→L•,eof],
[L→L•P,eof],
[P→•(P),eof],
[P→•(),eof]
[L→L•P, ( ],
[P→•(P), ( ],
[P→•(), ( ]}

S2: {[L→P•,eof],
[L→P•,(],}

S3: {[P→(•P),eof], [P→(•),eof],
[P→ (•P), ( ], [P→ (•), ( ], [P→• (P),)],
[P→ • (),)] }

P

(

S4:
{[L→LP•,eof]
[L→LP•,(]}

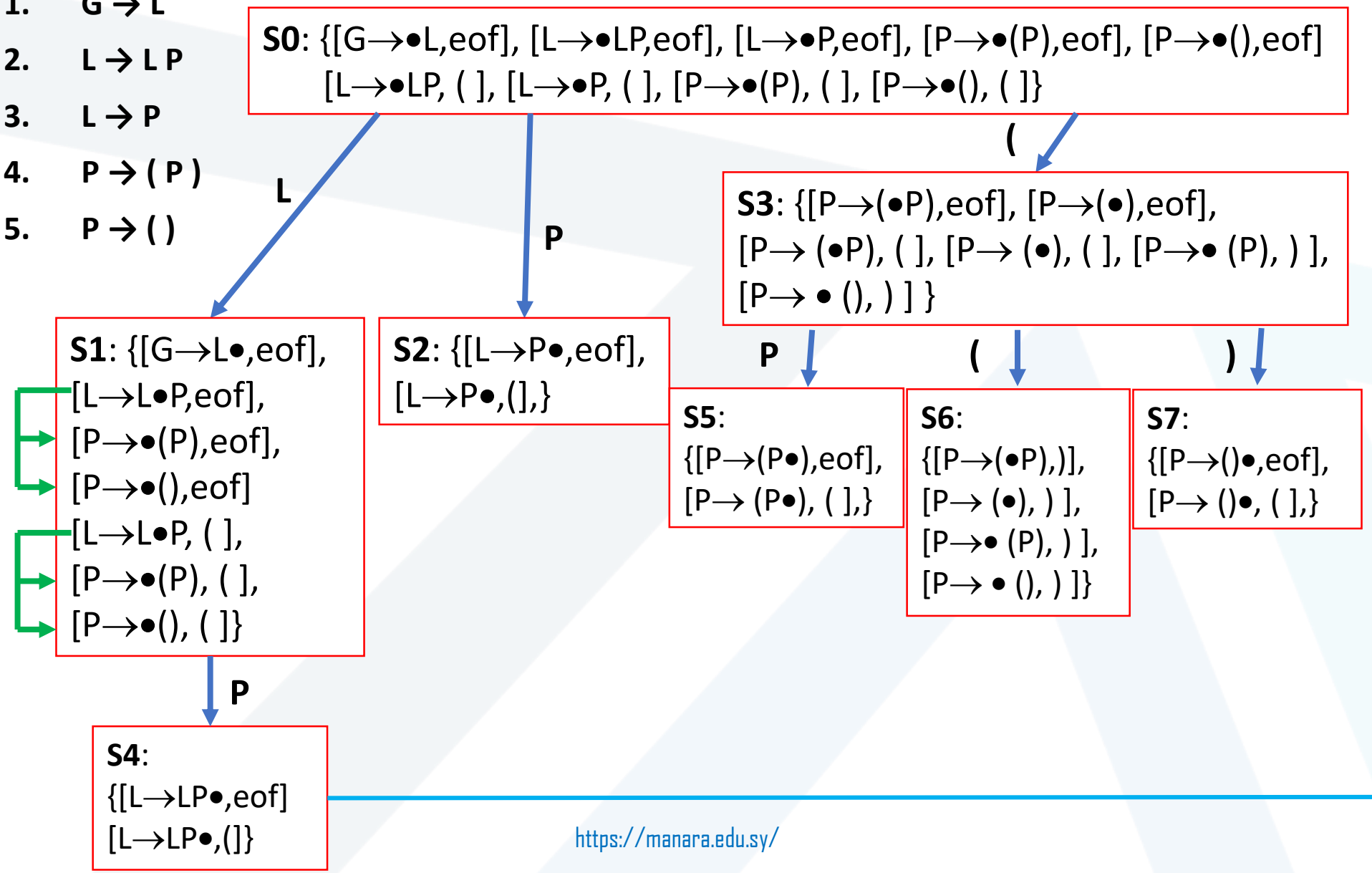
S5:
{[P→(P•),eof],
[P→ (P•), ( ],}

S6:
{[P→(•P),)],
[P→ (•),)],
[P→• (P),)],
[P→ • (),)]}

[P→(• P),)]
FIRST( )))={) }
(بتوقع الرمز Pهنا ندرس كل نتاجات

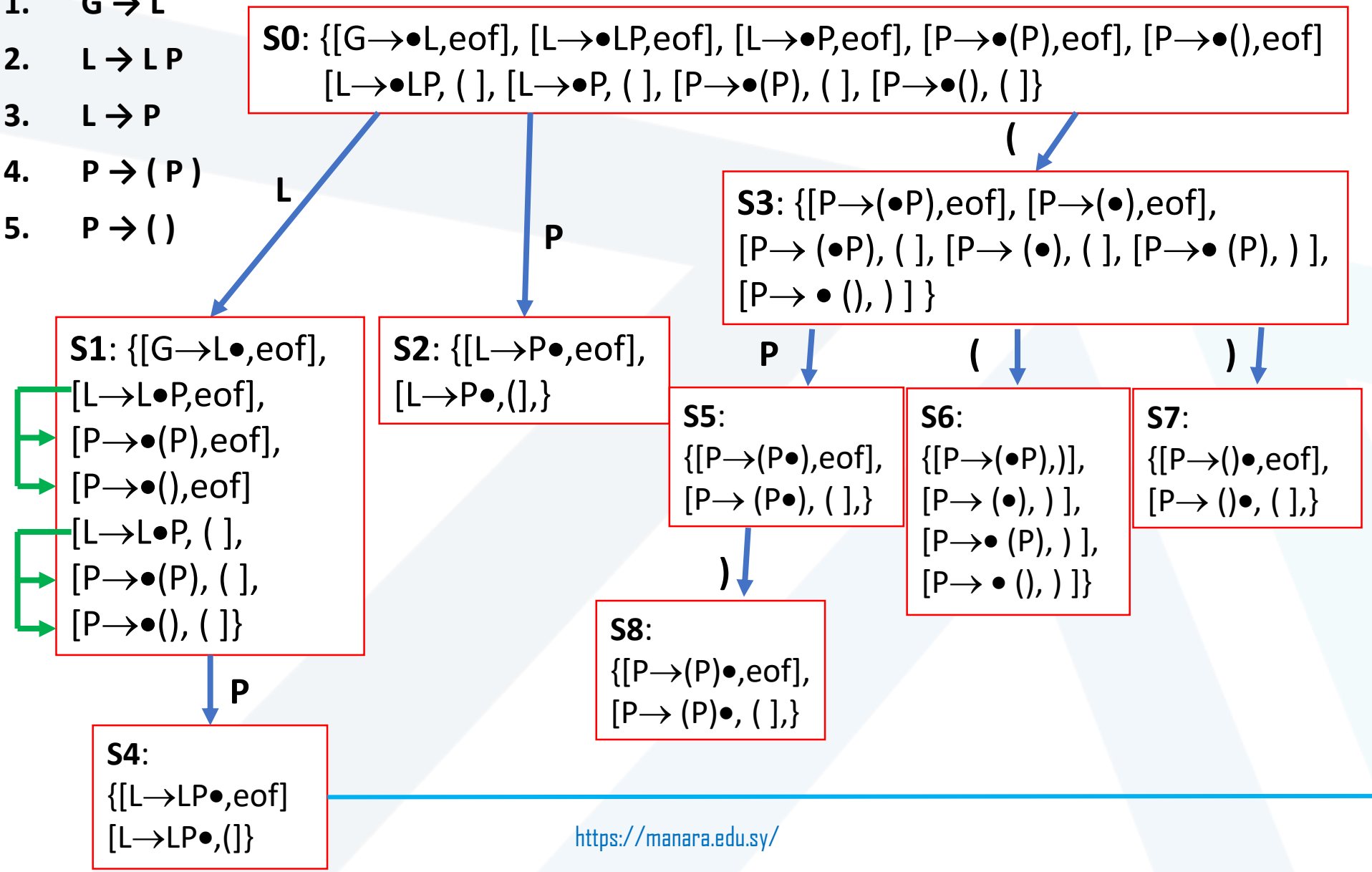
بناء جدول الإعراب LR (1) Parse Table للنحو الأصلي

1. G → L
2. L → L P
3. L → P
4. P → (P)
5. P → ( )



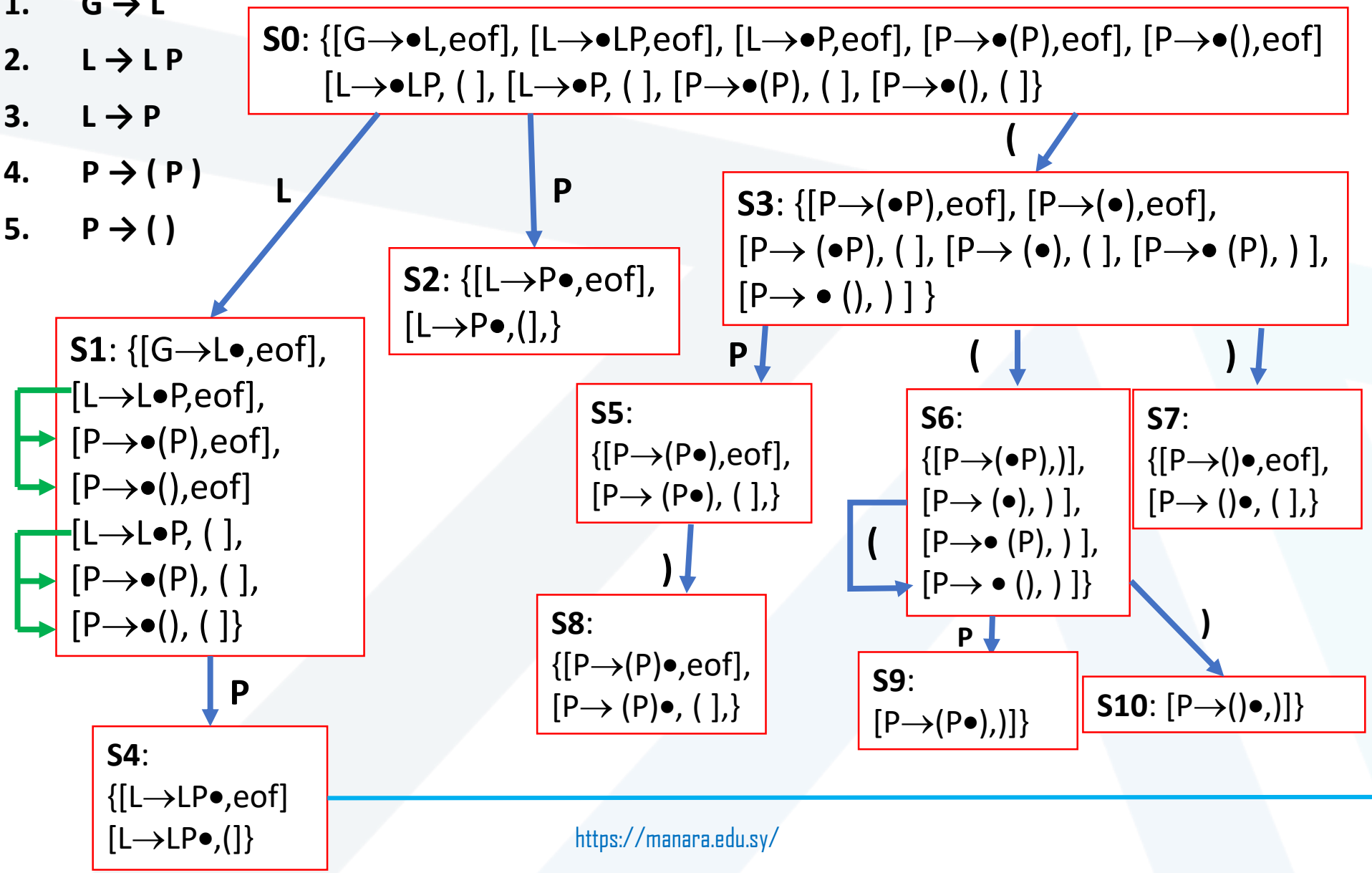
بناء جدول الإعراب LR (1) Parse Table للنحو الأصلي

1. G → L
2. L → L P
3. L → P
4. P → (P)
5. P → ( )



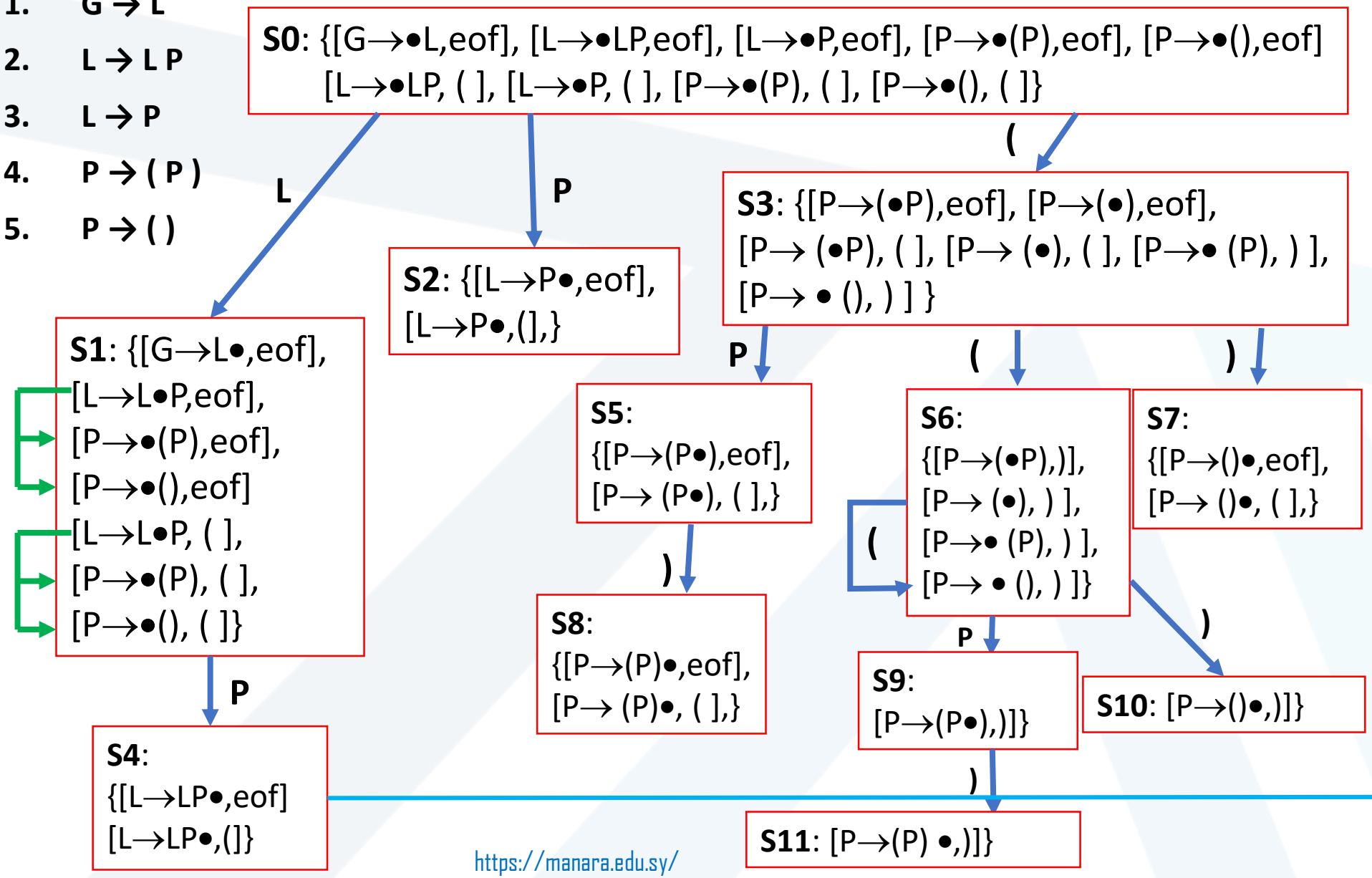
بناء جدول الإعراب LR (1) Parse Table للنحو الأصلي

1. G → L
2. L → L P
3. L → P
4. P → (P)
5. P → ( )



بناء جدول الإعراب LR (1) Parse Table للنحو الأصلي

1. G → L
2. L → L P
3. L → P
4. P → (P)
5. P → ( )



بناء جدول الإعراب LR (1) Parse Table للنحو الأصلي

1. G → L
2. L → L P
3. L → P
4. P → (P)
5. P → ( )

S0: {[G→•L,eof], [L→•LP,eof], [L→•P,eof], [P→•(P),eof], [P→•(),eof]
[L→•LP, ( ], [L→•P, ( ], [P→•(P), ( ], [P→•(), ( ]}

L

P

(

S1: {[G→L•,eof],
[L→L•P,eof],
[P→•(P),eof],
[P→•(),eof]
[L→L•P, ( ],
[P→•(P), ( ],
[P→•(), ( ]}

S2: {[L→P•,eof],
[L→P•,(],}

Reduce 3

S3: {[P→(•P),eof], [P→(•),eof],
[P→ (•P), ( ], [P→ (•), ( ], [P→• (P),)],
[P→ • (),)] }

eof

P

(

accept

S4:
{[L→LP•,eof]
[L→LP•,(]}

Reduce 2

S3

P

(

)

S5:
{[P→(P•),eof],
[P→ (P•), ( ],}

S6:
{[P→(•P),)],
[P→ (•),)],
[P→• (P),)],
[P→ • (),)]}

S7:
{[P→()•,eof],
[P→ ()•, ( ],}

Reduce 5

)

S8:
{[P→(P)•,eof],
[P→ (P)•, ( ],}

Reduce 4

P

S9:
[P→(P•),)]}

)

S10: [P→()•,)]}

Reduce 5

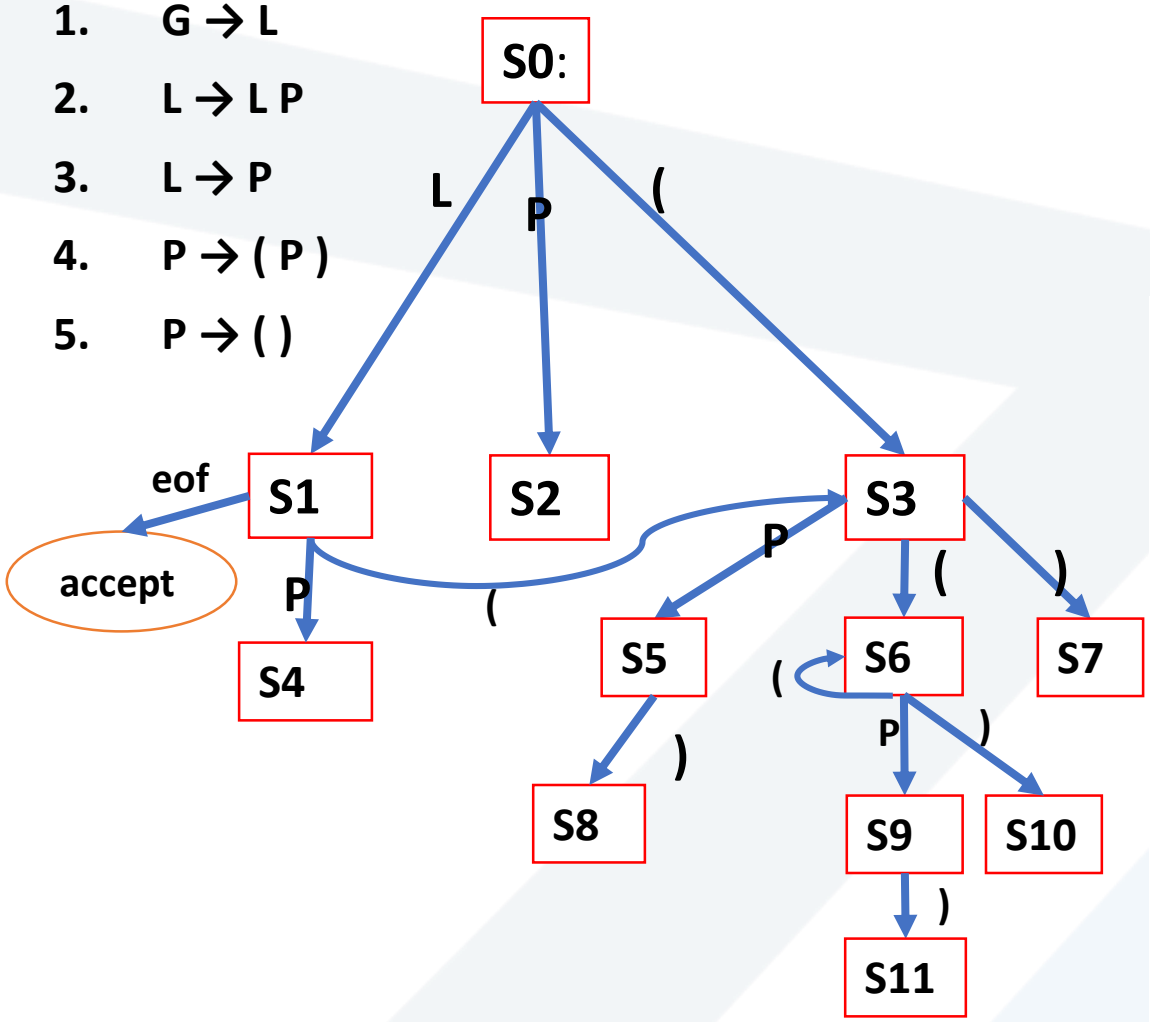
)

S11: [P→(P) •,)]}

Reduce 4

مخطط DFA للجدول + رسم جدول LR(1)

1. G → L
2. L → L P
3. L → P
4. P → (P)
5. P → ( )

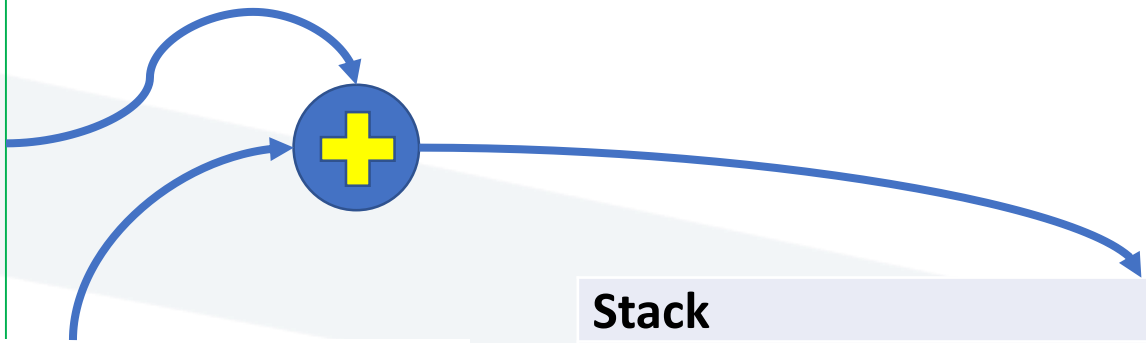


STATE	ACTION			GOTO	
	(	)	eof	L	P
0	S3			1	2
1	S3		accept		4
2	R3		R3		
3	S6	S7			5
4	R2		R2		
5		S8			
6	S6	S10			9
7	R5		R5		
8	R4		R4		
9		S11			
10		R5			
11		R4			

LR(1)الإعراب من أسفل لأعلى باستخدام جدول

إعراب السلسلة ()(())()

1. G → L
2. L → L P
3. L → P
4. P → (P)
5. P → ( )



STATE	ACTION			GOTO	
	(	)	eof	L	P
0	S3			1	2
1	S3		accept		4
2	R3		R3		
3	S6	S7			5
4	R2		R2		
5		S8			
6	S6	S10			9
7	R5		R5		
8	R4		R4		
9		S11			
10		R5			
11		R4			

Stack	Input	Action
$ s0	(())()	Shift 3
$ s0(s3	())()	Shift 6
$ s0(s3(S6	))()	Shift 10
$ s0(s3(S6)s10	)()	Reduce 5
$ s0(s3 P S5	)()	Shift8
$ s0(s3 P S5)S8	()	Reduce 4
$ S0 P S2	()	Reduce 3
$ S0 L S1	()	Shift 3
$ S0 L S1 (S3	)	Shift 7
$ S0 L S1 (S3) S7	$	Reduce5
$ S0 L S1 P S4	$	Reduce 2
$ S0 L S1	$	Accept

المطلوب تحويل الحل إلى شكل رسومي مثل المثال السابق ثم استنتاج جدول LR(1)

S0: {[Goal→•CatNoise,eof], [CatNoise→•CatNoise miau, eof], [CatNoise→•miau, eof], [CatNoise→•CatNoise miau, miau], [CatNoise→•miau, miau]}

1. *Goal → CatNoise*
2. *CatNoise → CatNoise miau*
3. | *miau*

S1 (x=CatNoise): [Goal→CatNoise•,eof], [CatNoise→CatNoise• miau, eof], [CatNoise→CatNoise• miau, miau]
S2 (x=miau): [CatNoise→miau•, eof], [CatNoise→miau•, miau]
S3 (from S1): [CatNoise→CatNoise miau•, eof], [CatNoise→CatNoise miau•, miau]

STATE	ACTION		GOTO
	eof	miau	CatNoise
0			
1			
2			
3			