اصول طراحی کامپایلرها

حسین کارشناس

دانشکده مهندسی کامپیوتر

ترم اول ۹۸ – ۹۷

تحلیل دستوری

- گرامرهای مستقل از متن
- روش‌های تجزیه
- تجزیه نزول بازگشتی
- تجزیه پیشگویانه
- مدیریت خطا
- تجزیه پایین به بالا
- تجزیه انتقال کاهش
- تجزیه LR
  - SLR، CLR، LALR

تجزیه پایین به بالا

تجزیه پایین به بالا

- تجزیه با شروع از رشته ورودی تا رسیدن به نشانه شروع گرامر
- ساختن درخت تجزیه از پایین به بالا
  - تجزیه از برگ‌های درخت شروع می‌شود
  - در هر مرحله ادغام تعدادی از شاخه‌ها با هم و ساخت درخت بزرگتر
  - توقف با ساخت درخت حاوی نشانه شروع گرامر در ریشه و شامل همه برگ‌ها
- بکارگیری قواعد تولید به صورت معکوس
  - **کاهش** (reduction) رشته ورودی به نشانه شروع گرامر
    - معکوس اشتقاق
  - در هر گام کاهش یک زیر رشته منطبق با بدنه یک قاعده تولید با نشانه غیرپایانی سمت چپ آن قاعده جایگزین می‌شود

تجزیه پایین به بالا

- مثال: کاهش رشته ورودی به نشانه شروع گرامر

$E \rightarrow T + E \mid T$

$T \rightarrow int * T \mid int \mid (E)$

int * int + int

int * int + int

تجزیه پایین به بالا

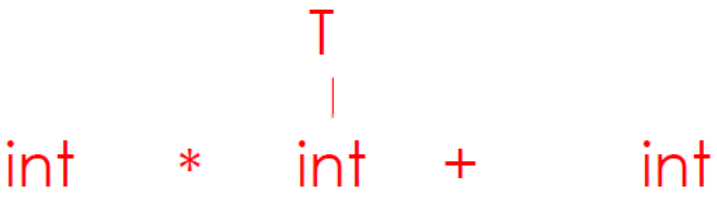
- مثال: کاهش رشته ورودی به نشانه شروع گرامر

$E \rightarrow T + E \mid T$

$T \rightarrow int * T \mid int \mid (E)$

int * int + int

int * T + int



تجزیه پایین به بالا

- مثال: کاهش رشته ورودی به نشانه شروع گرامر

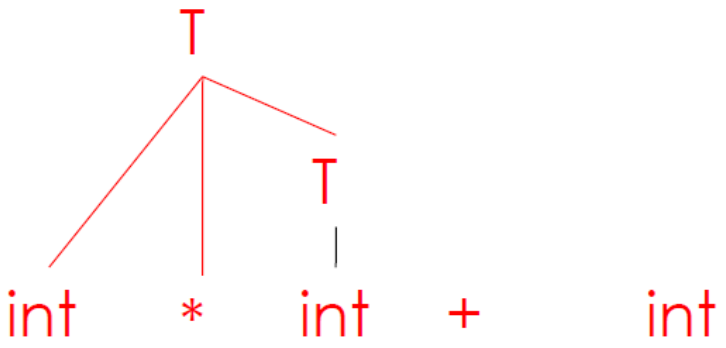
$E \rightarrow T + E \mid T$

$T \rightarrow int * T \mid int \mid (E)$

int * int + int

int * T + int

T + int



تجزیه پایین به بالا

- مثال: کاهش رشته ورودی به نشانه شروع گرامر

$E \rightarrow T + E \mid T$

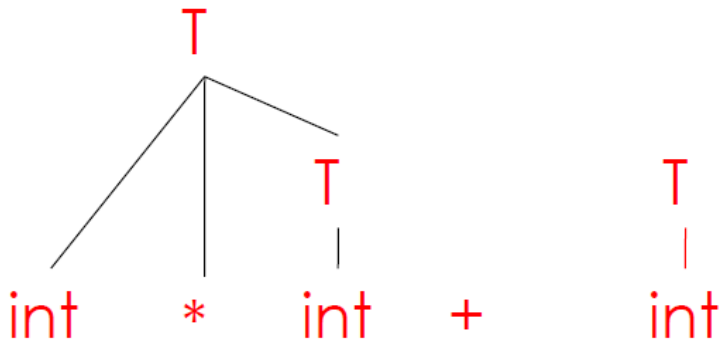
$T \rightarrow int * T \mid int \mid (E)$

int * int + int

int * T + int

T + int

T + T



تجزیه پایین به بالا

- مثال: کاهش رشته ورودی به نشانه شروع گرامر

$E \rightarrow T + E \mid T$

$T \rightarrow int * T \mid int \mid (E)$

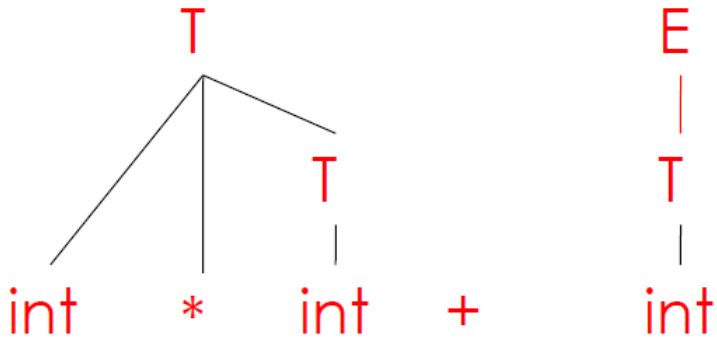
int * int + int

int * T + int

T + int

T + T

T + E



تجزیه پایین به بالا

- مثال: کاهش رشته ورودی به نشانه شروع گرامر

$E \rightarrow T + E \mid T$

$T \rightarrow int * T \mid int \mid (E)$

int * int + int

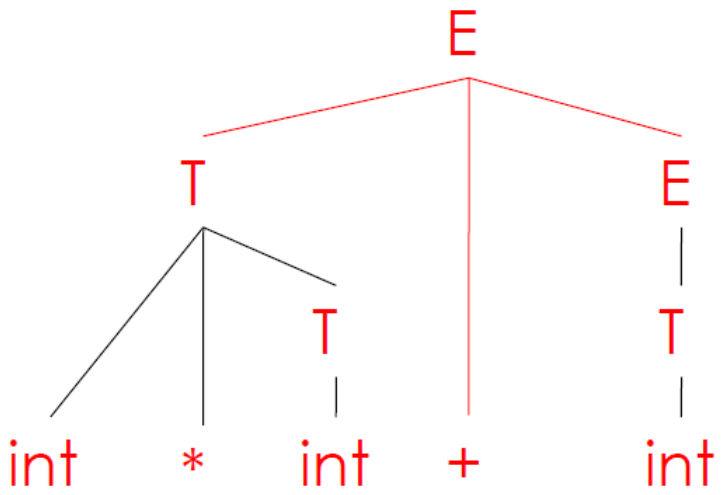
int * T + int

T + int

T + T

T + E

E

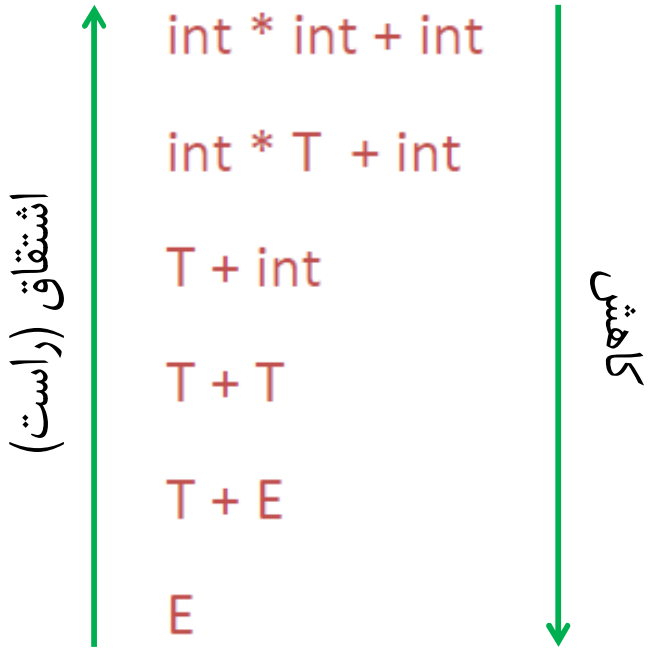


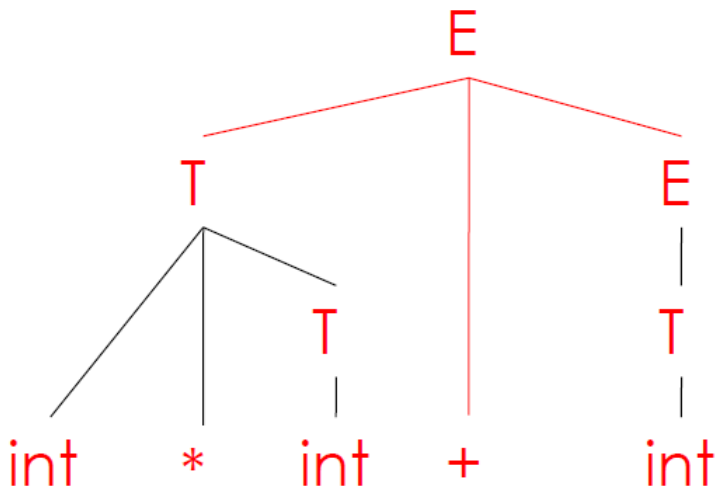
تجزیه پایین به بالا

- مثال: کاهش رشته ورودی به نشانه شروع گرامر

$E \rightarrow T + E \mid T$

$T \rightarrow int * T \mid int \mid (E)$





تجزیه پایین به بالا

- سوال: کدامیک از دنباله‌های کاهش روی رشته **–(id + id) + id** برای گرامر داده شده صحیح است؟

$E \rightarrow E' \mid E' + E$

$E' \rightarrow -E' \mid id \mid (E)$

- -(id + id) + id
  -(id + E') + id
  -(id + E) + id
  -(E' + E) + id
  -(E) + id
  -E' + id
  E' + id
  E' + E'
  E' + E
  E

- -(id + id) + id
  -(E' + id) + id
  -(E' + E') + id
  -(E' + E) + id
  -(E) + id
  -E' + id
  E' + id
  E' + E'
  E' + E
  E

- -(id + id) + id
  -(E' + id) + id
  -(E' + E') + id
  -(E' + E') + E'
  -(E' + E) + E'
  -(E) + E'
  -E' + E'
  E' + E'
  E' + E
  E

- -(id + id) + id
  -(id + id) + E'
  -(id + id) + E
  -(E' + id) + E
  -(E' + E') + E
  -(E' + E) + E
  -(E) + E
  -E' + E
  E' + E
  E

تجزیه پایین به بالا

- دنبال کردن یک اشتقاق راست به صورت معکوس در حین تجزیه
  - معکوس دنباله گام‌های کاهش معتبر یک اشتقاق راست را نشان می‌دهد
  - زیررشته سمت راست بدنه کاهش داده شده فقط شامل نشانه‌های پایانی است
    - همیشه یک اشتقاق راست دنبال می‌شود
    - زیررشته w فقط شامل نشانه‌های پایانی است

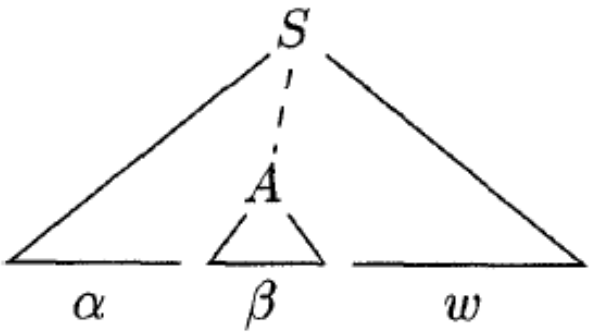
$$S \underset{rm}{\overset{*}{\Rightarrow}} \alpha A w \underset{rm}{\Rightarrow} \alpha \beta w$$

- آیا به محض مشاهده بدنه یک قاعده تولید می‌توان آن را کاهش داد؟
- دو تصمیم مهم در هر مرحله از تجزیه
  - چه هنگام کاهش صورت گیرد؟
  - کاهش با توجه به کدام قاعده تولید انجام شود؟

تجزیه پایین به بالا

- دستگیره (handle)
  - زیررشته‌ای که با بدنه یک قاعده تولید تطابق داشته و در مکانی از رشته باشد که کاهش آن یک گام معکوس در یک اشتقاق راست معتبر باشد

$$S \underset{rm}{\overset{*}{\Rightarrow}} \alpha A w \underset{rm}{\Rightarrow} \alpha \beta w$$



- کاهش هر بدنه قاعده تولیدی لزوماً منجر به تجزیه صحیح ورودی نخواهد شد
- در روند تجزیه باید فقط دستگیره‌ها را کاهش داد
  - امکان کاهش‌های بعدی تا رسیدن به نشانه شروع گرامر وجود دارد

تجزیه پایین به بالا

- تجزیه به صورت هرس دستگیره‌ها (handle pruning)
  - شناسایی و کاهش دستگیره‌ها تا رسیدن به نشانه شروع گرامر

$$S \Rightarrow \cdots \Rightarrow \alpha A_i u \Rightarrow \alpha\beta u \Rightarrow \cdots \Rightarrow y A_1 w \Rightarrow yzw = x$$

    - هر بار تعدادی ورودی تا تشکیل یک دستگیره خوانده می‌شود
    - ورودی همیشه از چپ به راست خوانده می‌شود
  - با شناسایی یک دستگیره، آن را با استفاده از قاعده تولید مربوطه کاهش می‌دهیم
  - سایر ورودی‌های سمت راست محل کاهش هنوز بررسی نشده‌اند

- دو کنش اصلی در حین تجزیه پایین به بالا
  - انتقال (shift) و کاهش (reduce)

تحلیل دستوری

- گرامرهای مستقل از متن
- روش‌های تجزیه
- تجزیه نزول بازگشتی
- تجزیه پیشگویانه
- تجزیه انتقال کاهش
- تجزیه LR
  - SLR، CLR، LALR

تجزیه انتقال – کاهش (shift - reduce)

- راه‌برد اصلی در تجزیه پایین به بالا
- استفاده از یک **پشته** برای نگهداری نشانه‌های در دست پردازش
  - حاوی نشانه‌هایی پایانی و غیرپایانی که تاکنون در روند تجزیه مشاهده شده
  - در هر انتقال یک ورودی خوانده شده و به بالای پشته اضافه می‌شود
  - در هر کاهش دستگیره تشکیل شده در بالا پشته با نشانه غیرپایانی سمت چپ قاعده تولید مربوطه جایگزین می‌شود
- دستگیره‌ها همیشه در بالای پشته قرار دارند
  - تجزیه‌گر هیچگاه مجبور به بررسی درون پشته نیست
  - دستگیره هیچگاه پایین‌تر از سمت راست‌ترین نشانه غیرپایانی در پشته نیست

تجزیه انتقال - کاهش

- مثال: تجزیه انتقال – کاهش برای رشته **int * int + int**

$E \rightarrow T + E \mid T$

$T \rightarrow int * T \mid int \mid (E)$

|int * int + int	shift

int * int + int

↑

تجزیه انتقال - کاهش

- مثال: تجزیه انتقال – کاهش برای رشته **int * int + int**

$E \rightarrow T + E \mid T$

$T \rightarrow int * T \mid int \mid (E)$

|int * int + int	shift
int | * int + int	shift

int ↑ * int + int

تجزیه انتقال - کاهش

- مثال: تجزیه انتقال – کاهش برای رشته **int * int + int**

$E \rightarrow T + E \mid T$

$T \rightarrow int * T \mid int \mid (E)$

|int * int + int	shift
int | * int + int	shift
int * | int + int	shift

int * int + int

↑

تجزیه انتقال - کاهش

- مثال: تجزیه انتقال – کاهش برای رشته **int * int + int**

$E \rightarrow T + E \mid T$

$T \rightarrow int * T \mid int \mid (E)$

|int * int + int	shift
int | * int + int	shift
int * | int + int	shift
int * int | + int	reduce $T \rightarrow int$

int * int + int

↑

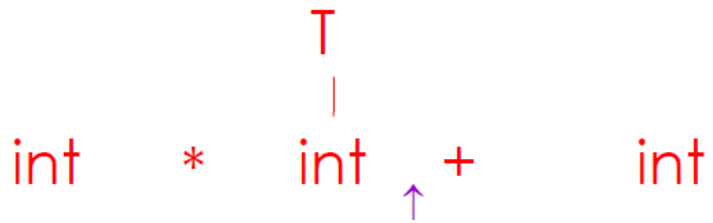
تجزیه انتقال - کاهش

- مثال: تجزیه انتقال – کاهش برای رشته **int * int + int**

$E \rightarrow T + E \mid T$

$T \rightarrow int * T \mid int \mid (E)$

|int * int + int	shift
int | * int + int	shift
int * | int + int	shift
int * int | + int	reduce $T \rightarrow int$
int * T | + int	reduce $T \rightarrow int * T$



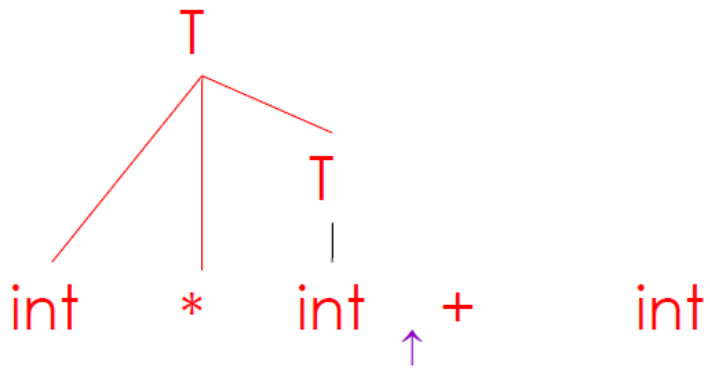
تجزیه انتقال - کاهش

- مثال: تجزیه انتقال – کاهش برای رشته **int * int + int**

$E \rightarrow T + E \mid T$

$T \rightarrow int * T \mid int \mid (E)$

|int * int + int	shift
int | * int + int	shift
int * | int + int	shift
int * int | + int	reduce $T \rightarrow int$
int * T | + int	reduce $T \rightarrow int * T$
T | + int	shift



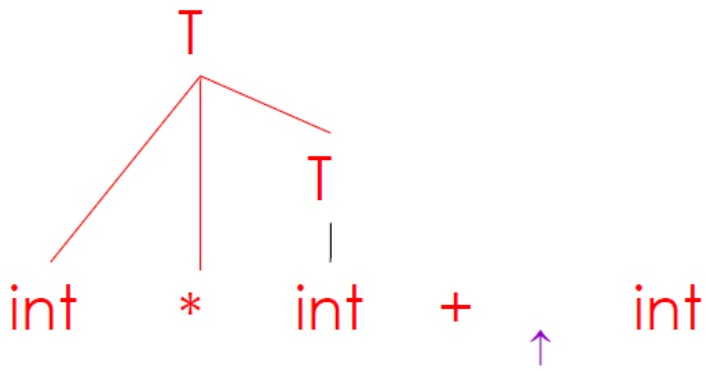
تجزیه انتقال - کاهش

- مثال: تجزیه انتقال – کاهش برای رشته **int * int + int**

$E \rightarrow T + E \mid T$

$T \rightarrow int * T \mid int \mid (E)$

|int * int + int	shift
int | * int + int	shift
int * | int + int	shift
int * int | + int	reduce $T \rightarrow int$
int * T | + int	reduce $T \rightarrow int * T$
T | + int	shift
T + | int	shift



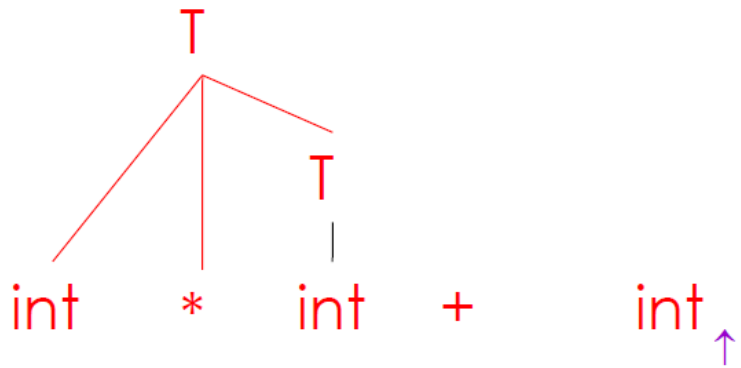
تجزیه انتقال - کاهش

- مثال: تجزیه انتقال – کاهش برای رشته **int * int + int**

$E \rightarrow T + E \mid T$

$T \rightarrow int * T \mid int \mid (E)$

|int * int + int	shift
int | * int + int	shift
int * | int + int	shift
int * int | + int	reduce $T \rightarrow int$
int * T | + int	reduce $T \rightarrow int * T$
T | + int	shift
T + | int	shift
T + int |	reduce $T \rightarrow int$



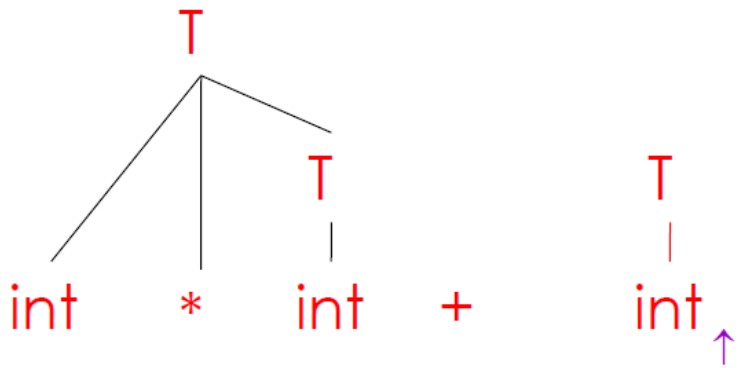
تجزیه انتقال - کاهش

- مثال: تجزیه انتقال – کاهش برای رشته **int * int + int**

$E \rightarrow T + E \mid T$

$T \rightarrow int * T \mid int \mid (E)$

|int * int + int	shift
int | * int + int	shift
int * | int + int	shift
int * int | + int	reduce $T \rightarrow int$
int * T | + int	reduce $T \rightarrow int * T$
T | + int	shift
T + | int	shift
T + int |	reduce $T \rightarrow int$
T + T |	reduce $E \rightarrow T$



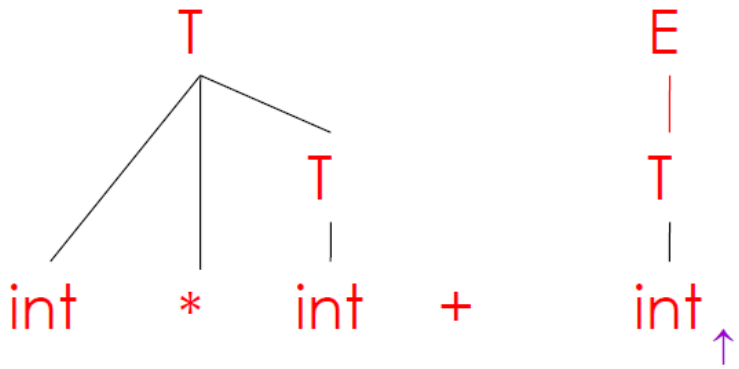
تجزیه انتقال - کاهش

- مثال: تجزیه انتقال – کاهش برای رشته **int * int + int**

$E \rightarrow T + E \mid T$

$T \rightarrow int * T \mid int \mid (E)$

|int * int + int	shift
int | * int + int	shift
int * | int + int	shift
int * int | + int	reduce $T \rightarrow int$
int * T | + int	reduce $T \rightarrow int * T$
T | + int	shift
T + | int	shift
T + int |	reduce $T \rightarrow int$
T + T |	reduce $E \rightarrow T$
T + E |	reduce $E \rightarrow T + E$



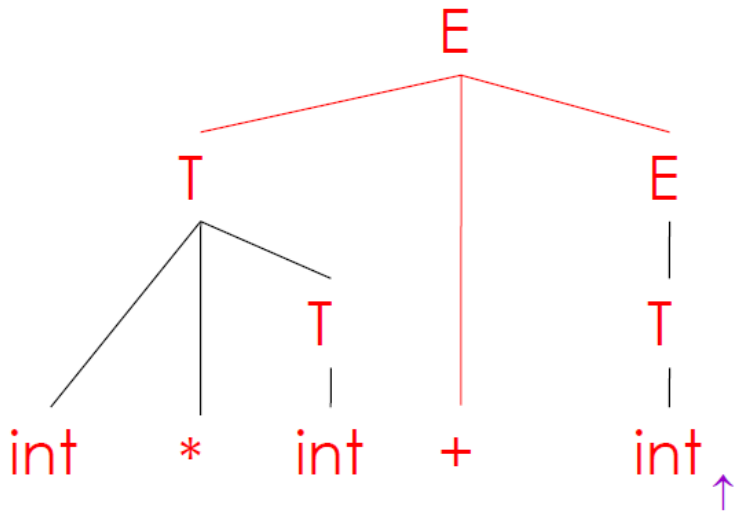
تجزیه انتقال - کاهش

- مثال: تجزیه انتقال – کاهش برای رشته **int * int + int**

$E \rightarrow T + E \mid T$

$T \rightarrow int * T \mid int \mid (E)$

|int * int + int	shift
int | * int + int	shift
int * | int + int	shift
int * int | + int	reduce $T \rightarrow int$
int * T | + int	reduce $T \rightarrow int * T$
T | + int	shift
T + | int	shift
T + int |	reduce $T \rightarrow int$
T + T |	reduce $E \rightarrow T$
T + E |	reduce $E \rightarrow T + E$
E |	accept



تجزیه انتقال - کاهش

- سوال: با توجه به گرامر داده شده، دستگیره شناسایی شده توسط تجزیه‌گر در وضعیت زیر کدام است؟

$E \rightarrow E' \mid E' + E$

$E' \rightarrow -E' \mid id \mid (E)$

$E' + -id \mid + -(id + id)$

- $E' + -id$
- id
- $-id$
- $E' + -E'$

تجزیه انتقال - کاهش

- سوال: کدامیک از گزینه‌های زیر تجزیه انتقال – کاهش صحیح رشته **id + –id** را برای گرامر داده شده نشان می‌دهد؟

$E \rightarrow E' \mid E' + E$

$E' \rightarrow -E' \mid id \mid (E)$

○ |id + -id
id|+ -id
E' +|-id
E' + -|id
E' + -id|
E' + -E'|
E' + E'|
E' + E|
E|

○ |id + -id
id|+ -id
id +|-id
id + -|id
id + -id|
id + -E'|
id + E'|
id + E|
E' + E|
E|

○ |id + -id
|E' + -id
E'|+ -id
E' +|-id
E' + -|id
E' + -|E'
E' +|-E'
E' +|E'
E' +|E
E'|+ E
|E' + E
|E

○ |id + -id
id|+ -id
E'|+ -id
E' +|-id
E' + -|id
E' + -id|
E' + -E'|
E' + E'|
E' + E|
E|

تجزیه انتقال - کاهش

- برخورد (conflict) در تجزیه برای برخی از گرامرها
  - امکان انجام بیش از یک کنش صحیح در یک وضعیت خاص از تجزیه‌گر
  - تجزیه‌گر عملکرد قطعی (معین) نخواهد داشت
    - عدم امکان بکارگیری تجزیه پایین به بالا برای چنین گرامری
  - برخورد انتقال/کاهش (shift/reduce)
    - وضعیتی در تجزیه که هم انتقال ورودی به پشته و هم کاهش دستگیره تشکیل شده در بالای پشته ممکن باشد
    - مثال: برخورد برای گرامر else آویزان

$$
\begin{array}{lcl}
stmt & \rightarrow & \textbf{if } expr \textbf{ then } stmt \\
 & | & \textbf{if } expr \textbf{ then } stmt \textbf{ else } stmt \\
 & | & \textbf{other}
\end{array}
$$

    - امکان استفاده از برخی راه‌کارها برای حل این نوع برخورد
      - مثلاً قوانین ابهام‌زدا

تجزیه انتقال - کاهش

- برخورد در تجزیه (ادامه)
  - برخورد کاهش/کاهش (reduce/reduce)
    - وجود وضعیتی در تجزیه که همزمان دو دستگیره متفاوت در بالای پشته شناسایی شود
      - امکان کاهش با بیش از یک قاعده تولید وجود دارد
    - مثال: برخورد در تجزیه رشته **id (id , id)** با گرامر زیر

$$
\begin{array}{rcl}
stmt & \rightarrow & \mathbf{id}\ (\ parameter\_list\ ) \mid expr := expr \\
parameter\_list & \rightarrow & parameter\_list\ ,\ parameter \mid parameter \\
parameter & \rightarrow & \mathbf{id} \\
expr & \rightarrow & \mathbf{id}\ (\ expr\_list\ ) \mid \mathbf{id} \\
expr\_list & \rightarrow & expr\_list\ ,\ expr \mid expr
\end{array}
$$

- حل این نوع برخورد در هنگام تجزیه دشوار است

تجزیه انتقال - کاهش

- پیشوندهای ممکن (viable prefixes)
  - اگر رشته $\beta_1\beta_2 w$ یک وضعیت معتبر در تجزیه انتقال – کاهش باشد، آنگاه رشته β_1 یک پیشوند ممکن است
    - وضعیت معتبر: وجود یک اشتقاق راست از نشانه شروع گرامر به این وضعیت
    - $\beta_1\beta_2$ محتوای پشته تجزیه‌گر و w باقیمانده ورودی است
  - *پیشوندهای ممکن در حقیقت پیشوند یک دستگیره هستند*
    - حداکثر می‌تواند شامل کل دستگیره شود
  - مثال: در تجزیه رشته **(int)** با گرامر روبرو

$E \rightarrow T + E \mid T$
$T \rightarrow \text{int} * T \mid \text{int} \mid (E)$

    - (E| یک وضعیت معتبر و (E پیشوند ممکن آن است
      - $T \rightarrow (E)$ پیشوندی از دستگیره

تجزیه انتقال - کاهش

- راهکار: پشته در تجزیه صحیح باید فقط شامل پیشوندهای ممکن باشد
  - در اکثر مواقع در حین تجزیه فقط بخشی از بدنه قواعد تولید در پشته است
  - بدنه قواعد تولید در روند تجزیه به صورت بازگشتی کامل می‌شود
    - محتوای پشته در هر زمان از تجزیه شامل دنباله‌ای از پیشوندهای بدنه قواعد تولید است

$$\beta_1\beta_2\cdots\beta_i\cdots\beta_{n-1}\beta_n \mid w$$

    - اگر β_i پیشوند ممکن از دستگیره $A_i \rightarrow \alpha_i$ باشد
      - پس از تعدادی کنش انتقال و کاهش، نهایتاً β_i با A_i جایگزین خواهد شد
        - پیشوندهای ممکن $\beta_{i+1}\cdots\beta_n$ نهایتاً تکمیل کننده دستگیره $A_i \rightarrow \alpha_i$ هستند
    - تکمیل پیشوند ممکن قبلی (β_{i-1}) از A_i ادامه پیدا می‌کند تا دستگیره بعدی تشکیل شود: $A_{i-1} \rightarrow \alpha_{i-1} = \beta_{i-1}A_i\gamma$

تجزیه انتقال - کاهش

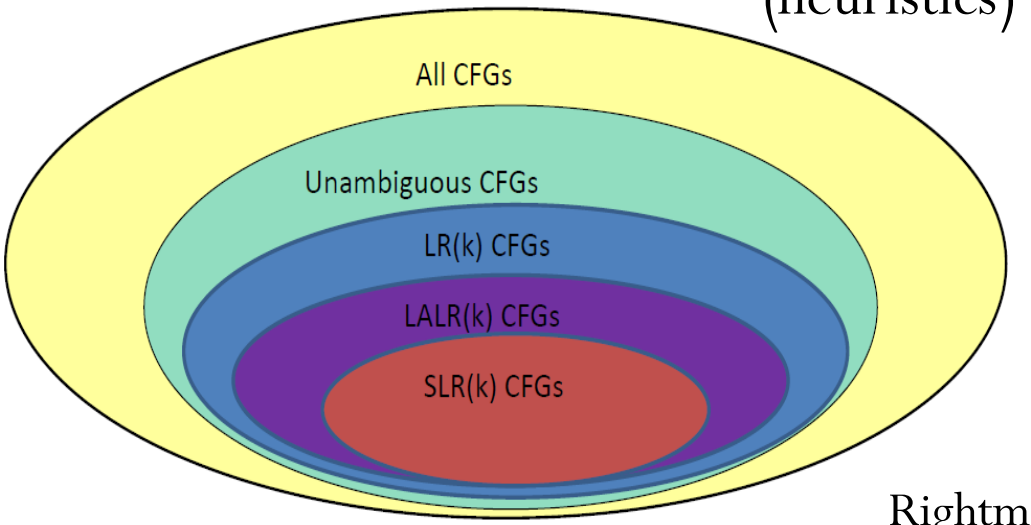
- ساختار محتوای پشته
  - مثال: اگر در تجزیه رشته **(int * int)** با گرامر روبرو در وضعیت (int * |int) باشیم:
    - "ϵ" پیشوند ممکن از دستگیره $E \rightarrow T$
    - "(" پیشوند ممکن از دستگیره $T \rightarrow (E)$
    - "ϵ" پیشوند ممکن از دستگیره $E \rightarrow T$
    - "**int ***" پیشوند ممکن از دستگیره $T \rightarrow int * T$
    - ...

$E \rightarrow T + E \mid T$
$T \rightarrow int * T \mid int \mid (E)$

- برای اطمینان از شناسایی صحیح دستگیره‌ها باید بررسی کرد که محتوای پشته تجزیه‌گر همیشه شامل پیشوندهای ممکن باشد

تجزیه انتقال - کاهش

- شناسایی دستگیره‌ها
  - عدم وجود الگوریتم‌های کارآمد در حالت کلی برای هر گرامر دلخواه
  - امکان شناسایی صحیح دستگیره‌ها برای برخی از گرامرهای مستقل از متن با استفاده از روش‌های ابتکاری (heuristics)



- امکان تجزیه معین
  - تجزیه LR
- گرامرهای LR(k)
  - Left to right scan
  - Rightmost derivation in reverse
  - **k** lookahead input symbol
- در عمل معمولاً $k = 1$ (LR(1))

تحلیل دستوری

- گرامرهای مستقل از متن
- روش‌های تجزیه
- تجزیه نزول بازگشتی
- تجزیه پیشگویانه
- تجزیه انتقال کاهش
- تجزیه LR
  - SLR، CLR، LALR

تجزیه LR

تجزیه LR

- برای هر گرامر مستقل از متن، مجموعه پیشوندهای ممکن در حین تجزیه انتقال – کاهش یک زبان منظم است
  - امکان ساخت یک ماشین متناهی برای بررسی محتوای پشته تجزیه‌گر
    - ماشین LR(0)
- موارد (items) LR(0) یک قاعده تولید از گرامر
  - قواعد تولید حاوی نقطه (“.”) در هر جای ممکن از بدنه آن قاعده تولید

$$A \to XYZ \quad \Rightarrow \quad \begin{array}{l} A \to \cdot XYZ \\ A \to X \cdot YZ \\ A \to XY \cdot Z \\ A \to XYZ \cdot \end{array}$$

- نشان دهنده بخش‌های ممکن دیده شده و مورد انتظار از بدنه آن قاعده تولید

تجزیه LR

- موارد LR(0)
  - مثال: اگر در تجزیه رشته **(int * int)** برای گرامر روبرو در وضعیت (int * |int) باشیم:

$E \rightarrow T + E \mid T$
$T \rightarrow int * T \mid int \mid (E)$

    - هیچ (“ϵ”) بخشی از قاعده تولید $E \rightarrow T$ دیده نشده است: $E \rightarrow .T$
    - بخش ”(“ از قاعده تولید $T \rightarrow (E)$ دیده شده است: $T \rightarrow (.E)$
    - هیچ (“ϵ”) بخشی از قاعده تولید $E \rightarrow T$ دیده نشده است: $E \rightarrow .T$
    - بخش ”int *“ از قاعده تولید $T \rightarrow int * T$ دیده شده است: $T \rightarrow int * .T$
    - ...
  - هر حالت ماشین LR(0) مجموعه‌ای از موارد LR(0) است
    - پشته تجزیه‌گر شامل مجموعه‌های موارد LR(0) است

تجزیه LR

- مراحل ساخت ماشین LR(0) برای گرامر G (نوع معین – DFA)
  - بدست آوردن گرامر مضاعف (augmented) از روی G
  - محاسبه بسته استاندارد (canonical collection) مجموعه‌های موارد LR(0)
    - تعیین کننده حالت‌ها و انتقال‌های ماشین LR(0)
      - مجموعه‌های موارد LR(0) حالت‌های ماشین هستند
      - ماشین در هر حالت روی نشانه‌های گرامر (پایانی و غیرپایانی) انتقال دارد
  - تمام حالت‌های ماشین نهایی هستند
  - فقط یک حالت اولیه
    - حاوی نشانه شروع گرامر مضاعف
- امکان ساخت ماشین نامعین (NFA)

ماشین LR(0)

- بدست آوردن گرامر مضاعف
  - افزودن قاعده تولید $S' \longrightarrow S$ به مجموعه قواعد تولید گرامر
    - S نشانه شروع گرامر اصلی است
    - S' یک نشانه غیرپایانی جدید است
    - نشانه شروع گرامر مضاعف است
  - هدف تشخیص زمان توقف موفقیت‌آمیز روند تجزیه است
    - هنگامی که قصد کاهش نشانه شروع گرامر اصلی (S) را داشته باشیم
- محاسبه بسته استاندارد مجموعه‌های موارد LR(0)
  - استفاده از دو عملگر اصلی
    - بستار (Closure) یک مجموعه از موارد
    - انتقال (Goto) از یک مجموعه از موارد با یک نشانه گرامر

ماشین LR(0)

- بستار یک مجموعه موارد (I): Closure(I)
  - ساخت افزایشی تا هنگامی که مورد جدیدی قابل اضافه نباشد
    - در ابتدا تمام موارد LR(0) عضو مجموعه I در بستار آن هستند
    - اگر مورد $A \rightarrow \alpha.B\beta$ عضو مجموعه I باشد:
      - برای تمام قواعد تولید $B \rightarrow \gamma_i$، مورد $B \rightarrow .\gamma_i$ به بستار اضافه می‌شود

- سوال: برای گرامر زیر بستار مجموعه $\{E' \rightarrow .E\}$ را محاسبه کنید.

$$
\begin{array}{lcl}
E' & \rightarrow & E \\
E & \rightarrow & E + T \mid T \\
T & \rightarrow & T * F \mid F \\
F & \rightarrow & (E) \mid \mathbf{id}
\end{array}
$$

ماشین LR(0)

- بستار یک مجموعه موارد (ادامه)
  - دسته‌بندی موارد در بستار
    - موارد اصلی (kernel items)
      - مورد $S' \rightarrow .S$ و تمام مواردی که نقطه در انتهای سمت چپ آنها قرار **ندارد**
    - موارد غیراصلی (nonkernel items)
      - تمام مواردی که نقطه در انتهای سمت چپ آنها قرار دارد به جز $S' \rightarrow .S$
  - در روند محاسبه بستار فقط موارد غیراصلی به بستار یک مجموعه از موارد اضافه می‌شود
  - شبیه به محاسبه بستار ε برای مجموعه حالت‌های NFA
    - در حال ساختن یک DFA هستیم

ماشین LR(0)

- انتقال از یک مجموعه موارد (I) با نشانه X: Goto(I, X)
  - اگر مجموعه موارد I شامل مواردی به شکل $A \longrightarrow \alpha.X\beta$ باشد، انتقال از I با نشانه X برابر با بستار مجموعه تمام موارد $A \longrightarrow \alpha X.\beta$ است
  - اگر مجموعه موارد I هیچ موردی به شکل $A \longrightarrow \alpha.X\beta$ نداشته باشد، انتقالی برای نشانه X نخواهد داشت
  - انتقال‌های ماشین LR(0) با نشانه‌های گرامر را محاسبه می‌کند

- سوال: برای گرامر زیر انتقال از مجموعه موارد $\begin{Bmatrix} E' \rightarrow .E \\ E \rightarrow .E + T \end{Bmatrix}$ با نشانه "E" را محاسبه کنید.

$$\begin{array}{lcl} E' & \rightarrow & E \\ E & \rightarrow & E+T \mid T \\ T & \rightarrow & T*F \mid F \\ F & \rightarrow & (E) \mid \mathbf{id} \end{array}$$

ماشین LR(0)

- الگوریتم محاسبه بسته استاندارد مجموعه‌های موارد LR(0)

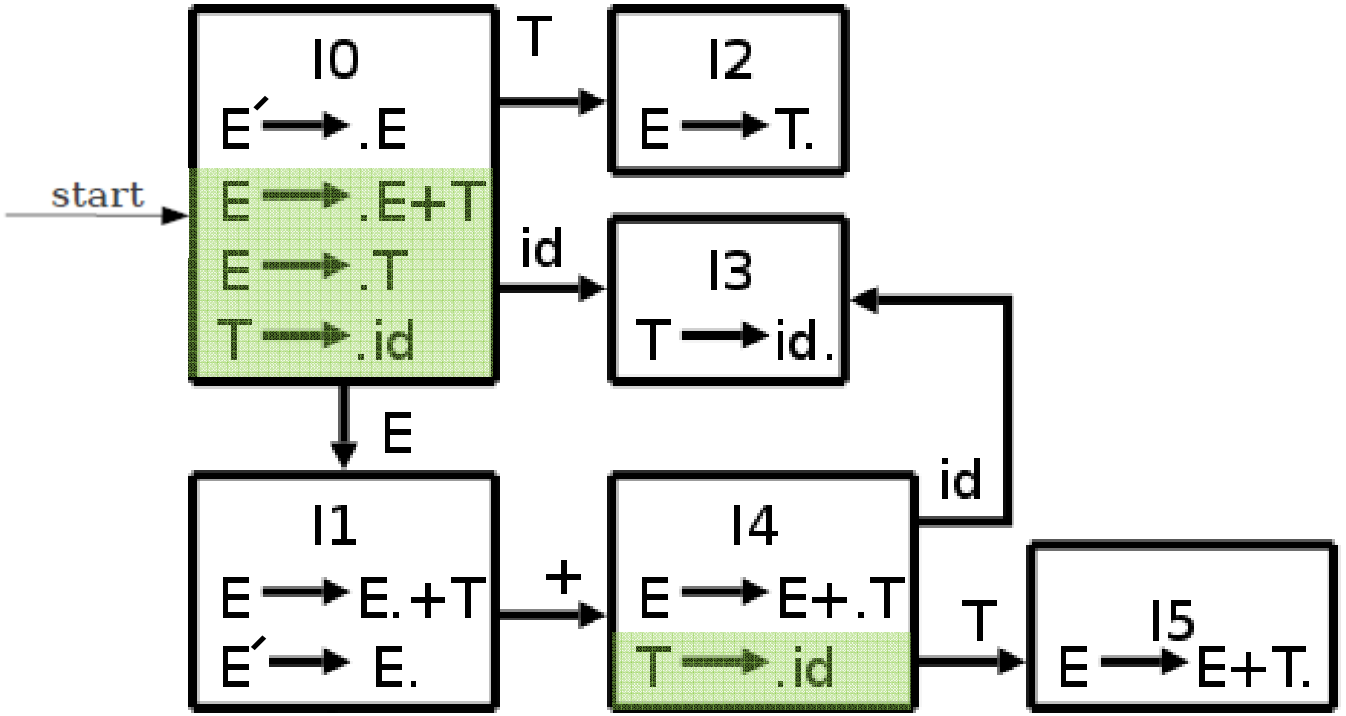
```
void items(G') {                                  ← ورودی الگوریتم گرامر مضاعف
    C = CLOSURE({[S' → ·S]});                     ← حالت اولیه ماشین
    repeat
        for ( each set of items I in C )
            for ( each grammar symbol X )
                if ( GOTO(I, X) is not empty and not in C )
                    add GOTO(I, X) to C;
    until no new sets of items are added to C on a round;
}
```

- با محاسبه بسته استاندارد می‌توان ماشین LR(0) را ساخت
  - مجموعه‌های موارد حالت‌ها هستند
  - انتقال‌های مختلف برای هر مجموعه موارد توسط یالها نشان داده می‌شود

ماشین LR(0)

- مثال: ماشین LR(0) برای گرامر روبرو

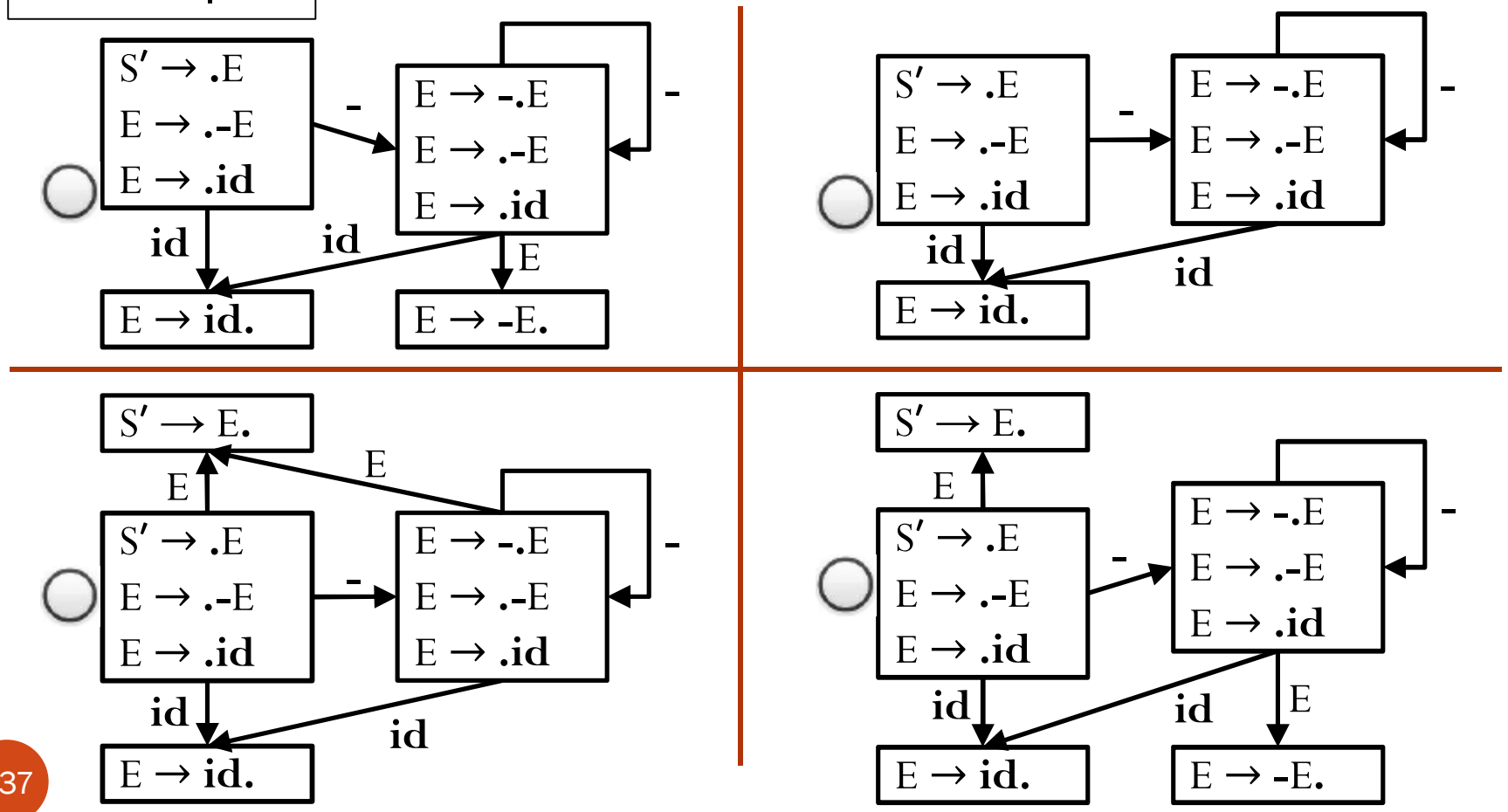
$$
\begin{array}{lcl}
E' & \rightarrow & E \\
E & \rightarrow & E + T \mid T \\
T & \rightarrow & \mathbf{id}
\end{array}
$$



ماشین LR(0)

- سوال: ماشین LR(0) برای گرامر روبرو کدام است؟

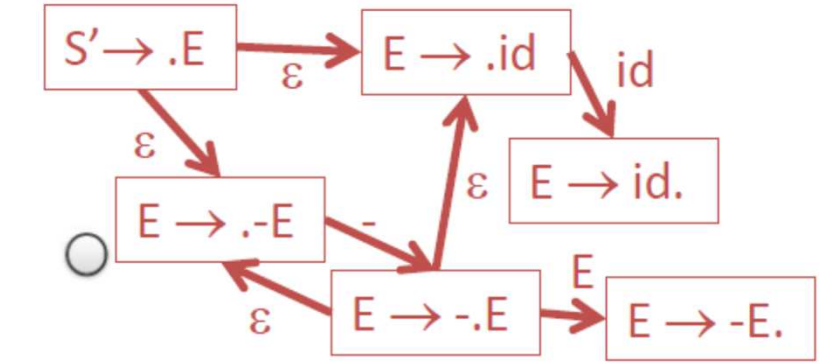
$S' \rightarrow E$
$E \rightarrow -E \mid \mathbf{id}$

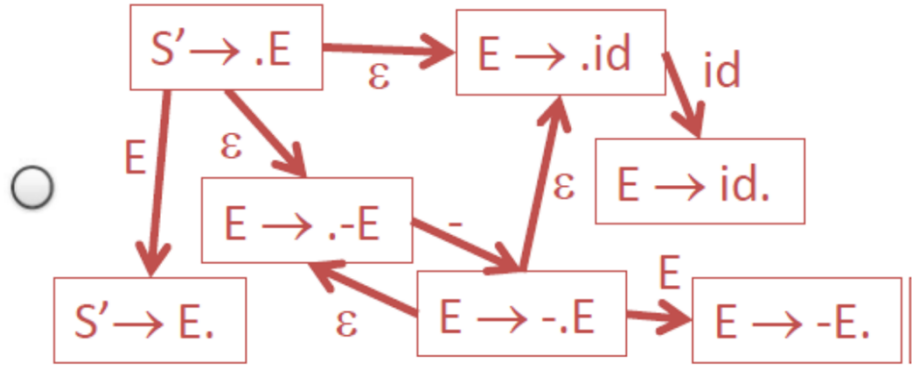


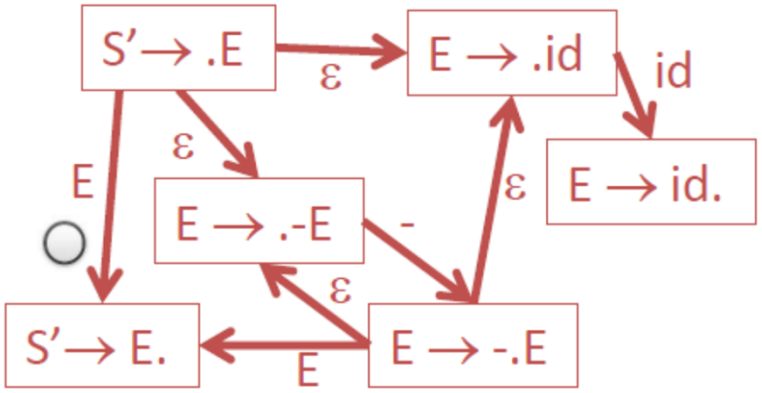
ماشین LR(0)

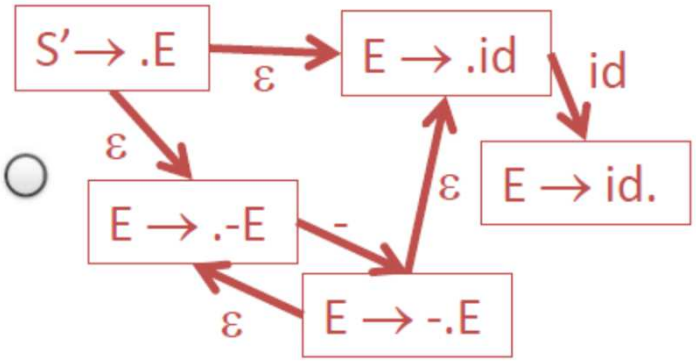
$S' \rightarrow E$
$E \rightarrow -E \mid \mathbf{id}$

- سوال: ماشین LR(0) برای گرامر روبرو کدام است؟







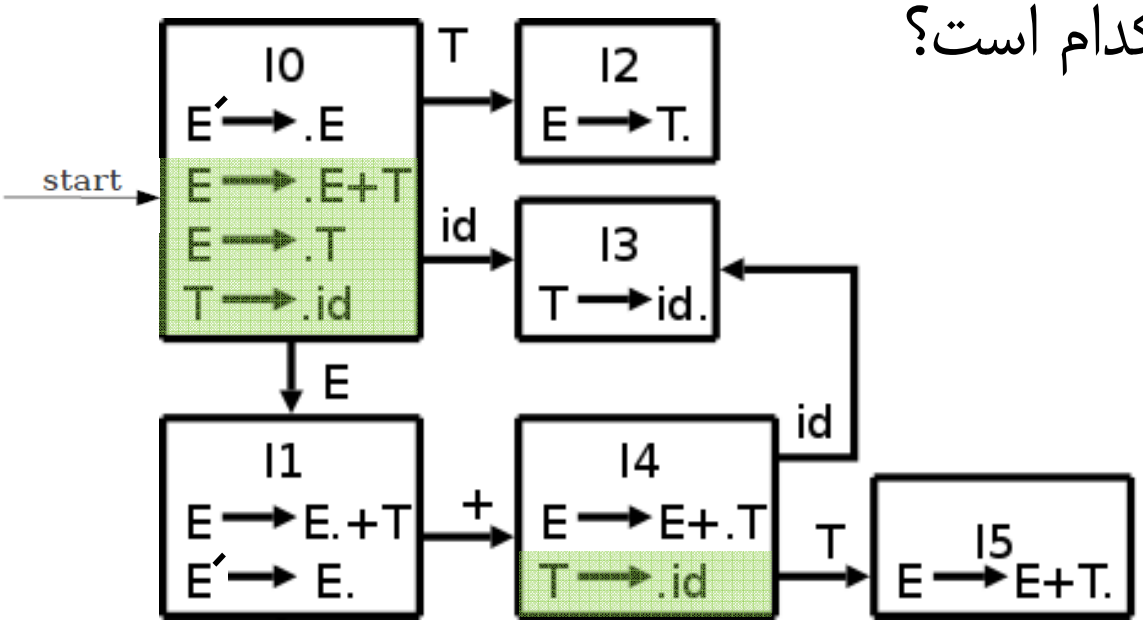


ماشین LR(0)

- موارد معتبر (valid) برای یک پیشوند ممکن
  - $A \rightarrow \beta.\gamma$ یک مورد معتبر برای پیشوند ممکن $\alpha\beta$ است اگر اشتقاق راست $S' \overset{*}{\Rightarrow} \alpha A w \Rightarrow \alpha\beta\gamma w$ وجود داشته باشد
  - یک مورد ممکن است برای چندین پیشوند ممکن معتبر باشد
    - مثال: مورد $T \rightarrow (.E)$ برای هر دنباله‌ای از "(" معتبر است: (، ((، (((، ...
  - مجموعه موارد معتبر برای پیشوند ممکن δ دقیقاً مجموعه مواردی است که با پیمایش مسیر δ روی ماشین LR(0) با شروع از حالت اولیه بدست می‌آید
    - ماشین LR(0) با پذیرش δ در این مجموعه موارد متوقف می‌شود
  - *امکان قرار دادن مجموعه‌های موارد معتبر بجای پیشوندهای ممکن در پشته*
    - پس از بررسی $\alpha\beta$ در حین تجزیه فقط موارد معتبر در بالای پشته هستند

ماشین LR(0)

- سوال: مجموعه موارد معتبر برای پیشوند ممکن "E +" با توجه به ماشین LR(0) زیر کدام است؟

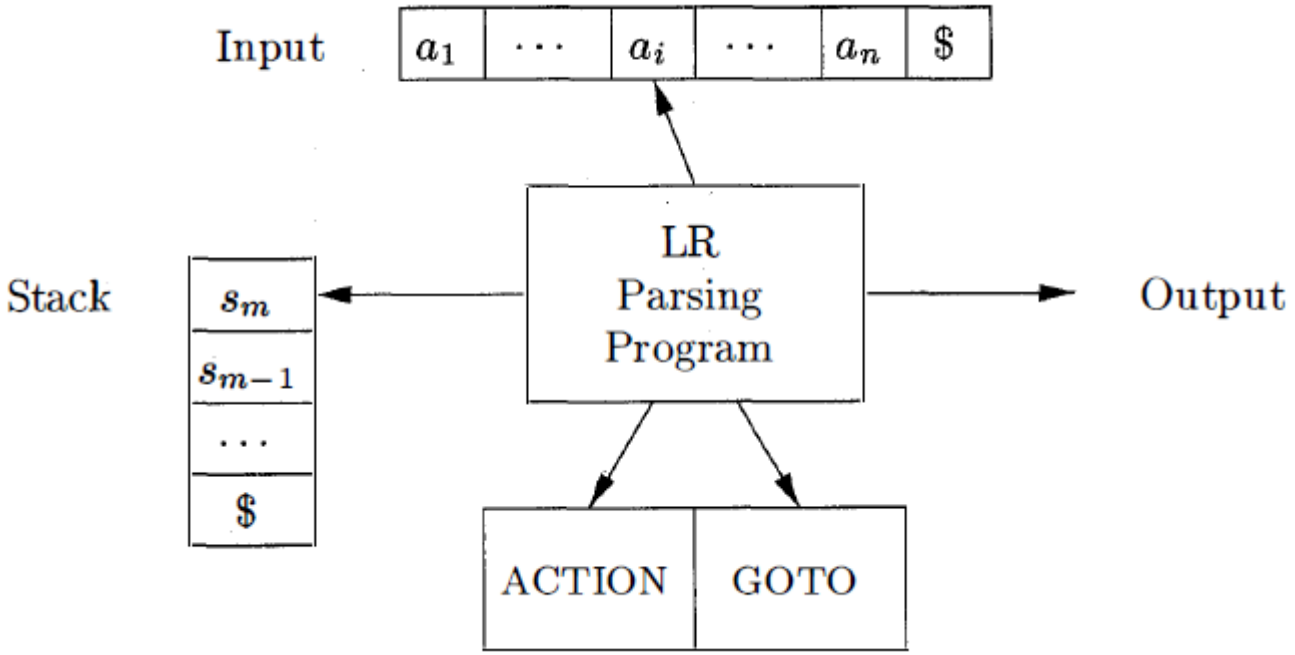


- سوال: ماشین LR(0) برای گرامر روبرو؟

$$E \rightarrow T + E \mid T$$
$$T \rightarrow int * T \mid int \mid (E)$$

تجزیه LR

- پیاده‌سازی تجزیه‌گرهای LR با استفاده از جدول تجزیه



- پشته یک دنباله از حالت‌های ماشین پذیرنده را نگهداری می‌کند
  - هر حالت دقیقاً متناظر با یک نشانه از گرامر است

تجزیه LR

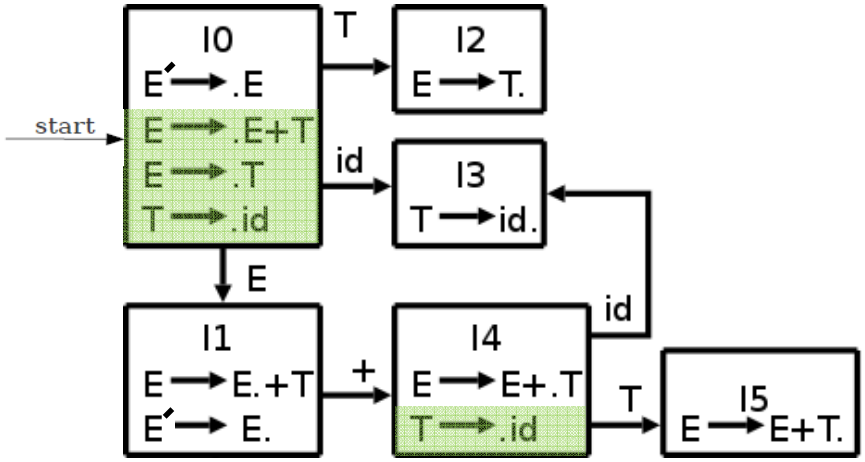
- استفاده از ماشین پذیرنده برای بررسی اعتبار پشته
  - ماشین پذیرنده روند تجزیه را هدایت می‌کند
- تفاوت اصلی گونه‌های مختلف الگوریتم تجزیه LR در جدول تجزیه
- جدول تجزیه LR دارای دو بخش اصلی
  - کنش‌ها در هر حالت (Action)
    - انتقال، کاهش، پذیرش و خطا
  - انتقال بین حالت‌ها (Goto)
    - نشان‌دهنده مدل انتقال ماشین پذیرنده
    - در هر حالت با هر نشانه گرامر به چه حالت دیگری می‌رویم
  - به ازای هر حالت یک سطر در جدول تجزیه وجود دارد

تجزیه LR

- ساده‌ترین الگوریتم تجزیه LR: تجزیه LR(0)
  - برای انتخاب کنش‌ها به ورودی توجهی نمی‌شود
- جدول تجزیه LR(0)
  - بخش کنش‌ها (Action[s])
    - اگر حالت s حاوی مورد $A \rightarrow \alpha.$ باشد ($A \neq S'$) آنگاه: $Action[s] = Reduce\ A \rightarrow \alpha$
    - اگر حالت s حاوی مورد $S' \rightarrow S.$ باشد آنگاه: $Action[s] = Accept$
    - اگر حالت s حاوی مورد $A \rightarrow \alpha.X\beta$ باشد آنگاه: $Action[s] = Shift$
    - اگر s حالت خطا (حالت بن‌بست) باشد آنگاه: $Action[s] = Error$
  - بخش انتقال (Goto[s, X])
    - پیاده‌سازی مدل انتقال ماشین LR(0)

تجزیه LR

- مثال: جدول تجزیه LR(0)



	Action	Goto			
		+	id	E	T
S_0	Shift		S_3	S_1	S_2
S_1	Shift	S_4			
S_2	Reduce E → T				
S_3	Reduce T → **id**				
S_4	Shift		S_3		S_5
S_5	Reduce E → E + T				

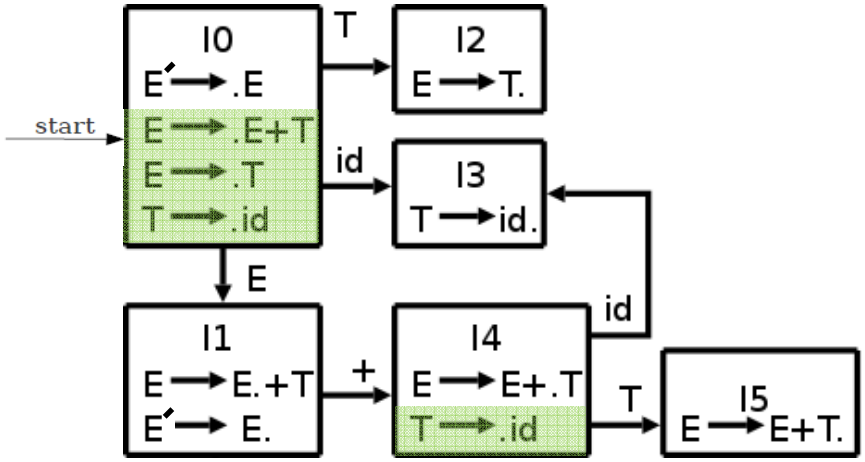
تجزیه LR

- الگوریتم تجزیه LR(0)

```
Stack.push(s0);
while (!Stack.empty())
        s = Stack.top();
        if Action[s] is Shift
                a = nextInput(); //advance input
                Stack.push(Goto[s, a]);
        elseif Action[s] is Reduce A → α
                for i from 1 to |α| do
                        Stack.pop();
                s = Stack.top();
                Stack.push(Goto[s, A]);
        elseif Action[s] is Accept
                Stack.pop();
                break;
        elseif Action[s] is Error
                error_routine();
```

تجزیه LR

- مثال: جدول تجزیه LR(0)



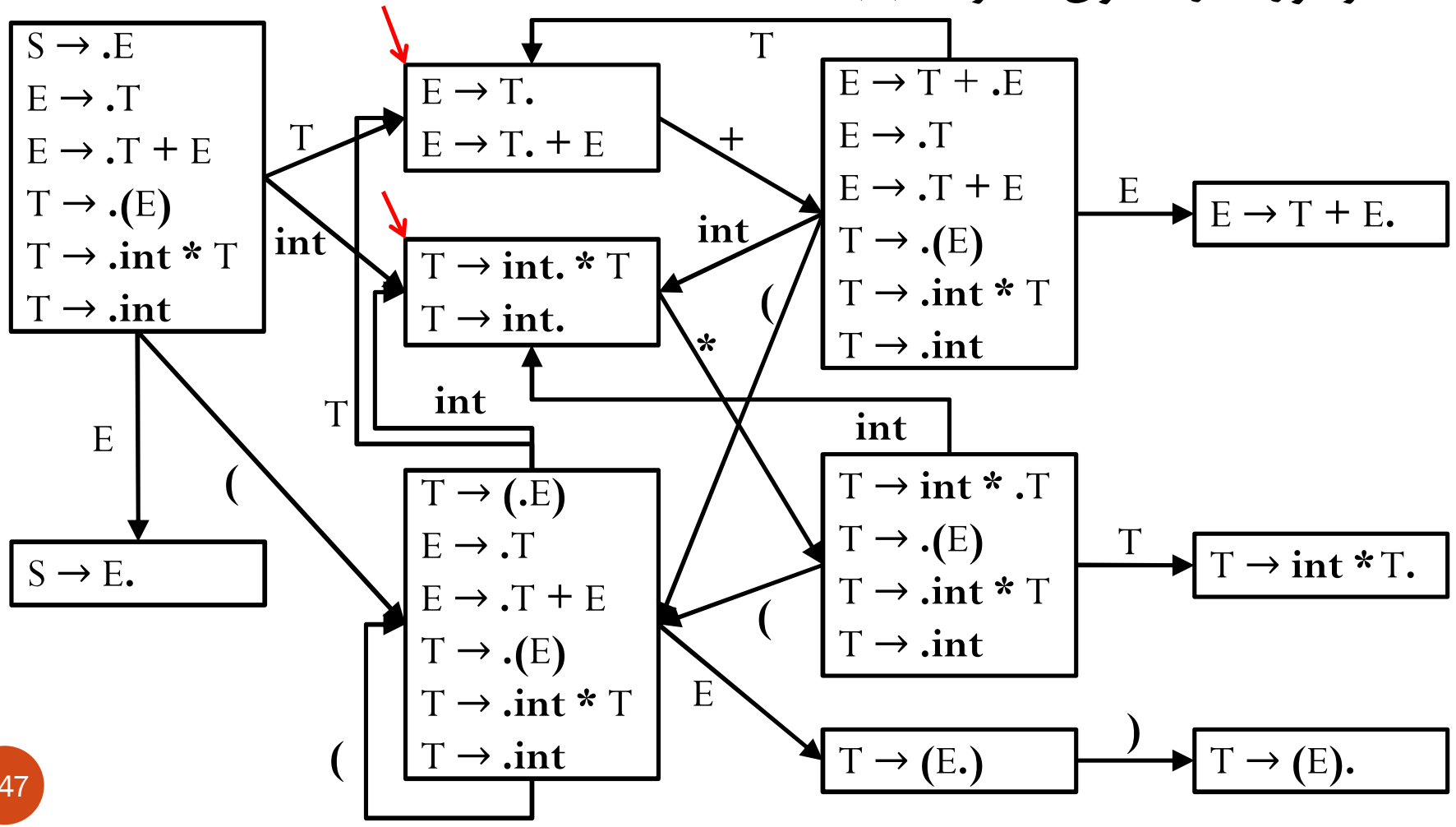
	Action	Goto			
		+	id	E	T
S_0	Shift		S_3	S_1	S_2
S_1	Shift	S_4			
S_2	Reduce E → T				
S_3	Reduce T → **id**				
S_4	Shift		S_3		S_5
S_5	Reduce E → E + T				

تجزیه LR

$E \rightarrow T + E \mid T$

$T \rightarrow int * T \mid int \mid (E)$

- برخورد در جدول تجزیه LR(0)



تجزیه LR

- تجزیه SLR (Simple LR)
  - بهبود تجزیه LR(0) با نگاه به ورودی برای تصمیم‌گیری در مورد کاهش
    - باعث حذف بسیاری از برخوردهای انتقال/کاهش خواهد شد
- جدول تجزیه SLR
  - بخش کنش‌ها (Action[s, **a**])
    - اگر حالت s حاوی مورد $A \rightarrow \alpha.$ باشد ($A \neq S'$) و $\mathbf{a} \in Follow(A)$ آنگاه:

$$Action[s, \mathbf{a}] = Reduce\ A \rightarrow \alpha$$

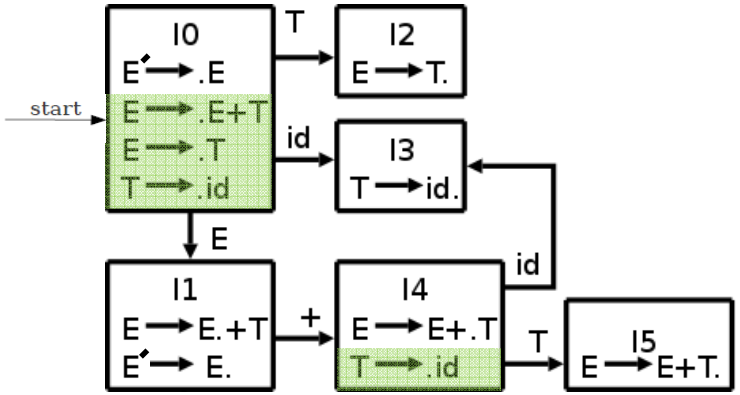
    - اگر حالت s حاوی مورد $S' \rightarrow S.$ باشد آنگاه: $Action[s, \$] = Accept$
    - اگر حالت s حاوی مورد $A \rightarrow \alpha.\mathbf{a}\beta$ باشد آنگاه: $Action[s, \mathbf{a}] = Shift$
    - اگر s حالت خطا (حالت بن‌بست) باشد آنگاه: $Action[s, :] = Error$
  - بخش انتقال (Goto[s, X]) مانند جدول تجزیه LR(0)

تجزیه LR

- امکان ساده‌سازی جدول تجزیه با ترکیب بخش کنش‌ها و انتقال
  - نشانه‌های پایانی در هر دو بخش مبنای تصمیم‌گیری است
  - مشخص کردن کنش و انتقال به صورت همزمان در یک خانه از جدول
- گرامر SLR(1)
  - گرامری که در هر خانه از بخش کنش‌های جدول SLR ساخته شده برای آن حداکثر یک کنش وجود داشته باشد
    - نباید برخوردی بین کنش‌ها وجود داشته باشد
  - گرامرهایی که با دو راهکار بکارگیری پشته و نگاه به ورودی بتوان برای آنها تجزیه معین (قطعی) انجام داد

تجزیه LR

- مثال: جدول تجزیه SLR



	Action			Goto	
	+	id	$	E	T
S_0		Shift S_3		S_1	S_2
S_1	Shift S_4		Accept		
S_2	Reduce E → T		Reduce E → T		
S_3	Reduce T → **id**		Reduce T → **id**		
S_4		Shift S_3			S_5
S_5	Reduce E → E + T		Reduce E → E + T		

تجزیه LR

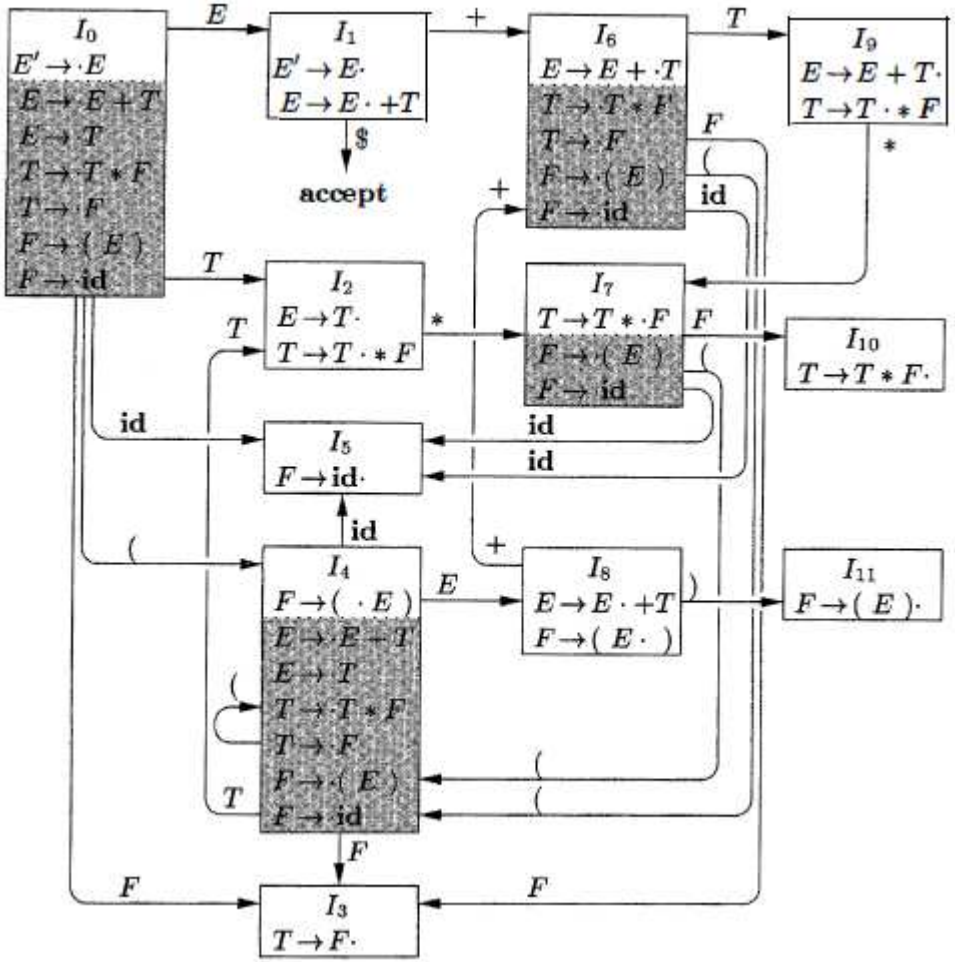
- الگوریتم کلی تجزیه LR

```
let a be the first symbol of w$;                    /*در ابتدا حالت s0 در پشته قرار داده می‌شود*/
while(1) { /* repeat forever */
        let s be the state on top of the stack;
        if ( ACTION[s, a] = shift t ) {             /*خواندن ورودی و رفتن به حالت متناظر ماشین*/
                push t onto the stack;
                let a be the next input symbol;
        } else if ( ACTION[s, a] = reduce A → β ) {
                pop |β| symbols off the stack;
                let state t now be on top of the stack;
                push GOTO[t, A] onto the stack;
                output the production A → β;        /*عملیات تکمیلی تجزیه*/
        } else if ( ACTION[s, a] = accept ) break; /* parsing is done */
        else call error-recovery routine;
}
```

تجزیه LR

- مثال: تجزیه SLR (۲)

(1) $E \to E + T$
(2) $E \to T$
(3) $T \to T * F$
(4) $T \to F$
(5) $F \to (E)$
(6) $F \to \mathbf{id}$



تجزیه LR

- مثال: تجزیه SLR (۲)

(1) $E \rightarrow E + T$
(2) $E \rightarrow T$
(3) $T \rightarrow T * F$
(4) $T \rightarrow F$
(5) $F \rightarrow (E)$
(6) $F \rightarrow \mathbf{id}$

STATE	ACTION						GOTO		
	id	+	*	(	)	$	*E*	*T*	*F*
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

تجزیه LR

- مثال: تجزیه SLR (۲)

	STACK	SYMBOLS	INPUT	ACTION
(1)	0		**id** $*$ **id** $+$ **id** \$	shift
(2)	0 5	**id**	$*$ **id** $+$ **id** \$	reduce by $F \to$ **id**
(3)	0 3	F	$*$ **id** $+$ **id** \$	reduce by $T \to F$
(4)	0 2	T	$*$ **id** $+$ **id** \$	shift
(5)	0 2 7	$T*$	**id** $+$ **id** \$	shift
(6)	0 2 7 5	$T*$ **id**	$+$ **id** \$	reduce by $F \to$ **id**
(7)	0 2 7 10	$T*F$	$+$ **id** \$	reduce by $T \to T*F$
(8)	0 2	T	$+$ **id** \$	reduce by $E \to T$
(9)	0 1	E	$+$ **id** \$	shift
(10)	0 1 6	$E+$	**id** \$	shift
(11)	0 1 6 5	$E+$ **id**	\$	reduce by $F \to$ **id**
(12)	0 1 6 3	$E+F$	\$	reduce by $T \to F$
(13)	0 1 6 9	$E+T$	\$	reduce by $E \to E+T$
(14)	0 1	E	\$	accept

تجزیه LR

- سوال: با در نظر گرفتن وضعیت زیر برای تجزیه‌گر SLR کنش بعدی را تعیین کنید؟

int * int|+ int $

$E \rightarrow T + E \mid T$
$T \rightarrow int * T \mid int \mid (E)$