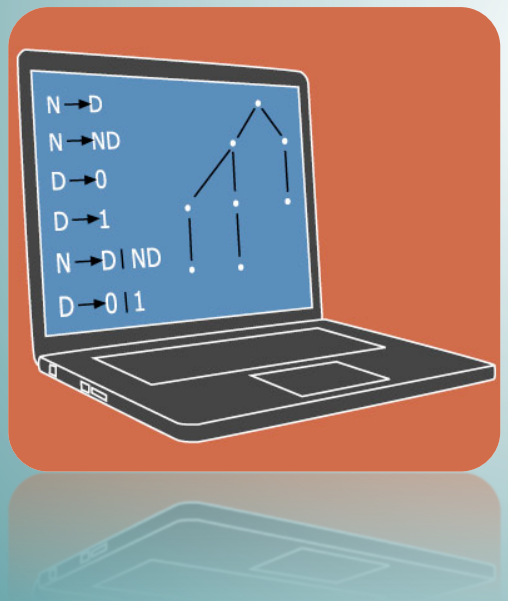
Procesadores de Lenguaje

Analizador SLR y LR canónico



Cristina Tirnauca

Domingo Gómez Pérez

Dpto. de Matemáticas,

Estadística y Computación

Ejemplo de Autómata LR(0)

Expresiones muy sencillas

```
%%
E : E + T^{1} |
    T^{2};
T : T * F^{3} |
    F^{4};
F : ( E )^{5}
  | id^{6};
```

Los items se representan solamente por el corazón del item.

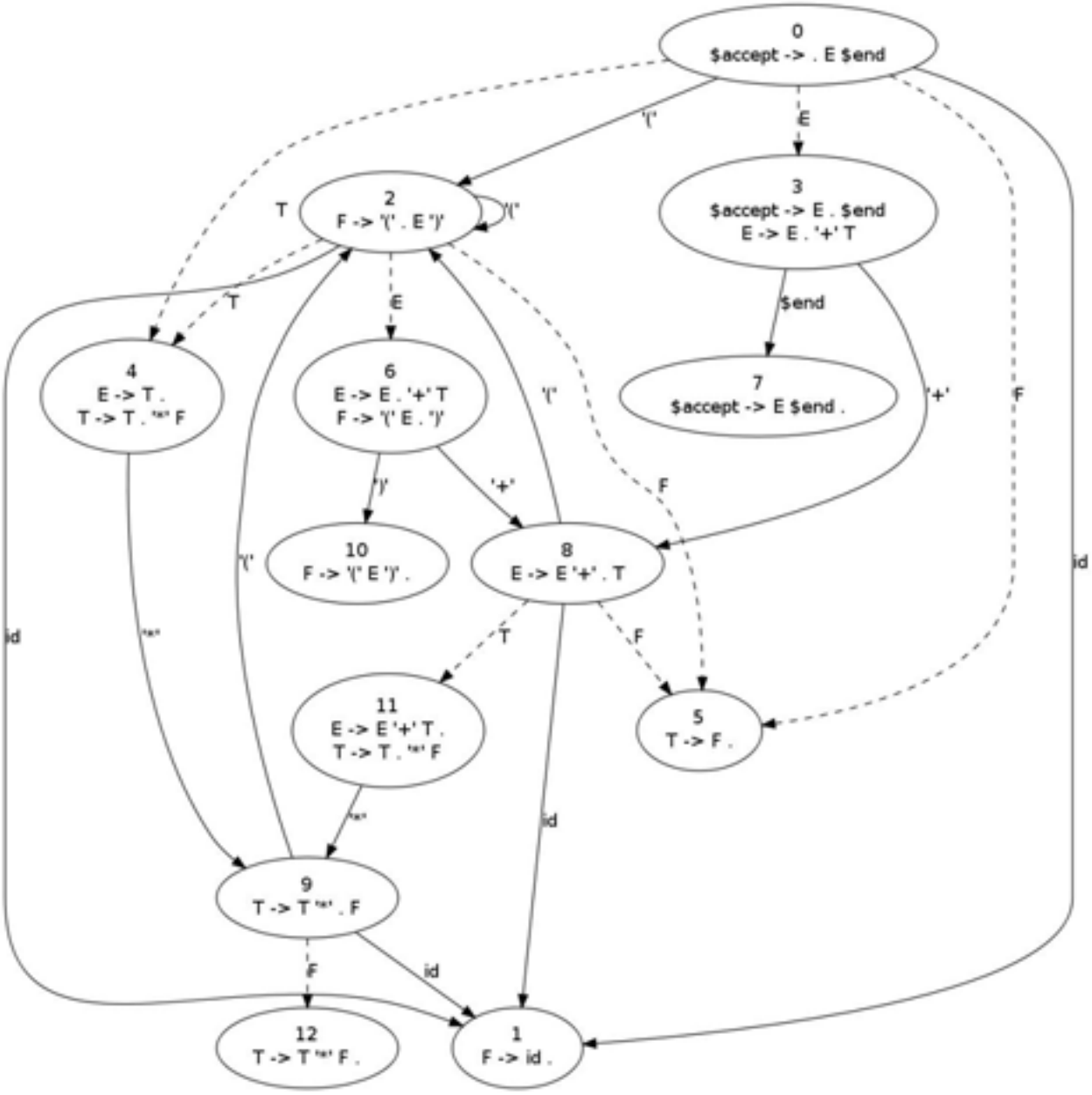


Tabla SLR(1)

state	id	+	*	(	)	$	E	T	F
0	s:5			s:4			g:1	g:2	g:3
1		s:6				acc			
2		r:2	s:7		r:2	r:2			
3		r:4	r:4		r:4	r:4			
4	s:5		s:4				g:8	g:2	g:3
5		r:6	r:6		r:6	r:6			
6	s:5		s:4					g:9	g:3
7	s:5		s:4						g:10
8		s:6			s:11				
9		r:1	s:7		r:1	r:1			
10		r:3	r:3		r:3	r:3			
11		r:5	r:5		r:5	r:5			

Diferencias entre LR(0) y SLR(1)

Los pasos a seguir para generar una tabla SLR(1):

- Calcular el autómata LR(0).
- Añadir todas las acciones de desplazar para las transiciones del autómata.
- Colocaremos la reducción correspondiente cuando el símbolo que este en la cinta este en el FOLLOW de la variable por la que reducimos.
- El símbolo en la cinta se reutiliza despúes de la reducción.

Límites del Análisis SLR

Hay que ir un poco lejos pero se llega

Un ejemplo un tanto artificial:

Formato inhabitual de declaraciones de identificadores.

- Forman listas de variables, o bien son un nombre de tipo.
- En este ejemplo un tipo es la palabra reservada MATRIZ seguida de las dimensiones, que son números.
- Permitimos varias maneras de declarar tipos y variables.

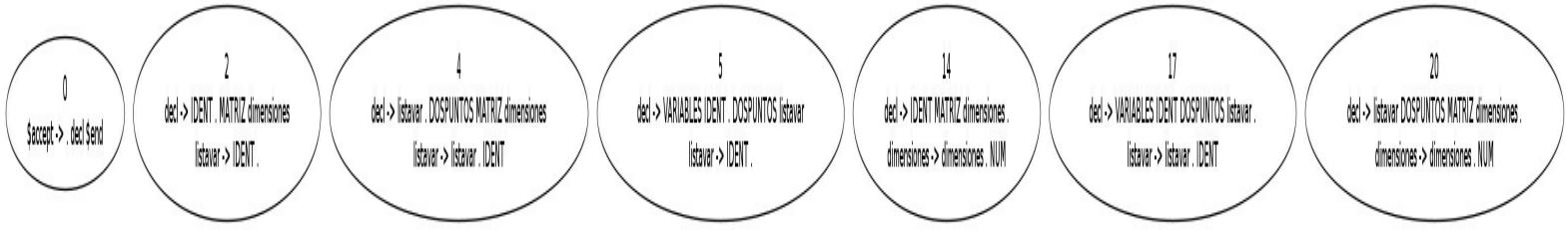
```
%%
decl : VARIABLES listavar MATRIZ dimensiones
     | listavar DOSPUNTOS MATRIZ dimensiones
     | IDENT MATRIZ dimensiones
     | VARIABLES IDENT DOSPUNTOS listavar ;

listavar : IDENT | listavar IDENT ;

dimensiones : NUM | dimensiones NUM ;
```

Comparamos el analizador LR(0) y el SLR (I)

Construyendo algunos de los estados

Empezamos escribiendo algunas declaraciones aceptadas por la gramática y otras que no lo sean. Abreviamos los *tokens* a una sola letra para no tardar tanto. Luego calculamos FOLLOW.

Ahora calculamos una parte del autómata LR(0):



Seguiremos la numeración asignada por `bison`.

Comparamos el analizador LR(0) y el SLR (II)

Identificamos dónde tiene un conflicto el analizador LR(0)

¿Que ocurre con el analizador LR(0)?

Se nos presentan varios conflictos `shift-reduce`:

- En el estado 2, puede reducir `IDENT` a `listavar`, o desplazar `MATRIZ`;
- En el estado 5, puede reducir `IDENT` a `listavar`, o desplazar `DOSPUNTOS`;
- En el estado 14, puede reducir `decl` o desplazar `NUM`;
- En el estado 17, puede reducir `decl` o desplazar `IDENT`.
- En el estado 18, puede reducir `decl` o desplazar `NUM`.
- En el estado 20, puede reducir `decl` o desplazar `NUM`.

Los repasamos de abajo a arriba: SLR resuelve perfectamente los conflictos en los estados 14, 17, 18 y 21, porque el FOLLOW de `decl` sólo incluye el fin de datos.

Comparamos el analizador LR(0) y el SLR (III)

Encontramos que el analizador SLR aún tiene dificultades

¿Qué información tiene el analizador SLR en los estados 2 y 5?

- En el estado 2, puede reducir `IDENT` a `listavar`, o desplazar `MATRIZ`; pero `MATRIZ` está en el FOLLOW de `listavar`.
- En el estado 5, puede reducir `IDENT` a `listavar`, o desplazar `DOSPUNTOS`; pero `DOSPUNTOS` está en el FOLLOW de `listavar`.

Los conflictos de los estados 2 y 5 no se pueden resolver mediante análisis SLR.
Sin embargo, `bison` los resuelve sin ningún tipo de queja.
¿Cómo lo hace?
Antes: ¿Cómo lo haríamos nosotros?

Análisis LR canónico

Acarreamos anotada la parte de FOLLOW que nos importa

El proceso será similar a SLR.

La diferencia será que mantendremos más información en los *ítems*:

- qué símbolo no terminal estamos buscando,
- por qué punto vamos de la parte derecha de la regla,
- y qué símbolo terminal esperamos ver a continuación para reducir.

Así pues, los ítems LR(1) traerán consigo la parte relevante del FOLLOW; esa parte se denomina el *look-ahead* del ítem.

Por ejemplo, tendremos un ítem LR(1):

$$[\texttt{decl} \rightarrow \bullet\, \texttt{V listavar M dim}, \$]$$

y pasaremos (con una λ-transición) de

$$[\texttt{decl} \rightarrow \texttt{V} \bullet \texttt{listavar M dim}, \$] \text{ a } [\texttt{listavar} \rightarrow \bullet\, \texttt{I}, \texttt{M}].$$

El Autómata LR(1)

Con el que se construyen las tablas de análisis LR canónico

A base de ítems LR(1).

Un ítem LR(1) es un par formado por un ítem LR(0) y un *token*; éste último se denomina *look-ahead* del ítem.

- Estados del autómata indeterminista: los ítems LR(1).
- Transiciones: como en el autómata indeterminista LR(0):
  - Transición con un símbolo a la vez que el punto avanza el mismo símbolo; los *look-ahead* simplemente se conservan: $[A \to \alpha \bullet X\ \beta,\ a]$, con X, vamos a $[A \to \alpha\ X \bullet \beta,\ a]$.
  - Por palabra vacía cuando el punto está delante de un símbolo no terminal; de $[A \to \alpha \bullet B\ \beta,\ a]$ a $[B \to\ \bullet\, \gamma,\ b]$:
    - los *look-ahead* b se obtienen como el FIRST de lo que viene a continuación, o heredando el *look-ahead* si el símbolo no terminal está al final de la parte derecha de la regla ($\beta = \lambda$);
    - o, lo que es lo mismo, FIRST(βa), generando tantas λ-transiciones como elementos tenga este conjunto.
- Después se determiniza como siempre.

Ejemplo Completo de Analizador LR Canónico (I)

Con otra gramática minúscula

```
%token A B C

%%

comp : A igual A
     | A dist B
     | B dist A
     | B igual B ;

igual : C ;

dist: C ;
```

Ejemplo Completo de Analizador LR Canónico (II)

Con una gramática minúscula

Tareas:

1. Encontramos todas las entradas generadas,
2. identificamos el árbol de análisis sintáctico de cada una,
3. calculamos el FOLLOW de `comp`, de `igual` y de `dist`,
4. calculamos todos los estados LR(1) y
5. construimos la tabla de análisis LR canónico;
6. luego calculamos también los estados LR(0) y
7. comparamos el analizador LR canónico con el analizador SLR, el cual vemos que tiene un par de conflictos *reduce-reduce* que no aparecen en el analizador LR canónico.