Lenguajes de Programación y Procesadores de Lenguajes 2014/15
BISON III: Atributos y acciones semánticas en Bison

Escuela Técnica Superior de Ingeniería Informática
Universitat Politècnica de València

V.14.0

Índice

1. Objetivos **1**

2. Bison y los esquemas de traducción dirigidos por la sintaxis (ETDS) **1**

3. Definición de los tipos de los atributos **2**

4. ETDS con atributos sintetizados **2**

5. Acciones a mitad de una regla **3**

6. ETDS con atributos heredados **3**

1. Objetivos

Los objetivos de esta práctica son

- Aprender a incorporar acciones semánticas en la especificación Bison.
- Conocer cómo definir los tipos de los atributos en Bison.
- Asociar atributos sintetizados y heredados a los símbolos de la gramática

Para realizar este trabajo se recomienda al alumno que, además de estudiar detenidamente este documento, consulte el manual de Bison.

2. Bison y los esquemas de traducción dirigidos por la sintaxis (ETDS)

En un programa Bison se pueden asociar a las reglas gramaticales acciones semánticas escritas en el lenguaje C. Cada vez que se aplique una regla gramatical se ejecutarán las acciones semánticas que tiene asociadas. El orden de ejecución de las acciones vendrá determinado por el análisis sintáctico y la posición que ocupan estas acciones en el lado derecho de las reglas.

En la mayoría de los programas se necesitarán diferentes tipos de atributos asociados a cada símbolo de la gramática. Cuando se apila un símbolo (terminal o no-terminal) en la pila del analizador sintáctico LALR(1), también se apilan sus valores semánticos (atributos). Por esa razón es necesario definir el tipo de de los atributos que se asocian a cada símbolo. Para ello se debe:

- Especificar la colección completa de tipos de datos (atributos) posibles en la declaración de Bison %union.
- Elegir uno de estos tipos para cada símbolo (terminal o no-terminal) que necesite un valor semántico. Para los terminales esto se hace con la declaración de Bison %token y para los no-terminales con la declaración de Bison %type.

3. Definición de los tipos de los atributos

En un programa Bison con acciones semánticas se emplea %union para definir el conjunto de valores semánticos (atributos) que puede tomar cada símbolo o acción. Así por ejemplo se puede definir esta unión como:

```
%union
{
  char *ident;     /* Nombre del identificador */
  int cent;         /* Valor de la cte numerica entera */
}
```

Tras esta declaración cualquier símbolo terminal o no-terminal podrá tener asociado un valor entero (*cent*), o un puntero a un carácter (*∗ident*). Pero para que esto sea así es necesario indicar en la sección de declaraciones Bison qué tipo de atributo irá asociado a cada símbolo terminal (mediante *%token*) o no-terminal (mediante *%type*). Si por ejemplo se desea que el símbolo terminal *cte* tenga asociado un valor entero (campo *cent* de la unión definida), deberá escribirse en Bison:

```
%token <cent> CTE_
```

De igual forma, si se desea que el símbolo no-terminal *expresion* tenga asociado un valor entero (campo *cent*), será necesario indicarlo mediante *%type*:

```
%type <cent> expresion_
```

Los valores semánticos de los no-terminales se asignarán en las acciones de las reglas Bison que los definen. Los de los terminales se asignan a la variable *yylval* en las reglas del Flex donde se define cada token. Así por ejemplo, podríamos encontrar en el fichero Flex.

```
{digito}+      {yylval.cent= atoi(yytext); return(CTE_);}
```

Donde *cent* es un campo de la %union definida en la sección de declaraciones de Bison, y *CTE_* el nombre dado al símbolo léxico correspondiente a un número entero.

Como se puede observar en el ejemplo anterior, Flex proporciona información sobre el token reconocido mediante varias variables globales, entre las que se pueden destacar *char* yytext* que será una cadena que contiene el lexema analizado, y *int yyleng* que contiene la longitud del lexema analizado.

4. ETDS con atributos sintetizados

Una acción semántica consiste en instrucciones de C encerradas entre llaves. Se pueden situar en cualquier posición dentro de la regla, aunque frecuentemente aparecen al final.

El código C en una acción puede usar o dar valor a los valores semánticos de los símbolos y acciones de la regla mediante la construcción $n, que hace referencia al valor semántico de la componente n-ésima del lado derecho de la regla. El valor semántico (atributo sintetizado) para el no-terminal del lado izquierdo de la regla viene representado por $$[1]. Veamos un pequeño ejemplo:

```
Factor: ( Expresion ) { $$=$2 } ;
```

Esta regla devuelve como valor semántico del no-terminal *Factor* (representado en la acción semántica por $$), el valor semántico del símbolo no-terminal *Expresion* (representado por $2 ya que se trata del segundo símbolo del lado derecho de la regla).

Si no se especifica una acción para una regla, Bison asume una por defecto: $$ = $1. De este modo el valor del primer símbolo del lado derecho se convierte en el valor del no-terminal del lado izquierdo. Esta acción por defecto solo es válida si concuerdan los dos tipos de datos. No hay una acción por defecto con significado para la regla vacía.

Si para la escritura de la acción semántica se necesita alguna variable local temporal, se puede declarar dentro de las llaves de la acción semántica, del mismo modo que se haría en el lenguaje C.

[1] Pero solo cuando aparece en la última acción del lado derecho.

5. Acciones a mitad de una regla

Ocasionalmente es de utilidad poner una acción en medio de una regla. Estas acciones se escriben como las acciones al final de la regla, pero se ejecutan antes de que el analizador llegue a reconocer los componentes que la siguen. Una acción en mitad de una regla puede hacer referencia a los componentes que la preceden utilizando $n, pero no puede hacer referencia a los componentes que la siguen porque ésta se ejecuta antes de que sean analizados (solo se permiten gramáticas L-atribuidas).

Las acciones en mitad de una regla por sí mismas cuentan como uno de los símbolos de la regla. Esto produce una diferencia cuando aparece alguna otra acción a su izquierda, ya que *se debe contar esta acción a la hora de calcular los números que siguen a $* para hacer referencia a los valores semánticos.

Cualquier acción en mitad de una regla puede tener también su propio valor semántico. Para asignar un valor a una acción, basta con asignar *dentro de la acción* un valor a $$. Las acciones que sigan a ésta en la regla pueden hacer referencia a este valor utilizando $n (recordar que a la hora de numerar los símbolos del lado derecho también hay que contar las acciones que haya entre ellos). Ya que no hay un símbolo que identifique la acción, no hay manera de declarar por adelantado un tipo de dato para el atributo. Por esta razón Bison nos ofrece la construcción $< *tipo* >, (donde *tipo* representa uno de los tipos definidos en la %union para asignar valor semántico a la acción semántica que lo incluye).

Por ejemplo, en las dos siguientes reglas las acciones semánticas generarían el mismo resultado:

```
F: L  B  {$$ = $1 * 2 + $2; }
     |     L { $<cent>$ = $1 * 2 ; } B { $$ = $<cent>2 + $3; }
```

Donde en la segunda producción:

- $< *cent* >$ hace referencia al valor semántico de la propia acción (que será del tipo < *cent* > que aparece en %union).
- $< *cent* >2 hace referencia al valor semántico de la acción a mitad regla, el 2 corresponde a la posición que esta acción ocupa dentro de la parte derecha de la producción, contando símbolos y acciones semánticas.

Es importante destacar que no se puede asignar un valor al no-terminal del lado izquierdo en una acción mediante una acción que aparezca en medio de la regla, ya que *$$ representa en este caso el valor semántico de la propia acción.* La única forma de establecer el valor para el no-terminal del lado izquierdo es mediante una acción *al final* de la regla.

6. ETDS con atributos heredados

Si se desea utilizar atributos heredados en Bison se debe emplear un mecanismo que permita acceder a una posición de la pila del analizador en busca del valor del atributo. Esto se hace usando $n con n tomando un valor cero o negativo. De esta forma se pueden emplear valores semánticos de símbolos que se metieron anteriormente en la pila. Ésta es una práctica muy arriesgada, ya que es necesario conocer el contexto en el que se aplica la regla para conocer exactamente en que posición de la pila se encuentra el valor al que queremos acceder. Esto se puede ver en el siguiente ejemplo:

```
F: E B '+' E { ... } | E B '-' E { ... } ;
B: /* vacío */ { valor = $0; } ;
```

Siempre que el no-terminal B se utilice solamente de la manera mostrada aquí, el $0 de la última acción semántica hará referencia al valor semántico de E que precede a B en la definición de F. En concreto, el valor del $0 que aparece en la segunda regla de B será siempre el del símbolo que se encuentre justo debajo de B en la pila en el momento de la ejecución de la acción semántica: En este ejemplo siempre E. De igual forma, $-1 sería el atributo del símbolo que se encontrase dos posiciones por debajo de B en la pila.