Traductor para la generación de Tablas de Verdad

Pablo Alejandro Gancharov Diaz
Universidad Tecnológica Nacional
Facultad Regional Concepción del Uruguay
Departamento de Ingeniería en Sistemas de Información

Abstract
Las herramientas disponibles para el cálculo matemático son muchas y cubren prácticamente todas las necesidades que un estudiante pueda tener durante el transcurso de su carrera, pero existen aspectos por mejorar en algunos tipos de herramientas, como por ejemplo las que generan tablas de verdad para el análisis de proposiciones lógicas.
El cálculo de tablas de verdad consiste en analizar una proposición lógica y calcular su valor de verdad para cada una de las combinaciones de valores que puedan tomar sus variables. El resultado se muestra en una tabla, en la que cada fila representa una combinación distinta de los valores que tomaron las variables y el resultado que proporcionó el cálculo de la expresión lógica. El número de filas corresponde a una función exponencial que depende de la cantidad de variables que posea la proposición lógica.
El cálculo de este tipo de ejercicios requiere gran cantidad de tiempo y se convierte en una tarea repetitiva y monótona, a la que hay que sumarle que el estudiante es muy susceptible a cometer errores.
En este artículo se presenta una aplicación para la generación de tablas de verdad aplicando conceptos de traductores, en particular, analizadores léxicos y sintácticos y autómatas de reconocimiento por pila vacía. El programa se denomina "Tablas de Verdad", fue construido en el lenguaje Pascal y brinda la oportunidad al estudiante de verificar sus ejercicios en forma inmediata e independiente.

Palabras Clave:
Tablas de Verdad. Proposiciones Lógicas. Aplicaciones de Traductores e Intérpretes.

1. Introducción

La construcción de Tablas de verdad forma parte del estudio de la lógica booleana y del cálculo proposicional; temas que se dictan en nuestra facultad en la asignatura *Matemática Discreta.* Hasta el momento, este tipo de ejercicios se resolvían de forma manual.

Debido a que la construcción de tablas de verdad suele convertirse en una tarea monótona y repetitiva, surgió la idea de crear una herramienta que facilite esta labor.

Este proyecto es una iniciativa propia y posee dos aspectos, uno el poder contar con una herramienta que sea útil tanto para el estudiante como para el docente, y el otro, el poder aplicar conceptos de compiladores y teoría de lenguajes para resolver un problema de la vida real. Este proyecto integra las materias *Sintaxis y Semántica de Lenguajes* y *Matemática Discreta*, poniendo en práctica por un lado conceptos sobre analizadores léxicos y autómatas finitos, analizadores sintácticos y autómatas de reconocimiento por pila vacía; y por otro lado, conceptos de lógica booleana.

Aplicaciones similares

Buscando software en Internet para este propósito encontramos:

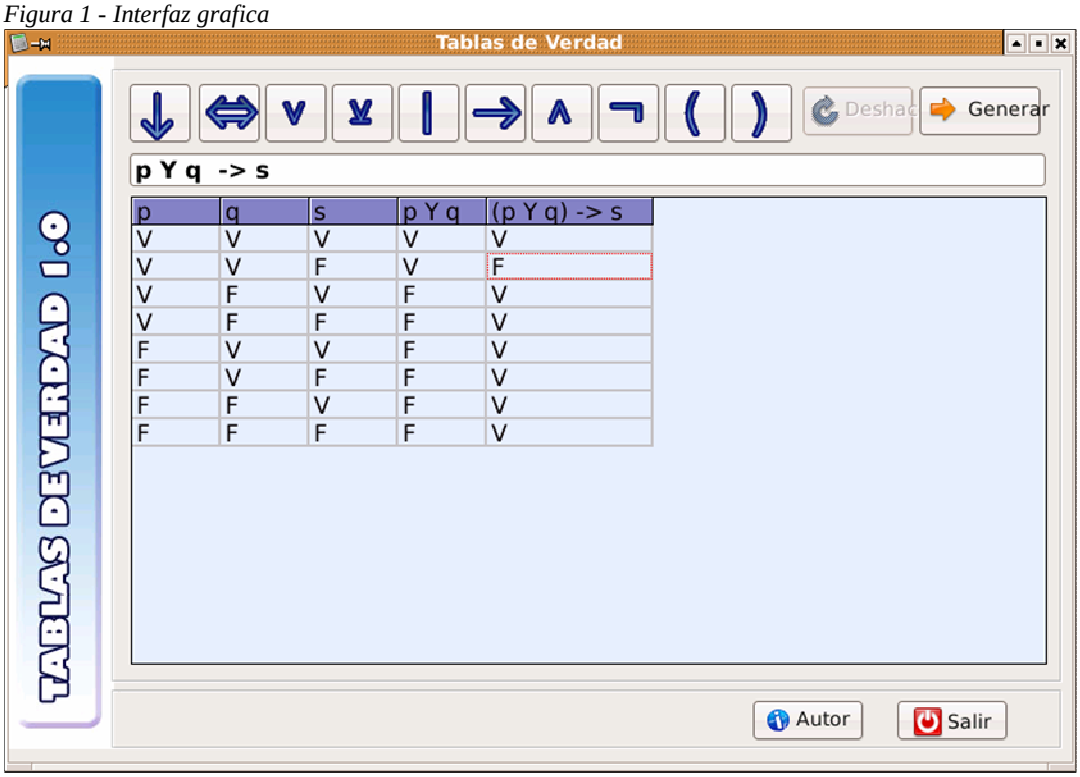
"TRUTH TABLES" [1] programado por David Lovelock del departamento de Matemáticas de la universidad de Arizona. Es un software para DOS que posee una interfaz en modo texto. Su funcionalidad no alcanza a cubrir totalmente los requerimientos derivados de los ejercicios presentados por la cátedra. Por ejemplo, el programa solo posee cuatro operadores lógicos básicos, provocando que ciertos ejercicios deban simplificarse para poder ser procesarlos por el programa, lo que le añade más trabajo al estudiante y aumenta la probabilidad de cometer errores.

Otro software para cubrir las necesidades que citamos es una aplicación desarrollada en Java llamada “Truth Table Constructor” [2] desarrollada por Brian S. Borowski la cual cumple con los requerimientos de funcionalidad, pero que impone una restricción: por ser una aplicación Web requiere de una conexión a Internet para funcionar.

En este artículo se presenta una aplicación desarrollada en el lenguaje Pascal, que fue realizada teniendo en cuenta los requisitos de funcionalidad y disponibilidad establecidos por la cátedra de *Matemática Discreta*, para la resolución de los ejercicios presentados por los docentes de la misma. La aplicación contiene un analizador léxico, un analizador sintáctico, y una estructura adicional para la interpretación de las expresiones lógicas que se ingresen. T odo el código del dominio del problema se encuentra encapsulado en un único modulo. La entrada de este programa es una cadena que representa una expresión lógica, y su salida es la tabla de verdad asociada, que se almacena completamente en una estructura de datos dinámica (lista de listas). El programa fue escrito en lenguaje Pascal utilizando el paradigma estructurado.
La interfaz grafica es totalmente independiente del generador de tablas, lo que permite que el producto funcione en distintas plataformas; se desarrolló una interfaz en modo texto para DOS, una gráfica en Delphi para Windows y una gráfica en Lazarus para Gnu/Linux. La elección de Delphi 7 para la versión de Windows, y Lazarus para su contraparte en Linux, fueron por conveniencia del Programador. Cabe destacar que producto estará bajo la licencia GPL.

Para su diseño se tuvieron en cuenta atributos de calidad, como por ejemplo que la interfaz (ver *Fig 1*) sea simple, estéticamente agradable, y que prácticamente no requiera aprendizaje para su utilización.

Figura 1 - Interfaz grafica



2. Marco Teórico

En esta sección se presentan los conceptos teóricos aplicados en el diseño de la aplicación.

Traductor [4]: Es un programa que toma como entrada un programa escrito en un lenguaje, el *lenguaje fuente*, y lo convierte en un programa equivalente, escrito en otro lenguaje, el *lenguaje objeto*.

Componente léxico [4]: Conjunto de cadenas que poseen el mismo significado desde el punto de vista de la sintaxis.

Análisis léxico [4]: Agrupa secuencias de caracteres con significado colectivo. Controla errores léxicos. Para su implementación se utilizan autómatas finitos [5]

Análisis sintáctico [4]: Controla que la secuencia de componentes léxicos sea válida, y crea el árbol sintáctico donde queda reflejado el orden en que se generaran y ejecutaran las instrucciones.

Gramáticas independientes del contexto (CFG) [5]: Es una notación que genera lenguajes independientes del contexto (CFL), se utilizan para especificar la sintaxis de lenguajes de programación. Se define como una 4-upla (V, T, P, S), donde V es el conjunto finito de variables sintácticas, T el conjunto finito de terminales (componentes léxicos, desde el punto de vista del análisis léxico), $S \in V$ el símbolo inicial, y P el conjunto finito de producciones de la forma $A \rightarrow \alpha$ con $\alpha \in (V \cup T)^*$.

Analizador sintáctico descendente predictivo no recursivo (ASDPNR) [4]: Es un analizador sintáctico basado en el PDA (Push Down Automata) determinístico con reconocimiento por pila vacía, que permite controlar la sintaxis en base a una gramática $LL_{(1)}$. Mantiene una Tabla de Análisis Sintáctico para decidir cual producción aplicar en cada paso de una derivación por izquierda para la cadena de entrada.

3. Diseño y Construcción

El funcionamiento de la aplicación se divide en dos funcionalidades separadas: la interfaz con el usuario y la que procesa la expresión y genera la tabla de verdad.

Interfaz

La interfaz es la encargada de interactuar con el usuario, obtener la cadena que representa la expresión lógica y luego mostrar la tabla correspondiente. En la *Figura 1* se muestra la interfaz gráfica para Linux, la cual es exactamente igual

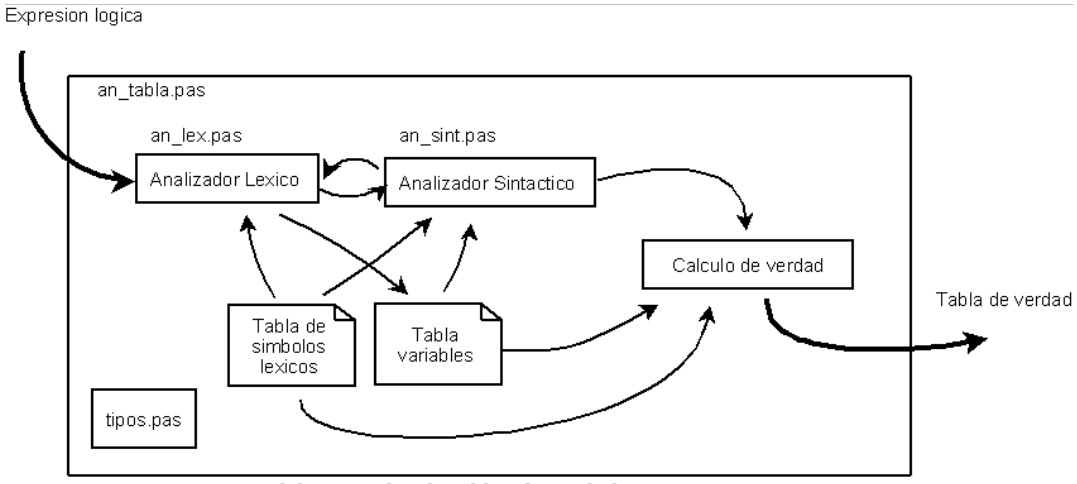


Fig. 2.-Estructura interna del generador de tablas de verdad.

a la de Windows.
Para facilitar la utilización del programa se incorporó una barra de botones, que agiliza el ingreso de las expresiones y que permite al usuario la utilización de la herramienta de forma intuitiva.

Generador de tablas
En el modulo del generador de tablas se agrupan 3 tareas, que consisten en: (a) aplicar el análisis léxico y sintáctico a la expresión lógica de entrada, (b) calcular el valor de verdad para cada combinación de valores lógicos y (c) liberar la memoria ocupada durante el proceso. En la *Figura 2* se muestra la estructura interna del modulo generador de tablas de verdad.

a. Análisis léxico y sintáctico
El análisis sirve para comprobar que la expresión sea correcta desde el punto de vista estructural, y de esta manera se asegura que se podrá construir su tabla de verdad correspondiente. Este proceso se vale de un analizador léxico y un analizador sintáctico.
La tarea del analizador léxico es reconocer cada componente léxico de la expresión e ir conformando dos estructuras: una pila para los operadores lógicos y otra para las cadenas que actuarán como operandos. En este momento se detectan los posibles errores léxicos de la cadena de entrada.
Luego se realiza el análisis sintáctico mediante un ASDPNR, el cual utiliza la tabla de operadores de la *Figura 3*, la tabla de operandos que se obtuvo en el análisis léxico y la Tabla de Análisis Sintáctico correspondiente a la gramática que específica la sintaxis de las expresiones lógicas (*Figuras 4.1, 4.2, 4.3*).

Como resultado del análisis se obtiene la tabla operadores lógicos utilizados y la tabla de variables lógicas, o sea las variables que tomarán los valores Verdadero o Falso al momento de calcular las producciones lógicas que conformarán la tabla de verdad.

Prioridad	Nombre	Operador
0	Negación	-
1	Conjunción	Y
1	Negación conjunta	NEGCONJ
2	Disyunción débil	O
2	Negación alternativa	NEGALT
3	Condicional	COND
4	Disyunción fuerte	OFUERTE
4	Bicondicional	BICOND
	Paréntesis izq.	(
	Paréntesis der.	)

Figura 3.- Tabla con los operadores y sus prioridades

```
bA → bA "OFUERTE" bB
       | bA "BICOND" bB
       | bB
bB → bB "COND" bC | bC
bC → bC "O" bD | bC "NEGALT" bD
     | bD
bD → bD "Y" bE | bD "NEGCONJ" bE
     | bE
bE → "-" bE | "(" bA ")"
     | Variable
```

Figura 4.1 - Gramática utilizada para definir la sintaxis

b. Cálculo de la expresión lógica
Este paso utiliza un autómata de reconocimiento por pila vacía y dos

pilas, una para los operadores y otra para los operandos.
Este análisis se debe efectuar tantas veces como combinaciones de V o F sean posibles [3] para las variables lógicas. es decir 2^n veces, siendo n el numero de variables lógicas de la expresión.

```
bA → bBM
M → "OFUERTE" bB M |
     "BICOND"  bB M | ε
bB → bCN
N → "COND" bC N | ε
bC → bD T
T → "O" bD T | "NEGALT" bD T | ε
bD → bE U
U → "Y" bE U |"NEGCONJ" bE U | ε
bE → "-" bE | "(" bA ")" |
     Variable
```

Figura 4.2 - Gramática. factorizada y sin recursividad por izquierda.

combinación, el cual es almacenado. A continuación hay que volver a asignar valores y reiniciar el análisis; esta tarea se repite para cada una de las posibles combinaciones.
Luego la tabla de verdad completa es almacenada en una estructura de lista de listas.

c. Liberación de memoria

Una vez utilizada la tabla de verdad por la interfaz es necesario que se libere la memoria que ésta ocupa.

4. Implementación

Este software se instaló para ser utilizado por parte de docentes y alumnos de la materia *Matemática Discreta*, que se dicta en primer año de la carrera Ingeniería en Sistemas de Información, de la Facultad Regional Concepción del Uruguay, con el objetivo de contribuir al dictado de clases, y mejorar la calidad del aprendizaje.
Finalmente, después de un riguroso examen por parte de los docentes, se lo incluyó como herramienta al servicio de

	OFUERTE	BICOND	COND	O	Y	NEGALT	NEGCONJ	-	(	)	Variables	$
bA								bBM	bBM		bBM	
M	OFUERTE bBM	BICOND bBM								ε		ε
bB								bCN	bCN		bCN	
N	ε	ε	COND bCN							ε		ε
bC								bDT	bDT		bDT	
T	ε	ε	ε	O bDT		NEGALT bDT				ε		ε
bD								bEU	bEU		bEU	
U	ε	ε	ε	ε	Y bEU	ε	NEGCONJ bEU			ε		ε
bE								- bE	(bA)		Variables	

Figura 4.3. – TAS.

A las variables que se encuentran almacenadas en la pila de variables lógicas se les asigna los valores Verdadero o Falso y se inicia el cálculo de la expresión, de acuerdo a los operadores utilizados y sus prioridades. Esto devuelve el valor para la primera

los alumnos del curso de ingresantes del ciclo lectivo 2008.

5. Conclusiones

La aceptación del sistema fue excelente y en este momento se lo ha adoptado como una herramienta más del curso y

se distribuye de forma gratuita entre los estudiantes. También se incluyó una referencia en el modulo de estudio, y en el nuevo sitio Web de la materia.
Cabe destacar que el código es de libre acceso y se encuentra bajo la licencia GPL.

Agradecimientos.
Agradezco a Esteban Tripodi y Andrés Pascal, por la ayuda que me prestaron y sin la que este proyecto no seria posible, y a Mercedes Martinelli por ser la que inició todo esto.

Referencias.

[1] http://archives.math.utk.edu/software/msdos/discrete.math/truth
[2] http://www.brian-borowski.com
[3] Martinelli, Mercedes Beatriz: Modulo de la cátedra Matemática discreta. Universidad Tecnológica Nacional. Facultad Regional de Concepción del Uruguay (2006).
[4] Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman. Compilers: Principles, Techniques, and Tools. Addison Wesley.
[5] John E. Hopcroft, Jeffrey D. Ullman Introduction to Automata Theory, Languages, and Computation Addison Wesley.

Software utlizado.

1. Turbo Pascal 7 - Borland
2. Turbo Delphi 7 - Borland
3. Free Pascal
4. Lazarus

Docente Tutor.

Ing. Andres J. Pascal.
andrespascal2003@yahoo.com.ar

Datos de contacto.

Pablo Alejandro Gancharov Diaz.
Universidad Tecnológica Nacional.
Facultad Regional Concepción del Uruguay
Moreno 659.
(3260) Concepción del Uruguay- Entre Rios.
pablogancharov@gmail.com