Interprete Neupro utilizando la NeuPro Abstract Machine

Ulises M. A. Rapallini / Jorge H. Nanclares

Universidad Tecnológica Nacional
Facultad Regional Concepción del Uruguay
Departamento de Ingeniería en Sistemas de Información
GEINAR - Grupo de Estudio de Inteligencia Artificial

Ing. Pereyra 676 – Concepción del Uruguay
Entre Ríos - Argentina
e-mail : rapa@frcu.utn.edu.ar

Resumen

El lenguaje de programación Prolog es uno de los principales lenguajes de la Inteligencia Artificial. Pertenece a la programación lógica y al desarrollo de bases de datos simbólicas. Existen varios descendientes como el Hilog, el XSB, y el NEUPRO. Todos ellos se construyen por cláusulas del tipo : CABEZA :- TERMINO1, TERMINO2,...,TERMINOK.
Para validar CABEZA los términos TERMINO1,..,TERMINOK deben validarse previamente. Una cláusula Neupro es diferente. Cada término es un factor, con un cierto peso, la cabeza tiene un umbral determinado: si la suma de los factores excede dicho umbral, entonces la cláusula es verdadera. Esta cláusula permite identificarla con una neurona artificial. Cada conjunto de cláusulas será equivalente a una red neuronal.
Si el umbral es igual al número de términos, y cada término tiene un peso igual a 1, el programa es un programa Prolog clásico. El trabajo se basa en la Máquina Abstracta de Warren - WAM, que reduce el Prolog a un conjunto de rutinas, facilitando su modificación y la escritura de un compilador eficiente.
Este trabajo describe un intérprete para el lenguaje NeuPro, utilizando la NeuPro Abstract Machine que resultó del estudio de la WAM y modificaciones al algoritmo tradicional de unificación.

Palabras Clave

Prolog, Redes Neuronales, Bases de Datos, Data Mining, Algoritmos Genéticos, Sistemas Dinámicos

Workshop

VI Workshop de Agentes y Sistemas Inteligentes

Introducción

La idea principal del intérprete es pasar todo proceso recursivo necesario para la Unificación a proceso procedural (iterativo).
El intérprete cuenta con diferentes partes, cada parte resuelve un problema en particular. Conceptualmente se identifican los siguientes elementos: un analizador léxico, un analizador semántico, un elemento para cargar las estructuras diseñadas con la que trabajará la NPAM, y finalmente la Máquina Abstracta Neupro. El diseño del programa se realizó orientado a objetos y se programó en C++.

Descripción del Intérprete

Para leer el archivo se utilizó la técnica de doble-buffer permitiendo aumentar la eficiencia de lectura del archivo en disco. Para el analizador léxico se utilizan autómatas finitos para reconocer los lexemas del código fuente, los lexemas se almacenan en estructuras auxiliares; con estos datos luego se determina la semántica. Una vez reconocidos todos los elementos del programa, se cargan en las estructuras de la NPAM para interpretar el programa.

Terminología

Antes de comenzar a explicar el funcionamiento de la NPAM analizaremos los conceptos que se utilizaron para el diseño y la programación.

Sentencia Neupro (cláusula)

Una sentencia Neupro tiene dos partes una cabeza y una cola, las dos separadas por un "si" (:-), una sentencia Neupro es similar a una sentencia Prolog, agrega a cada predicado un peso luego del carácter ":" (dos puntos). El formato general de una sentencia Neupro es :

predicado(Var,Var,...,Var):peso :- predicado(Var,Var,...,Var):peso, .., predicado(Var, Var, ..., Var):peso.

Donde "predicado" es el nombre de un predicado, "Var" es el nombre de una variable, "peso" es un valor indicando cual es el peso del predicado.

Por Ejemplo :

violeta(X,Y):W :- rojo(X):20, azul(Y):10.

En esta sentencia identificamos la cabeza "violeta(X,Y):W", y la cola "rojo(X):20, azul(Y):10." . Siguiendo el comportamiento del lenguaje, se puede ver que el peso del predicado "violeta" dará como resultado 30, se asume que al no indicar el peso del predicado, este tomará el valor correspondiente al disparo, esto es, la cláusula será siempre verdadera.

Entorno de sentencia

Un entorno de sentencia se define como el conjunto de todas las variables que aparecen a la cláusula, este conjunto de variables tomarán diferentes valores durante el proceso de unificación. El entorno de la sentencia junto con los valores que toman las variables definen el estado de la sentencia. Por ejemplo, para el siguiente programa Neupro[1] :

```
1. marron(X,Y,Z):W :- rojo(X):10, amarillo(Y):30, azul(Z):45.
2. rojo(50):Z.
3. rojo(32):10.
4. amarillo(30):12.
5. amarillo(Q):30.
6. azul(12):Q.
```

Prg. A-1

Aquí se identifican 6 sentencias, una por cada línea de programa, para el intérprete existen 6 entornos de sentencia, para la sentencia en la línea 1 el entorno "Env_1" esta formado por las Variables X, Y, Z y W

[1] Se indica el número de línea de programa para una explicación posterior.

$$Env_1(X,Y,Z,W)$$

El sub-índice "1" indica la posición de la sentencia en el vector de sentencias[2], un posible estado para este entorno seria :

$$ST_k^{Env_1}(50,30,free,85)$$

Donde "*ST*" indica que es un estado de sentencia, el super-índice "*Env1*" indica que el estado pertenece al Entorno 1, el sub-índice "*k*" indica un estado genérico del entorno de sentencias "*Env1*", entre los paréntesis están los posibles valores de las variables durante la unificación, la posición de los valores se corresponde con la posición de las variables en el Entorno.
Este estado de sentencia indica que :

$$X = 50, Y = 30, Z = free, W = 85$$

Otro posible estado es :

$$ST_x^{Env_1}(50,free,free,85)$$

El entorno de una sentencia pasará por n estados durante la unificación de la cláusula hasta que todas las variables queden instanciadas en algún valor o libres si no es posible instanciarlas. Para el caso anterior, el primero y ultimo estado respectivamente serán :

$$ST_1^{Env_1}(free,free,free,free),.....,ST_n^{Env_1}(50,30,30,85)$$

Para la cláusula de la línea 3 el entorno de sentencia es el siguiente :

$$Env_3(\_VAR,\_PESO)$$

el intérprete considera las dos variables ya instanciadas, aquí existe sólo un estado para este entorno:

$$ST_1^{Env_3}(32,10)$$

Tabla de entornos

La tabla de entornos almacena los entornos de las sentencias durante los procesos iterativos de unificación, funciona como una pila, apilando los entornos de cada sentencia cada vez que se requiere. Esto se modela con un simple Objeto Pila con las funciones para su tratamiento.

Vector de Sentencias

Un vector de sentencias es un array de objetos sentencia, por ejemplo para el programa Prg. A-1 un vector de sentencias posible podría ser :

Cabeza de la cláusula	marron	rojo	rojo	amarillo	amarillo	azul
Indice	1	2	3	4	5	6

Vec. A-1

Cada celda del vector contiene un objeto sentencia, en este ejemplo se indica sólo el nombre de la cabecera de la cláusula, con el objetivo de explicar el formato.

Tabla de referencias a predicados

Una tabla de predicados es una lista indexada de predicados indicando para cada cláusula la posición en el vector de sentencias de los predicados que la componen, continuando con el programa Prg. A-1 y considerando el vector de sentencias Vec. A-1 , la tabla de predicados sería :

Name	**ID**	**Posición**
marron	1	1

[2] Vector de sentencias es una estructura que forma parte de la NPAM, conceptualmente es un array donde cada elemento es una sentencia.

rojo	2	2,3
amarillo	3	4,5
azul	4	6

donde **Name** es el nombre del predicado, **ID** : es un identificador único del predicado, **Posición** : es una lista : son las posiciones del predicado en el vector de sentencias.
La función que cumple esta tabla dentro de la NPAM es almacenar las direcciones de los predicados para acceder directamente a ellos en el proceso de unificación, para esto se implementó un Objeto "Tabla de Predicados" con una función que retorna la lista de índices donde se encuentran los predicados requeridos para unificar. El identificador del predicado coincide con la posición del predicado en la tabla, de forma que no se requiere realizar una búsqueda para obtener las posiciones.

Representación de una sentencia (cláusula)

Un objeto sentencia Neupro (o cláusula Neupro) tiene la siguiente estructura conceptual

Index	**Rp**	**Aridad**	**Peso**
	Variables Indexadas		
	Referencias a predicados		

Elementos de una cláusula.

Se describen a continuación cada uno de los elementos indicados en el gráfico :

- **Index:** es un índice que identifica la cláusula (o sentencia), este índice indica el orden dentro del vector de sentencias, es el índice de la cláusula en el vector de sentencias.
- **Rp:** "Referencia a Predicado", es una referencia a la cabeza de la cláusula en la tabla de predicados. Este predicado es la cabecera de la sentencia, la parte derecha del if (:-)
- **Aridad**: Es la cantidad **n** de variables de la cabeza de la cláusula (predicado de la cabeza), este número se utiliza para indicar cuales son las variables que pertenecen a la cabeza en la lista de variables indexadas, las variables entre el primer elemento y **n** son las variables del predicado.
- **Peso**: Es el valor del peso asignado al predicado, puede ser una variable libre.
- **Variables Indexadas:** es una lista de variables que forma el entorno de sentencia, desde la primer variable hasta la indicada por el número "Aridad" : son las que pertenecen al predicado **Rp.**
- **Referencias a predicados:** es una lista de elementos, la lista guarda, en cada elemento, las posiciones de los predicados, los pesos y las variables que forma parte de cada uno, un elemento de la lista indica la posición de un predicado en el vector de sentencias y las posiciones de las variables que lo forman en las Variables Indexadas. La siguiente tabla hace referencia a esta descripción.

Predicado 1		Predicado 2			Predicado N	
Id pred.	**Posición Variables**	**Id pred.**	**Posición Variables**		**Id pred.**	**Posición Variables**
ID Pred1, peso	Ind Var1	ID Pred2, peso	Ind Var1		ID PredN, peso	Ind Var1
	Ind Var2		Ind Var2			Ind Var2
	.....		.....			.....
	Ind VarN		Ind VarN	...		Ind VarN

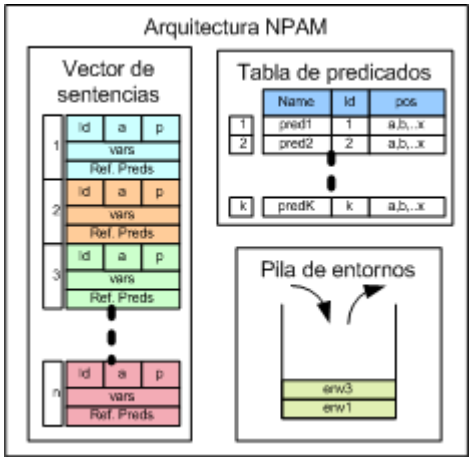
El gráfico ilustra las referencias a los predicados de la sentencia, el campo **Id Pred** es un número que identifica al predicado de forma única y coincide con el **Id** del vector de sentencias, también coincide con el índice de entrada a la tabla de predicados.

.

NeuPro Abstract Machine

Los componentes principales de la NPAM son : un vector de Sentencias-Predicados (cláusulas), una tabla de referencias a predicados y una pila de entornos de sentencias.

La NPAM es una máquina secuencial, en cada paso trata de unificar dos entornos de sentencia o parte del entorno de sentencia, para ésto recorre una vez el vector de sentencias, unificando cada sentencia con la ayuda de la tabla de predicados.
La tabla de predicados indica las posiciones de los predicados donde posiblemente unifique parte del entorno de sentencia.
El gráfico conceptual a continuación muestra la NPAM y sus componentes :



Para mostrar como se cargan estas estructuras se realizará un ejemplo con el programa siguiente :

1. salida(W):35 :- entrada1(X,W):20, entrada2(Y,W):10, entrada3(X,Y):P.
2. salida(W):35 :- salida(Y):20, entrada1(Y,W):20.
3. entrada1(20,9):P.
4. entrada2(10,2):P.
5. entrada3(20,10):12.

Este programa genera las siguientes estructuras :

Tabla de sentencias (cláusulas)

1			1			35		
W			X			Y		
2	20	2 3	3	10	3 1	4	P	2 3
2			1			35		
W			Y					
1	20	2	2	20	2 1			
3			2			P		
20			9					
4			2			P		
10			2					
5			2			12		
20			10					

Tabla de predicados :

1	salida	1	2
2	entrada1	3	
3	entrada2	4	
4	entrada3	5	

Conclusión

Se diseñó y programó un intérprete para el lenguaje Neupro; utilizando algunas ideas de la WAM se confeccionó un algoritmo que permite ejecutar código NeuPro directamente sin necesitar de una traducción a Prolog.

Uno de los nuevos objetivos es analizar la eficiencia del intérprete, así como también realizar diferentes pruebas para determinar los pesos de distinta forma dentro del compilador. Un ejemplo interesante sería determinar el valor de un peso como una variable en un predicado de la siguiente manera :

predicado1(X,[Y Z]):P :- predicado2(X,Y):7, predicado3(P,Z):1.

Se observa que el peso denominado "P" forma parte de predicado "predicado3". Por otro lado se observa que el algoritmo implementado junto con los objetos implementados son aptos para realizar un compilador de Hilog.

Bibliografía

1. Hassan Ait-Kaci: Warren's Abstract Machine, A Tutorial Reconstruction. The MIT Press 1991, ISBN 0-262-01123-9, UB Trier 72 GC f7140
2. The WAM Definition and Compiler Correctness - Egon B.orger, Dean Rosenzweig
3. A Verified Prolog Compiler for the Warren Abstract Machine - David M. Russino
4. Compiling Prolog From the PLM to the WAM and Beyond -Joshua S. Hodas
5. Using the GNU Compiler Collection - Richard M. Stallman – 2002
6. The GNU C Programming Tutorial - Mark Burgess, Ron Hale-Evans
7. The GNU C++ Iostream Library - Per Bothner Cygnus Support
8. The GNU C Library Reference Manual - Sandra Loosemore wich Richard M. Stallman, Roland McGrath, Andrew Oram, and Ulrich Drepper
9. C/C++Language Reference - IBM VisualAge C++ Professional for AIX