GILENA: GENERADOR DE INTERFACES EN LENGUAJE NATURAL

John Atkinson A. Javier Cañas R.

Departamento de Informática
Universidad Técnica Federico Santa María
Valparaíso, Chile

RESUMEN

En este trabajo se describe una herramienta de software llamada GILENA, que permite facilitar la tarea de diseñar interfaces en lenguaje natural.

GILENA se basa en un modelo lingüístico llamado Gramática Sistémica, aunque también es posible usarla en otros modelos debido a que su núcleo principal es un intérprete de ATN.
Se describe el modelo gramatical, sus facilidades, uso y estructura.

La experiencia obtenida con la herramienta se obtuvo de una aplicación práctica que previamente había sido desarrollada con métodos tradicionales. Se pudo constatar que a pesar que se genera una sobrecarga en el software, reduce en un porcentaje muy importante los tiempos involucrados en el ciclo de vida de una aplicación.

1. INTRODUCCION

Las interfaces en lenguaje natural (ILN) resultan atractivas en ambientes tales como: sistemas de consultas, pequeñas bases de datos, programas de aplicación orientados hacia usuarios inexpertos etc...

Lo que no resulta natural, es el diseño de estas interfaces. Diversos factores hacen compleja y tediosa esta tarea: el diseñador se debe relacionar con aspectos computacionales y lingüísticos, y por otro lado, la mantención del software es aún más complicada que el diseño, ya que los usuarios en forma rápida tienden a hacer fuertes exigencias a la interfaz una vez que la dominan.

GILENA es una herramienta que permite automatizar el ciclo de vida de las ILN basadas en enfoques ligüísticos. GILENA es un intérprete de ATN [Wood7Ø] al cual se le han incorporado facilidades de programación. Genera código C y hace posible ligar la interfaz con otras aplicaciones.

2. MODELO GRAMATICAL

La mayor dificultad que presentan las ILN, es que aún en ambientes muy restringidos, el universo lingüístico suele ser bastante extenso. Por esta razón en los últimos años se han realizado esfuerzos para desarrollar formalismos y metodologías que permitan un manejo computacional eficiente del lenguaje.

GILENA tiene su fundamento en la Gramática Sistémica desarrollada por Halliday [Wino73], sin embargo también es posible adaptar GILENA a modelos que usen la Gramática Transformacional y Gramática de Estructura de Frase.

La Gramática Sistémica tiene sus raíces en la antropología y sociología. En este enfoque se considera al lenguaje como una actividad social, razón por la cual el contexto del discurso tiene una gran importancia.
Se considera al lenguaje en términos de distintos estratos: fonológicos, morfológicos, sintáctico y semántico. El análisis completo del discurso requiere del análisis en los diferentes estratos, pero como existe una fuerte relación entre ellos, en la comprensión de oraciones por parte del computador, basta considerar el nivel sintáctico y semántico.

En el nivel sintáctico se hace una nueva división en tres niveles: la palabra, los sintagmas y las oraciones. Las palabras se agrupan en clases de acuerdo a su categoría léxica (nombres, verbos, preposiciones, etc...), los sintagmas se agrupan en categorías sintácticas (nominal, verbal, preposicional).

Para ilustrar este método de análisis se considera en la Fig Nº1 la oración: "El futbolista murió en el hospital"

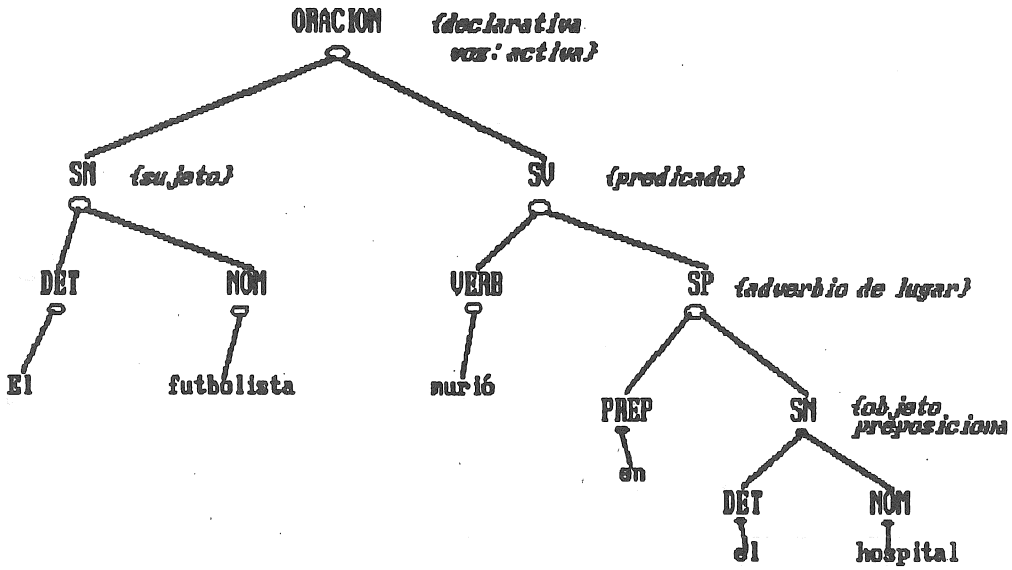

FIG Nº1

La estructura arbórea muestra los diferentes niveles del análisis, es decir palabras, sintagmas y oraciones. GILENA permite a partir de una oración o frase de entrada, generar este tipo de estructura para, a partir de su conocimiento sintáctico a través de este análisis, obtener información semántica.

3. PRESENTACION DE GILENA

En vez de definir formalmente el lenguaje GILENA, lo presentaremos a través de ejemplos.

Como primer ejemplo, consideremos el lenguaje:

$$L = \{a^n b^n c^n / n >= \emptyset\}$$

El ATN que reconoce palabras válidas de L se muestra en la figura Nº2.

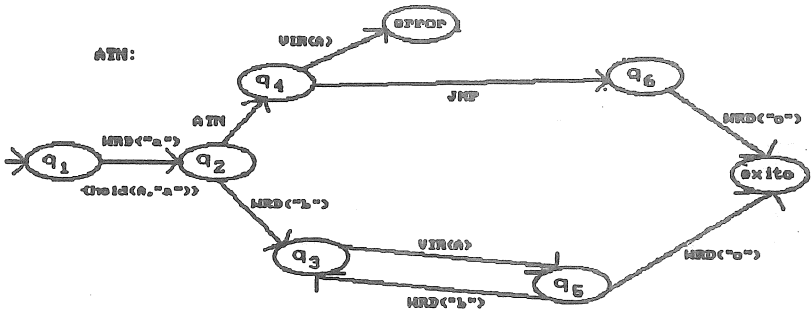

FIG N°2

El estado inicial es q1. El arco rotulado WRD("a") verifica si el primer simbolo leido corresponde al simbolo "a". En caso positivo consume el simbolo, pasa al estado q2 y como efecto lateral guarda el simbolo "a" en el stack A. Esto se realiza mediante la acción hold.
Si el próximo simbolo es "b", se pasa a q3 consumiendo el simbolo. En caso contrario se activa el arco rotulado ATN cuyo efecto es invocar recursivamente al mismo ATN.
El arco VIR(A) hace un POP del stack global A y en caso que no esté vacio, pasa al próximo estado. El arco JMP pasa incondicionalmente al próximo estado.
Una conclusión importante que muestra este ejemplo es que el poder computacional de un ATN es equivalente a una máquina de Turing ya que el lenguaje L es sensible al contexto.

La expresión de GILENA de este ATN es :

```
ATN:
        q1 -> WRD("a") {hold(A,"a");} to q2;
        q2 -> ATN to q4 |
              WRD("b") to q3;
        q3 -> VIR(A) to q5;
        q4 -> VIR(A) to error |
              JMP to q6;
        q5 -> WRD("c") to exito |
              WRD("b") to q3;
        exito -> ;
        q6 -> WRD("c") to exito;
        error -> ;
enda
```

Se aprecia la estrecha relación entre el ATN y el correspondiente código en GILENA.

Estructura del Lenguaje

El lenguaje consta de las siguientes 25 palabras reservadas:

dictionary	special	enda
end	symbols	to
structures	useless	jmp
ends	inseparable	cat
type	conjuction	wrd
endt	separator	vir
categories	finish	integer
endc	atn	word
single		

El lenguaje se divide en 5 áreas principales:

```
I     < Declaraciones del usuario: definiciones
        en lenguaje C >
      dictionary
II    < Definiciones del diccionario >
      endd

      special symbols
III   < Especificación de símbolos especiales >
      ends

      atn
IV    < Definición del ATN >
      enda

V     < Procedimientos definidos por el usuario >
```

El área I es opcional. Las variables definidas por el usuario deben estar escritas en código C.
El área II especifica el diccionario de palabras. Se encuentra dividido en categorías sintácticas y además a cada palabra se le asocia sus rasgos de acuerdo a la categoría.
Esta área también es opcional.
Como ejemplo se da un diccionario que incluye las categorías sintácticas verbo, nombre sustantivo y clítico:

```
dictionary
      structures
            type est_verbo
                  tiempo = (pas, pte, fut)
                  numero = (sing, plur, neu)
                  persona = integer
                  modo = word
            endt
            type est_sust
                  tipo = (comun, propio)
                  genero = (mas, fem, neu)
                  numero = (sing, plu, neu)
                  persona = integer
                  humano = (si, no)
            endt
      ends
      categories
            verbo "verbos.cat" type est_verbo
            sust "sustan.cat" type est_sust
            clitico "clitic.cat" type single
      endc
endd
```

Las palabras mismas están almacenadas en los archivos: verbos.cat, sustan.cat y clitic.cat.
Una entrada del archivo verbo.cat podría ser:

listar, pte, sing, 1, indicativo

Una entrada del archivo sustan.cat podría ser:

juan, propio, mas, sing, 3, si

El archivo de clíticos no tiene estructura definida. Por ejemplo:

me
se
le

Es importante considerar que al revisar el diccionario, GILENA verifica si la categoría almacenada es compatible con la estructura definida en él.

El área III es el área de símbolos especiales. Estos son los símbolos que al no poder ser manajados por el diccionario, merecen un trato especial. Un ejemplo sería:

```
special symbols
      useless (por favor, tenga la amabilidad)
      inseparable (tarjeta de crédito; TC; sustantivo)
      conjunction (y, e)
      separator " ;:.!"
      finish (me voy)
ends
```

El Area de símbolos especiales es obligatoria. Esto es así porque la sección "finish" se debe incluir indicando la secuencia que provocará el término de la sesión. Las demás secciones son opcionales.
Los símbolos "useless" son secuencias de palabras que no aportan información semántica. Al marcarlos como "useless" son eliminados del proceso de análisis.
Los símbolos "inseparable" son secuencias de palabras que por su naturaleza deben ser consideradas como un todo. Por ejemplo: Juan Pérez, Tarjeta de Crédito etc...
GILENA contiene funciones predefinidas que permiten un tratamiento automático de conjunciones. Estas funciones son activadas por aquellos símbolos que están declarados como conjunciones.
Los símbolos "separator" son aquellos caracteres que se usarán como separadores de palabras.

El Area IV es el núcleo principal de GILENA porque especifica el ATN correspondiente a la gramática del lenguaje. Un segundo ejemplo de ATN correspondiente a una gramática arbitraria se muestra en la figura Nº3:

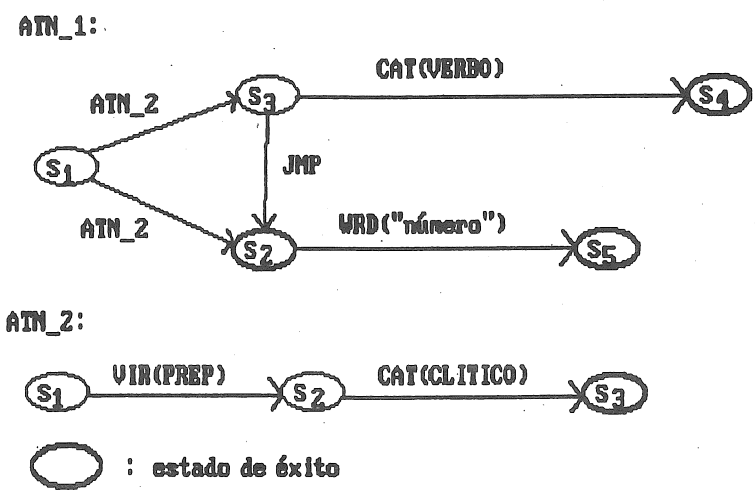

FIG Nº3

La especificación en GILENA de esta red es:

```
atn
      atn_1 :
                  s1 -> atn_2 to s3 |
                        atn_2 to s2
                        ;
                  s2 -> wrd("número") to s5 |
                        ;
                  s3 -> cat(verbo)
                        {
                        /* ejemplo de efectos
                           laterales */
                        printf("atnword vale ");
                        printf("%s\n",atnword);
                        analice(atnword);
                        } to s4 |
                        jmp to s2
                        ;
                  s4 -> ;
                  s5 -> ;
      &
      atn_2 :
                  s1 -> vir(prep) to s2 ;
                  s2 -> cat(clitico) to s3;
                  s3 -> ;
enda
```

El área V es la llamada área de procedimientos del usuario. Su finalidad es permitir a los usuarios desarrolladores, escribir sus propios procedimientos en lenguaje C.
Existe un procedimiento predefinido llamado atnparse, el cual es encargado de administrar y controlar el avance del ATN. La definición de atnparse es:

```
int atnparse(frase_entrada)
char *frase_entrada;

retorna:
      -1: si hubo error sintáctico
      +1: si se analizó correctamente la frase
       2: si se reconoció el símbolo final de la
          sesión
```

Una llamada típica tiene la siguiente estructura:

```
atn
........
enda

main()
{
char frase[80];
int status;
      do {
            printf("Ingrese Consulta: ");
            gets(frase);
            status = atnparse(frase);
            switch (status) {
                  case -1 : Error();
                  case 1  : generar_respuesta();
                  }
      } while (status != 2);
      printf("Fin sesión ..\n");
```

4. USO DE GILENA Y FACILIDADES DE DEPURACION

GILENA provee dos modos de entrada: a través de un archivo o de la entrada estándar. Si se especifica un archivo el formato es:

gilena [-t] <arch_especif>

la opción -t, indica que se debe generar una traza para la ejecución de la interfaz, y <arch_especif> corresponde al nombre del archivo de especificación de entrada.
Bajo cualquier condición aparece en la salida estándar mensajes que indican las áreas de análisis:

```
Analizando área diccionario.......
Analizando área de símbolos especiales.......
Analizando área de ATN.........
Compilación OK......
```

GILENA es una herramienta para ser utilizada por desarrolladores de interfaces, por esta razón se le ha incorporado facilidades para realizar trazas del recorrido del ATN en el análisis de las oraciones.
La traza se indica con la opción -t. La traza se utiliza para el recorrido de los arcos del ATN. Para avanzar al siguiente estado en esta modalidad, el usuario debe responder 'y' para continuar o 'n' para finalizar. Como resultado del recorrido de un arco se puede obtener: TRUE si la condición es verdadera, FAIL si falla y REDO si la condición falla y se debe realizar un backtracking.

Existen una serie de mensajes antes de atravesar cada arco. A modo de ejemplo, parte de una sesión con el ATN de la figura Nº3 sería:

```
Starting ATN atn_1
PUSHing ATN atn_1 to state s1; Continue (y/n)? y
      in state s1
PUSHing ATN atn_2 to state s1; Continue (y/n)? y
      in state s1


.................
Trying an WRD arc to state s5, expecting ("número");
Continue (y/n)? y
..............
```

5. EXPERIENCIA CON GILENA

Para probar GILENA como herramienta de desarrollo de interfaces, se realizó una aplicación que consistió en un sistema de consultas automatizadas para clientes de un Banco. Se eligió esta aplicación porque ya existía experiencia previa en este universo lingüístico a través de desarrollos en C, Lisp y PROLOG.

No se hicieron mediciones cuantitativas de ingeniería de software, pero el tiempo de desarrollo fue sustancialmente menor. Si bien el código generado es más grande que la versión escrita en C, una vez diseñada la interfaz resultó muy fácil introducir modificaciones y ampliar el universo.

Una sesión típica con esta interfaz es:

```
C) Por favor, me puede dar mi saldo total
R) Su saldo total es $856.987

C) Cuáles son los últimos cheques cobrados?
R) Los 4 últimos cheques cobrados son: 7091, 7095,
     7092, 7090

C) Saldo?
R) Saldo total o disponible?
C) Disponible y total!
R) Su saldo disponible es $750.432 y el total es $856.978

etc...
```

donde C) indica la consulta del cliente y R) la respuesta de la interfaz.
De este ejemplo se puede visualizar también que si bien las consultas son simples, involucran aspectos lingüísticos complejos como tratamiento de elipsis y conjunciones.

6. DISEÑO DE GILENA

La arquitectura de GILENA se puede ver en la figura Nº4:

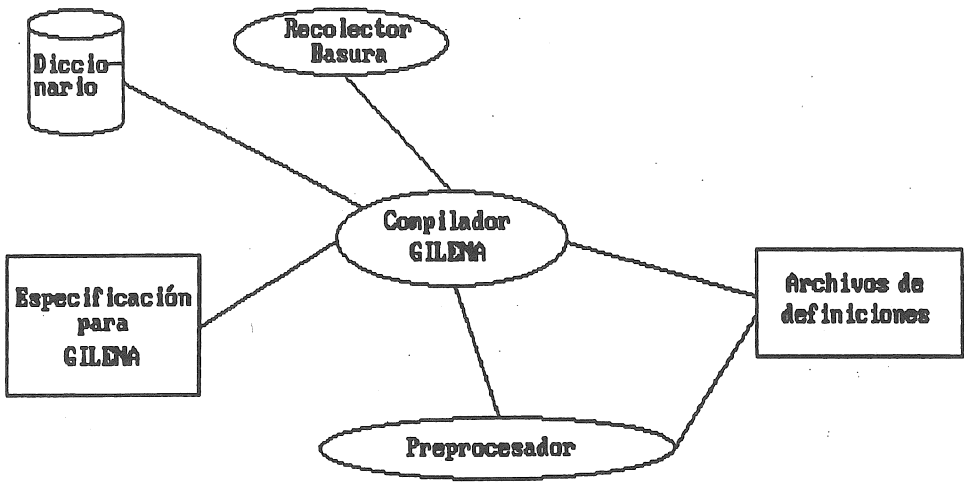

FIGURA Nº4

El núcleo de esta arquitectura es el compilador GILENA. Este compilador es el encargado de recibir el archivo de especificación, analizarlo y generar el código fuente para la interfaz.
El compilador GILENA genera código C. Por esta razón necesita información de librerías y de archivos de definición.
El preprocesador tiene por función reemplazar secuencias de símbolos para facilitar la tarea del compilador.
El recolector de basura corresponde a un módulo que contiene las funciones que permiten optimizar el uso de la memoria.
El diccionario son archivos ASCII, uno por cada categoría sintáctica para facilitar su mantención.

7. CONCLUSIONES

El diseño y construcción de GILENA forma parte de una línea de investigación del Departamento de Informática de la Universidad Técnica Federico Santa María.
GILENA es el resultado de la experiencia lograda a través de varios proyectos en los cuales se experimetó con varias metodologías, ambientes y lenguajes.
La conclusión más relevante es que como herramienta es útil y

permite una fácil experimentación con gramáticas. Como el desarrollo se realizó en lenguaje C se hicieron también experiencias de portabilidad entre estaciones SUN y computadores personales. La experiencia en el desarrollo y utilización de GILENA también permitió detectar sus debilidades. Aspectos de definición del lenguaje y eficiencia del compilador se han considerado para una segunda versión.

REFERENCIAS BIBLIOGRAFICAS

[Aho86] Aho Alfred, Ullman Jefrey, "Compilers: Principles, Techniques and Tools", Addison-Wesley, 1986

[Bolc86] Bolc Leonard, "The Design of Interpreters, Compilers and Editors for Augmented Transition Networks", Springer-Verlag, 1983

[Caña85] Cañas Javier, "Interface en lenguaje natural para consultas interactivas", Universidad de Chile, 1985

[Cis78] Cisternas Ricardo, " Creación y evaluación de métodologia para construcción de interfaces en lenguaje natural", Universidad de Chile, 1987

[Yang9Ø] Yang Gi-Chul, "On deterministic control of Augmented Transition Networks", X conferencia Chilena de Ciencias de la Computación, Chile, 199Ø

[Wino73] Winograd Terry, "Research in Natural Languages", Artificial Intelligence Laboratory, MIT, Mayo 1973

[Wood7Ø] Woods W, "Transition Networks Grammars for natural language analysis", Communications of the ACM, Octubre 197Ø