Logic-Web
Un ejemplo teórico-práctico del uso de PARLOG

Carlos Matesanz Rodríguez

- ¿Qué es Logic-Web?

  Es un modelo de programación concurrente lógica para web.
- ¿En que se basa?

  Considera la web como una colección de programas lógicos que pueden componerse para formar nuevas entidades.
- ¿Qué ventajas tiene?

  El alto nivel de representación que enfatiza las relaciones lógicas.
- ¿Qué desventajas tiene?

  Como siempre que se habla de programación declarativa se le puede achacar falta de eficiencia. En este caso en particular no tanto en consumo de CPU sino en ancho de banda.
- ¿Cómo está implementado?

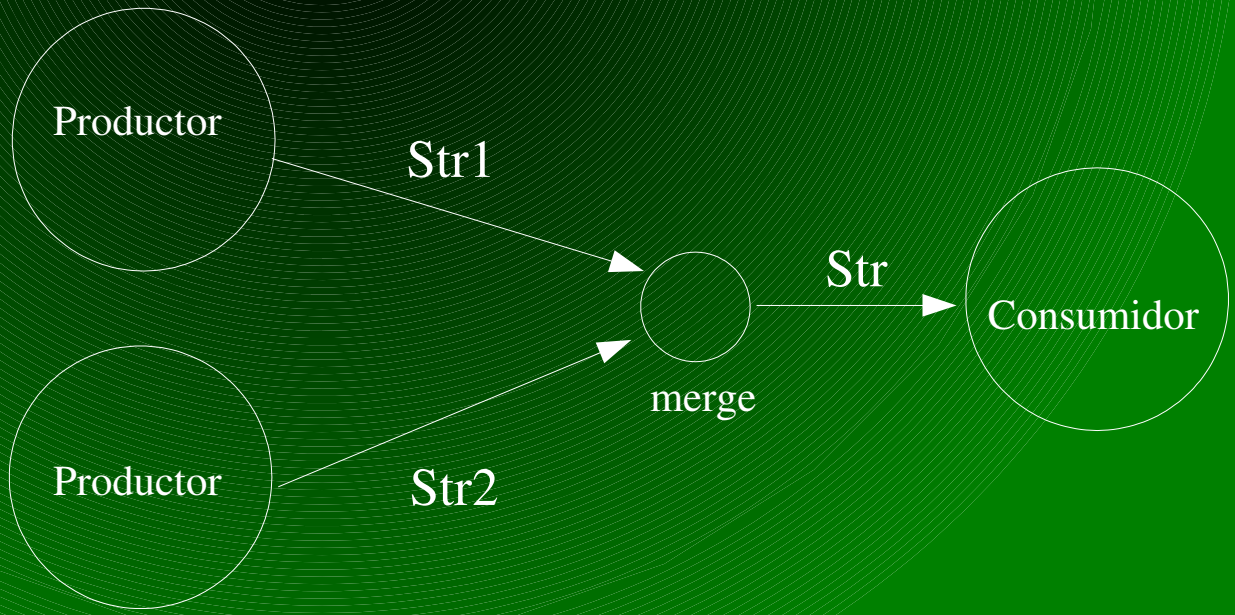
  Combina PARLOG con módulos en C para las operaciones de bajo nivel.

El Predicado merge/3

```
mode merge( ? , ? , ^ ).

merge( [ El | X ], Y, [El | Z]) :- merge(X,Y,Z).
merge( X, [El | Y], [ El | Z]) :- merge(X,Y,Z).
merge([], Y, Y).
merge(X,[],X).
```

?- producer(Str1), producer (Str2), merge(Str1,Str2,Str),consumer(Str)



El predicado download/4

mode download(Request?, Text^, Result^, Stop?).

Request: req(get, 'http://www.yonkis.com')

Text: devolverá la cadena ASCII correspondiente a la página web

Result: será 'ok' si todo va bien o err(msg) si ocurre algún error.

Stop: si se unifica (se “instancia”), el proceso se detiene.

El predicado download/4 (2)

```
download(req(get, URL), Text, Result, Stop) :-
    make_fnm(Fnm),
    isd(URL, Fnm, Text, Result),
    set_stop(Stop, Result, Fnm).
```

make_fnm/1 : crea un nombre de fichero único que utilizan isd y set_stop

isd/4 : invoca al módulo en C y obtiene en Text la página web solicitada.

set_stop/3 : está suspendido hasta que se unifica o Stop o Result (es decir, hasta que isd termina) entonces crea el archivo de stop y termina.

Uso de download/4

```
?- download( req(get, 'http://www.lcc.uma.es'),T1,R1,_),
     download( req(get, 'http://www.matap.uma.es'), T2,R2,_).
```

```
mode or_get( ? , ^ ).
or_get( [URL | Ds], T ) :- download(get(req, URL), Text, ok, _ ) : T=Text.
or_get( [_ | Ds], T ) :- or_get ( Ds, Text) : T=Text.
```

```
?- or_get(['http://www.mirror1.com','http://www.mirror2.com'], Text).
```

Descarga con Timeout

```
mode timeout(?, ?, ^, ^, ?).

timeout(Time, Request, Text, Result, Stop) :-
    download(Resquest, Text, Result, Stop) : true.

timeout(Time, _, _, err(timeout), _) :-
    sleep(Time) : true.
```

Descarga con reintentos

```
mode repeat( ?, ?, ^, ^, ?).

repeat(Limit, Request, Text, Result, Stop) :-
    repeat1(0, Limit, Request, Text, Result, Stop).


mode repeat1(?, ?, ?, ^, ^, ?).

repeat1(Limit, Limit, _, _, err(limit), _).
repeat1(Count, Limit, Request, Text, ok, Stop) :-
    Count < Limit , download(Request, Text, ok, Stop) : true ;
repeat1(Count, Limit, Request, Text, ok, Stop) :-
    Count<Limit, Count1 is Count+1,
    repeat1(Count1, Limit, Request, Text, ok, Stop).
```

Combinando predicados.
Descarga de varios mirrors con timeout.

```
mode or_time( Time?, ListaMirrors?, Text^).

or_time(Time, ListaMirrors, Text) :-
    or_get(ListaMirrors, Text) : true.

or_time(Time, _, _) :-
    sleep(Time) : true.
```

Bibliografía y trabajos posteriores

Lo que acabamos de ver está (o está basado) en el artículo de Andrew Davison y Seng Wai Loke “A Concurrent Logic Programming Model of the Web”, de 1998.

Existen publicaciones más recientes que citan éste artículo en sus bibliografías, como por ejemplo “Logic Object-Oriented Model of Asynchronous Concurrent Computations” de A. A. Morozov, en el que se plantea un modelo de programación lógica de Agentes de Internet. Extiende las ideas de Davison y Loke para cubrir las necesidades propias de la programación de agentes.

Este artículo tiene fecha de 2003, lo que quiere decir que ésta no es una rama de investigación muerta.