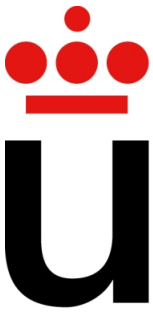


Universidad Rey Juan Carlos

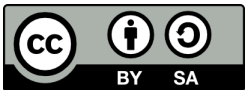
ESCUELA TÉCNICA SUPERIOR
DE INGENIERÍA INFORMÁTICA

DeduccionNatural.pl

MANUAL DE USUARIO

VERSIÓN 2022

Autores: **Iván Ramírez**
Joaquín Arias



http://hdl.handle.net/10115/20168

DeduccionNatural.pl *es un programa escrito en Ciao Prolog, en el contexto de la asignatura de Lógica en la Escuela Técnica Superior de Ingeniería Informática de la Universidad Rey Juan Carlos. El programa está disponible en* `https://github.com/Xuaco/DeduccionNatural` *y un snapshoot de la versión 2022 (a la que corresponde el presente Manual) está archivada en Software Heritage con identificador swh:1:dir:69fac171d014bb46c5bd7818a65806ec454812bb.*

1. Introducción

El objetivo de *DeduccionNatural.pl* es facilitar el aprendizaje de la Deducción Natural, introducida por Gentzen, 1935. La Deducción Natural busca capturar la manera en que las personas razonan naturalmente al construir demostraciones matemáticas. En vez de contar con unos pocos axiomas a los que se aplican unas pocas reglas de inferencia, la deducción natural propone vaciar la lista de axiomas y ampliar la de reglas de inferencia, introduciendo dos reglas para cada constante lógica ($\wedge$, $\vee$, $\neg$, $\rightarrow$, $\leftrightarrow$): (i) Una para introducirla, (ii) Otra para eliminarla. Por lo tanto, una demostración se construye partiendo de unas premisas y aplicando las reglas para llegar al consecuente correspondiente. Sirve para demostrar la validez de un argumento.

Para facilitar el aprendizaje hemos desarrollado *DeduccionNatural.pl* (disponible en `https://github.com/Xuaco/DeduccionNatural`), que permite evaluar si, dada unas premisas y un consecuente, la demostración planteada por el alumno es correcta. La Fig. 1 muestra una captura de la ejecución, en el Playground de Ciao Prolog[1], del ejemplo 1.

Ejemplo 1 *Ejemplo extraído del Tema 2, pág. 46, de los apuntes de Arias, 2022.*

Desarrollo con notación lógica:

$T[s \wedge (p \vee q), p \rightarrow \neg r, q \rightarrow \neg r] \vdash s \wedge \neg r$

1.	$s \wedge (p \vee q)$	*premisa*
2.	$p \vee q$	$E_{\wedge}(1)$
3.	$p \rightarrow \neg r$	*premisa*
4.	$q \rightarrow \neg r$	*premisa*
5.	$\neg r$	$E_{\vee}(2,3,4)$
6.	s	$E_{\wedge}(1)$
7.	$s \wedge \neg r$	$I_{\wedge}(5,6)$

Traducción en `DeduccionNatural.pl`:

```
ejemplo1 :-
    main([ s and p or q, p ->
! r, q -> ! r ], s and ! r, [
          'Premisa'(1),
          'E' and b(1),
          'Premisa'(2),
          'Premisa'(3),
          'E' or (2,3,4),
          'E' and a(1),
          'I' and (6,5)
          ]).
```

A continuación, la Sección 2 describe como se han traducido los operadores y las reglas de inferencia. La Sección 3 explica como ejecutar el programa para comprobar

[1] Ciao Prolog es un intérprete de Prolog, disponible en `https://ciao-lang.org/`, que cuenta con un Playground online en `https://ciao-lang.org/playground/index.html`

si una demostración es correcta dado unas premisas y un consecuente. Finalmente, la Sección 4 muestra como definir reglas derivadas (o de usuario) para factorizar las demostraciones.

2. Constantes lógicas y reglas de inferencia

Como hemos visto en los apuntes de la asignatura de Lógica, redactados por Arias, 2022, en el contexto de la lógica proposicional, contamos con un alfabeto compuesto por símbolos de proposición y conectivas. Para facilitar la representación de dichos símbolos en Prolog hemos decidido realizar la siguiente representación:

- Los símbolos de proposición (que representan hechos), se representan como átomos en Prolog, palabras que empiezan por minúscula (con o sin números), p.ej., $p, q, r, \ldots, p1, p2, \ldots$.
- Las 5 conectivas lógicas se traducen de la siguiente manera:

Notación Lógica	$\neg$	$\vee$	$\wedge$	$\rightarrow$	$\leftrightarrow$
DeduccionNatural.pl	`!`	`or`	`and`	`-->`	`<->`

La preferencia de los operadores está definida en las líneas 11-13.[2]

De igual manera hemos traducido las 10 reglas de inferencia básicas (2 por cada conector lógico). A continuación mostramos la sintaxis de dichas reglas según la implementación de DeduccionNatural.pl e indicamos donde se encuentra la implementación correspondiente:

Conector	Introducción	Eliminación	Implementación
`and`	`'I' and (_,_)`	`'E' and a(_)` `'E' and b(_)`	Líneas 127-138
`or`	`'I' or a(_,F)` `'I' or b(F,_)`	`'E' or (_,_,_)`	Líneas 139-151
`!`	`'I' ! (_)`	`'E' ! (_)`	Líneas 152-159
`-->`	`'Supuesto'(F)` `'I' --> (_,_)`	`'E' --> (_,_)`	Líneas 160-176
`<->`	`'I' <-> (_,_)`	`'E' <-> a(_)` `'E' <-> b(_)`	Líneas 177-176

Nótese que las reglas de inferencia están en mayúsculas y, por lo tanto, para que Prolog las trate como átomos (y no como variables) hay que usar comillas, i.e., `'I'`. Dependiendo del número de fórmulas que intervienen en cada regla de inferencia tenemos diferente número de argumentos en la implementación, p. ej. para la negación tanto la introducción como la eliminación solo precisa de una fórmula, y para la eliminación

[2] El operador `!` es unario, está también definido como binario para definir las reglas de inferencia.

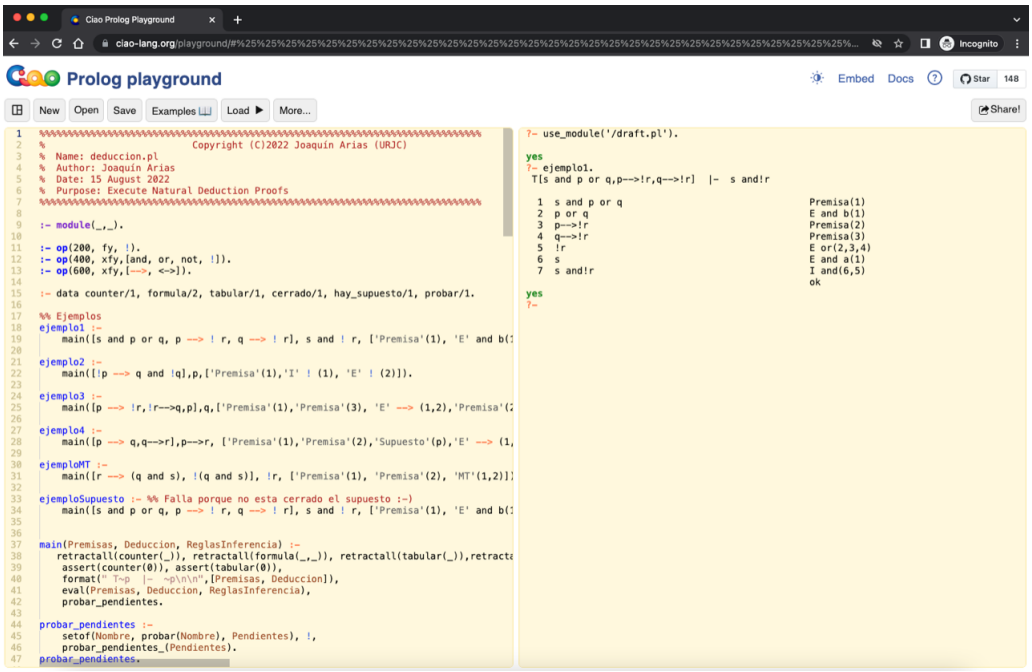


Figura 1: Captura de pantalla con la ejecución de `DeduccionNatural.pl`

de la disjunción nos encontramos con tres. Adicionalmente, las reglas de inferencia, `'I' or a(_,F)`, `'I' or b(F,_)`, y `'Supuesto'(F)`, permiten al usuario introducir nuevas fórmulas. En este caso hemos usado la sintaxis de Prolog para las variables, i.e., `F`, que en cada caso se instanciará con la fórmula pertinente.

3. Ejecución del programa

Para ejecutar DeduccionNatural.pl recomendamos utilizar el Playground de Ciao siguiente el siguiente enlace `https://tinyurl.com/deduccionnatural22`. Como hemos comentado anteriormente evaluando la consulta `?- ejemplo1` se obtiene el resultado mostrado en la Fig. 1. Para evaluar un ejercicio basta introducir en el editor del Playground una nueva cláusula de la siguiente manera:

Ejemplo 2 *Supongamos que queremos usar la regla* Modus Ponens *o eliminación de la implicación. Primero escribimos la cabeza de la regla (p. ej. indicando el ejercicio,* `ejercicio1`*) y luego invocamos el predicado* `main/3`*, que recibe tres argumentos: (i) una lista con las premisas, (ii) el consecuente, (iii) una lista con las reglas de inferencia a aplicar para* demostrar *la deducción natural.*

```
ejercicio1 :-
    main([ p -> q, p], q, [
        'Premisa'(1),
        'Premisa'(2),
        'E' -> (1,2)
        ]).
```

Para poder aplicar las reglas de inferencia hay que bajar *las premisas usando el predicado* `'Premisa'(_)`*. Las premisas se numeran según su posición en la lista de premisas (empezando por* 1*). Sin embargo, las fórmulas involucradas en las reglas de inferencia se referencia considerando la posición que ocupa la regla que* generó/bajó *dicha fórmula. Es decir, el siguiente código es también válido:*

```
ejercicio1 :-
    main([ p -> q, p], q, [
        'Premisa'(1),
        'Premisa'(1),
        'Premisa'(2),
        'E' -> (1,3)
        ]).
```

Para facilitar el aprendizaje de DeduccionNatural.pl hemos creado varios tutoriales que están disponibles en TV URJC.

4. Definir reglas derivadas

También es posible definir reglas derivadas. Para facilitar su comprensión en las líneas 208-216 se facilita como ejemplo *Modus Tolens*:

Ejemplo 3 *Recordemos que* Modus Tolens *indica que puede negarse el antecedente de una implicación si se niega su consecuente:*

```
regla('MT',[FA -> FB,!FB],!FA, [
      'Premisa'(1),
      'Premisa'(2),
      'Supuesto'(FA),
      'E' -> (1,3),
      'I' and (4,2),
      'I' -> (3,5),
      'I' ! (6)
      ]).
```

El predicado que utilizaremos para definir reglas derivadas en `regla/4`, que recibe cuatro argumentos: (i) nombre de las reglas, (ii) lista de premisas, (iii) consecuente, (iv) lista de reglas de inferencia básicas que *demuestran* la validez de la regla. Obsérvese que en las reglas derivadas nos referimos a las fórmulas en mayúsculas para indicar que son variables y que las instanciaremos con las fórmulas que correspondan según el ejemplo o ejercicio correspondiente.

Ejemplo 4 *Para utilizar una regla derivada podemos proceder como con una regla de inferencia básica. En este caso, el número de argumentos y su orden viene determinado por en número y orden de las fórmulas en la lista de premisas definida en la implementación de la regla.*

```
1 ejemploMT :-
2     main([r -> (q and s), !(q and s)], !r, [
3         'Premisa'(1),
4         'Premisa'(2),
5         'MT'(1,2)
6         ]).
```

Si implementa nuevas reglas derivadas y estas no están escritas a continuación de la regla de *Modus Tolens* (línea 220), el compilador de Ciao emitirá un mensaje de advertencia indicando que algunas cláusulas del predicado `regla/4` no son contiguas:

```
{Reading /draft.pl
WARNING: (lns xxx-xxx) clauses of regla/4 are discontiguous
}
```

El programa funcionará correctamente, la advertencia tiene como finalidad evitar repetir el nombre de un predicado ya implementado.

Aunque la posibilidad de introducir reglas derivadas es muy tentadora, es importante observar que DeduccionNatural.pl comprobará que las definiciones de las reglas derivadas utilizadas en una demostración son correctas.

Ejemplo 5 *La ejecución del Ejemplo 4 produce la siguiente salida:*

```
?- ejemploMT.
 T[r->q and s,! (q and s)]  |-  !r

  1  r->q and s                                    Premisa(1)
  2  ! (q and s)                                    Premisa(2)
  3  !r                                             MT(1,2)
                                                    ok

 Prueba de la regla derivada MT:  T[A->B,!B]  |-  !A

  1  A->B                                          Premisa(1)
  2  !B                                             Premisa(2)
  3       A                                         Supuesto(A)
  4       B                                         E->(1,3)
  5       B and!B                                   I and(4,2)
  6  A->B and!B                                    I->(3,5)
  7  !A                                             I!6
                                                    ok

yes
```

Referencias

Arias, Joaquín (2022). **Lógica: desde Aristóteles hasta Prolog**. Madrid: Servicio de Publicaciones de la Universidad Rey Juan Carlos. ISBN: 978-84-09-38265-1.

Gentzen, Gerhard (1935). **Untersuchungen über das logische schließen. I.** En: *Mathematische zeitschrift* 35.