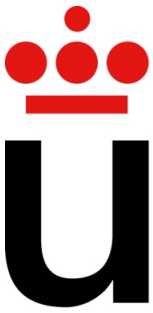
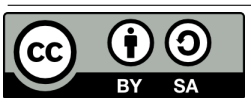


Universidad
Rey Juan Carlos

ESCUELA TÉCNICA SUPERIOR
DE INGENIERÍA INFORMÁTICA

Artículo presentación DeduccionNatural.pl

Autor: Joaquín Arias



Copyright (c) 2023 Joaquín Arias. Esta obra está bajo la licencia CC BY-SA 4.0, Creative Commons Atribuciónn-CompartirIgual 4.0 Internacional.

DeduccionNatural.pl: herramienta escrita en Prolog para el aprendizaje de la asignatura de Lógica

Joaquín Arias joaquin.arias@urjc.es
Iván Ramírez ivan.ramirez@urjc.es
Alessandra Gallinari alessandra.gallinari@urjc.es
Dpto. de Informática y Estadística, ETSII
Área de Matemática Aplicada, ESCET
Universidad Rey Juan Carlos, 28933 Móstoles, Madrid

Resumen

La lógica matemática y computacional son materias básicas en la mayoría de las titulaciones universitarias relacionadas con la Ingeniería Informática, tanto en España como en el extranjero. Son fundamentales en el estudio de bases de datos, complejidad computacional, lenguajes de programación, inteligencia artificial, diseño y verificación de sistemas *hard* y *soft*, entre otros. Sin embargo, para estudiantes de un primer curso no es inmediato reconocer las conexiones entre la lógica y lo que necesitarán aprender en estudios futuros. Para mitigar esta sensación de desconexión entre contenidos básicos y más avanzados, hemos desarrollado DeduccionNatural.pl, un programa con licencia de código abierto escrito en Ciao Prolog (lenguaje de programación lógico), que permite comprobar si una demostración de deducción natural (en lógica proposicional) es correcta. En DeduccionNatural.pl, las demostraciones son programas, las reglas de inferencia son funciones de una librería predefinida, y las reglas derivadas son subrutinas (para refactorizar las demostraciones) definidas por los estudiantes. Durante el curso 2021–2022 hemos utilizado DeduccionNatural.pl en los grados de Ingeniería de Ciberseguridad y de Inteligencia Artificial en la Universidad Rey Juan Carlos y hemos realizado una encuesta de satisfacción. Los resultados no son concluyentes, pero nos animan a seguir mejorando la herramienta.

Abstract

Mathematical and computational logic are basic subjects in most university degrees related to Computer Engineering, both in Spain and abroad. They are fundamental in the study (among other topics) of databases, computational complexity, programming languages, artificial intelligence, design and verification of hard and soft systems. However, for first-year undergraduate students it is not immediate to recognize the connections between logic and what they will need to learn in future studies. To mitigate this feeling of disconnect between basic and more advanced content, we have developed DeduccionNatural.pl, an open source licensed program written in Ciao Prolog (a Logic Programming Language), which allows to check if a natural deduction proof (in propositional logic) is correct. In DeduccionNatural.pl, the proofs are programs, the inference rules are functions of a predefined library, and the derived rules are subroutines (to refactor the proofs) defined by the students. During the academic year 2021-2022, we have used DeduccionNatural.pl in the Cybersecurity Engineering and Artificial Intelligence degrees at Universidad Rey Juan Carlos and we have conducted a satisfaction survey. The results are not conclusive, but they encourage us to continue improving the tool.

Palabras clave

Material docente, Lógica, Prolog, Programación Lógica, Deducción natural.

1. Introducción

El material docente que presentamos en este artículo se enmarca dentro de la asignatura de Lógica, que se ocupa de la ciencia que estudia la validez formal de los razonamientos. Por medio de la formalización del lenguaje y de sus reglas básicas proporciona las herramientas necesarias para poder tratar y resolver rigurosamente problemas que tienen sus orígenes y aplicaciones en todas las áreas de las ciencias [2, 5, 8].

Es, por lo tanto, natural que la lógica matemática y computacional sean materias básicas en la mayoría de las titulaciones universitarias relacionadas con la Ingeniería Informática [1, 3]. Su estudio tiene a menudo la difícil tarea de compensar las posibles carencias de la

En el marco del proyecto Scaffolding Online University Learning: Support Systems (2022-1-IT02-KA220-HED-000090206), financiado por la Comisión Europea-Agencia Erasmus (2022-2025).

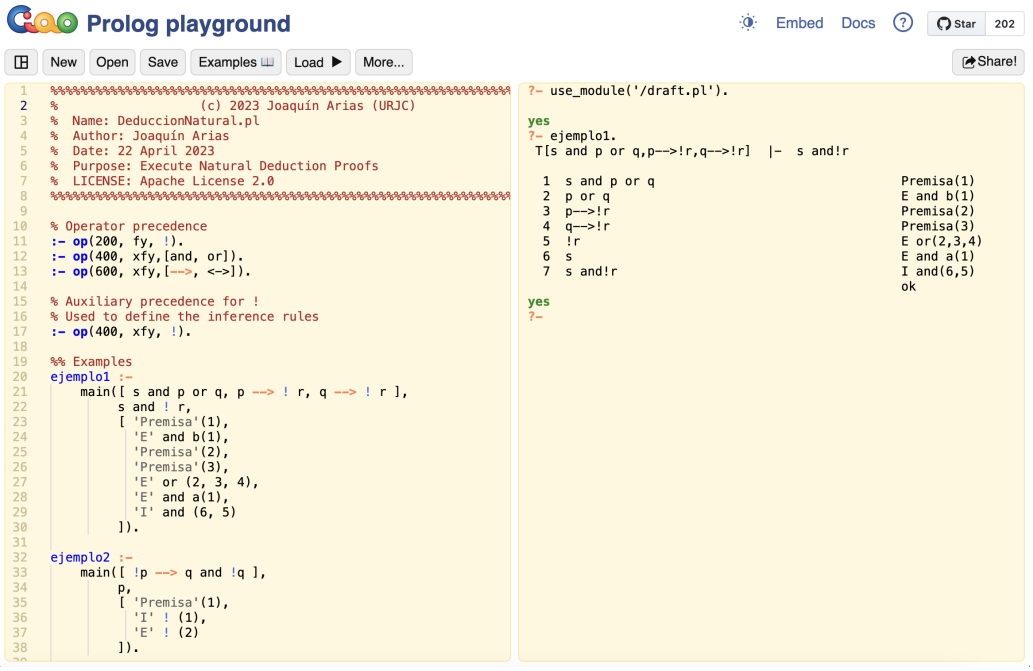


Figura 1: Captura de pantalla con la ejecución de `DeduccionNatural.pl`.

formación previa de los alumnos. Además, sirve para mejorar las tres competencias básicas de lectura, matemáticas y ciencias (evaluadas en el contexto español a través del informe PISA 2018 [12]) y la capacidad de razonar de forma analítica y crítica. Uno de los principales objetivos didácticos de las asignaturas de lógica es "la capacidad para adquirir, formalizar y representar el conocimiento humano en una forma computable para la resolución de problemas mediante un sistema informático en cualquier ámbito de aplicación".

Desde nuestro punto de vista, un objetivo principal de la asignatura de Lógica es el de introducir algunos sistemas de demostración formal. Para la mayoría de los alumnos no es inmediato justificar correctamente la validez de un razonamiento o la resolución de un problema concreto. Los métodos deductivos de la lógica matemática están en la base de la demostración automática de teoremas. Sin embargo, en los últimos años hemos observado que no es fácil conseguir que alumnos de primer curso entiendan la importancia de la lógica y de sus aplicaciones a la programación.

Por su formulación más intuitiva, en nuestras asignaturas decidimos presentar el método de Deducción Natural de Gentzen [2, 5, 6]. La deducción natural busca capturar la manera en que las personas razonan naturalmente al construir demostraciones matemáticas: partiendo de unas premisas y aplicando las reglas de inferencia se demuestra la validez del consecuente correspondiente. Como veremos más adelante, las reglas de inferencia se pueden fácilmente interpretar como funciones predefinidas con una API (Interfaz de Programación de Aplicaciones) concreta, y las reglas derivadas (patrones de demostración basadas en las reglas de inferencia) son entonces subrutinas que el usuario puede definir para reutilizarlas en diversas demostraciones sin tener que repetir un patrón de demostración concreto; sirva como ejemplo *ModusTollens* (ver páginas 51-52 del Tema 2 en [2]).

En este contexto, el objetivo de la herramienta que presentamos, denominada *DeduccionNatural.pl*, es facilitar el aprendizaje de la deducción natural considerando que las demostraciones son programas que invocan funciones predefinidas. *DeduccionNatural.pl* (disponible en `https://github.com/Xuaco/DeduccionNatural`) permite evaluar si, dada unas premisas y un consecuente, la demostración planteada por el alumno es correcta. La Figura 1 muestra una captura de la ejecución, en el Playground de Ciao Prolog[1], del siguiente ejemplo (que usaremos durante el artículo para explicar detalladamente el funcionamiento de DeduccionNatural.pl):

Ejemplo 1. En la página 46 del Tema 2 en [2] se pide demostrar mediante deducción natural la validez de:

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

Es decir, que dadas las premisas $s \wedge p \vee q$, $p \rightarrow \neg r$ y $q \rightarrow \neg r$, la conclusión $s \wedge \neg r$ es consecuencia lógica.

En la Sección 2 presentamos el marco teórico y trabajo relacionado con este trabajo. En la Sección 3

[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`

presentamos la contribución principal de este trabajo que evidentemente es el programa DeduccionNatural.pl. Adicionalmente queremos destacar aquí algunas de sus principales ventajas:

- Disponemos de un manual y vídeos tutoriales para facilitar su aprendizaje.
- Aunque podría implementarse en cualquier lenguaje de programación, el uso de Prolog visualiza una de las más inmediatas aplicaciones de la lógica: la programación lógica.
- Al ser de código abierto se puede adaptar para ajustarlo al planteamiento de otros profesores.
- Además, hemos podido comprobar que es posible incorporar su uso en la asignatura de Lógica, y que al tener una curva de aprendizaje muy baja y no requerir instalación previa, su incorporación se puede realizar sin necesidad de alterar el temario diseñado en las guías docentes.

En la Sección 4 mostramos los resultados de usar DeduccionNatural.pl en el desarrollo de actividades prácticas en el grado de Ingeniería de Ciberseguridad y el grado de Inteligencia Artificial de la Universidad Rey Juan Carlos (durante el curso 2021–2022) y aportamos sugerencias para facilitar su implantación en otras universidades.

2. Marco teórico

En esta exposición asumimos que el lector tiene cierta familiaridad con los contenidos de las asignaturas de Lógica en el contexto de los grados relacionados con la Ingeniería Informática.

En la Sección 2.1 recordaremos la definición de sistema de Deducción Natural de Gentzen para la lógica proposicional. En la Sección 2.2 realizamos una breve descripción de herramientas similares a DeduccionNatural.pl y que se han utilizado o se podrían utilizar en la enseñanza de Lógica.

2.1. Deducción natural de Gentzen

Recordamos que un sistema de demostración formal (o sistema de pruebas) se define matemáticamente como $S = (A, F, X, R)$, donde:

A Es el alfabeto del sistema: el conjunto de símbolos que se pueden utilizar.

F Es el conjunto de reglas de sintaxis: las reglas que permiten definir las fórmulas bien formadas (sintácticamente correctas).

X Es el conjunto de axiomas: las fórmulas válidas por definición.

R Es el conjunto de reglas de inferencia: reglas de transformación que permiten inferir una fórmula, el consecuente (o conclusión) a partir de un conjunto de fórmulas, las premisas.

Una deducción o razonamiento consiste de un conjunto de fórmulas $\Phi = \{\varphi_1, \varphi_2, \dots \varphi_n\}$, llamadas las premisas, y de una fórmula φ, llamada la conclusión de la deducción.

Una deducción es correcta (válida) si la conclusión φ es consecuencia lógica (deducible usando las reglas de inferencia del sistema de demostración) del conjunto de las premisas Φ y se escribe $\Phi \vdash \varphi$.

El sistema de deducción natural de Gerald Gentzen (1934-1935) [6] es el sistema de demostración que mejor modela el razonamiento informal o intuitivo. Para la lógica de proposiciones, este sistema se basa en deducciones y solo tiene diez reglas de inferencia (y ningún axioma). A continuación, presentamos las reglas de inferencia (una de introducción y otra de eliminación para cada conectiva lógica, $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$):

(I $\neg$) Regla de introducción de la negación.
Método de demostración por contradicción.

$$\{\varphi \rightarrow \psi \wedge \neg\psi\} \vdash \neg\varphi$$

(E $\neg$) Regla de eliminación de la (doble) negación.

$$\{\neg\neg\varphi\} \vdash \varphi$$

(I $\wedge$) Regla de introducción de la conjunción.
De dos fórmulas se deduce su conjunción.

$$\{\varphi, \psi\} \vdash \varphi \wedge \psi$$

(E $\wedge$) Regla de eliminación de la conjunción.
De una conjunción se deduce la fórmula de la izquierda (a) o la de la derecha (b).

$$\{\varphi \wedge \psi\} \vdash \varphi \quad \text{(a)}$$
$$\{\varphi \wedge \psi\} \vdash \psi \quad \text{(b)}$$

(I $\vee$) Regla de introducción de la disyunción.
La disyunción con ψ se deduce añadiéndola por la derecha (a) o por la izquierda (b).

$$\{\varphi\} \vdash \varphi \vee \psi \quad \text{(a)}$$
$$\{\varphi\} \vdash \psi \vee \varphi \quad \text{(b)}$$

(E $\vee$) Regla de eliminación de la disyunción.
Método de demostración por casos.

$$\{\varphi \vee \psi, \varphi \rightarrow \chi, \psi \rightarrow \chi\} \vdash \chi$$

(I $\rightarrow$) Regla de introducción de la implicación.
1) Se *supone* la premisa auxiliar φ.
2) Si suponiendo φ se deduce la fórmula ψ, entonces se demuestra la validez de $\varphi \rightarrow \psi$.

$$\{\varphi(supuesto), \psi\} \vdash \varphi \rightarrow \psi$$

(E $\rightarrow$) Regla de eliminación de la implicación.

$$\{\varphi \rightarrow \psi, \varphi\} \vdash \psi$$

(I $\leftrightarrow$) Regla de introducción de doble implicación.

$$\{\varphi \rightarrow \psi, \psi \rightarrow \varphi\} \vdash \varphi \leftrightarrow \psi$$

(E $\rightarrow$) Regla de eliminación de doble implicación.

$$\{\varphi \leftrightarrow \psi\} \vdash \varphi \rightarrow \psi \quad \text{(a)}$$
$$\{\varphi \leftrightarrow \psi\} \vdash \psi \rightarrow \varphi \quad \text{(b)}$$

El uso exclusivo de deducciones presupone razonamientos que dependen de un conjunto (que puede ser vacío) de premisas.

Ejemplo 2. Dado el argumento del Ejemplo 1, una posible demostración en deducción natural es:

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.	$s \wedge \neg r$	$I_\wedge(5,6)$

donde *premisa* se usa para incorporar una fórmula del conjunto de premisas a la demostración. Además, para indicar las fórmulas a las que se aplica una determinada regla de inferencia usamos como referencia la línea que ocupa la fórmula en la demostración. Por ejemplo, la segunda premisa, $p \rightarrow \neg r$, ocupa la línea 3 (paso en el que se incorporó a la demostración), de modo que en la línea 5, los índices $(2, 3, 4)$ de la regla $E_\vee$ corresponden a las fórmulas $p \vee q$, $p \rightarrow \neg r$, y $q \rightarrow \neg r$ respectivamente. En la línea 7 se deduce $s \wedge \neg r$ luego queda demostrada la validez del argumento.

Por su simplicidad, su carácter sintáctico y por ser equivalente a la teoría interpretativa de la lógica de proposiciones, el sistema de deducción natural representa un buen ejemplo de cómo la lógica simbólica se pueda aplicar a la Informática.

2.2. Herramientas similares

En el estudio de la lógica computacional existen varios trabajos mostrando las semejanzas entre el proceso de demostración en deducción natural y la programación [10]. Además, en el contexto de la deducción natural, se han desarrollado herramientas de varias características y complejidad como Taut-Logic[2], Fitch[3], el Asistente para la Deducción Natural (ADN) desarrollado en la Universidad de Alicante [9], la Lógica Simbólica Deductiva (LSD) desarrollado en la Universidad de Girona [7], y otras, como las descritas en [11].

Sin embargo, estas herramientas no resaltan las similitudes entre deducción natural y la programación, y por otra parte, pese a que la mayoría son herramientas de código abierto no es fácil acceder a su código fuente (en el caso de estar operativas, que no todas lo están) para poder adaptarlas. Además, dicha adaptación no sería fácil porque las implementaciones incluyen interfaces de usuario ad hoc y modificaciones en la lógica de los sistemas implicaría adaptar la interfaz.

Basado en el análisis de estas herramientas, hemos implementado DeduccionNatural.pl con los siguientes objetivos: (i) crear un programa sencillo (tiene menos de 300 líneas de código y las demostraciones se pueden escribir con pocas líneas de código), (ii) distribuirla bajo licencia de código abierto, (iii) implementarla en Ciao Prolog, un intérprete (de código abierto) de Prolog, y (iv) que se pudiese ejecutar directamente en cualquier navegador web usando el Playground de Ciao Prolog. El siguiente enlace, `https://tinyurl.com/deduccionnatural23`, da acceso directo a la versión de 2023 en el navegador, sin necesidad de instalación previa.

Por último hemos consultado algunas de las propuestas existentes a la hora de formalizar las demostraciones en deducción natural y ejemplos de juegos lógicos utilizados en la enseñanza de Lógica:

- Pepa Hernández[4] (UPM), propone las reglas de inferencia y el formato basado en sangría para las demostraciones utilizadas en este trabajo.
- José A. Alonso Jiménez[5] (US), tiene multitud de trabajos sobre Lógica, desde apuntes hasta ejercicios introductorios a Prolog.[6]
- José Morales[7] (UPM), usa demostradores de teoremas, por ejemplo, comprueban el problema lógico del *lobo, la oveja y la lechuga* con Z3 [4].

Como resultado, hemos decidido usar la sangría frente a otras propuestas (como las basadas en la notación de Huth/Ryan o la de Fitch, con cajas o líneas) por ser más cercana a la estrategia utilizada en distintos lenguajes de programación (por ejemplo, C y Python). Obsérvese que mientras en C la sangría se usa para facilitar la lectura, en Python se usa para definir bloques de código (ver `https://www.w3schools.com/python/python_syntax.asp`).

3. DeduccionNatural.pl

En esta sección explicamos la implementación y el funcionamiento de DeduccionNatural.pl. Primero, en la Sección 3.1 proponemos una traducción e implementación de los operadores, dado un criterio de precedencia concreto. En la Sección 3.2 mostramos la interfaz y parte de la de las reglas de inferencia. En la Sec-

[2] `https://www.taut-logic.com`

[3] `http://logic.stanford.edu/logica`

[4] `http://www.dia.fi.upm.es/es/phernan`

[5] `https://www.cs.us.es/~jalonso`

[6] El formato para representar los supuestos difiere del formato usado en este trabajo (aunque son equivalentes). La introducción de constantes y variables en lógica de primer orden también es diferente, pero no es relevante para este trabajo.

[7] `https://jfmc.github.io`

ción 3.3 explicamos como usar el programa para comprobar si una demostración es correcta, y finalmente en la Sección 3.4 mostramos cómo definir y usar reglas derivadas para factorizar las demostraciones.

3.1. Sintaxis y precedencias

El alfabeto de la lógica proposicional está compuesto por símbolos de proposición, conectivas y un criterio de precedencia (ver [2,5]). A continuación explicamos la implementación de este alfabeto en Prolog:

- Los símbolos de proposición se representan directamente como átomos (palabras que empiezan por una letra minúscula). Por ejemplo, $p, q, r, \ldots, p1, p2, \ldots$.
- Las conectivas lógicas se traducen en operadores:

Conectiva	$\neg$	$\vee$	$\wedge$	$\rightarrow$	$\leftrightarrow$
Operador	`!`	`or`	`and`	`-->`	`<->`

- El criterio de precedencia de las conectivas lógicas, según [2,5], establece que $\neg$ tiene mayor precedencia que $\wedge$ y $\vee$, que estos tienen mayor precedencia que $\rightarrow$ y $\leftrightarrow$, y que a igual precedencia se asocian por la derecha. La implementación de los operadores considerando este criterio[8] es:

```
1  :- op(200, fy, !).
2  :- op(400, xfy,[and, or]).
3  :- op(600, xfy,[-->, <->]).
```

  donde el primer argumento de `op/3` define la precedencia (a menor índice mayor precedencia), el segundo especifica el tipo de operador y cómo se asocia (la `f` determina su posición, la `x` es un término de menor precedencia y la `y` un término de precedencia menor o igual[9]), y el tercero define el nombre del operador u operadores.

Con esta implementación, de solo tres líneas de código, podemos comprobar si una fórmula está bien formada (suprimiendo paréntesis redundantes). Por ejemplo, al invocar `?- X = s and (p or q).`, el programa devuelve `X = s and p or q` porque como `and` y `or` tienen la misma precedencia y se asocian por la derecha, los paréntesis no son necesarios.

3.2. Reglas de inferencia

El Cuadro 1 muestra la interfaz de las 10 reglas de inferencia en DeduccionNatural.pl. Las reglas de inferencia `'I'`, `'E'` y `'Supuesto'` están entre comillas para que Prolog las trate como átomos (y no como variables). Además para facilitar la legibilidad de las reglas hemos aprovechado la definición infija de las conectivas (sobrecargando `!` como operador binario).

[8] Hay autores que definen criterios de precedencias diferentes. En la Sección 4 explicamos como adaptar esta implementación.

[9] La posición del término `y` indica el criterio de asociación.

Conectiva	Introducción	Eliminación
`and`	`'I' and (_,_)`	`'E' and a(_)` `'E' and b(_)`
`or`	`'I' or a(_,F)` `'I' or b(F,_)`	`'E' or (_,_,_)`
`!`	`'I' ! (_)`	`'E' ! (_)`
`-->`	`'Supuesto'(F)` `'I' --> (_,_)`	`'E' --> (_,_)`
`<->`	`'I' <-> (_,_)`	`'E' <-> a(_)` `'E' <-> b(_)`

Cuadro 1: Interfaz de las reglas de inferencia

El número de argumentos (fórmulas) en cada regla se corresponde con el número de premisas de las reglas de inferencia definidas en la Sección 2.1. Por ejemplo, la introducción de la negación, `'I' ! (_)`, tiene un único argumento, mientras que en la eliminación de la disyunción, `'E' or (_,_,_)`, tiene tres argumentos.

Para las reglas que tienen dos posibles consecuentes a y b, usamos los functores `a/1` y `b/1` respectivamente. Por ejemplo, `'E' and a(_)` representa $\{\varphi \wedge \psi\} \vdash \varphi$, mientras que `'E' and b(_)` representa $\{\varphi \wedge \psi\} \vdash \psi$.

Adicionalmente, las reglas `'I' or a(_,F)`, `'I' or b(F,_)`, y `'Supuesto'(F)` permiten al usuario introducir nuevas fórmulas instanciando la variable `F` con la fórmula deseada. La Figura 2 muestra la implementación de las reglas de inferencia en Prolog y se puede apreciar que gracias a la expresividad del Prolog implementación es una traducción uno a uno de las reglas enumeradas en la Sección 2.1.

3.3. Implementar demostraciones

Para ejecutar DeduccionNatural.pl y poder comprobar las demostraciones, recomendamos utilizar el Playground de Ciao siguiendo el siguiente enlace `https://tinyurl.com/deduccionnatural23`, y posteriormente instanciar e invocar el predicado `main/3`.

Ejemplo 3. La instanciación de `main/3` que representa la demostración descrita en el Ejemplo 2 es:

```
1   main( [ s and p or q,  % Hypotheses
2           p --> ! r,
3           q --> ! r ],
4         s and ! r,       % Deduction
5         [ 'Premisa'(1),  % Proof
6           'E' and b(1),
7           'Premisa'(2),
8           'Premisa'(3),
9           'E' or (2,3,4),
10          'E' and a(1),
11          'I' and (6,5) ]).
```

donde el primer argumento es la lista con las tres premisas, el segundo es el consecuente y el tercero es la

```
% Conjunction
'I' and (A, B) :-
    formula(A, FA),
    formula(B, FB),
    add_formula(FA and FB).
'E' and a(A) :-
    formula(A, FA and _FB),
    add_formula(FA).
'E' and b(A) :-
    formula(A, _FA and FB),
    add_formula(FB).
% Disjunction
'E' or (A, B, C) :-
    formula(A, FB or FC),
    formula(B, FB --> F),
    formula(C, FC --> F),
    add_formula(F).
'I' or a(A, Formula) :-
    formula(A, FA),
    add_formula(FA or Formula).
'I' or b(Formula, B) :-
    formula(B, FB),
    add_formula(Formula or FB).
% Negation
'I' ! (A) :-
    formula(A, FA --> B and ! B),
    add_formula(! FA).
'E' ! (A) :-
    formula(A, ! ! FA),
    add_formula(FA).
```

```
% Implication
'E' --> (A, B) :-
    formula(A, FB --> FC),
    formula(B, FB),
    add_formula(FC).
'I' --> (A, B) :-
    opened(A),
    formula(A, FA),
    valid(B),
    formula(B, FB),
    close_assumption(A),
    add_formula(FA --> FB).

'Supuesto'(FA) :-
    increase_tab,
    add_formula(FA),
    counter(C),
    assert(opened(C)).

% Bi-Implication
'I' <-> (A, B) :-
    formula(A, FA --> FB),
    formula(B, FB --> FA),
    add_formula(FA <-> FB).
'E' <-> a(A) :-
        formula(A, FA <-> FB),
        add_formula(FA --> FB).
'E' <-> b(A) :-
        formula(A, FA <-> FB),
        add_formula(FB --> FA).
```

Figura 2: Extracto del código de DeduccionNatural.pl

lista de reglas a aplicar. Este objetivo se puede invocar directamente en la consola del Playground (panel de la derecha) o a través de un predicado auxiliar. La Figura 1 muestra la implementación de `ejemplo1/0` como parte de DeduccionNatural.pl y en la consola muestra su invocación y la demostración generada (que coincide con la del Ejemplo 2). Nótese tres detalles:

- Para incorporar una premisa a la demostración tenemos que indicar su índice en la lista de premisas. Por ejemplo, `'Premisa'(1)` incorpora la primera premisa, `s and p or q`.
- Los functores `a/1` y `b/1` seleccionan un consecuente concreto, i.e., con `'E' and b(1)` deducimos la fórmula de la derecha, `p or q`.
- Mientras que en el ejemplo 2 aceptamos la línea 7 como válida, en DeduccionNatural.pl hay que pasar los argumentos en orden, i.e., con `'I' and (6,5)` deducimos `s and !r`, pero con `'I' and (5,6)` deduciríamos `!r and s`.

Para facilitar el aprendizaje de DeduccionNatural.pl hemos creado un manual [13] y tres vídeos, disponibles, junto a DeduccionNatural.pl en `https://github.com/Xuaco/DeduccionNatural`.

3.4. Programar reglas derivadas

Como hemos mencionado anteriormente, una de las principales ventajas de DeduccionNatural.pl es la posibilidad de definir reglas derivadas (para factorizar demostraciones) usando el predicado `rule/4`

Ejemplo 4. Quizás la regla derivada más conocida es *Modus Tollens*, $\{\varphi \to \psi, \neg\psi\} \vdash \neg\varphi$ cuya implementación (que incluimos en DeduccionNatural.pl a modo de ejemplo) es la siguiente:

```
rule('MT',                  % Name
     [ FA --> FB,           % Hypotheses
       !FB ],
     !FA,                   % Deduction
     [ 'Premisa'(1),        % Proof
       'Premisa'(2),
       'Supuesto'(FA),
       'E' --> (1,3),
       'I' and (4,2),
       'I' --> (3,5),
       'I' ! (6) ]).
```

donde el primer argumento define el nombre de la regla derivada, `'MT'`, el segundo la lista de premisas, `[FA --> FB, !FB]`, el tercero el consecuente de la

regla derivada, `!FA`, y el cuarto la lista de reglas a aplicar. Nótese que usamos meta-fórmulas, i.e., fórmulas que en lugar de símbolos de proposición tiene variables (palabras que empiezan con mayúsculas) y que durante la evaluación se instanciarán con las fórmulas correspondientes.[10]

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 el número y orden de las fórmulas en la lista de premisas definida en la implementación de la regla.

Ejemplo 5. Dado el siguiente ejemplo (ver páginas 51–52 del Tema 2 en [2]):

$$T[r \to q \wedge s, \neg(q \wedge s)] \vdash \neg r$$

una posible demostración en DeduccionNatural.pl con la regla `'MT'` (*ModusTollens*) del Ejemplo 4, sería:

```
main( [ r --> q and s,      % Hypotheses
        !(q and s) ],
      !r,                   % Deduction
      [ 'Premisa'(1),       % Proof
        'Premisa'(2),
        'MT'(1,2) ]).
```

donde los argumentos que pasamos a `'MT'` coinciden con las premisas definidas en su implementación. La salida que obtendríamos es:

```
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 ... 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
```

donde a continuación de la demostración se comprueba la corrección de las reglas derivadas utilizadas, en este caso la regla `'MT'`.

4. Aplicabilidad

Para validar que DeduccionNatural.pl satisface las expectativas que nos hemos marcado a la hora de facilitar el aprendizaje de los alumnos hemos incorporado el uso de DeduccionNatural.pl en la asignatura de Lógica en dos grados de la Universidad Rey Juan Carlos y hemos realizado una encuesta sobre el grado de satisfacción y la utilidad de *DeduccionNatural.pl*.

4.1. Resultados de la encuesta

La Figura 3 muestra los resultados de la encuesta realizada por los 40 alumnos del grado en Inteligencia Artificial. El número de respuestas a cada pregunta disminuye porque cuando para las preguntas A, B o C contestaban no, podían dejar de rellenar la encuesta.

Pese a que la gran mayoría dice haber utilizado la herramienta para realizar la práctica, pregunta B, no sucede así cuando se les pregunta si la han utilizado como herramienta de aprendizaje, pregunta C. Es importante resaltar que los grupos eran de tres alumnos, por lo tanto, no todos tenían por qué haber utilizado DeduccionNatural.pl al realizar la práctica –los resultados observados en la encuesta coinciden con la percepción que tuvimos durante el desarrollo de las prácticas, pues todos los alumnos asistieron a clase y trabajaron de manera solidaria en todos los ejercicios.

Por otro lado, respecto a la pregunta sobre la facilidad de uso, pregunta D, se aprecia una cierta neutralidad en las valoraciones. Nótese que usamos la escala 1..4 para evitar que los alumnos se decantasen por una calificación intermedia (como pasaría si usásemos la escala Likert, 1..5). De nuestra experiencia en clase, interpretamos que la herramienta ha resultado más fácil de usar a los alumnos con experiencia en programación y no tan fácil a aquellos sin experiencia.

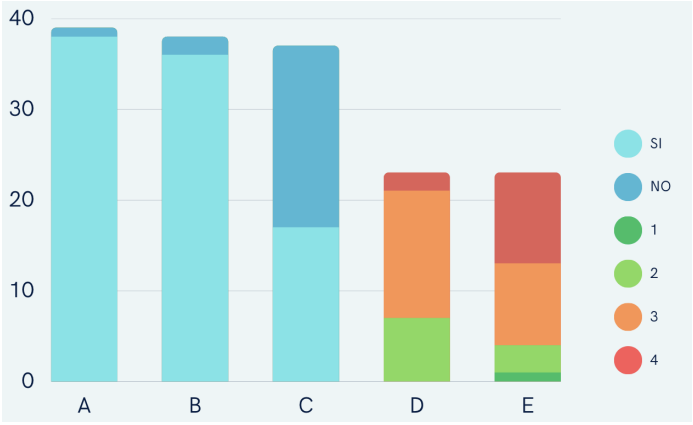
Finalmente, las respuestas a la pregunta E, sobre utilidad para aprender deducción natural, (objetivo principal de esta herramienta), mayoritariamente indican que la herramienta les ha parecido útil. A la vista de estos resultados, concluimos que es preciso mejora el diseño de la interfaz, pero que el objetivo de facilitar el aprendizaje de deducción natural se ha conseguido.

4.2. Sugerencias para la adaptación

En la Sección 2.2 comentamos que uno de los objetivos de DeduccionNatural.pl era la de crear es una herramienta intuitiva y de fácil uso para alumnos y docentes. De hecho, el programa ha despertado la curiosidad de alumnos y profesores de grados relacionados con la Ingeniería en Informática, distintos de los analizados en este artículo.

Adicionalmente, uno de los aspectos positivos de la simplicidad del programa es su facilidad para ser adaptarlo. Por ejemplo, si en lugar del criterio de precedencia visto en la sección 3.1 establecemos que `and` tiene mayor precedencia que `or` y que `-->` tiene mayor precedencia que `<->`, la implementación sería:

[10] Si implementamos otras reglas derivadas y estas no están escritas a continuación de la regla *Modus Tollens*, el compilador de Ciao emitirá un `WARNING` pero el programa funcionará correctamente.



Contestar Si o No:

A: ¿Has utilizado el programa?

B: ¿Has utilizado el programa para comprobar demostraciones?

C: ¿Has utilizado el programa como herramienta de estudio?

Califica de 1 (mal) a 4 (muy bien):

D: Facilidad de uso.

E: Utilidad al estudiar deducción natural.

Figura 3: Resultados de la encuesta de satisfacción.

```
:- op(200, fy, !).
:- op(300, xfy, and).
:- op(400, xfy, or).
:- op(500, xfy, -->).
:- op(600, xfy, <->).
```

Adicionalmente, los docentes que tengan conocimientos de programación lógica podrían modificar la sintaxis de las reglas de inferencia para adaptarlas a otra formalización, añadir más reglas derivadas, etc.

5. Conclusiones y trabajo futuro

En este artículo hemos presentado y descrito una versión preliminar de DeduccionNatural.pl, una herramienta implementada en *Prolog* (lenguaje de programación lógica), para la resolución y corrección de las demostraciones de deducción natural en lógica proposicional. Con *DeduccionNatural.pl* las demostraciones son programas, las reglas de inferencia funciones de una librería predefinida, y las reglas derivadas subrutinas que permiten refactorizar las demostraciones.

Una primera experiencia con alumnos en la URJC nos ha permitido detectar aspectos a mejorar pero los resultados obtenidos nos animan a seguir trabajando para ofrecer nuevas funcionalidades.

La primera acción es mejorar la usabilidad de DeduccionNatural.pl, para ello Marco, alumno de TFG de la URJC, está desarrollando una interfaz gráfica más amigable que esperamos esté lista para el próximo curso. Como trabajo futuro ya tenemos planteado (e iniciado) un TFG para extender el lenguaje a lógica de primer orden, y estaríamos encantados de colaborar en su adaptación a otras notaciones, como por ejemplo, las ya comentadas de Huth/Ryan y de Fitch.

Adicionalmente, contemplamos como trabajo futuro la adaptación de DeduccionNatural.pl para corregir prácticas de manera automática.

Referencias

[1] ANECA. Libro Blanco del Título de Grado en Ingeniería Informática. Proyecto EICE, 2005.

[2] Joaquín Arias. *Lógica: desde Aristóteles hasta Prolog*. Servicio de Publicaciones URJC, 2022. http://hdl.handle.net/10115/20014.

[3] Association for Computing Machinery (ACM) and IEEE. Computing Curricula 2020: Paradigms for Global Computing Education, 2020.

[4] Leonardo De Moura y Nikolaj Bjørner. Z3: An efficient SMT solver. En *TACAS*, pp. 337–340. Springer, 2008.

[5] Alessandra Gallinari. *Apuntes y problemas de la lógica matemática*. Dykinson, Madrid, 2009.

[6] Gerhard Gentzen. Untersuchungen über das logische schließen. I. *Mathematische Zeitschrift*, 35:176–210, 1935.

[7] Josep Humet. LSD, una herramienta didáctica para el aprendizaje de la lógica. *JENUI 2001*, pp. 482–485, 2001.

[8] Jose Emilio Labra Gayo. ¿Hay Lógica en la situación actual de las titulaciones informáticas? En *JENUI 2004*, pp. 227–234, 2004.

[9] Faraón Llorens Largo y Sergio Mira Cabrera. Herramienta para la enseñanza de la Deducción Natural. En *JENUI 2000*, pp. 496–502, 2000.

[10] Faraón Llorens Largo, Rosana Satorre Cuerda, Francisco Escolano, y Pilar Arques Corrales. Deducción natural versus computación. *JENUI 1999*, pp. 259–265, 1999.

[11] Petr Manas. CLPractice 2.0. Tools for Learning: Computation and Logic. *University of Edinburgh*, 2021. https://shorturl.at/boW47.

[12] OECD. Resultados de PISA 2018 para España. https://www.oecd.org/pisa, 2019.

[13] Iván Ramírez y Joaquín Arias. *DeduccionNatural.pl, Manual*. BURJC-Digital, 2022. http://hdl.handle.net/10115/20168.