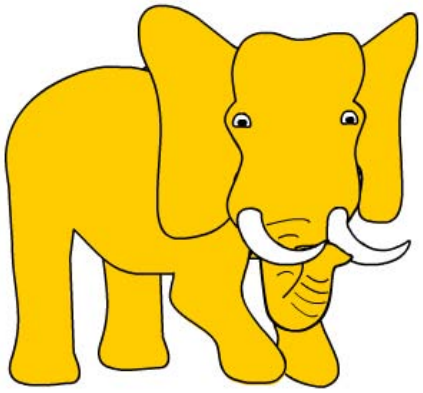
Trabajo Final Lóg. Inf. y la I.A.
febrero, 2007

easy logic



Rubén de la Peña Ramos

Departamento de Informática y Automática
Universidad de Salamanca

Tabla de Contenidos

Lista de Imágenes

1. Introducción

Easy Logic se define según sus autores como:

"Interfaz web interactiva para demostradores de teoremas"

Ciertamente es así, pues la aplicación para unas premisas y una conclusión dadas, presentará en pantalla:

a) Bien los pasos seguidos para probar la conclusión a partir de las premisas (basándose en el procedimiento de cálculo).

b) O bien un contraejemplo que demuestre que la prueba no es posible.

Actualmente dispone de los siguientes procedimientos de cálculo:

- Tablas de verdad.
- Normalización
- Tableaux Semánticos
- Deducción Natural
- Cálculo de Secuentes

El servidor donde está instalada la aplicación se encuentra de manera provisional en http://easylogic.yi.org

Una vez tecleemos esa dirección en nuestro navegador habitual nos encontramos con lo siguiente:

Figura 01: Interfaz Principal

Podemos ver que desde la página principal se nos presentan las cuatro funcionalidades de la aplicación a las que podremos acceder haciendo clic en cada una de las pestañas.

Aparece también información de contacto de los creadores para el envío de dudas, sugerencias…

A continuación se detallarán cada una de las opciones de la aplicación.

2. Syntax

El motivo de empezar por esta pestaña es que nos indicará en que manera hemos de introducir nuestros datos para que sean procesados por la aplicación.

Representación de **Conectivas Lógicas** (de menor a mayor orden de preferencia):

- Negación: `-` (guión)
  Nota: La Doble Negación se representa como `-(-a)`, (no `--a !!`)
- Conjunción: `&` (ampersand)
- Disyunción: `v` (letra "v")
- Condicional: `=>`. (igualdad y simbolo de mayor)
- Bicondicional: `<=>` (símbolo de menor, igualdad y símbolo mayor)

El orden de preferencia establecido permite evitar el uso de paréntesis sin caer en la ambigüedad. Ejemplo: `p v q & r`

Se interpreta como: `p v (q & r)`

(Dado que la disyunción tiene mayor preferencia que la disjunción).

Conjunción y disyunción son asociativas por la izquierda, es decir: `a & b & c`

Se interpreta como: `(a & b) & c`

Condicional y bicondicional son asociativos por la derecha, es decir: `a => b => c`

Se interpreta como: `a => (b => c)`

Para representar **variables proposicionales**, se usan las letras (o combinaciones de latras) minúsculas (salvo "v" que está reservada para la disyunción).

Cuando se quiera introducir más de una premisa se separarán por comas, pudiendo ocupar cada una más de una línea.

Ejemplo:

Supongamos que queremos introducir como premisas: a → b, y a

Y como conclusión: b

Introduciríamos los datos de la siguiente manera:

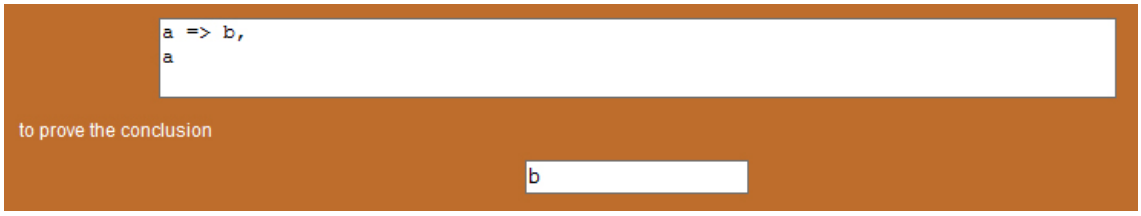


Figura 02: Syntax

(Evidentemente se trata de la regla de inferencia de Modus Ponens, por lo que como veremos más adelante la aplicación nos dirá que se trata de un razonamiento correcto)

3. Prove!

Esta pestaña aglutina la funcionalidad central de la aplicación, esto es, la prueba de teoremas mediante distintos procedimientos de cálculo.

Al seleccionar esta pestaña nos encontraremos con lo siguiente:

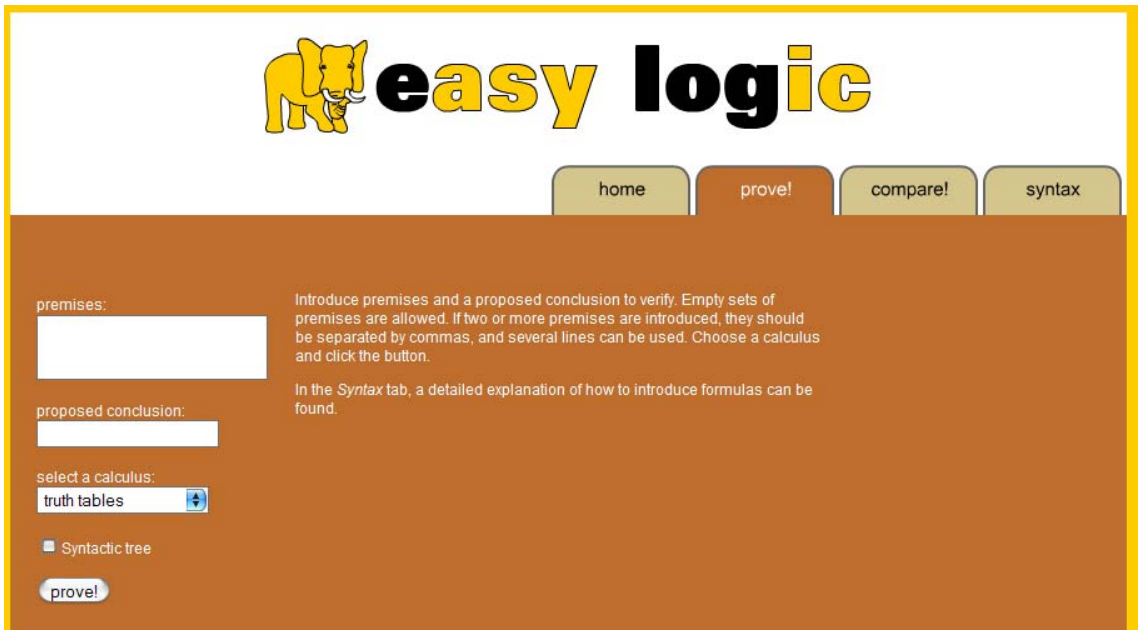


Figura 03: Prove

En los campos "**premises**" y "**proposed conclusión**" el usuario introducirá los datos de manera análoga a la vista en el apartado anterior.

A través del desplegable "**select a calculus**" se seleccionará el procedimiento de cálculo a utilizar en la prueba del teorema.

Por último, si el usuario activa la pestaña "**Syntactic tree**" la aplicación mostrará tras la ejecución, el árbol sintáctico de la implicación que tiene como antecedente la

conjunción de las premisas, y como consecuente la conclusión y que habitualmente da pistas sobre cómo se han usado las reglas de los cálculos.

A continuación se probará cada uno de los procedimientos de cálculo, partiendo para ello, de dos conjuntos de premisas, uno que permitirá probar un teorema y otro que no, de manera que se pueda observar la respuesta de la aplicación para cada caso.

Así pues los datos que se introducirán en la aplicación serán:

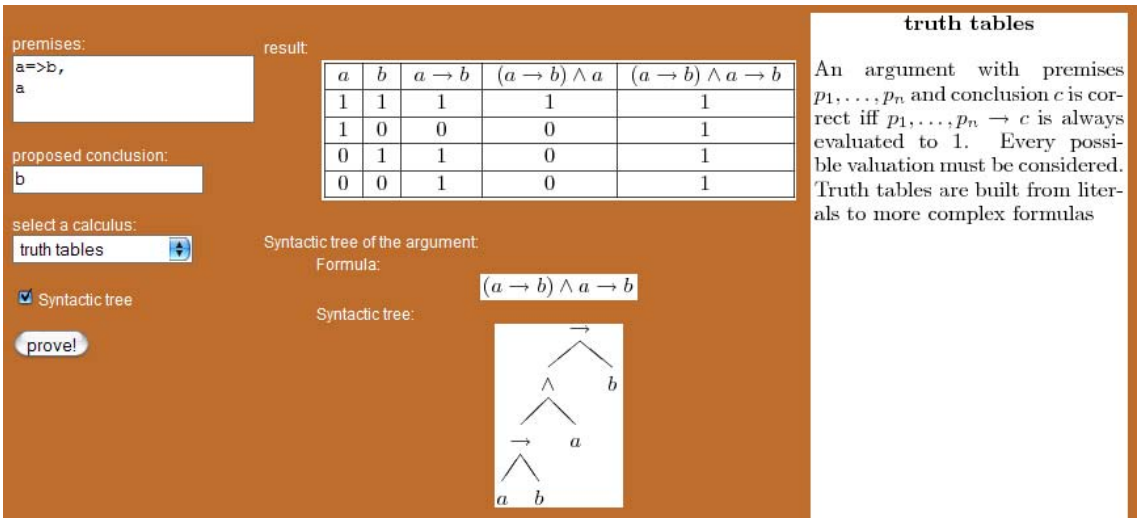
- { a → b, a } ⊢ b (nos referiremos a este conjunto de prueba como **caso A**)
- { a → b, ¬a } ⊢ b (nos referiremos a este conjunto de prueba como **caso B**)

3.1. Tablas de Verdad

"Un razonamiento con premisas p_1 ... p_n y conclusión c es correcto si p_1 ... $p_n \rightarrow c$ es siempre evaluado a 1 (Se debe considerar cada posible valoración)."

En el apartado 8 (Apéndice) pueden consultarse las tablas de verdad básicas.

CASO A



a	b	$a \rightarrow b$	$(a \rightarrow b) \wedge a$	$(a \rightarrow b) \wedge a \rightarrow b$
1	1	1	1	1
1	0	0	0	1
0	1	1	0	1
0	0	1	0	1

Figura 04: Tablas de Verdad A

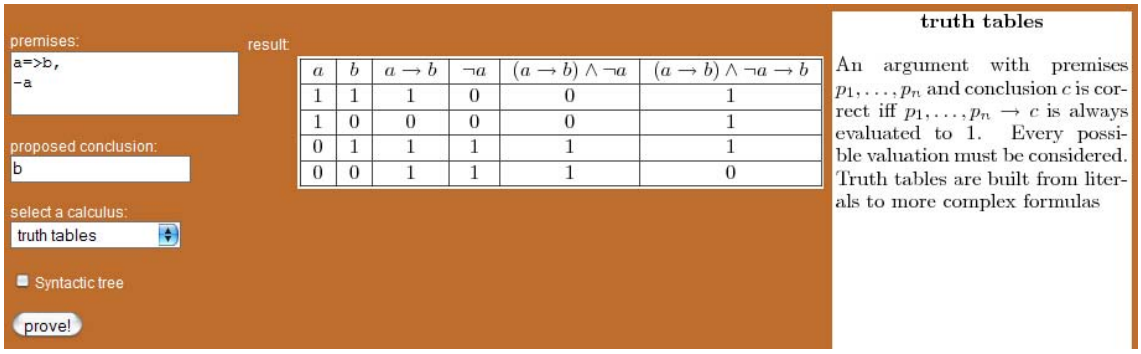
Tras la ejecución se observa una característica interesante de la aplicación, y es que en la parte derecha muestra un pequeño resumen del procedimiento de cálculo utilizado, lo cual puede servir muy bien como repaso.

En este caso y por cuestiones de espacio se ha recortado dicha explicación, no obstante para consultarla de manera completa se remite al lector al apartado 8 (Apéndice).

Respecto al ejemplo en sí, se observa que todas las valoraciones son 1, por lo que efectivamente se puede probar "b" a partir de { a → b, a }, es decir el razonamiento es correcto. (¿a alguien le suena Modus Ponens? ☺).

Por ser éste el primer ejemplo se ha optado por mostrar también el árbol sintáctico comentado en el apartado anterior.

CASOB



a	b	$a \rightarrow b$	$\neg a$	$(a \rightarrow b) \wedge \neg a$	$(a \rightarrow b) \wedge \neg a \rightarrow b$
1	1	1	0	0	1
1	0	0	0	0	1
0	1	1	1	1	1
0	0	1	1	1	0

Figura 05: Tablas de Verdad B

Vemos que efectivamente el caso B no es un razonamiento correcto debido a la valoración de verdad de la entrada 4 de la tabla: { ¬a , ¬b }.

3.2. Normalización

"Un razonamiento con premisas p_1 ... p_n y conclusión c es correcto si la forma normal conjuntiva (CNF) de $\vdash (p_1 \wedge \ldots \wedge p_n) \rightarrow c$ tiene literales complementarios en cada disyunción."

En el apartado 8 (Apéndice) pueden consultarse reglas para pasar a CNF una fórmula dada.

CASO A

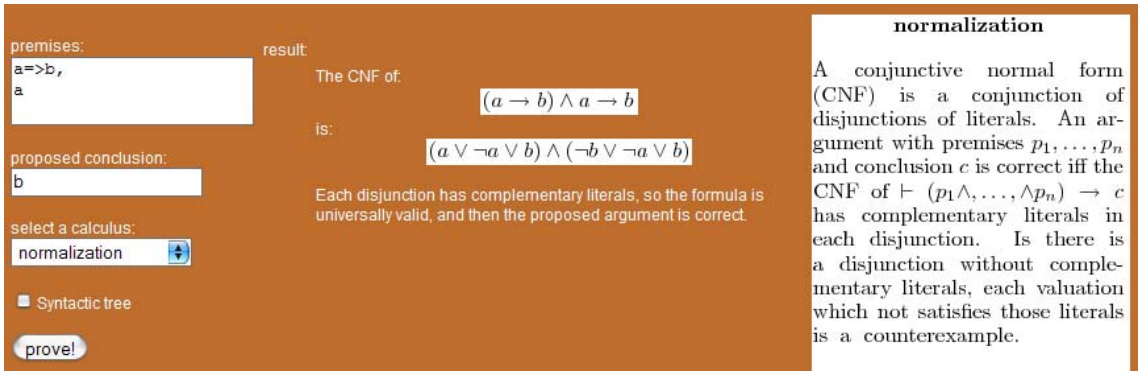


Figura 06: Normalización A

Se observa que cada disyunción tiene literales complementarios, por lo que el razonamiento es correcto.

CASOB

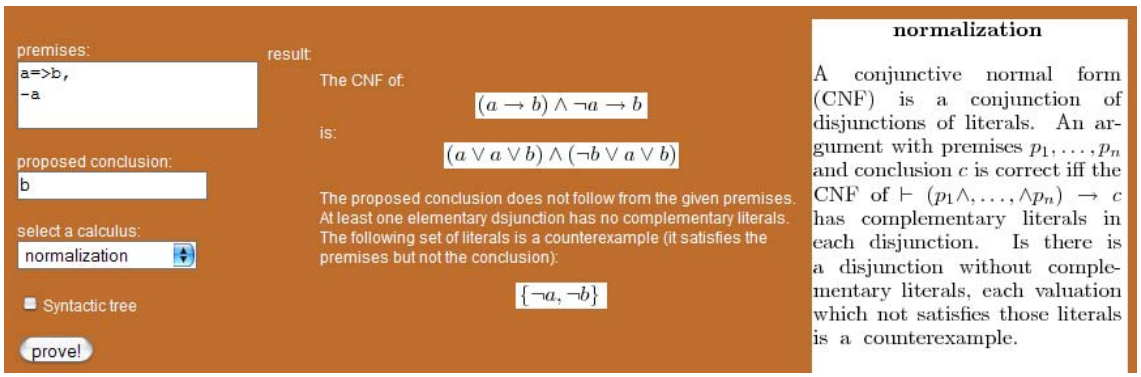


Figura 07: Normalización B

Vemos que la primera disyunción no tiene literales complementarios, por lo que el razonamiento no es correcto, y de hecho cada valoración que no coincida con los literales de dicha disyunción constituiría un contraejemplo. En este caso solo hay una posibilidad: { ¬a , ¬b }

3.3. Tableaux Semánticos

"Un razonamiento con premisas p_1 ... p_n y conclusión c es correcto si el tableaux de { p_1, ... ,p_n, ¬c } es cerrado, es decir, cada rama acaba en ⊗."

En el apartado 8 (Apéndice) pueden consultarse reglas para la construcción de tableaux semánticos.

CASO A

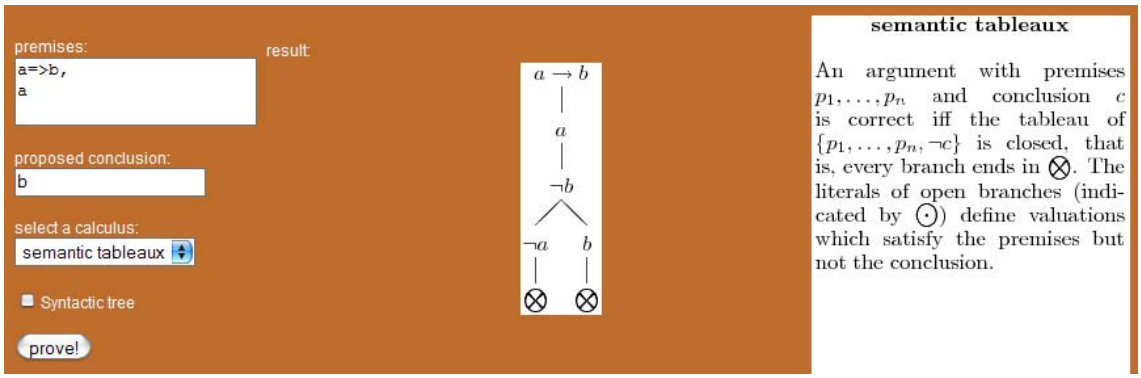


Figura 08: Tableaux A

Se observa que cada rama acaba termina en ⊗, por lo que el tableaux es cerrado y en consecuencia el razonamiento es correcto.

CASOB

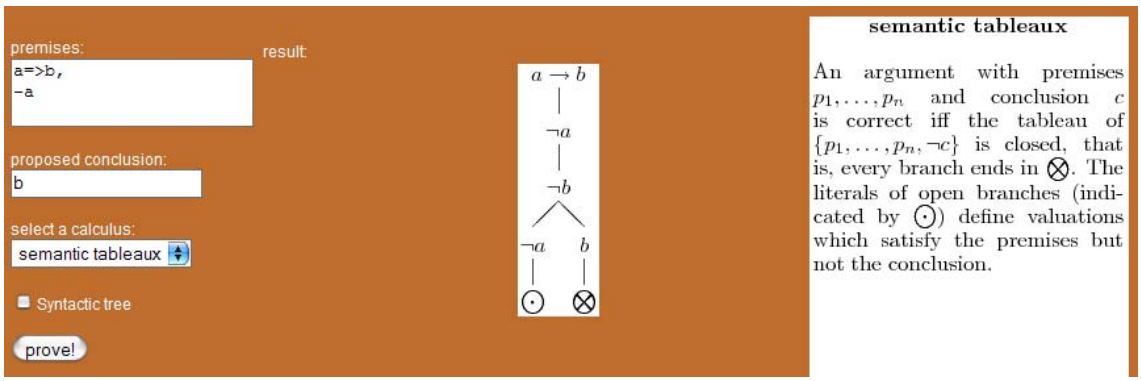


Figura 09: Tableaux B

En este caso vemos que la primera rama del tableaux acaba en ⊙, por lo que el tableaux no es cerrado y en consecuencia el razonamiento no es correcto.

Se pueden obtener las valoraciones de los literales que constituirían un contraejemplo recorriendo la rama abierta del tableaux. En este caso: { ¬a , ¬b }

3.4. Deducción Natural

"Un razonamiento con premisas p_1 ... p_n y conclusión c es correcto si partiendo del conjunto de premisas p_1 ... p_n podemos llegar a c mediante la aplicación de una serie de reglas."

En el apartado 8 (Apéndice) pueden consultarse dichas reglas.

La aplicación hace uso de la notación Fitch para la representación de esquemas de deducción natural.

CASO A

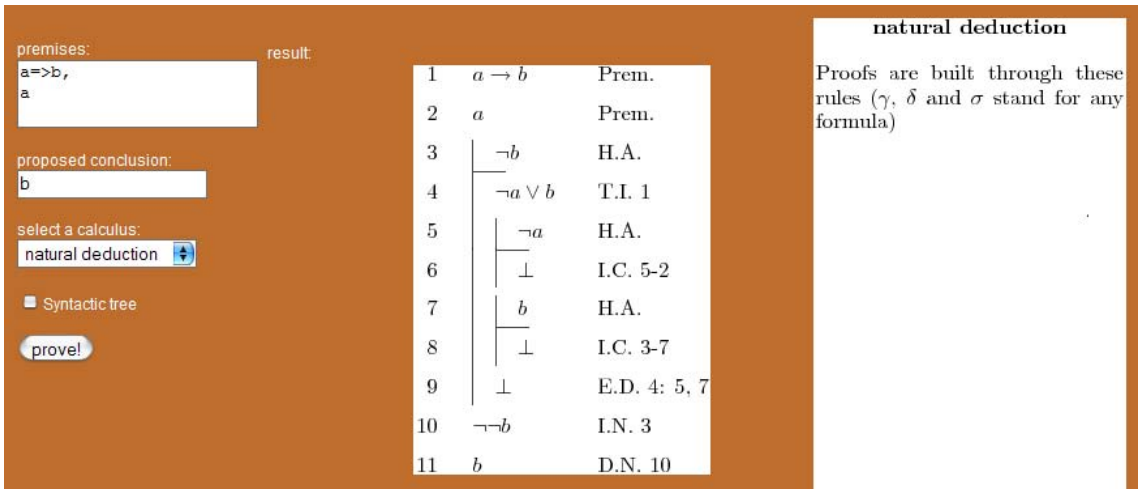


Figura 10: Deducción A

Se observa que se llega correctamente a la conclusión por lo que el razonamiento es correcto.

CASOB

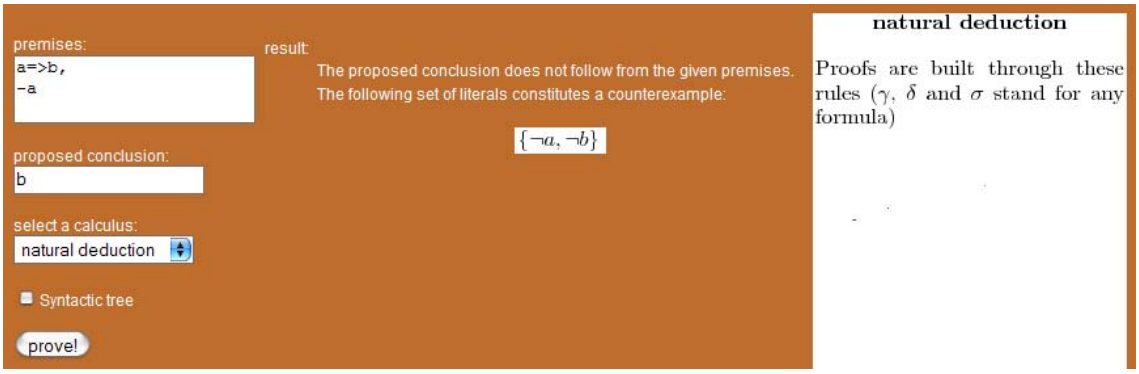


Figura 11: Deducción B

En este caso no se puede obtener la conclusión a partir de las premisas, por lo que el razonamiento no es correcto.

La valoración { ¬a , ¬b } hace ciertas las premisas y falsa la conclusión, por lo que constituiría un contraejemplo.

3.5. Cálculo Secuentes

"(O cálculo de consecuencias lógicas) Un razonamiento con premisas $p_1 \ldots p_n$ *y conclusión c es correcto si se puede demostrar el secuente* $\vdash (p_1 \wedge \ldots \wedge p_n) \rightarrow c$ *, lo que implica que el árbol de prueba esté completo (todas las ramas acaben con un axioma)."*

En el apartado 8 (Apéndice) pueden consultarse las reglas para el cálculo de secuentes.

CASO A

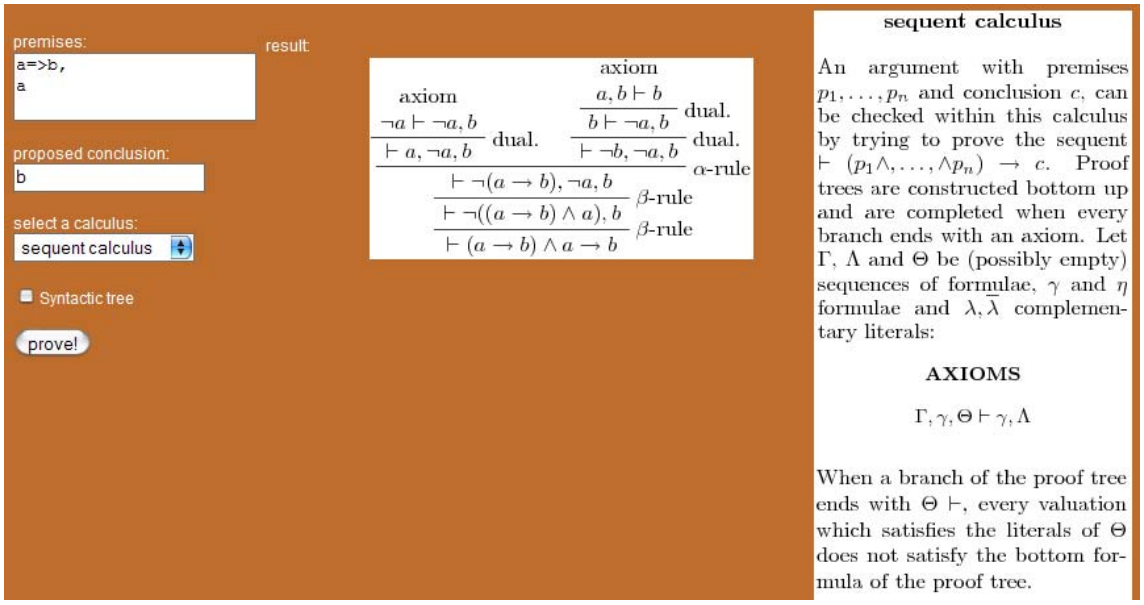


Figura 12: Secuentes A

Se observa que cada rama del árbol finaliza con un axioma, por lo que el razonamiento es correcto.

CASOB

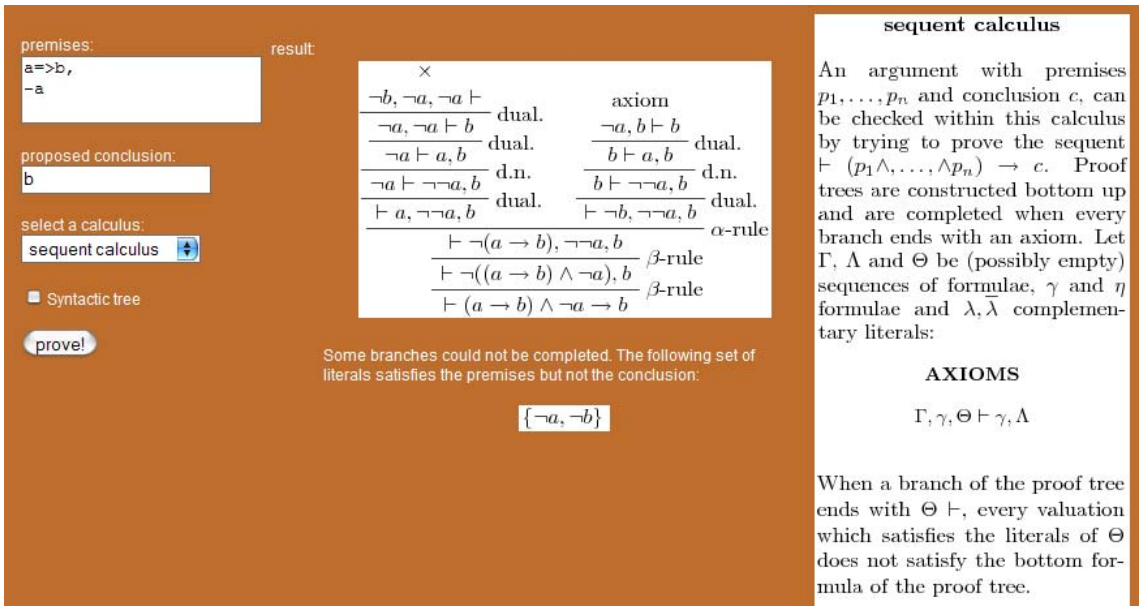


Figura 13: Secuentes B

Vemos que la primera rama acaba con una estructura tipo $\Theta \vdash$, por lo que toda valoración que satisfaga los literales de Θ constituirá un contraejemplo.

En este caso solo hay una posibilidad: { ¬a , ¬b }.

4. Compare!

Esta pestaña, la cual muestra aspecto de la figura 14, nos permitirá comparar los dos procedimientos de cálculo que queramos para los datos que introduzcamos.

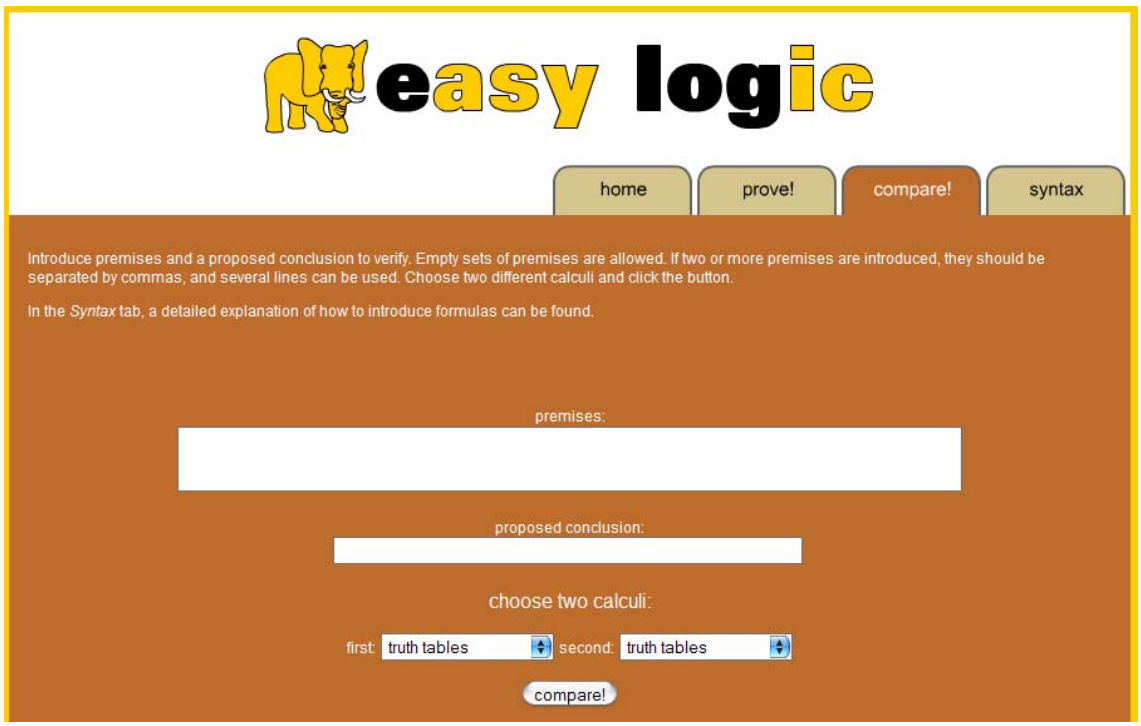
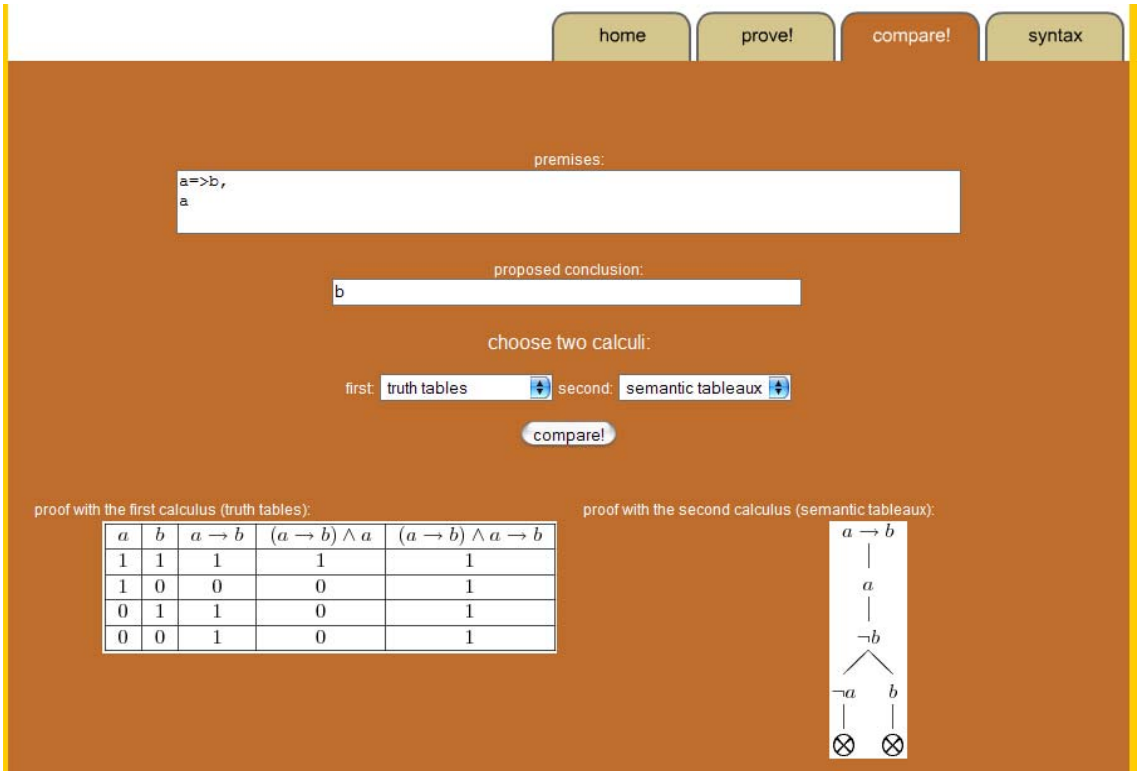


Figura 14: Compare – Vista Principal

Para ello tras introducir las premisas y la conclusión tal y como se vio en el apartado de sintaxis, solo tenemos que escoger los dos cálculos a comparar de los desplegables de la parte inferior de la pantalla y posteriormente hacer clic en *compare!*.

En este caso y para ilustrar el funcionamiento de *compare!* se han elegido los conjuntos de datos de los casos A y B del apartado anterior y como procedimientos de cálculo, tablas de verdad y tableaux semánticos.

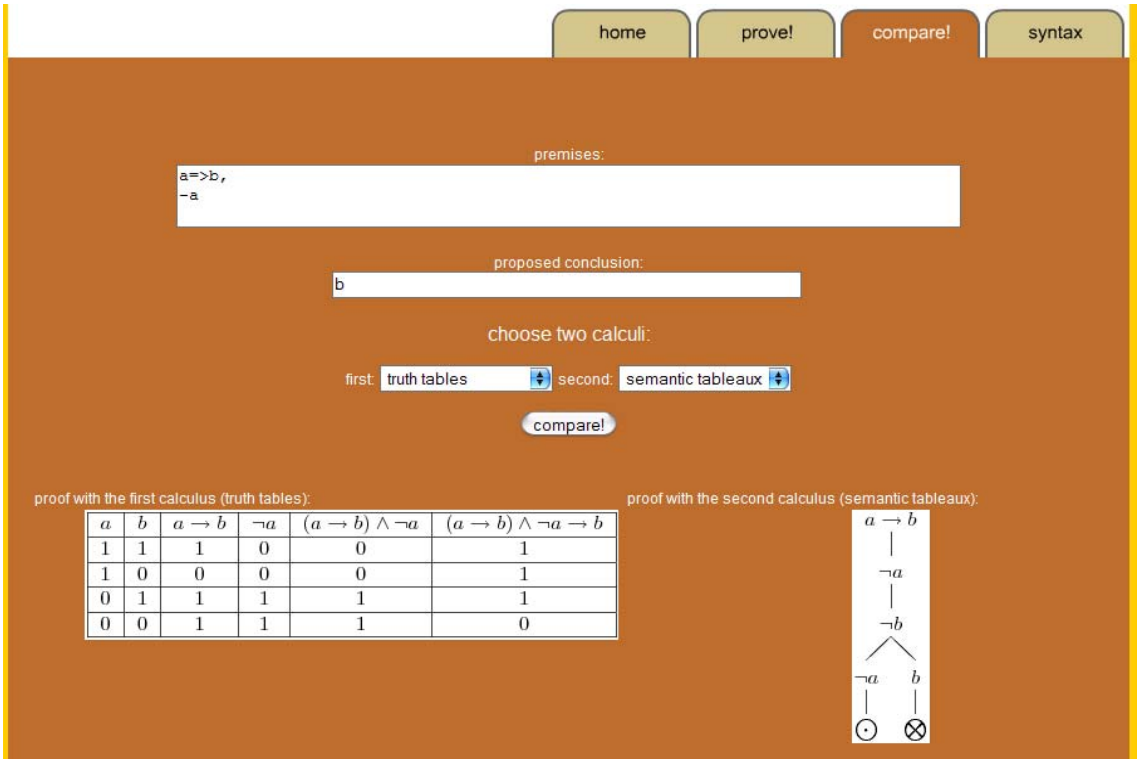
CASO A



a	b	$a \to b$	$(a \to b) \wedge a$	$(a \to b) \wedge a \to b$
1	1	1	1	1
1	0	0	0	1
0	1	1	0	1
0	0	1	0	1

Figura 15: Compare – A

CASOB



a	b	$a \to b$	$\neg a$	$(a \to b) \wedge \neg a$	$(a \to b) \wedge \neg a \to b$
1	1	1	0	0	1
1	0	0	0	0	1
0	1	1	1	1	1
0	0	1	1	1	0

Figura 16: Compare – B

La principal ventaja de este modo de operación, es que nos presenta un contraste directo de dos métodos, por lo que para un caso concreto podremos escoger el que nos parezca mas adecuado.

Por contra no ofrece explicación alguna de los procedimientos de cálculo, característica que sin ninguna duda los usuarios poco experimentados echaran en falta.

5. Detalles de Implementación

A continuación se muestra la estructura de Easy Logic:

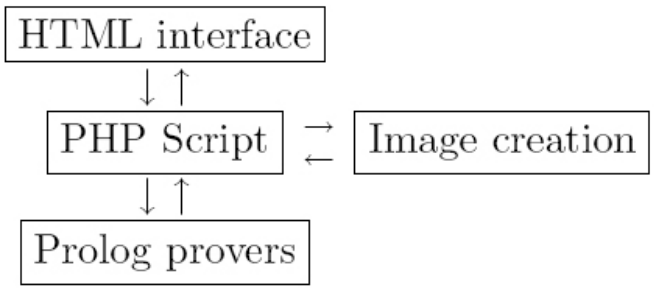


Figura 17: Estructura Easy Logic

¿Cómo funciona Easy Logic? El procedimiento es el siguiente:

El usuario interacciona con el demostrador a través de la interfaz web, para ello se tiene un script en PHP que procesa la entrada y se la pasa a los demostradores.

Los demostradores están escritos en Prolog, fundamentalmente debido a su caracter declarativo, lo cual otorga una manera natural de codificar las operaciones relativas a los procedimientos. En concreto se ha elegido SWI-Prolog por seguir el éstandar de Edimburgo y por su licencia LGPL.

La salida de los scripts de Prolog será la información devuelta al usuario, junto con una representación visual (por ejemplo una tabla de verdad, un tableaux semántico, etc) generada mediante compilación con LaTeX y que dará una salida más amigable a los resultados.

Los componentes usados por Easy Logic (SWI-Prolog, LaTeX, y otros para la generación de las imágenes) no se encuentran normalmente disponibles en un servidor, por lo que se tuvo que preparar uno en concreto para la explotación de la aplicación.

Una vez se encuentren todos los componentes necesarios configurados correctamente en el servidor, la instalación de la aplicación no puede ser más sencilla, pues basta con “arrastrar” al mismo los ficheros de Easy Logic y editar algunas líneas en el código para indicar las rutas de los programas usados.

6. Conclusiones

Easy Logic es una aplicación sencilla e intuitiva, ideal como herramienta de aprendizaje para el usuario que comience el estudio de la prueba de teoremas y sus diferentes procedimientos de cálculo, en parte gracias a los breves resúmenes que muestra la aplicación en el modo *prove!*.

No obstante, para usuarios más avanzados, puede ser un excelente complemento que les libre de tediosas comprobaciones manuales.

Es por todo esto que Easy Logic es una herramienta adecuada tanto para asignaturas iniciales de lógica (como herramienta de aprendizaje) como para niveles más avanzados (como herramienta de soporte).

Una limitación que se le podría achacar a Easy Logic es que solo ofrece soporte para lógica proposional, algo que no es para nada definitivo, pues debido a la modularidad que presenta la aplicación, en cualquier momento se podrían implementar módulos adicionales que dieran soporte a lógicas diferentes, y que se integrarían perfectamente en la aplicación.

De hecho los creadores de Easy Logic instan a esto mismo, que los usuarios se animen a implementar diferentes tipos de demostradores para hacer de Easy Logic la herramienta de demostración de teoremas más completa.

7. Referencias

- http://easylogic.yi.org/

  ¡Fundamental!

- http://www.departamento.us.es/dpfilogi/

  Departamento de Filosofía y Lógica y Filosofía de la Ciencia de la Universidad de Sevilla.

- http://www.danielclemente.com/logica/dn.html

  Introducción a la deducción natural.

- http://en.wikipedia.org/wiki/Sequent_calculus

  Cálculo de secuentes (en inglés).

8. Apéndice

En este apartado se mostrarán los resúmenes completos proporcionados por la aplicación para cada tipo de cálculo.

- *Tablas de Verdad*

truth tables

An argument with premises $p_1, \ldots, p_n$ and conclusion c is correct iff $p_1, \ldots, p_n \rightarrow c$ is always evaluated to 1. Every possible valuation must be considered. Truth tables are built from literals to more complex formulas by following:

negation:

α	$\neg\alpha$
1	0
0	1

conjunction:

α	β	$\alpha \wedge \beta$
1	1	1
1	0	0
0	1	0
0	0	0

disjunction:

α	β	$\alpha \vee \beta$
1	1	1
1	0	1
0	1	1
0	0	0

conditional:

α	β	$\alpha \rightarrow \beta$
1	1	1
1	0	0
0	1	1
0	0	1

double conditional:

α	β	$\alpha \rightarrow \beta$
1	1	1
1	0	0
0	1	0
0	0	1

Figura 18: Apéndice – Tablas de Verdad

- *Normalización*

normalization

A conjunctive normal form (CNF) is a conjunction of disjunctions of literals. An argument with premises $p_1, \ldots, p_n$ and conclusion c is correct iff the CNF of $\vdash (p_1 \wedge, \ldots, \wedge p_n) \rightarrow c$ has complementary literals in each disjunction. Is there is a disjunction without complementary literals, each valuation which not satisfies those literals is a counterexample. A given formula can be transformed to an equivalent CNF by using the equivalences (γ, δ and σ stand for any formula):

$$\neg\neg\gamma \Leftrightarrow \gamma$$
$$\neg(\gamma \vee \delta) \Leftrightarrow \neg\gamma \wedge \neg\delta$$
$$\neg(\gamma \wedge \delta) \Leftrightarrow \neg\gamma \vee \neg\delta$$
$$\neg(\gamma \rightarrow \delta) \Leftrightarrow \gamma \wedge \neg\delta$$
$$\neg(\gamma \leftrightarrow \delta) \Leftrightarrow (\gamma \wedge \neg\delta) \vee (\neg\gamma \wedge \delta)$$
$$\gamma \rightarrow \delta \Leftrightarrow \neg\gamma \vee \delta$$
$$\gamma \leftrightarrow \delta \Leftrightarrow (\gamma \wedge \delta) \vee (\neg\gamma \wedge \neg\delta)$$
$$\gamma \vee \delta \Leftrightarrow \delta \vee \gamma$$
$$\gamma \wedge \delta \Leftrightarrow \delta \wedge \gamma$$
$$\sigma \vee (\gamma \wedge \delta) \Leftrightarrow (\sigma \vee \gamma) \wedge (\sigma \vee \delta)$$

Figura 19: Apéndice – Normalización

- *Tableaux Semánticos*

semantic tableaux

An argument with premises $p_1, \ldots, p_n$ and conclusion c is correct iff the tableau of $\{p_1, \ldots, p_n, \neg c\}$ is closed, that is, every branch ends in $\otimes$. The literals of open branches (indicated by $\odot$) define valuations which satisfy the premises but not the conclusion. The rules for semantic tableaux are (γ and η stand for any formula):

double negation:

$$\frac{\neg(\neg\gamma)}{\gamma}$$

closing:

$$\frac{\begin{array}{c}\gamma\\ \neg\gamma\end{array}}{\otimes}$$

β-rules:

$$\frac{\gamma \vee \eta}{\gamma \mid \eta}$$

$$\frac{\gamma \rightarrow \eta}{\neg\gamma \mid \eta}$$

$$\frac{\neg(\gamma \wedge \eta)}{\neg\gamma \mid \neg\eta}$$

$$\frac{\gamma \leftrightarrow \eta}{\gamma \wedge \eta \mid \neg\gamma \wedge \neg\eta}$$

$$\frac{\neg(\gamma \leftrightarrow \eta)}{\gamma \wedge \neg\eta \mid \neg\gamma \wedge \eta}$$

α-rules:

$$\frac{\gamma \wedge \eta}{\begin{array}{c}\gamma\\ \eta\end{array}}$$

$$\frac{\neg(\gamma \vee \eta)}{\begin{array}{c}\neg\gamma\\ \neg\eta\end{array}}$$

$$\frac{\neg(\gamma \rightarrow \eta)}{\begin{array}{c}\gamma\\ \neg\eta\end{array}}$$

Figura 20: Apéndice – Tableaux Semánticos

- *Deducción Natural*

Nota: T ≡ transformación, I ≡ introducción, E ≡ eliminación,

H.A ≡ hipótesis auxiliar, I ≡ →, D.I. ≡ ↔, D ≡ ∨, C ≡ ∧, N ≡ ¬

natural deduction

Proofs are built through these rules (γ, δ and σ stand for any formula):

$$\frac{\gamma \to \delta}{\neg\gamma \vee \delta}\ \text{T.I.} \qquad \frac{\neg(\gamma \to \delta)}{\gamma \wedge \neg\delta}\ \text{N.I.}$$

$$\frac{\neg(\gamma \vee \delta)}{\neg\gamma \wedge \neg\delta}\ \text{N.D.} \qquad \frac{\neg\neg\gamma}{\gamma}\ \text{D.N.}$$

$$\frac{\begin{array}{c}\gamma\\ \neg\gamma\end{array}}{\bot}\ \text{I.C.} \qquad \frac{\neg(\gamma \wedge \delta)}{\neg\gamma \vee \neg\delta}\ \text{N.C.}$$

$$\frac{\gamma \wedge \delta}{\begin{array}{c}\gamma\\ \delta\end{array}}\ \text{E.C.}$$

$$\frac{\neg(\gamma \leftrightarrow \delta)}{(\gamma \wedge \neg\delta) \vee (\neg\gamma \wedge \delta)}\ \text{N.D.I.}$$

$$\frac{\gamma \leftrightarrow \delta}{(\gamma \wedge \delta) \vee (\neg\gamma \wedge \neg\delta)}\ \text{T.D.I.}$$

$$\frac{\begin{array}{l}\gamma \vee \eta\\ \vdots\\ \left|\begin{array}{ll}\gamma & \text{H.A.}\\ \vdots & \\ \sigma & \end{array}\right.\\ \left|\begin{array}{ll}\eta & \text{H.A.}\\ \vdots & \\ \sigma & \end{array}\right.\\ \vdots\end{array}}{\sigma}\ \text{E.D.}$$

$$\frac{\begin{array}{l}\left|\begin{array}{ll}\eta & \text{H.A.}\\ \vdots & \\ \bot & \end{array}\right.\\ \vdots\end{array}}{\neg\eta}\ \text{I.N.}$$

Figura 21: Apéndice – Deducción Natural

- *Cálculo de Secuentes*

sequent calculus

An argument with premises $p_1, \ldots, p_n$ and conclusion c, can be checked within this calculus by trying to prove the sequent $\vdash (p_1 \wedge, \ldots, \wedge p_n) \rightarrow c$. Proof trees are constructed bottom up and are completed when every branch ends with an axiom. Let Γ, Λ and Θ be (possibly empty) sequences of formulae, γ and η formulae and $\lambda, \overline{\lambda}$ complementary literals:

AXIOMS

$$\Gamma, \gamma, \Theta \vdash \gamma, \Lambda$$

RULES

double negation (d.n.):

$$\frac{\Gamma \vdash \gamma, \Lambda}{\Gamma \vdash \neg(\neg\gamma), \Lambda}$$

duality (dual.):

$$\frac{\lambda, \Gamma \vdash \Lambda}{\Gamma \vdash \overline{\lambda}, \Lambda}$$

β-rules:

$$\frac{\Gamma \vdash \gamma, \eta, \Lambda}{\Gamma \vdash \gamma \vee \eta, \Lambda}$$

$$\frac{\Gamma \vdash \neg\gamma, \eta, \Lambda}{\Gamma \vdash \gamma \rightarrow \eta, \Lambda}$$

$$\frac{\Gamma \vdash \neg\gamma, \neg\eta, \Lambda}{\Gamma \vdash \neg(\gamma \wedge \eta), \Lambda}$$

$$\frac{\Gamma \vdash \gamma \wedge \eta, \neg\gamma \wedge \neg\eta, \Lambda}{\Gamma \vdash \gamma \leftrightarrow \eta, \Lambda}$$

$$\frac{\Gamma \vdash \gamma \wedge \neg\eta, \neg\gamma \wedge \eta, \Lambda}{\Gamma \vdash \neg(\gamma \leftrightarrow \eta), \Lambda}$$

α-rules:

$$\frac{\Gamma \vdash \gamma, \Lambda \qquad \Gamma \vdash \eta, \Lambda}{\Gamma \vdash \gamma \wedge \eta, \Lambda}$$

$$\frac{\Gamma \vdash \neg\gamma, \Lambda \qquad \Gamma \vdash \neg\eta, \Lambda}{\Gamma \vdash \neg(\gamma \vee \eta), \Lambda}$$

$$\frac{\Gamma \vdash \gamma, \Lambda \qquad \Gamma \vdash \neg\eta, \Lambda}{\Gamma \vdash \neg(\gamma \rightarrow \eta), \Lambda}$$

When a branch of the proof tree ends with $\Theta \vdash$, every valuation which satisfies the literals of Θ does not satisfy the bottom formula of the proof tree.

Figura 22: Apéndice – Cálculo de Secuentes