

Universidad Rey Ju a11 C ar lo s **1 Slf",ri.eie dill i lfH:::li**

**u**

ISBN: 978-84-09-52634-5



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

**Apuntes de Lógica**

DESDE ARISTÓTELES HASTA PROLOG

### Autor: Joaquín Arias

Copyright c 2023 Joaquín Arias .Este obra está bajo la licencia CC BY-SA 4.0, [Creative Commons Atribuciónn-CompartirIgual 4.0 Internacional](https://creativecommons.org/licenses/by-sa/4.0/deed.es). Como citar es- ta obra: Arias, Joaquín (2023). Apuntes de Lógica: desde Aristóteles hasta Prolog. Madrid, Servicio Publicaciones de la URJC. ISBN: 978-84-09-52634-5



*Q*

# Índice general

#### Índice general [i](#_bookmark0)

#### Introducción [1](#_bookmark1)

* 1. Por qué es necesaria, tipos e historia. [2](#_bookmark2)
     1. Introducción [2](#_bookmark3)
     2. Lógica [3](#_bookmark4)
     3. Tipos de lógicas [4](#_bookmark5)
     4. Algo de historia [4](#_bookmark6)
  2. Teoría de conjuntos, relaciones, funciones y álgebra de Boole. [6](#_bookmark7)
     1. Teoría de Conjuntos [6](#_bookmark8)
     2. Relaciones Binarias [10](#_bookmark9)
     3. Relaciones n-arias [12](#_bookmark10)
     4. Funciones n-arias [12](#_bookmark11)
     5. Algebra de Boole [12](#_bookmark12)

#### Lógica Proposicional [15](#_bookmark13)

* 1. Sintaxis, semántica y teoría interpretativa. [16](#_bookmark14)
     1. Lenguaje proposicional [16](#_bookmark15)
     2. Semántica de la lógica proposicional [19](#_bookmark17)
     3. Satisfacibilidad [20](#_bookmark18)
     4. Teoría interpretativa [22](#_bookmark19)
  2. Sistemas formales y deducción natural. [26](#_bookmark22)
     1. Motivación [26](#_bookmark23)
     2. Sistemas formales [27](#_bookmark24)
     3. Deducción Natural [28](#_bookmark25)

#### Lógica Primer Orden [37](#_bookmark27)

* 1. Sintaxis, semántica y teoría interpretativa. [39](#_bookmark28)
     1. Sintaxis de la lógica de primer orden. [39](#_bookmark29)
     2. Semántica de la lógica de primer orden [47](#_bookmark34)
     3. Satisfabilidad [51](#_bookmark36)
     4. Teoría interpretativa de la lógica de primer orden. [52](#_bookmark38)
  2. Teorema de la demostración y deducción natural. [56](#_bookmark42)
     1. Decibilidad de la lógica de Primer Orden. [56](#_bookmark43)

ÍNDICE GENERAL

|  |  |  |
| --- | --- | --- |
|  | 3.2.2 Deducción Natural | [57](#_bookmark44) |
| 3.3 Tema 3.3: | [59](#_bookmark45) |
| 3.3.1 Objetivo. | [59](#_bookmark46) |
| 3.3.2 Forma Normal de Skolem | [60](#_bookmark47) |
| 3.3.3 Forma Clausular | [62](#_bookmark49) |
| 3.4 Método de resolución de Robinson. | [62](#_bookmark50) |
| 3.4.1 Introducción | [62](#_bookmark51) |
| 3.4.2 Interpretaciones de Herbrand | [63](#_bookmark52) |
| 3.4.3 Teorema de Herbrand | [65](#_bookmark54) |
| 3.4.4 Método de Resolución de Robinson | [66](#_bookmark55) |
| 3.4.5 Resolución con Unificador de Máxima Generalidad | [68](#_bookmark57) |
| 3.4.6 Estrategias de resolución | [71](#_bookmark59) |
| **4** | **Introducción a la Programación Lógica** | [**73**](#_bookmark60) |
|  | 4.1 Tema 4.1: | [74](#_bookmark61) |
|  | 4.1.1 Estrategia de resolución SLD | [74](#_bookmark62) |
|  | 4.1.2 Árbol de derivación | [74](#_bookmark63) |
|  | 4.2 Tema 4.2: | [75](#_bookmark65) |
|  | 4.2.1 Programación Declarativa | [75](#_bookmark66) |
|  | 4.2.2 Programación Funcional | [76](#_bookmark67) |
|  | 4.2.3 Programación Lógica | [79](#_bookmark70) |
|  | 4.2.4 UTD HackReason | [82](#_bookmark74) |

## Agradecimientos

Esta obra, contiene en formato de apuntes las transparencias de Arias, Joaquín (2022). [Lógica: desde Aristóteles hasta Prolog](http://hdl.handle.net/10115/20014). Madrid: Servicio de Publicaciones de la Uni- versidad Rey Juan Carlos. ISBN:978-84-09-38265-1, las cuales están basada en tras- parencias de Antonio González Pardo (URJC’20) & Pepa Hernández (UPM’11),



# Cápitulo 1 Introducción

|  |  |  |
| --- | --- | --- |
| 1.1 | Por qué es necesaria, tipos e historia. | [**2**](#_bookmark2) |
|  | 1.1.1 Introducción | [2](#_bookmark3) |
|  | Lenguaje natural | [2](#_bookmark3) |
|  | Lenguaje formal | [3](#_bookmark3) |
|  | 1.1.2 Lógica | [3](#_bookmark4) |
|  | Sistemas lógicos | [4](#_bookmark4) |
|  | 1.1.3 Tipos de lógicas | [4](#_bookmark5) |
|  | 1.1.4 Algo de historia | [4](#_bookmark6) |
|  | Lógica y filosofía | [4](#_bookmark6) |
|  | Lógica y matemáticas | [5](#_bookmark6) |
|  | Lógica e informática | [6](#_bookmark6) |
| 1.2 | Teoría de conjuntos, relaciones, funciones y álgebra de Boole. | [**6**](#_bookmark7) |
|  | 1.2.1 Teoría de Conjuntos | [6](#_bookmark8) |
|  | Representación | [6](#_bookmark8) |
|  | Inclusión e igualdad | [7](#_bookmark8) |
|  | Operaciones | [7](#_bookmark8) |
|  | Propiedades de las operaciones | [9](#_bookmark8) |
|  | 1.2.2 Relaciones Binarias | [10](#_bookmark9) |
|  | Definiciones | [11](#_bookmark9) |
|  | Homogéneas | [11](#_bookmark9) |
|  | 1.2.3 Relaciones n-arias | [12](#_bookmark10) |
|  | 1.2.4 Funciones n-arias | [12](#_bookmark11) |
|  | 1.2.5 Algebra de Boole | [12](#_bookmark12) |
|  | Definición | [12](#_bookmark12) |
|  | Interpretación | [13](#_bookmark12) |

Propiedades y teoremas [13](#_bookmark12)

Simplificación de ecuaciones [14](#_bookmark12)

## Por qué es necesaria, tipos e historia.

### Introducción

La lógica formal es la ciencia que estudia las leyes de inferencia en los razona- mientos.

Trata de resolver diversos problemas basándose en la formación del lenguaje y sus reglas básicas.

Se aplica en multitud de áreas:

* + - * En matemáticas para demostrar teoremas
      * En ciencias de la computación para verificar si son o no correctos los pro- gramas
      * En las ciencias física y naturales, para sacar conclusiones de experimentos
      * En las ciencias sociales y en la vida cotidiana, para resolver una multitud de problemas.

#### Lenguaje natural

Lenguaje natural: lenguaje usado en la comunicación humana. Dentro del lenguaje encontramos sintaxis y semántica.

* + - * Sintaxis: las reglas de formación de frases correctas.
      * Semántica: las frases deben tener significado completo e independiente.

#### Análisis sintáctico



No es válida sintácticamente.

#### Análisis semántico

Una vez que una frase es válida sintácticamente, se puede analizar su semántica.

* + - * + Si tiene (o no) sentido desde el punto de vista semántico.



#### Limitaciones

Es complicado (o casi imposible) dar una representación completa de las reglas sintácticas de los lenguajes hablados.

No podemos formular afirmaciones que se definan como correctas (sintáctica- mente) o verdaderas (semánticamente) sin ambigüedad.

Tenemos que definir un lenguaje más preciso: **Un lenguaje formal**.

#### Lenguaje formal

El lenguaje formal se construye a partir de proposiciones

* + - * + Elementos básicos (atómicos)
        + Simples.
        + Se les puede asociar valores de verdad sin ninguna ambigüedad.

### Lógica

El objetivo de la lógica es estudiar la validez formal de un razonamiento.

**Razonamiento**: la obtención de un nuevo conocimiento (conclusión) a partir de una serie de conocimientos (premisas).

**Validez formal de un razonamiento**: si la conclusión es verdadera, siendo las premisas verdaderas.

#### Sistemas lógicos

La estructura de todo sistema lógico viene definido por:

* + - * **Sintaxis**: definición axiomática de los elementos básicos del lenguaje y de las reglas que permiten obtener nuevas expresiones correctas.
      * **Semántica**: definición de un conjunto de significados (verdadero o falso) que permiten definir la validez de un razonamiento.
      * **Sistemas de demostración**: sistemas formales que permiten averiguar cuándo un razonamiento es (o no es) válido.

### Tipos de lógicas

**Lógica proposicional**: se estudian las fórmulas proposicionales construidas a partir de fórmulas atómicas y conectivos lógicos.

**Lógica de predicados de primer orden**: es una generalización de la lógica de proposiciones. Distingue entre los objetos del discurso y sus relaciones entre ellos.

#### Lógica proposicional

“Tiene un perro” *p*.

“Tiene una mascota” *m*.

“Es adorable” *a*.

#### Lógica de primer orden

“*x* es una mascota” *M*(*x*).

“*x* es un perro” *P*(*x*).

“*x* es adorable” *A*(*x*).

“Si tiene un perro y tiene una mascota, entonces es adorable”

*p^m ! a*

“Todas las mascotas que sean pe- rros, son adorables”

*8x.M*(*x*) *^P*(*x*) *! A*(*x*)

### Algo de historia

#### Lógica y filosofía

**S. IV a.C.:** Aristóteles formaliza el razonamiento humano. Fundó la lógica clá- sica o lógica aristotélica. Por Ejemplo:

*Todos los hombres son mortales. Sócrates en un hombre. Luego Sócrates es mortal.*

**En el s. XIII:** Santo Tomás de Aquino empleó la lógica en el contexto de discu- siones teológicas.

**s. XVIII:** Leibniz fue el primero en formular la lógica como base del razona- miento matemático.

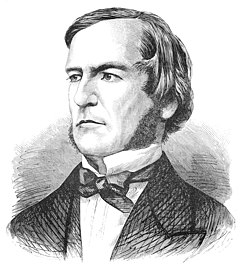


Aristóteles Leibniz

#### Lógica y matemáticas

**En 1854:** George Boole definió la lógica como sistemas formal. Esto proporcio- nó un modelo algebraico de la lógica de proposiciones.

**En 1879:** Gottlob Frege formalizó la lógica de predicados.

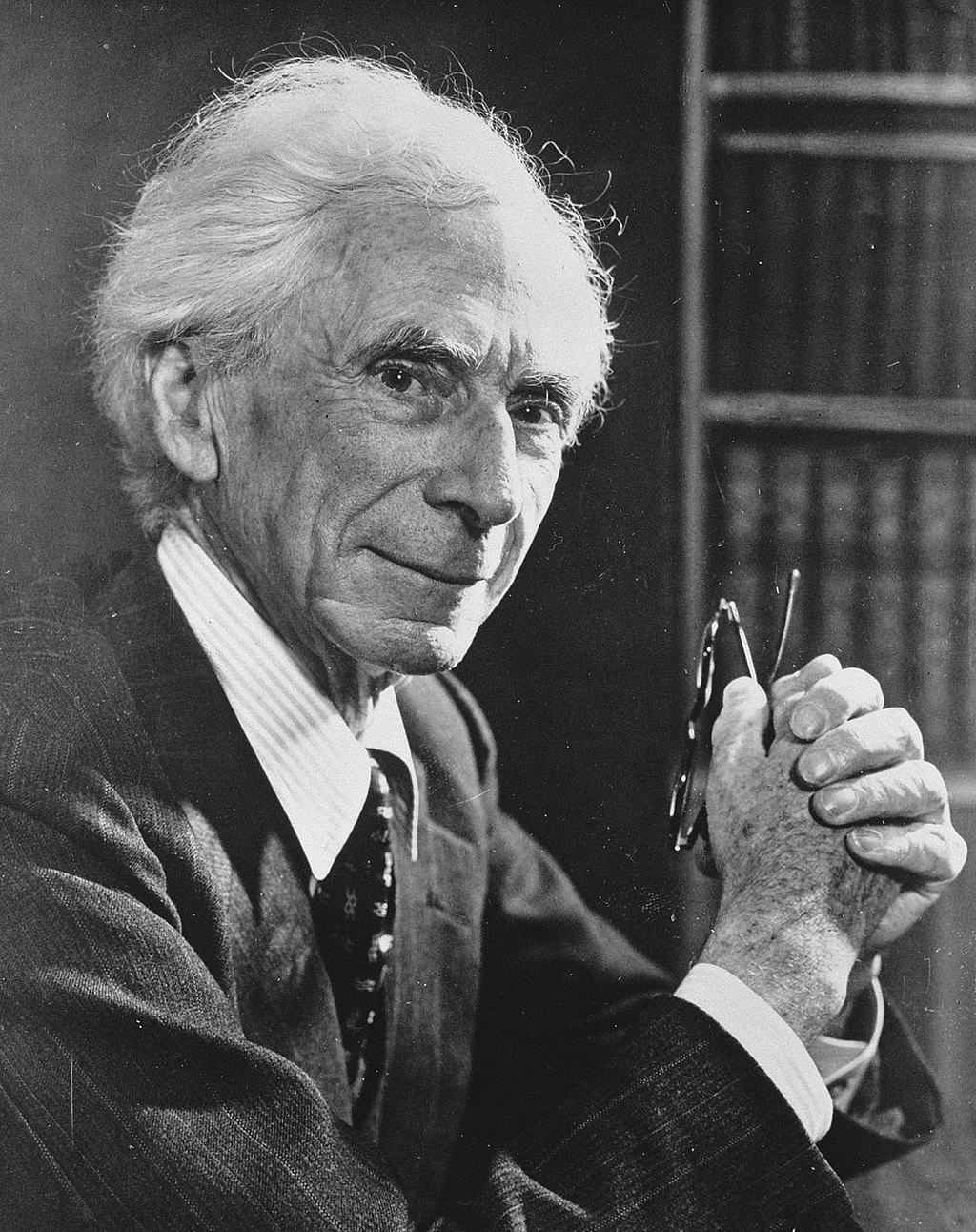


Boole Frege

**1910-1913:** Alfred North Whitehead y Bertrand Russell publican “Principia Mathematica”.

**1920:** Hilbert propone el problema de la axiomatización de las matemáticas...

**...en 1930:** Gödel rebate esta posibilidad al demostrar los teoremas de incomple- titud.



Russel Gödel

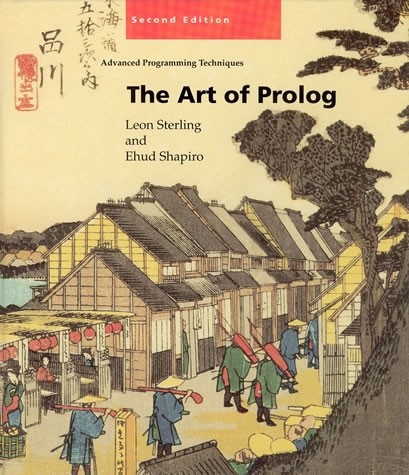
#### Lógica e informática

**1965:** Alan Robinson publica un método de resolución para lógica de primer orden. Sienta las bases de la deducción automática:

* Verificación automática de programas: a partir de su especificación formal y utilizando demostradores automáticos de teoremas.

**1972:** Alain Colmerauer crea Prolog, el primer lenguaje de programación lógica. Sienta las bases de la inteligencia artificial:

* Permite inferir/deducir conocimiento a partir de una base de conocimientos y una serie de reglas (de inferencia).



Prolog Colmerauer

## Teoría de conjuntos, relaciones, funciones y álge- bra de Boole.

### Teoría de Conjuntos

Definición de conjunto y pertenencia.

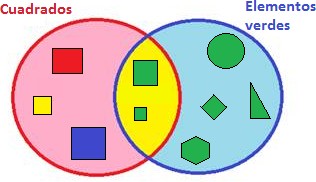
Un conjunto *A* es una colección, finita o infinita, de objetos de un universo *U* tal que para todo objeto *x* se puede determinar si *x* pertenece a *A* (*x 2 A*).

Si no perteneciera se indicaría como: (*x 62 A*).

Los objetos que pertenecen a un conjunto se conocen como **elementos** del con- junto.

#### Representación

Una manera muy común de representar relaciones y operaciones entre conjuntos son los diagramas de Venn.



#### Inclusión e igualdad

Si todo elemento *x* de un conjunto *A* es también elemento de un conjunto *B*, diremos que *A* está contenido en *B*, o que *A* es un subconjunto de *B*:

*A ✓ B*

Si *A* es un subconjunto de *B* y existe un elemento de *B* que no pertenece a *A*, entonces *A* es un subconjunto propio de *B*:

*A ⇢ B*

Dos conjuntos serás iguales (*A ⌘ B*), si contienen los mismo elementos:

*A ✓ B y B ✓ A*

#### Conjunto vacío

El conjunto vacío 0/ es el conjunto que no tiene elementos:

/0 = *{}*

Dado el conjunto *A* = *{*/0*}*. ¿Cual es la respuesta correcta?:

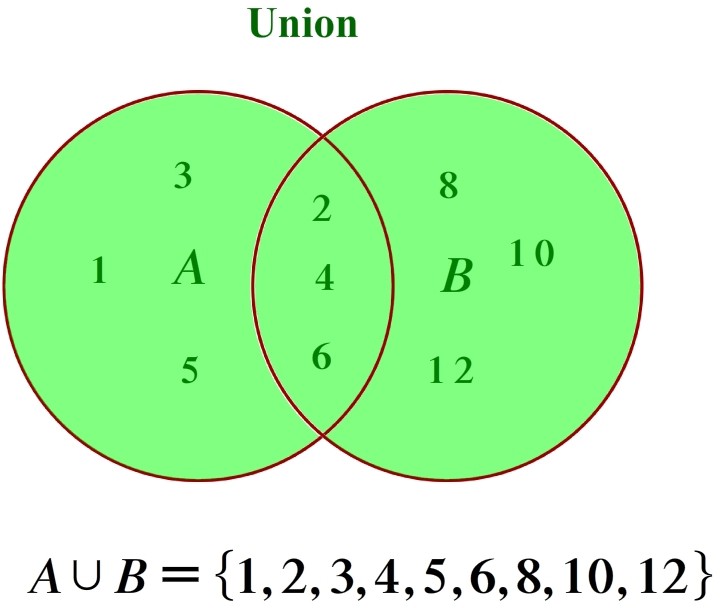
*A* = /0 *o A 6*= /0

#### Operaciones Unión

La unión de dos conjuntos *A* y *B*, es el conjunto *A B* de todos los elementos de *A* o de *B*, es decir:

*[*

*A[ B* = *{x* : *x 2 A* o *x 2 B}*

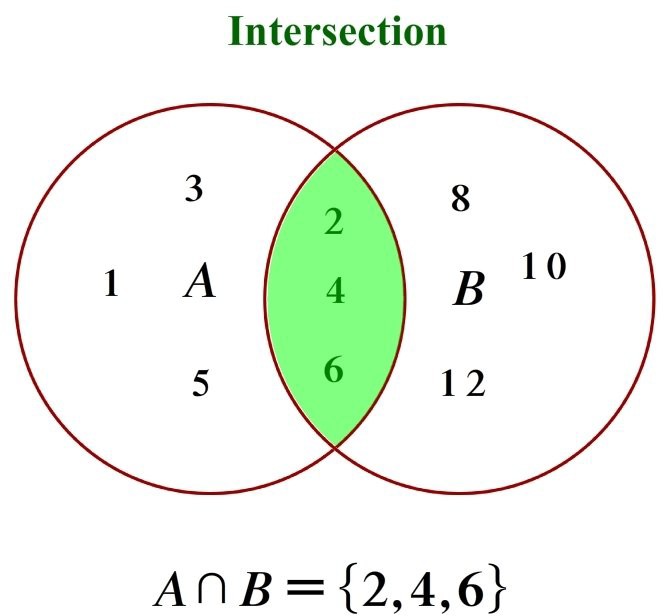


#### Intersección

La intersección de dos conjuntos *A* y *B*, es el conjunto *A B* de todos los ele- mentos que pertenecen a *A* a de *B*, es decir:

*\*

*A\ B* = *{x* : *x 2 A* y *x 2 B}*

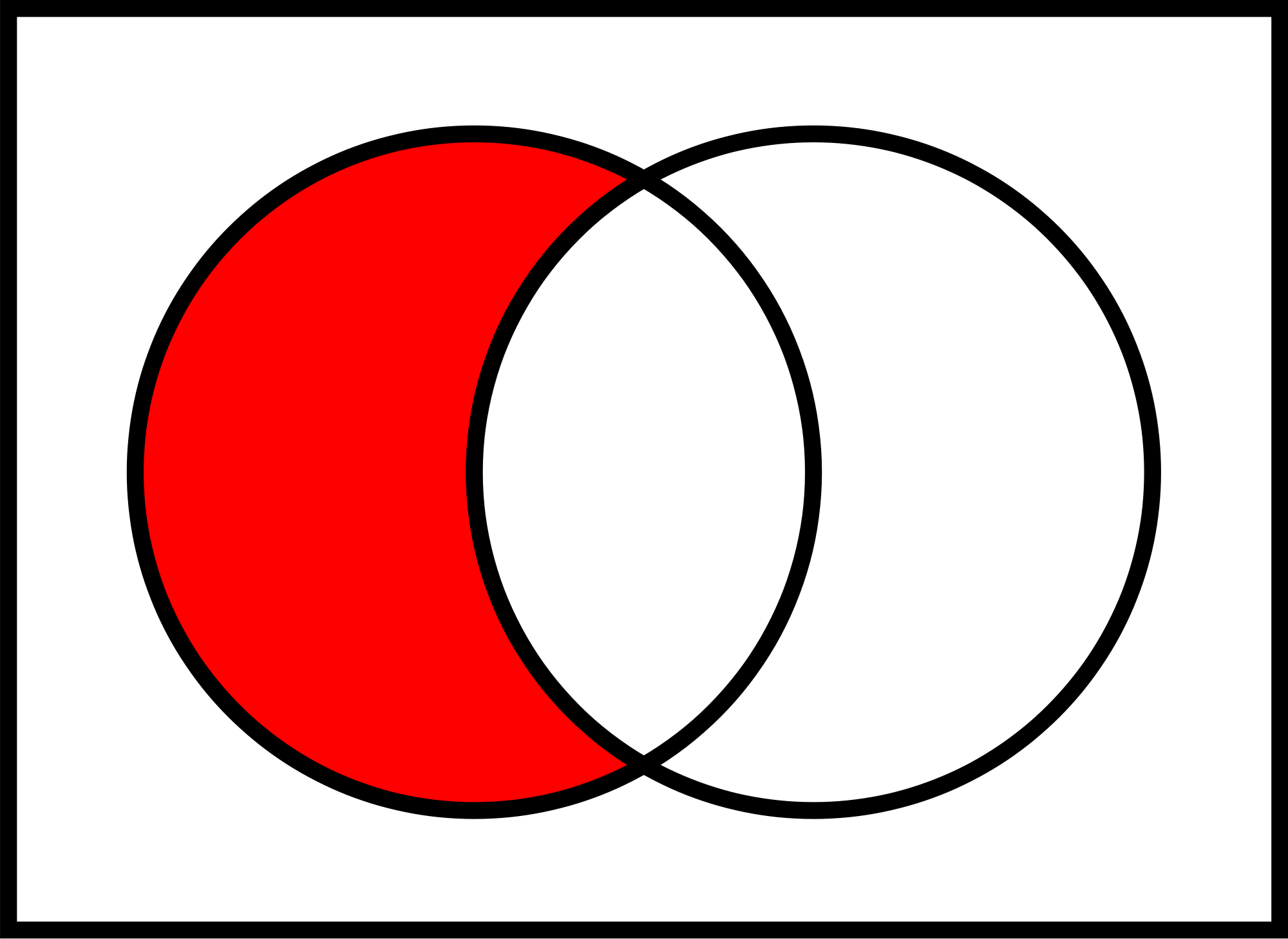


#### Complemento relativo

El complemento relativo de *B* respecto de *A* (o diferencia *A- B*), es el conjunto

*A\B* de todos los elementos de *A* que no pertenecen a *B*, es decir:

*A\B* = *{x* : *x 2 A* y *x 62 B}*



#### Producto cartesiano

El producto cartesiano de dos conjuntos no vacíos *A* y *B*, es el conjunto de todos los pares ordenados (*a, b*) con *a 2 A* y *b 2 B*, es decir:

*AxB* = *{*(*a, b*) *\ a 2 A* y *b 2 B}* Dado *A* = *{*1*,* 2*}* y *B* = *{a, b, c}*

*AxB* = *{*(1*, a*)*,* (1*, b*)*,* (1*, c*)*,* (2*, a*)*,* (2*, b*)*,* (2*, c*)*}*

#### Partes de un conjunto

Sea *A* un conjunto, se llama conjunto de las partes de *A*, *P*(*A*), al conjunto cuyos elementos son exactamente los subconjuntos de *A*.

Dado *A* = *{*1*,* 2*,* 3*}*, el conjunto de las partes de *A* es:

*P*(*A*)= *{*/0*, {*1*},{*2*},{*3*},{*1*,* 2*},{*1*,* 3*},{*2*,* 3*},{*1*,* 2*,* 3*}}*

#### Propiedades de las operaciones

Sean *A*, *B* y *C* tres conjuntos. Las principales propiedades de las operaciones con conjuntos son las siguientes:

1. **Idempotencia** de la unión y de la intersección:

*A[ A* = *A*

*A\ A* = *A*

1. **Conmutatividad** de la unión y de la intersección:

*A[ B* = *B[ A*

*A\ B* = *B\ A*

1. **Asociatividad** de la unión y de la intersección:

*A[* (*B[C*)= (*A[ B*) *[C*

*A\* (*B\C*)= (*A\ B*) *\C*

1. **Distributiva** de la unión respecto de la intersección y de la intersección respecto a la unión:

*A[* (*B\C*)= (*A[ B*) *\* (*A[C*)

*A\* (*B[C*)= (*A\ B*) *[* (*A\C*)

#### Leyes de Morgan:

*C\*(*A[ B*)= (*C\A*) *\* (*C\B*) *A[ B* = *A\ B*

*C\*(*A\ B*)= (*C\A*) *[* (*C\B*) *A\ B* = *A[ B*

1. **Otras** propiedades interesantes:

(*A\B*) *[ B* = *A[ B*

(*A\B*) *\ B* = /0

*A\*(*A\B*)= *A\ B*

#### Cardinal

El cardinal de un conjunto, *Card*(*A*) o *A* , es el número de elementos de dicho conjunto.

*| |*

Sean *A* y *B* dos conjuntos finitos cualesquiera:

*Card*(*A[ B*)= *Card*(*A*)+*Card*(*B*) *-Card*(*A\ B*)

*|A[ B|* = *|A|* + *|B|- |A\ B|*

### Relaciones Binarias

Una relación binaria entre *A* y *B* es un subconjunto *R* del producto cartesiano

*AxB*. Si (*a, b*) *2 R* se dirá que *a* y *b* están relacionados:

*aRb*

Dado *A* = *{*1*,* 2*}* y *B* = *{a, b, c}*, cuyo producto cartesiano es:

*AxB* = *{*(1*, a*)*,* (1*, b*)*,* (1*, c*)*,* (2*, a*)*,* (2*, b*)*,* (2*, c*)*}*

Si consideramos *R* = *{*(1*, c*)*,* (2*, b*)*,* (1*, a*)*}* entonces tenemos que:

1*Rc,* 2*Rb,* 1*Ra*

#### Definiciones

Dada una relación binaria *R ✓ AxB*, se denomina:

* Domino de *R*:

*dom*(*R*)= *{x 2 A* : *9y 2 B |* (*x, y*) *2 R} dom*(*R*) *✓ A*

* Imagen directa (o rango) de *R*:

*Im*(*R*)= *{y 2 B* : *9x 2 A |* (*x, y*) *2 R} Im*(*R*) *✓ B*

* Imagen inversa (o recíproca) de un subconjunto *C* de *B*:

*R-*1(*C*)= *{x 2 A* : *9y 2 C |* (*x, y*) *2 R} R-*1(*C*) *✓ A*

* Codominio de *R* al conjunto *B*

#### Homogéneas

Sea *R* una relación binara homogénea en un conjunto *A*, no vacío (*R AxA*). La relación binaria *R* puede ser:

*✓*

1. **Reflexiva**: *8x 2 A |* (*x, x*) *2 R*
2. **Simétrica**: *8x, y 2 A |* (*x, y*) *2 R !* (*y, x*) *2 R*
3. **Antisimétrica**: *8x, y 2 A |* (*x, y*) *2 R^* (*y, x*) *2 R ! x* = *y*
4. **Transitiva**: *8x, y, z 2 A |* (*x, y*) *2 R^* (*y, z*) *2 R !* (*x, z*) *2 R*

#### de Equivalencia

Una relación binaria homogénea *R*, en un conjunto no vacío *A*, se denomina relación de equivalencia si es *reflexiva*, *simétrica* y *transitiva*.

Si *R* es una relación de equivalencia en lugar de *R* se escribe *⇠*.

Si *a A* y es una relación de equivalencia en *A*, se define la clase de equiva- lencia como:

*2 ⇠*

*C*(*a*)= *{x 2 A* : *x ⇠ a}*

### Relaciones n-arias

Una relación n-aria entre *n* conjuntos *A*1, *A*2, *.. .* , *An*, es un subconjunto *R* del producto cartesiano *A*1*xA*2*x... xAn*.

Es una generalización de la relación binaria donde R está formada por una tupla de *n* términos:

*R* = *{*(*x*1*, x*2*,..., xn*) : *x*1 *2 X*1 *^ x*2 *2 X*2 *^ .. .*

*.. . ^ xn 2 Xn ^ R*(*x*1*, x*2*,..., xn*)= verdadero*}*

El predicado n-ario: *R*(*x*1*, x*2*,..., xn*) es una función que asigna el valor de verdad verdadero si y solo si (*x*1*, x*2*,..., xn*) *2 R*.

### Funciones n-arias

Una función (o aplicación) n-aria, *f* : *A*1*xA*2*x... xAn B*, de un conjunto no va- cío *A* = *A*1*xA*2*x... xAn* a un conjunto no vacío *B* se puede definir como una “regla de correspondencia” que asigna a cada elemento elemento(*a*1*, a*2*,..., an*) *2 A* un único elemento *b 2 B*:

*!*

*f* (*a*1*, a*2*,..., an*)= *b*

Una función *f* : *A*1*xA*2*x... xAn B* es a su vez una relación binaria en- tre el conjunto *A* = *A*1*xA*2*x... xAn* y el conjunto *B*, tal que a cada elemento (*a*1*, a*2*,..., an*) *A* le corresponde un único elemento *f* (*a*1*, a*2*,..., an*) del con- junto *B*.

*!*

*2*

### Algebra de Boole

#### Definición

Definida en 1847 por el matemático inglés George Boole.

Es un álgebra con sólo dos valores, p.ej.: **Falso** o **Verdadero**.

...una variable *lógica* tiene dos valores posibles, uno excluye al otro. Sobre estos valores definió tres operaciones básicas, p.ej.:

* + - * NOT (*¬*): negación lógica o función complemento.
        + OR (*\_*): suma lógica o función unión.
        + AND (*^*): producto lógico o función intersección.

*¬ \_* V F *^* V F

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| V | F | V | V V | V | V | F |
| F | V | F | V F | F | F | F |

Son Álgebras de Boole (entre otras):

(*{*/0*,U }, ⇠, EB, 0*), (*{F,V}, ¬, \_, ^*), (*{*0*,* 1*}, ,* +*,·*),y (*P*(*U* )*, ,[,\*).

#### Interpretación

Los elementos del álgebra de Boole representa estados diferentes de un disposi- tivo, p.ej., abierto/cerrado, encendido/apagado, etc.

Matemáticamente se representan por 1 y 0, y responde a la pregunta:

* + - * + ¿Cómo expresar la clase *not-X* (clase de individuos que no son Xs)?
        + La clase *X* y la *not-X* juntas hacen el Universo, representado por 1.
        + Por lo tanto, si la clase *X* es *x*, la clase *not-X* es 1 *- x*.

... y muchas más. p.ej.:

* + - * + Todos *Xs* son *Ys* y todos *Ys* son *Xs*. *x* = *y*
        + Todos *Xs* son *Ys*. *x*(1 *- y*)= 0
        + No *Xs* son *Ys*. *xy* = 0
        + Todos *Ys* son *Xs* y algunos *Xs* son *Ys*. *...*
        + etc...

#### Propiedades y teoremas

Usando la notación matemática del álgebra de Boole:

Propiedades

Conmutativa Elemento neutro Distributiva Asociativa Complementario

Teoremas

Idempotencia Identidad Absorción DeMorgan

Suma Producto

*x* + *y* = *y* + *x x · y* = *y · x*

0 + *x* = *x* 1 *· x* = *x*

*x ·* (*y* + *z*)= (*x · y*)+ (*x · z*) *x* + (*y · z*)= (*x* + *y*) *·* (*x* + *z*)

*x* + (*y* + *z*)= (*x* + *y*)+ *z x ·* (*y · z*)= (*x · y*) *· z x* + *x* = 1 *x · x* = 0

*x* + *x* = *x x · x* = *x*

*x* + 1 = 1 *x ·* 0 = 0

*x* + *x · y* = *x x ·* (*x* + *y*)= *x*

*x* + *y* = *x · y x · y* = *x* + *y*

#### Simplificación de ecuaciones

Aplicando propiedades y teoremas podemos simplificar ecuaciones:

*x*2 *· x*1 + *x*2 *· x*1 + *x*2 *· x*0 ? *x*2

=

*x*2 *· x*1 + *x*2 *· x*1 + *x*2 *· x*0 = Distributiva

= *x*2 *·* (*x*1 + *x*1)+ *x*2 *· x*0 = Complemento

= *x*2 *·* 1 + *x*2 *· x*0 = Distributiva

= *x*2 *·* (1 + *x*0)= Identidad

= *x*2 *·* 1 = Elemento neutro

= *x*2

...para ver si son equivalentes y/o para construir circuitos mas baratos. Las ecuaciones simplificadas se llaman formas canónicas. Hay dos:

* + - * Suma de productos. • Producto de sumas.



# Cápitulo 2

**Lógica Proposicional**

|  |  |  |
| --- | --- | --- |
| 2.1 | Sintaxis, semántica y teoría interpretativa. | [**16**](#_bookmark14) |
|  | 2.1.1 Lenguaje proposicional | [16](#_bookmark15) |
|  | Alfabeto | [16](#_bookmark15) |
|  | Fórmula Bien Formada | [17](#_bookmark15) |
|  | Precedencia | [17](#_bookmark15) |
|  | Paréntesis | [17](#_bookmark15) |
|  | Ejercicios: FBF y paréntesis | [18](#_bookmark15) |
|  | Ejercicios: Formalización | [18](#_bookmark16) |
|  | 2.1.2 Semántica de la lógica proposicional | [19](#_bookmark17) |
|  | Funciones de verdad | [19](#_bookmark17) |
|  | Funciones de interpretación | [20](#_bookmark17) |
|  | Ejercicios | [20](#_bookmark17) |
|  | 2.1.3 Satisfacibilidad | [20](#_bookmark18) |
|  | Validez | [21](#_bookmark18) |
|  | Conclusión | [21](#_bookmark18) |
|  | Ejecicios: Satisfacibilidad | [21](#_bookmark18) |
|  | 2.1.4 Teoría interpretativa | [22](#_bookmark19) |
|  | Consecuencia Lógica | [22](#_bookmark19) |
|  | Ejercicios: Consecuencia Lógica | [24](#_bookmark20) |
|  | Equivalencia lógica | [25](#_bookmark21) |
|  | Ejercicios: Equivalencia lógica | [26](#_bookmark21) |
| 2.2 | Sistemas formales y deducción natural. | [**26**](#_bookmark22) |
|  | 2.2.1 Motivación | [26](#_bookmark23) |
|  | 2.2.2 Sistemas formales | [27](#_bookmark24) |
|  | Propiedades | [27](#_bookmark24) |

2.2.3 Deducción Natural [28](#_bookmark25)

Supuestos y Premisas [28](#_bookmark25)

Teorema de la Deducción [29](#_bookmark25)

Reglas de inferencia I [29](#_bookmark25)

Deducción Natural: Ejemplo [30](#_bookmark25)

Ejercicios: Deducción Natural [32](#_bookmark25)

Reglas de inferencia Derivadas [32](#_bookmark26)

Teorema de intercambio [34](#_bookmark26)

Ejercicios: Deducción Natural [34](#_bookmark26)

## Sintaxis, semántica y teoría interpretativa.

### Lenguaje proposicional

En el contexto de la lógica formal se requiere un lenguaje:

* + - * Sintácticamente preciso y no ambiguo.
      * Con un significado (semántica) unívoco, y no que una palabra pueda signi- ficar cosas distintas según algún tipo de contexto.
      * Cuya definición sea muy compacta.

Queremos símbolos para representar hechos lógicos, es decir, enunciados por los que tiene sentido preguntarse si son verdaderos o falsos.

Los enunciados más sencillos son los que no dependen de otros:

* + - * Llueve.
      * Nadal ganó el US Open 2019.

#### Alfabeto

Los símbolos de proposición representan este tipo de hechos:

* + - * *p* puede representar “llueve”.
      * *q* puede representar “Nadal ganó el US Open 2019”.

Para enunciados más complejos necesitamos símbolos que representen la rela- ción entre los más sencillos que los componen:

* + - * llueve o nieva.
      * si *p* representa “llueve” y *r* representa “nieva”,

*p\_r* representaría “llueve o nieva”.

**Alfabeto** de un lenguaje proposicional:

* + - * + Símbolos de proposición: *p*, *q*, *r*, *.. .* , *p*1, *p*2, *.. .*
        + Conectivas lógicas: *¬*, *\_*, *^*, *!*, *$*.

#### Fórmula Bien Formada

Del conjunto de todas las posibles secuencias (finitas) de símbolos de un alfabe- to, ¿cuáles se consideran expresiones bien formadas?

Son **fórmula bien formada** (*FBF*):

* + - * + *F* si *F* es un símbolo de proposición.
        + *¬F* si F es un símbolo de proposición.
        + y si *F* y *G* son FBFs, también son FBFs:

*F ^G* (conjunción).

*F \_G* (disyunción).

*F ! G* (implicación).

*F $ G* (doble implicación).

*Definición recursiva de fórmulas bien formadas.*

Dado un alfabeto A, el conjunto (infinito) de FBFs que se pueden definir sobre A se denomina FBF*A*.

#### Precedencia

¿Que dice la siguiente fórmula?

(*! ^*

*p q r* **p** *!* **q** *^r* (*a*)

*p !* **q** *^* **r** (*b*)

1. Si *p* entonces *q*, y además *r*.
2. Si *p* entonces suceden dos cosas: *q* y *r*.

Se define la siguiente precedencia entre las conectivas:

* *¬* mayor precedencia que *^* y *\_*.
* *^* y *\_* mayor precedencia que *!* y *$*.
* En caso de igual precedencia se agrupan de derecha a izquierda.

#### Paréntesis

Si los paréntesis siguen las reglas de precedencia no son necesarios:

* *p^q^r* es igual a *p^* (*q^r*)
* *p^q\_r* es igual a *p^* (*q\_r*)
* *q ! r*1 *^¬r*2 es igual a *q !* (*r*1 *^* (*¬r*2))
* *cgm^ p\_¬¬q $ ll* es igual a (*cgm^* (*p\_* (*¬*(*¬q*)))) *$ ll*
* *p ! q ! r ! s* es igual a *p !* (*q !* (*r ! s*))
* *¬¬¬¬¬¬¬p* es igual a *¬*(*¬*(*¬*(*¬*(*¬*(*¬*(*¬p*))))))

Los paréntesis son necesarios para dar otro “significado”:

* *p^q ^ r* sería (*p^q*) *^r*
* *cgm^ p \_ ¬¬q $ ll* sería (*cgm^ p*) *\_* (*¬¬q $ ll*)
* *p ! q ! r ! s* sería (*p ! q*) *!* (*r ! s*)

#### Ejercicios: FBF y paréntesis

Determinar si son FBFs las siguientes fórmulas y eliminar paréntesis superfluos usando las convenciones de precedencia:

1. (*q^¬*(*p*)) *!* (*¬*(*q*) *\_r*).

2. (*p^¬*(*q\_r*)) *\_* (*¬*(*p*) *\_q*).

3. ((*p \_q*) *\_¬*(*r*)) *$* (*r ^q*). 4. (*p^¬*(*q\_r*)) *\_* (*¬*(*! p*)).

5. (*p*) *$* ((*p ^q*) *\_* (*p^* (*¬*(*q*)))).

#### Ejercicios: Formalización

Formalizar en lógica proposicional:

1. Es necesario abrir la botella para disfrutar su contenido.
2. Basta con romper el sello para perder la garantía.
3. Llama al 112 cuando tengas una emergencia.
4. Perderé el tren a menos que coja un taxi.
5. Cambia la bombilla sólo y siempre que esté fundida.
6. No iré al cine a no ser que me inviten.
7. No pulse la alarma de incendios excepto cuando detecte humo en la esca- lera.
8. Pulse la alarma sólo y únicamente si detecta humo en la escalera.

Formalizar en lógica proposicional (Manzano y Huertas, 2004):

1. Sólo si Pedro juega(p) jugará también Alex(q)
2. Pedro irá al dentista(p), tanto si quiere(q) como si no quiere(*¬*q)
3. La magia se revela(p) sólo si Pinocho miente(q) o Blancanieves muerde la manzana(r)
4. El certificado tiene validez(p) si está firmado por el director(q) o el tutor del proyecto(r)
5. La inflación aumentará(p) a menos que baje la emisión de moneda(q) u ocurra un milagro(r)

Formalizar en lógica proposicional (Manzano y Huertas, 2004):

1. Leeré a Proust(p) si me voy de vacaciones(q) y encuentro sus libros en oferta(r)
2. Si el mal existe en el mundo(p) y no se origina por las acciones humanas(q), entonces Dios no quiere(r) o no puede(s) impedirlo
3. Te regalaré el cuadro que te gusta(p) y viajaremos juntos a Italia(q) cuando me toque la lotería(r), o dejo de llamarme Ernesto(s)
4. Es necesario que llueva(p) o que haga viento(q) para que disminuya la con- taminación(r)
5. Si llueve(p) y hace viento(q), disminuye la contaminación(r)

### Semántica de la lógica proposicional

Proposición: Una condición/afirmación posible del mundo sobre el que quere- mos decir algo.

Una proposición puede ser *Verdadera* (V) o *Falsa* (F). Proposiciones simples:

* + - * Su valor de verdad no depende de otra proposición. Proposiciones compuestas (FBF):
      * Su valor de verdad depende del que tengan las proposiciones simples que la definan y
      * del significado de las conectivas (definido por las funciones de verdad).

#### Funciones de verdad

Primero definimos una función de verdad (*s*) para cada conectiva:

* + - * *s¬*(*V* )= *F s¬*(*F*)= *V*
      * *s^*(*V,V* )= *V s^*(*V, F*)= *s^*(*F,V* )= *s^*(*F, F*)= *F*
      * *s\_*(*F, F*)= *F s\_*(*V, F*)= *s\_*(*F,V* )= *s\_*(*V,V* )= *V*
      * *s!*(*V, F*)= *F s!*(*F, F*)= *s!*(*F,V* )= *s!*(*V,V* )= *V*
      * *s$*(*V,V* )= *s$*(*F, F*)= *V s$*(*V, F*)= *s$*(*F,V* )= *F*

#### Funciones de interpretación

Una función de interpretación, *i*, asignar un significado a todas y cada una de las FBFs de un alfabeto *A*.

*i* : *FBFA ) {V, F}*

La interpretación de una FBF se define como:

* *i*(*p*)= *V/F* si *p* es una proposición *p*.
* *i*(*¬G*)= *s¬*(*i*(*G*)) si *G* es una FBF.
* y si *G* y *H* son FBFs:

*o i*(*G^H*)= *s^*(*i*(*G*)*, i*(*H*)).

*o i*(*G\_H*)= *s\_*(*i*(*G*)*, i*(*H*)).

*o i*(*G ! H*)= *s!*(*i*(*G*)*, i*(*H*)).

*o i*(*G $ H*)= *s$*(*i*(*G*)*, i*(*H*)).

#### Ejercicios

Asignar significado a las siguientes fórmulas cuando:

*i*(*p*)= *i*(*q*)= *V* y *i*(*r*)= *F*.

1. (*p ! q\_r*) *!* (*p^q ! ¬r*)

2. (*p^* (*¬q\_¬r*)) *$* (*¬*(*p\_q*) *! r*)

Asignar significado a las siguientes fórmulas para toda posible interpretación:

1. *¬*(*p\_q*) *$ ¬p^¬q*

2. (*¬p ! q*) *^* (*¬q^ p*)

3. (*p\_q*) *!* (*p^q*)

### Satisfacibilidad

Definición Se dice que una interpretación *i* satisface una fórmula proposi- cional *G 2 FBFA*, si y sólo si (sii) es verdadera bajo esa interpretación:

*i*(*G*)= *V*

Para conjuntos de fórmulas *{G*1*,..., Gn}, Gi 2 FBFA* para todo *i* : 1 *i n*:

Una interpretación *i* **satisface** *{G*1*,..., Gn}* sii

*i*(*Gi*)= *V* para todo *i* :1 *i n*

Definición

Una fórmula *G FBFA* es **satisfacible** sii

*2*

existe (al menos) una interpretación *i* tal que *i*(*G*)= *V* .

Una fórmula *G FBFA* es **insatisfacible** sii

*2*

no existe ninguna interpretación *i* tal que *i*(*A*)= *V*

**Modelo** de una fórmula: Una interpretación que la satisface. **Contramodelo** de una fórmula: Una interpretación que la hace falsa.

#### Validez

Atendiendo a su semántica, una fórmula *G* es:

**Contradicción** sii no existe una interpretación *i* tal que *i*(*G*)= *V* . **Válida** (o tautología) sii no existe una interpretación *i* tal que *i*(*G*)= *F* (se representa *`G*).

**Contingente** sii existe alguna interpretación *i* tal que *i*(*G*)= *V* y existe alguna interpretación *i0* tal que *i0*(*G*)= *F*.

Una fórmula *G* es válida sii *¬G* es una contradicción. Una fórmula *G* es contingente sii *¬A* es contingente.

#### Conclusión

Una fórmula es válida sii

* no tiene contramodelos sii
* todas sus interpretaciones son modelos sii
* todas sus interpretaciones la satisfacen. Una fórmula es una contradicción sii
* no tiene modelos sii
* todas sus interpretaciones son contramodelos sii
* es insatisfacible.

Una fórmula es contingente sii

* tiene modelos y contramodelos.

#### Ejecicios: Satisfacibilidad

Determinar para cada una de las siguientes fórmulas si es válida, contradictoria o contingente, indicando la(s) interpretación(es) que lo demuestran:

1. *p^q ! p*.
2. *p\_q ! p*.

3. *p ! ¬p*.

4. *p\_q !* (*r \_s ! p*).

5. (*p ! q*) *^* (*p^¬q*).

6. (*p ! q*) *^* (*q ! r*) *!* (*p ! r*).

7. *¬*(*p\_q*) *$ ¬p\_¬q*.

8. *¬*(*p\_q*) *$ ¬p^¬q*.

Determinar para cada una de las siguientes fórmulas si es válida, contradictoria o contingente, indicando la(s) interpretación(es) que lo demuestran:

9. (*p ! ¬q*) *^¬*(*r ^¬p*) *!* (*q ! ¬r*). 10. (*p^q ! r*) *!* (*p !* (*q ! r*)).

11. *¬*(*p ! q*) *$ p^¬q*. 12. *p !* (*q ! r*).

13. *p !* (*q^¬q ! ¬p*). 14. (*p ! q*) *^* (*q ! p*).

15. (*p ! q*) *!* ((*p ! r*) *!* (*p ! q^r*)).

### Teoría interpretativa

#### Consecuencia Lógica

Dado un lenguaje proposicional LP, un conjunto de fórmulas *{A*1*,..., An}*, *Ai 2*

*FBFLP* para todo *i* :1 *i n*, y una fórmula *B 2 FBFLP*: Consecuencia lógica

* + - * B es consecuencia lógica de *{A*1*,..., An}*.

[*A*1*,..., An*] ✏ *B*

sii toda interpretación que satisface *A*1*,..., An* también sa- tisface *B*.

*o { }*

*o* sii no existe ninguna interpretación que satisfaga

*{A*1*,..., An}* y no satisfaga a *B*.

Argumento correcto

Un argumento con premisas *A*1*,..., An* y consecuente *B* es correcto sii [*A*1*,..., An*] ✏ *B*

*{ }*

Para decidirlo se pueden hacer dos análisis:

1. Ver si todas las interpretaciones que satisfacen *A*1*,..., An* también satis- facen *B*.

*{ }*

1. o bien ver que no existe una sola interpretación que satisfaga *A*1*,..., An*

*{ }*

y no satisfaga *B*.

* El caso 1 requiere examinar todas las interpretaciones posibles y ver si se cumple la condición.
* El caso 2 podemos centrarnos en definir una interpretación i tal que i(*{A*1*,..., An}*) = V y i(*B*)= F.

#### Teoría interpretativa: Consecuencia Lógica: Ejemplo I

Analiza si se cumple la relación de consecuencia lógica:

*{p !* (*q ! r*)*, p^q}* ✏ *r*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *p* | *q* | *r* | *q ! r* | *p !* (*q ! r*) | *p^q* | [*A*1*,..., An*]  *{p !* (*q ! r*)*, p^q}* | B  *r* |
| V | V | **V** | V | V | V | **V** | **V** |
| V | V | F | F | F | V | F | F |
| V | F | V | V | V | F | F | V |
| V | F | F | V | V | F | F | F |
| F | V | V | V | V | F | F | V |
| F | V | F | F | V | F | F | F |
| F | F | V | V | V | F | F | V |
| F | F | F | V | V | F | F | F |

De todas las interpretaciones posibles, sólo una hace verdad a las dos premisas (*A*1 y *A*2), y esa interpretación también hace verdad al consecuente (*B*). Por tanto, sí hay relación de consecuencia lógica.

#### Teoría interpretativa: Consecuencia Lógica: Ejemplo II

Analiza si se cumple la relación de consecuencia lógica:

*{p^¬¬q, r}* ✏ *q\_s*

tratamos de definir un contramodelo del argumento:

1. i(*p^¬¬q*) = V sii
   1. i(*p*)=V **b)** y i(*¬¬q*) = V sii i(*¬q*) = F sii i(*q*)=V
2. i(*r*)=V
3. i(*q \_s*) = F sii
   1. i(q) = F (en contradicción con 1.b) **b)** y i(s) = F

Puesto que **no** es posible definir un contramodelo:

El argumento es correcto, **hay relación de consecuencia lógica**.

#### Teoría interpretativa: Consecuencia Lógica: Ejemplo III

Analiza si se cumple la relación de consecuencia lógica:

*{p^q, ¬*(*p ! r*)*}* ✏ *q^* (*p ! r*)

tratamos de definir un contramodelo del argumento:

1. i(*p^q*) = V sii
   1. i(*p*)=V **b)** y i(*q*)=V
2. i(*¬*(*p ! r*)) = V sii i(*p ! r*) = F sii
   1. i(*p*)=V **b)** y i(*r*)=F
3. i(*q ^* (*p ! r*)) = F sii
   1. i(*q*) = F (en contradicción con 1.b)
   2. o i(*p ! r*) = F (OK, compatible con 2). Existe un contramodelo al argumento: i(*p*) = i(*q*) = V, i(*r*)= F. El argumento no es correcto. **NO** hay relación de consecuencia lógica.

#### Ejercicios: Consecuencia Lógica

Determinar si las siguientes argumentaciones son correctas. Si no lo son, indicar la interpretación que lo demuestra (contramodelo).

1. *{p, p ! q}* ✏ *q*

2. *{¬p, p\_q}* ✏ *q*

3. *{p ! q, ¬p}* ✏ *¬q*

4. *{p ! q, ¬q}* ✏ *¬p*

5. *{p $ q, ¬p}* ✏ *q*

6. *{p^q}* ✏ *p*

7. *{¬*(*p^q*)*}* ✏ *¬p^¬q*

8. *{¬*(*p\_q*)*}* ✏ *¬p^¬q*

Determinar si las siguientes argumentaciones son correctas. Si no lo son, indicar la interpretación que lo demuestra (contramodelo).

9. *{p ! q, p}* ✏ *q*

10. *{p\_q}* ✏ *q\_ p*

11. *{p^* (*q\_r*)*}* ✏ *p*

12. *{p\_q ! r}* ✏ *q ! r*

13. *{¬¬r ^¬q}* ✏ *¬r*

14. *{p !* (*q ! r*)*}* ✏ *q !* (*p ! r*)

15. *{¬q ! r, t ! ¬q, ¬s ! ¬q}* ✏ *t \_¬s ! r*

16. *{p\_* (*q ! r*) *! q, p}* ✏ *q*

17. *{¬p ! ¬s, ¬p\_ r, r ! ¬t}* ✏ *¬s\_¬t*

18. *{*(*p ! q*) *^t,* (*r \_ p*) *^¬q, ¬t $ ¬s}* ✏ *r ^s*

#### Equivalencia lógica

Dos fórmulas A y B son (lógicamente) equivalentes (*A B*) sii para toda inter- pretación *i* se cumple que i(*A*) = i(*B*).

*$*

Esta definición implica que:

* A y B son consecuencia lógica una de la otra (*A* ✏ *B* y *B* ✏ *A*).
* La fórmula *A $ B* es válida (es una tautología). Por ejemplo: *p ! q $ ¬p\_q*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *p* | *q* | *p ! q* | *¬p\_q* | *p ! q $ ¬p\_q* |
| V | V | V | V | V |
| V | F | F | F | V |
| F | V | V | V | V |
| F | F | V | V | V |

La equivalencia entre fórmulas proporciona numerosas ventajas prácticas, entre ellas:

* permite utilizar indistintamente las fórmulas equivalentes en una demostra- ción (lo utilizaremos más adelante).
* permite reducir el tamaño de un lenguaje proposicional (disminuir el n*o* de conectivas que emplea).

Por ejemplo, cualquier lenguaje proposicional puede reducirse a otro que sólo utiliza *{¬, \_}*

* Esta reducción simplifica tareas como:

*o* construcción de sistemas sintácticos de demostración.

*o* demostración de las propiedades metalógicas del sistema formal.

*¬*(*¬p*) *$ p* doble negación

*p^ p $ p* ley de idempotencia

*p^q $ q^ p* conmutatividad de la conjunción

*p\_q $ q\_ p* conmutatividad de la disyunción *p^* (*q^r*) *$* (*p^q*) *^r* asociatividad de la conjunción *p\_* (*q\_r*) *$* (*p\_q*) *\_r* asociatividad de la disyunción

*p* (*q r*) (*p q*) (*p r*) distributividad de la conjunción res-

*^ \_ $ ^ \_ ^*

pecto a la disyunción

*p* (*q r*) (*p q*) (*p r*) distributividad de la disyunción res-

*\_ ^ $ \_ ^ \_*

pecto a la conjunción

*p q* (*p q*) (*q p*) definición de la doble implicación

*$ $ ! ^ !*

en función de la implicación

*p ! q $ ¬p\_q* implicación vs. disyunción

*¬*(*p\_q*) *$ ¬p^¬q* ley de De Morgan

*¬*(*p^q*) *$ ¬p\_¬q* ley de De Morgan

#### Ejercicios: Equivalencia lógica

De los pares de fórmulas siguientes, ¿en cuáles se cumple *A* ✏ *B*?, ¿en cuáles se cumple *B* ✏ *A*?, ¿en cuáles *A* y *B* son equivalentes?

1. A: *p\_q*, B: *q\_ p*
2. A: *p ! q*, B: *q ! p*
3. A: *p\_q ! r*, B: *p ! r*

4. A: (*p ! q*) *! r*, B: *p\_q ! r*

5. A: *p ! q*, B: *p $ q*

## Sistemas formales y deducción natural.

### Motivación

¿Por qué queremos construir un cálculo deductivo?

* + - * Es difícil determinar *{A*1*,..., An}* ✏ *B* por medios **semánticos**.

*o* Confirmar la corrección de un argumento puede ser muy costoso.

* + - * Alternativa: determinar que *B* se deduce de *{A*1*,..., An}* por medios sintác- ticos: *{A*1*,..., An} `B*

*o* Encontrar un procedimiento que nos permita construir una argumenta-

ción paso a paso, manipulando los símbolos de las fórmulas, sabiendo que cada paso es válido.

El análisis de la corrección de un argumento por medios **sintácticos** se hace siempre en un contexto o marco formal, denominado **sistema formal**.

En un sistema formal **los símbolos carecen de significado**, y al manipularlos hemos de ser cuidadosos y no presuponer nada sobre sus propiedades, salvo lo que se especifique en el sistema.

### Sistemas formales

Un sistema formal de demostración consiste en:

* + - * Un **lenguaje** formal (alfabeto y reglas sintácticas de formación de fórmu- las).
      * Un conjunto de axiomas lógicos o **axiomas** (fórmulas válidas sin prueba, podría ser vacío).
      * Un conjunto de **reglas de inferencia** para demostrar fórmulas.
      * Una definición de prueba o **demostración**.

Una teoría *T* es un sistema formal ampliado con un conjunto G de axiomas no lógicos o premisas (es decir, que se consideran como verdad): *T* [G].

* + - * Si G = 0/ entonces *T* es la teoría básica del sistema formal.

Una **demostración** o prueba de una fórmula *B* en una teoría *T* [G] (escrito *T* [G]

*`*

1. es una secuencia finita de fórmulas tal que:
   * Toda fórmula de la secuencia es:

*o* Un axioma o premisa de la teoría.

o, el resultado de aplicar una regla de inferencia a fórmulas anteriores en la secuencia.

*o*

* + *B* es la última fórmula de la secuencia

Un **teorema** de una teoría *T* [G] es una fórmula para la que existe al menos una demostración en *T* [G]

*T* [G] *`B* indica que *B* se deduce de *T* [G] (*B* es teorema de *T* [G])

#### Propiedades

**Corrección**: Teorema de validez. Todos los teoremas de *T* [G] son conse- cuencias lógicas de G:

si *T* [G] *`B* entonces G ✏ *B*.

**Completitud**: Teorema de completitud.

Dada una teoría *T* [G], todas las consecuencias lógicas de G son teoremas de *T* [G]:

si G ✏ *B* entonces *T* [G] *`B*.

Si el cálculo es **correcto** y **completo** entonces *`* y ✏ son equivalentes.

### Deducción Natural

Es un sistema formal para la lógica proposicional.

* + - * Lenguaje: Lenguaje proposicional.
      * Axiomas: No tiene.
      * Reglas de inferencia: dos por cada conectiva:

*o* Introducción y eliminación.

* + - * Demostración: es una secuencia finita de fórmulas en la que cada elemento es:

*o* Un *supuesto* o *premisa* de la teoría.

o, el resultado de la aplicación de una regla de inferencia a fórmulas anteriores en la secuencia.

*o*

y la última fórmula de la secuencia es la fórmula probada.

#### Supuestos y Premisas

Son fórmulas añadidas a una teoría básica *T* y pueden aparecer en una prueba sin requerir demostración. Sin embargo:

* + - * Las **premisas**: son añadidas permanentemente
      * Los **supuestos**: son incorporados temporalmente a *T* :

*o* es **introducido** en un determinado punto de la demostración

*o* luego, es **cancelado** (descargado) en otro punto posterior,

*o* y como resultado una **nueva fórmula** queda demostrada. Lo que significa usar un supuesto es lo siguiente:

* + - * “supongamos que A”
      * “entonces demuestro (usando A) que B”
      * “en realidad acabo de mostrar que si tuviera A como premisa, entonces podría demostrar B”
      * “eso equivale a decir que he demostrado la implicación *A ! B*”

#### Teorema de la Deducción

Siendo *T* [*A*1*, A*2*,..., An*] una teoría básica ampliada con un conjunto de *n* **pre- misas**, *Ai*, si la incorporación como **supuesto** de un fórmula *A* permite deducir otra fórmula *B*: entonces

*T* [*A*1*, A*2*,..., An*] *[{A} `B*

entonces:

*T* [*A*1*, A*2*,..., An*] *`A ! B*

Teorema de la deducción: En general, tanto para premisas como para su- puestos:

*T* [*A*] *`B* si y sólo si *T `A ! B*

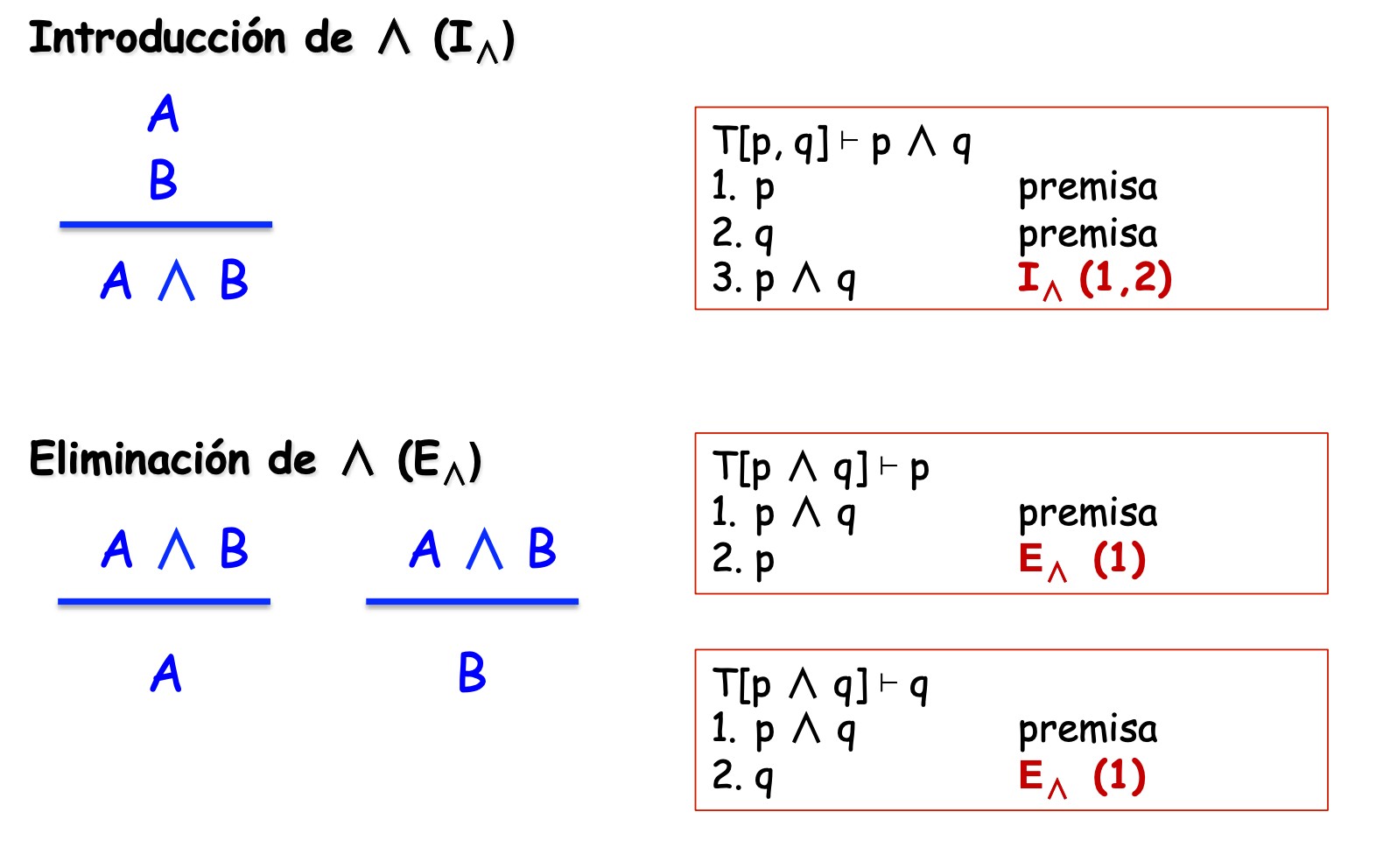
#### Reglas de inferencia I

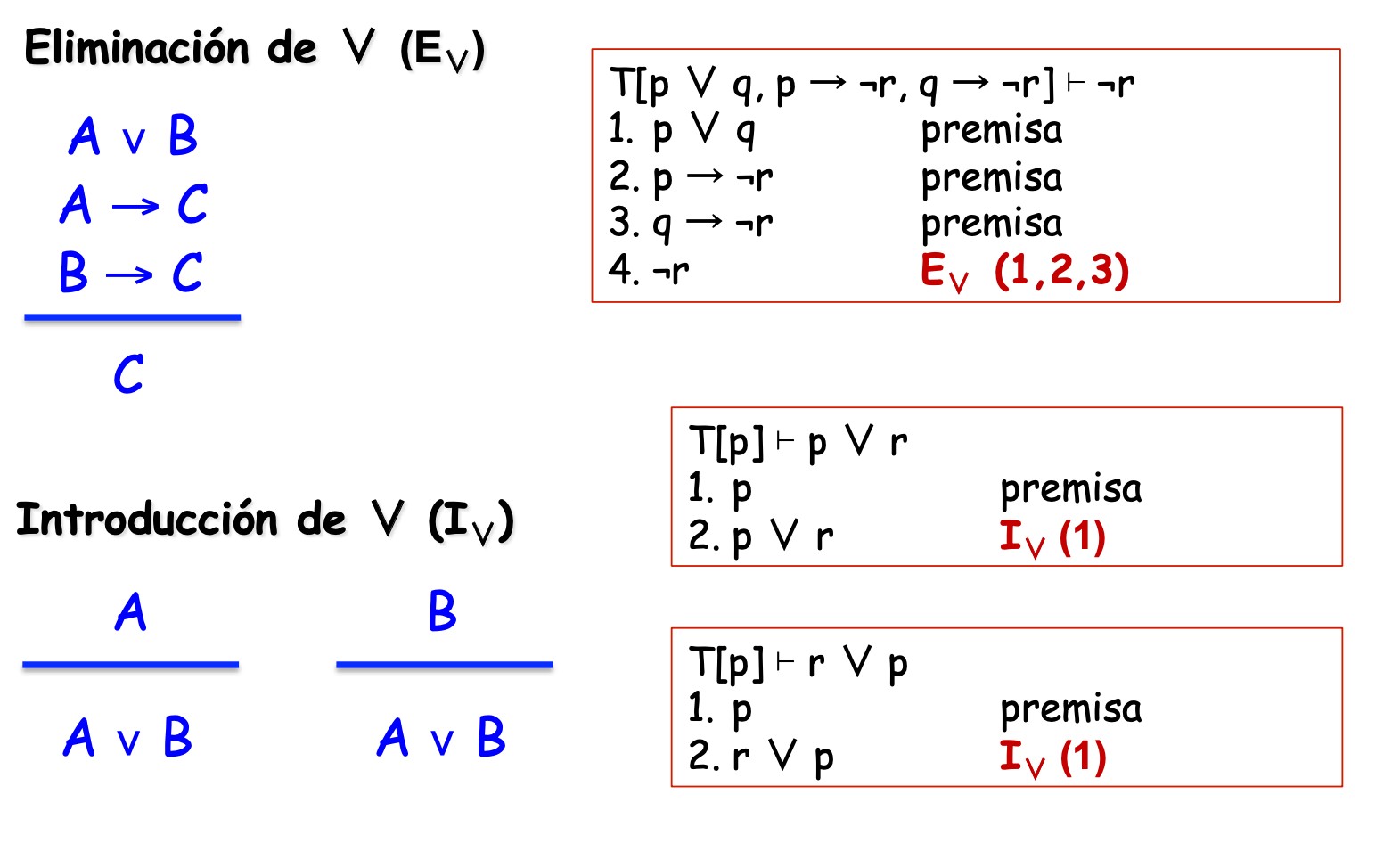
En la definición de las reglas de inferencia vamos a usar A y B, que no son sím- bolos de proposición: son variables sobre fórmulas del lenguaje (**metavariables**)

* + - * + Mediante metavariables podemos razonar sobre conjuntos (infinitos) de fórmulas que comparten una misma forma lógica.

Por ejemplo: *A^¬A* agrupa *p^¬p*, *q^¬q*, *.. .* .

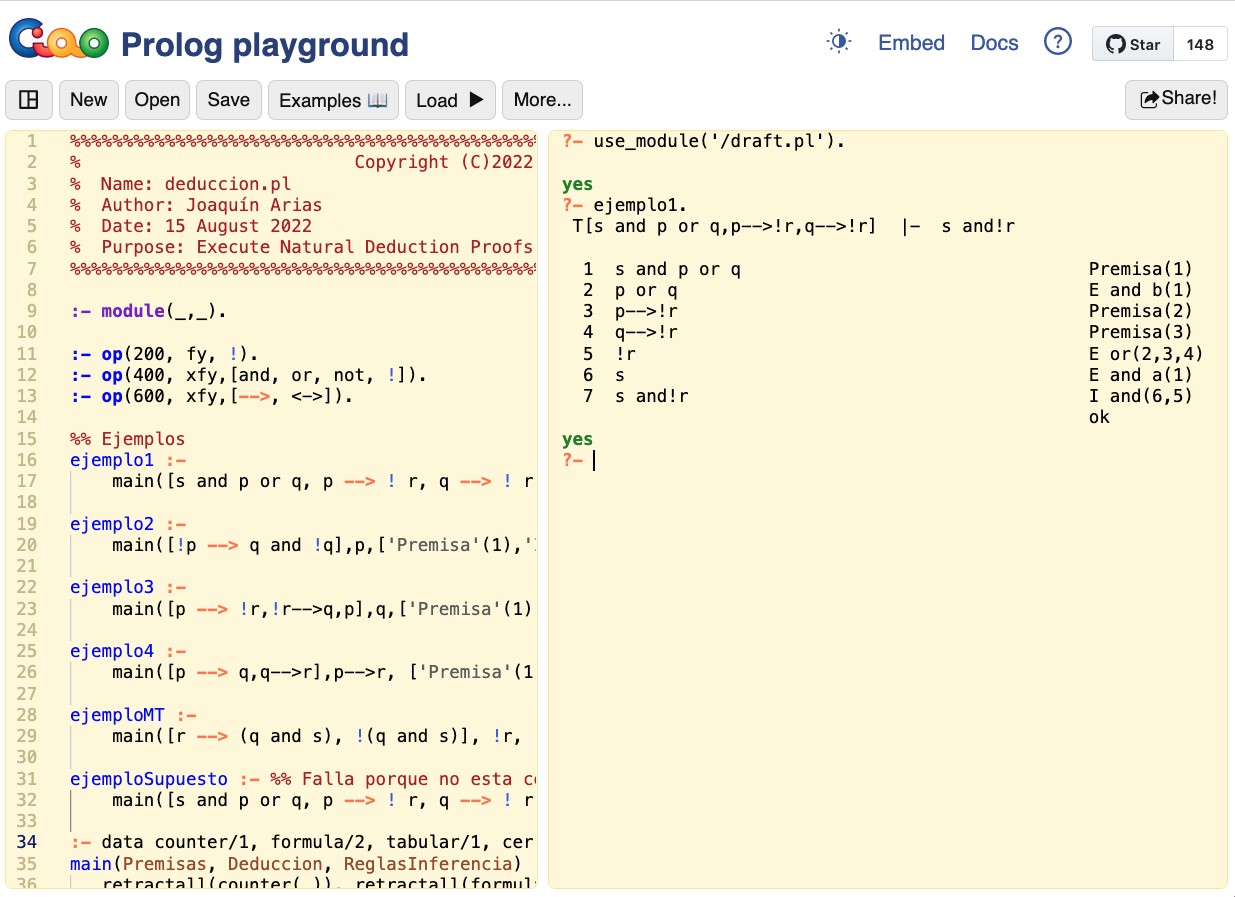
* + - * + Cada regla de inferencia es una **metaregla** con infinitas instancias.

**Deducción Natural: Reglas de inferencia: Conjunción**

**Deducción Natural: Reglas de inferencia: Disyunción**

**Deducción Natural: Ejemplo**

Prueba el ejemplo online

*T* [*s ^* (*p\_q*)*, p ! ¬r, q ! ¬r*] *`s^¬r*

1*. s^* (*p\_q*) *premisa*

2*. p\_q E^*(1)

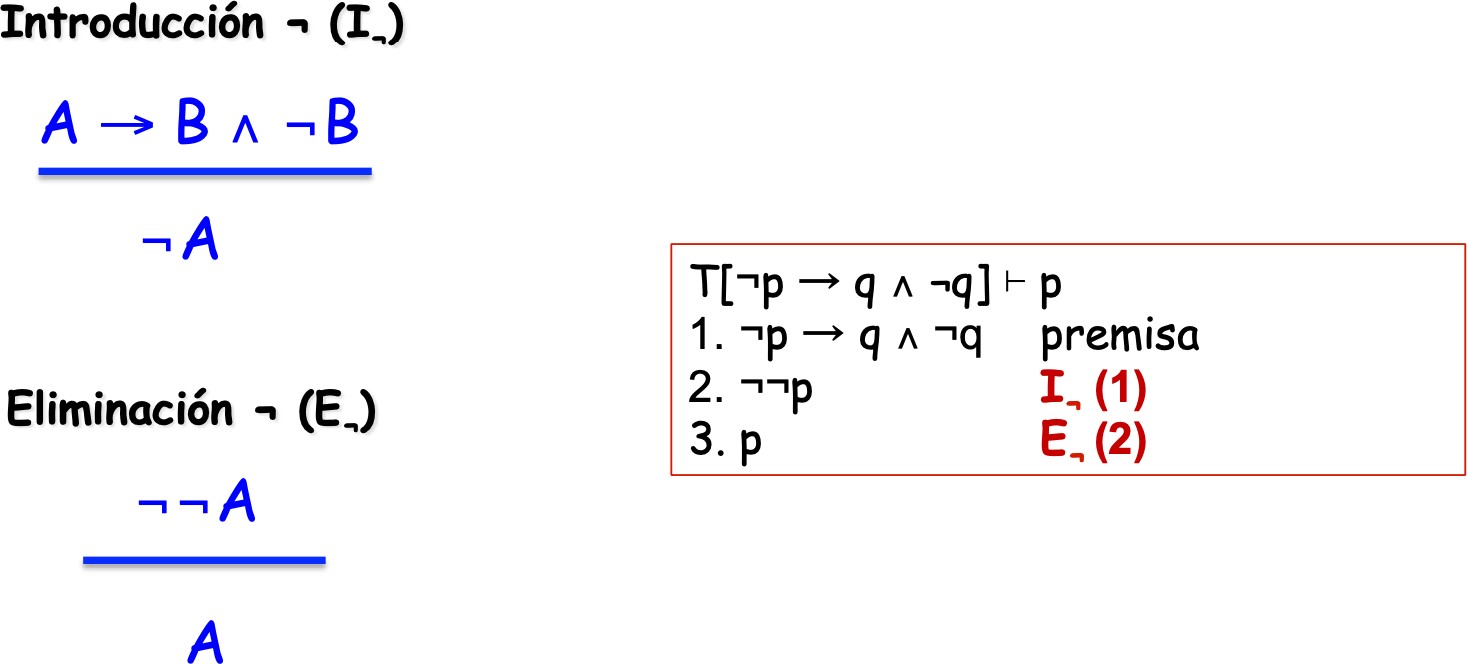
1. *p ! ¬r premisa*
2. *q ! ¬r premisa*

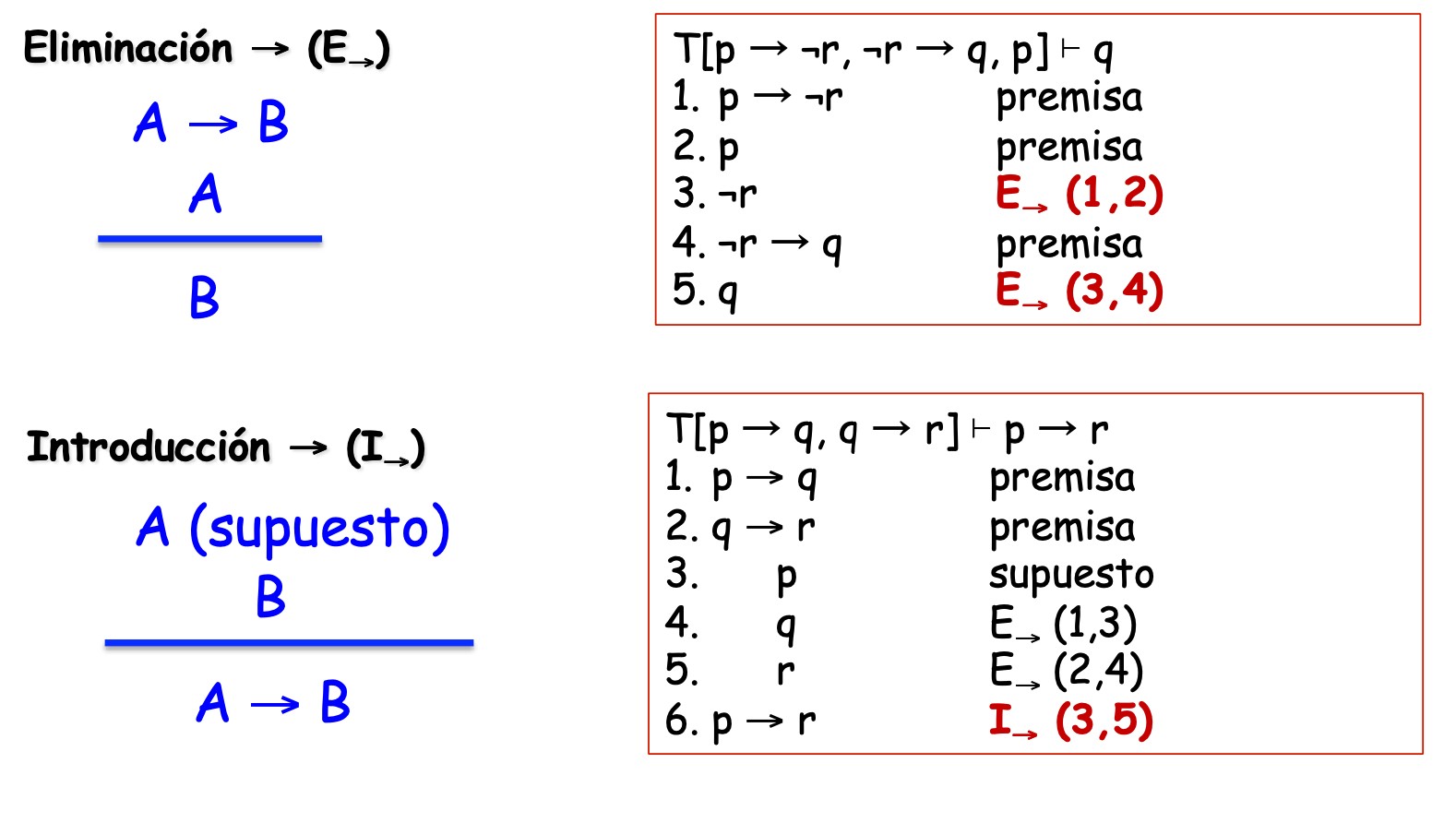
5*. ¬r E\_*(2*,* 3*,* 4)

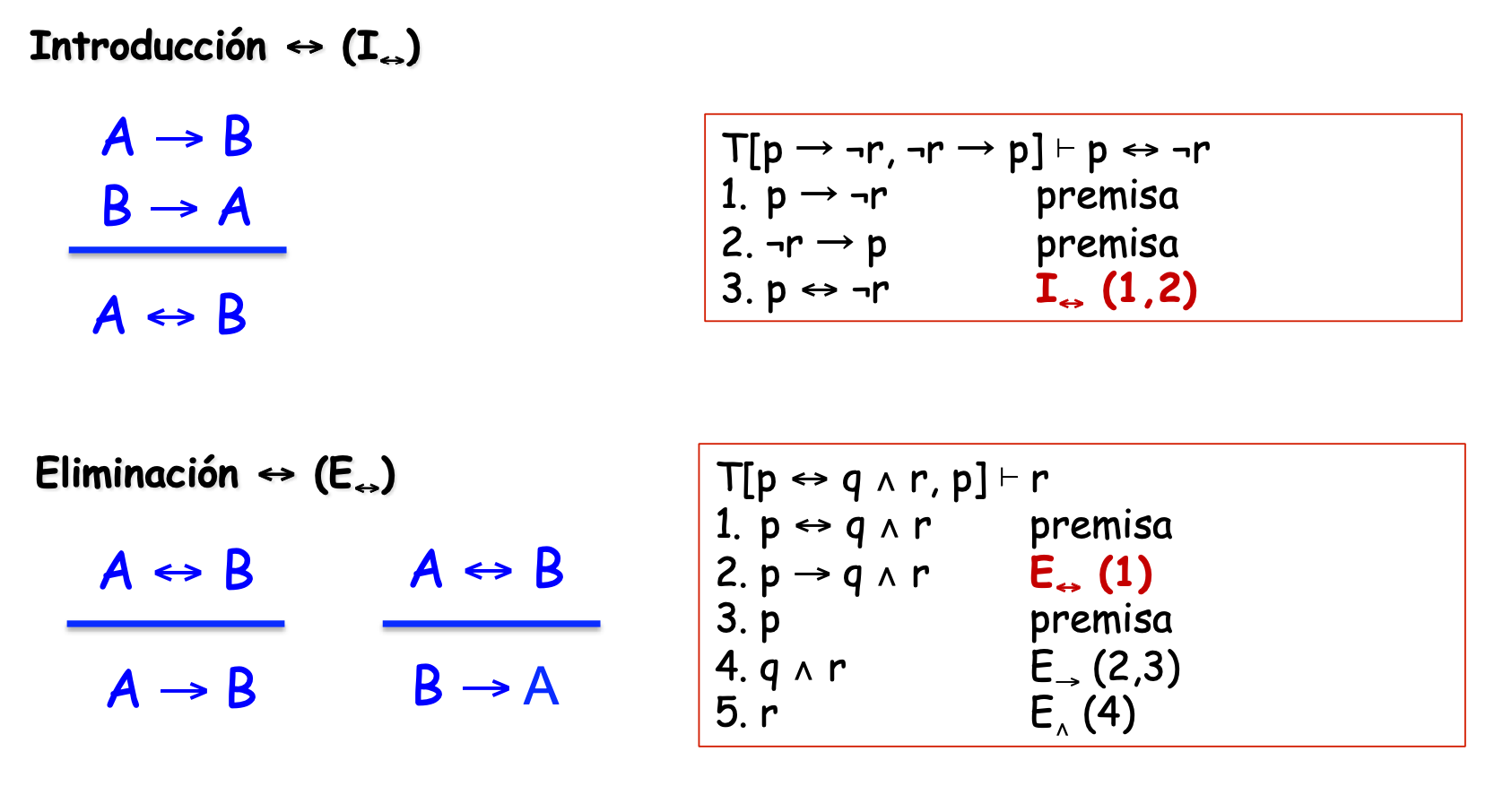
6*. s E^*(1)

7*. s^¬r I^*(5*,* 6)

**Deducción Natural: Reglas de inferencia: Negación**



**Deducción Natural: Reglas de inferencia: Implicación**

**Deducción Natural: Reglas de inferencia: Doble Implicación**

**Ejercicios: Deducción Natural**

Demostrar los siguientes esquemas argumentales:

* 1. *T* [*p*] *` q ! p*

2. *T* [*p !* (*q ! r*)] *`* (*p ! q*) *!* (*p ! r*)

3. *T* [*¬p ! ¬q*] *`* (*¬p ! q*) *! p*

4. *T* [*¬p ! ¬q*] *` q ! p*

5. *T* [*p ! q, q ! r*] *` p ! r*

6. *T* [*p !* (*q ! r*)*, q*] *` p ! r*

7. *T* [*p^q ! r, r ^s ! t*] *` p^q^s ! t*

8. *T* [*p^q ! r*] *` p^¬r ! ¬q*

9. *T* [*p\_q ! r, s ! p*] *` s ! r*

10. *T* [*p^q ! r, ¬*(*p\_r*) *! s, p ! q*] *` ¬s ! r*

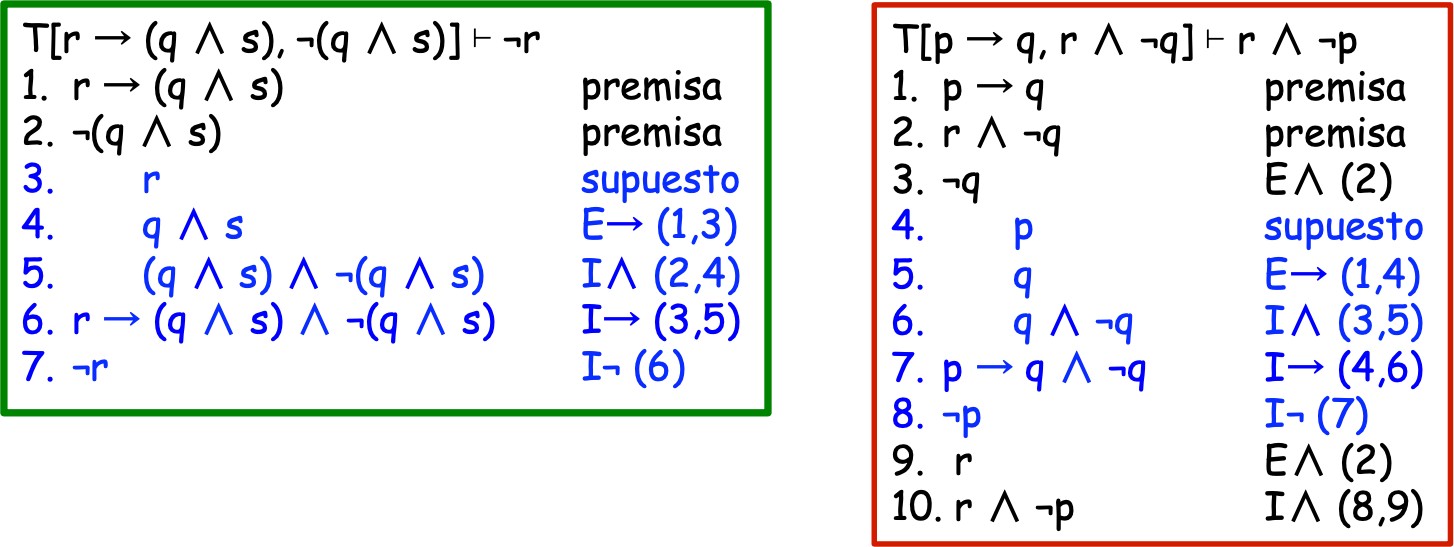
11. *T* [*p\_ p*] *` p*

12. *T* [*p*] *` ¬¬p*

1. *T* [*p\_q ! r*] *` q ! r*

#### Reglas de inferencia Derivadas

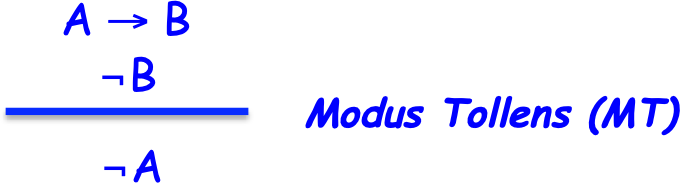
En distintas demostraciones se repiten con frecuencia ciertos pasos:



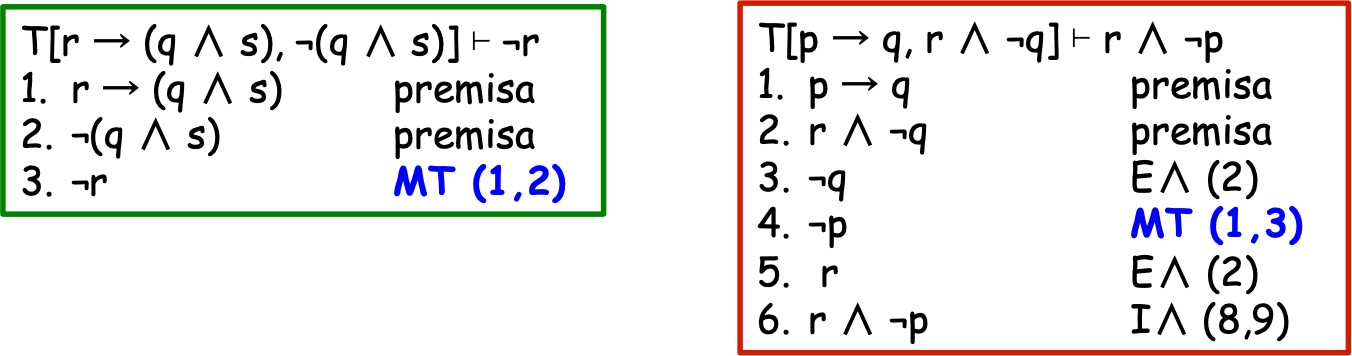
Aunque son distintas las fórmulas que aparecen en estos dos ejemplos, las líneas destacadas tienen una estructura común

Podríamos acortar las dos demostraciones si previamente demostramos con ca- rácter general que *T* [*A ! B, ¬B*] *`¬A*, para cualesquiera fórmulas *A* y *B*.

Las reglas derivadas se representan como las reglas básicas:



Las demostraciones anteriores quedarían ahora:



#### Reglas para la implicación

*T* [*A ! B, B ! C*] *` A ! C* Transitividad

*T* [*A ! B, ¬B*] *` ¬A* Modus Tollens

#### Reglas para la disyunción

*T* [(*A \_B*) *\_C*] *` A\_* (*B\_C*) Asociatividad

*T* [*A\_B*] *` B\_A* Conmutatividad

#### Reglas de Morgan

*T* [*¬*(*A^B*)] *` ¬A\_¬B* De Morgan

*T* [*¬*(*A\_B*)] *` ¬A^¬B* De Morgan

#### Reglas de corte

*T* [*A\_B, ¬A*] *` B* Corte

*T* [*A\_B, ¬B*] *` A* Corte

*T* [*A\_B, ¬A\_C*] *` B\_C* Corte

#### Teorema de intercambio

Sea:

* A una fórmula.
* B1 una sub-fórmula de A. Si tenemos:
* *`A*.
* *`B*1 *$ B*2. Entonces tenemos:
* *`A0*

Donde *A0* resulta de sustituir en *A* todas o alguna aparición de *B*1 por *B*2. Ejemplo:

*T* [*p $ r, q ! s, s ! t ^r*] *` q ! p^t*

* 1. *q ! s* premisa
  2. *s ! t ^r* premisa
  3. *q* supuesto

4. *s* E*!* (1,3)

5. *t ^r* E*!* (2,4)

1. *p $ r* premisa
2. *t ^ p* **Intercambio** (5,6)
3. *p^t* conmutatividad (7) 9. *q ! p^t* I*!* (3,8)

#### Ejercicios: Deducción Natural

Demostrar los siguientes formulas y/o esquemas argumentales:

1. *T ` p^q ! p*

2. *T ` p !* (*q ! p*)

3. *T `* (*p\_q*) *$* (*q\_ p*)

4. *T `* (*p ! q*) *^* (*q ! r*) *!* (*p ! r*)

5. *T ` ¬*(*p\_q*) *$ ¬p^¬q*

6. *T `* (*p ! ¬q*) *^¬*(*r ^¬p*) *!* (*q ! ¬r*)

7. *T `* (*p^q ! r*) *!* (*p !* (*q ! r*))

8. *T ` ¬*(*p ! q*) *$ p^¬q*

9. *T ` p !* (*q^¬q ! ¬p*)

10. *T `* (*p ! q*) *!* ((*p ! r*) *!* (*p ! q^r*))

Demostrar los siguientes formulas y/o esquemas argumentales:

* 1. *T* [*p ! q, p*] *` q*
  2. *T* [*p*] *` p\_q*
  3. *T* [*p\_ p*] *` p*

14. *T* [*p^* (*q\_r*)] *` p*

15. *T* [*p*] *` ¬¬p*

16. *T* [*p\_q ! r*] *` q ! r*

17. *T* [*p !* (*q ! r*)] *` q !* (*p ! r*)

18. *T* [*p ! q*] *` p\_r ! q\_r*

19. *T* [*¬q ! r, t ! ¬q, ¬s ! ¬q*] *` t \_¬s ! r*

20. *T* [*p\_* (*q ! r*) *! q, p*] *` q*

21. *T* [*¬p ! ¬s, ¬p\_ r, r ! ¬t*] *` ¬s\_¬t*

22. *T* [(*p ! q*) *^t,* (*r \_ p*) *^¬q, ¬t $ ¬s*] *` r ^s*



# Cápitulo 3

**Lógica Primer Orden**

* 1. Sintaxis, semántica y teoría interpretativa. [**39**](#_bookmark28)
     1. Sintaxis de la lógica de primer orden. [39](#_bookmark29)

Alfabeto [40](#_bookmark29)

Expresiones [41](#_bookmark29)

Ejercicios I: Formalización [42](#_bookmark29)

Variables [43](#_bookmark30)

Ejercicios II: Ligadura de variables [43](#_bookmark30)

Sustituciones [43](#_bookmark31)

Ejercicios III: Sustituciones [44](#_bookmark31)

Composición de Sustituciones [45](#_bookmark32)

Ejercicios IV: Composiciones [45](#_bookmark32)

Ejercicios V: Formalización de argumentos [46](#_bookmark33)

* + 1. Semántica de la lógica de primer orden [47](#_bookmark34)

Dominios interpretación [48](#_bookmark34)

Interpretación de Fórmulas [48](#_bookmark34)

Ejemplo Interpretación I [49](#_bookmark34)

Ejemplo Interpretación II [49](#_bookmark34)

Ejercicio [50](#_bookmark35)

* + 1. Satisfabilidad [51](#_bookmark36)

Formulas Abiertas [51](#_bookmark36)

Fórmulas Cerradas [51](#_bookmark37)

Modelos vs. Contramodelos [52](#_bookmark37)

* + 1. Teoría interpretativa de la lógica de primer orden. [52](#_bookmark38)

Validez y Consecuencia Lógica [52](#_bookmark38)

Ejemplo Validez I [53](#_bookmark38)

Ejemplo Validez II [53](#_bookmark39)

Ejemplo Consecuencia I y II [54](#_bookmark40)

Ejercicios Validez Argumentos [55](#_bookmark41)

* 1. Teorema de la demostración y deducción natural. [**56**](#_bookmark42)
     1. Decibilidad de la lógica de Primer Orden. [56](#_bookmark43)
     2. Deducción Natural [57](#_bookmark44)

Deducción Natural: Reglas de inferencia [57](#_bookmark44)

Ejemplos: Deducción Natural [58](#_bookmark44)

3.3 Tema 3.3: [**59**](#_bookmark45)

* + 1. Objetivo. [59](#_bookmark46)
    2. Forma Normal de Skolem [60](#_bookmark47)
       1. Forma Prenex [60](#_bookmark48)
       2. Cierre existencial [61](#_bookmark48)
       3. Forma normal conjuntiva [61](#_bookmark48)
       4. Eliminación de cuantificadores existenciales [61](#_bookmark48)
    3. Forma Clausular [62](#_bookmark49)

Forma Clausular: De una deducción [62](#_bookmark49)

* 1. Método de resolución de Robinson. [**62**](#_bookmark50)
     1. Introducción [62](#_bookmark51)
     2. Interpretaciones de Herbrand [63](#_bookmark52)

Universo de Herbrand [63](#_bookmark52)

Base de Herbrand [64](#_bookmark52)

Definición [64](#_bookmark52)

Ejemplos: Interpretaciones de Herbrand [65](#_bookmark52)

Satisfabilidad [65](#_bookmark52)

* + 1. Teorema de Herbrand [65](#_bookmark54)
    2. Método de Resolución de Robinson [66](#_bookmark55)

Algoritmo [66](#_bookmark55)

Ejemplo: Método de Resolución de Robinson [67](#_bookmark56)

* + 1. Resolución con Unificador de Máxima Generalidad [68](#_bookmark57)

Sustitución [68](#_bookmark57)

Composición sustituciones [68](#_bookmark57)

Unificadores y UMG [69](#_bookmark57)

Algoritmo de Unificación [69](#_bookmark57)

Definición [70](#_bookmark58)

Ejemplo: Resolución con UMG [71](#_bookmark58)

* + 1. Estrategias de resolución [71](#_bookmark59)

Motivación [71](#_bookmark59)

Ejemplos y propiedades [72](#_bookmark59)

Resolución lineal [72](#_bookmark59)

## Sintaxis, semántica y teoría interpretativa.

### Sintaxis de la lógica de primer orden.

Si este fin de semana nieva y Juan ha terminado el trabajo, irá a esquiar.

Nieva

Juan ha terminado el trabajo Juan irá a esquiar

Si este fin de semana nieva y Juan ha terminado el trabajo, irá a esquiar. Si este fin de semana nieva y Paco ha terminado el trabajo, irá a esquiar.

Nieva

Juan ha terminado el trabajo Paco ha terminado el trabajo Juan y Paco irán a esquiar

Si este fin de semana nieva, los que terminan su trabajo van a esquiar

Nieva

Juan ha terminado el trabajo Paco ha terminado el trabajo Nacho ha terminado el trabajo Juan, Paco y Nacho irán a esquiar

Cada vez que nieva, los que terminan su trabajo van a esquiar Nieva este fin de semana

Va a nevar el próximo fin de semana

Juan ha terminado el trabajo esta semana Paco ha terminado el trabajo esta semana Juan y Paco irán a esquiar este fin de semana y (si terminan su trabajo) también el siguiente

La deducción clásica por excelencia:

Si Sócrates es un hombre entonces Sócrates es mortal Sócrates es un hombre

Sócrates es mortal

Todos los hombres son mortales Sócrates es un hombre

Sócrates es mortal

Las proposiciones se refieren a individuos:

* Un empleado de los de la empresa.
* Un fin de semana del año.

Si tenemos que hablar demás de un individuo (para expresar un hecho más ge- neral), las proposiciones se nos quedan cortas.

Podríamos multiplicar las proposiciones, una para cada individuo y hecho lógico que se refiere a él, pero:

* Esto lo haría todo mucho mas largo y pesado...
* ... o imposible. En campos como las matemáticas se podría requerir hablar de conjuntos infinitos.

Necesitamos ampliar la capacidad expresiva de los lenguajes proposicionales con nuevos símbolos.

#### Alfabeto

Un alfabeto de un LPO se define con los siguientes tipos de símbolos:

* Símbolos de constante: *a, b, c,..., a*1*, a*2*,...*
* Símbolos de variable: *x, y, z,..., x*1*, x*2*,...*
* Símbolos de función: *f* (\_)*, g*(\_*,* \_)*,... o f* 1*, g*2*, o f /*1*, g/*2
* Símbolos de predicado: *P*(\_)*, Q*(\_)*,... o P*1*, Q*2*, o P/*1*, Q/*2
* Conectivas lógicas: *¬, \_, ^,!, $*
* Cuantificadores: *9, 8*

Un símbolo de predicado 0-ario es una proposición

En un LPO no puede haber símbolos comunes entre los conjuntos de variables y constantes, ni entre los símbolos de función y de predicado

Símbolos auxiliares de puntuación: paréntesis y comas

* Innecesarios si la aridad es parte del nombre del símbolo de predicado/fun- ción
* Mejoran la legibilidad: *P*(*a, f* (*x, g*(*b*))) en lugar de *P*2*af* 2*xg*1*b*

El uso de los paréntesis se puede reducir al mínimo mediante las convenciones de precedencia:

* + *{8, 9, ¬}* tienen mayor precedencia que *{^, \_}*
  + *{^, \_}* tienen mayor precedencia que *{!, $}*

#### Expresiones

Una expresión de un LPO es cualquier concatenación finita de símbolos de su alfabeto. Las expresiones relevantes son:

Términos:

* + Un símbolo de constante es un término.
  + Un símbolo de variable es un término.
  + Si *f* es un símbolo de función n-aria y *t*1*,...tn* son términos entonces

*f* (*t*1*,...tn*) es un término.

Fórmulas (Bien Formadas, FBF):

* + Si *P* es un símbolo de predicado n-ario y *t*1*,...tn* son términos entonces

*P*(*t*1*,...tn*) es una fórmula (átomo o fórmula atómica).

* + Si *F* y *G* son fórmulas entonces también lo son *¬F, F ^G, F \_G, F ! G* y

*F $ G*.

* + Si *F*(*x*) es una fórmula en la que *x* es una variable libre entonces *8xF*(*x*) y

*9xF*(*x*) son fórmulas.

Ejemplos de términos:

* + Constantes: *a, b, c,* 1*, spot, john*
  + Funciones: *f /*1*, g/*3*, h/*2*,* +*/*3
  + Variables: *x, y, m*

Correctas: *spot, f* ( *john*)*, f* (*x*)*,* +(1*,* 2*,* 3)*,* +(*x, y, m*)*, h*( *f* (*h*(1*,* 2))*, m*)

Incorrectas: *spot*(*x*)*,* +(1*,* 2)*, g, f* ( *f* (*h*))

Ejemplos de fórmulas:

* + Términos correctos...
  + Predicados: *dog/*1*, p/*2*, q/*0*, r/*0*, barks/*1

Correctas: *q, q ! r, barks*(*x*) *! dog*(*x*)*, p*(*x, y*)*, 9x*(*dog*(*x*) *^ barks*(*x*) *^*

*¬q*)*, 9y*(*dog*(*y*) *! barks*(*y*))*, dog*(*x*)*,*

Incorrectas: *q\_, 9p*

#### Ejercicios I: Formalización

Seleccionar los predicados, constantes y funciones necesarios para definir un LPO en el que formalizar las siguientes oraciones:

* + - 1. Vicente es mejicano.
      2. Mi casa es roja.
      3. Luisa y María son brasileñas pero Vicente es mejicano.
      4. Jorge adora a Juan.
      5. Jorge adora a su hermano Juan.
      6. Juan ama a Rosa pero ella no le corresponde.
      7. Pedro sujetó a Juan y María le atizó.
      8. Homero escribió la Ilíada y la Odisea.
      9. Nieves se peina a sí misma y también peina a Juan.
      10. Titán es satélite de Saturno pero Europa no lo es.

Seleccionar los predicados, constantes y funciones necesarios para definir un LPO en el que formalizar las siguientes oraciones:

* + - 1. O Pedro o María (pero no ambos) son hermanos míos.
      2. Si Colón descubrió América, merece un lugar en la Historia.
      3. El asesino de mi padre es Juan o Pedro, pero no Alberto.
      4. María ama a mi padre mientras que Julia me ama a mi.
      5. Cela leía a Borges aunque éste lo detestaba públicamente.
      6. María está enamorada de alguien.
      7. Hay al menos un número primo.
      8. Algunas cantantes de ópera no están gordas.
      9. Cualquier crimen será castigado.
      10. No todos los crímenes merecen la pena capital.

Seleccionar los predicados, constantes y funciones necesarios para definir un LPO en el que formalizar las siguientes oraciones:

* + - 1. Las novelas de Cela me fascinan.
      2. Hay profesores que no saben explicar.
      3. Sólo los suecos entienden a Bergman.
      4. Todo ciudadano tiene derecho a una vivienda.
      5. Hay genios, pero no todos los poetas lo son.
      6. No todos los satélites de Júpiter tienen atmósfera.
      7. Todos los estudiantes de tercer curso ayudan a al menos uno de primero.
      8. Los caballeros las prefieren rubias pero se casan con las morenas.
      9. Nadie respeta a quien no se respeta a sí mismo.
      10. Hay un pintor a quien todo el mundo admira.

#### Variables

Ámbito (alcance) de un cuantificador en una fórmula:

* La subfórmula inmediatamente a su derecha (puede ser necesario el uso de paréntesis para indicar el alcance del cuantificador):

*9xP*(**x***, y*) *\_Q*(*x, y*); *9x*(*P*(**x***, y*) *\_Q*(**x***, y*));

*8y9xP*(**x,y**) *\_Q*(*x, y*); *8y9x*(*P*(**x,y**) *\_Q*(**x,y**))

Variables libres y ligadas en fórmulas de un LPO. Una variable x:

* Se encuentra ligada en una fórmula A cuando está en el ámbito de un cuan- tificador *9x* o *8x* en *A*.
* Se encuentra libre en una fórmula cuando no se encuentra ligada.

NOTA: Un mismo símbolo de variable puede estar libre y ligado en una misma fórmula.

Fórmulas abiertas y cerradas en un LPO:

* Es abierta cuando contiene al menos una variable libre.
* Es cerrada cuando todas sus variables están ligadas.

#### Ejercicios II: Ligadura de variables

Señalar las variables libres y ligadas en las siguientes fórmulas: 1. *9x*(*P*(*x, f* (*y*)) *! 9yQ*(*x, y*))

2. *9xP*(*x*) *! 8yQ*(*x, f* (*y*))

3. *9x9y*(*P*(*x, y*) *\_Q*(*x, y*)) *^R*(*a, y*) 4. *9x9y*((*P*(*x, y*) *\_Q*(*x, y*)) *^R*(*x, y*)) 5. *8x*(*x* = *y ! 9zP*(*x, z*))

6. *9x8yP*(*x, f* (*x, y*)) *! 9yQ*(*x, y*)

1. *x* = *y* + *z ! x y* + *z*

8. *8x*(*x* + 0 = *x*)

9. *8x*(*N*(*x*) *! N*(*s*(*x*)))

10. *8x9y*(*P*(*g*(*x, a*)*, y*) *\_Q*(*x*) *\_R*(*z, b*)) *^ 9zS*(*x, y, z*)

#### Sustituciones

Una **sustitución** es una función finita de un conjunto de variables de un lenguaje en el de términos. Se representa como *{x*1*/t*1*, x*2*/t*2*,..., xn/tn}* donde *x*1*,..., xn*

son variables diferentes y *t*1*,...,tn* son términos.

* Cada par *xi/ti* se denomina **ligadura**.
* Una sustitución que no sustituye ninguna variable se llama **sustitución va- cía** (0/ )

Dada una fórmula *A* y una sustitución *a* = *x*1*/t*1*,..., xn/tn* , se denomina **apli- cación de** *a* **a** *A* **(***Aa***)** a la fórmula obtenida reemplazando simultáneamente cada ocurrencia en *A* de *xi* por *ti*, para cada *xi/ti 2 a*.

*{ }*

* *a* = *{x/ f* (*a*)*, y/x, z/h*(*b, y*)*, w/a}*
* *P*(*x, y, z*)*a* = *P*( *f* (*a*)*, f* (*a*)*, h*(*b, f* (*a*))) incorrecto
* *P*(*x, y, z*)*a* = *P*( *f* (*a*)*, x, h*(*b, y*)) correcto

Notación: Siendo *A* una fórmula y *x* una variable de un LPO:

* *A*(*x*) indica la aparición de al menos una ocurrencia **libre** de *x* en *A*
* *A x/t* representa la fórmula obtenida a partir de *A* sustituyendo **todas** las apariciones de la **variable libre** *x* por el término *t*.

*{ }*

Ejemplos:

*A*(*x*) : *P*(**x***, f* (*y*)) *! 9yQ*(**x***, y*) *A{x/a}* : *P*(**a***, f* (*y*)) *! 9yQ*(**a***, y*)

*A*(*y*) : *9x*((*P*(*x,* **y**) *\_Q*(*x,* **y**)) *^R*(*x,* **y**))

*A{y/ f* (*z*)*}* : *9x*((*P*(*x,* **f(z)**) *\_Q*(*x,* **f(z)**)) *^R*(*x,* **f(z)**))

Condiciones para la aplicación de una sustitución a una fórmula *A*:

* Se aplica **única y exclusivamente** sobre variables **libres** presentes en *A*. De no haberlas, la sustitución no tiene ningún efecto sobre la expresión inicial.
* Su aplicación reemplaza **todas y sólo** las ocurrencias de la variable libre en la fórmula por el término. Errores comunes: (*8x*(*P*(*x, f* (*y*)) *! 9yQ*(*x, y*)))*{y/a}* : *9x*(*P*(*x, f* (*a*)) *! 9yQ*(*x, y*))

(*9x*(*P*(*x, f* (*y*)) *! 9yQ*(*x, y*)))*{y/a}* : *9x*(*P*(*x, f* (*a*)) *! Q*(*x, a*))

(*9x*(*P*(*x, f* (*y*)) *^Q*(*x, y*)))*{y/a}* : *9x*(*P*(*x, f* (*a*)) *! Q*(*x, y*))

* Una sustitución *a* **no** se puede aplicar a *A* si pasa lo siguiente:
  1. *a* contiene una ligadura *x/t* y *t* contiene la variable *y*.
  2. hay una ocurrencia de *x* en *A* que se puede reemplazar por *t*
  3. dicha ocurrencia está dentro del ámbito de un cuantificador sobre *y*.

*9y*(*8zP*(*x, z*) *^Q*(*y*))*{x/ f* (*y*)*}* : *9y*(*8zP*( *f* (*y*)*, z*) *^Q*(*y*))

#### Ejercicios III: Sustituciones

Realizar las siguientes sustituciones:

1. (*9x*(*P*(*x, f* (*y*)) *! 9yQ*(*x, y*)))*{y/g*(*z*)*}*

2. (*8x8y*(*P*(*x, y*) *! Q*(*x, y*)))*{y/a}*

3. (*8x*(*8yP*(*x, y*) *! Q*(*x, y*)))*{y/a}*

4. (*9x*(*8y*(*P*(*x, y*) *\_Q*(*x, y*)) *^R*(*x, y*))*{y/b}*

5. (*9x*(*8y*(*P*(*x, y*) *\_Q*(*x, y*)) *^R*(*x, y*))*{x/b}*

6. (*9x8y*(*P*(*x, y*) *\_Q*(*x, y*)) *^R*(*x, y*))*{x/a, y/b}*

7. (*8x*(*P*(*x, y*) *! Q*(*x, y*)))*{y/ f* (*x, a*)*}*

8. (*8yP*(*x, y*) *! 8xQ*(*x, y*))*{y/ f* (*x, a*)*}*

9. (*x* = *y* + *z ! x y* + *z*)*{x/*1*, y/*2*}*

10. (*x* = *y* + *z ! x y* + *z*)*{x/s*(*x*)*}*

11. (*x* = *y* + *z ! x y* + *z*)*{x/s*(*y*)*}*

12. (*8x*(*x* + 0 = *x*))*{x/*1*}*

#### Composición de Sustituciones

Dadas dos sustituciones *a* = *x*1*/t*1*,..., xn/tn* y *b* = *y*1*/s*1*,..., ym/sm*

*{ } { }*

su **composición** *ab* se define eliminando del conjunto

*{x*1*/t*1*b,..., xn/tnb, y*1*/s*1*,..., ym/sm}*

* + las ligaduras *xi/tib* tales que *xi ⌘ tib* ,
  + y las ligaduras *yi/si* tales que *yi 2 {x*1*,..., xn}*

Ejemplo:

* + si *a* = *{x/*3*, y/ f* (*x,* 1)*}* y *b* = *{x/*4*}* entonces

*ab* = *{x/*3*, y/ f* (4*,* 1)*}* y *ba* = *{x/*4*, y/ f* (*x,* 1)*}*

Propiedades de la composición:

* + (*Fa*)*b* = *F*(*ab* )
  + (*ab* )*g* = *a*(*bg*)
  + *al* = *la* = *a*
  + *ab 6*= *ba*

#### Ejercicios IV: Composiciones

Realizar la composición de las siguientes sustituciones: 1. *{x/a, y/g*(*z*)*, z/ f* (*x*)*} {x/b, z/y}*

2. *{x/b, z/y} {x/a, y/g*(*z*)*, z/ f* (*x*)*}*

3. *{x/ f* (*a, y*)*, y/b} {x/a, y/c}*

4. *{x/a, y/c} {x/ f* (*a, y*)*, y/b}*

5. *{x/y, y/z, z/w} {x/a, y/b, z/c, w/d}*

6. *{x/a, y/b, z/c, w/d} {x/y, y/z, z/w}*

7. *{x/ f* (*z*)*, y/w} {z/a, w/ f* (*y, z*)*}*

8. *{z/a, w/ f* (*y, z*)*} {x/ f* (*z*)*, y/w}*

#### Ejercicios V: Formalización de argumentos

Definir LPOs en los que formalizar los siguientes argumentos:

1. La Tierra orbita en torno al Sol. La Luna orbita en torno a la Tierra. Todo cuerpo que orbita en torno al Sol es un planeta. Son satélites los cuerpos que orbita en torno a planetas. Luego la Tierra es un planeta y la Luna un satélite.
2. Cero es un número natural. El sucesor de un número natural es también un número natural. Uno es sucesor de cero. Luego uno es un número natural.
3. Cero es un número natural. El sucesor de un número natural es también un número natural. La suma de cualquier número natural y cero es igual a ese mismo número. La suma de un número y el sucesor de otro es igual al sucesor de la suma del primero más el antecesor del segundo.

Definir LPOs en los que formalizar los siguientes argumentos:

1. Aquel que no existe no puede engañarse. Yo me engaño. Luego yo existo. (A. Deaño)
2. Solamente las personas bien educadas están suscritas al Times. Ningún puercoespín sabe leer. Las personas bien educadas saben leer. Luego nin- gún puercoespín está suscrito al Times. (L. Carroll)
3. Todos los filósofos se han preguntado qué es la Filosofía. Todos los que se han preguntado qué es la Filosofía han dado en la locura. Nietzsche es un filósofo. El Padre Ceballos no acabó loco. Luego Nietzsche y el Padre Ceballos no son la misma persona. (A. Deaño)

Definir LPOs en los que formalizar los siguientes argumentos:

1. Todos los libros de texto son tediosos Algunos libros de texto están llenos de ejercicios. Luego algunos libros llenos de ejercicios son tediosos
2. Todos los políticos tienen algo que ocultar Algunos políticos salen en tele- visión. Luego hay quienes tienen algo que ocultar y salen en televisión
3. Todos los chimpancés son primates Sara es un chimpancé. Luego Sara es un primate
4. Algunos primates no son chimpancés Algunos chimpancés saben hablar. Luego algunos primates no saben hablar
5. Ningún individuo que no sea chimpancé es un primate Ninguno que no sea primate es inteligente Sara no es un chimpancé. Luego Sara no es inteli- gente

Definir LPOs en los que formalizar los siguientes argumentos:

1. Hay individuos inteligentes o que saben hablar. Juan no sabe hablar. Luego

Juan no es inteligente.

1. Todo elemento químico es oxidante o reductor. El carbono es un elemento químico no oxidante. Luego el carbono es reductor.
2. No todos los seres humanos saben hablar o son inteligentes. Sara es un ser humano pero no sabe hablar. En consecuencia, Sara es inteligente.
3. Todos los chimpancés saben hablar. Algunos primates no saben hablar. Al- gunos primates son humanos. Por tanto, algunos seres humanos son chim- pancés y saben hablar.

Definir LPOs en los que formalizar los siguientes argumentos:

1. Todos los chimpancés son primates. Algunos seres humanos son inteligen- tes. Algunos primates son seres humanos. Juan es un chimpancé y Sara es un ser humano que sabe hablar. Así pues , Juan es un primate y Sara es inteligente.
2. Todos los rinocerontes tienen un cuerno. Todos y sólo los rinocerontes son dignos de ser cazados. Luego todos los animales dignos de ser cazados tienen un cuerno.
3. Todas las selvas tropicales tienen color verde. Nada que tenga color verde esta seco. Por tanto, ninguna selva tropical esta seca.
4. Ningún fotógrafo es pintor. Todo aquel que no es fotógrafo es buen dibu- jante. Así pues, todos los pintores son buenos dibujantes.

### Semántica de la lógica de primer orden

Dado un LPO, definir de modo preciso el significado de sus fórmulas. Conceptos semánticos empleados en semántica formal:

* **Dominio** de interpretación **D**: conjunto no vacío de objetos
  + Relaciones n-arias: subconjuntos de Dominio*n*
  + Funciones n-arias: n-tuplas de obj. del dominio *!* obj. del dominio
* **Función** de interpretación *i*():
  + Fórmulas *! {V, F}*
  + Términos *!* objetos del dominio
  + Predicados y funciones *!* relaciones y funciones en obj. del dominio
* **Interpretación** I: *hD, i*()*i*
  + Un dominio no vacío de individuos, *D*

Una función *i*() de individuos de *D*, funciones y relaciones sobre *D* a todas las constantes, funciones y predicados del LPO.

*o*

#### Dominios interpretación

Las expresiones de un lenguaje sólo significan algo cuando se refieren o hablan de algo. Esto es el dominio o universo de discurso.

Punto de partida: elección de un dominio no vacío de interpretación:

* *D* = *{Sol, Tierra, Luna}*; *D* = *{*1*,* 2*,* 3*,* 4*,* 5*,...}*; *D* = *{*⌥*, Q,* ⇤*}*

A continuación, se define la función de interpretación para el lenguaje L:

* Para toda constante *a 2 L* : *i*(*a*)= *d, d 2 D*
  + Todo individuo de D debe tener un nombre (distinto) en *L*

Si *L* no tuviera suficientes constantes, se amplía con las constantes necesarias y se denomina *L*(*D*)

*o*

* Para toda función n-aria *f 2 L* : *i*( *f* )= *fD*
  + *fD* : *hd*1*,..., dni! d* (aplicación de *Dn* en *D*)
* Para todo predicado n-ario *P 2 L* : *i*(*P*)= *PD*
  + *PD* : *hd*1*,..., dni! {V, F}* (aplicación de *Dn* en *{V, F}*)

#### Interpretación de Fórmulas

Dada una interpretación *I* = *hD, i*()*i* para un lenguaje de primer orden *L*:

* Asignar significado a las fórmulas de *L* implica:
  + Asignar significado a las fórmulas atómicas
  + Asignar significado al resto de fórmulas mediante inducción
* Asignación de valor de verdad a las fórmulas atómicas de *L*:

*i*(*P*(*t*1*,...,tn*)) = *V/F* sii *PD*(*i*(*t*1)*,..., i*(*tn*)) = *V/F*

Cada interpretación concreta asignará un y sólo un valor de verdad a cada fórmula atómica de L. Dos interpretaciones sobre un mismo do- minio y para un mismo lenguaje difieren entre sí en el valor de verdad que asignan.

*o*

En el caso del predicado =, su semántica es fija, sin variación entre interpretaciones:

*o*

*i*(*t*1 = *t*2)= *V/F* sii *i*(*t*1) es idéntico a / no es idéntico a *i*(*t*2)

Asignación de valor de verdad a fórmulas:

*i*(*¬A*)= *V* sii *i*(*A*)= *F i*(*A^B*)= *V* sii *i*(*A*)= *i*(*B*)= *V i*(*A\_B*)= *F* sii *i*(*A*)= *i*(*B*)= *F*

*i*(*A ! B*)= *F* sii *i*(*A*)= *V* y *i*(*B*)= *F*

*i*(*A $ B*)= *V* sii *i*(*A*)= *i*(*B*)

*i*(*9xA*)= *V* sii *i*(*Ax/a*)= *V* para al menos una constante a de *L*(*D*)

*i*(*8xA*)= *V* sii *i*(*Ax/a*)= *V* para toda constante a de *L*(*D*)

#### Ejemplo Interpretación I

**Ejemplo I:** *8x*(*M*(*a, x*) *^P*(*x*)) *! ¬9yQ*(*y*)

Construimos una interpretación sobre el dominio de los números naturales:

*D* = *{*0*,* 1*,* 2*,* 3*,* 4*,...}*

El lenguaje de la fórmula sólo tiene un símbolo de constante (*a*), por lo que ampliamos el conjunto de constantes y definimos *L*(*D*):

*Ctes.* = *{a, a*1*, a*2*, a*3*, a*4*,...}*

La función de interpretación i podría ser:

* *i*(*a*)= 0*, i*(*a*1)= 1*, i*(*a*2)= 2*, i*(*a*3)= 3*,...*
* *PD*(*x*): *x* es par; *QD*(*x*): *x* es impar; *MD*(*x, y*): *x < y*

Bajo esta interpretación: *i*(*8x*(*M*(*a, x*) *^P*(*x*)) *! ¬9yQ*(*y*)) = *V*

*i*(*8x*(*M*(*a, x*) *^P*(*x*))) = *F* porque:

* + *i*((*M*(*a, x*) *^P*(*x*))*{x/a}*)= *i*(*M*(*a, a*) *^P*(*a*)) =

*MD*(*i*(*a*)*, i*(*a*)) *^ PD*(*i*(*a*)) = *MD*(0*,* 0) *^ PD*(0)= *F*

*i*(*¬9yQ*(*y*)) = *F* porque:

* + *i*(*9yQ*(*y*)) = *V* porque:

*o i*(*Q*(*y*)*{y/a*1*}*)= *i*(*Q*(*a*1)) = *QD*(*i*(*a*1)) = *QD*(1)= *V*

#### Ejemplo Interpretación II

**Ejemplo II:** *8x*(*M*(*a, x*) *^P*(*x*)) *! ¬9yQ*(*y*)

Construimos una interpretación sobre un dominio finito:

*D* = *{Q,* ⇤*,* ⌥*}*

El lenguaje de la fórmula sólo tiene un símbolo de constante (a), por lo que ampliamos el conjunto de constantes y definimos L(D):

*Ctes.* = *{a, b, c}*

La función de interpretación i podría ser:

* i(a) = *Q*, i(b) =⇤, i(c) =⌥

|  |  |
| --- | --- |
| *PD* |  |
| *Q* | V |
| ⇤ | V |
| ⌥ | V |

|  |  |
| --- | --- |
| *QD* |  |
| *Q* | V |
| ⇤ | F |
| ⌥ | V |

|  |  |  |  |
| --- | --- | --- | --- |
| *MD* |  |  |  |
| *Q* | V | V | V |
| ⇤ | F | F | V |
| ⌥ | V | F | F |

Bajo esta interpretación:

*i*(*8x*(*M*(*a, x*) *^P*(*x*)) *! ¬9yQ*(*y*)) = *F*

*i*(*8x*(*M*(*a, x*) *^P*(*x*))) = *V* porque:

1. *i*((*M*(*a, x*) *^P*(*x*))*{x/a}*)= *MD*(*i*(*a*)*, i*(*a*)) *^ PD*(*i*(*a*)) =

*MD*(*Q, Q*) *^ PD*(*Q*)= *V*

2. *i*((*M*(*a, x*) *^P*(*x*))*{x/b}*)= *MD*(*i*(*a*)*, i*(*b*)) *^ PD*(*i*(*b*)) =

*MD*(*Q,* ⇤) *^ PD*(⇤)= *V*

3. *i*((*M*(*a, x*) *^P*(*x*))*{x/c}*)= *MD*(*i*(*a*)*, i*(*c*)) *^ PD*(*i*(*c*)) =

*MD*(*Q,* ⌥) *^ PD*(⌥)= *V*

*i*(*¬9yQ*(*y*)) = *F* porque:

* *i*(*9yQ*(*y*)) = *V* porque:

*o i*(*Q*(*y*)*{y/a}*)= *i*(*Q*(*a*)) = *QD*(*i*(*a*)) = *QD*(*Q*)= *V*

#### Ejercicio

**Ejercicio: Aritmética de Peano**

1. Dadas las fórmulas:

*{N*(*a*)*, 8x*(*N*(*x*) *! N*(*s*(*x*)))*, 8x*(*N*(*x*) *! x* + *a* = *x*)*,*

*8x8y*(*N*(*x*) *^N*(*y*) *! x* + *s*(*y*)= *s*(*x* + *y*)*}*

interpretarlas en el dominio de los números naturales (con el 0), siendo *N*(\_) la propiedad de ser un número natural, = (\_*,* \_) la relación de identidad y *s*(\_) y

+(\_*,* \_) las funciones sucesor y suma, respectivamente.

### Satisfabilidad

#### Formulas Abiertas

Hemos visto como establecer claramente el significado de una fórmula cuando ésta es cerrada, pero ¿cómo interpretar una fórmula abierta?

Ejemplos de fórmulas abiertas:

* + - 1. Alguien es el rey de Canadá: *R*(*x, a*)
      2. Algo es igual a dos: *x* = 2
      3. Alguien va a morir: *M*(*x*)

¿Que significado tienen estas fórmulas abiertas?

1. *R*(*x, a*) es **falsa** pues ninguna sustitución de x produce una afirmación ver- dadera: ninguna persona es rey de Canadá.
2. *x* = 2 es una fórmula que **puede ser verdadera** si sustituimos *x* por el número 2, mientras que será falsa si elegimos otro número.
3. *M*(*x*) es una fórmula que **siempre es verdadera** para cualquier nombre de persona que sustituya a *x*.

Sea *A* una fórmula abierta de *L* (LPO):

* *A* es **satisfacible** cuando puede ser verdadera:

Hay una interpretación *D, i*() y una sustitución *q* = *x*1*/c*1*,..., xn/cn* de todas sus variables libres por constantes de *L*(*D*) que la hacen verdadera

*{ }*

*o h i*

(existe una *hD, i*()*i* y una *q* tal que *i*(*Aq* )= *V* )

* *A* es **verdadera** en una interpretación *hD, i*()*i* cuando:

Toda sustitución *q* de sus variables libres por constantes de *L*(*D*) la hacen verdadera (*i*(*Aq* )= *V* para toda *q* )

*o*

* *A* es **válida** cuando es verdadera en toda interpretación

*o i*(*Aq* )= *V* para toda *hD, i*()*i* y toda *q*

#### Fórmulas Cerradas

Dada una fórmula abierta *A*(*x*1*,..., xn*), definimos:

* **Cierre existencial** de *A*(*x*1*,..., xn*): *9x*1*,..., xnA*
* **Cierre universal** de *A*(*x*1*,..., xn*): *8x*1*,..., xnA*

Relaciones semánticas entre fórmulas abiertas y cerradas:

*A*(*x*1*,..., xn*) es satisfacible sii *9x*1*,..., xnA* es satisfacible.

*A*(*x*1*,..., xn*) es insatisfacible sii *9x*1*,..., xnA* es insatisfacible. *A*(*x*1*,..., xn*) es verdadera en *I* sii *8x*1*,..., xnA* es verdadera en *I*. *A*(*x*1*,..., xn*) es válida sii *8x*1*,..., xnA* es válida.

Satisfabilidad de una fórmula:

* Una interpretación *hD, i*()*i* **satisface** una fórmula *A* sii *i*(*A*)= *V*
* Una fórmula *A* es **satisfacible** sii existe al menos una interpretación *D, i*()

*h i*

tal que *i*(*A*)= *V*

* Una fórmula *A* es **insatisfacible** sii no existe ninguna interpretación *D, i*()

*h i*

tal que *i*(*A*)= *V*

Extensión a conjuntos de fórmulas *{A*1*,..., An}*:

* Una interpretación *D, i*() satisface *A*1*,..., An* sii *i*(*Ai*)= *V* para todo

*h i { }*

*i* :1 *i n*

* *A*1*,..., An* es satisfacible sii existe al menos una interpretación *D, i*()

*{ } h i*

tal que *i*(*Ai*)= *V* para todo *i* :1 *i n*

* *A*1*,..., An* es insatisfacible sii no existe ninguna interpretación *D, i*()

*{ } h i*

tal que *i*(*Ai*)= *V* para algún *i* :1 *i n*

Una fórmula *A* es **válida** sii *i*(*A*)= *V* para toda interpretación *hD, i*()*i*

#### Modelos vs. Contramodelos

**Modelo**: una interpretación *I* es modelo de una fórmula *A* sii

* *i*(*A*)= *V* cuando *A* es una fórmula cerrada
* *i*(*Aq* )= *V* para toda sustitución *q* de todas sus variables libres, cuando *A*

es una fórmula abierta

**Contramodelo**: una interpretación *I* es contramodelo de *A* sii

* *i*(*A*)= *F* cuando *A* es una fórmula cerrada
* *i*(*Aq* )= *F* para alguna sustitución *q* de todas sus variables libres, cuando

*A* es una fórmula abierta

### Teoría interpretativa de la lógica de primer orden.

#### Validez y Consecuencia Lógica

**Validez lógica**

* Una fórmula *A 2 L* es válida (lógicamente válida) sii

es verdadera en toda interpretación: ✏ *A*

#### Consecuencia lógica

* + Dado un conjunto de fórmulas G = *{A*1*,..., An}* (*Ai 2 L*) y una fórmula

*B 2 L*, *B* es consecuencia lógica de G (G ✏ *B*) sii:

*o* Todo modelo de G es también modelo de *B*

(toda interpretación que haga verdad a G también hace verdad a *B*) No existe ninguna interpretación que haga verdad a G y que no haga verdad a *B*

*o*

#### Ejemplo Validez I

**Ejemplo Validez I:** ✏ *9x8yP*(*x, y*) *! 8y9xP*(*x, y*)

Considerando *D* = *{*1*,* 2*}*, *L*(*D*)= *{P/*2*}[{a, b}* La fórmula **es válida**.

No hay forma de encontar contramodelos. Justificación:

* + - 1. *i*(*9x8yP*(*x, y*) *! 8y9xP*(*x, y*)) = *F* sii *i*(*9x8yP*(*x, y*)) = *V* sii
         * *i*(*8yP*(*a, y*)) = *V* sii *i*(*P*(*a, a*)) = *V* y *i*(*P*(*a, b*)) = *V*
         * o bien *i*(*8yP*(*b, y*)) = *V* sii *i*(*P*(*b, a*)) = *V* y *i*(*P*(*b, b*)) = *V*
      2. y *i*(*8y9xP*(*x, y*)) = *F* sii
         * *i*(*9xP*(*x, a*)) = *F* sii *i*(*P*(*a, a*)) = *F* y *i*(*P*(*b, a*)) = *F*
         * o bien *i*(*9xP*(*x, b*)) = *F* sii *i*(*P*(*a, b*)) = *F* y *i*(*P*(*b, b*)) = *F*

La imposibilidad de encontrar contramodelos NO es exclusiva de esta interpre- tación. Verificar el antecedente es incompatible con hacer falso el consecuente.

#### Ejemplo Validez II

**Ejemplo Validez II:** ✏ *8y9xP*(*x, y*) *! 9x8yP*(*x, y*)

Considerando *D* = *{*1*,* 2*}*, *L*(*D*)= *{P/*2*}[{a, b}* La fórmula **NO es válida**.

Porque la siguiente interpretación es un contramodelo:

* i(a) = 1
* i(b) = 2
* *PD*(1*,* 1)= *V* , *PD*(1*,* 2)= *F*, *PD*(2*,* 1)= *F*, *PD*(2*,* 2)= *V*

Justificación:

1. *i*(*8y9xP*(*x, y*)) = *V* porque:
   1. *i*(*P*(*a, a*)) = *V* porque *PD*(*i*(*a*)*, i*(*a*)) = *PD*(1*,* 1)= *V*
   2. y *i*(*P*(*b, b*)) = *V* porque *PD*(*i*(*b*)*, i*(*b*)) = *PD*(2*,* 2)= *V*
2. *i*(*9x8yP*(*x, y*)) = *F* porque:
   1. *i*(*P*(*a, b*)) = *F* porque *PD*(*i*(*a*)*, i*(*b*)) = *PD*(1*,* 2)= *F*
   2. y *i*(*P*(*b, a*)) = *F* porque *PD*(*i*(*b*)*, i*(*a*)) = *PD*(2*,* 1)= *F*

#### Ejemplo Consecuencia I y II Ejemplo Consecuencia Lógica I

Sea el argumento: Hay individuos inteligentes o que saben hablar. Juan no sabe hablar. Luego Juan no es inteligente.

Y su formalización: *{9x*(*I*(*x*) *\_H*(*x*))*, ¬H*(*a*)*}* ✏ *¬I*(*a*)

Para determinar la corrección de este argumento (si hay consecuencia lógica) por medios semánticos, intentamos encontrar un contramodelo, es decir una in- terpretación que:

1. Verifique las premisas: *i*(*9x*(*I*(*x*) *\_H*(*x*))) = *V* y *i*(*¬H*(*a*)) = *V*
2. Haga falsa la pretendida conclusión: *i*(*¬I*(*a*)) = *F*

Elijamos para ello un dominio de interpretación, p. ej., *D* = *{Juan, Pedro}* Ampliemos el lenguaje de formalización con una nueva constante: *b*

La interpretación:

1. *i*(*a*)= *Juan*, *i*(*b*)= *Pedro*
2. *ID*(*Juan*)= *V* , *ID*(*Pedro*)= *V* , *HD*(*Juan*)= *F*, *HD*(*Pedro*)= *V*

Verifica las premisas 1 y 2 y hace falsa la conclusión 3:

1. *i*(*9x*(*I*(*x*) *\_H*(*x*))) = *V* sii *i*((*I*(*x*) *\_H*(*x*))*{x/c}*)= *V*

para alguna constante *c* de *L*.

Sea *a* esa constante: *i*(*I*(*a*) *\_H*(*a*)) = *V* sii

* 1. *i*(*I*(*a*)) = *V* sii: *i*(*I*(*a*)) = *V* , *i*(*a*)= *Juan* y *ID*(*Juan*)= *V*
  2. o bien *i*(*H*(*a*)) = *V* ...

1. y *i*(*¬H*(*a*))) = *V* sii *i*(*H*(*a*)) = *F* sii: *i*(*a*)= *Juan* y *HD*(*Juan*)= *F*
2. y *i*(*¬I*(*a*)) = *F* sii *i*(*I*(*a*)) = *V* sii: *i*(*a*)= *Juan* y *ID*(*Juan*)= *V*

... es un contramodelo, luego el **argumento NO es correcto**.

#### Ejemplo Consecuencia Lógica II

Sea el argumento: Todo elemento químico es oxidante o reductor. El carbono es un elemento químico no oxidante. Luego el carbono es reductor.

Y su formalización: *{8x*(*E*(*x*) *! O*(*x*) *\_R*(*x*))*, E*(*a*) *^¬O*(*a*)*}* ✏ *R*(*a*)

Para determinar la corrección de este argumento por medios semánticos, trate- mos de encontrar un contramodelo que:

* 1. Verifique las premisas:

*i*(*8x*(*E*(*x*) *! O*(*x*) *\_R*(*x*))) = *V* y *i*(*E*(*a*) *^¬O*(*a*)) = *V*

* 1. Haga falsa la pretendida conclusión: *i*(*R*(*a*)) = *F*

Elijamos para ello un dominio de interpretación, p. ej., *D* = *{carbono, oxigeno}* Ampliemos el lenguaje de formalización con una nueva constante: *b*

La interpretación:

1. *i*(*a*)= *carbono*, *i*(*b*)= *oxigeno*
2. *ED*(*carbono*) = *ED*(*oxigeno*) = *V, OD*(*carbono*) = *F*, *OD*(*oxigeno*) =

*V, RD*(*carbono*)= *RD*(*oxigeno*)= *F*

No puede verificar las premisas y hacer falsa la conclusión:

1. *i*(*8x*(*E*(*x*) *! O*(*x*) *\_R*(*x*))) = *V* sii

* 1. *i*(*E*(*a*) *! O*(*a*) *\_R*(*a*)) = *V* sii *i*(*E*(*a*)) = *F* o *i*(*O*(*a*) *\_R*(*a*)) = *V*
  2. *i*(*E*(*b*) *! O*(*b*) *\_R*(*b*)) = *V* sii *i*(*E*(*b*)) = *F* o *i*(*O*(*b*) *\_R*(*b*)) = *V*

1. *i*(*E*(*a*) *^¬O*(*a*)) = *V* sii
   1. *i*(*E*(*a*)) = *V* y *i*(*O*(*a*)) = *F* por 1.1 *R*(*a*)= *V*
2. *i*(*R*(*a*)) = *F* por 1.1 y 2.1 contradicción.

... lo que impide crear el contramodelo no depende de *I*. Como no existe contramodelo, el **argumento es correcto**.

#### Ejercicios Validez Argumentos

Determina por medios semánticos la validez de los siguientes argumentos:

1. 0 *>* 1. 1 *>* 2. La relación ‘ *>0* es transitiva. Por tanto, 0 *>* 2.
2. Robin Hood es generoso. Todos los ladrones son generosos. Por tanto, Ro- bin Hood es un ladrón.
3. Siempre que Pedro discute con María, ésta se enfada con su padre. Una persona que se enfade con otra no la invita a su boda. Luego, si Pedro discute con María, ésta no invitará a su padre a su boda.
4. Todos los ladrones son generosos. Sólo los ricos son generosos. Robin Hood es un ladrón. Por tanto, Robin Hood es rico.
5. Juan Carlos ama a Sofía. Quienes trabajan con Juan Carlos no conocen a fondo a Sofía. Si se ama a alguien, se le conoce a fondo. Sabino trabaja con Juan Carlos. Por tanto, Sabino no ama a Sofía.

Determina por medios semánticos la validez de los siguientes argumentos:

1. Hay un ladrón generoso. Por tanto, hay un ladrón y hay alguien generoso.
2. Hay un ladrón y hay alguien generoso. Por tanto, hay un ladrón generoso.
3. Todo elemento pesado es metálico. Por tanto, ningún no metal es un ele-

mento pesado.

1. No es cierto que todo sea blanco o negro. Por tanto, hay algo que no es ni blanco ni negro.

## Teorema de la demostración y deducción natural.

### Decibilidad de la lógica de Primer Orden.

Entscheidungsproblem: Denominación original del problema planteado por Hil- bert, que consiste en determinar si un argumento dado es correcto o no para la Lógica de Primer Orden.

Problema de la parada: saber si dada una tarea (descripción de sus instrucciones) y un input para la misma es posible determinar de manera efectiva si dicha tarea finalizaría arrojando un output, no importa cuál.

Teorema (Church y Turing 1936)

La Lógica de Primer Orden con identidad no es de- cidible.

Demostración: Determinar el problema de parada se reduce a demostrar en LPO la fórmula que expresa la existencia de un output a partir de la aplicación de una serie de instrucciones previamente fijadas. ⇤

Es decir, NO es posible decidir, para un conjunto cualquiera de fórmulas G de un LPO, si G *`A* o bien G *6` A*.

Ejemplo:

* Sea G = *{P*(*a*)*, 8x*( *P*(*x*) *! P*( *f* (*x*)) )*}*,
* G *`P*(*a*)*,* G *`P*( *f* (*a*))*,* G *`P*( *f* ( *f* (*a*)))*,* G *`P*( *f* ( *f* ( *f* (*a*))))*,. ..*
* ¿G *`P*( *f* (*b*))?o ¿G *6` P*( *f* (*b*))?

Estrategias:

* + - 1. Extender la deducción Natural de la lógica proposicional.
      2. Utilizar el método de resolución de Robinson (1965).
         * A partir de fórmulas simplificadas (Forma Normal de Skolem).
         * Resolver usando el unificador de máxima generalidad (UMG).
         * Elegir una estrategia de resolución computacionalmente eficiente.

### Deducción Natural

Al igual que en Lógica Proposicional podemos aplicar Deducción Natural inclu- yendo 4 nuevas reglas de inferencia.

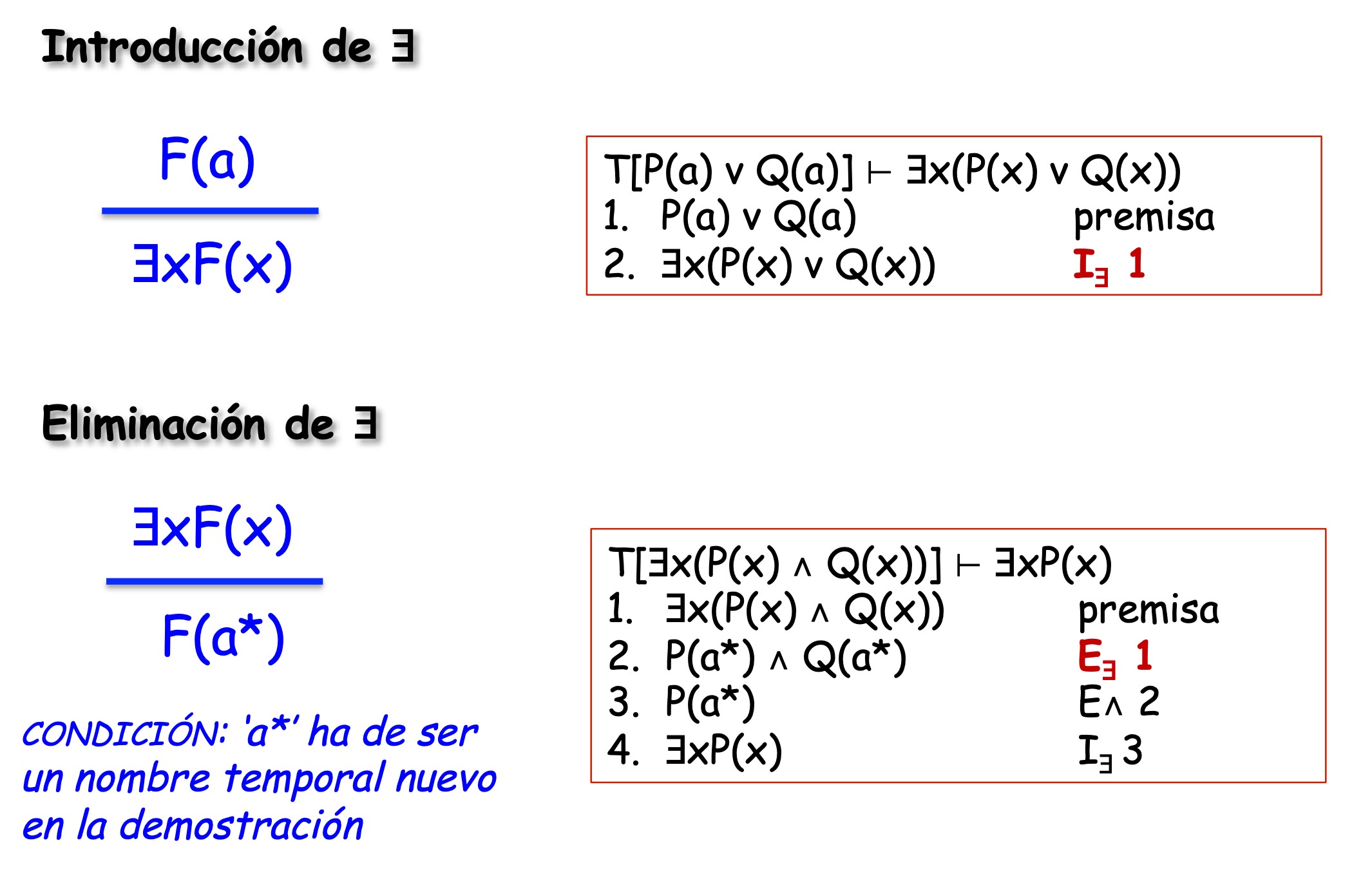
Hay fórmulas de LPO que podemos demostrar sin añadir nuevas reglas:

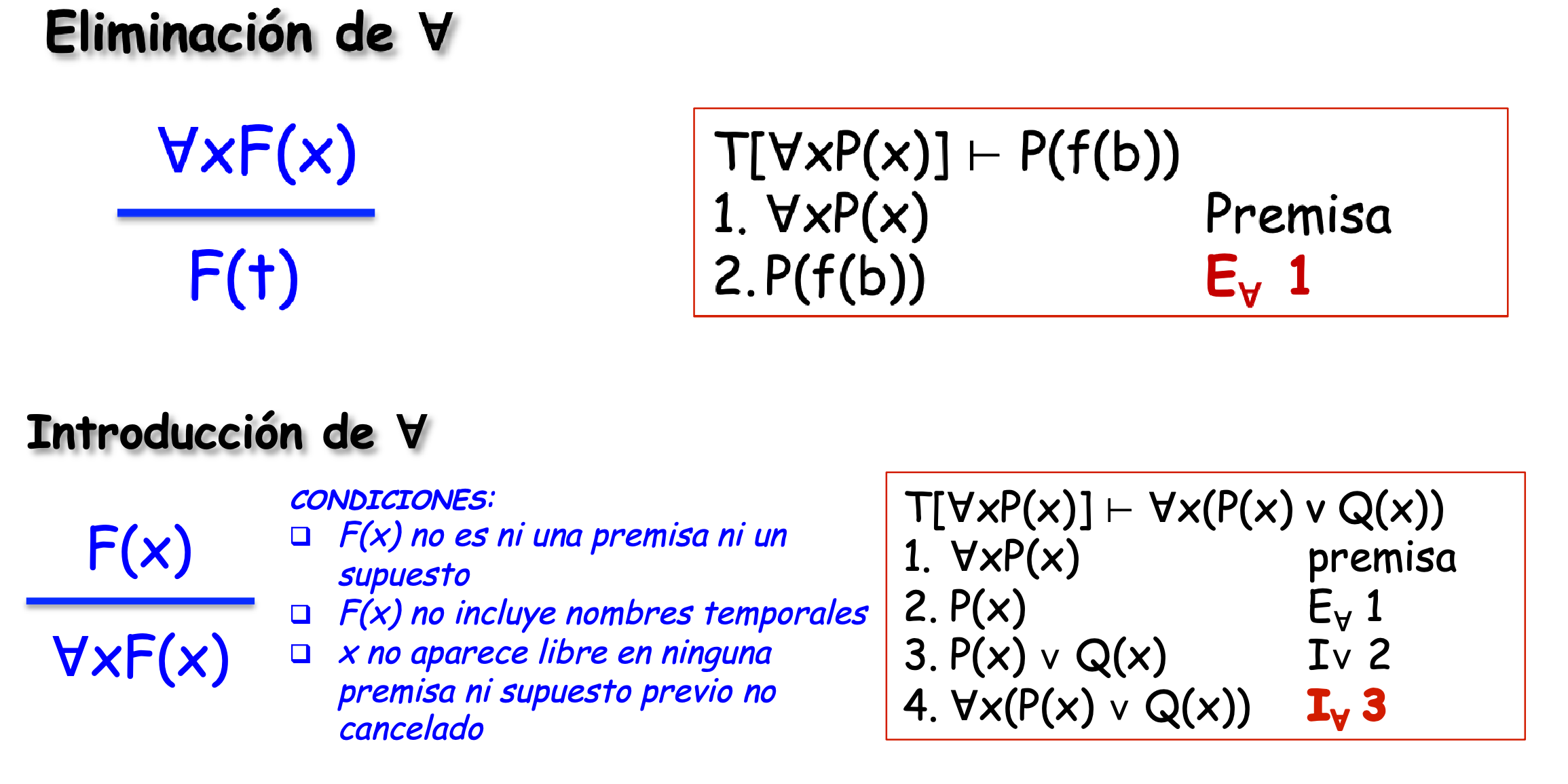
*T* [*9xP*(*x*) *! 8yQ*(*y*)*, 8yQ*(*y*) *! 8zR*(*z*)] *` 9xP*(*x*) *! 8zR*(*z*)

En general las nuevas reglas son necesarias para:

* “abrir” las fórmulas cuantificadas (eliminar cuantificadores).
* aplicar las reglas de inferencia proposicionales.
* “cerrar” las fórmulas resultantes (introducir cuantificadores).

#### Deducción Natural: Reglas de inferencia

**Existencial**

**Universal**

#### Ejemplos: Deducción Natural

*T* [*9x*(*P*(*x*) *^Q*(*x*))] *` 9xP*(*x*) *^ 9xQ*(*x*)

* + - 1. *9x*(*P*(*x*) *^Q*(*x*)) premisa

2. *P*(*a⇤*) *^Q*(*a⇤*) *E9*(1)

3. *P*(*a⇤*) *E^*(2)

4. *9xP*(*x*) *I9*(3)

5. *Q*(*a⇤*) *E^*(2)

6. *9xQ*(*x*) *I9*(5)

7. *9xP*(*x*) *^ 9xQ*(*x*) *I^*(4*,* 6)

Se introduce *a⇤* como una constante nueva, sólo vigente mientras llegamos a deducir la fórmulas que nos interesan, que son *9xP*(*x*) y *9xQ*(*x*)

#### Deducción Natural: Ejemplo (Incorrecto)

*T* [*9xP*(*x*) *^ 9xQ*(*x*)] *` 9x*(*P*(*x*) *^Q*(*x*))

1. *9x*(*P*(*x*) *^ 9xQ*(*x*) premisa 2. *9xP*(*x*) *E^*(1)

3. *P*(*a⇤*) *E9*(2)

4. *9xQ*(*x*) *E^*(1)

5. *Q*(*a⇤*) *E9*(4)

6. *P*(*a⇤*) *^Q*(*a⇤*) *I^*(3*,* 5)

7. *9*(*xP*(*x*) *^xQ*(*x*)) *I9*(6)

Al usar otra vez *a* damos a entender que el elemento que cumple *Q* es el mismo que cumple *P*, y no tiene por qué ser así.

*⇤*

#### Deducción Natural: Ejemplo

*T* [*8x*(*P*(*x*) *! Q*(*x*))*, 8x*(*Q*(*x*) *! R*(*x*))] *` 8x*(*P*(*x*) *! R*(*x*))

* 1. *8x*(*P*(*x*) *! Q*(*x*)) premisa
  2. *8x*(*Q*(*x*) *! R*(*x*)) premisa 3. *P*(*x*) *! Q*(*x*) *E8*(1)

4. *Q*(*x*) *! R*(*x*) *E8*(2)

1. *P*(*x*) supuesto

6. *Q*(*x*) *E!*(3*,* 5)

7. *R*(*x*) *E!*(4*,* 6)

8. *P*(*x*) *! R*(*x*) *I!*(5*,* 7)

9. *8x*(*P*(*x*) *! R*(*x*)) *I8*(8)

#### Deducción Natural: Ejemplo (Incorrecto)

*T* [*9xP*(*x*)] *` 8xP*(*x*)

* 1. *9xP*(*x*) premisa 2. *P*(*a⇤*) *E8*(1) 3. *8xP*(*x*) *I8*(2)

*T* [*8x9yM*(*y, x*)] *` 9y8xM*(*y, x*)

1. *8x9yM*(*y, x*) premisa 2. *9yM*(*y, x*) *E8*(1)

3. *M*(*a⇤, x*) *E9*(2)

4. *8xM*(*a⇤, x*) *I8*(3)

5. *9y8xM*(*y, x*) *I9*(4)

No se puede generalizar una fórmula que tiene constantes temporales.

## Simplificación de fórmulas.

### Objetivo.

Queremos simplificar las fórmulas mediante transformaciones de una fórmula *F* a otra *F0* (preservando ciertas propiedades):

* La satisfabilidad: Existe un modelo *I* de *F* sii existe un modelo *I0* (proba- blemente no el mismo) de *F0*

SAT(*9x p*(*x*)) sii SAT(*p*(*a*))

* La semántica: Para toda interpretación *I*, *I* es un modelo de *F* sii es un modelo de *F0*

*8x p*(*x*) es semánticamente equivalente a *¬9x ¬p*(*x*)

Ejemplo: De LPO a Forma Normal de Skolem (FNS) a Forma Clausular:

Fórmula en LPO *9y*(*8x*(*p*(*x, f* (*y*)) *! q*(*z*)) *^¬9w p*(*g*(*w*)*, y*))

*# #*

Fórmula en FNS *8x8w*(*¬p*(*x, f* (*b*)) *\_q*(*a*) *^¬p*(*g*(*w*)*, b*))

*# #*

Forma Clausular *{¬p*(*x, f* (*b*)) *\_q*(*a*)*, ¬p*(*g*(*w*)*, b*)*}*

Preserva la satisfabilidad pero NO preserva todos los modelos (la semántica): el resultado no es equivalente a la fórmula original.

### Forma Normal de Skolem

Procedimiento para obtener la Forma Normal de Skolem (FNS) de una fórmula

*A*:

* + - 1. **Obtener la forma Prenex**: Todos los cuantificadores a la cabeza de la fór- mula. La forma Prenex(A) siempre existe, aunque puede no ser única.
      2. **Realizar el cierre existencial**: Se quitan las variables libres. Una fórmula

*A*(*x*) es satisfacible sii *9x A*(*x*) es satisfacible.

* + - 1. **Obtener la forma normal conjuntiva**: Conjunción de disyunciones. La forma normal conjuntiva de *A* siempre existe.
      2. **Eliminar los cuantificadores existenciales**: Dejar solo los cuantificadores universales. El resultado es la FNS.

Una fórmula *A* es satisfacible sii *FNS*(*A*) es satisfacible.

#### Forma Prenex

Poner cuantificadores a la cabeza de la fórmula usando:

* + Cambio de nombre de variables ligadas:

*8xA*(*x*) *$ 8yA*(*x/y*) *9xA*(*x*) *$ 9yA*(*x/y*)

... si *y* no está libre en *A*

* + Interdefinición de cuantificadores:

*¬8xA*(*x*) *$ 9x¬A*(*x*) *¬9xA*(*x*) *$ 8x¬A*(*x*)

* + Distribución de conectivas respecto a cuantificadores:

*8xA^C $ 8x*(*A^C*) (*8xA ! C*) *$ 9x*(*A ! C*)

*9xA^C $ 9x*(*A^C*) (*9xA ! C*) *$ 8x*(*A ! C*)

*8xA\_C $ 8x*(*A\_C*) (*A ! 8xC*) *$ 8x*(*A ! C*)

*9xA\_C $ 9x*(*A\_C*) (*A ! 9xC*) *$ 9x*(*A ! C*)

... si x no está libre en la otra subfórmula

*8xA^ 8xC $ 8x*(*A^C*) (*9xA\_ 9xC*) *$ 9x*(*A\_C*)

#### Cierre existencial

Las variables libres de la fórmula se ligan existencialmente poniendo el cuanti- ficador correspondiente en cabeza de la fórmula.

Ejemplo:

*8y9z*(*P*(*x*) *^Q*(*y*) *! r*( *f* (*z*)*, x*)) *#*

*9x8y9z*(*P*(*x*) *^Q*(*y*) *! r*( *f* (*z*)*, x*))

#### Forma normal conjuntiva

Transformar en conjunción de disyunciones usando:

* + Interdefinición:

(*A ! B*) *$* (*¬A\_B*)

(*A $ B*) *$* (*A ! B*) *^* (*B ! B*)

* + Leyes de De Morgan

*¬*(*A^B*) *$ ¬A\_¬B*

*¬*(*A\_B*) *$ ¬A^¬B*

* + Distribución de *\_* y *^ A^* (*B\_C*) *$* (*A^B*) *\_* (*A^C*) *A\_* (*B^C*) *$* (*A\_B*) *^* (*A\_C*)

#### Eliminación de cuantificadores existenciales

Se elimina el cuantificador existencial sustituyendo la variable que ligaba por una función de Skolem o constante de Skolem.

La función de Skolem será una función nueva en la fórmula, aplicada a todas la variables cuantificadas universalmente que aparecen antes que el cuantificador existencial a eliminar. Si no hay tales variables se utilizará una constante nueva en la fórmula para hacer la sustitución.

Ejemplos:

* + *8x9y*(*p*(*x*) *! ¬q*(*y*)) se transforma en *8x*(*p*(*x*) *! ¬q*(**f(x)**)
  + *9x8z*(*q*(*x, z*) *\_r*(*a, x*)) se transforma en *8z*(*q*(**b***, z*) *\_r*(*a,* **b**))
  + *9x8y9z*(*p*(*x*) *^q*(*y*) *! r*( *f* (*z*)*, x*)) se transforma en

*8y*(*p*(**a**) *^q*(*y*) *! r*( *f* (**g(y)**)*,* **a**))

### Forma Clausular

Una **cláusula** es una disyunción de literales.

La **forma clausular** es la conjunción de las cláusulas, cuyas variables están todas ellas ligadas universalmente.

Ejemplo:

*A* : *8x8w*(*¬p*(*x, f* (*b*)) *\_q*(*a*) *^¬p*(*g*(*w*)*, b*)*e*)

*#*

*FC*(*A*) : *{¬p*(*x, f* (*b*)) *\_q*(*a*)*, ¬p*(*g*(*w*)*, b*)*}*

Teorema Una fórmula F es satisfacible sii FC(F) es satis- facible

#### Forma Clausular: De una deducción

Una deducción *T* [*P*1*, P*2*,..., Pn*] *`C* es correcta sii

*P*1 *^ P*2 *^··· ^¬C* es insatisfacible Por lo tanto dada una deducción *T* [*P*1*, P*2*,..., Pn*] *`C*:

* Obtener la forma clausular de cada *Pi*, *i i n*.
* Obtener la forma clausular de *¬C*.
* Realizar la unión de todos los conjuntos de cláusulas.
* Comprobar la satisfacibilidad (p.ej. mediante el método de resolución de Robinson).

Objetivo Una deducción *T* [*P*1*, P*2*,..., Pn*] *C* es correcta sii

*`*

*{P*1*, P*2*,..., ¬C}* es insatisfacible

## Método de resolución de Robinson.

### Introducción

Una fórmula *A* es insatisfacible sii no existe ninguna interpretación *I* tal que

*I*(*A*)= *V* (sii todas las interpretaciones la evalúan como falsa)

* Si A es una fórmula proposicional el número de interpretaciones a analizar es finito: 2*n*, siendo *n* el número de proposiciones diferentes que la integran.
  + Si A es una fórmula de primer orden el número de interpretaciones a anali- zar sería infinito y no numerable.

Solución: Las interpretaciones de Herbrand:

* + Cuyo número sea menor (finito o infinito numerable como mucho).
  + Cuyo análisis fuese suficiente para determinar la insatisfaciblidad de A (si A es insatisfacible para esas interpretaciones también lo es para cualquier otra).

### Interpretaciones de Herbrand

#### Universo de Herbrand

El dominio de la interpretación de un fórmula *A* en las interpretaciones de Her- brand, es *H*, el **Universo de Herbrand de** *A*.

Se define de manera recursiva a partir de las constantes y funciones de *A*, aplica- das de todas las formas posibles.

* Sea *C* el conjunto de constantes de *A*.
* Sea *F* el conjunto de funciones de *A*.

*H*0 = *C* si *C 6*= /0 o *H*0 = *{a}* si *C* = /0

*Hi* = *{ f* (*t*1*,...,tn*)*/t j 2* (*Hi-*1 *[···[ H*0)*, f /n 2 F} H* = *H*0 *[ H*1 *[···[ Hi [...*

#### Ejemplos: Universos de Herbrand

*A* = *{P*(*x*)*, Q*(*y*)*} C* = /0 *F* = /0

*H*0 = *{a}*

*H*1 = *H*2 = *·· ·* = /0

*H* = *{a}*

*A* = *{P*(*a*)*, Q*(*y*) *\_¬R*(*b, f* (*x*))*} C* = *{a, b} F* = *{ f /*1*}*

*H*0 = *{a, b}*

*H*1 = *{ f* (*a*)*, f* (*b*)*}*

*H*2 = *{ f* (*a*)*, f* ( *f* (*a*))*, f* (*b*)*, f* ( *f* (*b*))*}*

*.. .*

*H* = *{a, b, f* (*a*)*, f* (*b*)*, f* ( *f* (*a*))*, f* ( *f* (*b*))*,. ..}* = = *{a, b, f n*(*a*)*, f n*(*b*)*, 8n 2:* 1*}*

**Átomo básico**: Se obtiene aplicando un símbolo de predicado de la fórmula a términos de su universo de Herbrand.

**Base de Herbrand** (BH) de una fórmula *A*: Conjunto de todos los átomos bási- cos de *A*.

Contiene todos los predicados de *A* aplicados a todos los términos de *H* de todas las formas posibles.

* Sea *P* el conjunto de predicado de *A*

*BH* = *{P*(*t*1*,...,tn*)*/t j 2 H, P/n 2 P}*

*A* = *{P*(*x*)*, Q*(*y*)*} H* = *{a}*

*BH* = *{P*(*a*)*, Q*(*a*)*}*

*A* = *{P*(*a*)*, Q*(*y*) *\_¬R*(*b, f* (*x*))*} H* = *{a, b, f* (*a*)*, f* (*b*)*,...}*

*BH* = *P*(*a*)*, P*( *f* (*a*))*,. .., P*( *f n*(*a*))*,. ..,*

*{*

*P*(*b*)*, P*( *f* (*b*))*,. .., P*( *f n*(*b*))*,. ..,*

*Q*(*a*)*, Q*( *f* (*a*))*,. .., Q*( *f n*(*a*))*,. ..,*

*Q*(*b*)*, Q*( *f* (*b*))*,. .., Q*( *f n*(*b*))*,. ..,*

*R*(*a, a*)*, R*(*a, b*)*,..., R*(*a, f n*(*a*))*,. ..,*

*R*(*b, a*)*,..., R*( *f n*(*b*)*, f m*(*a*))*,. ..}* =

= *{P*(*t*)*, Q*(*t*)*, R*(*t, t0*)*, 8t, t0 2 H}*

#### Definición

Finalmente, una interpretación de Herbrand (IH) de una fórmula *A* es un inter- pretación sobre *H* tal que:[1](#_bookmark53)

* Cada constante *a 2 C* se asocia consigo misma: *IH*(*a*)= *a*
* Cada símbolo de función *f /n 2 F* se asocia con la función *fH/n*
  + *fH* : *Hn ) H*

*o IH*( *f* (*t*1*, t*2*,...,tn*)) = *fH*(*IH*(*t*1)*, IH*(*t*2)*,..., IH*(*tn*)) =

*fH*(*t*1*, t*2*,...,tn*)= *t*

* Cada símbolo de predicado *P/n 2 P* se asocia con el predicado *PH/n*
  + *PH* : *Hn ) {V, F}*

*o IH*(*P*(*t*1*,...,tn*)) = *PH*(*IH*(*t*1)*,..., IH*(*tn*)) =

*PH*(*t*1*, t*2*,...,tn*) = V (o F)

* Cada átomo básico de *BH* tiene un valor de verdad.

1Notación: *C* es el conjunto de constantes, *F* el conjunto de símbolos de función y

*P* el conjunto de símbolos de predicado del LPO de *A*

Una interpretación Herbrand se puede representar como el conjunto de los áto- mos básicos de la Base de Herbrand, afirmados si se interpretan como verdaderos y negados si se interpretan como falsos

*A* = *{P*(*x*)*, Q*(*y*)*} BH* = *{P*(*a*)*, Q*(*a*)*}*

*IH*1 = *{P*(*a*)*, Q*(*a*)*} IH*1(*P*(*a*))=V, *IH*1(*Q*(*a*))=V

*IH*2 = *{P*(*a*)*, ¬Q*(*a*)*} IH*2(*P*(*a*))=V, *IH*2(*Q*(*a*))=F

*IH*3 = *{¬P*(*a*)*, Q*(*a*)*} IH*3(*P*(*a*))=F, *IH*1(*Q*(*a*))=F

*IH*4 = *{¬P*(*a*)*, ¬Q*(*a*)*} IH*4(*P*(*a*))=F, *IH*4(*Q*(*a*))=F

#### Satisfabilidad

Una fórmula *A* es insatisfacible sii

*A* es falsa para todas sus interpretaciones Herbrand (IH*i*)

Por lo tanto, para estudiar la insatisfacibilidad de una fórmula basta con estudiar las interpretaciones Herbrand de su forma clausular:

* Si alguna IH*i* evalúa *A* como verdadera entonces *A* es satisfacible.
* Si ninguna IH*i* evalúa *A* como verdadera entonces es insatisfacible. Ejemplo: *A* = *{P*(*x*)*, Q*(*y*)*}*
* *IH*1 = *{P*(*a*)*, Q*(*a*)*}*

es un modelo de *A* (hace verdad todas las instancias).

* *IH*2 = *{P*(*a*)*, ¬Q*(*a*)*}, IH*3 = *{¬P*(*a*)*, Q*(*a*)*}, IH*4 = *{¬P*(*a*)*, ¬Q*(*a*)*}*,

son contramodelos de *A* (falsifican una o las dos instancias).

### Teorema de Herbrand

Cualquier interpretación de Herbrand que evalúe una fórmula *A* como verdadera, debe hacer verdad a todas y cada una de las instancias básicas de la fórmula.

Teorema de Herbrand

Un conjunto de cláusulas *C* es insatisfacible sii existe un conjunto finito de instancias básicas de cláusulas de *C* que es insatisfacible.

El teorema sugiere que dado un conjunto de cláusulas *C* insatisfacible, mediante un procedimiento mecánico generar incrementalmente conjuntos de instancias básicas *S*1*, S*2*,..., Sn,...* de cláusulas de *C* y analizar de forma sucesiva la insa- tisfacibilidad de *S*1*, S*2*,..., Sn,...* , se podría detectar un *k* tal que *Sk* es insatisfa- cible.

* P.ej., método de Resolución de Robinson (1965).

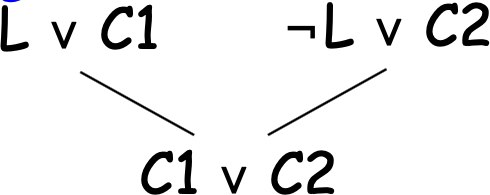
### Método de Resolución de Robinson

**Idea general**: Plantear un método de obtención de nuevas instancias deducidas del conjunto original, de forma que si llega a deducirse un literal y su negación puede concluirse que el conjunto original es insatisfacible.

Está basado en la **regla de resolución básica**: De dos instancias básicas *L C*1 y *L C*2 (*L* es un literal) puede deducirse una nueva instancia básica *C*1 *C*2, llamada **resolvente**:

*¬ \_ \_*

*\_*

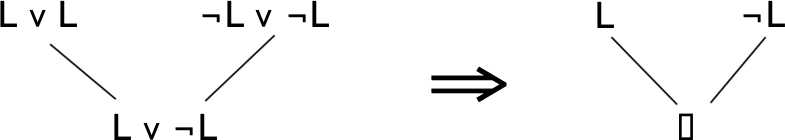


La aplicación sucesiva de la regla de resolución permite obtener una **contradic- ción** cuando el conjunto original es insatisfacible.

La contradicción se obtiene cuando se deducen dos instancias básicas (literales aislados) *L* y *L*. La aplicación de la regla sobre *L* y *L* genera [], llamada cláusula vacía.

*¬ ¬*

Para asegurarnos deducir la cláusula vacía hay que tener en cuenta la idempo- tencia *L\_L ) L*



o aplicar la regla de resolución básica extendida, que de dos instancias *L \_··· \_ L\_C*1y *¬L\_··· \_¬L\_C*1 deduce *C*1 *\_C*2.

#### Algoritmo

Dado un conjunto C de instancias básicas:

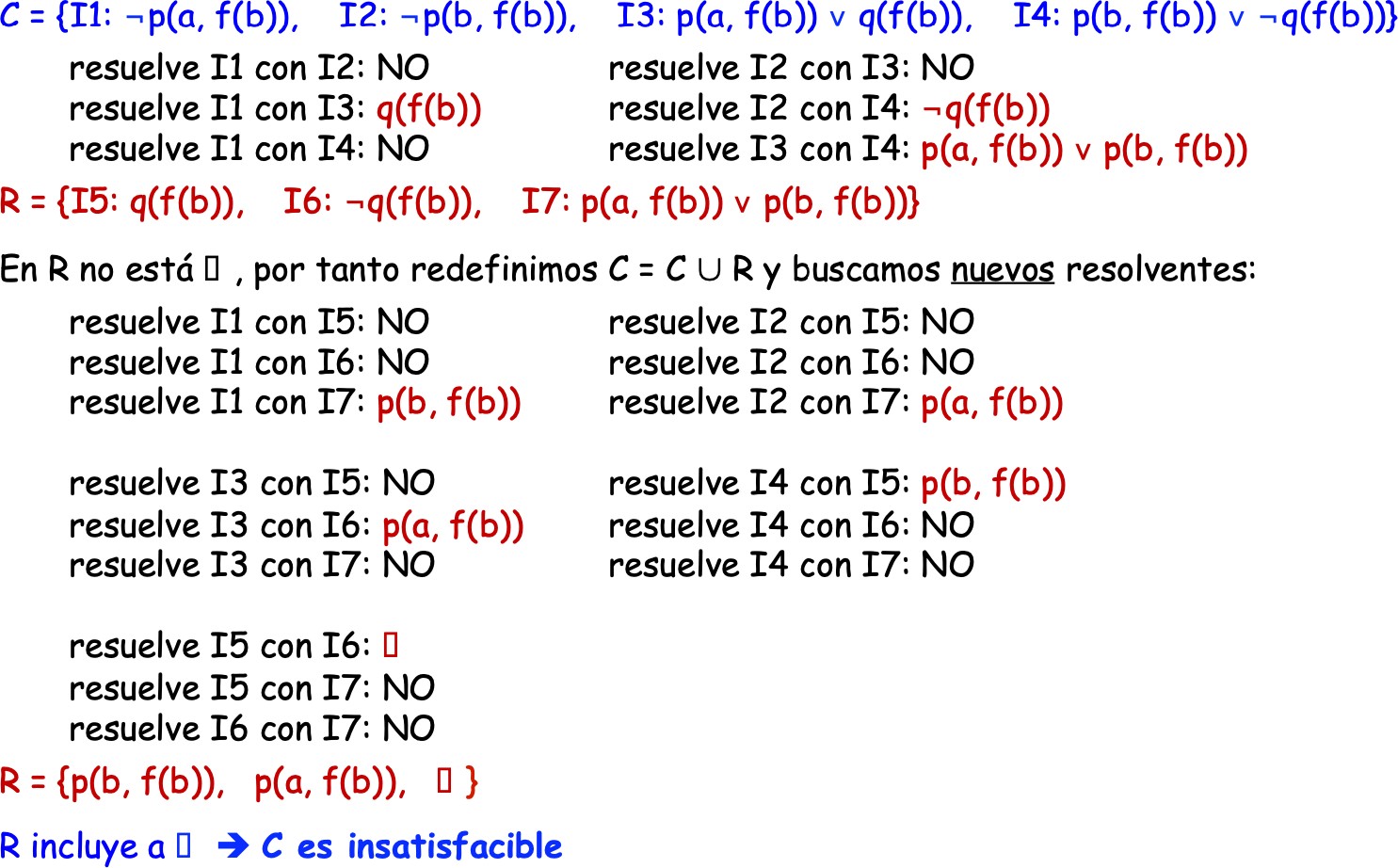
* + - 1. Generar el conjunto *R* de todos los resolventes que pueden obtenerse apli- cando la regla de resolución entre instancias del conjunto *C* de todas las formas posibles.
      2. Si [] está incluida en *R* entonces terminar *) C* es insatisfacible.
      3. Si *R ✓ C* significa que ya se han generado todos los resolventes posibles, entonces terminar *) C* es satisfacible.
      4. Hacer *C* = *C [ R* y repetir desde paso 1.

El método es **correcto**: Si deducimos [] entonces *C* es insatisfacible.

El método es **completo**: Si *C* is insatisfacible, entonces con la aplicación de la regla de resolución deduciremos [].

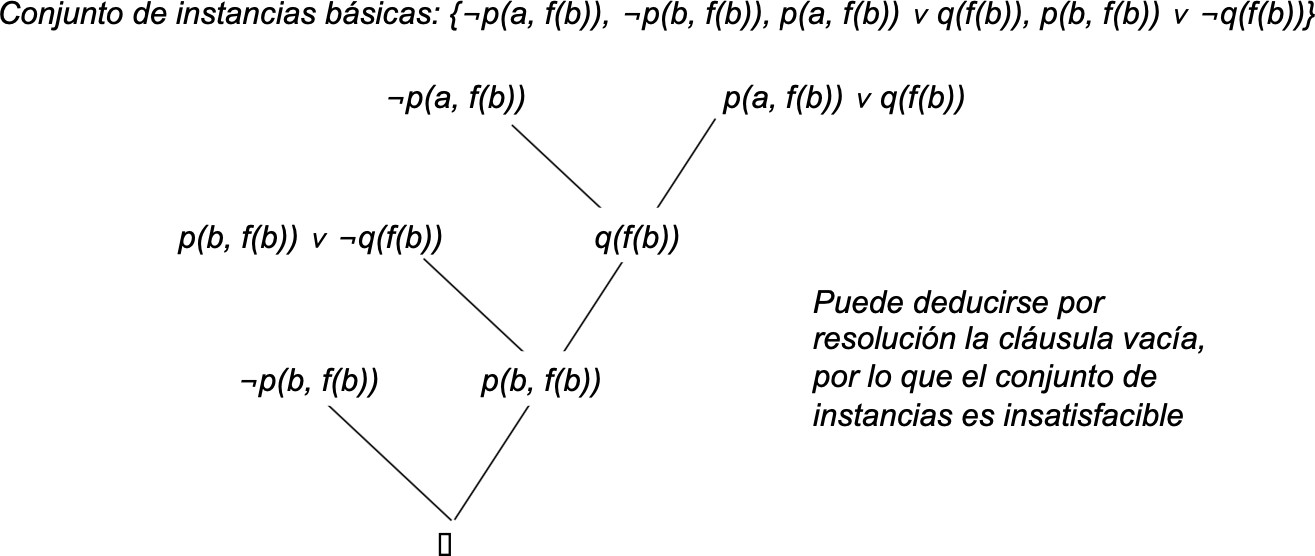
#### Ejemplo: Método de Resolución de Robinson

Resolución aplicando el procedimiento de saturación:



En la práctica, la aplicación de sucesivos pasos de resolución se puede represen- tar en forma de árbol (**árbol de resolución**):

* + - * + Árbol binario invertido (cada dos nodos tienen un “hijo” común)
        + Cada nodo representa una instancia básica.
        + Sólo se representan los pasos relevantes para llegar a [].



### Resolución con Unificador de Máxima Generalidad

#### Sustitución

Un **sustitución** *a* = *x*1*/t*1*,..., xn/tn* es una función finita de un conjutno de variables de un lenguaje en el de términos. Donde *xi/ti* es una **ligadura**.

*{ }*

Dominio(*a*)=*{xi* : *xi/ti 2 a}*

Rango(*a*)=*{yi* : *yiapareceenti, xi/ti 2 a}*

Ejemplos

*a*1 = *{x/ f* (*a*)*, y/x, z/h*(*b, y*)*, w/a} Dominio*(*a*1)= *{x, y, z, w} Rango*(*a*1)= *{x, y}*

*a*2 = *{x/a, y/a, z/h*(*b, c*)*, w/ f* (*d*)*} Dominio*(*a*2)= *{x, y, z, w} Rango*(*a*2)= /0

*a*3 = *{x/y, z/w}* **Renombrado**

*l* = *{x/x, y/y, z/z}* **Sustitución vacía**

Dada una fórmula *F* y una sustitución *a* = *x*1*/t*1*,..., xn/tn* , se denomina **apli- cación de** *a* **a** *F* **(***Fa***)** a la fórmula obtenida reemplazando simultáneamente cada ocurrencia en *F* de *xi* por *ti*, para cada *xi/ti 2 a*.

*{ }*

Ejemplo: *a* = *{x/ f* (*a*)*, y/x, z/h*(*b, y*)*, w/a}*

* *P*(*x, y, z*)*a* = *P*( *f* (*a*)*, x, h*(*b, y*)) Correcto
* *P*(*x, y, z*)*a* = *P*( *f* (*a*)*, f* (*a*)*, h*(*b, f* (*a*))) Incorrecto

Si *Dominio*(*a*) *\ Rango*(*a*)= 0/ , *a* es **idempotente** y (*Fa*)*a* = *Fa*:

* *a*1 = *{x/a, y/ f* (*b*)*, z/v}* Es idempotente

*P*(*x, y, w, z*)*a*1 = *P*(*a, f* (*b*)*, w, v*) *P*(*a, f* (*b*)*, w, v*)*a*1 = *P*(*a, f* (*b*)*, w, v*)

* *a*2 = *{x/a, y/ f* (*b*)*, z/x}* No es idempotente

*P*(*x, y, w, z*)*a*2 = *P*(*a, f* (*b*)*, w, x*) *P*(*a, f* (*b*)*, w, x*)*a*2 = *P*(*a, f* (*b*)*, w, a*) *F0* es instancia de *F* si existe una sustitución *a 6*= 0/ tal que *F0* = *Fa*

#### Composición sustituciones

Dadas dos sustituciones:

*a* = *{x*1*/t*1*,..., xn/tn}* y *b* = *{y*1*/s*1*,..., ym/sm}*

su composición *ab* se define como el conjunto

*{x*1*/t*1*b,..., xn/tnb, y*1*/s*1*,..., ym/sm}* una vez eliminadas:

* las ligaduras *xi/tib* tales que *xi ⌘ tib* ,
* y las ligaduras *yi/si* tales que *yi 2 {x*1*,..., xn}*

**Ejemplo**: Si *a* = *{x/*3*, y/ f* (*x,* 1)*}* y *b* = *{x/*4*}* entonces:

* *ab* = *{x/*3*, y/ f* (4*,* 1)*}*
  + *ba* = *{x/*4*, y/ f* (*x,* 1)*}*

Propiedades de la composición:

* + (*Fa*)*b* = *F*(*ab* )
  + (*ab* )*g* = *a*(*bg*)
  + *al* = *la* = *a*
  + *ab 6*= *ba*

#### Unificadores y UMG

Una sustitución *a* es un **unificador** de dos fórmulas *A* y *B* si *Aa* = *Ba*. En este caso se dice que *A* y *B* son **unificables**.

Un unificador *a* de *A* y *B* se denomina **unificador de máxima generalidad** (umg) sii para cualquier otro unificador *b* de *A* y *B* existe alguna sustitución *g* tal que *b* = *ag*

* + Si dos fórmulas son unificables entonces tienen umg
  + El umg de dos fórmulas es único (salvo renombrado) Ejemplo: *A ⌘ P*(*x, f* (*x, g*(*y*))*, z*) y *B ⌘ P*(*r, f* (*r, u*)*, a*)

*a*1 = *{x/r, u/g*(*y*)*, z/a} a*2 = *{x/a, r/a, y/b, u/g*(*b*)*, z/a}*

*Aa*1 = *Ba*1 = *P*(*r, f* (*r, g*(*y*))*, a*) *Aa*2 = *Ba*2 = *P*(*a, f* (*a, g*(*b*))*, a*)

*a*1 y *a*2 son unificadores de *A* y *B*.

* + *a*1 es el umg de *A* y *B*, porque con *g* = *{r/a, y/b}*, *a*2 = *a*1*g*.

#### Algoritmo de Unificación

Sean A y B dos átomos con el mismo símbolo de predicado:

* + - 1. *a* = *l*
      2. Mientras A*a 6*= B*a*:

1. Encontrar el símbolo más a la izquierda en A*a* tal que el símbolo corres- pondiente en B*a* sea diferente.
2. Sean *tA* y *tB* los términos de A*a* y B*a* que empiezan con esos símbolos:
   * Si ni *tA* ni *tB* son variables o, si uno de ellos es una variable que aparece en el otro *)* terminar con fallo (*A* y *B* no son unificables)
   * En otro caso, sea *tA* una variable *)* el nuevo *a* es el resultado de

*a{tA/tB}*

* + - 1. Terminar, siendo *a* el umg de *A* y *B*

#### Ejemplo: Resolución con UMG: Algoritmo de Unificación

Ejemplo 1: *A* = *P*(*x, x*) y *B* = *P*( *f* (*a*)*, f* (*b*))

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *a* | A*a* | B*a* | | (*ta, tb*) |  |
| *l*  *{x/ f* (*a*)*}* | P(x,x) P(f(a),f(a)) | P(f(a),f(b))  P(f(a),f(b)) | | (x,f(a)) (a,b) |
| FALLO A y B NO son unificables  Ejemplo 2: *A* = *P*(*x, f* (*y*)) y *B* = *P*(*z, x*) | | | | |
| *a* | A*a* | | B*a* | (*ta, tb*) | |
| *l* | P(x,f(y)) | | P(z,x) | (x,z) | |
| *{x/z}* | P(z,f(y)) | | P(z,z) | (f(y),z) | |

*{x/ f* (*y*)*, z/ f* (*y*)*}* P(f(y),f(y)) P(f(y),f(y)) ÉXITO A y B son unificables, su umg es *{x/ f* (*y*)*, z/ f* (*y*)*}*

#### Definición

**Regla de resolución con umg**: Sean *L*1 *\_··· \_Ln \_C*1 y *¬L*1*0 \_··· \_¬Lm0 \_C*2 dos

cláusulas, donde todos los *Lij* son literales con el mismo símbolo de predicado. Puede deducirse una nueva cláusula (*C*1*r*1 *\_C*2*r*2)*b* , llamada resolvente, donde

* *r*1 y *r*2 son renombrados cuyos dominios respectivos son todas las varia- bles de cada cláusula y *Rango*(*r*1) *\ Rango*(*r*2)= /0
* *b* es **umg** de *{L*1*r*1*,..., Lnr*1*, ¬L*1*0 r*2*,..., ¬Lm0 r*2*}*

La regla de resolución con umg se apoya en una versión de la **regla de factori- zación** para LPO: Dada una cláusula *L*1 *Ln C*, siendo *L*1*,..., Ln* literales con el mismo símbolo de predicado, puede deducirse una nueva cláusula *L Cb* donde

*\_*

*\_··· \_ \_*

* *b* es unificador de *L*1*,..., Ln*
* *L* = *L*1*b* = *·· ·* = *Lnb* El literal *L* se denomina

factor de *L*1 *\_··· \_Ln \_C*

La aplicación de la regla de resolución con UMG es **correcta**. La aplicación de la regla de resolución con UMG es **completa**.

Teorema

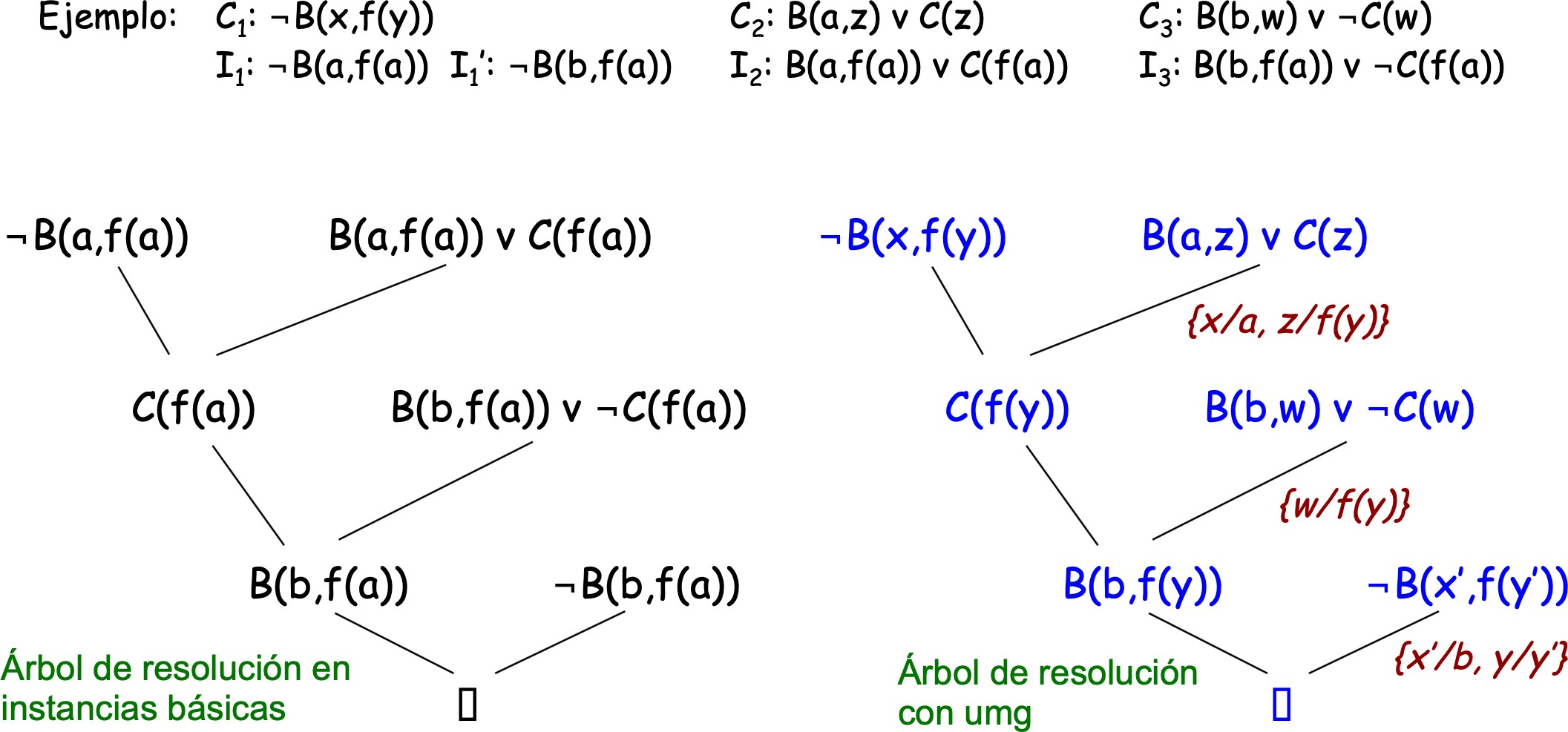
Un conjunto de cláusulas es insatisfacible sii se puede deducir o a partir de él por resolución con umg.

Por tanto, el método general de insatisfacibilidad se puede reducir a la búsqueda de [] a partir del conjunto de cláusulas, en lugar de tener que generar conjuntos de instancias básicas.

#### Ejemplo: Resolución con UMG

Pueden construirse árboles de resolución en los que los resolventes de cada dos cláusulas se obtienen en un paso de resolución con umg.

* + Por cada paso de resolución en instancias básicas puede definirse un paso de resolución con umg.



### Estrategias de resolución

#### Motivación

Distintas estrategias de resolución tienen sus ventajas e incovenientes.

* P.ej., la aplicación del procedimiento de saturación, sin limitaciones, genera normalmente muchas cláusulas irrelevantes y redundantes.

Para hacer el proceso de resolución computacionalmente eficiente es necesario aplicar criterios selectivos de forma sistemática que simplifiquen el proceso.

* Estrategias de **simplificación**: con el objetivo de reducir el número de cláu- sulas en el conjunto.
* Estrategias de **refinamiento**: con el objetivo de limitar la generación de cláusulas.

#### Ejemplos y propiedades

|  |  |  |
| --- | --- | --- |
| Estrategia | Correcta | Completa |
| Saturación | Si | Si |
| Lineal | Si | Si |
| Input | Si | No, en el caso general |
| Dirigida | Si | Si, si el conjunto soporte es satisfacible |
| Ordenada | Si | No |

**Correcta**: [] se deduce sólo si el conjunto de cláusulas S es insatisfacible:

[] *!* S insatisfacible

**Completa**: Si el conjunto de cláusulas S es insatisfacible, se deduce []:

S insatisfacible *!* []

#### Resolución lineal

Aplica las siguientes reglas de simplificación:

* **Eliminación de cláusulas idénticas**: Es posible deducir por resolución [] a partir de un conjunto de cláusulas *C* sii es posible deducir por resolución [] a partir de *C* tras eliminar cláusulas idénticas.
* **Eliminación de cláusulas con literales puros**: Un literal *L* de un conjunto de cláusulas es puro si y sólo si no existe ningún otro literal complementa- rio *¬L0* en el conjunto tal que *L* y *L0* son unificables.

Una cláusula con un literal puro es inútil de cara a la refutación porque el literal nunca podrá eliminarse en el proceso de resolución.

*o*

* **Eliminación de cláusulas tautológicas**: Una cláusula tautológica es ver- dad para cualquier interpretación. P.ej., *P*(*x*) *\_¬P*(*x*) *\_Q*(*y*).

Aplica la siguiente regla de refinamiento:

* **Derivación lineal**: Una derivación lineal de *Cm* a partir de *C*1*,... .,Cn* es una secuencia *C*1*,...,Cn,Cn*+1*,...,Cm* tal que:

*{ }*

* + *Cn*+1 es el resolvente de

dos cláusulas *2 {C*1*,...,Cn}* (cláusulas de cabecera) y

* + Para todo *i > n* + 1, *Ci* es el resolvente de

*Ci-*1 con otra cláusula *Cj, j < i*.

**Resolución lineal**: Sólo genera derivaciones lineales

La resolución lineal es **completa**: Un conjunto de cláusulas *C* es insatisfacible sii existe una refutación lineal de *C*.

* Podemos restringir las derivaciones posibles a derivaciones lineales.



# Cápitulo 4

**Introducción a la Programación Lógica**

4.1 Tema 4.1: [**74**](#_bookmark61)

* + 1. Estrategia de resolución SLD [74](#_bookmark62)

Ejercicio: Estrategia de resolución SLD [74](#_bookmark62)

* + 1. Árbol de derivación [74](#_bookmark63)

Ejemplo: Árbol de derivación SLD [75](#_bookmark64)

4.2 Tema 4.2: [**75**](#_bookmark65)

* + 1. Programación Declarativa [75](#_bookmark66)
    2. Programación Funcional [76](#_bookmark67)

Cálculo Lambda [76](#_bookmark67)

Haskell [77](#_bookmark67)

Booleanos en cálculo lambda [77](#_bookmark68)

Características y Ventajas [78](#_bookmark69)

* + 1. Programación Lógica [79](#_bookmark70)

Cláusulas Horn [79](#_bookmark70)

Prolog [80](#_bookmark71)

Hay algo más ... [80](#_bookmark72)

... mucho más (CLP) [81](#_bookmark73)

* + 1. UTD HackReason [82](#_bookmark74)

## Estrategia de resolución SLD.

### Estrategia de resolución SLD

SLD significa resolución **L**ineal con funciónn de **S**elección para cláusulas

**D**efinidas.

Se trata de la estrategia usada por el lenguaje de programación Prolog. Es un caso particular de la resolución general para cláusulas de Horn:

* + - * Las cláusulas objetivo no tiene literal afirmado.
      * Las cláusulas soporte tienen un literal afirmado (el primero). Dado un conjunto inicial de cláusulas de Horn *{C*1*,...,Ci,...,Cn}*:
      * Existe una secuencia (*derivación*) *< Ci,Cn*+1*,...,* [] *>* tal que:
      * *Cn*+1 es el resolvente de la cláusula objetivo *Ci* y una cláusula soporte.
      * *Ck*, con *k > n* + 1, es el resolvente de *Ck-*1 con una cláusula soporte.
      * Cada paso de resolución es de la forma *L\_C, ¬L\_C0 ! C \_C0*. Si y solo si el conjunto inicial es insatisfacible.

#### Ejercicio: Estrategia de resolución SLD

1. Obtener la forma clausular y resolver usando la estrategia SLD:

*8ls Concatenar*([ ]*, ls, ls*)

*8x, xs, ls, ns Concatenar*(*xs, ls, ns*) *! Concatenar*([*x|xs*]*, ls,* [*x|ns*])

*9la, lb Concatenar*(*la, lb,* [1*,* 2*,* 3*,* 4])

1. Comprobar que es equivalente al programa Prolog:

1 concatenar([],Ls,Ls).

2 concatenar([X|Xs],Ls,[X|Ns]) :- concatenar(Xs,Ls,Ns).

3

4 ?- concatenar(La, Lb, [1,2,3,4]).

### Árbol de derivación

La estrategia de resolución SLD, desde el punto de vista de la evaluación de un programa Prolog, es un árbol de derivación:

1. La cabeza es la consulta *Q*.
2. Los nodos hijos son el cuerpo (resolvente) de las cláusulas cuya cabeza unifica con la consulta.
3. Si el resolvente es [] la consulta tiene éxito:
   * Si la consulta tiene variables la solución es la composición de las sus- titución calculadas (en las aristas).
4. En caso contrario, se “resuelve” un literal del resolvente.
   * Se crea un hijo por cada cláusula que unifica con dicho literal.
   * Se añade el cuerpo de la clausula al resolvente.
   * Se vuelve al punto 3.

#### Ejemplo: Árbol de derivación SLD

concatenar(La,Lb,[1,2,3,4])

La=[], Lb=[1,2,3,4] La=[1|Xs]

[] concatenar(Xs,Lb,[2,3,4])

Xs=[]

La=[1], Lb=[2,3,4]

Xs=[2|Xs']

[] concatenar(Xs',Lb,[3,4])

Árbol de *derivación* de:

Xs'=[]

La=[1,2], Lb=[3,4]

Xs'=[3|Xs'']

 [] concatenar(Xs'',Lb,[4])

1 concatenar([],Ls,Ls).

Xs''=[]

La=[1,2,3], Lb=[4]

X''=[4|Xs''']

2 concatenar([X|Xs],Ls,[X|Ns]) :-

3 concatenar(Xs,Ls,Ns).

4

5 ?- concatenar(La, Lb, [1,2,3,4]).

[] concatenar(Xs''',Lb,[]) X'''=[]

La=[1,2,3,4], Lb=[]

[]

## Prolog & la Programación Declarativa.

### Programación Declarativa

#### Programación Declarativa

¿Que es la programación declarativa?

* + - * Paradigma de programación diferente a la imperativa (**R**) o la orientada a objetos (**Java**).
      * Los programas especifican las propiedades del problema a resolver.
      * La ejecución del programa consiste en “encontrar” la(s) solución(es).

Asignación Destructiva

1 def sumalista(lista):

2 sum = 0

3 for elem in lista:

4 sum += elem

5 return sum

6

Recursión

1 sumaLista :: [Int] -> Int

2 sumaLista [] = 0

3 sumaLista (n:list) = n + (sumaLista list)

4

5

6

7 print(sumalista([1,2,3,4]))7 main = print (sumaLista [1,2,3,4])

Ejemplos de programación declarativa:

Lenguajes funcionales: Haskell, Scala by EPFL. Lenguajes lógicos: Prolog, ASP, Logica by Google. Lenguajes algebraicos: Maude, SQL. etc...

### Programación Funcional

La programación funcional esta basada en funciones matemáticas.

Función: Una función es una regla de correspondencia entre dos conjuntos de tal manera que a cada elemento del primer conjunto le corresponde uno y sólo un elemento del segundo conjunto.

Cualquier función computable puede expresarse y evaluarse con el cálculo lamb- da.

El cálculo lambda fue usado por Church para resolver el Entscheidungsproblem (1936):

* + - * No hay un algoritmo que determine si dos expresiones lambda arbitraria son equivalentes.

#### Cálculo Lambda

Introducción al cálculo lambda.

Reglas de formación de las expresiones lambda (*l* -expresiones):

* + - * x es una *l* -expresión si x es una variable.
      * (*l* x.t) es una *l* -expresión (función) si t una expresión y x una variable.
      * (t s) es una *l* -expresión (aplicación) si t y s son expresiones.

Evaluando *l* -expresiones

Función identidad aplicada al 3: ((*l* x.x) 3) *⌘* 3

Función suma aplicada al 2 y el 3: (((*l* x.*l* y.x+y) 2) 3) *⌘* ((*l* y.2+y) 3) *⌘* (2+3)

Función identidad aplicada a la suma: ((*l* x.x) (*l* x.*l* y.x+y)) *⌘* (*l* x.*l* y.x+y)

#### Haskell

**Programación Funcional: Haskell en honor a Haskell Curry (1900-1982).**

*Currificación:* Transforma *f* : (*X*1 *⇥X*2 *⇥··· ⇥Xn*) *! Z* en una secuencia de funciones con un único argumen- to:

*curry*( *f* ) : *X*1 *! X*2 *! · · · ! Xn ! Z*. [haskell.org](https://www.haskell.org/)

Ejemplos de funciones en Haskell

1 suma :: Int -> Int -> Int % suma a y b

2 suma a b = a + b % version con dos argumentos

3 suma' = \a -> \b -> a + b % version currificada

4 sucesor :: Int -> Int % sucesor de a

5 sucesor = suma 1

6 aplica :: (Int -> Int) -> [Int] -> [Int]

7 aplica \_ [] = [] % caso base

8 aplica f (n : list) = ((f n) : (aplica f list)) % caso recursivo

9

10 main = print (aplica sucesor [1,3,4]) % imprime [2,4,5]

#### Booleanos en cálculo lambda

Definición de los booleanos en *l* :

* + - * + true: *l* x.*l* y.x
        + false: *l* x.*l* y.y
        + If-then-else: *l* x.*l* y.*l* z.x y z

Evaluación

If-then-else True P Q *⌘* (*l* x.*l* y.*l* z. x y z) (*l* x.*l* y.x) P Q *⌘* (*l* x.*l* y.x) P Q *⌘* P

Implementación usando Haskell:

1 true = \x -> \y -> x

2 false = \x -> \y -> y

3 if\_then\_else = \x -> \y -> \z -> x y z

4

5 k = if\_then\_else true 3 2 % ¿cuánto vale k?

.. ahora, basadas en estas expresiones definimos And, Or y Not:

* + - * + And: *l* p.*l* q.p q false *⌘ l* p.*l* q.p q (*l* x.*l* y.y)
        + Or: *l* p.*l* q.p true q *⌘ l* p.*l* q.p (*l* x.*l* y.x) q
        + Not: *l* p.p false true *⌘ l* p.p (*l* x.*l* y.y) (*l* x.*l* y.x)

Evaluación

And True False *⌘* (*l* p.*l* q.p q (*l* x.*l* y.y)) (*l* x1.*l* y1.x1) (*l* x2.*l* y2.y2) *⌘*

(*l* x1.*l* y1.x1) (*l* x2.*l* y2.y2) (*l* x.*l* y.y) *⌘* (*l* x2.*l* y2.y2) *⌘* False Or True False *⌘* (*l* p.*l* q.p (*l* x.*l* y.x) q) (*l* x1.*l* y1.x1) (*l* x2.*l* y2.y2) *⌘*

(*l* x1.*l* y1.x1) (*l* x.*l* y.x) (*l* x2.*l* y2.y2) *⌘* (*l* x.*l* y.x) *⌘* True

Not True *⌘* (*l* p.p (*l* x.*l* y.y) (*l* x.*l* y.x)) (*l* x1.*l* y1.x1) *⌘*

*.. . ⌘* (*l* x.*l* y.y) *⌘* False

Implementación usando Haskell (cont.):

6 my\_and = \x -> \y -> x y false

7 my\_or = \x -> \y -> x true y

8 my\_not = \x -> x false true

9

10 k = if\_then\_else (my\_and true false) 3 2 % ¿cuánto vale k?

**Alternativa** para And, Or y Not:

* And: *l* p.*l* q.p q p
* Or: *l* p.*l* q.p p q
* Not: *l* p.*l* x.*l* y.p y x ¿Cuantos argumentos tiene?

Evaluación

#### Sin hacer :-), deberes para casa

Implementación usando Haskell:

6 {-# LANGUAGE Rank2Types #-}

7 type CB = forall a . a -> a -> a % Ojo, requiere tipos

8

9 my\_and :: CB -> CB -> CB

10 my\_and = \p -> \q -> p q p

11 my\_or :: CB -> CB -> CB

12 my\_or = \p -> \q -> p p q

13 my\_not :: CB -> CB % tiene 1 argumento !!!

14 my\_not = \p -> \x -> \y -> p y x

#### Características y Ventajas

**Características:**

Evaluación de funciones vs. ejecución de instrucciones (recursión vs. iteración).

El valor de una función sólo depende de sus argumentos (siempre se obtiene el mismo valor, transparencia referencial).

Las funciones son “ciudadanos de primera clase” (argumentos y/o valores)

#### Ventajas:

Código más limpio, conciso y expresivo.

Sin efectos secundarios, al ser el estado inmutable.

* + Adecuado para sistemas concurrentes/paralelos. Permite verificación formal y demostración automática.

Concatenar listas

1 concatenar :: [a] -> [a] -> [a] % declaración de tipos

2 concatenar [] list = list % caso base

3 concatenar (x:xs) list = (x: (concatenar xs list)) % recursión

4

5 k = concatenar [1,2] [3,4] % ¿cuánto vale k?

### Programación Lógica

La programación lógica esta basada en lógica de 1*er* orden (LPO).

Predicados: Un predicado es una afirmación sobre propiedades de un objeto y/o una relación entre dos o más objetos.

Dado un conjunto de fórmulas inferimos nuevo conocimiento. P.ej.:

*T* [*8x* ( *Hombre*(*x*) *! Mortal*(*x*) )*, Hombre*(*socrates*)] *`Mortal*(*socrates*)

1. Se reescribe como el siguientes conjunto de cláusulas:

*{ Mortal*(*x*) *\_¬Hombre*(*x*)*, Hombre*(*socrates*)*, ¬Mortal*(*socrates*) *}*

donde el consecuente, *Mortal*(*socrates*), están negado.

1. Si es **insatisfacible**, significa que hay consecuencia lógica.
2. Se resuelve aplicando método de Robinson con estrategia SLD.

#### Cláusulas Horn

Introducción a las cláusulas de Horn, definidas por Alfred Horn en 1951.

Dada una cláusula (disyunción de literales) cualquiera *L*1 *L*2 *Ln*, es una cláusula de Horn si tiene como máximo un literal positivo y esta reescrita como una implicación. Por ejemplo:

*\_ \_··· \_*

*¬p\_¬q\_··· \_¬t \_u* es una regla y se reescribe como *p^q^··· ^t ! u u* es un hecho y se reescribe como *u*

*¬p\_¬q\_··· \_¬t* sin literal positivo, es una consulta *p^q^··· ^t !*

#### Aristóteles:

*Hombre*(*x*) *Mortal*(*x*) *Hombre*(*socrates*)

*!*

*Mortal*(*socrates*)

#### Cláusulas:

*Mortal*(*x*) *Hombre*(*x*) *Hombre*(*socrates*)

*\_¬*

*¬Mortal*(*socrates*)

#### Prolog:

mortal(X) :- hombre(X). hombre(socrates).

?- mortal(socrates).

#### Prolog

*Predicados*: Transforma *f* : (*X*1 *⇥X*2 *⇥··· ⇥Xn*) *! Z* en una relación (n+1)-aria *R* y define el predicado *r* tal que:

*r*(*x*1*, x*2*,..., zn, z*)= *true !* (*x*1*, x*2*,..., zn, z*) *2 R*. [SWI-Prolog](https://www.swi-prolog.org/) [Ciao Prolog](https://www.ciao-lang.org/)

Aristóteles

1 mortal(X) :- hombre(X). % Todos los hombres son mortales.

2 hombre(socrates). % Sócrates es un hombre.

3

4 ?- mortal(socrates). % ¿Sócrates es mortal?

Al invocar la consulta ?- mortal(socrates)., Prolog contesta yes si la “pregunta” es consecuencia lógica (no en caso contrario).

Concatenar listas

1 concatenar([],Lista,Lista).

2 concatenar([X|Xs],Lista,[X|N\_Lista]) :- concatenar(Xs,Lista,N\_Lista).

3

4 ?- concatenar([1,2],[3,4],Lista). % ¿Cuánto vale Lista?

Al invocar la consulta ?- concatenar([1,2],[3,4],Lista)., Prolog devuelve la(s) sustitución(es) que la hace(n) consistente: Lista = [1,2,3,4] ?

#### Hay algo más ...

Mientras las funciones devuelven un único resultado:

k = concatenar [1,2] [3,4]

Los predicados pueden “consultarse” de diferentes formas sin cambiar el pro- grama:

?- concatenar([1,2], L, [1,2,3,4]).

devuelve:

L = [3,4] ?

Pero, hay algo más ...

?- concatenar(LA, LB, [1,2,3,4]).

devuelve 5 respuestas:

1. LA = [], LB = [1,2,3,4] ?;

2. LA = [1], LB = [2,3,4] ?;

3. LA = [1,2], LB = [3,4] ?;

4. LA = [1,2,3], LB = [4] ?;

5. LA = [1,2,3,4], LB = [] ?

Esto permite usar un único predicado para codificar/decodificar mensajes

1 char2morse('A','.-'). char2morse('B','-...'). char2morse('C','-.-.')...

2

3 ?- char2morse('B', Morse). % devuelve Morse = '-...'

4 ?- char2morse(Char, '-.-.'). % devuelve Char = 'C'

#### ... mucho más (CLP)

Constraint Logic Programming (CLP): Incorpora restricciones que nos permite expresar relaciones entre variables mediante ecuaciones:

3*x* + 5*y* = 2

(

5*x* + 3*y* = *-*2

1 sol(X,Y) :-

2 3 \* X + 5 \* Y #= 2,

3 5 \* X + 3 \* Y #= -2.

?- sol(X,Y).

X = -1, Y = 1 ?

CLP(Q) nos permite definir la relación de una hipoteca como:

1 mg(P,T,\_,\_,B) :- T #= 0, B #= P.

2 mg(P,T,R,I,B) :- T #>= 1, NP #= P + P\*I - R, NT #= T - 1, mg(NP,NT,R,I,B).

P=principal, T=time periods, R=repayment each period, I=interest rate, B=balance owing.

Podemos consultar de diferentes formas: ... y mucho más.

?- mg(1000,10,150,0.10,B).?- mg(P,10,150,0.10,0). ?- mg(P,10,R,0.10,B).

B = 203.13 ?

P = 921.68 ?

P = 6.14\*R + 0.38\*B ?

### UTD HackReason

“Una cosa más...”\*

**UTD HackReason** [**2021**](https://utd-hackreason-2021.devpost.com/project-gallery)**,** [**2022**](https://utd-hackreason-2022.devpost.com/project-gallery)**,** *.. .* **: January 14-15 (World Logic Day)**

\*“One More Thing...” is a reference to a practice that started in 1999, where Steve Jobs would leave (often quite big) announcements to the end of a presentation.