Semántica de la Lógica de Proposiciones
Lógica Computacional 2023-I, Nota de clase 2

Favio Ezequiel Miranda Perea Araceli Liliana Reyes Cabello
Lourdes Del Carmen González Huesca Pilar Selene Linares Arévalo

Facultad de Ciencias UNAM
24 de agosto de 2022

En esta nota revisaremos brevemente los conceptos semánticos más relevantes de la lógica proposicional. Todos los conceptos involucrados en esta nota fueron estudiados a detalle en el curso de Estructuras Discretas.

1. Significado de los conectivos lógicos

Veamos el significado de cada conectivo lógico:

La Negación La *negación* de la fórmula A es la fórmula $\neg A$.

- Símbolo utilizado: $\neg$
- Correspondencia con el español: No, no es cierto que, es falso que, etc.
- Otros símbolos: $\sim A,\ \overline{A}$.
- Semántica (tabla de verdad):

A	$\neg A$
1	0
0	1

La Disyunción La *disyunción* de las fórmulas A, B es la fórmula $A \vee B$. Las fórmulas A, B se llaman *disyuntos*.

- Símbolo utilizado: $\vee$
- Correspondencia con el español: o.
- Otros símbolos: $A + B,\ A \mid B$.
- Semántica (tabla de verdad):

A	B	$A \vee B$
1	1	1
1	0	1
0	1	1
0	0	0

La Conjunción La *conjunción* de las fórmulas A, B es la fórmula $A \wedge B$. Las fórmulas A, B se llaman *conyuntos*.

- Símbolo utilizado: $\wedge$
- Correspondencia con el español: y, pero.
- Otros símbolos: $A \,\&\, B,\ A \cdot B$ o también AB sin usar un operador explícito.
- Semántica (tabla de verdad):

A	B	$A \wedge B$
1	1	1
1	0	0
0	1	0
0	0	0

La Implicación La *implicación* o *condicional* de las fórmulas A, B es la fórmula $A \to B$. La fórmula A es el *antecedente* y la fórmula B es el *consecuente* de la implicación.

- Símbolo utilizado: $\to$
- Correspondencia con el español: $A \to B$ significa: si A entonces B; B, si A; A sólo si B; A es condición suficiente para B; B es condición necesaria para A.
- Otros símbolos: $A \Rightarrow B,\ A \supset B$.
- Semántica (tabla de verdad):

A	B	$A \to B$
1	1	1
1	0	0
0	1	1
0	0	1

La Equivalencia La equivalencia o bicondicional de las fórmulas A, B es la fórmula $A \leftrightarrow B$.

- Símbolo utilizado: $\leftrightarrow$
- Correspondencia con el español: A es equivalente a B; A si y sólo si B; A es condición necesaria y suficiente para B.
- Otros símbolos: $A \Leftrightarrow B,\ A \equiv B$.
- Semántica (tabla de verdad):

A	B	$A \leftrightarrow B$
1	1	1
1	0	0
0	1	0
0	0	1

2. Semántica formal de los conectivos lógicos

Definición 1 *El tipo de valores booleanos denotado* Bool *se define como* $\mathsf{Bool} = \{0, 1\}$

Definición 2 *Un estado o asignación de las variables (proposicionales) es una función*

$$\mathcal{I} : VarP \to \mathsf{Bool}$$

Dadas n variables proposicionales existen 2^n estados distintos para estas variables.

Definición 3 *Dado un estado de las variables* $\mathcal{I} : VarP \to \mathsf{Bool}$*, definimos la interpretación de las fórmulas con respecto a* $\mathcal{I}$ *como la función* $\mathcal{I}^\star : \mathsf{PROP} \to \mathsf{Bool}$ *tal que:*

- $\mathcal{I}^\star(p) = \mathcal{I}(p)$ *para* $p \in VarP$*, es decir* $\mathcal{I}^\star{\restriction_{VarP}} = \mathcal{I}$.
- $\mathcal{I}^\star(\top) = 1$
- $\mathcal{I}^\star(\bot) = 0$
- $\mathcal{I}^\star(\neg A) = 1$ *syss* $\mathcal{I}^\star(A) = 0$.
- $\mathcal{I}^\star(A \wedge B) = 1$ *syss* $\mathcal{I}^\star(A) = \mathcal{I}^\star(B) = 1$.
- $\mathcal{I}^\star(A \vee B) = 0$ *syss* $\mathcal{I}^\star(A) = \mathcal{I}^\star(B) = 0$.
- $\mathcal{I}^\star(A \to B) = 0$ *syss* $\mathcal{I}^\star(A) = 1$ *e* $\mathcal{I}^\star(B) = 0$.
- $\mathcal{I}^\star(A \leftrightarrow B) = 1$ *syss* $\mathcal{I}^\star(A) = \mathcal{I}^\star(B)$.

Obsérvese que dado un estado de las variables $\mathcal{I}$ (o también $\mathcal{I}_v$), la interpretación $\mathcal{I}^\star$ generada por $\mathcal{I}$ está determinada de manera única por lo que de ahora en adelante escribiremos simplemente $\mathcal{I}$ en lugar de $\mathcal{I}^\star$. Este abuso notacional es práctica común en ciencias de la computación y se conoce como *sobrecarga* de operadores.

La definición 3 puede reescribirse de manera más adecuada para ser implementada como sigue:

- $\mathcal{I}^\star(p) = \mathcal{I}(p)$ para $p \in VarP$
- $\mathcal{I}^\star(\top) = 1$
- $\mathcal{I}^\star(\bot) = 0$
- $\mathcal{I}^\star(\neg A) = \mathsf{notb}\ (\mathcal{I}^\star(A))$.
- $\mathcal{I}^\star(A \wedge B) = \mathsf{andb}\ (\mathcal{I}^\star(A),\ \mathcal{I}^\star(B))$.
- $\mathcal{I}^\star(A \vee B) = \mathsf{orb}\ (\mathcal{I}^\star(A),\ \mathcal{I}^\star(B))$.
- $\mathcal{I}^\star(A \to B) = \mathsf{impb}\ (\mathcal{I}^\star(A),\ \mathcal{I}^\star(B))$.
- $\mathcal{I}^\star(A \leftrightarrow B) = \mathsf{eqb}\,(\mathcal{I}^\star(A),\ \mathcal{I}^\star(B))$.

donde $\mathsf{notb} : \mathsf{Bool} \to \mathsf{Bool},\ \mathsf{andb}, \mathsf{orb}, \mathsf{impb}, \mathsf{eqb} : \mathsf{Bool} \to \mathsf{Bool}$ son las funciones booleanas que implementan las tablas de verdad. Esta definición puede verse como un compilador del lenguaje lógico de proposiciones al lenguaje booleano de bajo nivel.
El siguiente lema es de importancia para restringir las interpretaciones de interés al analizar una fórmula.

Lema 1 (Coincidencia) *Sean $\mathcal{I}_1, \mathcal{I}_2 : \mathsf{PROP} \to \mathsf{Bool}$ dos estados que coinciden en las variables proposicionales de la fórmula A, es decir $\mathcal{I}_1(p) = \mathcal{I}_2(p)$ para toda $p \in vars(A)$.*
Entonces $\mathcal{I}_1(A) = \mathcal{I}_2(A)$.

Demostración. Inducción estructural sobre A ⊣
Este lema implica que aún cuando existen una infinidad de estados, dada una fórmula A basta considerar únicamente aquellos estados que consideran las variables proposicionales de A, a saber 2^n estados distintos si A tiene n variables proposicionales.

Definición 4 (Estado modificado o actualizado) *Sean $\mathcal{I} : \mathsf{Var} \to \mathsf{Bool}$ un estado de las variables, p una variable proposicional y $v \in \mathsf{Bool}$. Definimos la actualización de $\mathcal{I}$ en p por v, denotado $\mathcal{I}[p/v]$ como sigue:*

$$\mathcal{I}[p/v\,](q) = \begin{cases} v & si\ q = p \\ \\ \mathcal{I}(q) & si\ q \neq p \end{cases}$$

El estado $\mathcal{I}[p/v]$ se conoce como un estado modificado o una actualización de $\mathcal{I}$.

El concepto de estado modificado se relaciona con el concepto de sustitución textual de una manera importante que permite eficientar el proceso de interpretación de una fórmula. Dicho proceso corresponde al siguiente

Lema 2 (Sustitución) *Sean $\mathcal{I}$ una interpretación, p una variable proposicional y B una fórmula tal que $\mathcal{I}^\star(B) = v$. Entonces*

$$\mathcal{I}(A[p := B\,]) = \mathcal{I}[p/v\,](A)$$

Demostración. Inducción sobre A. ⊣

2.1. Observaciones sobre la implementación funcional

- Estados: un estado $\mathcal{I}$ puede implementarse directamente como una función

  ```
  i :: VarP -> Bool
  ```

  o bien, dado que las listas son una estructura de datos muy conveniente en HASKELL podemos definir el tipo de Estados como

  ```
  type Estado = [VarP]
  ```

  Las variables proposicionales se pueden implementar como cadenas usando el tipo `String`.

  Observemos que se representa a un estado mediante la lista de variables proposicionales que están definidas como **verdaderas**.
  Por ejemplo si definimos $\mathcal{I}_1(p) = \mathcal{I}_1(q) = 1$ e $\mathcal{I}_1(r) = 0$ entonces implementamos a dicho estado como `i1=[p,q]`. Similarmente el estado `i2=[r,s,t]` representa al estado $\mathcal{I}_2(r) = \mathcal{I}_2(s) = \mathcal{I}_2(t) = 1$. En caso de presentarse otra variable diferente a las pertenecientes en la lista, se supondrá que su valor es cero.

- Función de interpretación: la función de interpretación $\mathcal{I}^\star$ se implementa mediante una función

  ```
  interp :: Estado -> Prop -> Bool
  ```

  especificada por: `interp` i $A = \mathcal{I}^\star(A)$ donde el estado i representa al estado $\mathcal{I}$ de la teoría.

- Estados posibles: dada una fórmula A con n-variables proposicionales, la función
  `estados :: Prop -> [Estado]` devuelve la lista con los 2^n estados distintos para A. Su definición es

  ```
  estados phi = subconj (vars phi)
  ```

  donde `vars:: Prop -> [VarP]` es la función que devuelve la lista de variables proposicionales que figuran en A (sin repetición) y `subconj :: [a] -> [[a]]` es una función que dada una lista ℓ, devuelve la lista con las sublistas de ℓ. Por ejemplo:

  ```
  subconj [1,2] = [[1,2],[1],[2],[]]
  ```

3. Conceptos Semánticos Básicos

Dada una fórmula A podemos preguntarnos ¿Cuántas interpretaciones hacen verdadera a A? las posibles respuestas llevan a las siguientes definiciones:

Definición 5 *Sea A una fórmula. Entonces*

- *Si $\mathcal{I}(A) = 1$ para toda interpretación $\mathcal{I}$ decimos que A es una tautología o fórmula válida y escribimos $\models A$.*
- *Si $\mathcal{I}(A) = 1$ para alguna interpretación $\mathcal{I}$ decimos que A es satisfacible, que A es verdadera en $\mathcal{I}$ o que $\mathcal{I}$ es modelo de A y escribimos $\mathcal{I} \models A$*
- *Si $\mathcal{I}(A) = 0$ para alguna interpretación $\mathcal{I}$ decimos que A es falsa o insatisfacible en $\mathcal{I}$ o que $\mathcal{I}$ no es modelo de A y escribimos $\mathcal{I} \not\models A$*

- *Si $\mathcal{I}(A) = 0$ para toda interpretación $\mathcal{I}$ decimos que A es una contradicción o fórmula no satisfacible.*

Similarmente si Γ es un conjunto de fórmulas decimos que:

- *Γ es satisfacible si tiene un modelo, es decir, si existe una interpretación $\mathcal{I}$ tal que $\mathcal{I}(A) = 1$ para toda $A \in \Gamma$. Lo cual denotamos a veces, abusando de la notación, con $\mathcal{I}(\Gamma) = 1$.*
- *Γ es insatisfacible o no satisfacible si no tiene un modelo, es decir, si no existe una interpretación $\mathcal{I}$ tal que $\mathcal{I}(A) = 1$ para toda $A \in \Gamma$.*

Con respecto a las tablas de verdad tenemos las siguientes observaciones:

- Una fórmula A es satisfacible si en alguna linea de la tabla de verdad A toma el valor 1. En caso contrario, es decir si en **todas** las lineas toma el valor 0 es insatisfacible (contradicción).
- Un conjunto de fórmulas Γ es satisfacible si existe alguna linea de la tabla de verdad en la que **todas** las fórmulas de Γ toman el valor 1.

Proposición 1 *Sea Γ un conjunto de fórmulas, $A \in \Gamma$, τ una tautología y C una contradicción.*

Si Γ es satisfacible entonces

- *$\Gamma \setminus \{A\}$ es satisfacible.*
- *$\Gamma \cup \{\tau\}$ es satisfacible.*
- *$\Gamma \cup \{C\}$ es insatisfacible.*

Si Γ es insatisfacible entonces

- *$\Gamma \cup \{B\}$ es insatisfacible, para cualquier $B \in \mathsf{PROP}$.*
- *$\Gamma \setminus \{\tau\}$ es insatisfacible.*

Demostración. Se deja como ejercicio. ⊣
Si Γ es satisfacible y B también nada se puede decir acerca de $\Gamma \cup \{B\}$. Análogamente si Γ es insatisfacible y B no es tautología nada se puede decir de $\Gamma \setminus \{B\}$.

Proposición 2 *Sea $\Gamma = \{A_1, \ldots, A_n\}$ un conjunto de fórmulas.*

- *Γ es satisfacible syss $A_1 \wedge \ldots \wedge A_n$ es satisfacible.*
- *Γ es insatisfacible syss $A_1 \wedge \ldots \wedge A_n$ es una contradicción.*

Demostración. Se deja como ejercicio. ⊣

3.1. Observaciones sobre la implementación funcional

Para implementar los conceptos de la definición 5 nos serviremos de la función

```
modelos :: Prop -> [Estados]
```

que dada una fórmula A devuelve la lista de todos sus modelos. Esta función se especifica como sigue:

```
modelos( A) = [i | i ∈ estados( A), interp i  A = True]
```

Es decir `modelos(`A`)` devuelve la lista con todos aquellos estados $\mathcal{I}$ tales que $\mathcal{I}(A) = 1$. Los conceptos de la definición 5 se implementan comparando las listas `estados(`A`)` y `modelos(`A`)`.

Tautología

`tautologia:: Prop -> Bool` de la definición A es tautología syss todos sus estados (interpretaciones) son modelos:

```
tautologia( A) = estados( A) == modelos( A)
```

Satisfacible en una interpretación

`satisfen:: Estado -> Prop -> Bool` donde A es satisfacible en $\mathcal{I}$ ($\mathcal{I} \models A$) syss $\mathcal{I}(A) = 1$.

```
satisfen i  A = interp i  A == True
```

Satisfacible

`satisf:: Prop -> Bool` es decir A es satisfacible syss algún estado (interpretación) es modelo, es decir, si la lista de modelos no es vacía:

```
satisf( A) = modelos( A) /= []
```

Insatisfacible en una interpretación

`insatisfen:: Estado -> Prop -> Bool` donde A es insatisfacible en $\mathcal{I}$ i.e. $\mathcal{I} \not\models A$ syss $\mathcal{I}(A) = 0$.

```
insatisfen i  A = interp i  A == False
```

Insatisfacible o contradicción

`contrad:: Prop -> Bool` donde A es insatisfacible o contradicción syss ningún estado (interpretación) es modelo, es decir, si la lista de modelos es vacía:

```
contrad( A) = modelos( A) = []
```

Extensión a conjuntos Las funciones que generan los estados y modelos para un conjunto (lista) de fórmulas Γ y aquellas que verifican si Γ es satisfacible o insatisfacible se denotan:

- `estadosConj :: [Prop] -> [Estado]`
- `modelosConj :: [Prop] -> [Estado]`
- `satisfenConj:: Estado -> [Prop] -> Bool`
- `satisfConj:: [Prop] -> Bool`
- `insatisfenConj:: Estado -> [Prop] -> Bool`
- `insatisfConj:: [Prop] -> Bool`

La implementación de estas funciones es similar al caso de las funciones para fórmulas y se deja como ejercicio.

4. Equivalencia de Fórmulas

Definición 6 *Dos fórmulas A, B son equivalentes si $\mathcal{I}(A) = \mathcal{I}(B)$ para toda interpretación $\mathcal{I}$. En tal caso escribimos $A \equiv B$.*

Es fácil ver que la relación $\equiv$ es una relación de equivalencia.

Proposición 3 *Sean A, B fórmulas. Entonces $A \equiv B$ syss $\models A \leftrightarrow B$*

Demostración. Ejercicio ⊣

Dado que ya implementamos la función `tautología` podemos definir fácilmente una función `equiv :: Prop -> Prop -> Bool` que decida si dos fórmulas son equivalentes usando la proposición anterior:

```
equiv  A   B = tautología ( A <->  B)
```

La siguiente propiedad es fundamental para el razonamiento ecuacional con equivalencias lógicas.

Proposición 4 (Regla de Leibniz) *Si $A \equiv B$ y $p \in \mathsf{ATOM}$ entonces $C[p := A] \equiv C[p := B]$*

Demostración. Inducción estructural sobre C. ⊣

Las fórmulas equivalentes son de gran importancia pues nos permiten definir algunos conectivos en términos de otros, así como simplificar fórmulas compuestas.

4.1. Algunas equivalencias lógicas

- Conmutatividad:

$$A \vee B \equiv B \vee A \qquad A \wedge B \equiv B \wedge A$$

- Asociatividad:

$$A \vee (B \vee C) \equiv (A \vee B) \vee C \qquad A \wedge (B \wedge C) \equiv (A \wedge B) \wedge C$$

- Distributividad:

$$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C) \qquad A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$$

- Leyes de la negación:
  - Doble Negación: $\neg\neg A \equiv A$
  - De Morgan: $\neg(A \vee B) \equiv \neg A \wedge \neg B \qquad \neg(A \wedge B) \equiv \neg A \vee \neg B$
  - $\neg(A \to B) \equiv A \wedge \neg B$
  - $\neg(A \leftrightarrow B) \equiv \neg A \leftrightarrow B \equiv A \leftrightarrow \neg B$.
- Propiedades de la implicación:
  - Contrapositiva: $A \to B \equiv \neg B \to \neg A$
  - Importación/Exportación: $A \wedge B \to C \equiv A \to (B \to C)$

- Eliminación de conectivos:

$$A \to B \equiv \neg A \vee B \qquad A \to B \equiv \neg(A \wedge \neg B)$$

$$A \leftrightarrow B \equiv (A \to B) \wedge (B \to A) \qquad A \leftrightarrow B \equiv (A \wedge B) \vee (\neg A \wedge \neg B)$$

$$A \vee B \equiv \neg A \to B \qquad A \wedge B \equiv \neg(A \to \neg B)$$

- Leyes simplificativas:
  - Idempotencia: $A \vee A \equiv A \qquad A \wedge A \equiv A$
  - Absorción: $A \vee (A \wedge B) \equiv A \qquad A \wedge (A \vee B) \equiv A$
  - Neutros: $A \vee \bot \equiv A \qquad A \wedge \top \equiv A$
  - Inversos:
    - Tercero Excluido: $A \vee \neg A \equiv \top$
    - Contradicción: $A \wedge \neg A \equiv \bot$
  - Dominancia: $A \vee \top \equiv \top \qquad A \wedge \bot \equiv \bot$

Veamos un ejemplo de simplificación de una fórmula mediante el uso de equivalencias lógicas y la regla de Leibniz:

$$\begin{array}{rcl} (p \vee q) \wedge \neg(\neg p \wedge q) & \equiv & (p \vee q) \wedge (\neg\neg p \vee \neg q) \\ & \equiv & (p \vee q) \wedge (p \vee \neg q) \\ & \equiv & p \vee (q \wedge \neg q) \\ & \equiv & p \vee \bot \\ & \equiv & p \end{array}$$

4.2. Eliminación de implicaciones y equivalencias

Las equivalencias lógicas que permiten eliminar los conectivos $\to, \leftrightarrow$ son de gran importancia para la llamada lógica clausular que estudiaremos más delante. Se deja como ejercicio definir recursivamente e implementar las siguientes funciones:

- `elimEquiv:: Prop -> Prop` tal que

  `elimEquiv` $A = B$ syss $B \equiv A$ y B no contiene el símbolo $\leftrightarrow$.

- `elimImp:: Prop -> Prop` tal que

  `elimImp` $A = B$ syss $B \equiv A$ y B no contiene el símbolo $\to$.

Obsérvese que estas funciones pueden definirse de diversas formas pero las más adecuadas son usando las equivalencias usuales:

$$A \leftrightarrow B \equiv (A \to B) \wedge (B \to A) \qquad A \to B \equiv \neg A \vee B$$

5. Consecuencia Lógica

Definición 7 *Sean Γ un conjunto de fórmulas y A una fórmula. Decimos que A es consecuencia lógica de Γ si para toda interpretación $\mathcal{I}$ que sastisface a Γ se tiene $\mathcal{I}(A) = 1$. Es decir si se cumple que siempre que $\mathcal{I}$ satisface a Γ entonces necesariamente $\mathcal{I}$ satisface a A. En tal caso escribimos $\Gamma \models A$.*

Nótese que la relación de consecuencia lógica esta dada por una implicación de la forma

$$\mathcal{I}(\Gamma) = 1 \Rightarrow \mathcal{I}(A) = 1$$

lo cual informalmente significa que *todo modelo de Γ es modelo de A.*
De manera que no se afirma nada acerca de la satisfacibilidad del conjunto Γ, simplemente se asume que es satisfacible y en tal caso se prueba que la fórmula A también lo es con la misma interpretación.

Obsérvese también la sobrecarga del símbolo $\models$ que previamente utilizamos para denotar satisfacibilidad $\mathcal{I} \models A$ y tautologías $\models A$.

Proposición 5 *La relación de consecuencia lógica cumple las siguientes propiedades:*

- *Si $A \in \Gamma$ entonces $\Gamma \models A$.*
- *Principio de refutación: $\Gamma \models A$ syss $\Gamma \cup \{\neg A\}$ es insatisfacible.*
- *$\Gamma \models A \to B$ syss $\Gamma \cup \{A\} \models B$.*
- *Insatisfacibilidad implica trivialidad: Si Γ es insatisfacible entonces $\Gamma \models A$ para toda $A \in \mathsf{PROP}$.*
- *Si $\Gamma \models \bot$ entonces Γ es insatisfacible.*
- *$A \equiv B$ syss $A \models B$ y $B \models A$.*
- *$\models A$ (es decir si A es tautología) syss $\varnothing \models A$ (es decir A es consecuencia lógica del conjunto vacío).*

Demostración. Ejercicio ⊣

5.1. Observaciones sobre la implementación funcional

La función `consecuencia: [Prop] -> Prop -> Bool` se especifica como:

$$\texttt{consecuencia}\ \Gamma\ A = \texttt{True}\ \text{syss}\ \Gamma \models A.$$

La implementación se sirve del principio de refutación dado en la proposición 5:

```
consecuencia gamma phi = insatisfConj ((neg phi):gamma)
```

donde `neg: Prop -> Prop` es una función que construye la negación de una fórmula.

Otra posibilidad es verificar que no existen modelos de Γ que no sean modelos de A, es decir, que la lista de modelos de Γ que no son modelos de A es vacía. Esto se puede hacer directamente usando el mecanismo de listas por comprensión de HASKELL:

```
consecuencia gamma phi = null [i | i <- estadosConj (phi:gamma),
                                   satisfenConj i gamma,
                                   not (satisfen i phi)
                              ]
```

5.2. Corrección de Argumentos Lógicos

Definición 8 *Un argumento lógico es una sucesión de fórmulas $A_1, \ldots, A_n$ llamadas premisas y una fórmula B llamada conclusión. Dicha sucesión se escribe usualmente como*

$$\begin{array}{c} A_1 \\ \vdots \\ A_n \\ \hline \therefore B \end{array}$$

Un problema central de la lógica consiste en decidir si un argumento dado es correcto o no, para lo cual la herramienta principal es la noción de consecuencia lógica.

Definición 9 *Un argumento con premisas $A_1, \ldots, A_n$ y conclusión B es lógicamente correcto o válido si la conclusión se sigue lógicamente de las premisas, es decir si $\{A_1, \ldots, A_n\} \models B$*

Esta definición deja ver de inmediato una implementación `argcorrecto :: [Prop] -> Prop -> Bool` dado que ya tenemos un programa (función) para decidir la consecuencia lógica.

```
argcorrecto [phi1,...,phin] psi = consecuencia [phi1,...,phin] psi
```

Es importante tener un método formal para decidir la correctud de argumentos del lenguaje natural los cuales pueden parecer correctos y no serlo. Veamos un ejemplo sencillo considerando el siguiente argumento:

Si hoy es viernes entonces mañana es sábado, mañana es sábado, por lo tanto hoy es viernes.

Palabras como "por lo tanto, así que, luego entonces, de forma que, etc." señalan la conclusión del argumento. La representaciónn lógica formal del argumento anterior es:

$$\begin{array}{c} v \to s \\ s \\ \hline \therefore v \end{array}$$

De manera que el argumento es correcto si y sólo si $\{v \to s, s\} \models v$. Pero la interpretación $\mathcal{I}(v) = 0$, $\mathcal{I}(s) = 1$ confirma que el argumento es incorrecto.

Y más importante resaltar que al analizar argumentos lo único que interesa es su forma lógica y no su significado, como lo muestra el siguiente ejemplo:

Si hoy tirila y Chubaka es kismi entonces Chubaka es borogrove.
Si hoy no tirila entonces hay fefos.
Más aún sabemos que no hay fefos y que
Chubaka es kismi, luego entonces Chubaka es borogrove.

Lo cual nos lleva a

$$\begin{array}{c} t \wedge k \to b \\ \neg t \to f \\ \neg f \wedge k \\ \hline \therefore b \end{array}$$

Veamos que $\{t \wedge k \to b, \neg t \to f, \neg f \wedge k\} \models b$ es un argumento correcto:
Demostración.

Mostraremos que b se sigue lógicamente de $\Gamma = \{t \wedge k \to b, \neg t \to f, \neg f \wedge k\}$, es decir que para toda interpretación $\mathcal{I}$ tal que $\mathcal{I}(\Gamma) = 1$ entonces $\mathcal{I}(b) = 1$.

Suponer $\mathcal{I}$ tal que $\mathcal{I}(\Gamma) = 1$, entonces $\mathcal{I}(t \wedge k \to b) = 1$ (**1**), $\mathcal{I}(\neg t \to f) = 1$ (**2**) y $\mathcal{I}(\neg f \wedge k) = 1$ (**3**).
Analizando las suposiciones anteriores, de (3) podemos saber que $\mathcal{I}(\neg f) = 1$ es decir $\mathcal{I}(f) = 0$ (**4**) y que $\mathcal{I}(k) = 1$ (**5**).
Utilizando (4) y siguiendo la definición de interpretación para (2) podemos concluir que $\mathcal{I}(\neg t) = 0$ forzosamente, así que $\mathcal{I}(t) = 1$ (**6**).
Por lo tanto, $\mathcal{I}(t \wedge k) = 1$ ya que tenemos (6) y (5) y entonces podemos concluir que $\mathcal{I}(b) = 1$ de (1).
De lo anterior hemos **construido una interpretación** tal que b es consecuencia lógica de Γ y por tanto el argumento es correcto. ⊣

5.3. Métodos semánticos para demostrar la correctud de argumentos

Para mostrar la correctud del argumento lógico $A_1, \ldots, A_n \ / \ \therefore \ B$ mediante interpretaciones, se puede proceder de alguna de las siguientes dos formas:

- Método directo: probar la consecuencia $A_1, \ldots, A_n \models B$.
- Método indirecto (refutación): probar que el conjunto $\{A_1, \ldots, A_n, \neg B\}$ es insatisfacible.

Por supuesto que existen otros métodos para mostrar la correctud de un argumento lógico, que resultan más adecuados para la automatización, como son el cálculo exhaustivo de una tabla de verdad (sumamente ineficiente), el uso de tableaux (recordado brevemente en clase) o el método de resolución binaria que veremos más adelante.

5.4. Algunos argumentos correctos relevantes

$$\frac{\begin{array}{c}A\\A \to B\end{array}}{\therefore\ B}\ \textsc{Modus Ponens} \qquad \frac{\begin{array}{c}\neg B\\A \to B\end{array}}{\therefore\ \neg A}\ \textsc{Modus Tollens}$$

$$\frac{\begin{array}{c}A \to B\\B \to C\end{array}}{\therefore\ A \to C}\ \textsc{Silogismo hipótetico} \qquad \frac{\begin{array}{c}A \to C\\B \to C\end{array}}{\therefore\ A \vee B \to C}\ \textsc{Prueba por casos}$$

$$\frac{\begin{array}{c}A\\B\end{array}}{\therefore\ A \wedge B}\ \textsc{Introd.}\ \wedge \qquad \frac{A \wedge B}{\therefore\ A}\ \textsc{Elim.}\ \wedge \qquad \frac{A \wedge B}{\therefore\ B}\ \textsc{Elim.}\ \wedge$$

$$\frac{A}{\therefore\ A \vee B}\ \textsc{Introd.}\ \vee \qquad\qquad \frac{B}{\therefore\ A \vee B}\ \textsc{Introd.}\ \vee$$

$$\frac{\begin{array}{l} A \vee B \\ \neg B \vee C \end{array}}{\therefore\ A \vee C}\ \textsc{Resolución binaria} \qquad\qquad \frac{\begin{array}{c} A \to B_1 \\ C \to B_2 \\ A \vee C \end{array}}{\therefore\ B_1 \vee B_2}\ \textsc{Dilema constructivo}$$

$$\frac{A \to B_1 \qquad C \to B_2 \qquad \neg B_1 \vee \neg B_2}{\therefore\ \neg A \vee \neg C}\ \textsc{Dilema destructivo}$$

6. Ejercicios

Verifique si los siguientes argumentos son correctos. Si no lo son, dé un modelo contraejemplo.

1. $p \to q,\ q \to p \wedge r,\ r\ /\ \therefore p$
2. $m \wedge \neg b \to j,\ f \vee s \to m,\ b \to t,\ f \to \neg t,\ \neg j\ /\ \therefore \neg f$
3. $p \wedge r \to s,\ q \to p,\ \neg r \to \neg t,\ q,\ \neg s\ /\ \therefore \neg t \vee w$
4. $p \to q \vee \neg r,\ q \to p \wedge r\ /\ \therefore\ p \to r$
5. $p \vee q,\ q \to r,\ p \wedge s \to t,\ \neg r,\ \neg q \to u \wedge s\ /\ \therefore\ t$
6. $r \to p,\ \neg p \vee q,\ s \to p \wedge r,\ \neg p \wedge \neg r \to s \vee t,\ \neg q\ /\ \therefore\ t$
7. $p \vee (q \wedge r),\ \neg(p \wedge q)\ /\ \therefore\ r$
8. $p \vee q,\ p \to \neg q,\ p \to r\ /\ \therefore\ r$
9. $\neg p \vee q \to r,\ s \vee \neg q,\ \neg t,\ p \to t,\ \neg p \wedge r \to \neg s\ /\ \therefore\ \neg q$
10. $\neg(l \to d),\ d \to \neg h \wedge \neg b\ /\ \therefore\ d \to \neg l$