Semántica de la lógica proposicional

Tablas de verdad

Una forma alternativa (y válida) de calcular los posibles valores de verdad de una frase de $PROP$ para las combinaciones de valores de verdad de las variables que sean subfórmula de la frase (en el caso extremo átomos p_i), es a través de las tablas de verdad.

Por ejemplo, si quisiéramos comprobar la equivalencia de las frases:

$\alpha =$ "Si trabajo más horas, me pagan extra"
$\beta =$ "Me pagan extra o no trabajo más horas"

Podríamos realizar el siguiente modelado:

$p_0 =$ "trabajo más horas"
$p_1 =$ "me pagan extra"

Por el cual las frases α y β se podrían escribir como:

$$\alpha = p_0 \to p_1$$
$$\beta = p_1 \vee \neg p_0$$

Probar que $\alpha \; eq \; \beta$, o probar que para toda valuación v se cumple que $v((p_0 \to p_1) \leftrightarrow (p_1 \vee \neg p_0)) = 1$, se reduce a determinar si para todas las combinaciones posibles de valores de verdad de p_0 y p_1, el valor de verdad de la frase $(p_0 \to p_1) \leftrightarrow (p_1 \vee \neg p_0)$ siempre es uno.

p_0	p_1	$p_0 \to p_1$	$\neg p_0$	$p_1 \vee \neg p_0$	$(p_0 \to p_1) \leftrightarrow (p_1 \vee \neg p_0)$
0	0	1	1	1	1
0	1	1	1	1	1
1	0	0	0	0	1
1	1	1	0	1	1
					||
					Tautología

Ejercicio:

- Demuestre que $\models (\alpha \vee \beta) \wedge \neg\alpha \to \beta$

Conjunto completo de conectivos

Un conjunto de conectivos C es completo si podemos expresar cualquier función de verdad con los conectivos de C. En la primera parte del curso ya habíamos visto que con compuertas OR y NOT podíamos construir cualquier circuito lógico, y es exactamente la misma idea.

Se puede demostrar que $\{\vee, \neg\}$ es un conjunto completo de conectivos (por ejemplo construyendo el conectivo $\wedge$ y escribiendo cualquier función lógica como suma de productos canónicos), y también son completos los conjuntos $\{\wedge, \neg\}$, $\{\bot, \to\}$ y $\{\neg, \to\}$.

Formas de razonamiento

Hasta el momento, mediante el cálculo del valor de verdad de una fórmula, sea a través de las definiciones recursivas de las valuaciones o a través de tablas de verdad, podemos dar una justificación semántica del razonamiento.

Asumamos que ya hemos probado de esta forma las dos siguientes equivalencias:

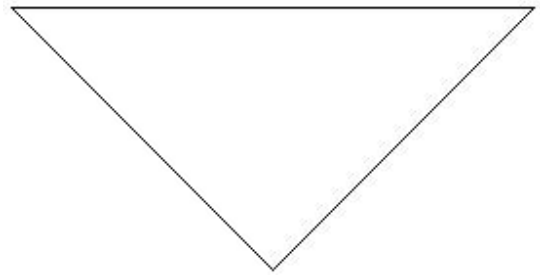
1. $\alpha \vee \beta \ eq \ \beta \vee \alpha$
2. $\alpha \to \beta \ eq \ \beta \vee \neg \alpha$

Para probar la equivalencia $\alpha \to \beta \ eq \ \neg \alpha \vee \beta$ podríamos apelar a las equivalencias ya demostradas, utilizándolas como "reglas". En este caso estaríamos dando una justificación sintáctica del razonamiento.

Veremos en lo sucesivo, la forma de construir pruebas formales basadas en la forma de la prueba. La idea es subdividir una prueba en pruebas más simples (estrategia: *"divide & conquer"*) donde los pasos simples son *"reglas prefabricadas"*.

Estos procesos se pueden abstraer al punto de poder programarse, y hacer que una máquina *"razone"*.

$$\alpha \vee \beta \leftrightarrow \beta \vee \alpha \qquad (\alpha \to \beta) \leftrightarrow \beta \vee \neg \alpha$$



$$(\alpha \to \beta) \leftrightarrow \neg \alpha \wedge \beta$$

Las pruebas tendrán la forma de árboles, donde las hojas serán las hipótesis de la prueba y la raíz será la conclusión de la misma. El camino de las hojas a la raíz se hará mediante reglas de construcción, que serán de dos tipos:

- Reglas de introducción (indican cómo probar una fórmula para un conectivo)
- Reglas de eliminación (indican cómo usar un conectivo)

Por ejemplo, ya demostramos la siguiente consecuencia lógica: $\alpha, \beta \models \alpha \wedge \beta$

Podríamos definir una regla entonces, que nos permitiera demostrar una fórmula del tipo $\alpha \wedge \beta$ a partir de ciertas hipótesis $\delta_1, \delta_2, ..., \delta_n$, demostrando por un lado α a partir de las hipótesis, y por otro lado β a partir de las mismas hipótesis, y finalmente nuestra regla nos permitiría afirmar que es cierto $\alpha \wedge \beta$ (a esta regla la llamaríamos introducción de $\wedge$, ya que nos indica cómo probar una conjunción).

$$\frac{\begin{array}{cc}\delta_1, \delta_2, ..., \delta_n & \delta_1, \delta_2, ..., \delta_n \\ \alpha & \beta\end{array}}{\alpha \wedge \beta} \text{ Introducción de } \wedge$$