EL (GRAN) JUEGO DE LA LÓGICA #2

Segundo juego: tablas

Inventamos ahora una propiedad de las fbf. Por la presente, **queda establecido que las fbf *valen***, tienen valor o valores. Qué significa esto, o qué quiere decir "valor" en este contexto es algo que ahora no nos importa. Simplemente, igual que las fbf pueden tener una longitud u otra, pueden incluir o no un determinado operador, pueden ser bonitas o feas etcétera, decidimos ahora que las fbf pueden tener un valor. Los valores posibles en este juego son 1 y 0. No busquen vds. ningún significado a estos dos símbolos. Simplemente, de acuerdo con sus posibles valores, clasificamos cualquier fbf en una de estas tres categorías (y solo una):

1. **Tautologías** (o "fórmulas tautológicas"): son aquellas fbf's que *siempre* valen 1 (signifique esto lo que signifique). Por ejemplo, aunque esto sólo lo sé yo de momento, y vds. no, **p → p**

2. **Contradicciones** (o "fórmulas contradictorias"): son las fbf's cuyo valor es *siempre* 0. Por ejemplo, **p ∧ ¬ p** (ya veremos por qué)

3. **Contingencias** (o "fórmulas contingentes"): son fbf's que, a veces valen 1, y a veces valen 0. **¬ p → q**, por ejemplo (se verá luego).

Objetivo de este juego

Decidir, dada una fbf cualquiera, a qué categoría pertenece. Para averiguarlo, habremos de elaborar, siguiendo unas reglas concretas, una **tabla** de unos y ceros que especifique, de forma inequívoca, todos los valores posibles de la fórmula. Una tabla es una matriz con filas y columnas.

Reglas de este juego

1. Los valores finales de una fbf dependen de
a. Las variables *DIFERENTES* que contenga
b. Los diversos operadores de la fbf
Así, **[(p → q) ∧ p] → q** arrojará resultados distintos, en número y naturaleza, de **(r → q) ∧ p**

2. Los valores finales de la fbf (y que determinarán a qué categoría pertenece) son los que aparecerán, del modo que especificaremos luego, verticalmente bajo el operador principal de la fbf.

Veamos la tabla de la siguiente fbf (que voy a hacer yo, porque vosotros aún no sabéis, pero en seguida lo podréis comprobar por vuestra cuenta):

(	**p**	**∧**	**q**	)	**→**	¬	**p**
	1	1	1		**0**	0	1
	1	0	0		**1**	0	1
	0	0	1		**1**	1	0
	0	0	0		**1**	1	0

el resultado final es la columna 0 1 1 1 que aparece bajo la flecha, que es el operador principal. Se trata, por tanto, de una contingencia.

3. El número de columnas (de valores, de unos y ceros) de que constará la tabla es igual al de símbolos principales (operadores + variables, repetidos o no). En la tabla anterior son seis columnas, porque los paréntesis no cuentan al tratarse de símbolos auxiliares.

4. El número de filas debe calcularse de antemano, de acuerdo con esta fascinante fórmula matemática:

$nf = 2^{nvd}$, donde:

- "nf" es el número de filas que tendrá la tabla
- "nvd" es el número de variables *diferentes* que aparecen en la fbf que estamos "tabulando" o analizando. No el número total de variables, sino las que sean distintas.
- (2 es el número de valores posibles de nuestro juego: 1, 0)

Como podéis ver, el número de filas de la tabla es cuatro: aunque hay tres variables, sólo son dos las diferentes, de modo que el cálculo es como sigue: $nf = 2^{nvd} = 2^2 = 4$

5. Una vez conocidos estos datos, vamos a escribir las columnas que corresponden a cada una de las variables, de acuerdo con el siguiente plan:

a. Localizamos la primera variable, y colocamos, verticalmente, la **mitad** (de nf, el número de filas calculado con la fórmula matemática) de "unos" y la otra **mitad** de "ceros". En este caso, ponemos bajo la **p** dos "unos" y dos "ceros" (la mitad de cuatro es dos: usen una calculadora si no me creen). Tal que así:

(	**p**	∧	**q**	)	→	¬	**p**
	1						
	1						
	0						
	0						

Debe asignarse el mismo orden de valores a todas las apariciones de **p**:

(	**p**	∧	**q**	)	→	¬	**p**
	1						**1**
	1						**1**
	0						**0**
	0						**0**

b. Localizamos la siguiente variable distinta, que es la **q**, y le asignamos la **mitad de la mitad** (hagan los cálculos: nf es 4, la mitad es 2 y la mitad de la mitad es 1) de "unos" y la **mitad de la mitad** de "ceros". Completamos, con la misma pauta, la columna hasta el final:

(	**p**	∧	**q**	)	→	¬	**p**
	1		**1**				1
	1		**0**				1
	0		**1**				0
	0		**0**				0

Si se repitiese la **q**, colocaríamos la misma ordenación de "unos" y "ceros"[1]. ¿Y si existiese otra variable distinta más? Habría que seguir el juego: escribir bajo ella la

[1] Esto es pura combinatoria: queremos recoger, en cada fila, *todas* las combinaciones posibles de los valores de las variables: una fila en la que las dos variables valgan 1; otra en la que la primera valga 1 y la segunda cero, otra a la inversa, y otra en la que las dos variables valgan 0. No hay más combinaciones posibles (con dos variables distintas), ni tampoco menos. Por ello es necesaria una tabla con cuatro filas, ni una más (entonces se habría repeticiones innecesarias), ni una menos (en cuyo caso perderíamos alguna posible combinación).

mitad de la mitad de la mitad de "unos" y luego de "ceros" hasta completar la columna y terminar con todas las variables.[2]

c. Ahora ya sólo nos queda colocar los valores bajo los operadores. Y es el momento de revelarles un secreto: aunque este juego es convencional y arbitrario, estaba pensando en este asunto cuando bauticé a ciertos símbolos como operadores[3]: los llamamos operadores porque, efectivamente, expresan una operación. Una operación muy similar –pero distinta– a los operadores aritméticos como +, – etc, aunque sobre una variedad de valores sumamente limitada: sólo dos (el uno y el cero, ¿recuerdan? Las matemáticas utilizan infinitos valores. Nuestro juego es "bivalente" nada más). Cada operador efectúa una operación diferente sobre los valores de su (o sus dos) argumentos. De la misma forma que, en matemáticas, multiplicar no es lo mismo que restar, en lógica no es lo mismo "flechear" que "capiruchear"[4]. **Nos queda definir en qué consiste cada operación lógica**.

Para ello, vamos a echar mano de una elegante categoría de símbolos auxiliares, las metavariables. A partir de este momento, **una metavariable va a representar una fbf completa cualquiera**. Es decir, allá donde vean, por ejemplo, Q, entiendan "una fbf cualquiera", por ejemplo **p → ¬ r**, o **¬ s,** y si leen R entiendan "otra fbf cualquiera, es posible que hasta idéntica a la anterior, o no". Es para ahorrar tinta y esfuerzos.

Ah, pero, ¿y si yo escribo **¬ P**? Vale, vale, estará mal formada, porque las reglas de formación no mencionan a las metavariables, pero no seamos tan quisquillosos: ¿no tendrá, a pesar de todo, algún sentido? Claro, no será una fbf, pero sí un *esquema*, un *modelo*, una *plantilla* de determinado tipo de fbf. Piensen un momento y determinen cuáles de entre las siguientes fbf's corresponden a, o casan con, o parecen familia de **¬ P**:

1 **p ↔ r**
2 **¬ s**
3 **p → ¬ r**
4 **¬ (p → q)**

¡Claro! Las fbf's 2 y 4 son otros tantos ejemplos de **¬ P**, pero no las fbf's 1 y 3. Con este esquema de fórmula (llamaremos a este tipo de cosa "metafórmula" o metafbf), lo único que hacemos es especificar y referenciar fbf's cuyo principal operador es el monádico, sin especificar lo demás. Así, la metafbf **P ↔ Q** engloba a todas las (infinitas) fbf's cuyo principal operador es la flecha doble. Con este sistema podemos referirnos a clases de fbf's con la estructura que queramos y con el nivel de detalle que queramos, desde ninguno en absoluto (pinten simplemente P, por ejemplo) pasando por los dos ejemplos anteriores, hasta cualquier nivel de detalle (por ejemplo, **¬ (P → Q)** querrá decir "fbf's que consistan en un operador monádico principal que afecta a dos fbf's enlazadas con una flecha").

Volvamos ahora a las tablas. Recuerden que nos falta especificar qué tipo de operación es cada una de las correspondientes a los operadores. Voy a definir estas operaciones con el mismo espíritu de siempre: como a mí me dé la gana, porque esto es

[2] Si hubiera una tercera variable distinta, entonces el número de filas de la tabla no sería 4, sino 8, y, en ese caso, la 'mitad de la mitad de la mitad' sería 1 de nuevo. La columna que corresponde a la última variable distinta será siempre una serie de unos y ceros alternados.

[3] Otros denominan a estos símbolos "conectivas", precisamente porque permiten unir dos fbf (o una). Por cierto, y ya que estamos, los nombres de "díadico" y "monádico", aunque convencionales, no son casuales: los operadores diádicos se llaman así porque unen *dos* fbf, y el monádico debe su nombre a su *único* argumento.

[4] Estos nombres no son serios. Habrá que cambiarlos por otros más impresionantes. Más tarde.

convencional y yo hago lo que quiero con mi juego (si no les gusta, créense su propio juego):

¬	P
0	1
1	0

Ya está. ¿Cómo? ¿Qué no está claro? Expliquemos más la cosa. Observen la columna que aparece bajo el operador, y compárenla con la que aparece bajo su argumento (recuerden que, al tratarse de una metavariable, se trataría de una fbf de cualquier complejidad). Esta tablita está indicándonos, ni más ni menos, el comportamiento del operador "ángulo recto", o el resultado de "angulorectear" los valores 1 y 0: cuando deban calcular los resultados de este operador (no de otro), recuerden que arroja un resultado de 0 si su argumento vale 1 (como se ve en la primera fila) y de 1 si su argumento vale 0 (en la segunda fila). Así, que ya lo saben: cada vez que tropiecen con un operador "ángulo recto" simplemente determinen cuál es su argumento e inviertan sus valores. Eso es lo que hace este operador: llevar la contraria. Los valores así calculados, aunque aparecen sólo debajo del operador, son el resultado de *toda* la fórmula dominada por el operador.

Los otros operadores efectúan otras operaciones (por eso son *otros* operadores, y utilizamos símbolos distintos). Pero son diádicos, lo que quiere decir que se aplican a *pares* de valores, y no a valores individuales como el monádico. O sea, que funcionan no como el operador matemático √ (eso lo hace nuestro monádico), por ejemplo, sino como la multiplicación o la suma: siempre se multiplican o suman *dos* números. Y, dado que nuestra lógica es bivalente, debemos considerar cuatro (y no dos, como con el monádico) casos en la definición de su comportamiento. Vean, vean:

	P	∧	Q
(a)	1	**1**	1
(b)	1	**0**	0
(c)	0	**0**	1
(d)	0	**0**	0

P	∨	Q
1	**1**	1
1	**1**	0
0	**1**	1
0	**0**	0

P	→	Q
1	**1**	1
1	**0**	0
0	**1**	1
0	**1**	0

P	↔	Q
1	**1**	1
1	**0**	0
0	**0**	1
0	**1**	0

Estas tablas definen las cuatro operaciones diádicas cuando sus argumentos valen (a) los dos 1, (b) el primero 1 y el segundo 0, (c) el primero 0 y el segundo 1 y (d) los dos valen 0.

Pueden recordar fácilmente el operador "capirucho" reteniendo simplemente que su resultado es 0 excepto cuando sus dos argumentos valen 1. O multiplicando los valores de sus argumentos[5].

El operador "uve" produce siempre 1, excepto cuando sus argumentos valen, ambos, 0. Y se comporta casi como una suma (salvo en el caso primero, donde debería valer 2, pero nosotros no tenemos ese símbolo)[6].

[5] Razón por la cual a veces se denomina a este operador "producto (o multiplicación) lógico". Pero recuerden, *no* es una multiplicación, porque es un operador lógico, no aritmético, aunque se *comporta como si* lo fuese.

[6] Exacto. También se llama a este operador "suma lógica".

El operador "flecha" sólo vale 0 cuando su primer argumento vale 1 y el segundo 0, y uno en los demás casos[7].

El operador "doble flecha" parece comportarse exactamente como eso, una doble flecha: convierte en 0 la combinación de valores 1-0 (como la flecha simple) pero también, como una especie de flecha inversa, convierte en 0 la combinación 0-1. Más fácil de recordar: arroja un resultado de 1 cuando sus argumentos valen lo mismo y 0 en otro caso[8].

Continuemos ahora con la tabla que teníamos a medias:

(	**p**	**∧**	**q**	)	**→**	**¬**	**p**
	1		1				1
	1		0				1
	0		1				0
	0		0				0

Ya sabemos cómo efectuar las operaciones, así que "capirucheemos" **p** y **q** (porque estas son sus dos argumentos), siguiendo paso a paso su tabla definitoria, o recordando simplemente que "capiruchear" es como multiplicar. Esto es lo que queda ahora (en gris las columnas empleadas, y en amarillo claro el operador que estamos calculando):

(	**p**	**∧**	**q**	)	**→**	**¬**	**p**
	1	**1**	1				1
	1	**0**	0				1
	0	**0**	1				0
	0	**0**	0				0

Recuerden que el resultado es el conjunto de valores NO del "capirucho", sino de la operación establecida dentro del paréntesis entre **p** y **q**.

Ahora, a "flechear". ¿Qué con qué? Pues su primer argumento con su segundo argumento. Atención aquí: el primer argumento de la flecha es todo el paréntesis. Sus valores correspondientes, que habrá que "flechear", NO son los que aparecen debajo de la **p**, ni debajo de la **q**: estos valores fueron "capirucheados" para obtener el resultado que habrá ahora que "flechear". Tomen nota de esto: cada columna de valores se emplea UNA y sólo una vez en un cálculo. Si usamos la columna de la **q** para "capiruchear" no la usaremos para "flechear" (pues la **q** es argumento del "capirucho", no de la "flecha")

Pero tenemos otro problema con el segundo argumento. Es **¬p**, y sólo tenemos los valores de **p**. Creo que toda la audiencia comprende que es necesario efectuar primero el cálculo monádico para, posteriormente, terminar la tabla:

---

[7] Este operador es un tanto caprichoso (de hecho da lugar a ciertas paradojas que no podemos comentar aquí. No, por favor, no insistan). Si tienen vds. una mente amante de las simetrías, pueden encontrar una operación matemática equivalente, pero ya más compleja: Eleven Q a P y obtendrán los mismos resultados (que yo sepa, nadie ha llamado a este operador "exponenciación inversa lógica", pero podría hacerse con igual derecho).

[8] Si tienen una mente obsesionada ya con las simetrías, lo siento, pero no conozco una operación aritmética simple que simule esta relación lógica (vaaaale, pueden vds multiplicar las dos exponenciaciones, directa e inversa – $Q^P \cdot P^Q$ –y obtendrán los mismos resultados, pero eso es ya más complicado que aprenderse la tablita). Por otra parte, este operador se llama también equivalencia o igualdad lógica, porque, como apreciarán más tarde (o ahora mismo, si piensan un poco) **P ↔ Q** es como P=Q. Pero vuelvan a las tablas.

(	**p**	**∧**	**q**	)	**→**	**¬**	**p**
	1	1	1			**0**	1
	1	0	0			**0**	1
	0	0	1			**1**	0
	0	0	0			**1**	0

Y ahora sí que sí (de nuevo, en gris las columnas empleadas, y en amarillo claro el operador que estamos calculando:

(	**p**	**∧**	**q**	)	**→**	**¬**	**p**
	1	1	1		**0**	0	1
	1	0	0		**1**	0	1
	0	0	1		**1**	1	0
	0	0	0		**1**	1	0

Repitamos el proceso con otra fbf, de forma más rápida: **[(p → q) ∧ ¬ q] → ¬ p**

1. Calculamos número de filas. Cuatro variables, pero sólo dos distintas; así que $2^2 = 4$

[(p → q) ∧ ¬ q] → ¬ p

2. Construimos el "esqueleto" de la tabla: colocamos los valores bajo las variables. Dos unos y dos ceros bajo las **p** y uno cero uno cero bajo las **q**

[	(	**p**	→	**q**	)	∧	¬	**q**	]	→	¬	**p**
		1		1				1				1
		1		0				0				1
		0		1				1				0
		0		0				0				0

3. Resolvemos el operador flecha, porque conocemos los valores de sus argumentos **p** y **q**: de acuerdo con su tabla definitoria, este operador sólo arroja un resultado de cero en el segundo caso

[	(	**p**	→	**q**	)	∧	¬	**q**	]	→	¬	**p**
		1	**1**	1				1				1
		1	**0**	0				0				1
		0	**1**	1				1				0
		0	**1**	0				0				0

4. Los otros dos operadores diádicos aún no pueden calcularse, puesto que todavía no tenemos los valores de todos sus argumentos. Fíjense que el segundo argumento de ∧ NO es **q** (no nos valdrían, por tanto, sus valores, que aparecen debajo), sino ¬**q** (cuyos valores, que aún no tenemos, serán los que aparezcan bajo el operador monádico, cuando lo resolvamos). Lo mismo pasa con la otra flecha.
Por tanto, resolvemos los operadores monádicos (tenemos los valores de sus argumentos **q** y **p**, respectivamente). De acuerdo con la tabla del operador monádico, se trata de invertir sus valores

[	(	**p**	→	**q**	)	∧	¬	**q**	]	→	¬	**p**
		1	1	1			**0**	1			**0**	1
		1	0	0			**1**	0			**0**	1
		0	1	1			**0**	1			**1**	0
		0	1	0			**1**	0			**1**	0

5. Calculamos ahora el "capirucho", porque tenemos los valores de sus dos argumentos (no podemos aún resolver el operador principal porque nos faltan los valores, precisamente, de su primer argumento, que es todo el corchete y aparecerá bajo su operador principal, el capirucho) El primero es **p → q** y el segundo es **¬q.** Los valores que deben "capiruchearse" son los que aparecen debajo de sus operadores principales (y únicos, en este caso). De acuerdo con la tabla definitoria de **∧**, sólo se obtiene un uno en el último caso

[	(	**p**	**→**	**q**	)	**∧**	**¬**	**q**	]	**→**	**¬**	**p**
		1	1	1		**0**	0	1			0	1
		1	0	0		**0**	1	0			0	1
		0	1	1		**0**	0	1			1	0
		0	1	0		**1**	1	0			1	0

6. Resolvemos finalmente el último operador, que es el operador principal y determinará el tipo de fbf que hemos tabulado. Observen que se trata de otra "flecha", pero ahora no se da el único caso en el que una "flecha" arroja un valor de 0. Se trata, por lo tanto, de una hermosa tautología[9].

[	(	**p**	**→**	**q**	)	**∧**	**¬**	**q**	]	**→**	**¬**	**p**
		1	1	1		0	0	1		**1**	0	1
		1	0	0		0	1	0		**1**	0	1
		0	1	1		0	0	1		**1**	1	0
		0	1	0		1	1	0		**1**	1	0

[9] Tan hermosa que hasta tiene nombre: *Modus tollendo tollens*. Todo esto tiene una explicación: se trata de… pero no, no se lo contaré todavía.