



Lógica y conectivas

Teoría 3.A.

La lógica

¿Qué es la lógica y por qué nos importa?

- La lógica es la ciencia que estudia los principios de los razonamientos y la inferencia, así como la validez de los mismos.
- Es una ciencia formal, es decir, estudia las formas de los razonamientos, y no los razonamientos en sí (de la misma forma que la matemática estudia la forma de la suma, independientemente si lo que sumamos son manzanas o naranjas).
- En las ciencias de la computación, la lógica es sumamente importante, ya que da sustento a la disciplina, estando presente en la lógica de los circuitos de las computadoras, el diseño de lenguajes de programación, la validación de programas, el enrutado de redes y por supuesto, la programación, entre muchos otros.
- Hay muchos tipos de lógica. Tantas que estudiar “lógica” completamente es una carrera en sí mismo.
- En la carrera de informática, los fundamentos de la lógica en profundidad se ven en la materia de “matemática para informática”.
- Nosotros vamos a ver solo algunos pocos detalles, que nos son relevantes para programar en esta materia.



Preparando una ensalada de frutas

- Deseamos preparar una ensalada de frutas, y todos sabemos que una buena ensalada de frutas lleva 3 ingredientes, bananas, manzanas y naranjas.
- Tenemos que comprar la fruta, ya que no tenemos nada en casa para prepararla, y vivimos en un barrio con una única verdulería, que cada dos por tres tiene faltantes.
- Queremos poder preguntarle al verdulero si nos es posible preparar una ensalada si compramos en su negocio, pero el verdulero no sabe absolutamente nada de ensaladas de frutas. Él solo sabe de frutas y verduras.
- Podemos preguntarle al verdulero por una fruta o verdura en particular, para saber si tiene o no esa fruta. Ej. ¿Hay naranjas?



Preparando una ensalada de frutas

- ¿Qué preguntas podemos hacer al verdulero?



Preparando una ensalada de frutas

¿Hay banana?	¿Hay manzana?	¿Hay naranja?	¿Se puede preparar la ensalada de frutas?
--------------	---------------	---------------	-------------------------------------------

Preparando una ensalada de frutas

¿Hay banana?	¿Hay manzana?	¿Hay naranja?	¿Se puede preparar la ensalada de frutas?
--------------	---------------	---------------	-------------------------------------------

Esta respuesta va a depender de las respuestas de las anteriores. Están relacionadas entre sí.



Preparando una ensalada de frutas

- ¿Qué preguntas podemos hacer al verdulero?
- ¿Qué respuesta esperamos en cada caso para saber si efectivamente podremos preparar nuestra ensalada?

Preparando una ensalada de frutas

¿Hay banana?	¿Hay manzana?	¿Hay naranja?	¿Se puede preparar la ensalada de frutas?
SI	SI	SI	SI
SI	SI	NO	NO
SI	NO	SI	NO
SI	NO	NO	NO
NO	SI	SI	NO
NO	SI	NO	NO
NO	NO	SI	NO
NO	NO	NO	NO

Preparando una ensalada de frutas

¿Hay banana?	¿Hay manzana?	¿Hay naranja?	¿Se puede preparar la ensalada de frutas?
SI	SI	SI	SI
SI	SI	NO	NO
SI	NO	SI	NO
SI			NO
NO			NO
NO			NO
NO	NO	SI	NO
NO	NO	NO	NO


Valuación

Cada fila de la tabla se conoce como valuación, y con respecto a las preguntas que podemos hacer al verdulero, cada una evalúa una situación con respuestas distintas.

Preparando una ensalada de frutas

¿Hay banana?	¿Hay manzana?	¿Hay naranja?	¿Se puede preparar la ensalada de frutas?
SI	SI	SI	SI
SI	SI	NO	NO
SI	NO	SI	NO
SI	NO	NO	NO
NO	SI	SI	NO
NO	SI	NO	NO
NO	NO	SI	NO
NO	NO	NO	NO

¡Se puede preparar la ensalada de frutas!
Solo en el caso de que haya de cada una de las frutas.





Proposiciones atómicas y compuestas

- Nuestras preguntas ¿Hay banana? ¿Hay manzana?, etc. son lo que en la lógica formal se conoce como proposiciones (Aunque pensadas en forma de pregunta, en los formalismos lógicos son oraciones informativas).
- Las respuestas Si/No son lo que llamamos un **valor de verdad**. En lógica formal hablamos de Verdadero/Falso.
- Hay **proposiciones** que son **atómicas**. No podemos descomponerla en partes más pequeñas, y su respuesta no depende de nadie más que de sí misma.
- Hay proposiciones que dependen de otras, y las llamamos **proposiciones compuestas**. Luego veremos cómo se definen estas últimas.



Universos posibles

- A veces la respuesta a una pregunta no es conocida a priori. Esto sucede en el caso del verdulero: no sabemos si va a tener o no de una determinada fruta. La respuesta a una pregunta solo se conoce al contrastarla con la realidad.
- Para saber en qué casos vamos a poder preparar la ensalada y en cuales no, debemos analizar todos los posibles universos. Es decir, todas las posibles combinaciones de respuestas que podría dar el verdulero.
- En lógica, tenemos que siempre analizar los diversos casos, incluso ante preguntas que parece no tener sentido analizar más de una respuesta, como “¿Los canguros viven en Australia?”. Hay que pensar siempre en “¿Qué pasaría sí...?”, y analizar qué sucedería por ej. en un universo donde los canguros hubieran surgido en otro lugar que no sea Australia.
- Lo que queremos analizar son las proposiciones atómicas. Al hacerlo, las proposiciones compuestas, que dependen de las anteriores, irán cambiando en cada caso.



Valuaciones

Una **valuación** es la **asignación** de una respuesta a cada una de las preguntas posibles (**proposición atómica**). Así, por ej. podemos tener la valuación:

- ¿Hay bananas? - Verdadero
- ¿Hay manzanas? - Falso
- ¿Hay naranjas? - Verdadero

O la valuación:

- ¿Hay bananas - Verdadero
- ¿Hay manzanas? - Verdadero
- ¿Hay naranjas ?- Verdadero

Cada valuación podríamos considerarla uno de esos posibles universos que nos interesa.



Tabla de verdad

- Una **tabla de verdad** es una herramienta para visualizar rápidamente todas las **valuaciones** posibles.
- Cada **columna contiene una proposición distinta** (en el ej. comenzamos con las atómicas por practicidad, pero el orden no es relevante).
- La cabecera contiene las proposiciones en sí, para identificar rápidamente la columna.
- Cada fila contiene una valuación posible. Es decir, una asignación de Verdadero/Falso a cada una de las proposiciones atómicas.
- Luego se completan los espacios de las proposiciones compuestas en base a la asignación de las atómicas en esa fila.



Equivalencia de proposiciones

Si tuviéramos que escribir la pregunta “¿Se puede preparar la ensalada de frutas?” de forma que sea el verdulero (que no sabe nada sobre ensaladas) pueda contestar, ¿Qué podemos hacer?

Debemos escribir la pregunta en términos de preguntas que él conoce.



Equivalencia de proposiciones

Si tuviéramos que escribir la pregunta “¿Se puede preparar la ensalada de frutas?” de forma que sea el verdulero (que no sabe nada sobre ensaladas) pueda contestar, ¿Qué podemos hacer?

Debemos escribir la pregunta en términos de preguntas que él conoce.

¿Hay bananas, manzanas y naranjas?

O si relajamos un poco la gramática del español:

¿hay bananas? y ¿hay manzanas? y ¿hay naranjas?

Conectivas

La “y” en una proposición es lo que se conoce como una **conectiva**.

Una conectiva une dos proposiciones, para formar una proposición compuesta.

¿hay bananas? y ¿hay manzanas?

Conectiva

Este elemento une a “hay bananas” con “hay manzanas”.

Conectivas

Pueden haber varias conectivas, en cuyo caso debe interpretarse de forma asociativa, de izquierda a derecha

¿hay bananas? y ¿hay manzanas? y ¿hay naranjas?

Este elemento une a
“¿hay bananas?” con
“¿hay manzanas?”.

Este elemento une a “¿hay bananas? y ¿hay
manzanas?” con “¿hay naranjas?”.



Conectivas

Podemos usar una notación más formal, y escribir paréntesis cuando hubiera ambigüedad, o cuando queramos que la asociación no sea de izquierda a derecha. Ej.

¿hay bananas? y (¿hay manzanas? y ¿hay naranjas?)



Conectivas

Cada proposición compuesta tiene un **valor de verdad** que está asociado a los valores de verdad de los elementos que la componen.

La proposición compuesta:

¿hay bananas? y ¿hay manzanas?

Depende de las proposiciones:

¿hay bananas?

¿hay manzanas?

El valor va a depender de la **semántica** de la conectiva.

Conjunción



Conjunción

La conectiva “y” es lo que se conoce como conjunción. En español la usamos todo el tiempo. El valor de verdad del elemento compuesto será verdadero, cuando el de cada una de las partes sea verdadero.

La proposición compuesta:

¿hay bananas? y ¿hay manzanas?

Es verdadera sólo cuando

¿hay bananas?

¿hay manzanas?

es verdadera



Notación formal

La semántica de la conectiva no tiene nada que ver con las proposiciones que une.

Ej.

¿hay manzanas? y ¿hay bananas?

¿está lloviendo? y ¿está ventoso?

Para que la proposición compuesta sea verdadera, ambas proposiciones unidas por el “y” deben ser verdaderas.

Por eso queremos analizar las conectivas a nivel **formal**. Es decir, nos va a importar la forma, el hecho de dos cosas unidas por un “y”, sin importar qué son esas cosas.

Así como en matemática se usa “a”, “b”, “c”, ..., etc. para nombrar números constantes en ecuaciones, en lógica proposicional se usan “p”, “q”, “r”, “s”, etc. (es convención empezar por la “p” de proposición, pero cualquier letra valdría)

Conjunción: Semántica

Podemos expresar la semántica de una conectiva con una tabla de verdad, que representa los valores posibles de las proposiciones que son unidas por la conectiva, y el valor del elemento total.

p	q	p “y” q
VERDADERO	VERDADERO	VERDADERO
VERDADERO	FALSO	FALSO
FALSO	VERDADERO	FALSO
FALSO	FALSO	FALSO

Conjunción: Semántica

Podemos expresar la semántica de una conectiva con una tabla de verdad, que representa los valores posibles de las proposiciones que son unidas por la conectiva, y el valor del elemento total.

p	q	p “y” q
VERDADERO	VERDADERO	VERDADERO
VERDADERO	FALSO	FALSO
FALSO	A partir de esta tabla queda claro que el único caso en el que “p y q” es verdadero, es cuando tanto “p” como “q” son verdaderos.	FALSO
FALSO		FALSO

Disyunción



Preparando una torta

- Queremos ahora preparar una torta, y toda buena torta lleva la ralladura de un cítrico, ya sea naranja o limón.
- ¿Qué pregunta podemos hacerle al verdulero para saber si podemos preparar nuestra torta? (Solo nos falta el cítrico, el resto de los elementos están)



Preparando una torta

¿hay limón? o ¿hay naranja?



Conectiva

El “o” es otra conectiva. Une a “¿hay limón?” con “¿hay naranja?”.



Disyunción

La conectiva “o” es lo que se conoce como disyunción. En español también la usamos todo el tiempo, pero hay casos particulares con los que hay que tener cuidado. El valor de verdad del elemento compuesto será verdadero, cuando alguno de los elementos que lo componen sea verdadero (o incluso cuando ambas lo sean)

La proposición compuesta:

¿hay limón? o ¿hay naranjas?

Es verdadera si **¿hay limón?** es verdadero

Es verdadera si **¿hay naranjas?** es verdadero

Es verdadera si tanto **¿hay limón?** como **¿hay naranjas?** son verdaderas

Disyunción: Semántica

La tabla de verdad de la disyunción.

p	q	p "o" q
VERDADERO	VERDADERO	VERDADERO
VERDADERO	FALSO	VERDADERO
FALSO	VERDADERO	VERDADERO
FALSO	FALSO	FALSO

Negación



Negación

La conectiva “no” es lo que se conoce como negación. Es más esquiva en el español, ya que no solemos hacer muchas preguntas que la involucren, pero sí afirmaciones. Esta conectiva no une dos proposiciones, sino que cambia el valor de verdad de una proposición.

La proposición compuesta:

no ¿hay limón?

Es verdadera si **¿hay limón?** es falso

Es falsa si **¿hay limón?** es verdadero

Negación: Semántica

La tabla de verdad de la negación.

p	“no” p
VERDADERO	FALSO
FALSO	VERDADERO



Símbolos formales

Sí bien podemos escribir “y” y “o” la realidad es que se vuelve difícil entender proposiciones compuestas con varias conectivas de esa forma.

Por ejemplo:

$p \text{ “y” } (q \text{ “o” } (\text{“no” } r))$

Necesitamos una mejor forma de escribirlo. Para eso, usamos símbolos que nos permiten hablar de las conectivas. Los símbolos que se asocian a “y”, “o” y “no” varían entre distintas disciplinas. Por ej. en matemática se usan \wedge , \vee y \neg . En electrónica suelen hablar de AND, OR y NOT, o de multiplicación, suma y complemento. Todas refieren a lo mismo.



Símbolos formales

Vamos a usar símbolos que se usan en matemática y lógica formal para las conectivas. Así “y” va a estar dado por “ \wedge ”, “o” se escribe con “ \vee ” y “no” se escribe como “ \neg ”.

Así:

$p \text{ “y” } (q \text{ “o” } (\text{“no” } r))$

Lo vamos a escribir:

$p \wedge (q \vee (\neg r))$

Precedencia

Las fórmulas con muchos paréntesis confunden, y son difíciles de entender. Sería mejor no tener que escribirlos. Para eso, la lógica proposicional utiliza una convención de orden de precedencia, es decir, un orden en el que, ante ausencia de paréntesis, puede usarse para entender qué debe interpretarse primero, y qué después.

El orden de precedencia es: \neg , (\wedge, \vee)

Es decir, primero se resuelven las negaciones, luego se resuelven las conjunciones, y las disyunciones, de izquierda a derecha, según los paréntesis. Es decir, si la negación no lleva paréntesis inmediatamente después, solo afecta a la expresión que le sigue.

La fórmula anterior se escribe

$$p \wedge (q \vee \neg r)$$



Momento de dudas o consultas

¿No terminaste de entender del todo
lógica y no estás cursando
“Matemática para informática”?

¡Entonces no dejes de leer el material
complementario!



¿Qué relación tiene esto con la programación?

Cuando hablamos de **proposición**, con un valor de verdad asociado, podemos pensar inmediatamente en los **sensores** que usamos en las alternativas condicionales.

Cuando hablamos de **universos posibles**, estamos hablando de los distintos **escenarios** iniciales o **estados** que pueden darse en una ubicación.

Un problema con conectivas

Supongamos que queremos preparar una ensalada de frutas, por lo que en la ubicación del escenario, deben haber manzanas y bananas (obviemos las naranjas de momento). Sí contamos con los sensores **¿hay manzanas?** y **¿hay bananas?**, y el comando **Preparar ensalada de frutas**, la única forma viable que teníamos hasta ahora de solucionar el problema es anidando alternativas.



Un problema con conectivas

Podríamos separar en subtarefas, claro, pero seguiríamos con un código complejo, en donde alto acoplamiento entre las partes, algo que no es bueno.



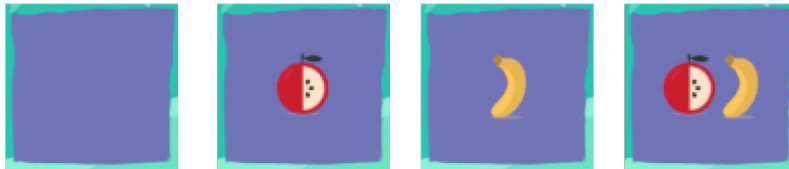
Un problema con conectivas

Con conectivas, podemos solucionar el problema sin anidar elementos. A esto se suma que podemos resolver problemas que, de otra forma, directamente no podríamos resolver.



Un problema con conectivas

¿Cuándo se prepara la ensalada?



p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

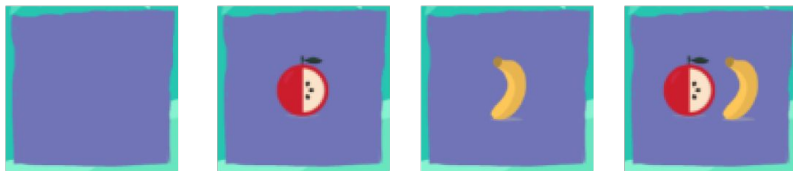
Un problema con conectivas

¿Cuándo se prepara la ensalada?



La tabla nos deja claro que solo se cumple una conjunción cuando ambas partes se cumplen.

p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

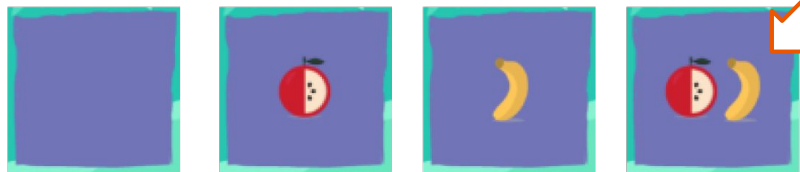


Un problema con conectivas

¿Cuándo se prepara la ensalada?



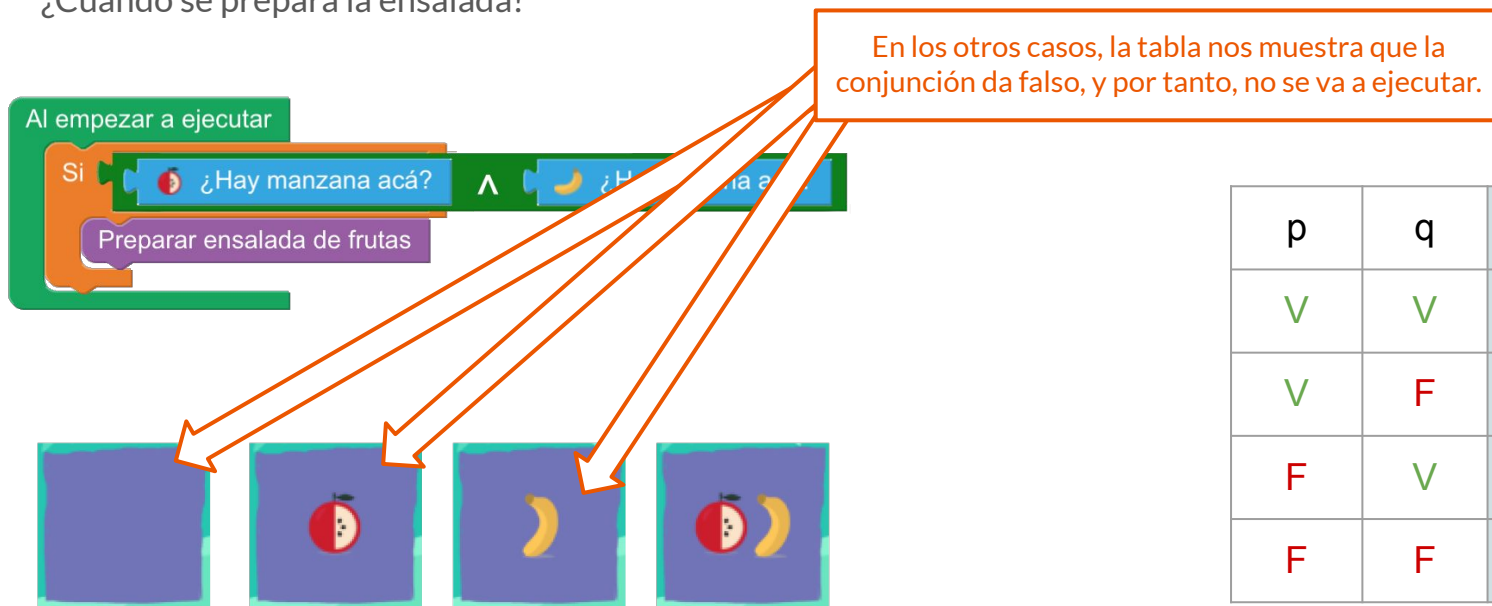
O sea, deben haber ambas frutas para que entre a ejecutar el código dentro de la alternativa.



p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

Un problema con conectivas

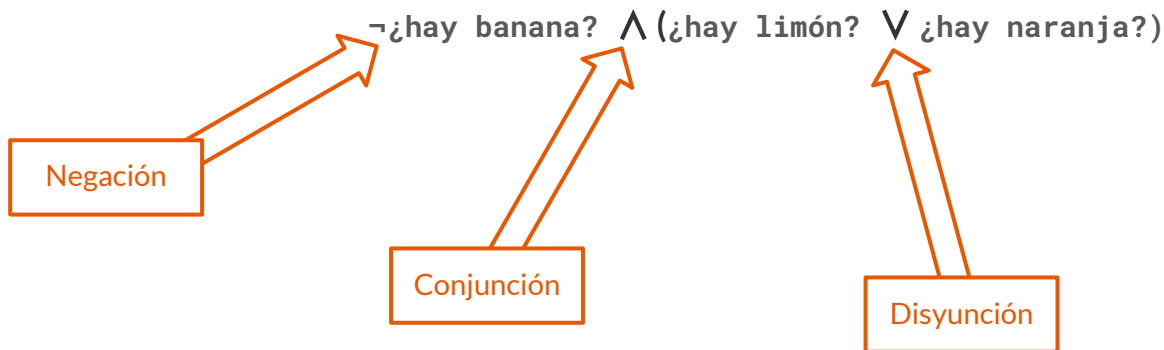
¿Cuándo se prepara la ensalada?



**PilasBloques no tiene
conectivas, pero podemos
usarlas en los ejercicios en
papel.**

Sintaxis para conectivas

Podemos escribir conectivas con la sintaxis matemática que mostramos, uniendo los distintos elementos.





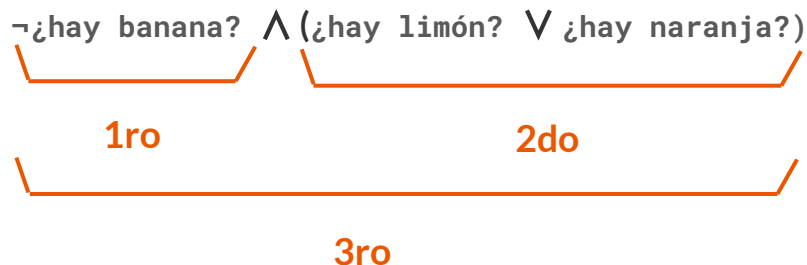
Orden de resolución

¿Cómo entender en qué situación entonces eso da verdadero y cual da falso? Debemos resolver paso a paso siguiendo los paréntesis de adentro a afuera y manejando la precedencia de la negación.

$$\neg \text{¿hay banana?} \wedge (\text{¿hay limón?} \vee \text{¿hay naranja?})$$

Orden de resolución

¿Cómo entender en qué situación entonces eso da verdadero y cual da falso? Debemos resolver paso a paso siguiendo los paréntesis de adentro a afuera y manejando la precedencia de la negación.





Momento de dudas o consultas

Hagamos un ejercicio concreto



La preparadora de ensaladas

Sally es una experta ensaladera de frutas. A Sally la contrataron de un restaurante para estar todo el día recorriendo la mesada, y preparando ensaladas de fruta, pero no siempre puede cumplir satisfactoriamente su trabajo. Como todos sabemos, una buena ensalada lleva 3 ingredientes, y Sally necesita contar con los tres en el bowl donde preparará la ensalada de frutas para poder hacerlo.

Sally será el personaje a manipular en este problema, y lo que se busca es un programa que haga que Sally prepare una ensalada de frutas en todo bowl en el que pueda hacerlo. El escenario inicial comienza con Sally en el extremo izquierdo, en la ubicación de un bowl, y hay otros 4 bowls hacia la derecha. En cada ubicación hay un bowl, y algunos ingredientes (manzanas, bananas y naranjas). Los ingredientes varían entre uno y otro bowl, habiendo a veces algunos, a veces otros, a veces ningunos y a veces todos, y esto es algo variable entre ejecuciones. El siguiente gráfico ilustra un posible escenario de inicio (pero no es el único posible, leer bien la descripción)

La preparadora de ensaladas





La preparadora de ensaladas

Las primitivas y sensores posibles para utilizar son los siguientes:

primitivas:

- **Mover a la derecha:** Mueve a Sally una ubicación a la derecha. Falla si no hay lugar a la derecha.
- **Preparar ensalada de frutas:** Prepara una ensalada de frutas en el bowl de la ubicación donde está Sally, consumiendo los ingredientes en el proceso. Falla si no están todos los ingredientes en la ubicación.

sensores:

- **¿Hay naranja?** Sensa para saber si hay una naranja en la ubicación actual.
- **¿Hay manzana?** Sensa para saber si hay una manzana en la ubicación actual.
- **¿Hay banana?** Sensa para saber si hay una naranja en la ubicación actual.



La preparadora de ensaladas

¡A resolver!

La preparadora de ensaladas

Primera aproximación. Resolvemos solo con lo que tenemos, pero anidamos alternativas. MAL.

Al empezar a ejecutar

```
| Repetir 4 veces
| | Preparar Ensalada Si Hay Ingredientes
| | Mover a la derecha
| Preparar Ensalada Si Hay Ingredientes
```

Definir Preparar Ensalada Si Hay Ingredientes

```
| Si ¿hay naranja? Entonces
| | Si ¿hay manzana? Entonces
| | | Sí ¿hay banana? Entonces
| | | Preparar ensalada de frutas
```



La preparadora de ensaladas

Podemos lograrlo mucho mejor si usamos conectivas lógicas.

Al empezar a ejecutar

```
| Repetir 4 veces
```

```
| | Preparar Ensalada Si Hay Ingredientes
```

```
| | Mover a la derecha
```

```
| Preparar Ensalada Si Hay Ingredientes
```

Definir Preparar Ensalada Si Hay Ingredientes

```
| Si ¿hay naranja? ^ ¿hay manzana? ^ ¿hay banana? Entonces
```

```
| | Preparar ensalada de frutas
```

Conectivas en nuestros condicionales

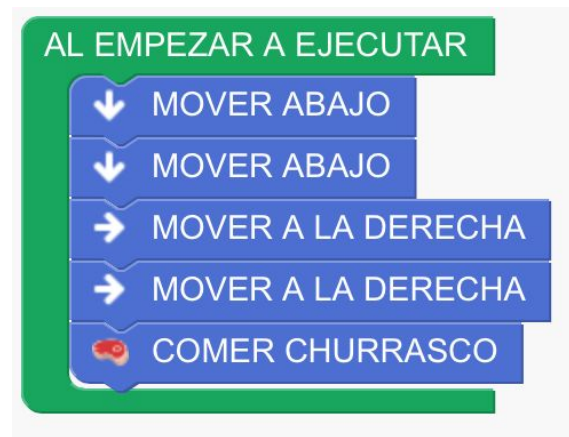
- Podemos conectar sensores con conectivas.
- Vamos a escribirlas con \wedge , \vee y \neg .
- La semántica está dada por la tabla de verdad de cada conectiva, y la precedencia implica que la negación se resuelve primero. Podemos usar paréntesis para cambiar la precedencia.
- Podemos tener ahora condiciones complejas, con varias partes.
- La condición de la alternativa es la condición completa. ¿Cuándo se cumple? Dependerá de las partes y las conectivas usadas.
- Es importante entender la lógica para saber qué conectiva usar y cuándo.



Momento de dudas o consultas

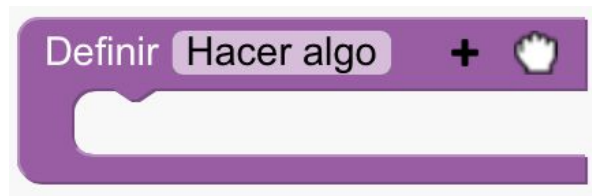
Repaso

- Un **programa** es una **descripción** de la **solución** a un **problema computacional**.
- Un **problema computacional** es aquel que puede expresarse como una **transformación de estado**.
- En PilasBloques lo expresamos mediante bloques que se encastran entre sí, para expresar un cambio de estado en el escenario.
- Todo programa tiene un **punto de entrada**.
- Los elementos fundamentales del programa son los **comandos** (descripciones de acciones).
- Los comandos se organizan en **secuencia**, y la solución se ejecuta según esa secuencia.
- Hay **infinitos** programas que solucionan un problema. Decimos que dos programas que solucionan el mismo problema son **equivalentes**.



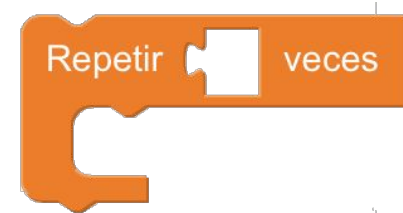
Repaso

- Los **procedimientos** son una herramienta del lenguaje que permite definir nuevos comandos.
- Tienen un **cuerpo** y un **nombre**.
- **El nombre debe ser claro y legible, comenzar con un verbo en infinitivo y estar relacionado a su propósito.**
- La **definición** va por un lado, y el **uso (invocación o llamada)** va en el cuerpo de algún otro bloque.
- Aportan claridad, legibilidad y modificabilidad al código.
- Pueden ser **reutilizados** muchas veces.
- Permiten transmitir claramente las ideas pensadas en nuestra **estrategia**.
- Permiten separar el problema (tarea) en partes más pequeñas para su más fácil resolución (**subtareas**).



Repaso

- La **repetición simple** es una herramienta del lenguaje que **permite cambiar el flujo** del programa (**estructura de control de flujo**).
- Permite estructurar el código de una forma distinta a la secuencia.
- Es un comando (y se puede usar junto con otros comandos en un cuerpo), pero tiene a su vez un cuerpo (es un **comando compuesto**)
- Espera una **expresión numérica** para indicar la cantidad de veces a repetir.
- **No hay que anidar repeticiones (ni ninguna otra estructura de control).**



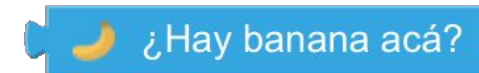
Repaso

- La **alternativa condicional** es una herramienta del lenguaje que **permite cambiar el flujo** del programa (**estructura de control de flujo**).
- Permite **elegir** entre posibles **ramas**, según una **condición** en el estado del programa.
- Es un comando (y se puede usar junto con otros comandos en un cuerpo), pero tiene a su vez un cuerpo (es un **comando compuesto**)
- Espera una **expresión de valor de verdad** para indicar cuándo se elige un camino y cuando otro..
- **No hay que anidar alternativas (ni ninguna otra estructura de control).**



Repaso

- Un **comando** es la descripción de una **acción**.
- Una **expresión** en la descripción de un **dato** (un **valor, información**)
- Puede ser **numéricas**, describen una cantidad, y las usamos en la repetición simple.
- Pueden ser de **valor de verdad**, se responden con verdadero o falso y las usamos en la alternativa.
- Los **sensores** son la herramienta mediante la cual obtenemos valores de verdad basados en el estado del programa (escenario)
- Podemos usar **conectivas** para combinar sensores que describan valores de verdad, para tener valores de verdad más complejos que dependen de esos sensores.



Repaso

- La **conjunción** une dos sensores, dando verdadero sólo cuando ambos evalúan a verdadero.
- La **disyunción** une dos sensores, dando verdadero sólo cuando alguno de los dos (o ámbos) son verdaderos.
- La **negación** aplica a un único sensor, y cambia el valor de verdad del mismo, transformando el verdadero en falso y viceversa.
- Se pueden usar expresiones con varias conectivas, y se resuelven por orden, primero las negaciones que aplican solo a un sensor, luego según indiquen los paréntesis, y si no hay más paréntesis, de izquierda a derecha.

p	$\neg p$
V	F
F	V

p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

p	q	$p \vee q$
V	V	V
V	F	V
F	V	V
F	F	F



Recordatorio: **iiiProgramar es comunicar!!!**

- Tus programas deberían quedar claros a partir de la lectura.
- Sí leo el punto de entrada, tiene que quedar más que explícita la estrategia elegida.
- Usamos procedimientos para la claridad, legibilidad y expresar la estrategia.
- Es importantísimo elegir nombres adecuados para los procedimientos que definimos.
- No anidar estructuras. Usamos procedimientos para dividir el problema en partes pequeñas y darles nombres adecuados.

Tarea

Realizamos la guía práctica de actividades en papel “P3A. Lógica y conectivas”.



Conectivas lógicas

Nos vemos la próxima