Ayudantía 4

Repaso Control 2

Por Bernardita Alliende y Rodrigo Figueroa

7 de abril 2025

Control 2 2025-1

Pregunta 1

¿Cuántos modelos tiene el siguiente programa?

```
1 {p_1; q_1; r_1 } 1.
1 {p_2; q_2; r_2 } 1.
...
1 {p_n; q_n; r_n } 1.
:- p_1.
:- q_1.
```

Pregunta 1

¿Cuántos modelos tiene el siguiente programa?

```
1 {p_1; q_1; r_1 } 1.
1 {p_2; q_2; r_2 } 1.
```

- 3 posibles opciones:
  - p_1
  - q_1
  - r_1
- 3 posibles opciones:
  - p_2
  - q_2
  - r_2

Total de combinaciones: $3x3 = 3^2$

Pregunta 1

¿Cuántos modelos tiene el siguiente programa?

```
1 {p_1; q_1; r_1 } 1.
1 {p_2; q_2; r_2 } 1.
...
1 {p_n; q_n; r_n } 1.
```

Total de combinaciones:

$3x3x3x...x3x3 = 3^n$

Pregunta 1

¿Cuántos modelos tiene el siguiente programa?

```
1 {p_1; q_1; r_1 } 1.
1 {p_2; q_2; r_2 } 1.
...
1 {p_n; q_n; r_n } 1.
:- p_1.
:- q_1.
```

- 1 posible opción

Total de combinaciones:

$1x3x3x...x3x3 = 3^{n-1}$

Pregunta 2

Considera que tienes el predicado `alumno(N, G, M)` con `N` su identificador, `G` un número entero que representa su generación y `M` el major al que pertenece, por ejemplo:

```
alumno(a, 2021, computacion).
alumno(b, 2022, software).
```

¿Cuál de las siguientes afirmaciones describe correctamente la siguiente regla?

```
5 {seleccionado(N) : alumno(N, G, computacion), G < 2023}.
```

Pregunta 2

```
5 {seleccionado(N) : alumno(N, G, computacion), G < 2023}.
```

Alumno de computación

Generación anterior a 2023

Al menos 5

Hasta n

Pregunta 2

```
5 {seleccionado(N) : alumno(N, G, computacion), G < 2023}.
```

Alumno de computación

Generación anterior a 2023

Al menos 5

Hasta n

Se deben seleccionar al menos 5 alumnos del major de computación conjunto de aquellos que pertenecen a una generación anterior a 2023.

Pregunta 3

Considera un programa en Clingo que modela distintos robots en una grilla. Estos se definen de la siguiente manera:

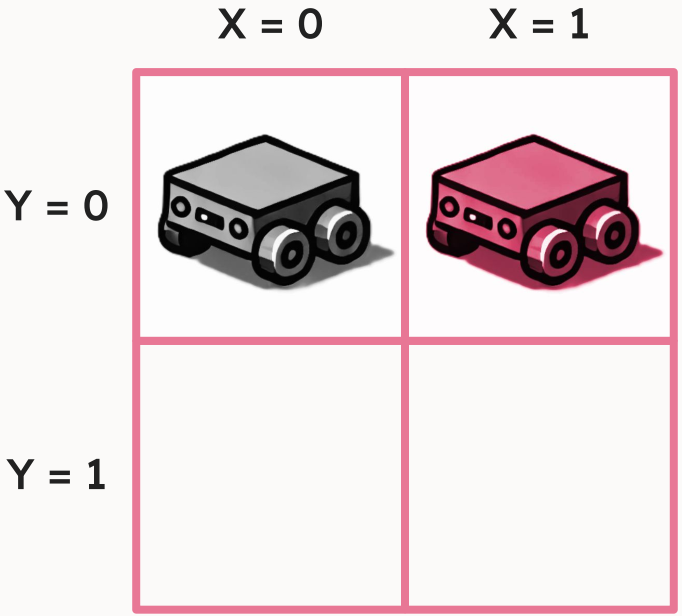
```
robot(id). % representa que id es un robot
robotOn(A, X, Y, T). % El robot de id A está en la posición (X, Y) en el tiempo T
```

Utilizando el predicado `robotOn`, queremos definir una regla que impida que dos robots intercambien sus posiciones entre un instante y el siguiente para evitar, como en nuestro ejemplo de clases, que los robots puedan "atravesarse" físicamente tanto horizontal como verticalmente.

¿Cuál de las siguientes alternativas representa la restricción propuesta?

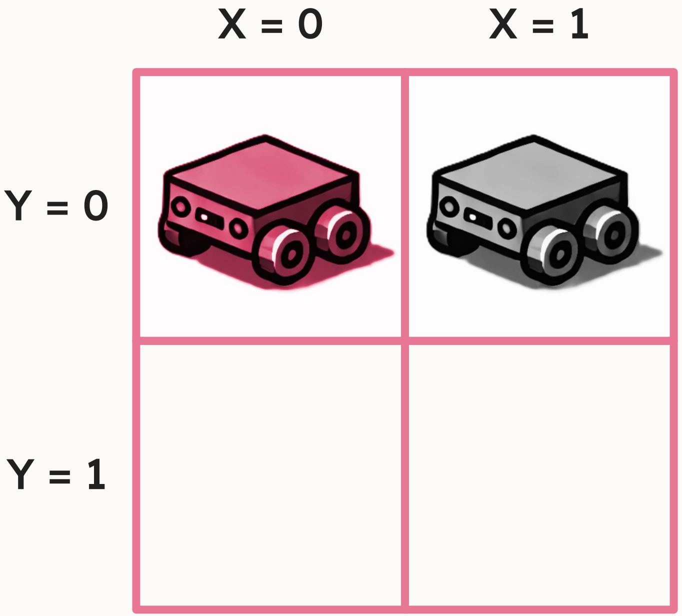
Pregunta 3

T = 0



Pregunta 3

T = 1



Pregunta 3

```
:- robotOn(A, X1, Y1, T), robotOn(B, X1+1, Y2, T), robotOn(A, X1+1, Y2, T),
robotOn(B, X1, Y2, T), A != B.
```

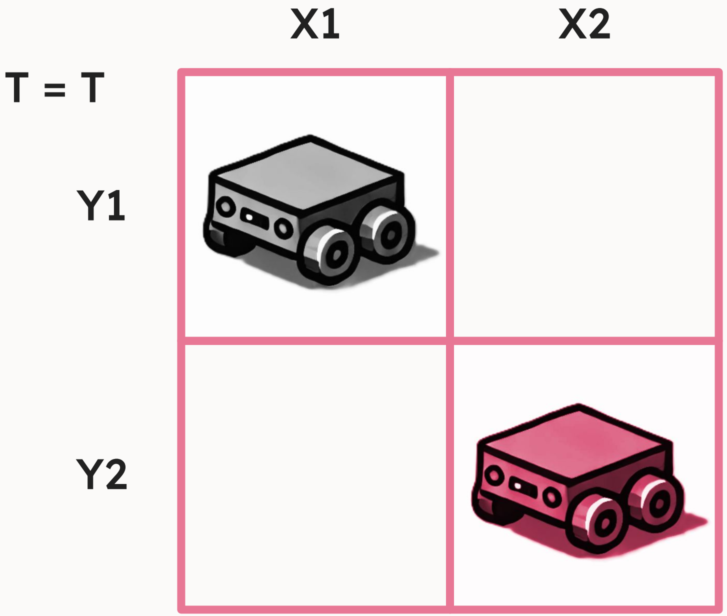
No son el mismo robot

A

≠

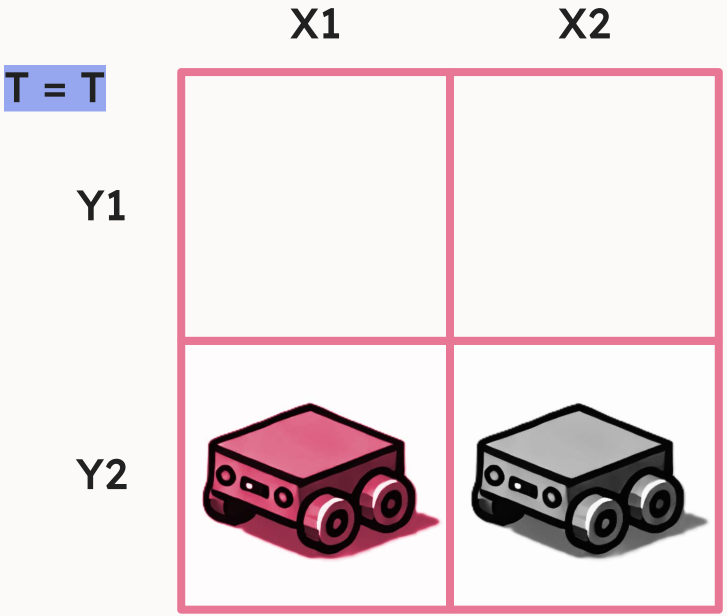
B

Pregunta 3



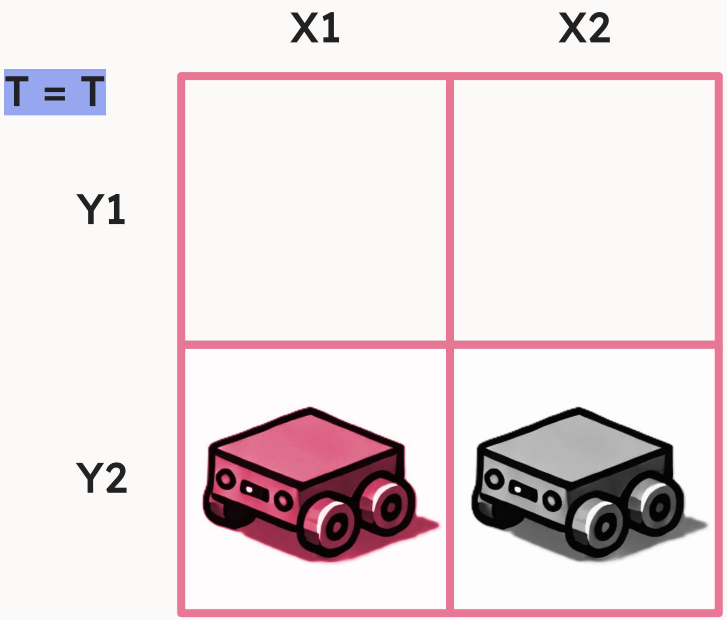

```
:- robotOn(A, X1, Y1, T), robotOn(B, X1+1, Y2,
T), robotOn(A, X1+1, Y2, T), robotOn(B, X1, Y2,
T), A != B.
```

Pregunta 3




```
:- robotOn(A, X1, Y1, T), robotOn(B, X1+1, Y2,
T), robotOn(A, X1+1, Y2, T), robotOn(B, X1, Y2,
T), A != B.
```

Pregunta 3




```
:- robotOn(A, X1, Y1, T), robotOn(B, X1+1, Y2,
T), robotOn(A, X1+1, Y2, T), robotOn(B, X1, Y2,
T), A != B.
```

- Mismo robot están en distintas partes en el mismo instante
- No se cumple con la condición pedida

Pregunta 3

```
:- robotOn(A, X1, Y1, T), robotOn(B, X1+1, Y2, T), robotOn(A, X1+1, Y2, T+1),
robotOn(B, X1, Y2, T+1), A != B.
```

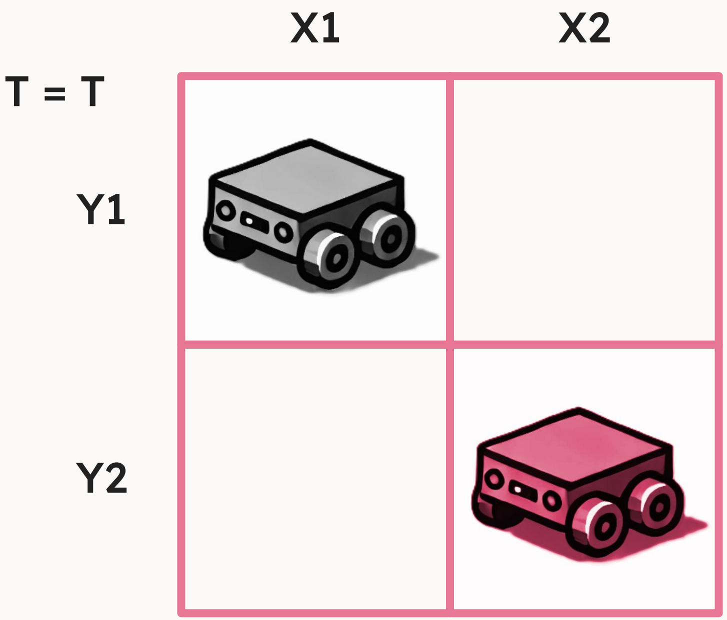
No son el mismo robot

A

≠

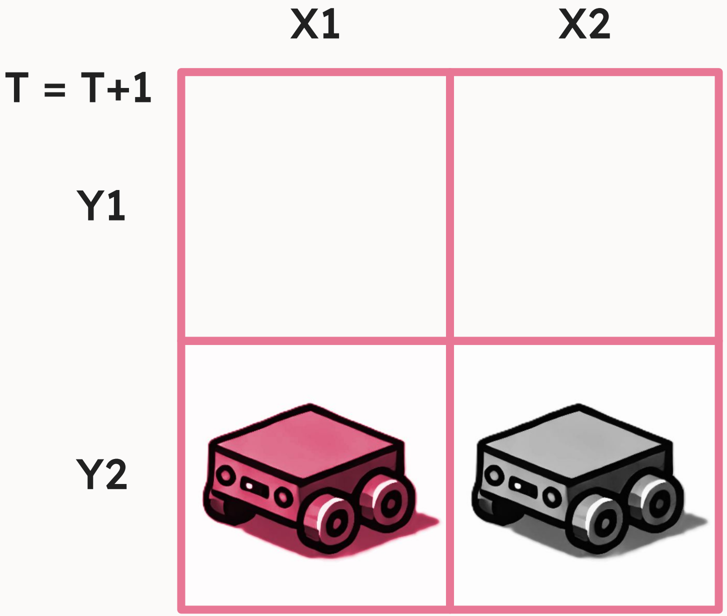
B

Pregunta 3




```
:- robotOn(A, X1, Y1, T), robotOn(B, X1+1, Y2,
T), robotOn(A, X1+1, Y2, T+1), robotOn(B, X1,
Y2, T+1), A != B.
```

Pregunta 3




```
:- robotOn(A, X1, Y1, T), robotOn(B, X1+1, Y2,
T), robotOn(A, X1+1, Y2, T+1), robotOn(B, X1,
Y2, T+1), A != B.
```

- No se cumple con la condición pedida

Pregunta 3

```
:- robotOn(A, X1, Y1, T), robotOn(B, X2, Y2, T), robotOn(A, X2, Y2, T+1),
robotOn(B, X1, Y1, T+1).
```

Podrían ser el mismo robot

A

≠

B

Pregunta 3

```
:- robotOn(A,X,Y,T), robotOn(B,X,Y), A != B.
```

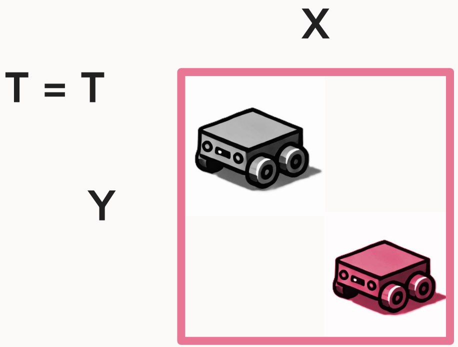
No son el mismo robot

A

≠

B

Pregunta 3




```
:- robotOn(A,X,Y,T), robotOn(B,X,Y), A != B.
```

- No se cumple con la condición pedida
- RobotOn de B no tiene el parámetro de tiempo

Pregunta 3

```
:- robotOn(A, X1, Y1, T), robotOn(B, X2, Y2, T), robotOn(A, X2, Y2, T+1),
robotOn(B, X1, Y1, T+1), A != B.
```

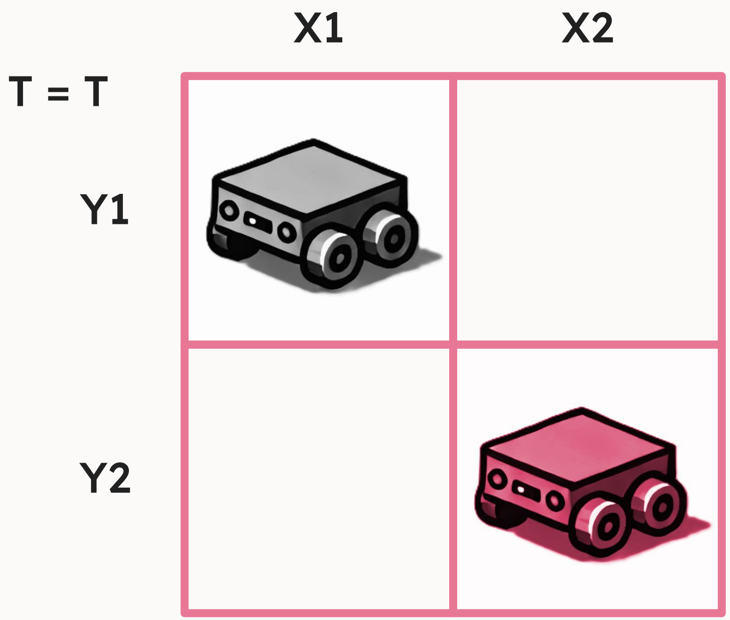
No son el mismo robot

A

$\neq$

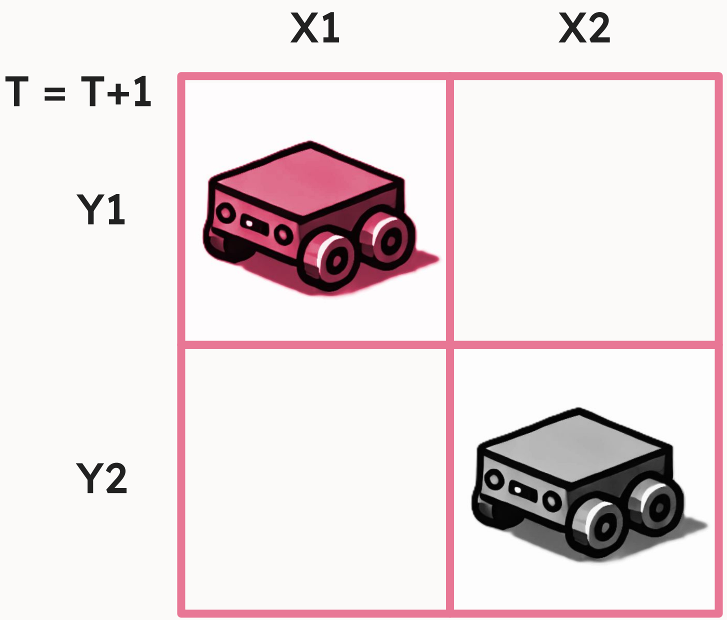
B

Pregunta 3



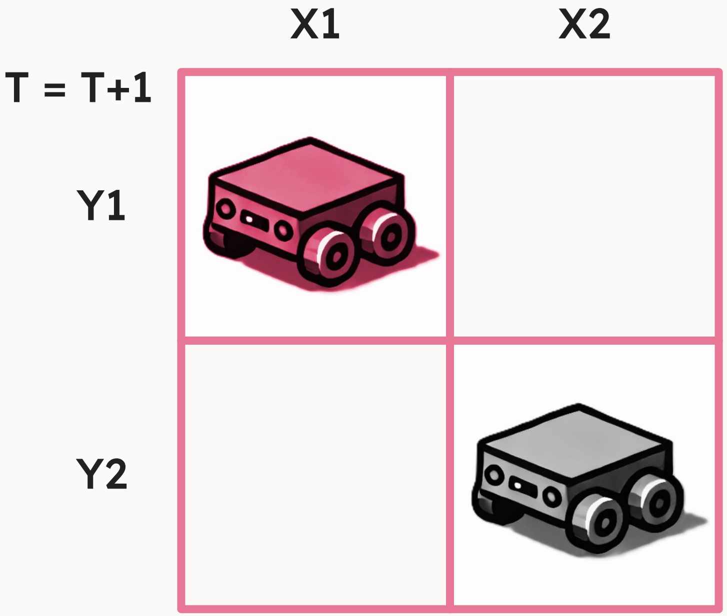

```
:- robotOn(A, X1, Y1, T), robotOn(B, X2, Y2, T),
robotOn(A, X2, Y2, T+1), robotOn(B, X1, Y1,
T+1), A != B.
```

Pregunta 3




```
:- robotOn(A, X1, Y1, T), robotOn(B, X2, Y2, T),
robotOn(A, X2, Y2, T+1), robotOn(B, X1, Y1,
T+1), A != B.
```

Pregunta 3




```
:- robotOn(A, X1, Y1, T), robotOn(B, X2, Y2, T),
robotOn(A, X2, Y2, T+1), robotOn(B, X1, Y1,
T+1), A != B.
```

Pregunta 4

Selecciona la alternativa que contiene un conjunto que no es modelo del siguiente programa:

```
1 { x; y; z } 2.
x :- y.
z :- not x.
```

```
a)  {x}
b)  {z}
c)  {x,z}
d)  {y}
```

Pregunta 4

```
1 { x; y; z } 2.
x :- y.
z :- not x.
```

```
a)  {x}
b)  {z}
c)  {x,z}
d)  {y}
```

x	z	z x
y	y x	z y

Pregunta 4

```
1 { x; y; z } 2.
x :- y.
z :- not x.
```

```
a)  {x}
b)  {z}
c)  {x,z}
d)  {y}
```

```
x          z          z x

y          y x        z y
```

Pregunta 4

```
1 { x; y; z } 2.
x :- y.
z :- not x.
```

```
a)  {x}
b)  {z}
c)  {x,z}
d)  {y}
```

x	z	z x
y	y x	z y

Pregunta 4

```
1 { x; y; z } 2.
x :- y.
z :- not x.
```

```
a)  {x}
b)  {z}
c)  {x,z}
d)  {y}
```

x	z	z x
y	y x	z y

No puede estar {y} solo, pues este siempre irá acompañado de x, dada la regla en rosado.

Pregunta 5

Considera el programa Π y el conjunto X

```
a :- not b.
b :- c, not d.
c :- not e.
d :- f, g.
```

```
X = {b, e}
```

Marca todas las reglas que pertenezccan a la reducción de Π respecto de X.

```
a)  d :- f,g.
b)  c :- not e.
c)  b :- c.
d)  a.
```

Pregunta 5

Considera el programa Π y el conjunto X

```
a :- not b.
b :- c, not d.
c :- not e.
d :- f, g.
```

```
X = {b, e}
```

Marca todas las reglas que pertenezccan a la reducción de Π respecto de X.

```
a)  d :- f,g.
b)  c :- not e.
c)  b :- c.
d)  a.
```

Recordatorio: Teoría de ASP

Definición

Un conjunto A que cumple una propiedad P es un conjunto **minimal** que cumple P ssi no existe un subconjunto propio de A que cumpla P.

Definición

M es un modelo de un programa Π si es un conjunto minimal que satisface que para cada regla $\mathit{Head} \leftarrow \mathit{Tail} \in \Pi$ tal que $\mathit{Tail} \subseteq M$, se cumple que $\mathit{Head} \cap M \neq \emptyset$.

Recordatorio: Teoría de ASP

Para construir un modelo M de un programa básico Π:

1. Agregamos todos los hechos del programa Π a M.
2. Si hay una regla $Head \leftarrow Tail$ en Π que cumple:
   (a) $Head$ no está contenido en M
   (b) Todos los elementos de $Tail$ están en M

   entonces agregamos $Head$ a M.
3. Si el paso 2 agrega algo a M volvemos al paso 2. En caso contrario, terminar.

Recordatorio: Teoría de ASP

Definición (Reducción)

La *reducción* un programa Π relativa a un conjunto X, denotada por Π^X es la que resulta de hacer:

1. $\Pi^X := \Pi$
2. **Borrar** toda regla $Head \leftarrow Pos \cup not(Neg)$ de Π^X cuando $Neg \cap X \neq \emptyset$.
3. **Reemplazar** cada regla $Head \leftarrow Pos \cup not(Neg)$ en Π^X por $Head \leftarrow Pos$ cuando $Neg \cap X = \emptyset$.

Definición (Modelo de un programa con negación)

X es un modelo de un programa con negación Π ssi X es un modelo para Π^X.

Pregunta 5

```
a :- not b.
```

Considera el programa Π y el conjunto X

```
a :- not b.
b :- c, not d.
c :- not e.
d :- f, g.
```

```
X = {b, e}
```

2. **Borrar** toda regla $Head \leftarrow Pos \cup not(Neg)$ de Π^X cuando $Neg \cap X \neq \emptyset$.
3. **Reemplazar** cada regla $Head \leftarrow Pos \cup not(Neg)$ en Π^X por $Head \leftarrow Pos$ cuando $Neg \cap X = \emptyset$.

- La negación not b, es sobre un átomo perteneciente al conjunto X?

Sí

- Entonces, borramos la regla.

Pregunta 5

Considera el programa Π y el conjunto X

```
a :- not b.
b :- c, not d.
c :- not e.
d :- f, g.
```

```
X = {b, e}
```

```
b :- c, not d.
```

2 **Borrar** toda regla $Head \leftarrow Pos \cup not(Neg)$ de Π^X cuando $Neg \cap X \neq \emptyset$.

3 **Reemplazar** cada regla $Head \leftarrow Pos \cup not(Neg)$ en Π^X por $Head \leftarrow Pos$ cuando $Neg \cap X = \emptyset$.

- La negación not d, es sobre un átomo perteneciente al conjunto X?

No

- Entonces, reemplazamos la regla por:

b :- c.

Pregunta 5

Considera el programa Π y el conjunto X

```
a :- not b.
b :- c, not d.
c :- not e.
d :- f, g.
```

```
X = {b, e}
```

```
a :- not b.
```

2. **Borrar** toda regla $Head \leftarrow Pos \cup not(Neg)$ de Π^X cuando $Neg \cap X \neq \emptyset$.
3. **Reemplazar** cada regla $Head \leftarrow Pos \cup not(Neg)$ en Π^X por $Head \leftarrow Pos$ cuando $Neg \cap X = \emptyset$.

- La negación not e, es sobre un átomo perteneciente al conjunto X?

Sí

- Entonces, borramos la regla.

Pregunta 5

Considera el programa Π y el conjunto X

```
a :- not b.
b :- c, not d.
c :- not e.
d :- f, g.
```

```
X = {b, e}
```

```
a :- not b.
```

2 **Borrar** toda regla $Head \leftarrow Pos \cup not(Neg)$ de Π^X cuando $Neg \cap X \neq \emptyset$.

3 **Reemplazar** cada regla $Head \leftarrow Pos \cup not(Neg)$ en Π^X por $Head \leftarrow Pos$ cuando $Neg \cap X = \emptyset$.

- La regla d :- f, g. , tiene alguna negación?

No

- Entonces, se mantiene como está

Pregunta 5

```
a :- not b.
```

Considera el programa Π y el conjunto X

```
a :- not b.
b :- c, not d.
c :- not e.
d :- f, g.
```

```
X = {b, e}
```

2. **Borrar** toda regla $Head \leftarrow Pos \cup not(Neg)$ de Π^X cuando $Neg \cap X \neq \emptyset$.
3. **Reemplazar** cada regla $Head \leftarrow Pos \cup not(Neg)$ en Π^X por $Head \leftarrow Pos$ cuando $Neg \cap X = \emptyset$.

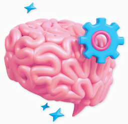
- **Finalmente, la reducción del modelo será:**

b :- c.
d :- f, g.

Pregunta 6

Selecciona todas las afirmaciones correctas acerca del requisito de minimalidad de un modelo M para un cierto programa Π

- La minimalidad de M establece que en M deben estar al menos todos los átomos que se deducen de Π.
- La minimalidad de M establece que no hay un modelo M' de Π cuya cardinalidad sea menor que la de M.
- La minimalidad de M establece que en M no pueden haber átomos injustificados (no deducidos desde el programa).
- La minimalidad de M establece que no hay un modelo M' de Π que sea subconjunto propio de M.



Pregunta 6

Selecciona todas las afirmaciones correctas acerca del requisito de minimalidad de un modelo M para un cierto programa Π

- La minimalidad de M establece que en M deben estar al menos todos los átomos que se deducen de Π. ❌
- La minimalidad de M establece que no hay un modelo M' de Π cuya cardinalidad sea menor que la de M.
- La minimalidad de M establece que en M no pueden haber átomos injustificados (no deducidos desde el programa).
- La minimalidad de M establece que no hay un modelo M' de Π que sea subconjunto propio de M.

Pregunta 6

Selecciona todas las afirmaciones correctas acerca del requisito de minimalidad de un modelo M para un cierto programa $\mathbf{\Pi}$

- La minimalidad de M establece que en M deben estar al menos todos los átomos que se deducen de $\mathbf{\Pi}$. ❌
- La minimalidad de M establece que no hay un modelo M' de $\mathbf{\Pi}$ cuya cardinalidad sea menor que la de M. ❌
- La minimalidad de M establece que en M no pueden haber átomos injustificados (no deducidos desde el programa).
- La minimalidad de M establece que no hay un modelo M' de $\mathbf{\Pi}$ que sea subconjunto propio de M.

Pregunta 6

Selecciona todas las afirmaciones correctas acerca del requisito de minimalidad de un modelo M para un cierto programa Π

- La minimalidad de M establece que en M deben estar al menos todos los átomos que se deducen de Π. ❌
- La minimalidad de M establece que no hay un modelo M' de Π cuya cardinalidad sea menor que la de M. ❌

```
a :- not b.
b :- not a.
c :- a.
```

```
M_1 = {a, c}
M_2 = {b}
```

Pregunta 6

Selecciona todas las afirmaciones correctas acerca del requisito de minimalidad de un modelo M para un cierto programa Π

- La minimalidad de M establece que en M deben estar al menos todos los átomos que se deducen de Π. ❌
- La minimalidad de M establece que no hay un modelo M' de Π cuya cardinalidad sea menor que la de M.❌
- La minimalidad de M establece que en M no pueden haber átomos injustificados (no deducidos desde el programa).✅
- La minimalidad de M establece que no hay un modelo M' de Π que sea subconjunto propio de M.

Pregunta 6

Selecciona todas las afirmaciones correctas acerca del requisito de minimalidad de un modelo M para un cierto programa $\mathbf{\Pi}$

- La minimalidad de M establece que en M deben estar al menos todos los átomos que se deducen de $\mathbf{\Pi}$. ❌
- La minimalidad de M establece que no hay un modelo M' de $\mathbf{\Pi}$ cuya cardinalidad sea menor que la de M. ❌
- La minimalidad de M establece que en M no pueden haber átomos injustificados (no deducidos desde el programa).✅
- La minimalidad de M establece que no hay un modelo M' de $\mathbf{\Pi}$ que sea subconjunto propio de M. ✅

Interrogación 1 2024-2

Pregunta 1 a

a) Supón que Π es un programa en lógica que tiene **n** modelos. ¿Es correcto que al agregar la siguiente línea a Π:

```
1 {p ; q} 1.
```

a) El programa tendrá 2^n modelos?

Pregunta 1 a

a) Supón que Π es un programa en lógica que tiene **n** modelos. ¿Es correcto que al agregar la siguiente línea a Π:

```
1 {p ; q} 1.
```

a) El programa tendrá 2^n modelos?
   - Falso: Hay que considerar que Π puede contener:

p.

q.

Pregunta 1 b.

Supón que Π es un programa básico (sin negación ni restricciones de cardinalidad, pero posiblemente con reglas con disyunciones en las cabezas) con m reglas. Sea M un modelo de Π. ¿Cuántos modelos de Π podrían existir que son subconjunto propio de M?

Pregunta 1 b.

Supón que Π es un programa básico (sin negación ni restricciones de cardinalidad, pero posiblemente con reglas con disyunciones en las cabezas) con m reglas. Sea M un modelo de Π. ¿Cuántos modelos de Π podrían existir que son subconjunto propio de M?

=> La definición de modelo expresa que todo modelo es minimal. Esto implica que no puede haber ningún modelo que es subconjunto propio de *M*. La respuesta, entonces, es 0.

Pregunta 2 a.

Usando la definición de modelo, demuestra que el siguiente programa no tiene modelos.

```
p.
:- p.
```

Recordatorio: Teoría de ASP

Definición

M es un modelo de un programa Π si es un conjunto minimal que satisface que para cada regla $Head \leftarrow Tail \in \Pi$ tal que $Tail \subseteq M$, se cumple que $Head \cap M \neq \emptyset$.

Pregunta 2 a.

Usando la definición de modelo, demuestra que el siguiente programa no tiene modelos.

```
p.
:- p.
```

- Por la primera regla, se debe cumplir que $p \in M$.
- Por la segunda regla, dado que $\{p\} \subseteq M$, se debe cumplir que $M \cap \varnothing \neq \varnothing$.
- Esto contradice el hecho que $p \in M$, ya que $M \cap \varnothing$ siempre es igual a $\varnothing$.
- Concluimos que no existe un modelo para el programa Π

Pregunta 2 b.

¿Cuantos modelos tiene este programa?

p_1 :- not p_2.

p_2 :- not p_1.

p_2 :- not p_3.

p_3 :- not p_2.

p_{n-1} :- not p_n.

p_n :- not p_{n-1}

Justifica tu respuesta utilizando la definición de modelo.

Pregunta 2 b.

¿Cuantos modelos tiene este programa?

p_1 :- not p_2.

p_2 :- not p_1.

p_2 :- not p_3.

p_3 :- not p_2.

p_{n-1} :- not p_n.

p_n :- not p_{n-1}

- Por la estructura del programa, p_i y p_i+1 no pueden pertenecer simultáneamente a un modelo.

- Tampoco es posible que ambos (p_i y p_i+1) estén ausentes simultáneamente del modelo.

- Estas restricciones solo permiten dos posibles modelos:

    - $M_1 = \{p_i \mid i$ es par$\}$ (conjunto de todos los p con índice par)
    - $M_2 = \{p_i \mid i$ es impar$\}$ (conjunto de todos los p con índice impar)
- La reducción Π^M_1 tiene como modelo estable a M_1
- La reducción Π^M_2 tiene como modelo estable a M_2

Pregunta 2 c.

Sea II el siguiente programa:

```
m :- not n.
p :- not q, not r, s.
t :- not u.
v :- x, not y.
```

Dado X = {n, y}, describe el contenido de Π^X, o, dicho de otra forma, escribe las reglas que tiene la reducción (o simplificación) de Π con respecto a X.

Pregunta 2 c.

Sea II el siguiente programa:

```
m :- not n.
p :- not q, not r, s.
t :- not u.
v :- x, not y.
```

```
p :- s.
t.
```

Dado X = {n, y}, describe el contenido de Π^X, o, dicho de otra forma, escribe las reglas que tiene la reducción (o simplificación) de Π con respecto a X.

Pregunta 3

El objetivo de esta pregunta es modelar el razonamiento de un agente que se puede mover entre ubicaciones, y recoger y dejar dos objetos: una taza o una pelota.

Las ubicaciones se representan con un predicado **`ubicación`**. Por ejemplo:

```
ubicacion(oficina).
ubicacion(cocina).
ubicacion(pieza).
```

Para establecer dónde está cada objeto en el tiempo T, se usa el predicado en, de la siguiente manera.

```
en(pelota, oficina, 0). % la pelota está en la oficina en el tiempo 0
en(taza, cocina, 0). % la taza est´ a en la cocina en el tiempo 0
```

Pregunta 3

Para expresar que el agente tiene la pelota o la taza se utiliza el predicado **`tiene`.**
De esta forma, cuando el agente tiene a la pelota en tiempo 4, esperamos que el modelo del programa contenga:

```
tiene(pelota,4).
```

Pregunta 3

Las acciones disponibles se describen a través del predicado **acción**, por ejemplo,

```
accion(mover_oficina_cocina). % describe la acción de moverse desde la oficina a la
cocina
accion(mover_cocina_oficina). % describe la acción de moverse desde la cocina a la
oficina
accion(recoger_pelota). % describe la acción de recoger la pelota
accion(dejar_pelota). % describe la acción de dejar la pelota
```

Suponiendo que se ocupa un predicado **exec**, tal que **exec(a,t)** representa que a se ejecuta en el tiempo **t**

```
exec(a,t)
```

Pregunta 3.a

I. Una regla que exprese que cuando el **agente** se mueve **desde la oficina a la cocina** en tiempo **T,** entonces en tiempo **T+1** el agentes se encuentra en la **cocina**

Pregunta 3.a

I. Una regla que exprese que cuando el **agente** se mueve **desde la oficina a la cocina** en tiempo **T,** entonces en tiempo **T+1** el agente se encuentra en la **cocina**

Entonces en tiempo T+1 el agentes se encuentra en la cocina **:-** Si el agente se mueve desde la oficina a la cocina en tiempo T

Pregunta 3.a

I. Una regla que exprese que cuando el **agente** se mueve **desde la oficina a la cocina** en tiempo **T,** entonces en tiempo **T+1** el agentes se encuentra en la **cocina**

Entonces en tiempo T+1 el agentes se encuentra en la cocina **:-** Si el agente se mueve desde la oficina a la cocina en tiempo T

```
en(agente, cocina, T+1) :- exec(mover_oficina_cocina, T), en(agente, oficina, T).
```

Pregunta 3.a

II. Una regla que exprese que cuando el **agente** se mueve **desde la oficina a la cocina**, la **pelota sigue estando donde sea que estaba.**

Pregunta 3.a

II. Una regla que exprese que cuando el **agente** se mueve **desde la oficina a la cocina**, la **pelota sigue estando donde sea que estaba.**

Entonces la pelota sigue estando donde sea que estaba.	**:-**	Si el agente se mueve desde la oficina a la cocina

Pregunta 3.a

II. Una regla que exprese que cuando el **agente** se mueve **desde la oficina a la cocina**, la **pelota sigue estando donde sea que estaba.**

Entonces la pelota sigue estando donde sea que estaba. **:-** Si el agente se mueve desde la oficina a la cocina

```
en(pelota, X, T+1) :- exec(mover_oficina_cocina, T), en(pelota, X, T).
```

Pregunta 3.b

I. Una regla que **prohiba** que el **agente** se pueda mover de la **oficina a la cocina** en tiempo **T a menos que esté en la cocina en tiempo T**.

Pregunta 3.b

I. Una regla que **prohiba** que el **agente** se pueda mover de la **oficina a la cocina** en tiempo **T a menos que esté en la cocina en tiempo T**.

```
:- exec(mover_oficina_cocina, T), not en(agente, oficina, T).
```

Pregunta 3.b

II. Una regla que prohibe al agente acarrear más de un objeto a la vez

Pregunta 3.b

II. Una regla que prohibe al agente acarrear más de un objeto a la vez

```
:- tiene(O1, T), tiene(O2,T), O1 != O2.
```

Pregunta 3.c

I. Una regla que exprese que cuando el **agente recoge la pelota**, ahora el **agente tiene la pelota**.

Pregunta 3.c

I. Una regla que exprese que cuando el **agente recoge la pelota**, ahora el **agente tiene la pelota**.

Entonces ahora el agente tiene la pelota. **:-** Si agente recoge la pelota

Pregunta 3.c

I. Una regla que exprese que cuando el **agente recoge la pelota**, ahora el **agente tiene la pelota**.

Entonces ahora el agente tiene la pelota. **:-** Si agente recoge la pelota

```
tiene(pelota, T+1) :- exec(recoger_pelota, T).
```

Pregunta 3.c

II. Una regla que **prohiba recoger la pelota** a menos que el **agente** se encuentre en el **lugar donde está la pelota.**

Pregunta 3.c

II. Una regla que **prohiba recoger la pelota** a menos que el **agente** se encuentre en el **lugar donde está la pelota.**

```
:- exec(recoger_pelota, T), en(agente, L, T), not en(pelota, L, T).
```

Pregunta 3.d

Escribe las reglas que permitan expresar el **objetivo** de que la **pelota** esté en la **cocina** en el **tiempo 5**

Pregunta 3.d

Escribe las reglas que permitan expresar el **objetivo** de que la **pelota** esté en la **cocina** en el **tiempo 5**

```
objetivo :- en(pelota, cocina, 5).
 :- not objetivo.
```

Dudas