~~Yingo~~

Clingo

Ayudantía 3 por Daniel Florea y Bruno Farfán

~~Yingo~~

Clingo reloaded

Ayudantía 3 por Daniel Florea y Bruno Farfán

¿Qué haremos hoy?

Negación

Modelación

not

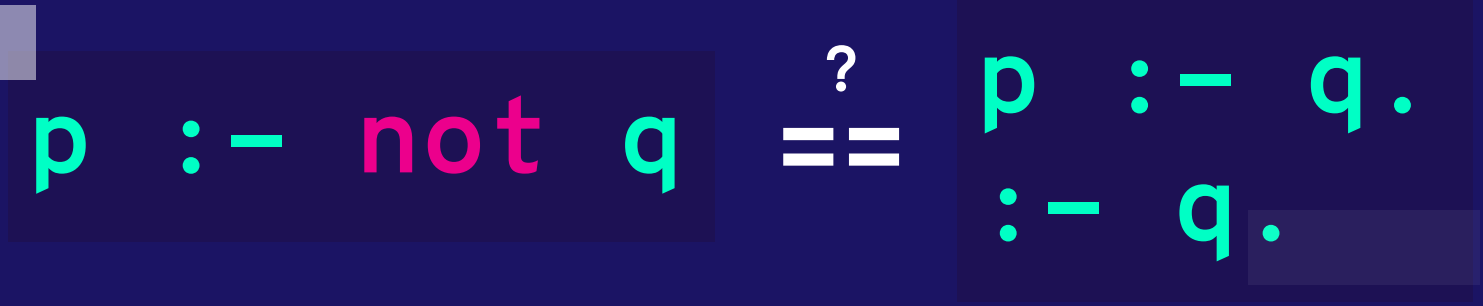
Negación

Esto es una regla.

```
p :- q.
```

Esto también es una regla.

```
p :- not q.
```


p :- not q
?
==
p :- q.
:- q.


Mentimeter

```
p :- q.     # p está en el
            # modelo si q lo
            # está

:- q.       # q no está en el
            # modelo
```

```
p :- not q. # p está en el
            # modelo si este
            # no contiene a q
```

Veamos los output…

```
p :- q.     # p está en el
            # modelo si q lo
            # está

:- q.       # q no está en el
            # modelo
```

```
p :- not q.     # p está en el
                # modelo si este
                # no contiene a q
```

Veamos los output…

```
p :- q.     # p está en el
            # modelo si q lo
            # está

:- q.       # q no está en el
            # modelo
```

```
p :- not q.     # p está en el
                # modelo si este
                # no contiene a q
```

Answer:
{}

Veamos los output…

```
p :- q.     # p está en el
            # modelo si q lo
            # está

:- q.       # q no está en el
            # modelo
```

Answer:
{}

```
p :- not q.     # p está en el
                # modelo si este
                # no contiene a q
```

Answer:
{p}

Veamos los output…

```
p :- q.     # p está en el
            # modelo si q lo
            # está

:- q.       # q no está en el
            # modelo
```

Answer:
{}

```
p :- not q.     # p está en el
                # modelo si este
                # no contiene a q
```

Answer:
{p}

```
p :- not q

q :- not p
```

```
p :- not q

q :- not p
```

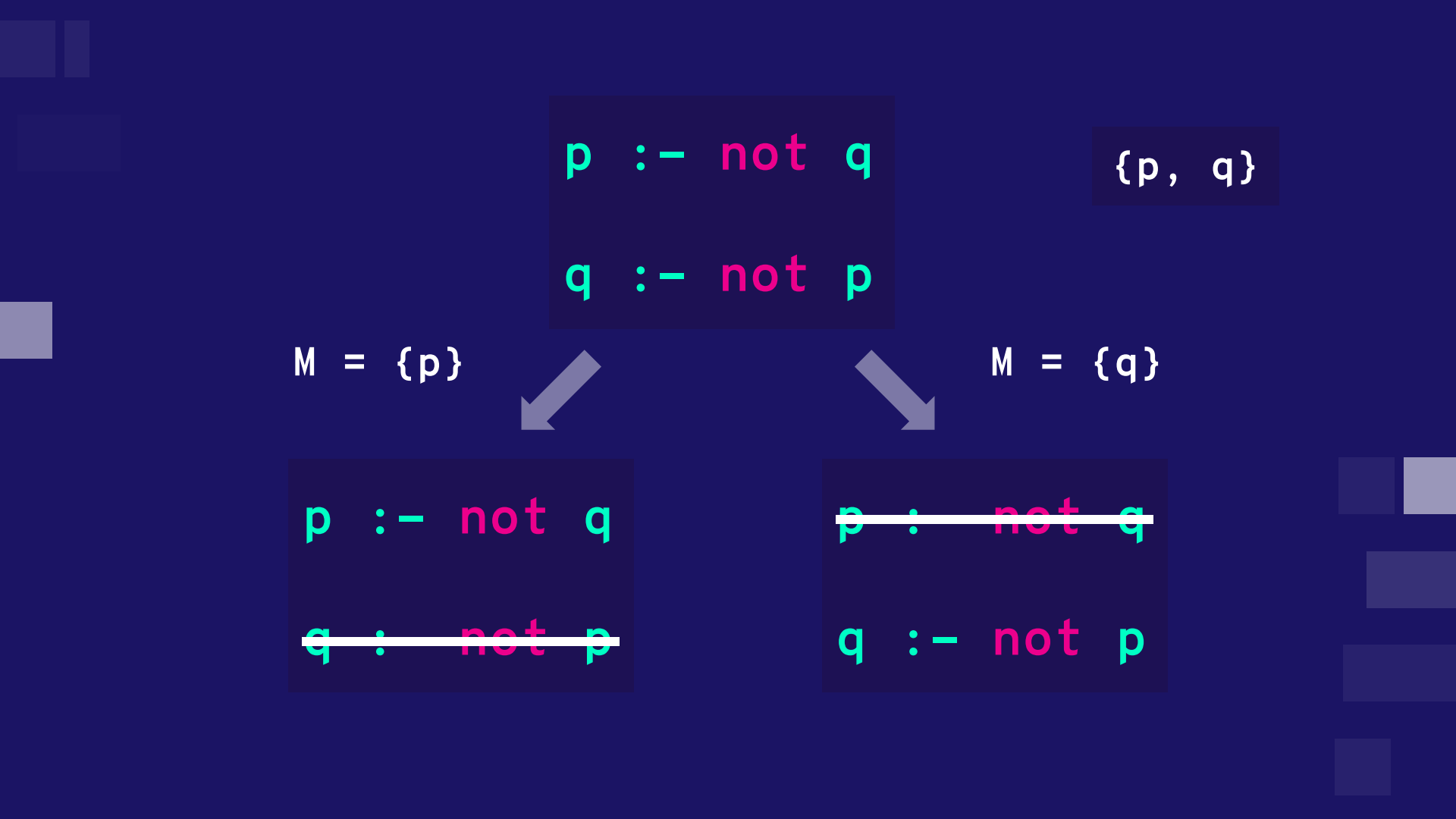
{p, q}

```
p :- not q

q :- not p
```

{p, q}

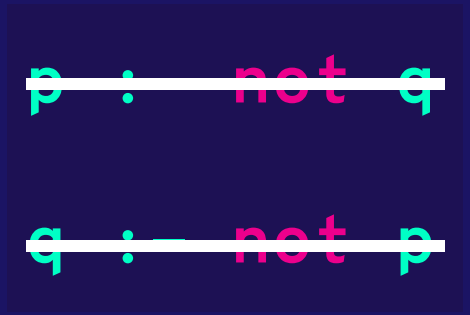
M = {p}

```
p :- not q

~~q :- not p~~
```


p :- not q
q :- not p
{p, q}
M = {p}
M = {q}
p :- not q
q :- not p
p :- not q
q :- not p

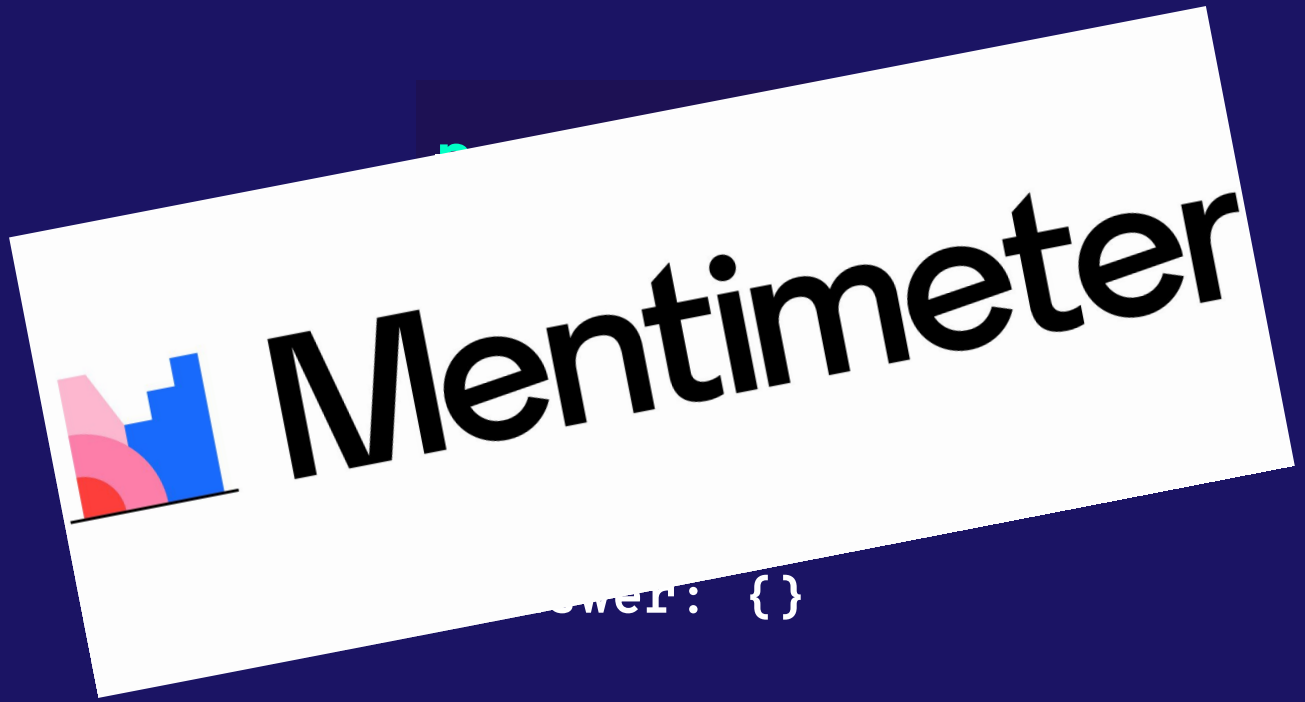
```
p :- not q

q :- not p
```

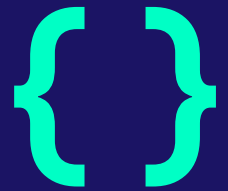
¿M = {p, q}?



¿M = {p, q}?

NO PERTENECE A
LA SOLUCIÓN


Mentimeter



Restricciones de Cardinalidad

(usando negación)

¿Qué nos entrega este programa?

```
dueño.
{gato; perro; loro} :- dueño.
```

¿Qué nos entrega este programa?

```
dueño.
{gato; perro; loro} :- dueño.
```

- Nos entrega un modelo por combinación posible.

¿Qué nos entrega este programa?

```
dueño.
{gato; not perro} :- dueño.

perro :- gato.
```

¿Qué nos entrega este programa?

```
dueño.
{gato; not perro; loro} :- dueño.
```

- Nos entrega un modelo por combinación posible.
- Cuando considera a *not perro*, lo excluye del modelo.

Veamos el desarrollo…

```
dueño.
1 {gato; not perro; loro} 2 :- dueño.
```

	gato	not perro	loro
gato	M = {dueño, gato}	M = {dueño, gato}	M = {dueño, gato, loro}
not perro	M = {dueño, gato}	M = {dueño}	M = {dueño, loro}
loro	M = {dueño, gato, loro}	M = {dueño, loro}	M = {dueño, loro}

Veamos el desarrollo…

```
dueño.
1 {gato; not perro; loro} 2 :- dueño.
```

4 Soluciones

	gato	not perro	loro
gato	M = {dueño, gato}		M = {dueño, gato, loro}
not perro		M = {dueño}	M = {dueño, loro}
loro			

Veamos el desarrollo…

```
dueño.
1 {gato; not perro; loro} 2 :- dueño.
```

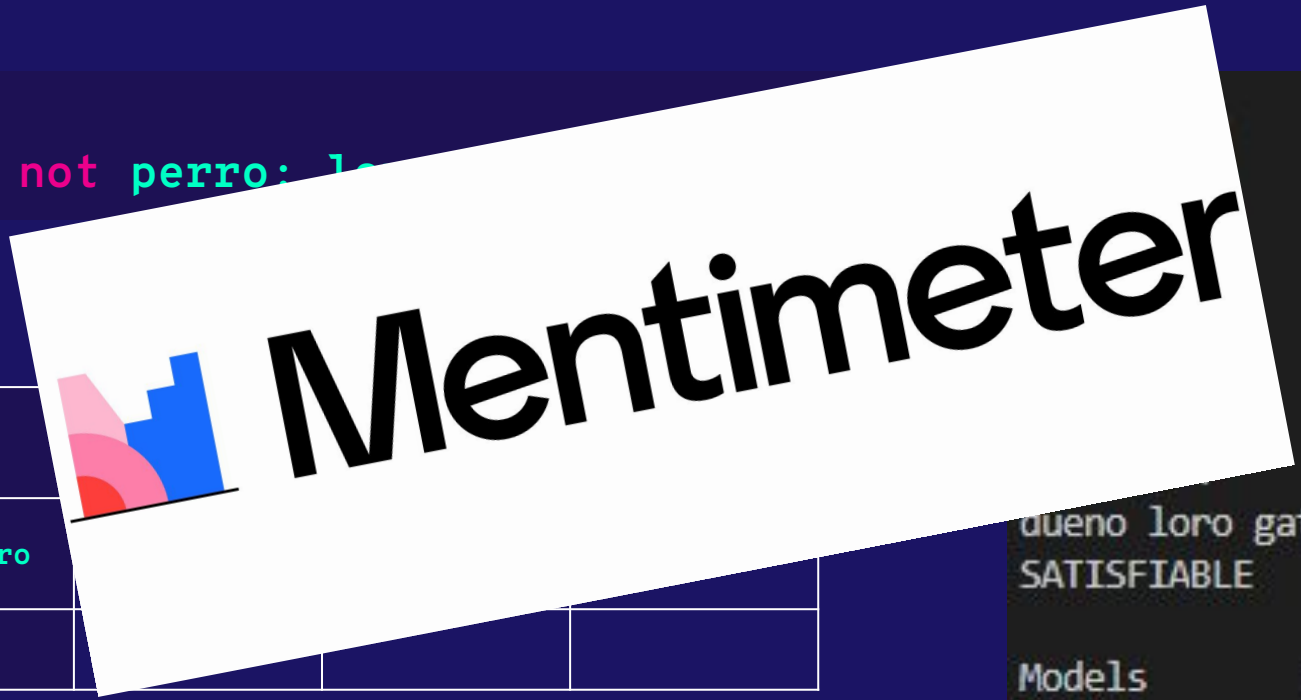
	gato	not perro	loro
gato	M = {dueño, gato}		M = {dueño, gato, loro}
not perro		M = {dueño}	M = {dueño, loro}
loro			

```
Solving...
Answer: 1
dueno
Answer: 2
dueno gato
Answer: 3
dueno loro
Answer: 4
dueno loro gato
SATISFIABLE

Models       : 4
Calls        : 1
```

Veamos el desarrollo…

```
dueño.
1 {gato; not perro; lo
```

gato			
not perro			
loro			




```
dueno loro gato
SATISFIABLE

Models       : 4
Calls        : 1
```

Ahora a Modelar!

Modelemos el siguiente problema:

1) La primera ampolleta se prenderá si no es de día (es de tarde o de noche).
2) La segunda ampolleta se prenderá si la 4ta ampolleta está encendida.
3) La tercera ampolleta se prenderá si es de noche y la quinta ampolleta está encendida.
4) La cuarta ampolleta se prenderá si algún par de ampolleta consecutivas están apagadas o bien, si no es de noche.
5) La quinta ampolleta se prenderá si hay más de 3 ampolletas encendidas.
6) La sexta ampolleta se prenderá si hay al menos una ampolleta encendida, además, si cumple esta regla, encenderá otra ampolleta que no cumpla su respectiva regla.
7) La séptima ampolleta se prenderá si hay al menos una ampolleta que no esté junto a ella que no cumpla su respectiva regla.
8) La octava ampolleta se prenderá si la 1ra y la 3ra ampolleta tienen el mismo estado (encendido o apagado).

Extra: Maximiza el número de ampolletas encendidas

Anexo

Algunos comentarios durante la ayudantía de hoy:

- Podemos visualizar todas las soluciones de un modelo mediante la línea `clingo -n 0 nombre_archivo.lp`, con el número de soluciones que deseemos observar (si escribimos 0 nos retornará todas ellas).

- La función `#count{ X: predicado(X) }` nos retorna el número de instancias que cumplen un cierto predicado.

- La función `#maximize{ C: count_algo(C) }` nos permite maximizar el valor de la variable X de algún predicado que pueda adoptar algún valor numérico (tal como `count_algo(C) :- C = #count{ X: predicado(X)}`).

- Cada vez que afirmamos `not var` en nuestro programa, automáticamente, todo modelo que se derive de esta rama de su ejecución y que tenga reglas que hacen a `var` pertenecer al modelo, se descartan.

No duden en hacerme llegar sus dudas sobre la materia a dflorea@uc.cl
o en las issues del repo :)