Inteligencia Artificial II

TP 2 - Guía de resolución

Sistemas expertos con PROLOG

Martin Marchetta
martin.marchetta@ingenieria.uncuyo.edu.ar

UNCUYO
UNIVERSIDAD NACIONAL DE CUYO

FACULTAD DE INGENIERÍA

LABSIN

Unificación


```
presion(Tuberia, 60) ≡ presion(t1, X)
```

Unificación

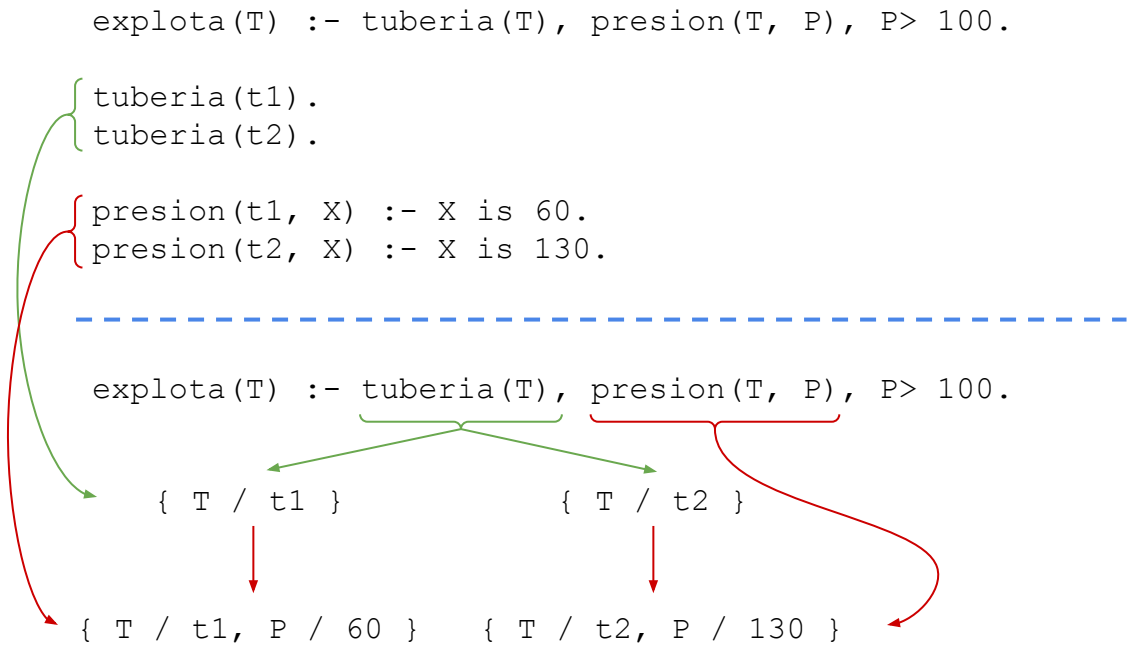
```
presion(Tuberia, 60) ≡ presion(t1, X)
```

```
{ Tuberia / t1, X / 60 }
```

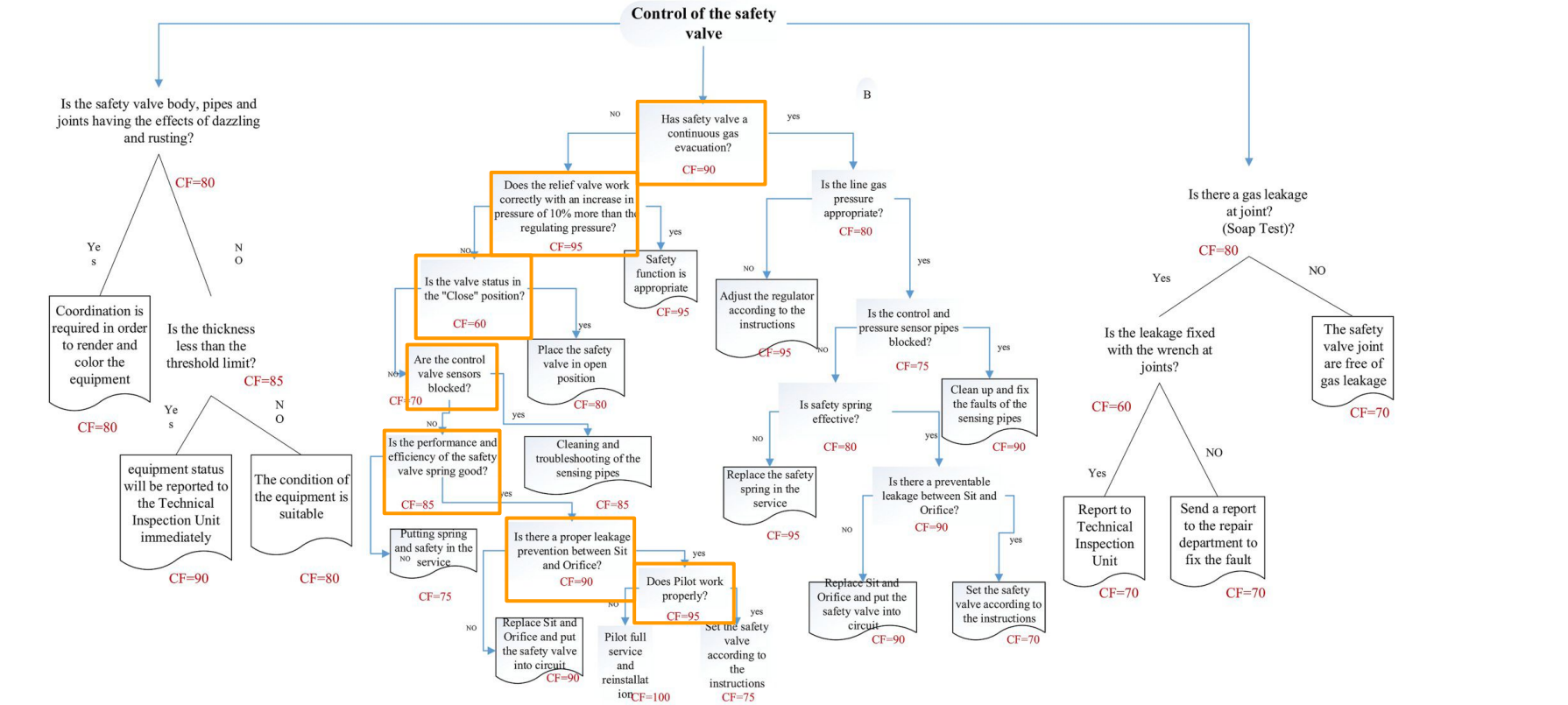
Unificación

Proceso de encontrar las sustituciones que hagan que 2 expresiones lógicas se hagan idénticas. Es una operación esencial de cualquier algoritmo basado en el conocimiento (demostradores de teoremas, lenguajes de programación lógica, planning, etc)

Unificación en PROLOG

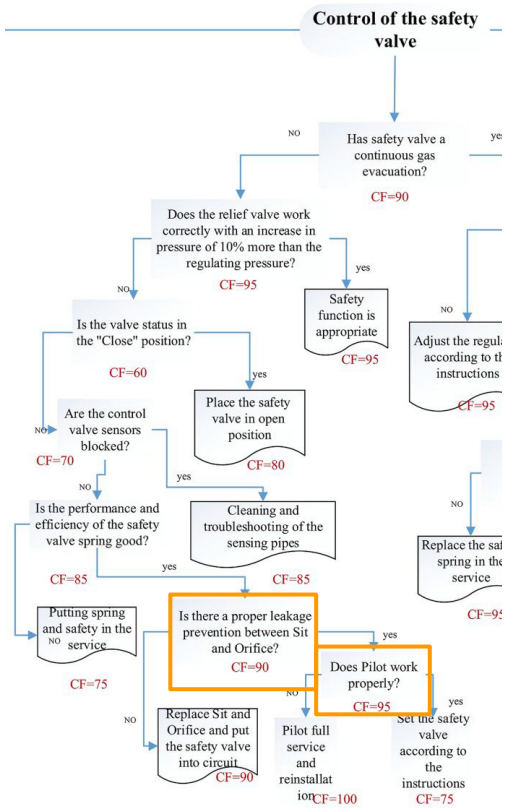


Sistema experto en PROLOG



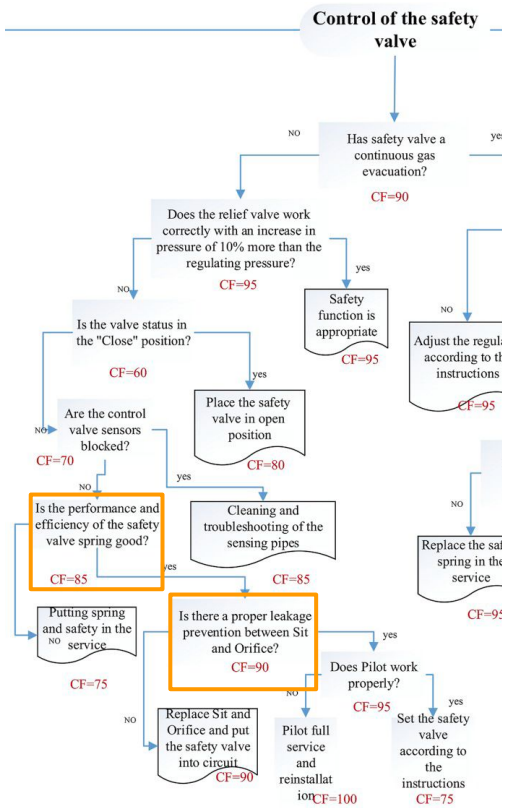
Sistema experto en PROLOG

```
verificar(piloto) :-
    estado(piloto, ok), writeln('Todo OK').

verificar(piloto) :-
                estado(piloto, desconocido),
                (
                    (
                       estado(leakage_prevention_between_sit_and_orifice, ok),
                       writeln('Verificar Pilot')
                    )
                    ;
                    verificar(leakage_prevention_between_sit_and_orifice)
                ).
```



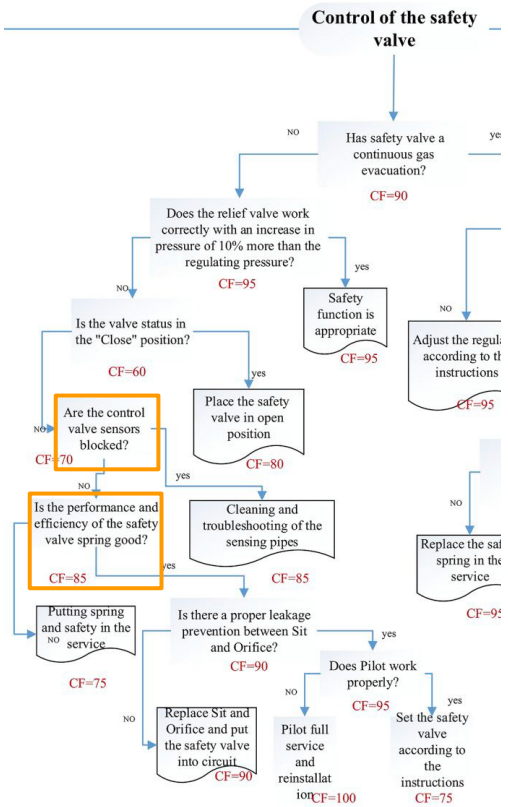
Sistema experto en PROLOG

```
verificar(piloto) :-
    estado(piloto, ok), writeln('Todo OK').

verificar(piloto) :-
                estado(piloto, desconocido),
                (
                    (
                       estado(leakage_prevention_between_sit_and_orifice, ok),
                       writeln('Verificar Pilot')
                    )
                    ;
                    verificar(leakage_prevention_between_sit_and_orifice)
                ).

verificar(leakage_prevention_between_sit_and_orifice) :-
                estado(leakage_prevention_between_sit_and_orifice, desconocido),
                (
                    (
                       estado(safety_valve_spring, ok),
                       writeln('Verificar leakage prevention between sit and ...')
                    )
                    ;
                    verificar(safety_valve_spring)
                ).
```



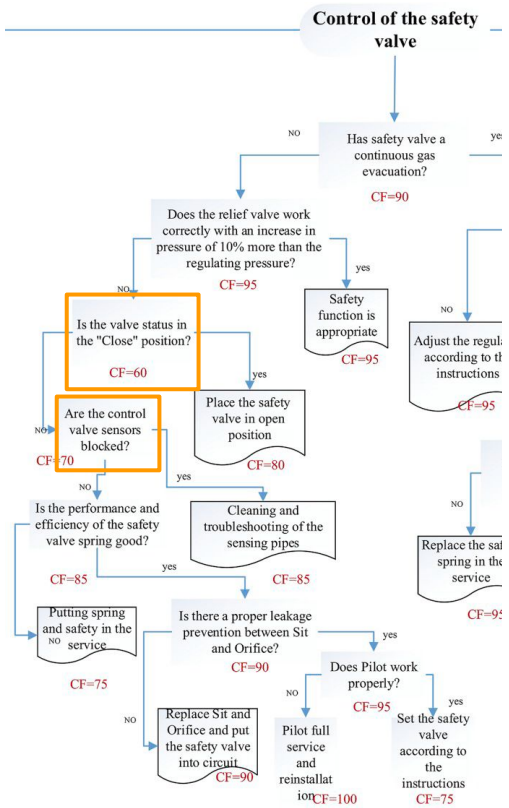
Sistema experto en PROLOG

```
verificar(leakage_prevention_between_sit_and_orifice) :-
                estado(leakage_prevention_between_sit_and_orifice, desconocido),
                (
                    (
                       estado(safety_valve_spring, ok),
                       writeln('Verificar leakage prevention between sit and ...')
                    )
                    ;
                    verificar(safety_valve_spring)
                ).

verificar(safety_valve_spring) :-
                estado(safety_valve_spring, desconocido),
                (
                    (
                       estado(control_valve_sensors_blocked, no),
                       writeln('Verificar safety valve spring')
                    )
                    ;
                    verificar(control_valve_sensors_blocked)
                ).
```



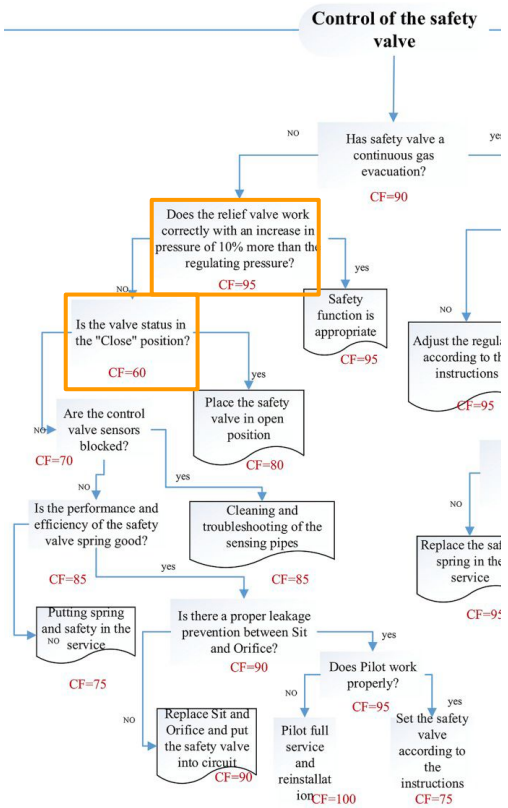
Sistema experto en PROLOG

```
verificar(safety_valve_spring) :-
                estado(safety_valve_spring, desconocido),
                (
                    (
                       estado(control_valve_sensors_blocked, no),
                       writeln('Verificar safety valve spring')
                    )
                    ;
                    verificar(control_valve_sensors_blocked)
                ).

verificar(control_valve_sensors_blocked) :-
                estado(control_valve_sensors_blocked, desconocido),
                (
                    (
                       estado(valve_status_closed, no),
                       writeln('Verificar control valve sensors blocked')
                    )
                    ;
                    verificar(valve_status_closed)
                ).
```



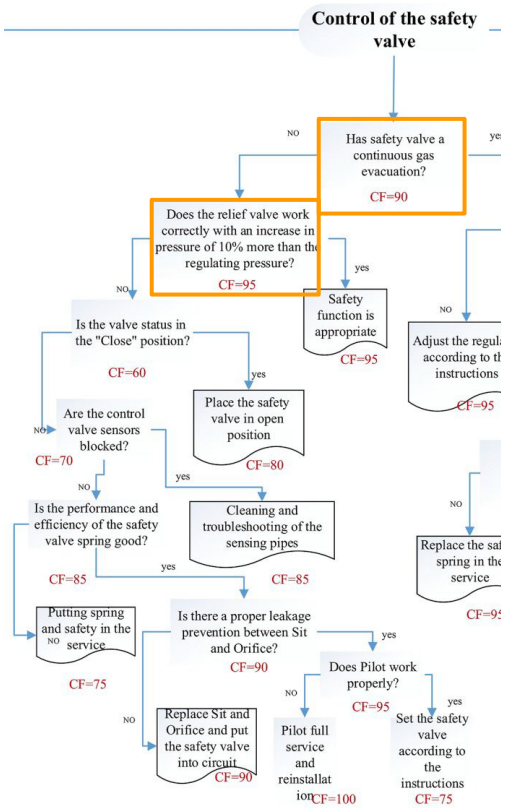
Sistema experto en PROLOG

```
verificar(control_valve_sensors_blocked) :-
                estado(control_valve_sensors_blocked, desconocido),
                (
                    (
                       estado(valve_status_closed, no),
                       writeln('Verificar control valve sensors blocked')
                    )
                    ;
                    verificar(valve_status_closed)
                ).

verificar(valve_status_closed) :-
                estado(valve_status_closed, desconocido),
                (
                    (
                       estado(relief_valve_ok_with_10_percent_more_pressure, no),
                       writeln('Verificar valve status "Close"')
                    )
                    ;
                    verificar(relief_valve_ok_with_10_percent_more_pressure)
                ).
```



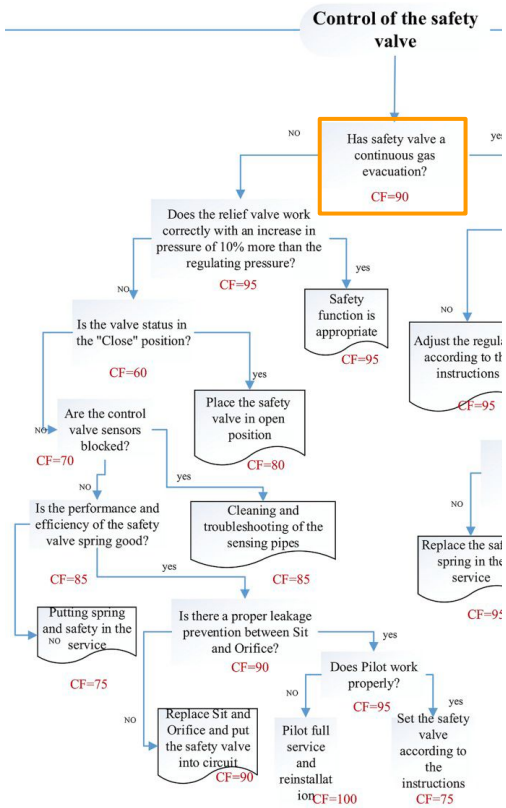
Sistema experto en PROLOG

```
verificar(valve_status_closed) :-
                estado(valve_status_closed, desconocido),
                (
                    (
                       estado(relief_valve_ok_with_10_percent_more_pressure, no),
                       writeln('Verificar valve status "Close"')
                    )
                    ;
                    verificar(relief_valve_ok_with_10_percent_more_pressure)
                ).

verificar(relief_valve_ok_with_10_percent_more_pressure) :-
                estado(relief_valve_ok_with_10_percent_more_pressure, desconocido),
                (
                    (
                       estado(safety_valve_has_continuous_evacuation, no),
                       writeln('Verificar relief valve works correctly with +10%...')
                    )
                    ;
                    verificar(safety_valve_has_continuous_evacuation)
                ).
```



Sistema experto en PROLOG

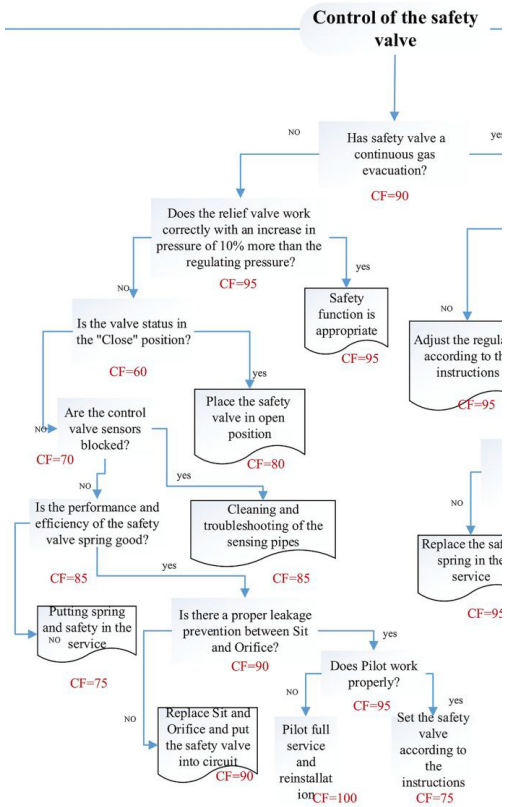
```
verificar(relief_valve_ok_with_10_percent_more_pressure) :-
                estado(relief_valve_ok_with_10_percent_more_pressure, desconocido),
                (
                    (
                       estado(safety_valve_has_continuous_evacuation, no),
                       writeln('Verificar relief valve works correctly with +10%...')
                    )
                    ;
                    verificar(safety_valve_has_continuous_evacuation)
                ).


verificar(safety_valve_has_continuous_evacuation) :-
                estado(safety_valve_has_continuous_evacuation, desconocido),
                writeln('Verificar safety valve has continuous evacuation').
```



Sistema experto en PROLOG

Ground facts

```
estado(piloto, desconocido).
estado(leakage_prevention_between_sit_and_orifice, desconocido).
estado(safety_valve_spring, desconocido).
estado(control_valve_sensors_blocked, desconocido).
estado(valve_status_closed, desconocido).
estado(relief_valve_ok_with_10_percent_more_pressure, desconocido).
estado(safety_valve_has_continuous_evacuation, desconocido).
```



Inteligencia Artificial II

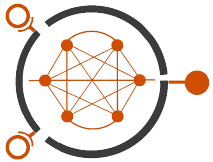
TP 2 - Guía de resolución

Planning

Martin Marchetta
martin.marchetta@ingenieria.uncuyo.edu.ar

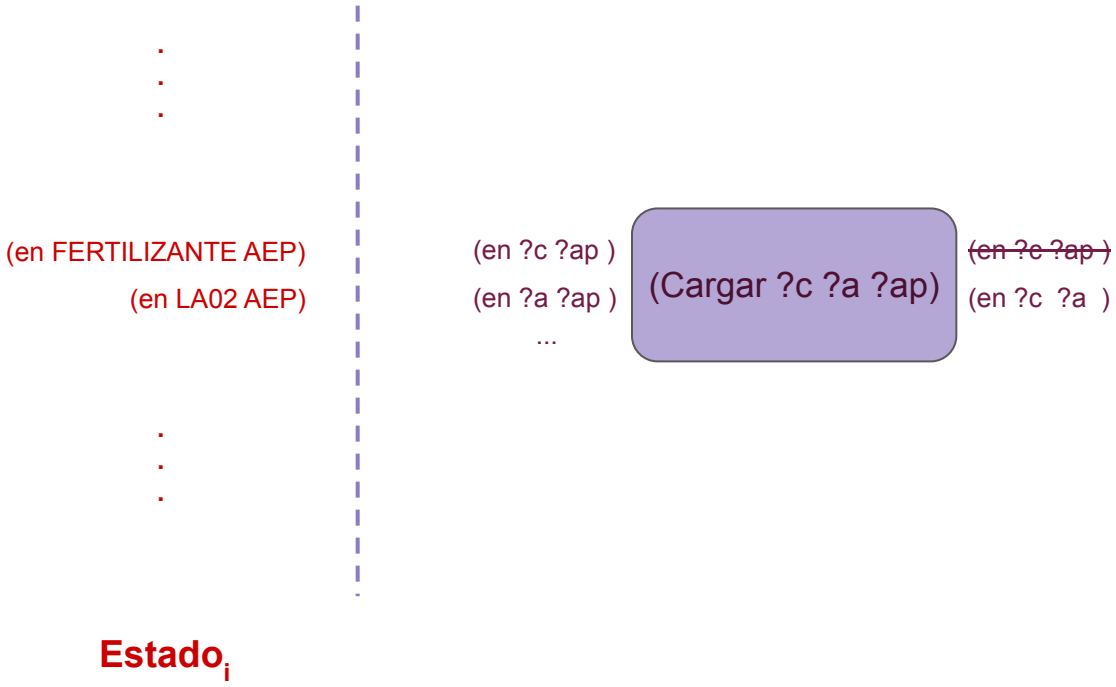
UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO

FACULTAD
DE INGENIERÍA

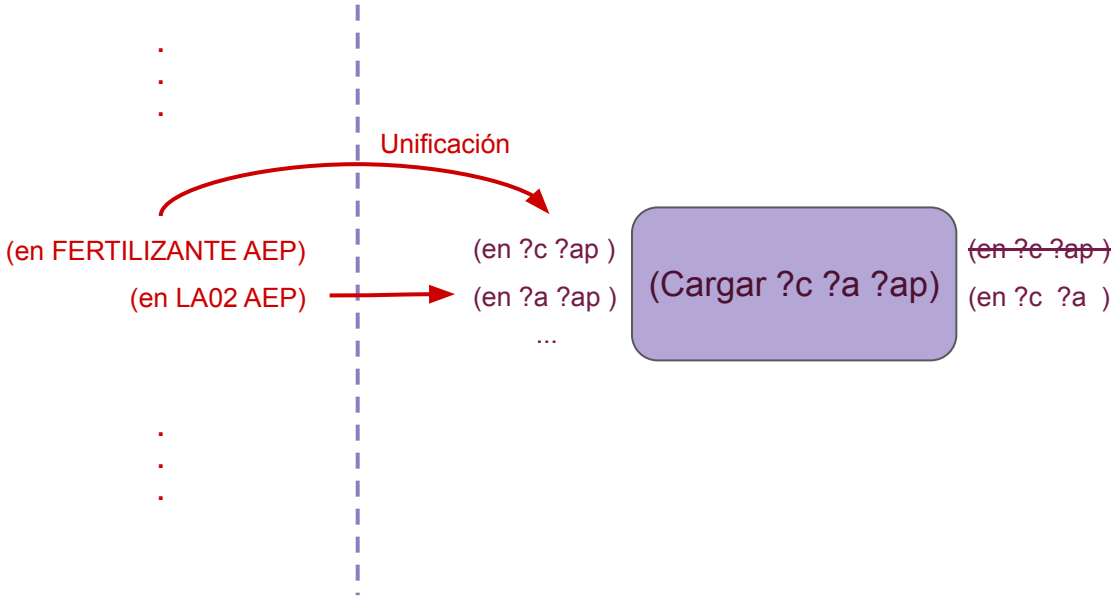


LABSIN

Instanciación de operadores



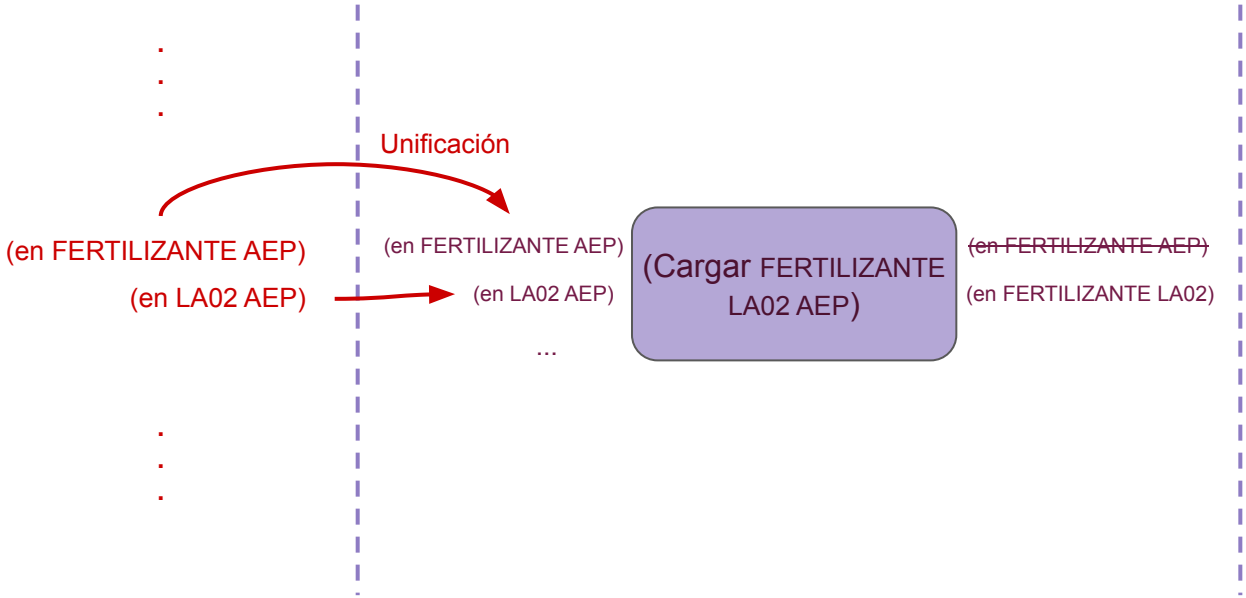
Instanciación de operadores



Estado$_i$

Estado$_{i+1}$

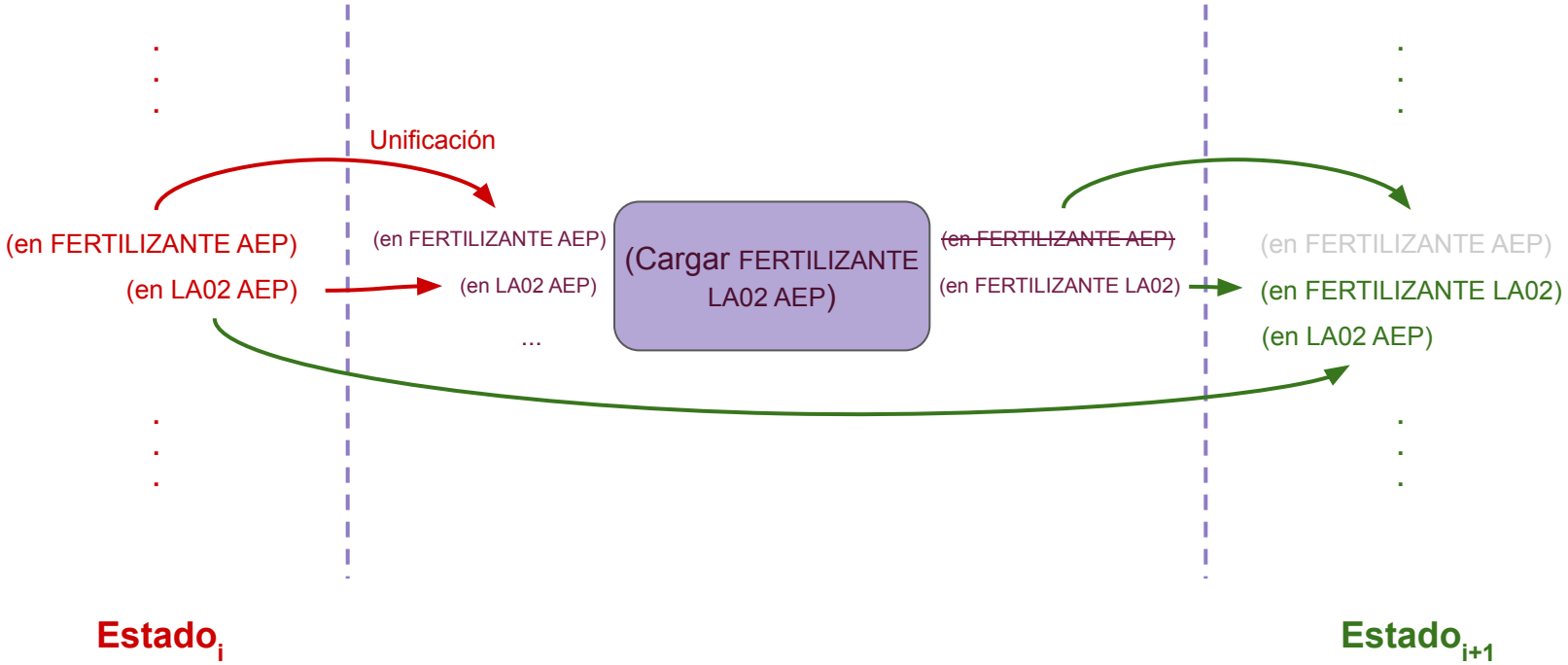
Instanciación de operadores



Estado$_i$

Estado$_{i+1}$

Instanciación de operadores



Dominio de transporte aéreo

Dominio
(invariante para distintas instancias)

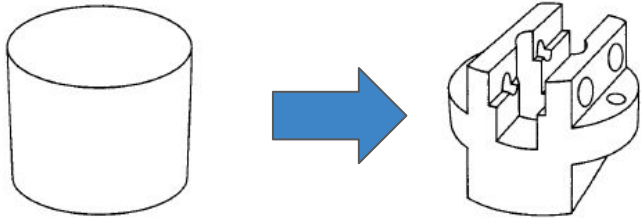
```
(define (domain aviones)
(:predicates
   (en ?a ?b)
   (avion ?a)
   (carga ?c)
   (aeropuerto ?a)
)

...

(:action cargar
 :parameters ( ?c ?a ?ap)
 :precondition
   (and (en ?c ?ap) (en ?a ?ap)
        (carga ?c) (avion ?a) (aeropuerto ?ap)
   )
 :effect
   (and
       (en ?c ?a)
       (not (en ?c ?ap))
   )
)

...
```

Problema
(estado inicial y objetivos concretos)

```
(define (problem carga-aerea)
   (:domain aviones)
   (:objects
       LA01
       AA01
       MDZ
       AEP
       FERTILIZANTE
       AUTOPARTES
        ...
   )
   (:init
       (avion LA01)
       (aeropuerto MDZ)
       (carga FERTILIZANTE)
       (carga AUTOPARTES)
       (en LA01 MDZ)
       (en AUTOPARTES COR)
       ...)
   (:goal
       (and
           (en FERTILIZANTE SFN)
           (en AUTOPARTES AEP)
       ...))
)
```

Dominio CAPP

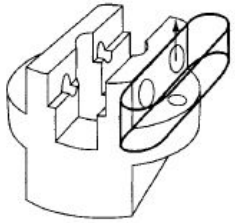
PROBLEMA



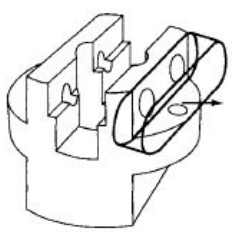
Raw material

Product

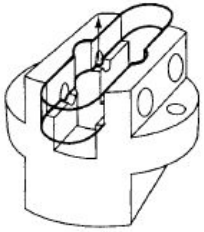
ALGUNAS FEATURES



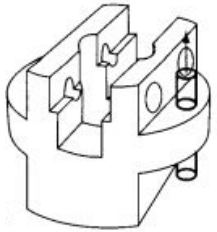
Step A (Opción 1)



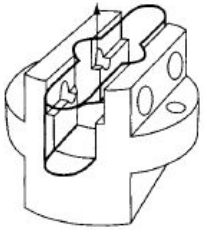
Step A (Opción 2)



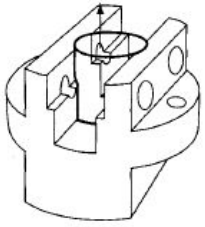
Slot A



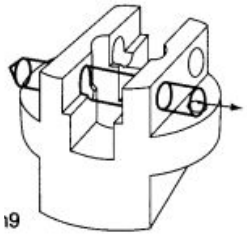
Through Hole A



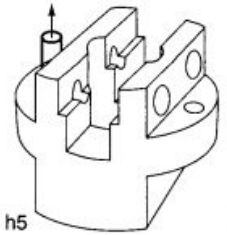
Slot B



Blind Hole A



Through Hole B



Through Hole C

Dominio CAPP

Dominio

- Esquemas de acción son
  - Operaciones de maquinado
    - Precondiciones
      - Presencia de cierta feature
      - Setup adecuado
      - Dependencia de la feature cumplida
    - Efectos
      - Feature fabricada
    - Ejemplos
      - Operaciones de fabricación
        - Fresado
        - Torneado
        - Taladrado
  - Setups
    - Orientación de pieza
    - Cambio de herramienta
    - Etc.
    - Ejemplos
      - Orientación de pieza
      - Cambio de herramienta

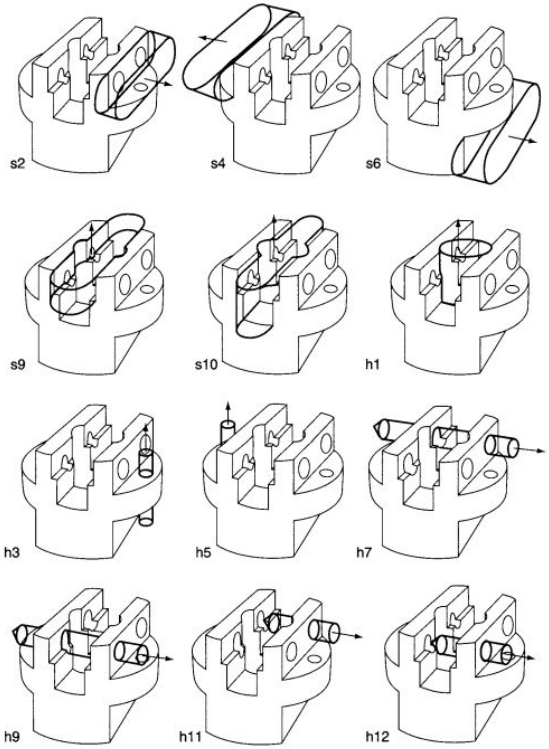
Problema

- Estado inicial
  - Features
    - Tipo
    - Orientacion
  - Dependencias de features
    - Algunas features requieren que otras se fabriquen antes
- Estado final
  - Lista de features fabricadas

Dominio CAPP

UNCUYO UNIVERSIDAD NACIONAL DE CUYO · FACULTAD DE INGENIERÍA · LABSIN

Features



Inteligencia Artificial II

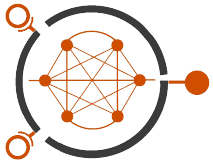
TP 2 - Guía de resolución

Fuzzy Logic

Martin Marchetta
martin.marchetta@ingenieria.uncuyo.edu.ar

UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO

FACULTAD
DE INGENIERÍA



LABSIN

Base de reglas

- Una herramienta de modelado: Fuzzy Abstract Machine (FAM)

FAM

Theta / *Theta'*	NG	NP	Z	PP	PG
NG	?	?	PG	PP	?
NP	?	?	?	Z	?
Z	?	?	Z	?	NG
PP	?	?	NP	NG	NG
PG	?	?	?	NG	?

Reglas equivalentes

…

Theta is PP **and** Theta' is NG **then** F is PP

…

Theta is PG **and** Theta' is PP **then** F is NG

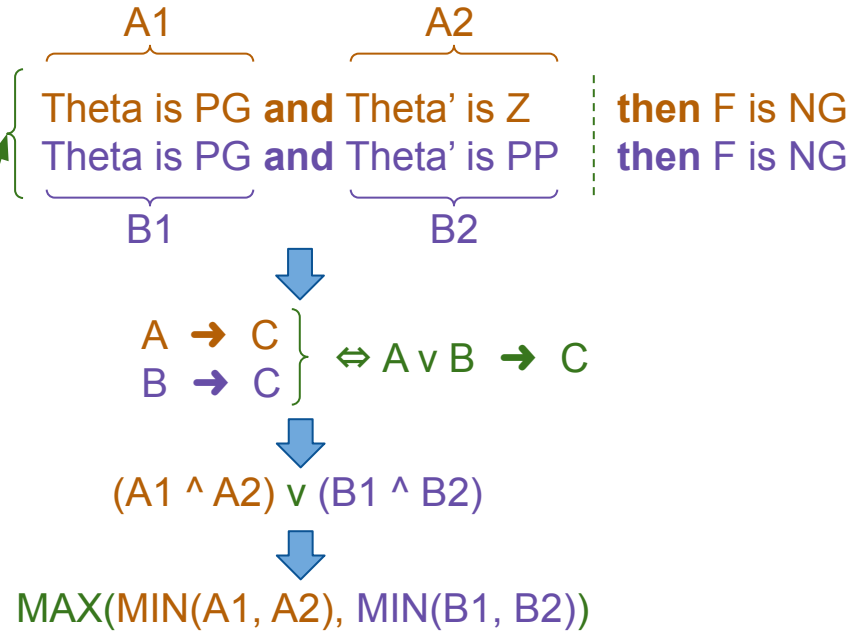
…

Inferencia

- Procesamiento de reglas con el ***mismo consecuente***

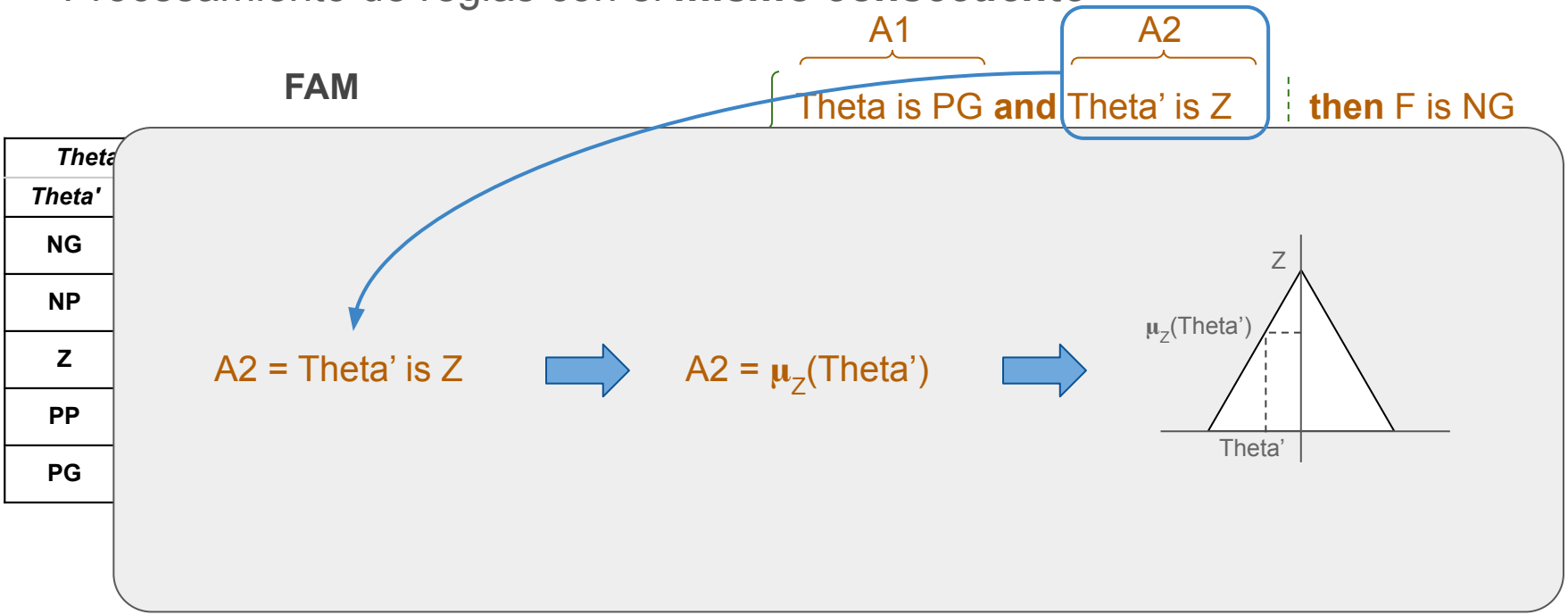
FAM

Theta / *Theta'*	NG	NP	Z	PP	PG
NG	?	?	PG	PP	?
NP	?	?	?	Z	?
Z	?	?	Z	?	NG
PP	?	?	NP	NG	NG
PG	?	?	?	NG	?



Inferencia

- Procesamiento de reglas con el ***mismo consecuente***



Inferencia

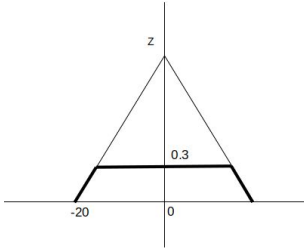
- **Combinación de consecuentes** para obtener conjuntos borrosos de salida

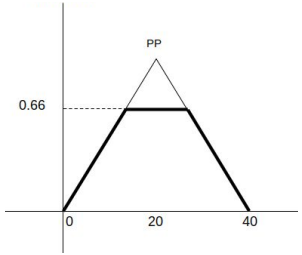
FAM

Theta / *Theta'*	NG	NP	Z	PP	PG
NG	?	?	PG	PP	?
NP	?	?	?	Z	?
Z	?	?	Z	?	NG
PP	?	?	NP	NG	NG
PG	?	?	?	NG	?

Theta is PP **and** Theta' is NP **then** F is Z

Theta is PP **and** Theta' is NG **then** F is PP





- Normalmente relación de conjunción entre oraciones lógicas
- En fuzzy logic hay interpretaciones alternativas
  - Conjución (ej: MIN de los conjuntos borrosos de salida)
  - Disyunción (ej: MAX de los conjuntos borrosos de salida)
- Disyunción es la más usada (participación más "equitativa" de los conjuntos borrosos de salida)

Inferencia

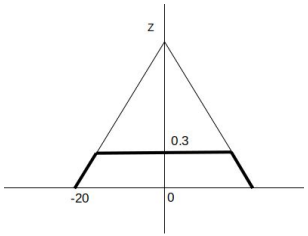
- **Combinación de consecuentes** para obtener conjuntos borrosos de salida

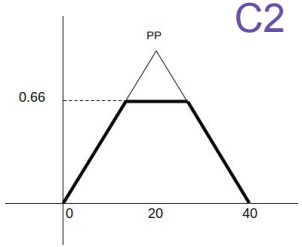
FAM

Theta / *Theta'*	NG	NP	Z	PP	PG
NG	?	?	PG	PP	?
NP	?	?	?	Z	?
Z	?	?	Z	?	NG
PP	?	?	NP	NG	NG
PG	?	?	?	NG	?

Theta is PP **and** Theta' is NP **then** F is Z (C1)

Theta is PP **and** Theta' is NG **then** F is PP (C2)





$\mu_{MAX(C1, C2)}$