15

1/4

1	2	3	4
B-	B	B-	R

A Julián

①

a)

gana(?E1, ?E2)

(sabemos q' ya esta función considera todas las comb. válidas.)

gana(E1, E2) :- probabilidadDeGanar(E1, E2, P), P > 0.5.

empatan(?E1, ?E2)
empatan(E1, E2) :- probabilidadDeGanar(E1, E2, P), P = 0.5.

pierde(?E1, ?E2)
pierde(E1, E2) :- not(gana(E1, E2)), not(empatan(E1, E2)).

b)

puntaje(+E, +G, -P)

puntaje(E, [], 0).

puntaje(E, [X|XS], P) :- gana(E, X), puntaje(E, XS, P'), P = P' + 3.

puntaje(E, [X|XS], P) :- empatan(E, X), puntaje(E, XS, P'), P = P' + 1.

puntaje(E, [X|XS], P) :- pierde(E, X), puntaje(E, XS, P).

c) paraAOctavos(+E, ?G)

si G no está instanciada, puntaje no anda. error grave

paraAOctavos(E, G) :- puntaje(E, G, PuntajeE), not(hayAlguienCosMarPuntos(G, PuntajeE)).

hayAlguienCosMarPuntos(+G, +Puntos, +E)

hayAlguienCosMarPuntos(G, Puntos) :- member(Equipo, G), Equipo \= E, ← no hace falta.
borrar(G, Equipo, G'),
~~puntaje(Equipo~~ append([E], G', G''),
puntaje(Equipo, G'', N), N > Puntos.

("borrar" es el q' definimos en el 8.III de la práctica 5)

d) vamosArgentina(?G)

~~vamosArgentina([X|Y|Z|W|[]]) :-~~

vamosArgentina(G) :- ~~member(argentina, G)~~ participantes(Lista),
member(E1, Lista), member(E2, Lista), member(E3, Lista),
argentina \= E1, argentina \= E2, argentina \= E3, E1 \= E2, E1 \= E3,
E2 \= E3, permutacion([E1, E2, E3, argentina], G),
paraAOctavos(argentina, [E1, E2, E3]).

("permutación" es la q' definimos en el 8.V de la práctica 5).

es una de las características fundamentales del paradigma.

todo lo de permutación se debería haber resuelto con la reversibilidad de paraAOctavos.

(2) (a)

Antes pasar las afirmaciones a forma clausal

- $\forall x \forall y \; A(x,y) \supset A(y,x)$

$\equiv \forall x \forall y \; \neg A(x,y) \vee A(y,x) \equiv \{ \underbrace{\{\neg A(x_1,y_1), A(y_2,x_2)\}}_{C_1} \}$ ✓

- $\forall x \forall y \forall z \; (A(x,y) \wedge A(y,z)) \supset A(x,z)$

$\equiv \forall x \forall y \forall z \; \neg A(x,y) \vee \neg A(y,z) \vee A(x,z) \equiv \{ \underbrace{\{\neg A(x_2,z_2), \neg A(y_2,z_2), A(x_2,z_2)\}}_{C_2} \}$ ✓

Ahora vemos que la negación de lo q' queremos probar es insatisfacible

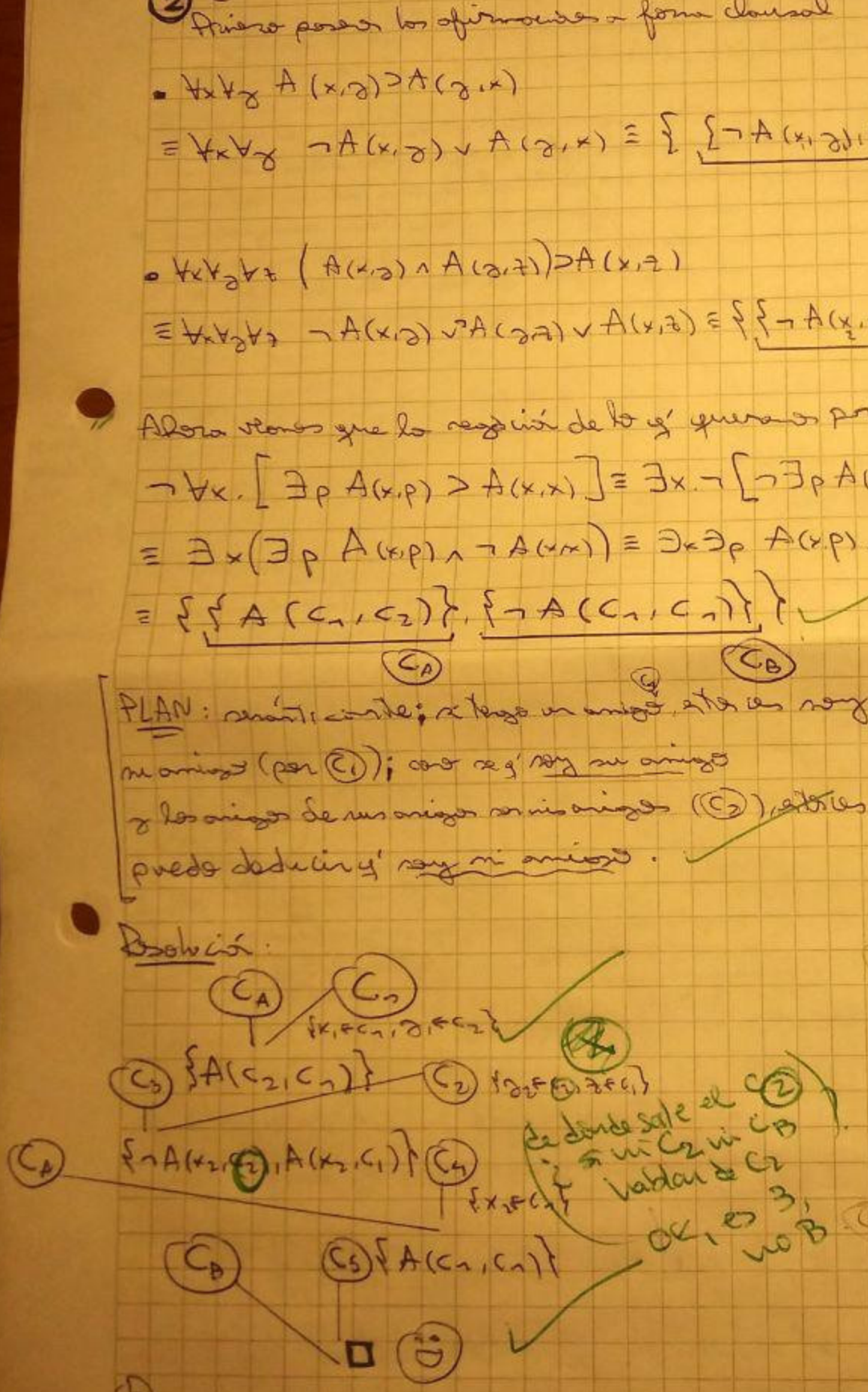
$\neg \forall x. [\exists y \; A(x,y) \supset A(x,x)] \equiv \exists x. \neg [\neg \exists y \; A(x,y) \vee A(x,x)]$

$\equiv \exists x (\exists y \; A(x,y) \wedge \neg A(x,x)) \equiv \exists x \exists y \; A(x,y) \wedge \neg A(x,x)$

$\equiv \{ \underbrace{\{A(c_1,c_2)\}}_{C_A}, \underbrace{\{\neg A(c_1,c_1)\}}_{C_B} \}$ ✓

PLAN: sabemos cierto; x luego no amigo, ¿a los soy no amigos (por C_1); con x q' soy no amigo y los amigos de mis amigos son mis amigos (C_2), ¿a los puedo deducir q' soy mi amigo.

Resolución:



(b) Si bien es cierto q' todas las cláusulas tenían a lo sumo un lit. positivo y la resolución fue lineal (de hecho lineal), para usarse C_A dos veces en la resolución a partir de C_B. Luego no es SLD.

15 4/4

④ ⓐ

$$\frac{\{S'_1,\dots,S'_m\} \subseteq \{S_1,\dots,S_n\} \qquad S_i = S'_j \Rightarrow \sigma_i \to r_i <: \sigma_j \to r_j}{API[(S_1,\sigma_1\to r_1),\dots,(S_n,\sigma_n\to r_n)] <: API[(S'_1,\sigma'_1\to r'_1),\dots,(S'_m,\sigma'_m\to r'_m)]} \ (*)$$

/ espero

recibo

- La primera condición se debe cumplir porque las nuevas versiones tienen todo lo que tenía la vieja, pero además puede tener más cosas. Si no tuviera algo de lo de la vieja habría un problema
- La segunda condición es porque si la nueva implementa algo que estaba en la vieja, está puede tener más funcionalidades, pero NO al revés. La nueva no debería no responder algo que la vieja sí podía. (Por ejemplo, si espero un Alumno y le paso una persona, esta no va a saber responder a "dame nro de libreta universitaria")

TAPP dice que el tipo de la izq. de la → y el del argumento tienen que ser exactamente el mismo.

ⓑ

$$\frac{\emptyset \vdash (\lambda a : API[(inc., Nat \to Int].(con\langle a, inc\rangle\ 0)]\ expon\langle inc = \lambda x : Nat.succ(x), notId = \lambda x : Nat.x\rangle : Int}{\emptyset \vdash \lambda a : API[(inc, N\to I].(con\langle a, inc\rangle\ 0) : \underbrace{API[(inc, N\to I)]}_{①} \to I \quad \emptyset \vdash expon\langle inc = \lambda x : N.succ(x), notId = \lambda x : N.x\rangle : \underbrace{API[(inc, N\to N), (notId, N\to N)]}_{②}}$$

(T-App) N.Nat, I.Int

(T-Abs)

$$\frac{\Gamma = \{a : API[(inc, N\to I)]\} \vdash con\langle a, inc\rangle\ 0 : I}{\Gamma \vdash con\langle a, inc\rangle : N \to I}$$

(T-App)

$$\frac{\Gamma \vdash con\langle a, inc\rangle : N \to I \qquad \Gamma \vdash 0 : N \qquad N <: N}{\checkmark \quad (\text{IT-Zero}) \quad \checkmark}$$

(T-Repl)

(T-Consumir)

$$\frac{\Gamma \vdash a : API[(inc, N\to I)]}{\{a : API(inc, N\to I)\} \in \Gamma}$$

(T-Var) ✓

(T-Exponer)

$$\frac{\emptyset \vdash \lambda x : N.succ(x) : N \to N \qquad \emptyset \vdash \lambda x : N.x : N \to N}{\{x : N\} \vdash x : N}$$

(T-Abs)

$$\frac{\{x : N\} \vdash succ(x) : N}{\{x : N\} \vdash x : N}$$

(T-Succ)

$$\frac{\{x : N\} \vdash x : N}{\{x : N\} \in \{x : N\}}$$

(T-Var) ✓

(T-Abs)

$$\frac{\{x : N\} \vdash x : N}{\{x : N\} \in \{x : N\}}$$

(T-Var) ✓

② <: ①

Estás mezclando reglas.

② <: ①

$$\frac{API[(inc, N\to N), notId(N\to N)] <: API[(inc, N\to I)]}{\{inc\} \subseteq \{inc, notId\}} \ (*)$$

✓

$$\frac{N \to N <: N \to I}{N \supseteq N \qquad N <: I}$$

(S-Func/Arrow)

(T-Repl) ✓ ✓ (T-NotSnt)

(En realidad, hay que escribir la derivación de abajo para arriba ⓑ).

Está mal el pasaje de tipado a subtipado. Mezclaste reglas de tipado con T subs.