Άσκηση 1

1.

$(p \Leftrightarrow \neg q) \Rightarrow ((r \wedge s) \vee t)$ →

$((p \Rightarrow \neg q) \wedge (\neg q \Rightarrow p)) \Rightarrow ((r \wedge s) \vee t)$ →

$((\neg p \vee \neg q) \wedge (q \vee p)) \Rightarrow ((r \wedge s) \vee t)$ →

$(\neg ((\neg p \vee \neg q) \wedge (q \vee p))) \vee ((r \wedge s) \vee t)$ →

$((p \wedge q) \vee (\neg q \wedge \neg p)) \vee ((r \wedge s) \vee t)$ →

$((p \vee \neg q) \wedge (p \vee \neg p) \wedge (q \vee \neg q) \wedge (q \vee \neg p)) \vee ((r \wedge s) \vee t)$ →

$((p \vee \neg q) \wedge (p \vee \neg p) \wedge (q \vee \neg q) \wedge (q \vee \neg p)) \vee ((r \vee t) \wedge (s \vee t))$ →

$(((r \vee t) \wedge (s \vee t)) \vee (p \vee \neg q))) \wedge ((r \vee t) \wedge (s \vee t)) \wedge (((r \vee t) \wedge (s \vee t)) \vee (q \vee \neg p)))$
→

$(r \vee t \vee p) \wedge (r \vee t \vee \neg q) \wedge (s \vee t \vee p) \wedge (s \vee t \vee \neg q) \wedge (r \vee t) \wedge (s \vee t) \wedge (r \vee t \vee q) \wedge (r \vee t \vee \neg p) \wedge (s \vee t \vee q) \wedge (s \vee t \vee \neg p)$ →

{[r, t, p], [r, t, ¬ q], [s, t, p], [s, t, ¬ q], [r, t], [s, t], [r, t, q], [r, t, ¬ p], [s, t, q], [s, t, ¬ p]}

2.

$(\forall x.\ \forall y.\ \exists z.\ q(x, y, z) \vee \exists x.\ \forall y.\ p(x, y)) \wedge \neg (\exists x.\ \exists y.\ p(x, y))$ →

$(\forall x.\ \forall y.\ \exists z.\ q(x, y, z) \vee \exists x.\ \forall y.\ p(x, y)) \wedge (\forall x.\ \forall y.\ \neg p(x, y))$ →

$(\forall x_1.\ \forall y_1.\ \exists z_1.\ q(x_1, y_1, z_1) \vee \exists x_2.\ \forall y_2.\ p(x_2, y_2)) \wedge (\forall x_3.\ \forall y_3.\ \neg p(x_3, y_3))$ →

$(\forall x_1.\ \forall y_1.\ q(x_1, y_1, f(x_1,y_1)) \vee \forall y_2.\ p(c, y_2)) \wedge (\forall x_3.\ \forall y_3.\ \neg p(x_3, y_3))$ →

$\forall x_1.\ \forall y_1.\ \forall y_2.\ ((q(x_1, y_1, f(x_1,y_1)) \vee p(c, y_2))) \wedge (\forall x_3.\ \forall y_3.\ \neg p(x_3, y_3))$ →

$\forall x_1.\ \forall y_1.\ \forall y_2.\ \forall x_3.\ \forall y_3.\ ((q(x_1, y_1, f(x_1, y_1)) \vee p(c, y_2))) \wedge \neg p(x_3, y_3))$ →

$((q(x_1, y_1, f(x_1,y_1)) \vee p(c, y_2))) \wedge \neg p(x_3, y_3))$ →

{[$q(x_1, y_1, f(x_1,y_1))$, $p(c, y_2)$],[$\neg p(x_3, y_3)$]}

Άσκηση 2

<u>$1^η$ Πρόταση + $2^η$ Πρόταση:</u>

Η πρώτη πρόταση λέει ότι η R είναι ανακλαστική και η δεύτερη πρόταση λέει ότι η R είναι συμμετρική. Θέλουμε ένα μοντέλο που να ικανοποιεί τις πρώτες δύο και να μην ικανοποιεί τη μεταβατικότητα.

$$Δ^I = \{α, β , γ\}$$

$$R^I = \{(α,α), (β, β), (γ, γ), (α, β), (β, α), (α, γ), (γ, α)\}$$

Ικανοποιείται η ανακλαστική και η συμμετρική ιδιότητα αλλά δεν ικανοποιείται η μεταβατική ιδιότητα.
Η ανακλαστική ιδιότητα ισχύει γιατί έχουμε τα ζεύγη (α,α), (β, β), (γ, γ).

Η συμμετρική ιδιότητα ισχύει γιατί έχουμε τα ζεύγη (α, β), (β, α), (α, γ), (γ, α).

Η μεταβατική ιδιότητα δεν ισχύει γιατί ενώ έχουμε τα ζεύγη (β, α), (α, γ) απουσιάζει το ζεύγος (β, γ).

<u>$1^η$ Πρόταση + $3^η$ Πρόταση:</u>

Η πρώτη πρόταση λέει ότι η R είναι ανακλαστική και η τρίτη πρόταση λέει ότι η R είναι μεταβατική. Θέλουμε ένα μοντέλο που να ικανοποιεί τις πρώτες δύο και να μην ικανοποιεί τη συμμετρικότητα.

$$Δ^I = \{α, β , γ\}$$

$$R^I = \{(α,α), (β, β), (γ, γ), (β, α), (α, γ), (β, γ)\}$$

Η ανακλαστική ιδιότητα ισχύει γιατί έχουμε τα ζεύγη (α,α), (β, β), (γ, γ).

Η μεταβατική ιδιότητα ισχύει γιατί έχουμε τα ζεύγη (β, α), (α, γ), (β, γ).

Η συμμετρική ιδιότητα δεν ισχύει γιατί ενώ έχουμε το ζεύγος (β, α) απουσιάζει το ζεύγος
(α, β).

<u>$2^η$ Πρόταση + $3^η$ Πρόταση:</u>

Η δεύτερη πρόταση λέει ότι η R είναι συμμετρική και η τρίτη πρόταση λέει ότι η R είναι μεταβατική. Θέλουμε ένα μοντέλο που να ικανοποιεί τις πρώτες δύο και να μην ικανοποιεί τη ανακλαστικότητα.

$$Δ^I = \{α, β , γ\}$$

$$R^I = \{(α, β), (β, α), (α, γ), (γ, α), (β, γ)\}$$

Η συμμετρική ιδιότητα ισχύει γιατί έχουμε τα ζεύγη (α, β), (β, α), (α, γ), (γ, α).

Η μεταβατική ιδιότητα ισχύει γιατί έχουμε τα ζεύγη (β, α), (α, γ), (β, γ).

Η ανακλαστική ιδιότητα δεν ισχύει διότι απουσιάζει το ζεύγος (γ, γ).

Άσκηση 3

1) $\forall x. \exists y. (A(x) \Rightarrow R(x, y) \wedge C(y))$

2) $\forall x. \exists y. (B(x) \Rightarrow S(y, x) \wedge D(y))$

3) $\forall x. (D(x) \Rightarrow A(x))$

4) $\forall x. \forall y. (S(x, y) \Rightarrow T(y, x))$

5) $\forall x. \forall y. \forall z. (T(x, y) \wedge R(y, z) \wedge C(z) \Rightarrow Q(x))$

Μετατρέπουμε τις προτάσεις της γνώσης μας K σε ΚΣΜ. Ονομάζουμε αλλιώς τις μεταβλητές στην καθεμία πρόταση και έχουμε τα παρακάτω.

1) $\neg A(x) \vee ((R(x, f(x)) \wedge C(f(x))) \rightarrow ((\neg A(x) \vee R(x, f(x))) \wedge (((\neg A(x) \vee C(f(x)))$

2) $\neg B(w) \vee (S(f(w), w) \wedge D(f(w))) \rightarrow (\neg B(w) \vee S(f(w), w)) \wedge (\neg B(w) \vee D(f(w))$

3) $\neg D(t) \vee A(t)$

4) $\neg S(s, r) \vee T(r, s)$

5) $(\neg T(q, p) \vee \neg R(p, z) \vee \neg C(z)) \vee Q(q) \rightarrow \neg T(q, p) \vee \neg R(p, z) \vee \neg C(z) \vee Q(q)$

Επίσης, μετατρέπουμε σε ΚΣΜ την άρνηση της $\forall x.(B(x) \Rightarrow Q(x))$

Έχουμε

$$\neg (\forall x.(B(x) \Rightarrow Q(x))) \rightarrow$$
$$\exists x \neg (\neg B(x) \vee Q(x)) \rightarrow$$
$$\exists x(B(x) \wedge \neg Q(x)) \rightarrow$$
$$B(a) \wedge \neg Q(a)$$

Το a είναι σταθερά, ενώ τα x, y, z, w, t, s, r, q, p είναι μεταβλητές.

Θέλουμε να δείξουμε το παρακάτω:

$(1),(2),(3),(4),(5) \models \forall x.(B(x) \Rightarrow Q(x))$

Εισάγουμε στη βάση της γνώσης μας και τους δύο όρους [B(a)], [¬Q(a)] και έχουμε τα παρακάτω. Από τους κόκκινους όρους κάθε φορά λαμβάνουμε ως αναλυθέν την επόμενη πρόταση.

Ξεκινάμε:

{[¬A(x), R(x, f(x))], [¬A(x), C(f(x))], [¬B(w), S(f(w), w)],[¬B(w), D(f(w)], [¬D(t), A(t)], [¬S(s, r), T(r, s)], [¬T(q, p), ¬R(p, z), ¬C(z), Q(q)], [B(a)], [¬Q(a)]} →

Αντικατάσταση w -> a

{[¬A(x), R(x, f(x))], [¬A(x), C(f(x))], [¬B(w), S(f(w), w)],[¬B(w), D(f(w)], [¬D(t), A(t)], [¬S(s, r), T(r, s)], [¬T(q, p), ¬R(p, z), ¬C(z), Q(q)], [B(a)], [¬Q(a)], [S(f(a), a)]} →

Αντικατάσταση s -> f(a), r -> a

{[¬A(x), R(x, f(x))], [¬A(x), C(f(x))], [¬B(w), S(f(w), w)],[¬B(w), D(f(w)], [¬D(t), A(t)], [¬S(s, r), T(r, s)], [¬T(q, p), ¬R(p, z), ¬C(z), Q(q)], [B(a)], [¬Q(a)], [S(f(a), a)], [T(a, f(a)]} →

Αντικατάσταση q -> a

{[¬A(x), R(x, f(x))], [¬A(x), C(f(x))], [¬B(w), S(f(w), w)], [¬B(w), D(f(w)], [¬D(t), A(t)], [¬S(s, r), T(r, s)], [¬T(q, p), ¬R(p, z), C(z), Q(q)], [B(a)], [¬Q(a)], [S(f(a), a)], [T(a, f(a)], [¬T(a, p), ¬R(p, z), ¬C(z)]} →

Αντικατάσταση w -> a

{[¬A(x), R(x, f(x))], [¬A(x), C(f(x))], [¬B(w), S(f(w), w)], [¬B(w), D(f(w)], [¬D(t), A(t)], [¬S(s, r), T(r, s)], [¬T(q, p), ¬R(p, z), C(z), Q(q)], [B(a)], [¬Q(a)], [S(f(a), a)], [T(a, f(a)], [¬T(a, p), ¬R(p, z), ¬C(z)], [D(f(a)]} →

Αντικατάσταση t -> f(a)

{[¬A(x), R(x, f(x))], [¬A(x), C(f(x))], [¬B(w), S(f(w), w)], [¬B(w), D(f(w)], [¬D(t), A(t)], [¬S(s, r), T(r, s)], [¬T(q, p), ¬R(p, z), C(z), Q(q)], [B(a)], [¬Q(a)], [S(f(a), a)], [T(a, f(a)], [¬T(a, p), ¬R(p, z), ¬C(z)], [D(f(a)], [A(f(a))]} →

Αντικατάσταση x -> f(a)

{[¬A(x), R(x, f(x))], [¬A(x), C(f(x))], [¬B(w), S(f(w), w)], [¬B(w), D(f(w)], [¬D(t), A(t)], [¬S(s, r), T(r, s)], [¬T(q, p), ¬R(p, z), C(z), Q(q)], [B(a)], [¬Q(a)], [S(f(a), a)], [T(a, f(a)], [¬T(a, p), ¬R(p, z), C(z)], [D(f(a)], [A(f(a))], [C(f(f(a)))]} →

Αντικατάσταση x -> f(a)

{[¬A(x), R(x, f(x))], [¬A(x), C(f(x))], [¬B(w), S(f(w), w)], [¬B(w), D(f(w)], [¬D(t), A(t)], [¬S(s, r), T(r, s)], [¬T(q, p), ¬R(p, z), C(z), Q(q)], [B(a)], [¬Q(a)], [S(f(a), a)], [T(a, f(a)], [¬T(a, p), ¬R(p, z), ¬C(z)], [D(f(a)], [A(f(a))], [C(f(f(a)))], [R(f(a)), f(f(a))]} →

Αντικατάσταση p -> f(a)

{[¬A(x), R(x, f(x))], [¬A(x), C(f(x))], [¬B(w), S(f(w), w)], [¬B(w), D(f(w)], [¬D(t), A(t)], [¬S(s, r), T(r, s)], [¬T(q, p), ¬R(p, z), C(z), Q(q)], [B(a)], [¬Q(a)], [S(f(a), a)], [T(a, f(a)], [¬T(a, p), ¬R(p, z), C(z)], [D(f(a)], [A(f(a))], [C(f(f(a)))], [R(f(a)), f(f(a))], [¬R(f(a), z), ¬C(z)]} →

Αντικατάσταση z -> f(f(a))

{[¬A(x), R(x, f(x))], [¬A(x), C(f(x))], [¬B(w), S(f(w), w)], [¬B(w), D(f(w)], [¬D(t), A(t)], [¬S(s, r), T(r, s)], [¬T(q, p), ¬R(p, z), C(z), Q(q)], [B(a)], [¬Q(a)], [S(f(a), a)], [T(a, f(a)], [¬T(a, p), ¬R(p, z), C(z)], [D(f(a)], [A(f(a))], [C(f(f(a)))], [R(f(a)), f(f(a))], [¬R(f(a), z), ¬C(z)], [¬C(f(f(a)))]} → [] Αντίφαση

Επομένως ισχύει K |= ∀x.(B(x) ⇒ Q(x))

Άσκηση 4

1)Όλες οι χώρες ανήκουν σε κάποια ήπειρο.

∀x. Χώρα(x) ⇒ ∃y. (Ήπειρος(y) ∧ ΑνήκειΣε(x, y))

2) Μερικές χώρες έχουν πληθυσμό πάνω από 300 εκατομμύρια.

∃x(Χώρα(x) ∧ ΜεγαλύτεροΑπο(πληθυσμός(x), 300.000.000))

3) Δεν υπάρχουν χώρες που να ανήκουν σε τρεις ηπείρους

¬∃x(Χώρα(x) ∧ ∃y_1∃y_2∃y_3(Ήπειρος(y_1) ∧ Ήπειρος(y_2) ∧ Ήπειρος(y_3) ∧ ¬(y_1=y_2) ∧ ¬(y_2=y_3) ∧ ¬(y_1=y_3) ∧ ΑνήκειΣε(x, y_1) ∧ ΑνήκειΣε(x, y_2) ∧ ΑνήκειΣε(x, y_3)))

4)Κάποια χώρα της Αμερικής είναι πολυπληθέστερη από όλες τις χώρες της Ευρώπης

∃x(Χώρα(x) ∧ ΑνήκειΣε(x, Αμερική) ∧ ∀y((Χώρα(y) ∧ ΑνήκειΣε(y, Ευρώπη)) ⇒ ΜεγαλύτεροΑπο(πληθυσμός(x), πληθυσμός(y)))

5) Υπάρχουν ακριβώς δύο χώρες με πληθυσμό πάνω από 1 δισεκατομμύριο

∃x_1∃x_2(Χώρα(x_1) ∧ Χώρα(x_2) ∧ ¬(x_1=x_2) ∧ ΜεγαλύτεροΑπο(πληθυσμός(x_1), 1.000.000.000) ∧ ΜεγαλύτεροΑπο(πληθυσμός(x_2), 1.000.000.000) ∧ ∀y_1∀y_2 (Χώρα(y_1) ∧ Χώρα(y_2) ∧ ¬(y_1=y_2) ∧ ΜεγαλύτεροΑπο(πληθυσμός(y_1), 1.000.000.000) ∧ ΜεγαλύτεροΑπο(πληθυσμός(y_2), 1.000.000.000 ⇒ (y_1 = x_1) ∧ (y_2=x_2))

6) Η Κίνα και η Ινδία είναι οι δύο πολυπληθέστερες χώρες

∀x((Χώρα(x) ∧ ¬ (x=Κίνα) ∧ ¬ (x=Ινδία)) ⇒ ΜεγαλύτεροΑπο(πληθυσμός(Κίνα), πληθυσμός(x)) ∧ ΜεγαλύτεροΑπο(πληθυσμός(Ινδία), πληθυσμός(x))

Άσκηση 5

1)

Έχουμε την πρώτη πρόταση της 1 παρακάτω:

$\forall x.(p(x) \Rightarrow q(a)) \rightarrow \forall x.(\neg p(x) \vee q(a))$

Έχουμε την δεύτερη πρόταση της 1 παρακάτω:

$(\forall x.p(x)) \Rightarrow q(a) \rightarrow \exists x(\neg p(x) \vee q(a))$

Αν q(a) είναι True τότε θα έχουμε και στις 2 προτάσεις αποτίμηση True.

Άρα εξετάζουμε την περίπτωση όπου q(a)=False.

Τότε η πρώτη πρόταση μας λέει ότι αν για κάθε x το p(x) είναι False τότε έχουμε αποτίμηση True. Αυτομάτως όμως στη δεύτερη πρόταση όποιο x και να πάρουμε το p(x) θα είναι πάλι False άρα θα ισχύει και η δεύτερη πρόταση.

Κοινώς δεν υπάρχει ερμηνεία που να ικανοποιεί την πρώτη και όχι τη δεύτερη πρόταση.

2)

Έχουμε την πρώτη πρόταση της 2 παρακάτω:

$\exists x.(p(x) \Rightarrow q(a)) \rightarrow \exists x.(\neg p(x) \vee q(a))$

Έχουμε τη δεύτερη πρόταση της 2 παρακάτω:

$(\exists x.p(x)) \Rightarrow q(a) \rightarrow \forall x.(\neg p(x) \vee q(a))$

Προφανώς εξετάζουμε πάλι την περίπτωση όπου q(a)=False.

Ουσιαστικά έχουμε την περίπτωση 1 ανάποδα. Σε αυτή την περίπτωση αν υπάρχει x με p(x) False τότε η πρώτη πρόταση αποτιμάται σε true. Όμως αυτό δε σημαίνει πως για κάθε x η p(x) θα είναι False στη δεύτερη πρόταση.
Οπότε έχουμε ερμηνεία την I={p->F, q->F}.

Άσκηση 6

1) r(x, b) ← r(a, x).
   r(x, z) ← r(x, y), r(y, z).

UP = {a, b}

BP = {r(a, a), r(a, b), r(b, a), r(b, b)}

2) q(0) ←.
   p(x) ← p(f(x))

UP = {0, f(0), f(f(0)), f(f(f(0))), …}

BP = {q(0), q(f(0)), q(f(f(0))), …, p(0), p(f(0)), p(f(f(0))), …}

Άσκηση 7

1) Αρχικά κάνουμε forwardchaining.

Η rule base μας περιλαμβάνει τους κάτωθι κανόνες:

1) parent(x, y) ← father(x, y).
2) parent (x, y) ← mother (x ,y).
3) sibling (y, z) ← parent(y, x),parent (z, x).
4) sibling (x, y) ← sibling(y, x).
5) grandparent (x, z) ← parent(x, y), parent(y, z).
6) cousin (y, z) ← grandparent(y, x), grandparent(z, x).
7) mother (A, B) ←.
8) father (A, C) ←.
9) mother (B, D) ←.
10) mother (E ,D) ←.
11) father (F, E) ←.
12) father (G, E) ←.

Θέλουμε να δείξουμε αν ισχύει ή οχι το κατηγόρημα : cousin(A,F) και sibling(A, G). Ο λύση μας ξεκινάει από κάτω προς τα πάνω, δηλαδή παρακάτω ξεκινάει από το Βήμα 1 για να καταλήξει στο Βήμα 6 και σε κάθε κελί εκτελείται από κάτω προς τα πάνω. Δηλαδή πρώτα παράγεται ο στόχος από το δέντρο του SLD, και μετά την επιτυχή ενοποίηση προχωράμε στον επόμενο στόχο. Τέλος θεωρούμε πως ο αλγόριθμος επιλέγει κάθε φορά τους κανόνες από πάνω προς τα κάτω και πάντα ξεκινάει από το αριστερότερο άτομο. Προφανώς υπάρχουν αρκετοί συνδυασμοί σε κάθε βήμα αλλά δεν ικανοποιούνται όλοι, οπότε παρουσιάζουμε μόνο εκείνους που ικανοποιούνται.

Βήμα 1	**Έχω: mother (A, B)**	**Έχω: father (A, C)**	**Έχω: mother (B, D)**	**Έχω: mother (E ,D)**	**Έχω: father (F, E)**	**Έχω: father (G, E)**
Βήμα 2	***Κανόνας: parent(x, y) ← mother (x, y)*** **Ενοποίηση: x/A, y/B** **Προκύπτει: parent (A, B)**	***Κανόνας: parent(x, y)*** **← father(x, y)** **Ενοποίηση: x/A, y/C** **Προκύπτει: parent (A, C)**	***Κανόνας: parent(x, y) ← mother (x, y)*** **Ενοποίηση: x/B, y/D** **Προκύπτει: parent (B, D)**	***Κανόνας: parent(x, y)*** **← mother (x, y)** **Ενοποίηση: x/E, y/D** **Προκύπτει: parent (E ,D)**	***Κανόνας: parent(x, y) ← father(x, y)*** **Ενοποίηση: x/F, y/C** **Προκύπτει: parent (F, E)**	***Κανόνας: parent(x, y) ← father(x, y)*** **Ενοποίηση: x/G, y/E** **Προκύπτει: parent (G, E)**
Βήμα 3	**Έχω: parent (B, D)parent (E, D)** ***Κανόνας: sibling (y, z) ← parent(y, x),parent (z, x)*** **Ενοποίηση: y/B, z/E** **Προκύπτει: Sibling(B, E)**			**Έχω: parent (F, E)parent (G, E)** ***Κανόνας: sibling (y, z) ← parent(y, x),parent (z, x)*** **Ενοποίηση: y/F, z/G** **Προκύπτει: Sibling(F, G)**		
Βήμα 4	**Έχω: Sibling(B, E)** ***Κανόνας: sibling (x, y) ← sibling(y, x)*** **Προκύπτει: Sibling(E, B)**			**Έχω: Sibling(F, G)** ***Κανόνας: sibling (x, y) ← sibling(y, x)*** **Προκύπτει: Sibling(G, F)**		
Βήμα 5	**Έχω: parent (A, B)parent (B, D)** ***Κανόνας: grandparent (x, z) ← parent(x, y), parent(y, z)*** **Ενοποίηση: x/A, z/D** **Προκύπτει: grandparent(A, D)**			**Έχω: parent (F, E)parent (E,D)** ***Κανόνας: grandparent (x, z) ← parent(x, y), parent(y, z)*** **Ενοποίηση: x/F, z/D** **Προκύπτει: grandparent(F, D)**		
Βήμα 6	**Έχω: grandparent(A, D)grandparent(F, D)** ***Κανόνας: cousin (y, z) ← grandparent(y, x), grandparent(z, x)*** **Ενοποίηση: y/A, z/F** **Προκύπτει: cousin (A, F)**					

Άρα καταλήγουμε πως ισχύει Sibling(B, E), Sibling(F, G), Sibling(E, B), Sibling(G, F), grandparent(A, D), grandparent(F, D) και cousin (A, F). <u>Άρα επιτυχία ως προς το και cousin (A, F) και αποτυχία ως προς το Sibling(A, G).</u> Τον παραπάνω πίνακα μπορούσαμε να τον επαναλάβουμε 2 φορές (μία για κάθε ερώτημα), ώστοσο θεωρούμε πως ο αλγόριθμος επιλέγει κάθε φορά τους κανόνες από πάνω προς τα κάτω, συνεπώς για να δούμε αν ικανοποιείται το cousin (A, F) πρέπει αναγκαστικά να διατρέξουμε και τις περιπτώσεις των κανόνων 3, 4.

2) Τέλος εκτελούμε backwardchaining.

Όπως και πριν θεωρούμε πως ο αλγόριθμος επιλέγει κάθε φορά τους κανόνες από πάνω προς τα κάτω και πάντα ξεκινάει από το αριστερότερο άτομο.

Θέλουμε να δείξουμε αρχικά αν ισχύει ή όχι το κατηγόρημα : cousin(A,F)

Βήμα 1	Έχω: cousin(A, F) *cousin (y, z) ← grandparent(y, x)grandparent(z, x)* Ενοποίηση: y/A, z/F Προκύπτει: grandparent(A, x)grandparent(F, x)
Βήμα 2	Έχω: grandparent(A, x)grandparent(F, x) Κανόνας: grandparent(x_1, z_1) ← *parent(x_1, y_1)parent(y_1, z_1)* Ενοποίηση: x_1/A Προκύπτει: parent(A, y_1)parent(y_1, z_1)grandparent(F, z_1)
Βήμα 3	Έχω: *parent(A, y_1)parent(y_1, z_1)*grandparent(F, z_1) *Κανόνας: parent(x_2, y_1) ← father(x_2, y_1)* Ενοποίηση: x_2/A Προκύπτει: father(A, y_1)parent(y_1, z_1)grandparent(F, z_1)
Βήμα 4	Έχω: father(A, y_1)parent(y_1, z_1)grandparent(F, z_1) *Κανόνας: father(A, C) ←.* Ενοποίηση: y_1/C Προκύπτει: father(A, C)parent(C, z_1)grandparent(F, z_1)
Βήμα 5	Έχω: father(A, C)parent(C, z_1)grandparent(F, z_1) *Κανόνας: parent(x_3, z_1) ← father(x_3, z_1)* Ενοποίηση: x_3/C Προκύπτει: father(A, C)father(C, z_1)grandparent(F, z_1)
Βήμα 6	Έχω: father(A, C)father(C, z_1) grandparent(F, z_1) *☒ (δεν υπάρχει ενοποίηση-δεν υπάρχει κανόνας father (C,_) ←)* Προκύπτει: : *parent(A, y_1)parent(y_1, z_1)*grandparent(F, z_1)
Βήμα 7	Έχω: *parent(A, y_1)parent(y_1, z_1)*grandparent(F, z_1) *Κανόνας: parent(x_4, y_1) ← mother(x_4, y_1)* Ενοποίηση: x_4/A Προκύπτει: *mother*(A, y_1)parent(y_1, z_1)grandparent(F, z_1)
Βήμα 8	Έχω: *mother*(A, y_1)parent(y_1, z_1)grandparent(F, z_1) *Κανόνας: mother(A, B) ←.* Ενοποίηση: y_1/B Προκύπτει: *mother*(A, B)parent(B, z_1)grandparent(F, z_1)
Βήμα 9	Έχω: *mother*(A, B)parent(B, z_1)grandparent(F, z_1) *Κανόνας: parent(x_5, z_1) ← father(x_5, z_1)* Ενοποίηση: x_5/B Προκύπτει: *mother*(A, B)father(B, z_1)grandparent(F, z_1)
Βήμα 10	Έχω: *mother*(A, B)father(B, z_1)grandparent(F, z_1) *☒ (δεν υπάρχει ενοποίηση-δεν υπάρχει κανόνας father (B,_) ←)* Προκύπτει: *mother* (A, B) parent(B, z_1) grandparent(F, z_1)
Βήμα 11	Έχω: *mother*(A, B)parent(B, z_1)grandparent(F, z_1) *Κανόνας: parent(x_6, z_1) ← mother(x_6, z_1)* Ενοποίηση: x_6/B Προκύπτει: *mother* (A, B)mother(B, z_1)grandparent(F, z_1)
Βήμα 12	Έχω: *mother*(A, B)mother(B, z_1)grandparent(F, z_1) *Κανόνας: mother (B, D) ←.* Ενοποίηση: z_1/D Προκύπτει: *mother*(A, B)mother(B, D)grandparent(F, D)
Βήμα 13	Έχω: *mother*(A, B)mother(B, D)grandparent(F, D) Κανόνας: grandparent(F, D)← *parent(x_7, y_7)parent(y_7, z_7)* Ενοποίηση: x_7/F, z_7/D Προκύπτει: *mother*(A, B)mother(B, D) *parent(F, y_7)parent(y_7, D)*
Βήμα 14	Έχω: *mother*(A, B)mother(B, D)*parent(F, y_7)parent(y_7, D)* *Κανόνας: parent(F, y_7) ← father(x_8, y_7)* Ενοποίηση: x_7/F Προκύπτει: *mother*(A, B)mother(B, D) *father (F, y_6)parent(y_6, D)*
Βήμα 15	Έχω: *mother*(A, B)mother(B, D) *father (F, y_7)parent(y_7, D)* *Κανόνας: father (F, E) ←.* Ενοποίηση: y_7/E Προκύπτει: *mother*(A, B)mother(B, D) *father (F, E)parent(E, D)*
Βήμα 16	Έχω: *mother*(A, B)mother(B, D)*father(F, E)parent(E, D)* *Κανόνας: parent(E, D) ← father(E, D)* *☒ (δεν υπάρχει ενοποίηση-δεν υπάρχει κανόνας father (E,D) ←)* Προκύπτει: *mother*(A, B)mother(B, D)*father (F, E)parent(E, D)*
Βήμα 17	Έχω: *mother*(A, B)mother(B, D)*father(F, E)parent(E, D)* *Κανόνας: parent(E, D) ← mother(E, D)* Προκύπτει: *mother*(A, B)mother(B, D)*father(F, E)mother(E, D)* Προκύπτει: []

Άρα επιτυχία το cousin(A, F)

Θέλουμε να δείξουμε αν ισχύει ή οχι το κατηγόρημα : sibling(A, G)

Άρα αποτυχία το sibling(A, G)

Βήμα 1	**Έχω: sibling(A, G)** **Κανόνας: *sibling (y, z) ← parent(y, x)parent (z, x)*** **Ενοποίηση: y/A, z/G** **Προκύπτει: : parent(A, x)parent(G, x)**
Βήμα 2	**Έχω: parent(A, x)parent(G, x)** ***Κανόνας: parent(x_1, y_1) ← father(x_1, y_1)*** **Ενοποίηση: x_1/A** **Προκύπτει: father(A, x)parent(G, x)**
Βήμα 3	**Έχω: father(A, x)parent(G, x)** ***Κανόνας: father(A, C) ←.*** **Ενοποίηση: x/C** **Προκύπτει: father(A, C)parent(G, C)**
Βήμα 4	**Έχω: father(A, C)parent(G, C)** ***Κανόνας: parent(G, C) ← father(G, C)*** ***☒ (δεν υπάρχει κανόνας father (G, C) ←)*** **Προκύπτει: : father(A, C) *father*(G, C)**
Βήμα 5	**Έχω: father(A, C)parent(G, C)** ***Κανόνας: parent(G, C) ← mother(G, C)*** ***☒ (δεν υπάρχει κανόνας mother(G, C) ←)*** **Προκύπτει: parent(A, x)parent(G, x)**
Βήμα 6	**Έχω: parent(A, x)parent(G, x)** ***Κανόνας: parent(x_1, y_1) ← mother(x_1, y_1)*** **Ενοποίηση: x_1/A** **Προκύπτει: *mother(A, x)*parent(G, x)**
Βήμα 7	**Έχω: *mother(A, x)*parent(G, x)** ***Κανόνας: mother (A, B) ←.*** **Ενοποίηση: x/B** **Προκύπτει: *mother*(A, B)parent(G, B)**
Βήμα 8	**Έχω: *mother*(A, B)parent(G, B)** ***Κανόνας: parent(G, B) ← father(G, B)*** ***☒ (δεν υπάρχει κανόνας father (G, C) ←)*** **Προκύπτει: : *mother*(A, B)parent(G, B)**
Βήμα 9	**Έχω: *mother*(A, B)parent(G, B)** ***Κανόνας: parent(G, B) ← mother(G, B)*** ***☒ (δεν υπάρχει κανόνας mother (G, B) ←)*** **Προκύπτει: *☒***

Άσκηση 8

Η rule base μας περιλαμβάνει τους κάτωθι κανόνες:

1) *add(x, 0, x) ← .*
2) *add(x, s(y), s(z)) ← add(x, y, z).*

Εφαρμόζουμε τον αλγόριθμο ανάλυσης backward SLD για το ερώτημα

add(s(0), u, s(s(0)))

Βήμα 1	**Έχω: add(s(0), u, s(s(0)))** **Κανόνας: *add(x, s(y), s(z)) ← add(x, y, z).*** **Αντικατάσταση: u/s(0)** **Ενοποίηση: x/s(0), z/s(0), y/s(0)** **Προκύπτει: add(s(0), 0, s(0))**
Βήμα 2	**Έχω: add(s(0), 0, s(0))** **Κανόνας: *add(x_1, 0, x_1) ← .*** **Ενοποίηση: x_1/s(0)** **Προκύπτει: []**

Άρα έχουμε επιτυχία με αντικατάσταση u/s(0).

Άσκηση 9

Έχουμα τα παρακάτω αξιώματα

1) $A \sqsubseteq \exists r.B$
2) $B \sqsubseteq \exists s.(A \sqcap C)$
3) $s \equiv r^-$
4) A(a)
5) ¬C(a)

Έχουμε: CN = {A, B, C}, RN = {r, s}, IN = {α}

Μοντέλο της γνώσης δε μπορεί να υπάρχει καθώς από τα αξιώματα 4, 5 έχουμε πως το άτομο a είναι μια έννοια A και όχι μια έννοια C. Επίσης από αξιώμα 2 το a δε μπορεί να ανήκει στην έννοια A και στην έννοια C.