Assignment #3

Parsa Youssefpour

student #: 1004749659

Question #1

a) $\underbrace{\exists R.\exists S.T}_{C} \sqsubseteq \underbrace{B \sqcup C}_{D}$

Using $t_x(C \sqsubseteq D) = \forall_x . t_x(C) \supset t_x(D) \longrightarrow \forall_x\, t_x(\exists R.\exists S.T) \supset t_x(B \sqcup C)$

$t_x(\exists R.\exists S.T)$

Using $t_x(\exists P.C) = \exists_y . P(x,y) \wedge t_y(C)$

$t_x(\exists R.\exists S.T) = \exists_y . R(x,y) \wedge t_y(\exists S.T)$

$t_y(\exists S.T) = \exists_z S(y,z) \wedge \underbrace{t_z(T)}$ ⟶ Always TRUE, remove it

$t_x(\exists R.\exists S.T) = \exists_y . R(x,y) \wedge \exists_z S(y,z) = \exists_y \exists_z (R(x,y) \wedge S(y,z))$

$t_x(B \sqcup C)$

Using $t_x(C \sqcup D) = t_x(C) \vee t_x(D)$

$t_x(B \sqcup C) = t_x(B) \vee t_x(C) = B(x) \vee C(x)$

$$\exists R.\exists S.T \sqsubseteq B \sqcup C = \forall_x ((\exists_y \exists_z (R(x,y) \wedge S(y,z)))) \supset (B(x) \vee C(x))$$

b) $\underbrace{A \sqcap \neg B}_{C} \sqsubseteq \underbrace{\forall R.C}_{D}$

Using $t_x(C \sqsubseteq D) = \forall_x . t_x(C) \supset t_x(D) \longrightarrow \forall_x . t_x(A \sqcap \neg B) \supset t_x(\forall R.C)$

$\forall_x . t_x(A \sqcap \neg B)$

Using $t_x(C \sqcup D) = t_x(C) \vee t_x(D)$

$t_x(A \sqcap \neg B) = t_x(A) \wedge t_x(\neg B) = A(x) \wedge \neg B(x)$

$t_x(\forall R.C)$

Using $t_x(\forall P.C) = \forall_y . P(x,y) \supset t_y(C)$

$t_x(\forall R.C) = \forall_y . R(x,y) \supset t_y(C)$

$= \forall_y . R(x,y) \supset C(y)$

$$A \sqcap \neg B \sqsubseteq \forall R.C = \forall_x (A(x) \wedge \neg B(x)) \supset \forall_y . R(x,y) \supset C(y)$$

c) $\underbrace{A \sqcup \exists R.B}_{C} \sqsubseteq \underbrace{\exists S.T}_{D}$

$\forall x \left(\left(A(x) \vee \exists y \left(R(x,y) \wedge B(y) \right) \right) \supset \exists z \left(S(x,z) \right) \right)$

Using $t_x(C \sqsubseteq D) = \forall x. t_x(C) \supset t_x(D) \longrightarrow \forall x. t_x(A \sqcup \exists R.B) \supset t_x(\exists S.T)$

$\underline{t_x(A \sqcup \exists R.B)}$

Using $t_x(C \sqcup D) = t_x(C) \vee t_x(D)$

$t_x(A \sqcup \exists R.B) = t_x(A) \vee t_x(\exists R.B)$

$t_x(\exists R.B) = \exists y. R(x,y) \wedge t_y(B) = \exists y. R(x,y) \wedge B(y)$

$t_x(A \sqcup \exists R.B) = A(x) \vee \exists y. R(x,y) \wedge B(y)$

$\underline{t_x(\exists S.T)}$

Using $t_x(\exists P.C) = \exists y. P(x,y) \wedge t_y(C)$

$t_x(\exists S.T) = \exists z. S(x,z) \wedge t_z(T)$

Always TRUE, Remove it

$A \sqcup \exists R.B \sqsubseteq \exists S.T = \forall x \left(A(x) \vee \exists y. R(x,y) \wedge B(y) \right) \supset \exists z. S(x,z)$

Question #2

a)

i. Cars are exactly those vehicles that have at least a wheel and are powered by an engine.

$Car \equiv Vehicle \sqcap (\geq 1\, hasPart.Wheel) \sqcap \exists poweredBy.Engine$

ii. Bicycles are exactly those vehicles that have at least a wheel and are powered by a human.

$Bicycle \equiv Vehicle \sqcap (\geq 1\, hasPart.Wheel) \sqcap \exists poweredBy.Human$

iii. Boats are exactly those vehicles that travel on water.

$Boat \equiv Vehichle \sqcap \exists travelsOn.Water$

iv. Boats have no wheels.

$Boat \sqsubseteq \neg hasPart.Wheel$

v. Cars and Bicycles do not travel on Water.

$Car \sqsubseteq \neg \exists travelsOn.Water$

$Bicycle \sqsubseteq \neg \exists travelOn.Water$

vi. Wheels are exactly those devices that have an axle and are capable of rotation.

$Wheel \equiv Device \sqcap \exists hasPart.Axel \sqcap \exists CapableOf.Rotation$

vii. Drivers are exactly those humans who control a vehicle.

$Driver \equiv Human \sqcap \exists controls.Vehicle$

viii. Drivers of cars are adults.

$Dirver \sqcap \exists contros.Car \sqsubseteq Adult$

ix. Humans are not vehicles.

$Humans \sqsubseteq \neg Vehicles$

x. Wheels and engines are not human.

$Wheel \sqsubseteq \neg Human$

$Engine \sqsubseteq \neg Human$

xi. Humans are either adults or children.

$Human \equiv Adult \sqcup Child$

xii. Adults are not children.

$Adult \sqsubseteq \neg Child$

b)

i. Bob is a human

$Human(bob)$

ii. Bob Controls QE2

$Controls(bob, qe2)$

iii. QE2 is a vehicle that travels on water

$Vehicle(eq2)$

$travelsOn.water(eq2)$

c)

i. $Boat \sqcap \exists hasPart.Wheel$ is satisfiable w.r.t K

From TBox $Boat \sqsubseteq \neg \exists hasPart.Wheel$

Therefore, if there is an $x \in Boat$ then $x \in \neg \exists hasPart.Wheel$

Based on the definition of satisfiability :

C is satisfiable if there exists a model $\mathcal{I}$ such that $C^{\mathcal{I}}$ is not $\bot$

As there is no $x \in Boat$ that $x \in \exists hasPart.Wheel$ $\longrightarrow$ $Boat \sqcap \exists hasPart.Wheel = \bot$ $\longrightarrow$ unsatisfiable

This statement is FALSE

ii. $Car \sqcap Bicycle$ is satisfiable w.r.t. K

Assuming $Car \sqcap Bicycle$ is True then there exists an $x \in Car$ and $x \in Bicycle$

From T-Box

$Car \equiv Vehicle \sqcap (\geq 1 hasPart.Wheel) \sqcap \exists poweredBy.Engine$ $\longrightarrow$ $x \in \exists poweredBy.Engine$

$Bicycle \equiv Vehicle \sqcap (\geq 1 hasPart.Wheel) \sqcap \exists poweredBy.Human$ $\longrightarrow$ $x \in \exists poweredBy.Human$

There is no axiom in the T-Box preventing x to be powered by both Engine and Human. Hence, $Car \sqcap Bicycle \neq \bot$ and therefore satisfiable.

This Statement is TRUE

iii. $\exists$ Controls. Car $\sqcap$ Child is satisfiable w.r.t. K

Assuming $\exists$ Controls. Car $\sqcap$ Child is Satisfiable then there is an $x \in \exists$ Controls. Car and $x \in$ Child

From T-Box Driver $\sqcap$ $\exists$ Contros. Car $\sqsubseteq$ Adult

Driver $\equiv$ Human $\sqcap$ $\exists$ controls. Vehicle

Human Can be an Adult or a child Based on Human $\equiv$ Adult $\sqcup$ Child

Based on defintion of a Car (Car $\equiv$ Vehicle $\sqcap$ (≥ 1 has Part. Wheel) $\sqcap$ $\exists$ poweredBy. Engine)

Car $\sqsubseteq$ vehicle $\longrightarrow$ And the driver Controls the car

Based on Driver $\sqcap$ $\exists$ Contros. Car $\sqsubseteq$ Adult a driver that Controls a car is subsumed by Adult

Since Adult and Child are disjoint (Adult $\sqsubseteq$ $\neg$Child). $\exists$ Contols. Car $\sqcap$ Child $= \bot$

This statement is FALSE

iv. bob is an instance of (Adult $\sqcap$ Driver) w.r.t. K We know that there exists an $x \in \exists$ Controls. Car and $x \in$ Adult

From T-Box

Driver $\equiv$ Human $\sqcap$ $\exists$ controls. Vehicle Human(bob) Controls(bob, qe2) Vehicle(eq2)

Based on the assertions above it can be concluded That Driver(bob)

From T-Box

Human $\equiv$ Adult $\sqcup$ Child $\longrightarrow$ Adult(bob) or Child(bob) $\longrightarrow$ Bob Can be either a child or Adult

Since eq2 is a Vehicle and travelsOn. water, based On Boat $\equiv$ Vehichle $\sqcap$ $\exists$ travelsOn. Water

it Can be concluded that eq2 is a Boat.

Since, there is no Axiom that prevents a child from Driving a Boat there can exist a model where bob is a child and drives a Boat.

Therefore bob $\in$ (Adult $\sqcap$ Driver) is non True in every model of K.

This statement is FALSE

V. qez is an instance of Boat w.r.t K.

From K

$travelsOn.water(eq2)$ → eq2 is an instance of travelsOn.Water

$Vehicle(eq2)$ → eq2 is an instance of Vehicle

From T-Box → $Boat \equiv Vehichle \sqcap \exists travelsOn.Water$

Hence, eq2 is an instance of Boat

This statement is TRUE

Vi. $\exists Contols.Car$ is subsumed by Adult w.r.t. K. $\exists Contols.Car \sqsubseteq Adult$

From T-Box

$Car \equiv Vehicle \sqcap (\geq 1 hasPart.Wheel) \sqcap \exists poweredBy.Engine$ → $Car \sqsubseteq Vehicle$

Based on $Driver \equiv Human \sqcap \exists controls.Vehicle$ → $Driver \sqsubseteq Human \equiv Adult \sqcup Child$

Based on $Dirver \sqcap \exists contros.Car \sqsubseteq Adult$ a Driver that Controls the car is subsumed by Adult

However, there is no axiom that states Car needs to be Contolled by a driver. Therefore, there could exist a model that the car is self driving and doesn't have a driver. In which case $\exists Controls.Car$ is not subsumed by Adult.

This statement is FALSE

Question #3

Fluent

$Contains(p, w, s)$ p = pot
w = liters of water
s = situation

empty(P): discard all water in the pot P

transfer(P, P'): Pour as much water as possible without spilling from pot p to p'. No change when p=p'

Capacity(P): represent maximum Capacity of a pot P

a)

$Poss(empty(P), s) \equiv True$

$Poss(transfer(p, p'), s) \equiv True$

b) $Poss(empty(p), s) \supset Contains(p, 0, do(empty(p), s))$

$Poss(transfer(p,p'), s) \supset [Contains(p, w, s) \wedge Contains(p', w', s)] \wedge$

$[Contains(p, w - \min(w, Capacity(p') - w'), do(transfer(p,p'), s)) \vee$

$Contains(p', w' + \min(w, Capacity(p') - w'), do(transfer(p,p'), s))]$

c) New language

function $\rightarrow$ amount(p): amount of water in P at the current state

Operators:

Capactiy(P): Max amount of water pot P can hold $Capacity(P_1) = 5$ $Capacity(P_2) = 2$

empty(p, amount(p))

Pre Conditions: Contains(p, amount(P)), Pot(p)

Delete list: Contains(p, amount(p))

Add list: Contains(p, 0)

transfer(p, amount(p), p', amount(p'))

Preconditions: Contains(p, amount(P)), Contians(p', amount(p')), Pot(p), Pot(p')

Delete list: Contains(p, amount(P))

Contains(p', amount(P'))

Add list: $Contains(p, amount(P) - \min(amount(P), Capacity(P') - amount(P')))$

$Contains(p', amount(P') + \min(amount(P), Capacity(P') - amount(P')))$

Note: amount(P) can be any value from 0 to Capacity(P)

d)

Initial State DataBase (DB)

$Contains(P_1, 5)$, $Contains(P_2, 0)$, $Pot(P_1)$, $Pot(P_2)$

Progressed databases

1. $empty(P_1, 5)$

$Contains(P_1, 5)$
$Pot(P_1)$ in DB

DB after exectuting:

$Pot(P_1)$, $Pot(P_2)$

$Contains(P_1, 0)$, $Contains(P_2, 0)$

3. $transfer(P_1, 5, P_1, 5)$

$Contains(P_1, 5)$, $Pot(P_1)$ in DB

DB after exectuting:

$Pot(P_1)$, $Pot(P_2)$

$Contains(P_1, 5)$, $Contains(P_2, 0)$

2. $empty(P_2)$

$Contains(P_2, 2)$
$Pot(P_2)$ in DB

DB after exectuting:

$Pot(P_1)$, $Pot(P_2)$

$Contains(P_1, 0)$, $Contains(P_2, 0)$

4. $transfer(P_2, 0, P_2, 0)$

$Contains(P_2, 0)$, $Pot(P_2)$ in DB

DB after exectuting:

$Pot(P_1)$, $Pot(P_2)$

$Contains(P_1, 5)$, $Contains(P_2, 0)$

5. transfer $(P_2, 0, P_1, 5)$

$Contains(P_2, 0), Contains(P_1, 5), Pot(P_1), Pot(P_2)$ in DB

DB after executing:

$Pot(P_1)$, $Pot(P_2)$

$Contains(P_2, 2 - \min(2, \cancel{5-5}^{\,0})) = \underline{Contains(P_2, 2)}$

$Contains(P_1, 5 + \min(2, \cancel{5-5}^{\,0})) = \underline{Contains(P_1, 5)}$

6. transfer $(P_1, 5, P_2, 0)$

$Contains(P_1, 5), Contains(P_2, 0), Pot(P_1), Pot(P_2)$ in DB

DB after executing:

$Pot(P_1)$, $Pot(P_2)$

$Contains(P_1, 5 - \min(\cancel{5}, 2-0)^{2}) = \underline{Contains(P_1, 3)}$

$Contains(P_2, 0 + \min(\cancel{5}, 2-0)^{2}) = \underline{Contains(P_2, 2)}$

For the rest, initial Conditions are not satisfied.

e) Goal: $Pot(P_2), Contains(P_2, 1)$

transfer $(P_2, 2, P_1, 4)$

Goal' : Goal + Preconditions − Add

Goal [$Pot(P_2)$, ~~$Contains(P_2, 1)$~~ +

Pre [$Contains(P_1, 4)$, $Contains(P_2, 2)$, $Pot(P_1)$, $Pot(P_2)$ −

Add [$Contains(P_1, 5)$, ~~$Contains(P_2, 1)$~~

regressed Goal (Goal'):

$Pot(P_1), Pot(P_2), Contains(P_1, 4), Contains(P_2, 2)$

This is physically not possible as there is more than 5L of water in this world. which is not possible.

transfer $(P_1, 1, P_2, 0)$

Goal' : Goal + Preconditions − Add

Goal [$Pot(P_2)$, ~~$Contains(P_2, 1)$~~ +

Pre [$Contains(P_1, 1)$, $Contains(P_2, 0)$, $Pot(P_1)$, $Pot(P_2)$ −

Add [$Contains(P_1, 0)$, ~~$Contains(P_2, 1)$~~

regressed Goal (Goal'):

$Pot(P_1), Pot(P_2), Contains(P_1, 1), Contains(P_2, 0)$

f) Initial state DB_0:

$Contains(P_1, 5)$, $Contains(P_2, 0)$, $Pot(P_1)$, $Pot(P_2)$

Plan:

1. transfer $(P_1, 5, P_2, 0)$

Preconditions : $Pot(P_1), Pot(P_2)$, $Contains(P_1, 5)$, $Contains(P_2, 0)$ in DB_0 therefore action is legal

DB_1: $Pot(P_1)$, $Pot(P_2)$

$Contains(P_1, 3)$, $Contains(P_2, 2)$

2. empty $(P_2, 2)$

Preconditions : $Pot(P_2)$, $Contains(P_2, 2)$ in DB_1 therefore action is valid.

DB_2: $Pot(P_1)$, $Pot(P_2)$

$Contains(P_1, 3)$, $Contains(P_2, 0)$

3. transfer(P_1, 3, P_2, 0)

Precondition: Pot(P_1), Pot(P_2), Contains (P_1, 3), Contains (P_2, 0) in DB_2 therefore action is legal

DB_3: Pot (P_1), Pot(P_2), Contains (P_1, 1), Contains (P_2, 2)

4. empty (P_2, 2)

Preconditions: Pot(P_2), Contains (P_2, 2) in DB_3 therefore action is legal.

DB_4: Pot (P_1), Pot (P_2) Contains (P_1, 1), Contains (P_2, 0)

5. transfer (P_1, 1, P_2, 0)

Precondition: Pot(P_1), Pot(P_2), Contains (P_1, 1), Contains (P_2, 0) in DB_4 therefore action is legal

DB_5: Pot (P_1), Pot(P_2) Contains (P_1, 0), Contains (P_2, 1)

Goal reached in DB_5: Pot (P_2), Contains (P_2, 1)