Logik18 Lösningsförslag till tenta
190417

1.

1.	$\neg p \to q$	Prem
2.	$p \to r$	Prem
3.	$\neg r \wedge t$	Prem
4.	$\neg r$	$\wedge e_1$ 3
5.	$\neg p$	MT 4, 2
6.	q	$\to$e 5, 1

2. Båda konsekvenserna är sanna. Bevisen kan se ut så här

1.	$\exists z(\neg P(z) \to Q(z))$	Prem
2.	$c \quad \neg P(c) \to Q(c)$	Ant
3.	$\exists y(\neg P(c) \to Q(y))$	$\exists y$ i 2
4.	$\exists x \exists y(\neg P(x) \to Q(y))$	$\exists x$ i 3
5.	$\exists x \exists y(\neg P(x) \to Q(y))$	$\exists z$e 1, 2

Se nästa blad.

1.	$\exists x \exists y (\neg P(x) \to Q(y))$	Prem
2.	$a \quad \exists y (\neg P(a) \to Q(y))$	Ant
3.	$b \quad \neg P(a) \to Q(b)$	Ant
4.	$Q(b) \vee \neg Q(b)$	LEM
5.	$Q(b)$	Ant
6.	$\neg P(b)$	Ant
7.	$Q(b)$	Copy 5
8.	$\neg P(b) \to Q(b)$	$\to_i$ 6-7
9.	$\exists z (\neg P(z) \to Q(z))$	$\exists z_i$ 8
10.	$\neg Q(b)$	Ant
11.	$\neg P(a)$	Ant
12.	$Q(b)$	Copy 5
13.	$\bot$	$\neg e$ 12, 10
14.	$Q(a)$	$\bot e$ 13
15.	$\neg P(a) \to Q(a)$	$\to_i$ 11-14
16.	$\exists z (\neg P(z) \to Q(z))$	$\exists z_i$ 15
17.	$\exists z (\neg P(z) \to Q(z))$	$\vee e$ 4, 5-9, 10-16
18.	$\exists z (\neg P(z) \to Q(z))$	$\exists y e$ 2, 3-17
19.	$\exists z (\neg P(z) \to Q(z))$	$\exists x e$ 1, 2-18

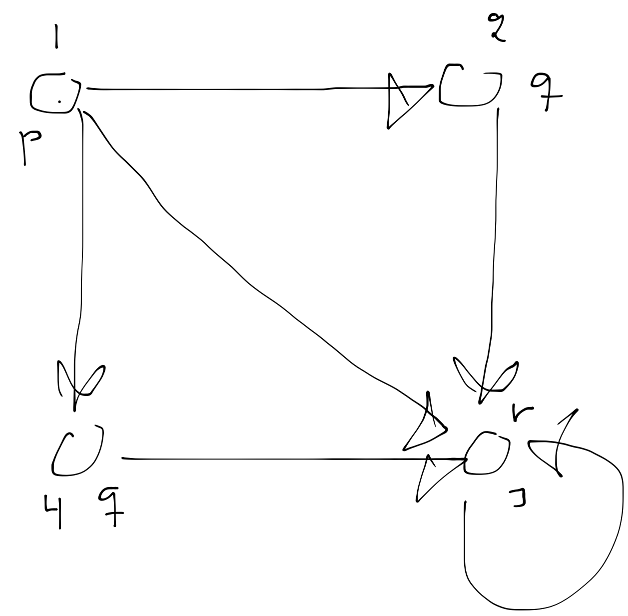
3. $p \vee q \quad \neg q \vee r \quad r \to q \wedge \neg r \quad q \vee r \to p$

p	q	r	$p \vee q$	$\neg q \vee r$	$r \to q \wedge \neg r$	$q \vee r \to p$
0	0	0	0	1	1	1
0	0	1	0	1	0	1
0	1	0	1	0	1	1
0	1	1	1	1	0	0
1	0	0	1	1	1	1
1	0	1	1	1	0	1
1	1	0	1	0	1	1
1	1	1	1	1	0	1

Vi ser att valuering p: T, q: F, r: F satisfierar alla formlerna

Ett sätt att göra beviset enklare är att se att om $p \to q \wedge \neg r$ ska vara sann måste då p vara falsk.
Då är frågan om $p \vee q$ och $\neg q$ kan satisfieras. Det kan de genom att sätta q falsk och p sann

4. Modellen ser ut så här



AG EF $p \vee q$

a. AG EF $p \vee q$

Vi kontrollerar först för vilka tillstånd EF $p \vee q$
EF $p \vee q$ betyder att det är möjligt att nå ett tillstånd där $p \vee q$ gäller. Vi ser att EF $p \vee q$ gäller i s_1, s_2, s_4
AG EF $p \vee q$ betyder att i alla tillstånd som kan nås gäller EF $p \vee q$. Men från alla tillstånd kan man nå s_3 där EF $p \vee q$ <u>inte</u> gäller. Så AG EF $p \vee q$ gäller inte i något tillstånd.

b. EG AF $q \vee r$

AF $q \vee r$ betyder att $q \vee r$ nödvändigtvis är sant förr eller senare. Det är sant i alla tillstånd
EG AF $q \vee r$ betyder att det finns en väg där AF $q \vee r$ är sann överallt. Eftersom AF $q \vee r$ är sann överallt är EG AF $q \vee r$ sann för alla tillstånd.

5. Regeln lyder
$$\frac{t_1 = t_2 \quad \varphi[t_1/x]}{\varphi[t_2/x]}$$

I vårt fall är $t_1 = y$, $t_2 = z$

Så $\varphi[y/x] = [y = g(y, z)]$
och $\varphi[z/x] = [y = g(z, z)]$
Så φ måste vara en formel som innehåller x (och kanske y och z) och som är sådan att om man byter ut x mot y får vi $y = g(y, z)$ och om vi byter ut x mot z får vi $y = g(z, z)$.
Men då måste vi ha $\varphi : y = g(x, z)$

6.

1.	$\exists x \exists y\, F(x,y)$	Prem
2.	$\forall x\, \neg F(x,x)$	Prem
3.	$a \quad \exists y\, F(a,y)$	Ant
4.	$b \quad F(a,b)$	Ant
5.	$a = b$	Ant
6.	$F(b,b)$	$=$e 4,5
7.	$\neg F(b,b)$	$\forall x$e 2
8.	$\perp$	$\neg$e 6,7
9.	$a \neq b$	$\neg$i 5-8
10.	c	Ant
11.	$c = a \vee c \neq a$	LEM
12.	$c = a$	Ant
13.	$c = c$	$=$i
14.	$a = c$	$=$e 13, 12
15.	$c \neq b$	$=$e 9, 14
16.	$\exists y\,(c \neq y)$	$\exists y$i 15
17.	$c \neq a$	Ant
18.	$\exists y\,(c \neq y)$	$\exists y$i 17
19.	$\exists y\,(c \neq y)$	$\vee$e 11, 12-16, 17-18
20.	$\forall x \exists y\,(x \neq y)$	$\forall x$i 10-19
21.	$\forall x \exists y\,(x \neq y)$	$\exists y$e 3, 4-20
22.	$\forall x \exists y\,(x \neq y)$	$\exists x$e 1, 3-21

7. Version DD1351

a)
app(L1,L2,L) där L är instantierat till en lista och L1 och L2 är obundna variabler.

?- app(L1,L2,[1,2,3]), write((L1,L2)), nl, fail.
skriver ut alla svar och terminerar sedan.

b)
De två predikaten app3a/4 och app3b/4 har samma deklarativa (avsedda/ modellteoretiska, eng: "intended") semantik, dvs mängden av lösningar som ett idealt prologsystem kan
Generera är identiska.
Skillnaden mellan app3a och app3b visar sig i den operationella (bevisteoretiska) semantiken.
Anropade på liknande sätt som i uppgift a) dvs app3a(L1,L2,L3,L) eller app3b(L1,L2,L3,L) där L är instantierat till en lista och L1, L2 och L3 är obundna variabler,
är att de presenterar svaren i olika ordning, samt att den ena, app3a, terminerar, medan app3b, hamnar i en loop efteråt pga att backtracking får det "vänstraste" anropet till app(L2,L3,R)
att försöka generera längre och längre dellistor L2 och L3 och lägga ihop dem till R, vilket sedan får app(L1,R,L) att misslyckas, vilket i sin tur triggar backtracking etc.
I app3b anropas app(L1,R,L) först vilket delar upp L i två listor L1 och R, anropet till app(L2,L3,R) som anropas därefter har då ett definierat värde (en lista) för R och L2 och L3 kan då genereras
på ett finit antal sätt.

c) Betydelsen av att skriva H=[_|_]. är att listan H måste ha minst ett element.

d) I de två rekursiva lösningarna appn/2 och appm2 är situationen likartad som i b) ovan. Anropade som angivet ovan presenteras i båda fallen alla lösningar,
dvs alla olika sätt att dela upp en lista i listor med vardera minst ett element, möjligen ordnade lite olika.
Anropet appm(Lm,L) med L instantierat till en lista och Lm obunden loopar efter att alla lösningar skrivits ut.

I både b) och d) är den deklarativa (modellteoretiska) semantiken lika men den operationella semantiken skiljer sig åt.

7. Version DD1350

a. Riff ([ ], L, L)

Riff (L, [ ], L)

Riff ([a | L₁], [b | L₂], [a | [b | Riff (L₁, L₂)]])

b. Vi behöver ha en rekursiv definition av length.
Den vanliga definitionen är

length ([ ]) = 0

length ([a | L]) = 1 + length (L)

Vi använder nu induktion.

Basfallen är $L_1 = [\,]$, $L_2 = L$, $L_3 = L$

Här gäller length (L_3) = length (L)

och length (L_1) + length (L_2) = length ([ ]) + length (L)

= 0 + length (L) = length (L)

Så length (L_3) = length (L_1) + length (L_2)

Fallet $L_1 = L$, $L_2 = [\,]$, $L_3 = L$ hanteras på samma sätt.

Antag nu att $L_1 = [a \mid S_1]$, $L_2 = [b \mid S_2]$

och $L_3 = [a \mid [b \mid Riff(S_1, S_2)]]$

Då har vi length (L_3) = length ([a | [b | Riff (S_1, S_2)]]) =

1 + length ([b | Riff (S_1, S_2)]) = 2 + length (Riff (S_1, S_2)) =

= {Induktionsantagande} = 2 + length (S_1) + length (S_2) =

1 + length (S_1) + 1 + length (S_1) = length (L_1) + length (L_2).

8. a) T går alltid att satisfiera. Vi kan sätta $p_i : F$ för alla i. Varje formel av typen $p_i \to p_j$ är då sann. (Det fungerar att sätta $p_i : T$ också.)

b) Denna mängd går alltid att satisfiera. Sätt alla $p_i : F$. Som i a gäller att alla formler i T är satisfierade. Dessutom är formeln $p_1 \to \neg p_i$ sann eftersom p_1 är falsk.

c) Denna mängd går inte alltid att satisfiera. Antag att T bl.a. innehåller formlerna $p_1 \to p_2$, $p_2 \to p_1$, $p_2 \to p_3$, $p_3 \to p_2$. Då är p_1, p_2, p_3 ekvivalenta dvs har samma sanningsvärde. Antag nu att de extra formlerna är $p_1 \to \neg p_2$ och $\neg p_1 \to p_3$. Då ger $p_1 \to p_2$ och p_2 tillsammans att p_1 måste vara falsk. Då ger $\neg p_1 \to p_3$ att p_3 är sann.
Men det är omöjligt eftersom p_1 och p_3 är ekvivalenta.

9. a. Konsekvensen är inte sann. Vi kan hitta ett enkelt motexempel. Sätt $\mathcal{U} = \{1, 2\}$ och $A(1): T$, $A(2): F$, $B(1): F$, $B(2): T$. Då gäller $\exists x \exists y (A(x) \wedge B(y))$ (med $x = 1$ och $y = 2$). Eftersom det bara finns ändligt många element i $\mathcal{U}$ gäller också $\exists^F_x \exists^F_y (A(x) \wedge B(y))$. Men $\exists z (A(z) \wedge B(z))$ gäller inte och därför inte heller $\exists^F_z (A(z) \wedge B(z))$

b. Konsekvensen är inte sann. Som motexempel kan vi ta $\mathcal{U} = \{0, 1, 2, 3, \ldots\}$ och sätta

$A(x)$ är sann för alla x.

$B(y)$ är sann $\Longleftrightarrow x = 0$.

Då gäller $\exists^F_z (A(z) \wedge B(z))$ med $z = 0$

För godtyckligt x gäller $\exists^F_y (A(x) \wedge B(y))$ med $y = 0$. Men då gäller $\exists^F_y (A(x) \wedge B(y))$ för oändligt många värden på x, så $\exists^F_x \exists^F_y (A(x) \wedge B(y))$ är falsk.

c. Också här kan man konstruera ett motex. Sätt $\mathcal{U} = \{a\} \cup \mathbb{Z}$ där $\mathbb{Z}$ är mängden av positiva och negativa heltal. Låt $G(x, y): T$ om $x = a$ och $y = a$ eller något heltal.

Låt $G(x, y): T$ om $x, y \in \mathbb{Z}$ och $x \leq y$. I alla andra fall är $G(x, y): F$. Då gäller $\exists^F_x \forall_y G(x, y)$ med $x = a$. Men $\forall_y \exists^F_x G(x, y)$ är falsk eftersom det för varje $y \in \mathbb{Z}$ finns oändligt många x så att $G(x, y)$ gäller.

10. Vi använder de vanliga konstruktionsreglerna. Enda problemet är att vi inte kan säga utan vidare att $\neg A$ är en formel om A är en formel eftersom vi kan få dubbla negationer då. Vi måste förutsätta att A inte inleds med en negation.

a. Om p är en variabel gäller att $var(p) = 1$ och $kon(p) = 0$. Då gäller $kon(p) \geq var(p) - 1$.

Antag att $\varphi = \neg A$. Då gäller $kon(\neg A) = kon(A) + 1$
$\geq var(A) - 1 + 1 = var(\neg A) \geq var(\neg A) - 1$.

Antag att $\varphi = A K B$ där $K = \vee, \wedge, \rightarrow$

Då gäller $kon(A K B) = kon(A) + kon(B) + 1 \geq$
$\geq var(A) - 1 + var(B) - 1 + 1 = var(A K B) - 1$

Så induktionen visar att $kon(\varphi) \geq var(\varphi) - 1$ för alla φ.

[Obs: $var(A K B) = var(A) + var(B)$ gäller eftersom en variabel kan förekomma högst en gång.]

b. Om $\varphi = p$ gäller $kon(p) = 0 \leq 1 = 4var(p) - 3$

Om vi har en formel $\varphi = \neg A$ får vi ett problem med induktionen.

Det enklaste sättet att hantera svårigheten är förmodligen att använda en skarpare olikhet: Om φ *inte* inleds med en negation så gäller $kon(\varphi) \leq 4kon(\varphi) - 4$.

Om $\varphi = p$ gäller $kon(p) = 0 \leq var(p) - 4$

Vi visar sedan båda olikheterna *samtidigt* med induktion

Antag att $\varphi = A K B$ där $K = \vee, \wedge, \rightarrow$.

Då inleds inte φ med negation.

$$kon(AKB) = kon(A) + kon(B) + 1 \leq 4kon(A) - 3 + 4kon(B) - 3 + 1$$

$$= 4var(AKB) - 5 \leq 4kon(AKB) - 4$$

Så båda olikheterna gäller.

Antag nu att $\varphi = \neg A$ där A inte inleds med negation

Då gäller $kon(\neg A) = kon(A) + 1 \leq 4kon(A) - 4 + 1 = 4kon(A) - 3$

Så för alla formler gäller $kon(\varphi) \leq 4kon(\varphi) - 3$

11. $(\| u = u_0 \wedge u \geq 0 \|)$

* $(\| \exists n\, (u_0 - u = 3n \|)$

while $(u >= 3)$ {

$(\| u \geq 3 \wedge \exists n\, (u_0 - u = 3n \|)$

* $(\| \exists n\, (u_0 - (u - 3) = 3n \|)$

$u = u - 3;$

$(\| \exists n\, (u_0 - u = 3n \|)$

}

$(\| \neg (u \geq 3) \wedge \exists n\, (u_0 - u = 3n) \|)$

if $(u = 0)$ {

$(\| u = 0 \wedge \neg (u \geq 3) \wedge \exists n\, (u_0 - u = 3n \|)$

* $(\| 1 = 1 \wedge \exists n\, (u_0 = 3n) \|)$

$R = 1;$

$(\| R = 1 \wedge \exists n\, (u_0 = 3n) \|)$

* $(\| [R = 1 \wedge \exists n\, (u_0 = 3n)] \vee [R = 0 \wedge \neg \exists n\, (u_0 = 3n)] \|)$

} else {

$(\| \neg (u = 0) \wedge \neg (u \geq 3) \wedge \exists n\, (u_0 - u = 3n) \|)$

* $(\| 0 = 0 \wedge \neg \exists n\, (u_0 = 3n) \|)$

$R = 0;$

$(\| R = 0 \wedge \neg \exists n\, (u_0 = 3n) \|)$

* $(\| [R = 1 \wedge \exists n\, (u_0 = 3n)] \vee [R = 0 \wedge \neg \exists n\, (u_0 = 3n)] \|)$

* anger bevisförpliktelser.