CS3050 Exam

200007413

1

a. Syntactic entailment is whether or not a formula is syntactically equivalent to another given the syntax of the logic system. This determines the validity of a formula.

Semantic entailment is whether or not a formula actually is true in a domain. Therefore, a formula can syntactically entail, but the facts of the world mean it is false and so not syntactically entail.

For a proof system to be complete, it must be able to formulate any preposition syntactically valid. However, to be sound, any formula that syntactically entails must also semantically entail and actually be true.

b. (i)

1.	$\neg(P \wedge Q)$	Premise
2.	P	Assumption
3.	Q	Assumption
4.	$P \wedge Q$	$\wedge I$ on 2,3
5.	**F**	$\neg E$ on 1,5
6.	$\neg P$	$\neg I$ on 2,5
7.	$\neg P \vee \neg Q$	$\vee I$ on 6

1.	$\neg P \vee \neg Q$	Premise
2.	$\neg P$	Assumption
3.	$P \wedge Q$	Assumption
4.	P	$\wedge E$ on 3
5.	**F**	$\neg E$ on 2,4
6.	$\neg(P \wedge Q)$	$\neg I$ on 2,5
7.	$\neg Q$	Assumption
8.	$P \wedge Q$	Assumption
9.	Q	$\wedge E$ on 8
10.	**F**	$\neg E$ on 7,9
11.	$\neg(P \wedge Q)$	$\neg I$ on 10
12.	$\neg(P \wedge Q)$	$\vee E$ on 1,2-6,7-11

	1.	$\exists x.(P(x) \vee Q(x))$	Premise
	2.	$P(a) \vee Q(a)$	Assumption
	—		
	3.	$P(a)$	Assumption
	4.	$\exists x.P(x)$	$\exists I$ on 3 $\theta[a/x]$
	5.	$(\exists x.P(x)) \vee (\exists x.Q(x))$	$\vee I$ on 4
(ii)	—		
	6.	$Q(a)$	Assumption
	7.	$\exists x.Q(x)$	$\exists I$ on 6 $\theta[a/x]$
	8.	$(\exists x.P(x)) \vee (\exists x.Q(x))$	$\vee I$ on 7
	—		
	9.	$(\exists x.P(x)) \vee (\exists x.Q(x))$	$\vee E$ on 2,3-5,6-8
	10.	$(\exists x.P(x)) \vee (\exists x.Q(x))$	$\exists E$ on 1,2,9

2

1. To prove: $\forall a \forall b \forall c.(gcd(a, bc) | gcd(a, b) \cdot gcd(a, c))$

1.	$gcd(a, b) = x_1 a + y_1 b$	by definition of gcd
2.	$gcd(a, c) = x_2 a + y_2 c$	by definition of gcd
3.	$gcd(a, bc) = x_3 a + y_3 bc$	by definition of gcd
4.	$gcd(a, b) \cdot gcd(a, c) = (x_1 a + y_1 b)(x_2 a + y_2 c)$	from arithmetic on 1,2
5.	$= x_1 x_2 a^2 + x_1 y_2 ac + x_2 y_1 ab + y_1 y_2 bc$	from arithmetic on 1,2
6.	$gcd(a, bc) \mid a$	from 3 by definition of gcd
7.	$gcd(a, bc) \mid bc$	from 3 by definition of gcd
8.	$gcd(a, bc) \mid a^2$	from 6 by arithmetic
9.	$gcd(a, bc) \mid ac$	from 6 by arithmetic
10.	$gcd(a, bc) \mid ab$	from 6 by arithmetic
11.	$gcd(a, bc) \mid x_1 x_2 a^2 + x_1 y_2 ac + x_2 y_1 ab + y_1 y_2 bc$	from 7,8,9,10 by arithmetic
12.	$gcd(a, bc) \mid (x_1 a + y_1 b)(x_2 a + y_2 c)$	from 11,5
13.	$gcd(a, bc) \mid gcd(a, b) \cdot gcd(a, c)$	from 12,4

2. TODO

3. (i)

$$\forall x.(x - x = 0)$$

$$\forall x \forall y.(x - y = s(x - s(y))$$

(ii)

$$\forall b.(b^0 = 1)$$

$$\forall b.(b^1 = b)$$

$$\forall b \forall e.(b^{s(e)} = b^e \cdot b)$$

4. When running printAll(a), the variable Start is associated with a, then test(a, Goal, Result) is ran which then gets all results matching route(a, Goal). Because of our road facts, the first route definition satisfies Goal as b and d as there are direct roads from a. The second route definition also allows Goal to associate to c because there is a road(a,b) and road(b, c).

With these Goals, test then resolves the Result for each of the options from route(Start, Goal). For Goal as b, result associates check(Goal) as check(b) which then associates Result with free because no busy(b) is set and so check(Goal) fails.

For Goal as d, Result associates to free because check(d) fails causing the first result to fail so Result is associated to free.

Then for Goal as c, result associates check(Goal) as c which associates Result with check(c) busy as busy(c) is defined causing check(c) to succeed.

Then, printall gets each Goal and Result combination, and writes the Goal with the associated Result which is:

b:free

d:free

c:busy

3

1. First order logic is undecidable. This means that it is not always possible to determine whether or not a formula in FOL is provable.

   This is best demonstrated using Turing Machines which solve formula. Unprovable formula will give a turing machine that never halts, and provable formulas will halt. We can call this machine T.

   To be decidable, it must be that we can determine whether or not each machine T will halt. This can be expressed in first order logic as and so a halting function H can be created. This halts with a 1 if T will halt or halt with a 0 if T will not halt.

   We can then feed H into a similar H_2 which halts when H returns 0 and loops when H returns 1. When T loops, H halts with a 0 which then means H_2 halts with a 1. When T halts, then H halts with a 1, but then H_2 loops which means we can't determine whether or not H will stop or not and so H cannot exist.

2.

A1	$\forall x.(\exists y.B(x,y))$	
	$= \forall x.(B(x,s(x)))$	skolemisation of y
	$= B(X,s(X))$	skolemisation of x
A2	$\forall x.((B(x,dairy) \vee V(S(x)) \rightarrow \neg B(x,vegetables))$	
	$= (B(X,dairy) \vee V(S(X)) \rightarrow \neg B(X,vegetables)$	skolemization of x
	$= \neg(B(X,dairy) \vee V(S(X))) \vee \neg B(X,vegetables)$	
A3	$\forall x((\exists y.(C(x,y)) \rightarrow E(x))$	
	$= \forall x(((C(x,s_1(x))) \rightarrow E(x))$	skolemization of y
	$= (C(X,s_1(X))) \rightarrow E(X)$	skolemization of x
	$= \neg(C(X,s_1(X))) \vee E(X)$	$\rightarrow$ expanison
A4	$\exists x.C(x,oven)$	
	$= C(s_3(X),oven)$	skolemization of x
A5	$(\neg B(onions,dairy)) \wedge C(onions,oven)$	
C6	$\neg(B(onions,vegetables) \rightarrow (\neg S(onions) \wedge E(onions)))$	
	$= \neg\neg(B(onions,vegetables) \vee (\neg S(onions) \wedge E(onions)))$	
	$= B(onions,vegetables) \vee (\neg S(onions) \wedge E(onions)))$	
	$= (B(onions,vegetables) \vee \neg S(onions)) \wedge (B(onions,vegetables) \vee E(onions))$	

1	$\{B(X, s(X))\}$
2	$\{\neg(B(X, dairy), V(S(X)), \neg B(X, vegetables)\}$
3	$\{\neg(C(X, s_1(X))), E(X)\}$
4	$\{C(s_3(X), oven)\}$
5.1	$\{\neg B(onions, dairy)\}$
5.2	$\{C(onions, oven)\}$
6.1	$\{B(onions, vegetables) \vee \neg S(onions)\}$
6.2	$\{B(onions, vegetables) \vee E(onions)\}$
7	1,

3. base case: $num_1(concat(\epsilon, reverse(\epsilon))) = num_1(concat(\epsilon, \epsilon) = num_1(\epsilon) = 0$