Problem Set on Syntactic Operations, Semantic Operations, and Homomorphisms: Answers and Notes

1. Notes on the Answers

In Section 2, I have copied some illustrative answers from the problem sets submitted to me. In this section, I provide some notes on the answers below as well as on the problems themselves.

(1) **Exercise on the Syntactic Operations for PL** (no comments)

(2) **Exercise on the Syntactic Operations for FOL** (no comments)

(3) **Exercise on (Montagovian) Syntactic Rules**

- (no comments on (3a)-(3d))

- The challenging part of this problem is, of course, (3e). First, everyone correctly observed that there is no way in our 'triplet notation' for syntactic rules to have a *single* rule that states what (3e) states.
  - o Rather, we're going to need two separate rules, one creating VPs consisting of a V and an NP (easy), and one creating VPs that contain only a single V (the challenge).

- The true challenge here is finding a way of stating in our triplet notation that VPs can contain only a verb. **The issue is that the syntactic operation Merge was defined to be binary, and so a rule like <Merge, <V>, VP> would be ill-formed.**
  - o After all Merge(φ) (where φ is a V) is simply undefined.

- There were two solutions offered, one of which is slightly better than the other.
  - o The optimum one is to re-define 'Merge', splitting it into two different operations: (i) Binary Merge (Merge_2), which is as defined in the homework, and (ii) Unary Merge (Merge_1), which is the identity function on strings.
    - With these two Merge operations, we can now have the rules:
      (i) < Merge_2 , <V, NP> , VP>
      (ii) < Merge_1 , <V> , NP>

  - o The less optimum solution is to write a rule of the following sort, where 'ε' refers to the 'empty string': < Merge, < V, ε >, VP>
    - Informally read, this rule would state 'The result of applying (binary) Merge to an expression of category V and the empty string is a VP'.
    - This solution is less optimal, however, because 'ε' is not the label for a syntactic category, but rather the name of a specific string.
      - However, since I never said last week that a syntactic rule can only make reference to *syntactic categories*, this is still an acceptable solution.

Important Note:

- To actually capture the PS rule 'VP → V' in a Montagovian analysis of English, we *couldn't* use something like '<Merge, < V, ε >, VP>' (again, because 'ε' is not a syntactic category label)
- Instead, we'd have to suppose that the syntactic operation forming non-branching VPs is a *distinct* syntactic operation from the one forming branching VPs.

(4) **Our New Definition of a Model**

- In the answer to (4a), some folks tries to write out I(Q) in set notation, rather than matrix notation.
  - **Unfortunately, the folks who tried this generally wrote I(Q) as a set of triples, rather than a set of pairs.**
  - Note that if we were to represent (the curried characteristic function of) of I(Q) by listing its members in set notation, it would look as follows:

$$\left\{\begin{array}{l} \langle a, \{\langle a,0\rangle, \langle b,1\rangle, \langle c,1\rangle\}\rangle, \\ \langle b, \{\langle a,0\rangle, \langle b,0\rangle, \langle c,1\rangle\}\rangle, \\ \langle c, \{\langle a,0\rangle, \langle b,0\rangle, \langle c,0\rangle\}\rangle \end{array}\right\}$$

- In the answer to (4b), the calculations should always begin with an explicit statement to the effect of 'let *g* be an arbitrary variable assignment based on $\mathcal{M}$. Otherwise, you're leaving *g* undefined (or only implicitly defined).

(5) **Exercise on 'Interpretations' for PL** (no comments)

(6) **Exercise on Homomorphisms and Compositions**

- Everyone basically hit on the key components of the proof. I have presented below an especially clear formulation of the argument.
- One issue to be mindful of, however, is that several folks wrote things like the following: "for all $i \in \mathbb{N}$, f_i has the same arity as g_i"
  - **The problem is that, strictly speaking, this implies that there are a (countably) infinite number of operations f_i and g_i**
  - What would work better here is simply the following: "for all *i*, f_i has the same arity as g_i", since it's clear from context that for all *i*, $i \leq n$.

2. Illustrative Answers from Submitted Problem Sets

(1) **Exercise on the Syntactic Operations for PL**
Please show how the following formulae of PL can be constructed via applications of the syntactic operations Not, And, Or, and If.

a. Not(And(p, If(p, Or(q, r))))

b. PL:
$(((p \vee r) \to q) \vee s)$

Syntactic operation for PL:
Or (If (Or (p, r), q), s)

c. $((p \to q) \to (\neg q \to \neg p)) \wedge ((p \to q) \to (\neg p \vee q))$

And(If(If(p,q), If(Not(q), Not(p))), If(If(p,q), Or(Not(p),q)))

(2) **Exercise on the Syntactic Operations for FOL**
Please show how the following formulae of FOL can be constructed via applications of the syntactic operations Concat, Not, And, Or, If, Ext, and All.

a. $\forall x((Px) \to \exists y((Lx)y))$

All(x,If(Concat(P,x), Ext(y, Concat(Concat(L,x),y))))

b. $\exists z(((Ra)z) \vee \sim((Ra)z))$:

Ext(z, Or(Concat(Concat(R, a), z), Not(Concat(Concat(R, a), z))))

c. $\underline{((Ab) \vee \sim \exists x(Ax))}$:

Or(Concat(A, b), Not(Ext(x, Concat(A, x))))

(3) **Exercise on (Montagovian) Syntactic Rules**
Let Merge be an operation that takes two strings φ and ψ and forms the string 'φ ψ'. Using this operation Merge, please show how each of the PS rules below can be abstractly characterized as triple <Op, <Cat_1, … Cat_n>, Cat>, where Op is an *n*-ary syntactic operation, and each of Cat_i and Cat is some syntactic category label.

a. ⟨Merge, ⟨D, NP⟩, DP⟩
b. ⟨Merge, ⟨P, NP⟩, PP⟩
c. ⟨Merge, ⟨A, NP⟩, NP⟩
d. ⟨Merge, ⟨NP, VP⟩, S⟩
e. Because 'VP → V (NP)' is nothing more than a convenient shorthand for two distinct PS rules ('V → V' and 'V → V NP'), it is equivalent to the two rules below, where ε is the null string:
⟨Merge, ⟨V, NP⟩, VP⟩, ⟨Merge, ⟨V, ε⟩, VP⟩

<u>Note:</u>
Under the official definitions put forward this week, '< Merge, < V, ε >, VP>' would not be an acceptable rule, since ε is not a syntactic category label.

<u>*Another Approach to (3e):*</u>

- I modeled optionality in (3e) by giving two separate Merge rules for VPs: one for intransitive verbs and one for transitive verbs.
- Having both an intransitive and a transitive VP Merge rule would require Merge to have two different syntaxes.
    - The intransitive rule would take a single string V, and in essence, relabel it as a VP. This would parallel our rule for 'Not': <Not, <WFF>, WFF>
    - The transitive rule would make a VP out of two strings, namely V and NP. This parallels the syntax we have for 'And,' 'Or,' and 'If.'
- If we use two separate Merge rules, we could define them in the following way:

$\text{Merge}_{\text{unary}}(\phi)$ = The result of mapping ϕ to ϕ = ϕ
$\text{Merge}_{\text{binary}}(\phi,\Psi)$ = The result of concatenating ϕ and Ψ = ϕ Ψ

Requiring two types of Merge operations is not completely ideal. On the one hand, it seems to produce the right category output (i.e. VP) out of the proper component parts (V and NP). But there's a certain sense in which it is inelegant because it requires two different syntaxes (one that takes one string out of two strings and another that produces one string out of one string). Perhaps this is simply a labeling issue though, and it is actually perfectly acceptable for our various Merge rules.

<u>The Implied Rules:</u> (i) <$\text{Merge}_{\text{Unary}}$, <V>, VP>
(ii) <$\text{Merge}_{\text{Binary}}$, <V, NP>, VP>

(4) **Our New Definition of a Model**

Let $\mathcal{M}$ be a model <D,I> as defined in the handout "First Order Logic: Formal Semantics and Models", where D = { a, b, c }, and I consists of at least the mappings below:

I(P) = { x : x ∈ D and x is a vowel }

I(Q) = { <x,y> : x,y ∈ D and x precedes y in the alphabet }

a. Please convert $\mathcal{M}$ into a model as defined in (18) on the handout "An Algebraic Perspective on Propositional Logic." That is, state what D and I should be under the new definition in (18).

I(a)=a, I(b)=b, I(c)=c

I(P) = {<a, 1>, <b, 0>, <c, 0>}

$$I(Q) = \begin{bmatrix} a & \rightarrow & \begin{bmatrix} a & \rightarrow 0 \\ b & \rightarrow 1 \\ c & \rightarrow 1 \end{bmatrix} \\ \\ b & \rightarrow & \begin{bmatrix} a & \rightarrow 0 \\ b & \rightarrow 0 \\ c & \rightarrow 1 \end{bmatrix} \\ \\ c & \rightarrow & \begin{bmatrix} a & \rightarrow 0 \\ b & \rightarrow 0 \\ c & \rightarrow 0 \end{bmatrix} \end{bmatrix}$$

b. Please use the new definition of 'valuation' in (19) on the handout "An Algebraic Perspective on Propositional Logic" to show how the converted model $\mathcal{M}$ assigns truth-values to the following formulae.

(i) <u>∃x(Px)</u>

- Let g be any variable assignment based on $\mathcal{M}$.

1. $V_{M,g}(\exists x(Px)) = 1$ *iff* (by definition of Ext and Concat)
2. $V_{M,g}(\text{Ext}(x, \text{Concat}(P, x))) = 1$ *iff* (by 19vi)
3. There is an a ∈ D such that $V_{M,g^{(x/a)}}(\text{Concat}(P, x)) = 1$ *iff* (by 19i)
4. There is an a ∈ D such that $I(P)([\![x]\!]^{M,g^{(x/a)}}) = 1$ *iff*
5. There is an a ∈ D such that $I(P)(g^{(x/a)}(x)) = 1$ *iff*
6. There is an a ∈ D such that $I(P)(a) = 1$

Since there is an a ∈ D such that I(P)(a) = 1, namely "a", ∃x(Px) is true.

ii. Let g be any variable assignment based on M.

1. $V_{M,g}(\forall x \exists y((Qy)x)) = 1$ *iff* (by definition of FOL)
2. $V_{M,g}(\text{All}(x, \text{Ext}(y, \text{Concat}(\text{Concat}(Q, y), x)))) = 1$ *iff* (by 19vii)
3. For every $\alpha \in D$, $V_{M,g(x/\alpha)}(\text{Ext}(y, \text{Concat}(\text{Concat}(Q, y), x))) = 1$ *iff* (by 19vi)
4. For every $\alpha \in D$, there is a $\beta \in D$ such that $V_{M,g(x/\alpha)(y/\beta)}(\text{Concat}(\text{Concat}(Q, y), x)) = 1$ *iff* (by 19i)
5. For every $\alpha \in D$, there is a $\beta \in D$ such that $I(Q)([\![y]\!]^{M,g(x/\alpha)(y/\beta)})([\![x]\!]^{M,g(x/\alpha)(y/\beta)}) = 1$ *iff* (by definition of $[\![]\!]^{M,g}$)
6. For every $\alpha \in D$, there is a $\beta \in D$ such that $I(Q)(g(x/\alpha)(y/\beta)(y))(g(x/\alpha)(y/\beta)(x)) = 1$ *iff* (by definition of g)
7. For every $\alpha \in D$, there is a $\beta \in D$ such that $I(Q)(\beta)(\alpha) = 1$
8. Therefore, $V_{M,g}(\forall x \exists y((Qy)x)) = 0$ given that $a \in D$ and nothing precedes a alphabetically (there is no $\beta \in D$ such that $I(Q)(\beta)(a) = 1$)

(5) **Exercise on 'Interpretations' for PL**
Let <{0,1}, Neg, Conj, Disj, Imp, V > be an interpretation for PL, as defined in (29) of "An Algebraic Perspective on Propositional Logic." Moreover, assume that V is such that V(p) = 1, V(q) = 0, and V(r) = 1. Please use the assumption that V is a homomorphism to calculate truth-values of the following formulae.

a. $\underline{\sim(p \;\&\; (p \rightarrow (q \vee r)))}$

1. $V(\sim(p \;\&\; (p \rightarrow (q \vee r))))$ = (by definition of PL)
2. V(Not(And(p, If(p, Or(q, r))))) = (by homomorphic property of V)
3. Neg(V(And(p, If(p, Or(q, r))))) = (by homomorphic property of V)
4. Neg(Conj(V(p), V(If(p, Or(q, r))))) = (by homomorphic property of V)
5. Neg(Conj(V(p), Imp(V(p), V(Or(q, r))))) = (by homomorphic property of V)
6. Neg(Conj(V(p), Imp(V(p), Disj(V(q), V(r))))) = (by definition of V)
7. Neg(Conj(1, Imp(1, Disj(0, 1)))) = (by definition of Disj)
8. Neg(Conj(1, Imp(1, 1))) = (by definition of Imp)
9. Neg(Conj(1, 1)) = (by definition of Conj)
10. Neg(1) = (by definition of Neg)
11. 0

b. Calculation for $(((p \vee \neg r) \rightarrow q) \vee p)$

1.	$V(((p \vee \neg r) \rightarrow q) \vee p)$	=	(by defs of PL)
2.	$V(\text{Or}(\text{If}(\text{Or}(p, \text{Not}(r)), q), p))$	=	(by HP of V)
3.	$\text{Disj}(V(\text{If}(\text{Or}(p, \text{Not}(r)), q)), V(p))$	=	(by HP of V)
4.	$\text{Disj}(\text{Imp}(V(\text{Or}(p, \text{Not}(r)), V(q)), V(p))$	=	(by HP of V)
5.	$\text{Disj}(\text{Imp}(\text{Disj}(V(p), V(\text{Not}(r))), V(q)), V(p))$	=	(by HP of V)
6.	$\text{Disj}(\text{Imp}(\text{Disj}(V(p), \text{Neg}(V(r))), V(q)), V(p))$	=	(by def of V)
7.	$\text{Disj}(\text{Imp}(\text{Disj}(1, \text{Neg}(1)), 0), 1)$	=	(by def. of Neg)
8.	$\text{Disj}(\text{Imp}(\text{Disj}(1, 0), 0), 1)$	=	(by def. of Disj)
9.	$\text{Disj}(\text{Imp}(1, 0), 1)$	=	(by def. of Imp)
10.	$\text{Disj}(0, 1)$	=	(by def. of Disj)
11.	1		

c.

1.	$V\big((((p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)) \wedge ((p \rightarrow q) \rightarrow (\neg p \vee q)))\big) =$	(by definition of PL)
2.	$V\big(\text{And}\big(\text{If}(\text{If}(p,q),\text{If}(\text{Not}(q),\text{Not}(p))),$ $\text{If}(\text{If}(p,q),\text{Or}(\text{Not}(p),q))\big)\big) =$	(by homomorphism property of V)
3.	$\text{Conj}\big(V\big(\text{If}(\text{If}(p,q),\text{If}(\text{Not}(q),\text{Not}(p)))\big),$ $V\big(\text{If}(\text{If}(p,q),\text{Or}(\text{Not}(p),q))\big)\big) =$	(by homomorphism property of V 2x)
4.	$\text{Conj}\big(\text{Imp}\big(V\big(\text{If}(p,q)\big),V\big(\text{If}(\text{Not}(q),\text{Not}(p))\big)\big),$ $\text{Imp}\big(V\big(\text{If}(p,q)\big),V\big(\text{Or}(\text{Not}(p),q)\big)\big)\big) =$	(by homomorphism property of V 4x)
5.	$\text{Conj}\big(\text{Imp}\big(\text{Imp}\big(V(p),V(q)\big),\text{Imp}\big(V(\text{Not}(q)),V(\text{Not}(p))\big)\big),$ $\text{Imp}\big(\text{Imp}\big(V(p),V(q)\big),\text{Disj}\big(V(\text{Not}(p)),V(q)\big)\big)\big) =$	(by homomorphism property of V 3x)
6.	$\text{Conj}\big(\text{Imp}\big(\text{Imp}\big(V(p),V(q)\big),\text{Imp}\big(\text{Neg}(V(q)),\text{Neg}(V(p))\big)\big),$ $\text{Imp}\big(\text{Imp}\big(V(p),V(q)\big),\text{Disj}\big(\text{Neg}(V(p)),V(q)\big)\big)\big) =$	(by definition of V)
7.	$\text{Conj}\big(\text{Imp}\big(\text{Imp}(1,0),\text{Imp}\big(\text{Neg}(0),\text{Neg}(1)\big)\big),$ $\text{Imp}\big(\text{Imp}(1,0),\text{Disj}\big(\text{Neg}(1),0\big)\big)\big) =$	(by definition of Neg 3x)
8.	$\text{Conj}\big(\text{Imp}\big(\text{Imp}(1,0),\text{Imp}(1,0)\big),\text{Imp}\big(\text{Imp}(1,0),\text{Disj}(0,0)\big)\big) =$	(by definition of Disj)
9.	$\text{Conj}\big(\text{Imp}\big(\text{Imp}(1,0),\text{Imp}(1,0)\big),\text{Imp}\big(\text{Imp}(1,0),0\big)\big) =$	(by definition of Imp 3x)
10.	$\text{Conj}\big(\text{Imp}(0,0),\text{Imp}(0,0)\big) =$	(by definition of Imp 2x)
11.	$\text{Conj}(1,1) =$	(by definition of Conj)
12.	1	

(6) **Exercise on Homomorphisms and Compositions**
Let **A** < A, f_1, … f_n > and **B** < B, g_1, … g_n > and **C** < C, h_1, … h_n > all be algebras. Assume that k: A→ B is a homomorphism from **A** to **B**, and that j: B → C is a homomorphism from **B** to **C**. Please show that j°k is a homomorphism from **A** to **C**.

a. Given the definition of function composition, since k is a function from A to B and j is a function from B to C, $j \circ k$ is a function from A to C.

b. It follows from k being a homomorphism from $\mathscr{A}$ to $\mathscr{B}$ that $\mathscr{A}$ and $\mathscr{B}$ are homomorphic. Therefore, given the definition of homomorphic, there is a one-to-one correspondence between the operations $f_1,\dots,f_n$ and $g_1,\dots,g_n$ and $\forall i \in \mathbb{N}$, f_i is of the same arity as g_i.

It follows from j being a homomorphism from $\mathscr{B}$ to $\mathscr{C}$ that $\mathscr{B}$ and $\mathscr{C}$ are homomorphic. Therefore, given the definition of homomorphic, there is a one-to-one correspondence between the operations $g_1,\dots,g_n$ and $h_1,\dots,h_n$ and $\forall i \in \mathbb{N}$, g_i is of the same arity as h_i.

Therefore, since there are one-to-one correspondences between the operations $f_1,\dots,f_n$ and $g_1,\dots,g_n$ and the operations $g_1,\dots,g_n$ and $h_1,\dots,h_n$, there is a one-to-one correspondence between the operations $f_1,\dots,f_n$ and $h_1,\dots h_n$. This is illustrated below.

$$\begin{tikzcd}
f_1 \arrow[d, dash] & , & f_2 \arrow[d, dash] & , & \dots & , & f_n \arrow[d, dash] \\
g_1 \arrow[d, dash] & , & g_2 \arrow[d, dash] & , & \dots & , & g_n \arrow[d, dash] \\
h_1 & , & h_2 & , & \dots & , & h_n
\end{tikzcd}$$

Therefore, since $\forall i \in \mathbb{N}$, f_i is of the same arity as g_i and g_i is of the same arity as h_i, it follows that $\forall i \in \mathbb{N}$, f_i is of the same arity as h_i.

c. i. It follows from k being a homomorphism from $\mathscr{A}$ to $\mathscr{B}$ that:
$\forall i \in \mathbb{N}$, $k\big(f_i(a_1,\dots,a_m)\big) = g_i\big(k(a_1),\dots,k(a_m)\big)$.

ii. It follows from j being a homomorphism from $\mathscr{B}$ to $\mathscr{C}$ that:
$\forall i \in \mathbb{N}$, $j\big(g_i(a_1,\dots,a_m)\big) = h_i\big(j(a_1),\dots,j(a_m)\big)$.

iii. Given the definition of function composition, $j \circ k(x) = j(k(x))$.

iv. $\forall i \in \mathbb{N}$,

1. $j \circ k\big(f_i(a_1,\dots,a_m)\big) =$ (by iii)
2. $j\big(k\big(f_i(a_1,\dots,a_m)\big)\big) =$ (by i)
3. $j\big(g_i\big(k(a_1),\dots,k(a_m)\big)\big) =$ (by ii)
4. $h_i\big(j(k(a_1)),\dots,j(k(a_m))\big) =$ (by iii)
5. $h_i\big(j \circ k(a_1),\dots,j \circ k(a_m)\big)$

v. Thus, $\forall i \in \mathbb{N}$, $j \circ k\big(f_i(a_1,\dots,a_m)\big) = h_i\big(j \circ k(a_1),\dots,j \circ k(a_m)\big)$

Therefore, given (a), (b), and (c), $j \circ k$ is a homomorphism from $\mathscr{A}$ to $\mathscr{C}$. □