An Algebraic Perspective on the Syntax of First Order Logic (Without Quantification) [1]

1. Statement of the Problem, Outline of the Solution to Come

(1) The Key Problem

- There is much to recommend an 'algebraic' treatment of interpretation, where it is conceived of as a homomorphism from a 'syntactic' algebra to a 'semantic' one.

- Based on our algebraic semantics for PL, we were lead to the (preliminary) generalization in (2) below.

- However, its not possible to directly extend this conception of interpretation to FOL:
  - Unlike PL, the structure < WFF_{FOL} , Concat, Not, And, Or, If, Ext, All > is *not* an algebra, **because WFF_{FOL} isn't closed under those operations…**

(2) Algebraic Perspective on 'Interpretation' (To Be Revised)

Let $L = < A, f_1, \ldots, f_n >$ be a 'syntactic algebra' for a given language. An *interpretation* of L is a structure $< B, g_1, \ldots, g_n, j>$ such that:

a. $< B, g_1, \ldots, g_n >$ is an algebra.
b. For any i, f_i and g_i have the same arity.
c. j is a function from the 'non-logical constants' in A to B.

If **B** = $<B, g_1, \ldots, g_n, j>$ is an interpretation of L, then the *meaning assignment* for L determined by **B** is the unique homomorphism h from L to $<B, g_1, \ldots, g_n>$ such that $j \subseteq h$.

(3) General Outline of a Solution

We're going to loosen what it means to be a 'syntactic algebra' for a given language.

- In a 'syntactic algebra' for L, $< A, f_1, \ldots, f_n >$, the set A can be a *strict superset* of the well-formed sentences of the language.

- For FOL, we'll construct an algebra < A, Concat, Not, And, Or, If, Ext, All > such that $WFF_{FOL} \subset A$.

- We'll then see how we can define/characterize WFF_{FOL} in terms of this algebra (with a few added formal ingredients)

- **In the next notes, we'll build an algebraic semantics for WFF_{FOL} from these tools**

[1] These notes are based upon material in the following readings: Halvorsen & Ladusaw (1979), Dowty *et al.* (1981) Chapter 8, and Thomason (1974) Chapter 7 (Montague's "Universal Grammar").

(4) **Zooming In: Obtaining WFF_{FOL} From Our Syntactic Algebra**

We'll construct WFF_{FOL} from < A, Concat, Not, And, Or, If, Ext, All > by using a 'generate and filter' approach not too unlike classic GB:

- In the algebra < A, Concat, Not, And, Or, If, Ext, All >, A will contain all the WFF's of FOL, *but also a whole bunch of 'syntactic garbage'* (e.g., (P & a), ~x, *etc.*)
- We'll then show how syntactic rules function as a kind of *filter* over A (in the GB sense), separating out the WFFs from the 'syntactic garbage'.
- We'll use these rules ('syntactic filters') to formally define that subset of A which is the well-formed formulae of FOL.
- In the next notes, we'll make a minimal change to (2) so that it will straightforwardly extend to the complex system consisting of the algebra and these 'filters'.
- **Funky Property of Our Resulting Semantics:**
  - We'll still keep the idea that an interpretation of FOL is a homomorphism from < A, Concat, Not, And, Or, If, Ext, All > to a 'semantic algebra'.
  - **Thus, our 'algebraic semantics' will interpret – not only the WFFs of FOL – *but also all the syntactic garbage (!!!)***

As always, though, to make our lives easier, we'll start out by considering only the set of sentences of FOL that don't have any quantifiers.
We'll also assume a minimal set of sentence connectives (&, ~)…

(5) **Fragment of FOL We Will Focus on For Now: 'FOL-NoQ'**
The set of well-formed formulae of FOL-NoQ, $WFF_{FOL\text{-}NoQ}$, is the smallest set such that:

a. If φ is an n-ary predicate letter and each of α_1, …, α_n is an individual constant, then Concat(…(Concat(Concat(φ,α_1), α_2), …, α_n) ∈ $WFF_{FOL\text{-}NoQ}$

b. If φ ∈ WFF, then Not(φ) ∈ WFF

c. If φ, ψ ∈ WFF, then And(φ,ψ) ∈ WFF

Examples of WFF of FOL-NoQ: (((Qa)b) & ~(Rc))
~(~(Ab) & ~((Sc)d)) (≈ ((Ab) v ((Sc)d)))
~(((Tg)b) & ~(Lg)) (≈ (((Tg)b) → (Lg)))

2. An Algebraic Characterization of $WFF_{FOL\text{-}NoQ}$

In this section, we're going to show how the set $WFF_{FOL\text{-}NoQ}$ can be characterized in terms of a syntactic algebra. There will be six major steps:

1. Building the **Syntactic Algebra**
2. Introducing the **Syntactic Category Labels** (*i.e.*, the Types)
3. Using the Category Labels to Define the **'Basic Expressions'** of FOL-NoQ (*i.e.*, the non-logical constants)
4. Introducing the **Syntactic Rules** of FOL-NoQ
5. Using the Syntactic Rules and the Basic Expressions (non-logical constants) to Obtain the **Syntactic Categories** of FOL-NoQ
6. Putting It All Together: Defining $WFF_{FOL\text{-}NoQ}$ in Terms of these Ingredients

Note:
As we'll see, the resulting system generating $WFF_{FOL\text{-}NoQ}$ can be viewed as a kind of abstract mathematical characterization of 'what's going on' in recursive definitions like (5).

2.1 Building the Syntactic Algebra

(6) **The Syntactic Algebra**
Let < A, Concat, Not, And > be the algebra such that:

a. Concat, Not, And are the syntactic operations defined previously.

b. A is the smallest set such that
   (i) If Φ is a predicate letter, then $\Phi \in A$
   (ii) If α is an individual constant, then $\alpha \in A$
   (iii) A is closed under Concat, Not, And

(7) **Remarks**

a. $WFF_{FOL\text{-}NoQ} \subseteq A$
   - Anything in $WFF_{FOL\text{-}NoQ}$ can be created by iterated application of Concat, Not, And to the predicate letters and individual constants.

b. $WFF_{FOL\text{-}NoQ} \neq A$
   - A includes such 'syntactic garbage' as (P & a), ~x, (ab).

c. Not every string in the vocabulary of FOL-NoQ is in A (*e.g.,* P~, &(aB))

(8) a. Question:
Since $WFF_{FOL\text{-}NoQ} \subseteq A$, why not equate FOL-NoQ with the algebra <A, Concat, Not, And>, like we did for PL?

b. Answer:
There's not enough 'information' in <A, Concat, Not, And> alone to construct/define the set $WFF_{FOL\text{-}NoQ}$

- But, if we *paired* this algebra with some other formal devices that could be used to construct/define $WFF_{FOL\text{-}NoQ}$ …
- …**Then we could equate the language FOL-NoQ with that system!**

This will probably become a bit clearer to you once you actually see how we do this…

2.2 Introducing the Syntactic Category Labels (The Types)

(9) **Background: Syntactic Categories and Syntactic Category Labels**

a. Syntactic Categories
We can view the 'syntactic categories' of a given language sets of strings in the vocabulary of that language.

Illustration:

- We could say that the category 'NP' in English is the set of strings { dog, student of history, ugly man with a telescope, big funny book about bees, … }
- We could say that the category 'VP' in English is the set of strings { runs, ate a pickle, laughs in the face of danger, always sings horribly, …}

b. Syntactic Category Labels
If syntactic categories are sets of strings, we could view the 'category labels' as being an index set, used to index the family of categories. (See Handout 1).

Illustration:

- Category Labels: {NP, N, VP, V, PP, P, DP, D, S, AP, A, …. }
- Syntactic Categories of English:
  - $C_{NP}, C_{N}, C_{VP}, C_{N}, C_{V}, C_{PP}, C_{P}, C_{DP}, C_{D}, C_{S}, C_{AP}, C_{A}, \ldots$
  - $\{ C_{\delta} \}_{\delta \in \{NP, N, VP, V, PP, P, DP, D, S, AP, A, \ldots.\}}$

We're now going to define a set of syntactic category labels and syntactic categories for FOL…
These will later be used to define a set of syntactic rules that can be used to 'extract' $WFF_{FOL\text{-}NoQ}$ from the set A….
The category labels will be very familiar to the semanticists…

(10) **The Types**

The set T of types is the smallest set such that:

a. $e \in T$
- *This is the category label for expressions denoting 'entities'; i.e., the terms*

b. $t \in T$
- *This is the category label for things denoting 'truth-values'; i.e., the WFFs.*

c. If $\sigma \in T$ and $\tau \in T$, then $\langle\sigma, \tau\rangle \in T$
- $\langle\sigma, \tau\rangle$ *will be the category label for things denoting functions from* σ *to* τ
- *Thus,* $\langle e,t\rangle$ *is the category label for functions from entities to truth-values (i.e., the unary predicates)…*

2.3 Defining the Basic Expressions of FOL-NoQ

(11) **Basic Expressions of a Category**
If δ is a category label, then X_δ are all the basic (primitive / lexical) expressions of category δ.

Illustration for English:

- X_N = *The set of nouns in the English lexicon.*
- X_{NP} = { one }[2]
- X_V = *The set of verbs in the English lexicon.*
- X_{VP} = { so }[3]
- X_P = { in, on, under, from, with … }
- X_{PP} = { there, then, … }
- X_D = {a, the, every, most, … }
- X_{DP} = { he, she, it, they, … }
- X_S = $\varnothing$

We can use these sets X_δ *in conjunction with the syntactic rules (to be defined below) to generate the full set of categories* C_δ for the language.

[2] This would be 'anaphoric *one*' in English, which some analyze as an NP pronouns (*e.g.* 'a student of chemistry from France, and **one** from Germany').
[3] This would be the VP-proform 'so', that appears in sentences like 'Mary went to the park, and Frank did **so** too'.

(12) **The Basic Expressions of FOL-NoQ**

a. X_e = The set of individual constants, CONS

b. $X_{<e, \ldots, t>}$ = The set of *n*-ary predicate letters, where *n* = the number of times e appears in the category label $X_{<e, \ldots, t>}$

Illustration: $X_{<et>}$ = The set of unary predicate letters.
$X_{<e<et>>}$ = The set of binary predicate letters.
$X_{<e<e<et>>>}$ = The set of ternary predicate letters, *etc.*

c. For all other types $\tau \in T$, $X_\tau = \varnothing$.

Note: This states that our 'lexicon' for FOL-NoQ contains (i) individual constants, and (ii) n-ary predicate letters for every $n \in \mathbb{N}$, **but nothing else.** (i.e., the non-logical constants)

2.4 Introducing the Syntactic Rules of FOL-NoQ

Another important use of the syntactic category labels is in stating syntactic rules. Recall the syntactic rules that we had constructed for PL and FOL in the last handout.

(13) **Abstract Representation of Syntactic Rules for FOL (To Be Revised)**

a. < Not, <WFF>, WFF>
'The result of applying Not to a member of WFF is a WFF'

b. < And, <WFF, WFF>, WFF>
'The result of applying And to a pair consisting of a WFF and a WFF is a WFF'

Recall that the WFFs of FOL-NoQ are going to be of category *t*. We could then rewrite the key rules in (13a,b) as follows:

(14) **Syntactic Rules for FOL-NoQ (To Be Revised)**

a. < Not, < t >, t>
'The result of applying Not to a member of category t is also a member of category t'.

b. < And, <t, t>, t>
'The result of applying And to a pair consisting of a member of category t and a member of category t is itself also a member of category t.

Finally, in addition to the rules in (14a,b), let's also add every rule of the general form in (15c), where $\sigma, \tau \in T$.

(15) **Syntactic Rules for FOL-NoQ (Final Version)**

a. $< Not, < t >, t>$

b. $< And, <t, t>, t>$

c. $< Concat < <\sigma, \tau>, \sigma >, \tau >$
'The result of concatenating a member of category $<\sigma, \tau>$ with a member of category σ will be a member of category τ.'

IMPORTANT NOTE:
(15c) is a kind of 'meta-rule', a short hand for representing *an infinite set* of rules. It encompasses all the following (concrete) rules:

- $< Concat < <e, t>, e >, t >$
'The result of applying 'Concat' to a member of category <e,t> and a member of category e will be a member of category t.'

  Illustration: If P is of category <et>, and a is of category e, then (Pa) is of category t

- $< Concat < <e, <e, t>>, e >, <e,t> >$
'The result of applying 'Concat' to a member of category <e, <e,t>> and a member of category e will be a member of category <e,t>.'

  Illustration: If R is of category <e, <e, t>>, and a is of category e, then (Ra) is of category <et>.
  Consequently, if b is of category e, then ((Ra)b) is of category t

2.5 Using the Rules and Basic Expressions to Generate the Syntactic Categories

With our basic expressions in (12) and our syntactic rules in (15), we can define the syntactic category C_δ for every category label δ.

(16) **Definition of the Syntactic Categories C_δ**
For any category label $\tau \in T$, C_τ is the smallest set such that both the following hold:

a. $X_\tau \subseteq C_\tau$

b. If (i) $< f , < \sigma_1, \ldots, \sigma_n>, \tau >$ is a syntactic rule, and (ii) $\varphi_1, \ldots, \varphi_n$ are such that each $\varphi_i \in C_{\sigma i}$, then $f(\varphi_1 \ldots \varphi_n) \in C_\tau$

In plain English, the (full) category C_τ *will contain all the basic (lexical) expressions of that category, as well as all the complex expressions of that category that you can create from the syntactic rules!*

- Note, then, that the definition in (16) is simply a formal, mathematically precise statement of how we've been informally reading rules like (15).

(17) **Illustration of the Definitions in (16)**

a. If P is a unary predicate letter, then $P \in X_{<et>}$ (12b), and so $P \in C_{<et>}$ (16a).

b. If Q is a binary predicate letter, then $Q \in X_{<e,<e,t>>}$, and so $Q \in C_{<e,<e,t>>}$

c. If a,b are individual constants, then $a,b \in X_e$ (12a), and so $a,b \in C_e$ (16a).

d. If P is a unary predicate letter, and a is an individual constant, then $(Pa) \in C_t$
   - $P \in C_{<et>}$ (17a)
   - $a \in C_e$ (17c)
   - < Concat < <e, t>, e >, t > is a syntactic rule (15c)
   - $Concat(P,a) \in C_t$ (16b)
   - $(Pa) \in C_t$

e. If Q is a binary predicate letter, and a,b are individual constants, then $((Qa)b) \in C_t$
   - $Q \in C_{<e,<e,t>>}$ (17b)
   - $a,b \in C_e$ (17c)
   - < Concat < <e, <e, t>>, e >, <e,t> > is a rule (15c)
   - $Concat(Q,a) \in C_{<et>}$ (16b)
   - $(Qa) \in C_{<et>}$
   - < Concat < <e, t>, e >, t > is a syntactic rule (15c)
   - $Concat((Qa),b) \in C_t$ (16b)
   - $((Qa)b) \in C_t$

f. If P is a unary predicate letter, and a is an individual constant, then $\sim(Pa) \in C_t$
   - $(Pa) \in C_t$ (17d)
   - < Not, < t >, t> is a syntactic rule (15a)
   - $Not((Pa)) \in C_t$ (16b)
   - $\sim(Pa) \in C_t$

g. If P is a unary predicate letter, Q is a binary predicate letter, and a,b are individual constants, then $(((Qa)b) \ \& \ \sim(Pa)) \in C_t$
   - $\sim(Pa) \in C_t$ (17f)
   - $((Qa)b) \in C_t$ (17e)
   - < And, <t, t>, t> is a syntactic rule (15b)
   - $And(((Qa)b), \sim(Pa)) \in C_t$ (16b)
   - $(((Qa)b) \ \& \ \sim(Pa)) \in C_t$

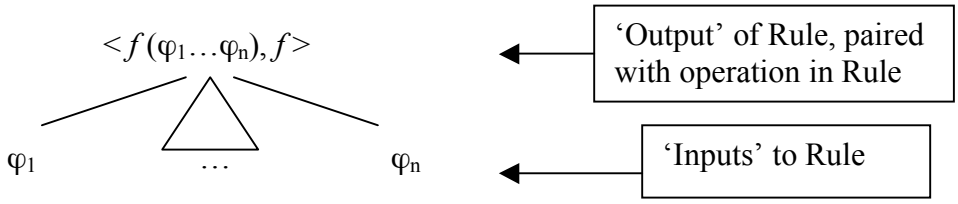
(18) **Introducing (Montagovian) Analysis Trees**

- A handy way of representing calculations like those in (17a-g) is through the use of trees of form in (18a).

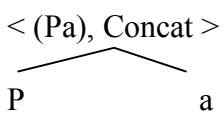
- These trees represent how a particular syntactic rules works to create an expression of a particular category.

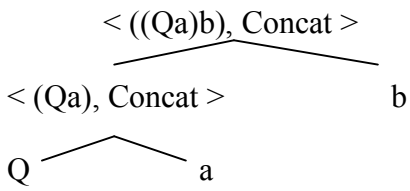
a. General Form of An Analysis Tree

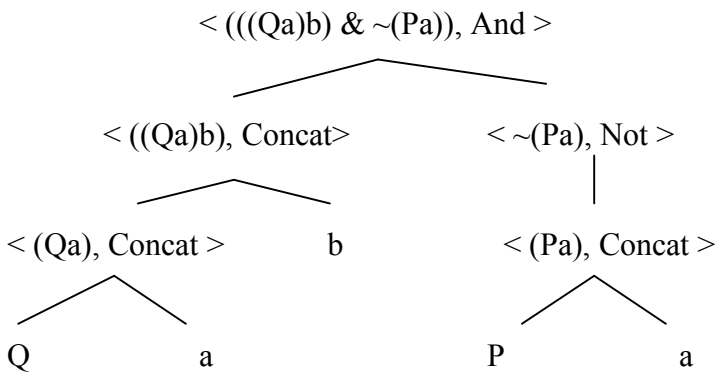
Suppose the rule $< f, < \sigma_1, \ldots, \sigma_n >, \tau >$ with (16) entails that $f(\varphi_1 \ldots \varphi_n) \in C_\tau$
This can be represented via a tree of the following form:



b. Illustrations

(i)


(ii)


(iii)


Note: *Although these are superficially similar to PS trees, they are importantly different, in that the non-terminal nodes do not list the category of the resulting expression, but rather the syntactic operation used to obtain it…*

2.6 Putting It All Together: Defining $WFF_{FOL\text{-}NoQ}$ in Terms of these Ingredients

(19) **The Crucial Observation:** $C_t = WFF_{FOL\text{-}NoQ}$

That is, we can take our syntactic algebra < A, Concat, Not, And > and obtain that subset of A equal to $WFF_{FOL\text{-}NoQ}$ by adding:

a. A set of category labels T (10)
b. A set of 'basic expressions' X_τ , for each $\tau \in T$ (12)
c. A set of syntactic rules (15)

Given the definition in (16) for creating C_τ from these ingredients (for every $\tau \in T$), we are able to obtain the set $C_t = WFF_{FOL\text{-}NoQ}$

Thus, if we add the ingredients in (19a,b,c) to our syntactic algebra <A, Concat, Not, And>, we will have 'enough information' to obtain the set $WFF_{FOL\text{-}NoQ}$...

…Thus, given the comment in (8b), we *could* abstractly characterize FOL-NoQ as in (20)

(20) **Algebraic Characterization of FOL-NoQ**

The language FOL-NoQ is the structure $< A, Concat, Not, And, X_\tau, S, t >_{\tau \in T}$, where:

a. < A, Concat, Not, And > is the algebra defined in (6)

b. $< \ldots X_\tau \ldots >_{\tau \in T}$ is shorthand for the infinite family of basic expressions X_τ defined in (12).

c. S is the (infinite) set of syntactic rules defined in (15).

d. t is the category label τ such that C_τ is the set of WFFs of FOL-NoQ (19).

Again, we've seen how the elements of the structure $< A, Concat, Not, And, X_\tau, S, t >_{\tau \in T}$ can be used to obtain $WFF_{FOL\text{-}NoQ}$

- For this reason, we are now justified in equating FOL-NoQ with this structure.
- Note that $< A, Concat, Not, And, X_\tau, S, t >_{\tau \in T}$ is not actually itself an algebra. Rather, it contains an algebra as one of its proper parts.

3. Generalizing Our Treatment of FOL-NoQ to All Languages

In Montague Grammar, our method for defining FOL-NoQ by means of a 'syntactic algebra' is generalized to all languages, in (approximately) the following way.

(21) **General Definition of a Language (To be Revised)**

A language *L* is a structure $< A , F_\gamma , X_\delta , S, \delta_0 >_{\gamma \in \Gamma , \delta \in \Delta}$ such that:

a. $< A , F_\gamma >_{\gamma \in \Gamma}$ is an algebra.

- Note: the set Γ is an index set, used to index the (syntactic) operations in the (syntactic) algebra $< A , F_\gamma >_{\gamma \in \Gamma}$

b. A is the smallest set such that:
   (i) For all $\delta \in \Delta$, $X_\delta \subseteq A$.
   (ii) A is closed under the operations F_γ for all $\gamma \in \Gamma$

- Note: The set Δ is the set of category labels, being used to index each of the set of 'basic categories' (lexical items) X_δ

- Note: The conditions in (i) and (ii) are what give us that $< A , F_\gamma >_{\gamma \in \Gamma}$ is a 'syntactic algebra', where A contains all the well-formed structures of the language *L* (as well as a ton of junk).

c. S is a set of sequences of the form $< F_\gamma , < \delta_1, \ldots, \delta_n >, \delta >$, where $\gamma \in \Gamma$, F_γ is an n-ary operation, and $\delta_1, \ldots, \delta_n, \delta \in \Delta$

- Note: This condition states that a syntactic rule consists of:
  (i) some n-ary operation F_γ from the algebra $< A , F_\gamma >_{\gamma \in \Gamma}$
  (ii) a n-tuple of category labels from the set of labels Δ (the categories of the expressions 'input' to F_γ)
  (iii) some category label $\delta \in \Delta$ (the category of the resulting output)

d. $\delta_0 \in \Delta$
- Note: δ_0 will be the category label of the 'sentences' of *L*.
- Note: Given that a language *L* is often characterized as a 'set of sentences', this element δ_0 will allow us to define such a set *L* given our structure above.

Key Observation:
Our 'algebraic characterization' of FOL-NoQ in (20) is a specific instance of the general kind of structure in (21).

With the general definition in (21) at our disposal, we can also offer the following general definition of what it is to be a 'syntactic category' of the language *L*.

(22) **General Definition of Syntactic Categories of a Language**
Let *L* be a language $< A , F_\gamma , X_\delta , S, \delta_0 >_{\gamma \in \Gamma , \delta \in \Delta}$. Language *L* **generates the family CAT of syntactic categories iff:**

a. CAT is a family of subsets of A, indexed by Δ

- Note: This means that the members of CAT are the sets C_δ , for all $\delta \in \Delta$
- Note: This all means that for all $\delta \in \Delta$, $C_\delta \subseteq A$.

b. $X_\delta \subseteq C_\delta$, for all $\delta \in \Delta$

- Note: This means that for every category label $\delta \in \Delta$, the category C_δ contains all the 'basic expressions' (lexical items) of that category.

c. If the sequence $< F_\gamma , < \delta_1, \ldots, \delta_n >, \delta > \in S$, and $\varphi_1, \ldots, \varphi_n$ are such that $\varphi_i \in C_{\delta i}$, then $F_\gamma(\varphi_1 \ldots \varphi_n) \in C_\delta$

- Note: This is just a straightforward generalization of (16b)
- Note: This says that C_δ will contain – not just the basic expressions lexical items X_δ - but also all the expressions of category δ you can generate via the syntactic rules.

d. For each $C_\delta \in$ CAT, C_δ is the smallest set satisfying conditions (22a)-(22c).

(23) **The 'Meaningful Expressions' of a Language *L***
Let *L* be a language $< A , F_\gamma , X_\delta , S, \delta_0 >_{\gamma \in \Gamma , \delta \in \Delta}$ and let *L* generate the family CAT of syntactic categories. The **meaningful expressions of *L* (ME_L)** is the union of all the categories in CAT (*i.e.*, $\cup_{\delta \in \Delta} C_\delta$)

4. Applying Our Definition of a 'Language' to (a Fragment of) English

If (21) truly is a viable definition of what a 'language' is, then it should be possible to represent a *natural language*, like English, as such a system.

- In this section, we'll show that this is possible for a subpart (fragment) of English, roughly corresponding in its expressive power to FOL-NoQ

(24) **Some Comments**

a. In representing (a fragment of) English as a system like (21), we are basically giving a theory of English syntax.

- I.e., the system responsible for constructing complex expressions of English from the primitive lexical items.

b. One early and perennial criticism of Montague Grammar was its accompanying theory of syntax, stated in (21).

c. Note, though, that the intellectual motivation behind Montague's theory of syntax was *completely different* from that of generative linguists.

- Montague wanted a theory broad enough to cover *anything that we'd reasonably call a 'language' (including artificial and non-human ones)*

- Thus, it's a *feature* – not a *bug* – that his system of syntax doesn't (necessarily) capture generalizations regarding the structure of human languages.

(25) **The Syntactic Category Labels**
Let the set Δ consist of the following syntactic category labels:

a. NP (noun phrases; proper names for now)
b. IV (intransitive verbs *and derived verb phrases*)
c. TV (transitive verbs)
d. S (sentences)

(26) **The Basic Expressions (The Lexicon)**
For each of the category labels above, the basic expressions of that category are:

a. X_{NP} = { *Barack, Michelle, Mitt* }
b. X_{IV} = { *smokes* }
c. X_{TV} = { *loves* }
d. X_{S} = ∅

(27) **The Syntactic Operations**

a. Merge:
Binary operation that maps two strings 'x', 'y' to the string 'x y'

Illustration: Merge(*loves*, *Mitt*) = *loves Mitt*

b. And_E
Binary operation that maps two strings 'x', 'y' to the string 'x *and* y'

Illustration: And_E(*Mitt*, *Barack*) = *Mitt and Barack*

c. Not_E
Unary operation that maps a string 'x' to the string '*it is not the case that* x'

Illustration: Not_E(*Barack smokes*) = *It is not the case that Barack smokes*

(28) **The Syntactic Rules**
Let the set S_E consist of the following tuples:

a. < Merge, < TV, NP >, IV >
'The result of merging a TV with an NP is a IV'

b. < Merge, < NP, IV >, S >
'The result of merging an NP with an IV is an S'

c. < And_E , <S , S> , S >
'The result of applying And_E to an S and an S is an S'

d. < Not_E, <S>, S >
'The result of applying Not_E to an S is an S'.

(29) **The Definition of 'Mini-English'**

The language 'Mini-English' is the structure < A, Merge, And_E, Not_E, X_δ , S_E, S > $_{\delta \in \Delta}$ such that:

a. A is the smallest set such that:
(i) For all $\delta \in \Delta$, $X_\delta \subseteq A$.
(ii) A is closed under the operations Merge, And_E and Not_E

b. Merge, And_E, Not_E, X_δ , S_E, S, and Δ are all as defined in (25)-(29)

(30) **Some Remarks**

a. The structure 'Mini-English' in (29) clearly satisfies our general definition of a 'language' in (21).

b. The set A contains infinitely many well-formed sentences of English (*Mitt smokes, Barack loves Michelle and Michelle smokes,* etc.)

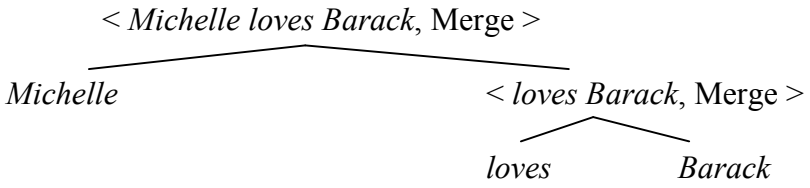
c. The set A also contains a whole bunch of 'syntactic garbage' (*smokes loves*, *it is not the case that Mitt*, *Michelle Barack*, etc.)

d. We can use the rules in S_E and the general definition in (22) to generate the syntactic categories C_δ of English.
- As with FOL-NoQ, we can use 'analysis trees' as a handy way of representing the calculations (31).

e. Note that given rule (28a) the string *loves Michelle* is the same category (IV) as the intransitive verb *smokes*.
- This is a recurring idea in Montague's papers and other early MG works
- That is, Montague and others don't distinguish syntactically between 'VPs' and single intransitive verbs.
- **I believe the main reason for this is that – from their perspective – this is more elegant than having an extra rule converting all IVs to VPs…**

(31) **Some Sentences of Mini-English**

a. *Michelle loves Barack.*



b. *Barack smokes and it is not the case that Mitt smokes.*