A Linguistics Look on programming languages

*On the Need of Linguistics Theory of Programming Languages*

By Shikhar Yadav

Roll Number: 1601049

Introduction

On the Need of Linguistics Theory of Programming Languages





“Programming Languages are possible the most interesting linguistics creation of our times. They are extensively used to express algorithms and concepts.”

With both the approaches, the goal remains the same: to allow people and computers to communicate freely. However, it’s not easy as it sounds. First approach wants to teach computers to speak English, not considering that computers are not humans and therefore cannot speak English, or at least cannot speak proper, complex, human English, full of subtleties and ambiguities. Second approach wants humans to learn computer language (at least when talking about machine code - other programming languages are bit more human-friendly) and fails to recognize that humans are not computers and often have problems dealing with things that look simple from the computer's point of view.





Background on Programming Languages





Main Reason

WHY I CHOSE THIS TOPIC, WHY IT FASCINATED ME



ANALYSIS

WHAT MY TAKE ON IT



Double Articulation characteristics:

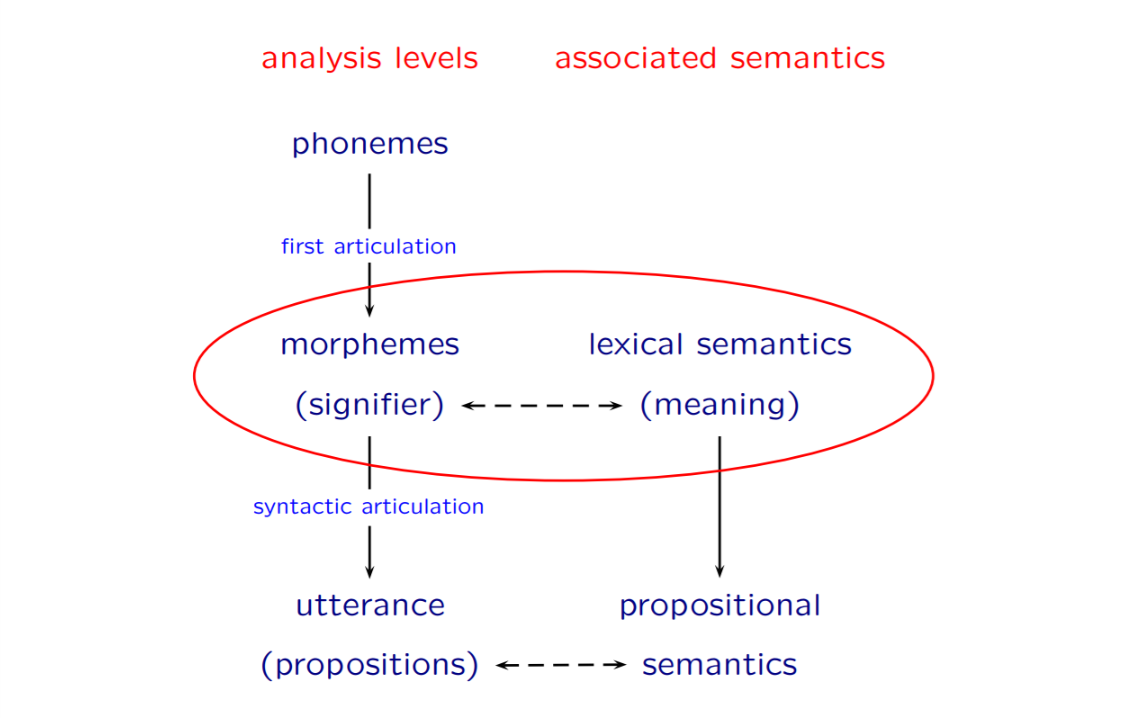
* Natural languages are composed of 3 levels of discrete units:

1. phonemes: equivalence classes of meaningless distinctive sounds,

2. morphemes: smallest meaningful units, and

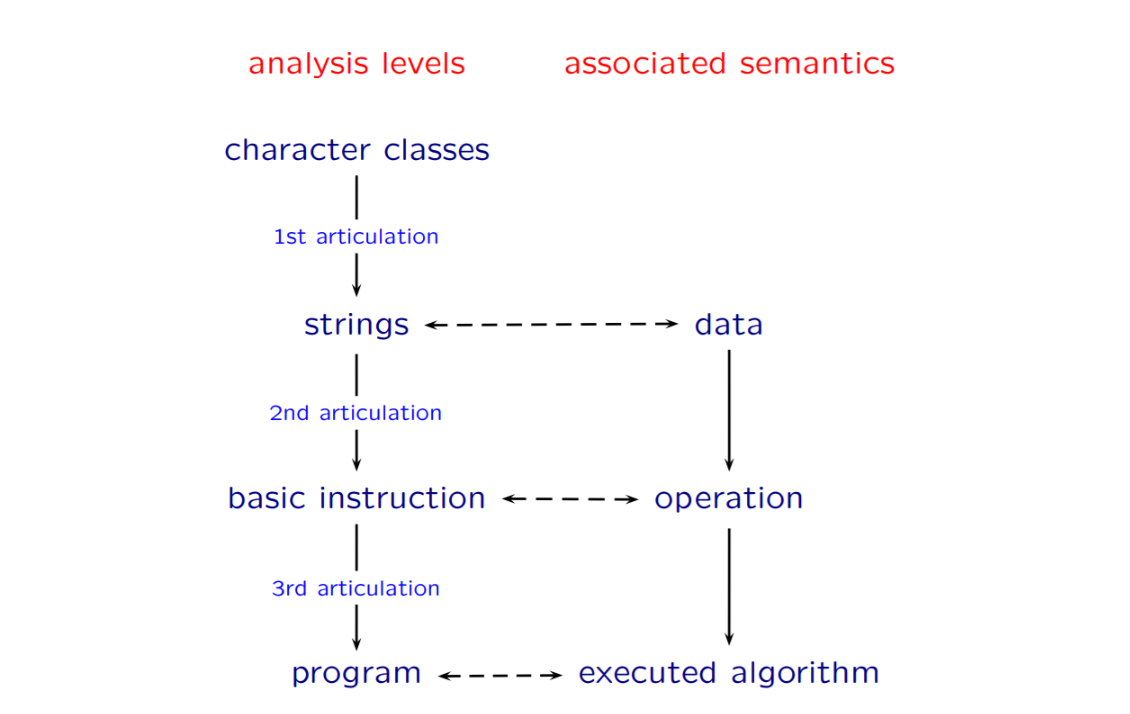
3. propositions: syntactically combined morphemes expressing potentially true/false utterances.

* It is connected by two levels of combination rules.
* It has associated semantics are potentially ambiguous at each level.



* Interesting enough, double articulation applies to Programming Languages. They are composed of discrete units:

1. Alphanumeric characters play the role of phonemes: meaningless distinct basic units.
2. Strings (either keywords or variables/function names...) play the role of morphemes: smallest meaningful units.
3. Basic instructions play the role of syntactically combined strings expressing a potentially successful/not (or true/false) “utterance”.
4. It can be taken a step further; well combined basic instruction can form a ‘program’. Thus, concluding that Programming Languages could have a triple articulation.



On the basis of Semantic and Syntactic

'Semantic role' of a phrase in a sentence means its relationship to the overall meaning of the sentence. For example, a phrase may be an 'action', i.e. what is going on in sentence. In following example 'Shikhar' is an agent while 'walks' is an action.

There are some intuitively felt common roles in natural language sentence and in method call statement in a programming language. First of all, it is the concept of 'action'. In natural languages this role is performed by a verb (a predicate). In programming languages, same role is taken care of by method name.

“The grammar of the Natural Language can be taken as the syntax of the Programming Language.”

English: Shikhar walks.

C: walk (shikhar);

C++/Java/Python: shikhar.walk();

Conjunction

In natural languages, phrases of same kind can be grouped using conjunctions.

English: Shikhar and Ashutosh hit Amit.

Shikhar hits and injures Amit.

Shikhar hits Ashutosh and Amit.

There is no equivalent of this in programming languages, however, every programmer, in case he has learned to program by himself, had probably attempted to use naturally looking at some point:

C: if (a > 0 && a < 10) ...

This way Programming Language is quite similar to Natural Languages.

About Ambiguity:

In Natural Language, different words can have same meaning. Also, same word can have different meaning. For example:

* Mouse can mean the animal, and the device used in computer to control the cursor.

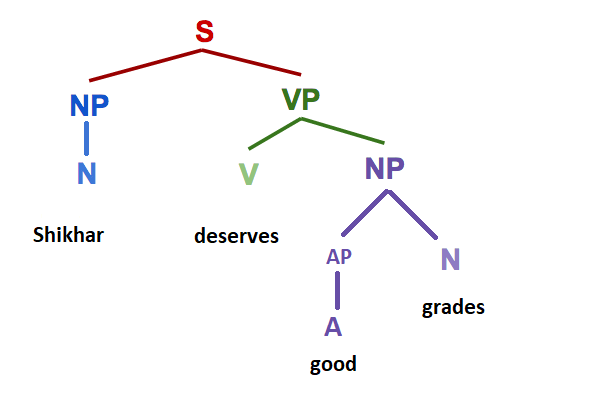
Similar in Programming Language, at morphemes level, it can be taken as different variables can have same data. At sentence level, the same basic instruction can cause different result. And at higher level, same program can lead to different result based on different inputs.

On the contrary, it can also mean that a program can be implemented with different algorithms leading to same result on same input data.

Conclusion

While Programming Language and Natural Language looks quite different on surface, they have a lot of similarities.

Here’s an example of a simple linguistic tree structure for the sentence “Shikhar deserves good grades”.



HTML could represent the following sentence as:

<S>

<NP><N> Shikhar </N></NP>

<VP>

<V> deserves </V>

<NP>

<AP><A> good </A></AP>

<N> grades </N>

</NP>

</VP>

</S>

All of these formats represent a different way of looking at the exact same structure. Structured representations like this are common in all of the subfields of linguistics: the example here uses syntax to show the relationships between words, but hierarchal structures are also used in phonology to show syllable structure and feature geometry, and in semantics and morphology to show the relationships between smaller pieces of meaning.

Realizing the underlying similarities between their two fields can give both linguists and programmers a head start in learning about each other. Thus, applying analogy with Natural Languages to Programming Languages may yield quite an interesting result.

Thank YOU!