BioBIKE Language Syntax
General Consideration of Syntax

A. English syntax as a model for the syntax of computer languages

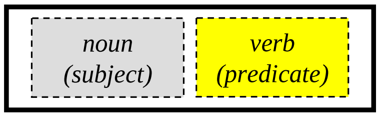
You might think that people who seem to know a computer language either possess some special knowledge or are endowed with some magical ability to sense what's right.

You were probably able to extract some meaning from that complicated sentence, and if so, you have all the basic tools necessary to understand a computer language. You did it, no doubt, without thinking, so let's try it again, this time WITH thinking. So wipe out your knowledge of English.

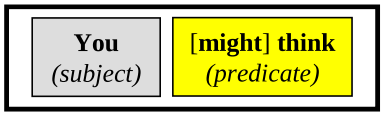
NOW how do you understand the sentence?

Well, of course you'd need an English dictionary. You look up *You* and get the basic meaning and the fact that it is a noun. You could look up the rest the words as well, one by one, but that clearly isn't enough. You also need to know some rules as to how the words work together,... syntax!

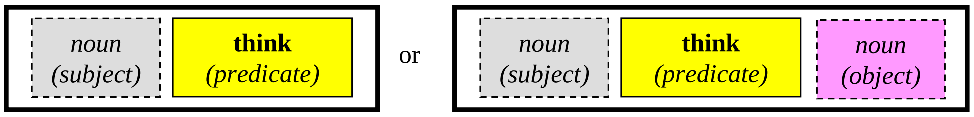
Suppose you learn that English sentences may take a variety of structures, one of the simplest being:



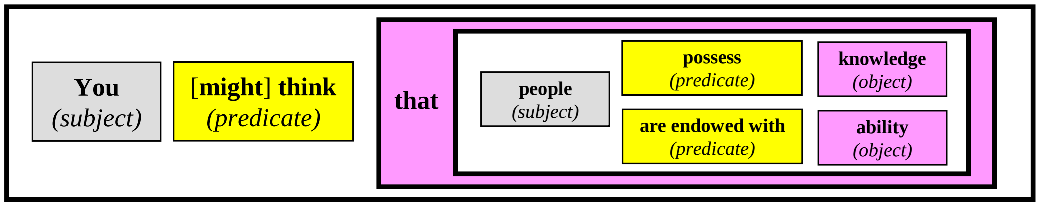
You is a noun, *might* is a verb modifier, and *think* is a verb, says your dictionary. So far:



But what about the other 25 words of the sentence? Looking more closely at the dictionary entry for *think*, you find that it can serve as either a transitive verb (taking an object) or an intransitive verb (taking no object). So either of the following structures is legal:



This is useful, because your dictionary says that the fourth word, *that*, can introduce phrases that can replace nouns, perhaps like so:



Look back on these boxes:

- Boxes with ***dotted boundaries*** represent defined holes
- Boxes with ***solid boundaries*** represent holes filled with an object
- Boxes with ***thick solid boundaries*** represent holes filled with functions that return an object

Complicated! Fortunately, computer languages are much simpler (otherwise computers couldn't understand them), and BioBIKE Language (BBL) is about as simple as you can get and still retain the ability to express everything you need in a language. However, like human languages, computer languages increase their powers of expression by permitting forms to be placed within forms multiple times.

B. Syntactical forms in BioBIKE Language (BBL)

BBL has one form:

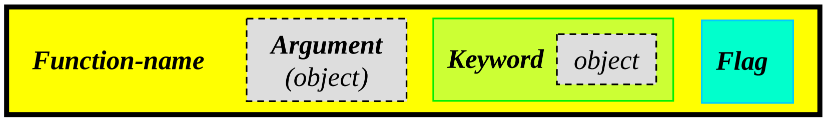
where *object* is a hole that may be filled by a number, a string (a collection of characters), a list (a collection of objects), and other things we needn't concern ourselves with now. For example, if you go to the BioBIKE web listener and type into the command box:

```
      47
   or "banana"
   or {47 "banana"}
```

the listener will be satisfied and respond by repeating back what you typed.

As in English, the ***object*** box may be filled with something that produces an object. In BBL, all legal functions produce objects of one type or another.

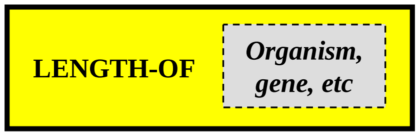
You will be pleased to learn that all BBL functions (unlike English generational structures) have a single general format:



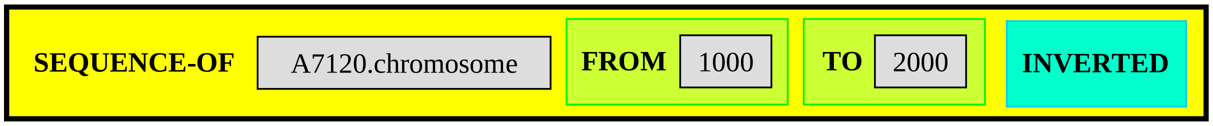
You might think of this as *verb – noun – modifiers*. The function box is represented with a thick solid boundary because it produces an object and therefore can fit into an object hole. The ***argument*** is an object given to the function (just like 90° may be an argument given to the sin function). An optional ***keyword clause*** contains an object, but optional ***flags*** do not. Keyword clauses and flags are represented by colored lines because they cannot be filled with objects. They're part of the function syntax. Depending on the function, there may be more than one argument, more than one keyword clause, or more than one flag.

All functions have at least the verb, the function name. Some consist ***only*** of the function name. For example:

…a verb-only function that closes your session. Some functions have only the function name and a required argument, such as:

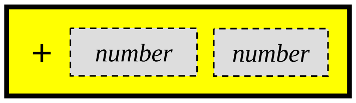


Many functions, perhaps most, allow optional keyword clauses to supply the function with additional values and optional flags to modify the operation of the function. For example:



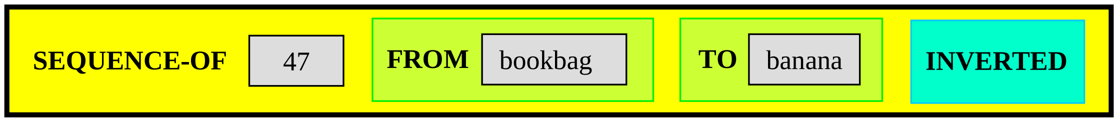
Like all BBL functions, this one produces an object (in this case a DNA sequence) that logically fits into an object hole. When the function is evaluated, the DNA sequence replaces the function, and you're left with just that sequence, an object, which is a legal BBL form.

It's important to see that ***all*** BBL functions have this general verb-first form, even functions you might expect to behave otherwise. For example, 1 + 1 is fit into the same form: Verb first, then arguments:



It may seem strange to write addition in this way, but at the cost of promising the computer that all functions will be rendered verb first, we gain a freedom of expression unmatched in other languages. If the verb always comes first, then whatever comes first must be a verb, and we can ***invent*** new verbs with the confidence that they will be recognized as such.

The addition example illustrates that the syntax of a function may go beyond the mere specification of object holes: it can also impose a restriction as to the type of an object, e.g. a number. Here's another example:



BBL will reject this sentence because SEQUENCE-OF demands that its argument produces a sequence and that FROM and TO be followed by numbers.

It's clearly important to find out what are the syntactical requirements of any function you want to use. How do you do this? In fact, how do you know the function exists that does what you want? The first strategy is to click on the HELP box at the bottom of the BioBIKE web-listener, and then click on Available Functions.[*] For example, if you do this, and then click on Sequence Functions, and finally on SEQUENCE-OF, you'll see a brief description of the syntax, followed by much documentation and many examples. The syntax, given as:

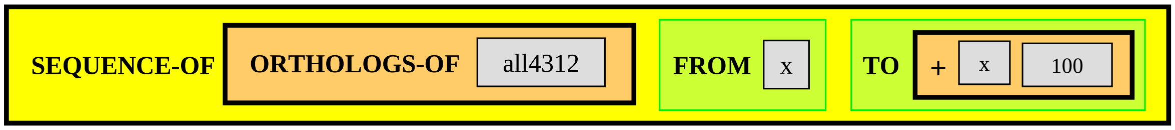
(`SEQUENCE-OF` *entity* [`DNA`] [`PROTEIN`] [`INVERT`] [`TRUNCATE`] [`WRAP`] [`FROM` *number*]
[`FROM-END` *number*] [`TO` *number*] [`TO-END` *number*] [`LENGTH` *positive-number*])

may be mentally translated into boxes, noting that […] indicates an optional flag or keyword clause.

If all the language could do is perform a necessarily limited number of functions, it would be as limited as English would be if limited to simple subject-predicate-object sentences. But like

[*] Be warned! This list of functions is far from complete! Until it is (soon I hope), there is no exhaustive public list of functions you can use within BioBIKE.

English, BBL allows you to replace any object with a function that produces an object. This makes complex expression possible. For example:



C. Textual representation of BBL forms

With colors and boxes, it is possible to represent BBL forms so that their structure is evident, and someday, BBL forms ***will*** be represented in this way. However, for now, we're stuck with pure text. Removing the color and box boundaries, what remains is:

```
SEQUENCE-OF ORTHOLOGS-OF all4312 FROM x TO + x 100
```

This representation is more difficult for humans to read but poses even worse problems for BBL. How does it know whether `ORTHOLOGS-OF` is intended to be the name of a function or rather the name of a simple variable? BBL solves this problem by including in the textual representation the representation of the boxes. Here's how:

Rules:

1. Place a left parenthesis at the left side of a box with thick boundaries (i.e. functions)
2. Place a right parenthesis at the right side of a box with thick boundaries
3. Erase all the boundaries and colors

Applying these rules yields:

```
(SEQUENCE-OF (ORTHOLOGS-OF all4312) FROM x TO (+ x 100))
```

The language can use the parentheses to recreate the boxes. Humans can too: when you see a left parenthesis, look for the ***corresponding*** right parenthesis, which may or may not be the nearest right parenthesis. An important consequence of BBL syntax and the conversion rules is that a***ll functions are represented textually the same way:***

(*Function-name argument(s) keyword-clause(s) flag(s)* **)**