Introduction to the SFGPL

Eruhitsuji

2025-03-22

Contents

Part I.
Overview and basic grammar of the SFGPL

1. About SFGPL

1.1. Introduction

SFGPL stands for “Simple Functional General Purpose Language” and is a language for formalising natural languages. The language was designed to make sentence structure and meaning easily interpretable and communicable. In particular, long and complex sentences containing conjunctions and relative pronouns are often difficult to interpret. The language was created by me as a hobby and has not been rigorously tested, so there may be flaws.

The project then makes the materials and programmes available on GitHub:https://github.com/Eruhitsuji/SFGPL.

1.2. Background and purpose of creating the SFGPL

In the grammars of many natural languages, there are many exceptions and many cases that annoy the learner. To solve this problem, artificial languages have been proposed for a universal language, but like many natural languages, they have ambiguous meanings and are open to multiple interpretations. In particular, long and complex sentences containing conjunctions and relative pronouns are often difficult to interpret. To solve these problems, the SFGPL is an artificial language created with the aim of making languages formally and logically understandable.

1.3. SFGPL Features

SFGPL is a functional language and the types of arguments taken by functions are strictly defined. In SFGPL, functions are assigned to each sentence structure, so that grammatical roles such as subject, predicate, object, and complement are easy to understand. In addition, complex sentences can be created by combining sentence structures.

1.4. Basic grammar of the SFGPL

- Only function words and a few words exist in the SFGPL and have a strictly defined meaning. Other words are borrowed from other languages.
- Function words are followed by a number of arguments, the meaning of which is determined by the arguments.
- In principle, each argument corresponds to a word or an object, but if the source word is more than one word, it can be regarded as a single word by connecting it with an underscore.
- Borrowed words are distinguished by placing a single quotation mark before and after them.
- There are no grammatical distinctions between genders, numbers, etc., and there are no articles.
- A semicolon (;) is added at the end of a sentence. However, it can be omitted in the case of a single sentence.

1.4.1. Sentence structure of the SFGPL

The word order of the SFGPL is SVO, but a function word that determines the structure of the sentence is attached to the beginning of the sentence. Also, the sentence structure of the SFGPL is strictly defined by proper words. The following table shows the sentence structures that can be expressed in the SFGPL. The details of how to use them are described in Sentence Pattern.

		word	function	arguments	supplement
1	S V	ta	Noun.do	S,V	
2	S V C	ma	Noun.eq	S,V,C	C is the noun
2	S V C	me	Noun.haveP	S,V,C	C is the modifier
3	S V O	te	Noun.doT	S,V,O	
4	S V O1 O2	ti	Noun.give	S,V,O1,O2	
5	S V O C	tu	Noun.makeN	S,V,O,C	C is the noun
5	S V O C	to	Noun.makeM	S,V,O,C	C is the modifier
-	A has B	mi	Noun.have	A,V,B	
-	A belongs to B	mu	Noun.belong	A,V,B	
-	A is more B than C	mo	Noun.gt	A,V,B,C	

		word	function	arguments	supplement
-	According to C, A V B	moa	Noun.hearSay	A,V,B,C	A(Subject) V(Verb) that B(Content) according to C(Source)

1.5. Pronunciation of SFGPL

There are no pronunciation exceptions in the SFGPL's native words. The International Phonetic Alphabet (IPA) in the table below is an example of pronunciation.

Consonants of the SFGPL are listed in the table below.

Spell	IPA
p	/p/
b	/b/
f	/f/
m	/m/
t	/t/
d	/d/
s	/s/
n	/n/
l	/l/
k	/k/
g	/g/
j	/j/
w	/w/

On the other hand, the vowels in the SFGPL are as shown in the table below. SFGPL unique words do not have double vowels, except in a few words.

Spell	IPA
a	/a/
e	/e/
i	/i/
u	/u/
o	/o/
oa	/oa/

Borrowed words are read with the pronunciation specific to the borrowed words.

1.6. SFGPL Words

The SFGPL word is mainly divided into SFGPL-specific words and loan words.

The unique words are mainly function words necessary for sentence structure, and basic words for verbs and modifiers. The rest of the words are loan words.

And in the sentence structure of the SFGPL, the position of the part of speech is determined and words must be used according to their part of speech.

1.6.1. Parts of speech in the SFGPL

There are three parts of speech in the SFGPL: Noun, Verb and Modifier. Phrase, Pronoun, BoolList, LangList, LangFunc, LangVar and NumberList exist as subclasses of Noun.

BoolList, LangList, LangFunc, and LangVar are used to create logical statements in addition to general statements. Then, there is a Bool type that represents true/false.

NumberList is mainly used as a numeral. There is also a Number class as a base numeral. This Number class is not normally used by itself.

In addition, there are two special words that modify nouns and verbs: noun determiners (DeterminerN) and verb determiners (DeterminerV).

Each part of speech has its own function words, which change the part of speech and determine its meaning. Other words that implement the basic vocabulary are Word. The SFGPL's specific words are classified according to their parts of speech: verbs are "WordV", modifiers are "WordM".

Nouns are words that describe any concept, such as any object, substance, person or place. Verbs are words that describe any action, action, state, being, etc. Modifiers are words that modify other words. Modifiers are words that modify other words; the SFGPL makes no distinction between adjectives and adverbs.

In the Python library SFGPL, there are classes for each part of speech. LangObj is the basic structure of a word and _BaseList defines the basic structure for a List. In addition, DeterminerN and DeterminerV are words that are only used to add meaning to words and are represented in Python as just static functions.

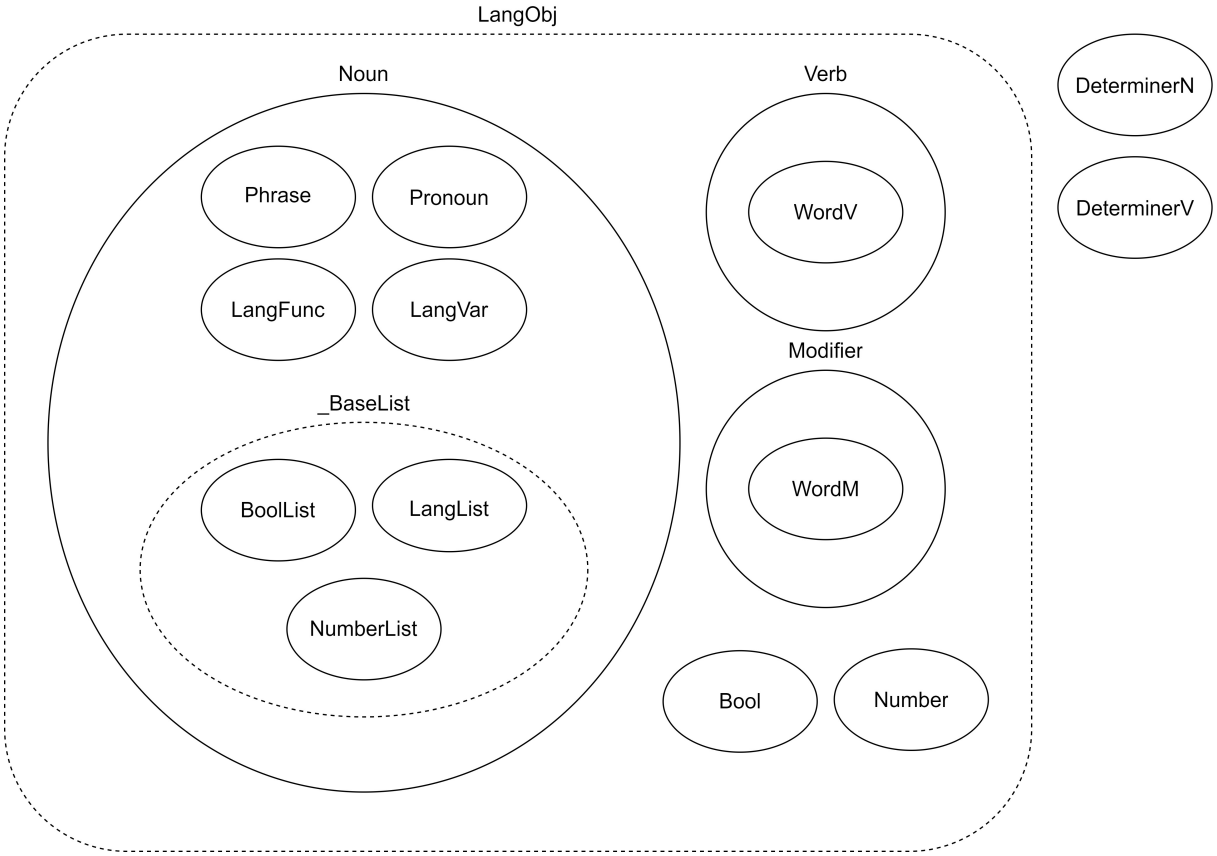

Figure 1: PartOfSpeach

1.6.2. Function words in the SFGPL

Function words determine the role, part of speech, etc. of a sentence. The function, role and meaning of function words are only applicable within arguments.

These function words are one-to-one with Python functions. They also have a fixed number of arguments, and the role of each argument is determined by its location.

For a list of function words and how to use them, see dict.csv.

1.6.3. Borrowed words in the SFGPL

Borrowed words are used for words that do not exist in the SFGPL. It is preferable to borrow words from languages commonly used in the world, such as English, but this should not be a problem as long as the words can be understood by others. However, it is recommended that borrowed words are used in their original form, and if there is a conjugation, it should be done in SFGPL function words.

1.7. SFGPL and programming

SFGPL sentences can be rewritten into Python objects. This project contains a file in which the SFGPL is defined. To use the SFGPL in Python, use SFGPL.py can be used by importing it. Examples of use are samples in the Python files. Also, for detailed instructions on how to run the SFGPL library in Python, see How_to_Use_SFGPL_in_Python.ipynb.

2. Basic Grammar

This chapter explains the basic knowledge and grammar for learning the SFGPL. In particular, it describes the basic sentences in the affirmative form of the present tense.

It is also recommended to read aboutSFGPL as a prerequisite. Furthermore, as a whole this material is based on English, e.g. example sentences, so it is desirable to know a little English.

2.1. About the features of the SFGPL sentence structure

Like English, SFGPL is an SOV-type language in which the role of a word is determined by its position. One of the most important features of the SFGPL is its emphasis on sentence pattern. Each of these sentence patterns defines what arguments (words of what part of speech) and how many to take. The meaning of the sentence is therefore uniquely determined. The function word that determines this sentence type is attached to the beginning of the sentence (clause). The whole sentence (clause) is considered as a noun and can be nested (Compound Sentences).

2.2. Specific examples of SFGPL sentence patterns

2.2.1. Syntax with ta

First, we present an example using `ta`. This `ta` has two arguments, where the first argument is the subject of the sentence and the second argument is the verb of the sentence. In other words, `ta` can produce sentences equivalent to the English first sentence type SV.

For example, to express “I run.” in SFGPL, use the following.

```
ta ga sa 'run'
```

In this case, `ta` is the word to be added when the sentence type is “SV”.

The `ga` denotes the first person pronoun “I”.

And `sa 'run'` denotes the verb “run”. This `sa 'run'` consists of two words. In loan words such as these, a part-of-speech indicator (in this case, `sa`) is attached to the word. There are three words that represent such parts of speech.

	SFGPL
Noun	fa
Verb	sa
Modifier	la

2.2.2. Syntax with ma

Next, we present an example using `ma`. This `ma` has three arguments: the first is the subject of the sentence, the second is the verb of the sentence and the third is the complement of the subject. Also, the complement of the third argument must be a noun. In other words, `ma` can produce sentences equivalent to the English second sentence type SVC.

As an example, to express “I am a student.” in the SFGPL, use the following.

```
ma ga so fa 'student'
```

In this case, `ma` is the word to be added when the sentence type is “SVC”.

The `ga` denotes the first person pronoun “I”.

Next, `so` is a word indicating that the verb is nonsense. In `so`, the meaning changes depending on the location. In the example sentence, the meaning is equivalent to that of the English verb “be”.

And `fa 'student'` denotes the noun “student”. In this case, the article that exists in English and other languages does not exist in the SFGPL, so there is no need to add any article.

2.2.3. Syntax with me

Next, we present an example using `me`. This `me` has three arguments: the first is the subject of the sentence, the second is the verb of the sentence and the third is the complement of the subject. Also, the complement of the third argument must be a modifier. In other words, `me` can produce sentences equivalent to the English second sentence type SVC.

As an example, to express “I am happy.” in the SFGPL, do the following.

```
1 me ga so la 'happy'
```

In this case, `me` is the word to be added when the sentence type is “SVC”.

The `ga` denotes the first person pronoun “I”.

Next, `so` is a word indicating that the verb is nonsense. In `so`, the meaning changes depending on the location. In the example sentence, the meaning is equivalent to that of the English verb “be”.

And `la 'happy'` denotes the modifier “happy”.

2.2.4. Syntax with te

We then present an example using `te`. This `te` has three arguments, the first representing the subject of the sentence, the second the verb of the sentence and the third the object. In other words, `te` can produce sentences equivalent to SVO, the third sentence type in English.

For example, to express “I open the door.” in SFGPL, use the following.

```
1 te ga sa 'open' fa 'door'
```

In this case, `te` is the word to be added when the sentence type is “SVO”.

The `ga` denotes the first person pronoun “I”.

Next, `sa 'open'` denotes the verb “open”.

Next, `fa 'door'` denotes the noun “door”.

2.2.5. Syntax with ti

Next, we present an example using `ti`. This `ti` has four arguments, the first representing the subject of the sentence, the second the verb of the sentence, the third the indirect object and the fourth the

direct object. In other words, `ti` can produce sentences equivalent to SVOO, the fourth sentence type in English.

For example, to express “I give you a box.” in SFGPL, you can do the following.

```
1 ti ga so ge fa 'box'
```

In this case, `ti` is a word that is added when the sentence type is “SVOO”.

The `ga` denotes the first person pronoun “I”.

Next, `so` is a word indicating that the verb is nonsense. In `so`, the meaning changes depending on the location. In the example sentence, the meaning is equivalent to the English word give.

And `ge` denotes the second person pronoun “you”.

Furthermore, `fa 'box'` denotes the noun “box”.

2.2.6. Syntax with tu

Next, we present an example using `tu`. This `tu` has four arguments: the first is the subject of the sentence, the second the verb of the sentence, the third the object and the fourth the complement of the object. The complement of the fourth argument must be a noun. In other words, `tu` can produce sentences equivalent to the English fourth sentence type SVOC.

For example, to express “I make you a teacher.” in SFGPL, use the following.

```
1 tu ga so ge fa 'teacher'
```

In this case, `tu` is the word to be added when the sentence type is “SVOC”.

The `ga` denotes the first person pronoun “I”.

Next, `so` is a word indicating that the verb is nonsense. In ‘`so`, the meaning changes depending on the location. In this case, in the example sentence, the meaning is equivalent to the English causative verb make.

And `ge` denotes the second person pronoun “you”.

Furthermore, `fa 'teacher'` represents the noun “teacher”.

2.2.7. Syntax with to

Next, we present an example using `to`. This `to` has four arguments: the first is the subject of the sentence, the second the verb of the sentence, the third the object and the fourth the complement

of the object. The complement of the fourth argument must be a modifier. In other words, `to` can produce sentences equivalent to the English fourth sentence type SVOC.

As an example, to express “I make you happy.” in the SFGPL, use the following.

```
1 to ga so ge la 'happy'
```

In this case, `to` is the word to be added when the sentence type is “SVOC”.

The `ga` denotes the first person pronoun “I”.

Next, `so` is a word indicating that the verb is nonsense. In ‘`so`, the meaning changes depending on the location. In this case, in the example sentence, the meaning is equivalent to the English causative verb make.

And `ge` denotes the second person pronoun “you”.

Furthermore, `la 'happy'` represents the modifier “happy”.

2.2.8. Syntax with mi

Next, we present an example using `mi`. This `mi` has three arguments, the first representing the subject of the sentence (the owner), the second the verb of the sentence and the third the object (the possession). Therefore, `mi` can represent the sentence “S has O”.

As an example, to express “I have a box.” in SFGPL, use the following.

```
1 mi ga so fa 'box'
```

In this case, `mi` is the word used to make a sentence expressing possession.

The `ga` is the first person pronoun “I”.

Next, `so` is a word that indicates that the verb is meaningless. In `so`, the meaning changes depending on the location. In the example sentence, the meaning is equivalent to the English word have.

Furthermore, `fa 'box'` denotes the noun “box”.

2.2.9. Syntax with mu

Next, we present an example using `mu`. This `mu` has three arguments, the first representing the subject of the sentence (the person or thing to which it belongs), the second the verb of the sentence and the third the object (place of affiliation). Therefore, `mu` can represent the sentence “S belongs to O”.

As an example, to express “I belong to a school.” in SFGPL, use the following.

```
mu ga so fa 'school'
```

In this case, `mu` is a word that is added to make a statement of belonging.

The `ga` denotes the first person pronoun “I”.

Next, `so` is a word that indicates that the verb is meaningless. In `so`, the meaning changes depending on the location. In the example sentence, the meaning is equivalent to “belong to” in English.

Furthermore, `fa 'school'` denotes the noun “school”.

2.2.10. Other syntaxes

Other syntaxes are indicated by sentence pattern.

2.3. Modification methods

2.3.1. How to modify nouns

There are two main ways of modifying nouns: with modifiers and with nouns.

How to modify a noun with a modifier For example, to express that a certain box is big, in English we say “The box is big.”. Similarly, the SFGPL uses `me` to express an SVC as follows:.

```
me fa 'box' so wan
```

In this case, `wan` means big.

How to modify a noun with a noun for a noun This method is used in situations where the English preposition “of” or the Japanese particle “の” is used. However, the SFGPL does not allow for concise notation, and even in such cases it is qualified by sentences like English relative pronouns (compound sentences).

For example, “My box is big.” can be expressed by the following sentence.

```
me mi ga so san fa 'box' so wan
```

In this sentence, the sentence `mi ga so san fa 'box'` is nested in the subject of the main sentence. This `mi ga so san fa 'box'` means “I have a box. In addition, the use of `san` can be used to emphasise words that are particularly important in the sentence. Thus, in this sentence, `san fa 'box'` is used to emphasise the word”box”.

Overall, the literal translation means “[I have a **box**] is big.”, which is equivalent to “My box is big.”.

2.3.2. How to modify verbs

Simple verb modification methods Verbs can be modified using `na`. In `na`, the first argument is the verb and the second argument is the modifier.

For example, to express “I quickly run.”.

```
ta ga na sa 'run' la 'quickly'
```

In this case, `la 'quickly'` means “quickly”. In `na sa 'run'la 'quickly'`, the verb “run” is modified by the modifier “quickly”, meaning “quickly run”.

How to convert a noun phrase into a modifier and modify a verb The SFGPL allows you to convert a noun phrase into a modifier, which then modifies a verb. This is similar to adverbialisation using prepositions in English.

First, the SFGPL has words that can be converted between parts of speech, which in this case uses `li` to convert a noun to a modifier. In this usage, a noun determiner is used in parallel with `li` to limit the meaning of the noun phrase.

For example, to express “I go to Tokyo.” in SFGPL.

```
ta ga na sa 'go' li pun fa 'Tokyo'
```

Firstly, `sa 'go'` expresses the English word “go”, and the destination is expressed by modifying the verb. In particular, the four words `li pun fa 'Tokyo'` represent “to Tokyo”. The three words `li` are noun-to-modifier words, and `pun` is a nominal determiner of place. Combining these two words with `fa 'Tokyo'` (Tokyo) to express the meaning “to Tokyo”.

2.3.3. How to modify modifiers

The modifier modification is expressed using ‘`ka`. In this `ka`, the modifier of the second argument modifies the modifier of the first argument.

For example, to express “Your box is a little big.” in the SFGPL, use the following.

```
me mi ge so san fa 'box' so ka wan la 'little'
```

In this case, `la 'little'` (= “a little”) modifies `wan` (= “big”), so that `ka wan la 'little'` means “a little big”.

2.4. Wordbook

English	SFGPL
I	ga
run	sa ‘run’
student	fa ‘student’
happy	la ‘happy’
open	sa ‘open’
door	fa ‘door’
you	ge
box	fa ‘box’
teacher	fa ‘teacher’
school	fa ‘school’
big	wan
quickly	la ‘quickly’
go	sa ‘go’
Tokyo	fa ‘Tokyo’
a little	la ‘little’

Part II.
Syntax of the SFGPL

3. Sentence Pattern

3.1. List of SFGPL sentence patterns

In the SFGPL, a function word that determines the sentence type is always attached to the beginning of a sentence in order to form a sentence. In the SFGPL, there are sentence types as shown in the table below, and the sentences themselves are composed by the combination of these sentence types. In

addition, modification of words is also performed.

		word	function	arguments	supplement
1	S V	ta	Noun.do	S,V	
2	S V C	ma	Noun.eq	S,V,C	C is the noun
2	S V C	me	Noun.haveP	S,V,C	C is the modifier
3	S V O	te	Noun.doT	S,V,O	
4	S V O1 O2	ti	Noun.give	S,V,O1,O2	
5	S V O C	tu	Noun.makeN	S,V,O,C	C is the noun
5	S V O C	to	Noun.makeM	S,V,O,C	C is the modifier
-	A has B	mi	Noun.have	A,V,B	
-	A belongs to B	mu	Noun.belong	A,V,B	
-	A is more B than C	mo	Noun.gt	A,V,B,C	
-	According to C, A V B	moa	Noun.hearSay	A,V,B,C	A(Subject) V(Verb) that B(Content) according to C(Source)

3.2. Noun.do (ta)

In Noun.do `ta`, in particular, S is the subject and V is the verb in the same form as the English first sentence form, and the subject is said to perform some action. It can express simple sentences. “I run.” can be expressed in SFGPL as follows.

```
ta ga sa 'run'
```

3.3. Noun.eq (ma)

Noun.eq `ma` corresponds to the English second sentence pattern “S is C”, in which the complement C is a noun. This construction also shows that S and C are equivalent. If V corresponds to a be verb in

English, use `so` as the verb. To express “This is a table.” in SFGPL, it is as follows.

```
ma gu so fa 'table'
```

“You become a teacher.” can be expressed in SFGPL as follows.

```
ma ge sa 'become' fa 'teacher'
```

3.4. Noun.haveP (me)

Noun.haveP `me` corresponds to the English second sentence pattern “S is C”, in which the complement C can be used as a modifier. In this construction, S is the property or state of C. If V corresponds to a be verb in English, use `so` as the verb. To express “The table is red.” in SFGPL, it is as follows.

```
me fa 'table' so la 'red'
```

“You look sad.” can be expressed in SFGPL as follows.

```
me ge sa 'look' la 'sad'
```

3.5. Noun.doT (te)

Noun.doT `te`, in particular, corresponds to the third sentence pattern in English, where S is the subject, V is the verb, and O is the object. “I study English.” can be expressed in SFGPL as follows.

```
te ga sa 'study' fa 'English'
```

3.6. Noun.give (ti)

In Noun.give `ti`, in particular, it corresponds to the English fourth sentence pattern, where S is the subject, V is the verb, and O1 and O2 are the objects. In particular, this construction means “S gives O1 O2”. If V corresponds to “give” in English, use `so` as the verb. “I give you a table.” can be expressed in SFGPL as follows.

```
ti ga so ge fa 'table'
```

3.7. Noun.makeN (tu) and Noun.makeM (to)

Noun.makeN `tu` and Noun.makeM `to`, in particular, correspond to the English fifth sentence pattern, where S is the subject, V is the verb, O is the object and C is the complement. Noun.makeN is used

when C is a noun and Noun.makeM when C is a modifier. In this construction, it means “S makes O C”. If V corresponds to “make” in English, use `so` as the verb.

“I make you a teacher.” can be expressed in SFGPL as follows.

```
tu ga so ge fa 'teacher'
```

“I make you sad.” can be expressed in SFGPL as follows.

```
to ga so ge la 'sad'
```

3.8. Noun.have (mi)

Noun.have `mi` means “A owns B”. If V corresponds to “have” in English, use `so` as the verb. “I have a table.” can be expressed in SFGPL as follows.

```
mi ga so fa 'table'
```

3.9. Noun.belong (mu)

Noun.belong `mu` means “A belongs to B”. If V corresponds to “belong to” in English, use `so` as the verb. “I belong to a school.” can be expressed in SFGPL as follows.

```
mu ga so fa 'school'
```

3.10. Noun.gt (mo)

Noun.gt `mo` means “A is more B than C”. In this case, A and B are the nouns being compared and C is a modifier. If V corresponds to a be verb in English, use `so` as the verb. “The bed is bigger than yours.” can be expressed in the SFGPL as follows.

```
mo fa 'bed' so wan sen ge
```

3.11. Noun.hearSay (moa)

Noun.hearSay `moa` means “A(Subject) V(Verb) that B(Content) according to C(Source)”. In this case, A is the person or thing receiving the information, V is the verb, B is the content of the information and C is the source person or thing. If V corresponds to a verbs related to hearsay, such as hear, say and see in English, use `so` as the verb. “According to the book, I saw that Japan is located in East Asia.” can be expressed in the SFGPL as.

```
di moa ga so ta fa 'Japan' na ne sa 'locate' li fun pun me fa 'Asia' so
    la 'east' fa 'book'
```

3.12. How to modify nouns using sentence structures

SFGPL uses these sentence structures to modify nouns. When a sentence is generated, the entire sentence becomes a noun, which can be embedded in another sentence.

"Your table is red." can be expressed in SFGPL as follows.

```
me mi ge so fa 'table' so la 'red'
```

Thus, `mi ge so fa 'table'`, which is "You have table", becomes the subject, and it can be explained that the table is red `la 'red'`. The equivalent "You have red table." can be expressed as follows.

```
mi ge so me fa 'table' so la 'red'
```

3.12.1. Stressed Form

Emphasis `san` can also be used, especially when you want to emphasize a word other than the subject in a sentence. To stress the word "table" in "Your table is red.".

```
me mi ge so san fa 'table' so la 'red'
```

3.13. Wordbook

English	SFGPL
I	ga
run	sa 'run'
this	gu
table	fa 'table'
red	la 'red'
you	ge
become	sa 'become'
teacher	fa 'teacher'

English	SFGPL
look	sa ‘look’
sad	la ‘sad’
study	sa ‘study’
English	fa ‘English’
school	fa ‘school’
bed	fa ‘bed’
big	wan
yours	sen ge
book	fa ‘book’
Japan	fa ‘Japan’
in East Asia	li fun pun me fa ‘Asia’ so la ‘east’

4. Negative sentences and Negative expressions

4.1. Negative statement

Use `pa` to create a normal negative sentence. This word is attached to a sentence to make a negative sentence. “I have a table.” is `mi ga so fa 'table'` under the SFGPL. To negate this whole sentence and make it mean “I don’t have a table.” in the negative sentence, the SFGPL can be expressed as follows.

```
1 pa mi ga so fa 'table'
```

4.2. Negation of verbs

In SFGPL, besides negating whole sentences, it is also possible to negate verbs alone. Negating an entire sentence and negating only the verb may have different meanings. In satellite-framed languages such as English, in particular, the interpretation of their meanings may differ.

For example, in “I don’t have a table.”, the negation of the whole sentence and the negation of the verb alone are almost synonymous.

All Sentence	pa te ga sa ‘make’ fa ‘table’
Only Verb	te ga pa sa ‘make’ fa ‘table’

In “I didn’t run to my school.”, the negation of the whole sentence and the negation of the verb alone have different meanings.

All Sentence	di pa ta ga na sa ‘run’ li pun mu ga so san fa ‘school’
Only Verb	di ta ga na pa sa ‘run’ li pun mu ga so san fa ‘school’

In the case of the negation of the whole sentence, all events other than “I ran to my school.” are represented. In other words, it also implies “I walked to my school”, “I didn’t go to my school”, etc.

However, in the case of verb-only negation, it implies an action other than “running” in the event “I went to my school.”. In other words, it implies other means, such as “I walked to my school.”, but not “I didn’t go to my school.”.

On the other hand, in verb-framed languages such as Japanese, such differences in meaning tend to disappear because they are expressed by compound verbs such as “走って行く”. In this case, the compound verb “走って行く” contains both the method of action “走る (to run)” and the result of the action “行く (to go)”, unlike in English.

4.3. Negative forms of modifiers

In a modifier, the suffix `ke` can be used to indicate a synonym of the modifier.

For example, the synonym “small” for `wan`, which means “big”, can be expressed by adding `ke wan`.

“My table is small.” can be expressed in SFGPL as follows.

```
me mi ga so san fa 'table' so ke wan
```

4.4. Wordbook

English	SFGPL
I	ga
table	fa ‘table’

English	SFGPL
big	wan

5. Interrogative Sentence

5.1. Yes–no question

Use `da` to create interrogative sentences. When this word is added to a sentence, it becomes a interrogative sentence. “You have a table.” is `mi ge so fa 'table'` under the SFGPL. To make it mean “Do you have a table?”, it can be expressed as follows in the SFGPL.

```
da mi ge so fa 'table'
```

Such a question reply is expressed by using Bool.B2N`pis` to indicate whether the proposition is true or false, as follows.

```
pis mi ga so pen gi pos
pis mi ga so pen gi pas
```

5.2. wh-questions

In the case of interrogative sentences containing interrogatives, the indefinite is expressed by replacing the indefinite with an interrogative. Interrogatives are represented by a combination of the interrogative pronoun `wa` and noun determiner.

“Who has a table?” is expressed as follows.

```
da mi ben wa so fa 'table'
```

“What do you have?” is expressed as follows.

```
da mi ge so pen wa
```

5.3. Wordbook

English	SFGPL
you	ge

English	SFGPL
table	fa ‘table’
who	ben wa
what	pen wa

6. Imperative Sentence

Use de to create imperative sentences. This word is added to a sentence to make it an imperative sentence. “You buy a table.” is `te ge sa 'buy'fa 'table'` under the SFGPL. To make it mean “Buy a table, you!”, it can be expressed as follows in the SFGPL.

```
de te ge sa 'buy' fa 'table'
```

6.1. Wordbook

English	SFGPL
you	ge
buy	sa ‘buy’
table	fa ‘table’

7. Compound sentences

The SFGPL allows you to create sentences that combine several within a single sentence.

7.1. Parallel clauses

A conjunction is used to connect two or more sentences in parallel.

In the SFGPL, “I went to Tokyo and I was shopping there.” can be expressed as follows.

```
ba di ta ga na sa 'go' li pun fa 'Tokyo' di ta ga na ni sa 'shop' li
   pun gu
```

And while English-like tense agreement requires clause-by-clause utilisation in this way, the SFGPL allows the basic tense to be utilised throughout the sentence.

```
di ba ta ga na sa 'go' li pun fa 'Tokyo' ta ga na ni sa 'shop' li pun
    gu
```

7.2. Dependent clauses

A subordinate modification of a noun in the main clause can be achieved by inserting a sentence describing the noun instead of the noun. In addition, the SFGPL generally uses subordinate clauses to modify nouns.

7.2.1. General subordinate clauses

In the SFGPL, “My bag is big.” can be expressed as follows. In this case, “My bag” is expressed as “I have a bag”. The noun is then marked with `san` because “bag” is the noun being modified.

```
me mi ga so san fa 'bag' so wan
```

The meaning of “I have a bag is big.” is almost the same as “I have a bag is big. In this case, the”bag” in “a bag is big” is the subject of the subordinate clause, so `san` need not be added.

```
mi ga so me fa 'bag' so wan
```

Then, to express “I give you the desk I built.”, do the following.

```
ti ga so ge di te ga sa 'build' san fa 'desk'
```

The tense of only the subordinate clause can be changed in this way.

7.2.2. Adverbial clauses

Adverbial clauses can be used to modify predicates and whole sentences. In the SFGPL, “I ate sushi, when I went to Tokyo.” can be expressed as follows.

```
di te ga na sa 'eat' li ta ga na sa 'go' li pun fa 'Tokyo' fa 'sushi'
```

Or, to express “I went grocery shopping while my kids were sleeping.” in the SFGPL.

```
di ta ga na sa 'go' ba li ma fi ni sa 'shop' so fa 'grocery' li ta mi
    ga so san don fa 'kid' ni sa 'sleep'
```

7.3. Modification of nouns by nouns

When Y modifies X in a noun X and Y, it is expressed as “Y の X” in Japanese and “Y X” or “X of Y” in English, but the SFGPL uses three main types of usage. In the SFGPL, as mentioned earlier, modifications are often made in subordinate clauses, and the case of nouns modifying nouns with nouns is no exception. Therefore, nouns can be modified in different ways: `ma`, `mi` and `mu`.

7.3.1. Noun.eq (ma)

First, `ma` is mainly used when the modifier and the moderated are equivalent. For example, to express “This pen is big.” in SFGPL as follows.

```
me ma gu so san fa 'pen' so wan
```

In this case, “this” and “pen” are equivalent. Therefore, `ma` is used.

7.3.2. Noun.have (mi)

Next, `mi` is mainly used when something has something. To express “My pen is big.” in the SFGPL, use the following.

```
me mi ga so san fa 'pen' so wan
```

7.3.3. Noun.belong (mu)

Also, `mu` is mainly used when something belongs to something. To express “My school is big.” in the SFGPL, use the following.

```
me mu ga so san fa 'school' so wan
```

7.4. Wordbook

English	SFGPL
I	ga
go	sa ‘go’
to Tokyo	li pun fa ‘Tokyo’
shop (Verb)	sa ‘shop’

English	SFGPL
there	pun gu
bag	fa ‘bag’
big	wan
you	ge
build	sa ‘build’
desk	fa ‘desk’
eat	sa ‘eat’
sushi	fa ‘sushi’
grocery	fa ‘grocery’
kid	fa ‘kid’
sleep	sa ‘sleep’
this	gu
pen	fa ‘pen’
school	fa ‘school’

8. Verb Conjugation

The SFGPL has words that modify verbs, such as tense, aspect and auxiliary verbs. These words are mainly attached directly to the verb and modify it, while others modify the whole sentence.

8.1. Verb tenses

Verb tenses exist in the SFGPL as shown in the figure below.

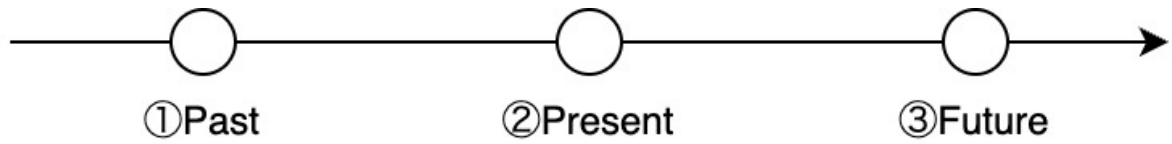

Figure 2: BasingPoint

Thus, there are three tenses in the SFGPL: ① past tense, ② present tense, and ③ future tense. These tenses are fundamental to verb conjugation and serve as reference points for sentence time. Example sentences using the tenses are shown in the following table.

Tense	English	SFGPL
① Past Tense	I lived in Tokyo.	di ta ga na sa ‘live’ li pun fa ‘Tokyo’
② Present Tense	I live in Tokyo.	ta ga na sa ‘live’ li pun fa ‘Tokyo’
③ Future Tense	I will live in Tokyo.	du ta ga na sa ‘live’ li pun fa ‘Tokyo’

In particular, `di` and `du` are attached to the sentence itself.

The present tense in ②, with nothing attached, usually denotes the present. However, it is essentially an indefinite tense and is also used when no particular tense is required.

8.1.1. Extended verb tenses

The verbs described in the previous section are the most basic way of expressing verb tenses. However, in the SFGPL, there are words that are mainly used to combine tenses, depending on the DetermineV class. The extended tense by the DeterminerV class has a lower priority than the base tense by the Phrase class, and the base tense basically represents the tense of the entire sentence. The following table shows the words that represent the extended tense.

Tense	Word
① Past Tense	bak
② Present Tense	bik
③ Future Tense	bok

These tenses can be combined to form compound tenses such as future past tense and past future tense. The following is an example of the future past tense, which expresses the past at a future point in time.

```
1 du ta ga na bak sa 'live' li pun fa 'Tokyo'
```

In summary, the tenses in the SFGPL are as shown in the table below. The column names in the table below indicate the types of the base tense by Phrase, and the row names indicate the types of the extended tense by DeterminerV. In `A/B`, A denotes the base tense and B the extended tense.

	Past Tense	-	Future Tense
-	di/-	-/-	du/-
Past Tense	di/bak	-/bak	du/bak
Present Tense	di/bik	-/bik	du/bik
Future Tense	di/bok	-/bok	du/bok

8.2. Aspect on the time axis of operation

In SFGPL, there are six aspects as shown in the figure below: ① start aspect, ② transitional aspect, ③ completion aspect, ④ continuation aspect, ⑤ finish aspect, and ⑥ progression aspect.

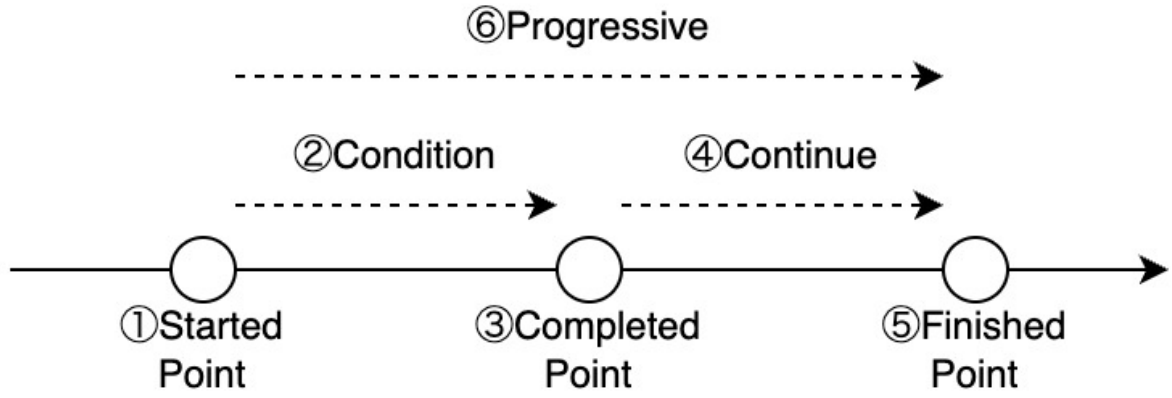

Figure 3: ProgressiveForm

The following table shows example sentences in each aspect for `te ga sa 'wear'fa 'dress'` meaning “I wear dress”.

Aspect	Word	English	SFGPL
① Start Aspect	tak	I begin wear a dress.	te ga tak sa ‘wear’ fa ‘dress’
② Transitional Aspect	tek	I am (in the process of) wearing a dress.	te ga tek sa ‘wear’ fa ‘dress’

Aspect	Word	English	SFGPL
③ Completion Aspect	tik	I wear a dress. (I just finished wearing it.)	te ga tik sa ‘wear’ fa ‘dress’
④ Continuation Aspect	tuk	I am wearing a dress. (The state in which it is worn.)	te ga tuk sa ‘wear’ fa ‘dress’
⑤ Finish Aspect	tok	I finish wear a dress. (I stopped wearing it.)	te ga tok sa ‘wear’ fa ‘dress’
⑥ Progression Aspect	ni	I am wearing a dress.	te ga ni sa ‘wear’ fa ‘dress’

The ① start aspect, ③ completion aspect and ⑤ finish aspect represent only one point in time for a certain action.

The ② transitional aspect, ④ continuation aspect and ⑥ progression aspect represent a period of time for a certain action. ⑥ Progression aspect represents an indistinct period that includes ② transitional aspect and ④ continuation aspect. For some verbs, the interval between aspect with each may be momentary and almost indistinguishable.

These aspects can be in the past or future tense in addition to the present tense. “I begin wear a dress.” in the past and future tenses is as follows.

```
di te ga tak sa 'wear' fa 'dress'
du te ga tak sa 'wear' fa 'dress'
```

As a rule, these aspects by themselves express an action focused on a certain point in time. In particular, in order to emphasise cases where the action has continued past the point in time, the perfect tense is used in addition to these aspects. The progressive form plus the perfect form to express “I have been wearing a dress.”.

```
te ga nu ni sa 'wear' fa 'dress'
```

8.2.1. General progressive form

In SFGPL, we can make a simple progressive form as in ⑥ without considering the aspects ① to ⑤ in the previous section. The SFGPL can be expressed in the progressive form meaning “I am wearing the dress.” as follows.

```
te ga ni sa 'wear' fa 'dress'
```

Progressive forms ni are attached to verbs. They can be past or future tense as well as present tense. “I am wearing the dress.” in the past and future tenses is as follows.

```
di te ga ni sa 'wear' fa 'dress'
du te ga ni sa 'wear' fa 'dress'
```

8.3. Perfect tense

In the SFGPL, there is a perfect tense equivalent to English, as shown in the figure below.

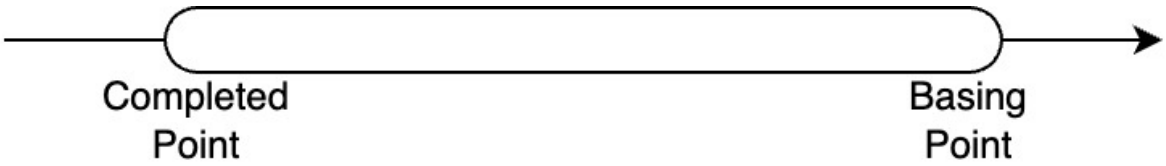

Figure 4: PerfectForm

This perfect tense is used to indicate that something that has happened in the past is continuing. Examples of the perfect tense for the three tenses are as follows.

Tense	English	SFGPL
① Past Perfect Tense	I had lived in Tokyo.	di ta ga nu na sa ‘live’ li pun fa ‘Tokyo’
② Present Perfect Tense	I have lived in Tokyo.	ta ga nu na sa ‘live’ li pun fa ‘Tokyo’
③ Future Perfect Tense	I will have lived in Tokyo.	du ta ga nu na sa ‘live’ li pun fa ‘Tokyo’

In nu, the perfective form is attached to and modifies the verb itself.

8.4. Summary of time expressions in the SFGPL

The following table exists with regard to the time expressions of the SFGPL.

Base tense	Extended tense	Perfect tense	Progressive form
-	-	-	-
di	bak	nu	tak
du	bik		tek
	bok		tik
			tuk
			tok
			ni

このように，SFGPL では 3×4×2×7=168 通りの時間表現が存在し，あらゆる場面に対して表現することが可能である．

8.5. Passive voice

SFGPL can express the passive voice with the meaning “The dress is worn.”.

```
ta fa 'dress' ne sa 'wear'
```

The ne, which indicates the passive form, is attached to the verb. These can be in the past or future tense as well as the present tense. “The dress is worn.” in the past and future tenses is as follows.

```
di ta fa 'dress' ne sa 'wear'
du ta fa 'dress' ne sa 'wear'
```

8.6. Other verb modifiers

Functions in the DeterminerV class can modify other verbs. They are similar to English auxiliary verbs.

8.7. Wordbook

English	SFGPL
I	ga
live	sa ‘live’

English	SFGPL
in Tokyo	li pun fa ‘Tokyo’
wear	sa ‘wear’
dress	fa ‘dress’

9. Detailed Grammar

Basically, the SFGPL must adhere strictly to the grammar as described in sentence pattern, but the rest may be decided to some extent by the user. However, an exemplary grammar is described in this chapter.

9.1. How to qualify a sentence

To modify a whole sentence, you basically modify the verbs in that sentence by using `na`. For example, in the example sentence “I go to Tokyo.”, the “to Tokyo” part is a modifier. In this case, the SFGPL uses the following.

```
ta ga na sa 'go' li pun fa 'Tokyo'
```

Another alternative is to use `me`.

```
me ta ga sa 'go' so li pun fa 'Tokyo'
```

9.1.1. Prepositional usage in English

In particular, when modifying verbs, like prepositions in English, they are expressed using `li` and DeterminerN. Examples of English prepositions and SFGPLs are given in the following table.

English	Meaning	SFGPL
at/in/on/to/from	Time	li pin
at/in/on/to/from	Place	li pun
for	Reason	li pon
for	Way/Means	li ban
from	Start	li fan

English	Meaning	SFGPL
to	End	li fen
between/among	Section	li fin
in	In	li fun
into	Into	li tun fun
out	Out	li fon
up/over	Move&Above	li tun man
above	Above	li man
down	Move&Below	li tun men
under	On&Below	li min men
below	Below	li men
on	On	li min
right	Right	li mun
left	Left	li mon
near	Near	li tin
by/about	By/About	li tan tin
with	With	li ten tin

9.2. Grammar of comparative expressions

In the SFGPL, comparative expressions using comparative classes in English are defined by `mo`, but not comparisons using superlative or equivalent classes. It is recommended that such sentences be expressed as follows.

9.2.1. Comparative degree

Comparative expressions such as “A is B(-er) than C” are expressed by `mo`. “My bag is bigger than yours.” is expressed as follows.

```
1 mo mi ga so san fa 'big' so wan sen ge
```

9.2.2. Superlative

Comparative expressions such as “A is the B(-est) in/of C” are expressed with the following syntax.

```
me A V ka ki B li fun C
```

“My bag is the biggest in my class.” is expressed as follows.

```
me mi ga so san fa 'bag' so ka ki wan li fun mu ga so san fa 'class'
```

When expressing “the N-th X(-est)”, a numerical value is added to the modifier, as in `ka X li N`. “My bag is the second biggest in my class.” using ordinal numbers is expressed as follows.

```
me mi ga so san fa 'bag' so ka ki ka wan li mal pil li fun mu ga so san
    fa 'class'
```

9.2.3. Equivalent classes

Comparative expressions such as “A is as B as C” are expressed with the following syntax. In this case, use `wen` to mean “similar”.

```
me ba A C V ka B wen
```

“My bag is as big as his.” is expressed as follows.

```
me ba mi ga so san fa 'bag' sen lan gi so ka wan wen
```

9.3. Diachronic sentences

Constant matters and facts, such as customs, periodic matters and unchanging facts, are expressed by not adding a tense, as is the case with the present.

To express “I cook every day.” in SFGPL, use the following.

```
ta ga na sa 'cook' li pin me fa 'day' so la 'every'
```

“The Earth revolves around the Sun.” in the SFGPL can be expressed as follows.

```
ta fa 'Earth' na sa 'revolve' li tun tin fa 'Sun'
```

And to express “English is spoken all over the world.” in the SFGPL as follows.

```
ta fa 'English' na ne sa 'speak' li fun dan fa 'world'
```

9.4. Syntax for expressing existence

When creating a sentence that simply states that something exists, use gen. This has the same meaning as the English There is/are construction. For example, “There is a book on this table.”.

```
ta fa 'book' na gen li pun ma gu so fa 'table'
```

9.5. Topic-prominent linguistic grammar

It is possible to produce sentences like those in topic-prominent languages, which are common in East Asian languages such as Japanese, Chinese, Korean, and Indonesian. A topic-prominent language is a language in which, in addition to the usual subject, there is a grammar that allows the subject of the sentence to be presented. This makes it easy to produce sentences that contain both a topic and a subject. The SFGPL allows for the production of sentences containing topics in a simplified manner, though not in the explicit manner of the East Asian languages.

9.5.1. Sentences containing a subject or one of the topic or subject

Sentences containing a topic or subject fragment are constructed in the same way as sentence type.

9.5.2. Sentences containing both a topic and a subject

A sentence containing both a topic and a subject is expressed as follows. In this case, “T” is the topic, and “C” consists of comments (sentences, words, etc. that explain the topic).

```
ma T so C
```

As an example, the Japanese phrase “象は鼻が長い”(“Elephants have long noses” [topic: elephant, subject: nose]) can be expressed in SFGPL as follows.

```
ma fa '象' so me fa '鼻' so la '長い'
ma fa 'elephant' so me fa 'nose' so la 'long'
```

9.6. Wordbook

English	SFGPL
I	ga

English	SFGPL
go	sa ‘go’
to Tokyo	li pun fa ‘Tokyo’
bag	fa ‘bag’
big	wan
yours(possessive)	sen ge
my class	mu ga so san fa ‘class’
his(possessive)	sen lan gi
cook	sa ‘cook’
every day	me fa ‘day’ so la ‘every’
the Earth	fa ‘Earth’
revolve	sa ‘revolve’
the Sun	fa ‘Sun’
English	fa ‘English’
speak	sa ‘speak’
all over the world	li fun dan fa ‘world’
book	fa ‘book’
on this table	li pun ma gu so fa ‘table’
象 (elephant)	fa ‘象’
鼻 (nose)	fa ‘鼻’
長い (long)	fa ‘長い’
elephant	fa ‘elephant’
nose	fa ‘nose’
long	la ‘long’

Part III.
SFGPL Word

10. Word

The SFGPL words have a basic set of usages. For example, the way in which loan words are used is defined. This chapter describes the types of these words and how they are used. The details of the words are also available in dict.csv.

In general, SFGPL words are not transformed by articles, number, gender or case. If you want to indicate number or gender, use noun determiner.

10.1. Borrowed Words

The SFGPL uses loan words for all but the basic words. However, half-width double quotation marks (") and half-width spaces ( ) cannot be used. The single quotation mark (') is a symbol to indicate a loanword, so care must be taken if you wish to add it to the beginning or end of a word.

To use loan words for nouns, verbs, and modifiers, use the following table.

Root Word	Part of Speech	SFGPL
apple	Noun	fa 'apple'
open	Verb	sa 'open'
tall	Modifier	la 'tall'

Examples using these words are shown below.

English	SFGPL
I have an apple.	mi ga so fa 'apple'
I open a door.	te ga sa 'open' fa 'door'
I am tall.	me ga so la 'tall'

10.1.1. Borrowed words and the language from which they are borrowed

Borrowing words can be from any language. However, it is preferable to choose words that are understood by both speakers.

For example, the word ‘language’ from any language can be borrowed into the SFGPL as shown in the table below.

Language	Raw Word	SFGPL
English	language	fa ‘language’
Japanese	言語	fa ‘言語’
Spanish	idioma	fa ‘idioma’
French	langue	fa ‘langue’
Russian	я з ы к	fa ‘я з ы к’
Portuguese	linguagem	fa ‘linguagem’
Esperanto	lingvo	fa ‘lingvo’

Thus, it can borrow from a variety of languages. In addition, the borrowed words in this material are basically borrowed from the English language.

10.1.2. How to make borrowed words explicit

The following words exist to make it clear from which language a loanword has been borrowed.

Type	Word
Noun	foa
Verb	soa
Modifier	loa

Noun To borrow the English noun word “language”, do the following.

```
1 foa 'language' 'English'
```

Verb To borrow the English verb word “go”, do the following.

```
soa 'go' 'English'
```

Modifier To borrow the English modifier word “big”, do the following.

```
loa 'big' 'English'
```

10.2. About unique words

The SFGPL provides several unique words for verbs and modifiers. In the WordV and WordM classes, these are word groups that are unique to the SFGPL.

These word groups are highly versatile because their parts of speech have already been determined and they have a broad meaning, but it is difficult to specify the details of their meaning.

The following table gives examples of unique words.

English	SFGPL
create	kan
big	wan

Examples using these words are shown below.

English	SFGPL
I create a door.	te ga kan fa ‘door’
The apple is big.	me fa ‘apple’ so wan

10.2.1. Unique word rules

There is uniqueness with respect to the SFGPL’s unique words: words with different meanings have different pronunciations. The syllable structure is one word and one syllable (CV or CVC).

10.3. About the determiners

As a grammatical function, there are determiners, which are words that simply modify a word. There are two types of determiners: noun determiners, which limit nouns, and verb determiners, which limit verbs.

10.3.1. DeterminerN

The SFGPL has a noun determiner. This is a special word that originally modifies a noun. However, they can also be used as nouns in the same sense as the determiners themselves. To do so, use `fo`. Examples are given in the following table.

English	SFGPL
human	ben fo

Examples using these words are shown below.

English	SFGPL
I am human.	ma ga so ben fo

10.3.2. DeterminerV

There is a verb determiner in the SFGPL. These are special words that modify verbs. These include words used as verb tenses and aspects, and words that add meaning to the verb in an auxiliary verb-like manner.

10.4. About meaningless words

There are words in the SFGPL that do not add meaning. These words exist for each part of speech and are used when grammatically necessary.

First, `fo` of meaningless noun is often used to express noun determiners as they are. Also, `so` of meaningless verb is used when a verb is not needed, especially in sentence pattern. On the other hand, `lo` of meaningless modifier is rarely used. Examples of these are given below.

English	SFGPL
I am human.	ma ga so ben fo
I have an apple.	mi ga so fa ‘apple’

	SFGPL
Noun	fo
Verb	so
Modifier	lo

10.5. About pronouns

Pronouns exist in SFGPL. Pronouns are listed in the following table.

	English	SFGPL
First Person Pronoun	I	ga
Second Person Pronoun	you	ge
Third Person Pronoun	he/she/it	gi
Proximate Pronoun	this	gu
Distant Pronoun	that	go
Interrogative Pronoun	what	wa
Indefinite Pronoun	something	we

10.6. Words used numerically and logically

There are numerical words, words for boolean values, words for lists, words for functions and words for variable in the SFGPL. These words are not often used in general sentences, but are used to indicate logic.

11. Modifier

11.1. About modifiers

There is no difference between adjectives and adverbs in the SFGPL; all words that modify are modifiers.

Modifiers provide words to express the opposite of the modification. It is thereby possible to make `wan` corresponding to the English word “big” into `ke wan`, which means “small”.

11.2. Comparative expressions

The SFGPL has a `mo` of sentences that make comparisons between nouns of two terms. `mo A F B C`, meaning “A is more B than C.”.

Comparative expressions such as “My table is bigger than yours.” are expressed as follows.

```
1 mo mi ga so san fa 'table' so wan sen ge
```

11.3. Modifiers for each part of speech

To simply modify each part of speech with a modifier, the following table is used.

	SFGPL
Noun	me Noun Modifier
Verb	na Verb Modifier
Modifier	ka Modifier Modifier

11.4. Applications of modifiers

Modifiers allow us to substitute English prepositions and noun phrases as modifiers. In this case, the `li`, which converts nouns to modifiers, and noun determiners are often combined to form expressions. For example, “I live in Tokyo.”.

```
1 ta ga na sa 'live' li pun fa 'Tokyo'
```

The `pun` is a determiner of place.

11.5. Wordbook

English	SFGPL
I	ga
table	fa ‘table’
yours	sen ge
live	sa ‘live’
in Tokyo	li pun fa ‘Tokyo’

12. Part of Speech Conversion

The SFGPL can convert nouns, verbs, and modifiers into each other’s parts of speech. The following table lists the words converted to parts-of-speech by the SFGPL.

	Before	After	Word
V2N	Verb	Noun	fi
M2N	Modifier	Noun	fu
M2V	Modifier	Verb	si
N2V	Noun	Verb	su
N2M	Noun	Modifier	li
V2M	Verb	Modifier	lu

Verb to noun and noun to modifier are especially common.

12.1. Verb to Noun

Verb to noun is used as in “This is building.”.

```
ma gu so fi sa 'build'
```

The verb of the original word can also be pre-conjugated according to verb conjugation.

12.2. Noun to Modifier

Noun to modifier is used to create the equivalent meaning of a phrase that combines an English preposition and a noun. In such cases, `li` and DeterminerN are used in combination. “I live in Tokyo.” in SFGPL becomes the following. In this case, `pun` is a determiner of location.

```
ta ga na sa 'live' li pun fa 'Tokyo'
```

It can also be combined with the word `son`, which abstracts the noun, to mean “-like”. “My daughter has a cat-like stuffed toy.” can be expressed in SFGPL as follows.

```
mi mi ga so san fa 'daughter' so me me fa 'toy' so lu ne sa 'stuff' so
    li son fa 'cat'
```

12.3. Verb to Modifier

Verb to modifier conversion allows for the use of the participle equivalent, which is common in the Indo-European language family. The verb of the original word can also be pre-conjugated according to verb conjugation.

“There is a sleeping boy.” can be expressed in the SFGPL as follows.

```
ma pun go so me fa 'boy' so lu ni sa 'sleep'
```

The phrase “I lived in that destroyed building.” can be expressed as follows.

```
di ta ga na sa 'live' li pun ma go so san me fi sa 'build' so lu ne sa
    'destroy'
```

12.4. Wordbook

English	SFGPL
this	gu
build	sa ‘build’
I	ga
live	sa ‘live’
in Tokyo	li pun fa ‘Tokyo’
daughter	fa ‘daughter’

English	SFGPL
cat	fa ‘cat’
stuffed	lu ne sa ‘stuff’
toy	fa ‘toy’
there	pun go
sleep	sa ‘sleep’
boy	fa ‘boy’
that	go
destroy	sa ‘destroy’

13. Conjunction

In the SFGPL, conjunctions exist as connections between sentences and between words. The main conjunctions of the SFGPL are as follows.

Word	English Word	English	SFGPL
pe	because	I go to a store, because I want it.	pe ta ga na sa ‘go’ li pun fa ‘store’ te ga sa ‘want’ pen gi
pu	so	I want it, so I go to a store.	pu te ga sa ‘want’ pen gi ta ga na sa ‘go’ li pun fa ‘store’
pi	if	I go to a store, if I want it.	pi ta ga na sa ‘go’ li pun fa ‘store’ te ga sa ‘want’ pen gi
po	but	I want it, but I don’t go to a store.	po te ga sa ‘want’ pen gi pa ta ga na sa ‘go’ li pun fa ‘store’
ba	and	I go to a store, and I go to a library.	ba ta ga na sa ‘go’ li pun fa ‘store’ ta ga na sa ‘go’ li pun fa ‘library’

Word	English Word	English	SFGPL
be	or	I go to a store, or I go to a library.	I go to a store, or I go to a library.

You can also connect words together, such as `ba fa 'store'fa 'library'` or `be fa 'store'fa 'library'`.

13.1. Wordbook

English	SFGPL
I go to a store	ta ga na sa ‘go’ li pun fa ‘store’
I don’t go to a store	pa ta ga na sa ‘go’ li pun fa ‘store’
I want it	te ga sa ‘want’ pen gi
I go to a library	ta ga na sa ‘go’ li pun fa ‘library’
store	fa ‘store’
library	fa ‘library’

14. Pronoun

14.1. List of pronouns

Pronouns are listed in the following table.

	English	SFGPL
First Person Pronoun	I	ga
Second Person Pronoun	you	ge
Third Person Pronoun	he/she/it	gi
Proximate Pronoun	this	gu
Distant Pronoun	that	go
Interrogative Pronoun	what	wa

	English	SFGPL
Indefinite Pronoun	something	we

14.2. Pronoun applications

As a rule, SFGPL pronouns do not distinguish between people, organisms, objects, concepts, places, times, reasons, methods, etc. There is no distinction based on gender or number. These distinctions can be made by using noun determiner.

14.2.1. Interrogative word

The following table shows the use of noun determiners for interrogatives.

English	SFGPL
what	pen wa
who	ben wa
when	pin wa
where	pun wa
why	pon wa
how	ban wa

14.2.2. Plural pronouns

To indicate plurals, use `don`. For example, `don ga` is used to denote “We”.

Clusivity of person pronoun In particular, the plural of first-person pronouns may be distinguished based on whether they include the speaker or the addressee. The SFGPL cannot directly distinguish between these, but it can do so by doing the following.

	Include the addressee	Exclude the addressee
Include the Speaker	`don ba ge ga`	`don ba ga gi`

	Include the addressee	Exclude the addressee
Exclude the Speaker	`don ge`	`don gi`

14.2.3. Examples of conjugation of third person pronouns

Gender distinctions do not exist in the SFGPL. Nor is there a distinction between persons and things. For example, to make explicit the third person pronouns masculine, feminine and thing, one can do the following.

	English	SFGPL
male	he	lan gi
female	she	len gi
thing	it	pen gi

14.2.4. Possessive and Recursive pronouns

In addition, you can create possessive and reflexive pronouns using `sen` and `sin`. The following table shows the possessive and reflexive pronouns for first person pronouns.

	English	SFGPL
Possessive Pronoun	mine	sen ga
Reflexive Pronoun	myself	sin ga

15. DeterminerN

DeterminerN are the simplest of all noun modifiers. They are also often used with pronouns or with `li`, which is used to convert a noun to a modifier.

The following table shows examples of Noun DeterminerN.

Word	Base Meaning	English	SFGPL
lan	male	He is student.	ma lan gi so fa ‘student’
len	female	She is student.	ma len gi so fa ‘student’
don	plural	They are student.	ma don gi so fa ‘student’
pun	place	I go to Tokyo.	ta ga na sa ‘go’ li pun fa ‘Tokyo’
pin	time	I go today.	ta ga na sa ‘go’ li pin fa ‘today’

DeterminerN can be added in multiples.

In general, in the case of the DeterminerN A, B and the noun N, the clause `A B N` means ‘(N of B) of A’.

15.1. Wordbook

English	SFGPL
he/she/they	gi
student	fa ‘student’
I	ga
go	sa ‘go’
Tokyo	fa ‘Tokyo’
today	fa ‘today’

16. DeterminerV

Verb DeterminerV are the simplest to modify verbs. They are the equivalent of English auxiliary verbs. The following table shows some examples of Verb DeterminerV.

Word	Base Meaning	English	SFGPL
nak	possible	I can see a sea.	te ga nak sa ‘see’ fa ‘sea’
nek	ability	I can swim.	ta ga nek sa ‘swim’

Word	Base Meaning	English	SFGPL
nuk	obligation	I should swim.	ta ga nuk sa ‘swim’
nok	necessary	I need to swim.	ta ga nok sa ‘swim’
lak	duty	I must swim.	ta ga lak sa ‘swim’
lik	want to	I want to swim.	ta ga lik sa ‘swim’

We can also do verb conjugation, such as aspect.

16.1. Wordbook

English	SFGPL
I	ga
see	sa ‘see’
sea	fa ‘sea’
swim	sa ‘swim’

17. Bool related classes

SFGPL has classes related to Bool, Bool type and BoolList type. These classes are used to represent boolean values, numerical values, and so on.

17.1. About Bool class

The Bool type is a class for representing true or false. False and True of type Bool are represented as follows.

	word
False	pas
True	pos

You can also use `pis` to connect a Bool type to a noun to indicate the truth or falsehood of a noun. The following statement is an example. In such a statement, the whole is inherited as True.

```
pis ma ga so fa 'student' pos
```

Bool types can also use NOT `pa`, OR `be`, AND `ba`, NOR `bo` and NAND `bu`, which are provided in LangObj. They can then perform logic operations.

For example, to represent `True OR False`, the following is used.

```
be pos pas
```

Similarly, the Bool type can be used for inherited nouns. The following shows the negation of "It is true that I am a student.

```
pa pis ma ga so fa 'student' pos
```

Besides the usual IFELSE `bi`, LangObj has a logicIFELSE `ja`. This word allows you to change the sentence (word) to be executed internally depending on whether or not the condition is met. "If true, I am a student." can be expressed as follows.

```
ja pos ma ga so fa 'student' pa ma ga so fa 'student'
```

17.2. About BoolList class

BoolList can create an array of boolean values. The following functions exist in BoolList.

Word	Explanation
fas	Create a list of true/false (BoolList)
fes A B	Gets the B-th value of BoolList(A)
fis A B	Add one Bool (B) to the end of the BoolList (A)
fus A B C	Get the B-th through C-th lists for a BoolList (A)
fos A B	Combine two BoolLists (A,B)
foas A	Get the length of the BoolList (A)
mas A B	Create a BoolList consisting of 2 Bool values (A,B)
mis X1~X4	Create a BoolList consisting of 4 Bool values (x1~x4)
mos X1~X8	Create a BoolList consisting of 8 Bool values (x1~x8)
tas A	BoolList (A) is considered a binary natural number

Word	Explanation
tes A	BoolList (A) is considered a binary integer
tis A	BoolList (A) is considered a binary floating number
tus A	BoolList (A) is considered an ASCII character

4-byte data can be used by doing the following.

```
fos fos mos pas pos pas pas pas pas pas pas mos pas pos pas pas pos pas pas pos fos mos pas pas pas pas pos pos pos pos mos pos pos pas pos pos pas pos pos
```

This represents `0100 0000 0100 1001 0000 1111 1101 1011` in binary. It can also be used as a number by doing the following.

Type	SFGPL	Value
Natural Number	tas fos fos mos pas pos pas pas pas pas pas pas mos pas pos pas pas pos pas pas pos fos mos pas pas pas pas pos pos pos pos mos pos pos pas pos pos pas pos pos	1078530011
Integer Number	tes fos fos mos pas pos pas pas pas pas pas pas mos pas pos pas pas pos pas pas pos fos mos pas pas pas pas pos pos pos pos mos pos pos pas pos pos pas pos pos	1078530011
Floating Point Number	tis fos fos mos pas pos pas pas pas pas pas pas mos pas pos pas pas pos pas pas pos fos mos pas pas pas pas pos pos pos pos mos pos pos pas pos pos pas pos pos	3.1415927410125732

Floating point corresponds to half, single, double and quadruple precision in IEEE 754. Therefore, they

must be expressed in 16-bit, 32-bit, 64-bit and 128-bit respectively.

To express 1/3 in each precision is as follows. First, the hexadecimal representation is as follows.

Type	HEX
Half	3555
Single	3eaa aaab
Double	3FD5 5555 5555 5555
Quadruple	3ffd 5555 5555 5555 5555 5555 5555 5555

This is converted into the SFGPL as follows.

Type	SFGPL
Half	tis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fas pas pas pos pos pas pos pas pos pas pos pas pos pas pos pas pos
Single	tis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fas pas pas pos pos pos pos pos pas pos pas pos pas pos pas pos pas pos pas pos pas pos pas pos pas pos pas pos pas pos pos

Type	SFGPL
Double	`tis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fis fas pas pas pos pos pos pos pos pos pos pos pas pos`

Type	SFGPL
Quadruple	tis fas pas pas pos pos pos pos pos pos pos pos pos pos pos pos pas pos pos pas pos pas pos pas pos pas pas pos pas pos pas pos pas pos pas pos pas pos pas pos pas pos pas pos pas pos pas pos pas pos pas pos pas pos

17.3. BoolList date/time representation

BoolList can be used to represent dates and times based on Unix time. There are three types of date/time expressions, depending on their accuracy.

SFGPL	Type	Unit
das	yyyy-mm-dd	Day
des	yyyy-mm-dd HH:MM:SS	Second
dis	yyyy-mm-dd HH:MM:SS.nnnnnnnnn	Nano Second

These expressions are based on `1970-01-01 00:00:00.00000000000` and represent the date and time by the difference in days, seconds and nanoseconds respectively. They are also based on UTC time.

For example, to represent `2024-09-19 09:27:27` by `des` as follows.

First, the Unix time for this hour is `1726738047`. Converting this to a binary number gives `0110 0110 1110 1011 1110 1110 0111 1111`. Therefore, converting it to a BoolList gives the following.

```
fos fos mos pas pos pos pas pas pos pos pas mos pos pos pos pas pos pas
    pos pos fos mos pos pos pos pas pos pos pos pas mos pas pos pos pos
    pos pos pos pos
```

Furthermore, the conversion to date and time using `des` gives the following.

```
des fos fos mos pas pos pos pas pas pos pos pas mos pos pos pos pas pos
    pas pos pos fos mos pos pos pos pas pos pos pos pas mos pas pos pos
    pos pos pos pos pos
```

This allows `2024-09-19 09:27:27` to be represented by the SFGPL.

17.4. Wordbook

English	SFGPL
I am a student	ma ga so fa ‘student’

18. LangList

The LangList type exists as a basic data structure type in SFGPL. The following functions exist in LangList.

Word	Explanation
fat	Create a list of LangObj (LangList)
fet A B	Gets the B-th value of LangList (A)
fit A B	Add one LangObj (B) to the end of the LangList (A)
fut A B C	Get the B-th through C-th lists for a LangList (A)
fot A B	Combine two LangLists
foat A	Get the length of the LangList (A)
tat A B C	Function for iteration using LangList
tet A B	Function to perform certain processing on all elements of LangList

LangList can store all classes that inherit from LangObj. The following is an example of creating a LangList using append.

```
fit fit fit fit fit fat ga fa 'pen' sa 'go' la 'happy' ma ga so fa '
    student'
```

To retrieve the first value from this LangList, do the following. In this case `fis fas pas` represents 0 in BoolList.

```
fet fit fit fit fit fit fat ga fa 'pen' sa 'go' la 'happy' ma ga so fa
    'student' fis fas pas
```

18.1. Iteration in LangList

LangList can be iterated using `tat` to iterate through LangList. The *x*s below are all the same LangList variables used in the While function.

The first argument A sets the initial value of the loop. The value of A is first assigned to *x*.

The second argument B sets the name of the predefined LangFunc. The function named B is a conditional statement on the loop, and *x* is assigned to the argument of this function. The zeroth

element of the function's return value, LangList, is of type Bool, indicating whether the condition is satisfied, and if this value is True, the loop continues.

The third argument C is set to the name of a predefined LangFunc. The function named C is the content of the iterative process, and *x* is assigned to the argument of this function. The function then sets the updated *x* as the return value.

At the end of the loop, the final *x* result is output.

The following is an example using "`tat`.

```
pat fa 'condition_func' fit fat sal tel mal pol fet pit tol mal pal
pat fa 'process_func' fit fit fit fat tal fet pit tol mal pal mal pel
    tal fet pit tol mal pel mel pel pal til fet pit tol mal pil mal pil
tat fit fit fit fat mal pal mal pal mal pel fa 'condition_func' fa '
    process_func'
```

The first line sets up the function of the conditional statement. The function of the conditional statement is defined as "condition_func", which executes a loop while `4-x[0]>=0` is True.

In the second line, the function of the processing statement is set. The function of this processing statement is "process_func" and updates each element. The contents to be updated are set to `[x[0]+1,x[1]+10,x[2]*2]`.

The third line actually executes the iteration. In this case, `[0,0,1]` is given as the initial value.

18.2. LangList map functions

There is a function `tet` that performs certain operations on all elements of a LangList. In this case, the first argument is the LangList A to be adapted and the second argument is the function name B for certain processing.

In this case, the function B is passed `[the data of each element, the index (NumberList) of that element, LangList A]` of type LangList as arguments. The value of LangList[0] resulting from the execution of the function in B is used as the value of the new element.

Then, to add 1 to all elements using `tet`, do the following.

```
pat fa 'map_func' fit fat tal fet pit tol mal pal mal pel
tet fit fit fit fat mel pel pal mel pel pel mel pel pil fa 'map_func'
```

The first line sets up the function for processing. This process adds 1 to the data of each element.

In the second line, the process of actually assigning `[10,11,12]` and adding 1 to all elements is executed. In this case, `[10,11,12]` can be represented by `fit fit fit fat mel pel pal mel pel pel mel pel pil`.

18.3. Wordbook

English	SFGPL
I	ga
pen	fa ‘pen’
go	sa ‘go’
happy	la ‘happy’
I am a student	ma ga so fa ‘student’

19. LangFunc

The LangFunc type exists as a basic function type in SFGPL. The following functions exist in LangFunc.

Word	Explanation
pat A B	Set up a function that returns B named A with a certain LangList as an argument
pit	Used for pat arguments
pot A B	Execute the configured LangFunc named A with argument B

LangFunc sets the function by `pat`. Also, `pit` can be included in the second argument of `pat` statement. This will cause the actual value to be assigned and processed when the function is executed. The first argument of `pat` is a function name. And the function name cannot be duplicated. The following is an example of a function setup.

```
pat fa 'xor' fit fat bu bu fet pit mas pas pas bu fet pit mas pas pas
    fet pit mas pas pos bu bu fet pit mas pas pas fet pit mas pas pos
    fet pit mas pas pos
```

The function takes the XOR of the zeroth and first values of a LangList. When (false,false) is given to the function, do the following.

```
pot fa 'xor' fit fit fat pas pas
```

20. LangVar

In SFGPL, a variable that stores LangList can be created.

Word	Explanation
bat A B	Assign LangList B to the variable named A
bot A	Get a variable named A

To store LangList`["apple","banana"]` in a variable named var1, do the following.

```
bat fa 'var1' fit fit fat fa 'apple' fa 'banana'
```

To obtain var1, do the following.

```
bot fa 'var1'
```

21. How numbers are expressed

The Number and NumberList classes exist in SFGPL to represent decimal numbers.

21.1. Number class

The Number class is a class for cardinal numerals and is not used by itself. In this class, values from 0 to 9 are defined, as shown in the table below.

Meaning	SFGPL
0	pal
1	pel
2	pil
3	pul
4	pol
5	bal
6	bel
7	bil

Meaning	SFGPL
8	bul
9	bol

21.2. NumberList class

Use the NumberList class when used as a normal numeral. This class can store radix data in a list. Numbers are represented as decimal numbers, with the largest digit stored first, starting with the 0th digit.

The NumberList class has the following list-type functions. However, these functions cannot be applied to NumberList after numerical calculation as described below.

Word	Explanation
fal	Create a list of Number(NumberList)
fel A B	Gets the B-th value of NumberList(A)
fil A B	Add one Number to the end of the NumberList
ful A B C	Get the B-th through C-th lists for a NumberList (A)
fol A B	Combine two NumberLists
foal A	Get the length of the NumberList (A)

In addition, dedicated functions are available to create 1~5-digit integers, as shown in the table below.

Word	Explanation
mal	Create a NumberList consisting of one decimal digit
mel	Create a NumberList consisting of two decimal digit
mil	Create a NumberList consisting of three decimal digit
mul	Create a NumberList consisting of four decimal digit
mol	Create a NumberList consisting of five decimal digit

In the SFGPL, “I have five apples.” can be expressed as follows.

```
mi ga so ma fa 'apple' so mal bal
```

The expression “I have fifteen apples.” can be expressed as follows.

```
mi ga so ma fa 'apple' so mel pel bal
```

Furthermore, the representation of numbers with more than five digits in decimal can be achieved by using `fol` and concatenating NumberList. The following sentence represents “Japan has 125416877 people.” in the SFGPL.

```
mi fa 'Japan' so ma fa 'people' so fol mul pel pil bal pol mol pel bel
    bul bil bil
```

21.2.1. Numeric calculation

As shown in the following table, NumberList has functions that perform numerical calculations such as four arithmetic operations.

	Python	SFGPL
Addition	`A+B`	tal A B
Subtraction	`A-B`	tel A B
Multiplication	`A*B`	til A B
Division	`A/B`	tul A B
Power	`A**B`	dal A B
Int Division	`A//B`	del A B
Remainder	`A%B`	dil A B
Minus	`-A`	sel A
Absolute value	`abs(A)`	sil A

21.2.2. Interconversion between BoolList and NumberList

As shown in the following table, there are functions that convert BoolList and NumberList into each other.

Type	SFGPL	from	to
Int	tol	NumberList	BoolList
Int	tos	BoolList	NumberList
Float	dol	NumberList	BoolList
Float	dos	BoolList	NumberList

Mutual Conversion in Integer Types There are `tol` and `tos` functions that convert each other as integers. The numeric values handled by these conversions consider the BoolList as an integer type (`tes`). In other words, the value of the BoolList is equivalent to the two's complement representation of a binary number. These values can also be adapted if numerical calculations, such as four arithmetic operations, are performed by NumberList. However, if NumberList is a real number due to the result of division, etc., the conversion cannot be performed and an error occurs.

Mutual Conversion in Floating-Point Type (Real Number) There are `dol` and `dos` that convert each other as floating-point (real) numbers. The numbers handled by these conversions consider BoolList as a floating-point type (`tis`). In other words, the BoolList values in this case use the half-precision, single-precision, double-precision, and quadruple-precision floating-point representation methods in IEEE754. When converting from NumberList to BoolList, it is converted as a 64-bit double-precision floating-point number and stored in BoolList.

21.2.3. How to handle real numbers

To handle real numbers, you can use NumberList division (`tul`), or you can represent floating-point numbers in a BoolList and convert it to a NumberList.

For example, 3.14 can be expressed by division as follows.

```
1 tul mil pul pel pol mil pel pal pal
```

Similarly, to express 3.14 in double-precision floating point, do the following.

```
1 dos fos fos fos mos pas pos pas pas pas pas pas pas mos pas pas pas pas
      pos pas pas pos fos mos pas pas pas pos pos pos pos pas mos pos pas
      pos pos pos pas pas pas fos fos mos pas pos pas pos pas pas pas pos
      mos pos pos pos pas pos pas pos pos fos mos pos pas pas pas pas pos
      pas pos mos pas pas pas pos pos pos pos pos
```

21.2.4. Determination of positive numbers

To determine whether a NumberList is a positive number, use `sal`. This will output the SFGPL Bool type, which is equivalent to `pos` if the number is greater than or equal to zero. For example, to determine whether an integer is positive in the two cases of 4 and -4, do the following.

```
sal mal pol
sal sel mal pol
```

21.3. Wordbook

English	SFGPL
I	ga
apple	fa ‘apple’
Japan	fa ‘Japan’
people	fa ‘people’

Part IV.
Appendix

22. Examples of the use of loan words other than those of English origin

The SFGPL also allows the use of loanwords that are not of English origin. In such cases, the usage is basically the same as for English. However, the word order cannot be changed, so it may differ from the word order of the original language.

22.1. Borrowed words of Japanese origin

For example, to form the sentence “私はりんごを持っている。”, use the following.

```
mi ga so fa 'りんご'
```

The sentence “私の鞄は赤い。”, which Compound Sentences contains, should be as follows.

```
me mi ga so san fa '鞄' so la '赤い'
```

22.2. Borrowed words of Esperanto origin

When using Esperanto words as loan words, it is recommended that, as a rule, the form without the part-of-speech suffix should be used.

For example, to form the sentence “Mi havas pomon.” do the following.

```
mi ga so fa 'pom'
```

For example, to form the sentence “Mia sako estas ruĝa.” do the following.

```
me mi ga so san fa 'sak' so la 'ruĝ'
```

23. Example Sentence

The following table shows example sentences from the SFGPL.

SFGPL	Python	Translation
ma ga so me fa ‘worker’ so li pun fa ‘office’	`Noun.eq(Pronoun.I(),Verb.none(),Noun.haveP(Noun("'worker'"),Verb.none(),Modifier.N2M(DeterminerN.place(Noun("'office'")))))`	I am an office worker.
ma ge so me fa ‘worker’ so li pun fa ‘office’	`Noun.eq(Pronoun.you(),Verb.none(),Noun.haveP(Noun("'worker'"),Verb.none(),Modifier.N2M(DeterminerN.place(Noun("'office'")))))`	You are an office worker.

SFGPL	Python	Translation
ma lan gi so me fa ‘worker’ so li pun fa ‘office’	`Noun.eq(DeterminerN.male(Pronoun.he()),Verb.none(),Noun.haveP(Noun("'worker'"),Verb.none(),Modifier.N2M(DeterminerN.place(Noun("'office'")))))`	He is an office worker.
ma len gi so me fa ‘worker’ so li pun fa ‘office’	`Noun.eq(DeterminerN.female(Pronoun.he()),Verb.none(),Noun.haveP(Noun("'worker'"),Verb.none(),Modifier.N2M(DeterminerN.place(Noun("'office'")))))`	She is an office worker.
ma don ga so me fa ‘worker’ so li pun fa ‘office’	`Noun.eq(DeterminerN.plural(Pronoun.I()),Verb.none(),Noun.haveP(Noun("'worker'"),Verb.none(),Modifier.N2M(DeterminerN.place(Noun("'office'")))))`	We are office workers.
ma don ge so me fa ‘worker’ so li pun fa ‘office’	`Noun.eq(DeterminerN.plural(Pronoun.you()),Verb.none(),Noun.haveP(Noun("'worker'"),Verb.none(),Modifier.N2M(DeterminerN.place(Noun("'office'")))))`	You are office workers.
ma don gi so me fa ‘worker’ so li pun fa ‘office’	`Noun.eq(DeterminerN.plural(Pronoun.he()),Verb.none(),Noun.haveP(Noun("'worker'"),Verb.none(),Modifier.N2M(DeterminerN.place(Noun("'office'")))))`	They are office workers.

SFGPL	Python	Translation
di ma ga so me fa ‘worker’ so li pun fa ‘office’	`Phrase.past(Noun.eq(Pronoun.I(),Verb.none(),Noun.haveP(Noun("'worker'"),Verb.none(),Modifier.N2M(DeterminerN.place(Noun("'office'"))))))`	I was an office worker.
du ma ga so me fa ‘worker’ so li pun fa ‘office’	`Phrase.future(Noun.eq(Pronoun.I(),Verb.none(),Noun.haveP(Noun("'worker'"),Verb.none(),Modifier.N2M(DeterminerN.place(Noun("'office'"))))))`	I will be an office worker.
ta ga na sa ‘go’ li pun mu ga so san fa ‘school’	`Noun.do(Pronoun.I(),Verb.add(Verb("'go'"),Modifier.N2M(DeterminerN.place(Noun.belong(Pronoun.I(),Verb.none(),DeterminerN.stressed(Noun("'school'")))))))`	I go to my school.
di ta ga na sa ‘go’ li pun mu ga so san fa ‘school’	`Phrase.past(Noun.do(Pronoun.I(),Verb.add(Verb("'go'"),Modifier.N2M(DeterminerN.place(Noun.belong(Pronoun.I(),Verb.none(),DeterminerN.stressed(Noun("'school'")))))))))`	I went to my school.

SFGPL	Python	Translation
du ta ga na sa ‘go’ li pun mu ga so san fa ‘school’	`Phrase.future(Noun.do(Pronoun.I(),Verb.add(Verb("'go'"),Modifier.N2M(DeterminerN.place(Noun.belong(Pronoun.I(),Verb.none(),DeterminerN.stressed(Noun("'school'"))))))))`	I will go to my school.
te ga sa ‘read’ fa ‘book’	`Noun.doT(Pronoun.I(),Verb("'read'"),Noun("'book'"))`	I read a book.
di ti ga na sa ‘send’ li pin fa ‘yesterday’ lan gi fa ‘letter’	`Phrase.past(Noun.give(Pronoun.I(),Verb.add(Verb("'send'"),Modifier.N2M(DeterminerN.time(Noun("'yesterday'"))))),DeterminerN.male(Pronoun.he()),Noun("'letter'")))`	I sent him a letter yesterday.
di tu ga so lan gi fa ‘teacher’	`Phrase.past(Noun.makeN(Pronoun.I(),Verb.none(),DeterminerN.male(Pronoun.he()),Noun("'teacher'")))`	I made him a teacher.
di to ga so lan gi la ‘happy’	`Phrase.past(Noun.makeM(Pronoun.I(),Verb.none(),DeterminerN.male(Pronoun.he()),Modifier("'happy'")))`	I made her happy.
mo lan gi so la ‘tall’ ga	`Noun.gt(DeterminerN.male(Pronoun.he()),Verb.none(),Modifier("'tall'"),Pronoun.I())`	He is taller than me.

SFGPL	Python	Translation
di te ga na sa ‘put’ li pun min fa ‘table’ ba fa ‘apple’ fa ‘peach’	`Phrase.past(Noun.doT(Pronoun.I(),Verb.add(Verb("'put'"),Modifier.N2M(DeterminerN.place(DeterminerN.on(Noun("'table'"))))),LangObj.AND(Noun("'apple'"),Noun("'peach'"))))`	I put an apple and a peach on the table.
ta ga na sa ‘go’ li pun fa ‘Osaka’	`Noun.do(Pronoun.I(),Verb.add(Verb("'go'"),Modifier.N2M(DeterminerN.place(Noun("'Osaka'")))))`	I go to Osaka.
di ta ga na sa ‘go’ li pun fa ‘Osaka’	`Phrase.past(Noun.do(Pronoun.I(),Verb.add(Verb("'go'"),Modifier.N2M(DeterminerN.place(Noun("'Osaka'"))))))`	I went to Osaka.
du ta ga na sa ‘go’ li pun fa ‘Osaka’	`Phrase.future(Noun.do(Pronoun.I(),Verb.add(Verb("'go'"),Modifier.N2M(DeterminerN.place(Noun("'Osaka'"))))))`	I will go to Osaka.
te ga sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(),Verb("'create'"),Noun("'table'"))`	I create a table.
te ga sa ‘create’ ma gu so san fa ‘table’	`Noun.doT(Pronoun.I(),Verb("'create'"),Noun.eq(Pronoun.proximal(),Verb.none(),DeterminerN.stressed(Noun("'table'"))))`	I create this table.

SFGPL	Python	Translation
pa te ga sa ‘create’ fa ‘table’	`LangObj.NOT(Noun.doT(Pronoun.I(),Verb("'create'"),Noun("'table'")))`	I don’t create a table.
te ge sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.you(),Verb("'create'"),Noun("'table'"))`	You create a table.
da te ge sa ‘create’ fa ‘table’	`Phrase.interrogative(Noun.doT(Pronoun.you(),Verb("'create'"),Noun("'table'")))`	Do you create a table?
da di te ge sa ‘create’ fa ‘table’	`Phrase.interrogative(Phrase.past(Noun.doT(Pronoun.you(),Verb("'create'"),Noun("'table'"))))`	Did you create a table?
da te ben wa sa ‘create’ fa ‘table’	`Phrase.interrogative(Noun.doT(DeterminerN.human(Pronoun.interrogative()),Verb("'create'"),Noun("'table'")))`	Who create the table?
da te ge sa ‘create’ pen wa	`Phrase.interrogative(Noun.doT(Pronoun.you(),Verb("'create'"),DeterminerN.thing(Pronoun.interrogative())))`	What do you create?

SFGPL	Python	Translation
da te ge na sa ‘create’ li pin wa fa ‘table’	`Phrase.interrogative(Noun.doT(Pronoun.you(),Verb.add(Verb("'create'"),Modifier.N2M(DeterminerN.time(Pronoun.interrogative()))),Noun("'table'")))`	When do you create the table?
da te ge na sa ‘create’ li pon wa fa ‘table’	`Phrase.interrogative(Noun.doT(Pronoun.you(),Verb.add(Verb("'create'"),Modifier.N2M(DeterminerN.reason(Pronoun.interrogative()))),Noun("'table'")))`	Why do you create the table?
de te we sa ‘create’ fa ‘table’	`Phrase.imperative(Noun.doT(Pronoun.indefinite(),Verb("'create'"),Noun("'table'")))`	Create a table!
di te ga sa ‘create’ fa ‘table’	`Phrase.past(Noun.doT(Pronoun.I(),Verb("'create'"),Noun("'table'")))`	I created a table.
du te ga sa ‘create’ fa ‘table’	`Phrase.future(Noun.doT(Pronoun.I(),Verb("'create'"),Noun("'table'")))`	I will create a table.

SFGPL	Python	Translation
ta fa ‘table’ na ne sa ‘create’ li tan tin ga	`Noun.do(Noun("'table'"),Verb.add(Verb.passive(Verb("'create'")),Modifier.N2M(DeterminerN.affect(DeterminerN.near(Pronoun.I()))))`	The table is created by me.
te ga ni sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(),Verb.progressive(Verb("'create'")),Noun("'table'"))`	I am creating a table.
te ga nu sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(),Verb.perfective(Verb("'create'")),Noun("'table'"))`	I have created a table.
du te ga pak sa ‘create’ fa ‘table’	`Phrase.future(Noun.doT(Pronoun.I(),DeterminerV.Estimation100(Verb("'create'")),Noun("'table'")))`	I 100% probability will create a table.
du te ga pek sa ‘create’ fa ‘table’	`Phrase.future(Noun.doT(Pronoun.I(),DeterminerV.Estimation75(Verb("'create'")),Noun("'table'")))`	I 75% probability will create a table.
du te ga pik sa ‘create’ fa ‘table’	`Phrase.future(Noun.doT(Pronoun.I(),DeterminerV.Estimation50(Verb("'create'")),Noun("'table'")))`	I 50% probability will create a table.

SFGPL	Python	Translation
du te ga puk sa ‘create’ fa ‘table’	`Phrase.future(Noun.doT(Pronoun.I(), DeterminerV.Estimation25(Verb("'create'")),Noun("'table'")))`	I 25% probability will create a table.
du te ga pok sa ‘create’ fa ‘table’	`Phrase.future(Noun.doT(Pronoun.I(), DeterminerV.Estimation0(Verb("'create'")),Noun("'table'")))`	I 0% probability will create a table.
te ga fak sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(), DeterminerV.Frequency100(Verb("'create'")),Noun("'table'"))`	I 100% frequently create a table.
te ga fek sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(), DeterminerV.Frequency75(Verb("'create'")),Noun("'table'"))`	I 75% frequently create a table.
te ga fik sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(), DeterminerV.Frequency50(Verb("'create'")),Noun("'table'"))`	I 50% frequently create a table.
te ga fuk sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(), DeterminerV.Frequency25(Verb("'create'")),Noun("'table'"))`	I 25% frequently create a table.

SFGPL	Python	Translation
te ga fok sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(), DeterminerV.Frequency0 (Verb("'create'")), Noun("'table'"))`	I 0% frequently create a table.
te ga bik sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(), DeterminerV.present( Verb("'create'")),Noun ("'table'"))`	I create a table.
te ga bak sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(), DeterminerV.past(Verb( "'create'")),Noun("' table'"))`	I created a table.
te ga bok sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(), DeterminerV.future( Verb("'create'")),Noun ("'table'"))`	I will create a table.
di te ga bak sa ‘create’ fa ‘table’	`Phrase.past(Noun.doT( Pronoun.I(), DeterminerV.past(Verb( "'create'")),Noun("' table'")))`	I created a table.(Past in the past at a point in time)
di te ga bik sa ‘create’ fa ‘table’	`Phrase.past(Noun.doT( Pronoun.I(), DeterminerV.present( Verb("'create'")),Noun ("'table'")))`	I created a table.(Present in the past at a point in time)
di te ga bok sa ‘create’ fa ‘table’	`Phrase.past(Noun.doT( Pronoun.I(), DeterminerV.future( Verb("'create'")),Noun ("'table'")))`	I would create a table.(Future in the past at a point in time)

SFGPL	Python	Translation
di te ga bak sa ‘create’ fa ‘table’	`Phrase.past(Noun.doT(Pronoun.I(),DeterminerV.past(Verb("'create'")),Noun("'table'")))`	I will have created a table.(Past in the future at a point in time)
di te ga bik sa ‘create’ fa ‘table’	`Phrase.past(Noun.doT(Pronoun.I(),DeterminerV.present(Verb("'create'")),Noun("'table'")))`	I will create a table.(Present in the future at a point in time)
di te ga bok sa ‘create’ fa ‘table’	`Phrase.past(Noun.doT(Pronoun.I(),DeterminerV.future(Verb("'create'")),Noun("'table'")))`	I will create a table.(Future in the future at a point in time)
te ga nak sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(),DeterminerV.Possible(Verb("'create'")),Noun("'table'"))`	I can create a table.
te ga nek sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(),DeterminerV.Ability(Verb("'create'")),Noun("'table'"))`	I can create a table.
te ga nik sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(),DeterminerV.Will(Verb("'create'")),Noun("'table'"))`	I will create a table.
te ga nuk sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(),DeterminerV.Obligation(Verb("'create'")),Noun("'table'"))`	I should create a table.

SFGPL	Python	Translation
te ga nok sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(), DeterminerV.Necessary( Verb("'create'")),Noun ("'table'"))`	I need to create a table.
te ga lak sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(), DeterminerV.Duty(Verb( "'create'")),Noun("' table'"))`	I must create a table.
te ga lek sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(), DeterminerV.forced( Verb("'create'")),Noun ("'table'"))`	I am forced to create a table.
te ga lik sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(), DeterminerV.want(Verb( "'create'")),Noun("' table'"))`	I want to create a table.
te ga luk sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(), DeterminerV.dare(Verb( "'create'")),Noun("' table'"))`	I dare create a table.
te ga lok sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(), DeterminerV.allow(Verb ("'create'")),Noun("' table'"))`	I allow to create a table.
te ga kak sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(), DeterminerV.easy(Verb( "'create'")),Noun("' table'"))`	I am easy to create a table.
te ga kek sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(), DeterminerV.hard(Verb( "'create'")),Noun("' table'"))`	I am hard to create a table.

SFGPL	Python	Translation
te ga kik sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(), DeterminerV.habit(Verb("'create'")),Noun("'table'"))`	I habitually create a table.
te ga kuk sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(), DeterminerV.Polite(Verb("'create'")),Noun("'table'"))`	I create a table.(polite expression)
te lan gi kok sa ‘create’ fa ‘table’	`Noun.doT(DeterminerN.male(Pronoun.he()), DeterminerV.Respect(Verb("'create'")),Noun("'table'"))`	He creates a table.(respectful expression)
te ga gak sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(), DeterminerV.volitional(Verb("'create'")), Noun("'table'"))`	I consciously create a table.
te ga gek sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(), DeterminerV.nonVolitional(Verb("'create'")),Noun("'table'"))`	I unconsciously create a table.
da te ge gik sa ‘create’ fa ‘table’	`Phrase.interrogative(Noun.doT(Pronoun.you(),DeterminerV.Requests(Verb("'create'")),Noun("'table'")))`	Can you create a table?
da te ga guk sa ‘create’ fa ‘table’	`Phrase.interrogative(Noun.doT(Pronoun.I(), DeterminerV.Permission(Verb("'create'")), Noun("'table'")))`	May I create a table?

SFGPL	Python	Translation
da te ga gok sa ‘create’ fa ‘table’	`Phrase.interrogative(Noun.doT(Pronoun.I(), DeterminerV.Suggestion(Verb("'create'")), Noun("'table'")))`	Shall I create a table?
te ga sa ‘get’ ma fa ‘information’ so te lan gi nu sa ‘create’ fa ‘table’	`Noun.doT(Pronoun.I(), Verb("'get'"),Noun.eq(Noun("'information'"), Verb.none(),Noun.doT(DeterminerN.male(Pronoun.he()),Verb.perfective(Verb("'create'")),Noun("'table'"))))`	I get the information that he has create a table.
di te ga sa ‘get’ ma fa ‘information’ so te lan gi nu sa ‘create’ fa ‘table’	`Phrase.past(Noun.doT(Pronoun.I(),Verb("'get'"),Noun.eq(Noun("'information'"),Verb.none(),Noun.doT(DeterminerN.male(Pronoun.he()),Verb.perfective(Verb("'create'")),Noun("'table'")))))`	I got the information that he has create a table.

SFGPL	Python	Translation
di te ga sa ‘get’ ma fa ‘information’ so te lan gi nu sa ‘create’ ma don fa ‘table’ so mal pul	`Phrase.past(Noun.doT(Pronoun.I(),Verb("'get'"),Noun.eq(Noun("'information'"),Verb.none(),Noun.doT(DeterminerN.male(Pronoun.he()),Verb.perfective(Verb("'create'")),Noun.eq(DeterminerN.plural(Noun("'table'")),Verb.none(),NumberList.digit1(Number.three()))))))`	I got the information that he has create three tables.
di moa ga so te lan gi sa ‘create’ fa ‘table’ fa ‘John’	`Phrase.past(Noun.hearSay(Pronoun.I(),Verb.none(),Noun.doT(DeterminerN.male(Pronoun.he()),Verb("'create'"),Noun("'table'")),Noun("'John'")))`	According to John, I heard that he create a table.
di moa ge so te lan gi sa ‘create’ fa ‘table’ fa ‘John’	`Phrase.past(Noun.hearSay(Pronoun.you(),Verb.none(),Noun.doT(DeterminerN.male(Pronoun.he()),Verb("'create'"),Noun("'table'")),Noun("'John'")))`	According to John, you heard that he create a table.

SFGPL	Python	Translation
di pa te ga sa ‘know’ di te ben we ni sa ‘call’ ga	`Phrase.past(LangObj.NOT(Noun.doT(Pronoun.I(),Verb("'know'"),Phrase.past(Noun.doT(DeterminerN.human(Pronoun.indefinite()),Verb.progressive(Verb("'call'")),Pronoun.I())))))`	I didn’t know that someone was calling me.
pe di pa ta ga na nak sa ‘go’ li pun ma go so san fa ‘event’ pa mi ga so fa ‘car’	`LangObj.Because(Phrase.past(LangObj.NOT(Noun.do(Pronoun.I(),Verb.add(DeterminerV.Possible(Verb("'go'")),Modifier.N2M(DeterminerN.place(Noun.eq(Pronoun.distal(),Verb.none(),DeterminerN.stressed(Noun("'event'")))))))),LangObj.NOT(Noun.have(Pronoun.I(),Verb.none(),Noun("'car'"))))`	I could not go to that event because I do not have a car.

SFGPL	Python	Translation
di to ge na so li pon di te ge soa ‘eat’ ‘English’ te ga soa ‘make’ ‘English’ san foa ‘cake’ ‘English’ fa ‘Mary’ loa ‘sad’ ‘English’	`Phrase.past(Noun.makeM (Pronoun.you(),Verb. add(Verb.none(), Modifier.N2M(DeterminerN.reason(Phrase.past(Noun.doT(Pronoun.you(),Verb. borrowed("'eat'","' English'"),Noun.doT(Pronoun.I(),Verb. borrowed("'make'","' English'"),DeterminerN .stressed(Noun. borrowed("'cake'","' English'")))))))),Noun ("'Mary'"),Modifier. borrowed("'sad'","' English'")))`	You made Mary sad, for you ate the cake I made.
di te ga na soa ‘meet’ ‘English’ li pin di te ga soa ‘go’ ‘English’ fi soa ‘shop’ ‘English’ fa ‘Mary’	`Phrase.past(Noun.doT( Pronoun.I(),Verb.add( Verb.borrowed("'meet'" ,"'English'"),Modifier .N2M(DeterminerN.time( Phrase.past(Noun.doT( Pronoun.I(),Verb. borrowed("'go'","' English'"),Noun.V2N( Verb.borrowed("'shop'" ,"'English'")))))), Noun("'Mary'")))`	I met Mary when I went shopping.

SFGPL	Python	Translation
di te ga na soa ‘meet’ ‘English’ li pin di te ga soa ‘go’ ‘English’ fi soa ‘shop’ ‘English’ fa ‘Mary’	`Phrase.past(Noun.doT(Pronoun.I(),Verb.add(Verb.borrowed("'meet'","'English'"),Modifier.N2M(DeterminerN.time(Phrase.past(Noun.doT(Pronoun.I(),Verb.borrowed("'go'","'English'"),Noun.V2N(Verb.borrowed("'shop'","'English'")))))),Noun("'Mary'")))`	I met Mary when I went shopping.
ta fa ‘apple’ na gen li pun ma gu so fa ‘table’	`Noun.do(Noun("'apple'"),Verb.add(WordV.exist(),Modifier.N2M(DeterminerN.place(Noun.eq(Pronoun.proximal(),Verb.none(),Noun("'table'"))))))`	There is an apple on this table.
ta don fa ‘apple’ na gen li pun ma gu so fa ‘table’	`Noun.do(DeterminerN.plural(Noun("'apple'")),Verb.add(WordV.exist(),Modifier.N2M(DeterminerN.place(Noun.eq(Pronoun.proximal(),Verb.none(),Noun("'table'"))))))`	There are apples on this table.
di ti ga so mi ga so don fa ‘student’ don fa ‘apple’	`Phrase.past(Noun.give(Pronoun.I(),Verb.none(),Noun.have(Pronoun.I(),Verb.none(),DeterminerN.plural(Noun("'student'"))),DeterminerN.plural(Noun("'apple'"))))`	I gave my students apples.

SFGPL	Python	Translation
di te ben we sa ‘eat’ fa ‘apple’	`Phrase.past(Noun.doT(DeterminerN.human(Pronoun.indefinite()),Verb("'eat'"),Noun("'apple'")))`	Someone ate an apple.
ta len gi na sa ‘sing’ la ‘beautifully’	`Noun.do(DeterminerN.female(Pronoun.he()),Verb.add(Verb("'sing'"),Modifier("'beautifully'")))`	She sings beautifully.
di ta don gi na sa ‘arrive’ la ‘late’	`Phrase.past(Noun.do(DeterminerN.plural(Pronoun.he()),Verb.add(Verb("'arrive'"),Modifier("'late'"))))`	They arrived late.
ta lan gi na sa ‘work’ la ‘hard’	`Noun.do(DeterminerN.male(Pronoun.he()),Verb.add(Verb("'work'"),Modifier("'hard'")))`	He works hard.
ta ma bin gi so san fa ‘cat’ ni sa ‘sleep’	`Noun.do(Noun.eq(DeterminerN.animal(Pronoun.he()),Verb.none(),DeterminerN.stressed(Noun("'cat'"))),Verb.progressive(Verb("'sleep'")))`	The cat is sleeping.
te ga sa ‘like’ fa ‘coffee’	`Noun.doT(Pronoun.I(),Verb("'like'"),Noun("'coffee'"))`	I like coffee.
ta len gi na sa ‘run’ la ‘fast’	`Noun.do(DeterminerN.female(Pronoun.he()),Verb.add(Verb("'run'"),Modifier("'fast'")))`	She runs fast.

SFGPL	Python	Translation
mi don ga so fa ‘plan’	`Noun.have(DeterminerN.plural(Pronoun.I()),Verb.none(),Noun("'plan'"))`	We have a plan.
te lan gi sa ‘play’ fa ‘guitar’	`Noun.doT(DeterminerN.male(Pronoun.he()),Verb("'play'"),Noun("'guitar'"))`	He plays the guitar.
te len gi sa ‘play’ fa ‘guitar’	`Noun.doT(DeterminerN.female(Pronoun.he()),Verb("'play'"),Noun("'guitar'"))`	She plays the guitar.
ta ma gi so san fa ‘phone’ ni sa ‘ring’	`Noun.do(Noun.eq(Pronoun.he(),Verb.none(),DeterminerN.stressed(Noun("'phone'"))),Verb.progressive(Verb("'ring'")))`	The phone is ringing.
ta mi ga so san fa ‘phone’ ni sa ‘ring’	`Noun.do(Noun.have(Pronoun.I(),Verb.none(),DeterminerN.stressed(Noun("'phone'"))),Verb.progressive(Verb("'ring'")))`	My phone is ringing.
ta ma lan gi so san fa ‘phone’ ni sa ‘ring’	`Noun.do(Noun.eq(DeterminerN.male(Pronoun.he()),Verb.none(),DeterminerN.stressed(Noun("'phone'"))),Verb.progressive(Verb("'ring'")))`	His phone is ringing.

SFGPL	Python	Translation
ta ma len gi so san fa ‘phone’ ni sa ‘ring’	`Noun.do(Noun.eq(DeterminerN.female(Pronoun.he()),Verb.none(),DeterminerN.stressed(Noun("'phone'"))),Verb.progressive(Verb("'ring'")))`	Her phone is ringing.
te don gi ni sa ‘watch’ fa ‘movie’	`Noun.doT(DeterminerN.plural(Pronoun.he()),Verb.progressive(Verb("'watch'")),Noun("'movie'"))`	They are watching a movie.

24. Dictionary

See dict.csv for details.

ID	word	func	How to use	Japanese	English
0	pa	`LangObj.NOT`	pa A	A でない	not A
1	pe	`LangObj.Because`	pe A B	A なぜならば B	A because B
2	pi	`LangObj.IF`	pi A B	もし A ならば B である	if A, B
3	pu	`LangObj.So`	pu A B	A だから B	A so B
4	po	`LangObj.But`	po A B	A しかし B	A but B
5	ba	`LangObj.AND`	ba A B	A かつ B	A and B
6	be	`LangObj.OR`	be A B	A または B	A or B

ID	word	func	How to use	Japanese	English
7	bi	LangObj.IFELSE	bi A B C	もし A ならば B である，そうでなければ C である	If A, B, otherwise C
8	bu	LangObj.NAND	bu A B	A かつ B でない	A nand B
9	bo	LangObj.NOR	bo A B	A または B でない	A nor B
10	ja	LangObj.logicIFELSE	ja A B C	もし A ならば B を出力，そうでなければ C を出力する	If A, output B, otherwise output C
11	fa	Noun	fa A	A は存在する	There be A / A exist
12	foa	Noun.borrowed	foa A B	A という名詞の単語を B という言語から借用する	Borrowing noun words called A from the language B
13	fi	Noun.V2N	fi A	動詞から名詞に変換する	Converting verbs to nouns.
14	fu	Noun.M2N	fu A	修飾語から名詞に変換する	Converting modifiers to nouns.
15	fo	Noun.none	fo	品詞が名詞の無意味の語を作る	The part of speech makes the noun nonsensical
16	ma	Noun.eq	ma A F B	A は B で (F) ある	A F(Verbs such that A means equal to B) B

ID	word	func	How to use	Japanese	English
17	me	Noun.haveP	me A F B	A が B という性質が (F) ある	A F(Verbs such that A means equal to B) B
18	mi	Noun.have	mi A F B	A は B を所有している/A は B を含んでいる	A have B/ A include B
19	mu	Noun.belong	mu A F B	A は B に所属している/A は B に含まれている	A belongs to B/ A is included in B
20	mo	Noun.gt	mo A F B C	A は C より（B という比較基準で）大きい	A is more B than C
21	moa	Noun.hearSay	moa A F B C	B という内容を C という情報源から，A は F する	A(Subject) F(Verb) that B(Content) according to C(Source)
22	ta	Noun.**do**	ta A F	A は F（〜する）	A F(do)
23	te	Noun.doT	te A F B	A は B を F（〜する）	A F(do) B
24	ti	Noun.give	ti A F B C	A は B に C を F（〜与える）	A F(give) B C
25	tu	Noun.makeN	tu A F B C	A は B を C の状態に F（〜する）	A F(make) B CNoun
26	to	Noun.makeM	to A F B C	A は B を C の状態に F（〜する）	A F(make) B CModifier

ID	word	func	How to use	Japanese	English
27	da	`Phrase.interrogative`	da A	A（〜ですか）[疑問]	A(interrogative form)
28	de	`Phrase.imperative`	de A	A（〜しろ）[命令]	A(imperative form)
29	di	`Phrase.past`	di A	A（〜した）[過去]	A(did)
30	du	`Phrase.future`	du A	A（〜するだろう/する予定である）[未来]	A(will do / be going to do)
31	sa	`Verb`	sa A	A（〜する）行為が存在する	There is an act of A
32	soa	`Verb.borrowed`	soa A B	A という動詞の単語を B という言語から借用する	Borrowing verb words called A from the language B
33	si	`Verb.M2V`	si A	修飾語から動詞に変換する	Converting from modifiers to verbs.
34	su	`Verb.N2V`	su A	名詞から動詞に変換する	Converting from nouns to verbs.
35	so	`Verb.none`	so	品詞が動詞の無意味の語を作る	The part of speech makes the verb nonsensical

ID	word	func	How to use	Japanese	English
36	na	Verb.add	na A B	A（〜する）に B（〜く/〜に）[副詞]という意味を付加する	Adding the meaning of B to A verb
37	ne	Verb.passive	ne A	A（〜される）[受動]	A(be done)
38	ni	Verb.progressive	ni A	A（〜している）[継続]	A(be doing)
39	nu	Verb.perfective	nu A	A（〜したことのある）[経験/完了/結果/継続]	A(have done)
40	la	Modifier	la A	A（〜な/〜の/〜に/〜く）[形容詞/副詞]という修飾語が存在する	There is a modifier [adjective/adverb] called A
41	loa	Modifier.borrowed	loa A B	A という修飾詞の単語を B という言語から借用する	Borrowing modifier words called A from the language B
42	li	Modifier.N2M	li A	A（〜の/〜に/〜で）	(of/ in/ at/ on/ by/ with etc.) A
43	lu	Modifier.V2M	lu A	動詞から修飾語に変換する	Converting verbs to modifiers.
44	lo	Modifier.none	lo	品詞が修飾語の無意味の語を作る	The part of speech makes the modifier nonsensical

ID	word	func	How to use	Japanese	English
45	ka	`Modifier.add`	ka A B	A に B[副詞]という意味を付加する	Adding the meaning of B to A modifier
46	ke	`Modifier.Neg`	ke A	A でなく	not A
47	ki	`Modifier.Very`	ki A	とても A である	very A
48	pan	`DeterminerN.biology`	pan A	名詞を「A が何らかの人や生物」と限定する	Limit nouns to 'A is some kind of person or creature'
49	pen	`DeterminerN.thing`	pen A	名詞を「A が何らかの物や概念」と限定する	Limit nouns to 'A is some object or concept'
50	pin	`DeterminerN.time`	pin A	名詞を「A が何らかの時間」と限定する	Limit a noun to 'A is some time'
51	pun	`DeterminerN.place`	pun A	名詞を「A が何らかの場所」と限定する	Limit a noun to 'A is some place'
52	pon	`DeterminerN.reason`	pon A	名詞を「A が何らかの理由」と限定する	Limit a noun to 'A is some reason'
53	ban	`DeterminerN.method`	ban A	名詞を「A が何らかの方法や道具や手段」と限定する	Limit nouns to 'A is some way or tool or means'

ID	word	func	How to use	Japanese	English
54	ben	DeterminerN.human	ben A	名詞を「A が何らかの人間」と限定する	Limit nouns to 'A is some kind of human being'
55	bin	DeterminerN.animal	bin A	名詞を「A が何らかの動物」と限定する	Limit nouns to 'A is some kind of animal'
56	bun	DeterminerN.plant	bun A	名詞を「A が何らかの植物」と限定する	Limit the noun to 'A is some kind of plant'
57	bon	DeterminerN.material	bon A	名詞を「A が何らかの材料」と限定する	Limit the noun to 'A is some kind of material'
58	fan	DeterminerN.start	fan A	名詞を「A が何らかの始点」と限定する	Limit a noun to 'A is some starting point'
59	fen	DeterminerN.end	fen A	名詞を「A が何らかの終点」と限定する	Limit a noun to 'A is some end point'
60	fin	DeterminerN.section	fin A	名詞を「A が何らかの区間」と限定する	Limit a noun to 'A is some interval'
61	fun	DeterminerN.In	fun A	名詞を「A が何らかの中」と限定する	Limit nouns to 'A is in some'
62	fon	DeterminerN.Out	fon A	名詞を「A が何らかの外」と限定する	Limit nouns to 'A is out some'

ID	word	func	How to use	Japanese	English
63	man	DeterminerN.above	man A	名詞を「A が何らかの上」と限定する	Limit nouns to ‘A above some’
64	men	DeterminerN.below	men A	名詞を「A が何らかの下」と限定する	Limit nouns to ‘A is below some’
65	min	DeterminerN.on	min A	名詞を「A が何らかに接地している」と限定する	Limit nouns to ‘A is grounded to something’
66	mun	DeterminerN.right	mun A	名詞を「A が何らかの右」と限定する	Limit nouns to ‘A is some right’
67	mon	DeterminerN.left	mon A	名詞を「A が何らかの左」と限定する	Limit nouns to ‘A is some left’
68	tan	DeterminerN.affect	tan A	名詞を「A が何らかの影響を与えるものや関連していること」と限定する	Limit the noun to ‘something that A affects or is related to in some way’
69	ten	DeterminerN.affected	ten A	名詞を「A が何らかの影響が与えられるもの」と限定する	Limit noun to ‘something that A is affected by in some way’
70	tin	DeterminerN.near	tin A	名詞を「A が近くにあるものや関連しているもの」と限定する	Limit the noun to ‘something that A is near or related to’.

ID	word	func	How to use	Japanese	English
71	tun	DeterminerN.move	tun A	名詞を「A が横切る」や「A が通る」「A が向かう」と動きのあるものに限定する	Limit nouns to those with motion, such as ‘A crosses’, ‘A passes’ or ‘A heads’.
72	ton	DeterminerN.stop	ton A	名詞を静止のあるものに限定する	Limit nouns to those with static
73	dan	DeterminerN.all	dan A	名詞を「すべての A」と限定する	Limit the noun to ‘all A’
74	den	DeterminerN.many	den A	名詞を「多くの A」と限定する	Limit the noun to ‘many A’, ‘much A’ or ‘a lot of A’
75	din	DeterminerN.some	din A	名詞を「いくつかの A」と限定する	Limit the noun to ‘some A’, ‘a few A’ or ‘several A’
76	dun	DeterminerN.one	dun A	名詞を「ある（一つの）A」と限定する	Limit the noun to ‘a certain A’, ‘one certain A’.
77	don	DeterminerN.plural	don A	名詞を複数形にする	Making nouns plural
78	san	DeterminerN.stressed	san A	名詞を強調形する	Stressed form of nouns.
79	sen	DeterminerN.possessive	sen A	所有代名詞を作成する	Creating possessive pronouns

ID	word	func	How to use	Japanese	English
80	sin	DeterminerN.reflexive	sin A	再帰代名詞を作成する	Creating recursive pronouns
81	sun	DeterminerN.etc	sun A	名詞を「A など」と限定する	Limit nouns to 'A etc.'
82	son	DeterminerN.**abstract**	son A	～的/～のようなもの/大体の～/おおよそ～ぐらい/名詞を抽象化する	something like A/ Abstracting nouns
83	nan	DeterminerN.front	nan A	名詞を「A が何らかの空間的に前」と限定する	Limit nouns to 'A is in front of some space'
84	nen	DeterminerN.behind	nen A	名詞を「A が何らかの空間的に後ろ」と限定する	Limit nouns to 'A is behind in some space'
85	nun	DeterminerN.future	nun A	名詞を「A が何らかの時間的に未来」と限定する	Limit nouns to 'A is some time in the future'
86	non	DeterminerN.past	non A	名詞を「A が何らかの時間的に過去」と限定する	Limit nouns to 'A is some time in the past'
87	lan	DeterminerN.male	lan A	名詞を「A が何らかの男性や雄」と限定する	Limit noun to 'A is some male'

ID	word	func	How to use	Japanese	English
88	len	DeterminerN.female	len A	名詞を「A が何らかの女性や雌」と限定する	Limit the noun to ‘A is some female’
89	lin	DeterminerN.every	lin A	名詞を「A があらゆる何らかのもの」と限定する	Limit the noun to ‘A is every something’
90	lun	DeterminerN.each	lun A	名詞を「A がそれぞれの何らかのもの」と限定する	Limit the noun to ‘A is each something’
91	lon	DeterminerN.other	lon A	名詞を「A が他の何らかのもの」と限定する	Limit the noun to ‘A is other something’
92	pak	DeterminerV.Estimation10	pak A	100%の確率で A する	100% probability A
93	pek	DeterminerV.Estimation75	pek A	75%の確率で A する	75% probability A
94	pik	DeterminerV.Estimation50	pik A	50%の確率で A する	50% probability A
95	puk	DeterminerV.Estimation25	puk A	25%の確率で A する	25% probability A

ID	word	func	How to use	Japanese	English
96	pok	DeterminerV.Estimation0	pok A	0%の確率でAする	0% probability A
97	fak	DeterminerV.Frequency100	fak A	100%ぐらいの頻度でAする	100% frequently A
98	fek	DeterminerV.Frequency75	fek A	75%ぐらいの頻度でAする	75% frequently A
99	fik	DeterminerV.Frequency50	fik A	50%ぐらいの頻度でAする	50% frequently A
100	fuk	DeterminerV.Frequency25	fuk A	25%ぐらいの頻度でAする	25% frequently A
101	fok	DeterminerV.Frequency0	fok A	0%ぐらいの頻度でAする	0% frequently A
102	tak	DeterminerV.Start	tak A	Aし始める	Someone starts doing something
103	tek	DeterminerV.Condition	tek A	Aしている途中である	Someone is in the middle of doing something

ID	word	func	How to use	Japanese	English
104	tik	DeterminerV.Complete	tik A	A している途中だったが完了した	Someone was in the middle of doing something but has completed
105	tuk	DeterminerV.Continue	tuk A	A している状態が続いている	Someone is still doing something
106	tok	DeterminerV.End	tok A	A し終える	Someone finishes doing something
107	bak	DeterminerV.past	bak A	過去には A であった	In the past it was A
108	bik	DeterminerV.present	bik A	現在 A である	In the present it is A
109	bok	DeterminerV.future	bok A	未来には A だろう	In the future it will be A
110	nak	DeterminerV.Possible	nak A	A できる/A することが可能である	can
111	nek	DeterminerV.Ability	nek A	A する能力がある	can
112	nik	DeterminerV.Will	nik A	A しよう	will/ shall
113	nuk	DeterminerV.Obligation	nuk A	A すべきだ	should/ ought to
114	nok	DeterminerV.Necessary	nok A	A する必要がある	need to
115	lak	DeterminerV.Duty	lak A	A しなければならない	must/ have to

ID	word	func	How to use	Japanese	English
116	lek	DeterminerV.forced	lek A	外部からの強い力で強制的に A させられる	be forced to A by a strong external force
117	lik	DeterminerV.want	lik A	A したい/A することを願望する	want to A
118	luk	DeterminerV.dare	luk A	あえて A する/思い切って A する/A する勇気がある	dare A
119	lok	DeterminerV.allow	lok A	A することを許す	allow to A
120	kak	DeterminerV.easy	kak A	A しやすい	be easy to A
121	kek	DeterminerV.hard	kek A	A しにくい	be hard to A
122	kik	DeterminerV.habit	kik A	習慣的に A する	Habitually A
123	kuk	DeterminerV.Polite	kuk A	A します（丁寧表現）	Make the verb a polite expression
124	kok	DeterminerV.Respect	kok A	A される（尊敬表現）	Make the verb a respectful expression
125	gak	DeterminerV.volitional	gak A	意識的に A する	Consciously A
126	gek	DeterminerV.nonVolitiona	gek A	無意識的に A する	Unconsciously A

ID	word	func	How to use	Japanese	English
127	gik	DeterminerV.Requests	gik A	A してください	will/ would/ can/ could
128	guk	DeterminerV.Permission	guk A	A してもよいですか	can/ may
129	gok	DeterminerV.Suggestion	gok A	A しましょうか	shall
130	ga	Pronoun.I	ga	私	I
131	ge	Pronoun.you	ge	あなた	you
132	gi	Pronoun.he	gi	彼/彼女/それ	he/ she/ it
133	gu	Pronoun.proximal	gu	これ	this
134	go	Pronoun.distal	go	それ/あれ	that
135	wa	Pronoun.interrogative	wa	どれ	what
136	we	Pronoun.indefinite	we	どれか	something
137	kan	WordV.create	kan	生み出す/作る/産む	create/ make/ bear
138	ken	WordV.destroy	ken	破壊する/壊す/死ぬ	destroy/ break/ die
139	kin	WordV.act	kin	行動する/動く/実行する/歩く/働く	act/ move/ do/ walk/ work
140	kun	WordV.turn	kun	回る/回転する/急ぐ/走る	turn/ rotate/ hurry/ run

ID	word	func	How to use	Japanese	English
141	kon	WordV.receive	kon	感じ取る/受信する/受け取る/入れる/摂取する/取得する/得る/習う/聞く/見る/食べる/飲む	receive/ accept/ acquire/ get/ learn/ hear/ see/ listen/ look at/ watch/ eat/ drink
142	gan	WordV.stimulate	gan	発する/発信する/発射する/出す/送信する/送る/教える/刺激する/言う/話す/攻撃する	emit/ transmit/ put out/ send/ give/ teach/ stimulate/ say/ speak/ attack
143	gen	WordV.exist	gen	ある/いる/存在する/生きている/住んでいる/留まる/止まっている/休む	be/ exist/ live/ stay/ be stopping/ get rest
144	gin	WordV.use	gin	使う/使用する	use
145	gun	WordV.change	gun	変わる/なる/成長する/移行する/移動する	change/ become/ grow/ transfer
146	wan	WordM.big	wan	大きい/長い/広い/高い/多い/重い	big/ long/ wide/ tall/ many/ heavy/ large
147	wen	WordM.near	wen	近い/親しい/似ている/好きである	near/ familiar/ close to/ similar/ like

ID	word	func	How to use	Japanese	English
148	win	`WordM.good`	win	良い/新しい/若い/美しい	good/ new/ young/ beautiful
149	won	`WordM.bright`	won	明るい/白い/色鮮やかな	bright/ white/ colourful
150	pas	`Bool.false`	pas	偽	False (Boolean)
151	pos	`Bool.true`	pos	真	True (Boolean)
152	pis	`Bool.B2N`	pis A B	A は B である（B は真偽）	A is B (B is true or false)
153	fas	`BoolList`	fas	真偽のリスト（BoolList）を作成する	Create a list of true/ false (BoolList)
154	fes	`BoolList.get`	fes A B	`BoolList(A)` の B 番目の値を取得する	Gets the B-th value of `BoolList(A)`
155	fis	`BoolList.append`	fis A B	BoolList に 1 つの Bool を末尾に加える	Add one Bool to the end of the BoolList
156	fus	`BoolList.slice`	fus A B C	A という BoolList に対して，B 番目から C 番目までのリストを取得する	Get the B-th through C-th lists for a BoolList (A).
157	fos	`BoolList.add`	fos A B	2 つの BoolList を結合する	Combine two BoolLists
158	foas	`BoolList.len`	foas A	BoolList の長さを取得する	Get the length of the BoolList

ID	word	func	How to use	Japanese	English
158	mas	`BoolList.twoBit`	mas A B	2 つ Bool の値からなる BoolList を作成する	Create a BoolList consisting of 2 Bool values
159	mis	`BoolList.fourBit`	mis A B C D	4 つ Bool の値からなる BoolList を作成する	Create a BoolList consisting of 4 Bool values
160	mos	`BoolList.byte`	mos X1 X2 X3 X4 X5 X6 X7 X8	8 つ Bool の値からなる BoolList を作成する	Create a BoolList consisting of 8 Bool values
161	tas	`BoolList.NaturalNum`	tas A	BoolList を 2 進数の自然数とみなす	BoolList is considered a binary natural number
162	tes	`BoolList.Int`	tes A	BoolList を 2 進数の整数とみなす	BoolList is considered a binary integer
163	tis	`BoolList.Float`	tis A	BoolList を 2 進数の浮動小数とみなす	BoolList is considered a binary floating number
164	tus	`BoolList.ASCII`	tus A	BoolList を ASCII 文字とみなす	BoolList is considered an ASCII character
165	tos	`BoolList.IntBL2NL`	tos A	整数の BoolList を NumberList に変換する	Convert an integer BoolList to a NumberList

ID	word	func	How to use	Japanese	English
166	das	`BoolList.UnixTimeD`	das A	BoolList を日単位の UnixTime とする	BoolList as UnixTime in days
167	des	`BoolList.UnixTimeDT`	des A	BoolList を秒単位の UnixTime とする	BoolList as UnixTime in seconds
168	dis	`BoolList.UnixTimeDTN`	dis A	BoolList をナノ秒単位の UnixTime とする	BoolList as UnixTime in nanoseconds
169	dos	`BoolList.FloatBL2NL`	dos A	浮動小数点の実数の BoolList を NumberList に変換する	Convert an Floating-point real numbers BoolList to a NumberList
169	pat	`LangFunc.setFunc`	pat A B	ある LangList を引数とする A という名前の B を返す関数を設定する	Set up a function that returns B named A with a certain LangList as an argument.
170	pit	`LangFunc.arg`	pit	`LangFunc.setFunc()` の引数用に使用する	Used for `LangFunc.setFunc()` arguments
171	pot	`LangFunc.runFunc`	pot A B	設定した A という名前の LangFunc を引数 B として実行する	Execute the configured LangFunc named A with argument B

ID	word	func	How to use	Japanese	English
172	bat	LangVar.set	bat A B	グローバル変数として A という名前の変数を定義し，LangList B を代入する	Define a variable named A as a global variable and assign LangList B to it.
173	bot	LangVar.get	bot A	定義された A という名前のグローバル変数を取得する	Obtain the defined global variable named A
174	fat	LangList	fat	LangObj のリスト LangList を作成する	Create a list of LangObj (LangList)
175	fet	LangList.get	fet A B	LangList(A) の B 番目の値を取得する	Gets the B-th value of LangList(A)
176	fit	LangList.append	fit A B	LangList に 1 つの LangObj を末尾に加える	Add one LangObj to the end of the LangList
177	fut	LangList.slice	fut A B C	A という LangList に対して，B 番目から C 番目までのリストを取得する	Get the B-th through C-th lists for a LangList (A).
178	fot	LangList.add	fot A B	2 つの LangList を結合する	Combine two LangLists

ID	word	func	How to use	Japanese	English
179	foat	`LangList.len`	foat A	LangList の長さを取得する	Get the length of the LangList
180	tat	`LangList.While`	tat A B C	繰り返し処理を行う	Repeat processing
181	tet	`LangList.map`	tet A B	LangList A のすべての要素に対して，B という名前の LangFunc を適用する	Apply a LangFunc named B to all elements of LangList A
182	pal	`Number.zero`	pal	0	0
183	pel	`Number.one`	pel	1	1
184	pil	`Number.two`	pil	2	2
185	pul	`Number.three`	pul	3	3
186	pol	`Number.four`	pol	4	4
187	bal	`Number.five`	bal	5	5
188	bel	`Number.six`	bel	6	6
189	bil	`Number.seven`	bil	7	7
190	bul	`Number.eight`	bul	8	8
191	bol	`Number.nine`	bol	9	9
192	fal	`NumberList`	fal	Number のリスト NumberList を作成する	Create a list of Number (NumberList)

ID	word	func	How to use	Japanese	English
193	fel	NumberList.get	fel A B	NumberList(A) の B 番目の値を取得する	Gets the B-th value of NumberList(A)
194	fil	NumberList.append	fil A B	NumberList に 1 つの Number を末尾に加える	Add one Number to the end of the NumberList
195	ful	NumberList.slice	ful A B C	A という NumberList に対して，B 番目から C 番目までのリストを取得する	Get the B-th through C-th lists for a NumberList (A).
196	fol	NumberList.add	fol A B	2 つの NumberList を結合する	Combine two NumberLists
197	foal	NumberList.len	foal A	NumberList の長さを取得する	Get the length of the NumberList
198	mal	NumberList.digit1	mal A	10 進数 1 桁からなる NumberList を作成する	Create a NumberList consisting of one decimal digit
199	mel	NumberList.digit2	mel A B	10 進数 2 桁からなる NumberList を作成する	Create a NumberList consisting of two decimal digit

ID	word	func	How to use	Japanese	English
200	mil	`NumberList.digit3`	mil A B C	10 進数 3 桁からなる NumberList を作成する	Create a NumberList consisting of three decimal digit
201	mul	`NumberList.digit4`	mul A B C D	10 進数 4 桁からなる NumberList を作成する	Create a NumberList consisting of four decimal digit
202	mol	`NumberList.digit5`	mol A B C D E	10 進数 5 桁からなる NumberList を作成する	Create a NumberList consisting of five decimal digit
203	tal	`NumberList.calcAdd`	tal A B	2 つの NumberList に対して加算をする	Perform addition on two NumberLists
204	tel	`NumberList.calcSub`	tel A B	2 つの NumberList に対して減算をする	Perform subtraction on two NumberLists
205	til	`NumberList.calcMul`	til A B	2 つの NumberList に対して乗算をする	Perform multiplication on two NumberLists
206	tul	`NumberList.calcDiv`	tul A B	2 つの NumberList に対して除算をする	Perform division on two NumberLists

ID	word	func	How to use	Japanese	English
207	tol	NumberList.IntNL2BL	tol A	整数の NumberList を BoolList に変換する	Convert an integer NumberList to a BoolList
208	dal	NumberList.calcPow	dal A B	2 つの NumberList に対して累乗をする	Performs a power over two NumberLists
209	del	NumberList.calcIntDiv	del A B	2 つの NumberList に対して整数除算をする	Perform integer division on two NumberLists
210	dil	NumberList.calcMod	dil A B	2 つの NumberList に対して剰余をする	Performs remainder with respect to two NumberLists
211	dol	NumberList.FloatNL2BL	dol A	浮動小数点の実数の NumberList を BoolList に変換する	Convert a Floating-point real numbers NumberList to a BoolList
212	sal	NumberList.isPN	sal A	正の数かを判定する	Determine if it is a positive number
213	sel	NumberList.minus	sel A	符号を反転させる	Reversing the sign
214	sil	NumberList.abs	sil A	整数の絶対値を取得する	Obtaining the absolute value of an integer

25. About version

The version of this project is __version__.py. In particular, if you want to run it in Python, you can check it by executing the following code.

```
SFGPL.__version__.__version__
```

In addition, the version of the corpus at the time it was executed is listed in the JSON file of the corpus output by `SFGPL.SFGPLCorpus.saveJson` of Python code.

25.1. Version naming conventions

The SFGPL uses and manages versions like `A.B.C`. The content of updates due to changes in version names is based on the following table.

Version	Update	Contents
A	Main update	When there are major changes to words, programs, etc.
B	Minor update	When there are small changes to words, programs, etc.
C	Patch update	When there are small changes or changes in the documentation due to bug fixes in the program etc.

25.2. Version update details

Version	Update contents
1.0.0	Official Release
1.0.1	Add or modify example sentences
1.0.2	Add or modify example sentences
1.0.3	Addition of details of updates per version
1.1.0	Added details on how to use SFGPL in Python
1.1.1	How_to_Use_SFGPL_in_Python.ipynb fixed

Version	Update contents
2.0.0	Add classes for logical values
2.0.1	Add and modify to Python programs
2.0.2	Add and modify to documents
2.1.0	Add BoolList.get() and BoolList.slice()
3.0.0	Add LangList and LangFunc classes
3.0.1	How_to_Use_SFGPL_in_Python.ipynb fixed
3.1.0	Fixed LangFunc.runFunc()
3.1.1	Add and modify to documents
3.1.2	Add and modify to documents
3.1.3	Add and modify to documents
4.0.0	Add DeterminerV class
4.0.1	Fixed dictionary
4.0.2	Add and modify to documents
4.0.3	Add and modify to documents
4.0.4	Add and modify to documents
4.0.5	Add and modify to documents
4.0.6	Add and modify to documents
4.0.7	Add and modify to documents
4.0.8	Add and modify to documents
4.0.9	Add and modify to documents
4.0.10	Add and modify to documents
4.0.11	Add and modify to documents
4.0.12	Add and modify to documents
4.0.13	Add and modify to documents
4.1.0	Add Noun.hearSay()
4.1.1	Fixed dictionary
4.1.2	Add and modify to documents

Version	Update contents
4.1.3	Add and modify to documents
5.0.0	Add Number and NumberList classes
5.0.1	Add and modify to documents
5.0.2	Add and modify to documents
5.0.3	Add and modify to documents
5.0.4	Add and modify to documents
5.0.5	Add and modify to documents
5.0.6	Add and modify to documents
5.0.7	Add and modify to documents
5.0.8	Add and modify to documents
5.0.9	Add and modify to documents
5.0.10	Add and modify to documents
5.0.11	Add and modify to documents
5.0.12	Add and modify to documents
5.0.13	Add and modify to documents
5.0.14	Add and modify to documents
5.0.15	Add and modify to documents
5.0.16	Add and modify to documents
5.0.17	Add and modify to documents
5.0.18	Add and modify to documents
5.1.0	Add LangObj.logicIFELSE() and NumberList.isPN()
5.1.1	Add and modify to documents
5.1.2	Add and modify to documents
5.1.3	Add and modify to documents
5.1.4	Add and modify to documents
5.1.5	Add and modify to documents
5.1.6	Add and modify to documents

Version	Update contents
5.1.7	Add and modify to documents
5.1.7	Add and modify to documents
5.2.0	Add dictionary to documentation
5.2.1	Add and modify to documents
5.3.0	Add SFGPLLib.checkType()
5.3.1	Add and modify to documents
6.0.0	Add LangVar
6.1.0	Add functions related to the structuring of the SFGPL
6.1.1	Adds and modifies documentation and SFGPL.py
7.0.0	Adding words and associated libraries
7.1.0	List length functions in list-related classes added
7.2.0	Addition of `LangList.map`
7.2.1	Add and modify to documents
7.2.2	Add and modify to documents
7.2.3	Add and modify to documents
7.3.0	Add a Unix time and various floating point numbers in BoolList
7.3.1	Add and modify to documents
7.3.2	Add and modify to documents
7.4.0	Additional floating point conversions between BoolList and NumberList, additional operation types for NumberList
7.4.1	Add and modify to documents
7.4.2	Adds and modifies documentation and SFGPL.py
7.4.3	Adds and modifies documentation and SFGPL.py

Version	Update contents
7.4.4	Add and modify to documents