Introduction to the SFGPL

Eruhitsuji

2023-12-07

Contents

1 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.

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	
-	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/
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/

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 and NumberList exist as subclasses of Noun.

BoolList, LangList, and LangFunc 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.

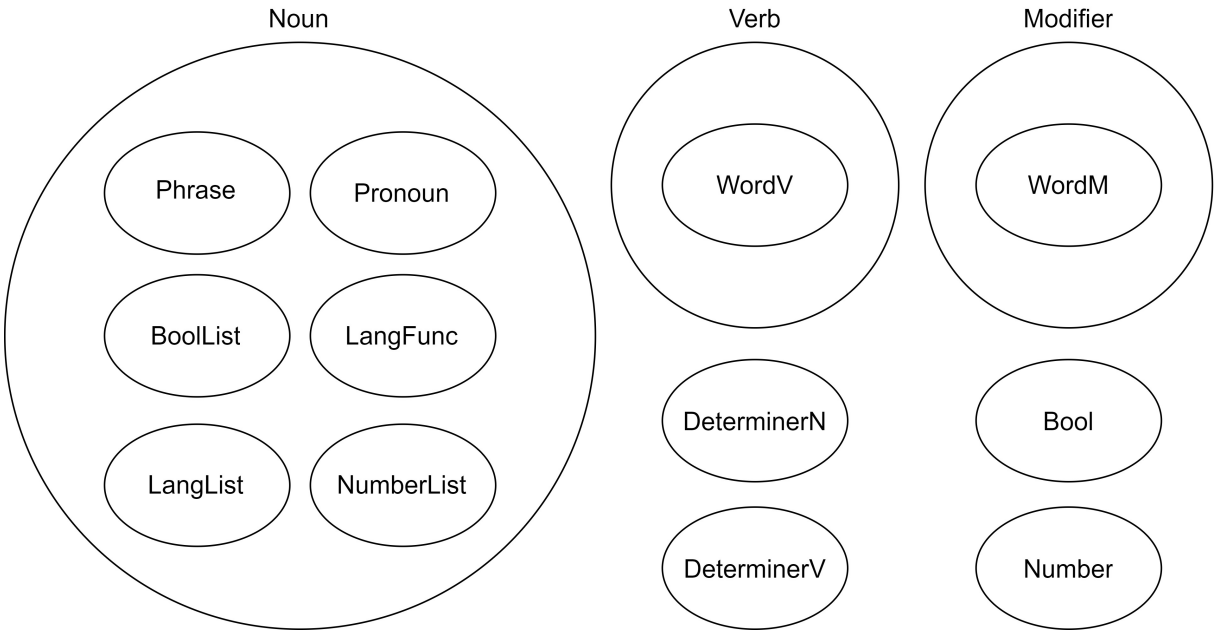

Figure 1: PartOfSpeach

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 2. Sentence Pattern

2.1 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)

2.2 Noun.do

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.

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

2.3 Noun.eq

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.

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

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

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

2.4 Noun.haveP

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.

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

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

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

2.5 Noun.doT

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.

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

2.6 Noun.give

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.

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

2.7 Noun.makeN and Noun.makeM

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.

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

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

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

2.8 Noun.have

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.

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

2.9 Noun.belong

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.

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

2.10 Noun.gt

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.

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

2.11 Noun.hearSay

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.

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

2.12 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.

```
1 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.

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

2.12.1 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.”.

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

2.13 Wordbook

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

3 3. Negative Sentence

Use `pa` to create a 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 make it mean “I don’t have a table.”, it can be expressed as follows in the SFGPL.

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

3.1 Wordbook

English	SFGPL
I	ga
table	fa ‘table’

4 4. Interrogative Sentence

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.

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

In the case of interrogative sentences containing interrogatives, the indefinite is expressed by replacing the indefinite with an interrogative.

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

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

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

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

4.1 Wordbook

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

5 5. 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'
```

5.1 Wordbook

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

6 6. Compound sentences

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

6.1 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
```

6.2 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.

6.2.1 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.

6.2.2 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'
```

6.3 Wordbook

English	SFGPL
I	ga

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

7 7. 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.

7.1 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.

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

Another alternative is to use me.

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

7.1.1 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
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

7.2 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.

7.2.1 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.

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

7.2.2 Superlative

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

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

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

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

7.2.3 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 'big' sen lan gi so ka wan wen
```

7.3 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

8 8. 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.

8.1 Borrowed Words

The SFGPL uses loan words for all but the basic words. 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’

8.2 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

8.3 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.

8.3.1 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

8.3.2 DeterminerV

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

8.4 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

8.5 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

8.6 Words used numerically and logically

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

9 9. Verb Conjugation

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

9.1 Verb tenses

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

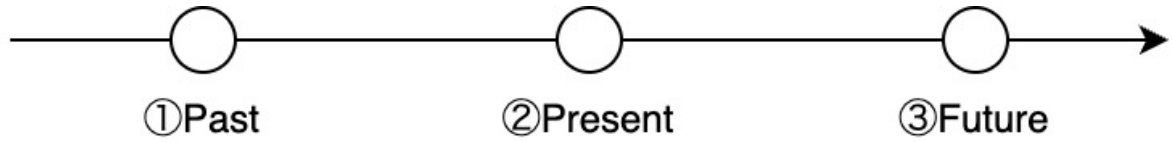

Figure 2: BasingPoint

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.

9.1.1 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

9.2 Phases

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

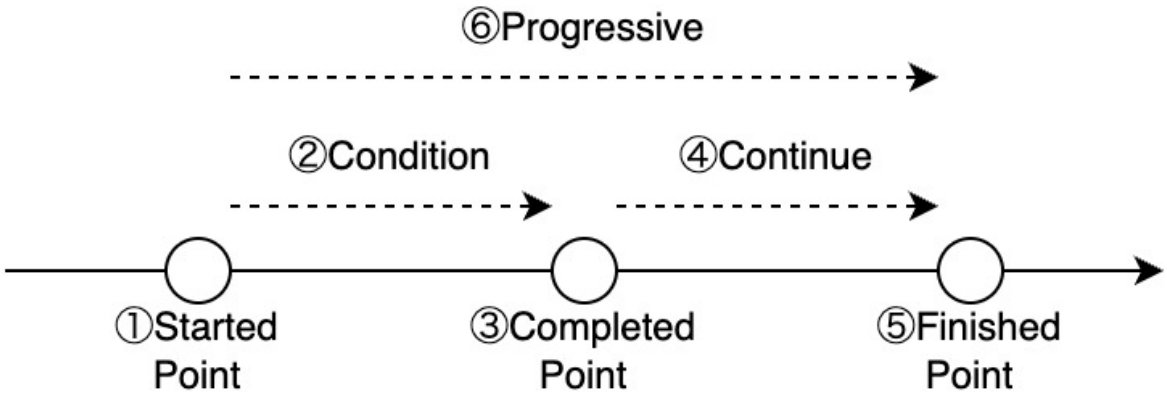

Figure 3: ProgressiveForm

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

Phase	Word	English	SFGPL
① Start Phase	tak	I begin wear a dress.	te ga tak sa 'wear' fa 'dress'
② Transitional Phase	tek	I am (in the process of) wearing a dress.	te ga tek sa 'wear' fa 'dress'
③ Completion Phase	tik	I wear a dress. (I just finished wearing it.)	te ga tik sa 'wear' fa 'dress'
④ Continuation Phase	tuk	I am wearing a dress. (The state in which it is worn.)	te ga tuk sa 'wear' fa 'dress'
⑤ Finish Phase	tok	I finish wear a dress. (I stopped wearing it.)	te ga tok sa 'wear' fa 'dress'
⑥ Progression Phase	ni	I am wearing a dress.	te ga ni sa 'wear' fa 'dress'

These phases can be in the past or future tense in addition to the present tense. ⑥ Progressive phase includes ② transitional phase and ④ continuation phase. There are also cases where ③ the completion phase and ⑤ the finish phase are the same. "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, a phase alone does not indicate a range of time, but only the moment in which it occurs. When expressing a range of time, the perfect tense is added. The progressive form plus the perfect form to express "I have been wearing a dress.".

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

9.2.1 General progressive form

In SFGPL, we can make a simple progressive form as in ⑥ without considering the phases 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'
```

9.3 Perfect tense

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

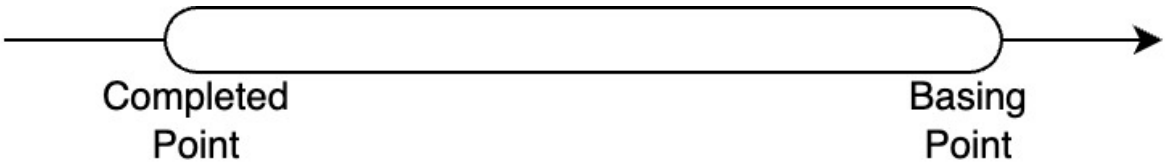

Figure 4: PerfectForm

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.

9.4 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'
```

9.5 Other verb modifiers

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

9.6 Wordbook

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

10 10. Modifier

10.1 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”.

10.2 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.

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

10.3 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

10.4 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.

10.5 Wordbook

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

11 11. 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.

11.1 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.

11.2 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'
```

11.3 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'
```

11.4 Wordbook

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

12 12. 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’
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'`.

12.1 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’

13 13. Pronoun

13.1 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
Indefinite Pronoun	something	we

13.2 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.

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

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

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

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

14 14. 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’.

14.1 Wordbook

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

15 15. 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’
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.

15.1 Wordbook

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

16 16. 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.

16.1 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.

```
1 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.

16.2 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)
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
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

Type	SFGPL	Value
Integer Number	tes fos fos mos pas pos 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 mos pas pos pas pas pos pas pas pos fos mos pas pas pas pas pos pos pos pos mos pos pos pas pos pas pos pos	3.1415927410125732

16.3 Wordbook

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

17 17. 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

LangList can store all classes that inherit from LangObj. The following is an example of LangList creation.

```
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
```

17.1 Wordbook

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

18 18. 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
```

19 19. How numbers are expressed

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

19.1 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
8	bul
9	bol

19.2 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

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.

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

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

```
1 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
```

Then, as shown in the following table, there are functions in NumberList that perform the four arithmetic operations.

	SFGPL
Addition	tal
Subtraction	tel
Multiplication	til
Division	tul

In addition, there are functions that convert integer BoolList and NumberList into each other, as shown in the table below.

SFGPL	from	to
tol	NumberList	BoolList
tos	BoolList	NumberList

19.3 Wordbook

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

20 20. 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.
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 an office worker.

SFGPL	Python	Translation
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 an office worker.
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 an office worker.
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.

SFGPL	Python	Translation
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.
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.

SFGPL	Python	Translation
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.
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.

SFGPL	Python	Translation
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.
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?

SFGPL	Python	Translation
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?
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.

SFGPL	Python	Translation
du te ga sa ‘create’ fa ‘table’	Phrase.future(Noun.doT(Pronoun.I( ) , Verb(“ ‘create’ ”) , Noun(“ ‘table’ ”)))	I will create a table.
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.
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.

SFGPL	Python	Translation
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)
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)

SFGPL	Python	Translation
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.
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.

SFGPL	Python	Translation
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.
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)

SFGPL	Python	Translation
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?
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.

SFGPL	Python	Translation
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.
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.

21 21. 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.

21.1 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.

21.2 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
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()

Version	Update contents
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
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

Version	Update contents
5.0.6	Add and modify to documents