# HG7021 Write-up

## Introduction

Singlish is largely similar to Standard English with its similar basic word order and the majority of the lexicon used. However, there are also many differences and this small grammar of Singlish was made to reflect both sorts of features.

The initial grammar was generated using the Matrix by Emily Bender. Through the input of different basic features of the language, a foundation of the grammar was created. It was then enhanced by various tweaks and many additions to produce a small grammar as the final product of this module.

## Initial features from matrix

* Word order: SVO
* Number: singular, plural
* Gender: -not used-
* Case: nominative, accusative
* Tense: past and non-past
* Aspect: experiential and perfective (not required by grammar)
* FORMS:
  + Finite
  + Nonfinite
    - Present perfect
    - Base
* Sentential negation: single morpheme – “never”
* Coordination: Monosyndeton – “and”
* Matrix Y/N questions: sentence-final “is it” (but could be made to allow sentence-initial as well)
* Information structure: -not used-
* Morphology:
  + Number: plural suffix “-s” (plural-lex-rule)
  + PNG: 3sg suffix “-s” (3sg-lex-rule)
* Lexicon:

|  |  |  |
| --- | --- | --- |
| Stem | Predicate | Comments/example |
| **nouns** | | |
| Cat | \_cat\_n\_1\_rel |  |
| Ant | \_ant\_n\_1\_rel |  |
| Tree | \_tree\_n\_1\_rel |  |
| I | \_I\_n\_1\_rel |  |
| He | \_he\_n\_1\_rel |  |
| They | \_they\_n\_1\_rel |  |
| It | \_it\_n\_1\_rel |  |
| We | \_we\_n\_1\_rel |  |
| Me | \_me\_n\_1\_rel | Subsequently changed to \_I\_n\_1\_rel |
| Him | \_him\_n\_1\_rel | Subsequently changed to \_he\_n\_1\_rel |
| You | \_you\_n\_1\_rel |  |
| Us | \_us\_n\_1\_rel | Subsequently changed to \_we\_n\_1\_rel |
| **Verbs** | | |
| Tekan | \_bully\_v\_rel |  |
| Kacau | \_disturb\_v\_rel |  |
| Sleep | \_sleep\_v\_rel |  |
| Chase | \_chase\_v\_rel |  |
| Think | \_think\_v\_1\_rel | Transitive. But could not form any right sentences at this stage. |
| Think | \_think\_v\_2\_rel | intransitive |
| **Auxiliaries** | | |
| Will | \_will\_v\_rel |  |
| Can | \_can\_v\_rel |  |
| **Determiners** | | |
| The | \_the\_q\_rel |  |
| Those | \_those\_q\_rel |  |
| That | \_that\_q\_rel |  |
| This | \_this\_q\_rel |  |
| These | \_these\_q\_rel |  |
| **Others** | | |
| Is it | - |  |
| Never | neg\_rel |  |
| And | \_and\_coord\_rel |  |

After a few sessions with the matrix, the grammar could parse simple sentences such as “the cat and I sleep” and “Ant never eat the tree.” The nominative and accusative pronouns were also parsed correctly. In general, this grammar generated sentences with the correct word order, but it also tended to give many false positives such as “he will sleeps.” Although the experiential aspect and the prp suffix *-ing* exist at this stage, the auxiliary *is* was not

## Additions to the tdl

### Lexical types and labels

In order to improve the precision and scope of the grammar, many new lexical entries and corresponding lexical types were made.

#### Adverbs

Two adverbs – *never* and *not* – were included for the purpose of negation. The same predicate *neg­\_rel* was initially used for them both. Both used the same scopal modification but what they modify differs.

1. a. It never sleep.

b. \*It not sleep.

c. \*It does never sleep.

d. It does not sleep.

e. It will never sleep.

f. It will not sleep.

g. \*It never cute.

h. It not cute.

Although they seem to occur in similar positions in the sentence, *never* is POSTHEAD – and modifies non-auxiliary verbs while *not* is POSTHEAD + when it modifies auxiliary verbs and POSTHEAD – when it modifies adjectives. Therefore, one lexical was created for *never*, and two were created for *not*.

never\_neg-adv-lex := basic-scopal-adverb-lex &

[ SYNSEM.LOCAL.CAT [ VAL [ SPR < >,

COMPS < >,

SUBJ < >,

SPEC < >],

POSTHEAD -,

HEAD.MOD < [ LOCAL.CAT [ HEAD verb &

[ AUX - ],

VAL [ SUBJ cons,

COMPS null ] ] ] > ] ].

notaux\_neg-adv-lex := basic-scopal-adverb-lex &

[ SYNSEM.LOCAL.CAT [ VAL [ SPR < >,

COMPS < >,

SUBJ < >,

SPEC < >],

POSTHEAD +,

HEAD.MOD < [ LOCAL.CAT [ HEAD verb &

[ AUX +,

CAN - ],

VAL [ SUBJ cons,

COMPS < #comps > ] ] ] > ] ].

notadj\_neg-adv-lex := basic-scopal-adverb-lex &

[ SYNSEM.LOCAL.CAT [ VAL [ SPR < >,

COMPS < >,

SUBJ < >,

SPEC < >],

POSTHEAD -,

HEAD.MOD < [ LOCAL.CAT [ HEAD adj ] > ] ].

It was found that *not* however, cannot modify *can* as this would instead be expressed by one word: *cannot*. Hence, a new verb feature was created and *notaux\_neg-adv-lex* has to modify something that is CAN – ( the only things that are CAN + are *can* and *cannot*).

*Very* was also added into the lexicon. And it inherits from *notadj\_neg-adv-lex.* Initially, the HEAD value of the modified for *very* was set to be +jr to include adverbs as *very* can modify phrases like *very cute* and it was not semantically clear to me whether it could also be *very very* that modifies *cute*. However, to avoid unnecessary generation and for the grammar to look clearer, *very* was made to follow *not\_modadj* and only modify adjectives.

The aspect marker *already* was also added to the grammar as an adverb (with a predicate). And this lexical type was created for it:

aspect-adv-lex := basic-scopal-adverb-lex &

[ SYNSEM.LOCAL [ CONT.HOOK.INDEX.E.ASPECT perfective,

CAT [ VAL [ SPR < >,

COMPS < >,

SUBJ < >,

SPEC < >],

HEAD.MOD < [ LOCAL.CAT [ HEAD verb,

VAL [ SUBJ cons,

COMPS < > ] ] ] > ] ] ].

It can modify VPs that are either before or after it and adds in the perfective aspect.

The last adverb, *agak¸* which means “roughly” also has a relatively similar distribution.

agak-adv-lex := basic-scopal-adverb-lex &

[ SYNSEM.LOCAL.CAT [ VAL [ SPR < >,

COMPS < >,

SUBJ < >,

SPEC < >],

POSTHEAD -,

HEAD.MOD < [ LOCAL.CAT [ HEAD verb,

VAL [ SUBJ cons,

COMPS < > ] ] ] > ] ].

However, it is specifically POSTHEAD -. One of the earlier ideas was to further restrict the verb that *agak* could modify as although there does seem to be an obvious rule to what can be modified by it as certain verbs like *eat* in 2b) seem very unnatural as its head.

1. a. He agak can sing.

b. ?He agak eat the cat.

c. He agak know.

There is also a possibility that there is nothing syntactic preventing the construction, and it is more due to the semantics of the words. Thus, no further constraints would be added to the lexical type now to avoid the risk of unnecessary exclusion.[[1]](#footnote-2)

#### Determiners

All determiners inherit from the *determiner-lex* which has an empty valence list.

determiner-lex := basic-determiner-lex & norm-zero-arg & non-mod-lex-item &

[ SYNSEM.LOCAL.CAT.VAL [ SPR < >,

COMPS < >,

SUBJ < > ] ].

The determiners then differ from each other by their PRED values and the NUM value of their SPEC.

#### Nouns

A new lexical type *noun+det-lex-item* was created for when the demonstratives like *this* and *that* act like nouns (with an expanded meaning from their initial usage as determiners). The lexical item has an empty SPR list and unlike the countable common nouns, cannot be inflected for number as its NUM value is determined by the lexical entry (eg. *This*: PNG.NUM singular vs *these*: PNG.NUM plural).

noun+det-lex-item := norm-hook-lex-item & non-mod-lex-item &

[ SYNSEM [ LOCAL [ CAT [ HEAD noun,

VAL [ SPR < >,

COMPS < >,

SUBJ < >,

SPEC < > ]],

CONT [RELS <! relation &

[ LBL #nh, ARG0 #s ],

quant-relation & #det &

[ ARG0 #s, RSTR #h ]!>,

HCONS <! qeq & [ HARG #h,

LARG #nh ] !> ]],

LKEYS [ KEYREL relation,

ALTKEYREL #det ]]].

Although it has a HEAD noun value, unlike the *noun-*lex, this lexical type has an empty VAL.SPR list. In order to account for the difference in number value, two subtypes that inherit from this lexical type were made:

sg\_n+det-lex := noun+det-lex-item &

[ SYNSEM.LOCAL.CONT.HOOK.INDEX.PNG.NUM singular ].

pl\_n+det-lex := noun+det-lex-item &

[ SYNSEM.LOCAL.CONT.HOOK.INDEX.PNG.NUM plural ].

The next lexical noun entry that was added was *glass*. The existing nouns in the lexicon were either pronouns or countable common nouns, thus a new lexical type had to be created.

mass\_noun-noun-lex := noun-lex &

[ SYNSEM.LOCAL.CONT.HOOK.INDEX.PNG [ PER 3rd,

NUM singular,

COUNT - ] ].

This is exactly the same as *common\_noun-noun-lex* with the exception of the COUNT value which was a new png value added in as a result. This differentiates mass nouns from the common nouns which can undergo the plural-lex-rule.

Another lexicon addition that needed a new lexical type was “there”. This word was included in order to translate from the Mandarin *nali* (那里) which means *there* as in the adverb. In the small Mandarin Chinese grammar made, *nali* was created as a noun type with a constrained usage such that sentences like (3) can be generated while (4) cannot.

1. 那里 有 猫。

Nali you mao

There have cat

“There is a cat there.”

1. 猫 在 那里 唱歌。

Mao zai nali change

Cat prep there sing

“the cat sings there.”

Conveniently, the dummy subject is not required in Singlish and an acceptable Singlish translation of that Mandarin sentence is *there got cat*, a word-for-word translation of the original sentence. I think it is extremely likely for this Singlish *there* to have been heavily influenced by Chinese, which is why in this case, I have chosen to categorise it as a noun instead of an adverb. And the *loc\_noun-noun-lex* was specially created for it as it does not behave like other nouns.

loc\_noun-noun-lex := no-spr-noun-lex &

[ SYNSEM.LOCAL [ CAT.HEAD.CASE nom,

CONT.HOOK.INDEX.PNG.COUNT - ] ].

*There* cannot take any specifier and, in this limited grammar, only appears at the start of the sentence in the subject position[[2]](#footnote-3). Thus, it inherits from the supertype *no-spr-noun-lex* and has a CASE value of nom.

However, this was not enough as this *there* can only occur as the ARG1 of certain verbs.

1. a. There got a tree.

b. \*There is singing.

c. \*There eat the cat.

d. \*There will give me a dog.

e. There can sleep.

In sentence 5e however, the *can* is subject raising but the subject of *sleep* obviously cannot be *there* and thus, such constructions have been excluded in this stage of the grammar. Thus, in order to distinguish between verbs that can and cannot take *there* as their subject, a new HEAD feature LOC was added to the lexical type.

loc\_noun-noun-lex := no-spr-noun-lex &

[ SYNSEM.LOCAL [ CAT.HEAD [ LOC +,

CASE nom ],

CONT.HOOK.INDEX.PNG.COUNT - ] ].

#### Verbs

In order to have a translation of sentence (3), *got* was also added to the lexicon. It is a transitive verb that takes a noun as its complement thus it inherits from the *transitive-verb-lex* and has its subject in a nominative case and object in accusative case.

got\_tr-verb-lex := transitive-verb-lex &

[ SYNSEM.LOCAL.CAT.HEAD.NONINFL +,

ARG-ST < [ LOCAL.CAT.HEAD noun &

[ CASE nom ] ],

[ LOCAL.CAT.HEAD noun &

[ CASE acc ] ] > ].

It is, however, different from the other transitive verbs as it does not inflect. Thus, a new HEAD value of NONINFL was added to the grammar. In addition, it is the only verb that is able to take *there* as its subject. Thus, unlike other verb types, it has the LOC value of its subject unspecified. Due to the addition of *there* in the section above, other transitive verbs (*gen\_tr-verb-lex*) have HEAD.LOC – specified for their first argument in the ARG-ST.

A ditransitive verb, *give*, was also added into the lexicon.

ditransitive-verb-lex := main-verb-lex & ditransitive-lex-item &

[ SYNSEM.LOCAL.CAT.VAL.COMPS < #objind, #objdir >,

ARG-ST < [ LOCAL.CAT.HEAD noun ],

#objind &

[ LOCAL.CAT cat-sat &

[ VAL [ SPR < >,

COMPS < > ],

HEAD noun ] ],

#objdir &

[ LOCAL.CAT cat-sat &

[ VAL [ SPR < >,

COMPS < > ],

HEAD noun ] ] > ].

This lexical type performs similar to the existing verb-lex except that it has two objects and inherits from *ditransitive-lex-item* instead of *transitive-lex-item.* A further subtype was created for *give*:

ditr-verb-lex := ditransitive-verb-lex &

[ ARG-ST < [ LOCAL.CAT.HEAD noun &

[ CASE nom,

LOC - ] ],

[ LOCAL.CAT.HEAD noun &

[ CASE acc ] ],

[ LOCAL.CAT.HEAD noun &

[ CASE acc,

PRON - ] ] > ].

The cases of the complements of this lexical item were further specified here. During the testing of the grammar, it was found that sentences like “I give the cat her” were parsed. Having the pronoun in as the last argument seems slightly unnatural thus, the last item in the ARG-ST was given a PRON -. The HEAD.LOC value of its first argument (subject) is also set to “-“ as *give* cannot take *there* as a subject.

With the addition of embedded clauses, and verbs like *think* and *ask*, new types had to be created.

clausal-verb-lex := main-verb-lex &

[ SYNSEM.LOCAL.CAT.VAL.COMPS < #clause >,

ARG-ST < [ LOCAL.CAT.HEAD noun &

[ LOC - ] ],

#clause &

[ LOCAL.CAT.VAL [ COMPS < >,

SUBJ < > ] ] > ].

It takes a LOC – noun as a subject and a clause as its complement. Instead of an empty SPR list, its clausal complement was changed to have an unspecified SPR value so Singlish clauses like “cat cute” can be a complement.

For *think* and *know*, the *decl\_comp-verb-lex* was created.

decl\_comp-verb-lex := clausal-verb-lex & clausal-second-arg-trans-lex-item &

[ SYNSEM.LOCAL.CAT.VAL [ COMPS < [ LOCAL [ CAT.HEAD +vjc,

CONT.HOOK.INDEX.SF prop ] ] >,

SUBJ < [ LOCAL.CAT.HEAD.CASE nom ] > ] ].

To accommodate for Singlish clauses or clauses headed by a complementizer, adjectives and complementizers were added to the HEAD value of the complement. And in order to prevent statements like “I think if he sleeps”, the complementizer is limited by a propositional sentence force.

Similarly, a lexical item was also created for *ask*.

int\_tr-comp-verb-lex := clausal-verb-lex & clausal-second-arg-trans-lex-item &

[ SYNSEM.LOCAL.CAT.VAL [ COMPS < [ LOCAL [ CAT.HEAD comp,

CONT.HOOK.INDEX.SF ques ] ] >,

SUBJ < [ LOCAL.CAT.HEAD.CASE nom ] > ] ].

The first difference is the SF which is now *ques*. In addition, since in the Singlish grammar, regular clauses without the complementizer have an SF value of *prop-or-ques*, merely changing the SF value of the COMPS still allows sentences like “I ask he sleep” to parse. Hence, the HEAD value was further constrained to only allow complementizers. This correctly allows its complements to be clauses headed by *is it* or *if* and disallows interrogative clauses with subject verb inversions.

*Ask* is also able to take another complement - the person to whom the question is asked – creating the need for a ditransitive version of this lexical item.

int\_ditr-comp-verb-lex := ditr-clausal-verb-lex & clausal-third-arg-ditrans-lex-item &

[ SYNSEM.LOCAL.CAT.VAL [ COMPS < [ LOCAL.CAT [ HEAD.CASE acc,

VAL.SPR < > ] ],

[ LOCAL [ CAT.HEAD comp,

CONT.HOOK.INDEX.SF ques ] ] >,

SUBJ < [ LOCAL.CAT.HEAD.CASE nom ] > ] ].

The first COMPS item is the new argument which has to be in the accusative case. Its empty SPR list was included to ensure only bare NPs are accepted to reduce over-generation.

Auxiliaries were also added. Some, like *will* add a predicate.

will-aux-lex := subj-raise-aux-with-pred &

[ SYNSEM.LOCAL.CAT.VAL.COMPS.FIRST.LOCAL [ CAT.HEAD.FORM base,

CONT.HOOK.INDEX.E.ASPECT no\_aspect ],

ARG-ST < [ LOCAL.CAT.HEAD.LOC - ],

#comps > ].

*Will-aux-lex* inherits from *subj-raise-aux-with-pred* which is a subtype of *trans-first-arg-raising-lex-item-1*, the subtype of first argument raising lexical items that inserts a predicate. The auxiliaries like *will* and *can* are also unable to take complements modified by *already* hence the *no\_aspect*. This ensures that the right trees are formed when sentences like “she can sleep already” are parsed (with *already* modifying *can sleep*).

Although *want*­*\_aux* also adds a predicate, it only takes complements which have a *nonfinite* FORM which includes *base* and *toinf.*  Thus, a sister subtype identical to *will-aux-lex* with that exception was made for *want*.

Auxiliaries that do not add predicates inherit from *trans-first-arg-raising-lex-item-2*. For *does*, which is used in sentences like “it does hurt me”,

does\_subj-raise-aux-no-pred := subj-raise-aux & trans-first-arg-raising-lex-item-2 &

[ SYNSEM.LOCAL.CAT.VAL.COMPS.FIRST.LOCAL [ CAT.HEAD.FORM base,

CONT.HOOK.INDEX.E.ASPECT no\_aspect ] ].

The lexical type for the auxiliary verbs to used in present perfect sentences (various forms of *be*)is largely similar except for the HEAD.FORM value of the complement which is set as *prp* instead of *base*.

In addition, *to* was added to the lexicon for the *want* sentences. Initially, the entry for *to-infinitive-lex* was completely like *does\_subj-raise-aux-no-pred* except for an addition of a HEAD.FORM *toinf* value. However, since in my grammar, auxiliaries do not take auxiliaries as complements it inherits instead from *verb-lex*:

to-infinitive-lex := trans-first-arg-raising-lex-item-2 & verb-lex &

[ SYNSEM.LOCAL.CAT [ HEAD.FORM toinf,

VAL.COMPS.FIRST.LOCAL.CAT [ HEAD.FORM base,

VAL.COMPS < > ] ] ].

Lastly, under verbs, there is the copula. Using the matrix, a lexical type was already created for copulas:

cop-lex := basic-verb-lex-super & trans-first-arg-raising-lex-item-2 &

non-mod-lex-item & basic-two-arg &

[ SYNSEM.LOCAL [ CAT.VAL [ SUBJ < [ LOCAL [ CONT.HOOK.INDEX #xarg,

CAT cat-sat &

[ VAL [ SPR < >,

COMPS < > ],

HEAD noun ] ] ] >,

COMPS < [ LOCAL.CAT cat-sat &

[ HEAD.PRD +,

VAL [ SUBJ < >,

COMPS < > ] ] ] >,

SPR < >,

SPEC < > ],

CONT.HOOK.XARG #xarg ] ].

Inheriting from this supertype, the lexical type for the adjectival copulas (*be-cop-lex*)was created by specifying the HEAD value of its complement and inheriting from *verb-lex.*

adj-comp-copula-verb-lex := cop-lex &

[ SYNSEM.LOCAL.CAT.VAL.COMPS.FIRST.LOCAL.CAT.HEAD adj ].

be-cop-lex := adj-comp-copula-verb-lex & verb-lex.

#### Complementizers

Three complementizers were added to this grammar and all of them select for clausal complements. They also inherit from *complementizer-lex-item* which inherits from *raise-sem-lex-item. Raise-sem-lex-item* allows the complementizer to take the things from the VAL of its complement and pass them up while complementizer-lex-item specifies having a clause (HEAD +vjc in Singlish) as its complement.

*Is it* is a question particle which takes something that is a main clause (“MC +”) and ensures the phrase has a sentence force of *que*.

qpart-lex-item := complementizer-lex-item &

[ SYNSEM.LOCAL [ CONT.HOOK.INDEX.SF ques,

CAT.VAL.COMPS.FIRST.LOCAL.CAT [ MC +,

HEAD.FORM finite ] ] ].

*If* and *that* are used in sentences with *ask* and *think* respectively. Unlike *qpart-lex-item*, both of them are INIT + which is to say since they are the heads, they would appear only before their complements and they both take non-main clauses as their complements. The *decl\_comps-complementizer-lex-item* was created for *that* and *int\_comps-complementizer-lex-item* for *if*.

decl\_comps-complementizer-lex-item := complementizer-lex-item &

[ SYNSEM.LOCAL [ CONT.HOOK.INDEX.SF prop,

CAT [ MC -,

VAL.COMPS.FIRST.LOCAL.CAT.MC -,

HEAD.INIT + ] ] ].

int\_comps-complementizer-lex-item := complementizer-lex-item &

[ SYNSEM.LOCAL [ CONT.HOOK.INDEX.SF ques,

CAT [ VAL.COMPS.FIRST.LOCAL.CAT.MC -,

HEAD.INIT + ] ] ].

The “MC –“ feature was added in to ensure that a phrase headed by *that* or *if* cannot unify with root.

### Aspect

There are three subtypes of aspect: perfective, imperfective and no\_aspect. The perfective aspect is introduced by the word *very*, the *imperfective* by the present participle, and *no\_aspect* was created to ensure that there was a subtype that excluded the other two aspects.

For example, in a sentence like “I can sleep already” only one parse would be ideal and that is where *already* modifies “can sleep”. To prevent ambiguity, it had to be ensured that the auxiliary verb like *can* does not take something that has been modified by *already* as a complement. Thus, the ASPECT value of *no\_aspect* was assigned to its complement. This was the case as well for other auxiliary verbs like *does* and *will*.

### Form

Although Singlish uses many standard English words, the forms proved to be slightly complicated as they affected a lot in the grammar but were not identical to how the forms in the standard English grammar.

This hierarchy was created. The FORM *prp* was initially a daughter of *nonfinite* but since Singlish sentences can be headed by verbs in the present participle, it was changed to a daughter of *finite*. As the matrix specified in the roots file that the FORM of a sentence to be finite. The *underspec* FORM value was created due to the Singlish tendency to allow for uninflected verbs in otherwise ungrammatical instances.

1. a. A cat sing.

b. A cat sings.

c. Cats sing.

d. \*Cats sings.

For instance, while 6a) would not be accepted in standard English, it is in Singlish although 6b) would also be accepted. However, this does not work the other way as the wrongly inflected form in 6d) is not accepted. To unify *a cat* and *sing*, *sing* would have to have a FORM value of *underspec*.

### Head feature values & others

Due to the new lexicon, new features as mentioned earlier had to be introduced.

Under the head features, INIT was added as an indicator as to whether the lexical item is head-initial. Only the POSTHEAD feature was used at first, but it could not be used on an item that was the head of the phrase. This feature was added to the head-comp and comp-head phrases. Due to the addition of the noun *there*, the feature LOC was added to constrain which verbs may take it as a subject. Other features such as the PRD, AUX, PRON and FORM were added by the matrix and all except FORM take Boolean values.

Under png, COUNT was introduced to differentiate between count nouns like *cat* and non-count nouns like *glass*.

Verbs have the feature INV which is used in the subject-verb inversion to indicate interrogatives. They also have the NONINFL feature which is used to indicate when a verb like *got* does not inflect. Inflectional verbal rules like the prp-lex-rule and the 3sg-lex-rule are constrained to only apply on NONINFL – verbs. The CAN feature for verbs was made to distinguish *can-like* verbs that will not be modified by *not*. This feature has a very narrow usage and it is uncertain if it is the best solution for this issue.

This grammar also has three daughters: 1st, 2nd and 3rd which are mostly used for pronouns, as the daughters of Person while it has *singular* and *plural* as its two Number daughters.

## Parse resulrs

In the final testsuite, 39/79 sentences were parsed with two false negatives and no false positives.

True Positives:

|  |  |  |
| --- | --- | --- |
| Number | Sentence | Phenomena |
| 1 | The cat eat the ant. | Word order |
| 5 | The cat sleep. | Word order |
| 7 | Cat sleep | Determiner |
| 10 | The cats sleep. | Determiner, common noun |
| 11 | Cats sleep. | Determiner, common noun |
| 13 | That cat sleeps. | Agreement |
| 14 | That cat sleep. | Agreement |
| 15 | Those cat sleep. | Agreement |
| 19 | Those cats sleep. | Agreement |
| 21 | I sleep. | Agreement |
| 23 | We sleep. | Agreement |
| 25 | You sleep. | Agreement |
| 27 | He sleep. | Case |
| 29 | I kacau him. | Case |
| 34 | He never sleep. | Negation |
| 35 | He never sleeps. | Negation |
| 40 | He does not sleep. | Negation |
| 42 | Is it sleeping? | Matrix y/n |
| 43 | It sleeping is it? | Matrix y/n |
| 44 | Is it it sleeping? | Matrix y/n |
| 46 | It sleeping? | Matrix y/n |
| 47 | The cat can sleep. | Auxiliary |
| 50 | He can never sleep. | Negation, auxiliary |
| 51 | I can sleep. | Agreement, auxiliary |
| 53 | The cat kacau the tree and the ant. | Coordination |
| 56 | The cat and the ant and the tree kacau me. | Coordination |
| 58 | The cat, the ant and the tree kacau me. | Coordination |
| 59 | Big cat sleep. | Adjective |
| 61 | big big cat sleep. | Adjective |
| 62 | Cat is big. | Adjective, copula |
| 63 | Cat big. | Adjective, copula |
| 64 | Cat are big. | Adjective, copula |
| 66 | The cat think he is sleeping. | Embedding |
| 67 | The cat think that he cute. | Embedding, complementizer |
| 70 | The cat think he think it cute. | Embedding |
| 72 | He sleep already. | Aspect |
| 73 | He already sleep. | Aspect |
| 74 | He can sleep already. | Aspect, auxiliary |
| 78 | He give us a cat. | Ditransitive, case |

True Negatives:

|  |  |  |
| --- | --- | --- |
| Number | Sentencer | Phenomena |
| 3 | The ant the cat eat. | Word order |
| 4 | Eat the ant the cat. | Word order |
| 6 | Sleep the cat. | Word order |
| 8 | Cat the sleep. | Word order |
| 9 | The he sleep. | Determiner, pronoun |
| 12 | The cat kacau. | Argument optionality |
| 16 | Those cat sleeps. | Agreement |
| 17 | That cats sleep. | Agreement |
| 18 | That cats sleeps. | Agreement |
| 20 | Those cats sleeps. | Agreement |
| 22 | I sleeps. | Agreement |
| 24 | We sleeps. | Agreement |
| 28 | You sleeps. | Agreement |
| 28 | Him sleep. | Case |
| 30 | Me kacau him. | Case |
| 31 | I kacau he. | Case |
| 32 | Me kacau he. | Case |
| 36 | He sleep never. | Negation |
| 37 | He sleep not. | Negation |
| 38 | He not sleep. | Negation |
| 39 | He not does sleep. | Negation, auxiliary |
| 41 | He does not sleeps. | Negation, auxiliary |
| 45 | It is it sleeping? | Matrix y/n |
| 48 | The cat sleep can. | Auxiliary |
| 49 | The cat can sleeps. | Auxiliary |
| 52 | I can sleeps. | Agreement, auxiliary |
| 54 | The cat kacau and the tree the ant. | Coordination |
| 55 | The cat kacau the tree the ant and. | Coordination |
| 57 | The cat and the ant the tree kacau me. | Coordination |
| 60 | Big he sleep. | Adjective |
| 65 | Cats is big. | Adjective, copula |
| 68 | The cat think he cute that. | Embedding, complementizer |
| 69 | The cat that think he cute. | Embedding, complementizer |
| 71 | The cat think if he is sleeping. | Embedding, complementizer |
| 75 | He can already sleep. | Aspect, auxiliary |
| 76 | He give us her | Ditransitive, case |
| 77 | He give us she. | Ditransitive, case |
| 79 | He give a cat us | Ditransitive, cases |

False Negatives:

|  |  |  |
| --- | --- | --- |
| Number | Sentence | Phenomena |
| 2 | The ant the cat eat. | Word order |
| 33 | He never. | Negation |

False Positives: -

## Translation & semi.vpm

A line was added E.ASPECT:

no\_aspect << \*

This reduces the number of sentences generated by preventing sentences with underspecified aspect values in other languages to generate sentences in the imperfective aspect. For instance, when (7) was translated without the line above, both “I sing” and “I singing” were generated. This addition reduced the possible generations to “I sing”.

1. 我 唱歌

I sing

“I sing.”

This is valid for the translation from Mandarin Chinese as it has an imperfective marker. In a language that does not indicate aspect at all, it may be better to generate both sentences.

## Areas for improvement

*There* was a word that was given a very limited scope in the current version of the grammar. The same sense of *there* could also be represented by another lexical entry that is an adverb such that a Singlish version of sentence (4) could also be parsed. This entry would modify VPs with empty COMPS lists and would be POSTHEAD +. However, this would also raise the issue of the probability of having two *there* in a sentence eg. “there got cat there” and additional constraints would have to be made to prevent it.

Another area for improvement would be adding in different forms of the irregular verbs using the irregs.tab file. Even though the past and non-past tenses were included when the grammar was created, those features have not been used. By adding in the different forms of the verbs, words like *eat* and *sleep* would be able to inflect for past tense and the grammar would be able to generate and parse sentences in the past tense.

In addition, there is only one singular indefinite article: *a.* This is because the current grammar does not distinguish whether the first letter of a noun is a vowel and thus, *an* has not been added to the grammar even though the common-noun *ant* exists in the lexicon. However, according to native speaker intuition, utterances like “a ants” are quite unmarked in Singlish unlike “an cat”. And in this early a stage of the grammar, such additions are not as vital.

*Cannot* was added in the later stage of the grammar and it is used almost in the same context as other no-pred auxiliaries and has the same supertype as *will.* However, I noticed that it behaves differently in sentences with negation. In general in this grammar, *not* modifies only adjectives or auxiliary verbs but not *can* or *cannot.* However, in real-life sentences with *cannot*, *not* would be emphasised and modify the non-auxiliary verb like *sleep* in (8).

1. I cannot NOT sleep.

Thus, as an improvement a new type of *not* could be added specifically for such special sentences where *not* modifies the non-auxiliary verb.

The hierarchy of forms created is inadequate to address the different forms of *be.* Although the verb *are* is used for plural subjects, it is also used when the subject is 2nd person singular and this has not been accounted for in the grammar. Currently, this presents no visible problems as there is only one *you* in qsg and it has an unspecified number value. However, this means that every time *you* is used in a sentence with *are*, it only refers to its plural version and this is not refelective of the usage. Possible solutions to solve this could be:

1. To have two *are,* one for the plural and one for the 2nd person singular
2. To further extend the form hierarchy, to have non\_3sg branch out into 1sg and non\_1sg

In addition, in the current grammar, verbs that are uninflected and have FORM *form* are also able to unify in phrases. Since the *base* form looks identical to the uninflected form, a sentence involving a verb in *base* (eg. ‘he can sleep’) would have at least two parses (with *sleep* in base and uninflected), causing an over-generation. A solution would be to have a rule that always forces the verb to be in the lowest common denominator for the unification.

Furthermore, due to the abovementioned limitation, the lexical rule that goes from *form* to *underspec* has been omitted from this grammar as a solution for over-generation. But if the rule preventing uninflected lexical entries is in place, a new rule for the *underspec* form would be required.

Lastly, although changing the *prp* form to a *finite* one worked fine, a pumping rule instead that allows a *prp* (nonfinite) form to become *finite* could be another way to deal with this issue.

## Commands used

* Getting to the virtual environment
  + Source <path\_to\_environment>/bin/activate
* Grammar compilation
  + $ ace -G qsg.dat -g ace/config.tdl
* To check if sentence parses
  + $ ace -g qsg.dat -l (for lui)
  + $ ace -g qsg.dat -m (for mrs)
  + :c (to see where unification failed)
  + :r <rule\_name> (to open rule and drag the different values from the avm in to see where unification failed)
* Making a testsuite and testing it

1. $ ./make\_item –map translat i-comment qsg\_testsuite\_part1 item
2. If permission is denied, $ chmod +x make\_item
3. Transfer item to the new testsuite folder in tsdb/skeletons
4. $ delphin mkprof -s tsdb/skeletons/testsuite-17/ trees/testsuite.17
5. $ delphin process -g qsg.dat trees/testsuite.17

* Viewing parse results
  + $ delphin select ‘i-id readings i-input where i-wf = 1 and readings = 0’ trees/testsuite.17
  + For false negatives
* Treebanking

1. $ art -f -a ‘ace - -disable-generalization -g qsg.dat -O’ trees/testsuite.17
2. $ fftb -g qsg.dat - -browser - -webdir ~/bin/acetools-x86-0.9.30/assets/ tree/testsuite.17

(this launches the browser so that the right trees can be chosen)

1. $ fftb -g qsg.dat - -browser - -webdir ~/bin/acetools-x86-0.9.30/assets/ --gold tree/testsuite.16 trees/testsuite.17 - -auto

(to transfer the gold profile from testsuite.16 to testsuite.17)

* Looking at generation and semantics
  + $ echo “<sentence>” | ace -g qsg.dat -Tfq | ace -g qsg.dat -e

(to give parses with the same semantics)

* + $ echo “<sentence>” | ace -g qsg.dat -Tfq

(to give the semantics)

* Machine translation
  + $ echo “<sentence>” | ace -g qsg.dat -Tf1 | python <filter\_rule> | ace -g <address\_other\_grammar\_.dat> -e - -disable-subsumption-test

1. More improvements could be made on this. [↑](#footnote-ref-2)
2. Though this is not the case in regular Singlish and the phenomena will be slightly elaborated on in section 6. [↑](#footnote-ref-3)