Representing Japanese Reflexive Pronouns Through Finite State Tree Automata

Julian M. Rice
UCLA | March 21st, 2019
Computational Linguistics I

| Introducing Japanese Reflexives & FSTA Conversion

This study discusses how reflexive pronouns in Japanese have different characteristics through finite state tree automatas. Unlike in English, there are multiple reflexives in Japanese. This investigation will look at the different characteristics of three of these reflexives - zibun, zibun-zishin, and mizukara. Each of these reflexives differ in regards to if they require a local antecedent and support long distance binding. By creating a finite state tree automaton that allows the use of Japanese reflexives with valid ending, nullary, unary and binary transitions, this study will consist of trees in Haskell that work with the tree automata and aim to use differing sentences to delineate the valid and invalid use of zibun, zibun-zishin, and mizukara in Japanese.

Japanese is a head-final language; this means that the head of a phrase, or 'main portion', follows the complement. This changes how syntax trees are generated, as the complementary will appear on the left branch of each part of the tree. However, for the sake of simplicity and due to time restraints, the following data will be adjusted to a head-initial structure that allows finite state tree automaton to more simplistically describe the rules of the many Japanese reflexives at hand. Zibun is a reflexive pronoun that means self, and is known to support both short and long distance binding - local and non local antecedents[1].

The first sentence, labeled (**S1)**, is listed below:

(S1)	Yukino$_1$	ga	zibun$_1$	wo	hihan-shi-ta.
	Yukino	NOM	self	ACC	criticize-do.Pst
	'Yukino$_1$ criticized herself$_1$.'				

In (S1), zibun is a reflexive pronoun that refers to its local antecedent, Yukino. Japanese reflexive pronouns like zibun cannot be used without an antecedent that is located before it; this follows Condition A of Chomsky's traditional binding theory[2], which states that an anaphora, or reflexive, must have a local, nearby antecedent. The sentence's original, valid syntax tree has been mapped out on **Figure A1**, and its modified tree that will be used in the Haskell program has been mapped out on **Figure A2**.

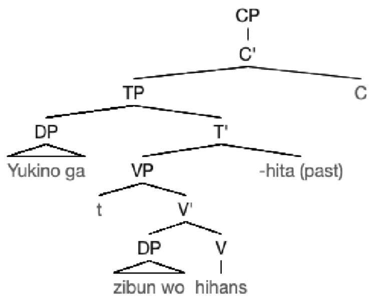


Figure A1: Original Syntax Tree for S1

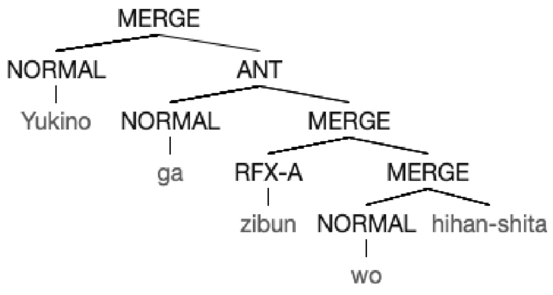


Figure A2: Modified Tree for S1 (FSTA)[3]

[1] Kishida, M. (2011). Reflexives in Japanese (Published doctoral dissertation). University of Maryland, College Park.
[2] Chomsky, N. (1982). Some Concepts and Consequences of the Theory of Government and Binding. Linguistic Inquiry Monographs, doi:10.2307/3729191
[3] **Abbreviations**: ANT = Antecedent (in lieu of ADJ, covered during the course), RFX-A = Reflexive A (zibun)

The next sentences, labeled S2 and S3, are listed below:

(S2)	Yukino$_1$	ga	zibun-zishin$_1$	wo	hihan-shi-ta.
	Yukino	NOM	self-self	ACC	criticize.do.Pst
	'Yukino$_1$ criticized herself$_1$.'				
(S3)	Yukino$_1$	ga	mizukara$_1$	wo	hihan-shi-ta.
	Yukino	NOM	self	ACC	criticize.do.Pst
	'Yukino$_1$ criticized herself$_1$.'				

In (S2), zibun-zishin is a reflexive pronoun that also refers to its local antecedent, Yukino. There are actually two known uses for zibun-zishin; its first is that of a reflexive pronoun. Its other use is as a self-intensifier, where zibun-zishin may be interpreted, in English, along the lines of 'He himself'. This study will only focus on the reflexive zibun-zishin and its characteristics. Like zibun and as demonstrated in (S2), zibun-zishin supports also Condition A.

Mizukara is also a reflexive pronoun that has some unique traits. Unlike zibun and zibun-zishin, mizukara originally consisted of three elements before it was combined into one. The combination of the preposition *kara* (from) and genitive marker *zu* was attached to *mi* (body), which is not the same character as *zi* in zibun. The reason why mizukara changed from its original, more adverbial nature can be derived from the Japanese government's 1946 Modern Kana Usage movement, which changed the character for *mi* from body (身) to self (自); this new character is the same character as the zi in zibun and zibun-zishin[4]. Despite this morphological change, the pronunciation for mizukara remains in its native Japanese kun-yomi[5] as *mi* instead of its derived Chinese pronunciation, *zi*. In (S3), we see that despite mizukara's morphologically unique traits, mizukara remains similar to zibun in regards to its support for attachment to a local antecedent (Condition A).

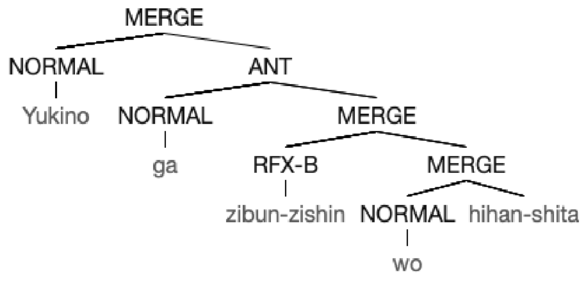


Figure A3: Modified Tree for S2 (FSTA)

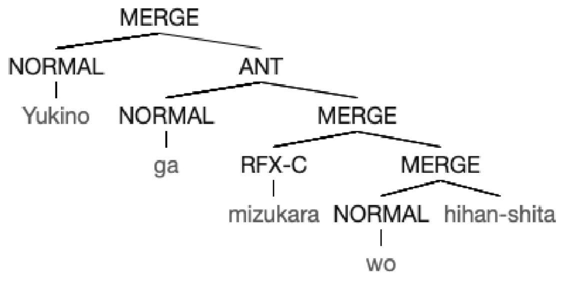


Figure A4: Modified Tree for S3[6] (FSTA)

The following sentences, labeled (S4), (S5), and (S6), are listed below:

(S4)	Yukino$_1$-wa [Kuribo$_2$-ga	zibun$_{1/2}$-no	koe-ga	kikoe-ta]	to	omot-ta.
	Yukino-TOP Kuribo-NOM	self-GEN	voice-NOM	hear-Pst	Comp	think.Pst
	'Yukino$_1$ thought Kuribo$_2$ heard their$_{1/2}$ voice.'					
(S5)	Yukino$_1$-wa [Kuribo$_2$-ga	zibun-zishin$_{1/*2}$-no koe-ga		kikoe-ta]	to	omot-ta.

[4] Rice, Julian. (2018). Syntactic Properties of the Reflexive Mizukara in Japanese. UCLA.
[5] Kanji often times have two pronunciations - Kunyomi corresponds to the native Japanese pronunciation for a Kanji character, whereas Onyomi corresponds towards the borrowed Chinese pronunciation / way to read.
[6] Abbreviations: RFX-B = Reflexive B (zibun-zishin), RFX-C = Reflexive C (mizukara)

	Yukino-TOP	Kuribo-NOM	self-self-GEN	voice-NOM	hear-Pst	Comp	think.Pst
	‘Yukino$_1$ thought Kuribo$_2$ heard their$_{1/2}$ voice.’						
(S6)	Yukino$_1$-wa	[Kuribo$_2$-ga	mizukara$_{1/*2}$-no	koe-ga	kikoe-ta]	to	omot-ta.
	Yukino-TOP	Kuribo-NOM	self-GEN	voice-NOM	hear-Pst	Comp	think.Pst
	‘Yukino$_1$ thought Kuribo$_2$ heard their$_{1/2}$ voice.’						

The following data shows that the reflexive pronouns in (S4) and (S6), zibun and mizukara, do in fact support long-distance binding and a non-local antecedent[7], which means that Condition B is also satisfied. On the other hand, (S5) shows that the reflexive pronoun zibun-zishin only adheres to Condition A and does not support non-local antecedents. With that being said, this means that two different sets of rules have to be established for the anaphor status of zibun, mizukara, and zibun-zishin when modeling the finite state tree automatas based on our Haskell implementation in Computational Linguistics I (**L185A**).

Figure A5 and A6 show what (S4) can originally be interpreted to be, as well as its simplified form. Figure A7 shows the modified tree for (S5). Appendix A8 shows the modified tree for (S6). Below these trees lies a summary of the characteristics for each Japanese reflexive.

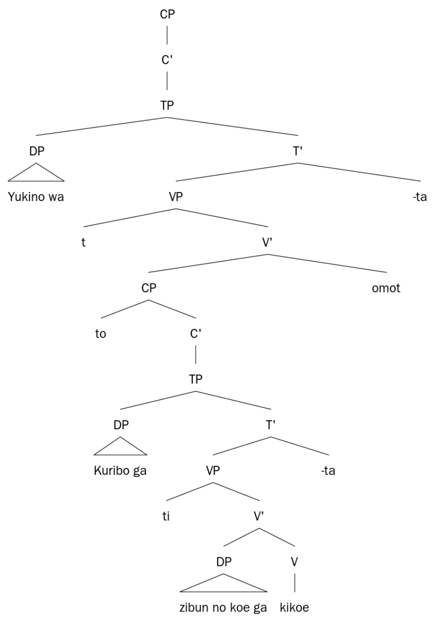


Figure A5: Original Syntax Tree for S4

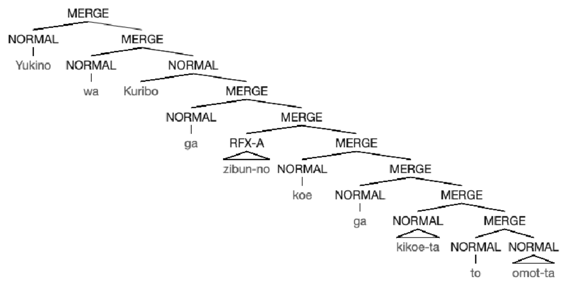


Figure A6: Modified Tree for S4 (FSTA)

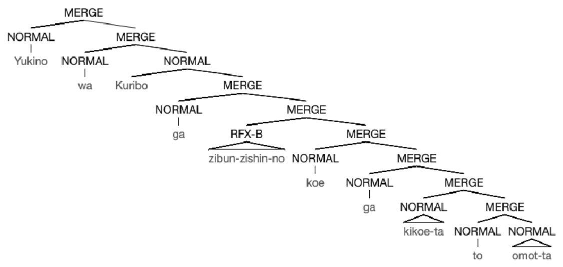


Figure A7: Modified Tree for S5 (FSTA)

Japanese Reflexive	Local Antecedent	Non-Local Antecedent
Zibun	OK	OK
Zibun-Zishin	OK	NO
Mizukara	OK	OK*

Table 1: Basic Summary of Japanese Reflexive Characteristics

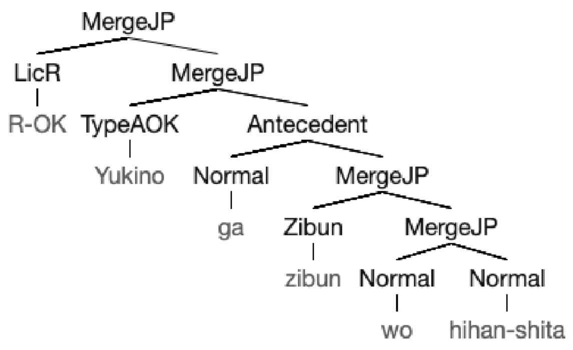
[7] One important note to make is that mizukara does not support a non-local antecedent when the GEN (genitive) marker is absent. However, the GEN marker allows mizukara to support a non-local antecedent.

| Running the FSTA & Analysis

The following rules have been identified as valid rules for trans:

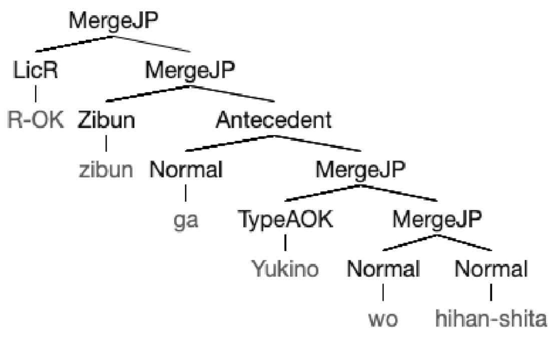
```haskell
data AnaphoraStatus = Normal |
                     Zibun | ZibunZishin | Mizukara |
                     TypeAOK | TypeBOK |
                     LicR deriving (Eq,Ord,Show)

data Transitions = MergeJP |
                   Antecedent deriving (Eq, Ord, Show)

fsta_jp :: Automaton AnaphoraStatus String Gam1 Transitions
fsta_jp = ( -- Ending States
    [Normal, TypeAOK, TypeBOK],
    -- Nullary Transitions
    [(s, Normal)| s <- ["wo","ga","koe","kikoe-ta","omot-ta","to","no","hihan-shita","wa"]] ++
    [(s, TypeAOK) | s <- ["Yukino"]] ++ --Long distance or singular only antecedent
    [(s, TypeBOK) | s <- ["Kuribo"]] ++ --Example for short distance and the latter antecedent
    [(s, LicR) | s <- ["R-OK"]] ++
    [(s, Zibun) | s <- ["zibun", "zibun-no"]] ++
    [(s, ZibunZishin) | s <- ["zibun-zishin", "zibun-zishin-no"]] ++
    [(s, Mizukara) | s <- ["mizukara", "mizukara-no"]],
    -- Unary Transitions
    [],
    -- Binary Transitions
    [(Normal, Normal, MergeJP, Normal),
    (Normal, Zibun, MergeJP, Zibun),

    -- Local Antecedent Rules
    (TypeAOK, Zibun, MergeJP, Normal),
    (TypeAOK, Mizukara, MergeJP, Normal),
    (TypeAOK, Normal, MergeJP, Normal),

    -- Non-local Antecedent Rules
    (TypeBOK, Zibun, MergeJP, Normal),
    (TypeBOK, Mizukara, MergeJP, Normal),
    (TypeBOK, ZibunZishin, MergeJP, Normal),
    (TypeBOK, Normal, MergeJP, Normal),

    -- Unsatisfied Reflexive (First Blood!) Rules
    (Normal, Zibun, Antecedent, Zibun),
    (Normal, Mizukara, Antecedent, Zibun),
    (Normal, ZibunZishin, Antecedent, Zibun),

    -- Search for Antecedent Rules
    (Zibun, Normal, MergeJP, Zibun),
    (ZibunZishin, Normal, MergeJP, ZibunZishin),
    (Mizukara, Normal, MergeJP, Mizukara),

    -- Finishing State
    (LicR, Normal, MergeJP, Normal)]
)
```

Figure B1: Japanese Reflexive Finite State Tree Automata v1



```
tree_jp1a :: Tree String Gam1 Transitions
tree_jp1a = Binary MergeJP (Leaf "R-OK")
(Binary MergeJP (Leaf "Yukino")  (Binary
Antecedent (Leaf "ga") (Binary MergeJP (Leaf
"zibun") (Binary MergeJP (Leaf "wo") (Leaf
"hihan-shita")))))
```

Figure B2: (S1a) Syntax Tree Representation & Corresponding Haskell Code (Valid)



```
tree_jp1b :: Tree String Gam1 Transitions
tree_jp1b = Binary MergeJP (Leaf "R-OK")
(Binary MergeJP (Leaf "zibun")  (Binary
Antecedent (Leaf "ga") (Binary MergeJP (Leaf
"Yukino") (Binary MergeJP (Leaf "wo") (Leaf
"hihan-shita")))))
```

Figure B3: (S1b) Syntax Tree Representation & Corresponding Haskell Code (Invalid)

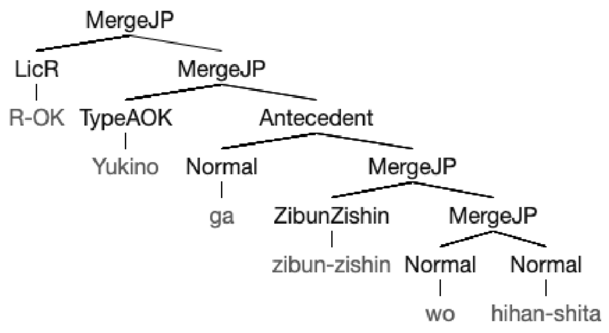
Two variations of (S1) were created to test reflexive validity of zibun. The following two are:

(S1a)	Yukino$_1$	ga	zibun$_1$	wo	hihan-shi-ta.
	Yukino	NOM	self	ACC	criticize-do.Pst
	'Yukino$_1$ criticized herself$_1$.'				
(S1b)	*Zibun$_1$	ga	Yukino$_1$wo		hihan-shi-ta.
	Self	NOM	Yukino ACC		criticize-do.Pst
	'Herself$_1$ criticized Yukino$_1$.'				

The following code was run using a function, **insideSet**, which determines whether or not there is a finishing state given a tree and its finite state tree automaton. In this case, the above trees were used on the finite state tree automaton given in **Figure B1**, and the results follow:

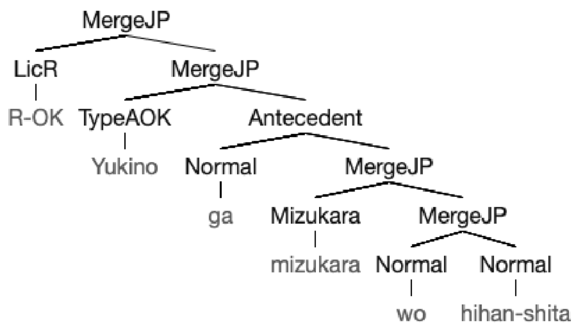
```
*JapaneseReflexives> insideSet fsta_jp tree_jp1a
[Normal]
*JapaneseReflexives> insideSet fsta_jp tree_jp1b
[]
```

This reveals that our the finite state tree automaton `fsta_jp` successfully went through both (S1a) and (S1b) and determined that the former was a valid sentence and the latter an invalid sentence. The reasoning behind this is that reflexive zibun follows Condition A (and B) and, as an anaphora, is grammatically incorrect if there is no antecedent to trace back and refer to. (S1a) shows that zibun reaches Yukino through the rule `(TypeAOK, Zibun, MergeJP, Normal)`, and by merging into the normal state, generates a valid sentence. On the other hand, (S1b) ends with a rule along the line of `(Zibun, Normal, MergeJP, Normal)`, which does not exist, or `(Zibun, Normal, MergeJP, Zibun)`, which is also invalid because `Zibun` cannot be an ending state.



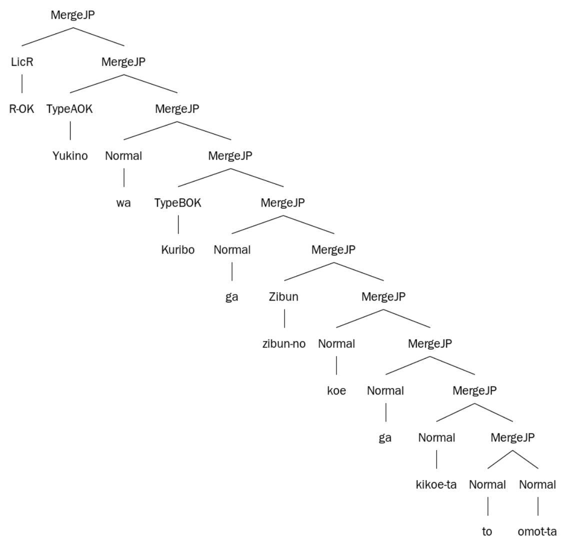
```
tree_jp2 :: Tree String Gam1 Transitions
tree_jp2 = Binary MergeJP (Leaf "R-OK") (Binary
MergeJP (Leaf "Yukino")  (Binary Antecedent
(Leaf "ga") (Binary MergeJP (Leaf
"zibun-zishin") (Binary MergeJP (Leaf "wo")
(Leaf "hihan-shita")))))
```

Figure B4: (S2) Syntax Tree Representation & Corresponding Haskell Code (Valid)



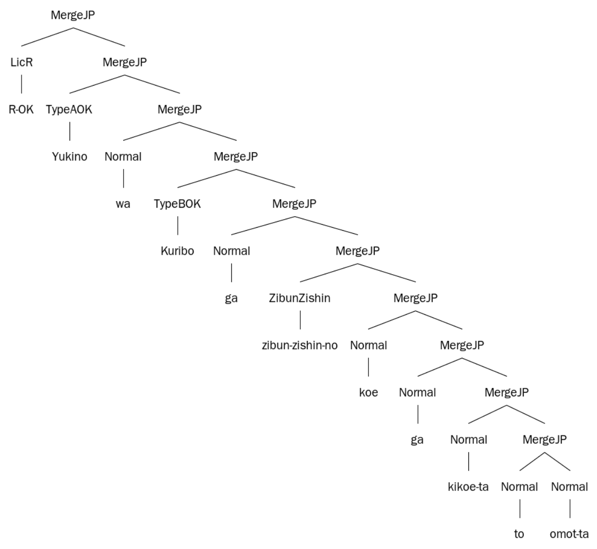
```
tree_jp3 :: Tree String Gam1 Transitions
tree_jp3 = Binary MergeJP (Leaf "R-OK")
(Binary MergeJP (Leaf "Yukino")  (Binary
Antecedent (Leaf "ga") (Binary MergeJP (Leaf
"mizukara") (Binary MergeJP (Leaf "wo") (Leaf
"hihan-shita")))))
```

Figure B5: (S3) Syntax Tree Representation & Corresponding Haskell Code (Valid)



```
tree_jp4 :: Tree String Gam1 Transitions
tree_jp4 = Binary MergeJP (Leaf "R-OK") (Binary
MergeJP (Leaf "Yukino") (Binary MergeJP (Leaf
"wa") (Binary MergeJP (Leaf "Kuribo") (Binary
Antecedent (Leaf "ga") (Binary MergeJP (Leaf
"zibun-no") (Binary MergeJP (Leaf "koe")
(Binary MergeJP (Leaf "ga") (Binary MergeJP
(Leaf "kikoe-ta") (Binary MergeJP (Leaf "to")
(Leaf "omot-ta"))))))))))
```

Figure B6: (S4) Syntax Tree Representation & Corresponding Haskell Code (Valid)



```
tree_jp5 :: Tree String Gam1 Transitions
tree_jp5 = Binary MergeJP (Leaf "R-OK")
(Binary MergeJP (Leaf "Yukino") (Binary
MergeJP (Leaf "wa") (Binary MergeJP (Leaf
"Kuribo") (Binary Antecedent (Leaf "ga")
(Binary MergeJP (Leaf "zibun-zishin-no")
(Binary MergeJP (Leaf "koe") (Binary MergeJP
(Leaf "ga") (Binary MergeJP (Leaf "kikoe-ta")
(Binary MergeJP (Leaf "to") (Leaf
"omot-ta"))))))))))
```

Figure B7: (S5) Syntax Tree Representation & Corresponding Haskell Code (Valid)

Now, take another look at (S2) - (S5), as well as Figure B4-B7.

(S2)	Yukino$_1$	ga	zibun-zishin$_1$	wo	hihan-shi-ta.			
	Yukino	NOM	self-self	ACC	criticize.do.Pst			
	'Yukino$_1$ criticized herself$_1$.'							
(S3)	Yukino$_1$	ga	mizukara$_1$	wo	hihan-shi-ta.			
	Yukino	NOM	self	ACC	criticize.do.Pst			
	'Yukino$_1$ criticized herself$_1$.'							
(S4)	Yukino$_1$-wa [Kuribo$_2$-ga		zibun$_{1/2}$-no		koe-ga	kikoe-ta]	to	omot-ta.
	Yukino-TOP Kuribo-NOM		self-GEN		voice-NOM	hear-Pst	Comp	think.Pst
	'Yukino$_1$ thought Kuribo$_2$ heard their$_{1/2}$ voice.'							
(S5)	Yukino$_1$-wa [Kuribo$_2$-ga		zibun-zishin$_{1/*2}$-no koe-ga			kikoe-ta]	to	omot-ta.
	Yukino-TOP Kuribo-NOM		self-self-GEN		voice-NOM	hear-Pst	Comp	think.Pst
	'Yukino$_1$ thought Kuribo$_2$ heard their$_{1/*2}$ voice.'							

The same rules apply for zibun-zishin and mizukara when it comes down to matching to the local antecedent, and the finite state tree automaton successfully identifies this through its set of rules. The following can be run on `JapaneseReflexives.hs` for verification, as seen below:

```
*JapaneseReflexives> insideSet fsta_jp tree_jp2
[Normal]
*JapaneseReflexives> insideSet fsta_jp tree_jp3
[Normal]
*JapaneseReflexives> insideSet fsta_jp tree_jp4
[Normal]
*JapaneseReflexives> insideSet fsta_jp tree_jp5
[Normal]
*JapaneseReflexives> insideSet fsta_jp tree_jp6
[Normal]
```

The tree automaton, `fsta_jp`, successfully detects that if zibun-zishin is the Japanese reflexive in the sentence, a non-local antecedent cannot be used as the antecedent for zibun-zishin. This was done by setting up the two different possible antecedents (Yukino, Kuribo) as different data types (TypeAOK and TypeBOK). By running local and non-local rules through the TypeAOK data type and only local rules through the TypeBOK data type, the program is able to delineate between reflexives that support both Condition A and B (zibun, mizukara), and reflexives that only support Condition A (zibun-zishin). Scaling this for further research purposes with different proper nouns would be as trivial as adding what can be used as either a local or non-local antecedent as the TypeAOK and what can be used only as a local antecedent as data type TypeBOK.

The binary transitions are split into a couple of different categories - local antecedent, non local antecedent, unsatisfied reflexive, search for the antecedent, and finishing state. R-OK, or `LicR`, acts as an indicator of the sentence being completed and the reflexive successfully finding its antecedent. This transition is only possible if the Normal state is achieved, which is wholly dependent on whether a Japanese reflexive finds its antecedent or not. This is the main logic behind whether or not a sentence is valid after `insideCheck` has been run through the entire binary tree structure.

| Discussion & Further Tests

This short study has provided insight on how Japanese reflexives work through programming in Haskell and through finite state tree automaton. The driving logic behind what dictates whether or not a sentence is valid depends on whether or not a Japanese reflexive has found a nearby antecedent to pair up with. This follows Condition A logic, and works particularly well for sentences that are simple and do not have any independent clauses (S1) - (S3). For sentences that have independent clauses (S4) - (S6), the creation of two different data types that delineate whether a proper noun (antecedent) was a local one or non-local one helped solve this added difficulty. However, one weakness that arises from this logic is that a certain proper noun (like Kuribo) cannot be both only a local antecedent and a local antecedent and non-local antecedent, otherwise Kuribo would have to appear as an entry in both data types.

Reworking the original code to make it more adaptive towards head-final languages that rely on the complement being on the left side could act as a potential task for future studies that are related to this one. Given that Japanese is a rather differently structured language from English, it only makes sense that there would be a different set of rules to follow when working with finite state tree automaton for Japanese, or any other SOV language like German or Korean. Another way of pushing study more in depth would be analyzing Japanese reflexives with a higher variety of sentences. Some that come to mind would be developing more sophisticated finite state tree automata that accommodate more delicate characteristics seen in these Japanese reflexives. Some examples include sentences where there is a plural antecedent, sentences for mizukara where the genitive marker is absent, logophoric sentences, animacy and non-animacy, as well as first, second, and third person antecedent sentences. With the groundwork for seeing Japanese reflexives work from a software perspective, studies that take this experiment to a deeper level will likely find out more about the origin and inner workings of Japanese.

| Works Cited & Appendix

Kishida, M. (2011). Reflexives in Japanese (Published doctoral dissertation). University of Maryland, College Park.

Chomsky, N. (1982). Some Concepts and Consequences of the Theory of Government and Binding. Linguistic Inquiry Monographs, doi:10.2307/3729191

Rice, Julian. (2018). Syntactic Properties of the Reflexive Mizukara in Japanese. UCLA.

Hagstrom, Paul. (2000). Articulating the Tree, and some Applied Syntax. Boston University CAS Linguistics Program. PowerPoint file

Appendix

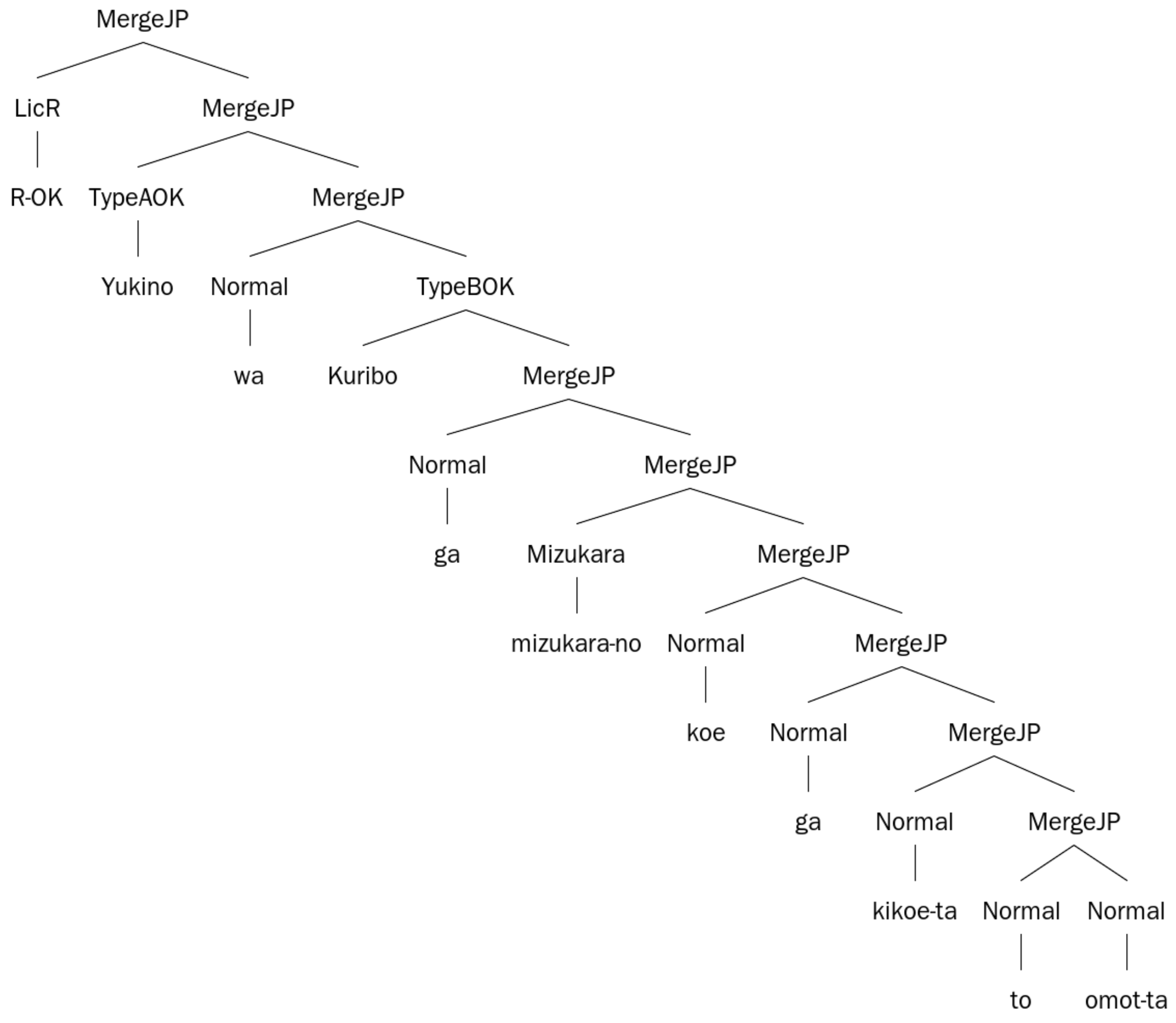


Figure A8: Modified Tree for S6 (FSTA)

Link for Syntax Tree Creation: http://mshang.ca/syntree/
Link 2 for Syntax Tree Creation: https://yohasebe.com/rsyntaxtree/

Sentence 1: [MERGE [NORMAL Yukino] [ANT [NORMAL ga] [MERGE [RFX-A zibun] [MERGE [NORMAL wo] [hihan-shita]]]]]

Sentence 1a: [MergeJP [LicR R-OK] [MergeJP [TypeAOK Yukino] [Antecedent [Normal ga] [MergeJP [Zibun zibun] [MergeJP [Normal wo] [Normal hihan-shita]]]]]]

Sentence 1b: [MergeJP [LicR R-OK] [MergeJP [Zibun zibun] [Antecedent [Normal ga] [MergeJP [TypeAOK Yukino] [MergeJP [Normal wo] [Normal hihan-shita]]]]]]

Sentence 2: [CP [C′ [TP [^DP Yukino ga] [T′ [VP [t] [V′ [^DP zibun wo] [V hihans]]] -hita]] C]]

Sentence 3: [CP [C′ [TP [^DP Yukino ga] [T′ [VP [t] [V′ [^DP mizukara wo] [V hihans]]] -hita]] C]]

Sentence 4 Original: [CP [C′ [TP [^DP Yukino wa] [T′ [VP [t] [V′ [CP [C′ [TP [^DP Kuribo ga] [T′ [VP [ti] [V′ [^DP zibun no koe ga] [V kikoe]]] -ta]]] to] omot]] -ta]]]]

Sentence 4: [MERGE [NORMAL Yukino] [MERGE [NORMAL wa] [NORMAL Kuribo [MERGE [NORMAL ga] [MERGE [^RFX-A zibun-no] [MERGE [NORMAL koe] [MERGE [NORMAL ga] [MERGE [^NORMAL kikoe-ta] [MERGE [NORMAL to] [^NORMAL omot-ta]]]]]]]]]

Sentence 4 v2: [MergeJP [LicR R-OK] [MergeJP [TypeAOK Yukino] [MergeJP [Normal wa] [MergeJP [TypeBOK Kuribo] [MergeJP [Normal ga] [MergeJP [^Zibun zibun-no] [MergeJP [Normal koe] [MergeJP [Normal ga] [MergeJP [^Normal kikoe-ta] [MergeJP [Normal to] [^Normal omot-ta]]]]]]]]]]

Sentence 5: [MERGE [NORMAL Yukino] [MERGE [NORMAL wa] [NORMAL Kuribo [MERGE [NORMAL ga] [MERGE [^RFX-B zibun-zishin-no] [MERGE [NORMAL koe] [MERGE [NORMAL ga] [MERGE [^NORMAL kikoe-ta] [MERGE [NORMAL to] [^NORMAL omot-ta]]]]]]]]]

Sentence 5 v2: [MergeJP [LicR R-OK] [MergeJP [TypeAOK Yukino] [MergeJP [Normal wa] [MergeJP [TypeBOK Kuribo] [MergeJP [Normal ga] [MergeJP [^ZibunZishin zibun-zishin-no] [MergeJP [Normal koe] [MergeJP [Normal ga] [MergeJP [^Normal kikoe-ta] [MergeJP [Normal to] [^Normal omot-ta]]]]]]]]]]

Sentence 6: [MERGE [NORMAL Yukino] [MERGE [NORMAL wa] [NORMAL Kuribo [MERGE [NORMAL ga] [MERGE [^RFX-C mizukara-no] [MERGE [NORMAL koe] [MERGE [NORMAL ga] [MERGE [^NORMAL kikoe-ta] [MERGE [NORMAL to] [^NORMAL omot-ta]]]]]]]]]

Sentence 6 v2: [MergeJP [LicR R-OK] [MergeJP [TypeAOK Yukino] [MergeJP [Normal wa] [TypeBOK Kuribo [MergeJP [Normal ga] [MergeJP [^Mizukara mizukara-no] [MergeJP [Normal koe] [MergeJP [Normal ga] [MergeJP [^Normal kikoe-ta] [MergeJP [Normal to] [^Normal omot-ta]]]]]]]]]]