How to choose article and number for nouns when translating from Japanese to Dutch in the context of syntax-based machine translation

L.D.S.P. Broers (3117243)
Supervisor: Jan Odijk (Utrecht University)
Second reader: Yoad Winter (Utrecht University)

13th October 2014

Abstract

When translating from a language which does not have distinction in number (singular/plural) or an indication of definiteness for nouns (Japanese), into a language which does have these distinctions (Dutch), it is not straightforward to make these decisions.

This problem in machine translation is studied using a compositional translation framework. This framework uses synchronized (isomorphic) grammars to deal with translation. As the grammars are reversible, no separate analysis and production for both languages are necessary. The basic expressions and rules in both languages correspond in meaning, and when rules with the same meaning are applied in both languages, the resulting sentences are each other's translations.

The grammar also uses features to indicate syntactic and semantic properties of the words and phrases. The rules check for certain settings of the features to decide whether or not they are applicable. If necessary, features of the resulting output expressions are set as well. A rule also has conditions, which all have to be met before a rule can be applied. These restrictions are intended to prevent ungrammatical combinations from being generated.

The features 'divisible', 'generic' and 'count/mass noun' are the ones the rules make the most use of in deciding number and definiteness for nouns.

The output created by the rules consists of all possible translations. A bonus system is used to order the possible translations by plausibility.

Contents

1 Introduction

Japanese is a language without definite and indefinite articles like 'the' and 'a' in English. Dutch is a language which has two definite articles ('de' for the grammatical male and female genders, and 'het' for neuter) and one indefinite article ('een'). When translating from a language without overt articles to a language which does have them, how is a system to decide which one to choose (and whether to insert an article at all)?

In Japanese, the same word can be used for singular and plural: in most cases, there is no difference between the singular and plural word. A human reader can easily understand when the author meant singular or plural by looking at the context of the word, but using context isn't very easy for a machine translation system. So, are there hints in the same sentence that a machine translation system could use to decide the correct number?

The main question of this thesis is "When do you have to insert a definite, indefinite or no article and when do you have to choose a singular or plural form for nouns when translating from Japanese to Dutch in the context of syntax-based machine translation?".

This is not a new subject, as it's one of the most obvious differences between Japanese and English. Many people have already written about this problem, which I review in section 3. The reason I am also looking into this topic, is that available machine translation systems still make errors in these areas.

However, I'm looking at translation between Japanese and *Dutch*, while so far other systems have looked at translation between Japanese and *English* (and sometimes another language, but not Dutch yet). Number marking and definiteness on their own aren't that different between English and Dutch, but to determine number and definiteness of nouns, it is necessary to use information from elsewhere in the sentence. And there, English and Dutch aren't exactly the same. The most notable difference is the word order:

In declarative sentences, Dutch has Verb-second word order, which means that the finite verb (which agrees with the subject) is in the second position of the sentence. The rest of the verbs are at the end of the sentence. In subordinate clauses, Dutch sentences have the Subject-Object-Verb word order. This same SOV order is used when the sentence doesn't start with the subject, but for example with an adverb.

English has Subject-Verb-Object word order in both declarative sentence and subordinate clauses. A related difference between English and Dutch word orders, is that Dutch show a great deal of scrambling, which appears to be typical of OV-languages [50], and scrambling can be of influence to determine definiteness of nouns in Dutch.

The usage of verb tenses also differs between English and Dutch, which sometimes causes a preference for a certain definiteness in a noun clause.

The fact that Dutch distinguishes two grammatical genders (male/female and neutral) for nouns, makes it necessary to choose between the definite male/female article 'de' or the neutral article 'het'. In English, there is only the definite article 'the'. This distinction also influences the morphology of the other words in the noun clause (articles, adverbs, adjectives). Even though most of the time, article use between Dutch and English seems similar, there are differences where English uses an article, while Dutch doesn't. An example are sentence structures with job names, for example in (1).

(1) Mijn vader is leraar. (Dutch)

*My father is teacher. (literal translation into English)
My father is a teacher. (correct English sentence)

A similar case occurs with generics: abstract generics in English often appear without an article, while in Dutch an article is often required, as in (2).

(2) Time flies. (English)
*Tijd vliegt. (incorrect translation into Dutch, without article)
De tijd vliegt. (correct Dutch sentence, with article)

Also, if all translation systems work by translating via English, this could make details from one language get lost in translation to the other language (which is something you can already see happening when you use an existing translation system like Google-Translate where it gives different results when you translate from language A to English to language B, instead of directly from language A to B).

The goal of this thesis is to make explicit rules that make use of grammatical information, and which can be automatically processed. If the cases for using articles and number can be represented in a schematic way that a computer will be able to parse easily, the translations might improve too. This is more 'syntax-based' than 'statistics-based', hence the title of this thesis.

I will first look at the context and background (chapter 2) and the related literature (chapter 3). Next, I will look at existing translation frameworks and define the one I will be working with (chapter 4). I propose here a compositional translation system based on the Rosetta framework, used by Philips in the 1980s (detailed in section 4.1.3). This framework uses synchronized (isomorphic) grammars to deal with translation. As the grammars are reversible, no separate analysis and production for both languages are necessary. There are basic expressions in both languages which correspond in meaning (a basic expression is, for example, a single noun), as well as rules that have the same meaning (which are called corresponding rules). When the corresponding rules are applied to the corresponding basic expressions and the output of other corresponding rules in both languages, the resulting sentences are each other's translations.

I do restrict this thesis' research to single sentences, as the Rosetta system and most other machine translation systems also work on single sentences. I briefly discuss this problem in section 5.3, because the choice between definite or indefinite also depends on the context from preceding sentences.

In chapter 5 and 6 I will look at ways of determining the correct article and number for nouns when translating from Japanese to Dutch. In chapter 7 I will give the details of the translation theory, including the compositional translation rules. This chapter is the main part of the thesis: the rules of the translation system, and the features that need to be attached to the words which are necessary for the rules.

The features are used to indicate syntactic and semantic properties of the words and phrases. Without these, it's not possible to make decisions about number and definiteness in the compositional translation framework.

The rules check for certain settings of the features to decide whether or not they are applicable. If necessary, features of the resulting output expressions are set as well. These restrictions are intended to prevent ungrammatical combinations from being generated.

The features 'divisible', 'generic' and 'count/mass noun' are the ones the rules make the most use of in deciding number and definiteness for nouns.

The output created by the rules consists of all possible translations. A bonus system is used to order the possible translations by plausibility.

The appendix contains topics related to the main topic of this thesis: rules for deciding which article a Dutch noun gets (chapter A), parsing and the internal structure of NPs (chapter B).

2 Context and background

The best-known machine translation system available at the moment is Google Translate [27]. It does not always translate articles and plurality correctly. An example can be seen in (3), where Google Translate does recognize that 'banana' is the fruit and should be translated as the Dutch word 'banaan' in 2013, but a year later it seems to be translated as a name (which could make sense, as there is a Japanese author called 'Yoshimoto Banana'. However, the author's name is written in hiragana characters, while I typed 'banana' in katakana characters). It doesn't make a choice for an article or for number. When changing the colour to something more plausible (4), in 2013 'banana' was translated as a name, and a year later as a fruit.

(3) banana-wa akairo datta. (entered into GoogleTranslate in Japanese characters)
banana-TOP red is-PAST
"A/The banana(s) was/were red."
Banaan was rood. (Google Translate Japanese to Dutch, 2013-02-19)
Banana was rood. (Google Translate Japanese to Dutch, 2014-01-27)
banana/'Banana' was red.

(4) banana-wa kiiro datta.
banana-TOP yellow is-PAST
"A/The banana(s) was/were yellow."
Banana was geel. (Google Translate Japanese to Dutch, 2013-02-19)
Banaan was geel. (Google Translate Japanese to Dutch, 2014-01-27)
'Banana'/banana was yellow.

Google bases its machine translation on statistics [71]. They apply statistical learning techniques to a large amount of text in both the source and target language, as well as aligned human-translated text, to build a translation model (and statistical machine translation models use monolingual data of the target language to create a language model of target language). They also allow a certain amount of user feedback [72]. However, no linguistic or grammatical knowledge is used directly in this way of translating. To train a statistical translation model, it needs texts which have already been translated by humans ('aligned texts' or 'parallel texts', the same text in both languages). The more human-translated texts are available, the better the statistical machine translation will be for new texts [28].

Until the 1990s, machine translation was mostly based on rules and heuristics, making heavy use of dictionaries containing relevant information. In the 1950s and 1960s, some statistics were used to discover grammatical and lexical regularities, but other approaches used a linguistic basis [35]. Most developed systems used a transfer or interlingua approach ([35], see also section 4 for descriptions of these approaches).

However, the 'pure syntax-based models' ran into limitations: a lot of syntactic information was necessary to be able to translate texts and a lot of ambiguities couldn't be solved on the basis of grammar alone. So in the last decade of the 20th century, research started to focus more on statistical machine translation, which since then has become the

main method for machine translation (though a number of research groups at universities have continued with transfer- and interlingua-based projects [35]).

In the statistical approach, the information for translating mostly comes from the language data itself. Probability models like Hidden Markov Models, Probabilistic Context-free Grammars and n-grams (see also section 4.2.2 for more on n-grams) can be used to describe probabilities in a language, but it is a topic of debate whether or not probabilistic information should be within the grammar itself [32]. In the rule-based view, the theory of a language can be described by rules for the syntax, morphology and semantics, and a translation dictionary.

Chomsky [17] claims that statistical models are only a simulation of linguistic problems because they don't look at the structure of the language, and thus don't give any insight into the language itself. Statistics like Bayesian models[1] give good results, but the models are very simple and don't really take into account the structure of languages.

Norvig [69] replies that it is possible to gain insight into the phenomenon of language by examining the properties of the statistical model. He does agree that a statistical (Markov) model alone can't model all of language, but only a tree-structure[2] model can't do that either, so Norvig says that "a probabilistic model[3] which covers words, trees, semantics, context, discourse, etc. would be the best tool for representing facts about language, for algorithmically processing language, and for understanding how humans process language." [69].

Chomsky also states that when you integrate statistical analysis with fundamental properties of languages (like the prosodic cues that mark word boundaries), you do get better results [17].

This integration can also be seen in the hybrid approaches to machine translation. Because it is very difficult to cover all possible constructions and ambiguities in a language using only a rule-based approach, something else is needed to be able to create better translations: world knowledge or statistics, though it has been tried to create machine translation systems using only statistics.

Nowadays most of the commercial systems use a hybrid of statistically-trained and rule-based approaches. All competitors in machine translation competitions use statistical methods [69].

Norvig [69] writes that "of the 4000 language pairs covered by machine translation systems, a statistical system is by far the best for every pair except Japanese-English, where the top statistical system is roughly equal to the top hybrid system."

The fact that a purely statistical system wasn't the best for Japanese-English, shows that at least for Japanese it is also useful to look at a syntax-based approach.

An idea for a hybrid syntax/statistical system can be to first use syntactic rules as far as possible (to produce a list of possible translations), and then (for example) sort those sentences by most common words or word combinations using statistics.

[1] A Bayesian model calculates the probability that a variable will take on a specific value. The most simple model, Naive Bayes, assumes the variables are independent [91].

[2] Representing sentences as tree structures is a common method in linguistics.

[3] A Bayesian model is an example of a probabilistic model.

3 Related literature

Most literature focusing on translating articles and number when translating is about Japanese to English. I have not come across any which is about translating from Japanese to Dutch directly.

Nishida [68] translates from Japanese to English and has information in their 'word dictionary' about the conventional use of articles for every noun.

Rösner [87] has written a paper about a prototypical Japanese to German translation system for titles of Japanese papers they had worked on for 2.5 years. The standard structure of those titles are noun phrases, but they still have to infer missing articles and plurals. For this, they use heuristics. The heuristics they give as an example in their paper are the following.
'SLEX' is their semantics-to-German dictionary. :OBJECT and :NAME are representations in this semantic dictionary.

- a nominalized case frame has to be realized with definite article in singular ("*Die Generierung* natürlicher Sprache").
- the :OBJECT role of a nominalized case frame should be realized indefinite and plural ("Die Generierung *von Titeln*'), except in cases with an exception information in SLEX ("Die Wartung *von Software*").
- concepts that have a :NAME role will be realized definite and singular ("*Die Fourier-Transformation*").
- If no heuristic is applicable and if no SLEX information is found we use as title defaults 'indefinite' and 'singular' ("*Ein Verfahren*").

Murata e.a. [66] also uses heuristics to decide which article and number to use for nouns when translating from Japanese to English. Their heuristics are rules that create non-exclusive categories [66]), so the categories resulting from the rules could overlap (different rules could apply to the same sentence). They first transform the sentences into dependency structure representations. These structures are used to first decide the referential property for each noun (generic or non-generic and definite or indefinite), then the number (uncountable or countable and singular or plural). They use 86 heuristic rules for the referential property and 48 heuristic rules for the number. Their rules look at context (other words in the same sentence), as illustrated in the rules below. Their system applies all possible rules for a certain noun and looks at the accumulated scores to make a decision. The default values are 'definite' for the referential property and 'singular' for the number. The first number between brackets is the possibility (1 or 0), the second number is the plausibility (value of 0 to 9).

- When a noun is accompanied by a particle (WA), and the predicate has past tense, then {indefinite (1,0), definite (1,3), generic (1,1)}
  Example: *INU-WA(dog)* MUKOUE(away there) IKIMASHITA(went)
  The dog went away.

- When a noun is accompanied by a particle HE(to), MADE(up to) or KARA(from), then {indefinite (1, 0) definite (1, 2) generic (1, 0)}
  Example: KARE-O(he) KUUKOU-MADE(airport) MUKAE-NI(to meet) YUKI-MASHOO(let us go)
  Let us go to meet him at *the airport*.

- When a noun is accompanied by a particle (WA), and the predicate has present tense, then {indefinite (1,0), definite (1,2), generic (1,3)}
  Example: *INU-WA* YAKUNITATSU(useful) DOUBUTSU(animal) DESU(is)
  Dogs are useful animals. (but: 'a dog' and 'the dog' are also possible because of the generic subject)

- When a predicate, SUKI(like), TANOSHIMU(enjoy), etc. has a generic noun as an object[4], and the noun is accompanied by GA(for SUKI), or WO(for TANOSHIMU), then {singular (1,0), plural (1,2), uncountable (1,0)}
  Example: WATASHI-WA(I) *RINGO-GA(apple)* SUKI-DESU(like)
  I like *apples*.

They also note that definiteness of a noun can be decided by the meaning of the noun itself ('chikyuu' - the earth, 'uchyuu' - the universe[5]), or that a noun is definite if it has appeared before ('He has a car and a truck, but only the car is insured').

Bond e.a. [8] also uses heuristics to improve the translations for articles and number in the ALT-J/E machine translation system, which improved the accuracy by 8% in comparison to the system before Bond's article/noun heuristics were added. In his book [6] he notes that the accuracy of translating determiners (articles and possessives) became 85% because of his additions to the system, but that it made no sense to try and improve on the determiners before other parts of the translations were improved.

Knight and Chander [44] found that when 'the' was inserted to all article locations in an English text with the articles replaced by blanks, 67% was correct. When humans have to guess the missing articles, their scores are between 94% and 96%. These results would imply that in 4-6%, there are several options for the articles. If you use a simple algorithm that always chooses 'the', 67% of the articles will already be correct.

[4]If the noun which acts as an object for these verbs is not a proper noun or specified using a word like 'that', then it's a generic noun.

[5]'Uchyuu' is not always definite singular, as the same word can be used for words like 'daitai-uchyuu' (alternate universe(s)) or 'ryoushi-uchyuu' (quantum universe(s)).

4 Translation framework

4.1 Rule-based machine translation methods

Rule-based translation methods use rules to perform the translation from one language into another. The rules are written by humans with (much) linguistic knowledge, so the strength of this method is that it's good at syntactic and semantic analyses, but it's difficult and in practice not feasible to write rules that cover the entire language [15].

The rules can be applied in a certain order to prevent linguistically incorrect results, but most systems don't have ordered rules. A sentence is translated by repeating pattern matching and applying transformations of the sentence's syntactic/semantic tree structure [40].

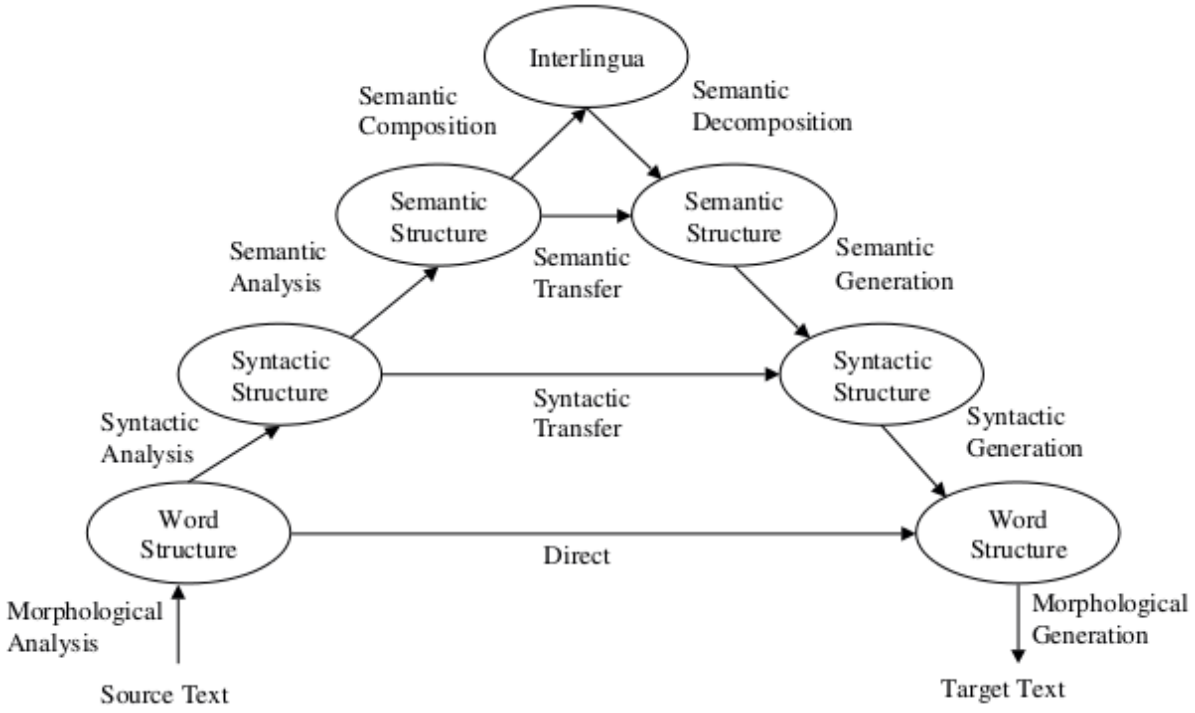


Figure 1: The possible levels of analysis in machine translation in the so-called 'Vauquois triangle', a figure taken from [20], based on the one created by Bernard Vauquois in 1968 in a slightly different form [36]. The arrows in the center and the node at the top contain the translation methods.

The source text can be analyzed on several levels, as can be seen in figure 1. After the analyses, rules/heuristics and algorithms are used to create the syntactic structure of the target language ('syntactic transfer') or the semantic structure ('semantic transfer') [94]. The words are replaced by words in the target language and the words are re-ordered and morphological changes are applied to make them fit into the target language's syntax.

In section 4.1.1, I describe the syntactic and semantic transfer methods. These transfer methods are used when the analysis is done on word structure, syntactic structure and/or semantic structure. One step higher up in the Vauquois triangle, an interlingua model is

used, where the source language text is translated into an abstract, language-independent representation. This model is described in section 4.1.2.

4.1.1 Transfer method

The transfer model was widely used from the mid-1960s till the end of the 1980s [94]. This model is a bit more complicated than just translating word by word (including some morphological analysis and maybe some word rearrangements), which is called a direct or dictionary approach.

The drawback of a transfer model is that it needs rules for every translation direction. For example, to translate between Dutch and English you need a model for Dutch to English and another model for English to Dutch. If you add a third language, you need models for Dutch-French, Dutch-English, French-Dutch, French-English, English-Dutch and English-French. If you add a fourth, you'll need even more models. So for n languages you need $n * (n-1)$ transfer models. Using an interlingua model, which uses an abstract intermediate representation (further explained in section 4.1.2), is advantageous if you want to translate between more than three languages.

The 'semantic transfer' model was widely used in Japan in the 1990s. In this, the source text was analyzed syntactically and (at least partially) semantically prior to the transfer phase. In the transfer phase, the semantic representations of both the source and target texts were mapped onto each other [13]. The semantic transfer gives better results than the purely syntactic transfer method, where the source text is only parsed into syntactic structures, because some ambiguities in meaning are solved in the semantic parsing stage.

When translating using the transfer approach, all information from the entire sentence is available everywhere, so at any place in the sentence you can use information from other places in the sentence (which is not always the case in the compositional translation approach described in section 4.1.3). For example, if you take the sentence in (5), the first thing you have to decide is the words. As Japanese doesn't use spaces except in books for very young children, it first needs to be decided which characters make up the words (which is not a trivial step, but I assume that a module is used to split up the text in words, for example a morphological parser like ChaSen/IPADIC [78], Juman [79] or MeCab/Unidic [80]). The dictionary should contain at least some basic information about the words and their translations, for example as in (6).

(5) 石が白いです。(The sentence as appearing in a Japanese text)
    石　が　白い　です(The sentence split into words)
    ishi ga shiroi desu (the words in our alphabet[6])
    stone SUBJ white to-be (English gloss)

(6) 石(ishi), N, *de steen / een steen / de stenen / stenen*
    が(ga), PARTICLE indicating that preceding phrase = SUBJECT

[6]Japanese kanji-characters can be pronounced in different ways, so-called 'on' and 'kun' readings. It is not really necessary to know how a kanji is pronounced if the word and its translation are in the dictionary, though a large Japanese dictionary like JMdict [11] does contain the pronunciation as well.

白い(shiroi), ADJ, *wit / witte*
です(desu), $V_{auxiliary}$, *zijn*

The next step is going to be the translation into Dutch, but this is where the difficulties start: which choice should be made for the translation of *ishi*? Should it be singular or plural and should an article be added and if so, which one? And the verb form depends on the choice for the noun form, so to translate the verb into the correct verb form, information from the rest of the sentence is necessary. To decide on solutions for these translation problems, rules are created which are applied in a certain order (all systems described in chapter 3 use these kind of rules).

A theoretical example of a Japanese-to-Dutch transfer system could work like this:

1. 石が白いです。
   Source language = Japanese; Target language = Dutch.
   Now we need
   - a parser that decides which characters belong together to form words.
   - a dictionary that maps all Japanese words to Dutch words.
   - rules for the regular Japanese sentence structure.
   - rules for the regular Dutch sentence structure.
   - rules to relate the Japanese and Dutch structures to each other.
2. Get basic part-of-speech information about each word.
   - 石(ishi), N, *steen, singular; stenen, plural*
   - が(ga), PARTICLE indicating that preceding phrase = SUBJECT
   - 白い(shiroi), ADJ, *wit*
   - です(desu), $V_{auxiliary}$, *is, singular; zijn, plural*
3. Get syntactic information about the verb 'desu'.
   - Auxiliary verb
   - Polite form
   - At the end of a sentence:
     - NP-ga-NP-desu
     - NP-ga-ADJ-desu
     - NP-desu
     - ...
   - Can be followed by end-of-sentence particles like 'ka' (question particle).
4. Parse the source sentence:
   - ishi ga shiroi desu = NP-SUBJ ADJ V_{aux}
5. Translate Japanese words into Dutch:

- ishi (category = N) ⇒ steen/stenen (category = N)
- ga (category = subject particle) ⇒ (no equivalent in Dutch for nouns)
- shiroi (category = ADJ) ⇒ wit (category = ADJ)
- desu (category = V_{aux}) ⇒ is/zijn (category = V_{aux})

6. Map the translated words onto a basic Dutch sentence structure:
   - Literally translated: *steen/stenen wit is/zijn*
   - Dutch sentence order: *steen/stenen is/zijn wit*
7. Choose correctly inflected forms. The system has rules to decide which one to choose; maybe it chooses singular by default. Only one of the options is chosen.
   - Singular: *steen is wit*
   - Plural: *stenen zijn wit*
8. See if there are any rules in the system about whether to add articles:
   - Singular: *de steen is wit*
   - Plural: *de stenen zijn wit*

4.1.2 Interlingua model

In the interlingua approach, the source language text is translated into an abstract, language-independent representation, which is then used to create the target language text. After both a complete syntactic and a complete semantic analysis are done, it is possible to create an 'interlingua' (a meaning representation not containing specific elements of the source and target language). The interlingua can contain just the meaning, or also a language-independent description of the linguistic form (for example 'active' or 'passive') that was used in the source text so that effects such as focus can be recreated properly in the target [13]. From this interlingua representation, the text can be translated into one or more target languages. So, $n + n - 1$ transfer models are needed.

The difficulty with the interlingua approach is how to exactly define 'meaning' in a language-neutral way and how detailed the representation should be, as for example different languages make different distinctions; the English 'wear' can be translated into different words in Japanese depending on where the object is worn (as can be seen in table 1) [20, 36]. You could say the English word 'wear' is tenfold ambiguous and has ten different meanings, or that English 'wear' is not ambiguous, but that all the Japanese translations have the same meaning and that a form is chosen depending on different conditions formulated in terms of the target language (Japanese) elements.

To understand meaning, world knowledge needs to be stored as well, but this has to be entered into a system by humans, which requires a lot of work and so the biggest problem with this approach is the incompleteness of data for non-specialized domains [20].

Japanese	Object which is worn
haoru	coat, jacket
haku	shoes, trousers
kaburu	hat
hameru	ring, gloves
shimeru	belt, tie, scarf
tsukeru	brooch, clip
kakeru	glasses, nechlace
hayasu	moustache
kiru	general, unspecific

Table 1: Different words in Japanese as translations for the English word 'wear' and/or 'put on', depending on the object which is worn [36].

One commercial system, used in the middle 1990s, used the interlingua approach: KANT [20, 70]. It uses explicitly coded lexicons, grammars, and semantic rules to perform translation from Controlled English into multiple languages. For the interlingual system, it uses "an explicit intermediate representation which acts as a 'pivot[7]' between the source and target languages" [20, 70].

4.1.3 Compositional translation framework

In the compositional translation method, a set of basic expressions and syntactic rules are specified for each language. It is called 'compositional' because the meaning of an expression is composed of the meaning of its parts (basic expressions) and the way they are combined. A basic expression consists of a form and a meaning. It can be a single word or a few words, like fixed idioms (for example *kant-en-klaar* ('ready-made')).

Taking the example from (5), the basic expressions would be the following. The basic expressions are written as a B followed by the word itself.

Japanese basic expressions	Dutch basic expressions	basic meanings
N(Bishi)	N_{sing}(Bsteen)	*stone'*
N(Bishi)	N_{plur}(Bstenen)	*stones'*
ADJ(Bshiro)	ADJ(Bwit)	*white'*
$V_{auxiliary}$ (Bdesu)	$V_{auxiliary,sing}$ (Bis)	*is'*
$V_{auxiliary}$ (Bdesu)	$V_{auxiliary,plur}$ (Bzijn)	*are'*

The translation rules in Rosetta are local, so if there are any dependencies on items in other nodes, they have to be solved locally in the grammar.

To translate, you need corresponding basic expressions, as well as rules that correspond in meaning (these are called 'corresponding rules', as they are applied at the same moment in each language's derivation). Then you build syntactic structures in parallel for both languages, by sequentially applying rules to the basic expressions or to the parts

[7] A pivot is an element around which a list of items is divided [18]. In this case, the source language is on one side of the pivot and the target language on the other side, so the pivot is the interlingua.

that have been created by using rules. When you have applied the same meaning rules in both languages, the results are each other's translation [86].

For the example sentence, the rules (following the simple example system used in Rosetta [86], which is a simplified version of the real Rosetta system and doesn't use (surface) trees, but only labelled strings of the form $X(\alpha)$) could be:

- Japanese rules:
  $R_{Japanese}1 : N(\alpha) \Rightarrow NP(\alpha)$
  $R_{Japanese}2 : ADJ(\alpha) + V_{auxiliary}(\beta) \Rightarrow VP(\alpha\, i\, \beta)$
  $R_{Japanese}3 : VP(\alpha) + NP_{subj}(\beta) \Rightarrow S(\beta\ ga\ \alpha)$

- Dutch rules:
  $R_{Dutch}1a : N_{sing}(\alpha) \Rightarrow NP(de\ \alpha)$
  $R_{Dutch}1b : N_{sing}(\alpha) \Rightarrow NP(een\ \alpha)$
  $R_{Dutch}1c : N_{plur}(\alpha) \Rightarrow NP(de\ \alpha)$
  $R_{Dutch}1d : N_{plur}(\alpha) \Rightarrow NP(\alpha)$
  $R_{Dutch}2 : ADJ(\alpha) + V_{auxiliary}(\beta) \Rightarrow VP(\beta\ \alpha)$
  $R_{Dutch}3a : VP_{sing}(\alpha) + NP_{sing}(\beta) \Rightarrow S(\beta\ \alpha)$
  $R_{Dutch}3b : VP_{plur}(\alpha) + NP_{plur}(\beta) \Rightarrow S(\beta\ \alpha)$

Two grammars are isomorphic if each basic expression in one grammar has at least one meaning-equivalent basic expression in the other grammar, and each rule in one grammar has at least meaning-equivalent rule in the other grammar [86]. So, if corresponding rules have the same meaning and corresponding basic expressions have the same meaning, then the translations of the sentences are guaranteed to have the same meaning.

In this method, the grammars/rules for both languages have to be developed simultaneously to be able to assure isomorphy between both grammars, which is not the case with other approaches. The grammars are also reversible, so you don't need separate grammars for translating from one language to another and the other way around.

In the following table, the grammars/rules of both languages are mapped to their meaning, so indirectly they're mapped onto the rules of the other language. When rules are on the same row, they have the same meaning.

Japanese syntactic rule	Dutch syntactic rule	meaning rule
$R_{Japanese}1$	$R_{Dutch}1a$	S1a
$R_{Japanese}1$	$R_{Dutch}1b$	S1b
$R_{Japanese}1$	$R_{Dutch}1c$	S1a
$R_{Japanese}1$	$R_{Dutch}1d$	S1b
$R_{Japanese}2$	$R_{Dutch}2$	S2
$R_{Japanese}3$	$R_{Dutch}3a$	S3
$R_{Japanese}3$	$R_{Dutch}3b$	S3

Mappings between the basic expressions (for the translation):

Japanese basic expression	Dutch basic expression	meaning
$N(Bishi)$	$N_{sing}(Bsteen)$	*stone'*
$N(Bishi)$	$N_{plur}(Bstenen)$	*stones'*
$ADJ(Bshiro)$	$ADJ(Bwit)$	*white'*
$V_{auxiliary}(Bdesu)$	$V_{auxiliary,sing}(Bis)$	*is'*
$V_{auxiliary}(Bdesu)$	$V_{auxiliary,plur}(Bzijn)$	*are'*

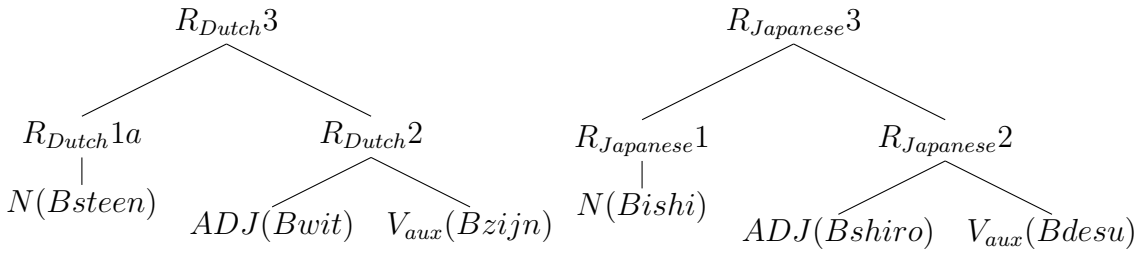
Adding the distinctions 'singular' and 'plural' to the rules in this simple example isn't a problem, but if you add these (and more) distinctions to the rules this way, but this will cause an explosion of features in a larger system. Then you get many similar rules where only the (number-)feature differs.

In the compositional translation method used in Rosetta, these features aren't listed in the subscripts of the rules, but the words themselves have attributes like number (singular/plural/unspecified), form (finite, participle, infinitive, etc) and tense (present, past, unspecified, etc.) [86]. The problem is of course that Japanese makes less distinctions than Dutch, so for example nouns don't automatically have the attribute 'plural'. These things need to be decided when analyzing the sentence prior to translating.

The two sentences (in both languages) are each other's translation, because they have syntactic derivation trees (D-trees) with the same geometry and corresponding basic expressions at the leaves and corresponding rules at the nodes ('isomorphic trees'). In the trees below you can see two syntactic derivations for the example sentence, and the corresponding semantic derivation tree. The syntactic derivation trees consist of the names of the basic expressions and the names of the syntactic operations, while the semantic derivation trees contain the meanings of the basic expressions and the meaning rules (see the tables above). By applying these rules, you get a derived tree (S-tree). A derivation tree is shows how a sentence is derived: it specifies which rules are applied to which basic expressions and results from earlier appliances.

The semantic derivation trees contain all the information necessary for the translation, so they play the role of an interlingua.

Example syntactic derivation trees:



Example semantic derivations:

The interlingua is not independently defined, but it is a result of creating two isomorphic grammars. So when you add a third language, it can still cause changes in the already existing grammars, for example if the third language has a semantic distinction that doesn't exist in the other two languages [86].

When translating in the compositional translation framework, the only information you can use in each node, is available within that node of the syntactic derivation trees itself.

However, if you construct your grammar well, you can get information about the entire sentence at the top level of the tree, which is called the surface tree (S-tree) level in Rosetta: a tree in which constituency (phrases), grammatical relations, syntactic categories, attribute:value pairs (like *tense:past*) and linear order are all represented [86].

The S-tree of the sentence above would look like figure 2 in a graphical representation:

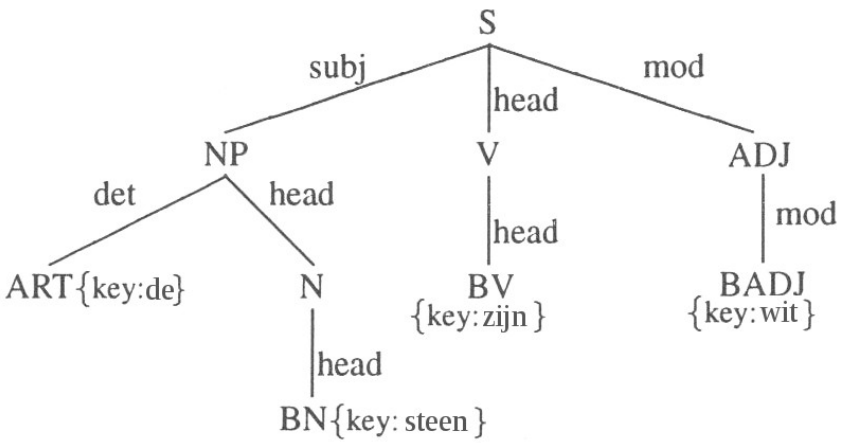


Figure 2: Graphical S-tree of the sentence 'De steen is wit', according to the graphical S-tree representation used in the Rosetta system [86].

In the S-tree, the leaves correspond to the words in the actual sentence. The 'B' in front of most of the leaf nodes means that it's a *basic* expression.

The Rosetta system generates all possible translations, but there are differences in quality. Rosetta's priority is preservation of the sentence's meaning, so they accept differences in syntactic form (like active/passive) if they are inevitable [86]. The rules are local, but the syntactic differences appear on a global level. They thus use a 'preference' or 'bonus' system to order all possible translations.

With every rule application, the D-tree can receive a bonus value which is increased or decreased every time a rule is applied to the sentence. In the default case, the bonus value of the D-tree is 0 and rules don't change the value. Only under certain conditions

a rule gives a bonus. Then, the bonus value changes: each parent node gets the sum of its children's bonuses plus the bonus assigned by the rule applied in that node. If there is more than one output sentence at the end, the bonus values determine the order in which the output sentences appear [86].

For example, the two English output translations for the Dutch sentence 7 can be seen in 8. The bonus given by the Rosetta system is higher for sentence 8.(1), so that one gets ordered higher. They do try to preserve the syntactic order of arguments as much as possible.

(7) Twee kinderen aten veel snoepjes niet op (Dutch sentence)
Two children ate many sweets not up (word-by-word English translation)

(8) (1) Two children did not eat many sweets
(2) Many sweets two children did not eat

The bonus system can also be used to rank preferred syntactic constructions, for example give a lower rank to passive constructions in Spanish, certain topicalizations in English, or the (2)-sentences in example 9 and 10 below [86].

(9) de *mooie* vrouw (the beautiful woman)
(1) the *beautiful* woman
(2) the woman *that is beautiful*

(10) zij *kan* komen (she can come)
(1) she *can* come
(2) she is *able* to come

4.2 Syntax-based statistical machine translation

In the last decade, more researchers have started looking into the so-called 'syntax-based statistical machine translation', in which syntax is combined with statistics to improve the translations. Figure 3 shows different options that can be used. The structure of the triangles in this figure has obviously been based on the Vauquois triangle mentioned earlier (figure 1).

4.2.1 Word-based

The basic idea is that each word is translated into its corresponding target language word and that these translated words are reordered to create a sentence in the target language. The simplest word-by-word translation systems contains word lists for all possible word forms (eat, ate, eaten, ...), as it just looks up every word in the dictionary and returns the translations in the same order as the source sentence. A slightly more sophisticated word-based translation system analyzes the words morphologically prior to looking them up in the dictionary, so only the base forms (like 'eat') need to be in the dictionary and rules are needed for things like tense and number [96]. Typically, no large reorderings take place and there is a strong localization effect [105], which means that word clusters in a sentence tend to stay in the same place in the target language sentence (as the system does not make (much) use of rules which try to change the order

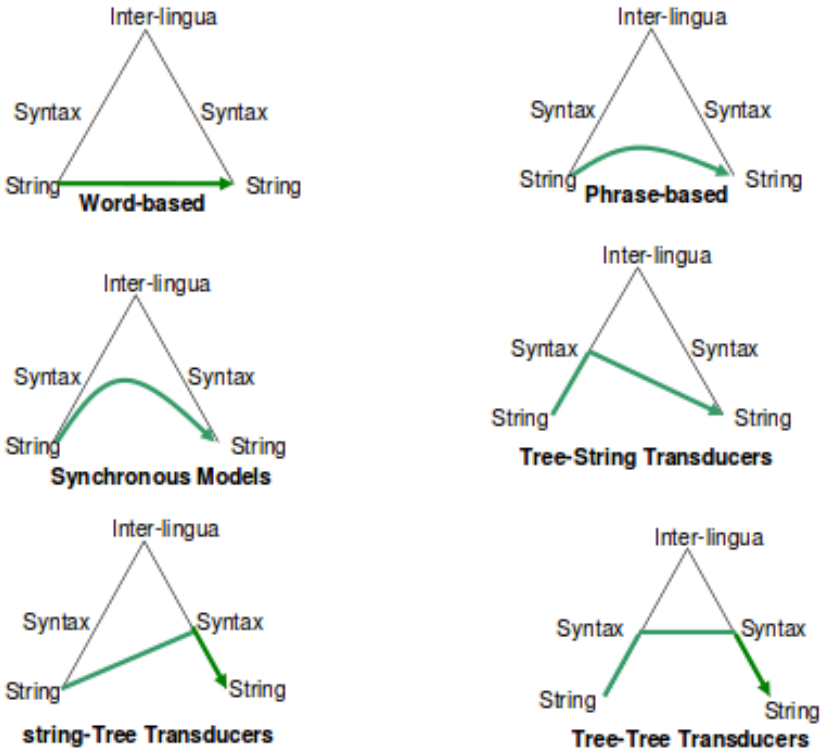


Figure 3: Statistical machine translation models using syntax. The arrows indicate the direction of the translation model (decoding goes in the other direction). From [3].

of the words in a translated sentence). In the word-based method, each word is assigned a probability of how often it occurs in the language (in the source texts used for training the system). The lexicon probabilities are based on single words [105].

4.2.2 Phrase-based

The basic idea is that a sentence in the source language is split into phrases, those phrases are translated, and then the translations of those phrases are combined (and may be reordered) into a new sentence in the target language. A way of splitting a sentence into phrases can be by using n-grams. An n-gram (most often 2-gram, 3-gram or 4-gram) is a consecutive sequence of n items (in this case, n consecutive words in a sentence). A 'phrase' can be just a sequence of words (n-gram), and it doesn't have to be syntactically motivated [105]. Punctuation marks are also treated as words, so when you create n-grams you get a lot of 'phrases' containing dots and commas. This can be useful to see which kind of words are often preceded by a comma, for example. Because the word-based model doesn't really use the context for translating, it makes errors when the translation of a certain word depends on the surrounding words. A way of looking at context is looking at n-grams.

Koehn e.a. [47] used a language model based on 3-grams. When you use 2-grams, you get a lot of combinations like 'in the', 'this is', 'is a', 'of a', 'with a', etc. These phrases are not very informative, as it's not clear in which contexts they are used. With 4-grams, the amount of occurrences becomes very low very quickly, so 3-grams is the best choice.

Koehn [46] notes that not all grammatical problems in machine translation can be solved using n-grams. He notes that a system might produce good n-grams, while the grammar of the sentences is incorrect.

Small word order changes are usually done well by using the phrase-based statistical machine translation, but word order changes over long distances fail most of the time [26]. Local reordering and local idiomatic expressions are strengths of the phrase-based models in comparison to the word-based models [82].

To get around small word-order changes, you could use unordered n-grams, called n-perms [14, 42]. N-perms are all the permutations of the n-grams. Unordered n-grams are not useful when the order of words is important, for example in the sequence 'article, adjective, noun' in English. However, they can be useful when you want to check how often certain words occur together and the order doesn't matter (for example, in languages with a (partly) free syntactic phrase order where a phrase is just one word), or split verbs (which occur frequently in Dutch). A Dutch example is 'afstappen' (to get off), which gets split into 'stap [...] af', with possibly a lot of words between the brackets. In English you generally don't have such large distances between words belonging together, so looking at (un)ordered n-grams might be enough to notice if a word is missing in a sentence. What does occur in English (and is a large source of mistakes for non-native speakers) are prepositions belonging to certain verbs, for example 'give [a discount] *on*'. The verb 'give' can occur with many different prepositions, but if [give a discount] appears in the sentence, there is a large chance that 'on' is present in the sentence somewhere as well (though not necessarily; you can also have a sentence like 'He gave me a discount').

Lin e.a. [51] created unordered ('rotated') n-grams to find all n-grams containing a certain word. All n-grams they found were rotated (each word in the n-gram appeared in the first position once) and these rotated n-grams were all stored with the count and POS-tags of the original n-gram. At the end they sorted the rotated n-grams, so all n-grams which contained the target word (even it wasn't at the first position in the original n-gram) were listed consecutively in the sorted output.

An example of a rotated n-gram (from Lin e.a. [51]) is:

```
faster than a cheetah
than a cheetah >< faster
a cheetah >< than faster
cheetah >< a than faster
```

Liu e.a. [54] mentioned that using "a syntax-based language model improved the fluence and semantic accuracy". Khalilov e.a. [43] compared a syntax-augmented translation system with an n-gram translation system. 'Syntax-augmented' means that there is only proper linguistic syntax (syntax with noun phrases NP, verb phrases VP, etc) in the output language [65]. The syntax model in Khalilov's experiments had more missing words than the n-grams system (25.17% missing words in the syntax-augmented translation and 15.44% in the n-grams, mostly content words) and had a bit more problems with unknown words (21.85% errors in the syntax system, 17.45% in the n-grams system). It did make correct word choices (including word forms) more often (27.15% errors in the

syntax system, 34.23% in the n-grams system) and also created the correct word order more often (15.89% errors in the syntax system, 23.49% in the n-grams system). However, the running time of the syntax-augmented algorithm was longer than the n-gram one. Koehn e.a. [47] also tried to limit the phrases their program learned to only syntactic phrases (subtrees of a syntactic parse tree), so phrases like "house the" were filtered out. However, this did not improve the results - it scored worst of all their models. They assume it's because their system had less data to work with in that case.

4.2.3 Synchronous models

Synchronous models have tree structures in both the source and target language models, but they don't have an interlingua.

According to Razmara [82], syntax-based models have two subcategories: synchronous-grammar-based models and tree-transducer-based models. A synchronous model can be a [82]:

- **synchronous context-free grammar (SCFG)**
  A context-free grammar consists of a finite set of production rules of the form $A \rightarrow \alpha$, where A is a non-terminal symbol and α is a sequence of terminal and non-terminal symbols, and there is exactly one terminal symbol on the left hand side [37].
  A synchronous context-free grammar is when you have a context-free grammar of both languages and they're aligned ("synchronized") to each other, as in example 4 below.

Dutch CFG:	Japanese CFG:
S → NP VP	S → NP VP
NP → de steen	NP → ishi ga
VP → V ADJ	VP → ADJ V
ADJ → wit	ADJ → shiroi
V → is	V → desu

Figure 4: Example of a synchronous context-free grammar. On the left there is the context-free grammar for the Dutch sentence 'de steen is wit' ('the stone is white'), on the right the context-free grammar for the corresponding Japanese sentence 'ishi ga shiroi desu'.

- **synchronous tree-substitution grammar (STSG)**
  A synchronous tree-substitution grammar is a translation model with pairs of elementary trees containing terminal and non-terminal symbols, and the non-terminals in those two trees are linked. A rule is only applied to linked non-terminals [56, 16]. One of those trees can contain words in the source language, while the other tree contains the words in the target language. An example can be seen in figure 5.

- **synchronous tree-adjoining grammar (STAG)**
  A tree-adjoining grammar is a model in which part-of-sentence trees can be combined in specific ways (*adjoined*: a tree is inserted in another tree, moving down the subtree of the node it replaces (which is only possible if the node labels correspond), or *substituted*: a non-terminal node of a tree is replaced by another tree, which is also only possible if the node labels correspond) [39] . The part-of-sentence trees are small trees with English words at (specific) leaves and show the structure of the sentence in which those words should appear. This makes it possible to have long-distance dependencies (trees with gaps in them), necessary if an expression expects some other word to fill in the gap.
  A synchronous tree-adjoining grammar is created by pairing the elementary trees of the 'natural language' (syntactic tree) and 'logical form language' (semantic tree) grammars and linking the corresponding nodes, resulting in a new grammar whose elements are linked pairs of elementary trees [88]. Not all syntactic nodes have to be linked to a semantic node, only if there is an operation that must occur at both ends of the link, and it's also possible for a syntactic node to link to more than one semantic node and the other way around.
  STAG can also be used for machine translation by linking source and target language tree structure together (as can be seen in figure 6). TAG requires $O(n^6)$ time required for parsing, but they also mention that it requires $O(2n^6)$ time for bilingual parsing in the worst case, which is only needed when developing the system, not when using it afterwards [19].

- **generalized multitext grammar (GMTG)**
  Texts that are translations of eachother are called a 'multitext'. Multitext grammar (MTG) models generate arbitrarily many parallel texts via production rules of arbitrary length to parse texts for translational equivalence (to find out which components of both texts are equivalent) [62]. *Generalized* multi-text grammar generalizes MTG by allowing rewrites of string tuples instead of just single strings, and it is notated slightly differently (with tuples, for example as $[(S),(S)] \rightarrow [(PN\ VP),(PN\ VP)]$ with the left side being the source language and the right side being the target language) [63].

4.2.4 Tree-string transducers

A tree-to-string transducer is a system which starts with the syntax tree of a sentence in the source language and uses tree-to-string templates to transform it into a string in the target language [55]. The target string doesn't have any labelling. In figure 7 there is an example of a template used by a tree-to-string transducer (from [55]). Such a template always gives the same result wherever it is applied. On the left-hand side of the template, there is a subtree pattern. On the right-hand side there is a sequence of variables and words in the target language (没 means 'not' and 有 means 'is/are'). The variables are transformed by applying other templates, until no variables are left. The templates are created using statistical algorithms which also make use of n-grams for the target language [55].

4.2.5 String-tree transducers

A string-to-tree transducer is a system which starts with a string in the source language and creates a tree structure in the target language. The tree in the target language is created by applying tree-forming rules [25]. An example can be seen in figure 8.

4.2.6 Tree-tree transducers

A tree-to-tree transducer contains rules for copying, transforming, deleting, reordering, and translating subtrees [29, 82]. The 'translation' operation translates all the words in the leaf nodes into the target language.

An example of translating an English sentence into Japanese using a tree-to-tree transducer can be seen in figure 9 from [102]. First, the word order of the English sentence is changed so it looks more like the Japanese word order. Then the particles, which don't exist in English but are necessary in Japanese, are inserted. Next is the translation of the English words into Japanese words and finally the translation can be read from the nodes. There are more possibilities for word and particle choices during translation, but the transformation rules have probabilities attached, so the ones with the highest probability are chosen [45].

4.3 Translation method choice

4.3.1 Parsing and dictionary

Whichever translation method I choose, the Japanese text first has to be split into words before it can be translated. First, the sentence should be parsed; split into words and those words should get information attached to them, for example part-of-speech (POS) tags, which can be general or more specific.

After parsing, there's the option to create a tree-structure of the source language sentence and use that to create a string or tree structure as the translation into the target language.

It is possible to parse the text using an existing parser and then (manually or automatically) add annotations to the tags it gives to the words to improve translation quality [34]. Because the translation of an NP can also depend on things outside of the NP-phrase, an idea would be to add information about number and article to the NP-tags, derived from hints in the rest of the sentence, which could then be used to translate them correctly locally. To decide which information you have and don't have available, a translation method/framework choice needs to be made.

Another important thing to decide is which information is available in the dictionary. At least the following information is necessary:

- Word type: noun, verb, case suffix (though case suffixes can also be introduced by rules).
- Grammatical form (inherent features like gender and mass/count noun distinction, and non-inherent features like number, case and tense). Non-inherent features do not necessarily have to be in the dictionary, as that information can also be in rules

(for example, a rule about whether or not a noun can be made plural, for which mass/count noun information is also necessary). The stem of the verb can be in the dictionary, so the tense can be in the rules and be translated separately.

- Translation/meaning.

There is one dictionary available which contains Japanese words and translations into several languages (all Japanese words have an English translation and some also have a Dutch translation), namely JMdict [11]. This is a downloadable dictionary in XML-format. However, the dictionary isn't immediately usable for (word-by-word) machine translation. Especially the Dutch entries look more like dictionary entries meant for humans, as can be seen in the example from the JMdict [12] in (11).

(11)
```
<entry>
<ent_seq>1014740</ent_seq>
<r_ele>
<reb>アウトコース</reb>
</r_ele>
<r_ele>
<reb>アウト・コース</reb>
</r_ele>
<info>
<audit>
<upd_date>2013-05-10</upd_date>
<upd_detl>Entry created</upd_detl>
</audit>
</info>
<sense>
<pos>&n;</pos>
<lsource ls_wasei="y">out course</lsource>
<gloss>outside track</gloss>
</sense>
<sense>
<gloss>outside pitch (baseball)</gloss>
<gloss xml:lang="dut">(1) [honkb.] outside</gloss>
<gloss xml:lang="dut">(2) [golf] out course [= eerst 9 holes van een golfbaan]</gloss>
<gloss xml:lang="dut">(3) [atlet.] buitenbaan</gloss>
<gloss xml:lang="ger">(n) Außenbahn</gloss>
<gloss xml:lang="ger">die vom Schlagmann entfernte Seite. .</gloss>
</sense>
</entry>
```

The steps the parser has to take are:

- Identify the words.
  - Split the sentence into words and (case-indicating) particles.

  - Check if those words are in the dictionary and get their word type, grammatical form and meaning.

- Identify the word order / sentence structure.
  - Scrambling is possible in Japanese (and to a certain extent in Dutch, see section B.2.5), so the parser should be able to recognize different word orders.

- Identify 'phrases' (words that belong together).
  - In Japanese this could be done by splitting on the particles and verbs.
  - Also identify subordinate clauses (and split these in phrases as well, if necessary).

- Identify the structure of the sentence.
  - Add cases to the words/phrases. The case is decided from the particle following the phrase or from the position in the structure. This can be used for the translation.

4.3.2 Translation method

I think a method like the Rosetta (compositional translation) method as described in section 4.1.3, or the synchronous context-free grammar from section 4.2.3 would be most useful. The synchronous context-free grammar is also similar to the compositional translation method in that they use synchronous grammars, but apparently it's impossible for the entire language of Dutch to be written in a context-free way, as structures like cross-serial dependencies ($a^n c^m b^n d^m$) occur [81, 86]. You could solve the problem by setting the maximum for n and m to 4, which is enough for daily language use. However, the grammar becomes more complex when you need to incorporate these restrictions into the rules. Maybe it even becomes so complex, that it's more manageable when you use the unrestricted version. Though parts of the language (for example, one kind of phrase, like an Noun Phrase) can be written as a context-free grammar.

So, because the compositional method does not depend on writing a context-free grammar, this method might be the better choice to look at the problem of this thesis: "When do you have to insert a definite or indefinite article and when do you have to choose a singular or plural form for nouns when translating from Japanese to Dutch in the context of syntax-based machine translation?".

The papers about the synchronous models (section 4.2.3), on combining syntax with statistics, are all quite recent. The compositional translation method from Rosetta originates in the 1980s, but I think it does fit in with the current research (even though the modern syntax-based systems do use more statistics).

Even though I'll be looking at translation from Japanese to Dutch in this thesis, an advantage of the compositional translation method is that it's reversible: you don't have to make two grammars (one for Japanese-to-Dutch and another one for Dutch-to-Japanese).

5 Articles

In this chapter, we first take a quick look at nouns in Dutch (section 5.1) and Japanese (section 5.2). It is important to look at the structure of nouns in both languages, as for both languages a grammar will be described in section 7. Then we take a look at when articles should (section 5.3) or shouldn't (section 5.4) be used in Dutch when translating from Japanese.

5.1 Nouns in Dutch

Officially there are three grammatical genders for nouns in Dutch: male, female and neutral. However, in Dutch spoken in The Netherlands, the male/female distinction has disappeared almost entirely and male has become the standard (not so in the Dutch-speaking part of Belgium) [83]. For both male and female singular nouns, the article is 'de'. For neutral nouns, the singular article is 'het'. About three-quarters of the Dutch nouns are preceded by 'de' [53], so if you do make a guess, guessing 'de' will be correct about 75% of the time. Plural nouns that get an article, always get 'de'.

However, the article a noun takes can be expected to be present in noun's lexical entry in the dictionary. Rules for determining articles can be found in Appendix A.

Oosterhof [74], in his book about generic sentences, mentioned the following cases for when articles are and aren't used:

- In kind predicate sentences, which are sentences in which the predicate can only refer to a kind/species, the definite singular is the most unmarked case (12), not the bare plural (13). For some speakers of Dutch, the bare plural sentence is unacceptable [74].

  (12) 'De dodo is uitgestorven.'
  The dodo is died out.
  'The dodo has gone extinct.'

  (13) 'Dodo's zijn uitgestorven.'
  Dodos are died out.
  'Dodos have died out.'

- As subjects of characterizing sentences (which is a sentence that expresses a generalization, but the subject noun phrase does not refer to a kind), bare plurals (plurals without an article) are highly unmarked (14, 15). The second best choice is the indefinite singular (16). However, if such a sentence contains the word *er* ('there', dummy subject), the indefinite singular can't be generic anymore (as in (17)). In direct object position the definite singular is the best choice. [74]

  (14) 'Chinezen eten met twee stokjes.'
  Chinese eat with two sticks-DIM.
  'Chinese people eat with chopsticks.'

  (15) 'Wielen zijn rond.'
  Weels are round.
  'Weels are circular.'

(16) 'Een Chinees eet met twee stokjes.'
A Chinese eat with two sticks-DIM.
'A Chinese person eats with chopsticks.'

(17) 'Er is een Chinees die met twee stokjes eet.'
There is a Chinese that with two sticks-DIM eat.
'There is a Chinese person who eats with chopsticks.'

- For mass nouns, in subject position and PP-complement[8] (18, 19) the bare mass term is best, for object position the mass term with definite article is best. [74]

(18) 'Hij vertelde een verhaal over de uitvinder van marsepein.'
He told a story about the inventor of marzipan.

(19) 'Hij vertelde een verhaal over de uitvinder van de marsepein.'
He told a story about the inventor of the marzipan.

- Higher taxonomic levels correspond to a lower frequency of definite singulars (and, correspondingly, to a higher frequency of bare plurals) [74]. For example, when a word for a set of animal species 'higher up in the taxonomic order' (like 'mammal') is used, the definite singular ('the') is used less than when a word for a specific animal species is used (like 'horse').

- Definite and indefinite singulars are used more frequently with animal names than with nationality names; definite plurals appear more frequently with nationality names [74].

- When a noun is a proper name, no article is used [74].

- Definite and indefinite singulars appear more frequently with 'mens' ('human/man') than with 'man' ('man') and 'vrouw' ('woman'). The opposite result was found for bare plurals [74].

- Complement positions in postnominal PPs are to some degree similar to the direct object position (20). The complement position is the objective or subjective genitive.

(20) 'Het doden van ongeboren kinderen is in Ierland verboden.'
The killing of unborn children is in Ireland forbidden.
'Killing unborn children is forbidden in Ireland.'

5.2 Nouns in Japanese

Just as in Dutch and English, a Japanese noun phrase can consist of just a noun, but also contain adjectives, adverbs, numbers, classifiers, pronouns, etc. The main difference is that there exist no equivalents for 'the' and 'a(n)'.

[8] A PP-complement in Japanese is indicated by the particle 'ni'[33] or 'de'[24].

Nouns can also have modifiers: preceding (like adjective phrases) or succeeding (only particles, as listed in table 2, and bound morphemes like honorific suffixes, as listed in table 7). Bond [6] lists the following noun-preceding modifiers: special adnominal-modifiers (*rentaishi*) such as *aru* '(a) certain'[9]; finite adjective and verb phrases; other nouns, either directly or with an adnominal modifier; and certain prefixes (such as *shin-* 'new').

In Japanese grammar, sentences are viewed as being composed of sets of phrases called *bunsetsu*. A bunsetsu typically consists of a content expressions and a function word that indicates the role of the content expression in the overall sentence [13]. These function words can be particles indicating case (like '-ga' for subject and '-wo' for object), or words corresponding to prepositions in English (like '-kara' (meaning 'from')), called postpositions by Bond [6].

Ono [73] [10] divides the postpositions into 3 classes, which can be seen in table 2. A phrase can have more than one postposition, but only one can be a case marker, and it's not possible to have 'ga' or 'wo' combined with 'wa' or 'mo'.

Type	Number	Examples
case	(3)	-ga 'nominative', -wo 'accusative', -ni 'dative'
semantic	(8)	-ni 'locative/goal', -e 'locative/goal', -de 'locative/instrumental', -to 'commitative', '-kara 'source', -made 'goal', -yori 'source/comparative', -no 'adnominal'
adverbial	(10)	-wa 'topic', -mo 'emphatic', -nado 'such as', -dake 'only', -made 'even', -bakari 'only', -sae 'even', -demo 'even', -shika 'only', -sura 'even', -chuu 'during'

Table 2: Postpositions divided into 3 classes according to Ono [73], from Bond [6]. I added 'during' from Bond [6], which was not in Ono's original table.

5.3 Using articles

Bond [6] notes that in many cases, reference to known entities is marked by a focus marker ('-wa' (the topic particle) or '-mo' ('also')), where in English a definite article would be used. He also notes that Japanese can mark the subject of characterizing sentences with the grammaticalized phrase *-to-iu* "called", followed by a nominalizer and a case-marker, as in (21). Other nominalizers Bond lists are *no* "thing", *mono* "thing", and *koto* "abstract thing", *tokoro* "place", and *jikan* "time". These nominalizers make a noun phrase of the preceding clause.

(21) *zou-to-iu-mono-ga banana-ga suki-da* [6]
elephant-QUO-called-NOMINALIZER-NOM banana-NOM like
(things called) Elephants like bananas.

[9] Actually *aru* is a verb meaning 'to be' (for non-living things) used as a nominal modifier.
[10] Reproduced in Bond [6].

In section 6.2, Bond [6] notes that semantically, common nouns in Japanese and mass/uncountable nouns in English can be viewed as the name of a kind, as they can be used for kind reference with generic predicates, as in (31). To get a specific/individual reading, a classifier has to be added (32). In English, nouns have an individual reading by default. So, the standard reading would be the 'kind' reading, where the NP would be definite singular. In the next chapter (chapter 6) I will go into more detail about number (singular and plural forms of nouns).

However, if you also look at context, the first time an NP is encountered it is indefinite. The following times the 'definite singular' could apply.

Context: When an NP has been referenced to at least once in preceding sentences (by use of a name, an indefinite NP, or another noun), its value should be definite, with number the same as the preceding NP.
Each time a new NP is encountered, set the value for definiteness to 'indefinite', except if there is a rule stating otherwise. Remember its value for number. The next time this same NP is encountered, the value for 'definiteness' becomes 'definite' (or otherwise if there's an applicable rule). Something like:

```
if (NP not in NPlist) then def=indef, num=(use rule for number)
else def=def (OR use rule), num=(num from NPlist)
```

where NPlist = list/database or such containing NPs that have been encountered in this text (and their referents, as 'Jan' and 'the boy' in example 22 point to the same entity).

(22) *Jan* kwam binnen. *De jongen* ging zitten.
Jan entered. *The boy* sat down.

However, for this you need discourse rules and the Rosetta system doesn't look at context, so I will leave this aside for the moment as it's outside of the current scope of this thesis, and all existing machine translation systems don't look at context either. However, it is desirable to have a system that looks past the sentence boundaries.

5.4 Using no articles

Bond [6] mentions the noun suffix '-muke' that requires its head to be generic (referring to a kind [74]), as in (23). So when a noun phrase contains the suffix '-muke', the noun gets translated as an indefinite plural. In example (24), there is a difference between the English and Dutch translations: with the suffix '-toshite', the Dutch translation does *not* get an article when it's predicatively used, which is the case here. The noun does get an article when it's an identificational sentence like (25), which is a sentence in which two referential noun phrases are equated [67].

(23) *hataraku josei muke-no zasshi* [6]
working woman aimed-at-ADN magazine
a magazine aimed at working women
Dutch: een tijdschrift gericht op werkende vrouwen

(24) *watashi-wa tsuuyaku-toshite hataraku* [6]
I-TOP interpreter-as work
I work as an interpreter
Dutch: Ik werk als tolk.

(25) *watashi-wa Utrecht-no shichou desu*
I-TOP Utrecht-POSS mayor be
I am the mayor of Utrecht.
Dutch: Ik ben de burgemeester van Utrecht.

In the next chapter (chapter 6), we look more closely at number: singularity and plurality of nouns. This topic has been touched on a bit in this chapter as well, but more details can be found in the next one.

6 Number

This section contains descriptions and rules for indicating singularity and plurality for nouns in both Dutch and Japanese.

6.1 Forms indicating singularity/plurality in Japanese

Nouns in Japanese don't have distinct singular and plural forms. There are several suffixes which indicate plural, but all of them are optional.

Bond [6] lists the collectivizing suffixes in table 3:

Japanese	Gloss	Politeness
-kata/-gata	and others	Very polite
-tachi	and others	Polite
-ra	and others	Neutral
-domo	and others	Humble

Table 3: Collectivizing suffixes in Japanese, according to Bond [6].

The suffixes '-tachi', '-domo', '-gata' and '-ra' are plural suffixes used only if the antecedent is animate (with humans or personified animals [6, 7, 1]). Bond [6] also notes that 'collectivizing suffixes can be used only with noun phrases with locatable individuated referents (corresponding to English countable definite noun phrases). However, '-tachi' doesn't always indicate plural. Martin ([57] in [95]) mentions that 'Yamada-sensei-tachi' doesn't mean multiple entities of Yamada-sensei ('Yamada' is a name, 'sensei' a suffix meaning 'teacher'), but 'a group of people Yamada-sensei is a member of'. So '-tachi' is more a 'collectivizer' than a 'plural'.

Another plural suffix is '-ra', which can be used when the noun consists of more than one individual entity [6]. An example of this is 'korera' (these), which consists of 'kore' (this) + 'ra' (plural).

6.2 Numeral classifiers

A plural in Japanese can be indicated by using a numeral classifier, as in example (26-27) from [9]. As numerals cannot directly modify nouns, a classifier needs to be inserted ('-tsu', '-hako', etc), which depends on the kind of noun that is modified (see [99] for a list), as well as the genitive particle 'no'. This is similar to the case of 'pluralizing' uncountable nouns like 'fruit', which becomes *twee stuks fruit* ('two pieces of fruit') in Dutch. In this case, no article is necessary. According to Koiso ([48] in [6]), there exist more than three hundred numeral classifiers. Downing [21] estimates the amount between 200 and 300. However, individual speakers use between 30 to 80, and even less in actual daily use ([21]). In Downing's collection of 500 samples (textual and spoken), 24% of the classifiers was the uninformative general classifier '-tsu'.

(26) 2-tsu-no koppu
2-piece-GEN cup
Dutch: “2 koppen”
English: “2 cups”

(27) 2-hako-no pen
2-box-GEN pen
Dutch: “2 dozen pennen”
English: “2 boxes of pens”)

The structure of the noun phrase in (27) is of the form *XC-no-N*, where C is a group classifier, which will be translated into English as *X' C' of N'*, where N will be plural if it is headed by a fully or strongly countable noun or a plural only [6]. In Dutch, it would be *X' C' N'* with N plural. Bond [6] also translates (28) into the same construction in English (*a C' of N'*), even though the Japanese construction is *N-no-C*. However, when you have ‘N1-no-N2’ it is possible to translate it both as *N2' of N1'* and *N1"s N2'* in English.

(28) pen-no hako
pen-GEN box
Dutch: “een doos pennen”
English: “a box of pens”

Syntactically, numeral classifiers (‘C’) are a subclass of nouns (‘N’; ‘N2’ in the example above). The main difference is that classifiers cannot stand alone, but postfix to numerals or combine with quantifiers like *suu* (‘some’) or the interrogative *nan(i)* (‘what’), as can be seen in (29) [6]. In combination with a numeral, classifiers can form their own noun phrase, as in example (30) from [6].

(29) ni-hiki ‘two animals’ (numeral)
suu-hiki ‘some animals’ (quantifier)
nan-biki ‘how many animals’ (interrogative)

(30) [context in which some letters are salient]
ni-tsuu-wo yonda (‘tsuu’ is a counting word specifically for letters [99])
two-CL-ACC read
I read two (of the) letters

Bond [6] notes that semantically, common nouns in Japanese and mass/uncountable nouns in English can be viewed as the name of a kind, as they can be used for kind reference with generic predicates, as in (31). To get a specific/individual reading, a classifier has to be added (32). In English, nouns have an individual reading by default.

(31) watashi-wa keeki-ga suki-da
I-TOP cake-NOM like
I like cake.

(32) watashi-wa hito-tsu-no keeki-wo tabeta
I-TOP one-CL-ADN cake-ACC eat-PAST
I ate one cake

However, also in Japanese there are fully countable nouns which can be directly modified by numerals: the nouns that refer to discrete enumerable individuals, like all numeral classifiers, words that denote times, wins/losses/draws etc. [6].

In table 4, taken from Bond [6], the major numeral-classifier patterns are listed. It is important to know how such measure phrases can be scrambled in Japanese, as it is necessary to know to which noun a numeral-classifier construction refers.

Type:	Form:	Example:	Translation:	Comment:
pre-nominal	Q-no T-m	ni-hiki-no inu-ga	'two dogs'	indefinite, introduces important referents
appositive	TQ-m	inu ni-hiki-ga	'two dogs'	indefinite
floating	T-m Q	inu-ga ni-hiki	'two dogs'	indefinite, introduce new number information about a known referent,
	Q T-m	ni-hiki [...] inu-ga	'two dogs'	used when the nominal has other modifiers
partitive	T-no Q-m	inu no ni-hiki-ga	'two of the dogs'	the quantified N is definite, quantifier restricts a subset of a known amount
anaphoric/deictic	T-m	ni-hiki-ga	'the two dogs'	definite

Table 4: Patterns of numeral-quantifier constructions, from Bond [6]. T is a the quantified noun phrase, Q a numeral-classifier and m is a case-marker.

An interesting observation from Gunji and Hasida [31] is that "prenominal measure phrases (for example *san-nin-no gakusei*[11]) are normally used to refer to a definite group of people or things of a definite mass". When a counting word follows the noun (called 'postnominal measure phrases'), for example *gakusei sannin* ('students 3-people', '3 students'), the phrase is (usually) introduced to the discourse for the first time. For example, *san-nin-no gakusei* ('3-people-POSS students', '3 students') may refer to a set of three students that has already been introduced in the discourse, while 'gakusei san-nin' usually introduces a new set of three students into the discourse [31]. So when the three students have already been introduced in the discourse, the definite article should be used.

In table 4 there are several phrases in Japanese that are all translated into English as 'two dogs'. It is possible to scramble words/phrases and still get the same translation, but there are restrictions: if the counting word+classifier is associated with the subject (*ga*), the object phrase (*wo*) cannot be inbetween. If the counting phrase is associated with the object, it's no problem for the subject to be inbetween [31, 92].

[11] *san-nin-no gakusei* is glossed as '3-people-POSS students' and means '3 students'.

That the object cannot appear inside a subject phrase, is a result of the way scrambling works in Japanese. Scrambling is possible with entire phrases (moving the entire subject or object phrase, for example), or within phrases (changing the order of the words inside the VP phrase, for example, though it is possible that the meaning changes with clause-internal scrambling) [97]. It is also not possible to scramble a topic phrase into the middle of an object phrase, as can be seen in example 33 from [97].

In (34) below, there is an object-marker *-wo* inbetween the noun and the measure phrase, but in English the measure phrase would be included in the object. The 'three men' are newly introduced. It is not possible that 'sannin' (*three people*) refers to 'onna' (*woman*), as a measure phrase cannot immediately precede a noun: there should be a connecting particle 'no' inbetween: 'sannin no onna' (see also the first case in table 4).

(33) a. Watashi-wa akai hon-o mita.
I-TOPIC red book-ACC saw
'I saw the red book'
b. *Akai watashi-wa hon-o mita
red I-TOPIC book-ACC saw
'The red I saw the book'[12]

(34) Otoko-wo sannin onna-ga mitsuketa.
man-OBJ three woman-NOM find-PAST
"There were three men who the woman found."

Bond [8] notes that when you want to count nouns that don't have distinct singular/plural forms, you need a classifier. This is the same in Dutch: you can use *een stuk...* ('a piece of...') for Dutch nouns like *rommel* ('junk/rubbish'), *honger* ('hunger'), *vee* ('cattle'), *hersenen* ('brain/brains'), *notulen* ('minutes/notes'), etc., as in: *drie hoopjes rommel* ('three little piles of rubbish'), *drie stuks vee* ('three head of cattle'), *twee stel hersenen* ('two brains'), *veel gevallen van honger* ('many cases of hunger'), etc.

There are also nouns, like 'cake', that can be used in both countable and uncountable (mass) noun phrases [8]. This is the same in Dutch: *Ik hou van cake* ('I like cake') and *Ik wil een cake* ('I want a cake').

Bond [8] divides the noun phrases into three categories:

- Generic noun phrases refer to a kind/species. They can be expressed with 'the', 'a' and nothing, but only no article is acceptable in all contexts, so Bond e.a.'s system generates generic noun phrases as bare noun phrases. The number is determined by the countability preference (which is in their dictionary). '*Mammoths* are extinct'.

- Referential noun phrases refer to some specific referent. Number and countability are ideally determined by the properties of the referent.
'*Two dogs* chase *a cat*'.

- Ascriptive noun phrases are used to ascribe a property to something, for example 'a mammoth' in 'That animal is a mammoth' [10]. Normally they have the same

[12]The translation for sentence b was added by me.

number and countability as the noun phrase whose property they are describing. In this example, Bond [8] chooses the article 'an', but doesn't explain why. I think this is because it's a predicative sentence, which takes an indefinite predicate (if you'd take a definite predicate, you'd have an identificational sentence).
'Hathi is *an elephant*'.

Bond [8, 6] used a number of conditions to determine the category, with 'referential' as the default. If it says '...according to...' in the list below, it means that the necessary information is in the dictionary they used. Information about countability of nouns is also in their dictionary (the ALT-J/E dictionary) [5].

1. if the Japanese is explicitly plural then countable and plural
   tachi → plural and countable

2. some lexical items are plural by default (in English)
   men "noodles"

3. determine according to determiner
   one dog, all dogs

4. determine according to classifier
   hito-kire-no-keeki a slice of cake
   hito-yama-no-keeki a pile of cakes

5. determine according to quantifier
   ono'ono-no keeki each cake
   ryouhou-no keeki both cakes

6. determine according to complement (= the phrase that assigns a property to the head noun)
   zenkoku no gakkou "schools/*a school all over the country"

7. ascriptive NPs match their subjects
   A computer is a piece of equipment

8. match with antecedent[13]
   Two men, strangers, came in

9. determine according to verb
   I gather flowers/*a flower
   wadai-ga tsukita I ran out of topics/*a topic

10. use default value

    (a) uncountable, weakly countable[14] become:
        uncountable and singular

[13]Bond uses the word 'antecedent' here, even though it is an apposition. As an antecedent is the noun phrase earlier in the sentence to which is referred, I think he calls it thus because 'strangers' refers back to 'two men'.

[14]Weakly countable: uncountable nouns that are readily convertible to countable, such as 'beer'.[5]

(b) pluralia tanta become:
countable and plural

(c) countable and strongly countable[15] become:
countable and singular or plural according to the dictionary default

Instead of numerals, measure words like *subete* ('all'), *sorezore/onoono* ('each'), *hotondo* ('most'), *hanbun* ('half'), *takusan* ('much/many'), *kazukazu* ('many'), *sukoshi* ('a little/few'), *samazama* ('various'), etc. also have to be connected to the noun by *no* [31, 7]. Both *takusan* and *kazukazu* translate to 'many', but in the case of *kazukazu* the word itself implies that the noun phrase it modifies is made up of discrete entities, and thus should be translated as plural [7].

An example can be seen in (35) from [31] and in (36) from [98].

(35) subete-no gakusei
all-GEN students
Dutch: "alle studenten" OR "iedere student"
English: "all (the) students" OR "every student"

(36) onoono no fairu ni-tsuite samazama no jouhou ga hyouji-sareru.
each-POSS file about various-POSS information-SUBJ is-displayed
Various information is displayed about each file (It displays various information about each file).

6.3 Set nouns

Jan Rijkhoff [84, 85] grouped nouns into four types, which can be seen in both table 5 and 6. The word 'space' in the tables' upper left corner, means that these features are based on spatial aspects of the nouns. The grouping is based on the semantic features 'shape' and 'structure/homogeneity'.

- The feature **shape** (in both tables) means that the entity referred to by the noun has a well-defined (physical) outline.
- When the feature **structure** is positive, the noun is divisible and the property indicated by the noun holds for all parts of the space the entity referred to by the noun occupies.
- The feature **homogeneity** specifies for number. When it is positive, it means that the entity of the noun is strictly singular in number.

I do think that both 'structure' and 'homogeneity' refer to the same aspect, looking at the types of nouns in the table, but his second table is more extensive as he added the type 'general nouns' here.

[15] Strongly countable: countable nouns that can be converted to uncountable, such as 'cake'.[5]

SPACE	-STRUCTURE	+STRUCTURE
-SHAPE	conceptual	mass
+SHAPE	set	
	individual	collective

Table 5: Different types of nouns from Rijkhoff's 1992 distinction [84].

SPACE	-HOMOGENEITY	+HOMOGENEITY
-SHAPE	general	
	sort	mass
+SHAPE	set	
	singular object	collective

Table 6: Different types of nouns from Rijkhoff's 2002 distinction [85].

The types in these tables that have been renamed, are described in a similar way by Rijkhoff:

- **Conceptual nouns** and **sort nouns**:
  The referent of a conceptual/sort noun is not divisible and also doesn't have a definite physical outline. The real-world referent is characterized by the property of the noun. Rijkhoff [84] gives the example of the pseudo-English noun 'grapiness', which can mean "one or more individual grapes, a mass characterized by 'grapiness' (e.g. juice of grapes), or one or more collectives (bunches) of grapes".

  Rijkhoff [84] does write that concept nouns are rather common in Southeast Asian languages, where they are also used for referents that are individuals, so that the best test for deciding whether a noun is a concept noun is to check whether or not it can occur in a direct construction with a cardinal numeral. Conceptual nouns do need classifiers when combined with numerals: numeral classifiers, in Japanese for example '-tsu' and '-nin' (see also section 6.2 above).

- **Individual nouns** and **singular object nouns**:
  The referents of these nouns do have shape, but are not divisible. Rijkhoff [84] notes that many European languages have a large number of individual nouns, for example count nouns like 'car', which aren't divisible in the way that when you divide it, you get two cars (see the example below at the 'mass noun' item). The phrasing 'singular object noun' does make this meaning more clear.

The common noun types in both tables are mass nouns, collective nouns and set nouns.

- **Mass nouns** are nouns like 'water' and 'furniture'. The referents of these nouns do have structure (they are divisible), but they do not have a definite shape (physical outline). For example, the mass noun 'water' is divisable: when the space for which this property 'water' holds, is divided, all parts of that space are still 'water'. The

count noun ‘car’, for example, is not divisible: when the space which ‘car’ occupies is divided, you don’t get two cars.

- **Collective nouns** define a ‘collection’ of individual entities which all share the same property and have a definite (physical) outline in space. If this collection is divided, the property still holds for each part. Examples would be nouns like ‘family’ or ‘bunch’ (a bunch of grapes, a bunch of flowers).

- **Set nouns** have a physical outline (+SHAPE) and are divisible, but they are not specified for number: they can refer to one entity or more than one entity.
  They do not need classifiers when combined with numerals. The difference with individual/singular object/collective nouns is that with collective nouns, the numeral acts as a multiplier, which is reflected in some form of plural marking: ‘one car’, ‘two car-s’. When a numeral is followed directly by a set noun (so, without the use of a classifier), the numeral refers to the number of individuals within the set. Rijkhoff [84] again gives a pseudo-English example of ‘two car’ or ‘a car-set of two’, if ‘car’ were a set noun. Plural nouns in Dutch are always set nouns. Dutch examples can be seen in (37).

The ‘set’ category is also relevant to the number problem for Japanese nouns: is the noun a group (set) consisting of separate entities (+SET) or not (-SET). This is a semantic feature, but since Rosetta has no explicit semantic component, we treat it here as a syntactic feature. The reason that Rosetta does not have this distinction, was the idea that similarity in meaning or translation equivalence would be enough for translation.

This distinction is visible in Japanese as well, as it’s not possible to use counter words (‘hito-*tsu*’) with -SET nouns (see the example in (38)). The word ‘mizu’ also doesn’t occur with any other counter words/classifiers, though there is a counter word for ‘drops of liquid’ (滴, teki), as can be seen in 39.

(37) **Mass noun, -SET**: hout (‘wood’), water (‘water’)
Mass noun, +SET: vee (‘cattle’), meubilair (‘furniture’), politie (‘police’)
Count noun, -SET: man (‘man’), hond (‘dog’), auto (‘car’), postzegel (‘stamp’)
Count noun, +SET: groep (‘group’), kudde (‘herd, flock’), horde (‘horde’)

(38) *一つの水
*hitotsu no mizu
*one water$_{massnoun,-SET}$

(39) 一滴の水
itteki no mizu
one drop of water

There is also a new category in the second table, presumably added for symmetry with the set noun type:

- **General nouns** are nouns that don't distuinguish between classifiers used with singular object nouns (sortal classifiers, like 'many' in English) and classifiers used with mass nouns to indicate size/volume/weight (mensural classifiers, like 'much' in English).

6.4 Generic NPs

In English (and Dutch), a bare plural (plural noun without an article) or a singular NP with an article can denote a generic NP. In Japanese, this is not possible, as all NPs are 'bare' (without articles) and don't have separate singular/plural forms. However, no NP is uniquely identified in the (Dutch/English) syntax itself as a generic NP [89]. In Japanese, the sentences in (40) and (41) are generic, because the topic marker 'wa' is used [89]. In (42), the subject particle 'ga' is used, so that sentence/noun phrase can be interpreted as a generic or specific interpretation.

(40) Itariazin-wa yooki-da
Italian-TOP cheerful-COP
'Italians are cheerful'

(41) Tori-wa tobun-da
bird-TOP fly-COP
'A bird flies'

(42) Tori-ga tonde-iru
bird-TOP flying
'A/The bird is flying'

When a plural marking like '-tachi' is used, the noun phrase is specific, not generic [89].

So, if encountering a noun phrase followed by the particle 'wa', in Dutch you would get a bare plural or definite singular (which are the most unmarked cases, as described by Oosterhof in section 5 with examples in (12)-(13)). When 'ga' is used, all article/number options are still possible.

Generic terms behave syntactically like proper nouns, as they don't interact with a modal operators (a word like 'usually', 'possibly', etc), quantifiers ('all', 'many', etc) or negatives [90]. An example can be seen in 43 from [90].

(43) Every woman likes dogs.
Every woman likes Peter.

7 Translation theory

In this section I'll describe the theory on how to choose the correct article and number for nouns when translating from Japanese to Dutch. To indicate definiteness and number for nouns, I use features (detailed in section 7.1). Next, I describe the translation mechanism for proper nouns and normal noun phrases with rules based on the Rosetta system.

7.1 Features

Features are properties containing extra information about the word, which is necessary for the grammar of the language or for translation.

Each node in a derivation (D-)tree, regardless of their category, has a feature 'bonus'. The feature 'bonus' is a property of the nodes in the D-tree. The default value of this feature is the integer value 0, but when a rule is applied that gives a bonus, this value is increased. Not all rules give a bonus. The bonus can be raised or lowered in the rules, but only positive bonus points are used here to prevent a net bonus of 0. The bonuses of the child nodes are summed with the bonus of the parent node, so at the top of the tree all bonuses are summed up, which becomes the score for this result tree. The bonus values are used for sorting the possible translations, as each translation is a tree with a certain bonus score.

For example, you have the following tree and you can apply various rules.

Applying a rule with a bonus of 1 in A and also a rule with a bonus of 1 in C, results in a total bonus value of 2 in S. Another rule which can also be applied in A, does not give a bonus. Then the total bonus value of S is 1, as the rule with bonus in C has been applied. So the first set of applied rules gives a higher bonus value, and as a result that sentence is ordered higher than the second sentence.

In a compositional translation system, it tries to get all possible translations. A bonus system is used to create an ordering, for when you only want one result (see also at the end of section 4.1.3 for details about the bonus system in the Rosetta compositional translation framework).

7.1.1 N and NP

Features an N and NP have in both Dutch and Japanese:

- **sex**: this is the natural gender (values: male, female, undefined). This feature will only be filled if it is known whether or not a proper noun is used for males or females, otherwise its value is 'undefined' (also the case for objects without a natural gender, like a table). If the noun [16] in the noun phrase has a sex indicated, this value of 'sex' will become the sex value of the NP.
- **divisible**: True or False

[16] This can be a proper noun, or a normal noun which is only used for a specific natural gender, like an occupation (for example, Dutch 'secretaresse' means 'female secretary'). In the first part of section 5 there are examples of nouns that indicate a certain gender.

Some verbs require their object or subject to have the +SET feature and/or be of the mass noun type (so, 'mass ±SET' or 'count +SET', or even shorter: 'mass noun' or '+SET noun'). As these nouns are divisible, this is the name of the feature. Examples are V = gather (gather flowers), run out of (run out of topics). Dutch examples: verzamelen (object); scheiden, uit elkaar gaan (subject). More about set nouns can be read in section 6.3.

- **generic**: True or False
- **animate**: True or False (indicates whether or not the noun is animate; for example, a dog is animate, but a road isn't). If the NP has been created from a PN, this feature is always True.
- **person**: 1st/2nd/3rd. A normal noun is always 3rd person, but pronouns are nouns as well and those do differ between 1st/2nd/3rd person. With a pronoun, the value 'definite' should be set to definite (rule 7.2.2).

Extra features for Dutch Ns and NPs:

- **case**: nom/gen/dat/acc
- **number**: singular/plural
- **gender**: this is the grammatical gender; utrum[17] (male/female, 'de') or neutrum ('het'). Utrum singular nouns get the article 'de' in Dutch, while neutrum singular nouns get the article 'het'. When plural, all nouns get the article 'de'.
- **definite**: def/indef. These are the values 'definite' ('de' (singular and plural), 'het' (singular), or 'empty' (in case of proper nouns)) or 'indefinite' ('een' (singular) or 'empty' (plural)).
  - If the article is 'de' or 'het', the value is definite. The articles in Dutch are "de" or "het" for singular, depending on the grammatical gender (which should be indicated in the Dutch dictionary), and "de" for plural. As the distinction between male and female grammatical gender has disappeared in Standard Dutch [106], it is only important to know whether or not a (proper) noun is male/female or neutral, if it needs an article.
  - If the article is 'een', then the value is indefinite. The indefinite article "een" is used for singular and no article for plural. The indefinite article is used with proper nouns if it's not clear which one is meant: "Een Karin uit mijn klas" ('A Karin from my class', meaning that there are multiple girls called Karin in the class and the speaker does not know which Karin).
- **countmass**: count noun or mass noun
- **canBeBare**: True or False (indicates whether a noun can occur as a bare noun). For example, nouns indicating an occupation (*without* adjectives, so just the bare N) can appear as bare nouns. An example can be seen in (44) and (45) from

[17] Also called 'common gender' [84].

Rijkhoff [84], who calls these nouns 'predicate nouns'. While 'een soldaat' in (45) is an individual/singular object noun (see tables 5 and 6 in section 6.3), 'soldaat' in (44) "seems to have been deprived of its nominal (i.e. individual) aspect, i.e. 'soldaat' is "aspectless" here."

(44) Jan is soldaat.
Jan is soldier.

(45) Jan is een soldaat.
Jan is a soldier.

7.1.2 PrN: Pronouns

Indefinite pronouns are words like 'alles' (everything), 'iemand' (someone), 'iedereen' (everyone).
Definite pronouns are words like 'deze' (this one), 'degene' (the person), 'die' (that one), 'dezelfde' (the same one).

Features an pronoun has in both Dutch and Japanese:

- **sex**: this is the natural gender (values: male, female, undefined). This feature will only be filled if it is known whether or not a proper noun is used for males or females, otherwise its value is 'undefined' (also the case for objects without a natural gender, like a table).
- **animate**: True or False (indicates whether or not the noun is animate; for example, a dog is animate, but a road isn't). If the NP has been created from a PN, this feature is always True. Animate pronouns are for example 'iemand' (someone), 'iedereen' (everyone), and 'degene' (the person), while inanimate pronouns are for example 'alles' (everything), 'deze' (this one), and 'dezelfde' (the same one).

Extra features for Dutch pronouns:

- **person**: 1st/2nd/3rd. A normal noun is always 3rd person, as are these pronouns. Also, the value 'definite' should be set to definite (rule 7.2.3).
- **case**: nom/gen/dat/acc
- **number**: singular/plural
- **definite**: def/indef. These are the values 'definite' or 'indefinite'.

7.1.3 PersPrN: Personal Pronouns

Features a personal pronoun has in both Dutch and Japanese:

- **sex**: this is the natural gender (values: male, female, undefined). This feature will only be filled if it is known whether or not a pronoun is used for males or females, otherwise its value is 'undefined' (also the case for objects without a natural gender, like a table).

- **animate**: True or False (indicates whether or not the noun is animate; for example, a dog is animate, but a road isn't). For personal pronouns, this is always True.

Extra features for Dutch personal pronouns:

- **person**: 1st/2nd/3rd. A normal noun is always 3rd person, but personal pronouns are nouns as well and those do differ between 1st/2nd/3rd person. With a personal pronoun, the value 'definite' should be set to definite (rule 7.2.2).
- **case**: nom/gen/dat/acc
- **number**: singular/plural
- **gender**: this is the grammatical gender; utrum[18] (male/female, 'de') or neutrum ('het'). Utrum singular nouns get the article 'de' in Dutch, while neutrum singular nouns get the article 'het'. When plural, all nouns get the article 'de'.

7.1.4 Proper Nouns

For proper nouns, it's a bit easier to choose number and article. By far in most cases, proper nouns don't get an article in Dutch. There is one class of proper nouns which does get an article, but these proper nouns are always preceded by the definite article (which should be specified in the dictionary): *de Verenigde Staten* ('the United States'), *de Europese Unie* ('the European Union'), *de Seychellen* ('the Seychelles', called just 'Seychelles' in English but requiring an article in Dutch). You could say these are common nouns with a proper noun meaning.

As proper nouns behave as definite nouns in the grammar, whether they get an article or not, the feature 'definite' will get have the value 'definite' for proper nouns. However, the article for proper nouns is empty, except when it is specified in the dictionary (as in the examples above).

Most proper nouns are also singular, but when they're +SET, this is indicated in Japanese by collectivizing suffixes (also described in section 6.1).

Features a PN has in both Dutch and Japanese:

- **number**: singular/plural (in Japanese it's default value is singular, plural when a plural affix is attached)
- **sex**: natural gender; male/female/undefined
- **hon**: True or False (whether or not the PN has an honorific affix attached. The Dutch equivalents can be seen in table 7.)
- **divisible**: True or False
- **animate**: True or False (indicates whether or not the noun is animate; for example, a dog is animate, but a road isn't). If the NP has been created from a PN, this feature is always True.

[18] Also called 'common gender' [84].

7.1.5 Affixes

Features an affix (aff) has in Japanese:

- **sex**: natural gender; male/female/undefined
- **type**: honorific (-san, -sama etc), group (kata/gata/tachi/ra/domo).

In Dutch, an affix is actually a noun which has special features. As well as all the N features, it also has a feature 'type':

- **type**: honorific (meneer, mevrouw etc).

7.1.6 V and VP

Features a V and VP has in both Dutch and Japanese:

- **tense**: past/present, infinitive
- **objectDivisible**: True/False (True if the object of V must be +SET or a mass noun, False if the object of V must be -SET or not a mass noun).
- **subjectDivisible**: True/False (True if the subject of V must be +SET or a mass noun, False if the subject of V must be -SET or not a mass noun).
- **indirectobjectDivisible**: True/False (True if the indirect object (dative) of V must be +SET or a mass noun, False if the indirect object of V must be -SET or not a mass noun).
- **objectPrefGeneric**: True/False (True if V prefers its object to be generic).
- **subjectPrefGeneric**: True/False (True if V prefers its subject to be generic).
- **indirectobjectPrefGeneric**: True/False (True if V prefers its indirect object to be generic).

Extra features specific for Dutch Vs and VPs:

- **number**: singular/plural
- **person**: 1st (I)/2nd (you)/3rd (he)/1st (we)/2nd (you)/3rd (they)

7.1.7 Numerals

Features a Numeral has in Dutch:

- **requires_number** (also indicated by $_{num}$ in the rules in section 7.2): singular/plural (which number does the noun following this numeral need to have? Dutch examples are: 1 boek(*en); 2 boek*(en); anderhalf boek(*en); 1,5 boek*(en)

7.1.8 Quantifiers

Quantifiers are words like 'many', 'each', 'various', etc. A table listing a number of quantifiers with their translations can be found in section 7.2.20, which contains the quantifier rules. Japanese quantifiers are actually nouns, which act as a quantifier when they are followed by the genitive particle 'no'.

Features a Quantifier has in Dutch:

- **requiresNumber**: singular/plural/both. Whether the noun needs to be singular/plural/both (singular if 'elk(e)' (each), otherwise plural).
- **requiresCountmass**: count/mass. Whether the noun needs to be a count noun or a mass noun.
- **definite**: def/indef. Whether the quantifier needs to be definite or not ('de meeste' or 'veel').

7.1.9 P

The P is a preposition or a postposition. In Japanese, these are indicated by particles and in Dutch these are translated by prepositional or postpositional phrases. Examples are the directional suffixes like 'from', of which there are some examples in section 7.2.17.

Features a P has in Dutch:

- **type**: pre/post (preposition or postposition)

7.2 Mini grammar

The rules necessary for the translation system would be as below. They are written as follows for each language (Dutch and Japanese):

- Input: Which objects does the rule apply to?
- Output: What is the result of the rule?
- C: Conditions that apply to the input. If there is more than one condition that applies, the conditions are connected by the logical &.
- A: Actions that are applied by the rule to create the output. If there is more than one action applied, the actions are listed using the & character.

It is possible to have multiple C/A pairs, as well as nested pairs, for example:

- C1
  - C2
    A2

  - C3
    A3

- A1

- C4
  A4

In such a case, *all* conditions need to be checked and applied to the input structure.

7.2.1 RBasicNP$_{generic=x}$: Basic NP rules

The parameter x in the rule name is used when the value of NP.generic can be both True or False.

- *Japanese, RBasicNPJP$_{generic=x}$:*
  Input: N'(α)
  Output: NP(α)
  C:
  N' contains an N, possibly with Adjective, relative clause etc.
  NP.generic = False
  A:
  NP.generic = x

- *Dutch, RBasicNPNL$_{generic=x}$:*
  Input: N'(α)
  Output: NP(α)
  C:
  N' contains an N, possibly with Adjective, relative clause etc. It already has a value for the feature 'number' and 'countmass'.
  A:
  NP.countmass = N'.countmass &
  NP.number = N'.number &
  NP.generic = x

- *Dutch, RindefsingcountNL$_{generic=x}$:*
  Input: N'(α)
  Output: NP(ART(een) N'(α))
  C:
  N'.num = sing &
  N'.def = indef $\vee$ undefined
  A:
  NP.num = N'.num &
  NP.def = indef &
  NP.countmass = N'.countmass &
  NP.divisible = N'.divisible &

NP.gen = N'.gen &
NP.sex = N'.sex &
NP.animate = N'.animate &
NP.generic = x

- *Dutch, Rindefplurcount$NL_{generic=x}$:*
  Input: N'(α)
  Output: NP(N'(α))
  C:
  N'.num = plur &
  N'.def = indef $\vee$ undefined
  N'.countmass = count &
  N'.divisible = True
  A:
  NP.num = N'.num &
  NP.def = indef &
  NP.countmass = N'.countmass &
  NP.divisible = N'.divisible &
  NP.gen = N'.gen &
  NP.sex = N'.sex &
  NP.animate = N'.animate &
  NP.generic = x

- *Dutch, Rdefnsingcount$NL_{generic=x}$:*
  Input: N'(α)
  Output: NP(ART(het) N'(α))
  C:
  N'.num = sing &
  N'.def = def $\vee$ undefined
  N'.gender = neutrum &
  N'.countmass = count &
  N'.divisible = False
  A:
  NP.num = N'.num &
  NP.def = def &
  NP.countmass = N'.countmass &
  NP.divisible = N'.divisible &
  NP.gen = N'.gen &
  NP.sex = N'.sex &
  NP.animate = N'.animate &
  NP.generic = x

- *Dutch, Rdefusingcount$NL_{generic=x}$:*
  Input: N'(α)

Output: NP(ART(de) N'(α))
C:
N'.num = sing &
N'.def = def $\vee$ undefined
N'.gender = utrum &
N'.countmass = count &
N'.divisible = False
A:
NP.num = N'.num &
NP.def = def &
NP.countmass = N'.countmass &
NP.divisible = N'.divisible &
NP.gen = N'.gen &
NP.sex = N'.sex &
NP.animate = N'.animate &
NP.generic = x

- *Dutch, $RdefplurcountNL_{generic=x}$:*
  Input: N'(α)
  Output: NP(ART(de) N'(α))
  C:
  N'.num = plur &
  N'.def = def $\vee$ undefined
  N'.countmass = count &
  N'.divisible = True
  A:
  NP.num = N'.num &
  NP.def = def &
  NP.countmass = N'.countmass &
  NP.divisible = N'.divisible &
  NP.gen = N'.gen &
  NP.sex = N'.sex &
  NP.animate = N'.animate &
  NP.generic = x

- *Dutch, $RindefmassNL_{generic=x}$:*
  Input: N'(α)
  Output: NP(N'(α))
  C:
  N'.num = sing &
  N'.def = indef $\vee$ undefined
  N'.countmass = mass &
  N'.divisible = True
  A:
  NP.num = N'.num &

NP.def = indef &
NP.countmass = N'.countmass &
NP.divisible = N'.divisible &
NP.gen = N'.gen &
NP.sex = N'.sex &
NP.animate = N'.animate &
NP.generic = x

- *Dutch, RdefnmassNL$_{generic=x}$:*
  Input: N'(α)
  Output: NP(ART(het) N'(α))
  C:
  N'.num = sing &
  N'.def = def $\vee$ undefined
  N'.gender = neutrum &
  N'.countmass = mass &
  N'.divisible = True
  A:
  NP.num = N'.num &
  NP.def = def &
  NP.countmass = N'.countmass &
  NP.divisible = N'.divisible &
  NP.gen = N'.gen &
  NP.sex = N'.sex &
  NP.animate = N'.animate &
  NP.generic = x

- *Dutch, RdefumassNL$_{generic=x}$:*
  Input: N'(α)
  Output: NP(ART(de) N'(α))
  C:
  N'.num = sing &
  N'.def = def $\vee$ undefined
  N'.gender = utrum &
  N'.countmass = mass &
  N'.divisible = True
  A:
  NP.num = N'.num &
  NP.def = def &
  NP.countmass = N'.countmass &
  NP.divisible = N'.divisible &
  NP.gen = N'.gen &
  NP.sex = N'.sex &
  NP.animate = N'.animate &
  NP.generic = x

Japanese syntactic rule	Dutch syntactic rule	meaning rule
RBasicNPJP$_{generic=T/F}$	RBasicNPNL$_{generic=T/F}$	LBasicNP$_{generic=T/F}$
RBasicNPJP$_{generic=T/F}$	RindefsingcountNL$_{generic=T/F}$	LindefsingcountNP$_{generic=T/F}$
RBasicNPJP$_{generic=T/F}$	RindefplurcountNL$_{generic=T/F}$	LindefplurcountNP$_{generic=T/F}$
RBasicNPJP$_{generic=T/F}$	RdefnsingcountNL$_{generic=T/F}$	LdefnsingcountNP$_{generic=T/F}$
RBasicNPJP$_{generic=T/F}$	RdefusingcountNL$_{generic=T/F}$	LdefusingcountNP$_{generic=T/F}$
RBasicNPJP$_{generic=T/F}$	RindefmassNL$_{generic=T/F}$	LindefmassNP$_{generic=T/F}$
RBasicNPJP$_{generic=T/F}$	RdefnmassNL$_{generic=T/F}$	LdefnmassNP$_{generic=T/F}$
RBasicNPJP$_{generic=T/F}$	RdefumassNL$_{generic=T/F}$	LdefumassNP$_{generic=T/F}$

7.2.2 RPersPronoun: To make an NP of a Personal Pronoun

Personal pronouns are also nouns, but they do have their own category, as there features differ slightly from the normal noun features. The personal pronouns are words like I/we/you/he/she/it/they.

- *Japanese, RPersPronounJP:*
  Input: PersPrN(η)
  Output: NP(PersPrN(η))
  C: True
  A:
  NP.sex = PersPrN.sex &
  NP.animate = PersPrN.animate

- *Dutch, RPersPronounNL:*
  Input: PersPrN(η)
  Output: NP(PersPrN(η))
  C: True
  A:
  NP.def = def &
  NP.person = PersPrN.person &
  NP.sex = PersPrN.sex &
  NP.gnd = PersPrN.gnd &
  NP.case = PersPrN.case &
  NP.num = PersPrN.num &
  NP.animate = PersPrN.animate &
  NP.canBeBare = False

Japanese syntactic rule	Dutch syntactic rule	meaning rule
RPersPronounJP	RPersPronounNL	LPersPronoun

7.2.3 RPronoun: To make an NP of a Pronoun

Indefinite pronouns are words like ‘alles’ (everything), ‘iemand’ (someone), ‘iedereen’ (everyone).
Definite pronouns are words like ‘deze’ (this one), ‘degene’ (the person), ‘die’ (that one), ‘dezelfde’ (the same one).

- *Japanese, RPronounJP:*
  Input: PrN(η)
  Output: NP(PrN(η))
  C: True
  A:
  NP.sex = PrN.sex &
  NP.animate = PrN.animate

- *Dutch, RPronounNL:*
  Input: PrN(η)
  Output: NP(PrN(η))
  C: True
  A:
  NP.sex = PrN.sex &
  NP.animate = PrN.animate &
  NP.person = PrN.person
  NP.def = PrN.def &
  NP.case = PrN.case &
  NP.num = PrN.num &
  NP.canBeBare = False

Japanese syntactic rule	Dutch syntactic rule	meaning rule
RPronounJP	RPronounNL	LPronoun

7.2.4 RSingPluralN: Make N’ singular or plural

In Japanese there is no true plural (for the group-forming suffixes, see rule 7.2.9), so the Japanese rule does nothing and also corresponds to both Dutch rules.

All features from N’ are also copied to the output.

- *Japanese, RSingPluralNJP:*
  Input: N’(η)
  Output: N’(η)
  C: True
  A: @

- *Dutch, RSingNNL:*
  Input: N’(η)

Output: N'(η)
C:
N'.number = undefined[19]
A:
N'.number = sing

- *Dutch, RPluralNNL:*
  Input: N'(η)
  Output: N'(η)
  C:
  N'.number = undefined
  A:
  N'.number = plur

Japanese syntactic rule	Dutch syntactic rule	meaning rule
RSingPluralNJP	RSingNNL	LSingN
RSingPluralNJP	RPluralNNL	LPluralN

7.2.5 RSentence: Basic sentence rule

This rule takes a V and between 0 to 3 NPs, and makes a sentence of the input.
The NPs are distinguished by a number: NP1 for NP-ga, NP2 for NP-wo and NP3 for NP-ni. These are variables in a tree-structure model. 'NP$x(\alpha)$-particle' is shorthand for a valid NP+particle structure (which can be created by using other rules).
Notice that particles like 'de' and 'e' aren't in this basic sentence rule.
The locative 'NP-de' can be seen as an adverbial, and the 'NP-de' in passive sentence is introduced by a rule for passive sentence. The particle 'e' is used for motion to a location (lative case) and can only be used with verbs that indicate motion. The particle 'ni' can also be used in the lative case, which I also don't address explicitly here; these cases have to be translated using prepositions in Dutch.
I'm leaving out these cases for the moment in this basic sentence rule.

The number of NPs and the particles the NPs get, depend on the properties of the verb. If there are any properties that must be true, they are listed in the Conditions C. If the conditions hold true, the Actions A are applied. An Action '@' means 'take no action'.

If you only want to create a VP, use this rule with an empty NP1.

Some verbs require their object or subject to be divisible. The rules C2 through C7 are used for these cases. The feature V.objectDivisible (and similarly V.subjectDivisible and V.indirectobjectDivisible) is True if V requires its object to have the divisible feature.
Examples are V = gather (gather flowers), run out of (run out of topics). Dutch examples: verzamelen (object); scheiden, uit elkaar gaan (subject). For these cases there are the

[19] In the dictionary, the number value is listed as 'undefined'.

sub-rules C2 and C3, in which a certain NP can only combine with a certain VP if all conditions apply.

Whether or not a noun phrase is generic depends for example on the tense of the verb (typically, the verb is in the present tense in a generic sentence, see conditions C16-21 in 7.2.5), or whether the verb requires a generic object (C10, C14 and C22). In these cases, the particle of the NP2-clause is preferred to be -ga instead of -wo (so you get sentences like *NP1-wa NP2-ga suki da*).

Examples of verbs that allow/prefer a generic object are for example suki, kirai and tanoshimu. Of these, only 'tanoshimu' is technically a verb. The words 'suki' and 'kirai' are actually adjectives that get translated as a verb in Dutch and English, but combined with the copular verb 'da/desu' they do form a VP.

There are also (temporal) adverbs that specify one specific point in time which influence genericity, for example when the word 'today' is present, the sentence isn't generic (as in 'Elephants eat grass today').

The first time an NP is encountered, it is often an indefinite NP in Dutch (more on 'first time encounters' can be read in section 5.3). Most of the time (at least in English [38], and I assume most likely also in Dutch), an NP is definite. For this reason, the preferred count noun NP translation is the definite singular.
The definite singular is also the most unmarked case for generic count NPs, as described in example sentence 12, section 5. The sub-rules C8 through C11 are for adding articles and number to count nouns.

The sub-rules C12 through C15 are for adding articles and number to mass nouns. For generic mass NPs in subject (NP1) and PP-complement (NP3) positions, the bare noun is the most unmarked case (described in example sentence 18, section 5). In object position (NP2) the definite singular is preferred.
For non-generic cases, these are also the preferred forms. To create generic NPs, rule 7.2.6 can be used.

As more than one NP in the sentence can be generic, it is possible to apply more than one condition. The conditions C16 through C25 are for setting NPs to generic depending on the verb tense.

- *Japanese, RSentenceJP:*
  Input: V(α), NP1(β), NP2(γ), NP3(δ)
  Output: S(NP1(β)-ga NP3(δ)-ni NP2(γ)-wo V(α))
  C1:
  True (always apply the following actions)

  - C2:
    V.objectDivisible=True &
    NP2.divisible=True

A2: @

- C3: V.objectDivisible=False
  A3: @

- C4: V.subjectDivisible=True &
  NP1.divisible=True
  A4: @

- C5: V.subjectDivisible=False
  A5: @

- C6: V.indirectobjectDivisible=True &
  NP3.divisible=True
  A6: @

- C7: V.indirectobjectDivisible=False
  A7: @

- C8: NP1.generic = False[20]
  A8: @

- C9: NP2.generic = False
  A9: @

- C10:
  NP2.generic = False &
  V.objectPrefGeneric = True
  A10: @

- C11: NP3.generic = False
  A11: @

- C12: NP1.generic = False
  A12: @

- C13: NP2.generic = False
  A13: @

- C14:
  NP2.generic = False &

[20]These numbered C-rules for Japanese correspond to the numbered C-rules for Dutch.

V.objectPrefGeneric = True
A14: @

- C15: NP3.generic = False
  A15: @

- C16:
  NP1.generic = True &

  * C17:
    V.tense = present
    A17: bonus += 1

  * C18:
    V.tense = past
    A18: @

  A16:
  @

- C19:
  NP2.generic = True &

  * C20:
    V.tense = present
    A20:
    bonus += 1

  * C21:
    V.tense = past
    A21: @

  * C22:
    V.objectPrefGeneric = T
    A22: @

  A19:
  @

- C23:
  NP3.generic = True &

* C24:
  V.tense = present
  A24:
  bonus += 1

* C25:
  V.tense = past
  A25: @

A23:
@

A1:
NP1.case = nom &
NP2.case = acc &
NP3.case = dat

- *Dutch, RSentenceNL:*
  Input: V(α), NP1(β), NP2(γ), NP3(δ)
  Output: S(NP1(β) V(α) NP3(δ) NP2(γ))[21]
  C1[22]:
  NP1.canBeBare = False &
  NP2.canBeBare = False &
  NP3.canBeBare = False &
  NP1.number = V.number &
  NP1.person = V.person &

  - C2:
    V.objectDivisible=True &
    NP2.divisible=True
    A2: @

  - C3:
    V.objectDivisible=False
    A3: @

  - Examples for C2 and C3:

    * Count noun: Japanese:
      kitte wo atsumeru

[21]This order is for main clauses in Dutch (verb-second). For subordinate clauses there would be other rules, as the verb appears at the end of the sentence then. In a full grammar, you would add both the main clause order and the subordinate clause order.

[22]Bare nouns can only appear in other contexts, which are detailed in other rules.

stamp-OBJ collect
NP(kitte) + V(atsumeru) ⇒ VP(NP(kitte)-wo VP(atsumeru))
Dutch:
$NP_{num=plur,divisible=true}$(postzegels) + $V_{objectDivisible=true}$(verzamelen) ⇒
VP(NP(postzegels) V(verzamelen))

* Mass noun: Japanese:
  Willem wa chikara wo tsukaitsukushita
  Willem-TOPIC power-OBJ used up/ran out
  'Willem used up his energy'
  Dutch:
  Willem heeft (zijn)[23] energie opgebruikt.

- C4: V.subjectDivisible=True &
  NP1.divisible=True
  A4: @

- C5: V.subjectDivisible=False
  A5: @

- C6: V.indirectobjectDivisible=True &
  NP3.Divisible=True
  A6: @

- C7: V.indirectobjectDivisible=False
  A7: @

- C8:
  NP1.generic = False &
  NP1.countmass = count &

  * C8a:
    NP1.number = sing &
    NP1.def = def
    A8a:
    bonus += 1

  * C8b:
    NP1.num = plur
    A8b: @

[23]The possessive isn't explicit in Japanese, but it sounds better in the translation to use a possessive here instead of a normal article or no article. I won't go into this problem in this thesis.

* C8c:
  NP1.num = sing &
  NP1.def = indef
  A8c: @

A8:
@

– C9:
NP2.generic = False &
NP2.countmass = count &

  * C9a:
    NP2.number = sing &
    NP2.def = indef
    A9a:
    bonus += 1

  * C9b:
    NP2.num = plur
    A9b: @

  * C9c:
    NP2.num = sing &
    NP2.def = def
    A9c: @

  * C10:
    V.objectPrefGeneric = True &

    · C10a:
      NP2.def = def &
      NP2.num = sing
      A10a:
      @

      Example:
      Mama wa neko ga suki desu.
      Mama houdt van de kat.
      'Mama likes the cat.'

    · C10b:
      NP2.def = def &
      NP2.num = plur
      A10:

@

Example:
Mama wa neko ga suki desu.
Mama houdt van de katten.
‘Mama likes the cats.’

A9:
@

– C11:
NP3.generic = False &
NP3.countmass = count

  * C11a:
    NP3.number = sing &
    NP3.def = indef
    A11a:
    bonus += 1

  * C11b:
    NP3.num = plur
    A11b: @

  * C11c:
    NP3.num = sing &
    NP3.def = def
    A11c: @

A11:
@

– C12:
NP1.generic = False &
NP1.countmass = mass &

  * C12a:
    NP1.num = sing &
    NP1.def = indef
    A12a:
    bonus += 1

  * C12b:
    NP1.num = plur

A12b: @

* C12c:
  NP1.num = sing &
  NP1.def = def
  A12c: @

A12:
@

– C13:
NP2.generic = False &
NP2.countmass = mass &

* C13a:
  NP2.number = sing &
  NP2.def = def
  A13a:
  bonus += 1

* C13b:
  NP2.num = plur
  A13b: @

* C13c:
  NP2.num = sing &
  NP2.def = indef
  A13c: @

* C14:
  V.objectPrefGeneric = True &
  NP2.def = def &
  NP2.num = sing
  A14:
  @

  Example:
  Mama wa sake ga suki desu.
  Mama houdt van de sake.
  'Mama likes the sake.'

A13:
@

- C15:
  NP3.generic = False &
  NP3.countmass = mass &

  * C15a:
    NP3.number = sing &
    NP3.def = indef
    A15a:
    bonus += 1

  * C15b:
    NP3.num = plur
    A15b: @

  * C15c:
    NP3.num = sing &
    NP3.def = def
    A15c: @

  A15:
  @

- C16:
  NP1.generic = True &

  * C17a:
    NP1.countmass = count &
    NP1.number = plural &
    NP1.def = indef &
    V.tense = present
    A17a:
    bonus += 1

  * C17b:
    NP1.countmass = count &
    NP1.number = sing &
    NP2.def = indef &
    V.tense = present
    A17b: @

  * C17c:
    NP1.countmass = mass &
    V.tense = present
    A17c:

bonus += 1

* C18a:
  NP1.countmass = count &
  NP1.number = plural &
  NP1.def = indef &
  V.tense = past
  A18a: @

* C18b:
  NP1.countmass = count &
  NP1.number = sing &
  NP2.def = indef &
  V.tense = past
  A18b: @

* C18c:
  NP1.countmass = mass &
  V.tense = past
  A18c: @

A16:
@

– Examples:

  * Japanese:
    zou-to-iu-mono-ga happa-wo taberu
    elephants-such_as-things-SUBJ leaf-OBJ eat
    (Things such as) elephants eat leaves
    'Elephants eat leaves'.
    Dutch:
    C17a: *Olifanten* eten bladeren.
    C17b: *Een olifant* eet bladeren.
    C17c: *Vee* eet gras.

  * Japanese:
    subject: NP1(zou) ⇒ NP(NP1(zou)-to-iu)
    zou-to-iu-mono-wa banana-ga suki
    elephants-such_as-things-TOPIC banana-SUBJ like
    (Things such as) elephants like bananas
    'Elephants like bananas'.

  * Japanese:
    jazu-to-iu-mono-ga kikitai

jazz-what-is-called-thing-SUBJ want-to-hear
I want to hear what is called jazz.
'I want to hear jazz.'

Dutch:
C17c (jazz = mass noun)
Ik wil *jazz* horen. (here, the subject in Japanese becomes the object in Dutch/English)

- C19:
  NP2.generic = True &

  * C20a:
    NP2.countmass = count &
    NP2.number = plur &
    NP2.def = def &
    V.tense = present
    A20a:
    bonus += 1

  * C20b:
    NP2.countmass = count &
    NP2.number = sing &
    NP2.def = def &
    V.tense = present
    A20b: @

  * C20c:
    NP2.countmass = mass &
    V.tense = present
    A20c: @

  * Example for C20:
    Japanese:
    zou-wa happa-to-iu-mono-wo taberu
    elephant-TOPIC leaf-such-as-thing-OBJ eat
    Elephants eat (things such as) leaves.
    Dutch:
    C20a: Olifanten eten *bladeren*.
    C20b: Olifanten eten *gras*.

  * C21a:
    NP2.countmass = count &
    NP2.number = plur &
    NP2.def = def &

V.tense = past
A21a:
bonus += 1

* C21b:
  NP2.countmass = count &
  NP2.number = sing &
  NP2.def = def &
  V.tense = past
  A21b: @

* C21c:
  NP2.countmass = mass &
  V.tense = past
  A21c: @

* C22:
  V.objectPrefGeneric = True &

  · C22a:
    NP2.countmass = count &
    NP2.def = indef &
    NP2.num = plur
    A22a:
    bonus += 1

    Example :
    Mama wa neko ga suki desu.
    Mama houdt van katten.
    'Mama likes cats.'

  · C22b:
    NP2.countmass = mass &
    NP2.def = indef &
    NP2.num = sing
    A22b:
    bonus += 1

    Example:
    Mama wa sake ga suki desu.
    Mama houdt van sake.
    'Mama likes sake.'

A19:
@

- C23:
  NP3.generic = True &

  * C24a:
    NP3.countmass = count &
    NP3.number = plur &
    NP3.def = def &
    V.tense = present
    A24a:
    bonus += 1

  * C24b:
    NP3.countmass = count &
    NP3.number = sing &
    NP3.def = def &
    V.tense = present
    A24b: @

  * C24c:
    NP3.countmass = mass &
    V.tense = present
    A24c: @

  * C25a:
    NP3.countmass = count &
    NP3.number = plur &
    NP3.def = def &
    V.tense = past
    A25a:
    bonus += 1

  * C25b:
    NP3.countmass = count &
    NP3.number = sing &
    NP3.def = def &
    V.tense = past
    A25b: @

  * C25c:
    NP3.countmass = mass &
    V.tense = past
    A25c: @

  A23:

@

A1:
NP1.case = nom &
NP2.case = acc &
NP3.case = dat &
V.number = Np1.number &
V.person = Np1.person

Japanese syntactic rule	Dutch syntactic rule	meaning rule
RSentenceJP	RSentenceNL	LSentence

7.2.6 RGenericNP$_{generic=x}$: Creating a generic NP

- *Japanese, RGenericNPtoiuJP$_{generic=x}$:*
  Input: N'(η)
  Output: NP(N'(η)-to-iu-NOMINALIZER)
  NOMINALIZER = mono/koto
  C:
  x = True
  A:
  NP.generic = x

Japanese syntactic rule	Dutch syntactic rule	meaning rule
RGenericNPtoiuJP$_{generic=x}$	RindefplurcountNL$_{generic=x}$	Lindefplurcount$_{generic=True}$

7.2.7 RPNtoNP: Proper Nouns

A rule that makes a normal N' from a PN:

- *Japanese, RPNtoNJP:*
  Input: PN(ϵ)
  Output: N'(PN(ϵ))
  C: True
  A:
  N'.animate = PN.animate

- *Dutch, RPNtoNNL:*
  Input: PN(ϵ)
  Output: N'(PN(ϵ))
  C:
  N'.sex = PN.sex &
  N'.gender = utrum &
  N'.countmass = count &

N'.animate = True

The value of 'gender' depends on the kind of PN. The rules for this are:

- Names of humans and animals are male/female: "de". These personal names optionally get an article if they are preceded by an adverb/adjective: "de kleine Karin" (*the little Karin*) or "kleine Karin" (*little Karin*). Here, you can use a rule that makes a normal noun from a proper noun, as the resulting proper noun behaves like a normal noun (with articles/adverbs/adjectives/singular/plural).

- Names of countries and cities are neuter: "het" [83]. Country/city names get an article if they are preceded by an adverb/adjective: "het mooie Utrecht" ('the beautiful (city of) Utrecht)'; "het kleine Nederland" ('the small (country of) the Netherlands').

- *Japanese, RPNtoNPJP:*
  Input: PN(ϵ)
  Output: NP(PN(ϵ))
  C: True
  A:
  NP.animate = PN.animate

- *Dutch, RPNtoNPNL:*
  Input: PN(ϵ)
  Output: NP(PN(ϵ))
  C:
  NP.sex = PN.sex &
  NP.gender = utrum &
  NP.countmass = count &
  NP.animate = True

The difference between N'(PN) and NP(PN) is that N' can be used to derive 'de kleine Karin' (*the little Karin*), while the NP is just for 'Karin' without any adjectives or other modifiers.

Example:

- Japanese:
  PN(Wiremu) $\Rightarrow$ NP$_{animate=True}$(PN(Wiremu))
  Dutch:
  PN(Willem) $\Rightarrow$ NP$_{sex=male,gender=utrum,count=count,animate=True}$
  (PN$_{sex=male,gender=utrum,count=count}$(Willem))

Japanese syntactic rule	Dutch syntactic rule	meaning rule
RPNtoNJP	RPNtoNNL	LPNtoN
RPNtoNPJP	RPNtoNPNL	LPNtoNP

7.2.8 RHonAff: Adding honorific affixes to Proper Nouns

The next rules are for adding affixes to proper nouns (basic expressions, the affixes, are shown in table 7 on page 71 (based on table 19 on page 126)). Honorific affixes can only be added when the 'hon' feature is still False, as you cannot have multiple honorific affixes.

- *Japanese, RHonAffJP:*
  Input: PN1(ϵ), aff(ζ)
  Output: PN2(PN1(ϵ) aff(ζ))
  C:
  PN1.sex = aff.sex &
  aff.type = honorific &
  PN1.hon = False &
  PN1.divisible = False
  A:
  PN2.sex = PN1.sex &
  PN2.hon = True &
  PN2.divisible = PN1.divisible

- *Dutch, RHonAffNL:*
  Input: PN1(ϵ), aff(ζ)
  Output: PN2(aff(ζ) PN1(ϵ))
  C:
  PN1.sex = aff.sex &
  aff.type = honorific &
  PN1.hon = False &
  PN1.divisible = False
  A:
  PN2.def = indef &
  PN2.hon = True &
  PN2.sex = PN1.sex &
  PN2.num = PN1.num &
  PN2.divisible = PN1.divisible

- Examples:

  - Japanese:
    $\text{PN}_{sex=male,num=sing,hon=False,divisible=False}$(Wiremu) + $\text{aff}_{sex=male,num=sing}$(kun)
    $\Rightarrow$
    $\text{PN}_{sex=male,num=sing,hon=True,divisible=False}$(Wiremu-kun)
    Dutch:
    $\text{aff}_{sex=male,num=sing}$(meneer) + $\text{PN}_{sex=male,num=sing,hon=False,divisible=False}$(Willem)
    $\Rightarrow$ $\text{PN}_{sex=male,num=sing,def=indef,hon=True,divisible=False}$(meneer Willem)

Japanese syntactic rule	Dutch syntactic rule	meaning rule
RHonAffJP	RHonAffNL	LHonAff

7.2.9 RPlurSuff: Adding group-forming suffixes

This rule can be applied to proper nouns, as there is a rule that makes an N' from a PN, and a rule which makes an NP from an N'. However, the NP should be animate.
The pluralizing affixes are also listed in table 7 on page 71.

- *Japanese, RPlurSuffJP:*
  Input: PN(η)
  Output: NP(PN(η)-aff(ζ))
  C:
  PN.animate = True &
  PN.divisible = False &
  PN $\neq$ PN(η)-aff(ζ)
  A:
  NP.divisible = True &
  aff.type = group[24]

- *Dutch, RPlurSuffNL:*
  Input: PN(η)
  Output: NP(PN(η) en de anderen)
  C:
  PN.num = sing &
  PN.def = def &
  PN.animate = True &
  PN.divisible = False
  A:
  NP.num = plur &
  NP.divisible = True

- Example:

  - Japanese:
    NP(Wiremu) $\Rightarrow$ NP(NP(Wiremu)-tachi)
    Dutch:
    NP(Willem) $\Rightarrow$ NP(NP(Willem) en de anderen)

Japanese syntactic rule	Dutch syntactic rule	meaning rule
RPlurSuffJP	RPlurSuffNL	LPlurSuff

[24]Values of aff.type = group are -kata/-gata/-tachi/-ra/-domo.

Japanese basic expressions	Dutch basic expressions	basic meanings
$\text{aff}_{num=sing}$(san)	$\text{aff}_{sex=m}$(meneer), $\text{aff}_{sex=f}$(mevrouw), $\text{aff}_{sex=undef}$(∅)	*Mr/Mrs'*
$\text{aff}_{num=sing}$(sensei)	$\text{aff}_{sex=m}$(meester), $\text{aff}_{sex=f}$(juffrouw), $\text{aff}_{sex=undef}$(∅)	*teacher'*
$\text{aff}_{num=sing}$(sama)	$\text{aff}_{sex=m}$(hooggeachte meneer), $\text{aff}_{sex=f}$(hooggeachte mevrouw), $\text{aff}_{sex=undef}$(hooggeachte)	*Esteemed Mr/Mrs'*
$\text{aff}_{num=sing}$(chan)	$\text{aff}_{sex=undef}$(kleine)	*little'*
$\text{aff}_{sex=m,num=sing}$(bou)	$\text{aff}_{sex=m}$(kleine)	*little'*
$\text{aff}_{sex=m,num=sing}$(kun)	$\text{aff}_{sex=m}$(∅)	*Mr'*
$\text{aff}_{num=sing}$(senpai)	$\text{aff}_{sex=undef}$(∅)	*senior'*
$\text{aff}_{num=sing}$(hakase)	$\text{aff}_{sex=undef}$(professor)	*professor'*
$\text{aff}_{num=plur}$(tachi)	$\text{aff}_{num=plur}$(en de anderen)	*and the others'*
$\text{aff}_{num=plur}$(kata)	$\text{aff}_{num=plur}$(en de anderen)	*and the others'*
$\text{aff}_{num=plur}$(gata)	$\text{aff}_{num=plur}$(en de anderen)	*and the others'*
$\text{aff}_{num=plur}$(ra)	$\text{aff}_{num=plur}$(en de anderen)	*and the others'*
$\text{aff}_{num=plur}$(domo)	$\text{aff}_{num=plur}$(en de anderen)	*and the others'*

Table 7: Basic expressions for Japanese and Dutch honorific affixes.

7.2.10 RAimedAtNP: 'aimed at' + NP

- *Japanese, RAimedAtNPJP:*
  Input: NP(η)
  Output: NP(NP(η Particle(muke))
  C: True
  A: @

- *Dutch, RAimedAtNPNL:*
  Input: NP(η)
  Output: NP(gericht op NP(η))
  C1:
  NP.num = plur &
  NP.countmass = count &
  NP.def = indef
  A1:
  bonus += 1
  C2:
  NP.countmass = mass &
  NP.def = indef
  C3:
  NP.num = sing &
  NP.def = def
  A3:

bonus += 1
C4:
NP.num = plur &
NP.def = def
A4:
@

- Examples:
  - Japanese:
    NP(josei) ⇒ NP(NP(josei) Particle(muke))
    Dutch:
    C1: NP(vrouw) ⇒ PP(gericht op $NP_{num=plur,def=indef}$(vrouwen)).
    C3: NP(vrouw) ⇒ PP(gericht op $NP_{num=sing,def=def}$(de vrouw)).
    C4: NP(vrouw) ⇒ PP(gericht op $NP_{num=plur,def=def}$(de vrouwen)).

Japanese syntactic rule	Dutch syntactic rule	meaning rule
RAimedAtNPJP	RAimedAtNPNL	LAimedAtNP

7.2.11 RIdentSent: Identificational sentences

An identificational sentence has a definite NP in predicate position (NP2). A predicational sentence has an indefinite NP2. If the N2 is an occupation (which is one of the kind of nouns that has the canBeBare-feature value set to True), then there's no article in Dutch.

- *Japanese, RIdentSentJP:*
  Input: NP1(β), NP2(γ)
  Output: S(NP1(β) wa NP2(γ) V(da))
  C: True
  A: @

- *Dutch, RIdentSentNL:*
  Input: NP1(β), NP2(γ)
  Output: S(NP1(β) V(zijn) NP2(γ))
  C:
  NP1.num = V.num &
  NP1.sex = NP2.sex &
  NP2.def = def &
  - C1:
    NP2 = bare N &
    NP2.canBeBare = True

A1: @

- C2: NP2.canBeBare = False
  A2: @

A: @

- Examples:

  - Japanese:
    NP1(Tokyo) + NP2(shuto) ⇒ S(NP1(Tokyo) wa NP2(shuto) V(desu))
    Dutch:
    NP1(Tokyo) + NP2(hoofdstad) ⇒ S($NP1_{num=sing}$(Tokyo)
    $V_{num=num(NP2),person=person(NP2)}$(is) $NP2_{num=sing,def=def}$(de hoofdstad))
  - Japanese:
    NP1(Wiremu) + $NP2_{canBeBare=true}$(sensei) ⇒ S(NP1(Wiremu) wa $NP2_{canBeBare=true}$(sensei)
    V(desu))
    Dutch:
    NP1(Willem) + $NP2_{canBeBare=true}$(docent) ⇒ S($NP1_{num=sing,sex=male}$(Willem)
    $V_{num=num(NP1),person=person(NP1)}$(is) $NP2_{num=sing,def=indef,sex=male,canBeBare=true}$(docent))

Japanese syntactic rule	Dutch syntactic rule	meaning rule
RIdentSentJP	RIdentSentNL	LIdentSent

7.2.12 RPredSent: Predicational sentences

Just as verbs can prefer a generic argument, NP2 here could also prefer a generic argument (NP1). However, I don't explicitly check for that in these rules.

- *Japanese, RPredSentJP:*
  Input: NP1(β), NP2(γ)
  Output: S(NP1(β) ga NP2(γ) V(da))
  C: True
  A: @

- *Dutch, RPredSentNL:*
  Input: NP1(β), NP2(γ)
  Output: S(NP1(β) V(zijn) NP2(γ))
  C:
  NP1.num = V.num &
  NP2.def = indef &

– C1:
NP2 = bare N &
NP2.canBeBare = True
A1: @

– C2: NP2.canBeBare = False
A2: @

A:
@

- Example:

  – Japanese:
  NP1(Wiremu) + NP2(ningen) ⇒ S(NP1(Wiremu) ga NP2(ningen) V(desu))
  Dutch:
  NP1(Willem) + NP2(mens) ⇒ S(NP1$_{num=sing,sex=male}$(Willem)
  V$_{num=num(NP2),person=person(NP2)}$(is) NP2$_{num=sing,def=indef,sex=male}$(een mens))
  – Japanese:
  NP1(Wiremu) + NP2(sensei) ⇒ S(NP1(Wiremu) ga NP2(sensei) V(desu))
  Dutch:
  NP1(Willem) + NP2(docent) ⇒ S(NP1$_{num=sing,sex=male}$(Willem)
  V$_{num=num(NP2),person=person(NP2)}$(is) NP2$_{canBeBare=True}$(docent))

Japanese syntactic rule	Dutch syntactic rule	meaning rule
RPredSentJP	RPredSentNL	LPredSent

7.2.13 RPredSentAdj: Predicational sentences with adjectives

Just as verbs can prefer a generic argument, ADJ here could also prefer a generic argument (NP1). However, I don't explicitly check for that in these rules.

- *Japanese, RPredSentAdjJP:*
  Input: NP1(β), ADJ(γ)
  Output: S(NP1(β) ga ADJ(γ) V(da))
  C: True
  A: @

- *Dutch, RPredSentAdjNL:*
  Input: NP1(β), ADJ(γ)
  Output: S(NP1(β) V(zijn) ADJ(γ))
  C: True

- C1:
  NP1.num = plur &
  NP1.def = indef &
  NP1.generic = True
  A1:
  @

- C2:
  NP1.num = sing &
  NP1.def = def &
  NP1.generic = False
  A2:
  bonus += 1

- C3:
  NP1.num = plur &
  NP1.def = def &
  NP1.generic = False
  A3:
  @

A:
V.number = NP.number
V.person = NP.person

- Example:

  - Japanese:
    NP1(bara) + ADJ(akai) ⇒ S(NP(bara) ga ADJ(akai) V(desu))
    Dutch:
    NP(roos) + ADJ(rood) ⇒ S(NP$_{num=plur,def=indef}$(rozen)
    V$_{num=num(NP),person=person(NP)}$(zijn) ADJ(rood))

There is also the Japanese word 'aru', which is the 'to be'-verb for inanimate objects. I didn't write a rule for this, however. An example would be 'resutoran ga aru', meaning 'there are restaurants'.

Japanese syntactic rule	Dutch syntactic rule	meaning rule
RPredSentAdjJP	RPredSentAdjNL	LPredSentAdj

7.2.14 RAsNP: 'as' (toshite)

- *Japanese, RAsNPJP:*
  Input: NP(η)
  Output: NP(NP(η) Particle(toshite))

C: True
A: @

- *Dutch, RAsNPNL:*
  Input: NP(η)
  Output: AdvP(als NP(η))
  C:
  NP.canBeBare = True &
  NP.num = sing &
  NP.def = indef
  A: @

- Example:
  - Japanese:
    NP(sensei) ⇒ NP(NP(sensei) Particle(-toshite))
    Dutch:
    $\text{NP}_{canBeBare=True,sex=male}$(docent) ⇒ AdvP(als $\text{NP}_{def=indef,num=sing,sex=male}$(docent))
    For example in the sentence 'Hij werkt als docent' (he works as a teacher).

Japanese syntactic rule	Dutch syntactic rule	meaning rule
RAsNPJP	RAsNPNL	LAsNP

7.2.15 RTopic: Making a topic phrase of a ga/ni/wo/de phrase (i)

Topicalization in Japanese doesn't have to correspond with putting the topicalized phrase in front of the sentence in Dutch, but that is a subject you could write another thesis about, so that is a problem outside the scope of this thesis.
If the NP is a topic, a definite NP is preferred in Dutch.
These rules should only be applicable once (in Japanese this is blocked by the inclusion of case particles, but in Dutch there are no visible case particles), so for this reason these topicalization rules can only be applied *once per sentence.*
As these rules should be reversible, they actually should be formulated in a more complex way.

- *Japanese, RTopicSubjJP$_i$:*
  Input: S[...NP(NP1$_i$(β)-ga)...V]
  Output: S[NP(NP1$_i$(β)-wa)...V]
  C: True
  A: @

- *Dutch, RTopicSubjNL$_i$:*
  Input: S[...NP1$_i$(β)...V]
  Output: S[NP1$_i$(β)...V]

C: True
A: @

- *Japanese, RTopicObjJP$_i$:*
  Input: S[...NP(NP2$_i$(γ)-wo)...V]
  Output: S[NP(NP2$_i$(γ)-wa)...V]
  C: True
  A: @

- *Dutch, RTopicObjNL$_i$:*
  Input: S[...NP2$_i$(γ)...V]
  Output: S[NP2$_i$(γ)...V]
  C: True
  A: @

- *Japanese, RTopicPPJP$_i$:*
  Input: S[...PP(NP$_i$(η)-Particle(θ))...V]
  Output: S[PP(NP$_i$(η)-Particle(θ) wa)...V]
  C: True
  A: @

- *Dutch, RTopicPPNL$_i$:*
  Input: S[...PP$_i$(ϑ)...V]
  Output: S[PP$_i$(ϑ)...V]
  C: True
  A: @

Each NP in a sentence has an index i. If NPs have the same index, they refer to the same entity. The index is a parameter of this rule.
The particles 'ni' and 'de' get translated with words like 'in', 'at', 'with', etc.

Japanese syntactic rule	Dutch syntactic rule	meaning rule
RTopicSubjJP$_i$	RTopicSubjNL$_i$	LTopicSubj$_i$
RTopicObjJP$_i$	RTopicObjNL$_i$	LTopicObj$_i$
RTopicPPJP$_i$	RTopicPPNL$_i$	LTopicPP$_i$

For each of the above rules, the following conditions can also be applied to the output (the -wa clause).

- *Japanese, RTopicTensePastJP:*
  C:
  V.tense = past

A:
bonus += 1

- *Dutch, RTopicTensePastNL:*
  C:
  V.tense = past &
  NP1.num = V.num &

  - C1:
    NP1.num = sing &
    NP1.def = def
    A1:
    bonus += 1

  - C2:
    NP1.num = plur &
    NP1.def = def
    A2:
    @

  - C3:
    NP1.num = sing &
    NP1.def = indef
    A3:
    @

  - C4:
    NP1.num = plur &
    NP1.def = indef
    A4:
    @

  A:
  @

- Examples:
  Japanese: densha-wa hashitta
  train-TOPIC run-PAST
  "The train(s) ran."
  Dutch:
  C1: De trein reed.
  C2: De treinen reden.

C3: Een trein reed.[25]
C4: Treinen reden.

- *Japanese, RTopicTensePresentJP:*
  C:
  V.tense = present
  A:
  @

- *Dutch, RTopicTensePresentNL:*
  C:
  NP1.num = V.num &
  V.tense = present

  - C1:
    NP1.num = plur &
    NP1.def = indef
    A1:
    bonus += 1

  - C2:
    NP1.num = sing &
    NP1.def = indef
    A2:
    @

  - C3:
    NP1.def = def
    A3:
    @

  A:
  @

- Examples:
  Japanese: densha-wa hashiru
  train-TOPIC run
  "The train(s) run."
  Dutch:

[25] In Dutch it's more natural to say 'Er rijdt een trein' (There runs a train). 'Een trein' is a non-specific NP, and when such an NP appears in subject position, the temporary subject *er* ('there') is normally used.

C1: Treinen rijden.
C2: Een trein rijdt.
C3: De trein rijdt. & De treinen rijden.

Japanese syntactic rule	Dutch syntactic rule	meaning rule
RTopicTensePastJP	RTopicTensePastNL	LTopicTensePast
RTopicTensePresentJP	RTopicTensePresentNL	LTopicTensePresent

7.2.16 RAlso: 'also/too' (mo)

The particle 'mo' means 'also'. It attaches to an NP, which is not part of a sentence yet. When the NP is inserted into a sentence structure and the particle 'ga' or 'wo' has to be added while the NP has 'mo' attached already, 'ga' and 'wo' are not added. If an NP has another particle attached, for example 'ni', 'de', or 'kara', 'mo' follows that particle (resulting in 'ni-mo', 'de-mo', etc). These correct particle structures would be created by using other rules (not specified here).

- *Japanese, RAlsoJP:*
  Input: NP(η)
  Output: NP(NP(η-mo))
  C: True
  A: @

- *Dutch, RAlsoNL:*
  Input: NP(η)
  Output: NP(ook NP(η))
  C1:
  NP.def = def
  A1:
  bonus += 1
  C2:
  NP.def = indef
  A2:
  @

- Examples:
  - Japanese:
    NP(Wiremu) ⇒ NP(NP(Wiremu)-mo)
    NP(hon) ⇒ NP(NP(hon)-mo)
    Dutch:
    NP(Willem) ⇒ NP(ook NP(Willem))
    NP(boek) ⇒ NP(ook NP(het boek))

- Japanese:
  NP(hon) ⇒ NP(NP(hon)-mo)
  Dutch:
  NP(boek) ⇒ NP(ook NP(boeken))

Japanese syntactic rule	Dutch syntactic rule	meaning rule
RAlsoJP	RAlsoNL	LAlso

7.2.17 RDirSuff: Directional suffixes

- *Japanese, RDirSuffJP:*
  Input: Particle(ζ), NP(η)
  Output: NP(NP(η) Particle(ζ))
  C: True
  A: @

- *Dutch, RDirSuffPreNL:*
  Input: P(ζ), NP(η)
  Output: PP(P(ζ) NP(η))
  C:
  P.type = pre
  A:
  @

- *Dutch, RDirSuffPostNL:*
  Input: P(ζ), NP(η)
  Output: PP(NP(η) P(ζ))
  C:
  P.type = post
  A:
  @

- Examples:

  - Japanese:
    Particle(e) + NP(Toukyou) ⇒ NP(NP(Toukyou) Particle(e))
    Dutch:
    $P_{type=pre}$(naar) + NP(Tokyo) ⇒ PP($P_{type=pre}$(naar) NP(Tokyo))

  - Japanese:
    NP(yama) + Particle(no) + NP(shita) + Particle(e) ⇒ NP(NP(yama)-no-NP(shita)-Particle(e))
    Dutch:
    NP(de berg) + $P_{type=post}$(af) ⇒ PP(NP(de berg) $P_{type=post}$(af))

Particles from Japanese lexicon	Prepositions from Dutch lexicon	meaning
e	naar($P_{type=prep}$)	*to'*
kara	vanaf($P_{type=prep}$)	*from'*
made	tot($P_{type=prep}$)	*up to'*
(NP-no) shita e	af($P_{type=post}$)	*down'*

Japanese syntactic rule	Dutch syntactic rule	meaning rule
RDirSuffJP	RDirSuffPreNL	LDirSuff
RDirSuffJP	RDirSuffPostNL	LDirSuff

7.2.18 RNomVerb: Nominalizing verbs

- *Japanese, RNomVerbJP:*
  Input: VP(α)
  Output: NP(VP(α)-NOMINALIZER)
  nominalizer = no, mono, koto
  C: True
  A: @

- *Dutch, RNomVerbNL:*
  Input: VP(α)
  Output: NP(VP(α))
  C:
  NP.def = def &
  NP.num = sing &
  NP.gnd = neutr &
  VP.person = infinitive
  A: @

- Examples:
  - kaku (to write) ⇒ kaku-koto
    schrijven-nominalizer
    schrijven (Example sentence: 'Brieven schrijven is leuk.')
  - Japanese: watashi wa tegami-wo-kaku-koto-ga suki desu. (I like to write letters)
    Dutch: Brieven schrijven vind ik leuk.

Japanese syntactic rule	Dutch syntactic rule	meaning rule
RNomVerbJP	RNomVerbNL	LNomVerb

7.2.19 RNumeral: Numerals

- *Japanese, RNumeralJP:*
  Input: NUMERAL(ν), NP(η)

Output: NP(NUMERAL(ν)-CLASSIFIER[26]-no-NP(η)[27]
C: True
A: @

- *Dutch, RNumeralNL:*
  Input: NUMERAL(ν), NP(η)
  Output: NP(NUMERAL(ν) NP(η))
  C:
  NUMERAL.requiresnumber = NP.num
  NP.def = indef

- Examples:

  - Japanese:
    NUMERAL(1) + NP(yakusoku) ⇒ NP(NUMERAL(1)-tsu-no-NP(yakusoku))
    Dutch:
    1 + NP(belofte) ⇒ NP(NUMERAL$_{requiresnumber=sing}$(1) NP$_{def=indef,num=sing}$(belofte))

Japanese syntactic rule	Dutch syntactic rule	meaning rule
RNumeralJP	RNumeralNL	LNumeral

7.2.20 RQuant: Quantifiers specifying sing/plur

- *Japanese, RQuantJP:*
  Input: QUANTIFIER(μ), NP(η)
  Output: NP(QUANTIFIER(μ)-no-NP(η))
  C: True
  A: @

- *Dutch, RQuantNL:*
  Input: QUANTIFIER(μ), NP(η)
  Output: NP(QUANTIFIER(μ) NP(η))
  C:
  QUANTIFIER.requiresCountmass = NP.countmass &
  QUANTIFIER.def = NP.def &
  QUANTIFIER.requiresNumber = NP.num

- Example:

[26]The classifier is chosen on the basis of the features of the NP; default is the classifier '-tsu', as in '1-tsu'.

[27]As can be seen in table 4 on page 34, other word orders in Japanese can give the same translation, but for the moment I am not expanding on those here.

– Japanese:
QUANTIFIER(hotondo) + NP(hito) ⇒ NP(QUANTIFIER(hotondo)-no-NP(hito))
Dutch:
QUANTIFIER$_{countmass=count,def=def,num=plur}$(de meeste) + NP$_{countmass=count}$(mens)
⇒ NP(QUANTIFIER(de meeste)-NP$_{countmass=count,def=def,num=plur}$(mensen)

Quantifiers in Japanese lexicon	Quantifiers/Articles in Dutch lexicon	meaning
onoono	elke + $NP_{def=indef,num=sing}$	*each'*
zenbu/subete	al het/de + $NP_{countmass=mass,def=def,num=sing}$	*all'*
zenbu/subete	alle + $NP_{countmass=count,def=indef,num=plur}$	*all'*
ryouhou	beide + $NP_{countmass=count,def=indef,num=plur}$	*both'*
zenkoku	$NP_{countmass=count,def=indef,num=plur}$ + in het hele land	*all over the country'*
takusan/kazukazu	veel + $NP_{countmass=count,def=indef,num=plur}$	*many'*
samazama	diverse + $NP_{countmass=count,def=indef,num=plur}$	*various'*
hotondo	de meeste + $NP_{countmass=count,def=def,num=plur}$	*most'*
hanbun	de halve + $NP_{countmass=count,def=def,num=sing}$	*most'*
sukoshi	een beetje + $NP_{countmass=mass,def=indef,num=plur}$	*a little'*
sukoshi	een paar + $NP_{countmass=count,def=indef,num=plur}$	*a few'*

Japanese syntactic rule	Dutch syntactic rule	meaning rule
RQuantJP	RQuantNL	LQuant

7.3 Overview of the mapping rules

Japanese syntactic rule	**Dutch syntactic rule**	**meaning rule**
RBasicNPJP$_{generic=x}$	RBasicNPNL$_{generic=x}$	LBasicNP$_{generic=x}$
RBasicNPJP$_{generic=x}$	RindefsingcountNL$_{generic=x}$	LindefsingcountNP$_{generic=x}$
RBasicNPJP$_{generic=x}$	RindefplurcountNL$_{generic=x}$	LindefplurcountNP$_{generic=x}$
RGenericNPtoiuJP$_{generic=x}$	RindefplurcountNL$_{generic=x}$	Lindefplurcount$_{generic=True}$
RBasicNPJP$_{generic=x}$	RdefnsingcountNL$_{generic=x}$	LdefnsingcountNP$_{generic=x}$
RBasicNPJP$_{generic=x}$	RdefusingcountNL$_{generic=x}$	LdefusingcountNP$_{generic=x}$
RBasicNPJP$_{generic=x}$	RindefmassNL$_{generic=x}$	LindefmassNP$_{generic=x}$
RBasicNPJP$_{generic=x}$	RdefnmassNL$_{generic=x}$	LdefnmassNP$_{generic=x}$
RBasicNPJP$_{generic=x}$	RdefumassNL$_{generic=x}$	LdefumassNP$_{generic=x}$
RPersPronounJP	RPersPronounNL	LPersPronoun
RPronounJP	RPronounNL	LPronoun
RSingPluralNJP	RSingNNL	LSingN
RSingPluralNJP	RPluralNNL	LPluralN
RSentenceJP	RSentenceNL	LSentence
RPNtoNJP	RPNtoNNL	LPNtoN
RPNtoNPJP	RPNtoNPNL	LPNtoNP
RHonAffJP	RHonAffNL	LHonAff
RPlurSuffJP	RPlurSuffNL	LPlurSuff
RAimedAtNPJP	RAimedAtNPNL	LAimedAtNP
RIdentSentJP	RIdentSentNL	LIdentSent
RPredSentJP	RPredSentNL	LPredSent
RPredSentAdjJP	RPredSentAdjNL	LPredSentAdj
RAsNPJP	RAsNPNL	LAsNP
RTopicSubjJP$_i$	RTopicSubjNL$_i$	LTopicSubj$_i$
RTopicObjJP$_i$	RTopicObjNL$_i$	LTopicObj$_i$
RTopicPPJP$_i$	RTopicPPNL$_i$	LTopicPP$_i$
RTopicTensePastJP	RTopicTensePastNL	LTopicTensePast
RTopicTensePresentJP	RTopicTensePresentNL	LTopicTensePresent
RAlsoJP	RAlsoNL	LAlso
RDirSuffJP	RDirSuffPreNL	LDirSuff
RDirSuffJP	RDirSuffPostNL	LDirSuff
RNomVerbJP	RNomVerbNL	LNomVerb
RNumeralJP	RNumeralNL	LNumeral
RQuantJP	RQuantNL	LQuant

7.4 Example derivations

The basic expressions are written as a B followed by the word itself.

7.4.1 'The book is red.'

- I'm going to derive a Japanese sentence 本が赤いです。 (*hon ga akai desu*) and a Dutch sentence (multiple possible translations) simultaneously. Because the grammars are reversible, you can go from Japanese to Dutch, but also from Dutch to Japanese.

- Basic expressions from dictionary[28]:

Japanese basic expressions	**Dutch basic expressions**
Bhon (N): - N.sex = undefined - N.divisible = False - N.generic = False - N.animate = False - N.person = 3rd	Bboek (N): - N.sex = undefined - N.divisible = False - N.generic = False - N.animate = False - N.person = 3rd - N.case = undefined - N.number = undefined - N.gender = neutrum - N.def = undefined - N.countmass = count - N.canBeBare = False
Bakai (Adj): no features defined	Brood (Adj): no features defined

  There is a rule that takes the basic expression N(Bboek) and makes it *N'(Bboek)* singular or *N'(Bboeken)* plural. It copies all the features of N, and also sets the number feature (sing/plur). This rule is described in 7.2.4 and I assume below this rule has been applied.

- Applying rules:

Japanese rules	**Dutch rules**
Apply rule 7.2.1 RBasicNPJP$_{generic=False}$*:* Input: N'(Bhon) Output: NP(N'(Bhon)) - NP.sex = undefined - NP.divisible = False - NP.generic = False - NP.animate = False	*Apply rule 7.2.1 RindefsingcountNL*$_{generic=False}$*:* Input: N'(Bboek) Output: NP(ART(een) N'(Bboek)) - NP.sex = undefined - NP.divisible = False - NP.generic = False - NP.animate = False

[28]To look up the words in the dictionary, the Japanese characters are used.

- NP.person = 3rd	- NP.person = 3rd - NP.case = undefined - NP.number = sing - NP.gender = neutrum - *NP.def = indef* - NP.countmass = count - NP.canBeBare = False *Apply rule 7.2.1 RdefnsingcountNL$_{generic=False}$:* Input: N'(Bboek) Output: NP(ART(het) N'(Bboek)) - NP.sex = undefined - NP.divisible = False - NP.generic = False - NP.animate = False - NP.person = 3rd - NP.case = undefined - NP.number = sing - NP.gender = neutrum - *NP.def = def* - NP.countmass = count - NP.canBeBare = False *Apply rule 7.2.1 RindefplurcountNL$_{generic=False}$:* Input: N'(Bboeken) Output: NP(N'(Bboeken)) - NP.sex = undefined - NP.divisible = True - NP.generic = False - NP.animate = False - NP.person = 3rd - NP.case = undefined - NP.number = plur - NP.gender = neutrum - *NP.def = def* - NP.countmass = count - NP.canBeBare = False *Apply rule 7.2.1 RdefplurcountNL$_{generic=False}$:* Input: N'(Bboeken) Output: NP(ART(de) N'(Bboeken)) - NP.sex = undefined - NP.divisible = True - NP.generic = False - NP.animate = False - NP.person = 3rd - NP.case = undefined - NP.number = plur - NP.gender = neutrum

	- *NP.def = def* - NP.countmass = count - NP.canBeBare = False
Apply rule 7.2.13 RPredSentAdjJP: Input: NP(N'(Bhon)), ADJ(Bakai) Output: S(NP1(NP(N'(Bhon))) ga ADJ(Bakai) V(da))[29]	*Apply rule 7.2.13 RPredSentAdjNL:* Input: NP1(Bboek), ADJ(Brood) Output (bonus): S(NP(NP(ART(het) N'(Bboek)) V(is) ADJ(Brood)) *V.number = NP.number* *V.person = NP.person* Output (no bonus): S(NP(NP(ART(de) N'(Bboeken))) V(zijn) ADJ(Brood)) *V.number = NP.number* *V.person = NP.person*

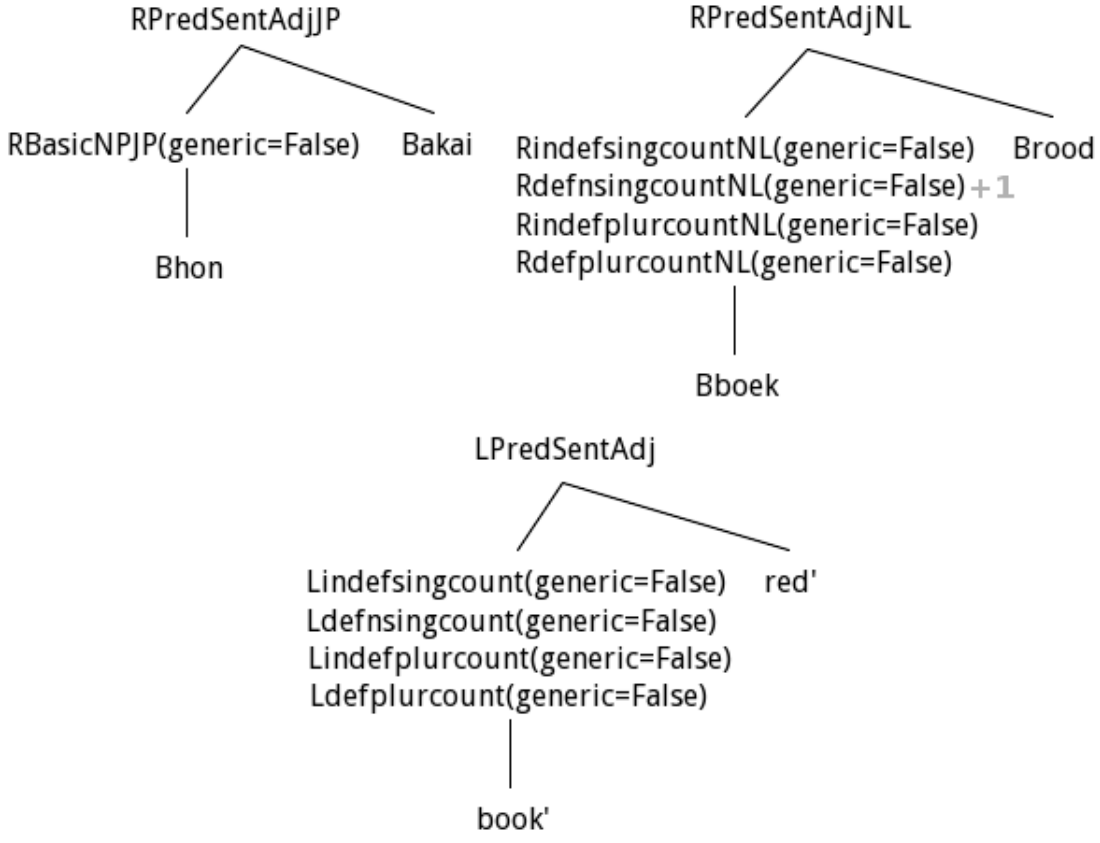


Figure 10: Syntactic derivation trees and the semantic interlingua derivation tree for 'The book is red'. In the Dutch syntactic derivation tree and the semantic interlingua derivation tree, there are multiple rules in one node. This is a shorthand notation to avoid writing four very similar trees with only a different rule in one of the nodes. The *+1* means that a bonus of 1 has been applied.

[29]The verb 'desu' is the more polite conjugation of 'da'.

7.4.2 'Anna is reading the book.'

- Japanese sentence 杏奈は本を読んでいる。 (*Anna wa hon wo yondeiru*) and possible Dutch translations.

- Basic expressions from dictionary[30]:

Japanese basic expressions	**Dutch basic expressions**
BAnna (PN) - PN.number = sing - PN.sex = female - PN.hon = False - PN.divisible = False - PN.animate = True	BAnna (PN) - PN.number = sing - PN.sex = female - PN.hon = False - PN.divisible = False - PN.animate = True
Bhon (N): - N.sex = undefined - N.divisible = False - N.generic = False - N.animate = False - N.person = 3rd	Bboek (N): - N.sex = undefined - N.divisible = False - N.generic = False - N.animate = False - N.person = 3rd - N.case = undefined - N.number = undefined - N.gender = neutrum - N.def = undefined - N.countmass = count - N.canBeBare = False
Byondeiru (V)[31]: - V.tense = present[32] - V.objectDivisible = False - V.subjectDivisible = False - V.indirectobjectDivisible = False - V.objectPrefGeneric = False - V.subjectPrefGeneric = False - V.indirectobjectPrefGeneric = False	Blezen (V): - V.tense = present - V.objectDivisible = False - V.subjectDivisible = False - V.indirectobjectDivisible = False - V.objectPrefGeneric = False - V.subjectPrefGeneric = False - V.indirectobjectPrefGeneric = False

  The rule 7.2.4 that takes the basic expression N(Bboek) and makes it *N'(Bboek)* singular or *N'(Bboeken)* plural, has been applied. It copies all the features of N, and also sets the number feature (sing/plur).

- Applying rules:

[30]To look up the words in the dictionary, the Japanese characters are used.
[31]Information about verb forms should be present as well.
[32]I didn't write any rules to get tense.

Japanese rules	**Dutch rules**
Apply rule 7.2.7 RPNtoNPJP: Input: PN(BAnna) Output: NP(PN(BAnna)) - NP.sex = female - NP.hon = False - NP.divisible = False - NP.animate = True	*Apply rule 7.2.7 RPNtoNPNL:* Input: PN(BAnna) Output: NP(PN(BAnna)) - NP.number = sing - NP.sex = female - NP.hon = False - NP.divisible = False - NP.animate = True - *NP.countmass = count* - *NP.gender = utrum*
Apply rule 7.2.1 RBasicNPJP$_{generic=False}$: Input: N'(Bhon) Output: NP(N'(Bhon)) - NP.sex = undefined - NP.divisible = False - NP.generic = False - NP.animate = False - NP.person = 3rd	*Apply rule 7.2.1 RindefsingcountNL$_{generic=False}$:* Input: N'(Bboek) Output: NP(ART(een) N'(Bboek)) - NP.sex = undefined - NP.divisible = False - NP.generic = False - NP.animate = False - NP.person = 3rd - NP.case = undefined - NP.number = sing - NP.gender = neutrum - *NP.def = indef* - NP.countmass = count - NP.canBeBare = False *Apply rule 7.2.1 RdefnsingcountNL$_{generic=False}$:* Input: N'(Bboek) Output: NP(ART(het) N'(Bboek)) - NP.sex = undefined - NP.divisible = False - NP.generic = False - NP.animate = False - NP.person = 3rd - NP.case = undefined - NP.number = sing - NP.gender = neutrum - *NP.def = def* - NP.countmass = count - NP.canBeBare = False *Apply rule 7.2.1 RindefplurcountNL$_{generic=False}$:* Input: N'(Bboeken) Output: NP(N'(Bboeken)) - NP.sex = undefined - NP.divisible = True - NP.generic = False

	- NP.animate = False - NP.person = 3rd - NP.case = undefined - NP.number = plur - NP.gender = neutrum - *NP.def = def* - NP.countmass = count - NP.canBeBare = False *Apply rule 7.2.1 RdefplurcountNL$_{generic=False}$:* Input: N'(Bboeken) Output: NP(ART(de) N'(Bboeken)) - NP.sex = undefined - NP.divisible = True - NP.generic = False - NP.animate = False - NP.person = 3rd - NP.case = undefined - NP.number = plur - NP.gender = neutrum - *NP.def = def* - NP.countmass = count - NP.canBeBare = False
Apply rule 7.2.5 RSentenceJP: Input: V(Byondeiru), NP1(NP(N'(PN(BAnna)))), NP2(NP(N'(Bhon))) Output: S(NP1(NP1(N'(PN(BAnna))))-ga NP2(NP(N'(Bhon)))-wo V(Byondeiru)) *NP1.case = nom* *NP2.case = acc*	*Apply rule 7.2.5 RSentenceNL:* Input: V(Blezen), NP1(NP(N'(PN(BAnna)))), NP(ART(een) N'(Bboek)) ∨ NP(ART(het) N'(Bboek)) ∨ NP(N'(Bboeken)) ∨ NP(ART(de) N'(Bboeken)) Output (C8 in the rule): S(NP1(NP(N'(PN(BAnna))) V(Bleest) NP2(NP(ART(het)) N'(Bboek)))$_{bonus=1}$ S(NP1(NP(N'(PN(BAnna))) V(Bleest) NP2(NP(N'(Bboeken)))) S(NP1(NP(N'(PN(BAnna))) V(Bleest) NP2(NP(ART(de) N'(Bboeken)))) S(NP1(NP(N'(PN(BAnna))) V(Bleest) NP2(NP(ART(een) N'(Bboek)))) *NP1.case = nom* *NP2.case = acc* *V.number = NP1.number*

	V.person = NP1.person

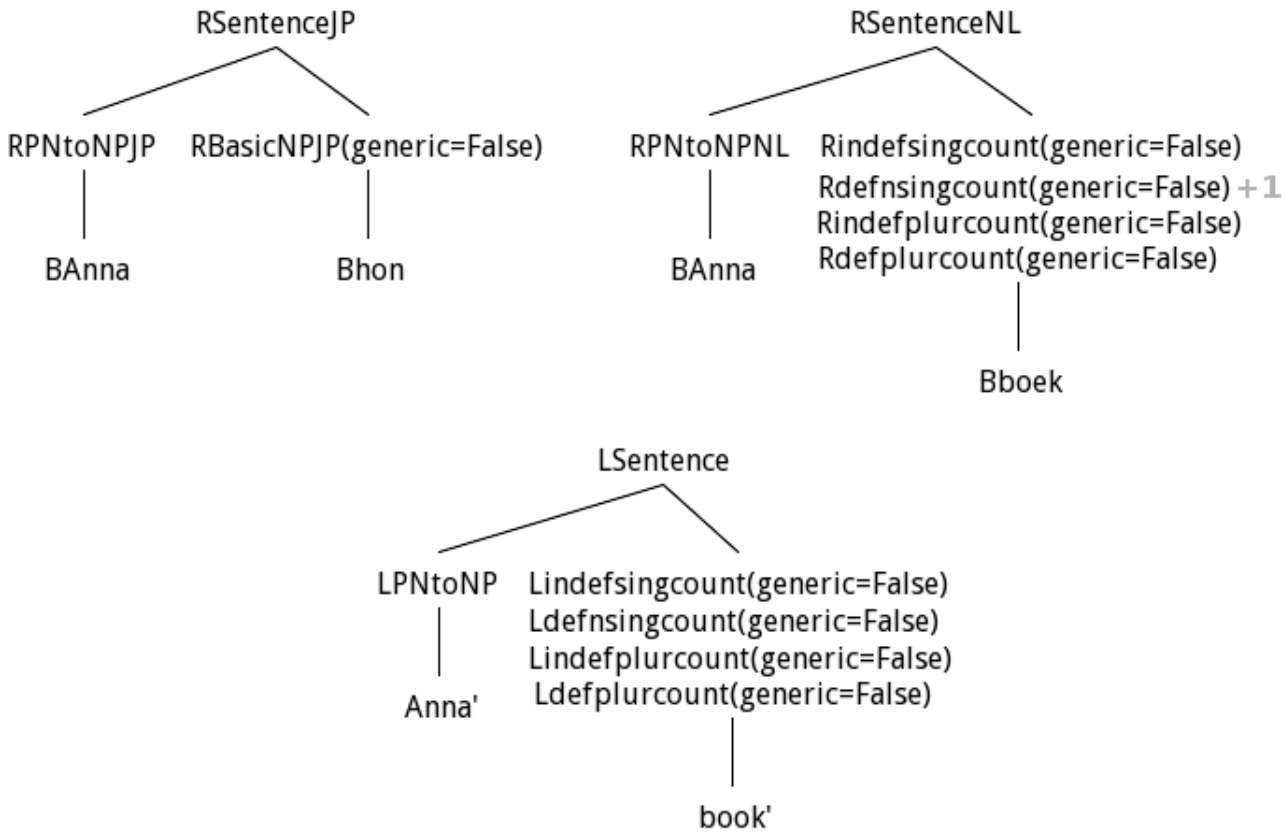


Figure 11: Syntactic derivation trees and the semantic interlingua derivation tree for 'Anna is reading the book'.

7.4.3 'I saw everything.'

The following sentence is from a fiction story by Yoshimoto Banana, 夜と夜の旅人 (*Yoru to yoru no tabibito*, published in English as "Night and night's travelers"). I will use the transcribed Japanese words for better readability.

- Sentence:
  私はすべてを見ていた。[104] (*watashi wa subete wo miteita*).
- Basic expressions from dictionary[33]:

Japanese basic expressions	**Dutch basic expressions**
Bwatashi (PersPrN): - PersPrN.sex = undefined - PersPrN.animate = True	Bik (PersPrN): - PersPrN.sex = undefined - PersPrN.person = 1st - PersPrN.animate = True - PersPrN.case = undefined

[33]To look up the words in the dictionary, the Japanese characters are used.

	- PersPrN.num = sing - PersPrN.gnd = utrum
Bsubete (PrN): - PrN.sex = undefined - PrN.animate = False	Balles (PrN): - PrN.sex = undefined - PrN.person = 3rd - PrN.animate = False - PrN.case = undefined - PrN.num = sing - PrN.def = indef
Bmiru (V)[34]: - V.tense = past - V.objectDivisible = False - V.subjectDivisible = False - V.indirectobjectDivisible = False - V.objectPrefGeneric = False - V.subjectPrefGeneric = False - V.indirectobjectPrefGeneric = False	Bzien (V)[35]: - V.tense = past - V.objectDivisible = False - V.subjectDivisible = False - V.indirectobjectDivisible = False - V.objectPrefGeneric = False - V.subjectPrefGeneric = False - V.indirectobjectPrefGeneric = False - V.number = undefined - V.person = undefined

Assume there is a rule that takes the basic expression V(Bzien) and makes it past singular (*V(Bzag)*) or plural *V(Bzagen)*. It copies all the features of V, and also sets the number feature (sing/plur).
I assume below that this rule has been applied.

- Applying rules:

Japanese rules	**Dutch rules**
Apply rule 7.2.2 RPersPronounJP: Input: PersPrN(Bwatashi) Output: NP(PersPrN(Bwatashi)) - NP.person = 1st - NP.sex = undefined - NP.animate = True	*Apply rule 7.2.2 RPersPronounNL:* Input: PersPrN(Bik) Output: NP(PersPrN(Bik)) - NP.person = 1st & - NP.sex = undefined & - NP.animate = True & - NP.case = undefined & - NP.num = sing & - NP.gnd = utrum & - *NP.def = def* - *NP.canBeBare = False*
Apply rule 7.2.3 RPronounJP: Input: PrN(Bsubete) Output: NP(PrN(Bsubete))	*Apply rule 7.2.3 RPronounNL:* Input: PrN(Balles) Output: NP(PrN(Balles))

[34]Information about verb forms should be present as well.

[35]This is the past tense of the verb 'zien', but I didn't write rules to create past tense. The sentence rule will specify person and number.

- NP.sex = undefined - NP.divisible = False - NP.generic = False - NP.person = 3rd - NP.animate = False	- NP.sex = undefined - NP.divisible = False - NP.generic = False - NP.person = 3rd - NP.animate = False - NP.case = undefined - NP.num = sing - NP.def = def - *NP.canBeBare = False*
Apply rule 7.2.5 RSentenceJP: Input: V(Bmiteita), NP1(NP(PersPrN(Bwatashi))), NP2(NP(PrN(Bsubete))) Output: S(NP1(NP(PersPrN(Bwatashi)))-ga NP2(NP(PrN(Bsubete)))-wo V(V(Bmiteita))) - *NP1.case = nom* - *NP2.case = acc*	*Apply rule 7.2.5 RSentenceNL:* Input: V(Bzag)[36], NP(PersPrN(Bik)), NP(PrN(Balles)) Output: S(NP1(NP(PersPrN(Bik))) V(V(Bzag)) NP2(NP(PrN(Balles)))) - *NP1.case = nom* - *NP2.case = acc*

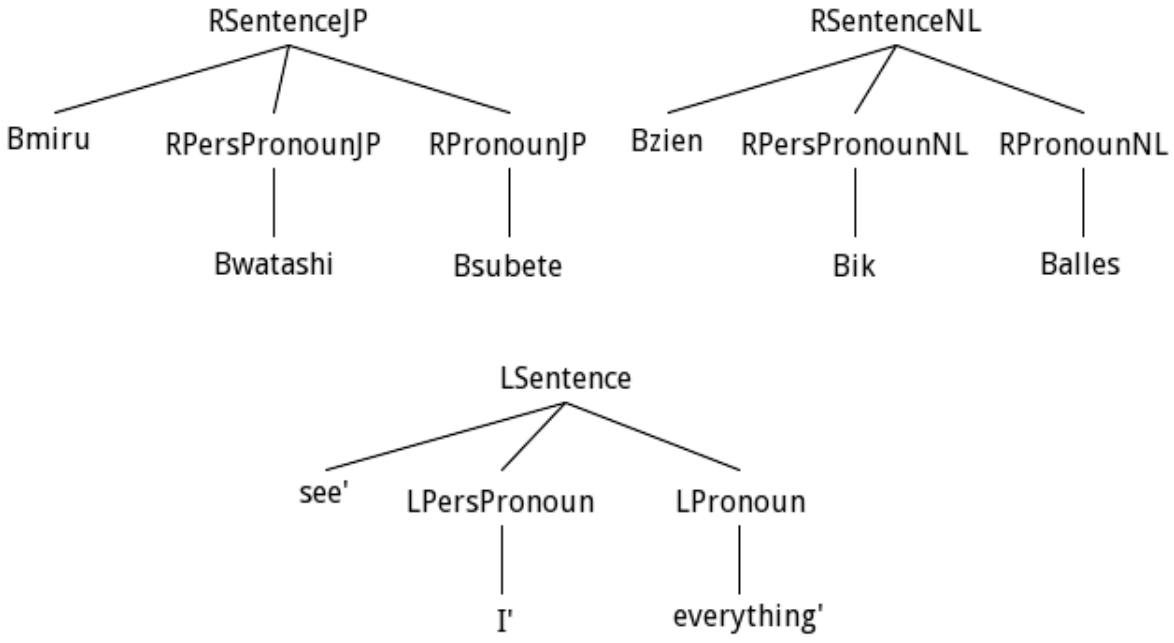


Figure 12: Syntactic derivation trees and the semantic interlingua derivation tree for 'I saw everything'.

[36]As one of the conditions in RSentence is that NP1.num = V.num, 'zagen' is not possible anymore.

7.4.4 'Mama likes cats.'

- Japanese sentence ママは猫が好きだ。 (*Mama wa neko ga suki da*) and possible Dutch translations.

- Basic expressions from dictionary[37]:

Japanese basic expressions	**Dutch basic expressions**
Bmama (PN[38]) - PN.number = sing - PN.sex = female - PN.hon = False - PN.divisible = False - PN.animate = True	Bmama (PN) - PN.number = sing - PN.sex = female - PN.hon = False - PN.divisible = False - PN.animate = True
Bneko (N): - N.sex = undefined - N.divisible = False - N.generic = False - N.animate = True - N.person = 3rd	Bkat (N): - N.sex = undefined - N.divisible = False - N.generic = False - N.animate = True - N.person = 3rd - N.case = undefined - N.number = undefined - N.gender = utrum - N.def = undefined - N.countmass = count - N.canBeBare = False
Bsuki da (V)[39]: - V.tense = present[40] - V.objectDivisible = False - V.subjectDivisible = False - V.indirectobjectDivisible = False - V.objectPrefGeneric = True - V.subjectPrefGeneric = False - V.indirectobjectPrefGeneric = False	Bhouden van (V): - V.tense = present - V.objectDivisible = False - V.subjectDivisible = False - V.indirectobjectDivisible = False - V.objectPrefGeneric = True - V.subjectPrefGeneric = False - V.indirectobjectPrefGeneric = False

The rule 7.2.4 that takes the basic expression N(Bkat) and makes it *N'(Bkat)* singular or *N'(Bkatten)* plural, has been applied. It copies all the features of N, and also sets the number feature (sing/plur).
Similar for the verb *V(Bhouden van)*. I assume below that these rules have been applied.

[37]To look up the words in the dictionary, the Japanese characters are used.
[38]'Mama' can be viewed as a proper noun in both languages, as it's used in the same way as a name.
[39]Information about verb forms should be present as well.
[40]I didn't write any rules to get tense.

- Applying rules:

Japanese rules	**Dutch rules**
Apply rule 7.2.7 RPNtoNPJP: Input: PN(Bmama) Output: NP(PN(Bmama)) - NP.sex = female - NP.hon = False - NP.divisible = False - NP.animate = True	*Apply rule 7.2.7 RPNtoNPNL:* Input: PN(Bmama) Output: NP(PN(Bmama)) - NP.number = sing - NP.sex = female - NP.hon = False - NP.divisible = False - NP.animate = True - *NP.countmass = count* - *NP.gender = utrum*
Apply rule 7.2.1 RBasicNPJP$_{generic=False}$: Input: N'(Bneko) Output: NP(N'(Bneko)) - NP.sex = undefined - NP.divisible = False - NP.generic = False - NP.animate = True - NP.person = 3rd	*Apply rule 7.2.1 RindefsingcountNL$_{generic=False}$:* Input: N'(Bkat) Output: NP(ART(een) N'(Bkat)) - NP.sex = undefined - NP.divisible = False - NP.generic = False - NP.animate = True - NP.person = 3rd - NP.case = undefined - NP.number = sing - NP.gender = neutrum - *NP.def = indef* - NP.countmass = count - NP.canBeBare = False *Apply rule 7.2.1 RdefusingcountNL$_{generic=False}$:* Input: N'(Bkat) Output: NP(ART(de) N'(Bkat)) - NP.sex = undefined - NP.divisible = False - NP.generic = False - NP.animate = True - NP.person = 3rd - NP.case = undefined - NP.number = sing - NP.gender = utrum - *NP.def = def* - NP.countmass = count - NP.canBeBare = False *Apply rule 7.2.1 RindefplurcountNL$_{generic=False}$:* Input: N'(Bkatten) Output: NP(N'(Bkatten)) - NP.sex = undefined - NP.divisible = True

	- NP.generic = False - NP.animate = True - NP.person = 3rd - NP.case = undefined - NP.number = plur - NP.gender = utrum - *NP.def = def* - NP.countmass = count - NP.canBeBare = False *Apply rule 7.2.1 RdefplurcountNL$_{generic=False}$:* Input: N'(Bkatten) Output: NP(ART(de) N'(Bkatten)) - NP.sex = undefined - NP.divisible = True - NP.generic = False - NP.animate = True - NP.person = 3rd - NP.case = undefined - NP.number = plur - NP.gender = utrum - *NP.def = def* - NP.countmass = count - NP.canBeBare = False
Apply rule 7.2.1 RBasicNPJP$_{generic=True}$: Input: N'(Bneko) Output: NP(N'(Bneko)) - NP.sex = undefined - NP.divisible = False - NP.generic = True - NP.animate = True - NP.person = 3rd	*Apply rule 7.2.1 RindefsingcountNL$_{generic=True}$:* Input: N'(Bkat) Output: NP(ART(een) N'(Bkat)) - NP.sex = undefined - NP.divisible = False - NP.generic = True - NP.animate = True - NP.person = 3rd - NP.case = undefined - NP.number = sing - NP.gender = neutrum - *NP.def = indef* - NP.countmass = count - NP.canBeBare = False *Apply rule 7.2.1 RdefusingcountNL$_{generic=True}$:* Input: N'(Bkat) Output: NP(ART(de) N'(Bkat)) - NP.sex = undefined - NP.divisible = False - NP.generic = True - NP.animate = True - NP.person = 3rd - NP.case = undefined

	- NP.number = sing - NP.gender = utrum - *NP.def = def* - NP.countmass = count - NP.canBeBare = False *Apply rule 7.2.1 $RindefplurcountNL_{generic=True}$:* Input: N'(Bkatten) Output: NP(N'(Bkatten)) - NP.sex = undefined - NP.divisible = True - NP.generic = True - NP.animate = True - NP.person = 3rd - NP.case = undefined - NP.number = plur - NP.gender = utrum - *NP.def = def* - NP.countmass = count - NP.canBeBare = False *Apply rule 7.2.1 $RdefplurcountNL_{generic=True}$:* Input: N'(Bkatten) Output: NP(ART(de) N'(Bkatten)) - NP.sex = undefined - NP.divisible = True - NP.generic = True - NP.animate = True - NP.person = 3rd - NP.case = undefined - NP.number = plur - NP.gender = utrum - *NP.def = def* - NP.countmass = count - NP.canBeBare = False
Apply rule 7.2.5 RSentenceJP: Input: NP(PN(Bmama)), NP(N'(Bneko)), V(Bsuki da) Output: S(NP(PN(Bmama))-wa NP(N'(Bneko))-ga V(Bsuki da)) Output: S(NP(PN(Bmama))-wa NP(N'(Bneko))-ga V(Bsuki da))	*Apply rule 7.2.5 RSentenceNL:* Input: NP1(PN(Bmama)) NP2(N'(Bkatten)) V(Bhouden van) Output ($C22_{bonus+=1}$) S(NP1(PN(Bmama)) V(Bhoudt van) NP2(N'(Bkatten)) Output (C10a) S(NP1(PN(Bmama)) V(Bhoudt van) NP2(N'(de Bkat))

Output: S(NP(PN(Bmama))-wa NP(N'(Bneko))-ga V(Bsuki da))	Output (C10b) S(NP1(PN(Bmama)) V(Bhoudt van) NP2(N'(de Bkatten))

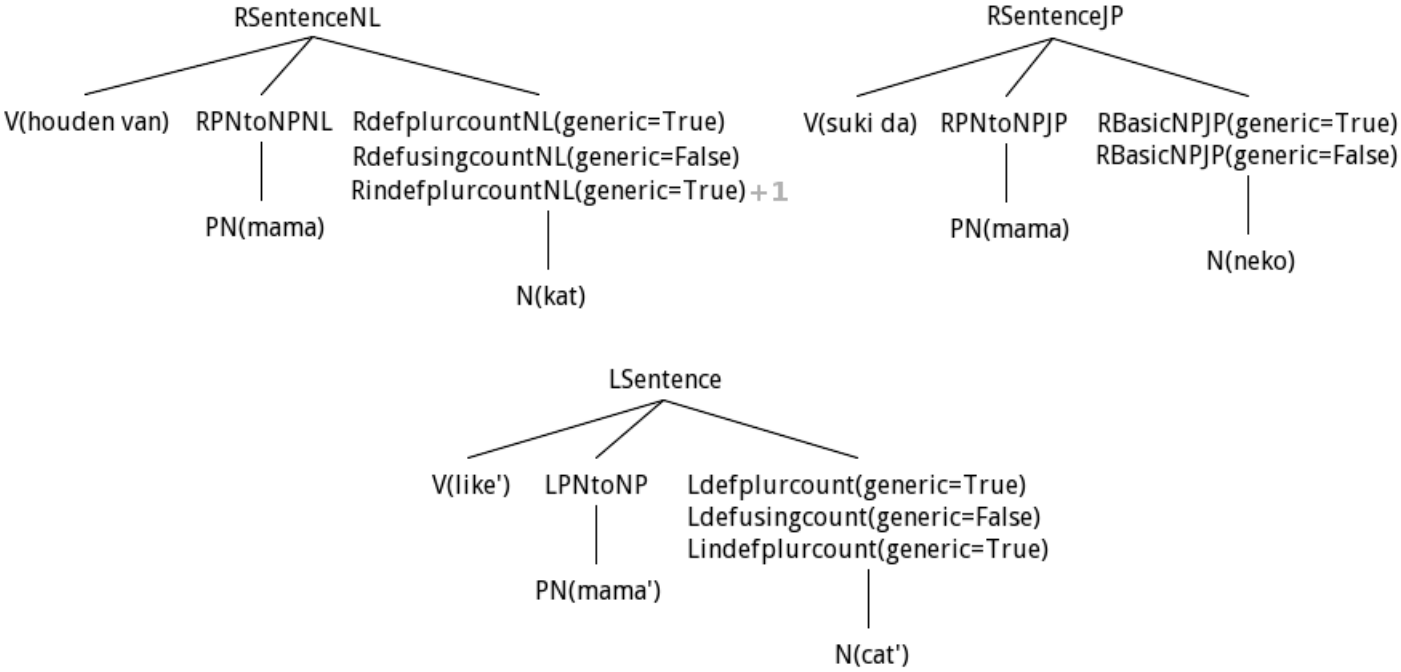


Figure 13: Syntactic derivation trees and the semantic interlingua derivation tree for 'Mama likes cats'.

7.4.5 'Elephants eat leaves.'

- Japanese sentence 像は葉っぱと言うものを食べる。 (*zou-wa happa-to-iu-mono-wo taberu*) and possible Dutch translations. This is one of the example sentences for C20 in 7.2.5.

- Basic expressions from dictionary[41]:

Japanese basic expressions	**Dutch basic expressions**
Bzou (N): - N.sex = undefined - N.divisible = False - N.generic = False - N.animate = True - N.person = 3rd	Bolifant (N): - N.sex = undefined - N.divisible = False - N.generic = False - N.animate = True - N.person = 3rd - N.case = undefined - N.number = undefined

[41]To look up the words in the dictionary, the Japanese characters are used.

	- N.gender = utrum - N.def = undefined - N.countmass = count - N.canBeBare = False
Bhappa (N): - N.sex = undefined - N.divisible = False - N.generic = False - N.animate = False - N.person = 3rd	Bblad (N): - N.sex = undefined - N.divisible = False - N.generic = False - N.animate = False - N.person = 3rd - N.case = undefined - N.number = undefined - N.gender = neutrum - N.def = undefined - N.countmass = count - N.canBeBare = False
Btaberu (V)[42]: - V.tense = present[43] - V.objectDivisible = False - V.subjectDivisible = False - V.indirectobjectDivisible = False - V.objectPrefGeneric = True - V.subjectPrefGeneric = False - V.indirectobjectPrefGeneric = False	Beten (V): - V.tense = present - V.objectDivisible = False - V.subjectDivisible = False - V.indirectobjectDivisible = False - V.objectPrefGeneric = True - V.subjectPrefGeneric = False - V.indirectobjectPrefGeneric = False

The rule 7.2.4 that takes the basic expression N(Bolifant) and makes it *N'(Bolifant)* singular or *N'(Bolifanten)* plural, has been applied. It copies all the features of N, and also sets the number feature (sing/plur).
Similar for the noun *N(Bblad))* and the verb *V(Beten)*. I assume below that these rules have been applied.

- Applying rules:

Japanese rules	**Dutch rules**
Apply rule 7.2.1 RBasicNPJP$_{generic=False}$: Input: N'(Bzou) Output: NP(N'(Bzou)) - NP.sex = undefined - NP.divisible = False - NP.generic = False - NP.animate = True - NP.person = 3rd	*Apply rule 7.2.1 RindefsingcountNL$_{generic=False}$:* Input: N'(Bolifant) Output: NP(ART(een) N'(Bolifant)) - NP.sex = undefined - NP.divisible = False - NP.generic = False - NP.animate = True - NP.person = 3rd - NP.case = undefined

[42]Information about verb forms should be present as well.
[43]I didn't write any rules to get tense.

		- NP.number = sing - NP.gender = neutrum - *NP.def = indef* - NP.countmass = count - NP.canBeBare = False *Apply rule 7.2.1 $RdefusingcountNL_{generic=False}$:* Input: N'(Bolifant) Output: NP(ART(de) N'(Bolifant)) - NP.sex = undefined - NP.divisible = False - NP.generic = False - NP.animate = True - NP.person = 3rd - NP.case = undefined - NP.number = sing - NP.gender = utrum - *NP.def = def* - NP.countmass = count - NP.canBeBare = False *Apply rule 7.2.1 $RindefplurcountNL_{generic=False}$:* Input: N'(Bolifanten) Output: NP(N'(Bolifanten)) - NP.sex = undefined - NP.divisible = True - NP.generic = False - NP.animate = True - NP.person = 3rd - NP.case = undefined - NP.number = plur - NP.gender = utrum - *NP.def = def* - NP.countmass = count - NP.canBeBare = False *Apply rule 7.2.1 $RdefplurcountNL_{generic=False}$:* Input: N'(Bolifanten) Output: NP(ART(de) N'(Bolifanten)) - NP.sex = undefined - NP.divisible = True - NP.generic = False - NP.animate = True - NP.person = 3rd - NP.case = undefined - NP.number = plur - NP.gender = utrum - *NP.def = def* - NP.countmass = count

	- NP.canBeBare = False
Apply rule 7.2.1 RBasicNPJP$_{generic=True}$: Input: N'(Bzou) Output: NP(N'(Bzou)) - NP.sex = undefined - NP.divisible = False - NP.generic = True - NP.animate = True - NP.person = 3rd	*Apply rule 7.2.1 RindefsingcountNL$_{generic=True}$:* Input: N'(Bolifant) Output: NP(ART(een) N'(Bolifant)) - NP.sex = undefined - NP.divisible = False - NP.generic = True - NP.animate = True - NP.person = 3rd - NP.case = undefined - NP.number = sing - NP.gender = neutrum - *NP.def = indef* - NP.countmass = count - NP.canBeBare = False *Apply rule 7.2.1 RdefusingcountNL$_{generic=True}$:* Input: N'(Bolifant) Output: NP(ART(de) N'(Bolifant)) - NP.sex = undefined - NP.divisible = False - NP.generic = True - NP.animate = True - NP.person = 3rd - NP.case = undefined - NP.number = sing - NP.gender = utrum - *NP.def = def* - NP.countmass = count - NP.canBeBare = False *Apply rule 7.2.1 RindefplurcountNL$_{generic=True}$:* Input: N'(Bolifanten) Output: NP(N'(Bolifanten)) - NP.sex = undefined - NP.divisible = True - NP.generic = True - NP.animate = True - NP.person = 3rd - NP.case = undefined - NP.number = plur - NP.gender = utrum - *NP.def = def* - NP.countmass = count - NP.canBeBare = False *Apply rule 7.2.1 RdefplurcountNL$_{generic=True}$:* Input: N'(Bolifanten) Output: NP(ART(de) N'(Bolifanten))

	- NP.sex = undefined - NP.divisible = True - NP.generic = True - NP.animate = True - NP.person = 3rd - NP.case = undefined - NP.number = plur - NP.gender = utrum - *NP.def = def* - NP.countmass = count - NP.canBeBare = False
Apply rule 7.2.1 RBasicNPJP$_{generic=False}$: Input: N'(Bhappa) Output: NP(N'(Bhappa)) - NP.sex = undefined - NP.divisible = False - NP.generic = False - NP.animate = False - NP.person = 3rd	*Apply rule 7.2.1 RindefsingcountNL$_{generic=False}$:* Input: N'(Bblad) Output: NP(ART(een) N'(Bblad)) - NP.sex = undefined - NP.divisible = False - NP.generic = False - NP.animate = False - NP.person = 3rd - NP.case = undefined - NP.number = sing - NP.gender = neutrum - *NP.def = indef* - NP.countmass = count - NP.canBeBare = False *Apply rule 7.2.1 RdefnsingcountNL$_{generic=False}$:* Input: N'(Bblad) Output: NP(ART(het) N'(Bblad)) - NP.sex = undefined - NP.divisible = False - NP.generic = False - NP.animate = False - NP.person = 3rd - NP.case = undefined - NP.number = sing - NP.gender = neutrum - *NP.def = def* - NP.countmass = count - NP.canBeBare = False *Apply rule 7.2.1 RindefplurcountNL$_{generic=False}$:* Input: N'(Bbladeren) Output: NP(N'(Bbladeren)) - NP.sex = undefined - NP.divisible = True - NP.generic = False - NP.animate = False

	- NP.person = 3rd - NP.case = undefined - NP.number = plur - NP.gender = neutrum - *NP.def = def* - NP.countmass = count - NP.canBeBare = False *Apply rule 7.2.1 $RdefplurcountNL_{generic=False}$:* Input: N'(Bbladeren) Output: NP(ART(de) N'(Bbladeren)) - NP.sex = undefined - NP.divisible = True - NP.generic = False - NP.animate = False - NP.person = 3rd - NP.case = undefined - NP.number = plur - NP.gender = neutrum - *NP.def = def* - NP.countmass = count - NP.canBeBare = False
Apply rule 7.2.1 $RBasicNPJP_{generic=True}$: Input: N'(Bhappa) Output: NP(N'(Bhappa)) - NP.sex = undefined - NP.divisible = False - NP.generic = True - NP.animate = False - NP.person = 3rd	*Apply rule 7.2.1 $RindefsingcountNL_{generic=True}$:* Input: N'(Bblad) Output: NP(ART(een) N'(Bblad)) - NP.sex = undefined - NP.divisible = False - NP.generic = True - NP.animate = False - NP.person = 3rd - NP.case = undefined - NP.number = sing - NP.gender = neutrum - *NP.def = indef* - NP.countmass = count - NP.canBeBare = False *Apply rule 7.2.1 $RdefnsingcountNL_{generic=True}$:* Input: N'(Bblad) Output: NP(ART(het) N'(Bblad)) - NP.sex = undefined - NP.divisible = False - NP.generic = True - NP.animate = False - NP.person = 3rd - NP.case = undefined - NP.number = sing - NP.gender = neutrum

Apply rule 7.2.6 RGenericNPtoiuJP$_{generic=True}$ Input: N'(Bhappa) Output: NP(N'(Bhappa)-to-iu-mono)	- *NP.def = def* - NP.countmass = count - NP.canBeBare = False *Apply rule 7.2.1 RindefplurcountNL*$_{generic=True}$*:* Input: N'(Bbladeren) Output: NP(N'(Bbladeren)) - NP.sex = undefined - NP.divisible = True - NP.generic = True - NP.animate = False - NP.person = 3rd - NP.case = undefined - NP.number = plur - NP.gender = neutrum - *NP.def = def* - NP.countmass = count - NP.canBeBare = False *Apply rule 7.2.1 RdefplurcountNL*$_{generic=True}$*:* Input: N'(Bbladeren) Output: NP(ART(de) N'(Bbladeren)) - NP.sex = undefined - NP.divisible = True - NP.generic = True - NP.animate = False - NP.person = 3rd - NP.case = undefined - NP.number = plur - NP.gender = neutrum - *NP.def = def* - NP.countmass = count - NP.canBeBare = False
Apply rule 7.2.5 RSentenceJP: Input: NP(N'(Bzou)), NP(N'(Bzou)), NP(N'(Bzou)), NP(N'(Bzou)), NP(N'(Bhappa)), NP(N'(Bhappa)), NP(N'(Bhappa)), NP(N'(Bhappa)), **NP(N'(Bhappa)-to-iu-mono)** V(Btaberu) Output: S(NP1(NP(N'(Bzou)))-ga NP2(NP(N'(Bhappa)-to-iu-mono))-wo	*Apply rule 7.2.5 RSentenceNL:* Input: NP(ART(een) N'(Bolifant)) NP(ART(de) N'(Bolifant))$_{C8generic=F:bonus+=1}$ NP(N'(Bolifanten))$_{C16generic=T:bonus+=1}$ NP(ART(de) N'(Bolifanten)) NP(ART(een) N'(Bblad))$_{C9generic=F:bonus+=1}$ NP(ART(het) N'(Bblad)) NP(N'(Bbladeren))$_{C22ageneric=T:bonus+=1}$ NP(ART(de) N'(Bbladeren))$_{C20ageneric=T:bonus+=1}$ **NP(N'(Bbladeren))**$_{C20a+C22ageneric=T:bonus+=2}$ V(Beten) Output (bonus = 3): S(NP1(NP(ART(de) N'(Bolifant))) V(Beten)

V(Btaberu)) Output: S(NP1(NP(N'(Bzou)))-ga NP2(NP(N'(Bhappa)-to-iu-mono))-wo V(Btaberu))	NP2(NP(N'(Bbladeren)) *De olifant eet bladeren.* Output (bonus = 3): S(NP1(NP(N'(Bolifanten))) V(Beten) NP2(NP(N'(Bbladeren)) *Olifanten eten bladeren* There are other derivations with less bonus points, which get ranked lower.

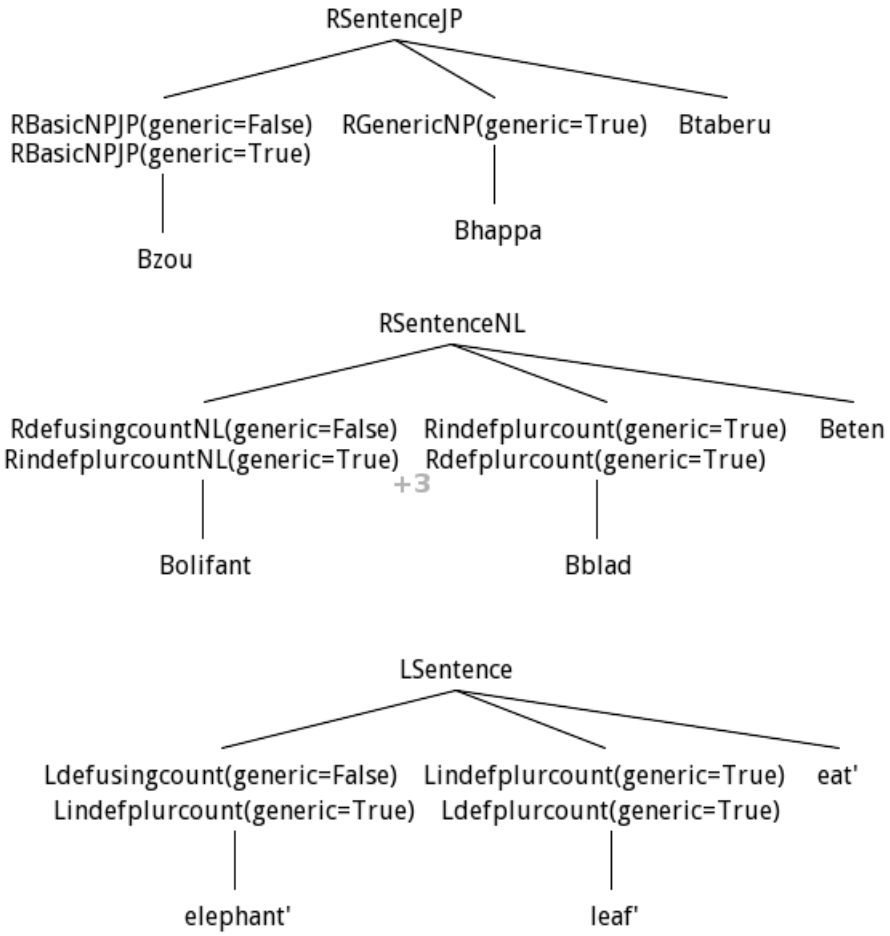


Figure 14: Syntactic derivation trees and the semantic interlingua derivation tree for 'Elephants eat leaves'. Only the two derivations with the highest bonus (3) are included.

7.5 Evaluation

Evaluating the rules in a working system requires a working system with an implemented lexicon/dictionary and implemented rule set, for both languages. It requires the dictionary to be set up in a structure that includes features and values. A reference corpus (a corpus containing real texts) is then necessary for testing. However, the rules above only describe a part of the phenomena that appear in a real sentence. So, the test suite for these rules should focus on the described phenomena.

In section 4.3.1, I showed an example from JMdict. There isn't any existing Japanese-Dutch dictionary to use, as the Dutch translations in JMdict are very minimal and of a non-useful format. However, XML is useful for a formal representation of the lexicon, as it has a tree structure (useful for adding features to a word) and the option to add extra information to the tags (like the language of certain information). Another markup language based on XML could also be used, for example LMF (Lexical Markup Framework) or TEI (Text Encoding Initiative).

When the input is an existing sentence, there should also be a parser which divides the sentence into separate words, which can be checked in the dictionary for part-of-speech. However, existing sentences contain word conjugations, so these should be recognized as well [44].

For a human reader, a translated sentence looks better if the conjugations are correct (for example 'eats' instead of just 'eat'), so extra rules to implement conjugations for the output would also need to be added.

A test suite, a set containing sentences that cover all relevant cases, can be used to test the translation quality of the system.

As an example, the NTT (Nippon Telegraph and Telephone Corporation) Linguistic Intelligence Research Group has a machine translation test set online from 1998 [30]. In this test set, the difficulty (and length) of the sentences increases as the sentence numbers get higher. One of the test sentences from this set can be seen in 46-47. There are quite a few sentences which appear multiple times in the set, because of the multiple ways you can write a word in Japanese. In (46) the kanji-character 釦 for 'button' is used, while in (47) the katakana-characters ボタン are used. In the dictionary entries above, such differences are listed in the 'reading_elements' tags.

(46) 01070 彼女は釦を付ける。
She sews on buttons.
———
She sews on a button.

(47) 01080 彼女はボタンを付ける。
She sews on buttons.
———
She sews on a button.

However, not all sentences have the singular/plural distinction in their English translation. For example, '私は林檎を食べる。' only has 'I eat apples.' as a translation. This would be the preferred translation, that would get the highest bonus in a compositional translation system. However, there are some reasons why this test set cannot be used as-is. As can be seen in (48), alternative translations are listed in brackets. In some cases, like (49), there is an alternative translation between brackets with 'OR', while the alternative translation in (48) doesn't include an 'OR'. There also are a lot of comments in Japanese about the alternative translations in (49), of which one is between brackets

[44] I wrote more on parsing and the dictionary in section 4.3.1.

above the line, but written in two separate sentences below the line. So, this test suite is not formal to be used immediately for testing without any editing.

(48) 09430 彼は口が利けない。
He cannot (is unable to) speak.

———

He cannot speak.

(49) 24260 彼は人の前に並んだ。
He stood in front of the people (OR, in front of everybody). (彼は人（複数）の前に立った)
(He lined up in front of a person. He stood in front of a person. 何れもおかしいい。というのも日本語原文がおかしいのであって、並ぶといえば複数のものが列をなして立つことをいい、一人がまたは一個の物体が並ぶといわない。)

———

He lined up in front of a person.
He stood in front of a person.

The example in (50) is of a sentence which can be translated as a fixed expression (the first translation) or literally (the second translation). These are very different, but fixed expressions would be in the dictionary in their entirety. If you only have a dictionary with separate words, you will only be able to get a more literal translation.

(50) 09440 彼女は口を拭った。
She feigned innocence.
She wiped her mouth.

So, while there are useful sentences in this test set, there are still things that require editing or more sophisticated parsing to prepare it for actual use.

And of course, if you want to use this test set with another language than English, it has to be edited manually to add that language as well.

8 Conclusion

In this study, I tried to find a way to improve the quality of machine translation with respect to one specific area: number and definiteness of noun phrases, when translating between a language that does differentiate between number and definiteness (Dutch) and a language that doesn't have these differences in its word forms or phrases (Japanese). The question I asked in the beginning was: "When do you have to insert a definite, indefinite or no article and when do you have to choose a singular or plural form for nouns when translating from Japanese to Dutch in the context of syntax-based machine translation?"

After studying the literature to find out what other people had already written about this problem, I discovered that no-one has really written about this translation problem between Dutch and Japanese (only between English and Japanese). As there are differences between English and Dutch, it is not possible to copy all findings from the English-Japanese translation literature exactly to Dutch. For this reason I also looked at the literature on Dutch numbers and determiners.

I approached the question using the theoretical 'compositional translation' framework, based on the Rosetta framework used by Philips in the late 1980s. This is a machine translation framework in which a set of basic expressions and syntactic rules is specified for each language. It uses synchronized (isomorphic) grammars to deal with translation. I made an analysis of the translation problem in the spirit of this translation framework and described a concrete grammar fragment to show how the approach can be made to work.

The notation of rules is based on the Rosetta system, but it does differ from the simplified notation used in the Rosetta book [86], because I needed to use a notation which made the different conditions and features more clear. Though without examples, it's not that clear for a human reader to quickly see what kind of sentences result from a specific rule.

The main advantage of this framework is that the grammars are reversible, so you don't need separate grammars for generation and analysis of both languages (for translating from one language to another and the other way around). These reversible grammars create sentences that are each other's translations. This framework intentionally derives all possible translations, and a bonus system is used to rank them into a preferred order.

Writing grammar fragments for natural language requires sophisticated grammars, with powerful operations, and syntactic categories with *features* to express properties of words and phrases. To achieve a better translation, attaching features to the words and phrases is most important, because then you can write/use rules that add and use these features to decide on number and definiteness of the nouns. When you have a finite amount of features, each with a finite amount of values, it would theoretically be possible to write a grammar without using features. However, then you would not have the option to make generalizations the way you can do when using features, and the grammar rules would become very complex and impossible for a human to manage, as there will be too many rules for a human to calculate (it would be possible to calculate for a computer, but to enter these rules into a computer, human input is necessary).

However, when you have features which could have an infinite amount of values, it is necessary to use features (you can't do without them). There is one feature in the system I described where this is the case: the topicalization rule 7.2.15 which makes use of an index. This index could have an infinite amount of values. Using such a rule with an index wouldn't be possible if you don't use features.

The rules check for certain settings of the features to decide whether or not they are applicable. If necessary, features of the resulting output expressions are set as well.

When it is possible to make a choice based on grammatical or semantic properties, all conditions for a specific rule should be met before it can be applied. These restrictions are intended to prevent ungrammatical combinations from being generated. However, it is still possible that multiple rules can be applied. Then the output consists of all possible translations, as the compositional translation system generates all possible output sentences. However, a bonus system is used to order the possible translations by plausibility and/or preference.

The most important features for deciding on number and definiteness are 'divisible' (is the noun divisible, so does it consist of separate entities (+SET) and/or is it a mass noun?), 'generic' (is it used as a generic noun or not?) and 'countmass' (is it a count noun or a mass noun?). The features 'divisible' and 'generic' are semantic properties, which have the largest influence, but the compositional translation framework does not have an explicit semantic component. Because of this, all semantic restrictions have to be included in the syntax, which makes the syntax a lot more complicated. So, adding an explicit semantic component for including semantic restrictions on combining certain phrases, would be more efficient and manageable.

I did look exclusively at single sentences, as the Rosetta system and most other machine translation systems also work on single sentences. However, a full treatment of the phenomena requires taking the preceding context into account as well.

To be able to use this system, a dictionary with a rich set of features attached to the words is required. Setting this up still requires a lot of manual work, as there are no existing systems for translating between Dutch and Japanese in which I could have tested these rules. Testing it in the Rosetta framework isn't possible anymore either, because the implemented version is not running anymore [45].

To improve the translation quality, you can use syntactical rules. However, to order the results you get with these rules (as there is often more than one possible output), you can use statistics (I used a simple 'bonus' mechanism in the rules, but this could be extended to refine the result ordering).

Future work can include building a dictionary and translation program using these rules, extending them to include other details and part-of-speech classes, and including information from preceding sentences (context) as well.

[45] Personal comment from Jan Odijk, who has worked on this system at Philips.

References

[1] Namiko Abe. Personal pronouns - Japanese language, June 2013. http://japanese.about.com/od/Grammar/a/Personal-Pronouns.htm.

[2] Anne Abeillé, Yves Schabes, and Aravind K. Joshi. Using lexicalized tags for machine translation. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING'90*, 1990.

[3] Amr Ahmed and Greg Hanneman. Syntax-based statistical machine translation: A review. *Association for Computational Linguistics*, 2005.

[4] Erica Asai. Orandago – kanshi, February 2012. http://easai.web.fc2.com/linguistics/Dutch/537/.

[5] Timothy Baldwin and Francis Bond. Learning the countability of english nouns from corpus data. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 463–470, July 2003.

[6] Francis Bond. *Translating the untranslatable: A solution to the problem of generating English determiners*. CSLI Publications, Stanford, California, 2005.

[7] Francis Bond and Satoru Ikehara. When and how to disambiguate? countability in machine translation. In *MIDDIM-96 Seminar (the International Seminar on Multimodal Interactive Disambiguation)*, Le Col de Porte, France, August 1996.

[8] Francis Bond, Kentaro Ogura, and Satoru Ikehara. Countability and number in Japanese to English machine translation. In *COLING*, pages 32–38, 1994.

[9] Francis Bond, Kentaro Ogura, and Satoru Ikehara. Classifiers in Japanese-to-English machine translation. In *COLING*, pages 125–130, 1996.

[10] Francis Bond, Kentaro Ogura, and Tsukasa Kawaoka. Noun phrase reference in Japanese-to-English machine translation. In *Proceedings of the Sixth International Conference on Theoretical and Methodological issues in machine translation*, pages 1–14, 1995.

[11] Jim Breen. The JMdict project, July 2013. http://www.csse.monash.edu.au/~jwb/jmdict.html.

[12] Jim Breen. Jmdict.gz - the full JMdict file, including English, German, French, Russian and Dutch glosses, July 2013. http://ftp.monash.edu.au/pub/nihongo/JMdict.gz.

[13] Jaime Carbonell, Elaine Rich, David Johnson, Masaru Tomita, Muriel Vasconcellos, and Yorick Wilks. JTEC panel report on machine translation in Japan, January 1992.

[14] E. Carrera and H. Flake. Automated structural classification of malware. http://www.sourceconference.com/publications/bos08pubs/carrera-AutomatedStructuralMalwareClassification.pdf.

[15] Paisarn Charoenpornsawat, Virach Sornlertlamvanich, and Thatsanee Charoenporn. Improving translation quality of rule-based machine translation. In *Proceedings of the 2002 COLING workshop on Machine translation in Asia - Volume 16*, COLING-MTIA '02, pages 1–6, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[16] David Chiang. An introduction to synchronous grammars, June 2006.

[17] Pinker/Chomsky Q&A from MIT150 Panel: The Golden Age - A look at the original roots of artificial intelligence, cognitive science, and neuroscience, 2011. `http://languagelog.ldc.upenn.edu/myl/PinkerChomskyMIT.html`.

[18] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, Massachusetts, USA, 2nd edition, 2001.

[19] Steve DeNeefe and Kevin Knight. Synchronous tree adjoining machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 727–736, Singapore, August 2009. ACL and AFNLP.

[20] Bonnie J. Dorr, Eduard Hovy, and Lori Levin. Machine translation: Interlingual methods. In Keith Brown, editor, *Encyclopedia of Language and Linguistics 2nd edition*, pages 383–394. Elsevier Science, 2004.

[21] Pamela Downing. The anaphoric use of classifiers in japanese. In Colette G. Craig, editor, *Noun Classes and Categorization: Proceedings of a symposium on categorization and noun classification, Eugene, Oregon*, pages 345–376. John Benjamins Publishing Company, 1986.

[22] Janet Dean Fodor. Comprehending sentence structure. In D.N. Osherson, editor, *An Invitation to Cognitive Science: Language*, chapter 8, pages 209–246. MIT Press, Cambridge, MA, USA, 1995.

[23] Janet Dean Fodor and Atsu Inoue. Information-paced parsing of japanese. In Reiko Mazuka and Noriko Nagai, editors, *Japanese Sentence Processing*, chapter 2. Lawrence Erlbaum Associates, Inc., Publishers, Hillsdale, New Jersey, USA, 1995.

[24] Victoria Fromkin, Robert Rodman, and Nina Hyams. *An introduction to language*. Wadsworth, Cengage Learning, Boston, MA, USA, 9th edition, 2011.

[25] Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. What's in a translation rule? In *Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, 2004.

[26] Ulrich Germann. Syntax-aware phrase-based statistical machine translation: system description. In *Proceedings of the 7th workshop on statistical machine translation*, pages 292–297, Montréal, Canada, June 2012. Association for Computational Linguistics. `http://statmt.org/wmt12/pdf/WMT35.pdf`.

[27] Google Translate. http://translate.google.com/.

[28] Inside Google Translate, 2013. http://translate.google.com/about/.

[29] Jonathan Graehl and Kevin Knight. Training tree transducers. In *HLT-NAACL*, pages 105–112, Boston, Massachusetts, May 2004.

[30] NTT Linguistic Intelligence Research Group. Machine translation test set (version 2), September 1998. http://www.kecl.ntt.co.jp/icl/lirg/resources/mt-test-set-1.txt.

[31] Takao Gunji and Kôiti Hasida. Measurement and quantification. In Takao Gunji and Kôiti Hasida, editors, *Topics in Constraint-Based Grammar of Japanese*, Studies in Linguistics and Philosophy, chapter 3, pages 39–79. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.

[32] Michael Hammond. Introduction to the mathematics of language, April 2006. http://dingo.sbs.arizona.edu/~hammond/ling178-sp06/.

[33] Yasunari Harada. On the distinction between complement and adjunct in japanese. In *The sixth Japanese-Korean joint conference on formal linguistics*, pages 35–48, 1991.

[34] Bryant Huang and Kevin Knight. Relabeling syntax trees to improve syntax-based machine translation quality. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL*, pages 240–247, New York, USA, June 2006. Association for Computational Linguistics.

[35] John Hutchins. Machine translation: History. In Keith Brown, editor, *Encyclopedia of Language and Linguistics 2nd edition*, pages 375–383. Elsevier Science, 2004.

[36] W. John Hutchins and Harold L. Somers. *An introduction to machine translation*. Academic Press, London, UK, 1992. http://www.hutchinsweb.me.uk/IntroMT-TOC.htm.

[37] Johan Jeuring and Doaitse Swierstra. Languages and compilers (talen en compilers), November 2011.

[38] Stig Johansson and Knut Hofland. The tagged LOB corpus: Description and analyses. In Willem Meijs, editor, *Corpus Linguistics and Beyond: Proceedings of the Seventh International Conference on English Language Research on Computerized Corpora*, volume 59, Amsterdam, The Netherlands, 1987. Editions Rodopi B.V.

[39] Aravind K. Joshi and Yves Schabes. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 69–123. Springer-Verlag, New York, NY, USA, 1997.

[40] Hiroyuki Kaji. An efficient execution method for rule-based machine translation. In *Proceedings of the 12th COLING*, pages 824–829, Budapest, Hungary, 1988. http://aclweb.org/anthology//C/C88/C88-2167.pdf.

[41] Yuki Kamide. Incrementality in japanese sentence processing. In Mineharu Nakayama, Reiko Mazuka, Yasuhiro Shirai, and Ping Li, editors, *The Handbook of East Asian Psycholinguistics*, volume II: Japanese, chapter 33, pages 249–256. Cambridge University Press, 2006.

[42] E. Karim, A. Walenstein, A. Lakhotia, and L. Parida. Malware phylogeny generation using permutations of code. *Journal in Computer Virology*, 1(1-2):13–23, November 2005. http://www.cacs.louisiana.edu/~walenste/pubs/2005-jicv-karim-walenstein-lakhotia-parida.pdf.

[43] M. Khalilov and J.A.R. Fonollosa. N-gram-based statistical machine translation versus syntax augmented machine translation: comparison and system combination. In *Proceedings of the 12th Conference of the European Chapter of the ACL*, pages 424–432, Athens, Greece, March-April 2009. Association for Computational Linguistics. http://aclweb.org/anthology-new/E/E09/E09-1049.pdf.

[44] Kevin Knight and Ishwar Chander. Automated postediting of documents. In *Proceedings of the 12th National Conference on Artificial Intelligence: AAAI-94*, pages 779–784, Seattle, USA, 1994.

[45] Kevin Knight and Jonathan Graehl. An overview of probabilistic tree transducers for natural language processing. In *Proceedings of the Sixth International Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*, Lecture notes in computer science. Springer Verlag, 2005.

[46] P. Koehn. What is a better translation? reflections on six years of running evaluation campaigns. http://homepages.inf.ed.ac.uk/pkoehn/publications/tralogy11.pdf.

[47] Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[48] H. Koiso. On classifiers. Master's thesis, Department of Literature, Chiba University, 1994.

[49] J.G. Kooij. *Aspekten van woordvolgorde in het Nederlands*. Publikaties van de Vakgroep Nederlandse Taal- en Letterkunde, Leiden, The Netherlands, 1978. http://www.dbnl.org/tekst/kooi003aspe01_01/kooi003aspe01_01_0005.php.

[50] Jan Koster. The word orders of english and dutch: Collective vs. individual checking. *Groninger Arbeiten zur germanistischen Linguistik*, 43:1–42, 1999.

[51] D. Lin, K. Church, H. Ji, S. Sekine, D. Yarowsky, S. Bergsma, K. Patil, E. Pitler, R. Lathbury, V. Rao, K. Dalwani, and S. Narsale. New tools for web-scale n-grams. http://nlp.cs.nyu.edu/publication/papers/heng-ngram.pdf.

[52] Seymour Lipschutz. *Schaum's outline of theory and problems of set theory and related topics*. Schaum's outline series. McGraw-Hill, 1998.

[53] F. Litmaath. Genus: Het lidwoord in het nederlands, December 2013. `http://www.inventio.nl/genus/uitleg.html#doc1`.

[54] D. Liu and D. Gildea. Syntactic features for evaluation of machine translation. `http://www.cs.rochester.edu/~gildea/pubs/liu-gildea-eval05.pdf`.

[55] Ding Liu and Daniel Gildea. Improved tree-to-string transducer for machine translation. In *Proceedings of the third workshop on statistical machine translation*, pages 62–69, Columbus, Ohio, USA, June 2008. Association for Computational Linguistics.

[56] Andread Maletti. Why synchronous tree substitution grammars? In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics – Proceedings of the Conference*, pages 876–884, Los Angeles, California, USA, June 2010. The Association for Computational Linguistics.

[57] Samuel E. Martin. *A reference grammar of Japanese*. Yale University Press, New Haven, CT, USA, 1975.

[58] James A. Mason. ASDParser algorithm overview, 1995. `http://www.yorku.ca/jmason/ASDParserOverview.html`.

[59] James A. Mason. Augmented syntax diagram grammars, August 1996. `http://www.yorku.ca/jmason/asdgram.htm`.

[60] James A. Mason. ASDParser algorithm details - Part 1: Phrase structure representation, 2008. `http://www.yorku.ca/jmason/ASDParserDetailsPart1.html`.

[61] James A. Mason. Augmented syntax diagram (ASD) home page, December 2012. `http://www.yorku.ca/jmason/asdindex.htm`.

[62] I. Dan Melamed. Multitext grammars and synchronous parsers. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 79–86, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[63] I. Dan Melamed, Giorgio Satta, and Benjamin Wellington. Generalized multitext grammars. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[64] Michael Meng and Markus Bader. Ungrammaticality detection and garden path strength: Evidence for serial parsing. *Language and Cognitive Processes*, 15(6):615–666, 2000.

[65] Moses - moses/syntaxtutorial, June 2012. `http://www.statmt.org/moses/?n=Moses.SyntaxTutorial`.

[66] Masaki Murata and Makoto Nagao. Determination of referential property and number of nouns in Japanese sentences for machine translation into English. In *In Proceedings of the 5th TMI*, pages 218–225, 1993.

[67] Masato Niimura. A syntactic analysis of copular sentences. *Nanzan Linguistics, Special Issue 3*, 1:203–237, 2007.

[68] Fujio Nishida and Shinobu Takamatsu. Japanese-English translation through internal expressions. In *COLING*, pages 271–276, 1982.

[69] Peter Norvig. On Chomsky and the two cultures of statistical learning, 2011. `http://norvig.com/chomsky.html`.

[70] Erik Nyberg, Teruko Mitamura, and Jaime G. Carbonell. The KANT machine translation system: From R&D to initial deployment. *Research Showcase - Carnegie Mellon University, Computer Science Department, School of Computer Science*, 1997.

[71] Franz Josef Och. Statistical machine translation live, 2006. `http://googleresearch.blogspot.nl/2006/04/statistical-machine-translation-live.html`.

[72] Franz Josef Och. Doubling up, 2008. `http://googleresearch.blogspot.nl/2008/09/doubling-up.html`.

[73] Kiyoharu Ono. Syntactic behaviour of case and adverbial particles in Japanese. *Australian Journal of Linguistics*, 16(1):81–129, 1996.

[74] Albert Oosterhof. *The semantics of generics in Dutch and related languages*. John Benjamins Publishing, Amsterdam, The Netherlands, 2008.

[75] Orandago nyuumon - Integral Dutch Course Nihongoban - 2.1 Kanshi. `http://dutch21.free.fr/ch02.html`.

[76] Web rangaku kotohajime, maki no 3, meishi to kanshi. `http://www.asahi-net.or.jp/~mx2y-soy/oranda/kap03.html`.

[77] Roxanne Marie Parent. Asd networks.com - augmented Syntax Diagrams - Java Demos, November 2013. `http://www.asdnetworks.com/home/`.

[78] Chasen demonstration, December 2013. `http://www.edrdg.org/~jwb/chasendemo.html`.

[79] Try juman, December 2013. `http://reed.kuee.kyoto-u.ac.jp/nl-resource/cgi-bin/juman.cgi`.

[80] Mecab/unidic demonstration, December 2013. `http://www.edrdg.org/~jwb/mecabdemo.html`.

[81] G.K. Pullum. *The great Eskimo vocabulary hoax and other irreverent essays on the study of language*. The University of Chicago Press, Chicago/London, 1991.

[82] Majid Razmara. Application of tree transducers in statistical machine translation. Depth report, Simon Fraser University, Canada, 2011.

[83] Jan Renkema. Genus: Het geslacht van zelfstandige naamwoorden, December 2013. `http://www.inventio.nl/genus/uitleg.html#doc2`.

[84] Jan Rijkhoff. *The Noun Phrase: A typological study of its form and structure*. Universiteit van Amsterdam, Amsterdam, The Netherlands, 1992.

[85] Jan Rijkhoff. *The Noun Phrase*. Oxford Studies in Typology and Linguistic Theory. Oxford University Press, 2002.

[86] M.T. Rosetta. *Compositional Translation*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1994.

[87] Dietmar Rösner. When Mariko talks to Siegfried - experiences from a Japanese/German machine translation project. In *COLING*, pages 652–654, 1986.

[88] Stuart M. Shieber and Yves Schabes. Generation and synchronous tree-adjoining grammars. *Computational Intelligence*, 7(4):220–228, 1992.

[89] Neal Snape, María del Pilar García Mayo, and Ayşe Gürel. Spanish, Turkish, Japanese and Chinese L2 learners' acquisition of generic reference. In Melissa Bowles, editor, *Proceedings of the 10th Generative Approaches to Second Language Acquisition Conference*, Somerville, MA, USA, 2009.

[90] Sangweon Suh. *Extracting Generic Statements for the Semantic Web*. PhD thesis, University of Edinburgh, 2006.

[91] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Pearson Education, Inc., Boston, MA, USA, 2006.

[92] Koichi Tateishi. *The syntax of 'subjects'*. CSLI Publications, Stanford, California, 1994.

[93] Shingo Tokimoto. Serial sentence processing in japanese. In *The Second International Conference on Cognitive Science and The 16th Annual Meeting of the Japanese Cognitive Science Society Joint Conference (ICCS/JCSS99)*, pages 3–15, 1999.

[94] Sneha Tripathi and Juran Krishna Sarkhel. Approaches to machine translation. *Annals of Library and Information Studies*, 57:388–393, December 2010.

[95] Natsuko Tsujimura. *An introduction to Japanese linguistics*. Blackwell Publishers Inc., Cambridge, Massachusetts, USA, 1996.

[96] Frank van Eynde. Automatische vertaling. *Colloquium Neerlandicum*, 11, 1991.

[97] Véronique van Gelderen. *Scrambling Unscrambled*. Landelijke Onderzoeksgroep Taalwetenschap: LOT (Netherlands Graduate School of Linguistics), Utrecht, The Netherlands, 2003.

[98] Onoonosama no eigo/eiyaku - eiwajiten/waeijiten - Weblio jisho, April 2013. http://ejje.weblio.jp/content/%E3%81%8A%E3%81%AE%E3%81%8A%E3%81%AE%E3%81%95%E3%81%BE.

[99] Wikipedia. Japanese counter words, March 2013. http://en.wikipedia.org/wiki/Japanese_counter_word.

[100] Place–manner-time (wikipedia), 2012. https://en.wikipedia.org/wiki/Place%E2%80%93manner%E2%80%93time.

[101] Time–manner–place (wikipedia), 2012. https://en.wikipedia.org/wiki/Time%E2%80%93manner%E2%80%93place.

[102] Kenji Yamada and Kevin Knight. A syntax-based statistical translation model. In *ACL*, pages 523–530, 2001.

[103] Hiroko Yamashita. Structural computation and the role of morphological markings in the processing of japanese. *Language and Speech*, 43(4):429–459, 2000.

[104] Banana (吉本ばなな) Yoshimoto. *Shirakawayofune (*白河夜船*)*. Kadokawa Bunko (角川文庫), 1992.

[105] Richard Zens, Franz Josef Och, and Hermann Ney. Phrase-based statistical machine translation. In *KI 2002: Advances in Artificial Intelligence*, Lecture Notes in Computer Science Volume 2479, pages 18–32. Springer Verlag, 2002.

[106] Hendrikje Ziemann, Fred Weerman, and Esther Ruigendijk. Nederlands later geleerd: gebruik van lidwoorden en flexie van bijvoeglijke naamwoorden door Duitstalige kinderen en volwassenen. *Internationale Neerlandistiek*, 49(3), October 2011. http://www.internationaleneerlandistiek.nl/vol49/nr03/a01.

A Rules for deciding which article a Dutch noun gets

As mentioned in section 5, this information can be expected to already be in the dictionary. However, if a noun is encountered that is not in the dictionary, these rules can be used.

On Japanese sites to learn Dutch [75, 4, 76], the following rules for articles are listed:

- Use 'de' with singular male and female nouns, use 'het' with singular neutral nouns. As the gender is not visible from the noun, learn them by heart.
- When a noun is plural, always use 'de'.
- All singular diminutive nouns have 'het'.
- The English indefinite article 'a(n)' and the Dutch indefinite article 'een' have the same meaning.
- Use 'het' with the following prefixes and suffixes: be-, ge-, ont-, ver-[46], -isme, -ment, -sel, -(t)je (etc.)
- Use 'de' with the following suffixes: -de, -te, -heid, -ie, -ij, -ing, -nis, -iteit, -st (etc.)

No rules are listed for when *not* to use an article.

The following are rules for deciding on the gender of a noun in Dutch (translated from [83], with extra references listed if I added items from other sources):

- Male words ('de'):
  1. Words with the suffixes -aar, -aard, -er and -erd.
     For example: *leugenaar, dronkaard, bakker , engerd* (except 'baker', which is female as it's traditionally a female occupation)
  2. Verb stems used as nouns.
     For example: *bloei, dank, groei, schrik, slaap*
  3. Words indicating male persons and animals.
     For example: *oom, neef, dief, verpleger, hengst, reu, haan*
- Female words ('de'):
  1. Words with the native suffixes
     -heid, -nis, -schap: *waarheid, kennis, beterschap*
     -de or -te: *liefde, diepte*
     -ij, -erij, -arij, -enij and -ernij: *voogdij, bedriegerij, rijmelarij, artsenij, razernij*
     -ing or -st following a verb stem: *wandeling, winst* (but *dienst* is male).

[46] It is also possible to have words starting with 'ver-' and still need 'de', for example in 'de vergadering', so I don't think it's correct to include this suffix in this list.

2. Words with the foreign suffixes or elements
   -ie, -tie, -logie, -sofie, -agogie: *familie, politie, biologie, filosofie, demagogie* (but *kanarie* is male)
   -iek, -ica: *muziek, logica* (but *lambiek* is male)
   -theek, -teit, -iteit: *discotheek, puberteit, subtiliteit*
   -tuur, -suur: *natuur, censuur*
   -ade, -ide, -ode, -ude: *tirade, asteroïde, periode, amplitude*
   -age, -ine, -se: *tuigage, discipline, analyse*
   -sis, -xis, -tis: *crisis, syntaxis, bronchitis*
3. Words indicating female persons and animals.
   For example: *tante, nicht, dievegge, verpleegster, merrie, teef, kip*

- Words that are both male and female ('de'):

1. Most object names that originally were exclusively female.
   For example: *bank, kast, naald, pijp*
2. General geographical names and names of heavenly bodies.
   For example: *stad, rivier, maan, ster*
3. Adjectives and past participles used as nouns.
   For example: *zieke, blinde, betrokkene, gewonde*
4. Words that can be used to indicate both males and females.
   For example: *baby, deugniet, arts, babbelkous*

- Neutral words ('het'):

1. Diminutives.
   For example: *tientje, bloempje, lammetje*
2. Verb stems with the prefixes 'be-', 'ge-' and 'ont-'.
   For example: *beraad, gedoe, ontslag*
3. Verbs used as nouns.
   For example: *het spreken, het voetballen, het gebeuren* [53]
4. Most verb stems used as nouns.
   For example: *het werk, het feest, het cijfer* [53]
5. Adjectives used as nouns.
   For example: *het fijne, het gekke, het donker, het donkere* [53]
6. Almost all two-syllable nouns that begin with 'be-', 'ge-' or 'ver-' if the first syllable is not stressed.
   For example: *het bezwaar, het bestek, het benul,*
   het gevaar, het gezin, het gemak,
   het verlof, het verband, het vernuft
   and NOT *de gever, de bezem, de verte* [53]
7. Multi-syllable words that start with unstressed 'ge-' followed by a verb stem.
   For example: *het gedonder, het geroddel, het geouwehoer* [53]

8. Collective nouns that start with unstressed 'ge-' and end with unstressed '-te'.
   For example: *het gedierte, het gebergte, het gesteente*
   and not, for example, *geboorte* and *gelofte* [53]
9. The majority of loanwords.
   For example: *het toilet, het manuscript, het cadeau* [53]
10. The names of countries and cities.
    For example: *Engeland, Brussel,*
    het Amerika van vroeger [53]
11. The names of languages, metal, colours, directions of the wind.
    For example: *het Frans, het Papiaments, het Chinees,*
    het goud, het ijzer, het kwik,
    het rood, het pimpelpaars, het lichtblauw,
    het oosten, het noordoosten, het zuidwesten [53]
12. Words ending in '-isme', '-ment' and unstressed '-sel' [4, 53].
    For example: *het egoïsme, het kapitalisme, het protestantisme,*
    het evenement, het sacrament, het cement,
    het doopsel, het voedsel, het speeksel (though there are exceptions to the '-sel' rule, for example *de wissel*)

B Parsing

This part of the appendix is about parsing the input and the notation, as it's important to know what information is available to make a translation. For the section on parsing (B.1), I looked at psycholinguistic experiments to see how humans parse Japanese, followed by a description of how a computer can parse this language. The following section (B.2) describes a syntax diagram notation for the internal structure of NPs, and ends with a discussion of the problem of scrambling (chapter B.2.5).

B.1 Parsing of Japanese by humans and by machines

Parsing (either by humans or by a computer) is done by reading through a sentence sequentially and doing something with those strings. In the psycholinguistic literature, this is called 'serial parsing' and the evidence from experiments shows that this is the parsing strategy humans use [22, 103, 41, 64, 93, 23].

B.1.1 Parsing of Japanese by humans

When humans hear or read a sentence, it is not the case that they perceive the entire sentence all at once - it becomes available one word at a time (either by hearing the sentence being spoken, or reading the sentence). The human sentence processing mechanism already starts interpreting the sentence before knowing it entirely: you don't have to wait till the end of the sentence before being able to understand the first few words.

A question is whether or not the processor builds all possible sentence structures simultaneously (parallel processing), or just chooses one possibility and ignores the rest of the possible sentence structures (serial processing). When a sentence doesn't fit the expected pattern, as is the case with ambiguous and garden-path sentences (like the well-known 'A horse raced past the barn fell'), the processor has to choose another structure (in the case of the parallel processing mechanism), or backtrack and explore another possible syntactic structure (in the case of the serial processor).

Tokimoto [93] experimentally explored evidence for and against parallelism in Japanese sentence processing. Because of the results of his experiment, he concluded that structure building in human sentence processing is serial: "The human sentence processor does not construct parallel representations unless required by lexical ambiguity."

Fodor [23] noted that every Japanese sentence is "infinitely ambiguous up to its final word, and often multiply ambiguous even when complete", because Japanese is a head-final language (all important information like verbs, nouns, etc comes at the end), any argument can be left out of the sentence if it is already clear from the context, and phrases can be scrambled (see also section B.2.5). They conducted experiments and concluded that even though Japanese seems like a language full of ambiguities, the parser works in a way similar to the parser for English. They propose a so-called 'ranked flagged serial parsing model', in which "the processor pursues only one analysis at a time, but at a choice point posts a flag to note the existence of each alternative. Then if the parse later breaks down, the processor can go back and select one of these alternatives to pursue" [23]. The possible alternative sentence structures are ranked by their difficulty: the alternatives that are most difficult, take longer to compute. So their model is not purely serial, but it contains a little bit of parallel processing. However, unlike in a standard parallel model, not all possible sentence structures are completely developed.

Numerous experiments on the human parsing mechanism have shown that the sentence processor likes "simple but compact structures, which have no more tree branches than are necessary, and minimal tree-distance (walking up one branch and down another) between any pair of adjacent words" [22]. This is what the parsing mechanisms bases its guesses on when it first encounters a sentence. It does make use of grammatical information as soon as it becomes available, for example case markings. The parser attaches new information by a principle like Minimal Attachment or Local Association [23], which means that it builds the simplest possible structure.

Yamashita [103] observed in her experiments that there was a strong preference for a simplex clause when encountering three different cases in succesion. The sentence in (51) is a simplex sentence, but (52) contains the same case-markings in the same order in a sentence with a relative clause.

(51) Mary-*ga* John-*ni* ringo-*wo* ageta
Mary-NOM John-DAT apple-ACC gave
Mary gave an apple to John.

(52) Mary-*ga* John-*ni* [ringo-*wo* tabeta hito-wo] shoukai-shita
Mary-NOM John-DAT [apple-ACC ate person-ACC] introduced
Mary introduced [the person who ate an apple] to John.

Kamide [41] described an eye-tracking experiment which also showed that Japanese was parsed incrementally, so data was processed as soon as it came in. Sentences with a ga-ni-wo (simplex) structure were preferred in her experiments as well. She noted that "the Japanese processor can not only process pre-head arguments incrementally, but can also predict a plausible sub-sequent argument using case-marking information and real-world knowledge."

B.1.2 Parsing of Japanese by machine

To parse the Japanese input text, a dictionary is needed. The most complete open-source Japanese dictionary I've been able to find is JMdict [11], of which an example entry can be seen in section 4.3.1.

When using a serial notation for the language, like railroad diagrams (described in section B.2), taking the shortest path is similar to the way humans parse a sentence: incrementally, trying to fit each incoming word into a structure as quickly as possible. A depth-first search (like Mason uses for his Augmented Syntax Diagrams in section B.2.1) would be most similar to the way humans parse the input, but a breadth-first search can be used as well to quickly 'close' certain paths in the railroad diagram.

B.2 Railroad diagrams

Railroad diagrams (also called 'syntax diagrams') are a way to graphically represent context-free grammars in Backus-Naur Form (BNF), as this notation is easier to understand by both humans and automatical parsers. Railroad diagrams are mostly used to describe the syntax of a programming language, but people have also used them for English (as 'Augmented Syntax Diagrams' [59, 77], see below in section B.2.1. In this notation only the nodes contain information, not the edges. The way to read these diagrams is from left to right, following the arrows. These diagrams are useful, because you can see all the possible structures of a sentence/phrase at once. It is also shorter than drawing all possible tree structures separately.

Railroad diagrams can be a useful notation for natural language as well, because it's a serial notation: if you have a string of words, you can 'walk through' the diagram and choose the paths that fit your string of words to decide on the structure. It is similar to the way humans parse linguistic input, as described in section B.1. Even though the normal railroad diagrams are context-free, natural language isn't entirely context-free. So if I want to use these for natural language, I can introduce context by making use of features (which I also used in the rules described in section 7) and context-nodes. These context-nodes can contain information about the context of previous sentences, for example that the subject or topic of the previous sentence was a man. This problem was also described in section 5.3, but just as in the mini grammar, I did not include the context here either.

I chose not to draw the tree structure models as used in the Rosetta system, because it's easier to express loops in a railroad diagram than to try and draw loops in tree

structures. When writing rules, you can show loops by making use of the + or ∗ as used in regular expressions, where + means '1 or more' and ∗ means '0 or more'.

But similar to the Rosetta system, I am making use of features. These are necessary to make words related to the noun get the same number (singular/plural) so the translation is consistent. Loops are important to be able to express repetitions of (for example) adjectives easily. In many languages, it is possible to insert a long string of adjectives: 'the big blue soft (etc...) cushion'.

B.2.1 Mason's "Augmented Syntax Diagrams"

When I had started drawing the railroad diagrams for Japanese and Dutch (simple) noun phrases, I came across someone who had created similar syntax diagrams with features in 1996 and wrote about it on his website, which was last updated in December 2012 at the time of writing [61]. An example of such a diagram, which he calls an 'augmented syntax diagram' (ASD), can be seen in figure 15 for the structure of an NP. The structures of ADJ and PP would be in a separate diagram.

Figure 16 is an image from his website where he shows the structures a noun phrase in English can take, and figure 17 shows all possible structures for an English cardinal number phrase.

He has only created ASDs for English and hasn't used them for translation. When given an English phrase, the parser performs a depth-first search to try and find matching grammar nodes. It uses a depth-first search so it first returns one possible parse and its semantic type, after which it can backtrack and try to find other possible parses. This parser could also be modified to a breadth-first search [58]. If the parser reaches a dead end, it should also be able to backtrack to the most recent point of ambiguity and then try an alternative route [60]. If a breadth-first search was used, some paths would just end and others would reach the final node.

The parser can set feature values and semantic values of nodes, like number and case, in a phrase structure. These features and values belong to objects (nodes are objects; a feature can be any string and a value can be any Java object, as the author implemented it in Java). The parser maintains a set of feature-value pairs at the top level of the phrase structure for the current subphrase (the current search path) being parsed [58].

B.2.2 Analysis of the Proper Noun structure

Proper nouns are tagged as such in the JMdict (n-pr), though at the moment of writing there are only five proper nouns in this dictionary. However, often-used morphological parsers like ChaSen/IPADIC [78], Juman [79] and MeCab/Unidic [80] do recognize proper nouns.

In Japanese, honorific suffixes are often attached to proper names, though it is also possible to have a proper noun without a honorific in Japanese. In Dutch, it is much more common to not use any 'honorifics', so if there is not a really well-fitting translation, it's probably best to leave off the honorific entirely.

The most common (honorific) suffixes that can follow proper nouns are listed in table 18 (singular) and table 3 (plural). A similar table can be found in section 7.2.9 (table 7).

Most of these suffixes fall into the 'suffix' (suf) or 'noun used as a suffix' (n-suf) POS-category in JMdict. Only 'senpai' is tagged as a noun ('n'), and 'sensei' and 'kun' have both POS-tags. The 'suffix' category also contains many more suffixes, but most aren't used with pronouns.

Suffix:	**JMdict POS:**	**Used for:**	**Possible Dutch translation:**
A-san	suf	male and female (Mr., Miss, Mrs., Ms.)	meneer / mevrouw (for older people and/or people the speaker doesn't know well; or nothing, for younger people or people familiar to the speaker)
A-sama	suf	male and female (very respectful version of '-san')	hooggeachte meneer / hooggeachte mevrouw
A-kun	suf, n	male (mostly)	(probably, translating it with nothing is best)
A-chan	suf	male and female little children or grandparents	kleine A (but probably, translating it with nothing is best)
A-bou	suf	male babies and young boys	kleine A
A-senpai	n	male and female senior colleagues, students of a higher grade	(probably translating it with nothing is best)
A-sensei	suf, n	male and female teachers	meneer / mevrouw / meester / juffrouw
A-hakase	n-suf	professor ('Dr.' or 'Doctor' as a doctorate/PhD title)	professor A

Table 18: Suffixes (honorifics) in Japanese and their Dutch translations. 'A' is a proper noun. When encountering these suffixes with a proper noun, the translation of the proper noun will be singular.

There is a short list of suffixes that indicate plural: '-kata', '-gata', '-tachi', '-ra', and '-domo' (see also table 3). All of these suffixes can follow proper nouns, but a name cannot (in general) be pluralized. It's best then to translate the plural suffix then as "en de anderen" ("and the others").

The structure of the 'proper noun' node in the railroad diagrams in figure 22 and figure 20 is detailed as a railroad diagram in figure 18.

Japanese basic expressions	Dutch basic expressions	basic meanings
$\text{aff}_{num=sing}$(san)	$\text{aff}_{gnd=m}$(meneer), $\text{aff}_{gnd=f}$(mevrouw), $\text{aff}_{gnd=?}$(∅)	*Mr/Mrs'*
$\text{aff}_{num=sing}$(sensei)	$\text{aff}_{gnd=m}$(meester), $\text{aff}_{gnd=f}$(juffrouw), $\text{aff}_{gnd=?}$(∅)	*teacher'*
$\text{aff}_{num=sing}$(sama)	$\text{aff}_{gnd=m}$(hooggeachte meneer), $\text{aff}_{gnd=f}$(hooggeachte mevrouw), $\text{aff}_{gnd=?}$(hooggeachte)	*Esteemed Mr/Mrs'*
$\text{aff}_{num=sing}$(chan)	$\text{aff}_{gnd=?}$(kleine)	*little'*
$\text{aff}_{gnd=m,num=sing}$(bou)	$\text{aff}_{gnd=m}$(kleine)	*little'*
$\text{aff}_{gnd=m,num=sing}$(kun)	$\text{aff}_{gnd=m}$(∅)	*Mr'*
$\text{aff}_{num=sing}$(senpai)	$\text{aff}_{gnd=?}$(∅)	*senior'*
$\text{aff}_{num=sing}$(hakase)	$\text{aff}_{gnd=?}$(professor)	*professor'*
$\text{aff}_{num=plur}$(tachi)	$\text{aff}_{num=plur}$(en de anderen)	*and the others'*
$\text{aff}_{num=plur}$(kata)	$\text{aff}_{num=plur}$(en de anderen)	*and the others'*
$\text{aff}_{num=plur}$(gata)	$\text{aff}_{num=plur}$(en de anderen)	*and the others'*
$\text{aff}_{num=plur}$(ra)	$\text{aff}_{num=plur}$(en de anderen)	*and the others'*
$\text{aff}_{num=plur}$(domo)	$\text{aff}_{num=plur}$(en de anderen)	*and the others'*

Table 19: Honorific affixes in Japanese and their translations into Dutch and English.

B.2.3 Analysis of the simple NP structure

I thought to start with simple NPs of the structure (article-adverb-adjective-noun singular/plural), but quite soon I found that there are many more possible structures for an NP without it becoming a complex NP (which is an NP that contains an S, or sentence [49]).

The basic NP diagram for Dutch and Japanese, containing article/adverb/adjective/noun can be seen in figure 19. A noun is obligatory in a noun phrase (in both languages), but all other nodes are optional. The features are not shown, but they are attached to the nodes (which are the squares that contain the information). The reason is that this diagram just shows the syntactic structures. The diagrams can easily be extended to include more NP-internal structures.

The extended version for Japanese can be seen in figure 20 and 21, and for Dutch in figure 22 and 23. To translate, each node in one language is mapped to a node in the other language's diagram (mappings are indicated by numbers on the images, but it is not the case that the same number on each graph maps the node to the same number on the other graph, for example number 5 on the Japanese graph maps to 25 on the Dutch graph). A dictionary is used to find the translations for the words. These diagrams can be used for parsing to map the words onto the nodes. Then rules are applied to make decisions about extra nodes that might need to be added in the other language (like an article node in Dutch). The rules described in section 7.2 for the features might also be used for these graphs with a few alterations.

In the extended NP-structure diagrams I drew, there are nodes with "van" in Dutch (figure 22) and "no" (meaning "of", or genitive "'s") in Japanese (figure 20). It might seem that the phrase starting with "van" is a PP, but Kooij [1] argues for Dutch that it's part of the NP and not a separate PP. He calls 'NPs with a PP included' (the "van" part is a subtree in the NP-structure) *larger NPs*. There's another argument for including "no" in the NP-structure instead of viewing this genitive particle as part of a PP: PPs starting with 'in' or 'over' are translated into Japanese as an NP followed by a particle, while the translation of 'of' is a "no" that appears inside of the NP itself.

B.2.4 Some small examples using my railroad diagram mappings

The question is, what are the results when using these railroad diagrams? Using the mapping between figures 20-21 and 22-23, I compared a few NPs with Google Translate [27], which uses statistics for translation (and does not use context either). The comparison can be found in table 20.

B.2.5 Scrambling

We already saw some various word orders within an NP, but those could all be described using one railroad diagram.

With only a noun phrase and a verb phrase, there is no other possible word order in a sentence. But if the sentences get longer, putting all possible word orders into rules also causes an explosion of rules, if you keep to the strict isomorphy rules. When the isomorphy rules are relaxed a bit, this does not have to be the case: if you allow 'identity rules' which don't need to get a corresponding rule. These rules take the meaning of the input and give the same meaning as a result (similar to identity functions[47]). However, even though scrambling is possible in Japanese (and to a certain extent in Dutch), there is still a base word order which can be used for machine translation (though the system should be able to recognize the different word orders and construct a correct 'base word order' sentence in the target language from it).

The basic word order in Japanese is subject-object-verb, though subjects are optional and it is possible to change the word order (for emphasis) as the words are case-marked by using particles. Tsujimura [95] writes that "In formal speech or written language, it is rare to see sentences with the object-subject word order, and many Japanese speakers intuitively feel that the subject-object word order is more basic."

Scrambling in Dutch can only happen in a few cases. The subject normally appears before the finite verb, as in (53). However, the adpositional phrase can also appear before the finite verb instead of the subject. In that case, the subject moves to the position directly following the finite verb, as in (54). The object can also appear in front of the finite verb, as in (55). In short, every phrase could appear in front of the finite verb (except for other verbs), but it's not common to put other phrases than the subject in front of the verb. Phrases appearing in front of the finite verb can be a topic/given

[47] An identity function ($I_A : A \to A$ such that $I_A(a) = a$ for every element $a \in A$) [52], so the output is the same as the input, in short $I(x) = x$.

Japanese sentence	**Japanese diagram**	**Dutch diagram**	**GoogleTranslate Japanese to Dutch (2014/05/16)**
赤い本	4 5 akai hon	35 26 25 een rood boek 36 26 25 het rode boek 37 26 25 de rode boeken 26 25 rode boeken	Red book
一冊の赤い本	19 16 10 4 5 issatsu no akai hon	43 26 25 één rood boek	Red een boek
杏奈の赤い本	13 10 4 5 Anna no akai hon	30 29 26 25 Anna's rode boek 36 26 25 13 11 het rode boek van Anna (All other 4 options for 'akai hon' in Dutch can appear here as well, so instead of 36 you can also have 35, 37, or only '26 25'.)	Rode boek van Anna
この赤い本	25 4 5 kono akai hon	49 26 25 dit rode boek 51 26 25 deze rode boeken	Het rode boek
たくさんの赤い本	19 10 4 5 takusan no akai hon	40 26 25 veel rode boeken	Rode veel boeken

Table 20: Small examples using the railroad diagrams and comparing them with GoogleTranslate results.

information (*Dat weet ik niet*, literally 'That know I not' meaning "I don't know that"), or the phrase has been put at the front of the sentence for emphasis.

(53) [Jan] leest [op zondag] [het boek].
[Jan] reads [on Sunday] [the book].

(54) [Op zondag] leest [Jan] [het boek].
[On Sunday] reads [Jan] [the book].

(55) [Het boek] leest [Jan] [op zondag].
[The book] reads [Jan] [on Sunday].
'Jan reads the book on Sunday.'

An object determiner phrase can also appear on both sides of an adverb, as can be seen in example (56) and (57) from [97] (brackets added by me).

(56) Iedereen weet dat Jan [op zondag] [het boek] heeft gelezen.
everyone knows that Jan [on Sunday] [the book] has read

(57) Iedereen weet dat Jan [het boek] [op zondag] heeft gelezen.
everyone knows that Jan [the book] [on Sunday] has read
'Everyone knows that Jan has read the book on Sunday.'

Adpositional phrases in Japanese have the same order as in German and Dutch (time-manner-place) [101], but in English this order is exactly the other way around (place-manner-time) [100]. Examples of this can be seen in (58) and (59). Particles that can be used for 'manner' are 'de' and 'ni', particles that can be used for 'place' are 'e' and 'ni'.

(58) Adpositional phrases in Japanese/German/Dutch:
time - manner - place
kinou - kuruma de - mise e (Japanese)
gisteren - met de auto - naar de winkel (Dutch)
yesterday - by car - to the store

(59) Adpositional phrases in English:
place-manner-time
to the store - by car - tomorrow

References

[1] Kooij, J.G.: *Aspekten van woordvolgorde in het Nederlands*, Publikaties van de Vakgroep Nederlandse Taal- en Letterkunde, Leiden, The Netherlands, 1978. `http://www.dbnl.org/tekst/kooi003aspe01_01/kooi003aspe01_01_0005.php`

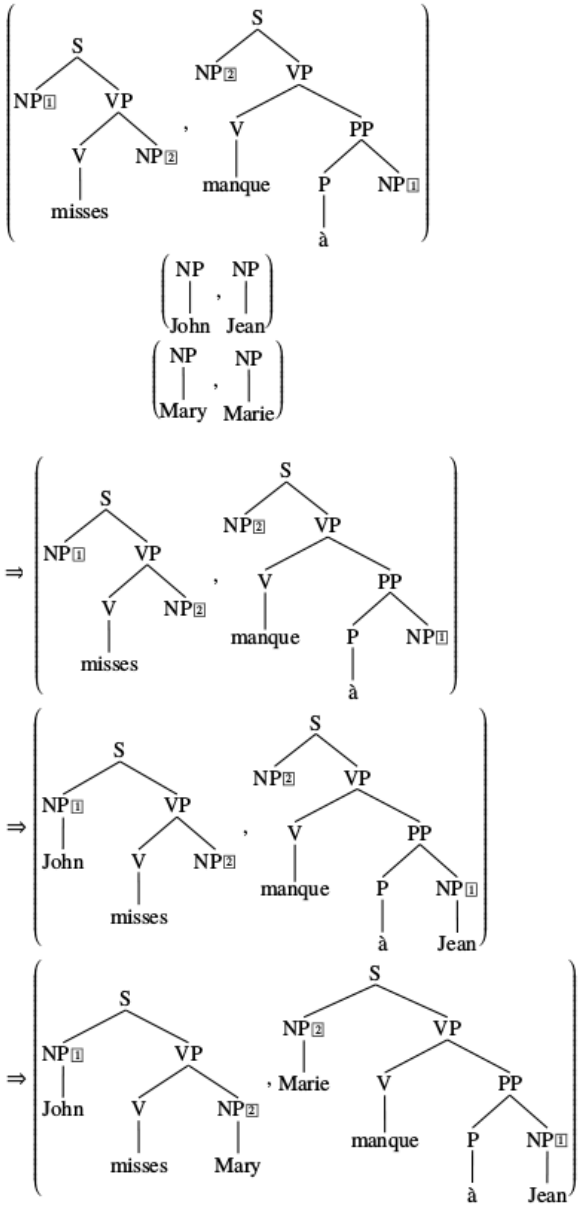


Figure 5: Example of a part of an English-French STSG, from [16]

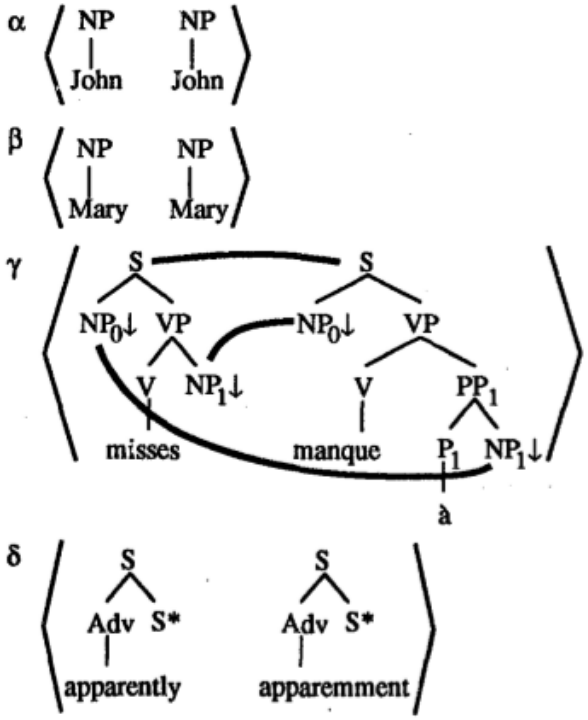


Figure 6: Example of a part of an English-French STAG transfer lexicon, from [2]

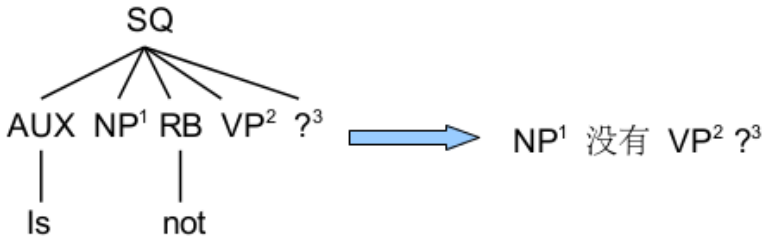


Figure 7: Example of a template (from [55]) used by a tree-to-string transducer, from English to Chinese.

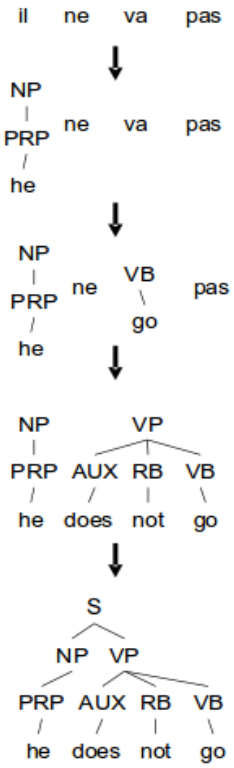


Figure 8: Example of a derivation from French to English using a string-to-tree transducer. From [25].

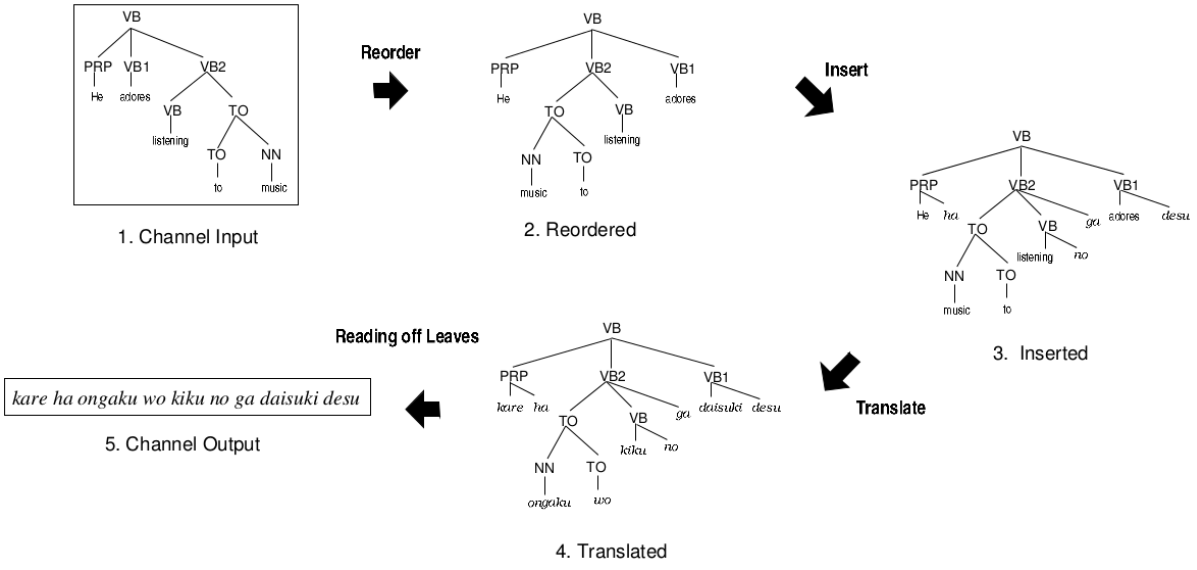


Figure 9: Example of the different operations applied by the tree-to-tree transducer from [102].

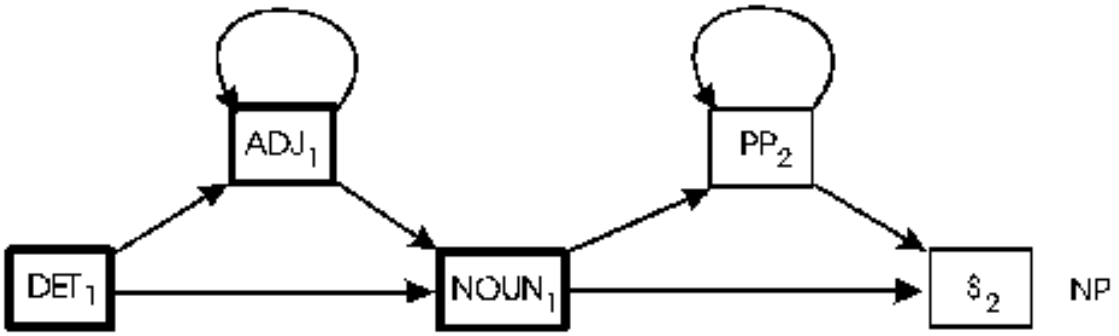


Figure 15: 'Augmented syntax diagram' for a noun simple phrase (bottom) [59]. The nodes containing a $ symbol are empty nodes.

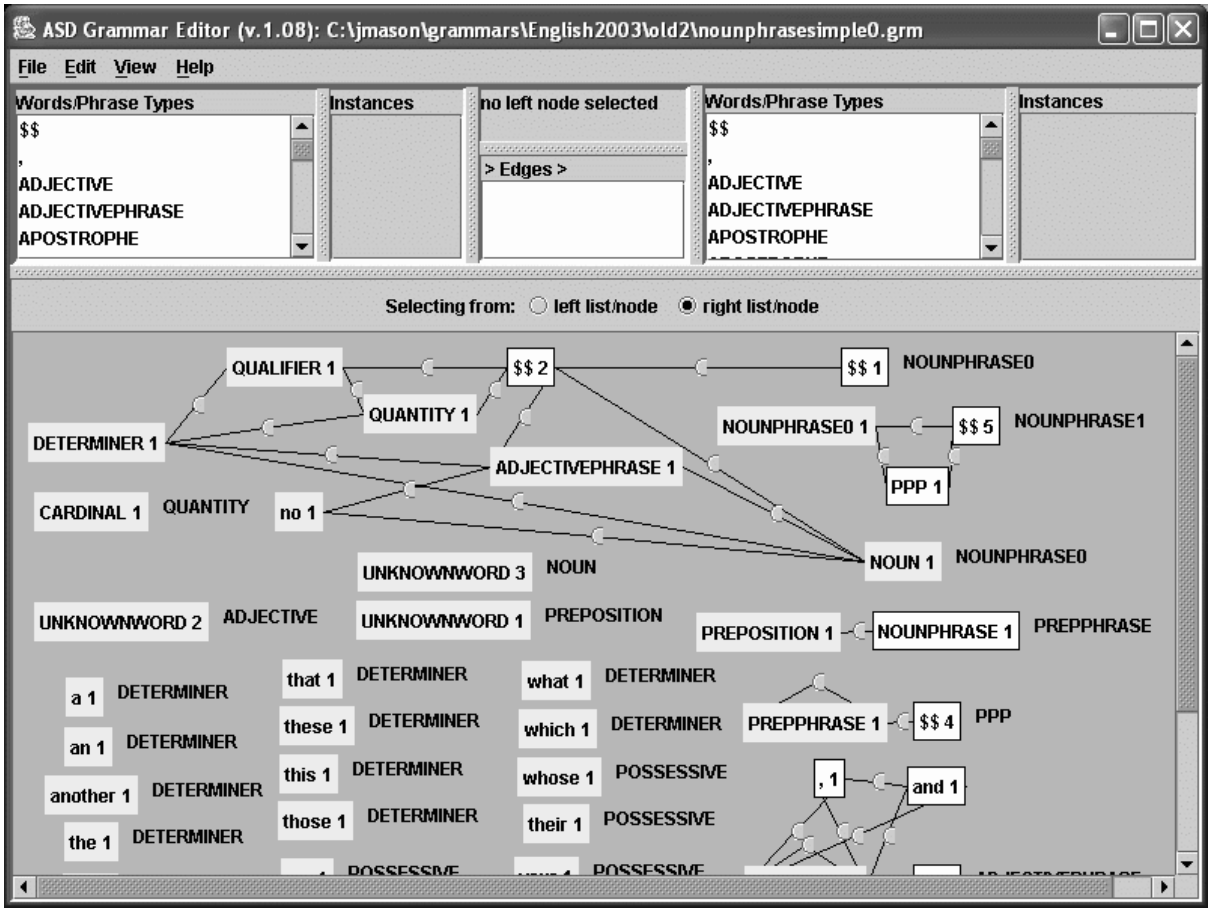


Figure 16: The structures a noun phrase in English can take, notated as augmented syntax diagrams [61].

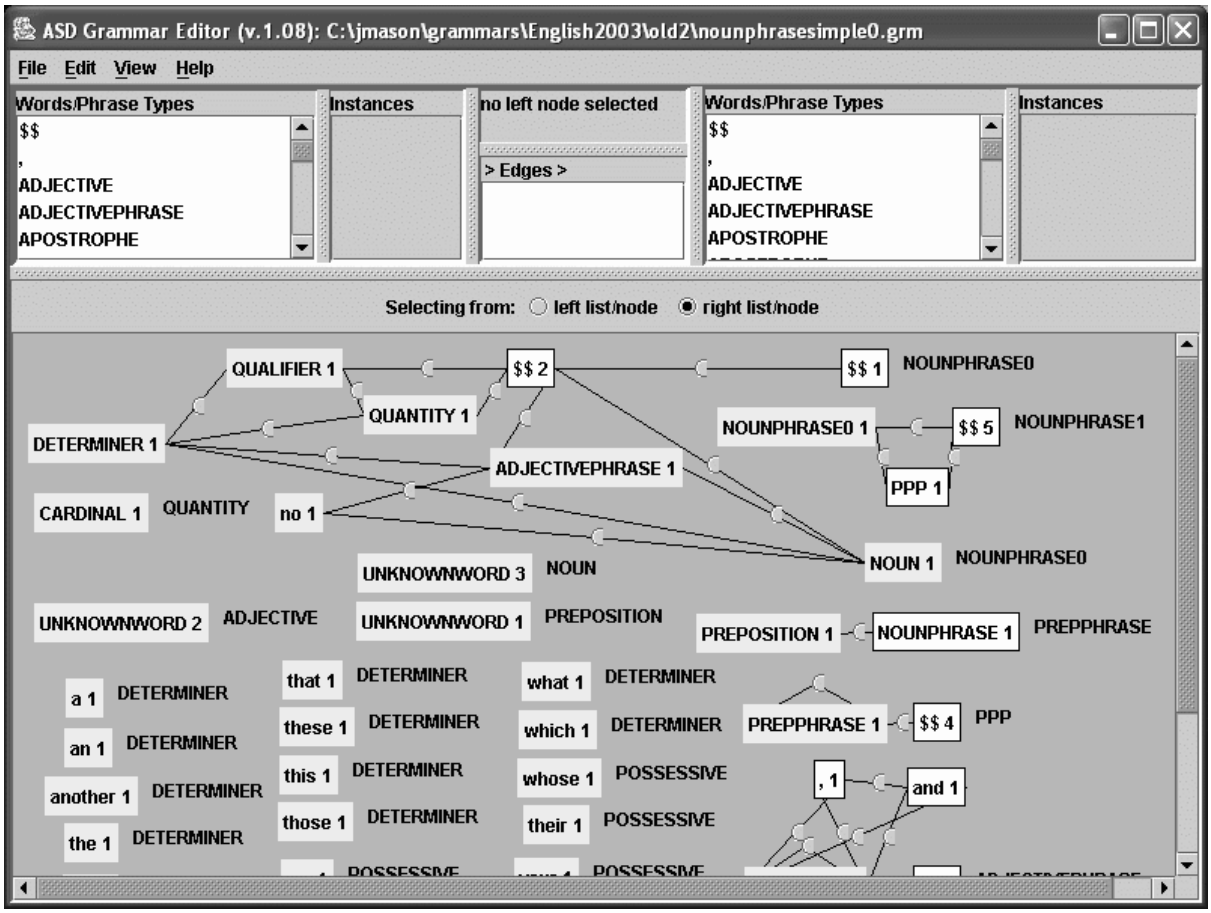


Figure 17: The structures a cardinal number phrase in English can take, notated as augmented syntax diagrams [61].

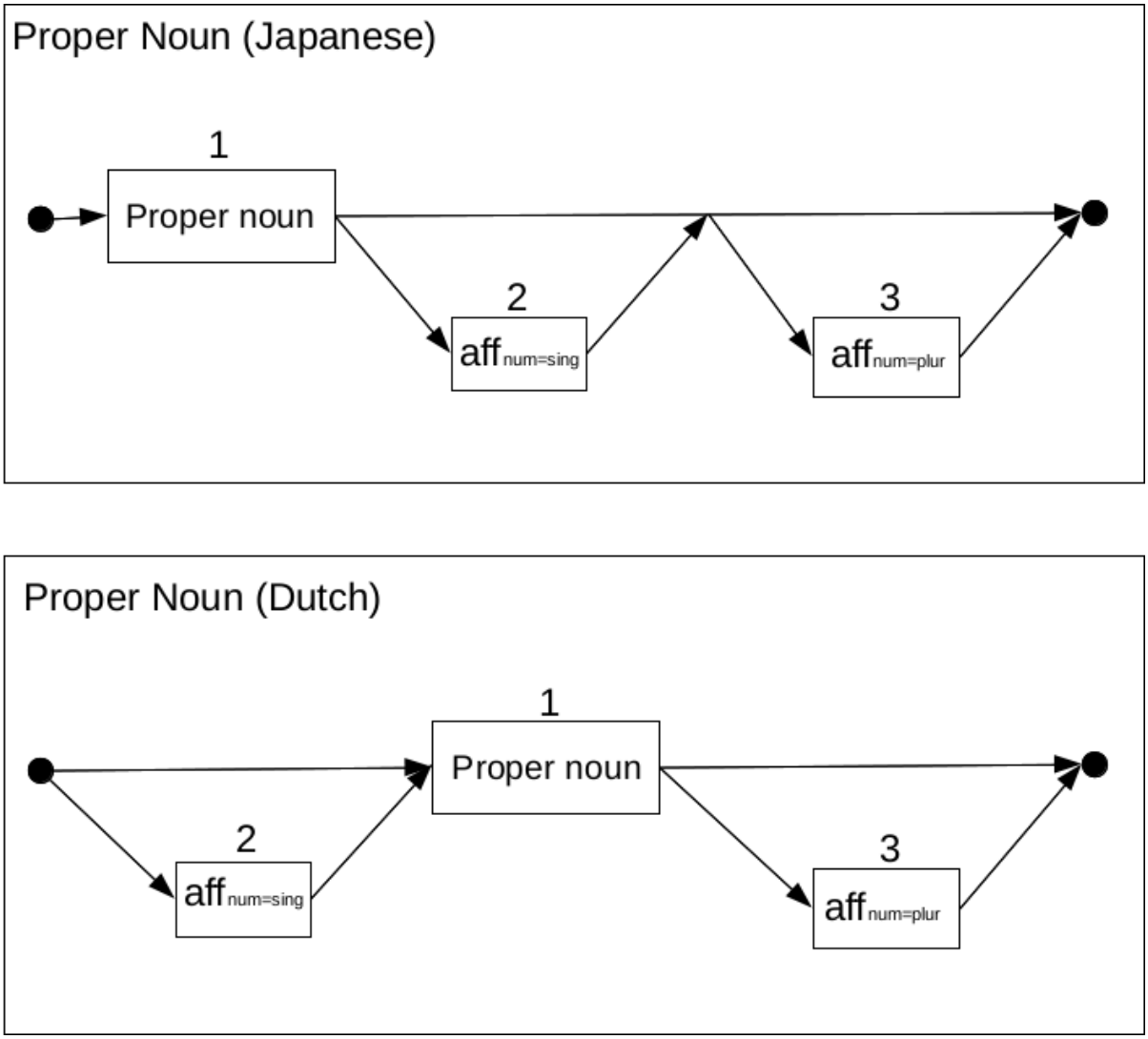


Figure 18: Railroad structure diagram for proper nouns.

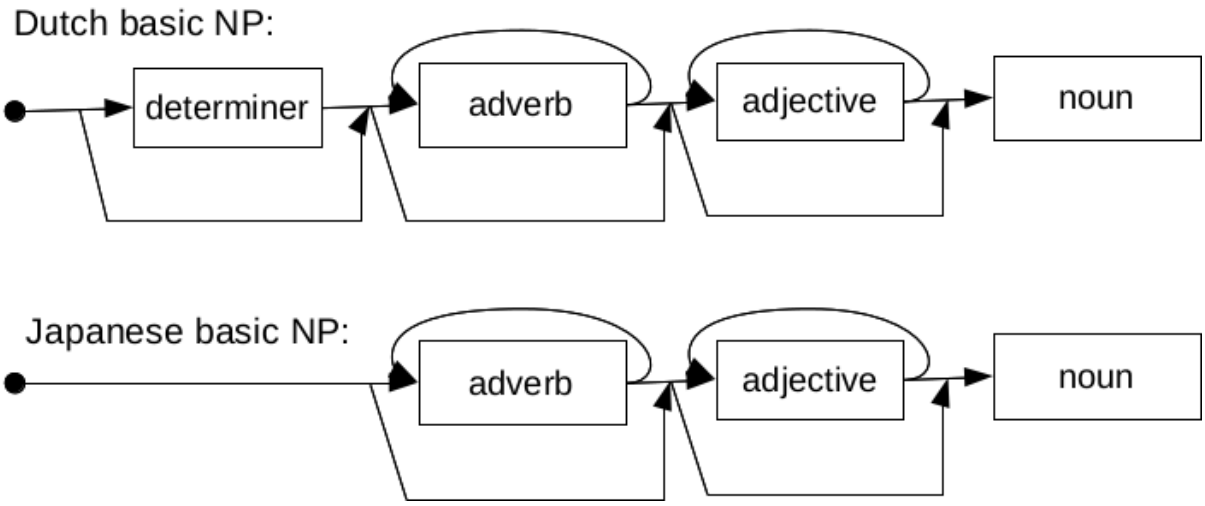


Figure 19: Basic structure for a Dutch and Japanese noun phrase.

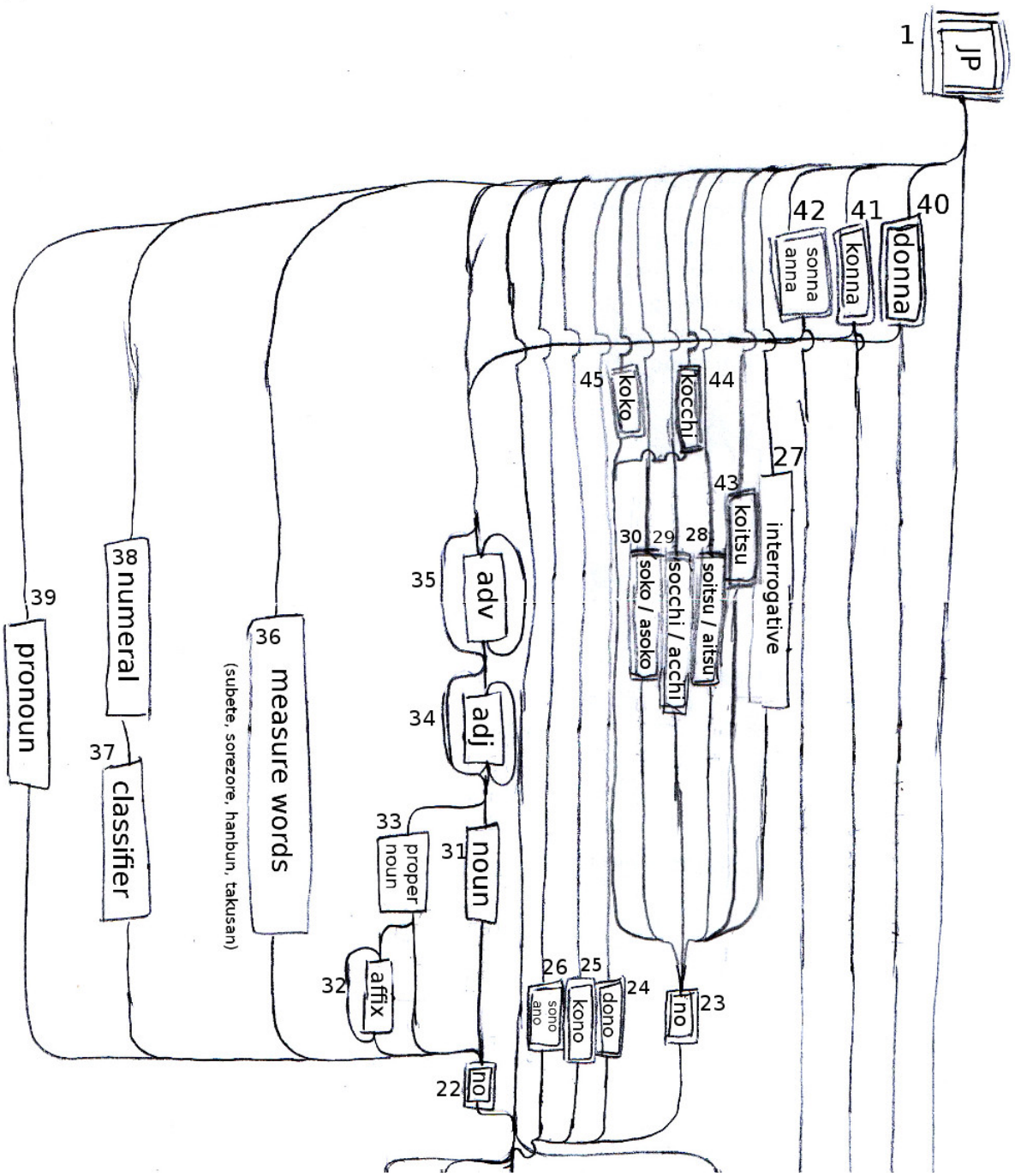


Figure 20: Railroad structure diagram for Japanese noun phrases, first half.

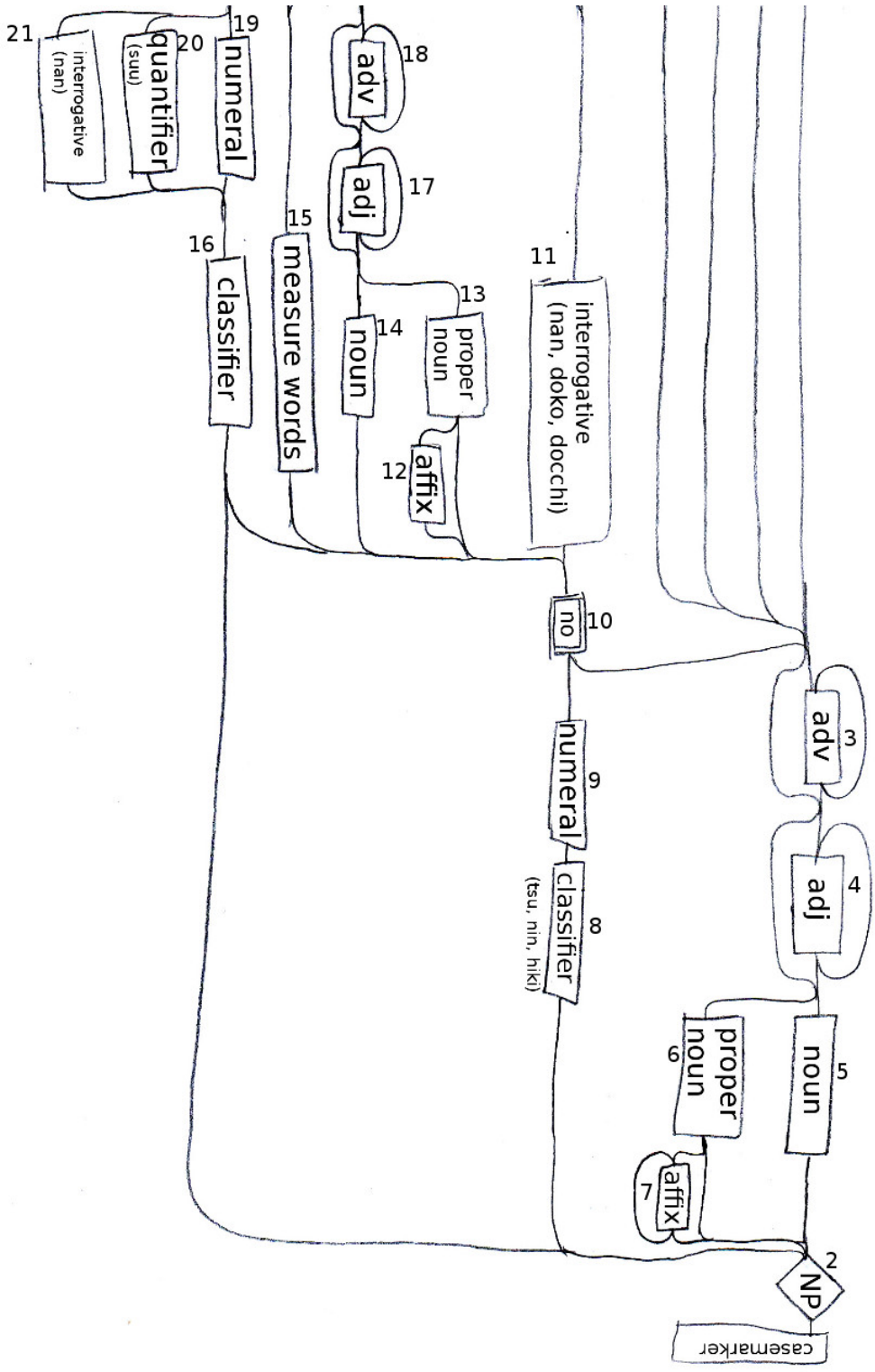


Figure 21: Railroad structure diagram for Japanese noun phrases, second half.

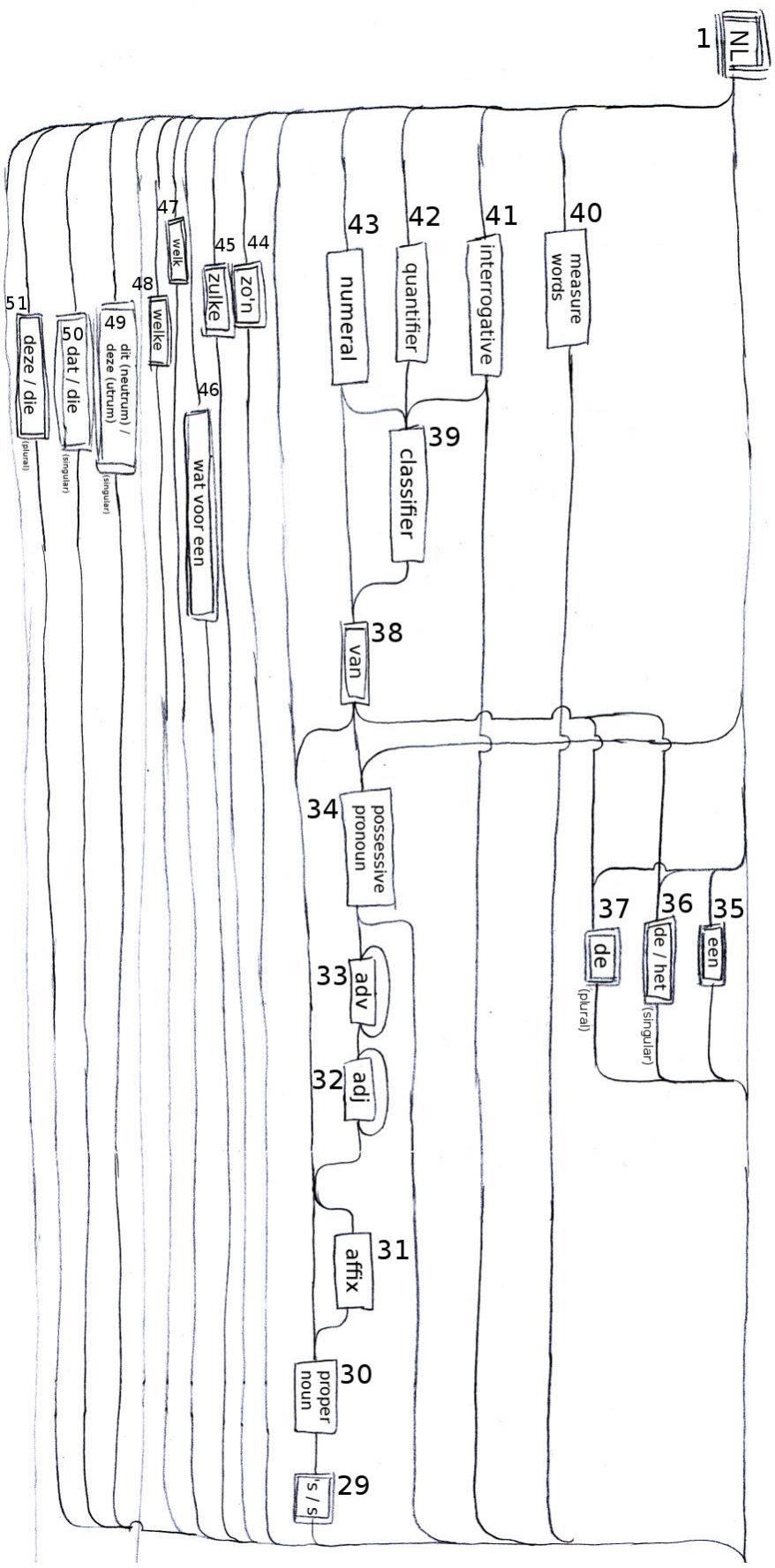


Figure 22: Railroad structure diagram for Dutch noun phrases, first half.

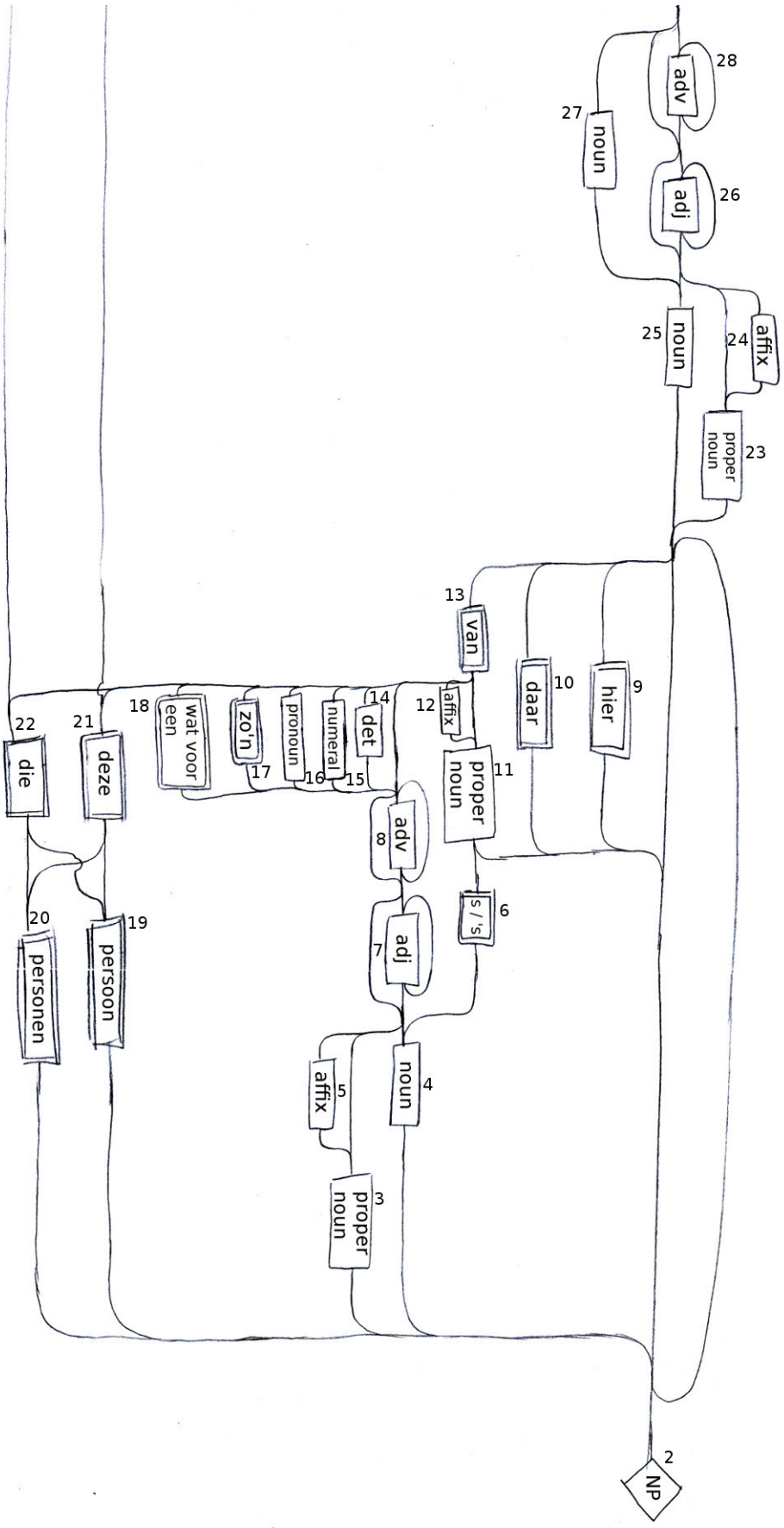


Figure 23: Railroad structure diagram for Dutch noun phrases, second half.