Automatic Acquisition of Transfer Rules from Translation Examples

Werner Winiwarter

Faculty of Computer Science, University of Vienna,
Liebiggasse 4, A-1010 Vienna, Austria,
werner.winiwarter@univie.ac.at,
WWW home page: http://www.ifs.univie.ac.at/~ww/

Abstract. In our research, we have developed a transfer-based machine translation architecture for the translation from Japanese into German. One main feature of the system is the fully automatic acquisition of transfer rules from translation examples by using structural matching between the parsing trees. The translation system has been implemented as part of a language learning environment with the aim to provide personalized translations for the students. In this paper we present our formalism to represent syntactic and transfer knowledge, and explain the various steps involved in acquiring and applying transfer rules.

1 Introduction

The main aim of our research is the development of a machine translation system, which produces high quality translations from Japanese into German. A second important requirement is full customization of the system because, in our opinion, there exists no "perfect" translation but only a preferred one for a certain user. Therefore, the post-editing of a translation should result in an automatic update of the translation knowledge.

Furthermore, we had to consider the two constraints that we had neither a large Japanese–German bilingual corpus nor resources to manually build a large knowledge base available. In any case, a large handcrafted knowledge base is in conflict with our need for flexible adaptation, and the insufficient data quality of today's large corpora interferes with our demand for high quality translations.

In our approach we use a transfer-based machine translation architecture (for good overviews of this topic see [1–3]). However, we learn all the transfer rules incrementally from translation examples provided by a user. For the acquisition of new transfer rules we use structural matching between the parsing trees for a Japanese–German sentence pair. To produce the input for the *acquisition component* we first compute the correct segmentation and tagging, and then transform the token lists into parsing trees. For the translation of a sentence, the *transfer component* applies the transfer rules to the Japanese parsing tree to transform it into a corresponding German parsing tree, from which we generate a token list and surface form. In this paper we focus on the acquisition and transfer components; for a description of the other system modules we refer to [4].

We have developed the machine translation system as part of *PETRA – a Personal Embedded Translation and Reading Assistant* [5]. PETRA is a language learning environment, which assists German-speaking language students in reading and translating Japanese documents, in particular, educational texts. It is fully embedded into Microsoft Word so that the students can invoke all the features from within the text editor. The incremental improvement of the translation quality encourages a bidirectional knowledge transfer between the student and the learning environment. Besides the translation features, PETRA offers the access to the large Japanese–German dictionary WaDokuJT and a user-friendly interface to add new dictionary entries. PETRA has been implemented using Amzi! Prolog, which provides full Unicode support and an API to Visual Basic for the communication with Microsoft Word.

The rest of the paper is organized as follows. We first introduce our formalism to represent the parsing trees and transfer rules in Sect. 2 and 3 before we describe the principal steps for the acquisition and application of the transfer rules in Sect. 4 and 5.

2 Parsing Trees

The parsing trees represent the input to the acquisition component. Instead of using a fixed tree structure we decided on a more flexible and robust representation. We model a sentence as a set of constituents (represented as list in Prolog). Each constituent is a compound term of arity 1 with the constituent name as principal functor. Regarding the argument of a constituent we distinguish two different constituent types: *simple constituents* representing features or words, and *complex constituents* representing phrases as sets of subconstituents.

This representation is compact because empty optional constituents are not stored explicitly, and is not affected by the order of the different subconstituents in the arguments of complex constituents. The latter is essential for a robust and effective application of transfer rules (see Sect. 4). Figure 1 shows an example of a Japanese parsing tree. The ta-form of the main verb indicates English past tense, expressed as perfect tense in German.

For the efficient traversal, processing, and manipulation of the arguments of complex constituents we have implemented several generic predicates. In the following, we list just those predicates that are used later on in the paper:

- *find_req*$(Csub, A, Asub)$: searches for subconstituent $Csub(Asub)$ in A, fails if $Csub(\_) \notin A$;
- *replace*$(Csub, A1, Asub, A2)$: replaces $Csub(\_) \in A1$ with $Csub(Asub)$ resulting in $A2$; if $Csub(\_) \notin A1$, $Csub(Asub)$ is inserted as additional subconstituent;
- *repl_diff*$(Csub1, Csub2, A1, Asub, A2)$: same as *replace* except that also the constituent name is changed to $Csub2$;
- *split*$(A, A1, A2)$: unifies all subconstituents of $A1$ with the corresponding subconstituents in A and computes $A2 = A \setminus A1$ (used for applying transfer rules with shared variables for unification, see Sect. 3.2).

いまのような形の本は、中世になって、はじめてあらわれた。
Das Buch in seiner heutigen Form ist im Mittelalter zum ersten Mal aufgetreten.
The book in its present form appeared in the Middle Ages for the first time.

[hew(あらわれる/ver),	head word – arawareru/verb – to appear
hwf(vta),	head word form – ta-form
pav(はじめて/adv),	predicative adverb – hajimete/adverb – for the first time
adp([hew(中世/nou),	adverbial phrase – head word – chuusei/noun – Middle Ages
php(になって/par)]),	phrase particle – ninatte/particle – in
sub([hew(本/nou),	subject – head word – hon/noun – book
anp([hew(形/nou),	attributive noun phrase – head word – katachi/noun – form
anp([hew(いま/nou)])])])]	attributive noun phrase – head word – ima/noun – present

Fig. 1. Example of Japanese parsing tree

3 Transfer Rules

The transfer rules are stored as *facts* in the rule base. We have defined several *predicates* for the different types of rules. Therefore, when we talk about rules in the following, we always refer to transfer rules for machine translation in the general sense, not to logical rules in the strict sense of Prolog.

One characteristic of our transfer rules is that we can cover most translation problems with a very small number of generic abstract predicates. For some common problems we have defined some additional specialized predicates. However, all of these specialized predicates could also be expressed by using our generic predicates. We introduce them merely to increase the compactness of our rule base and the efficiency of rule application.

In the next subsections we give an overview of the different rule types along with illustrative examples. For the ease of the reader we use Roman transcription for the Japanese examples instead of the original Japanese writing.

3.1 Rules for Translating Simple Constituents

For simple context-insensitive translations at the word level, the argument of a simple constituent $A1$ is changed to $A2$ by the predicate:

$$tr\_asc(A1, A2). \tag{1}$$

Example 1. The default transfer rule to translate the Japanese noun HON (`book`) into the German counterpart `Buch` is stated as the fact:

$$tr\_asc(\text{HON}/nou, \texttt{'Buch'}/nou).$$

This general fact is then supplemented with more specific context-sensitive facts to handle different word meanings in other contexts. We have included the word category in the fact specification so that we can easily model the situation where a translation changes the word category.

Example 2. The Japanese adjectival noun ONAJI (`same`) is translated into the German adjective `gleich`:

$$tr\_asc(\text{ONAJI}/ano, \texttt{gleich}/adj).$$

More complex changes that also affect the constituent itself can be defined by using the predicate:

$$tr\_sc(C1, C2, A1, A2). \tag{2}$$

A fact for this predicate changes a simple constituent $C1(A1)$ to $C2(A2)$, i.e. constituent name and argument are replaced.

Example 3. The Japanese attributive suffix (*asf*) DAKE (`only`) is expressed as attributive adverb (*aav*) `nur` in German:

$$tr\_sc(asf, aav, \text{DAKE}/suf, \texttt{nur}/adv).$$

This predicate can also be used in situations where a Japanese word is just expressed as syntactic feature in German.

Example 4. The Japanese attributive adverb MOTTOMO (`most`) corresponds to the superlative degree of comparison (*com*) of an adjective in German:

$$tr\_sc(aav, com, \text{MOTTOMO}/adv, sup).$$

The second constituent $C2$ is not restricted to a simple constituent, it can also be a complex constituent. This way we can model translations of simple words into whole phrases.

Example 5. The Japanese predicative adverb HAJIMETE (`for the first time`) is translated into the German adverbial phrase `zum ersten Mal`:

$$tr\_sc(pav, adp, \text{HAJIMETE}/adv,\\ [php(\texttt{zu}/prp), det(def), num(sng), seq(\texttt{erst}/ord), hew(\texttt{'Mal'}/nou)]).$$

The adverbial phrase is specified as set of five subconstituents: phrase particle `zu`/preposition, determiner type definite, number singular, sequence `erst`/ordinal numeral, and head word `Mal`/noun. Please note that `zum` is a contraction of the preposition `zu` and the definite article `dem` for the dative case.

Finally, for modelling situations where there exist different translations for a Japanese word depending on the constituent name, we also provide a shortcut for $tr\_sc(C, C, A1, A2)$:

$$tr\_scn(C, A1, A2). \tag{3}$$

3.2 Rules for Translating Complex Constituents

Just as in the case of simple constituents, the following predicate enables the substitution of arguments for complex constituents:

$$tr\_acc(Hew, Req1, Req2). \tag{4}$$

Facts for this predicate change the argument of a complex constituent from $A1 = Req1 \cup Add$ to $A2 = Req2 \cup Add$ if $hew(Hew) \in A1$. The head word Hew serves as index for the fast retrieval of matching facts and the reduction of the number of facts that have to be further analyzed. The application of a transfer rule requires that the set of subconstituents in $Req1$ is included in an input constituent to replace $Req1$ by $Req2$. Besides $Req1$ any additional constituents can be included in the input, which are transferred to the output unchanged. This allows for a flexible and robust realization of the transfer module (see Sect. 5) because one rule application changes only certain aspects of a constituent whereas other aspects are translated by other rules in subsequent steps.

Example 6. The Japanese adverbial phrase CHUUSEI NINATTE (`in the Middle Ages`) is translated into the German adverbial phrase `im Mittelalter`:

$$\begin{aligned} &tr\_acc(\text{CHUUSEI}/nou, [php(\text{NINATTE}/par), hew(\text{CHUUSEI}/nou)]\,, \\ &\quad [php(\texttt{in}/prp), det(def), num(sng), hew(\texttt{'Mittelalter'}/nou)]). \end{aligned}$$

Because $A1$ and $Req1$ are sets of constituents, the order of the subconstituents must not influence the matching of the two sets. Therefore, by applying the predicate *split* (see Sect. 2) we retrieve each element of $Req1$ in $A1$ to create a list of constituents $Req1_s$ in the same order as in $Req1$ and then try to unify the two lists. As a byproduct of this sorting process we obtain the set difference $Add = A1 \setminus Req1$ as all remaining constituents in $A1$ that were not retrieved.

Example 7. The expression JI O KAKU (literally `to write characters`) in the Japanese verb phrase KATAMEN NI JI O KAKU (`to write on one side`) is replaced by the verb `beschreiben` within the corresponding German verb phrase:

$$\begin{aligned} &tr\_acc(\text{KAKU}/ver, [hew(\text{KAKU}/ver), dob([hew(\text{JI}/nou)])], [hew(\texttt{beschreiben}/ver)]). \\ &A1 = [dob([hew(\text{JI}/nou)]), hwf(vdi), hew(\text{KAKU}/ver), \\ &\quad adp([php(\text{NI}/par), hew(\text{KATAMEN}/nou)])] \\ &Req1_s = [hew(\text{KAKU}/ver), dob([hew(\text{JI}/nou)])] \\ &Add = [hwf(vdi), adp([php(\text{NI}/par), hew(\text{KATAMEN}/nou)])] \\ &A2 = [hew(\texttt{beschreiben}/ver), hwf(vdi), [adp([php(\text{NI}/par), hew(\text{KATAMEN}/nou)])] \end{aligned}$$

The rule specifies that for any complex constituent with head word KAKU and direct object (*dob*) JI these two subconstituents are replaced by the head word `beschreiben`. In this example the input $A1$ for the application of this rule is a verb phrase with direct object JI, head word form *vdi* (verb dictionary form), head word KAKU, and adverbial phrase KATAMEN NI. $Req1_s$ is extracted from $A1$ in the correct order, leaving Add as list of remaining subconstituents. The result $A2$ is then formed by appending the list Add to $Req2$.

The expressiveness of this formalism is increased decisively by using shared variables for unification within the facts. This makes it possible to change certain parts of subconstituents and leave other parts intact. It also allows to define general rules that can be overwritten by more specific rules.

Example 8. The following rule states that a verb phrase with head word TSUNAGIAWASERU and any direct object X (`to put together from` X) is translated into a verb phrase with head word `zusammenfügen` and the prepositional object consisting of the phrase particle `aus` (`from`), number plural, determiner type indefinite, and X:

$$tr\_acc(\text{TSUNAGIAWASERU}/ver, [hew(\text{TSUNAGIAWASERU}/ver), dob(X)],\\ [hew(\texttt{zusammenfügen}/ver), pob([php(\texttt{aus}/prp), det(ind), num(plu)|X]).$$

If this transfer rule is applied to the verb phrase PAPIRUSU O NANMAI MO TSUNAGIAWASETA (`put together from several sheets of papyrus`), we obtain:

$$Req1 = [hew(\text{TSUNAGIAWASERU}/ver), dob(X)]\\ Req2 = [hew(\texttt{zusammenfügen}/ver), pob([php(\texttt{aus}/prp), det(ind), num(plu)|X]\\ A1 = [dob([hew(\text{PAPIRUSU}/nou), qua([hew(\text{MAI}/cou), php(\text{MO}/par),\\ amo(\text{NAN}/ipr)])]), hwf(vta), hew(\text{TSUNAGIAWASERU}/ver)]\\ Req1_s = [hew(\text{TSUNAGIAWASERU}/ver), dob([hew(\text{PAPIRUSU}/nou),\\ qua([hew(\text{MAI}/cou), php(\text{MO}/par), amo(\text{NAN}/ipr)])])]\\ Add = [hwf(vta)]\\ A2 = [hew(\texttt{zusammenfügen}/ver), pob([php(\texttt{aus}/prp), det(ind), num(plu),\\ hew(\text{PAPIRUSU}/nou), qua([hew(\text{MAI}/cou), php(\text{MO}/par), amo(\text{NAN}/ipr)])]),\\ hwf(vta)]$$

As can be seen, the variable X is bound to the head word PAPIRUSU and the complex constituent *qua* expressing a quantity. The quantity expression (`several sheets`) consists of the head word MAI (counter for thin objects like sheets), the phrase particle MO (`also`), and the interrogative pronoun (*ipr*) NAN (`what`).

One important use of this predicate is the translation of Japanese postpositional objects into corresponding German prepositional objects because the choice of the German preposition depends in most cases on the main verb and the postposition.

As for the case of a simple constituent, we also provide a predicate to not only change the argument of a complex constituent but also the constituent name:

$$tr\_cc(C1, C2, Hew, Req1, Req2). \tag{5}$$

This changes a complex constituent $C1(A1)$ to $C2(A2)$. $A1$ is defined as union of a set of required subconstituents $Req1$ and a set of optional subconstituents Opt: $A1 = Req1 \cup Opt$. $A2$ is then computed as union of the translation $Req2$ of the required subconstituents and Opt: $A2 = Req2 \cup Opt$. Again, Hew is used to speed up the rule access. Different from the unrestricted set Add in (4), Opt is limited to certain optional constituents, which do not change the translation of the rest of the phrase.

Example 9. The adverbial phrase KATAMEN NI (`on one side`) introduced in Example 7 is translated as predicative adjectival phrase (*pap*) with head word `einseitig` and comparison positive. An additional attributive suffix DAKE (`only`) as in KATAMEN DAKE NI is transferred unchanged to be translated in a second step as shown in Example 3:

$$tr\_cc(adp, pap, \text{KATAMEN}/nou, [php(\text{NI}/par), hew(\text{KATAMEN}/nou)],$$
$$[hew(\texttt{einseitig}/adj), com(pos)]).$$
$$A1 = [php(\text{NI}/par), hew(\text{KATAMEN}/nou), asf(\text{DAKE}/suf)]$$
$$Req1_s = [php(\text{NI}/par), hew(\text{KATAMEN}/nou)]$$
$$Opt = [asf(\text{DAKE}/suf)]$$
$$A2 = [hew(\texttt{einseitig}/adj), com(pos), asf(\text{DAKE}/suf)]$$

Of course, also facts for predicate (5) can contain shared variables for unification to enable the acquisition of general transfer rules.

Example 10. The Japanese adjectival noun NITA together with a comparative phrase (*cmp*) X forms an attributive adjectival phrase (*aap*) (`being similar to` X) that is translated into a relative clause (*rcl*) with head word `ähneln` (`to resemble`) in present tense and an indirect object (*iob*) X:

$$tr\_cc(aap, rcl, \text{NITA}/ano, [hew(\text{NITA}/ano), cmp(X)],$$
$$[hew(\texttt{ähneln}/ver), ten(prs), iob(X)].$$

3.3 Rules for Translating Conjunctions

German (or English) conjunctions are expressed in Japanese mainly with the help of conjunctive particles. However, the translation of a conjunctive particle often depends on the constituent name of the complex constituent in which it is included, i.e. the phrase type.

Therefore, we provide the following predicate for the definition of general transfer rules for situations where the argument $A1$ of a simple constituent is only translated to $A2$ if the constituent name of the complex constituent in which it is included equals CI:

$$tr\_sci(CI, A1, A2). \tag{6}$$

Example 11. The default transfer rule to translate the Japanese conjunctive particle TO (`and`) for combining a noun phrase with a coordinated noun phrase (*cnp*) is formulated as the fact:

$$tr\_sci(cnp, \text{TO}/par, \texttt{und}/con).$$

However, when TO is used to combine a clause with a preceding clause (*pcl*), the meaning changes to the conditional sense `wenn` (`if, when`):

$$tr\_sci(pcl, \text{TO}/par, \texttt{wenn}/con).$$

One particular characteristic of Japanese grammar is that there exist certain verb forms with conjunctional meaning to combine two clauses, e.g. the te-form. For this frequent situation we have defined the following predicate, which chooses a conjunction to insert into the preceding clause depending on the verb form and the head words of the two clauses:

$$tr\_cvf(Vf, Hew1, Hew2, Con). \tag{7}$$

This predicate is just provided for reasons of convenience and efficiency, it can also be realized with the help of predicate (4) applied to the main clause:

$$tr\_acc(Hew2, [hew(Hew2), pcl([hew(Hew1), hwf(Vf)|X])],\\ [hew(Hew2), pcl([hew(Hew1), hwf(Vf), php(Con)|X])]).$$

3.4 Rules for Translating Syntactic Features

One of the main problems with translating Japanese into German is the large discrepancy between these two languages with respect to their explicitness in expressing syntactic features at the surface level. German grammar has a complex system of declensions and conjugations to express number, gender, case, tense, mood, voice, etc. Japanese, however, is highly ambiguous regarding most of these features: it has no declension at all, does not distinguish between singular and plural, has no articles, and indicates only two tenses through conjugation. To bridge these two very different representations of linguistic knowledge, we have to find the right values for all the syntactic features required for the generation of the surface form of a German sentence.

Maybe the most difficult problem is to determine the referential properties of a Japanese noun phrase in order to generate the correct values for the syntactic features number and determiner type. In our translation model we use default values wherever possible, which are overwritten to cover special cases. To model the general situation that the determiner type and the number of a constituent with name C depend on its head word and on the head word of the constituent in which it is included, we provide the predicate:

$$tr\_dn(C, Hew1, Hew2, Det, Num). \tag{8}$$

Example 12. In the expression HIMO O TOOSU (`to thread a lace`) the direct object is an indefinite singular noun phrase:

$$tr\_dn(dob, \text{HIMO}/nou, \text{TOOSU}/ver, ind, sng).$$

As before, this rule is only a shortcut instead of writing:

$$tr\_acc(Hew2, [hew(Hew2), C([hew(Hew1)|X])],\\ [hew(Hew2), C([hew(Hew1), def(Det), num(Num)|X])]).$$

Rules of this type can again be overwritten by more specific rules. This way we can create a flexible hierarchy of rules reaching from the most general to the

most specific cases. For postpositional phrases the correct number and tense is determined in parallel with the translation of the postposition whenever possible.

Syntactic features regarding verbs in verb phrases, i.e. the correct tense, voice, mood, etc., are mainly derived from the verb form. For this purpose we provide a predicate to translate a verb form into a list of syntactic features:

$$tr\_vff(Vf, FL). \tag{9}$$

Except for the tense we use default values for the syntactic features and indicate only different values to keep the German parsing tree compact. As the information derived from the conjugation of the main verb is often ambiguous, in many cases the acquisition of additional, more specific rules is necessary.

4 Acquisition of Transfer Rules

For the automatic acquisition of new transfer rules from Japanese–German sentence pairs, we first compute both parsing trees as input to the acquisition component. The acquisition algorithm traverses both syntax trees in a top-down fashion. We start the search for new rules at the sentence level before we look for corresponding subconstituents to continue the search for finer-grained transfer rules recursively. The matching algorithm performs a complete traversal of the parsing trees, i.e. rules are learnt even if they are not needed for the translation of the Japanese sentence in order to extract as much information as possible from the example.

In addition to the predicates we have already described in Sect. 2 we use the predicate *find_opt_par*($Csub, A1, A2, Asub1, Asub2$) to search for the subconstituent with name $Csub$ in $A1$ and $A2$, and retrieve the arguments of $Csub(Asub1)$ and $Csub(Asub2)$; $Asub1$=`nil` if $Csub(\_) \notin A1$, $Asub2$=`nil` if $Csub(\_) \notin A2$.

In the following we give an overview of the steps we perform to match between complex constituents. Because of space limitations, we can only present the basic principles for the most common cases. To match between two *verb phrases* we:

- derive a transfer rule of type *tr_asc* for the main verb;
- derive a transfer rule of type *tr_acc* for subconstituents of which the translation depends on the main verb, e.g. to map a postpositional object to a prepositional object, and continue the matching recursively for the two subconstituents without adpositions;
- derive transfer rules of type *tr_cc* or *tr_sc* to map Japanese subconstituents to different German subconstituents (e.g. a predicative adverb to an adverbial phrase, an adverbial phrase to a predicative adjectival phrase, etc.); if possible, matching is continued recursively for the congruent parts of the two subconstituents;
- apply *find_opt_par* to search for corresponding subconstituents for subject, direct object, preceding clause, etc., and apply matching recursively to these subconstituents;
- derive transfer rules for conjunctions and syntactic features.

To match between two *adverbial phrases* we derive a transfer rule of type *tr_acc* to translate the postposition and continue matching recursively for both phrases without adpositions. Finally, to match between two *noun phrases* we:

- derive either a default transfer rule of type *tr_asc* for the head noun or a transfer rule of type *tr_acc* for specific translations of head/modifier combinations (e.g. head noun and attributive noun phrase);
- derive transfer rules of type *tr_cc* or *tr_sc* to map Japanese subconstituents to different German subconstituents (e.g. an attributive adjectival phrase to a relative clause); if possible, matching is continued recursively;
- apply *find_opt_par* to search for corresponding subconstituents for attributive verb phrase, attributive adjectival phrase, coordinated noun phrase, etc., and apply matching recursively to these subconstituents;
- derive transfer rules for conjunctions and syntactic features.

Each rule which is not already in the rule base is validated against the existing rules to resolve any conflicts resulting from adding the new rule. This resolution is achieved by making general rules more specific. The distinction between general rules for default situations and specific rules for exceptions is drawn according to the frequency of occurrence in the collection of sentence pairs translated in the past. This way we are independent from the chronological order of analyzing new examples, i.e. the rule acquisition is not affected if an exception is learnt before a general case. Figure 2 shows the rules that are learnt from the German translation of the Japanese sentence in Fig. 1 (the syntactic features for the subject correspond with the default values so that no new rule is derived).

5 Application of Transfer Rules

The transfer component traverses the Japanese syntax tree in a top-down fashion and searches for transfer rules to be applied. We always check first whether the conditions for more specific transfer rules are satisfied before applying more general rules. As explained in Sect. 3 transfer rules can also perform only partial translations of complex constituents, leaving some parts unchanged to be transformed later on. This flexible and robust approach requires that the transfer component is able to deal with parsing trees that contain mixed representations consisting of original Japanese parts, and German parts that were already translated. This mixture gradually turns into a fully translated German parsing tree. Figure 3 shows the application of the transfer rules from Fig. 2.

As in Sect. 4, we can only highlight the basic principles of the transfer algorithm. By making use of the generic predicates to manipulate complex constituents (see Sect. 2) we have defined the predicate *tf_arg*$(C, A1, A2)$ to translate the argument $A1$ of a constituent $C(A1)$ into $A2$. For simple constituents this involves just the application of rules of type *tr_asc*, for complex constituents we perform the following principal steps: find and apply transfer rules of type *tr_acc*, transfer rules of type *tr_asc* for the head word, and transfer rules for conjunctions and syntactic features; recursively call predicates for the translation of all subconstituents.

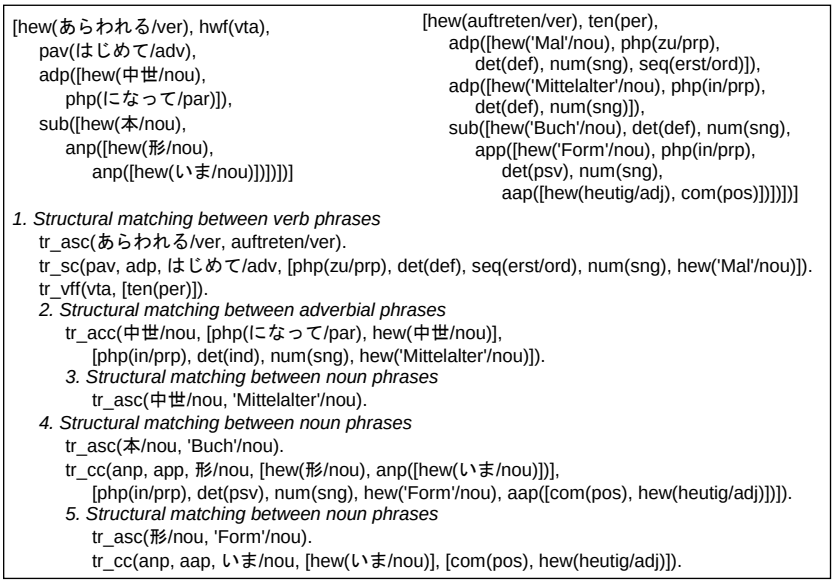

Fig. 2. Example of rule acquisition

The predicate $tf\_acc(A1, A2)$ is used for finding and applying transfer rules of type tr_acc; if no transfer rule can be applied, the constituent is left unchanged:

$$tf\_acc(A1, A2) :- find\_req(hew, A1, Hew), tr\_acc(Hew, Req1, Req2), split(A1, Req1, Add), append(Req2, Add, A2).$$
$$tf\_acc(A, A).$$

The recursive call for translating a subconstituent $Csub(Asub)$ is realized with the predicate $tf\_sub(Csub, A1, A2)$:

$$tf\_sub(Csub, A1, A2) :- find\_req(Csub, A1, Asub), tf\_sub\_arg(Csub, Asub, A1, A2).$$
$$tf\_sub(\_, A, A).$$

and the predicate $tf\_sub\_arg(Csub, Asub, A1, A2)$, which consists of several rules that either:

- find and apply a rule of type tr_sc: $tf\_sub\_arg(Csub, Asub, A1, A2) :- tr\_sc(Csub, Csub2, Asub, Asub2), repl\_diff(Csub, Csub2, A1, Asub2, A2)$.
- find and apply rules of type tr_cc (tf_cc is defined in a similar way to tf_acc): $tf\_sub\_arg(Csub, Asub, A1, A2) :- tf\_cc(Csub, Csub2, Asub, Asub2), repl\_diff(Csub, Csub2, A1, Asub2, A2)$.
- recursively call tf_arg for translating $Asub$: $tf\_sub\_arg(Csub, Asub, A1, A2) :- tf\_arg(Csub, Asub, Asub2), Asub$ \== $Asub2, replace(Csub, A1, Asub2, A2)$.
- otherwise leave the constituent unchanged: $tf\_sub\_arg(\_, \_, A, A)$.

[hew(あらわれる/ver), hwf(vta), pav(はじめて/adv), adp([hew(中世/nou), php(になって/par)]),
sub([hew(本/nou), anp([hew(形/nou), anp([hew(いま/nou)])])])]
1. tr_asc(あらわれる/ver, auftreten/ver).
[hew(auftreten/ver), hwf(vta), pav(はじめて/adv), adp([hew(中世/nou), php(になって/par)]),
sub([hew(本/nou), anp([hew(形/nou), anp([hew(いま/nou)])])])]
2. tr_vff(vta, [ten(per)]).
[hew(auftreten/ver), ten(per), pav(はじめて/adv), adp([hew(中世/nou), php(になって/par)]),
sub([hew(本/nou), anp([hew(形/nou), anp([hew(いま/nou)])])])]
3. tr_sc(pav, adp, はじめて/adv, [php(zu/prp), det(def), seq(erst/ord), num(sng), hew('Mal'/nou)]).
[hew(auftreten/ver), ten(per), adp([php(zu/prp), det(def), seq(erst/ord), num(sng), hew('Mal'/nou)]),
adp([hew(中世/nou), php(になって/par)]), sub([hew(本/nou), anp([hew(形/nou), anp([hew(いま/nou)])])])]
4. tr_acc(中世/nou, [php(になって/par), hew(中世/nou)], [php(in/prp), det(ind), num(sng), hew('Mittelalter'/nou)]).
[hew(auftreten/ver), ten(per), adp([php(zu/prp), det(def), seq(erst/ord), num(sng), hew('Mal'/nou)]),
adp([php(in/prp), det(ind), num(sng), hew('Mittelalter'/nou)]),
sub([hew(本/nou), anp([hew(形/nou), anp([hew(いま/nou)])])])]
5. tr_asc(本/nou,'Buch'/nou).
[hew(auftreten/ver), ten(per), adp([php(zu/prp), det(def), seq(erst/ord), num(sng), hew('Mal'/nou)]),
adp([php(in/prp), det(ind), num(sng), hew('Mittelalter'/nou)]),
sub([hew('Buch'/nou), det(def), num(sng), anp([hew(形/nou), anp([hew(いま/nou)])])])]
6. tr_cc(anp, app, 形/nou, [hew(形/nou), anp([hew(いま/nou)])],
[php(in/prp), det(psv), num(sng), hew('Form'/nou), aap([com(pos), hew(heutig/adj)])]).
[hew(auftreten/ver), ten(per), adp([php(zu/prp), det(def), seq(erst/ord), num(sng), hew('Mal'/nou)]),
adp([php(in/prp), det(ind), num(sng), hew('Mittelalter'/nou)]),
sub([hew('Buch'/nou), det(def), num(sng), app([php(in/prp), det(psv), num(sng), hew('Form'/nou),
aap([hew(com(pos), heutig/adj)])])])]

Fig. 3. Example of rule applications

6 Conclusion

In this paper we have presented a machine translation system, which automatically learns transfer rules from translation examples by using structural matching between parsing trees. We have completed the implementation of the system and are now in the process of creating a rule base of reasonable size with the assistance of several language students from our university. So far, their feedback regarding the usefulness of PETRA for their language studies has been very positive. After we have reached a certain level of linguistic coverage, future work will concentrate on a thorough evaluation of our system.

References

1. Hutchins, J., Somers, H.: An Introduction to Machine Translation. Academic Press (1992)
2. Newton, J., ed.: Computers in Translation: A Practical Appraisal. Routledge (1992)
3. Somers, H., ed.: Computers and Translation: A Translator's Guide. John Benjamins (2003)
4. Winiwarter, W.: Incremental learning of transfer rules for customized machine translation. Proc. of the 15th Intl. Conf. on Applications of Declarative Programming and Knowledge Management, Berlin, Germany (2004) 183–192
5. Winiwarter, W.: PETRA – the personal embedded translation and reading assistant. Proc. of the InSTIL/ICALL 2004 Symposium on NLP and Speech Technologies in Advanced Language Learning Systems, Venice, Italy (2004) 111–114