Tricolor DAGs for Machine Translation
Koichi Takeda
Tokyo Research Laboratory, IBM Research
1623-14 Shimotsuruma, Yamato, Kanagawa 242, Japan
Phone: 81-462-73-4569, 81-462-73-7413 (FAX)
takeda@trl.vnet.ibm.com

Abstract
Machine translation (MT) has recently been formulated in terms of constraint-based knowledge representation and unification theories, but it is becoming more and more evident that it is not possible to design a practical MT system without an adequate method of handling mismatches between semantic representations in the source and target languages. In this paper, we introduce the idea of "information-based" MT, which is considerably more flexible than interlingual MT or the conventional transfer-based MT.

Introduction
With the intensive exploration of contemporary theories on unification grammars[6, 15, 13] and feature structures[7, 19] in the last decade, the old image of machine translation (MT) as a brutal form of natural language processing has given way to that of a process based on a uniform and reversible architecture[16-1, 27].

The developers of MT systems based on the constraint-based formalism found a serious problem in "language mismatching," namely, the difference between semantic representations in the source and target languages.\(^1\) Attempts to design a pure interlingual MT system were therefore abandoned,\(^2\) and the notion of "semantic transfer"[24, 22] came into focus as a practical solution to the problem of handling the language mismatching. The constraint-based formalism[2] seemed promising as a formal definition of transfer, but pure constraints are too rigid to be precisely imposed on target-language sentences.

Some researchers(e.g., Russell[14]) introduced the concept of defeasible reasoning in order to formalize what is missing from a pure constraint-based approach, and control mechanisms for such reasoning have also been proposed[5, 3]. With this additional mechanism, we can formulate the "transfer" process as a mapping from a set of constraints into another set of mandatory and defeasible constraints. This idea leads us further to the concept of "information-based" MT, which means that, with an appropriate representation scheme, a source sentence can be represented by a set of constraints that implies and that, given a target sentence, the set \(C_0\) of constraints can be divided into three disjoint subsets:

- The subset \(C_0\) of constraints that is also implied by the target sentence
- The subset \(C_+\) of constraints that is not implied by, but is consistent with, the translated sentence
- The subset \(C_-\) of constraints that is violated by the target sentence

The target sentence may also imply another set \(C_{\text{new}}\) of constraints, none of which is in \(C_+\). That is, the set \(C_t\) of constraints implied by the target sentences is a union of \(C_0\) and \(C_{\text{new}}\), while \(C_+ = C_0 \cup C_+ \cup C_-\). When \(C_+ = C_0 = C_t\), we have a fully interlingual translation of the source sentence. If \(C_+ \neq \phi, C_- = \phi,\) and \(C_{\text{new}} = \phi\), the target sentence is said to be under-generated, while it is said to be over-generated when \(C_+ = \phi, C_- = \phi,\) and \(C_{\text{new}} \neq \phi.\)\(^3\) In either case, \(C_-\) must be empty if a consistent translation is required. Thus, the goal of machine translation is to find an optimal pair of source and target sentences that minimizes \(C_+, C_-\), and \(C_{\text{new}}\). Intuitively, \(C_0\) corresponds to essential information, and \(C_+\) and \(C_{\text{new}}\) can be viewed as language-dependent supportive information. \(C_-\) might be the inconsistency be-

1\(^1\)For example, Yasuhara[26] reported there was an overlap of only 10% between his group's English and Japanese concept dictionaries, which covered 0.2 million concepts.

2\(^2\)Even an MT system with a controlled input language[12] does not claim to be a pure interlingual system.

3\(^3\)The notions of completeness and coherence in LFG[6] have been employed by Wedekind[25] to avoid over- and under-generation.
tween the assumptions of the source- and target-language speakers.

In this paper, we introduce tricolor DAGs to represent the above constraints, and discuss how tricolor DAGs are used for practical MT systems. In particular, we give a generation algorithm that incorporates the notion of semantic transfer by gradually approaching the optimal target sentence through the use of tricolor DAGs, when a fully interlingual translation fails. Tricolor DAGs give a graph-algorithmic interpretation of the constraints, and the distinctions between the types of constraint mentioned above allow us to adjust the margin between the current and optimal solution effectively.

**Tricolor DAGs**

A tricolor DAG (TDAG, for short) is a rooted, directed, acyclic graph with a set of three colors (red, yellow, and green) for nodes and directed arcs. It is used to represent a feature structure of a source or target sentence. Each node represents either an atomic value or a root of a DAG, and each arc is labeled with a feature name. The only difference between the familiar usage of DAGs in unification grammars and that of TDAGs is that the color of a node or arc represents its degree of importance:

1. Red shows that a node (arc) is essential.
2. Yellow shows that a node (arc) may be ignored, but must not be violated.
3. Green shows that a node (arc) may be violated.

For practical reasons, the above distinctions are interpreted as follows:

1. Red shows that a node (arc) is derived from lexicons and grammatical constraints.
2. Yellow shows that a node (arc) may be inferred from a source or a target sentence by using domain knowledge, common sense, and so on.
3. Green shows that a node (arc) is defeasibly inferred, specified as a default, or heuristically specified.

When all the nodes and arcs of TDAGs are red, TDAGs are basically the same as the feature structures of grammar-based translation [25, 17]. A TDAG is well-formed iff the following conditions are satisfied:

4. Acyclicity is not crucial to the results in this paper, but it significantly simplifies the definition of the tricolor DAGs and semantic transfer.
5. We will only consider the semantic portion of the feature structure although the theory of tricolor DAGS for representing entire feature structures is an interesting topic.

1. The root is a red node.
2. Each red arc connects two red nodes.
3. Each red node is reachable from the root through the red arcs and red nodes.
4. Each yellow node is reachable from the root through the arcs and nodes that are red and/or yellow.
5. Each yellow arc connects red and/or yellow nodes.
6. No two arcs start from the same node, and have the same feature name.

Conditions 1 to 3 require that all the red nodes and red arcs between them make a single, connected DAG. Condition 4 and 5 state that a defeasible constraint must not be used to derive an imposed constraint. In the rest of this paper, we will consider only well-formed TDAGs. Furthermore, since only the semantic portions of TDAGs are used for machine translation, we will not discuss syntactic features.

The subsumption relationship among the TDAGs is defined as the usual subsumption over DAGs, with the following extensions:

- A red node (arc) subsumes only a red node (arc).
- A yellow node (arc) subsumes a red node (arc) and a yellow node (arc).
- A green node (arc) subsumes a node (arc) with any color.

The unification of TDAGs is similarly defined. The colors of unified nodes and arcs are specified as follows:

- Unification of a red node (arc) with another node (arc) makes a red node (arc).
- Unification of a yellow node (arc) with a yellow or green node (arc) makes a yellow node (arc).
- Unification of two green nodes (arcs) makes a green node (arc).

Since the green nodes and arcs represent defeasible constraints, unification of a green node (either a root of a TDAG or an atomic node) with a red or yellow node always succeeds, and results in a red or yellow node. When two conflicting green nodes are to be unified, the result is indefinite, or a single non-atomic green node.

Now, the problem is that a red node/arc in a source TDAG (the TDAG for a source sentence)
may not always be a red node/arc in the target TDAG (the TDAG for a target sentence). For example, the functional control of the verb “wish” in the English sentence

John wished to walk

may produce the TDAG1 in Figure 1, but the red arc corresponding to the agent of the *WALK predicate may not be preserved in a target TDAG2. This means that the target sentence alone cannot convey the information that it is John who wished to walk, even if this information can be understood from the context. Hence the red arc is relaxed into a yellow one, and any target TDAG must have an agent of *WALK that is consistent with *JOHN. This relaxation will help the sentence generator in two ways. First, it can prevent generation failure (or non-termination in the worst case). Second, it retains important information for a choosing correct translation of the verb “walk”.

Another example is the problem of identifying number and determiner in Japanese-to-English translation. This type of information is rarely available from a syntactic representation of a Japanese noun phrase, and a set of heuristic rules[11] is the only known basis for making a reasonable guess. Even if such contextual processing could be integrated into a logical inference system, the obtained information should be defeasible, and hence should be represented by green nodes and arcs in the TDAGs. Pronoun resolution can be similarly represented by using green nodes and arcs.

It is worth looking at the source and target TDAGs in the opposite direction. From the Japanese sentence,

ジョンは歩くことを望んだ。

we get the source TDAG3 in Figure 1, where functional control and number information are missing. With the help of contextual processing, we get the target TDAG4, which can be used to generate the English sentence “John wished to walk.”

Semantic Transfer

As illustrated in the previous section, it is often the case that we have to solve mismatches between source and target TDAGs in order to obtain successful translations. Syntactic/semantic transfer has been formulated by several researchers[18, 27] as a means of handling situations in which fully interlingual translation does not work. It is not enough, however, to capture only the equivalent relationship between source and target semantic representations: this is merely a mapping among red nodes and arcs in TDAGs. What is missing in the existing formulation is the provision of some margin between what is said and what is translated. The semantic transfer in our framework is defined as a set of successive operations on TDAGs for creating a sequence of TDAGs $t_0, t_1, \ldots, t_k$ such that $t_0$ is a source TDAG and $t_k$ is a target TDAG that is a successful input to the sentence generator.

A powerful contextual processing and a domain knowledge base can be used to infer additional facts and constraints, which correspond to the addition of yellow nodes and arcs. Default inheritance, proposed by Russell et al.[14], provides an efficient way of obtaining further information necessary for translation, which corresponds to the addition of green nodes and arcs. A set of well-known heuristic rules, which we will describe later in the “Implementation” Section, can also be used to add green nodes and arcs. To complete the model of semantic transfer, we have to introduce
A painter maps a red node to either a yellow or a green node, a yellow node to a green node, and so on. It is used to loosen the constraints imposed by the TDAGs. Every application of the painter monotonically loses some information in a TDAG, and only a finite number of applications of the painter are possible before the TDAG consists entirely of green nodes and arcs except for a red root node. Note that the painter never removes a node or an arc from a TDAG, it simply weakens the constraints imposed by the nodes and arcs.

Formally, semantic transfer is defined as a sequence of the following operations on TDAGs:

- Addition of a yellow node (and a yellow arc) to a given TDAG. The node must be connected to a node in the TDAG by a yellow arc.
- Addition of a yellow arc to a given TDAG. The arc must connect two red or yellow nodes in the TDAG.
- Addition of a green node (and a green arc) to a given TDAG. The node must be connected to a node in the TDAG by the green arc.
- Addition of a green arc to a given TDAG. The arc can connect two nodes of any color in the TDAG.
- Replacement of a red node (arc) with a yellow one, as long as the well-formedness is preserved.
- Replacement of a yellow node (arc) with a green one, as long as the well-formedness is preserved.

The first two operations define the logical implications (possibly with common sense or domain knowledge) of a given TDAG. The next two operations define the defeasible (or heuristic) inference from a given TDAG. The last two operations define the painter. The definition of the painter specifies that it can only gradually relax the constraints. That is, when a red or yellow node (or arc) X has other red or yellow nodes that are only connected through X, X cannot be “painted” until each of the connected red and yellow nodes is painted yellow or green to maintain the reachability through X.

In the sentence analysis phase, the first four operations can be applied for obtaining a source TDAG as a reasonable semantic interpretation of a sentence. The application of these operations can be controlled by “weighted abduction”[5], default inheritance, and so on. These operations can also be applied at semantic transfer for augmenting the TDAG with a common sense knowledge of the target language. On the other hand, these operations are not applied to a TDAG in the generation phase, as we will explain in the next section. This is because the lexicon and grammatical constraints are only applied to determine whether red nodes and arcs are exactly derived. If they are not exactly derived, we will end up with either over- or under-generation beyond the permissible margin. Semantic transfer is applied to a source TDAG as many times as necessary until a successful generation is made. Recall the sample sentence in Figure 1, where two painter calls were made to change two red arcs in TDAG1 into yellow ones in TDAG2. These are examples of the first substitution operation shown above. An addition of a green node and a green arc, followed by an addition of a green arc, was applied to TDAG3 to obtain TDAG4. These additions are examples of the third and fourth addition operations.

Sentence Generation Algorithm

Before describing the generation algorithm, let us look at the representation of lexicons and grammars for machine translation. A lexical rule is represented by a set of equations, which introduce red nodes and arcs into a source TDAG.10 A phrasal rule is similarly defined by a set of equations, which also introduce red nodes and arcs for describing a syntactic head and its complements.

For example, if we use Shieber’s PATR-II[15] notation, the lexical rule for “wished” can be represented as follows:

\[
V \leftarrow \text{wished} \\
(V \text{ cat}) = v \\
(V \text{ form}) = \text{past} \\
(V \text{ subj cat}) = np \\
(V \text{ obj cat}) = v \\
(V \text{ obj form}) = \text{infinitival} \\
(V \text{ pred}) = \text{*WISH} \\
(V \text{ pred agent}) = (V \text{ subj pred}) \\
(V \text{ pred theme}) = (V \text{ obj pred}) \\
(V \text{ pred theme agent}) = (V \text{ subj pred})
\]

The last four equations are semantic equations. Its TDAG representation is shown in Figure 2. It would be more practical to further assume that such a lexical rule is obtained from a type inference system,11 which makes use of a syntactic class hierarchy so that each lexical class can inherit general properties of its superclasses. Similarly, semantic concepts such as *WISH and *WALK should be separately defined in an ontological hierarchy together with necessary domain knowledge (e.g., selectional constraints on case
Figure 2: TDAG representation of the verb "wished" (embedded in the entire feature structure)

Figure 3: Source TDAG for the sentence "The Boston Office called."

Fillers and part-of relationships. See KBMT-89[8].

A unification grammar is used for both analysis and generation. Let us assume that we have two unification grammars for English and Japanese. Analyzing a sentence yields a source TDAG with red nodes and arcs. Semantic interpretation resolves possible ambiguity and the resulting TDAG may include all kinds of nodes and arcs. For example, the sentence[12]

The Boston office called

would give the source TDAG in Figure 3. By utilizing the domain knowledge, the node labeled *PERSON is introduced into the TDAG as a real caller of the action *CALL, and two arcs representing *PERSON work-for *OFFICE and *OFFICE in *BOSTON are abductively inferred.

Our generation algorithm is based on Wedekind’s DAG traversal algorithm[25] for LFG. The algorithm runs with an input TDAG by traversing the nodes and arcs that were derived from the lexicon and grammar rules. The termination conditions are as follows:

1. Apply the painter to change the yellow of the definite node and the def arc to green.
2. Apply the painter to change the yellow of the singular node and the num arc to green. The resulting TDAG is shown in Figure 4.
3. Run the sentence generator with an input feature structure, which has a root and an arc pred connecting to the given TDAG. (See the node marked “1” in Figure 4.)
4. The generator applies a phrasal rule, say S → NP VP, which derives the subj arc connecting to the subject NP (marked “2”), and the agent arc.
5. The generator applies a phrasal rule, say NP → MOD NP, which derives the npmod arc to the

These conditions are identical to those of Wedekind except that yellow (or green) nodes and arcs may or may not be derived. For example, the sentence "The Boston Office called" in Figure 3 can be translated into Japanese by the following sequence of semantic transfer and sentence generation.

1. Apply the painter to change the yellow of the definite node and the def arc to green.
2. Apply the painter to change the yellow of the singular node and the num arc to green. The resulting TDAG is shown in Figure 4.
3. Run the sentence generator with an input feature structure, which has a root and an arc pred connecting to the given TDAG. (See the node marked “1” in Figure 4.)
4. The generator applies a phrasal rule, say S → NP VP, which derives the subj arc connecting to the subject NP (marked “2”), and the agent arc.
5. The generator applies a phrasal rule, say NP → MOD NP, which derives the npmod arc to the

There are several phrasal rules for deriving this LHS NP in Japanese: (1) A noun-noun compound, (2) a noun, copula, and a noun, and (3) a noun, postpositional particle, and a noun. These three rules roughly correspond to the forms (1) Boston Office, (2) office of Boston, and (3) office in Boston. Inference of the "*OFFICE in *BOSTON" relation is easiest if rule (3)
modifier of the NP (marked “3”) and the mod arc.

6. Lexical rules are applied and all the semantic nodes, *CALL, *OFFICE, and *BOSTON are derived.

The annotated sample run of the sentence generator is shown in Figure 5. The input TDAG in the sample run is embedded in the input feature structure as a set of PRED values, but the semantic arcs are not shown in the figure. The input feature structure has syntactic features that were specified in the lexical rules. The feature value "UNDEFINED" is used to show that the node has been traversed by the generator.

The basic property of the generation algorithm is as follows:

Let *t* be a given TDAG, *t* min be the connected subgraph including all the red nodes and arcs in *t*, and *t* max be the connected subgraph of *t* obtained by changing all the colors of the nodes and arcs to red. Then, any successful generation with the derived TDAG *t* g satisfies the condition that *t* min subsumes *t* g and *t* g subsumes *t* max.

The proof is immediately obtained from the definition of successful generation and the fact that the generator never introduces a new node or a new arc into an input TDAG. The TDAGs can also be employed by the semantic head-driven generation algorithm[17] while retaining the above property. Semantic monotonicity always holds for a TDAG, since red nodes must be connected. It has been shown by Takeda[21] that semantically non-monotonic representations can also be handled by introducing a functional semantic class.

**Implementation**

We have been developing a prototype English-to-Japanese MT system, called Shalt2[22], with a lexicon for a computer-manual domain including about 24,000 lexemes each for English and Japanese, and a general lexicon including about 50,000 English words and their translations. A sample set of 736 sentences was collected from the "IBM AS/400 Getting Started" manual, and was tested with the above semantic transfer and generation algorithm. The result of the syntactic analysis by the English parser is mapped to a TDAG using a set of semantic equations ob-

---

15We used McCord's English parser based on his English Slot Grammar[10], which covered more than 93% of the sentences.
16We call such a set of semantic equations mapping rules (see Shalt2[20] or KBMT-89[8]).
tained from the lexicons. We have a very shallow knowledge base for the computer domain, and no logical inference system was used to derive further constraints from the given source sentences. The Japanese grammar is similar to the one used in KBMT-89, which is written in pseudo-unification equations, but we have added several new types of equation for handling coordinated structures. The Japanese grammar can generate sentences from all the successful TDAGs for the sample English sentences.

It turned out that there were a few collections of semantic transfer sequences which contributed very strongly to the successful generation. These sequences include:

- Painting the functional control arcs in yellow.
- Painting the gaps of relative clauses in yellow.
- Painting the number and definiteness features in yellow.
- Painting the passivization feature in green.\(^1\)

Other kinds of semantic transfer are rather idiosyncratic, and are usually triggered by a particular lexical rule. Some of the sample sentences used for the translations are as follows:\(^1\)

Make sure you are using the proper edition for the level of the product.

Publications are not stocked at the address given below.

This publication could contain technical inaccuracies or typographical errors.

The overall accuracy of the translated sentences was about 63%. The main reason for translation errors was the occurrence of errors in lexical and structural disambiguation by the syntactic/semantic analyzer. We found that the accuracy of semantic transfer and sentence generation was practically acceptable.

Though there were few serious errors, some occurred when a source TDAG had to be completely “paraphrased” into a different TDAG. For example, the sentence

Let’s get started.

was very hard to translate into a natural Japanese sentence. Therefore, a TDAG had to be paraphrased into a totally different TDAG, which is another important role of semantic transfer. Other serious errors were related to the ordering of constituents in the TDAG. It might be generally acceptable to assume that the ordering of nodes in a DAG is immaterial. However, the different ordering of adjuncts sometimes resulted in a misleading translation, as did the ordering of members in a coordinated structure. These subtle issues have to be taken into account in the framework of semantic transfer and sentence generation.

Conclusions

In this paper, we have introduced tricolor DAGs to represent various degrees of constraint, and defined the notions of semantic transfer and sentence generation as operations on TDAGs. This approach proved to be so practical that nearly all of the source sentences that were correctly parsed were translated into readily acceptable sentences. Without semantic transfer, the translated sentences would include greater numbers of incorrectly selected words, or in some cases the generator would simply fail.\(^2\)

Extension of TDAGs for disjunctive information and a set of feature structures must be fully incorporated into the framework. Currently only a limited range of the cases are implemented. Optimal control of semantic transfer is still unknown. Integration of the constraint-based formalism, defeasible reasoning, and practical heuristic rules are also important for achieving high-quality translation. The ability to process and represent various levels of knowledge in TDAGs by using a uniform architecture is desirable, but there appears to be some efficient procedural knowledge that is very hard to represent declaratively. For example, the negative determiner “no” modifying a noun phrase in English has to be procedurally transferred into...

\(^1\)We decided to include the passivization feature in the semantic representation in order to determine the proper word ordering in Japanese.

\(^2\)Japanese translation reflects the errors made in English analysis. For example, the auxiliary verb “could” is misinterpreted in the last sample sentence.
the negation of the verb governing the noun phrase in Japanese. Translation of “any”, “yet”, “only”, and so on involves similar problems.

While TDAGs reflect three discrete types of constraints, it is possible to generalize the types and so on involves similar problems. This approach will provide a considerably more flexible margin that defines a set of permissible translations, but it is not clear whether we can successfully define a numeric value for each lexical rule in order to obtain acceptable translations.

Acknowledgments

The idea of the tricolor DAGs grew from discussions with Shiho Ogino on the design and implementation of the sentence generator. I would also like to thank the members of the NL group - Naohiko Uramoto, Tetsuya Nasukawa, Hiroshi Maruyama, Hiroshi Nemiyama, Hideo Watanabe, Masayuki Morohashi, and Taijiro Tsutsuri - for stimulating comments and discussions that directly and indirectly contributed to shaping the paper. Michael McDonald, who has always been the person I turn to for proofreading, helped me write the final version.

References


