System Architecture and Control in the Multra System

Björn Beskow
Uppsala

Abstract

This paper discuses the system architecture and control in the Multra system. The Multra system is briefly described, and its modular architecture is discussed. The control in the system is divided into global and module-internal control. In the inter-modular control, a *blackboard architecture* is introduced to control the interaction and synchronization of the modules. The blackboard architecture is also shown to enable parallel solutions. In the intra-modular control, the *specificity principle* is introduced. Its relation to subsumption is discussed, and the principle is shown to provide a declarative way to control interaction between linguistic rules. Finally, the preference formalism is presented, used to express preferences between analysis results.

Introduction

The MULTRA system

The MULTRA system is a prototype of a multilingual computer support for translation and writing, and has been developed within the project *Multilingual Support for Translation and Writing* at Uppsala University (see Sågvall Hein (1993)). One of the functionalities of the Multra system is machine translation. The user, working in an interactive document processing environment can mark a region of the document and have it translated on the fly. The region can, in principle, range from a single word to the whole document.

The Multra machine translation component is transfer-based. Translation is performed on a sentence level, but exploiting the type information provided by the document representation format. The translation is performed by four independent modules (see figure 1), responsible for analysis of the source language, preference ordering of the analysis results, transfer, and synthesis of the target language. An attribute-value logic is the common representation formalism for all the modules.

The different modules are implemented as separate Unix processes, communicating through TCP/IP sockets. The processes can thus execute on different machines or on different processors on the same machine.

Proceedings of NODALIDA 1993, pages 41-49

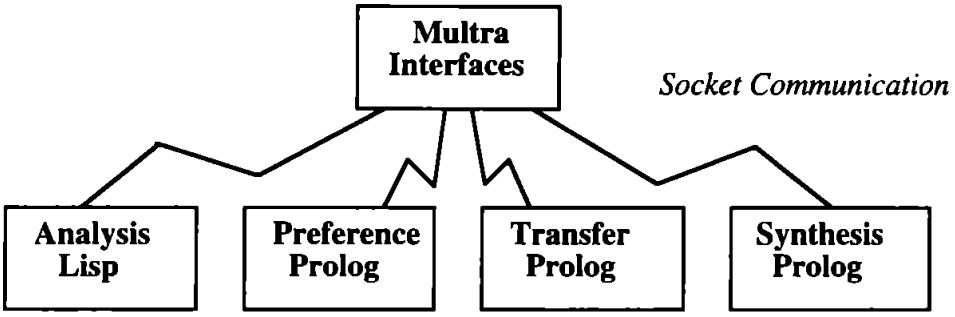

FIG. 1 : Multra System Architecture

Logic and Control

The four main modules implement the global logic of the system. A set of rules implements the module internal logic for each module.[1] The task of *control* is to specify the interaction between the parts, both on a global and a module-internal level. The logic of the Multra system has been described elsewhere.[2] In the following sections, we will discuss the intra-modular and inter-modular control in the Multra system.

Inter-Modular Control

As we have seen, the modules in the Multra system are fully autonomous. However, the result of a module may form input to another module, resulting in a sequential information flow through the system. Because of the modules being fully autonomous, they may however very well execute in parallel. For instance, the parser starts by parsing the first segment. When ready, the parser output constitute input to the preference machine. The parser may however start parsing the next segment without having to wait for the other modules to process the first segment. The same holds for all modules.

The inter-modular control must therefore enable the sequential flow of information through the system by providing a communication channel between the modules, and by synchronizing the work of the modules. This control is achieved by using a *Blackboard*.[3] The blackboard is a common data area accessible by all modules.

---

[1]The set of rules within a module is conceptually divided into general and domain specific rules.

[2]See e.g. Beskow (1992; 1993a; 1993b) and Sågvall Hein (1987; 1993a; 1993b).

[3]Blackboard systems have mainly been used to provide data-driven processing, to integrate information from many different sources and to have several competing threads working on the same problem. A blackboard architecture is however also very suitable for controlling interactions between modules, to synchronize modules and to exploit parallellism. See Linda (1988) for a discussion of Blackboard systems.

In the Multra system, each module can read or write first order terms on the blackboard. A module is triggered by its own special term providing its input. When the module has completed its processing, it writes its own special result term back on the blackboard and then waits for the next input term. This situation is illustrated in figure 2 below.

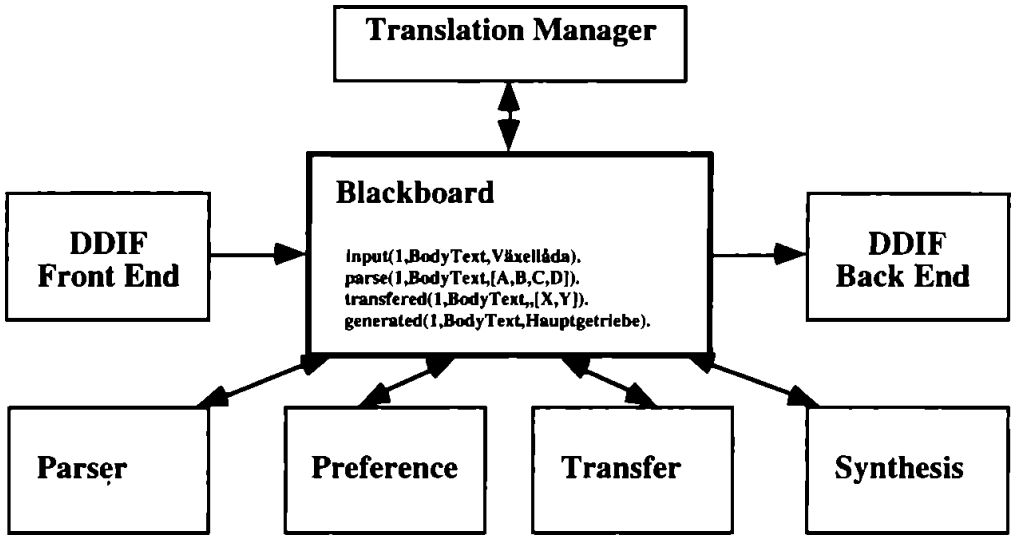

FIG. 2 : Blackboard

The parser reads terms of the form *input(N,Type,String)*, and writes terms of the form *parsed(N,Type,ParseSet)*. The preference machine reads terms of the form *parsed(N,Type,ParseSet)* and writes terms of the form *preferred(N,Type,ParseList)*. Each term has as its first argument the segment number of the processed segment. This enables a sequential flow of information through the system, in spite of the parallel nature or the processing. The synchronization of the modules is automatically achieved through the Blackboard system.

Since the modules are fully autonomous, only communicating via the blackboard, multiple instances of a module may very well exist and execute in parallel. Hence it is possible to use several instances of a module to perform a computationally heavy task. This situation is illustrated in figure 3.

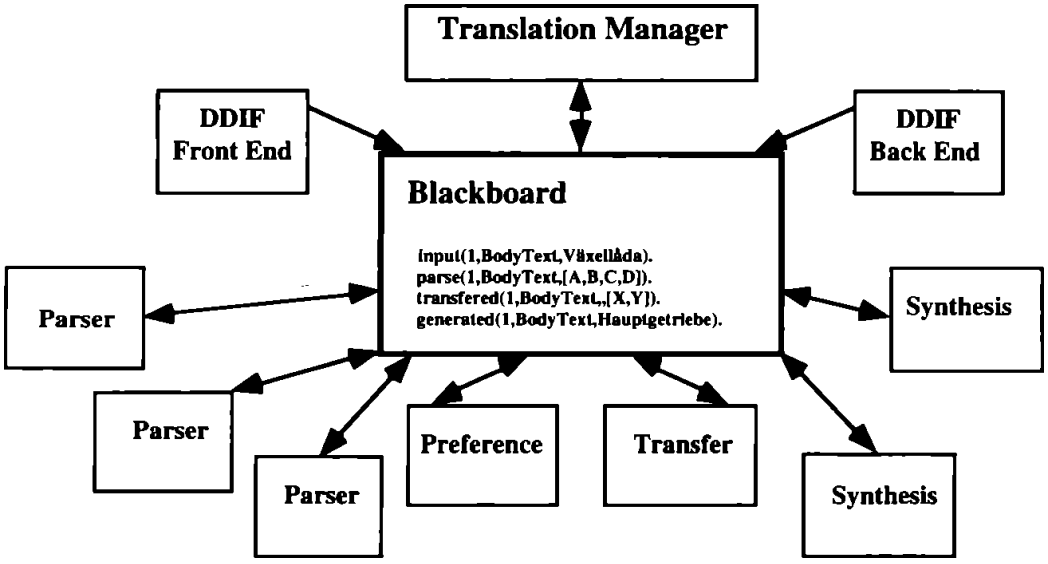

FIG. 3 : Multiple instances of modules

In fact, it is even possible to dynamically assign resources to tasks, within a *Processor Farm* model. The global controller can act as a 'farmer', having a number of processors or 'workers' under his command. The relative number of pending input terms on the blackboard for a certain module constitutes a work load measurement, measuring how heavy a certain task is. The farmer may dynamically assign more workers to a heavy task to maximize the efficiency. Labour division orders are just special control terms, written on the blackboard by the farmer.

Intra-Modular Control

Now we shall turn our attention to the module-internal control in the Multra system. The key concept here is the notion of *specificity*. The general idea is that more specific solutions should block or precede other more general solutions. A more specific translation should be preferred before a more general translation.

In terms of attribute-value logic, the subsumption relation forms a partial information ordering on attribute-value structures. Since the rule formalisms in all the modules are based on attribute-value logic, subsumption can be used to define specificity orders on rule sets. In a logical framework, the specificity principle may then be defined in terms of specificity between rules: Prefer a (constructive) translation proof based on more specific rules before a translation proof based on more general rules.

Let us look at an example. A transfer rule in the Multra formalism consists of two feature structures describing the source and target structures, and a (possibly empty) set of recursive transfer equations on

subparts of the source and target structures.[1] Consider the two transfer rules in figures 4 and 5 below, describing transfer relations between Swedish and German:

```
Label NOUN.OBJ
Source
 <* NOUN.OBJ> = ?NOUN.OBJ1
Target
 <* NOUN.OBJ> = ?NOUN.OBJ2
Transfer
 ?NOUN.OBJ1 <=> ?NOUN.OBJ2
```

FIG. 4 : Transfer rule 1

```
Label NOUN.OBJ_PP-NP
Source
 <* NOUN.OBJ PHR.CAT> = PP
 <* NOUN.OBJ PREP LEX> = AV1.PP.4
 <* NOUN.OBJ RECT> = ?RECT1
Target
 <* NOUN.OBJ> = ?NOUN.OBJ2
 <* NOUN.OBJ CASE> = GENITIVE
Transfer
 ?RECT1 <=> ?NOUN.OBJ2
```

FIG. 5 : Transfer rule 2

Rule 1 is a general rule, saying that in general, a noun object should be translated compositionally. Rule 2 is more specific, saying that a noun object that is a preposition phrase with the Swedish preposition 'av' should be translated into a genitive construction in German. The source attribute-value structure of rule 1 subsumes the source attribute-value structure of rule 2, hence rule 2 is considered more specific than rule 1. A translation based on rule 2 should be preferred before a translation based on rule 1.

[1] See Beskow (1993a) for a description of the Multra transfer formalism. Examples of complex transfer relations described in the formalism can be found in Sågvall Hein (1993b) and in Wikholm (1992).

The Preference Machine

The preference machine in the Multra system takes as input a set of attribute-value structures. This set represents the different analyses for a sentence produced by the parser.[1] If the set contains more than one element, the sentence is ambiguous. The task of the preference machinery is to compute a preference ordering on this set, returning a list with the attribute-value structures partially sorted.

The Multra preferences are defined by a set of preference rules. A preference rule defines a binary preference relation between two attribute-value structures. The set of preference rules thus defines a weak order on the set of attribute-value structures.[2] A preference rule consists of two attribute-value structures *Minor* and *Major*, representing the preferred and the dispreferred analysis result.

Figure 6 below is a simple example of a preference rule. It defines the preference relation between two attribute-value structures having different values for the attribute-value 'NUMB'. It says that the structure with value 'SING' is preferred before the structure with value 'NUMB'.

```
Preference SING-PLUR
     <* NUMB> = SING
precedes
     <* NUMB> = PLUR
```

FIG. 6 : Preference rule 1

Figure 7 below is an example of a preference rule that defines the preference relation between two attribute-value structures both having the value NP for the attribute-value 'PHR.CAT', but only the first one has the attribute-value 'POST.ATTR' defined.

```
Preference POST.ATTR
     <* PHR.CAT> = NP
     <* POST.ATTR> = ANY
precedes
     <* PHR.CAT> = NP
```

FIG. 7 : Preference rule 2

---

[1]For a description of the Multra parser, see Sågvall Hein (1987).

[2]The preference order is indeed a partial order of equivalence classes of feature structures, which correspond to a weak order (see e.g. Berge (1962) or Ore (1963)).

As all rules in the Multra system, the preference rules are themselves ordered by the specificity principle. A preference rule r is more specific than another rule r', written r $\leq_{prefrule}$ r', if and only if Minor(r') subsumes Minor(r).

The preference relation has the following semantics:

Let ϕ and ψ be two attribute-value structures, P a set of preference rules, and '$\leq_{pref}$' the preference ordering symbol. ϕ is preferred before ψ, written $\phi \leq_{pref} \psi$, if and only if

* there exists a preference rule r such that
  - Minor(r) subsumes ϕ and
  - Major(r) subsumes ψ and
  - for all r':
    if Major(r') subsumes ϕ and Minor(r') subsumes ϕ then
    r $\leq_{prefrule}$ r',

  or
* there exists a path p in both ϕ and ψ such that
  - $\delta(p,\phi) = \phi'$ and
  - $\delta(p,\psi) = \psi'$ and
  - $\phi' \leq_{pref} \psi'$

Consider the set of attribute-value structures in figure 8 below:

$$\left\{ a = \left[\begin{array}{l} CAT:NP \\ NUM:SING \\ DEF:INDEF \\ HEAD:\left[\begin{array}{l} LEX:VÄXELLÅDSHUS \\ WORD:CAT:NOUN \end{array} \right] \end{array} \right], \; b = \left[\begin{array}{l} CAT:NP \\ NUM:PLUR \\ DEF:INDEF \\ HEAD:\left[\begin{array}{l} LEX:VÄXELLÅDSHUS \\ WORD:CAT:NOUN \end{array} \right] \end{array} \right] \right\}$$

FIG. 8 : Singular/plural ambiguity

The two attribute-value structures represent the two possible readings for the Swedish noun 'växellådshus': one singular and one plural reading. If we take as our set of preference rules to be the rules in figure 6 and 7, we can see how they define a weak order on the attribute-value structures in figure 8. We can see that a $\leq_{pref}$ b holds, because of rule 1 whose Minor structure subsumes a, and whose Major structure subsumes b.

Digression: Non-existence of attribute-values

The preference formalism presented above has an interesting property: it allows for implicit non-existence conditions of attribute-values. Identity equation constraints used for describing attribute-value structures can only express positive constraints on the attribute-value structure being described. It is not possible in an identity equation to say that a certain attribute-value must not be defined, or must not have a certain value. It has been much discussed whether negative values are necessary to gain enough expressive power.[1]

The Multra preference formalism allows for an implicit way of stating negative attribute-value conditions. We have already seen a rule (in figure 7 above) which exploits this property. Consider the general reformulation of such a rule below:

```
Preference Non-existence
    <* F> = ANY
precedes
    <*> = ANY
```

FIG. 9 : Non-existence condition rule example

Consider further the two attribute-value structure pairs below:

$$
\begin{array}{lcl}
a & = & [F{:}A] \\
b & = & [\,] \\
a' & = & [F{:}A] \\
b' & = & [F{:}B]
\end{array}
$$

FIG. 10 : Non-existence condition structures example

We can see that a $\leq_{pref}$ b must hold because of the preference rule in figure 9. We can also see that a' $\leq_{pref}$ b' holds, by virtue of the same rule. However, we also find that b' $\leq_{pref}$ a' using the same rule. Since a partial order is asymmetric, it follows that a' $\equiv$ b', that is, they belong to the same equivalence class according to the preference ordering.

The example above shows how a preference rule implicitly can express a negative attribute-value condition. The rule in figure 9 above says that a attribute-value structure with the attribute F defined precedes a structure that does not have the attribute F defined.

[1] See for example Eisele & Dörre (1988).

Summary and conclusions

In this paper, the system architecture and control in the Multra system have been discussed. We have seen how a strictly modular system design enables the exploitation of parallelism. A blackboard architecture may be used to control both the global interaction between modules and the synchronization of parallel threads. The module-internal control has also been discussed. The notion of specificity has been introduced, and its relation to subsumption shown. The preference machine has also been presented. We have seen that the intra-modular control in multra is based on declarative notions and formalisms within the unification-based paradigm. The control defines and computes partial orders on attribute-value structures and on rule sets. I hope to have shown that the control mechanism of the Multra system provides a both elegant and efficient way of handling interaction on different levels.

References

Berge, C. 1962. *The Theory of Graphs and its Applications*. Methuen & Co Ltd.

Beskow, B. 1992. *Unifieringsbaserad Transfer*. Masters Thesis, Gothenburg University.

Beskow, B. 1993a. *Unification-Based Transfer. Multilingual Support for Translation and writing*. Forthcoming. Uppsala University.

Beskow, B. 1993b. *Generation in the Multra System*. Uppsala University.

Eisele, A. and J. Dörre. 1988. *Unification of Disjunctive Attribute-value Descriptions*. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*.

Ore, O. 1963. *Graphs and their uses*. Random House.

Shieber, S. 1986. *An Introduction to Unification-based Approaches to Grammar*. CSLI Lecture Notes.

Sågvall Hein, A. 1987. *Parsing by Means of Uppsala Chart Processor*. In Bolc, L. (ed.), *Natural Language Parsing Systems*. Berlin.

Sågvall Hein, A. 1993a. *Multilingual Support for translation and Writing. MULTRA*. Project report, NUTEK/HSFR Language Technology Program.

Sågvall Hein, A. 1993b. *On The Translation of Nominal Expressions in a Multilingual Unification Based Setting*. In Hajicova, E. (ed.) *Proceedings of the Functional Description of Language*. Prague 1993. pp. 209-224.

Wikholm, E. 1992. *Schwedish-deutsche lexikalische transferregeln. Nicht-flektierbare funktionale Phrasen*. Project report. Uppsala University.