NEW DIRECTIONS IN MT SYSTEMS: A CHANGE IN PARADIGM.

Margaret KING
ISSCO and ETI, University of Geneva
Geneva
Switzerland

There are essentially two aspects to a machine translation system; the linguistics which provides the intellectual basis for the treatment of the languages concerned and of the relation between them, and the software which puts computational constraints on the expression of the linguistic treatment. In this contribution I want to concentrate on the latter, although clearly in practice the two cannot be so sharply distinguished, if only because of their strong mutual influence.

In the very early machine translation systems, indeed, it was quite impossible to separate out these two aspects. Instructions on how to get from the source language to a translation were directly programmed in some low-level programming language, producing huge amounts of code that was impossible to decipher and dangerous to modify. A limit on improveability of the system is reached when attempts to correct one mistake succeed, but lead to new mistakes where before the system's behaviour was correct.

It was this problem which led to a first major shift in computational paradigm. System designers, becoming aware of the complexity of the linguistic descriptions necessary to achieve adequate translations, began to produce high-level languages specially designed as a medium for the expression of linguistic facts. It became much easier to structure a linguistic description well, so that, in turn, both understanding the description and modifying it became easier. (It is of course no accident that this development was taking place soon after computer scientists had also discovered the utility of high level languages and soon after the first impact of formal syntax had begun to be felt).

There is no doubt that this first shift in paradigm was of major importance, and contributed greatly to the successful development of systems like Taum-Meteo, the family of Grenoble systems, the first Mu project in Japan and several others. Nonetheless, linguists working on the linguistic descriptions discovered that the conception of the high-level languages forced upon them a problem of procedurality. Naturally enough, given the time at which the languages were being defined, most took tree-structures and their manipulation as the most natural way for a linguist to express himself. Thus, in effect, the linguist found himself describing a series of tree-to-tree transformations aimed at producing a canonical structure of some sort, (most often a dependency grammar representation of the input). Throwing all the rules describing the transformations into one big pot and allowing them to interact freely not only leads to programs

which cannot be guaranteed to terminate but can also lead to wrong results: if a canonical structure is aimed at and it has to be achieved by successive transformations of a tree structure, some transformations have to be ordered in order to produce the correct results. (And once again, it is no accident that transformational grammarians were discovering the same problem at the same time).

Initially, several attempts were made to solve or at least reduce the problems by methods which might be described as patching: leaving the fundamental design the same, but sticking some extra control mechanism over the top or imposing a constraint to guarantee termination. More recently, a radical change in the way of thinking about how to describe a language can be discerned, which may well, I believe, constitute a new fundamental shift in paradigm.

The radical change is to insist on declarativity and monotonicity as being of fundamental importance. A linguist who is provided with a declarative language in which to express himself no longer has to worry about the order in which operations will be carried out. The easiest way to think of it is to think of the system gradually accumulating more and more constraints on what a correct solution would be and producing the solution only when all the constraints are known rather than working step by step towards a solution, producing at each step one or more incomplete solutions some of which may turn out to be wrong and have to be abandoned. The basic idea is, of course, not confined to the world of machine translation. It is the basis also of logic programming languages and familiar too from recent work in generative linguistics, as well as having an obvious connection with the old declarative/procedural debate in artificial intelligence. The main benefit of using a monotonic system is obvious: a linguistic description can be constructed incrementally, is faster to construct, easier to understand and easier to maintain. The relative speed with which a description can be produced also allows the linguist increased freedom to experiment with different solutions to the real problems of translation.

So far as I know, work on declarative systems in machine translation is currently limited to research groups. (They can usually be spotted by their use of unification based formalisms or of logic programming languages). Although no-one would want to claim that any ultimate solution has been found - especially since there are so many fundamental problems to whose solution any shift in computational paradigm is simply irrelevant - experience so far does seem to justify optimism about the benefits to be gained from this new direction.