

NOTES ON SESSION 2 (03.11.2008)

We started the session with Benni telling us some details about the notion of computation and how it might, and actually is, implemented into particular sorts of automata. We talked about three main categories of these computing devices:

- (1) *The Finite-State-Automaton*
- (2) *The Push-Down-Automaton*
and the famous
- (3) *Turing Machine*

Why is all this relevant to linguists? Answer: Modelling things might provide us with insights as to the nature of the modelled entity.¹ Maybe it will turn out that language is not that organized a phenomenon and therefore not to be modelled by any means known to computer linguists.²

As to (1) Benni exemplified this kind of computing machine via an example from morphology: Imagine you would like to add some kind of prefix or suffix to a (free) root, such as adding the comparative *-er* to an adjectival root, e.g. *rich* → *rich-er*. Now, say you have an initial state (IS), an intermediate state (IM) and a final state (FS). In (IS), *rich* starts out, remains unchanged and is carried over to (IM) where the adjective undergoes an operation: (IM: + *-er*). This then yields the output *rich-er* for (FS). This might look like a nice device to put together comparative forms in a language; however there are some problems with Finite-State-Automata:

(i) As Richard correctly remarked: **a lot of exceptions** would have to be written into the computing device in order to be able to account for all the irregularities in the English morphology of comparison.

(ii) This is a relatively **stative automaton**. All it does is that it works on a terminal symbol and, essentially, cannot “look back”. It simply has no working memory. If there are phenomena which are sensitive across boundaries (this is, I believe a notion of minimal context sensitivity), then this

automaton will not be able to yield the right output.³ What we need is a memory stack that would enable the automaton to deal with a minimal embedding structure.

(2) can do exactly this. These automata have a memory programmed. This enables the automaton to retrieve information after each step in a computation. Remember (IS), (IM) and (FS) – during each “hand over” of information from one state to the other, memory can be accessed (this access is not obligatory, restrictions on this rest with the programmer). This is actually a portfolio for modelling some psycholinguistic states. The notion of memory very nicely constructs an analogy to the working memory of our mind/brain. Still, also this type of automaton has some problems:

(i) In naturally occurring languages, there is a phenomenon called **cross-serial dependency**. This was exemplified with the help of an example from Swiss German. I do not recall the example in full detail, but the point did come across: ($a^1 a^2 a^3 \dots b^1 b^2 b^3$), the thing being that the pairs ($a^1 b^1 \dots a^3 b^3$) cross each other's boundaries. If one takes away b^3 , then a^3 must also be eliminated – this can only successfully happen if there is a memory that “saves” all the lists of *as* and *bs*. The Push-Down-Automaton cannot deal with this.

(ii) I do not know whether this also applies to multiple embedding constructions such as ($a b c \dots c' b' a'$). Maybe here the boundary makeup is more simple – still, there is crossing going on, but if the automaton can deal with ($a b a'$), then it should (logically?) be able to deal with infinitely many embeddings. This, I do not know precisely.

Finally, the Turing Machine appears to be the most powerful computing device. As it is nicely put: It can compute any computable representation.⁴ Problems with this machine seem to be:

(i) There are **non-computable problems** for a Turing Machine. Put very simply: You cannot instruct one Turing Machine to instruct another Turing Machine. You cannot “ask” one automaton about another automaton.

¹ Actually, I believe that some philosophers of science think that modelling or, to be more precise, rebuilding natural phenomena, is the *only* way of thoroughly understanding how they work.

² For the time being, i.e. for the largest part of linguistics, we just assume that language is an orderly phenomenon.

³ Benni gave us an example of the form: ($a' b^* a'$). This simplest form of an embedding structure, the Finite-State-Automaton is not able to deal with appropriately.

⁴ The Chomsky Hierarchy is famous for attributing this type of competence to the Turing Machine.

(ii) In the field of physics, there are natural phenomena which are ruled by (allegedly) “unruly” (i.e. **chaotic**) systems. The climate is a popular example here: Turing Machines would have the greatest problems in dealing with this kind of natural occurrence.

To me, one question of essential interest would be the following: **Is language (are languages) Turing computable?**⁵ Benni said that, largely, today nobody would expect language to run like a Turing device. When I think of it more deeply, a whole bunch of questions crops up, which I will just for the moment ignore.⁶

The next issue was Neven’s short handout on Interface Conditions (ICs). We seem to have agreed upon the following terminological distinction: **internal ICs** and **external ICs**.

Internal ICs: Ouhalla (1999) discusses the X-bar theoretic relations as one LF interface condition. This clearly appears to be a *condition imposed on the syntactic system as such*. It is a design feature of the syntactic system.

External ICs: These are conditions which shape the syntactic system from the outside. Neven brought up the constraint on multiple fronting in English. There seem to be pragmatic reasons (general constraints on Information Structure) why this is not possible.

One of the more complex questions is *how external ICs could be coded in a syntactic representation*. The *cartography approach* posits a rich Complementizer Domain into which the fronted elements can be inserted. The more projections you have, the more you can front. But I am baffled: *Why should the C-Structure of Catalan look different than the C-Structure of English?* If one assumes all the projections to be there all the time, then this does not explain to you why in one language they can be filled and in another not. I think Michael suggested this to be subject to some sort of *parametric variation*.

⁵ I don’t want to push this point in a footnote, but I think that guys such as Pinker and Jackendoff believe a large part of language actually is Turing computable. Is this a point of contact between Pinker and Chomsky? In how far is the MP influenced by Turing-Machine-thinking?

⁶ There could be a lengthy footnote here.

Perhaps this has something to do with the typological and morphological makeup of the language. The core question would be: *Is there a correlation between morphology and the number of (constituent) elements that can be preposed? How strong is this correlation?*

The next session we should be going into Ouhalla (1999) and try to disentangle core concepts such as **domain**, **minimal domain**, **chain**, **domination**, **containment** and the opaque term **equidistance**.