Wolfgang Koch[*]

Iconicity in instructional texts

Abstract

Diagrammatic iconicity is usually investigated at the surface syntactic level of texts. In this paper, I try to show that a meaningful concept of iconicity cannot be found on this level in non-trivial instructional texts. Instead, we have to dive deeper into semantic and conceptual structure. I present a model of Conceptual Structure that can cope with the demands that understanding an instructional text puts on the reader, and after analyzing a concrete text (a cooking recipe), I show that the concept of control structure is of essential importance for the description of the mapping between a conceptual model and a text. Control structures can be expressed explicitly through linguistic means or be inherent to the semantics of lexical predicates. In both cases, the presence of a dynamic conceptual model is necessary in order to establish iconicity relations between the text and the underlying mental representation.

1. Introduction

In a general semiotic sense, iconicity refers to an analogy or similarity between a sign and the concept in our cognitive model of the world which this sign represents. With respect to linguistic signs, iconicity can be found on all representational levels of language, from sound (onomatopoeia) to discourse or text structure. This last type, i.e. the mapping of relations between objects and actions onto sequences (or more complicated structures) of signs, is called diagrammatic iconicity. (Peirce 1931-58, May 1999)

In this paper, I will concentrate on diagrammatic iconicity as it can be found in the structure of instructional texts. First, I point to the fact that (good) instructional texts should show a high degree of iconicity. Second, I focus on the exact nature of mapping taking place between real world actions and textual instructions. Language is a very flexi-

* *Wolfgang Koch*
Aarhus School of Business, University of Aarhus
Department of Language and Business Communication
Fuglesangs Allé 4
DK-8210 Aarhus V
wk@asb.dk

ble system, so we cannot expect to find all (clusters of) actions mapped into full sentences, but actions are encoded in different lexical or syntactic units. Finally, I discuss by which mechanisms our linguistic and non-linguistic knowledge systems can cooperate in order to process this linguistically broken information (in the sense of a lens breaking light) and to reconstruct the natural flow of actions from the text and its structure.

2. What is ”iconicity”?

In its simplest form, a definition of iconicity could look like the one to be found in Wikipedia[1]:

> Iconicity is the conceived similarity between a form of language and its meaning.

It has to be noticed here that in this definition there are a number of terms that have an intuitive appeal: ‘conceive’, ‘similarity’, ‘form’, ‘meaning’, but that have to be defined more precisely. A somewhat more elaborated definition of diagrammatic iconicity, formulated by Karin Wenz, University of Kassel, can be found on *http://www.uni-kassel.de/fb8/privat/wenz/space/diagram.html*:

> Diagrams, according to Peirce, are icons which represent the relation of the parts of one thing by analogous relations in the sign vehicle. There are structural correspondences between the sign vehicle and its referential object. Since only relations and structures are considered, diagrammatic icons evince a certain degree of arbitrariness. Because of the digital and linear character of language, diagrammatic icons are more frequent in texts than imaginal icons. Diagrammatic iconicity appears in texts when linear relations within the text stand for temporal, spatial, causal, or social relations in the described world. These extra linguistic relations, which structure our experience as complex principles of order, are mirrored in the text as icons.

Iconicity has been found and analyzed on practically any level of language. It starts with phonetics (onomatopoetica), goes on with morphology and syntax and ends up in logical, temporal and argumentative text and discourse structures and strategies. Even the physical length of an utterance can be given an iconic interpretation, like in the following pair (Anderson 2001):

1 *en.wikipedia.org/wiki/Iconicity*

(1a) We didn’t have sex!

(1b) I did not have sexual relations with that woman <pause, gaze averted> Ms. Lewinsky.

It is probably no coincidence that (1b), and not (1a), was actually uttered!

3. Ordo rerum = ordo sermonis?

This paper is dedicated to one specific form of iconicity that often is called diagrammatic iconicity. Many scholars assume that there is an ‘ordo naturalis’ in texts that describe real world phenomena or processes.

> In natural order, […] text and discourse (sermo) have the same arrangement as things in the universe of discourse (ordo rerum) (Enkvist, 1981:98)

It will come as no surprise to the reader that good instructional texts should show a high degree of iconicity. Obviously, it would be rather difficult to cook a meal, to fit an IKEA kitchen or to install and use new software if instructions for these activities were given randomly. Nevertheless, as we will see later on, this is not as trivial as it may sound.

One of the most cited examples in this connection is Gaius Julius Caesar’s famous utterance:

(2) Veni, vidi, vici.

Here the actual order of events in the world, that he arrived, looked around, and was victorious, is mirrored in the text. That is, the real events are verbalized in exactly the same order in the text. As we will see later, these simple sequences are just one among various possibilities of “natural” ordering.

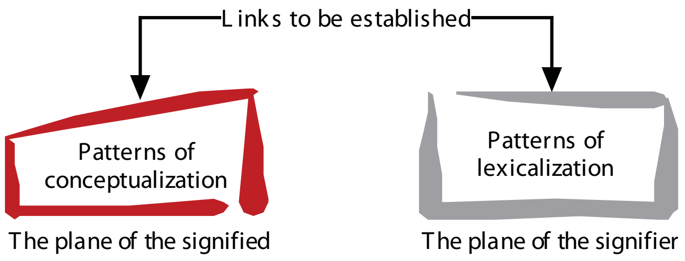
There have been attempts to give natural order a cognitive explanation. For instance, as does Talmy (1990), one could suggest that it seems quite reasonable to think that language evolved after vision, and that therefore it may have partly incorporated its organization. Lakoff and Johnson (1980) have shown that our conceptual system is ultimately grounded in basic spatial and ontological experience through metaphors. (See Larsen Pehrzon 1993 for an elaboration of these aspects.)

In other words, we would expect to have certain patterns of lexicalization on the plane of the signified that correspond (in a systematic way) to patterns of lexicalization on the plane of the signifier (figure 1).

This looks rather simple, and in fact, as we will se later on, it looks too simple.

The reader will also notice that these ideas of iconicity as a kind of direct mapping from signified to signifier on first sight seem to be standing in opposition to one of the most prevalent paradigms of modern linguistics: the idea of autonomous modules in cognition and language. I hope to be able to show that there is no such conflict.

Figure 1


Links to be established
Patterns of conceptualization
Patterns of lexicalization
The plane of the signified
The plane of the signifier

But let us return to the concept of naturality. As Slobin (2005) asks:

> What sort of resemblance is ”natural”? (What would an ”unnatural resemblance” be?) What part of the linguistic message is to be considered the relevant ”form of the sign” for a particular resemblance or analogy? And, […], what are the referents for the terms ”object” and ”concept” – two very different notions; and what cognitive activities or states are to be construed as ”our perception of the world”? In fact, we are dealing with mental constructs on both ends of the equation: construals of linguistic forms and construals of nonlinguistic experiences.

I hope to be able to shed at least some light on these questions.

In an attempt to use Wollheim's (1980) model of diagrammatic iconicity in the domain of art, May (1999) distinguishes between three levels and two relations, ‘seeing as’ and ‘seeing in’.

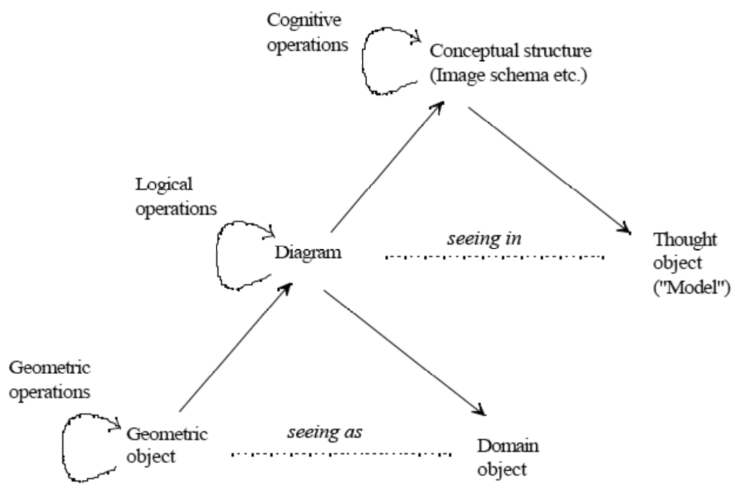
> From the logical point of view the *seeing as* can be considered as an abstract relation of representation (*see* X *as* Y) and from the cognitive point of view as an instance of categorial perception, where some perceived phenomena (X) is identified by assigning a type (Y) to it, i.e. *subsuming* it under a concept (Wollheim 1980). Where the *seeing as*

> phenomenologically is an experience of an identified particular, the *seeing in* is an experience of a state of affairs seen in the particular. The *seeing in* involves a “twofold attention” […] where the viewer identifies a set of figurative objects and simultaneously sees a whole state of affairs in their depiction. (May 1999:1)

In figure 2, we see that the spatial aspects are basic (as geometric operations). When the objects in the domain are mapped onto a mental model (as thought objects), we can perform logical operations on them. Finally, the thought objects are integrated into conceptual structures on which cognitive operations can run.

While these distinctions are helpful, since they move the focus away from the text surface to underlying levels and structures, the concept of Conceptual Structure has still to be clarified, which I will try to do in the next chapter.

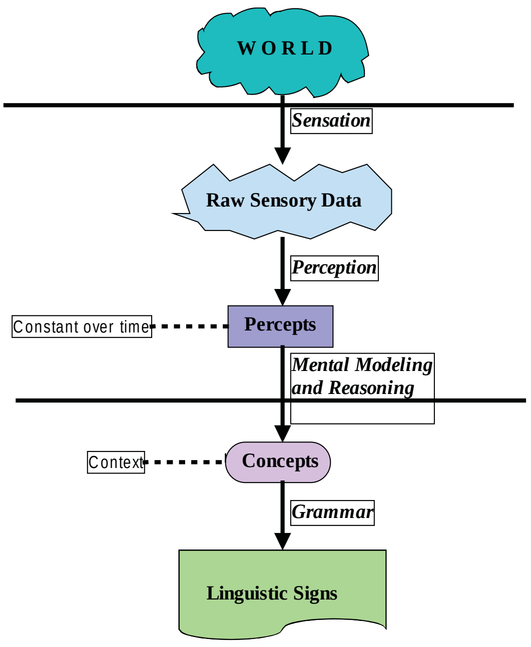
Figure 2 (adopted from May 1999:13)



4. The need for conceptual structures

One of my main arguments will be that a satisfying description of iconicity – as well as many other language related phenomena – is not possible without an idea of how mental models – the ‘construal of non-linguistic experiences”, as Slobin puts it – look like, since the task of grammar is to map linguistic structures on conceptual structures – and vice versa.

Figure 3

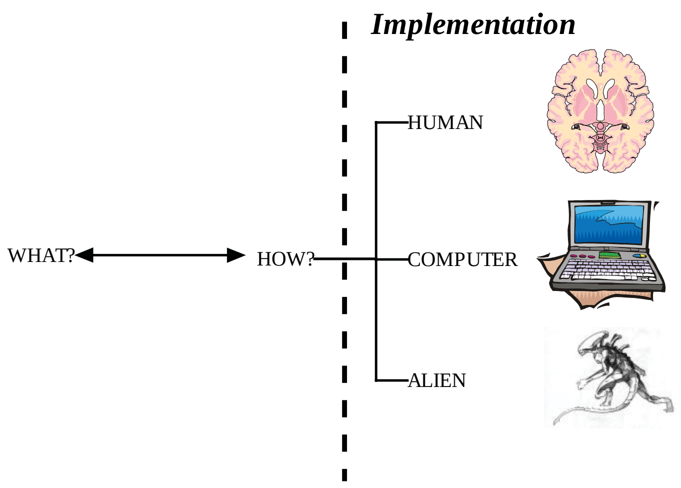


For this reason, I start out with a brief presentation of Conceptual Structure (CS).

The model of CS used in this paper originates from Koch (1978) and was developed further in Andric et alies (1989) and Koch/Rosengren (1995). It has been partly influenced by the concept of CS and Semantic Form proposed by Bierwisch (1983) and others. There is also a certain affinity to the work of Pustejovsky (1995), which I, however, cannot explore further in this paper. Models that distinguish between (linguistic) semantic structures and (non-linguistic) conceptual structures are often called "two-level-semantics", which, at least as far as the model presented here is concerned, is a misnomer. There is only one level of semantics.

The other important circumstance to keep in mind is that this CS-theory is a "high level" theory in the sense this term is used in computer science. What I am trying to model is the flow of information between different modules as a concrete task, like decoding a text, is being solved. My emphasis does thus not lie on the "implementation" of the system, that is the actual physical means involved in the process, be it the human brain or some computational model. The underlying hypothesis is that it is the same kind of symbolic information ("What?") that has to be processed by the brain and a computer program, even if the physical and logical representation ("How?") of this information is very different. Figure 4 is meant to illustrate this fact.

Figure 4



This of course, does not imply that psycholinguistic facts are irrelevant to the architecture of the system, but the implications – and also the problems – of the chosen approach cannot be discussed in this paper.

The CS proposed is in a certain sense minimalistic in that it presupposes only four kinds of ontological classes.

- Entities
- Attributes of entities (also called parameters), defined by sensory input
- Local relations between entities
- Time

Developed originally with verb semantics and argument structure in mind, this model has worked well in implementing a conceptual parser (see figure 6 and the explanation in the text below). Entities can roughly be compared to physical objects, nota bene not "real world objects" but the representation of these objects in a mental model[2]. Entities are defined by their attributes that, at least in the case of concrete entities, are given by the sensory input of the host. Even the relations between two or three entities (encoded in language as adjectives or verbs) are treated as attributes of these objects.

When it comes to encoding CS into linguistic structures, there are three (interdependent) aspects to consider:

Figure 5

CS ***corresponds roughly to***	**SF***
Configurational aspect	Argument structure
Sortal parametric aspect	Inherent features of entities or processes
Algorithmic aspects	Event types, aktionsart/aspect, control structures

* SF = Semantic Form, Bierwisch's designation of the semantic level

In this paper, I will only discuss the configurational and the sortal aspect when they are relevant for control structures. With regard to iconicity, it is above all the concept of control structures that should bother us.

The need for a conceptual model emanates from the simple fact that semantic structures are always underspecified. Let us for example look at an adjective like *good* (see also Pustejovsky 1995). What is a good knife? Probably, the first thing we think of is a sharp knife. However, it is easy to see that a good knife on one hand can have very complex characteristics: there are factors like stability, balance, stainlessness, protection for the user and so on. On the other hand, a good knife can mean different qualities to boy scouts, chefs, circus artists or hobbyists carving flutes. All this boils down to that the semantics of *good* (i.e., the context free meaning) has to be something very vague or general, like

2 I use the label "mental" model also in the case of computer models. The mentioning of aliens is just a precaution!

ranging high on some scale of quality. As soon as we combine *good* with a noun *(good knife, good girl*) we know a little bit more – that is, the Fregean concept of semantic composition adds to our understanding, or rather narrows the infinite possibilities down a bit. Nevertheless, in order to really understand *good* in a given context, we have to add world and situational knowledge, i.e. pragmatic knowledge.

Linguists in the tradition of structuralism have often had a tendency to overlook these complications, or – worse – declared them not to lie within the realms of linguistics. The ”semantics” of *good* would typically look like

(3) /good/: λx [GOOD(x)]

Hm, very elusive.

One of the first to acknowledge the importance of CS was Manfred Bierwisch (1983). The fact that *fast* has to be mapped on different velocities for snails and jets leads to a somewhat more explicit semantic description (in Bierwisch’s terminology SF):

(4) /fast/: λx [VELOCITY(x) = VELOCITY(n) + d]

where n is a prototypical (’normal’) element of the ontological class that x belongs to, and d is some addition to the ’normal’ velocity for that ‘normal’ element. In Bierwisch’ model, VELOCITY is a so-called ‘conceptual constant’ and is not further analyzed. This is still only partially helpful, but definitely a step in the right direction.

However, if it is the task of grammar to map linguistic structure onto mental structures then this ignoring of underspecified meaning is not acceptable.

We face the same problem in truth-value based Tarskian semantics and in Model Theory. The (intensional) meaning of *good* in Model Theory is the set of all objects that are ’good’ in all possible worlds. This definition works fine for compositional semantics, as long as you do not feel that it is a problem that you still do not know what *good* means. To push it to extremes, formal semanticists are more concerned with combining meanings than with defining meaning.

Another motivation for the need for Conceptual Structure is anaphoric resolution. When potatoes are washed, peeled and cut into quarters, the next instance of *the potatoes* is clearly not sufficiently described as, for instance, ”a starchy plant tuber which is cooked and eaten as a vegetable[3]” – and therefore, this definition does not enable us to identify the object in the world model (or in the real world) that *the potatoes* is meant to designate. As we will see shortly, we need not only a conceptual system but also a dynamic runtime model that keeps track of what **happens** to *the potatoes* in the course of, for instance, a cooking recipe.

The third need for a Conceptual System, the dynamics of algorithms and control structures, I will explain in the next chapter.

However, let us first have a look at how the problems mentioned above can be handled by a conceptual parser with explicit world knowledge and a dynamic runtime system. Fig. 6 shows the architecture of AUTOKOCH, the aforementioned conceptual parser developed between 1988 and 1992 at Lund University.

The main objective of the research project “Weltwissen, Sprachsystem und Textstruktur in einem intergrierten Modell der automatischen Sprachverarbeitung[4]” was to investigate the cooperation of different knowledge modules in order to ’understand’ a German cooking recipe. Understanding was defined operationally: the system ‘understands’ a cooking recipe if it can cook an eatable meal based upon it or, to put it in a somewhat weaker form: to have all the necessary information at hand, be it linguistic or non-linguistic, in order to enable you in principle to do the cooking. In order to achieve this, a basic programming language for a virtual kitchen robot was developed, and the task of the conceptual parser was to translate a natural language cooking recipe into a program for this virtual robot. In this process, lexical, morphological, syntactic, semantic and conceptual knowledge – both static and dynamic – had to been drawn upon and their interaction was investigated.

The corpus contained 300 German cooking recipes that also were the base for the lexicon and the grammar. In a prescanning phase, the text was normalized, i.e. abbreviations were expanded to standard tokens, and interpunctuation was transferred to tokens like *fullstop* or *com-*

3 Compact Oxford English Dictionary, http://www.askoxford.com

4 “World knowledge, language system and text structure in an integrated model of automated language processing”. (1988 – 1992).

ma. The syntax was a simple context free phrase structure grammar[5]. Syntactic and semantic structures were built in a parallel, compositional mode. These two outputs of the linguistic processing were given to the conceptual system for specification of the undefined slots in the – underspecified – semantic structure.

It is really then – after the syntactic and semantic structure has been determined – that the real work of understanding begins. All the slots have to be filled[6].

CS consists of four databases and a runtime model. The databases (right hand in figure **6**), are

- Object Hierarchies
- Operation Hierarchies
- Object Related Operational Knowledge (OROK)
- Robot Programming Language

The first two modules are connected to the linguistic lexicon, since lexical elements 'point' to objects or operations in CS. In CS, these objects are described with all their relevant features (certainly not exhaustive, since we only had to handle concrete things and actions, tailor suited to the domain in question). Both objects and operations are organized as hierarchies of inheritance relations. In this way, the common feature of a casserole and a saucepan can be described in one single place, the class concept <container>. The same goes for operations. Again, the linguistic knowledge

/peel/: remove the outer covering or skin from (a fruit, vegetable, etc.)[7]

alone is not very helpful in a real language understanding application

The OROK module is essential for conceptual parsing and thus understanding. Here, the world knowledge is placed. I give you just one trivial example: you peel a raw potato in a different way than a cooked potato. The runtime model passes the characteristics of an actual potato

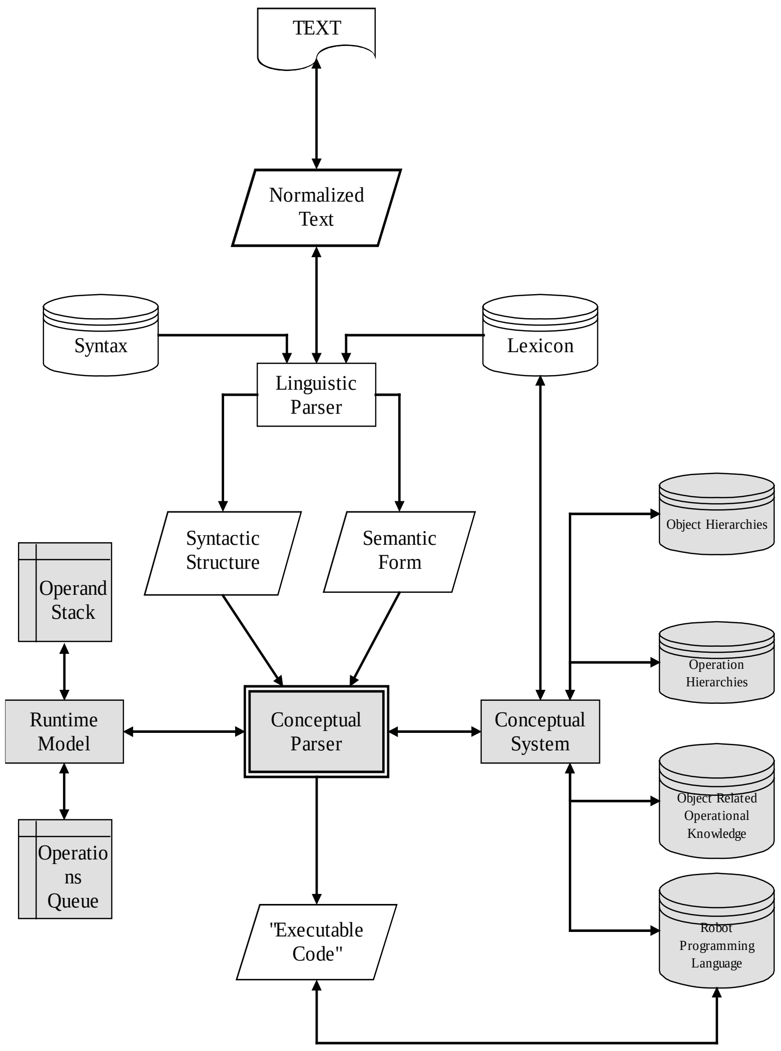
5 The system was implemented in LPA's MacProlog, which had a built-in Definite Clause Grammar (DCG).

6 World and situational knowledge can in turn help us to resolve linguistic ambiguities. We will see an example of this in figure 7 below.

7 Compact Oxford English Dictionary, http://www.askoxford.com

to OROK, which then decides on an appropriate algorithm and appropriate instruments for peeling the actual potato.

Figure 6



The Robot Programming Language is, as mentioned before, the target of the conceptual parser. The output of the whole process is a program that enables the virtual robot to execute the cooking recipes, with all procedures, parameters and control structures defined and in place.

5. Basic control structures

The concept of control structures is well known from numerical analysis and computer science and is one of the most important concepts one has to grasp when trying to program a computer (Cormen et alies 1990). However, which control structures one has to work with depends partly on the programming language used. Furthermore, these "basic" control structures are not basic, in that they can be expressed in terms of one another[8]. Nevertheless, there is a fair consensus among computer scientists today about the most usable control structures, and as we will see shortly, these fit perfectly to the processes going on in an instructional text, like for instance a cooking recipe.

5.1. Sequence of instructions:

<instruction 1>, <instruction 2>,…, <instruction n>

There is an inherent temporal ordering in a sequence of instructions in that <instruction i> has to be performed *before* < instruction i+1>.

5.2. Conditional instructions:

if <condition c> then <go to instruction X> else <go to instruction Y>

Conditional instructions thus consist of two parts, a test of a given condition (whether some number n is greater than zero, whether some element e is in a set S, whether the milk is hot or not, and so on) and the (number of the) instruction that the program has to execute next in case the condition is true/fulfilled. Which instruction that has to be executed when the condition is *not* fulfilled is given in the *else* part, or if there is no *else* part, simply the next instruction will be executed (which is equivalent to saying "else simply go on").

8 Basic in the sense of primitive axioms are only „sequence", „conditional test" and „jump" (to another instruction), where "sequence" in turn is just a special case of "jump" – thus leaving purists with only two axioms.

5.3. Iterations with either a terminating or an initial condition.

- *do <instruction X> until <condition c is true>*
- *do <instruction X> while <condition c is true>*
- *while <condition c is true> do <instruction X>*
- *do <instruction X> n times*

As I mentioned above, these last four control structures are all implemented as <condition> + <jump> on a lower level (assembly and machine language), but this translation is usually hard to read for a human and more easily done by the compiler.

5.4. Parallel processes

Computers have traditionally worked sequentially in that they only can execute one instruction at a time. Parallelism, that is two or more (blocks of) instructions executed simultaneously, had to be simulated in one way or another. Today, a computer can have many processors, so true parallel processing is not a problem anymore (apart from being more difficult to program!). For our problem at hand, it is irrelevant whether the parallelism is simulated or real.

Simple as all of this may sound, it is not that simple neither in computer programs nor in instructional texts, since control structures can be combined and integrated into each other ad libitum et infinitum.

6. The case of Shepherd's pie

Before we can investigate the implications of these considerations for our understanding of diagrammatic iconicity, I will demonstrate how algorithms are realized in an everyday instructional text, namely a cooking recipe.

The following conventions are used in the recipe.

- Operations (**Preheat**, **add**,…) are written in bold style.
- Control operators (<u>to</u>, <u>for</u>, <u>until</u>,…) are underscored (this includes full stops<u> . </u>, commas<u> , </u>, and the conjunction <u>and</u> as sequence markers).
- Control conditions (*5-8 minutes, softened…*) are set in italic.

Furthermore, I have numbered the sentences (not the instructions!) from (a) to (p) for easy reference.

Shepherd's pie[9]

500 g Minced Lamb
800 g Potatoes
1 large Carrot
1 large Onion
60 ml Milk
30 g Butter
50 g Cheddar (Mature)
200 ml Beef Stock
1 tablespoon Tomato Puree
1 tablespoon Vegetable Oil
1 tablespoon Flour (plain)
1/2 teaspoon Rosemary (dried)
1/2 teaspoon Thyme (dried)
Salt
Black Pepper

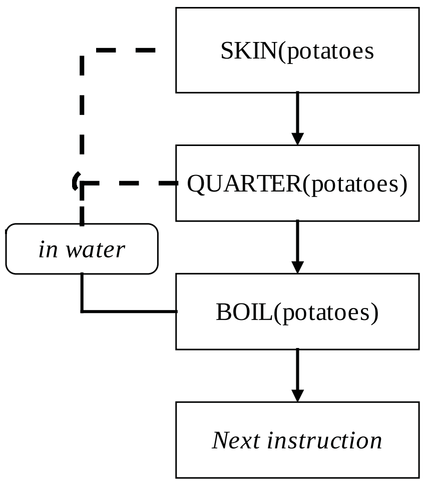
a) **Preheat** the oven to *200°C.*
b) **Dice** the onion and carrot.
c) **Heat** the oil in a large saucepan and add the onion and carrot.
d) **Fry** for *5-8 minutes* **stirring** occasionally until *the onion is lightly browned.*
e) Meanwhile, **skin**, **quarter** and **boil** the potatoes in water until *softened*.
f) **Add** the minced lamb to the onion and carrot and **fry** for *a further 10 minutes*, **stirring** and **breaking** up the mince with a wooden spoon.
g) **Stir** in the flour.
h) **Add** the beef stock, rosemary, thyme, and tomato puree.
i) **Season** with black pepper.
j) **Simmer** for *15 minutes* until *most of the liquid has evaporated.*
k) **Mash** the potatoes with the butter and milk until *smooth*.
l) **Grate** the cheddar and mix into the potato with a pinch of salt.
m) **Check** the lamb for seasoning, **adding** salt if *necessary*.
n) **Spoon** the lamb into a baking dish, and **cover** with the mashed potato.
o) **Use** a fork to **draw** ridges across the top if *desired*.
p) **Bake** the shepherd's pie for *35- 40 minutes* until *the potato is lightly browned*.

9 The recipe was found on the website www.letscook.co.uk.

I could have translated this recipe to pseudo robot programming code, as would be the output of the conceptual parser described in chapter 4 above, but for the sake of perspicuity, I will use flow charts instead. Furthermore, I will not analyze the whole recipe, but only give examples of the control structures described earlier in this chapter.

6.1. Sequences

Figure 7 (sentence e)
…skin, quarter and boil the potatoes in water…



This is simply a sequence of three instructions: *skin the potatoes, quarter the potatoes and boil the potatoes in water.* We will have to do this straightforwardly in this order. However, there are several problems here, which I will only mention. The first problem is that the predicate **boil** in itself contains a control structure (i.e. *keep the potatoes in water with a temperature of approximately 100 C until they are soft, or for about 25 minutes, depending on their size*). The other problem is that the prepositional phrase *in water* is syntactically ambiguous. Do we only boil the potatoes in water, or do we even skin and quarter them in water? This is why I have used dotted lines, because after the syntactic and semantic analysis, we still do not know the scope of *in water*.

The third problem is the ontological state of the potatoes. When the sequence starts, they are whole, hard and unpeeled. After skinning, they are whole, hard, and peeled. After quartering, they are divided into four parts, hard and peeled. Finally, they are divided in quarters, soft and peeled. This is an identificational problem for the conceptual system, since the properties, i.e. the criteria by which CS can identify entities, are changing during the process. As described above in chapter 4, this problem cannot be solved by linguistic means alone either, but there has to be world knowledge and a conceptual runtime system involved.

6.2. Conditional instructions

Figure 8 (sentence n)
draw ridges across the top, if desired

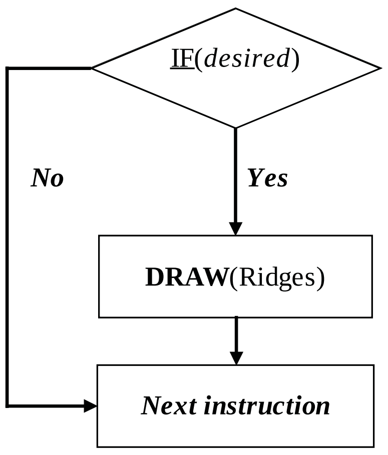


Figure 8 is a classical example of a conditional instruction. The condition is: “Is it desired to have ridges drawn across the top”? If the answer is yes, we have to draw the ridges (this is a control structure of its own, which I have omitted here for the sake of simplicity), if not, we do nothing and continue with the next instruction. One interesting fact to notice here is that the sequential order in the text is not “natural” since the condition *mentioned last* in the text has to be *considered/executed first*.

6.3. Iterations (loops)

As I mentioned before, a predicate can contain a control structure. *Heat the oil* means: *Keep warming the oil until the temperature is the one you want*. Clearly, the "right" temperature is another one for oil to fry something in than for water to boil something in. Again, we have to use conceptual knowledge in order to decide on an appropriate value.

Figure 9 (sentence c)

Heat the oil

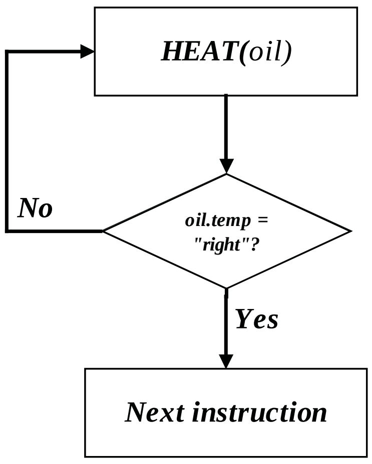
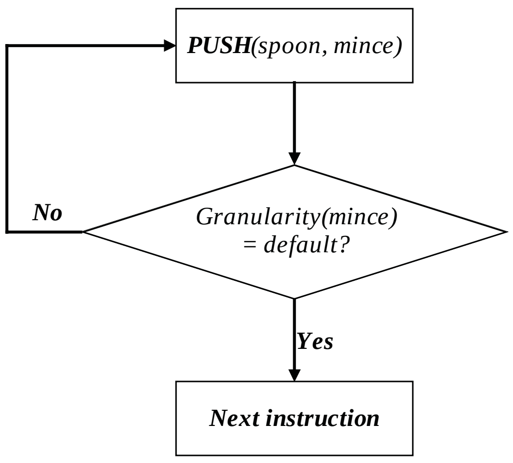


Figure 10 is another example of an iteration, but this time it is not a simple parameter like temperature that has to be observed, but the more complex property of being "broken up" with a desired (and normally also quite uniform) granularity. The linguistic marker for the control structure is the particle *up*, since *break a thing* would imply only one act of breaking. In fact, our conceptual system would consider this a confusing instruction, since the properties of *mince* are not easily compatible with *break*. The instruction gives us no hint whatsoever on how one should *break up mince with a spoon*. Again, we have to draw on conceptual knowledge[10]. I have inserted PUSH in the diagram, because that would be what the OROK module would come up with.

Fig. 10 (sentence f)

...break[ing] up the mince with a wooden spoon



6.4. Parallel processes

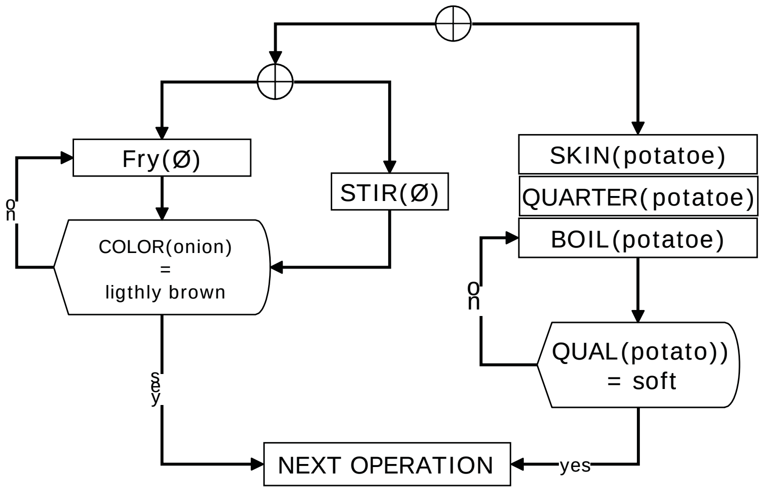
When describing parallel processes we use the term ‘thread’. A thread is a process that runs independently and (at least partially) parallel to another thread.

We know already the thread on the right hand of figure 11: skin, quarter and boil the potatoes (cf. figure 7). This thread has to run simultaneously with the thread on the left hand, which in turn comprises two sub threads: the frying thread and the stirring thread. Therefore, in fact, at times there are three activities going on simultaneously.

This last example is the most complex one, since it contains not only parallel instructions but several other control structures as well.

10 In case you are asking yourself at this point where the system gets this knowledge from: You have to store it explicitly in the OROK database, a task that without doubt is the most time consuming part of building a conceptual parser.

Figure 11 (sentence d + e)



Fry [...] stirring occasionally until the onion is lightly browned. Meanwhile, skin, quarter and boil the potatoes in water until softened.

7. On the relation between algorithms and linguistic form

If we simply consider diagrammatic iconicity to be a question of mirroring sequences then we have seen that this will not be sufficient. This is simply the case because language – including texts – is linear, whereas algorithms in general are not. We have explicit and implicit loops, we have parallelism, and we have even ‘garden path’ phenomena, that is points in the text where we find evidence for the need of “jumping back”. As is the case with the *meanwhile* example (sentence e), we cannot even be sure that we are on the safe side after parsing a whole sentence. The natural conclusion is that we need a two-pass-processing of the text. For a human reader and experienced cook it will suffice to read the recipe once first in order to establish a mental picture of the recipe including the timeline.

For an AI-system, it will suffice incrementally to build a model that can be modified until the last instruction is read. In the AUTOKOCH system (figure 6) this was achieved by generating a program for the virtual robot, and executing this program first after it was finished.

A number of explicit control structures are present in the recipe, but we also find implicit control structures in the semantics of lexical predicates.

7.1. Explicit control structures:

<op1 and op2>
do first op1, then op2. However, there is one example of *and* in the text that induces parallel threads: *Stirring and breaking up the mince*.

<to x degrees>
A final condition to an until-loop.

<for x time units>
The duration of a while-loop.

<until some quality is achieved>
A final condition to an until-loop too, but here the condition is qualitative rather than quantitative.

<meanwhile>
A "jump back" instruction to the previous sentence and a trigger of a new thread.

<in>
As in *Stir in the flour*. A while-loop that exits when all the flour is mixed with the other stuff.

7.2. Implicit control structures:

Dice:	Cut or part until divided into dices (of which size?).
Heat:	Continue supply of energy until a certain high temperature (how many degrees?) is reached.
Quarter:	As *dice*, but now the result has to be four parts of approximately the same size.
Boil:	As *heat*, but now we know that the final temperature of the medium (as default a water-based fluid) has to be 100° C. The length of the boiling process is explicitly

	given in the recipe or has to be inferred from OROK (world knowledge).
Break up:	Already discussed in the text 6.3.
Season:	Add spices and stir until the resulting taste is “good enough”.
Mash:	Here the final state of the object is the degree of “mashedness”, a type of consistency.
Grate:	A special process that generally uses a specific instrument. This means that the granularity is given by the instrument. The action has still to be repeated (= a loop) until the whole object is processed.
Mix:	Blend two or more ingredients until a sufficiently homogenous blending is resulting.
Cover:	Place some object X on some other object Y until X forms a “roof” for Y. The algorithm is highly dependent on the quality of X.

8. Conclusions

We have seen that there is no prima facie diagrammatic iconicity in the recipe because the ‘world’ the recipe tries to mirror is only partly organized linearly. Since the temporal structure of speech is strictly linear, however, we have to use other means than simple sequences in order to make sure that the instructions are executed in the desired order.

In the case of a rather complicated enterprise like following a cooking recipe, we find that some (in our example in fact most) algorithms are inherent to verb semantics. Examples of this are the verbs *season* and *mash* which mirror a loop and a final condition, where you have to add spices and check the taste until you are satisfied, or where you have to put some pressure or impact on the object(s), continuously checking the resulting consistency and adapting your further processing to it until you “get it right”. That these inherent conceptual properties are relevant for linguistic analysis, too, can be seen by the simple fact that a modifier like *cautiously* has to be interpreted with reference to the *amount* of spices in the case of *season* but the *force* implied in the case of *mash*. You do not understand phrases like “mash cautiously” or “season cautiously” unless these processes are represented correctly in your (conceptual) knowledge system.

Thus, we can say that a relatively complex structure of actions has been clustered into the predicates, whereas other actions are verbalized as explicit control structures (with linguistic means like *until*, *meanwhile*, *for*, *and*, *if*, or even a simple comma). Our first conclusion is accordingly that diagrammatic iconicity has to be looked for on a higher level than single actions. Language maps **clusters** of actions.

When there is a clash between the sequence of clusters in the ‘world’ and the sequence of clusters in the text, we can repair this mismatch by the fore-mentioned explicit linguistic means.

Our second and final conclusion is, therefore, that diagrammatic iconicity has to be looked for on the semantic and/or conceptual level rather than in the surface structure and/or some simple concept of “sequencing”.

Linguistic knowledge alone does not help us.

10. References

Anderson, Richard D., Jr. 2001: “I did not have sexual relations with that woman <pause, gaze averted> Ms. Lewinsky” – *The Iconicity of Democratic Speech in English*. *http://www.polisci.ucla.edu/faculty/anderson/Lewinsky.htm*

Andric, Barbro/Hansson, Kerstin/Koch, Wolfgang 1989: Prozeßsteuerung und Verbsemantik in einem computersimulierten Küchenmodell: Teilen, Mischen und Erwärmen . In *Sprache und Pragmatik*. 1989; årg. 14, p. 1-106

Bierwisch, Manfred 1983: Semantische und konzeptuelle Relationen lexikalischer Einheiten. In *Rucicka/Motsch* (Eds.), p. 61-99

Bierwisch, Manfred/Schreuder, Robert 1992: From concepts to lexical items. In *Cognition* 42, 23-60

Cormen, Thomas H./Charels E. Lesierson/Ronald R. Rivest 1990: *Introduction to Algorithms*. The MIT Press: Cambridge Massachusetts & London, England

Enkvist, Nils 1981: Experiential iconicism in text strategy. In *Text* 1 (1). 77-111.

Koch, Wolfgang/Rosengren, Inger 1996: Locative Alternations‘ in English and German: different lexicalisations of the same conceptual structure. In *Sprache und Pragmatik*. 1996; årg. 45.

Koch, Wolfgang/Rosengren, Inger 1995: Secondary predications: their grammatical and conceptual structure. In *Sprache und Grammatik*. 1995; årg. 35, s. 1-100

Koch, Wolfgang 1978: *Kasus - Kognition - Kausalität. Zur semantischen Analyse der instrumentalen "mit"-Phrase*. Lund: Lunder germanistische Forschungen 47, 1978. 182 p.

Lakoff, George/Johnson, Mark 1980: *Metaphors we live by*. Chicago: University of Chicago Press.

Larsen Pehrzon, Mariann 1993: Between text and grammar: the principle of iconicity. In *Estudios Ingleses De La Universidad Complutense* 1/1993, p. 111-126.

May, Michael 1999: Diagrammatic reasoning and levels of schematization In T.R. Johansson/M. Skov /B. Brogaard (Eds.): Iconicity. A Fundamental Problem in Semiotics. NSU Press, Århus 1999. *Citations are taken from http://imv.au.dk/~pba/Preprints/DiagReasoning.pdf*

Peirce, Charles Sanders 1931-58: *Collected Papers I-VIII*. Hartshorne, C, Weiss, P, & Burks, A, (eds.). Cambridge, Mass. Harvard University Press 1931-58

Pustejovski J. 1995: *The Generative Lexicon*. Cambridge (USA): MIT Pres

Rucicka, Rudolf/Motsch, Wolfgang (eds.) 1983: *Untersuchungen zur Semantik*. Berlin (=Studia grammatica XXII).

Slobin, Dan I. 2005: Linguistic representations of motion events: What is signifier and what is signified? In C. Maeder/O. Fischer/W. Herlofsky (Eds.) (2005): *Iconicity Inside Out: Iconicity in Language and Literature* 4. Amsterdam/Philadelphia: John Benjamins.

Talmy, L. 1990: *How Language Structures its Concepts: the Role of Grammar*. Unpub. MS. International Center for Semantic and Cognitive Studies. Republic of San Marino.

Wollheim, Richard 1980: *Art and its Objects*. Cambridge: Cambridge University Press (2. ed.)