DEDUCTIVE MODELING OF HUMAN COGNITION

Goran Hagert

Department of Psychology
Uppsala University
Sweden

Sten-Ake Tarnlund

UPMAIL
Department of Computing Science
Uppsala University
Sweden

ABSTRACT

Deductive analysis (DA) is presented as an approach to the study and simulation of human cognitive processes. DA is composed of a psychological theory and a methodology that can sheed light on mental phenomenas. The cognitive theory is embedded in the problem space and the control structure hypotheses. The methodology consists of logical derivations of computer models from an abstract specification. The item-recognition task and the three-term series task are analyzed for purpose of illustration. Several aspects of human cognition in these environments are discussed. It is argued that DA brings new notions to the study of human cognition, for instance, to design sets of models and to distinguish between empirically equivalent models.

I INTRODUCTION

We will present and discuss a few new aspects of studying cognitive processing. They were called Deductive Analysis (DA) in <5> and this type of deductive modeling is further developed here. The main aspect of DA is cognitive simulation, so it belongs to one of the main streams in psychology. Thus, our goal is to design programs that can simulate human thinking, and one such program is a cognitive model. The hypothesis that the architecture of mind can be viewed as a symbol manipulating system or an information processing system (IPS) <12> is related to our deductive modeling. There are two other hypotheses about mind as an IPS-structure. The problem space <11,12> and the control structure hypotheses <10>, respectively.

Our deductive analysis gives a method for designing computer models, in particular, this means that models of thought are derived, in logic, from an abstract model specification. This specification is abstract in the sense that it only contains a precise definition of the model. No information about representation and execution is explicit in the specification. Such issues have to be specified on the way toward the design and implementation of a model, i.e., during the derivation.

II COGNITIVE THEORY

The problem space hypothesis states that problem solving, decision processes, as well as reasoning, are search processes through a problem space. Each state in the space contains a set of symbol structures which correspond to what the problem solver knows about the problem at a particular time. A new state is generated by modifying the symbol structures. *The* successive series of knowledge states that the problem solver passes is not a random series, but is generated by a strategy that the problem solver is following.

This is the general idea of the problem space hypothesis. The particular details of the construction of a space and the search through it depends on the task and on the person who solves it. For instance, if the task is simple, as in the item-recognition task <16> and the three-term series task <8>, the process of constructing a problem space is trivial. The task is easily assimilated to the knowledge of the problem solver. A person will probably also generate a strategy quickly and execute it immediately and the time to solution is short. In contrast, the Tower of Hanoi puzzle and the Missionaries and Cannibals problem are examples where a human problem solver may have troubles in constructing a relevant problem space and a strategy. The time to solution will, of course, be longer compared to simple tasks.

We will be restricted to cognitive prooesses in simple task environments. The general characterization of such processes is that the task is easily assimilated. This means that the person is able to represent and understand the task quickly.

Given a conceptual framework in which mental events are viewed as search prooesses through problem spaces and guided by strategies, it can be asked how this strategy is organized and executed. This is the problem about control structure.

We are currently developing DA in simple environments, where we think of a strategy as composed of a set of goals. The goals are achieved by operations. Let us show a simple abstract example of a strategy.

Assume that the goal-state GOAL can be reached by a means-ends analysis using the operator:

GOAL <— SUBG0AL1 6 SUBG0AL2 (1)

which means that SUBGQALI and SUBOTAL2 have to be attained to reach GOAL. To solve the two subgoals, which are our new state we need some operators:

SUBG0AL1 <— P1 & P2 (2)

SUBG0AL2 <— P1 & P3 6 P4 (3)

SUBOOALI can be attained by activating the operators called PI and P2, whereras SUBO0AL2 can be resolved by calling PI, P3 and P4. Each operator call may lead to an operator that transforms a state to a new state or leaves a state without change. The latter case means that a subgoal has been solved, and we write:

P2 (4)

This example illustrates the simplest form of the language that we will use in developing DA. It is a subset of first-order logic, called Horn clauses <10>. As a consequence, it is possible to read the first expression above as: GOAL is true if SUBOOALI and SUBG0AL2 are true. In order to run Horn clauses we have to specify how the formulas are executed. There are many ways to execute a set of Horn clauses <9>.

I I I ON THE METHODOLOGY

We shall discuss three aspects of our method. First, we take up the task analysis or the identification of possible goals and operators, secondly, the derivation and execution of strategies in different problem spaces, and thirdly, the empirical evaluation or the comparisons between simulations and empirical observations.

A. Task analysis

We think of a task analysis as an identification of goals that can be involved in a solution to a particular task. It means also that the set of goals, or a subset of them, is specified in a model specification. Each specification contains a set of possible simulation models that are abstractly but precisely defined. A model specification leaves several aspects unspecified. For example, the representational format that goals and procedures operate on is not defined. Therefore, no decision can be made about how to manipulate information.

Design issues:

1 Representational format: How is internal information represented?

2 Operational assumptions: How is information manipulated?

3 Operational mode: How is each operation transformed? How is a transformational time function defined for an operation?

4 Operational order: How is the set of operations organized?

5 Operational rule: How is the entire set of operations transformed? How is this time function defined?

Fig. 1 Five issues on the design of models.

Fig. 1 contains a list of issues not included in a model specification. Hypotheses have to be formulated on each of the five design issues to implement a oanputer model. There may exist several different hypotheses on each issue, for exanple, on internal information formats. Consequently, this step can be as important as the specification since the hypotheses on the design issues indirectly restrict the set of possible models. In short, the specification and hypotheses contain a set of possible simulation models, i.e., a set of pairs of problem space and strategy.

B. Derivation and simulation

The task analysis leads to a model specification and a set of hypotheses about the design issues. We write them in first order predicate logic. These statements about mental processing in the given task domain can usually not be run as simulations, they do not constitute a computational model. To reach a form that represents mental phenomenas and is computable we may have to translate the specification into a set of more basic statements. In. DA it is carried out by logical derivation, in particular, we use a natural deductive system of Gentzen type <4>. Such a system has been used for formal program development in <6>. However, in this oontext we are not interested in automatic deduction, although developments in that area may influence DA.

An abstract specification consists of a precise definition of a set of goals. A derivation means that all the aspects, which are left unspecified are successively defined by design hypotheses. For instance, a specification does not contain any specified representation, so we can introduce an assumption on mental representation. There are, of course, several such hypotheses, leading to different derivation paths. When one representation has been introduced we have defined a problem space. The next steps in the derivation involve the strategy that will be used in that space. Again, there may be several ways to define a strategy, since the specification does not contain any information about it.

At each derivation step we use a logical inference rule, so a derived simulation model is a logical consequence of the specification. This property is rarely satisfied for simulation models.

The result of a derivation is a set of statements that forjti a cognitive model. The model includes both a problem space and a strategy to use in that space. When the statements are Horn clauses we can run them in a top-down manner by Prolog <14,18>, in this way we get our simulations automatically.

C. Empirical evaluations

The models are logical consequences of the specification, but it remains to see if they are empirically true. It may not be the case that they constitute problem spaces and/or strategies that people spontaneously use in the particular task

environment. A model may have properties not shared by the human cognitive system, or properties that may be learned. Thus, we have to evaluate them by comparing the results from the simulations by empirical observations.

The simulations contain two types of data: reaction times and traces. In addition, we know which representation each model is working on so, in fact, we have three different types of data. Mental representations are, however, impossible to use, at least with the type of empirical methods used today.

When evaluations have been made it is possible to compare the models that can reproduce the empirical observations with those that cannot. These comparisons result in an identification of the problem spaces and/or strategies that people do not use* It is also possible to compare the successful models with each other to see what problem spaces and/or strategies people do use. These models share properties in the form of design hypotheses that are empirically or functionally equivalent. This comparison may also show types of data that could distinguish them further.

IV TWO EXAMPLES

Two simple task environments will be used for illustration. The first task is a very simple one and called the item-recognition task <16>. The other task is called the three-term series task <8>.

A. The item-recognition task

In this task a person is instructed to memorize a list of symbols, for example, letters or digits. The memorized list is called the positive set. The person is then presented with a probe symbol shortly after a ready signal. The task is to answer "yes" if the probe symbol is a member of the positive set, and "no" otherwise. The size of the positive set in terms of number of symbols is usually varied from 1 to 7. This task has been investigated a number of times <16>.

Even if the task is simple, a person who is trying to solve the task needs to do something mentally. There are at least four intermediate states before an answer is given. Following our terminology there are four subgoals to attain before an answer can be reached <16>.

1. The encoding goal is a process where the external probe is perceived by the person and transformed to an internal symbol in memory. Hence, there must be an operation called *"ENCODE* from an external probe to an internal representation of that probe.

2. To solve the task the person must in some way compare the probe with the symbols in the positive set. Thus, there must exist an operation from the probe and the set to a note that signals a match or a mismatch. This operation is called COMPARE.

3. The note from the second subgoal has to be examined. EXAMINE is an operation from the note to an internal answer.

4. The person has to translate the internal answer into an external action. This is the DBCODE operation.

In prepositional logic we can write a simple specification

SOLVE <~ ENCODE & COMPARE & EXAMINE & DBCODE (5)

which we read as: to attain SOLVE (the goalstate) resolve ENCODE and COMPARE and EXAMINE and DECODE. This is an insufficient specification for our purpose so we reformulate it. The goal-state can be attained if the probe is enooded and if the positive set is empty the answer is "no"; otherwise compare the probe with elements in the set, and examine the note. Finally transform the answer into action. In predicate logic we write

$$\forall p' \forall s \exists a' (SOLVE(p\backslash s) = a' \leftarrow\!- (6)$$
$$\exists p \exists n \exists a (ENCODE(p) = p 6$$
$$(s = nil \;\&\; a = \text{"no"} \;\&\; DBCODE(a) = a^1) \vee$$
$$(CEMPARE(p,s) - n \;\&\; EXAMINE(n) = a \;\&$$
$$DBCODE(a) = a')))$$

This is an abstract model specification because it leaves several aspects unspecified. For instance, what is COMPARE? How is the information (the positive set) represented, etc.,. The aspects that are not specified can be found in the design issues. We have included five issues in Fig. 1. It may be possible to extend the list further, but for our present purpose it is sufficient.

Different hypotheses can be formulated on each of the design issues. Let us start with a representation of the positive set. At least two formats can be suggested, simple-list and d-list, respectively <1,17>. The positive set is represented as a simple-list if and only if the set is empty or constructed of a proposition element(e) and a simple-list <5>. Formally,

$$\forall set\ (simple\text{-}list(set) \longleftrightarrow set = nil \vee \quad (7)$$
$$\exists e \exists set'(set = e.sef \;\&$$
$$element(e) \;6\; simple\text{-}list(set')))$$

So far, we have formulated two hypotheses. Four issues remain. For instance, the second issue concerns how information is manipulated. Given a simple-list format of the positive set membership can be defined either by an operation from the beginning of the list or from the end, corresponding to the notions of stack and queue, respectively. For a stack an element e is a member of a list set if and only if there exists another element x and a list y where element e is equal to element x or is a member of list y.

$$\forall e \forall set(e \in set \longleftrightarrow \exists x \exists y(set = x.y \;\& \quad (8)$$
$$element(x) \;\&\; simple\text{-}list(y) \;\&$$
$$(e - x \vee e \in y)))$$

The third issue is operational mode. This issue is particulary relevant for the COMPARE goal. At least two different modes are possible. It can

operate in either an exhaustive mode, or it can be self-terminating. Thus, we can postulate two hypotheses about COMPARE <5>.

The fourth issue concerns operational order. The four subgoals are computed by four operators and in this simple example, the order is given. However, it should be pointed out that this may not be the case in another environment. This issue also includes timing assumptions which we will return to below.

The fifth issue concerns the global control of a program that, for example, can be in a serial or parallel! ordering. It also includes timing assumptions. We assume an operational rule corresponding to a serial top-down execution as e.g., in Prolog.

In summary, we have formulated two hypotheses about representation, two about manipulation two about operational mode, one about operational order, and one about the operational rule. The first six are particulary relevant for the derivations and the remaining ones for the simulations.

We focus on psychological notions and can thus for reasons of space leave out formal derivations <6>, but we shall outline the general structure. For each goal in the specification we derive a set of operations by using our design hypotheses. We assume a representation e.g., the simple-list format, and also a modifier, for instance, the stack hypothesis, Then we assume the mode of the operation, for example, a self-terminating mode. The resulting set is Model-1. By substituting one hypothesis by another, we can derive a second set Mortel-2. In this way we can derive six simulation models differing in at least one cognitive aspect, but satisfying the model specification.

In Fig. 2 we summarize the derivations in a deduction tree.

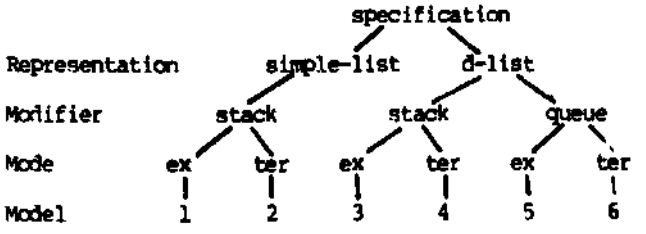

Fig. 2. The deduction tree summarizing the general schema of the derivations (ex = exhaustive, ter = self-terminating).

As can be seen in Fig. 2, two of the six models correspond to two different strategies within the same problem space, whereas the other four are strategies in another space. Fig. 3 shows Model-1 that uses a simple-list representation and searches the positive set exhaustively by a stack modifier.

We have arrived at six computable models from a specification by task analysis and deduction.

Specification (7)

Deduction

Model-1

```
SOLVE(p',nil,a) <— ENCODE(p',p) & DECODE(NO,a)
SOLVE(p',s,a') <— ENCODE(p',p) & COMPARE(p,s,n) &
                  EXAMINE(n,a) & DECODE(a,a')
ENCODE(p',p)
COMPARE(p,nil,n) <— n=1 v n=0
COMPARE(p,p.s,1) <— COMPARE(p,s,1)
COMPARE(p,e.s,0) <— COMPARE(p,s,0)
COMPARE(p,e.s,n) <— COMPARE(p,s,n)
EXAMINE(1,YES)
EXAMINE(0,NO)
DECODE(YES, say(YES))
DECODE(NO, say(NO))
```

Fig. 3. Model-1 in Horn clauses.

The six models are programs that can solve the item-recognition task. Timing assumptions on design issues 3 and 5 make it possible to collect reaction time data and trace data in the simulations, in which we assume a simple additative function i.e., time is added linearly both within an operation and between operations. The particular estimates used in the simulations, the reasons for each timing assumption, as well as a discussion of the empirical evaluations can be found in <5>.

We want to point out that DA makes it possible to be precise in the empirical evaluation. Data that distinguish the models can be understood in terms of the design hypotheses. Furthermore, it is possible to identify types of data that could distinguish empirically equivalent models. In Fig. 4 we illustrate these points.

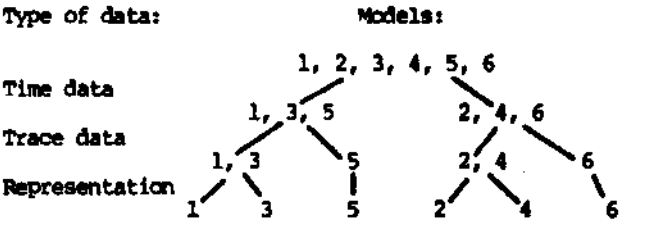

Fig. 4. Empirical discrimination between models by reaction time data, trace data, and representation. Representation is included for sake of complete discrimination. This type of data is impossible to collect in empirical research.

B. The three-term aeries task

In our second example of DA we will analyze the three-term series task introduced by Piaget <13>. The subject is given two premises that describe the relations between three objects. For instance, "Adam is shorter than Bob" and "Bob is