

This paper was originally published in  
 ERGONOMICS 1990, vol. 33(10/11), 1215-1229

as part of a special issue on  
*Errors in the operation and transport systems*  
 edited by John Groeger and Ivan Brown

## Errors and driver support systems

JOHN A. MICHON, ALISON SMILEY<sup>1</sup> and JANS AASMAN

Traffic Research Centre, University of Groningen,  
 PO Box 69, 9750 AB Haren (Gn), The Netherlands

Recent technological developments seem to pave the way to sophisticated electronic co-driver systems that may help automobile drivers to cope with an ever increasing information load, to avoid certain errors, and to recover from others. GIDS—which stands for Generic Intelligent Driver Support—is a research project (under the EEC DRIVE Programme) to study the feasibility of an adaptive co-driver system. The conceptualization of a GIDS system requires close attention to performance errors as they may occur in certain subtasks of the driving task. One important issue that should be considered in some detail is that GIDS may eliminate errors as well as introduce them. Should various types of errors be represented *formally* and, if so, how they should be represented in order that GIDS can detect and cope with behavioural errors that drivers are likely to make under certain conditions? The requirements imposed by the project's goal to actually implement various driver support functions into a GIDS system are imposing tight constraints on error definition and identification. Some of the requirements will be discussed in terms of Soar. Soar is an intelligent computer architecture which is the embodiment of the theory of human problem solving formulated by Newell and Simon (1972). To the extent that the driving task is representable in Soar, the error theory that is required for any type of GIDS system to function must also be representable in Soar.

### 1. Introduction

In driving, much as in other human activities, people will usually try to attain goals effectively, efficiently and safely. In other words, they will try to behave intelligently. **In** this respect driving simply amounts to problem solving, that is, to reach a goal by means of successively diminishing the distance between a present state and a goal state, in a problem space that allows certain actions but not others (Newell and Simon 1972). Considered from this angle there is a lot of error in the world: any intermediate state that is not identical with the goal state leaves a gap to be bridged. In terms of problem solving theory as well as systems control theory, this is exactly the way we look at errors. An error is simply any difference between the set points and the actual state of a system at a given time. This is a straightforward view of error, but it leaves out a number of aspects that are considered relevant in error theories, for instance Reason's Generic Error Modelling System, GEMS (1987). From our point of view, a system will be in error most of the time. It also defines almost all behaviour as error-driven, or at least as recovery-driven.

Leaving aside the merits of an elaborate semantics of error, such as Reason's, to describe human behaviour, GEMS may help to adopt a slightly less austere designers' view of error than we suggested in the previous paragraph. Recognizing the essential goal-directedness of human behaviour, we will treat as an *error* only those consequences of behaviour that actually *increase* the distance between the present state and the goal state eventually to be reached. Of course this requires a provision to

recover from local minima that force the behaving system to move away from its goal in order to be able, subsequently, to approach it more efficiently or safely. A simple example is the performance of the pianist who raises her hand before lowering it on the keyboard; we would consider such behaviour erroneous only if, as a result of her action she would fall out of beat. This view appears consistent with the view of Harvey *et al.* (1975), who defined error (in the traffic situation) as 'any action or lack of action by drivers that would require them and/or other road users to implement a correction in order to make the situation safe again'.

Most psychological theories about the nature and form of behavioural errors take a different, if complementary, point of view. They identify and classify performance failures in ways that do not derive directly from the task domain in which the problem solver is active. Instead they attribute error tendencies to innate or acquired traits (and especially the limitation) of the human being. The variety of approaches here appears to be considerable, reaching from the moralistic (pivoting on ideas of moral turpitude and blame) to the psychonomic (describing limitations of the human being as information processing system). In the latter case there are two major research traditions. The first derives from statistical studies of the likelihood of certain events and combinations of events, eventually leading to the quantitative techniques of safety analysis and accident epidemiology.<sup>2</sup> The other approach is one that has been preferred by experimental psychologists and ergonomists. They have attempted in several ways to recognize the functionality—and especially the lack of functionality—of system performance that may induce erroneous behaviour and accidents.

From our point of view, these approaches to the issue of behavioural error seem to leave unanswered the important question, how such theories would contribute to the computational formalization of an activity domain—that is, a goal-oriented *task*—so that the resulting behavioural syntax would not only produce correct task behaviour, but also, under the proper circumstances, generate the full range of behavioural errors observed in this domain. The recent idea that it may be possible to provide drivers with in-car support systems has added strongly to the relevance, and even the urgency of this question.

## 2. Driver support

Drivers must cope with a considerable, and growing, amount of information of an increasingly complicated nature. This information covers all three levels of the driving task that we normally distinguish, viz., planning and navigation, manoeuvring, and vehicle control (Michon 1971, 1985). Of course, as long as people have been active as automobile drivers, there has been driver support to help them cope with this information. Initially, however, driver support was restricted to such items as road signs, lane markers, and speedometers. By now we have actually lost the awareness that these permanent features of the road and the vehicle were indeed meant as driver support, that is, to help drivers to avoid errors and to recover from errors once they had been committed.

At this point it should be mentioned in passing that driver support is of consequence to travel management (where support means help in getting where you want to be, fast and efficiently), as well as to safety aspects of driving (where support means avoiding accidents and recovery from dangerous situations).

It will be clear that the advent of microelectronics has vastly increased the potential for driver support, including adaptive (or intelligent) functions which take account of specific aspects of the situation and the drivers. Due to the technical developments in

this field, we can now extend the idea of driver support to a considerable array of error avoidance, warning, and recovery functions, including the following:

- enhancing information;
- augmentation (providing extra information);
- warning;
- advice;
- explanation;
- instruction;
- intervention;
- substitute (secondary) control;
- autonomous (primary) control.

The potential for driver support is presently being studied in the framework of the DRIVE Programme of the EEC, in a project that is called GIDS—which stands for *Generic Intelligent Driver Support*.<sup>3</sup> On the basis of this study a limited prototype co-driver system will be developed as a demonstration of the feasibility of the concept of adaptive driver support (Smiley and Michon 1989). In the GIDS system, information from various environmental and in-vehicle sources will be filtered, prioritized, and made available to the driver, in such a way that the information will be maximally relevant to the driver's momentary support requirements. At the same time, the presented information will have to be responsive—at least in principle—to the level of experience, the states and traits of the driver. The GIDS Project concerns the requirements of various driver tasks, such as overtaking or crossing an intersection, the adaptive presentation of the information required and the integrative features (dialogue control, filtering principles, and display characteristics).

The present conceptualization of GIDS (Smiley and Michon 1989) involves three distinct task levels, corresponding to planning and navigation, manoeuvring, and vehicle control, respectively. Each requires separate behavioural strategies of the driver and, consequently, different support functions. The adaptive functions of GIDS should not be restricted to an 'awareness' of the specific needs of the driver on the basis of the prevailing conditions. The system should also be capable of acting on past experiences with *this* particular driver. The conceptualization of GIDS goes even one step further: a GIDS system should ultimately be able to provide tutorial information too. Finally it will offer support under circumstances that are not strictly related to the driving task as such, but that are increasingly becoming a part of normal driving habits, such as carrying on a telephone conversation while driving. All this requires a complicated architecture with a tight set of structural and functional constraints, which are presently under investigation in the GIDS project (see also Rothengatter *et al.* 1990, Janssen 1990).

### 3. GIDS architecture

The GIDS system, as it is presently conceptualized, will incorporate the following essential subcomponents in order to provide adaptive support to the driver (see figure 1).

#### 3.1. Analyst/planner

The first component of the GIDS system is called the *analyst/planner*. It accepts inputs from the various sensors connected to the GIDS system. Such sensors may be

1.

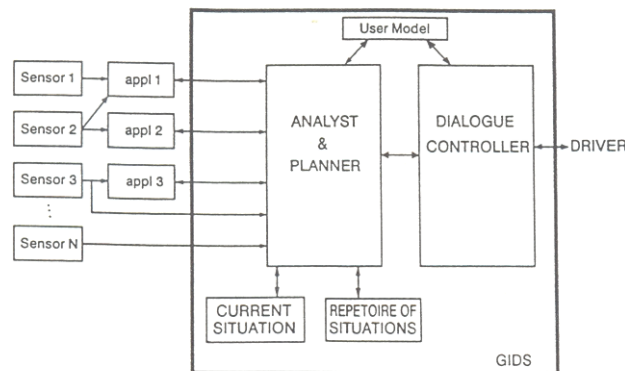


Figure 1. Subcomponents of the GIDS system.

conventional in-vehicle indicators or roadside signals, but they may also be sophisticated 'devoted' applications that provide specific driver support, such as a navigation system, or an anti-collision radar. These sensors provide GIDS with its actual, real-time information about the present driving conditions. The analyst/planner will match the inputs received from the various sensors with concurrent inputs it receives from two other components of the GIDS system, the actual reference situation and the user model that are discussed below. If a mismatch occurs the analyst/planner should be able to infer what may be wrong and what ought to be done about it. An example illustrating the role of the analyst/planner might be that the actual driver, in a given situation, would embark on a merging manoeuvre that would not fit the available time frame (as determined by the current situation). The conclusion of the analyst/planner might then be that it has three options: (a) to enhance the information available to the driver to increase the speed of decision-making; (b) to advise the driver to speed up, to extend the available dynamic time headway; or (c) actively interrupt the manoeuvre. On further analysis it might then find, for instance, that only the second alternative would fit prevailing circumstances.

Although a simple support system would consider only momentary discrepancies, eventually a GIDS system will have to be able to represent the history of previous behaviours of a driver. The system should be capable of acting on the driver's performance history as well as on his present performance, all in the light of the prevailing circumstances.

### 3.2. Reference situations

A second feature of the GIDS system is a *repertoire of reference situations*. This component contains a set of scenarios that describe in sufficient detail all the knowledge and the rules of operation that are required to correctly perform certain specified parts of the driving task. Either on the basis of an explicit request from the driver, or on the basis of specific information received from the sensors, the most appropriate reference situation is selected from the repertoire to be the *current* or *actual reference situation*.

Reference situations must include implicit knowledge (if you steer too briskly you may lose control) as well as explicit rules (traffic regulations such as, if you are within a built-up area you are not allowed to exceed 50 km/h). They also incorporate knowledge about properties of the standard human processor (Card *et al.* 1983), including facts about the means and distributions of reaction times, visual acuity, and other

perceptual, cognitive, motor, and motivational parameters. A repertoire of reference situations is not readily available for the driving task as a whole and will not be for a long time to come. However, the elaborate analysis of the driving task carried out by McKnight and Adams (1970) is an excellent first approximation. In the GIDS project a selection will be made of a small set of relevant (sub)tasks that *can* be formulated in the strict terms of a computational model. The selection includes such relatively uncomplicated tasks as entering and exiting a roundabout, merging, curve following, or crossing a simple intersection. These tasks are carried out in a specially designed 'small world', consisting of a single roundabout, with four branches and a peripheral oval. The small world paradigm provides a way to integrate various empirical studies, (real world, closed track, driving simulator) and computer simulation studies can all reproduce the small world conditions (see figure 2).

### 3.3. *User model*

A GIDS system is—or rather, will eventually be—there to support a real driver in a real driving environment. Consequently a third component of such a system must be a model of the present driver's representation (his or her knowledge) of the driving task and of the prevailing circumstances, including the driver's characteristics such as age, experience, or functional deficits.

We will call this component the *user model*. In order to be truly adaptive, the GIDS system must be able to compose what it learns about, e.g., the driver's state of alertness, or the fact that a particular person habitually accepts a time headway of less than 1 s, so that the output of the user model can be compared with the output from the actual reference situation. While the repertoire of reference situations should be based on a

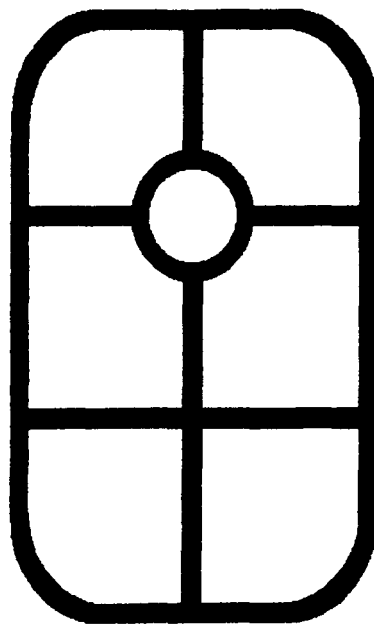


Figure 2. The 'small world' within which the GIDS project will model, specify and evaluate a prototype support system.

detailed analysis of normative task requirements and performance requirements, the user model is a descriptive model that is to a large extent to be based on empirical research.

#### 3.4. *The dialogue controller*

Finally, the GIDS system involves a *dialogue controller* to determine how the information, warning, or intervention that the analyst/planner would decide upon, ought to be communicated to the driver, given the actual circumstances and the state of the driver. For instance, a warning issued but not attended to might be repeated at intervals with increasing urgency. Indeed the impact of voice communication with varying levels of intoned urgency is being studied in the framework of the GIDS project. Apart from its form, the priority level of a message, its size, and the rate of presentation would be under the jurisdiction of the dialogue controller. In addition, it should be mentioned that the driver will be able to address the dialogue controller directly (either vocally or by means of manual action), among other things to set a required level of support or to request support for particular actions, such as merging, or negotiating a roundabout.

### 4. An approach to error

In designing a GIDS system, errors will have to be considered from two points of view, the user's perspective and the designer's perspective. In the first case the system will have to deal with the representation of imminent and actual errors, made by the driver, that GIDS helps this driver to avoid or to recover from. In the latter case the main point of concern is the reliability or error-proneness of the GIDS system as a co-driver. The first category will allow us to determine the level of user performance (in terms of error-proneness and safety), whereas the second perspective is required as soon as the issue becomes one of designing a system that will actually provide driver support in real driving. These are two entirely different issues, and the difficulty of translating insights from one domain to the other is, in our opinion, basic for the troubled interaction between human factors specialists and systems designers (see Michon 1989).

In the following discussion we shall deal with both aspects. First we analyse the GIDS concept in terms of a well-established error theory (GEMS). Such an analysis will provide the background on which to base a performance evaluation of the behaviour of drivers who are actually supported by a GIDS system. Then we will raise some issues that deal with the actual implementation of GIDS systems, particularly the knowledge representations that are required to provide reliable and 'error free' support to drivers, within the operational domain of such implementations.

In the following discussion we will exclude from consideration errors that are the result of explicit malfunctioning on the part of the system, the information channels, and/or the driver.

#### 4.1. *Reliability*

As stated before, the driving task in its entirety is considered far too complex for incorporation into a GIDS system in the short run. Consequently, only a number of subtasks or manoeuvres will be selected. The requirement is that these subtasks can be formalized to a sufficient degree so as to be representable in the database of an operational system.

The knowledge that is thus represented must possess 'closure', in the mathematical sense of the word. That is, neither the circumstances, nor the driver's behaviour, should ever bring the system outside the domain within which it is designed to function properly. Any failure to stay within these boundaries would destroy the purpose of the support system. The closure requirement means that, in order to qualify as a task for GIDS, any permissible action on a permissible state should result in a permissible state. A system for which this does not hold is unreliable; it would involve, for instance, advice about actions from which the driver would not be able to recover. Such would be the case, for instance, if a collision were imminent and the GIDS system would urge the driver to brake to a stop from 100 km/h in, say, half a second. Such advice would clearly fall outside the range of the mechanically possible, and therefore considering this possibility would already constitute an error.

Assuming the availability of suitable subtasks, symbolic task representations will have to be implemented in sufficient detail to allow GIDS to function properly. This does not necessarily require highly sophisticated computational procedures. There might, for instance, be a 'module' that can actually be implemented as a simple look-up table, which would tell the driver that a particular manoeuvre is not going to fit the available time frame (perhaps expressed as time headway). If the GIDS system establishes such a fact, its analyst/planner may then decide to either provide additional information so that the time constraints (decision speed) may be changed, to advise the driver to slow down, or—perhaps—it may take control and brake to a halt. Under all circumstances, however, this decision would have to be consistent with the whole set of possible behaviours and also take into account the relevant aspects of the trip history leading up to the situation to which the information (or the intervention) pertains, and the idiosyncrasies of the driver.

#### *4.2. Driver errors that GIDS ought to detect and correct*

As explained earlier, there are two types of errors which are of concern with a GIDS system. The first are those errors that human drivers would make and which result in unsafe manoeuvres or inefficient driving. The second are those errors made by the GIDS system itself, arising from its problem-solving attempts. There is no single error model which provides a fully satisfactory prediction of the errors that might be made, by both the driver and the system. However, Reason (1987) has described a cognitive framework for identifying common error types which may be useful in this regard. This framework incorporates Reason's earlier work, in which he developed a model describing automated behaviour as well as Rasmussen's distinction between skill-based, rule-based, and knowledge-based behaviour. The question that would need to be answered in our present context is whether we may expect a GIDS system to be better in detecting one or the other type or error. This question can be restated as follows: to what extent does this error typology follow from the functionalities of the system and/or the interfacing between the system and the driver? Only if either of these possibilities applies will the taxonomy be applicable when designing driver support systems.

Reason makes a distinction between slips and mistakes. Slips are 'departures of action from intention', while mistakes are 'errors in which the actions may run according to plan, but where the plan is inadequate to achieve its desired outcome'. Reason also notes that slips, which are primarily related to monitoring failures, precede problem detection, whereas mistakes follow problem detection and are due to

problem solving failures. Reason classifies errors as skill-based slips, rule-based mistakes and knowledge-based mistakes. Skill-based behaviour implies automated behaviour of high quality, where the principal role of the human is to monitor, and thus the principal form of error is a monitoring failure. Rule-based and knowledge-based behaviour imply problem solving activity where errors result from actions that were insufficient to meet the intended goal.

Let us now apply this framework to the three levels at which the driving task is being performed; planning, manoeuvring, and control. In the GIDS project these levels are exemplified by three concrete tasks, namely route finding, obstacle avoidance, and speed and course maintenance.<sup>4</sup> In this section we will deal with the types of errors the driver might make while performing these tasks and that GIDS would need to detect and compensate for (saving for the next section a review of the type of errors which GIDS itself might make in the problem solving process).

At all three task levels a human driver is likely to make slips or monitoring failures.<sup>5</sup> To avoid the consequences of failures to monitor, the GIDS system must be continually engaged in problem solving activity, that is, in matching the reference situation with the actual situation and recovering from mismatches between the two. It is, however, not sufficient to detect the problem caused by a monitoring failure (e.g., passing the intended off-ramp from the highway, imminent collision with the vehicle in front, or imminent departure from the road because the high speed at which one entered the curve was not noticed). Detection is not sufficient, simply because, by the time the failure is detected, it may not be possible anymore for GIDS to satisfactorily solve the problem created. In order to prevent this condition, or at least to diminish the likelihood of its happening, GIDS must be continually informed of how close the driver is to a threshold and not only when he exceeds it.

At all three task levels the human driver is also likely to make mistakes or to fail to solve problems. In the speed control task, drivers may misjudge the speed with which they should enter a particular curve because of an overestimation of their car's steering characteristics, or an overestimation of their own skill. In either case a problem solving failure occurs. Again, GIDS will have to keep one step ahead of the driver and also solve this problem, by determining the highest speed at which the upcoming curve can be entered safely and, for instance, sounding a warning when the driver's speed exceeds this limit. In avoiding obstacles, drivers may misjudge the speed of oncoming cars and attempt a pass that might result in a collision. Here GIDS must use distance and speed data to immediately warn the driver whether or not the pass is safe. In the process of route finding, finally, a driver may incorrectly assume that a certain road is a direct route to his destination, whereas in fact it is not. The navigation system can re-plan the route to take into account this error. In the latter case, the time factor is less critical and continuous monitoring by the GIDS system should therefore not be necessary.

Reason (1987) has not only described the character of errors at each of the three levels of behaviour, he also describes error-shaping tendencies at each of these levels. These could be used to predict the form of both human driver errors and GIDS system errors. In our opinion, however, it is not particularly useful to examine these error-shaping tendencies to predict in greater detail the form of the errors that a human driver will make, particularly at the skill-based level. The most frequent result of an attentional failure while driving, for instance, is the driver continuing with the same reference situation activated, despite the fact it is no longer appropriate, rather than switching to a different but equally inappropriate schema. It is this latter form of error on which Reason's framework is focusing.

#### 4.3. Errors that GIDS would make

Considering now the errors produced by the GIDS system itself, the error-shaping tendencies described by Reason offer useful insights. Here only error-shaping tendencies related to rule- and knowledge-based behaviour are of interest. As noted above, computers—unlike humans—do not make unintended slips. That is, monitoring failures or skill-based slips will not be a problem for a system in good mechanical order. On the other hand, mistakes due to hardware limitations and to designer-induced error are likely to occur. These will be both rule-based and knowledge-based mistakes generated by problem solving activity. These mistakes will have the outward appearance described by Reason. The question that one is facing when designing a GIDS system is how this error phenomenology is actually generated by the underlying mechanisms upon which the system will operate.

At the rule-based level, Reason's error-shaping tendencies illuminate the type of errors that might be expected from a GIDS system.

*4.3.1. 'Mind set' errors.* This is the tendency to use a rule which has been employed in the (recent) past even though it is no longer appropriate. Such an error might occur with a GIDS system where the operators available to the system are insufficient. This would be the case, for instance, if operators allowed only a monotonic decrease in the distance between the intended destination and the actual position. One can imagine a navigational system working on this principle, which would involve setting up subgoals in a particular order, only to have the technique backfire because the particular road layout the driver was operating in required the driver to first drive away from his destination in order to reach it the quickest way.

*4.3.2. Availability.* This biases the actor towards rules that come readily to mind. However, a GIDS system can easily avoid this problem if parallel processing is admitted. Each rule is then given equal consideration.

*4.3.3. Matching bias.* Inappropriate matches are likely to be made on the basis of the similarity between one prominent feature of the current system state and a stored situational representation also possessing this feature. This could be a GIDS error if the system has insufficient knowledge to distinguish between the two system states and if its stored representation possesses too few aspects to allow differentiation. Thus, for instance, a pylon on the road might be mistaken for a bicycle if both produced the same pattern in the obstacle detection system. This is the most serious possibility of error in GIDS. It can be solved in two ways, either by implementing a highly differentiated task description or, alternatively, by providing the GIDS system with a powerful learning mechanism that allows it to increase its powers of discrimination.

*4.3.4. Oversimplification.* Related to the previous point is the possibility of oversimplification. This may definitely count as an error shaping factor for a GIDS system. Knowledge that GIDS has available will always be a much reduced version of that which is available to the human driver.

*4.3.5. Overconfidence.* Due to confirming bias whereby people justify their actions by looking for confirming evidence instead of looking for disconfirming evidence to ensure their judgement is correct. A GIDS system, whether implemented in Soar or not, is not

human. Consequently it will not favour one diagnosis over another, and therefore would not subject to this error shaping factor.

A similar set of considerations applies to the functioning of GIDS at the knowledge-based level.

*4.3.6. Selectivity.* Errors arise if attention is not given to the logically important features of a task, but to the psychologically salient features. This error could occur with a GIDS system if, for example, the detection system were set up to respond to the strongest signals. A driver might be told to move his car to the right to avoid hitting a reflective marker on the centre line, and in doing so might hit a pedestrian. The driver himself might have detected the pedestrian from a combination of subtle cues, none of which would be strong enough to be counted a signal by the obstacle detection system. In this case important cues are missed because less important but stronger signals guide the response of the GIDS system.

*4.3.7. Working memory limitations.* Many driving situations present problems that must be solved very rapidly, usually in less than a second. Since it is likely that working memory of the GIDS system will be limited, GIDS will make errors because of insufficient time. For example, it will probably be impossible to determine whether or not a large piece of cardboard falling from a truck the driver is following is a threat to the driver, before the driver reaches it.

*4.3.8. Out of sight out of mind.* This is the tendency to give undue weight to those facts that come readily to mind and ignore those that do not. This type of error is not the type one would expect to result from computer control. The error described relates to knowledge, operators, or problem spaces that are available, but not being accessed.

*4.3.9. Thematic vagabonding and encystment.* The former term refers to the tendency to move from one problem to another without resolving anything; the latter to the tendency to linger over one problem while ignoring others. Since GIDS will need to operate on a hierarchical basis, giving priority to safety threatening issues first, thematic vagabonding may be said to occur if a succession of priorities occurs, perhaps even as a result of leaving a previous problem unsolved. However, thematic vagabonding implies a lack of direction which is not the cause of the difficulty in this situation. Encystment is not a likely source of errors since priority interrupts will always be allowed to intervene in an ongoing action.

## 5. The formalization of error

### *5.1. The designer's need for syntax of error*

With this inventory of error tendencies to be encountered and coped with by GIDS, we have provided a basis for a policy of performance evaluation, once a GIDS system has become available. What this approach is not capable of supplying, however, are guidelines or requirements for design. Attributing a specific error to a lapse of attention does not tell the designer what the system should do in such a case to avoid, cope, or recover. The contention made here is simply that, for design purposes, an error typology (even if based, as GEMS undoubtedly is, on a psychological theory of information processing) is inappropriate. We leave aside the fascinating possibility that we might be able to design a better GIDS system if we took care to implement the

various human biases that lead to observed errors. Do we perhaps have our biases because we need them to cope with reality?<sup>6</sup> An error theory that would qualify for the stated purpose should resemble a generative grammar or syntax; that is, it should be able to generate particular errors under certain situational constraints, and these efforts should be predictable from the behavioural constraints that are imposed by the syntax itself (as is the case with linguistic errors, such as spoonerisms, which follow from the way this syntax is implemented in the live language user).

In order to be able to reconstruct the rich phenomenology of error that we discussed in the preceding section, we need to be able to formalize knowledge and the rules for manipulating knowledge in terms of specific syntactic rules, or production rules, in such a way that errors of predictable type and predictable amount will indeed materialize in the observed behaviour, when the appropriate circumstances are realized.

In short, errors are treated as events that result from the outcomes of applying rules to knowledge representations. If we wish to take errors seriously, as relevant (systematic and therefore reducible) aspects of our adaptive, intelligent models of behaviour (and more specifically the ones that we are constructing to provide driver support), then these errors must not just fit some intelligent taxonomic scheme, they must be derivable (*a*) from the representation the performer holds about his or her environment; and (*b*) from the operations the performer has available to act upon these representations.

This is essential for a computational approach to error. In this sense we will have to deal with the analysis in section 4 above. This approach, incidentally, is not in opposition to existing theories. The difference is that the latter are specified in terms of intentional (goal-directed) behaviour, not on information processing functions as required by a computational model, viz., a GIDS system. They rest on the so-called rationality assumption: given some knowledge about the performer's goal (i.e., the nature of the task), and some information about the circumstances under which the task is performed, errors can be described and understood as deficiencies of rationality. For instance, an 'error of judgment' does indeed presuppose a failure of rationality, but such an attribution is necessarily *post hoc*. On the other hand, if we look at errors at the lower syntactic level, errors will result from the application of the wrong rule to a database, as a result of specific mismatches between the actual situation and the representation the performer has of it, in brief, the simple kinds of error that belong to the perspective that is provided by control theory and the theory of human problem solving. In other words, the task at hand may be perceived as one of determining what syntactic structure is required to generate the rich error phenomenology that is revealed by such theories as Reason's, which we discussed earlier. This analysis is currently under way, as a part of the GIDS project, but it is too early yet to discuss concrete results. Therefore we can only touch upon a few issues that are indicative of some of the conceptual difficulties we will be forced to consider.

### 5.2. *Soar: a paradigmatic intelligent architecture*

It has not been decided yet in what architecture the GIDS system will eventually take shape. Nevertheless there is an intelligent architecture, Soar—which stands for State Operator And Result—which at least provides a medium for discussing the formalization of the driving task at a level that will be required in a GIDS system. Soar offers just the kind of flexibility that is required for a model of the multifarious and hierarchically structured performance that is characteristic of traffic behaviour. Soar

also qualifies because it is essentially the embodiment of Newell and Simon's (1972) theory of human problem solving, the theory that has been a major source of inspiration for both Rasmussen's and Reason's (1987) theories of human error.

Soar was recently developed by Laird, Newell and Rosenbloom, who have also provided the most explicit review of the Soar architecture to date (Laird *et al.* 1987, Rosenbloom *et al.* 1990).

In the context of this theory, problem solving consists of finding a chain of operators that transform an—undesired—initial state into a desired end state or, in other words, finding the steps that lead to a goal. In driving, for instance, solving a navigation problem means finding the route sequence of decisions about speed and heading to negotiate an intersection; and in terms of control it implies, for instance, selecting a series of correct manipulations that allow gears to be changed without a growling gearbox or a joyless jolt of the vehicle.

The cornerstone of the theory of Newell and Simon (1972) is the problem space hypothesis. This hypothesis holds that all intelligent human behaviour can be described as problem solving, and that problem solving, in turn, can be described as a heuristic search through problem spaces. A problem space is in some respects analogous to what we usually call a task domain. It incorporates all task specific knowledge that a system holds about a particular task. Formally we may consider a problem space as a collection of states and operators, operators being data structures that can modify certain states.

*5.2.1. Operation.* Soar is essentially a production system, consisting of a set of IF-THEN rules and a working memory which can temporarily hold the information to which these rules are applied. When the IF part of a rule is confirmed—which is the case when it matches the contents of working memory—this rule will become active and execute its THEN part, thereby adding new information to its working memory. This will allow the match-and-execute process (called the elaboration phase) to continue with other rules whose IF-part is matching the new situation. Soar is essentially a parallel system. All production rules are tested simultaneously<sup>7</sup> for their truth value and since many rules may be true at the same time they will execute their THEN part in parallel.

Working memory in Soar contains two classes of objects or data structures. The first class consists of problem spaces, states, and operators, the basic elements of the theory of human problem solving. The second class comprises the so-called preferences. A preference determines the applicability of an object from the first class, given certain conditions. Preferences can be absolute, for instance, 'Operator X is acceptable' or 'Operator Y is rejected', or relative, for instance, 'State S is better than State T'. A simple preference semantics specifies the relations between preference terms such as 'best', 'acceptable', 'worse', and so on.

On the basis of an evaluation of the preferences of all active, potentially applicable operators, a decision is made about the preferred action to take. During this decision phase an impasse may occur, for instance when two operators are found that both have a preference 'best', or when all operators have a preference 'reject'.

Given a particular goal to achieve, Soar will first make an attempt, on the basis of its available knowledge (productions) to find a problem space in which it can reach this goal. If a suitable problem space is found (by a match-and-execute cycle as defined above), Soar will make an attempt to represent its current state in this problem space and subsequently search for operators that can transform the current state into a new

state, closer to the goal state. If Soar encounters a difficulty, for instance because it cannot choose from among several, equally attractive, 'best' operators, a process of subgoaling will be initiated that is specifically directed at solving the particular difficulty encountered. If the subgoaling process meets with success, perhaps only after a continued series of subgoaling steps, Soar will learn from this experience how to avoid the difficulty in the future, by adding a new production to its rule base that specifies the action to be taken if the same problem would arise again. This process is known as chunking and is central to the operation of Soar: Soar is a relentless learner.

*5.2.2. Application.* Soar as a theory is evidently quite general. Consequently it should apply also to the strategic, the tactical, and the operational subtasks of the driving task (Michon 1987). These are the levels considered in detail in the GIDS project. At the strategic level GIDS studies the problem of navigation support, that is, helping the driver to find a route on an unfamiliar (or changing) road network. The states in this problem space are all the locations where one could possibly be at some given time and the operators to be handled by GIDS are the directions from which one may choose. At the manoeuvring level GIDS is dealing with obstacle avoidance. States are, in this case, the critical spatio-temporal configuration of objects in the traffic environment, including road furniture and other road users; the support problem being to determine who and what will be where and when, to the driver's own vehicle? The operators in this case consist of the potential changes in speed and heading. Finally, at the operational level, heading and accelerator support is considered in the context of GIDS. For this kind of support, states and operators need to be defined in a different way than at the manoeuvring or strategic level because the time constants involved are well below 1 s in some cases. An intelligent accelerator will provide kinaesthetic feedback to the driver, requiring more or less force to depress the pedal, rather than providing the driver with explicit messages about the best way to treat the pedal.

*5.2.3. Error.* Soar deals with the representation of the external world at four levels—goals, problem spaces, states and operators. Barring 'errors' resulting from capacity or speed limitations, an error theory in Soar should, in principle, be determined simply by the occurrence of a pattern mismatch at any of these four levels. A goal mismatch would result from a discrepancy between the definition of the intended desired state and that of the actual state that is pursued, suggesting errors that are generated at the planning level. An example would be travelling to The Hague to see the Rijksmuseum (the desired state). The issue is not as simple as that, however, since it is in the nature of Soar that goals are generated internally (except the top goal). This raises the question as to whether goals, as such, can indeed be mismatched at all: under what circumstances is it conceivable that someone would drive to Oxford if her goal is to get to Cambridge.

Mismatch at the problem space level seems to constitute a more plausible condition. It implies that a desired state (the goal state) cannot be reached (*a*) as a consequence of the fact that a problem space of which the desired state is not a part has been selected by the system, or (*b*) because there is no legitimate chain of operators that will lead to the desired state. The operation of the Soar architecture as such—or of any other intelligent architecture, for that matter—may also introduce errors. Subgoaling to solve an impasse may, indeed, lead to errors of control. Many complex tasks require that the performer be able to switch, at any given moment, temporarily from one subtask to another, and to return a short while later to the exact point where he or she left off. In the first place, this creates the need to remember exactly where one did leave

off and, in the second place, circumstances are likely, in this kind of complex task, to change in the course of the interval during which the operator was attending the other subtask. In the latter case the question becomes one of being able to extrapolate quickly and accurately where one must pick up the ongoing chain of events. This is an important point that creates a number of serious computational problems if we wish to deal with it in a driver support system.

### Acknowledgement

This research derives in part from the DRIVE Project VI041 under the title Generic Intelligent Driver Support (GIDS) under contract with EEC. It was also stimulated by a study contract with IBM Nederland.

### Notes

1. On sabbatical from Human Factors North, Inc. Toronto, Canada
2. In the paper by Brown for this issue (Brown 1990), the value of this approach for our insight into human error is questioned, (p. 1307).
3. The GIDS project (DRIVE Project V 1041) is carried out by a consortium of 12 partners, INRETS-LEN (F), TNO Institute for Perception (NL), MRC Applied Psychology Unit (GB), Philips Research Laboratories (NL) Renault (F), Saab-Scania (S) Traffic Research Centre, Groningen (NL), Technical University Delft (NL), University College Dublin (EIR), University of the Bundeswehr, München (D), VTI Swedish Road and Traffic Research Institute (S), and Yard Ltd (GB). The project is co-ordinated by the senior author of this paper.
4. Hale (1990), elsewhere in this issue, clarifies the distinction between planning, manoeuvring, and control on the one hand, and knowledge-based, rule-based, and skill-based behaviour on the other.
5. It should be noted that not all failures to monitor are due to slips, some are due to mistakes and betray an inadequate problem solving strategy on the part of the driver.
6. We owe this question to Dr John Duncan, MRC APU, Cambridge, UK; see also Duncan (1990), this issue, p. 1265.
7. Depending on the architecture of the computer on which Soar is implemented, this parallel testing may actually be performed sequentially.

### References

- CARD, S. K., MORAN, T. P. and NEWELL, A. 1983, *The Psychology of Human-Computer Interaction* (Lawrence Erlbaum Associates, Hillsdale NJ).
- HALE, A. R., STOOP, J. and HOMMELS, J. 1990, Human error models as predictors of accident scenarios for designers in road transport systems, *Ergonomics*, 33, 1377.
- HARVEY, C. F., JENKINS, D. and SUMNER, R. 1975, *Driver Error*, Technical Report SR149 (Transport and Road Research Laboratory, Crowthorne, UK).
- JANSSEN, W. H. 1990, *The Impact of Collision Avoidance Systems on Driver Behavior and Traffic Safety: Preliminaries to Studies within the GIDS Project*, DRIVE Project V 1041: Generic Intelligent Driver Support Systems Deliverable Report 1041/MAN1 (Traffic Research Centre, University of Groningen, Haren, NL).
- LAIRD, J. E., NEWELL, A. and ROSENBLOOM, P. S. 1987, Soar: an architecture for general intelligence, *Artificial Intelligence*, 33, 1-64.
- McKNIGHT, A. J. and ADAMS, B. B. 1970, Driver Education Task Analysis, Volume I: Task descriptions, Final Report, Contract Nr. FH 11-7336 (Human Resources Research Organization, Alexandria, V A).
- MICHON, J. A. 1971, *Psychonomie onderweg*. Inaugural lecture, University of Groningen (Wolters-Noordhoff, Groningen).
- MICHON, J. A. 1985, A critical view of driver behavior models: What do we know, what should we do? In L. A. Evans and R. C. Schwing (eds), *Human Behavior and Traffic Safety* (Plenum Press, New York), 487-525.
- MICHON, J. A. 1987, On the multidisciplinary dynamics of traffic science, *IATSS Research*, 11, 31-40.

- MICHON, J. A. 1989, Explanatory pitfalls and rule-based driver models, *Accident Analysis and Prevention* 21, 341-353.
- NEWELL, A. and SIMON, H. A. 1972, *Human Problem Solving* (Prentice Hall, Englewood Cliffs, NJ)
- REASON, J. 1987, Generic error-modelling system (GEMS): a cognitive framework for locating common human error forms. In J. Rasmussen, K. Duncan and J. Leplat (eds), *New Technology and Human Error* (John Wiley, Chichester), 63-83.
- ROSENBLOOM, P. S., NEWELL, A. and LAIRD, J. E., Towards the knowledge level in Soar: the role of the architecture in the use of knowledge, In K. van Lehn (ed.), *Architectures for Intelligence* (Lawrence Erlbaum Associates, Hillsdale, NJ) (in the press).
- ROTHENGATTER, J. A. (00.) 1990, *Navigation Information Requirements: A Literature Review*, DRIVE Project V 1041: Generic Intelligent Driver Support Systems—Deliverable Report 1041/NAVI (Traffic Research Centre, University of Groningen, Haren, NL).
- SMILEY, A. and MICHON, J. A. 1989, *Conceptual Framework for Intelligent Driver Support*, DRIVE Project V 1041: Generic Intelligent Driver Support Systems—Deliverable Report 1041/GEN1 (Traffic Research Centre, University of Groningen, Haren, NL).

Les progrès technologiques récents semblent avoir contribué au développement de systèmes électroniques sophistiqués qui pourraient permettre aux conducteurs de voiture de faire face à une charge informationnelle croissante, d'éviter certaines erreurs et de parer certaines autres. GIDS (Generic Intelligent Driver Support) est un projet de recherche afférant au programme DRIVE de la CEE et qui vise l'élaboration d'un système de conduite assistée adaptatif. La conceptualisation du système GIDS requiert une prise en compte étroite des défaillances possibles dans certaines sous-tâches impliquées dans la conduite d'une voiture. Un aspect important qu'il ne faudra pas négliger est que GIDS peut éliminer des erreurs, tout aussi bien qu'en introduire d'autres. Faut-il présenter formellement les différents types d'erreurs? Et, si oui, comment faudra-t-il les présenter afin que GIDS puisse détecter et faire face aux défaillances comportementales dont sont sujets les conducteurs, dans certaines circonstances? Ces exigences imposent une lourde contrainte à la définition et à l'identification précise de l'erreur. Certaines de ces contraintes seront prises en compte par Soar qui est une structure informatique intelligente basée sur la théorie de la résolution des problèmes formulée par Newell et Simon (1972). Dans la mesure où la tâche de conduite est formulable par Soar, la théorie des erreurs requise pour la mise en œuvre des systèmes GIDS doit également être formulable par Soar.

Jüngste technologische Entwicklungen scheinen den Weg für hochentwickelte elektronische Beifahrersysteme zu bahnen, die Autofahrer helfen, mit einer immer weiter steigenden Informationsmenge fertig zu werden, bestimmte Fehler zu vermeiden und sich von anderen zu befreien. GIDS—was für Generic Intelligent Driver Support steht—ist ein Forschungsprojekt (innerhalb des CEC DRIVE Programms) zur Untersuchung der Möglichkeiten eines adaptiven Beifahrersystems. Die Konzeptualisierung eines GIDS Systems benötigt eine genaue Beachtung von Fehlern in der Leistung, die in bestimmten Teilaufgaben der Fahraufgabe auftreten können. Ein wichtiger Punkt der näher berücksichtigt werden sollte, ist die Tatsache, dass GIDS sowohl Fehler beseitigen als auch auf Fehler hinweisen kann. Sollen verschiedene Fehlertypen *formal* dargestellt werden und falls ja, in welcher Ordnung sollten sie dargestellt werden, damit GIDS Verhaltensfehler, die von Fahrern unter bestimmten Bedingungen gemacht werden, entdecken und mit ihnen fertig werden kann? Aus den Anforderungen, die durch das Ziel des Projektes entstehen, verschiedene Funktionen zur Unterstützung des Fahrers in ein GIDS zu implementieren, ergeben sich enge Vorgaben für die Fehlerdefinition und -identifikation. Einige dieser Anforderungen werden unter dem Gesichtspunkt von Soar besprochen. Soar ist eine intelligente Rechnerarchitektur, die die Anwendung der von Newell und Simon (1972) formulierten Theorie des menschlichen Problemlösens (*human problem solving*) darstellt. Bis zu dem Umfang, in dem die Fahraufgabe in Soar darstellbar ist, muss auch die Fehlertheorie die für jedes funktionierende GIDS-System benötigt wird, in Soar darstellbar sein.