Action reversibility in human-machine systems

Sergio Pizziol, Catherine Tessier, Michael Feary, Fédéeic Dehais

Action reversibility in human-machine systems

Sergio Pizziol
ISAE
sergio.pizziol@gmail.com

Catherine Tessier
ONERA
Catherine.Tessier@onera.fr

Michael Feary
NASA Ames
michael.s.feary@nasa.gov

Fédéric Dehais
ISAE
frederic.dehais@isae.fr

ABSTRACT

This paper focuses on the reversibility of human actions in the frame of human-machine interaction, with a special focus on the interaction between a pilot and a flight management system controlling an aircraft. A multi-level reversibility scale is defined for human actions. A reversibility property is defined for each level. An algorithm implementing a reversibility property check on the machine logic described in ADEPT is proposed. Specifically this paper describes a method for formally identifying actions that are not reversible within one step, that are eventually totally unrecoverable and that are totally unrecoverable.

CCS Concepts

•**Human-centered computing** → *Heuristic evaluations;*

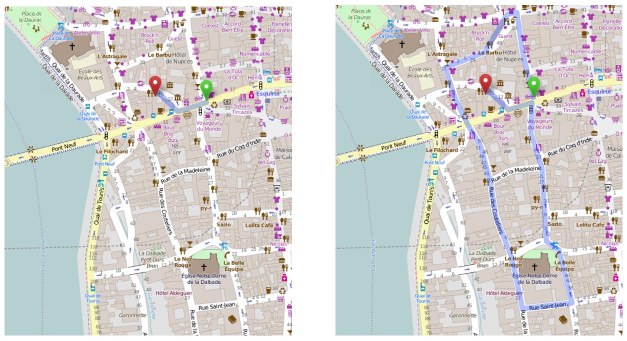
Keywords

Reversibility, human-machine interaction, undo.

1. INTRODUCTION

Suppose you are driving in down-town Toulouse, more precisely on Rue de Metz (near Esquirol metro station, see figure1a point A) heading west. You want to reach Rue de l'Écharpe (see figure1a point B), so the second street to the right. That is a 100 m trip. Suppose you turn on the first street to the right by mistake and immediately notice your mistake: you will reach the destination in 1.3 km (see figure1b). This happens because of the one-way streets: you cannot perform a U-turn in order to recover the situation where the error occurred. Many cities have one way systems like this one.

Whether by omission or commission, mistake or slip [Rea90], errors are an everyday part of human existence. While it is possible to design systems to be less vulnerable to errors, the systems also require the ability to gracefully recover from errors when they do happen. The process of error recovery (i.e. detection, explanation, correction) [AW97, Kon99] could be achieved only if the machine logic allows the reversibility of erroneous actions. For that reason the analysis of the ease of recoverability could be very useful in the evaluation of designs, particularly for the design of safety critical systems like the interaction between a pilot and a flight management system.
While a complete assessment of recoverability is beyond the current scope, this paper focuses on the reversibility of commands as one dimension of recoverability. Specifically this paper describes a method for formally identifying actions that are not reversible within one step, or are irreversible.



(a) Shortest way: 100 m. (b) Turn on the first right, length: 1.3 km.
Figure 1: (Map data: @OpenStreetMap contributors)

Reversibility [CCH08] is generically meant as the property to undo the effects of some action after the execution of a sequence of actions. When defining the reversibility property, it is important to define the domain of interest that is relevant for the designer's purposes. If time is an explicit state variable, all the actions are non reversible. The same is true if some of the state variables are monotonous functions of time. Using aviation as an example domain, there are many action sequences that are not reversible due to the passage of time (e.g. the fuel level always decreases, distance to destination should decrease, etc.). In the same way, there are aircraft automation modes that are irreversible due to the time dynamics.

In the next section the context is presented: a brief review of the existing literature is carried out. Some useful definitions for the following sections are given and the underlying structure of the human-machine system is presented.
Then a six-level reversibility scale is defined for human action-driven state changes (Undo, One action reversibility, Reversibility, Irreversibility, Eventually totally unrecoverable, Total unrecoverability). One reversibility property corresponds to each level. It is shown that the set of state changes verifying higher level properties is included in the sets of state changes verifying lower level properties.

© 2015 ACM. ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

An automated test to assess the reversibility degree of each state change is formalized next. The test can verify four out of six properties (Undo, One action reversibility, Eventually totally unrecoverable, Total unrecoverability). The check of the last two properties (Reversibility, Irreversibility) is out of the scope of this paper. The results of the automated test are presented for a toy example.
In the last section the automated test is performed for the reversibility of the Go-around mode engagement for an autopilot simplified machine logic.

2. CONTEXT

A *human–machine system* (see figure 2) is a two-agent team formed by a *human operator* and a *machine* with a common *goal* [Jen95], they *communicate* and act on a *physical system* (or just *system*) for the achievement of their goal. Note that we call *machine* only the part of the automation logic structure and current state the human operator has to know in order to operate effectively.
The goal achievement is pursued through the execution of *functions* [MC99]. Some of those functions can only be executed by the human operator, some only by the machine. In this work we focus on the case of a pilot interacting with a flight management system in order to control an aircraft. Nevertheless the concepts developed in this paper are applicable in the wider context of human-machine systems.

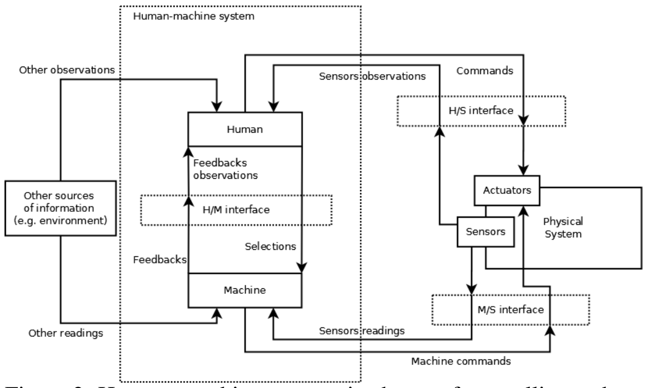


Figure 2: Human–machine systems in charge of controlling a physical system

The human/machine interface (H/M interface) is composed by input devices (e.g. buttons, knobs etc.) and output devices (e.g. displays, visual and aural alarms etc.).
Feedback is the machine to human communication through the human/machine interface. In the same way the *selections* are human actions that constitute the human to machine communication through the human/machine interface.
In principle feedback allows a certain degree of observability on the state of the machine so that the human could make *observations*; selections should affect the state. State changes that are the result of selections are *selection triggered transitions*. We call *automated transitions* state changes that are not caused by selections.
The human/physical system interface (H/S interface) is composed of input devices (e.g. control stick, yoke, control pedals, etc.) and output devices (e.g. sensors displays, control stick with retrofeedback etc.).
The machine/physical system interface (M/S interface) is composed of the software and the hardware systems meant to provide the connection between the machine and the sensors and actuators of the physical system.
We define the *sensors* as a part of the controlled physical system providing information about the physical system state itself through the H/S interface and through the M/S interface. For instance we consider a Global Positioning System (GPS) as a sensor.
We define the *actuators* as parts of the controlled physical system that can change the system state and are controlled through the H/S interface and the M/S interface. We consider the servoing systems (as the automatic control loops) as part of the actuators.
Commands are human actions on the H/S interface meant to control the actuators. Similarly *machine commands* are data sent from the machine to the actuators through the M/S interface.
Sometimes, for ergonomics reasons, a single physical action may embed *commands* and *selections*: that is the case for an operator taking the authority on the steering wheel of a unmanned ground vehicle (a *selection*) and controlling the direction of the vehicle (a *command*) with a single movement on the control stick. Sometimes the same display may be part of the H/M interface and the H/S interface at the same time.
We call *other observations* all other pieces of information the human receives neither from sensor observations nor from feedback (e.g. a visual estimate of the altitude based on the observation of the ground, communications received by the radio coming from the air traffic control (ATC)). Similarly we call *other readings* all other pieces of information the machine receives neither from sensor reading nor from selections (e.g. radio communication from the traffic collision avoidance system (TCAS) of another aircraft).
A function may be performed by the human or by the machine (respectively *human-function* and *machine-function*).

For a wise function allocation the HMS (Human Machine System) designer should keep in mind the human and machine strengths and weaknesses. The machine is good at performing repetitive tasks and at quick and precise computation, but is bad at unexpected event management [LSMC10]. The human is usually better in *adaptability*, i.e. in failure and unexpected event management and in coping with uncertainty and lack of information [Ras83].
Nevertheless unexpected HMS performance degradations may arise because of errors [PSW00], bad communication [DCV14] and bad coordination between the human and the machine [Wic05, SMW07, DCT11, PDT11, PTD14, TD12, Deh02] or automation surprises [SW95, DPS15], i.e. the wrong assessment of the states, due to an unexpected automated state change. Moreover some works have focused on formal properties of the machine logic that could reveal bad designs, which could in turn cause human errors. For instance [JMDK00] define objective properties (e.g. the presence of a feedback for a function) and subjective properties (e.g. the learnability of a function) for the functions of the interaction, [LPS97, Fea05] define critical transitions for the state machine model (e.g. an automated transition, transitions with similar feedback), [DH02] evaluate whether the interface provides the necessary information about the machine, so as to enable the operator to perform a specified task successfully, [CCH08] define patterns for potential unforeseen interaction problems and discover discrepancies between the user manual and the real machine behaviour, [PH00, Piz13] develop action-based models to detect dangerous situations, [Pal97] compares and evaluates different formal approaches.

While all those studies have contributed to improve the human-machine interaction, users are fallible [Rea90] and the total elimination of errors is an unrealisable ambition. Therefore interfaces have to take into account and exhibit error resistance i.e. prevent users from making errors, e.g. a human/machine interface

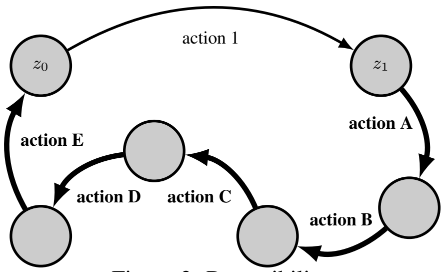


Figure 3: Reversibility

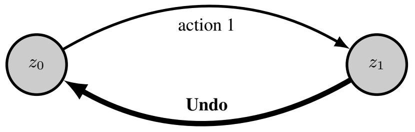


Figure 4: Undo

that requires more than one selection to perform a bank transfer, [MDK93] and error tolerance: errors should be reversible (e.g. a bank web portal that allows a bank transfer to be nullified within 30 minutes).

For instance in the NASA ASRS database 21 safety incident/situation reports (in the period 2001-2014) are related to situations in which the pilots performed an involuntary action whose effects they tried to reverse [NAS]. More precisely those involuntary actions made the automation drop some safety constraints (e.g. requirements on the minimal altitude).

In the next section a six-level reversibility scale is defined for human action-driven state changes.

3. DEFINITIONS AND SIX-LEVEL REVERSIBILITY SCALE

3.1 Reversibility

Definition [Reversibility][CCH08]: a state change from state z_0 to state z_1 triggered by action 1 is said *reversible* if from state z_1, it exists at least one sequence of actions (action A, action B...) that at some point in the future can take back the state to the initial state z_0, see figure 3.

3.2 Undo

Definition [Undo](inspired by [CCH08]): The *undo by a specific action*, is a special case of the reversibility definition: the sequence of actions is replaced by a unique action, always the same whatever the action to undo (see figure 4).

The *undo* is desirable for non critical domains, as for instance text editor software. Nevertheless, even for this application, predictability issues arise: *despite the general agreement of the importance of undo, few systems supply more than the simplest single-step undo command and, even then, the effect of command and when it can be applied is often far from obvious. (...) Undo support for single user systems is regarded as essential, but recognised to be fraught with potential pitfalls* [AD91].

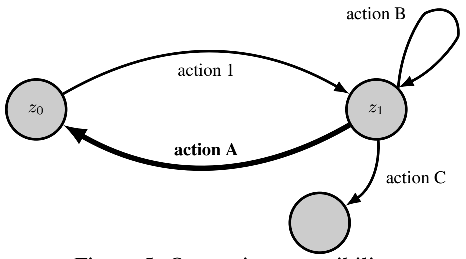


Figure 5: One action reversibility

In the literature the emphasis is put on action triggered state changes whose consequences, for the same action, are not always the same: they are defined as “inconsistent behaviour” [LPS97, CCH08] or also “moded behaviour” [Fea05, Fea07]. They are widely recognized as state changes that may be misunderstood by the operator.
The *specific action* is always the same whatever the action to undo, for instance to press a specific *undo button*, or to enter a specific key stroke combination (e.g. *control+Z*). The last performed action is in general not shown on the interface: the human operator has to remember it in order to predict the effect of the *undo*. For that reason the *undo* is in general an “inconsistent behaviour”.

For safety critical domains such as aeronautics predictability is highly desirable [Fea05]. So the *undo*, because of its predictability issues [AD91, LPS97, CCH08, Fea05], is not a desired feature. The absence of the *undo* function does not lead to the loss of reversibility. For instance it is enough to show that any action can be undone simply by a proper re-action.

Definition [One action reversibility] (inspired by [CCH08]): a state change from state z_0 to state z_1 triggered by action 1 is said *one action reversible* if from state z_1, it exists at least one single action (action A) that can take back the state to the initial state z_0, see figure 5.

Example: Consider a machine with a three-value state variable *high/medium/low* and three actions *up/down/undo*. The initial state is *medium*. After performing an *up* action the new state is *high*. To reverse the effects of this action it is possible to perform the *undo* action. To obtain the same result it is also possible to execute the *down* action. In the first case *the effects of the up action via a specific action (undo) are nullified*, in the second one *the effects of the up action are reversed via a relevant reaction (one action reversibility)*.

3.3 Unrecoverability

Definition [Irreversibility]: The execution of an action from a state z_0 can lead to a state z_1 from which there is no possible way back to z_0. Such a state change, according to definition [Reversibility], is *irreversible*, see figure 6.

Definition [Total unrecoverability. 1]: The execution of an action from a state z_0 can lead to a state z_1 from which there is no possible way out, i.e. there is no further sequence of actions that can lead to a new state that is different from z_1. z_1 is a *blocking state* (figure 7). Such states could correspond to a physical limit of the machine or a machine failure. They can also be the result of a design error: this is typically the case of machine deadlocks after the execution of an unexpected sequence of actions. Those state changes are said to be *totally unrecoverable*.

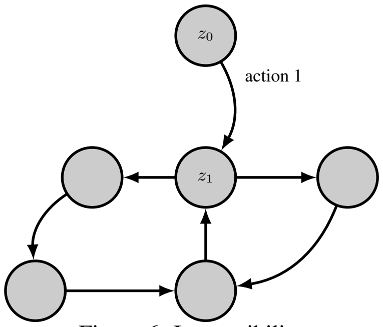


Figure 6: Irreversibility

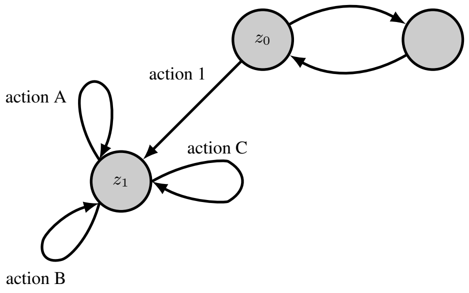


Figure 7: Total unrecoverability

Example: let us consider a machine with a three-value state variable *high/medium/low*, a two-value state variable *ok/out of order* and three possible actions *Up/Down/Oops! (put out of order)*. Initial state is *(medium, ok)*. After the execution of the action *Oops! (put out of order)* the state becomes *(medium, out of order)*. At that point there is no possible sequence of actions to change the state (in this simplified model there is no *repair* action).

Another definition that is equivalent to the previous one is:

Definition [Total unrecoverability. 2]: a state change triggered by an action 1 from a state z_0 resulting in the new state z_1 is said *totally unrecoverable* if from z_1 there is no action that can lead to a state that is different from z_1.

This second definition is easier to verify because it only needs the evaluation of the state after the execution of just one action. Therefore it will be used in the algorithm to verify the *total unrecoverability* property.

And finally:
Definition [Eventual total unrecoverability]: a state change triggered by an action 1 from a state z_0 may lead to an *impasse state* ($\neq z_0$) from which any possible action leads to a *blocking state* or to another *impasse state* (figure 8). This state change is called *eventually totally unrecoverable*.
The detection of *eventually totally unrecoverable state changes* will need an iterative process (see section 4.2.2).

3.4 Reversibility scale

Different types of reversibilities and irreversibilities have been defined in the previous sections. Table 1 summarizes as a scale the different cases of reversibility from the "most reversible" to the "less reversible".
If we note as $\subset$ "is a particular case of", we get the following relations:

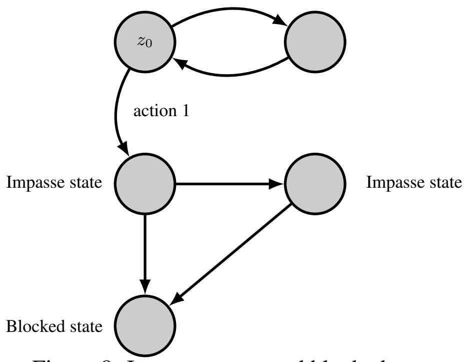


Figure 8: Impasse states and blocked state

Case	Reverse $z_0 \to z_1$
Undo	A particular action to come back to z_0
One action reversibility	One action to come back to z_0
Reversibility	A sequence of actions to come back to z_0
Irreversibility	No sequence of actions to come back to z_0
Eventually totally unrecoverable	Eventually no action to leave some state from z_1
Total unrecoverability	No action to leave z_1

Table 1: Reversibility scale

Undo $\subset$ *One action reversible* $\subset$ *Reversible*

Totally unrecoverable $\subset$*Eventually totally unrecoverable* $\subset$ *Irreversible*

Combining them in just one formula:
$\neg$ *Undo* $\supset$ $\neg$ *One action reversible* $\supset$ *Irreversible* $\supset$ *Eventually totally unrecoverable* $\supset$ *Totally unrecoverable*

or also:

Undo $\subset$ *One action reversible* $\subset$ *Reversible* $\subset$ $\neg$ *Eventually totally unrecoverable* $\subset$ $\neg$ *Totally unrecoverable*

4. AN AUTOMATED TEST FOR REVERSIBILITY ASSESSMENT WITHIN ADEPT

There are many formal models of human-automation interfaces, however the vast majority of them are computationally equivalent to a finite state transition system or finite state machine [BBS13]. Among such models ADEPT (Automation Design and Evaluation Prototyping Toolset) [Fea05, Fea07] is well suited to multimodal systems as the modes can be easily represented as system state variables. Those state variables constitute preconditions that allow or forbid state transitions involving other state variables. Moreover the ADEPT model may be made smaller and more efficient by combining state transitions that lead to the same final state [Fea05], therefore it can partially address the problem of big size systems representation. ADEPT performs a set of automated analyses on the structure of the human-machine interface in order to verify properties like *completeness* and *consistency* in addition to vulnerability checks detailed hereafter. However it does not perform any reversibility verification [CCH08]. Our contribution is to define an additional algorithm within ADEPT so that it can evaluate a set of reversibility properties, as defined previously.

4.1 ADEPT

ADEPT checks a number of properties that might flag potential vulnerabilities of the human-machine interaction:

Moded input: the same action performed by the human operator has many possible effects, depending on the state of the machine.

Armed behaviour: the effects of an action of the human operator may be delayed.

Automated behaviour: a state transition that does not need an action of the human operator to be triggered.

Inhibited behaviour: an action of the human operator that has no effect on the machine state.

Similar feedback: the same display is used for more than one behaviour of the machine state.

The following definitions are inspired by [Fea05].

Definition [Machine, Machine model, State variable, Transition]: in the human-machine interaction the human operator has to know part of the automation logic structure and current state in order to operate effectively. In this work we call *machine* this part of the automation. The *machine* is modelled as a finite-state machine (FSM). Within the FSM a *state variable* is one of the variables that are used to describe the state of the machine. Each state variable may accept two or more values. The machine is in only one state at a time, that state is described by the values taken by its state variables. A state change can occur when an event happens (e.g. a condition is verified); this state change is called a *transition*. Note that the machine model logic representation in ADEPT is provided in the form of a *logic table* (for the relevant definition see hereafter) that is a compact equivalent of an FSM.

Definition [Input/Output state]: given a transition an *input/output state* is the state before/after the firing of the transition.

Definition [Selection]: a *selection* is a human action that constitutes the human to machine communication through the human/machine interface.

In ADEPT state transitions are represented as triples (input state, selection, output state). If the state change does not need a selection to be triggered, the triplet is noted (input state, "no selection", output state).

Definition [Situation]: a *situation* is defined as the conjunction between a proposition about the *selections* and a proposition concerning the *input states*: *selections* are described as a disjunction of selections and input states are described as a conjunctive normal form, i.e. a logic conjunction between the state variables and a disjunction of values of the same variable.
Selections and *input states* are either defined explicitly or take the parametric value "no matter which value/no matter which action (noted as **)".

Definition [Behaviour]: a *behaviour*, which is the result of a situation, is defined as a logic conjunction between state variables that take just one value at a time. This value is either defined explicitly or take the parametric value "same as input (noted as *)".

The link between situation and behaviour is summarized on figure 9.

Example:

$$situation : (action = [a_1 \vee a_3]) \wedge (x_1 = [v_{11} \vee v_{12}]) \wedge$$

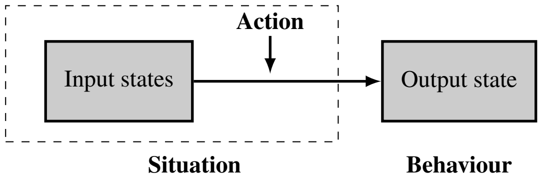


Figure 9: Situation and Behaviour in ADEPT.

$$\wedge (x_2 = [**])$$

$$behaviour : (x_1 = [*]) \wedge (x_2 = [v_{22}])$$

In this example the situation is expressed in natural language as: "action is either a_1 or a_3 and variable x_1 value is either equal to v_{11} or v_{12}, and variable x_2 takes any value". The behaviour is expressed as: "variable x_1 value is the same as input and variable x_2 value becomes v_{22}".

Definition [Logic table]: The set of pairs (situation, behaviour) is represented in ADEPT in a compact table called *logic table*. The first column of the table contains actions and state variables names, the second column contains actions and state variables values. From the third column, each column represents a pair (situation, behaviour), consequently pair number 1 will be represented in the column called c1. In those columns an empty box is set to 0 (or false). If for some *situation* all the boxes corresponding to the actions or a state variable are empty, the action or variable takes [**]. If for some *behaviour* all the boxes for a state variable are empty, the variable has the same value as the input [*].

Example A (see table 2):

Let us consider a machine that is entirely represented by a three-value state variable *X* and a two-value state variable *Keyboard*. *X* can take the *High/Medium/Low* values. The human operator can change the value of *X* through the keyboard, which can be *Ok* or *Out of order*. The human operator's actions on the keyboard are *Increase X*, *Decrease X*, *Oops!* (action *Oops!* represents the accidental breakage of the keyboard). The behaviour of the machine is shown in table 2. Note that in this representation the transition labels represent the column of the logic table they embody, not the transition name.
The logic table columns (c1 to c8) representing the pairs (situation, behaviour) are as follows:

- from c1 to c6: user increases or decreases X.
- c7: accidental breakage of the keyboard, from this point it is out of order.
- c8: in the case the keyboard is out of order, any action of the human operator has *no effect* (i.e. all the variables keep the same value as the input).

Regarding columns c3, c4 and c8 their behaviour is *no effect*. For instance, column c8 represents the following pair (situation, behaviour):

$$situation : (action = [**]) \wedge (X = [**]) \wedge$$

$$\wedge (keyboard = [Out\, of\, order])$$

$$behaviour : (X = [*]) \wedge (keyboard = [*])$$

		c1	c2	c3	c4	c5	c6	c7	c8
SITUATION		Increase X starting from Low	Increase X starting from Medium	Increase X starting from High	Decrease X starting from Low	Decrease X starting from Medium	Decrease X starting from High	Accidental breakage of the keyboard	Any action when keyboard out of order
Actions									
	Increase X	1	1	1					
	Decrease X				1	1	1		
	Oops!							1	
	No action								
State									
Keyboard	Ok	1	1	1	1	1	1	1	
	Out of order								1
X	High			1			1		
	Medium		1			1			
	Low	1			1				
BEHAVIOUR		X actually increases	X actually increases	no effect	no effect	X actually decreases	X actually decreases	Keyboard is out of order	no effect
State									
Keyboard	Ok								
	Out of order							1	
X	High		1(ok)	1(ok)					
	Medium	1(ok)	1(a)				1		
	Low	1(c)		1(b)		1			

Table 2: ADEPT Logic table for Example A

The natural language description of this pair (situation, behaviour) is *if the keyboard is out of order any action for any value of X has no effect*.
For more details on the construction of a logic table in ADEPT see [Fea05, Fea10].

Definition [Complete logic table]: A *logic table* is *complete* if each combination (input state, action) is listed in at least one *situation*.

Definition [Input consistent logic table]: A *logic table* is *input consistent* (or simply *consistent*) if each combination (input state, action) is listed in at most one *situation*.

Consequently, in a *complete and consistent logic table*, for any combination (input state, action) there is always one and only one *situation*, and one resulting *behaviour*. Note that in a *complete logic table* any combination (input state, action) must be explicitly listed in the table, including those whose behaviour is *no effect*.

4.2 Towards reversibility assessment with ADEPT

4.2.1 Definitions

As in ADEPT each pair (situation, behaviour) may represent several state changes at the same time (e.g.: $situation : (action = [a_1]) \land (X = [v_1 \lor v_2])$, $behaviour : (X = [v_3])$) represents the transitions $t_1 : action = [a_1] \land X = [v_1] \rightarrow X = [v_3]$ and $t_2 : action = [a_1] \land X = [v_2] \rightarrow X = [v_3]$) the formal definitions of reversibility need to be adapted, according to a conservative choice:

Definition [Reversibility property]: We define generically as a reversibility property each of the already defined properties: Undo, One action reversibility, Reversibility.

Definition [Irreversibility property]: We define generically as an irreversibility property each of the already defined properties: Irreversibility, Eventually totally unrecoverable, Total unrecoverability.

Definition [Pair reversibility property in ADEPT]: a pair (situation, behaviour) satisfies a given reversibility property when all the state changes represented by the pair satisfy this property.

Definition [Pair irreversibility property in ADEPT]: a pair (situation, behaviour) satisfies a given irreversibility property if at least one state change represented by the pair satisfies this property.

4.2.2 Algorithm

The verification of the *reversibility* property needs the evaluation of the effects of an unknown-length sequence of actions. In this algorithm we have decided to evaluate the effects of one action and one further action to reverse the effects of the first one. That greatly reduces the algorithm complexity. Consequently we will check the *undo*, *one action reversibility*, and *total unrecoverability* properties. The *eventual total unrecoverability* property check is performed as well. Therefore the algorithm for reversibility assessment in ADEPT is shown in Algorithm 1.

4.3 Algorithm applied on example A

```
Data: Define the logic table for the machine logic in the form
      of N pairs (situation, behaviour). One of the actions
      may be labelled as Undo action
Result: The evaluation of the reversibility properties for each
      pair (situation, behaviour): Undo, One action
      reversible, Non one action reversible, Eventually
      totally unrecoverable, Totally unrecoverable.
Prepare a list of all the possible states (for instance as a
Cartesian product of the state variables), each of them will be a
called state instance;
for every pair (situation, behaviour) do
    if any of the state instances is a valid input state for this
    pair, i.e. is included in the relevant situation then
        The state instance is called $z_0$ and stored for later use;
        Compute the resulting output state, called $z_1$, store for
        later use;
        Verify which pairs have a situation that includes $z_1$. $z_2$
        is the resulting output state for those pairs (a different
        $z_2$ for each pair) and stored for later use;
        if Undo action is included in the relevant situation
        then
            store the information in the boolean variable
            Undid = true;
        end
    end
end
for every pair (situation, behaviour) do
    if for at least one $z_1$ of this pair all the $z_2$ are identical to
    $z_1$ then
        the pair is classified as Totally unrecoverable, and $z_1$
        is classified as blocked state;
    end
    if for at least one $z_1$ of this pair all the $z_2$ are different
    from $z_0$ then
        the pair is classified as Non one action reversible;
    end
    if for all the $z_1$ of this pair at least one $z_2$ is identical to $z_0$
    then
        the pair is classified as One action reversible;
    end
    if for all the $z_1$ of this pair at least one $z_2$ is identical to $z_0$
    and Undid==True then
        the pair is classified as Undo;
    end
    while there are no new states left classified as impasse
    state do
        if for at least one $z_1$ of this pair all the $z_2$ are identical
        to a state classified as blocked or impasse then
            the pair is classified as Eventually totally
            unrecoverable;
            $z_1$ is classified as an impasse state;
        end
    end
end
```

Algorithm 1: Reversibility check for ADEPT.

Let us consider again the example described by table 2. First the automatic analysis is performed on a correct design. Afterwards it is performed on three different design error scenarios. In the three error scenarios the reversibility properties of the machine logic is not as expected. That incoherence with the expected behaviour for the correct design may alert the system designer of the presence of design errors. The results of the analysis are intuitively explained for the four scenarios (i.e. correct design and design errors).

Correct design: the "1(ok)" values in columns c1, c2, c3 describe the correct logic design for the machine.
The results of the analysis are:

```
pair number 1 is 1-action reversible for all
  of its input states
pair number 2 is 1-action reversible for all
  of its input states
pair number 3 has no effect
pair number 4 has no effect
pair number 5 is 1-action reversible for all
  of its input states
pair number 6 is 1-action reversible for all
  of its input states
pair number 7 is totally unrecoverable
  for input state: Keyboard(Ok), X(High)
  for input state: Keyboard(Ok), X(Medium)
  for input state: Keyboard(Ok), X(Low)
pair number 8 has no effect
```

Not surprisingly pairs number 1, 2, 5 and 6 are found to be 1-action reversible for all their input states: they describe the action of increasing or decreasing the value of variable X.
Pair number 7 is found to be totally unrecoverable for three input states.
The result of the analysis is coherent with the expected behaviour: putting the keyboard out of order (represented by column c7 that has as input state [Keyboard (Ok), X (any value)]) should be the only totally unrecoverable state change for the user operational domain (if no modelling error has been committed).
Pairs number 3, 4 and 8 are found to have no effect (according to their definitions in Table 2) and are not classified on the reversibility scale as they have no effect to be reversed.

Design error "1(a)" (column c2): action *increase X* in the case of *X/Medium* has no effect. The result of the automatic analysis is the same as in the nominal case, with the exception of pair 6 that is classified as:

```
NON 1-action reversible
```

Indeed pair 6 (embodied by column 6) represents the transition from *X/High* to *X/Medium*. Due to the fact that the transition from *X/Medium* to *X/High* is corrupted, pair 6 has become *non reversible* and therefore *non one action reversible*. As the automatic analysis cannot recognize *non reversible* state changes the column is recognized as "just" *non one action reversible*.

Design error "1(b)" (column c3): the action *increase X* in the case of *X/High* leads to the value *X/Low*. The result of the automatic analysis is the same as in the nominal case, with the exception of pair 3 that is classified as:

```
NON 1-action reversible
```

Indeed two actions *increase X* are needed to restore the input state.

Design error "1(c)" (column c1): the action *increase X* in the case of *X/Low* has no effect. The result of the automatic analysis is the same as in the nominal case, with the exception of pair 5 that becomes:

```
eventually totally unrecoverable
```

for input state [Keyboard (Ok), X (Medium)]. Pair 5 (embodied by column 5) represents the transition from *X/Medium* to *X/Low*. Due to the fact that the transition from *X/Low* to *X/Medium* is corrupted, pair 5 has become *eventually totally unrecoverable*. Indeed the result of the firing of the transition described by pair 5 is that afterwards only the transition described by pair 7 may be fired, leading to a blocking state (i.e. it is no more possible to change X but it is still possible to put the keyboard out of order). Consequently the firing of the transition described by pair 5 leads to an impasse state: therefore pair 5 is precisely "eventually totally unrecoverable".

As far as real systems are concerned an intuitive guess for reversibility may be very difficult. In those cases the automatic analysis may be particularly useful in the frame of a trial and error method to evaluate the impact of a design change on the reversibility of each pair of the logic table. This approach is illustrated in the next section on a simplified autopilot model.

5. ANALYSIS OF THE REVERSIBILITY OF THE GO-AROUND MODE ENGAGEMENT

A "Go-around" is an aborted landing of an aircraft that is on final approach. Many modern aircraft include a "Go-around mode" that automatically sets the throttle to the maximum level. Moreover, depending on the manufacturer, it either switches off the autopilot or it just switches off the instrument landing system mode. Reversing a Go-around decision can be hazardous (e.g. the human operator initiating a late go-around, and successively overriding it and trying to land the aircraft in the short amount of time left [AIR07]). An untimely or involuntary Go-around activation can be hazardous as well (e.g. the Nagoya accident [AAIC94]). Runway overruns, impact with obstructions and major aircraft damage (or post impact fire) often are the consequences of reversing an already initiated rejected landing [AIR]. Moreover autopilot systems are often designed to require many steps to disengage a Go-around mode so that the crew should be fully conscious of the new state of the autopilot.

An autopilot mock-up model, inspired by the autopilots commonly in use on modern commercial aircraft, is presented in this section in order to analyse the reversibility of the Go-around mode engagement. First the automatic analysis is performed on a correct design. Afterwards it is performed on a design error scenario.

In this simplified model a limited number of state transitions are modelled. The state variables are the Autopilot modes (Mode 1, Mode 2, Mode 3, Go-around)[1], Throttle level ("+" for idle, "++" for 40 percent throttle, "+++" for wide-open throttle), Autopilot state (On, Off), Autopilot engagement conditions (satisfied, not satisfied). In this model the human operator (i.e. the crew) may switch the autopilot On or Off, may select Autopilot modes and Throttle level. Switching the Autopilot Off has the effect to reset the Autopilot mode. The human operator may switch the Autopilot On if the engagement conditions are satisfied, otherwise pressing the On button has no effect. Moreover the selection of modes 1, 2 and 3 has no effect if the current mode is Go-around.
For the detailed behaviour of the machine see the logic table 3.

Columns 5, 6 and 7 represent autopilot mode changes. Columns 9 to 14 represent thrust level changes. Columns 2 and 3 represent autopilot state changes. Column 2, unlike column 3, also involves the autopilot mode: when disengaged the autopilot is set on Mode 1. Column 1, corresponding to the engagement of the Go-around mode, changes both the autopilot mode and the throttle level. Note that column 1 represents 12 state transitions at the same time (12 transitions that have as input the cartesian product of 4 autopilot modes and 3 throttle levels).

[1]Typical autopilot modes are Climb, Descend, Climb ignoring altitude constraints, Final descent, Landing etc. For the sake of simplicity in this study only three modes and the Go-around mode are modelled.

Correct design: the "1(ok)" value in column c2 describes the correct logic design for the machine. Note that in those representations the transition labels represent the column of the logic table they embody, not the transition name.

The results of the automated analysis are as follows.
Pairs number 1, 2 and 3 are found to be

```
NON 1-action reversible
```

Pairs number 4, 8, 11 and 14 are found to have (according to their definitions in table 3)

```
no effect
```

Pairs number 5, 6, 7, 9, 10, 12, 13 and 14 are found to be

```
1-action reversible
```

Regarding pair 1 (Go-around mode activation) it is not reversible in a single action because it changes the values of several state variables at the same time, when almost all other pairs act only on one variable at a time. For example in the case of the initial state:
[Autopilot mode (Mode 3), Autopilot state (On), Throttle level (+)],
Go-around mode engagement brings the machine to state:
[Autopilot mode (Go-around), Autopilot status (On), Throttle level (+ + +)]
and the initial state cannot be recovered in less than five actions: Off selection, On selection, Mode 3 selection, Decrease throttle level, Decrease throttle level. This design indeed represents a machine logic for which many steps are needed to disengage a Go-around mode so that the crew should be fully conscious of the new state of the autopilot.

Pairs 2 and 3 non 1-action reversibility is due to the fact that pair 2 changes not only the autopilot state but also the autopilot mode. For example in the case of initial state:
[Autopilot mode (Mode 3), Autopilot state (On), Throttle level (+)],
the "Off selection" brings the machine to state:
[Autopilot mode (Mode 1), Autopilot status (Off), Throttle level (+)]
and the initial state cannot be recovered in less than two actions: "On selection" and "Mode 3 selection". This lack of reversibility is not necessarily desired.

Design error: The machine designer could intuitively imagine a change of the machine logic, for instance simplifying the behaviour of pair 2: nullifying the autopilot mode reset.
More precisely the action *Autopilot engagement: Off selection* would affect only the Autopilot state (from On to Off). The "0(error)" value in column c1 describes this logic design. Nevertheless the results of the automated reversibility analysis is indeed an issue:

Pair number 1 is found to be

```
totally unrecoverable
```

Indeed the selection of the Go-around mode has become irreversible: the only way to disengage it has been lost. The results of the automated analysis warns for undesired side effects in the trial and error process.

		c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14
SITUATIONS		Button Go-around pressed	Button Autopilot Off pressed	Button Autopilot On pressed	Button Autopilot On pressed, engagement criteria not satisfied	Mode 1 selection	Mode 2 selection	Mode 3 selection	Mode Autopilot 1, 2, 3 selection when Go-around mode engaged	Throttle level from + to ++	Throttle level from ++ to +++	Increase throttle level when already at +++	Throttle level from +++ to ++	Throttle level from ++ to +	Decrease throttle level when already at +
Pilot actions															
Actions, button pressed	Go-Around engaged	1													
	Mode 1 selection					1			1						
	Mode 2 selection						1		1						
	Mode 3 selection							1	1						
Actions, Throttle level	Increase throttle level									1	1	1			
	Decrease throttle level												1	1	1
Autopilot engagement	On selection			1	1										
	Off selection		1												
No action															
State															
Autopilot mode	Mode 1					1	1	1							
	Mode 2					1	1	1							
	Mode 3					1	1	1							
	Go-around								1						
Throttle level	+									1					1
	++										1			1	
	+++											1	1		
Autopilot state	On														
	Off														
Autopilot engagement conditions	satisfied			1											
	not satisfied				1										
BEHAVIOUR		Go-Around engaged	Disengage autopilot	Engage autopilot	No effect	Mode 1 engaged	Mode 2 engaged	Mode 3 engaged	No effect	Throttle level from + to ++	Throttle level from ++ to +++	No effect	Throttle level from +++ to ++	Throttle level from ++ to +	No effect
State															
Autopilot mode	Mode 1		1(Ok), 0(error)			1									
	Mode 2						1								
	Mode 3							1							
	Go-around	1													
Throttle level	+													1	1
	++									1			1		
	+++	1									1	1			
Autopilot state	On			1											
	Off		1												

Table 3: Part of the logic table for a simplified autopilot model

6. LIMITATIONS

Definition [Reachability] [Ski09]: In graph theory, reachability is the ability to get from one vertex to another within a graph. We say that a vertex t is reachable from s if there exists a sequence of adjacent vertices (i.e. a path) which starts with s and ends with t.

In the logic table representation of ADEPT, state transitions are represented as pairs (situation, behaviour), and each pair (situation, behaviour) may represent several transitions at once. The set of transitions described in the logic table may be represented as a graph: for the construction of such a graph an exhaustive exploration of pairs (situation, behaviour) is required (for instance by means of an iterative algorithm) to define the list of transitions. For instance column c1 in logic table 3 represents 12 transitions (for the two state variables Autopilot mode/Throttle level), they are:

1. (Mode 1/+) → (Go around/+++)
2. (Mode 2/+) →(Go-around/+++)
3. (Mode 3/+) → (Go-around/+++)
4. (Mode Go-around/+)→(Go-around/+++)
5. (Mode 1/++)→ (Go-around/+++)
6. (Mode 2/++)→ (Go-around/+++)
7. (Mode 3/++)→ (Go-around/+++)
8. (Mode Go-around/++) →(Go-around/+++)
9. (Mode 1/+++)→ (Go-around/+++)
10. (Mode 2/+++)→ (Go-around/+++)
11. (Mode 3/+++)→ (Go-around/+++)
12. (Mode Go-around/+++) →(Go-around/+++)

Therefore some formal analyses of the properties of the graph cannot be performed directly from the logic table representation.
It should also be noted that the logic table exhaustively defines the Cartesian product of all states, even for non-reachable ones starting from the initial state.

For instance the effect of the pilot action *Mode 1 selection* in the non-reachable state:
[Autopilot mode (Go-Around), Autopilot state (On), Throttle level (+)],
is expressed by column (c8).
The reversibility analysis that is developed in this study for ADEPT therefore recognizes irreversibilities regarding states that are actually not reachable.

Moreover the reversibility check presented in this paper cannot handle continuous value variables. To partially address this drawback a pre-discretization of some continuous variables of particular interest could be performed (e.g. flight speed discretized in 4 levels: underspeed, normal speed, near overspeed, overspeed).

7. CONCLUSION

This paper is part of the studies that have been conducted to identify formal properties of the machine logic that are related to bad designs and that could result in HMS performance degradation.
An algorithm to implement a reversibility check on the ADEPT logical tables has been implemented. Four irreversibility properties are detected: the absence of undo, the absence of one action reversibility, the total unrecoverability and the eventual total unrecoverability.
Note that the lack of reversibility may be either the result of a design error, or a conscious design choice, or an intrinsic lack of reversibility or the representation of a failure. The designer, once aware of those vulnerabilities, will possibly decide to modify the machine logic. Thanks to this algorithm they will be able to estimate the impact of those changes.
Further work will focus on the assessment of reversibility properties from a model of the machine represented by Petri nets. Indeed classical characteristics of Petri nets such as transition and place invariants can be investigated in relation to reversibility, and classical algorithms could be used. An algorithm to automatically translate an ADEPT logic table into its equivalent Petri net has already been designed and is currently investigated.

8. REFERENCES

[AAIC94] JAPAN. Aircraft Accident Investigation Commission, Ministry of Transport, *Aircraft accident investigation report. China airlines, Airbus A300b4-622r, B-1816, april 26 1994.*

[AD91] G. D. Abowd and A. J. Dix, *Giving undo attention*, Interacting with Computers **4**, 1991, 317–342.

[AIR] AIRBUS, *Flight operations briefing notes. Landing techniques: Bounce recovery - rejected landing.*

[AIR07] AIRBUS, M. Parisis, *Go-around (presentation)*, 15th performance and operations conference, 2007.

[AW97] R. Amalberti and L. Wioland, *Human error in aviation.*, Aviation safety. (H.M Soekkha, ed.), 1997, p. 91.

[BBS13] M. L. Bolton, E. J. Bass, and R. I. Siminiceanu, *Using formal verification to evaluate human-automation interaction: A review*, Systems, Man, and Cybernetics: Systems, IEEE Transactions on **43** (2013), no. 3, pp. 488–503.

[CCH08] J. Creissac Campos and M. D. Harrison, *Systematic analysis of control panel interfaces using formal tools*, Interactive Systems. Design, Specification, and Verification, Springer, 2008, pp. 72–85.

[Deh02] F. Dehais, *Modelling cognitive conflict in pilot's activity.*, STAIRS 2002: Starting Artificial Intelligence Researchers Symposium, 2002, pp. 45–54.

[DCT11] F. Dehais, M. Causse, and S. Tremblay, *Mitigation of conflicts with automation*, Human Factors: The Journal of the Human Factors and Ergonomics Society **53**, 2011, no. 3, pp. 448–460.

[DCV14] F. Dehais, M. Causse, F. Vachon, N. Régis, E. Menant and S. Tremblay, *Failure to Detect Critical Auditory Alerts in the Cockpit Evidence for Inattentional Deafness.*, Human Factors: The Journal of the Human Factors and Ergonomics Society **56**, no. 4, 2014, pp. 631–644.

[DPS15] F. Dehais, V. Peysakhovich, S. Scannella, J. Fongue, and T. Gateau, *Automation Surprise in Aviation: Real-Time Solutions.*, Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, 2015, pp. 2525–2534.

[DH02] A. Degani and M. Heymann, *Formal verification of human-automation interaction*, Human Factors: The Journal of the Human Factors and Ergonomics Society **44**, 2002, no. 1, pp. 28–43.

[Fea05] M. Feary, *Formal identification of automation surprise vulnerabilities in design*, Ph.D. thesis, Cranfield University, 2005.

[Fea07] M. Feary, *Automatic detection of interaction vulnerabilities in an executable specification*, D. Harris (Ed.):

Engin. Psychol. and Cog. Ergonomics, HCII 2007, LNAI 4562, 2007, pp. 487–496.

[Fea10] M. Feary, *A toolset for supporting iterative human automation interaction in design*, NASA Ames Research Center, Tech. Rep. 20100012861, 2010.

[Jen95] N.R. Jennings, *Controlling cooperative problem solving in industrial multi-agent systems using joint intentions*, Artificial Intelligence **75**, 1995, no. 2, pp. 195–240.

[JMDK00] D. Javaux, M. Masson, and V. De Keyser, *Beware of agents when flying aircraft: Basic principles behind a generic methodology for the evaluation and certification of advanced aviation systems*, Human Factors in Certification. Lawrence Erlbaum Associates, 2000, pp. 375–405.

[Kon99] T. Kontogiannis, *User strategies in recovering from errors in man–machine systems.*, Safety Science, 1999, p. 49.

[LPS97] N.G. Leveson, L.D. Pinnel, S.D. Sandys, S. Koga, and J.D. Reese, *Analyzing software specifications for mode confusion potential*, Workshop on Human Error and System Development, 1997, pp. 132–146.

[LSMC10] C. Layton, P. Smith, and C. Mc Coy, *Design of a cooperative problem-solving system for en-route flight planning: An empirical evaluation*, Human factors: The Journal of the Human Factors and Ergonomics Society **1**, 2010, no. 36, pp. 94–119.

[MC99] M. Modarresa and S. W. Cheon, *Function-centered modeling of engineering systems using the goal tree-success tree technique and functional primitives*, Reliability Engineering and System Safety, vol. 64, 1999, pp. 181–200.

[MDK93] M. Masson and V. De Keyser, *Preventing human errors in skilled activities through a computerized support system*, Advances in human factors ergonomics **19**, 1993, pp. 802–802.

[Mer11] S. Mercier, *Contrôle du partage de l'autorité dans un système d'agents hétérogènes*, Ph.D. thesis, ISAE, 2011.

[NAS] NASA, *asrs.arc.nasa.gov/search/database.html*.

[Pal97] P. Palanque, *Formal methods in human-computer interaction*, Springer-Verlag New York, Inc., 1997.

[PH00] A. R. Pritchett and R. J. Hansman, *Use of testable responses for performance-based measurement of situation awareness*, Situation awareness analysis and measurement, 2000, pp. 189–209.

[PSW00] R. Parasuraman, T.B. Sheridan, and C.D. Wickens, *A model for types and levels of human interaction with automation*, IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans **30**, 2000, no. 3, pp. 286–297.

[PDT11] S. Pizziol, F. Dehais and C. Tessier, *Towards human operator state assessment*, Proceedings of the 1st International Conference on Application and Theory of Automation in Command and Control Systems, 2011, pp. 99–106.

[Piz13] S. Pizziol, *Conflict prediction in human-machine systems*, Ph.D. thesis, Univeristé de Toulouse, 2013, pp. 63-103.

[PTD14] S. Pizziol, C. Tessier, and F. Dehais, *Petri net-based modelling of human–automation conflicts in aviation*, Ergonomics **57**, 2014, no. 3, pp. 319–331.

[Ras83] J. Rasmussen, *Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models*, IEEE Transactions on Systems, Man and Cybernetics **3**, 1983, no. 13, pp. 257–266.

[Rea90] J. Reason, *Human error*, Cambridge University Press (New York, NY), 1990.

[Ski09] S.S. Skiena, *The Algorithm Design Manual*, Cambridge Springer London, 2009.

[SMW07] N. Sarter, R. Mumaw, and C. Wickens, *Pilots' monitoring strategies and performance on automated flight decks: An empirical study combining behavioral and eye-tracking data*, Human Factors: The Journal of the Human Factors and Ergonomics Society, 2007, no. 49, pp. 347–357.

[SW95] N.B. Sarter and D.D. Woods, *How in the world did we ever get into that mode? Mode error and awareness in supervisory control*, Human Factors: The Journal of the Human Factors and Ergonomics Society **37**, 1995, no. 1, p. 5.

[TD12] C. Tessier and F. Dehais, *Authority management and conflict solving in human-machine systems.*, AerospaceLab-journal, no. 4, 2012.

[Wic05] C.D. Wickens, *Attentional tunneling and task management*, 13th International Symposium on Aviation Psychology (Dayton, OH), 2005.

[WS00] D. Woods and N. Sarter, *Learning from automation surprises and going sour accidents.*, Cognitive engineering in the aviation domain. (N. Sarter and R. Amalberti, eds.), 2000, p. 347.