Int j simul model 11 (2012) 2, 89-96
ISSN 1726-4529 Short scientific paper

SEMI-DISCRETE EVENTS AND MODELS IN CATEGORICAL LANGUAGE

Raczynski, S.
Universidad Panamericana, Mexico City, 498 Augusto Rodin, 03920 Mexico City, Mexico
E-Mail: stanracz@stanr.com

Abstract
Some problems related to discrete event simulation and model validation are discussed. The validity of models with simultaneous events is discussed. The basic concepts of the theory of categories are recalled and applied to model construction process. The language of the theory of categories is used to treat some problems in modelling and simulation of discrete event, namely the ambiguity that may appear when simultaneous events occur. It is pointed out that the theory of categories may be useful as a unified language for model description. In particular, the categorical point of view can be useful while treating the model validity. Using this language we can get a high level of abstractions in model building.
(Received in September 2011, accepted in March 2012. This paper was with the author 1 month for 1 revision.)

Key Words: Discrete Events, Modelling and Simulation, Devs, Category Theory

1. INTRODUCTION

Discrete event simulation is a widely used and quiet fast manner to simulate many physical (real existing or proposed) systems. Roughly speaking, a discrete event consists in a change in the state of the model, which may affect one or more model components. For example, if in the real queuing system a client enters and occupies a place in a waiting line, the corresponding discrete event should include an instruction "add one to the queue length" or "insert an object (a client) to the queue". The event occurs in certain time instant t_e, and the change in the model state is immediate. So, the model state, treated as a function of time, is discontinuous. The model state at the time instant t_e is undefined, unless we define it as a left- or right-hand limit of a sequence of the model states at t_e. As in the discrete event model nothing happens between the consecutive events, the model time can jump from one event to another, which makes the simulation very fast.

However, there are some things that should not be forgotten. First of all, in the real world discrete events does not exists, and they should be treated only as an approximation while extracting from the real system its logic only. For example, most packages for electrical circuit simulation support both logical simulation, and "physical" continuous simulation that provides the trajectories with transient processes. If the duration of the transient process in 0-1 logic approaches zero, then the results of this physical version of the model should tend to the results obtained from the logical version. In this case we reject the physical interpretation of the circuit operation. One could say that the discrete event models are a priori invalid, because any change of the state of a physical model is associated with some energy flow. Permitting discrete events, we get energy flows of infinite intensity. But the validity of models refers rather to the final results that are limited by the actual experimental frame. In this sense discrete event model should be validated.

There are some questions that arise when dealing with discrete events. One of these questions is continuity. Given a discrete event that consists in changing the state of a model component from value a to b at time instant t_e, let us define a corresponding continuous

process that performs the same change continuously during a time interval $t_e - e/2$ to $t_e - e/2$, e being a finite time interval. If we apply this procedure to all discrete events in our discrete model M_d, we obtain a continuous model $M(e)$. Let us define:

$$M(0) = \lim_{\varepsilon \to 0} M(e) \tag{1}$$

To do this, we need to define the "lim" symbol. We can use here the convergence in the metrics in the model space, introduced in [11, 12]. This metrics is based on the (Hausdorff) distance between the sets of results generated by different models. So, the discrete model should be $M_d = M(0)$. Unfortunately, this is not always the case.

Generally speaking, a discrete event can be defined as follows:

$$X(t) = \begin{cases} X_1 \text{ for } t < t_e \\ X_2 \text{ for } t > t_e \end{cases} \tag{2}$$

$X(t_e)$ being undefined or defined arbitrarily as X_1 or X_2.

The above can be also written using a Dirac's δ function (supposing no other events occur in the considered time interval):

$$\begin{aligned} &X(t) = X_1 + \int_g^h \delta(t - t_e)(X_2 - X_1)dt \\ &\text{where } g = t_e - e/2, \quad h = t_e + e/2 \end{aligned} \tag{3}$$

The corresponding semi-discrete event can be defined replacing δ by any continuous distribution $f(t)$ that approaches δ, as follows:

$$X(t) = X_1 + \int_g^h f(t - t_e)(X_2 - X_1)dt \tag{4}$$

So, if f approaches δ, then the event becomes a discrete one.

Remember that f is a distribution, so $\int_{-\infty}^{\infty} f(t) = 1$

There are many other possibilities of relaxing discrete events without talking about distributions, like that used in the **example 2** below.

2. THE CONTINUITY

Consider the following **example 1.**

The model M_d has two components A and B. Denote the state of the components as a and b (real numbers), respectively.

Denote : model state x , a real number, define $x = a + b$.

Result space = R (space of real numbers), the result is $y = x$

Let the model includes only two following discrete events.

E1 add one to the state a of the component A

E2 replace the actual value b with a^2

The initial condition: $a = 3$, $b = 0$.

Let the two events execute in time instant t_1 and t_2 respectively. If $t_1 < t_2$ then the final outcome is equal to 20, and if $t_1 > t_2$ then the outcome is equal to 13.

What happens when $t_1 = t_2$?

In this case, we have simultaneous events at t_1. The two events occur simultaneously in model time. In the real (computer) time, they must be executed in some order. If the order is E1, E2 then the outcome is 20, otherwise it is equal to 13. Which result is valid? Recall that the DEVS (Discrete Event Specification) formalism [6], in the case of multi-component

coupled models, provides the formal model component, named *select.* It is a function that embodies the rules employed to choose which of the imminent components (those having the minimum time of next event) is allowed to carry out its next event. However, *select* is not related to any information provided by the modelled system, at least in this case. Consequently, the modeller must define it. As there are only two options for the execution of the events, the two possible results are equal to 20 for $t_1<t_2$ or 13 for $t_1>t_2$. If the modeller has no idea about the selection, then he/she could define the order as random, and take the average value over a set of simulations, which will tend to 16.5. Such outcome, however, never takes place for this discrete model.

Certain specialists explain the select component as follows: "....*conclusions drawn from a simulation study with discrete event systems always flow from a collection of experiments developed to properly deal with the underlying stochastic behaviour. The simultaneous event situation would surely be adequately accommodated in the analysis of the results from this collection of experiments*". Observe, however, that if we can have a "collection of experiments" from the real system, we need no modelling and simulation at all.

Relaxing the discrete events to semi-discrete so that each of them has a finite duration (the changes are continuous over some small interval), we can overcome the need of any "select" rule and get rid of the discontinuity and ambiguity caused by the simultaneous events.

The example 1 may appear to be somewhat artificial, so let us show this procedure applied to the following.

Example 2: 3D rotations.

In the kinetics of mechanisms and in the kinetics of robots, the system dynamics is not taken into account, so the corresponding simulations do not consider the inertia of moving bodies. Consequently, one can treat the movements as discrete events (instant displacements or rotations). This may accelerate simulations, but also implies certain problems. Consider, for example, 3D rotations. Let the model includes three following events:

- RX – rotate around the X axis,
- RY – rotate around the Y axis,
- RZ – rotate around the Z axis.

We will apply the rotations to a line between (0, 0, 0) and (x, y, z). Each rotation event consists in multiplying the corresponding rotation matrix by the (column) vector (x, y, z). Everything works correctly, unless we intend to simulate simultaneous rotation events. If the three events occur in the same model time, they must be executed in some order in the real (computer) time. We have six possible permutations, and we must select one of them (again the *select* function of DEVS). Unfortunately, the final results for each possible permutation are different (indeed, the 3D rotations around distinct spatial axes do not commute). For example, let initially (x, y, z)=(1, 1, 1) and let the rotation angle be equal to 30 degrees, the same for the three rotation events. Here are results, to show only three possible permutations:

- Rotation RX-RY-RZ gives the final position (x, y, z) = (0.444, 0.851, 1.442).
- Rotation RX-RZ-RY gives the final position (x, y, z) = (-0.116, 0.548, 1.639).
- Rotation RZ-RY-RX gives the final position (x, y, z) = (-0.669, 0.090, 1.595).

Other permutations provide results always different from each other. The vector length is constant, equal to square root of 3.

Now, as in the previous example, we "relax" the rotation events, replacing them with semi-discrete events. Let the angle changes in n small steps within a time interval $[t_0,\ t_0 + H]$.

The change in the position (x, y, z) in each step can be expressed as follows:

$$\hat{x}(t+d) = R_x(\frac{\alpha}{n})\hat{x}(t) \tag{5}$$

where $\hat{x} = (x, y, z)$, $\hat{x}_0$ is the initial condition, $d = H / n$, R_x (α / n) is the rotation matrix around X by the angle α / n, and α is the total rotation angle. Eq. (5) can be directly used to code the computer algorithm. So, Eq. (5) is a numerical approximation of a continuous process (a semi-discrete event). Using R_y and R_z instead of R_x we have the rotations around Y and Z axes. Being continuous, the three rotations can be executed simultaneously without any need to select a particular rotation order. The result, with the same initial condition and $\alpha = 30$ degrees is $(x, y, z) = (-0.999, -0.520, 1.315)$, quite different from any permutation of discrete-event rotations. The results are the same, independently on the order of small rotations in each rotation step, up to the small differences due to the time discretization (5). These errors tend to zero with $n \to \infty$, for any value of $H > 0$. This means that we can permit that H approaches zero, but the discrete event model with simultaneous events is not the limit of a sequence of semi-discrete models. In other words, there is a discontinuity in $H = 0$.

The concept of semi-discrete events has been proposed earlier [10, 11], in some different way. The main idea is to create a model where the events have finite duration e, and which results converge to the result of the discrete event model whenever the discrete model is not ambiguous (no simultaneous events). Semi-discrete events are nothing more than continuous changes of the system state. This term suggests that the event duration e is small in relation to the simulated time period, and that it is possible to make e arbitrarily small (but always greater than zero).

3. USING THE CATEGORICAL LANGUAGE

In 1942-45, Samuel Eilenberg [2] and Saunders Mac Lane introduced *categories, functors*, and *natural transformations* as part of their work in topology, especially algebraic topology (see also [1, 5]). As the concepts of the theory of categories provide a high level of abstracting, the language of the theory may be useful while constructing models and examine their properties. In [8] the reader can find interesting remarks on application of the Theory of Categories to cognitive modelling. However, the main and most widely spread field of applications is the mathematics itself (consult, for example, [1, 7]).

Recall the definition of the *category*. The reader familiar with this terminology (recalled also in [13]) may skip this part.

A category is defined by its data and the axioms the data must satisfy. The data are the following:

1. A collection of things called objects. By default, A, B, C, ... vary over objects.
2. A collection of things called morphisms, sometimes called arrows.

By default, f, g, h, ... vary over morphisms.

3. A relation on morphisms and pairs of objects, called typing of the morphisms.

By default, the relation is denoted $f : A \to B$, for morphism f and objects A, B . In this case we also say that $A \to B$ is the type of f, and that f is a morphism from A to B. Object A is called the source of f, and the object b is its target. In other words, $src\, f = A$ and $tgt\, f = B$.

4. A binary partial operation on morphisms, called composition.

By default, $f ; g$ is the notation of the composition of morphisms f and g.

5. For each object A, a distinguished morphism, called identity on A. By default, idA, or id when A is clear from the context, denotes the identity on object A.

The morphisms of a category must satisfy the following axioms.

A3.1 $f: A \to B$ and $f: A_0 \to B_0$ => $A = A_0$ and $B = B_0$ (unique-Type).

A3.2 $f: A \to B$ and $g: B \to C$ => $f ; g : A \to C$ (composition-Type).

A3.3 $idA: A \rightarrow A$ (identity- type).

A3.4 $(f ; g) ; h = f ; (g ; h)$ (composition).

A3.5 $id ; f = f = f ; id$ (identity).

Note that $f : A \rightarrow A$ does not mean that f is an identity (it is an *endomorphism*). For example, if X is the space of real numbers, the function $y(x)=x^3$ maps X to itself, but it is not an identity. Also $f : A \rightarrow B$ and $g : A \rightarrow B$ does not mean that $f = g$.

Example 3 (following Maarten M. Fokkinga [4])

Each pre-ordered set $(A, \leq)$ can be considered a category, in the following way. The elements a, b, ... of A are the objects of the category and there is a morphism from a to b precisely when $a \leq b$. Formally, the category is defined as follows:

An object is: an element in A.

A morphism is: a pair (a, b) with $a \leq b$ in A.

$(a, b): c \rightarrow d \equiv a = c \ \Lambda \ b = d$.

$(a, b) ; (b, c) = (a, c)$.

$id_a = (a, a)$.

Another example can be a mechanical system, composed by a set of bodies. The corresponding category $\mathbf{M_0}$ can be defined as follows:

An object of $\mathbf{M_0}$ is the pair $(X(t), t)$ where X is the system state and t stands for time.

A morphism is a pair $((X(t_1), t_1), (X(t_2), t_2))$ with $t_1 \leq t_2$.

$((X(t_1), t_1), (X(t_2), t_2)): (Y, r) \rightarrow (Z, s) \equiv (Y, r) = (X(t_1), t_1) \ \Lambda \ (Z, s) = (X(t_2), t_2)$.

$id = ((X(t), t), X(t), t))$

If $\mathbf{M_0}$ is a real physical system, then the morphisms are physical rules of movement (state transitions) from a given time instant to some other future time instant.

Now, recall the concept of the *functor*.

The functor is a mapping from one category to another, preserving the categorical structure. It preserves the property of being an object, a morphism, typing, composition and identity. To be more precise, the definition is as follows (as given in [4]).

Let A and B be categories; then a functo*r* from A to B is: a mapping $\boldsymbol{F}$ that sends objects of A to objects of B, and morphisms of A to morphisms of B in such a way that:

A3.6 $\boldsymbol{F} f : \boldsymbol{F}A \rightarrow_B \boldsymbol{F}B$ whenever $f : A \rightarrow_A B$.

A3.7 $\boldsymbol{F} id_A = id_{\boldsymbol{F}A}$ for each object A in A.

A3.8 $\boldsymbol{F}(f ; g) = \boldsymbol{F} f ; \boldsymbol{F} g$.

The symbol $\rightarrow_a$ denotes a mapping in the category A . The axioms A3.7 and A3.8 imply the following: $\boldsymbol{F}(f ; \ldots ; g) = \boldsymbol{F} f ; \ldots ; \boldsymbol{F} g$.

Now, consider again the system $\mathbf{M_0}$**.** What we can do is merely to observe its behaviour. This is rather a philosophical assertion telling us that no object or system of the real world can be completely understood and its functioning completely known. So, let us assume that $\mathbf{M_0}$ is a category. If so, we can look for another category **M** in such a way that:

A3.9 The properties of **M** are simple enough to be understood.

A3.10 There exists a functor $\boldsymbol{F} : \mathbf{M_0} \rightarrow \mathbf{M}$.

We will call the category **M** the *model of* $\mathbf{M_0}$, and the functor $\boldsymbol{F}$ the *operation of modelling*. The model **M** does not have to be unique, so we can look for other, simpler or more sophisticated models for the same physical (or proposed) system. The definition of the

validity implies that any valid model must form a category, and the operation of modelling must satisfy the above functor axioms.

For example, define the objects of $\mathbf{M_0}$ as pairs $(X_0(t), t)$, where X_0 is the state of a set of small bodies (3D positions, velocities, body temperature, spatial orientation, spin etc.) moving in space and subject to gravitational forces in such the way that they do not collide. Then, the model **M** may be a category of material points, the objects of **M** being pairs $(X(t), t)$, and the morphisms $f_{a,b}$ of **M** being defined by solutions to a system of differential equations (Newtonian or relativistic) that describe the state transition over a model time interval $[a, b]$. The functor $\boldsymbol{F}$ maps the real world objects $(X_0(t), t)$ into $(X(t), t)$ taking the positions and velocities of the mass centre of each body only. The morphisms of **M** are defined as $f_{1,2}$: $(X(t_1), t_1) \rightarrow (X(t_2), t_2)$ where $f_{1,2}$ means to solve the model differential equations over $[t_1, t_2]$ with the initial condition $X(t_1)$. In the sequel we will omit the sub indices of f. Of course, if the system $\mathbf{M_0}$ is subject to some external excitation (forces), then this (input) functions should be added to the definition of the morphism.

Note that a consequence of the fact that $\boldsymbol{F}$ is a functor (see A3.6) is that the graph of Fig. 1 must commute (passing from $X_0(t_1)$ to $X(t_2)$). If it does not, there is something wrong with the definition of the functor. This is a well known condition of model validity. Bernard Zeigler [14] defines the validity in the same way, avoiding the categorical language, perhaps in order to make the definition simple and not to bother the reader with the concepts of the category theory.

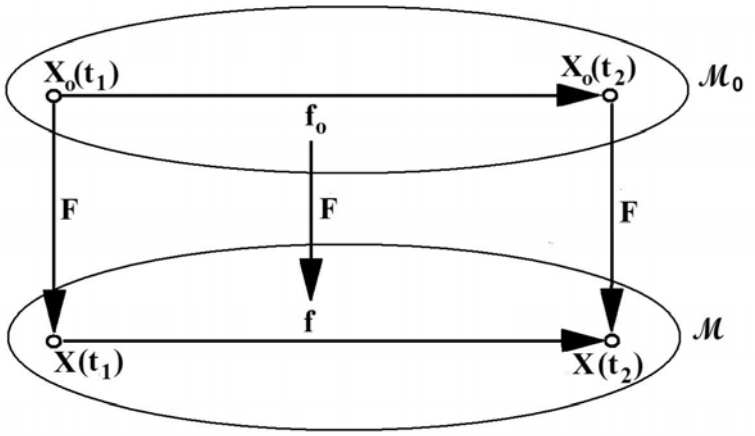

Figure 1: Model validity.

Note that in Fig. 1 we have $(X(t), t) = \boldsymbol{F}\ (X_0(t), t)$, where the sub 0 means the original (real) system. Remember that the functor $\boldsymbol{F}$ maps category objects as well as morphisms.

Now suppose that in the real system the bodies can collide (due to their finite diameter or to the movement itself). In such the case, the graph 1 does not commute, simply because the real transition rule f_0 takes into account possible collisions and the morphism f does not (the model contemplates a movement of material points only, not the real bodies). In this case the model is invalid.

The examples of models of Section 2 can be also expressed in the language of category theory. For the basic model (the real world), consider the pairs (*state,time*) as category objects, and morphisms f as the rules of state-time transition. While constructing a discrete event model, the functor maps (*state,time*) pairs into (*model state, model time*) pairs, and the real system morphism f into the rules of transitions in a discrete event case, or some differential equation or other rule for the continuous case. As the discrete model provides false results for simultaneous events, the corresponding graph of Fig. 1 does not commute.

This is a categorical interpretation only, in fact to see that these models are invalid we do not need the category theory.

Now, take a closer look at simultaneous events. Let category **M** be a set of pairs (X_n, t_n), where X is the model state and t_n a time instant. We introduce a weak order in **M** with respect to time, namely we define $(X_i, t_i) \leq (X_j, t_j)$ if $t_i \leq t_j$. Let for any two objects A and B in **M,** such $A \leq B$ there exists a morphism between A and B. We will interpret morphisms as discrete events. More precisely, an event morphism between A and B can be expressed as follows.

$f: A \rightarrow B$ ≡ " take the value of X, perform an operation over it and substitute the result of the value of X in B, take the value of time for *src f* add a non-negative (may be zero) value to it and substitute it as the time of *tgt f*. We will say the event occurs at time equal to the time of *tgt f* ".

Consequently, if there are two or more objects with the same value of t, then morphisms must exist between each other. This is the case of simultaneous events. Fig. 2A shows a diagram for three events that occur in time instants $t_n < t_n + e < t_{n+1}$. If e approaches zero, the events f and h have nearly the same target time. One may suppose, that the model with simultaneous events f and h is the limit case for $e \rightarrow 0$. However, this is not the case. Fig. 2B shows the diagram for $e = 0$. As the targets of morhisms f and g have the same value of t, there exist morphisms between the corresponding nodes, namely h and i. The morphism h ; i is not necessarily an identity. As for the morphisms j and k we may assume that these are identities. Observe the following.

* The model **changes its structure** at when e reaches 0.

** We have f; h ; $k = g$; i ; j which means that f; $h = g$; i.

Now, return to the example 1 of section 2. Let both f and i correspond to the event EV1, , g and h to the event EV2 (add one to X and take the square of X, respectively).

Obviously, f; h cannot be equal to g ; i. So, we have an ambiguity at t_{n+1}.

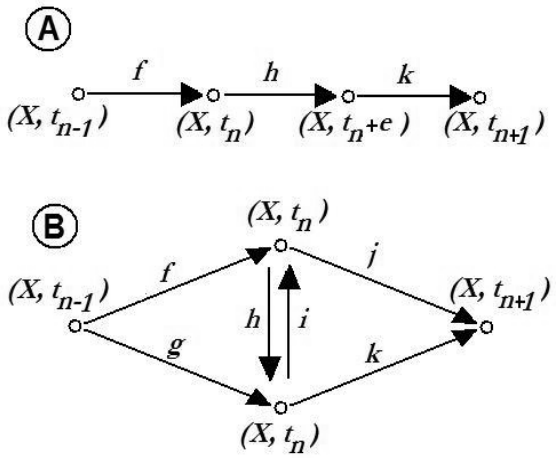

Figure 2: Two versions of the model. The suffix of X is omitted, supposed to be the same as that of t. Identity morphisms are assumed, but not shown.

If we take the model 2B as a (new) model of 2A with $e = 0$ (simultaneous events), then we obtain an invalid model, because the diagram 2B does not commute.

4. CONCLUSIONS

Section 3 shows how to express models and their properties in the categorical language. The categorical language may be useful when we want to have a higher level of abstraction. Each

morphism can be a function, an algorithm, a piece of software or even a textual specification of state transitions. In the case of the discrete events, we show again (see also [9, 10]) that they form a singularity in the space of models. This may result in certain problems in model validity. A more general remark might be formulated regarding the relation between the real world (system) and the model we create. Is the real world a category? If so, any valid model should also be a category.

At the fist glance, the reader might be somewhat confused and lost in the definitions of the theory of categories. In fact, the theory was created by mathematicians, and most of the books and texts about the theory are full of examples taken from advanced mathematics. Note, however, that the definitions of the basic concepts are not very complicated and need not be applied to structures in advanced mathematics. If the knowledge on the category theory were a little bit more propagated in the modelling and simulation community, it might become a good and uniform way to express model specifications.

REFERENCES

[1] Asperti, A.; Longo, G. (1991). *Categories, Types and Structures*, MIT Press, ISBN 0262011255, *ftp://ftp.di.ens.fr/pub/users/longo/CategTypesStructures/book.pdf*

[2] Eilenberg, S. (1974). *Automata, Languages, and Machines* (2 vols.), Academic Press, New York, ISBN 0-12-234001-9

[3] Fokkinga, M. M. (1992). *Law and Order in Algorithmics,* PhD thesis, University of Twente, Department of Computer Science, Enschede, The Netherlands

[4] Fokkinga, M. M. (1994). A Gentle Introduction to Category Theory, from *http://wwwhome.cs.utwente.nl/~fokkinga/mmf92b.pdf,* accessed on 12-01-2012

[5] Hoare, C. A. R. (1989). Notes on an Approach to Category Theory for Computer Scientists, Broy, M. (Ed.), *Constructive Methods in Computing Science*, 245-305, International Summer School, Springer Verlag, NATO Advanced Science Institute Series (Series F: Computer and System Sciences, Vol. 55)

[6] Hu, X.; Zeigler, B. P.; Hwang, M. H.; Mak, E. (2007). DEVS Systems-Theory Framework for Reusable Testing of I/O Behaviors in Service Oriented Architectures, *IEEE International Conference on Information Reuse and Integration*, Las Vegas, NV, 6 pages, doi:10.1109/IRI.2007.4296652

[7] Lawvere, F. W.; Rosebrugh, R. (2003). *Sets for Mathematics*, Cambridge University Press, Cambridge, UK, doi:10.1017/CBO9780511755460

[8] Gómez, J.; Sanz, R. (2009). *Modeling cognitive systems with Category Theory: Towards rigor in cognitive sciences*, Universidad Politécnica de Madrid*,* ASLab A-2009-014 v 1.0 Final, from: *http://tierra.aslab.upm.es/documents/controlled/ASLAB-A-2009-014.pdf*

[9] Raczynski, S. (2003). Are discrete models valid?, *VIth Conference on Computer Simulation and Industry Applications,* McLeod Institute for Simulation Sciences, Tijuana, B. C., Mexico

[10] Raczynski, S. (2000). Alternative mathematical tools for modeling and simulation: Metric space of models, Uncertainty, Differential Inclusions and Semi-discrete Events, *Proceeedings, European Simulation Symposium ESS 2000*, Hamburg, Germany (Keynote plenary speach)

[11] Raczynski, S. (2006). *Modeling and Simulation: The Computer Science of Illusion*, Wiley, Chichester

[12] Raczynski, S. (1998). On the metric structure in the space of dynamic system models, *Transactions of the Society for Computer Simulation International*, Vol. 15, No. 2, 70-75

[13] Raczynski, S. (2011). Uncertainty, dualism and inverse reachable sets, *International Journal of Simulation Modelling*, Vol. 10, No. 1, 38-45, doi:10.2507/IJSIMM10(1)4.180

[14] Zeigler, B. P. (1976). *Theory of Modelling and Simulation*, Wiley, New York