"Smart Entity" – How to Build DEVS Models from Large Amount of Data and Small Amount of Knowledge?

Thierry Antoine-Santoni, Bastien Poggi, Evelyne Vittori, Ho Van Hieux(✉), Marielle Delhom, and Antoine Aiello

University of Corsica, UMR CNRS SPE, Corte, France
{antoine-santoni_t, ho_vh}@univ-corse.fr

Abstract. University of Corsica and CNRS are working on a scientific program called "Smart Paesi". This project focus on a sustainable rural territories development using advanced artificial intelligence concepts in order to adapt smart city concept (including sustainable development, ICT with by example wireless sensors network, education, e-citizenship, governance) to rural territories and their specificities. In this paper, we introduce a new approach combining discrete event modelling concepts and machine learning methods. This work is a first step towards the conception of a generic and scalable framework allowing model generation from large amount of data.

Keywords: Machine learning · DEVS formalism · Model generation · Decision supports · Internet of Thing · Big data

1 Introduction

In the recent past, boosted by artificial intelligence (AI) progress, a horde of automation and acceleration technologies in the IT field have been proposed to resolve smart cities problematics with great success [1, 2]. Indeed fewer have been proposed for people who live in rural and isolated territories.

In the same time, we assist to the deployment of low cost Wireless Sensor Network (WSN) over all territories. Due to the success of IoT [3], related technologies have generated an exponential increase of collected data. The challenge today is to explore the potential of this amount of data over artificial intelligence process base and machine learning (ML) in a modeling and simulation context as describe in Fig. 1.

Supported by a "European Fund for Research and Development" (ERDF) and the "Regional Council of Corsica" (RCC), the "Smart Village Scientific Program" (SVSP) [4] proposes newer approaches based on the concept of "real-world people-centric applications" [5]. Built in partnership with two company: EDF (Electricity of France) and SITEC (IT Service Company) the project focuses on four areas of research: environmental data gathering, data visualization, e-democracy and simulation. This paper deals with the last.

H. Song et al. (Eds.): SIMUtools 2019, LNICST 295, pp. 615–626, 2019.
https://doi.org/10.1007/978-3-030-32216-8_60

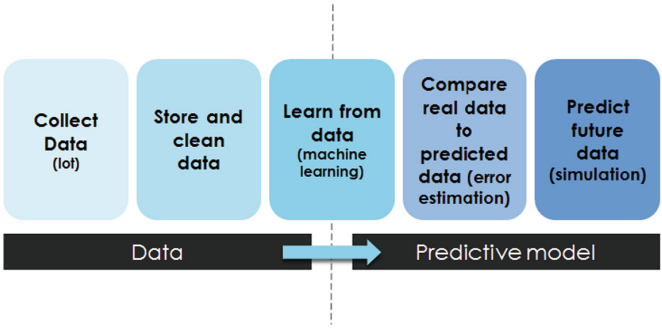

Fig. 1. From data to predictive model

Our goal is to provide a robust framework allowing an easy integration of collected data inside a modelling and simulation process by replacing "physical model" with "data model".

2 M&S, Data and Machine Learning

Recently, researches in the field of Modeling and Simulation (M&S) have intensively evolved towards hybrid approaches combining M&S background and Artificial Intelligence advances in data mining and machine learning. Nowadays, complementarity of both approaches seems to appear as an evidence. Miller and Buckley [6] argue that they should be used in conjunction through a "modeling continuum" and illustrate their vision upon examples from health care and supply chain management. In [7], the authors provide an extensive comparison of both approaches called respectively "simulation modeling" and "data modeling" and detail their advantages and limitations. They demonstrate their complementarity and suggest a new modeling approach involving both of them. In according to Andreas Tolk [8], the next generation of Modeling & Simulation applications will integrate big data and deep learning tools and methods: "bringing all the three topics together will create synergy that will allow us to significantly improve our services to others science".

An interesting review about ways of combining both approaches in the context of manufacturing and logistics is done in [9]. They focus on the integration of Machine Learning (ML) process from a simulation perspective.

According to literature in the field of M&S, the main benefit of the use of Machine Learning is to improve the efficiency simulation analysis. In other words, it may help to reduce simulation cost. This is particularly useful in the context of complex models requiring extensive resource allocation and leading to very expensive experiments. Recently [10], it has been proposed to use machine learning mechanisms inside a DEVS simulator in order to optimize simulation execution by learning from past simulations.

Wang and Marek-Sadowska [11] suggest a double level learning flow applied to the field of circuit design. ML is first used to reduce input samples by discarding

unimportant samples. The selected inputs are then predicted by ML rather than simulated and so simulation cost is reduced.

In the case study of a green-house control system introduced in [4], the global model is a "simulation oriented" one but it internally uses results given by a "data oriented model" ("the controller"). Furthermore, data obtained as outputs from the simulation model enhance the dataset used for the "controller" data oriented model thru another data model called "Optimal Controller Model". Experimental results show that the use of such hybrid model improves significantly control performance and reduce the rate of error.

These hybrid modeling approaches may also be related to "grey-box" modelling in the field of "system (or model) identification" [12]. Grey-box models are defined as combination of "data driven models" (black-box), and "physical based models" (white-box). They combine physics based methods for building the model structure and use data driven to estimate the model parameters. They also benefit from the advantages of both approaches: generalization capabilities from physical models and better accuracy from data driven ones.

On the other hand, M&S models may be used to help in building ML models. Results from simulation may provide data sets allowing to construct data-oriented models. In the field of personalized medicine [13], it shown that the prediction of a cancer treatment efficiency cannot be processed using "pure data" approaches. Authors suggest to integrate simulation as a "pre-processing step in a machine learning pipeline to include detailed expert knowledge".

Similar hybrid approaches are introduced in the domain of "smart manufacturing" [14]. In a case study, simulation results are used to generate data streams that can be used by a diagnostic analytics application ("data oriented model").

In summary, ML may assist M&S in building input samples, estimating unknown input parameters, analyzing output data and validating simulation results. M&S can also assist ML by providing data sets as output results from simulation process.

In the context of smart village, according to the diversity of data and associated processes, it clearly appears that we need to combine both data centered and simulation based approaches. However, we think that defining an integration framework will be a guaranty of the global model coherence.

In order to provide a high level of genericity and interoperability between models and their formalism we choose to build our approach on Discrete Event System Specification (DEVS) formalism. We introduce the concept of "Smart-Entity" (SE) as a specific DEVS model. Background on the formalism is described in next part.

3 Back Ground: DEVS Formalism

DEVS is a modelling and simulation formalism proposed by Zeigler [15]. Due to its success since its publication and a large community of users, high number of extension DEVS extensions have been proposed to enrich the classic formalism: dynamic DEVS [16], parallel DEVS [17], Cell-DEVS [18], etc. Due to the number of extensions DEVS is today one of the main used formalism for modeling and simulation in research teams.

This formalism can be considered as a multi-formalism [19] integrating other formalisms such as Petri-Net [20] or differential equations [21].

DEVS allows to represent a wide range of systems. It has been used with success for many applications in various fields such as: agriculture, military, anthropology, engineering, ecology, etc.

The main idea beside this formalism is an explicit separation between modeling description and simulation core. The formalism is based on two mains concepts: "Atomic model" (AM) and "Coupled model" (CM). AM describes the system behavior in a modular way and CM describes the system structure by abstraction levels and model encapsulations. This genericity of model description provides a reusable abstract simulator independent of studied systems.

3.1 Atomic DEVS Model

Atomic DEVS model (AM) is the lowest level of abstraction of studied system. It describes the component behavior. This model is defined by the following structure:

$$AM = < XY, S, \delta ext, \delta int, \lambda, ta > \quad (1)$$

Where:

- X: the set of input ports of the model defined by tuple (port, value)
- Y: the set of output ports of the model defined by tuple (port, value)
- S: the set of model states
- δext: the external transition function activated when events are received on model inputs port
- δint: the internal transition function activated during state change (state time exceed)
- λ: the ouput function activated when outputs are produced by model
- ta: time advance function defining state duration for each model state

Complex systems are described over several atomic models. Inputs and outputs (IO) of model must be connected to others models IO. This part is insured by coupled DEVS model (CM).

3.2 Coupled DEVS Model

CM describe model structure over interconnections and encapsulations. Indeed CM can encapsulate AM and CM models allowing different granularity of system description. This models are described by the following tuple:

CM = <X, Y, D, EIC, IC, EOC, Select>

Where:

- X: the set of input ports of the model defined by tuple (port, value)
- Y: the set of output ports of the model defined by tuple (port, value)
- D: the set of components (AM or CM)
- EIC: External Input Coupling (input to input)

– IC: Internal Coupling (input to output)
– EOC: External Ouput Coupling (output to output)
– Select: selection function used to ordinate model execution when their states expire at the same time.

After this description of DEVS formalism we introduce our approach of Smart Entity".

4 Our Approach "Smart Entity"

We choose to build our approach on DEVS formalism in order to maintain a high level of genericity in the decision support framework.

Our framework will allow to define three kinds of DEVS models:

– "white-box" model: atomic or coupled classical DEVS models
– "black-box" model: DEVS atomic wrapper model that encapsulates ML capabilities ("Smart entity Model").
– "grey-box" model: coupled models including at least one Smart entity.

In this part, we focus on "black-box" model by introducing the concept of Smart-Entity Model. We detail structural and behavioral conceptual representations of the Smart Entity Model and we show how these capabilities are embedded into a DEVS atomic model.

4.1 SEM Conceptual Structure

The "smart entity model" (SEM) is a generic model based on data approach modeling concepts. As describe on Fig. 2, the model is defined with a fixed number of inputs and outputs ports. This constraint maintains a high level of interoperability between different model on smart entities to represent a global system.

In our approach, two inputs ports are defined: "Environmental Data" (XE) and "Decision data" (XD). XD port can be optional if SE represents an entity on witch no influences can be made. The XE port is connected to outputs of models that represent the unmanageable phenomena of studied system as weather or environmental events (e.g. fire, storm, etc.). The XD port is connected to models that represent the manageable interactions such as decision or users choices.

Two outputs ports are also defined. The outputs ports are: "Interactions" (YI) and "Results" (YR). The YI port is connected to other SEM models in case of interactions between them. The YR port is connected to a "decision model" (DM) in order to estimate efficiency of different decisions scenarios by collecting and observing decisions effects on results. Their outputs values are computed by ML methods provided inside the SEM attributes.

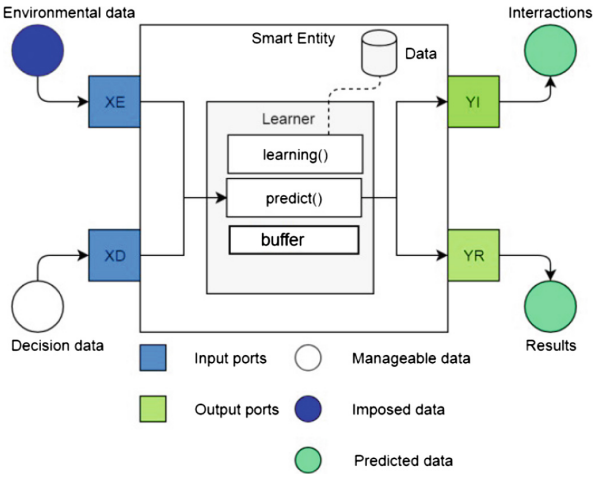

Fig. 2. Conceptual structure of SEM

As illustrated in Fig. 2, the SEM also contains a buffer. The goal is to increase the prediction precision by considering previous received data without compromising simulation coherence. Indeed many applications need more than one previous value to predict next value (e.g. past three day rain to predict day rain). In this case SE stores the previous values in a buffer. When prediction function is called, this buffer is passed as an input parameter. At each new event on input port, this buffer is updated with the new value received.

4.2 SEM Dynamic Behavior

SEM execution can be decomposed in three parts: "learning", "testing" and "simuling". Transitions between these states are explain on Fig. 3.

At the beginning of process execution, SEM is in the "learning" state. During this state, the model is built from a dataset by using machine learning algorithms. When this step is over, SEM state turns into "testing". Model outputs are collected and analyzed. If their accuracy exceeds a specified threshold SEM state becomes "simuling". Otherwise it goes back to "learning" state and a new machine learning process is performed. At the end of simulations, if some new data has been added to the initial dataset, the size of the added data is quantified. If the increase is significant, a new learning process occurs.

Using of ML methods is based on the principle of "cross-validation". Usually the algorithms use 80% of data to make their learning process and 20% of data to make the validation process. This principle provides an estimation of prediction quality. We reproduce this concept in our architecture and we add the concept of "choosing the best method" for SEM linked dataset.

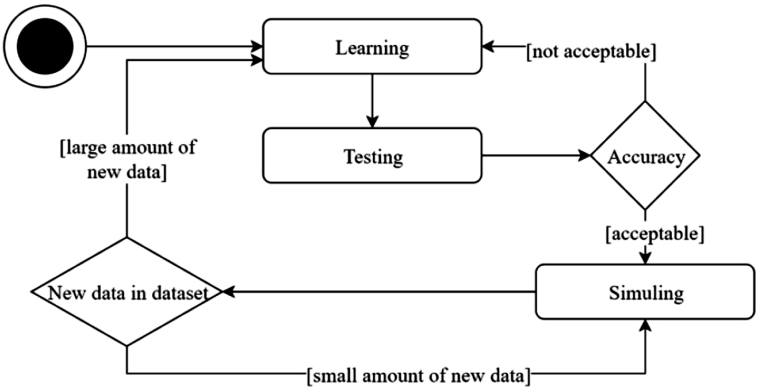

Fig. 3. Conceptual representation of SEM states

During "learning" step, the SEM generates several learners based on different machine learning algorithms family. The "learning ()" method is called with database link as parameter. The different algorithms provided by SEM are listed in Table 1.

Table 1. Algorithm provided by smart entity

Category	Method name
Linear Regressor	Linear Regression (LR) Huber Regression (HR) Random Sample Concesus Regression (RSCR)
Generalized linear model	Stochastic Gradient Descent (SGD)
Support vector Regression	SVR with kernel
Gaussian Processes Regrettion	Gaussian Process Regression (GPR)
Ensemble methods	Kernel Ridge Regression (KRR) Bayesian Ridge Regression (BRR)
Decision Tree	Decision Tree Regressor Random Forest Regressor Extra Trees Regressor Gradient Bosting for regression
Neural Networks	Neural Network with 2 hidden layers (NN2) Convolutional Neural Network (CNN) Recurrent Neural Network (RNN) Bi-directional Recurrent Neural Network (LSTM)
Others	Least Angle Regression (LAR) Automatic Relevance Determination Regression (ARDR)

During "testing" step, "loss value" (LV) is computed for each instanced learner. Different equations are introduced in literature. In our models, LV is represented by men squared error (MSE) and given by:

$$MSE = \frac{1}{N} \sum_{(x,y) \in D} (y - prediction(x))^2 \quad (2)$$

where

- N is the dataset size
- x is the input of the prediction function
- y is the observed value

The learner with minimum LV is selected to produce model behavior. During simulation when data are received on inputs port, they are combined with buffer values to make predictions. At regular interval the SEM checks the size of its learning database. If number of new records exceed a specified threshold, a new "learning" step occurs. It allows to perform better predictions over simulation time.

4.3 SEM as a DEVS Atomic Model

As said before, SEM outputs are generated by a learner object encapsulated inside the SEM. To make an efficient learning, this object needs a large amount a data, pretreated and stored inside a database. Each record is described by

$$< \text{XE, XD,YI,YR} > \quad (3)$$

Where:

- XE: {XE1,…,XEn}: the inputs environmental data
- XD: {XD1,…,XDn}: the decision data
- YI = {YI1, …, YIn}: the results (interactions)
- YR = {YR1, …, YRn}: the results

This record is enriched by model state variables C added to each record to enhance the learning process with model characteristics. At this end of configuration process, the SEM is ready to learn from the following dataset:

$$< XE, XD, YI, YR, C > \quad (4)$$

This conception allows us to define a SEM as a DEVS atomic model wrapper encapsulating ML capabilities.

The input sets XE and XD are linked to X values of DEVS atomic model (AM). The output sets YI and YR are linked to Y values of DEVS atomic model. C values are linked to DEVS states (S).

Concerning the DEVS functions, they are the same for each SEM independently of modelized system. The δext function stores input values and updates the buffer (function save). The λ function calls the "predict" function of learner and sends value

on specified output port. The δint function checks the SEM database and starts a new learning process when needed.

The mapping between DEVS concepts and ML concepts is summarized in Fig. 4.

DEVS	init	X		Y		S	δext()	δint()	λ()	ta()
SEM	learning()	XE	XD	YI	YR	C	save()	check_data()	predict()	ta()

Fig. 4. Analogy between DEVS & SEM

5 First Results

In order to validate the SE concepts, we build our own machine learning library called "PredictSV". This library is based on several well know Python optimized and scalable libraries: Kereas, Scikit-Learn, Pytorch and Tensorflow. This "PredictSV" library not only merges several ML libraries but also tries to automatize the data pretreatment (e.g. normalization) and the method configuration process in a not fastidious way for the modeler. These aspects are not presented in this paper.

Before applying the SEM on collected data from SVSP, we choose to use robust datasets to compare obtained results with literature. In next step of our development, these datasets will be replaced by SVSP data that we are collecting today.

We choose a dataset relevant to sustainable development as studied in (SVSP): "Weather in Madrid" (WIM). This data has been linked to a SEM integrated in a DEVS architecture as described in Fig. 5.

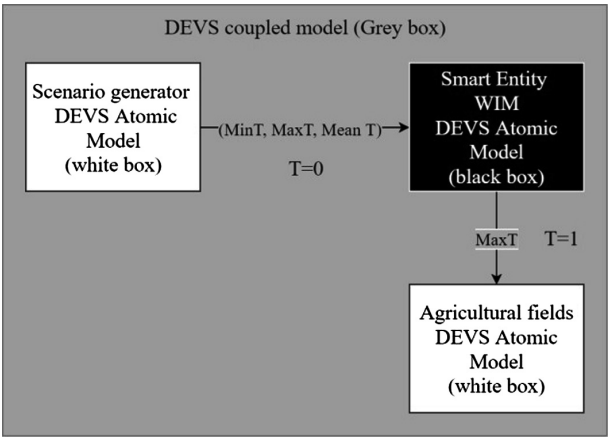

Fig. 5. Example of SM utilization for ECE dataset

Smart Entity wrapping has been tested in different ways considering different number of features. For WIM dataset, we try to predict day+1 temperature from

minimum, maximum, mean temperature of 8 previous day. An example of best obtained results with double layers neural network is visible on Fig. 6.

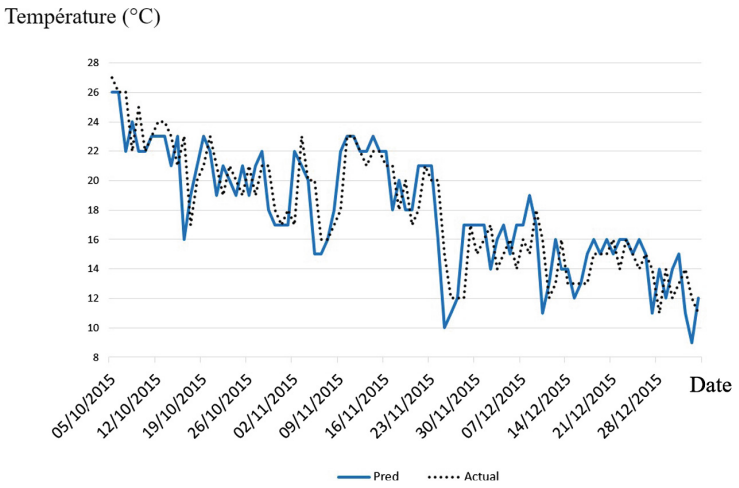

Fig. 6. Comparison of SM predict data and real data

First results confirm the interest of our unified approach to approximate physical model behavior.

6 Conclusion

Smart Village is a scientific program which combines the collected data using Wireless Sensor Networks, data storage, visualization and prediction.

For the prediction, our goal is to propose a generic model using DEVS formalism whose inputs are produced by Machine Learning methods using large amount of data. This model had to be compatible with classic DEVS models in order to benefit of available models. The "smart entity model" offers a promising solution to this problematic for system on which user dispose of big data but few knowledges.

First results have replaced a model of weather with an acceptable precision level during a DEVS simulation. At this step of our research, we need to perform other tests on different systems and test interactions between different smart entities. Moreover many improvement must be proposed. Indeed, data pretreatment is a central task in machine learning field. We need to propose efficient software solution helping modeler to exploit efficiently provided data. We also need to develop an efficient ML method comparison process in order to provide to the SEM the better predict core.

An important part of our work will be to integrate SEM inside combinational optimization process to provide an intuitive but powerful tool helping rural decision makers to benefit of most recent advances in M&S, ML and optimization fields.

References

1. Silva, B.N., Khan, M., Han, K.: Towards sustainable smart cities: a review of trends, architectures, components, and open challenges in smart cities. Sustain. Cities Soc. **38**, 697–713 (2018)
2. Kitchin, R.: The real-time city? Big data and smart urbanism. GeoJournal **79**(1), 1–14 (2014)
3. Stankovic, J.A.: Research directions for the internet of things. IEEE Internet Things J. **1**(1), 3–9 (2014)
4. https://smartvillage.universita.corsica/
5. Campbell, A.T., et al.: The rise of people-centric sensing. IEEE Internet Comput. **12**(4), 12–21 (2008)
6. Miller, J.A., Cotterell, M.E., Buckley, S.J.: Supporting a modeling continuum in scalation: from predictive analytics to simulation modeling. In: Proceedings of the 2013 Winter Simulation Conference: Simulation: Making Decisions in a Complex World, pp. 1191–1202 (2013)
7. Kim, B.S., Kang, B.G., Choi, S.H., Kim, T.G.: Data modeling versus simulation modeling in the big data era: case study of a greenhouse control system. Simulation **93**(7), 579–594 (2017)
8. Tolk, A.: The next generation of modeling & simulation: integrating big data and deep learning. In: Proceedings of the Conference on Summer Computer Simulation, pp. 1–8 (2015)
9. Laroque, C., Skoogh, A., Gopalakrishnan, M.: Functional interaction of simulation and data analytics – potentials and existing use-cases. In: Wenzel, S., Peter, T. (eds.) Simulation in Produktion und Logistik 2017, Kassel University Press, Kassel, pp. 403–412 (2017)
10. Saadawi, H., Wainer, G., Pliego, G.: DEVS execution acceleration with machine learning. In: Proceedings of the Symposium on Theory of Modeling & Simulation (TMS-DEVS 2016), Pasadena, California (2016)
11. Wang, L.-C., Marek-Sadowska, M.: Machine Learning in Simulation-Based Analysis, pp. 57–64 (2015)
12. Afram, A., Janabi-Sharifi, F.: Review of modeling methods for HVAC systems. Appl. Thermal Eng. **67**(1–2), 507–519 (2014)
13. Deist, T., Patti, A., Wang, Z., Krane, D., Sorenson, T., Craft, D.: Simulation assisted machine learning. arXiv preprint arXiv:1802.05688 (2018)
14. Shao, G., Shin, S.-J., Jain, S.: Data analytics using simulation for smart manufacturing. In: Proceedings of the 2014 Winter Simulation Conference, pp. 2192–2203 (2014)
15. Zeigler, B.P., Kim, T.G., Praehofer, H.: Theory of Modeling and Simulation, 2nd edn. Academic Press, Inc., Orlando (2000)
16. Barros, F.J.: Modeling formalisms for dynamic structure systems. ACM Trans. Model. Comput. Simul. **7**(4), 501–515 (1997)
17. Chow, A.C.H., Zeigler, B.P.: Parallel DEVS: a parallel, hierarchical, modular, modeling formalism. In: Proceedings of the 26th Conference on Winter Simulation, San Diego, CA, USA, pp. 716–722 (1994)
18. Wainer, G.A.: Cellular modeling with Cell-DEVS: a discrete-event cellular automata formalism. In: Wąs, J., Sirakoulis, G.C., Bandini, S. (eds.) ACRI 2014. LNCS, vol. 8751, pp. 6–15. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11520-7_2
19. Vangheluwe, H.L.M.: DEVS as a common denominator for multi-formalism hybrid systems modelling. In: CACSD. Conference Proceedings. IEEE International Symposium on Computer-Aided Control System Design (Cat. No.00TH8537), pp. 129–134 (2000)

20. Boukelkoul, S., Redjimi, M.: Mapping between Petri nets and DEVS models. In: 2013 3rd International Conference on Information Technology and e-Services (ICITeS), pp. 1–6 (2013)
21. Kofman, E., Junco, S.: Quantized-state systems: a DEVS approach for continuous system simulation. Trans. Soc. Comput. Simul. Int. **18**(3), 123–132 (2001)