McCulloch-PittsArtificial Neuron and Rosenblatt's Perceptron: An abstract specification in Z

McCulloch-Pitts Artificial Neuron y Rosenblatt's Perceptron: una especificación abstracta en Z

W. Zamora-Cárdenas,
Escuela de Computación
Instituto Tecnológico de Costa Rica, Costa Rica.
wizaca23@estudiantec.cr

M. Zumbado
Escuela de Computación Instituto Tecnológico de Costa Rica, Costa Rica.
mzumbadog@estudiantec.cr

Trejos-Zelaya
Escuela de Computación
Instituto Tecnológico de Costa Rica, Costa Rica.
itrejos@tec.ac.cr

Fecha de recibido: 01 de marzo de 2020

Fecha de aprobado: 7 de abril de 2020

Abstract-The human brain is the most complex biological structure known; it can be seen as a highly complex, nonlinear and parallel computer capable of organizing its structural units (neurons) to perform different tasks, such as pattern recognition, perception, and motor control-and in many times-faster than nowadays' computers. The McCulloch-Pitts Artificial Neuron was the first attempt to model the behavior of a biological neuron, serving as a base model for Rosenblat's Perceptron. The Perceptron has been recognized as a fundamental step towards automated learning in the pattern recognition field. This paper will present a Z-notation abstract specification of the Perceptron and its learning process.

Key words- Specification in Z, neurons, brain, human, Rosenblat's Perceptron

Resumen- El cerebro humano es la estructura biológica más compleja conocida; Puede verse como una computadora altamente compleja, no lineal y paralela capaz de organizar sus unidades estructurales (neuronas) para realizar diferentes tareas, como reconocimiento de patrones, percepción y control motor, y en muchas veces más rápido que las computadoras de hoy en

día. La neurona artificial McCulloch-Pitts fue el primer intento de modelar el comportamiento de una neurona biológica, sirviendo como modelo base para el perceptrón de Rosenblat. El Perceptron ha sido reconocido como un paso fundamental hacia el aprendizaje automatizado en el campo del reconocimiento de patrones. Este artículo presentará una especificación abstracta de notación Z del Perceptron y su proceso de aprendizaje.

Palabras clave- especificación en Z, neuronas, cerebro, humano, perceptrón de Rosenblat

I. INTRODUCTION

The human brain is the most complex structure known in science; one of the most difficult and exciting challenges is the understanding of its operation [1]. The human brain can be seen as a highly complex, nonlinear, and parallel computer [1], [2], capable of organizing its structural units (neurons) to perform different tasks, such as pattern recognition, perception, and motor control- and in many times-faster than nowadays' computers. The human brain's outstanding performance comes from its great structure and capability to build its own rules through what humans know as experience [2]. The plasticity of the neurons allows the nervous system to adapt to its surrounding environment, being essential to the functioning of neurons as an information-processing unit in the human brain [2].

There are about 10_{11} electrically active neurons in the brain [1]. Most of them share the common features shown in Figure 1: the dendrites provide a set of inputs to the neuron while the axon acts as an output, and the synapse serves as the communication channel between neurons [1], [3]. Neurons are normally connected to many other neurons, so the information processing may appear relatively slow. This is far from true; the synapse occurs simultaneously, so massive parallelism of information processing takes place [1]. Hence, the neurons offer effective processing power together with fault tolerance. The latter is justified as neurons die on a daily basis with a few adverse effects on their performance [1], [2].

The neurons' behavior is normally binary, meaning that they either fire an electrical impulse (*i.e.* action potential) or not. This electrical impulse propagates from the cell's body through the axon, and when it reaches a synapse, the chemical neurotransmitters are released to the next neuron (see Figure 1). The synapse has an associated weight to determine the magnitude of the electrical impulse's effect on the post-synaptic neuron; therefore, the impulse can either be an excitatory synapse or an inhibitory synapse with impact on the activation probability of the post-synaptic neuron [1].

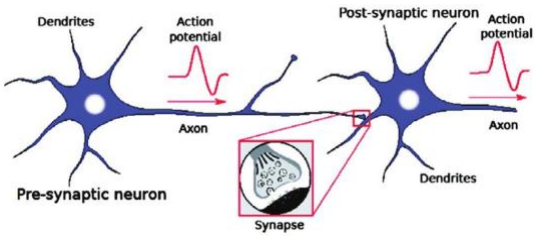

Fig. 1. Neuron Synapse [3]

The human nervous system can be seen as a three- stage system (see Figure 2) in which the neural network continually receives information in order to make appropriate decisions [2]. The input of the neural network comes from the receptors that convert the stimuli from the external environment into electrical impulses. The effectors transform the electrical impulses from the neural network output to discernible responses as system outputs. The interaction among these three components goes from left-to-right (feed-forward) and right-to-left (feedback), making the learning of the neural networks sensitive to both the input impulse and the output impulse.

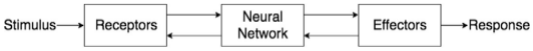

Fig. 2. Nervous system diagram [2]

A formal specification is a way to precisely describe the functional properties and behavior of an information system using mathematical notation. It is an abstraction that helps to specify what the system does without delving into how it is achieved [4]. This was first presented by Jean-Raymond Abrial, Steve Schuman, and Bertrand Meyer. Afterward, it was developed within Oxford University's Programming Research Group [5].

The Z-notation uses mathematical data types to model data in a system and predicates logic notation to describe the system's behavior. The

formal specification is completely independent of the code; thus, it is a tool for precisely describing requirements and constraints at an early stage of the systems development. Later on, the programs can be verified to determine whether they satisfy the formally stated requirements and constraints, or be even derived via systematic transformation of (non-executable) specifications into programming code [6], [7].

Artificial Neural Networks (ANNs) formal specification models for neuron-level and network-level structure have been proposed [8]-[10], that focus on the importance of understanding the network behaviour rather than its implementation. They thus serve as mechanisms to validate the viability of the network and provide foundations for the expansion and exploration of concepts. In [9], the authors discuss a series of properties recommended in the formal specification of ANNs systems. A modeling language named Pann was developed using process calculi and hybrid algebra; however, the formalization becomes complex and the authors state that is not possible to present an instance of behaviour for a network of a thousand neurons due to the complexity of the equations involved [8]. Pann is an example of how critical it is to keep simplicity on the notation and enlightens a reason to use better- defined, yet powerful, modeling languages. Others have used formal definitions in order to perform a formal validation of the ANN [11], [12], where the goal is to verify the correctness of the model by reasoning over the network behavior. In [13], the authors present a proof assistant framework for ANNs to help developers detect implementation errors systematically, illustrating a use case of formal specification applied to developing machine learning systems.

This paper will show the first models created to resemble the aforementioned behavior (Section II). Section III will present a Z-notation abstract specification for the McCullochPitts Artificial Neuron and Rosenblatt's Perceptron. Finally, Section IV will provide some insights into the generated specification.

II. THEORETICAL FRAMEWORK

The biological function of the brain and the neuron behavior inspired the creation of artificial models whose goal has been to mimic the capability of acquiring knowledge from the environment through a learning process [2]. This research will address the first steps taken by W. McCulloch and W. Pitts to emulate the neuron's behavior. This paper we will review how F. Rosenblat created a learning process built on top of the McCulloch-Pitts Artificial Neuron, inspired by the brain learning process.

A. McCulloch-Pitts Artificial Neuron

The first model of an artificial neuron was presented in [14] (see Figure 3). This model can be described as a nonlinear function which transforms a set of input variables $\underset{x}{\rightarrow} = (x_1, x_2, \cdots, x_d)$ into an output variable y [1], [8]. The signal x_1 at input i is first multiplied by a weight w_1 (analogous to the synaptic weight in a biological neural network) and sent to the summing junction along with all the other inputs of (see Equation 1) where b is the bias (firing threshold in a biological neuron). The bias can be w_0 seen as a special case x_0 of a weight from whose input $= 1$; therefore, Equation 1 can be rewritten as in Equation 2.

$$a = \sum_{i=1}^{d} w_i x_i + b \tag{1}$$

$$a = \sum_{i=0}^{d} w_i x_i \tag{2}$$

The weights $\underset{w}{\rightarrow}$ can be of either sign, corresponding to excitatory or inhibitory synapses of biological neurons. Then the neuron output y (see Equation 3) is calculated by operating on nonlinear activation function $\varphi()$ (see Equation 4).

$$y = \varphi(a) \tag{3}$$

$$\varphi(a) = \begin{cases} 1, & a \geq A \\ 0, & a < A \end{cases} \tag{4}$$

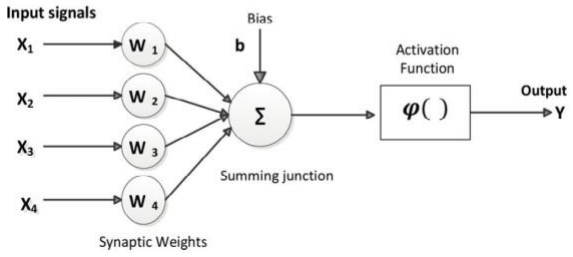

Fig. 3. McCulloch-Pitts artificial neuron [9]

B. Rosenblat's Perceptron

The perceptron (*i.e.* perceptron algorithm) is a single- layer network with a threshold-activation function [10] studied by F. Rosenblat and built around the McCulloch- Pitts nonlinear neuron [11]. The perceptron is a linear discriminant normally applied to classification problems [1], [12]. In the pattern recognition field, the perceptron was a great step towards automated learning using Artificial Neural Networks.

Rosenblatt's perceptron used a layer of fixed processing elements to transform the raw input data. These processing elements can be seen as the basis functions of a generalized linear discriminant with the form of adaptive weights w connected to a fixed nonlinear transformation of the input vector x denoted as ϕ and a threshold activation function (see Figure 4) [10], [12]. Similar to the McCulloch-Pitts neuron, the bias is introduced as an extra basis function with $\phi_0 = 1$and a corresponding weight w_0.

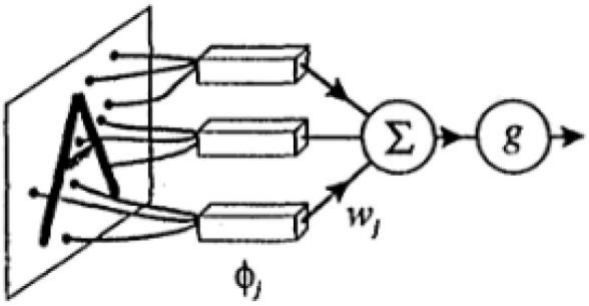

Fig. 4. Perceptron [10]

The perceptron output y is then defined as:

$$y = g(\sum_{j=0}^{M} w_j \phi_j(x)) = g(w^T \phi) \quad (5)$$

Where the threshold's g function is given by:

$$g(a) = \begin{cases} -1 & a < 0 \\ +1 & a \geq 0 \end{cases} \quad (6)$$

The perceptron error function (*i.e.* perceptron criterion) is defined in terms of the total number of misclassifications over the training set [10], [12]. The perceptron criterion is a continuous piecewise linear error function that considers each sample vector x^n with $n = 1, \cdots, N$and a target value t^n to minimize the error. The target t^n indicates the desired output y

(*i.e.* the class C membership) for the sample such that:

$$t^n = \begin{cases} -1 & x^n \in C_1 \\ +1 & x^n \in C_2 \end{cases} \quad (7)$$

From Equations 4 and 6, it can be noticed that we want $w^t\phi^n < 0$ for samples belonging to c_1 and $w^t\phi^n > 0$ for the samples in c_2; therefore, we want all samples to have $w^t(\phi^n t^n) > 0$. The perceptron criterion is then given by:

$$Ep(w) = -\sum_{\phi^n \in \mu} w^T(\phi^n t^n) \quad (8)$$

It should be noticed that only the set of misclassified samples μ are considered in equation 8. This happens because the perceptron criterion associates 0 error for correctly classified samples [10], [12]. The learning process for the perceptron is carried out by the application of a stochastic gradient descent algorithm [10], [12] to the error function in Equation 6. The change on the weight vectors $\underset{w}{\rightarrow}$ in the step τ with the learning rate α is then defined as:

$$w^{(\tau+1)} = w^{(\tau)} - \alpha\nabla Ep(w) = w^{(\tau)} + \alpha\phi^n t^n \quad (9)$$

The perceptron algorithm guarantees that-if the dataset is linearly separable (see Figure 5)-the learning Equation 8 will always find a solution in a finite number of steps [10], [12]; this is known as the perceptron convergence theorem. On the other hand, if the dataset is not linearly separable (see Figure 5), learning Equation 8 will never terminate. Also, stopping the perceptron at an arbitrarily step τ will not ensure the algorithm's capability to generalize for new data [10], [12]. The previously mentioned downsides are closely related to the basis functions being fixed and not adaptive to the input dataset [16], [18].

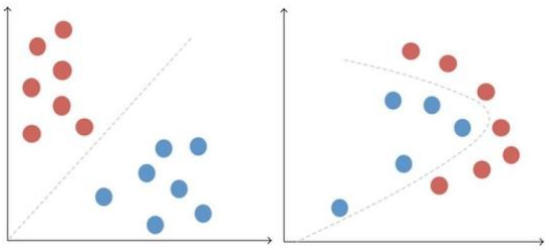

Fig. 5. Linear vs nonlinear separable data [13]

III. Z SPECIFICATION

First, the data types to be used must be defined:

$$RR == \mathbb{Z}$$
$$BB ::= False \mid True$$

The neuron is the humans base schema as it serves to specify both the McCulloch-Pitts artificial neuron and the Perceptron. It is made of a sequence of weights, a sequence of inputs, and an activation value. The bias, as explained before, will be assumed to be part of the weights' sequence.

Neuron

$$weights : \text{seq}_1\ RR$$
$$input : \text{seq}_1\ RR$$
$$activation : RR$$
$$target : \mathbb{Z}$$

Since the weights of the neuron are randomly initialized, it will be defined that a neuron cannot be initialized without weights.

InitNeuron

$$Neuron'$$
$$weights? : \text{seq}_1\ RR$$

$$weights' = weights?$$
$$input' = \langle\rangle$$
$$activation' = 0$$
$$target' = 0$$

The perceptron's core support functions are now specified.

$$dotProduct : (\text{seq}_1\ RR \times \text{seq}_1\ RR) \pfun RR$$
$$vectorByScalar : (\text{seq}_1\ RR \times RR) \fun \text{seq}_1\ RR$$
$$vectorSum : (\text{seq}_1\ RR \times \text{seq}_1\ RR) \pfun \text{seq}_1\ RR$$

$$\forall s1, s2 : \text{seq}_1\ RR \mid \#s1 = \#s2 \bullet (s1, s2) \in \text{dom}\ dotProduct$$
$$\forall s1, s2 : \text{seq}_1\ RR \mid \#s1 = \#s2 \bullet (s1, s2) \in \text{dom}\ vectorSum$$

As mentioned before, the Perceptron is built on top of the McCulloch-Pitts artificial neuron; hence, the perceptron will be specified using the neuron specification as a base. The creation of the Perceptron will be restricted to have a learning rate (alpha) greater than 0, a non-empty sequence of inputs and targets.

PerceptronParams

$$alpha : RR$$
$$inputs : \text{seq}_1\ (\text{seq}\ RR)$$
$$targets : \text{seq}_1\ \mathbb{Z}$$

$$alpha > 0$$

PerceptronError

$$error : \text{seq}\ RR$$

Perceptron

$$Neuron$$
$$PerceptronParams$$
$$PerceptronError$$

InitPerceptron

$$Perceptron'$$
$$PerceptronParams?$$
$$InitNeuron$$

$$alpha? > 0$$
$$inputs? \neq \langle\rangle$$
$$targets? \neq \langle\rangle$$
$$\theta\ PerceptronParams' = \theta\ PerceptronParams?$$

$$PerceptronUpdate \mathrel{\widehat=} \Delta Perceptron \land \Xi PerceptronParams \land \Xi Neuron$$

$$NeuronUpdate \mathrel{\widehat=} \Delta Perceptron \land \Xi PerceptronParams \land \Xi PerceptronError$$

$$NeuronUpdateWeights \mathrel{\widehat=} \Xi Neuron \setminus (weights, weights') \land \Delta Perceptron$$

Using the dotProduct definition, Equation 2 can be defined as:

NeuronActivation

$$NeuronUpdate$$

$$activation' = dotProduct\,(weights, input)$$
$$weights' = weights$$
$$inputs' = inputs$$
$$target' = target$$

The Perceptron learning process is performed with a finite number of iterations in which each iteration carries out the same forward and backward operations on each of the input sequences. The following pseudocode snippet describes this algorithm:

```
function PerceptronTrain
  for (i = 1; i<=iterations; i++)
    for (j = 1; j<=sizeof(inputs); j++)
      SetNeuronInput(inputs[j], targets[j])
      NeuronActivation()
      if(PerceptronCriterion() == false)
          NeuronConditionalUpdateWeights()
```

Following the algorithm's steps, the need to specify three new operations is identified. The perceptron forward calculates the neuron-activation value of a given input, so the input must be set to the neuron before obtaining its activation. The schema to modify the neuron's input is defined as follows:

SetNeuronInput

$$NeuronUpdate$$
$$newInput? : \operatorname{seq} RR$$
$$newTarget? : \mathbb{Z}$$

$$\#newInput? = \#input$$
$$weights' = weights$$
$$activation' = activation$$
$$input' = newInput?$$
$$target' = newTarget?$$

The perceptron criterion for a single input can be specified thus:

PerceptronCriterion

$$PerceptronUpdate$$
$$correct! : BB$$

$$correct! = \textbf{if } activation * target > 0 \textbf{ then } True \textbf{ else } False$$
$$error' = \textbf{if } correct! = True \textbf{ then } error \textbf{ else } vectorByScalar\,(input, target)$$

Then we specify how neuron weights are updated:

NeuronConditionalUpdateWeights

$$NeuronUpdateWeights$$
$$correct? : BB$$

$$weights' = \textbf{if } correct? = True \textbf{ then } weights \textbf{ else } vectorSum\,(weights, vectorByScalar\,(error, alpha))$$

The parts presented above can be composed using Z Schema Calculus to specify a neuron's update step.

$$UpdateStep \mathrel{\widehat=} NeuronActivation \fatsemi PerceptronCriterion \gg NeuronConditionalUpdateWeights$$

Training a Perceptron requires an iterative process involving the update of the neuron using a sequence of inputs.

$$\begin{array}{|l} BatchTrain : Perceptron \longrightarrow \mathrm{seq}\ Perceptron \\ \hline \forall\ Perceptron;\ ps : \mathrm{seq}_1\ Perceptron \\ \quad \mid ps\,1 = (\theta\, Perceptron) \\ \quad \land\ (\forall\, j : \mathrm{dom}\ inputs;\ newInput? : \mathrm{seq}\ RR; \\ \qquad newTarget? : \mathbb{Z};\ PerceptronUpdate \\ \quad \mid (\exists\ NeuronUpdate \\ \quad \mid ps\,j = \theta Perceptron \\ \quad \land\ SetNeuronInput \land newInput? = inputs\,j \\ \quad \land\ newTarget? = targets\,j \bullet UpdateStep\,) \\ \quad \bullet\ ps\,(j+1) = \theta Perceptron'\,) \\ \bullet\ BatchTrain\,\theta\, Perceptron = ps \end{array}$$

Finally, the Perceptron's Training process can be defined via the following schema:

$$\begin{array}{|l} PerceptonTrain \\ \hline \Delta Perceptron \\ iterations? : \mathbb{N}_1 \\ trainedPerceptron! : Perceptron \\ \hline \exists\ ps : \mathrm{seq}_1\ Perceptron \\ \mid ps\,1 = (\mu\ InitPerceptron \bullet \theta Perceptron') \\ \land\ (\forall\, i : 1\,..\,iterations? \\ \bullet\ \exists\ PerceptronUpdate \\ \mid ps\,i = \theta Perceptron \\ \land\ \theta Perceptron' = last\,(BatchTrain\,\theta Perceptron) \\ \bullet\ ps\,(i+1) = \theta Perceptron'\,) \\ \bullet\ trainedPerceptron! = last\,ps \end{array}$$

This operation performs several iterations that correspond to the stages during the perceptron's training. It delivers as output the final (last) approximation obtained by the training process.

IV. CONCLUSIONS & FUTURE WORK

The abstract specification provides an educational overview of two aspects. First, it shows how the Perceptron is built on top of McCulloch-Pitts artificial neuron by expanding its concepts and functions. Second, it enlightens the capabilities of the model to be adaptable to any input and processed by a variety of activation functions. The specification shown in this paper has a high level of abstraction as it focuses on the general workflow of the Perceptron rather than on the detail of the functions involved in the process. Therefore, the given specification can serve as a basis to expand the described Perceptron into Multi-Layered Perceptrons, and formally verifying them [20 - 28].

V. REFERENCES

[1] C. M. Bishop, "Neural networks and their applications," Review of scientific instruments, vol. 65, no. 6, pp. 1803- 1832, 1994.

[2] S. Haykin, Neural networks: a comprehensive foundation. Prentice Hall PTR, 1994.

[3] A. Huang, X. Zhang, R. Li, and Y. Chi, "Memristor neural network design," in Memristor and Memristive Neural Networks. IntechOpen, 2017.

[4] J. M. Spivey, The Z notation, 2nd. Ed. Prentice Hall International, 1992.

[5] J.-R. Abrial, S. Schuman, and B. Meyer, "Specification language," On the Construction of Programs, 1980.

[6] C. Morgan, Programming from Specifications, 2nd. Ed. Hertfordshire, UK: Prentice Hall International (UK) Ltd., 1994.

[7] J.-R. Abrial, The B-book: Assigning Programs to Meanings. New York, NY, USA: Cambridge University Press, 1996.

[8] R. J. Colvin, "Modelling and analysing neural networks using a hybrid process algebra," Theoretical Computer Science, vol. 623, pp. 15-64, 2016.

[9] S. A. Seshia, A. Desai, T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, S. Shivakumar, M. Vazquez-Chanlatte, and X. Yue, "Formal specification for deep neural networks," in International Symposium on Automated Technology for Verification and Analysis. Springer, 2018, pp. 20-34.

[10] D. Zaharakis and A. D. Kameas, "Modeling spiking neural networks," Theoretical computer science, vol. 395, no. 1, pp. 57-76, 2008.

[11] R. Ehlers, "Formal verification of piece-wise linear feed-forward neural networks," in International Symposium on Automated Technology for Verification and Analysis. Springer, 2017, pp. 269-286.

[12] L. Pulina and A. Tacchella, "An abstraction- refinement approach to verification of

artificial neural networks," in International Conference on Computer Aided Verification. Springer, 2010, pp. 243-257.

[13] D. Selsam, P. Liang, and D. L. Dill, "Developing bug- free machine learning systems with formal mathematics," in Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017, pp. 3047-3056.

[14] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," The bulletin of mathematical biophysics, vol. 5, no. 4, pp. 115-133, 1943.

[15] L. Guesmi, H. Fathallah, and M. Menif, "Modulation format recognition using artificial neural networks for the next generation optical networks," Advanced Applications for Artificial Neural Networks, p. 11, 2018.

[16] C. M. Bishop et al., Neural networks for pattern recognition. Oxford University Press, 1995.

[17] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." Psychological review, vol. 65, no. 6, p. 386, 1958.

[18] C. M. Bishop, Pattern recognition and machine learning. Springer, 2006.

[19] Z. M. Hira and D. F. Gillies, "A review of feature selection and feature extraction methods applied on microarray data," Advances in bioinformatics, vol. 2015, 2015.

[20] J. Abrial, M. J. Butler, S. Hallerstede, T. S. Hoang, F. Mehta, and L. Voisin, "Rodin: an open toolset for modelling and reasoning in Event-B," Int. J. Softw. Tools Technol. Transf., vol. 12, no. 6, pp. 447-466, 2010. [Online]. Available: https://doi.org/10.1007/s10009-010-0145-y

[21] G. Pan, M. Li, and G. Ou, "Modeling and reasoning Event-B models based on Mathematica," in Proceedings of the 11th Asia-Pacific Symposium on Internetware,

ser. Internetware '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3361242.3362773

[22] D. Plagge and M. Leuschel, "Validating Z specifications using the ProB animator and model checker," in Integrated Formal Methods, 6th International Conference, IFM 2007, Oxford, UK, July 2-5, 2007, Proceedings, ser. Lecture Notes in Computer Science, Davies and J. Gibbons, Eds., vol. 4591. Springer, 2007, pp. 480-500. [Online]. Available: https://doi.org/10.1007/978-3-540-73210-5 25

[23] A. Mashkoor, F. Yang, and J.-P. Jacquot, "Refinement- based validation of Event-B specifications," Softw. Syst. Model., vol. 16, no. 3, p. 789-808, Jul. 2017. [Online]. Available: https://doi.org/10.1007/s10270-016-0514-4

[24] J.-R. Abrial and S. Hallerstede, "Refinement, decomposition, and instantiation of discrete models: Application to Event-B," Fundam. Inf., vol. 77, no. 1-2, p. 1-28, Jan. 2007.

[25] J. Abrial, Modeling in Event-B - System and Software Engineering. Cambridge University Press, 2010. [Online]. Available: https://doi.org/10.1017/CBO9781139195881

[26] A. Fürst, T. S. Hoang, D. Basin, K. Desai, N. Sato, and K. Miyazaki, "Code generation for Event-B," in Integrated Formal Methods, E. Albert and E. Sekerinski, Eds. Cham: Springer International Publishing, 2014,pp. 323-338.

[27] Amrani, L. Lúcio, and A. Bibal, "ML + FV = ~ A survey on the application of machine learning to formal verification," CoRR, vol. abs/1806.03600, 2018. [Online]. Available: http://arxiv.org/abs/1806.03600

[28] S. Gulwani, P. Kohli, S. Russell, M. Girolami, A. Lowndes, F. Rossi, C. Szegedy, M. Vardi, M.

Vechev, A. Darwiche, M. Kwiatkowska, A. Platzer, and A. Nori, "Summit on machine learning meets formal methods," in Federated Logic Conference (FLoC), M. Kwiatkowska, Fijalkow, and S. Roberts, Eds. Oxford: University of Oxford, 2018.