MODELING OF BIOMETRIC IDENTIFICATION SYSTEM USING THE COLORED PETRI NETS

G. R. Petrosyan [a] , L. A. Ter-Vardanyan [a], A.V. Gaboutchian [b,]

[a] Institute for Informatics and Automation Problems of NAS RA, International Scientific - Educational Center of NAS RA, Armenia, Yerevan, petrosyan_gohar@list.ru, lilit@sci.am,

Russian Federation, Moscow, [b] armengaboutchian@mail.ru

KEY WORDS: Petri Net (PN), Colored Petri Net (CPN), position, transition, guard, identification, biometric system

ABSTRACT:

In this paper we present a model of biometric identification system transformed into Petri Nets. Petri Nets, as a graphical and mathematical tool, provide a uniform environment for modelling, formal analysis, and design of discrete event systems. The main objective of this paper is to introduce the fundamental concepts of Petri Nets to the researchers and practitioners, both from identification systems, who are involved in the work in the areas of modelling and analysis of biometric identification types of systems, as well as those who may potentially be involved in these areas. In addition, the paper introduces high-level Petri Nets, as Colored Petri Nets (CPN). In this paper the model of Colored Petri Net describes the identification process much simpler.

1. Introduction

Petri Nets (PN) are a graphical tool for formal description of the flow of activities in complex systems. Compared to other more popular techniques of graphical system representation (for instance, block diagrams or logical trees), PN are particularly matched for representation of logical interactions among parts or activities in a system in a natural way. Typical situations that can be modelled by PN are: synchronization, concurrency and conflict [1,2].

Definition. Petri Net $M(C,\mu)$ pair, where $C=(P,T,I,O)$ is the network structure and μ is the network condition. In structure C of a P-positions, T-transitions are finite sets. $I:T\to P^{\infty}, O:T\to P^{\infty}$ are the input and output functions, respectively, where P^{∞} are all possible collections (repetitive elements) of P. $\mu:P\to N_0$ is the function of condition, where $N_0=\{0,1,...\}$ is the set of integers. We determine (in a known manner) the allowed transitions of Petri Nets and the transitions from one state to another, as well the set of reachable states.

Places, transitions, and arcs are the basic Petri Net components. A Petri Net can be thought of as a bipartite graph consisting of two types of nodes, places and transitions. Places are displayed pictorially as circles (or ovals) and transitions are displayed as rectangles. An example Petri Net consisting of two places P1 and P2 and one transition T2 is shown in Figure 1. Note that arcs connect a place to a transition or a transition to a place, but they do not connect two places or two transitions.

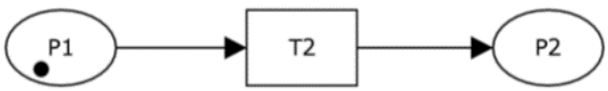

Fig. 1. Basic Petri Net configuration

The interpretation of places and transitions depends on the system being modeled. Places could represent resource status or operations. Arcs often represent the flow of data or resources. Transitions could represent the start/finish of processes. In terms of simulations, transitions can be used to model both *activities* and *events.* Activities can be thought of as the processes and logic of the system, while events occur at a single point in time and cause a change in the state of a system (White and Ingalls 2009). In fact, a transition may act as a super-process consisting of many sub-processes. This is where hierarchical nets come into play (which we will explain later). Often transitions can change the state of a net through the manipulation of *tokens* via the *firing rule* which is explained next.

Colored Petri Net (CPN) are considered as Classical Petri Net modern expansion which was created by K. Jensen [3]. Colored Petri Net (CPN) is a graphical oriented language for design, specification, simulation and verification of systems. It is in particular well-suited for systems that consist of a number of processes which communicate and synchronize.

Typical examples of application areas are communication protocols, distributed systems, automated production systems, work flow analysis. The CPN language allows the model to be represented as a set of modules, allowing complex nets (and systems) to be represented in a hierarchical manner. In the classical or traditional Petri Net tokens do not differ from each other, we can say that they are colorless. Unlike Classical Petri

Net in Colored Petri Net of a position can contain tokens of arbitrary complexity, example, lists, etc., that enables modeling more reliable models.

Definition. The mathematical definition of Colored Petri Net: CPN is a nine-tuple

$$CPN = (\Sigma, P, T, A, N, C, G, E, I)$$, where:

1. Σ is a finite set of non-empty types, also called color sets. In the associated CPN Tool, these are described using the language CPN-ML [6]. A token is a value belonging to a type.

2. P is a finite set of places. In the associated CPN Tool these are depicted as ovals/circles.

3. T is a finite set of transitions. In the associated CPN Tool these are depicted as rectangles.

4. A is a finite set of arcs. In the associated CPN Tool these are depicted as directed edges. The sets of places, transitions, and arcs are pairwise disjoint, that is

$$P \bigcap T = P \bigcap A = T \bigcap A = \varnothing.$$

N is a node function. It is defined from A into $P \times T \cup T \times P$. In the associated CPN Tool this depicts the source and sink of the directed edge.

5. C is a color-function, $C : P \to \Sigma$.

6. G is a guard function. It is defined from T into expressions such that:

$$t \in T : [Type(G(t)) = B \,\&\, Type(Var(G(t))) \subseteq \Sigma]$$

7. E is an arc expression function. It is defined from A into expressions such that:

$$\forall a \in A : [Type(E(a)) = \\ = C(p)_{MS} \,\&\, Type(Var(E(a))) \subseteq \Sigma],$$

where p is the place of $N(A)$ and $C(p)_{MS}$ denotes the multi-set type over the base type $C(p)$.

8. I is an initialization function. It is defined from P into closed expressions so that:

$$\forall p \in P : [Type(I(p)) = C(p)_{MS}].$$

The distribution of tokens, called marking, in the places of a CPN determines the state of a system being modelled.

CPN models can be constructed using CPN Tools, a graphical software tool used to create, edit, simulate, and analyze models. CPN Tools has a graphical editor that allows the user to create and arrange the various Petri Net components. One of the key features of CPN Tools is that it visually divides the hierarchical components of a CPN, enhancing its readability without affecting the execution of the model. CPN Tools also provides a monitoring facility to conduct performance analysis of a system. In addition, unlike traditional discrete event systems, CPNs allow for state space based exploration and analysis, which is complementary to pure simulation based analysis. State space analysis can be used to detect system properties such as the absence of deadlocks.

The dynamic behavior of a CPN is described in terms of the firing of transitions. The firing of a transition takes the system from one state to another. A transition is enabled if the associated arc expressions of all incoming arcs can be evaluated to a multi-set, compatible with the current tokens in their respective input places, and its guard is satisfied.

CPNs are an extension of ordinary Petri Nets. Petri Nets can be used to model a wide range of various systems. Thus in a CPN model, tokens can be coded as data values of a rich set of types (called color sets) and arc inscriptions can be computed expressions and not just constants. So that's the fundamental idea of CPNs: tokens have types, and each token type has some data value associated with it. Below the fold, we'll look at how we do that and what it means.

Colored Petri Nets add a collection of extensions to the other elements of the net to take advantage of typed tokens carrying values:

Each place in the net is also assigned a data type, and can only hold tokens of its assigned type.

Incoming edges of a transition can have *conditions*: the transition is only enabled when some set of tokens from the source places satisfy the full set of conditions for its incoming edges. The conditions for the incoming edges of a transition can reference the values from other incoming edges of the same transition – so, for example, the conditions can require that the values of two tokens coming from different incoming values match.

The edges going out of a transition can have *expressions* specifying how to compute the values of tokens being produced by the transition. When a transition is successfully fired, the expressions on its outgoing edges are evaluated to

produce new tokens to feed into the place at the end of the edge.

Let's look at a quick example of Figure 2 of CPN transition. Here's a very simple CPN. It's got 3 places: two of them have type (Int×String), and one has type (Int). The transition takes one token of the pair type, and one of the integer type; and it produces one token of the pair type. The edges coming into the transition declare names for the elements of the token values, and the edge leaving the transition describes how to generate the values for outgoing tokens. This little net starts with two tokens, (4,"Foo"), and (2). The transition only fires if the integer token has a value greater than or equal to one, and produces a token multiplying the two integers from the incoming token. So the transition would fire, consuming two tokens shown in the graph, and producing a token (8, "Foo") in the bottom place.

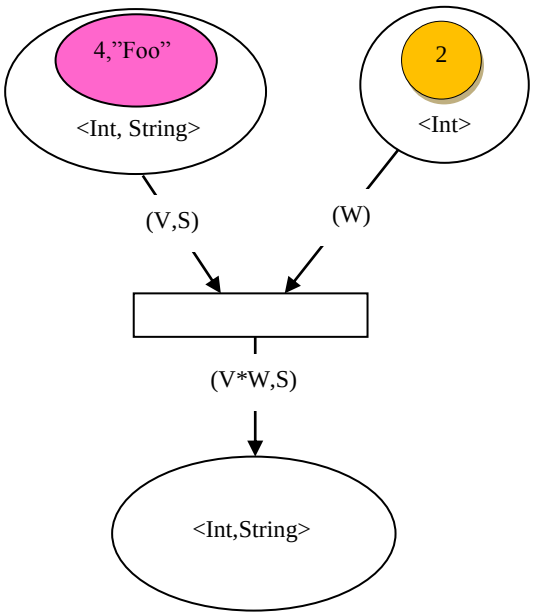

Fig.2 Example of CPN transition

For colored nets, there are a ton of variations. The basic idea is that there's some simple version of λ-calculus which is associated with elements of the CPN:

- Places are assigned lambda calculus types, which define the types of values carried by tokens that can be located in the place. At runtime, the place is a *bag* of tokens.
- Transitions are basically functions, where the incoming edges define a tuple of input parameters to the function, and the outgoing edges form a tuple of results from the function. For our example, the transition is, basically "&lamba; (v,s),w → (v*w,s)".
- The transition can have *conditions* for its firing: the condition will only be enabled for incoming token values which meet the condition. In the example, the condition is that the value coming on the edge from the integer-place must have a value greater than or equal to one. This turns the lambda function into a guarded partial function: in our example,

"&lamba; (v,s),w | w >= 1 → (v*w,s)".

Biometrics is often used by companies, governments, military, border control, hospitals, banks etc. to either verify a person's identity, for physical access control, computer log-in, welfare disbursement, international border crossing and national ID cards, e-passports, allowing access to certain building area or to identify individuals to retain information about them, i.e. criminals, forensics, etc. In automobiles, biometrics is being adopted to replace keys for keyless entry and keyless ignition [7].

The objective of a biometrical identification system is to identify individuals on the basis of physical (passive or active) features. One of the oldest and probably best known of such features is the human fingerprint. One can safely say that for a long time fingerprinting-based identification and biometrical identification have been seen as one and the same thing.

The last decade other human features have become practical, and there is now an active research community on iris-based recognition, face recognition and others [8].

Biometric identification systems were studied by O'Sullivan and Schmid [9] and Willems et al. [10]. They assumed storage of biometric enrollment sequences in the clear and determined the corresponding identification capacity. Later Turcel [11] analysed the trade-off between the capacity of a biometric identification system and the storage space (compression rate) required for the biometric templates. It should be noted that Turcel's method realizes a kind of privacy protection scheme. Recall that secrecy capacity introduced by Ahlswede and Csiszar [12] can be regarded as the amount of common secret information that can be obtained in an authentication system in which helper data are (publicly) available. Interestingly this secrecy capacity, which is equal to the mutual information between enrollment and authentication biometric sequences in the biometric setting, equals the identification capacity found by O'Sullivan and Schmid and Willems et al.

2. Model description

Biometrical identification in general involves two phases (Fig.3). In an enrollment phase all individuals are observed and for each individual a record is added to a database. This record contains enrollment data, i.e. a noisy version of the biometrical data corresponding to the individual. In the identification phase an unknown individual is observed again.

The resulting identification data of an unknown individual, is compared to (all) the enrollment data in the database and the system has to come up with an estimate of the individual.

Essential in this procedure is that both in the enrollment phase and in the identification phase noisy versions of the biometrical data are obtained. The actual biometrical data of each individual remains unknown.

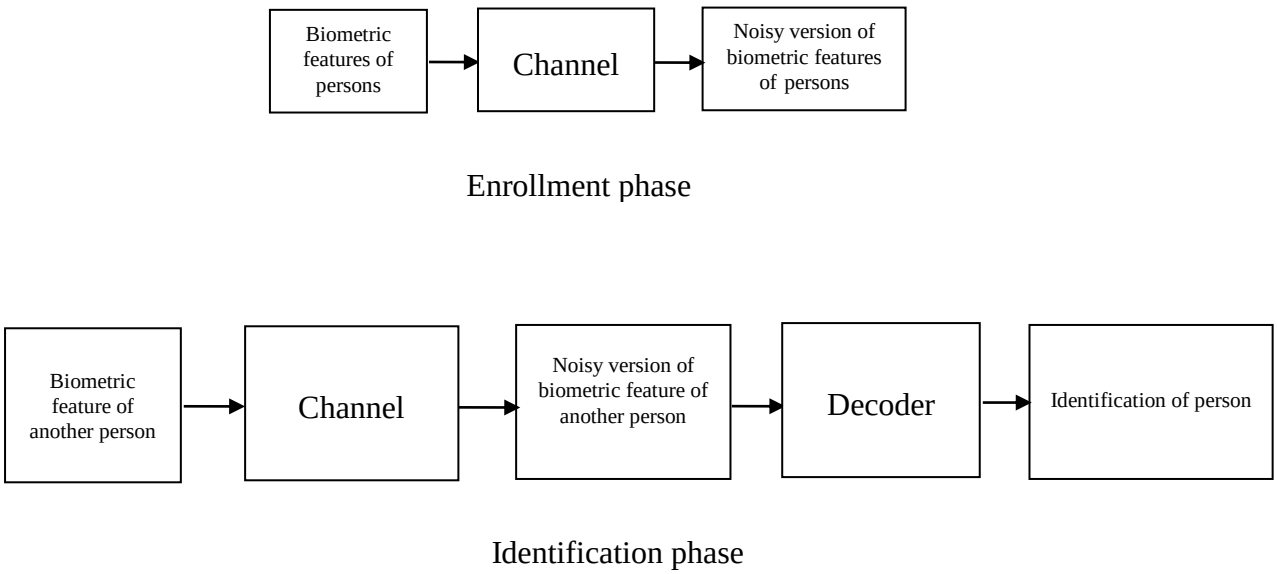

Fig3. The model of biometric identification system

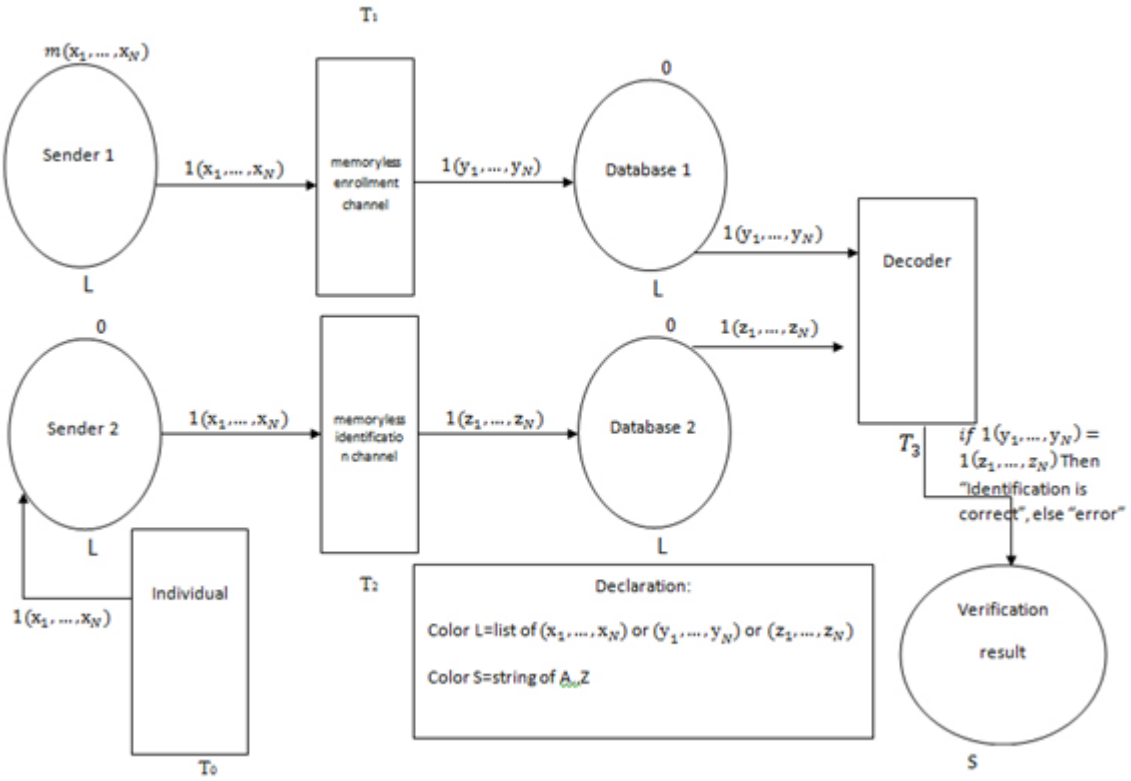

Fig.4 The model of biometric identification system by Colored Petri Net

In Fig. 4 Colored Petri Net consists of the following positions: *Sender 1, Sender2, Database1, Database2, Verification result* and the following transitions: T_0, T_1, T_2, T_3, which have corresponding names. *Sender1* includes N-dimensional biometric data from m-dimensional set. Element $(x_1,...x_N)$ passes through the channel to T_1 transition, and its corresponding encoded data $(y_1,...y_N)$ is placed in *Database1*.

After the firing of T_0 transition, the element of m-dimensional set is placed in *Sender2* position, after that the result of firing of T_2 transition, through the respective channel, it is obtained the encoded data $(z_1,...z_N)$, which is placed in the *Database2* position.

Through the firing of T_3 transition, it is checked the correspondence of vectors $(y_1,...y_N)$ and $(z_1,...z_N)$, gets the responce: if the identification is correct, or there is an error in the system. The arc, which is directed from T_3 transition to the *Verification result* position, is bound by a corresponding logical expression.

Then firing of T_0 transition, the next element can queue up and the cycle can be repeated again.

Sender1, Sender2, Database1, Database2 positions that are attached to the **L** type, which is a set of N-dimensional vectors. The **S** type is attached to *Verification result* position, which is presented as a type string. The corresponding information about the types is shown in the declaration table.

3. Conclusion

In the result of further studies, Colored Petri Network will be built, modelling complex Biometric Identification Systems, which will explore verification, validation, error detection and functional interactions problems in the Biometric Identification Systems.
The obtained results of the study will be used for processing of biometric data used in medical equipment, reducing time and resource consumption.

REFERENCES

Peterson, James Lyle (1981).
Petri Net Theory and the Modelling of Systems. Prentice Hall.
ISBN 0-13-661983-5

Tadao Murata. "Petri nets: Properties, Analysis and Applications." Proc. of the IEEE, 77(4), 1989.

K. Jensen. Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. Springer - Verlag, Berlin, 1992.

Jensen K. Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. Springer, 1996. Vol. 1–3.

S. Pankanti, R. M. Bolle and A. Jain, "Biometrics-The Future of Identification", IEEE Computer, vol. 33, no.2, pp. 46-49, February, 2002.

J. D. Ullman, "Elements of ML Programming," Prentice- Hall, Upper Saddle River, 1998.

S. Pankanti, R. M. Bolle and A. Jain, "Biometrics – The Future of Identification", IEEE Computer, V33, N2, pp. 46-49, 2002.

J. A. O'Sullivan and N. A. Schmid, "Performance prediction methodology for biometric systems using a large deviations approach", IEEE Trans. On it Signal Proc., vol. 52, no. 10, pp. 3036-3045, 2004.

F. WIllems, T. Kalker, J. Goselig, and J.-P. Linnartz, "On the capacity of a biometric identification system", Intenational Symposium on Information Theory, Yokohama, Japan, p. 82, 2003.

E. Trucel, "Capacity/storage tradoff in high-dimentional identification system", IEEE International Symposium on Information Theory, Washington, USA, pp 1929-1933, 2006.

R. Ahlswede and Csiszar, "Common randomness in information theory and cryptography – Part I: Secret sharing", IEEE Trans. Information Theory, vol. IT-39, pp. 1121-1132, July 1993.