Joint Conference of 1996 International Computer Symposium
December 19~21, Kaohsiung, Taiwan, R.O.C.

Synthesizing Timed Transitions and Timed Places in a Petri Net*

Steve J. Yang[1], Jeffrey J.P. Tsai [2], Eric Y.T. Juan[2], and Zheng Hao, Lai[1]

[1]Department of Computer Science and Information Engineering,
National Central University, ROC
[2]Department of Electrical Engineering and Computer Science,
University of Illinois at Chicago, USA

Abstract

Our research is to extend Petri nets by associating timing constraints with places and transitions in the original Petri nets, we called the extended places and transitions as timed places and timed transitions, respectively, and called the extended Petri nets as timing constraint Petri nets (TCPNs). TCPNs can be used to model and verify whether a real-time system specification satisfies the imposed timing constraints. Based on the definition of TCPNs, in this paper, we have defined three synthesis rules—"sequential" synthesis rule, "or" synthesis rule, "and" synthesis rule to synthesize a time range. The time range can be either a span of fireable time (earliest and latest fireable times) associated with each timed transition or a span of enabling time (earliest and latest enabling times) associated with each timed places. A TCPN is verified to be satisfied with timing constraints if none of the synthesized spans of time is negative. The synthesis rules presented in this paper can be used not only in TCPNs, but also in other time-related extensions of Petri nets.

Key words: Petri nets, timing constraints, verification, synthesis, real-time systems, specification.

1. Introduction

Real-time systems are becoming more and more important to our everyday life. Examples include command and control systems, flight control systems, space shuttle landing control systems, aircraft avionics control systems, robotics, patient monitoring systems, and nuclear power plant control systems. These systems are often required to interact with the environment that evolves independently from the control systems. A common character of these the correctness of such systems. This time-related correctness distinguishes real-time systems from non-real-time systems. For non-real-time systems, time simply affects the performance, but not the correctness. For real-time systems, time plays the most critical roles, a delay or a failure of the systems to react to certain input signals from the environment within some specified time bounds may results in severe damage or even fatal disasters. In summary, real-time systems differ from non-real-time systems because of the timing constraints. Therefore, the most essential issue regarding specifying and analyzing real-time systems is to find timing constraint violations (or timing errors in general).

Real-time system specification and verification tries to find the most timely valid responses for the system. Most of the verification methods proposed nowadays are able to verify whether a timely response is valid with respect to the timing requirements, but not many of them can synthesize the approximate range of such timely responses.

Timing constraint Petri nets (TCPNs) are derived from the concepts of timing constraints and Petri nets [16]. TCPNs extend Petri nets by associating a pair of minimum timing constraint and maximum timing constraint ($TC_{min}(p_j)$, $TC_{max}(p_j)$) with each place p_j, and ($TC_{min}(t_j)$, $TC_{max}(t_j)$) associated with each transition t_j, and a duration timing constraint [$FIRE_{dur}(t_j)$] associated with each transition t_j. We called the extended places and transitions as timed places and timed transitions, respectively, and called the extended Petri nets as timing constraint Petri nets (TCPNs). TCPNs can be used to model and verify whether a real-time system specification satisfy the imposed timing constraints. Besides proving those responses are timely with respect to the specified timing requirements, the other objective of our research is to synthesize the best approximation of time range for the timely response. Based on the definition of TCPNs, in this paper, we have defined three synthesis rules—"*sequential*" synthesis rule, "*or*" synthesis rule, "*and*" synthesis rule-- to synthesize a time range. The time range can be either a span of fireable time (earliest and latest

*This research is supported by National Science Council under the grant NSC 86-2213-E-008-030.

fireable times) associated with each timed transition or a span of enabling time (earliest and latest enabling times) associated with each timed places. A TCPN is verified to be satisfied with timing constraints if none of the synthesized spans of time is negative. The synthesis rules presented in this paper can be used not only in TCPNs, but also in other time-related extensions of Petri nets.

The rest of the paper is organized as follows. In Section 2 we present numerous time related extensions of the Petri nets and compare them with our TCPNs. Section 3 presents the three synthesis rules based on the TCPNs. The computation procedures for synthesizing the spans of firing time for real-time system specifications is also presented in this section. Section 4 addresses our analysis-via-synthesis method for verifying the real-time system specification. We conclude this paper with our future research in Section 5.

2. Petri Nets for Real-Time Systems

Petri nets (PNs) have gained popularity in recent years because of their ability to model and analyze concurrent systems [9]. However. the concept of time is not explicitly provided in PNs. which limits their usefulness for real-time systems. Many efforts to extend PNs can be found in the areas of temporal behavior analysis [1,2,3,4,6,8,10,15] and performance evaluation [5,11,12,13,14]. Most of the extensions have been achieved by imposing additional timing constraints onto the enabling and firing rules of the original PNs. Therefore. I conduct the following comparison from the enabling and firing rules point of view. as well as from the timing constraints point of view.

We classify enabling rules as two types: *typeless enabling rules* and *typed enabling rules*. *Typeless enabling rules* treat all tokens as the same. Therefore, for the enabling of a transition, t_j, one only considers the presence of tokens in each input place of t_j. In contrast, *typed enabling rules* treat tokens individually so that each token may possess different attributes. Therefore, for the enabling of a transition, t_j, one not only considers the presence of tokens, but also the types of tokens, i.e., t_j is not enabled until each input place of t_j has the right combination of token types.

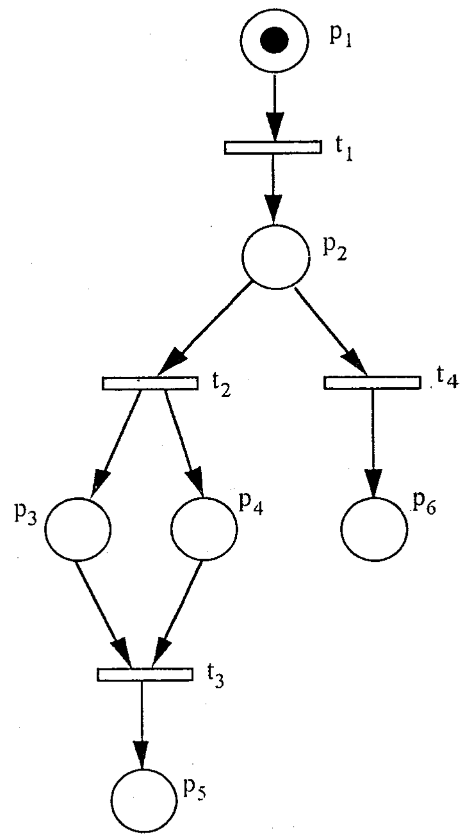

Figure 1(a). A fragment of a Petri Net.

Based on the classification of *typeless* and *typed enabling rules*, we classify firing rules as *typeless firing rules* and *typed firing rules*, respectively. *Typeless firing rules* are associated with *typeless enabling rules* which treat all tokens as the same. The result of a transition firing is implicit, which means tokens will be removed from each input place of t_j and added into each output place of t_j based on the arcs' weights. Petri nets use the *typeless enabling* and *firing rules*. In contrast, for those nets following *typed enabling rules*, the firing rules also have to be typed. The firing of a typed transition, t_j, will remove specific colored tokens from each input place of t_j and add specific colored tokens into each output place of t_j. As a result, a table is used to specify what combinations of input colored tokens can be used to enable transitions, and what combinations of colored tokens should be removed from input places and be added into output places after transition firing. Colored Petri nets are a typical example of nets that follow both typed enabling and firing rules.

We also define two firing modes based on how soon an enabled transition has to fire: *weak firing mode* and *strong firing mode*. The *weak firing mode* does not force any enabled transition to fire. In other words, an enabled transition may or may not fire. The *weak firing mode* is used in Petri nets modeling. The *strong firing mode* forces an enabled transition to fire immediately. In other words, a transition is forced to fire as soon as it is enabled. The *strong firing mode* is used in conflict-free firing Petri nets modeling [9]. The *strong firing mode* is not suitable for some nets with conflict-free structures because this mode will result in a contradiction, as explained in the following example. As shown in Figure 1(a), transitions t2 and t4 are in a conflict structure, i.e., only one transition can fire at a time. However, according to the definition of the *strong firing mode*, both t_2 and t_4 are enabled and begin to fire at the same time when the token arrives at p2, which results in a token being added into p_3, p_4, and p_6, respectively. This contradicts the definition of conflict structures, in which only one transition can fire.

The imposed timing constraints can be represented as constants or functions. The former includes timed Petri nets which treat a timing constraint as a single delay [5,12,13], and time Petri nets which treat a timing constraint as a time pair consisting of lower and upper bounds [6,8,14]. The latter includes stochastic Petri nets which treat a timing constraint as a probability function of transition firing rate [7,11], and ER nets which treat a timing constraint as a function of colored tokens in input places [3,4].

Timed Petri nets (timed PNs) were first proposed by Ramchandani [13] who examined the timing analysis of asynchronous concurrent systems. Ramamoorthy and Ho [12] extended the use of timed PNs to the area of performance evaluation. Timed PNs follow the *strong firing mode*, i.e., a transition, t_j, with a delayed time, T_{del}, will immediately fire at time when necessary tokens have arrived at time T_0. Before T_0, the tokens which have arrived are not preserved and can be used to enable other transitions if t_j is in a conflict structure. During the time period from T_0 to (T_0+T_{del}), the tokens are preserved for t_j so that no other transitions can use these tokens. At time (T_0+T_{del}), the tokens will and must be removed from t_j's input places to t_j's output places.

Stochastic Petri nets (SPNs) are also mainly used for performance evaluation. In contrast to the constant delay used in timed PNs, SPNs use the average delay which is a probability function of a transition's firing rate. Peng and Shin use generalized stochastic Petri nets (GSPNs) to model real-time control activities of a distributed system [11]. The modeled activities in GSPNs are then formed as a sequence of homogeneous continuous-time Markov chains (CTMC) in order to analyze the probability of missing deadlines. Timed and stochastic PNs are mostly used for performance evaluation, such as finding how fast a transition can initiate consecutive firing in a periodically operated timed PNs or SPNs. In other words, the performance is evaluated by finding a minimum cycle time for completing a firing sequence (each transition fires at least once) that leads back to the initial marking, i.e., finding the minimum cycle time for the execution of a periodical process.

Time Petri Nets (time PNs) were introduced by Merlin and Farber [8] for analyzing the recoverability of communication protocols. Time PNs are similar to timed PNs except that time PNs use a time pair instead of a single delay. A transition in time PNs is associated with (TC_{min}, TC_{max}), where TC_{min} represents the minimum delay and the TC_{max} represents a time-out. If a transition, t_j, is enabled at time T_0, then t_j can fire neither before (T_0+TC_{min}) nor after (T_0+TC_{max}). Since time PNs follow *strong firing mode*, if the firing does not take place during the time period from (T_0+TC_{min}) to (T_0+TC_{max}), then t_j must fire at (T_0+TC_{max}). Leveson and Stolzy use time PNs to model a real-time system for safety analysis [6]. To perform the safety analysis, a reachability graph of the time PNs which models the system behavior is first constructed to determine whether a high-risk state will be reached. The timing constraints imposed by the modeled system can then be derived to avoid such high risk states. Berthomieu and Diaz propose an enumerative method to analyze the temporal behavior of a concurrent system [1]. Through the technique of reachability analysis used in time PNs, Berthomieu and Diaz claim that their method can exhaustively validate the behavior of the modeled system.

Timing constraint Petri nets (TCPNs) were introduced by Tsai, Yang, and Chang [16]. TCPNs extend Petri nets by adding *minimum timing constraint* and *maximum timing constraint* pairs to places or transitions, and by adding *durational timing constraint* to transitions or places. For example, $(TC_{min}(t_2), TC_{max}(t_2))$ is denoted as (0,5) and

[FIRE$_{dur}$(t2)] is denoted as [6] as shown in Figure 1(b). The major difference between TCPNs and time (timed) Petri nets is that TCPNs follow the *weak firing mode* and the analysis method is based on either the relative or absolute time mode, whereas time (timed) Petri nets follow the *strong firing mode* and the analysis method is based on the relative time mode only. For example, due to the *strong firing mode*, transition t2 in time (timed) Petri nets must fire by 5 units of time once t2 has been enabled, this leave transition t4 has no chance to fire at all because t4 cannot fire until 6 units of time after t4 has been fired. In contrast, TCPNs follows *weak firing mode*, which are more suitable for systems with conflict structures (priority, decision, and choice structures) as shown in Figure 1(b).

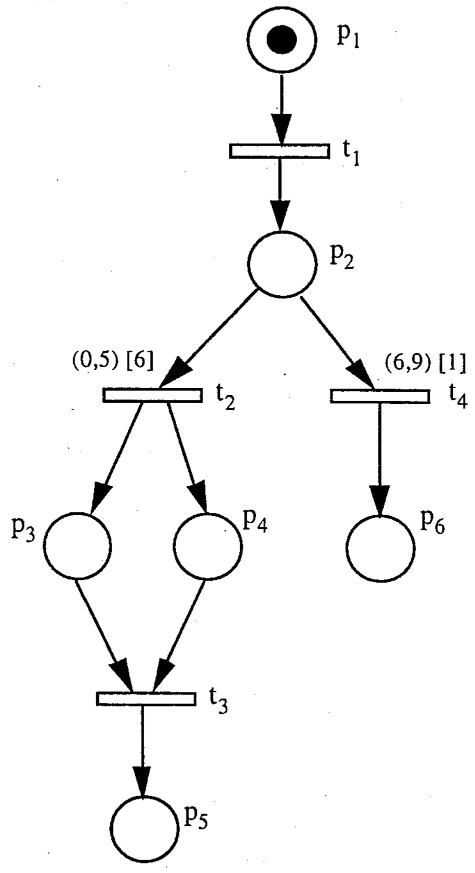

Figure 1(b). A fragment of a Petri Net with timing constraint

We summarize above extensions of PNs by distinguishing their difference in firing modes. Timed (stochastic) PNs use the *strong firing mode* in which a transition is forced to fire immediately after it is enabled. Firing of a transition will last for a period of time, and tokens will be preserved during the firing period. Time PNs use the *strong firing mode* in which a transition is forced to fire at time T_0+TC_{max} if the transition has neither been fired nor disabled due to other transitions' firing. In time PNs modeling, tokens will not be preserved because the firing of a transition is instantaneously. TCPNs follow the *weak firing mode*, which not only preserve the same firing mode used by PNs, but also is capable of modeling and analyzing conflict structures.

3. The Three Synthesis Rules

Based on the enabling and firing rules of TCPNs, three synthesis rules are presented here to compute the time instants of each timed transition and timed places with given imposed timing constraints. We construct the three synthesis rules which can be used
to synthesize span of fireable time associated with each timed transition and span of enabling time associated with each timed places. A TCPN is verified to be satisfied with timing constraints if none of the synthesized spans of time is negative. The synthesis rules presented in this paper can be used not only in TCPNs, but also in other time-related extension of Petri nets. The three synthesis rules are

- *SEQ* (sequential) synthesis rule,
- *OR* synthesis rule,
- *AND* synthesis rule.

3.1. Sequential Synthesis Rule

Please be noted that we have classified types of time into *time instant* and *time interval* which are denoted as @T and T, respectively. For example, an airplane from Chicago to Springfield is scheduled to departure at a time instant of @8:05:April-5-1995, and the flight will take a time interval of 30 minutes As shown in Figure 2, given a transition t_i may fired within a span of time ranging from time instant @a to @b, and place p_i is associated with timing constraint interval of (c,d) and transition t_j is associated with timing constraint interval (e,f). In Figure 2, we highlight the synthesized pair of earliest and latest enabling time instants of p_i as (@Xp_i ,@Yp_i), and the synthesized pair of earliest and latest fireable time instants of t_j ($@Xt_j$, $@Yt_j$). The rule for obtaining the time instants is as follows:.

$@Xp_i = @a + c$
$@Yp_i = @b + d$

$@Xt_j = @a+c+e = @Xp_i + e$
$@Yt_j = \min(@b+d, @b+c+f) = @b + \min(d, c+f)$

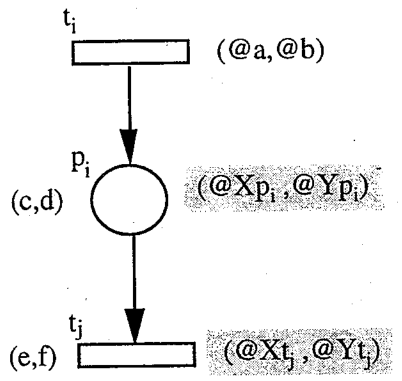

Figure 2. The *SEQ* synthesis rule.

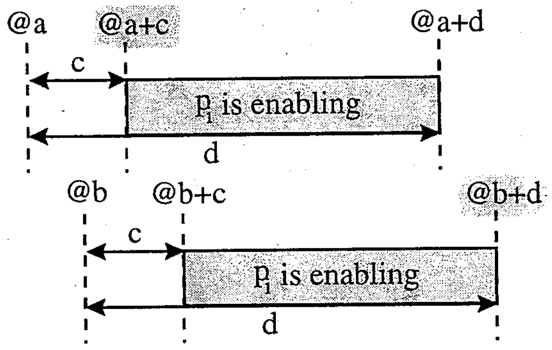

Figure 3. The earliest and latest enabling time instants of t_j in Figure 2.

As shown in Figure 3, since t_i may fired within a span of time ranging from time instant @a to @b, and place p_i is associated with timing constraint interval of (c,d), it is clear that p_i can begin its enabling to t_j as early as @a+c and as late as @b+d. Due to the minimum delay, the enabled t_j needs to wait for *e* amount of time before it can fire, thus the earliest time for t_j to be fireable is the earliest time that t_j is enabled plus e, which is @a+c+e as shown in Figure 2. Since t_j must remains enabled to be fireable, thus from Figure 4, we notice that the earliest time instant for t_j to be fireable is the minimum of (@b+d, @b+c+f). Thus, the result of ($@Xt_j$, $@Yt_j$) is (@a+c+e, @b+min(d,c+f)).

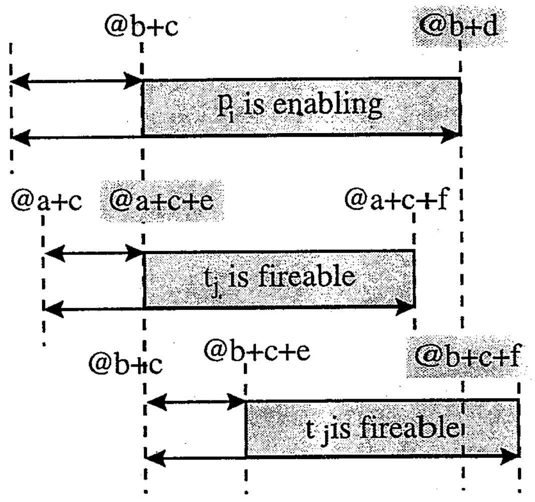

Figure 4. The earliest and latest fireable time instants of t_j in Figure 2.

3.2. *AND* and *OR* Synthesis Rules

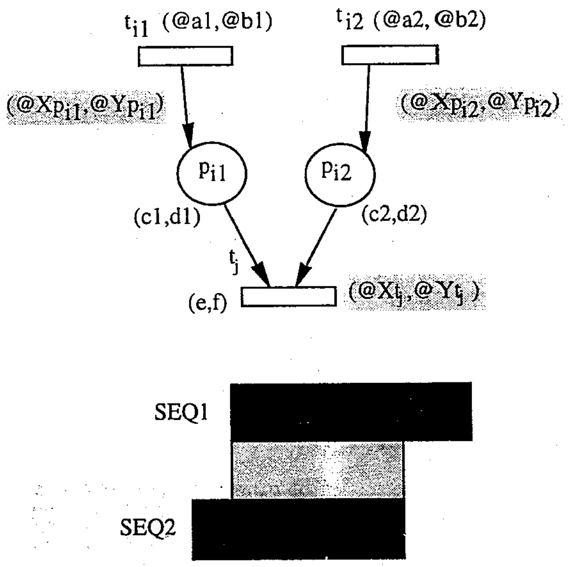

Figure 5. The *AND* synthesis rule.

As shown in Figure 5, to obtain the time instants of t_j within which t_j is fireable, both p_{i1} and p_{i2} have to remain enabling, thus this is an *AND* relationship.

The way we conduct this synthesis rule is to treat the firing sequence from t_{i1}, p_{i1}, to t_j as sequence 1 and the firing sequence from t_{i2}, p_{i2}, to t_j as sequence 2. Thus, the time instants associated with the timed places and transitions along the two sequences can be constructed by using the sequential synthesis rule, then we take a conjunction of the synthesis results.

In sequence 1, given t_{i1} is fireable from @a1 to @b1, then based on the *SEQ* synthesis rule,

$$@Xp_{i1} = @a1 + c1,$$
$$@Yp_{i1} = @b1 + d1.$$

Similarly, in sequence 2, let t_{i2} is fireable ranging from @a2 to @b2, then

$$@Xp_{i2} = @a2 + c2,$$
$$@Yp_{i2} = @b2 + d2.$$

Then we can take the conjunction of the two synthesized results, $@X_{tj}$ and $@Y_{tj}$ as shown in Figure 5, then we have

$$\begin{aligned} @Xtj &= \max(@Xpi1+e, @Xpi2+e) \\ &= \max(@Xpi1, @Xpi2)+e \end{aligned}$$

$$@Ytj = \min(@b1+\min(d1,c1+f), @b2+\min(d2,c2+f))$$

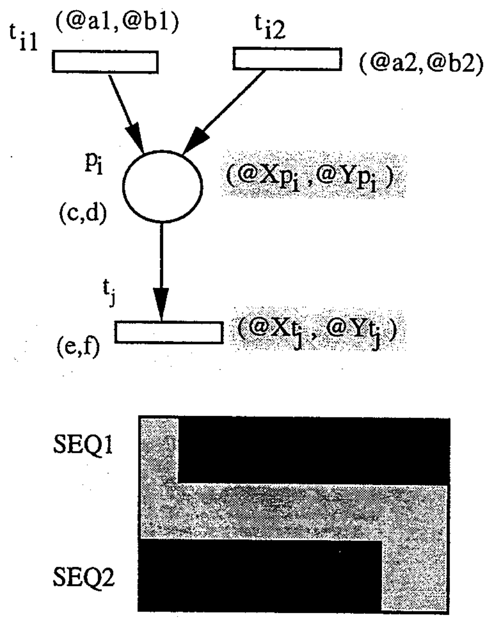

Figure 6. The *OR* synthesis rule.

The mechanism for deriving *OR* synthesis rule as shown in Figure 6, is similar to the one we have *AND* synthesis rule, the difference is in *OR* synthesis rule, we have to take a disjunction instead of conjunction to obtain the time instants associated with t_j.

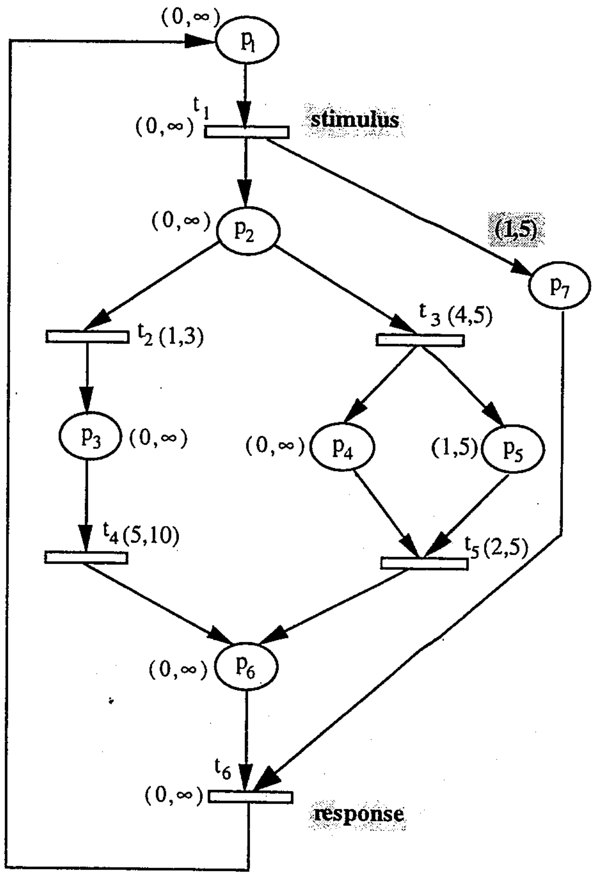

Figure 7. A real-time control system with timing constraints.

4. Analysis via Synthesis

The three steps for analyzing a real-time system specification via the result of synthesis is as follows:

1. Synthesizing span of enabling time associated with each timed places in a TCPN.
2. Synthesizing span of fireable time associated with each timed transitions in TCPN.
3. TCPN is verified to be satisfied if none of the synthesized spans of time is negative.

We use a hypothetical TCPN as shown in Figure 7 as an example to illustrate why the results of synthesis can verify the real-time specification. From Figure 7. if there is a stimulus occur, the system has to take actions in order to give a response back to the controlled environment, in addition, the response must occurs timely. i.e. within a time interval (1, 5) since the stimulus occurs. It is very difficult to tell whether the system can satisfy the timing constraint form Figure 7.

The result of synthesis using the three rules we just presented is shown in Figure 8. In Figure 8, if we know the time instant of the stimulus is @T, then the result of synthesis tell us that t_6 has no way to meet the imposed timing constraint because t_6 is never fireable.

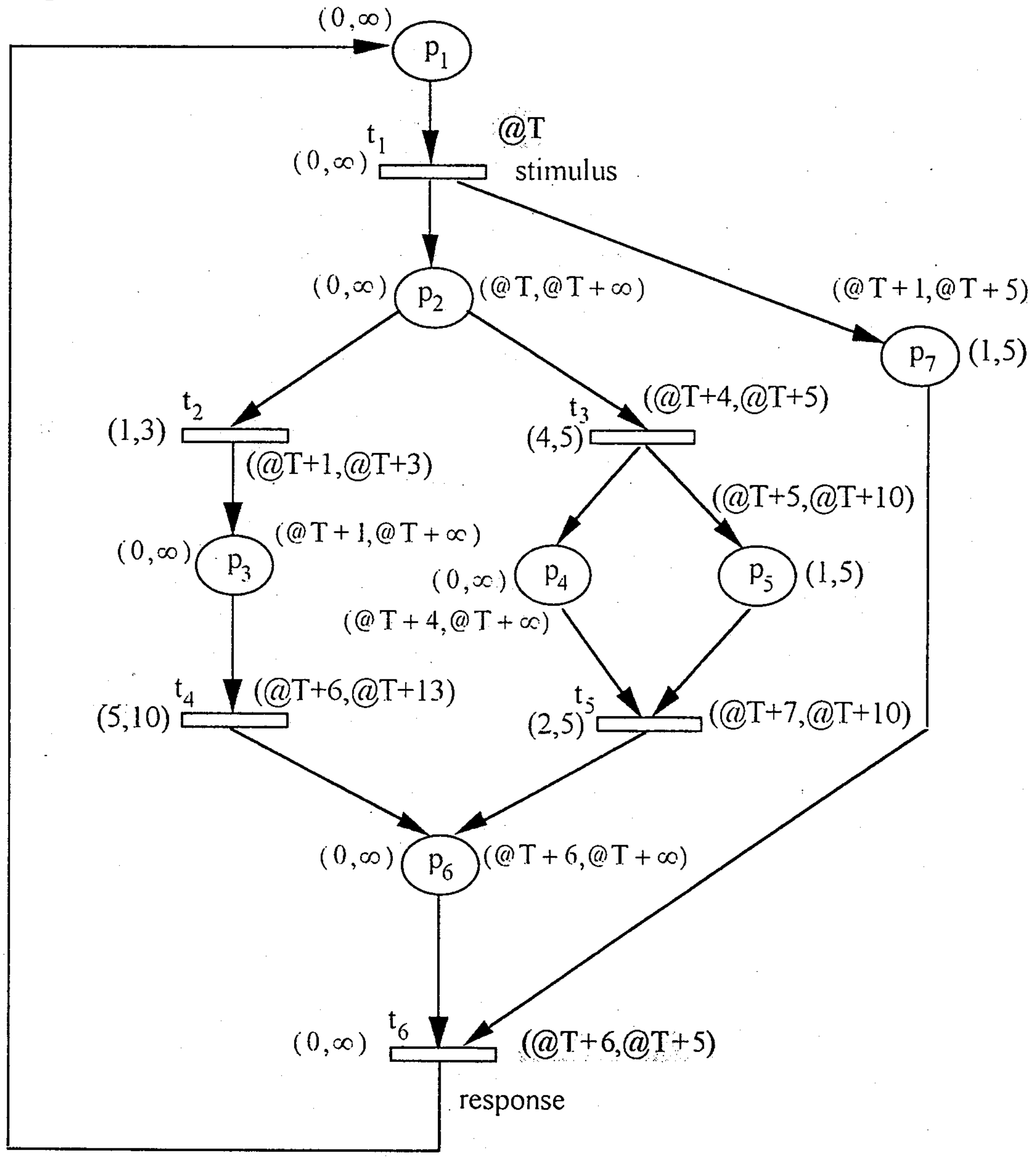

Figure 8. A real-time control system and the results of synthesis.

5. Conclusion and Future Research

In this paper, we have introduced the extension of the original Petri nets in terms of timing constraints. The extended transitions and places associated with timing constraints are called as timed transitions and time places, respectively, and the resulting Petri nets extension is called as timing constraint Petri nets. Three synthesis rules are presented in this paper to compute the time instants of each timed transition and timed place with given timing constraints. A net is said to satisfy the imposed timing constraints if and only if the timed transitions are fireable. The synthesis rules presented in this paper can be used not only in TCPNs, but also in other time-related extension of Petri nets. Of course, besides timing constraints, real-time systems have to satisfy other important properties, such as safety, liveness, fairness, and temporal properties such as nest, eventually, always. The approach we are currently working is to combine temporal logic along with the TCPNs.

References

[1] B. Berthomieu and M. Diaz, "Modeling and Verification of time Dependent Systems Using time Petri Nets," *IEEE Trans. on Software Eng.*, vol. SE-17, no. 3, pp. 259-273, March 1991.

[2] J.E. Coolahan, Jr. and N. Roussopoulos, "Timing Requirements for Time Driven Systems Using Augmented Petri Nets," *IEEE Trans. on Software Eng.*, vol. SE-9, no. 5, pp. 603-616, Sept. 1983.

[3] M. Felder, D. Mandrioli, and A. Morzenti, "Proving Properties of Real-Time Systems Through Logical Specifications and Petri Net Models," *IEEE Trans. on Software Eng.*, vol. SE-20, no. 2, pp. 127-141, Feb. 1994.

[4] C. Ghezzi, D. Mandrioli, S. Morasca, and M. Pezze, "A Unified High-Level Petri Net Formalism for Time-Critical Systems," *IEEE Trans. on Software Eng.* vol. SE-17, no. 3, pp. 160-171, Feb. 1991.

[5] M.A. Holliday and M.K. Vernon, "A Generalized Timed Petri Net Model for Performance Analysis," *IEEE Trans. on Software Eng.*, vol. SE-13, no. 12, pp. 1297-1310, Dec. 1987.

[6] N.G. Leveson and J.L. Stolzy, "Safety Analysis Using Petri Nets," *IEEE Trans. on Software Eng.*, vol. SE-13, no. 3, pp. 386-397, March 1987.

[7] N. Lopez-Benitez, "Dependability Modeling and Analysis of Distributed Programs," *IEEE Trans. on Software Eng.*, vol. SE-20, no. 5, pp. 345-352, May 1994.

[8] P.M. Merlin and D.J. Farber, "Recoverability of Communication Protocols Implications of a Theoretical Study," *IEEE Trans. on Communications*, vol. COM-24, no. 9, pp. 1036-1043, Sept. 1976.

[9] T. Murata, "Petri Nets: Properties, Analysis and Application," *Proceeding of IEEE*, vol. 77, no. 4, 1989, pp. 541-580.

[10] M. Notomi and T. Murata, "Hierarchical Reachability Graph of Bounded Petri Nets for Concurrent-Software Analysis," *IEEE Trans. on Software Eng.*, vol. SE-20, no. 5, pp. 325-336, May 1994.

[11] D. Peng and K.G. Shin, "Modeling of Concurrent Task Execution in a Distributed System for Real-time Control," *IEEE Trans. on Computers*, vol. C-36, no. 4, pp. 500-516, April 1987.

[12] C.V. Ramamoorthy and G.S. Ho, "Performance Evaluation of Asynchronous Concurrent Systems Using Petri Nets," *IEEE Trans. on Software Eng.*, vol. SE-6, no. 9, pp. 440-449, Sept. 1980.

[13] C. Ramchandani, "Analysis of Asynchronous Concurrent Systems by Petri Nets," Project MAC, TR-120, MIT, Cambridge, MA, Feb. 1974.

[14] R.R. Razouk and C.V. Phelps, "Performance Analysis Using time Petri Nets," in *Proc. 4th IFIP Protocol Specification, Testing and Verification*, eds. Y. Yemini et al., Amsterdam, The Netherlands, North-Horlland, 1985.

[15] I. Suzuki and H. Lu, "Temporal Petri Nets and Their Application to Modeling and Analysis of a Handshake Daisy Chain Arbiter," *IEEE Trans. on Computers*, vol. C-38, no. 5, May 1989, pp. 696-704.

[16] J.J.P.Tsai, S. J. Yang, Y. S. Chang, "Timing Constraints Petri Nets and Their Application to Schedulability Analysis of Real-Time System Specifications," *IEEE Transactions on Software. Engineering*, Vol. 21, No. 1, Jan. 1995, pp. 32-49.