2005

TIME CONDITION SYSTEMS

Prashanth Thumu
University of Kentucky

Recommended Citation

Thumu, Prashanth, "TIME CONDITION SYSTEMS" (2005). *University of Kentucky Master's Theses*. 374.
https://uknowledge.uky.edu/gradschool_theses/374

ABSTRACT OF THESIS

TIME CONDITION SYSTEMS

The current thesis considers the issue of state estimation of condition systems, a form of petri net with signal inputs and outputs. In previous research the problem of unobservability due to progress confusion was identified, in the presence of which state estimation is not possible. Here we introduce the notion of “Time Condition Systems", a class of condition systems that uses timing information from condition models to overcome state estimation problem caused by progress confusion. To make use of the timing information in the plant model, a procedure called “Exploded Time Plant" is synthesized. This procedure makes the plant model an observable model. It is proved that this procedure does not alter the structural and temporal behavior of the plant model and the plant maintains its integrity. The time plant(s) and the corresponding Exploded time plant(s) are subsequently used to develop observer(s) and controller(s) for Time condition models.

KEYWORDS: Time condition systems, Petri nets, State estimation, Exploded Time Plant, Timed observer, Controller.

Prashanth Thumu

June 5,2005

TIME CONDITION SYSTEMS

By

Prashanth Thumu

Dr. Lawrence Emory Holloway.

Director of Thesis.

Dr. Ibrahim Jawahir.

Director of Graduate Studies.

THESIS

Prashanth Thumu

The Graduate School
University of Kentucky
2005

TIME CONDITION SYSTEMS

THESIS

A thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science in the
College of Engineering
at the University of Kentucky

By
Prashanth Thumu

Lexington, Kentucky

Director: Dr. Lawrence E. Holloway,
Professor of Electrical & Computer Engineering,
University of Kentucky,

Lexington, Kentucky

2005

MASTER'S THESIS RELEASE

I authorize the University of Kentucky Libraries to reproduce this thesis in whole or in part for the purpose of research

Signed:________________________

Date:________________________

Acknowledgments

This Thesis is a significant scientific accomplishment in my life and it would be impossible without people who supported me and believed in me.

Most of all I would like to thank my Thesis chair and research advisor, Dr. Larry E.Holloway. He is not only a great professor with deep vision but also and most importantly a kind person. His trust and scientific excitement inspired me in the most important moments of making right decisions and I am glad to work with him.

I am also very thankful to my research colleague and good friend Praveen Mandavilli whose help, stimulating suggestions and encouragement helped me in all the time of research for and writing of this thesis.

I sincerely thank my committee members, Dr. Alan Male and Dr. Marwan Khraisheh for offering valuable insights and feedback that heled me improve my thesis.

Most importantly, I would like to express my deepest gratitude to my mother for her moral and emotional support in every phase of my life.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Overview

1.1 Introduction

Automatic control code synthesis for automated system is relatively easy and less time consuming than manually writing it. Also, automatic code synthesis minimizes or completely eliminates the need for debugging the control code, caused due to human errors.

Spectool is one such software package developed in University of Kentucky. This is a control code synthesis package that allows user to enter high-level specifications and then have them automatically converted into low level languages (like PLC ladder logic or assembly language.) used in actuating control signals and sensing responses. The modelling framework used for developing spectool is condition systems, a class of condition event model introduced by Sreenivas and Krogh [Krogh00]

In all the previous research of spectool project, controller synthesis, observability issues, state observer synthesis have been addressed by former researchers in an untimed world [Holl00], [Holl01], [Holl02], [Holl03], [Holl04]. The goal of this thesis is to create a framework that encompasses the notion of time in our condition systems, a class of discrete event systems. We use the timing information from our time condition models to resolve some state estimation problems in our models. The time domain to our condition systems enhances the expressive power of condition systems and uses the timing information to tackle the progress confusion problem , a state estimation problem discussed in [Holl01]

The thesis is divided into five chapters, in chapter 2 we begin discussing "Condition Systems" and then introduce the concept of "Time Condition Systems" for single layer condition models, a form of discrete event systems with input and output signals called conditions. A procedure is then developed for converting a nondeterministic plant model into a deterministic and observable plant model called exploded time plant, without losing structural and temporal integrity. A mapping function is then developed; this defines “correspondence” of places, transitions and timing between time plant and exploded time plant. At the end of chapter 2 we have a lemma and a detailed proof that demonstrates the structural and temporal integrity between time plant and exploded time plant. At the beginning of chapter 3, problem of “unobservability” is discussed, then the need for observer is identified and then the different approaches in the literature for for generating observers are briefly discussed. We then introduce the concept of timed observers and present requirements for creating a timed observer A timed observer is then automatically synthesized using the deterministic exploded time plant as input.A short description of the procedure is then presented followed by the proof observability of timed observer.

In chapter 4 we derive "Generated state label plant" from the exploded time plant. This plant follows all the properties required to develop a controller. A procedure is developed to build controller for the time plant using generated state label plant as input. The controller drives the time plant through a sequence of desired outputs. In chapter 5 an example is used to illustrate the results; The thesis is concluded by discussing different areas, where the time condition systems approach can be used. Potential areas for future research are also discussed.

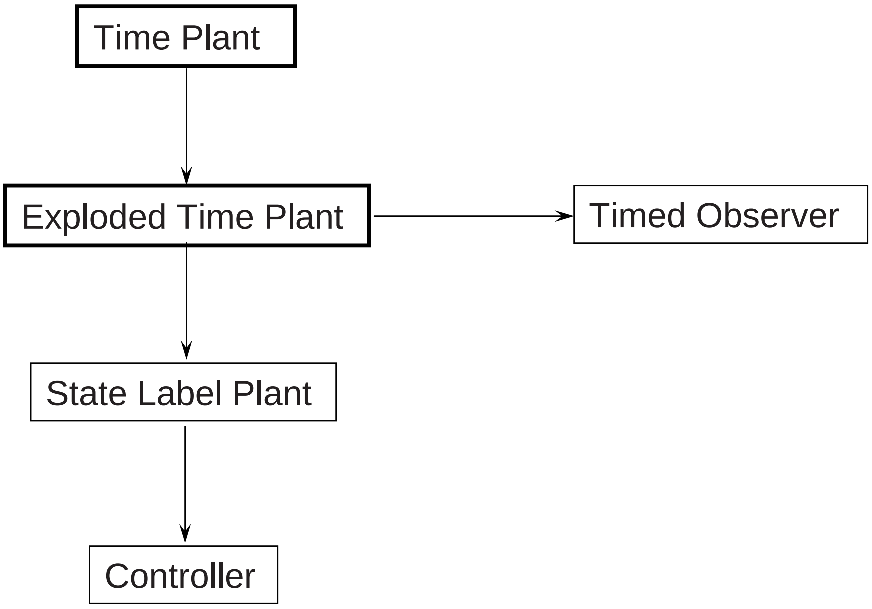

Figure 1.1: Stages In The Generation of Controller From Time Plant.

1.2 Motivation

All our prior research to automatically synthesize control for condition system models was done in absence of state estimation problem and in an untimed world. There are many real world situations like cost implications and complexity of having sensors in certain places/situations, where state estimation through signals using sensors might not be practical. Time condition systems deals the problem of state estimation due to inadequate sensing by using the timing information of the system to determine the state change. A simple example of one such system and its description is given below.

- ***Goal:***

  Make sure that the ball bearing is on the block by the time it reaches its final station.

- ***Description and Working:***

  Initially the block is in station 1. After the block reaches station 2, it stops and triggers the "motor on" signal for the ball bearing motor. The motor starts, and spits out the bearing form the ball bearing dispenser. The bearing falls on the ramp, rolls on it and finally falls on the block. Then the block moves on to the final station.

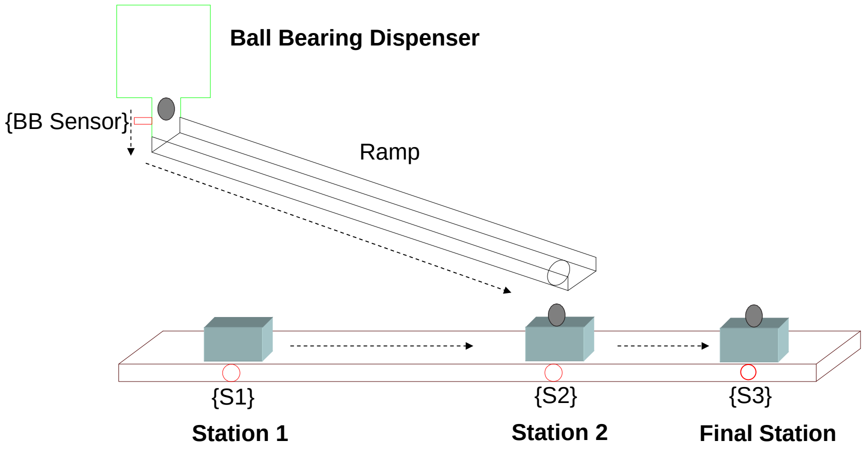

Figure 1.2: Example Model.

- ***Problem Description:***

  The position of the block at station 1, station 2 and at the final station is sensed by sensors S1, S2 and S3. The ball bearing sensor (BB sensor) confirms the falling of ball bearing on the ramp. But, there is no signal to indicate that the bearing actually dropped on the block at station 2 (inadequate sensing). Therefore there is no signal or indication that can trigger the movement of the block to the next station. Controlling such a model and guiding it to its target state is not possible with current condition systems.

- ***Proposed Solution:***

  Time condition systems formalism uses the timing information of the model to generate signals. In this example it uses the information pertaining to the length of the time it takes the bearing to roll on the ramp and drop on the block. It uses this information to generate a signal that triggers the movement of block from station 2 at the right time (block with bearing on it.)

Chapter 2

Time Condition Systems

2.1 Time Condition System Models

Condition systems interact with each other and with their outside environment through conditions. A condition can be considered as a signal that either has a true or a false value for a period of time. Condition systems were considered in [Holl00]. Conditions which are associated with places are true whenever that place is marked, and conditions which are associated with transitions act as guards which must be true in order for the transition to be enabled to fire. In condition systems state estimation is not possible due to subsequent states or, more than one possible next state corresponding to similar input/output observations. The reason for the introduction of time condition systems is to remove progress confusion from condition systems using timing information from the condition models. Time-condition systems are defined as a form of Petri net that requires conditions and timing information for enabling and firing transitions, and that outputs conditions according to its markings. Timing in our model is provided by timed transitions. “Merlin’s Time Petri Nets"[Merlin00], [Merlin01] with the addition of input/output signals, are used to model “Time-Condition Systems". Merlin’s TPN’s are used because of their demonstrated expressive power for temporal constraints in Petri Net formalism. The timing information on the transition is denoted by the interval of the format $\theta(t)$. We use the notation $\theta(t)|_{min}$ to denote the minimum time of the interval, representing the minimum time the transition should be enabled before which it qualifies to fire. The notation $\theta(t)|_{max}$ denotes the maximum time of the interval, representing the maximum time allowed by the transition to fire once it is enabled.

A Time-Condition system $G_{sys\theta}$ is characterized by a form of Petri net represented by the tuple $(\mathcal{P}_{G_{sys\theta}}, \mathcal{T}_{G_{sys\theta}}, \text{AllC}, \mathcal{A}_{G_{sys\theta}}, C_{G_{sys\theta}}(\cdot), \theta(\cdot))$, where $\mathcal{P}_{G_{sys\theta}}$ is a set of places, $\mathcal{T}_{G_{sys\theta}}$ is a set of transitions, AllC is a set of conditions, and $\mathcal{A}_{G_{sys\theta}}$ is a set of directed arcs connecting places to transitions and vice versa. The function $\theta(\cdot)$ maps enabling time intervals to transitions. The function $C_{G_{sys\theta}}(\cdot)$ is a mapping from transitions and places to conditions.

The state of the system over time τ is represented by the pair of functions $(m(\tau), \xi(\tau))$, where the MARKING $m(\tau)$ is a function mapping each place to non-negative integers, and the CLOCK FUNCTION $\xi(\tau)$ is a time function mapping each transition to non-negative real numbers. Conditions are signals that at any given time can have value TRUE or FALSE. For each condition $c \in \text{AllC}$, there exists a complement condition $\neg c \in \text{AllC}$, such that c has value true at a given time if and only if $\neg c$ has value false. A time-condition function is a function from time τ to subsets of conditions. We use the time condition function $TrueC(\tau)$ to represent the set of conditions which are true at a given time τ, where $\forall \tau$, $TrueC(\tau) \subseteq \text{AllC}$. Define the set of all time-condition functions as $\mathcal{L}$.

Given a time τ, let $m(\tau^-) = \lim_{\tau' \to \tau} m(\tau')$, $\xi(\tau^-) = \lim_{\tau' \to \tau} \xi(\tau')$, and
$TrueC(\tau^-) = \lim_{\tau' \to \tau} TrueC(\tau')$, respectively denote the marking, the timing vector, and the set of true conditions immediately before time τ. For a given place p, we use the notation $p^{(t)}$ to represent the set of transitions with arcs from p, and ${}^{(t)}p$ to represent the set of transitions with arcs leading to p. For a given transition t, the notation $t^{(p)}$ is the set of places with arcs from t, and ${}^{(p)}t$ is the set of places with arcs to t. For the function $C_{G_{sys\theta}}(\cdot)$, for a place p, we refer to $C_{G_{sys\theta}}(p)$ as the set of OUTPUT CONDITIONS for the place, and for a transition t, we refer to $C_{G_{sys\theta}}(t)$ as the set of enabling conditions for the transition.

The dynamics of a time-condition system are defined as follows:

1. ALLOWABLE CLOCK TIMES: For all τ, for all transitions $t \in \mathcal{T}_{G_{sys\theta}}$, $0 \leq \xi(\tau)(t) \leq \theta(t)|_{max}$.

2. CONDITIONS ASSOCIATED WITH MARKED PLACES HAVE VALUE OF TRUE: Given time τ, ($\forall p$ s.t. $m(\tau)(p) \geq 1$), $C_{G_{sys\theta}}(p) \subseteq TrueC(\tau)$.

3. Given $m(\tau^{-})$, $\xi(\tau^{-})$ and $TrueC(\tau^{-})$, the next state $(m(\tau), \xi(\tau))$ satisfies the following:

   (a) T is MARKING ENABLED, meaning $(\forall p \in \mathcal{P}_{G_{sys\theta}}\ m(\tau^{-})(p) \geq 1|\{t \in T \mid p \text{ is input to } t\}|$,

   (b) T is CONDITION ENABLED, meaning $(\forall t \in T)\ C_{G_{sys\theta}}(t) \subseteq TrueC(\tau^{-})$,

   (c) T is TIME ENABLED, meaning $(\forall t \in T)$, $\theta(t)|_{min} \leq \xi(\tau^{-}) \leq \theta(t)|_{max}$

   (d) Transitions at limit of time must fire: for any t such that $\xi(\tau^{-}(t)) = \theta(t)|_{max}$, then $t \in T$ must fire.

   (e) Each next possible marking satisfies the Petri Net firing rule: $\forall p \in \mathcal{P}_{G_{sys\theta}}, m(\tau)(p) = m(\tau')(p) - |\{t \in T \mid p \text{ is input to } t\,|$

   (f) Each fired transition has reset clock function: $(\forall t \in T)$, if transition t fires, then $\xi(\tau)(t) = 0$.

4. TRANSITIONS WHICH ARE MARKING ENABLED AND CONDITION ENABLED AND HAVE NOT FIRED HAVE CLOCK FUNCTION INCREASE WITH TIME:
   Given a time transition t, say t becomes marking and condition enabled at time τ' and is continuously enabled till time $\tau' + \Delta\tau'$ (such that $\theta(t)|_{min} \leq \Delta\tau' \leq \theta(t)|_{max}$) but does not fire at time τ', then:

   - $\xi(\tau' + \Delta\tau')(t) = \xi(\tau')(t) + \Delta\tau'$

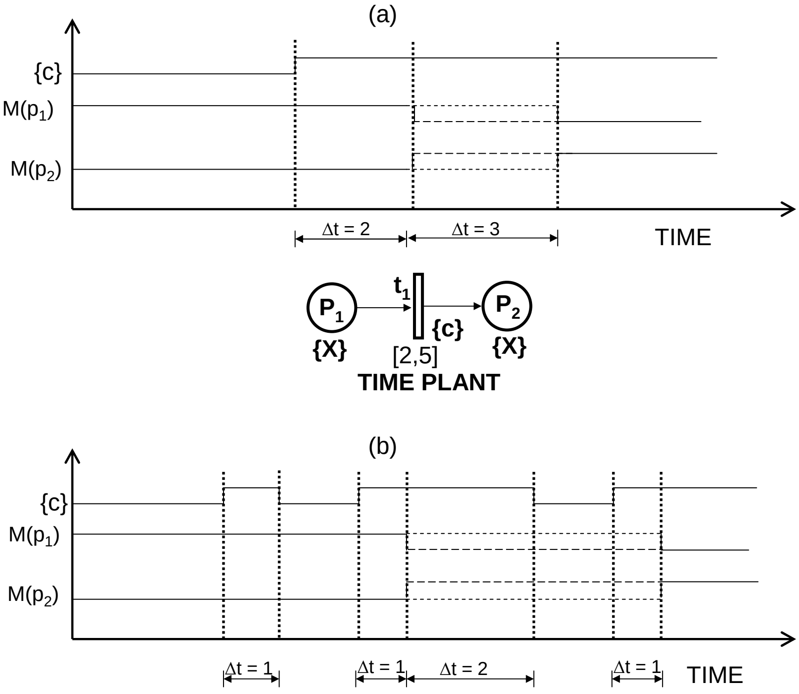

Figure 2.1: Examples of Time line diagrams for a Time Plant Model .

The cumulative time enabling of the time condition systems is illustrated in figure 2.1. In part (a) of the figure 2.1 it is shown that p_1 is marked for at least 2 time units only after condition {c} is true for a total of two time units. After condition {c} remains true for 2 time units, either p_1 can remain marked for a maximum period of 3 time units ({c} must be true during this time)or p_2 can become marked during this period; but not both. This phenomenon is indicated by small dashed lines and long dashed lines in fig.(a). This period of uncertainty is referred to as non-deterministic period. After the condition {c} is true for a cumulative of 2+3 = 5 time units, p_1 is no longer marked and p_2 becomes marked.

In part (b) of figure 2.1, condition {c} becomes true for one time unit. After that it becomes false for sometime and then it again becomes true for one more time unit. p_1 remains marked all this time but after {c} remains true for cumulative of 2 time units, p_2 *can* become marked. As discussed in part (a), only after {c} remains true for a cumulative time of 5 units, we can be certain that p_2 is marked.

Note: In either cases, if p_2 *becomes marked once, the marking cannot change back to* p_1 *at any future time.*

We say that time functions $(m(\cdot), \xi(\cdot), \mathrm{TrueC}(\cdot))$ are consistent with system $G_{sys\theta}$ if they satisfy the above for $G_{sys\theta}$. Given some initial state m_0, ξ_0 we define $\mathcal{L}^{Time}(G_{sys\theta}, (m_0, \xi_0))$ as the set of all time-condition functions $\mathrm{TrueC}(\cdot)$ such that there exists some time-marking function $m(\cdot)$ and clock function $\xi(\cdot)$ s.t. $m(0) = m_0, \quad \xi(0) = \xi_0$, and $(m(\cdot), \xi(\cdot), \mathrm{TrueC}(\cdot))$ are consistent. Timing information in many cases can be used to identify when a transition might fire. Due to the ranges of firing times that are possible within the model, it may be possible to determine the exact state only before a given time, and then after a given time. We thus can have periods of uncertainty in our state.An example time condition system is shown in figure 2.2 below. For example, if place p1 is marked, then the condition A will be true. The transition t_{p1} is *enabled* if the place p_1 is marked and condition x_1 is true. The timing interval of [2,4] on the transition indicates that the transition can only fire after at least two time units of being continuously enabled, and must fire within 4 time units of being continuously enabled.

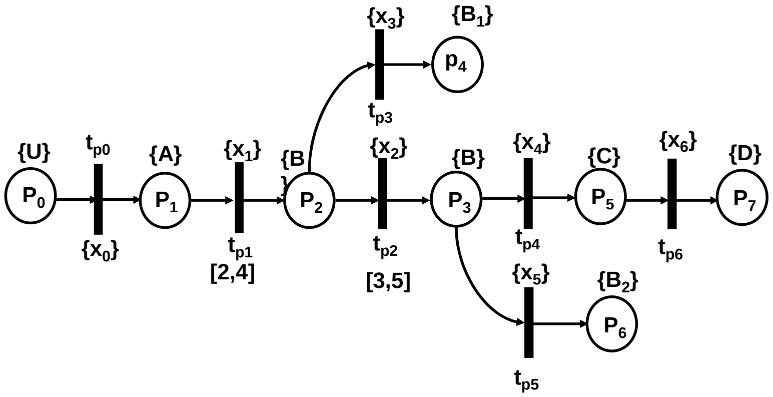

Figure 2.2: Time Plant.

Definition 2.1 **Progress Confusion**: A Time Condition system $G_{sys\theta}$ has progress confusion if there exists a transition t' such that the observed conditions on the input place p to the transition are the same as on output place p' to the transition: $C_{G_{sys\theta}}(p) = C_{G_{sys\theta}}(p')$.

For example P_2, P_3 in the Time Plant in figure 2.2 have progress confusion $C_{G_{sys\theta}}(P_2) = C_{G_{sys\theta}}(P_3)$

Definition 2.2 ***Direction Confusion*** A Time Condition System $G_{sys\theta}$ has direction confusion, if there exist two transitions t' and t'' such that each of the following is true:

- both transitions share the same input place.
- the observed conditions on output place are identical: $C_{G_{sys\theta}}(p')|_{C_{obsd}} = C_{G_{sys\theta}}(p'')|_{C_{obsd}}$ for $p' \in t'^{(p)}$ and $p'' \in t''^{(p)}$;
- the observed condition sets $C_{G_{sys\theta}}(t')|_{C_{obsd}}$ and $C_{G_{sys\theta}}(t'')|_{C_{obsd}}$ are not exclusive.

Definition 2.3 *A system model represented by a condition system is state observable under observed condition set C_{obsd} if: For times τ_0 and τ such that $\tau \geq \tau_0$, given any known initial marking m_{τ_0} and clock $\xi(\tau_0)$at time τ_0, and given the observed input and output conditions $[(C_{in}(G_{sys\theta}) \cup C_{out}(G_{sys\theta})) \cap C_{obsd}]$ over period τ_0 to τ, we can uniquely determine marking m_τ at time τ.*

Definition 2.4 ***Time plant model limitations and assumptions***

Following are the assumptions of the time condition system models for the rest of this paper:

- $G_{sys\theta}$ does not have direction confusion.
- No three consecutive places in the time plant model should have progress confusion.

- For each non-deterministic time transition t, (i.e. $\theta(t)|_{min} \neq \theta(t)|_{max}$ with progress confusion, $|t^{(p)}| = |^{(p)}t| = 1$

- For any places p_i, p_j in $G_{sys\theta}$ if p_i, p_j have progress confusion, then $(\forall p'_i \in (p_i^{(t)})^{(p)})$ and $(\forall p'_j \in (p_j^{(t)})^{(p)})$, $C_{G_{sys\theta}}(p'_i) \neq C_{G_{sys\theta}}(p'_j)$

2.2 Creating The Exploded Time Plant

In order to make these condition systems that we consider to be “State observable", we synthesize an “Exploded Time plant". In the resulting exploded time plant, all timing uncertainty is made deterministic. To do this, intermediate states are created which explicitly represent the state uncertainty. These intermediate states are similar to the “macro-states" of Giua in [Giua00], [Giua01]. Procedure to synthesize “Exploded Time Plant" from time plant model is shown in fig.2.3

Definition 2.5 Given a plant $G_{sys\theta}$, and given a place in it denoted by p, Define $\text{ALLSTATE} = \{State(p) \mid p \in G_{sys\theta}\} \subseteq \text{ALLC}$ as the set of such conditions. We refer to such conditions as State Labels.

Definition 2.6 **ProgConfPlaceSet(**$G_{sys\theta}$**)** is defined as the set, where all the places corresponding to progress confusion are stored. **ProgConfPlaceSet(**$G_{sys\theta}$**)** $=$ $\{p,\ p' | C_{G_{sys\theta}}(p) = C_{G_{sys\theta}}(p')\}$ where p is the parent place of p'.

Note: **ProgConfPlaceSet(**$G_{sys\theta}$**)** and ProgConfPlaceSet are interchangeable unless mentioned specifically.

Definition 2.7 **ProgConfTranSet(**$G_{sys\theta}$**)** is a set of transitions in the timed plant $G_{sys\theta}$, whose input and output places have same conditions. **ProgConfTranSet(**$G_{sys\theta}$**)**$= \{t | \exists p \in^{(p)} t, p' \in t^{(p)} \text{ s.t. } C_{G_{sys\theta}}(p) = C_{G_{sys\theta}}(p')\}$

Note: **ProgConfTranSet(**$G_{sys\theta}$**)** and ProgConfTranSet are interchangeable unless mentioned specifically.

Definition 2.8 Non-Deterministic Place: If a place has more than one State label condition as its output condition, it is said to be a Non-Deterministic Place.

Multiple state label conditions indicate the non-determinism of the plant, i.e, we don't know if the token is in (P_i) or (P_{i+1}) but we are sure that it is either in (P_i) or in (P_{i+1}).

Definition 2.9 We define **NDset** as a set of non-deterministic places in $G_{\mathrm{Exp}\theta}$, i.e, it is the set of all the places in exploded time plant with more than one state label condition. **NDset** $= \{p \,|p \in P_{G_{Exp\theta}},\ |C_{G_{Exp\theta}}(p) \cap AllState| > 1\}$

Definition 2.10 AllState is defined as a set of all state label conditions in the exploded time plant.

Figure 2.3: Exploded Time Plant procedure

```
Initially define $G_{\mathrm{Exp}\theta}$ s.t $P_{G_{Exp\theta}}, T_{G_{Exp\theta}}, A_{G_{Exp\theta}}$
duplicates the timed plant $P_{G_{sys\theta}}, T_{G_{sys\theta}}, A_{G_{sys\theta}}$
Let $p_{\mathrm{Exp}\theta} \in P_{G_{Exp\theta}}$ and $t_{\mathrm{Exp}\theta} \in T_{G_{Exp\theta}}$ indicate the
corresponding place and transition in $G_{\mathrm{Exp}\theta}$ for
each $p \in P_{G_{sys\theta}}$ and $t \in T_{G_{sys\theta}}$
**For** each $p \in P_{G_{sys\theta}}$ **{**
    $C_{G_{Exp\theta}}(p_{\mathrm{Exp}\theta}) \Leftarrow \{State(p)\} \bigcup C_{G_{sys\theta}}(p)$ **}**
**For** each $t \in T_{G_{sys\theta}}$ **{**
    **If** $t \notin ProgConfTranSet(G_{sys\theta})$ {
        $C_{G_{Exp\theta}}(t_{\mathrm{Exp}\theta}) \Leftarrow C_{G_{sys\theta}}(t)$
        $\theta(t_{\mathrm{Exp}\theta}) \Leftarrow \{\theta(t)\}$    }**}**
**For** each $t \in ProgConfTranSet(G_{sys\theta})${
            EXPAND-TIME$(t)$;}
```

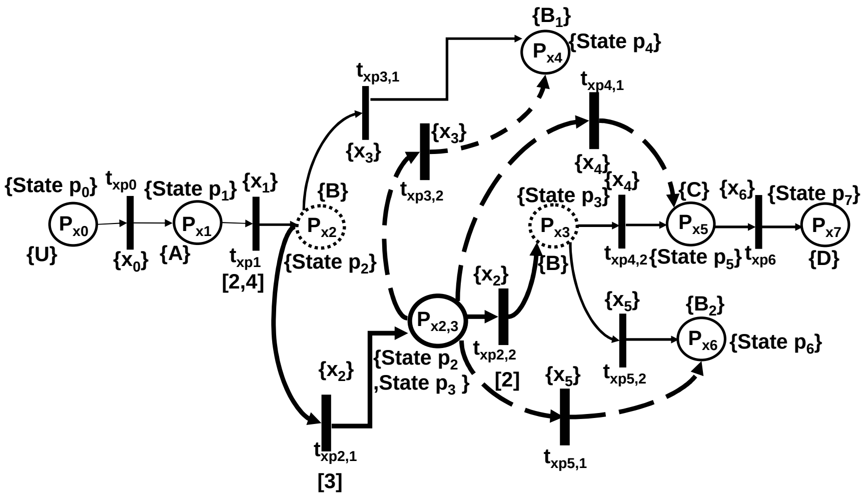

Figure 2.4: Exploded Time Plant.

Exploded Time plant (ExTP) Description:

1. **Replicating time plant structure:** Original plant structure is reproduced. For every place p and transition t in the plant, there will be a corresponding place $p_{\text{Exp}\theta}$ and transition $t_{\text{Exp}\theta}$ in the ExTP. Each place $p_{\text{Exp}\theta}$ in the ExTP outputs condition $C_{G_{sys\theta}}(p)$ corresponding to place p in the timed plant. By line 4, It also outputs a unique state label “State(p)" assigned to it.

2. **Progress confusion check:** Lines 5-10 check for progress confusion in the timed plant by comparing conditions on input $({}^{(p)}t)$ and output places $(t^{(p)})$ of a transition ‘t’. All the transitions associated with progress confusion are stored in the set “ProgConfTranSet($G_{sys\theta}$)". For each of the remaining transitions ‘t’ in the time plant there already exists a corresponding transition $t_{\text{Exp}\theta}$ in ExTP; Conditions and time are assigned to these transitions in lines 7,8. Each of the transitions in “ProgConfTranSet($G_{sys\theta}$)" is passed on as a parameter to the function Expand-time(t)

The function Expand-time(t) performs the following operations

Figure 2.5: Expand-time(t) Procedure

```
**Procedure** Expand-time(t)
For this transition t in $G_{Exp\theta}$ let
    $t_{Exp\theta}$ be the corresponding transition in $G_{Exp\theta}$
**Let** $P_{input} = \{p|p \in {}^{(p)}t_{Exp\theta}\}$
    and $P_{output} = \{p|p \in t_{Exp\theta}^{(p)}\}$
**DELETE** arcs from $p \in {}^{(p)}t_{Exp\theta}$ to $t_{Exp\theta}$ and
    from $t_{Exp\theta}$ to $t_{Exp\theta}^{(p)}$
**Create** transitions $t_L$, $t_{U-L}$, and place $p_\theta$
$C_{G_{Exp\theta}}(t_L) \Leftarrow C_{G_{sys\theta}}(t)$
$\theta(t_L) \Leftarrow [\theta(t)|_{min}]$
$C_{G_{Exp\theta}}(t_{U-L}) \Leftarrow C_{G_{sys\theta}}(t)$
$\theta(t_{U-L}) \Leftarrow [\theta(t)|_{max} - \theta(t)|_{min}]$
$C_{G_{Exp\theta}}(p_\theta) \Leftarrow C_{G_{Exp\theta}}(P_{input}) \bigcup C_{G_{Exp\theta}}(P_{output})$
**CREATE** arcs from all $p \in P_{input}$ to $t_L$
    and from $t_{U-L}$ to all $p \in P_{output}$
**CREATE** arcs from $t_L$ to $p_\theta$ and from $p_\theta$ to $t_{U-L}$
    **For** each $p \in P_{input}$ **{**
        **FOR** each $t \in p^{(t)}/t_L$ {
            Create transition $t_{dup}$ with $C_{G_{sys\theta}}(t_{dup}) = C_{G_{sys\theta}}(t)$
            Create arcs from $p_\theta$ to $t_{dup}$
            Create arcs from $t_{dup}$ to each $p \in t^p$ } **}**
    **For** each $p \in P_{output}$ **{**
        **FOR** each $t \in p^{(t)}$ {
            Create transition $t_{dup}$ s.t.$C_{G_{sys\theta}}(t_{dup})=C_{G_{sys\theta}}(t)$
            Create arcs from $p_\theta$ to $t_{U-L}$
            Create arcs from $t_{dup}$ to each $p \in t^p$ } **}**
Delete $t_{Exp\theta}$
```

- **Extracting, isolating and exploding the timed transition(1-4):** All the input and output place(s) of the transition are extracted in the sets P_{input} and P_{output} respectively. After this the transition is isolated by deleting all its input and output arcs. This isolated timed transition $t_{Exp\theta}$ is now exploded , i.e, replaced by t_L p_θ t_{U-L}. t_L and t_{U-L} are the replica of timed transition $t_{Exp\theta}$; they have the same conditions assigned to them as $t_{Exp\theta}$ The only difference is the timing associated with these transitions.As discussed earlier t_L represents the transition with lower limit of time $\theta(t)|_{min}$ on it. t_{U-L} represents the the transition with deterministic time $\theta(t)|_{max} - \theta(t)|_{min}$ on it. p_θ is the ND place between these two transitions.

- **Assigning conditions and times to the generated places and transitions:** Lines 5, 7 assign the conditions to transitions t_L and t_{U-L}respectively. Timing on these transitions are provided by lines 6,8 in the algorithm. p_θ is a Non-deterministic place that outputs all the conditions of both the input and output place(s)of the current transition $t_{Exp\theta}$. Input arcs are drawn to t_L from all place(s) that were input to $t_{Exp\theta}$ (P_{input}) Similarly output arcs are drawn from t_{U-L} to all place(s)that were output to $t_{Exp\theta}$ (P_{output}).

- **Non-deterministic Arcs (Lines 12-21)** Except for t_L, all the transitions output from places in $P_{inputset}$ are duplicated. All these duplicated transitions have input arcs from p_θ and output arcs to corresponding $p \in t^p$. Similarly all the transitions output from places in $P_{outputset}$ are duplicated and have input arcs from p_θ and output arcs to corresponding $p \in t^p$.

Definition 2.11 Given time plant $G_{sys\theta}$ and its corresponding exploded time plant $G_{Exp\theta}$. Let $P_{sys\theta} = \{p_0, p_1, p_2 ... p_n\}$ be the set of places in $G_{Exp\theta}$. Define $P_{Exp\theta}$ as the set of places in corresponding exploded time plant, i.e., $P_{Exp\theta} = \mathbf{\{}P_{Orig}, P_{ND}\mathbf{\}}$, where

- $P_{Orig} = \{p_{xi} | 0 \leq i \leq n\}$

- $P_{ND} = \{p_{xi,j} \,|\, p_i, p_j \in P_{sys\theta}$ s.t., $p_i^{(t)} \cap^{(t)} p_j \in \{ProgConfTranSet\}\}$ s.t $0 \leq i \leq n$, $0 \leq j \leq n$, where p_{xi} indicates a deterministic place and $p_{xi,j}$ a non-deterministic place.

  Note: $p_{xi,j}$ *is the non-deterministic place in* $G_{Exp\theta}$ *that is attributed to progress confusion between places* p_i, p_j *in time plant.*

Let $T_{sys\theta} = \{t_0, t_1, t_2, ...t_n\}$, Define $T_{Exp\theta} \in G_{Exp\theta}$ as the set of transitions in exploded time plant; $T_{Exp\theta} = \{T_{Orig} \cup T_{Prog} \cup T_{Dup}\}$, where

- $T_{Orig} = \{t_{xi} | 0 \leq i \leq n\}$, where n = number of transitions in $T_{sys\theta}$

  1. $\theta(t_{xi}) := \theta(t_i)$

- $T_{Prog} = \{t_{xi,1}, t_{xi,2} \,|\, 0 \leq i \leq n \; \forall t_i \in \{ProgConfTranSet\}\}$

  1. $\theta(t_{xi,1}) := \theta(t_i)|_{min}$
  2. $\theta(t_{xi,2}) := \theta(t_i)|_{max} - \theta(t_i)|_{min}$

- $T_{Dup} = \boldsymbol{\{}\bar{t}_{xi} \,|\, t_i \in T\}$ where, $T \in [^{(p)}t_j]^{(t)} \cup [t_j^{(p)}]^{(t)} - \{t_j\}\, and\, t_j \in \{ProgConfTranSet\}\}$

We define mapping function(s) that give correspondence relationship between place(s), transition(s) in time plant and place(s), transition(s) in exploded time plant.

There are two types of mapping functions:

- Pre-Mapping function := $^{\leftrightarrow}()$
- Post-Mapping function := $()^{\leftrightarrow}$

1.

$$^{\leftrightarrow}(p_i) = \begin{cases} p_{xi} & \text{if } \{^{(t)}p_i\} \cap \{ProgConfTranSet\} = \phi \\ p_{xj,i} & \text{for j s.t. } \{(p_j)^{(t)} \cap^{(t)} p_i\} \cap \{ProgConfTranSet\} \neq \phi \end{cases}$$

2.

$$(p_i)^{\leftrightarrow} = \begin{cases} p_{xi,j} & \text{for j s.t. } \{(p_i)^{(t)} \cap^{(t)} p_j\} \cap \{ProgConfTranSet\} \neq \phi \\ p_{xi} & \text{for j s.t. } \{(p_j)^{(t)} \cap^{(t)} p_i\} \cap \{ProgConfTranSet\} = \phi \end{cases}$$

Before defining Pre-Mapping and Post-Mapping functions for the transitions we define the following conditions

A $:= t_i \in \{ProgConfTranSet\}$

B $:= \exists$ at least one $t \in (t_i)_{sib}$ such that $t \in \{ProgConfTranSet\}$

C $:= {}^{(p)}t_i \in \{ProgConfPlaceSet\}$

The above conditions are key to determine the correspondence mapping between the time plant and the exploded time plant.

Note 1: Both A, B cannot be true at the same time due to structural limitations mentioned in "TIME PLANT MODEL LIMITATIONS."

Note 2: Condition A true implies condition C is true

1.

$$^{\leftrightarrow}(t_i) = \begin{cases} t_{xi} & \text{if conditions } \bar{A}BC,\ \bar{A}\bar{B}\bar{C} \text{ are satisfied} \\ t_{xi,1} & \text{if conditions } A\bar{B}C \text{ is satisfied} \\ \bar{t}_{xi} & \text{if conditions } \bar{A}\bar{B}C \text{ is satisfied} \end{cases}$$

2.

$$(t_i)^{\leftrightarrow} = \begin{cases} t_{xi} & \text{if conditions } \bar{A}\bar{B}C,\ \bar{A}\bar{B}\bar{C} \text{ are satisfied} \\ t_{xi,2} & \text{if conditions } A\bar{B}C \text{ is satisfied} \\ \bar{t}_{xi} & \text{if conditions } \bar{A}BC \text{ is satisfied} \end{cases}$$

The conditions ABC, $AB\bar{C}$, $A\bar{B}\bar{C}$, $\bar{A}B\bar{C}$ are not structurally possible

Lemma 2.1 Given a timed component system $G_{sys\theta}$ with progress confusion. Let $G_{Exp\theta}$ be Exploded time plant created by the algorithm in figure 2.3, 2.5 Following are the properties of exploded time plant $G_{Exp\theta}$

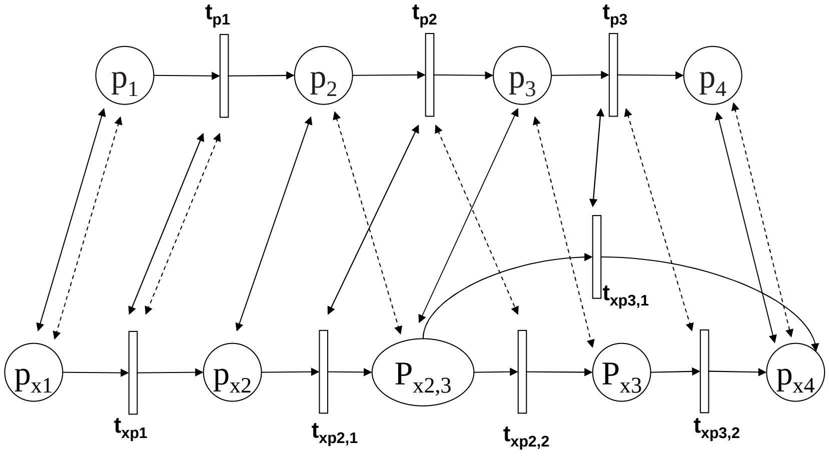

Figure 2.6: Example of Mapping.

- **State labels :** For each place p_{xi} in $G_{Exp\theta}$, $|\,C_{G_{Exp\theta}}(p_{xi}) \cap \textsc{AllState}\,| \geq 1$

- **Matching of Output conditions :** For each place p_i in $G_{sys\theta}$, there exists exactly one place p_{xi} in $G_{Exp\theta}$ such that $C_{G_{Exp\theta}}(p_{xi}) \cap \textsc{AllState} = \{State(p_i)\}$. Furthermore, $C_{G_{Exp\theta}}(p_{xi}) \cap$ ***AllC-AllState*** $= C_{G_{sys\theta}}(p_i)$.

- **Structural integrity :**

  FORWARD CONNECTIVITY: Since the structure of Exploded time plant $G_{Exp\theta}$ is duplicated from the timed plant $G_{sys\theta}$, For any p_i, p_j in time plant $G_{sys\theta}$, there exists corresponding places p_{xi}, p_{xj} in the exploded time plant $G_{Exp\theta}$ such that $C_{G_{sys\theta}}(p_i) = C_{G_{Exp\theta}}(p_{xi})\ |_{\textsc{AllC-AllState}}$ and $C_{G_{sys\theta}}(p_j) = C_{G_{Exp\theta}}(p_{xj})\ |_{\textsc{AllC-AllState}}$

  $|_{AllC-AllState}$ is the projector operator[1]

  For any $t_i \notin$ProgConfTranSet($G_{sys\theta}$),Such that $p_i \in {}^{(p)}t_i$ and $p_j \in t_i^{(p)}$, there exists a corresponding transition $t_{xi} \in G_{Exp\theta}$ and $p_{xi}, p_{xj} \in G_{Exp\theta}$ such that $t_{xi} \in p_{xi}^{(t)} \cap {}^{(t)}p_{xj}$ and $C_{G_{Exp\theta}}(t_{xi}) = C_{G_{sys\theta}}(t_i)$

  For any transition $t_i \in$ProgConfTranSet($G_{sys\theta}$), let $p_i \in {}^{(p)}t$ and $p_j \in t^{(p)}$ de-

[1] $|_{entry}$ is the projector operator that filters away the conditions that do not belong to its subscript argument from the condition set to which it is applied. It is also called masking operator.

note I/P and O/P places and let p_{xi}, p_{xj} represent corresponding places of p_i and p_j in $G_{Exp\theta}$ (satisfying statement 2 of lemma 2.1). Then, there exists a place $p_{xi,j} \in NDset$ and transitions $t_{xi,1}$, $t_{xi,2}$ in $G_{Exp\theta}$ such that

1. $C_{G_{Exp\theta}}(p_{xi}) \mid_{AllC\text{-}AllState} = C_{G_{Exp\theta}}(p_{xj}) \mid_{AllC\text{-}AllState} = C_{G_{Exp\theta}}(p_{xi,j}) \mid_{AllC\text{-}AllState} = C_{G_{sys\theta}}(p_i) = C_{G_{sys\theta}}(p_j)$
2. $t_{xi,1} \in p_{xi}^{(t)} \cap {}^{(t)}p_{xi,j}$
3. $t_{xi,2} \in p_{xi,j}^{(t)} \cap {}^{(t)}p_{xj}$
4. $C_{G_{Exp\theta}}(t_{xi,1}) = C_{G_{Exp\theta}}(t_{xi,2}) = C_{G_{sys\theta}}(t_i)$.

All the transitions output from $p_{xi,j}$ are replica of transitions output from p_{xi}, p_{xj} in $G_{Exp\theta}$ corresponding to p_i, p_j in $G_{sys\theta}$. All these replicated transitions have same output place(s) as p_{xi} and p_{xj} in $G_{Exp\theta}$ corresponding to p_i and p_j in $G_{sys\theta}$.

Reverse Connectivity: For any places p_{xi}, $p_{xj} \in \{G_{Exp\theta} - NDset\}$, let t_{xi} be a transition such that $t_{xi} \in p_{xi}^{(t)} \cap^{(t)} p_{xj}$; Then, there exists a transition t_i with corresponding places p_i, p_j in $G_{sys\theta}$ satisfying statement 2 in lemma 2.1 such that $t \in p_i^{(t)} \cap^{(t)} p_j$ and $C_{G_{sys\theta}}(t_i) = C_{G_{Exp\theta}}(t_{xi})$

For each $p_{xi,j}$ in $G_{Exp\theta}$ with $t_{xi,1}$ and $t_{xi,2}$ as I/P and O/P transitions ($t_{xi} =^{(t)} p_{xi,j}$, $t_{xi,2} = p_{xi,j}^{(t)}$) and with $p_{xi} =^{(p)} t_{x,1}$; $p_{xj} =^{(p)} t_{xi,2}$, there exists t in $G_{sys\theta}$ with p_i as it's input and p_j as its output place such that

1. $C_{G_{Exp\theta}}(t_{xi,1}) = C_{G_{Exp\theta}}(t_{xi,2}) = C_{G_{sys\theta}}(t_i)$
2. $C_{G_{Exp\theta}}(p_{xi,j}) \mid_{AllC\text{-}AllState} = C_{G_{sys\theta}}(p_i) = C_{G_{sys\theta}}(p_j)$

Lemma 2.2 Consider a time plant $G_{sys\theta}$ satisfying "Timed plant model limitations" and with uniquely identifiable initial state. Let $G_{Exp\theta}$ be its corresponding exploded time plant with matching structure and conditions, At initial time τ_0, if

$TrueC_{sys\theta}(\tau_0)|_{AllC-AllState} = TrueC_{Exp\theta}(\tau_0)|_{AllC-AllState}$,for any future time such that non-state conditions have continued to map since the beginning, then at any future time $\tau \geq \tau_0$ such that $TrueC_{sys\theta}(\tau')|_{AllC-AllState} = TrueC_{Exp\theta}(\tau')|_{AllC-AllState}$ for $\tau_0 \geq \tau' \geq \tau$, if p_i is marked in original time plant, then either ${}^{\leftrightarrow}(p_i)$ $P_{Exp\theta}$ or $(p_i)^{\leftrightarrow} \in P_{Exp\theta}$ is marked. i.e. the states of the systems map.

Proof: The state of the system is defined by marking and the timing associated to it. Say at initial time$\tau = \tau_0$, a place $p_{i(i=0)} \in G_{sys\theta}$ is marked. i.e., $m_{sys\theta}(\tau_0)(p_i) = 1.$; i=0. Since $G_{sys\theta}$ satisfies the "Time plant model limitations", there will be no progress and direction confusion from initially marked place "p_0" in $G_{sys\theta}$. From lemma statement, there exists a unique corresponding place $p_{xi} \in G_{Exp\theta}$ for initially marked place $p_i \in G_{sys\theta}$ (i=0) and from lemma 2.1 $C_{G_{sys\theta}}(p_i)|_{AllC-AllState}$ =$C_{G_{Exp\theta}}(p_{xi})|_{AllC-AllState}$.
$\Rightarrow m_{Exp\theta}(\tau_0)(p_{x0}) = 1.$
hence, the initial condition $TrueC_{sys\theta}(\tau_0)|_{AllC-AllState} = TrueC_{Exp\theta}(\tau_0)|_{AllC-AllState}$ has been established.

Now, say at time $\tau = \tau_0 + \Delta\tau$ an untimed transition $t_j \in p_i^{(t)}$ fires in time plant(i=0, j=0). For t_0 to fire it should be both state and condition enabled i.e. for i=0, j=0

$$C_{G_{sys\theta}}(p_i)|_{AllC-AllState}, C_{G_{sys\theta}}(t_j) \in TrueC_{sys\theta}(\tau_0 + \Delta\tau)|_{AllC-AllState} \tag{2.1}$$

From structural integrity statement in Lemma 2.1

$$C_{G_{sys\theta}}(t_j) = C_{G_{Exp\theta}}(t_{xj}) \tag{2.2}$$

and from initial condition, if p_i is marked so is p_{xi}

$$\Rightarrow C_{G_{sys\theta}}(p_i)|_{AllC-AllState} = C_{G_{Exp\theta}}(p_{xi})|_{AllC-AllState} \tag{2.3}$$

$\therefore$ at time $\tau_0 + \Delta\tau$

$$C_{G_{Exp\theta}}(p_{xi})|_{AllC-AllState}, C_{G_{Exp\theta}}(t_{xj}) \in TrueC_{Exp\theta}(\tau_0 + \Delta\tau)|_{AllC-AllState} \tag{2.4}$$

From equations 2.1, 2.2, 2.3, 2.4 $TrueC_{sys\theta}(\tau_0 + \Delta\tau)|_{AllC-AllState} = TrueC_{Exp\theta}(\tau_0 + \Delta\tau)|_{AllC-AllState}$. So if the transition t_0 fires in $G_{sys\theta}$, the transition t_{x0} in $G_{Exp\theta}$ would also fire, hence the places $p_1 = t_0^{(p)} \in G_{sys\theta}$ and corresponding place $p_{x1} = t_{x0}^{(p)} \in G_{Exp\theta}$ become marked at the same time instant τ' s.t. $(\tau_0 + \Delta\tau) < \tau' < \tau$. Hence, $m_{sys\theta}(\tau')(p_1){=}1$, $m_{Exp\theta}(\tau')(p_{x1}){=}1$ and

$$Truec_{sys\theta}(\tau')|_{AllC-AllState} = TrueC_{Exp\theta}(\tau')|_{AllC-AllState}$$

Case 1: Say the newly marked place $p_1 \notin ProgConfPlaceSet$ and let t_1 be one of its output transitions, and let p_2 be its output transitions, and let p_2 be its output place i.e. $t_1 \in p_1^{(t)}$ and $p_2 \in t_1^{(p)}$. Since $p_1 \notin ProgConfPlaceSet$, for p_1,$t_1 \in G_{sys\theta}$, there exists a unique corresponding place p_{xi} and transition t_{xi} in exploded time plant $G_{Exp\theta}$. Equations 2.1,2.2, 2.3, 2.4 hold true for i=1, k=1 at time $\tau = \tau'$. The transition t_1 in the time plant is state enabled at time $\tau = \tau'$ Now say at time $\tau = \tau' + \Delta\tau$, t_1 becomes both state and condition enabled and fires marking new place p_2 in the time plant; the corresponding transition t_{x1} in exploded time plant $G_{Exp\theta}$ is also state and condition enabled (equations 2.2, 2.3)and fires and p_{x2} becomes marked. Hence, at time $\tau = (\tau' + \Delta\tau)|_{AllC-AllState}$

$$TrueC_{sys\theta}(\tau' + \Delta\tau)|_{AllC-AllState} = TrueC_{Exp\theta}(\tau' + \Delta\tau)|_{AllC-AllState}$$

Case 2: Say $p_1 \in \{ProgConfPlaceSet\}$ and let t_1 be one of its output transitions such that $t_1 \in \{ProgConfTranSet\}$ and $\theta(t_1) = [\tau_{min}, \tau_{max}]$ be the time interval on t_1 and let p_2 be output place of t_1 i.e. $t_1 \in p_1^{(t)}$, $p_2 = t_1^{(p)}$

For $p_1, p_2 \in \{ProgConfPlaceSet\}$ there each exists two corresponding places (Pre-mapped, Post-mapped) p_{x1}, $p_{x1,2}$ for p_1 and $p_{x1,2}$, p_{x2} for p_2 in exploded time plant. Similarly for $t_1 \in \{ProgConfTranSet\}$ in the time plant, there exists corresponding (Pre-mapped, Post-mapped) transition $t_{1,1}$, $t_{1,2}$ in exploded time plant with time intervals $[\theta(t_1)|_{min}]$ and $[\theta(t_1)|_{max}] - [\theta(t_1)|_{min}]$ respectively. Following cases illustrate all the possible dynamics of the time plant and the corresponding exploded time plant when it is in progress confusion state:

1. At time $\tau = \tau' + [\theta(t_1)|_{min}]$ equation 2.2 holds true for t_1, $t_{x1,1}$ and equation 2.3 holds true for p_1, p_{x1}.

   (a) Now say at time $\tau = \tau' + \tau_{min}$ transition t_1 fires and p_2 gets marked. Following the algorithm in fig. 2.3, fig. 2.5, the transition $t_{x1,1}$ in exploded time plant is forced to fire at deterministic time $\tau = \tau' + [\theta(t_1)|_{min}]$, thus marking the place $p_{x1,2}$ in exploded time plant. Hence at time $\tau = \tau' + [\theta(t_1)|_{min}]$, places p_2 in the time plant and the corresponding pre-mapped place $p_{x1,2}$ in exploded time plant are marked. Places p_2, $p_{x1,2}$ satisfy equation 2.3 at time $\tau = \tau' + [\theta(t_1)|_{min}]$ and also for any transition that belongs to $\{p_2^{(t)}\}$ in time plant, there exists a corresponding transition that belongs to $\{p_{x1,2}\}$ in exploded time plant those transitions satisfy equation 2.2.

   (b) Say at time $\tau = \tau' + \tau_{min}$, transition t_1 doesn't fire in the time plant and p_1 is still marked. According to the algorithm of the exploded time plant, the corresponding transition $t_{x1,1}$ in the exploded time plant is forced to fire after it is time enabled in $G_{Exp\theta}$ for time (deterministic) $\tau = \tau' + [\theta(t_1)|_{min}]$. Therefore after time $\tau = \tau' + [\theta(t_1)|_{min}]$, place $p_{x1,2}$ in the time plant is marked. Hence at time $\tau = \tau' + [\theta(t_1)|_{min}]$, places p_1 in the time plant and the corresponding post-mapped place $p_{x1,2}$ in exploded time plant are marked. Places p_1, $p_{x1,2}$ satisfy equation 2.3 at time $\tau = \tau' + [\theta(t_1)|_{min}]$ and also for any transition that belongs to $\{p_1^{(t)}\}$ in time plant, there exists a corresponding transition that belongs to $\{p_{x1,2}^{(t)}\}$ in exploded time plant and those transitions satisfy equation 2.2.

   Therefore at time $\tau = \tau' + [\theta(t_1)|_{min}]$

   $TrueC_{sys\theta}(\tau' + [\theta(t_1)|_{min}])|_{AllC-AllState} = TrueC_{Exp\theta}(\tau' + [\theta(t_1)|_{min}]$ $)|_{AllC-AllState}$.

   Note: $\tau_{min} = [\theta(t_1)|_{min}]$.

2. Now say transition t_1 fires at time $\tau = \tau' + \tau_{max}$ and p_2 becomes marked,

$$(\text{Here } \tau_{max} \geq \tau_{min} \ \& \ \tau_{max} = \lceil \theta(t_1)|_{max} \rceil)$$

$\tau = \tau' + \lceil \theta(t_1)|_{max} \rceil$ can also be written as $\tau = (\tau' + \lceil \theta(t_1)|_{min} \rceil) + (\theta(t_1)|_{max} - \theta(t_1)|_{min})$. It was already shown in (b) that at time $\tau = \tau' + \tau_{min}$, $t_{x1,1}$ fires in the exploded time plant and the post-mapped place of p_1, i.e. $p_{x1,2}$ becomes marked.

At time $\tau = \tau' + \tau_{min}$, p_1 in the time plant and corresponding post mapped place $p_{x1,2}$ in exploded time plant are marked. At the same time transition t_1 in the time plant and its corresponding post-mapped transition $t_{x1,2} \in \{p_{x1,2}^{(t)}\}$ in exploded time plant are enabled. From our definition of time condition models, if t_1 remains state and condition enabled for time τ_{max},i.e. at $\tau = \tau' + \tau_{max}$, it would fire. Since t_1 was state and condition enabled at time $\tau' + \tau_{min}$, if it remains enabled for remaining time of $(\tau' + \tau_{max}) - (\tau' + \tau_{min}) = \tau_{max} - \tau_{min}$, it has to fire and p_2 would get marked. From our construction of exploded time plant $G_{Exp\theta}$, $\theta(t_{x1,2}) = \theta(t_1)|_{max} - \theta(t_1)|_{min}$ and t_1, $t_{x1,2}$ satisfy equation 2.2. So if t_1 fires and p_2 gets marked the corresponding post mapped transition $t_{x1,2}$ fires and the post mapped place p_{x2} becomes marked; and for any transition that belongs to $\{p_2^{(t)}\}$, there exists corresponding transition that belongs to $\{p_{x2}^{(t)}\}$ in exploded time plant and these transitions satisfy equation 2.2. Therefore at $\tau = \tau' + \theta(t_1)|_{max}$,

$TrueC_{sys\theta}(\tau' + \theta(t_1)|_{max}|_{AllC-AllState} = TrueC_{Exp\theta}(\tau' + \theta(t_1)|_{max}|_{AllC-AllState}$.
Note: $\tau_{max} = \lceil \theta(t_1)|_{max} \rceil$.

3. Suppose transition t_1 fires at time $\tau = \tau' + \tau''$, $(\tau_{min} < \tau'' < \tau_{max})$, and p_2 becomes marked. It is shown in (b) that after time $\tau = \tau' + \tau_{min}$, $p_{x1,2}$ is marked in exploded time plant $G_{Exp\theta}$ and in (c) it is shown that $p_{x1,2}$ will remain marked at least till time $\tau = \tau' + \tau_{max}$. Now, at time $\tau = \tau' + \tau''$ the

transition t_1 in time plant fires and p_2 gets marked and at the same time $p_{x1,2}$ in exploded time plant is marked from (b,c). Hence at time $\tau = \tau' + \tau''$, p_2 in the time plant and the corresponding pre-mapped place $p_{x1,2}$ in exploded time plant are marked. p_2 and $p_{x1,2}$ satisfy equation 2.3 and for any transition that belongs to $\{p_2^{(t)}\}$, there exists corresponding transition that belongs to $\{p_{x1,2}^{(t)}\}$ in exploded time plant and these transitions satisfy equation 2.2.

So at $\tau = \tau' + \tau''$, $\mathtt{TrueC}_{sys\theta}(\tau' + \tau'')|_{\mathtt{AllC-AllState}} = \mathtt{TrueC}_{Exp\theta}(\tau' + \tau'')|_{\mathtt{AllC-AllState}}$.

Case 3: Say $p_1 \in \{\mathtt{ProgConfPlaceSet}\}$ and let t'_1 be one of its output transitions such that $t_1 \in \{\mathtt{ProgConfTranSet}\}$. Since $p_1 \in \{\mathtt{ProgConfPlaceSet}\}$ there exists t_1 such that $t_1 \in \{\mathtt{ProgConfTranSet}\}$ and t'_1 is sibling of t_1, i.e. , they share the same parent place(p_1). Let p_2 be the output place of t_1 and $\theta(t_1) = [\tau_{min}, \tau_{max}]$ and let p'_2 be the output place of t'_1.For p_1, t1, p_2 in the time plant, there exists corresponding places, transitions, time assignments and mapping functions similar to one discussed in *Case 2*. Now for $p'_2 \in G_{Exp\theta}$ there exists a corresponding place p_{x2} in $G_{exp\theta}$; and for t'_1 in $G_{sys\theta}$ there exists two corresponding transitions (pre-mapped, post-mapped) $t'_{1,1}$ $t'_{1,2}$ in $G_{Exp\theta}$

Say at time $\tau = \tau'$, $p_1 \in G_{sys\theta}$ becomes marked, the corresponding place p_{x1} in $G_{Exp\theta}$ is also marked. p_1, p_{x1} satisfy equation 2.3, Therefore at time $\tau = \tau'$ t_1, $t'_1 \in \{p_1^{(t)}\}$ and the corresponding pre-mapped transitions $t_{x1,1}$, $t'_{x1,1} \in \{p_{x1}^{(t)}\}$ are state enabled. Following cases illustrate dynamics of time plant at this point:

(a) Say at time $\tau = \tau' + \Delta\tau$, t'_1 becomes condition enabled and fires marking new place p'_2 in the plant, then the corresponding pre-mapped transition $t'_{x1,1}$ in $G_{Exp\theta}$ also becomes state and condition enabled (2.2, 2.3) and fires and p'_{x2} becomes marked. Hence at time $\tau = \tau' + \Delta\tau$

$$\mathtt{TrueC}_{sys\theta}(\tau' + \Delta\tau)|_{\mathtt{AllC-AllState}} = \mathtt{TrueC}_{Exp\theta}(\tau' + \Delta\tau)|_{\mathtt{AllC-AllState}}.$$

(b) Say at time $\tau = \tau' + \Delta\tau$ both t_1, t'_1 become condition enabled and say t'_1 fires at time $\tau = \tau' + \Delta\tau + \Delta\tau_1$ such that $\Delta\tau_1 < \tau_{min}$, then the time plant and the exploded time plant follow the same path as discussed previously in (a). But if $\tau_{max} < \Delta\tau_1 \leq \tau_{min}$ then according to the algorithm of exploded time plant the post mapped place of p_1, i.e. $p_{x1,2}$ becomes marked. Now the post-mapped transitions of t_1, t'_1, i.e. $(t_{x1,2}, t'_{x1,2})$ are condition enabled in exploded time plant. So whenever t'_1 fires in time plant and p'_2 becomes marked, the corresponding post-mapped transition $t'_{x1,2}$ in the exploded time plant fires and p'_{x2} becomes marked. Therefore at $\tau = \tau' + \Delta\tau + \Delta\tau_1$

$\mathit{TrueC}_{sys\theta}(\tau' + \Delta\tau + \Delta\tau_1)|_{AllC-AllState} = \mathit{TrueC}_{Exp\theta}(\tau' + \Delta\tau + \Delta\tau_1)|_{AllC-AllState}$

Note: If $\Delta\tau_1 \geq \tau_{max}$, then t_1 in the plant is forced to fire and t'_1 becomes disabled. (This is discussed extensively in Case 2.)

Therefore, the state of the systems match at any time instant, once the initial conditions are satisfied. Hence, at any future time $\tau > \tau_0$

$\mathit{TrueC}_{sys\theta}(\tau)|_{AllC-AllState} = \mathit{TrueC}_{Exp\theta}(\tau)|_{AllC-AllState}$

The following figure gives the gist of different cases present in Lemma 2.2

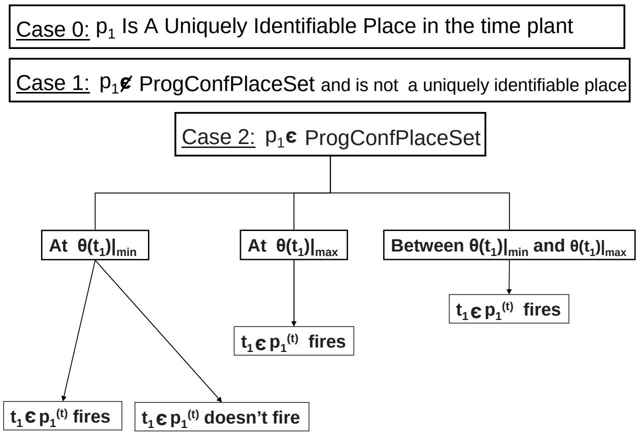
Case 0: p₁ Is A Uniquely Identifiable Place in the time plant
Case 1: p₁∉ ProgConfPlaceSet and is not a uniquely identifiable place
Case 2: p₁∈ ProgConfPlaceSet
At θ(t₁)|min
At θ(t₁)|max
Between θ(t₁)|min and θ(t₁)|max
t₁∈p₁^(t) fires
t₁∈p₁^(t) doesn't fire
t₁∈p₁^(t) fires
t₁∈p₁^(t) fires

Figure 2.7: Gist of Lemma 2.2.

Chapter 3

Timed Observer

3.1 Timed Observer Synthesis

Discrete event observers have been considered by many authors in discrete event community. Giua and Seatzu[Giua00],[Giua01] examined the observer problem in Petri nets, when the net structure is known but initial marking is unknown. They developed an observer that calculated marking estimate with the error function monotonically decreasing. Holloway and Gong [Holl01], [Holl02] defined state and condition observability criteria and developed an observer system to provide state and condition signal estimate for single-layer and multi-layer systems. Aguirre-Salas et al [Sala00] used live, cyclic and conservative interpreted Petri nets (IPN) for state estimation and for constructing asymptotic observers in discrete event systems.

Unobservability issues arise in condition systems when the state of the system cannot be determined from the observed input and output signals. In [Holl00], two types of structures were identified that result in a condition system not being observable. The first type of structure is called direction confusion, and this corresponds to having sibling places which could be marked under the observed condition inputs and outputs. The second type of structure, and the one that we focus on in this thesis, is when two places in sequence have the same condition signals, so that it is not possible to determine the transition firing from the condition observations. This is called progress confusion.

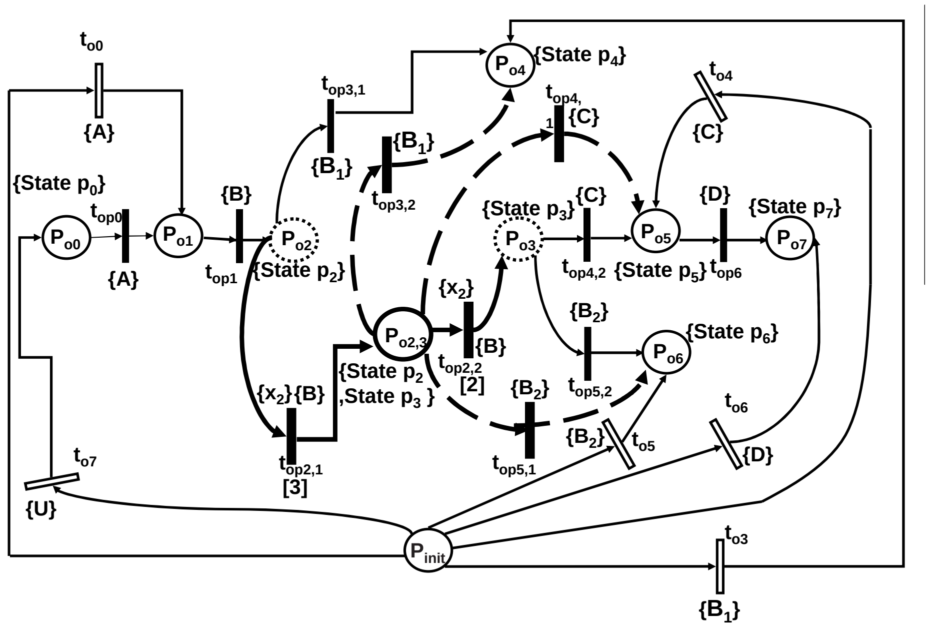

Figure 3.1: Timed observer.

Our earlier research with observers did not have a time dimension to it. Effectiveness of our observer was proved under the limitation that we don't have progress confusion and direction confusion in our model. In this section we synthesize a timed observer and prove its effectiveness even under progress confusion.

We are building a timed observer for single-layer system. A system is classified as a single layer only if it is modeled by a single plant model, i.e a single component. A single layered condition system satisfies the **StateGraph** property.

Definition 3.1 State Graph or SG property

1. State graph assumption: Each transition has exactly one input and one output place.

2. Unique marked place: For every marking for each component, there exists exactly one marked place under M.

Figure 3.2: "Timed observer synthesis" procedure.

```
Procedure Timed observer synthesis($G_{Exp\theta}$)
    Initially define $G_{obv\theta}$ such that $P_{G_{obv\theta}}, T_{G_{obv\theta}}, A_{G_{obv\theta}}$
    duplicates the exploded time plant structure $P_{G_{Exp\theta}}, T_{G_{Exp\theta}}, A_{G_{Exp\theta}}$
    For $p \in P_{G_{Exp\theta}}$ and $t \in T_{G_{Exp\theta}}$
        LET $p_{obv\theta} \in P_{G_{obv\theta}}, t_{obv\theta} \in T_{G_{obv\theta}}$ indicate corresponding
        place and transition in $G_{obv\theta}$
    For each $p \in P_{G_{Exp\theta}}$
    {
    $C_{G_{obv\theta}}(p_{obv\theta}) \Leftarrow (\{C_{G_{Exp\theta}}(p)\} \cap \{Statelabel(p)\}) \cup A(p) \cup B(p)$
    For each $t' \in p^{(t)}$
        {
        LET $p'$ denote the output place from $t'$
        If $(\{p\} \cap NDset) \cup (\{p'\} \cap NDset) = \emptyset$
    $C_{G_{obv\theta}}(t'_{obv\theta}) \Leftarrow (\{C_{G_{Exp\theta}}(p') - Statelabel\}) \cup (F(pt'p') \cup E(pt'p') \cup H(pt'p'))$
        Else {
    $C_{G_{obv\theta}}(t'_{obv\theta}) \Leftarrow (\{C_{G_{Exp\theta}}(p') - Statelabel\}) \cup (F(pt'p') \cup E(pt'p')$
                    $\cup H(pt'p')) \cup C_{G_{Exp\theta}}(t')$
        $\theta(t'_{obv\theta}) \Leftarrow \theta(t')$
            }
        }
    Create $p_{init} \in P_{G_{obv\theta}}$
    Let U={$p \in P_{G_{obv\theta}}$|p is uniquely identified without State labels}
        For each $p \in$ U
        {
        Create $t_{obv\theta} \in T_{G_{obv\theta}}$ with arc from $p_{init}$ and arc to p
        $C_{G_{obv\theta}}(t_{obv\theta}) \Leftarrow (C_{G_{Exp\theta}}(p) - Statelabel) \cup M(p) \cup N(p)$
        }
    }
```

3. I/O DISTINCTION: $C_{in}(G) \bigcap C_{out}(G) = \phi$.

Observer is a system that inputs observations from a plant and outputs an indication of the state of a plant. In timed observers we explicitly consider timing in our plant models. we define fast transitions T_{fast} as a special case of set of time transitions, which when enabled fire without any delay. Following are the conditions (Timing constraints) that have to be satisfied by the transitions to be considered as fast transitions:

- *Time plant:*

  For any $t \in G_{sys\theta}$ if $\theta(t)|_{min} = 0$ and $\theta(t)|_{max} = 0$, then t is a fast transition in the time plant.

- *Exploded time plant:*

  For any $t \in G_{Exp\theta}$ if $\theta(t) = [0]$, then t is a fast transition in exploded time plant.

Definition 3.2 Given a time τ and a system G with marking m_τ, m_τ is called a transient state at time τ if there exists a transition t in T_{fast} enabled under m_τ and C_τ.

Definition 3.3 *[Holl01], [Holl02] A system model represented by a condition system is state observable under observed condition set C_{obsd} if: For times τ_0 and τ such that $\tau_0 \geq \tau$, given any known initial marking m_{τ_0} at time τ_0, and given the observed input and output conditions $[(C_{in}(G_{sys\theta}) \cup C_{out}(G_{sys\theta})) \cap C_{obsd}]$ over period τ_0 to τ, we can uniquely determine marking m_τ at time τ.*

So from the definition, a system will be state observable if knowledge of its past observable conditions is sufficient to uniquely determine its current state.

A Time plant with progress confusion is not “state observable". This state estimation problem is solved by creating exploded time plant procedure in fig.2.3 and fig.2.5. It has also been proved from lemma 2.2 that exploded time plants maintains the structural and temporal integrity of time plant.

A Timed observer is built for the time plant using the exploded time plant as input. The procedure in fig.3.2 to synthesize timed observer from exploded time plant (ExTP) is described below [Holl01], [Holl02].

1. Replicating the Exploded time plant(Lines 1-3)

   This is done by copying the structure of ExTP. Each place 'p' in ExTP has a corresponding place $p_{obv\theta}$ in the timed observer and each transition 't' in ExTP has a corresponding transition $t_{obv\theta}$ in the timed observer. Lines 4,5 assign the state labels to places in timed observer by extracting state labels from corresponding places in ExTP. For a place p in the exploded time plant, define sets $A(p), B(p) \subseteq AllC$ such that

   A(p):=$\{c|c \in C_{G_{Exp\theta}}(p) \setminus C_{obsd}\}$
   B(p):= $\{c|c \notin C_{obsd}$ and $\forall t \in^{(} t)p, C_{G_{Exp\theta}}(t) = c$ and
   $\exists t' \in p^{(t)}, C_{G_{Exp\theta}}(t')= \bar{c}\ \}$

2. Checking for Progress Confusion(Lines 6-8)

   Since the ExTP satisfies SG property, every transition t' in the ExPT has only one input place (p) and output place (p'). This property would also hold true for timed observer, because the structure of timed observer was replicated from ExTP. Lines 7,8 check for progress confusion by comparing the state labels of the places in NDset with the state labels of the input and output places of the transition t'.

3. Ascertain conditions and timing on the transitions of timed observer(Lines 9-12)

Line 9 assigns conditions to the transitions whose input and output places are deterministic, i.e their state labels don't $\in$ NDset .Lines 11, 12 assign conditions to the transitions whose input or output or both I/P,O/P places are non deterministic, i.e their state labels $\in$ NDset. The conditions associated with the transition $(t'_{obv\theta})$ consists of the following sets.

- $C_{G_{Exp\theta}}(p')$

  These are the conditions that are output by the place that is newly marked in the ExTP. These must be true if p' is marked, and thus must be true for $(t'_{obv\theta})$to be enabled and fire.

- $F_\theta(pt'p')$

  $F_\theta(pt'p') = F(pt'p')$-{State label}

  The place p denotes the previously marked place in Exploded time plant. Upon the firing of the transition t' in the plant, the conditions for it that are not in the newly marked place will become false. The set $F(pt'p')$ corresponds to the negation of these conditions. Thus, if the previously marked place in the plant is still marked, some condition in F (if $F \neq 0$) will be false, and the transition $(t'_{obv\theta})$ will not fire.

- $E_\theta(pt'p')$

  $E_\theta(pt'p') = E(pt'p')$-{State label}

  $E(pt'p')$ indicates the set of conditions that should be false unless some sibling place of p' became marked. If a sibling place of p' has become marked, then some condition in $E(pt'p')$ will be false, and so $(t'_{obv\theta})$ will not fire.

- $H_\theta(pt'p')$

  In cases when the conditions between the place p' and its sibling are identical, then determining which transition fired must be done by considering the enabling condition sets on the transitions leading to the place, and those condition sets must be exclusive since the system and the exploded time plant have no direction confusion. In such a case, H is the enabling condition set for t' . Thus, $(t'_{obv\theta})$ can fire only if t' was enabled and transitions leading to indistinguishable sibling places were not disabled.

4. CREATING INITIAL TIMED OBSERVER STATES(Lines 13-18)

   A place p_{init} is created. From p_{init} are created transitions leading to observer places corresponding places in ExTP, which are uniquely identifiable without state labels. The initial place of the observer indicates that the state of the plant is not yet known. As soon as a uniquely labelled place in the plant becomes marked, then the timed observer can follow the observable plant thereafter.

Following lemma proves the effectiveness of synthesized timed observer.

Lemma 3.1 *Consider a single-layer timed plant $G_{sys\theta}$, its corresponding exploded time plant $G_{Exp\theta}$ and an observed condition set C_{obsd} such that $G_{sys\theta}$ and $G_{Exp\theta}$ satisfy SG property without direction confusion under C_{obsd}. Let $G_{obv\theta}$ be the timed observer synthesized by the algorithm in figure 3.2. Let p_{init} be marked in $G_{obv\theta}$ at time τ_{init}, and let the plant first visit a uniquely labelled state at some time $\tau_0 > \tau_{init}$. Then under Timed observer/System Speed Assumption, where the system and the observer* *are in non-transient states, any actual plant state belongs to observer state estimate set and observer state estimate contains all and only possible states in the plant.*

From the definition of uniquely labelled place we know that at initial time τ_0, the plant visits a uniquely labelled place $p_{sys\theta}$; by construction the place $p_{Exp\theta} \in G_{Exp\theta}$ corresponding to $p_{sys\theta} \in G_{sys\theta}$ is marked. We should show that any change of state in the exploded time plant is immediately followed by the corresponding state change in the timed observer. In other words any subsequent transition firing in the exploded time plant is immediately followed by the corresponding transition firing in the timed observer. Suppose that for some time τ, $m_{Exp\theta}(\tau^-)$ and $TrueC_{Exp\theta}(\tau^-)$ are known. Let p denote the marked place in $G_{Exp\theta}$ under $m_{Exp\theta}(\tau^-)$.

First, assuming that there is no progress confusion in time plant, and in the corresponding exploded time plant,suppose that no transition fires at time τ so p is marked under $m_{Exp\theta}(\tau)$. Then $[TrueC_{Exp\theta}(\tau^-)\bigcap C_{out}(G_{Exp\theta})]|_{AllC-AllState}$ $= [TrueC_{Exp\theta}(\tau)\bigcap C_{out}(G_{Exp\theta})]|_{AllC-AllState}$. Since there is no progress confusion, then for any $t' \in p^{(t)}_{Exp\theta}$ and $p'_{Exp\theta} \in t'^{(p)}$, $C_{G_{Exp\theta}}(p)|_{AllC-AllState} \neq C_{G_{Exp\theta}}(p')|_{AllC-AllState}$. The transition $t'_{obv\theta}$ will not fire unless each $c \in (C_{G_{Exp\theta}}(p') \setminus C_{G_{Exp\theta}}(p))|_{AllC-AllState}$ is true, since $C_{G_{Exp\theta}}(p') \subseteq C_{G_{obv\theta}}(t'_{obv\theta})$ (Line 5). This will not happen while p is marked. It will also not fire unless all $c \in C_{G_{Exp\theta}}(p) \setminus C_{G_{Exp\theta}}(p')$ are false (since $F_\theta(pt'p') \subseteq C_{G_{obv\theta}}(t'_{obv\theta})$), which again will not occur while p is marked. Thus, if no transition fires in $G_{Exp\theta}$, then no transition will fire in $G_{obv\theta}$.

Next suppose the transition t' fires in $G_{Exp\theta}$. We want to show that transition $t'_{obv\theta}$ in the observer is enabled and will fire. For this we must show each $c \in C_{G_{obv\theta}}(t'_{obv\theta})$ is true. If the transition $t' \in G_{Exp\theta}$ fires, p' becomes marked. hence, each $c \in C_{G_{Exp\theta}}(p')$ is true. Now that p' is marked and there is no progress confusion, $TrueC_{Exp\theta}(\tau^-)|_{AllC-AllState} \neq TrueC_{Exp\theta}(\tau)|_{AllC-AllState}$. So, if

$c \in F_\theta(pt'p') \bigcup E_\theta(pt'p')$, then c is now true since $\bar{c} \notin C_{G_{Exp\theta}}(p')$. If $c \in H(pt'p')$, then c is true since it was necessarily true for t' to fire in the plant. Thus $t'_{obv\theta}$ is enabled and can fire.

Now say at time τ^- ($\tau^- > \tau_0$), the time plant becomes marked at a place having progress confusion, then the corresponding place p in the exploded time plant is also marked. Suppose that no transition fires at time τ ($\tau > \tau^-$); i.e., at time τ the transition $t' \in p^{(t)}$ is (i) not condition enabled (or) (ii) condition enabled but not time enabled. So p is marked under $m_{Exp\theta}(\tau)$. If t' is not condition enabled then $[TrueC_{Exp\theta}(\tau^-) \bigcap C_{out}(G_{Exp\theta})]|_{AllC-AllState} = [TrueC_{Exp\theta}(\tau) \bigcap C_{out}(G_{Exp\theta})]|_{AllC-AllState}$.

To prove that no transition will fire in $G_{obv\theta}$ at time τ , we have to show that some $c \in C_{G_{obv\theta}}(t'_{obv\theta})$ is false or $\theta(t'_{obv\theta})$ is not satisfied at time τ. If t' is not enabled then at least one $c \in C_{G_{Exp\theta}}(t')$ is false. From line 11 in fig.3.2 we see that $C_{G_{Exp\theta}}(t') \in C_{G_{obv\theta}}(t'_{obv\theta})$. hence, if $C_{G_{Exp\theta}}(t')$ is false $t'_{obv\theta}$ will not fire.

Now if t' is condition enabled but not time enabled, i.e. at time τ every $c \in C_{G_{Exp\theta}}(t')$ is true but $\xi(\tau)(t') < \theta(t')$, the transition t' would not fire. From construction the transition should be both condition and time enabled before it can fire. From line 12 in fig.3.2$\theta(t'_{obv\theta}) = \theta(t') > \xi(\tau)(t')$. hence $t'_{obv\theta}$ will not fire. Thus if no transition fires in $G_{Exp\theta}$, then no transition will fire in $G_{obv\theta}$.

Next suppose t' fires in $G_{Exp\theta}$. We want to show that transition $t'_{obv\theta}$ in the observer is enable and will fire. For this we must show that each $c \in C_{G_{obv\theta}}(t'_{obv\theta})$ is true and $\xi(\tau)(t'_{obv\theta}) = \theta(t')$. If the transition t' fires then $p' \in G_{Exp\theta}$ becomes marked. Therefore each $c \in C_{G_{Exp\theta}}(p')$ is true. Since there is progress confusion $F_\theta(pt'p') = \emptyset$. If $c \in E_\theta(pt'p')$, then c is now true since $\bar{c} \in C_{G_{Exp\theta}}(p')$. If $c \in H_\theta(pt'p')$ then c is true since it was necessarily true for t' to fire and by construction and temporal properties of the exploded time plant $\xi(\tau)(t')$ should be equal to $\theta(t')$ for the transition t' to fire in the plant. Thus $t'_{obv\theta}$ is enabled and can fire.

We showed that regardless of presence of progress confusion, $t'_{obv\theta}$ is enabled in $G_{obv\theta}$, now it is to be shown that no other transition is enabled in $G_{Exp\theta}$ at time τ. Consider another transition t'' with input p and p'' as its output. Since the time plant model is built under " Time Plant Model Limitations" there is no direction confusion. So, either $C_{G_{Exp\theta}}(p')|_{Allc-AllState} \neq C_{G_{Exp\theta}}(p'')|_{Allc-AllState}$ or There exists $c \in C_{G_{Exp\theta}}(t'')$ which is exclusive to $C_{G_{Exp\theta}}(t')$

1. $C_{G_{Exp\theta}}(p')|_{Allc-AllState} \neq C_{G_{Exp\theta}}(p'')|_{Allc-AllState}$ implies either

   (a) $[C_{G_{Exp\theta}}(p'')\backslash C_{G_{Exp\theta}}(p')]|_{AllC-AllState} \neq \phi$

   or

   (b) $[C_{G_{Exp\theta}}(p')\backslash C_{G_{Exp\theta}}(p'')]|_{AllC-AllState} \neq \phi$

   or

   (c) both

if (a), it means that there exists c such that $c \in C_{G_{Exp\theta}}(p')$, that implies $t''_{obv\theta}$ cannot be enabled since $C_{G_{Exp\theta}}(p'')|_{AllC-AllState} \subseteq\in C_{G_{obv\theta}}(t''_{obv\theta})$ and condition c is false. If (b) is true then $c \in E_\theta(pt''p'') \subseteq C_{G_{obv\theta}}(t''_{obv\theta})$. in any of the cases $t''_{obv\theta}$ is not enabled.

If $C_{G_{Exp\theta}}(p')|_{Allc-AllState} = C_{G_{Exp\theta}}(p'')|_{Allc-AllState}$ then there exists $c \in H_\theta(pt''p'')$ that is exclusive to $c \in H_\theta(pt'p')$ hence there is some $c \in C_{G_{obv\theta}}(t'')$ which is exclusive to $C_{G_{obv\theta}}(t')$ and so transition $t''_{obv\theta}$ is not enabled. Thus, we have shown that in absence of progress confusion and direction confusion, the only transition enabled in $G_{obv\theta}$ corresponds to exploded time plant transition that fired, which corresponds to plant transition that fired. Thus, we have shown that the only transition enabled in $G_{obv\theta}$ corresponds to the plant transition fired. Under the assumption that enabled transitions in observer fire immediately, then for any place p in $G_{Exp\theta}$ which is marked, the place $p_{obv\theta}$ is also marked, and so by line 5 of the observer synthesis procedure, condition label $\mathsf{State}(p) \in C_{G_{obv\theta}}(p_{obv\theta})$, and thus will have a value true. Since $G_{obv\theta}$ satisfies state graph property, $p_{obv\theta}$ will be the only place marked, and so $\mathsf{State}(p')$ for all other $p' \neq p$ will be false.

Theorem 3.1 Given any time-condition system plant $G_{sys\theta}$ satisfying the "Time Plant Model Limitations"; and the corresponding exploded time plant $G_{Exp\theta}$ created from procedures in fig. 2.3, 2.5, then $G_{Exp\theta}$ is state observable:

Proof: To show observability, it is sufficient to show that each transition firing can be known with certainty. Note that the marking and clock vector is known with certainty at time τ_θ. Consider some transition t in exploded time plant $G_{Exp\theta}$, if transition t has a non-deterministic time interval $(i.e.\ \ \theta(t)|_{min} \neq \theta(t)|_{max})$, then it must not correspond to a progress confusion transition in time plant, $t \notin \mathrm{ProgConfTranSet}(G_{sys\theta})$, so firing of t can be determined from observing a change in condition outputs of $G_{Exp\theta}$. If we have a transition t with deterministic time interval $(i.e.\ \ \theta|_{min}(t) = \theta|_{max}(t))$, then as long as we know by the timing when the transition became enabled (BY PLACES BECOMING MARKED VIA PREVIOUS TRANSITION FIRINGS), then we know by the timing when t fires.

Next, we must show there will be no direction confusion in the exploded time plant. From "Time Plant Model Limitations" it is clear that the original time plant $G_{sys\theta}$ has no direction confusion. Structure of the time plant with **NO PROGRESS CONFUSION** is absolutely identical to the corresponding exploded time plant structure. and since, direction confusion is a structural property; if there is no direction confusion in time plant, there is no direction confusion in exploded time plant. Now it only remains to show that the exploded time plant $(G_{Exp\theta})$ does not have any direction confusion if the corresponding time plant **HAS PROGRESS CONFUSION**. For every

p_i, p_j in time plant $G_{sys\theta}$, there exists corresponding places p_{xi}, p_{xj} in exploded time plant $G_{Exp\theta}$. Say some p_i and p_j in the time plant have progress confusion. The structural change in the corresponding exploded time plant by introduction of progress confusion in time plant would be the creation of non-deterministic (macro) place $p_{xi,j}$ in $G_{exp\theta}$. This place $p_{xi,j}$ has all the outputs of p_{xi}, p_{xj} as its output; $p_{xi,j}$ is the parent place of all the outputs of p_{xi}, p_{xj}; From the rule 4 of “Time Plant Model Limitations", we can say that for all $p'_{xi} \in (p_{xi}^{(t)})^{(p)}$ and $p'_{xj} \in (p_{xj}^{(t)})^{(p)}$, $C_{G_{Exp\theta}}(p'_{xi}) \neq C_{G_{Exp\theta}}(p'_{xj})$. This makes all the output places of $p_{xi,j}$ distinguishable, hence there is no direction confusion in $G_{Exp\theta}$. Hence $G_{Exp\theta}$ is state observable.

Chapter 4

Controller Synthesis

4.1 Generated State Label Plant

Figure 4.1: Generated State label plant procedure

```
**Procedure** Generated State label plant($G_{Exp\theta}$)
Initially define $G_{Gsp\theta}$ such that $P_{G_{Gsp\theta}}, T_{G_{Gsp\theta}}, A_{G_{Gsp\theta}}$
duplicate the exploded time plant structure $P_{G_{Exp\theta}}, T_{G_{Exp\theta}}, A_{G_{Exp\theta}}$
For $p \in P_{G_{Gsp\theta}}$ and $t \in T_{G_{Gsp\theta}}$
    LET $p_{Gsp\theta} \in P_{G_{Gsp\theta}}, t_{Gsp\theta} \in T_{G_{Gsp\theta}}$ indicate corresponding
    place and transition in $G_{Gsp\theta}$
For each $p \in P_{G_{Exp\theta}}$
$C_{G_{Gsp\theta}}(p_{Gsp\theta}) \Leftarrow (C_{G_{Exp\theta}}(p) \cap Statelabel(p))$
For each $t \in T_{G_{Exp\theta}}$
{
    $C_{G_{Gsp\theta}}(t_{Gsp\theta}) \Leftarrow C_{G_{Exp\theta}}(p)$
        $\theta(t_{Gsp\theta}) \Leftarrow \theta(t)$
}
```

In our previous research, discrete event controllers have been developed for condition systems [Holl00]. The controller developed by *Holloway et al.* [Holl00] is a set of interacting taskblocks,each of which is automatically synthesized from a corresponding component model satisfying state structure assumption. The controller developed in this chapter uses the same principle and construct as discussed in [Holl00]. The key difference is that the taskblocks in this controller are synthesized from a generalized state label plant, an intermediate component model that represents the original model and also satisfies observability criteria of *System Structure Assumption*, discussed later

Note: Original model doesn't satisfy System Structure Assumption (SSA).

In the Generated state label plant, all the conditions on places other than state label conditions are erased. Hence, the condition set on each of the places in this plant is now unique, making it state observable. The algorithm in fig.4.1 converts the Exploded time plant into Generated state label plant.

- Lines 1-4 in the algorithm reproduce the Exploded time plant structure. For every place 'p' and transition 't' in the Exploded time plant, there will be a corresponding place $p_{Gsp\theta}$ and transition $t_{Gsp\theta}$ in Generated state label plant.

- Lines 5-6 extract only the state label conditions from the places in Exploded time plant and assign them to corresponding places in Generated state label plant. Lines 8-9 assign the conditions and timing information on the transitions from the Exploded time plant to their corresponding transitions in Generated state label plant.

One of the main goals of the thesis is to synthesize controller for the *time plant*. This is done by creating an intermediate model, which we call *Generated State Label Plant* and by using timed observer that is synthesized from exploded time plant. It is then shown that controller generated from generated state label Plant will be effective when applied for the composition of time plant with a timed observer.

In chapter 2 it was shown that the exploded time plant models and actual time plant models share the same behavioral and temporal properties, " Lemma.2.2". In chapter 3 It was shown that timed observer built from exploded time plant can also act as an observer for the time plant from lemma 3.1. In this chapter we show that generated state label plant shares the same behavioral and temporal properties with exploded time plant, therefore the timed observer built from the exploded time plant also acts as an observer for the generated state label plant.
From *Holloway et al., 2000*, The input/output behavior of the system can be described by sequences of condition sets. A condition set sequence, called a C-sequence, is a finite length sequence of condition sets. Each condition set sequence is of the form $(C_1, C_2, C_3, ...C_n)$ for some integer n and sets $C_i \subseteq \mathcal{A}\updownarrow\updownarrow\mathcal{C}$ for all $0 \leq i \leq n$. Here, In generated state label plant has all the places have unique state labels as condition set on them. Given C-sequences s_1, s_2; let expression $s_1 s_2$ indicate concatenation of s_2 on the end on s_1. A set of C-sequences is called language, and the language consisting of all C-sequences is denoted $\mathcal{L}$.

Definition 4.1 *Holloway et al., 2000 Define the descriptive ordering $\leq$ over condition sequences such that*

1. $(C_1, C_1') \leq (C_2)$ *if* $C_1 \subseteq C_2$ *and* $C_1' \subseteq C_2$.

2. $(C_1) \leq (C_2 C_2')$ *if* $C_1 \subseteq C_2$ *and* $C_1 \subseteq C_2'$.

3. *Given C-sequences* s_1, s_1', s_2, s_2' *such that* $s_1 \leq s_1'$ *and* $s_2 \leq s_2'$*, then* $s_1 s_2 \leq s_1' s_2'$

4. *If* $s_1 \leq s_2$ *and* $s_2 \leq s_3$*, then* $s_1 \leq s_3$

In brief given s_1, s_2 as specification of conditions which are known to be true sequentially overtime, $s_1 \leq s_2$ means that s_2 contains at least as much specification of condition values as s_1, i.e. s_2 is at least as *descriptive* as s_1.

Theorem 4.1 *Consider a single layer timed plant* $G_{sys\theta}$*, its corresponding exploded* *time plant* $G_{Exp\theta}$ *such that* $G_{sys\theta}$ *and* $G_{Exp\theta}$ *satisfy* ***S.G property*** *and* ***Time Plant Model Limitations****. Let* $G_{Gsp\theta}$ *be the generated state label plant synthesized by algorithm in fig. 4.1, then*

For any timed condition sequence $\mathrm{TrueC}(\cdot)$ *in the time condition model*

$\mathrm{TrueC}(\cdot)|_{(C_{in}(G_{Gsp\theta})\cup AllState)} \in L(G_{Gsp\theta}, (m_0, \xi_0))|_{C_{in}(G_{Gsp\theta})\cup AllState}$ ***iff***

$\mathrm{TrueC}(\cdot)|(C_{in}(G_{Exp\theta})\cup AllState) \in L(G_{Exp\theta}, (m_0, \xi_0))|_{C_{in}(G_{Exp\theta})\cup AllState}$

proof: From construction in algorithm 4.1, for any place $p_{xi} \in P_{Exp\theta}$, there exists $p_{gsi} \in P_{Gsp\theta}$ such that $C_{G_{Exp\theta}}(p_{xi})|_{AllState} = C_{G_{Gsp\theta}}(p_{gsi})|_{AllState}$, and for each $t \in p_{xi}^{(t)}$ there exists a unique $t' \in p_{gsi}^{(t)}$ such that $C_{G_{Exp\theta}}(t) = C_{G_{Gsp\theta}}(t')$.

Now say at time τ_0, p_{xi} is marked; i.e., $m(\tau_0)(p_{xi}) \neq 0$. From the generated state label plant procedure in fig.4.1 line 6 we see that $C_{G_{Exp\theta}}(p_{xi}) \subseteq C_{G_{Gsp\theta}}(p_{gsi})$. Hence, if $p_{xi} \in G_{Exp\theta}$ is marked so is $p_{gsi} \in G_{Gsp\theta}$. We know that for any $t \in P_{xi}^{(t)}$ there is a unique $t' \in p_{gsi}^{(t)}$, both having identical conditions for state and condition enabling. So, if t fires in $G_{Exp\theta}$ at time τ, corresponding p_{gsi} would also fire in $G_{Gsp\theta}$ at the same time instant. Therefore

$$\mathrm{TrueC}_{Gsp\theta}(\tau) = \mathrm{TrueC}_{Exp\theta}(\tau)$$

$$\xi_{Gsp\theta}(\tau)(t') = \xi_{Exp\theta}(\tau)(t)$$

Hence for each c-sequence in exploded time plant there is a corresponding c-sequence in generated state label plant. Similarly from construction, for every c-sequence in generated state label plant there exists a corresponding c-sequence in exploded time plant.

4.1.1 TaskBlocks

The plants that we consider to be controlled are modeled by collections of condition models components of the plant. Let this set of condition models representing

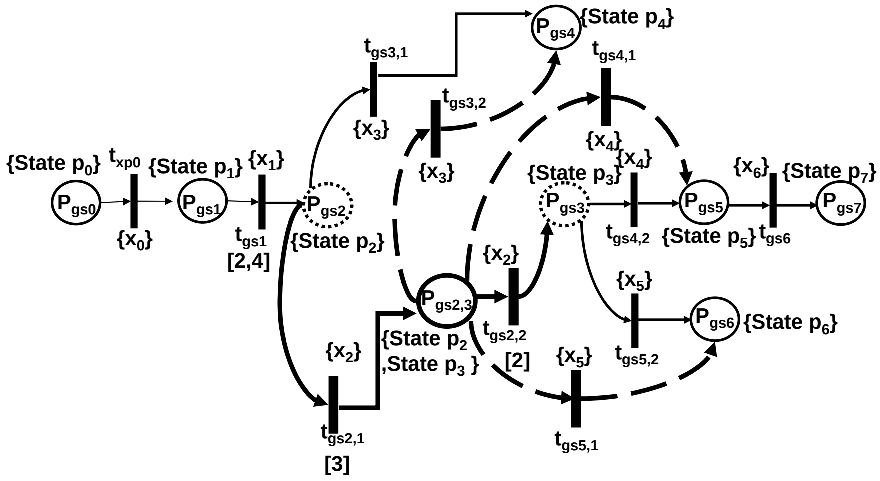

Figure 4.2: Generated State Label Plant.

components be denoted as $\mathcal{G}_{\mathrm{compo}}$. The controllers that we consider are also represented as collections of condition models. The set of these controller models, representing elements of control logic, are called *taskblocks*, are denoted as the set $\mathcal{G}_{\mathrm{tasks}}$. A system $\mathcal{G}$ then can consist of a collection of both component models and taskblocks operating together.

Each taskblock has a specific control function. A taskblock becomes *activated* to begin its control function upon its *activation condition*, which uniquely identifies the taskblock. Let $\mathrm{C}_{\mathrm{do}} \subset \mathcal{A}\updownarrow\updownarrow\mathcal{C}$ be the set of *activation conditions* associated with taskblocks. For each element $\mathrm{do} \in \mathrm{C}_{\mathrm{do}}$ we associate the following:

- $\mathbb{TB}(\mathrm{do}) \in \mathcal{G}_{\mathrm{tasks}}$ is the unique taskblock (condition system model) for which $\mathrm{do} \in \mathrm{C}_{\mathrm{in}}(\mathbb{TB}(\mathrm{do}))$. No other taskblocks or components have do as an input.

- $\mathtt{compl}(\mathrm{do}) \in \mathrm{C}_{\mathrm{out}}(\mathbb{TB}(\mathrm{do}))$ is a condition output from the taskblock, indicating task completion.

- $\mathtt{idle}(\mathrm{do}) \in \mathrm{C}_{\mathrm{out}}(\mathbb{TB}(\mathrm{do}))$ is a condition output from the taskblock and indicates that the taskblock is not activated. There exists exactly on place p in $\mathbb{TB}(\mathrm{do})$ for which $\mathtt{idle}(\mathrm{do})$ is an output, and furthermore, it is the only output of that place, $C_{\mathbb{TB}(\mathrm{do})}(p) = \{\mathtt{idle}(\mathrm{do})\}$. In all subsequent discussion, we will assume each task block only has this place marked under any initial marking considered.

- $\mathcal{G}_{\mathrm{compo}}(\mathrm{do}) \in \mathcal{G}_{\mathrm{compo}}$ is a component model associated with the task do. The same component model may be associated with many different tasks. When the activation condition has a subscript indicating its goal (such as do_x for $\mathtt{goal}(\mathrm{do}_x)) = x$), and a unique component outputs that condition, then we use the subscript to indicate the component net which outputs the target. Thus, $G_{\mathrm{compo}}(\mathrm{do}_x) = G_x$ is the net which outputs condition x.
- $\mathtt{goal}(\mathrm{do}) \in C_{\mathrm{out}}(G_{\mathrm{compo}})(\mathrm{do}_x) = G_x$ is the net which outputs condition x.
- $C_{\mathrm{init}}(\mathrm{do}) \subseteq C_{\mathrm{in}}(\mathbb{TB}(\mathrm{do})) \cap C_{\mathrm{out}}(G_{\mathrm{compo}}(\mathrm{do}))$ is a set of *initiation conditions* for the taskblock that are output from the component $G_{\mathrm{compo}}(\mathrm{do})$.

We interpret a taskblock as follows: The output condition $\mathtt{idle}(\mathrm{do})$ indicates that the taskblock is not currently outputting any other conditions. A taskblock $\mathbb{TB}(\mathrm{do})$ becomes active (and thus *idle(do)* becomes false) upon the conditions $\{\mathrm{do}\} \cup C_{\mathrm{init}}(\mathrm{do})$ becoming all true. As long as do remains true, the taskblock and system components will interact until eventually the condition $\mathtt{goal}(\mathrm{do})$ is output from the taskblock, indicating completion of the task. Whenever do becomes false, the taskblock returns to the idle state. The following definition of *effective* formally describes the behavior of a taskblock when it interacting with a system in its intended manner.

Definition 4.2 *(Modified from (Holloway et al., 2000)) Given a system* $\mathcal{G} \subseteq \mathcal{G}_{\mathrm{tasks}} \cup \mathcal{G}_{\mathrm{compo}}$ *with initial state* $m(\tau_0)$ *and a condition* $\mathrm{do} \in C_{\mathrm{in}}(\mathcal{G}) \cap \mathcal{C}_{\mathrm{do}}$ *such that* $\mathtt{idle}(\mathrm{do}) \in \mathtt{TrueC}(\tau_0)$*,* do *is effective for* $\mathcal{G}$ *under* $m(\tau_0)$ *if each of the following statements are true:*

1. *Continued activation implies eventual completion:* For all $s \in L(G_{gsp\theta}), m(\tau_0)$, if

$(\phi(\{do\}\cup C_{init}(do))) \leq s$, then for any C_{ext} and $C_{ext}\cap(C_{out})(G_{Gsp\theta})\cup\{\neg do\} = \phi$, there exists s' such that $ss' \in L(G_{Gsp\theta}, m(\tau_0)), (C_{ext}) \leq s'$, and

$$(\{do\}\{do, compl(do)\}) \leq s'$$

2. *Completion implies earlier activation:* For all $s \in L(G_{Gsp\theta}, m(\tau_0))$, if $(\phi\{compl(do)\}) \leq s$, then

$$(\phi(\{do\} \cup C_{init}(do))\phi) \leq s$$

3. *Completion implies achieved goal:* For any condition set string s and any condition set C such that $sC \in L(G_{Gsp\theta}, m(\tau_0))$, if $\{compl(do)\} \subset C$, then

$$\{compl(do), goal(do)\} \subseteq C$$

4. *Leaving completion implies earlier deactivation:* For all $s \in L(G_{Gsp\theta}, m(\tau_0))$, if $(\phi\{compl(do)\}\{\neg compl(do)\}) \leq s$ then

$$(\phi\{\neg do\}\phi) \leq s$$

5. *Deactivation implies eventual return to idle:* For all $s \in L(G_{Gsp\theta}, m(\tau_0))$, if $(\phi\{\neg do\}) \leq s$, for any C_{ext} such that $\neg do \in C_{ext}$ and $C_{ext} \cap (C_{out}(G_{Gsp\theta}) \cup \{do\}) = \phi$, there exists s' such that $ss' \in L(G_{Gsp\theta}, m(\tau_0)), (C_{ext}) \leq s'$, and

$$(\{\neg do\}\{\neg do, idle(do)\}) \leq s'$$

The first statement states that after do and $C_{init}(do)$ conditions are true,if do remains true, then there will eventually follow a completion condition $compl(do)$ from the taskblock, and completion is reached entirely through the interaction of taskblocks and components in $G_{Gsp\theta}$ and not any other external condition.

Now, we consider methods of synthesizing taskblocks. For each component model and each output condition of the components, we consider two types of

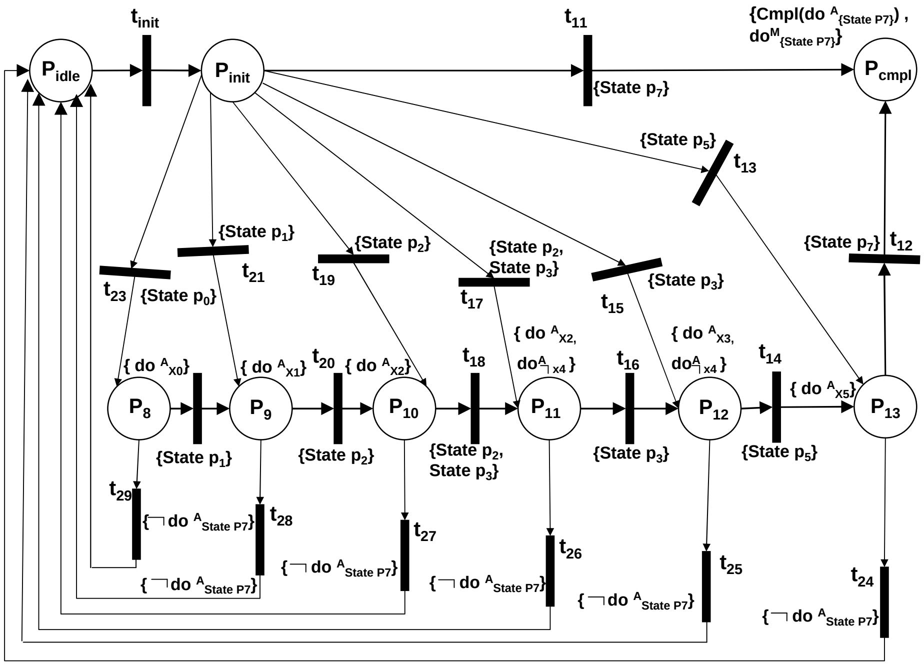

Figure 4.3: Controller.

taskblocks. The first type is called a maintain-type, and its purpose is to keep a condition of the system true, given that it was already true when the taskblock was activated. The second type is called an action-type. Its type is to drive the system to a given condition from any initial state. Below, we present definitions of maintain-type and action type taskblocks, we then present an algorithm for synthesizing action-type blocks, and later we present conditions under which they are effective. For a given condition x, we distinguish between the action-type and maintain-type taskblocks through the activation signals: do_x^A is the activation condition for the action-type taskblock $TB(do_x^A)$ with $goal(do_x^A) = x$, and do_x^M is the activation condition for the maintain-type taskblock $TB(do_x^M)$ with $goal((do_x^M)) = x$.

A maintain-type taskblock will keep a given system condition true, as long as the condition was true initially when the taskblock was activated. This is formally stated as follows:

Definition 4.3 (*Holloway et al., 2000*) Given a target condition x from the system, a taskblock with activation signal do_x^M is a maintain-type taskblock for x if:

1. $C_{init}(do_x^M) = \{x\}$

2. $goal(do_x^M) = x$

Action-type taskblocks are intended to drive a component to an intended goal from any initial state. This is formally stated as follows:

Definition 4.4 (*Holloway et al., 2000*) Given a target condition x from the component, a taskblock with activation signal do_x^A is an action-type taskblock for x if

1. $C_{init}(do_x^A) = \{\phi\}$

2. $goal(do_x^A) = x$

We note in advance that for a given desired system condition, there may be many maintain-type and action-type taskblocks. Finding an "optimal" such taskblock is a subject of future research, but is not addressed in this thesis. we assume that the taskblocks might not interact directly with the system components. Instead, there may be intermediate taskblocks or components. Thus, the taskblocks that we synthesize will only output activation conditions. These are either inputs to other taskblocks or to a direct translator that then interacts with the system component.

To translate a set of conditions C into activation conditions(either for input to other taskblocks or to a direct translator), we introduce the function

$$Act(C) = \{do_c^A | c \in C\}$$

Thus, $Act(C)$ gives the activation signals for the action-type taskblocks for each condition in C.

Throughout the chapter, we assume the systems consist of components that satisfy the following assumption.

*System Structure Assumption (SSA):*Consider a system component $G_{Gsp\theta}$ with set of states $M_{Gsp\theta}$. The system satisfies the System Structure Assumption (SSA) for $M_{Gsp\theta}$ if the following are true:

1. *Structure:* For all transitions t in $G_{Gsp\theta}$, there exists exactly one input place and one output place for t. For every place, there exists a path to every other place.

2. *States:* $M_{Gsp\theta}$ consists of all states with single place marked.

3. *Observability:* Each place of $G_{Gsp\theta}$ is uniquely identified by its conditions, so for each place p, there exists some $C \subseteq C_{Gsp\theta}(p)$ such that for all $m \in M_{Gsp\theta}$, $(m(p) = 1) \Leftrightarrow (C \subseteq TrueC(\cdot))$.

4. *Transition selectability* For any place p in $G_{Gsp\theta}$, for all transitions t, t' out from p, where $t \neq t'$, $C_{Gsp\theta}(t) \nsubseteq C_{Gsp\theta}(t')$. Also, for all transitions t in $G_{Gsp\theta}$, condition input set $C_{Gsp\theta}(t)$ is contradiction free and nonempty.

5. *All output conditions have truth established by marking:* For any condition output $c \in C_{out}(G_{Gsp\theta})$, for all markings $m \in M_{Gsp\theta}$, either $c \in TrueC$ or$\neg c \in TrueC$, but not both.

Action-Type Taskblock synthesis

In figure 4.6, we show an algorithm to make an action-type taskblock for a given component model and target condition. The algorithm consists of three procedures. CREATE AB$\theta(C_{targ})$ is the top level function. It begins by creating three places: p_{idle} is the place marked when the taskblock is idle, p_{init} is a place visited momentarily following taskblock activation, and p_{cmpl} is the place that will be marked when the task is completed and the component is outputting goal condition c_{targ} . Note that p_{cmpl} outputs the activation condition $do^{M}_{c_{targ}}$, thus activating the maintain taskblock associated with c_{targ} and thus keeping c_{targ} as long as the

Figure 4.4: "Timed Controller Synthesis" procedure

```
**Procedure** CREATE AB$\theta(C_{targ})$
$P_{TB} = \emptyset$ $T_{TB} = \emptyset$ $A_{TB} = \emptyset$
**Create** place $p_{idle}$ in $P_{TB}$ with $C_{TB}(p_{idle}) \Leftarrow \{idle(do^{A}_{C_{targ}})\}$
**Create** place $p_{init}$ in $P_{TB}$ with $C_{TB}(p_{init}) \Leftarrow \emptyset$
**Create** place $p_{cmpl}$ in $P_{TB}$ with $C_{TB}(P_{cmpl}) \Leftarrow \{cmpl(do^{A}_{C_{targ}}), do^{M}_{C_{targ}}\}$
**Create** transition $t_{init}$ in $T_{TB}$ from $p_{idle}$ to $p_{init}$ with $C_{TB}(t_{init}) \Leftarrow \{do^{A}_{C_{targ}}\}$
**Define** $P_{targ} \Leftarrow \{p_{G_{Gsp\theta}}$ in $G_{Gsp\theta} | C_{targ} \in C_{G_{Gsp\theta}}(p_{Gsp\theta})\}$
    **For** each $p_{Gsp\theta} \in P_{targ}$
    BUILDP$(p_{G_{Gsp\theta}}, p_{cmpl}, P_{targ})$
    **For** each $p \in P_{TB} - \{p_{idle}, p_{init}\}$
    **Create** transition $t \in T_{TB}$ from p to $p_{idle}$ , $C_{TB} \Leftarrow \{\neg do^{A}_{C_{targ}}\}$
```

action-type taskblock is activated. The procedure builds the remaining net by recursively walking back in the component net from places that output c_{targ} . The first of these recursive functions is BuildP($p_{G_{G_{sp\theta}}}$, p_{prev}, *Previsit*), where $p_{G_{G_{sp\theta}}}$ is the place in the component being considered, p_{prev} is the previously considered node in the taskblock on which we are building, and *Previsit* is a set of places in the component model that are not to be explored along this path, either because they satisfy the target condition or because they have already been visited in the algorithm. This procedure creates new transitions into p_{prev} in the taskblock that are conditioned on the output of p_{sys}. Thus, p_{prev} can only become marked when p_{sys} is marked. One of the new transitions comes directly from p_{init}, thus directing the taskblock marking to this place p_{prev} immediately after activation when p_{sys} is marked. The remaining new transitions each correspond to place preceding p_{sys} in the system net. This is very similar to action block generated for untimed plant with no progress confusion. however, there is one notable difference. The preceding p_{sys} places in the action block are decided on the basis of, if p_{sys} is a "Non-Deterministic" place or a "Deterministic" place. Deterministic place is given preference over nondeterministic place when "p_{sys}" is selected from system net. If there is no deterministic place to choose, the non-deterministic place in the system becomes p_{sys} and will have a corresponding transition in the taskblock. Lines 2,3,4 in procedure 4.6 determines this prioritization. The function BUILDT() is then called with each of these transitions. p_{sys} BUILDT($t_{G_{sp\theta}}$, t_{prev}, *Previsit*) is a second procedure, called recursively with BUILDP(). t_{sys} is a transition in the system that leads along a path (included among the places in *Previsit*) toward the target condition. p_{sys} is the place that inputs to t_{sys}. The procedure creates a place p_{prev} in the taskblock that outputs conditions that enable t_{sys} but disable other transitions leading from the place p_{sys}. Thus, when p_{sys} is marked in the system and p_{prev} is marked in the taskblock, the output of p_{prev} will only enable the firing of t_{sys}, thus allowing movement of system towards target. The recursion then continues, with a call to BUILDP() to build back

Figure 4.5: Action-type Taskblock BUILDP() function.

```
Procedure BUILDP $(p_{Gsp\theta}, p_{prev}, \textit{Previsit})$
    **Create** transition $t$ in $T_{TB}$ from $P_{init}$ to $P_{prev}$ with $C_{TB}(t) \Leftarrow C_{Gsp\theta}(P_{Gsp\theta})$
    **Define** $T_{inputset} \Leftarrow \{t_{Gsp\theta} \mid t_{Gsp\theta} \in^{(t)} p_{Gsp\theta}\}$
    **Define** $P_{inputset} \Leftarrow \{$ All $p_{Gsp\theta}$ input to each $t_{Gsp\theta} \in T_{inputset}\}$
    **If** $(P_{inputset} \cap NDset \neq P_{inputset})$ or $(P_{inputset} \cap NDset \neq \emptyset\}$
    $\{\ T_{inputset} = \{T_{inputset} - t \mid^{(p)} t \in NDset\}\ \}$
    **For** each $t_{Gsp\theta} \in T_{inputset}$ and $C_{Gsp\theta}(^{(p)}t_{Gsp\theta}) \neq C_{targ}$
        {
        **If** the place input to $t_{Gsp\theta}$ is not in *Previsit*
            **{ Create** transition $t$ in $T_{TB}$ into $p_{prev}$ with $C_{TB}(t) \Leftarrow C_{Gsp\theta}(p_{Gsp\theta})$
            BUILDT$(t_{Gsp\theta}, t, \textit{Previsit})$ **}**
        }
```

further from p_{sys} and p_{prev}.

The recursions finally end when each path cannot be extended further back without revisiting a place already considered. When all recursions are done. CREATEAB()then creates a transition from each of the places (except p_{init}, and p_{idle}) that will then move the marking to p_{idle} when $do^{A}_{C_{targ}}$ becomes false and taskblock is deactivated.

Figure 4.6: Action-type Taskblock BUILDT() function.

```
Procedure BUILDT($t_{Gsp\theta}, t_{prev}, Previsit$)
**Define** $C' \Leftarrow C_{Gsp\theta}(t_{Gsp\theta}) \cup \{\neg c | c \in C_{Gsp\theta}(t'_{Gsp\theta}) - C_{Gsp\theta}(t_{Gsp\theta})$
        for any $t'_{Gsp\theta}$ sharing an I/P place with $t_{Gsp\theta}$ }
**Create** place $p$ with $C_{TB}(p) \Leftarrow Act(C')$
connect $p$ into $t_{prev}$
**For** $p_{Gsp\theta}$ input to $t_{Gsp\theta}$
{
        $Previsit \Leftarrow Previsit \cup \{p_{Gsp\theta}\}$
        BUILDP($p_{G_{Gsp\theta}}, p, Previsit$)
}
```

Chapter 5

Example

5.1 Time condition system - example model

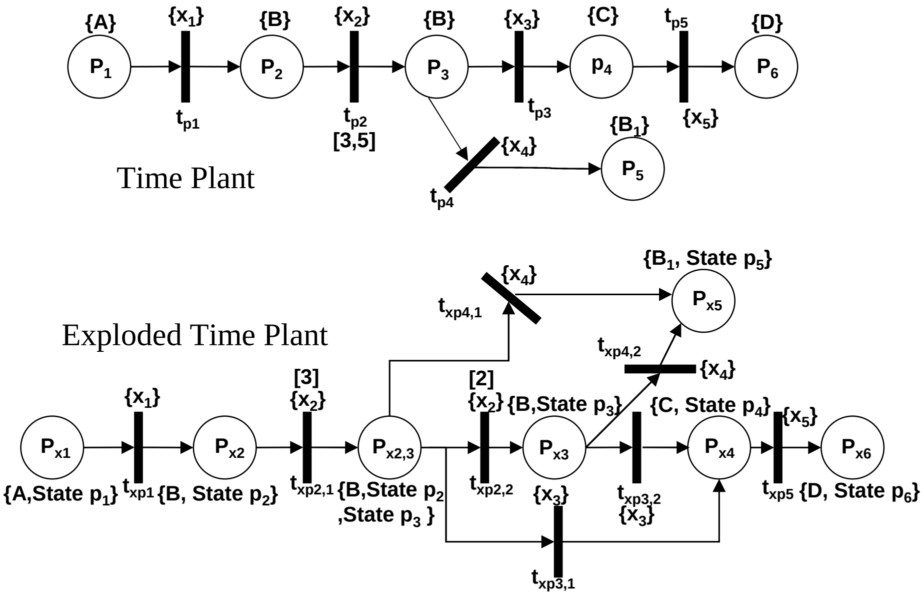

Figure 5.1: Example of Time Plant and Corresponding Exploded Time Plant.

An example is then used to illustrate the working of all the algorithms and procedures discussed in previous chapters. Figure 5.1 shows an example of a time plant with progress confusion, and its corresponding exploded time plant. The time plant in the above example figure has six places and five transitions, with each place indicating the state of the system. Places p_2 and p_3 have progress confusion, i.e, both these states have same observable output conditions $C_{G_{sys\theta}}(p_2) = C_{G_{sys\theta}}(p_3)$ $=\{B\}$. The "exploded time plant" in the above figure is generated using procedures in fig.2.3,2.5 in chapter 2. Lines 3, 4 of the procedure in fig.2.3 gives us the relation between the conditions on the places in Time plant and Exploded time plant. For $i = 1,2,....6$ let p_i indicate the place in the time plant that has corresponding place(s)in exploded time plant. Following are the conditions on the places in the exploded time plant

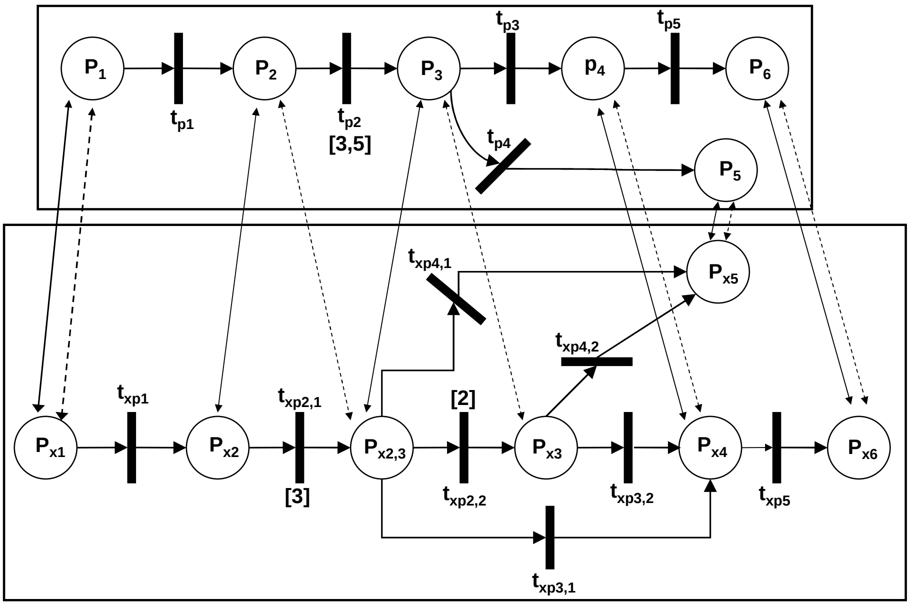

Figure 5.2: Place(s) Mapping.

$C_{G_{Exp\theta}}(p_{x1}) = \{A, \text{State } p_1\}$; $C_{G_{Exp\theta}}(p_{x2}) = \{\mathbf{B}, \text{State } p_2\}$, $C_{G_{Exp\theta}}(p_{x3}) = \{\mathbf{B}, \text{State } p_3\}$ and so on till

$C_{G_{Exp\theta}}(p_{x6}) = \{D, \text{State } p_6\}$

Figure 5.2 shows the correspondence mapping between the places in time plant and the exploded time plant. The mapping arrows (dual head solid and dotted arrows)connecting places in the time plant to those in exploded time plant show the pre-mapping and post-mapping relationship between the plants. For instance ${}^{\leftrightarrow}(p_1)=p_{x1}$, $(p_1)^{\leftrightarrow} = p_{x1}$

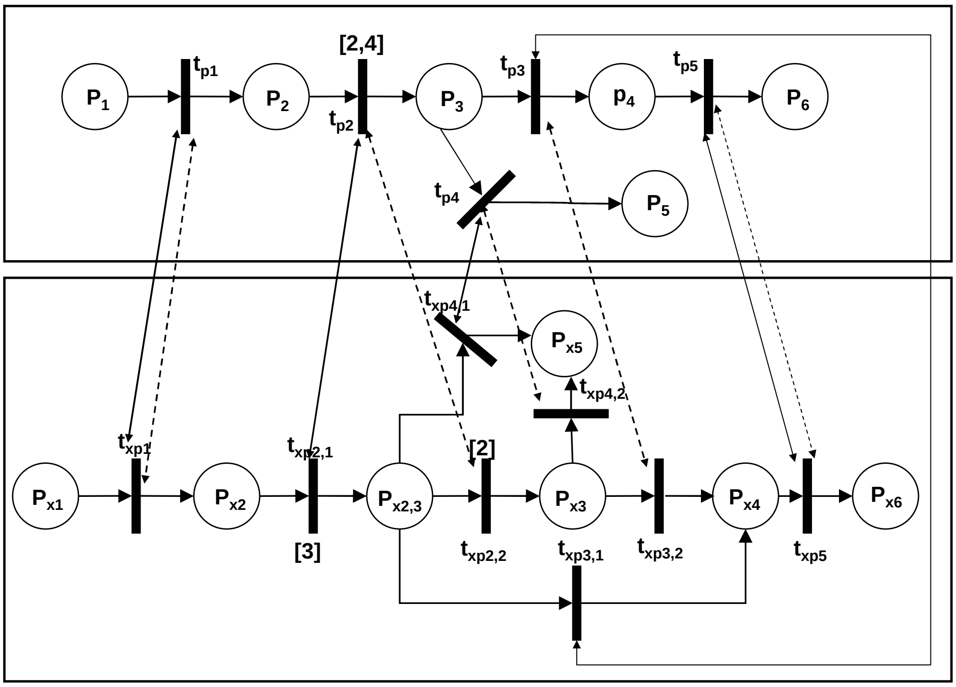

Figure 5.3: Transition(s) Mapping.

$^{\leftrightarrow}(p_2)=p_{x2},(p_2)^{\leftrightarrow}=p_{x2,3}$

$^{\leftrightarrow}(p_3)=p_{x2,3}$, $(p_3)^{\leftrightarrow}=p_{x3}$

Similarly, For $i = 1,...,5$ let t_{pi} indicate the transition in the time plant that have corresponding transition(s) in exploded time plant. Figure 5.1 shows the conditions assigned to transitions in time plant and the corresponding exploded time plant.

$C_{G_{Exp\theta}}(t_{xp1}) = C_{G_{sys\theta}}(t_{p1}) = X_1$, $C_{G_{Exp\theta}}(t_{xp3}) = C_{G_{sys\theta}}(t_{p3}) = X_3$ and so on till $C_{G_{Exp\theta}}(t_{xp5}) = C_{G_{sys\theta}}(t_{p5}) = X_5$

Figure 5.3 shows the correspondence mapping between the transitions in the time plant and the exploded time plant. The mapping arrows show the pre-mapping and post-mapping relationships between the plants. For instance

$^{\leftrightarrow}(t_{p1})=t_{xp1}$, $(t_{p1})^{\leftrightarrow}= t_{xp1}$

$^{\leftrightarrow}(t_{p2})=t_{xp2,1},(t_{p2})^{\leftrightarrow}=t_{xp2,2}$

$^{\leftrightarrow}(t_{p3})=t_{xp3,1}$, $(t_{p3})^{\leftrightarrow}=t_{xp3,2}$

Transition t_{p2} in the “Time plant" that belongs to “ProgConfTranSet$(G_{sys\theta})$" is replaced by $t_{xp2,1}p_{x2,3}t_{xp2,2}$ in the “Exploded time plant". $p_{x2,3}$ is the non-deterministic place that is created in the exploded time plant to solve the progress confusion problem in time plant. $C_{G_{Exp\theta}}(p_{x2,3}) = \{B, State\ p_2, State\ p_3\}$

Following conditions are true for $t_{xp2,1}, t_{xp2,2}$ in Exploded time plant

1. $C_{G_{Exp\theta}}(t_{xp2,1}) = C_{G_{Exp\theta}}(t_{xp2-2}) = C_{G_{sys\theta}}(t_{p2}) = X_2$ (Line 5,7 fig.2.5)

2. $\theta(t_{xp2,1}) = [3]$ (Line 6 fig.2.5)

3. $\theta(t_{xp2-2}) = [2]$ (Line 8 fig.2.5)

To maintain structural and temporal integrity of the system, all the child place(s) of p_{x2} and p_{x3} are also made child place(s) of $p_{x2,3}$ in the exploded time plant (fig.5.1); $\{p_{x2}^{(p)}\} \cup \{p_{x3}^{(p)}\} = \{p_{x2,3}^{(p)}\} = \{p_{x3}, p_{x4}, p_{x5}\}$ by creating duplicate output transitions from $p_{x2,3}$.

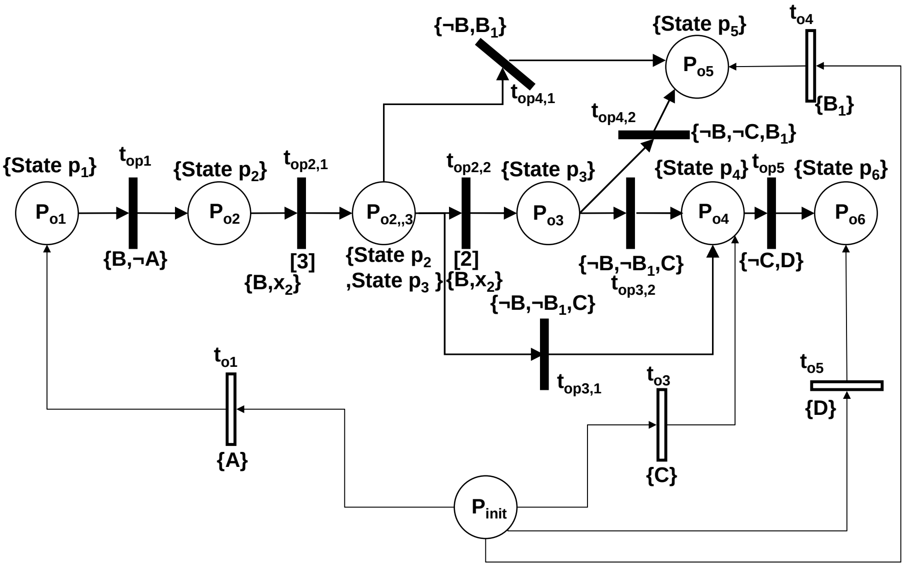

Figure 5.4: Example-Timed Observer .

Using the exploded time plant as input a timed observer is created for this plant. The algorithm in figure 3.2 is used to create the timed observer. This observer identifies the state of the exploded time plant at any instant of time and passes the information to the controller. Figure 5.4 illustrates the timed observer for the time plant in fig.5.1. In the first part of algorithm, a net structure, similar to the exploded time plant is copied. All the places in the observer are given state labels, $\{\mathtt{State}(P_i)\}$ extracted from corresponding places in exploded time plant. Now, the condition set for each of the transitions in the observer is determined (Lines 9-12 in procedure3.2).

Let's consider transitions t_{op1}, $t_{op2,1}$, $t_{op2,2}$

$$\begin{aligned} C_{Gobv\theta}(t_{op1}) =& (\{C_{G_{Exp\theta}}(p_{x2}) - \mathtt{StateLabel}\}) \cup F_\theta(p_{x1}t_{xp1}p_{x2}) \cup E_\theta(p_{x1}t_{xp1}p_{x2}) \cup \\ & H_\theta(p_{x1}t_{xp1}p_{x2}) \\ =& \{B\} \cup \{\neg A\} \cup \{\phi\} \cup \{\phi\} \\ =& \{B, \neg A\} \end{aligned}$$

$$\begin{aligned} C_{Gobv\theta}(t_{op2,1}) =& (\{C_{G_{Exp\theta}}(p_{x2,3}) - \mathtt{StateLabel}\}) \cup F_\theta(p_{x2}t_{xp2,1}p_{x2,3}) \cup \\ & E_\theta(p_{x2}t_{xp2,1}p_{x2,3}) \cup H_\theta(p_{x2}t_{xp2,1}p_{x2,3}) \cup C_{G_{Exp\theta}}(t_{xp2,1}) \\ =& \{B\} \cup \{\phi\} \cup \{\phi\} \cup \{\phi\} \cup \{x_2\} \\ =& \{B, x_2\} \end{aligned}$$

$$\begin{aligned} \theta(t_{op2,1}) =& \theta(t_{xp2,1}) \\ =& [3] \end{aligned}$$

Other transitions' conditions will be created by the algorithm in a similar manner. At the end, the algorithm creates an initial state and connects it to places corresponding to uniquely labeled plant states (refer fig.5.4)

In our next step towards generating controller for the time plant, we create an intermediate model that will be used to build the controller. This intermediate model is called "*Generated State label plant*". Figure 5.5 is the Generated State label plant for the time plant in fig.5.1. It is the replica of exploded time plant but with state labels as the only conditions on it's places. Generated state label plant satisfies all the system structure assumptions (SSA) for generating a controller from it (discussed in chapter 4). Following are the condition outputs of the generated state label plan

$C_{G_{Gsp\theta}}(p_{gs1})$ =

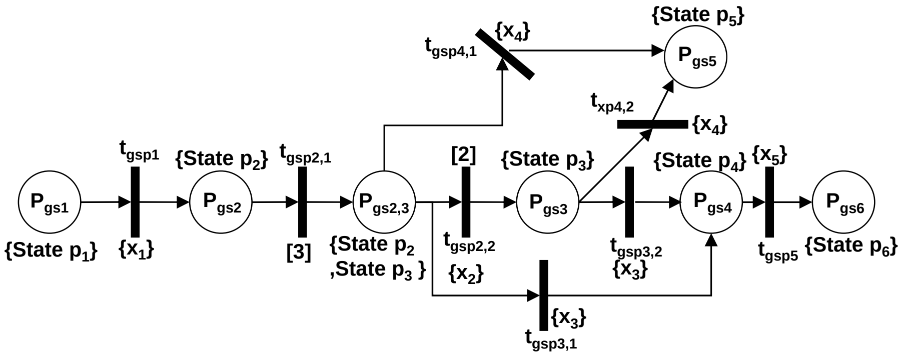

Figure 5.5: Generated State Label Plant Example.

$C_{G_{Gsp\theta}}(p_{gs2}) = \{\text{State } p_2\}$

$C_{G_{Gsp\theta}}(p_{gs2,3}) = \{\text{State } p_2,\text{State } p_3\}$

$C_{G_{Gsp\theta}}(p_{gs3}) = \{\text{State } p_3\}$ and so on till $C_{G_{Gsp\theta}}(p_{gs6}) = \{\text{State } p_6\}$.

Now say, if p_{x1} is marked in the exploded time plant, conditions $\{A, State p_1\}$ are true in $G_{Exp\theta}$. From our algorithm of generated state label plant, the corresponding place p_{gs1} in $G_{gsp\theta}$ becomes marked and condition $\{State p_1\}$ becomes true. Say after sometime if p_{x2} is marked in $G_{Exp\theta}$, conditions $\{B, State p_2\}$ become true, the corresponding place p_{gs2} in $G_{gsp\theta}$ becomes marked and condition $State p_2$ becomes true. This holds true for all the places in $G_{Exp\theta}$ and $G_{gsp\theta}$. There is a similar correspondence relationship between transitions of Exploded time plant and Generated state label plant. Say, if t_{x1} is enabled so is t_{gs1} and so on.

Finally a controller is synthesized from the generated state label plant, the controller interacts with the timed observer and generated state label plant to drive the plant to its target state. Figure 5.6 is the controller that drives the plant from its initial state to its target state, p_6. This is done by backtracking path from the

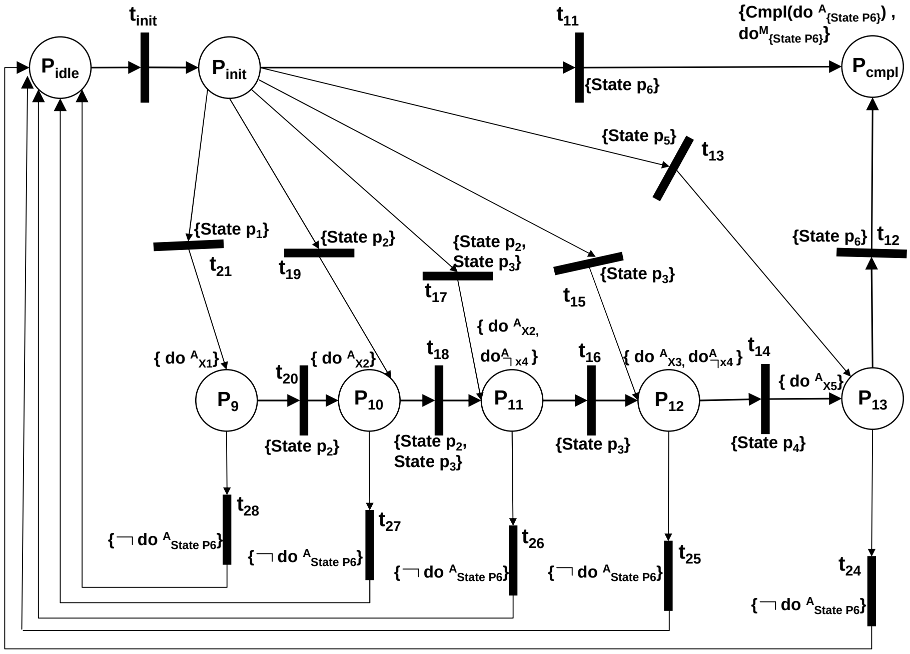

Figure 5.6: Example Controller.

target state to the initial state, p_{gs1}, and then driving the plant from current state to desired target state. This is done in a systematic manner. First the parent place of p_{gs6} is determined (i.e. p_{gs4}) and all the transitions that are output form p_{gs4} are disabled, except for one that leads to target place p_{gs6}. The conditions on the transition leading to p_{gs6} are made true. In this example p_{gs6} has no sibling place and has condition x_5 on its input transition, hence the condition $do^A_{x_5}$ would become true after the parent place p_{gs4} gets marked. So, to reach the target state p_{gs6}, we should first reach p_{gs4}. Now our intermediate target is to reach p_{gs4}. Again the same principle of disabling sibling transition from the parent place of target state is followed; in addition to that, the controller first looks at all the possible transitions that lead directly to place p_{gs4}. It then sees if the parent place(s) are deterministic place or a non-deterministic place, and chooses deterministic over non-deterministic.

Note 1: If there is more than one deterministic parent place, a random selection of a single place is done at first. The other places are explored in the later stages of con-

struction.

Note 2: If there is only a non-deterministic parent place to choose, then this place becomes the parent place.

p_{gs4} has p_{gs3} (deterministic place) and $p_{gs2,3}$ (Non-deterministic place) as its parent places. Following the algorithm p_{gs3} is chosen as the parent place and all the transitions that are input to sibling place(s) of p_{gs4} are disabled by negating condition(s) exclusive to those transitions. hence the condition $do_{x_3}^{A}$ would become true and the condition $do_{x_3}^{A}$ would become false after the parent place p_{gs3} gets marked. This back tracking continues till all the connected places are covered not more than once by the controller. Once the controller net is formed, the plant can be guided to any state from any state as long as there is an actual legal path in the plant. The timed observer assists the controller in guiding the plant by giving the current state information of the plant.

Therefore the controller, timed observer and the Generated state label plant work in a closed loop, guiding the plant to its target state.

Chapter 6

Conclusion and Future Research

6.1 Conclusion

This thesis presented the concept of timed condition systems, a new class of condition system that adds the notion of time to the condition system models (introduced by holloway et al., 2000 [Holl00]). Time condition systems relaxes the "no progress confusion" limitation that is present in all the previous research ([Holl01]).

An algorithm to develop a deterministic time plant (Exploded time plant) from a non-deterministic time plant has been discussed in chapter 2. We then synthesize a timed observer in chapter 3. The observer determines the current state of the plant by observing its condition outputs and state labels and gives this information to the controller. The controller synthesized in chapter 4, uses this information to drive the plant from its current state to the target state.

6.2 Future Research

There are many areas of potential future research in the time condition systems. The current thesis only deals with single layered condition systems. One of the areas of future research can be extending the time condition systems to multi-layer models. Time condition systems also have a potential of being used in the area of fault detection and monitoring. The timing information on transitions can be used as guards to detect faults occurred during unintended state changes. Other area of potential research can be exploring the usage of time in solving direction confusion problem.

Bibliography

[Holl00] Lawrence E. Holloway, Xiaoyi Guan, Ranganathan Sundaravadivelu, Jeff Ashley, Jr. Automated Synthesis and Composition of Taskblocks for Control of Manufacturing Systems. IEEE TRANSACTIONS ON SYSTEMS AND CYBERNETICS-PART B: CYBERNETICS, 30(5), October 2000.

[Holl01] Yu Gong, Lawrence E. Holloway State Observer Synthesis For a Class of Condition Systems. 5TH WORKSHOP ON DISCRETE EVENT SYSTEMS (WODES2000), GHENT, BELGIUM, August 2000.

[Holl02] Yu Gong and Lawrence E. Holloway Multi-layer State Observers for Condition Systems. EMERGING TECHNOLOGIES AND FACTORY AUTOMATION, 2001. PROCEEDINGS. 2001 8TH IEEE INTERNATIONAL CONFERENCE ON VOLUME 2, 15-18 OCT. 2001 PAGE(S):421 - 428 VOL.2

[Holl03] Lawrence E.Holloway, Xiaoyi Guan Assembling Taskblocks for Control of Discrete Event Systems. PROCEEDINGS OF 38TH IEEE CONFERENCE ON DECISION AND CONTROL, PHOENIX, ARIZONA, DECEMBER 1999, PP. 1749-1755.

[Holl04] Lawrence E.Holloway, Sundaravadivelu Ranganathan Taskblocks for Synthesis of Controllers for Condition Systems. 1998 IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS, SAN DIEGO, CALIFORNIA, OCTOBER 1998.

[Krogh00] R. Sreenivas and B. Krogh On Condition/Event Systems with Discrete State Realizations. DISCRETE EVENT DYNAMIC SYSTEMS: THEORY AND APPLICATIONS 1, KLUWER ACADEMIC, BOSTON, USA, 1991, PP. 209-236.

[Merlin00] P. Merlin A Study of Recoverability of Computer System. PHD THESIS, DEP.COMPUTER SCIENCE, MIT, CAMBRIDGE, Feb. 1974.

[Merlin01] P. Merlin, D.J. Farber Recoverability of Communication Protocols. IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. COM-24, NO.9, Sep. 1976.

[Aura00] Tuamos Aura, Johan Lilius Time Processes for Time Petri Nets. DIGITAL SYSTEMS LABORATORY, HELSINKI UNIVERSITY OF TECHNOLOGY,FIN-02150, ESPOO, FINLAND

[Diaz00] Marc Boyer and Michel Diaz Multiple Enabledness of Transitions in Petri Nets with Time. LAAS Ű CNRS 7, AVENUE DU COLONEL ROCHE 31077 TOULOUSE CEDEX 4 FRANCE, August 2000.

[Okawa00] Tomohiro Yoneda, Yasukichi Okawa Discrete Analysis of Time PetriNets. DEPARTMENT OF COMPUTER SCIENCE, TOKYO INSTITUTE OF TECHNOLOGY, 2-12-1 O-OKAYAMA MEGURO-KU TOKYO 152 JAPAN.

[Berth00] Berthomieu, B.; Menasche, M. An Enumerative Approach for Analyzing Time Petri Nets. DIFIP CONGRESS SERIES, VOL. 9, PAGES 41-46. ELSEVIER SCIENCE PUBL. COMP. (NORTH HOLLAND), 1983.

[Giua00] A. Giua Petri Net State Estimators Based on Event Observation. IEEE 36TH INT. CONF. ON DECISION AND CONTROL (SAN DIEGO, CALIFORNIA), PP. 4086-4091, DECEMBER, 1997.

[Giua01] A. Giua, C. Seatzu Observability of place/transition nets. IEEE TRANS. ON AUTOMATIC CONTROL, VOL. 47, NO. 9, PP. 1424-1437, SEPTEMBER, 2002.

[Ozv00] C. Ozveren and A. Willsky. Observability of discrete event dynamic systems. IEEE TRANSACTIONS ON AUTOMATIC CONTROL, 35(7):797–806, 1990.

[Sala00] Aguirre-Salas, L. Begovich, O. Ramirez-Trevino, A. State estimation in DES modeled by a class of interpreted Petri nets. INTELLIGENT CONTROL, 2002. PROCEEDINGS OF THE 2002 IEEE INTERNATIONAL SYMPOSIUM PAGE(S): 574- 579.

Vita

Date and place of birth

March - 15 - 1979, Andhra Pradesh, India.

Educational institutions attended and degrees awarded

Bachelor of Engineering, M.V.S.R Engineering College, Osmania University, Hyderabad, INDIA.

Major: Mechanical Engineering.

Experience

1. June 2004 - Current

   Lean Transformation Engineer, Affinia Under Vehicle Group, 101 industrial park drive, Stanford, KY.

2. August 2001 - December 2003

   Research Assistant, UK Center for Manufacturing, University of Kentucky.

Honors

1. August 2001. Awarded a full scholarship (Research Assistantship) by the Department of Manufacturing Engineering, University of Kentucky.