The development of hard real-time systems using a formal approach

Jaspal Singh Sagoo

Submitted for the degree of Doctor of Philosophy

THE UNIVERSITY OF ASTON IN BIRMINGHAM

September 1992

The University of Aston in Birmingham

The development of hard real-time systems using a formal approach

Jaspal Singh Sagoo

Submitted for the degree of Doctor of Philosophy
1992

Summary of thesis

Hard real-time systems are a class of computer control systems that must react to demands of their environment by providing 'correct' and timely responses. Since these systems are increasingly being used in systems with safety implications, it is crucial that they are designed and developed to operate in a correct manner. This thesis is concerned with developing formal techniques that allow the specification, verification and design of hard real-time systems.

Formal techniques for hard real-time systems must be capable of capturing the system's functional and performance requirements, and previous work has proposed a number of techniques which range from the mathematically intensive to those with some mathematical content. This thesis develops formal techniques that contain both an informal and a formal component because it is considered that the informality provides ease of understanding and the formality allows precise specification and verification. Specifically, the combination of Petri nets and temporal logic is considered for the specification and verification of hard real-time systems.

Approaches that combine Petri nets and temporal logic by allowing a consistent translation between each formalism are examined. Previously, such techniques have been applied to the formal analysis of concurrent systems. This thesis adapts these techniques for use in the modelling, design and formal analysis of hard real-time systems. The techniques are applied to the problem of specifying a controller for a high-speed manufacturing system. It is shown that they can be used to prove liveness and safety properties, including qualitative aspects of system performance. The problem of verifying quantitative real-time properties is addressed by developing a further technique which combines the formalisms of timed Petri nets and real-time temporal logic. A unifying feature of these techniques is the common temporal description of the Petri net.

A common problem with Petri net based techniques is the complexity problems associated with generating the reachability graph. This thesis addresses this problem by using concurrency sets to generate a partial reachability graph pertaining to a particular state. These sets also allows each state to be checked for the presence of inconsistencies and hazards.

The problem of designing a controller for the high-speed manufacturing system is also considered. The approach adopted involves the use of a model-based controller. This type of controller uses the Petri net models developed, thus preserving the properties already proven of the controller. It also contains a model of the physical system which is synchronised to the real application to provide timely responses. The various way of forming the synchronization between these processes is considered and the resulting nets are analysed using concurrency sets.

Key words: Hard real-time systems, safety-critical systems, formal methods, Petri nets, temporal logic, controller design.

Acknowledgements

I would like to express my sincere gratitude to all persons who have assisted me throughout this research programme. I am especially indebted to my supervisor Dr D. J. Holding for this guidance and support in all aspects of my research. I would also like to thank Dr G. F. Carpenter and my research colleagues, in the department of Electronic Engineering and Applied Physics, for their useful discussions and suggestions. A special thank you is due to my wife Mindy for being very patient and understanding and to my parents for their constant help and encouragement.

Finally, I would like to thank the Science and Engineering Research Council for funding me on a research studentship and as a Research Officer on grant GR/F/26584.

Table of contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Rapid advances in technology have created the desire to use real-time computer controlled systems in safety-critical applications. For instance, these systems are currently used in flight control systems, nuclear power plants and manufacturing systems; these applications require the control of on-going physical processes and, in some cases, the computer is required to operate with little or no human intervention. Although real-time systems have been used for a wide variety of applications, which have been operational for many years, no consensus on the definition of the term 'real-time' exists [Shin 87].

Real-time [Bennett 84] was originally introduced to distinguish between batch processing and interactive processing. In batch processing systems, the completion times of tasks are completely determined by the computer. However, interactive processing systems execute tasks as a result of the direct interaction between the computer and the 'real-world'. Computing systems have since evolved and typically real-time systems refer to a class of systems which must produce correct and timely responses to input stimuli; so that their environment is controlled in a predictable manner.

Despite the lack of a clear definition for real-time systems there is some agreement [Mok 83], [Shin 87], [Gorski 88] about their characteristic features, which distinguish them from other systems. For instance:

(i) time: a real-time system must respond to inputs from its environment within pre-defined time constraints. The consequences of producing a tardy response to an environmental stimulus has formed the basis of classifying these systems [Mok 83], [Shin 87], [Gorski 88], such as:

(a) soft real-time systems: in these systems the violation of response times does not lead to damage or cause catastrophic effects to occur in the environment; such systems can operate correctly if response times are occasionally violated. Examples of such systems include, airline reservation systems and computer controlled cash dispensers,

(b) hard real-time systems: in these systems the violation of response times may cause damage or destruction to their environment, such as in nuclear applications. Thus, hard real-time systems must not only provide correct and timely responses but must also ensure the safety of their environment. Various definitions for safety exist. For instance, there is the dictionary definition of safety, as used above, which is concerned with achieving accident-free situations, or safety that is related to the computations within a systems [Peterson 81], [Alpern 85], [Leveson 86]; these latter definitions will be examined in Chapter 2.

This thesis is essentially concerned with issues affecting the development of hard real-time systems,

(ii) environment: real-time systems must react to their environment by providing timely and correct responses to demands made upon them. Since these demands can occur periodically (occurring on a regular basis) as well as sporadically (where a minimum period is defined between the occurrence of two demands), the computer cannot predict *a priori* which resources it needs to provide the required service. Typically, this problem is handled by devising scheduling algorithms that ensures the system provides the required service. Generally, real-time systems may be required to control several environmental processes simultaneously, therefore these systems are concurrent and there is an increasing trend for them to become distributed [Gorski 88],

(iii) reliability: a real-time system must perform its intended function for a specified period of time under a set of environmental conditions [Leveson 86]. By and large, reliability requirements are concerned with making a system failure-free. This issue is particularly crucial for hard-real-time systems, in which failure could result in loss of human lives. Typically, it can be addressed by using formal methods for the specification and verification of these systems and structured design techniques to develop these systems.

It is important to emphasize that the above characteristics should not be considered in isolation [Shin 87]; instead there is a strong interplay between them. This is particularly manifested in hard real-time systems, in which timely and logically correct responses must be provided to control the environment in a specified and predictable manner. However, failure to provide such a service because of reliability problems, violation of timing specifications or otherwise, may lead to unsafe conditions arising.

Due to the nature of hard real-time systems the need to design safe systems is of critical importance and the above characteristics must be considered in the design and development of these systems. Although impressive hard real-time systems have been built, such as for space shuttle control systems [Carlow 84] and fighter plane control systems [Kaplan 85], current approaches for developing them have shown to be inadequate [Stankovic 88]. The deficiencies of current approaches are due to many factors, which include a general lack of awareness of the characteristics (see pp 11 - 12) of real-time systems, but mainly stem from the inability of existing techniques to adequately combine performance and functional requirements in the specification, design and implementation of these systems [Stankovic 88]. For example, currently it is not until the system has undergone costly simulations or rigorous testing that it is required to conform to performance requirements such as timing [Garman 81].

Research into hard real-time systems has addressed many aspects of systems development and has aimed to develop techniques and methods which ensure that these systems produce timely and logically correct responses to their environment. In particular, much research has been devoted to the areas of:

(i) scheduling,
(ii) programming languages,
(iii) specification and verification.

These areas will be examined in the following sections.

1.1.1 Scheduling

Scheduling for hard real-time systems addresses the problem of meeting specified timing constraints of periodic and aperiodic tasks, under resource limitations, according to some well-understood algorithm; so that the timing behaviour of the system is understandable, predictable and maintainable [Mok 84], [Burns 90].

Early research was restricted to scheduling tasks with hard real-time constraints on single processor and multiprocessor systems. For example, Liu and Layland [Liu 73] developed the rate monotonic algorithm for scheduling a set of periodic tasks on a single processor. This algorithm used a pre-emptive sheduler that allocated a priority to each task according to its period; the shorter the peroid, the higher the priority. Moreover, Liu and Layland [Liu 73] derived necessary and sufficient conditions for scheduling periodic tasks in which the deadline and period of tasks are equal. Blazewicz [Blazewicz 76] proposed an algorithm for a single processor system with a precedence relation over a set of sporadic tasks, whilst Garey and Johnson [Garey 77] provide algorithms for two processors with a precedence relation over a set of identical sporadic tasks.

Dhall and Liu [Dhall 78] consider the case of a set of independent periodic tasks on multiprocessors. Mok and Dertouzos [Mok 78] considered multiprocessor scheduling for a hard real-time environment. They showed that for a single processor system both the earliest deadline scheduling and the least laxity scheduling (where laxity represents the difference between the deadline of a task and its computation time) are optimal, however, neither schemes are optimal for multiprocessor systems. Moreover, they show that for two or more processors, no scheduling algorithm can be optimal without *a priori* knowledge of task deadlines, computation times and start times. Graham [Graham 79] argues that since optimal scheduling for a multiprocessing environment is NP-hard, then this result also applies to loosely coupled distributed systems.

The problem of scheduling loosely coupled distributed systems has been addressed by Ramamritham and Stankovic [Ramamritham 84], which uses suboptimal schedules and certain heuristics. Their approach caters for periodic and aperiodic tasks; the response times of periodic tasks are guaranteed at a particular site, however, if aperiodic tasks cannot be guaranteed at their site of origin then, they can migrate to other sites for their response times to be satisfied. This work is elaborated by Stankovic, Ramamritham and Cheng [Stankovic 85] which considers the relative loading of tasks at the sites of a distributed system. Specifically, they show that a system which contains an unbalanced loading of tasks at its sites enhances the performance of the algorithm, instead of a system with a balanced loading of tasks. Moreover, Stankovic, Ramamritham and Zhao [Stankovic 87] considers this algorithm under resource constraints. The use of this dynamic scheduling approach has been incorporated into the design of a real-time operating system referred to as the Spring Kernel.
The work of Leinbaugh [Leinbaugh 80] develops a hard real-time scheduling algorithm that uses a suboptimal schedule which is supplemented by procedures for estimating the upper bounds to the worst-case task completion times. This algorithm schedules tasks in a multiprocessor system and requires information about the device and resource requirements of each task and the cost of performing system functions. This work has also been extended for scheduling tasks in hard real-time distributed systems [Leinbaugh 86] This approach is useful at the system design level to determine statically the upper bounds of task response times. However, the limitations of this algorithm are: it does not consider

periodic tasks, it provides no means for guaranteeing that a new task will meet its deadline and this technique is computationally expensive.
It follows that current research [Audsley 92] considers scheduling based on the rate monotonic theory, originally developed by [Liu 73], to form the basis for constructing hard real-time systems comprising single processor systems. However, no equivalent results appear to be available for multiprocessor or distributed systems.

1.1.2 Programming

Programming hard real-time systems is primarily concerned with generating software that provides logically correct results within pre-defined time constraints [Wirth 77], and for providing support for real-time program development. The conventional approach for generating real-time software [Wirth 77], [Berry 82] was to write a logically correct concurrent program, and then to augmented it with scheduling primitives, to satisfy real-time constraints. However, this approach was inadequate because:

(i) real-time constraints were considered as a secondary issue,
(ii) there are distinct differences between concurrent and real-time programming; specifically the former was developed for the efficient utilisation of the underlying computer hardware, whilst the latter is primarily concerned with producing logically correct results within a pre-defined time constraint [Gligor 83].

More recent approaches concern the development of programming languages which include constructs for expressing timing constraints [Stankovic 88]. Many existing real-time programming languages [Stotts 82], [DOD 83], [Burns 87] and approaches to programming these systems [Wirth 77], [Gligor 83] have shown to be inadequate. For instance, there are insufficient constructs for specifying timing constraints in programs [Berry 82], [Volz 87] and the iterative method of trial and error is used for real-time program development [Lee 85]. These problems have been addressed by research aimed at developing programming language constructs that allow the specification of timing constraints and providing provisions for missed deadlines [Lee 85], [Dasarathy 85], [Lee 87]. Much of this work is based on the use of constructs known as Temporal Scopes, which allow the specification of time constraints that define the minimum time interval between two events, the maximum time interval between two events and the time interval during which an event may occur. There are three kinds of temporal scopes:

(i) global temporal scopes, that encapsulate a whole process and is used to define a periodic process,
(ii) local temporal scopes, which specify timing constraints within a process,
(iii) communication temporal scopes, which specify timing constraints associated with interprocess communication.

These temporal scopes have been used in the design of a predictable real-time kernel for distributed systems [Lee 89] and allow the programmer to specify timing constraints for process execution and interprocess communication. The kernel uses these timing constraints both for process scheduling and for scheduling communications.

Shaw [Shaw 89] presents an alternative approach that proposes to predict the timing behaviour of high-level programming language constructs. More specifically, this approach uses the Hoare logic [Hoare 69] to estimate the upper and lower bounds on the execution times of simple sequential and time related constructs, as opposed to calculating exact execution times. Whilst this technique may be suitable for simple constructs, it becomes complicated for large program segments, because details such as compiler optimization, resource allocation strategies and target machine details need to be known. Hence, much research is needed in this approach to combine theoretical techniques with experimental results to produce accurate estimations.

1.1.3 Specification and verification

The specification and verification of hard real-time systems addresses the problem of combining functional and performance requirements into specification techniques and provides means of proving the 'correctness' of the resulting specification [Stankovic 88]. Traditional approaches [Thayse 89] formulated specifications using natural languages, however, these specifications suffered from deficiencies inherent in natural languages, such as ambiguity, imprecision and the inability to verify. Alternative approaches based on structured methods produce precise specifications, although they lack the mathematical structure necessary to verify the 'correctness' of such specifications [Jackson 83], [Ratcliff 87]. However, structured methods have been adapted for hard real-time systems [Ward 86], [Hatley 87], [Harel 90] and are widely used in industry. Approaches using a mathematical notation to precisely capture user requirements and then to provide verification have attracted considerable attention and research [Froome 88], [MOD 89], [IEC 89]. These approaches are known as formal methods or techniques and aim to provide a rigorous approach to specification. Formal methods or formal techniques for the specification and verification of systems are based on concepts peculiar to discrete mathematics [Stanat 77] - typically propositional or predicate logic, and set theory - which allow the formulation of precise specifications that can be verified using their mathematical framework [Wing 90]. However, the requirement of formal methods for specifying performance constraints (such as timing) as well as the functionality of hard real-time systems has produced a fundamental challenge of incorporating time into such techniques [Shin 87], [Hoogeboom 91]. In particular, it is not clear how to specify time constraints, or how to extend existing techniques with time [Shin 85], [Shin 87], [Stankovic 88]. Various approaches have been proposed that extend existing techniques with the notion of time [Pnueli 88], [Reed 88], [Gorski 88], [Davis 88], [Joseph 89], [Scholefield 90] but much of this work is still under investigation. However, two types of approach are emerging from this research [Zave 88]. The first comprises a classical formal approach which is essentially mathematically intensive and much work may be required to develop these techniques so that they are usable and acceptable. The second approach involves the use of executable specifications which allows user requirements to be captured using a specification language, which when executed, within its software environment, shows the execution of the specification. This approach provides a direct means of validating the specification (where validation involves showing that the specification conforms to user requirements) in which the user, who is usually unfamiliar with the technology used, can see a prototype of the proposed system 'in action'. Hence, one of the most significant advantages of this technique is that a close interaction is formed between the user and the systems analyst. However, executable specifications cannot formally verify specifications and for this to be achieved they need to be used with formal techniques [Zave 86].

The research presented in this thesis is conducted in the area of formal specification and verification, and is concerned with the development of these techniques for the design, specification and verification of hard real-time systems. The approach adopted in this thesis is essentially based on classical formal methods rather than executable specifications. However, in order to introduce a more tangible approach, a graphically based technique is used for the specification capture phase, which is then translated into a mathematical formalism to allow for formal analysis and verification.

1.2 Aims and objectives of this research

This research is concerned with the initial activities of a real-time system design life cycle, such as specification capture, verification, hazard analysis and design. This work forms the initial stages of a broader research programme aimed at developing and applying techniques to various aspects of the real-time system's life cycle. Hence, the aim of this research is to investigate and develop techniques that are suitable for the specification of the functional and performance features and the verification of hard real-time systems.

Generally, formal techniques proposed for hard real-time systems - such as techniques derived from CSP [Hoare 85] and temporal logic [Pnueli 77] - require a high level of mathematical skill for formal specification and verification, so this aspect can lead to difficulties in the formal development of systems [Hall 90]. Hence, this thesis is particularly motivated by the desire to develop formal techniques that integrate an informal and formal component, instead of techniques that are mathematically intensive. This approach is taken because it is considered that the resulting technique could utilise the informal and formal approaches in a complementary manner [Blyth 90]; the informality providing ease of understanding and the formality allowing precise specification and verification.

This thesis considers informal approaches based on graphical formalisms to be worthy of investigation, rather than using natural languages, because they provide a visual representation that can be both intuitive and easily comprehensible. Most graphical formalisms include analysis tools which can be used to determine the consistency and behavioural aspects of the system being modelled [Peterson 81], [Harel 87], [Gabrielian 88]. However, most graphical formalisms are unsuitable for formal reasoning about a system or for formally proving system properties because they lack the necessary proof system. Therefore, it is required either to augment the graphical formalism by a separate and consistent formal logic or to translate the graphical formalism into a formal logic.
The formal approaches considered in this thesis are those that include the necessary mathematical framework for specification and verification of time related properties. Hence this research:

(i) evaluates existing formal techniques,
(ii) extends selected techniques (which satisfy the above motivation) so that they are applicable to the specification, verification and design of hard real-time systems.

The evaluation of existing techniques was carried out in two stages, by:

(i) a comparative survey,

(ii) application of techniques, selected from the comparative survey, to the modelling, design and analysis of a hard real-time control problem.

The comparative survey fulfilled the purposes of examining the state of the art notations used to describe hard real-time systems, and allowing the comparison of each technique's ability to specify and verify these systems. The techniques selected from the comparative survey are Petri nets [Peterson 81] and temporal logic [Pnueli 77]. The formalism of Petri nets captures the causal properties of a system within a graphical representation that is both intuitive and simple understand. However, these nets do not contain the notion of time and cannot verify properties, hence they are unsuitable for formally specifying and verifying hard real-time systems. Temporal logic is an 'extension' of modal logic [Chellas 80] and is capable of specifying and verifying the temporal behaviour of concurrent systems. Thus, this research considers techniques based on the combination of Petri nets and temporal logic to contain both a formal and informal component, and has the necessary attributes for modelling hard real-time systems.

The analysis performed on the Petri net models is invariably centred on generating the reachability graph. However, the generation of the total reachability graph can become a difficult task as the size of the net increases. Thus in order to reduce this complexity this thesis uses the approach of generating a partial reachability graph pertaining to a particular system state. This is achieved using the concurrency sets developed by Skeen and Stonebraker [Skeen 83]. Since these sets were originally used to analyse the recoverability and check the consistency of commit protocols, in distributed databases in the presence of failures, their application in this thesis allows similar consistency checks to be performed on the Petri net models as well as the generation of reachability graph information.

The ability of such techniques to model and analyse a hard real-time control problem is assessed. In particular, this problem comprised the design of control and synchronization logic for the motion of two independently-driven interacting mechanisms, which form part of a high-speed packaging machine.

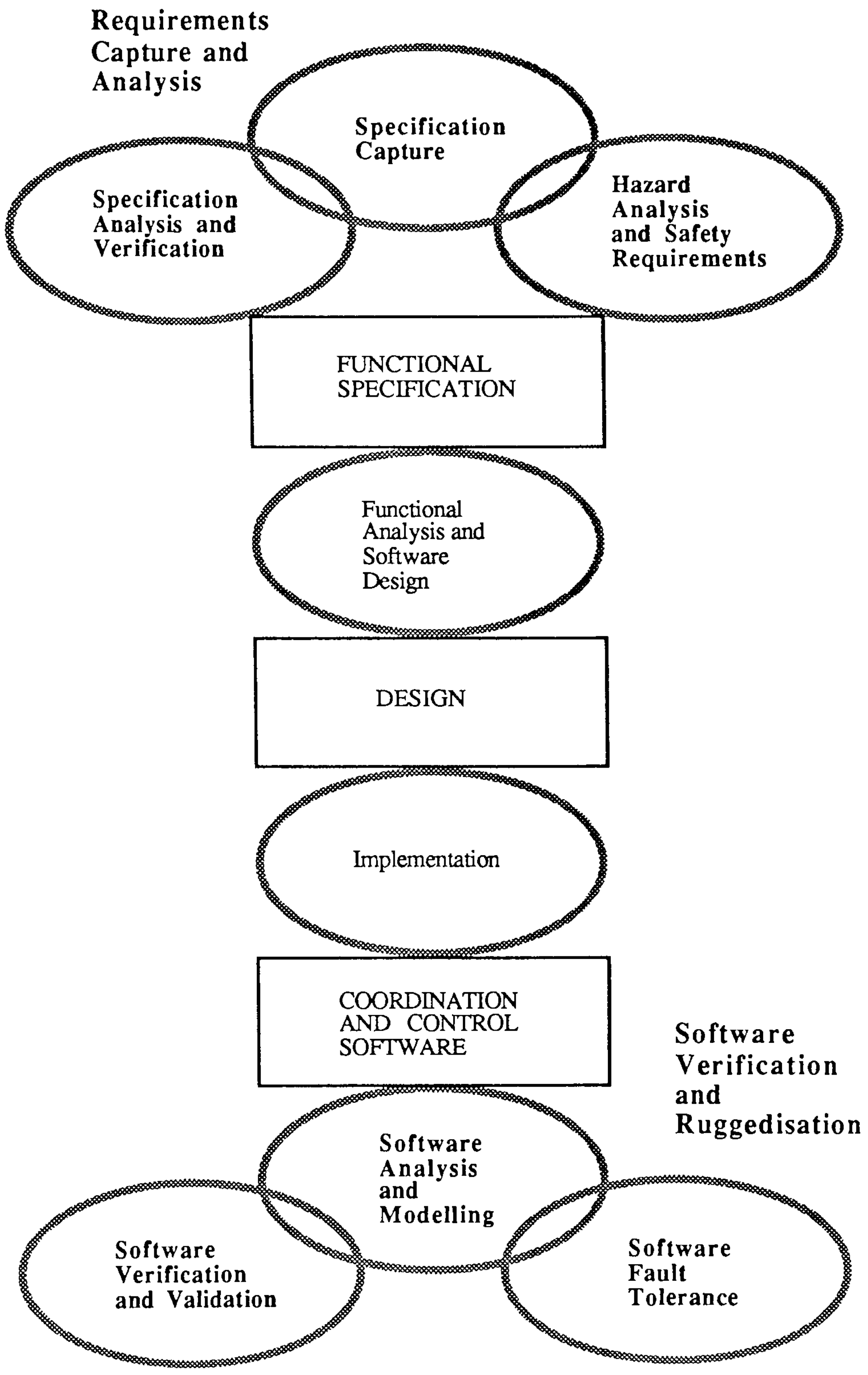
The issue of designing a controller for the above hard real-time control problem is also addressed. Specifically, a strategy is proposed that divides the system into a controller and the physical system, in which the controller contains a model of the physical system. The task of synchronising the model of the controller and the physical system is an important consideration in this strategy. This stems from the need for the controller to provide a timely and correct response to the physical system in order to coordinate the mechanisms in

the physical system in a safe manner. The analysis of the resulting controller/physical system Petri net is performed using concurrency sets by checking for the presence of hazardous states, inconsistent states and hazardous sequences of states (achieved by generating partial reachability graphs).
The applicability of the above techniques and approaches can be illustrated by examining their role in the life cycle of hard real-time systems. The particular type of life cycle for hard real-time systems used in this thesis is shown in Figure 1.1, where the oval shapes represent activities that yield a product, which is shown as the rectangular shapes. The life of a system commences with requirements capture and analysis which involves consultations with the user concerning the system that is to be produced. The process of specification capture models or details what function is to be performed by the system using the requirement specification.

This specification is subjected to various forms of analysis and verification procedures. The aim of these activities is to ascertain whether the specification is correct and conforms to user requirements. Safety requirements, which can involve performance requirements, details the conditions or events which must occur in the system. Hazard analysis is used to detect for the presence of undesirable conditions in the specification and to prevent their occurrence. Successful completion of the above activities allows the formulation of the functional specification. The activities of functional analysis and software design separates the functional specification into hardware and software modules. Thus hardware architectures and software algorithms are defined which satisfies the specification; moreover a suitable approach is selected from which to form the design. The implementation is a realization of the design and actually produces the software and any related hardware. The activities of software verification and ruggedisation involves software verification and validation, software analysis and modelling, and software fault tolerance - which assists in securing the system.

The techniques developed in this thesis are particularly relevant to the formal specification and verification of systems, hence their main area of applicability on Figure 1.1 is the overlap between specification analysis and verification, specification capture, hazard analysis and safety requirements. Although this thesis is also concerned with the design of controllers (which is a phase applicable further down the life cycle), the main emphasis is on formal specification and verification, and hazard analysis.

Figure 1.1 Design life cycle for hard real-time systems


Requirements Capture and Analysis
Specification Capture
Specification Analysis and Verification
Hazard Analysis and Safety Requirements
FUNCTIONAL SPECIFICATION
Functional Analysis and Software Design
DESIGN
Implementation
COORDINATION AND CONTROL SOFTWARE
Software Verification and Ruggedisation
Software Analysis and Modelling
Software Verification and Validation
Software Fault Tolerance

1.3 Summary of thesis

Chapter 2 presents a comparative survey of formal techniques that are considered to be suitable for the specification and verification of hard real-time systems; the techniques examined basically model systems in terms of discrete-events. For formal techniques to be applicable to hard real-time systems they must be capable of describing both the functional and performance requirements of these systems; this means that such techniques must be capable describing features such as the functionality of the system, timing constraints and concurrency. The survey of Chapter 2 evaluates each technique according to their ability to describe these features. Since the aim of this thesis is to develop techniques that contain an informal and formal component, the combination of techniques such as Petri nets and temporal logic is deemed to be suitable for further investigation because:

(i) Petri nets have a graphical representation, but cannot be used to formally verify properties [Koymans 85],
(ii) temporal logic is a formal language that is suitable for reasoning about the temporal behaviour of concurrent systems.

Hence the combination of these techniques will yield a technique that complements each constituent technique. Specifically, it integrates a graphical and formal approach which has the potential attributes for describing hard real-time systems.

Chapter 3 presents a survey of recent works which combine the formalisms of Petri nets and temporal logic. This survey shows that the temporal Petri nets of Suzuki and Lu [Suzuki 89] and the approach proposed by He and Lee [He 90] are worthy of further investigation because they allow a consistent translation between Petri nets and temporal logic. Each technique is examined for its ability to model the classical mutual exclusion problem. This application shows the attributes and limitations of each technique; the appropriate modifications and extensions are suggested. Specifically, temporal Petri nets are enhanced with a proposition that allows a wider range of Petri nets to be formally analysed. The approach of He and Lee, which previously combined first order temporal logic and high-level Petri nets [Peterson 81], is adapted to combine temporal logic and low-level Petri net [Peterson 81]. Thus this extended temporal Petri net allows a different class of systems to be modelled as well as allowing the analysis associated with low-level Petri nets to be used.

Chapter 4 considers the application of temporal Petri nets, extended temporal Petri nets and timed Petri nets [Ramachandani 74], [Razouk 85] to the modelling and formal analysis of a real-time control problem. Previously, temporal and extended temporal Petri nets have only been applied for the analysis of concurrent systems [Suzuki 89], [He 90], however, this chapter considers their application to hard real-time systems. The type of analysis performed by each technique is demonstrated and compared.

Chapter 5 concentrates on the inability of temporal and extended temporal Petri nets to prove real-time properties; this is a serious handicap for modelling hard real-time systems. Hence, this chapter proposes a novel technique that combines the timed Petri nets of [Razouk 85], examined in Chapters 2 and 4, and the real-time temporal logic of [Ostroff 89]. These real-time temporal Petri nets are shown to have the ability to specify and verify both real-time and non real-time properties. Specifically, timed Petri nets are primarily used to specify quantitative time values on the net structures and real-time temporal logic allows the formal specification and verification of real-time properties, which involve quantitative time values. This real-time temporal Petri net is formally defined and subsequently applied to the Petri net model of the real-time control problem presented in Chapter 4.

Chapter 6 recognises the importance of reachability graphs in the analysis of timed Petri nets, temporal Petri nets, extended temporal Petri nets and real-time temporal Petri nets. It also recognises the difficulty involved in generating the total reachability graph. Hence, this chapter aims to reduce the complexity in generating reachability graphs by proposing a technique that allows the generation of a partial reachability graph associated with a particular system state. Specifically, the concurrency sets of Skeen and Stonebraker [Skeen 83] are applied to the Petri net models shown in Chapter 4. Application of this technique shows that it is useful for generating partial reachability graphs as well as for inspecting the semantics of the reachable markings for the presence of hazardous and inconsistent states. A useful subset of concurrency sets is developed, which has the attributes of allowing all the unique states of the net to be identified and thus inspected for hazards. This minimum concurrency set can also be used to provide an alternative method of generating the total reachability graph of the net.

Chapter 7 establishes the need to develop a controller for the Petri net models of the real-time control problem considered in Chapter 4. A strategy is proposed for synthesizing a controller which produces a Petri net model that contains a representation of the physical system, the controller and interfaces that link the controller and physical system. The controller comprises the Petri net model developed in Chapter 4, which is linked to the physical system; thus the controller executes a synchronised model of the physical system. The linking of the physical system and the controller can be achieved in several ways and they are examined in this chapter. The controller/physical system Petri nets formed using the above strategy are analysed and assessed using techniques based on concurrency sets.

Chapter 8 attempts to derive the formal base common to all the techniques developed and proposed in this thesis, such as temporal Petri nets, extended temporal Petri nets, real-time temporal Petri nets. In particular, it is noted that temporal Petri nets, extended temporal Petri nets and real-time temporal Petri nets are used to verify either real-time or non real-time properties; hence this chapter examines the axioms fundamental to each technique. This examination shows that the axioms of real-time temporal Petri nets can be used to derive both temporal and extended temporal Petri nets, hence the former can be used to perform the same analysis as the latter techniques. Thus using the theory of real-time temporal Petri nets, in conjunction with concurrency sets, used in Chapters 6 and 7, the formal analysis of a controller/physical system Petri net selected from Chapter 7 is performed.

Finally in Chapter 9, the research is concluded by briefly comparing the various attributes of the Petri net based techniques developed. Details of further work are also discussed.

Chapter 2

Survey

2.1 Introduction

This chapter presents a survey of techniques that are suitable for modelling real-time systems in terms of discrete-event systems. In particular, it examines the formal models and notations used to describe these systems and concentrates on the fields of Petri nets, temporal logic, the theory of communicating sequential processes, VDM, Z, and executable specification languages such as PAISLey; various forms of this survey have appeared in [Sagoo 90], [Sagoo 92a], [Holding 92]. However, in order to present a clear and consistent survey it is necessary to discuss the definitions of some of the terms used to describe the behaviour of real-time processes, such as interleaving, safety, liveness and fairness, because they have different interpretations in other areas of research.

2.1.1 Definition and discussion of terms

Discrete event processes can be represented by a sequence of distinct atomic activities or actions; the execution of concurrent processes can be modelled using either maximum parallelism or interleaving [Pnueli 88]. Maximum parallelism allows processes to execute their atomic actions simultaneously; thus two processes are never both waiting to achieve a shared communication. The interleaving model represents the execution of the concurrent processes by interleaving their atomic actions; thus at any instant only one atomic action can be executing. This model has been criticised in [Jahanian 86], [Pnueli 88] for inadequately modelling concurrency. However, because it allows only one atomic action to execute at any instant, fewer states are considered compared to the maximum parallelism model (where at any instant many actions can execute). This aspect of the interleaving model leads to a simpler analysis and as a consequence it is widely used. Hence, this thesis uses the interleaving model to describe concurrent systems.

The terms safety and liveness are used in the analysis of the properties of concurrent systems. Unfortunately, conflicting definitions occur in formal methods [Lamport 77] and Petri net theory [Peterson 81]. The definitions used in this research are derived from the former, since they correspond more directly with the use of the term 'safety' in engineering and general contexts. Safety and liveness properties were introduced by Lamport [Lamport

77] and later they were formally and informally defined by Alpern and Schneider [Alpern 85]. Since the formal definition of these properties were shown to be non-trivial [Alpern 85] this research will use the informal definitions.

A safety property is informally defined as saying that a 'bad thing' does not happen during execution of a program or system; formally it is defined in [Alpern 85]. An example of a safety property is mutual exclusion, in which the bad thing is two processes using a shared resource simultaneously. Thus, safety properties specify what may or may not happen during the execution of the system. Alternatively, Leveson [Leveson 86] considers safety as a value which is defined as the probability that conditions which can lead to an accident (or hazard) do not occur whether the system performs the intended function or not. This definition emphasizes the conditions prevailing in the environment, as a consequence of the operation of the system, rather than what occurs within the system. The definition of a safety property used in this thesis is derived from [Alpern 85] which views the 'bad thing' as the system state that causes an accident. Thus a 'safe' system is one which satisfies the specified safety properties. This definition combines features of [Alpern 85] and [Leveson 86] that are useful from an engineering perspective.

A liveness property is defined as saying that a 'good thing' happens during execution of a program or system; a formal definition is shown in [Alpern 85]. An example of a liveness property is termination which asserts that a program does not run forever; the 'good thing' being the completion of the final instruction.
Thus liveness properties specify what events or actions will occur in the modelled system. A concept that is closely related to liveness is fairness [Queille 83], which is a generic name for a multitude of concepts used in different contexts [Francez 86]. For example, repetitive choice among alternatives (in sequential programs); freedom from starvation in accessing resources by concurrent processes; preventing a process from executing forever whilst other executable concurrent processes do not execute, in a system modelled by interleaving. However, in the context of concurrent systems, various grades of fairness are defined, ie strong fairness (referred to as fairness) and weak fairness (or justice) [Lehman 81]. For example, a process is fair if, given an infinite execution, every optional computation is executed a number of times proportional to the number of times the choice is enabled, whereas, a process is just if, for every option that is infinitely enabled (assuming an infinite computation), that computation is chosen at least once.

2.2 Petri nets

Petri nets [Petri 62], [Peterson 81], [Reisig 85] are bi-partite directed graphs that are based on an extension of set theory known as bag theory, which allows multiple occurrences of the same element in a set. These nets are a powerful means of modelling concurrency and non-determinism, and are useful for capturing the causal properties and the control flow in systems. Petri nets consist of two types of node (known as places and transitions) and arcs that interconnect them. These nets have two different, but equivalent, representations: a formal structure and a graphical representation. An example of each is shown below:

(i) Formal structure of Petri nets

Many formal definitions of Petri nets exist [Peterson 81], all of which are equivalent. However, the definition used in this thesis follows that of Peterson [Peterson 81] and considers the structure of a marked Petri net (C) as the tuple:

$$C = \{P,T,I,O,M_0\} \qquad 2.1$$

where

P is a set of finite places: $P = \{p_1,p_2, \ldots , p_n\}$ where $n > 0$,

T is a set of finite transitions: $T = \{t_1,t_2, \ldots , t_m\}$ where $m > 0$,

I is the input function that maps a transition to places defined as:

$$I: T \rightarrow P^\infty$$

where

P^∞ represents the set of all strings made up of elements of P.

A $p_i \in P$ is known as an input place of $t_j \in T$, iff $p_i \in I(t_j)$, ie. p_i is connected to t_j by an arc,

O is the output function that maps a transition to places defined as:

$$O: T \rightarrow P^\infty$$

A $p_i \in P$ is an output place of $t_j \in T$ iff $p_i \in O(t_j)$, ie. an arc connects t_j to p_i,

M_0 is the initial state of the net and is referred to as the initial marking. Any net marking M reachable from M_0 is essentially a distribution of tokens to selected places in the net and can be formally defined as the function:

$$M: P \rightarrow \{0,1,2, \ldots ,m\}.$$

The marking in C changes when a transition becomes enabled and eventually fires. For a specific marking, several transitions can become enabled, however, using the interleaving model to represent concurrency only one transition is allowed to fire at any instant. The conditions which cause the enabling and firing of transitions are defined as follows:

(a) Transition enabling:
for C in which marking M is reachable from M_0, $t_j \in T$ becomes enabled iff for all $p_i \in P$

$$M(p_i) \geq \#(p_i, \mathcal{I}(t_j)) \qquad 2.2$$

where
$M(p_i)$ represents the number of tokens in p_i under marking M,
$\#(p_i, \mathcal{I}(t_j))$ represents the number of occurrences of p_i in the input function $\mathcal{I}(t_j)$.

(b) Transition firing:
for C in which marking M is reachable from M_0, if $t_j \in T$ becomes enabled then it *may* fire; t_j fires instantaneously and produces the new marking M' given by:

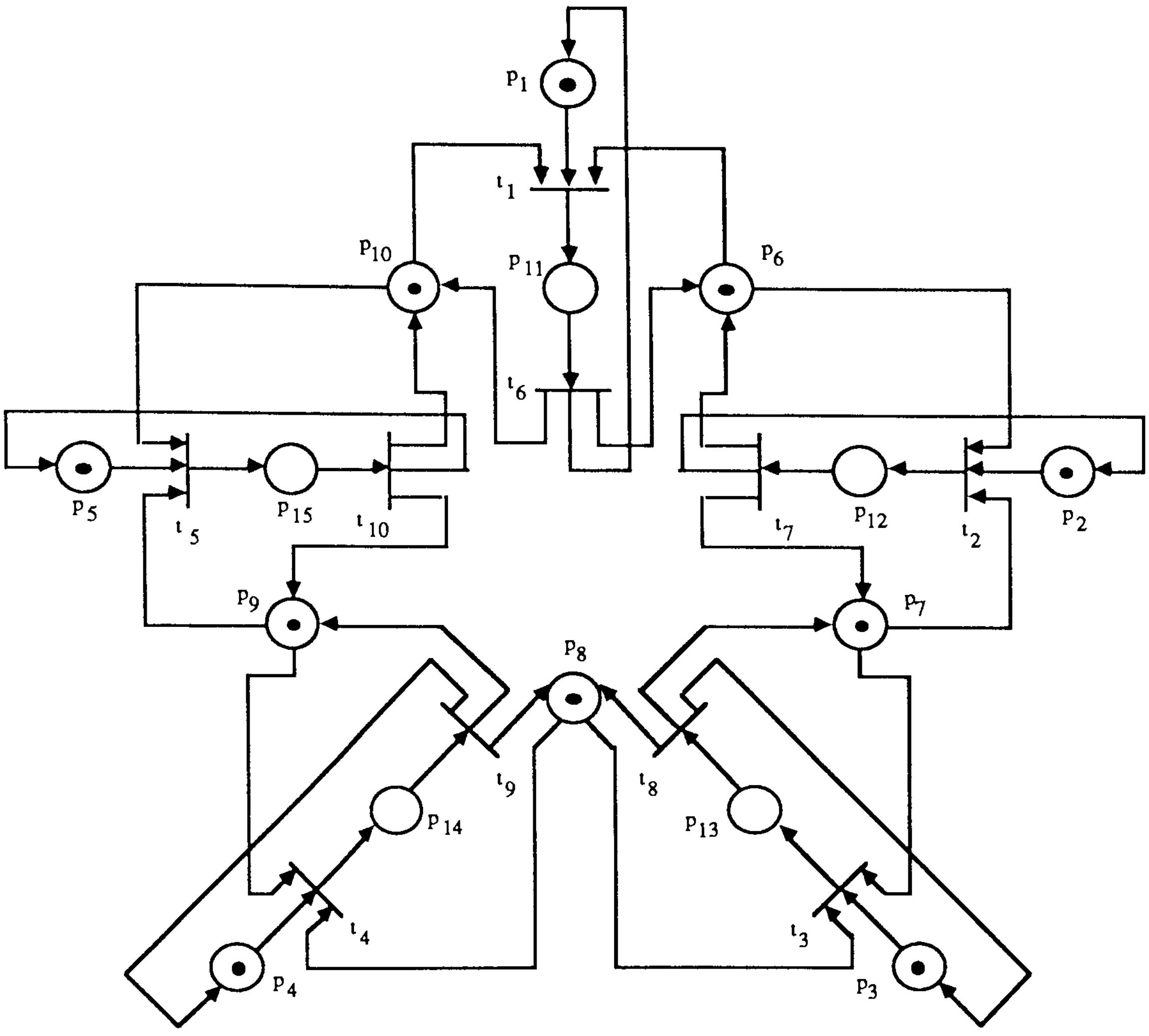
$$M'(p_i) = M(p_i) - \#(p_i, \mathcal{I}(t_j)) + \#(p_i, \mathcal{O}(t_j)) \qquad 2.3$$

where
$\#(p_i, \mathcal{O}(t_j))$ means the number of occurrences of p_i in the output function $\mathcal{O}(t_j)$.

(ii) Graphical representation

In its graphical form [Reisig 85] the structure of a Petri net is illustrated in the following manner:
a circle 'O' represents a place, a bar '|' represents a transition,
an arrow represents an arc which connects a place to a transition, or vice versa,
a black dot '•', which resides in a circle, represents a token.
Figure 2.1 shows a Petri net in its graphical form. This net, which was proposed by Peterson [Peterson 81], models the well-known dining philosophers problem consisting of five philosophers. In this problem each philosopher can either be thinking or eating; for eating purposes each philosopher requires two chopsticks. However, problems arise when philosophers attempt to eat because five chopsticks must be shared amongst five philosophers. This problem is represented by Figure 2.1 in the following manner.

Figure 2.1 A Petri net (low-level) representation of the dining philosophers problem



The semantics assigned to the places and transitions of Figure 2.1 are:

p_1 to p_5 represent the condition that philosophers are thinking,

p_6 to p_{10} represent available chopsticks,

p_{11} to p_{15} represent philosophers eating,

t_1 to t_5 stand for the actions of picking up chopsticks and

t_6 to t_{10} stand for actions of putting down chopsticks.

The marking shown in Figure 2.1 represents the state when all philosophers are thinking, and they have the potential to start eating in subsequent markings, denoted by the enabling of transitions t_1 to t_5. However, only the combination of philosophers represented by the following places can actually eat:

p_{11} and p_{13}, if t_1 and t_3 are fired,

p_{11} and p_{14}, if t_1 and t_4 are fired,

p_{12} and p_{14}, if t_2 and t_4 are fired,

p_{12} and p_{15}, if t_2 and t_5 are fired, or

p_{13} and p_{15}, if t_3 and t_5 are fired.

Since each philosopher must share the available chopsticks, a situation can easily occur in which certain philosophers cannot eat. A starvation-free solution to this problem has been proposed by Wu and Murata in [Wu 83].

Petri net theory has been well researched, extended and widely applied [Peterson 81], [Murata 89] as a consequence, many types of nets exist [Reisig 85], [David 91]. These can be broadly divided into the categories of low-level and high-level Petri nets.

2.2.1 Low-level Petri nets

Low-level Petri nets include condition-event (CE) nets and place-transition (PT) nets. The CE net is the simplest form of Petri net in which the token capacity of places is restricted to 0 or 1, and such places are termed as safe, ie.

$$M(p_i) \leq 1 \qquad \forall p_i \in P \text{ under any marking } M \qquad 2.4$$

This restriction gives a unique interpretation to the nodes of a CE net: places represent conditions that arise in the system, whilst transitions represent events. A natural extension of CE nets allow places to have a token capacity of k (where $k \geq 1$), and such Petri nets are termed as k-bounded. These nets are referred to as PT nets in which the interpretation given to the nodes of a CE net is not applicable. One of the useful features of PT nets is that they can produce models that are structurally less complex and compact, compared to CE nets.

Methods of analysing low-level Petri nets [Genrich 80], [Murata 89] are based on either formulating a reachability graph or using matrix manipulation. The analysis using reachability graphs enumerates all markings reachable from the initial marking of the net, by successively firing an enabled transition in each marking. This allows inspection of all possible markings that can occur in the net and is useful for detecting properties such as:

(i) reachability: this property refers to the possibility of reaching a particular marking, via a finite transition firing sequence, from a given marking,

(ii) boundedness: if places in a Petri net are k-bounded then they can accept a maximum of k tokens (where $k > 1$). Hence checks for boundedness ensure that places do not exceed their token capacities,

(iii) deadlock: this refers to a situation occurring in a Petri net model in which a marking is reached but no transitions are firable.

However, the major disadvantage of this approach is that the reachability graph can easily become complex, large and unmanageable even for relatively small Petri nets [Murata 89].

Matrix manipulation uses matrix equations and linear algebra, producing a technique referred to as Invariant Analysis. This approach allows the marking of the net to be represented by state equations of the form (considered in detail in Appendix A):

$$\mathbf{M_k} = \mathbf{M_{k-1}} + \mathbf{N^T}\,\mathbf{U_k} \qquad 2.5$$

where

$\mathbf{M_{k-1}}$	is a column vector characterising the current marking of the net,
$\mathbf{N}$	is referred to as the incidence matrix whose elements represent the arcs connecting a transition to a place or vice-versa. $\mathbf{N^T}$ is the transpose of $\mathbf{N}$,
$\mathbf{U_k}$	is a column vector which specifies the transition that can fire to yield the new marking given by $\mathbf{M_k}$,
$\mathbf{M_k}$	represents the immediate successor marking to $\mathbf{M_{k-1}}$.

An important part of the invariant analysis is the calculation of place-invariants (also known as S-invariants) and transition-invariants (T-invariants). More specifically, S-invariants allow detection of places that do not change their token count during firing of transitions. S-invariants can be calculated by finding the solutions ($\mathbf{y}$) to the set of homogeneous equations given by:

$$\mathbf{Ny} = 0 \qquad 2.6$$

T-invariants show the number of times a particular marking is reachable from a transition firing sequence of a Petri net. This invariant can be calculated by finding the solutions ($\mathbf{x}$) to the equation:

$$\mathbf{N^T x} = 0 \qquad 2.7$$

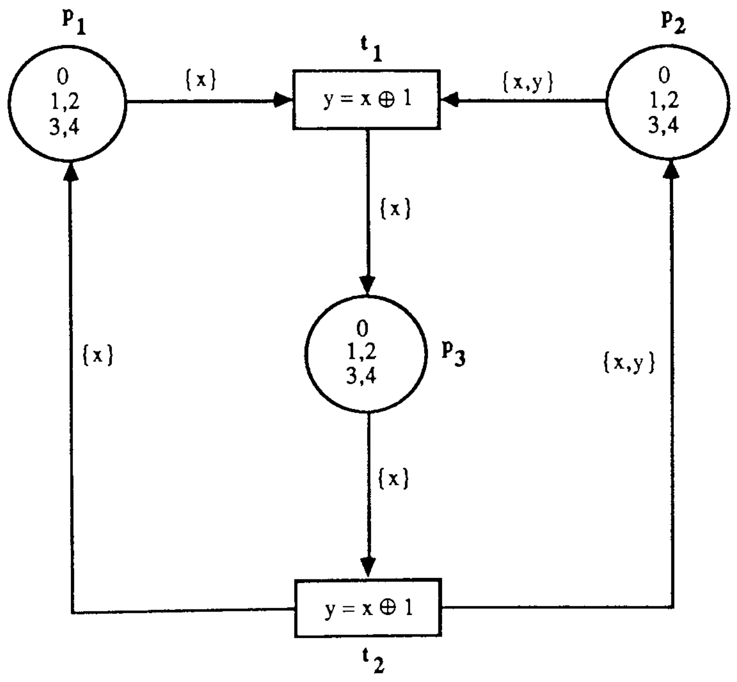
Knowledge of S-invariants and T-invariants is invaluable for deducing properties of the net, such as reachability, boundedness and deadlock. Since the calculation of S-invariants and T-invariants is done using matrix algebra, this provides a means of proving the existence of properties such as boundedness and freedom from deadlock. In particular S-invariants and T-invariants have been used to prove that Petri nets can produce correct models of concurrent systems [Lautenbach 74], [Reisig 85], and communication protocols [Berthelot 82]. However, the calculation of S and T-invariants can become tedious and laborious as the size of the net increases [Agerwala 79].

2.2.2 High-level Petri nets

Low-level Petri nets have a simple visual notation for specifying the causal relationships of concurrent systems. However, when modelling complex systems they have the disadvantage of producing large models, which are unmanageable and difficult to analyse [Genrich 81], [He 91]. This problem has been addressed by the development of high-level Petri nets, such as: Predicate-transition nets (PrT) [Genrich 81] and Coloured Petri nets (CP) [Jensen 81]. The following provides a brief discussion of these nets.

PrT nets [Genrich 81] were developed to provide a formal basis for describing large complex systems. PrT nets are essentially folded versions of low-level Petri nets [Murata 89] and it is possible to mechanically unfold the former to yield the latter. The abstractions used in PrT nets to produce a compact representation of PT nets are: a predicate of a PrT net represent sets of places of a low-level Petri net; its tokens represents individual variables. Similarly, a transition represents sets of transitions of a low-level Petri net; logical formulas are assigned to transitions that specify which tokens can participate in the enabling and firing. The arcs of a PrT net are restricted by relational expressions that specify which combination of tokens can engage in the firing process. An example of a PrT net is shown in Figure 2.2, that models the five dining philosophers problem, which was proposed by He and Lee [He 91].

Figure 2.2 A PrT net of a solution to the five dining philosophers problem.



where

p_1 is the predicate for philosophers are thinking; each philosopher in this predicate is referred to as $p_1(n)$ (where $0 \leq n \leq 4$),

p_2 is the predicate for the available chopsticks; each chopstick is referred to as $p_2(n)$ (where $0 \leq n \leq 4$),

p_3 is the predicate for philosophers are eating; each philosopher in this predicate is referred to as $p_3(n)$ (where $0 \leq n \leq 4$),

t_1 and t_2 stands for the actions of picking up and putting down chopsticks, respectively. The assignment of variables to the arcs show the number and the kind of tokens that will be removed from or added to the predicates. For instance, p_2 loses two tokens when t_1 fires, $\{x, y\}$; they are given as $p_2(x)$ and $p_2(y)$, where $y = x \oplus 1$.

The relational expression $y = x \oplus 1$ (where $\oplus$ is the modulo 5 addition operator) defines the condition between a philosopher and the chopsticks that the philosopher can use. For example, consider the situation when philosopher 2 is thinking (expressed as $p_1(2)$) and attempts to picks up the chopsticks from predicate p_2. Since the enabling conditions of t_1 are given by: $p_1(x)$, $p_2(x)$, $p_2(y)$ and $y = x \oplus 1$, this means that $p_1(2)$ can only use chopsticks 2 and 3, represented as $p_2(2)$ and $p_2(3)$.

PrT nets have been applied to many problems such as the analysis of distributed database systems [Genrich 79], [Genrich 81]; modelling of logic programs [Murata 88]; as a graphical tool for the design and prototyping of distributed systems [Dahler 87]. However, despite their representational advantages, PrT nets have a serious drawback concerning analysis. More specifically, the invariant analysis for low-level Petri nets, can be adapted to PrT nets [Genrich 81], but becomes mathematically complicated and needs much simplification and improvements to be practicable [Genrich 87]. This complication primarily arises because the elements of the incidence matrix are no longer integers (as in low-level Petri nets) but logical expressions; these contain free variables and currently there appear to be no rules to allow for their interpretation.

CP nets [Jensen 81] are similar to PrT nets, but have the advantage of being easier to analyse. A distinct feature of CP nets is that a set of colours are allocated to their tokens, transitions and places. The set of colours assigned to a place specify which coloured tokens it can accept; a transition may fire with respect to each of its colours (in accordance with the transition enabling and firing rules defined in Equs. 2.2 and 2.3); a functional dependency can be specified between the colour of the transition firing and the colours of the involved tokens. Furthermore, the firing of a transition may cause the colour of the token to change, and this can be used to represent a complex information unit. The input and output arcs of a transition are labelled with functions that specify which coloured tokens should be removed from its input places and which should be deposited in the output places, respectively. Graphically, places are represented by ellipses, transitions by rectangular boxes and tokens are not explicitly shown. The colour set associated with a place or a transition is indicated on one of its sides. A more detailed formal definition of CP nets may be found in [Jensen 81], [Jensen 87].

The similarities between PrT and CP nets have been investigated in [Jensen 87] and their strengths exploited in a new form of CP net, referred to as CP' net. In particular it combines the graphical form of PrT nets (which assists in the informal description of the system) and the matrix form of CP nets (that is useful for formal analysis). The invariant analysis explained in Section 2.2.1 has been extended to high-level nets. This analysis is easier to apply to CP' nets than PrT nets, but, there appears to be no general method for calculating S-invariants. Although Jensen [Jensen 87] presents transformation rules that assist in the detection of these invariants, it is, in general, not possible to find all invariants using this technique.

The application of CP nets is steadily increasing, some examples of which include the modelling of flexible manufacturing systems [Kamath 86] and the modelling of databases [Jensen 81], [Jensen 87].

2.3 Time in Petri nets

Both low and high-level Petri nets contain no explicit notion of time, in their definition or analysis methods, hence, they are unsuitable for modelling and analysing real-time systems. Approaches have been proposed that incorporate time in Petri nets, however, they have usually been restricted to extending low-level Petri nets. These extended nets can be classified according to the way in which time is introduced: time and timed Petri nets [Ramchandani 74], [Merlin 76] use the concept of deterministic times or delays, and stochastic Petri nets [Murata 89] view time in a probabilistic manner.

Although stochastic nets have been used for evaluating the performance of real-time systems, they have not been seriously considered for modelling hard real-time systems because time constraints in hard real-time systems are usually deterministic, rather than probabilistic. Therefore, time and timed Petri nets will be considered in the following discussion.

2.3.1 Time and Timed Petri nets

Both time and timed Petri nets extend the basic Petri net model by allocating quantitative time values to its nodes; however, these models differ by the way in which this time parameter is introduced.

2.3.1.1 Time Petri nets

Time Petri nets, developed by Merlin and Farber [Merlin 76], incorporated the notion of time by allocating a range of time values, a minimum and maximum value, to each transition in the net. In these nets an enabled transition could only fire within the range specified by its min. and max. time bounds. Time Petri nets were originally developed for examining the recoverability of communication protocols and the model of time used in these nets was considered to be particularly suited to modelling time-outs. The method was also used for analysing and verifying communication protocols [Berthomieu 83], [Menasche 83]. These approaches used a verification method which was based on manually enumerating the reachability graph, from which properties such as boundedness could be verified and the operation of the protocol validated. However, since a transition can fire within the range specified by the min. and max. time bounds then, theoretically, an infinite number of states can be considered. Thus, this technique suffers from the disadvantages associated with producing a reachability graph with a very large number of states. Recent approaches have successfully used time Petri nets to analyse the safety and fault tolerance aspects of hard real-time systems [Leveson 85], [Leveson 87]. In particular, they propose an algorithm that can detect the occurrence of hazardous states and sequences. This algorithm is based on generating a partial or total reachability graph, in which once a hazardous state is detected then, the reachability graph is used to backtrack from this hazardous state to the state causing this particular sequence. Thus, once these hazardous sequences are indentified then provisions can be made to eliminate them. These techniques have been successfully applied to the analysis of model problems (such as a railway level-crossing control problem), however, they are difficult to implement because of the problems associated with generating the reachability graph.

2.3.1.2 Timed Petri nets

Timed Petri nets, developed by Ramchandani [Ramchandani 74], introduced time into Petri nets by simply allocating a fixed time delay to each transition in the net. Ramamoorthy and

Ho [Ramamoorthy 80] used this technique to analyse the performance of a restricted class of Petri nets known as decision-free nets (which have no-decision places or are completely deterministic). In particular, they showed that evaluating the performance of decision-free nets can be done efficiently using timed Petri net theory but the performance evaluation of the general timed Petri net is NP-complete.

Zuberek [Zuberek 80] relaxed the restrictions imposed by Ramamoorthy and Ho [Ramamoorthy 80] by applying timed Petri nets to free-choice nets. More specifically, the model of time used was stochastic in nature: combining the transition delay with a probability of the transition firing. This model of time differs from that of Merlin and Farber [Merlin 76], not only because of the stochastic nature of the decision which governs the firing of a transition but also for the non-atomic way the transitions fire. For instance, in [Zuberek 80] when the most probable enabled transition starts to fire, tokens are absorbed from the input places (ie they disappear) and re-emerge in the output places after the delay has expired. The analysis of these nets was based on formulating a timed reachability graph, in which states are assigned time vectors, known as the remaining firing time, that keep track of the firing times of each transition in the net. These timed Petri nets were considered useful for evaluating the performance of communication protocols and for the derivation of expressions for resource utilisation.

Razouk and Phelps [Razouk 84], [Razouk 85] considered the application of timed Petri nets for deriving performance expressions and evaluating the performance of communication protocols. In particular, they proposed a timed Petri net that combined certain features of the models proposed by Zuberek [Zuberek 80] and Merlin and Farber [Merlin 76]; and further claimed that a more accurate model of time was obtained. For example, this model of time was based on assigning an enabling and firing time to each transition in the net. This required a transition to remain enabled for the enabling time, during which the transition can become disabled by the firing of a conflicting transition. However, if the transition reaches the firing phase then:

(i) firing commences and tokens are absorbed from the input places,
(ii) the tokens are retained by this transition until the firing time expires,
(iii) at the instant the firing time expires, tokens are deposited into the appropriate output places.

Thus during the firing phase the transition cannot become disabled and must complete firing.

A comparison of the various forms of time and timed Petri nets reveals some subtle differences between them. For instance, in Merlin and Farber's model an enabled transition must fire instantaneously within the range specified by its min. and max. times, unless it becomes disabled by the firing of a conflicting transition. However, in Razouk and Phelps's model the firing of a transition occurs in phases; during the enabling phase a transition can become disabled; during the firing phase the transition must complete firing. Merlin and Farber's model may be considered as a more general and flexible version than Razouk and Phelps's model, because more states must be considered in the former than the latter. However, this feature of the former makes it difficult to analyse. In terms of modelling events and activities Merlin and Farber's model can represent events that occur instantaneously but in Razouk and Phelp's model events consume time.

The approach proposed by Ghezzi et al [Ghezzi 91] attempts to integrate the functional and temporal behaviours of safety-critical systems, within the formalism of high-level Petri nets. In particular, they extend the basic Petri net to form a high-level Petri net referred to as ER (Environment/Relationship) net; this net is not derived from PrT or CP nets, but can capture functional properties in a similar manner. The following components of ER nets differ from those of the classical Petri nets: transitions are associated with actions, and their firing models the occurrence of events. In an ER net a token is referred to as an environment, and it is defined as a function which associates a value to a variable. Time is incorporated into ER nets to form time ER nets (TER nets) by assigning a timestamp to an environment. The progression of time is indicated when a transition fires and its associated action allocates a new timestamp to an environment. The TER net incorporates two notions of time: a weak model in which an enabled transition *may* fire after its enabling time, and a strong model in which an enabled transition is forced to fire after its enabling time.
Comparing ER nets to time and timed Petri nets, Ghezzi argues that the latter are useful for the performance analysis of communication protocols, but claims that they are inadequate for modelling the temporal aspects of general real-time systems because of the alleged '*ad-hoc*' manner in which time has been introduced into the net. Significantly, Ghezzi shows that the generalised model of time in TER nets can easily model time and timed Petri nets. Although this strengthens the case for TER nets, much of this work is in its infancy and needs further development. For instance, currently, ER nets are weak for proving such properties as reachability, liveness and do not really support formal verification.

2.4 Temporal logic

Temporal logic [Rescher 71], [Pnueli 77] is a suitable formalism for specifying and verifying the behaviour of concurrent systems. It was derived from modal logic [Chellas 80], which is an extension of propositional and predicate logic. More specifically, propositional logic is concerned with establishing the truth value (truth or falsity) of statements by examining the truth value of their constituents; predicate logic is a form of propositional logic that allows reasoning about complicated statements involving universal and existential quantification; modal logic, based on either propositional or predicate logic, allows reasoning about statements, whose truth value may change under different states (or worlds).

The underlying computational model of modal logic [Chellas 80], which gives an interpretation (or semantics) to formulas written in this language, consists of a universe of states and an accessibility relation, which specifies the possibility of getting from one state to another. Temporal logic differs from modal logic [Manna 83a] because the interpretation given to the accessibility relation is the passage of time. Thus, a state s is accessible from another state s' if through a process in time s can change into s'.

Temporal logic embraces two views regarding the underlying nature of time [Rescher 71]. One is that time is linear: at each moment there is only one possible future. The other is that time has a branching, tree-like nature: at each moment, time may split into alternative courses representing different possible futures. Depending upon which view is chosen, a system of temporal logic is linear time temporal logic, LTL, [Pnueli 77], if the semantics of the time structure is linear, or branching time temporal logic, BTL, [Emerson 86], if the semantics correspond to a branching time structure. There has been much confusion regarding the use of these logics which almost developed into a controversy.

2.4.1 Branching time and Linear time

Initial comparisons of BTL and LTL by Lamport [Lamport 80], Ben-Ari et al [Ben-Ari 83] suggested that the expressive power of these logics are such that different types of systems could be modelled. For instance, BTL was deemed suitable for reasoning about non-deterministic programs (programs in which a state can have more than one successor, corresponding to a non-deterministic choice, and each of these possible choices must be systematically explored) and LTL was suitable for reasoning about concurrent programs (in which a state may have several possible futures, the choice of which to take is non-deterministic but can be resolved by imposing a fairness requirement). Although, both logics could express safety and liveness properties, LTL was considered to be much superior to BTL, because of the simple and natural way in which these properties could be proved.

Emerson and Hailpern [Emerson 86] suggested that the comparison of Lamport [Lamport 80] was only applicable to the particular logics examined and that his results are not generally applicable. They emphasize that a more useful comparison should consider several types of BTLs and LTLs. For such a comparison, a BTL was developed known as computation tree logic (CTL). Their comparison showed that whilst LTLs are generally adequate for proving the correctness of pre-existing concurrent programs, however, BTLs can be enriched with modal operators to incorporate the concept of fairness and hence be made applicable to concurrent programs. Furthermore, they suggest that BTL is suitable for model checking, which allows properties of finite state concurrent programs to be verified. For instance, given a finite state concurrent program, its global state graph can be viewed as a CTL structure. The problem of checking whether the program has a certain property reduces to that of checking whether the formula describing that property holds for the CTL structure corresponding to the program. Model checking for large subclasses of CTL can be done very efficiently, and, in general, it is easier to achieve using BTL than LTL. Clarke, Emerson and Sistla [Clarke 86] develop the notion of model checking by presenting an algorithm which automatically verifies finite state concurrent systems. In particular, they extend CTL with the notion of fairness (producing a language called CTL*) so that fair execution sequences can be considered. However, in this research LTL will be considered because it is considered to be most suited to describing concurrent systems and has a more developed proof system.

2.4.2 Qualitative time LTL

In linear time temporal logic, LTL, [Pnueli 88] time is further subdivided into qualitative time and quantitative time. Qualitative time uses the concept of eventuality and fairness to guarantee the eventual occurrence of an event but, provides no time bound on how soon. On the other hand, quantitative time expresses time as a value (usually an integer) by which an event may occur. Logics based on quantitative time are particularly suited to expressing properties of hard real-time systems: hence they are referred to as real-time temporal logics, RTTLs, [Pnueli 88]. Since RTTLs are formed by extending Qualitative LTLs, typically by introducing the set of integers to express time constraints, and both logics share the same basic proof system, this thesis will consider both types of logics.

The computational model of LTL consists of a possibly infinite sequence of states (σ) and an accessibility relation (R), and allows an interpretation be given to temporal formulas. This model can be represented as:

$\sigma = s_0, s_1, \ldots$ 2.8

$R(s_n, s_{n+1})$ 2.9

where

s_n (where $n \geq 0$) represents a state,

$R(s_n, s_{n+1})$ is the accessibility relation.

In LTL, R is restricted to, and defined as, a function that specifies the possibility of getting from one state into another state, within the temporal domain. The notation of LTL avoids explicit mention of R but, instead, defines temporal operators which, in conjunction with logical formulas, specify the progression of time over σ. These temporal operators are syntactically represented as:

always - $\Box$ eventually - $\Diamond$ next - O until - $\mathcal{U}$

The semantics of the above operators are defined as follows. For a state sequence (σ), let the sequence $\sigma^{(k)}$ be the k-shifted sequence given by:

$\sigma^{(k)} = s_k, s_{k+1}, \ldots$ 2.10

then,

(i) if w is a classical formula (which is constructed from propositions or predicates and logical operators such as NOT ($\neg$), AND ($\wedge$), OR ($\vee$) and implication ($\Rightarrow$)) containing no temporal operators then,

$\sigma \models w \quad \text{iff} \quad s_0 \models w$ 2.11

in this case w can be interpreted over a single state in σ, which is the initial state s_0,

(ii) temporal operator always ($\Box$) is defined as:

$\sigma \models \Box w \quad \text{iff} \quad \forall k \geq 0, \quad \sigma^{(k)} \models w$ 2.12

$\Box w$ holds on σ iff all states in σ satisfy w,

(iii) temporal operator eventually ($\Diamond$)

$\sigma \models \Diamond w \quad \text{iff} \quad \exists k \geq 0, \quad \sigma^{(k)} \models w$ 2.13

$\Diamond w$ holds on σ iff at least one state in σ satisfy w. Alternatively, eventually ($\Diamond$) can be defined in terms of the temporal operator always ($\Box$) as: $\Diamond w \equiv \neg\Box\neg w$

(iv) temporal operator next (O)

$\sigma \models \mathrm{O}w \quad \text{iff} \quad \sigma^{(1)} \models w$ 2.14

Ow holds on σ iff $\sigma^{(1)}$ satisfies w,

(v) temporal operator until ($\mathcal{U}$)

$$\sigma \models x\mathcal{U}y \quad \text{iff} \quad \exists k \geq 0 \text{ such that, } \sigma^{(k)} \models y, \text{ and } \forall i,\ 0 \leq i \leq k,\quad \sigma^{(i)} \models x \qquad 2.15$$

$x\mathcal{U}y$ holds on σ iff sometime y holds and until then x holds continuously.

Initially, Pnueli [Pnueli 77] proposed LTL as a suitable formalism for the specification and verification of concurrent systems; a basic proof system was developed that used the temporal operators always ($\Box$) and eventually ($\Diamond$). Specifically, this logic and its proof system was considered suitable for proving safety and liveness properties of concurrent systems. Later this proof system was made more expressive in [Gabbay 80] with the introduction of the 'until' temporal operator and the proof system was proved to be both sound and complete.

LTL was further developed by Lamport [Owicki 82], [Lamport 83a], [Lamport 83b] for proving the correctness of concurrent systems. For example, the work of [Owicki 82] applies LTL for proving safety and liveness properties of concurrent programs, in which the mutual exclusion problem is considered, with and without synchronization primitives. The proof system used consisted of axioms and inference rules peculiar to temporal logic and others that are unique to the program under consideration; which were developed for proving liveness properties. Also the notion of proof lattices [Owicki 82] was introduced, that provide a diagrammatically concise representation of a proof, in which the nodes of the lattice represented temporal formulas and edges represented steps made to obtain these formulas. The LTL proposed in [Pnueli 77], [Owicki 82], [Manna 83a] is inherently global in the sense that it is used to verify the properties of the complete program. This is evident because only global state changes are considered in the proofs. However, if specifications are described in a modular manner then this methodology does not provide rules for deducing properties of the complete program from its constituents. As a consequence, this has led to the compositional approach of formulating temporal specifications [Barringer 84]. The general framework of modular specifications is composed of a module and its interface. The specification should describe the behaviour of a module and its interaction with the environment as observed on the interface. An important feature of the compositional approach is the need to distinguish between actions performed by the module and actions performed by the environment. This problem has been addressed in [Lamport 83a], [Pnueli 86] and [Lamport 89]. In [Lamport 83a], [Lamport 89] the behaviour of concurrent modules is specified, such that actions are given predicates or labels and the logic is augmented by action propositions: these can sense whether the next atomic transition to be taken is a module transition or an environment transition. Pnueli [Pnueli 86] addresses the above problem by partitioning the actions performed by the module and its environment. The main restriction imposed by this partition is that the former can only modify its own variables, whereas the latter may never modify a variable owned by the module. The proof system for compositional verification has been proposed by [Pnueli 85].

Although the above logics use a proof system that is adequate for verification, it is unique to the problem under consideration. This is because such a proof system uses axioms that explicitly considers each transition and its possible outcome when executed. Since a full proof system would become lengthy and tedious, this approach is not suited to verifying systems in general. Research into the development of a general proof system for LTL was presented by [Manna 83c], [Pnueli 86]. Specifically, the language of LTL is considered to consist of symbols, terms and temporal formulas. Terms are used to name objects in the universe considered and are constructed from individual constants, individual variables and function symbols. Temporal formulas make assertions about terms and are constructed from predicate symbols, individual constants, individual variables, function symbols, and the logical and temporal operators (shown in Section 2.4.2). Therefore, the computational model of LTL can be represented by the tuple:

$$(I,\beta,\sigma) \tag{2.16}$$

where

- I is the interpretation that specifies a meaning to the individual constants and function symbols according to their sort and specifies the domain under consideration,
- β is an assignment that assigns a value over the appropriate domain to each of the global individual variables,
- σ represents the possibly infinite state sequence: $\sigma = s_0, s_1, \ldots$

The generic model [Manna 83c], [Pnueli 86] proposed for describing concurrent systems was referred to as the Fair Transition System (FTS), which has the following constituents:

Σ	A possibly infinite set of states,
T	A finite set of transitions,
Θ	A set of initial states,
J	A justice family. This is a family of sets $J = \{J_1, \ldots, J_n\}$ where each $J_i \subset T$ represents a justice requirement,
F	A Fairness family. This is a family of sets $F = \{F_1, \ldots, F_n\}$ where each $F_i \subset T$ represents a fairness requirement.

Although fairness and justice are representations of the same concept [Pnueli 86] a distinction between them needs to be made because of the different costs associated with implementing them. For instance, in a concurrent configuration justice is normally guaranteed and needs no special mechanisms for it to be guaranteed. Fairness, on the other hand, requires special task scheduling algorithms for it to be guaranteed. The decision to use justice or fairness will depend upon the application being considered. However, once these requirements are chosen then the FTS provides a proof system (which is sound and complete), that can be used for specification and verification purposes. The application of FTS [Pnueli 86] has been illustrated by modelling the mutual exclusion problem using several techniques such as, shared variables and Petri nets. Furthermore, in [Pnueli 86] it is suggested that FTS can be applied to any technique that uses operational semantics (which defines the execution of a system by a sequence of states) to represent its computational model.

The general proof system of FTS consists of three parts (or layers): general, domain and program part. The general part is composed of axioms and inference rules that are valid for any temporal logic system. The domain part restricts the reasoning to a particular domain in which all the predicate and function symbols have a fixed interpretation and the individual variables range over several fixed domains. The program part confines reasoning to the particular program under consideration.

2.4.3 Quantitative time LTL

Bernstein and Harter [Bernstein 87] proposed a real-time temporal logic, RTTL, which extended the LTL of Pnueli [Pnueli 77] and adopted the approach of Owicki and Lamport [Owicki 82] for proving properties of real-time programs. In this approach the execution of the program is represented by a computational model (consisting of an infinite sequence of states entered by the program); a terminating execution is represented by repeating the final state indefinitely. The model uses a global time domain which is the set of real numbers. Every state in the semantic description of a program has associated with it the time at which it is entered. Real-time is introduced into LTL by extending the logical operator implication ($\Rightarrow$) with time bounds. For example, the formula:

$$s_1 \overset{<n}{\Rightarrow} s_2 \qquad 2.17$$

asserts that if execution reaches a state satisfying s_1 then it will reach a state satisfying s_2 in less than n units of time. On the other hand, the formula:

$$s_1 \overset{>n}{\Rightarrow} s_2 \tag{2.18}$$

asserts that if execution reaches a state satisfying s_1 then s_2 will not be true in that state and the execution will not reach a state satisfying s_2 within the next n units of time. The proof system of this logic is limited to establishing application specific real-time safety properties and cannot be used to prove liveness properties.

The timed model of distributed systems proposed by Shanker and Lam [Shanker 87] uses both the notions of states and events. Each process has a set of state variables and a set of events. An event is modelled as a state-transformer together with its enabling conditions. Safety and liveness properties are verified in a first order logic extended by the temporal operator 'leads-to'; this operator was introduced by Owicki and Lamport [Owicki 82] and is equivalent to the temporal formula: $\Box[s_1 \Rightarrow \Diamond s_2]$ - which is interpreted as "s_1 leads to s_2". However, this operator does not have explicit time associated with it. Instead, each process is assumed to have a set of 'local timers' which are distinguished state variables. These timer variables take values from the domain of natural numbers and can be enabled (reset to 0) or disabled by events. Shanker and Lam apply their approach to verify the correct operation of real-time protocols, which are used in a network of processes that communicate via message passing.

Jahanian and Mok [Jahanian 86] propose a somewhat different approach for the formal specification and verification of timing properties of hard real-time systems. They use an event-action model to informally capture the timing relationships and data dependencies of the system and develop a real-time logic (RTL) that specifies and verifies its timing behaviour. The event-action model gives a precise interpretation to events and actions. Events, which are central to RTL, are described using an occurrence function, that maps the occurrence of an event to the time of occurrence (events impose no requirement on system resources). An action is an operation which requires a bounded amount of system resources. An action has two associated events - the 'initiation' event and the 'termination' event. This informal description can be mechanically translated into RTL, that allows formal analysis. More specifically, RTL consists of first order logic that is augmented with time, which is introduced as the set of non-negative integers. The description of the system during the 'interval' given by pairs of event occurrences is represented by the state predicate. The state predicte is a logical expression that asserts a truth value, which can change as a result of the execution of an action or an external event.

Jahanian, Mok, Stuart [Jahanian 88a] suggests that RTL adopts a different approach to reasoning about time-dependent systems, compared to approaches centred on transition based systems (such as temporal logic). For instance, the unique features of RTL are:

(i) it makes no mention of states or transitions, either in the language or in its models; although, the state predicate does seem to provide a form of state representation,
(ii) the RTL's computational model is centred on the describing a set of events and actions, in which actions are performed in response to events. The events occurring in the event-action model may be interpreted as a sequence of events. However, these sequences may not be interpreted as an interleaved sequence of events, as for transition based systems, because such sequences are considered to be inadequate for modelling the timing behaviour of systems [Jahanian 86].

The above allows the following distinctions to be made: in RTL time passes between sets of events, and the actual sets of events are instantaneous, while in transition based systems, time passes in states, and transitions between them are instantaneous. Secondly, transition based systems are generally defined by their transitions, so that if two actions of the modelled system occur concurrently, a transition must be explicitly included to reflect that concurrent act. However, in RTL such concurrency is implicit because a system using the occurrence of a set of events which cause actions to be executed. RTL has been shown to be more expressive than LTL [Jahanian 88a] but the full logic is undecidable and only a small subset is decidable. Consequently in RTL, the decision procedures used to prove safety properties work well for small systems, but become difficult and inefficient for larger systems. Improvements to these decision procedures have been proposed using graph-theoretic techniques [Jahanian 87], however, they are not complete decision procedures and work well only for small subsets of RTL specifications.

Jahanian, Lee and Mok [Jahanian 88b] use RTL to define the semantics of a graphical language known as Modechart to provide a visual representation of real-time systems. Modechart is a variant of statecharts developed by Harel [Harel 87]. Statecharts are in turn an extension of the conventional state-transition diagrams and include features such as concurrency, communication and the notion of hierarchy. Modecharts differ from statecharts because they use the notion of modes (which represent classes of states that are viewed as containing control information, that impose a structure on the operation of systems) instead of states; they incorporate the notion of time and limit the conditions needed to fire transitions.

Decision procedures for verifying properties of modechart specifications [Jahanian 88c] are centred on generating computation graphs; which are superficially similar to the reachability graphs used in Petri net theory. In particular, the nodes of a computation graph define the occurrence of events, whilst the arcs represent causality relationships between these events (however, it does not necessarily specify that events occur consecutively), the separation of these events are denoted by time. Although, computation graphs are similar to reachability graphs, it is not possible to perform invariant analysis on computation graphs, because RTL does not use the notion of states and transitions.
The decision procedures of RTL in modechart specifications are based on establishing whether a given computation graph satisfies a particular property, for a decidable class of RTL formulas. However, verification using this approach suffers from the same problems as those encountered in reachability analysis. In particular, it is impractical to generate the entire graph for a complex modechart specification.

An alternative approach that extends the fair transition system, FTS, of [Pnueli 86] to produce a RTTL is proposed by Ostroff [Ostroff 87], [Ostroff 89]. In particular, he shows that RTTL can be combined with extended state machines (ESM), to produce a technique referred to as ESM/RTTL. This technique is specifically developed for modelling hard real-time systems in terms of discrete-event systems and allows for formal specification and verification. This approach considers systems to comprise the process to be controlled (the plant) and the controlling software (the controller). ESMs are state machines that have been augmented with the notion of real-time, concurrency and the ability to specify interprocess communications, to provide an intuitive means of informally specifying the problem. The transitions of an ESM are given an upper time bound (u) and lower time bound (l) which are related to an ESM clock process. This clock, which is assumed to never terminate, produces a periodic 'tick' infinitely often. Thus when a transition becomes enabled, it is prevented from occurring for at least l ticks of the clock, but must not be continuously enabled for more than u ticks of the clock. Furthermore, the proof system for FTS, which consists of a three layer proof system (see Section 2.4.2), is extended to incorporate a fourth layer (referred to as an ESM layer) which contains axioms and inference rules for proving real-time properties of ESMs, such as delay, response times - corresponding to the upper and lower time bounds etc. The decision procedures [Ostroff 89] used to prove properties of real-time systems are centred on enumerating a reachability graph, however, they suffer from the limitations that are peculiar to reachability graph type analysis, namely state-explosion problem. Although ESM/RTTL is useful for describing real-time systems, there appears to be no published work that shows its application to a 'real world' problem.

2.5 CSP

CSP (Communicating Sequential Processes) developed by Hoare [Hoare 85] is a mathematical approach for specifying and verifying concurrent systems and their associated communications. CSP models concurrent systems by using the notion of a process (or processes) that interacts with its environment, which is external to the process and provides it with input stimuli. A process may represent a single sequential system or several concurrent systems. A process is described in terms of events (which occur instantaneously) that are considered to be observations made by an imaginary observer of the interaction of the process and its environment. The relative ordering of events can be recorded to provide a trace of a process's behaviour. This representation provides a convenient abstraction for modelling large complex systems. The notion of state is not explicitly defined in CSP, although inclusion of states in CSP has been considered by Josephs [Josephs 88]. The exclusion of states in CSP can be considered as a disadvantage because experience with other specification methods (such as temporal logic and Petri nets) has shown the use of states to be a very convenient way of describing complex systems.

More formally, the CSP computational model of a process is defined by its alphabet and the sets of traces and failures. The alphabet represents the set of all possible events a process may engage in. A trace defines the behaviour of a process and enumerates the sequence of events that the process has participated in up to some moment in time. Consider a trace (t) of process (P) and a set of actions (X) offered by the environment to P. If it is possible for P to deadlock on its first step when placed in this environment, then X is a refusal of P. Thus a failure of P occurs if P has engaged in the sequence of events recorded by t and then refuses to do anything more, in spite of the fact that its environment is prepared to engage in any of the events of X. Processes in CSP are defined using a process algebra which is used to construct rules that describe the behaviour of the process. Various proof systems exist in CSP for proving properties which can be divided into safety and liveness (these properties are defined in the same way as [Alpern 85]). For instance, [Olderog 86] examines several proof systems which are based on different semantic domains, such as the traces model and the failures model, and shows that the traces model is restricted to proving safety properties, whilst the failures model can be used to prove liveness properties. CSP has undergone much research (mainly performed by the Programming Research Group, Oxford University) and has been extended in various ways to be applicable to programming languages [Joseph 89], a well known example of which is Occam [INMOS 87]. Although CSP is a useful formalism for modelling concurrent systems, it does not incorporate the notion of time and thus is not suited to describing real-time systems.

The notion of time has been introduced into CSP by Reed and Roscoe [Reed 88] who developed a mathematical theory in which the semantics of a time dependent CSP, known as time CSP (TCSP), is defined. The underlying assumptions made about time are:

(i) the occurrence of events is related to a conceptual global clock,
(ii) there is a non-zero lower time bound specified between any two events in the history of a sequential process,
(iii) there is no lower time bound on the time interval between two independent actions.

TCSP provides a concise and clear description of systems and is useful for modelling real-time systems. It has been used by Davies and Schneider [Davies 89] in the modelling of time-outs for the alternating bit communication protocol. However, at present, the proof system of TCSP is limited to proving properties such as partial correctness [Reed 88]; moreover, the proof system is difficult to use and its verification procedures need to be simplified before it will find widespread application.

2.6 VDM and Z

VDM [Jones 86] and Z [Spivey 88] are specification languages that have received widespread acceptance in industry. These methods are based on set theory and predicate logic, and are centred on a state-transition based computational model. Although these methods have many similarities they differ in their approach to specifying systems. For example, VDM is not just a specification language but a complete methodology that assists in all stages of systems development from specification to implementation. This is achieved by augmenting the initial specification with some implementation detail at each stage of the systems evolution (a process referred to as reification) until a point is reached when software can be directly derived from the specification. On the other hand, Z specifies a system in terms of schemas (a schema is part of the system that performs a specific function) and a schema calculus is used to join these schemas in order to form a complete specification. Currently, VDM and Z do not provide direct means of dealing with concurrency and real-time, therefore they are unsuitable for describing real-time systems.

2.7 PAISLey

An alternative approach to using formal methods for specifying and verifying real-time systems is proposed by Zave [Zave 82], in her work on the development of executable specifications using the specification language PAISLey (Process orientated, Applicative and Interpretable Specification Language). Zave argues that either a conventional or operational approach can be adopted for software development [Zave 84]. The conventional approach uses the principle of top-down decomposition of black boxes. In this approach the initial specification treats the system as a black box and specifies all required characteristics of its external behaviour but, not the internal structure of the black box. As the system is developed the structure of the black box is systematically defined. The alternative operational approach is said to be problem-orientated because a system is represented as a problem (or a series of problems) and implementation-independent structures are proposed for solving this problem (or problems). Thus the internal and external behaviour of the system are 'interleaved' and specified at the specification phase. Zave favours the operational approach and incorporates it in the language PAISLey [Zave 86] which combines the computational models of asynchronous processes and functional programming. These formalisms allow a system to be described as a set of asynchronous processes where each process may be represented as an infinite sequence of discrete states, and the computations performed within a process are described using functional programming. PAISLey is capable of specifying parallelism and timing constraints, and it can also cope with incomplete specifications. Execution of a PAISLey specification can show all the sequences of states that can be reached. This provides a powerful means of validating the specification, ie checking that a specification conforms to user requirements. However, PAISLey does not provide the means for formally verifying its specifications [Zave 91], ie proving that a specification possess desired properties. Although PAISLey provides a useful approach for modelling real-time systems, it needs to be enhanced to support such features as modularity, reliability analysis and a graphical representation of languages.

2.8 Discussion

This chapter aimed to provide a survey of techniques suitable for specifying and verifying real-time systems. This survey concentrated on techniques which had a mathematical basis and allowed the reasoning of the behaviour of these systems. The salient features of each technique were examined and the notations and formal models were illustrated. These features are summarised in Table 2.1.

Table 2.1 Summary of formal techniques

Features	Petri nets			Temporal Logic				CSP	TCSP	VDM/Z	PAISLey
	Low & High-Level	Time(d)	ER	CTL	Qualitative	RTTL	RTL/ Modechart				
Formal base	Bag theory Graph theory	Bag theory Graph theory	Set theory	Modal Logic	Modal Logic	Modal Logic	Predicate Logic	Set theory + Predicate Logic	Set theory + Predicate Logic	Set theory + Predicate Logic	Execution Semantics
Computational Model	Interleaved State-transition	Interleaved State-transition	Interleaved State-transition + Time	Interleaved State-transition	Interleaved State-transition	Interleaved State-transition + Time	Event-Action	Event-based	Event-based	State-transition	asynchronous processes + functional programming
Concurrency	√	√	√	√	√	√	√	√	√	x	√
Real-time	x	√	√	x	x	√	√	x	√	x	√
Analysis Method	Matrix theory or Reachability graph	Matrix theory or Reachability graph	Reachability graph	Automated Algorithm	Deductive Proof System	Deductive Proof System	Deductive Proof System + Computation graphs	Deductive Proof System	Deductive Proof System	Deductive Proof System	Executable Specification
Status	Extensively Researched and Applied	Extensively Researched and Applied	Current Research	Current Research	Current Research	Current Research	Current Research	Current Research	Current Research	Widely Used in Industry	Current Research

The following discussion compares the subset of each method which accommodates both concurrency and time.

Petri nets have the desirable feature of modelling concurrent systems of all sizes, using a concise and intuitive graphical notation. The analysis techniques for Petri nets can be effectively used to detect interesting properties, such as deadlock, however they can not be used to prove safety and liveness properties in a similar manner to temporal logic or CSP. Petri nets suffer from the state-explosion problem and various methods have been developed for coping with complexity. The inclusion of time in Petri nets is considered to be inflexible because it restricts their application to evaluating the performance of systems. However, this problem has been recognised and is being addressed by the development of ER nets.

Temporal logic is based on a more flexible model of time, embracing the notion of branching and linear time. Its proof system allows verification of both safety and liveness properties. Although temporal logic is adequate for modelling concurrent systems, it can produce specifications that are somewhat illegible and difficult to verify. For modelling real-time systems various RTTLs exist: these are basically time extensions of qualitative time LTL and allow proof of some real-time properties. A particularly promising RTTL is RTL, which has the following distinctive features:

(i) it does not use the notion of states or transitions,
(ii) it does not use the interleaving model to represent concurrency,
(iii) it does not use temporal operators for expressing time dependent behaviour.

Although RTL does not explicitly define the notion of states, it appears implicitly as the state predicate in RTL and as modes (sets of states) in the derived notation of modechart. The exclusion of states in RTL means it is not possible to show or prove the existence of an invariant holding throughout an execution, in a similar manner to temporal logic or Petri nets. This may be considered as a disadvantage of RTL and is probably the main reason for the development of analysis techniques based on computation graphs, which are similar to reachability graphs, in modechart specifications. However, RTL has been proved to be more expressive than temporal logic, and may be considered to use a more accurate model of concurrency. Currently, the full RTL is undecidable and decision procedures exist for only a small subset of RTL.

CSP is an established formalism that has some similarities with temporal logic and Petri nets. Specifically, CSP does not use the notion of states or transitions and describes systems in terms of observable events - similar to RTL. Moreover the notion of observable events is a useful abstraction for specifying the internal structure of systems. In CSP the behaviour of a process is defined in terms of its traces, which is superficially similar, but not equivalent, to the execution sequences in a reachability graph of a Petri net. It is possible to prove safety and liveness properties in CSP, a distinction also made in temporal logic. TCSP is also suitable for representing real-time systems, however, because its proof system is not fully developed it can be difficult and complicated to use. The mathematical nature of CSP may limit its widespread use (especially in industry) compared to such methods as Petri nets or temporal logic.

PAISLey is a formal approach that produces an executable specification and allows the formal validation but not the formal verification of specifications written in its language. Hence PAISLey uses a different approach for modelling and analysing real-time systems than techniques such as temporal logic, CSP, VDM and Z, which are based on a deductive proof system that is used for proving relevant properties. There are obvious advantages in using PAISLey that stem from its ability to develop an early prototype of the system and show an execution of the specification. However, it needs to be supplemented by:

(i) formal techniques to allow for formal verification,
(ii) software tools to provide adequate features for specifying real-time systems, hence, the provision of CASE support tools for PAISLey is a current area of research activity.

The above survey has shown there to be a whole spectrum of mathematically based techniques for describing real-time systems. Typically, techniques based on logical formalisms allow verification of specifications, however, they may be too mathematical to use - producing specifications that are difficult to understand and difficult to verify. This in turn may prevent their widespread acceptance and use. Techniques based on graphical formalisms may be more understandable but may not allow the proof of properties. This limits their use to exploring the behaviour of systems and the inference, rather than verification, of properties. Alternative techniques using executable specifications lead rapidly to executable models, but may require domain specific environments.

This thesis considers that formal methods for specifying real-time systems should ideally contain a mixture of a mathematical proof system and graphical features. The mathematical basis should be sufficient for verifying properties and the graphical features should provide a user friendly interface that facilitates easy understanding of the specification. To achieve this requires the combination of techniques to produce a unified method in which each technique can be used to specify complementary properties. Thus the resulting method should draw on the strength of its constituents. While it may be possible to combine various techniques to yield a unified approach, constituent techniques must be compatible to ensure that an inconsistent specification is not written in each formulism. For example, the combination of TCSP with Petri nets may produce useful results; however, since each technique is centred on a different computational model, then problems may occur in relating their semantics.

A useful combination may be made by relating techniques that have related features such the same computational model, as in temporal logic and Petri nets. Intuitively, such a combination is desirable and it is interesting to note that, while Petri nets provide a simple and intuitive means of representing concurrency via the graphical notation, they lack an adequate proof system for verifying safety and liveness properties. On the other hand, temporal logic allows specification of time related behaviour and has the ability to verify liveness and safety properties. Thus the combination of Petri nets and temporal logic is considered to be advantageous because each technique can specify complementary properties. This direction is pursued in the following chapters.

Chapter 3

Combining Petri nets and Temporal logic

3.1 Introduction

No existing formal technique fully satisfies the requirements for describing hard real-time systems. An alternative to creating a completely new technique is to combine existing formalisms to produce a unified technique. Ideally this unified technique should inherit positive features from its constituent techniques, and as a consequence become more expressive than either of its constituent techniques. Examples of this approach have been used for describing general computer systems, such as in [Fraser 91], concurrent systems, such as in [Wing 89], and hard real-time systems, ie presented in [Jahanian 88b]. In particular, Fraser, Kumar and Vaishnavi [Fraser 91] bridged the gap between informal and formal approaches for specifying computer based systems by combining structured analysis and VDM. Wing and Nixon [Wing 89] extend the specification language Ina Jo by combining it with branching time temporal logic, BTL, to enhance its expressibilty for modelling concurrent systems. Jahanian, Lee and Mok [Jahanian 88b] introduce the specification language modechart for describing hard real-time systems which combines RTL and a graphical language, which is a variant of statecharts.

The prospect of combining Petri nets and temporal logic is attractive, not only because they have the same computational model, but also for their ability to capture specific characteristics of hard real-time systems. For example, Petri nets can capture the causal aspects of a system using a graphical notation, which allows the designer to informally specify what the system is required to do; this essentially represents the safety properties of a system. In addition, temporal logic can be used to reason about the temporal behaviour of the system, providing proof of both safety and liveness properties (see Section 2.1.1 for definition of these properties). This chapter investigates how this combination can be achieved by examining previous approaches to the problem; much of the work presented in this chapter has also appeared in [Sagoo 91], [Holding 92].

3.2 Merging Petri nets and temporal logic

The problem of combining Petri nets and temporal logic was discussed briefly by Genrich, Lautenbach and Thiagarajan [Genrich 80] and has subsequently been addressed by a number of research workers [Queille 82], [Diaz 83], [Anttila 83], [Suzuki 85], [Reisig 88], [He 90]. Typically, the approach taken has been to produce a separate specification in each formalism; thus it is common to find the use of variants of Petri nets to validate safety properties and temporal logic to prove liveness properties of the Petri net. However, the combination of the dissimilar formalisms of Petri nets and temporal logic may lead to inconsistencies. Specifically, if a separate specification is written in each formalism, there is no way of checking the consistency between them. However, the use of separate formalisms has continued. For instance, Queille and Sifakis [Queille 82] used a form of Petri nets (known as Interpreted Petri nets) and BTL in their work on CESAR, an interactive design tool for specifying and verifying distributed systems. Similarly, Diaz and Silverira [Diaz 83] related a form of Petri nets (known as predicate-action nets) and BTL for the specification and verification of communication protocols. Anttila, Erikson and Ikonen [Anttila 83] combined CE nets (see Section 2.2.1) and linear time temporal logic, LTL, for the specification of systems in their work of developing a Petri net analyser. In particular, temporal logic was used to describe features of the Petri net and temporal formulas were used to express either the net markings (yielding a state-based logic) or the firing of transitions (resulting in an event-based logic) or the interval between firing a transition (producing an interval-based logic). This research showed that event-based logics are deficient in specifying systems because of the ambiguity which is introduced when trying to infer the state of the net from a formula written in this logic. This difficulty is not encountered in state-based logics.

Recently, Suzuki [Suzuki 85] combined Petri nets and LTL and introduced it as a new class of Petri nets known as Temporal Petri nets. This combination was achieved by constructing propositions, using LTL, that formalised the firing of transitions in a Petri net. Although Suzuki [Suzuki 85] does not explicitly address the issue of consistently combining Petri nets and LTL, the propositions of temporal Petri nets implicitly allow a consistent translation of Petri nets into LTL. The propositions of temporal Petri nets, which formalise the transition firing sequences of Petri nets, form the basis for proving liveness properties in an axiomatic manner. Temporal Petri nets were primarily developed for the design and verification of concurrent systems. This was illustrated by Suzuki and Lu [Suzuki 89] by the formal analysis of a daisy chain arbiter, where it was shown that the process of verification using an axiomatic technique was much simpler than the traditional approaches. It transpired that safety properties could be elegantly represented by Petri nets, while liveness properties can be easily specified and proved using LTL. Further usefulness of temporal Petri nets has been demonstrated in [Suzuki 90] in which the formal analysis of the alternating bit protocol is performed using the Buchi-automaton.

Subsequently, Reisig [Reisig 88] combined Petri nets and LTL by considering the manner in which the state execution sequences (marking sequences) of the net are formulated. In particular the interleaving semantics (see Section 2.1.1) and the causality based semantics (that use partial ordering of events to produce state execution sequences) were considered. Reisig related a temporal logic to the state execution sequences of a Petri net obtained using interleaving semantics and causality based semantics, ie:

(i) the LTL of Manna and Pnueli [Manna 83b] (also discussed in Section 2.4.2) was related to the state sequences obtained using interleaving semantics,
(ii) a LTL, developed in [Reisig 88], was related to the state sequences obtained using the causality based semantics.

The combination of Petri nets and temporal logic in the above manner allows a consistent translation between these formalisms; however, this issue was not addressed or mentioned in [Reisig 88]. Although, a LTL based on interleaving semantics has been fully developed in [Manna 83b], [Pnueli 86], Reisig suggests that a LTL centred on causality based semantics would more accurately specify system behaviour. Such a LTL and its proof system is being developed by Reisig [Reisig 88].

More recently, an alternative approach was proposed by He and Lee [He 90] that presents a unified technique, based on the combination of PrT nets and a first order LTL, for use in the specification and verification of concurrent systems. In particular they explicitly address the problem of combining Petri nets and LTL such that a specification written in the unified technique, is consistent in each formalism. The combination of PrT nets and first order LTL was based on relating the computational models of each technique. This relationship formed the basis of an algorithm that allowed PrT nets to be translated into first order LTL. This algorithm formalised the firing rules of each transition into a set of inference rules and the initial marking was expressed as a logical axiom. Thus, the proof system comprises domain independent axioms and inference rules, which are obtained from the temporal logic proof system [Manna 83b], [Pnueli 86], and domain dependent axioms and inference rules that are unique to the particular Petri net under consideration. This complete set of axioms and inference rules can be used in the verification of both safety and liveness properties.

The main motivation for the work of He and Lee [He 90] was to provide an alternative approach for analysing PrT nets, since linear algebra is difficult to apply to PrT nets (see Section 2.2.2). The use of PrT nets and first order LTL suggests that this technique was primarily developed for handling large concurrent systems, since PrT nets have the capability of producing concise and compact specifications for relatively large systems. Although PrT nets are well suited to modelling problems such as the dining philosophers [He 90] they do not produce structured specifications. Hence, this aspect needs to be addressed if techniques based on PrT nets are to be effective in modelling large concurrent systems [He 91].

This section has described various attempts to combine Petri nets and temporal logic. Some approaches, such as [Genrich 80], [Queille 82], [Diaz 83], [Anttila 83], allow separate specifications to be written in each formalism which has the disadvantage that there appears to be no means of checking the consistency between these specifications. Other approaches address this problem, either implicitly as in [Suzuki 89] or explicitly as in [He 90]. These approaches, which attempt to produce a unified method, are considered to be worthy of further examination. The remaining sections of this chapter are devoted to providing a formal description of these techniques and examining their attributes and limitations.

3.3 Temporal Petri nets

The following provides a formal basis for the study of temporal Petri nets. Much of this work is derived from [Suzuki 85], [Suzuki 89] in which temporal Petri nets were originally defined.

3.3.1 Definition of temporal Petri nets

Formally, a temporal Petri net is defined as the pair:

$$TN = (C,f) \tag{3.1}$$

where:

C is the Petri net structure defined in Equ 2.1 and is represented as: $C = \{P,T,I,O,M_0\}$,

f represents a set of temporal formulas that describe the temporal behaviour of C.

f is described in a language which is based on LTL of [Manna 83b]; in [Suzuki 89] this language was referred to as L_N. The syntax and semantics of L_N is defined as follows.

(i) Syntax of L_N

A formula of L_N is built from:

(a) atomic propositions, where $p \in P$ and $t \in T$:
(p has a token), (t is firable), (t fires) and (p has no token),

(b) logical operators (also introduced in Section 2.4.2): $\neg$ (NOT), $\wedge$ (AND), $\vee$ (OR) and $\Rightarrow$ (implication),

(c) temporal operators (also introduced in Section 2.4.2):
O (next), $\Box$ (henceforth or always), $\Diamond$ (eventually) and $\mathcal{U}$ (until).

The formation rules of L_N are:

(a) an atomic proposition is a formula,

(b) if f_1 and f_2 are formulas, then so are $(f_1 \wedge f_2)$, $(f_1 \vee f_2)$, $\neg f_1$, $(f_1 \Rightarrow f_2)$, Of_1, $\Box f_1$, $\Diamond f_1$ and $f_1 \ \mathcal{U} \ f_2$.

(ii) Semantics of L_N

The above syntax is given semantics which assigns an interpretation (or meaning) to each formula of L_N. Informally, the atomic propositions of (i), are intended to mean:

(p has a token)	-	p has at least one token at the current marking,
(t is firable)	-	t is firable at the current marking,
(t fires)	-	t fires at the current marking,
(p has no token)	-	p has no tokens at the current marking.

The meaning of the logical operators are obvious.

If formula f uses temporal operators then its intended meanings are:

Of_1	-	f_1 is satisfied at the next marking,
$\Box f_1$	-	f_1 is satisfied at every marking reachable from the current marking,
$\Diamond f_1$	-	f_1 is satisfied at some marking reachable from the current marking,

$f_1 \,\mathcal{U}\, f_2$ - f_1 remains satisfied at least until f_2 becomes satisfied at a marking reachable from the current marking.

The formal semantics of L_N are given as follows.
Let α be a possibly infinite transition firing sequence from marking M. Also for each i, $0 \leq i \leq |\alpha|$ - (where $|\alpha|$ denotes the length of α), let β_i and γ_i be sequences such that $|\beta_i| = i$ and $\alpha = \beta_i \wedge \gamma_i$. That is, β_i is the prefix of α with length i, and γ_i is the postfix of α excluding β_i.
The notation $M \Rightarrow_t M'$ is used to denote that, if $t \in T$ is firable at M, then it *may* fire and yield another marking M'. Let M_i be the marking such that $M \Rightarrow_{\beta_i} M_i$, which denotes that from marking M it is possible to reach marking M_i by a firing sequence β_i. For α and an integer j, α^j is the sequence obtained by concatenating j occurrences of α, therefore α^0 is defined to be λ (where λ represents the empty sequence).
For a formula f,

$<M,\alpha>$	$\models$	f	means that f is satisfied by the pair M and α
$<M,\alpha>$	$\models$	(p has a token)	iff p has at least one token at M
$<M,\alpha>$	$\models$	(t is firable)	iff t is firable at M
$<M,\alpha>$	$\models$	(t fires)	iff $\alpha \neq \lambda$ and $t = \beta_1$
$<M,\alpha>$	$\models$	$f_1 \wedge f_2$	iff $<M,\alpha> \models f_1$ and $<M,\alpha> \models f_2$
$<M,\alpha>$	$\models$	$f_1 \vee f_2$	iff $<M,\alpha> \models f_1$ or $<M,\alpha> \models f_2$
$<M,\alpha>$	$\models$	$\neg f_1$	iff not $<M,\alpha> \models f_1$
$<M,\alpha>$	$\models$	$f_1 \Rightarrow f_2$	iff $<M,\alpha> \models f_1$ implies $<M,\alpha> \models f_2$
$<M,\alpha>$	$\models$	Of	iff $\alpha \neq \lambda$ and $<M_1,\gamma_1> \models f$
$<M,\alpha>$	$\models$	$\Box f$	iff $<M_i,\gamma_i> \models f$ for every $0 \leq i \leq |\alpha|$
$<M,\alpha>$	$\models$	$\Diamond f$	iff $<M_i,\gamma_i> \models f$ for some $0 \leq i \leq |\alpha|$
$<M,\alpha>$	$\models$	$f_1 \,\mathcal{U}\, f_2$	iff $(<M_i,\gamma_i> \models f_1$ for every $0 \leq i \leq |\alpha|)$ or (for some $0 \leq i \leq |\alpha|$, $<M_i,\gamma_i> \models f_2$ and $<M_j,\gamma_j> \models f_1$ for every $0 \leq j < i$).

The formulas of L_N are constructed from the nodes of the Petri net; the following shows the abbreviated notation used in this thesis to construct these formulas:

$p \equiv$ (p has a token)	t(ok) $\equiv$ (t is firable)	t $\equiv$ (t fires)
$\neg p \equiv$ (p has no token)	t($\neg$ok) $\equiv$ (t is not firable).	

The following shows some formulas expressed using the above notation and their semantics:

(a) $<M,\alpha> \models \bigcirc t$:

transition t must fire at the next marking reachable from M when α occurs,

(b) $<M,\alpha> \models \Box p$:

place p must not become token-free at any marking reachable from M when α occurs,

(c) $<M,\alpha> \models \Diamond p$:

place p must eventually have at least one token at any marking reachable from M when α occurs,

(d) $<M,\alpha> \models \Box(t_1 \Rightarrow \Diamond t_2)$:

whenever transition t_1 fires, transition t_2 must eventually fire in α,

(e) $<M,\alpha> \models \Box\Diamond t$:

transition t fires infinitely often in α,

(f) $<M,\alpha> \models p \Rightarrow (p\ \mathcal{U}\ t)$:

if place p has at least one token at M, then p must not become token-free at least until t fires.

The work of Suzuki and Lu [Suzuki 89] defines a temporal Petri net (which is given by the pair: TN = (C,f)) such that f is interpreted as a restriction on the firing sequences generated from C; thus only those firing sequences that satisfy f are allowed to occur. Formally, this is defined as:

For a marking M the set L(TN, M) denotes the set of all finite firing sequences from M in TN which is defined by:

$$L(TN, M) = L^{\infty}(C, M) \cap \{\alpha \in T^{\infty} \mid <M,\alpha> \models f\}$$

where

$L^{\infty}(C, M)$ denotes the set of all infinite firing sequences from M in C,

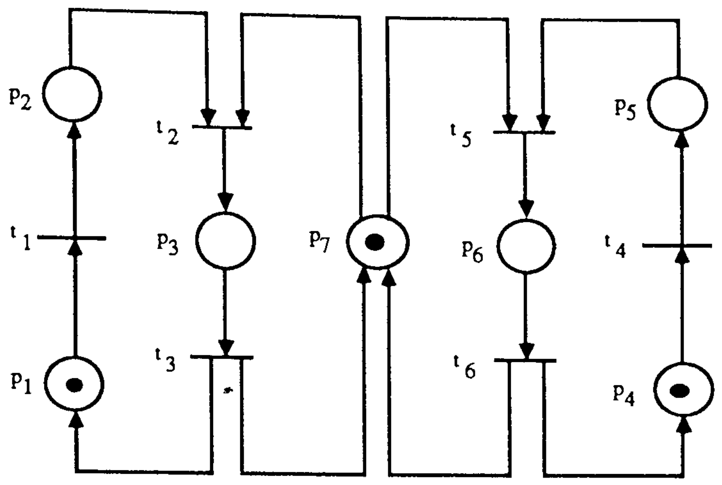
T^{∞} denotes the set of all infinite sequences of T.

The set of markings reachable from M in TN is defined by:

$$R(TN, M) = \{ M' \mid M \Rightarrow_{\beta} M' \text{ for some prefix } \beta \text{ of some } \alpha \in L(TN, M)\}.$$

In order to illustrate the use of the notation and language of L_N, the temporal behaviour of a Petri net will be described. Consider the Petri net, Figure 3.1, which shows the mutual exclusion problem where a common resource (CR) is shared by two processes. At any moment, the resource can be used only by one process.

Figure 3.1 Petri net C_1



The places and transitions of C_1 represent the following conditions and events:

t_1 (or t_4): Process A (or B) requests CR,
t_2 (or t_5): Process A (or B) acquires CR,
t_3 (or t_6): Process A (or B) releases CR,
p_1 (or p_4): Process A (or B) not using CR,
p_2 (or p_5): Process A (or B) is waiting to use CR,
p_3 (or p_6): Process A (or B) is using CR,
p_7: CR is available.

The formulas that specify the possible temporal behaviours of C_1 can be written as:
Process A (or B) may not continue to use the resource forever:

(i) Process A: $f_1 = \Box(t_2 \Rightarrow \Diamond t_3)$,
(ii) Process B: $f_2 = \Box(t_5 \Rightarrow \Diamond t_6)$.

Assuming that Processes A and B do not "halt" while they are requesting the resource, then this situation is described as:

(iii) Process A: $f_3 = \Box(t_2(ok) \Rightarrow \Diamond t_2(\neg ok))$,
(iv) Process B: $f_4 = \Box(t_5(ok) \Rightarrow \Diamond t_5(\neg ok))$.

Under the above assumptions, the fairness requirement that Process A (or Process B) should not fail to acquire the resource forever is represented by:

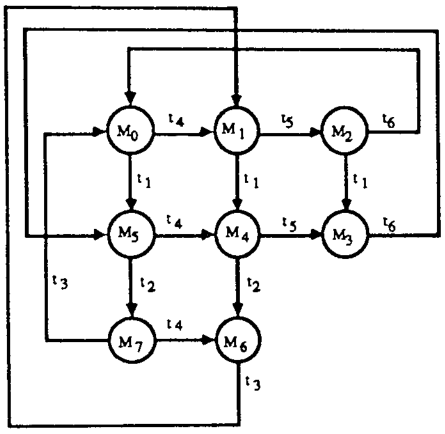
(v) Process A: $f_5 = (\Box\Diamond t_2(ok)) \Rightarrow \Box\Diamond t_2$
f_5 is interpreted as, if t_2 is firable at infinitely many markings, then t_2 must fire infinitely often,
(vi) Process B: $f_6 = (\Box\Diamond t_5(ok)) \Rightarrow \Box\Diamond t_5$
f_6 is interpreted as, if t_5 is firable at infinitely many markings, then t_5 must fire infinitely often.

Let $TN_1 = (C_1, f)$ be the temporal Petri net of Figure 3.1, where f is the conjunction of the set of formulas such that $f = f_1 \wedge f_2 \wedge f_3 \wedge f_4 \wedge f_5 \wedge f_6$.

For TN_1, if the initial marking is $M_0 = \{p_1, p_4, p_7\}$, then the set of firing sequences of TN_1 consists of all finite and infinite sequences that can be obtained as a "shuffle" of $(t_1, t_2, t_3)^i$ and $(t_4, t_5, t_6)^i$ where i is a positive integer or $i = \omega$, in which case $i < \omega$ for any integer i.

The restrictions imposed by f on TN_1 can be illustrated by considering the type of transition firing sequences that can be obtained from the reachability graph (shown in Figure 3.2).

Figure 3.2. Reachability graph for Petri net C_1



MARKING	PLACES
M_0	p_1, p_4, p_7
M_1	p_1, p_5, p_7
M_2	p_1, p_6
M_3	p_2, p_6
M_4	p_2, p_5, p_7
M_5	p_2, p_4, p_7
M_6	p_3, p_5
M_7	p_3, p_4

From Figure 3.2 it is seen that any transition firing sequence that includes the sequence (t_2, t_3) or (t_2, t_4, t_3) satisfies f_1, whilst sequences involving (t_5, t_6) or (t_5, t_1, t_6) satisfy f_2. Similarly, any changes in net marking involving the firing of transition t_2 (or t_5) satisfy f_3 (or f_4). However, requirements f_5 and f_6 prevent any firing sequence that contains an infinite loop consisting of the firing sequence (t_4, t_5, t_6) and (t_1, t_2, t_3), respectively. These requirements are important because an infinite loop of sequence (t_4, t_5, t_6) denies Process A access to the resource, similarly an infinite sequence (t_1, t_2, t_3) prevents access to Process B. Examples of such forbidden sequences are:

(i) $(t_4, t_1, t_5, t_6), (t_4, t_5, t_6)^\infty$ and $t_1, (t_4, t_5, t_6)^\infty$ or

(ii) $(t_1, t_4, t_2, t_3), (t_1, t_2, t_3)^\infty$ and $t_4, (t_1, t_2, t_3)^\infty$.

3.3.2 Temporal Petri net analysis

Suzuki [Suzuki 89] suggests that safety properties, which characterize what can possibly occur in the modelled system, are manifested by the structure of the Petri net, whereas liveness properties are realised using temporal logic. More specifically, the specification and verification of liveness properties are achieved by defining propositions concerning the firing of transitions of the Petri net. These propositions can be proved to be valid within the temporal logic proof system of [Manna 83b] and are used in conjunction with the axioms and inference rules of this temporal logic proof system to verify liveness properties in an axiomatic manner. The temporal Petri net analysis introduced by Suzuki and Lu [Suzuki 89] was based on a proposition which formed the basis of proving liveness properties. This proposition formalised the firing of a transition and is defined below.

Proposition 1:
For a temporal Petri net TN = (C,f) whose initial marking is M_0.

If

(i) for any marking M reachable from M_0 by a firing sequence α, if t is firable at M then t remains firable until t itself fires:

$$<M_0,\alpha> \models \Box[t(ok) \Rightarrow t(ok)\,\mathcal{U}t] \qquad 3.2$$

the interpretation of Equ. 3.2 is, for an initial marking M_0 and a possibly infinite transition firing sequence α it is true that whenever t becomes firable then it will remain firable until t fires,

and

(ii) whenever, t becomes firable, then it will eventually become disabled:

$$f \text{ implies } <M_0,\alpha> \models \Box[t(ok) \Rightarrow \Diamond t(\neg ok)] \qquad 3.3$$

f represents the condition that for an initial marking M_0 and any firing sequence α it is true that whenever t becomes firable then it will eventually become disabled.

Then

for any transition firing sequence α generated from M_0, using (i) and (ii) it can be proved, via the temporal proof system of [Manna 83b], that:

(iii) whenever t becomes enabled then it must eventually fire:

$$<M_0,\alpha> \models \Box[t(ok) \Rightarrow \Diamond t] \qquad 3.4$$

Equ. 3.4 is interpreted as for an initial marking M_0 and a possibly infinite transition firing sequence α it is true that whenever t becomes enabled then eventually it will fire.

However, the universal validity of the above proposition is subject to doubt, most obviously for Petri nets which include conflict. Therefore, it seems prudent to examine Proposition 1 in some detail. Inspection of Proposition 1 shows that:

(a) the premise of Proposition 1(i) asserts a safety property. This is evident by observing the assertion which it makes: a firable transition t remains firable *until* it actually fires. This premise only asserts what happens to a firable transition (t); it does not assert (or guarantee) that t will fire. Similarly, in Petri net theory, the enabling and firing rules of transitions (see Equs 2.2 and 2.3) specify that an enabled transition *may* fire, they do not guarantee that such a transition will fire. The claim that Proposition 1(i) represents a safety property is further substantiated from the work of Pnueli [Pnueli 85], in which it is revealed that formulas using the *until* operator are classified as asserting a safety property.

(b) Proposition 1(ii), asserts what must eventually happen to a firable transition and thus is a manifestation of a liveness property. In Suzuki and Lu's work, Proposition 1(ii) is needed because Proposition 1(i) by itself is not sufficient to guarantee the firing of a transition. It is interesting to note that there appears to be no need for Proposition 1(ii) because from the temporal proof system of [Manna 83b] the following is a valid theorem (expressed in terms of the notation of temporal Petri nets):

$$t(ok)\,\mathcal{U}\,t \Rightarrow \Diamond t \qquad 3.5$$

Thus combining Proposition 1(i) and Equ. 3.5 it is possible to deduce:

$$[t(ok) \Rightarrow \Diamond t] \qquad 3.6$$

which is the conclusion of Proposition 1. However, as pointed out earlier Proposition 1(ii) must be used in conjunction with Proposition 1(i) in order to guarantee that the enabled transition t actually fires.

Therefore, Proposition 1 combines both a safety and a liveness property but it is used to prove liveness properties. However, before Proposition 1 can be applied it is necessary to check the validity of Proposition 1(i); Suzuki [Suzuki 89] suggests that this premise must be tested by either examining the structure of the Petri net or examining its reachability graph.

(c) Proposition 1(ii) asserts that a firable transition must eventually become disabled, however, it does not specify how the transition becomes disabled, only that it does. Therefore this premise seems to cover two forms of transition disabling: either the enabled transition becomes disabled by the firing of itself or by the firing of a conflicting transition (conflict occurs when two transitions are enabled and the firing of one transition also disables the other as well as itself). However, the conjunction of Proposition 1(i) and (ii) restricts the type of transition firings to those that allow an enabled transition to fire and only disable itself, but it does not cater for conflicting transitions. For example, consider the Petri net of Figure 3.1 in which Proposition 1 is applicable to the firing of all transitions except for transitions t_2 and t_5 enabled under marking: $M_4 = \{p_2, p_5, p_7\}$.
In this case the firing of t_2 disables itself and t_5, whereas firing t_5 disables itself and t_2. Therefore, both situations violate the assertion of Proposition 1(i) and thus Proposition 1 cannot be used in the proof of liveness properties which involve the simultaneous enabling of transitions t_2 and t_5. It follows that, Proposition 1 is applicable to Petri nets which permit the firing of simple transitions, which contain no conflicting transitions. The inability of temporal Petri net analysis, based on Proposition 1, to cater for conflicting transitions limits the range of systems that can be verified.

In [Suzuki 89], [Suzuki 91], Suzuki used Proposition 1 to prove the 'correctness' of a Petri net model comprising the handshake signals between two processes; this net contained no conflicting transitions. This proof was then used to verify other claims about the global properties of a Petri net comprising a daisy chain arbiter consisting of n processes (where $n > 2$). However, an examination of this daisy chain arbiter Petri net showed the existence of many conflicting transitions for which Proposition 1 was clearly not valid. Since, Suzuki and Lu proved the 'correctness' of the daisy chain arbiter net in [Suzuki 89], they must have developed a proposition which was less restrictive than Proposition 1 that would cater for the firing of conflicting transitions (because the 'correctness' of the daisy chain arbiter could not be proved without it). However, this type of proposition was neither published nor the need for it mentioned in the temporal Petri net analysis presented in [Suzuki 89].

Since conflicting situations can arise in concurrent systems (as shown in the relatively simple example of the mutual exclusion problem Figure 3.1) and real-time systems (typically in applications where processes must make a choice between several optional execution paths), it is important to explicitly define a proposition which will cater for conflicting situations. Such a proposition would have the following form.

Proposition 2 (Less restricted alternative to Proposition 1)
For a temporal Petri net TN = (C,f) whose initial marking is M_0.
If

(i) for any marking M reachable from M_0, if transitions t_1 and t_2, with input functions $I(t_1)$ and $I(t_2)$ respectively, are firable at M, and are in conflict, which is defined by:

$$I(t_1) \cap I(t_2) \neq \varnothing \qquad 3.7$$

where $\varnothing$ is the empty set,

then either t_1 remains firable until either t_1 itself fires or t_2 fires, else t_2 remains firable until either t_2 itself fires or t_1 fires. This statement is formalised as:

$$<M_0,\alpha> \models \Box[t_1(ok) \Rightarrow t_1(ok)\mathcal{U}(t_1 \vee t_2)] \vee$$
$$\Box[t_2(ok) \Rightarrow t_2(ok)\mathcal{U}(t_1 \vee t_2)]. \qquad 3.8$$

In the above situation only one transition is allowed to fire because the interleaving model is used to represent concurrency and the choice of which fires is made non-deterministically or according to a fairness requirement, if specified,

and

(ii) whenever t_1 becomes firable then it must become disabled and whenever t_2 becomes firable then it must become disabled, ie:
f implies

$$<M_0,\alpha> \models \Box[t_1(ok) \Rightarrow \Diamond t_1(\neg ok)] \wedge$$
$$\Box[t_2(ok) \Rightarrow \Diamond t_2(\neg ok)]. \qquad 3.9$$

Then

(iii) for any transition firing sequence α from M_0, using (i) and (ii) it can be proved, using the temporal logic proof system of [Manna 83b] that:

$$<M_0,\alpha> \models \Box[t_1(ok) \wedge t_2(ok) \Rightarrow \Diamond(t_1 \vee t_2)] \qquad 3.10$$

Proof of Proposition 2 using the temporal logic proof system of [Manna 83b] is shown in Appendix B.

In temporal Petri net analysis Propositions 1 and 2 should be applied in the following manner: the particular liveness property to be verified is identified as well as its associated reachability graph, then the validity of the more restrictive Proposition 1(i) is tested against the appropriate marking sequence in the reachability graph. If, however, Proposition 1(i) does not hold true, in the case of enabled transitions in conflict, then Proposition 2 is used. It is worth noting that Proposition 2 reduces to Proposition 1, if there are no conflicting transitions.

3.4 Integrating Petri nets and temporal logic - He and Lee's approach

The work of He and Lee [He 90] explicitly considered the issue of combining Petri nets and temporal logic in a consistent manner to yield a unified technique suitable for specifying and verifying concurrent systems. In particular they combined PrT nets (see Section 2.2.2) and first order LTL, developed by [Manna 83b], by relating the computational model of each formalism. This is possible because both formalisms use operational semantics to model systems. Specifically, Petri nets generate an interleaved marking sequence and temporal logic describes system behaviour using a sequence of interleaved states. He and Lee [He 90] considered relating PrT nets and first order LTL by translating the marking sequences of PrT nets into the state sequences of first order LTL. Although this semantic relationship is conceptually elegant, it is impractical to achieve because both execution sequences, and the number of execution sequences, that need to be considered could be infinite. He and Lee achieved the combination of PrT nets and first order LTL by developing an algorithm that translated the former into the latter. The consequences of performing this translation is:

(i) the PrT net is described in terms of first order LTL,

(ii) the proof system of first order LTL, *shown in [Manna 83b], is extended and* incorporates: axioms and inference rules of LTL, which already exist in the proof system of [Manna 83b], and an axiom and inference rules that are obtained via the translation algorithm and are unique to the net under consideration.

Thus the above approach allows the system which is modelled using Petri nets to be specified and verified.

In He and Lee' s algorithm the PrT net is translated into first order LTL in two stages:

(i) the initial marking of the PrT net is expressed as an axiom, which is referred to as the system dependent axiom, and is constructed from the conjunction of the predicates that form this marking,

(ii) the pre and postcondition of every transition in a PrT net are converted into an inference rule, which is referred to as the system dependent inference rule and has the form:

$$X \Rightarrow \bigcirc Y$$

where X and Y are called the antecedent (the precondition of a transition) and consequent (the postcondition of a transition) respectively, all variables occurring in X and Y are implicitly universally quantified, and the symbol $\Rightarrow$ denotes that if X is satisfiable in the current state then Y will be satisfiable in the next state of the computation.

An illustration of the translation algorithm for Figure 2.2 is shown as follows:

System dependent axiom

A1. $p_1(0) \wedge p_1(1) \wedge p_1(2) \wedge p_1(3) \wedge p_1(4) \wedge p_2(0) \wedge p_2(1) \wedge p_2(2) \wedge p_2(3) \wedge p_2(4)$

A1 asserts an axiom concerning the initial marking shown in Figure 2.2 in which all philosophers are thinking (given by $p_1(0) \wedge \ldots \wedge p_1(4)$) and all chopsticks are available (given by $p_2(0) \wedge \ldots \wedge p_2(4)$).

System dependent inference rules

I1. $p_1(x) \wedge p_2(x) \wedge p_2(y) \wedge \neg p_3(x) \wedge (y = x \oplus 1)$
$$\Rightarrow O(\neg p_1(x) \wedge \neg p_2(x) \wedge \neg p_2(y) \wedge p_3(x) \wedge (y = x \oplus 1))$$

I2. $\neg p_1(x) \wedge \neg p_2(x) \wedge \neg p_2(y) \wedge p_3(x) \wedge (y = x \oplus 1)$
$$\Rightarrow O(p_1(x) \wedge p_2(x) \wedge p_2(y) \wedge \neg p_3(x) \wedge (y = x \oplus 1))$$

I1 and I2 assert, in a logical form, the pre and postconditions of transitions t_1 and t_2 respectively.

The system dependent axiom and inference rules, and the axioms and inference rules of first order LTL form the framework for a proof system which is used to prove both safety and liveness properties. In the work of He and Lee [He 90] this proof system is used to prove safety and liveness properties by using a refutation proof procedure. Specifically, this procedure works by negating the property that needs to be proved and using the above proof system to obtain a contradiction of this negated property.

3.4.1 Extension of He and Lee's approach

The use of PrT nets and first order LTL in the unified technique of He and Lee [He 90] suggests that it was developed for modelling large concurrent systems, and this is evident because PrT nets were specifically developed for reducing the complexity of Petri net structures produced using low-level Petri nets (see Section 2.2.2). However, for systems that can be adequately modelled by low-level Petri nets, this technique may be too expressive. This is because the functionality of the system may be 'hidden' within the abstract notation of PrT nets, producing a more concise and abstract description of the problem than is necessary. More importantly, although first order LTL uses the qualitative notion of time, PrT nets do not include the notion of time and therefore the unified technique of [He 90] cannot fully model or analyse the temporal behaviour of hard real-time systems. However, the usefulness of this unified technique for proving safety and liveness properties has motivated the desire to adapt it to low-level Petri nets, such as CE nets. Such an extension would be beneficial for modelling and analysing hard real-time systems. Thus, once the system is modelled in low-level Petri nets then its properties may be formally verified, using the proof system to derive safety and liveness properties, and its real-time behaviour analysed using timed Petri net theory. Hence, this section is devoted to describing the development of such an extension to the work of He and Lee [He 90].

Prior to adapting the approach of He and Lee to CE nets, it is constructive to re-assert some similarities between PrT and CE nets:

(i) structurally, PrT nets can be considered to be folded versions of CE or PT nets (see Section 2.2.2). Thus, by mechanically unfolding the predicates and transitions of the PrT net a low-level Petri net will result,

(ii) the basic formal structure of PrT and CE nets is similar with the notable difference that PrT nets have a range of constants, functions and relations associated with its predicates and transitions that allow the places and transitions of CE nets to be represented in a compact manner,

(iii) formally, the computational model of PrT and CE nets is the same, because both nets use identical transition enabling and firing rules to generate new markings from an initial marking, and they both use interleaving semantics to generate marking sequences.

3.4.1.1 Development of a relationship between the computational models of CE nets and LTL

In He and Lee [He 90], the computational models of PrT nets and first order LTL were related via a theorem. Intuitively, using the arguments concerning the similarities between PrT nets and CE nets, it seems reasonable that a similar theorem can be derived for integrating CE nets and LTL. However, it must be pointed out that integrating CE nets and LTL requires the use of propositional LTL instead of first order LTL, because first order LTL is too expressive for CE nets. Thus the theorem which relates the computational models of CE nets and propositional LTL is:

Theorem 1
For a CE net, $C = \{P,T,I,O,M_0\}$, a set of temporal logic models (I,β,Σ) that characterise the set of marking sequences of C can be derived.

Theorem 1 represents the temporal logic model as the tuple (I,β,Σ). I and β are referred to as the interpretation and assignment (see Section 2.4.2) and Σ denotes the operational semantics of the model under consideration and represents all the different execution sequences that can be generated. Thus any possibly infinite state sequence σ, is such that:

$$\sigma \in \Sigma$$

Therefore, for any state sequence σ a temporal logic model is induced and is given by:

$$(I,\beta,\sigma)$$

Proof

The theorem proposed in [He 90], which related the computational models of PrT nets and first order temporal logic, was proved using a constructive proof. Since this theorem is similar to Theorem proposed above, it is sufficient for this thesis to show a similar constructive proof for the latter. This proof has the form:

(i) since the meanings of the individual constants and function symbols do not change during the execution of the CE net, their number and meanings should be fixed in any derived temporal logic model (I,β,σ) and thus they belong to I,

(ii) any assignment of individual variables in the CE net does not change meaning from state to state. Therefore, the assignment of β is a correct characterisation in any derived temporal logic model,

(iii) each marking sequence in the CE net derives a state sequence (σ) and thus a temporal logic model. Hence, for a temporal logic model (I,β,σ), $\sigma \in \Sigma$ iff there is a marking sequence of the CE net, where σ is defined in terms of this marking sequence.

Therefore the above has established a correspondence between the set of marking sequences of a given CE net and a set of temporal logic models (I,β,Σ) that characterise it.

3.4.1.2 Development of an algorithm for translating CE nets to LTL

The adaptation of the translation algorithm for PrT nets, detailed in Section 3.4, to CE nets requires no change to the way in which the system dependent axiom is formed. However, changes must be made to the system dependent inference rules (X and Y) because:

(i) a PrT net has relational expressions and variables associated with its transitions, whereas these are not used in CE nets,

(ii) in [He 90] only pure PrT nets are considered (ie nets that have no self loops, see [Murata 89], in which a place appears as both the input and output place of a transition), but in this case non-pure CE nets are considered.

The system dependent inference rules for CE nets can be formed as follows:
for a marked CE net $C = \{P,T,\mathcal{I},\mathcal{O},M_0\}$, a system dependent inference rule for each transition $t_j \in T$ in C can be represented as:

$$X \Rightarrow OY$$

where:

(i) the precondition (X) of $t_j \in T$ is constructed from two propositional formulas (using the abbreviated notation of Section 3.3.1(ii)):
   (a) the conjunction of the tokenised places of the input function $\mathcal{I}(t_j)$ (following Equ. 2.2),
   (b) the state of the postcondition before t_j fires,

(ii) the postcondtion (Y) of $t_j \in T$ is developed from two propositional formulas:
   (a) the conjunction of the tokenised places in the output function, obtained from the expression when t_j fires (following Equ. 2.3):
   $\#[p_i,\mathcal{O}(t_j)] \qquad \forall p_i \in P$ under marking M reachable from M_0
   (b) the conjunction of the negation of the places in the input function of the transition t_j when t_j fires, unless t_j contains a self-loop.

Additionally, for conflicting transitions, only one transition can fire at any instant and the choice of which transition fires is made non-deterministically or is based on a fairness requirement.

On the basis of the above translation algorithm, the proof system (L) used to verify safety and liveness properties is composed of:

(i) a system dependent axiom and inference rules, which are unique to the CE net under consideration,
(ii) a set of system independent axioms and inference rules, which form the temporal logic proof system of [Manna 83b], that are valid for any system using temporal logic.

The similarities between the above proof system and that of [He 90] is such that Theorem 2 of [He 90], which was used by He and Lee to prove the soundness of the PrT net and first order LTL can also be stated in this case:

Theorem 2

The temporal logic system L is sound relative to the set of temporal logic models ie. for any temporal logic formula p: L |- p implies $(I,\beta,\Sigma) \models p$

The way in which the approach of [He 90] has been adapted in this thesis (by relating the similarities of PrT and CE nets, and forming the same links between CE nets and LTL as for PrT nets and first order LTL) allows the refutation proof procedures introduced in [He 90] to be used. The use of these procedures will be illustrated in subsequent chapters of this thesis. The following shows an example of the use of the extended translation algorithm by translating the CE net of the mutual exclusion problem defined in Figure 3.1 into temporal logic.

System dependent axiom

A1: $p1 \wedge p4 \wedge p7$

System dependent inference rules

IR1: $p1 \wedge \neg p2 \Rightarrow \mathrm{O}\,(\neg p1 \wedge p2)$

IR2: $p2 \wedge p7 \wedge \neg p3 \Rightarrow \mathrm{O}\,(\neg p2 \wedge \neg p7 \wedge p3)$

IR3: $p3 \wedge \neg p1 \wedge \neg p7 \Rightarrow \mathrm{O}\,(\neg p3 \wedge p1 \wedge p7)$

IR4: $p4 \wedge \neg p5 \Rightarrow \mathrm{O}\,(\neg p4 \wedge p5)$

IR5: $p5 \wedge p7 \wedge \neg p6 \Rightarrow \mathrm{O}\,(\neg p5 \wedge \neg p7 \wedge p6)$

IR6: $p6 \wedge \neg p4 \wedge \neg p7 \Rightarrow \mathrm{O}\,(\neg p6 \wedge p4 \wedge p7)$

3.5 Discussion

This chapter has surveyed approaches that combined the formalisms of Petri nets and temporal logic. Approaches that required a separate specification to be written in each formalism were deemed unsuitable because there appeared to be no way of checking the consistency of such specifications. However, approaches that integrated Petri nets and temporal logic in order to produce a unified technique, such as Suzuki's temporal Petri nets and He and Lee's translation between PrT nets and temporal logic, were considered worthy of further examination.

Temporal Petri net analysis is centred on an event-based logic that formalises the firing of transitions based on a set of propositions, such as Proposition 1. This work was extended by the introduction of a less restrictive proposition, Proposition 2, which allows the technique to be applied to a wider range of Petri nets. Temporal Petri nets allow specification of properties such as fairness and eventuality (as shown in the mutual exclusion problem of Section 3.3.1); this could not have been achieved using Petri net theory. For instance, for the Petri net model of Figure 3.1, the notation of temporal Petri nets developed temporal formulas that prevented a process from accessing the common resource forever but allowed each process to access this resource. However, using the notation of Petri net theory such restrictions, or behaviours, could not be specified, they could only be implicitly assumed when constructing the reachability graph. Although temporal Petri nets allow liveness properties to be proved, they cannot prove safety properties, which can only be checked by examining the structure of the Petri net or its reachability graph.

The unified technique proposed by He and Lee (for integrating PrT nets and first order LTL) is centred on a state-based logic that formalised the initial marking of the net and the transition firing rules in terms of submarkings. This approach allowed the proof of both safety and liveness properties. Moreover, this technique is more suited to describing complex systems because the notation of PrT nets allows the modelling of large concurrent systems. Since PrT nets do not incorporate the notion of time (or in any of its extensions), the technique of He and Lee was considered to be unsuitable for describing hard real-time systems. As a consequence, rather than extending PrT nets with time, the approach of He and Lee was adapted to combine CE nets and propositional LTL. This adaptation had the advantages of:

(i) smaller systems could be described without hiding the essential functions of the system, which may be the case if PrT nets are used, and
(ii) time or timed Petri net analysis could be used to analyse properties of hard real-time systems.

A close examination of the temporal Petri nets of Suzuki and the approach of He and Lee show that, although the former is centred on an event-based logic (which can prove liveness properties), and the latter uses a state-based logic (allowing proof of both safety and liveness properties), both techniques make similar assertions about Petri nets. For instance the assumptions made in Proposition 1, of temporal Petri nets, are also implicitly made in the extended translation algorithm of He and Lee. This can be illustrated using the CE net of Figure 3.1 by applying each technique to the firing of an enabled transition at marking M_0. In this initial marking both transitions t_1 and t_4 are enabled. Since these transitions are not in conflict then the following will consider the firing of t_1.

(a) Temporal Petri net analysis

At M_0 Proposition 1 is applicable to the firing of both t_1 and t_4. Hence, for t_1 Proposition 1 asserts that:

if

(i) $<M_0,\alpha> \models \Box[t_1(ok) \Rightarrow t_1(ok)\,\mathcal{U}\,t_1]$ 3.11

and

(ii) $<M_0,\alpha> \models \Box[t_1(ok) \Rightarrow \Diamond t_1(\neg ok)]$ 3.12

then

(iii) $<M_0,\alpha> \models \Box[t_1(ok) \Rightarrow \Diamond t_1]$. 3.13

When t_1 fires, as described by Equ. 3.13, the next transitions to become firable are t_2 and t_4, hence, it can be deduced from Figure 3.1 that:

$$<M_0,\alpha> \models \Box[t_1 \Rightarrow O(t_2(ok) \wedge t_4(ok))]. \quad 3.14$$

Combining Equs. 3.13 and 3.14 yields:

$$<M_0,\alpha> \models \Box[t_1(ok) \Rightarrow \Diamond(t_2(ok) \wedge t_4(ok))]. \quad 3.15$$

(b) Extended approach of He and Lee

Applying the extended translation algorithm to Figure 3.1 (see Section 3.4.1.2) the following can be written for transition t_1:

(i) system dependent axiom

A1: $p1 \wedge p4 \wedge p7$,

(ii) the applicable system dependent inference rule for t_1 is:

IR1: $p1 \wedge \neg p2 \;\Rightarrow\; O(\neg p1 \wedge p2)$.

Since A1 satisfies the antecedent for IR1 (note: A1 also satisfies the antecedent of IR4, which is concerned with the firing of t_4, however, since the interleaving model is used to fire Petri net transitions then only the firing of t_1 is considered), the application of A1 and IR1 yields the following marking sequence:

(iii) $p1 \wedge p4 \wedge p7 \;\Rightarrow\; O(p2 \wedge p4 \wedge p7)$ 3.16

Comparing the formulas of (a) and (b) shows that the premises of Proposition 1 in (a) occur implicitly in (b). For instance:

(i) Equ. 3.12 can be combined with formulas such as: $[(p1 \wedge \neg p2) \Rightarrow t_1(\mathrm{ok})]$ and $[t_1(\neg \mathrm{ok}) \Rightarrow (p2 \wedge \neg p1)]$ to yield:

$$<M_0,\alpha> \models \Box[(p1 \wedge \neg p2) \Rightarrow \Diamond(p2 \wedge \neg p1)] \qquad 3.17$$

hence, it can be seen that Equ. 3.17 is an equivalent but weak form of IR1,

(ii) both A1 and IR1 assert the conclusion of Proposition 1 ie Equ. 3.13. For instance, the markings in the pre and postconditions of Equ. 3.16 assert that:
In marking $p1 \wedge p4 \wedge p7$ transitions t_1 and t_4 are firable, and in markings $p2 \wedge p4 \wedge p7$ transitions t_2 and t_4 are firable. Hence, transforming the premise of Equ. 3.16 in terms of the event-based logic, the following can be obtained:

$$<M_0,\alpha> \models \Box[(t_1(\mathrm{ok}) \wedge t_4(\mathrm{ok})) \Rightarrow O(t_2(\mathrm{ok}) \wedge t_4(\mathrm{ok}))] \qquad 3.18$$

which can be simplified to

$$<M_0,\alpha> \models \Box[t_1(\mathrm{ok}) \Rightarrow O(t_2(\mathrm{ok}) \wedge t_4(\mathrm{ok}))] \qquad 3.19$$

Equ. 3.19 is essentially a strong form of Equ. 3.15.

Chapter 4

Application of Petri net based techniques

4.1 Introduction

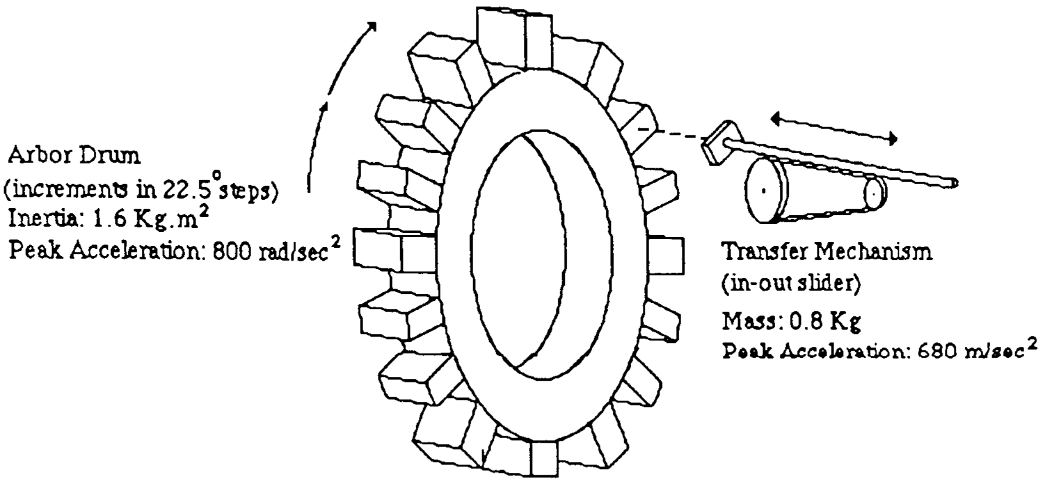
Previous chapters concentrated on surveying techniques suitable for specifying and verifying hard real-time systems. From this survey Petri net based techniques were selected for further examination because they allow concurrency to be represented by a simple and intuitive graphical notation, and coupled with techniques such as temporal logic they allow time related behaviour to be captured. Two principle techniques were extended to allow their application to a range of simple real-time problems.

The aim of this chapter is to evaluate the application of several Petri net based techniques to a hard real-time control problem. This allows the modelling and analytical capabilities of each technique to be examined and compared. Specifically, the techniques considered are timed Petri nets (described in Section 2.3.1.2), temporal Petri nets (described and modified in Section 3.3) and the extended technique of He and Lee (described and extended in Section 3.4.1) - which will be referred to as extended temporal Petri nets. It has been shown in Section 2.3.1.2 that timed Petri nets are suitable for modelling real-time systems and their analysis procedure can be used effectively for evaluating the performance of systems. Similarly, temporal and extended temporal Petri nets are suitable for specifying and verifying properties of concurrent systems. Although these techniques have been used to model classical problems, such as the dining philosophers and mutual exclusion, they have not been used to model real-time systems. Hence, in addition to providing an evaluation of these techniques, this chapter shows how effective these techniques are for modelling hard real-time systems.

4.1.1 Specification of the hard real-time control problem

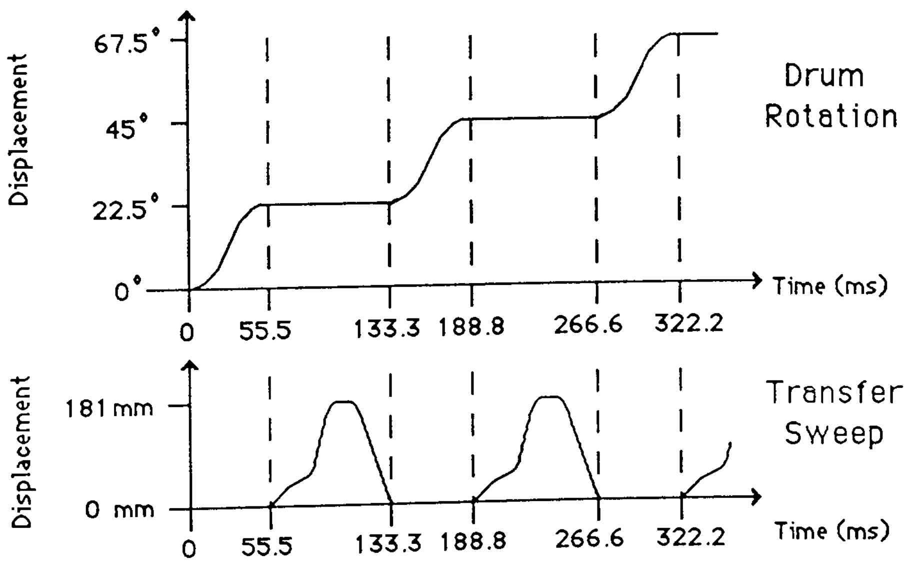
To evaluate the above techniques, this thesis considers a hard real-time control problem which originates from the design of a high-speed packaging machine [Fenney 88]. The problem concerns the design of synchronization logic for a time-critical real-time system which consists of an incremental arbor drum and an intermittent transfer slider as shown in Figure 4.1.

Figure 4.1 Arbor drum and transfer slider


Arbor Drum
(increments in 22.5° steps)
Inertia: 1.6 Kg.m²
Peak Acceleration: 800 rad/sec²
Transfer Mechanism
(in-out slider)
Mass: 0.8 Kg
Peak Acceleration: 680 m/sec²

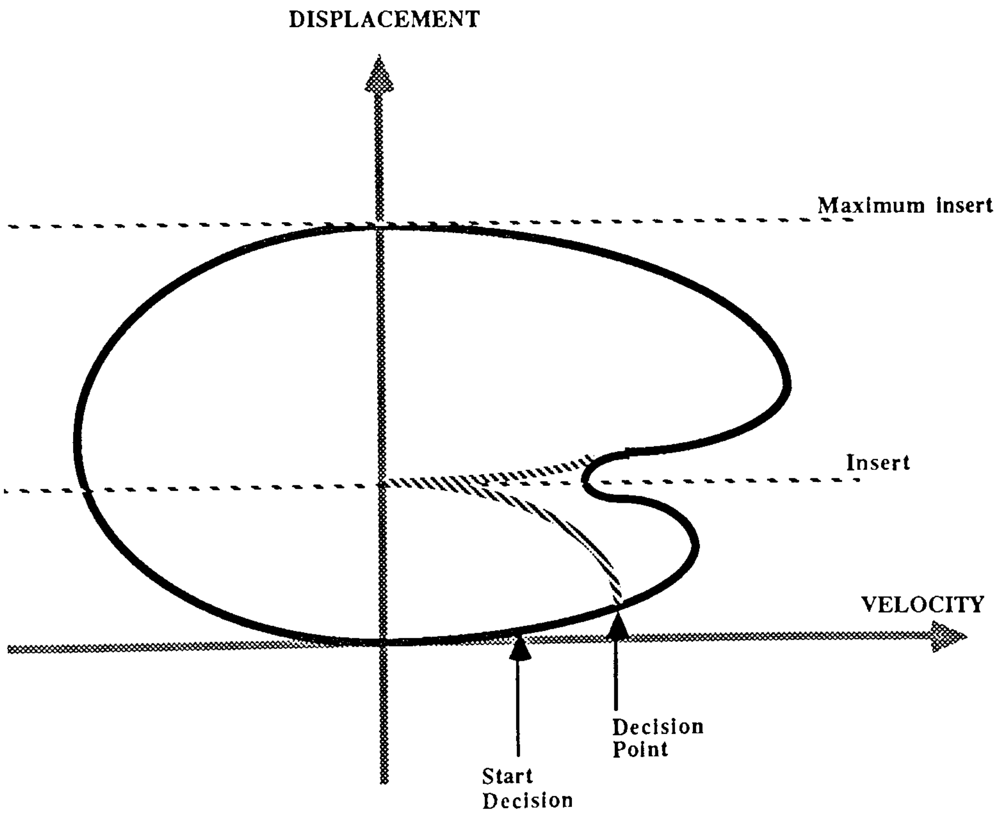
The drum is required to increment in 22.5⁰ steps and to dwell between increments with tight constraints on angular position. The slider is required to move into an arbor following a tightly constrained motion profile. In addition, each interacting mechanism has to meet tight constraints on positional accuracy or velocity, and at the normal operating speed of 450 cycles per minute, the peak velocities and accelerations of the drum and arbor are 14 rads^{-1}, 800 rads^{-2} and 8 ms^{-1} and 680 ms^{-2} respectively. The motion profiles which the arbor drum and transfer slider must exhibit are shown in Figure 4.2.

Figure 4.2 Arbor drum and transfer slider motion profiles



The principal role of the control system is to ensure the intermittent synchronization of the transfer slider and the drum. The asynchronous concurrent motion of the drum and slider is permitted. Clearly, the drum must be at rest and in position before the slider is inserted into the arbor. Similarly, the slider must be withdrawn from the arbor before the drum can rotate. The critical point in the slider's motion occurs when it has to make a decision (while in motion) either to continue moving and insert in the arbor (if the drum is stationary and in position) or to decelerate and stop (if the drum is rotating or not in position). This safety-critical decision must be computed in a timely manner, as shown in Figure 4.3. The following sections will model this problem and demonstrate the type of analysis that can be performed using the techniques mentioned in Section 4.1. Much of the modelling and analysis work to be presented has appeared in [Sagoo 90], [Sagoo 91], [Sagoo 92a], [Holding 92].

Figure 4.3 Motion profile and time-critical decision



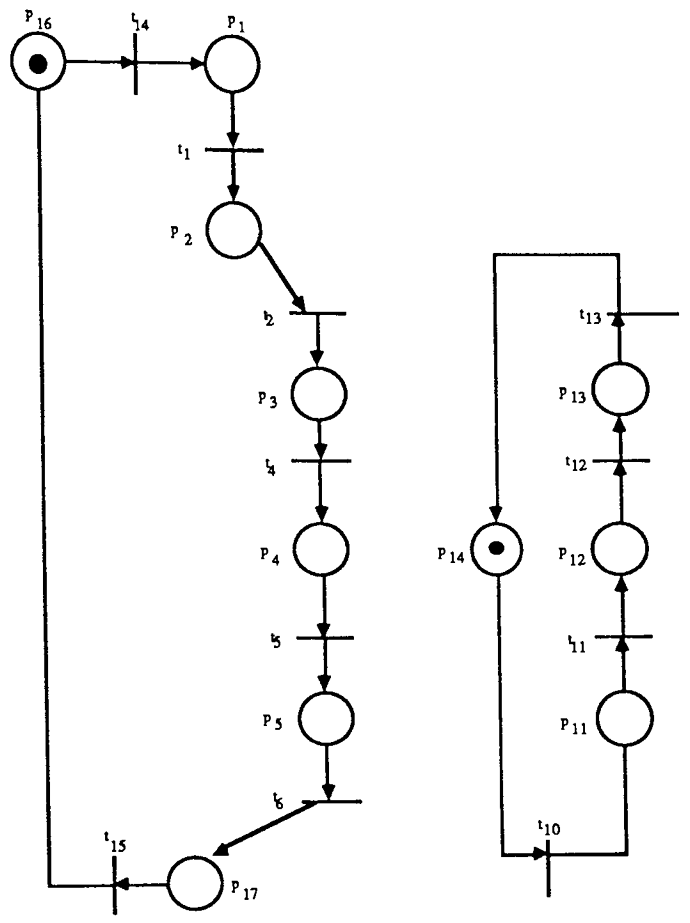
4.2 Development of a Petri net model for the hard real-time problem

The production of a Petri net model of the problem can be tackled in two stages. The first stage involves the development of a Petri net model which captures the simple asynchronous cyclic behaviour of the drum and slider and the simple synchronization logic required to coordinate their motion. The second stage considers timing aspects of the motion of the drum and slider and the synchronization decision processes. Specifically, timed Petri nets are used rather than time Petri nets because the former models the occurrence of events by a fixed delay instead of an event occurring within a range of delays. Several types of timed Petri nets exist and two methods are considered to investigate the expressive power of each techniques model of time. For comparative purposes, the simplest form of timed Petri nets developed by Ramchandani [Ramchandani 74], and the timed Petri nets of Razouk and Phelps [Razouk 85], which use a more elaborate model of time, are used.

4.2.1 Petri net model of the motion of the mechanisms

The motion of the drum and slider can be modelled as two simple asynchronous cyclic concurrent processes. The Petri net of Figure 4.4 shows a possible representation of these processes, in which the motion of the drum and the insert motion of the slider are modelled using information abstracted from Figures 4.2 and 4.3. Specifically, in Figure 4.4 the places represent conditions that model phases in the motion of the mechanisms and transitions represent events that cause changes in these phases. The semantics of the places and transitions are summarised in the Tables of Figure 4.4. Since the physical system has inertia, the model includes implicit information about the position and velocity of the system. For example, place p_2 represents the point at which the safety-critical decision must be taken at an instant when the slider is still moving towards the arbor at considerable velocity, thus the decision must be computed and implemented in a timely manner.

Figure 4.4 Petri net models showing the motions of the drum and slider



Slider's Places & Transitions	Semantics
P_1	Slider accelerating towards drum
P_2	Slider at decision point
P_3	Slider insert motion profile
P_4	Slider fully inserted
P_5	Slider withdrawing from full insertion
P_{16}	Slider at rest
P_{17}	Slider decelerates to rest
t_1	Slider enters decision point
t_2	Slider decides to proceed with insertion
t_4	Slider inserts into drum
t_5	Slider starts to withdraw from insertion
t_6	Slider clears drum
t_{14}	Slider starts to move
t_{15}	Slider reaches zero velocity

Drum's Places & Transitions	Semantics
P_{11}	Drum accelerating
P_{12}	Drum rotating at constant velocity
P_{13}	Drum decelerating
P_{14}	Drum stationary
t_{10}	Drum starts to rotate
t_{11}	Drum reaches constant velocity
t_{12}	Drum starts to decelerate
t_{13}	Drum stops rotating

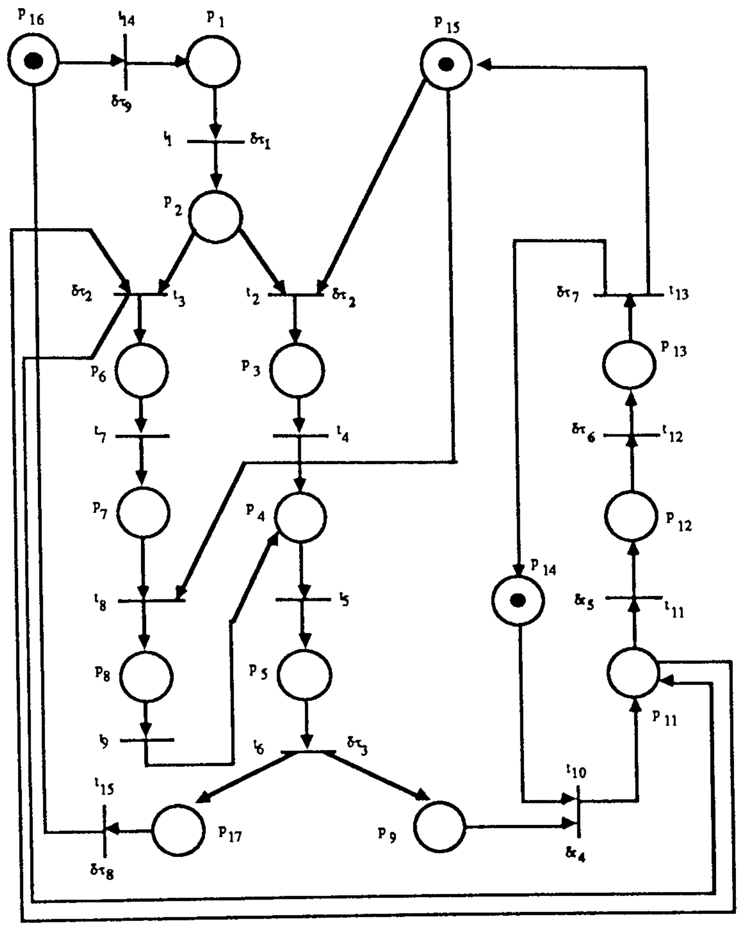
In order to coordinate the motion of the drum and slider so that they do not collide, it is necessary to augment the Petri net of Figure 4.4 with the appropriate control structures. Using the motion profile of Figure 4.3, the Petri net of Figure 4.5(a) has been produced, which is considered to achieve the desired coordination of the mechanisms. In this net the mechanisms are controlled by the following net structures:

(i) slider insert decision mechanism represented by the 'insert' decision (t_2) and the 'abort' decision (t_3),
(ii) synchronization logic for the insert decision represented by the insert interlock (p_{15}),
(iii) abort mechanisms (given by p_6, p_7, p_8 and t_3, t_7, t_8, t_9),
(iv) synchronization logic for the abort decision represented by the availability of a token in p_{11},
(v) drum rotate decision represented by t_{10},
(vi) synchronization logic for the drum rotate decision represented by the rotate interlock p_9.

The semantics of each place and transition in the net are also shown in Figure 4.5(a). However, before the Petri net of Figure 4.5(a) can be considered to be a 'correct' representation of the system being modelled, analysis must be performed. Such analysis can be based on the following approaches:

(i) enumerating the Petri net's reachability graph and checking the semantics of each reachable marking for the presence of hazardous or unsafe situations; these markings will be referred to as hazardous markings. Examples of hazardous situations are those that show either a collision of the mechanisms or those that lead to a collision,
(ii) using timed Petri net analysis to establish the system's time related behaviour.

Figure 4.5(a) Petri net model showing the synchronization logic, and motion of the drum and slider.



Places	Semantics
p_1	Slider accelerating towards drum
p_2	Slider at decision point
p_3	Slider inserting drum
p_4	Slider fully inserted
p_5	Slider withdrawing
p_6	Slider decelerating
p_7	Slider at rest
p_8	Slider accelerating
p_9	Drum rotate status
p_{11}	Drum accelerating
p_{12}	Drum rotating at constant velocity
p_{13}	Drum decelerating
p_{14}	Drum stationary
p_{15}	Drum stationary status
p_{16}	Slider at rest
p_{17}	Slider's motion to rest

Transition	Semantics	Time Parameters
t_1	Slider enters decision point	$\delta\tau_1$
t_2	Slider proceeds with insertion	$\delta\tau_2$
t_3	Slider starts to abort motion	$\delta\tau_2$
t_4	Slider makes an insertion into drum	
t_5	Slider starts to withdraw from insertion	
t_6	Slider clears drum	$\delta\tau_3$
t_7	Slider comes to rest	
t_8	Slider starts its motion	
t_9	Slider makes an insertion into drum	
t_{10}	Drum starts to rotate	$\delta\tau_4$
t_{11}	Drum reaches constant velocity	$\delta\tau_5$
t_{12}	Drum starts to decelerate	$\delta\tau_6$
t_{13}	Drum stops rotating	$\delta\tau_7$
t_{14}	Slider commences its motion	$\delta\tau_9$
t_{15}	Slider reaches zero velocity	$\delta\tau_8$

4.2.1.1 Reachability graph analysis

An inspection of the markings and the semantics of the reachability graph shown in Figure 4.5(b) reveals no hazardous markings, since no place contains combinations of p_3, p_4, p_5 representing a slider insert operation and p_{11}, p_{12}, p_{13} representing the motion of the drum. Thus, this static (place-by-place) analysis appears to show that no collision will take place.

However, further examination of Figure 4.5(b) reveals that the sequence of markings (M_{19}, M_{21}, M_{23}) requires further analysis. Specifically, M_{19} represents the situation in which the drum is rotating, the slider is in motion and is at the decision point. Also the decision to abort t_3 is enabled, as is the drum event t_{11}. Clearly, a timely decision is necessary in markings subsequent to M_{19} in order to prevent a collision. Since t_3 and t_{11} are in conflict, if t_{11} fires leading to marking M_{21} then the abort decision is disabled and the slider decision mechanism is unable to take a decision to insert or abort until the drum event t_{12} has occurred and marking M_{23} is reached. If the slider remains in this 'no-decision' state (in which case it will continue to move towards the drum) and the drum is still rotating then the slider will collide with the drum. This motion is not explicitly modelled by Figure 4.5(a) and can only be inferred by inspecting the semantics of the appropriate markings. Hence, markings M_{19}, M_{21}, M_{23} form a safety-critical sequence and any net sequence that includes this sequence can lead to a collision of the mechanisms. It follows that this hazardous marking sequence (M_{haz}) is given by:

$$M_{haz} = (M_{19}, M_{21}, M_{23}) \tag{4.1}$$

where

$$M_{19} = \{p_2, p_{11}\}, M_{21} = \{p_2, p_{12}\} \text{ and } M_{23} = \{p_2, p_{13}\}$$

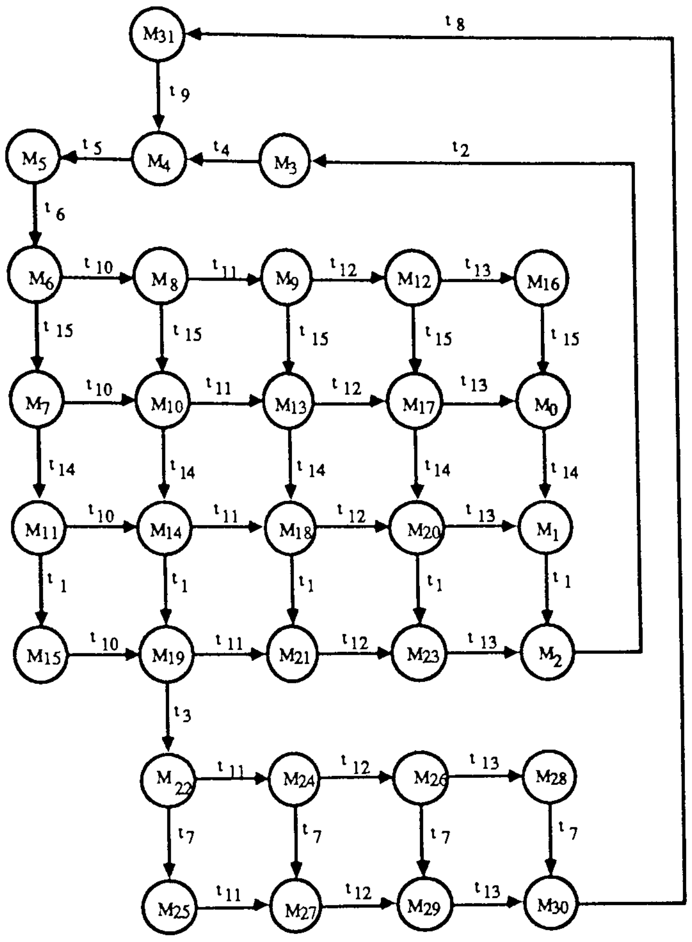
The above method of inspecting all reachable markings of the Petri net is clearly insufficient for determining whether the mechanisms will collide. This is because the slider's decision making operation is both time dependent and time-critical and Petri net analysis does not include timing information. Therefore, the hazard analysis of a Petri net must comprise two phases:

(i) static analysis of reachable markings to detect the occurrence of hazardous combination of states,

(ii) for time-critical systems, the timing analysis of sequences of markings to determine the ability of the system to respond in a timely manner.

The timing analysis of such Petri nets is considered in the following sections.

Figure 4.5(b) Reachability graph for the Petri net model of Figure 4.5(a)



MARKING	PLACES
M_0	P_{14}, P_{15}, P_{16}
M_1	P_1, P_{14}, P_{15}
M_2	P_2, P_{14}, P_{15}
M_3	P_3, P_{14}
M_4	P_4, P_{14}
M_5	P_5, P_{14}
M_6	P_9, P_{14}, P_{17}
M_7	P_9, P_{14}, P_{16}
M_8	P_{11}, P_{17}
M_9	P_{12}, P_{17}
M_{10}	P_{11}, P_{16}
M_{11}	P_1, P_9, P_{14}
M_{12}	P_{13}, P_{17}
M_{13}	P_{12}, P_{16}
M_{14}	P_1, P_{11}
M_{15}	P_2, P_9, P_{14}

MARKING	PLACES
M_{16}	P_{14}, P_{15}, P_{17}
M_{17}	P_{13}, P_{16}
M_{18}	P_1, P_{12}
M_{19}	P_2, P_{11}
M_{20}	P_1, P_{13}
M_{21}	P_2, P_{12}
M_{22}	P_6, P_{11}
M_{23}	P_2, P_{13}
M_{24}	P_6, P_{12}
M_{25}	P_7, P_{11}
M_{26}	P_6, P_{13}
M_{27}	P_7, P_{12}
M_{28}	P_6, P_{14}, P_{15}
M_{29}	P_7, P_{13}
M_{30}	P_7, P_{14}, P_{15}
M_{31}	P_8, P_{14}

4.2.1.2 Timed Petri net analysis

A timed Petri net can be easily derived from a Petri net by assigning time parameters to each of its transition. For the net of Figure 4.5(a) time parameters $\delta\tau_1$ - $\delta\tau_9$ are assigned to those transitions that are considered to play an important role in the motion of the mechanisms. Various time related behaviours of the net can be explored by forming relationships between the above time parameters. For example, the behaviour of the net may be assessed for various values of the time parameters, or it is possible to derive time constraints which impose a particular type of behaviour on the net. The following shows how these time parameters can be used to constrain the behaviour of the net to produce the desired motion and to avoid undesirable markings.

Consider the desired motion of the mechanisms to be such that, whenever the slider commences its motion it will always decide to insert into the drum, which must be stationary and in position. From the initial marking $M_0 = \{p_{14}, p_{15}, p_{16}\}$ this requires that when the decision point p_2 is subsequently reached then p_{15} must be tokenised. This is naturally satisfied on the first occurrence of the decision (see initial marking) and on subsequent occurrences it requires t_{13} to fire and tokenise p_{15} before t_1 fires and tokenises p_2. From inspection of the reachability graph and backward-chain reasoning this involves consideration of all initial sequences from marking $M_6 = \{p_9, p_{14}, p_{17}\}$ to marking $M_2 = \{p_2, p_{14}, p_{15}\}$ in order to avoid the marking sequence (M_{15}, M_{19}, M_{21}, M_{23}, M_2).

If this requirement is translated in terms of the time parameters then the following relational expressions of relative timing are obtained:

$$\delta\tau_1 + \delta\tau_8 + \delta\tau_9 \geq \delta\tau_4 + \delta\tau_5 + \delta\tau_6 + \delta\tau_7 \qquad 4.2$$

Assuming that time constraints of $\delta\tau_1$, $\delta\tau_8$, and $\delta\tau_5$, $\delta\tau_6$, $\delta\tau_7$, which represents the motion of the slider and drum respectively, can be related to physical timing constraints, then Equ. 4.2 can be simplified to:

$$\delta\tau_9 + k_1 \geq \delta\tau_4 + k_2 \qquad 4.3$$

where

$$k_1 = \delta\tau_8 + \delta\tau_1 \quad \text{and } k_2 = \delta\tau_5 + \delta\tau_6 + \delta\tau_7$$

From Equ. 4.3, it follows that $\delta\tau_4$ (the drum enable time) and $\delta\tau_9$ (the slider rest period) are parameters which can be controlled, such as by software, and thus they form the main design parameters of the system. Hence, using this type of analysis various system constraints can be applied to constrain the behaviour of the Petri net. The timing requirements are determined as follows:

(i) if the drum starts to rotate before the slider comes to rest (ie t_{10} fires before t_{15} thus $\delta\tau_8 > \delta\tau_4$), then the timing requirement for the desired motion is:

$$\delta\tau_1 + \delta\tau_8^* + \delta\tau_9 \geq \delta\tau_5 + \delta\tau_6 + \delta\tau_7 \qquad 4.4$$

where

$\delta\tau_8^*$ is the remaining time of $\delta\tau_8$ at the instant that t_{10} fires

(ii) if the slider comes to rest before the drum starts rotating (ie t_{15} fires before t_{10} thus $\delta\tau_8 < \delta\tau_4$), then the timing requirements is:

$$\delta\tau_1 + \delta\tau_9 \geq \delta\tau_4^* + \delta\tau_5 + \delta\tau_6 + \delta\tau_7 \qquad 4.5$$

where

$\delta\tau_4^*$ is the remaining time of $\delta\tau_4$ at the instant that t_{15} has fired.

If the requirement of Equ. 4.5 is too stringent and is not satisfied, then for the abort decision to occur the less stringent requirement must be ensured:

$$\delta\tau_1 + \delta\tau_3 + \delta\tau_9 \leq \delta\tau_4^* + \delta\tau_5 \qquad 4.6$$

Following failure to satisfy Equs. 4.4 and 4.5, the 'catch all' Equ. 4.6 will prevent the occurrence of the markings in M_{haz}. This situation causes the slider to abort motion and ensures that the mechanisms do not collide. In cases, (i) and (ii), time parameter $\delta\tau_1$ can be varied to produce the desired motion.

4.2.2 Modification of the Petri net model to remove hazardous sequences

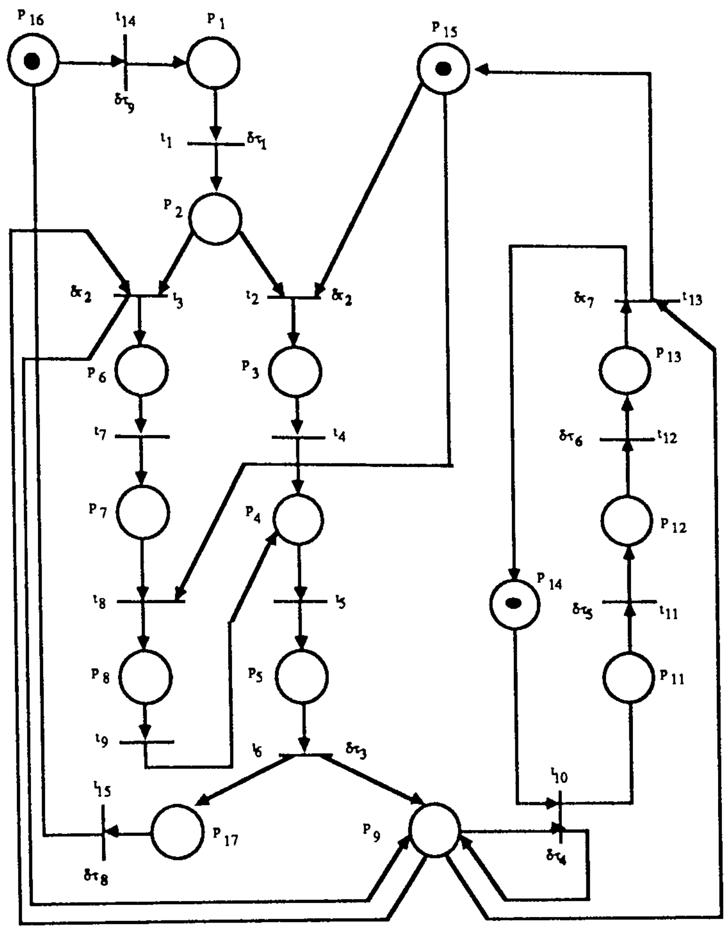
The above analysis has shown that the Petri net of Figure 4.5(a) contains hazardous time-critical marking sequences that can only be prevented from occurring by imposing timing constraints using timed Petri net analysis. However, rather than relying on timing constraints to guarantee the safe operation of the system, it is preferable to re-design the model in order to eliminate these time dependencies. Thus, using the above analysis an iterative procedure was used to devise a means of preventing the hazardous marking sequences of Figure 4.5(a). This resulted in the development of the new Petri net of Figure 4.6(a).

The essential differences between the Petri nets of Figure 4.5(a) and Figure 4.6(a) are:

(i) the latter uses a sampling point from p_9 for its abort transition t_3, whereas the former uses p_{11},

(ii) the semantics of p_9 differ in these nets. For instance, in Figure 4.5(a) p_9 is referred to as 'the drum rotate status' and, in conjunction with p_{14}, acts to enable the drum to rotate. However, in Figure 4.6(a) p_9 is referred to as 'the drum rotating status' and is interpreted as either representing that the drum is rotating (ie drum is at p_{11}, p_{12} or p_{13}), or that the drum is rotate enabled and will subsequently start rotating. Hence, in Figure 4.6(a) p_9 has a dual role of indicating the drum is rotating or that it is stationary and will rotate.

Figure 4.6(a) Modified version of Figure 4.5(a)



Places	Semantics
p_1	Slider accelerating towards drum
p_2	Slider at decision point
p_3	Slider inserting drum
p_4	Slider fully inserted
p_5	Slider withdrawing
p_6	Slider decelerating
p_7	Slider at rest
p_8	Slider accelerating
p_9	Drum rotating status
p_{11}	Drum accelerating
p_{12}	Drum rotating at constant velocity
p_{13}	Drum decelerating
p_{14}	Drum stationary
p_{15}	Drum stationary status
p_{16}	Slider at rest
p_{17}	Slider's motion to rest

Transition	Semantics	Time Parameters
t_1	Slider enters decision point	$\delta\tau_1$
t_2	Slider proceeds with insertion	$\delta\tau_2$
t_3	Slider starts to abort motion	$\delta\tau_2$
t_4	Slider makes an insertion into drum	
t_5	Slider starts to withdraw from insertion	
t_6	Slider clears drum	$\delta\tau_3$
t_7	Slider comes to rest	
t_8	Slider starts its motion	
t_9	Slider makes an insertion into drum	
t_{10}	Drum starts to rotate	$\delta\tau_4$
t_{11}	Drum reaches constant velocity	$\delta\tau_5$
t_{12}	Drum starts to decelerate	$\delta\tau_6$
t_{13}	Drum stops rotating	$\delta\tau_7$
t_{14}	Slider commences its motion	$\delta\tau_9$
t_{15}	Slider reaches zero velocity	$\delta\tau_8$

4.2.2.1 Analysis of modified Petri net model

The analysis of this Petri net was based on the same approach as illustrated in Section 4.2.1 and the reachability graph for Figure 4.6(a) is shown in Figure 4.6(b). Analysis of the reachable markings and their semantics show that there are no hazardous markings. However, analysis of the marking sequences shows that a hazardous situation can arise in the modelled system for the following time-critical sequence:

$$M_{15}, M_{19}, M_{22}, M_{25}, M_2 \qquad 4.7$$

The interpretation of Equ. 4.7 shows that it is hazardous because the slider is at the decision point, but remains in a 'no-decision' state whilst the drum makes progress in its motion. This type of sequence is 'unfair' because both the drum and slider can progress in their motion, but only the drum actually progresses. A comparison of the marking sequence of Equ. 4.7 and M_{haz} of Equ. 4.1 shows that, whilst both sequences may lead to a collision of the mechanisms, for each marking in Equ. 4.7 the slider can make a decision (since t_3 is enabled) while in Equ. 4.1 markings M_{21} and M_{23} allow no decision to be made by the slider. Thus the sequence in the modified net is intrinsically safe. However, timing analysis must be performed to ascertain the timing constraints necessary to prevent the occurrence of the sequence represented by Equ. 4.7 or whether this sequence can occur without causing a collision of the mechanisms.

A timed Petri net for Figure 4.6(a) is obtained by assigning time parameters ($\delta\tau_1$ - $\delta\tau_9$) to those transitions which play an important role in the motion of the mechanisms. Thus, if it is required that the slider makes a decision to insert into the drum, which must be stationary and in position, then from marking $M_6 = \{p_9, p_{14}, p_{17}\}$ place p_{15} must be reached before place p_2 (ie it must be guaranteed that when marking M_6 is reached, marking $M_2 = \{p_2, p_{14}, p_{15}\}$ must eventually follow, rather than M_{15}, M_{19}, M_{22} or M_{25}). This can be expressed in terms of the timing requirement:

$$\delta\tau_1 + \delta\tau_8 + \delta\tau_9 \geq \delta\tau_4 + \delta\tau_5 + \delta\tau_6 + \delta\tau_7 \qquad 4.8$$

Equ. 4.8 specifies the same timing requirements as Equ. 4.2 and using the reasoning developed in Section 4.2.1.2 the same equations as Equ. 4.2, 4.3, 4.4, and 4.5 can be obtained for Figure 4.6(a). However, if Equ. 4.5 for the Petri net of Figure 4.6(a) imposes a requirement that cannot be satisfied then the slider must take an abort decision; in this case the timing requirement is:

$$\delta\tau_1 + \delta\tau_3 + \delta\tau_9 \leq \delta\tau_4^* + \delta\tau_5 + \delta\tau_6 + \delta\tau_7 \qquad 4.9$$

Equ. 4.9 will ensure that the slider takes an abort decision if the drum is rotating, ie at either position p_{11}, p_{14} or p_{17}. A comparison of the timing requirement of Equ. 4.9 for Figure 4.6(a) with Equ. 4.6 for Figure 4.5(a) shows that the former imposes a more relaxed timing requirement than the latter. This improved requirement has been brought about by changing the sampling point for the abort decision (t_3) from p_{11} in Figure 4.5(a) to p_9 Figure 4.6(a).

This section has shown that the Petri net models of Figures 4.5(a) and 4.6(a) both contain hazardous sequences, and timed Petri net analysis has shown that the timing requirements for these Petri nets are similar. However, this analysis has shown that the underlying Petri net model of Figure 4.6(a) is better than that of Figure 4.5(a). This is because whenever the slider is at the decision point it can decide whether to insert or abort, however, in the model of Figure 4.5(a) this is not possible.

Figure 4.6(b) Reachability graph for the Petri net model of Figure 4.6(a)

$$\begin{tikzcd}
 & M_{33} \arrow[d, "t_9"] & & & \\
M_5 \arrow[d, "t_6"] & M_4 \arrow[l, "t_5"'] & M_3 \arrow[l, "t_4"'] & & \\
M_6 \arrow[r, "t_{10}"] \arrow[d, "t_{15}"] & M_8 \arrow[r, "t_{11}"] \arrow[d, "t_{15}"] & M_9 \arrow[r, "t_{12}"] \arrow[d, "t_{15}"] & M_{12} \arrow[r, "t_{13}"] \arrow[d, "t_{15}"] & M_{16} \arrow[d, "t_{15}"] \\
M_7 \arrow[r, "t_{10}"] \arrow[d, "t_{14}"] & M_{10} \arrow[r, "t_{11}"] \arrow[d, "t_{14}"] & M_{13} \arrow[r, "t_{12}"] \arrow[d, "t_{14}"] & M_{17} \arrow[r, "t_{13}"] \arrow[d, "t_{14}"] & M_0 \arrow[d, "t_{14}"] \\
M_{11} \arrow[r, "t_{10}"] \arrow[d, "t_1"] & M_{14} \arrow[r, "t_{11}"] \arrow[d, "t_1"] & M_{18} \arrow[r, "t_{12}"] \arrow[d, "t_1"] & M_{21} \arrow[r, "t_{13}"] \arrow[d, "t_1"] & M_1 \arrow[d, "t_1"] \\
M_{15} \arrow[r, "t_{10}"] \arrow[d, "t_3"] & M_{19} \arrow[r, "t_{11}"] \arrow[d, "t_3"] & M_{22} \arrow[r, "t_{12}"] \arrow[d, "t_3"] & M_{25} \arrow[r, "t_{13}"] \arrow[d, "t_3"] & M_2 \arrow[uuuull, "t_2"'] \\
M_{20} \arrow[r, "t_{10}"] \arrow[d, "t_7"] & M_{23} \arrow[r, "t_{11}"] \arrow[d, "t_7"] & M_{26} \arrow[r, "t_{12}"] \arrow[d, "t_7"] & M_{28} \arrow[r, "t_{13}"] \arrow[d, "t_7"] & M_{30} \arrow[d, "t_7"] \\
M_{24} \arrow[r, "t_{10}"] & M_{27} \arrow[r, "t_{11}"] & M_{29} \arrow[r, "t_{12}"] & M_{31} \arrow[r, "t_{13}"] & M_{32} \arrow[uuuuuuulll, "t_8"']
\end{tikzcd}$$

MARKING	PLACES
M_0	$P_{14} \cdot P_{15} \cdot P_{16}$
M_1	$P_1 \cdot P_{14} \cdot P_{15}$
M_2	$P_2 \cdot P_{14} \cdot P_{15}$
M_3	$P_3 \cdot P_{14}$
M_4	$P_4 \cdot P_{14}$
M_5	$P_5 \cdot P_{14}$
M_6	$P_9 \cdot P_{14} \cdot P_{17}$
M_7	$P_9 \cdot P_{14} \cdot P_{16}$
M_8	$P_9 \cdot P_{11} \cdot P_{17}$
M_9	$P_9 \cdot P_{12} \cdot P_{17}$
M_{10}	$P_9 \cdot P_{11} \cdot P_{16}$
M_{11}	$P_1 \cdot P_9 \cdot P_{14}$
M_{12}	$P_9 \cdot P_{13} \cdot P_{17}$
M_{13}	$P_9 \cdot P_{12} \cdot P_{16}$
M_{14}	$P_1 \cdot P_9 \cdot P_{11}$
M_{15}	$P_2 \cdot P_9 \cdot P_{14}$
M_{16}	$P_{14} \cdot P_{15} \cdot P_{17}$
M_{17}	$P_9 \cdot P_{13} \cdot P_{16}$
M_{18}	$P_1 \cdot P_9 \cdot P_{12}$
M_{19}	$P_2 \cdot P_9 \cdot P_{11}$
M_{20}	$P_6 \cdot P_9 \cdot P_{14}$
M_{21}	$P_1 \cdot P_9 \cdot P_{13}$
M_{22}	$P_2 \cdot P_9 \cdot P_{12}$
M_{23}	$P_6 \cdot P_9 \cdot P_{11}$
M_{24}	$P_7 \cdot P_9 \cdot P_{14}$
M_{25}	$P_2 \cdot P_9 \cdot P_{13}$
M_{26}	$P_6 \cdot P_9 \cdot P_{12}$
M_{27}	$P_7 \cdot P_9 \cdot P_{11}$
M_{28}	$P_6 \cdot P_9 \cdot P_{13}$
M_{29}	$P_7 \cdot P_9 \cdot P_{12}$
M_{30}	$P_6 \cdot P_{14} \cdot P_{15}$
M_{31}	$P_7 \cdot P_9 \cdot P_{13}$
M_{32}	$P_7 \cdot P_{14} \cdot P_{15}$
M_{33}	$P_8 \cdot P_{14}$

4.2.3 Using timed Petri nets of Razouk and Phelphs

The implications of using a more elaborate model of time in timed Petri nets to that used in Section 4.2.2 will be shown in this section. In particular, the Petri net of Figure 4.6(a) will be transformed into a timed Petri net of Razouk and Phelps (see Section 2.3.1.2). This is achieved by allocating two variable time parameters: enabling time (δe) and firing time (δf) to each transition in the Petri net. Table 4.1 summarises the allocation of these times to the transitions of Figure 4.6(a).

Table 4.1 Allocation of enabling and firing times to transitions of Figure 4.6(a)

Transition	Enabling time (τ_e)		Firing time (τ_f)	
t_1	τ_{e1}	Slider acceleration time	τ_{f1}	Time taken to reach decision point
t_2	τ_{e2}	Time taken to make decision	τ_{f2}	Time taken to adjust motion profile for insertion
t_3	τ_{e3}	Time taken to make decision	τ_{f3}	Time taken to start deceleration
t_4	τ_{e4}	Time taken to dec. according to motion profile	τ_{f4}	Time taken to adopt insertion motion profile
t_5	τ_{e5}	Time taken to fully insert	τ_{f5}	Time taken to start deceleration motion
t_6	τ_{e6}	Time taken to clear drum	τ_{f6}	Time taken to reach withdrawal velocity
t_7	τ_{e7}	Time taken to decelerate	τ_{f7}	Time taken to move slider due to inertia
t_8	τ_{e8}	Time taken to enable slider to move	τ_{f8}	Time taken to reach velocity for insertion
t_9	τ_{e9}	Time taken to achieve required acceleration	τ_{f9}	Time taken to adopt insertion motion profile
t_{10}	τ_{e10}	Time taken to enable drum to rotate	τ_{f10}	Time taken to move drum due to inertia
t_{11}	τ_{e11}	Time taken to accelearte drum to required velocity	τ_{f11}	Time taken to reach acceleration velocity
t_{12}	τ_{e12}	Constant velocity period of drum	τ_{f12}	Time taken to decelerate slider
t_{13}	τ_{e13}	Time taken for Drum to decelerate to zero velocity	τ_{f13}	Time taken for drum to be stationary
t_{14}	τ_{e14}	Slider stationary period	τ_{f14}	Time taken to accelerate slider
t_{15}	τ_{e15}	Time taken for slider to reach zero velocity	τ_{f15}	Time taken to stop

Since the firing of transitions in a timed Petri net of Razouk and Phelps consumes time, some places in Figure 4.6(a) could be included within the enabling and firing times of transitions. For example:

(i) place p_6 can be removed if the following semantics are assigned to the enabling and firing times to t_3,

δe_3 - time taken to make a decision,

δf_3 - time taken for the slider to become stationary, due to inertia.

Thus δf_3 subsumes the function of p_6,

(ii) place p_8 can be removed if the following semantics are assigned to enabling and firing times to t_8,

δe_8 - time taken to start motion of slider, due to inertia,

δf_8 - time taken to accelerate the slider.

Thus δf_8 subsumes the function of p_8,

(iii) p_{11} can be removed using the following assignment:

δe_{10} - time taken to cause the drum to be set in motion,

δf_{10} - time taken for the drum to reach constant velocity.

Thus δf_{10} subsumes the function of p_{11},

(iv) p_{13} can be removed by the following assignment:

δe_{13} - time taken to for the drum to remain at constant velocity,

δf_{13} - time taken for the drum to decelerate to reach zero velocity.

Thus δf_{13} subsumes the function of p_{13}.

If the above modifications (i) - (iv) are made to Figure 4.6(a) then the Petri net of Figure 4.7(a) is produced. A timed reachability graph for Figure 4.7(a) is shown in Figure 4.7(b). This reachability graph is generated using the criteria that, when an enabled transition fires tokens are absorbed from its input places, ie they 'disappear', thus creating an intermediate marking; these tokens re-appear in their appropriate output places when the transition actually fires.

An examination of the reachability graph of Figure 4.7(b) shows that it exhibits the same behaviours as the reachability graph of Figure 4.6(b). In particular, Figure 4.7(b) contains no hazardous markings, but contains a hazardous sequence of markings, represented by:

$$M_{12}, M^*_{21}, M_{15}, M^*_{25}, M_2 \qquad 4.10$$

The marking sequence of Equ. 4.10 has the same interpretation as the marking sequence of Equ. 4.7 for Figure 4.6(a). Hence, the above has shown, by inspection that Petri net Figure 4.7(b) is equivalent to that of Figure 4.6(a).

In order to eliminate the marking sequence of Equ. 4.10 timed Petri net analysis is used to constrain the behaviour of Figure 4.7(a) to the desired behaviour (ie whenever the slider reaches the decision point then it will decide to insert into the drum which must be stationary and in position). Hence, the time relationships for the net of Figure 4.7(a) are calculated as follows.

For the Petri net of Figure 4.7(a) at initial marking $M_0 = \{p_{14}, p_{15}, p_{16}\}$, when marking $M_6 = \{p_9, p_{14}, p_{17}\}$ is reached, then in order to ensure the desired system behaviour the following constraint must be guaranteed:

$$(\delta e_1 + \delta f_1) + (\delta e_{14} + \delta f_{14}) + (\delta e_{15} + \delta f_{15}) \geq (\delta e_{10} + \delta f_{10}) + (\delta e_{13} + \delta f_{13}) \qquad 4.11$$

Further timing analysis can be performed to constrain the behaviour of the Petri net, for instance:

(i) if the drum starts to rotate before the slider becomes stationary (ie t_{10} fires after t_{15} thus $\delta e_{15} > \delta e_{10} + \delta f_{10}$), then in order to achieve the desired motion the following timing constraints must be satisfied:

$$\delta^* e_{15} + (\delta e_{14} + \delta f_{14}) + (\delta e_1 + \delta f_1) \geq \delta e_{13} + \delta f_{13} \qquad 4.12$$

where
$\delta^* e_{15} = \delta e_{15} - (\delta e_{10} + \delta f_{10})$ and represents the remaining enabling time of t_{15} at the instant t_{10} fires.

From the requirement of Equ. 4.12 the period for which the slider must remain at rest can be determined as:

$$\delta e_{14} \geq (\delta e_{13} + \delta f_{13}) - \delta^* e_{15} - \delta e_{14} - (\delta e_1 + \delta f_1) \qquad 4.13$$

(ii) if the drum becomes stationary before the drum starts to rotate (ie t_{15} fires before t_{10} thus $\delta e_{10} > \delta e_{15} + \delta f_{15}$) then in order to achieve the desired motion the following timing constraints must be satisfied:

$$(\delta e_{14} + \delta f_{14}) + (\delta e_1 + \delta f_1) \geq (\delta^* e_{10} + \delta f_{10}) + (\delta e_{13} + \delta f_{13}) \qquad 4.14$$

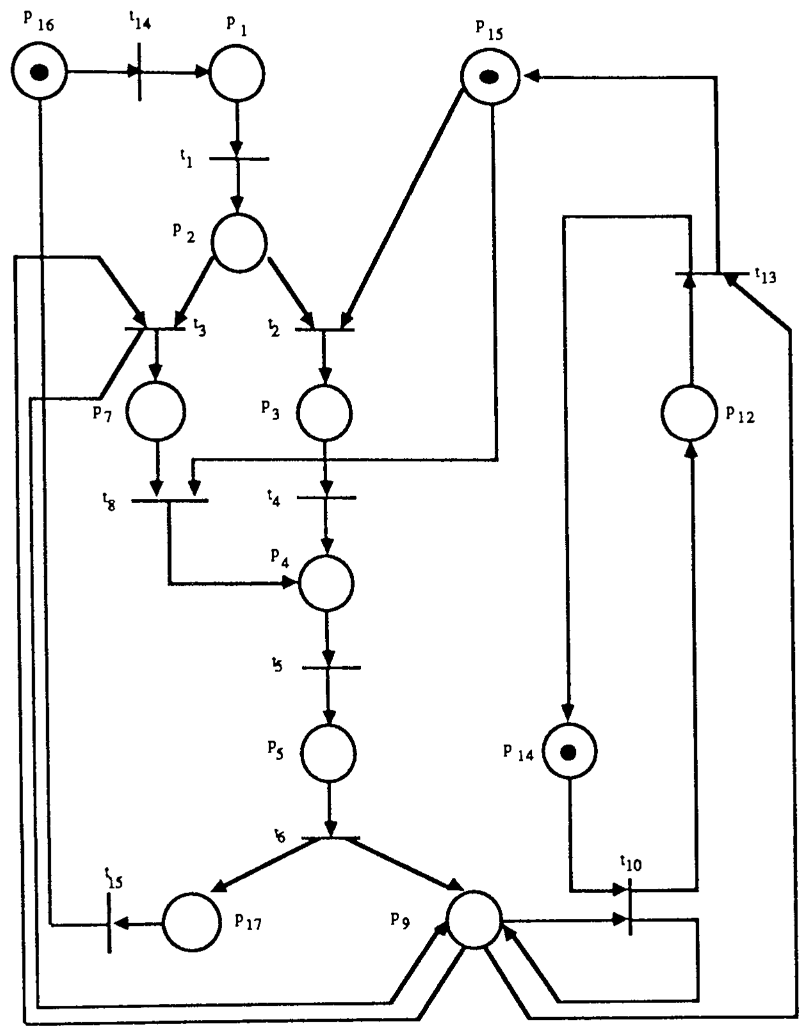
where
$\delta^* e_{10} = \delta e_{10} - (\delta e_{15} + \delta f_{15})$ and represents the remaining enabling time of t_{10} at the instant t_{15} fires.

From Equ. 4.14 the time for which the slider must remain at rest is:

$$\delta e_{14} \geq (\delta^* e_{10} + \delta f_{10}) + (\delta e_{13} + \delta f_{13}) - \delta f_{14} - (\delta e_1 + \delta f_1) \qquad 4.15$$

A comparison of the use of the timed Petri nets of Razouk and Phelps (shown above) and the timed Petri nets of Ramchandani (presented in Section 4.2.1.2) on Figure 4.6(a) has shown that both types of nets produced models of the system that exhibit equivalent behaviours; the same hazardous sequences existed in both nets. However, the main attribute of the timed Petri nets of Razouk and Phelps is that more accurate estimates of timing requirements can be obtained.

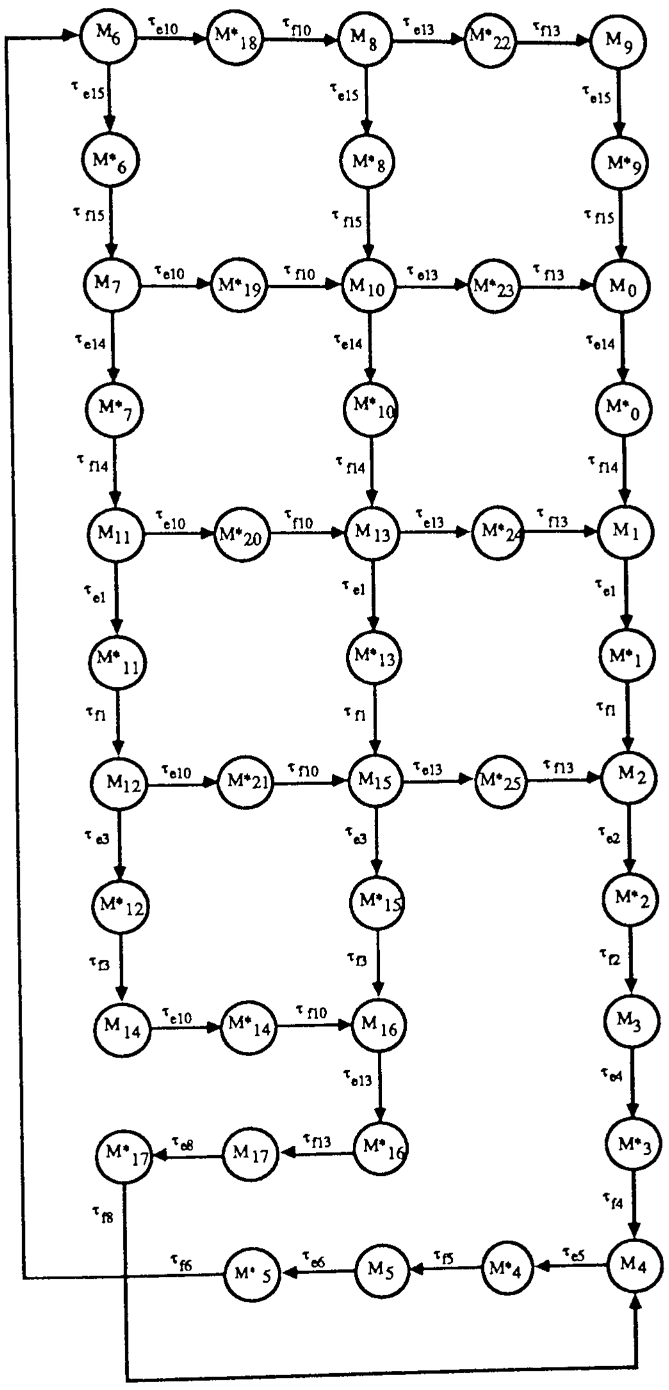
Figure 4.7(a) Petri net model using the timed Petri nets of Razouk and Phelps



Places	Semantics
P_1	Slider accelerating towards drum
P_2	Slider at decision point
P_3	Slider inserting drum
P_4	Slider fully inserted
P_5	Slider withdrawing
P_7	Slider at rest
P_9	Drum rotating status
P_{12}	Drum rotating at constant velocity
P_{14}	Drum stationary
P_{15}	Drum stationary status
P_{16}	Slider at rest
P_{17}	Slider's motion to rest

Transition	Semantics
t_1	Slider enters decision point
t_2	Slider proceeds with insertion
t_3	Slider starts to abort motion
t_4	Slider makes an insertion into drum
t_5	Slider starts to withdraw from insertion
t_6	Slider clears drum
t_8	Slider starts its motion
t_{10}	Drum starts to rotate
t_{13}	Drum stops rotating
t_{14}	Slider commences its motion
t_{15}	Slider reaches zero velocity

Figure 4.7(b) Reachability graph for the Petri net model of Figure 4.7(a)



MARKING	PLACES
M_0	p_{14}, p_{15}, p_{16}
M_1	p_1, p_{14}, p_{15}
M_2	p_2, p_{14}, p_{15}
M_3	p_3, p_{14}
M_4	p_4, p_{14}
M_5	p_5, p_{14}
M_6	p_9, p_{14}, p_{17}
M_7	p_9, p_{14}, p_{16}
M_8	p_9, p_{12}, p_{17}
M_9	p_{14}, p_{15}, p_{17}
M_{10}	p_9, p_{12}, p_{16}
M_{11}	p_1, p_9, p_{14}
M_{12}	p_2, p_9, p_{14}
M_{13}	p_1, p_9, p_{12}
M_{14}	p_7, p_9, p_{14}
M_{15}	p_2, p_9, p_{12}
M_{16}	p_7, p_9, p_{12}
M_{17}	p_7, p_{14}, p_{15}

MARKING	PLACES
M^*_0, M^*_1, M^*_9	p_{14}, p_{15}
M^*_2, M^*_3, M^*_4	p_{14}
$M^*_5, M^*_{12}, M^*_{17}$	p_{14}
M^*_6, M^*_7, M^*_{11}	p_9, p_{14}
$M^*_8, M^*_{10}, M^*_{13}$	p_9, p_{12}
M^*_{18}, M^*_{22}	p_{17}
M^*_{19}, M^*_{23}	p_{16}
M^*_{20}, M^*_{24}	p_1
M^*_{21}, M^*_{25}	p_2
M^*_{14}	p_7
M^*_{15}	p_{12}

4.3 Temporal Petri net analysis

Temporal Petri net analysis is based on setting up propositions (such as Propositions 1 and 2, which were detailed in Section 3.3.2) that make assertions about the firing of transitions in a Petri net. These propositions are applied by first checking the validity of their premises against the reachability graph of the net, and then using them to prove a class of properties known as liveness properties. This section will present a temporal Petri net analysis for verifying properties of the Petri net model of Figure 4.6(a). Since this analysis will be based on applying Propositions 1 and 2 they are re-stated as follows:

Proposition 1:
For a temporal Petri net TN = (C,f) whose initial marking is M_0.
If

(i) for any marking M reachable from M_0, if t is firable at M, then t remains firable until t itself fires. This statement can be formalised as:

$$<M_0,\alpha> \models \Box[t(ok) \Rightarrow t(ok)\mathcal{U}t] \qquad 4.16$$

and

(ii) f implies $<M_0,\alpha> \models \Box[t(ok) \Rightarrow \Diamond t(\neg ok)]$ 4.17
where f represents the condition that for an initial marking M_0 and a firing sequence α it is true that whenever t becomes firable then it will eventually become disabled.

Then
for any transition firing sequence (α) from M_0, using (i) and (ii) it can be deduced that:

(iii) $<M_0,\alpha> \models \Box[t(ok) \Rightarrow \Diamond t]$ 4.18

Proposition 1 is applicable to the firing of a basic transition, which fires and only disables itself. However, for the firing of conflicting transitions (in which the firing of a transition disables itself and another enabled transition) Proposition 2 applies and is defined as:

Proposition 2
For a temporal Petri net TN = (C,f) whose initial marking is M_0.
If

(i) for any marking M reachable from M_0, if t_1 (having input function $I(t_1)$) and t_2 (with input function $I(t_2)$) are firable at M and are in conflict, which is defined as:

$$I(t_1) \cap I(t_2) \neq \varnothing$$

where

$\varnothing$ is the empty set

then either t_1 remains firable until either t_1 itself fires or t_2 fires or t_2 remains firable until either t_2 itself fires or t_1 fires. This statement is formalised as:

$$<M_0,\alpha> \models \quad \Box[\, t_1(ok) \Rightarrow t_1(ok)\,\mathcal{U}(t_1 \vee t_2)] \vee$$
$$\Box[\, t_2(ok) \Rightarrow t_2(ok)\,\mathcal{U}(t_1 \vee t_2)] \qquad 4.19$$

(As before only one transition is allowed to fire at any instant because the interleaving model is used to represent concurrency and the choice of which fires is made non-deterministically or according to a fairness requirement if specified),

and

(ii) f implies

$$<M_0,\alpha> \models \Box[t_1(ok) \Rightarrow \Diamond t_1(\neg ok)] \wedge \Box[t_2(ok) \Rightarrow \Diamond t_2(\neg ok)]. \qquad 4.20$$

Then

for any transition firing sequence α from Mo, using (i) and (ii) it can be proved that:

(iii) $<M_0,\alpha> \models \Box[t_1(ok) \wedge t_2(ok) \Rightarrow \Diamond(t_1 \vee t_2)]$ 4.21

It must be emphasized that temporal Petri net analysis is not restricted to using the above propositions, however, the definition of temporal Petri nets allows propositions to be constructed that describe the firing of transitions that are unique to the net under consideration.

Typically, the validity of Propositions 1 and 2 is checked for a partial reachability graph that is relevant to the proof of the desired property. However, since the total reachability graph for the Petri net of Figure 4.6(a) is available, it can be seen by inspection that Proposition 1 applies to all transitions except for the transitions firable under marking $M_{25} = \{p2, p9, p13\}$. For this marking the firing of t_{13} disables itself and the enabled transition t_3, hence a form of Proposition 2 is applicable. An illustration of the proof method associated with temporal Petri nets analysis is shown by proving properties of Figure 4.6(a). Specifically, this proof method uses the temporal proof system developed by Manna and Pnueli [Manna 83b], which was also used by Suzuki and Lu [Suzuki 89].

Consider the following liveness properties of the Petri net of Figure 4.6(a):

Lemma 1

The slider decides to insert and subsequently withdraws from the drum. This liveness property is formalised as:

$$<M_0, \alpha> \models \Box[t_2 \Rightarrow \Diamond t_6] \qquad 4.22$$

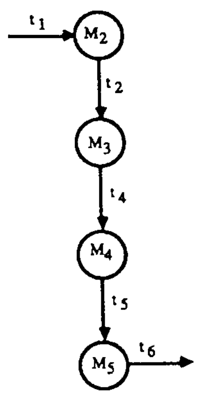
Proof

For Figure 4.6(a) the initial marking M_0 is defined by the set:

$$M_0 = \{p_{14}, p_{15}, p_{16}\}$$

From this marking the part of the reachability graph of Figure 4.6(b) that is relevant to the proof of Lemma 1 is shown in Figure 4.8.

Figure 4.8 Partial reachability graph of Figure 4.6(b)



MARKING	PLACES
M_2	p_2, p_{14}, p_{15}
M_3	p_3, p_{14}
M_4	p_4, p_{14}
M_5	p_5, p_{14}

Applying the propositions of temporal Petri nets to Figure 4.8 shows that for transitions t_4, t_5, and t_6 the condition of Proposition 1(i) holds and Proposition 1(ii) is satisfied by the following assertions:

1. $\langle M_0, \alpha \rangle \models \Box[t_4(ok) \Rightarrow \Diamond t_4(\neg ok)]$,
2. $\langle M_0, \alpha \rangle \models \Box[t_5(ok) \Rightarrow \Diamond t_5(\neg ok)]$,
3. $\langle M_0, \alpha \rangle \models \Box[t_6(ok) \Rightarrow \Diamond t_6(\neg ok)]$.

Therefore by Proposition 1(iii) it can be asserted that:

4. $\langle M_0, \alpha \rangle \models \Box[t_4(ok) \Rightarrow \Diamond t_4]$,
5. $\langle M_0, \alpha \rangle \models \Box[t_5(ok) \Rightarrow \Diamond t_5]$,
6. $\langle M_0, \alpha \rangle \models \Box[t_6(ok) \Rightarrow \Diamond t_6]$.

From Figure 4.8 it can be seen that when t_2 fires, then t_4 becomes firable at the next marking. This and similar observations about t_5 and t_6 yields:

7. $\langle M_0, \alpha \rangle \models \Box[t_2 \Rightarrow \bigcirc t_4(ok)]$,
8. $\langle M_0, \alpha \rangle \models \Box[t_4 \Rightarrow \bigcirc t_5(ok)]$,
9. $\langle M_0, \alpha \rangle \models \Box[t_5 \Rightarrow \bigcirc t_6(ok)]$.

From 4. and 7. using the temporal logic proof system yields:

10. $\langle M_0, \alpha \rangle \models \Box[t_2 \Rightarrow \Diamond t_4]$

From 5. and 8. using temporal logic yields:

11. $\langle M_0, \alpha \rangle \models \Box[t_4 \Rightarrow \Diamond t_5]$

From 6. and 9. using temporal logic yields:

12. $\langle M_0, \alpha \rangle \models \Box[t_5 \Rightarrow \Diamond t_6]$

From 10. - 12. using temporal logic yields:

11. $\langle M_0, \alpha \rangle \models \square[t_2 \Rightarrow \Diamond t_6]$

This completes the proof of Lemma 1 ■

An important liveness property about the slider asserts that:

Lemma 2

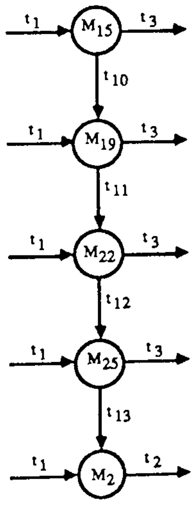
Whilst at the decision point the slider must make a decision to either abort or insert motion; this is expressed as:

$$\langle M_0, \alpha \rangle \models \square[p_2 \Rightarrow \Diamond(t_2 \vee t_3)] \qquad 4.23$$

Proof

In order to prove Lemma 2 it is sufficient to prove that for any marking sequence reachable from M_0 and a possibly infinite firing sequence α, that whenever p_2 is tokenised then either t_2 or t_3 will fire. The relevant portion of the reachability graph is shown in Figure 4.9.

Figure 4.9 Partial reachability graph of Figure 4.6(b)



MARKING	PLACES
M_2	$p_2 \cdot p_{14} \cdot p_{15}$
M_{15}	$p_2 \cdot p_9 \cdot p_{14}$
M_{19}	$p_2 \cdot p_9 \cdot p_{11}$
M_{22}	$p_2 \cdot p_9 \cdot p_{12}$
M_{25}	$p_2 \cdot p_9 \cdot p_{13}$

The following temporal formulas will represent the net marking in propositional form rather than in terms of set theory. For instance, marking $M_{15} = \{p_2, p_9, p_{14}\}$ in propositional form is written as: $M_{15}: p_2 \wedge p_9 \wedge p_{14}$.

Hence, by observing Figure 4.9 the following can be deduced:

1. $<M_0, \alpha> \models \Box[M_{15} \Rightarrow O(t_3(ok) \wedge t_{10}(ok))]$

Since transitions t_3 and t_{10} can fire independently of each other, and only one transition can fire at any instant, then applying Proposition 1 to each transition in turn yields:

2. $<M_0, \alpha> \models \Box[M_{15} \Rightarrow \Diamond t_{10}] \vee \Box[M_{15} \Rightarrow \Diamond t_3]$

Assuming that t_{10} fires, then from Figure 4.9 the next reachable marking is M_{19}, thus:

3. $<M_0, \alpha> \models \Box[t_{10} \Rightarrow O M_{19}]$

Under M_{19} either t_{11} or t_3 can fire independently of each other, thus applying Proposition 1 to these transitions yields:

4. $<M_0, \alpha> \models \Box[M_{19} \Rightarrow \Diamond t_{11}] \vee \Box[M_{19} \Rightarrow \Diamond t_3]$

Assuming that t_{11} fires then the next reachable marking becomes M_{22}:

5. $<M_0, \alpha> \models \Box[t_{11} \Rightarrow O M_{22}]$

The transitions enabled under M_{22} can fire independently and are formalised as:

6. $<M_0, \alpha> \models \Box[M_{22} \Rightarrow \Diamond t_{12}] \vee \Box[M_{22} \Rightarrow \Diamond t_3]$

If t_{12} fires then from Figure 4.9:

7. $<M_0, \alpha> \models \Box[t_{12} \Rightarrow O M_{25}]$

The firable transitions under M_{25} are t_3 and t_{13}. If t_3 fires then t_{13} remains firable and is not disabled, however, if t_{13} fires then t_3 is also disabled. This means that the firing of t_3 can be represented by Proposition 1, but the firing of t_{13} must be represented by Proposition 2. Hence using a form of Proposition 2 to represent the firing of transitions firable in M_{25}, it can be concluded that:

8. $<M_0, \alpha> \models \Box[M_{25} \Rightarrow \Diamond(t_3 \vee t_{13})]$

Combining 2. - 8. by using the temporal logic proof system it can be concluded that:

9. $<M_0, \alpha> \models \Box[M_{15} \Rightarrow \Diamond(t_3 \vee t_{13})] \vee \Box[(M_{15} \vee M_{19} \vee M_{22}) \Rightarrow \Diamond t_3]$

However, from 8., if t_{13} fires then by Proposition 1 it can be concluded that:

10. $<M_0, \alpha> \models \Box[t_{13} \Rightarrow \Diamond t_2]$

Combining 9. and 10. it can be concluded that:

11. $<M_0, \alpha> \models \Box[M_{15} \Rightarrow \Diamond(t_2 \vee t_3)] \vee \Box[(M_{15} \vee M_{19} \vee M_{22}) \Rightarrow \Diamond t_3]$

Using the sets of places for M_{15}, M_{19} and M_{22} then 11. can be further reduced to:

12. $<M_0, \alpha> \models \Box[p_2 \Rightarrow \Diamond(t_2 \vee t_3)] \vee \Box[p_2 \Rightarrow \Diamond t_3]$

Hence, 12. can be reduced to:

13. $<M_0, \alpha> \models \Box[p_2 \Rightarrow \Diamond(t_2 \vee t_3)]$

This completes the proof of Lemma 2 ■

Further liveness properties of Figure 4.6(a) are:

Lemma 3

Whenever the drum rotates it will eventually stop:

$$<M_0, \alpha> \models \Box[t_{10} \Rightarrow \Diamond t_{13}] \qquad 4.24$$

Lemma 4

Whenever the drum comes to rest the slider makes an insertion:

$$<M_0, \alpha> \models \Box[t_{13} \Rightarrow \Diamond(t_2 \vee t_8)] \qquad 4.25$$

Lemma 5

Whenever the slider aborts its motion then eventually it will insert and withdraw from the drum:

$$<M_0, \alpha> \models \Box[t_3 \Rightarrow \Diamond t_6] \qquad 4.26$$

Proofs of Lemmas 3 - 5 are shown in Appendix C.

This section demonstrates that temporal Petri nets can be used to prove liveness properties of the Petri net model, which specify what the system must do. However, they do not prove safety properties such as the slider will never insert into the drum, while the drum is rotating.

4.4 Extended temporal Petri net analysis

The extended temporal Petri net analysis (detailed in Section 3.4.1) is based on transferring the Petri net model into temporal logic via a translation algorithm. This translation yields a system dependent axiom and a set of system dependent inference rules, which are used within the framework of temporal logic to prove various properties. The analysis of the Petri net model of Figure 4.6(a) will be considered using extended temporal Petri nets. Thus, the translation algorithm for this Petri net yields:

(i) System dependent axiom:
the initial marking of the net is $M_0 = \{p_{14}, p_{15}, p_{16}\}$, thus the system dependent axiom becomes:

$$p14 \wedge p15 \wedge p16 \qquad A1$$

A1 should also contain the negation of the places that do not contain a token, however, this is not shown but is implicitly assumed.

(ii) System dependent Inference Rules (IR): these contain the pre and postconditions of each transition in the Petri net. For Figure 4.6(a) this is given by:

IR1. $p16 \wedge \neg p1 \Rightarrow O(p1 \wedge \neg p16)$
IR2. $p1 \wedge \neg p2 \Rightarrow O(p2 \wedge \neg p1)$
IR3. $p2 \wedge p15 \wedge \neg p3 \Rightarrow O(\neg p2 \wedge \neg p15 \wedge p3)$
IR4. $p3 \wedge \neg p4 \Rightarrow O(p4 \wedge \neg p3)$
IR5. $p4 \wedge \neg p5 \Rightarrow O(p5 \wedge \neg p4)$
IR6. $p5 \wedge \neg p9 \wedge \neg p17 \Rightarrow O(\neg p5 \wedge p9 \wedge p17)$
IR7. $p2 \wedge p9 \wedge \neg p6 \Rightarrow O(\neg p2 \wedge p9 \wedge p6)$
IR8. $p6 \wedge \neg p7 \Rightarrow O(\neg p6 \wedge p7)$
IR9. $p7 \wedge p15 \wedge \neg p8 \Rightarrow O(p8 \wedge \neg p7 \wedge \neg p15)$
IR10. $p8 \wedge \neg p4 \Rightarrow O(p4 \wedge \neg p8)$
IR11. $p9 \wedge p14 \wedge \neg p11 \Rightarrow O(\neg p14 \wedge p9 \wedge p11)$
IR12. $p11 \wedge \neg p12 \Rightarrow O(p12 \wedge \neg p11)$
IR13. $p12 \wedge \neg p13 \Rightarrow O(p13 \wedge \neg p12)$
IR14. $p9 \wedge p13 \wedge (\neg p14 \wedge \neg p15) \Rightarrow O(\neg p13 \wedge \neg p9 \wedge p14 \wedge p15)$
IR15. $p17 \wedge \neg p16 \Rightarrow O(p16 \wedge \neg p17)$

The above system dependent axiom and inference rules are used in conjunction with a proof procedure that is referred to as a refutation proof procedure in [He 90]); this attempts to prove the negation of the desired property. Both safety and liveness properties can be verified using this procedure and the following examples illustrate its use for proving selected properties of the arbor drum and transfer slider system.

4.4.1 Proving safety properties

Lemma 1

A situation will *never occur* when the slider has taken a decision to insert and the drum is rotating:

$$\Box \neg [p4 \wedge p9] \qquad 4.27$$

Lemma 2

A situation will *never occur* when the slider is at the decision point and no decision is made:

$$\Box \neg[p2 \wedge \neg(p9 \vee p15)] \qquad 4.28$$

The refutation proof procedure is illustrated for proving Lemma 1, proof of Lemma 2 is shown in Appendix C:

Lemma 1

$$\Box \neg[p4 \wedge p9] \qquad 4.27$$

Proof

Assuming that Lemma 1 is not true, then

1. $\neg \Box \neg[p4 \wedge p9]$

From the definition of $\Diamond$ and $\Box$, 1. can be written as:

2. $\Diamond [p4 \wedge p9]$

The following proof procedure uses the proof system of extended temporal Petri nets to derive the premise of 2. Therefore, by successively applying the system dependent inference rules IR1. - IR6. to A1, it is seen that when $p4$ is tokenised then $p9$ is not tokenised and when $p9$ is tokenised then $p4$ is not tokenised. Hence, the following can be deduced:

3. $\neg[p4 \wedge p9]$

A pure temporal axiom (where w is any well formed propositional formula) obtained from the temporal logic proof system [Manna 83b] asserts that:

4. $w \wedge \Diamond \neg w \Rightarrow \Diamond(w \wedge \bigcirc \neg w)$

The conjunction of 2. and 3. yields:

5. $\neg[p4 \wedge p9] \wedge \Diamond [p4 \wedge p9]$

By applying Modus Ponens to 4. (where $\neg[p4 \wedge p9]$ is substituted for w) and 5. yields:

6. $\Diamond(\neg[p4 \wedge p9] \wedge \bigcirc[p4 \wedge p9])$

Expressing the first conjunct of 6., by De Morgans theorem:

7. $\Diamond([\neg p4 \vee \neg p9] \wedge \bigcirc[p4 \wedge p9])$

Expanding 7. yields:

8. $\Diamond([\neg p4 \wedge \bigcirc[p4 \wedge p9] \vee [\neg p9 \wedge \bigcirc[p4 \wedge p9])$

Examining each disjunct of 8. in turn

9.	$[\neg p4 \wedge \bigcirc[p4 \wedge p9]]$	First disjunct of 8.
10.	$\neg p4 \wedge \bigcirc p4$	by weakening 9.
11.	$\bigcirc p9$	by weakening 9.

The inference rules that will yield 10. are IR10 and IR4, ie

$$p8 \wedge \neg p4 \;\Rightarrow\; O(p4 \wedge \neg p8)$$
$$p3 \wedge \neg p4 \;\Rightarrow\; O(p4 \wedge \neg p3)$$

By applying the appropriate inference rules, which include IR10 and IR4, to A1 it can be shown that the conclusion of 11. cannot be obtained. Similar reasoning as 9. - 11. can be applied to the second disjunct of 8. which also yields a contradiction. The contradiction of both disjuncts of 8. proves that 1., the negation of the desired property, is false for the system given by A1 and the inference rules IR. Thus Lemma 1 is proved. ■

4.4.2 Proving liveness properties

An interesting liveness property of the slider in Figure 4.6(a) can be stated as:

Lemma 3

Every time the slider starts moving towards the drum then it will eventually insert into the drum. This can be formalised as:

$$\Box\Diamond(p14 \wedge p15 \wedge p16) \;\Rightarrow\; \Box\Diamond p4 \qquad 4.29$$

This property can be simply stated as:

$$\Box\Diamond p4 \qquad 4.30$$

Proof

Using the refutation proof procedure Lemma 3 is negated, which yields:

$$\Diamond\Box\neg p4 \qquad 4.31$$

The following demonstrates the proof of the validity of this temporal proposition:

The system dependent axiom A1 yields

1. $p14 \wedge p15 \wedge p16$

Successive application of IR1, IR2, IR3, IR4 to 1. yields the following (where $O^4 = OOOO$ and represents the next temporal operator applied for four successive occasions):

2. $O^4(p14 \wedge p4 \wedge \neg p3)$

By weakening 2. it can be concluded that:

3. $O^4(p4)$ ie. the slider will insert into the drum

Further application of IR5 and IR6 to 2. yields:

4. $O^6(p14 \wedge p9 \wedge p17 \wedge \neg p5)$

For the marking shown in 4., both IR11 and IR15 are applicable; moreover since each rule can be applied independently, then each rule can be applied independently in either order to yield:

5. $O^8((p16 \wedge \neg p17) \wedge (p9 \wedge p11 \wedge \neg p14))$

From 5. by applying IR1, (IR1,IR12), (IR1,IR12,IR13), (IR1,IR12,IR13,IR14), or (IR12,IR13,IR14). In order to determine the various reachable markings from 5. each combination of IR's will be applied, which produces the following respective expressions:

6. $O^9((p1 \wedge \neg p16) \wedge (p9 \wedge p11)) \vee O^{10}((p1 \wedge \neg p16) \wedge (p9 \wedge p12)) \vee$
$O^{11}((p1 \wedge \neg p16) \wedge (p9 \wedge p13)) \vee O^{12}((p1 \wedge \neg p16) \wedge (p15 \wedge p14)) \vee$
$O^{11}((p16 \wedge \neg p17) \wedge (p15 \wedge p14))$

Considering each disjunct of 6. in turn and applying the appropriate inference rules, it is possible to trace each disjunct to its eventual marking. Thus the first disjunct of 6. is:

7. $O^9((p1 \wedge \neg p16) \wedge (p9 \wedge p11))$

Applying IR2, IR7 and IR8 successively to 7. yields:.

8. $O^{12}((p7 \wedge \neg p6) \wedge (p9 \wedge p11))$

Applying IR12,IR13 and IR14 to 8. yields:

9. $O^{15}((p7 \wedge \neg p6) \wedge (p14 \wedge p15))$

from 9. by applying IR9 and IR10.

10. $O^{17}((\neg p8 \wedge p4 \wedge p14))$

by weakening 10.

11. $O^{17}(p4)$

Using the temporal theorem $Ow \Rightarrow \Diamond w$ a weakened form of 11. can be written as $\Diamond p4$.

Similarly, result $O^{17}(p4)$ is obtained by the second and third disjunct and $O^{15}(p4)$ from the fourth and fifth disjunct of 6., so that in each case the result is $p4$.

Since the motion of the drum and slider is repetitive, then from 1.-11. it can be concluded that, for any marking sequence generated from the initial marking M_0, place p4 will eventually be tokenised. This contradicts Equ. 4.31 (which is interpreted as, eventually a situation will occur in which place p4 will never have a token) and thus proves Lemma 3 to be true. ■

4.5 Conclusions

The aim of this chapter was to evaluate the usefulness of several Petri net based techniques for modelling and analysing a hard real-time control problem. Particular interest was attached to the type of properties that could be analysed by each technique. The research into the application of these techniques allows the following conclusions to be drawn.

Petri net analysis is concerned with generating a reachability graph which is used to:

(i) detect the presence of static hazardous states,
(ii) detect the presence of potentially hazardous sequences (which is important for analysing time-critical systems),
(iii) demonstrate the existence of liveness properties.

However, the main limitations of Petri net analysis are:

(i) they cannot be used to determine real-time performance of systems, because Petri nets do not contain the notion of time.
(ii) they cannot be used to specify and verify safety and liveness properties.

Timed Petri net analysis, which is also based on generating a reachability graph, can be used to:

(i) identify the presence of hazardous markings by examining the timing of time-critical sequences,
(ii) impose timing constraints on the Petri net which has the effect of:
    (a) eliminating hazardous time-critical sequences,
    (b) eliminating parts of the reachability graph which corresponds to certain types of behaviours.

The usefulness of Petri net and timed Petri net analysis in the contexts of the hard real-time control problem modelled in this chapter is as follows. Petri net analysis showed that the Petri net of Figure 4.5(a) contained:

(i) a potentially hazardous time-critical sequence,
(ii) a number of states from which the necessary abort decision could not be reached directly.

Timed Petri net analysis was used to provide constraints which removed the hazardous sequences. However, the abort problem remains unless a further timing constraint is applied.
The modified Petri net model of Figure 4.6(a) also included a potentially hazardous sequence; again timing constraints could be applied. However, unlike the Petri net of Figure 4.5(a), Figure 4.6(a) contained no states from which the necessary abort could not be reached directly. In order to ensure that the slider could take the abort decision when necessary, a further timing constraint was applied; this timing constraint imposed a less stringent requirement than for Figure 4.5(a).

Razouk and Phelps's timed Petri net was applied to the Petri net of Figure 4.6(a) and it was noted that the model of time used could allow the structure of the Petri net to be simplified, shown in Figure 4.7(a). Although Figure 4.7(a) was structurally a simplified version of Figure 4.6(a), the former did not remove the hazardous sequence which existed in the latter. In fact examination of each Petri net's reachability graph showed that they exhibit equivalent behaviours. The main attribute of the timed Petri nets of Razouk and Phelps is that they allowed a more sophisticated and detailed timing analysis to be performed than the timing analysis performed on Figure 4.6(a). However, the cost of such analysis was an increase in the complexity of generating the reachability graph; again complexity of this form limits the size of the net which can be analysed.

Although both timed Petri nets allowed the causal properties of the system to be modelled, and its temporal behaviour to be analysed, they suffered from the following deficiencies:

(i) no support for the formal specification and verification of properties of the model,
(ii) timed Petri net analysis is limited to the size of Petri net that can be examined, because the generation of reachability graphs for relatively large nets can easily become complex and unmanageable. This problem is particularly mainfested in relatively small timed Petri nets of Razouk and Phelps because extra 'intermediate' markings (see Section 4.2.3) must be generated in the reachability graph.

Temporal Petri net analysis, centred on an event-based logic, was used to make assertions (incorporated in Propositions 1 and 2) concerning the firing of Petri net transitions. The main attributes of temporal Petri nets and their analysis, compared to Petri nets and timed Petri nets are:

(i) they allow proof of liveness properties. The absence of such proofs, in Petri nets and timed Petri nets, is a major drawback in their application to the formal specification and verification of systems [Koymans 85]. This is perhaps the main reason why Petri nets are confined to the area of performance evaluation of systems,

(ii) properties such as eventuality and fairness could be specified, this could not be expressed using the notation of Petri nets or timed Petri nets.

Although temporal Petri nets do not replace timed Petri nets [Suzuki 89], they are complementary to the latter and extend the application of Petri net based techniques to the area of formal specification and verification of systems.

A possible criticism of temporal Petri net analysis is that the proof of liveness properties (achieved via the temporal logic component of temporal Petri nets) could be simply discerned by inspecting the reachability graph of the Petri net, thus implying that these properties are known to be 'correct' before they are proved. Although this argument is valid and provides a good heuristic, the main purpose of formal analysis (as seen in other formal methods such as Z and VDM) is:

(i) to confirm that the model contains properties which are known to be intuitively correct, or those that the model should possess. Hence providing a greater confidence in the model,

(ii) the logic component of formal analysis allows the detection of any inconsistent arguments that may be used to assert a certain property.

In summary, logic provides an abstract tool for analysing a system rather than resorting to a reachability state analysis.

Temporal Petri nets are not suited to proving safety properties, instead they must be inferred via an exhaustive search of the reachability graph. This aspect of temporal Petri net analysis limits the size of Petri net that can be considered because the total reachability graph may have to be enumerated.

Extended temporal Petri nets are centred on a state-based logic. This is manifested in the translation algorithm by the formulation of the initial marking as the system dependent axiom, and the transition firing rules as the system dependent inference rules.
The proof procedures allow verification of safety and liveness properties, which implicitly generates a partial reachability graph of the Petri net. In comparison to temporal Petri nets,

extended temporal Petri nets have the desirable feature of being capable of proving safety properties. However, the proof procedure for liveness properties needs to be improved. For instance, such proofs are centred on generating a marking sequence using the system dependent axiom and inference rules; this corresponds to generating a partial reachability graph. However, if the reachability graph is large then the proof can become tedious and unmanageable. Thus a more abstract proof procedure is required similar to that employed in proving safety properties. Furthermore, it was noted that whilst both state and event-based logics allow the proof of liveness properties, the event-based logic can handle these properties more naturally. This is because liveness properties represent the dynamic behaviour of the system, which can be easily thought of in terms of the occurrence of sequences of events rather than states.

Finally, application of the techniques examined in this chapter has shown that they suffer from the following limitations:

(i) they cannot formally specify and verify real-time properties which involve the use of quantitative time; although timed Petri nets can analyse such real-time properties, they are restricted to the performance evaluation of systems.

(ii) each technique uses the reachability graph in its analysis procedure, however, the construction of a reachability graph becomes tedious as the size of the net increases.

Hence, if the above techniques are to be suitable for the formal specification and verification of hard real-time systems, the above deficiencies must be addressed and this will be done in the following chapters of this thesis.

Chapter 5

Real-time Temporal Petri nets

5.1 Introduction: time in Petri nets

Previous chapters have examined two categories of Petri nets that include the notion of time. Timed Petri nets introduce time by allocating quantitative time values to net transitions. These nets can be used to evaluate the real-time performance of the modelled system by deriving timing relationships, using the quantitative time values, that bound the behaviour of the net. Techniques based on Petri nets and temporal logic, such as temporal and extended temporal Petri nets, that will be collectively referred to as 'temporal Petri nets', are suitable for the formal specification and verification of systems. These nets provide a notation for specifying properties such as eventuality and fairness and allow the proof of safety and liveness properties. The type of properties that can be verified using 'temporal Petri nets' involve the use of qualitative time which do not specify a time bound on the occurrence of an event or state.

Since timed Petri nets and 'temporal Petri nets' cannot specify and verify properties involving quantitative time, they are unsuitable for formally modelling hard real-time systems. This is because the formal modelling of these systems require the specification and verification of real-time properties, which use quantitative time to describe the occurrence of an event or a state.

The aim of this chapter is to address the above deficiency by proposing an extension to 'temporal Petri nets', which allows the use of quantitative time in temporal logic. This requires the following changes to be made to the constituents of 'temporal Petri nets':

(i) Petri nets must have the capacity for specifying quantitative time constraints,
(ii) the notation of temporal logic must be able to specify and verify properties using quantitative time constraints: this type of logic was referred to as real-time temporal logic (RTTL) in Section 2.4.3.

Timed Petri nets can accommodate a quantitative notion of time. Specifically, the timed Petri nets of Razouk and Phelps are perhaps the most suited to modelling hard real-time systems because they use a more accurate model of time as demonstrated in Section 4.2.3.

Various types of RTTL exist, as surveyed in Section 2.4.3, but not all of them are suitable for use in the proposed extension. For example, the RTTL of Bernstein and Harter [Bernstein 87] is limited to proving safety properties, the RTL of Jahanian and Mok [Jahanian 86] is based on a different computational model to Petri nets and this logic is not fully decidable. Fortunately, the RTTL of Ostroff [Ostroff 89], developed for the combined formalisms of ESM/RTTL (see Section 2.4.3) appears to be suitable for use with Petri nets because both formalisms have compatible computational models. This link between Petri nets and RTTL was made implicitly by Ostroff [Ostroff 89] who suggested that the formulas of his RTTL are applicable to any model that can be described by the computational model of the FTS (Fair Transition System). Since, Pnueli [Pnueli 86] had previously shown that Petri nets can be represented by the FTS, it can be inferred that a direct link between Petri nets and RTTL of [Ostroff 89] can be formed. Furthermore, due to the similarities between the computational models of Petri nets and timed Petri nets a link between timed Petri nets and RTTL also exists. Therefore, the combination of timed Petri nets and RTTL proposed in this chapter will be closely guided by the way in which ESMs were linked to RTTL in [Ostroff 89]. The technique resulting from the combination of timed Petri nets and RTTL will be referred to as real-time temporal Petri nets. Moreover, this technique will be applied to the hard real-time control problem of Section 4.1.1 in order to establish its usefulness.

5.2 Definition of real-time temporal Petri nets

The real-time temporal Petri net (RTTN) is defined as the pair:

$$\text{RTTN} = \{\text{TC},\text{Tf}\} \qquad 5.1$$

where

TC is the timed Petri net developed by Razouk and Phelps [Razouk 85],

Tf represents the formulas that characterise the real-time temporal behaviour of TC.

The following sections are devoted to defining components TC and Tf.

5.2.1 Formal description of the timed Petri net (TC)

Following the definition of timed Petri nets presented in [Razouk 85], the structure of TC is described as the tuple:

$$TC = \{P, T, I, O, M_0, \delta e, \delta f\} \qquad 5.2$$

where P, T, I, O and M_0 were defined in Equ. 2.1 and are re-stated as follows:

(i) P is a set of finite places: $P = \{p_1, p_2, \ldots, p_n\}$ and $n > 0$,

(ii) T is a set of finite transitions: $T = \{t_1, t_2, \ldots, t_m\}$ and $m > 0$,

(iii) I is the input function that maps the transition to places defined as:

$$I : T \rightarrow P^{\infty} \qquad 5.3$$

where P^{∞} represents the set of all strings made up of elements of P.

A place $p_i \in P$ is an input place of $t_j \in T$, iff $p_i \in I(t_j)$; the notation $\#(p_i, I(t_j))$ represents the number of occurrences of p_i in $I(t_j)$,

(iv) O is the output function that maps the transition to places defined as:

$$O : T \rightarrow P^{\infty} \qquad 5.4$$

A place $p_i \in P$ is an output place of $t_j \in T$, iff $p_i \in O(t_j)$; $\#(p_i, O(t_j))$ represents the number of occurrences of p_i in $O(t_j)$,

(v) M_0 represents the initial marking that characterises the initial state of the net. Any marking M reachable from M_0 is formally defined as:

$$M: P \rightarrow \{0, 1, 2, \ldots, m\}, \qquad 5.5$$

(vi) A transition $t_j \in T$ becomes enabled (or firable) in a marked Petri net C (defined above) under marking M iff the following condition is satisfied:

$$\forall p_i \in P \bullet M(p_i) \geq \#(p_i, I(t_j)). \qquad 5.6$$

In TC an enabled transition must remain continuously enabled (in accordance with Equ. 5.6) for the time interval given by the enabling time (δe) before it can commence firing; δe is defined by the function:

$$\delta e: T \rightarrow \mathbf{N} \qquad 5.7$$

where **N** is the set of natural numbers,

(vii) An enabled transition $t_j \in T$ in TC completes firing by the time interval denoted by its firing time (δf), which is defined by the function:

$$\delta f: T \rightarrow \mathbf{N} \qquad 5.8$$

The definition of TC in [Razouk 85] uses the notion of a global clock, which is assumed to 'tick' infinitely often, to describe the time related behaviour of TC. In this discussion, the time indicated by this clock will be denoted by the time variable (τ). This time may be measured relative to any appropriate instant in the execution of the net, such as the instant at which a transition becomes enabled, or when a transition fires. The following discussion uses the instant at which a transition becomes enabled as a reference point, in order to observe the effect of the enabling and firing times on the behaviour of the net.

5.2.2 Formal description of the temporal formulas (Tf)

The proposed formulas Tf, which characterise the real-time behaviour of TC, are constructed from RTTL and concern the enabling and firing of transitions in TC. In order to develop these formulas it is necessary to express features of TC in terms of formulas of propositional logic. These formulas are then used with temporal logic to make assertions concerning the sequence of enabling and firing of a transition in TC; such assertions are made using qualitative and quantitative time. The formulas for the marking and the sequence of enabling and firing of a transition in TC, defined using set theory in Section 5.2.1, are expressed in terms of propositions in the following section.

5.2.2.1 Marking of TC

The net marking M, defined as a function in Equ. 5.5, can be represented as a formula of propositional logic, which will be denoted by *M*. This formula comprises the conjunction of all tokenised places that form M. Each conjunct of *M* is a place that is constructed from the atomic proposition:

$$p \equiv \text{place } \mathrm{p} \in \mathrm{P} \text{ and } \mathrm{p} \in \mathrm{M}. \qquad 5.9$$

Similarly, propositional formulas for the input and output functions $I(\mathrm{tj})$ and $O(\mathrm{tj})$ can be written as $I(tj)$ and $O(tj)$ respectively. Thus in essence M and *M*, $I(\mathrm{tj})$ and $I(tj)$, $O(\mathrm{tj})$ and $O(tj)$ are notationally different representations of the same entities.

5.2.2.2 Enabling and firing of transitions in TC

The derivation of propositional formulas for the enabling and firing of a transition in TC can be achieved by using definitions 5.2.1(vi) and 5.2.1(vii), and formalising the sequence of enabling and firing of a transition. The sequence of enabling and firing of a transition in TC can be described as follows.

Consider a TC in which marking M_j is reachable from M_0, and $t_j \in T$ becomes firable (denoted by t_j(ok)) under M_j with enabling conditions defined by $\mathcal{I}(t_j)$. If t_j becomes firable at the instant when the global clock indicates time $\tau = \mathcal{T}_1$ then, either t_j remains continuously enabled for its enabling time δe_j or t_j becomes disabled by the firing of a conflicting transition also enabled at M_j. If t_j remains continuously enabled for δe_j, then it will commence firing (the firing of t_j is denoted by t_j(f)) at time $\mathcal{T}_2 = (\mathcal{T}_1 + \delta e_j)$. The firing of t_j causes the instantaneous absorption of tokens from its input places ($\mathcal{I}(t_j)$), producing a new net marking M_j' defined as:

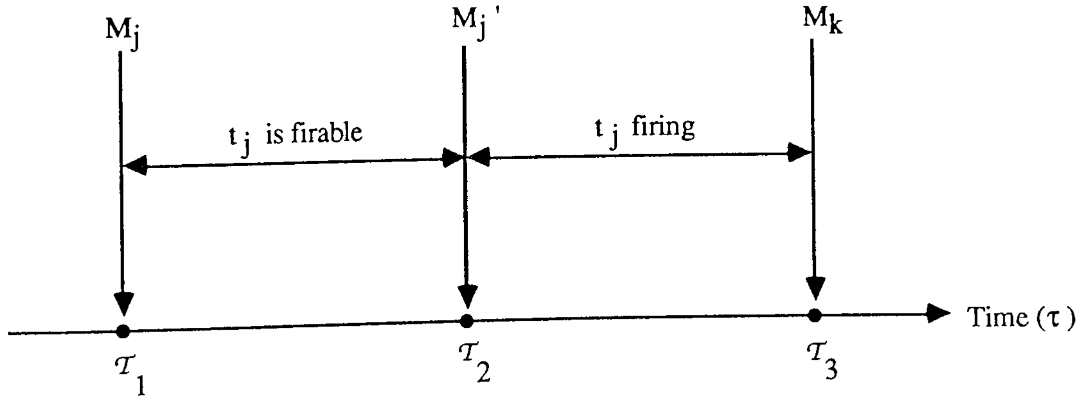
$$M_j' = M_j - \mathcal{I}(t_j) \qquad 5.10$$

These absorbed tokens do not engage in the enabling or firing of any other transitions in TC. The firing of t_j continues for its firing time δf_j and completes firing (denoted by t_j(c)) at time $\mathcal{T}_3 = (\mathcal{T}_2 + \delta f_j)$. At this instant the absorbed tokens are instantaneously deposited into the appropriate output places, defined by $\mathcal{O}(t_j)$, producing a new net marking M_k. The relationship between markings M_j and M_k can be defined as:

$$M_k(p_i) = M_j(p_i) - \#(p_i, \mathcal{I}(t_j)) + \#(p_i, \mathcal{O}(t_j)) \qquad 5.11$$

The timing relationships for the enabling and firing of transition t_j are summarised in Figure 5.1.

Figure 5.1 Enabling and firing of t_j in TC



where

$\mathcal{T}_1$ represents the instant at which $\tau = \mathcal{T}_1$ and $t_j(ok)$ begins to hold, under net marking M_j,

$\mathcal{T}_2$ represents the instant at which $\mathcal{T}_2 = (\mathcal{T}_1 + \delta e_j)$ and t_j commences firing (where t_j firing is denoted by $t_j(f)$) and the net marking changes instantaneously from M_j to M_j',

$\mathcal{T}_3$ represents the instant at which $\mathcal{T}_3 = (\mathcal{T}_1 + \delta e_j + \delta f_j)$ and t_j completes firing (denoted by $t_j(c)$) and the net marking changes instantaneously from M_j' to M_k.

Using the above sequence of enabling and firing of t_j the following temporal formulas, which involve qualitative time, can be written.

(i) The enabling of t_j

(a) the enabling of t_j, defined by Equ. 5.6, can be formalised as:

$$[I(t_j) \Rightarrow t_j(ok)] \wedge [t_j(ok) \Rightarrow I(t_j)] \qquad 5.12$$

Equ 5.12 can be simplified using the laws of propositional logic to:

$$I(t_j) \equiv t_j(ok), \qquad 5.13$$

(b) when t_j becomes firable then it can either commence firing or become disabled by the firing of a conflicting transition also enabled under M_j. This can be formalised as:

$$M_j \wedge t_j(ok) \Rightarrow O(t_j(f) \vee t_j(\neg ok)) \qquad 5.14$$

Equ. 5.14 is interpreted as, if t_j becomes firable at marking M_j then, the next event (which represents the next significant change in the net) occurs when either t_j commences firing or it becomes disabled by the firing of a conflicting transition.

(ii) The firing of t_j

(a) if t_j commences firing then, by Equ. 5.10, changes in the net marking can be formalised as:

$$t_j(f) \Rightarrow M_j' \qquad 5.15$$

and when t_j is in the process of firing then the next event to occur in the net is when t_j completes firing. This is formalised as:

$$t_j(f) \Rightarrow O t_j(c), \qquad 5.16$$

(b) when t_j actually completes firing the net marking becomes M_k which is formalised as:

$$t_j(c) \Rightarrow M_k \qquad 5.17$$

Combining Equs. 5.14 - 5.17 using temporal and propositional reasoning the enabling and eventual firing of t_j can be deduced as:

$$M_j \wedge tj(ok) \Rightarrow O((M_j{}' \wedge OM_k) \vee tj(\neg ok)) \qquad 5.18$$

If tj actually fires then using temporal reasoning, the first disjunct in the consequence of Equ. 5.18 can be simplified to:

$$M_j \wedge tj(ok) \Rightarrow OM_j{}' \wedge \Diamond M_k \qquad 5.19$$

5.2.2.3 Simplifying assumption

The 'intermediate' markings of TC, such as $M_j{}'$, produce a large number of reachable markings compared to the markings reachable from a similar Petri net C. This yields a reachability graph that can become unacceptably large, even for relatively small nets, a feature that was also noticed in Section 4.2.3. However, an examination of the role of these 'intermediate' markings shows that they are used to:

(i) prevent other enabled transitions in M_j from firing once tj is firing,
(ii) show that the firing of transitions in TC consumes time.

In order to reduce the number of reachable markings and the size of the reachability graph, these 'intermediate' markings may be eliminated if it is assumed that, when an enabled transition (tj) starts firing the tokens from its input places remain in their respective places. Thus, these tokens are not absorbed and they do not engage in the firing of any other transitions, until tj completes firing. Under this assumption the formulas for the enabling and firing of tj become:

(i) enabling of tj

$$I(tj) \equiv tj(ok) \qquad 5.20$$

$$M_j \wedge tj(ok) \Rightarrow O(tj(c) \vee tj(\neg ok)) \qquad 5.21$$

The interpretation of Equ. 5.21 is: when tj becomes firable at marking M_j then the next event to occur is either tj will complete firing or become disabled by the firing of a conflicting transition also enabled under M_j. Note if there are no conflicting transitions enabled when tj is enabled under marking M_j then Equ. 5.21 simply reduces to:

$$M_j \wedge tj(ok) \Rightarrow Otj(c) \qquad 5.22$$

The reason for not including tj(f) in Equs. 5.21 and 5.22 is because (under the above assumption) the marking remains the same when tj becomes firable and commences firing, hence when t_j becomes firable the next significant change (which produces a change in marking) in the net occurs when either t_j completes firing or becomes disabled.

(ii) firing of t_j

$$t_j(f) \Rightarrow M_j \qquad 5.23$$

and

$$t_j(c) \Rightarrow M_k \qquad 5.24$$

Combining Equs 5.21 - 5.24 the enabling and firing of transition t_j can be deduced as:

$$M_j \wedge t_j(ok) \Rightarrow O(M_k \vee t_j(\neg ok)) \qquad 5.25$$

Additionally, if the enabled transition t_j fires then this action can be represented by combining Equs. 5.20, 5.22 and 5.24 into the following formula:

$$M_j \wedge t_j(ok) \Rightarrow I(t_j) \wedge OM_k \qquad 5.26$$

A close examination of the formulas developed in this section show that they are similar to those developed by Ostroff in [Ostroff 89] for the combined formalism of ESM/RTTL. Specifically, Equ. 5.26 makes the same assertion as the transition axiom (TA) which was introduced in [Ostroff 89] to formalise the firing of an enabled transition in an ESM. Specifically, TA was constructed using a temporal implication, in which its antecedent was the logical expression composed of the marking and its enabled transition; the consequence contained the successor marking and the enabled transitions enabling conditions. Hence TA has the same composition as Equ. 5.26.

5.2.2.4 Real-time temporal formulas (Tf)

Since the generic formulas of RTTNs are equivalent to the ESM/RTTL framework of [Ostroff 89], it is sufficient for this thesis to derive the real-time temporal formulas for RTTN in the same manner as shown for the latter. Specifically, in [Ostroff 89] these formulas were developed and shown to be sound; this approach will also be used in this section.

Applying the notion of real-time constraints to the enabling and firing rules of TC, Equs. 5.20 - 5.26, the following can be deduced.

(i) Real-time enabling of a transition

For a TC, let $t_j \in T$ become enabled at time $\tau = \mathcal{T}_1$ with enabling conditions $I(t_j)$, under marking M_j which is reachable from M_0. If t_j remains continuously enabled for the enabling time δe_j then t_j will fire (unless it becomes disabled by the firing of a conflicting transition also enabled under M_j). Hence, if:

$$M_j \Rightarrow t_j(ok) \tag{5.27}$$

and $$M_j \wedge t_j(ok) \Rightarrow O(t_j(c) \vee \neg I(t_j)) \tag{5.28}$$

then the following can be deduced:

$$M_j \wedge t_j(ok) \wedge \tau = \mathcal{T}_1 \Rightarrow O(\neg t_j(c)\ \mathcal{U}\ (\tau \geq \mathcal{T}_1 + \delta e_j)) \tag{5.29}$$

Soundness of the real-time enabling of transition t_j.

For a timed Petri net TC, let any marking sequence (σ) be represented by:

$$\sigma = M_0, M_1, \ldots, M_{j-1}, M_j, \ldots$$

Let the point at which marking M_j is reached be when the global clock indicates $\tau = \mathcal{T}_1$, this can be represented as:

$$\sigma \models (M_j \wedge \tau = \mathcal{T}_1).$$

If at M_j the net has neither reached a state of deadlock nor terminated execution, then $t_j \in T$ will be enabled. Hence this establishes the truth of Equ. 5.27:

$$\sigma \models M_j \Rightarrow t_j(ok)$$

The enabling rules of transitions in a TC - which is represented in 5.2.1(vi) and by Equ. 5.21 - establishes the truth of Equ. 5.28, hence:

$$\sigma \models M_j \wedge t_j(ok) \Rightarrow O(t_j(c) \vee \neg I(t_j))$$

From 5.2.1(vi) t_j must remain continuously enabled from the instant of being enabled for a duration of δe_j, before it can commence firing and subsequently fire. Therefore, this can be represented by:

$$\sigma \models O(\neg t_j(c)\ \mathcal{U}\ (\tau \geq \mathcal{T}_1 + \delta e_j))$$

From the above arguments it can be seen that Equ. 5.29 is valid for any enabled transition in TC.

(ii) Real-time firing of a transition

If $t_j \in T$ has remained continuously enabled, from the instant of being enabled at time $\tau = \mathcal{T}_1$, for δe_j with enabling conditions $I(t_j)$ under marking M_j, then firing of t_j can commence on completion of δe_j. Transition t_j continues firing for the firing time δf_j and at the instant at which δf_j expires, t_j completes firing by instantaneously changing the net marking from M_j to M_{j+1}. Hence, if:

$$M_j \wedge t_j(ok) \Rightarrow O(t_j(c) \vee \neg I(t_j)), \qquad 5.30$$

$$t_j(f) \Rightarrow M_j \qquad 5.31$$

and $$t_j(c) \Rightarrow M_{j+1} \qquad 5.32$$

then it can be deduced that:

$$M_j \wedge t_j(ok) \wedge \tau = \mathcal{T}_1 \Rightarrow M_j \,\mathcal{U}\, (M_{j+1} \wedge (\tau = \mathcal{T}_1 + \delta e_j + \delta f_j) \vee \neg I(t_j)) \qquad 5.33$$

Soundness of real-time firing of transitions in TC

Any marking sequence (σ) in TC can be represented by:

$$\sigma = M_0, M_1, \ldots, M_{j-1}, M_j, M_{j+1}, \ldots$$

Let t_j become enabled at the point when marking M_j is reached; this can be represented by Equ. 5.28:

$$\sigma \models M_j \wedge t_j(ok) \Rightarrow O(t_j(c) \vee \neg I(t_j)).$$

When enabled transition t_j reaches the firing stage then the net marking must remain at M_j (thus preventing the firing of any other transitions whilst t_j is firing). Hence, Equ. 5.31 must be valid:

$$\sigma \models t_j(f) \Rightarrow M_j$$

Once t_j completes firing then the net marking changes to M_{j+1}. Hence, Equ. 5.32 is valid:

$$\sigma \models t_j(c) \Rightarrow M_{j+1}$$

However by 5.2.1(vii) the new marking M_{j+1} cannot be realised until the firing of t_j has completed, in which case δf_j must have expired. Hence, the following must hold true:

$$\sigma \models M_j \,\mathcal{U}\, (M_{j+1} \wedge (\tau = \mathcal{T}_1 + \delta e_j + \delta f_j) \vee \neg I(t_j))$$

This section has developed real-time temporal formulas for RTTN and shown them to be sound. It is interesting to note that the formulas represented by Equs. 5.29 and 5.33 are similar to the rule of delay and the rule of real-time response respectively, which were developed for ESMs in [Ostroff 89]. The equivalent relationship between the formalisms of RTTN and ESM/RTTL is convenient because it can allow further formulas to be derived that are similar in each technique. Moreover, this relationship allows the proof system and proof procedures derived in [Ostroff 89] to be used in RTTN analysis. The usefulness of RTTNs will be explored by applying them to the hard real-time control problem detailed in Section 4.1.1.

5.3 Application of RTTNs

The Petri net model of Figure 4.7(a) for the hard real-time control problem will be analysed using RTTN. More specifically, RTTN will be used to verify both real-time and non real-time properties.

5.3.1 Proof of non real-time properties

The non real-time properties that can be proved using the RTTN framework will be presented; these properties specify the motion of the mechanisms.

Lemma 1

Once the drum completes its incremental rotation then the slider must have inserted into the stationary drum prior to the next completion of the drum's rotation. This property is formalised, using Figure 4.7(a), as:

$$t_{13}(\mathrm{ok}) \Rightarrow \mathrm{O}[(t_2(\mathrm{ok}) \vee t_8(\mathrm{ok}))\ \mathbf{P}\ t_{13}(\mathrm{ok})] \qquad 5.34$$

Proof

The proof of Lemma 1 uses the temporal theorem for "precedes (**P**)", which has been proved to be valid in the LTLs developed in [Manna 83b], and has the form:

$$(w1 \Rightarrow \neg w3) \wedge (w1 \wedge \neg w2 \Rightarrow \mathrm{O}w1) \Rightarrow (w1 \Rightarrow w2\mathbf{P}w3) \qquad 5.35$$

where

$w1$, $w2$ and $w3$ are any propositional formulas.

Equ. 5.35 is interpreted as: at a particular instant, if w_1 is true then w_3 is false and if w_1 is true but w_2 is false then at the next instant w_1 is true, then it can be deduced that when w_1 is true then w_2 occurs before w_3.
The following proof is based on deriving the premises, which form the antecedent of Equ. 5.35.

Using Equ. 5.25 it can be deduced from the reachability graph of Figure 4.7(b) that when t_{13} becomes firable then the next net marking could be either M_0, M_1, M_2, M_9 or M_{17}:

1. $t_{13}(ok) \Rightarrow O(M_0 \vee M_1 \vee M_2 \vee M_9 \vee M_{17})$

Substituting, M_0: $p_{14} \wedge p_{15} \wedge p_{16}$, M_1: $p_1 \wedge p_{14} \wedge p_{15}$, M_2: $p_2 \wedge p_{14} \wedge p_{15}$, M_9: $p_{14} \wedge p_{15} \wedge p_{17}$ and M_{17}: $p_7 \wedge p_{14} \wedge p_{15}$., into 1. and simplifying the resulting expression, the following can be deduced:

2. $t_{13}(ok) \Rightarrow O p_{15}$

From the Figure 4.7(a) it can be inferred that p_{15} and p_9 cannot be tokenised at the same instant. Hence, it can be deduced that:

3. $p_{15} \Rightarrow \neg p_9$

Using Equ. 5.12 to describe the enabling condition for t_{13} and reducing the resulting expression:

4. $t_{13}(ok) \Rightarrow (p_{12} \wedge p_9)$

Simplifying 4. and applying the laws of propositional logic, the following can be deduced:

5. $\neg p_9 \Rightarrow \neg t_{13}(ok)$

Combining 3. and 5., the following can be deduced:

6. $p_{15} \Rightarrow \neg t_{13}(ok)$

From the reachability graph of Figure 4.7(b) it can be checked that the following premise is valid:

7. $(p_{15} \wedge \neg(t_2(ok) \vee t_8(ok))) \Rightarrow O p_{15}$

Forming the conjunction of 6. and 7., which becomes the antecedent of Equ. 5.35. the following can be deduced:

8. $p_{15} \Rightarrow (t_2(ok) \vee t_8(ok)) \mathbf{P}\, t_{13}(ok)$

From 2. and 8. it follows by propositional logic that:

9. $t_{13}(ok) \Rightarrow O[(t_2(ok) \vee t_8(ok)) \mathbf{P}\, t_{13}(ok)]$

This completes the proof of Lemma 1 ■

Lemma 2

Once the slider has taken a decision to insert into the drum, which is stationary, then the drum must have completed its rotation prior to the next insertion of the slider.

$$(t_2(\text{ok}) \vee t_8(\text{ok})) \Rightarrow \Diamond(t_{13}\ \mathbf{P}\ (t_2(\text{ok}) \vee t_8(\text{ok}))) \qquad 5.36$$

Proof

Proof of Lemma 2 is based on the same structure as the proof of Lemma 1. From the reachability graph, Figure 4.7(b), it can be inferred that when t_2 becomes firable, there is no conflicting enabled transition, and the next reachable net marking is M_3. Thus using a simplified form of Equ. 5.25 it can be deduced that:

1. $t_2(\text{ok}) \Rightarrow \text{O}M_3$

Similarly, using Equ. 5.25 and Figure 4.7(b) to describe the enabling of transitions in markings immediately reachable from M_3, the following formulas can be deduced:

2. $M_3 \wedge t_4(\text{ok}) \Rightarrow \text{O}M_4$
3. $M_4 \wedge t_5(\text{ok}) \Rightarrow \text{O}M_5$
4. $M_5 \wedge t_6(\text{ok}) \Rightarrow \text{O}M_6$

From 1. - 4. it can be deduced that:

5. $t_2(\text{ok}) \Rightarrow \Diamond M_6$

Substituting $M_6{:}\ p9 \wedge p14 \wedge p17$ into 5. and simplifying the resulting expression yields:

6. $t_2(\text{ok}) \Rightarrow \Diamond p9$

However, if t_8 becomes firable, then:

7. $t_8(\text{ok}) \Rightarrow \text{O}M_4$

At marking M_4 the enabled transitions and subsequent reachable markings are described by 3., and 4.; thus combining 7., 3. and 4. it can be deduced that:

8. $t_8(\text{ok}) \Rightarrow \Diamond M_6$

Substituting $M_6{:}\ p9 \wedge p14 \wedge p17$ into 8. and simplifying the resulting expression yields:

9. $t_8(\text{ok}) \Rightarrow \Diamond p9$

For the slider to insert into the drum either t_2 or t_8 must become firable; therefore combining 6. and 9. yields:

10. $[(t_2(\text{ok}) \vee t_8(\text{ok})) \Rightarrow \Diamond p9]$

From the structure of the Petri net of Figure 4.7(b) it can be inferred that:

11. $p9 \Rightarrow \neg p15$

From Equ. 5.20, the enabling condition for t_2 is:

12. $t_2(\text{ok}) \Rightarrow (p2 \wedge p15)$

Simplifying 12. yields:

13. $t_2(\mathrm{ok}) \Rightarrow p15$

Similarly the enabling condition for t_8 can be simplified to:

14. $t_8(\mathrm{ok}) \Rightarrow p15$

Since either t_2 or t_8 can become enabled then 13. and 14. yields:

15. $(t_2(\mathrm{ok}) \vee t_8(\mathrm{ok})) \Rightarrow p15$

Using laws of propositional logic, 15. can be represented as:

16. $\neg p15 \Rightarrow \neg(t_2(\mathrm{ok}) \vee t_8(\mathrm{ok}))$

From 11. and 16. it can be deduced that:

17. $p9 \Rightarrow \neg(t_2(\mathrm{ok}) \vee t_8(\mathrm{ok}))$

From the reachability graph of Figure 4.7(b) it can be checked that the following is valid:

18. $(p9 \wedge \neg t13) \Rightarrow \mathrm{O}p9$

From the conjunction of 17. and 18. which form the antecedent of Equ. 5.34, it can be deduced that:

19. $p9 \Rightarrow t13\ \mathbf{P}\ (t_2(\mathrm{ok}) \vee t_8(\mathrm{ok}))$

From 10. and 19 it can be deduced that:

20. $(t_2(\mathrm{ok}) \vee t_8(\mathrm{ok})) \Rightarrow \Diamond(t13\ \mathbf{P}\ (t_2(\mathrm{ok}) \vee t_8(\mathrm{ok})))$

This completes the proof of Lemma 2 ■

5.3.2 Proof of real-time properties

There are several safety critical real-time properties of the Petri net model, Figure 4.7(a) and its reachability graph, Figure 4.7(b), which must be proven. These are:

Lemma 3

When the slider starts its motion then the drum must become stationary within a time constraint for the slider to insert into the drum, without making an abort motion.

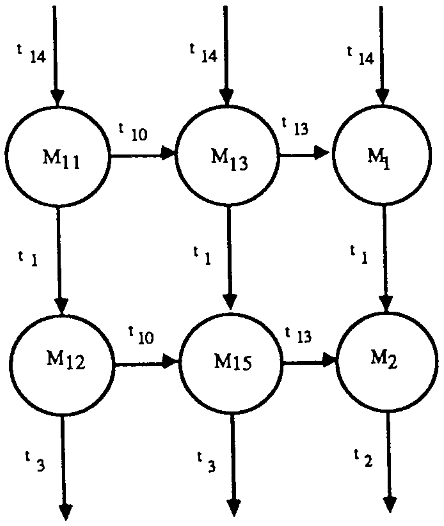
Assuming that the slider starts accelerating towards the drum at time $\tau = 0$, then the following can be written:

$$p1 \wedge (\tau = 0) \Rightarrow \Diamond((p14 \wedge p2) \wedge (\tau > 0)) \qquad 5.37$$

Proof

The proof of properties involving the temporal operator $\Diamond$, referred to as real-time eventualities, are tackled as follows. The relevant portion of the reachability graph is selected by identifying the marking containing the goal places (in Lemma 3 the goal places are $(p14 \wedge p2)$). Then the reachability graph is used to determine all marking sequences that lead to the goal place from the marking containing the origin place (in Lemma 3 the origin place is $p1$). For Lemma 3 the relevant part of the reachability graph is shown in Figure 5.2, in which $M2$ contains the goal place and $M11$ contains the last occurrence of the origin place.

Figure 5.2 Partial reachability graph of Figure 4.7(b)



MARKING	PLACES
M_1	P_1, P_{14}, P_{15}
M_2	P_2, P_{14}, P_{15}
M_{11}	P_1, P_9, P_{14}
M_{12}	P_2, P_9, P_{14}
M_{13}	P_1, P_9, P_{12}
M_{15}	P_2, P_9, P_{12}

In Figure 5.2, the paths indicated by t_3 are not considered because they lead to the abort motion of the slider. In this case, the slider will only insert into the drum when it is stationary and as a consequence no timing constraints need to be satisfied by the slider.

From Figure 5.2 all marking sequences that lead to the goal places are considered using Equ. 5.33; for instance consider marking sequence generated by the firing of t_{10}, t_{13} and t_1:

1. $M11 \wedge t_{10}(\text{ok}) \wedge \tau = \mathcal{T}1 \Rightarrow M11\ \mathcal{U}\,((M13 \wedge (\tau = \mathcal{T}1 + \delta_{10})) \vee \neg I(t10))$
   where $\delta_{10} = \delta e_{10} + \delta f_{10}$
2. $M13 \wedge t_{13}(\text{ok}) \wedge \tau = \mathcal{T}2 \Rightarrow M13\ \mathcal{U}\,((M1 \wedge (\tau = \mathcal{T}2 + \delta_{13})) \vee \neg I(t13))$
   where $\delta_{13} = \delta e_{13} + \delta f_{13}$ and $\mathcal{T}2 = \mathcal{T}1 + \delta_{10}$
3. $M1 \wedge t_1(\text{ok}) \wedge \tau = \mathcal{T}3 \Rightarrow M1\ \mathcal{U}\,((M2 \wedge (\tau = \mathcal{T}3 + \delta_1)) \vee \neg I(t1))$
   where $\delta_1 = \delta e_1 + \delta f_1$ and $\mathcal{T}3 = \mathcal{T}2 + \delta_{13}$

If transitions t_{10}, t_{13} and t_1 actually fire then combining 1., 2., and 3. it can be deduced that:

4. $M11 \wedge t_{10}(\text{ok}) \wedge \tau = \mathcal{T}1 \Rightarrow (M11 \vee M13 \vee M1)\ \ \mathcal{U}\,(M2 \wedge (\tau = \mathcal{T}1 + \delta_{x1}))$
   where $\delta_{x1} = \delta_1 + \delta_{10} + \delta_{13}$

Propositions $M11: p1 \wedge p9 \wedge p14$ and $M2: p2 \wedge p14 \wedge p15$ can be substituted into 4. and the resulting expression simplified to yield:

5. $p1 \wedge t_{10}(\text{ok}) \wedge \tau = \mathcal{T}_1 \Rightarrow (M_{11} \vee M_{13} \vee M_1))\ \mathcal{U}\,((p2 \wedge p14) \wedge (\tau = \mathcal{T}_1 + \delta_{x1}))$

Using similar reasoning as 1. - 5. the firing of sequences (t_{10}, t_1, t_{13}) and (t_1, t_{10}, t_{13}) yields 6. and 7. respectively:

6. $p1 \wedge t_{10}(\text{ok}) \wedge \tau = \mathcal{T}_1 \Rightarrow (M_{11} \vee M_{13} \vee M_{15}))\ \mathcal{U}\,((p2 \wedge p14) \wedge (\tau = \mathcal{T}_1 + \delta_{x1}))$

7. $p1 \wedge t_1(\text{ok}) \wedge \tau = \mathcal{T}_1 \Rightarrow (M_{11} \vee M_{12} \vee M_{15}))\ \mathcal{U}\,((p2 \wedge p14) \wedge (\tau = \mathcal{T}_1 + \delta_{x1}))$

Combining 5., 6. and 7. yields:

8. $p1 \wedge (t_1(\text{ok}) \wedge t_{10}(\text{ok})) \wedge \tau = \mathcal{T}_1 \Rightarrow (M_{11} \vee M_{12} \vee M_{13} \vee M_{15} \vee M_1))$
$\mathcal{U}\,((p2 \wedge p14) \wedge (\tau = \mathcal{T}_1 + \delta_{x1}))$

For any marking sequence represented by 5. or 6.or 7. and summarised in 8. t_2 is not enabled and is not ready for firing until ($\tau = \mathcal{T}_1 + \delta_{x1}$). Therefore 8. can be simplified to:

9. $p1 \wedge (t_1(\text{ok}) \wedge t_{10}(\text{ok})) \wedge \tau = \mathcal{T}_1 \Rightarrow \neg t_2(\text{f})\,\mathcal{U}\,((p2 \wedge p14) \wedge (\tau = \mathcal{T}_1 + \delta_{x1}))$

The above premise defines the general time relationships for the decision process. Considering each combination of sequences from ($p1 \wedge \tau = 0$) to M_2, then from 9. and Figure 4.7(b) it is apparent that:

10. $p1 \wedge \tau = 0 \wedge (p14 \wedge \neg p15) \Rightarrow \Diamond((p2 \wedge p14) \wedge (\tau = \delta_{x1}))$

Similarly, using Figure 4.7(b). and a simplified form of a combination of 2. and 3. it can be reasoned that:

11. $p1 \wedge \tau = 0 \wedge (\neg p14 \wedge \neg p15) \Rightarrow \Diamond((p14 \wedge p2) \wedge (\tau = \delta_{x2}))$
where $\delta_{x2} = \delta_{13} + \delta_1$

12. $p1 \wedge \tau = 0 \wedge (p14 \wedge p15) \Rightarrow \Diamond((p14 \wedge p2) \wedge (\tau = \delta_1))$
where $\delta_1 = \delta_{e1} + \delta_{f1}$

However, if the drum is already stationary, when the slider starts its motion (as represented by 12.) then the slider will insert into the drum without resorting to the abort motion and without need to consider timing constraints. Therefore, premise 12. does not need to be considered in the following proof. Hence, combining 10. and 11. yields:

13. $p1 \wedge \tau = 0 \Rightarrow \Diamond((p14 \wedge p2) \wedge (\delta_{x2} \le \tau \le \delta_{x1}))$ ■

Step 13. concludes the proof of Lemma 3 and asserts that: if the slider starts accelerating towards the drum (which is either rotate enabled or rotating) at time $\tau = 0$ then the drum must become stationary whilst the slider is at the decision point within the time constraint defined by δ_{x1} and δ_{x2}, for the slider to insert into the drum without making an abort motion.

Another real-time property, which can be proved in a similar manner as Lemma 3, can be written as:

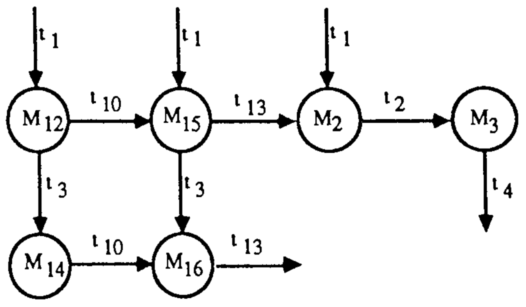
Lemma 4

When the slider enters the decision point then a decision must be taken either to continue inserting or abort its motion within a real-time constraint; this is formalised as:

$$p2 \wedge (\tau = 0) \;\Rightarrow\; \Diamond(p7 \vee (p3 \wedge p14)) \wedge (\tau > 0) \qquad 5.38$$

Proof

Figure 5.3 Partial reachability graph used in the proof of Lemma 4



MARKING	PLACES
M_2	p_2, p_{14}, p_{15}
M_3	p_3, p_{14}
M_{12}	p_2, p_9, p_{14}
M_{14}	p_7, p_9, p_{14}
M_{15}	p_2, p_9, p_{12}
M_{16}	p_7, p_9, p_{12}

Using the partial reachability graph of Figure 5.3, under marking M_{12} either transition t_3 or t_{10} can complete firing. Therefore, using Equ. 5.33 to represent the firing of t_3 and t_{10} the following can be deduced:

1. $M_{12} \wedge \tau = \mathcal{T}_1 \Rightarrow M_{12}\ \mathcal{U}\ (M_{15} \wedge (\tau = \mathcal{T}_1 + \delta_{10}) \vee M_{14} \wedge (\tau = \mathcal{T}_1 + \delta_3))$

   where

   $\delta_{10} = (\delta e_{10} + \delta f_{10})$, $\delta_3 = (\delta e_3 + \delta f_3)$,

   $\mathcal{T}_1$ represents the instant at which marking M_{12} becomes tokenised.

In 1., if t_{10} fires then the net marking becomes M_{15} in which either transition t_3 or t_{13} can complete firing. Hence, using Equ. 5.33 it can be concluded that:

2. $M_{15} \wedge \tau = \mathcal{T}_2 \Rightarrow M_{15}\ \mathcal{U}\ (M_2 \wedge (\tau = \mathcal{T}_2 + \delta_{13}) \vee M_{16} \wedge (\tau = \mathcal{T}_2 + \delta_3))$

   where

$\mathcal{T}_2 = (\mathcal{T}_1 + \delta_{10})$, $\delta_{13} = (\delta e_{13} + \delta f_{13})$,

$\mathcal{T}_2$ represents the instant at which marking M_{15} is reached.

From 2., if t_{13} fires then the net marking will become M_2 and it can be concluded that:

3. $M_2 \wedge \tau = \mathcal{T}_3 \Rightarrow M_2 \; \mathcal{U} \; (M_3 \wedge (\tau = \mathcal{T}_3 + \delta_2))$

where

$\mathcal{T}_3 = (\mathcal{T}_2 + \delta_{13})$, $\delta_2 = (\delta e_2 + \delta f_2)$,

$\mathcal{T}_3$ represents the instant at which marking M_2 becomes tokenised.

Considering, the three sets of transition firing sequences which can occur from M_{12} the following can be concluded for all firing sequences generated from M_{12}:

4. $M_{12} \wedge \tau = \mathcal{T}_1 \Rightarrow (M_{12} \vee M_{15} \vee M_2) \; \mathcal{U} \; (M_3 \wedge (\tau = \mathcal{T}_1 + \delta_{x3})) \vee$

$M_{12} \wedge \tau = \mathcal{T}_1 \Rightarrow M_{12} \; \mathcal{U} \; (M_{14} \wedge (\tau = \mathcal{T}_1 + \delta_3)) \vee$

$M_{12} \wedge \tau = \mathcal{T}_1 \Rightarrow (M_{12} \vee M_{15}) \; \mathcal{U} \; (M_{16} \wedge (\tau = \mathcal{T}_1 + \delta_{x4}))$

where

$\delta_{x3} = (\delta_{10} + \delta_{13} + \delta_2)$ and $\delta_{x4} = (\delta_{10} + \delta_3)$

Substituting the propositional formulas for the markings:

M_2: $p2 \wedge p14 \wedge p15$, M_3: $p3 \wedge p14$, M_{12}: $p2 \wedge p9 \wedge p14$, M_{14}: $p7 \wedge p9 \wedge p14$, M_{15}: $p2 \wedge p9 \wedge p12$ and M_{16}: $p7 \wedge p9 \wedge p12$ into 4. and simplifying the resulting expression, it can be deduced that:

5. $p2 \wedge \tau = \mathcal{T}_1 \Rightarrow \Diamond((p3 \wedge p14) \wedge (\tau = \mathcal{T}_1 + \delta_{x3})) \vee$

$p2 \wedge \tau = \mathcal{T}_1 \Rightarrow \Diamond p7 \wedge (\tau = \mathcal{T}_1 + \delta_3) \vee$

$p2 \wedge \tau = \mathcal{T}_1 \Rightarrow \Diamond p7 \wedge (\tau = \mathcal{T}_1 + \delta_{x4})$

Simplification of 5. yields:

6. $p2 \wedge \tau = \mathcal{T}_1 \Rightarrow \Diamond((p3 \wedge p14) \wedge (\tau = \mathcal{T}_1 + \delta_{x3})) \vee$

$p2 \wedge \tau = \mathcal{T}_1 \Rightarrow \Diamond p7 \wedge ((\mathcal{T}_1 + \delta_3) \leq \tau \leq (\mathcal{T}_1 + \delta_{x4}))$

Further simplification of 6. yields:

7. $p2 \wedge \tau = \mathcal{T}_1 \Rightarrow \Diamond[((p3 \wedge p14) \wedge (\tau = \mathcal{T}_1 + \delta_{x3})) \vee$

$(p7 \wedge ((\mathcal{T}_1 + \delta_3) \leq \tau \leq (\mathcal{T}_1 + \delta_{x4})))]$

Premise 7. defines the general time relationships when the slider enters the decision point. Considering that the slider enters the decision point at $\tau = 0$, then from 7. and Figure 4.7(b) the following can be asserted:

8. $p2 \wedge \tau = 0 \Rightarrow \Diamond[((p3 \wedge p14) \wedge (\tau = \delta_{x3})) \vee$

$(p7 \wedge (\delta_3 \leq \tau \leq \delta_{x4}))]$ ■

Step 8. concludes the proof of Lemma 4 which asserts that when the slider enters the decision point then the slider will decide either to insert into the drum at time constraint defined by δ_{x1} or to abort motion by the time constraint defined by $(\delta_3 \leq \tau \leq \delta_{x2})$.

5.4 Summary and conclusions

This chapter has a proposed a technique referred to as real-time temporal Petri nets (RTTNs), and illustrated their use in the formal analysis of the timing properties of a hard real-time control problem.

The formal framework of RTTN was based on formalising the enabling and firing rules of transitions in a timed Petri net (TC). This formalisation was performed using qualitative time temporal logic and quantitative time temporal logic. Hence, RTTNs could be used to prove non-real-time properties, involving qualitative time, and real-time properties, that use quantitative time. The type of non real-time properties that could be proved were based on using the precedence constraint temporal operator; they were used to specify and verify the motion of the drum and slider. Although the proof of these properties used similar assertions as the liveness properties proved using temporal Petri nets (see Section 4.3), they are classed as safety properties [Pnueli 85], [Ostroff 89].

The real-time properties verified, using RTTNs, in this chapter essentially involved deriving real-time constraints needed to achieve a desired behaviour; the verification procedure relied on identifying and generating the relevant partial reachability graph. However, RTTNs are not restricted to performing these type of proofs, instead the existing proof procedures can be easily used to verify whether or not a pre-defined time constraint is satisfiable. It is interesting to observe that the timing constraints, derived using RTTN, could be obtained using relational expressions of timed Petri net analysis (see Section 4.2); however, the main attributes of RTTN are:

(i) it allows a means of formally reasoning about the real-time aspects of the system modelled using Petri nets,
(ii) the application of timed Petri nets via RTTN is no longer restricted to the performance evaluation of systems, but also allows for the formal specification, verification and design of hard real-time systems.

A comparison of the formal framework of RTTNs and temporal Petri net shows that there some similarities in their verification procedures. For instance the proofs performed in both techniques are based making assertions about the firing of transitions. In particular, RTTN analysis is based on formalising the sequence of enabling and firing rules of transitions, and temporal Petri net analysis is based on formalising the firing of transitions (via Propositions 1 and 2). This suggests that the formalism of one technique could be used to derive the basic propositions of the other. For example, an examination of Equ. 5.22 of RTTN shows that it is essentially a form of Proposition 1 used in temporal Petri net. However, in terms of the type of properties that can be proved, RTTNs allows proof of properties which could be classed as safety, liveness and real-time properties, whilst temporal Petri nets can only be used to verify liveness properties, which are based on qualitative time.

Chapter 6

Analysing Petri net based models using concurrency sets

6.1 Introduction

The techniques presented in previous chapters have shown that the reachability graph forms an integral part of the analysis procedure. For instance, the following techniques use information from the reachability graph.

(i) Timed Petri nets:
   - (a) the semantics of the reachable markings are used to determine the occurrence of hazardous situations in the Petri net model,
   - (b) performance expressions are developed from the reachability graph to avoid these situations and ensure the safety of the model.

(ii) Temporal Petri nets:
   - (a) safety properties are inferred from the reachability graph,
   - (b) the validity of propositions used to prove liveness properties are checked using the reachable markings,
   - (c) proof of liveness properties require knowledge of reachable markings.

(iii) Extended temporal Petri nets:
   knowledge of reachable markings are central to the proof of safety and liveness properties.

(iv) Real-time temporal Petri nets:
   the identification of particular marking sequences form the basis of proving real-time and non real-time properties.

Although the above techniques allow the analysis of useful properties of hard real-time systems, their main limitation stems from the need to generate the reachability graph which can easily become large and unmanageable, even for small Petri nets. Since the reachability graph provides valuable state information for analysing Petri net models, this chapter addresses the issue of reducing the complexity of generating the reachability graph. This is achieved by providing a direct means of generating a partial reachability graph pertaining to a particular system state.

The merit of this approach can be illustrated by examining its usefulness in the techniques (i) - (iv) summarised above. For instance:

(i) timed Petri net analysis requires a partial reachability graph for identifying certain system states that can occur under hazardous situations. Timing analysis is used to prevent the occurrence of these situations; this may require the total reachability graph,

(ii) temporal Petri net analysis identifies a partial reachability graph that is relevant to the proof of a particular liveness property. Specifically, it is used to check the validity of propositions of temporal Petri nets as well as in the proof procedures,

(iii) extended temporal Petri net analysis uses a partial reachability graph to assist in the verification of safety and liveness properties; this is especially useful in the latter part of safety proofs, see proof of Lemma 1 in Section 4.4.1,

(iv) real-time temporal Petri net analysis needs to identify a partial reachability graph that is relevant to the proof of the real-time property under consideration, as shown by the proof of Lemma 3 in Section 5.3.2.

In this chapter the technique used to generate a partial reachability graph is the concurrency sets developed by Skeen and Stonebraker [Skeen 83]. These sets were applied to distributed database systems in [Skeen 83] to verify the 'correctness' of commit protocols and to determine the consistency of the database in the presence of failures. Although this chapter primarily uses concurrency sets to generate partial reachability graphs, these sets will also be used to check the consistency of reachable states in the Petri net models considered.

6.2 Concurrency sets

Concurrency sets were introduced by Skeen and Stonebraker [Skeen 83] as part of a formal model for assessing the resiliency of protocols in a distributed database system composed of multiple processes or sites. This work considered commit protocols and assessed their behaviour under site failures and network failures. Specifically, concurrency sets were used to 'verify' whether commit protocols could independently recover from these failures. Essentially these sets were used to classify the type of failures the protocol could recover from, or alternatively to provide information necessary to make it recoverable. The main criteria used in the analysis of these protocols, via concurrency sets, was to detect the presence of inconsistent states in the database. These states represented a global situation in which some processes have taken a decision to commit on a transaction whilst others have taken a decision to abort; obviously such situations produce an inconsistent database and are clearly to be avoided.

Skeen and Stonebraker [Skeen 83] modelled commit protocols as finite state machines and used the global state reachability graph to derive the concurrency sets. Specifically, concurrency sets allowed the state (referred to as the local state) of all processes to be determined for a particular processor state, in a system comprising of a network of distributed concurrent processes.

Formally, these sets were defined in [Skeen 83] as:

Definition 6.1:
Let *s* be an arbitrary local state.
The concurrency set of *s* is the set of all local states that are potentially concurrent with it .
This set is denoted by C(*s*).

Recent work by Hill and Holding [Hill 90] has extended the application of concurrency sets to protocols modelled using Petri nets. They use concurrency sets for the design of robust protocols in a distributed system, which uses synchronous message passing to communicate between sites. This chapter uses the work of Hill and Holding [Hill 90] by also applying concurrency sets to Petri nets. However, the proposed use of concurrency sets differs from Hill and Holding [Hill 90] because:

(i) they are applied to Petri net models of hard real-time systems, in which neither the modelling of protocols nor the behaviour of the system under failures is considered,

(ii) they are primarily used to generate reachability graph information.

6.3 Analysis using concurrency sets

For a Petri net the concurrency set is obtained via Definition 6.1, which allows the identification of all states (or places) that are potentially concurrent with a chosen state (or place). This set has the potential for detecting the presence of hazardous markings because it allows the direct generation and inspection of all combination of places which are potentially concurrent with the chosen place.

The generation of concurrency sets and its associated analysis will be illustrated for the Petri net of Figure 4.6(a), which is re-produced in this chapter. For example, the concurrency set for place p_2 is obtained by 'fixing' a token at p_2 and observing all possible sets of places that may be concurrently tokenised. Thus, when p_2 is tokenised, the net marking can either be (p2, p14, p15), (p2, p9, p13), (p2, p9, p12), (p2, p9, p11) or (p2, p9, p14). From these markings the concurrency set for p2 can be easily inferred as:

$$C(p2) = \{(p14, p15), (p9, p13), (p9, p12), (p9, p11), (p9, p14)\} \qquad 6.1$$

It must be emphasized that for a particular concurrency set, the set of all possible net markings (M_T) that it represents is easily derived. For instance, C(p2) has the set of all possible markings M_T:

$$M_T = \{(p2, p14, p15), (p2, p9, p13), (p2, p9, p12), (p2, p9, p11), (p2, p9, p14)\} \qquad 6.2$$

In the above manner a concurrency set for each place in Figure 4.6(a) can be generated. This is shown in Table 6.1.

The interpretation and usefulness of the concurrency sets of Table 6.1 will be illustrated as follows. Consider the concurrency set of Equ 6.1. This is interpreted as, when the slider is at the decision point (p2) the possible states in which the remaining processes of the net can exist in are:

(i) the drum is stationary (p14) and the interlock (p15) indicates that the drum is stationary,

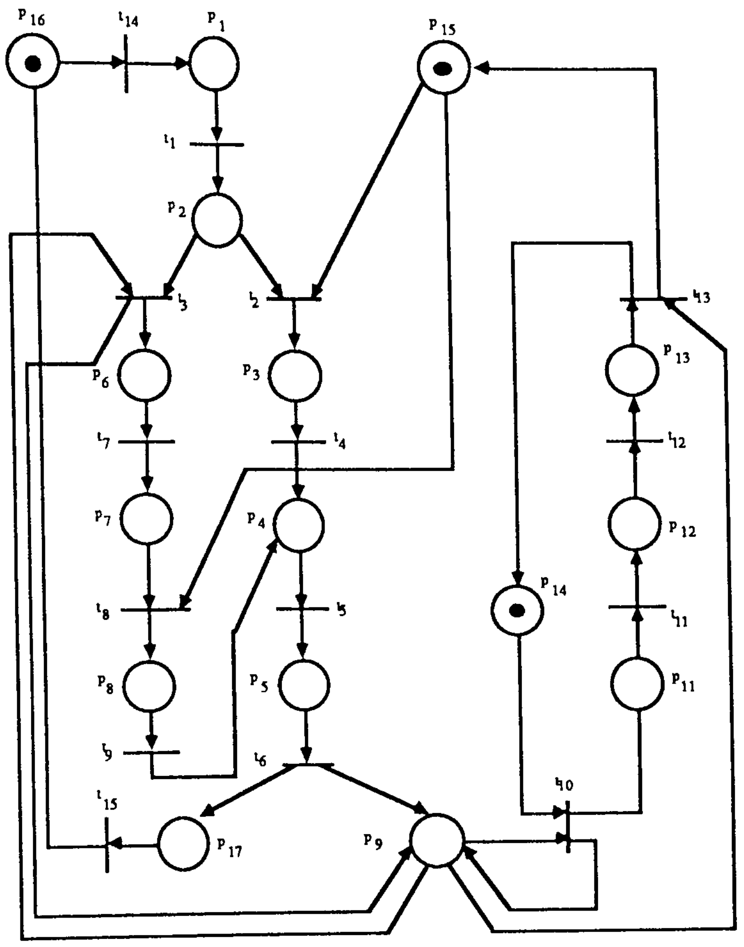
(ii) the drum is decelerating (p13) and the interlock (p9) indicates that the drum is rotating. It must be emphasized that place p9 has dual semantics (see Section 4.2.2) which can indicate that the drum is either rotating or it is rotate enabled; the choice of which semantics apply is dependent on the position of the drum,

(iii) the drum is rotating at constant velocity (p12) and the interlock (p9) indicates that the drum is rotating,

(iv) the drum is accelerating to achieve constant velocity (p11) and the interlock (p9) indicates that the drum is rotating,

(v) the drum is stationary (p14) and the interlock (p9), in this case, indicates that the drum is rotate enabled.

Figure 4.6(a) Modified version of the Petri net model of Figure 4.5(a)



Places	Semantics
p_1	Slider accelerating towards drum
p_2	Slider at decision point
p_3	Slider inserting drum
p_4	Slider fully inserted
p_5	Slider withdrawing
p_6	Slider decelerating
p_7	Slider at rest
p_8	Slider accelerating
p_9	Drum rotating status
p_{11}	Drum accelerating
p_{12}	Drum rotating at constant velocity
p_{13}	Drum decelerating
p_{14}	Drum stationary
p_{15}	Drum stationary status
p_{16}	Slider at rest
p_{17}	Slider's motion to rest

Transition	Semantics
t_1	Slider enters decision point
t_2	Slider proceeds with insertion
t_3	Slider starts to abort motion
t_4	Slider makes an insertion into drum
t_5	Slider starts to withdraw from insertion
t_6	Slider clears drum
t_7	Slider comes to rest
t_8	Slider starts its motion
t_9	Slider makes an insertion into drum
t_{10}	Drum starts to rotate
t_{11}	Drum reaches constant velocity
t_{12}	Drum starts to decelerate
t_{13}	Drum stops rotating
t_{14}	Slider commences its motion
t_{15}	Slider reaches zero velocity

Table 6.1 The set of all concurrency sets for Figure 4.6(a)

Concurrency set	Sets of Markings					
$C(p_1)$	(p_{14}, p_{15})	(p_9, p_{13})	(p_9, p_{12})	(p_9, p_{11})	(p_9, p_{14})	
$C(p_2)$	(p_{14}, p_{15})	(p_9, p_{13})	(p_9, p_{12})	(p_9, p_{11})	(p_9, p_{14})	
$C(p_3)$	p_{14}					
$C(p_4)$	p_{14}					
$C(p_5)$	p_{14}					
$C(p_6)$	(p_{14}, p_{15})	(p_9, p_{13})	(p_9, p_{11})	(p_9, p_{12})	(p_9, p_{14})	
$C(p_7)$	(p_{14}, p_{15})	(p_9, p_{13})	(p_9, p_{11})	(p_9, p_{12})	(p_9, p_{14})	
$C(p_8)$	p_{14}					
$C(p_9)$	(p_7, p_{14})	(p_6, p_{14})	(p_2, p_{14})	(p_1, p_{14})	(p_{14}, p_{16})	(p_{14}, p_{17})
	(p_7, p_{11})	(p_6, p_{11})	(p_2, p_{11})	(p_1, p_{11})	(p_{11}, p_{16})	(p_{11}, p_{17})
	(p_7, p_{12})	(p_6, p_{12})	(p_2, p_{12})	(p_1, p_{12})	(p_{12}, p_{16})	(p_{12}, p_{17})
	(p_7, p_{13})	(p_6, p_{13})	(p_2, p_{13})	(p_1, p_{13})	(p_{13}, p_{16})	(p_{13}, p_{17})
$C(p_{11})$	(p_7, p_9)	(p_6, p_9)	(p_2, p_9)	(p_1, p_9)	(p_9, p_{16})	(p_9, p_{17})
$C(p_{12})$	(p_7, p_9)	(p_6, p_9)	(p_2, p_9)	(p_1, p_9)	(p_9, p_{16})	(p_9, p_{17})
$C(p_{13})$	(p_7, p_9)	(p_6, p_9)	(p_2, p_9)	(p_1, p_9)	(p_9, p_{16})	(p_9, p_{17})
$C(p_{14})$	(p_7, p_9)	(p_6, p_9)	(p_2, p_9)	(p_1, p_9)	(p_9, p_{16})	(p_9, p_{17})
	(p_7, p_{15})	(p_6, p_{15})	(p_2, p_{15})	(p_1, p_{15})	(p_{15}, p_{16})	(p_{15}, p_{17})
	p_5	p_4	p_8	p_3		
$C(p_{15})$	(p_7, p_{14})	(p_6, p_{14})	(p_2, p_{14})	(p_1, p_{14})	(p_{14}, p_{16})	(p_{14}, p_{17})
$C(p_{16})$	(p_{14}, p_{15})	(p_9, p_{13})	(p_9, p_{12})	(p_9, p_{11})	(p_9, p_{14})	
$C(p_{17})$	(p_{14}, p_{15})	(p_9, p_{13})	(p_9, p_{12})	(p_9, p_{11})	(p_9, p_{14})	

Thus by inspecting the semantics of the combination of places in a concurrency set it is possible to detect:

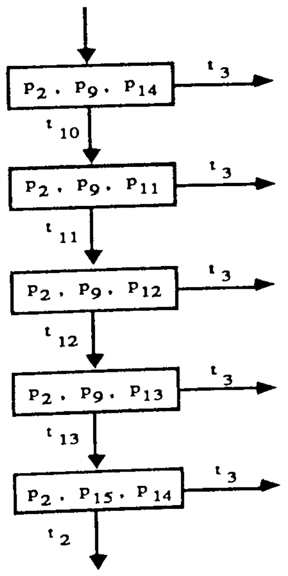
(i) hazardous combinations of states, which show a collision of the mechanisms,

(ii) inconsistent combination of states, which show that certain processes are in a different state to that assumed by other processes. For instance, an inconsistent combination of states would arise if the interlock of p9 was given the semantics of 'the drum is rotating'. Specifically, state combination (p9, p14) would be inconsistent because the p9 would indicate that 'the drum is rotating' but the drum is actually stationary (p_{14}). It must be emphasized that the term inconsistent state used in this chapter differs from that used in Skeen and Stonebraker [Skeen 83], because the latter uses this term to refer to a global state indicating that processes have taken a conflicting decision (commit or abort) on a transaction.

An inspection of the semantics of Equ. 6.1 shows that there are no hazardous combination of states nor inconsistent combination of states. In fact, this result is also obtained for all the remaining concurrency sets of Table 6.1; the result concerning the presence of no hazardous markings is consistent with that obtained in Section 4.2.2.1 for Figure 4.6(a).

Since the concurrency set analysis is based on examining static combination of states and net markings, it cannot detect the presence of hazardous marking sequences. This problem can be remedied by generating a partial reachability graph for places suspected of creating a hazardous situation. For example, consider the slider at the decision point (p_2) which has the possible concurrent markings given by Equ. 6.2. For each concurrent marking all the enabled transitions can be identified and fired using the Petri net of Figure 4.6(a). If this procedure is applied, the partial reachability graph for p_2 can be generated and is shown in Figure 6.1.

Figure 6.1 Partial reachability graph associated with place p2



$p_2 \cdot p_9 \cdot p_{14}$
t_3
t_{10}
$p_2 \cdot p_9 \cdot p_{11}$
t_3
t_{11}
$p_2 \cdot p_9 \cdot p_{12}$
t_3
t_{12}
$p_2 \cdot p_9 \cdot p_{13}$
t_3
t_{13}
$p_2 \cdot p_{15} \cdot p_{14}$
t_3
t_2

Examination of Figure 6.1 shows that a hazardous sequence of markings does result from the C(p2); confirming the results obtained in Section 4.2.2.1. If a partial reachability graph is generated for each concurrency set and the resulting partial reachability graphs are joined together, then this will form the total reachability graph of the net. Hence this approach can provide an alternative structured method of generating the total reachability graph.

This section has shown that concurrency sets have two important uses, which are:

(i) they are very useful for detecting the presence of static hazardous states and inconsistent combination of states for particular parts of the net, that are suspected of containing hazardous situations,

(ii) they can be used in conjunction with the Petri net model to easily generate partial reachability graphs of the net. Moreover, this allows the detection of hazardous sequences of states.

6.3.1 Minimum concurrency set

An inspection of the concurrency sets of Table 6.1 shows that the following observations can be made.

(i) The following places have identical concurrency sets:
- (a) C(p1), C(p2), C(p6), C(p7), C(p16), C(p17);
- (b) C(p3), C(p4), C(p5), C(p8);
- (c) C(p11), C(p12), C(p13).

(ii) The following concurrency sets are related by subset relations:
- (a) C(p3), C(p4), C(p5) C(p8) $\subset$ C(p14)
- (b) C(p15) $\subset$ C(p9);
- (c) C(p11), C(p12), C(p13) $\subset$ C(p14)

From the above observations it transpires that since the concurrency sets of Figure 4.6(a) are related then, if a minimum set of concurrency sets could have been derived, from which the concurrency set for any place in the net could be derived, then there would be no need to generate a concurrency set for every place in the net. This section is devoted to investigated the existence of such a minimum set of concurrency sets for the Petri net of Figure 4.6(a).

Further inspection of Table 6.1 shows that most concurrency sets contain place p_{14} and the remaining sets contain either place p_{11}, p_{12} or p_{13}. This leads to the conclusion that the concurrency sets of Table 6.1 could have been constructed from the concurrency sets of places p_{11}, p_{12}, p_{13} and p_{14}. In order to show this to be the case, Table 6.2 shows the concurrency sets for these places and attributes each component to its parent process. From this table it can be shown that the concurrency set for any place in Figure 4.6(a) can be generated. For example, the concurrency set for place p_{15} can be generated from the first six rows of Table 6.2, which yields the set:

$$C(p15) = \{(p7, p14), (p6, p14), (p2, p14), (p1, p14), (p16, p14), (p17, p14)\} \qquad 6.3$$

Table 6.2 Minimum concurrency set for the Petri net model of Figure 4.6(a)

Drum C(p)	Interlock places	Slider places
	p_{15}	p_{17}
	p_{15}	p_{16}
	p_{15}	p_{1}
	p_{15}	p_{2}
	p_{15}	p_{6}
	p_{15}	p_{7}
		p_{3}
$C(p_{14})$		p_{4}
		p_{5}
		p_{8}
	p_{9}	p_{17}
	p_{9}	p_{16}
	p_{9}	p_{1}
	p_{9}	p_{2}
	p_{9}	p_{6}
	p_{9}	p_{7}
	p_{9}	p_{17}
$C(p_{11}) =$	p_{9}	p_{16}
$C(p_{12}) =$	p_{9}	p_{1}
$C(p_{13})$	p_{9}	p_{2}
	p_{9}	p_{6}
	p_{9}	p_{7}

A comparison of Equ 6.3 and C(p15) of Table 6.1 shows both to be identical. Hence the concurrency sets of Table 6.2 represents the minimum set of concurrency set of Figure

4.6(a) because it can be used to derive the concurrency set of any place in the net. This type of set will be referred to as the minimum concurrency set (C_{min}) and for Figure 4.6(a) it can be represented as:

$$C_{min} = \{C(p_{14}), C(p_{11}), C(p_{12}), C(p_{13})\} \qquad 6.4$$

A comparison of the C_{min} (Table 6.2) and the reachability graph (shown in Figure 4.6(b)) of the Petri net of Figure 4.6(a) reveals some interesting features of the C_{min}. For instance:

(i) all the markings that can be inferred from C_{min} are also contained in the reachability graph, and vice-versa,

(ii) the reachability graph shows information concerning both the reachable markings as well as the marking sequences, however, C_{min} only contains a list of all the reachable markings. Although, the reachability graph possesses more information than C_{min}, the latter represents state information in a manner that allows easy inference of hazardous or inconsistent markings. This information becomes difficult to infer from the reachability graph as its size increases,

(iii) since C_{min} contains all the reachable markings in the net, the transition firing rules of the net can be combined with C_{min} to infer any particular marking sequence in the net. In this manner the C_{min} can be used to infer total reachability graph of the net.

From the above it follows that Skeen and Stonebraker [Skeen 83] used the reachability graph to deduce the concurrency sets, however, the C_{min}, in conjunction with the firing rules, has the potential for the reverse to be accomplished.

This section has shown that a useful subset of the set all concurrency sets exists which is referred to as the minimum concurrency set. This set has the attributes of allowing the concurrency set for any place in the net to be derived and the behaviour of the net to be easily inferred, because it contains all the unique markings in the net. It must be emphasized that the C_{min} of Figure 4.6(a) is not unique to the concurrency sets of places p_{11}, p_{12}, p_{13} and p_{14}. In fact it could have been constructed from the concurrency sets of other places, such as p_3, p_4, p_5, p_8, p_9 and p_{15}. However, the main characteristic feature of C_{min} is that it must contain all the unique markings of the net.

6.4 Reduction of the Petri net model

The preceding section showed that the set of all concurrency sets (Table 6.1) contained many identical or related concurrency sets and as a consequence this set could be reduced to a minimum concurrency set shown in Table 6.2. This section examines the implications which identical or related concurrency sets have on the Petri net of Figure 4.6(a).

In Section 6.3 it was stated that the concurrency sets can be interpreted as showing all possible states that could exist concurrently with a chosen state. Therefore, the interpretation which can be given to places that have identical concurrency sets is that the overall 'state' of the model remains the same for these places. For instance, consider places p11, p12 and p13, which have the semantics of: the drum is accelerating, the drum is rotating at constant velocity and the drum is decelerating, respectively. Their concurrency sets are identical and are given by:

$$C(p11) = C(p12) = C(p13) = \{(p9, p16), (p9, p1), (p9, p2), (p9, p6), (p9, p7), (p9, p17)\} \qquad 6.5$$

Using the above interpretation it is apparent that, if the drum is at p11, p12 or p13 the possible states of other processes (the slider and the synchronization logic) in Figure 4.6(a) remains unchanged. This leads to the conclusion that places p11, p12 and p13 could be represented by a single place, which is given the semantics of the drum is rotating, without changing the functionality represented by Figure 4.6(a). The implications of performing such an abstraction of places are:

(i) it reduces the information contained in the net, but preserves the overall interpretation and behaviour of the net,

(ii) it also reduces the number of reachable markings, thus reducing the size of the reachability graph, which simplifies the subsequent analysis.

The above abstraction could be made by merging places p11 and p13 into p12, which is given the semantics of the drum is rotating.

Applying similar reasoning as used above the following abstraction to Figure 4.6(a) can be made:

(i) since C(p6) = C(p7) then p6 can be merged into p7, which is given the semantics of the drum is at rest.

(ii) $C(p_1) = C(p_2)$ allows p_1 to be merged into p_2; the semantics given to this p_2 is the same as that used in Figure 4.6(a).

(iii) $C(p_{16}) = C(p_{17})$ allows p_{17} to be merged into p_{16}; the semantics given to this p_{16} is the same as in Figure 4.6(a).

(iv) $C(p_3) = C(p_4)$ allows p_3 to be merged into p_4; the semantics of this p_4 is the same as those assigned in Figure 4.6(a).

Note the removal of places p_6, p_{11} and p_{13} was also performed on Figure 4.6(a) in Section 4.2.3 when this net was represented as a timed Petri net of Razouk and Phelps (shown as Figure 4.7(a)). The basis for this reduction was that the model of time, used in this timed Petri net, could be used to subsume the function of places p_6, p_{11} and p_{13}. Therefore, in essence the Petri net of Figure 4.6(a) contained the same information as Figure 4.7(a). However, the above merging of places p_6, p_{11} and p_{13} differs from the reduction performed in Section 4.2.3 because the former produces a Petri net model that is more abstract than Figure 4.6(a).

Further abstractions could have been performed on Figure 4.6(a) by observing that:

(i) $C(p_1) = C(p_2) = C(p_{16}) = C(p_{17})$; this could allow the abstraction of places p_1, p_2, p_{16} and p_{17} into a single place (p_x). If this reduction is made then p_x would receive an input arc from transition t_6 (which represents the event of the slider clearing the drum after insertion) and output arcs to t_2 (slider decides to insert into the drum) and t_3 (slider decides to abort the insert motion). Although suitable semantics could be assigned to p_x, which would represent the motions of the slider withdrawing from the drum, the slider at rest and the slider at decision point, in this case it is desired to preserve the various phases in the motion of the slider.

(ii) $C(p_1) = C(p_2) = C(p_6) = C(p_7) = C(p_{16}) = C(p_{17})$. The abstraction of places p_1, p_2, p_6, p_7, p_{16}, p_{17} into a single place is not performed because according to the structure of the Figure 4.6(a) this abstraction is not possible; usually abstraction of places is only possible if the appropriate places are connected sequentially by arcs and transitions and they are free from decisions such as the structure represented by place p_2 and transitions t_2 and t_3.

Applying the above abstractions, the Petri net model of Figure 4.6(a) is transformed to that of Figure 6.2. The minimum concurrency set of this Petri net is represented by:

$$C_{min} = \{C(p_{12}), C(p_{14})\} \qquad 6.6$$

The markings associated with C_{min} of Equ. 6.6 are shown in Table 6.3.

Figure 6.2 Abstract Petri net model of Figure 4.6(a)

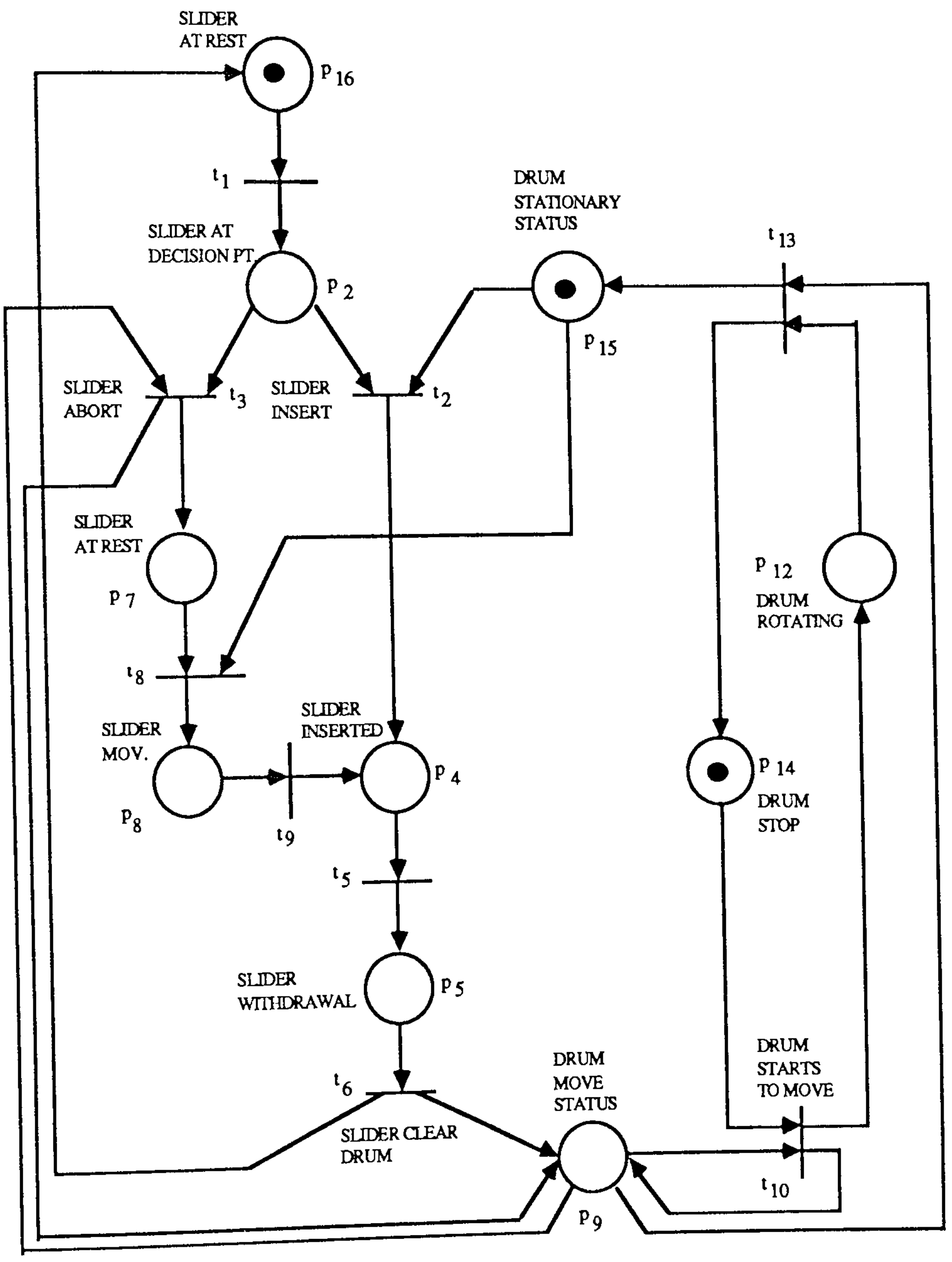


Table 6.3 The minimum concurrency set for the Petri net of Figure 6.2

DRUM	CONTROLLER	SLIDER
$C(p_{14})$	p_{15}	p_{16}
	p_{15}	p_2
	p_{15}	p_7
	p_9	p_{16}
	p_9	p_2
	p_9	p_7
		p_4
		p_5
		p_8
$C(p_{12})$	p_9	p_{16}
	p_9	p_2
	p_9	p_7

An examination of Table 6.2 shows that there are no hazardous markings, a result which is to be expected because the Petri net of Figure 6.2 is only an abstract version of Figure 4.6(a), that was found to contain no hazardous markings.

This section has shown the use of concurrency sets in producing a more abstract Petri net model of Figure 4.6(a). This abstraction reduces the information contained in the Petri net model but preserves the functionality of the system modelled. Also, this abstraction is limited to the level of abstraction required and by the structure of the Petri net model.

6.5 Conclusion and discussion

In [Skeen 83] concurrency sets were applied to the area of transaction processing in distributed databases and used to determine the independent recoverability of atomic commit protocols in the presence of failures. Specifically, these sets were used to detect inconsistent global states, in which sites in the distributed database made a decision to commit and abort on a particular transaction.
This chapter applied concurrency sets to Petri net models of a hard real-time system, which do not involve the consideration of protocols. The main reason for their use was to provide a means of generating partial reachability graphs of the Petri net models under consideration. Since concurrency sets were associated with the detection of inconsistent

states arising in commit protocols under failure conditions, this chapter performed similar analysis by detecting the presence of inconsistent states but not under failure conditions.

The application of concurrency sets to the Petri net model of Figure 4.6(a) showed that they provide an effective means of detecting the presence of hazardous and inconsistent states for a particular part of the Petri net. Moreover, when these sets are used in conjunction with the Petri net model, a partial reachability graph of the net could be easily generated. Hence, the approach of generating concurrency sets and using them to construct a partial reachability graph provides a means of examining both the static reachable markings and the sequences of markings.

The minimum concurrency set is a useful subset of the set of all concurrency sets of the Petri net model and has the attribute of containing a list of all the reachable markings of the net. It is in effect a subset of the reachability graph because the reachability graph contains all the reachable markings as well as all sequences of markings. This set provides information that would be beneficial in the analysis procedures for the Petri net based techniques examined in this thesis, because it allows:

(i) the semantics of all reachable markings to be examined, therefore, hazardous and inconsistent markings could be easily detected,
(ii) the generation of a partial reachability graph (in conjunction with the Petri net model) pertaining to any system state.

Although the minimum concurrency set is potentially useful in the analysis of Petri net based techniques, further work needs to be done on these sets to:

(i) develop better heuristics for deducing the places that constitute this set, rather than resorting to the approach used in this thesis,
(ii) provide an automated means of the generating these sets.

An important limitation of the minimum concurrency set is that it is impossible to generate for a Petri net containing an infinite set of reachable markings.

The 'reduction' of the Petri net model by identifying identical concurrency sets provides a means of producing a more abstract model of the system, ie the abstraction of Figure 4.6(a) to Figure 6.2. Although this approach reduces the information contained in the original net, it produces a net that exhibits the same behaviour as its original version.

Chapter 7

The synthesis and safety analysis of a real-time controller

7.1 Introduction

The Petri net models proposed in Chapters 4 - 6 were essentially developed for evaluating the usefulness of the Petri net based techniques applied in this thesis. However, a careful examination of these models show that they comprise an abstract model of the causal description of the motion of the drum and slider, and the synchronization logic, which effectively coordinates the motion of these mechanisms. Hence, if these models are used to design and implement a controller for the drum and slider system, then the functionality of the controller needs to be elaborated. For instance, consider the Petri net model of Figure 6.1 which has been conceptually partitioned in Figure 7.1 to emphasize the models of the motion of the drum and slider, and the synchronization logic (referred to as the controller). This shows that the controller is only partially represented by:

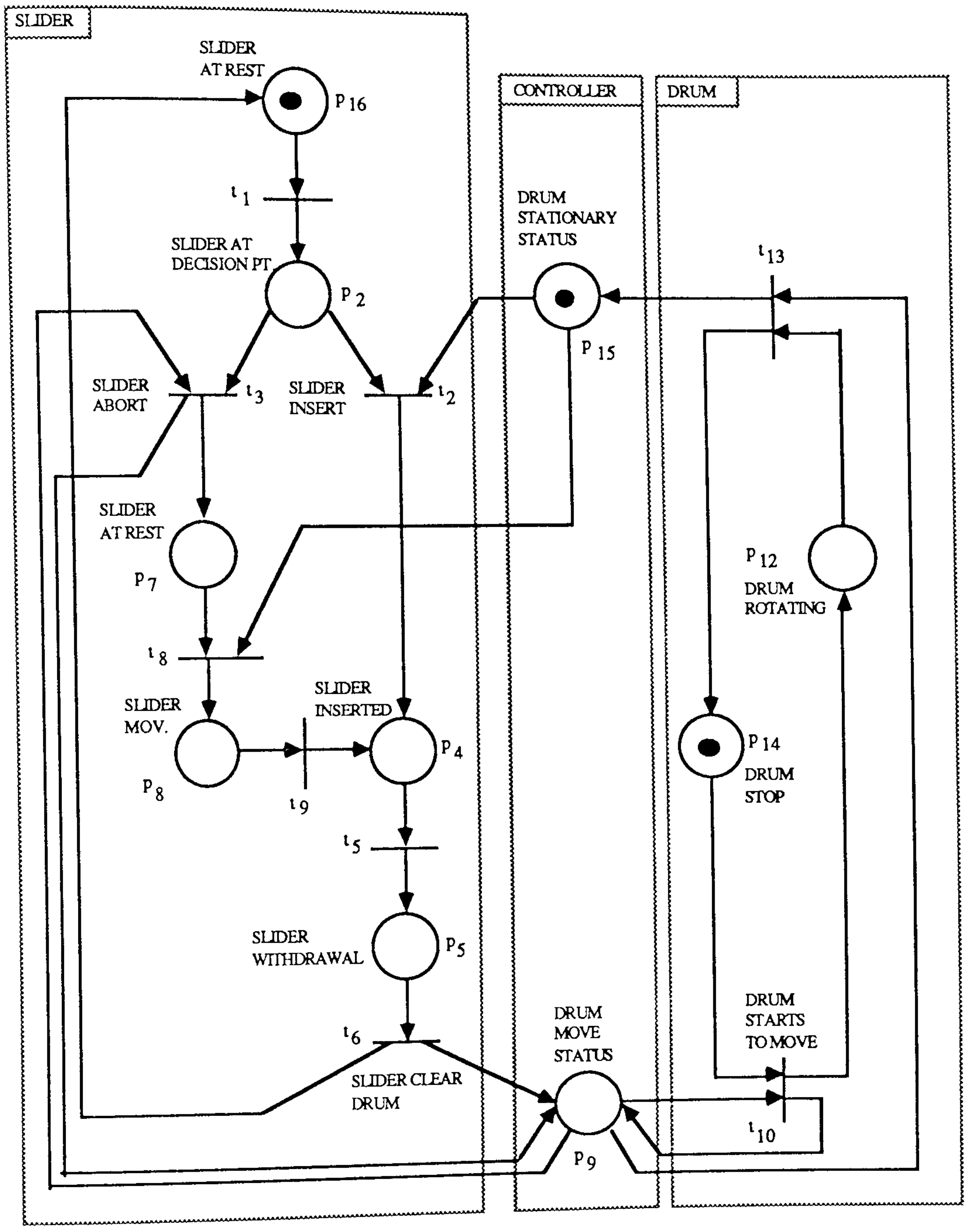
(i) the interlocks of places p_9 and p_{15},
(ii) the controller outputs composed:
  (a) arcs from place p_{15} to transitions t_2 and t_8,
  (b) self-loops between place p_9 and transitions t_3 and t_{10},
  (c) an arc from place p_9 to t_{13},
(iii) the controller input signals composed:
  (a) an arc from transition t_6 to place p_9,
  (b) an arc from transition t_{13} to place p_{15}.

However, Figure 7.1 does not explicitly show the functionality of the controller and thus a controller algorithm cannot be formed directly from this model. Therefore, the aims of this chapter are:

(i) to propose a strategy for synthesizing a controller from the Petri net model of Figure 7.1. This yields a Petri net comprising abstract models of the controller and the physical system, which represents the drum and slider mechanisms,
(ii) the development of techniques for analysing the controller/physical system Petri net using concurrency sets.

Some aspects of the work presented in this chapter have appeared in [Sagoo 92b].

Figure 7.1 Partitioned form of the Petri net model of Figure 6.1, showing the motion of the drum and slider, and the controller

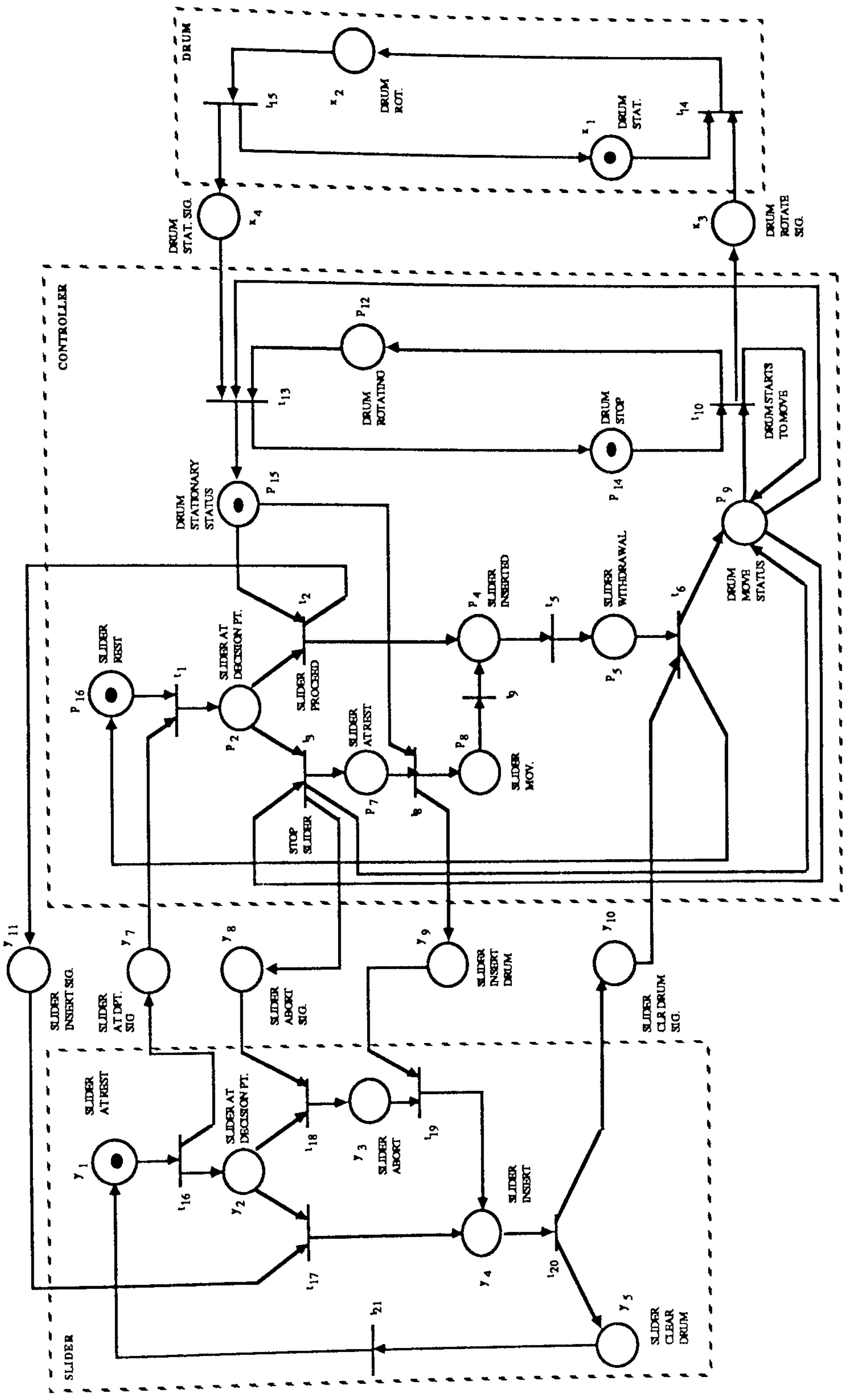


7.2 Synthesis of the controller

The strategy proposed for synthesizing a controller from Figure 7.1 is to embed this Petri net as the controller into a Petri net consisting of the physical system. This approach is taken because it allows the analysis performed on the model of Figure 7.1 to be inherited in the design of the controller, which can be subsequently used to analyse the controller/physical system Petri net. Hence, this approach would have the effect of preserving proven features of the controller.

Using this approach a controller is developed which contains a model of the processes in the physical system. However, if this embedded controller is to coordinate the physical system without causing a collision of the mechanisms then, there must be maximum interaction between the controller and the physical system. This type of interaction is necessary so that the controller can accurately predict the position of the mechanisms, in the physical system, and provide 'correct' and timely responses. These interactions can take the form of signals which allow the controller to sample the physical system, or send outputs to the physical system, or receive inputs from the physical system. Figure 7.2 shows one possible way of embedding a controller into the physical system.

Figure 7.2 Controller design obtained using transition sampling



7.2.1 Types of controller/physical system interactions

The task of embedding the controller into the physical system needs to address the issue of how to synchronise the model of the controller and physical system. Such synchronizations must specify or model the interactions between the controller and the physical system. In particular, how inputs from the physical system are sampled and registered by the controller, and how the controller provides relevant outputs to the physical system. By using a method of trial and error various ways of forming these interactions were considered. This produced the following categories of interaction which are referred to as:

(i) transition sampling
(ii) place sampling

In transition sampling the interaction between the controller and the physical system is achieved through dedicated places. For instance, if a controller sends an output signal to the physical system, this is represented in Petri nets by an input arc from a controller transition to a dedicated place, which in turn connects an output arc to a transition in the physical system. Figures 7.3(a) and (b) shows examples of transition sampling.

Figure 7.3(a) This Petri net shows how the controller receives an input signal from the physical system, in which the dedicated place is p5

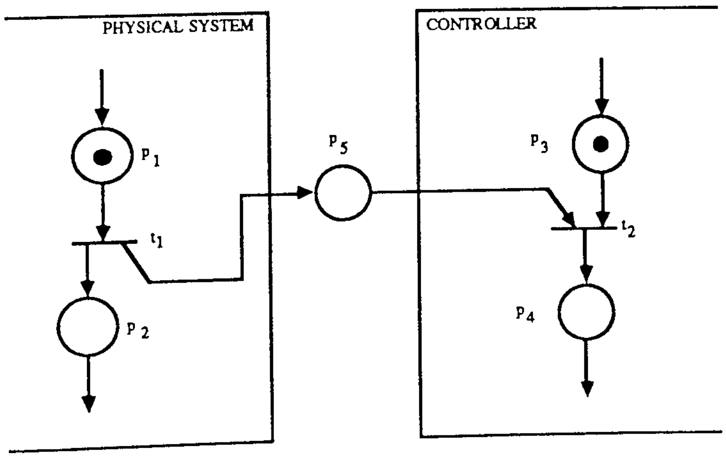
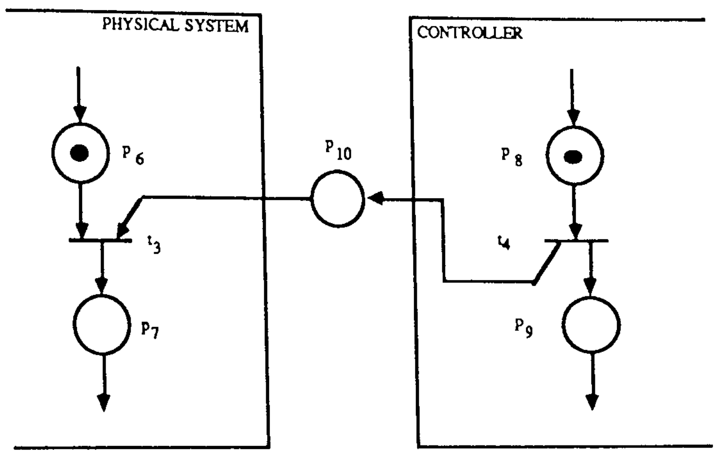


Figure 7.3(b) This Petri net illustrates how the controller sends an output signal to the physical system, in which the dedicated place is p_{10}



An examination of Figures 7.3(a) and (b) shows that transition sampling forms a type of interaction in which the process sending a signal does not wait for an acknowledgment from the process receiving the signal. For instance, in Figure 7.3(a) when transition t_1 fires places p_2 and p_5 are tokenised. This allows the physical system to continue executing whilst the controller accepts the synchronization signal (represented by p_5) by firing t_2. However, if t_1 fired and the controller had not reached p_3 then the controller can still accept the synchronization signal (p_5) by firing t_2 at a later stage. Thus, this type of sampling 'forces' the process receiving the signal to synchronise, but the process sending the signal does not have to wait for the synchronization to take place. Hence, the nature of transition sampling makes it suitable for applications in which an asynchronous transfer of signals or messages is required.

Place sampling forms a type of interaction between the controller and the physical system in which, if the controller sends an output signal to the physical system, a place from the controller Petri net is used to form a self-loop with a transition of the physical system Petri net. Thus the controller place enables a transition in the physical system. Place sampling is illustrated in Figures 7.4(a) and (b).

Figure 7.4(a) This Petri net shows how the controller receives an input from the physical system using place sampling

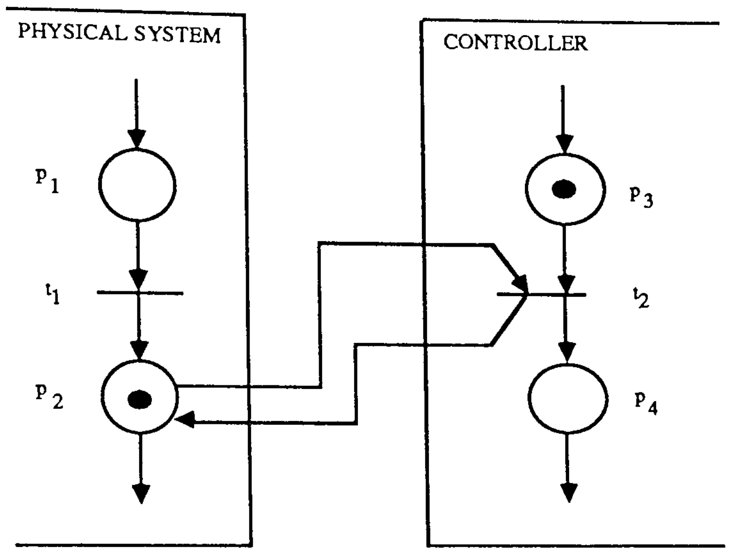
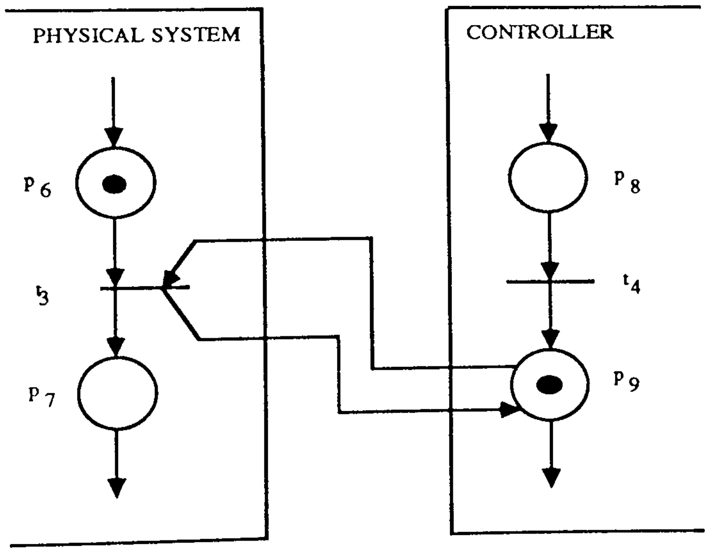


Figure 7.4(b) This Petri net shows how the controller sends an output signal to the physical system using place sampling



An examination of Figures 7.4(a) and (b) shows that place sampling forms a type of interaction in which both processes must be at specific points in their execution for the signal transfer to take place correctly. For instance, consider the Petri net of Figure 7.4(a) in which the physical system is at state given by place p2 and sends an output signal to the controller, which is at state given by p3. The controller receives the signal when t2 fires, which subsequently causes the tokenisation of p4 and the re-tokenisation of p2. However, if the controller had not reached p3 and the physical system was at p2, then the physical system would either wait for the controller to reach p3 or it would continue with its execution sequence, in which case synchronization would not occur. This form of sampling is synchronous, in which the process sending a signal waits for an acknowledgment from the process receiving the signal. Hence, this type of sampling would be suitable for systems requiring synchronous transfer of signals.

Since the above interactions are composed of signals which provide the controller with information about the relative position of the physical system, as well as allowing it to send outputs to the physical system, it is necessary to consider which signals should form this interaction. In the case of the drum and slider system, if the controller is to ensure a collision-free operation of the mechanisms, by providing timely responses to the slider at the decision point, then it must have the following information:

(i) when the slider reaches the decision point,
(ii) when the slider has cleared the drum,
(iii) when the drum has completed its rotation.

The controller needs positional information (i) in order to send the slider either an insert signal, if the drum is at position (iii), or an abort signal, if the drum is rotating. However, if controller receives this information inaccurately or in an untimely manner then a collision of the mechanisms may occur. Also the controller needs to know position information (ii) in order to enable the drum to start rotating.

7.2.2 Petri net models of the controller/physical system

This section uses the strategy for embedding the controller (Section 7.2) and the ways of forming the controller/physical system interactions (Section 7.2.1), to produce Petri net models of the controller/physical system. Using this information three categories of Petri net models have emerged. These models are distinguished by the way in which the controller and the physical system interactions are formed, ie:

(i) Petri net models formed using transition sampling,
(ii) Petri net models formed using place sampling,
(iii) Petri net models formed using a combination of transition and place sampling, which is referred to as hybrid sampling.

The use transition sampling yields the Petri net model of the controller/physical system shown in Figure 7.2. In this Petri net the positional information discussed in Section 7.2.1 (i), (ii) and (iii) is represented by places y_7, y_{10} and x_4 respectively. Specifically the semantics of these places are:

y_7: the slider is at the decision point,

y_{10}: the slider has cleared the drum and has completed its insertion,

x_4: the drum is stationary.

The controller output signals to the physical system are given by places:

y_8: this represents the signal which causes the slider to abort motion,

y_9: this represents the signal which causes the slider to insert into the drum; this signal is sent after the slider has taken the abort motion,

y_{11}: this represents the signal which causes the slider, which is at the decision point, to insert into the drum slider,

x_3: this represents the signal that causes the drum to rotate.

The semantics of the Petri net structure representing the motion of the drum (places x_1 and x_2) and slider (places y_1, y_2, y_3, y_4 and y_5) in the physical system should be obvious from Figure 7.2.

Place sampling yields the controller/physical system Petri net shown in Figure 7.5. Again, the positional information discussed in Section 7.2.1(i), (ii) and (iii) is shown by the Petri net structure representing the self-loops between y_2 and t_1, y_5 and t_6, and x_1 and t_{13} respectively. The remaining self-loops between the controller and physical system places and transitions represent controller outputs (ie p_4 and t_{17}, p_7 and t_{18}, p_8 and t_{19}, p_{12} and t_{14}) and have the same interpretations as the controller outputs discussed for Figure 7.2.

Petri net models formed by hybrid sampling are based on combining transition and place sampling. This has the effect of using a mixture of synchronous and asynchronous transfer of signals between the controller and physical system. Specifically, two types of Petri net models result from this type of sampling, which are represented by Figures 7.6 and 7.7. Hybrid sampling used in Figure 7.6 is referred to as hybrid sampling-I. In this type of sampling the controller outputs are represented by transition sampling and the controller inputs are represented by place sampling. The interaction signals for this Petri net can be easily discerned on comparison with Figures 7.2 and 7.5. Figure 7.7 shows the type of hybrid sampling referred to as hybrid sampling-II. In this net the controller and drum interactions are represented using transition sampling, and the controller and slider interactions are represented using place sampling. Again the interpretation of the interaction signals used in this Petri net can be easily obtained by examining Figures 7.2 and 7.5.

This section has produced several Petri net designs for the controller that can coordinate the motion of the drum and slider. These Petri net models only differ by the way in which the controller and physical system are synchronised. These synchronizations are achieved using either place sampling, transition sampling or a combination of each. In order to assess the usefulness of each type synchronization the following section analyses each Petri net using the notion of concurrency sets developed in Chapter 6.

Figure 7.5 Controller design obtained using place sampling.

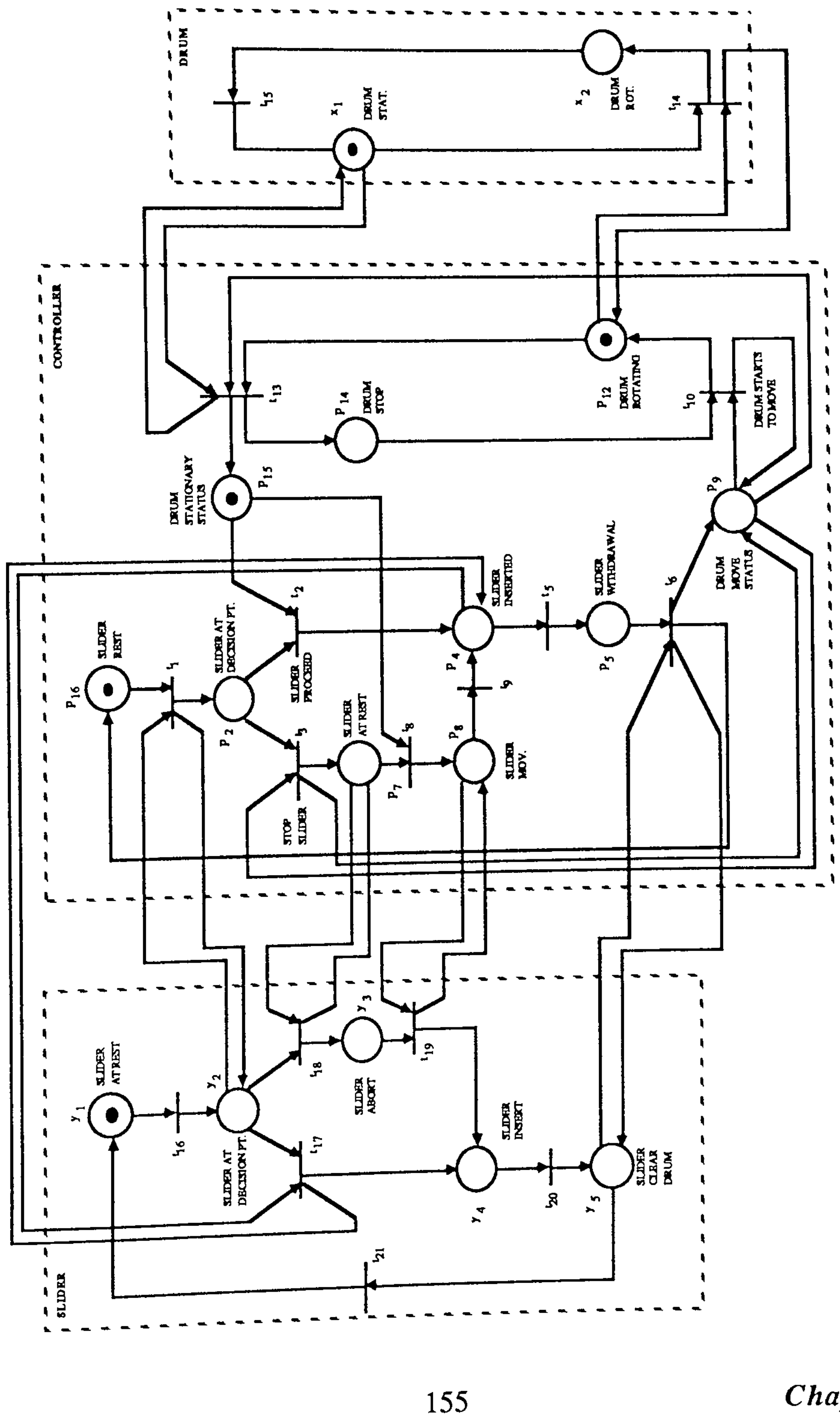


Figure 7.6 Controller design obtained using hybrid sampling-I

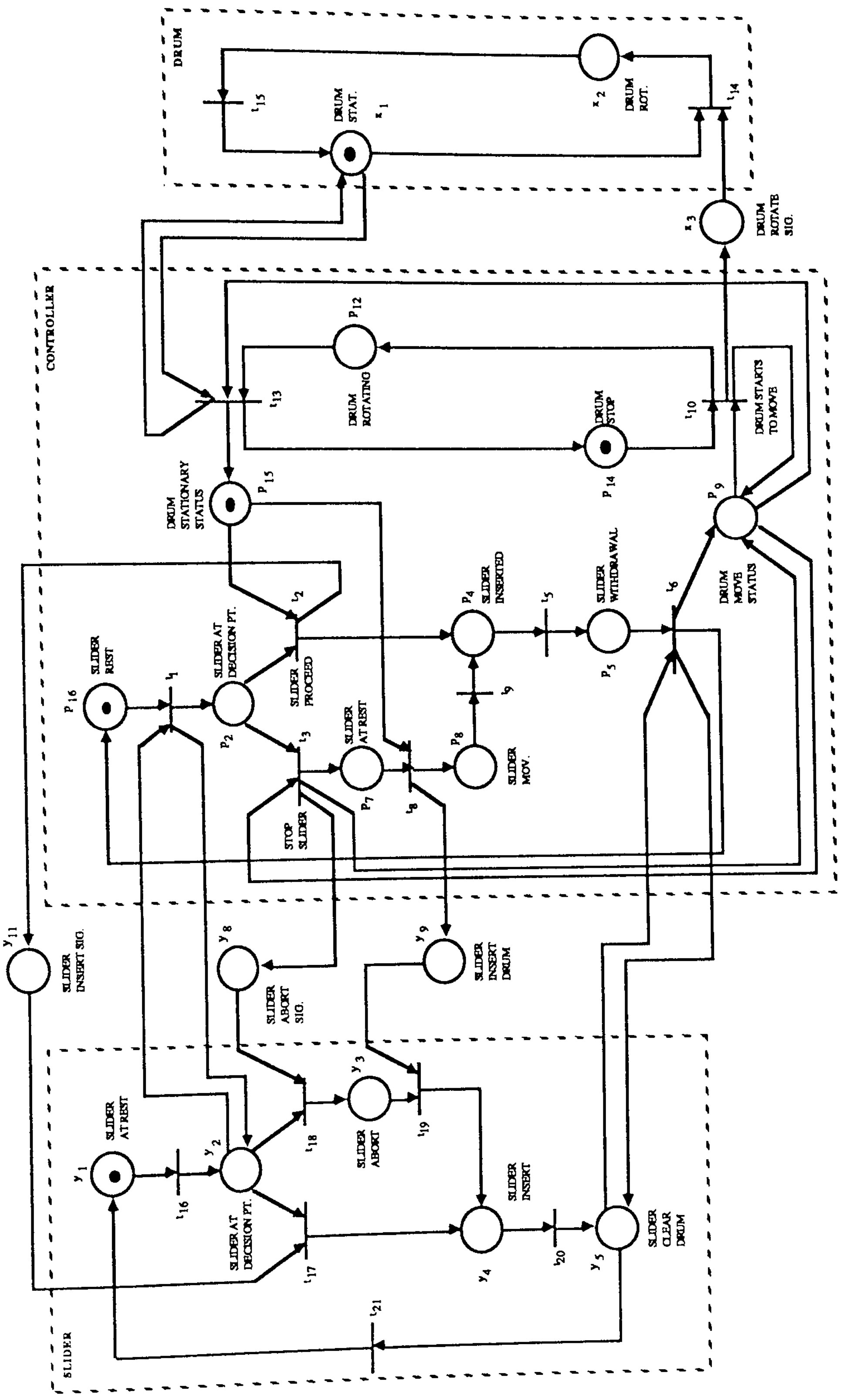
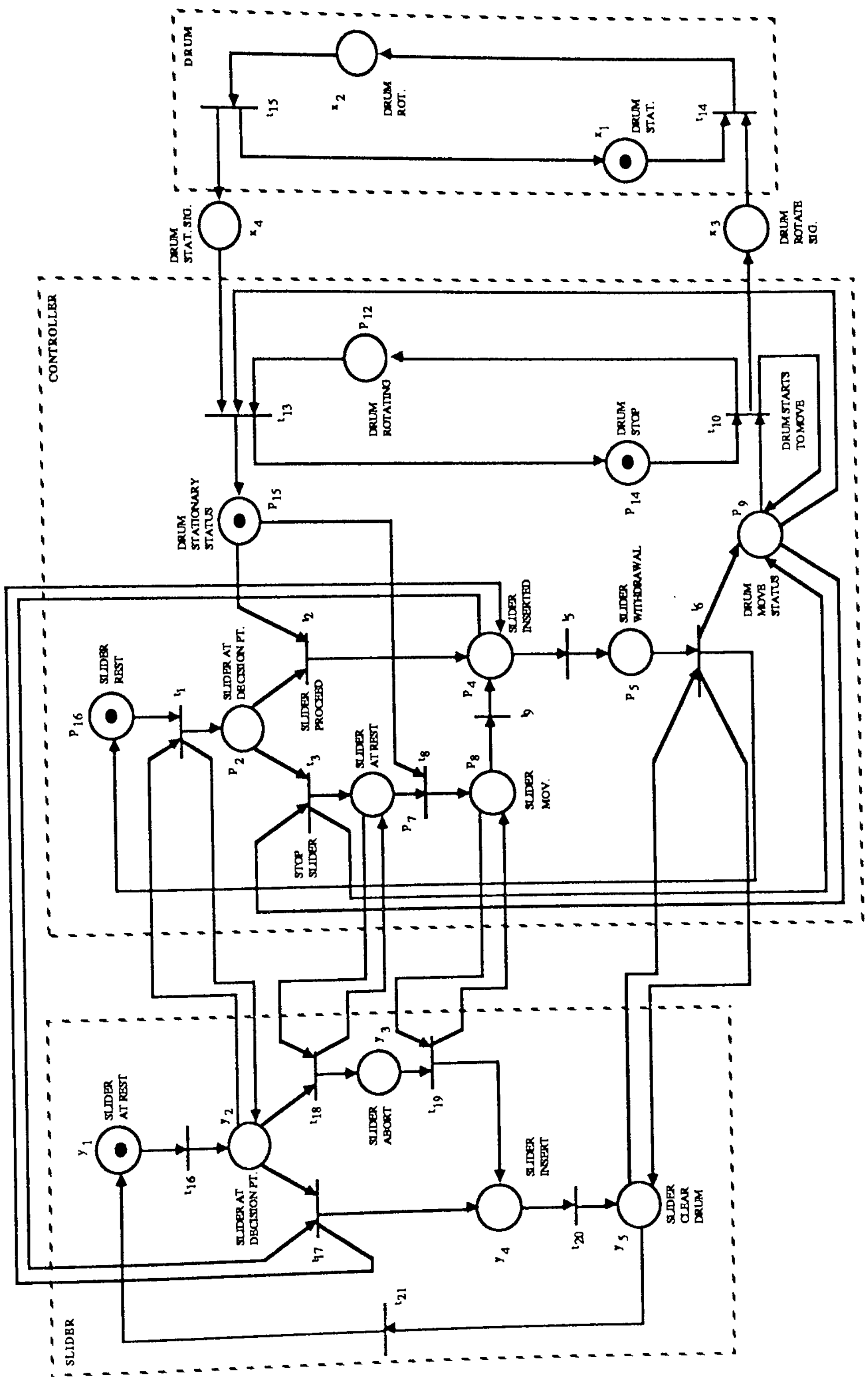


Figure 7.7 Controller design obtained using hybrid sampling-II



7.3 Analysis of the controller/physical system Petri nets

Since the controller (Figure 7.1) is embedded into the Petri net models of the controller/physical system (ie Figures 7.2, 7.5 - 7.7) and it has already been analysed, this section uses the controller analysis to analyse the controller/physical system Petri net. In particular, it is observed that all the reachable markings of the controller can be easily determined by either modifying the reachability graph of Figure 4.6(b), or inspecting the concurrency set for this net in (see Chapter 6). This section analyses the controller/physical system Petri net by using the definition of concurrency sets to determine all combination of places in the physical system Petri net for each controller marking. This approach is used because it allows the identification of any undesirable markings and the behaviour of the Petri net to be determined.

7.3.1 Analysis of the controller/physical system Petri net using transition sampling

Using the concurrency set analysis shown in Chapter 6 the set of all reachable markings for the controller (Figure 7.1) are represented in Table 7.1. For each marking in Table 7.1 the position of the drum and slider, in the physical system of Figure 7.2, can be easily deduced by using the definition of concurrency sets. This task can be systematized by the following:

(i) select a controller marking (represented as MC_n, where $0 \leq n \leq 11$, in Table 7.1),

(ii) for this controller marking use the definition of concurrency sets (see Definition 6.1, Section 6.2) to determine the states in which the slider can concurrently exist in,

(iii) repeat step (ii) to determine the concurrent states of the drum.

If the above procedure is applied to each controller marking then the concurrency sets for the controller/physical system are produced. Table 7.2 shows these sets for Figure 7.2.

The notation used in Table 7.2 to represent the concurrency set for each controller marking is as follows. Consider the first row of Table 7.2 (in which the controller marking is given by: $MC_0 = \{p_{16}, p_{15}, p_{14}\}$) which can be written as:

$$\{y_5, y_1, (y_2, y_7)\}, \{MC_0\}, \{x_1\} \qquad 7.1$$

The set shown by Equ 7.1 produces the following set of possible controller/physical system markings (M_T):

$$M_T = \{y_5, MC_0, x_1\}, \{y_1, MC_0, x_1\}, \{(y_2, y_7), MC_0, x_1\} \qquad 7.2$$

Hence, the concurrency set for the controller marking MC_0 can be easily deduced from Equ.7.2 as:

$$C(MC_0) = \{(y_5, x_1), (y_1, x_1), ((y_2, y_7), x_1)\} \qquad 7.3$$

The interpretation of Equ. 7.3 is that, when the controller is at state MC_0 the position of the mechanisms, in the physical system, are such that the slider has either cleared the drum (y_5), or it is at rest (y_1), or it is at the decision point (y_2, y_7), whilst the drum remains stationary and in position (x_1).

Table 7.1 All possible reachable markings for the Petri net of Figure 7.1

MARKING	PLACES
MC_0	P_{14}, P_{15}, P_{16}
MC_1	P_2, P_{14}, P_{15}
MC_2	P_4, P_{14}
MC_3	P_5, P_{14}
MC_4	P_9, P_{14}, P_{16}
MC_5	P_9, P_{12}, P_{16}
MC_6	P_2, P_9, P_{12}
MC_7	P_7, P_9, P_{12}
MC_8	P_7, P_{14}, P_{15}
MC_9	P_8, P_{14}
MC_{10}	P_2, P_9, P_{14}
MC_{11}	P_7, P_9, P_{14}

Table 7.2 C_{min} for the Petri net of Figure 7.2

SLIDER MARKINGS	CONTROLLER MARKINGS	DRUM MARKINGS
$y_5, y_1, (y_2, y_7)$	(p_{16}, p_{15}, p_{14}) 0	x_1
y_2	(p_2, p_{15}, p_{14}) 1	x_1
$(y_3, y_9), (y_2, y_8, y_9), (y_2, y_7, y_{10}), (y_1, y_{10}), (y_5, y_{10}), y_4, (y_2, y_{11})$	(p_4, p_{14}) 2	x_1
$(y_3, y_9), (y_2, y_8, y_9), (y_2, y_7, y_{10}), (y_1, y_{10}), (y_5, y_{10}), y_4, (y_2, y_{11})$	(p_5, p_{14}) 3	x_1
$(y_3, y_9), (y_2, y_8, y_9), (y_2, y_7, y_{10}), (y_1, y_{10}), (y_5, y_{10}), y_4$	(p_8, p_{14}) 9	x_1
$y_5, y_1, (y_2, y_7)$	(p_{16}, p_9, p_{14}) 4	x_1
y_2	(p_2, p_9, p_{14}) 10	x_1
$y_3, (y_2, y_8)$	(p_7, p_9, p_{14}) 11	x_1
$y_3, (y_2, y_8)$	(p_7, p_{15}, p_{14}) 8	x_1
$y_5, y_1, (y_2, y_7)$	(p_{16}, p_9, p_{12}) 5	$(x_1, x_3), x_2, (x_1, x_4)$
y_2	(p_2, p_9, p_{12}) 6	$(x_1, x_3), x_2, (x_1, x_4)$
$y_3, (y_2, y_8)$	(p_7, p_9, p_{12}) 7	$(x_1, x_3), x_2, (x_1, x_4)$

An inspection of the markings produced from Table 7.2 show that no hazardous markings exist in the Petri net model of Figure 7.2 which show a collision of the mechanisms. However, certain markings are identified that can lead to time-critical hazardous sequences and cause a collision of the mechanisms. These markings are given by the following:

$$M = \{(y2, y7, y10), (p4, p14), x1\} \tag{7.4}$$
$$M = \{(y2, y7, y10), (p5, p14), x1\} \tag{7.5}$$
$$M = \{(y2, y7, y10), (p8, p14), x1\} \tag{7.6}$$
$$M = \{(y2, y7), (p16, p9, p14), x1\} \tag{7.7}$$
$$M = \{(y2, y7), (p16, p9, p12), (x1, x3)\} \tag{7.8}$$
$$M = \{(y2, y7), (p16, p9, p12), (x2)\} \tag{7.9}$$
$$M = \{(y2, y7), (p16, p9, p12), (x1, x4)\} \tag{7.10}$$
$$M = \{y2, (p2, p9, p12), (x1, x3)\} \tag{7.11}$$
$$M = \{y2, (p2, p9, p12), (x2)\} \tag{7.12}$$
$$M = \{y2, (p2, p9, p12), (x1, x4)\} \tag{7.13}$$
$$M = \{y2, (p2, p9, p14), x1 \tag{7.14}$$

An inspection of the semantics of Equ. 7.4 shows that this marking represents the state of the physical system such that, the slider has signalled to the controller that it has cleared the drum and it is at the decision point, whilst the drum is stationary. However, the controller assumes the slider has inserted into the drum which is stationary and in position. Since the

slider's decision period is time-critical and therefore it needs a response from the controller to either abort or continue inserting, in subsequent markings, the controller must reach place p_2 and fire transition t_3 in a timely manner. If the controller fails to provide this timely response then the slider will enter a 'no-decision' state (see Section 4.2.1.1). In this case the slider will continue to move towards the drum. This situation is unspecified and will lead to a collision of the mechanisms if the drum starts to rotate. Markings of Equ. 7.5 and 7.6 depict the same conditions in the physical system and have the same implications as Equ. 7.4
In the marking of Equ. 7.7 the physical system is in a similar state to that of markings of Equs. 7.4 - 7.6, except that the slider has only signalled to the controller that it has entered the decision point and the drum is rotate enabled. In this case the controller needs to tokenise p_2 and fire t_3, in subsequent markings, in a timely manner. This situation would prevent the slider from reaching a 'no-decision' state and thus proceeding towards the drum, which can rotate at any time. This situation is also represented by the markings of Equs. 7.8 - 7.14.

A careful examination of the markings obtained from Table 7.2 shows the existence of inconsistent markings. These markings represent a situation in which the controller assumes that the physical system is at a certain position, when actually the physical system is in a completely different position (also see Section 6.3). For example, such markings exist in the concurrency set for controller marking $MC_2 = (p_4, p_{14})$ which can be represented by the following:

$$C(MC_2) = \{(y_2, y_{11}, x_1), (y_4, x_1), (y_5, y_{10}, x_1), (y_1, y_{10}, x_1), (y_2, y_7, y_{10}, x_1), (y_2, y_8, y_9, x_1), (y_3, y_9, x_1)\}. \qquad 7.15$$

The above set shows that all combination of states corresponding to the position of the drum in the physical system and the controller's model of the drum are consistent. However, inconsistencies arise between the controller's model of the slider and the actual slider which are represented by the following elements of $C(MC_2)$:

(i) (y_1, y_{10}, x_1),
(ii) (y_2, y_7, y_{10}, x_1).

The place combination (y_1, y_{10}, x_1) represents the physical system as, the slider is at rest and has signalled to the controller that it has cleared the drum, but the controller assumes the slider has inserted into the drum. The inconsistencies in place combination (y_2, y_7, y_{10}, x_1) have already been revealed by examining Equ. 7.4. The main consequences of these combination of places is that they can form part of a time-critical sequence that can lead to a collision of the mechanisms, as discussed above. Hence inconsistent markings are undesirable. Further inconsistent markings are produced in Table 7.2 by:

(i) C(MC_3): in particular the place combinations (y_2, y_7, y_{10}) and (y_1, y_{10}),
(ii) C(MC_9): in particular the place combinations (y_2, y_7, y_{10}) and (y_1, y_{10}).

However, they have similar interpretations as the inconsistent markings of C(MC_2). The inconsistent markings of Figure 7.2 and their associated time-critical sequences can be prevented from occurring by ensuring that the controller responds to the physical system in a timely manner. This type of analysis can be performed using timed Petri nets (see Section 4.2.1.2).

The analysis of the embedded controller obtained using transition sampling, Figure 7.2, has shown that the Petri net does not contain any hazardous markings. However, inconsistent markings can occur which form part of hazardous sequences that may lead to a collision of the mechanisms. These undesirable situations can be prevented from occurring by imposing timing constraints on the Petri net, using timed Petri net analysis. The approach used to analyse the controller/physical system Petri net of Figure 7.2 provided a simple means of using analysis previously performed on the controller (ie Table 7.1) to obtain state information about the controller/physical system Petri net. However, it is not clear whether this approach produced all possible combinations of the states in the Petri net of Figure 7.2. Hence, in order to assess the effectiveness of this analysis approach this issue needs to be addressed and the following section is devoted to this cause.

7.3.1.1 Minimum concurrency set for the controller/physical system Petri net using transition sampling

One possible way of showing that the analysis approach used in Section 7.3.1 produced all possible state information of the controller/physical system Petri net is to show that the concurrency sets of Table 7.2 is in fact the minimum concurrency set (C_{min}) of Figure 7.2. It is sufficient to show this because in Chapter 6 it was shown that the C_{min} of a Petri net contains all possible reachable markings. Hence, it follows that the use of this set allows all possible combinations of states to be determined. This section demonstrates that Table 7.2 is the C_{min} of Figure 7.2 by using the following procedure:

(i) the concurrency set for each place in Figure 7.2 is generated and their associated markings determined,

(ii) the marking for each concurrency set shown in Table 7.2 is generated,

(iii) each marking produced by (i) is compared with each marking produced by (iii),

(iv) if the comparison of (iii) shows that all the markings in (i) are contained in (ii) then it has been shown that Table 7.2 is the C_{min} of Figure 7.2. However, if (iii) showed that all the markings of (i) are not contained in (ii) then Table 7.2 is not a C_{min} and thus the analysis approach of Section 7.3.1 requires further state information about the controller/physical system to provide a 'complete' state analysis.

The concurrency sets and associated markings of (i) and the markings of (ii) are shown in Appendix D. Comparing these markings, by performing (iii), shows that the markings of (i) are indeed contained in (ii). Hence, it can be concluded that Table 7.2 is the C_{min} of Figure 7.2. This result allows the following conclusion to be drawn about the analysis approach of Section 7.3.1. Since the markings of Table 7.1 can be determined from the C_{min} of Figure 7.1 and Table 7.1 was used to determine Table 7.2, then effectively the C_{min} of Figure 7.1 was used to deduce the C_{min} for Figure 7.2. The implications of this is that C_{min} can be used as an analysis tool in conjunction with the strategy for synthesizing the controller (discussed in Section 7.2), to effectively analyse the controller/physical system Petri net. The following sections will use the analysis approach of Section 7.3.1 to analyse the controller/physical system Petri nets formed using place sampling and hybrid sampling, and compare these results with those obtained for Figure 7.2.

7.3.2 Analysis of the controller/physical system Petri net using place sampling

This section examines the Petri net of Figure 7.5 which uses place sampling to achieve the interaction between the controller and the physical system. Using the procedure outlined in Section 7.3.1 and the results obtained in Section 7.3.1.1 the C_{min} for the controller/physical system Petri net of Figure 7.5 can be deduced, and is shown in Table 7.3. An inspection of the markings of this C_{min} reveals that, although no hazardous markings exist, which show a collision of the two mechanisms, some markings will deadlock and may lead to a collision of the mechanisms. For instance, deadlock will occur when the controller is at marking $MC_3 = (p_5, p_{14})$ and the global marking of the controller/physical system is:

$\{y_2, MC_3, x_1\}$ 7.16

$\{y_3, MC_3, x_1\}$ 7.17

Some markings may lead to deadlock because it is possible to reach the markings shown in Equ. 7.16 and 7.17, for example:

$\{y_3, MC_2, x_1\}$ 7.18

$\{y_1, MC_3, x_1\}$ 7.19

$\{y_2, MC_9, x_1\}$ 7.20

$\{y_1, MC_9, x_1\}$ 7.21

An inspection of the semantics of marking, Equ. 7.16, shows that the slider is at the decision point and the drum is stationary. However, the controller cannot enable the slider to proceed with the insertion or abort motion. Since the slider remains in a 'no-decision' state and the drum cannot rotate, unless the slider completes the insertion motion, the system remains deadlocked. However, the slider is required to make a decision while in motion; hence the slider in the state of 'no-decision' is still moving towards the drum. If this state persists then the slider will insert into the drum (which is stationary), complete the insertion motion and come to rest. If the system remains in this state of deadlock forever, then the slider can make an infinite number of insertions into a stationary drum. This motion is not explicitly modelled in Figure 7.5 but can only be inferred from the semantics given to the places and transitions of the Petri net. Additionally, since this motion represents unspecified motion of the mechanisms, it must be prevented from occurring. Typically, this can be done by imposing timing constraints using the timed Petri net analysis of Chapter 4.
Similarly the marking of Equ. 7.17 represents the situation in which the slider has aborted its insert motion and is stationary; the drum is also stationary. However, the controller cannot enable the insert motion of the slider, thus the slider is unable to proceed with its insert motion and the drum is unable to rotate, because the slider has not completed its insert motion. The controller, on the other hand, is unable to proceed because the slider has not cleared the drum. Thus, in this state of deadlock all processes remain stationary. Again this type of situation can be prevented from occurring by imposing timing constraints as shown in Chapter 4 using timed Petri net theory.

In terms of marking sequences, a marking sequence in which the slider remains in a 'no-decision' state, whilst the remaining processes are executing, is hazardous. This is particularly crucial if the drum is in the process of rotating as shown by:

$$\sigma_1 = \{y_2, M_5, x_1\}, \{y_2, M_{11}, x_1\}, \{y_2, M_7, x_1\}, \{y_2, M_7, x_2\}, \ldots \qquad 7.22$$

$$\sigma_2 = \{y_2, M_5, x_1\}, \{y_2, M_{11}, x_1\}, \{y_2, M_7, x_1\}, \{y_2, M_2, x_1\}, \{y_2, M_3, x_1\}, \{y_2, M_4, x_1\} \text{ - Deadlock occurs} \qquad 7.23$$

Furthermore the markings of Equs. 7.16 - 7.21 are inconsistent because the controller assumes that the slider has inserted or will insert but the slider, in the physical system, is either approaching the decision point or at the decision point.

Table 7.3 C_{min} for Figure 7.5

SLIDER MARKINGS	CONTROLLER MARKINGS	DRUM MARKINGS
y_5, y_1, y_2	(p_{16}, p_{15}, p_{14}) 0	x_1
y_2	(p_2, p_{15}, p_{14}) 1	x_1
y_3, y_2, y_4, y_5, y_1	(p_4, p_{14}) 2	x_1
y_3, y_2, y_4, y_5, y_1	(p_5, p_{14}) 3	x_1
y_3, y_2, y_4, y_5, y_1	(p_8, p_{14}) 9	x_1
y_5, y_1, y_2	(p_{16}, p_9, p_{14}) 4	x_1
y_2	(p_2, p_9, p_{14}) 10	x_1
y_2, y_3	(p_7, p_9, p_{14}) 11	x_1
y_2, y_3	(p_7, p_{15}, p_{14}) 8	x_1
y_5, y_1, y_2	(p_{16}, p_9, p_{12}) 5	x_1, x_2
y_2	(p_2, p_9, p_{12}) 6	x_1, x_2
y_2, y_3	(p_7, p_9, p_{12}) 7	x_1, x_2

The above situation of deadlock and associated markings did not occur in the controller/physical system Petri nets based on transition sampling, Figure 7.2, instead they are peculiar to the controller/physical system Petri net based on place sampling.

7.3.3 Analysis of the controller/physical system Petri net using hybrid sampling

Two types of controller/physical system Petri nets formed using hybrid sampling will be examined in this section. These nets are shown in Figures 7.6 and 7.7. The following shows the analysis of each type of Petri net using concurrency sets.

7.3.3.1 Analysis of the controller/physical system Petri net using hybrid sampling-I

Application of the procedure described in Section 7.3.1 to the Petri net of Figure 7.6 shows that it is impossible to generate a C_{min} for this Petri net. This is because each place in Figure 7.6 produces an infinite concurrency set, which means that this Petri net has an infinite set of reachable markings and thus an infinite reachability graph. For example, in order to show how easily an infinite set of markings may be generated from Figure 7.6 consider the controller at marking $MC_0 = (p_{16}, p_{15}, p_{14})$ and its associated concurrent markings in the physical system. The following sequence of markings can be generated:

$$\begin{aligned}\sigma = \; & \{y_1, MC_0, x_1\}, \{y_2, MC_1, x_1\}, \; \dots, \; \{y_5, MC_4, x_1\}, \\ & \{y_2, MC_5, (x_1, x_3)\}, \{y_5, MC_0, (x_1, x_3)\}, \; \dots, \\ & \{y_2, MC_1, (x_1, x_3)\}, \; \dots, \; \{y_5, MC_4, (x_1, x_3)\}, \\ & \{y_5, MC_5, (x_1, x_3, x_3)\} \qquad 7.24\end{aligned}$$

From Equ. 7.24 it can be inferred that, if the drum remains in position (x_1, x_3) then there will be an accumulation of tokens in place x_3. This situation could lead to an infinite number of markings, thus producing an infinite reachability graph. A similar situation can occur for the remaining controller markings. Therefore, the Petri net of Figure 7.6 is not a condition event (CE) net (see Section 2.2.1) because multiple tokens can exist in places; however it can be restricted to a CE net by imposing timing constraints. In order to analyse the behaviour of this Petri net the following considers the concurrency sets which do not have multiple tokens in places. These sets are shown in Table 7.4.

Table 7.4 Concurrency sets for the Petri net of Figure 7.6

SLIDER MARKINGS	CONTROLLER MARKINGS	DRUM MARKINGS
y_5, y_1, y_2	(p_{16}, p_{15}, p_{14}) 0	$(x_1, x_3), x_2, x_1$
y_2	(p_2, p_{15}, p_{14}) 1	$(x_1, x_3), x_2, x_1$
$(y_3, y_9), (y_2, y_8, y_9), y_5, y_1, y_2, y_4, (y_2, y_{11})$	(p_4, p_{14}) 2	$(x_1, x_3), x_2, x_1$
$(y_3, y_9), (y_2, y_8, y_9), y_5, y_1, y_2, y_4, (y_2, y_{11})$	(p_5, p_{14}) 3	$(x_1, x_3), x_2, x_1$
$(y_3, y_9), (y_2, y_8, y_9), y_5, y_1, y_2, y_4$	(p_8, p_{14}) 9	$(x_1, x_3), x_2, x_1$
y_5, y_1, y_2	(p_{16}, p_9, p_{14}) 4	$(x_1, x_3), x_2, x_1$
y_2	(p_2, p_9, p_{14}) 10	$(x_1, x_3), x_2, x_1$
$y_3, (y_2, y_8)$	(p_7, p_9, p_{14}) 11	$(x_1, x_3), x_2, x_1$
$y_3, (y_2, y_8)$	(p_7, p_{15}, p_{14}) 8	$(x_1, x_3), x_2, x_1$
y_5, y_1, y_2	(p_{16}, p_9, p_{12}) 5	$(x_1, x_3), x_2, x_1$
y_2	(p_2, p_9, p_{12}) 6	$(x_1, x_3), x_2, x_1$
$y_3, (y_2, y_8)$	(p_7, p_9, p_{12}) 7	$(x_1, x_3), x_2, x_1$

An inspection of the markings of Table 7.4, show that hazardous and inconsistent markings do exist, which deadlock and show a collision of the mechanisms. These unacceptable markings are discussed in detail in Appendix D. However, in order to illustrate the types of hazardous and inconsistent markings that can arise in Figure 7.6, the following considers the presence of such markings with the controller marking $MC_0 = \{p_{16}, p_{15}, p_{14}\}$:

The inconsistent markings associated with controller marking $M_0 = \{p_{16}, p_{15}, p_{14}\}$ are:

$$M = \{y_1, M_0, (x_1, x_3)\} \quad 7.25$$
$$M = \{y_2, M_0, (x_1, x_3)\} \quad 7.26$$
$$M = (\{y_5, M_0, (x_1, x_3)\} \quad 7.27$$
$$M = \{y_1, M_0, x_2\} \quad 7.28$$
$$M = \{y_2, M_0, x_2\} \quad 7.29$$
$$M = \{y_5, M_0, x_2\} \quad 7.30$$

Markings of Equs 7.25 - 7.30 show inconsistencies between the states of the controller and the drum because the controller assumes the drum is at rest (represented by p_{14} and p_{15}), whilst, in the physical system, the drum is either rotate enabled (represented by x_1 and x_3) or rotating (represented by x_2). Moreover, the markings of Equs. 7.26 - 7.30 also show inconsistencies between the controller and the slider, because the slider is either at the decision point (shown by y_2) or clears the drum after insertion (represented by y_5), but the controller assumes that the slider is at rest (given by p_{16}). These inconsistent markings are hazardous because they can form part of hazardous sequences and lead to a collision of the mechanisms. Specifically, these sequences represent the situation in which the slider decides to insert into the drum, whilst the drum is rotating (as shown by the sequence formed by Equs 7.25 - 7.27) or is rotate enabled (as formed by the sequence of Equs 7.28 - 7.30).

The analysis of the controller/physical system Petri net formed using hybrid sampling-I has shown the existence of markings which deadlock and markings which actually show a collision of the mechanisms; such severe conditions did not arise in Petri nets formed by transition or place sampling. Moreover, this Petri net produces an infinite set of reachable markings, which makes analysis of its behavioural properties impossible.
In order to prevent the above disastrous conditions from occurring timing constraints must be enforced. However, it is appears that these timing constraints would be much more stringent than those necessary for the Petri nets obtained using place or transition sampling. The analysis of this Petri net leads to the conclusion that hybrid sampling-I produces a net in which many hazardous problems can occur. Hence, this type of sampling should be avoided.

7.3.3.2 Analysis of the controller/physical system Petri net using hybrid sampling-II

The procedure of Section 7.3.1 yields a C_{min} for Figure 7.7 which is shown in Table 7.5. An inspection of the semantics of the markings in Table 7.5 shows that, although no markings represent a collision of the mechanisms, some markings do show the occurrence of deadlock. Specifically, these markings are:

$$M = \{y_3, MC_2, x_1\} \qquad 7.31$$
$$M = \{y_2, MC_3, x_1\} \qquad 7.32$$
$$M = \{y_3, MC_3, x_1\} \qquad 7.33$$
$$M = \{y_2, MC_9, x_1\} \qquad 7.34$$

Equs. 7.31 and 7.34 represent markings which will lead to deadlock, and markings of Equs. 7.32 and 7.33 actually show that deadlock has occurred. It is interesting to note that the markings which lead to deadlock, or deadlock, in Figure 7.7 also occurred in the Petri nets obtained by place sampling in Figure 7.5. In fact, an inspection of the markings of

Table 7.5 and those of Table 7.3 show that the controller of Figure 7.7 and 7.5 both produce similar markings and coordinate the physical system in the same manner. Thus both Petri nets may be considered to exhibit equivalent behaviours.

Table 7.5 C_{min} for the Petri net of Figure 7.7

SLIDER MARKINGS	CONTROLLER MARKINGS	DRUM MARKINGS
y_5, y_1, y_2	(p_{16}, p_{15}, p_{14}) 0	x_1
y_2	(p_2, p_{15}, p_{14}) 1	x_1
y_3, y_2, y_4, y_5, y_1	(p_4, p_{14}) 2	x_1
y_3, y_2, y_4, y_5, y_1	(p_5, p_{14}) 3	x_1
y_3, y_2, y_4, y_5, y_1	(p_8, p_{14}) 9	x_1
y_5, y_1, y_2	(p_{16}, p_9, p_{14}) 4	x_1
y_2	(p_2, p_9, p_{14}) 10	x_1
y_2, y_3	(p_7, p_9, p_{14}) 11	x_1
y_2, y_3	(p_7, p_{15}, p_{14}) 8	x_1
y_5, y_1, y_2	(p_{16}, p_9, p_{12}) 5	$(x_1, x_3), x_2, (x_1, x_4)$
y_2	(p_2, p_9, p_{12}) 6	$(x_1, x_3), x_2, (x_1, x_4)$
y_2, y_3	(p_7, p_9, p_{12}) 7	$(x_1, x_3), x_2, (x_1, x_4)$

7.4 Controller reduction heuristics

Previous sections analysed the controller/physical system Petri net models of Figures 7.2, 7.5 - 7.7 and produced the following results:

(i) transition sampling (Figure 7.2) produced a Petri net that did not contain any hazardous markings but, it did contain a hazardous sequences of states that could be removed by applying timing analysis,

(ii) place sampling (Figure 7.5) produced a Petri net that caused deadlock and contained hazardous markings sequence, which again could be removed by applying timing constraints,

(iii) hybrid sampling-I (Figure 7.6) produced a Petri net that caused many hazardous situations to occur, such a collision of the mechanisms.

(iv) hybrid sampling-II (Figure 7.7) produced a Petri net that produced the same behaviour as the net of Figure 7.5. Hence the similarities between the behaviours of Figures 7.7 and 7.5 showed that place sampling and hybrid sampling-II are equivalent.

From the above it can be seen that the Petri net produced using transition sampling yields the most acceptable form of controller. This section examines this Petri net with the aim of determining the minimum amount of control necessary to coordinate the mechanisms in the physical system. This reduction is performed because:

(i) it is considered that the strategy of embedding the controller may have produced a controller Petri net that contains 'redundant' places and transitions,

(ii) to determine the minimum amount of interaction required by the controller to safely coordinate the mechanisms of the physical system. This will also show the minimum information needed by the controller, concerning the relative positions of the drum and slider, in order to coordinate these mechanisms in a safe manner.

The following heuristics (which were developed by inspection) have been useful for reducing the Petri net:

(i) merging places that occur sequentially between sampling points into a single place. This merging is only applied to places that have similar semantics. For instance, places which represent the motion of the drum and have the following semantics can be merged into a single place, drum accelerating, drum at constant velocity and drum decelerating, because they correspond to the drum in motion. Such places are always characterised by having identical concurrency sets (see Section 6.4),

(ii) systematic removal of the sampling points between the controller and the physical system. This will determine the minimum interaction needed to coordinate the mechanisms of the physical system in the required manner. The effect of removing the sampling points would be to deprive the controller of information concerning the state of the physical system.

Heuristic (i) can be applied to Figure 7.2 by observing that places p_4, p_5 and p_8 have identical concurrency sets. Moreover, an inspection of their semantics: p_4, slider fully inserted into drum, p_5, slider with drawing from full insertion, and p_8, slider accelerating towards stationary drum, show that they represent the situation in which either the slider is inside the drum or is moving towards the drum and will insert. Hence, the notion of abstracting p_4, p_5, p_8 into a single place, which is given the semantics of the slider inserts into the drum is plausible; to achieve this places p_5 and p_8 can be merged into p_4. A further inspection of Figure 7.2 shows that the semantics of p_{12}, drum at constant velocity, and p_{14}, drum is stationary, are essentially duplicated by p_9, drum rotating status, and p_{15}, drum stationary status, respectively. Hence, it seems feasible to remove this duplication by removing p_{12} and p_{14} from the Petri net. If the preceding reductions are implemented on Figure 7.2 then the Petri net of Figure 7.8 is obtained. The C_{min} for this net is shown in Table 7.6, which can be easily derived by inspection of Table 7.2. An inspection of the markings in Table 7.6 shows that no new hazardous markings are obtained. Moreover, the controller of Figure 7.8 essentially produces similar markings and controls the physical system in the same manner as the controller of Figure 7.2. Hence both nets are considered to exhibit equivalent behaviours.

Figure 7.8 A reduced controller/physical system Petri net of Figure 7.2

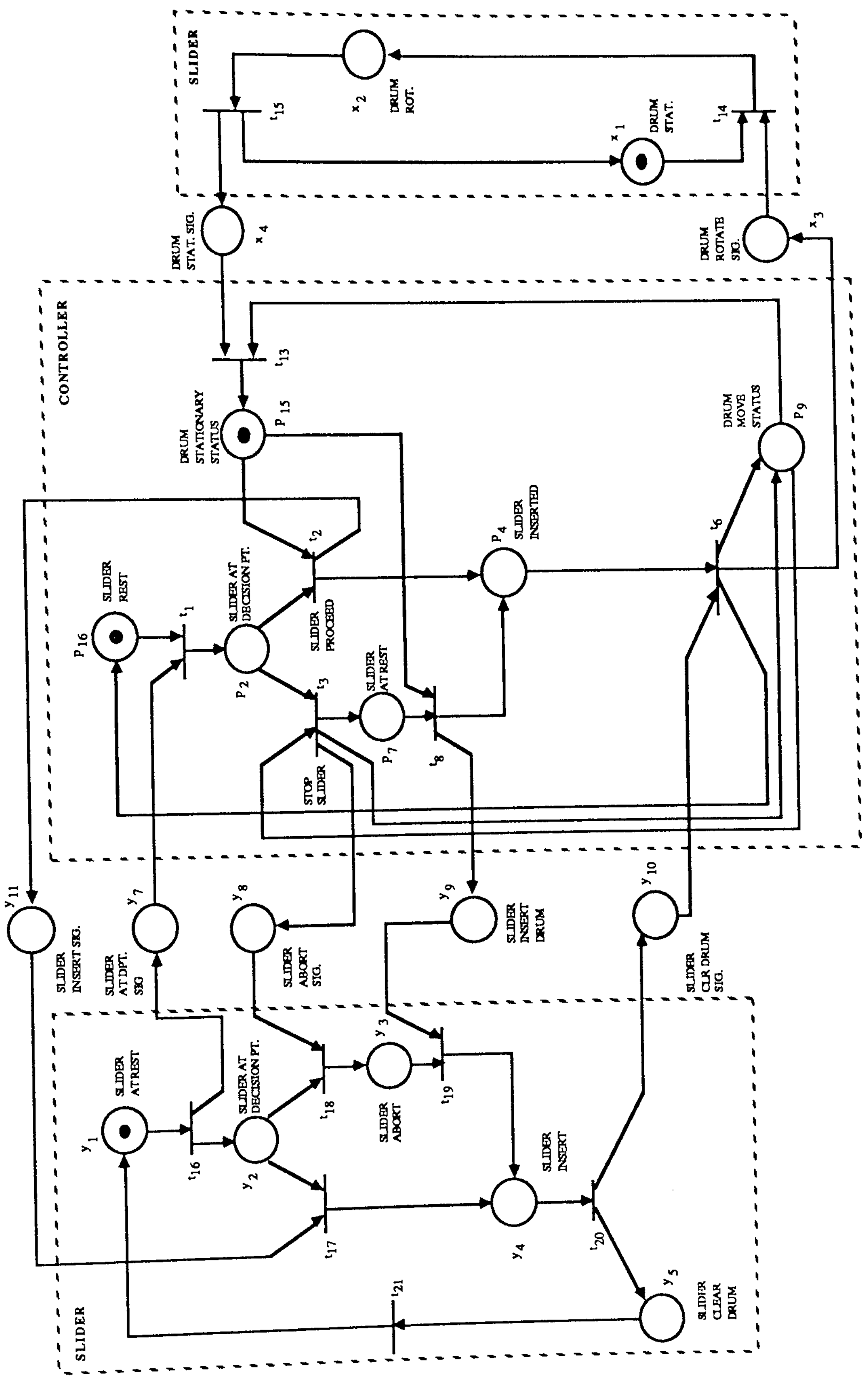


Table 7.6 C_{min} for the Petri net model of Figure 7.8

SLIDER MARKINGS	CONTROLLER MARKINGS	DRUM MARKINGS
$y_5, y_1, (y_2, y_7)$	(p_{16}, p_{15}) 0	x_1
y_2	(p_2, p_{15}) 1	x_1
$(y_3, y_9), (y_2, y_8, y_9), (y_2, y_7, y_{10}), (y_1, y_{10}), (y_5, y_{10}), y_4, (y_2, y_{11})$	p_4 2	x_1
$y_5, y_1, (y_2, y_7)$	(p_{16}, p_9) 4	$(x_1, x_3), x_2, (x_1, x_4)$
y_2	(p_2, p_9) 10	$(x_1, x_3), x_2, (x_1, x_4)$
$y_3, (y_2, y_8)$	(p_7, p_9) 11	$(x_1, x_3), x_2, (x_1, x_4)$
$y_3, (y_2, y_8)$	(p_7, p_{15}) 8	x_1

A reduction in the maximum interaction between the controller and the physical system of Figure 7.8 can be performed using heuristic (ii), by removing the controller input y_7. This controller input represents the point at which the slider is at the decision point; its removal allows the merger of p_{16} into p_2 (via heuristic (i)). Thus, from the Petri net model of Figure 7.8 in which heuristic (ii) is applied the reduced Petri net of Figure 7.9 results. This net's C_{min} is shown in Table 7.7.

An inspection of the markings in Table 7.7 shows that, this reduced Petri net does not present any new hazardous markings or hazardous sequences compared to the nets of Figures 7.2 and 7.8. In fact, the controller of this Petri net coordinates the mechanisms of the physical system in the same way as Figures 7.2 and 7.8; hence the former can be regarded as producing equivalent behaviours as the latter. However, any further removal of sampling points will cause catastrophic conditions to arise, such as a collision of the mechanisms and accumulation of tokens in certain places. In order to prevent these conditions from occurring very stringent timing constraints need to be imposed for the controller to coordinate the mechanisms in a safe manner.

This section has used heuristics to reduce the controller in the Petri net model of Figure 7.2. These heuristics were developed by inspection of the Petri net and allowed the following reductions to be made in the controller of Figure 7.2:

(i) merging of places p_8 and p_5 into p_4,
(ii) removal of places p_{12} and p_{14},
(iii) removal of the sampling point represented by y_7,
(iv) merging of place p_{16} into place p_2.

Using analysis based on concurrency sets it was shown that if the above reduction were made then the resulting Petri net, Figure 7.9, produced that the same behaviour as the Figure 7.2; hence both nets are equivalent.

Figure 7.9 A reduced controller/physical system Petri net of Figure 7.8

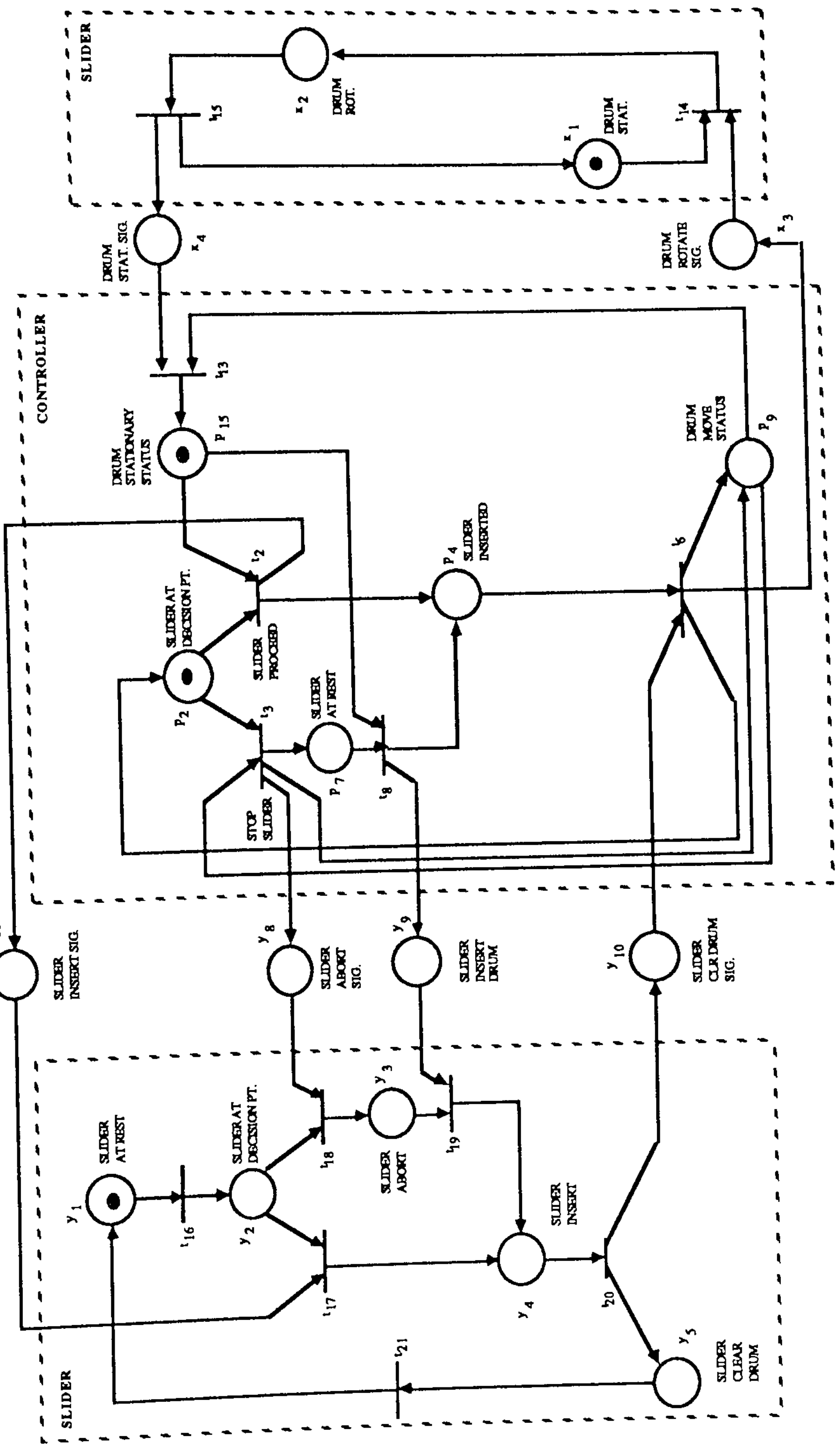


Table 7.7 C_{min} for the Petri net of Figure 7.9

SLIDER MARKINGS	CONTROLLER MARKINGS	DRUM MARKINGS
y_5, y_1, y_2	(p_2, p_{15}) 1	x_1
$(y_5, y_{10}), (y_1, y_{10}), (y_2, y_{10}), (y_5, y_{11}), (y_1, y_{11}), (y_2, y_{11}), (y_3, y_9), (y_5, y_8, y_9), (y_1, y_8, y_9), (y_2, y_8, y_9)$	p_4 2	x_1
y_5, y_1, y_2	(p_2, p_9) 10	$(x_1, x_3), x_2, (x_1, x_4)$
$(y_5, y_8), (y_1, y_8), y_3, (y_2, y_8)$	(p_7, p_9) 11	$(x_1, x_3), x_2, (x_1, x_4)$
$(y_5, y_8), (y_1, y_8), y_3, (y_2, y_8)$	(p_7, p_{15}) 8	x_1

7.5 Comparisons and conclusions

This chapter was primarily concerned with the synthesis and analysis of a controller that could coordinate the motion of the drum and slider in a safe manner. The need for this controller has arisen because previous Petri net models such as Figures 4.6(a) and 7.1 did not contain enough detail to directly obtain a controller for this system. Hence, a strategy was proposed that a synthesized a controller from the Petri net of Figure 7.1, by embedding it as the controller into a Petri net consisting of the physical system. Such a controller contained a model of the processes in the physical system, but needed to be synchronised to the physical system, in order to provide a useful response. The process of synchronising the controller and physical system required consideration of:

(i) which signals should form the interaction between the controller and physical system, and
(ii) how to model these interactions.

From the description of the motion of the slider and drum (see Section 4.1.1) the signals required by the controller to safely coordinate these mechanisms can be derived. A method of trial and error was used to model these interactions. This resulted in the following ways of forming this interaction which were referred to as:

(i) transition sampling,
(ii) place sampling.

Examination of these forms of sampling revealed that transition sampling allows an asynchronous transfer of signals and place sampling provides a synchronous transfer of signals. These forms of sampling were used to embed the controller into the physical system. This resulted in several controller/physical system Petri nets that only differed by the way in which the interaction between the controller and physical system was formed. These Petri nets were classified into the categories of:

(i) controller/physical system Petri net based on transition sampling,
(ii) controller/physical system Petri net based on place sampling,
(iii) controller/physical system Petri nets based on a hybrid sampling. This form of sampling combined transition sampling and place sampling; two types of combinations were considered which were referred to as hybrid sampling-I and hybrid sampling-II.

The attribute of each class of Petri net was assessed by inspecting the various reachable markings using concurrency sets. The analysis of each type of Petri net showed that:

Transition sampling - modelled the controller and the physical system interaction which coordinated the mechanisms of the physical system in a safe manner, ie no markings showed a collision of the mechanisms. Hazardous marking sequences did exist which were characterised by the slider remaining in the 'no-decision' state. However, such sequences were inherited from the Petri net model of Figure 7.1. Moreover, it was noted that these sequences existed in all versions of controller/physical system Petri nets, and could easily be removed by applying timed Petri net theory.

Place sampling - analysis of this type of Petri net produced markings that deadlocked causing the model to exhibit unspecified behaviour. However, these markings did not cause a collision of the mechanisms.

Hybrid sampling-I - this type of Petri net produced a controller that caused several hazardous markings; amongst other situations a collision of the mechanisms was shown. Moreover, this net contained an infinite set of markings, which were primarily caused by an accumulation of tokens in certain places; hence this net is not a CE net.

Hybrid sampling-II - this form of Petri net contrasted sharply against that produced by hybrid sampling-I, because with the exception of several markings causing deadlock, there were no hazardous markings. Furthermore, this Petri net was shown to be equivalent to that produced using place sampling, because both Petri nets controlled the physical system in the same manner and produced that same reachable markings.

The approach used to analyse the controller/physical system Petri net using concurrency sets provided a way of using the analysis already performed on the controller to determine all possible markings in the controller/physical system Petri net. This was facilitated by using the notion of a minimum concurrency set (C_{min}). The use of this set, coupled with the strategy of embedding the Petri net model of Figure 7.1 into the physical system, was considered to be much simpler than manually generating the reachability for determining state information. This is because the C_{min} for Figure 7.1 could be used to easily deduce the C_{min} of the controller/physical system Petri net by the using the definition of concurrency sets.

The above analysis showed that the controller/physical system Petri net obtained using transition sampling produced the most safe design of controller than place sampling, and hybrid sampling.

Reduction of the controller/physical system Petri net obtained using transition sampling was performed to determine the minimum amount of control necessary to coordinate the physical system in the required manner, and to remove any 'redundant' net structures. When selected transitions and places of controller were removed the behaviour of the Petri net remained unchanged. In fact, this result was obtained even when the sampling signal that showed the controller when the slider reached the decision point was removed. However, problems do occur if further removal of sampling signals is performed.

Chapter 8

Towards the unification of Petri net based techniques

8.1 Introduction

The application of temporal Petri nets, extended temporal Petri nets and real-time temporal Petri nets has shown that these techniques can be used to prove particular types of safety and liveness properties. For instance, temporal Petri nets allow safety properties to be represented and liveness properties (which specify the occurrence of particular transition sequences) to be proved; extended temporal Petri nets allow proof of both safety properties (which specify that a certain state never occurs) and liveness properties (specifying the occurrence of particular state sequences); real-time temporal Petri nets allow proof of safety properties (involving precedence constraints), liveness properties and real-time properties.

Formally, each technique generates temporal formulas involving the net marking and either the firing of a transition, or the firing of the set of all transitions in the net, but it is used to prove a specific set of properties. Since each technique makes similar assertions about the Petri net, this has motivated the desire to consider the possibility of combining these techniques into a unified formalism. Such an investigation would need to identify the set of generic formulas that allow the specification and verification of the system modelled using Petri nets. Hence, this chapter aims to compare and contrast the temporal component of temporal Petri nets, extended temporal Petri nets and real-time temporal Petri nets, and considers the feasibility of unifying them. Generic approaches are identified and are used with the notion of concurrency sets to prove system properties for the arbor drum control system discussed in Chapter 7.

8.2 Attributes of the Petri net based techniques

The comparison of temporal Petri nets, extended temporal Petri nets and real-time temporal Petri nets will be illustrated by applying each technique to the Petri net model of Figure 3.1 which was presented in Chapter 3 but is re-produced in this chapter.
The definition of Figure 3.1 in terms of Petri net theory (see Section 2.2(i)) is shown as follows. The Petri net structure of Figure 3.1 is given by:

$C_1 = \{P, T, I, O, M_0\}$ 8.1

where

$P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}$ 8.2

$T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$ 8.3

$M_0 = \{p_1, p_4, p_7\}$ 8.4

$I(t_1) = \{p_1\}$ 8.5 $O(t_1) = \{p_2\}$ 8.11

$I(t_2) = \{p_2, p_7\}$ 8.6 $O(t_2) = \{p_3\}$ 8.12

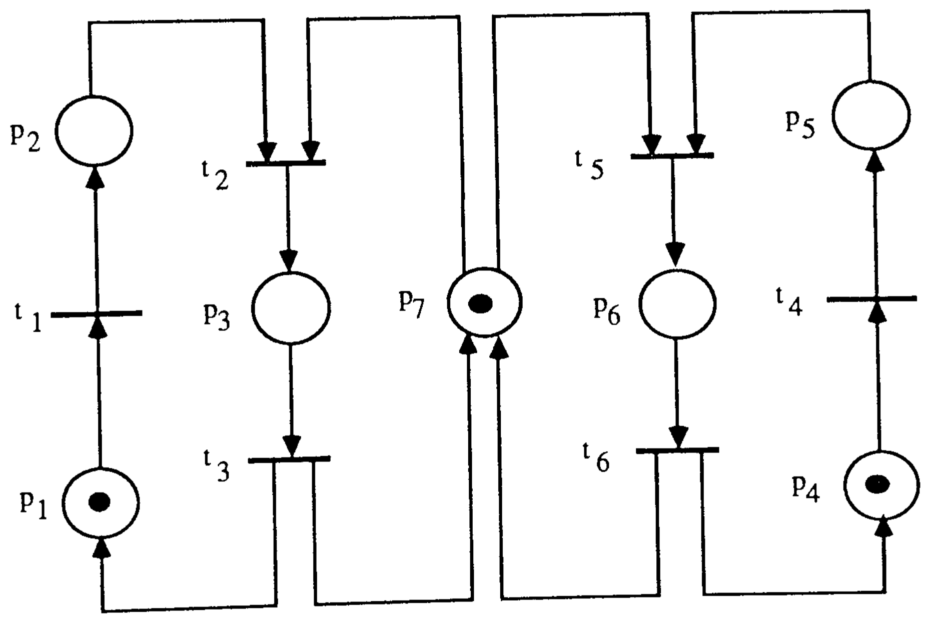
$I(t_3) = \{p_3\}$ 8.7 $O(t_3) = \{p_1, p_7\}$ 8.13

$I(t_4) = \{p_4\}$ 8.8 $O(t_4) = \{p_5\}$ 8.14

$I(t_5) = \{p_5, p_7\}$ 8.9 $O(t_5) = \{p_6\}$ 8.15

$I(t_6) = \{p_6\}$ 8.10 $O(t_6) = \{p_4, p_7\}$ 8.16

Figure 3.1 A Petri net representation of the mutual exclusion problem



8.2.1 Examination of temporal Petri nets

The temporal component of temporal Petri nets is based on formalising the firing of transitions using an event-based logic, which is useful for proving liveness properties. These formalisations were described in Section 3.3.2 and are represented by Propositions 1 and 2. Proposition 1 is applicable to the firing of simple transitions, which must fire once they are enabled, and Proposition 2 is applicable to the firing of transitions which are in conflict. Since the firing of transitions represented by Proposition 1 is a special case of

Proposition 2 (see Section 3.3.2), it seems useful to examine the latter rather than both. Proposition 2 can be written in an abbreviated form as follows:

Proposition 2: For TN = {C, f} in which marking M is reachable from M_0 and transitions $t_m \in T$ and $t_n \in T$ (where m and n are positive integers) are enabled and in conflict.

If (i) $<M_0,\alpha> \models \Box[t_m(ok) \Rightarrow t_1(ok)\mathcal{U}(t_m \vee t_n)] \vee$
$\Box[t_n(ok) \Rightarrow t_2(ok)\mathcal{U}(t_m \vee t_n)]$ 8.17

and (ii) f implies $<M_0,\alpha> \models \Box[t_m(ok) \Rightarrow \Diamond t_m(\neg ok)] \wedge$
$\Box[t_n(ok) \Rightarrow \Diamond t_n(\neg ok)]$ 8.18

(the temporal formulas (f) are interpreted as imposing a restriction on the firing sequence that can be generated by the net. Thus, only those firing sequences which satisfy these temporal formulas are allowed to occur).

Then (iii) $<M_0,\alpha> \models \Box[t_m(ok) \wedge t_n(ok) \Rightarrow \Diamond(t_m \vee t_n)]$ 8.19

Proposition 2 is applied by first checking whether the assertion of Equ. 8.17 holds true for the marking M in which transitions t_m and t_n become firable, then making assertions of Equs. 8.18 and 8.19. For C_1 the special case of Proposition 2, which is essentially Proposition 1 and concerns the firing of non-conflicting transition, is applicable all net markings in which transitions t_1, t_3, t_4 and t_6 are enabled. In this case the following can be asserted:

$<M_0,\alpha> \models \Box[t_1(ok) \Rightarrow \Diamond t_1(\neg ok)]$ 8.20
$<M_0,\alpha> \models \Box[t_3(ok) \Rightarrow \Diamond t_3(\neg ok)]$ 8.21
$<M_0,\alpha> \models \Box[t_4(ok) \Rightarrow \Diamond t_4(\neg ok)]$ 8.22
$<M_0,\alpha> \models \Box[t_6(ok) \Rightarrow \Diamond t_6(\neg ok)]$ 8.23
$<M_0,\alpha> \models \Box[t_1(ok) \Rightarrow \Diamond t_1]$ 8.24
$<M_0,\alpha> \models \Box[t_3(ok) \Rightarrow \Diamond t_3]$ 8.25
$<M_0,\alpha> \models \Box[t_4(ok) \Rightarrow \Diamond t_4]$ 8.26
$<M_0,\alpha> \models \Box[t_6(ok) \Rightarrow \Diamond t_6]$ 8.27

This proposition is also applicable to the firing of transitions t_2 and t_5, except for net marking $M_4 = \{p_2, p_5, p_7\}$ (see Figure 3.2). For this marking both transitions become enabled and are in conflict, hence Proposition 2 is applicable.

The features of temporal Petri nets and their use in the modelling of Petri net models can be summarised as follows:

(i) the firing of a transition is formalised using temporal logic by setting up various propositions. These propositions are generated for the initial marking (M_0), which is unique to the Petri net under consideration and any transition firing sequence (α) that can be generated from it,

(iii) the validity of these propositions is determined for the particular net marking and the enabled transitions under consideration, as demonstrated above,

(iv) the propositions of temporal Petri nets are used to prove specific liveness properties of the model, hence these propositions are only applied to those transitions which are involved in the proof of the property.

8.2.2 Examination of extended temporal Petri nets

Extended temporal Petri net analysis (see Section 3.4.1) is based on a translation algorithm that translates a Petri net model into a temporal logic description. This is achieved by transforming the initial marking into a propositional axiom (referred to as the system dependent axiom) and the set of transition firing rules into inference rules (termed as the system dependent inference rules). This translated description is used in conjunction with the temporal logic proof system to prove properties of the modelled system. The translation of Figure 3.1 yields the following:

system dependent axiom

A1. $p1 \wedge p4 \wedge p7$

system dependent inference rules

IR1. $(p1 \wedge \neg p2) \Rightarrow \mathrm{O}(p2 \wedge \neg p1)$

IR2. $(p2 \wedge p7 \wedge \neg p3) \Rightarrow \mathrm{O}(p3 \wedge \neg p2 \wedge \neg p7)$

IR3. $(p3 \wedge \neg p1 \wedge \neg p7) \Rightarrow \mathrm{O}(p1 \wedge \neg p3 \wedge p7)$

IR4. $(p4 \wedge \neg p5) \Rightarrow \mathrm{O}(p5 \wedge \neg p4)$

IR5. $(p5 \wedge p7 \wedge \neg p6) \Rightarrow \mathrm{O}(p6 \wedge \neg p5 \wedge \neg p7)$

IR6. $(p6 \wedge \neg p4 \wedge \neg p7) \Rightarrow \mathrm{O}(p4 \wedge p7 \wedge \neg p6)$

Since the above translation considers the enabling and firing of each transition separately, it does not represent the situation when transitions are in conflict. However, the issue of conflicting transitions is addressed in extended temporal Petri nets by allowing a transition to fire either non-deterministically or based on a fairness requirement (see Section 3.4.1.2).

The features of extended temporal Petri nets and their modelling of systems can be summarised as follows:

(i) the translation algorithm transforms the complete Petri net model into a description in temporal logic,

(ii) once this translation is achieved there is no need to refer to the Petri net structure C_1, since the initial marking and the set of all transition firing rules of the net is the minimum information needed to:

(a) generate the net's reachability graph

(b) derive the structure of the Petri net,

(iii) the translation of Petri nets into temporal logic makes no explicit mention of the transition involved, instead the change in marking which occurs as a consequence of the transition firing is specified. Thus, for a particular transition, the system dependent inference rule defining the antecedent and consequence of the transition is equivalent to the transition's input and output places respectively. Hence, for the above system, the system dependent inference rules IR1 - IR6 can be re-written by the following formulas respectively:

$$I(t_1) \Rightarrow \bigcirc O(t_1) \qquad 8.28$$
$$I(t_2) \Rightarrow \bigcirc O(t_2) \qquad 8.29$$
$$I(t_3) \Rightarrow \bigcirc O(t_3) \qquad 8.30$$
$$I(t_4) \Rightarrow \bigcirc O(t_4) \qquad 8.31$$
$$I(t_5) \Rightarrow \bigcirc O(t_5) \qquad 8.32$$
$$I(t_6) \Rightarrow \bigcirc O(t_6) \qquad 8.33$$

8.2.3 Examination of real-time temporal Petri nets

Real-time temporal Petri net analysis (see Chapter 5) is based on translating the enabling and firing rules of a transition into temporal formulas concerning the sequence of enabling and firing of a transition. These rules are generally applicable to Petri nets and are defined in Section 2.2. Since real-time temporal Petri nets combine timed Petri nets and RTTL, both qualitative and quantitative temporal formulas can be constructed concerning the firing of a transition. The Petri net of Figure 3.1, C_1, is easily transformed into a timed Petri net (TC_1) by allocating an enabling and firing time (δe_i and δf_i, where $1 \le i \le 6$) to each transition $t_j \in T$, where $1 \le j \le 6$, in C_1.
Thus the net of Figure 3.1 can be represented as the tuple:

$$RTTN_1 = \{TC_1, Tf_1\} \qquad 8.34$$

where

Tf_1 represents the temporal formulas concerning the qualitative and quantitative temporal behaviour of TC_1.

The qualitative and quantitative temporal formulas of Tf_1 are generated as follows:

(i) Qualitative time behaviour of TC_1

(a) enabling rules of t_j

$$I(t_j) \equiv t_j(ok) \qquad 8.35$$

$$M_j \wedge t_j(ok) \Rightarrow O(t_j(c) \vee t_j(\neg ok)) \qquad 8.36$$

(b) firing rules of t_j

If $t_j(f) \Rightarrow M_j$ 8.37

and $t_j(c) \Rightarrow M_k$ 8.38

Then $M_j \wedge t_j(ok) \Rightarrow O(M_k \vee t_j(\neg ok))$ 8.39

(ii) Quantitative time behaviour of TC_1

(a) enabling rules of t_j

$$M_j \wedge t_j(ok) \wedge \tau = \mathcal{T}_1 \Rightarrow O(\neg t_j(c)\ \mathcal{U}\,(\tau \geq \mathcal{T}_1 + \delta e_j)) \qquad 8.40$$

(b) firing rules of t_j

$$M_j \wedge t_j(ok) \wedge \tau = \tau_1 \Rightarrow M_j\ \mathcal{U}\,(M_k \wedge (\tau = \mathcal{T}_1 + \delta e_j + \delta f_j) \vee \neg I(t_j)) \qquad 8.41$$

The modelling of systems using real-time temporal Petri nets can be summarised as follows:

(i) the above set of temporal formulas are applicable to the firing of any type of transition in a Petri net. Hence, they are applicable to transitions which may or may not be in conflict,

(ii) it is not necessary to apply (a) or (b) to all the transitions in the net, but only to those transitions which are encountered in the proof of a certain property.

8.3 Comparison of each technique

From Section 8.2 the basic difference between the above techniques can be summarised as follows.

The propositions of temporal Petri nets formalise the firing of transitions without referring to their input places, output places or the global markings; although the notation of temporal

Petri nets allows such features to be expressed. Hence, these propositions are general in the sense that it is not possible to infer the marking of the net unless the reachability graph is also available. Since these propositions concern the firing of transitions, propositions must be constructed that represent different types of transition firings that can occur, so that the proof of various liveness properties can be performed. For example, it has already been noted that Proposition 1 is applicable to transitions which must fire once they are enabled and Proposition 2 is applicable to the firing of enabled transitions which are in conflict; hence the former cannot be used to prove liveness properties that involve decisions, whilst this is catered for in the latter.

Extended temporal Petri nets formalises the structure of the Petri net in which the firing of a transition is represented by an inference rule, which refers to the transitions input and output places. However, it does not explicitly mention the transition involved as in temporal Petri nets. An inference rule of extended temporal Petri nets provides the same information as Proposition 1 of temporal Petri nets. This can be easily illustrated by considering, the firing of t_1 in C_1 which can be represented in temporal Petri nets as:

$$<M_0,\alpha> \models \Box[t_1(ok) \Rightarrow \Diamond t_1] \qquad 8.42$$

and in extended temporal Petri nets this is represented as:

$$[(p1 \wedge \neg p2) \Rightarrow \bigcirc(p2 \wedge \neg p1)] \qquad 8.43$$

Since extended temporal Petri nets cannot represent transitions explicitly, the equivalence of the above formulas can be established by using temporal Petri nets to derive Equ. 8.43 from Equ. 8.42. Hence, when the input places of t_1 are tokenised, this transition becomes firable and using the notation of temporal Petri nets this can be written as:

$$<M_0,\alpha> \models \Box[(p1 \wedge \neg p2) \Rightarrow t_1(ok)] \qquad 8.44$$

Similarly, when t_1 fires its output places become tokenised instantaneously, hence the following holds true:

$$<M_0,\alpha> \models \Box[t_1 \Rightarrow (p2 \wedge \neg p1)] \qquad 8.45$$

combining Equs. 8.42, 8.44 and 8.45 yields:

$$<M_0,\alpha> \models \Box[(p1 \wedge \neg p2) \Rightarrow \Diamond(p2 \wedge \neg p1)] \qquad 8.46$$

By weakening Equ. 8.43, using the temporal axiom: $Ow \Rightarrow \Diamond w$ where w is any well-formed propositional formula, the following is to produced:

$$[(p1 \wedge \neg p2) \Rightarrow \Diamond(p2 \wedge \neg p1)] \qquad 8.47$$

From the above it can be seen that Equ 8.46 is essentially the same as Equ 8.47, which is a weakened form of Equ 8.43.

Although Proposition 1 of temporal Petri nets has an equivalent representation for the system dependent inference rules of extended temporal Petri nets, the same is not true for Proposition 2, which defines the firing of conflicting transitions. The reason for this is because the situation of conflict is not explicitly formalised in extended temporal Petri nets, however, it is implicitly treated while proving properties. For instance, consider the Petri net of Figure 3.1 at marking $M_4 = \{p_2, p_5, p_7\}$ in which the conflicting transitions t_2 and t_5 are firable. Using temporal Petri nets the firing of these transitions can be represented by Proposition 2 as:

$$<M_0,\alpha> \models \Box[t_2(ok) \wedge t_5(ok) \Rightarrow \Diamond(t_2 \vee t_5)] \qquad 8.48$$

However, in terms of extended temporal Petri nets the firable transitions at M_4 are given by the inference rules:

IR2. $(p2 \wedge p7 \wedge \neg p3) \Rightarrow O(p3 \wedge \neg p2 \wedge \neg p7)$
IR5. $(p5 \wedge p7 \wedge \neg p6) \Rightarrow O(p6 \wedge \neg p5 \wedge \neg p7)$

These inference rules do not explicitly show that the transitions are in conflict instead this can be inferred by applying these rules to the appropriate net marking.

Real time temporal Petri nets is based on translating the enabling and firing rules of a transition into formulas of temporal logic. Since these rules are generally applicable in Petri net theory, their representations in temporal logic should also be applicable to the firing of transitions in general. The obvious advantage of real time temporal Petri nets over temporal Petri nets and extended temporal Petri nets is the ability of the former to specify and verify real-time properties as well as non real-time properties. Hence, the following will compare the qualitative time temporal formulas of this technique.

Real-time temporal Petri nets make the following assertion concerning the enabling and firing of transition $t_j \in T$ in a Petri net given by the tuple RTTN = {TC,Tf}:

$$M_j \wedge t_j(ok) \Rightarrow O(t_j(c) \vee t_j(\neg ok)) \qquad 8.49$$

An examination of Equ. 8.49 shows that it represents the case in which t_j fires (given by the first disjunct of the consequent of Equ. 8.49) or that t_j becomes disabled by the firing of a conflicting transition (given by the second disjunct in the consequent of Equ. 8.49). Hence, Equ. 8.49 contains the same information as Propositions 1 and 2. However, it is more general than Proposition 2 because it does not mention the specific conflicting transition involved; although this may included using the notation of real-time temporal Petri nets. Since Equ. 8.49 is similar to Proposition 2, the following shows how easily it can be used to derive the conclusion of Proposition 1. For the enabling and firing of a non conflicting transition t_j Proposition 1 has the form:

$$<M_0,\alpha> \models \Box[t_j(ok) \Rightarrow \Diamond t_j] \qquad 8.50$$

Since t_j is not in conflict then Equ. 8.49 can be reduced to:

$$M_j \wedge t_j(ok) \Rightarrow O t_j(c) \qquad 8.51$$

By weakening the consequent of Equ. 8.51, using the temporal axiom: $Ow \Rightarrow \Diamond w$ (where w is any well-formed propositional formula), the following can be deduced:

$$t_j(ok) \Rightarrow \Diamond t_j(c) \qquad 8.52$$

In the notation of real-time temporal Petri nets when t_j completes firing this is written as $t_j(c)$, however, in the notation of temporal Petri nets this is represented as t_j. Hence, allowing for the differences in the notation of both techniques, the above has shown that Equ. 8.50 can be derived using real-time temporal Petri nets. It is also worth noting that Equ. 8.50 explicitly mentions the initial marking and the firing sequence as $<M_0,\alpha>$, however, in Equ. 8.51 this feature is implicit; since M_j must be reachable from M_0 via a firing sequence. Hence, Equs. 8.49, 8.51 and 8.52 could be represented using $<M_0,\alpha> \models$. It follows that the temporal formulas of temporal Petri nets can be easily derived from real time temporal Petri nets, in this sense both techniques are closely related.

Similarly, a comparison of real-time temporal Petri nets and extended temporal Petri nets shows that the notation of real-time temporal Petri nets allows the net marking to be constructed as a proposition. Hence the initial marking can be written as a propositional axiom. Moreover, the basic formulas of real-time temporal Petri nets can be used to derive formulas that are equivalent to the inference rules of extended temporal Petri nets theory. For example, consider the firing of t_1 in Figure 3.1 which is represented by extended temporal Petri nets as the inference rule given by Equ 8.43.

In real-time temporal Petri nets the firing of t_1 can be represented as:

$$M_1 \wedge t_1(\mathrm{ok}) \Rightarrow O(M_2 \vee t_1(\neg \mathrm{ok})) \qquad 8.53$$

When transition t_1 becomes firable, it eventually completes firing, hence Equ. 8.53 can be simplified to:

$$M_1 \wedge t_1(\mathrm{ok}) \Rightarrow OM_2 \qquad 8.54$$

Since M_1 also contains the places that enable t_1, ie the input places of t_1, and similarly M_2 contains places that are tokenised when t_1 completes firing , ie the output places of t_1, then Equ. 8.54 can be simplified to:

$$I(t_1) \wedge t_1(\mathrm{ok}) \Rightarrow O\mathcal{O}(t_1) \qquad 8.55$$

Weakening the antecedent of Equ. 8.55 yields Equ 8.28. It can be seen that Equ. 8.28 is essentially equivalent to Equ. 8.43. Hence the above has shown that real-time temporal Petri nets can be used to derive formulas that are equivalent to the system dependent axiom and inference rules of extended temporal Petri nets. Since the system dependent axiom and inference rules are the main components needed for the proving properties using extended temporal Petri nets, this suggests that the proof procedures of extended temporal Petri nets can also be incorporated within real-time temporal Petri nets.

This section has shown that the formalism of real-time temporal Petri nets contains generic formulas (concerning the enabling and firing of a transition) which are sufficient for deriving the propositions of temporal Petri nets, and the axioms and inference rules of extended temporal Petri nets. This is possible because the notation of real-time temporal Petri nets can assert formulas which explicitly mention the net marking and transitions. It is also noted that extended temporal Petri nets cannot derive the exact propositions of temporal Petri nets or the formulas of real-time temporal Petri nets, because their notation does not allow the explicit representation of a transition. However, equivalent formulas in terms of input and output places may be derived. The consequence of the above result is that the analysis of temporal Petri nets and extended temporal Petri nets could be easily performed or incorporated in real-time temporal Petri nets.

8.4 Formal Analysis of the arbor drum control system Petri net

This section essentially describes the application of real-time temporal Petri nets to the formal analysis of the arbor drum control system described in Chapter 7. The properties of this system were analysed by temporal Petri nets, extended temporal Petri nets and real-time temporal Petri nets in Chapters 4 and 5 and are summarised for comparison as follows.

8.4.1 Specification of the Problem

(i) The temporal Petri nets analysis of Section 4.3 verified the following properties of the Petri net model of Figure 4.6(a), which modelled the motion of the drum and slider, and the synchronization logic:

(a) the slider decides to insert and subsequently withdraws from the drum,

$$<M_0,\alpha> \models \Box[t_2 \Rightarrow \Diamond t_6] \qquad 4.22$$

(b) whilst at the decision point the slider must make a decision to either follow an insert or abort motion profile,

$$<M_0,\alpha> \models \Box[p_2 \Rightarrow \Diamond(t_2 \vee t_3)] \qquad 4.23$$

(c) the drum rotation and termination,

$$<M_0,\alpha> \models \Box[t_{10} \Rightarrow \Diamond t_{13}] \qquad 4.24$$

(d) whenever the drum comes to rest the slider makes an insertion,

$$<M_0,\alpha> \models \Box[t_{13} \Rightarrow \Diamond(t_2 \vee t_8)] \qquad 4.25$$

(e) whenever the slider aborts its motion then eventually it will insert and withdraw from the drum.

$$<M_0,\alpha> \models \Box[t_3 \Rightarrow \Diamond t_6] \qquad 4.26$$

(ii) The extended temporal Petri nets of Section 4.4 proved the following properties of the Petri net model of Figure 4.6(a):

(a) a situation will never occur when the slider has taken a decision to insert and the drum is rotating,

$$\Box\neg[p4 \wedge p9] \qquad 4.27$$

(b) a situation will never occur when the slider is at the decision point a no decision is made,

$$\Box\neg[p2 \wedge \neg(p9 \vee p15)] \qquad 4.28$$

(c) if the slider starts moving towards the drum infinitely often then it will insert into the drum infinitely often.

$$\Box\Diamond[p14 \wedge p15 \wedge p16] \Rightarrow \Box\Diamond p4 \qquad 4.29$$

(iii) The real-time temporal Petri nets of Sections 5.3.1 and 5.3.2 verified the following properties of the Petri net model of Figure 4.6(a):

(a) once the drum completes its incremental rotation then the slider must have inserted into the stationary drum prior to the next completion of the drum's rotation,

$$t_{13}(ok) \Rightarrow O[(t_2(ok) \vee t_8(ok))\ \mathbf{P}\ t_{13}(ok)] \qquad 5.34$$

(b) once the slider has taken a decision to insert into the drum, which is stationary, then the drum must have completed its rotation prior to the next insertion of the slider,

$$(t_2(ok) \vee t_8(ok)) \Rightarrow \Diamond[t_{13}\ \mathbf{P}\ (t_2(ok) \vee t_8(ok))] \qquad 5.36$$

(c) when the slider starts its motion then the drum must become stationary within a time constraint for the slider to insert into the drum, without making an abort motion,

$$(p1 \wedge (\tau = 0)) \Rightarrow \Diamond[(p14 \wedge p2) \wedge (\boldsymbol{\delta_{x2}} \leq \tau \leq \boldsymbol{\delta_{x1}})] \qquad 5.37$$

(d) when the slider enters the decision point then a decision must be taken either to continue inserting or abort its motion within a real-time constraint.

$$(p2 \wedge (\tau = 0)) \Rightarrow \Diamond\{[(p3 \wedge p14) \wedge (\tau = \boldsymbol{\delta_{x1}})] \vee [p7 \wedge (\boldsymbol{\delta_3} \leq \tau \leq \boldsymbol{\delta_{x2}})]\} \qquad 5.38$$

Note properties represented by Equs. 5.37 and 5.38 show the timing constraints in bold to prevent confusion with similar symbols used in this chapter.

By translating the above properties in terms of the physical system represented by the Petri net model of Figure 7.2, the particularly relevant properties of this net have been categorised and expressed as follows.

(i) Safety Properties

S1: A situation will never occur when the slider has inserted into the drum and the drum is rotating or rotate enabled.

$$\Box\neg[y4 \wedge x2], \qquad 8.56$$

$$\Box\neg[y4 \wedge (x1 \wedge x3)]. \qquad 8.57$$

S2: A situation will never occur when the slider is at the decision point and no decision is taken by the slider.

$$\Box\neg[y2 \wedge \neg(p9 \vee p15)] \qquad 8.58$$

S3: Once the drum completes its incremental rotation then the slider must have inserted into the stationary drum prior to the completion of the drum's rotation.

$$t_{x2} \Rightarrow \bigcirc[(t_{y2}(ok) \vee t_{y4}(ok))\ \mathbf{P}\ t_{x2}] \qquad 8.59$$

S4: Once the slider has taken a decision to insert into the drum, which must be stationary, then the drum must have completed its rotation prior to the next insertion of the slider.

$$(t_{y2}(ok) \vee t_{y4}(ok)) \Rightarrow \Diamond[t_{13}\ \mathbf{P}\ (t_{y2}(ok) \vee t_{y4}(ok))] \qquad 8.60$$

(ii) Liveness Properties

L1: Whenever the drum comes to rest then the slider must insert into the drum.

$$<M_0,\alpha> \models \Box[t_{x2}(c) \Rightarrow \Diamond(t_{y2}(ok) \vee t_{y4}(ok))] \qquad 8.61$$

L2: Whenever the slider reaches the decision point then the slider eventually inserts into the drum.

$$<M_0,\alpha> \models \Box[t_{y1}(c) \Rightarrow \Diamond(t_{y2}(ok) \vee t_{y4}(ok))] \qquad 8.62$$

L3: Whenever the slider withdraws from the drum then eventually the slider should return to its home position and the drum should rotate.

$$<M_0,\alpha> \models \Box[t_{y5}(c) \Rightarrow \Diamond(t_{y6}(ok) \wedge t_{x1}(ok))] \qquad 8.63$$

(iii) Real-time Properties.

RL1: When the slider starts its motion then the drum must become stationary within a time constraint for the slider to insert into the drum without making an abort motion.

$$(y_1 \wedge (\tau = 0)) \Rightarrow \Diamond[(p_{15} \wedge p_{14} \wedge p_2) \wedge (\delta_{x2} \leq \tau \leq \delta_{x1})] \qquad 8.64$$

RL2: When the slider enters the decision point then a decision must be taken either to continue inserting or abort its motion within a time constraint.

$$(y_2 \wedge (\tau = 0)) \Rightarrow \Diamond\{[(p_3 \wedge p_{14}) \wedge (\tau = \delta_{x1})] \vee [p_7 \wedge (\delta_3 \leq \tau \leq \delta_{x2})]\} \qquad 8.65$$

8.4.2 Formal Analysis

This section illustrates the use of real-time temporal Petri nets to show whether properties S1-S4, L1-L3 and RL1, RL2 are satisfied by the Petri net model of Figure 7.2. These proofs will also make use of the C_{min} for Figure 7.2. which was shown in Table 7.2. Both Figure 7.2 and Table 7.2 are re-produced below for convenience. Although much of the formal analysis presented uses formulas and proof procedures of real-time temporal

Petri nets, the proof of some safety properties are based on using the proof procedures of extended temporal Petri nets.

Table 7.2 C_{min} for the Petri net of Figure 7.1

SLIDER MARKINGS	CONTROLLER MARKINGS	DRUM MARKINGS
$y_5, y_1, (y_2, y_7)$	(p_{16}, p_{15}, p_{14}) 0	x_1
y_2	(p_2, p_{15}, p_{14}) 1	x_1
$(y_3, y_9), (y_2, y_8, y_9), (y_2, y_7, y_{10}), (y_1, y_{10}), (y_5, y_{10}), y_4, (y_2, y_{11})$	(p_4, p_{14}) 2	x_1
$(y_3, y_9), (y_2, y_8, y_9), (y_2, y_7, y_{10}), (y_1, y_{10}), (y_5, y_{10}), y_4, (y_2, y_{11})$	(p_5, p_{14}) 3	x_1
$(y_3, y_9), (y_2, y_8, y_9), (y_2, y_7, y_{10}), (y_1, y_{10}), (y_5, y_{10}), y_4$	(p_8, p_{14}) 9	x_1
$y_5, y_1, (y_2, y_7)$	(p_{16}, p_9, p_{14}) 4	x_1
y_2	(p_2, p_9, p_{14}) 10	x_1
$y_3, (y_2, y_8)$	(p_7, p_9, p_{14}) 11	x_1
$y_3, (y_2, y_8)$	(p_7, p_{15}, p_{14}) 8	x_1
$y_5, y_1, (y_2, y_7)$	(p_{16}, p_9, p_{12}) 5	$(x_1, x_3), x_2, (x_1, x_4)$
y_2	(p_2, p_9, p_{12}) 6	$(x_1, x_3), x_2, (x_1, x_4)$
$y_3, (y_2, y_8)$	(p_7, p_9, p_{12}) 7	$(x_1, x_3), x_2, (x_1, x_4)$

Using real-time temporal Petri nets, the Petri net Figure 7.2. can be represented by the tuple:

$$RTTN_1 = \{TC_1, Tf_1\}$$

where

$TC_1 = \{P, T, I, O, M_0, \delta e, \delta f\}$ and

$P = \{(p2, p4, p5, p7, p8, p9, p12, p14, p15, p16),$
$(y1, y2, y3, y4, y5, y7, y8, y9, y10, y11), (x1, x2, x3, x4)\}$

$T = \{(ty1, ty2, ty3, ty4, ty5, ty6), (t1, t2, t3, t5, t6, t8, t9, t10, t13),$
$(tx1, tx2)\}$

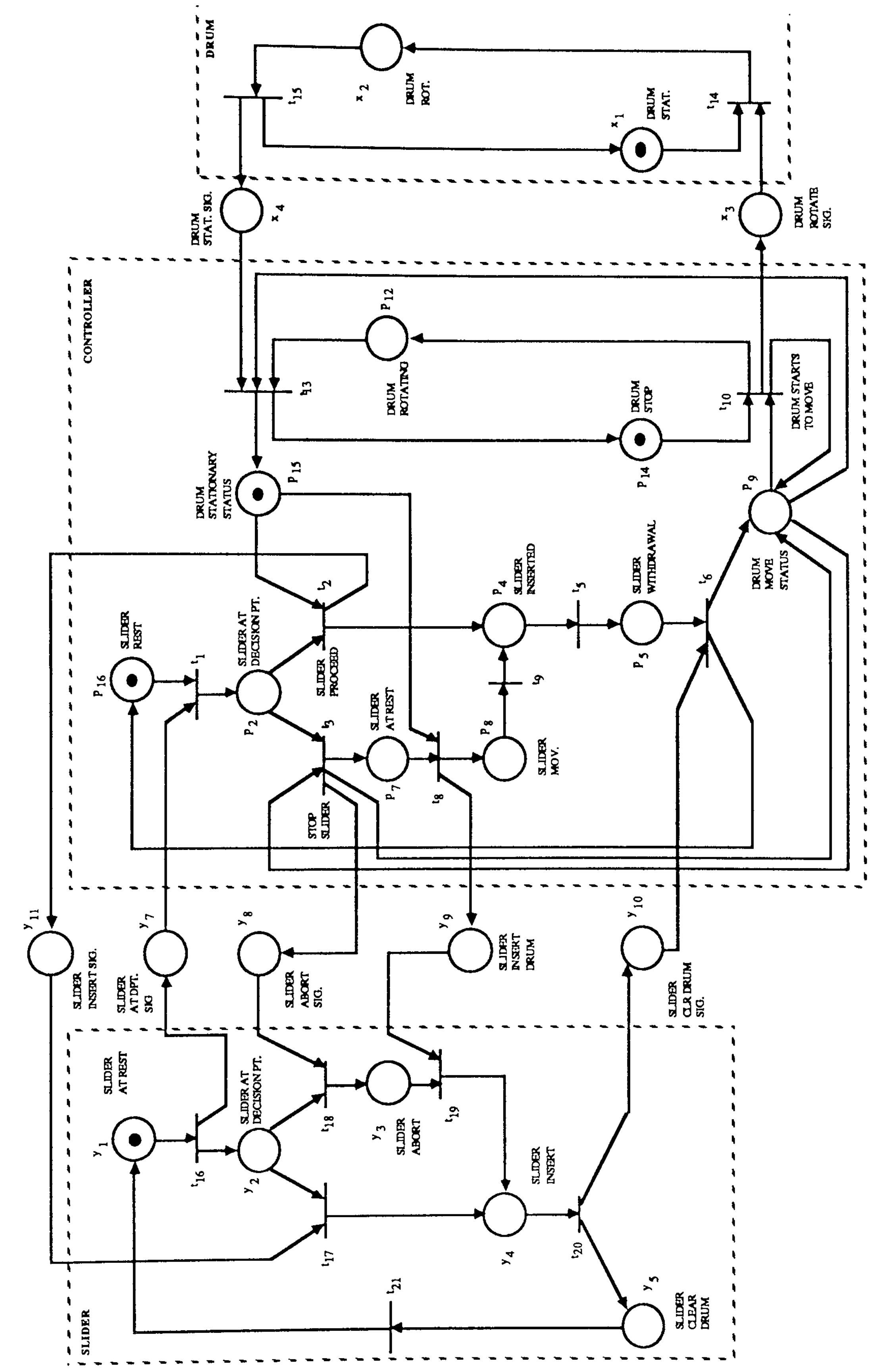
I and *O* are the input and output places of each transition in the net.

$M_0 = \{y1, x1, p14, p15, p16\}$ this can be formalised using propositional logic as $M_0{:}\ y1 \wedge x1 \wedge p14 \wedge p15 \wedge p16$

δe and δf are the enabling and firing times of a transition respectively, and are allocated to each transition in the net.

Tf_1 represents the temporal formulas (both qualitative and quantitative) concerning the sequence of enabling and firing of a transition

Figure 7.2 Controller design obtained using transition sampling



8.4.2.1 Proof of properties

An examination of the properties S1 - S4, L1 - L3 and RL1 - RL2 shows that the proof procedures for S1 and S2, S3 and S4, L1 - L3, RL1 and RL2 are the same, hence the following will illustrate the proof of a particular property from each group, ie S1, S3, L3 and RL1, and the proof of the remaining properties will be shown in Appendix E.

(i) Property S1

$$\Box\neg[y4 \wedge x2]$$

Proof

Assuming the negation of S1 yields:

1. $\neg\Box\neg[y4 \wedge x2]$

From the definition of $\neg\Box\neg w \equiv \Diamond w$, where w is any well-formed propositional formula, 1. can be expressed as:

2. $\Diamond[y4 \wedge x2]$

From M_0 applying each applicable system dependent inference rule it can be deduced that:

3. $\neg[y4 \wedge x2]$

From the temporal logic proof system the following theorem is valid:

4. $w \wedge \Diamond\neg w \Rightarrow \Diamond(w \wedge \mathrm{O}\neg w)$

The conjunction of 2. and 3. yields the following:

5. $\neg[y4 \wedge x2] \wedge \Diamond[y4 \wedge x2]$

By applying Modus ponens to 4. (where $\neg[y4 \wedge x2]$ is substituted for w) and 5., the following results:

6. $\Diamond\{\neg[y4 \wedge x2] \wedge \mathrm{O}[y4 \wedge x2]\}$

Expressing the first conjunct of 6. in terms of De Morgans theorem yields:

7. $\Diamond\{[\neg y4 \vee \neg x2] \wedge \mathrm{O}[y4 \wedge x2]\}$

Expanding 7. yields:

8. $\Diamond\{[\neg y4 \wedge \mathrm{O}(y4 \wedge x2)] \vee [\neg x2 \wedge \mathrm{O}(y4 \wedge x2)]\}$

Examining each disjunct of 8. in turn

9.	$[\neg y4 \wedge \mathrm{O}(y4 \wedge x2)]$	First disjunct of 8.
10.	$\neg y4 \wedge \mathrm{O}y4$	by weakening 9.
11.	$\mathrm{O}x2$	by weakening 9.

The only system dependent inference rule showing the sequence of 10. is:

$$y3 \wedge y9 \wedge \neg y4 \Rightarrow \mathrm{O}(\neg y3 \wedge \neg y9 \wedge y4)$$

The consequence of the above rule does not specify that position of the drum, hence the 'state' of the other processes in Figure 7.2 must be determined when y_4 is tokenised. This information can be easily inferred from the C_{min} of Table 7.2, by observing the concurrency set for y_4 which is:

$$C(y_4) = \{(p_4, p_{14}, x_1), (p_5, p_{14}, x_1), (p_8, p_{14}, x_1)\}$$

From $C(y_4)$ the set of all possible net markings is given by:

$$M_T = \{(p_4, p_{14}, x_1, y_4), (p_5, p_{14}, x_1, y_4), (p_8, p_{14}, x_1, y_4)\}$$

From M_T it can be inferred that the drum is at position x_1 when y_4 is tokenised. This contradicts 11. and thus falsifies 9.

Applying the same reasoning as 9. - 11. to the Second disjunct of 8. yields:

12.	$[\neg x2 \wedge O(y4 \wedge x2)]$	Second disjunct of 8.
13.	$\neg x2 \wedge Ox2$	by weakening 12.
14.	$Oy4$	by weakening 12.

The only system dependent inference rule that shows the sequence of 13. is:
$x3 \wedge x1 \wedge \neg x2 \;\Rightarrow\; O(\neg x3 \wedge \neg x1 \wedge x2)$
The 'state' of other processes in Figure 7.2 when x_2 is tokenised is given by:

$$C(x_2) = \{(p_{16}, p_9, p_{12}, y_5), (p_{16}, p_9, p_{12}, y_1), (p_{16}, p_9, p_{12}, y_7, y_2), (p_2, p_9, p_{12}, y_2), (p_7, p_9, p_{12}, y_3), (p_7, p_9, p_{12}, y_8, y_2)\}$$

From this set it can be deduced that when x_2 is tokenised y_4 is not tokenised. This contradicts 14. and thus 12. is false. Hence, the above has proved 8. and 1. to be false, thus proving S1 to hold true.

(ii) Property S2

$$t_{x2}(c) \Rightarrow O[(t_{y2}(ok) \vee t_{y4}(ok)) \mathbf{P}\, t_{x2}(c)]$$

Proof

This proof is based on deriving the antecedent and consequence of the Equ. 5.34 introduced in Chapter 5 and is re-stated as:

$$(w1 \Rightarrow \neg w3) \wedge (w1 \wedge \neg w2 \Rightarrow Ow1) \Rightarrow (w1 \Rightarrow w2\mathbf{P}w3) \qquad 5.34$$

where w_1, w_2 and w_3 are any propositional formulas

Equ. 5.34 is interpreted as: at a particular instant, if w_1 is true then w_3 is false and if w_1 is true but w_2 is false then at the next instant w_1 is true, then it can be deduced that when w_1 is true then w_2 occurs before w_3.

From Figure 7.2 it can be seen that when t_{x2} completes firing, places x_4 and x_1 become tokenised. The global net markings associated with these places can be deduced from the C_{min} of Table 7.2., by examining the concurrency sets for x_4 and x_1. Thus $C(x_4, x_1)$ yields the following set of markings:

$$M_T = \{M_1, M_2, M_3, M_4, M_5, M_6\}$$

where

$M_1 = (p_9, p_{12}, p_{16}, x_1, x_4, y_5)$ $\quad$ $M_4 = (p_2, p_9, p_{12}, x_1, x_4, y_2)$

$M_2 = (p_9, p_{12}, p_{16}, x_1, x_4, y_1)$ $\quad$ $M_5 = (p_7, p_9, p_{12}, x_1, x_4, y_3)$

$M_3 = (p_9, p_{12}, p_{16}, x_1, x_4, y_2, y_7)$ $\quad$ $M_6 = (p_7, p_9, p_{12}, x_1, x_4, y_2, y_8)$

Using the simplified version of Equ. 5.25, the markings obtained when t_{x2} completes firing is:

1. $t_{x2}(c) \Rightarrow (M_1 \vee M_2 \vee M_3 \vee M_4 \vee M_5 \vee M_6)$

Making the above substitution for markings in 1. and simplifying the resulting expression:

2. $t_{x2}(c) \Rightarrow (p_9 \wedge p_{12} \wedge x_4)$

The enabling of t_{13} can be represented using Equ. 5.20 as:

3. $(p_9 \wedge p_{12} \wedge x_4) \equiv t_{13}(ok)$

When t_{13} completes firing tokens are deposited into places p_{14} and p_{15}. Hence, the enabling and eventual firing of t_{13} can be represented by a simplified form of Equ. 5.25 as:

4. $t_{13}(ok) \Rightarrow \Diamond(p_{14} \wedge p_{15})$

Simplifying 4. yields:

5. $t_{13}(ok) \Rightarrow \Diamond p_{15}$

From Cmin of Table 7.2 it can be deduced that if p_{15} becomes tokenised then p_9 cannot be tokenised as well, hence:

6. $p_{15} \Rightarrow \neg p_9$

Using a simplified form of 3. it can be shown that:

7. $t_{13}(ok) \Rightarrow p_9$

Using laws of propositional logic 7. can be expressed as:

8. $\neg p_9 \Rightarrow \neg t_{13}(ok)$

Combining 6. and 8. yields:

9. $p_{15} \Rightarrow \neg t_{13}(ok)$

Combining 2. and a form of 3. yields:

10. $t_{x2}(c) \Rightarrow t_{13}(ok)$

Rearranging 10. using the laws of propositional logic yields:

11. $\neg t_{13}(ok) \Rightarrow \neg t_{x2}(c)$

Combining 9. and 11. yields:

12. $p15 \Rightarrow \neg t_{x2}(c)$

From the markings of the C_{min} of Table 7.2 and Figure 7.2. it can be deduced that the following formula holds:

13. $p15 \wedge \neg(t_{y2}(ok) \vee t_{y4}(ok)) \Rightarrow \bigcirc p15$

Equs. 12. and 13. form the antecedent of Equ. 5.34. Hence, it can be concluded that:

14. $p15 \Rightarrow (t_{y2}(ok) \vee t_{y4}(ok))\ \mathbf{P}\ t_{x2}(c)$

Combining 2., 3. and 5. yields:

15. $t_{x2}(c) \Rightarrow \Diamond p15$

Combining 14. and 15. yields:

16. $t_{x2}(c) \Rightarrow \Diamond[(t_{y2}(ok) \vee t_{y4}(ok))\ \mathbf{P}\ t_{x2}(c)]$ ■

Proof of properties S3 and S4 is shown in Appendix E, from which it transpires that S3 can be satisfied by Figure 7.2, however, S4 cannot be satisfied by all marking sequences of Figure 7.2. Thus, S4 is not an invariant property of the system and a situation can occur in which the slider is at the decision point, but is unable to decide whether to insert or abort, which could cause safety implications. This result is consistent with the analysis performed using concurrency sets (Chapter 7).

(iii) Property L3

$$<M_0,\alpha> \models \Box[t_{y5}(c) \Rightarrow \Diamond(t_{y6}(ok) \wedge t_{x1}(ok))]$$

Proof

When t_{y5} completes firing a token is removed from y_4 and deposited into y_5 and y_{10}. The global markings of the Figure 7.2 when y_5 and y_{10} are tokenised can be easily deduced from the C_{min} of Table 7.2 as:

$$M_T = \{M_1, M_2, M_3\}$$

where

$M_1 = (p4, p14, x1, y5, y10)$

$M_2 = (p5, p14, x1, y5, y10)$

$M_3 = (p8, p14, x1, y5, y10)$

Hence, it can be deduced that:

1. $t_{y5}(c) \Rightarrow (M_1 \vee M_2 \vee M_3)$

In markings M_1 - M_3, it can be deduced that t_{y6} is firable, and the enabling of t_{y6} is:

2. $ty_6(ok) \equiv y_5$

The subsequent firable transitions in the markings of 1. must be considered. For example the firable transitions in M_3 are:

3. $M_3 \wedge t_9(ok) \Rightarrow O(M_1 \vee ty_6(c))$

The firable transitions under marking M_1:

4. $M_1 \wedge t_5(ok) \Rightarrow O(M_2 \vee ty_6(c))$

The firable transitions under marking M_2:

5. $M_2 \wedge t_6(ok) \Rightarrow O(M_4 \vee ty_6(c))$

In 5. $M_4 = (p14,p16,p9,x1,y5)$, and under this marking ty_6 and t_{10} are firable Hence the following can be written:

6. $M_4 \wedge t_{10}(ok) \Rightarrow O(M_5 \vee ty_6(c))$

In 6. $M_5 = (p12,p16,p9,x1,x3,y5)$, and under this marking ty_6 and tx_1 are firable Hence the following can be written:

7. $M_5 \Rightarrow (tx_1(ok) \wedge ty_6(ok))$

Hence combining 1. - 7. yields:

8. $<M_0,\alpha> \models \Box[ty_5(c) \Rightarrow \Diamond(ty_6(ok) \wedge tx_1(ok))]$ ■

A similar conclusion to 8. can be produced for M_1, by combining 4. - 7. and M_2, by combining 5. - 7. Hence the above shows that in any marking if the slider withdraws from the drum then the slider will reach home position and the drum will start rotating.

Proof of properties L1 and L2 are shown in the Appendix E.

(iv) Property RL1

$$(y1 \wedge (\tau = 0)) \Rightarrow \Diamond[(p15 \wedge p14 \wedge p2) \wedge (\delta_{\mathbf{x2}} \leq \tau \leq \delta_{\mathbf{x1}})]$$

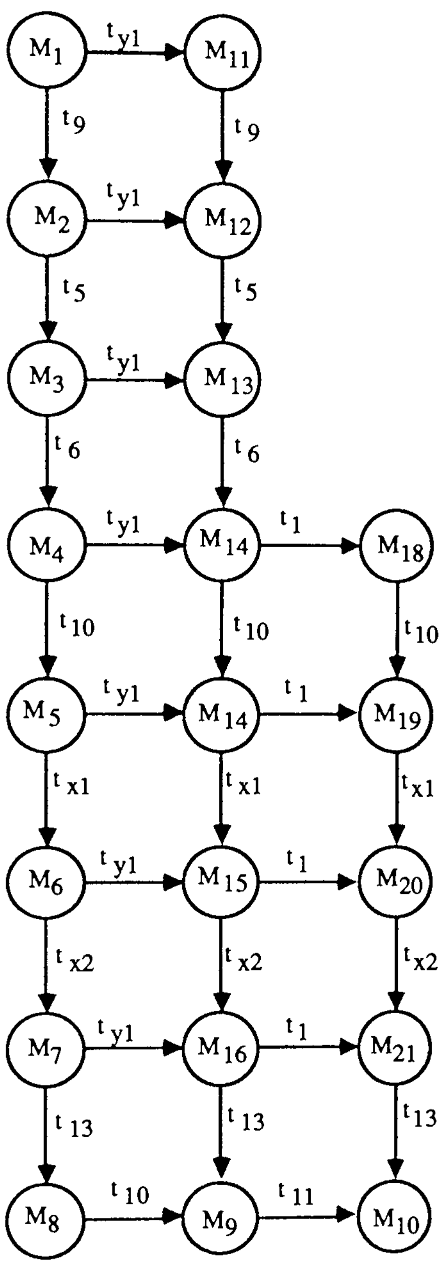
Proof

The proof of real-time properties requires identifying the partial reachability graph which starts from goal places (p15, p14, p2) and all paths are traced backwards until the origin place(s), ie y_1 is reached. This information can be easily obtained from the C_{min} of Table 7.2 by obtaining the concurrency set for place y_1 and from its markings generating all marking sequences that lead to (p15, p14, p2). The concurrency set for y_1 is:

$C(y_1) = \{(p14, p15, p16, x1), (p4, p14, y10, x1), (p5, p14, y10, x1),$
$(p8, p14, y10, x1), (p9, p14, p16, x1), (p9, p12, p16, x1, x3),$
$(p9, p12, p16, x2), (p9, p12, p16, x1, x4)\}$

From the markings arising from C(y_1) the partial reachability graph can be generated which is shown in Figure 8.1.

Figure 8.1 Partial reachability graph for the Petri net of Figure 7.2



MARKING	PLACES
M_1	$y_1, y_{10}, p_8, p_{14}, x_1$
M_2	$y_1, y_{10}, p_4, p_{14}, x_1$
M_3	$y_1, y_{10}, p_5, p_{14}, x_1$
M_4	$y_1, p_{16}, p_9, p_{14}, x_1$
M_5	$y_1, p_{16}, p_9, p_{12}, x_1, x_3$
M_6	$y_1, p_{16}, p_9, p_{12}, x_2$
M_7	$y_1, p_{16}, p_9, p_{12}, x_1, x_4$
M_8	$y_1, p_{16}, p_{14}, p_{12}, x_2$
M_9	$y_2, y_7, p_{16}, p_{14}, p_{15}, x_1$
M_{10}	$y_2, p_{14}, p_2, p_{15}, x_1$
M_{11}	$y_2, y_7, y_{10}, p_8, p_{14}, x_1$
M_{12}	$y_2, y_7, y_{10}, p_4, p_{14}, x_1$
M_{13}	$y_2, y_7, y_{10}, p_5, p_{14}, x_1$
M_{14}	$y_2, y_7, p_{16}, p_9, p_{14}, x_1$
M_{15}	$y_2, y_7, p_{16}, p_9, p_{12}, x_1, x_3$
M_{16}	$y_2, y_7, p_{16}, p_9, p_{12}, x_2$
M_{17}	$y_2, y_7, p_{16}, p_9, p_{12}, x_1, x_4$
M_{18}	$y_2, p_2, p_9, p_{14}, x_2$
M_{19}	$y_2, p_2, p_9, p_{12}, x_1, x_3$
M_{20}	$y_2, p_2, p_9, p_{12}, x_2$
M_{21}	$y_2, p_2, p_9, p_{12}, x_1, x_4$

From Figure 8.1 all marking sequences that start from the origin place (y_1) and lead to the goal places (p_{15}, p_{14}, p_2) are considered using Equ. 5.32, except for the marking sequence ($M_1, M_2, M_3, M_4, M_5, M_6, M_7, M_8$) in which the slider at rest, because RL1 is concerned with the instant at which the slider commences motion from rest. As an example of the proof method the following considers the sequence generated by the firing sequence (t_{y1}, t_9, t_5, t_6, t_{10}, t_{x1}, t_{x2}, t_{13}, t_1):

From markings M_1 - M_8 in Figure 8.1 it can be seen that when t_{y1} becomes firable and completes firing there are no conflicting transitions, hence the firing of t_{y1} can be expressed as:

1. $M_1 \wedge t_{y1}(ok) \wedge \tau=\mathcal{T}_1 \Rightarrow M_1 \ \mathcal{U}\, (M_{11} \wedge (\tau = \mathcal{T}_1 + \delta_{y1}))$

   where $\delta_{y1} = \delta e_{y1} + \delta f_{y1}$

Similarly, the firing of t_9:

2. $M_{11} \wedge t_9(ok) \wedge \tau=\mathcal{T}_2 \Rightarrow M_{11} \ \mathcal{U}\, (M_{12} \wedge (\tau = \mathcal{T}_2 + \delta_9))$

   where $\delta_9 = \delta e_9 + \delta f_9$ and $\mathcal{T}_2 = \mathcal{T}_1 + \delta_{y1}$

Firing of t_5:

3. $M_{12} \wedge t_5(ok) \wedge \tau=\mathcal{T}_3 \Rightarrow M_{12} \ \mathcal{U}\, (M_{13} \wedge (\tau = \mathcal{T}_3 + \delta_5))$

   where $\delta_5 = \delta e_5 + \delta f_5$ and $\mathcal{T}_3 = \mathcal{T}_2 + \delta_9$

Firing of t_6:

4. $M_{13} \wedge t_6(ok) \wedge \tau=\mathcal{T}_4 \Rightarrow M_{13} \ \mathcal{U}\, (M_{14} \wedge (\tau = \mathcal{T}_4 + \delta_6))$

   where $\delta_6 = \delta e_6 + \delta f_6$ and $\mathcal{T}_4 = \mathcal{T}_3 + \delta_5$

Firing of t_{10}:

5. $M_{14} \wedge t_{10}(ok) \wedge \tau=\mathcal{T}_5 \Rightarrow M_{14} \ \mathcal{U}\, (M_{15} \wedge (\tau = \mathcal{T}_5 + \delta_{10}))$

   where $\delta_{10} = \delta e_{10} + \delta f_{10}$ and $\mathcal{T}_5 = \mathcal{T}_4 + \delta_6$

Firing of t_{x1}:

6. $M_{15} \wedge t_{x1}(ok) \wedge \tau=\mathcal{T}_6 \Rightarrow M_{15} \ \mathcal{U}\, (M_{16} \wedge (\tau = \mathcal{T}_6 + \delta_{x1}))$

   where $\delta_{x1} = \delta e_{x1} + \delta f_{x1}$ and $\mathcal{T}_6 = \mathcal{T}_5 + \delta_{10}$

Firing of t_{x2}:

7. $M_{16} \wedge t_{x2}(ok) \wedge \tau=\mathcal{T}_7 \Rightarrow M_{16} \ \mathcal{U}\, (M_{17} \wedge (\tau = \mathcal{T}_7 + \delta_{x2}))$

   where $\delta_{x2} = \delta e_{x2} + \delta f_{x2}$ and $\mathcal{T}_7 = \mathcal{T}_6 + \delta_{x1}$

Firing of t_{13}:

8. $M_{17} \wedge t_{13}(ok) \wedge \tau=\mathcal{T}_8 \Rightarrow M_{17} \ \mathcal{U}\, (M_9 \wedge (\tau = \mathcal{T}_8 + \delta_{13}))$

   where $\delta_{13} = \delta e_{13} + \delta f_{13}$ and $\mathcal{T}_8 = \mathcal{T}_7 + \delta_{x2}$

Firing of t_1:

9. $M_9 \wedge t_1(ok) \wedge \tau=\mathcal{T}_9 \Rightarrow M_9 \ \mathcal{U}\, (M_{10} \wedge (\tau = \mathcal{T}_9 + \delta_1))$

   where $\delta_1 = \delta e_1 + \delta f_1$ and $\mathcal{T}_9 = \mathcal{T}_8 + \delta_{13}$

Combining 1. -9. yields the formula describing the firing sequence (t_{y1}, t_9, t_5, t_6, t_{10}, t_{x1}, t_{x2}, t_{13}, t_1):

10. $M_1 \wedge t_{y1}(ok) \wedge \tau=\mathcal{T}_1 \Rightarrow (M_1 \vee M_{11} \vee M_{12} \vee M_{13} \vee M_{14} \vee M_{15} \vee M_{16} \vee M_{17} \vee M_9) \ \mathcal{U}\, ((M_{10} \wedge (\tau = \mathcal{T}_1 + \delta_{T1})))$

    where $\delta_{T1} = \delta_9 + \delta_5 + \delta_6 + \delta_{10} + \delta_{x1} + \delta_{x2} + \delta_{13} + \delta_{y1} + \delta_1$

It can be concluded that considering other marking sequence of Figure 7.2 generated from marking M_1 to M_{10} as well as that of 10. yields:

11. $M_1 \wedge ty1(ok) \wedge \tau=\mathcal{T}_1 \Rightarrow M_T\ \mathcal{U}\ ((M_{10} \wedge (\tau = \mathcal{T}_1 + \delta_{T1})))$
    where M_T comprises of the disjunction of the markings
    $\{M_1,M_{10},M_{11},M_{12},M_{13},M_{14},M_{15},M_{16},M_{17},M_{18},M_{19},M_{20},M_{21}\}$

Similar to 11. markings sequences generated from M_2,M_3,M_4,M_5,M_6,M_7 or M_8 to M_{10}, are represented by the following respective formulas:

12. $M_2 \wedge ty1(ok) \wedge \tau=\mathcal{T}_1 \Rightarrow M_T\ \mathcal{U}\ ((M_{10} \wedge (\tau = \mathcal{T}_1 + \delta_{T2})))$
    where M_T comprises of the disjunction of the markings
    $\{M_2,M_9,M_{10},M_{12},M_{13},M_{14},M_{15},M_{16},M_{17},M_{18},M_{19},M_{20},M_{21}\}$
    $\delta_{T2} = \delta_{y1} + \delta_5 + \delta_6 + \delta_{10} + \delta_{x1} + \delta_{x2} + \delta_{13} + \delta_1$
13. $M_3 \wedge ty1(ok) \wedge \tau=\mathcal{T}_1 \Rightarrow M_T\ \mathcal{U}\ ((M_{10} \wedge (\tau = \mathcal{T}_1 + \delta_{T3})))$
    where M_T comprises of the disjunction of the markings
    $\{M_3,M_9,M_{10},M_{13},M_{14},M_{15},M_{16},M_{17},M_{18},M_{19},M_{20},M_{21}\}$
    $\delta_{T3} = \delta_{y1} + \delta_6 + \delta_{10} + \delta_{x1} + \delta_{x2} + \delta_{13} + \delta_1$
14. $M_4 \wedge ty1(ok) \wedge \tau=\mathcal{T}_1 \Rightarrow M_T\ \mathcal{U}\ ((M_{10} \wedge (\tau = \mathcal{T}_1 + \delta_{T4})))$
    where M_T comprises of the disjunction of the markings
    $\{M_4,M_9,M_{10},M_{14},M_{15},M_{16},M_{17},M_{18},M_{19},M_{20},M_{21}\}$
    $\delta_{T4} = \delta_{y1} + \delta_{10} + \delta_{x1} + \delta_{x2} + \delta_{13} + \delta_1$
15. $M_5 \wedge ty1(ok) \wedge \tau=\mathcal{T}_1 \Rightarrow M_T\ \mathcal{U}\ ((M_{10} \wedge (\tau = \mathcal{T}_1 + \delta_{T5})))$
    where M_T comprises of the disjunction of the markings
    $\{M_5,M_9,M_{10},M_{15},M_{16},M_{17},M_{19},M_{20},M_{21}\}$
    $\delta_{T5} = \delta_{y1} + \delta_{x1} + \delta_{x2} + \delta_{13} + \delta_1$
16. $M_6 \wedge ty1(ok) \wedge \tau=\mathcal{T}_1 \Rightarrow M_T\ \mathcal{U}\ ((M_{10} \wedge (\tau = \mathcal{T}_1 + \delta_{T6})))$
    where M_T comprises of the disjunction of the markings
    $\{M_6,M_9,M_{10},M_{16},M_{17}, M_{20},M_{21}\}$ and
    $\delta_{T6} = \delta_{y1} + \delta_{x2} + \delta_{13} + \delta_1$
17. $M_7 \wedge ty1(ok) \wedge \tau=\mathcal{T}_1 \Rightarrow M_T\ \mathcal{U}\ ((M_{10} \wedge (\tau = \mathcal{T}_1 + \delta_{T7})))$
    where M_T comprises of the disjunction of the markings $M_7,M_9,M_{10},M_{17},M_{21}$
    $\delta_{T7} = \delta_{y1} + \delta_{13} + \delta_1$
18. $M_8 \wedge ty1(ok) \wedge \tau=\mathcal{T}_1 \Rightarrow M_T\ \mathcal{U}\ ((M_{10} \wedge (\tau = \mathcal{T}_1 + \delta_{T8})))$
    where M_T comprises of the disjunction of the markings M_8,M_9,M_{10}
    $\delta_{T8} = \delta_{y1} + \delta_1$

Since any one of the firing sequences represented by 11. - 18. can occur, then combining 11. - 18. yields:

19. $M_1 \wedge ty1(ok) \wedge \tau=\mathcal{T}_1 \Rightarrow M_T\ \mathcal{U}\ ((M_{10} \wedge (\tau = \mathcal{T}_1 + \delta_{T1}))) \vee$
    $M_2 \wedge ty1(ok) \wedge \tau=\mathcal{T}_1 \Rightarrow M_T\ \mathcal{U}\ ((M_{10} \wedge (\tau = \mathcal{T}_1 + \delta_{T2}))) \vee$
    $M_3 \wedge ty1(ok) \wedge \tau=\mathcal{T}_1 \Rightarrow M_T\ \mathcal{U}\ ((M_{10} \wedge (\tau = \mathcal{T}_1 + \delta_{T3}))) \vee$

$M_4 \wedge \mathrm{ty1(ok)} \wedge \tau{=}\mathcal{T}_1 \Rightarrow M_T\ \mathcal{U}((M_{10} \wedge (\tau = \mathcal{T}_1 + \delta_{T4}))) \vee$
$M_5 \wedge \mathrm{ty1(ok)} \wedge \tau{=}\mathcal{T}_1 \Rightarrow M_T\ \mathcal{U}((M_{10} \wedge (\tau = \mathcal{T}_1 + \delta_{T5}))) \vee$
$M_6 \wedge \mathrm{ty1(ok)} \wedge \tau{=}\mathcal{T}_1 \Rightarrow M_T\ \mathcal{U}((M_{10} \wedge (\tau = \mathcal{T}_1 + \delta_{T6}))) \vee$
$M_7 \wedge \mathrm{ty1(ok)} \wedge \tau{=}\mathcal{T}_1 \Rightarrow M_T\ \mathcal{U}((M_{10} \wedge (\tau = \mathcal{T}_1 + \delta_{T7}))) \vee$
$M_8 \wedge \mathrm{ty1(ok)} \wedge \tau{=}\mathcal{T}_1 \Rightarrow M_T\ \mathcal{U}((M_{10} \wedge (\tau = \mathcal{T}_1 + \delta_{T8})))$

In all the expressions of 19. transition t_2 does not complete firing. Therefore substituting the place combinations for markings M_1, .. M_8 and simplifying the resulting expression yields:

20. $p1 \wedge \mathrm{ty1(ok)} \wedge \tau{=}\mathcal{T}_1 \Rightarrow \neg \mathrm{t2(c)}\ \mathcal{U}((M_{10} \wedge (\tau = \mathcal{T}_1 + \delta_{T1}))) \vee$
$p1 \wedge \mathrm{ty1(ok)} \wedge \tau{=}\mathcal{T}_1 \Rightarrow \neg \mathrm{t2(c)}\ \mathcal{U}((M_{10} \wedge (\tau = \mathcal{T}_1 + \delta_{T2}))) \vee$
$p1 \wedge \mathrm{ty1(ok)} \wedge \tau{=}\mathcal{T}_1 \Rightarrow \neg \mathrm{t2(c)}\ \mathcal{U}((M_{10} \wedge (\tau = \mathcal{T}_1 + \delta_{T3}))) \vee$
$p1 \wedge \mathrm{ty1(ok)} \wedge \tau{=}\mathcal{T}_1 \Rightarrow \neg \mathrm{t2(c)}\ \mathcal{U}((M_{10} \wedge (\tau = \mathcal{T}_1 + \delta_{T4}))) \vee$
$p1 \wedge \mathrm{ty1(ok)} \wedge \tau{=}\mathcal{T}_1 \Rightarrow \neg \mathrm{t2(c)}\ \mathcal{U}((M_{10} \wedge (\tau = \mathcal{T}_1 + \delta_{T5}))) \vee$
$p1 \wedge \mathrm{ty1(ok)} \wedge \tau{=}\mathcal{T}_1 \Rightarrow \neg \mathrm{t2(c)}\ \mathcal{U}((M_{10} \wedge (\tau = \mathcal{T}_1 + \delta_{T6}))) \vee$
$p1 \wedge \mathrm{ty1(ok)} \wedge \tau{=}\mathcal{T}_1 \Rightarrow \neg \mathrm{t2(c)}\ \mathcal{U}((M_{10} \wedge (\tau = \mathcal{T}_1 + \delta_{T7}))) \vee$
$p1 \wedge \mathrm{ty1(ok)} \wedge \tau{=}\mathcal{T}_1 \Rightarrow \neg \mathrm{t2(c)}\ \mathcal{U}((M_{10} \wedge (\tau = \mathcal{T}_1 + \delta_{T8})))$

Further simplifying 20. yields:

21. $p1 \wedge \mathrm{ty1(ok)} \wedge \tau{=}\mathcal{T}_1 \Rightarrow \neg \mathrm{t2(c)}\ \mathcal{U}((M_{10} \wedge (\mathcal{T}_1 + \delta_{T8}) \leq \tau \leq (\mathcal{T}_1 + \delta_{T1})))$

If the drum is stationary when the slider starts to move then the slider will definitely insert into the drum without resorting to the abort motion and the timing constraints are defined from 20. by the formula, assuming slider starts to move at $\tau = 0$:

22. $p1 \wedge \mathrm{ty1(ok)} \wedge \tau{=}\mathcal{T}_1 \Rightarrow \neg \mathrm{t2(c)}\ \mathcal{U}((M_{10} \wedge (\tau{=}\mathcal{T}_1 + \delta_{T8})$

Simplifying 21. yields:

23. $p1 \wedge \mathrm{ty1(ok)} \wedge \tau{=}\mathcal{T}_1 \Rightarrow \Diamond(M_{10} \wedge (\tau{=}\mathcal{T}_1 + \delta_{T8})$

Substituting M_{10}: $y2 \wedge p2 \wedge p14 \wedge p15 \wedge x1$ into 22. and simplifying the resulting expression yields:

24. $p1 \wedge \mathrm{ty1(ok)} \wedge \tau{=}\mathcal{T}_1 \Rightarrow \Diamond((p2 \wedge p14 \wedge p15) \wedge (\tau{=}\mathcal{T}_1 + \delta_{T8}))$

However, if the drum is rotating or rotate enabled when the slider starts to move then timing constraints must be satisfied in order to prevent the abort motion of the slider. Thus, if the drum is rotate enabled when the slider starts to move then the timing constraints are given by:

25. $p1 \wedge \mathrm{ty1(ok)} \wedge \tau{=}\mathcal{T}_1 \Rightarrow \neg \mathrm{t2(c)}\ \mathcal{U}(M_{10} \wedge (\tau = \delta_{T4}))$

Moreover, if the drum is rotating when the slider starts to move then the timing constraints are given by:

26. $p1 \wedge \mathrm{ty1(ok)} \wedge \tau{=}\mathcal{T}_1 \Rightarrow \neg \mathrm{t2(c)}\ \mathcal{U}(M_{10} \wedge (\tau = \delta_{T6}))$

Hence, the timing constraints that must be satisfied by the slider if the drum is rotate enabled or rotating is given by combining 24. and 25.:

27. $pl \wedge \mathrm{ty1(ok)} \wedge \tau=\mathcal{T}_1 \Rightarrow \neg t2(c)\ \mathcal{U}\,(M10 \wedge (\delta_{T6} \leq \tau \leq \delta_{T4}))$

From the above it is seen that the timing constraints are dependent upon the position of the drum when the slider starts to move. However, if the slider starts to move irrespective of the drum's motion then the maximum and minimum timing constraints are given by 27. and 22.:

27. $pl \wedge \mathrm{ty1(ok)} \wedge \tau=\mathcal{T}_1 \Rightarrow \neg t2(c)\ \mathcal{U}\,(M10 \wedge (\delta_{T8} \leq \tau \leq \delta_{T1}))$

Performing simplifications on 27. yields:

28. $pl \wedge \mathrm{ty1(ok)} \wedge \tau=\mathcal{T}_1 \Rightarrow \Diamond((p2 \wedge p14 \wedge p15) \wedge (\delta_{T8} \leq \tau \leq \delta_{T1}))$ ■

An inspection of the timing constraints of 28. shows that:

$$\delta_{T8} = \delta_{y1} + \delta_1$$

$$\delta_{T1} = \delta_{y1} + \delta_{x1} + \delta_{x2} + \delta_1 + \delta_5 + \delta_6 + \delta_9 + \delta_{10} + \delta_{13}$$

Relating these timing constraints to the motion of the drum and slider shows that, the period in which the slider remains at rest, ie δ_{ey1} where $\delta_{y1} = \delta_{ey1} + \delta_{fy1}$, may not be determinable, since the slider could stay at rest for time. However, the controller must respond within a time constraint, hence $\{\delta_1, \delta_5, \delta_6, \delta_9, \delta_{10}, \delta_{13}\}$ is determinable, moreover the drum rotation time (δ_{x2}) and rotate enable time (δ_{x1}) are also determinable. Thus the worst case maximum timing constraints needed to satisfy RL1 must be $\{\delta_{x1} + \delta_{x2} + \delta_1 + \delta_5 + \delta_6 + \delta_9 + \delta_{10} + \delta_{13}\}$. Similarly, the worst case minimum timing constraints must be $\{\delta_1\}$.

Proof of property RL2 is shown in Appendix E.

8.5 Conclusions

This chapter has investigated the possibility of unifying the techniques of temporal Petri nets, extended temporal Petri nets and real-time temporal Petri nets. It has shown that the enabling and firing rules of a transition in Petri net theory, which formed the basis of real-time temporal Petri nets, are generic formulas that can be used to derive the basic formulas of temporal Petri nets and extended temporal Petri nets. Although, temporal Petri nets and extended temporal Petri nets formalise these rules, they do so using either an event-based or state-based logic (see Section 3.2). This feature makes these techniques amenable to a particular type of formal analysis. The strength of real-time temporal Petri nets is that it uses a mixture of event-based and state-based logics to assert formulas concerning the features of the Petri net. This allows it to derive the fundamental formulas of temporal Petri nets and extended temporal Petri nets.

The formal analysis of the controller/physical system Petri net using real-time temporal Petri nets confirmed the analysis that was performed using concurrency sets in Chapter 7. Moreover, the use of C_{min} in conjunction with the formal analysis of real-time temporal Petri nets improved the efficiency of the proof procedures, by allowing the partial reachability graph of the net to be easily generated and the net markings to be inspected. Although the C_{min} provides useful information concerning the net, the complete set was not necessary in the formal analysis because the proof procedures indicated which concurency sets were required.

Chapter 9

Conclusions

9.1 Conclusions

This thesis has been primarily concerned with the development of techniques that combine both a formal and informal formalism for the specification and verification of hard real-time systems. It has also been concerned with the synthesis and design of controllers for these systems. The formal techniques developed are centred on the combined formalisms of Petri nets (which provide an informal graphical notation) and temporal logic (which provides the formal mathematical component). The synthesis of real-time controllers was achieved by proposing a strategy which essentially involved model-based control [Ramadge 87]. This produced controller designs which contained a model of the physical system and the chief concern was to synchronize it with the physical system.

For a formal method to be applicable to hard real-time systems it must be capable of describing features such as the functionality of the system, concurrency and timing constraints. This means that such techniques must specify the performance requirements as well as functional requirements and provide means of verifying these systems. Chapter 2 presented a detailed survey of the formal techniques including Petri nets, temporal logic, CSP, VDM, Z and PAISLey, and assessed their suitability for specifying and verifying hard real-time systems; various forms of this survey have been presented in [Sagoo 90], [Sagoo 92a]. Since formal methods are based on abstract mathematical concepts, they tend to be difficult to learn and require a high level of mathematical skill to apply. This aspect of formal methods has most hindered their use in industry. Although the mathematical component of these methods cannot be removed, formal methods must be made more tractable. This issue has been addressed in this thesis with the combination of formal and informal techniques (Petri nets and temporal logic) into a unified formalism.

Previous approaches, surveyed in Chapter 3, [Sagoo 91], [Sagoo 92a], [Holding 92], combined Petri nets and temporal logic and used them to analyse formally concurrent systems. From these approaches the temporal Petri nets of Suzuki and the technique proposed by He and Lee were considered to be usable because they combined Petri nets and temporal logic in a consistent manner; a specification written in these techniques is consistent in each constituent formalism. Temporal Petri nets are centred on an event-based logic and are useful for verifying liveness properties. However, a possible weakness of

this technique lies in its verification procedure. Specifically, this procedure requires propositions to be constructed that formalise the firing of transitions encountered within the particular liveness property being proved. Since transitions can fire under various conditions, many types of propositions may need to be constructed that are unique to a particular net. Hence, in order to perform temporal Petri net analysis a different set of propositions may be required for each type of net. As a formal method the main disadvantage of temporal Petri nets is that they cannot prove safety properties; these properties can only be inferred from the net's reachability graph. Since the reachability graph can become large for relatively small nets, this imposes a further limitation on the size of net that can be analysed for safety properties.
The technique of He and Lee, which combined high-level Petri nets and first order temporal logic, was suited to the formal analysis of large concurrent systems. In Chapter 3 this approach was adapted to allow the formal analysis of systems modelled using low-level Petri nets; the resulting technique was referred to as extended temporal Petri nets. This adaptation was performed because:

(i) He and Lee's technique can produce models in which the essential functionality of the system is 'hidden' in abstract detail,
(ii) low-level Petri nets have been extended with quantitative time and thus they are suited to the modelling of hard real-time systems. However, high-level Petri nets have not been extended in this manner, instead they have only been combined with qualitative time via first order temporal logic. Hence high-level Petri nets are not suited to the modelling hard real-time systems.

Extended temporal Petri nets are centred on a state-based logic that translates Petri nets into temporal logic. This technique allows the formal specification and verification of safety and liveness properties. Thus in the context of formal methods extended temporal Petri nets facilitate a better formal analysis than temporal Petri nets. However, weaknesses in extended temporal Petri nets can arise e.g:

(i) when dealing with relatively large Petri net models. In this case the translation of Petri nets into temporal logic can become unacceptably large and unmanageable. This is mainly because the firing rules for each transition in the former need to be expressed in terms of the latter,
(ii) in the procedure for proving liveness properties. Currently this procedure relies on considering the firing of every transition in the net; this can lead to a proof that is both tedious and lengthy. Hence a more abstract procedure needs to be developed.

Although temporal Petri nets and extended temporal Petri nets both allow the formal analysis of concurrent systems, each technique has particular attributes. For instance, a comparison of these techniques shows that the former is most suited to the verification of liveness properties, whereas the latter is the only technique capable of verifying safety properties.

The application of temporal Petri nets, extended temporal Petri nets and timed Petri nets (developed by Ramachandni, and Razouk and Phelps) to a hard real-time control problem was considered, in order to reveal the attributes and limitations of the analysis performed by each technique. This was demonstrated in Chapter 4, [Sagoo 90], [Sagoo 92a], [Holding 92] by the modelling of a hard real-time control problem which forms part of a high-speed packaging machine and concerns the coordination of two mechanisms, the drum and slider. The application of these techniques showed that timed Petri nets, are suitable for the detection of static hazardous states and the presence of potentially hazardous time-critical sequences. Timing analysis can be used to prevent their occurrence (thus ensuring the safety of the system being modelled) and to eliminate parts of the reachability graph that correspond to certain types of behaviours. However, timed Petri nets do not allow the formal specification and verification of the modelled system. Temporal Petri nets provide a notation that allows properties such as fairness and eventuality to be specified, a feature which cannot be expressed using Petri nets or timed Petri nets. These nets allow the formal verification of liveness properties which specify the eventual occurrence of a state or event using qualitative time. Similarly, extended temporal Petri nets allow the formal verification of both safety and liveness properties by using a proof procedure that attempts to prove the desired property does not occur. The results discussed above suggests that each technique could be applied to a Petri net model in a hierarchical manner to obtain various 'levels' of analysis. For instance, timed Petri nets can be used to perform preliminary analysis on a net to ascertain whether it exhibits the desired behaviour. For a more detailed formal analysis temporal Petri nets and extended temporal Petri nets can be used. Specifically, the former can be used to prove liveness properties and the latter for proving safety properties.

Temporal Petri nets and extended temporal Petri nets can be used to prove liveness properties which are based on qualitative time, but, they cannot be used to prove real-time properties involving quantitative time. Although timed Petri nets use the notion of quantitative time to evaluate the real-time performance and ensure the safety aspects of hard real-time systems, they cannot be used for formal specification and verification. Hence, the main limitation of the above Petri net based techniques is that they cannot be used to verify real-time properties. This problem was addressed in Chapter 5 by proposing a technique which combines the timed Petri net of Razouk and Phelps, with real-time temporal logic. This technique, referred to as real-time temporal Petri nets (RTTNs), was formally defined and applied to the Petri net model of the drum and slider system developed in Chapter 4. The main strength of this technique lies in its ability to perform the analysis of timed, temporal and extended temporal Petri nets within a single formalism. Specifically, the timed Petri net component of RTTN can perform timing analysis and its real-time temporal logic component can specify and verify both non real-time and real-time properties. Hence this technique removes the need to apply a set of techniques to analyse particular aspects of the modelled system.

A common feature of the analysis procedure of the Petri net based techniques, examined in this thesis, is that they require information from the reachability graph. However, the generation of the reachability graph becomes progressively difficult as the size of the net increases. Hence, this imposes a limitation on the size of net that can be analysed using these techniques. Chapter 6 addresses the issue of reducing the complexity in generating the reachability graph by proposing a technique that allows the generation of a partial reachability graph associated with a particular system state. This is achieved by using the notion of a concurrency set which was developed by Skeen and Stonebraker. Application of this set to a Petri net model showed that a concurrency set for each place in the net could be generated. Moreover, an inspection of the set of all concurrency sets for a net showed that it could be reduced to a minimum set of concurrency sets. The main attribute of this minimum concurrency set was that it contained all the unique markings of the net. Thus inspection of the states represented by this set provided an efficient means of detecting the presence of hazardous and inconsistent markings. Moreover, when this set is used in conjunction with the Petri net it allows a partial reachability graph to be easily generated. Since this set can generate partial reachability graphs, it would be useful in the verification procedures used in temporal and extended temporal Petri nets, and RTTNs. This is because the verification of safety and liveness properties using these techniques relies on identifying a particular partial reachability graph; this was illustrated in Chapter 8. However, the main weakness of the minimum concurrency set is that it is currently deduced by inspection of the net. Hence as the size of net increases, it becomes difficult to obtain this set.

The Petri net models of the hard real-time control problem, which involved the coordination of the drum and slider, were found to be inadequate for describing an implementable controller for this system. Hence, Chapter 7 and [Sagoo 92b] addressed this issue by proposing a strategy for designing such a controller. This strategy used the Petri net models of the drum and slider developed in Chapter 4 by embedding them as the controller

into the net comprising the physical system. This approach was used so that the controller would inherit the properties proved by previous analysis. This type of controller, containing a model of the physical system, needed to be synchronised to the physical system in order to provide a timely response and coordinate the mechanisms of the physical system in a safe manner. This problem was considered in detail in Chapter 7. Specifically, two methods of interfacing and synchronizing the controller and physical system were developed, and they were referred to as transition sampling and place sampling. Using these methods three categories of controller/physical system Petri nets were produced: nets obtained using transition sampling, place sampling and hybrid sampling (which uses a combination of transition sampling and place sampling). Analysis of these nets using concurrency sets showed that transition sampling produced the safest controller design, whereas hybrid sampling tended to produce an unsafe controller.
The controller produced using the above strategy had the main advantage of inheriting the desirable features of and the analysis performed on previous models. However, this approach had the disadvantage that the undesirable features of previous models were also inherited, as shown throughout Chapter 7.

A comparison of the Petri net based techniques examined in this thesis shows that the real-time temporal Petri net is the most suited for modelling, specifying and verifying hard real-time systems. This is because it allows both real-time and non real-time aspects of the system to be captured. Moreover, the use of concurrency sets in conjunction with this technique produces an efficient means of verifying system properties. Although this thesis has also investigated approaches for synthesizing an embedded controller for hard real-time systems, this work produced only preliminary results and further work is needed to assess its usefulness.

9.2 Further work

There are two directions of research which can be pursued from the work developed in this thesis. Firstly, the Petri net based techniques examined in this thesis could be further developed by:

(i) automating the proof procedures using a software verification tool,
(ii) enhancing these techniques so that they are applicable to large real-time systems.

The proofs presented in this thesis were done manually and whilst constructing these proofs it was noticed that at least some parts of the proof could be automated. This stemmed from that fact that all proofs are based on using information derived from a partial reachability graph; some proofs are directly derived from the appropriate partial reachability graph. Therefore, an approach which classifies the properties of the system that can be directly inferred from the reachability graph and uses this information in the development of a software verification tool would be beneficial. Moreover, this thesis has shown that concurrency sets can be used for generating a partial reachability graph as well as performing state consistency checks. Hence, a software tool that is based on generating concurrency sets would have the dual function of providing information useful for the proof procedures and for inspecting the semantics of the reachable states.

Although the hard real-time system modelled in this thesis could be adequately modelled using low-level Petri nets and the Petri net based techniques examined in this thesis are suited to analysing low-level Petri nets, the need to model larger systems is also an important issue. For larger systems low-level Petri nets can produce large Petri net models that can be difficult to understand. This problem can be alleviated to some extent by the use of high-level Petri nets. However, since these nets do not incorporate the notion of time, they are unsuitable for modelling hard real-time systems. This problem can be overcome by using the approach which translates a high-level Petri net into low-level Petri net in order to use timed Petri net analysis, but, again, it is limited because of the complexity of the resulting net. Hence future work must address the issue of extending high-level Petri nets with time and then considering the feasibility of combining it with real-time temporal logic to allow for formal specification and verification.

Further work must also make progress in developing a methodology for the design of real-time controllers. This methodology should contain a systematic procedure for deriving a controller from the specification of the system and techniques for analysing the resulting controller. The work presented in this thesis could form the basic foundations for this methodology. For instance, the strategy proposed in Chapter 7 allows the synthesis of an embedded controller and the techniques shown in Chapters 3 - 5 and 8 could be used as the analysis methods. However, this work must be elaborated in areas such as providing a detailed method of forming the interface between the controller and the physical system and for analysing the controller in the presence of failures.

References

[Agerwala 79] T. Agerwala, "Putting Petri nets to work", Computer, Vol. 12, No. 12, 1979, pp 85 - 94.

[Alpern 85] B. Alpern and F.B. Schneider, "Defining liveness", Information Processing Letters, Vol. 21, No. 4, 1985, pp 181 - 185.

[Anttila 83] M. Anttila, H. Erikson, H., and J. Ikonen, "Tools and studies of formal techniques-Petri nets and temporal logic", In H. Rudin and C.H. West (Eds), 'Protocol Specification Testing and Verification', III, Elsevier Science Pub. B.V., North Holland, 1983, pp 139 - 148.

[Audsley 92] N.C. Audsley, A. Burns, M.F. Richardson and A.J. Wellings, "Deadline monotonic scheduling theory", In L. Boullart and J.A. de la Puente (Eds), Int. Workshop on 'Real-time Programming WRTP'92', 23 - 26 June 1992, pp 55 - 60.

[Barringer 84] H. Barringer, R. Kuiper and A. Pnueli, "Now you may compose temporal logic specifications," Proc. of 16th ACM Symposium on 'Theory of Computing', 1984, pp 51 - 63.

[Ben-Ari 83] M. Ben-Ari, A. Pnueli and Z. Manna, "The temporal logic of branching time", ACTA Informatica, Vol. 20, No. 3, 1983, pp 207 - 226.

[Bennett 84] S. Bennett, "Construction of real-time software", In 'Real-time Computer Control', IEE Control Engineering series 24, Pub: Peter Peregrinus Ltd, 1984, pp 87 - 99.

[Bernstein 87] A. Bernstein and P.K. Harter, "Proving real-time properties of programs with temporal logic", Proc. of 8th Symposium on 'Operating Systems Principles', ACM SIGOPS, 1987, pp 1 - 11.

[Berry 82] D.M. Berry, C. Ghezzi, D. Mandrioli and F. Tisato, "Language constructs for real-time distributed systems", Computer Languages, Vol. 7, No. 1, 1982, pp 11 - 29.

[Berthelot 82] G. Berthelot and R. Terrat, "Petri nets for the correctness of protocols", In R.P. van de Riet and W. Litwin (Eds), 'Distributed Data Sharing Systems', North-Holland Pub. Co., 1982, pp 23 - 43.

[Berthomieu 83] B. Berthomieu and M. Menasche, "An enumerative approach for analyzing time Petri nets", In R.E.A. Mason (Ed), Proc. of IFIP Congress, 'Information Processing 83', Paris, 1983, pp 41 - 46.

[Blazewicz 76] J. Blazewicz, "Scheduling dependent tasks with different arrival times to meet deadlines", Proc. of Int. workshop on 'Modelling and Performance Evaluation of Computer Systems', 1976, pp 57 - 65.

[Blyth 90] D. Blyth, C. Boldyreff, C. Ruggles, and N. Tetteh-Lartey, "The case for formal methods in standards", IEEE Software, Vol. 7, No. 5, 1990, pp 65 - 67.

[Burns 87] A. Burns and A.J. Wellings, "Real-time ADA issues", ADA Letters, Vol. 6, No. 6, 1987 pp 43 - 46.

[Burns 90] A. Burns and A. Wellings, "Real-time systems and their programming languages", Wokingham, Addison-Wesley, 1990.

[Carlow 84] G.D. Carlow, "Architecture of the space shuttle primary avionics software systems", Comm. ACM, Vol. 27, No. 9, 1984, pp 926 -936.

[Chellas 80] B.F. Chellas, 'Modal logic: an introduction', Cambridge University Press, 1980.

[Clarke 86] E.M. Clarke, E.A. Emerson and A.P. Sistla, "Automatic verification of finite-state concurrent systems using temporal logic specifications", ACM Trans. Prog. Lang. and Systems, Vol. 8, No. 2, 1986, pp 244 - 263.

[Dahler 87] J. Dahler, P Gerber, H.P. Gisiger and A Kundig, "A graphical tool for the design and prototyping of distributed systems", ACM Software Eng. Notes, Vol. 12, No. 3, 1987, pp 25 - 36.

[Dasarathy 85] B. Dasarathy, "Timing constraints of real-time systems, constructs for expressing them, methods of validating them", IEEE Trans. Software Eng., Vol. SE-11, No. 1, 1985, pp 80 - 86.

[David 91] R. David, "Modelling of dynamic systems by Petri nets", Proc. of European Control Conference, Grenoble, France, 2 - 5 July 1991, pp 136 -147.

[Davies 89] J. Davies and S. Schneider, 'An Introduction to Timed CSP', PRG Technical Monograph, No. 75, Oxford University, 1989.

[Davis 88] A.M. Davis, "A comparison of techniques for the specification of external system behaviour", Comm. ACM, Vol. 31, No. 9, 1988, pp 200 - 217.

[Dhall 78] S.K. Dhall and C.L. Liu, "On a real-time scheduling problem", Operations Research, Vol. 26, No. 1, 1978, pp 127 - 140.

[Diaz 83] M. Diaz and G. Guidacci Da Silverira, "Specification and validation of protocols by temporal logic", In R.E.A. Mason (Ed), Proc. IFIP Congress, 'Information Processing 83', Elsevier Science Pub. B.V., North Holland, Sept. 1983, pp 47 - 52.

[Dijkstra 76] E.W. Dijkstra, 'A discipline of programming', Prentice-Hall, Englewood Cliffs, 1976.

[DOD 83] "ADA programming language", (ANSI / MIL - STD - 1815A), Washington D.C., 20301: ADA joint program office, DOD, January 1983.

[Emerson 82] E.A. Emerson and E. M. Clarke, "Using branching time temporal logic to synthesize synchronization skeletons," Science of Computer Programming, Vol. 2, 1982, pp 241 - 266.

[Emerson 86] E.A. Emerson and J.Y. Hailpern, "Sometime and not never revisited: on branching versus linear time temporal logic", Journal of ACM, Vol. 33, No. 1, 1986, pp 515 - 178.

[Fenney 88] L. Fenney, C.M. Draper, K. Foster and D.J. Holding, "Modular machine systems", Proc. of IMechE/SERC Conf. 'High Speed Machinery', I. Mech. E., London, Nov. 1988, pp 19 - 27.

[Francez 86] N. Francez, 'Fairness', Springer-Verlag, New York, 1986.

[Fraser 91] M.D. Fraser, K. Kumar and V.K. Vaishnavi, "Informal and formal requirements specification languages: bridging the gap", IEEE Trans. Software Eng., Vol. 17, No. 5, 1991, pp 454 - 466.

[Froome 88] P. Froome and B. Monahan, "The role of mathematically formal methods in the development and assessment of safet-critical systems", Microprocessors and Microsystems, Vol. 12, No. 10, 1988, pp 539 - 553.

[Gabbay 80] D. Gabbay, A. Pnueli, S. Shelah and J. Stavi, "On the temporal analysis of fairnes", Proc. on 7th ACM Annual Symposium on 'Principles of Programming Languages', Jan 1980, Las Vegas, N.Y., pp 163 - 173.

[Gabrielian 88] A. Gabrielian and M.K. Franklin, "State-based specification of complex real-time systems", In Proc. of IEEE Symposium on 'Real-time systems', Dec. 1988, pp 2 - 11.

[Garey 77] M.R. Garey and D.S. Johnson, "Two-processor scheduling with start-times and deadlines", Journal SIAM on Computing, Vol. 6, 1977, pp 416 - 426.

[Garman 81] J.R. Garman, "The bug heard round the world", Software Eng. Notes, Vol. 6, No. 5, October 1981, pp 3 - 10.

[Genrich 79] H.J. Genrich and K. Lautenbach, "The analysis of distributed systems by means of Predicate/Transition nets", In G. Kahn (Ed), 'Semantics of Concurrent Computation', Lecture Notes in Computer Science, Vol. 70, Springer-Verlag, 1979, pp 123 - 146.

[Genrich 80] H.J. Genrich, K. Lautenbach and P.S. Thiagarajan, "Elements of general net theory", In W. Brauer, (Ed.), 'Net Theory and Applications', Lecture Notes in Computer Science, Vol. 84, Springer-Verlag, 1980, pp 21 - 164.

[Genrich 81] H.J. Genrich and K. Lautenbach, "System modelling with high-level Petri nets", Theoretical Computer Science, Vol. 13, 1981, pp 109 - 136.

[Genrich 87] H.J. Genrich, "Predicate/transition nets", In W. Brauer, W. Reisig and G. Rozenburg (Eds), 'Petri nets: Central Models and their Properties', Lecture Notes in Computer Science, Vol. 254, 1987, pp 207 - 247.

[Ghezzi 91] C. Ghezzi, D. Mandrioli, S. Morasca and M. Pezze, "A Unified High-Level Petri Net Formalism for Time-Critical Systems", IEEE Trans. Software Eng., Vol. 17, No. 2, 1991, pp 160 - 172.

[Gligor 83] V.D. Gligor and G.L. Luckenbaugh, "Assessment of the real-time requirement for programming environments and languages", Proc. of IEEE Symposium on 'Real-time systems', 1983, pp 3 - 19.

[Gorski 88] J. Gorski, "Formal specification of real-time systems", Computer Physics Comm., Vol. 60, 1988, pp 71 - 88.

[Graham 79] R.L. Graham, "Optimisation and approximation in deterministic sequencing and scheduling: a survey", Annals of Discrete Mathematics, Vol. 5, 1979, pp 287 - 326.

[Hall 90] A. Hall, "Seven myths of formal methods", IEEE Software, Vol. 7, No. 5, 1990, pp 11 - 19.

[Harel 87] D. Harel, "State charts: a visual formulism for complex systems", Science of Computer Programming, Vol. 8, No. 3, 1987, pp 231 - 274.

[Harel 90] D. Harel, H. Lachover, A. Naamad, A. Pnueli, M. Politi, R. Sherman, A. Shtull-Trauring, and M. Trakhtenbrot, "STATEMATE: A working environment for the development of complex reactive systems", IEEE Trans. Software Eng., Vol. 16, No. 4, April 1990, pp 403 - 414.

[Hatley 87] D.J. Hatley and I.A. Pirbhai, 'Strategies for real-time system specifications', Dorset House Publishing Company, New York, 1987.

[He 90] X. He, and J.A.N. Lee, "Integrating Predicate Transition nets with first order temporal logic in the specification and verification of concurrent systems", Formal Aspects of Computing, Vol. 2, 1990, pp 226 - 246.

[He 91] X. He and J.A.N. Lee, "A methodology for constructing Predicate Transition Net specifications", Software Practice and Experience, Vol. 21, No. 8, August 1991, pp 845 - 875.

[Hill 90] M.R. Hill and D.J. Holding, "The modelling, simulation, and analysis of commit protocols in distributed computing systems", Proc. UKSC Conf. on 'Computer Simulation', Sept. 1990, Brighton, pp 207 - 212.

[Hoare 69] C.A.R. Hoare, "An axiomatic basis for computer programming", Comm. ACM, Vol. 12, No. 10, October 1969, pp 576 - 580.

[Hoare 85] C.A.R. Hoare, "Communicating sequential processes", Prentice-Hall Int., 1985.

[Holding 92] D.J. Holding and J.S. Sagoo, "A formal approach to the software control of high-speed machinery", In G.W. Irwin and P.J. Fleming (Eds), 'Transputers for control', Research Studies Press, 1992, Chapter 9, pp 239 - 283.

[Hoogeboom 91] B. Hoogeboom and W.A. Halang, "The concept of time in software engineering for real-time systems", Proc. of 3rd Int. Conf. on 'Software engineering for real-time systems', No. 344, 16-18 Sept. 1991, pp 156 - 163.

[IEC 89] "Software for compuetrs in the application of industrial safety-related systems", IEC draft standards 65A (Secretariat) 94, and draft British Standard, Document 89/33006, BSI, 1989.

[INMOS 87] INMOS, "Occam programming manual", Prentice-Hall, 1987.

[Jackson 83] M.A. Jackson, 'System development', Prentice-Hall, 1983.

[Jahanian 86] F. Jahanian and A.K. Mok, "Safety analysis of timing properties in real-time systems", IEEE Trans. Software Eng., Vol. SE-12, No. 9, 1986, pp 890 - 904.

[Jahanian 87] F. Jahanian and A.K. Mok, "A graph-theoretic approach for timing analysis and its implementation", IEEE Trans. Software Eng., Vol. SE-36, No. 8, 1987, pp 961 - 975.

[Jahanian 88a] F. Jahanian, A.K. Mok and D.A. Stuart, "Formal specification of real-time systems", Technical Report, TR - 88 - 25, Department of Computer Science, University of Texas at Austin, June 1988.

[Jahanian 88b] F. Jahanian, R. Lee and A.K. Mok, "Semantics of Modechart in real-time logic", Proc. of 21st Annual Hawaii Int. Conf. on 'System Sciences', Kailula-Kona, HI, USA, 5-8 January 1988, pp 479 - 489.

[Jahanian 88c] F. Jahanian and D.A. Stuart, "A method for verifying properties of Modechart specifications", Proc. of IEEE Symposium on 'Real-time Systems', 6 - 8 December 1988, pp 12 - 21.

[Jensen 81] K. Jensen, "Coloured Petri nets and the invariant-method", Theoretical Computer Science, Vol. 14, 1981, pp 317 - 336.

[Jensen 87] K. Jensen, "Coloured Petri nets", In W. Brauer, W. Reisig and G. Rozenburg (Eds), 'Petri nets: Central Models and their Properties', Lecture Notes in Computer Science, Vol. 254, 1987, pp 248 - 299.

[Jones 86] C.B. Jones, 'Systematic software development using VDM', Prentice-Hall Int., 1986.

[Joseph 89] M. Joseph and A. Goswami, "Formal description of real-time systems: a review", Information and Software Technology, Vol. 31, No. 2, 1989, pp 67 - 76.

[Josephs 88] M.B. Josephs, "A state-based approach to communicating processes", Distributed Computing, Vol. 3, 1988, pp 9 - 18.

[Kamath 86] M. Kamath and N. Vishwanadham, "Applications of Petri net based models in the modelling and analysis of flexible manufacturing systems", Proc. of IEEE Int. Conf. on 'Robotics and Automation', April 1986, San Francisco, California, pp 312 - 317.

[Kaplan 85] G. Kaplan, "The X-29: is it coming or going?", IEEE Spectrum, Vol. 22, No. 6, 1985, pp 54 - 60.

[Koymans 85] R. Koymans, R. Shyamasunder, W. de Rover, R. Gerth, and S. Arun-Kumar, "Compositional semantics for real-time distributed computing", In R. Parikh (Ed) 'Logic of Programs', Lecture Notes in Computer Science, Vol. 193, Springer-Verlag, June 1985, pp 167 - 187.

[Lamport 77] L. Lamport, "Proving the correctness of multiprocess programs", IEEE Trans. Software Eng., Vol. SE-3, No. 2, 1977, pp 125 - 143.

[Lamport 80] L. Lamport, "Sometime is sometimes not never on the logic of programs", Proc. of 7th ACM Annual Symposium on 'Principles of Programming Languages', Las Vegas, N.Y., Jan 1980, pp 174 - 185.

[Lamport 83a] L. Lamport, "Specifying concurrent program modules", ACM Trans. Prog. Lang. and Systems, Vol. 5, No. 2, 1983, pp 190 - 222.

[Lamport 83b] L. Lamport, "What good is temporal logic", In R.E.A. Mason (Ed), 'Information Processing 83', Elsevier Science Pub. B.V., North-Holland, 1983, pp 657 - 668.

[Lamport 89] L. Lamport, "A simple approach to specifying concurrent systems", Comm. ACM, Vol. 32, No. 1, 1989, pp 32 - 45.

[Lautenbach 74] K. Lautenbach and H. Schmid, "Use of Petri nets for proving correctness of concurrent process systems", In 'Information Processing 74', North-Holland Pub. Co., 1974, pp 187 - 191.

[Lee 85] I. Lee and V. Gehlot, "Language constructs for distributed real-time programming", Proc. of IEEE Symposium on 'Real-time Systems', December 1985, pp 57 - 66.

[Lee 87] I. Lee and S.B. Davidson, "Adding time to synchronous communication", IEEE Trans. Computers, Vol. C-36, No. 8, 1987, pp 941 - 948.

[Lee 89] I. Lee, R.B. King and R.P. Paul, "A predictable real-time kernel for distriuted multisensor systems", IEEE Computer, Vol. 22, No. 6, 1989, pp 78 - 83.

[Lehman 81] D. Lehman, A. Pnueli and J. Stavi, "Impartiality, justice and fairness: the ethics of concurrent termination", Proc. of 8th Int. Colloquium on 'Automata, Languages and Programming', Lecture Notes in Computer Science, Vol. 115, Springer-Verlag, 1981, pp 264 - 277.

[Leinbaugh 80] D.W. Leinbaugh, "Guaranteed response times in a hard real-time environment", IEEE Trans. Software Eng., Vol SE-6, No. 1, 1980, pp 85 - 91.

[Leinbaugh 86] D.W. Leinbaugh and M.R. Yamini, "Guaranteed response times in a distributed hard real-time environment", IEEE Trans. Software Eng., Vol SE-12, No. 12, 1986, pp 1139 - 1144.

[Leveson 85] N.G. Leveson and J.L. Stolzy, "Analysing safety and fault-tolerance using time Petri nets", TAPSOFT: Joint Conference on 'Theory and Practice of Software Development', 25 - 29 March, 1985, Berlin, pp 339 - 355.

[Leveson 86] N.G. Leveson, "Reliability and safety in real-time systems", IEEE Trans. Software Eng., Vol. SE-12, No. 9, 1986, pp 877 - 878.

[Leveson 87] N.G. Leveson and J.L. Stolzy, "Safety analyses using Petri nets", IEEE Trans. Software Eng., Vol SE-13, No. 3, March 1987, pp 386 - 397.

[Liu 73] C.L. Liu and J.W. Layland, "Scheduling algorithms for multiprogramming in hard real-time environment", Journal of ACM, Vol. 20, No. 1, 1973, pp 46 - 61.

[Manna 83a] Z. Manna and A. Pnueli, "Verification of concurrent programs: the temporal framework", In R.S. Boyer and J.S. Moore (Eds), 'The Correctness Problem in Computer Science', Mathematical Center Tracts, 159, Amsterdam,1983.

[Manna 83b] Z. Manna and A. Pnueli, "Verification of concurrent programs:a temporal proof system", In 'Foundations of Computer Science', IV, Amsterdam, Mathematical Center Tracts, 1983, pp 163 - 225.

[Manna 83c] Z. Manna and A. Pnueli, "How to cook a temporal proof system for your pet language", Proc. Symposium on 'Principles of Programming Languages', Austin, Texas, Jan. 1983, pp 141 - 154.

[Menasche 83] M. Menasche and B. Berthomieu, "Time Petri nets for analyzing and verifying time dependent communication protocols", In H. Rudin and C.H. West (Eds), 'Protocol Specification, Testing and Verification', III, Elsevier Science Pub. B.V., North-Holland, IFIP, 1983, pp 161 - 172.

[Merlin 76] P.M. Merlin and D.J. Farber, "Recoverability of communication protocols-implications of a theoretical study", IEEE Trans. Comm., Vol. COM-24, No. 9, 1976, pp 1036-1043.

[MOD 89] MOD (UK) Interim Defence Standards 00-55 and 00-56, May 1989.

[Mok 78] A.K. Mok and M.L. Dertouzos, "Multiprocessor scheduling in hard real-time environment", Proc. of 7th Texas Conference on 'Computing Systems', Houston, October 1978, pp 5.1 - 5.12.

[Mok 83] A.K. Mok, 'Fundamental design problems of distributed systems for the hard real-time environment', PhD Thesis, 1983, MIT/LCS/TR - 297.

[Mok 84] A.K. Mok, "Design of real-time programming systems based on process models", Proc. of IEEE Symposium on 'Real-time Systems', December 1984, pp 5 - 17.

[Murata 88] T. Murata and D. Zhang, "A predicate-transition net model for parallel interpretation of logic programs", IEEE Trans. Software Eng., Vol. 14, No. 4, 1988, pp 481 - 497.

[Murata 89] T. Murata, "Petri nets: properties, analysis and applications", Proc. of the IEEE, Vol. 77, No. 4, 1989, pp 541 - 577.

[Olderog 86] E.R. Olderog and C.A.R. Hoare, "Specification-Orientated Semantics for Communicating Processes ", ACTA Informatica, Vol. 23, No. 1, 1986, pp 9- 66.

[Ostroff 87] J.S. Ostroff, "Modelling, specifying and verifying real-time embedded computer systems", Proc. of IEEE Symposium on 'Real-time Systems', 1987, pp 124 - 132.

[Ostroff 89] J. S. Ostroff, 'Temporal logic for real-time systems', Research Studies Press Ltd, 1989.

[Owicki 82] S. Owicki and L. Lamport, "Proving liveness properties of concurrent programs", ACM Trans. Prog. Lang. and Systems, Vol. 4, No. 3, 1982, pp 455 - 495.

[Peterson 81] J.L. Peterson, 'Petri net theory and the modelling of systems', Prentice-Hall, 1981.

[Petri 62] C. A. Petri, "Kommunikation mit automaten", Bonn: Institut fur Instrumentelle Mathematik, Schriften des IIm No. 2. English translation: "Communication with automata", Tech. Report RADC-TR-65-377, Vol. 1, Suppl 1, Applied Data Research, Princeton, NJ, 1966.

[Pnueli 77] A. Pnueli, "Temporal logic of programs", Proc. of 18th Symposium on 'The Foundations of Computer Science', Nov. 1977, pp 46 - 57.

[Pnueli 85] A. Pnueli, "In transition from global to modular temporal reasoning about programs", In K.R. Apt (Ed), 'Logics and Models of Concurrent Systems', NATO ASI Series, Vol. F13, Springer-Verlag, 1985, pp 123 - 144.

[Pnueli 86] A. Pnueli, "Applications of temporal logic to the specification and verification of reactive systems: a survey of current trends", In J. de Bakker, W.P. de Roever and G. Rozenburg (Eds), 'Current Trends in Concurrency', Lecture Notes in Computer Science, Vol. 244, Springer-Verlag, 1986, pp 510 - 584.

[Pnueli 88] A. Pnueli and E. Harel, "Applications of temporal logic to the specification of real-time systems", In M. Joseph (Ed.), 'Formal Techniques in Real-time and Fault-tolerant Systems', Lecture Notes in Computer Science, Vol. 331, Springer-Verlag, 1988, pp 84 - 98.

[Queille 82] J.P. Queille and J. Sifakis, "Specification and verification of concurrent systems in CESAR", In J.W. de Bakker, W. P. de Roever and G. Rozenburg (Eds), 'Current Trends in Concurrency: Overview and Tutorials', Lecture Notes in Computer Science, Springer-Verlag, Vol. 224, 1982, pp 510 - 584.

[Queille 83] J.P. Queille and J. Sifakis, "Fairness and related properties in transition systems - a temporal logic to deal with fairness", ACTA Informatica, Vol. 19, No. 3, 1983, pp 195 - 220.

[Ramchandani 74] C. Ramchandani, "Analysis of asynchronous concurrent systems by timed Petri nets", PhD Thesis, MIT, Project MAC-TR-120, 1974.

[Ramamoorthy 80] C.V. Ramamoorthy and G.S. Ho, "Performance evaluation of asynchronous concurrency systems using Petri nets", IEEE Trans. Software Eng., Vol. SE-6, No 5, 1980, pp 440-449.

[Ramamritham 84] K. Ramamritham and J.A. Stankovic, "Dynamic task scheduling in hard real-time distributed systems", IEEE Software, Vol. 1, No. 3, 1984, pp 65 - 75.

[Ramage 87] P.J. Ramage and W.M. Wonham, "Supervisory control of a class of discrete-event process", SIAM Journal Control and Optimization, Vol. 25, 1987, pp 206 230.

[Ratcliff 87] B. Ratcliff, 'Software engineering: principles and methods', Blackwell Scientific Publications, 1987.

[Razouk 84] R.R Razouk, "The derivation of performance expressions for communication protocols from timed Petri net models", Computer Comm. Review (USA), Vol. 14, No. 2, 1984, pp 210-217.

[Razouk 85] R.R. Razouk and C.V. Phelps, "Performance analysis using timed Petri nets", In S. Yemini, Y. Yemini and R. Strom (Eds), 'Protocols Specification, Testing and Verification', IV, Elsevier Science Pub., North Holland, 1985, pp 561-576.

[Reed 88] G.M. Reed and A.W. Roscoe, "A timed model for communicating sequential processes", Theoretical Computer Science, Vol. 58, 1988, pp 249 - 261.

[Reisig 85] W. Reisig, 'Petri nets: an introduction', Springer-Verlag, 1985.

[Reisig 88] W. Reisig, "Temporal logic and causality in concurrent systems", In F.H. Vogt (Eds), 'Concurrency 88', Lecture Notes in Computer Science, Vol. 335, Springer-Verlag, 1988, pp 121 - 139.

[Rescher 71] N. Rescher and A. Urquhart, 'Temporal logic', Springer-Verlag, 1971.

[Sagoo 90] J.S. Sagoo and D.J. Holding, "The specification and design of hard real-time systems using time and temporal Petri nets", Microprocessing and Microprogramming, Vol. 30, No. 1 - 5, 1990, pp 389 - 396.

[Sagoo 91] J.S. Sagoo and D.J. Holding, "A comparison of temporal Petri net based techniques in the specification and design of hard real-time systems", Microprocessing and Microprogramming, Vol. 32, No. 1 - 5, 1991, pp 111 - 118.

[Sagoo 92a] J.S. Sagoo and D.J. Holding, "The use of temporal Petri nets in the specification and design of systems with safety implications", Proc. of 1st IFAC Workshop on 'Architectures and algorithms for real-time control', IFAC Workshop Series, No. 4, Pergamon Press, 1992, pp 231 - 236.

[Sagoo 92b] J.S. Sagoo and D.J. Holding, "Synthesis and analysis of a real-time controller using Petri nets and concurrency sets", IEE Colloquium on 'Discrete Event Dynamic Systems - a New Generation of Modelling, Simulation and Control Applications', No. 1992/138, 1992.

[Scholefield 90] D.J. Scholefield, "The formal development of real-time systems: a review", University of York, Computer Science, 145, 1990.

[Shanker 87] A.U. Shanker and S.S. Lam, "Time-dependent distributed systems: proving safety, liveness and real-time properties", Distributed Computing, Vol. 2, 1987, pp 61 - 79.

[Shaw 89] A.C. Shaw, "Reasoning about time in higher-level language software", IEEE Software Eng., Vol. 15, No. 7, 1989, pp 875 - 884.

[Shin 85] K.G. Shin, C.M. Krishna, and Y-H, Lee, "A unified method for evaluating real-time computer controllers and its application", IEEE Trans. Automatic Control, Vol. AC-30, No. 4, 1985, pp 357 - 366.

[Shin 87] K.G. Shin, "Introduction to the special issue on real-time systems", IEEE Trans. Computers, Vol. C-36, No. 8, 1987, pp 901 - 903.

[Skeen 83] D. Skeen and M. Stonebraker, "A formal model of crash recovery in a distributed system", IEEE Trans. Software Eng., Vol. SE-9, No. 3, 1983, pp 219 - 228.

[Spivey 88] J.M. Spivey, "An introduction to Z and formal specifications", Software Eng. Journal, Vol. 4, No. 1, 1988, pp 40 - 50.

[Stanat 77] D.F. Stanat, "Discrete mathematics in computer science", Englewood Cliffs London, Prentice-Hall, 1977.

[Stankovic 85] J.A. Stankovic, K. Ramamritham and S. Cheng, "Evaluation of a flexible task scheduling algorithm for distributed hard real-time systems", IEEE Trans. Computers, Vol. C-34, No. 12, December 1985, pp 1130 - 1139.

[Stankovic 87] J.A. Stankovic, K. Ramamritham and W. Zhao, "Pre-emptive scheduling under time and resource constraints", IEEE Trans. Computers, Vol. CE-36, No.8, 1987, pp 949 - 960.

[Stankovic 88] J.A. Stankovic, "A serious problem for next generation systems", IEEE Computer, Vol. 21, No. 10, 1988, pp 10 - 19.

[Stotts 82] P.D. Stotts, "A comparative study of concurrent programming languages", ACM SIGPLAN Notices, Vol. 17, No. 10, 1982, pp 50 - 61.

[Suzuki 85] I. Suzuki, "Fundamental properties and application of temporal Petri nets", Proc. of 9th Annual Conf. on 'Inform. Sci. Syst.' , John Hopkins Univ., Baltimore, MD, March 1985, pp 641 - 646.

[Suzuki 89] I. Suzuki and H. Lu, "Temporal Petri nets and their application to modelling and analysis of a handshake daisy chain arbiter", IEEE Trans. Computers., Vol. 38, No. 5, 1989, pp 696 - 704.

[Suzuki 90] I. Suzuki, "Formal analysis of the alternating bit protocol using temporal Petri nets", IEEE Trans. Software Eng., Vol. 16, No. 11, 1990, pp 1273 - 1281.

[Suzuki 91] I. Suzuki, *private communication on temporal Petri nets*, June, 1991.

[Thayse 89] A. Thayse, "From modal logic to deductive databases", John Wiley, 1989.

[Volz 87] R.A. Volz and T.N. Mudge, "Timing issues in the distributed execution of ADA programs", IEEE Trans. Computers, Vol. C-36, No. 4, 1987, pp 449 - 459.

[Ward 86] P.T. Ward and S.J. Mellor, 'Structured development for real-time systems', Englewood Cliffs NJ, Yourdon Press, 1986.

[Wayman 87] R. Wayman, "OCCAM 2: An overview from a software engineering perspective", Microsystems and Microprogramming, Vol. 11, No. 8, 1987, pp 413 - 422.

[Wing 89] J. M. Wing and M.R. Nixon,"Extending Ina Jo with temporal logic", IEEE Trans. Software Eng., Vol. 15, No. 2, 1989, pp 181 - 197.

[Wing 90] J. M. Wing, "A specifier's introduction to formal methods", Computer, Vol. 23, No. 9, 1990, pp 8 - 24.

[Wirth 77] N. Wirth, "Toward a discipline of real-time programming", Comm. ACM, Vol. 20, No. 8, 1977, pp 577 -583.

[Wu 83] Z. Wu and T. Murata, "A Petri net model of a starvation-free solution to the dining philosophers problem", IEEE workshop on 'Languages for Automation', Chicago, USA, 7 - 9 Nov. 1983, pp 192 - 195.

[Zave 82] P. Zave, "An operational approach to requirements specification for embedded systems", IEEE Trans. Software Eng., Vol SE-8, No. 3, 1982, pp 250 - 269.

[Zave 84] P. Zave, "The operational vs the conventional approach to software development", Comm. ACM, Vol. 27, No. 2, 1984, pp 104 - 118.

[Zave 86] P. Zave and W. Schell, "Salient features of an executable specification language and its environment", IEEE Trans. Software Eng., Vol SE-12, No. 2, 1986, pp 312 - 325.

[Zave 88] P. Zave, "Assessment", ACM SIGSOFT Software Eng. Notes, Vol. 13, No. 1, 1988, pp 40 - 43.

[Zave 91] P. Zave, "An insider's evaluation of PAISLey", IEEE Trans. Software Eng., Vol SE-17, No. 3, 1991, pp 212 - 225.

[Zuberek 80] W.M. Zuberek, "Timed Petri net and preliminary performance evaluation", Proc. of 7th Annual Symposium on 'Computer Architecture', 1980, pp 88-96.

Appendix A

Petri net analysis using matrix manipulation

State equations

The properties of Petri nets can be analysed using matrix theory and linear algebra, this can be used to define the following state equation for a Petri net:

$$\mathbf{M_k} = \mathbf{M_{k-1}} + \mathbf{N^T}\,\mathbf{U_k} \qquad \text{A.1}$$

where

k = 1,2, ...

M_k represents the Petri net marking such that the jth entry of **M_k** denotes the number of tokens in place p_j immediately after the kth firing in some transition firing sequence,

M_{k-1} represents the marking of the Petri net before the kth firing in some transition firing sequence,

U_k is referred to as the kth firing or control vector and is a column vector of size n x 1. It is made up of (n-1) 0's and one non zero entry; a 1 in the ith position indicates that transition t_i fires at the kth firing,

N represents the incidence matrix of the Petri net; its size is given by n x m for C with n transitions and m places. The elements of **N** are defined by the weights given to the arcs, that interconnect the places and transitions, of C, such that:

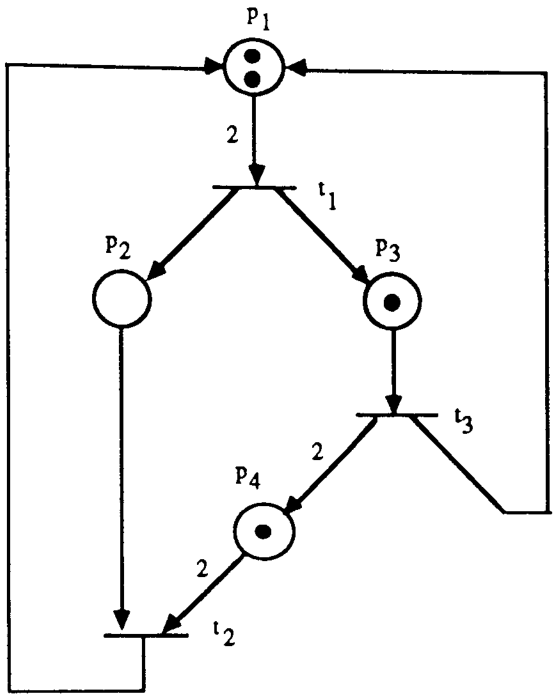
(i) $-x_{ji}$ is assigned to the number of arcs that connect a place p_j to a transition t_i, ie the input place p_j of t_i (where $x \geq 1$),

(ii) $+x_{ij}$ is assigned for the number of arcs that connect a transition t_j to a place p_i,

(iii) 0 is assigned for no connection between place p_i and transition t_j.

Application of Equation A.1 will be illustrated for the Petri net of Figure A.1.

Figure A.1 Petri net



For the marking shown in Figure A.1, assuming that transition t3 fires at the next instant, the components of the Equation A.1 can be defined as:

$$N = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \end{array}\begin{array}{c} \begin{array}{cccc} p_1 & p_2 & p_3 & p_4 \end{array} \\ \begin{bmatrix} -2 & 1 & 1 & 0 \\ 1 & -1 & 0 & -2 \\ 1 & 0 & -1 & 2 \end{bmatrix} \end{array}$$

the transpose of this incidence matrix **N** becomes:

$$N^T = \begin{bmatrix} -2 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & -2 & 2 \end{bmatrix}$$

$$U_k = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad M_{k-1} = \begin{bmatrix} 2 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Substituting the above matrices in Equation A.1 yields:

$$M_k = \begin{bmatrix} 2 \\ 0 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} -2 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & -2 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Therefore M_k can be calculated as:

$$M_k = \begin{bmatrix} 3 \\ 0 \\ 0 \\ 3 \end{bmatrix}$$

The above result for M_k can be easily verified to be correct by examining the reachability graph for Figure A.1.

Further consideration of Equation A.1 shows how expressions for the S and T - invariants can be derived, suppose that marking M_d is reachable from M_0 through a firing sequence $\{U_1, U_2, \ldots , U_d\}$. Therefore, writing the Equation A.1 for i = {1, 2, ... , d}, the following is obtained:

$$M_d = M_0 + N^T \sum_{k=1}^{d} U_k \qquad \text{A.2}$$

Equation A.2 can be rewritten as:

$$\mathbf{N}^T \mathbf{x} = \Delta \mathbf{M} \qquad \text{A.3}$$

where $\Delta \mathbf{M} = \mathbf{M_d} - \mathbf{M_o}$ and

$$\mathbf{x} = \sum_{k=1}^{d} \mathbf{U_k}$$

x is a (n x 1) column vector of non-negative integers and is called the firing count vector. The ith element of **x** denotes the number of times that transition t_i must fire to transform $\mathbf{M_o}$ to $\mathbf{M_d}$. However, if Equation A.3 is written as

$$\mathbf{N}^T \mathbf{x} = 0 \qquad \text{A.4}$$

then an integer solution **x** of the homogeneous Equation A.4 is referred to as T - invariants Similarly, the solution of **y** for the homogeneous equation

$$\mathbf{N}\mathbf{y} = 0 \qquad \text{A.5}$$

is called an S - invariant.

Appendix B

This appendix presents a proof of Proposition 2 which was stated in Section 3.3.2 as:

Proposition 2 (Less restricted alternative to Proposition 1)

For a temporal Petri net TN = (C,f) whose initial marking is M_0.

If

(i) for any marking M reachable from M_0, if transitions t_1 and t_2, with input functions $I(t_1)$ and $I(t_2)$ respectively, are firable at M and are in conflict, which is defined by:

$$I(t_1) \cap I(t_2) \neq \varnothing \qquad 3.7$$

where $\varnothing$ is the empty set,

then either t_1 remains firable until either t_1 itself fires or t_2 fires, else t_2 remains firable until either t_2 itself fires or t_1 fires. This statement is formalised as:

$$\begin{aligned}<M_0,\alpha> \models\ & \Box[t_1(ok) \Rightarrow t_1(ok)\mathcal{U}(t_1 \vee t_2)] \vee \\ & \Box[t_2(ok) \Rightarrow t_2(ok)\mathcal{U}(t_1 \vee t_2)] \qquad 3.8\end{aligned}$$

and

(ii) whenever t_1 becomes firable then it must become disabled and whenever t_2 becomes firable then it must become disabled, ie:

f implies

$$\begin{aligned}<M_0,\alpha> \models\ & \Box[t_1(ok) \Rightarrow \Diamond t_1(\neg ok)] \wedge \\ & \Box[t_2(ok) \Rightarrow \Diamond t_2(\neg ok)]. \qquad 3.9\end{aligned}$$

Then

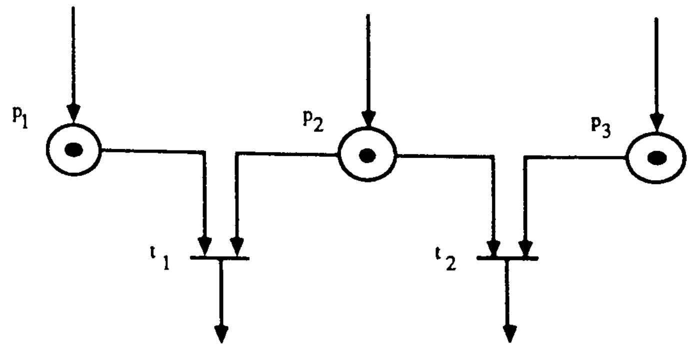
(iii) for any α from M_0, it follows from (i) and (ii) that:

$$<M_0,\alpha> \models \Box[t_1(ok) \wedge t_2(ok) \Rightarrow \Diamond(t_1 \vee t_2)] \qquad 3.10$$

Proof

Proposition 2 is applicable to situations in which transitions are in conflict and typically this can be represented by the Petri net structure shown in Figure B.1. The following proof of Proposition 2 will refer to Figure B.1 in order to better illustrate the steps used.

Figure B.1 A partial Petri net structure representing transitions in conflict



When transitions are in conflict their input functions are related by the expression given in Equ. 3.7; this is easily seen for the Petri net structure of Figure B.1 in which the input functions are:

$$I(t_1) = \{p_1, p_2\} \text{ and } I(t_1) = \{p_2, p_3\}.$$

Note if the expression of Equ 3.7 does not apply then transitions are not in conflict hence Propositions 1 can be applied.

Equs 3.8 and 3.9 assert the following:

1. $<M_0,\alpha> \models \Box[t_1(ok) \Rightarrow t_1(ok)\mathcal{U}(t_1 \vee t_2)] \vee \Box[t_2(ok) \Rightarrow t_2(ok)\mathcal{U}(t_1 \vee t_2)]$,
2. $<M_0,\alpha> \models \Box[t_1(ok) \Rightarrow \Diamond t_1(\neg ok)] \wedge \Box[t_2(ok) \Rightarrow \Diamond t_2(\neg ok)]$.

Since the transition firing rules for Petri nets is based on interleaving (see Section 2.2) then only one transition is allowed to fire at any instant. Hence, for the case in which enabled transitions are in conflict, only one transition can fire at any instant, and the choice of which transition to fire is determined either non-deterministically or based on a fairness requirement. Since there is no fairness requirement attached in the above proposition then these transitions will be fired non-deterministically. Thus the following will consider the consequences of firing each transition one at a time.

If transition t_1 fires then 1. can be simplified using the temporal logic proof system of [Manna 83b] and written in the form:

3. $<M_0,\alpha> \models \Box[t_1(ok) \Rightarrow \Diamond t_1] \vee \Box[t_2(ok) \Rightarrow \Diamond t_1]$

Simplifying 3. yields:

4. $<M_0,\alpha> \models \Box[(t_1(ok) \wedge t_2(ok)) \Rightarrow \Diamond t_1]$

Although the premise of 2. is only used in Proposition 2 in order to guarantee that transitions t_1 and t_2 actually become disabled once they have been enabled, it can be combined with 4. to yield:

5. $<M_0,\alpha> \models \Box[(t_1(ok) \wedge t_2(ok)) \Rightarrow (\Diamond t_1 \wedge \Diamond t_1(\neg ok) \wedge \Diamond t_2(\neg ok))]$

Simplifying 5. yields the same result of 4. and is re-written as:

6. $<M_0,\alpha> \models \Box[(t_1(ok) \wedge t_2(ok)) \Rightarrow \Diamond t_1]$

Applying the same reasoning as 3. - 6. to the firing of transition t_2 yields the following:

7. $<M_0,\alpha> \models \Box[(t_1(ok) \wedge t_2(ok)) \Rightarrow \Diamond t_2]$

Combining 6. and 7. using temporal reasoning yields:

8. $<M_0,\alpha> \models \Box[(t_1(ok) \wedge t_2(ok)) \Rightarrow \Diamond t_1] \wedge \Box[(t_1(ok) \wedge t_2(ok)) \Rightarrow \Diamond t_2]$

The premise of 8. can be simplified using propositional logic to:

9. $<M_0,\alpha> \models \Box[(t_1(ok) \wedge t_2(ok)) \Rightarrow \Diamond(t_1 \vee t_2)]$ ■

This completes the proof of Proposition 2 because 9. shows the same result as Equ. 3.10.

Appendix C

C.1 Proof of properties using temporal Petri net analysis

(i) *Lemma 3*

$$<M_0, \alpha> \models \Box[t_{10} \Rightarrow \Diamond t_{13}] \qquad 4.24$$

Proof

The part of the reachability graph Figure 4.6(b) relevant to the proof of Lemma 3 is that which commences with the firing of t_{10} and completes with the firing of t_{13}. All marking sequences included within t_{10} and t_{13} show that two transitions are firable, however, they fire independently of each other thus Proposition 1 is valid for the firing of these transitions. The proof should proceed by considering all marking sequences existing within the firing of t_{10} and t_{13}. However, the following will illustrate the firing sequence $\{t_{10}, t_{11}, t_{12}, t_{13}\}$ commencing at M_6. Thus, in this sequence the condition of Proposition 1(i) holds true, and Proposition 1(ii) is satisfied by the following:

1. $<M_0, \alpha> \models \Box[t_{11}(ok) \Rightarrow \Diamond t_{11}(\neg ok)]$,
2. $<M_0, \alpha> \models \Box[t_{12}(ok) \Rightarrow \Diamond t_{12}(\neg ok)]$,
3. $<M_0, \alpha> \models \Box[t_{13}(ok) \Rightarrow \Diamond t_{13}(\neg ok)]$.

Therefore, by Proposition 1 it can be asserted that:

4. $<M_0, \alpha> \models \Box[t_{11}(ok) \Rightarrow \Diamond t_{11}]$,
5. $<M_0, \alpha> \models \Box[t_{12}(ok) \Rightarrow \Diamond t_{12}]$,
6. $<M_0, \alpha> \models \Box[t_{13}(ok) \Rightarrow \Diamond t_{13}]$.

From the reachability graph of Figure 4.6(b) it can be inferred that when t_{10} fires then t_{11} becomes firable at the next marking. This and similar observations concerning t_{12} and t_{13} yields:

7. $<M_0, \alpha> \models \Box[t_{10} \Rightarrow O t_{11}(ok)]$,
8. $<M_0, \alpha> \models \Box[t_{11} \Rightarrow O t_{12}(ok)]$,
9. $<M_0, \alpha> \models \Box[t_{12} \Rightarrow O t_{13}(ok)]$.

Combining 4., 7., 8., 5., 9. and 6. yields:

10. $<M_0, \alpha> \models \Box[t_{10} \Rightarrow \Diamond t_{13}]$ ■

Note the same result as 10. is obtained by other firing sequences arising between t_{10} and t_{13}.

(ii) *Lemma 4*

$$<M_0, \alpha> \models \Box[t_{13} \Rightarrow \Diamond(t_2 \vee t_8)] \qquad 4.25$$

Proof

The proof of Lemma 4 uses the part of Figure 4.6(b) which commences by the firing of t_{13} and completes with the firing of t_2 or t_8. However, t_{13} can fire under several markings, thus all firing sequences commencing from these markings must be considered. The following demonstrates this reasoning for the firing sequences of $T_1 = \{t_{13}, t_{15}, t_{14}, t_1, t_2\}$ commencing at $M_{12} = \{p_9, p_{13}, p_{17}\}$ and $T_2 = \{t_{13}, t_7, t_8\}$ commencing at $M_{28} = \{p_6, p_9, p_{13}\}$. For the markings generated by T_1 Proposition 1(i) is satisfied hence Proposition 1(ii) can be asserted as:

1. $<M_0, \alpha> \models \Box[t_{15}(ok) \Rightarrow \Diamond t_{15}(\neg ok)]$,
2. $<M_0, \alpha> \models \Box[t_{14}(ok) \Rightarrow \Diamond t_{14}(\neg ok)]$,
3. $<M_0, \alpha> \models \Box[t_1(ok) \Rightarrow \Diamond t_1(\neg ok)]$,
4. $<M_0, \alpha> \models \Box[t_2(ok) \Rightarrow \Diamond t_2(\neg ok)]$.

By Proposition 1 it can be asserted that:

5. $<M_0, \alpha> \models \Box[t_{15}(ok) \Rightarrow \Diamond t_{15}]$,
6. $<M_0, \alpha> \models \Box[t_{14}(ok) \Rightarrow \Diamond t_{14}]$,
7. $<M_0, \alpha> \models \Box[t_1(ok) \Rightarrow \Diamond t_1]$,
8. $<M_0, \alpha> \models \Box[t_2(ok) \Rightarrow \Diamond t_2]$.

From the reachability graph of Figure 4.6(b) it can be inferred that:

9. $<M_0, \alpha> \models \Box[t_{13} \Rightarrow O t_{15}(ok)]$,
10. $<M_0, \alpha> \models \Box[t_{15} \Rightarrow O t_{14}(ok)]$,
11. $<M_0, \alpha> \models \Box[t_{14} \Rightarrow O t_1(ok)]$,
12. $<M_0, \alpha> \models \Box[t_1 \Rightarrow O t_2(ok)]$.

Combining 5., 9., 10., 6., 11., 7., 12. and 8. yields:

13. $<M_0, \alpha> \models \Box[t_{13} \Rightarrow \Diamond t_2]$.

For the markings generated by T_2 Proposition 1(i) is satisfied, thus Proposition 1(ii) can be asserted as:

14. $<M_0, \alpha> \models \Box[t_7(ok) \Rightarrow \Diamond t_7(\neg ok)]$,
15. $<M_0, \alpha> \models \Box[t_8(ok) \Rightarrow \Diamond t_8(\neg ok)]$.

By Proposition 1 the following can be stated:

16 $<M_0, \alpha> \models \Box[t_7(ok) \Rightarrow \Diamond t_7]$,
17. $<M_0, \alpha> \models \Box[t_8(ok) \Rightarrow \Diamond t_8]$.

From Figure 4.6(b) it can be inferred that:

18. $<M_0, \alpha> \models \Box[t_{13} \Rightarrow O t_7(ok)]$,
19. $<M_0, \alpha> \models \Box[t_7 \Rightarrow O t_8(ok)]$.

Combining 16. 18, 19. and 17. yields:

20. $<M_0, \alpha> \models \square[t_{13} \Rightarrow \bigcirc t_8]$.

Since the firing of t_{13} can lead to the firing of t_2 or t_8 then combining 13. and 20. yields:

21. $<M_0, \alpha> \models \square[t_{13} \Rightarrow \Diamond(t_2 \vee t_8)]$. ■

Note the same result as 10. is obtained by other firing sequences arising between t_{10} and t_{13}.

(iii) *Lemma 5*

$$<M_0, \alpha> \models \square[t_3 \Rightarrow \Diamond t_6] \qquad 4.26$$

Proof

Transition t_3 can fire under several markings of the net, which are shown in Figure 4.6(b) as $\{M_{15}, M_{19}, M_{22}, M_{25}\}$, each of these cases should be considered in the proof of Lemma 5. However, the following will consider the particular firing sequence generated from M_{25}, ie $\{t_3, t_{13}, t_7, t_8, t_9, t_5, t_6\}$, the same reasoning can be applied to firing sequences generated by other markings. Thus, in this firing sequence Proposition 1 holds true, hence the following can be written:

1. $<M_0, \alpha> \models \square[t_{13}(ok) \Rightarrow \Diamond t_{13}]$,
2. $<M_0, \alpha> \models \square[t_7(ok) \Rightarrow \Diamond t_7]$,
3. $<M_0, \alpha> \models \square[t_8(ok) \Rightarrow \Diamond t_8]$,
4. $<M_0, \alpha> \models \square[t_9(ok) \Rightarrow \Diamond t_9]$,
5. $<M_0, \alpha> \models \square[t_5(ok) \Rightarrow \Diamond t_5]$,
6. $<M_0, \alpha> \models \square[t_6(ok) \Rightarrow \Diamond t_6]$.

From the reachability graph of Figure 4.6(b) it can be inferred that :

7. $<M_0, \alpha> \models \square[t_3 \Rightarrow \bigcirc t_{13}(ok)]$,
8. $<M_0, \alpha> \models \square[t_{13} \Rightarrow \bigcirc t_7(ok)]$,
9. $<M_0, \alpha> \models \square[t_7 \Rightarrow \bigcirc t_8(ok)]$,
10. $<M_0, \alpha> \models \square[t_8 \Rightarrow \bigcirc t_9(ok)]$,
11. $<M_0, \alpha> \models \square[t_9 \Rightarrow \bigcirc t_5(ok)]$,
12. $<M_0, \alpha> \models \square[t_5 \Rightarrow \bigcirc t_6(ok)]$.

Combining 1. - 12., yields:

13. $<M_0, \alpha> \models \square[t_3 \Rightarrow \Diamond t_6]$. ■

C.2 Proof of properties using extended temporal Petri nets

Lemma 2

$$\Box\neg[p2 \wedge \neg(p9 \vee p15)] \qquad 4.28$$

Proof

Assuming the negation of Lemma 2 and using the definition $\neg\Box\neg w \equiv \Diamond w$, where w is any well-formed propositional formula, then Lemma 2 can be written as:

1. $\Diamond[p2 \wedge \neg(p9 \vee p15)]$

Applying each applicable inference rule of Figure 4.6(a) to A1, it can be deduced that:

2. $\neg[p2 \wedge \neg(p9 \vee p15)]$

In the temporal proof system of [Manna 83b] the following theorem is valid:

3. $w \wedge \Diamond\neg w \Rightarrow \Diamond(w \wedge \bigcirc\neg w)$

The conjunction of 1. and 2. yields:

4. $\Diamond[p2 \wedge \neg(p9 \vee p15)] \wedge \neg[p2 \wedge \neg(p9 \vee p15)]$

Applying Modus Ponens to 3. and 4.(where $\neg[p2 \wedge \neg(p9 \vee p15)]$is substituted for w) the following results:

5. $\Diamond\{\neg[p2 \wedge \neg(p9 \vee p15)] \wedge \bigcirc[p2 \wedge \neg(p9 \vee p15)]\}$

Using De Morgans theorem on the first conjunct of 5. yields:

6. $\Diamond\{[\neg p2 \vee (p9 \vee p15)] \wedge \bigcirc[p2 \wedge \neg(p9 \vee p15)]$

Expanding 6. yields:

7. $\Diamond\{[\neg p2 \wedge \bigcirc[p2 \wedge \neg(p9 \vee p15)]] \vee [(p9 \vee p15) \wedge \bigcirc[p2 \wedge \neg(p9 \vee p15)]]\}$

Examining each disjunct of 7.in turn to establish the truth value of 7:

8. $\neg p2 \wedge \bigcirc[p2 \wedge \neg(p9 \vee p15)]$ first disjunct of 7.
9. $\neg p2 \wedge \bigcirc p2$ by weakening 8.
10. $\bigcirc\neg(p9 \vee p15)]$ by weakening 8.

The system dependent inference rule of Figure 8.2. which show the sequence of 9. is:

IR2: $p1 \wedge \neg p2 \Rightarrow \bigcirc(p2 \wedge \neg p1)$

From Figure 4.6(b) it can inferred that when p2 is tokenised then the global marking can be: $\{M_2, M_{15}, M_{19}, M_{22}, M_{25}\}$; in these markings it is seen that either p9 or p15 is tokenised, this shows 10. to be false, thus the first disjunct of 7. is false.

Consider the second disjunct of 7:

11. $(p9 \vee p15) \wedge \bigcirc[p2 \wedge \neg(p9 \vee p15)]$

Expanding 11. gives:

12. $[p9 \wedge \bigcirc[p2 \wedge \neg(p9 \vee p15)]] \vee [p15 \wedge \bigcirc[p2 \wedge \neg(p9 \vee p15)]]$

Applying De Morgans to 12. gives:

13. $[p9 \wedge O[p2 \wedge \neg p9 \wedge \neg p15]] \vee [p15 \wedge O[p2 \wedge \neg p9 \wedge \neg p15]]$

Examining each disjunct of 13. produces:

14.	$[p9 \wedge O[p2 \wedge \neg p9 \wedge \neg p15]]$	first disjunct of 13.
15.	$p9 \wedge O\neg p9$	by weakening 14.
16.	$O(p2 \wedge \neg p15)]$	by weakening 14.

The system dependent inference rule of Figure 4.6(a) which show the sequence of 15. occurs is:

IR14: $p9 \wedge p13 . \wedge \neg p14 . \wedge \neg p15 \Rightarrow O(p14 \wedge p15 \wedge \neg p9 \wedge \neg p13)$

From the consequence of IR14. it can be seen that $Op15$ is obtained which contradicts 16., thus proving the falsity of 14.

Considering the second disjunct of 13.

17.	$[p15 \wedge O[p2 \wedge \neg p9 \wedge \neg p15]]$	
18.	$p15 \wedge O\neg p15$	by weakening 17.
19.	$O(p2 \wedge \neg p9)]$	by weakening 17.

The system dependent inference rules of Figure 4.6(a) which show the sequence of 18. occurs are:

IR3. $p2 \wedge p15 \wedge \neg p3 . \Rightarrow O(p3 \wedge \neg p2 \wedge \neg p15)$

IR9. $p7 \wedge p15 \wedge \neg p8 . \Rightarrow O(p8 \wedge \neg p7 \wedge \neg p15)$

Clearly, the consequence of IR3. contradicts 19. However, the consequence of IR9 shows that $p8$ is tokenised. To check if $(p2 \wedge \neg p9)$ is valid when $p8$ is tokenised it is sufficient to examine the markings which include $p8$. From the reachability graph of Figure 4.6(b) it can be seen that such a marking is: $M_3 = \{p8, p14\}$; this marking can be represented in logical form as:

$$M3 : \quad (p8 \wedge p14 . \wedge \neg p1 \wedge \neg p2 \wedge \neg p3 \wedge \neg p4 \wedge \neg p5 \wedge \neg p6 \wedge \neg p7 \wedge \neg p9 \wedge \neg p11 \wedge \neg p12 \wedge \neg p13 \wedge \neg p15 \wedge \neg p16 \wedge \neg p17)$$

From this marking it can be seen that a contradiction of 19. is obtained, because $M3$ can be simplified to $(\neg p2 \wedge \neg p9)$. Hence this result shows the falsity of 17.; moreover, the truth value of 11. and 8. falsifies 1. which in turn proves the truth of Lemma 2. ■

Appendix D

D.1 Comparison of the markings of C_{min} and the markings obtained from all the concurrency sets of Figure 7.2

In Section 7.3.1.2 it was stated that in order to show that Table 7.2 is the minimum concurrency set of Figure 7.2 it is sufficient to show that the markings obtained from this table are the same as the markings obtained from all the concurrency sets of Figure 7.2. This section produces these markings so that they can be compared.

The markings obtained from the C_{min} of Table 7.2 are shown in Table D.1. This table shows that the Petri net of Figure 7.2 contains fifty unique markings.

Table D.1 Markings generated from the concurrency sets of Table 7.2

MARKING	PLACES
M_0	$p_{14}.p_{15}.p_{16}.x_1.y_1$
M_1	$p_{14}.p_{15}.p_{16}.x_1.y_2.y_7$
M_2	$p_{14}.p_{15}.p_{16}.x_1.y_5$
M_3	$p_2.p_{14}.p_{15}.x_1.y_2$
M_4	$p_4.p_{14}.x_1.y_2.y_{11}$
M_5	$p_4.p_{14}.x_1.y_4$
M_6	$p_4.p_{14}.x_1.y_5.y_{10}$
M_7	$p_4.p_{14}.x_1.y_1.y_{10}$
M_8	$p_4.p_{14}.x_1.y_2.y_7.y_{10}$
M_9	$p_4.p_{14}.x_1.y_2.y_8.y_9$
M_{10}	$p_4.p_{14}.x_1.y_3.y_9$
M_{11}	$p_5.p_{14}.x_1.y_2.y_{11}$
M_{12}	$p_5.p_{14}.x_1.y_4$
M_{13}	$p_5.p_{14}.x_1.y_5.y_{10}$
M_{14}	$p_5.p_{14}.x_1.y_1.y_{10}$
M_{15}	$p_5.p_{14}.x_1.y_2.y_7.y_{10}$
M_{16}	$p_5.p_{14}.x_1.y_2.y_8.y_9$
M_{17}	$p_5.p_{14}.x_1.y_3.y_9$
M_{18}	$p_8.p_{14}.x_1.y_4$
M_{19}	$p_8.p_{14}.x_1.y_5.y_{10}$
M_{20}	$p_8.p_{14}.x_1.y_1.y_{10}$
M_{21}	$p_8.p_{14}.x_1.y_2.y_7.y_{10}$
M_{22}	$p_8.p_{14}.x_1.y_2.y_8.y_9$
M_{23}	$p_8.p_{14}.x_1.y_3.y_9$
M_{24}	$p_9.p_{14}.p_{16}.x_1.y_1$
M_{25}	$p_9.p_{14}.p_{16}.x_1.y_2.y_7$
M_{26}	$p_9.p_{14}.p_{16}.x_1.y_5$
M_{27}	$p_2.p_9.p_{14}.x_1.y_2$
M_{28}	$p_7.p_9.p_{14}.x_1.y_2.y_8$
M_{29}	$p_7.p_9.p_{14}.x_1.y_3$
M_{30}	$p_7.p_{14}.p_{15}.x_1.y_2.y_8$
M_{31}	$p_7.p_{14}.p_{15}.x_1.y_3$
M_{32}	$p_9.p_{12}.p_{16}.x_1.x_3.y_1$
M_{33}	$p_9.p_{12}.p_{16}.x_2.y_1$
M_{34}	$p_9.p_{12}.p_{16}.x_1.x_4.y_1$
M_{35}	$p_9.p_{12}.p_{16}.x_1.x_3.y_2.y_7$
M_{36}	$p_9.p_{12}.p_{16}.x_2.y_2.y_7$
M_{37}	$p_9.p_{12}.p_{16}.x_1.x_4.y_2.y_7$
M_{38}	$p_9.p_{12}.p_{16}.x_1.x_3.y_5$
M_{39}	$p_9.p_{12}.p_{16}.x_2.y_5$
M_{40}	$p_9.p_{12}.p_{16}.x_1.x_4.y_5$
M_{41}	$p_2.p_9.p_{12}.x_1.x_3.y_2$
M_{42}	$p_2.p_9.p_{12}.x_2.y_2$
M_{43}	$p_2.p_9.p_{12}.x_1.x_4.y_2$
M_{44}	$p_7.p_9.p_{12}.x_1.x_3.y_2.y_8$
M_{45}	$p_7.p_9.p_{12}.x_2.y_2.y_8$
M_{46}	$p_7.p_9.p_{12}.x_1.x_4.y_2.y_8$
M_{47}	$p_7.p_9.p_{12}.x_1.x_3.y_3$
M_{48}	$p_7.p_9.p_{12}.x_2.y_3$
M_{49}	$p_7.p_9.p_{12}.x_1.x_4.y_3$

The concurrency sets for Figure 7.2 are generated for each place in this Petri net. The following shows the set of markings for the concurrency set of each place in Figure 7.2; specifically these concurrency sets are shown in terms of the markings of Table D.1:

(i) Concurrency sets for each place in the controller net of Figure 7.2 are:

(a) Place p_2 indicates that the slider is at the decision point in the controller's model. This place yields a concurrency having the following set of markings:

$M(C(p_2)) = \{M_3, M_{27}, M_{41}, M_{42}, M_{43}\}$

The concurrency set can be easily derived from the above markings set ($M(C(p_2))$) as illustrated. The markings in the above set are:

$M_3 = \{p_2, p_{14}, p_{15}, x_1, y_2\}$ $M_{27} = \{p_2, p_9, p_{14}, x_1, y_2\}$

$M_{41} = \{p_2, p_9, p_{14}, x_1, y_2\}$ $M_{42} = \{p_2, p_9, p_{12}, x_2, y_2\}$

$M_{43} = \{p_2, p_9, p_{12}, x_1, x_4, y_2\}$

From the these markings the concurrency set for p_2 is:

$C(p_2) = \{(p_{14}, p_{15}, x_1, y_2), (p_9, p_{14}, x_1, y_2), (p_9, p_{14}, x_1, y_2), (p_9, p_{12}, x_2, y_2), (p_9, p_{12}, x_1, x_4, y_2)\}.$

(b) Place p_4 indicates that the slider has inserted into the drum, which is stationary and in position in the controller's model. This place yields a concurrency having the following set of markings:

$M(C(p_4)) = \{M_4, M_5, M_6, M_7, M_8, M_9, M_{10}\}$

(c) Place p_5 indicates that the slider withdraws from maximum insertion in the drum. This place yields a concurrency having the following set of markings:

$M(C(p_5)) = \{M_{11}, M_{12}, M_{13}, M_{14}, M_{15}, M_{16}, M_{17}\}$

(d) Place p_7 indicates that the slider has come to rest after an abort motion in the controller's model. This place yields a concurrency having the following set of markings:

$M(C(p_7)) = \{M_{28}, M_{29}, M_{30}, M_{31}, M_{44}, M_{46}, M_{47}, M_{48}, M_{49}, M_{50}\}$

(e) Place p_8 indicates that the slider is moving towards the drum in order to make an insertion, after it has taken an abort motion. This place yields a concurrency having the following set of markings:

$M(C(p_8)) = \{M_{18}, M_{19}, M_{20}, M_{21}, M_{22}, M_{23}\}$

(f) Place p9 indicates that the drum is either rotating or it is rotate enabled in the controller's model. This place yields a concurrency having the following set of markings:

$$M(C(p9)) = \{M24, M25, M26, M27, M28, M29, M32, M33, M34, M35, M36, M37, M38, M39, M40, M41, M42, M43, M44, M46, M47, M48, M49, M50\}$$

(g) Place p12 indicates that the drum is rotating in the controller's model. This place yields a concurrency having the following set of markings:

$$M(C(p12)) = \{M32, M33, M34, M35, M36, M37, M38, M39, M40, M41, M42, M43, M44, M46, M47, M48, M49, M50\}$$

(h) Place p14 indicates that the drum is stationary in the controller's model. This place yields a concurrency having the following set of markings:

$$M(C(p14)) = \{M0, M1, M2, M3, M4, M5, M6, M7, M8, M9, M10, M11, M12, M13, M14, M15, M16, M17, M18, M19, M20, M21, M22, M23, M24, M25, M26, M27, M28, M29, M30, M31\}$$

(i) Place p15 represents a drum status indicating that it is stationary in the controller's model. This place yields a concurrency having the following set of markings:

$$M(C(p15)) = \{M0, M1, M2, M3, M30, M31\}$$

(j) Place p16 indicates that the slider is at rest in the controller's model. This place yields a concurrency having the following set of markings:

$$M(C(p14)) = \{M0, M1, M2, M24, M25, M26, M32, M33, M34, M35, M36, M37, M38, M39, M40\}$$

(ii) Concurrency sets for places that represent the motion of the drum in the physical system

(a) Place x_1 represents the drum which stationary and in position. This place yields a concurrency having the following set of markings:

$$M(C(x_1)) = \{M_0, M_1, M_2, M_3, M_4, M_5, M_6, M_7, M_8, M_9, M_{10}, M_{11}, M_{12}, M_{13}, M_{14}, M_{15}, M_{16}, M_{17}, M_{18}, M_{19}, M_{20}, M_{21}, M_{22}, M_{23}, M_{24}, M_{25}, M_{26}, M_{27}, M_{28}, M_{29}, M_{30}, M_{31}, M_{32}, M_{34}, M_{35}, M_{37}, M_{38}, M_{40}, M_{41}, M_{43}, M_{44}, M_{47}, M_{48}, M_{50}\}$$

(b) Place x_2 indicates that the drum is rotating in the physical system. This place yields a concurrency having the following set of markings:

$$M(C(x_2)) = \{M_{33}, M_{36}, M_{39}, M_{42}, M_{46}, M_{49}\}$$

(iii) Concurrency sets for places that represent the interaction between the controller and drum in the physical system

(a) Place x_3 represents the signal which provides an output to the drum in order to enable it to rotate. This place yields a concurrency having the following set of markings:

$$M(C(x_3)) = \{M_{32}, M_{35}, M_{38}, M_{41}, M_{44}, M_{48}\}$$

(b) Place x_4 represents the signal which provides an input to the controller indicating that the drum is stationary. This place yields a concurrency having the following set of markings:

$$M(C(x_4)) = \{M_{34}, M_{37}, M_{40}, M_{43}, M_{47}, M_{50}\}$$

(iv) Concurrency sets for places that represent the interaction between the controller and slider in the physical system

(a) Place y_7 represents the signal which provides an input to the controller indicating that the slider is at the decision point. This place yields a concurrency having the following set of markings:

$$M(C(y_7)) = \{M_1, M_8, M_{15}, M_{24}, M_{35}, M_{36}, M_{37}\}$$

(b) Place y_8 represents the signal which provides an output to the slider indicating that it must abort the insert motion. This place yields a concurrency having the following set of markings:

$$M(C(y_8)) = \{M_9, M_{16}, M_{22}, M_{28}, M_{30}, M_{44}, M_{46}, M_{47}\}$$

(c) Place y_9 represents the signal which provides an output to the slider indicating that it must commence its motion to insert into the drum. This place yields a concurrency having the following set of markings:

$$M(C(y_9)) = \{M_9, M_{10}, M_{16}, M_{17}, M_{22}, M_{23}\}$$

(d) Place y_{10} represents the signal which provides an input to the controller indicating that the slider has cleared the drum. This place yields a concurrency having the following set of markings:

$$M(C(y_{10})) = \{M_6, M_7, M_8, M_{13}, M_{14}, M_{15}, M_{19}, M_{20}, M_{21}\}$$

(e) Place y_{11} represents the signal which provides an output to the slider indicating that it must insert into the drum. This place yields a concurrency having the following set of markings:

$$M(C(y_9)) = \{M_4, M_{11}\}$$

(v) Concurrency sets for places that represent the the motion of the slider in the physical system

(a) Place y_1 represents the slider at rest. This place yields a concurrency having the following set of markings:

$$M(C(y_1)) = \{M_0, M_7, M_{14}, M_{20}, M_{25}, M_{32}, M_{33}, M_{34}\}$$

(b) Place y_2 represents the slider at the decision point. This place yields a concurrency having the following set of markings:

$$M(C(y_2)) = \{M_1, M_3, M_4, M_8, M_9, M_{11}, M_{15}, M_{16}, M_{21}, M_{22}, M_{24}, M_{27}, M_{28}, M_{30}, M_{35}, M_{36}, M_{37}, M_{41}, M_{42}, M_{43}, M_{44}, M_{46}, M_{47}\}$$

(c) Place y_3 represents the slider at rest after an abort motion. This place yields a concurrency having the following set of markings:

$$M(C(y_3)) = \{M_{10}, M_{17}, M_{23}, M_{29}, M_{31}, M_{48}, M_{49}, M_{50}\}$$

(d) Place y_4 represents the slider has inserted into the drum. This place yields a concurrency having the following set of markings:

$M(C(y_4)) = \{M_5, M_{12}, M_{18}\}$

(e) Place y_5 represents the slider has cleared the drum. This place yields a concurrency having the following set of markings:

$M(C(y_5)) = \{M_2, M_6, M_{13}, M_{19}, M_{26}, M_{38}, M_{39}, M_{40}\}$

A comparison of the markings obtained from Table D.1 and the markings obtained from the concurrency sets of Figure 7.2 shows that both sets of markings are the same. Hence, it can be concluded from the above comparison that the concurrency set of Table 7.2 is the C_{min} of Figure 7.2.

D.2 Markings for the Petri net model of Figure 7.6

The following shows the remaining inconsistent and hazardous markings that were detected in Table 7.4 of Figure 7.6.

(i) For the controller marking $MC_1 = (p_2, p_{15}, p_{14})$, which assumes that the slider is at the decision point and the drum is stationary, the inconsistent markings are:

(a) $\{y_2, MC_1, (x_1, x_3)\}$ (b) $\{y_2, MC_1, x_2\}$

These markings are inconsistent because, in the physical system, the drum is either rotate enabled or rotating, but the controller assumes that the drum is stationary.
Markings of (a) and (b) are also hazardous because: the slider is at the decision point and will make an insertion into the drum, while the drum is rotating or about to rotate.

(ii) For controller marking $MC_2 = (p_4, p_{14})$ the inconsistent markings are:

(a)	$\{(y_2, y_8, y_9), MC_2, (x_1, x_3)\}$	(h)	$\{(y_2, y_8, y_9), MC_2, x_2\}$
(b)	$\{(y_3, y_9), MC_2, (x_1, x_3)\}$	(i)	$\{(y_3, y_9), MC_2, x_2\}$
(c)	$\{(y_2, y_{11}), MC_2, (x_1, x_3)\}$	(j)	$\{(y_2, y_{11}), MC_2, x_2\}$
(d)	$\{y_2, MC_2, (x_1, x_3)\}$	(k)	$\{y_2, MC_2, x_2\}$
(e)	$\{y_1, MC_2, (x_1, x_3)\}$	(l)	$\{y_1, MC_2, x_2\}$
(f)	$\{y_5, MC_2, (x_1, x_3)\}$	(m)	$\{y_5, MC_2, x_2\}$
(g)	$\{y_4, MC_2, (x_1, x_3)\}$	(n)	$\{y_4, MC_2, x_2\}$

The markings of (a) - (n) are inconsistent because the controller assumes that the drum is stationary, but in the physical system, the drum is either rotating or rotate enabled.

Moreover, markings of (a) - (g) and (h) - (n) are also inconsistent because the controller assumes that the slider has inserted into the drum, however, the slider in the physical system has not inserted into the drum. From the above inconsistent markings the following are hazardous:

Markings (b), (c), (i) and (j) show that a collision of the mechanisms will occur in the subsequent markings, and markings of (g) and (n) show that a collision of the mechanisms has occurred. As well as the hazardous nature of the above markings, the marking of (e) and markings $\{y_1, MC_2, x_1\}$, $\{y_1, MC_2, x_2\}$ will deadlock.

(iii) For controller marking $MC_3 = (p_5, p_{14})$ the inconsistent markings are:

(a)	$\{(y_2, y_8, y_9), MC_3, (x_1, x_3)\}$	(h)	$\{(y_2, y_8, y_9), MC_3, x_2\}$
(b)	$\{(y_3, y_9), MC_3, (x_1, x_3)\}$	(i)	$\{(y_3, y_9), MC_3, x_2\}$
(c)	$\{(y_2, y_{11}), MC_3, (x_1, x_3)\}$	(j)	$\{(y_2, y_{11}), MC_3, x_2\}$
(d)	$\{y_2, MC_3, (x_1, x_3)\}$	(k)	$\{y_2, MC_3, x_2\}$
(e)	$\{y_1, MC_3, (x_1, x_3)\}$	(l)	$\{y_1, MC_3, x_2\}$
(f)	$\{y_5, MC_3, (x_1, x_3)\}$	(m)	$\{y_4, MC_3, x_2\}$
(g)	$\{y_4, MC_3, (x_1, x_3)\}$	(n)	$\{y_5, MC_3, x_2\}$

The above markings of (a) - (n) are inconsistent and hazardous for the same reasons as those of markings (ii)(a) - (n). Moreover, the following markings deadlock: marking of (e) and $\{y_1, MC_3, x_1\}$, $\{y_1, MC_3, (x_1, x_3)\}$.

(iv) For controller marking $MC_{10} = (p_8, p_{14})$ the inconsistent markings are:

(a)	$\{(y_2, y_8, y_9), MC_{10}, (x_1, x_3)\}$	(h)	$\{(y_2, y_8, y_9), MC_{10}, x_2\}$
(b)	$\{(y_3, y_9), MC_{10}, (x_1, x_3)\}$	(i)	$\{(y_3, y_9), MC_{10}, x_2\}$
(c)	$\{(y_2, y_{11}), MC_{10}, (x_1, x_3)\}$	(j)	$\{(y_2, y_{11}), MC_{10}, x_2\}$
(d)	$\{y_2, MC_{10}, (x_1, x_3)\}$	(k)	$\{y_2, MC_{10}, x_2\}$
(e)	$\{y_1, MC_{10}, (x_1, x_3)\}$	(l)	$\{y_1, MC_{10}, x_2\}$
(f)	$\{y_5, MC_{10}, (x_1, x_3)\}$	(m)	$\{y_5, MC_{10}, x_2\}$
(g)	$\{y_4, MC_{10}, (x_1, x_3)\}$	(n)	$\{y_4, MC_{10}, x_2\}$

Again the above markings are inconsistent and hazardous for the same reasons as explained for the markings of (i), (ii), (iv). Also markings that deadlock are: $\{y_1, MC_{10}, x_1\}$, $\{y_1, MC_{10}, x_2\}$ and the marking of (e).

Appendix E

The following shows the proof of safety, liveness and real-time properties presented in Chapter 8.

Property S3

$$(ty_2(ok) \vee ty_4(ok)) \Rightarrow \Diamond[t_{13}(c)\ \mathbf{P}\ (ty_2(ok) \vee ty_4(ok))]$$

Proof

When ty_2 or ty_4 completes firing place y_4 becomes tokenised. The possible global marking of the net (Figure 7.2) at this instant can be inferred from $C(y_4)$, Table 7.2, as follows:

$M_1 = \{y_4, p_8, p_{14}, x_1\}$
$M_2 = \{y_4, p_4, p_{14}, x_1\}$
$M_3 = \{y_2, p_5, p_{14}, x_1\}$

Using the above information it can be deduced that:

1. $(ty_2(ok) \vee ty_4(ok)) \Rightarrow O(M_1 \vee M_2 \vee M_3)$

For markings M_1, M_1, and M_3 it can be deduced that:

2. $(M_1 \vee M_2 \vee M_3) \Rightarrow \Diamond t_6(c)$

When t_6 completes firing places p_{16} and p_9 become tokenised. Therefore:

3. $t_6(c) \Rightarrow (p_9 \wedge p_{16})$

Simplifying 3. yields:

4. $t_6(c) \Rightarrow p_9$

The enabling condition of t_{13} is:

5. $(p_9 \wedge p_{12} \wedge x_4) \equiv t_{13}(ok)$

Hence from 5. it can be deduced that when p_9 is tokenised t_{13} can not complete firing hence:

6. $p_9 \Rightarrow \neg t_{13}(c)$

Moreover if t_{13} can not complete firing then p_{15} cannot be tokenised:

7. $\neg t_{13}(c) \Rightarrow \neg p_{15}$

From 7. if p_{15} is not tokenised then t_2 or t_8 cannot be enabled, since the tokenisation of p_{15} is necessary for t_2 or t_8 to be enabled. Hence:

8. $\neg p_{15} \Rightarrow \neg(ty_2(ok) \vee ty_8(ok))$

If t_2 or t_8 cannot be enabled then it follows that ty_2 or ty_4 cannot become firable:

9. $\neg(ty_2(ok) \vee ty_8(ok)) \Rightarrow \neg(ty_2(ok) \vee ty_4(ok))$

Combining 6. - 9. yields:

10. $p9 \Rightarrow \neg(ty_2(ok) \vee ty_4(ok))$

From the concurrency set of Table 7.2 the following is valid:

11. $\neg t_{13}(c) \wedge p9 \Rightarrow Op9$

10. and 11. form the antecedent of Equ. 5.34, hence it can be deduced that:

12. $p9 \Rightarrow t_{13}(c)\, \mathbf{P}\, (ty_2(ok) \vee ty_4(ok))$

Combining 12., 4., 2. and 1. yields:

13. $(ty_2(ok) \vee ty_4(ok)) \Rightarrow \Diamond[t_{13}(c)\, \mathbf{P}\, (ty_2(ok) \vee ty_4(ok))]$ ■

Property S4

$$\Box\neg[y2 \wedge \neg(p9 \vee p15)]$$

Proof

Assuming the negation of S4 yields and using the definition $\neg\Box\neg w \equiv \Diamond w$, where w is any well-formed propositional formula, then S4 can be written as:

1. $\Diamond[y2 \wedge \neg(p9 \vee p15)]$

From figure 8.2., applying each applicable inference rule to M_0, it can be deduced that:

2. $\neg[y2 \wedge \neg(p9 \vee p15)]$

From the temporal proof system the following theorem is valid:

3. $w \wedge \Diamond\neg w \Rightarrow \Diamond(w \wedge O\neg w)$

The conjunction of 1. and 2. yields:

4. $\Diamond[y2 \wedge \neg(p9 \vee p15)] \wedge \neg[y2 \wedge \neg(p9 \vee p15)]$

Applying Modus ponens to 3. and 4.(where $\neg[y2 \wedge \neg(p9 \vee p15)]$is substituted for w) the following results:

5. $\Diamond\{\neg[y2 \wedge \neg(p9 \vee p15)] \wedge O[y2 \wedge \neg(p9 \vee p15)]\}$

Simplyfying the first disjunct of 5. using De Morgans theorem yields:

6. $\Diamond\{\neg[y2 \wedge \neg(p9 \vee p15)] \wedge O[y2 \wedge \neg(p9 \vee p15)]$

Expanding 6. yields:

7. $\Diamond\{\neg[y2 \wedge O[y2 \wedge \neg(p9 \vee p15)]] \vee p15) \wedge O[y2 \wedge \neg(p9 \vee p15)]]\}$

Examining each disjunct of 7.in turn

8. $\neg[y2 \wedge O[y2 \wedge \neg(p9 \vee p15)]$	first disjunct of 7.
9. $y2 \wedge O.y2$	by weakening 8.
10. $O\neg(p9 \vee p15)]$	by weakening 8.

Proof of 8. is established by selecting the system dependent inference rule (or rules) of Figure 7.2. which show the sequence of 9., and examining it to see if 10. holds true. Therefore, the inference rule applicable to 9. is:

$$y_1 \wedge \neg y_2 \wedge \neg y_7 \Rightarrow O(\neg y_1 \wedge \neg y_2 \wedge y_7)$$

To check if this inference rule contains 10., it is sufficient to examine the concurrency set of the consequence of this rule, i.e. y_2 and y_7 to infer whether p_9 or p_{15} are tokenised. Note the semantics of y_2 and y_7 represents the situation when the slider is at the decision and has signalled its position to the controller. From Table 7.2:

$$\begin{aligned} C(y_2,y_7) = \{ & (p_{16}, p_{15}, p_{14}, x_1), (p_4, p_{14}, x_1, y_{10}), (p_5, p_{14}, x_1, y_{10}) \\ & (p_8, p_{14}, x_1, y_{10}), (p_{16}, p_9, p_{14}, x_1), (p_{16}, p_9, p_{12}, x_1, x_3) \\ & (p_{16}, p_9, p_{12}, x_2), (p_{16}, p_9, p_{12}, x_1, x_4)\} \end{aligned} \qquad \text{E.1}$$

From $C(y_2,y_7)$ it is seen that p_9 or p_{15} are tokenised in all markings except for markings:

$M_1 = \{y_2, y_7, y_{10}, p_8, p_{14}, x_1\}$
$M_2 = \{y_2, y_7, y_{10}, p_4, p_{14}, x_1\}$
$M_3 = \{y_2, y_7, y_{10}, p_5, p_{14}, x_1\}$

in which holds true $\neg(p_9 \vee p_{15})$. Thus without applying reasoning to the second disjunct of 7. it can be deduced that under certain markings both 7. and 1. can hold true, which falsifies S4. The implications of this result must be determined by examining its effect on the safety aspects of Figure 7.2. If S4 is false, then a situation will occur when the slider is at the decision point but is unable to decide whether to insert or abort; this result can also be inferred by examining the semantics of the sequence $\{M_1, M_2, M_3\}$. In this 'state' a collision of the mechanisms could occur but is dependent upon when the slider enters the decision point. For instance:

(i) if the Petri net model executes the marking sequence:

$\{M_1, M_2, M_3, M_4\}$ or $\{M_2, M_3, M_4\}$

where $M_4 = \{y_2, y_7, p_{16}, p_9, p_{12}, x_1, x_3\}$

whilst the slider has reached the end of it's decision point then a collision will occur,

(ii) however, if the sequences detailed in (i) are executed whilst the slider is within ts decision point then a collision will not occur.

Hence, the occurrence of a collision is dependent upon the response time of the controller to the physical system. More specifically, if the slider enters the decision point and the controller is at submarking then the following time constraint must be satisfied by the controller:

$$\delta ey_2 > \delta_9 + \delta_5 + \delta_6 \qquad \text{where } \delta_9 = \delta e_9 + \delta f_9$$
$$\delta_5 = \delta e_5 + \delta f_5$$
$$\delta_6 = \delta e_6 + \delta f_6$$

Thus it can be concluded that property S4 is not valid and as a consequence is not an invariant property of the model, and it has safety implications.

Property L1

$$<M_0, \alpha> \models \Box[t_{x2}(c) \Rightarrow \Diamond(t_{y2}(ok) \vee t_{y4}(ok))]$$

Proof

When t_{x2} completes firing a token is removed from its input place (x_2) and deposited in its output places (x_1 and x_4). The global marking of the net (Figure 7.2) when x_1 and x_4 are tokenised can be inferred from $C(x_1,x_4)$ which can be easily obtained from Table 7.2; hence the markings from $C(x_1,x_4)$ are:

$M_1 = \{y_5, p_{16}, p_9, p_{12}, x_1, x_4\}$ $\quad$ $M_2 = \{y_1, p_{16}, p_9, p_{12}, x_1, x_4\}$

$M_3 = \{y_2, y_7, p_{16}, p_9, p_{12}, x_1, x_4\}$ $\quad$ $M_4 = \{y_2, p_2, p_9, p_{12}, x_1, x_4\}$

$M_5 = \{y_2, y_8, p_7, p_9, p_{12}, x_1, x_4\}$ $\quad$ $M_6 = \{y_3, p_2, p_9, p_{12}, x_1, x_4\}$

From the above it can be deduced that:

1. $t_{x2}(c) \Rightarrow (M_1 \vee M_2 \vee M_3 \vee M_4 \vee M_5 \vee M_6)$

The firable transitions in each marking must be considered. For example, in M_1 transitions t_{y6} and t_{13} are firable and can fire independently. Hence, the following can be deduced:

2. $M_1 \wedge t_{y6}(ok) \Rightarrow O(M_2 \vee t_{13}(c))$

From 2. considering the firable transitions in M_2:

3. $M_2 \wedge t_{y1}(ok) \Rightarrow O(M_3 \vee t_{13}(c))$

For M_3 the following can be written:

4. $M_3 \wedge t_1(ok) \Rightarrow O(M_4 \vee t_{13}(c))$

M_4 yields, where $M_7 = \{y_2, p_2, p_5, p_{14}, x_1\}$:

5. $M_4 \wedge t_{13}(ok) \Rightarrow OM_7$

M_7 yields, where $M_8 = \{y_2, y_{11}, p_4, p_{14}, x_1\}$:

6. $M_7 \wedge t_2(ok) \Rightarrow OM_8$

In M_8 the firable transitions are t_{y2} and t_5, therefore:

7. $M_8 \Rightarrow (t_{y2}(ok) \vee t_5(ok))$

Combining 2. - 7. and simplifying yields:

8. $M_1 \Rightarrow \Diamond ty2(ok)$

Applying the procedure of 2. - 7. to M_2 - M_4 in 1. also yields the result of 8. For M_5 and M_6 the following can be deduced:

9. $M_5 \wedge ty3(ok) \Rightarrow O(M_6 \vee t13(c))$

For M_6 in 9., where $M_9 = \{y3, p7, p14, p15, x1\}$:

10. $M_6 \wedge t13(ok) \Rightarrow OM_9$

For M_9 in 10., where $M_{10} = \{y3, y9, p8, p14, x1\}$:

11. $M_9 \wedge t8(ok) \Rightarrow OM_{10}$

The firable transitions in M_{10} are:

12. $M_{10} \Rightarrow (ty4(ok) \vee t9(ok))$

Combining 9. - 12. and simplifying yields:

13. $M_5 \Rightarrow \Diamond ty4(ok)$

For marking M_6 the same consequence of 13. results. Hence combining results for M_1 - M_4 using 8., M_5 - M_6 using 13. with 1., the following can be deduced:

14. $t_x2(c) \Rightarrow \Diamond(ty2(ok) \vee ty4(ok))$ ■

Property L2

$$<M_0, \alpha> \models \Box[ty1(c) \Rightarrow \Diamond(ty2(ok) \vee ty4(ok))]$$

Proof

When ty1 completes firing places y2 and y7 become tokenised; the global marking of the net (Figure 7.2) at this instantn can be inferred from C(y2,y7). From C(x1,x4) the global markings are:

$M_1 = \{y2, y7, p14, p15, p16, x1\}$	$M_5 = \{y2, y7, p9, p14, p16, x1\}$
$M_2 = \{y2, y7, y10, p8, p14, x1\}$	$M_6 = \{y2, y7, p9, p12, p16, x1, x3\}$
$M_4 = \{y2, y7, y10, p4, p14, x1\}$	$M_7 = \{y2, y7, p9, p12, p16, x2\}$
$M_4 = \{y2, y7, y10, p5, p14, x1\}$	$M_8 = \{y2, y7, p9, p12, p16, x1, x4\}$

From the above it can be deduced that:

1. $ty1(c) \Rightarrow (M_1 \vee M_2 \vee M_3 \vee M_4 \vee M_5 \vee M_6 \vee M_7 \vee M_8)$

Considering the firable transitions in markings M_1 - M_8 the following can be deduced. For M_1 transition t_1 becomes firable and when it fires the marking becomes $M_9 = \{y2, p2, p14, p15, x1\}$:

2. $M_1 \wedge t_1(\text{ok}) \Rightarrow O(y_2 \vee p_2 \vee p_{14} \vee p_{15} \vee x_1)$

Simplifying 2. yields:

3. $M_1 \Rightarrow Op_2$

Considering all sequent firing of transitions in M_2 - M_8 it can be deduced using similar reasoning to 2. - 3. that:

4. $(M_2 \vee M_3 \vee M_4 \vee M_5 \vee M_6 \vee M_7 \vee M_8) \Rightarrow \Diamond p_2$

Combining 3. and 4. :

5. $(M_1 \vee M_2 \vee M_3 \vee M_4 \vee M_5 \vee M_6 \vee M_7 \vee M_8) \Rightarrow \Diamond p_2$

From property 8.4.1.(a)(2) the following holds true for the controller:

6. $p_2 \Rightarrow \Diamond(t_2(c) \vee t_3(c))$

If t_2 fires then the next firable transitions are t_5 and t_{y2}, therefore:

7. $t_2(c) \Rightarrow O(t_{y2}(\text{ok}) \vee t_5(\text{ok}))$

Simplifying 7. yields:

8. $t_2(c) \Rightarrow Ot_{y2}(\text{ok})$

However, if t_3 completes firing then eventually t_8 must become firable:

9. $t_3(c) \Rightarrow \Diamond t_8(\text{ok})$

If t_8 completes firing then t_{y4} must become firable:

10. $t_8(c) \Rightarrow \Diamond t_{y4}(\text{ok})$

Combining 9. - 10. yields:

11. $t_3(c) \Rightarrow \Diamond t_{y4}(\text{ok})$

Combining 6., 8. and 11. yields:

12. $p_2 \Rightarrow \Diamond(t_{y4}(\text{ok}) \vee t_9(\text{ok}))$

Combining 5. and 12. and then 1. yields:

13. $t_{y1}(c) \Rightarrow \Diamond(t_{y4}(\text{ok}) \vee t_9(\text{ok}))$ ■

Property RL2

$$(y_2 \wedge (\tau = 0)) \Rightarrow \Diamond\{[(y_{11} \wedge p_4 \wedge p_{14}) \wedge (\tau = \delta_{\mathbf{x1}})] \vee [p_7 \wedge (\delta_3 \leq \tau = \delta_{\mathbf{x2}})]\}$$

Proof

The net markings existing when y_2 is tokenised can be easily deduced from $C(y_2)$ and are as follows:

$M_1 = \{y_2, y_7, p_{16}, p_{15}, p_{14}, x_1\}$	$M_{13} = \{y_2, y_8, p_7, p_9, p_{14}, x_1\}$
$M_2 = \{y_2, p_2, p_{14}, p_{15}, x_1\}$	$M_{14} = \{y_2, y_8, p_7, p_{14}, p_{15}, x_1\}$
$M_3 = \{y_2, y_{11}, p_4, p_{14}, x_1\}$	$M_{15} = \{y_2, y_7, p_{16}, p_9, p_{12}, x_1, x_3\}$
$M_4 = \{y_2, y_8, y_9, p_4, p_{14}, x_1\}$	$M_{16} = \{y_2, y_7, p_{16}, p_9, p_{12}, x_2\}$

$M_5 = \{y_2, y_7, y_{10}, p_4, p_{14}, x_1\}$

$M_6 = \{y_2, y_8, y_9, p_5, p_{14}, x_1\}$

$M_7 = \{y_2, y_7, y_{10}, p_5, p_{14}, x_1\}$

$M_8 = \{y_2, y_{11}, p_5, p_{14}, x_1\}$

$M_9 = \{y_2, y_8, y_9, p_8, p_{14}, x_1\}$

$M_{10} = \{y_2, y_7, y_{10}, p_8, p_{14}, x_1\}$

$M_{11} = \{y_2, y_7, p_{16}, p_9, p_{14}, x_1\}$

$M_{12} = \{y_2, p_2, p_9, p_{14}, x_1\}$

$M_{17} = \{y_2, y_7, p_{16}, p_9, p_{12}, x_1, x_4\}$

$M_{18} = \{y_2, p_2, p_9, p_{12}, x_1, x_3\}$

$M_{19} = \{y_2, p_2, p_9, p_{12}, x_2\}$

$M_{20} = \{y_2, p_2, p_9, p_{12}, x_1, x_4\}$

$M_{21} = \{y_2, y_8, p_7, p_9, p_{12}, x_1, x_3\}$

$M_{22} = \{y_2, y_8, p_7, p_9, p_{12}, x_2\}$

$M_{23} = \{y_2, y_8, p_7, p_9, p_{12}, x_1, x_4\}$

From these markings all marking sequences that lead to (p_3, p_{14}) or p_7 are considered; Figure E.1 shows such sequences, however, markings sequences of $\{M_3, M_8\}$ and $\{M_{23}, M_{14}, M_{19}, M_4, M_6\}$ are not considered because they represent the transitions within the goal places, ie (p_3, p_{14}) or p_7.

Figure E.1 Partial reachability graph for the proof of RL1

$$\begin{tikzcd}
M_{10} \arrow[d, "t_9"] & & \\
M_5 \arrow[d, "t_5"] & & \\
M_7 \arrow[d, "t_6"] & & \\
M_{11} \arrow[r, "t_1"] \arrow[d, "t_{10}"] & M_{12} \arrow[r, "t_3"] \arrow[d, "t_{10}"] & M_{13} \\
M_{15} \arrow[r, "t_1"] \arrow[d, "t_{x1}"] & M_{18} \arrow[r, "t_3"] \arrow[d, "t_{x1}"] & M_{21} \\
M_{16} \arrow[r, "t_1"] \arrow[d, "t_{x2}"] & M_{19} \arrow[r, "t_3"] \arrow[d, "t_{x2}"] & M_{22} \\
M_{17} \arrow[r, "t_1"] \arrow[d, "t_{13}"] & M_{20} \arrow[r, "t_3"] \arrow[d, "t_{13}"] & M_{23} \\
M_1 \arrow[r, "t_1"] & M_2 \arrow[r, "t_2"] & M_3
\end{tikzcd}$$

MARKING	PLACES
M_1	$y_2, y_7, p_{16}, p_{14}, p_{15}, x_1$
M_2	$y_2, p_{14}, p_2, p_{15}, x_1$
M_3	$y_2, y_{11}, p_4, p_{14}, x_1$
M_5	$y_2, y_7, y_{10}, p_4, p_{14}, x_1$
M_7	$y_2, y_7, y_{10}, p_5, p_{14}, x_1$
M_{10}	$y_2, y_7, y_{10}, p_8, p_{14}, x_1$
M_{11}	$y_2, y_7, p_{16}, p_9, p_{14}, x_1$
M_{12}	$y_2, p_2, p_9, p_{14}, x_1$
M_{13}	$y_2, y_8, p_7, p_9, p_{14}, x_1$

MARKING	PLACES
M_{15}	$y_2, y_7, p_{16}, p_9, p_{12}, x_1, x_3$
M_{16}	$y_2, y_7, p_{16}, p_9, p_{12}, x_2$
M_{17}	$y_2, y_7, p_{16}, p_9, p_{12}, x_1, x_4$
M_{18}	$y_2, p_2, p_9, p_{12}, x_1, x_3$
M_{19}	$y_2, p_2, p_9, p_{12}, x_2$
M_{20}	$y_2, p_2, p_9, p_{12}, x_1, x_4$
M_{21}	$y_2, y_8, p_7, p_9, p_{12}, x_1, x_3$
M_{22}	$y_2, y_8, p_7, p_9, p_{12}, x_2$
M_{23}	$y_2, y_8, p_7, p_9, p_{12}, x_1, x_4$

Using Figure E.1 and Formula 5.33 all marking sequences are considered that start from the orgin place (y_2) and end at the goal places (y_{11}, p_4, p_{14}) or p_7. The following illustrates formulae for the firing sequence $\{t_9, t_5, t_6, t_{10}, t_{x1}, t_{x2}, t_{13}, t_1, t_2\}$. Hence, consider the firing of t_9:

1. $M_{10} \wedge t_9(ok) \wedge \tau = \mathcal{T}_1 \Rightarrow M_{10}\ \mathcal{U}\ (M_5 \wedge (\tau = \mathcal{T}_1 + \delta_9))$
   where $\delta_9 = \delta e_9 + \delta f_9$

Firing of t_5:

2. $M_5 \wedge t_5(ok) \wedge \tau = \mathcal{T}_2 \Rightarrow M_5\ \mathcal{U}\ (M_7 \wedge (\tau = \mathcal{T}_2 + \delta_5))$
   where $\delta_5 = \delta e_5 + \delta f_5$ and $\mathcal{T}_2 = \mathcal{T}_1 + \delta_9$

Firing of t_6:

3. $M_7 \wedge t_6(ok) \wedge \tau = \mathcal{T}_3 \Rightarrow M_7\ \mathcal{U}\ (M_{13} \wedge (\tau = \mathcal{T}_3 + \delta_6) \vee \neg I(t_6))$
   where $\delta_6 = \delta e_6 + \delta f_6$ and $\mathcal{T}_3 = \mathcal{T}_2 + \delta_5$

Firing of t_{10}:

4. $M_{11} \wedge t_{10}(\mathrm{ok}) \wedge \tau = \mathcal{T}_4 \Rightarrow M_{11}\ \mathcal{U}\ (M_{15} \wedge (\tau = \mathcal{T}_4 + \delta_{10}) \vee \neg I(t_{10}))$

   where $\delta_{10} = \delta_{e10} + \delta_{f10}$ and $\mathcal{T}_4 = \mathcal{T}_3 + \delta_6$

Firing of t_{x1}:

5. $M_{15} \wedge t_{x1}(\mathrm{ok}) \wedge \tau = \mathcal{T}_5 \Rightarrow M_{15}\ \mathcal{U}\ (M_{16} \wedge (\tau = \mathcal{T}_5 + \delta_{x1}) \vee \neg I(t_{x1}))$

   where $\delta_{x1} = \delta_{ex1} + \delta_{fx1}$ and $\mathcal{T}_5 = \mathcal{T}_4 + \delta_{10}$

Firing of t_{x2}:

6. $M_{16} \wedge t_{x2}(\mathrm{ok}) \wedge \tau = \mathcal{T}_6 \Rightarrow M_{16}\ \mathcal{U}\ (M_{17} \wedge (\tau = \mathcal{T}_6 + \delta_{x2}) \vee \neg I(t_{x2}))$

   where $\delta_{x2} = \delta_{ex2} + \delta_{fx2}$ and $\mathcal{T}_6 = \mathcal{T}_5 + \delta_{x1}$

Firing of t_{13}:

7. $M_{17} \wedge t_{13}(\mathrm{ok}) \wedge \tau = \mathcal{T}_7 \Rightarrow M_{17}\ \mathcal{U}\ (M_1 \wedge (\tau = \mathcal{T}_7 + \delta_{13}) \vee \neg I(t_{13}))$

   where $\delta_{13} = \delta_{e13} + \delta_{f13}$ and $\mathcal{T}_7 = \mathcal{T}_6 + \delta_{x2}$

Firing of t_1:

8. $M_1 \wedge t_1(\mathrm{ok}) \wedge \tau = \mathcal{T}_8 \Rightarrow M_1\ \mathcal{U}\ (M_2 \wedge (\tau = \mathcal{T}_8 + \delta_1) \vee \neg I(t_1))$

   where $\delta_1 = \delta_{e1} + \delta_{f1}$ and $\mathcal{T}_8 = \mathcal{T}_7 + \delta_{13}$

Firing of t_2:

9. $M_2 \wedge t_2(\mathrm{ok}) \wedge \tau = \mathcal{T}_7 \Rightarrow M_2\ \mathcal{U}\ (M_3 \wedge (\tau = \mathcal{T}_7 + \delta_2) \vee \neg I(t_2))$

   where $\delta_2 = \delta_{e2} + \delta_{f2}$ and $\mathcal{T}_7 = \mathcal{T}_6 + \delta_1$

Combining 1. -9. yields the formula describing the firing sequence $\{t_9, t_5, t_6, t_{10}, t_{x1}, t_{x2}, t_{13}, t_1, t_2\}$ as:

10. $M_{10} \wedge t_9(\mathrm{ok}) \wedge \tau = \mathcal{T}_1 \Rightarrow (M_{10} \vee M_5 \vee M_7 \vee M_{11} \vee M_{15} \vee M_{16} \vee M_{17} \vee$
    $M_1 \vee M_2)\ \mathcal{U}\ ((M_3 \wedge (\tau = \mathcal{T}_1 + \delta_{T1})))$

    where $\delta_{T1} = \delta_9 + \delta_5 + \delta_6 + \delta_{10} + \delta_{x1} + \delta_{x2} + \delta_{13} + \delta_1 + \delta_2$

Formulising other marking sequence of Figure E.1 yields:

11. $M_{10} \wedge t_9(\mathrm{ok}) \wedge \tau = \mathcal{T}_1 \Rightarrow (M_{10} \vee M_5 \vee M_7 \vee M_{11} \vee M_{12} \vee M_{13})\ \mathcal{U}$
    $((M_{13} \wedge (\tau = \mathcal{T}_1 + \delta_{T2}))) \vee$

    $M_{10} \wedge t_9(\mathrm{ok}) \wedge \tau = \mathcal{T}_1 \Rightarrow (M_{10} \vee M_5 \vee M_7 \vee M_{11} \vee M_{15} \vee M_{12} \vee M_{18})\ \mathcal{U}$
    $((M_{21} \wedge (\tau = \mathcal{T}_1 + \delta_{T3}))) \vee$

    $M_{10} \wedge t_9(\mathrm{ok}) \wedge \tau = \mathcal{T}_1 \Rightarrow (M_{10} \vee M_5 \vee M_7 \vee M_{11} \vee M_{15} \vee M_{16} \vee M_{12} \vee$
    $M_{18} \vee M_{19})\ \mathcal{U}\ ((M_{22} \wedge (\tau = \mathcal{T}_1 + \delta_{T4}))) \vee$

    $M_{10} \wedge t_9(\mathrm{ok}) \wedge \tau = \mathcal{T}_1 \Rightarrow (M_{10} \vee M_5 \vee M_7 \vee M_{11} \vee M_{15} \vee M_{16} \vee M_{17} \vee$
    $M_{12} \vee M_{20} \vee M_{18} \vee M_{19})\ \mathcal{U}\ ((M_{23} \wedge (\tau = \mathcal{T}_1 + \delta_{T5}))) \vee$

    $M_{10} \wedge t_9(\mathrm{ok}) \wedge \tau = \mathcal{T}_1 \Rightarrow (M_{10} \vee M_5 \vee M_7 \vee M_{11} \vee M_{15} \vee M_{16} \vee M_{17} \vee$
    $M_{12} \vee M_{20} \vee M_{18} \vee M_{19})\ \mathcal{U}\ ((M_3 \wedge (\tau = \mathcal{T}_1 + \delta_{T6})))$

    where $\delta_{T2} = \delta_9 + \delta_5 + \delta_6 + \delta_1 + \delta_3$
    $\delta_{T3} = \delta_9 + \delta_5 + \delta_6 + \delta_{10} + \delta_1 + \delta_3$
    $\delta_{T4} = \delta_9 + \delta_5 + \delta_6 + \delta_{10} + \delta_{x1} + \delta_1 + \delta_3$
    $\delta_{T5} = \delta_9 + \delta_5 + \delta_6 + \delta_{10} + \delta_{x1} + \delta_{x2} + \delta_1 + \delta_3$

$$\delta_{T6} = \delta_9 + \delta_5 + \delta_6 + \delta_{10} + \delta_{x1} + \delta_{x2} + \delta_{13} + \delta_2$$

In 10. and 11. it can be seen that for sequences which lead to M_3 that t_2 does not complete firing and for sequences leading to $\{M_{13}, M_{21}, M_{22}, M_{23}\}$ t_3 does not complete firing. Therfore, combining 10. and 11. and simplifying yields:

12. $M_{10} \wedge t9(ok) \wedge \tau=\mathcal{T}_1 \Rightarrow \neg t2(c)\ \mathcal{U}\ ((M_{13} \wedge (\tau = \mathcal{T}_1 + \delta_{T2}))) \vee$
    $M_{10} \wedge t9(ok) \wedge \tau=\mathcal{T}_1 \Rightarrow \neg t3(c)\ \mathcal{U}\ ((M_{21} \wedge (\tau = \mathcal{T}_1 + \delta_{T3}))) \vee$
    $M_{10} \wedge t9(ok) \wedge \tau=\mathcal{T}_1 \Rightarrow \neg t3(c)\ \mathcal{U}\ ((M_{22} \wedge (\tau = \mathcal{T}_1 + \delta_{T4}))) \vee$
    $M_{10} \wedge t9(ok) \wedge \tau=\mathcal{T}_1 \Rightarrow \neg t3(c)\ \mathcal{U}\ ((M_{23} \wedge (\tau = \mathcal{T}_1 + \delta_{T5}))) \vee$
    $M_{10} \wedge t9(ok) \wedge \tau=\mathcal{T}_1 \Rightarrow \neg t2(c)\ \mathcal{U}\ ((M_3 \wedge (\tau = \mathcal{T}_1 + \delta_{T6})))$

Simplifying 12 yields:

13. $M_{10} \wedge t9(ok) \wedge \tau=\mathcal{T}_1 \Rightarrow \neg t2(c)\ \mathcal{U}\ (((p4 \wedge p14 \wedge y11) \wedge$
    $(\mathcal{T}_1 + \delta_{T6}) \leq \tau \leq (\mathcal{T}_1 + \delta_{T1}))) \vee$
    $M_{10} \wedge t9(ok) \wedge \tau=\mathcal{T}_1 \Rightarrow \neg t3(c)\ \mathcal{U}\ ((p7 \wedge (\mathcal{T}_1 + \delta_{T2}) \leq \tau \leq (\mathcal{T}_1 + \delta_{T5})))$

Further simplification of 13. and assuming $\mathcal{T}_1 = 0$:

14. $y2 \wedge t9(ok) \wedge \tau=\mathcal{T}_1 \Rightarrow \Diamond(((p4 \wedge p14 \wedge y11) \wedge (\mathcal{T}_1 + \delta_{T6}) \leq \tau \leq (\mathcal{T}_1 + \delta_{T1}))) \vee$
    $y2 \wedge t9(ok) \wedge \tau=\mathcal{T}_1 \Rightarrow \Diamond((p7 \wedge (\mathcal{T}_1 + \delta_{T2}) \leq \tau \leq (\mathcal{T}_1 + \delta_{T5})))$ ■