MODEL REDUCTION OF DISCRETE REAL-TIME SYSTEMS

by

Mark Stephen Lawford

January 1997

A thesis submitted in conformity with the requirements
for the Degree of Doctor of Philosophy
Graduate Department of Electrical and Computer Engineering
University of Toronto

MODEL REDUCTION OF DISCRETE REAL-TIME SYSTEMS

by

Mark Stephen Lawford

A thesis submitted in conformity with the requirements
for the Degree of Doctor of Philosophy
Graduate Department of Electrical and Computer Engineering
University of Toronto

For my parents

Model Reduction of Discrete Real-Time Systems

A thesis submitted in conformity with the requirements
for the Degree of Doctor of Philosophy
Graduate Department of Electrical and Computer Engineering
University of Toronto

Abstract

In many Discrete-Event Systems (DES) both state and event information are of importance to the systems designer. To obtain compositionally consistent hierarchical models of systems, the behavior of Discrete-Event Systems with unobservable transitions and state output maps is considered. Observers for deterministic DES are generalized to nondeterministic DES and characterized using the join semilattice of compatible partitions of a transition system. This characterization points to efficient algorithms for computing both strong and weak state-event observers as solutions to the Relational Coarsest Partition problem (RCP). The strong and weak observation equivalences of Milner are shown to be special cases of our observers under the trivial (constant) state output map. The state-event equivalence based upon the observers is shown to be a congruence for a parallel composition operator, allowing the replacement of modules by their quotient systems.

Logics such as Ostroff's RTTL allow for the specification and verification of a system's state-event behavior. To make realistic problems amenable to analysis, a designer must typically decompose the system into subsystems (modules) and use algebraic abstraction (quotient systems) to obtain hierarchical system models that preserve the properties to be verified. In this thesis we use state-event observational equivalence to perform compositionally consistent model reduction for a subclass of formulas of state-event linear temporal logics, with particular attention to a discrete time temporal logic that is a simplification of RTTL. The reduction technique allows limited use of immediate operators. In the process, we develop a method of specifying modules' input/output behavior by defining observable satisfaction for RTTL-style temporal logics. The results are applied to the shut-down system of a nuclear reactor.

Acknowledgments

My supervisor, Prof. Murray Wonham, helped to shape this thesis from beginning to end with his extraordinary knowledge, insight and particular mathematical vision. Prof. Jonathan Ostroff got me rolling by asking, “So how do you compute these observers for finite state systems?” He later provided constructive criticism of the model reduction results and kindly made York University’s computing facilities available to me, thus making possible the computational results for the example application.

Thanks to all my friends who are, or were, in the University of Toronto Systems Control Group for thoughtful discussions and moral support, especially when my life went nonlinear. Kai Wong’s expert knowledge of aggregation provided the counterexample in Figure 3.1.

Financial support during my graduate studies has been generously provided for by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the University of Toronto through NSERC Scholarships and U of T Open Fellowships.

Finally, my family has encouraged me from the beginning in this endeavor and helped me throughout in ways too numerous to mention here.

Contents

List of Figures

List of Tables

List of Symbols

Symbol	Page	Description
$\mathbb{N}$	11	Set of Natural Numbers
$\mathbb{Z}$	11	Set of Integers
$Q_1 - Q_2$	11	Subset difference of Q_1 and Q_2
$Q_1 \setminus Q_2$	11	Complement of Q_1 with respect to Q_2
$Eq(Q)$	12	Set of Equivalence Relations of set Q
Δ, ∇	12	$\inf(Eq(Q))$ and $\sup(Eq(Q))$
$\ker(P)$	12	Equivalence Kernel of function P
$\mathcal{P}(Q)$	13	Power set of the set Q
$Q_1 \times Q_2$	12	Product of sets Q_1 and Q_2
π_1, π_2	13	Canonical projections associate with product
π_B	13	Canonical projections $A' \mapsto A' \cap B$
$eq(f_1, f_2)$	13	Equalizer of f_1 and f_2
id_Q	14	Identity map on the set Q
$g \circ f$	14	Functional composition
f_*	14	Lifting of function f to the power set level
$f_1 \dot{\cup} f_2$	15	Disjoint union of f_1 and f_2
$f_1 \cup f_2$	16	Union of functions
$f_1 \times f_2$	16	Functional product
$f_1 \otimes f_2$	17	Setwise functional product
M, M_1, M_2	19	Timed Transition Models (TTMs)
$\mathcal{V}, \Theta, \mathcal{T}$	19	TTM variable set, initial condition and transition set
$\mathcal{Q}$	19	Set of state assignments of a TTM
$\mathcal{Q}_{\mathcal{V}}$	19	Set of state assignments over the variable set $\mathcal{V}$
$\overline{\mathcal{Q}}$	$22, 30$	Set of extended state assignments of a TTM

Symbol	**Page**	**Description**
$\Sigma(\mathcal{T})$	25	Set of transition labels appearing in transition set $\mathcal{T}$
$M_1\|M_2$	25	Parallel composition of TTMs
$\mathbb{Q}, \mathbb{Q}_1, \mathbb{Q}_2$	27	State Event Labeled Transition Systems (SELTS)
Σ	27	Event set of a SELTS
R_Σ	27	Set of transition relations associated with event set Σ
$\alpha^{\mathbb{Q}}$	27	Transition function for event α in SELTS $\mathbb{Q}$
$Range_M(c_\alpha)$	30	Reduced range space for c_α in TTM M
$\mathbb{Q}_M$	30	State Event Labeled Transition System generated by TTM M
$P_{\mathcal{Q}_{\mathcal{V}_1}}$	31	Canonical projection from $\mathcal{Q}_{\mathcal{V}}$ to $\mathcal{Q}_{\mathcal{V}_1}$ for $\mathcal{V}_1 \subseteq \mathcal{V}$
$r(\mathbb{Q})$	31	r relabeling of SELTS $\mathbb{Q}$
$\mathbb{Q}_1 \vert[\Sigma_s]\vert \mathbb{Q}_2$	33	Event synchronous composition of SELTS
I	35	SELTS interface
$\mathbb{Q}_1 \vert[I]\vert \mathbb{Q}_2$	35	State-Event synchronous composition of SELTS
$Con(\mathbb{Q})$	39	Set of congruence of the deterministic SELTS $\mathbb{Q}$
$\mathbb{Q}/\theta$	$40, 46$	Quotient system of $\mathbb{Q}$ by congruence/compatible partition θ
$CP(\mathbb{Q})$	42	Compatible partitions of $\mathbb{Q}$
$\theta_s(\mathbb{Q})$	44	Strong state-event observer of $\mathbb{Q}$
$\mathbb{Q}_1 \cup \mathbb{Q}_2$	47	Union of SELTS
$\sim_{se}$	47	Strong state event equivalence
$h : \mathbb{Q}_1 \rightarrow \mathbb{Q}_2$	48	SELTS homomorphism from $\mathbb{Q}_1$ to $\mathbb{Q}_2$
$q \Rightarrow_{se} q'$	56	Unobservable move from q to q'
$q \overset{\alpha}{\Rightarrow}_{se} q'$	57	$\alpha -$ observable move from q to q'
$\mathbb{Q}'_{se}$	58	Observational closure of $\mathbb{Q}$
$\theta_w(\mathbb{Q})$	60	Weak state-event observer
$\approx_{se}$	$60, 61$	Weak observation equivalence

Symbol	**Page**	**Description**
$\mathbb{Q}/\!\!/\theta$	61	weak quotient system of $\mathbb{Q}$ by θ
Σ_-	88	Symbol for $\Sigma \cup \{-\}$, the event set union the "null" event
η	88	Next transition variable
σ	88	Computation - a state-event sequence
$\|\sigma\|$	88	Length of σ
σ^k	90	k-shifted suffix of σ
$\mathcal{M}(\mathbb{Q})$	88	Set of computations of $\mathbb{Q}$
$\models$	90, 91	Temporal logic satisfaction relation
$\bigcirc, \mathcal{U}, \mathcal{U}^{\alpha}_{[l,u]}$	90	Temporal logic "next", "until", and "bounded until" operators
$\Diamond, \Box$	90	Temporal logic "eventually" and "henceforth" operators
$P^{\sim}$	92	Strongly observed projection of computations
$P^{\approx}$	96	Weakly observed projection of computations
$\models_{\approx}$	99	Weak satisfaction relation
m, m_1, m_2	107	TTM modules
$\mathcal{I}, \mathcal{I}_1, \mathcal{I}_2$	107	TTM module interfaces
$\widehat{M}$	109	Augmented TTM of a module $m := (M, \mathcal{I})$
$\mathbb{Q}_m$	109	SELTS generated by the TTM module m
$m_1 \\| m_2$	112	TTM module composition

Contents

List of Figures

List of Symbols

Symbol	Page	Description
$\mathbb{N}$	11	Set of Natural Numbers
$\mathbb{Z}$	11	Set of Integers
$Q_1 - Q_2$	11	Subset difference of Q_1 and Q_2
$Q_1 \setminus Q_2$	11	Complement of Q_1 with respect to Q_2
$Eq(Q)$	12	Set of Equivalence Relations of set Q
Δ, ∇	12	$\inf(Eq(Q))$ and $\sup(Eq(Q))$
$\ker(P)$	12	Equivalence Kernel of function P
$\mathcal{P}(Q)$	13	Power set of the set Q
$Q_1 \times Q_2$	12	Product of sets Q_1 and Q_2
π_1, π_2	13	Canonical projections associate with product
π_B	13	Canonical projections $A' \mapsto A' \cap B$
$eq(f_1, f_2)$	13	Equalizer of f_1 and f_2
id_Q	14	Identity map on the set Q
$g \circ f$	14	Functional composition
f_*	14	Lifting of function f to the power set level
$f_1 \dot{\cup} f_2$	15	Disjoint union of f_1 and f_2
$f_1 \cup f_2$	16	Union of functions
$f_1 \times f_2$	16	Functional product
$f_1 \otimes f_2$	17	Setwise functional product
M, M_1, M_2	19	Timed Transition Models (TTMs)
$\mathcal{V}, \Theta, \mathcal{T}$	19	TTM variable set, initial condition and transition set
$\mathcal{Q}$	19	Set of state assignments of a TTM
$\mathcal{Q}_{\mathcal{V}}$	19	Set of state assignments over the variable set $\mathcal{V}$
$\overline{\mathcal{Q}}$	$22, 30$	Set of extended state assignments of a TTM

Symbol	**Page**	**Description**
$\Sigma(\mathcal{T})$	25	Set of transition labels appearing in transition set $\mathcal{T}$
$M_1 \| M_2$	25	Parallel composition of TTMs
$\mathbb{Q}, \mathbb{Q}_1, \mathbb{Q}_2$	27	State Event Labeled Transition Systems (SELTS)
Σ	27	Event set of a SELTS
R_Σ	27	Set of transition relations associated with event set Σ
$\alpha^{\mathbb{Q}}$	27	Transition function for event α in SELTS $\mathbb{Q}$
$Range_M(c_\alpha)$	30	Reduced range space for c_α in TTM M
$\mathbb{Q}_M$	30	State Event Labeled Transition System generated by TTM M
$P_{\mathcal{Q}_{\mathcal{V}_1}}$	31	Canonical projection from $\mathcal{Q}_\mathcal{V}$ to $\mathcal{Q}_{\mathcal{V}_1}$ for $\mathcal{V}_1 \subseteq \mathcal{V}$
$r(\mathbb{Q})$	31	r relabeling of SELTS $\mathbb{Q}$
$\mathbb{Q}_1 \vert [\Sigma_s] \vert \mathbb{Q}_2$	33	Event synchronous composition of SELTS
I	35	SELTS interface
$\mathbb{Q}_1 \vert [I] \vert \mathbb{Q}_2$	35	State-Event synchronous composition of SELTS
$Con(\mathbb{Q})$	39	Set of congruence of the deterministic SELTS $\mathbb{Q}$
$\mathbb{Q}/\theta$	$40, 46$	Quotient system of $\mathbb{Q}$ by congruence/compatible partition θ
$CP(\mathbb{Q})$	42	Compatible partitions of $\mathbb{Q}$
$\theta_s(\mathbb{Q})$	44	Strong state-event observer of $\mathbb{Q}$
$\mathbb{Q}_1 \cup \mathbb{Q}_2$	47	Union of SELTS
$\sim_{se}$	47	Strong state event equivalence
$h : \mathbb{Q}_1 \rightarrow \mathbb{Q}_2$	48	SELTS homomorphism from $\mathbb{Q}_1$ to $\mathbb{Q}_2$
$q \Rightarrow_{se} q'$	56	Unobservable move from q to q'
$q \overset{\alpha}{\Rightarrow}_{se} q'$	57	$\alpha -$ observable move from q to q'
$\mathbb{Q}'_{se}$	58	Observational closure of $\mathbb{Q}$
$\theta_w(\mathbb{Q})$	60	Weak state-event observer
$\approx_{se}$	$60, 61$	Weak observation equivalence

Symbol	**Page**	**Description**
$\mathbb{Q}/\!\!/\theta$	61	weak quotient system of $\mathbb{Q}$ by θ
Σ_{-}	88	Symbol for $\Sigma \cup \{-\}$, the event set union the "null" event
η	88	Next transition variable
σ	88	Computation - a state-event sequence
$\lvert\sigma\rvert$	88	Length of σ
σ^k	90	k-shifted suffix of σ
$\mathcal{M}(\mathbb{Q})$	88	Set of computations of $\mathbb{Q}$
$\models$	90, 91	Temporal logic satisfaction relation
$\bigcirc, \mathcal{U}, \mathcal{U}^{\alpha}_{[l,u]}$	90	Temporal logic "next", "until", and "bounded until" operators
$\Diamond, \Box$	90	Temporal logic "eventually" and "henceforth" operators
$P^{\sim}$	92	Strongly observed projection of computations
$P^{\approx}$	96	Weakly observed projection of computations
$\models_{\approx}$	99	Weak satisfaction relation
m, m_1, m_2	107	TTM modules
$\mathcal{I}, \mathcal{I}_1, \mathcal{I}_2$	107	TTM module interfaces
$\widehat{M}$	109	Augmented TTM of a module $m := (M, \mathcal{I})$
$\mathbb{Q}_m$	109	SELTS generated by the TTM module m
$m_1 \Vert m_2$	112	TTM module composition

Chapter 1

Introduction

Does this real-time control system do what we want?

With the widespread use of computers in control applications, this is becoming an increasingly common question. Systems of previously unprecedented ability and complexity are fast becoming commonplace. The space shuttle and "fly-by-wire" F-16 jet fighter spring to mind as examples of systems that would not be possible without computer control. These new systems are also creating an unprecedented potential for catastrophic failures due to software errors. For example

In 1987 a cancer treatment machine subjected a patient to a lethal dose of radiation. The machine at fault was a new computer controlled version of the original machine, which relied upon mechanical interlocks. A software bug caused the new machine's shielding mechanism to disengage if the machine operator made rapid corrections to the machine's settings using a particular method [Lee91].

In January 1990 a software failure in a single Manhattan computerized telephone switch put almost half of AT&T's US long distance network out of service for almost 9 hours [Lee91].

On June 4, 1996, the maiden flight of the Ariane 5 launcher ended in failure when, less than 40 seconds after take-off, the launcher veered off its flight path, broke up and exploded. The cause was traced to the software for the Inertial Reference System. Estimated loss: $500 million.[Lio96]

The goal of this thesis is to provide a mathematical basis for the development and verification of real-time discrete event systems that will aid in answering the above question. The development of the theoretical results contained herein was driven by the practical considerations of solving a particular controller software verification problem. Although the author was originally motivated by a specific problem, these concepts are applicable to a wide range of real-time discrete event systems, as the original problem captures many of the general issues involved.

1.1 Setting and Issues

A *discrete event system* (DES) is a physical system that is discrete (in time and state space), asynchronous (event rather than clock-driven), and in some sense generative (or nondeterministic). An *event* in a DES can be thought of as an indivisible action that occurs when the system instantaneously changes from one discrete state to another. For example, for most purposes a relay can be adequately modeled as a system with the two discrete states, OPEN and CLOSED. When the relay is switched, it makes an instantaneous transition from one of these states to the other.

Concurrency is one of the key features of DES. Most complex systems are composed of several interacting components or "modules". The two most common semantics for interacting DES are *interleaving* and *maximal parallelism.* In interleaving semantics, concurrent execution of two DES modules is represented by the interleaving of their atomic actions, while maximal parallelism requires the simultaneous execution of atomic actions in all system modules capable of performing an operation. The maximal parallelism semantics is generally applied to tightly coupled systems such as integrated circuits with a common clock. Interleaving semantics are generally thought to be more natural for modeling loosely coupled systems such as real-time control systems that react to changes in the plant under control. Therefore we will confine ourselves to interleaving semantics.

For either choice of concurrency semantics, the state space for a composite system is usually represented by the cross product of the state sets of the individual components. As a result, the complete system's state space may grow exponentially with the number of interacting components. This is commonly referred to as the *state explosion* problem. In the absence of any hierarchical organization, seemingly modest discrete event systems quickly scale beyond what the designer can intuitively understand or verify, even with the aid of a computer.

A *real-time discrete event system* is a DES that must meet hard timing deadlines in order to ensure its correct operation. An example of a such a system is a nuclear reactor shutdown system that has the requirement, "Within 2 seconds of the reactor pressure exceeding the maximum allowable limit, open the trip relay to shut down the reactor." To model such systems we will use a combined state-event and discrete time setting.

While it is possible to represent systems using only state information or only event information, there are many applications where the use of both state and event information is quite natural and may aid a designer's intuition. In the state-event setting that we propose, the use of labeled transition relations permits the application of synchronous composition operators, thereby allowing interacting modules to perform synchronous execution of shared events such as *tick*s of a global discrete clock [Ost89]. The *tick* events provide concurrent systems with a uniform notion of time, without the restrictiveness of clock driven models like [EMSS92] where one transition is one time step, limiting the method to single processor systems. An example of the use of *tick* events is Ostroff's RG2 graphs [Ost89] that are employed for model-checking Real-Time Temporal Logic (RTTL) properties. RG2 graphs use event information to reduce infinite state timed systems to finite state systems while preserving the relative timing of state changes and event outputs through the use of "*tick*" transitions. While it has been suggested that discrete time models are inherently inaccurate [ACD90], they are sufficiently accurate in many instances, particularly when dealing with digital control systems that sample their inputs (eg. [LW95]). In [LW95] the authors argue that discrete time models such as Ostroff's Timed Transition Models (TTMs) [Ost89] allow for a straightforward application of well known process algebraic equivalences such as observation (bisimulation) equivalence from Milner's CCS [Mil89]. On the other hand continuous time extensions of CCS such as [Wan91] lack the abstracting power of a congruence relation like weak observation congruence [Mil89] because of technical difficulties associated with their continuous time semantics. Also, continuous time models are too discriminating in many cases. Typically the behavior of digital control systems depends only on the state of the plant at the sampling instants, regardless of precisely when state changes occur between samples.

The addition of event information is also crucial for performing synchronous composition of systems and thereby performing supervisory control through the disablement of controllable events [RW87]. The ability to perform supervisory control provides a means of modifying an existing module's behavior to meet a new specification and opens up the possibility of exploiting the synthesis techniques of the supervisory control community (eg. [RW87, ZW90, LW90, BW94]).

1.2 Related Work

In this section we outline algebraic equivalence verification and temporal logic model reduction, two of the main approaches that have been developed to address the state explosion problem. Both of these methods are further developed in the thesis using an approach based upon the belief that if you get the algebra "right," then "good things" will happen. In this vein we first rigorously develop an algebraic equivalence that is appropriate for equivalence verification in our real-time setting and then see how quotient systems with respect to our equivalence can be used for compositional temporal logic model reduction. The example at the end of the thesis illustrates the mutually beneficial relationship that can exist between equivalence verification and temporal logic model reduction.

1.2.1 Algebraic Equivalence Verification

For well over a decade computer specialists have been looking at the problem of formally verifying the "equivalence" of a system implementation and a system specification. Two of the most influential theories in this area have been Hoare's Communicating Sequential Processes (CSP) [Hoa85] and Milner's Calculus of Communicating Systems (CCS) [Mil80, Mil89]. These theories along with others are now often collectively referred to as *Process Algebra*.

Process algebra models discrete event systems by using algebraic equations to describe the behavior of processes that communicate via synchronized actions. Equations can be constructed that model finite state automata and even some infinite state transition structures. Operations are then defined that allow the equations to be combined to build larger, more complicated systems. The algebraic properties of the process equations and operations are studied to determine when, in a well defined sense, two processes can be considered equivalent. The notion of equivalence is chosen in such a way that equivalent processes can then be substituted for each other resulting in construction of behaviorally equivalent systems.

In [Mil89], the author establishes the equivalence of several example system implementations and their specifications using a set of equational laws.

A visual method was introduced in [FG89] for checking the reachability of a class of extended timed Petri nets. The net approach and the provision of net transformations led to a graph-based method of verification. As in [Ost89], the problem of constructing equivalent abstract real-time systems from a given system model was not considered. In the same spirit as [FG89], [LW95] develops a set of easily applicable, and demonstrably correct, transformations that preserve system equivalence, and lend themselves to abstraction, in the setting of TTMs. A developer need not be familiar with observation equivalence or process algebra to be able to use the simple set of visual transformations to prove that a system correctly implements its specification in a well defined way.

In addition to the transformational methods of equivalence verification identified above, computational methods exist for the verification of process algebraic equivalences on finite state systems [KS83, BC89, BCM92]. Computational tools for equivalence verification can confirm the equivalence of a system specification and its implementation, but they generally provide little information of use to the system design when the verification fails. This is in contrast to the temporal logic model-checkers discussed below. A model-checker typically demonstrates the failure of a system to satisfy a temporal logic specification by generating a counterexample computation which can then help the designer to correct the system.

1.2.2 Temporal Logic, Model-Checking and Model Reduction

In the case of equivalence verification, the implementation and system specification are modeled using the same technique – automata, process algebraic equations, TTMs, etc. In temporal logic model-checking, logic formulas are used to specify the desired behavior of the implementation model or "program." Just as predicate logic permits reasoning about states, so temporal logic permits reasoning about sequences of states.

The temporal logics in common use for formal verification can usually be classified as either *linear time* or *branching time.* Linear Temporal Logics such as [MP92] and its real-time derivative RTTL [Ost89] express properties of the set of infinite paths that can be generated from an initial state. In addition to *safety properties* that express the fact that no path ever reaches a set of "bad" states, linear temporal logics can also express *fairness properties* regarding the eventuality of certain states (e.g. "Henceforth if the reactor pressure exceeds its maximum allowable value in the current state, then eventually in a future state the relay will be OPEN"). In addition to safety properties, branching time temporal logics such as the Computational Tree Logic (CTL) [CE81, ES84], allow one to express properties of a state such as the existence of a path to another state satisfying a desirable property (e.g. "There exists a path from the current state to a state where the relay is OPEN").

For finite state systems one can "model-check" a temporal formula. The global state transition graph of the system is represented as a Kripke structure, a state transition graph with a state output map that maps each state to the set of atomic predicates satisfied by the state. A *model-checking* algorithm can then be used to determine if a program's Kripke structure is a valid model of (i.e. satisfies) a specification expressed as a temporal logic formula (e.g. [CE81, LP85, CES86]).

With recent advances in computing power and data structures, model-checking techniques such as [McM92] have proven effective for some very large systems [BCM92]. The largest of these systems typically come from the digital hardware domain and have a great deal of regularity in their state transition structure that can be exploited by the symbolic techniques to obtain compact representations of large systems. If one wishes to model-check large concurrent systems lacking in regularity, larger digital hardware systems, or simply to reduce the computation time required for the model-check, one must perform some sort of model reduction to cope with the state explosion problem.

In model reduction one starts out with a system for which one would like to verify (model-check) formulas from a particular set of formulas or class of temporal formulas that are of interest. To facilitate the verification process or, in some cases, render the problem tractable, a reduced model is obtained such that, if the reduced model satisfies the temporal formulas under investigation, then the original system satisfies the temporal formulas. One of the first results on model reduction came from the use of the Process Algebra "strong observation" equivalence of [Mil80] to generate property preserving quotient systems [BFH+92]. In [Kai96], [KV91], and [KV92], Kaivola *et al.* develop a compositional model reduction technique using an equivalence based upon the process algebraic "failure equivalence" of [Hoa85]. This model reduction technique allows for the reduction of modules before they are composed in an effort to control the state explosion problem before it appears. In a similar vein we will define an algebraic equivalence relation to perform compositional model reduction but we will apply it to a real-time setting.

In response to the need for formal methods with visual appeal, Ostroff *et al.* have introduced Timed Transition Models [Ost89, OW90]; but the Real-Time Temporal Logic on which the proof and verification system is based is quickly overwhelmed by the state explosion problem inherent in composite discrete systems. No method was provided for moving between levels of abstraction of real-time models to allow model reduction or behavioral comparison of two TTMs. Such flexibility would enable one to project out extraneous details to obtain high-level TTM models or, conversely, to refine high level TTM specifications into workable implementations. While [Law92, LW95] provided a means of abstraction for TTMs through equivalence preserving transformations, no effort was made to deal with the compositional aspects of TTMs. Also, the heuristic methods developed in these works require the active participation and insight of the systems designer in the verification process and are somewhat restrictive in the abstractions that they permit. By obtaining an algebraic characterization of the equivalence of [Law92, LW95], we will be able to provide computationally efficient algorithms for equivalence verification of finite state systems and extend the results to the verification of composite systems using model reduction and RTTL model-checking.

1.3 Contributions

Research on discrete-event systems (DES) has led to renewed appreciation of control architecture - decentralized and hierarchical decomposition - for the effective modeling of large systems. In theoretical treatment, such architectural features are brought in through standard algebraic constructs, namely unions, products and quotient structures of the state sets involved. Inasmuch as architecture amounts to decomposition of information transfer and decision making, the systemic notions of observation and observer are fundamental. These find their algebraic setting in lattices of equivalence relations (partitions), and the associated sublattices of congruences with respect to the dynamic flow. Thus in approaching any new class of state transition structures, a first item of business is to clarify the algebraic structure of observers (congruences) along with their computational complexity. Because, in general, equivalence is undecidable, these issues tend to be both nontrivial and of practical interest.

We begin the contributions of this thesis by generalizing previous observers - well known (under various guises) in either the control or process algebra literature - to a unified construct that we call a state-event observer. In this treatment both state changes and output events (or event signals) are assigned equal status, thus allowing a flexible modeling approach to DES in which both state- and event-based control are equally natural.

We recall the duality of states and events. Event-based models include most process-algebraic theory derived from [Mil80], [Mil89] and [Hoa85], as well as control-theoretic approaches such as [ZW90], [WW92] and [Won94]. States are really only viewed as a way of keeping track of what sequences of events have been executed and what future events are possible. Quotient structure is induced by projection of languages. In [FZ91], state outputs are used solely to provide additional information for the control of events; quotient structure is not considered. On the other hand in [Won76, Har87, GF91], state structure is preeminent, and behavior treated as sequences of states or groups of states. For instance state charts [Har87, BH93] offer a visual representation (nested boxes and arrows) of state set decomposition via nested products and disjoint unions, in principle to arbitrary depth. Of course the transition structure and control must admit compatible decomposition for the method to be computationally attractive, and to admit quotient structures induced by suitable state-transition homomorphisms.

In many applications both state occupancy and event sequencing are important, and so we need quotients with respect to both. One instance is Timed Transition Models (TTMs) [Ost89, OW90], which express behavior such as: "Do α only when $y = 2$ for 3 or more '*tick*s' of the clock." In [Law92, LW92] the authors adapted to TTMs the event-based observation equivalence of [Mil89] by projecting TTM states (the state assignments of [Ost89]) to their factors defined by selected subsets of data variables. Observable events are just those TTM state changes that affect the variables in question, and the event labels themselves are "projected out". The class of projections for which a quotient can be defined was severely restricted; but we shall show how this situation can be improved on.

In Chapter 3 we introduce strong and weak state-event observers for State-Event Labeled Transition Systems (SELTS) (the underlying model of many DES formalisms [DeN87] including, as we will see, TTMs); state output maps and event projections play symmetric roles. Our observers (congruences) induce consistent high-level abstractions (quotients) so that, just as in [ZW90], control designed at the abstract level can be consistently implemented at the detailed ('real-world') level. The development of strong observers and their quotient systems parallels the results on indistinguishability of finite transition systems in [Arn94]. On the basis of [KS83, PT87, BC89] we are able to appeal to efficient polynomial-time algorithms for computing our observers on finite-state SELTS.

We then investigate the algebraic properties of state-event equivalence, obtaining results on minimum state realizations of equivalent systems and compositional consistency. These results then form the basis of the applications of state-event equivalence to model reduction for temporal logic model-checking of Chapter 4. There we use state-event observational equivalence to perform compositional model reduction for a subclass of formulas of state-event linear temporal logics, with particular attention being paid to a discrete time temporal logic that is a simplification of RTTL.

In Chapter 5 we apply the theory of the previous chapters in an effort to answer the question, "Does this real-time control system do what we want?" for the real-time control system that originally motivated the author's investigation of formal methods. The Delayed Reactor Trip (DRT) control system exhibits many of the distinguishing characteristics of real-time discrete event systems. To operate correctly the implementation must meet hard real-time deadlines in response to inputs from the plant. For a simple shutdown system, it displays surprisingly complex behavior, and the final implementation incorporating 3 redundant controllers exhibits the characteristic state explosion at the implementation level.

Preliminary results applying compositional model-checking to the DRT illustrate compositional model reduction's potential for handling the state explosion problem and also demonstrate the technique's limitations.

Chapter 2

Preliminaries

In this chapter we introduce notation and concepts that will be used throughout the thesis. Subsequent theoretical chapters are relatively self-contained in that they rely upon different additional mathematical concepts. Each chapter introduces any additional mathematical concepts and notation as required to obtain the main results of the chapter.

2.1 Notation and Mathematical Preliminaries

In this thesis we use $\mathbb{Z}$ and $\mathbb{N}$ to denote the set of integers and the set of natural numbers ($\{0, 1, 2, \ldots\}$) respectively. We will use "iff" as an abbreviation of "if and only if." We also require some basic set notation. Let Q_1 and Q_2 be sets. If $Q_2 \subseteq Q_1$ then we define $Q_1 - Q_2 := \{q \in Q_1 : q \notin Q_2\}$. For two arbitrary sets Q and Q', we define $Q \setminus Q' := Q - (Q \cap Q')$. The cardinality of the set Q will be denoted by $|Q|$. When Q is a countably infinte set we will write $|Q| = \omega$.

Let Q be a set and $S \subset Q \times Q$ be a binary relation. Then S is an *equivalence relation* if S satisfies the following three conditions:

(i) Reflexivity: $(\forall q \in Q)\ (q, q) \in S$,

(ii) Symmetry: $(\forall q, q' \in Q)\ (q, q') \in S$ implies $(q', q) \in S$,

(iii) Transitivity: $(\forall q, q', q'' \in Q)\ (q, q') \in S \wedge (q', q'') \in S$ implies $(q, q'') \in S$.

Denote the set of all equivalence relations on Q by $Eq(Q)$. Any function $P : Q \rightarrow R$ induces an equivalence relation $\ker(P) \in Eq(Q)$, the equivalence kernel of P, given by

$$(q_1, q_2) \in \ker(P) \text{ if and only if } P(q_1) = P(q_2).$$

Similarly, any $\theta \in Eq(Q)$ defines a canonical output map $\theta : Q \rightarrow Q/\theta$, which projects each $q \in Q$ onto its θ-cell (equivalence class). $Eq(Q)$ becomes a complete lattice under the operations $\wedge, \vee$ when we define:

(i) $(q, q') \in \theta_1 \wedge \theta_2$ iff $(q, q') \in \theta_1$ and $(q, q') \in \theta_2$

(ii) $(q, q') \in \theta_1 \vee \theta_2$ iff $(\exists q_1, q_2, \ldots, q_n \in Q)(q_i, q_{i+1}) \in \theta_1$ or $(q_i, q_{i+1}) \in \theta_2$, $i = 1, \ldots, n-1$ and $q = q_1$ and $q' = q_n$.

A basic result of universal algebra is that when each $\theta \in Eq(Q)$ is associated with the partition of Q corresponding to the cells of θ, the lattice of equivalence relations is isomorphic to the poset lattice of partitions of Q with the partial order $\theta_1 \leq \theta_2$ iff each cell of θ_1 is a subset of a cell of θ_2. Thus we can talk interchangeably about equivalence relations and partitions. When talking about partitions $\theta_1 \wedge \theta_2 \in Eq(Q)$ ($\theta_1 \vee \theta_2$) is the coarsest (finest) partition finer (coarser) than both θ_1 and θ_2 [BS81]. We will denote the trivial partitions $\{\{q\} : q \in Q\} = \inf(Eq(Q))$ and $\{Q\} = \sup(Eq(Q))$ by Δ and ∇ respectively.

2.1.1 Products, Projections and Equalizers

In this subsection we borrow some basic category-theoretic definitions and notation. The interested reader is referred to [AM75] for a complete treatment of category theory.

Given two sets A and B, we define the *product* of A and B to be the standard Cartesian product:

$$A \times B := \{(a, b) : a \in A \text{ and } b \in B\}.$$

With any product we associate two special maps, the elementwise projections:

$$\pi_1 : A \times B \to A, \quad (a, b) \mapsto a$$
$$\pi_2 : A \times B \to B, \quad (a, b) \mapsto b$$

For a set A, we define the *power set* of A to be $\mathcal{P}(A) := \{A' : A' \subseteq A\}$. In addition to the two projections associated with a product, we will find it convenient to talk about the canonical projection from the power set $\mathcal{P}(A)$ of A to itself that results from intersection with a set B.

$$\pi_B : \mathcal{P}(A) \to \mathcal{P}(A)$$
$$\text{For } A' \subseteq A, A' \mapsto A' \cap B$$

In talking about the synchronous composition of systems with shared variables, we will find it convenient to identify the subset of a domain that "equalizes" two functions. In category theory, this set has the abstract, arrow theoretic characterization given below.

Definition 2.1 *(cf. [AM75]) A map $h : B \to A$ is an* equalizer *iff there exists a pair of maps $f_i : A \to R$, $i = 1, 2$ such that $f_1 \circ h = f_2 \circ h$ and such that whenever $h' : B' \to A$ satisfies $f_1 \circ h' = f_2 \circ h'$, there exists a unique map ϕ such that $h \circ \phi = h'$. In this situation we call h the* equalizer of f_1 and f_2, *and write $h = eq(f_1, f_2)$.*

$$\begin{tikzcd}
B \arrow[r, "h"] & A \arrow[r, shift left, "f_1"] \arrow[r, shift right, "f_2"'] & R \\
B' \arrow[u, dashed, "\phi"] \arrow[ur, "h'"'] & &
\end{tikzcd}$$

Figure 2.1: Commutative diagram defining $h = eq(f_1, f_2)$, the equalizer of f_1 and f_2.

Given any f_1, f_2 as above, because we are dealing with the category of sets, $eq(f_1, f_2)$ will always exist (see [AM75]). In fact we can take $B := \{a \in A : f_1(a) = f_2(a)\}$ and let $h : B \to A$ be the injection $a \mapsto a$. Henceforth we will use $eq(f_1, f_2)$ to denote both the injection into A and the set $\{a \in A : f_1(a) = f_2(a)\}$. The intended meaning of $eq(f_1, f_2)$ should be clear from the context.

2.1.2 Properties of Functional Operators

Throughout the thesis we will use several operators to combine functions to create new functions. Here we introduce the operators and establish some of their basic properties that will be used in proofs of results in subsequent chapters.

It will often be useful to talk about the *identity map* on a set. Henceforth we will denote the identity map on a set Q by $id_Q : Q \to Q$ (i.e. $q \mapsto q$).

Given two maps (functions) $f : Q_1 \to Q_2$ and $g : Q_2 \to Q_3$ such that the codomain of the first is the domain of the second, we define the *composite function* $g \circ f : Q_1 \to Q_3$ to be the function $q_1 \mapsto g(f(q_1))$. We call $\circ$ the *functional composition operator.*

Any function $f : Q_1 \to Q_2$ induces a function at the power set level, $f_* : \mathcal{P}(Q_1) \to \mathcal{P}(Q_2)$. For $Q \subseteq Q_1$, $f_*(Q) := \{f(q) : q \in Q\}$. We call f_* the *lifting of* f and refer to $_*$ as the *lifting operator.* Since the lifting of a function applies the original function to each element of a subset of the original function's domain, any f_* distributes over set union.

Claim 2.2 *Given $f : Q_1 \to Q_2$ and subsets $Q, Q' \subseteq Q_1$. Then*

$$f_*(Q \cup Q') = f_*(Q) \cup f_*(Q')$$

Proof:

$$\begin{aligned} f_*(Q \cup Q') &= \{f(q) : q \in Q \cup Q'\} \\ &= \{f(q) : q \in Q \text{ or } q \in Q'\} \\ &= \{f(q) : q \in Q\} \cup \{f(q) : q \in Q'\} \\ &= f_*(Q) \cup f_*(Q') \end{aligned}$$

□

Now that we have the composition and lifting operator it seems logical to consider whether the lifting operator distributes over the functional composition operator. The next claim proves that this is in fact the case.

Claim 2.3 *Given functions $f : Q_1 \to Q_2$ and $g : Q_2 \to Q_3$*

$$(g \circ f)_* = g_* \circ f_*$$

Proof: Let $Q \subseteq Q_1$. Then

$$\begin{aligned} g_* \circ f_*(Q) &= g_*(\{f(q) : q \in Q\}) \\ &= \{g(q') : q' \in \{f(q) : q \in Q\}\} \\ &= \{g \circ f(q) : q \in Q\} \\ &= (g \circ f)_*(Q) \end{aligned}$$

□

On occasion we will find it convenient to talk about various types of unions of functions as a notational convenience. The simplest form of functional union is the disjoint union. Given functions $f_1 : Q_1 \to R_1$ and $f_2 : Q_2 \to R_2$, if $Q_1 \cap Q_2 = \emptyset$ then we define the *disjoint union* of f_1 and f_2 to be the function $f_1 \dot{\cup} f_2 : Q_1 \dot{\cup} Q_2 \to R_1 \cup R_2$ such that for $q \in Q_1 \dot{\cup} Q_2$:

$$f_1 \dot{\cup} f_2(q) := \begin{cases} f_1(q), & q \in Q_1 \\ f_2(q), & q \in Q_2 \end{cases}$$

What we will call the "union of functions" is a more restricted operator. Suppose we have two functions with the same domain and codomain, where the codomain is closed under the operation of union (eg. a codomain of $\mathcal{P}(Q)$). Then the value of the union of the functions on an element of the domain is simply the union of the evaluations of each function. More formally, given functions $f_i : Q \to R$ for $i = 1, 2$, if for any $r_1, r_2 \in R$ we have $r_1 \cup r_2 \in R$, then we define the *union of f_1 and f_2*,

$f_1 \cup f_2 : Q \to R$, to be the function such that $q \mapsto f_1(q) \cup f_2(q)$.

Claim 2.4 *Given functions $f_i : Q \to \mathcal{P}(R)$ for $i = 1, 2$, $g : R \to S$ and $h : Q_1 \to Q$.*

(i) $g_* \circ (f_1 \cup f_2) = (g_* \circ f_1) \cup (g_* \circ f_2)$

(ii) $(f_1 \cup f_2) \circ h = (f_1 \circ h) \cup (f_2 \circ h)$

Proof: (i) Follows immediately from Claim 2.2 and the definition of functional union.

For (ii), let $q_1 \in Q_1$. Then

$$\begin{aligned} [(f_1 \cup f_2) \circ h](q_1) &= f_1(h(q_1)) \cup f_2(h(q_1)) \\ &= (f_1 \circ h)(q_1) \cup (f_2 \circ h)(q_1) \\ &= [(f_1 \circ h) \cup (f_2 \circ h)](q_1) \end{aligned}$$

Thus (ii) is demonstrated. □

A functional operator that we will use in the definition of synchronous product of systems later in this chapter is the product operator. For functions $f_1 : Q_1 \to R_1$ and $f_2 : Q_2 \to R_2$, we define the *product of f_1 and f_2* to be the function $f_1 \times f_2 : Q_1 \times Q_2 \to R_1 \times R_2$ such that $(q_1, q_2) \mapsto (f_1(q_1), f_2(q_2))$. The following claim demonstrates that the order in which product and composition operators are applied is irrelevant.

Claim 2.5 *Given functions $f_i : Q_i \to R_i$ and $g_i : R_i \to S_i$ for $i = 1, 2$. Then*

$$(g_1 \times g_2) \circ (f_1 \times f_2) = (g_1 \circ f_1) \times (g_2 \circ f_2)$$

Proof: Let $q_1 \in Q_1$ and $q_2 \in Q_2$. Then

$$\begin{aligned} (g_1 \times g_2) \circ (f_1 \times f_2)(q_1, q_2) &= g_1 \times g_2(f_1(q_1), f_2(q_2)) \\ &= (g_1(f_1(q_1)), g_2(f_2(q_2))) \\ &= (g_1 \circ f_1(q_1), g_2 \circ f_2(q_2)) \\ &= (g_1 \circ f_1) \times (g_2 \circ f_2)(q_1, q_2) \end{aligned}$$

as required. $\square$

We now define a variation of the functional product called the setwise functional product operator. The new operator will allow us to obtain an alternative functional characterization of synchronous product so that we may use arrow theoretic methods for proving properties of homomorphisms. Given $f_1 : Q_1 \to \mathcal{P}(R_1)$ and $f_2 : Q_2 \to \mathcal{P}(R_2)$, define the *setwise functional product of* f_1 *and* f_2 to be the function

$$f_1 \otimes f_2 : Q_1 \times Q_2 \to \mathcal{P}(R_1) \times \mathcal{P}(R_2)$$

such that $(q_1, q_2) \mapsto f_1(q_1) \times f_2(q_2)$. Thus if $R_i' \subset R_i$ and $f_i(q_i) = R_i'$ for $i = 1, 2$ then $f_1 \otimes f_2(q_1, q_2) = R_1' \times R_2' = \{(r_1, r_2) : r_1 \in R_1' \text{ and } r_2 \in R_2'\}$ while $f_1 \times f_2(q_1, q_2) = (R_1', R_2')$. We can extend the setwise product operator to handle functions that range over elements instead of sets. For example with f_1 as above, if $f_2 : Q_1 \to R_2$ then define $f_1 \otimes f_2(q_1, q_2) = f_1(q_1) \times \{f_2(q_2)\}$.

Next we present two specialized results regarding the composition of functional products and setwise functional products. These equalities will be used in proofs concerning the composition of equivalent systems in Section 3.4.

Claim 2.6 *Given functions* $\alpha_i : Q_i \to \mathcal{P}(Q_i)$, $h_i : Q_1 \to R_i$ *and* $\beta_i : R_i \to \mathcal{P}(R_i)$ *for* $i = 1, 2$*. Then*

(i) $(h_1 \times h_2)_* \circ (\alpha_1 \otimes \alpha_2) = (h_{1*} \circ \alpha_1) \otimes (h_{2*} \circ \alpha_2)$

(ii) $(\beta_1 \otimes \beta_2) \circ (h_1 \times h_2) = (\beta_1 \circ h_1) \otimes (\beta_2 \circ h_2)$

Proof:

(i) Let $q_1 \in Q_1$ and $q_2 \in Q_2$. Then

$$\begin{aligned}
(h_1 \times h_2)_* \circ (\alpha_1 \otimes \alpha_2)(q_1, q_2) &= (h_1 \times h_2)_*(\alpha_1(q_1) \times \alpha_2(q_2)) \\
&\quad \text{by Def. of } \otimes \\
&= (h_1 \times h_2)_*(\{(q_1', q_2') : q_1' \in \alpha_1(q_1) \text{ and } q_2' \in \alpha_2(q_2)\}) \\
&= \{(h_1 \times h_2)((q_1', q_2')) : q_1' \in \alpha_1(q_1) \text{ and } q_2' \in \alpha_2(q_2)\}
\end{aligned}$$

$$
\begin{aligned}
& \text{by Def. of}\,_* \\
= \; & \{(h_1(q_1'), h_2(q_2')) : q_1' \in \alpha_1(q_1) \text{ and } q_2' \in \alpha_2(q_2)\} \\
& \text{by Def. of } \times \\
= \; & \{(r_1, r_2) : r_1 \in h_{1*} \circ \alpha_1(q_1) \text{ and } r_2 \in h_{2*} \circ \alpha_2(q_2)\} \\
= \; & h_{1*} \circ \alpha_1(q_1) \times h_{2*} \circ \alpha_2(q_2) \\
= \; & (h_{1*} \circ \alpha_1) \otimes (h_{2*} \circ \alpha_2(q_2)) \\
& \text{by Def. of } \otimes
\end{aligned}
$$

Thus (i) is proved.

(ii) For any $q_1 \in Q_1$ and $q_2 \in Q_2$:

$$
\begin{aligned}
(\beta_1 \otimes \beta_2) \circ (h_1 \times h_2)(q_1, q_2) &= (\beta_1 \otimes \beta_2)(h_1(q_1), h_2(q_2)) \text{ by Def. of } \times \\
&= \beta_1(h_1(q_1)) \times \beta_2(h_2(q_2)) \text{ by Def. of } \otimes \\
&= (\beta_1 \circ h_1)(q_1) \times (\beta_2 \circ h_2)(q_2) \text{ by Def. of } \circ \\
&= (\beta_1 \circ h_1) \otimes (\beta_2 \circ h_2)(q_1, q_2) \text{ by Def. of } \otimes
\end{aligned}
$$

Thus (ii) is proved. □

2.2 System Models

In this section we introduce the mathematical models that will be used to describe Discrete Event Systems (DES) throughout the thesis. Timed Transition Models will be used as high level representations of systems that motivate the state-event approach taken in this work. The State-Event Labeled Transition Systems (SELTS) described later will be used as our underlying model of a Discrete Event System (DES).

2.2.1 Timed Transition Models

Ostroff's original work on Timed Transition Models [Ost89] centered around the use of Real Time Temporal Logic to verify that controlled systems met certain real-time specifications. No work was done on hierarchical or abstract representation of complex low level systems. In this subsection we introduce a modified version of the Timed Transition Models (TTMs) employed in [OW90]. We drop the Real Time Temporal Logic (RTTL) assertion language, although we still use the infinite string semantics it required. To simplify matters, the initial condition is limited to specifying a unique initial state instead of (possibly) multiple initial states. Originally transitions' operation functions were required to be deterministic but we allow non-determinisitc operation functions to allow the modeling of external behavior by TTM modules. The examples of this and subsequent chapters demonstrate that in this format TTMs provide a concise way of describing state-event transition structures representing real-time systems.

A *Timed Transition Model* (TTM) M is a triple given by

$$M := \langle \mathcal{V}, \Theta, \mathcal{T} \rangle$$

where $\mathcal{V}$ is a set of variables, Θ is an initial condition (a boolean-valued expression in the variables), and $\mathcal{T}$ is a finite set of transitions.

$\mathcal{V}$ always includes two special variables: the global time variable t and an activity variable which we will usually denote by x. For $v \in \mathcal{V}$ the range space of v is $Range(v)$ (eg. $Range(t) = \mathbb{N}$ where $\mathbb{N} := \{0, 1, 2, \ldots\}$). We define $\mathcal{Q}$, the set of *state assignments of* M, to be the product of the ranges of the variables in $\mathcal{V}$. That is

$$\mathcal{Q} := \times_{v_i \in \mathcal{V}} Range(v_i)$$

For a state assignment $q \in \mathcal{Q}$ and a variable $v \in \mathcal{V}$, we will denote the value of v in state assignment q by $q(v)$ where $q(v) \in Range(v)$. When we wish to distinguish between state assignments over different variable sets, we will use the variable set as a subscript (i.e. the *set of state assignments over* $\mathcal{V}$ will be denoted $\mathcal{Q}_{\mathcal{V}} := \times_{v_i \in \mathcal{V}} Range(v_i)$).

$\mathcal{T}$ is the transition set. A transition is a labeled 4-tuple

$$\alpha := (e_\alpha, h_\alpha, l_\alpha, u_\alpha)$$

where α is the transition's label. With a slight abuse of notation, we will then refer to the transition by its label (eg. $\alpha \in \mathcal{T}$). Whether α is meant to refer to the labeled 4-tuple or the transition's label itself should be clear from the context. In the above e_α is the transition's enablement condition (a boolean valued expression in the variables of $\mathcal{V}$), h_α is the operation function, and $l_\alpha \in Range(t) = \mathbb{N}$ and $u_\alpha \in \mathbb{N} \cup \{\infty\}$ are the lower and upper time bounds respectively with $l_\alpha \leq u_\alpha$. We say that α is *enabled* when $q(e_\alpha) = true$. The (possibly nondeterministic) operation function $h_\alpha : \mathcal{Q} \to \mathcal{P}(\mathcal{Q})$, maps the current state assignment to the set of new state assignment that are possible next states when the transition occurs. If $h_\alpha(q) = \emptyset$ then an α transition is not possible from q. $\mathcal{T}$ always contains the special transition *tick*,

$$tick := (true, [t : t + 1], -, -)$$

which represents the passage of time on the global clock. *tick* is the only transition that affects the time variable t and also has no lower or upper time bound. All other transition time bounds are given relative to numbers of occurrences of *tick*.

Θ is the initial condition, a boolean valued expression in the variables of $\mathcal{V}$ that is used to identify a unique initial state of the system.

2.2.2 TTM Semantics

A *trajectory* of a TTM is any infinite string of the TTM state assignments connected by transitions, of the form $q_0 \overset{\alpha_0}{\rightarrow} q_1 \overset{\alpha_1}{\rightarrow} q_2 \overset{\alpha_2}{\rightarrow} \ldots$. The interpretation is that q_i goes to q_{i+1} via the transition α_i. A state trajectory $\sigma := q_0 \overset{\alpha_0}{\rightarrow} q_1 \overset{\alpha_1}{\rightarrow} q_2 \overset{\alpha_2}{\rightarrow} \ldots$ is a *legal* trajectory of a TTM M if it meets the following four requirements:

1. **Initialization:** The initial state assignment satisfies the initial condition ($q_0(\Theta)$ $= true$ - i.e. q_0 satisfies Θ and hence is the unique initial state assignment).

2. **Succession:** For all i, q_{i+1} is obtained from q_i by applying the operation function of α_i ($q_{i+1} \in h_{\alpha_i}(q_i)$) and α_i is enabled in state assignment q_i (ie. $q_i(e_{\alpha_i}) = true$).

3. **Ticking:** The clock must *tick* infinitely often. That is, there are an infinite number of transitions $\alpha_i = tick$. This eliminates the possibility of "clock stoppers" in the trajectory where an infinite number of non-*tick* transitions occur consecutively without being interleaved with any *tick*s. This would imply that the TTM is performing an infinite number of actions in a finite time.

4. **Time Bounds:** To determine if the trajectory σ satisfies the time bound requirements of the TTM M, we associate with each non-*tick* transition α, a counter variable c_α with $Range(c_\alpha) = \mathbb{N}$. Each α transition's counter is initially set to zero and is reset to zero after an α transition or a transition that enters a new state assignment where α is disabled (ie. $e_\alpha = false$). The counter is only incremented by the occurrence of a *tick* transition when α is enabled ($e_\alpha = true$). Any non-*tick* transition α can legally occur only when when its counter is in the region specified by the transition's time bounds (ie. $l_\alpha \leq c_\alpha \leq u_\alpha$). The upper time bounds on transitions represent hard time bounds by which time the transitions are guaranteed to occur. Thus if α's counter reaches its upper time bound, then it is forced to occur before the next tick of the clock unless it is preempted by another non-*tick* transition that disables α (and hence resets α's counter). Hence for a *tick* transition to legally

occur, every enabled transition α must have a counter value less than its upper time bound ($c_\alpha < u_\alpha$). We now formalize the above description.

For the TTM $M := \langle \mathcal{V}, \Theta, \mathcal{T} \rangle$, we will denote the set of transition counters by $C := \{c_\alpha : \alpha \in \mathcal{T} - \{tick\}\}$. We then obtain the TTM's underlying state set $\overline{\mathcal{Q}} := \mathcal{Q} \times \mathbb{N}^C$, the set of *extended state assignments*. From the trajectory σ we derive the *full trajectory* $\bar{\sigma} := \bar{q}_0 \overset{\alpha_0}{\rightarrow} \bar{q}_1 \overset{\alpha_1}{\rightarrow} \bar{q}_2 \overset{\alpha_2}{\rightarrow} \ldots$, where each $\bar{q}_i \in \overline{\mathcal{Q}}$ is obtained from σ as follows:

For all $v \in \mathcal{V}$, $\bar{q}_i(v) = q_i(v)$.

For all $c_\alpha \in C$, $\bar{q}_0(c_\alpha) = 0$ and for $i = 0, 1, 2, \ldots$

$$\bar{q}_{i+1}(c_\alpha) = \begin{cases} \bar{q}_i(c_\alpha) + 1, \text{ if } q_i(e_\alpha) = true \text{ and } \alpha_i = tick \\ 0, \text{ if } q_{i+1}(e_\alpha) = false \text{ or } \alpha_i = \alpha \\ \bar{q}_i(c_\alpha), \text{ otherwise} \end{cases}$$

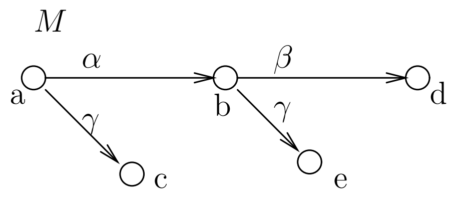
The trajectory σ satisfies the time bounds of M iff the following two conditions hold in $\bar{\sigma}$ for all $i = 0, 1, \ldots$:

(i) $\alpha_i = tick$ iff for all $\alpha \in \mathcal{T} - \{tick\}$, $q_i(e_\alpha) = true$ implies $\bar{q}_i(c_\alpha) < u_\alpha$.

(ii) $\alpha_i = \alpha$, $\alpha \in \mathcal{T} - \{tick\}$ iff $l_\alpha \leq \bar{q}_i(c_\alpha) \leq u_\alpha$.

A condition equivalent to (i) is that for all $c_\alpha \in C$, $\bar{q}_i(c_\alpha) \leq u_\alpha$. Note that any loop of transitions in a TTM (a sequence of transitions starting and ending in the same activity) must have at least one transition with a non-zero upper time bound. Otherwise, once the first transition of the loop is enabled, our transition rules could possibly force an infinite number of non-*tick* transitions to occur without being interleaved by an infinite number of *tick*s.

As a small example, consider the TTM $M := \langle \mathcal{V}, \Theta, \mathcal{T} \rangle$ shown in Figure 2.2. The full enablement conditions for the transitions should also include conditions that enable the transitions only when the TTM is in activities that they exit in the transition diagram. For instance in the case of γ, the full enablement condition is $e_\gamma := v \geq 0 \wedge (x = a \vee x = b)$. When describing TTM transitions we will usually



$$\begin{aligned}
\mathcal{V} &:= \{u, v, t, x\} \\
\Theta &:= u = 0 \land v = 1 \land x = a \\
\mathcal{T} &:= \{\alpha := (u \geq 0, [u : u + v], 0, 2), \\
&\quad\ \beta := (\mathit{true}, [u : u + 1, v : v - 1], 2, \infty), \\
&\quad\ \gamma := (v \geq 0, [\ ], 2, 2), \\
&\quad\ \mathit{tick} := (\mathit{true}, [t : t + 1], -, -)\}
\end{aligned}$$

Figure 2.2: An example of a simple TTM

omit these activity variable conditions since they are obvious from the transition diagram. From the above discussion it is apparent that the definition of a transition such as $\gamma \in \mathcal{T}$ can result in several arrows with the same label in a TTM transition graph. To allow us to distinguish between a transition and the arrows that it defines in a transition diagram, we will call the arrows in the transition diagram *instances* of the transitions they are labeled by. In the example TTM M, there is an instance of transition γ exiting activity a and another instance exiting activity b. Finally, the special transition *tick* is declared to be in $\mathcal{T}$ and may be omitted from future listings of transition sets.

In writing out the operation functions of the transitions of M we employ a version of Ostroff's assignment format. When a transition occurs, the new value of the activity variable x is obtained from the transition diagram. The other variables that are affected by the transition are listed in the form

$$\begin{gathered}
[v_1 : \mathit{expr}_{11}, v_2 : \mathit{expr}_{12}, \ldots, v_n : \mathit{expr}_{1n}; \\
v_1 : \mathit{expr}_{21}, v_2 : \mathit{expr}_{22}, \ldots, v_n : \mathit{expr}_{2n}; \\
\ldots; \\
v_1 : \mathit{expr}_{k1}, v_2 : \mathit{expr}_{k2}, \ldots, v_n : \mathit{expr}_{kn}]
\end{gathered}$$

with the interpretation that variables v_1 to v_n are assigned the new values given by the simultaneous evaluations of expressions $expr_{i1}$ to $expr_{in}$ respectively for some choice of $i = 1, \ldots, k$. Semicolons are used to separate different possible assignments of the variables in the next state when the operation function is nondeterministic. If the operation function is deterministic then no semicolons occur in the assignment format. The operation function acts as the identity on variables not listed in the assignment statement. For instance $h_\alpha := [u : u + v] = [u : u + v, v : v]$ for M above.

If we let the current state assignment be represented by a 4-tuple of the form (u, v, x, t), then a legal trajectory of M would be

$$q_0 \overset{tick}{\rightarrow} q_1 \overset{\alpha}{\rightarrow} q_2 \overset{tick}{\rightarrow} q_3 \overset{\gamma}{\rightarrow} q_4 \overset{tick}{\rightarrow} \ldots$$

$$(0,1,a,0) \overset{tick}{\rightarrow} (0,1,a,1) \overset{\alpha}{\rightarrow} (1,1,b,1) \overset{tick}{\rightarrow} (1,1,b,2) \overset{\gamma}{\rightarrow} (1,1,e,2) \overset{tick}{\rightarrow} \ldots$$

where from q_4 onward the trajectory is continued by an infinite string of *tick*s. Note that after the second occurrence of *tick*, γ is forced to occur. A *tick* could not take place from q_3 since γ has $u_\gamma = 2$ and, upon reaching q_3, e_γ has been true for two *tick*s already.

If the initial condition for M is $\Theta := (u = 0 \wedge v = -1 \wedge x = a)$, then a trajectory that by the above definition is "legal" is

$$(0,-1,a,0) \overset{\alpha}{\rightarrow} (-1,-1,b,0) \overset{tick}{\rightarrow} (-1,-1,b,1) \overset{tick}{\rightarrow} (-1,-1,b,2) \overset{tick}{\rightarrow} \ldots$$

where again this trajectory is continued by an infinite number of *tick* transitions. This trajectory illustrates our interpretation of $u_\beta = \infty$. We do not insist on "fairness," allowing trajectories such as the one above where β is a possible next transition for an infinitely long time, although it does not occur. Thus an upper time bound of ∞ means that a transition is possible but is not forced to occur in a legal trajectory.

Occasionally we will use the transition graph representation of a TTM, where each instance of a transition in the TTM is represented as shown in Figure 2.3. This can be informally interpreted as follows: "if the TTM is currently in activity a_s and if e_α

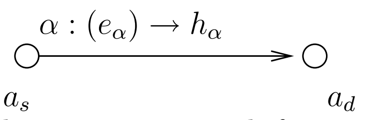


Figure 2.3: The transition graph format of a TTM

evaluates to *true*, then the edge labeled by α may be traversed while doing operation h_α, after which the TTM is in activity a_d." We will usually use this style of displaying TTM's when the time bounds are understood or not of particular importance.

To be useful for designing real systems, a formalism must provide a means of decomposing large systems into smaller, more manageable subsystems. Complex systems are then typically constructed from interacting components running in parallel. In [Ost90] Ostroff defines a TTM parallel composition operator that allows for shared variables and synchronous (shared) transitions. We extend this TTM parallel composition operator to handle nondeterministic operation functions. In the following definition we denote the state assignments over a set of variables $\mathcal{V}$ by $\mathcal{Q}_\mathcal{V} := \times_{v \in \mathcal{V}} Range(v)$. For $\mathcal{U} \subseteq \mathcal{V}$ the *natural state assignment projection* $P_\mathcal{U} : \mathcal{Q}_\mathcal{V} \rightarrow \mathcal{Q}_\mathcal{U}$ maps a state assignment over $\mathcal{V}$ to its corresponding state assignment over $\mathcal{U}$. In order to allow us to distinguish between a transition and its label, for $\mathcal{T}$, a given set of transitions (labeled 4-tuples), let $\Sigma(\mathcal{T})$ denote the set of transition labels. For the example TTM of Figure 2.2, $\Sigma(\mathcal{T}) = \{\alpha, \beta, \gamma, tick\}$. We are now ready to define the parallel composition of two TTMs.

Definition 2.7 *Given two TTMs* $M_i := \langle \mathcal{V}_i, \Theta_i, \mathcal{T}_i \rangle, i = 1, 2$*, the* parallel composition of M_1 and M_2 *is given by* $M_1 \| M_2 := \langle \mathcal{V}_1 \cup \mathcal{V}_2, \Theta_1 \wedge \Theta_2, \mathcal{T}_1 \| \mathcal{T}_2 \rangle$*, where the composite transition set* $\mathcal{T}_1 \| \mathcal{T}_2$ *is defined as follows.*

(i) If $\alpha := (e, h, l, u) \in \mathcal{T}_1$ *with operation function* $h : \mathcal{Q}_{\mathcal{V}_1} \rightarrow \mathcal{P}(\mathcal{Q}_{\mathcal{V}_1})$ *and* $\alpha \notin \Sigma(\mathcal{T}_2)$ *(the* α *transition label does not occur in* M_2*), then* $\alpha := (e, h', l, u) \in \mathcal{T}_1 \| \mathcal{T}_2$ *where* $h' : \mathcal{Q}_{\mathcal{V}_1 \cup \mathcal{V}_2} \rightarrow \mathcal{P}(\mathcal{Q}_{\mathcal{V}_1 \cup \mathcal{V}_2})$ *is the extension of* h *given by* $h' := h \otimes id_{Q_{\mathcal{V}_2 \setminus \mathcal{V}_1}}$.

(ii) Similarly if $\alpha := (e, h, l, u) \in \mathcal{T}_2$ *and* $\alpha \notin \Sigma(\mathcal{T}_1)$*, then* $\alpha := (e, h', l, u) \in \mathcal{T}_1 \| \mathcal{T}_2$ *where* $h' : \mathcal{Q}_{\mathcal{V}_1 \cup \mathcal{V}_2} \rightarrow \mathcal{P}(\mathcal{Q}_{\mathcal{V}_1 \cup \mathcal{V}_2})$ *is the extension of* h *given by* $h' := id_{Q_{\mathcal{V}_1 \setminus \mathcal{V}_2}} \otimes h$.

(iii) If α *is a shared transition, i.e.* $\alpha \in \Sigma(\mathcal{T}_1) \cap \Sigma(\mathcal{T}_2)$*, with* $\alpha := (e_1, h_1, l_1, u_1) \in$

$\mathcal{T}_1$ and $\alpha := (e_2, h_2, l_2, u_2) \in \mathcal{T}_2$ and operation functions $h_i : \mathcal{Q}_{\mathcal{V}_i} \to \mathcal{P}(\mathcal{Q}_{\mathcal{V}_i}), i = 1, 2$ then $\alpha := (e', h', l', u') \in \mathcal{T}_1 \| \mathcal{T}_2$ where

$e' := e_1 \wedge e_2$ is the enablement condition.

$h' : \mathcal{Q}_{\mathcal{V}_1 \cup \mathcal{V}_2} \to \mathcal{P}(\mathcal{Q}_{\mathcal{V}_1 \cup \mathcal{V}_2})$ is the function such that

$$h'(q) := \{q' \in \mathcal{Q}_{\mathcal{V}_1 \cup \mathcal{V}_2} : P_{\mathcal{V}_1}(q') \in h_1 \circ P_{\mathcal{V}_1}(q) \textit{ and } P_{\mathcal{V}_2}(q') \in h_2 \circ P_{\mathcal{V}_2}(q)\}$$

$l' := max(l_1, l_2)$ is the lower time bound.

$u' := min(u_1, u_2)$ is the upper time bound.

Condition (i) states that if the transition $\alpha := (e, h, l, u)$ of M_1 is not a shared transition then the new operation function in the composite system is given by $h'(q) = \{q' \in \mathcal{Q}_{\mathcal{V}_1 \cup \mathcal{V}_2} : P_{\mathcal{V}_1}(q') \in h \circ P_{\mathcal{V}_1}(q)$ and $P_{\mathcal{V}_2 \setminus \mathcal{V}_1}(q') = P_{\mathcal{V}_2 \setminus \mathcal{V}_1}(q)\}$. The value of variables not in M_1's variable set (i.e. $v \in \mathcal{V}_2 \setminus \mathcal{V}_1$) are left unchanged by a transition occurring only in M_1. Condition (iii) requires that any new assignment to the shared variables $(\mathcal{V}_1 \cap \mathcal{V}_2)$ made by a shared α transition must be possible assignments by α in both M_1 and M_2.

As an example, suppose M_1 and M_2 share the variable v and the transition label α. If $\alpha := (x_1 = a \wedge u = 0, [u : 1, v : 2; v : 1], 4, \infty)$ in M_1 and $\alpha := (x_2 = b, [v : 2, w : 1; v : 1, w : 0; v : 3, w : 0], 0, 5)$, then in $M_1 \| M_2$ we have

$$\alpha := (x_1 = a \wedge u = 0 \wedge x_2 = b, [u : 1, v : 2, w : 1; v : 1, w : 0], 4, 5)$$

The case when v is set to 3 by α in M_2 does not occur in the composite transition since no matching assignment of v to 3 can be made by α in M_1.

Now let us consider a transition that is not shared. Suppose $\beta := (x_1 = b, [u : 3, v : 4], 0, 1)$ is a transition of M_1 and the transition label β does not occur in M_2. Then $\beta := (x_1 = b, [u : 3, v : 4], 0, 1)$ is a transition of $M_1 \| M_2$. In this case the full operation function would be $[u : 3, v : 4, w : w, x_2 : x_2]$ and the new value of M_1's activity variable x_1 would be obtained from the graph of M_1. Thus an occurrence of β in the composite system does not affect M_2's private variables w and x_2.

The above TTM parallel composition operator places only minimal restrictions on the way variables and transitions interact in the composite system. Any TTM can arbitrarily access and modify another TTM's variables when the TTMs are composed. In Section 4.4 we restrict the way in which system components can interact through the definition of TTM Modules. The restrictions imposed upon TTM Modules will allow us to apply, at the TTM level, the compositional model reduction results of Chapter 4 for the less complex setting of State-Event Labeled Transition Systems.

2.2.3 State-Event Labeled Transition Systems

State-Event Labeled Transition Systems (SELTS) extend Labeled Transition Systems (LTS) [DeN87] by adding a state output map. In the temporal logic setting of Chapter 4 the state output will be the set of atomic propositions satisfied by a state. Until then we will consider the state output to be some kind of state observation. While (state based) Kripke structures are generally used as the underlying model for temporal logic model checkers [CES86] and are ultimately the model we would employ in any model checking algorithm for the temporal logics of Chapter 4, considering structures that are extended by transition labels has two main benefits. First, the use of *tick* transitions provides an easy method of incorporating system components' timing information in a concurrent setting. An example of such a use of SELTS is Ostroff's RG2 graphs [Ost89] that are used for model checking Real-Time Temporal Logic (RTTL) properties [Ost90]. RG2 graphs use event information to reduce infinite state timed systems, to finite state systems that preserve the relative timing of state changes and event outputs in a concurrent setting. Secondly, the addition of event information is also crucial for performing synchronous composition of systems and thereby permitting supervisory control through the disablement of controllable events [RW87].

Definition 2.8 *A* **State-Event Labeled Transition System** *(SELTS) is a 5-tuple $\mathbb{Q} := \langle Q, \Sigma, R_\Sigma, q_0, P\rangle$ where Q is an at most countable set of states, Σ is a finite set of elementary actions or events, $R_\Sigma = \{\xrightarrow{\alpha} : \alpha \in \Sigma\}$ is a set of binary relations on*

Q, $q_0 \in Q$ is the initial state and $P : Q \rightarrow R$ is the state output map, a function mapping each state into the set of state outputs.

In the above definition if $\alpha \in \Sigma$ and $q, q' \in Q$, then $q \xrightarrow{\alpha} q'$ means that the SELTS can move from state q to q' by executing elementary action α. Any transition relation $\xrightarrow{\alpha} \in R_\Sigma$ can be viewed as a function $\alpha^{\mathbb{Q}} : Q \rightarrow \mathcal{P}(Q)$, where $\mathcal{P}(Q)$ is the power set of Q. The function $\alpha^{\mathbb{Q}}$ maps q to the set of states reachable from q via a single α transition in the SELTS $\mathbb{Q}$. When the SELTS to which we are referring is obvious from the context, we will simply write $\alpha(q)$. For simplicity we assume $Q \neq \emptyset$ and $|Q|$ is finite. When discussing SELTS in Chapter 4, $AP, AP_1, AP_2, \ldots$ will represent sets of atomic propositions and the SELTS state output map will map each state to the set of atomic propositions satisfied by the state (ie. $P : Q \rightarrow \mathcal{P}(AP)$). For Chapter 3 it will suffice to consider state output maps of the more general form $P : Q \rightarrow R$ where R is an arbitrary set of state outputs.

A notion similar to LTS forms the basis of TTMs and many other models of concurrency. With the additional state output map, a SELTS provides a convenient way of modeling the state and event dynamics of a TTM. Figure 2.4 is the RG2 graph representing all legal trajectories of the simple TTM shown in Figure 2.2. The top line of each state in the graph contains the state assignments of the system variables in the format (u, v, x). The second line of each state contains the current values of each transition's counter variable in the format $[c_\alpha, c_\beta, c_\gamma]$. Thus the states of the RG2 graph are elements of M's set of extended state assignments $\overline{\mathcal{Q}}$. Note that in accordance with the fact that both of their lower time bounds equal 2, the β and γ transitions only exit states in which their respective counter variables equal or exceed 2. On the other hand both α and γ have upper time bounds of 2 so no *tick* transition exits a state where $c_\alpha = 2$ or $c_\gamma = 2$. In such states an α or γ transition is forced before the next clock *tick* unless it is preempted by another transition. For example, γ can be preempted by the β transition that enters the state in the graph's lower right corner. The initial state q_0 of the graph is indicated by an entering arrow. A TTM's legal trajectories are all infinite sequences and as can be seen from Figure 2.4, every path starting from q_0 can be extended to an infinite path. The transitions' counter

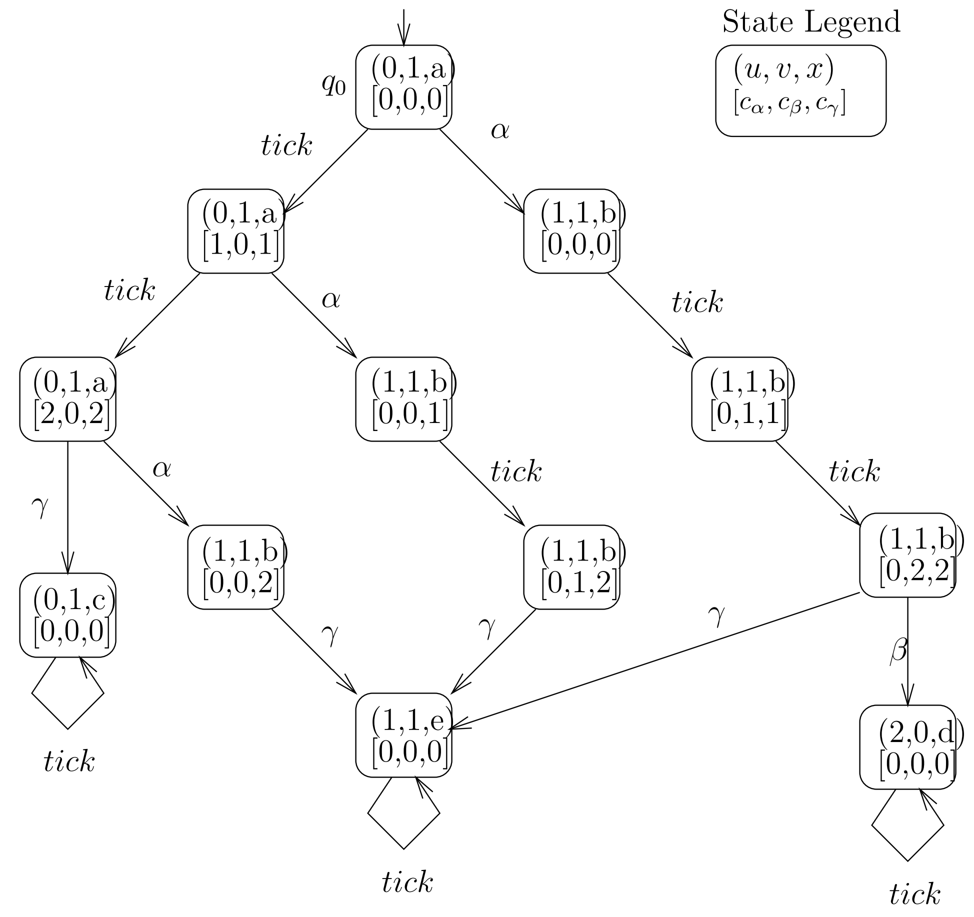


Figure 2.4: RG2 representing the legal trajectories of TTM M in Figure 2.2

variables are only used to obtain the structure of the graph. They are not part of the system's observed timed behavior. The counter variables are hidden variables, the values of which are crucial to determining the Markovian dynamics of the structure. Thus if we were to treat the RG2 graph of Figure 2.4 as a SELTS, the state output map would be the canonical projection from extended state assignments to state assignments $P : \overline{\mathcal{Q}} \rightarrow \mathcal{Q}$.

Although the β transition of M has an upper time bound of ∞, the RG2 graph (and hence the SELTS of M) is finite state since γ preempts β, preventing an infinite number of *tick*s from causing c_β from becoming unbounded. What if γ also had an upper time bound of ∞? How do we generate a finite state representation of the timed behavior of M?

The set of extended state assignments is reduced to produce a finite state set by redefining the Range of the counter variables as follows. For $M := \langle \mathcal{V}, \Theta, \mathcal{T} \rangle$ and $\alpha := (e, h, l, u) \in \mathcal{T}$

$$Range_M(c_\alpha) := \begin{cases} \{n \in \mathbb{N} : n < l\} \cup \{\omega\}, & \text{if } u = \infty \\ \{n \in \mathbb{N} : n \leq u\}, & u < \infty \end{cases}$$

If α has a finite upper time bound u_α, then TTM semantics prevent c_α from being incremented to a value exceeding u_α. For transitions with lower time bound l_α and upper time bound $u_\alpha = \infty$ when c_α is incremented to a value equal to l_α, we instead set $c_\alpha = \omega$ and henceforth define $\omega + 1 = \omega$. For comparison purposes we define for all $a \in \mathbb{N}, a < \omega < \infty$. We now redefine the set of extended state assignments to use this reduction as follows $\overline{\mathcal{Q}} := \mathcal{Q} \times \prod_{c_\alpha \in C} Range_M(c_\alpha)$ where $C := \{c_\alpha : \alpha \in \mathcal{T} - \{tick\}\}$. Henceforth when refering to the set of extended state assignments we will assume that we are dealing with the reduced set. We now formally define the SELTS obtained using the redefined extended state assignments to be the *SELTS generated by M*.

Definition 2.9 *Given a TTM $M := \langle \mathcal{V}, \Theta, \mathcal{T} \rangle$ with a finite RG2 graph, the* TTM generated by *M is defined as:*

$$\mathbb{Q}_M := \langle Q, \Sigma, R_\Sigma, q_0, P \rangle$$

where $Q = \overline{\mathcal{Q}}$ is the set of extended state assignments, and $\Sigma = \Sigma(\mathcal{T})$ is the set of transition labels for M. The transition relations of R_Σ are obtained from the definition of TTM semantics and $P : \overline{\mathcal{Q}} \to \mathcal{Q}$ is the canonical projection from extended state assignments to state assignments. The initial state $q_0 = \bar{q}_0$ is the unique extended state assignment such that $P(\bar{q}_0)$ satisfies Θ and $\bar{q}_0(c_\alpha) = 0$ for each TTM transition counter variable c_α.

Often a TTM's activity variable x plays a role similar to the counter variables in that it is only used to keep track of when transitions might possibly be enabled. Similarly, not all transition labels may be of significance. For instance M may be designed to share α and *tick* transitions while β and γ represent transitions that are internal to M. If one's real interest in the TTM M was the timed behavior of the variables u and v and the occurrence of α transitions, then this could be represented by the SELTS shown in Figure 2.5. We maintain the structure of $\mathbb{Q}_M$, the SELTS generated by M, and drop the extraneous information associated with the activity variable x and transition counter variables to obtain the SELTS's state output map $P' = P_{\mathcal{Q}_{\{u,v\}}} \circ P$ where $P_{\mathcal{Q}_{\{u,v\}}} : \mathcal{Q} \to \mathcal{Q}_{\{u,v\}}$ is the canonical projection from M's state assignments to the state assignments over $\{u, v\}$. The new state output values are shown as labels of the various cells of $ker(P')$ (eg. in state q_0, $(u, v) = P(q_0) = (0, 1)$). The SELTS transitions formerly labeled by γ and β have been relabeled as "unobservable" τ transitions since we do not need to distinguish them.

In defining TTM modules later we will find use for this process of "relabeling" a SELTS and so formalize the definition here.

Definition 2.10 *Given a SELTS $\mathbb{Q} := \langle Q, \Sigma, R_\Sigma, q_0, P\rangle$ where $P : Q \to R$, a* SELTS relabeling *is defined to be a pair of maps $r := (r_\Sigma, r_P)$, $r_\Sigma : \Sigma \to \Sigma'$ and $r_P : R \to R'$. The r* relabeling of $\mathbb{Q}$ *is given by:*

$$r(\mathbb{Q}) := \langle Q, (r_\Sigma)_*(\Sigma), R_{(r_\Sigma)_*(\Sigma)}, q_0, r_P \circ P\rangle$$

where $R_{(r_\Sigma)_*(\Sigma)}$ is obtained from R_Σ by replacing each transition $q \xrightarrow{\alpha} q'$ by $q \xrightarrow{r(\alpha)} q'$.

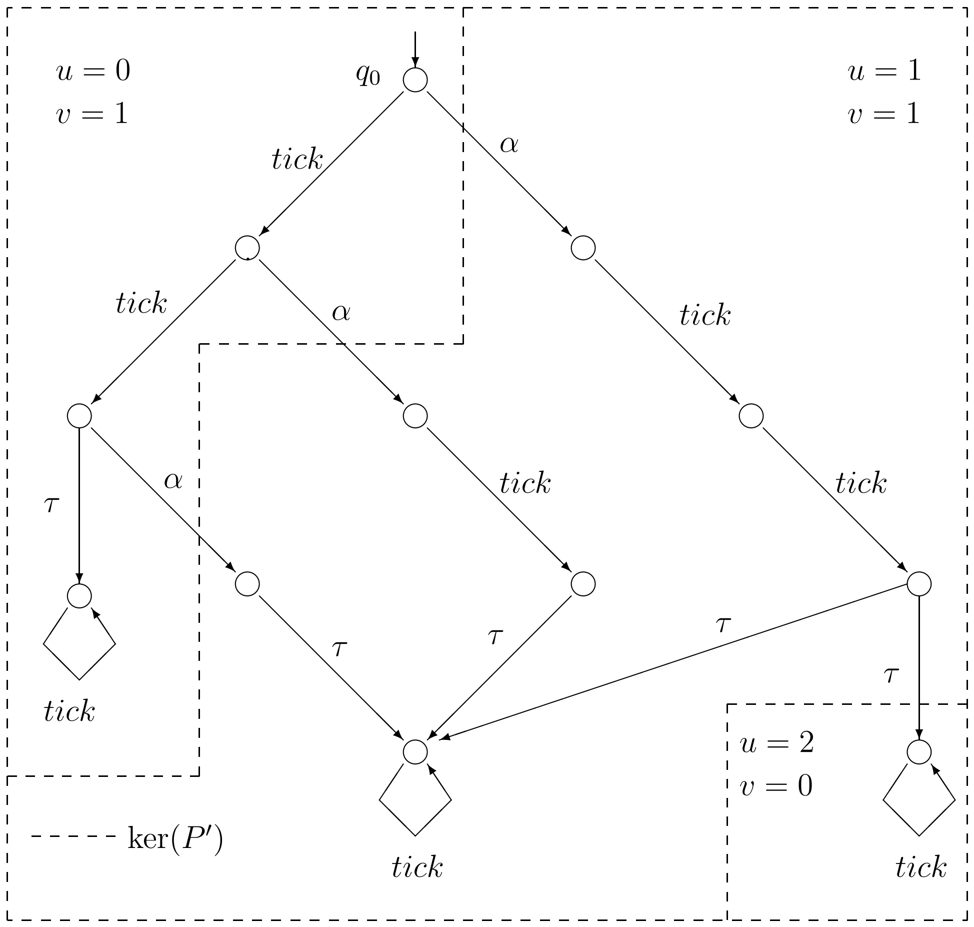


Figure 2.5: SELTS for timed behavior of u, v

We now define synchronous composition operators to provide a mechanism for constructing large systems consisting of interacting subsystems. Initially we deal with a strictly event based synchronization operator which is then extended to a variation of the more general state-event synchronization operator found in [GL93]. The event synchronous product operator below is a straightforward extension to the SELTS setting of the parallel composition operator of [Mil89].

Definition 2.11 *Given two SELTS, $\mathbb{Q}_i = \langle Q_i, \Sigma_i, R^i_\Sigma, q_{i0}, P_i \rangle$ with $P_i : Q_i \to R_i$ for $i = 1, 2$ and a set of synchronization events $\Sigma_s \subseteq \Sigma_1 \cap \Sigma_2$, the* Σ_s-synchronous product of $\mathbb{Q}_1$ and $\mathbb{Q}_2$ *is given by:* $\mathbb{Q}_1|[\Sigma_s]|\mathbb{Q}_2 := \langle Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, R_{\Sigma_1 \cup \Sigma_2}, (q_{10}, q_{20}), P_1 \times P_2 \rangle$, *where the elements of $R_{\Sigma_1 \cup \Sigma_2} = \{\xrightarrow{\alpha} : \alpha \in \Sigma_1 \cup \Sigma_2\}$ are binary relations over $Q_1 \times Q_2$ defined as follows: $(q_1, q_2) \xrightarrow{\alpha} (q'_1, q'_2)$ iff*

(i) $\alpha \in \Sigma_s$, and $q_i \xrightarrow{\alpha} q'_i$ in $\mathbb{Q}_i$ for $i = 1, 2$, or

(ii) $\alpha \notin \Sigma_s$, $q_1 \xrightarrow{\alpha} q'_1$ in $\mathbb{Q}_1$ and $q_2 = q'_2$, or

(iii) $\alpha \notin \Sigma_s$, $q_2 \xrightarrow{\alpha} q'_2$ in $\mathbb{Q}_2$ and $q_1 = q'_1$.

When $\Sigma_s = \Sigma_1 \cap \Sigma_2$ then the above definition of synchronous product specializes to the standard synchronous product operator used in [RW87]. In its more general form when $\Sigma_s \neq \Sigma_1 \cap \Sigma_2$, it is possible for an $\alpha \in \Sigma_1 \cap \Sigma_2 \setminus \Sigma_s$ to be executed independently by each subsystem and thereby introduce additional nondeterminism. In fact, in this case the synchronous product of two simple deterministic systems can result in a nondeterministic system (see Figure 2.6). Since our theory is specifically designed to deal with the nondeterminism that typically results from creating hierarchical models, this and the following generalization of the synchronous product operator do not pose a problem.

By viewing the transition relations as functions from states to the power set of states and using the fact that $Q \times \emptyset = \emptyset \times Q = \emptyset$ for any set Q, we can formulate an alternative functional definition of the event synchronous product transition relations as follows. For $(q_1, q_2) \in Q_1 \times Q_2$

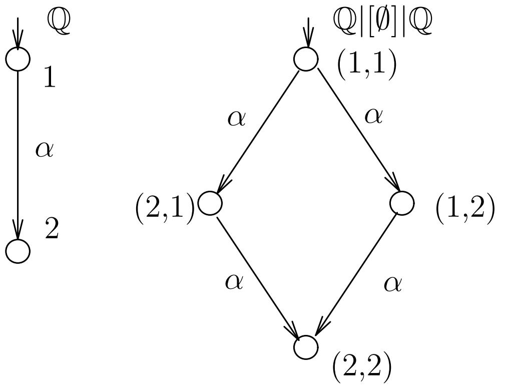


Figure 2.6: General synchronous product can create nondeterminism

$$\alpha^{\mathbb{Q}_1|[\Sigma_s]|\mathbb{Q}_2}(q_1, q_2) = \begin{cases} \alpha^{\mathbb{Q}_1}(q_1) \times \alpha^{\mathbb{Q}_2}(q_2), & \alpha \in \Sigma_s \\ (\alpha^{\mathbb{Q}_1}(q_1) \times \{q_2\}) \cup (\{q_1\} \times \alpha^{\mathbb{Q}_2}(q_2)), & \text{otherwise} \end{cases}$$

Thus we can use the setwise functional product operator to express $\alpha^{\mathbb{Q}_1|[\Sigma_s]|\mathbb{Q}_2}$ at a functional level that can then be used in composition with homomorphisms.

$$\alpha^{\mathbb{Q}_1|[\Sigma_s]|\mathbb{Q}_2} = \begin{cases} \alpha^{\mathbb{Q}_1} \otimes \alpha^{\mathbb{Q}_2}, & \alpha \in \Sigma_s \\ (\alpha^{\mathbb{Q}_1} \otimes id_{Q_2}) \cup (id_{Q_1} \otimes \alpha^{\mathbb{Q}_2}), & \text{otherwise} \end{cases}$$

Note that when $\alpha^{\mathbb{Q}_1|[\Sigma_s]|\mathbb{Q}_2}$ is applied to a state in the composite system $\alpha^{\mathbb{Q}_1} \otimes \alpha^{\mathbb{Q}_2}(q_1, q_2)$ results in the set of ordered state pairs given by $\alpha^{\mathbb{Q}_1}(q_1) \times \alpha^{\mathbb{Q}_2}(q_2)$ as stated above and *not* the ordered pair of sets $(\alpha^{\mathbb{Q}_1}(q_1), \alpha^{\mathbb{Q}_2}(q_2))$ (ie. $\alpha^{\mathbb{Q}_1|[\Sigma_s]|\mathbb{Q}_2} \neq \alpha^{\mathbb{Q}_1} \times \alpha^{\mathbb{Q}_2}$).

While the above event synchronous composition operator allows systems to synchronize on global *tick*s and other shared events, it does not have any way of modeling the "state output synchronization" associated with the shared variables of TTMs. As a first step towards representing TTM parallel composition at the SELTS level we extend event composition to state-event synchronous composition. We begin by defining the notion of a compatible interface for SELTS.

As a very general method of providing synchronization of state output changes, we assume that when two SELTS are composed, *state output synchronization maps* are associated with each system. These functions map the respective systems' state outputs to a common set. In the case of SELTS representing TTMs this common set will be the cross product of the ranges of variables shared by the TTMs (ie. $\mathcal{Q}_{\mathcal{V}_1 \cap \mathcal{V}_2}$). An SELTS interface will then be a set of synchronization events together with a pair of maps that have a common codomain. An interface will be compatible with a pair of SELTS if the state output synchronization maps are defined on appropriate domains and agree on their evaluation of the state outputs from their respective systems' initial states. More formally,

Definition 2.12 *Given two SELTS $\mathbb{Q}_i = \langle Q_i, \Sigma_i, R^i_\Sigma, q_{i0}, P_i \rangle$ with $P_i : Q_i \to R_i$ for $i = 1, 2$, a* compatible interface *for $\mathbb{Q}_1$ and $\mathbb{Q}_2$ is a 3-tuple $I := (\Sigma_s, f_1, f_2)$ where $\Sigma_s \subseteq \Sigma_1 \cap \Sigma_2$ is a set of synchronization events, and $f_i : R_i \to R$, $i = 1, 2$ are state output synchronization maps such that $f_1 \circ P_1(q_{10}) = f_2 \circ P_2(q_{20})$.*

In addition to the conditions imposed by event synchronization, for state-event composition we also require that systems "synchronize" on the value of the state output synchronization maps (ie. for any reachable state in the composite system (q_1, q_2), we have $f_1 \circ P_1(q_1) = f_2 \circ P_2(q_2)$).

Definition 2.13 *Given two SELTS, $\mathbb{Q}_i = \langle Q_i, \Sigma_i, R^i_\Sigma, q_{i0}, P_i \rangle$ with $P_i : Q_i \to R_i$ for $i = 1, 2$ and a compatible interface $I := (\Sigma_s, f_1, f_2)$, the I-*synchronous product *of $\mathbb{Q}_1$* and *$\mathbb{Q}_2$ is defined to be: $\mathbb{Q}_1|[I]|\mathbb{Q}_2 := \langle Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, R_{\Sigma_1 \cup \Sigma_2}, (q_{10}, q_{20}), P_1 \times P_2 \rangle$, where the elements of $R_{\Sigma_1 \cup \Sigma_2} = \{\xrightarrow{\alpha} : \alpha \in \Sigma_1 \cup \Sigma_2\}$ are binary relations over $Q_1 \times Q_2$ defined as follows: $(q_1, q_2) \xrightarrow{\alpha} (q_1', q_2')$ iff*

$$f_1 \circ P_1(q_1') = f_2 \circ P_2(q_2') \qquad (\dagger)$$

and

(i) $\alpha \in \Sigma_s$, and $q_i \xrightarrow{\alpha} q_i'$ in $\mathbb{Q}_i$ for $i = 1, 2$, or

(ii) $\alpha \notin \Sigma_s$, $q_1 \xrightarrow{\alpha} q_1'$ in $\mathbb{Q}_1$ and $q_2 = q_2'$, or

(iii) $\alpha \notin \Sigma_s$, $q_2 \xrightarrow{\alpha} q_2'$ in $\mathbb{Q}_2$ and $q_1 = q_1'$.

From the definition we see that $(q_1, q_2) \xrightarrow{\alpha} (q_1', q_2')$ in $\mathbb{Q}_1|[I]|\mathbb{Q}_2$ iff $(q_1, q_2) \xrightarrow{\alpha} (q_1', q_2')$ in $\mathbb{Q}_1|[\Sigma_s]|\mathbb{Q}_2$ and $f_1 \circ P_1(q_1') = f_2 \circ P_2(q_2')$.

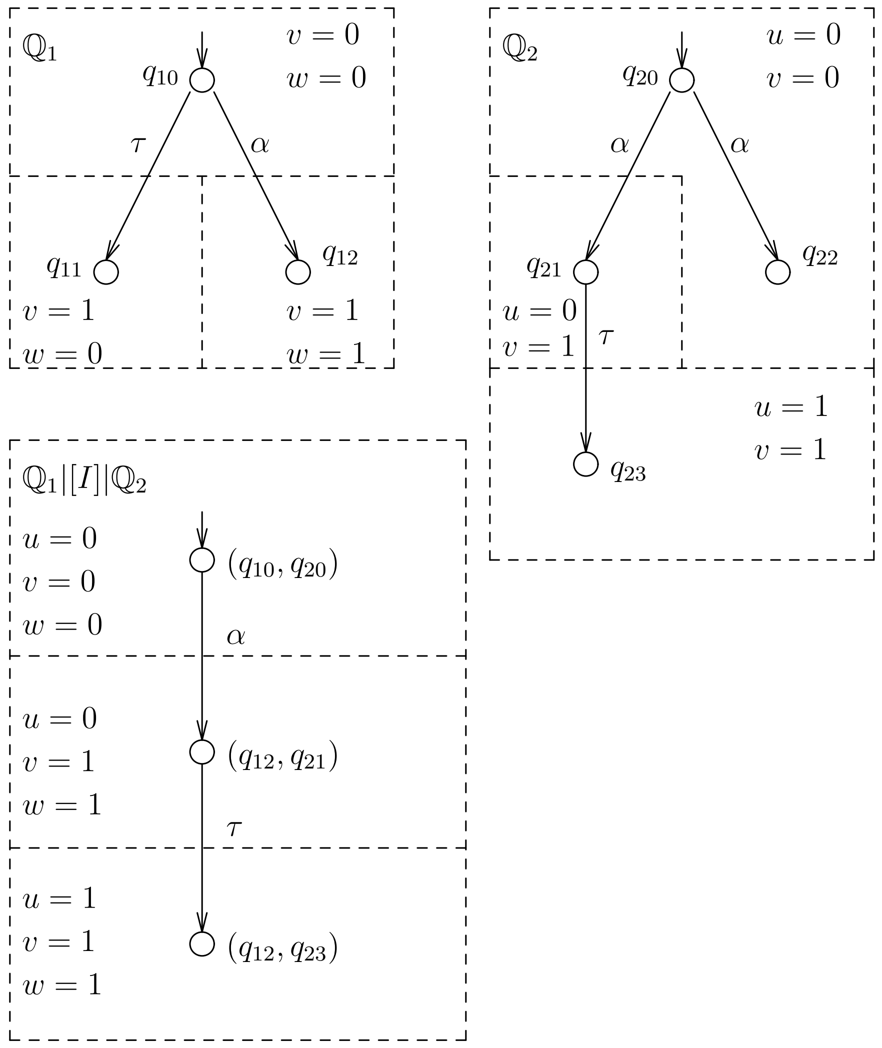


Figure 2.7: State-event synchronous product of $\mathbb{Q}_1$ and $\mathbb{Q}_2$ for $I := \{\{\alpha\}, \pi_1, \pi_2\}$

Figure 2.7 shows SELTS $\mathbb{Q}_1$ and $\mathbb{Q}_2$ and their state-event synchronous composition when synchronized on event α and value of their shared variable v. Formally both state output maps for $\mathbb{Q}_1$ and $\mathbb{Q}_2$ can be represented as functions from their state set to $\{0,1\}^2$ so $P_1(q_{11}) = (1,0)$ while $P_2(q_{21}) = (0,1)$ (ie. $P_1 : Q_1 \to \{0,1\}^2$ such that $q_1 \mapsto (v,w)$ for $q_1 \in Q_1$ and $P_2 : Q_2 \to \{0,1\}^2$ such that $q_2 \mapsto (u,v)$ for $q_2 \in Q_2$). So we can take the first state output synchronization map to be the canonical projection $\pi_1 : \{0,1\}^2 \to \{0,1\}$ where $(v,w) \mapsto v$ and the second state output synchronization map to be the canonical projection $\pi_2 : \{0,1\}^2 \to \{0,1\}$ where $(u,v) \mapsto v$. In this case $\pi_1 \circ P_1(q_{10}) = \pi_2 \circ P_2(q_{20})$ so $I := (\{\alpha\}, \pi_1, \pi_2)$ is a compatible interface for $\mathbb{Q}_1$ and $\mathbb{Q}_2$ and hence $\mathbb{Q}_1|[I]|\mathbb{Q}_2$ exists.

Closer examination of $\mathbb{Q}_1|[I]|\mathbb{Q}_2$ in Figure 2.7 reveals that only events from the synchronization set can modify shared variables (more generally, cause a change in the evaluations of the state output synchronization functions). The τ transition $(q_{10}, q_{20})\xrightarrow{\tau}(q_{11}, q_{20})$ would be allowed in $\mathbb{Q}_1|[\{\alpha\}]|\mathbb{Q}_2$ but does not take place in $\mathbb{Q}_1|[I]|\mathbb{Q}_2$ since $\mathbb{Q}_2$ cannot synchronize on τ to make a move to a new state that also changes the value of v to 1. Note that the τ transition $(q_{12}, q_{21})\xrightarrow{\tau}(q_{12}, q_{23})$ is allowed to take place in $\mathbb{Q}_1|[I]|\mathbb{Q}_2$, changing the value of the independent variable u, since it does not change the value of the shared variable v

For the synchronization event α, the transition $(q_{10}, q_{20})\xrightarrow{\alpha}(q_{12}, q_{22})$ cannot occur in $\mathbb{Q}_1|[I]|\mathbb{Q}_2$ because it would result in an inconsistent value of v since v changes from 0 to 1 when $q_{10}\xrightarrow{\alpha}q_{12}$ in $\mathbb{Q}_1$ but v retains the value of 0 when $q_{20}\xrightarrow{\alpha}q_{22}$ in $\mathbb{Q}_2$.

To obtain our arrow theoretic characterization of the state-event synchronous composition operator we can simply build upon the arrow theoretic definition of event synchronous composition. Recalling from Section 2.1.1 that the equalizer of a pair of functions with common domains and codomains, $f_i : A \to B$, $i = 1, 2$, is denoted by

$$eq(f_1, f_2) := \{a \in A : f_1(a) = f_2(a)\}$$

Considering the cross product $Q_1 \times Q_2$ with associate canonical projections $\pi_1 : Q_1 \times Q_2 \to Q_1$ and $\pi_2 : Q_1 \times Q_2 \to Q_2$, we can rephrase condition (†) of Definition 2.13

as $(q_1', q_2') \in eq(f_1 \circ P_1 \circ \pi_1, f_2 \circ P_2 \circ \pi_2)$. Thus for a compatible interface $I := (\Sigma_s, f_1, f_2)$ and SELTS as in Definition 2.13, regarding the transition relations as functions from the state set to the power set of states we have for $\alpha \in \Sigma_1 \cup \Sigma_2$

$$\begin{aligned} \alpha^{\mathbb{Q}_1|[I]|\mathbb{Q}_2} &= \alpha^{\mathbb{Q}_1|[\Sigma_s]|\mathbb{Q}_2} \cap eq(f_1 \circ P_1 \circ \pi_1, f_2 \circ P_2 \circ \pi_2) \\ &= \pi_{eq} \circ \alpha^{\mathbb{Q}_1|[\Sigma_s]|\mathbb{Q}_2} \end{aligned}$$

where $\pi_{eq} : \mathcal{P}(Q_1 \times Q_2) \to \mathcal{P}(Q_1 \times Q_2)$ is the projection resulting from intersection with the equalizer set $A \mapsto A \cap eq(f_1 \circ P_1 \circ \pi_1, f_2 \circ P_2 \circ \pi_2)$. Figure 2.8 illustrates the relationship between $|[\Sigma_s]|$ and $|[I]|$ as a commutative diagram.

$$\begin{tikzcd} Q_1 \times Q_2 \arrow[r, "\alpha^{\mathbb{Q}_1|[\Sigma_s]|\mathbb{Q}_1}"] \arrow[rr, bend right, "\alpha^{\mathbb{Q}_1|[I]|\mathbb{Q}_2}"'] & \mathcal{P}(Q_1 \times Q_2) \arrow[r, "\pi_{eq}"] & \mathcal{P}(Q_1 \times Q_2) \end{tikzcd}$$

Figure 2.8: Commutative diagram relating $|[\Sigma_s]|$ and $|[I]|$.

When the state output synchronization maps are constant over their domain (eg. the trivial maps $f_i : R_i \to \{\emptyset\}$, $q_i \mapsto \emptyset, i = 1, 2$), then $eq(f_1 \circ P_1 \circ \pi_1, f_2 \circ P_2 \circ \pi_2) = Q_1 \times Q_2$ so $\pi_{eq} = id_{\mathcal{P}(Q_1 \times Q_2)}$, the identity map. In this case $|[I]|$ reduces to $|[\Sigma_s]|$ as one might expect since the trivial state output synchronization maps provide synchronization of outputs for all states.

2.3 State Observers for a Class of Deterministic LTS

In this section the lattice of congruences of a deterministic transition system and its role in characterizing the (strong) state observers of [Won76] are reviewed.

In [Won76] the author considers SELTS of the form

$$\mathbb{Q} = \langle Q, \{\alpha\}, \{\xrightarrow{\alpha}\}, q_0, P \rangle$$

where $\xrightarrow{\alpha}$ is a deterministic transition relation (ie. the lone transition relation can be represented as a function $\alpha : Q \to Q$). In this case the author views the SELTS as a discrete time dynamical system, given by $x(0) = q_0$ and $x(t+1) = \alpha(x(t))$, where it is the sequence of states generated by the LTS that is of interest. The output map $P : Q \to R$ is assumed to have no special structure. Thus two states $q, q' \in Q$ produce the same output observation precisely when $P(q) = P(q')$.

Definition 2.14 *Given a deterministic SELTS $\mathbb{Q}$ as defined above, $\theta \in Eq(Q)$ is a* congruence *of the transition function α for $\mathbb{Q}$ iff $(q, q') \in \theta$ implies $(\alpha(q), \alpha(q')) \in \theta$. We let $Con(\mathbb{Q})$ denote the set of all congruences of the transition function for $\mathbb{Q}$.*

$Con(\mathbb{Q})$ forms a complete sublattice of $Eq(Q)$. Thus $Con(\mathbb{Q})$ is closed under $\wedge$ and $\vee$, and given any $\mathcal{F} \subseteq Con(\mathbb{Q})$, $\sup(\mathcal{F})$ exists as an element of $Con(\mathbb{Q})$.

Definition 2.15 *Given a deterministic SELTS $\mathbb{Q}$ as defined above and a state output map $P : Q \to R$, the* strong state observer*, $\theta_o(\mathbb{Q})$, is defined to be*

$$\theta_o(\mathbb{Q}) = \sup\{\theta \in Con(\mathbb{Q}) : \theta \leq \ker(P)\}$$

When $\mathbb{Q}$ is clear from the context we will simply write θ_o for $\theta_o(\mathbb{Q})$. The existence and uniqueness of θ_o are an immediate result of $Con(\mathbb{Q})$ being a complete sublattice of $Eq(Q)$. Here θ_o is the coarsest congruence with respect to the transition function α, that is finer than the equivalence kernel of P. For $(q, q') \in \theta_o$, $\theta_o \leq \ker(P)$ implies $P(q) = P(q')$ while $\theta_o \in Con(\mathbb{Q})$ so $(\alpha(q), \alpha(q')) \in \theta_o$ and hence $P(\alpha(q)) = P(\alpha(q'))$. Thus if $(q, q') \in \theta_o$, then q and q' produce the same current state output and sequence of future state outputs.

From an informational standpoint, θ_o represents the minimum information you need about the current state of the system to be able to predict the future state outputs.

Quotient Systems

Given a deterministic SELTS $\mathbb{Q} = \langle Q, \{\alpha\}, \{\xrightarrow{\alpha}\}, q_0, P\rangle$, for any $\theta \in \{\theta \in Con(\mathbb{Q}) : \theta \leq \ker(P)\}$ we can define the *quotient SELTS of* $\mathbb{Q}$ *by* θ as

$$\mathbb{Q}/\theta = \langle Q/\theta, \{\alpha\}, \{\xrightarrow{\alpha}_\theta\}, q_0/\theta, P_\theta\rangle$$

Here q_0/θ denotes the θ-cell (equivalence class) containing q_0 and Q/θ represents the set of all θ-cells. The transition relation $\{\xrightarrow{\alpha}_\theta\}$ can again be viewed as a function $\alpha^{\mathbb{Q}/\theta} : Q/\theta \to Q/\theta$ where for $q/\theta \in Q/\theta$, $\alpha^{\mathbb{Q}/\theta}(q/\theta) := \alpha^{\mathbb{Q}}(q)/\theta$. The state output map $P_\theta : Q/\theta \to R$ is the unique map such that $P_\theta \circ \theta = P$. The existence of P_θ follows from the fact that the partition θ is finer than $\ker(P)$ while the uniqueness of P_θ follows from the fact that the map $\theta : Q \to Q/\theta$ is onto.

Chapter 3

Observers for State-Event Labeled Transition Systems

In this chapter we introduce strong and weak state-event observers for State-Event Labeled Transition Systems. State output maps and event projections play symmetric roles. Our observers (congruences) induce consistent high-level abstractions (quotients) so that, just as in [ZW90], control designed at the abstract level can be consistently implemented at the detailed ('real-world') level.

The development of strong observers and their quotient systems in Section 3.1 parallels the results on indistinguishability of LTS in [Arn94]. On the basis of [KS83], [PT87], [BC89] we are able to appeal to efficient polynomial-time algorithms for computing our observers on finite-state SELTS. We end the section with some minimum realization results. In Section 3.2 the results are extended to the case when there is partial event information as well as partial state information. Section 3.3 provides a simple real-time system as an illustrative example of the theory discussed in the previous sections. We conclude the chapter with some key results on the compositional consistency of strong and weak state-event equivalence that will be used in the chapters on model reduction.

3.1 Strong State-Event Observers

We now wish to generalize the observers for deterministic SELTS with a single transition function to observers for general SELTS with multiple nondeterministic transition relations. In this case it is not only the state output sequences that are important, but also the connecting events (relations). This is illustrated by the following three sequences and their images under the state output map $P : Q \to R$.

$$\left.\begin{array}{l} q_{11}\xrightarrow{\tau}q_{12}\xrightarrow{\alpha}q_{13} \\ q_{21}\xrightarrow{\alpha}q_{22}\xrightarrow{\tau}q_{23} \\ q_{31}\xrightarrow{\tau}q_{32}\xrightarrow{\alpha}q_{33} \end{array}\right\} \overset{P}{\mapsto} \left\{\begin{array}{l} r_1\xrightarrow{\tau}r_1\xrightarrow{\alpha}r_2 \\ r_1\xrightarrow{\alpha}r_2\xrightarrow{\tau}r_2 \\ r_1\xrightarrow{\tau}r_2\xrightarrow{\alpha}r_2 \end{array}\right. \tag{3.1}$$

Later τ will be used to denote unobservable events but for now we assume that all τ transitions are observable. In this case the first output sequence differs from the other two in the second state output while the second and third differ in the ordering of their connecting relations or "events". Thus no two of these sequences of states and connecting events produce identical output sequences.

3.1.1 Compatible Partitions

Congruences are defined only for transition functions but we are now dealing with nondeterministic transition relations so we must find a class of partitions that plays the role of congruences for nondeterministic relations.

Definition 3.1 *Given a SELTS $\mathbb{Q} = \langle Q, \Sigma, R_\Sigma, q_0, P\rangle$, a partition $\theta \in Eq(Q)$ is a* compatible partition *for $\mathbb{Q}$ if for all $\alpha \in \Sigma$, whenever q, q' are in the same partition block (cell) C_i, then for any block C_j of θ,*

$$\alpha(q) \cap C_j \neq \emptyset \textit{ iff } \alpha(q') \cap C_j \neq \emptyset$$

The set of all compatible partitions for the SELTS $\mathbb{Q}$ will be denoted by $CP(\mathbb{Q})$.

From the above definition we see that for $\theta \in CP(\mathbb{Q})$ if $(q, q') \in \theta$ and $q\xrightarrow{\alpha}q_1$ then there exists q_1' such that $q'\xrightarrow{\alpha}q_1'$ and $(q_1, q_1') \in \theta$. The reader familiar with Milner's observa-

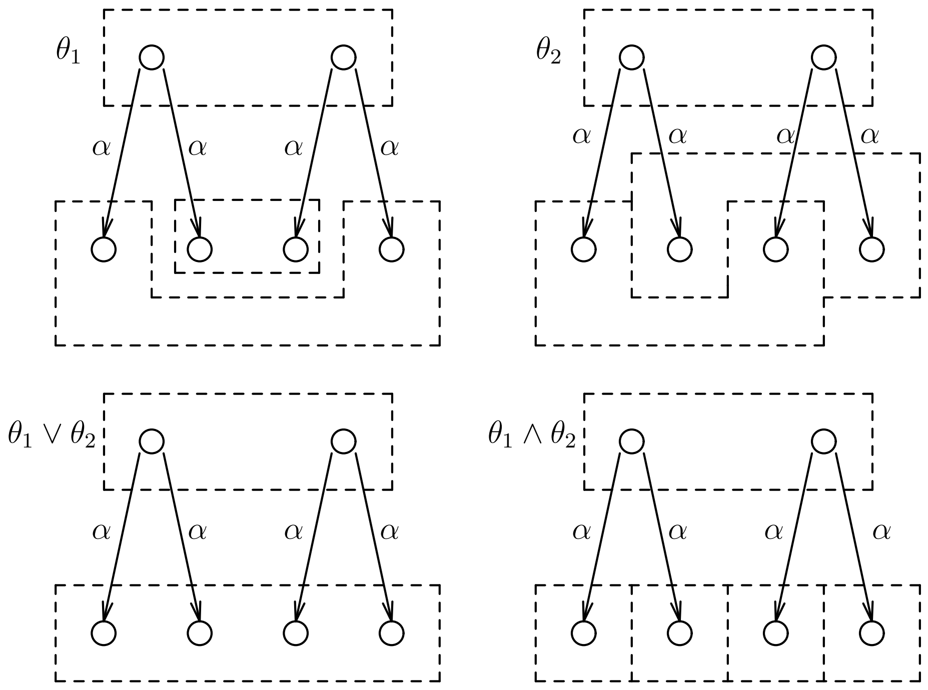


Figure 3.1: Compatible partitions are closed under $\vee$ but not $\wedge$

tion equivalence will note that compatible partitions are special cases of bisimulation relations and have been used for the efficient computation of (event) observation equivalence of LTS [KS83], [BC89]. We will have more to say about this later. First we will see if $CP(\mathbb{Q})$ has any special algebraic structure.

In the case of congruences, $Con(\mathbb{Q})$ forms a complete sublattice of $Eq(Q)$ so perhaps we can expect something similar for $CP(\mathbb{Q})$. Consider Figure 3.1. It is easy to verify that θ_1, θ_2 and $\theta_1 \vee \theta_2$ are compatible partitions of the given SELTS but $\theta_1 \wedge \theta_2$ is not. Thus $CP(\mathbb{Q})$ is not closed under the $\wedge$ operation of $Eq(Q)$. The following Lemma claims that $CP(\mathbb{Q})$ is closed under the $\vee$ operator of $Eq(Q)$ so although $CP(\mathbb{Q})$ is not a complete sublattice of $Eq(Q)$, it does retain the complete join semilattice property of $Con(\mathbb{Q})$ that was used in defining state observers in the Section 2.3. We were led to expect a join semilattice structure for defining observers on systems with nondeterministic transition relations from Wong's investigation of the algebraic properties of hierarchy in [Won94].

Lemma 3.2 *For a given SELTS $\mathbb{Q} = \langle Q, \Sigma, R_\Sigma, q_0, P \rangle$, the set of compatible partitions for $\mathbb{Q}$, $CP(\mathbb{Q})$, forms a complete sub-semilattice (with respect to join) of $Eq(Q)$,* *the lattice of equivalence relations on Q.*

Proof: We know that $Eq(Q)$ is a complete lattice so it only remains to show that $CP(\mathbb{Q})$ is closed under arbitrary join operations.

Let $\theta_i \in CP(\mathbb{Q})$ where I is an index set and write $\theta := \bigvee_{i \in I} \theta_i$. Suppose $(a, b) \in \theta$. Then by definition there exist $i_0, i_1, \ldots, i_k \in I$ such that

$$(a, b) \quad \in \quad \theta_{i_0} \circ \theta_{i_1} \circ \ldots \circ \theta_{i_k}.$$

That is there exist $a_0, \ldots, a_{k+1}$ such that $(a_j, a_{j+1}) \in \theta_{i_j}$ with $a_0 = a$ and $a_{k+1} = b$.

Assume that $\alpha(a) \neq \emptyset$ and let $c \in \alpha(a)$. We must show that there exists $d \in \alpha(b)$ such that $(c, d) \in \theta$. Now $(a, a_1) \in \theta_{i_0}$ and $\theta_{i_0} \in CP(\mathbb{Q})$ so there exists $c_1 \in \alpha(a_1)$ such that $(c, c_1) \in \theta_{i_0}$.

Inductively assume there exists $c_j \in \alpha(a_j)$ such that $(c_{j-1}, c_j) \in \theta_{i_{j-1}}$. Then for $(a_j, a_{j+1}) \in \theta_{i_j}$, $\theta_{i_j} \in CP(\mathbb{Q})$ so there exists $c_{j+1} \in \alpha(a_{j+1})$ such that $(c_j, c_{j+1}) \in \theta_j$.

Thus, by induction, with $a = a_0$, $b = a_{k+1}$ and $d = c_{k+1}$, we have

$$(c, d) \in \theta_{i_0} \circ \theta_{i_1} \circ \ldots \circ \theta_{i_k},$$

and hence we conclude that if $c \in \alpha(a)$ then there exists $d \in \alpha(b)$ such that $(c, d) \in \theta$. The argument is easily reversed by switching a and b giving us the desired result, $\theta \in CP(\mathbb{Q})$. $\square$

3.1.2 Computation of Strong State-Event Observers

An immediate result of Lemma 3.2 is that for any non-empty subset $\mathcal{F} \subseteq CP(\mathbb{Q})$, there is a unique supremal element $\theta^* := \sup(\mathcal{F})$ and $\theta^* \in CP(\mathbb{Q})$. We are now in a position to characterize a strong state-event observer for any given SELTS.

Definition 3.3 *Given a SELTS $\mathbb{Q} = \langle Q, \Sigma, R_\Sigma, q_0, P \rangle$ the* strong state-event observer*, $\theta_s(\mathbb{Q})$ is defined to be*

$$\theta_s(\mathbb{Q}) = \sup\{\theta \in CP(\mathbb{Q}) : \theta \leq \ker(P)\}.$$

Note that it is always the case that the trivial "bottom" partition $\Delta \in CP(\mathbb{Q})$ and for any state output map $P : Q \to R$, $\Delta = \ker(id_Q) \leq \ker(P)$ so $\theta_s(\mathbb{Q})$ always exists. When $\mathbb{Q}$ is clear from the context we will simply write θ_s for $\theta_s(\mathbb{Q})$. As was the case for the state observers of Section 2.3, θ_s is the coarsest compatible partition of $\mathbb{Q}$ that is finer than the equivalence kernel of the system's state output map P. Thus for $(q, q') \in \theta_s$ we have $P(q) = P(q')$ so q and q' produce the same current state output. Now suppose that $q \xrightarrow{\alpha} q_1$, thereby producing event output α and state output $P(q_1)$. Since $\theta_s \in CP(\mathbb{Q})$ there exists $q'_1 \in \alpha(q')$ such that $(q_1, q'_1) \in \theta_s$. Hence $q' \xrightarrow{\alpha} q'_1$ and $P(q_1) = P(q'_1)$ so q' can generate identical state and event outputs to q. As was the case with state observers, θ_s represents the minimum information one needs about the current state to be able to predict all *possible* future state and event outputs. We say "possible" future outputs since the general SELTS dealt with by state-event observers are nondeterministic. Hence knowing the cell of θ_s that a state belongs to lets one know what *may* happen, not what *will* happen, in contrast with the case with the state observers for deterministic SELTS.

The Relational Coarsest Partition problem (RCP) (as stated in [KS83]) can be phrased "Given a LTS $\mathbb{Q} = \langle Q, \Sigma, R_\Sigma, q_0 \rangle$ and θ_0, an initial partition of Q, find the coarsest compatible partition of $\mathbb{Q}$ that is finer than θ_0 (ie. find $\sup\{\theta \in CP(\mathbb{Q}) : \theta \leq \theta_0\}$)." Thus θ_s is the solution to the RCP with $\theta_0 := \ker(P)$. In the special case when $\theta_0 = \ker(P) = \nabla$ (no state information is provided by the state output map), the solution of the RCP is Milner's strong observation equivalence $\sim$ [KS83]. Therefore when there are only event outputs and no state outputs, our strong state-event observers reduce to Milner's strong observation equivalence.

An $O(m \log n)$ algorithm, where m is the size of R_Σ (the number of related pairs) and $n = |Q|$, for computing $\sim$ for finite state LTS, based upon Paige and Tarjan's solution to the (mono)-RCP (RCP with only one relation present) [PT87], can be found in [BC89]. In this case θ_0 is, of course, ∇. This algorithm is easily adapted to computing θ_s without any change in complexity (assuming $\ker(P)$ is provided) by allowing the initial partition for the RCP to be $\ker(P)$ which, in general, is not ∇. This close connection with $\sim$ leads us to write $q \sim_{se} q'$ when$(q, q') \in \theta_s$ and say that

q is strong state-event observation equivalent to q'.

What differentiates our work from that of [KS83] and [BC89], is the use, as suggested in [Arn94], of a nontrivial initial partition in the RCP, to consider both event and state outputs. The consideration of both state and event outputs takes on greater significance when we consider weak state-event observers in the next section. With little additional effort we can adapt [KS83] and [BC89] to provide an efficient algorithm for computing weak state-event observers.

3.1.3 Strong Quotient Systems and Homomorphisms

As a generalization of congruences, we might expect that compatible partitions can be used to construct quotient systems of nondeterministic SELTS (and their underlying LTS).

Definition 3.4 *Given a SELTS* $\mathbb{Q} := \langle Q, \Sigma, R_\Sigma, q_0, P\rangle$*, for* $\theta \in CP(\mathbb{Q})$ *such that* $\theta \leq \ker(P)$*, we define the* quotient system of $\mathbb{Q}$ by θ*,* $\mathbb{Q}/\theta$*, as follows:*

$$\mathbb{Q}/\theta := \langle Q/\theta, \Sigma, R_\Sigma/\theta, q_0/\theta, P_\theta\rangle$$

Here q_0/θ *denotes the cell of the partition* θ *containing* q_0 *and* Q/θ *denotes the set of all cells of* θ*. For* $\alpha \in \Sigma$*, the transition relations of* R_Σ/θ *are defined as* $\alpha^{\mathbb{Q}/\theta}(q/\theta) = \alpha^{\mathbb{Q}}(q)/\theta = \{q_1/\theta \in Q/\theta : q_1 \in \alpha^{\mathbb{Q}}(q)\}$*.* $P_\theta : Q/\theta \to R$ *is the unique map such that* $P_\theta \circ \theta = P$*.*

The existence of P_θ follows from the fact that $\theta \leq \ker(P)$ while uniqueness is guaranteed by the fact that $\theta : Q \to Q/\theta$ is onto.

The remainder of this section is dedicated to proving that the quotient system generated by the compatible partition θ_s is the "unique" (up to isomorphism) minimal state SELTS that is strongly state-event (observationally) equivalent to the original system. To do this we first have to have a definition of when two SELTS are state-event equivalent.

As was the case with observation equivalence in [DeN87], strong state-event observation equivalence can be extended to a relation $\sim_{se}$ between two *disjoint SELTS*, SELTS having disjoint state sets and state output maps. This is done by forming the union of the transition systems and the disjoint union of the original systems' state output maps. The two SELTS are then strongly state-event equivalent iff their initial states are strongly state-event observationally equivalent in the union system. More formally,

Definition 3.5 *Given two disjoint SELTS $\mathbb{Q}_i = \langle Q_i, \Sigma, R^i_\Sigma, q_{i0}, P_i\rangle$ with state output maps $P_i : Q_i \to R$ for $i = 1, 2$, we define the union of $\mathbb{Q}_1$ and $\mathbb{Q}_2$ to be*

$$\mathbb{Q}_1 \cup \mathbb{Q}_2 := \langle Q_1 \cup Q_2, \Sigma, R^1_\Sigma \cup R^2_\Sigma, q_{10}, P_1 \dot{\cup} P_2\rangle$$

Here the disjoint union of the state output functions, $P_1\dot{\cup}P_2 : Q_1 \cup Q_2 \to R$, is given by

$$P_1\dot{\cup}P_2(q) = \begin{cases} P_1(q), & \text{for } q \in Q_1 \\ P_2(q), & \text{for } q \in Q_2 \end{cases}$$

We then say that $\mathbb{Q}_1$ is strongly state-event equivalent *to $\mathbb{Q}_2$, written $\mathbb{Q}_1 \sim_{se} \mathbb{Q}_2$, iff $(q_{10}, q_{20}) \in \theta_s(\mathbb{Q}_1 \cup \mathbb{Q}_2)$.*

In the definition of $\mathbb{Q}_1 \cup \mathbb{Q}_2$ we have made the arbitrary choice of q_{10} as the initial state. We could just as easily use q_{20} as the initial state. Either one will do for our purposes of proving properties of quotient systems.

The notion of a homomorphism of a SELTS will, of course, play a central role in obtaining our results about quotient systems. The nondeterministic transition relations lead us to extend the notion of homomorphism in much the same way that we extended congruences of deterministic SELTS to compatible partitions of nondeterministic SELTS. Figure 3.2 illustrates the idea of a SELTS homomorphism. Any α move in the low level system can be matched by an α move in the high level system and vice versa. In addition, for a mapping to be a SELTS homomorphism, we also require that the initial state of the low level SELTS be mapped to the initial state of the high level SELTS. In the definition of a SELTS homomorphism we use the fact

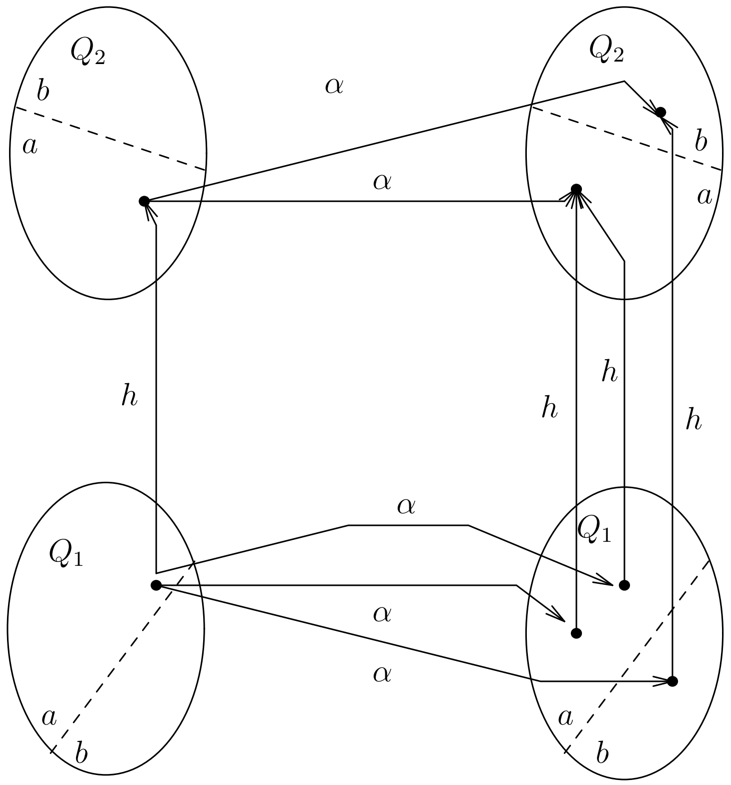


Figure 3.2: Graphical interpretation of a SELTS homomorphism

that any function $h : Q_1 \to Q_2$ induces a function at the power set level using the lifting operator, $h_* : \mathcal{P}(Q_1) \to \mathcal{P}(Q_2)$ (see Subsection 2.1.2).

Definition 3.6 *Given two SELTS $\mathbb{Q}_i = \langle Q_i, \Sigma, R^i_\Sigma, q_{i0}, P_i \rangle$ for $i = 1, 2$, a mapping $h : Q_1 \to Q_2$ is a* SELTS homomorphism *from $\mathbb{Q}_1$ to $\mathbb{Q}_2$ if*

(i) $h(q_{10}) = q_{20}$

(ii) For all $\alpha \in \Sigma, h_* \circ \alpha^{\mathbb{Q}_1} = \alpha^{\mathbb{Q}_2} \circ h$

(iii) $P_1 = P_2 \circ h$

In this case we will write $h : \mathbb{Q}_1 \to \mathbb{Q}_2$. Henceforth homomorphism will be understood to mean SELTS homomorphism. Any map satisfying (i) and (ii) will be said to be a LTS homomorphism *of the SELTS's underlying LTS.*

The relationships between the various maps for a SELTS homomorphism are displayed as the commutative diagram of Figure 3.3.

$$\begin{tikzcd}
 & Q_2 \arrow[ld, "P_2"'] \arrow[r, "\alpha^{\mathbb{Q}_2}"] & \mathcal{P}(Q_2) \arrow[rd, "(P_2)_*"] & \\
R & & & \mathcal{P}(R) \\
 & Q_1 \arrow[lu, "P_1"] \arrow[uu, "h"] \arrow[r, "\alpha^{\mathbb{Q}_1}"] & \mathcal{P}(Q_1) \arrow[uu, "h_*"'] \arrow[ru, "(P_1)_*"'] &
\end{tikzcd}$$

Figure 3.3: Commutative diagram for an SELTS homomorphism

Typically, the composition of homomorphisms is also a homomorphism. SELTS homomorphisms are no exception in this regard.

Lemma 3.7 *Given SELTS homomorphism $h_1 : \mathbb{Q}_1 \to \mathbb{Q}_2$ and $h_2 : \mathbb{Q}_2 \to \mathbb{Q}_3$,*

$$h_2 \circ h_1 : \mathbb{Q}_1 \to \mathbb{Q}_3$$

That is, $h_2 \circ h_1 : Q_1 \to Q_3$ is an SELTS homomorphism.

Proof: The composition h_1 followed by h_2 takes the initial state of $\mathbb{Q}_1$ to the initial state of $\mathbb{Q}_3$ as $h_2 \circ h_1(q_{10}) = h_2(h_1(q_{10})) = h_2(q_{20}) = q_{30}$ since h_1 and h_2 are both homomorphisms from $\mathbb{Q}_1$ to $\mathbb{Q}_2$ and $\mathbb{Q}_2$ to $\mathbb{Q}_3$ respectively. Similarly for the state output maps $P_1 = P_2 \circ h_1 = (P_3 \circ h_2) \circ h_1 = P_3 \circ (h_2 \circ h_1)$. Thus we need only show that $(h_2 \circ h_1)_* \circ \alpha^{\mathbb{Q}_1} = \alpha^{\mathbb{Q}_3} \circ (h_2 \circ h_1)$.

$$\begin{aligned}
(h_2 \circ h_1)_* \circ \alpha^{\mathbb{Q}_1} &= (h_2)_* \circ (h_1)_* \circ \alpha^{\mathbb{Q}_1}, \text{ by Claim 2.3} \\
&= (h_2)_* \circ ((h_1)_* \circ \alpha^{\mathbb{Q}_1}) \\
&= (h_2)_* \circ (\alpha^{\mathbb{Q}_2} \circ h_1), \text{ by def. of SELTS homomorphism.} \\
&= ((h_2)_* \circ \alpha^{\mathbb{Q}_2}) \circ h_1 \\
&= (\alpha^{\mathbb{Q}_3} \circ h_2) \circ h_1, \text{ by def. of SELTS homomorphism.} \\
&= \alpha^{\mathbb{Q}_3} \circ (h_2 \circ h_1)
\end{aligned}$$

□

If $\mathbb{Q}_2$ is a homomorphic image of $\mathbb{Q}_1$, any event and associated state output change in $\mathbb{Q}_1$ can be matched in $\mathbb{Q}_2$. This situation leads us to expect that homomorphisms and compatible partitions are closely related. This relationship is the subject of the following three lemmas.

Lemma 3.8 *Given a SELTS $\mathbb{Q} := \langle Q, \Sigma, R_\Sigma, q_0, P\rangle$, any $\theta \in CP(\mathbb{Q})$ defines a (natural) LTS homomorphism $\theta : Q \to Q/\theta$ of the underlying LTS of $\mathbb{Q}$ and its quotient system $\mathbb{Q}/\theta$.*

Proof: Let $\theta : Q \to Q/\theta$ be the map that takes each element of Q to its θ-cell. Then by definition of $\mathbb{Q}/\theta$ we have $\theta(q_0) = q_0/\theta$. It remains to show that for any $\alpha \in \Sigma$ and any $q \in Q$ we have

$$\theta_*(\alpha^{\mathbb{Q}}(q)) = \alpha^{\mathbb{Q}/\theta}(\theta(q))$$

Let $x' \in \theta_*(\alpha^{\mathbb{Q}}(q))$. Then there exists $q' \in \alpha^{\mathbb{Q}}(q)$ with $\theta(q') = x'$. But $q \xrightarrow{\alpha} q'$ and hence $\theta(q) \xrightarrow{\alpha} \theta(q')$ in $\mathbb{Q}/\theta$ by definition. Thus $x' = \theta(q') \in \alpha^{\mathbb{Q}/\theta}(\theta(q))$ giving

$$\theta_*(\alpha^{\mathbb{Q}}(q)) \subseteq \alpha^{\mathbb{Q}/\theta}(\theta(q))$$

Reversing the above argument gives the desired result. $\square$

Corollary 3.9 *For state output map $P : Q \to R$, if $\theta \in \{\theta \in CP(\mathbb{Q}) : \theta \leq \ker(P)\}$ then $\theta : \mathbb{Q} \to \mathbb{Q}/\theta$ is a SELTS homomorphism.*

The above lemma shows us that any compatible partition defines a LTS homomorphism and any compatible partition finer than the equivalence kernel of the state output map results in a SELTS homomorphism. The next lemma demonstrates that there is a compatible partition associated with every LTS homomorphism.

Lemma 3.10 *If $h : Q_1 \to Q_2$ is a LTS homomorphism for the underlying transition systems of $\mathbb{Q}_i := \langle Q_i, \Sigma, R^i_\Sigma, q_{0i}, P_i\rangle$ for $i = 1, 2$, then $\ker(h) \in CP(\mathbb{Q}_1)$.*

Proof: Suppose $(q, q') \in \ker(h)$. We want to show that if $q \xrightarrow{\alpha} q_1$, then there exists $q_1' \in Q_1$ such that $q' \xrightarrow{\alpha} q_1'$ and $(q_1, q_1') \in \ker(h)$.

Clearly $h(q_1) \in h_*(\alpha^{\mathbb{Q}_1}(q))$. But h is a LTS homomorphism and $h(q) = h(q')$ so

$$h_*(\alpha^{\mathbb{Q}_1}(q)) = \alpha^{\mathbb{Q}_2}(h(q)) = \alpha^{\mathbb{Q}_2}(h(q')) = h_*(\alpha^{\mathbb{Q}_1}(q'))$$

And $h(q_1) \in h_*(\alpha^{\mathbb{Q}_1}(q'))$ iff there exists $q_1' \in \alpha^{\mathbb{Q}_1}(q')$ such that $h(q_1') = h(q_1)$. Thus $(q_1, q_1') \in \ker(h)$ and $q' \xrightarrow{\alpha} q_1'$. Hence $\ker(h) \in CP(\mathbb{Q}_1)$. □

Corollary 3.11 *If* $h : \mathbb{Q}_1 \to \mathbb{Q}_2$ *is a SELTS homomorphism, then* $\ker(h) \in \{\theta \in CP(\mathbb{Q}) : \theta \leq \ker(P_1)\}$

We can now talk about *output compatible partitions* – those partitions of a SELTS that correspond to the kernel of a homomorphism of the SELTS $\mathbb{Q}$. The next lemma states that the only output compatible partition of the quotient system generated by θ_s is the trivial partition Δ. Thus any homomorphism of $\mathbb{Q}/\theta_s$ is an isomorphism.

Lemma 3.12 *If* $\mathbb{Q} := \langle Q, \Sigma, R_\Sigma, q_0, P\rangle$ *is an SELTS then*

$$\{\theta \in CP(\mathbb{Q}/\theta_s) : \theta \leq \ker(P_{\theta_s})\} = \{\Delta\}$$

Proof: Suppose $\theta \in CP(\mathbb{Q}/\theta_s), \theta \leq \ker(P_{\theta_s})$. By Lemma 3.7 $\theta \circ \theta_s : \mathbb{Q} \to (\mathbb{Q}/\theta_s)/\theta$ is a SELTS homomorphism of $\mathbb{Q}$. Therefore $\ker(\theta \circ \theta_s) \in CP(\mathbb{Q})$ and $\ker(\theta \circ \theta_s) \leq \ker(P)$ by Corollary 3.11. But $\theta_s = \sup(\{\theta \in CP(\mathbb{Q}) : \theta \leq \ker(P)\})$ so $\ker(\theta \circ \theta_s) \leq \ker(\theta_s)$. Thus $\ker(\theta \circ \theta_s) = \ker(\theta_s)$ which implies $\theta = \Delta$. □

We are now ready to prove the main result of this section, which states that two SELTS are strongly state-event equivalent iff they share an output compatible homomorphic image. As a corollary to this theorem, with the help of Lemma 3.12, we obtain the result that when $\mathbb{Q}$ is reachable, $\mathbb{Q}/\theta_s$ is the unique minimal state SELTS that is strongly state-event equivalent to $\mathbb{Q}$.

Theorem 3.13 *For two disjoint SELTS* $\mathbb{Q}_1$ *and* $\mathbb{Q}_2$ *as in Definition 3.6, we have* $\mathbb{Q}_1 \sim_{se} \mathbb{Q}_2$ *iff there exists a SELTS* $\mathbb{Q}_3$ *for which there are homomorphisms* $h_1 : \mathbb{Q}_1 \to \mathbb{Q}_3$ *and* $h_2 : \mathbb{Q}_2 \to \mathbb{Q}_3$.

Proof: (if) Let $h := h_1 \dot{\cup} h_2$ (ie. $h : Q_1 \cup Q_2 \to Q_3$ be the map such that $h|Q_i = h_i, i = 1, 2$). Since $\mathbb{Q}_1$ and $\mathbb{Q}_2$ have disjoint state sets and h_1 and h_2 are SELTS homomorphisms, it follows that

$$h : \mathbb{Q}_1 \cup \mathbb{Q}_2 \to \mathbb{Q}_3$$

is a homomorphism with $h(q_{10}) = h(q_{20}) = q_{30}$. Therefore $\ker(h) \in \{\theta \in CP(\mathbb{Q}_1 \cup \mathbb{Q}_2) : \theta \leq \ker(P_1 \dot{\cup} P_2)\}$ and $(q_{10}, q_{20}) \in \ker(h)$ which implies that $(q_{10}, q_{20}) \in \theta_s(\mathbb{Q}_1 \cup \mathbb{Q}_2)$. Hence, by definition, $\mathbb{Q}_1 \sim_{se} \mathbb{Q}_2$.

(only if) Take $\theta = \theta_s(\mathbb{Q}_1 \cup \mathbb{Q}_2)$. By Corollary 3.9

$$\theta : \mathbb{Q}_1 \cup \mathbb{Q}_2 \to (\mathbb{Q}_1 \cup \mathbb{Q}_2)/\theta$$

is a SELTS homomorphism. Take $\mathbb{Q}_3 := (\mathbb{Q}_1 \cup \mathbb{Q}_2)/\theta$, $h_1 := \theta|Q_1$ and $h_2 := \theta|Q_2$. From the fact that $\mathbb{Q}_1$ and $\mathbb{Q}_2$ have disjoint transition relations, it follows that h_1 and h_2 are SELTS homomorphisms. □

Corollary 3.14 *For any reachable SELTS $\mathbb{Q}$, the quotient system $\mathbb{Q}/\theta_s$ is the unique (up to isomorphism), minimal state SELTS such that $\mathbb{Q} \sim_{se} \mathbb{Q}/\theta_s$.*

Proof: We know that $\theta_s : \mathbb{Q} \to \mathbb{Q}/\theta_s$ is a homomorphism. Also the identity map on Q/θ_s is an SELTS homomorphism $I_{Q/\theta_s} : \mathbb{Q}/\theta_s \to \mathbb{Q}/\theta_s$. Thus $\mathbb{Q} \sim_{se} \mathbb{Q}/\theta_s$ by Theorem 3.13.

Uniqueness follows from Lemma 3.12. If $\mathbb{Q} \sim_{se} \mathbb{Q}_2$ we may assume that $\mathbb{Q}_2$ is reachable since $\mathbb{Q}$ is reachable. Otherwise we can just take the reachable part of $\mathbb{Q}_2$ and it will still be equivalent to $\mathbb{Q}$. It follows that $\mathbb{Q}/\theta_s \sim_{se} \mathbb{Q}_2$ so by Theorem 3.13 there exists $\mathbb{Q}_3$ with SELTS homomorphisms $h_1 : \mathbb{Q}/\theta_s \to \mathbb{Q}_3$ and $h_2 : \mathbb{Q}_2 \to \mathbb{Q}_3$. But by Lemma 3.12 h_1 is an isomorphism. Thus h_1^{-1} is a homomorphism and hence so is $h_1^{-1} \circ h_2 : \mathbb{Q}_2 \to \mathbb{Q}/\theta_s$. Therefore $|Q/\theta_s| \leq |Q_2|$, giving us uniqueness up to isomorphism. □

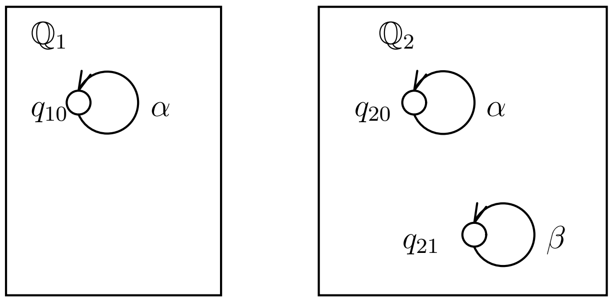


Figure 3.4: State-event equivalent SELTS quotient systems that are not isomorphic

Figure 3.4 demonstrates why we require $\mathbb{Q}$ to be reachable in Corollary 3.14. $\mathbb{Q}_2$ is not reachable and as can be easily verified, $\mathbb{Q}_2/\theta_s = \mathbb{Q}_2$. But $\mathbb{Q}_1 \sim_{se} \mathbb{Q}_2$ and $|Q_1| < |Q_2|$.

From the above result we now derive the more precise result stated in Corollary 3.15. Two reachable SELTS are strongly state-event equivalent iff their strong state-event observer quotient systems are isomorphic.

Corollary 3.15 *Let $\mathbb{Q}_1$ and $\mathbb{Q}_2$ be reachable SELTS and $\theta_{si} = \theta_s(\mathbb{Q}_i), i = 1, 2$. Then $\mathbb{Q}_1 \sim_{se} \mathbb{Q}_2$ iff $\mathbb{Q}_1/\theta_{s1}$ and $\mathbb{Q}_2/\theta_{s2}$ are isomorphic.*

Proof: (only if) By Corollary 3.14 $\mathbb{Q}_1/\theta_{s1} \sim_{se} \mathbb{Q}_2/\theta_{s2}$. Hence by Theorem 3.13 there exists $\mathbb{Q}_3$ with homomorphisms $h_1 : \mathbb{Q}_1/\theta_{s1} \to \mathbb{Q}_3$ and $h_2 : \mathbb{Q}_2/\theta_{s2} \to \mathbb{Q}_3$. But by Lemma 3.12 both h_1 and h_2 are isomorphisms. Therefore $h_1^{-1} \circ h_2$ is an isomorphism.

(if) Assume $\mathbb{Q}_1/\theta_{s1}$ and $\mathbb{Q}_2/\theta_{s2}$ are isomorphic. Then there exists a homomorphism $h : \mathbb{Q}_1/\theta_{s1} \to \mathbb{Q}_2/\theta_{s2}$. Also, by Corollary 3.9, $\theta_{si} : \mathbb{Q}_i \to \mathbb{Q}_i/\theta_{si}$ for $i = 1, 2$ are homomorphisms. Thus by Lemma 3.7 $h \circ \theta_{s1} : \mathbb{Q}_1 \to \mathbb{Q}_2/\theta_{s2}$. Since $\theta_{s2} : \mathbb{Q}_2 \to \mathbb{Q}_2/\theta_{s2}$ we can apply Theorem 3.13 to get $\mathbb{Q}_1 \sim_{se} \mathbb{Q}_2$ as required. □

The simple SELTS in Figure 3.4 illustrates why both $\mathbb{Q}_1$ and $\mathbb{Q}_2$ are required to be reachable in Corollary 3.15. Clearly for any state output maps such that $P_1(q_{10}) = P_2(q_{20})$ we have $\mathbb{Q}_1 \sim_{se} \mathbb{Q}_2$ since then $q_{10} \sim_{se} q_{20}$ in $(\mathbb{Q}_1 \cup \mathbb{Q}_2)$. But in this case both systems are their own quotient systems and are clearly not isomorphic.

We now use the result of Corollary 3.15 to prove a version of the "diamond property" of [Arn94] for SELTS homomorphism. As we will see, the diamond property

$$\begin{tikzcd}
 & \mathbb{Q}_1 \arrow[dl, "h_{12}"'] \arrow[dr, "h_{13}"] & \\
\mathbb{Q}_2 \arrow[dr, dashed, "h_{24}"'] & & \mathbb{Q}_3 \arrow[dl, dashed, "h_{34}"] \\
 & \mathbb{Q}_4 &
\end{tikzcd}$$

Figure 3.5: Commutative diagram for the diamond property of SELTS homomorphisms

is the key to providing the transitivity for the homomorphism version of state-event equivalence. The following corollary states that the diagram of Figure 3.5 commutes.

Corollary 3.16 *Given any pair of homomorphisms* $h_{12} : \mathbb{Q}_1 \to \mathbb{Q}_2$ *and* $h_{13} : \mathbb{Q}_1 \to \mathbb{Q}_3$*, there exists a pair of homomorphisms* $h_{24} : \mathbb{Q}_2 \to \mathbb{Q}_4$ *and* $h_{34} : \mathbb{Q}_3 \to \mathbb{Q}_4$ *such that* $h_{24} \circ h_{12} = h_{34} \circ h_{13}$.

Proof: If $\mathbb{Q}_2$ and $\mathbb{Q}_3$ are reachable then the result is immediate by Corollary 3.15 since we can take $\mathbb{Q}_4 := \mathbb{Q}_1/\theta_s(\mathbb{Q}_1)$. In this case for $i = 2, 3$ $\mathbb{Q}_i \sim_{se} \mathbb{Q}_1/\theta_s(\mathbb{Q}_1)$ since $\mathbb{Q}_i \sim_{se} \mathbb{Q}_1$ and $\mathbb{Q}_1 \sim_{se} \mathbb{Q}_1/\theta_s(\mathbb{Q}_1)$. The existence of h_{24} and h_{34} is then guaranteed by Theorem 3.13.

The general case is handled in a similar fashion by taking $\mathbb{Q}_4$ to be the strong state-event quotient system of the disjoint union of $\mathbb{Q}_2$ and $\mathbb{Q}_3$. □

The relationship to transitivity of Figure 3.5 is seen in Figure 3.6 where the diagram of Figure 3.5 occurs as a sub-diagram of the larger diagram. This larger diagram is then guaranteed to commute as an immediate result of the previous corollary. Thus SELTS homomorphisms can be used to provide an alternative definition of $\sim_{se}$ by saying that $\mathbb{Q}_1 \sim_{se} \mathbb{Q}_2$ iff there exists a SELTS $\mathbb{Q}_3$ for which there are homomorphisms $h_1 : \mathbb{Q}_1 \to \mathbb{Q}_3$ and $h_2 : \mathbb{Q}_2 \to \mathbb{Q}_3$. For this alternative definition of strong state-event equivalence, idempotence is guaranteed by the fact that the identity map $id_{Q_1} : \mathbb{Q}_1 \to \mathbb{Q}_1$ is a homomorphism, symmetry follows from the symmetry of the definition and transitivity follows from Figure 3.6. Theorem 3.13 guarantees

$$\begin{tikzcd}
\mathbb{Q}_5 \arrow[dr, "h_{52}"] & & \mathbb{Q}_1 \arrow[dl, "h_{12}"'] \arrow[dr, "h_{13}"] & & \mathbb{Q}_6 \arrow[dl, "h_{63}"'] \\
& \mathbb{Q}_2 \arrow[dr, dashed, "h_{24}"'] & & \mathbb{Q}_3 \arrow[dl, dashed, "h_{34}"] & \\
& & \mathbb{Q}_4 & &
\end{tikzcd}$$

Figure 3.6: Commutative diagram for the transitivity of SELTS homomorphism definition of $\sim_{se}$

that this alternative definition of $\sim_{se}$ coincides with the original compatible partition definition.

3.2 Weak State-Event Observers

Often in Discrete Event Systems it is the case that systems are event- rather than time-driven. In this case what is important is the sequence of changes in the outputs, ignoring intermediate states and events that do not generate any new outputs. Before applying this point of view in our state event setting, we will see how it is applied in the event setting of Milner's weak observation equivalence. Again we will see that (event) observation equivalence becomes the special case of our setting in which $\ker(P) = \nabla$.

Consider a LTS $\mathbb{Q} := \langle Q, \Sigma, R_\Sigma, q_0 \rangle$. In the style of [BC89], we assume there is a "silent event" $\tau \in \Sigma$ that represents unobservable actions. We then define the set of observable actions to be $\Sigma_o := \Sigma - \{\tau\}$. This leads to some new relations on Q. We say that q moves unobservably (from an event perspective) to q', written $q \overset{\tau}{\Rightarrow} q'$, iff there exist $q_0, q_1, \ldots, q_n \in Q$, $n \geq 0$, such that

$$q = q_0 \xrightarrow{\tau} q_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} q_{n-1} \xrightarrow{\tau} q_n = q'$$

By convention, for any $q \in Q$, $q \overset{\tau}{\Rightarrow} q$. For $\alpha \in \Sigma_o$ we can then say that q moves to q'

while producing event α, written $q\overset{\alpha}{\Rightarrow}q'$, iff there exist $q_1, q_2 \in Q$ such that

$$q\overset{\tau}{\Rightarrow}q_1\overset{\alpha}{\rightarrow}q_2\overset{\tau}{\Rightarrow}q'$$

In the weakly observable setting the actions $q\overset{\alpha}{\rightarrow}q'$ and $q\overset{\alpha}{\Rightarrow}q'$ are indistinguishable since both produce the single event output α. For a given $\mathbb{Q}$, these double arrow relations can be used to define a new transition system,

$$\mathbb{Q}' := \langle Q, \Sigma, R'_\Sigma, q_0 \rangle$$

where R'_Σ is defined as follows. For all $\alpha \in \Sigma_o$, $\alpha^{\mathbb{Q}'}(q) = \{q_1 \in Q : q\overset{\alpha}{\Rightarrow}q_1 \text{ in } \mathbb{Q}\}$ and $\tau^{\mathbb{Q}'}(q) = \{q_1 \in Q : q\overset{\tau}{\Rightarrow}q_1 \text{ in } \mathbb{Q}\}$.

In [KS83], two states are shown to be weakly observation equivalent in $\mathbb{Q}$ in the sense of [Mil80], written $q \approx q'$, iff the states are strongly observation equivalent ($q \sim q'$) in $\mathbb{Q}'$. Thus we have $\approx := \sup(CP(\mathbb{Q}'))$. In this case $\approx$ represents the minimum information one needs about Q to know what choices of future observable events are possible.

We now generalize weak observation equivalence to our state-event setting. Given a SELTS $\mathbb{Q} := \langle Q, \Sigma, R_\Sigma, q_0, P \rangle$ where, as usual, $P : Q \to R$ is the state output map, we assume that the special event τ represents unobservable events. When a τ transition occurs, it does not produce an output event, though it may cause a change in the state output. For instance, if $q\overset{\tau}{\rightarrow}q'$ and $P(q) = P(q')$ then there is no noticeable change in the system output. If, on the other hand, $q\overset{\tau}{\rightarrow}q'$ and $P(q) \neq P(q')$ then although no event is seen to take place, a change in state output takes place when τ occurs. This leads us to define, for a given SELTS $\mathbb{Q}$, an unobservable move from q to q', written $q \Rightarrow_{se} q'$ iff there exist $q_0, q_1, \ldots, q_n \in Q$, $n \geq 0$, such that

$$q = q_0\overset{\tau}{\rightarrow}q_1\overset{\tau}{\rightarrow}\ldots\overset{\tau}{\rightarrow}q_{n-1}\overset{\tau}{\rightarrow}q_n = q'$$

and for all $j = 0, 1, \ldots, n$ we have $P(q_j) = P(q) = P(q')$

Thus the relation $\Rightarrow_{se}$ is the transitive closure of the τ relation within each cell of $\ker(P)$. By convention $q \Rightarrow_{se} q$ always holds. While the $\Rightarrow_{se}$ relation captures a relation which is indistinguishable from the case when $q \xrightarrow{\tau} q'$ and $P(q) = P(q')$, we now wish to define a relation which captures both this case and the case when $q \xrightarrow{\tau} q'$ and $P(q) \neq P(q')$. We say that q moves to q' without an event output, written $q \overset{\tau}{\Rightarrow}_{se} q'$, iff $q = q'$ or, there exist $q_1, q_2 \in Q$ such that

$$q \Rightarrow_{se} q_1 \xrightarrow{\tau} q_2 \Rightarrow_{se} q'$$

By definition $q \overset{\tau}{\Rightarrow}_{se} q$. The relation $\overset{\tau}{\Rightarrow}_{se}$ is the transitive closure of $\xrightarrow{\tau}$ subject to the restriction that at most one boundary of the partition $\ker(P)$ is crossed. If $q \overset{\tau}{\Rightarrow}_{se} q'$, then no output events are generated and there is at most one change in the state output.

We now define a relation similar to $\overset{\tau}{\Rightarrow}_{se}$ except that it produces exactly one event output. For $\alpha \in \Sigma_o$, we say that q moves to q' producing event output α, written $q \overset{\alpha}{\Rightarrow}_{se} q'$ iff there exist $q_1, q_2 \in Q$ such that

$$q \Rightarrow_{se} q_1 \xrightarrow{\alpha} q_2 \Rightarrow_{se} q'$$

Thus if $q \overset{\alpha}{\Rightarrow}_{se} q'$, then q moves within a cell of $\ker(P)$ via unobservable τ transitions, then performs an α transition which could possibly (but not necessarily) take us to a new cell of $\ker(P)$ and then the system again moves unobservably via τ transitions within the current cell. We emphasize that if a boundary of $\ker(P)$ is crossed when $q \overset{\alpha}{\Rightarrow}_{se} q'$, then it is only crossed by the α transition.

There are four different types of one step moves that a SELTS $\mathbb{Q}$ can make and each of these moves can be matched by a double arrow relation defined above. In the following let q and q' be elements of Q such that $P(q) = P(q')$. Then the system can:

1. Make an unobservable transition within a cell of $\ker(P)$ ($q \xrightarrow{\tau} q_1$ and $P(q) = P(q_1)$). State q' can make the move $q' \overset{\tau}{\Rightarrow}_{se} q'_1$ with $P(q'_1) = P(q_1)$ to produce the same (lack of) output.

2. Make a τ transition that moves from one cell of $\ker(P)$ to another ($q \xrightarrow{\tau} q_1$ and

$P(q) \neq P(q_1)$). State q' can make the move $q' \overset{\tau}{\Rightarrow}_{se} q_1'$ with $P(q_1') = P(q_1)$ to produce the same change in state output.

3. Make an observable transition α within a cell of $\ker(P)$ ($q \xrightarrow{\alpha} q_1$ and $P(q) = P(q_1)$). State q' can make the move $q' \overset{\alpha}{\Rightarrow}_{se} q_1'$ with $P(q_1') = P(q_1)$ to produce the same event output.

4. Make an observable transition α that moves from one cell of $\ker(P)$ to another ($q \xrightarrow{\alpha} q_1$ and $P(q) \neq P(q_1)$). State q' can make the move $q' \overset{\alpha}{\Rightarrow}_{se} q_1'$ with $P(q_1') = P(q_1)$ to produce the same event output and change of state output.

Consider the state event sequences (3.1) of Section 3.1 from the point of view that only output (observable) events and changes in the state output are important. The first two sequences are indistinguishable when viewed from state and event outputs. In both sequences the event α and the state output change from r_1 to r_2 occur simultaneously. Hence $q_{11} \overset{\alpha}{\Rightarrow}_{se} q_{13}$ and $q_{21} \overset{\alpha}{\Rightarrow}_{se} q_{23}$ and in both cases at the output it appears as $r_1 \xrightarrow{\alpha} r_2$. In the case of the third string, the state output changes with the unobservable transition τ and *then* the event α occurs. In terms of our newly defined relations $q_{31} \overset{\tau}{\Rightarrow}_{se} q_{32} \overset{\alpha}{\Rightarrow}_{se} q_{33}$ but not $q_{31} \overset{\alpha}{\Rightarrow}_{se} q_{33}$ and so at the outputs the third sequence appears as $r_1 \xrightarrow{\tau} r_2 \xrightarrow{\alpha} r_2$.

From a control point of view it is important that an observer be able to distinguish the first two sequences from the third. Assume that r_2 is a bad state output that we wish to avoid and that α is a controllable event that can be disabled as in [RW87]. Disabling α prevents state output r_2 from occurring in the first two sequences of (3.1) but not in the third sequence!

With the above examples in mind, we are ready to define weak state-event observers by first considering the transition system generated by the double arrow relations. We call the transition system generated by the double arrow relations the *observational closure* of the given SELTS. The observational closure of an SELTS is obtained using the observational closure operator defined below.

Definition 3.17 *Given a SELTS $\mathbb{Q} = \langle Q, \Sigma, R_\Sigma, q_0, P \rangle$, the* observational closure of $\mathbb{Q}$*, denoted $\mathbb{Q}'_{se}$ is given by the* observational closure operator $'_{se}$: *SELTS* $\rightarrow$ *SELTS* *as follows:*

$$\mathbb{Q}'_{se} := \langle Q, \Sigma, R'_\Sigma, q_0, P\rangle$$

where R'_Σ is defined as: For all $\alpha \in \Sigma$, $\alpha^{\mathbb{Q}'_{se}}(q) = \{q_1 \in Q : q \overset{\alpha}{\Rightarrow}_{se} q_1 \text{ in } \mathbb{Q}\}$. Thus $q \xrightarrow{\alpha} q'$ in $\mathbb{Q}'_{se}$ iff $q \overset{\alpha}{\Rightarrow}_{se} q'$ in $\mathbb{Q}$.

We will now take the time to establish the idempotence of the observational closure operator for use in later proofs. The result is expected since the observational closure operator performs a variation of transitive closure.

Lemma 3.18 *For any SELTS $\mathbb{Q} = \langle Q, \Sigma, R_\Sigma, q_0, P\rangle$ the observational closure operator is idempotent (ie. $(\mathbb{Q}'_{se})'_{se} = \mathbb{Q}'_{se}$).*

Proof: From Definition 3.17 we can see that $\mathbb{Q}'_{se}$ and $(\mathbb{Q}'_{se})'_{se}$ only differ in the definitions of their transition relations which we will denote by R'_Σ and R''_Σ respectively. Thus proving the idempotence of the observational closure operator reduces to showing that $R'_\Sigma = R''_\Sigma$.

We trivially have $R'_\Sigma \subseteq R''_\Sigma$ because when $q \xrightarrow{\alpha} q'$ in $\mathbb{Q}'_{se}$, by definition $q \overset{\alpha}{\Rightarrow}_{se} q'$ also. Thus by Definition 3.17, $q \xrightarrow{\alpha} q'$ in $(\mathbb{Q}'_{se})'_{se}$.

It remains to show $R'_\Sigma \supseteq R''_\Sigma$. For any $\alpha \in \Sigma$, if $q \xrightarrow{\alpha} q'$ in $(\mathbb{Q}'_{se})'_{se}$ then $q \overset{\alpha}{\Rightarrow}_{se} q'$ in $\mathbb{Q}'_{se}$. Thus there exist $q_1, \ldots, q_m, q'_1, \ldots, q'_n \in Q$ such that

$$q \xrightarrow{\tau} q_1 \xrightarrow{\tau} \ldots q_{m-1} \xrightarrow{\tau} q_m \xrightarrow{\alpha} q'_1 \xrightarrow{\tau} q'_2 \xrightarrow{\tau} \ldots q'_{n-1} \xrightarrow{\tau} q'_n \xrightarrow{\tau} q'$$

in $\mathbb{Q}'_{se}$. Also, for all $i = 1, \ldots, m$ we have $P(q_i) = P(q)$ and for all $j = 1, \ldots, n$ it is the case that $P(q'_j) = P(q')$. But $q_a \xrightarrow{\tau} q_b$ in $\mathbb{Q}'_{se}$ with $P(q_a) = P(q_b)$ iff $q_a \Rightarrow_{se} q_b$ in $\mathbb{Q}$, and $q_c \xrightarrow{\alpha} q_d$ in $\mathbb{Q}'_{se}$ iff $q_c \overset{\alpha}{\Rightarrow}_{se} q_d$ in $\mathbb{Q}$. Hence

$$q \Rightarrow_{se} q_1 \Rightarrow_{se} \ldots q_{m-1} \Rightarrow_{se} q_m \overset{\alpha}{\Rightarrow}_{se} q'_1 \Rightarrow_{se} q'_2 \Rightarrow_{se} \ldots q'_{n-1} \Rightarrow_{se} q'_n \Rightarrow_{se} q'$$

in $\mathbb{Q}$. But the $\Rightarrow_{se}$ relation is a transitive closure so

$$q \Rightarrow_{se} q_m \overset{\alpha}{\Rightarrow}_{se} q'_1 \Rightarrow_{se} q'$$

in $\mathbb{Q}$ and we conclude $q \overset{\alpha}{\Rightarrow}_{se} q'$ in $\mathbb{Q}$. Thus, by Definition 3.17, $q \xrightarrow{\alpha} q'$ in $\mathbb{Q}'_{se}$ and so we conclude that $R'_\Sigma \supseteq {R'_\Sigma}'$. $\square$

The observational closure operator can now be combined with the previous definition of strong state-event observers to define weak state-event observers.

Definition 3.19 *Given a SELTS* $\mathbb{Q} = \langle Q, \Sigma, R_\Sigma, q_0, P \rangle$*, the* weak state-event observer*,* $\theta_w(\mathbb{Q})$ *is defined to be*

$$\theta_w(\mathbb{Q}) = \sup\{\theta \in CP(\mathbb{Q}'_{se}) : \theta \leq \ker(P)\}$$

By Lemma 3.2 and the fact that $\Delta \in \{\theta \in CP(\mathbb{Q}'_{se}) : \theta \leq \ker(P)\}$, θ_w always exists and is unique. Note that in $\mathbb{Q}'_{se}$ the transition relations are dependent upon P so θ_w is not just Milner's observation equivalence with a different initial partition (as was the case for strong state-event observers). It is easy to see that in the case when $\ker(P) = \nabla$ then θ_w is in fact $\approx$, Milner's weak observation equivalence, since in that case $\overset{\alpha}{\Rightarrow}_{se}$ becomes $\overset{\alpha}{\Rightarrow}$ and $\overset{\tau}{\Rightarrow}_{se}$ becomes $\overset{\tau}{\Rightarrow}$ and hence $\mathbb{Q}'_{se} = \mathbb{Q}'$. As was the case for strong state-event equivalence, when $(q, q') \in \theta_w$ for a given $\mathbb{Q}$, we will write $q \approx_{se} q'$, read "q is weak state-event observation equivalent to q'". The $O(n^3)$ algorithm ($n = |Q|$) for computing Milner's weak observation equivalence of finite state LTS given in [BC89] can be easily adapted to provide an $O(n^3)$ algorithm for θ_w. The increase in complexity over the algorithm for strong state-event observers is the result of having to compute the transitive closure of τ within each equivalence class (cell) to obtain the relation $\Rightarrow_{se}$ used in constructing $\mathbb{Q}'_{se}$. Once we have $\mathbb{Q}'_{se}$, the $O(m \log n)$ RCP algorithm (m is the size of R_Σ – the number of related pairs – and $n = |Q|$) of [PT87] can be employed to compute θ_w giving an overall complexity of $O(n^3 + m \log n)$. If we assume that we are dealing with a fixed event set Σ, then each event can label at most n^2 transitions (each state is connected to all states by every event). Then $m \leq |\Sigma| n^2$ hence the complexity of computing θ_w is $O(n^3)$.

Similar to the case of the state observers of Section 2.3 and the strong state-event observers of Section 3.1, θ_w is the coarsest compatible partition of $\mathbb{Q}'_{se}$ that is finer than the equivalence kernel of P. Although the double arrow relations used to construct $\mathbb{Q}'_{se}$ may or may not cross a boundary of the partition of $\ker(P)$, the use of $\ker(P)$ as the initial partition detects when a change in state output occurs. Thus for $(q, q') \in \theta_w$ we have $P(q) = P(q')$ so q and q' produce the same current state output. Now suppose that $q \xrightarrow{\alpha} q_1$ in $\mathbb{Q}$, thereby producing event output α and state output $P(q_1)$. Then $q \overset{\alpha}{\Rightarrow}_{se} q_1$ in $\mathbb{Q}$ so $q \xrightarrow{\alpha} q_1$ in $\mathbb{Q}'_{se}$, and since $\theta_w \in CP(\mathbb{Q}'_{se})$ there exists $q'_1 \in \alpha^{\mathbb{Q}'_{se}}(q')$ such that $(q_1, q'_1) \in \theta_w$. Hence $q' \xrightarrow{\alpha} q'_1$ in $\mathbb{Q}'_{se}$ and $P(q_1) = P(q'_1)$. But then in $\mathbb{Q}$, $q' \overset{\alpha}{\Rightarrow}_{se} q'_1$. Thus q' can generate state and event outputs that are indistinguishable from those produced from q. As was the case with strong state-event observers, θ_w represents the minimum information one needs about the current state to be able to predict all possible future changes in state and future event outputs.

Since the weak state-event observer for a SELTS $\mathbb{Q}$ is just the strong state-event observer for $\mathbb{Q}'_{se}$, we can use the results of the previous section to derive similar results about what we will term *weak quotient systems*. In defining weak quotient systems we use the intuition that in the weakly observable setting the actions $q \xrightarrow{\alpha} q'$ and $q \overset{\alpha}{\Rightarrow}_{se} q'$ are indistinguishable.

Definition 3.20 *Given a SELTS $\mathbb{Q} := \langle Q, \Sigma, R_\Sigma, q_0, P \rangle$, for $\theta \in CP(\mathbb{Q}'_{se})$ such that $\theta \leq \ker(P)$ we define the* weak quotient system of $\mathbb{Q}$ by θ *to be $\mathbb{Q} /\!\!/ \theta := \mathbb{Q}'_{se}/\theta$.*

Again we can extend weak state-event observation equivalence to a relation $\approx_{se}$ between LTS by forming the union of disjoint SELTS (see Definition 3.5).

Definition 3.21 *Given two disjoint SELTS $\mathbb{Q}_1 = \langle Q_1, \Sigma, R^1_\Sigma, q_{10}, P_1 \rangle$ and $\mathbb{Q}_2 = \langle Q_2, \Sigma, R^2_\Sigma, q_{20}, P_2 \rangle$ where $P_1 : Q_1 \to R$ and $P_2 : Q_2 \to R$. We say that $\mathbb{Q}_1$ is* weakly state-event equivalent *to $\mathbb{Q}_2$, written $\mathbb{Q}_1 \approx_{se} \mathbb{Q}_2$, iff $(q_{10}, q_{20}) \in \theta_w(\mathbb{Q}_1 \cup \mathbb{Q}_2)$.*

With Milner's (event) observation equivalence, strong equivalence of LTS implies weak equivalence LTS [Mil89]. A similar situation holds for the state-event equivalence of SELTS.

Lemma 3.22 *$\mathbb{Q}_1 \sim_{se} \mathbb{Q}_2$ implies $\mathbb{Q}_1 \approx_{se} \mathbb{Q}_2$.*

The above lemma is an immediate result of the fact that when two transition systems with associated state output maps are strongly state-event observation equivalent, any sequence of moves made in one system can be matched by an identical event sequence in the other system producing the same state outputs. We can now prove the main result of this section.

Theorem 3.23 *For any reachable SELTS $\mathbb{Q}$, the weak quotient system $\mathbb{Q}/\!\!/\theta_w$ is a minimal state SELTS such that $\mathbb{Q} \approx_{se} \mathbb{Q}/\!\!/\theta_w$.*

Proof: Let $\theta = \theta_w(\mathbb{Q})$. By Theorem 3.13

$$\mathbb{Q}'_{se} \sim_{se} \mathbb{Q}'_{se}/\theta = \mathbb{Q}/\!\!/\theta_w$$

Then by Lemma 3.22 $Q'_{se} \approx_{se} \mathbb{Q}/\!\!/\theta_w$. It then follows from the definition of $\approx_{se}$ that $(Q'_{se})'_{se} \sim_{se} (\mathbb{Q}/\!\!/\theta_w)'_{se}$. But by Lemma 3.18 $(\mathbb{Q}'_{se})'_{se} = \mathbb{Q}'_{se}$. Thus $\mathbb{Q}'_{se} \sim_{se} (\mathbb{Q}/\!\!/\theta_w)'_{se}$ and therefore $\mathbb{Q} \approx_{se} \mathbb{Q}/\!\!/\theta_w$, as required.

The minimality of the state set of $\mathbb{Q}/\theta_w$ follows from Theorem 3.13. □

In general $\mathbb{Q}/\!\!/\theta_w$ is one of many possible minimal state SELTS that can be equivalent to $\mathbb{Q}$ but that differ in the definition of their transition relations. Uniqueness of a minimal state equivalent SELTS is lost in the weak state-event observation equivalence setting because of the use of the many-to-one ${}'_{se}$ observational closure operator in Definition 3.19. Consider Figure 3.7. In this case $\mathbb{Q} \neq \mathbb{Q}'_{se}$ but by Lemma 3.18

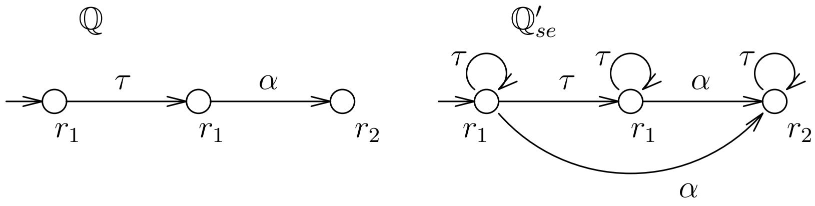


Figure 3.7: Example illustrating observational closure operator is many-to-one

$(\mathbb{Q}'_{se})'_{se} = \mathbb{Q}'_{se}$. Thus application of the observational closure operator to $\mathbb{Q}$ and $\mathbb{Q}'_{se}$ produces the same result.

In fact, as a result of the following lemma, we can conclude that $\mathbb{Q}/\!\!/\theta_w$ as defined has the maximum number of transitions.

Lemma 3.24 *Given an SELTS $\mathbb{Q}$ and $\theta \in \{CP(\mathbb{Q}'_{se}) : \theta \leq \ker(P)\}$ then $(\mathbb{Q}'_{se}/\theta)'_{se} = \mathbb{Q}'_{se}/\theta$.*

Proof: In the proof we will use the fact that θ defines a homomorphism, which we will also denote by θ, $\theta : \mathbb{Q}'_{se} \to \mathbb{Q}'_{se}$ that maps each state to its θ-cell (equivalence class).

As was the case for Lemma 3.18, demonstrating the equality of the transition systems reduces to showing that the transition relations of $\mathbb{Q}/\!\!/\theta_w$ contain those of $(\mathbb{Q}'_{se}/\theta)'_{se}$. That is, for any $x, x' \in Q/\theta$ and $\alpha \in \Sigma$, if $x \xrightarrow{\alpha} x'$ in $(\mathbb{Q}'_{se}/\theta)'_{se}$ then we must show that $x \xrightarrow{\alpha} x'$ in $\mathbb{Q}/\!\!/\theta_w$.

But if $x \xrightarrow{\alpha} x'$ in $(\mathbb{Q}'_{se}/\theta)'_{se}$ then by Definition 3.17 $x \overset{\alpha}{\Rightarrow}_{se} x'$ in $\mathbb{Q}/\!\!/\theta_w$. Thus there exists $x_1, \ldots, x_m, x'_1, \ldots, x'_n \in Q/\theta$ such that

$$x \xrightarrow{\tau} x_1 \xrightarrow{\tau} \ldots x_{m-1} \xrightarrow{\tau} x_m \xrightarrow{\alpha} x'_1 \xrightarrow{\tau} x'_2 \xrightarrow{\tau} \ldots x'_{n-1} \xrightarrow{\tau} x'_n \xrightarrow{\tau} x'$$

in $\mathbb{Q}'_{se}/\theta$ and for all $i = 1, \ldots, m$ we have $P(x_i) = P(x)$ and for all $j = 1, \ldots, n$ it is the case that $P(x'_j) = P(x')$. But θ is an epimorphism of SELTS, so $x_a \xrightarrow{\tau} x_b$ in $\mathbb{Q}'_{se}/\theta$ with $P(x_a) = P(x_b)$ iff there exist $q_a, q_b \in Q$ such that $\theta(q_a) = x_a$, $\theta(q_b) = x_b$ and $q_a \xrightarrow{\tau} q_b$ in $\mathbb{Q}'_{se}$. A similar situation holds for any α transition made in $\mathbb{Q}'_{se}/\theta$. Hence there exist $q, q_1, \ldots, q_m, q', q'_1, \ldots, q'_n \in Q$ such that $\theta(q) = x, \theta(q') = x'$, $\theta(q_i) = x_i$ where $i = 1, \ldots, m$, $\theta(q'_j) = x'_j$ for $j = 1, \ldots, n$ and

$$q \xrightarrow{\tau} q_1 \xrightarrow{\tau} \ldots q_{m-1} \xrightarrow{\tau} q_m \xrightarrow{\alpha} q'_1 \xrightarrow{\tau} q'_2 \xrightarrow{\tau} \ldots q'_{n-1} \xrightarrow{\tau} q'_n \xrightarrow{\tau} q'$$

in $\mathbb{Q}'_{se}$. Thus $q \overset{\alpha}{\Rightarrow}_{se} q'$ in $\mathbb{Q}'_{se}$, so by the definition of observational closure $rq \xrightarrow{\alpha} q'$ in $(\mathbb{Q}'_{se})'_{se}$. But $(\mathbb{Q}'_{se})'_{se} = \mathbb{Q}'_{se}$ by Lemma 3.18 so $q \xrightarrow{\alpha} q'$ in $\mathbb{Q}'_{se}$. Since $\theta(q) = x$, $\theta(q') = x'$ and θ is a homomorphism we can then conclude that $x \xrightarrow{\alpha} x'$ in $\mathbb{Q}'_{se}/\theta$. □

Corollary 3.25 *For any SELTS $\mathbb{Q}$*

$$(\mathbb{Q}/\!\!/\theta_w)'_{se} = (\mathbb{Q}'_{se}/\theta_w)'_{se} = \mathbb{Q}'_{se}/\theta_w = \mathbb{Q}/\!\!/\theta_w$$

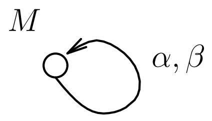
Thus the weak quotient system already has an instance of any transition that observational closure would add. The choice of transition relations used in Definition 3.20 was made for its algebraic and computational simplicity. In particular, all the self-looped τ transitions that result from the observational closure operator allow for the straightforward application of strong state-event equivalence in the definition of weak state-event equivalence. An obvious reduction that one can make in the number of transition of $\mathbb{Q}/\!\!/\theta_w$ and still maintain weak equivalence is to eliminate all self-looped transitions since these are always added to every state by the observational closure operator. One might then ask if it is possible find a system with a minimal number of transitions that is still weakly state-event equivalent to $\mathbb{Q}/\!\!/\theta_w$.

To obtain a system that is weak state-event equivalent to the weak quotient system and has a minimum number of transitions would involve the solution of multiple instances of the *minimal equivalent digraph problem* [AHU83]. A directed graph (or digraph) can be represented as $G := (V, E)$ where V is a set of vertices (states) and E is a set of edges (a transition relation). A graph $G' := (V, E')$ is said to be a *minimal equivalent digraph for* $G := (V, E)$ if E' is a minimal subset of E such that the transitive closure of both G and G' are the same. In [Sah74] the author shows that the problem of finding a minimal equivalent digraph in the general case is NP-complete. Thus there is little hope of finding an efficient algorithm to solve the problem.

In obtaining a system with a minimal number of transitions that is weak state-event equivalent to $\mathbb{Q}/\!\!/\theta_w$, one would have to solve the minimal equivalent digraph problem within each cell of the state output map for the graph with edges representing the silent τ transition relation (ie. we have to find the minimum number of τ transitions that would still generate the same $\Rightarrow_{se}$ relation). While generating a minimal transition equivalent system could certainly speed up some model checking and supervisory control computations, it appears that the effort required to obtain any such minimal transition system would outweigh any gain in performance as the result of such transition minimization.

3.3 Example: Weak State-Event Observer of a Simple Real-Time System

In this section we present a small example. The weak state-event observer theory will be applied to the Timed Transition Model (TTM) M of Figure 3.8.



$\alpha := (true, [z : z \oplus_2 1], 0, 1)$
$\beta := (z = 0, [y : y \oplus_3 1], 1, \infty)$
$\Theta := (y = z = 0)$

Figure 3.8: Example TTM M

A TTM is a guarded transition system with lower and upper time bounds on the transitions that relate to the number of occurrences of the special transition *tick*. For M there are three transitions, α, β and *tick*, and two program variables, y and z. The initial condition Θ specifies that M starts with both y and z set to 0. Now consider the transition $\alpha := (true, [z : z \oplus_2 1], 0, 1)$. The guard or "enablement condition" of α is *true*, hence the transition is always enabled. When the transition α occurs, it has the effect specified by its operation function: in this case z becomes $z \oplus_2 1$ (here $\oplus_n$ denotes addition mod n). The lower and upper time bounds for α are 0 and 1 respectively. For α to occur, its guard condition must evaluate to *true* continuously for at least 0 *tick* transitions and if its guard remains *true* after one *tick*, it will be forced to occur before the next *tick* event. Since α's guard transition always evaluates to *true*, the above time bounds force at least one, to a finite but unbounded number of α's to occur between successive *tick*s of the "clock". In the case of $\beta := (z = 0, [y : y \oplus_3 1], 1, \infty)$, the value of z must be 0 for at least one *tick* before β can occur. The upper time bound of ∞ indicates that even if β is continuously enabled for arbitrarily many occurrences of *tick*, it is never forced to occur. If β does occur then y changes to $y \oplus_3 1$.

The SELTS representing the "trajectories" of M is shown in Figure 3.9. The process of obtaining the SELTS representing the legal trajectories of a TTM is covered briefly in Section 2.2.3 The interested reader is referred to [Ost89] for complete details of the semantics of TTMs used to obtain the SELTS. Beside each state of the SELTS

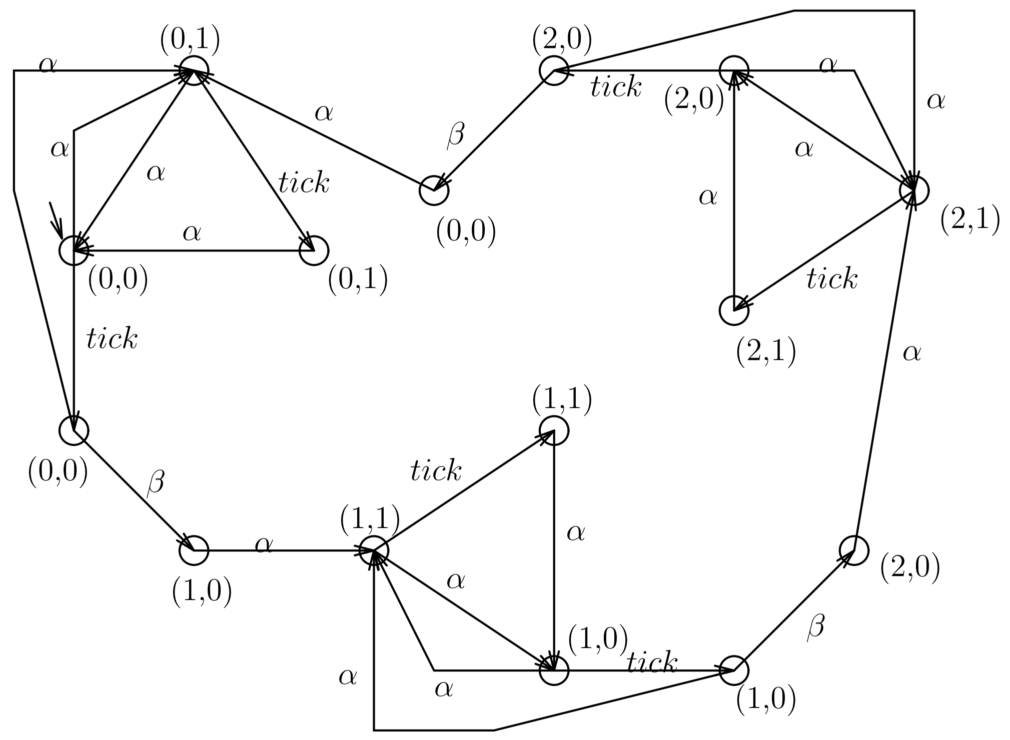


Figure 3.9: SELTS generated by TTM M

in Figure 3.9, we write the ordered pair (y, z) to give the current value of the program variables y and z. The initial state of the SELTS (q_0) is the state with the entering arrow.

Suppose we are interested in the timed behavior of M under the state output map,

$$P(q) := \begin{cases} a, \ y = 2 \\ b, \ \text{otherwise} \end{cases}$$

The SELTS resulting from these state and event observations is shown in Figure 3.10. The dashed line indicates the state partition induced by P. States to the left of the dashed line (states 1-9 and 15) result in a state output of b while those to the right (states 10-14) produce a state output of a. In this case the event $tick$ remains observable while α and β are replaced in the SELTS with unobservable τ transitions since it is only their effect on the state output that is of interest. Once the relations $\overset{\tau}{\Rightarrow}_{se}$

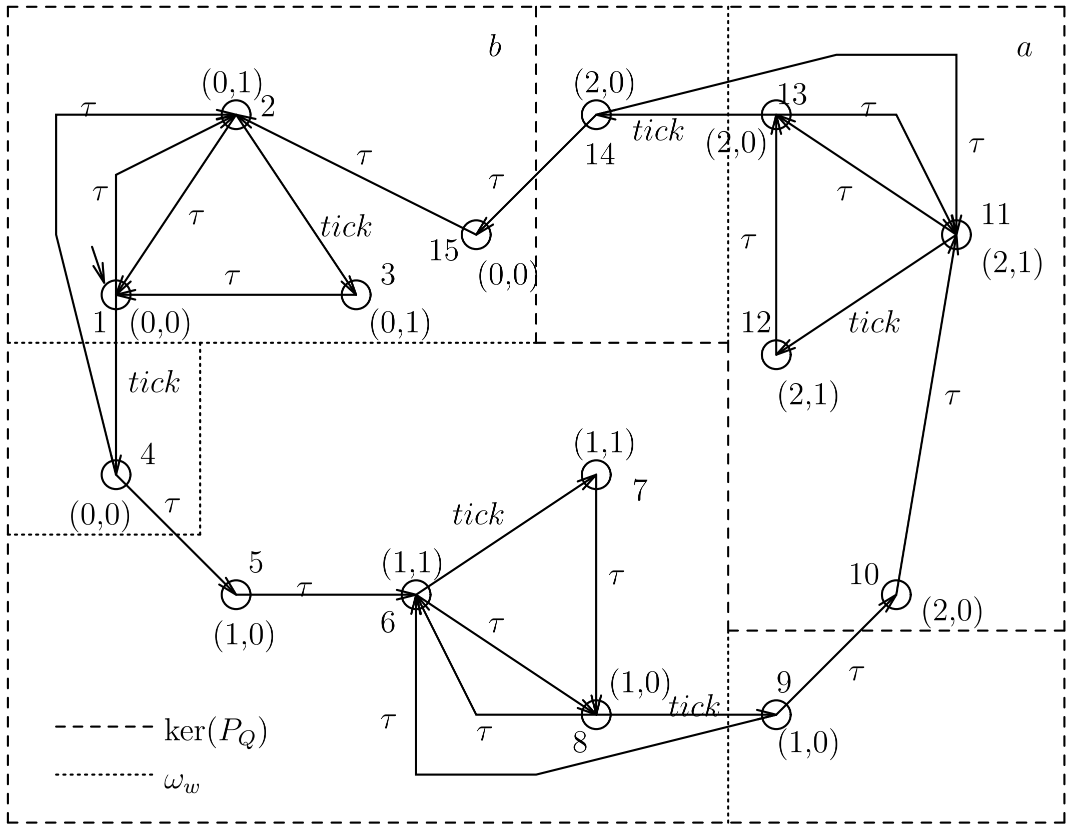


Figure 3.10: ker(P) and resulting θ_w for SELTS generated by TTM M

and $\overset{tick}{\Rightarrow}_{se}$ are determined, we can compute the weak state observer θ_w, the refinement of ker(P) shown as dotted lines in Figure 3.10.

To understand how θ_w is obtained from ker(P), consider the individual states of the SELTS. States 9 and 14 are the only two states that are the sources of sequences of unobservable τ transitions that change the state output (eg. $9\overset{\tau}{\Rightarrow}_{se}10$ and $P(10) = a \neq P(9)$). All other states must first execute at least one observable *tick* transition before causing a change in the state output. Hence 9 and 14 are sectioned off from their respective cells of ker(P). When the relation $\overset{tick}{\Rightarrow}_{se}$ is considered, further refinements of ker(P) result. State 4 can reach state 9 via silent τ transitions within a cell of ker(P) and a *tick* (eg. $4\overset{tick}{\Rightarrow}_{se}9$) while also being able to access states $1, 2, 3$ and 15, states that cannot reach state 9 via the $\overset{tick}{\Rightarrow}_{se}$ relation. As a result 4 is split off from the other states of ker(P). The rest of the refinement of ker(P) proceeds in a similar fashion. It is left to the reader to verify that the partition θ_w shown in Figure 3.10 is indeed a compatible partition for the relations $\overset{\tau}{\Rightarrow}_{se}$ and $\overset{tick}{\Rightarrow}_{se}$ as defined in the previous section.

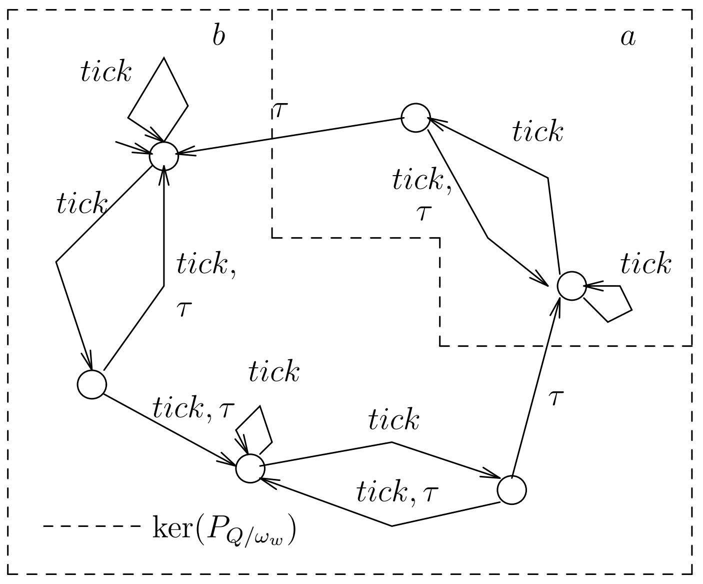


Figure 3.11: Weak Quotient system generated by θ_w

Figure 3.11 presents the weak quotient system with respect to the weak state-event observer θ_w of the SELTS of Figure 3.10. Note that in the weak quotient system there is a single state for each cell of θ_w. The cell containing state 1, the initial state of the original SELTS, becomes the initial state of the quotient system. Also, the map P_{Q/θ_w} is given by the displayed kernel partition. To simplify the graph for display purposes we have omitted the self-looped τ transitions that occur for definitional reasons at each state of the weak quotient system.

3.4 Compositional Consistency

The main goal of this section is to demonstrate that replacing a component system of a synchronous product with a state-event equivalent system results in a composite system that is state-event equivalent to the original synchronous product. The result is that when the properties of interest are preserved by state-event equivalence, one can use the synchronous product of quotient systems in place of the synchronous product of the original systems. This replacement has the potential to result in dramatic state reductions of the models used (eg. Section 5.3 and [Ost95]). The size of synchronous products typically grows as the product of the sizes of the state sets of the component systems: thus any state reductions performed before synchronous composition have a multiplicative effect.

The equivalence of a system with its quotient system together with the following results regarding the synchronous composition of equivalent systems provides us with the means for performing weak, compositionally consistent, model reduction for the temporal logic model checking of Chapter 4.

Our proofs of the results of this section are greatly simplified by the use of Theorem 3.13. Instead of arguing at the element level to show that a particular partition is an output compatible partition, we are able to argue at the arrow level, demonstrating that certain maps are SELTS homomorphisms.

3.4.1 Strong Compositional Consistency

As was the case with state-event observers, we begin by considering the strong state-event case and then extend the results to the weak state-event setting. The first lemma demonstrates how a homomorphism of a system can be used to construct a homomorphism of the new system created by the synchronous composition of the system with another SELTS. This lemma is followed by a simple corollary which together with Theorem 3.13 provides the means for proving the main result regarding synchronous composition of strongly state-event equivalent systems.

Lemma 3.26 *Given three SELTS $\mathbb{Q}$ and $\mathbb{Q}_i, i = 1, 2$, if $h : \mathbb{Q}_1 \to \mathbb{Q}_2$ is a SELTS homomorphism, then for all $\Sigma_s \subseteq \Sigma$*

$$h \times id_Q : \mathbb{Q}_1|[\Sigma_s]|\mathbb{Q} \to \mathbb{Q}_2|[\Sigma_s]|\mathbb{Q}$$

(ie. $h \times id_Q : Q_1 \times Q \to Q_2 \times Q$ is a SELTS homomorphism)

Proof: Since h and id_Q are homomorphisms $h \times id_Q((q_{10}, q_0)) = (q_{20}, q_0)$ and, with the help of Claim 2.5, $(P_2 \times P) \circ (h \times id_Q) = (P_2 \circ h) \times (P \circ id_Q) = P_1 \times P$. All that remains is to show that for all $\alpha \in \Sigma$,

$$(h \times id_Q)_* \circ \alpha^{\mathbb{Q}_1|[\Sigma_s]|\mathbb{Q}} = \alpha^{\mathbb{Q}_2|[\Sigma_s]|\mathbb{Q}} \circ (h \times id_Q)$$

For $\alpha^{\mathbb{Q}_1|[\Sigma_s]|\mathbb{Q}}$ there are two cases we have to consider.

Case 1: $\alpha \in \Sigma_s$. Then $\alpha^{\mathbb{Q}_1|[\Sigma_s]|\mathbb{Q}} = \alpha^{\mathbb{Q}_1} \otimes \alpha^{\mathbb{Q}}$ so

$$\begin{aligned}
(h \times id_Q)_* \circ \alpha^{\mathbb{Q}_1|[\Sigma_s]|\mathbb{Q}} &= (h \times id_Q)_* \circ (\alpha^{\mathbb{Q}_1} \otimes \alpha^{\mathbb{Q}}) \\
&= (h_* \circ \alpha^{\mathbb{Q}_1}) \otimes ((id_Q)_* \circ \alpha^{\mathbb{Q}}) \\
&\quad \text{by Claim 2.6 (i)} \\
&= (\alpha^{\mathbb{Q}_2} \circ h) \otimes (\alpha^{\mathbb{Q}} \circ id_Q) \\
&\quad \text{since } h \text{ and } id_Q \text{ are homomorphisms} \\
&= (\alpha^{\mathbb{Q}_2} \otimes \alpha^{\mathbb{Q}}) \circ (h \times id_Q) \\
&\quad \text{by Claim 2.6 (ii)} \\
&= \alpha^{\mathbb{Q}_2|[\Sigma_s]|\mathbb{Q}} \circ (h \times id_Q)
\end{aligned}$$

as required.

Case 2: $\alpha \notin \Sigma_s$. Then $\alpha^{\mathbb{Q}_1|[\Sigma_s]|\mathbb{Q}} = (\alpha^{\mathbb{Q}_1} \otimes id_Q) \cup (id_{Q_1} \otimes \alpha^{\mathbb{Q}})$ so

$$\begin{aligned}
(h \times id_Q)_* \circ \alpha^{\mathbb{Q}_1|[\Sigma_s]|\mathbb{Q}} &= (h \times id_Q)_* \circ (\alpha^{\mathbb{Q}_1} \otimes id_Q \cup id_{Q_1} \otimes \alpha^{\mathbb{Q}}) \\
&= (h \times id_Q)_* \circ (\alpha^{\mathbb{Q}_1} \otimes id_Q) \cup (h \times id_Q)_* \circ (id_{Q_1} \otimes \alpha^{\mathbb{Q}}) \\
&\quad \text{by Claim 2.4 (i).} \\
\text{But,} & \\
(h \times id_Q)_* \circ (\alpha^{\mathbb{Q}_1} \otimes id_Q) &= (h_* \circ \alpha^{\mathbb{Q}_1}) \otimes ((id_Q)_* \circ id_Q) \\
&\quad \text{by Claim 2.6 (i)} \\
&= (\alpha^{\mathbb{Q}_2} \circ h) \otimes (id_Q \circ id_Q) \\
&\quad \text{since } h \text{ and } id_Q \text{ are homomorphisms} \\
&= (\alpha^{\mathbb{Q}_2} \otimes id_Q) \circ (h \otimes id_Q) \\
&\quad \text{by Claim 2.6 (ii).} \\
\text{Similarly,} & \\
(h \times id_Q)_* \circ (id_{Q_1} \otimes \alpha^{\mathbb{Q}}) &= (id_{Q_2} \otimes \alpha^{\mathbb{Q}}) \circ (h \times id_Q) \\
\text{So,} &
\end{aligned}$$

$$\begin{aligned}(h \times id_Q)_* \circ \alpha^{\mathbb{Q}_1|[\Sigma_s]|\mathbb{Q}} &= (\alpha^{\mathbb{Q}_2} \otimes id_Q) \circ (h \times id_Q) \cup (id_{Q_2} \otimes \alpha^{\mathbb{Q}}) \circ (h \times id_Q) \\ &= [(\alpha^{\mathbb{Q}_2} \otimes id_Q) \cup (id_{Q_2} \otimes \alpha^{\mathbb{Q}})] \circ (h \times id_Q) \\ &\quad \text{by Claim 2.4 (ii)} \\ &= \alpha^{\mathbb{Q}_2|[\Sigma_s]|\mathbb{Q}} \circ (h \times id_Q)\end{aligned}$$

as required. □

Given that a homomorphism h can be used to construct the new homomorphism $h \times id_Q$ of the composed systems, it seems logical to ask if for homomorphisms h_L, h_R, their product $h_L \times h_R$ is a homomorphism of the synchronous product of their associated systems. This question is formalized and answered in the affirmative by the following corollary.

Corollary 3.27 *Given SELTS homomorphisms $h_L : \mathbb{Q}_{L1} \rightarrow \mathbb{Q}_{L2}$ and $h_R : \mathbb{Q}_{R1} \rightarrow \mathbb{Q}_{R2}$, then for any $\Sigma_s \subseteq \Sigma$:*

$$h_L \times h_R : \mathbb{Q}_{L1}|[\Sigma_s]|\mathbb{Q}_{R1} \rightarrow \mathbb{Q}_{L2}|[\Sigma_s]|\mathbb{Q}_{R2}$$

Proof: Since

$$h_L \times h_R = (h_L \times id_{Q_{R2}}) \circ (id_{Q_{L1}} \times h_R)$$

the result immediately follows from Lemma 3.26 together with the fact that the composition of SELTS homomorphisms is a homomorphism by Lemma 3.7. □

$$\begin{tikzcd}
Q_{L2} \times Q_{R2} \arrow[r, "\alpha^{\mathbb{Q}_{L2}|[\Sigma_s]|\mathbb{Q}_{R2}}"] & \mathcal{P}(Q_{L2} \times Q_{R2}) \\
Q_{L1} \times Q_{R1} \arrow[u, "h_L \times h_R"] \arrow[r, "\alpha^{\mathbb{Q}_{L1}|[\Sigma_s]|\mathbb{Q}_{R1}}"] & \mathcal{P}(Q_{L1} \times Q_{R1}) \arrow[u, "(h_L \times h_R)_*"']
\end{tikzcd}$$

Figure 3.12: Commutative diagram for Corollary 3.27

Corollary 3.27 states that the diagram in Figure 3.12 commutes.

For the remainder of this section we assume that we are given SELTS

$$\mathbb{Q}_{Li} = \langle Q_{Li}, \Sigma_i, R_\Sigma^{Li}, q_{Li0}, P_{Li}\rangle \text{ with } P_{Li} : Q_{Li} \to R_L, \text{ for } i = 1, 2$$
$$\mathbb{Q}_{Ri} = \langle Q_{Ri}, \Sigma_i, R_\Sigma^{Ri}, q_{Ri0}, P_{Ri}\rangle \text{ with } P_{Li} : Q_{Ri} \to R_R, \text{ for } i = 1, 2$$

Now assume that $I := (\Sigma_s, f_L, f_R)$ with state output synchronization maps $f_L : R_L \to R$ and $f_R : R_R \to R$ is a compatible interface for the $\mathbb{Q}_{L1}$ and $\mathbb{Q}_{R1}$. In order to extend this result of Corollary 3.27 on the compositional consistency of event composition to compositional consistency for state-event composition, we combined the above commutative diagram with the commutative diagram in Figure 2.8 relating $|[\Sigma_s]|$ to $|[I]|$ (see Section 2.2.3). We then obtain the commutative diagram shown in Figure 3.13. Here π_{eq1}, π_{eq2} are the "projections" resulting from intersection with their respective equalizer sets from the arrow theoretic definition of $|[I]|$. That is, for $i = 1, 2$

$$\pi_{eqi} : \mathcal{P}(Q_{Li} \times Q_{Ri}) \to \mathcal{P}(Q_{Li} \times Q_{Ri})$$
$$A \in \mathcal{P}(Q_{Li} \times Q_{Ri}), A \mapsto A \cap eq(f_L \circ P_{Li} \circ \pi_1, f_R \circ P_{Ri} \circ \pi_2)$$

where, $eq(f_L \circ P_{Li} \circ \pi_1, f_L \circ P_{Ri} \circ \pi_2)$

$$\begin{aligned} &= \{(q_{Li}, q_{Ri}) \in Q_{Li} \times Q_{Ri} : f_L \circ P_{Li} \circ \pi_1(q_{Li}, q_{Ri}) = f_R \circ P_{Ri} \circ \pi_2(q_{Li}, q_{Ri})\} \\ &= \{(q_{Li}, q_{Ri}) \in Q_{Li} \times Q_{Ri} : f_L \circ P_{Li}(q_{Li}) = f_R \circ P_{Ri}(q_{Ri})\} \end{aligned}$$

Given that the diagrams in Figures 2.8 and 3.12 commute, in order to prove that the diagram in Figure 3.13 commutes, we need only prove that the subdiagram in Figure 3.14 commutes. This is the subject of the following lemma.

Lemma 3.28 *The diagram in Figure 3.14 commutes, ie.*

$$(h_L \times h_R)_* \circ \pi_{eq1} = \pi_{eq2} \circ (h_L \times h_R)_*$$

$$\begin{tikzcd}
Q_{L2} \times Q_{R2} \arrow[r, "\alpha^{\mathbb{Q}_{L2}|[\Sigma_s]|\mathbb{Q}_{R2}}"] \arrow[rr, bend left, "\alpha^{\mathbb{Q}_{L2}|[I]|\mathbb{Q}_{R2}}"] & \mathcal{P}(Q_{L2} \times Q_{R2}) \arrow[r, "\pi_{eq2}"] & \mathcal{P}(Q_{L2} \times Q_{R2}) \\
Q_{L1} \times Q_{R1} \arrow[u, "h_L \times h_R"] \arrow[r, "\alpha^{\mathbb{Q}_{L1}|[\Sigma_s]|\mathbb{Q}_{R1}}"] \arrow[rr, bend right, "\alpha^{\mathbb{Q}_{L1}|[I]|\mathbb{Q}_{R1}}"'] & \mathcal{P}(Q_{L1} \times Q_{R1}) \arrow[u, "(h_L \times h_R)_*"] \arrow[r, "\pi_{eq1}"] & \mathcal{P}(Q_{L1} \times Q_{R1}) \arrow[u, "(h_L \times h_R)_*"']
\end{tikzcd}$$

Figure 3.13: Commutative diagram for Corollary 3.27

$$\begin{tikzcd}
\mathcal{P}(Q_{L2} \times Q_{R2}) \arrow[r, "\pi_{eq2}"] & \mathcal{P}(Q_{L2} \times Q_{R2}) \\
\mathcal{P}(Q_{L1} \times Q_{R1}) \arrow[u, "(h_L \times h_R)_*"] \arrow[r, "\pi_{eq1}"] & \mathcal{P}(Q_{L1} \times Q_{R1}) \arrow[u, "(h_L \times h_R)_*"']
\end{tikzcd}$$

Figure 3.14: Commutative diagram for Lemma 3.28

Proof: Let $A \subseteq Q_{L1} \times Q_{R1}$ and $(q_{L2}, q_{R2}) \in (h_L \times h_R)_* \circ \pi_{eq1}$. Then there exists $(q_{L1}, q_{R1}) \in \pi_{eq1}(A)$ such that

$$h_L \times h_R(q_{L1}, q_{R1}) = (h_L(q_{L1}, h_R(q_{R1})) = (q_{L2}, q_{R2}) \tag{3.2}$$

Since $(q_{L1}, q_{R1}) \in \pi_{eq1}(A)$, by the definition of π_{eq1} we know

$$f_L \circ P_{L1}(q_{L1}) = f_R \circ P_{R1}(q_{R1}) \tag{3.3}$$

Thus using the fact that h_L, h_R are homomorphisms so that $P_{L2} \circ h_L = P_{L1}$ and $P_{R2} \circ h_R = P_{R1}$ we have

$$\begin{aligned} f_L \circ P_{L2}(q_{L2}) &= f_L \circ P_{L2} \circ h_L(q_{L1}) \\ &= f_L \circ P_{L1}(q_{L1}) \\ &= f_R \circ P_{R1}(q_{R1}), \text{ by (3.3)} \\ &= f_R \circ P_{R2} \circ h_R(q_{R1}) \\ &= f_R \circ P_{R2}(q_{R2}) \end{aligned}$$

By the definition of π_{eq2}, if $(q_{L2}, q_{R2}) \in B$ for a given set $B \subseteq Q_{L2} \times Q_{R2}$, then $(q_{L2}, q_{R2}) \in \pi_{eq2}(B)$. But by (3.2), $(q_{L2}, q_{R2}) \in (h_L \times h_R)_*(A)$. Thus $(q_{L2}, q_{R2}) \in \pi_{eq2} \circ (h_L \times h_R)_*(A)$ Thus

$$(h_L \times h_R)_* \circ \pi_{eq1}(A) \subseteq \pi_{eq2} \circ (h_L \times h_R)_*(A)$$

Now assume $(q_{L2}, q_{R2}) \in \pi_{eq2} \circ (h_L \times h_R)_*(A)$. Then by definition of π_{eq2},

$$f_L \circ P_{L2}(q_{L2}) = f_R \circ P_{R2}(q_{R2}) \tag{3.4}$$

and $(q_{L2}, q_{R2}) \in (h_L \times h_R)_*(A)$ since $\pi_{eq2}(B) \subseteq B$. Thus there exists $(q_{L1}, q_{R1}) \in A$ such that (3.2) holds.

Again using the fact that h_L, h_R are homomorphisms, we have

$$\begin{aligned} f_L \circ P_{L1}(q_{L1}) &= f_L \circ P_{L2} \circ h_L(q_{L1}) \\ &= f_L \circ P_{L2}(q_{L2}) \\ &= f_R \circ P_{R2}(q_{R2}), \text{ by (3.4)} \\ &= f_R \circ P_{R2} \circ h_R(q_{R1}) \\ &= f_R \circ P_{R1}(q_{R1}) \end{aligned}$$

Thus $(q_{L1}, q_{R1}) \in \pi_{eq1}(A)$ by the definition of pi_{eq1} and hence $(q_{L2}, q_{R2}) \in (h_L \times h_R)_* \circ \pi_{eq1}(A)$. We conclude that

$$(h_L \times h_R)_* \circ \pi_{eq1}(A) \supseteq \pi_{eq2} \circ (h_L \times h_R)_*(A)$$

thereby proving the desired result. □

We are now ready to prove the first of the two main results contained in this section. Theorem 3.29 is the strong version of the result which states that equivalent subsystems may be substituted for the original subsystems in a synchronous product and the resulting new composite system will be equivalent to the composition of the original systems.

Theorem 3.29 *Given SELTS $\mathbb{Q}_{Li}, \mathbb{Q}_{Ri}, i = 1, 2$. If $\mathbb{Q}_{L1} \sim_{se} \mathbb{Q}_{L2}$ and $\mathbb{Q}_{R1} \sim_{se} \mathbb{Q}_{R2}$, then for any compatible interface $I := (\Sigma_s, f_L, f_R)$ as defined above such*

$$(\mathbb{Q}_{L1}|[I]|\mathbb{Q}_{R1}) \sim_{se} (\mathbb{Q}_{L2}|[I]|\mathbb{Q}_{R2})$$

Proof: By Theorem 3.13, there exist SELTS $\mathbb{Q}_L$ and $\mathbb{Q}_R$ together with homomorphisms $h_{Li} : \mathbb{Q}_{Li} \to \mathbb{Q}_L$ and $h_{Ri} : \mathbb{Q}_{Ri} \to \mathbb{Q}_R$, $i = 1, 2$. Hence

$$\mathbb{Q}_{L1}|[I]|\mathbb{Q}_{R1} \overset{h_{L1} \times h_{R1}}{\longrightarrow} \mathbb{Q}_L|[I]|\mathbb{Q}_R \overset{h_{L2} \times h_{R2}}{\longleftarrow} \mathbb{Q}_{L2}|[I]|\mathbb{Q}_{R2}$$

by Corollary 3.28. This then allows us to apply the opposite direction of Theorem 3.13 to obtain the desired result. □

Theorem 3.29 also includes the pure event synchronous composition operator $|[\Sigma_s]|$ since this is a special case of $|[I]|$ when f_L and f_R are trivial constant state output synchronization maps.

3.4.2 Weak Compositional Consistency

By definition, weak state-event equivalence is just strong state-event equivalence of the observational closures of the original systems. Thus we begin this section by investigating the relationship between the observational closure operator $'_{se}$ and the state-event synchronous composition operator $|[I]|$. The event synchronous composition operator can be treated as a special case of state-event composition. What we discover will allow us to reduce the weak compositional consistency problem to a case in which the results for strong compositional consistency can be applied.

If the observational closure operator distributed over the synchronous composition operator, then we could use the fact that observational closure is idempotent, together with Theorem 3.13 to obtain a weak version of the previous theorem. Unfortunately observation closure does not distribute over synchronous composition.

Consider Figure 3.15. Here we use the event synchronous composition operator with an empty synchronization set $\Sigma_s = \emptyset$. To avoid cluttering the illustration, in the lower two SELTS we do not show the self-looped τ transitions that are present at every state by the definition of $'_{se}$ operator. One can easily see that

$$(\mathbb{Q}_1|[\emptyset]|\mathbb{Q}_2)'_{se} \neq (\mathbb{Q}_1)'_{se}|[\emptyset]|(\mathbb{Q}_2)'_{se}$$

In fact, $(\mathbb{Q}_1|[\emptyset]|\mathbb{Q}_2)'_{se} \not\sim_{se} (\mathbb{Q}_1)'_{se}|[\emptyset]|(\mathbb{Q}_2)'_{se}$ either! This outcome results from the inability of the distributed observational closure to interleave silent τ transitions from one subsystem with transitions from the other subsystem. In Figure 3.15 this is reflected by the lack of an α transition from $(1,1)$ to $(2,2)$ in $(\mathbb{Q}_1)'_{se}|[\emptyset]|(\mathbb{Q}_2)'_{se}$. A quick inspection of the two composite systems shows that they are weakly state-event

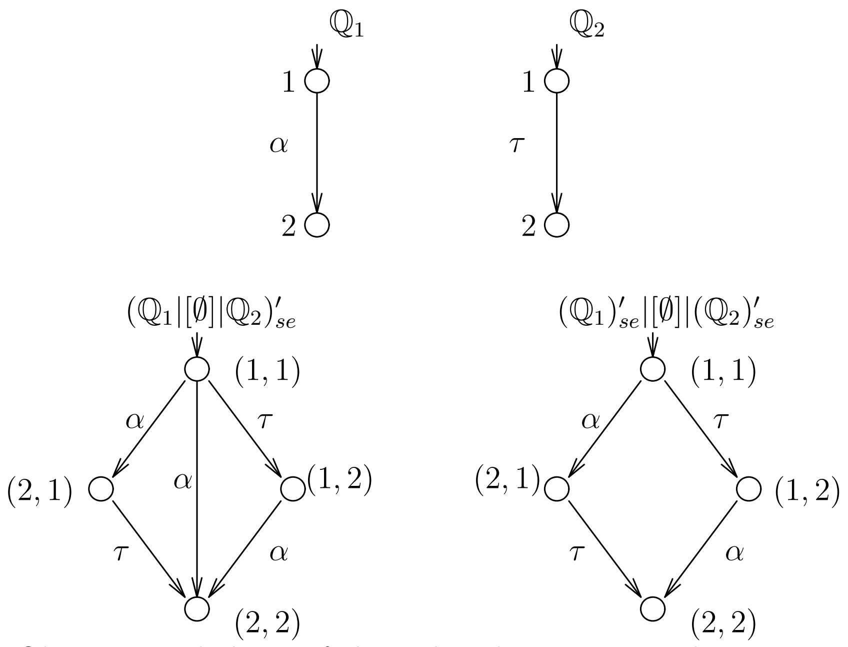


Figure 3.15: Observational closure fails to distribute over synchronous product.

equivalent since $(\mathbb{Q}_1|[\emptyset]|\mathbb{Q}_2)'_{se} = ((\mathbb{Q}_1)'_{se}|[\emptyset]|(\mathbb{Q}_2)'_{se})'_{se}$. In the lemma below we show that this property holds in general for state-event composition.

Lemma 3.30 *Let $\mathbb{Q}_L, \mathbb{Q}_R$ be SELTS with compatible interface $I := (\Sigma_s, f_L, f_R)$ as defined in Lemma 3.28. If $\Sigma_s \subseteq \Sigma_o$ (ie. $\tau \notin \Sigma_s$) then*

$$(\mathbb{Q}_L|[I]|\mathbb{Q}_R)'_{se} = ((\mathbb{Q}_L)'_{se}|[I]|(\mathbb{Q}_R)'_{se})'_{se}$$

Proof: Let $\mathbb{Q}_A := \mathbb{Q}_L|[\Sigma_s]|\mathbb{Q}_R$ and $\mathbb{Q}_B := (\mathbb{Q}_L)'_{se}|[\Sigma_s]|(\mathbb{Q}_R)'_{se}$. $\mathbb{Q}_A$ and $\mathbb{Q}_B$ differ only in their sets of transition relations which we denote by R_A and R_B. Clearly $R_A \subseteq R_B$ since the $'_{se}$ operator has the effect of adding transitions to a SELTS, hence the transition relations of the subsystems of $\mathbb{Q}_A$ are contained in the transition relations of the corresponding subsystems of $\mathbb{Q}_B$. If we denote the set of transition relations of ${\mathbb{Q}_A}'_{se}$ and ${\mathbb{Q}_B}'_{se}$ by R'_A and R'_B, then $R_A \subseteq R_B$ implies $R'_A \subseteq R'_B$. Thus to prove ${\mathbb{Q}_A}'_{se} = {\mathbb{Q}_B}'_{se}$, we need only show that $R'_A \supseteq R'_B$. For this containment, we will separate the cases when an event $\alpha \in \Sigma_s$ and $\alpha \notin \Sigma_s$.

Case 1: $\alpha \in \Sigma_s$. Assume that $(q_{L1}, q_{R1})\xrightarrow{\alpha}(q'_{L1}, q'_{R1})$ in ${\mathbb{Q}_B}'_{se}$. Therefore in $\mathbb{Q}_B$,

$(q_{L1}, q_{R1})\overset{\alpha}{\Rightarrow}_{se}(q'_{L1}, q'_{R1})$. That is, there exist (q_{L2}, q_{R2}) and (q'_{L2}, q'_{R2}) such that in $\mathbb{Q}_B$

$$(q_{L1}, q_{R1}) \Rightarrow_{se} (q_{L2}, q_{R2})\xrightarrow{\alpha}(q'_{L2}, q'_{R2}) \Rightarrow_{se} (q'_{L1}, q'_{R1})$$

But the $\Rightarrow_{se}$ relation is made up entirely of unobservable τ transitions that do not change the system's state output and hence are unaffected by the state output synchronization maps. Also, by assumption, $\tau \notin \Sigma_s$, so we can conclude that $q_{x1} \Rightarrow_{se} q_{x2}$ and $q'_{x2} \Rightarrow_{se} q'_{x1}$ in $\mathbb{Q}_x{}'_{se}$ for $x = L, R$. By the idempotence of observational closure we have that for any $\mathbb{Q}$, $q \Rightarrow_{se} q'$ in $\mathbb{Q}$ iff $q \Rightarrow_{se} q'$ in $\mathbb{Q}'_{se}$. Therefore from the above we can conclude that $q_{x1} \Rightarrow_{se} q_{x2}$ and $q'_{x2} \Rightarrow_{se} q'_{x1}$ in $\mathbb{Q}_x$ for $x = L, R$.

Again, since $\Rightarrow_{se}$ consists only of τ transitions that are unaffected by the state output synchronization maps and $\tau \notin \Sigma_s$, we now know that $(q_{L1}, q_{R1}) \Rightarrow_{se} (q_{L2}, q_{R2})$ and $(q'_{L2}, q'_{R2}) \Rightarrow_{se} (q'_{L1}, q'_{R1})$ in $\mathbb{Q}_A$. Thus in order to show that $(q_{L1}, q_{R1})\xrightarrow{\alpha}(q'_{L1}, q'_{R1})$ in $\mathbb{Q}_A{}'_{se}$, all that remains is to show that

$$\begin{aligned}(q_{L2}, q_{R2})\xrightarrow{\alpha}(q'_{L2}, q'_{R2}) \quad &\text{in} \quad \mathbb{Q}_B \text{ implies}\\ (q_{L2}, q_{R2})\overset{\alpha}{\Rightarrow}_{se}(q'_{L2}, q'_{R2}) \quad &\text{in} \quad \mathbb{Q}_A\end{aligned}$$

This follows since $(q_{L1}, q_{R1}) \Rightarrow_{se} (q_{L2}, q_{R2})\overset{\alpha}{\Rightarrow}_{se}(q'_{L2}, q'_{R2}) \Rightarrow_{se} (q'_{L1}, q'_{R1})$ implies $(q_{L1}, q_{R1})\overset{\alpha}{\Rightarrow}_{se}(q'_{L1}, q'_{R1})$.

Now, by the definition of $|[I]|$, since $(q_{L2}, q_{R2})\xrightarrow{\alpha}(q'_{L2}, q'_{R2})$ in $\mathbb{Q}_B$ and $\alpha \in \Sigma_s$, it must be the case that,

$$q_{x2}\xrightarrow{\alpha}q'_{x2} \text{ in } \mathbb{Q}_x{}'_{se}, \text{ for } x = L, R \text{ and} \tag{3.5}$$

$$f_L \circ P_L(q'_{L2}) = f_R \circ P_R(q'_{R2}) \tag{3.6}$$

But then by (3.5), $q_{x2}\overset{\alpha}{\Rightarrow}_{se}q'_{x2}$ in $\mathbb{Q}_x$ for $x = L, R$ (ie. for $x = L, R$ there exists q_{x3} and q'_{x3} such that $q_{x2} \Rightarrow_{se} q_{x3}\xrightarrow{\alpha}q'_{x3} \Rightarrow_{se} q'_{x2}$ in $\mathbb{Q}_x$). By (3.6) and the fact that $\Rightarrow_{se}$ does not change state outputs we can conclude that $f_L \circ P_L(q'_{L3}) = f_R \circ P_R(q'_{R3})$. But then by the definition of $|[I]|$ we have

$$(q_{L2}, q_{R2}) \Rightarrow_{se} (q_{L3}, q_{R3}) \xrightarrow{\alpha} (q'_{L3}, q'_{R3}) \Rightarrow_{se} (q'_{L2}, q'_{R2})$$

in $\mathbb{Q}_A$ so $(q_{L2}, q_{R2}) \overset{\alpha}{\Rightarrow}_{se} (q'_{L2}, q'_{R2})$ in $\mathbb{Q}_A$ and hence $(q_{L2}, q_{R2}) \xrightarrow{\alpha} (q'_{L2}, q'_{R2})$ in $\mathbb{Q}_A{}'_{se}$.

This completes our proof for Case 1.

Case 2: $\alpha \notin \Sigma_s$. Again assume that $(q_{L1}, q_{R1}) \xrightarrow{\alpha} (q'_{L1}, q'_{R1})$ in $\mathbb{Q}_B{}'_{se}$. Therefore $f_L \circ P_L(q'_{L1}) = f_R \circ P_R(q'_{R1})$ and in $\mathbb{Q}_B$ $(q_{L1}, q_{R1}) \overset{\alpha}{\Rightarrow}_{se} (q'_{L1}, q'_{R1})$. That is, there exists (q_{L2}, q_{R2}) and (q'_{L2}, q'_{R2}) such that in $\mathbb{Q}_B$

$$(q_{L1}, q_{R1}) \Rightarrow_{se} (q_{L2}, q_{R2}) \xrightarrow{\alpha} (q'_{L2}, q'_{R2}) \Rightarrow_{se} (q'_{L1}, q'_{R1})$$

Since $\alpha \notin \Sigma_s$, it must be the case that $q_{L2} = q'_{L2}$ and $q_{R2} \xrightarrow{\alpha} q'_{R2}$ in $\mathbb{Q}_R{}'_{se}$ or $q_{R2} = q'_{R2}$ and $q_{L2} \xrightarrow{\alpha} q'_{L2}$ in $\mathbb{Q}_L{}'_{se}$. Without loss of generality, assume that $q_{L2} = q'_{L2}$. From our work in Case 1, the problem reduces to showing that $(q_{L2}, q_{R2}) \xrightarrow{\alpha} (q'_{L2}, q'_{R2})$ in $\mathbb{Q}_B$ implies $(q_{L2}, q_{R2}) \overset{\alpha}{\Rightarrow}_{se} (q'_{L2}, q'_{R2})$ in $\mathbb{Q}_{\mathbb{A}}$. The argument is the same as for Case 1 except that we need only concern ourselves with the α transition in $\mathbb{Q}_R{}'_{se}$. $\square$

Since weak observation equivalence ignores differences resulting from unobservable τ transitions, in the weak state-event version of Theorem 3.29 below, we require that τ not be part of the synchronization set Σ_s.

Theorem 3.31 *Let SELTS $\mathbb{Q}_{Li}, \mathbb{Q}_{Ri}, i = 1, 2$ be given. If $\mathbb{Q}_{L1} \approx_{se} \mathbb{Q}_{L2}$ and $\mathbb{Q}_{R1} \approx_{se} \mathbb{Q}_{R2}$ then for all compatible interfaces $I := (\Sigma_s, f_L, f_R)$ such that $\tau \notin \Sigma_s$:*

$$(\mathbb{Q}_{L1}|[I]|\mathbb{Q}_{R1}) \approx_{se} (\mathbb{Q}_{L2}|[I]|\mathbb{Q}_{R2})$$

Proof: By the definition of weak state-event equivalence, $\mathbb{Q}_{L1}{}'_{se} \sim_{se} \mathbb{Q}_{L2}{}'_{se}$ and $\mathbb{Q}_{R1}{}'_{se} \sim_{se} \mathbb{Q}_{R2}{}'_{se}$. Thus, by Theorem 3.29

$$(\mathbb{Q}_{L1})'_{se}|[I]|(\mathbb{Q}_{R1})'_{se} \sim_{se} (\mathbb{Q}_{L2})'_{se}|[I]|(\mathbb{Q}_{R2})'_{se} \qquad (3.7)$$

The fact that $\sim_{se}$ implies $\approx_{se}$ together with (3.7) gives

$$(\mathbb{Q}_{L1})'_{se}|[I]|(\mathbb{Q}_{R1})'_{se} \approx_{se} (\mathbb{Q}_{L2})'_{se}|[I]|(\mathbb{Q}_{R2})'_{se}$$

Using the definition of $\approx_{se}$, we have

$$((\mathbb{Q}_{L1})'_{se}|[I]|(\mathbb{Q}_{R1})'_{se})'_{se} \sim_{se} ((\mathbb{Q}_{L2})'_{se}|[I]|(\mathbb{Q}_{R2})'_{se})'_{se}$$

By Lemma 3.30 $((\mathbb{Q}_{Li})'_{se}|[I]|(\mathbb{Q}_{Ri})'_{se})'_{se} = (\mathbb{Q}_{Li}|[I]|\mathbb{Q}_{Ri})'_{se}, i = 1, 2$ so

$$(\mathbb{Q}_{L1}|[I]|\mathbb{Q}_{R1})'_{se} \sim_{se} (\mathbb{Q}_{L2}|[I]|\mathbb{Q}_{R2})'_{se}$$

which by definition of $\approx_{se}$ is our desired result. □

If we restrict systems to synchronizing on observable transitions then the above result together with the fact that for any SELTS $\mathbb{Q} \approx_{se} \mathbb{Q}/\!\!/\theta_w$ means that we can take the synchronous product of the subsystems' quotient systems instead of the (typically) larger original systems. As we will see in the next chapter there are many system properties that are preserved by weak state-event equivalence, allowing us to use the reduced models resulting from the quotient systems for verification.

In the supervisory control of DES, the system supervisor can be viewed as another SELTS that imposes its control actions through running in a synchronous product with the plant. The importance of Theorem 3.31 then becomes apparent as it will eventually allow us to design a controller using the plant's weak quotient system in the case when all controllable transitions are observable.

3.5 Summary

The general state-event setting of SELTS with unobservable transitions is considered as a way of hiding complexity and inducing hierarchy through quotient systems. This setting leads to the development of state-event observers that are applicable to a wide variety of problems since SELTS are the underlying model of many discrete event formalisms.

State-event observers of SELTS represent a unifying framework for observers, and thereby hierarchy, in state and event based settings, enabling us to define observers in DES settings where both states and events are important (eg. Ostroff's TTMs). This unification of state and event methods is evidenced by the fact that the state observers of [Won76] and event based observation equivalences of Milner [Mil80], [Mil89] are both special cases of state-event observers. The unification of methodologies is obtained through the algebraic characterization of strong and weak state-event observers using the upper semilattice of compatible partitions of a SELTS. The algebraic characterization then enables appeal to efficient polynomial time algorithms for computing state-event observers based upon the Relational Coarsest Partition problem.

The algebraic characterization of state-event equivalence using SELTS homomorphisms aided us in demonstrating the compositional consistency of state-event equivalence. It is this important property that in the following two chapters will allow us to perform model reduction of composite systems for temporal logic model checking and hierarchically consistent control systems design.

Chapter 4

Model Reduction of Modules for State-Event Temporal Logics

In this chapter we utilize algebraic state-event structures to model systems together with state-event temporal logics as a means of specification. The main contribution of the chapter is a compositionally consistent model reduction technique for a class of "state-event stuttering-invariant" temporal formulas. In particular, the method provides a means of "weak" model reduction for a discrete time temporal logic that is a simplification of Ostroff's RTTL [Ost89]. The principal ideas of this chapter were first outlined in [LOW96]. Justification of our choice of a combined state-event and discrete time setting can be found in Section 1.1. We will therefore begin with a comparison of our work to previous works that outlines the sense in which our model reduction technique is both weak and compositionally consistent.

While symbolic model-checking techniques such as [McM92] have proven effective for some very large systems [BCM92], the largest of these systems typically come from the digital hardware domain and have a great deal of regularity in their state transition structure that can be exploited by the symbolic techniques to obtain compact representations of large systems. If one wishes to model-check large concurrent systems lacking in symmetry, larger digital hardware systems, or simply to reduce the computation time required, one must perform some sort of model reduction.

In model reduction one starts out with a system for which one would like to verify (model-check) formulas from a particular set of formulas or class of temporal formulas that are of interest. To facilitate the verification process, or, in some cases, make the problem tractable, a reduced model is obtained such that, if the reduced model satisfies the temporal formulas under investigation, then the original system satisfies the temporal formulas. If the model-checking of a formula on the reduced model provides a definitive answer regarding the satisfaction of the formula in the original (unreduced) system, we say the reduction technique is *exact*. If this model reduction technique is performed so that the mutual satisfaction of formulas is only guaranteed for a specific finite set of formulas, we say that the method is a *formula-specific* model reduction technique. But if, as in this thesis, the method always guarantees the mutual satisfaction of all formulas in a class of temporal formulas, we refer to the technique as being *formula-independent* for the given class of formulas. For example, the weak model reduction technique of Section 4.3 is formula-independent for the class of "State-Event Stuttering-Invariant" formulas defined later in this chapter. In addition to preserving the truth values of a particular class of temporal formulas, the model reduction technique presented here is "compositionally consistent" in the sense that for any formula from a defined class of formulas, the composition of two reduced models satisfies the formula iff the composition of the two original systems satisfies the formula.

In addition to the above terms, in our comparison of previous works with the work at hand, we will make distinctions between "strong" and "weak" model reduction techniques. In a strong reduction technique, a single transition in the original system model results in a single transition in the reduced model. In weak model reduction techniques, a single transition may be used by the reduced model to represent a finite sequence of transitions in the original system model. The result is that weak model reduction techniques tend to achieve a greater reduction in state size at the expense of preserving the truth values of fewer formulas and requiring greater computational effort to compute the reduced system. We now provide some additional motivation for the use of weak reduction techniques.

In concurrent systems built from interacting modules, we are interested in specifying a module's observable behavior or "interface" with other systems. If two modules produce identical behavior at their interfaces and differ only in their internal behavior, then they should satisfy the same interface specification. While many temporal logics have been successfully used to specify systems' behaviors, straightforward application of temporal logics is often too discriminating with respect to the internal actions of concurrent systems. Since we want to be able to reason about observed events and changes in the system's state output, we define "weak satisfaction" of a temporal formula to provide us with a class of state-event stuttering-invariant formulas which is similar to stuttering-invariant formulas of [MP92] with some key differences as a result of our state-event setting. The main result of the chapter provides a compositionally consistent, weak model reduction technique for the class of state-event stuttering-invariant formulas by model-checking a system's weak state-event quotient system.

Methods based upon abstract interpretations such as [BBCS92], [CGL94] and [DGG94], provide examples of strong, formula specific model reduction. Although they are "strong" techniques, these methods can provide a significant reduction in state size by an appropriate choice of abstraction. The development of the abstract model can be an iterative process, with the mapping between concrete and abstract domains being refined when there is insufficient information at the abstract level to determine the truth value of one of the formulas of interest. The creation of these abstractions typically requires some insight from the systems designer.

By dropping "immediate" operators from the temporal formulas and considering events with interleavings that do not affect the the truth values of the formulas of interest, Valmari has similarly been able to achieve substantial state reduction [Val90]. The method, which makes use of "stubborn sets," has an extension to a "weak" observational setting, but the reduced models still suffer from the fact that they are dependent upon the formulas to be verified. As a result of this dependency, changes to the system's specifications require the computation of a new reduced model. All of the above formula dependent techniques suffer from an inability to guarantee compositional consistency. Hence, to verify a composition of systems using these methods, one is forced to compute the composition of the original systems and then perform model reduction for the specific formulas on the (generally much larger) composite system.

For the logic CTL*, a superset of linear and branching temporal logics, strong bisimulation preserves the truth values of the standard satisfaction relation for all formulas [BCG87], [Jos90]. In practice strong bisimulation equivalence is often too strong to provide a significant reduction in the state size of the model. While this deficiency spawned the formula specific reductions described above, it also led to formula-independent methods that achieve greater reduction at the price of preserving the truth values of a smaller class of formulas.

The formula-independent methods of model reduction are typically based upon algebraic equivalences derived from the work of Hoare [Hoa85] and Milner [Mil89]. In [KV91], [KV92], Kaivola and Valmari provide a method of "weak" model reduction for a nexttime-less linear temporal logic based upon failure equivalence [Hoa85]. As one might expect, the algorithm is worst case exponential. The papers deal with state based models that are converted into event oriented models by labeling transitions with the changes they cause in the states (similar to [Law92], [LW95]). In [Kai96] Kaivola investigates the truth preserving properties of the equivalences of [KV91, KV92] in a compositional setting, with state and event based parallel composition operators. Real-time aspects are not explicitly considered in the temporal logic used and immediate operators are forbidden. Another state-event setting is that of [GL93] where the separation of state values and event labels allows the use of the standard event synchronization parallel composition operators. In [GL93] Graf and Loiseaux provide conditions under which abstractions preserving safety properties expressible in a fragment of the branching time μ-calculus are compositionally consistent. Their underlying model of state-event systems, which is equivalent to the State-Event Labeled Transition Systems (SELTS) used in this thesis, can model the state-event synchronous product of systems. This is a "strong" abstraction that does not deal with fairness properties.

In our work we provide a method for "weak," compositionally consistent model reduction for state event systems that preserves a class of safety and fairness properties related to systems' observed behaviors. The state-event equivalence relation we use for our form of formula-independent model reduction is an extension of Milner's weak observation (bisimulation) equivalence. Kaivola and Valmari rejected weak observation equivalence for model reduction on the grounds that it did not necessarily preserve fairness properties due to its inability to distinguish divergences (infinite sequences of unobservable events). This problem does not arise in Ostroff's RTTL which has the requirement that an (observable) *tick* of the global clock must occur infinitely often in any legal computation.

In the following section we use SELTS to model modules that can be combined via parallel composition operators to create new modules and systems. We also define a simple (real-time) state-event temporal logic that can be used for system specification. Section 4.2 demonstrates how strong state-event equivalence can be used as the basis of a strong, compositionally consistent and computationally efficient model reduction technique for our entire logic. Section 4.3 develops a weak model reduction technique for the subclass of state-event stuttering-invariant temporal formulas through the use of weak state-event equivalence and the results of the previous section. Greater state reduction is achieved through the restriction of the formulas to be preserved. It should be noted that all the reduced models of this chapter are computable in polynomial time, thereby permitting practical application of the methods. To conclude the chapter, we prove that our model reduction technique for SELTS can be applied to a special case of TTM parallel composition where the TTMs being composed have well defined "compatible" interfaces. A TTM together with an interface specification is called TTM module.

4.1 A Simple Real-Time State-Event Temporal Logic

In this section we first define the state-event sequences associated with a SELTS as a way of capturing the behavior of a SELTS. We then introduce state-event temporal logics as an abstract method for reasoning about SELTS behavior with particular attention being paid to a simple real-time logic.

In general when discussing SELTS throughout this chapter $AP, AP_1, AP_2, \ldots$ will represent sets of atomic propositions and the SELTS state output map will map each state to the set of atomic propositions satisfied by the state (i.e. $P : Q \to \mathcal{P}(AP)$). We make a slight modification of the state-event synchronous product operator of Definition 2.13 to allow the straightforward application of the definition of satisfaction of temporal formulas in subsequent sections. The state output map of the state-event synchronous product is changed to map a state of the composite system to the union of the state outputs for the component subsystems. This poses a problem when the systems share some variables or atomic propositions. For example, if $P_1(q_{10}) = \{u = 0, v = 1\}$ in $\mathbb{Q}_1$ and $P_2(q_{20}) = \{v = 2, w = 1\}$ in $\mathbb{Q}_2$ then (q_{10}, q_{20}), the initial state of the composite system, would have a state output of $\{u = 0, v = 1, v = 2, w = 1\}$ - which is not a consistent set of assignments for the shared variable v. In order to ensure that the set of atomic propositions satisfied by a state of the composite system remains consistent, we will restrict the state output synchronization maps of the interface definition. We will consider interfaces of the form $I := (\Sigma_s, \pi_{AP_2}, \pi_{AP_1})$. Otherwise the definition remains unchanged. This modification does not affect the results of Section 3.4 regarding compositional consistency.

Definition 4.1 *Given two SELTS,* $\mathbb{Q}_i = \langle Q_i, \Sigma_i, R^i_\Sigma, q_{i0}, P_i \rangle$, $i = 1, 2$ *where* $P_i : Q_i \to \mathcal{P}(AP_i)$ *for* $i = 1, 2$ *and a compatible interface* $I := (\Sigma_s, \pi_{AP_2}, \pi_{AP_1})$ *where* $\Sigma_s \subseteq \Sigma_1 \cap \Sigma_2$ *and* $\pi_{AP_2} : \mathcal{P}(Q_1) \to \mathcal{P}(Q_1)$ *such that* $A \mapsto A \cap AP_2$ *and* $\pi_{AP_2} : \mathcal{P}(Q_2) \to \mathcal{P}(Q_2)$ *such that* $A \mapsto A \cap AP_1$. *Then the* I-synchronous product of $\mathbb{Q}_1$ and $\mathbb{Q}_2$ *is given by:*

$$\mathbb{Q}_1|[I]|\mathbb{Q}_2 := \langle Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, R_{\Sigma_1 \cup \Sigma_2}, (q_{10}, q_{20}), P\rangle$$

Here $P : Q_1 \times Q_2 \to \mathcal{P}(AP_1 \cup AP_2)$ is defined by $P((q_1, q_2)) = P_1(q_1) \cup P_2(q_2)$ and the elements of $R_{\Sigma_1\cup\Sigma_2} = \{\xrightarrow{\alpha} : \alpha \in \Sigma_1 \cup \Sigma_2\}$ are binary relations over $Q_1 \times Q_2$ defined as follows: $(q_1, q_2)\xrightarrow{\alpha}(q_1', q_2')$ iff

$\pi_{AP2} \circ P_1(q_1') = \pi_{AP1} \circ P_2(q_2')$ (†)

and

(i) $\alpha \in \Sigma_s$, and $q_i\xrightarrow{\alpha}q_i'$ in $\mathbb{Q}_i$ for $i = 1, 2$, or

(ii) $\alpha \notin \Sigma_s$, $q_1\xrightarrow{\alpha}q_1'$ in $\mathbb{Q}_1$ and $q_2 = q_2'$, or

(iii) $\alpha \notin \Sigma_s$, $q_2\xrightarrow{\alpha}q_2'$ in $\mathbb{Q}_2$ and $q_1 = q_1'$.

Condition (†) states that state output maps of the reachable states agree on the subsets of propositions from $AP_1 \cap AP_2$ that they satisfy (e.g. $P_1(q_1') \cap AP_2 = P_2(q_2') \cap AP_1$).

4.1.1 Computations of SELTS

Before defining the computations of a SELTS, we will introduce some notation to aid in our discussion of generated and observed state-event sequences. We are interested in sequences of both states and events so for notational convenience we define $\Sigma_- := \Sigma \cup \{-\}$ and $S := Q \times \Sigma_-$. For $s = (q, \alpha) \in S$, in addition to the set of atomic propositions found in $P(q)$ we associate the atomic proposition $\eta = \alpha$. We refer to η as the (next) transition variable. The computations of the SELTS $\mathbb{Q}$ will then be a subset of the union of the set of all finite, non-empty, state-event sequences S^+, and the set of all infinite state-event sequences S^ω. As a notational convenience, we introduce the notation $|\sigma|$, which for $\sigma = s_0 s_1 s_2 \ldots s_n \in S^+$ is defined as $|\sigma| = n$ and for $\sigma = s_0 s_1 s_2 \ldots \in S^\omega$, $|\sigma| = \omega$.

Definition 4.2 *Given a SELTS $\mathbb{Q}$, the set of* **computations** *of $\mathbb{Q}$, denoted $\mathcal{M}(\mathbb{Q})$, is the largest subset of $S^+ \cup S^\omega$ such that for all $\sigma \in \mathcal{M}(\mathbb{Q})$,*

$$\sigma = \begin{cases} s_0 s_1 \dots s_n &= (q_0, \alpha_0)(q_1, \alpha_1) \dots (q_n, -) \in S^+, \quad \textit{or,} \\ s_0 s_1 \dots &= (q_0, \alpha_0)(q_1, \alpha_1) \dots \in S^\omega \end{cases}$$

and

(i) Initialization: q_0 is the initial state of $\mathbb{Q}$.

(ii) Succession: $0 \leq i < |\sigma|$ implies $\alpha_i \in \Sigma$ and $q_{i+1} \in \alpha^{\mathbb{Q}}(q_i)$ (i.e. $q_i \overset{\alpha}{\rightarrow}_i q_{i+1}$ in $\mathbb{Q}$).

(iii) Diligence: $\alpha_i = -$ iff $i = |\sigma|$ and for all $\alpha \in \Sigma, \alpha^{\mathbb{Q}}(q_i) = \emptyset$.

In Definition 4.2, the purposes of conditions (i) and (ii) are, respectively, to guarantee that the computation starts in the system's initial state and that the change from one state to the next via the given event is possible in $\mathbb{Q}$. Condition (iii) states that the only finite sequences in $\mathcal{M}(\mathbb{Q})$ are those which terminate in a state where no transitions are possible and hence the final "event" of the state-event sequence is denoted by $-$. This diligence condition differs from that of [MP92] in that there is no idling transition in our setting so we allow finite sequences of states to be computations and modify our definition of temporal semantics accordingly [Arn94].

4.1.2 Temporal Logic of State-Event Sequences

We now give a brief summary of temporal logic and refer the reader to [MP92], [Ost89] and [Arn94] for the full details. Following [Ost89], the state-event sequences defined above will play the role of the state sequences in [MP92]. This will allow us to distinguish state formulas and state-event formulas. RTTL, as an example of a state-event temporal logic, is based upon Manna-Pnueli temporal logic with additional proof rules for dealing with real-time (*tick* event) properties. To allow us to express simple real-time properties we add a bounded "until" operator.

State-event formulas are arbitrary boolean combinations of atomic predicates. We say that a state-event formula is *state formula* if is does not include any transition predicates such as $\eta = \alpha$. For example, $(y \leq 10 \wedge x = atdelay) \vee t = 5$ is both a state formula and a state-event formula while $\eta = \alpha \vee y = 3$ is a state-event formula but not a state formula. State-event formulas (and hence state formulas) do not contain any temporal operators. For a state formula F_s and a state q, we use the standard inductive definition of satisfaction and write $q \models F_s$ when F_s is true in state q. Similarly the definition of satisfaction can be extended to any state-event pair $s \in S$ and any state-event formula F_{se}.

In the following inductive definition of satisfaction of temporal state-event formulas we will consider an arbitrary (possibly finite) state-event sequence $\sigma = s_0 s_1 \ldots = (q_0, \alpha_0)(q_1, \alpha_1) \ldots$ Henceforth σ^k will be used to denote the k-shifted suffix of σ,

$$\sigma^k := s_k s_{k+1} \ldots = (q_k, \alpha_k)(q_{k+1}, \alpha_{k+1}) \ldots$$

when it exists (i.e. when $|\sigma| \geq k$). When talking about projections of computations we will denote the prefix of σ up to position k by $\sigma^{-k} = (q_0, \alpha_0)(q_1, \alpha_1) \ldots (q_k, \alpha_k)$. For each $\alpha \in \Sigma$ we use the notation $\#\alpha(\sigma, i)$ to denote the number of α transitions that occur between q_0 and q_i in the state-event sequence σ. If $|\sigma| < i$ then $\#\alpha(\sigma, i)$ is undefined.

Definition 4.3 *(Satisfaction) For temporal formulas F, F_1, F_2 and state-event sequence σ, the satisfaction relation is defined as follows:*

- *If $F \in AP$ is an atomic predicate, then $\sigma \models F$ iff $s_0 \models F$ (i.e. $F \in P(q_0)$)*
- *If $F := (\eta = \alpha)$, then $\sigma \models F$ iff $\alpha_0 = \alpha$*
- *$\sigma \models F_1 \vee F2$ iff $\sigma \models F_1$ or $\sigma \models F_2$*
- *$\sigma \models F_1 \wedge F2$ iff $\sigma \models F_1$ and $\sigma \models F_2$*
- *$\sigma \models \neg F$ iff $\sigma \not\models F$*

- $\sigma \models \bigcirc F$ *iff* σ^1 *exists and* $\sigma^1 \models F$
- $\sigma \models F_1 \mathcal{U} F_2$ *iff* $\sigma \models F_2$ *or* $\exists k > 0$ *such that* σ^k *is defined,* $\sigma^k \models F_2$ *and* $\forall i, 0 \leq i < k, \sigma^i \models F_1$.
- $\sigma \models F_1 \mathcal{U}^{\alpha}_{[l,u]} F_2$ *iff* $\sigma \models F_2$ *or* $\exists k > 0$ *such that* σ^k *is defined,* $\sigma^k \models F_2$ *and* $\forall i, 0 \leq i < k, \sigma^i \models F_1$ *and* $l \leq \#\alpha(\sigma, k) \leq u$.

The "next" operator $\bigcirc$ and "until" operator $\mathcal{U}$ are typically used to define addition operators. In particular the "eventually" operator $\Diamond F$, which denotes $(true)\mathcal{U}F$, and the "henceforth" operator $\Box F$, which is an abbreviation of $\neg\Diamond\neg F$. As an example of a temporal formula, consider $F := \Box \bigcirc true$. F is satisfied only by those σ such that $|\sigma| = \omega$. The $\mathcal{U}^{\alpha}_{[l,u]}$ operator is just the until operator subject to the restriction that for a formula $F_1 \mathcal{U}^{\alpha}_{[l,u]} F_2$, F_2 must become true after the lth occurrence of α and before the $(u+1)$th occurrence of α. In systems in which time is represented by discrete *tick* events the $\mathcal{U}^{tick}_{[l,u]}$ operator can be used to specify that a system meets hard time bounds. For example, any system satisfying the formula $(true)\mathcal{U}^{tick}_{[0,2]}(\eta = \beta)$ will produce a β event before 3 time units have passed. We will use $\mathcal{U}^{tick}_{\leq k}$ as an abbreviation for $\mathcal{U}^{tick}_{[0,k]}$. For example the above formula can be written as $(true)\mathcal{U}^{tick}_{\leq 2}(\eta = \beta)$.

Definition 4.4 *Given a SELTS* $\mathbb{Q}$ *and a temporal formula* F*, we say that* F *is* $\mathbb{Q}$-valid*, written* $\mathbb{Q} \models F$*, iff for all* $\sigma \in \mathcal{M}(\mathbb{Q})$*,* $\sigma \models F$*.*

Fairness

Typically when a given transition structure is used as the model for a system, a designer specifies some fairness constraints which a computation must satisfy if it is to be considered a "legal" computation of the system. For example, all systems in RTTL have the fairness constraint that the *tick* event must occur infinitely often ($\Box\Diamond(\eta = tick)$), that is the system must not stop the clock or permit an infinite number of non-*tick* transitions to occur between successive clock *tick*s. Given a specification as a temporal formula F, one then is not so much interested in verifying that *all* the computations of the transition structure satisfy F but rather in verifying that all the *legal* computations satisfy F. That is $\mathbb{Q} \models \neg F_{fair} \vee F$, where F_{fair} is the conjunction of all formulas that are to be satisfied by the system's legal computations. In performing such a verification one implicitly assumes that the set of legal computations considered is non-empty (i.e. $\exists \sigma \in \mathcal{M}(\mathbb{Q}), \sigma \models F_{fair}$).

4.2 Strong State-Event Model Reduction

In this section we assume that while we have perfect event information (all events including τ events are observable), only partial state information is provided via the state output map. The main result of this section is that strongly state-event equivalent systems satisfy the same temporal formulas and hence we can use a system's strong state-event quotient system to verify system properties. The compositional consistency of this model reduction technique then follows immediately from the result on strong ("algebraic") compositional consistency of Section 3.4. While the results obtained in this section follow easily from the truth preserving properties of strong bisimulation equivalence, the technique employed in this section will be utilized in the following section on weak state-event model reduction.

In the following, unless stated otherwise, we assume that we are dealing with a SELTS $\mathbb{Q} = \langle Q, \Sigma, R_\Sigma, q_0, P \rangle$ where P is the state output map $P : Q \to \mathcal{P}(AP)$, and AP is the set of atomic predicates of interest.

Given a computation σ, the *strongly observed computation* generated by σ is given by applying P to the state of each state-event pair in the computation. This provides a map from sequences over $Q \times \Sigma_-$ to sequences over $\mathcal{P}(AP) \times \Sigma_-$.

$$P^{\sim} : (Q \times \Sigma_-)^+ \cup (Q \times \Sigma_-)^\omega \to (\mathcal{P}(AP) \times \Sigma_-)^{\cup}(\mathcal{P}(AP) \times \Sigma_-)^\omega$$
$$(q_0, \alpha_0)(q_1, \alpha_1) \ldots (q_n, \alpha_n) \ldots \mapsto (P(q_0), \alpha_0)(P(q_1), \alpha_1) \ldots (P(q_n), \alpha_n) \ldots$$

For C, a set of computations, we define $P^{\sim}(C) := \{P^{\sim}(\sigma) : \sigma \in C\}$.

Lemma 4.5 *Let $\mathbb{Q}_i = \langle Q_i, \Sigma, R^i_\Sigma, q_{i0}, P_i \rangle, i = 1, 2$ be SELTS. If $\mathbb{Q}_1 \sim_{se} \mathbb{Q}_2$ then $P_1^{\sim}(\mathcal{M}(\mathbb{Q}_1)) = P_2^{\sim}(\mathcal{M}(\mathbb{Q}_2))$*

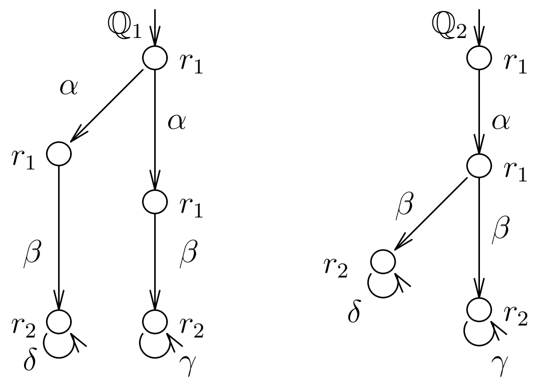


Figure 4.1: Counterexample to converse of Lemma 4.5.

Proof: Follows immediately from the definitions of $\sim_{se}$ and $P^{\sim}$. $\square$

The systems in Figure 4.1 demonstrate that the converse of Lemma 4.5 is false. The transition systems are shown with the state outputs generated by their respective state output maps P_1 and P_2 next to each state. The initial states of the two transition systems are marked by entering arrows. In this case $P_1^{\sim}(\mathcal{M}(\mathbb{Q}_1)) = P_2^{\sim}(\mathcal{M}(\mathbb{Q}_2)) = \{r_1 \xrightarrow{\alpha} r_1 \xrightarrow{\beta} r_2 \xrightarrow{\delta} r_2 \xrightarrow{\delta} \ldots, r_1 \xrightarrow{\alpha} r_1 \xrightarrow{\beta} r_2 \xrightarrow{\gamma} r_2 \xrightarrow{\gamma} \ldots\}$, but one can easily verify that $\mathbb{Q}_1 \not\sim_{se} \mathbb{Q}_2$. By extending Hoare's failure equivalence to a state-event failure equivalence in a manner similar to the way that (event) observation equivalence was extended to state-event observation equivalence, one obtains an equivalence which relates the two systems of Figure 4.1. Unfortunately the computation of failure equivalence is PSPACE-complete [KS83] making it unlikely that an efficient algorithm could be found to compute any extension to the state-event setting. On the other hand strong state-event equivalence is $O(n \log m)$ making state-event equivalence preferable as a practical model reduction technique.

As an immediate consequence of Lemma 4.5, we obtain the following result.

Theorem 4.6 *Given two SELTS as above, if* $\mathbb{Q}_1 \sim_{se} \mathbb{Q}_2$ *then for any temporal formula* F*, we have* $\mathbb{Q}_1 \models F$ *iff* $\mathbb{Q}_2 \models F$.

The above theorem allows us to use a system's strong state-event quotient system to reason about the state output and event behavior of the system since $\mathbb{Q} \sim_{se} \mathbb{Q}/\theta_s$. Lemma 3.29 provides the following Corollary to Theorem 4.6.

Corollary 4.7 *Strong state-event equivalence can be used for compositionally consistent model reduction of SELTS for all formulas in state-event temporal logic.*

4.3 Weak State-Event Model Reduction

We now turn our attention to the case with only partial event observations in addition to the partial state observations provided by the state output map. We assume that all unobservable transitions are labeled by τ. In this case we want to reason about the sequences of observed events and changes in state output. To this end we define a projection from computations to weakly observed computations similar to the strongly observed computation projection of the previous section. This time we delete a state-event pair from the strongly observed computation if the event is an unobservable τ transition and the state output remains unchanged in the next state (i.e. there is no way to observe whether we remain in the current state or take the τ transition to the next state). Since weak state-event equivalence suppresses system information regarding sequences of unobservable events that do not cause state changes, the equivalence can only be used for model reduction with a restricted set of temporal formulas. This restricted class, which we will call the class of State-Event Stuttering-Invariant (SESI) formulas, is characterized as those formulas that are satisfied by a computation iff the projected computation satisfies the formula.

We identify a set of SESI formulas, including some formulas making use of immediate operators ($\bigcirc, \eta =$). The main result of the section states that weakly state-event equivalent systems satisfy the same subset of SESI formulas. Thus for a given module we can perform compositionally consistent model reduction by computing the system's weak state-event quotient system and then using the quotient system to model-check all the formulas of interest, provided the formulas are SESI.

4.3.1 Weakly Observed Computations

For the remainder of the section, unless stated otherwise, we assume that we are dealing with a SELTS $\mathbb{Q} := \langle Q, \Sigma, R_\Sigma, q_0, P\rangle$ where the state output map $P : Q \rightarrow$

$\mathcal{P}(AP)$.

In [MP92] the authors use a state-based projection operator to develop a state-only version of weak satisfaction. They define the *reduced behavior* of a computation σ via a two step process that amounts to first applying $P^{\sim}$, the strong computation projection of the previous section, and then replacing uninterrupted sequences of identical "states" with a single copy of the state. In our case we are dealing with sequences of state-event pairs rather than just sequences of states. We cannot simply apply $P^{\sim}$ and then replace subsequences of uninterrupted state-event pairs by a single state-event pair since in this case important information relating state changes and event observations would be lost.

Consider the three state-event sequences shown below where *tick* is the event representing the passage of one second on the global clock.

$$\begin{array}{l}(q_0,\tau)(q_0,\tau)(q_0,tick)(q_0,\alpha)(q_1,tick)\ldots\\(q_0,\tau)(q_0,tick)(q_0,tick)(q_0,\alpha)(q_1,tick)\ldots\\(q_0,tick)(q_0,\tau)(q_0,tick)(q_0,\tau)(q_0,\alpha)(q_1,tick)\ldots\end{array}$$

If we assume that the state output map is the identity map, then following [MP92] the first and second sequences would result in the same reduced computation:

$$(q_0,\tau)(q_0,tick)(q_0,\alpha)(q_1,tick)\ldots$$

while the third sequence is its own reduced computation. This would lead us to believe that in the first two cases the system delays for one second and then changes state from q_0 to q_1 via an α transition when, in fact, the second and third computations do not make the α transition until after 2 seconds. While we want our projection operator to distinguish the first case from the other two, the second and third computations differ only by unobservable transitions that do not change the state output. Upon rewriting the three sequences in terms of the notation of weak state-event observation equivalence, the differences and similarities in observed behaviors become apparent:

$$\left.\begin{array}{c} q_0\xrightarrow{\tau}q_0\xrightarrow{\tau}q_0\xrightarrow{tick}q_0\xrightarrow{\alpha}q_1\xrightarrow{tick}\dots \\ q_0\xrightarrow{\tau}q_0\xrightarrow{tick}q_0\xrightarrow{tick}q_0\xrightarrow{\alpha}q_1\xrightarrow{tick}\dots \\ q_0\xrightarrow{\tau}q_0\xrightarrow{tick}q_0\xrightarrow{\tau}q_0\xrightarrow{tick}q_0\xrightarrow{\alpha}q_1\xrightarrow{tick}\dots \end{array}\right\} \overset{P}{\mapsto} \left\{\begin{array}{c} q_0\overset{tick}{\Rightarrow}_{se}q_0\overset{\alpha}{\Rightarrow}_{se}q_1\overset{tick}{\Rightarrow}_{se}\dots \\ q_0\overset{tick}{\Rightarrow}_{se}q_0\overset{tick}{\Rightarrow}_{se}q_0\overset{\alpha}{\Rightarrow}_{se}q_1\overset{tick}{\Rightarrow}_{se}\dots \\ q_0\overset{tick}{\Rightarrow}_{se}q_0\overset{tick}{\Rightarrow}_{se}q_0\overset{\alpha}{\Rightarrow}_{se}q_1\overset{tick}{\Rightarrow}_{se}\dots \end{array}\right.$$

To an external observer the second and third computations would produce the same observed state-event sequence: $(q_0, tick)(q_0, tick)(q_0, \alpha)(q_1, tick)\dots$. The projection defined below has the effect of replacing all the state-event pairs making up an observed transition $q_1\overset{\alpha}{\Rightarrow}_{se}$, with a single state-event pair $q_1\xrightarrow{\alpha}$. The following weak state-event sequence projection operator produces a system's weakly observed computations.

Definition 4.8 *Given an SELTS $\mathbb{Q}$ with state output map $P : Q \to \mathcal{P}(AP)$ and $\sigma = (q_0, \alpha_0)(q_1, \alpha_1)\dots$, $\sigma \in \mathcal{M}(\mathbb{Q})$, the* weakly observed behavior *of σ is denoted by $P^{\approx}(\sigma)$ which is defined inductively as follows:*

$$\begin{aligned} P^{\approx}(q_0) &= P(q_0) \\ P^{\approx}(q_0\xrightarrow{\alpha_0}q_1\xrightarrow{\alpha_1}\dots q_n\xrightarrow{\alpha_n}q_{n+1}) &= \begin{cases} P^{\approx}(q_0\xrightarrow{\alpha_0}q_1\xrightarrow{\alpha_1}\dots q_n), \text{if } \alpha_n = \tau \wedge P(q_n) = P(q_{n+1}) \\ P^{\approx}(q_0\xrightarrow{\alpha_0}q_1\xrightarrow{\alpha_1}\dots q_n)\xrightarrow{\alpha_n}P(q_{n+1}), \text{otherwise} \end{cases} \end{aligned}$$

For C a set of computations, we define $P^{\approx}(C) := \{P^{\approx}(\sigma) : \sigma \in C\}$.

Example 4.9 *In this example we consider the weak state-event observations generated by an SELTS with identity state output map $P := I_Q$ where $I_Q : Q \to Q$.*

$$\begin{aligned} \sigma_1 &= (q_0,\tau)(q_0,\alpha)(q_0,\tau)(q_1,\tau)(q_1,\beta)(q_2,\alpha)\dots = q_0\xrightarrow{\tau}q_0\xrightarrow{\alpha}q_0\xrightarrow{\tau}q_1\xrightarrow{\tau}q_1\xrightarrow{\beta}q_2\xrightarrow{\alpha}\dots \\ P^{\approx}(\sigma_1) &= q_0\xrightarrow{\alpha}q_0\xrightarrow{\tau}q_1\xrightarrow{\beta}q_2\xrightarrow{\alpha}\dots = (q_0,\alpha)(q_0,\tau)(q_1,\beta)(q_2,\alpha)\dots \\ \sigma_2 &= (q_0,\tau)(q_0,\tau)(q_0,\tau)\dots = q_0\xrightarrow{\tau}q_0\xrightarrow{\tau}q_0\xrightarrow{\tau}\dots \\ P^{\approx}(\sigma_2) &= q_0 = (q_0,-) \end{aligned}$$

In $P^{\approx}(\sigma_1)$ all the τ transitions are eliminated except for the $q_0\xrightarrow{\tau}q_1$ transition since this τ transition can be inferred from the external observer's observation of a state

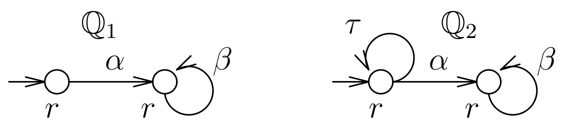


Figure 4.2: $\mathbb{Q}_1 \approx_{se} \mathbb{Q}_2$ but $P_1^{\approx}(\mathcal{M}(\mathbb{Q}_1)) \neq P_2^{\approx}(\mathcal{M}(\mathbb{Q}_2))$

change from q_0 to q_1 without any observed event. In this case we say that τ is an *implicitly observed transition.* The computation σ_2 is initially observed to be in state q_0 and then produces no state change or event observations. This is reflected in $P^{\approx}(\sigma_2)$ as $(q_0, -)$, the observed state output with no defined transition. Thus an infinite state-event sequence can result in a finite weakly observed sequence. This is why the effort was made earlier to extend the definition of temporal operators to finite as well as infinite sequences, allowing us to define weak satisfaction of temporal formulas below.

As the basis of weak state-event model reduction, we would like to obtain a result similar to Lemma 4.5 which stated that strongly state-event equivalent systems result in the same set of strongly observed computations. In this case we have to be careful with our treatment of the unobservable transitions that are erased by the weak projection. Consider the two weakly state-event equivalent systems shown in Figure 4.2. Here $r \in \mathcal{P}(AP)$ is the same state output for all the systems' states. In this case $P_1^{\approx}(\mathcal{M}(\mathbb{Q}_1)) = \{r \xrightarrow{\alpha} r \xrightarrow{\beta} r \xrightarrow{\beta} r \xrightarrow{\beta} \ldots\}$ but $P_2^{\approx}(\mathcal{M}(\mathbb{Q}_1)) = \{r, r \xrightarrow{\alpha} r \xrightarrow{\beta} r \xrightarrow{\beta} r \xrightarrow{\beta} \ldots\}$. The above systems agree upon their trajectories that produce an infinite number of observations. It is the infinite sequence of unobservable τ's that $\mathbb{Q}_2$ can produce that causes the discrepancy. This observation is formalized in the following two lemmas.

The first lemma states that a system and its observational closure (see Definition 3.17 on p. 58) produce the same infinite weakly observed computations (i.e. $\{P^{\approx}(\sigma) : \sigma \in \mathcal{M}(\mathbb{Q})$ and $|\mathbb{P}^{\approx}(\sigma)| = \omega\} = \{\mathbb{P}^{\approx}(\sigma) : \sigma \in \mathcal{M}(\mathbb{Q}'_{\sim})$ and $|\mathbb{P}^{\approx}(\sigma)| = \omega\}$).

Lemma 4.10 *Given an SELTS $\mathbb{Q} = \langle Q, \Sigma, R_\Sigma, q_0, P\rangle$, where $P : Q \to \mathcal{P}(AP)$,*

$$P^{\approx}(\mathcal{M}(\mathbb{Q})) \cap (\mathcal{P}(AP) \times \Sigma)^{\omega} = P^{\approx}(\mathcal{M}(\mathbb{Q}'_{se})) \cap (\mathcal{P}(AP) \times \Sigma)^{\omega}$$

Proof: $\mathcal{M}(\mathbb{Q}) \subseteq \mathcal{M}(\mathbb{Q}'_{se})$ since the transition relations of $\mathbb{Q}$ are a subset of those of

$\mathbb{Q}'_{se}$. Thus the containment of sets in the $\subseteq$ direction is trivial.

To show containment in the $\supseteq$ direction we begin by assuming that

$$\gamma \in P^{\approx}(\mathcal{M}(\mathbb{Q}'_{se})) \cap (\mathcal{P}(AP) \times \Sigma)^{\omega}$$

Then there exists $\sigma_r \in \mathcal{M}(\mathbb{Q}'_{se})$ such that $\gamma = P^{\approx}(\sigma_r)$. In particular, since σ_r produces an infinite number of observations (i.e. it does not end with an infinite sequence of self-looped τ transitions), for simplicity we can chose σ_r so that in $\sigma_r = q_0 \xrightarrow{\alpha_0} q_1 \xrightarrow{\alpha_1} q_2 \xrightarrow{\alpha_2} \dots$ there are no transitions resulting from self-looped τ transitions. Then as a result of the definition of the observational closure operator, each transition $q_i \xrightarrow{\alpha_i} q_{i+1}$ in $\mathbb{Q}'_{se}$ can be matched by a sequence of transitions $q_i \overset{\alpha_i}{\Rightarrow}_{se} q_{i+1}$ in $\mathbb{Q}$. The silent τ transitions that help make up the $\overset{\alpha_i}{\Rightarrow}_{se}$ relation leave the state output unchanged and hence the sequence of states and transitions making up the $q_i \overset{\alpha_i}{\Rightarrow}_{se} q_{i+1}$ relation will produce the same projected results as the $q_i \xrightarrow{\alpha_i} q_{i+1}$ transition in $\mathbb{Q}'_{se}$. We simply take the finite sequences of states and transitions making up each matching $q_i \xrightarrow{\alpha_i} q_{i+1}$ relation to obtain a σ_l with $P^{\approx}(\sigma_l) = P^{\approx}(\sigma_r)$. We know $\sigma_l \in \mathcal{M}(\mathbb{Q})$ because it is an infinite sequence of transitions in $\mathbb{Q}$. $\Box$

Lemma 4.11 *Given two SELTS,* $\mathbb{Q}_i = \langle Q_i, \Sigma, R^i_{\Sigma}, q_{i0}, P_i \rangle$, *where* $P_i : Q_i \to \mathcal{P}(AP)$, $i = 1, 2$, *if* $\mathbb{Q}_1 \approx_{se} \mathbb{Q}_2$ *then*

$$P^{\approx}(\mathcal{M}(\mathbb{Q}_1)) \cap (\mathcal{P}(AP) \times \Sigma)^{\omega} = P^{\approx}(\mathcal{M}(\mathbb{Q}_2)) \cap (\mathcal{P}(AP) \times \Sigma)^{\omega}$$

Proof: Let $\gamma \in P^{\approx}(\mathcal{M}(\mathbb{Q}_1)) \cap (\mathcal{P}(AP) \times \Sigma)^{\omega}$. Then there exists $\sigma_1 \in \mathcal{M}(\mathbb{Q}_1)$ such that $\gamma = P^{\approx}(\sigma_1)$. But $\mathcal{M}(\mathbb{Q}_1) \subseteq \mathcal{M}(\mathbb{Q}_1{}'_{se})$ so $\sigma_1 \in \mathcal{M}(\mathbb{Q}_1{}'_{se})$. By Lemma 4.5 there exists $\sigma'_2 \in \mathcal{M}(\mathbb{Q}_2{}'_{se})$ such that $P^{\sim}(\sigma_1) = P^{\sim}(\sigma'_2)$ and hence $P^{\approx}(\sigma_1) = P^{\approx}(\sigma'_2)$.

Now by Lemma 4.10 there exists $\sigma_2 \in \mathcal{M}(\mathbb{Q}_2)$ such that $P^{\approx}(\sigma'_2) = P^{\approx}(\sigma_2)$. This shows containment in the $\subseteq$ direction. Exchanging $\mathbb{Q}_1$ and $\mathbb{Q}_2$ in the above argument gives the opposite direction and hence the desired result. $\Box$

The above lemma states that weakly state-event equivalent systems produce identical infinite sequences of observations, though equivalent systems may disagree on sequences that produce finite observations as in the case of the systems in Figure 4.2. In RTTL and the simplified real-time state-event logic presented here, the fairness constraint $\Box\Diamond(\eta = tick)$ guarantees that the clock *tick*s infinitely often in all legal computations (i.e. all legal computations result in infinite sequences of observations). Thus if we can identify a subclass of formulas with truth values that are only dependent upon the observations a computation produces, the above lemma will allow us to use weak state-event equivalence to perform model reduction for those formulas. The systems in Figure 4.1 that provide a counterexample to the converse to Lemma 4.5 also provide a counterexample to the converse of Lemma 4.11.

4.3.2 Weak Satisfaction

As a first step towards obtaining a subclass of temporal formulas with truth values that are dependent upon the weakly observed computations, we will define weak satisfaction. While our main interest in introducing weak satisfaction is to obtain a subclass of formulas for weak state-event model reduction, weak satisfaction also provides a means of specifying the behavior of weakly projected computations and hence of specifying the behavior of the system at its outputs or interface with other modules.

Definition 4.12 *Given a SELTS $\mathbb{Q}$ and a temporal formula F, a computation $\sigma \in \mathcal{M}(\mathbb{Q})$ is said to* **weakly satisfy** *F, written $\sigma \models_{\approx} F$, iff $P^{\approx}(\sigma) \models F$. The SELTS $\mathbb{Q}$ weakly satisfies F, written $\mathbb{Q} \models_{\approx} F$, iff $P^{\approx}(\mathcal{M}(\mathbb{Q})) \models F$.*

Example 4.13 *For σ_1 and σ_2 as in Example 4.9 we have*

$$\begin{aligned} \sigma_1 &\models_{\approx} \eta = \alpha \wedge q = q_0 \\ \sigma_2 &\not\models_{\approx} \Box \bigcirc true \end{aligned}$$

In the case of σ_1 we are stating that the first observed action of the computation is an α transition that does not change the state output. In the case of σ_2 we are stating that the computation does not produce an infinite number of observations. A computation σ weakly satisfies $\Box \bigcirc true$ if the weak projection of the computation is an infinite sequence. Thus $\sigma \models_{\approx} \Box \bigcirc true$ becomes a concise way of saying that σ produces an infinite number of observations.

Theorem 4.14 *Given two SELTS, if $\mathbb{Q}_1 \approx_{se} \mathbb{Q}_2$ then for any temporal formula F we have $\mathbb{Q}_1 \models_{\approx} \neg(\Box \diamond \eta = tick) \vee F$ iff $\mathbb{Q}_2 \models_{\approx} \neg(\Box \diamond \eta = tick) \vee F$.*

Proof: Follows immediately from Lemma 4.11. $\Box$

The implication of the above theorem is that weak state-event equivalence can be used to perform model reduction for any real-time state-event temporal logic formula provided the satisfaction relation of interest is weak satisfaction. In general we are interested in performing model reduction for the standard satisfaction relation $\models$. In the following subsection Theorem 4.14 will be the key to developing model reduction results for the subclass of SESI formulas under the standard satisfaction relation.

4.3.3 State-Event Stuttering-Invariance and Model Reduction

We now consider those formulas with truth values that are robust with respect to unobservable τ transitions.

Definition 4.15 *Given a state-event temporal formula F over the set of atomic predicates AP, we say that F is* **State-Event Stuttering-Invariant** *(SESI) if for all SELTS $\mathbb{Q}$ with state output map $P : Q \to \mathcal{P}(AP)$, for all computations $\sigma \in \mathcal{M}(\mathbb{Q})$, the following equation holds:*

$$\sigma \models_{\approx} F \text{ iff } \sigma \models F \tag{4.1}$$

Equation (4.1) provides the link relating satisfaction to weak satisfaction that will be used to extend Theorem 4.14 to standard satisfaction of SESI formulas. Additionally, the existence of relatively complete proof systems, theorem provers and model-checkers for verifying $\models$ for variations of state-event temporal logics, together with (4.1), allow one to use existing tools to check $\models_{\approx}$ for SESI formulas. We now try to identify some SESI formulas before providing a formal statement that allows us to build more general SESI formulas.

Let F_s be a state formula. Then $\sigma \models_{\approx} F_s$ iff $\sigma \models F_s$ since $P^{\approx}$ does not affect the value of the initial state output. The case for general state-event formulas is complicated by references to the (next) transition variable η. Considering Example 4.9 we see that $\sigma_1 \models (\eta = \tau)$ but $\sigma_1 \models_{\approx} (\eta = \alpha)$ (i.e. the first transition of the computation is a τ transition but the first transition of the weakly observed computation is an α event). This difference results from the weak state-event projection operator deleting all τ transitions that do not cause any change in the state output. The formula $\diamond(\eta = \alpha)$ states that eventually an α transition occurs so clearly for any $\alpha \neq \tau$, $\sigma \models \diamond(\eta = \alpha)$ iff $\sigma \models_{\approx} \diamond(\eta = \alpha)$ since $P^{\approx}$ does not erase any non-τ transitions. With a similar argument one can also show that for $p \in AP$ and $\alpha \in \Sigma - \{\tau\}$, the formula $\Box[(\eta = \alpha) \rightarrow \bigcirc p]$, stating that in the state following an α transition p always holds, is SESI.

Such "base" formulas can be used to build up more complex temporal formulas as outlined in the following lemma.

Lemma 4.16 *Let σ be a computation and F, F_1, F_2 be SESI formulas. Then for all $\alpha \in \Sigma - \{\tau\}, l, u \in \mathbb{N}$ we have $\neg F$, $F_1 \vee F_2$, $F_1 \mathcal{U} F_2$ and $F_1 \mathcal{U}^{\alpha}_{[l,u]} F_2$ are all SESI formulas.*

Proof: The cases of $\neg F$ and $F_1 \vee F_2$ are immediate from the definitions so let us consider the case when $F := F_1 \mathcal{U} F_2$.
(only if) Assume $P^{\approx}(\sigma) \models F$. Then there exists $i \in \mathbb{N}$ such that $P^{\approx}(\sigma)^i \models F_2$ and for all $j = 0, 1, \ldots, i-1$, $P^{\approx}(\sigma)^j \models F_1$ by definition of $\mathcal{U}$.

Let $k_i \in \mathbb{N}$ be the smallest integer such that $P^{\approx}(\sigma^{k_i}) = P^{\approx}(\sigma)^i$. Therefore $P^{\approx}(\sigma^{k_i}) \models F_2$. But by our inductive assumption, $P^{\approx}(\sigma^{k_i}) \models F_2$ iff $\sigma^{k_i} \models F_2$.

Now, for all $l_i \in \{0, 1, \ldots, k_i - 1\}, P^{\approx}(\sigma^{l_i}) < P^{\approx}(\sigma^{k_i})$ (i.e. $P^{\approx}(\sigma^{l_i})$ is a strictly proper prefix of $P^{\approx}(\sigma^{k_i})$). Therefore there exists $j \in \{0, 1, \ldots, i-1\}$ such that $P^{\approx}(\sigma^{l_i}) = P^{\approx}(\sigma)^j$. But as noted above, $P^{\approx}(\sigma)^j \models F_1$ so $P^{\approx}(\sigma^{l_i}) \models F_1$ and hence $\sigma^{l_i} \models F_1$. By definition of $\mathcal{U}$, we have $\sigma \models F_1 \mathcal{U} F_2$ as required.

(if) The above proof can be reversed to obtain the if part of (4.1).

The case of $F := F_1 \mathcal{U}^{\alpha}_{[l,u]} F_2$ follows immediately from the $F := F_1 \mathcal{U} F_2$ case and the fact that $P^{\approx}$ does not erase any non-τ events. $\square$

From the above discussion we see that all *non-immediate formulas*, formulas composed solely of state predicates together with the $\vee, \wedge, \mathcal{U}, \mathcal{U}^{\alpha}_{[l,u]}$ operators (i.e. that do not contain the next operator $\bigcirc$ or next transition variable η) are SESI. Additionally, a formula of the form $\square \diamond (\eta = tick)$ is SESI since $\diamond(\eta = tick)$ is SESI and $\square F = \neg \diamond \neg F$. We can now extend Theorem 4.14 to provide results about $\models$ for formulas that belong to the subclass of SESI formulas.

Theorem 4.17 *Let F be an SESI formula. If $\mathbb{Q}_1, \mathbb{Q}_2$ are SELTS such that $\mathbb{Q}_1 \approx_{se} \mathbb{Q}_2$ then $\mathbb{Q}_1 \models \neg(\square \diamond \eta = tick) \vee F$ iff $\mathbb{Q}_2 \models \neg(\square \diamond \eta = tick) \vee F$.*

Proof: Assume $F, \mathbb{Q}_1, \mathbb{Q}_2$ are as in the theorem statement. Also assume that $\mathbb{Q}_1 \models \neg(\square \diamond \eta = tick) \vee F$. Using the fact that $\square \diamond (\eta = tick)$ and F are SESI together with Lemma 4.16, we know that the formula $\neg(\square \diamond (\eta = tick)) \vee F$ must be SESI.

Therefore, by the definition of SESI, $\mathbb{Q}_1 \models_{\approx} \neg(\square \diamond \eta = tick) \vee F$. But $\mathbb{Q}_1 \approx_{se} \mathbb{Q}_2$, so applying Theorem 4.14, we have $\mathbb{Q}_2 \models_{\approx} \neg(\square \diamond \eta = tick) \vee F$. We can then apply the opposite direction of the SESI definition to obtain $\mathbb{Q}_2 \models \neg(\square \diamond \eta = tick) \vee F$.

Switching $\mathbb{Q}_1$ and $\mathbb{Q}_2$ in the above argument provides the desired result. $\square$

Recalling from Section 3.2 that $\mathbb{Q} \approx_{se} \mathbb{Q} /\!\!/ \theta_w$, where $\mathbb{Q} /\!\!/ \theta_w$ is the weak state-event quotient system of $\mathbb{Q}$, Theorem 4.17 allows us to model-check SESI formulas on a system's quotient system and infer the result for the original system. Additionally, Lemma 3.31 guarantees that our model reduction technique is compositionally consistent. This allows one to avoid computing massive synchronous products before performing model reduction, by first doing model reduction on the component subsystems and *then* forming their synchronous product. This ability is significant since synchronous products typically grow as the product of the subsystem's state spaces. Once it is constructed we can take what should be the significantly smaller synchronous product of the quotient systems and compute its quotient system to further reduce our model.

4.4 Model Reduction of TTM Modules

Theorem 3.31 and Theorem 4.17 allow us to perform compositional model reduction for real-time systems modeled as interacting systems of SELTS. Typically though, a systems designer will want to work within a more expressive framework, such as TTMs, that provides a more compact representation of the system. In this section we adapt the results of the previous two chapters to the TTM setting. This is done by considering systems composed of interacting "TTM modules" – TTMs with the property that parallel composition at the TTM level can be modeled by state-event synchronous composition at the SELTS level. To motivate the introduction of TTM modules, we first provide an example of TTM parallel composition that does not correspond to state-event synchronous composition at the SELTS level.

Example 4.18 *Consider the following two simple TTMs.*

$M_1 := \langle \{y, z\}, y = z = 0, \mathcal{T}_1 \rangle$, *where* $\mathcal{T}_1 = \{\alpha := (y = z, [y : y \oplus_2 1], 1, 1)\}$ *and,*

$M_2 := \langle \{y, z\}, y = z = 0, \mathcal{T}_2 \rangle$, *where* $\mathcal{T}_2 = \{\beta := (y \neq z, [z : z \oplus_2 1], 1, 1)\}$.

In the above transitions' operation functions, $\oplus_2$ denotes addition mod 2. Thus transition α of M_1 will toggle the value of y between 0 and 1 when $y = z$ for one tick of the global clock. Transition β of M_2 performs a similar function on z when $y \neq z$. The TTMs' parallel composition is given by

$$\begin{aligned} M_1 \| M_2 &:= \langle \{y, z\}, y = z = 0, \mathcal{T}_1 \| \mathcal{T}_2 \rangle \\ &= \langle \{y, z\}, y = z = 0, \mathcal{T}_1 \cup \mathcal{T}_2 \rangle \end{aligned}$$

The SELTS generated by $M_1 \| M_2$ is shown in Figure 4.3. In the composite system first α and then β alternate with *tick* transitions. M_1 reacts to changes to its "input" z and produces "output" y while M_2 reacts to the input value of y and produces output z.

Now let us consider the SELTS generated by the individual TTM component systems before their composition (see Figure 4.4). M_1's α transition does not affect the value of z. With no other transitions to change z from its initial value, after an

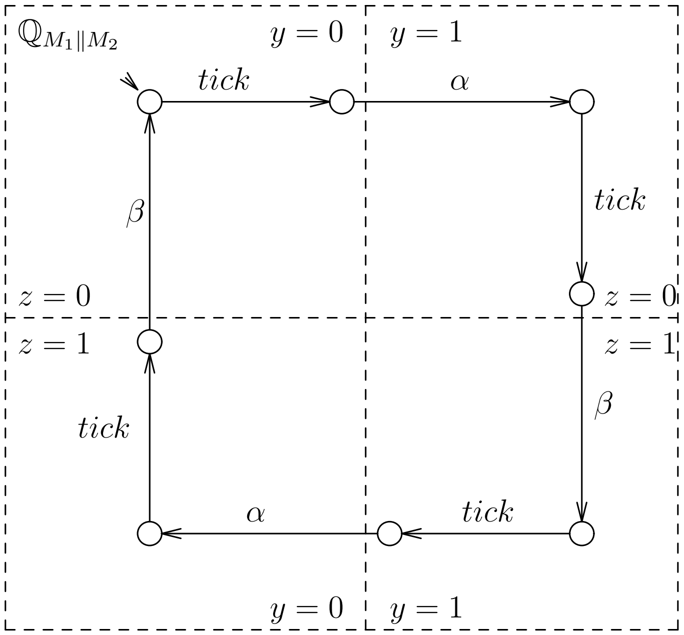


Figure 4.3: SELTS for $M_1\|M_2$.

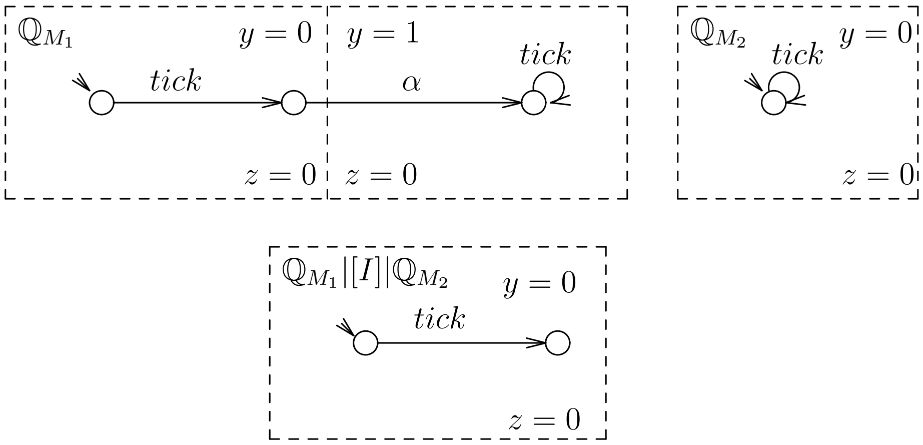
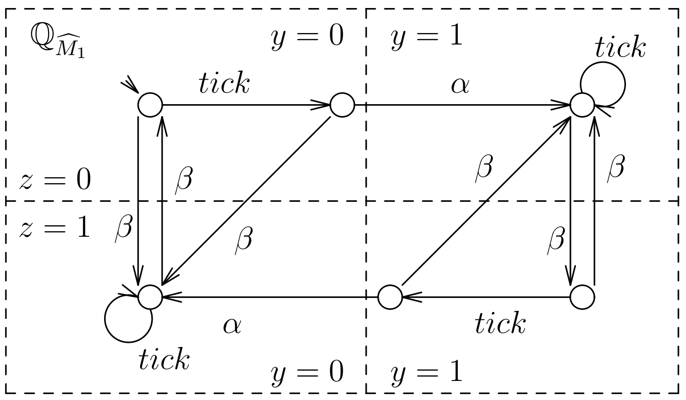


Figure 4.4: SELTS generated by M_1 and M_2 and their composition synchronizing on *tick* and the values of y, z.

initial *tick*, α changes the value of y to 1 and then only *tick* transitions are possible in M_1. This is reflected in $\mathbb{Q}_{M_1}$ as the $tick, \alpha$ sequence ending in a selflooped *tick* state. Considering M_2 in isolation, the transition β is not initially enabled and there are no other transitions of M_2 that could change the value of y or z to enable β. As a result, $\mathbb{Q}_{M_2}$ is simply a *tick* selflooped at an initial state with state output $(y, z) = (0, 0)$. $\mathbb{Q}_{M_1}$ and $\mathbb{Q}_{M_2}$ do not have any states with state output $(y, z) = (1, 1)$ so clearly for any compatible SELTS interface I that synchronizes on the values of y and z, $\mathbb{Q}_{M_1}|[I]|\mathbb{Q}_{M_2}$ will not produce the same computations or even observed computations as $\mathbb{Q}_{M_1\|M_2}$.

This result is not particularly surprising since the TTM parallel composition operator does not place any restrictions on how one TTM may access another TTM's variables other than the restrictions upon transitions with shared labels. These restrictions in turn can allow a TTM to prevent any transition in another TTM, simply by having a transition with the same label and a false enablement condition. In the case of M_1 and M_2, these two TTMs were designed with specific interfaces in mind. In order to produce interesting behavior, M_1 expects changes in the value of z. To avoid transition label conflicts we could associate a transition label (or labels) with the transitions which M_1 expects to change z, for example (β, z). By adding a nondeterministic transition $\beta := (true, [z : 0; z : 1], 0, \infty)$ to M_1, we provide a well defined interface for M_2 without restricting how M_2 may alter z. The semicolon occurring in the operation function of this particular β transition indicates that when β occurs, a choice is made between setting $z = 0$ and $z = 1$ in the next state (see Section 2.2.1 p. 24). Similarly we may add $\alpha := (true, [y : 0; y : 1], 0, \infty)$ to the transition set of M_2 to provide an interface for our new M_1. Denote these augmented TTMs by $\widehat{M_1}$ and $\widehat{M_2}$ respectively. Their generated SELTS are shown in Figure 4.5. For the sake of legibility, selflooped β and α transitions have been omitted from all of the states of $\mathbb{Q}_{\widehat{M_1}}$ and $\mathbb{Q}_{\widehat{M_2}}$ respectively. In the TTM parallel composition $\widehat{M_1}\|\widehat{M_2}$, the shared transition α would be given by

$$\alpha \ := \ (y = z \wedge true, h'_\alpha, \max(0, 1), \min(1, \infty))$$



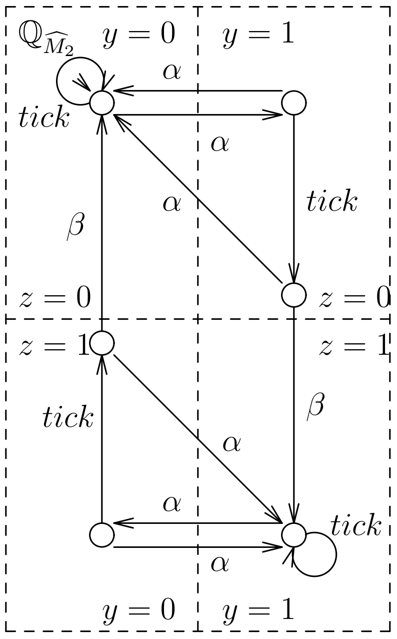


Figure 4.5: SELTS for augmented TTMs $\widehat{M_1}, \widehat{M_2}$.

$$= \quad (y = z, [y : y \oplus_2 1], 1, 1)$$

From Definition 2.7, h'_α is the operation function that results from making transitions only to those new state assignments that are possible in both component systems. Since α can make arbitrary changes to y in M_2, the composite transition results in changes to y that are identical to those produced by M_1. Similarly $\beta := (y \neq z, [z : z \oplus_2 1], 1, 1)$ in $\widehat{M_1} \| \widehat{M_2}$. Therefore we conclude that $\widehat{M_1} \| \widehat{M_2} = M_1 \| M_2$.

We can now take $I := (\{\alpha, \beta, tick\}, id_{\mathcal{Q}_{\{y,z\}}}, id_{\mathcal{Q}_{\{y,z\}}})$ to be our SELTS interface. This choice of I forces synchronization on α, β, and $tick$ events, and the value of the shared variables which is the state output of both systems. Applying the definition of state-event synchronous composition for $\mathbb{Q}_{\widehat{M_1}} |[I]| \mathbb{Q}_{\widehat{M_2}}$ we obtain a SELTS that is isomorphic to $\mathbb{Q}_{M_1 \| M_2}$. Thus we see that when TTMs are defined with "compatible interfaces", the composition of their generated SELTS indeed produces identical strongly observed computations to the generated SELTS of their composition .

This property is significant because in the special case of interface compatible TTMs, it reduces composition at the TTM level to composition at the SELTS level. Thus anything that we can say about the compositional properties of SELTS can be applied to these "interface compatible" TTMs. We call these TTMs with interfaces "TTM modules".

In formally defining modules we assume that system components are designed to have a particular interface with other components. This is specified in terms of input and output variables paired with labels of transitions (events) to be executed synchronously with other system modules when changes are made to the associated input or output variable.

Recalling from Section 2.2.2 that for $\mathcal{T}$, a set of TTM transitions, $\Sigma(\mathcal{T})$ denotes the set of transition labels used in $\mathcal{T}$, we are now ready to define TTM modules.

Definition 4.19 *A* TTM module *is defined to be a TTM-interface pair* $m := (M, \mathcal{I})$ *where* $M := \langle \mathcal{V}, \theta, \mathcal{T} \rangle$ *is a TTM and* $\mathcal{I}$ *is an* interface for M *such that*

$$\mathcal{I} := (\mathcal{V}_{in}, \Sigma_{in}, R_{in}, \mathcal{V}_{out}, \Sigma_{out}, R_{out})$$

The components of $\mathcal{I}$ *are:*

- $\mathcal{V}_{in} \subseteq \mathcal{V}$ *is a set of input variables*
- Σ_{in} *is a set of input transition labels such that* $\Sigma_{in} \cap \Sigma(\mathcal{T}) = \emptyset$
- $R_{in} \subseteq \Sigma_{in} \times \mathcal{V}_{in}$ *is the module's* input relation*, a relation between input transition labels and variables*
- $\mathcal{V}_{out} \subseteq \mathcal{V}$ *is a set of output variables*
- $\Sigma_{out} \subseteq \Sigma(\mathcal{T})$ *is a set of output transition labels. It includes all the transitions of* M *that modify one or more output variables.*
- $R_{out} \subseteq \Sigma_{out} \times \mathcal{V}_{out}$ *is the module's* output relation*, a relation that contains a* (α, v) *pair for every output transition* α *that modifies an output variable* v.

In the above definition, M is a TTM partially specifying the module's behavior. It does not specify the behavior of input transitions or input variables. The above definition requires that transition labels occurring in Σ_{in} are not already used by the TTM M and hence will not have any restrictions placed upon them by M. Such "input transitions" will only be allowed to affect the behavior of M through modifying the value of the input variables they are associated with in R_{in}. The constraint placed upon the module's output transition label set Σ_{out} states that if v is an output variable then the transition label of any α transition belonging to M that affects v must appear in Σ_{out}. The transition label/variable pair (α, v) must then appear in the output relation R_{out}. More formally, for $\alpha := (e, h, l, u) \in \mathcal{T}$, if there exist $v \in \mathcal{V}_{out}$ and state assignments $q, q' \in \mathcal{Q}_{\mathcal{V}}$ such that $q' \in h(q)$ (q' is an α successor of q) and $q'(v) \neq q(v)$ then $\alpha \in \Sigma_{out}$ and $(\alpha, v) \in R_{out}$.

Let us go back and redefine the TTMs M_1 and M_2 of Example 4.18 as TTM modules. The systems can be described by modules $m_i := (M_i, \mathcal{I}_i), i = 1, 2$ where

$$\begin{aligned}
\mathcal{I} &:= (\Sigma_{in}, \mathcal{V}_{in}, R_{in}, \Sigma_{out}, \mathcal{V}_{out}, R_{out}) \\
\mathcal{I}_1 &:= (\{\beta\}, \{z\}, \{(\beta, z)\}, \{\alpha\}, \{y\}, \{(\alpha, y)\}) \\
\mathcal{I}_2 &:= (\{\alpha\}, \{y\}, \{(\alpha, y)\}, \{\beta\}, \{z\}, \{(\beta, z)\})
\end{aligned}$$

Note that the interfaces of m_1 and m_2 appear to be "compatible" since one system's input relation equals the other's output relation. We will have more to say on the matter of TTM module compatibility following the definition of the SELTS generated by a module.

In Example 4.18 the TTMs M_1 and M_2 were augmented by adding nondeterministic input transitions that could make arbitrary changes to each ystem's input variables, thereby representing all possible actions that other modules could perform at the system's interface. As a consequence, these augmented TTMs $\widehat{M_1}, \widehat{M_2}$ generated SELTS that result in a SELTS composition that was isomorphic to the SELTS generated by the TTM parallel composition of the augmented TTMs. We will generalize this result to the composition of any two "compatible" TTM modules. We begin by defining the augmented TTM of a module. In the following definition we use the fact that for any function $f : A \to B$, $\ker(f)$, the equivalence kernel of f, defines a mapping $\ker(f) : A \to \mathcal{P}(A)$, $a \mapsto a/\ker(f)$. Here $a/\ker(f) := \{a' \in A : f(a') = f(a)\}$ is the $\ker(f)$ equivalence class of a.

Definition 4.20 *Consider m, a TTM module as in Definition 4.19. For $\alpha \in \Sigma_{in} \cup \Sigma_{out}$ let $\mathcal{V}_\alpha := \{v \in \mathcal{V} : (\alpha, v) \in R_{in} \cup R_{out}\}$. Then the* augmented TTM of *m, denoted $\widehat{M}$, is the TTM $\widehat{M} := \langle \mathcal{V}, \Theta, \mathcal{T}' \rangle$ where*

$$\mathcal{T}' := \mathcal{T} \cup \{\alpha := (true, h_\alpha, 0, \infty) : \alpha \in \Sigma_{in} \text{ and } h_\alpha = \ker(P_{\mathcal{Q}_{\mathcal{V}-\mathcal{V}_\alpha}})\}$$

Here $P_{\mathcal{Q}_{\mathcal{V}-\mathcal{V}_\alpha}} : \mathcal{Q}_\mathcal{V} \to \mathcal{Q}_{\mathcal{V}-\mathcal{V}_\alpha}$ is the canonical projection from state assignments over $\mathcal{V}$ onto the state assignments over $\mathcal{V} - \mathcal{V}_\alpha$.

The variable set and initial condition of $\widehat{M}$ are identical to those of m's TTM M, while $\mathcal{T}'$ is obtained from $\mathcal{T}$, the transition set of M, by the addition of *input* transitions. For each $\alpha \in \Sigma_{in}$, α's operation function h_α maps the current state assignment to the set of all state assignments that differ only in the value of variables $v \in \mathcal{V}_{in}$ such that $(\alpha, v) \in R_{in}$. More formally, for $\alpha \in \Sigma_{in}$, $h_\alpha : \mathcal{Q} \to \mathcal{P}(\mathcal{Q})$ such that $q \mapsto q/\ker(P_{\mathcal{Q}_{\mathcal{V}-\mathcal{V}_\alpha}})$. But

$$\begin{aligned} q/\ker(P_{\mathcal{Q}_{\mathcal{V}-\mathcal{V}_\alpha}}) &= \{q' \in \mathcal{Q} : P_{\mathcal{Q}_{\mathcal{V}-\mathcal{V}_\alpha}}(q') = P_{\mathcal{Q}_{\mathcal{V}-\mathcal{V}_\alpha}}(q)\} \\ &= \{q' \in \mathcal{Q} : \forall v \in \mathcal{V} (\alpha, v) \notin R_{in} \text{ implies } q'(v) = q(v)\} \end{aligned}$$

Thus variables in $\mathcal{V} - \mathcal{V}_\alpha$ are unchanged by the occurrence of an α transition in $\widehat{M}$.

Now we can define the SELTS generated by a module to be a relabeling of the SELTS generated by the module's augmented TTM. We relabel the SELTS events by replacing all transitions that are not *tick*, input, or output transitions by τ. The states are relabeled by projecting the state assignments (the state outputs for the augmented TTM's SELTS) onto the state assignments over the module's input and output variables. This SELTS relabeling allows us to hide internal transitions and variables while retaining the input/output behavior of the system.

Definition 4.21 *Let m be a TTM module as defined in Definition 4.19 Define $r :=$*

(r_Σ, r_P) *to be the SELTS relabeling such that*

$$r_\Sigma(\alpha) = \begin{cases} \alpha, & \textit{if } \alpha \in \Sigma_{in} \cup \Sigma_{out} \cup \{tick\} \\ \tau, & \textit{otherwise} \end{cases}$$

and $r_P := P_{\mathcal{Q}_{\mathcal{V}_{in} \cup \mathcal{V}_{out}}}$ *is the canonical projection from state assignments over* $\mathcal{V}$ *to state assignments over* $\mathcal{V}_{in} \cup \mathcal{V}_{out}$*. Then the* SELTS generated by the TTM module m *is defined to be* $\mathbb{Q}_m := r(\mathbb{Q}_{\widehat{M}})$.

In Definition 4.21 the operation functions of the augmenting "input" transitions of $\widehat{M}$ map the current state assignment to all possible variations of assignments to the input variables associated with the transition label α. Since we are restricting ourselves to finite state SELTS with finite event sets, we must restrict ourselves to input variables with finite range spaces. This is adequate to handle the industrial example of Chapter 5. While it should be possible to extend the theory to handle variables with infinite range spaces, that is beyond the scope of this thesis.

Having defined the SELTS generated by a module, we can immediately apply all the formal methods developed for transition systems to TTM modules. For example, if EQ is an SELTS equivalence relation then we say that *module* m_1 *is* EQ *equivalent to module* m_2, written m_1 EQ m_2 iff $\mathcal{I}_1 = \mathcal{I}_2$ and $\mathbb{Q}_{m_1}$ EQ $\mathbb{Q}_{m_2}$. Similarly the definition of satisfaction of a temporal logic formula $\models F$ for SELTS found in Section 4.1.2 can be applied to a TTM module m by saying that m *satisfies the temporal logic formula* F iff the SELTS generated by m satisfies F (ie. $m \models F$ iff $\mathbb{Q}_m \models F$).

We now generalize the "interface compatibility" of the modules from Example 4.18. For the formal definition of interface compatibility we will be dealing with a pair of modules m_1, m_2 and referencing specific elements of these modules' interfaces. As a notational convenience we will parameterize the interface specification of the TTM module definition by the module name.

Definition 4.22 *Let TTM modules* $m_1 := (M_1, \mathcal{I}(m_1))$ *and* $m_2 := (M_2, \mathcal{I}(m_2))$ *be* *given. For $i = 1, 2$ $M_i := \langle \mathcal{V}_i, \Theta_i, \mathcal{T}_i \rangle$, write and*

$$\mathcal{I}(m_i) := (\Sigma_{in}(m_i), \mathcal{V}_{in}(m_i), R_{in}(m_i), \Sigma_{out}(m_i), \mathcal{V}_{out}(m_i), R_{out}(m_i))$$

Then we say that m_1 and m_2 are interface compatible modules *iff all the following conditions hold:*

(i) $\Sigma(\mathcal{T}_1) \cap \Sigma(\mathcal{T}_2) = \{tick\}$

(ii) for $i = 1, 2$, $\mathcal{V}_1 \cap \mathcal{V}_2 \subseteq \mathcal{V}_{in}(m_i) \cup \mathcal{V}_{out}(m_i)$

(iii) for $i \neq j$, $R_{out}(m_i) \cap [\Sigma_{out}(m_i) \times (\mathcal{V}_1 \cap \mathcal{V}_2)] \subseteq R_{in}(m_j)$

(iv) for $i \neq j$, $R_{in}(m_i) \cap [\Sigma_{in}(m_i) \times (\mathcal{V}_1 \cap \mathcal{V}_2)] \subseteq R_{in}(m_j) \cup R_{out}(m_j)$

Condition (i) states that for interface compatible modules m_1 and m_2, their TTMs $M(m_1)$ and $M(m_2)$ only share the *tick* transition. Since $\Sigma_{out}(m_i) \subseteq \mathcal{T}_i$, this condition implies that $\Sigma_{out}(m_1) \cap \Sigma_{out}(m_2) = \emptyset$ so there can be no conflicts with output transition labels. By (ii) shared variables are required to appear in $\mathcal{V}_{in}(m_i)$ and/or $\mathcal{V}_{out}(m_i)$ of both modules to insure that interacting TTMs do not have naming conflicts of internal variables. Condition (iii) requires that all output transition labels associated with a shared variable in one system must be paired with the shared variable in the other system's $R_{in}(m_j)$ relation. Finally (iv) demands that if v is a shared variable and α is an input transition label for v in m_i, then in m_j α is either an input transition label for v or an output transition label. In this way input to a shared variable is expected to come from an outside source to both systems via the same transition label, or m_j is supplying the input to m_i via its output transition label α.

Note that the above conditions do not rule out the possibility of $v \in \mathcal{V}_1 \cap \mathcal{V}_2$ being both an input and an output to both systems. Conditions (iii) and (iv) force any event label one system uses for output of v to be used for input of v in the other system. The other system can then use a different transition label for its output of v which then must similarly be used for input of v in the first module.

After defining interface compatibility we can now define the composition of modules. The definition below relies on the intuition that while only one system can output to a shared variable at a given time, many systems can simultaneously receive that output at their inputs (via a broadcast mechanism). Hence the R_{out} relation of the composite system is simply the union of the $R_{out}(m_i)$ relations of the component systems. Even if a transition label/variable pair that is the output of one system is the input to the other system, the label/variable pair is removed from the composite system's R_{in} relation. It remains an output pair of the composite system to enable other systems to receive changes to the variable in further synchronous products.

Definition 4.23 *Given interface compatible modules m_1, m_2, as in Definition 4.22, the* synchronous composition of m_1 and m_2 *is defined to be the TTM module*

$$m_1 \| m_2 := (M_1 \| M_2, \mathcal{I}(m_1 \| m_2))$$

where the components of $\mathcal{I}$ are:

- $\Sigma_{in}(m_1\|m_2) = (\Sigma_{in}(m_1) \cup \Sigma_{in}(m_2)) \setminus (\Sigma_{out}(m_1) \cup \Sigma_{out}(m_2))$
- $\mathcal{V}_{in}(m_1\|m_2) = \{v \in \mathcal{V}_{in}(m_1) \cup \mathcal{V}_{in}(m_2) : \exists \alpha \in \Sigma_{in}(m_1\|m_2), (\alpha, v) \in R_{in}(m_1\|m_2)\}$
- $R_{in}(m_1\|m_2) = (R_{in}(m_1) \cup R_{in}(m_2)) \setminus (R_{out}(m_1) \cup R_{out}(m_2))$
- $\Sigma_{out}(m_1\|m_2) = \Sigma_{out}(m_1) \cup \Sigma_{out}(m_2)$
- $\mathcal{V}_{out}(m_1\|m_2) = \mathcal{V}_{out}(m_1) \cup \mathcal{V}_{out}(m_2)$
- $R_{out}(m_1\|m_2) = R_{out}(m_1) \cup R_{out}(m_2)$

In the above definition we can not merely set $\mathcal{V}_{in}(m_1\|m_2) = (\mathcal{V}_{out}(m_1) \cup \mathcal{V}_{out}(m_2)) \setminus (\mathcal{V}_{out}(m_1) \cup \mathcal{V}_{out}(m_2))$ as one might expect given the formulas for $\Sigma_{in}(m_1\|m_2)$ and $R_{in}(m_1\|m_2)$. To avoid output transition label conflicts in other systems, a given input transition label α (and hence transition label/variable input pair (α, v)) can only be used by one module with the output transition label α. This restriction does not prevent another module from writing to v via another input transition label as a given variable may have multiple input transitions.

In the case of the modules associated with the systems of Example 4.18, $m_1 \| m_2 := (M_1 \| M_2, \mathcal{I}(m_1 \| m_2))$ where

$$\mathcal{I}(m_1 \| m_2) := (\emptyset, \emptyset, \emptyset, \{\alpha, \beta\}, \{y, z\}, \{(\alpha, y), (\beta, z)\})$$

The absence of any input transition labels, variables and input pairs indicates that $M_1 \| M_2$ is a *closed system* that does not require input from an external source. Hence $M_1 \| M_2$ completely specifies the behavior of the output variables y, z.

The following lemma will help us to prove that for interface compatible modules, the SELTS generated by the composition of the modules and the SELTS resulting from the composition of the SELTS generated by each module, differ only in the labeling of their underlying state sets. Hence they produce identical sequences of state outputs and connecting events. Thus our ultimate goal is to show that for an appropriate SELTS interface I, $\mathbb{Q}_{m_1 \| m_2}$ is isomorphic to $\mathbb{Q}_{m_1} |[I]| \mathbb{Q}_{m_2}$. We begin by showing that $\mathbb{Q}_{\widehat{M_1 \| M_2}}$ is isomorphic to $\mathbb{Q}_{\widehat{M}_1} |[I]| \mathbb{Q}_{\widehat{M}_2}$. The desired result then follows by applying relabelings to these systems to produce $\mathbb{Q}_{m_1 \| m_2}$ and $\mathbb{Q}_{m_1} |[I]| \mathbb{Q}_{m_2}$.

Lemma 4.24 *Let m_1, m_2 be two interface compatible TTM modules as in Definition 4.22, and $I := (\Sigma_I, P_{\mathcal{Q}_{\mathcal{V}_1 \cap \mathcal{V}_2}}, P_{\mathcal{Q}_{\mathcal{V}_1 \cap \mathcal{V}_2}})$ be the SELTS interface, where*

$$\Sigma_I := [(\Sigma_{in}(m_1) \cup \Sigma_{out}(m_1)) \cap (\Sigma_{in}(m_2) \cup \Sigma_{out}(m_2))] \cup \{tick\}$$

and $P_{\mathcal{Q}_{\mathcal{V}_1 \cap \mathcal{V}_2}}$ is the canonical projection onto the state assignments over $\mathcal{V}_1 \cap \mathcal{V}_2$. Then the reachable parts of $\mathbb{Q}_{\widehat{M_1 \| M_2}}$ and $\mathbb{Q}_{\widehat{M}_1} |[I]| \mathbb{Q}_{\widehat{M}_2}$ are isomorphic.

Proof: As a notational convenience, let

$$\begin{array}{rcl} \mathbb{Q}_L := \mathbb{Q}_{\widehat{M_1 \| M_2}} & = & \langle Q_L, \Sigma_L, R_{\Sigma_L}, q_{L0}, P_L \rangle \\ \mathbb{Q}_R := \mathbb{Q}_{\widehat{M_1}} |[I]| \mathbb{Q}_{\widehat{M_2}} & = & \langle Q_R, \Sigma_R, R_{\Sigma_R}, q_{R0}, P_R \rangle \\ \mathbb{Q}_1 := \mathbb{Q}_{\widehat{M_1}} & = & \langle Q_1, \Sigma_1, R_{\Sigma_1}, q_{10}, P_1 \rangle \\ \mathbb{Q}_2 := \mathbb{Q}_{\widehat{M_2}} & = & \langle Q_2, \Sigma_2, R_{\Sigma_2}, q_{20}, P_2 \rangle \end{array}$$

First we establish that $\Sigma_L = \Sigma_R$. From Definitions 4.21 and 4.23 it follows that

$$\begin{array}{rcl} \Sigma_L & = & \Sigma(\mathcal{T}_1 \| \mathcal{T}_2) \cup \Sigma_{in}(m_1 \| m_2) \\ & = & (\Sigma(\mathcal{T}_1) \cup \Sigma(\mathcal{T}_2)) \cup [(\Sigma_{in}(m_1) \cup \Sigma_{in}(m_2)) \setminus (\Sigma_{out}(m_1) \cup \Sigma_{out}(m_2))] \\ & & \text{By def. 2.7 and Def. 4.23} \\ & = & \Sigma(\mathcal{T}_1) \cup \Sigma_{in}(m_1) \cup (\Sigma(\mathcal{T}_2) \cup \Sigma_{in}(m_2)) \\ & & \text{Since } \Sigma_{out}(m_i) \subseteq \Sigma(\mathcal{T}_i) \text{ for } i = 1, 2 \\ & = & \Sigma_1 \cup \Sigma_2 \text{ by def. 4.21} \\ & = & \Sigma_R \text{ by def. of } |[I]| \end{array}$$

We now examine the underlying state sets of $\mathbb{Q}_L$ and $\mathbb{Q}_R$. Recall from Section 2.2.3, p. 30, that in order to obtain a finite state representation of a TTM $M := \langle \mathcal{V}, \Theta, \mathcal{T} \rangle$, for $\alpha := (e, h, l, u) \in \mathcal{T}$, the range space of c_α, the counter variable associated with α, is defined to be:

$$Range_M(c_\alpha) := \begin{cases} \{n \in \mathbb{N} : n < l\} \cup \{\omega\}, & \text{if } u = \infty \\ \{n \in \mathbb{N} : n \leq u\}, & \text{otherwise} \end{cases}$$

Q_L, the state set of $\mathbb{Q}_L$, is the set of extended state assignments for $\widehat{M_1 \| M_2}$ (ie. the cross product of the state assignments over the variables of the TTM and the range spaces of the TTM transition counter variables). For $\alpha \in \Sigma_{in}(m_1 \| m_2)$, in $M_L := \widehat{M_1 \| M_2}$ we have the "input transition" $\alpha := (true, h_\alpha, 0, \infty)$. Therefore in this case $Range_{M_L}(c_\alpha) = \{\omega\}$. Thus,

$$\begin{aligned}
Q_L &= \mathcal{Q}_{\mathcal{V}_1 \cup \mathcal{V}_2} \times \prod_{\alpha \in \Sigma_L} Range_{M_L}(c_\alpha) \\
&= \mathcal{Q}_{\mathcal{V}_1 \cup \mathcal{V}_2} \times \prod_{\alpha \in \Sigma(\mathcal{T}_1) \cup \Sigma(\mathcal{T}_2)} Range_{M_L}(c_\alpha) \times \prod_{\alpha \in \Sigma_{in}(m_1 \| m_2)} Range_{M_L}(c_\alpha) \\
&= \mathcal{Q}_{\mathcal{V}_1 \cup \mathcal{V}_2} \times \prod_{\alpha \in \Sigma(\mathcal{T}_1) \cup \Sigma(\mathcal{T}_2)} Range_{M_L}(c_\alpha) \times \prod_{\alpha \in \Sigma_{in}(m_1 \| m_2)} \{\omega\} \\
&= \mathcal{Q}_{\mathcal{V}_1 \cup \mathcal{V}_2} \times \prod_{\alpha \in \Sigma(\mathcal{T}_1) \cup \Sigma(\mathcal{T}_2)} Range_{M_L}(c_\alpha) \times \{\omega\}^{|\Sigma_{in}(m_1 \| m_2)|}
\end{aligned}$$

From the definition of $|[I]|$ we know that the state set of $\mathbb{Q}_R$ is the cross product of the state sets of $\mathbb{Q}_{\widehat{M}_1}$ and $\mathbb{Q}_{\widehat{M}_2}$. The state set Q_i is the set of extended state assignments of $\widehat{M}_i$ for $i = 1, 2$. We now apply the method used in the development of Q_L to obtain Q_R.

$$\begin{aligned}
Q_R &= \mathcal{Q}_{\mathcal{V}_1} \times \prod_{\alpha \in \Sigma(\mathcal{T}_1)} Range_{\widehat{M}_1}(c_\alpha) \times \prod_{\alpha \in \Sigma_{in}(m_1)} Range_{\widehat{M}_1}(c_\alpha) \\
&\quad \times \mathcal{Q}_{\mathcal{V}_2} \times \prod_{\alpha \in \Sigma(\mathcal{T}_2)} Range_{\widehat{M}_2}(c_\alpha) \times \prod_{\alpha \in \Sigma_{in}(m_2)} Range_{\widehat{M}_2}(c_\alpha) \\
&= \mathcal{Q}_{\mathcal{V}_1} \times \prod_{\alpha \in \Sigma(\mathcal{T}_1)} Range_{\widehat{M}_1}(c_\alpha) \times \{\omega\}^{|\Sigma_{in}(m_1)|} \\
&\quad \times \mathcal{Q}_{\mathcal{V}_2} \times \prod_{\alpha \in \Sigma(\mathcal{T}_2)} Range_{\widehat{M}_2}(c_\alpha) \times \{\omega\}^{|\Sigma_{in}(m_2)|}
\end{aligned}$$

By the definition of interface compatible modules (def. 4.22) it follows that $\widehat{M}_1$ and $\widehat{M}_2$ only share the *tick* transition label. Thus by the TTM parallel composition definition, (def. 2.7) if $\alpha_1 := (e, h, l, u) \in \mathcal{T}_1$, then

$$\begin{aligned}
\alpha_1 &:= (e, h', l, u) \in \mathcal{T}_1 \| \mathcal{T}_2 \\
h' &:= h \otimes id_{\mathcal{Q}_{\mathcal{V}_2 \setminus \mathcal{V}_1}}
\end{aligned} \qquad (\ddagger)$$

Thus the only difference in the transitions is that h' has been extended appropriately to $\mathcal{V}_1 \cup \mathcal{V}_2$. But then $Range_{\widehat{M}_1}(\alpha_1) = Range_{M_L}(\alpha_1)$. By identical reasoning for $\alpha_2 := (e, h, l, u) \in \mathcal{T}_2$, we also know that $Range_{\widehat{M}_2}(\alpha_2) = Range_{M_L}(\alpha_2)$

From the above discussion, we see that a typical $q_L \in Q_L$ is

$$q_L = (q, c_{\alpha_{11}}, \ldots, c_{\alpha_{1m}}, c_{\alpha_{21}}, \ldots, c_{\alpha_{2n}}, \omega, \ldots, \omega)$$

where $\alpha_{11}, \ldots, \alpha_{1m} \in \mathcal{T}_1$ and $\alpha_{21}, \ldots, \alpha_{2m} \in \mathcal{T}_2$, while $q \in \mathcal{Q}_{\mathcal{V}_1 \cup \mathcal{V}_2}$. Thus we can define an embedding of Q_L into Q_R, denoted $f : Q_L \to Q_R$, by

$$f(q_L) = (P_{\mathcal{Q}_{\mathcal{V}_1}}(q), c_{\alpha_{11}}, \ldots, c_{\alpha_{1m}}, \omega, \ldots, \omega, P_{\mathcal{Q}_{\mathcal{V}_2}}(q), c_{\alpha_{21}}, \ldots, c_{\alpha_{2n}}, \omega, \ldots, \omega)$$

Clearly f is one to one. It is not onto Q_R if $\mathcal{V}_1 \cap \mathcal{V}_2 \neq \emptyset$ since for $v \in \mathcal{V}_1 \cap \mathcal{V}_2$ there are elements $(q_1, q_2) \in \mathcal{Q}_1 \times \mathcal{Q}_2$ such that $q_1(v) \neq q_2(v)$. But, by the definition of synchronous composition, these states are not reachable in $\mathbb{Q}_R$ because of their conflicting state output values. Therefore while f is not onto, the set of reachable states of $\mathbb{Q}_R$ is a subset of $f_*(Q_L)$.

For any state such as q_L, the state output of q_L in $\mathbb{Q}_L$ is $P_L(q_L) = q$. In $\mathbb{Q}_R$, $P_R(f(q_L)) = P_{\mathcal{Q}_{\mathcal{V}_1}}(q) \cup P_{\mathcal{Q}_{\mathcal{V}_1}}(q) = q$. Thus in order to show that f defines an isomorphism of the reachable parts of $\mathbb{Q}_L$ and $\mathbb{Q}_R$, all that remains is to show that for all $\alpha \in \Sigma_L$ and $q_L, q_L' \in Q_L$, $q_L \xrightarrow{\alpha} q_L'$ in $\mathbb{Q}_L$ iff $f(q_L) \xrightarrow{\alpha} f(q_L')$ in $\mathbb{Q}_R$.

This follows immediately from form the following two facts: First (‡) above guarantees that any non-input transition has time bounds and an operation function that are unchanged in the composite system. Thus the occurrence of a transition in the composite system produces the same variable and transition clock updates as in the individual components. Second, in the composition at either the TTM level or the SELTS level, a transition of one component is never block by the other TTM or SELTS. This follows from the interface compatibility of the modules ensuring that an input transition, if required, is always available to match another system's output transition. $\square$

The relabeling that produces $\mathbb{Q}_m$ from $\mathbb{Q}_{\widehat{M}}$ results in a coarser state output map to the system's input and output variables and relabels by τ internal transitions that do not take part in any synchronizations with compatible modules. Therefore as an immediate consequence of Lemma 4.24 we obtain the following corollary.

Corollary 4.25 *Given two interface compatible TTM modules m_1 and m_2 then for SELTS interface I as defined in Lemma 4.24, the reachable part of $\mathbb{Q}_{m_1\|m_2}$ is isomorphic to the reachable part of $\mathbb{Q}_{m_1}|[I]|\mathbb{Q}_{m_2}$.*

We do not have to worry about stopping time when composing interface compatible TTMs. Since each system allows arbitrary combinations of input events, the progress of time is never blocked by composition at the SELTS level, as it was in Example 4.18 when $\mathbb{Q}_2$ could not match the state output change of $\mathbb{Q}_1$.

We can now state the TTM module version of Theorem 3.31 as an immediate consequence of Corollary 4.25.

Theorem 4.26 *Let m_{Li}, m_{Ri} be TTM modules such that m_{Li} and m_{Ri} are interface compatible for $i = 1, 2$. If $m_{L1} \approx_{se} m_{L2}$ and $m_{R1} \approx_{se} m_{R2}$ then*

$$(m_{L1}\|m_{R1}) \approx_{se} (m_{L2}\|m_{R2})$$

This result together with Theorem 4.17 allow us to perform compositionally consistent model reduction of TTM modules.

4.5 Summary

The main contribution of this chapter is the development of a computationally effective weak model reduction technique for a simple discrete time temporal logic. The model reduction is done in a compositionally consistent way as a result of using the systems' state-event quotient systems.

The method works for the subclass of state-event stuttering-invariant formulas defined in Section 4.3.3. These formulas are robust with respect to state-event "stuttering," changes in the length of subsequences of unobservable events that do not cause observable state changes. In developing the subclass of SESI formulas we defined the notion of weak satisfaction of a temporal formula. Weak satisfaction, which can be thought of as a satisfaction relation capturing a system's observable behavior, reduces to standard satisfaction for SESI formulas. TTM modules were defined to allow the model reduction results for SELTS to be applied to systems modeled by TTMs.

Chapter 5

Design and Verification of an Industrial Real-time Controller

This chapter illustrates, via the design of a reactor shutdown system, the use of state-event equivalence as both a model reduction technique for the real-time linear temporal logic specification/verification method and an equivalence verification method in its own right. In the process we demonstrate how a design can benefit from the combined application of these two formal methods.

Traditionally in equivalence verification one has a "low level" detailed model of the implementation and a "high level" abstract model of the specification. One then verifies via computational or transformational methods, that the two models are equivalent in a well defined sense. In the case of the reactor shutdown system, equivalence-preserving transformations (Appendix A) have been used to demonstrate that an implementation TTM module is weakly state-event equivalent to a specification module (see Appendix B). This is the point where traditional equivalence verification schemes are finished with a verification problem. The assumption is that the specification model properly characterizes the requirements of the system. In the example presented here, after performing equivalence verification we then use real-time temporal logic as an alternative means of specification. Model checking is then used to verify that the specification TTM module satisfies the temporal logic specifications. By Section 4.4, the model checking results for the implementation TTM module can then be inferred from the specification module results. In the case that the high level system (and hence the low level system) fails to satisfy the desired temporal properties, the system is redesigned to satisfy the properties. One could then perform a top down design by using the equivalence preserving transformation to refine the redesigned specification into a workable implementation which is guaranteed to satisfy the temporal logic specifications.

We will see that model checking helps to identify subtle bugs that are often incorporated into high level specifications and therefore go undetected when only using equivalence verification techniques. Conversely, the ability to model check high level models and infer the results for low level models has the potential to dramatically improve the performance of model checking and in, some case, perform model checks that would otherwise be impossible due to the state explosion resulting from the composition of low level models.

The above concepts will be illustrated by the "simple" real-time control software verification problem that is described in Section 5.1. The equivalence verification proof that was done previously in [Law92] can be found in Appendix B. We take the software verification problem a step further through the application of temporal logic model-checking in Section 5.2. The scope of the verification problem is then widened in Section 5.3 to consider the behavior of the closed-loop system with redundant controllers operating concurrently. We attempt to verify the temporal logic specifications of the previous section's single controller implementation with interesting results.

5.1 The Delayed Reactor Trip System

In general it is easier to understand a mathematical theory if one can relate the theory to a physical example. This section introduces the Delayed Reactor Trip (DRT) problem, a real-time software verification example from the nuclear industry. After describing the DRT setting the software verification problem is recast as a TTM module equivalence verification problem. The solution to an equivalent formulation of the DRT example was first put forward in [Law92] and [LW95] is included in

Appendix B.

For the next generation of reactors a company hopes to use microprocessor implementations for many of the control systems that were previously implemented using discrete and analog components. The main reasons for the switch to digital control systems are the cost savings and greater flexibility typically associated with microprocessor based systems. A question that now arises is whether the new systems behave the same as the old systems. That is, are the two implementations equivalent?

5.1.1 Setting and Assumptions

The DRT system is typical of many real-time problems from industry. When a certain set of circumstances arises, we want the system to produce the correct response in a timely fashion. In this case when the reactor pressure and power exceed acceptable safety limits in a particular way, we want the DRT control system to trip a relay causing the reactor to shut down. The result of the DRT system failing to shut down

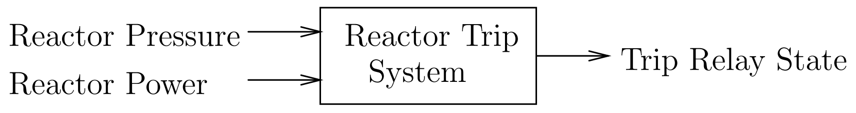


Figure 5.1: Block Diagram for the DRT System

the reactor could be catastrophic. Conversely, each time the reactor is improperly shut down, the utility operating the reactor may lose hundreds of thousands of dollars as fossil fuel powered generating stations have to be brought on line to meet demand. Clearly it is important that the DRT behave in a very specific manner.

The desired input/output relationship for the DRT block diagram has the following informal description: if the power exceeds power threshold PT and the pressure exceeds delayed set point DSP, then wait for 3 seconds. If after 3 seconds the power is still greater than PT, then open the relay for 2 seconds. The old implementation of the DRT using timers, comparators and logic gates is shown in Figure 5.2. The hardware implementation is almost a direct translation of the above informal specification. When the reactor power and pressure exceed PT and DSP respectively, the comparators cause Timer1 to start. Timer1 times out after 3 seconds, sending

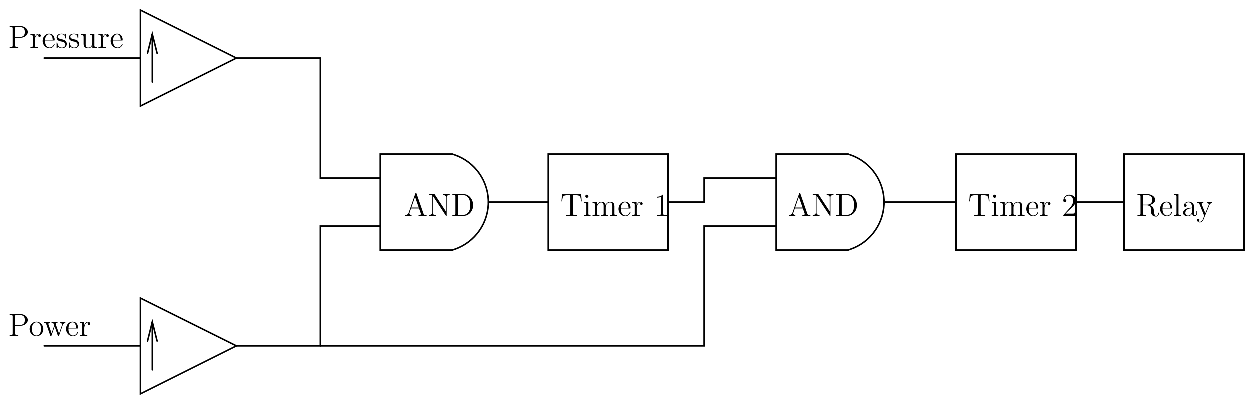


Figure 5.2: Analog Implementation of the DRT System

a signal to one input of the second AND gate. The other input of the second AND gate is reserved for the output of the power comparator. The output of the second AND gate causes Timer2 to start if the power is exceeding its threshold and Timer1 has timed out. Once Timer2 starts it runs for 2 seconds while signaling the relay to remain open.

The new DRT system is to be implemented on a microprocessor system with a cycle time of 100ms. That is, the system samples the inputs and passes through a block of control code every 0.1 seconds. We assume that the input signals have been properly filtered and that the sampling rate is sufficiently fast to ensure proper control. Figure 5.3 contains the pseudocode for a proposed control program for the microprocessor. The program makes use of the variables $Pressure$, $Power$ and $Relay$ for the sampled DRT system inputs and output respectively. Also, the code mimics the original analog implementation by using integer counter variables c_1 and c_2 in place of Timer1 and Timer2 respectively. In the pseudocode we say that a counter is "reset" when it is set to its initial value (ie. 0 in the pseudocode TTM model below). A counter variable may then be incremented in place of starting the timer it represents. With each subsequent pass through the block of code, the counter variable is incremented to represent the passage of another 100ms since the represented timer was started. Once the counter variable is equal to or exceeds a value appropriate for the particular timer and given cycle time (30 for c_1 and 20 for c_2 in the case when the cycle time is 100ms), we say that the counter variable has "timed out". The use of the terms “reset” and “timed out” in the pseudocode abstracts from the implementation details so that the pseudocode does not have to be rewritten if the cycle time of the microprocessor is changed.

That the original hardware implementation satisfies the informal specification seems obvious at a glance. The answer to the question of whether a microprocessor implementing the algorithm of Figure 5.3 satisfies the informal requirements above is somewhat more problematic. To help answer this question we now pose the DRT problem in the TTM framework.

5.1.2 Modeling the Delayed Reactor Trip Specification

By modeling the DRT specification as a TTM we can remove any ambiguities from the informal specification and ensure that the input/output behavior of the microprocessor system is completely determined. When the DRT is implemented in the actual reactor there are three identical DRT systems running in parallel, with the final decision on when to shut down the reactor implemented on a majority rule basis (see Section 5.3). As a result it is important that an individual system be able to recover when it is in disagreement with the other two systems. Also a system should never deadlock. For instance, after the power and pressure have exceeded their critical values and the system has waited 3 seconds to check the power level again, if the power is below its threshold value PT, then we wish the system to reset and go back to monitoring both inputs. This is *implicit* in the informal specification. Unfortunately, as most systems designers are painfully aware, computers require *explicit* instructions if their behavior is to be predictable.

In order to facilitate the verification process, the TTM representation of the desired I/O characteristics for the DRT is put in a form that closely resembles the microprocessor behavior. A *tick* of the global TTM clock is assumed to represent 100ms, the cycle time of the microprocessor. As mentioned in the previous subsection, we assume proper filtering of the input signals and a sufficiently high sample rate. Thus in the TTM specification SPEC of Figure 5.4, the enablement conditions of a transition must be satisfied for at least one clock tick before the transition can

```
If $Power \geq$ PT then
   If counter $c_1$ is reset then
     If counter $c_2$ is reset then
       If $Pressure \geq$ DSP then
         increment $c_1$          ]$\mu_1$
       Endif
     Else If counter $c_2$ timed out then
         reset $c_2$              ]$\gamma$
       Else
         increment $c_2$          ]$\mu_2$
         open $Relay$             ]
       Endif
     Endif
   Else If counter $c_1$ timed out then
       open $Relay$               ]
       reset $c_1$                |$\alpha$
       increment $c_2$            ]
     Else
       increment $c_1$            ]$\mu_1$
     Endif
   Endif
Else If counter $c_1$ is reset then
     If counter $c_2$ is reset then
       close $Relay$              ]$\beta$
     Else If counter $c_2$ timed out then
         close $Relay$            ]$\rho_2$
         reset $c_2$              ]
       Else
         increment $c_2$          ]$\mu_2$
         open Relay               ]
       Endif
     Endif
   Else If counter $c_1$ timed out then
       reset $c_1$                ]$\rho_1$
     Else
       increment $c_1$            ]$\mu_1$
     Endif
  Endif
Endif
```

Figure 5.3: Pseudocode for Proposed DRT Control Program

$$\begin{tikzcd}
a \arrow[loop left, "\beta"] \arrow[r, "\mu"] & b \arrow[r, "\omega_{29}"] & c \arrow[r, "\alpha"] \arrow[ll, bend left, "\rho_1"] & d \arrow[r, "\omega_{19}"] & e \arrow[llll, bend left=40, "{\rho_2, \gamma}"]
\end{tikzcd}$$

SPEC Transition Table

$$\begin{array}{rcl}
\Theta & := & x = a \wedge Relay = \text{CLOSED} \\
\mu & := & (e_\mu, [\ ], 1, 1) \\
\alpha & := & (Power \geq \text{PT}, [Relay : \text{OPEN}], 1, 1) \\
\omega_{29} & := & (True, [\ ], 29, 29) \\
\omega_{19} & := & (True, [\ ], 19, 19) \\
\rho_1 & := & (Power < \text{PT}, [\ ], 1, 1) \\
\rho_2 & := & (Power < \text{PT}, [Relay : \text{CLOSED}], 1, 1) \\
\gamma & := & (Power \geq \text{PT}, [\ ], 1, 1) \\
\text{where} & & \\
e_\mu & := & Power \geq \text{PT} \wedge Pressure \geq \text{DSP}
\end{array}$$

Figure 5.4: SPEC: TTM Representation of DRT Specification

occur. The transition μ has lower and upper time bounds of 1, exemplifying this filtering assumption.

After transition μ occurs, SPEC waits in activity b for 29 clock ticks (2.9 seconds) before proceeding to activity c. Activity c is where the power level is checked again. If the power is too high then the system opens the relay via transition α, else the system resets via ρ_1 to continue monitoring both inputs in activity a. After α the system waits in activity d for 19 clock ticks (1.9 seconds) and then moves to e. At e, as an added safety feature, the system is once again required to evaluate the power level. If $Power \geq$ PT, the system returns to activity a with the relay still open via transition γ. Otherwise the system resets to a via ρ_2 while closing the relay.

From the above paragraph it is apparent that the TTM SPEC gives a more thorough description of what is required of the DRT, expanding upon the previous informal specification. It now remains to model the microprocessor system in the TTM framework before formalizing the verification problem.

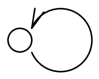
5.1.3 Modeling the Microprocessor DRT Implementation

On the right hand side of Figure 5.3 is a list of transition names. Each time the microprocessor passes through the block of code represented by the pseudocode it performs one of the group of operations identified by a transition name. Identical groups of operations on the program variables are identified by identical transition names. A group of program operations then becomes the operation function of the transition. The enablement conditions for these transitions are formed by taking the conjunction of the conditions specified by the 'If' statements for each occurrence of a given transition name's program operations. As an example consider e_{μ_2}, the enablement condition for μ_2. The first occurrence of μ_2 happens if $Pressure \geq$ DSP, $Power \geq$ PT, c_1 is reset, $\neg$(c_2 is reset) and $\neg$(c_2 has timed out). The second occurrence is executed if $\neg$($Pressure \geq$ DSP), c_1 is reset, $\neg$(c_2 is reset) and $\neg$(c_2 has timed out). Counting off 20 consecutive cycles through the code translates to an elapsed time of 2 seconds, the minimum time the relay is to remain open. If we consider the counter variables to be reset when they are equal to zero and counter c_2

as timed out when $c_2 \geq 20$, μ_2's enablement condition becomes:

$$\begin{aligned} e_{\mu_2} &= (\mathit{Pressure} \geq \text{DSP} \wedge \mathit{Power} \geq \text{PT} \wedge c_1 = 0 \wedge c_2 \neq 0 \wedge c_2 < 20) \\ &\quad \vee(\mathit{Pressure} < \text{DSP} \wedge c_1 = 0 \wedge c_2 \neq 0 \wedge c_2 < 20) \\ &= ((\mathit{Pressure} \geq \text{DSP} \wedge \mathit{Power} \geq \text{PT}) \vee \mathit{Pressure} < \text{DSP}) \\ &\quad \wedge c_1 = 0 \wedge 0 < c_2 < 20 \end{aligned}$$

In the final step we use the fact that c_2 can never be negative since it starts at $c_2 = 0$ and all transitions reset c_2 to zero or increment it.


selfloop($\mu_1, \mu_2, \alpha, \beta,$
γ, ρ_1, ρ_2)

PROG Transition Table

$$\begin{aligned} \Theta &:= c_1 = c_2 = 0 \wedge \mathit{Relay} = \text{CLOSED} \\ \mu_1 &:= (e_{\mu_1}, [c_1 : c_1 + 1], 1, 1) \\ \mu_2 &:= (c_1 = 0 \wedge 1 \leq c_2 \leq 19, [c_2 : c_2 + 1, \mathit{Relay} : \text{OPEN}], 1, 1) \\ \alpha &:= (\mathit{Power} \geq \text{PT} \wedge c_1 \geq 30, [c_1 : 0, c_2 : c_2 + 1, \mathit{Relay} : \text{OPEN}], 1, 1) \\ \beta &:= (\mathit{Power} < \text{PT} \wedge c_1 = c_2 = 0, [\mathit{Relay} : \text{CLOSED}], 1, 1) \\ \gamma &:= (\mathit{Power} \geq \text{PT} \wedge c_1 = 0 \wedge c_2 \geq 20, [c_2 : 0], 1, 1]) \\ \rho_1 &:= (\mathit{Power} < \text{PT} \wedge c_1 \geq 30, [c_1 : 0], 1, 1) \\ \rho_2 &:= (\mathit{Power} < \text{PT} \wedge c_1 = 0 \wedge c_2 \geq 20, [c_2 : 0, \mathit{Relay} : \text{CLOSED}], 1, 1) \end{aligned}$$

where

$$e_{\mu_1} := (\mathit{Power} \geq \text{PT} \wedge \mathit{Pressure} \geq \text{DSP} \wedge c_1 = c_2 = 0) \vee (1 \leq c_1 \leq 29)$$

Figure 5.5: PROG: TTM Representation of Pseudocode for DRT

Similarly we can obtain the enabling conditions for the other transitions. As mentioned earlier, with each pass through the code, the microprocessor picks out one of the labeled blocks of code. The block chosen is the one whose enabling conditions are satisfied. The program then loops back to the start and re-evaluates all the enabling conditions in the next cycle. Hence each transition has a lower and upper time bound of one.

All of the above information is used to construct the simple TTM PROG (see

Figure 5.5). The single activity is representative of the fact that the program is basically a large case statement implemented using If statements, the appropriate case being selected out of all possible cases on each pass through the code.

5.1.4 The Verification Problem in Terms of TTM Modules

Having modeled the internal workings of the specification and pseudocode as TTMs in the two preceding subsections, we can now create modules detailing the two models' interfaces. This will then allow us to recast the original question: 'Does the program do what we want?' in terms of module equivalence.

We will use *spec* to denote the module for the specification and *prog* for the module for the program. The modules' input variables are clearly $Power$ and $Pressure$. With each of these variables we associate an input transition label: α_w for $Power$ and α_p for $Pressure$. This gives us $\Sigma_{in} := \{\alpha_w, \alpha_p\}$, $\mathcal{V} := \{Power, Pressure\}$ and $R_{in} := \{(\alpha_w, Power), (\alpha_p, Pressure)\}$. The set of output variables for both modules is $\mathcal{V}_{out} := \{Relay\}$. Then by Definition 4.19 the set of output transition labels for both systems is $\Sigma_{out} := \{\alpha, \beta, \rho_2\}$, identifying the set of transitions that modify $Relay$ in both systems. Hence letting $R_{out} := \{(\alpha, Relay), (\beta, Relay), (\rho_2, Relay)\}$ we define $spec := (SPEC, \mathcal{I})$ and $prog := (PROG, \mathcal{I})$ for

$$\mathcal{I} := (\Sigma_{in}, \mathcal{V}_{in}, R_{in}, \Sigma_{out}, \mathcal{V}_{out}, R_{out})$$

as defined above.

The DRT verification problem has now been reduced to checking whether $spec \approx_{se} prog$. In [Law92] and [LW95] Equivalence Preserving Transformations were used to prove that $SPEC$ and $PROG$ would produce weakly equivalent timed input/output behavior. For completeness Appendix A contains a modified description of the theoretical explanation of equivalence preserving transformations that first appeared in [LW95] while Appendix B provides the proof of the input/output equivalence of $SPEC$ and $PROG$ that first appeared in [Law92]. In [Zha96] the algorithms for computing state-event equivalence outlined in Section 3.1 and Section 3.2 were implemented and used to verify the weak state-event equivalence of modified versions of the underlying SELTS for *spec* and *prog*. Since *spec* and *prog* have been defined with identical interfaces, the equivalence of $\mathbb{Q}_{spec}$ and $\mathbb{Q}_{prog}$ establishes that indeed $spec \approx_{se} prog$.

5.2 Model Checking the DRT

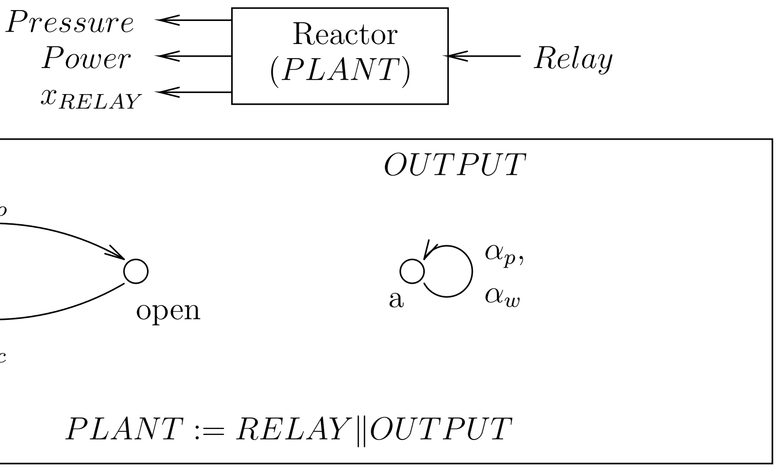
In [Law92],[LW95] and [Zha96], the DRT verification problem was deemed to be solved, in effect, as soon as *prog* was verified to be weakly state-event equivalent to *spec*. While the equivalence verification process proved to be useful (an error in the original pseudocode was found and fixed in [Law92]), the problem with such equivalence verification techniques is that while the implementation has been verified, its correct operation still depends upon the abstract specification model correctly capturing the desired system properties. An equivalent implementation is only as good as its specification. How can one verify that the original specification was correct? Is there any guarantee that the equivalence used in the verification process preserves the relevant system properties?

For the DRT we will attempt to state some desired system properties as SESI temporal logic formulas. By verifying the temporal logic specification formulas on the DRT specification module *spec* using model-checking, the satisfaction preserving properties of weak state-event equivalence will guarantee that the property holds in any equivalent implementation module. Each temporal logic formula that is model-checked on the specification will also be model-checked on the equivalent implementation. Verification of the detailed implementations provides some empirical confirmation of the correctness of Theorem 4.17, and also illustrates the computational benefits of using reduced models for verification purposes. We will see in one particular case from Section 5.3 that the state explosion of the detailed implementation as redundant controllers are added to the system, quickly causes the verification to become intractable while the reduced model experiences a roughly linear increase in time and space requirements.

5.2.1 Modeling the Reactor

Before model-checking our DRT design we have to "close the loop" by composing a model of our plant (the reactor system) with our controller model (*spec* or *prog*). Among computer specialists, modeling the plant is commonly referred to as "specifying the operating environment" of the "embedded system" (controller). Initially we will use an extremely simple model of the plant that places only minimal restrictions upon the behavior of the reactor *Power* and *Pressure* variables. Later we will make some further assumptions about the plant in order to guarantee desirable system properties and illustrate the compositional model reduction theory of the previous chapters.

A block diagram and the initial TTM model of the internal structure of the reactor are shown in Figure 5.6. $PLANT$ consists of two TTMs running in parallel. $OUTPUT$ models the "dynamics" of $PLANT$'s outputs $Power$ and $Pressure$ as



$PLANT$ Transition Table

$$\begin{aligned}
\Theta &:= x_{RELAY} = closed \wedge Relay = \text{CLOSED} \\
&\quad \wedge x_{OUTPUT} = a \wedge Power = \text{LO} \wedge Pressure = \text{LO} \\
\rho_o &:= (Relay = \text{OPEN}, [\,], 0, 0) \\
\rho_c &:= (Relay = \text{CLOSED}, [\,], 0, 0) \\
\alpha_w &:= (true, [Power : \text{LO}; Power : \text{HI}], 1, \infty) \\
\alpha_p &:= (true, [Pressure : \text{LO}; Pressure : \text{HI}], 1, \infty)
\end{aligned}$$

Figure 5.6: $PLANT := RELAY \| OUTPUT$ - TTM model of the plant.

variables that both initially have LO values (values below their respective threshold values – HI will be assigned to $Power$ or $Pressure$ to indicate values exceeding the respective thresholds). Each variable *may* be altered at most once between successive clock *tick*s by α_w (for $Power$) and α_p (for $Pressure$).

The reactor's relay is modeled by TTM $RELAY$. We assume that $RELAY$'s activity variable x_{RELAY} represents the current state of the reactor's relay. A change to the reactor's input variable *Relay* causes an "instantaneous" change in X_{RELAY} (ie. before the next clock *tick*, provided *Relay*'s value remains at the new value) so that after ρ_o or ρ_c occurs $X_{RELAY} = Relay$.

Although $RELAY$ provides the possibility of non-Zeno behavior, an infinite number of successive non-*tick* transitions, this would require non-Zeno behavior of the input variable *Relay*. In both $SPEC$ and $PROG$, all TTM transitions have lower time bounds ≥ 1 and so each can only perform a finite number of transitions between successive clock *tick*s. Similarly the $OUTPUT$ portion of $PLANT$ has all transition lower time bounds equal to 1 while the remaining $RELAY$ portion of the plant can only perform a single action without changes to its input variable *Relay*. Thus the composite system is guaranteed to have an infinite number of *tick*s in all computations and hence $control \| plant \models \Box\Diamond(\eta = tick)$ for $control \in \{spec, prog\}$. Therefore we may drop the $\neg\Box\Diamond(\eta = tick)$ disjunction that occurs in Theorem 4.17 since it is false for all computations of $control \| plant$.

We finish this subsection by formally defining the plant module to be $plant := (PLANT, \mathcal{I}_{plant})$ where the plant module interface is basically $\mathcal{I}$ for *spec* and *prog* with the input and output elements swapped and x_{RELAY}, the reactor relay state, added to the set of output variables. We add x_{RELAY} to the other *plant* output variables $Power$ and $Pressure$ because we wish to prove properties about the timed behavior of all these variables in the closed-loop system. Thus

$$\begin{aligned}\mathcal{I}_{plant} \;:=\; & (\Sigma_{out}(spec), \mathcal{V}_{out}(spec), R_{out}(spec),\\ & \Sigma_{in}(spec), \mathcal{V}_{in}(spec) \cup \{x_{RELAY}\}, R_{in}(spec))\end{aligned}$$

and so by Definition 4.22, *plant* is interface compatible with both *spec* and *prog*.

5.2.2 Model-Checking Details

The TTMs of the plant and controller systems were entered using a recent extension of the State-Time Tool developed by Ostroff *et. al.* at York University [Ost95]. State-Time is a visual modeling tool for the creation of StateChart-like hierarchical TTM systems. The reader is referred to [Ost95] for a full description of the State-Time tool and [Har87] for an introduction to StateCharts.

There has been a preliminary attempt to develop real-time model-checkers to allow the direct interpretation of TTMs and RTTL formulas [Ost92]. While the Verify tool of [Ost92] supports the data variables and interleaved, discrete-time semantics of TTMs, it cannot efficiently handle the large state spaces that are generated by the composite systems of Section 5.3 and hence was not employed in this thesis. The Verify tool has been applied to a single controller case of the DRT in [Ost95] but in that work it became apparent that the tool would not be able to handle the full 3 controller DRT version dealt with in Section 5.3.

In order to profit from the development effort already invested in untimed model-checking, StateTime has implemented a facility for translating its TTM models into (untimed) fair transition systems with integer variables and *tick* events [ON96] that can be used by the Stanford Temporal Theorem Prover (STeP) [Man94]. STeP has a Linear Temporal Logic (LTL) model-checker that can then interpret the fair transition system models and verify LTL properties. Recently StateTime has added integer "timer variables" that can be started or stopped and, while running, are decremented (or incremented) from their initial values with the occurrence of each *tick* transition. The timer variables enable the creation of "observer" systems that allow real-time properties to be verified by checking untimed temporal properties of the systems StateTime exports to STeP.

In the following model-checking results we will say that a real-time temporal logic formula F has been model-checked or verified for a given timed system when, in fact, we have verified an untimed temporal logic formula F' on the untimed system that incorporates timer variables and additional TTM transitions to "observe" the timed property. The construction of F' and the TTM transitions to be added to the system before it is translated into an untimed fair transition system can be difficult. Often the untimed model-check will fail to capture precisely the desired real-time behavior but may verify something close enough to the original real-time behavior to suit the designer's purposes. Below we assume that the untimed model-checks are "close enough" when stating that a timed property has been verified by the untimed model-check. In the absence of a powerful model-checking tool for RTTL, the untimed model-checks will have to suffice to illustrate our compositional model reduction theory.

All of the model checking results below are for the Solaris version of STeP-1.1 running on an UltraSparc1 with 288MB of RAM. The timing results are taken from STeP's estimate of the CPU time utilized in its computations. The state numbers below do not correspond to the number of states of the system being verified but rather to the number of states in a verification table used by STeP that is dependent upon the size of the system model and the formula to be model-checked [Man94].

5.2.3 Verification of System Response

This subsection demonstrates that specification models and formulas do not always embody the properties one initially thinks they capture. The first property we would like to check for our specification module, and hence the implementation module, is correct response to stimulae from the plant. The informal DRT system requirements from Section 5.1.1 may be restated in a form more suggestive of a Temporal Logic translation as:

> Henceforth, if *Power* and *Pressure* simultaneously exceed their threshold values for at least 2 *tick*s and 30 *tick*s later *Power* exceeds its threshold for another 2 *tick*s, then within 30 to 32 *tick*s open the reactor relay for at least 20 *tick*s.

In the rephrased informal specification we have added "at least 2 *tick*s" requirements to ensure that the DRT has time to react to the changes to its input.

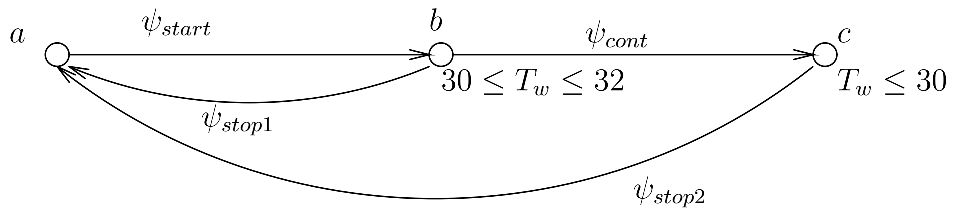
We call our temporal logic translation of this formula the System Response formula, F_{Res}:

$$\Box[\Box_{<2}(Power \geq \text{PT} \wedge Pressure \geq \text{DSP}) \wedge \Diamond_{30}\Box_{<2}Power \geq \text{PT} \\ \rightarrow \Diamond_{[30,32]}\Box_{<20}x_{RELAY} = open]$$

The first $\Box$ operator with the square braces around the rest of the formula says that the property contained within holds in the initial state of the computation and at all later points (all suffixes) of the computation. For a formula F, $\Diamond_{[30,32]}F$ is shorthand notation for $true\ \mathcal{U}_{[30,32]}F$ which translates directly as "eventually after at least 30 but no more than 32 *tick*s, F is true". $\Diamond_{30}F$ and $\Box_{<2}F$ are used to denote $\Diamond_{[30,30]}F$ and $\neg\Diamond_{[0,1]}\neg F$. We can paraphrase $\Box_{<2}F$ as "From now until 2 *tick*s have occurred, F holds".

As stated earlier, the STeP model checker does not explicitly support real-time properties. Thus in order to verify the real-time aspects of F_{Res} we will add the timer variable T_r to the $RELAY$ part of $PLANT$ to time how long $x_{RELAY} = open$. We assume that initially $T_r = 0$. The operation functions of ρ_o and ρ_c become $[cd(T_r, 20)]$ and $[stop(T_r)]$ respectively. Here $cd(T_r, 20)$ in the operation function of ρ_o has the effect of initializing T_r to a value of 20 whenever x_{RELAY} changes from *closed* to *open*. T_r will then count down with each *tick* until it reaches a value of 0 or is halted at its current value via the $stop(T_r)$ operation. Thus if $T_r = 0$ and $x_{RELAY} = open$, the reactor relay has been open for 20 *tick*s. The addition of the T_r operations to $RELAY$ will allow the untimed system to "observe" the $\Box_{<20}x_{RELAY} = open$ part of F_{Res}. The rest of the formula will be dealt with in the untimed system by an additional "property observer" TTM RES (see Figure 5.7) that will run in parallel with the rest of the system.

When $Power$ and $Pressure$ simultaneously exceed their threshold values, the ψ_{start} transition of RES starts the timer T_w counting down from 32. If $Power$ or $Pressure$ drop below their threshold values before two *tick*s of the the clock have



$$\begin{array}{rcl}
 & & \mathit{RES}\ \text{Transition Table} \\
\Theta & := & x_{RES} = a \wedge \mathit{Power} = \text{LO} \wedge \mathit{Pressure} = \text{LO} \wedge T_w = 0 \\
\psi_{start} & := & (\mathit{Power} \geq \mathit{PT} \wedge \mathit{Pressure} \geq \mathit{DSP}, [cd(T_w, 32)], 0, 0) \\
\psi_{stop1} & := & (\mathit{Power} < \mathit{PT} \vee \mathit{Pressure} < \mathit{DSP}, [stop(T_w)], 0, 0) \\
\psi_{stop2} & := & (\mathit{Power} < \mathit{PT} \wedge 0 \leq T_w \leq 2, [stop(T_w)], 0, 0) \\
\psi_{cont} & := & (T_w = 30, [\ ], 0, 0)
\end{array}$$

Figure 5.7: RES – TTM Observer for F_{Res} used in creating untimed formula F'_{Res}.

occurred (ie. before $T_w = 30$) then ψ_{stop1} occurs, stopping timer T_w. If T_w counts down to 30 then $\Box_{<2}Power \geq PT \wedge Pressure \geq DSP$ is true. Transition ψ_{cont} occurs to "observe" this fact. We then wait to check the power when $0 \leq T_w \leq 2$ (30 to 32 *tick*s after *Power* and *Pressure* first exceeded their threshold values). If during that time $Power < PT$, then the $\Diamond_{30}\Box_{<2}x_{RELAY}Power \geq PT$ conjunct in the antecedent of F_{Res} is violated so RES resets via ψ_{stop2}, stopping T_w. On the other hand, if RES is in activity c and $T_w = 0$, then $\Diamond_{30}\Box_{<2}Power \geq PT$ is *true* and previously $\Box_{<2}Power \geq PT \wedge Pressure \geq DSP$ was true since ψ_{cont} occurred to bring us to c in the first place. Thus we will approximate the antecedent of F_{Res} by $T_w = 0 \wedge x_{RES} = c$. Combining the above observations we have the untimed formula F'_{Res} that we will model-check with STeP:

$$\Box[(T_w = 0 \wedge x_{RES} = c) \rightarrow \Diamond(x_{RELAY} = open \wedge T_r = 0)]$$

Now that we have the formula F'_{Res} without timed operators we translate our system with the additional counter variables and property observer TTM into STeP compatible input and model-check F'_{Res} in place of property F_{Res}. Verifying F'_{Res}

with the STeP model-checker produces the following surprising results: We conclude

control	Result	States	Time(sec)
spec	fail	52375	2854
prog	fail	101689	21039

Table 5.1: Summary of model checking results of System Response property F'_{Res} for $control \| plant$

$spec \| plant \not\models F_{Res}$ and $prog \| plant \not\models F_{Res}$. The computational results are summarized in Table 5.1. The counterexample computation generated by $STeP$ reveals why our system specification module, implementation module, and indeed the original hardware implementation, all fail to satisfy this property.

While Timer 1 is running ($SPEC$ is in activity b or $PROG$ has a non-zero value of c_1), the system is effectively ignoring its inputs. Consider the possible input timing diagram in Figure 5.8. $Power$ and $Pressure$ simultaneously exceeding their threshold

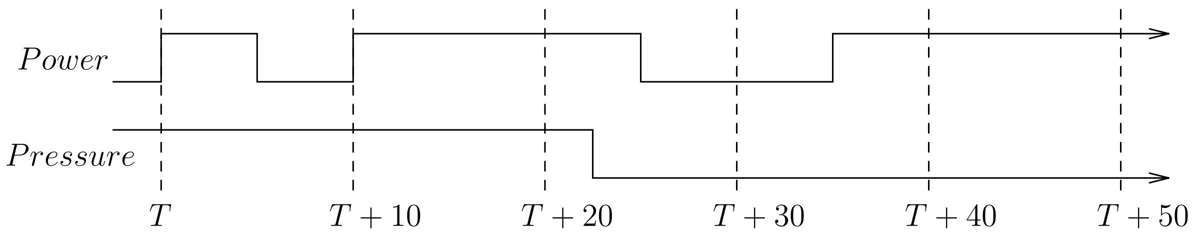


Figure 5.8: Input sequence generating a counter example to F_{Res}

values at time T will cause Timer 1 to start but at time $T+30$, $Power$ = LO so the $Relay$ = $open$ "signal" is not sent and the system goes back to monitoring its inputs. However, while Timer 1 was running, at $T+10$ $Power$ and $Pressure$ also exceeded their threshold values and 30 *tick*s later at time $T+40$ $Power$ is exceeding its threshold. Because Timer 1 was already running at $T+10$ in response to the conditions at time T, it is unable to respond to the conditions at $T+10$. The system therefore has no way of knowing that it should check the value of $Power$ at time $T+40$ and consequently open the relay.

While it is possible to design a relatively simple software implementation that does satisfy F_{Res} through the use of registers as bit arrays, for illustrative purposes we will assume that we are trying to design a software system that provides similar input/output behavior to the original system. In this case F_{Res} is an inappropriate temporal logic specification. Changing the antecedent of F_{Res} to require that the DRT controller be in its initial state (ie. neither timer is running) when $Power$ and $Pressure$ exceed their threshold values, we can alter F_{Res} to obtain a formula capturing the behavior of the original system. We call this new property the Initialized System Response formula, F_{IRes}:

$$\Box[\Theta_{CONTROL} \wedge \Box_{<2}(Power \geq \mathrm{PT} \wedge Pressure \geq \mathrm{DSP}) \wedge \Diamond_{30}\Box_{<2}Power \geq \mathrm{PT} \\ \rightarrow \Diamond_{[30,32]}\Box_{<20}X_{RELAY} = open]$$

Here $\Theta_{CONTROL} := \Theta_{SPEC}$ or $\Theta_{CONTROL} := \Theta_{PROG}$ depending on whether we are model-checking control *spec* or control *prog*.

The untimed formula F'_{Res} used in place of F_{Res} can be used as the untimed formula F'_{IRes} to model-check in place of F_{IRes} provided we modify the property observer TTM RES. We add the $\Theta_{control}$ conjunct to the enablement condition of ψ_{start} to obtain the new property observer TTM $IRES$. Thus the new enablement condition for ψ_{start} is $\Theta_{CONTROL} \wedge Power \geq PT \wedge Pressure \geq DSP$.

control	Result	States	Time(sec)
spec	pass	6305	55
prog	pass	12063	218

Table 5.2: Summary of model checking results of Initialized System Response property F'_{IRes} for $control\|plant$

The results of model-checking F'_{IRes} with its observer system are contained in Table 5.2. show that for both the specification and implementation, $control\|plant \models F'_{IRes}$.

The above pair of model checking results have helped us to gain a deeper understanding of the behavior of our system and, by the agreement of results for the use of *spec* and *prog* as the control, have illustrated Theorem 4.17. We will have more to say about the results regarding the space (number of states) and time requirements in Section 5.3.

5.2.4 Verification of System Recovery

In the original hardware implementation a signal to open the reactor relay is only sent during the 2 seconds that Timer 2 is running. As an added safety feature in our microprocessor design, $SPEC$ was set up to continue sending the $Relay = open$ signal until $Power$ was no longer exceeding its threshold. Since the DRT is but one of many reactor control systems operating in the actual reactor, a reasonable requirement might be that the closed-loop system "recover" in a timely fashion after the $Relay =$ OPEN signal has been sent for at least 20 *tick*s (2 seconds) and $Power$ returns to normal operating levels. An informal statement of this property might be:

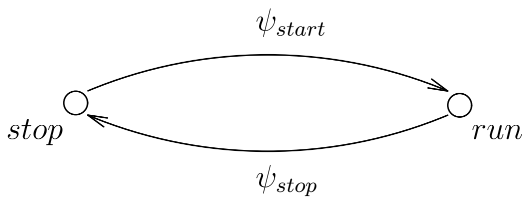
> Henceforth if $x_{RELAY} = open$ for the next 20 *tick*s and after the 20th *tick* $Power < PT$ for at least 2 *tick*s, then before the 22nd *tick* $X_{RELAY} = closed$.

We translate this statement into the System Recovery formula F_{Rec}:

$$\Box[(\Box_{<20}X_{RELAY} = open \wedge \Diamond_{20}\Box_{<2}Power = LO) \rightarrow (\Diamond_{<22}X_{RELAY} = closed)]$$

As we did for F_{Rec}, we can use the addition of the timer T_r to $RELAY$ to check the subproperty $\Box_{<20}X_{RELAY} = open$. Again the remainder of the formula will be handled by a property observer TTM. Figure 5.9 contains REC, the TTM property observer for F_{Rec}.

The transition ψ_{start} occurs once the reactor relay has been open for 20 *tick*s ($x_{RELAY} = open \wedge T_r = 0$) and $Power$ is LO ($Power < PT$). It starts timer T_w counting down from an initial value of 2. If $Power$ becomes HI or the reactor relay closes, transition ψ_{stop} takes place, immediately stopping the timer T_w and returning REC to activity *stop*. Thus if REC is in activity *run* and $T_w = 0$ then the reactor relay has been open for 20 *tick*s, and subsequently $Power$ has been LO for more than 2 clock *tick*s. This is a violation of F_{Rec}. Therefore we can reduce model-checking



$$
\begin{array}{rcl}
 & & \mathit{REC}\text{ Transition Table} \\
\Theta & := & x_{REC} = \mathit{stop} \wedge T_w = 0 \wedge x_{RELAY} = \mathit{closed} \wedge \mathit{Power} = \mathrm{LO} \\
\psi_{start} & := & (x_{RELAY} = \mathit{open} \wedge T_r = 0 \wedge \mathit{Power} < \mathit{PT}, [\mathit{cd}(T_w, 2)], 0, 0) \\
\psi_{stop} & := & (x_{RELAY} = \mathit{closed} \vee \mathit{Power} \geq \mathit{PT}, [\mathit{stop}(T_w)], 0, 0)
\end{array}
$$

Figure 5.9: REC – TTM Observer for F_{Rec} used in creating untimed property F'_{Rec}.

the timed property F_{Rec} to model-checking the untimed safety property F'_{Rec}:

$$\Box\neg(T_w = 0 \wedge x_{REC} = run)$$

Thus F'_{Rec} says that it is never the case that $T_w = 0$ when TTM REC is in activity *run*.

While it seems plausible that our current *spec* and *prog* will force the closed loop system to satisfy F_{Rec}, model-checking proves the contrary (see Table 5.3). The counterexamples generated by STeP show that the γ transitions of $SPEC$ and $PROG$ are at the root of the closed-loop systems' failures to meet the recovery specification.

Consider the TTM $SPEC$ in Figure 5.4 (p. 125). Activity e is where the value of $Power$ is reevaluated after $Relay = \text{OPEN}$ has been true for the required 20 *ticks* in activity d. If $Power \geq PT$ then $SPEC$ returns to activity a via transition γ, leaving $Relay = \text{OPEN}$. If upon returning to a, $Pressure \geq DSP$ then transition μ can occur, starting another cycle of the transition graph; only this time $Relay = \text{OPEN}$. The $Power$ may return to an acceptable level immediately following the μ transition, but the system will take 30 *tick*s to return to activity a and "recover" by finally executing a β transition that sets $Relay = \text{CLOSED}$.

control	Result	States	Time(sec)
spec	fail	1400	1
prog	fail	2528	2
$spec_r$	pass	4745	8
$prog_r$	pass	9161	16

Table 5.3: Summary of model checking results for System Recovery property F'_{Rec} for $control \| plant$

Removal of the γ transition will ensure that $SPEC$ remains at activity e until $Power < PT$. If $Power$ is less than PT while $SPEC$ is in e, then before two clock *tick*s ρ_2 occurs, setting $Relay =$ CLOSED, and thereby ensuring satisfaction of F_{Rec}. With the removal of γ, the β transition that resets $Relay$ in activity a becomes redundant, since the only way that $SPEC$ can enter a when $Relay =$ OPEN is via ρ_2. We will also delete the γ and β of $PROG$. Call the revised systems formed by the elimination of these transitions $SPEC_r$ and $PROG_r$. The new modules associated with these systems, similarly denoted by $spec_r$ and $prog_r$, can be obtained from their unprimed predecessors simply by removing all references to β from their interfaces. While the new systems are smaller and perhaps agree more closely with the designer's intuition of how the system should behave, changing the systems brings into question their equivalence and the satisfaction of the Initialized Response formula F_{IRes}, while creating the possibility that the closed-loop system will now satisfy F_{Rec}.

From Table 5.3 we see that $spec_r \| plant \models F_{Rec}$ and $prog_r \| plant \models F_{Rec}$. Further model-checks also confirm that $spec_r \| plant \models F_{IRes}$ and $prog_r \| plant \models F_{IRes}$. This mutual satisfaction of F_{Rec} and F_{IRes} by *spec* and *prog* was not merely accidental. It was forced by Theorem 4.17 because $spec_r \approx_{se} prog_r$. The Equivalence Preserving Transformation proof of $spec \approx_{se} prog$ in Appendix B can be used virtually without change to provide a proof of $spec_r \approx_{se} prog_r$. This is not particularly surprising given the simple structure of the systems and the one-to-one correspondence between β and γ transitions in *spec* and *prog*.

5.3 Model-Checking Concurrent Controllers

So far we have typically seen a factor of 2 improvement in model-checking time and space by using the reduced *spec* models instead of the full *prog* model. If this were always the case it would be hard to justify the additional complexity of the $O(n^3)$ weak state-event equivalence model reduction computation or the additional effort to reduce the system by hand using Equivalence Preserving Transformations. More dramatic gains from our model reduction technique can be made when there are multiple controllers running in parallel. Such redundant controller schemes, in particular 3-version control with majority voting logic, have been recommended for safety critical systems such as nuclear power plants [PAM91]. Each controller module is identical. Therefore, because of the compositional consistency of weak state-event equivalence for TTM modules, the model reduction computation or proof need only be performed once for a single controller module. The reduction can be used for each controller module added to the system providing a multiplicative effect in the reduction of the state size without any additional computational or manual effort. To illustrate the preceding concept, this section extends the basic DRT closed-loop system to 2 and 3 copies of our revised DRT controllers running in parallel with the plant. The enablement conditions of the plant's $RELAY$ transitions are changed to accommodate the additional controllers and the plant module's interface is modified accordingly.

We will attempt to verify F_{IRes} and F_{Rec} for compositions of the reduced and unreduced revised DRT models. The results demonstrate that the real benefits of modular model reduction are realized when multiple reduced models are composed. We will see that composition of reduced models can lead to a multiplicative effect in the reduction of the composite model that soon makes model-checking the unreduced system intractable due to memory limitations. Another interesting conclusion of the model-checking results for the composite systems is that properties satisfied by the single controller version do not necessarily hold for the multiple controller versions.

We will begin by considering the two controller case. The TTMs $SPEC_r$ and $PROG_r$ can have their transitions and internal and output variables subscripted by integers $i = 1, 2$ to avoid transition label and variable name conflicts. The $Power$ and $Pressure$ variables and their associated input transition labels α_w and α_p are the only parts of the controller TTMs that need not be subscripted. The reactor TTM $PLANT$ supplies these inputs to the multiple controllers. Thus when defining the modules, the input part of the control modules' interfaces remains the same. The new modules are, for $i = 1, 2$, $spec_{r_i} := (SPEC_{r_i}, \mathcal{I}_i)$ and $prog_{r_i} := (PROG_{r_i}, \mathcal{I}_i)$ where

$$\begin{aligned}\mathcal{I}_i \;:=\; & (\Sigma_{in}(spec_r), \mathcal{V}(spec_r), R_{in}(spec_r),\\ & \{alpha_i, \rho_{2_i}\}, \{Relay_i\}, \{(alpha_i, Relay_i), (\rho_{2_i}, Relay_i)\})\end{aligned}$$

In interfacing the plant with the two controllers we assume that the plant will only change the state of the reactor relay x_{RELAY} when both controllers are in agreement. To accomplish this we modify the $PLANT$ TTM of Figure 5.6 to obtain the TTM $PLANT_2$ as follows:

(i) Initial condition Θ_{PLANT_2} is obtained from Θ_{PLANT} by removing the $Relay =$ CLOSED conjunct and replacing it with $Relay_1 = \text{CLOSED} \wedge Relay_2 = \text{CLOSED}$.

(ii) The enablement conditions for ρ_c and ρ_o are changed to $Relay_1 = \text{CLOSED} \wedge Relay_2 = \text{CLOSED}$ and $Relay_1 = \text{OPEN} \wedge Relay_2 = \text{OPEN}$ respectively.

We then have the two version plant module $plant_2 := (PLANT_2, \mathcal{I}_2)$ where $\mathcal{I}_2$ is an appropriately defined modification of the interface used in $plant$. The results of model-checking are shown in Table 5.4. In this table and Table 5.5 below, '?' as a table entry indicates that the results of the attempted model-check were indeterminate as the computation out of memory and failed to terminate successfully.

We see that for the $control_1 \| control_2 \| plant_2$ case the model-checks of property F_{IRes} ran out of memory for both the reduced $control_i := spec_{r_i}$ case and the detailed $control_i := prog_{r_i}$ case. This can be attributed to the *PSPACE*-completeness of Linear Temporal Logic model checking [SC85]. Model-checking algorithms typically employed for Linear Temporal Logic have a complexity of $O(|\mathbb{Q}_m|^{|F|})$, where $|F|$ is

		F'_{IRes} - System Response			F'_{Rec} - System Recovery		
# in $\|$	*control*	Result	States	Time(sec)	Result	States	Time(sec)
1	$spec_r$	pass	6305	55	pass	4745	8
	$prog_r$	pass	12063	218	pass	9161	16
2	$spec_r$	?	?	?	fail	1141	1
	$prog_r$	?	?	?	fail	8025	10

Table 5.4: Summary of model checking $control \| plant$ and $control_1 \| control_2 \| plant_2$

the number of temporal operators in F and $|\mathbb{Q}_m|$ is the number of transitions plus the number of states of the SELTS generated by the module m [LP85]. In the case of F_{IRes} any state space reduction achieved by the use of $spec_r$ was negated by the $|F_{IRes}|$ exponent.

The results of the model-check for the somewhat simpler property F_{Rec} show a definite improvement in the time and space required to decide the property using the reduced models. The answer is somewhat unexpected. While operating in the single control environment both $spec_r$ and $prog_r$ result in closed loop systems that satisfy F_{Rec} but when run concurrently with another control, the closed-loop system fails to satisfy F_{Rec}. The counterexamples generated by STeP show that controllers can get out of synchronization from their initial states. If $Power \geq PT$ while $Pressure < DSP$ then the following state-event sequence can occur in $spec_{r_1} \| spec_{r_2} \| plant_2$:

$$(\mathrm{HI}, \mathrm{LO}, a, a) \xrightarrow{\alpha_p} (\mathrm{HI}, \mathrm{HI}, a, a) \xrightarrow{tick} (\mathrm{HI}, \mathrm{HI}, a, a) \xrightarrow{\mu_1} (\mathrm{HI}, \mathrm{HI}, b, a) \xrightarrow{\alpha_w} (\mathrm{LO}, \mathrm{HI}, b, a) \xrightarrow{tick} \ldots$$

The 4-tuples represent the value of the variables $(Power, Pressure, x_{SPEC_{r_1}}, x_{SPEC_{r_2}})$. We see that once $Pressure = \mathrm{HI}$ for one *tick*, module $spec_{r_1}$ reacts, but before $spec_{r_2}$ can react, α_w occurs setting $Power = \mathrm{LO}$ and disabling μ_2. The two systems are now out of synchronization and the situation deteriorates from there to a point where the reactor relay, once opened for more than 20 *tick*s will in some cases not close even if $Power < PT$ for up to 19 *tick*s! At first one might think the failure of the 2 controller system to satisfy F_{Rec} is the result of the lower time bounds of 1 on the reactor $OUTPUT$ transitions α_w and α_p but putting reactor outputs through a low pass filter to increase the lower bounds up to at least 19 would still fail to eliminate all possible counterexamples.

Instead we must place some restrictions upon our plant to ensure that multiple controllers are reacting to the same output samples. The plant behavior restrictions are implemented by replacing the TTM $OUTPUT$ in $PLANT$ with the new TTM $OUTPUT_{sh}$, a TTM that implements a sample and two *tick* hold on the reactor outputs. This change eliminates the previous problem. The sample and hold version of the plants for the one, two and three controller cases are shown in Figure 5.10. $PLANT_{sh_1}$ and $PLANT_{sh_2}$ are sample and hold versions of $PLANT$ and $PLANT_2$ respectively while $PLANT_{sh_3}$ implements a majority vote scheme in the enablement conditions of ρ_{o_3} and ρ_{c_3}. For X_{RELAY}, the reactor relay state, to change in $PLANT_{sh_3}$ at least 2 of the three control modules must agree to the change.

As was the case for *plant*, for $n = 1, 2, 3$ $control \| plant_{sh_n} \models \Box\Diamond(\eta = tick)$. Although μ_h, α_p and α_s have lower and upper time bounds of 0 in $OUTPUT_{sh}$, at most only one of each of these transitions can occur before the next μ_s transition. Since μ_s has a lower time bound of 2, $OUTPUT_{sh}$ can only generate a finite number of transitions between successive *tick*s. The remaining closed loop system components are the same as for the $PLANT$ case so the same arguments can be applied. Thus we can continue to check the desired system properties without adding the disjunctive clause $\neg\Box\Diamond(\eta = tick)$ of Theorem 4.17.

The results of model-checking for the sample and hold closed-loop systems are shown in Table 5.5. Due to memory limitations we are still unable to verify the response property F_{IRes} for the multiple controller case.

The results for the verification of F_{Rec} are very promising. Model reduction provides the same positive answer as the unreduced system in the one and two controller cases with the unreduced ($prog_r$ control) closed-loop taking approximately twice the number of states and time for the single controller closed-loop system and four times the number of states and time for the dual controller closed-loop system. This is in keeping with the multiplicative effect that the theory predicted would result from the composition of reduced models. When we go to the 3 control majority vote closed-

		F'_{IRes} - System Response			F'_{Rec} - System Recovery		
# in ‖	*control*	Result	States	Time(sec)	Result	States	Time(sec)
1	$spec_r$	pass	7023	129	pass	7965	11
	$prog_r$	pass	13342	398	pass	16191	29
2	$spec_r$	?	?	?	pass	9489	101
	$prog_r$	?	?	?	pass	34523	456
3	$spec_r$	?	?	?	pass	12897	35
	$prog_r$	?	?	?	?	> 540000	> 105min

Table 5.5: Summary of model-checking $control_1\|plant_{sh}$, $control_1\|control_2\|plant_{sh_2}$ and $control_1\|control_2\|control_3\|plant_{sh_3}$

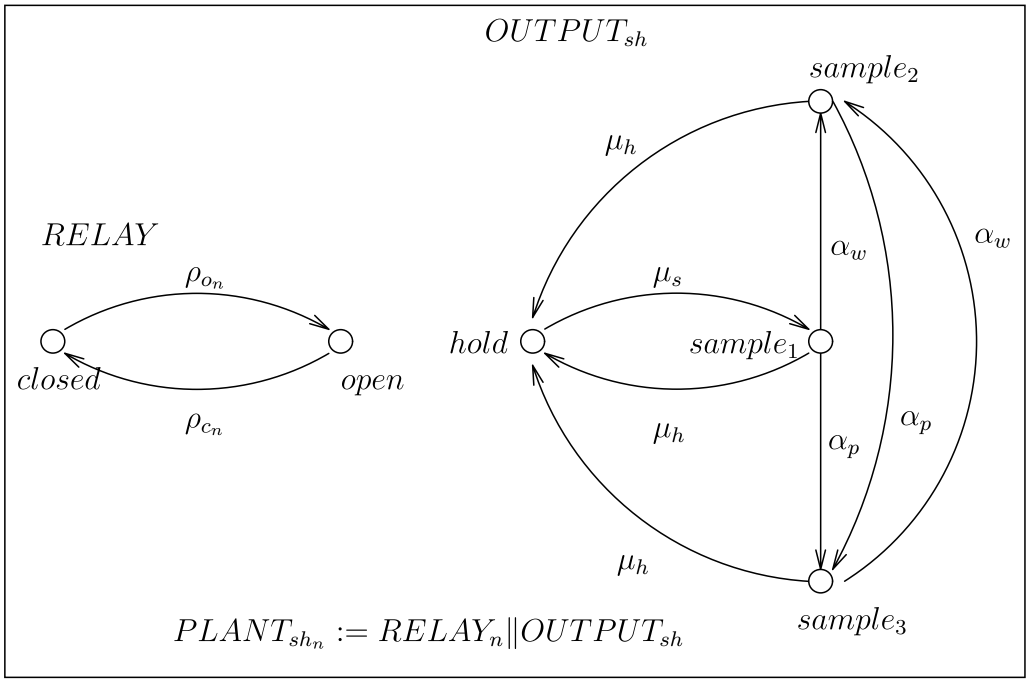
loop system, the results are even more dramatic. The state size of the reduced ($spec_r$ control) closed-loop system continues to grow in a roughly linear fashion but there is a sudden state explosion that results in the unreduced case that prevents us from finishing the model-check (the computation runs out of memory). The state explosion is in large part due to the interleavings of events that increment the internal counter variables of $prog_{r_i}$. Fortunately there is no need to model-check the unreduced 3 controller closed-loop system. We can already conclude that this system satisfies F_{Rec} because $spec_{r_1}\|spec_{r_2}\|spec_{r_3}\|plant_{sh_3} \models F_{Rec}$.

5.4 Summary

The model-checking results confirm the correctness of Theorem 4.17. Weakly state-event equivalent systems did satisfy the same formulas on all their computations in which time advances. The benefits of compositionally consistent model reduction have been demonstrated by the multiple controller F_{Rec} model-checking results. In this example verification of the composite system composed of unreduced models was impossible on the given hardware, but the model-check of the composition of reduced system was easily handled. This allowed us to know what the result for the unreduced composition would be without having to compute it.

We also discovered that model reduction alone is not enough in some cases (eg. F_{IRes} with two or more controllers). In such cases the reduced system model may still result in a composite system that is too large to be verified on the available hardware. This is particularly true of linear temporal logics where the complexity of the model checking algorithms grows exponentially with the number of temporal operators. In these cases a designer may wish to investigate the possibility of using branching time temporal logics such as CTL to express system properties. Unfortunately the StateTime tool does not currently support the transition system formats of other untimed model checking systems, so we leave the investigation of other temporal logic frameworks for future research.

During the temporal logic model-checking process two interesting additional points came to light. First, we found that specification models used in equivalence verification may not always accurately capture the designer's concept of the system requirements. Temporal logic model checking of specification models used in the equivalence verification process can be most useful for identifying any errors in the specification model, helping the designer to discover whether the correct specification model has been chosen. The counter example generation features of the model-checker are particularly useful in this regard. The counter examples from the failed model-checks of the DRT system illuminated system behavior that otherwise would not have been considered in the system design. The model-checking in turn benefited from the compositionally consistent equivalence verification technique as it provided a means of compositionally consistent model reduction. In the case of the DRT design, the combination of equivalence verification and model-checking were mutually beneficial, leading to a better design than would have been achieved by the application of either method in isolation.



$$
\begin{array}{rcl}
 & & PLANT_{sh_n}, n = 1, 2, 3 \text{ Transition Table} \\
\Theta & := & x_{RELAY} = closed \wedge \bigwedge_{i=1,\ldots,n} Relay_i = \text{CLOSED} \\
 & & \wedge x_{OUTPUT} = hold \wedge Power = \text{LO} \wedge Pressure = \text{LO} \\
\rho_{o_n} & := & (e_{o_n}, [\,], 0, 0) \\
\rho_{c_n} & := & (e_{c_n}, [\,], 0, 0) \\
\mu_h & := & (true, [\,], 0, 0) \\
\mu_s & := & (true, [\,], 2, 2) \\
\alpha_w & := & (true, [Power : \text{LO}; Power : \text{HI}], 0, \infty) \\
\alpha_p & := & (true, [Pressure : \text{LO}; Pressure : \text{HI}], 0, \infty) \\
 & & \text{where} \\
e_{\rho_{o_1}} & := & (Relay_1 = \text{OPEN}) \\
e_{\rho_{c_1}} & := & (Relay_1 = \text{CLOSED}) \\
e_{\rho_{o_2}} & := & (Relay_2 = \text{OPEN} \wedge Relay_2 = \text{OPEN}) \\
e_{\rho_{c_2}} & := & (Relay_2 = \text{CLOSED} \wedge Relay_2 = \text{CLOSED}) \\
e_{\rho_{o_3}} & := & (Relay_1 = \text{OPEN} \wedge Relay_2 = \text{OPEN}) \vee (Relay_1 = \text{OPEN} \\
 & & \wedge Relay_3 = \text{OPEN}) \vee (Relay_2 = \text{OPEN} \wedge Relay_3 = \text{OPEN}) \\
e_{\rho_{c_3}} & := & (Relay_1 = \text{CLOSED} \wedge Relay_2 = \text{CLOSED}) \vee (Relay_1 = \text{CLOSED} \\
 & & \wedge Relay_3 = \text{CLOSED}) \vee (Relay_2 = \text{CLOSED} \wedge Relay_3 = \text{CLOSED})
\end{array}
$$

Figure 5.10: $PLANT_{sh_n} := RELAY_n \| OUTPUT_{sh}$ - TTM models for plants with sample and hold on outputs.

Chapter 6

Conclusions

We have investigated State-Event Labeled Transition Systems (SELTS) with unobservable transitions, as a framework in which complexity may be hidden, and hierarchy induced through quotient systems. This effort has led to the discovery of unifying constructs, called state-event observers, which subsume both deterministic state observers [Won76] and event based observation equivalences [Mil80, Mil89]. The close relationship between state-event observers and (event) observation equivalences makes possible efficient polynomial time algorithms for computing state-event observers for finite state SELTS. Strong and weak state-event equivalences for comparison of SELTS are derived from state-event observers.

A SELTS synchronous composition operator is defined that models both shared transitions and shared variables. The algebraic characterization of state-event equivalence using SELTS homomorphisms was used to demonstrate that state-event equivalence is a congruence for the SELTS synchronous composition operator. Thus in a synchronous composition one may replace the system's modules (components) with equivalent quotient modules. Typically the resulting system has a smaller state space and is equivalent to the original composition. The size of a composite system's state space grows as the product of the sizes of the components' state spaces, so any reductions to system modules will have a multiplicative effect. In this way the state explosion problem is dealt with at the component level *before* it occurs in the composite system.

Any two strongly state-event equivalent systems are shown to satisfy identical sets of formulas of a simple discrete time temporal logic derived from Ostroff's RTTL. This result is extended to the preservation of truth values by weak state-event equivalent systems for the class of state-event stuttering invariant formulas. The latter result, together with the fact that state-event composition is a congruence for synchronous product, allows one to perform compositionally consistent model reduction. TTM modules are defined to allow the model reduction results for SELTS to be applied to systems modeled by TTMs.

The effectiveness of compositionally consistent model reduction techniques in dealing with the state explosion problem is illustrated by the application of weak state-event model reduction to the Delayed Reactor Trip (DRT) system. In this case the greatest return on model reduction efforts is seen when more than one identical module appears in a parallel composition, as in the case of redundant controllers implementing a majority vote scheme. In this case the reduction is performed once for a single module, and the reduced module is then substituted for all copies of the component.

6.1 Limitations and Future Research

In this thesis we have not addressed many of the details involved in creating working implementations. No attempt was made to deal with problems such as numerical overflow, data conversion errors and other troubles that often result in system failure. A method of dealing with these important issues in a rigorous fashion would go a long way towards bridging the gap between theory and practice in the design of safety critical systems.

As evidenced by the failure of our model reduction technique in dealing with the system response properties for the multiple controller cases of the DRT, there are cases when weak state-event model reduction alone does not succeed in making a verification problem tractable. The inability to verify the system response property was in no small part due to the particular choice of temporal logic used for specifications. As mentioned in the previous chapter, model-checking algorithms employed for Linear Temporal Logic (LTL) have a complexity of $O(|\mathbb{Q}_m|^{|F|})$, where $|F|$ is the number of temporal operators in F and $|\mathbb{Q}_m|$ is the number of transitions plus the number of states of the SELTS generated by the module m [LP85]. On the other hand model-checking for the branching time temporal logic CTL can be done in time $O(|\mathbb{Q}_m| * |F|)$ [CES86]! Although the expressive powers of LTL and CTL are incomparable, when a complex property can be expressed in both logics, CTL is clearly the preferable choice for model-checking purposes. In this circumstance, a real-time state-event extension to CTL should hold considerable promise for proving real-time properties of TTMs via model checking. The alternative definition of Milner's observation equivalence given in Section A.2 makes the branching nature of observation equivalence apparent with its use of existential quantifiers. Two states are equivalent iff they have the same future choices of observable events to equivalent states. Thus state-event equivalence should be better suited to performing model reduction for branching time logics. Results similar to those obtained for our RTTL style logic should be attainable for any similar real-time extensions of CTL.

While the initial model-checking results are promising, further practical application of weak state-event model reduction to industrial problems is needed to establish the method's advantages and limitations. Although the transformational technique of Appendix A was adequate for performing model reduction of the DRT example, an automated method for computing a finite state system's weak state-event quotient system would greatly facilitate the use of the equivalence in larger practical examples. One possible solution to this problem is the use of model-checkers for the μ-calculus [Par81] to compute the weak state-event quotient systems. In [BCM92] the authors provide not only a μ-calculus characterization of observation equivalence that could be easily adapted to computing weak state-event equivalence, but also characterizations of LTL and CTL, enabling a μ-calculus model-checking tool to perform both the model reduction and model-checking.

The power of state-event equivalence to support compositional model reduction and equivalence verification, its simple algebraic characterization, and the above-noted future research possibilities, indicate that further investigation of the state-event theory is warranted.

Appendix A

Equivalence Preserving Transformations of TTMs

This appendix summarizes the theoretical results of [Law92],[LW95] and points out some of the work's limitations that we attempt to address in this thesis. These previous results are covered in detail since they motivated much of the present work and are applied to the detailed example of Chapter 5.

We begin by detailing the model of TTM input/output behavior and resulting definition of equivalence that formed the basis of the transformations. Next we use bisimulations to define Milner's strong and weak observation equivalences that were used as the notion of system equivalence and then detail the equivalence preserving TTM transformations. Finally the limitations of the results are briefly discussed.

A.1 Equivalence of TTMs

In this section Labeled Transition Systems (LTS) are used to describe the behavior of TTMs and thus allow us to develop a notion of equivalence for TTMs. LTS have been used by De Nicola [DeN87] to compare different notions of equivalence proposed for concurrent systems. We now borrow some of the definitions and notation of [DeN87].

A *Labeled Transition System* $\mathbb{Q} := \langle Q, \Sigma, R_\Sigma, q_0 \rangle$ is simply a SELTS without the state output map. Following the notation of [Mil80] and [Mil89], the special symbol

$\tau \in \Sigma$, is used to denote *internal* (unobservable) actions. Hence $q\xrightarrow{\tau}q'$ means that the system can move from q to q' via an unobservable or *silent* transition.

The following notation is also helpful:

$\Sigma_o := \Sigma \setminus \{\tau\}$ denotes the set of observable actions.

Σ^* denotes the set of finite strings of actions.

$a\xrightarrow{s}a'$ where $s = \alpha_1\alpha_2\ldots\alpha_k \in \Sigma^*$ denotes $(\exists q_1, \ldots, q_{k-1} \in Q)$ $q\xrightarrow{\alpha_1}q_1\xrightarrow{\alpha_2}\ldots q_{k-1}\xrightarrow{\alpha_k}q'$ and $q\xrightarrow{s}$ will mean $(\exists q' \in Q)q\xrightarrow{s}q'$.

$q\overset{\alpha}{\Rightarrow}q'$ means $\begin{cases} q\overset{\tau^m\alpha\tau^n}{\rightarrow}q', \alpha \in \Sigma_o \\ q = q' \text{ or } q\overset{\tau^m\alpha\tau^n}{\rightarrow}q', \alpha = \tau \end{cases}$ for $m, n \in \mathbb{N}$.

The idea behind the relation $\overset{\alpha}{\Rightarrow}$ is that the system can move from q to q' while producing observation α. As before we will write $q\overset{\alpha}{\Rightarrow}$ as a short form for $(\exists q' \in Q)q\overset{\alpha}{\Rightarrow}q'$.

One of the operations of [Mil89] that we will find useful is that of relabeling an LTS. In this operation the structure of an LTS is left unaltered while the transition labels are changed in a consistent way. That is, if one instance of a label is changed to a new label, then all instances of the label must be changed to the same new label in the relabeled LTS.

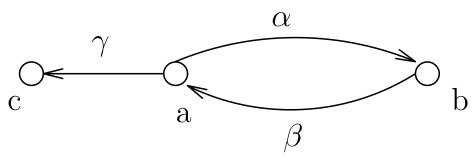
Definition A.1 *Let r be a function from transition labels to transition labels and $\mathbb{Q} := \langle Q, \Sigma, R_\Sigma, q_0\rangle$ be an LTS. Then the r* **relabeling of** $\mathbb{Q}$ *is given by:*

$$r(\mathbb{Q}) := \langle Q, \{r(\alpha) : \alpha \in \Sigma\}, R_{r(\Sigma)}, q_0\rangle$$

We now consider $\mathbb{Q}_M$, the Labeled Transition System generated by a TTM $M := \langle \mathcal{V}, \Theta, \mathcal{T}\rangle$. There are many possible LTS that represent the legal trajectories of a given TTM but for simplicity we adopt tree structures with all possible next transitions exiting the current LTS state to new LTS states. It is often the case that the transition names are unimportant. What is important is the effect the transitions have upon the variables of interest and how the latter affect the ordering of transitions. Accordingly the event labels of $\mathbb{Q}_M$ are the actual operation functions of the TTM transitions. We will see in the example below that h_α (where $h_\alpha = [w : w+1, y : y+z]$) is written in $\mathbb{Q}_M$ when transition α occurs in the legal trajectory of M. A convenient state set for $\mathbb{Q}_M$ is the set of all finite strings of transitions $\mathcal{T}^*$. We then let the initial state of $\mathbb{Q}_M$ be ϵ, the empty string. The transition relations follow naturally by defining for any $s \in \mathcal{T}^*$, $s \overset{h_\alpha}{\rightarrow} s\alpha$ if, starting from its unique initial state assignment, M can perform the transitions $s\alpha$ as the initial sequence of transitions of a legal trajectory. More formally, for a TTM $M := \langle \mathcal{V}, \Theta, \mathcal{T} \rangle$ we have $\mathbb{Q}_M := \langle \mathcal{T}^*, \{h_\alpha : \alpha \in \mathcal{T}\}, R_{\{h_\alpha\}}, \epsilon \rangle$.

As we have defined them, each TTM has a unique initial state and the operation functions of the TTM's transitions are deterministic. Thus the effect on a TTM's variables can be completely determined by knowing the sequence of transitions that has taken place. This is what will allow us to compare the behavior of TTMs by comparing forms of the LTS that they generate. From now on the LTS representing the behavior of a TTM will be the LTS $\mathbb{Q}_M$ as described above.

Consider M, the simple TTM of Figure A.1. The LTS representing the behavior



$$\begin{aligned}
\Theta &:= x = \mathrm{a} \wedge v = w = y = z = 0 \\
\alpha &:= (w = 0, [w : w+1, y : y+z], 0, 1) \\
\beta &:= (\mathit{true}, [w : w-1, z : z-1], 0, 0) \\
\gamma &:= (w = 0 \wedge y \leq 0, [w : -1, v : v+1], 1, 2)
\end{aligned}$$

Figure A.1: Simple TTM $M := \langle \mathcal{V}, \Theta, \mathcal{T} \rangle$

of M, which we denote by $\mathbb{Q}_M$, is shown in Figure A.2. Note that the *tick* transitions of the clock have been included in $\mathbb{Q}_M$ and that at each state all legal continuations of the trajectory are possible. The self-looped h_{tick} transition at the end of some paths is for display purposes only and helps indicate that the path can only be continued by an infinite string of *tick*s.

We now consider the restriction of an LTS (representing the behavior of a TTM)

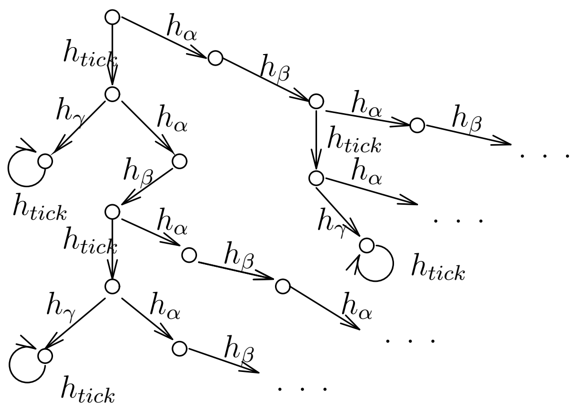


Figure A.2: $\mathbb{Q}_M$ - the LTS reachability tree for M

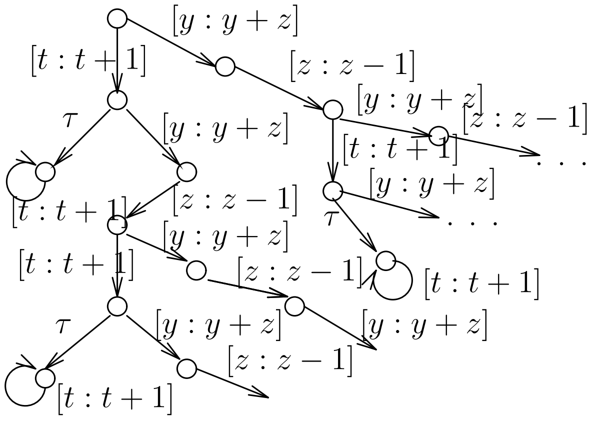


Figure A.3: $T_M|\{y, z\} = r(\mathbb{Q}_M)$ the restricted LTS for M

to a subset of variables of interest. We need some preliminary definitions.

Definition A.2 *For a TTM M with variable set $\mathcal{V}$ and a subset of variables $\mathcal{U} \subset \mathcal{V}$, we define the* **state assignments over** $\mathcal{U}$*, denoted by $\mathcal{Q}_\mathcal{U}$, to be the product of the ranges of the variables in $\mathcal{U}$. Hence*

$$\mathcal{Q}_\mathcal{U} := \times_{v_i \in \mathcal{U}} Range(v_i)$$

The **natural projection** $P_\mathcal{U} : \mathcal{Q} \to \mathcal{Q}_\mathcal{U}$ *maps a state assignment to its corresponding state assignment over $\mathcal{U}$.*

Definition A.3 *Suppose $M := \langle \mathcal{V}, \Theta, \mathcal{T} \rangle$ is a TTM, $\mathcal{U} \subset \mathcal{V}$ is a set of variables, and $\alpha \in \mathcal{T}$ is a transition. Let $h_\alpha : \mathcal{Q} \to \mathcal{Q}$ be the operation function of α and $P_\mathcal{U} : \mathcal{Q} \to \mathcal{Q}_\mathcal{U}$ be the natural projection from the state assignments $\mathcal{Q}$ to $\mathcal{Q}_\mathcal{U}$, the state* *assignments over $\mathcal{U}$. Then the* **map induced in $\mathcal{Q}_{\mathcal{U}}$ by** h_α*, when it exists, is the map $\overline{h_\alpha} : \mathcal{Q}_{\mathcal{U}} \to \mathcal{Q}_{\mathcal{U}}$ such that $P_{\mathcal{U}} \circ h_\alpha = \overline{h_\alpha} \circ P_{\mathcal{U}}$.*

The relationship between h_α and $\overline{h_\alpha}$ is illustrated in the commutative diagram, Figure A.4.

$$\begin{tikzcd}
\mathcal{Q} \arrow[r, "h_\alpha"] \arrow[d, "P_{\mathcal{U}}"'] & \mathcal{Q} \arrow[d, "P_{\mathcal{U}}"] \\
\mathcal{Q}_{\mathcal{U}} \arrow[r, dashed, "\overline{h_\alpha}"] & \mathcal{Q}_{\mathcal{U}}
\end{tikzcd}$$

Figure A.4: Commutative Diagram for Induced Operation Function

For a given $\mathcal{U}$, $\overline{h_\alpha}$ will exist if the operations of h_α upon the elements of $\mathcal{U}$ are independent of the values of the variables in $\mathcal{V} - \mathcal{U}$. For instance with $h_\alpha := [w : w + 1, y : y + z] = [w : w + 1, y : y + z, z : z]$ and $\mathcal{U} = \{y, z\}$ we have $\overline{h_\alpha} = [y : y + z]$. Note that $\overline{h_\alpha}$ is not defined for $\mathcal{U} = \{w, y\}$ since the new value of y depends upon the current value of z. The existence condition for $\overline{h_\alpha}$ can be formally stated as the mapping kernel condition $\ker(P_{\mathcal{U}}) \leq \ker(P_{\mathcal{U}} \circ h_\alpha)$.

We now have the machinery to define the timed behavior of a TTM M restricted to a subset of its variables.

Definition A.4 *For $M := \langle \mathcal{V}, \Theta, \mathcal{T} \rangle$, $\mathcal{U} \subset \mathcal{V}$ and $\mathbb{Q}_M := \langle \mathcal{T}^*, \{h_\alpha : \alpha \in \mathcal{T}\}, R_{\{h_\alpha\}}, \epsilon \rangle$ we define the* **restriction of $\mathbb{Q}_M$ to $\mathcal{U}$** *as follows. Let r be the LTS relabeling function such that $r(h_\alpha) = \overline{h_\alpha}$ where $\overline{h_\alpha}$ is the map induced in $\mathcal{Q}_{\mathcal{U}'}$ by h_α when $\mathcal{U}' := \mathcal{U} \cup \{t\}$. Then*

$$\mathbb{Q}_M|\mathcal{U} := r(\mathbb{Q}_M) = \langle \mathcal{T}^*, \{r(h_\alpha) : \alpha \in \mathcal{T}\}, R_{r(\{h_\alpha)\}}, \epsilon \rangle$$

We then denote the **timed behavior of M restricted to $\mathcal{U}$** *by $M|\mathcal{U} := \mathbb{Q}_M|\mathcal{U}$.*

Note that $\mathbb{Q}_M|\mathcal{U}$ is defined iff $(\forall \alpha \in \mathcal{T})$ $\overline{h_\alpha}$ exists. When $\mathbb{Q}_M|\mathcal{U}$ is defined we say that $\mathcal{U}$ is *restrictable* for M.

If the variables of interest for the TTM M of Figure 1 are $\mathcal{U} = \{y, z\}$ (and implicitly t to guarantee the timing) then the LTS of the behavior of M over these variables can be obtained by replacing the transitions' operation functions with their induced maps. For example we replace h_α in $\mathbb{Q}_M$ with $\overline{h_\alpha} := [y : y + z]$. In the case of the transition γ, $\overline{h_\gamma} := [\,]$ the identity or 'silent' function for $\{y, z, t\}$. $T_M|\{y, z\}$, the restriction of $\mathbb{Q}_M$ as described above, is shown in Figure A.3. Here we replace h_γ with the silent transition τ to help it stand out in the graph. Starting from the initial state of $T_M|\{y, z\}$, if the first transition is a clock tick, the next event may be y changing to $y + z$ or the system moving unobservably via τ to a state where no further changes can be made to $\{y, z\}$.

The example of Figure A.3 illustrates how restriction can create systems that can move unobservably to a deadlocking state - a state with only strings of *tick*s as possible legal continuations. We shall use a notion of equivalence that can distinguish between a deadlocking and a non-deadlocking system.

The main purpose of looking at the LTS generated by a TTM is to develop a notion of equivalence for TTMs. We will consider two TTMs to be equivalent over a set of variables $\mathcal{U}$ if their initial states agree on all variables in $\mathcal{U}$ and their respective LTS are equivalent when restricted to the variables of interest. More formally:

Definition A.5 *Given two TTMs $M_1 := \langle \mathcal{V}_1, \Theta_1, \mathcal{T}_1 \rangle$ and $M_2 := \langle \mathcal{V}_2, \Theta_2, \mathcal{T}_2 \rangle$ and EQ, an equivalence relation over the set of all LTS. Let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be the sets of state assignments for M_1 and M_2 and $P_1 : \mathcal{Q}_1 \to \mathcal{Q}_{\mathcal{U}'}$ and $P_2 : \mathcal{Q}_2 \to \mathcal{Q}_{\mathcal{U}'}$ be their respective natural projections, for some $\mathcal{U}$, a set of variables. We say that M_1* **is** *EQ* **equivalent over $\mathcal{U}$ to** *M_2, written M_1 EQ$/\mathcal{U}$ M_2, if and only if*

(i) If $q_1 \in \mathcal{Q}_1$ and $q_2 \in \mathcal{Q}_2$ then $q_1(\Theta_1) = true$ and $q_2(\Theta_2) = true$ implies $P_1(q_1) = P_2(q_2)$

(ii) $\mathbb{Q}_{M_1}|\mathcal{U}$ EQ $\mathbb{Q}_{M_2}|\mathcal{U}$

where $\mathbb{Q}_{M_1}$ and $\mathbb{Q}_{M_2}$ are the LTS generated by M_1 and M_2 respectively.

In practice usually $\mathcal{U} \subset \mathcal{V}_1 \cap \mathcal{V}_2$ though this need not be the case in general. The first condition in the definition guarantees that the systems start out in state assignments that are identical when restricted to $\mathcal{U}$ while the second condition guarantees that observed changes to variables in $\mathcal{U}$ will be equivalent.

A.2 Observation Equivalence

We begin this section by introducing an equivalence that is much stronger than what we require. The strong observation equivalence of [Mil89] treats silent τ transitions as if they were observable. Informally it requires that in a system each state reached by s, a string of transitions, there must be a state reachable by s in the equivalent system that has the same choice of next transitions (including the unobservable τ transition) at each step along the way and in the end state. We begin by defining the notion of a *strong bisimulation* that will help us to capture this informal property.

More formally, let $\mathbb{Q}_1 := \langle Q_1, \Sigma, R^1_\Sigma, q_{10} \rangle$ and $\mathbb{Q}_2 := \langle Q_2, \Sigma, R^2_\Sigma, q_{20} \rangle$.

Definition A.6 *A unary relation $S \subseteq Q_1 \times Q_2$ is a* **strong bisimulation** *if $(q_1, q_2) \in S$ implies $(\forall \alpha \in \Sigma)$,*

(i) Whenever $q_1 \xrightarrow{\alpha} q_1'$ then $(\exists q_2' \in Q_2)$ $q_2 \xrightarrow{\alpha} q_2'$ and $(q_1', q_2') \in S$.

(ii) Whenever $q_2 \xrightarrow{\alpha} q_2'$ then $(\exists q_1' \in Q_1)$ $q_1 \xrightarrow{\alpha} q_1'$ and $(q_1', q_2') \in S$.

From [Mil89] we know that the set of strong bisimulation relations over $Q_1 \times Q_2$ is closed under union and thus there is always a largest strong bisimulation relation $\sim$, relating the states of $\mathbb{Q}_1$ and $\mathbb{Q}_2$. Note that

$$\sim := \cup\{S | S \text{ is a strong bisimulation over } Q_1 \times Q_2\}$$

We will often use infix notation and write $q_1 \sim q_2$ when $(q_1, q_2) \in \sim$.

We can now formally define *strong equivalence* for LTS. We will use $\sim$ to denote this binary relation over LTS as well as the largest strong bisimulation relation over the state sets of a pair of LTS. This should not cause any confusion as the relation we wish to refer to will usually be clear from the context in which $\sim$ is used.

Definition A.7 Strong Equivalence $\sim$: *Suppose* $\mathbb{Q}_1 := \langle Q_1, \Sigma, R^1_\Sigma, q_{10}\rangle$ *and* $\mathbb{Q}_2 := \langle Q_2, \Sigma, R^2_\Sigma, q_{20}\rangle$ *are LTS. Then*

$$\mathbb{Q}_1 \sim \mathbb{Q}_2 \textit{ iff } (\exists S \in \{\textit{strong bisimulations over } Q_1 \times Q_2\})\ (q_{10}, q_{20}) \in S$$

Thus $\mathbb{Q}_1 \sim \mathbb{Q}_2$ iff $q_{10} \sim q_{20}$, which is the reason $\sim$ is used to denote both the relation over LTS and the relation over the state sets of two LTS.

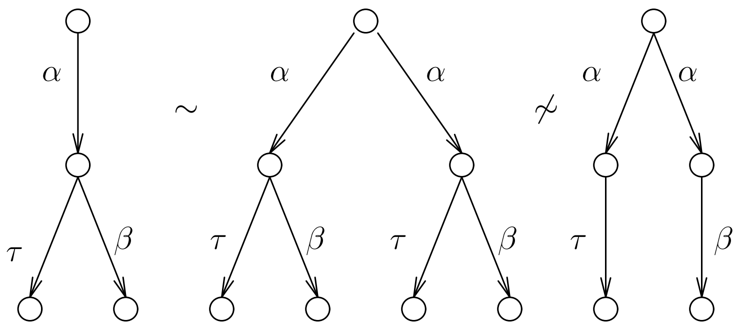


Figure A.5: Strong Equivalence Example

In Figure A.5, the first two LTS are strongly equivalent since they have the same choices after executing the same strings of events. The third LTS is not in the same equivalence class because there is no state reachable by executing α that can be continued by both β and τ. One can think of strongly equivalent systems deciding which futures will be possible at the same points in their execution. For example the third LTS chooses what transition will follow α when α actually occurs. The first and second systems are still free to choose the next transition *after* α happens. Having equivalent systems make their choices of possible future events at the same junctions in their execution eliminates the problems associated with mere string equivalence.

Figure A.6 demonstrates that strong equivalence is more discriminating than we would like because it "observes" τ. If we ignore τ transitions then after observing an α, both systems are in states that can be observably continued by β. To deal with this situation we now consider a weaker equivalence that is defined in a way that closely parallels strong equivalence.

Reducing the problem of TTM equivalence to one of LTS equivalence allows us to choose from the multitude of LTS equivalence relations in [DeN87]. For deadlock

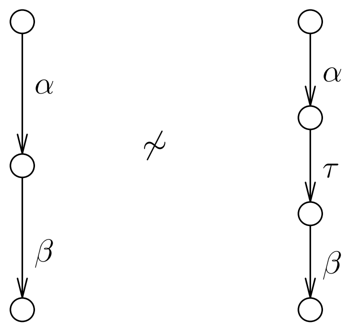


Figure A.6: Illustrating the need for a weaker equivalence

avoidance and other control properties described in [Law92], we will use Milner's weak observation equivalence (see [Mil89]). Recalling that $q \overset{\alpha}{\Rightarrow} q'$ denotes $q \overset{\tau^m \mu \tau^n}{\rightarrow} q'$ for some $m, n \in \mathbb{N}$ when $\alpha \neq \tau$ and q=q' or $q \overset{\tau^m \mu \tau^n}{\rightarrow} q'$ for $\alpha = \tau$, we now give the definition of Milner's observation equivalence.

Definition A.8 *Let $\mathbb{Q}_1 := \langle Q_1, \Sigma, R^1_\Sigma, q_{10} \rangle$ and $\mathbb{Q}_2 := \langle Q_2, \Sigma, R^2_\Sigma, q_{20} \rangle$ be LTS. A relation $S \subseteq Q_1 \times Q_2$ is a* **weak bisimulation** *if $(q_1, q_2) \in S$ implies for all $\alpha \in \Sigma$ (including $\alpha = \tau$),*

(i) Whenever $q_1 \overset{\alpha}{\rightarrow} q_1'$ then $(\exists q_2' \in Q_2)$ $q_2 \overset{\alpha}{\Rightarrow} q_2'$ and $(q_1', q_2') \in S$.

(ii) Whenever $q_2 \overset{\alpha}{\rightarrow} q_2'$ then $(\exists q_1' \in Q_1)$ $q_1 \overset{\alpha}{\Rightarrow} q_1'$ and $(q_1', q_2') \in S$.

In other words, two states $q_1 \in Q_1$, $q_2 \in Q_2$, are weakly bisimilar if any move from q_1 to a new state q_1' can be matched by a finite sequence of moves from q_2, that produces the same observation and leads to a state q_2' that is weakly bisimilar to q_1'. Also, any move from q_2 must be matched in a similar fashion. From [Mil89] we know that the set of weak bisimulation relations over $Q_1 \times Q_2$ is closed under union and thus there is always a largest weak bisimulation relation $\approx$, relating the states of $\mathbb{Q}_1$ and $\mathbb{Q}_2$. That is

$$\approx := \cup\{S | S \text{ is a weak bisimulation over } Q_1 \times Q_2\}$$

We write $q_1 \approx q_2$ when $(q_1, q_2) \in \approx$.

We can now formally define *weak observation equivalence* for LTS. We will use $\approx$ to denote both this binary relation over LTS and the largest weak bisimulation relation over the state sets of a pair of LTS.

Definition A.9 Weak Observation Equivalence $\approx$: *Let* $\mathbb{Q}_1 := \langle Q_1, \Sigma, R^1_\Sigma, q_{10}\rangle$ *and* $\mathbb{Q}_2 := \langle Q_2, \Sigma, R^2_\Sigma, q_{20}\rangle$ *be LTS. Then* $\mathbb{Q}_1 \approx \mathbb{Q}_2$ *iff there exists a weak bisimulation* S *over* $Q_1 \times Q_2$ *such that* $(q_{10}, q_{20}) \in S$

Thus $\mathbb{Q}_1 \approx \mathbb{Q}_2$ iff $a_0 \approx b_0$.

The relation $\approx$ is an equivalence relation over the set of LTS; the reader is referred to [Mil89] for the details in the setting of Milner's process algebra.

A.3 Equivalence Preserving Transformations

The purpose of this section is to explain transformations and their use. After demonstrating an intuitive notion of transformation with a simple example, we define a set of behavior preserving transformations and conclude by proving that these preserve the formal observation equivalence of TTMs.

A transformation is *behavior preserving* if it changes a TTM in such a way that the timed behavior of the transformed TTM restricted to the variables of interest, is equivalent (for a specified LTS equivalence relation) to the restricted timed behavior of the original TTM. Consider the two TTMs M_1 and M_2 of Figure A.7. Suppose

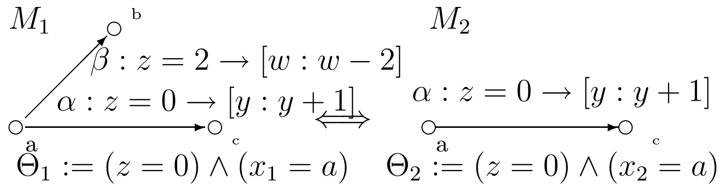

M_1
b
$\beta : z = 2 \rightarrow [w : w - 2]$
$\alpha : z = 0 \rightarrow [y : y + 1]$
a
c
$\Theta_1 := (z = 0) \wedge (x_1 = a)$
$\Longleftrightarrow$
M_2
$\alpha : z = 0 \rightarrow [y : y + 1]$
a
c
$\Theta_2 := (z = 0) \wedge (x_2 = a)$

Figure A.7: An example of Transition Addition/Transition Deletion

we are only interested in the timed behavior of the variables y and z. The initial condition Θ_1 prevents β from ever being enabled. If α has the same time bounds in both systems then it is apparent that M_1 and M_2 allow the same timed trajectories over y and z. In fact, since β is never enabled we could delete this transition from M_1 to *transform* M_1 into M_2. Similarly we could add a β transition to M_2 without changing its set of legal trajectories as the initial condition Θ_2 would prevent the new transition from ever occurring. Thus M_2 can also be transformed into M_1.

This is the idea behind the transformational technique of equivalence verification. Given a set of variables of interest $\mathcal{U}$, if it is possible to change one TTM into another by a set of behavior preserving transformations, then the two TTMs' timed behavior restricted to $\mathcal{U}$ will be equivalent (ie. $\mathbb{Q}_{M_1}|\mathcal{U} \approx \mathbb{Q}_{M_2}|\mathcal{U}$) and hence the TTMs will behave equivalently in a well defined sense. Clearly if our transformational method is correct, the transformations must abstract away unimportant details in such a way that the key features of the structure of $\mathbb{Q}_M|\mathcal{U}$ are preserved.

The addition of transition β to M_2 to form M_1 is an example of the Transition Addition transformation (**TA**). Going from M_1 to M_2 is an application of the dual of **TA**, the Transition Deletion transformation (**TD**). Below we describe these and the other transformation pairs needed to solve the verification problem of [LW95]. Throughout the section the transformations refer to the "set of variables of interest" $\mathcal{U}$. These are the variables we wish to "observe" so the transformations are designed to produce TTMs that generate equivalent timed behaviors when restricted to the variables in $\mathcal{U}$.

TA/TD Transition Addition/Transition Deletion: As demonstrated above one may add an instance of a transition to a TTM without changing its timed behavior if the transition's enablement condition is never satisfied in the new source activity. More formally, consider a TTM M with the transition $\alpha := (e, h, l, u)$, where α's full enablement condition from the transition graph of M is $e_\alpha := e \wedge (x = a_1 \vee x = a_2 \vee \ldots \vee x = a_n)$ implying that there are instances of α exiting activities $a_1, \ldots, a_n$ in the transition graph. One may add an instance of α exiting activity $a \notin \{a_1, \ldots, a_n\}$, with any other activity as its destination, provided that in any reachable state assignment q of M it is the case that $q(x) = a$ implies $q(e) = false$. The new full enablement condition for α after the transformation is $e_\alpha := e \wedge (x = a_1 \vee x = a_2 \vee \ldots \vee x = a_n \vee x = a)$

Similarly one can change the full enablement condition of the transition α from $e_\alpha := e \wedge (x = a_1 \vee x = a_2 \vee \ldots \vee x = a_n \vee x = a)$ to $e_\alpha := e \wedge (x = a_1 \vee x = a_2 \vee \ldots \vee x = a_n)$, thereby removing the instance of α exiting activity a in the transition graph of M if in all the reachable state assignments q of M it is the case that $q(x) = a$ implies $q(e) = false$. That is, one may remove an instance of a transition from a TTM if the transition's enablement condition is always false in the source activity from which the instance of the transition will be deleted.

CA/CD Control Addition/Control Deletion: This transformation lets one add or remove a condition from a transition's enablement condition under certain conditions. Consider a transition α with $e_\alpha := e$ and let p be some first order predicate over the variables in $\mathcal{V}$. If whenever a source activity for α is entered, p is true (p is false), then $e_\alpha^{new} := e \wedge p$ ($e_\alpha^{new} := e \vee p$).

Conversely if $e_\alpha := e \wedge p$ ($e_\alpha := e \vee p$) and, in every activity that α exits, p is guaranteed to be true (false), then $e_\alpha^{new} := e$.

AM/AS Activity Merge/Activity Split: This transformation is defined only when the activity variable x is not in the set of variables of interest (ie. $x \notin \mathcal{U}$). The basic idea of this transformation is that two activities can be merged if they have the same future. Hence, two activities may be merged if they have the same exiting transitions going to the same destination activities. In the example of Figure A.8, the activity merge transformation changes δ's full enablement condition from $e_\delta := e \wedge (x = a_1 \vee x = a_2 \vee \ldots)$ to $e_\delta := e \wedge (x = a \vee \ldots)$. For the merged activity one must be careful to choose a name that differs from the remaining TTM activities.

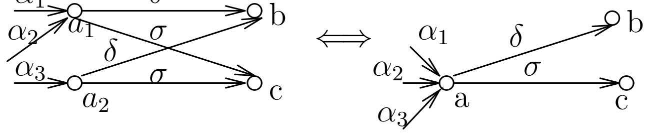


Figure A.8: Activity Merge/Activity Split

For activity splitting, if activity a is the destination activity of transitions $\alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n$ then split a into a_1 and a_2. $\alpha_1, \ldots, \alpha_k$ will have destination activity a_1 and $\alpha_{k+1}, \ldots, \alpha_n$ will have destination activity a_2. a_1 and a_2 will be the source activities for the same transitions to the same destination activities as in the case of activity a.

RT Rename Transition: This transformation is its own dual. It renames one or more instances of a transition with a new name provided the latter does not conflict with another name or change the structure of $\mathbb{Q}_M|\mathcal{U}$. Consider Figure A.9,

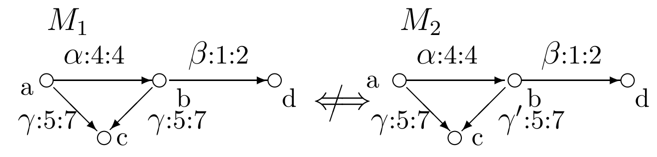


Figure A.9: A Problem with the Rename Transition Transformation

where one instance of the transition γ is renamed γ', altering the behavior of the system. If the numbers immediately following the transition labels denote the lower and upper time bounds respectively, it is apparent that transition γ is enabled across activities a and b in M_1 so although α happens before γ, γ has been enabled long enough that it can occur before β. On the other hand in M_2 γ is always preempted by α and γ' is always preempted by β.

In general, when it is possible for a transition to remain enabled when moving from one activity to another, then it is not possible to rename the two instances independently (ie. in any application of **RT** the two instances must be given the same name).

OM Operation Modification: If a variable does not occur in the enablement condition of any transition or the operations affecting any other variables and is not in $\mathcal{U}$, the set of variables of interest, then any operations affecting the variable can be added or deleted from any transition.

Let $v \in \mathcal{V} - \mathcal{U}$ and $P_{\mathcal{V}-\{v\}} : \mathcal{Q} \rightarrow \mathcal{Q}_{\mathcal{V}-\{v\}}$ be the natural projection from the state assignments to state assignments over $\mathcal{Q}_{\mathcal{V}-\{v\}}$. Then the OM transformation is defined if for all $\alpha \in \mathcal{T}$, the enablement condition e_α of α is independent of v, and $\ker(P_{\mathcal{V}-\{v\}}) \subseteq \ker(P_{\mathcal{V}-\{v\}} \circ h_\alpha)$ (ie. there exists an induced operation function $\overline{h_\alpha} : \mathcal{Q}_{\mathcal{V}-\{v\}} \rightarrow \mathcal{Q}_{\mathcal{V}-\{v\}}$ such that $\overline{h_\alpha} \circ P_{\mathcal{V}-\{v\}} = P_{\mathcal{V}-\{v\}} \circ h_\alpha$). If v satisfies these conditions then for any $\alpha \in \mathcal{T}$, h_α can be replaced with the $\overline{h_\alpha}$ induced by $P_{\mathcal{V}-\{v\}}$.

The rationale behind this transformation is that the value of v has no effect upon how the TTM operates on the variables in $\mathcal{U}$, hence we can set v to any value we wish, or ignore it altogether.

WM/WS (Wait Merge/Wait Split): A commonly occurring transition is the "wait" transition that serves the purpose of marking the passing of a fixed number of clock ticks. This transformation is a statement of the intuitive notion that waiting for n ticks and then waiting for m ticks is equivalent to waiting for n+m ticks. For technical reasons we require that $n \geq 1$.

$$\begin{tikzcd} \circ \arrow[r, "\omega_n"] & \circ \arrow[r, "\omega_m"] & \circ & \Longleftrightarrow & \circ \arrow[r, "\omega_{n+m}"] & \circ \end{tikzcd}$$

where $\omega_i := (True, [\,], i, i)$

Figure A.10: Wait Merge/Wait Split

Theorem A.10 *Let $M := \langle \mathcal{V}, \Theta, \mathcal{T} \rangle$ be a TTM, $\mathcal{U} \subset \mathcal{V}$ and $\mathbb{T}$ be one of the TTM transformations of subsection A.3. If $\mathbb{T}(M)$ is defined, then $M|\mathcal{U} \approx \mathbb{T}(M)|\mathcal{U}$.*

More transformations can be added to those listed in subsection A.3. One has to verify that each new transformation preserves observation equivalence. Theorem A.10 implies that any TTM derived from another TTM via a finite sequence of the transformations of subsection A.3 is observationally equivalent to the original TTM.

A.4 Limitations of Transformations

In [Law92] the set of transformations of Section A.3 is shown to be incomplete for proving observation equivalence of TTMs and it is further demonstrated that no finite set of transformations is complete for proving observation equivalence of general

TTMs. The proof closely follows a similar proof in Milner's process algebra [Mil89]. As in Milner's setting, the incompleteness property does not prevent the theory from being potentially useful in many practical applications. Indeed the exponential state explosion that occurs with the addition of new variables can make exhaustive verification routines impractical for even finite state TTMs. Thus heuristic methods such as the transformational technique introduced in this appendix provide a useful method of real-time system verification. Also, the transformations may be used to synthesize an implementation from a specification that is correct by construction. That is, the implementation resulting from the transformations will be guaranteed to be observationally equivalent to the specification, thereby eliminating the need to perform an exhaustive equivalence verification.

Appendix B

Equivalence Verification of the DRT

We now solve the DRT verification problem by applying the transformations described in Appendix A to formally check the equivalence of PROG and SPEC over the set of variables of interest $\mathcal{U} := \{Power, Pressure, Relay, t\}$. The inclusion of t in $\mathcal{U}$ guarantees that the timing as well as the ordering of changes to the inputs and outputs of the two systems will be the same. In future we will omit t since we are dealing with real-time systems and so the timing is assumed to always be of interest.

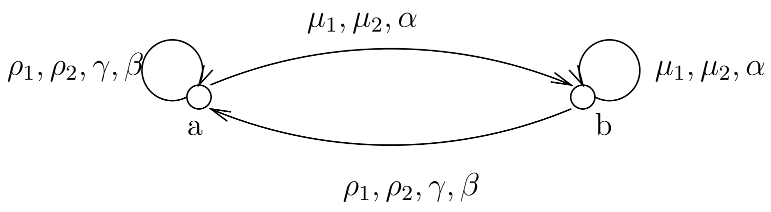
The proof that follows relies upon the system designer's intuition in places for the choice of the next applicable transformation. Starting from PROG with SPEC as our final goal, we progressively try to make PROG look more like SPEC until, if all goes well, we are left with a copy of SPEC at the end. At each step we check that the desired transformation is applicable and describe its effect. Starting with $PROG_0$:=PROG, at each step i we apply a transformation to $PROG_{i-1}$ to obtain $PROG_i$.

Claim B.1 *PROG is behaviorally equivalent to SPEC over* $\mathcal{U}$ *(ie.* $PROG \approx /\mathcal{U}SPEC$*).*

Proof: (PROG $\longrightarrow$ SPEC Over $\mathcal{U}$)

0. The original PROG TTM is shown in Figure 5.5.

1. **AS** The first transformation to be applied is the Activity Split transformation. We only have to make sure that instances of every transition exit both of the new activities. Since all the transitions are self-looped to the only activity in the



μ_1, μ_2, α
$\rho_1, \rho_2, \gamma, \beta$
μ_1, μ_2, α
a
b
$\rho_1, \rho_2, \gamma, \beta$

$\Theta_{new} := \Theta_{PROG} \wedge x = a$

Figure B.1: TTM for $PROG_1$

original system, we have some choice over how we distribute their destination activities. The reasons for the choice shown in Figure B.1 will become apparent in the next few steps. Although at this point it does not matter, we choose activity a as the initial activity for reasons that again will become clear later.

2. **TD** The Transition Deletion Transformation is applied next to remove the instance of the transition β exiting activity b and the instances of transitions α and ρ_1 exiting activity a. We are justified in these actions since:

   i. All transitions entering b increment either c_1 or c_2.

   ii. All transitions either increment or reset c_i to 0 so in any activity $c_i \geq 0$.

   iii. All transitions entering a either set $c_1 = 0$ or leave c_1 unaffected while requiring $c_1 = 0$ in their enablement conditions. Therefore in activity a $c_1 = 0$.

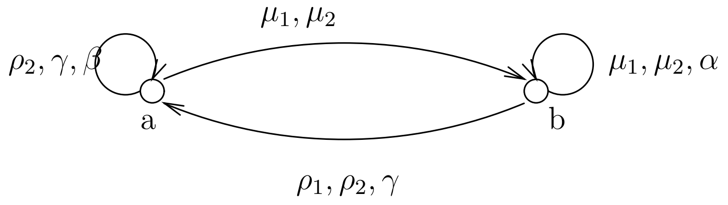
   Hence by (i), (ii) and (iii) we know that:

$$\begin{aligned} x = b &\Rightarrow c_1 > 0 \vee c_2 > 0 \\ &\Rightarrow e_\beta = false \end{aligned}$$

and

$$x = a \Rightarrow e_\alpha = e_{\rho_1} = false$$

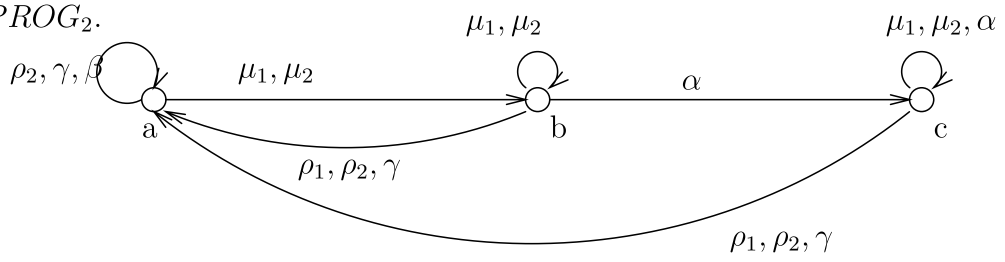
Thus we are justified in deleting the β exiting activity b and the α and ρ_1 exiting activity a.



$\Theta_{new} := \Theta_{PROG} \wedge x = a$

Figure B.2: TTM for $PROG_2$

3. AS This time we split activity b with α exiting b to the newly formed c activity (see Figure B.3). This is an effort to make the transformed PROG look more like SPEC. Notice that in splitting activity b into b and c we have not altered the dynamics. Both b and c have the same possible futures as activity b in $PROG_2$.



$\Theta_{new} := \Theta_{PROG} \wedge x = a$

Figure B.3: TTM for $PROG_3$

4. TD Upon entering activity c we know that $c_1 = 0 \wedge c_2 > 0$ since $h_\alpha := [c_1 : 0, c_2 : c_2 + 1, Relay : \text{OPEN}]$ and by 2(ii) we know that $c_i \geq 0$ in activity b. But e_{μ_1} requires that either $c_1 = c_2 = 0$ or $1 \leq c_1 \leq 29$, so e_{μ_1} is initially false in activity c. Also the other transitions entering c (α and μ_2) leave c_1 unaltered and only increment c_2. Hence

$$\begin{aligned} x = c &\implies c_1 = 0 \wedge c_2 > 0 \\ &\implies e_{\mu_1} = e_{\rho_1} = e_\alpha = false \end{aligned}$$

Conclusion: delete the instances of μ_1, ρ_1, and α with source activity c.

5. TD The initial condition Θ_{new} starts $PROG_4$ out in activity a with $c_2 = 0$. The only transitions that affect c_2 are μ_2 and α. Transition μ_2 requires $c_2 > 0$ to occur. Hence, starting from the initial state, α must precede μ_2. Once α has occurred we have x=c with only ρ_2 and γ exiting c. Both these transitions set $c_2 = 0$ so (i)$c_2 > 0$ iff x=c. Thus

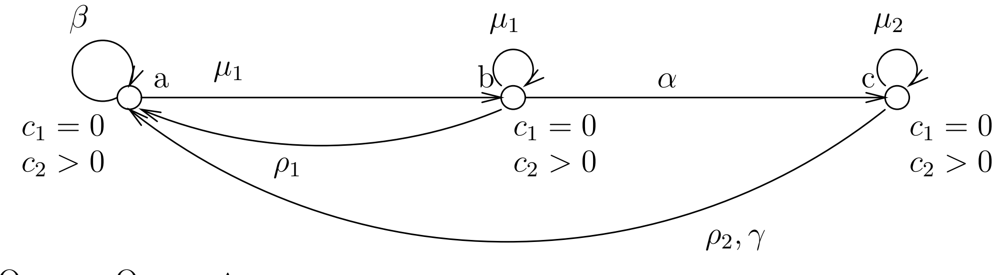
$$(i) \implies (e_{\mu_2} = true \iff x = c)$$

also

$$e_{\rho_2} = true \iff x = c$$

$$e_{\gamma} = true \iff x = c$$

Conclusion: delete all instances of μ_2, ρ_2 and γ except those with source activity



$\Theta_{new} := \Theta_{PROG} \wedge x = a$

Figure B.4: TTM for $PROG_5$

c. This leaves us with $PROG_5$ as shown in Figure B.4. The range of values that c_1 and c_2 take on in each activity can be easily deduced from 5(i) and 2(iii) so we include this information in Figure B.4 as well.

6. RT Referring to Figure B.4, we can rename the instance of μ_1 exiting activity a without affecting the dynamics of the variables of interest because μ_1 is now the only transition entering activity b and after a transition occurs, if it remains enabled, its time bounds are reset. This means that a problem like that illustrated in **RT** Figure A.9 cannot occur as a result of renaming only one of the instances of μ_1 since its time bounds are not carried across any group of activities. The new transition exiting a will be called μ. Of course $\mu := \mu_1$.

7. CD Considering the enablement conditions in $PROG_6$ we have:

$$e_\mu := e_{\mu_1} := \underbrace{(Power \geq \text{PT} \wedge Pressure \geq \text{DSP} \wedge c_1 = c_2 = 0)}_{p} \vee \underbrace{(1 \leq c_1 \leq 29)}_{q}$$

When x=a by 2(iii) and 5(i) we know that $c_1 = c_2 = 0$ and when x=b by 2(ii) we know that $c_1 > 0$. This gives:

$$\begin{aligned} (i)x = a &\implies q = false \\ (ii)x = b &\implies p = false \end{aligned}$$

Using the Control Deletion transformation with (i) as justification, we can change e_μ to $e_\mu^{new} := p$. Similarly using (ii) and Control Deletion again we obtain $e_{\mu_1}^{new} := q$.

Now consider e_μ^{new} and e_β:

$$\begin{aligned} e_\mu^{new} &:= Power \geq \text{PT} \wedge Pressure \geq \text{DSP} \wedge c_1 = c_2 = 0 \\ e_\beta &:= Power < \text{PT} \wedge c_1 = c_2 = 0 \\ \text{but} & \\ x = a &\implies c_1 = c_2 = 0 \end{aligned}$$

Applying **CD** yet again to simplify further we now have:

$$\begin{aligned} e_\mu^{new} &:= Power \geq \text{PT} \wedge Pressure \geq \text{DSP} \\ e_\beta^{new} &:= Power < \text{PT} \end{aligned}$$

In a similar fashion, again applying **CD**:

$$\begin{aligned} x = c &\implies c_1 = 0 \\ \text{so} & \\ e_{\rho_2}^{new} &:= Power < \text{PT} \wedge c_2 \geq 20 \end{aligned}$$

$$e_{\gamma}^{new} \quad := \quad Power \geq \mathrm{PT} \wedge c_2 \geq 20$$

8. AS The activity b is now split into thirty different activities with μ_1 taking the TTM from one new b activity to the next as c_1 is incremented. After μ occurs we are in an activity where $c_1 = 1$. μ_1 takes us to the next activity where $c_1 = 2$

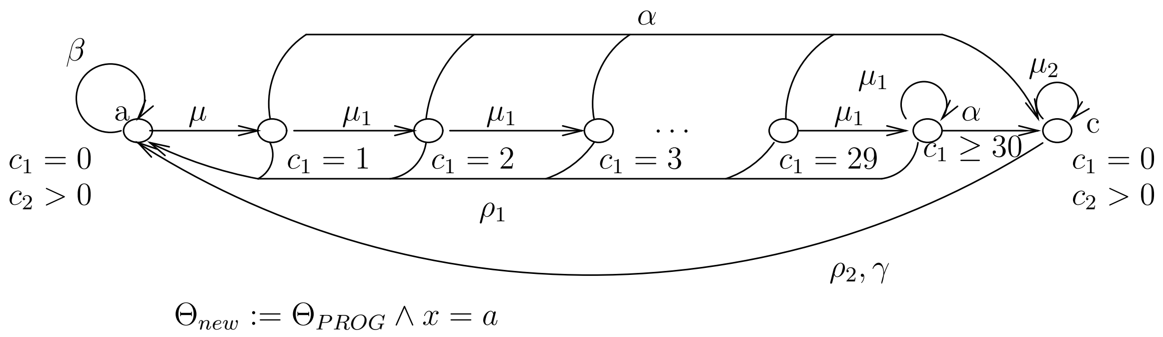


Figure B.5: TTM for $PROG_8$

and so on until we reach an activity where $c_1 \geq 30$ and μ_1 is self-looped. For each value of c_1 between 1 and 29, b has been spit into a new activity, with an additional activity for $c_1 \geq 30$. We are attempting to 'press out' the TTM's dependence on c_1 by flattening out the TTM to a point where for each value of c_1 between 1 and 30 there is an individual activity. Again note that we are in no way changing the dynamics of the system over $\mathcal{U}$ as the same transitions exit each of the new activities.

9. TD Knowing the value of c_1 in each of the newly added activities allows us to delete all instances of α and ρ_1 except for the activity where $c_1 \geq 30$ since both transitions enablement conditions require $c_1 \geq 30$. Also, the μ_1 transition self-looped at the $c_1 \geq 30$ activity may be removed because $e_{\mu_1} := (1 \leq c_1 \leq 29)$ in $PROG_8$.

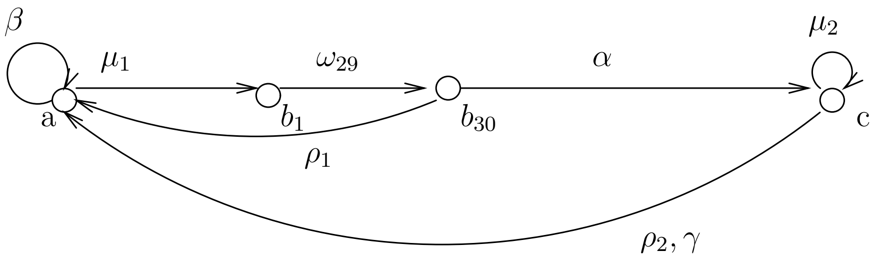
10. CD Now that transition μ_1 has as source activities only those activities for which $1 \leq c_1 \leq 29$, e_{μ_1} is always true in any of μ_1's source activities. Thus we can remove the c_1 dependence from e_{μ_1}. The same can be done for α and ρ_1 giving:

$$\begin{aligned} e_{\mu_1}^{new} &:= true \\ e_{\alpha}^{new} &:= Power \geq \text{PT} \\ e_{\rho_1}^{new} &:= Power < \text{PT} \end{aligned}$$

11. OM Variable c_1 no longer occurs in any transition's enabling conditions or operation functions that affect other variables. Hence we can drop the variable from all transition operations. The modified transitions are

$$\begin{aligned} \mu^{new} &:= (Power \geq \text{PT} \wedge Pressure \geq \text{DSP}, [\,], 1, 1) \\ \mu_1^{new} &:= (true, [\,], 1, 1) = \omega_1 \end{aligned}$$

12. WM All occurrences of μ_1 are now merged into one ω_{29} by the Wait Merge transformation (See Figure B.6).



$\Theta_{new} := Relay = \text{CLOSED} \wedge c_2 = 0 \wedge x = a$

Figure B.6: TTM for $PROG_{12}$

13-17. Now repeat steps 8-12 for activity c and transition μ_2 to map out the dynamics of variable c_2 and we have the desired result, a TTM identical to SPEC.

□

By transforming PROG into SPEC above we have shown that the corrected pseudocode implements an algorithm that satisfies the behavior requirements expressed by SPEC. The transformational proof above is sufficient to guarantee the formal observation equivalence of SPEC and PROG over U.

Bibliography

[ACD90] R. Alur, C. Courcoubetis, and D. Dill. Model-checking for real-time systems. In *Proc. of the 5th IEEE Symposium on Logic in Computer Science*, pages 414–425, 1990.

[AHU83] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *Data Structures and Algorithms.* Addison-Wesley, 1983.

[AM75] M. A. Arbib and E. G. Manes. *Arrows, Structures and Functors: The Categorical Imperative.* Academic Press, 1975.

[Arn94] A. Arnold. *Finite Transition Systems.* Prentice Hall, 1994.

[BBCS92] S. Bensalem, A. Bouajjani, C.Loiseaux, and J. Sifakis. Property preserving simulations. In *Proc. of 4th Conf. on Computer Aided Verification*, number 663 in LNCS, pages 260–275. Springer-Verlag, 1992.

[BC89] B. Bolognesi and M. Caneve. Equivalence verification: Theory, algorithms and a tool. In C. Vissers P. van Eijk and M. Diaz, editors, *The Formal Description Technique LOTOS*, pages 303–326. North-Holland, 1989.

[BCG87] M.C. Brown, E.M. Clarke, and O. Grümberg. Characterizing kripke structures in temporal logic. In G. Levi H. Erhig, R. Kowalski and U. Montanari, editors, *TAPSOFT'87, vol. I*, number 249 in LNCS, pages 256–270. Springer-Verlag, 1987.

[BCM92] J.R. Burch, E.M. Clarke, and K.L. McMillan. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98:142–170, 1992.

[BFH+92] A. Bouajjani, J.-C. Fernandez, N. Halbwachs, P. Raymond, and C. Ratel. Minimal state graph generation. *Science of Computer Programming*, 18:247–269, 1992.

[BH93] Y. Brave and M. Heymann. Control of discrete event systems modeled as hierarchical state machines. *IEEE Trans. Autom. Control*, 38:1803–1819, December 1993.

[BS81] S. Burris and H. P. Sankappanavar. *A Course in Universal Algebra*. Springer-Verlag, 1981.

[BW94] B. Brandin and W.M. Wonham. Supervisory control of timed discrete-event systems. *IEEE Trans. Autom. Control*, 39(2):329–342, Feb 1994.

[CE81] E.M. Clarke and E.A. Emerson. Synthesis of synchronization skeletons for branching time temporal logic. In *IBM Logic of Programs Workshop*, number 131 in LNCS, pages 52–71. Springer-Verlag, May 1981.

[CES86] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Programming Languages and Systems*, 8(2):244–263, April 1986.

[CGL94] E.M. Clarke, O. Grumberg, and D.E. Long. Model checking and abstraction. *ACM Trans. Programming Languages and Systems*, 16(5):1512–1542, September 1994.

[DeN87] R. DeNicola. Extensional equivalences for transition systems. *Acta Informatica*, 24:211–237, 1987.

[DGG94] D. Dams, O. Grümberg, and R. Gerth. Abstract interpretation of reactive systems: Abstraction preserving $\forall\mathrm{CTL}^*, \exists\mathrm{CTL}^*$ and CLT^*. In E.-R. Olderog, editor, *Programming Concepts, Methods and Calculi*, pages 573–592. North-Holland, 1994.

[EMSS92] E.A. Emerson, A.K. Mok, A.P. Sistla, and J. Srinivasan. Quantitative temporal reasoning. *Real-Time Systems*, 4:331–352, 1992.

[ES84] E.A. Emerson and A.P. Sistla. Deciding full branching time logic. *Information and Control*, 61:175–201, 1984.

[FG89] M. K. Franklin and A. Gabrielian. A transformational method for verifying safety properties in real-time systems. In *Proc. of 10th IEEE Real-Time Systems Symposium*, pages 112–123, December 1989.

[FZ91] J. Fa and Y. Zheng. Bi-observability of discrete event systems. In *Proc. of IFAC Workshop on Discrete Event System Theory and Applications in Manufacturing and Social Phenomena*, pages 71–74. International Academic Publishers, Schenyang, China, June 1991.

[GF91] A. Gabrielian and M.K. Franklin. Multilevel specification of real-time systems. *Communications of the ACM*, 34(5):50–60, May 1991.

[GL93] S. Graf and C. Loiseaux. Property preserving abstraction under parallel composition. In M.-C. Gaudel and J.-P. Jouannaud, editors, *TAPSOFT'93*, number 668 in LNCS, pages 644–657. Springer-Verlag, 1993.

[Har87] D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, 1987.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes.* International Series in Computer Science. Prentice-Hall International, Englewood Cliffs, NJ, 1985.

[Jos90] B. Josko. A context dependent equivalence relation between kripke structures. In *Proc. of 2nd Conf. on Computer Aided Verification*, number 531 in LNCS, pages 204–213. Springer-Verlag, 1990.

[Kai96] R. Kaivola. *Equivalences, Preorders and Compositional Verification for Linear Time Temporal Logic and Concurrent Systems.* PhD thesis, University of Helsinki, Department of Computer Science, Helsinki, Finland, 1996. Appears as Report A-1996-1.

[KS83] P.C. Kanellakis and S.A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. In *Proc. of 2nd ACM Symposium on the Principles of Distributed Computing*, pages 228–240, Montreal, Canada, August 1983. ACM.

[KV91] R. Kaivola and A. Valmari. Using truth-preserving reductions to improve the clarity of kripke-models. In *Proc. of CONCOUR'91*, number 527 in LNCS, pages 361–375. Springer-Verlag, 1991.

[KV92] R. Kaivola and A. Valmari. The weakest compositional semantic equivalence preserving nexttime-less linear temporal logic. In *Proc. of CONCOUR'92*, number 630 in LNCS, pages 207–221. Springer-Verlag, 1992.

[Law92] M.S. Lawford. Transformational equivalence of timed transition models. Master's thesis, Dept. of El. Eng., Univ. of Toronto, Canada, January 1992.

[Lee91] L. Lee. *The Day the Phones Stopped.* Donald I. Fine Inc., New York, 1991.

[Lio96] J.L. Lions and *et. al.* Rapport de la Commission d'enquête Ariane 501: Echec du vol Ariane 501. Communiqué de presse conjoint, ESA-CNES, Paris, France, 1996.

[LOW96] M. Lawford, J.S. Ostroff, and W.M. Wonham. Model reduction of modules for state-event temporal logics. In R. Gotzhein and J. Bredereke, editors, *Formal Description Techniques IX: Theory, application and tools, Proceedings of FORTE/PSTV'96*, pages 263–278. Chapman & Hall, 1996.

[LP85] O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *Proc. of 12th ACM Symposium* *on Principles of Programming Languages*, pages 97–107, New Orleans, January 1985.

[LW90] F. Lin and W.M. Wonham. Decentralized control and coordination of discrete-event systems with partial observation. *IEEE Trans. Autom. Control*, 35(12):1330–1337, December 1990.

[LW92] M. Lawford and W.M. Wonham. Equivalence preserving transformations for timed transition models. In *Proc. of 31st Conf. Decision and Control*, pages 3350–3356, Tucson, AZ, USA, December 1992.

[LW95] M. Lawford and W.M. Wonham. Equivalence preserving transformations of timed transition models. *IEEE Trans. Autom. Control*, 40:1167–1179, July 1995.

[McM92] K.L. McMillan. *Symbolic Model Checking.* Kluwer, 1992.

[Man94] Z. Manna and *et. al.* The Stanford Temporal Theorem Prover. Technical Report STAN-CS-TR-94-1518, Dept. of Computer Science, Stanford University, CA, USA, 1994.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS*. Springer-Verlag, New York, 1980.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice Hall, New York, 1989.

[MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems.* Springer-Verlag, New York, 1992.

[ON96] J.S. Ostroff and H.K. Ng. Verifying real-time systems using untimed tools. In *Proc. of 3rd AMAST Workshop on Real-Time Systems*, pages 132–146. ONR and Iowa University, Salt Lake City, Utah, March 1996.

[Ost89] J.S. Ostroff. *Temporal Logic for Real-Time Systems.* RSP. Research Studies Press / Wiley, 1989. Taunton, UK.

[Ost90] J.S. Ostroff. Deciding properties of timed transition models. *IEEE Trans. Parallel and Distributed Systems*, 1(2):170–183, April 1990.

[Ost92] J.S. Ostroff. A verifier for real-time properties. *Real-Time Journal*, 4:5–35, 1992.

[Ost95] J.S. Ostroff. A CASE tool for the design of safety critical systems. In H. A. Müller and R. J. Norman, editors, *Proc. of CASE'95*, pages 370–380. IEEE Computer Society Press, July 1995.

[OW90] J.S. Ostroff and W.M. Wonham. A framework for real-time discrete event control. *IEEE Trans. Autom. Control*, 35(4):386–397, April 1990.

[PAM91] D.L. Parnas, G.J.K. Asmis, and J. Madey. Assesment of safety-critical software in nuclear power plants. *Nuclear Safety*, 32(2):189–198, 1991.

[Par81] D. Park. Concurrency and automata on infinite sequences. In *5th GI Conference on Theoretical Computer Science*, pages 167–183. Berlin, Germany: Springer-Verlag, 1981. LNCS-104.

[PT87] R. Paige and R.E. Tarjan. Three partition refinement algorithms. *SIAM J. of Computing*, 16(6):973–989, December 1987.

[RW87] P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Control Optim.*, 25(1):206–230, January 1987.

[Sah74] S. Sahni. Computationally related problems. *SIAM J. of Computing*, 3(3):262–279, 1974.

[SC85] A.P. Sistla and E.M. Clarke. The complexity of propositional linear temporal logic. *J. ACM*, 32:733–749, 1985.

[Val90] A. Valmari. A stubborn attack on state explosion. In *Proc. of 2nd Conf. on Computer Aided Verification*, number 531 in LNCS, pages 156–165. Springer-Verlag, 1990.

[Wan91] Y. Wang. *CCS + Time = an Interleaving Model for Real Time Systems*, volume 510 of *LNCS*, pages 217–228. Springer–Verlag, 1991.

[Won76] W.M. Wonham. Towards an abstract internal model principle. *IEEE Trans. Systems Man and Cybernetics*, 6(11):730–752, November 1976.

[Won94] K.C. Wong. *Control Architecture of Discrete-Event Systems: An Algebraic Approach*. PhD thesis, Dept. of El. Eng., Univ. of Toronto, Canada, June 1994.

[WW92] K.C. Wong and W.M. Wonham. Hierarchical and modular control of discrete-event systems. In *Proc. of 30th Allerton Conference on Communication, Control and Computing*, pages 614–623, Champaign, IL, USA, September 1992.

[Zha96] Y. Zhang. Software for state-event observation theory and its application to supervisory control. Master's thesis, Dept. of El. Eng., Univ. of Toronto, Canada, Canada, July 1996.

[ZW90] H. Zhong and W.M. Wonham. On the consistency of hierarchical supervision in discrete-event systems. *IEEE Trans. Autom. Control*, 35(10):1125–1134, October 1990.