Hierarchical Interface-based Supervisory Control: Bi-level Systems

Version 1.1

by

R.J. Leduc, W.M. Wonham, and M. Lawford[1]

Systems Control Group Report No: 0103
University of Toronto
July 6, 2005

[1]Dept. of Computing and Software, McMaster University

Abstract

In this report we present a hierarchical method that decomposes a system into a *high level subsystem* which communicates with $n \geq 1$ parallel *low level subsystems* through separate interfaces, which restrict the interaction of the subsystems. We first define the setting for the serial case ($n = 1$), and then generalise it for $n \geq 1$. We present a definition for an interface, and define a set of interface consistency properties that can be used to verify if a discrete-event system (DES) is nonblocking and controllable. Each clause of the definition can be verified using a single subsystem; thus the complete system model never needs to be constructed, offering significant savings in computational effort. Additionally, the development of clean interfaces facilitates re-use of the component subsystems.

Contents

List of Figures

Chapter 1

Introduction

In the area of Discrete-Event Systems (DES), two common tasks are to verify that a composite system, based on a cartesian product of subsystems, is (i) nonblocking and (ii) controllable. The main obstacle to performing these tasks is the combinatorial explosion of the product state space. Although many methods have been developed to deal with this problem (modular control [1, 13, 20, 25], decentralised control [14, 22, 27], model aggregation methods [2, 3, 6, 24, 30], and multi-level hierarchy [5, 9, 15, 16, 23, 31]), large-scale systems are still problematic, particularly for verification of nonblocking.

To deal with the complexity of large scale systems, the software engineering community has long advocated the decomposition of software into modules (components) that interact via well defined interfaces (e.g., [18, 17, 19]). Recently the supervisory control community has begun to advocate a similar approach [11, 8]. These approaches develop well defined interfaces between components to provide the structure to allow local checks to guarantee global properties such as controllability [8] or nonblocking [11].

In this report, we present an interface-based hierarchical method to verify if a system is nonblocking and controllable. We describe the application of our method to bi-level systems where the system is split into a *high level subsystem* which interacts with $n \geq 1$ parallel *low level subsystems* via separate interface DES, which regulates the subsystems' interaction. The most significant feature that distinguishes the work from [8] is the results regarding nonblocking.

It is worth noting that Zhang et al. have recently developed algorithms that use Integer Decision Diagrams to verify centralized DES systems on the order of 10^{23} states [29, 28]. These results are complementary to the hierarchical method illustrated in this report, as their approach can be used to verify many of the required conditions, allowing HISC to scale to even larger systems.

In the remainder of this report we first describe the setting for the serial case ($n = 1$), which was introduced in [12]. We present a definition for an interface, and define a set of (local) consistency properties that can be used to verify if a discrete-event system is globally nonblocking and controllable. We then extend our definitions to the general case of $n \geq 1$. We present two full examples to illustrate our method, and then discuss the application of our method to a large manufacturing example with an estimated closed-loop state space size of 7×10^{21} (see [10]).

1.1 Supervisory Control Theory Preliminaries

Ramadge-Wonham supervisory control (RW theory:[21], [26], and [25]) provides a theoretical framework for the control of systems that are discrete in space and time. These systems are modelled as automata that generate a formal language of discrete events. These systems are customarily referred to as discrete event systems (DES). The DES are event-driven and may be non-deterministic[1]. The DES do not model when or why an event occurs, just the possible strings of events that the plant can generate. The events are considered to occur in an interleaving fashion. For a detailed discussion of Discrete-event Systems, please refer to [25]. Below, we present a summary of the terminology that we will be using in this report.

1.1.1 Generators

The DES automaton is represented as a 5-tuple as shown below.

$$\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$$

where Y is the state set (at most countable); Σ is a finite set of event labels (also referred to as the alphabet); δ is the transition function; $y_o \in Y$ is the initial state and $Y_m \subseteq Y$ is the subset of marker states. We will also use the notation $\Sigma_{\mathbf{G}}$ as a shorthand for the event set that DES $\mathbf{G}$ is defined over. This is an easy way to refer to the alphabet given in the 5-tuple definition of G, particularly in situations when it is not explicitly stated.[2] For DES $\mathbf{G}$ above, $\Sigma_{\mathbf{G}} = \Sigma$.

The transition function $\delta : Y \,\mathrm{x}\, \Sigma \rightarrow Y$ is a partial function and is only defined for a subset of Σ at a given $y \in Y$. The notation $\delta(y, \sigma)!$ indicates that δ is defined for σ at state y.

[1] Capable of choosing between two possible next states by chance or unmodelled system dynamics.

[2] For example, in the case of the DES created by the synchronous product operator.

We want to extend δ to operate on strings in Σ^*, where $\Sigma^* = \Sigma^+ \cup \{\epsilon\}$ where ϵ is the empty string and Σ^+ is the set of all sequences of symbols $\sigma_1\sigma_2\sigma_3\ldots\sigma_k$, $k \geq 1$ and $\sigma_i \in \Sigma$, $i = 1, 2, \ldots, k$. We now recursively extend δ to the partial function $\delta : Y \,\mathrm{x}\, \Sigma^* \rightarrow Y$ by applying the following rules for arbitrary $y \in Y$, $s \in \Sigma^*$ and $\sigma \in \Sigma$:

$$\begin{aligned} \delta(y, \epsilon) &= y \\ \delta(y, s\sigma) &= \delta(\delta(y, s), \sigma) \end{aligned}$$

as long as $y' := \delta(y, s)!$ and $\delta(y', \sigma)!$.

An example of a DES plant is given in Figure 1.1. Here, the plant is composed of two automata, **mach1** and **mach2**. The composite plant model is obtained by taking the synchronous product (defined below) of **mach1** and **mach2**. In the diagram, the entering arrow at state *0* of DES **mach1** indicates that this is the initial state. The exiting arrow indicates that this is a marked state. A transition or event in a DES $\mathbf{G}$ is a triple (y, σ, y') where $y, y' \in Y$, $\sigma \in \Sigma$, and $y' = \delta(y, \sigma)$. An example from DES **mach1** is the event $(0, \alpha_1, 1)$. We say an event $\sigma \in \Sigma$ is eligible in DES $\mathbf{G}$ at state $y \in Y$ if $\delta(y, \sigma)!$. For example, event α_1 is eligible at state **I1** in DES **mach1**.

For DES $\mathbf{G}$, the language generated, called the closed behaviour of $\mathbf{G}$, is denoted by $L(\mathbf{G})$, and is defined as follows:

$$L(\mathbf{G}) := \{s \in \Sigma^* |\ \delta(y_o, s)!\}$$

The marked behaviour of $\mathbf{G}$, $L_m(\mathbf{G})$, is defined as follows:

$$L_m(\mathbf{G}) := \{s \in \Sigma^* |\ \delta(y_o, s) \in Y_m\}$$

Clearly, $\emptyset \subseteq L_m(\mathbf{G}) \subseteq L(\mathbf{G})$ and $\epsilon \in L(\mathbf{G})$ as long as $\mathbf{G} \neq \mathbf{EMPTY}$ where **EMPTY** is the DES with an empty state set.

Finally, the reachable state subset of DES $\mathbf{G}$, denoted Y_r, is defined to be:

$$Y_r := \{y \in Y |\ (\exists s \in \Sigma^*)\, \delta(y_o, s) = y\}$$

A DES $\mathbf{G}$ is reachable if $Y_r = Y$. We will always assume $\mathbf{G}$ is reachable as unreachable states don't affect $L(\mathbf{G})$ or $L_m(\mathbf{G})$, and an equivalent reachable DES can always be constructed.

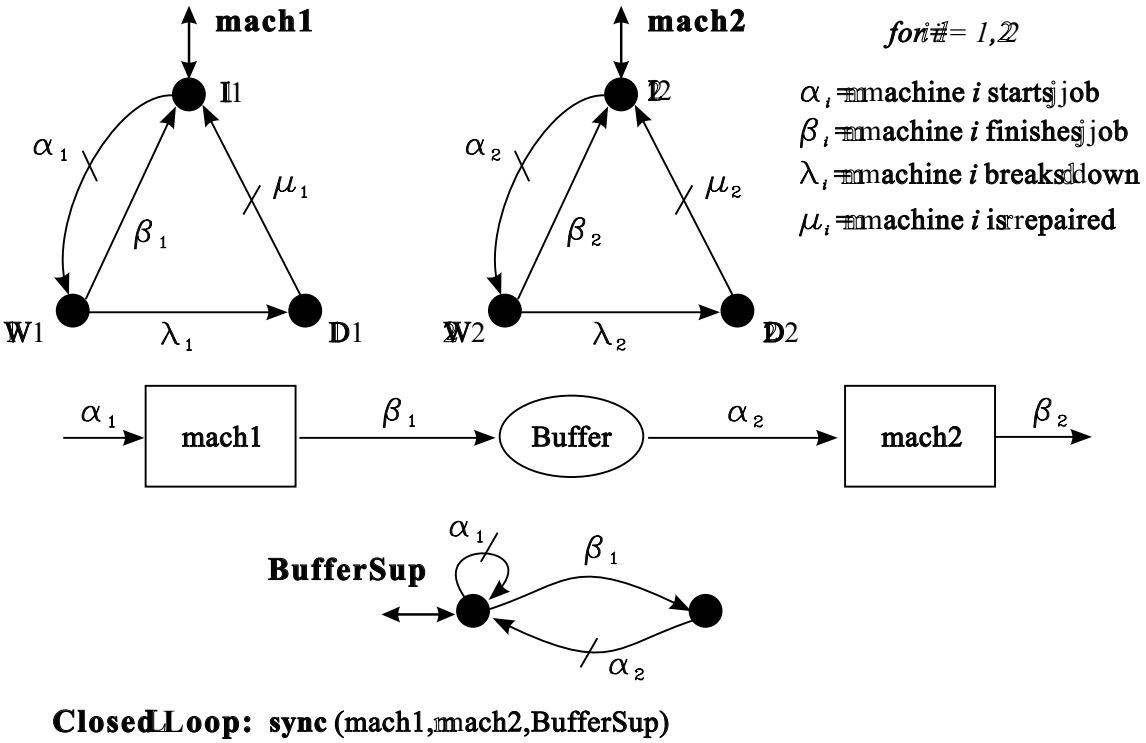


Figure 1.1: Simple Factory Example

1.1.2 Operations

In this section, we discuss some useful operations for languages and automata. We start with the **cat** operator. The catenation of strings is defined as follows: $\mathbf{cat} : \Sigma^* \text{ x } \Sigma^* \rightarrow \Sigma^*$ where:

$$\mathbf{cat}(\epsilon, s) = \mathbf{cat}(s, \epsilon) = s, \quad s \in \Sigma^*$$

$$\mathbf{cat}(s, t) = st \quad s, t \in \Sigma^+$$

This leads us to the **prefix closure** operator. A string $t \in \Sigma^*$ is a prefix of $s \in \Sigma^*$ if $s = \mathbf{cat}(t, u)$, for some $u \in \Sigma^*$. The relation "t is a prefix of s" is expressed as $t \leq s$. The prefix closure of a language $L \subseteq \Sigma^*$ is defined as:

$$\overline{L} = \{t \in \Sigma^* \mid t \leq s \text{ for some } s \in L\}$$

We say L is closed if $L = \overline{L}$. Clearly as the name indicates, $L(\mathbf{G})$ - the closed behaviour of DES $\mathbf{G}$, is closed.

The natural projection is defined with respect to the subset of a larger alphabet. Let $\Sigma_o \subseteq \Sigma^*$. We define the natural projection $\mathrm{P}_o : \Sigma^* \rightarrow \Sigma_o^*$ as follows:

$$\begin{aligned} \mathrm{P}_o(\epsilon) &= \epsilon \\ \mathrm{P}_o(\sigma) &= \begin{cases} \epsilon \text{ if } \sigma \notin \Sigma_o \\ \sigma \text{ if } \sigma \in \Sigma_o \end{cases} \\ \mathrm{P}_o(s\sigma) &= \mathrm{P}_o(s)\ \mathrm{P}_o(\sigma) \quad \text{where } s \in \Sigma^*, \sigma \in \Sigma \end{aligned}$$

Clearly, the natural projection is concatenative (i.e. $\mathrm{P}_o(ss') = \mathrm{P}_o(s)\mathrm{P}_o(s')$, where $s, s' \in \Sigma^*$). Also useful is the inverse image map of the natural projection and its extension to operate on sets, given below for $s \in \Sigma_o^*$ and $L \subseteq \Sigma_o^*$:

$$\begin{aligned} P_o^{-1}(s) &:= \{s' \in \Sigma^* \mid P_o(s') = s\} \\ P_o^{-1}(L) &:= \cup_{v \in L} P_o^{-1}(v) \end{aligned}$$

This brings us to the synchronous product of two languages $L_1 \subseteq \Sigma_1^*$ and $L_2 \subseteq \Sigma_2^*$ ($\Sigma = \Sigma_1 \cup \Sigma_2$) is defined using the natural projection. Let $\mathbf{P}_i$ be the natural projection of Σ^* onto Σ_i^*, $i = 1, 2$. The

synchronous product of L_1 and L_2 is defined to be:

$$L_1||L_2 = \mathbf{P}_1^{-1}(L_1) \cap \mathbf{P}_2^{-1}(L_2)$$

where $\mathbf{P}_i^{-1}$, $i = 1, 2$ is the inverse image map of the natural projections $\mathbf{P}_i$.

We can now define the $||_s$ operator for DES. For DES $\mathbf{G}_1 = (Y_1, \Sigma_1, \delta_1, y_{o_1}, Y_{m_1})$ and $\mathbf{G}_2 = (Y_2, \Sigma_2, \delta_2, y_{o_2}, Y_{m_2})$, the synchronous product is defined to be the reachable DES $\mathbf{G} = \mathbf{G}_1||_s\mathbf{G}_2 = (Y, \Sigma, \delta, y_o, Y_m)$ with the properties:

$$L_m(\mathbf{G}) = L_m(\mathbf{G_1})||L_m(\mathbf{G_2}), \quad L(\mathbf{G}) = L(\mathbf{G_1})||L(\mathbf{G_2}), \quad \Sigma = \Sigma_1 \cup \Sigma_2$$

where the natural projections $\mathbf{P}_i$, $i = 1, 2,$ are as defined above. The $||_s$ operator is essentially the same as the TCT **sync** operator (see [25]), except that the $||_s$ operator requires that the alphabets of DES G_i to be specified explicitly; the **sync** operator takes Σ_i to be the events that appear in G_i. Requiring the alphabets to be explicitly specified, ensures that $||_s$ is associative.

Finally, we define the eligibility operator. For a language $L \subseteq \Sigma^*$ and a string $s \in \Sigma^*$, the operator $\text{Elig}_L : \Sigma^* \to \text{Pwr}(\Sigma)$ is defined as below. The notation $\text{Pwr}(\Sigma)$ represents the power set of Σ (the set of all sets that are $\subseteq \Sigma$).

$$\text{Elig}_L(s) := \{\sigma \,|s\sigma \in L\}$$

1.1.3 Nonblocking and Controllability

For DES, the two main properties we want to check are nonblocking and controllability.

Nonblocking: A DES $\mathbf{G}$ is said to be nonblocking if the following is true:[3]

$$\overline{L_m(\mathbf{G})} = L(\mathbf{G})$$

Nonblocking means that every string in $L(\mathbf{G})$ can be completed to a marked string. This means that the DES can always return to a marked state. This is a method to check if the DES will deadlock.

To control the plant, we define a supervisor. Supervisors monitor the events generated by the plant, and disable events according to some control law. The supervisors are represented as automata and

[3] An equivalent condition is that every reachable state of $\mathbf{G}$ is coreachable.

defined as below:

$$\mathbf{S} = (X, \Sigma_S, \xi, x_o, X_m)$$

In [25], the closed loop behaviour of a plant $\mathbf{G}_1 = (Y_1, \Sigma_1, \delta_1, y_{o_1})$ under the control of supervisor $\mathbf{S}$ is achieved using the **meet** operator as below (assuming that $\Sigma_S = \Sigma_1$):

$$\mathbf{Closed_{meet}}(\mathbf{G}_1, \mathbf{S}) = \mathbf{meet}(\mathbf{G}_1, \mathbf{S})$$

From a practical point of view, using the **meet** operator can make specifying supervisors tedious and error prone; particularly when the event set is large, and the supervisor is specified as several modular supervisors that are then combined using the **meet** operator. As a supervisor must have the same event set as the plant, any event that the supervisor is indifferent to (doesn't care if it's enabled or not) must be selflooped (ie. as event α_1 is at the initial state of DES **BufferSup** in Figure 1.1) at every state. This creates a lot of clutter, and introduces the potential for error if a selfloop is missed.

Instead, we will use the synchronous product operator to specify the closed loop behaviour. When we specify a supervisor, we need only include the events it's concerned with in its event set. Particularly when using a graphical editor to specify and display the supervisor, this simplifies things. We thus define the closed loop behaviour of a plant $\mathbf{G}_1$ under the control of supervisor $\mathbf{S}$ as follows:

$$\mathbf{Closed}(\mathbf{G}_1, \mathbf{S}) = \mathbf{G}_1 ||_s \mathbf{S}$$

An oddity of using the synchronous product operator to specify the closed loop behaviour is that an event could be in the supervisor, but not in the plant. This may seem unintuitive, but it is useful for events that are not part of the original plant, but are artificially added to aid in the synchronization of supervisors.[4] One could create a new plant DES with such events selflooped at the initial state, but this would make things more cluttered, and possibly increase memory usage in DES analysis software. By using the synchronous product, the result is as if we created this new plant DES automatically.

We now introduce the concept of controllability. Controllability is a way to check if the behaviour restrictions specified by a supervisor are achievable. For the alphabet of interest, we partition it into two disjoint sets: Σ_u and Σ_c. These are the sets of *uncontrollable* and *controllable events,* respectively. Controllable events are events that a supervisor can disable, and thus prevent from occurring. Uncon-

[4] An example would be when one supervisor needs to wait until the other reaches a particular state, but there doesn't exist already an event that would signal this uniquely.

trollable events can't be disabled. Informally, a supervisor is controllable for a given plant if the plant can't leave the behaviour specified by the supervisor by means of an uncontrollable event.

We now present a formal definition for controllability. We first give the more standard definition with respect to the **meet** operator, again assuming that $\Sigma_S = \Sigma_1$.

Controllability (meet): A supervisor $\mathbf{S}$ is *controllable* for a plant $\mathbf{G}_1$ if:

$$L(\mathbf{S})\Sigma_u \cap L(\mathbf{G}_1) \subseteq L(\mathbf{S})$$

We will now present the version with respect to the synchronous product operator. First we need to define the event set Σ, the natural projections P_1 and P_S, and languages L_{G_1} and L_S as below:

$$\begin{aligned} \Sigma &:= \Sigma_1 \cup \Sigma_S \\ P_1 &: \Sigma^* \to \Sigma_1^* \\ P_S &: \Sigma^* \to \Sigma_S^* \\ L_{G_1} &:= P_1^{-1} L(\mathbf{G}_1) \\ L_S &:= P_S^{-1} L(\mathbf{S}) \end{aligned}$$

Controllability ($||_s$): A supervisor $\mathbf{S}$ is *controllable* for a plant $\mathbf{G}_1$ if:

$$L_S\Sigma_u \cap L_{G_1} \subseteq L_S$$

In this report, whenever we refer to controllability, we will be referring to the version with respect to synchronous product operator. We will actually use the following equivalent definition:

Alternative Controllability Definition ($||_s$): A supervisor $\mathbf{S}$ is controllable for a plant $\mathbf{G}_1$ if:

$$(\forall s \in L_{G_1} \cap L_S)\ \mathrm{Elig}_{L_{G_1}}(s)\ \cap\ \Sigma_u \subseteq \mathrm{Elig}_{L_S}(s)$$

Modular supervisors are implemented by taking the conjunction of two or more supervisors. We define the conjunction of two supervisors $\mathbf{S_1}$ and $\mathbf{S_2}$ (expressed as $\mathbf{S_1} \wedge \mathbf{S_2}$) as follows:

$$\mathbf{S_1} \wedge \mathbf{S_2} = \mathbf{S_1} ||_s \mathbf{S_2}$$

Returning to the example in Figure 1.1, our plant is $G = \mathbf{mach1}||_s\mathbf{mach2}$ and our supervisor is $S = \mathbf{BufferSup}$. Our event partition can be determined from the diagram by noting that transitions with a bar across them (such as α_1 and α_2) indicate that these are controllable events. Finally, our closed loop system is $\mathbf{Closed(S,G)} = (\mathbf{mach1}||_s\mathbf{mach2})||_s\mathbf{BufferSup} = \mathbf{mach1}||_s\mathbf{mach2}||_s\mathbf{BufferSup}$.

1.1.4 Related Propositions

In this section, we present some language propositions that will be used in later chapters. The propositions will refer to alphabets Σ_1, Σ_2, and $\Sigma := \Sigma_1 \cup \Sigma_2$, languages $L_1, L_1' \subseteq \Sigma_1^*$, and $L_2, L_2' \subseteq \Sigma_2^*$, and natural projections $P_i : \Sigma^* \to \Sigma_i^*$, where $i = 1, 2$. Finally, we apply these propositions to the synchronous product of DES $\mathbf{G}_1 = (Y_1, \Sigma_1, \delta_1, y_{o_1}, Y_{m_1})$ and $\mathbf{G}_2 = (Y_2, \Sigma_2, \delta_2, y_{o_2}, Y_{m_2})$.

The first two propositions are useful for working with the languages of a DES created by the synchronous product operator. The first proposition essentially says that the inverse natural projection of a closed language is also closed. The second proposition essentially says that the set of prefix closed subsets of Σ^* are closed under intersection.

Proposition 1 *If L_1 is closed, then $P_1^{-1}(L_1)$ is closed.*

Proof:

Assume L_1 is closed. **(1)**

We will now show this implies that $P_1^{-1}(L_1)$ is closed.

This means showing that $P_1^{-1}(L_1) = \overline{P_1^{-1}(L_1)}$

It is sufficient to show that $P_1^{-1}(L_1) \subseteq \overline{P_1^{-1}(L_1)}$ and $\overline{P_1^{-1}(L_1)} \subseteq P_1^{-1}(L_1)$. As $P_1^{-1}(L_1) \subseteq \overline{P_1^{-1}(L_1)}$ is automatic, all that remains to show is $\overline{P_1^{-1}(L_1)} \subseteq P_1^{-1}(L_1)$.

Let $s \in \overline{P_1^{-1}(L_1)}$ **(2)**

We will now show this implies that $s \in P_1^{-1}(L_1)$

We first note that $s \in \overline{P_1^{-1}(L_1)}$ implies that $(\exists s' \in \Sigma^*)\, ss' \in P_1^{-1}(L_1)$

$\Rightarrow P_1(ss') = P_1(s)P_1(s') \in L_1$

$\Rightarrow P_1(s) \in \overline{L_1}$

$\Rightarrow P_1(s) \in L_1$ by **(1)**

$\Rightarrow s \in P_1^{-1}(L_1)$, as required.

We thus have $\overline{P_1^{-1}(L_1)} \subseteq P_1^{-1}(L_1)$, and thus $P_1^{-1}(L_1) = \overline{P_1^{-1}(L_1)}$.

We thus conclude that $P_1^{-1}(L_1)$ is closed.

QED

Proposition 2 *If L_1 and L_2 are closed, then $L_1 \cap L_2$ is closed.*

Proof:

Assume L_1 and L_2 are closed. **(1)**

We will now show this implies that $L_1 \cap L_2$ is closed.

This means showing that $L_1 \cap L_2 = \overline{L_1 \cap L_2}$

It is sufficient to show that $L_1 \cap L_2 \subseteq \overline{L_1 \cap L_2}$ and $\overline{L_1 \cap L_2} \subseteq L_1 \cap L_2$. As $L_1 \cap L_2 \subseteq \overline{L_1 \cap L_2}$ is automatic, all that remains to show is $\overline{L_1 \cap L_2} \subseteq L_1 \cap L_2$.

Let $s \in \overline{L_1 \cap L_2}$ **(2)**

We will now show this implies that $s \in L_1 \cap L_2$

By **(2)**, we can conclude $(\exists s' \in \Sigma^*)\, ss' \in L_1 \cap L_2$

$\Rightarrow s \in \overline{L_1} \cap \overline{L_2}$

$\Rightarrow s \in L_1 \cap L_2$, by **(1)**

We thus have $\overline{L_1 \cap L_2} \subseteq L_1 \cap L_2$, and thus $L_1 \cap L_2 = \overline{L_1 \cap L_2}$, as required.

We thus conclude that $L_1 \cap L_2$ is closed.

QED

We now present a corollary that combines the above two propositions to get a similar result for the || operator.

Corollary 1 *If L_1 and L_2 are closed, then $L_1||L_2$ is closed.*

Proof:

Assume L_1 and L_2 are closed. **(1)**

We will now show this implies that $L_1||L_2$ is closed.

We first note that $L_1||L_2 = \mathbf{P}_1^{-1}(L_1) \cap \mathbf{P}_2^{-1}(L_2)$

From **(1)** and applying **Proposition 1**, we can conclude that languages $\mathbf{P}_1^{-1}(L_1)$ and $\mathbf{P}_2^{-1}(L_2)$ are closed.

We can now apply **Proposition 2** and conclude that $\mathbf{P}_1^{-1}(L_1) \cap \mathbf{P}_2^{-1}(L_2)$, as required.

QED

The next proposition says that the inverse natural projection respects subset ordering.

Proposition 3 *If* $L_1 \subseteq L_1'$, *then* $P_1^{-1}(L_1) \subseteq P_1^{-1}(L_1')$

Proof:

Assume $L_1 \subseteq L_1'$. **(1)**

We will now show this implies that $P_1^{-1}(L_1) \subseteq P_1^{-1}(L_1')$.

Let $s \in P_1^{-1}(L_1)$ **(2)**

We will now show this implies that $s \in P_1^{-1}(L_1')$.

By **(2)**, we have $P_1(s) \in L_1$

$\Rightarrow P_1(s) \in L_1'$, by **(1)**.

$\Rightarrow s \in P_1^{-1}(L_1')$, as required.

We can thus conclude $P_1^{-1}(L_1) \subseteq P_1^{-1}(L_1')$

QED

The last language proposition says that the synchronous product operator respects subset ordering. This is useful for working with the languages of a DES created by the synchronous product operator.

Proposition 4 *If* $L_1 \subseteq L_1'$ *and* $L_2 \subseteq L_2'$, *then* $L_1||L_2 \subseteq L_1'||L_2'$

Proof:

Assume $L_1 \subseteq L_1'$ and $L_2 \subseteq L_2'$. **(1)**

We will now show this implies that $L_1||L_2 \subseteq L_1'||L_2'$

Let $s \in L_1||L_2$ **(2)**

We will now show this implies that $s \in L_1'||L_2'$

From **(2)**, we have $s \in P_1^{-1}(L_1) \cap P_2^{-1}(L_2)$ **(3)**

From **(1)**, we can apply **Proposition 3** twice and conclude $P_1^{-1}(L_1) \subseteq P_1^{-1}(L_1')$ and $P_2^{-1}(L_2) \subseteq P_2^{-1}(L_2')$ **(4)**

Combining with **(3)**, we can now conclude $s \in P_1^{-1}(L_1') \cap P_2^{-1}(L_2')$

$\Rightarrow s \in L_1'||L_2'$ by the definition of the synchronous product operator. We thus have $L_1||L_2 \subseteq L_1'||L_2'$, as required.

QED

Next, we apply the above propositions to the synchronous product of DES.

Proposition 5 *If $G = G_1||_s G_2$, then language $L(G)$ is closed and $L_m(G) \subseteq L(G)$*

Proof:

Assume $G = G_1||_s G_2$. **(1)**

We will now show this implies that language $L(G)$ is closed and $L_m(G) \subseteq L(G)$.

By definition of the $||_s$ operator, we know that $L(G) = L(\mathbf{G_1})||L(\mathbf{G_2})$ and that $L_m(\mathbf{G}) = L_m(\mathbf{G_1})||L_m(\mathbf{G_2})$ **(2)**

We now note that languages $L(G_1)$, and $L(G_2)$ are closed by the definition of the closed behaviour of a DES.

We can immediately apply **Corollary 1** and conclude that $L(G)$ is closed.

From the definition of the closed behaviour and the marked language of a DES, we can conclude that $L_m(\mathbf{G_1}) \subseteq L(\mathbf{G_1})$ and $L_m(\mathbf{G_2}) \subseteq L(\mathbf{G_2})$.

We can now apply **Proposition 4** and conclude:

$$L_m(\mathbf{G_1})||L_m(\mathbf{G_2}) = \ L_m(G) \subseteq L(G) \ = L(\mathbf{G_1})||L(\mathbf{G_2})$$

QED

For our last proposition and its accompanying corollary, we need to introduce some different notation to avoid confusion with later notation. We will be using alphabets $\Sigma_a, \Sigma_b \subseteq \Sigma$ and natural projections

$P_k : \Sigma^* \to \Sigma_k^*$, where $k = a, b$.

The following proposition provides a useful relationship for natural projections when the language each is projecting onto has the given relationship. When we examine the parallel case, we will see many instances of this relationship.

Proposition 6 *If* $\Sigma_b \subseteq \Sigma_a$ *then* $P_a^{-1} \cdot P_a \cdot P_b^{-1} = P_b^{-1}$

Proof:

Assume $\Sigma_b \subseteq \Sigma_a$. **(1)**

We will now show this implies $P_a^{-1} \cdot P_a \cdot P_b^{-1} = P_b^{-1}$

Let $s \in \Sigma_b^*$. Sufficient to show $P_b^{-1}(s) = P_a^{-1} \cdot P_a \cdot P_b^{-1}(s)$

This means showing: **I)** $P_b^{-1}(s) \subseteq P_a^{-1} \cdot P_a \cdot P_b^{-1}(s)$ and **II)** $P_a^{-1} \cdot P_a \cdot P_b^{-1}(s) \subseteq P_b^{-1}(s)$

I) Show $P_b^{-1}(s) \subseteq P_a^{-1} \cdot P_a \cdot P_b^{-1}(s)$

Let $s' \in P_b^{-1}(s)$. Will show implies $s' \in P_a^{-1} \cdot P_a \cdot P_b^{-1}(s)$.

$\Rightarrow \; P_a(s') \in P_a \cdot P_b^{-1}(s)$

$\Rightarrow \; P_a^{-1} \cdot P_a(s') \subseteq P_a^{-1} \cdot P_a \cdot P_b^{-1}(s)$

Clearly $s' \in P_a^{-1} \cdot P_a(s')$, as $P_a^{-1} \cdot P_a(s') := \{s'' \in \Sigma^* \mid P_a(s'') = P_a(s')\}$.

$\Rightarrow \; s' \in P_a^{-1} \cdot P_a \cdot P_b^{-1}(s)$, as required.

Case I complete.

II) Show $P_a^{-1} \cdot P_a \cdot P_b^{-1}(s) \subseteq P_b^{-1}(s)$

Let $s' \in P_a^{-1} \cdot P_a \cdot P_b^{-1}(s) := \cup_{u \in P_a \cdot P_b^{-1}(s)} \{v \in \Sigma^* \mid P_a(v) = u\}$. **(2)**

Will show implies $s' \in P_b^{-1}(s)$. Sufficient to show $P_b(s') = s$.

From **(2)**, we have $P_a(s') \in P_a \cdot P_b^{-1}(s)$

$\Rightarrow \; (\exists s'' \in \Sigma^*)$ s.t. $s'' \in P_b^{-1}(s) \wedge P_a(s'') = P_a(s')$ **(3)**

As $s'' \in P_b^{-1}(s)$, we have $P_b(s'') = s$

Since $\Sigma_b \subseteq \Sigma_a$ (from **(1)**), we can conclude: $(\forall t \in \Sigma^*)\, P_b(t) = P_b \cdot P_a(t)$ **(4)**

From this we can conclude $P_b \cdot P_a(s'') = P_b(s'') = s$

From **(3)**, we have $P_a(s'') = P_a(s')$. We can thus conclude $P_b \cdot P_a(s') = P_b \cdot P_a(s'') = s$

From **(4)**, we can conclude $P_b(s') = P_b \cdot P_a(s') = s$

$\Rightarrow \ s' \in P_b^{-1}(s)$, as required.

Case II complete.

By **Case I** and **Case II**, we have $P_b^{-1}(s) = P_a^{-1} \cdot P_a \cdot P_b^{-1}(s)$ and thus conclude $P_a^{-1} \cdot P_a \cdot P_b^{-1} = P_b^{-1}$

QED

The corollary below applies **Proposition 6** to provide a useful result for strings. We will be using this corollary extensively when we examine the parallel case.

Corollary 2 *If* $\Sigma_b \subseteq \Sigma_a$ *and* $L_b \subseteq \Sigma_b^*$ *then* $(\forall s \in \Sigma^*)\ P_a(s) \in P_a \cdot P_b^{-1}(L_b) \Rightarrow s \in P_b^{-1}(L_b)$

Proof:

Assume $\Sigma_b \subseteq \Sigma_a$ and $L_b \subseteq \Sigma_b^*$. **(1)**

Let $s \in \Sigma^*$ and $P_a(s) \in P_a \cdot P_b^{-1}(L_b)$. **(2)**

We will now show this implies that $s \in P_b^{-1}(L_b)$

We start by noting that **(2)** implies that $s \in P_a^{-1} \cdot P_a \cdot P_b^{-1}(L_b)$

From **Proposition 6**, we can conclude $P_a^{-1} \cdot P_a \cdot P_b^{-1} = P_b^{-1}$, by **(1)**.

$s \in P_b^{-1}(L_b)$ follows automatically.

QED

Chapter 2

Serial Case: Nonblocking

With the serial case of *hierarchical interface-based supervisory control,* what we are proposing is essentially a master-slave system, where a *high level subsystem* sends a command to a *low level subsystem,* which then performs the indicated task and sends back a reply. Figure 2.1 shows conceptually the structure of the system.

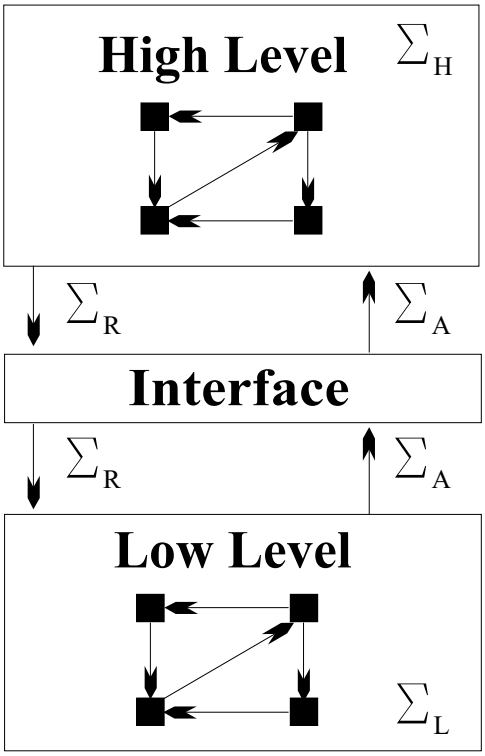


Figure 2.1: Interface Block Diagram.

To allow the system to be designed/maintained/verified on a component-wise basis, we impose an interface between the two subsystems that limits their interaction and knowledge of each other. The goal is to be able to work with each subsystem individually, requiring no information about the other subsystem beyond that provided by the interface.

To capture the restriction of the flow of information imposed by the *interface,* we split the alphabet of the system (Σ) into four mutually disjoint alphabets: Σ_H, Σ_L, Σ_R, and Σ_A. The events in Σ_H are called *high level events* and the events in Σ_L *low level events* as these events appear only in the high level and low level models, respectively.

The alphabets Σ_R and Σ_A are called collectively *interface events.* These events are common to both levels of the hierarchy and represent communication between the two subsystems. More specifically, the events in Σ_R are called *request events* and represent commands sent from the *high level subsystem* to the *low level subsystem.* The events in Σ_A are called *answer events* and represent the low level's responses to the *request events* (high-level commands). Figure 2.1 shows conceptually the flow of information in our setting.

In the remainder of the chapter, we will first present a definition for an interface, followed by a set of local consistency and nonblocking requirements that the interface and subsystems must satisfy in order to guarantee global nonblocking. We then provide several supporting propositions, followed by the serial nonblocking theorem.

2.1 Notation and Definitions

In this section, we present a definition for interface, and some notation that will be useful in the following proofs.

2.1.1 Interface Definition

In this section we, will present two interface definitions: *star interfaces* and *command-pair interfaces.* As we will see later, *star interfaces* are special case of the more general *command-pair interfaces.*

We start by describing a *star interface* as it has a regular structure and is thus easy to construct. To define a *star interface,* the designer selects a set of *request events*, and then for each *request event*, the designer defines a set of *answer events.* In essence, the designer defines a map $\mathbf{Answer} : \Sigma_R \rightarrow \mathrm{Pwr}(\Sigma_A)$. For $\rho \in \Sigma_R$, $\mathbf{Answer}(\rho)$ is the set of possible answers (referred to as the *answer set*) the *low level subsystem* could provide after receiving request ρ. For consistency, we add the constraints given below.

Point 1 states that the *low level subsystem* must provide at least one response for each request it receives, and **point 2** states that Σ_A does not contain any unused events.

1. $(\forall \rho \in \Sigma_R)\, \mathbf{Answer}(\rho) \neq \emptyset$
2. $\Sigma_A = \cup_{\rho \in \Sigma_R} \mathbf{Answer}(\rho)$

In Figure 2.2, we see how a *star interface,* with $n = |\Sigma_R|$ $(n \geq 0)$, is expressed as a DES. The required structure for a *star interface* is given by DES G_I. Analysing DES G_I, we see that the DES has the following properties:

- The initial state is the only marked state.
- *Request events* are the only events defined at the initial state.
- Each *request event* starts at the initial state, and ends at state other than the initial state.
- *Answer events* are not defined at the initial state
- At least one *answer event* transition can always follow a *request event* transition.

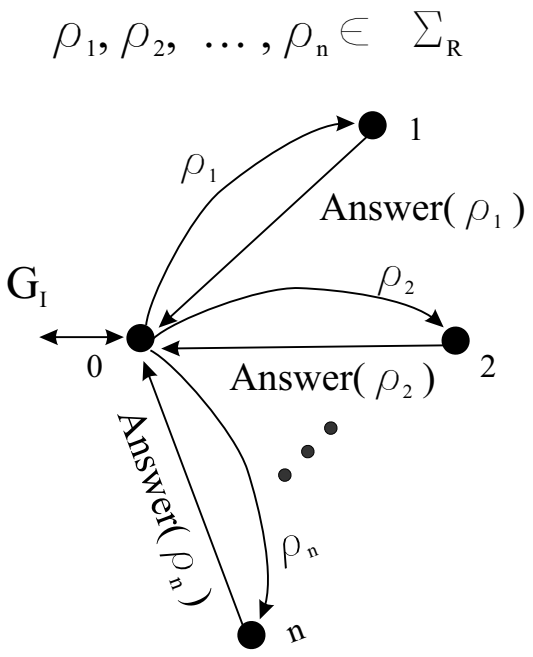


Figure 2.2: Interface Specification.

To allow for a minimal DES, we permit distinct *request events* to have the same next state. For example, if *request events* ρ_1 and ρ_2 in Figure 2.2 had the same *answer sets* (i.e. $\mathbf{Answer}(\rho_1) =$

Answer(ρ_2)), then their next states can be merged. In our example, this would mean states 1 and 2 of G_I would be combined. Finally, we require that the event set of G_I be set to $\Sigma_R \dot{\cup} \Sigma_A$ but we make no restrictions on whether a *request* or *answer event* is controllable or uncontrollable.

We now define *command-pair interfaces. Command-pair interfaces* were designed as a generalisation of *star interfaces. Star interfaces* were designed first as they were more intuitive, and then the key properties were identified and collected into the *command-pair interface* definition. A key difference is that the "star" shape is no longer required. A *command-pair interface* will still always have a *request event* followed by an *answer event,* but it can now contain additional state information. For example, in Figure 2.2 all possible *request events* are defined at the initial state. When an *answer event* has occurred, it always returns the *star interface* to the initial state, and thus the same choice of potential *request events*. With a *command-pair interface* we can have a DES structure as illustrated in Figure 2.3. *Request events* ρ_1 and ρ_2 might represent the regular behaviour of the system, while α_3 and ρ_3 represent breakdown and repair of the system. A *command-pair interface* allows the flexibility of only having the repair event eligible after a breakdown.

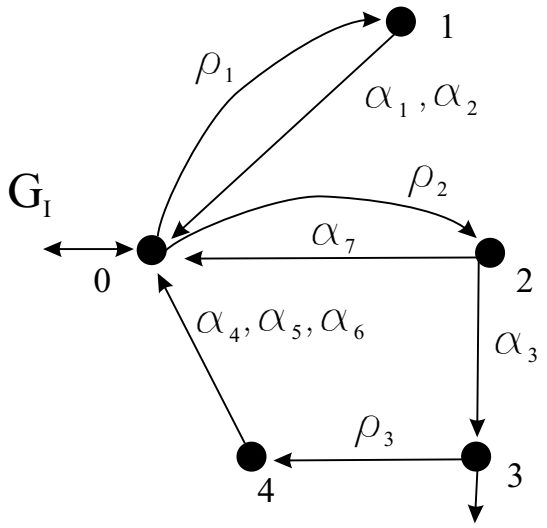


Figure 2.3: Example *Command-pair Interface.*

For the remainder of this work, when we refer to an *interface* we will mean explicitly a *command-pair interface,* and we will use the two synonymously. We define a *command-pair interface* as below:

Definition: A DES $G_I = (X, \Sigma_I, \xi, x_o, X_m)$ is a *command-pair interface* if the following conditions are satisfied:

(A) $\Sigma_I = \Sigma_R \dot{\cup} \Sigma_A$

(B) $(\forall s \in L(G_I))(\forall \rho \in \Sigma_R)\ s\rho \in L(G_I) \Rightarrow s \in L_m(G_I)$

(C) $(\forall s \in L_m(G_I))(\forall \sigma \in \Sigma_I)\ s\sigma \in L(G_I) \Rightarrow \sigma \notin \Sigma_A$

(D) $L_m(G_I) = \{\epsilon\} \cup (\Sigma_I^*.\Sigma_A \cap L(G_I))$

(E) $L(G_I) \subseteq \overline{(\Sigma_R.\Sigma_A)^*}$

The first point, **point A**, says that G_I's event set is restricted to *request* and *answer events.* It also states that the two sets are disjoint. **Point B** then states that *request event* transitions are only defined at marked states. **Point C** states that there are no *answer events* defined at marked states. **Point D** says that the marked language of G_I consists of the empty string, and strings that end in an *answer event.* Finally, **Point E** says that in the language of G_I, a *request event* always occurs first and then *request* and *answer events* alternate.

We will now prove that *star interfaces* are a special case of *command-pair interfaces.* This will allow us to prove our theorems for the more general *command-pair interfaces,* but use the simpler *star interfaces* when they are sufficient.[1]

Proposition 7 *If DES* $G_I = (X, \Sigma_I, \xi, x_o, X_m)$ *is a* star interface, *then* G_I *is a* command-pair interface.

Proof:

Assume that G_I is a *star interface.*

We will now show this implies it's a *command-pair interface* by showing that G_I satisfies **points A-E** of the *command-pair interface* definition.

Point A: Show $\Sigma_I = \Sigma_R \dot{\cup} \Sigma_A$

This is automatic from the *star interface* definition.

Point B: Show $(\forall s \in L(G_I))(\forall \rho \in \Sigma_R)\ s\rho \in L(G_I) \Rightarrow s \in L_m(G_I)$

The results follow immediately from observing Figure 2.2 and noting that *request event* transitions are only defined at the initial state, which is marked.

Point C: Show $(\forall s \in L_m(G_I))(\forall \sigma \in \Sigma_I)\ s\sigma \in L(G_I) \Rightarrow \sigma \notin \Sigma_A$

[1] We will actually use *star interfaces* exclusively in our algorithms and examples as the *command-pair interface* definition was only just developed and there wasn't time to update our software and examples.

The results follow immediately from observing Figure 2.2 and noting that *answer event* transitions are not defined at the initial state, which is the only marked state.

Point D: Show $L_m(G_I) = \{\epsilon\} \cup (\Sigma_I^*.\Sigma_A \cap L(G_I))$

It is sufficient to show **1)** $L_m(G_I) \subseteq \{\epsilon\} \cup (\Sigma_I^*.\Sigma_A \cap L(G_I))$ and **2)** $L_m(G_I) \supseteq \{\epsilon\} \cup (\Sigma_I^*.\Sigma_A \cap L(G_I))$

Case 1) From Figure 2.2, we can see that $L_m(G_I)$ includes the empty string, as the initial state is marked and that G_I is not the **EMPTY** DES. We also see that the only transitions ending at the initial state (the only marked state) are for *answer events.* The results follow immediately.

Case 2) From above, we have that $L_m(G_I)$ includes the empty string. From Figure 2.2, we can see that every *answer event* transition ends at the initial state, which is marked. The results follow immediately.

By **case 1** and **2**, we thus have $L_m(G_I) = \{\epsilon\} \cup (\Sigma_I^*.\Sigma_A \cap L(G_I))$

Point E: Show $L(G_I) \subseteq \overline{(\Sigma_R.\Sigma_A)^*}$

From Figure 2.2, we can see that $L(G_I)$ contains the empty string (as G_I is not the **EMPTY** DES), and strings that start with a *request event.* We further see that *request* and *answer events* then alternate. The results follow immediately.

QED

2.1.2 Terminology and Notation

We now present some terminology and notation that will be useful in simplifying proofs. For our setting, we assume the *high level subsystem* is modelled by DES G_H (defined over event set $\Sigma_H \cup \Sigma_R \cup \Sigma_A$), the *low level subsystem* by DES G_L (defined over event set $\Sigma_L \cup \Sigma_R \cup \Sigma_A$), and the interface by DES G_I. Also, the term *high level* will mean the DES $G_H||_s G_I$, and the term *low level* the DES $G_L||_s G_I$. The overall structure of the system is displayed in Figure 2.4.

We next assume that the alphabet partition is specified by $\Sigma := \Sigma_H \dot{\cup} \Sigma_L \dot{\cup} \Sigma_R \dot{\cup} \Sigma_A$ and define the *flat system* as below. By *flat system* we refer to the equivalent DES that would represent our system if we ignored the interface structure.

$$G = G_H||_s G_L||_s G_I$$

As we will often be referring to different groupings of events, we define the following subsets:

$$\begin{aligned} \Sigma_I &:= \quad \Sigma_R \dot{\cup} \Sigma_A \quad && \textit{Interface Events} \\ \Sigma_{IH} &:= \quad \Sigma_H \dot{\cup} \Sigma_R \dot{\cup} \Sigma_A \quad && \textit{Interface \& High Level Events} \end{aligned}$$

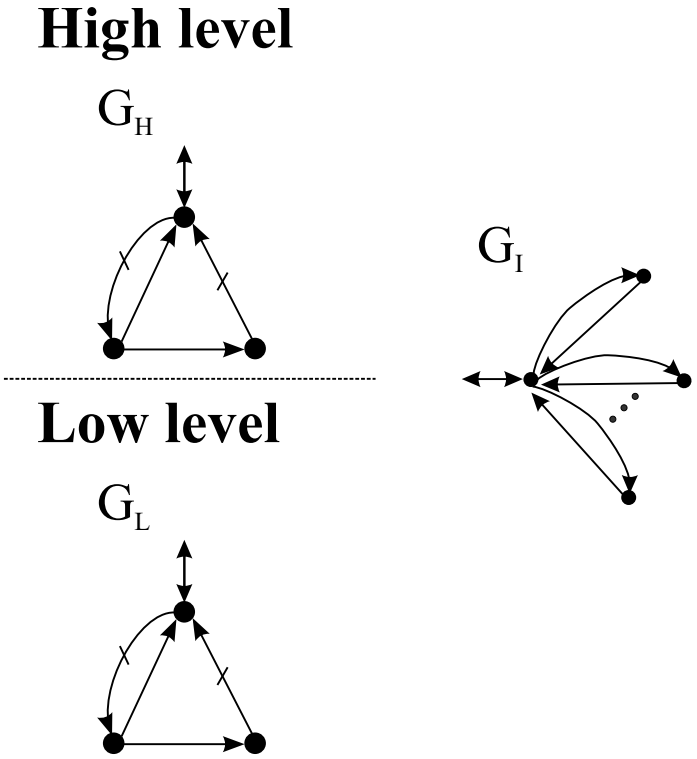


Figure 2.4: Two Tiered Structure of Serial System.

$$\Sigma_{IL} := \quad \Sigma_L \dot{\cup} \Sigma_R \dot{\cup} \Sigma_A \quad \textit{Interface \& Low Level Events}$$

To work with languages defined over subsets of Σ, we define the following natural projections:

$$\begin{aligned} P_{IH} : \Sigma^* \rightarrow \quad & \Sigma_{IH}^* \\ P_{IL} : \Sigma^* \rightarrow \quad & \Sigma_{IL}^* \\ P_I : \Sigma^* \rightarrow \quad & \Sigma_I^* \end{aligned}$$

As we want to express the languages of *flat system* in terms of their components, we need to define the following languages:

$$\begin{aligned} \mathcal{H} := P_{IH}^{-1}(L(G_H)), \quad & \mathcal{H}_m := P_{IH}^{-1}(L_m(G_H)) \quad \subseteq \Sigma^* \\ \mathcal{L} := P_{IL}^{-1}(L(G_L)), \quad & \mathcal{L}_m := P_{IL}^{-1}(L_m(G_L)) \quad \subseteq \Sigma^* \\ \mathcal{I} := P_I^{-1}(L(G_I)), \quad & \mathcal{I}_m := P_I^{-1}(L_m(G_I)) \quad \subseteq \Sigma^* \end{aligned}$$

We can now represent the closed behaviour of our *flat system* as follows:

$$\begin{aligned} L(G) := \quad & L(G_H||_s G_L||_s G_I) \\ = \; & P_{IH}^{-1}(L(G_H)) \cap P_{IL}^{-1}(L(G_L)) \cap P_I^{-1}(L(G_I)) \\ = \; & \mathcal{H} \cap \mathcal{L} \cap \mathcal{I} \end{aligned}$$

Similarly, the marked language of our *flat system* is:

$$L_m(G) = \mathcal{H}_m \cap \mathcal{L}_m \cap \mathcal{I}_m$$

This allows us to present the proposition below which collects together several similar propositions. As it will be common in the proofs in this report to show that membership in languages such as $\mathcal{H}$ are dependent only on events in specific subsets (for $\mathcal{H}$, events in subset Σ_{IH}), this proposition will be very useful.

Proposition 8

(a) $(\forall s, s' \in \Sigma^*)\ s \in \mathcal{H}$ *and* $P_{IH}(s) = P_{IH}(s') \Rightarrow s' \in \mathcal{H}$

(b) $(\forall s, s' \in \Sigma^*)\ s \in \mathcal{H}_m$ *and* $P_{IH}(s) = P_{IH}(s') \Rightarrow s' \in \mathcal{H}_m$

(c) $(\forall s, s' \in \Sigma^*)\ s \in \mathcal{L}$ *and* $P_{IL}(s) = P_{IL}(s') \Rightarrow s' \in \mathcal{L}$

(d) $(\forall s, s' \in \Sigma^*)\ s \in \mathcal{L}_m$ *and* $P_{IL}(s) = P_{IL}(s') \Rightarrow s' \in \mathcal{L}_m$

(e) $(\forall s, s' \in \Sigma^*)\ s \in \mathcal{I}$ *and* $P_I(s) = P_I(s') \Rightarrow s' \in \mathcal{I}$

(f) $(\forall s, s' \in \Sigma^*)\ s \in \mathcal{I}_m$ *and* $P_I(s) = P_I(s') \Rightarrow s' \in \mathcal{I}_m$

Proof:

Point a:

Let $s, s' \in \Sigma^*$, $s \in \mathcal{H}$ and $P_{IH}(s) = P_{IH}(s')$ **(1)**

$s \in \mathcal{H} \Rightarrow s \in P_{IH}^{-1}(L(G_H))$, by definition of $\mathcal{H}$.

$\Rightarrow P_{IH}(s) \in L(G_H)$

$\Rightarrow P_{IH}(s') \in L(G_H)$, by **(1)**.

$\Rightarrow s' \in P_{IH}^{-1}(L(G_H)) = \mathcal{H}$, as required.

Points b-f:

Identical to the proof of **point a** above, after substitution.

QED

2.2 Serial Interface Consistent and Nonblocking

In this section, we present the interface properties that our system must satisfy to ensure that it interacts with the *interface* correctly as well as the nonblocking requirements each level must satisfy. Together they provide a set of local conditions that can be evaluated using at most one level of our hierarchy at a time.

Our first definition is the *serial level-wise nonblocking* definition. It requires that each level be individually nonblocking.

Serial Level-wise Nonblocking: The system composed of DES G_H, G_L, and G_I, is said to be *serial level-wise nonblocking* if the following conditions are satisfied:

(I) $\overline{\mathcal{H}_m \cap \mathcal{I}_m} = \mathcal{H} \cap \mathcal{I}$ *Nonblocking at the high level*

(II) $\overline{\mathcal{L}_m \cap \mathcal{I}_m} = \mathcal{L} \cap \mathcal{I}$ *Nonblocking at the low level*

We now present the *serial interface consistent* definition. It defines the interface properties that our system must satisfy to ensure that it interacts with the *interface* correctly. It limits the information each level can have about the other, and what assumptions they can make about each other. As one can see from Appendix A, these conditions can be evaluated using only "local" information - properties of a given subsystem and its interface.

Serial Interface Consistent: The system composed of DES G_H, G_L and G_I, is *serial interface consistent* with respect to the alphabet partition $\Sigma := \Sigma_H \dot{\cup} \Sigma_L \dot{\cup} \Sigma_R \dot{\cup} \Sigma_A$, if the following properties are satisfied:

Multi-level Properties

1. The event set of G_H is Σ_{IH}, and the event set of G_L is Σ_{IL}.
2. G_I is a *command-pair interface.*

High Level Properties

3. $(\forall s \in \mathcal{H} \cap \mathcal{I})\ \text{Elig}_{\mathcal{I}}(s)\ \cap\ \Sigma_A \subseteq \text{Elig}_{\mathcal{H}}(s)$ *High level task completion agreement*

Low Level Properties

4. $(\forall s \in \mathcal{L} \cap \mathcal{I})\ \text{Elig}_{\mathcal{I}}(s)\ \cap\ \Sigma_R \subseteq \text{Elig}_{\mathcal{L}}(s)$ *Low level task request agreement*
5. $(\forall s \in \Sigma^*.\Sigma_R \cap \mathcal{L} \cap \mathcal{I})$

   $\text{Elig}_{\mathcal{L}\cap\mathcal{I}}(s\Sigma_L^*) \cap \Sigma_A = \text{Elig}_{\mathcal{I}}(s) \cap \Sigma_A$ *Low level task completion agreement*

   where $\text{Elig}_{\mathcal{L}\cap\mathcal{I}}(s\Sigma_L^*) := \cup_{l\in\Sigma_L^*}\text{Elig}_{\mathcal{L}\cap\mathcal{I}}(sl)$
6. $(\forall s \in \mathcal{L} \cap \mathcal{I})$

   $s \in \mathcal{I}_m \Rightarrow (\exists l \in \Sigma_L^*)\ sl \in \mathcal{L}_m \cap \mathcal{I}_m$ *Low level marking agreement*

We will now give a brief discussion of the meaning of each property.

Property 0: The first Property is inherent in the definition of the alphabet partition, $\Sigma := \Sigma_H \dot\cup \Sigma_L \dot\cup \Sigma_R \dot\cup \Sigma_A$. It states that the four alphabets are pairwise disjoint.

Property 1: This property asserts that the high and low levels can only share *request* and *answer events.* This is an "information hiding" statement. It restricts the *high level subsystem* from knowing (and directly affecting) internal details about the *low level subsystem* (ie to be able to view/disable *low level events*) and vice versa.

Property 2: This property states that DES G_I satisfies the definition of a *command-pair interface.*

Property 3: This property asserts that the *high level subsystem* (G_H) must always accept an *answer event* if the event is eligible in the *interface.* If the *answer event* is not eligible in the *interface,* the *high level subsystem* is not required to accept it. This is equivalent to a controllability condition where the *interface* is taken to be the plant, the *high level subsystem* the supervisor, and *answer events* to be the uncontrollable events. In other words, the *high level subsystem* is forbidden to assume more about when an *answer event* can occur than what is provide by the *interface.*

Property 4: This property asserts that the *low level subsystem* (G_L) must always accept a *request event* if the event is eligible in the *interface.* If the *request event* is not eligible in the *interface,* the *low level subsystem* is not required to accept it. This is equivalent to a controllability condition where the *interface* is taken to be the plant, the *low level subsystem* the supervisor, and *request*

events to be the uncontrollable events. In other words, the *low level* is forbidden to assume more about when an *request event* can occur than what is provide by the *interface*.

Property 5: This property asserts that immediately after a *request event* (some $\rho \in \Sigma_R$) has occurred (and before it is followed by any *low level events*), there exists one or more paths via strings in Σ_L^* to each *answer event* (ie all $\alpha \in \mathbf{Answer}(\rho)$, assuming that we are dealing with a *star interface*) that can follow the *request event.* A given path may only lead to one of the possible *answer events,* but a path to each one must exist. However, as soon as a single *low level event* has occurred, one or more *answer events* may be no longer reachable (ie the *low level subsystem* may no longer be able to provide that particular *answer event*).

For example, assume our *low level* represents a machine that accepts the *request events* **startJob** and **startRepair**. *Request event* **startJob** can be followed by *answer events* **jobCompleted** or **machineDown**. *Request event* **startRepair** can be followed by *answer event* **repairCompleted** (this information would be embodied in $L(G_I)$). Now, let's consider event **startJob**. Immediately after *request event* **startJob** has occurred, there must be one or more *low level event* sequences, accepted by the *low level subsystem*, that lead to *answer events* **jobCompleted** and **machineDown**. For example, sequence **taskAComplete-taskBComplete** would bring the *low level subsystem* into a state that it would accept *answer event* **jobCompleted** but event **machineDown** would no longer be possible. Similarly, sequence **taskAComplete-machineFailure** would bring the *low level subsystem* into a state that it would accept *answer event* **machineDown** but event **jobCompleted** would no longer be possible. What's important to note here is that both *answer events* **jobCompleted** and **machineDown** were initially reachable after **startJob** occurred. Which *answer event* was allowed to occur was determined afterwards solely by *low level events.* Also, after one or more *low level events* had occurred, there was no guarantee that both *answer events* were still reachable.

One could summarise the purpose of this condition as to guarantee "honest advertising." If the *interface* asserts that a given *answer event* can follow a given *request event,* this must always be true at least immediately after the *request event* occurs, and then the *low level subsystem* is allowed through the occurrence of *low level events* to decide which *answer event* actually occurs. In our above example, our *interface* advertises that either *answer event* **jobCompleted** or **machineDown** can follow *request event* **startJob**. If the *low level subsystem* was in a down state such that only **machineDown** was possible, and *request event* **startJob** occurred, this condition

would be violated. If the *high level* could only reach a marked state by the eventual occurrence of event **jobCompleted**, it would be deceived by the "false advertising" of the *interface* into believing that this could eventually be satisfied by repeatedly issuing **startJob** commands. Of course, a human designer would see the fallacy of this, but an automatic synthesis algorithm would be deceived.

Property 6: This property asserts that every string marked by the *interface* and accepted by the *low level subsystem*, can be extended by a low level string to a string marked by the *low level* (both G_I and G_L). In other words, once the *low level subsystem* has returned an *answer event*, it can always return to a marked state via a low level string (ie some $s \in \Sigma_L^*$).

From the point of the view of the *high level subsystem*, this property says that if one can bring the *high level subsystem* and the *interface* to a marked state, then one can bring the *low level subsystem* to a marked state via a low level string which would be ignored by the *high level* (ie they would stay in a marked state).

We are now ready to state the proposition below which establishes useful properties for often used languages.

Proposition 9 *If the system composed of DES G_H, G_L, and G_I is* serial interface consistent *with respect to the alphabet partition $\Sigma := \Sigma_H \dot\cup \Sigma_L \dot\cup \Sigma_R \dot\cup \Sigma_A$, then the following is true:*

(i) Languages $\mathcal{H}$, $\mathcal{L}$, and $\mathcal{I}$ are closed.

(ii) $\mathcal{H}_m \subseteq \mathcal{H}$, $\mathcal{L}_m \subseteq \mathcal{L}$, and $\mathcal{I}_m \subseteq \mathcal{I}$

Proof: See page 32.

Now that we have presented the *serial interface consistent* definition, we present the *serial interface strict marking* condition, which is a restriction of the *serial interface consistent* definition. As we will see later, this restriction is useful as it implies **Property 6** of the *serial interface consistent* definition, but is less expensive to evaluate.

Serial Interface Strict Marking: The system composed of DES G_H, G_L and G_I, is *serial interface strict marking* with respect to the alphabet partition $\Sigma := \Sigma_H \dot\cup \Sigma_L \dot\cup \Sigma_R \dot\cup \Sigma_A$, if:

$(\forall s \in \mathcal{L} \cap \mathcal{I})\ s \in \mathcal{I}_m \Rightarrow\ s \in \mathcal{L}_m \cap \mathcal{I}_m$

The above statement could be summarised by saying that if we can bring the *interface* to a marked state, we are guaranteed to also have brought the *low level subsystem* to a marked state. The above statement is equivalent to **Property 6** of the *serial interface consistent* definition above, with string l restricted to the empty string.

We will now prove that we can use the *serial interface strict marking* condition to replace **Property 6** of the *serial interface consistent* definition.

Proposition 10 *If the system composed of DES G_H, G_L and G_I, satisfies* **properties 1-5** *of the* serial interface consistent *definition and is* serial interface strict marking *with respect to the alphabet partition $\Sigma := \Sigma_H \dot{\cup} \Sigma_L \dot{\cup} \Sigma_R \dot{\cup} \Sigma_A$, then the system is* serial interface consistent.

Proof:

Assume that the system satisfies **properties 1-5** of the *serial interface consistent* definition and is *serial interface strict marking.* **(1)**

We now show this implies the system is *serial interface consistent.*

From **(1)**, we immediately have the system satisfying the first 5 properties of the *serial interface consistent* definition.

All that remains is to show that the system satisfies **Property 6** of the definition. This means showing:

$$(\forall s \in \mathcal{L} \cap \mathcal{I})$$
$$s \in \mathcal{I}_m \Rightarrow (\exists l \in \Sigma_L^*)\ sl \in \mathcal{L}_m \cap \mathcal{I}_m$$

Let $s \in \mathcal{L} \cap \mathcal{I}$, and $s \in \mathcal{L}_m$

We will now show this implies $(\exists l \in \Sigma_L^*)\ sl \in \mathcal{L}_m \cap \mathcal{I}_m$

From **(1)**, we have that the system is *serial interface strict marking.*

We can thus conclude: $s \in \mathcal{L}_m \cap \mathcal{I}_m$

We thus take $l = \epsilon$ and we have $sl \in \mathcal{L}_m \cap \mathcal{I}_m$, as required.

We thus have the system satisfying **properties 1-6,** and is thus *serial interface consistent.*

QED

2.3 Serial Nonblocking Theorem and Propositions

We will now present **Propositions 11-15**, followed by our main result for this chapter, **Theorem 1**. The following propositions perform two tasks: they break down the main theorem into a more manageable size, as well as provide useful results that will be re-used for the parallel case.

2.3.1 Low Level Nonblocking Proposition

Our first proposition is the *low level nonblocking proposition.* It asserts that a string s accepted by the system, can always be extended to a string accepted by the system, and marked by the *low level.* In other words, the *low level* is not dependent on high level events to reach a marked state.

Proposition 11 *If the system composed of DES G_H, G_L, and G_I is* serial level-wise nonblocking *and* serial interface consistent *with respect to the alphabet partition $\Sigma := \Sigma_H \dot\cup \Sigma_L \dot\cup \Sigma_R \dot\cup \Sigma_A$, then*

$(\forall s \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I})$
$\quad (\exists l \in \Sigma_{IL}^*)$ *s.t.* $(sl \in \mathcal{H} \cap \mathcal{L}_m \cap \mathcal{I}_m)$

Proof: See page 33.

2.3.2 Low Level Linkage Proposition

Our next proposition is the *low level linkage proposition.* It asserts that a string s accepted by the *system* and marked by the *high level,* implies that s can be extended by a low level string l such that sl is marked by the *system.* In other words, if you can get the *high level* to a marked state, you can always bring the low level to a marked state by a string containing events the *high level* is indifferent to.

Proposition 12 *If the system composed of DES G_H, G_L, and G_I is* serial level-wise nonblocking *and* serial interface consistent *with respect to the alphabet partition $\Sigma := \Sigma_H \dot\cup \Sigma_L \dot\cup \Sigma_R \dot\cup \Sigma_A$, then*

$(\forall s \in \mathcal{L} \cap \mathcal{H}_m \cap \mathcal{I}_m)\ (\exists l \in \Sigma_L^*)\ sl \in \mathcal{H}_m \cap \mathcal{L}_m \cap \mathcal{I}_m$

Proof:

Assume system is *serial level-wise nonblocking* and *serial interface consistent.* **(1)**

Let $s \in \mathcal{L} \cap \mathcal{H}_m \cap \mathcal{I}_m$ **(2)**

We will now show that we can construct a string $l \in \Sigma_L^*$ such that $sl \in \mathcal{H}_m \cap \mathcal{L}_m \cap \mathcal{I}_m$

From **(2)**, we have $s \in \mathcal{L} \cap \mathcal{I}_m$ and thus $s \in \mathcal{L} \cap \mathcal{I}$.

From **(1)**, we can apply **Point 6** of the *serial interface consistent* definition and conclude:

$$(\exists l \in \Sigma_L^*) \text{ s.t. } sl \in \mathcal{L}_m \cap \mathcal{I}_m \qquad \textbf{(3)}$$

As $l \in \Sigma_L^*$, we have $P_{IH}(s) = P_{IH}(sl)$. From **(2)**, we have $s \in \mathcal{H}_m$. We can now apply **Proposition 8**, **point b**, and conclude:

$$sl \in \mathcal{H}_m$$

Combining with **(3)**, have $l \in \Sigma_L^*$, and $sl \in \mathcal{H}_m \cap \mathcal{L}_m \cap \mathcal{I}_m$, as required.

QED

2.3.3 Event Agreement Propositions

We group the next three propositions together, as each builds upon the previous one.

One-step Event Agreement Proposition

Our first proposition is the one-step event agreement proposition. For this proposition, we are given a string s accepted by the *system* and a string h of the form $\Sigma_H^*.\Sigma_R.\Sigma_H^*.\Sigma_A$. This means that h contains exactly one request event and one *answer event* in the given order and that h may or may not contain *high level events* before or directly after the *request event*. The proposition asserts that if string h extends string s such that sh is acceptable to the *high level*, then a string u can be constructed such that u has a high level image equal to h, and that su is acceptable to the system and marked by the *interface*. In other words, we can use string h as a basis to construct string u by adding low level events so that the *low level subsystem* will accept the request and *answer event* contained in h. As these events are common to both levels, they must agree on their occurrence.

Proposition 13 *If the system composed of DES G_H, G_L, and G_I is* serial level-wise nonblocking *and* serial interface consistent *with respect to the alphabet partition $\Sigma := \Sigma_H \dot\cup \Sigma_L \dot\cup \Sigma_R \dot\cup \Sigma_A$, then*

$$(\forall s \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I})(\forall h \in \Sigma_H^*.\Sigma_R.\Sigma_H^*.\Sigma_A)$$
$$sh \in \mathcal{H} \cap \mathcal{I} \Rightarrow (\exists u \in \Sigma^*) \text{ s.t. } (su \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I}_m) \wedge (P_{IH}(u) = h)$$

Proof: See page 35.

Inductive Event Agreement Proposition

Our next proposition is the inductive event agreement proposition. This proposition is different from **Proposition 13** as **Proposition 13** only handled the case that the string h contains exactly one *answer event* (i.e. only one command-pair), while this proposition allows h to contain one or more *answer events* (i.e. multiple command-pairs). It uses **Proposition 13** in an inductive proof to handle an arbitrary number of *answer events* in string h.

Proposition 14 *If the system composed of DES G_H, G_L, and G_I is* serial level-wise nonblocking *and* serial interface consistent *with respect to the alphabet partition $\Sigma := \Sigma_H \dot{\cup} \Sigma_L \dot{\cup} \Sigma_R \dot{\cup} \Sigma_A$, then*

$(\forall s \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I}_m)(\forall h \in \Sigma_{IH}^*.\Sigma_A)$

$sh \in \mathcal{H} \cap \mathcal{I} \Rightarrow (\exists u \in \Sigma^*)\ (P_{IH}(u) = h) \wedge (su \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I}_m)$

Proof: See page 37.

General Event Agreement Proposition

The last proposition of the three is the general event agreement proposition. This proposition is more general than **Proposition 14** as it handles the case that string h doesn't contain *answer events* or doesn't end in an *answer event.* It make use of **Proposition 14** to handle the other cases.

Proposition 15 *If the system composed of DES G_H, G_L, and G_I is* serial level-wise nonblocking *and* serial interface consistent *with respect to the alphabet partition $\Sigma := \Sigma_H \dot{\cup} \Sigma_L \dot{\cup} \Sigma_R \dot{\cup} \Sigma_A$, then*

$(\forall s \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I}_m)(\forall h \in \Sigma_{IH}^*)$

$sh \in \mathcal{H}_m \cap \mathcal{I}_m \Rightarrow (\exists u \in \Sigma^*)\ s.t.\ (su \in \mathcal{H}_m \cap \mathcal{L}_m \cap \mathcal{I}_m) \wedge (P_{IH}(u) = h) \wedge (P_I(u) \in \{\epsilon\} \cup \Sigma_R.\Sigma_I^*)$

Proof: See page 40.

2.3.4 Serial Nonblocking Theorem

We now present our main result for this chapter, the *serial interface nonblocking theorem.* In essence the theorem says that if the *high level* and *low level* are individually nonblocking, and the system is *serial interface consistent,* then the nonblocking property will be preserved by the synchronous product operation. As the *serial level-wise nonblocking* and *serial interface consistent* definitions can be evaluated by examining only one level of our system at a time, we now have a means to verify nonblocking of our system using local checks.

Theorem 1 *If the system composed of DES G_H, G_L, and G_I is* serial level-wise nonblocking *and* serial interface consistent *with respect to the alphabet partition $\Sigma := \Sigma_H \dot{\cup} \Sigma_L \dot{\cup} \Sigma_R \dot{\cup} \Sigma_A$, then*

$L(G) = \overline{L}_m(G)$, *where* $G = G_H||_s G_L||_s G_I$

Proof:

Assume system is *serial level-wise nonblocking* and *serial interface consistent.* **(1)**

As $\overline{L_m(G)} \subseteq L(G)$ is automatic, it suffices to show $L(G) \subseteq \overline{L_m(G)}$

Let $s \in L(G) = \mathcal{H} \cap \mathcal{L} \cap \mathcal{I}$ **(2)**

We will now show this implies $s \in \overline{L_m(G)}$

It is sufficient to show: $(\exists u \in \Sigma^*)\ su \in L_m(G) = \mathcal{H}_m \cap \mathcal{L}_m \cap \mathcal{I}_m$

Our first step is to show that we can construct a string l, accepted by the high level, that will bring the low level to a marked state.

We can achieve this immediately by noting that $s \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I}$ and **(1)** allows us to exploit **Proposition 11**. We thus conclude:

$$(\exists l \in \Sigma_{IL}^*) \text{ s.t. } sl \in \mathcal{H} \cap \mathcal{L}_m \cap \mathcal{I}_m \quad \textbf{(3)}$$

We will now show that we can extend string sl to a string in $\mathcal{H}_m \cap \mathcal{L}_m \cap \mathcal{I}_m$. To do this, we will use **Proposition 15**. To apply the proposition, we must first construct a string $h \in \Sigma_{IH}^*$ with the property $slh \in \mathcal{H}_m \cap \mathcal{I}_m$

We first note that we have $sl \in \mathcal{H} \cap \mathcal{I}$ from **(3)**. This allows us to apply **Point I** of the *level-wise nonblocking* definition (nonblocking at the high level), and conclude:.

$$(\exists h' \in \Sigma^*) \text{ s.t. } slh' \in \mathcal{H}_m \cap \mathcal{I}_m \quad \textbf{(4)}$$

We next note that:

$$\begin{aligned} P_{IH}(slh') &= P_{IH}(sl)P_{IH}(h') \\ &= P_{IH}(sl)P_{IH}(P_{IH}(h')) \text{ as the natural projection is idempotent.} \\ &= P_{IH}(slP_{IH}(h')) \end{aligned} \quad (\mathbf{5})$$

We can now apply **Proposition 8**, **point b**, and conclude $slP_{IH}(h') \in \mathcal{H}_m$ **(6)**

As $\Sigma_I \subseteq \Sigma_{IH}$, we can conclude $P_I(slh') = P_I(slP_{IH}(h'))$ (by **(5)**).

We can now apply **Proposition 8**, **point f**, and conclude $slP_{IH}(h') \in \mathcal{I}_m$

Combining with **(6)**, we can conclude $slP_{IH}(h') \in \mathcal{H}_m \cap \mathcal{I}_m$

We thus take $h = P_{IH}(h')$ and we have:

$h \in \Sigma_{IH}^*$ and $slh \in \mathcal{H}_m \cap \mathcal{I}_m$ **(7)**

We next note we have $sl \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I}_m$, by **(3)**. We can now apply **Proposition 15**, taking sl to be string s in that proposition, and conclude:

$(\exists u' \in \Sigma^*)$ s.t. $slu' \in \mathcal{H}_m \cap \mathcal{L}_m \cap \mathcal{I}_m$

We then take $u = lu'$ and we have $su \in \mathcal{H}_m \cap \mathcal{L}_m \cap \mathcal{I}_m = L_m(G)$, as required.

QED

2.4 Proofs of Selected Propositions

In order to make this work more readable, the proofs of some propositions in this chapter were not given as the propositions were introduced. They are now presented in the following sections.

2.4.1 Proof of Proposition 9

Proof for **Proposition 9** on page 26: If the system composed of DES G_H, G_L, and G_I is *serial interface consistent* with respect to the alphabet partition $\Sigma := \Sigma_H \dot{\cup} \Sigma_L \dot{\cup} \Sigma_R \dot{\cup} \Sigma_A$, then the following is true:

(i) Languages $\mathcal{H}$, $\mathcal{L}$, and $\mathcal{I}$ are closed.

(ii) $\mathcal{H}_m \subseteq \mathcal{H}$, $\mathcal{L}_m \subseteq \mathcal{L}$, and $\mathcal{I}_m \subseteq \mathcal{I}$

Proof:

Assume system is *serial interface consistent.* **(1)**

Will now show this implies that **Points i** and **ii** are satisfied.

We first note that by **(1)**, the system is *serial interface consistent.* By **Points 1** and 2 of this definition, we can conclude:

$L(G_H), L_m(G_H) \subseteq \Sigma_{IH}^*$, $L(G_L), L_m(G_L) \subseteq \Sigma_{IL}^*$, and $L(G_I), L_m(G_I) \subseteq \Sigma_I^*$.

This tells us that languages $\mathcal{H} = P_{IH}^{-1}(L(G_H))$, $\mathcal{L} = P_{IL}^{-1}(L(G_L))$, $\mathcal{I} = P_I^{-1}(L(G_I))$, $\mathcal{H}_m = P_{IH}^{-1}(L_m(G_H))$, $\mathcal{L}_m = P_{IL}^{-1}(L_m(G_L))$, and $\mathcal{I}_m = P_I^{-1}(L_m(G_I))$ are defined.

Point i: Show that Languages $\mathcal{H}$, $\mathcal{L}$, and $\mathcal{I}$ are closed.

We now note that languages $L(G_H)$, $L(G_L)$, and $L(G_I)$ are closed by the definition of the closed behaviour of a DES.

We can now apply **Proposition 1** repeatedly and conclude that $\mathcal{H}$, $\mathcal{L}$, and $\mathcal{I}$ are closed, as required.

Point ii: Show that $\mathcal{H}_m \subseteq \mathcal{H}$, $\mathcal{L}_m \subseteq \mathcal{L}$, and $\mathcal{I}_m \subseteq \mathcal{I}$.

From the definition of the closed behaviour and the marked language of a DES, we can conclude that:

$L_m(G_H) \subseteq L(G_H)$, $L_m(G_L) \subseteq L(G_L)$, and $L_m(G_I) \subseteq L(G_I)$.

Applying **Proposition 3** repeatedly, we can conclude:

$$\begin{array}{rcl} P_{IH}^{-1}(L_m(G_H)) = \mathcal{H}_m & \subseteq & \mathcal{H} = P_{IH}^{-1}(L(G_H)) \\ P_{IL}^{-1}(L_m(G_L)) = \mathcal{L}_m & \subseteq & \mathcal{L} = P_{IL}^{-1}(L(G_L)) \\ P_{I}^{-1}(L_m(G_I)) = \mathcal{I}_m & \subseteq & \mathcal{I} = P_{I}^{-1}(L(G_I)) \end{array}$$

QED

2.4.2 Proof of Proposition 11

Proof for **Proposition 11** on page 28: *If the system composed of DES G_H, G_L, and G_I is* serial level-wise nonblocking *and* serial interface consistent *with respect to the alphabet partition $\Sigma := \Sigma_H \dot\cup \Sigma_L \dot\cup \Sigma_R \dot\cup \Sigma_A$, then*

$(\forall s \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I})$

$\quad (\exists l \in \Sigma_{IL}^*)$ *s.t.* $(sl \in \mathcal{H} \cap \mathcal{L}_m \cap \mathcal{I}_m)$

Proof:

Assume system is *serial level-wise nonblocking* and *serial interface consistent.* **(1)**

Let $s \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I}$ **(2)**

Will now show this implies $(\exists l \in \Sigma_{IL}^*)$ s.t. $(sl \in \mathcal{H} \cap \mathcal{L}_m \cap \mathcal{I}_m)$

To do this, we will construct a suitable string l. We start by applying **Point II** of the *serial level-wise nonblocking* definition (by **(1)**) to conclude:

$(\exists s' \in \Sigma^*)\ ss' \in \mathcal{L}_m \cap \mathcal{I}_m$ **(3)**

As we require a string in Σ_{IL}^*, we take $l' = P_{IL}(s') \in \Sigma_{IL}^*$. We will now show $sl' \in \mathcal{L}_m \cap \mathcal{I}_m$

From the definition of the natural projection, we have $P_{IL}(P_{IL}(s')) = P_{IL}(s')$

$\Rightarrow P_{IL}(ss') = P_{IL}(s)P_{IL}(s') = P_{IL}(s)P_{IL}(P_{IL}(s')) = P_{IL}(sP_{IL}(s')) = P_{IL}(sl')$, as the natural projection
is concatenative.

As $\Sigma_I \subseteq \Sigma_{IL}$, $P_I(s') = P_I(P_{IL}(s))$. We can thus argue as above and conclude: $P_I(ss') = P_I(sl')$

We can now use **(3)**, and apply **Proposition 8**, **points d** and **f**, and conclude:

$$sl' \in \mathcal{L}_m \cap \mathcal{I}_m \qquad \textbf{(4)}$$

We now note that if $sl' \in \mathcal{H}$, we can take $l = l'$ and we have the desired result. We can thus, with no loss in generality, assume:

$$sl' \notin \mathcal{H} \qquad \textbf{(5)}$$

We will now show this implies that string sl' is not accepted by $\mathcal{H}$ due to a *request event*. We will then use this fact to construct a suitable string l.

We first observe that $s \in \mathcal{H}$, $sl' \notin \mathcal{H}$, and $l' \in \Sigma_{IL}^*$ implies:
$(\exists l'' \in \Sigma_{IL}^*)(\exists \sigma \in \Sigma_{IL})$ s.t. $(l''\sigma \leq l') \wedge (sl'' \in \mathcal{H}) \wedge (sl''\sigma \notin \mathcal{H})$ **(6)**

We note the following points, which will be used later in the proof:

- $sl'' \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I}$ by the facts $sl'' \in \mathcal{H}$, $l'' < l'$, $sl' \in \mathcal{L}_m \cap \mathcal{I}_m$, and the fact $\mathcal{L}$ and $\mathcal{I}$ are closed languages. **(7)**
- $sl''\sigma \in \mathcal{I}$ by **(4)**, **(6)**, and fact $\mathcal{I}$ is closed. **(8)**

We now show that **(6)** implies $\sigma \in \Sigma_R$. We know:

- $\sigma \notin \Sigma_L$ as $\sigma \in \Sigma_L$ would imply $P_{IH}(sl''\sigma) = P_{IH}(sl'')$ Since $sl'' \in \mathcal{H}$, **Proposition 8**, **point a**, would then imply $sl''\sigma \in \mathcal{H}$, which would contradict **(6)**.
- $\sigma \notin \Sigma_A$ as $\sigma \in \Sigma_A$, $sl''\sigma \in \mathcal{I}$ (by **(8)**) and **point 3** of the *serial interface consistent* definition would imply $sl''\sigma \in \mathcal{H}$, which would contradict **(6)**.

As $\sigma \in \Sigma_{IL} = \Sigma_R \dot{\cup} \Sigma_A \dot{\cup} \Sigma_L$, we can thus conclude $\sigma \in \Sigma_R$ by process of elimination.

We now show that $\sigma \in \Sigma_R$ implies $sl'' \in \mathcal{I}_m$. This will allow us to use **Point 6** of the *serial interface consistent* definition to extend sl'' to a string marked by the *low level.*

From **Point 2** of the *serial interface consistent* definition, we have that DES G_I is a *command-pair interface.*

As $\sigma \in \Sigma_R$ and $sl''\sigma \in \mathcal{I}$ (from **(9)**), we can conclude: $P_I(sl''\sigma) = P_I(sl'')\sigma \in L(G_I)$

We can thus conclude by **Point B** of the *command-pair interface* definition that $P_I(sl'') \in L_m(G_I)$

$\Rightarrow sl'' \in \mathcal{I}_m$

From **(7)**, we have $sl'' \in \mathcal{L} \cap \mathcal{I}$ so we can now apply **Point 6** of the *serial interface consistent* definition and conclude:
$(\exists l''' \in \Sigma_L^*)$ s.t. $sl''l''' \in \mathcal{L}_m \cap \mathcal{I}_m$

As $l'' \in \Sigma_{IL}^*$ from **(6)**, we have $l''l''' \in \Sigma_{IL}^*$

We take $l = l''l'''$ and immediately have $sl \in \mathcal{L}_m \cap \mathcal{I}_m$ and $l \in \Sigma_{IL}^*$. All that remains is to show $sl \in \mathcal{H}$

From **(6)**, we have $sl'' \in \mathcal{H}$. As $l''' \in \Sigma_L^*$, we have $P_{IH}(sl'') = P_{IH}(sl''l''') = P_{IH}(sl)$

We can thus apply **Proposition 8**, **point a**, and conclude $sl \in \mathcal{H}$, as required.

QED

2.4.3 Proof of Proposition 13

Proof for **Proposition 13** on page 29: *If the system composed of DES G_H, G_L, and G_I is* serial level-wise nonblocking *and* serial interface consistent *with respect to the alphabet partition $\Sigma := \Sigma_H \dot\cup \Sigma_L \dot\cup \Sigma_R \dot\cup \Sigma_A$, then*

$(\forall s \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I})(\forall h \in \Sigma_H^*.\Sigma_R.\Sigma_H^*.\Sigma_A)$
$sh \in \mathcal{H} \cap \mathcal{I} \Rightarrow (\exists u \in \Sigma^*)\ s.t.\ (su \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I}_m) \wedge (P_{IH}(u) = h)$

Proof:

Assume system is *serial level-wise nonblocking* and *serial interface consistent.* **(1)**

Let $s \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I}$, $h \in \Sigma_H^*.\Sigma_R.\Sigma_H^*.\Sigma_A$, and $sh \in \mathcal{H} \cap \mathcal{I}$ **(2)**

We will now show this implies we can construct a string u with the desired properties.

We first note that $h \in \Sigma_H^*.\Sigma_R.\Sigma_H^*.\Sigma_A$ implies:
$(\exists h' \in \Sigma_H^*)(\rho \in \Sigma_R)(h'' \in \Sigma_H^*)(\alpha \in \Sigma_A)$ s.t. $h'\rho h''\alpha = h$ **(3)**

We will now show that we can construct a string $l \in \Sigma_L^*$ such that $h'\rho l h''\alpha \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I}_m$. We will also show that $P_{IH}(h'\rho l h''\alpha) = h$.

Our approach will be to show that $sh'\rho \in \mathcal{L} \cap \mathcal{I}$ and $\alpha \in \text{Elig}_{\mathcal{I}}(sh'\rho)$. We will then use **Point 5** of the *serial interface consistent* definition to construct a suitable string l.

We next note that $sh \in \mathcal{H} \cap \mathcal{I}$ (by **(2)**), and that $h'\rho \leq h$ (by **(3)**). As $\mathcal{H}$ and $\mathcal{I}$ are closed languages, we can now conclude:

$$sh'\rho \in \mathcal{H} \cap \mathcal{I} \qquad \textbf{(4)}$$

As $h' \in \Sigma_H^*$ (by **(3)**), we have $P_{IL}(sh') = P_{IL}(s)$. As $s \in \mathcal{L}$ (by **(2)**), we can now apply **Proposition 8**, **point c**, and conclude:

$$sh' \in \mathcal{L} \qquad \textbf{(5)}$$

We now have $sh' \in \mathcal{L} \cap \mathcal{I}$ and $\rho \in \text{Elig}_{\mathcal{I}}(sh')$ (by **(4)**).

This allows us to apply **Point 4** of the *serial interface consistent* definition (by **(1)**) and conclude:

$\rho \in \text{Elig}_{\mathcal{L}}(sh')$ and thus $sh'\rho \in \mathcal{L}$

$\Rightarrow sh'\rho \in \mathcal{L} \cap \mathcal{I}$, by **(4)**.

As $h'' \in \Sigma_H^*$ by **(3)**, we can conclude $P_I(sh'\rho h''\alpha) = P_I(sh'\rho)P_I(h'')P_I(\alpha) = P_I(sh'\rho\alpha)$

From **(2)** and **(3)**, we have $sh'\rho h''\alpha \in \mathcal{I}$. We can now apply **Proposition 8**, **point e**, and conclude:

$sh'\rho\alpha \in \mathcal{I}$

$\Rightarrow \alpha \in \text{Elig}_{\mathcal{I}}(sh'\rho)$.

We can now apply **Point 5** of the *interface consistency properties* and conclude:

$$(\exists l \in \Sigma_L^*) \text{ s.t. } \alpha \in \text{Elig}_{\mathcal{L}\cap\mathcal{I}}(sh'\rho l). \qquad \textbf{(6)}$$

$\Rightarrow sh'\rho l\alpha \in \mathcal{L} \cap \mathcal{I}$

As $h''in\Sigma_H^*$ by **(3)**, we can conclude $P_{IL}(sh'\rho l\alpha) = P_{IL}(sh'\rho lh''\alpha)$ and $P_I(sh'\rho l\alpha) = P_I(sh'\rho lh''\alpha)$. We can now apply **Proposition 8**, **points c** and **e**, and conclude:

$$sh'\rho lh''\alpha \in \mathcal{L} \cap \mathcal{I} \qquad \textbf{(7)}$$

We next note that DES G_I is a *command-pair interface* by **(1)**.

As $\alpha \in \Sigma_A$ (by **(3)**), we can now conclude:

$P_I(sh'\rho lh''\alpha) \in \Sigma_I^*.\Sigma_A \cap L(G_I)$

$\Rightarrow P_I(sh'\rho lh''\alpha) \in L_m(G_I)$ by **point D** of the *command-pair interface* definition.

$\Rightarrow sh'\rho lh''\alpha \in \mathcal{I}_m$ **(8)**

From **(2)** and **(3)**, we have $sh'\rho h''\alpha \in \mathcal{H}$. As $l \in \Sigma_L^*$ (by **(6)**), we can conclude:

$$P_{IH}(sh'\rho h''\alpha) = P_{IH}(sh'\rho lh''\alpha). \quad \textbf{(9)}$$

We can now apply **Proposition 8**, **point a**, and conclude:

$sh'\rho lh''\alpha \in \mathcal{H}$

Combining with **(7)**, **(8)** and **(9)**, we have $sh'\rho lh''\alpha \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I}_m$ and $P_{IH}(h'\rho lh''\alpha) = P_{IH}(h'\rho h''\alpha) = h$

We take $u = h'\rho lh''\alpha$, and the proof is complete.

QED

2.4.4 Proof of Proposition 14

Proof for **Proposition 14** on page 30: *If the system composed of DES G_H, G_L, and G_I is* serial level-wise nonblocking *and* serial interface consistent *with respect to the alphabet partition* $\Sigma := \Sigma_H \dot\cup \Sigma_L \dot\cup \Sigma_R \dot\cup \Sigma_A$, *then*

$(\forall s \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I}_m)(\forall h \in \Sigma_{IH}^*.\Sigma_A)$
$\quad sh \in \mathcal{H} \cap \mathcal{I} \Rightarrow (\exists u \in \Sigma^*)\ (P_{IH}(u) = h) \wedge (su \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I}_m)$

Proof:

Assume system is *serial level-wise nonblocking* and *serial interface consistent.* **(1)**

Let $s \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I}_m$, $h \in \Sigma_{IH}^*.\Sigma_A$, and $sh \in \mathcal{H} \cap \mathcal{I}$ **(2)**

We will now show this implies we can construct a string u with the desired properties.

Our approach will be to break string h into substrings containing pairs of *request* and *answer events*. We will then construct u iteratively, using these substrings.

We first let n be the number of *answer events* in string h ($n \geq 1$ as $h \in \Sigma_{IH}^*.\Sigma_A$). This implies:
$(\exists h_1, h_2, \ldots, h_n \in (\Sigma_H \cup \Sigma_R)^*.\Sigma_A)$ s.t. $h_1 h_2 \ldots h_n = h$ **(3)**

We have thus broken h into n strings each containing one answer event at the end of the string. From **(2)**, we immediately have: $sh_1h_2 \ldots h_n \in \mathcal{H} \cap \mathcal{I}$ **(4)**

Using an inductive proof, we will now show:

$(\exists u_0, u_1, \ldots, u_n \in \Sigma^*)$ s.t. $(su_0u_1 \ldots u_n \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I}_m) \wedge (P_{IH}(u_0, u_1, \ldots, u_n) = h_0h_1 \ldots h_n)$, where $h_0 := \epsilon$

Claim to be proven:

For $k \in \{0, 1, \ldots, n\}$, there exists $u_0, u_1, \ldots, u_k \in \Sigma^*$ such that the following are true: **(5)**

(a) $P_{IH}(u_0u_1 \ldots u_k) = h_0h_1 \ldots h_k$

(b) $su_0u_1 \ldots u_k \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I}_m$

We will first prove the initial case ($k = 0$), and then the general case of $k \in \{1, \ldots, n\}$. We can then conclude by induction that the claim has been proven.

Initial Case: $k = 0$

We take $u_0 = \epsilon$ and we immediately have $P_{IH}(u_0) = \epsilon = h_0$, and thus **Property a** of **(5)** is satisfied.

We have $su_0 \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I}_m$ as $su_0 = s$ and $s \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I}_m$ (by **(2)**), and thus **Property b** of **(5)** is satisfied.

Initial case complete.

Inductive Step:

Let $k \in \{1, \ldots, n\}$. Assume $\exists u_0, u_1, \ldots, u_{k-1} \in \Sigma^*$ and that they satisfy **Properties a** and **b** of **(5)** when $k - 1$ is substituted for k. **(6)**

We will show that this implies $\exists u_k \in \Sigma^*$ that satisfies **Properties a** and **b** of **(5)**.

Our approach will be to apply **Proposition 13**. To do this, our first step is to show that $su_0u_1 \ldots u_{k-1}h_k \in \mathcal{H} \cap \mathcal{I}$

We first note that $sh_0h_1 \ldots h_k \in \mathcal{H} \cap \mathcal{I}$ by **(4)**, and the facts that $h_0 = \epsilon$ and $\mathcal{H}$ and $\mathcal{I}$ are closed languages. **(7)**

From **(6)**, we have $P_{IH}(u_0u_1 \ldots u_{k-1}) = h_0h_1 \ldots h_{k-1}$. From **(3)** and fact $h_0 = \epsilon$, we have $P_{IH}(h_0h_1 \ldots h_k) = h_0h_1 \ldots h_k$. We can thus conclude:

$$\begin{aligned} P_{IH}(su_0u_1 \ldots u_{k-1}h_k) &= P_{IH}(s)P_{IH}(u_0u_1 \ldots u_{k-1})P_{IH}(h_k) \\ &= P_{IH}(s)h_0h_1 \ldots h_{k-1}P_{IH}(h_k) \\ &= P_{IH}(sh_0h_1 \ldots h_k) \qquad (8) \end{aligned}$$

With **(7)**, we can now apply **Proposition 8**, **point a**, and conclude $su_0u_1 \ldots u_{k-1}h_k \in \mathcal{H}$

(9)

As $\Sigma_I \subseteq \Sigma_{IH}$, we can conclude by **(8)** that:

$$P_I(su_0u_1 \dots u_{k-1}h_k) = P_I(sh_0h_1 \dots h_k)$$

With **(7)**, we can now apply **Proposition 8**, **point e**, and conclude $su_0u_1 \dots u_{k-1}h_k \in \mathcal{I}$

Combining with **(9)**, we have $su_0u_1 \dots u_{k-1}h_k \in \mathcal{H} \cap \mathcal{I}$ **(10)**

We will now show that $h_k \in \Sigma_H^*.\Sigma_R.\Sigma_H^*.\Sigma_A$. It is sufficient to show that $P_I(h_k) \in \Sigma_R.\Sigma_A$.

We first note that $P_I(su_0u_1 \dots u_{k-1}) \in L_m(G_I)$ as $su_0u_1 \dots u_{k-1} \in \mathcal{I}_m$, by **(6)**. **(11)**

Similarly by **(10)**, we have $P_I(su_0u_1 \dots u_{k-1}h_k) \in L(G_I)$. **(12)**

We now note that $h_k \in (\Sigma_H \cup \Sigma_R)^*.\Sigma_A$ (by **(3)**) implies that $P_I(h_k) \in \Sigma_R^*.\Sigma_A$

We next note that DES G_I is a *command-pair interface* by **(1)**.

From **(11)**, we can conclude $P_I(su_0u_1 \dots u_{k-1}) \in \{\epsilon\} \cup (\Sigma_I^*.\Sigma_A \cap L(G_I))$ by **point D** of the *command-pair interface* definition. **(13)**

This implies that either $P_I(su_0u_1 \dots u_{k-1}) = \epsilon$ or $P_I(su_0u_1 \dots u_{k-1})$ ends in an *answer event.*

In either case, **point E** of the *command-pair interface* definition implies that $P_I(su_0u_1 \dots u_{k-1})$ can only be extended to a string in $L(G_I)$ by a *request event.* **(14)**

From **(12)**, we have $P_I(su_0u_1 \dots u_{k-1}h_k) = P_I(su_0u_1 \dots u_{k-1})P_I(h_k) \in L(G_I)$

$\Rightarrow P_I(h_k) \in \Sigma_R.\Sigma_R^*.\Sigma_A$ since $P_I(h_k) \in \Sigma_R^*.\Sigma_A$ and therefore must contain an *answer event.* By **(14)**, we know that the *answer event* must be preceded by at least one *request event.*

From **point E** of the *command-pair interface* definition, we know that, in $L(G_I)$, a *request event* must be followed by an *answer event,* before another *request event* can occur.
$\Rightarrow P_I(h_k) \in \Sigma_R.\Sigma_A$

$\Rightarrow h_k \in \Sigma_H^*.\Sigma_R.\Sigma_H^*.\Sigma_A$

We may now apply **Proposition 13** by taking $su_0u_1 \dots u_{k-1}$ to be string s, and h_k to be string h in that proposition. We thus conclude:
$(\exists u' \in \Sigma^*)$ s.t. $(su_0u_1 \dots u_{k-1}u' \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I}_m) \wedge (P_{IH}(u') = h_k)$

We can also conclude $P_{IH}(u_0u_1 \dots u_{k-1}u') = h_0h_1 \dots h_k$ by **(6)** and the fact P_{IH} is concatenative.

We now take $u_k = u'$ and we have $u_k \in \Sigma^*$ and we have it satisfying **Properties a** and **b** of **(5)**.

Inductive step complete.

We have now proven the **initial case** and the **inductive step**. We now conclude that the **Claim** is true, by induction.

We thus take $k = n$ and have $u_0, u_1, \ldots, u_n \in \Sigma^*$, $su_0u_1 \ldots u_n \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I}_m$, and $P_{IH}(u_0u_1 \ldots u_n) = h_0h_1 \ldots h_n = h$

We thus take $u = u_0u_1 \ldots u_n$, and the proof is complete.

QED

2.4.5 Proof of Proposition 15

Proof for **Proposition 15** on page 30: *If the system composed of DES G_H, G_L, and G_I is* serial level-wise nonblocking *and* serial interface consistent *with respect to the alphabet partition $\Sigma := \Sigma_H \dot{\cup} \Sigma_L \dot{\cup} \Sigma_R \dot{\cup} \Sigma_A$, then*

$(\forall s \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I}_m)(\forall h \in \Sigma_{IH}^*)$

$sh \in \mathcal{H}_m \cap \mathcal{I}_m \Rightarrow (\exists u \in \Sigma^*)\ s.t.\ (su \in \mathcal{H}_m \cap \mathcal{L}_m \cap \mathcal{I}_m) \wedge (P_{IH}(u) = h) \wedge (P_I(u) \in \{\epsilon\} \cup \Sigma_R.\Sigma_I^*)$

Proof:

Assume system is *serial level-wise nonblocking* and *serial interface consistent.* **(1)**

Let $s \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I}_m$, $h \in \Sigma_{IH}^*$, and $sh \in \mathcal{H}_m \cap \mathcal{I}_m$ **(2)**

We will now show this implies we can construct a string u with the desired properties.

We have two cases to examine:
I) $h \in \Sigma_H^*$ (string h does not contain any *interface events*)
II) $h \notin \Sigma_H^*$ (string h contains one or more *interface events*)

Case I) $h \in \Sigma_H^*$

From **(2)**, we have $s \in \mathcal{L}$. As $h \in \Sigma_H^*$, we can conclude $P_{IL}(s) = P_{IL}(sh)$

We can thus apply **Proposition 8**, **point c**, and conclude $sh \in \mathcal{L}$

Combining with **(2)**, we thus have $sh \in \mathcal{L} \cap \mathcal{H}_m \cap \mathcal{I}_m$

We can now apply **Proposition 12** and conclude:

$(\exists l \in \Sigma_L^*)\ shl \in \mathcal{H}_m \cap \mathcal{L}_m \cap \mathcal{I}_m$

We take $u = hl$ and we immediately have $su \in \mathcal{H}_m \cap \mathcal{L}_m \cap \mathcal{I}_m$, $P_{IH}(u) = h$, and $P_I(u) = \epsilon \in \{\epsilon\} \cup \Sigma_R.\Sigma_I^*$

Case I complete.

Case II) $h \notin \Sigma_H^*$

This implies $P_I(h) \neq \epsilon$ **(3)**

As string h contains one or more *interface events*, events the two levels share, we must construct a string so that they will both accept the interface events, and arrive in a marked state together.

Our approach will be first to apply **Proposition 14** to enable us to construct a string u such that $su \in \mathcal{L} \cap \mathcal{H}_m \cap \mathcal{I}_m$. We will then use **Proposition 12** to show that $su \in \mathcal{H}_m \cap \mathcal{L}_m \cap \mathcal{I}_m$.

Our first step will to be to construct a string $h' \leq h$, $h' \in \Sigma_{IH}^*.\Sigma_A$, so that we can apply **Proposition 14** .

To construct a suitable h', we will start by showing that $P_I(h) \in \Sigma_R.\Sigma_I^*.\Sigma_A$

We first note that DES G_I is a *command-pair interface* by **(1)**.

From **(2)**, we have $s \in \mathcal{I}_m$

$\Rightarrow P_I(s) \in L_m(G_I)$

$\Rightarrow P_I(s) \in \{\epsilon\} \cup (\Sigma_I^*.\Sigma_A \cap L(G_I))$ by **point D** of the *command-pair interface* definition. **(4)**

This implies that either $P_I(s) = \epsilon$ or $P_I(s)$ ends in an *answer event.*

In either case, **point E** of the definition implies that $P_I(s)$ can only be extended to a string in $L(G_I)$ by a *request event.* **(5)**

Now, from **(2)** we also have $sh \in \mathcal{I}_m$.

$\Rightarrow P_I(sh) = P_I(s)P_I(h) \in L_m(G_I)$

As $P_I(h) \neq \epsilon$ (from **(3)**), we can conclude by **(5)** that $P_I(h) \in \Sigma_R.\Sigma_I^*$ **(6)**

As $P_I(s)P_I(h) \in L_m(G_I)$, we can conclude by **point D** of the *command-pair interface* definition that $P_I(s)P_I(h) \in \Sigma_I^*.\Sigma_A$

Combining with **(6)**, we can conclude $P_I(h) \in \Sigma_R.\Sigma_I^*.\Sigma_A$ **(7)**

$\Rightarrow h \in P_I^{-1}(\Sigma_R.\Sigma_I^*.\Sigma_A)$

$\Rightarrow h \in \Sigma_H^*.\Sigma_R.\Sigma_{IH}^*.\Sigma_A.\Sigma_H^*$ as $h \in \Sigma_{IH}^*$ **(8)**

$\Rightarrow (\exists h' \in \Sigma_{IH}^*.\Sigma_A)(h'' \in \Sigma_H^*)$ s.t. $h'h'' = h$ **(9)**

We can now conclude $sh' \in \mathcal{H} \cap \mathcal{I}$ as $sh \in \mathcal{H}_m \cap \mathcal{I}_m$, by **(2)** and the fact that $\mathcal{H}$ and $\mathcal{I}$ are closed languages.

We can now apply **Proposition 14** by taking h' to be string h in that proposition. We can now conclude:

$(\exists u' \in \Sigma^*)$ s.t. $(P_{IH}(u') = h') \wedge (su' \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I})$ **(10)**

We note for future use that $P_{IH}(u') = h'$ and the fact that $\Sigma_I \subseteq \Sigma_{IH}$ implies that $P_I(u') = P_I(h')$.
From **(8)**, we have $h \in \Sigma_H^*.\Sigma_R.\Sigma_{IH}^*.\Sigma_A.\Sigma_H^*$. **(11)**

We can thus conclude by **(9)** that $P_I(u') \in \Sigma_R.\Sigma_I^*$ **(12)**

We will now show that $su'h'' \in \mathcal{L} \cap \mathcal{H}_m \cap \mathcal{I}_m$ so that we can apply **Proposition 12** .

From **(10)**, we have $su' \in \mathcal{L}$. From **(9)**, we have $h'' \in \Sigma_H^*$. We can thus conclude:

$$P_{IL}(su'h'') = P_{IL}(su')$$

We can now apply **Proposition 8**, **point c**, and conclude $su'h'' \in \mathcal{L}$ **(13)**

From **(2)**, we have $sh \in \mathcal{H}_m \cap \mathcal{I}_m$. From **(9)**, we can conclude:

$$sh'h'' \in \mathcal{H}_m \cap \mathcal{I}_m \quad \textbf{(14)}$$

As $P_{IH}(u') = h'$ (from **(10)**), we have:

$$P_{IH}(sh'h'') = P_{IH}(s)P_{IH}(h')P_{IH}(h'') = P_{IH}(s)P_{IH}(u')P_{IH}(h'') = P_{IH}(su'h'').$$

We can now apply **Proposition 8**, **point b**, and conclude $su'h'' \in \mathcal{H}_m$ **(15)**

From **(11)**, we have $P_I(u') = P_I(h')$. We can argue as above and conclude $P_I(sh'h'') = P_I(su'h'')$

We can now apply **Proposition 8**, **point f**, and conclude $su'h'' \in \mathcal{I}_m$

Combining with **(13)** and**(15)**, we can conclude:

$$su'h'' \in \mathcal{L} \cap \mathcal{H}_m \cap \mathcal{I}_m$$

We can now apply **Proposition 12** by taking $su'h''$ to be string s in that proposition. We thus conclude:

$(\exists l \in \Sigma_L^*)$ s.t. $su'h''l \in \mathcal{H}_m \cap \mathcal{L}_m \cap \mathcal{I}_m$ **(16)**

We thus take $u = u'h''l$ and we have $P_{IH}(u) = P_{IH}(u'h''l) = h'h''\epsilon = h$ and $su \in \mathcal{H}_m \cap \mathcal{L}_m \cap \mathcal{I}_m$.

From **(12)**, we have $P_I(u') \in \Sigma_R.\Sigma_I^*$. We thus have:

$P_I(u) = P_I(u'h''l) = P_I(u') \in \Sigma_R.\Sigma_I^*$, as $h'' \in \Sigma_H^*$ (by **(9)**) and $l \in \Sigma_L^*$ (by **(16)**)

Case II is now complete.

By **Cases I** and **II**, we now have constructed a string $u \in \Sigma^*$ such that $P_I(u) \in \{\epsilon\} \cup \Sigma_R.\Sigma_I^*$, $P_{IH}(u) = h$ and $su \in \mathcal{H}_m \cap \mathcal{L}_m \cap \mathcal{I}_m$, as required.

QED

Chapter 3

Serial Case: Controllability

Now that we have discussed nonblocking in the serial interface setting, we now consider controllability. In the remainder of this chapter, we will define our setting and notation and then present some supporting propositions, followed by the serial controllability theorem.

3.1 Definitions and Notation

We now present some definitions and notation that will be useful in simplifying proofs. When we discussed nonblocking, we were only concerned with the *high* and *low level subsystems,* ignoring distinctions between plants and supervisors. For controllability, we need to split the subsystems into their plant and supervisor components. We will do so as shown in Figure 3.1. We define the *high level plant* to be DES $\mathcal{G}_H$, and the *high level supervisor* to be $\mathcal{S}_H$ (both defined over event set Σ_{IH}). Similarly, the *low level plant* and *supervisor* are $\mathcal{G}_L$ and $\mathcal{S}_L$ (defined over event set Σ_{IL}). To be consistent with our definitions in Chapter 2, we define the following identities for the *high* and *low level subsystems* as follows:

$$G_H := \mathcal{G}_H||_s\mathcal{S}_H \qquad\qquad G_L := \mathcal{G}_L||_s\mathcal{S}_L$$

We now have two ways to describe our system for the serial case, depending on the level of detail required. We will call the original method described in Chapter 2 in terms of an *interface* and *high* and *low subsystems*, the *serial subsystem based form.* This form is useful as it simplifies nonblocking definitions and proofs. We call the above method, given in terms of an interface and plants and supervisors, the *serial general form* as the *serial subsystem based form* can be recovered by applying the above

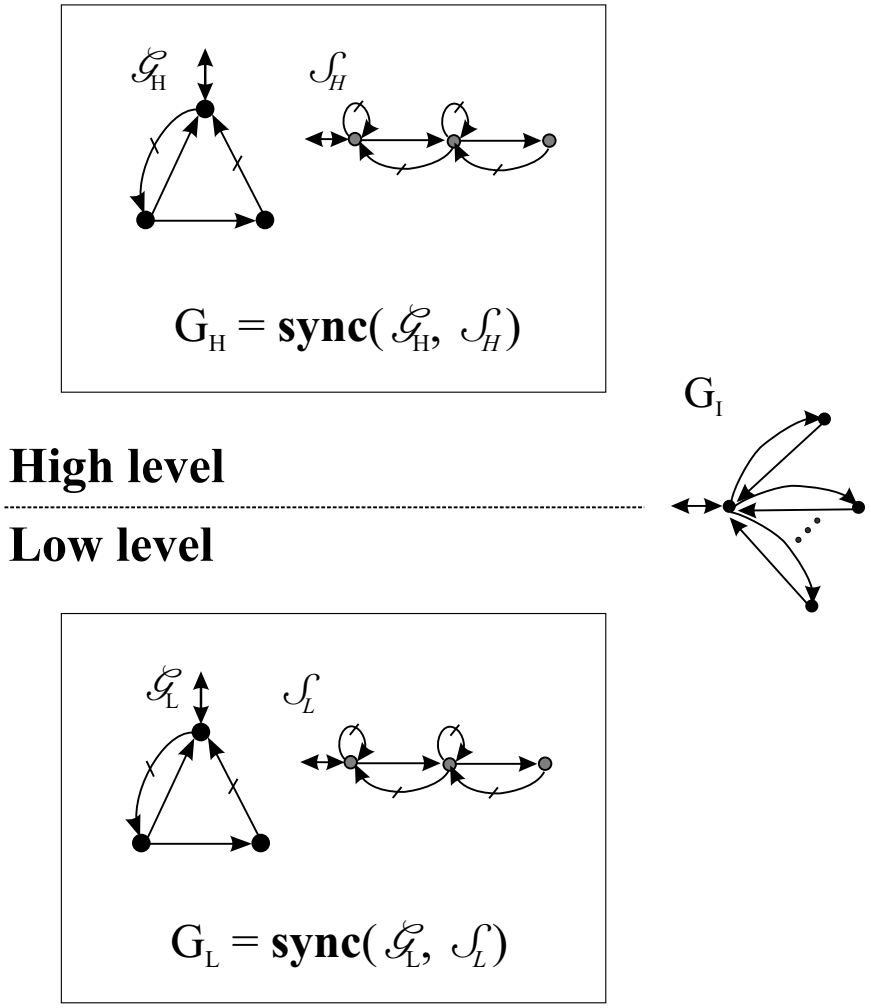


Figure 3.1: Plant and Supervisor Subplant Decomposition

identities for G_H and G_L. When we refer terms applicable to both forms (e.g. the *high level*), we will simply state the term, allowing the type of the system to make our meaning clear.

Our next step is to define the *flat supervisor* and *plant* for our system. By *flat supervisor* and *plant*, we refer to the equivalent DES supervisor and plant that would represent our system if we ignored the interface structure. The are defined as follows:

$$\textbf{Plant} := \mathcal{G}_H||_s\mathcal{G}_L \qquad\qquad \textbf{Sup} := \mathcal{S}_H||_s\mathcal{S}_L||_sG_I$$

In the above definition, we have taken the interface, G_I, as a supervisor. This is to recognise the fact that G_I is a specification of services that the *low level* is to provide to the *high level.* As such, low

level supervisors or usually required to implement these services. By treating G_I as a supervisor, we can verify to make sure that the desired pattern of *request* and *answer events* is achievable.

We next want to express the languages of **Plant** and **Sup** in terms of their components. To do this, we need to first define the following useful languages:

$$\begin{aligned} \mathbf{H} &:= P_{IH}^{-1}L(\mathcal{G}_H), \qquad \mathbf{H}_\mathcal{S} := P_{IH}^{-1}L(\mathcal{S}_H), \qquad \subseteq \Sigma^* \\ \mathbf{L} &:= P_{IL}^{-1}L(\mathcal{G}_L), \qquad \mathbf{L}_\mathcal{S} := P_{IL}^{-1}L(\mathcal{S}_L), \qquad \subseteq \Sigma^* \end{aligned}$$

We can now express the languages of **Plant** and **Sup** as follows:

$$L(\mathbf{Plant}) = \mathbf{H} \cap \mathbf{L} \qquad\qquad L(\mathbf{Sup}) = \mathbf{H}_\mathcal{S} \cap \mathbf{L}_\mathcal{S} \cap \mathcal{I}$$

This allows us to present the proposition below that collects together several similar propositions. As it will be common in the proofs in this report to show that membership in languages such as **H** are dependent only on events in specific subsets (for **H**, events in subset Σ_{IH}), this proposition will be very useful.

Proposition 16

(a) $(\forall s, s' \in \Sigma^*)\ s \in \mathbf{H}$ *and* $P_{IH}(s) = P_{IH}(s') \Rightarrow s' \in \mathbf{H}$

(b) $(\forall s, s' \in \Sigma^*)\ s \in \mathbf{H}_\mathcal{S}$ *and* $P_{IH}(s) = P_{IH}(s') \Rightarrow s' \in \mathbf{H}_\mathcal{S}$

(c) $(\forall s, s' \in \Sigma^*)\ s \in \mathbf{L}$ *and* $P_{IL}(s) = P_{IL}(s') \Rightarrow s' \in \mathbf{L}$

(d) $(\forall s, s' \in \Sigma^*)\ s \in \mathbf{L}_\mathcal{S}$ *and* $P_{IL}(s) = P_{IL}(s') \Rightarrow s' \in \mathbf{L}_\mathcal{S}$

Proof:

Points a-d:

Identical to the proof of **point a** of **Proposition 8**, after substitution.

QED

3.2 Serial Level-wise Controllability

The goal in this chapter is to develop a means of verifying that our system's *flat supervisor* is controllable for the *flat plant* that uses only local checks. To do this, we will use the *serial level-wise controllable* definition.

Serial Level-wise Controllable: The system composed of plant components $\mathcal{G}_H$, $\mathcal{G}_L$, supervisors $\mathcal{S}_H$, $\mathcal{S}_L$, and interface G_I, is said to be *serial level-wise controllable* with respect to the alphabet partition $\Sigma := \Sigma_H \dot{\cup} \Sigma_L \dot{\cup} \Sigma_R \dot{\cup} \Sigma_A$, if the following conditions are satisfied:

(I) The alphabet of $\mathcal{G}_H$ and $\mathcal{S}_H$ is Σ_{IH}, the alphabet of $\mathcal{G}_L$ and $\mathcal{S}_L$ is Σ_{IL}, and the alphabet of G_I is Σ_I

(II) $(\forall s \in \mathbf{L} \cap \mathbf{L}_{\mathcal{S}} \cap \mathcal{I})\ \ \mathrm{Elig}_{\mathbf{L}}(s) \cap \Sigma_u \subseteq \mathrm{Elig}_{\mathbf{L}_{\mathcal{S}} \cap \mathcal{I}}(s)$ *Controllability at the low level*

(III) $(\forall s \in \mathbf{H} \cap \mathcal{I} \cap \mathbf{H}_{\mathcal{S}})\ \ \mathrm{Elig}_{\mathbf{H} \cap \mathcal{I}}(s) \cap \Sigma_u \subseteq \mathrm{Elig}_{\mathbf{H}_{\mathcal{S}}}(s)$ *Controllability at the high level*

To summarise the definition, **Point I** simply states that the plants, supervisors, and interface have the indicated event sets. This is in essence restricting the control actions allowable by the supervisors to their specified alphabets. For example, this implies that $\mathcal{S}_H$ is forbidden to disable any low level events.

The next point states that the *interface* and $\mathcal{S}_L$ are together controllable for the low level plant $\mathcal{G}_L$. In other words, we are treating the *low level* as a self contained system and performing a standard controllability test for the modular supervisor $\mathcal{S}_L \wedge G_I$ with respect to the plant $\mathcal{G}_L$.

The last point states that supervisor $\mathcal{S}_H$ is controllable for the high level plant $\mathcal{G}_H$, when it is already under the control of the *interface.* In other words, we are treating the *high level* as a self contained system and performing a standard controllability test for supervisor $\mathcal{S}_H$ with respect to the composite plant $\mathcal{G}_H||_s G_I$. By treating the *interface* as a plant at the *high level*, we allow the high level supervisor, $\mathcal{S}_H$, to be more flexible as the *interface* may have more information about when interface events are eligible than the high level plant.

We are now ready to state the proposition below which establishes useful properties for often used languages.

Proposition 17 *If the system composed of plant components $\mathcal{G}_H$, $\mathcal{G}_L$, supervisors $\mathcal{S}_H$, $\mathcal{S}_L$, and interface G_I, is* serial level-wise controllable *with respect to the alphabet partition $\Sigma := \Sigma_H \dot{\cup} \Sigma_L \dot{\cup} \Sigma_R \dot{\cup} \Sigma_A$, then languages $\mathbf{H}$, $\mathbf{H}_{\mathcal{S}}$, $\mathbf{L}$, $\mathbf{L}_{\mathcal{S}}$, $\mathcal{I}$, $L(\mathbf{Plant})$, and $L(\mathbf{Sup})$ are closed.*

Proof: See page 50.

3.3 Theorem and Propositions

We are now ready to present our main results for this chapter. We will first present two supporting propositions, followed by our serial case controllability theorem.

3.3.1 Low level Controllability Proposition

Our first proposition asserts that if the system is *serial level-wise controllable*, then G_I and $\mathcal{S}_L$ are together controllable for the system's *flat plant.*

Proposition 18 *If the system composed of plant components* $\mathcal{G}_H$, $\mathcal{G}_L$, *supervisors* $\mathcal{S}_H$, $\mathcal{S}_L$, *and interface* G_I, *is* serial level-wise controllable *with respect to the alphabet partition* $\Sigma := \Sigma_H \dot{\cup} \Sigma_L \dot{\cup} \Sigma_R \dot{\cup} \Sigma_A$, *then:*

$$(\forall s \in L(\mathbf{Plant}) \cap \mathbf{L}_{\mathcal{S}} \cap \mathcal{I}) \quad Elig_{L(\mathbf{Plant})}(s) \cap \Sigma_u \subseteq Elig_{\mathbf{L}_{\mathcal{S}} \cap \mathcal{I}}(s)$$

where $\mathbf{Plant} = \mathcal{G}_H ||_s \mathcal{G}_L$

Proof: See page 51.

3.3.2 High Level Controllability Proposition

The last proposition asserts that if the system is *serial level-wise controllable*, then $\mathcal{S}_H$ is controllable for our *flat plant* when it is already under the control of the *interface.*

Proposition 19 *If the system composed of plant components* $\mathcal{G}_H$, $\mathcal{G}_L$, *supervisors* $\mathcal{S}_H$, $\mathcal{S}_L$, *and interface* G_I, *is* serial level-wise controllable *with respect to the alphabet partition* $\Sigma := \Sigma_H \dot{\cup} \Sigma_L \dot{\cup} \Sigma_R \dot{\cup} \Sigma_A$, *then:*

$$(\forall s \in L(\mathbf{Plant}) \cap \mathcal{I} \cap \mathbf{H}_{\mathcal{S}}) \quad Elig_{L(\mathbf{Plant}) \cap \mathcal{I}}(s) \cap \Sigma_u \subseteq Elig_{\mathbf{H}_{\mathcal{S}}}(s)$$

where $\mathbf{Plant} = \mathcal{G}_H ||_s \mathcal{G}_L$

Proof: See page 52.

3.3.3 Serial Controllability Theorem

We now present our main result for this chapter, the serial controllability theorem. In essence, this theorem asserts that if the system is *serial level-wise controllable*, then controllability can be checked

for each level separately in order to determine that the system's *flat supervisor* is controllable for the system's *flat plant.* As the *serial level-wise controllable* definition can be evaluated by examining only one level of our system at a time, we now have a means to verify controllability of our system using local checks.

Theorem 2 *If the system composed of plant components $\mathcal{G}_H$, $\mathcal{G}_L$, supervisors $\mathcal{S}_H$, $\mathcal{S}_L$, and interface G_I, is* serial level-wise controllable *with respect to the alphabet partition $\Sigma := \Sigma_H \dot\cup \Sigma_L \dot\cup \Sigma_R \dot\cup \Sigma_A$, then:*

$$(\forall s \in L(\mathbf{Plant}) \cap L(\mathbf{Sup})) \quad Elig_{L(\mathbf{Plant})}(s) \cap \Sigma_u \subseteq Elig_{L(\mathbf{Sup})}(s)$$

where $\mathbf{Plant} = \mathcal{G}_H ||_s \mathcal{G}_L$ *and* $\mathbf{Sup} = \mathcal{S}_H ||_s \mathcal{S}_L ||_s G_I$.

Proof:

Assume system is *serial level-wise controllable* **(1)**

Let $s \in L(\mathbf{Plant}) \cap L(\mathbf{Sup})$ and $\sigma \in \text{Elig}_{L(\mathbf{Plant})}(s) \cap \Sigma_u$ **(2)**

Will now show this implies $\sigma \in \text{Elig}_{L(\mathbf{Sup})}(s)$

From **(2)**, we have $s \in L(\mathbf{Plant}) \cap L(\mathbf{Sup}) = \mathbf{H} \cap \mathbf{L} \cap \mathbf{H}_\mathcal{S} \cap \mathbf{L}_\mathcal{S} \cap \mathcal{I}$ **(3)**

We also have $s\sigma \in L(\mathbf{Plant}) = \mathbf{H} \cap \mathbf{L}$ **(4)**

As $L(\mathbf{Sup}) = \mathbf{H}_\mathcal{S} \cap \mathbf{L}_\mathcal{S} \cap \mathcal{I}$, it is sufficient to show that $s\sigma \in \mathbf{H}_\mathcal{S} \cap \mathbf{L}_\mathcal{S} \cap \mathcal{I}$

From **(3)**, we have $s \in \mathbf{H} \cap \mathbf{L} \cap \mathbf{L}_\mathcal{S} \cap \mathcal{I} = L(\mathbf{Plant}) \cap \mathbf{L}_\mathcal{S} \cap \mathcal{I}$

From **(2)**, we have $\sigma \in \text{Elig}_{L(\mathbf{Plant})}(s) \cap \Sigma_u$. We can thus conclude by **Proposition 18** that $\sigma \in \text{Elig}_{\mathbf{L}_\mathcal{S} \cap \mathcal{I}}(s)$

$\Rightarrow s\sigma \in \mathbf{L}_\mathcal{S} \cap \mathcal{I}$ **(5)**

All that remains is to show that $s\sigma \in \mathbf{H}_\mathcal{S}$

Using **(4)** and **(5)**, we have $s\sigma \in \mathbf{H} \cap \mathbf{L} \cap \mathcal{I}$

$\Rightarrow s\sigma \in L(\mathbf{Plant}) \cap \mathcal{I}$

$\Rightarrow \sigma \in \text{Elig}_{L(\mathbf{Plant}) \cap \mathcal{I}}(s) \cap \Sigma_u$

From **(5)**, we have $s \in L(\mathbf{Plant}) \cap \mathcal{I} \cap \mathbf{H}_\mathcal{S}$

We can thus conclude by **Proposition 19** that $\sigma \in \text{Elig}_{\mathbf{H}_\mathcal{S}}(s)$

$\Rightarrow s\sigma \in \mathbf{H}_\mathcal{S}$

Combining with **(5)**, we have $s\sigma \in \mathbf{H}_\mathcal{S} \cap \mathbf{L}_\mathcal{S} \cap \mathcal{I}$, as required.

QED

3.3.4 Software Tool

To aid in investigating *hierarchical interface-based supervisory control,* we have developed software routines to verify that a system satisfies the conditions below. The routines were developed by Leduc during his collaboration with Siemens Corporate Research. The routines are an experimental add-on to Siemen's Valid software program.

- DES G_I satisfies the *star interface* definition.
- *Serial level-wise nonblocking*
- *Serial level-wise controllable*
- *Serial interface consistent,* using the *serial interface strict marking* condition to check for **Property 6**.

3.4 Proofs of Selected Propositions

In order to make this work more readable, the proofs of some propositions in this chapter were not given as the propositions were introduced. They are now presented in the following sections.

3.4.1 Proof of Proposition 17

Proof for **Proposition 17** on page 47: If the system composed of plant components $\mathcal{G}_H$, $\mathcal{G}_L$, supervisors $\mathcal{S}_H$, $\mathcal{S}_L$, and interface G_I, is *serial level-wise controllable* with respect to the alphabet partition $\Sigma := \Sigma_H \dot\cup \Sigma_L \dot\cup \Sigma_R \dot\cup \Sigma_A$, then languages $\mathbf{H}$, $\mathbf{H}_\mathcal{S}$, $\mathbf{L}$, $\mathbf{L}_\mathcal{S}$, $\mathcal{I}$, $L(\mathbf{Plant})$, and $L(\mathbf{Sup})$ are closed.

Proof:

Assume system is *serial level-wise controllable.* **(1)**

Will now show this implies that the indicated languages are closed.

We first note that by **(1)**, the system is *serial level-wise controllable.* By **Point 1** of this definition, we can conclude:

$$L(\mathcal{G}_H), L(\mathcal{S}_H) \subseteq \Sigma_{IH}^*,\ L(\mathcal{G}_L), L(\mathcal{S}_L) \subseteq \Sigma_{IL}^*, \text{ and } L(G_I) \subseteq \Sigma_I^*.$$

This tells us that languages $\mathbf{H} = P_{IH}^{-1}(L(\mathcal{G}_H))$, $\mathbf{H}_\mathcal{S} = P_{IH}^{-1}L(\mathcal{S}_H)$, $\mathbf{L} = P_{IL}^{-1}(L(\mathcal{G}_L))$, $\mathbf{L}_\mathcal{S} = P_{IL}^{-1}L(\mathcal{S}_L)$, $\mathcal{I} = P_I^{-1}(L(G_I))$ are defined.

We will start by showing that languages $\mathbf{H}$, $\mathbf{H}_\mathcal{S}$, $\mathbf{L}$, $\mathbf{L}_\mathcal{S}$, and $\mathcal{I}$ are closed.

We now note that languages $L(\mathcal{G}_H)$, $L(\mathcal{S}_H)$, $L(\mathcal{G}_L)$, $L(\mathcal{S}_L)$, and $L(G_I)$ are closed by the definition of the closed behaviour of a DES.

We can now apply **Proposition 1** repeatedly and conclude that $\mathbf{H}$, $\mathbf{H}_\mathcal{S}$, $\mathbf{L}$, $\mathbf{L}_\mathcal{S}$, and $\mathcal{I}$ are closed, as required. **(2)**

We will now show that languages $L(\mathbf{Plant})$, and $L(\mathbf{Sup})$ are closed.

We now note that $L(\mathbf{Plant}) = \mathbf{H} \cap \mathbf{L}$ and $L(\mathbf{Sup}) = \mathbf{H}_\mathcal{S} \cap \mathbf{L}_\mathcal{S} \cap \mathcal{I}$.

Combining with **(2)**, we can now apply **Proposition 2** repeatedly and conclude that $L(\mathbf{Plant})$, and $L(\mathbf{Sup})$ are closed, as required.

QED

3.4.2 Proof of Proposition 18

Proof for **Proposition 18** on page 48: *If the system composed of plant components $\mathcal{G}_H$, $\mathcal{G}_L$, supervisors $\mathcal{S}_H$, $\mathcal{S}_L$, and interface G_I, is* serial level-wise controllable *with respect to the alphabet partition* $\Sigma := \Sigma_H \dot\cup \Sigma_L \dot\cup \Sigma_R \dot\cup \Sigma_A$, *then:*

$$(\forall s \in L(\mathbf{Plant}) \cap \mathbf{L}_\mathcal{S} \cap \mathcal{I}) \quad Elig_{L(\mathbf{Plant})}(s) \cap \Sigma_u \subseteq Elig_{\mathbf{L}_\mathcal{S} \cap \mathcal{I}}(s)$$

where $\mathbf{Plant} = \mathcal{G}_H ||_s \mathcal{G}_L$

Proof:

Assume system is *serial level-wise controllable* **(1)**

Let $s \in L(\mathbf{Plant}) \cap \mathbf{L}_\mathcal{S} \cap \mathcal{I}$ and $\sigma \in \text{Elig}_{L(\mathbf{Plant})}(s) \cap \Sigma_u$ **(2)**

Will now show this implies $\sigma \in \text{Elig}_{\mathbf{L}_\mathcal{S} \cap \mathcal{I}}(s)$

From **(2)**, we have $s, s\sigma \in L(\mathbf{Plant}) = \mathbf{H} \cap \mathbf{L}$

Also using **(2)**, we can now conclude $s \in \mathbf{L} \cap \mathbf{L}_\mathcal{S} \cap \mathcal{I}$ and $\sigma \in \text{Elig}_\mathbf{L}(s) \cap \Sigma_u$

Using **(1)**, we can conclude by **Point II** of the *serial level-wise controllable* definition that $\sigma \in \text{Elig}_{\mathbf{L}_\mathcal{S} \cap \mathcal{I}}(s)$, as required.

QED

3.4.3 Proof of Proposition 19

Proof for **Proposition 19** on page 48: *If the system composed of plant components $\mathcal{G}_H$, $\mathcal{G}_L$, supervisors $\mathcal{S}_H$, $\mathcal{S}_L$, and interface G_I, is* serial level-wise controllable *with respect to the alphabet partition $\Sigma := \Sigma_H \dot\cup \Sigma_L \dot\cup \Sigma_R \dot\cup \Sigma_A$, then:*

$$(\forall s \in L(\mathbf{Plant}) \cap \mathcal{I} \cap \mathbf{H}_\mathcal{S}) \quad \mathit{Elig}_{L(\mathbf{Plant}) \cap \mathcal{I}}(s) \cap \Sigma_u \subseteq \mathit{Elig}_{\mathbf{H}_\mathcal{S}}(s)$$

Proof:

Assume system is *serial level-wise controllable* **(1)**

Let $s \in L(\mathbf{Plant}) \cap \mathcal{I} \cap \mathbf{H}_\mathcal{S}$ and $\sigma \in \text{Elig}_{L(\mathbf{Plant}) \cap \mathcal{I}}(s) \cap \Sigma_u$ **(2)**

Will now show this implies $\sigma \in \text{Elig}_{\mathbf{H}_\mathcal{S}}(s)$

From **(2)**, we have $s, s\sigma \in L(\mathbf{Plant}) \cap \mathcal{I} = \mathbf{H} \cap \mathbf{L} \cap \mathcal{I}$

Also using **(2)**, we can now conclude $s \in \mathbf{H} \cap \mathcal{I} \cap \mathbf{H}_\mathcal{S}$ and $\sigma \in \text{Elig}_{\mathbf{H} \cap \mathcal{I}} \cap \Sigma_u$

Using **(1)**, we can conclude by **Point III** of the *serial level-wise controllable* definition that $\sigma \in \text{Elig}_{\mathbf{H}_\mathcal{S}}(s)$, as required.

QED

Chapter 4

Simple Manufacturing Example

We now present a simple manufacturing example to illustrate the method for the serial case. The example presented was inspired in part by the examples given in Wang [23], and in Brandin [4]. Table 4.1 defines abbreviations used for the event labels.

Abbrev.	Meaning	Abbrev.	Meaning
pt	part (item)	str	start
cmpl	complete	attch	attach
fin	finish	ent	enter
rlse	release	lv	leave
pol	polish	recog	recognize
arr	arrive		

Table 4.1: Abbreviations Used in Event Labels

In the following sections, we will describe our problem setting, and then present the original plant components. We will then assign them to a particular level of our hierarchy, augmenting if necessary the low level plant models so that they work better with an interface. We will then define the interface, supervisors, and finally we will present the complete system. We will conclude by demonstrating that the *flat system* is nonblocking and that the *flat supervisor* is controllable for the *flat plant.*

4.1 Description of Manufacturing Unit

As shown in Figure 4.1, the manufacturing unit is composed of three cells connected by a conveyor belt. In front of each cell, is a part acquisition unit that automatically stops a part and holds it until it is given a release command. Parts enter the system at the far left and exit at the far right. After the item exits the conveyor system, it goes to a packaging machine.

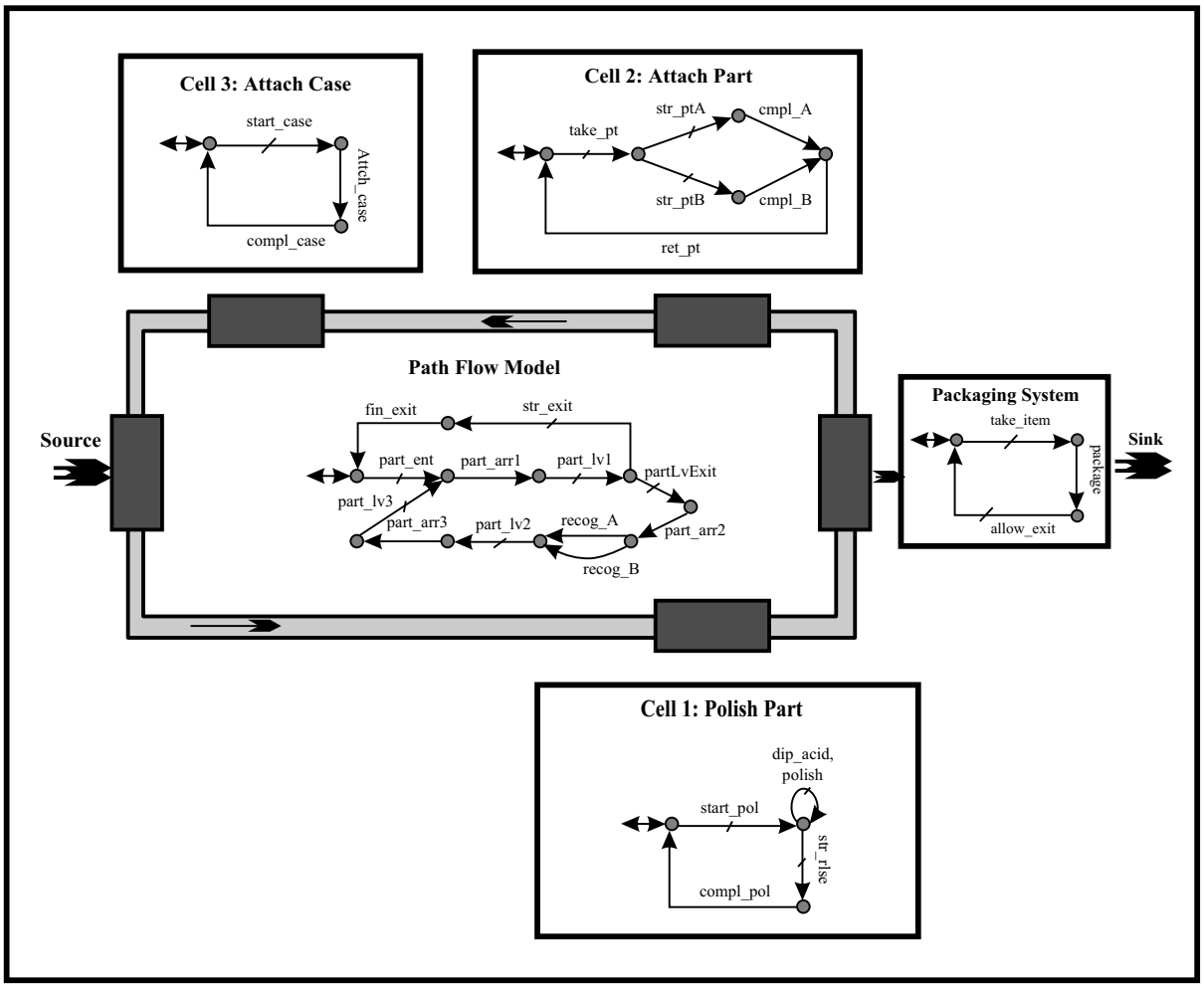


Figure 4.1: Block Diagram of Plant

The diagram shows a flat view of the plant (the supervisors will be added later). We see the plant models for cell one (polishes part), cell two (attaches part of type A or type B to the assembly of what's being built), cell three (attach case to assembly), and the path flow model that show how parts enter the system, travel around the belt, and finally leave the system. Of note in the path flow model are the events *recog_A,* and *recog_B.* The acquisition unit in front of cell two is capable of recognising if a part is of type A or type B. On the far right, we see the model for the packaging system.

4.1.1 Defining Infrastructure

The first step in the process is to decide which plant models belong to the *high level subsystem,* and which to the *low level subsystem.* The division we have chosen can be seen in Figure 4.2.

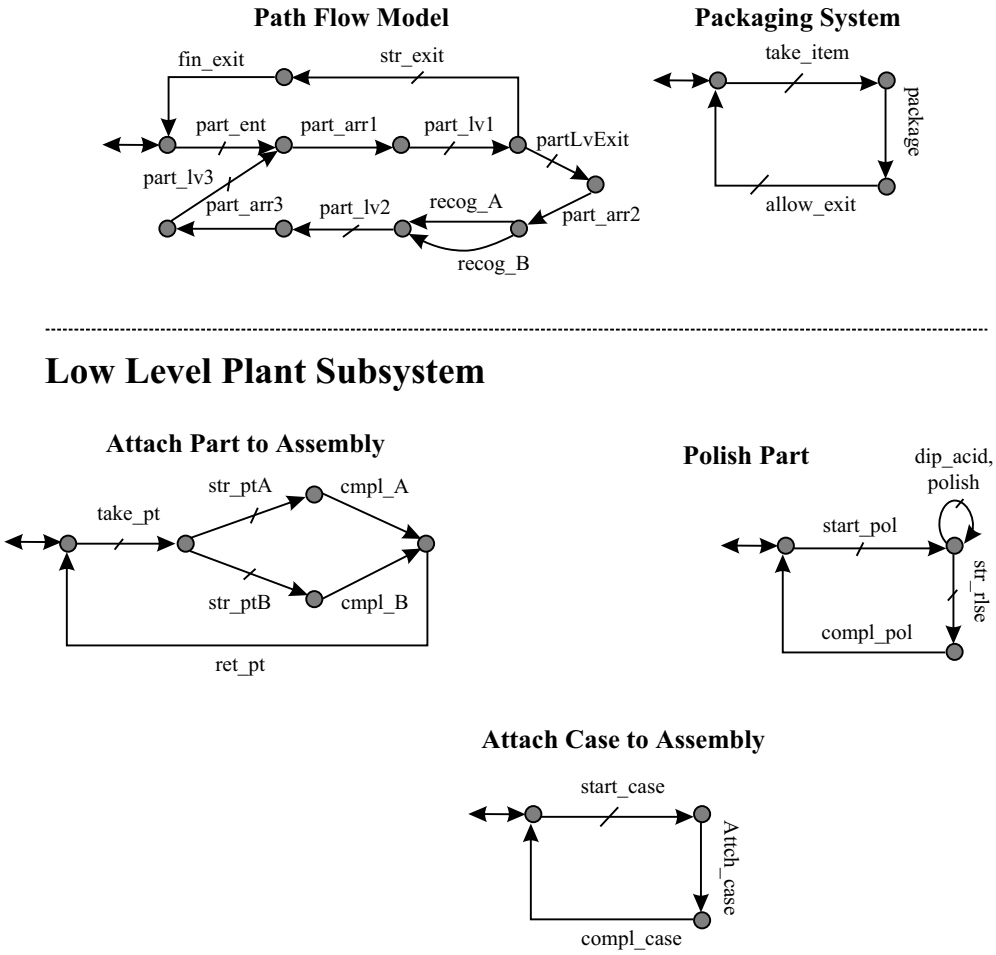


Figure 4.2: Original Plant

We now note that the model for cell two is not well suited to being accessed through an interface. It requires that the decision to attach part A or part B be made after event *take_pt* occurs. To make this functionality available to the upper level, we augment the model by adding the DES **Define New Events** shown in Figure 4.3. The new request events (*attch_ptA* and *attch_ptB*) will provide the high level with an easy selection method while the new finish events (*finA_attch* and *finB_attch*) will inform the high level of the completion of their respective tasks.

We are now ready to define our *interface*. Figure 4.4 shows the interface DES, G_I. From the diagram, we can see which events are *request events* and which events are *answer events*.

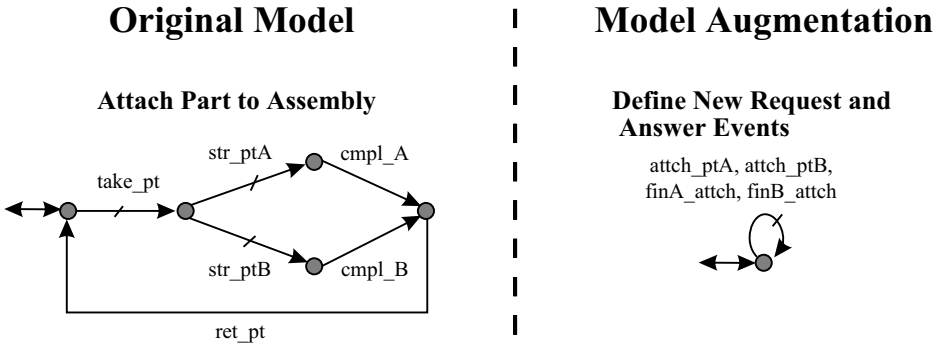


Figure 4.3: Augmenting Lower Plant

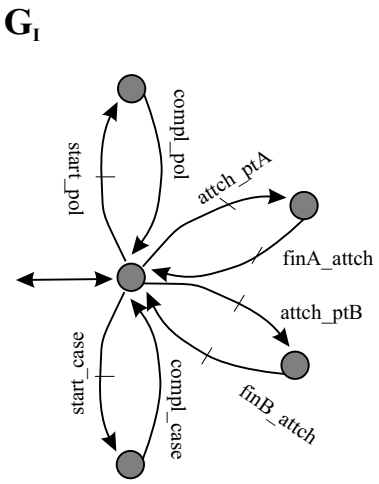


Figure 4.4: Interface Definition

We next define the alphabet partition $\Sigma := \Sigma_H \dot{\cup} \Sigma_L \dot{\cup} \Sigma_R \dot{\cup} \Sigma_A$ as follows:

$$
\begin{aligned}
\Sigma_R &= \{\textit{start\_pol}, \textit{attch\_ptA}, \textit{attch\_ptB}, \textit{start\_case}\} \\
\Sigma_A &= \{\textit{comp\_pol}, \textit{finA\_attch}, \textit{finB\_attch}, \textit{compl\_case}\} \\
\Sigma_H &= \{\textit{part\_ent}, \textit{part\_arr1}, \textit{part\_lv1}, \textit{partLvExit}, \\
&\quad \textit{str\_exit}, \textit{fin\_exit}, \textit{part\_arr2}, \textit{recog\_A}, \textit{recog\_B}, \\
&\quad \textit{part\_lv2}, \textit{part\_arr3}, \textit{part\_lv3}, \textit{take\_item}, \\
&\quad \textit{allow\_exit}, \textit{package}\} \\
\Sigma_L &= \{\textit{take\_pt}, \textit{str\_ptA}, \textit{str\_ptB}, \textit{compl\_A}, \textit{compl\_B}, \\
&\quad \textit{ret\_pt}, \textit{dip\_acid}, \textit{polish}, \textit{str\_rlse}, \textit{attch\_case}\}
\end{aligned}
$$

4.2 Designing Supervisors

Now that we have defined our *interface,* we are ready to design the low level supervisors that will provide the functionality for the *request events,* and give meaning to the *answer events.* The idea is for the *low level* to offer well-defined "services" to the *high level.*

We start with cell one. Here we want the sequence *dip_acid-polish* to be repeated twice, after a *start_pol* event occurs. The supervisor is shown in Figure 4.5, and is labelled **Polishing Sequence**. For cell two, we have to provide supervisors so that the cell reacts appropriately when events *attch_ptA* and *attch_ptB* occur. We also must guarantee that *answer events finA_attch* and *finB_attch* only occur when they have the appropriate meaning. The DES **Affix Part** in Figure 4.5 shows how this is done. Finally, we do nothing for cell three as it is so simple, its functionality being already present.

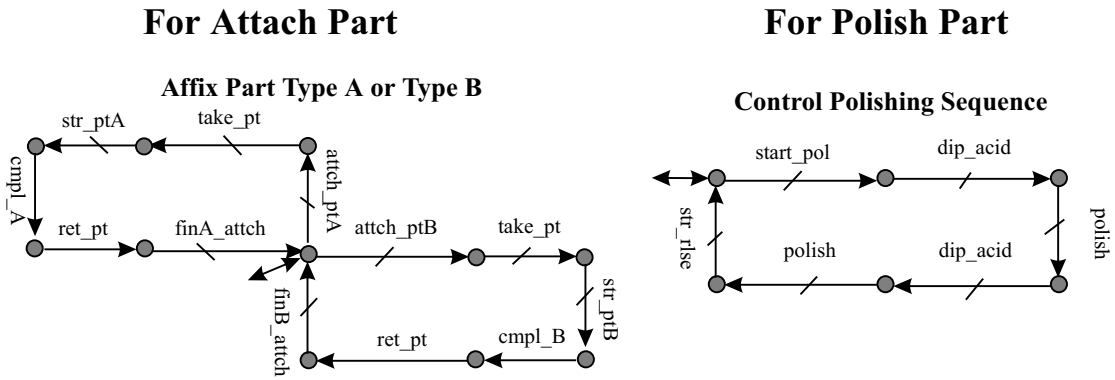


Figure 4.5: Supervisors to Support Interface

Now that the low level functionality is taken care of, we will design high level supervisors that use the *interface.* Figure 4.6 shows a supervisor (**Sequence Tasks**) that allows a part to visit each cell, executes the appropriate command for the cell and part type, and then allows the part to leave the conveyor system. The figure also shows a supervisor (**Exit Buffer**) that implements a two item buffer for the packaging system. Finally, we note that the above supervisors were designed by hand, but we could have also employed synthesis methods.

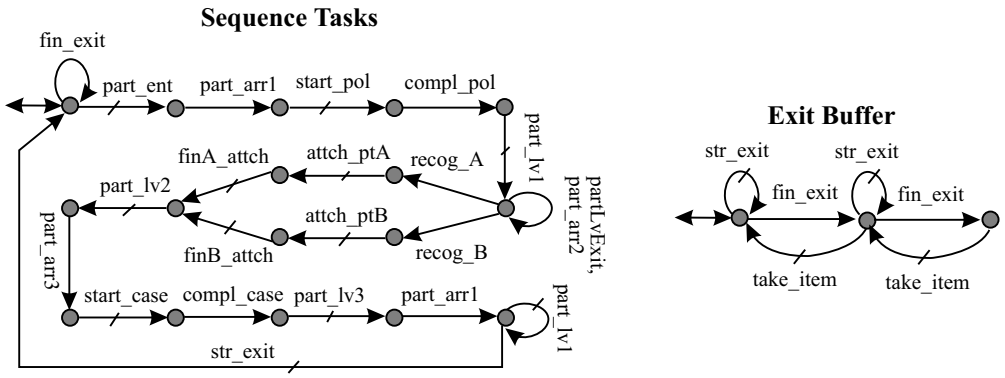


Figure 4.6: High Level Supervisors

4.3 The Final System

With the system components defined, it is time to put them together. Figure 4.7 shows our *high level subsystem*, plant, and supervisor, DES G_H, $\mathcal{G}_H$, and $\mathcal{S}_H$. We also have our *low level subsystem*, plant, and supervisor, DES G_L, $\mathcal{G}_L$, and $\mathcal{S}_L$. They are defined to be the synchronous product of the indicated automata.

We now want to determine whether the *flat system* is nonblocking. For this, we used our software tool to verify that the system is *serial interface consistent*, and *serial level-wise nonblocking*. We can thus conclude by **Theorem 1** that the *flat system* is nonblocking.

Next, we want to show that the *flat supervisor* is controllable for the *flat plant.* For this, we used our software tool to verify that the system is *serial level-wise controllable*. We can thus conclude by **Theorem 2** that the *flat supervisor* is controllable for the *flat plant.*

4.4 Concurrency of Subsystems

Before concluding this example, we comment on the inherent concurrency of the *high* and *low levels.* Unlike state expansion methods such as Wang [23] and Gohari [9] that expand a high level state into a group of low level states, the interface method is based on the *synchronous product*, limiting information flow, and a set of consistency rules. In general, there is no one-to-one association between a high level state and a set of low level states. This allows the *high level* to remain active while the *low level* is active, and thus operate concurrently. In the cited state expansion methods, the high level state would remain fixed while the low level becomes active.

This concurrency can be seen in the current example, by noting that once the event *fin_exit* has occurred, the string shown below is then possible.

part_ent part_arr1 start_pol dip_acid take_item polish

The string clearly shows how the *high level* event *take_item* can occur in the middle of a sequence of *low level* events, thus demonstrating that both levels are active.

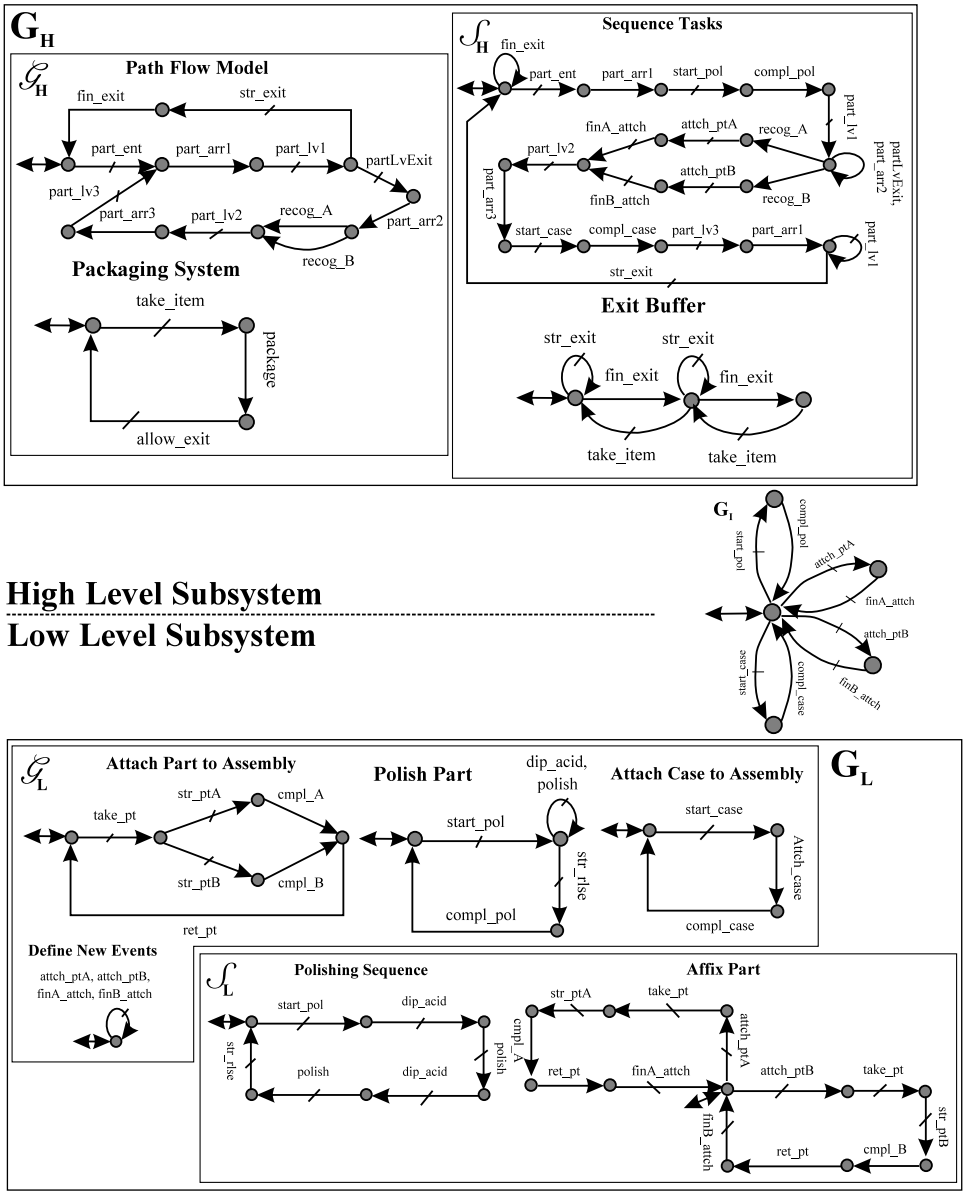


Figure 4.7: Complete System Definition

Chapter 5

Parallel Case: Nonblocking

In Chapter 2, we described our method for verifying nonblocking for the serial case where the number of *low levels* (n) is restricted to one. Such a system is also referred to as a *serial interface system.* We now extend our work to the more general setting where we have $n \geq 1$ *low levels* and we will refer to such a system as a *parallel interface system.* Figure 5.1 shows conceptually the structure and flow of information of a *parallel interface system.* In this new setting, we still have a single *high level,* but this time it is interacting with $n \geq 1$ autonomous low levels,[1] communicating with each *low level* in parallel through a separate *interface.* We will refer to the number of *low levels*, n, as the *degree* of the parallel system.

5.1 Definitions and Notation

We now introduce some terminology and notation that will be useful in simplifying proofs. For an n^{th} degree parallel system, we assume the *high level subsystem* is modelled by DES G_H (defined over event set $\dot{\bigcup}_{j\in\{1,\ldots,n\}}[\Sigma_{R_j} \,\dot{\cup}\, \Sigma_{A_j}] \;\dot{\cup}\, \Sigma_H$), the j^{th} *low level subsystem* is modelled by DES G_{L_j} (defined over event set $\Sigma_{L_j} \dot{\cup} \Sigma_{R_j} \dot{\cup} \Sigma_{A_j}$), the j^{th} *interface* by DES G_{I_j} (defined over event set $\Sigma_{R_j} \dot{\cup} \Sigma_{A_j}$), and that the overall system has the structure shown in Figure 5.2. Furthermore, we will refer to the j^{th} *low level* to mean $G_{L_j}||_s G_{I_j}$.

As in the serial case, in order to capture the restriction of the flow of information imposed by the *interface,* we partition the alphabet of the system into analogous pairwise disjoint alphabets, as below. For the remainder of this chapter. We define j to be $j \in \{1, \ldots, n\}$.

[1] By autonomous, we mean the event set of each *low level* is pairwise disjoint from the events sets of the other *low levels.*

Σ_H**:** These are events that exist only at the *high level.*

Σ_{R_j}**:** The set of *request events* for the j^{th} *interface.*

Σ_{A_j}**:** The set of *answer events* for the j^{th} *interface.*

Σ_{L_j}**:** The set of events that exist only at the j^{th} *low level.*

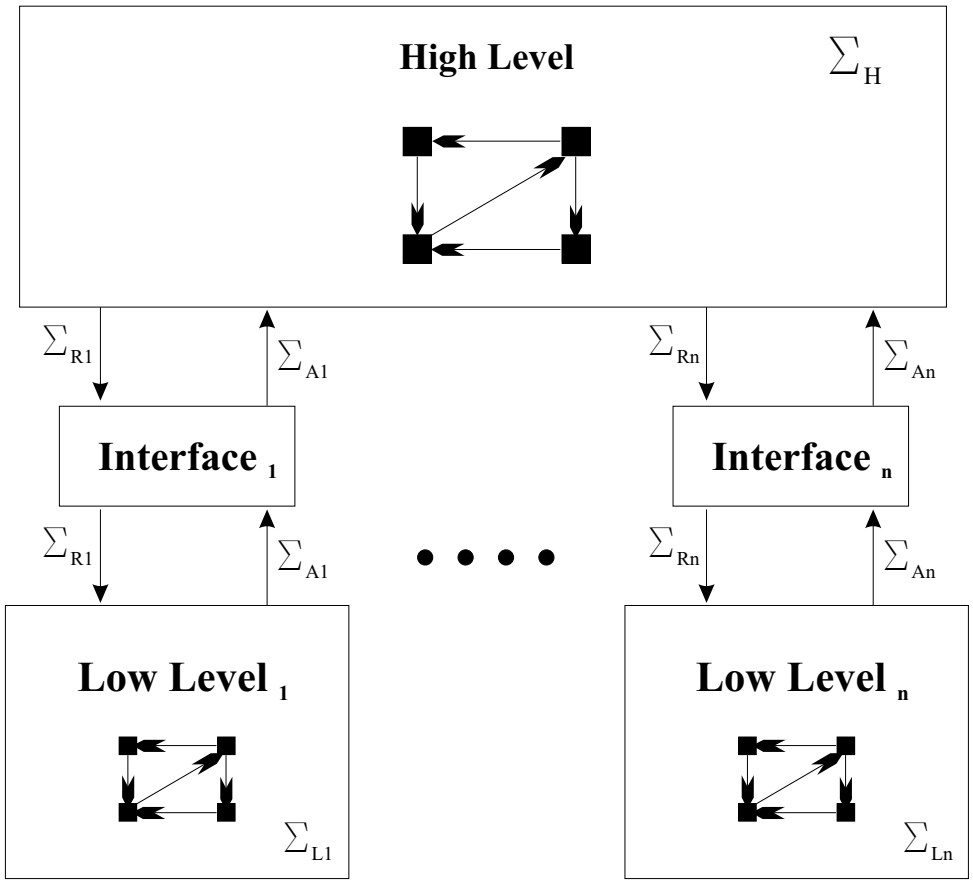


Figure 5.1: Parallel Interface Block Diagram.

We now assume that the alphabet partition is specified by $\Sigma := \dot{\cup}_{j\in\{1,\ldots,n\}}(\Sigma_{L_j} \,\dot{\cup}\, \Sigma_{R_j} \,\dot{\cup}\, \Sigma_{A_j}) \;\dot{\cup}\, \Sigma_H$ and that the *flat system* is taken to be:

$$G = G_H||_s G_{L_1}||_s \ldots ||_s G_{L_n}||_s G_{I_1}||_s \ldots ||_s G_{I_n}$$

We now introduce some useful event sets that we will be referring to often. They are defined as

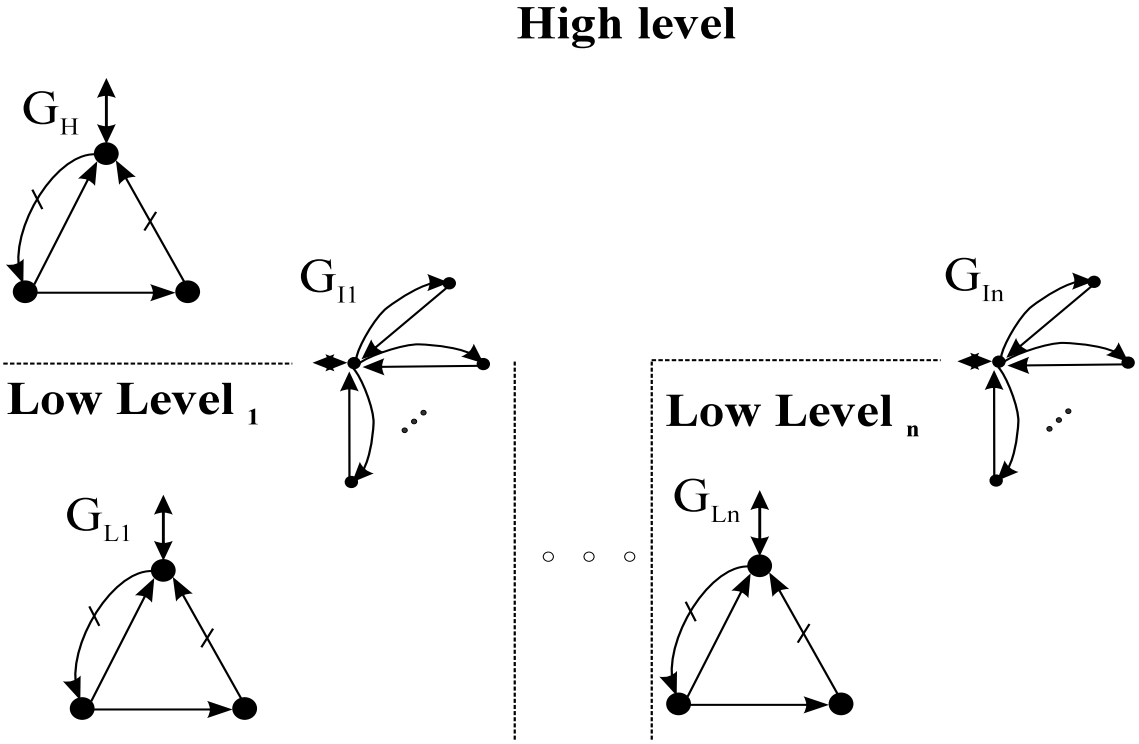


Figure 5.2: Two Tiered Structure of Parallel System

below:

$$\Sigma_{I_j} := \Sigma_{R_j} \cup \Sigma_{A_j} \quad \textit{Interface Events for the } j^{th} \textit{ Interface}$$

$$\Sigma_{IH} := \cup_{k\in\{1,\ldots,n\}}\Sigma_{I_k} \cup \Sigma_H \quad \textit{Interface and High Level Events}$$

$$\Sigma_{IL_j} := \Sigma_{L_j} \cup \Sigma_{I_j} \quad j^{th} \textit{ Set of Interface and Low Level Events}$$

$$\Sigma_{IL} := \cup_{k\in\{1,\ldots,n\}}\Sigma_{IL_k} \quad \textit{Set of all Interface and Low Level Events}$$

To be able to work with different languages defined over the above subsets, we define the following natural projections:

$$P_{IH} : \Sigma^* \rightarrow \Sigma^*_{IH}$$

$$P_{IL_j} : \Sigma^* \rightarrow \Sigma^*_{IL_j}$$

$$P_{I_j} : \Sigma^* \rightarrow \Sigma^*_{I_j}$$

As we want to express the languages of *flat system* in terms of their components, we need to define the

following languages:

$$\begin{aligned}
\mathcal{H} &:= P_{IH}^{-1}(L(G_H)), \quad & \mathcal{H}_m &:= P_{IH}^{-1}(L_m(G_H)) \quad &\subseteq \Sigma^* \\
\mathcal{L}_j &:= P_{IL_j}^{-1}(L(G_{L_j})), \quad & \mathcal{L}_{m_j} &:= P_{IL_j}^{-1}(L_m(G_{L_j})) \quad &\subseteq \Sigma^* \\
\mathcal{I}_j &:= P_{I_j}^{-1}(L(G_{I_j})), \quad & \mathcal{I}_{m_j} &:= P_{I_j}^{-1}(L_m(G_{I_j})) \quad &\subseteq \Sigma^*
\end{aligned}$$

We can now represent the closed behaviour of our *flat system* as follows:

$$\begin{aligned}
L(G) := \quad & L(G_H||_s G_{L_1}||_s \ldots ||_s G_{L_n}||_s G_{I_1}||_s \ldots ||_s G_{I_n}) \\
=& P_{IH}^{-1}(L(G_H)) \cap [\cap_{k\in\{1,\ldots,n\}}(P_{IL_k}^{-1}(L(G_{L_k})) \cap P_{I_k}^{-1}(L(G_{I_k})))] \\
=& \quad \mathcal{H} \cap [\cap_{k\in\{1,\ldots,n\}}(\mathcal{L}_k \cap \mathcal{I}_k)]
\end{aligned}$$

Similarly, the *flat marked language* of *system* is:

$$L_m(G) = \mathcal{H}_m \cap [\cap_{k\in\{1,\ldots,n\}}(\mathcal{L}_{m_k} \cap \mathcal{I}_{m_k})]$$

This allows us to present the proposition below that collects together several similar propositions. As it will be common in the proofs in this report to show that membership in languages such as $\mathcal{H}$ is dependent only on events in specific subsets (for $\mathcal{H}$, events in subset Σ_{IH}), this proposition will be very useful.

Proposition 20

(a) $(\forall s, s' \in \Sigma^*)\ s \in \mathcal{H}$ *and* $P_{IH}(s) = P_{IH}(s') \Rightarrow s' \in \mathcal{H}$

(b) $(\forall s, s' \in \Sigma^*)\ s \in \mathcal{H}_m$ *and* $P_{IH}(s) = P_{IH}(s') \Rightarrow s' \in \mathcal{H}_m$

(c) $(\forall k \in \{1,\ldots,n\})(\forall s, s' \in \Sigma^*)\ s \in \mathcal{L}_k$ *and* $P_{IL_k}(s) = P_{IL_k}(s') \Rightarrow s' \in \mathcal{L}_k$

(d) $(\forall k \in \{1,\ldots,n\})(\forall s, s' \in \Sigma^*)\ s \in \mathcal{L}_{m_k}$ *and* $P_{IL_k}(s) = P_{IL_k}(s') \Rightarrow s' \in \mathcal{L}_{m_k}$

(e) $(\forall k \in \{1,\ldots,n\})(\forall s, s' \in \Sigma^*)\ s \in \mathcal{I}_k$ *and* $P_{I_k}(s) = P_{I_k}(s') \Rightarrow s' \in \mathcal{I}_k$

(f) $(\forall k \in \{1,\ldots,n\})(\forall s, s' \in \Sigma^*)\ s \in \mathcal{I}_{m_k}$ *and* $P_{I_k}(s) = P_{I_k}(s') \Rightarrow s' \in \mathcal{I}_{m_k}$

Proof:

Points a-b:

Identical to the proof of **point a** of **Proposition 8**, after substitution.

Points c-f:

Let $k \in \{1, \ldots, n\}$, then identical to the proof of **point a** of **Proposition 8**, after substitution.

QED

5.2 Serial System Extraction: Subsystem Form

We now present a key definition for parallel interface systems. As the event set of each *low level* is mutually exclusive from the event sets of the other *low levels,* we can consider the *parallel interface system* as n *serial interface systems* by choosing one *low level* and ignoring the others. This will allow us to reuse our existing definitions and results for *serial interface systems.*

We are now ready to introduce the concept of *serial system extractions* for an n^{th} degree ($n \geq 1$) *parallel interface system.* For $j \in \{1, \ldots, n\}$, the j^{th} *serial system extraction* is essentially the original parallel system with the $1^{\text{st}}, \ldots, (j-1)^{\text{th}}, (j+1)^{\text{th}}, \ldots, n^{\text{th}}$ *low levels* removed. Figure 5.3 shows this conceptually.

j^{th} Serial System Extraction: Subsystem Form For the n^{th} degree ($n \geq 1$) *parallel interface system* composed of DES G_H,
$G_{L_1}, \ldots, G_{L_n}, G_{I_1}, \ldots, G_{I_n}$, with alphabet partition $\Sigma := \dot{\cup}_{k \in \{1,\ldots,n\}}(\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$,
the j^{th} *serial system extraction,* denoted by *system*(j), is composed of the following elements:

$$
\begin{aligned}
G_H(j) &:= G_H||_s G_{I_1}||_s \ldots ||_s G_{I_{(j-1)}}||_s G_{I_{(j+1)}}||_s \ldots ||_s G_{I_n} \\
G_L(j) &:= G_{L_j} \\
G_I(j) &:= G_{I_j}
\end{aligned}
$$

$$
\begin{aligned}
\Sigma_H(j) &:= \dot{\cup}_{k \in \{1, \ldots, (j-1), (j+1), \ldots, n\}} \Sigma_{I_k} \dot{\cup} \Sigma_H \\
\Sigma_L(j) &:= \Sigma_{L_j} \\
\Sigma_R(j) &:= \Sigma_{R_j} \\
\Sigma_A(j) &:= \Sigma_{A_j}
\end{aligned}
$$

$$\begin{aligned}\Sigma(j) &:= \Sigma_H(j) \dot{\cup} \Sigma_L(j) \dot{\cup} \Sigma_R(j) \dot{\cup} \Sigma_A(j) \\ &= \Sigma - \dot{\cup}_{k \in \{1,...,(j-1),(j+1),...,n\}} \Sigma_{L_k}\end{aligned}$$

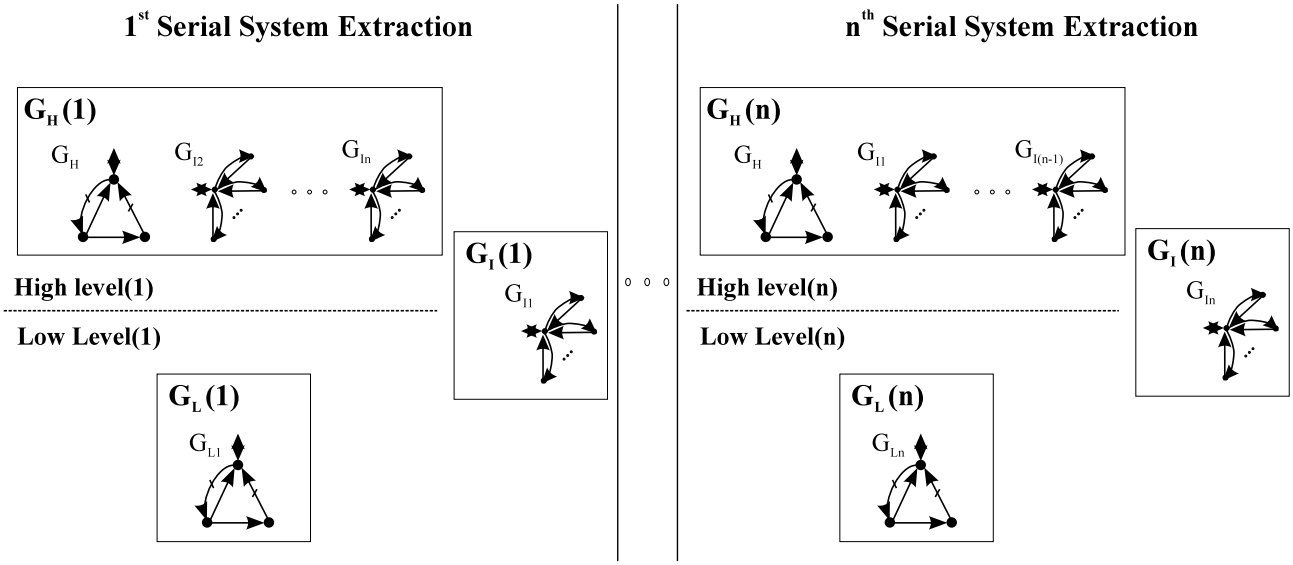


Figure 5.3: The Serial System Extraction

In the above definition, we defined *serial system extractions* in terms of a parallel subsystem based form system. As we did for the serial case, we will define in Chapter 6 a general form for parallel systems. We will also provide a corresponding general form definition for *serial system extractions.* We will simply refer to the j^{th} *serial system extraction,* as the type of the parallel system will make clear which definition is intended.

5.3 Interface Properties

We now present some important definitions that are analogous to equivalent definitions for the serial case. We then present some related propositions.

5.3.1 Parallel Interface Definitions

In this section we present a set of properties that are equivalent to their serial interface counterparts. They all involve interpreting the parallel system as n serial systems by using the *serial system extraction* definition.

Interface Consistent: The n^{th} degree ($n \geq 1$) *parallel interface system* composed of DES G_H, $G_{L_1}, \ldots, G_{L_n}, G_{I_1}, \ldots, G_{I_n}$, is *interface consistent* with respect to alphabet partition $\Sigma := \dot{\cup}_{k\in\{1,\ldots,n\}} (\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, if:

$(\forall j \in \{1, \ldots, n\})$ The j^{th} *serial system extraction* of the system is *serial interface consistent.*

Interface Strict Marking: The n^{th} degree ($n \geq 1$) *parallel interface system* composed of DES G_H, $G_{L_1}, \ldots, G_{L_n}, G_{I_1}, \ldots, G_{I_n},$, is *interface strict marking* with respect to the alphabet partition $\Sigma := \dot{\cup}_{k\in\{1,\ldots,n\}} (\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, if:

$(\forall j \in \{1, \ldots, n\})$ The j^{th} *serial system extraction* of the system is *serial interface strict marking.*

Level-wise Nonblocking: The n^{th} degree ($n \geq 1$) *parallel interface system* composed of DES G_H, $G_{L_1}, \ldots, G_{L_n}, G_{I_1}, \ldots, G_{I_n}$, is *level-wise nonblocking* with respect to the alphabet partition $\Sigma := \dot{\cup}_{k\in\{1,\ldots,n\}} (\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, if:

$(\forall j \in \{1, \ldots, n\})$ The j^{th} *serial system extraction* of the system is *serial level-wise nonblocking.*

5.3.2 Related Propositions

Now that we have the above definitions, we can present several related propositions that establish properties about the parallel system that will be useful in later proofs.

Our first proposition uses the *interface consistent* definition to establish the event set that the DES which make up an n^{th} degree ($n \geq 1$) *parallel interface system* are defined over. This is useful for defining the languages of a DES created by the synchronous product of one or more of these DES.

Proposition 21 *If the n^{th} degree ($n \geq 1$)* parallel interface system *composed of DES G_H, $G_{L_1}, \ldots, G_{L_n}, G_{I_1}, \ldots, G_{I_n}$, is* interface consistent *with respect to the alphabet partition $\Sigma := \dot{\cup}_{k\in\{1,\ldots,n\}} (\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$ then DES G_H is defined over event set Σ_{IH}, DES G_{I_j} is defined over event set Σ_{I_j}, and DES G_{L_j} is defined over event set Σ_{IL_j}, where $j \in \{1, \ldots, n\}$.*

Proof: See page 71.

We are now ready to state the proposition below which establishes useful properties for often used languages.

Proposition 22 *If the* n^{th} *degree* ($n \geq 1$) parallel interface system *composed of DES* G_H, $G_{L_1}, \ldots, G_{L_n}, G_{I_1}, \ldots, G_{I_n}$, *is* interface consistent *with respect to the alphabet partition* $\Sigma := \dot{\cup}_{k\in\{1,\ldots,n\}}(\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$ *then, for all* $j \in \{1, \ldots, n\}$, *the following is true:*

(i) Languages $\mathcal{H}$, $\mathcal{L}_j$, *and* $\mathcal{I}_j$ *are closed.*

(ii) $\mathcal{H}_m \subseteq \mathcal{H}$, $\mathcal{L}_{m_j} \subseteq \mathcal{L}_j$, *and* $\mathcal{I}_{m_j} \subseteq \mathcal{I}_j$

Proof: See page 73.

We now present a proposition that will aid in the use of *serial system extractions* in proofs. The proposition interprets terminology for the j^{th} *serial system extraction* (a *serial interface system*) in terms of the original parallel system.

Before we can present the proposition, we need to first define (for use in the proposition) a new natural projection, P_j, to map strings from Σ^* (the event set of the given parallel system) to strings from $\Sigma(j)^*$ (the event set of the j^{th} extracted system of the given parallel system). It is defined as follows:

$$P_j : \Sigma^* \rightarrow \Sigma(j)^*$$

Proposition 23 *If the* n^{th} *degree* ($n \geq 1$) parallel interface system *composed of DES* $G_H, G_{L_1}, \ldots, G_{L_n}$, $G_{I_1}, \ldots, G_{I_n}$, *is* interface consistent *with respect to the alphabet partition* $\Sigma := \dot{\cup}_{k\in\{1,\ldots,n\}}(\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, *then for the* j^{th} serial system extraction, system(j), *the following is true:*

(i) The flat system *is:* $G(j) = G_H||_s G_{L_j}||_s G_{I_1}||_s \ldots ||_s G_{I_n}$

(ii) The following event sets are: $\Sigma_I(j) = \Sigma_{I_j}$, $\Sigma_{IH}(j) = \Sigma_{IH}$, *and* $\Sigma_{IL}(j) = \Sigma_{IL_j}$

(iii) The following inverse natural projections are: $P_{IH}(j)^{-1} = P_j \cdot P_{IH}^{-1}$, $P_{IL}(j)^{-1} = P_j \cdot P_{IL_j}^{-1}$, *and* $P_I(j)^{-1} = P_j \cdot P_{I_j}^{-1}$

(iv) The event set of $G_H(j)$ *is* $\Sigma_{IH}(j)$, *the event set of* $G_L(j)$ *is* $\Sigma_{IL}(j)$, *and the event set of* $G_I(j)$ *is* $\Sigma_I(j)$.

(v) The following languages are:

$$\begin{aligned}
\mathcal{H}(j) &= P_j(\mathcal{H}) \cap [\cap_{k\in\{1,\ldots,(j-1),(j+1),\ldots,n\}} P_j(\mathcal{I}_k)] \\
\mathcal{H}_m(j) &= P_j(\mathcal{H}) \cap [\cap_{k\in\{1,\ldots,(j-1),(j+1),\ldots,n\}} P_j(\mathcal{I}_{m_k})] \\
\mathcal{L}(j) &= P_j(\mathcal{L}_j) \\
\mathcal{L}_m(j) &= P_j(\mathcal{L}_{m_j}) \\
\mathcal{I}(j) &= P_j(\mathcal{I}_j) \\
\mathcal{I}_m(j) &= P_j(\mathcal{I}_{m_j})
\end{aligned}$$

(vi) Languages $\mathcal{H}(j)$, $\mathcal{L}(j)$, and $\mathcal{I}(j)$ are closed.

(vii) $\mathcal{H}_m(j) \subseteq \mathcal{H}(j)$, $\mathcal{L}_m(j) \subseteq \mathcal{L}(j)$, and $\mathcal{I}_m(j) \subseteq \mathcal{I}(j)$

Proof: See page 74.

We close this section by noting that after examining the definition of the j^{th} *serial system extraction* and the above proposition, we see that for $n = 1$, a *parallel interface system* reduces to a single *serial interface system.* We thus see that a *serial interface system* is a special case of *parallel interface systems.* We will now talk of *parallel interface systems* and their definitions as the general case for bi-level interface systems.

5.4 Parallel Nonblocking Theorem and Propositions

We will now present **Propositions 24-26**, followed by our main result for this chapter, **Theorem 3**. The following propositions are analogous to their serial case counterparts.

Parallel Low Level Nonblocking Proposition

Our first proposition is analogous to **Proposition 11** for the serial case. It asserts that a string s accepted by the system, can always be extended to a string accepted by the system, and marked by the all *low levels.* In other words, the *low levels* are not dependent on high level events to reach a marked state.

Proposition 24 *If the* n^{th} *degree* ($n \geq 1$) parallel interface system *composed of DES* $G_H, G_{L_1}, \ldots, G_{L_n}$, $G_{I_1}, \ldots, G_{I_n}$, *is* level-wise nonblocking *and* interface consistent *with respect to the alphabet partition* $\Sigma := \dot{\cup}_{k\in\{1,\ldots,n\}}(\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, *then*

$(\forall s \in \mathcal{H} \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_j \cap \mathcal{I}_j)])$
$\quad (\exists l \in \Sigma_{IL}^*)$ *s.t.* $(sl \in \mathcal{H} \cap [\cap_{j\in\{1,2,\ldots,n\}}(\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})])$

Proof: See page 78.

5.4.1 Event Agreement Propositions

We group the last two propositions together as **Proposition 26** builds upon **Proposition 25**. The first proposition asserts that any string accepted by the system can always be extended by a string marked by the *high level.* The reader should note that this string is not necessarily accepted by any *low levels.*

Proposition 25 *If the* n^{th} *degree* ($n \geq 1$) parallel interface system *composed of DES* $G_H, G_{L_1}, \ldots, G_{L_n}$, $G_{I_1}, \ldots, G_{I_n}$, *is* level-wise nonblocking *and* interface consistent *with respect to the alphabet partition* $\Sigma := \dot{\cup}_{k\in\{1,\ldots,n\}}(\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, *then*

$(\forall s \in \mathcal{H} \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_j \cap \mathcal{I}_j)])$
$\quad (\exists h \in \Sigma_{IH}^*)$ $(sh \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}\mathcal{I}_{m_j}])$

Proof: See page 81.

Our last proposition is analogous to **Proposition 15** for the serial case. It asserts that if string h extends string s such that sh is acceptable to the *high level*, then a string u can be constructed such that u has a high level image equal to h, and that su is marked by the system. In other words, we can use string h as a basis to construct string u by adding low level events so that each *low level subsystem* will accept the request and *answer event* contained in h. As these events are common to both levels, they must agree on their occurrence.

Proposition 26 *If the* n^{th} *degree* ($n \geq 1$) parallel interface system *composed of DES* $G_H, G_{L_1}, \ldots, G_{L_n}$, $G_{I_1}, \ldots, G_{I_n}$, *is* level-wise nonblocking *and* interface consistent *with respect to the alphabet partition* $\Sigma := \dot{\cup}_{k\in\{1,\ldots,n\}}(\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, *then*

$(\forall s \in \mathcal{H} \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})])(\forall h \in \Sigma_{IH}^*)$
$\quad (sh \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}\mathcal{I}_{m_j}]) \Rightarrow (\exists u \in \Sigma^*)$ *s.t.* $(su \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})]$

Proof: See page 83.

5.4.2 Parallel Nonblocking Theorem

We are now ready to present our nonblocking theorem for *parallel interface systems.* It states that, to verify if a parallel system is nonblocking, it is sufficient to check that each of its *serial system extractions* is *serial level-wise nonblocking* and *serial interface consistent.* As the *level-wise nonblocking* and *interface consistent* definitions can be evaluated by examining only one level (the *high level* or one of the *low levels*) of our system at a time, we now have a means to verify nonblocking of our parallel system using local checks.

Theorem 3 *If the* n^{th} *degree* ($n \geq 1$) parallel interface system *composed of DES* G_H, $G_{L_1}, \ldots, G_{L_n}, G_{I_1}, \ldots, G_{I_n}$, *is* level-wise nonblocking *and* interface consistent *with respect to the alphabet partition* $\Sigma := \dot{\cup}_{k\in\{1,\ldots,n\}}(\Sigma_{L_k} \,\dot{\cup}\, \Sigma_{R_k} \,\dot{\cup}\, \Sigma_{A_k}) \,\dot{\cup}\, \Sigma_H$, *then*

$L(G) = \overline{L_m(G)}$, *where* $\quad G = G_H||_s G_{L_1}||_s \ldots ||_s G_{L_n}||_s G_{I_1}||_s \ldots ||_s G_{I_n}$

Proof:

Assume system is *level-wise nonblocking* and *interface consistent.* **(1)**

As $\overline{L_m(G)} \subseteq L(G)$ is automatic, it suffices to show $L(G) \subseteq \overline{L_m(G)}$

Let $s \in L(G) = \mathcal{H} \cap [\cap_{w\in\{1,\ldots,n\}}(\mathcal{L}_w \cap \mathcal{I}_w)]$ **(2)**

We will now show this implies $s \in \overline{L_m(G)}$

It is sufficient to show: $(\exists u \in \Sigma^*)\ su \in L_m(G) = \mathcal{H}_m \cap [\cap_{w\in\{1,\ldots,n\}}(\mathcal{L}_{m_w} \cap \mathcal{I}_{m_w})]$

Our first step is to show that we can construct a low level string, accepted by the *high level*, and that will bring all n *low levels* to a marked state. We can achieve this immediately by applying **Proposition 24** and conclude:

$$(\exists l \in \Sigma_{IL}^*)\ \text{ s.t. }\ (sl \in \mathcal{H} \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})]) \qquad \textbf{(3)}$$

Our next step will be to show that we can construct a string $u' \in \Sigma^*$ such that $slu' \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})]$.

To achieve this, we will apply **Proposition 26**. Before we can apply the proposition, we must first construct a suitable $h \in \Sigma_{IH}^*$.

We first note that **(3)** implies that $sl \in \mathcal{H} \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_j \cap \mathcal{I}_j)])$. We can now apply **Proposition**

25, taking sl to be string s in that proposition, and conclude:

$$(\exists h \in \Sigma_{IH}^*)\ \ (slh \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}\mathcal{I}_{m_j}]) \qquad \textbf{(4)}$$

Combining with **(3)**, we can now apply **Proposition 26**, taking sl to be string s in that proposition, and conclude:

$$(\exists u' \in \Sigma^*) \text{ s.t. } (slu' \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})]$$

We take string $u = lu'$ and we have $su \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})] = L_m(G)$, as required.

QED

5.5 Proofs of Selected Propositions

In order to make this work more readable, the proofs of some propositions in this chapter were not given as the propositions were introduced. They are now presented in the following sections.

5.5.1 Proof of Proposition 21

Proof for **Proposition 21** on page 66: *If* n^{th} *degree* ($n \geq 1$) parallel interface system *composed of DES* G_H, $G_{L_1}, \ldots, G_{L_n}, G_{I_1}, \ldots, G_{I_n}$, *is* interface consistent *with respect to the alphabet partition* $\Sigma := \dot\cup_{k\in\{1,\ldots,n\}}(\Sigma_{L_k}\dot\cup\Sigma_{R_k}\dot\cup\Sigma_{A_k}) \dot\cup \Sigma_H$ *then DES* G_H *is defined over event set* Σ_{IH}, *DES* G_{I_j} *is defined over event set* Σ_{I_j}, *and DES* G_{L_j} *is defined over event set* Σ_{IL_j}, *where* $j \in \{1, \ldots, n\}$.

Proof:

Assume that the n^{th} degree ($n \geq 1$) *parallel interface system* is *interface consistent* with respect to the alphabet partition. **(1)**

We will now show this implies that DES G_H is defined over event set Σ_{IH}, DES G_{I_j} is defined over event set Σ_{I_j}, and that DES G_{L_j} is defined over event set Σ_{IL_j}, where $j \in \{1, \ldots, n\}$.

We first note that **(1)** implies that $(\forall j \in \{1, \ldots, n\})$ the j^{th} *serial system extraction*, labelled *system*(j), of the system is *serial interface consistent.*

This allows us to conclude that: $(\forall j \in \{1, \ldots, n\})$ $G_H(j)$ is defined over $\Sigma_{IH}(j)$, $G_L(j)$ is defined over $\Sigma_{IL}(j)$, and that $G_I(j)$ is defined over $\Sigma_I(j)$

From **(1)**, we can now apply **Proposition 23**, **point ii** and conclude:[2]

$\Sigma_I(j) = \Sigma_{I_j}$, $\Sigma_{IH}(j) = \Sigma_{IH}$, and $\Sigma_{IL}(j) = \Sigma_{IL_j}$

We can now conclude that: $(\forall j \in \{1, \ldots, n\})$ $G_H(j)$ is defined over Σ_{IH}, $G_L(j)$ is defined over Σ_{IL_j}, and that $G_I(j)$ is defined over Σ_{I_j}. **(2)**

This implies: $(\forall j \in \{1, \ldots, n\})$ DES $G_{L_j} = G_L(j)$ is defined over Σ_{IL_j} and that $G_{I_j} = G_I(j)$ is defined over Σ_{I_j}. **(3)**

All that remains is to show that G_H is defined over alphabet Σ_{IH}. To do this, we first need to prove the following claim.

Claim: $\Sigma_{G_H} \subseteq \Sigma_{IH}$ and $(\forall j \in \{1, \ldots, n\})$ $\Sigma_{G_H} \supseteq (\Sigma_H \cup \Sigma_{I_j})$

Let $j \in \{1, \ldots, n\}$. We will now show this implies $\Sigma_{G_H} \subseteq \Sigma_{IH}$ and $\Sigma_{G_H} \supseteq (\Sigma_H \cup \Sigma_{I_j})$.

We start by noting $G_H(j) := G_H||_s G_{I_1}||_s \ldots ||_s G_{I_{(j-1)}}||_s G_{I_{(j+1)}}||_s \ldots ||_s G_{I_n}$. By the definition of the $||_s$ operator, we know $\Sigma_{G_H(j)} = \Sigma_{G_H} \cup [\cup_{k \in \{1, \ldots, (j-1), (j+1), \ldots, n\}} \Sigma_{G_{I_k}}]$. This implies that $\Sigma_{G_H} \subseteq \Sigma_{G_H(j)}$. As $\Sigma_{G_H(j)} = \Sigma_{IH}$ (from **(2)**), we immediately have $\Sigma_{G_H} \subseteq \Sigma_{IH}$.

From **(3)**, we have $\Sigma_{G_H(j)} = \Sigma_{G_H} \cup [\cup_{k \in \{1, \ldots, (j-1), (j+1), \ldots, n\}} \Sigma_{I_k}]$.

We now note that $\Sigma_{I_j} \subseteq \Sigma_{IH}$ but $\Sigma_{I_j} \cap [\cup_{k \in \{1, \ldots, (j-1), (j+1), \ldots, n\}} \Sigma_{I_k}] = \emptyset$ because of our event partition.

This implies: $\Sigma_{I_j} \subseteq \Sigma_{G_H}$

We next note that $\Sigma_H \subseteq \Sigma_{IH}$ but $\Sigma_H \cap [\cup_{k \in \{1, \ldots, (j-1), (j+1), \ldots, n\}} \Sigma_{I_k}] = \emptyset$ because of our event partition.

This implies: $\Sigma_H \subseteq \Sigma_{G_H}$

We thus have $\Sigma_{G_H} \supseteq (\Sigma_H \cup \Sigma_{I_j})$ as required.

Claim proven.

From the claim, we have $\Sigma_{G_H} \subseteq \Sigma_{IH}$. To show that $\Sigma_{G_H} = \Sigma_{IH}$ we now only have to show $\Sigma_{G_H} \supseteq \Sigma_{IH}$.

From the claim, we also have $(\forall j \in \{1, \ldots, n\})$ $\Sigma_{G_H} \supseteq (\Sigma_H \cup \Sigma_{I_j})$.

This implies $\Sigma_{G_H} \supseteq \Sigma_H \cup [\cup_{k \in \{1, \ldots, n\}} \Sigma_{I_k}] = \Sigma_{IH}$. We thus have $\Sigma_{G_H} = \Sigma_{IH}$.

We can now conclude that DES G_H is defined over Σ_{IH}, as required.

QED

[2] **Proposition 23**, **point iv** requires the proposition we are currently proving, but **point ii** of **Proposition 23** is independent of **point iv** and does not.

5.5.2 Proof of Proposition 22

Proof for **Proposition 22** on page 67: If the n^{th} degree ($n \geq 1$) *parallel interface system* composed of DES G_H, $G_{L_1}, \ldots, G_{L_n}, G_{I_1}, \ldots, G_{I_n}$, is *interface consistent* with respect to the alphabet partition $\Sigma := \dot{\cup}_{k\in\{1,\ldots,n\}}(\Sigma_{L_k} \,\dot{\cup}\, \Sigma_{R_k} \,\dot{\cup}\, \Sigma_{A_k}) \,\dot{\cup}\, \Sigma_H$ then, for all $j \in \{1, \ldots, n\}$, the following is true:

(i) Languages $\mathcal{H}$, $\mathcal{L}_j$, and $\mathcal{I}_j$ are closed.

(ii) $\mathcal{H}_m \subseteq \mathcal{H}$, $\mathcal{L}_{m_j} \subseteq \mathcal{L}_j$, and $\mathcal{I}_{m_j} \subseteq \mathcal{I}_j$

Proof:

Assume system is *interface consistent* and let $j \in \{1, \ldots, n\}$. **(1)**

Will now show this implies that **Points i** and **ii** are satisfied.

We first not that by **(1)**, the system is *interface consistent.* This allows us to apply **Proposition 21** and conclude:

$$L(G_H), L_m(G_H) \subseteq \Sigma^*_{IH},\ L(G_{L_j}), L_m(G_{L_j}) \subseteq \Sigma^*_{IL_j}, \text{ and } L(G_{I_j}), L_m(G_{I_j}) \subseteq \Sigma^*_{I_j}.$$

This tells us that languages $\mathcal{H} = P^{-1}_{IH}(L(G_H))$, $\mathcal{L}_j = P^{-1}_{IL}(L(G_{L_j}))$, $\mathcal{I}_j = P^{-1}_{I}(L(G_{I_j}))$, $\mathcal{H}_m = P^{-1}_{IH}(L_m(G_H))$, $\mathcal{L}_{m_j} = P^{-1}_{IL}(L_m(G_{L_j}))$, and $\mathcal{I}_{m_j} = P^{-1}_{I}(L_m(G_{I_j}))$ are defined.

Point i: Show that Languages $\mathcal{H}$, $\mathcal{L}_j$, and $\mathcal{I}_j$ are closed.

We now note that languages $L(G_H)$, $L(G_{L_j})$, and $L(G_{I_j})$ are closed by the definition of the closed behaviour of a DES.

We can now apply **Proposition 1** repeatedly and conclude that $\mathcal{H}$, $\mathcal{L}_j$, and $\mathcal{I}_j$ are closed, as required.

Point ii: Show that $\mathcal{H}_m \subseteq \mathcal{H}$, $\mathcal{L}_{m_j} \subseteq \mathcal{L}_j$, and $\mathcal{I}_{m_j} \subseteq \mathcal{I}_j$.

From the definition of the closed behaviour and the marked language of a DES, we can conclude that:

$$L_m(G_H) \subseteq L(G_H),\ L_m(G_{L_j}) \subseteq L(G_{L_j}), \text{ and } L_m(G_{I_j}) \subseteq L(G_{I_j}).$$

Applying **Proposition 3** repeatedly, we can conclude:

$$\begin{aligned} P^{-1}_{IH}(L_m(G_H)) = \mathcal{H}_m &\quad\subseteq\quad \mathcal{H} = P^{-1}_{IH}(L(G_H)) \\ P^{-1}_{IL}(L_m(G_{L_j})) = \mathcal{L}_{m_j} &\quad\subseteq\quad \mathcal{L}_j = P^{-1}_{IL}(L(G_{L_j})) \\ P^{-1}_{I}(L_m(G_{I_j})) = \mathcal{I}_{m_j} &\quad\subseteq\quad \mathcal{I}_j = P^{-1}_{I}(L(G_{I_j})) \end{aligned}$$

QED

5.5.3 Proof of Proposition 23

Proof for **Proposition 23** on page 67: *If the n^{th} degree ($n \geq 1$)* parallel interface system *composed of DES G_H, $G_{L_1}, \ldots, G_{L_n}$, $G_{I_1}, \ldots, G_{I_n}$, is* interface consistent *with respect to the alphabet partition $\Sigma := \dot{\cup}_{k\in\{1,\ldots,n\}}(\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, then for the j^{th}* serial system extraction, system(j), *the following is true:*

(i) The flat system *is:* $G(j) = G_H||_s G_{L_j}||_s G_{I_1}||_s \ldots ||_s G_{I_n}$

(ii) The following event sets are: $\Sigma_I(j) = \Sigma_{I_j}$, $\Sigma_{IH}(j) = \Sigma_{IH}$, *and* $\Sigma_{IL}(j) = \Sigma_{IL_j}$

(iii) The following inverse natural projections are: $P_{IH}(j)^{-1} = P_j \cdot P_{IH}^{-1}$, $P_{IL}(j)^{-1} = P_j \cdot P_{IL_j}^{-1}$, *and* $P_I(j)^{-1} = P_j \cdot P_{I_j}^{-1}$

(iv) The event set of $G_H(j)$ is $\Sigma_{IH}(j)$, the event set of $G_L(j)$ is $\Sigma_{IL}(j)$, and the event set of $G_I(j)$ is $\Sigma_I(j)$.

(v) The following languages are:

$$\begin{aligned}
\mathcal{H}(j) &= P_j(\mathcal{H}) \cap [\cap_{k\in\{1,\ldots,(j-1),(j+1),\ldots,n\}} P_j(\mathcal{I}_k)] \\
\mathcal{H}_m(j) &= P_j(\mathcal{H}) \cap [\cap_{k\in\{1,\ldots,(j-1),(j+1),\ldots,n\}} P_j(\mathcal{I}_{m_k})] \\
\mathcal{L}(j) &= P_j(\mathcal{L}_j) \\
\mathcal{L}_m(j) &= P_j(\mathcal{L}_{m_j}) \\
\mathcal{I}(j) &= P_j(\mathcal{I}_j) \\
\mathcal{I}_m(j) &= P_j(\mathcal{I}_{m_j})
\end{aligned}$$

(vi) Languages $\mathcal{H}(j)$, $\mathcal{L}(j)$, and $\mathcal{I}(j)$ are closed.

(vii) $\mathcal{H}_m(j) \subseteq \mathcal{H}(j)$, $\mathcal{L}_m(j) \subseteq \mathcal{L}(j)$, and $\mathcal{I}_m(j) \subseteq \mathcal{I}(j)$

Proof:

Assume that the n^{th} degree ($n \geq 1$) *parallel interface system* is *interface consistent* with respect to the alphabet partition. **(1)**

Let *system*(j) be the j^{th} *serial system extraction* of our parallel system. **(2)**

We will now show this implies *system*(j) satisfies **points i-vii.**

Point i: Show that $G(j) = G_H||_s G_{L_j}||_s G_{I_1}||_s \ldots ||_s G_{I_n}$

By definition, the *flat system* for a *serial interface system* is defined as follows:

$$G(j) = G_H(j)||_s G_L(j)||_s G_I(j)$$

Substituting in for DES $G_H(j)$, $G_L(j)$, and $G_I(j)$) (by **(2)**) gives as required:

$$G(j) = G_H||_s G_{L_j}||_s G_{I_1}||_s \ldots ||_s G_{I_n}$$

Point ii: Show that $\Sigma_I(j) = \Sigma_{I_j}$, $\Sigma_{IH}(j) = \Sigma_{IH}$, and $\Sigma_{IL}(j) = \Sigma_{IL_j}$

$$\begin{aligned}
\Sigma_I(j) &:= \Sigma_R(j) \dot{\cup} \Sigma_A(j), \text{ by definition.} \\
&= \Sigma_{R_j} \dot{\cup} \Sigma_{A_j}, \text{ by } \mathbf{(2)}. \\
&= \Sigma_{I_j} \\
\Sigma_{IH}(j) &:= \Sigma_H(j) \dot{\cup} \Sigma_R(j) \dot{\cup} \Sigma_A(j), \text{ by definition.} \\
&= \dot{\cup}_{k \in \{1,\ldots,(j-1),(j+1),\ldots,n\}} \Sigma_{I_k} \dot{\cup} \Sigma_H \dot{\cup} \Sigma_{R_j} \dot{\cup} \Sigma_{A_j}, \text{ by } \mathbf{(2)}. \\
&= \Sigma_{IH} \\
\Sigma_{IL}(j) &:= \Sigma_L(j) \dot{\cup} \Sigma_R(j) \dot{\cup} \Sigma_A(j), \text{ by definition.} \\
&= \Sigma_{L_j} \dot{\cup} \Sigma_{R_j} \dot{\cup} \Sigma_{A_j}, \text{ by } \mathbf{(2)}. \\
&= \Sigma_{IL_j}
\end{aligned}$$

Point iii: Show that $P_{IH}(j)^{-1} = P_j \cdot P_{IH}^{-1}$, $P_{IL}(j)^{-1} = P_j \cdot P_{IL_j}^{-1}$, and $\Sigma_I(j) = P_j \cdot P_{I_j}^{-1}$

By definition, the following natural projections for a *serial interface system* are defined as follows:

$$\begin{aligned}
P_{IH}(j) : \Sigma(j)^* &\rightarrow \Sigma_{IH}(j)^* \\
P_{IL}(j) : \Sigma(j)^* &\rightarrow \Sigma_{IL}(j)^* \\
P_I(j) : \Sigma(j)^* &\rightarrow \Sigma_I(j)^*
\end{aligned}$$

Substituting from **point ii**, gives:

$$\begin{aligned}
P_{IH}(j) : \Sigma(j)^* &\rightarrow \Sigma_{IH}^* \qquad (\mathbf{3}) \\
P_{IL}(j) : \Sigma(j)^* &\rightarrow \Sigma_{IL_j}^* \\
P_I(j) : \Sigma(j)^* &\rightarrow \Sigma_{I_j}^*
\end{aligned}$$

We first examine $P_{IH}(j)$. We first note that the following natural projections are defined as $P_{IH} : \Sigma^* \rightarrow \Sigma_{IH}^*$ and $P_j : \Sigma^* \rightarrow \Sigma(j)^*$. As $\Sigma_{IH} = \Sigma_{IH}(j) \subseteq \Sigma(j)$ and $\Sigma(j) \subseteq \Sigma$ (by **(2)** and **point ii**), we can see by **(3)** and the definition of the natural projection that the diagram in Figure 5.4 commutes. Similarly, we can see that the diagram in Figure 5.5 commutes.[3]

[3]To make the commutative diagram work, we extended the natural projections in the natural way to operate on subsets instead of strings.

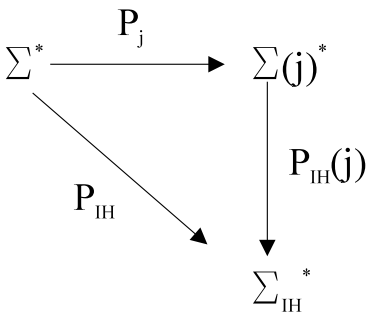


Figure 5.4: Commutative Diagram

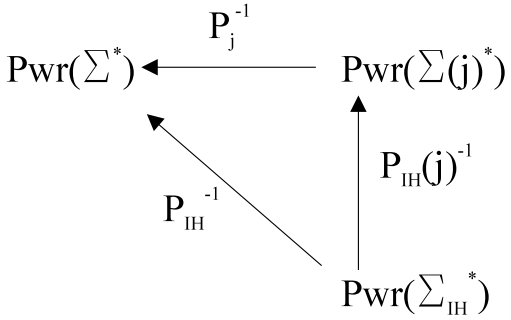


Figure 5.5: Commutative Diagram for Inverse Function

We thus have $P_{IH}^{-1} = P_j^{-1} \cdot P_{IH}(j)^{-1}$

$\Rightarrow P_j \cdot P_{IH}^{-1} = P_j \cdot P_j^{-1} \cdot P_{IH}(j)^{-1}$

$\Rightarrow P_j \cdot P_{IH}^{-1} = P_{IH}(j)^{-1}$

$\Rightarrow P_{IH}(j)^{-1} = P_j \cdot P_{IH}^{-1}$

Similarly, as $\Sigma_{IL_j} = \Sigma_{IL}(j) \subseteq \Sigma(j)$ (by **(2)** and **point ii**) and the following natural projection is defined as $P_{IL_j} : \Sigma^* \rightarrow \Sigma_{IL_j}^*$, we can conclude:

$$P_{IL}(j)^{-1} = P_j \cdot P_{IL_j}^{-1}$$

Similarly, as $\Sigma_{I_j} = \Sigma_I(j) \subseteq \Sigma(j)$ (by **(2)** and **point ii**) and the following natural projection is defined as $P_{I_j} : \Sigma^* \rightarrow \Sigma_{I_j}^*$, we can conclude:

$$P_I(j)^{-1} = P_j \cdot P_{I_j}^{-1}$$

Point iv: Show that the event set of $G_H(j)$ is $\Sigma_{IH}(j)$, the event set of $G_L(j)$ is $\Sigma_{IL}(j)$, and the event set of $G_I(j)$ is $\Sigma_I(j)$.

From **(1)**, we have that the system is *interface consistent*. This implies that *system*(j) is *serial interface consistent*. The result follows immediately.

Point v:
First, we must show that $\mathcal{H}(j) = P_j(\mathcal{H}) \cap [\cap_{k \in \{1,\ldots,(j-1),(j+1),\ldots,n\}} P_j(\mathcal{I}_k)]$

By definition, the language $\mathcal{H}(j)$ for a *serial interface system* is defined as follows:

$$\mathcal{H}(j) = P_{IH}(j)^{-1}(L(G_H(j)))$$

Substituting using **(2)** and **point iii**, we get:

$$\mathcal{H}(j) = P_j \cdot P_{IH}^{-1}(L(G_H||_s G_{I_1}||_s \ldots ||_s G_{I_{(j-1)}}||_s G_{I_{(j+1)}}||_s \ldots ||_s G_{I_n})) \quad \textbf{(4)}$$

We next note that, by **(1)**, we can apply **Proposition 21** and conclude that DES G_H is defined over event set Σ_{IH}, DES G_{I_i} is defined over event set Σ_{I_i}, and DES G_{L_i} is defined over event set Σ_{IL_i}, where $i \in \{1, \ldots, n\}$. **(5)**

This tells us that DES $G_H(j)$ is defined over Σ_{IH}.

To evaluate $L(G_H||_s G_{I_1}||_s \ldots ||_s G_{I_{(j-1)}}||_s G_{I_{(j+1)}}||_s \ldots ||_s G_{I_n})$, we need a natural projection $P_{I|IH_k} : \Sigma_{IH}^* \to \Sigma_{I_k}^*$, for $k \in \{1, \ldots, (j-1), (j+1), \ldots, n\}$.

After noting that $\Sigma_{I_k} \subseteq \Sigma_{IH}$ and $\Sigma_{IH} \subseteq \Sigma$, we can apply the same logic as in **point iii** and conclude:

$$P_{I|IH_k}^{-1} = P_{IH} \cdot P_{I_k}^{-1}$$

We can now evaluate $L(G_H||_s G_{I_1}||_s \ldots ||_s G_{I_{(j-1)}}||_s G_{I_{(j+1)}}||_s \ldots ||_s G_{I_n})$ and conclude:

$$L(G_H||_s G_{I_1}||_s \ldots ||_s G_{I_{(j-1)}}||_s G_{I_{(j+1)}}||_s \ldots ||_s G_{I_n}) = L(G_H) \cap [\cap_{k \in \{1, \ldots, (j-1), (j+1), \ldots, n\}} P_{IH} \cdot P_{I_k}^{-1}(L(G_{I_k}))]$$

Substituting into **(4)** gives:

$$\mathcal{H}(j) = P_j \cdot P_{IH}^{-1}(L(G_H) \cap [\cap_{k \in \{1, \ldots, (j-1), (j+1), \ldots, n\}} P_{IH} \cdot P_{I_k}^{-1}(L(G_{I_k}))])$$

$$\Rightarrow \mathcal{H}(j) = P_j \cdot P_{IH}^{-1}(L(G_H)) \cap [\cap_{k \in \{1, \ldots, (j-1), (j+1), \ldots, n\}} P_j \cdot P_{IH}^{-1} \cdot P_{IH} \cdot P_{I_k}^{-1}(L(G_{I_k}))] \quad \textbf{(6)}$$

As $\Sigma_{I_k} \subseteq \Sigma_{IH}$, we can now apply **Proposition 6** by taking $\Sigma_a = \Sigma_{IH}$, and $\Sigma_b = \Sigma_{I_k}$ ($k \in \{1, \ldots, (j-1), (j+1), \ldots, n\}$) and thus conclude:

$$P_{IH}^{-1} \cdot P_{IH} \cdot P_{I_k}^{-1} = P_{I_k}^{-1}$$

Substituting into **(6)** gives:

$$\mathcal{H}(j) = P_j \cdot P_{IH}^{-1}(L(G_H)) \cap [\cap_{k \in \{1, \ldots, (j-1), (j+1), \ldots, n\}} P_j \cdot P_{I_k}^{-1}(L(G_{I_k}))]$$

$$\Rightarrow \mathcal{H}(j) = P_j(\mathcal{H}) \cap [\cap_{k \in \{1, \ldots, (j-1), (j+1), \ldots, n\}} P_j(\mathcal{I}_k)]$$

Next, we must show that $\mathcal{H}_m(j) = P_j(\mathcal{H}) \cap [\cap_{k \in \{1, \ldots, (j-1), (j+1), \ldots, n\}} P_j(\mathcal{I}_{m_k})]$

Proof is identical to proof for $\mathcal{H}(j)$, after relabelling.

The proofs for the remaining languages for **point iv** are straightforward, and are presented together below.

$$\begin{aligned}
\mathcal{L}(j) &:= P_{IL}(j)^{-1}(L(G_L(j))), \text{ by definition.} \\
&= P_j \cdot P_{IL_j}^{-1}(L(G_{L_j})), \text{ by } \textbf{(2)} \text{ and } \textbf{point iii}. \\
&= P_j(\mathcal{L}_j) \\
\mathcal{L}_m(j) &:= P_{IL}(j)^{-1}(L_m(G_L(j))), \text{ by definition.}
\end{aligned}$$

$$
\begin{aligned}
&= P_j \cdot P_{IL_j}^{-1}(L_m(G_{L_j})), \text{ by } \textbf{(2)} \text{ and } \textbf{point iii}.\\
&= P_j\,(\mathcal{L}_{m_j})\\
\mathcal{I}(j) &:= P_I(j)^{-1}(L(G_I(j))), \text{ by definition.}\\
&= P_j \cdot P_{I_j}^{-1}(L(G_{I_j})), \text{ by } \textbf{(2)} \text{ and } \textbf{point iii}.\\
&= P_j\,(\mathcal{I}_j)\\
\mathcal{I}_m(j) &:= P_I(j)^{-1}(L_m(G_I(j))), \text{ by definition.}\\
&= P_j \cdot P_{I_j}^{-1}(L_m(G_{I_j})), \text{ by } \textbf{(2)} \text{ and } \textbf{point iii}.\\
&= P_j\,(\mathcal{I}_{m_j})
\end{aligned}
$$

Points vi-vii: Show that languages $\mathcal{H}(j)$, $\mathcal{L}(j)$, and $\mathcal{I}(j)$ are closed and that $\mathcal{H}_m(j) \subseteq \mathcal{H}(j)$, $\mathcal{L}_m(j) \subseteq \mathcal{L}(j)$, and $\mathcal{I}_m(j) \subseteq \mathcal{I}(j)$

From **(2)**, we know that $G_H(j) = G_H||_s G_{I_1}||_s \ldots ||_s G_{I_{(j-1)}}||_s G_{I_{(j+1)}}||_s \ldots ||_s G_{I_n}$

We can now apply **Proposition 5** and conclude that language $L(G_H(j))$ is closed, and $L_m(G_H(j)) \subseteq L(G_H(j))$

We next note that languages $L(G_L(j))$, and $L(G_I(j))$ are closed as $G_L(j) = G_{L_j}$ and $G_I(j) = G_{I_j}$ (by **(2)**), and by the definition of the closed behaviour of a DES.

From the definition of the closed behaviour and the marked language of a DES, we can conclude that:

$L_m(G_L(j)) \subseteq L(G_L(j))$, and $L_m(G_I(j)) \subseteq L(G_I(j))$.

We now apply **Proposition 1** repeatedly and conclude that $\mathcal{H}(j) = P_{IH}(j)^{-1}(L(G_H(j)))$, $\mathcal{L}(j) = P_{IL}(j)^{-1}(L(G_L(j)))$, and $\mathcal{I}(j) = P_I(j)^{-1}(L(G_I(j)))$ are closed.

We next apply **Proposition 3** repeatedly, and conclude:

$$
\begin{aligned}
P_{IH}(j)^{-1}(L_m(G_H(j))) = \mathcal{H}_m(j) &\subseteq \mathcal{H}(j) = P_{IH}(j)^{-1}(L(G_H(j)))\\
P_{IL}(j)^{-1}(L_m(G_L(j))) = \mathcal{L}_m(j) &\subseteq \mathcal{L}(j) = P_{IL}(j)^{-1}(L(G_L(j)))\\
P_I(j)^{-1}(L_m(G_I(j))) = \mathcal{I}_m(j) &\subseteq \mathcal{I}(j) = P_I(j)^{-1}(L(G_I(j)))
\end{aligned}
$$

QED

5.5.4 Proof of Proposition 24

Proof for **Proposition 24** on page 68: *If the* n^{th} *degree* ($n \geq 1$) parallel interface system *composed of DES* G_H, $G_{L_1}, \ldots, G_{L_n}, G_{I_1}, \ldots, G_{I_n}$, *is* level-wise nonblocking *and* interface consistent *with respect to the alphabet partition* $\Sigma := \dot\cup_{k\in\{1,\ldots,n\}}(\Sigma_{L_k} \,\dot\cup\, \Sigma_{R_k} \,\dot\cup\, \Sigma_{A_k}) \,\dot\cup\, \Sigma_H$, *then*

$(\forall s \in \mathcal{H} \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_j \cap \mathcal{I}_j)])$

$(\exists l \in \Sigma_{IL}^*)$ *s.t.* $(sl \in \mathcal{H} \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})])$

Proof:

Assume system is *level-wise nonblocking* and *interface consistent.* **(1)**

Let $s \in \mathcal{H} \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_j \cap \mathcal{I}_j)]$ **(2)**

We will now show this implies: $(\exists l \in \Sigma^*_{IL})$ s.t. $(sl \in \mathcal{H} \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})])$

To do this, we will use an inductive proof. We define $\Sigma^*_{IL_0} = \Sigma^*$, $\mathcal{L}_{m_0} = \mathcal{I}_{m_0} = \Sigma^*$ and $\mathcal{L}_{n+1} = \mathcal{I}_{n+1} = \Sigma^*$. Here we are using the fact that intersection with Σ^* is the identity operator as all other languages are subsets of Σ^*. This is to handle the boundary cases of $k = 0$ and $k = n$ in order to avoid intersection with $\emptyset$.

Claim to be proven:

For $k \in \{0, 1, \ldots, n\}$, there exist strings $l_i \in \Sigma^*_{IL_i}$, $i \in \{0, 1, \ldots, k\}$, such that:

$$sl_0l_1 \ldots l_k \in \mathcal{H} \cap [\cap_{v\in\{0,1,\ldots,k\}}(\mathcal{L}_{m_v} \cap \mathcal{I}_{m_v})] \cap [\cap_{w\in\{k+1,\ldots,n+1\}}(\mathcal{L}_w \cap \mathcal{I}_w)] \quad \textbf{(3)}$$

We will first prove the initial case $k = 0$, and then the general case of $k \in \{1, \ldots, n\}$. We can then conclude by induction that the claim has been proven.

Initial Case: $k = 0$

We take $l_0 = \epsilon \in \Sigma^*_{IL_0} = \Sigma^*_{IL}$. We immediately have $sl_0 = s \in \mathcal{H} \cap [\cap_{k\in\{1,\ldots,n\}}(\mathcal{L}_k \cap \mathcal{I}_k)]$ (by **(2)**)

We have automatically $sl_0 \in (\mathcal{L}_{n+1} \cap \mathcal{I}_{n+1}) = \Sigma^*$

Initial case complete.

Inductive Step:

Let $k \in \{1, \ldots, n\}$. Assume there exist strings $l_i \in \Sigma^*_{IL_i}$, $i \in \{0, 1, \ldots, k-1\}$, and that they satisfy **(3)** when $k-1$ is substituted for k.

$$\Rightarrow sl_0l_1 \ldots l_{k-1} \in \mathcal{H} \cap [\cap_{v\in\{0,1,\ldots,(k-1)\}}(\mathcal{L}_{m_v} \cap \mathcal{I}_{m_v})] \cap [\cap_{w\in\{k,\ldots,n+1\}}(\mathcal{L}_w \cap \mathcal{I}_w) \quad \textbf{(4)}$$

We will show this implies that we can construct string $l_k \in \Sigma^*_{IL_k}$, such that $sl_0l_1 \ldots l_k \in \mathcal{H} \cap [\cap_{v\in\{0,1,\ldots,k\}}(\mathcal{L}_{m_v} \cap \mathcal{I}_{m_v})] \cap [\cap_{w\in\{k+1,\ldots,n+1\}}(\mathcal{L}_w \cap \mathcal{I}_w)]$

Our approach will be to apply **Proposition 11** to *system*(k), the k^{th} *serial extraction system* of our parallel system.

We first note that **(4)** implies $sl_0l_1 \ldots l_{k-1} \in \mathcal{H} \cap [\cap_{w\in\{1,\ldots,n\}}(\mathcal{L}_w \cap \mathcal{I}_w)]$

$\Rightarrow P_k(sl_0l_1 \ldots l_{k-1}) \in P_k(\mathcal{H}) \cap P_k(\mathcal{L}_k) \cap [\cap_{w\in\{1,\ldots,n\}} P_k(\mathcal{I}_w)] = \mathcal{H}(k) \cap \mathcal{L}(k) \cap \mathcal{I}(k)$ by **Proposition 23**

We can now apply **Proposition 11** to *system* (k) by taking $P_k(sl_0l_1 \ldots l_{k-1})$ to be string s in that proposition. We thus conclude:

$$(\exists l_k \in \Sigma^*_{IL_k}) \text{ s.t. } P_k(sl_0l_1 \ldots l_{k-1})l_k \in \mathcal{H}(k) \cap \mathcal{L}_m(k) \cap \mathcal{I}_m(k)$$

We now note that $P_k(l_k) = l_k$ as $\Sigma_{IL_k} \subseteq \Sigma(k)$. We can thus conclude:

$$P_k(sl_0l_1 \ldots l_{k-1}l_k) \in \mathcal{H}(k) \cap \mathcal{L}_m(k) \cap \mathcal{I}_m(k) \quad \textbf{(5)}$$

We will now show that this implies:

$$sl_0l_1 \ldots l_{k-1}l_k \in \mathcal{H} \cap [\cap_{w\in\{1,\ldots,k-1,k+1,\ldots,n\}}\mathcal{I}_w] \cap \mathcal{L}_{m_k} \cap \mathcal{I}_{m_k}$$

Substituting into **(5)** for $\mathcal{H}(k)$, $\mathcal{L}_m(k)$, and $\mathcal{I}_m(k)$ (by **Proposition 23**), we have:

$$\begin{aligned} P_k(sl_0l_1 \ldots l_{k-1}l_k) \in P_k \cdot P^{-1}_{IH}(L(G_H)) \cap [\cap_{w\in\{1,\ldots,k-1,k+1,\ldots,n\}} P_k \cdot P^{-1}_{I_w}(L(G_{I_w}))] \\ \cap P_k \cdot P^{-1}_{IL_k}(L_m(G_{L_k})) \cap P_k \cdot P^{-1}_{I_k}(L_m(G_{I_k})) \end{aligned} \quad \textbf{(6)}$$

From **Proposition 23**, we have $\Sigma_{IH} \subseteq \Sigma(k)$. We can now apply **Corollary 2** by taking $\Sigma_a = \Sigma(k)$, $\Sigma_b = \Sigma_{IH}$, and $L_b = L(G_H)$ and thus conclude:

$$sl_0l_1 \ldots l_{k-1}l_k \in P^{-1}_{IH}(L(G_H)) = \mathcal{H} \quad \textbf{(7)}$$

Similarly, we have $\Sigma_{I_w} \subseteq \Sigma(k)$ for $w \in \{1, \ldots, k-1, k+1, \ldots, n\}$. We can now apply **Corollary 2** by taking $\Sigma_a = \Sigma(k)$, $\Sigma_b = \Sigma_{I_w}$, and $L_b = L(G_{I_w})$ and thus conclude:

$$sl_0l_1 \ldots l_{k-1}l_k \in P^{-1}_{I_w}(L(G_{I_w})) = \mathcal{I}_w \quad \textbf{(8)}$$

Similarly, we have $\Sigma_{IL_k} \subseteq \Sigma(k)$. We can now apply **Corollary 2** by taking $\Sigma_a = \Sigma(k)$, $\Sigma_b = \Sigma_{IL_k}$, and $L_b = L_m(G_{L_k})$ and thus conclude:

$$sl_0l_1 \ldots l_{k-1}l_k \in P^{-1}_{IL_k}(L_m(G_{L_k})) = \mathcal{L}_{m_k} \quad \textbf{(9)}$$

Similarly, we have $\Sigma_{I_k} \subseteq \Sigma(k)$. We can now apply **Corollary 2** by taking $\Sigma_a = \Sigma(k)$, $\Sigma_b = \Sigma_{I_k}$, and $L_b = L_m(G_{I_k})$ and thus conclude:

$$sl_0l_1 \ldots l_{k-1}l_k \in P^{-1}_{IL_k}(L_m(G_{I_k})) = \mathcal{I}_{m_k}$$

Combining with **(7)** - **(9)**, we thus have:

$$sl_0l_1 \ldots l_{k-1}l_k \in \mathcal{H} \cap [\cap_{w\in\{1,\ldots,k-1,k+1,\ldots,n\}}\mathcal{I}_w] \cap \mathcal{L}_{m_k} \cap \mathcal{I}_{m_k} \quad \textbf{(10)}$$

We also have automatically $sl_0l_1 \ldots l_{k-1}l_k \in (\mathcal{L}_{n+1} \cap \mathcal{I}_{n+1}) = \Sigma^*$ **(11)**

We will now show that $sl_0l_1 \ldots l_{k-1}l_k \in \mathcal{H} \cap [\cap_{v\in\{0,1,\ldots,k\}}(\mathcal{L}_{m_v} \cap \mathcal{I}_{m_v})] \cap [\cap_{w\in\{k+1,\ldots,n+1\}}(\mathcal{L}_w \cap \mathcal{I}_w)]$

This means showing: $sl_0l_1 \ldots l_{k-1}l_k \in \cap_{v\in\{0,1,\ldots,k-1\}}(\mathcal{L}_{m_v} \cap \mathcal{I}_{m_v}) \cap [\cap_{w\in\{k+1,\ldots,n\}}\mathcal{L}_w]$

As $\mathcal{L}_{m_0} = \mathcal{I}_{m_0} = \Sigma^*$, $sl_0l_1 \ldots l_{k-1}l_k \in \mathcal{L}_{m_0} \cap \mathcal{I}_{m_0}$ is automatic. **(12)**

We next note that by **(4)**, we have $sl_0l_1 \ldots l_{k-1} \in \mathcal{H} \cap [\cap_{v \in \{0,1,\ldots,(k-1)\}}(\mathcal{L}_{m_v} \cap \mathcal{I}_{m_v})] \cap [\cap_{w \in \{k+1,\ldots,n\}}\mathcal{L}_w]$

From **(1)** we have $\Sigma_{IL_k} \cap \Sigma_{IL_v} = \emptyset$, for $v \in \{1, \ldots, (k-1)\}$. As $l_k \in \Sigma_{IL_k}$, we have $P_{IL_v}(l_k) = \epsilon$. This implies $P_{IL_v}(sl_0l_1 \ldots l_{k-1}l_k) = P_{IL_v}(sl_0l_1 \ldots l_{k-1})$. We can now apply **Proposition 27**, **point d**, and conclude:

$$sl_0l_1 \ldots l_{k-1}l_k \in \mathcal{L}_{m_v} \quad \textbf{(13)}$$

Similarly we have $P_{I_v}(sl_0l_1 \ldots l_{k-1}l_k) = P_{I_v}(sl_0l_1 \ldots l_{k-1})$, for $v \in \{1, \ldots, (k-1)\}$. We can now apply **Proposition 27**, **point f**, and conclude:

$$sl_0l_1 \ldots l_{k-1}l_k \in \mathcal{I}_{m_v} \quad \textbf{(14)}$$

Similarly we have $P_{IL_w}(sl_0l_1 \ldots l_{k-1}l_k) = P_{IL_w}(sl_0l_1 \ldots l_{k-1})$, for $w \in \{k+1, \ldots, n\}$. We can now apply **Proposition 27**, **point c**, and conclude:

$$sl_0l_1 \ldots l_{k-1}l_k \in \mathcal{L}_w$$

Combining with **(12)** - **(14)**, we have:

$$sl_0l_1 \ldots l_{k-1}l_k \in \cap_{v \in \{0,1,\ldots,k-1\}}(\mathcal{L}_{m_v} \cap \mathcal{I}_{m_v}) \cap [\cap_{w \in \{k+1,\ldots,n\}}\mathcal{L}_w]$$

Combining with **(10)** and **(11)**, we have:

$$sl_0l_1 \ldots l_{k-1}l_k \in \mathcal{H} \cap [\cap_{v \in \{0,1,\ldots,k\}}(\mathcal{L}_{m_v} \cap \mathcal{I}_{m_v})] \cap [\cap_{w \in \{k+1,\ldots,n+1\}}(\mathcal{L}_w \cap \mathcal{I}_w)], \text{ as required.}$$

Inductive step complete.

We have now proven the **Initial case** and the **Inductive step**. We now conclude that the **Claim** is true, by induction.

Taking $k = n$ and using fact that $l_0 = \epsilon$, we thus can conclude there exists strings $l_i \in \Sigma^*_{IL_i}$, $i \in \{1, \ldots, n\}$, such that: $sl_1 \ldots l_n \in \mathcal{H} \cap [\cap_{j \in \{1,\ldots,n\}}(\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})]$

We thus take $l = l_1 \ldots l_n$ and we have $l \in \Sigma^*_{IL}$ and $sl \in \mathcal{H} \cap [\cap_{j \in \{1,\ldots,n\}}(\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})]$, as required.

QED

5.5.5 Proof of Proposition 25

Proof for **Proposition 25** on page 69: *If the* n^{th} *degree* ($n \geq 1$) parallel interface system *composed of DES* G_H, $G_{L_1}, \ldots, G_{L_n}$, $G_{I_1}, \ldots, G_{I_n}$, *is* level-wise nonblocking *and* interface consistent *with respect to the alphabet partition* $\Sigma := \dot\cup_{k \in \{1,\ldots,n\}}(\Sigma_{L_k} \dot\cup \Sigma_{R_k} \dot\cup \Sigma_{A_k}) \dot\cup \Sigma_H$, *then*

$(\forall s \in \mathcal{H} \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_j \cap \mathcal{I}_j)])$

$\quad (\exists h \in \Sigma_{IH}^*)\ \ (sh \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}\mathcal{I}_{m_j}])$

Proof:

Assume system is *level-wise nonblocking* and *interface consistent.* **(1)**

Let $s \in \mathcal{H} \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_j \cap \mathcal{I}_j)]$ **(2)**

We will now show this implies:

$$(\exists h \in \Sigma_{IH}^*)\ \ (sh \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}\mathcal{I}_{m_j}])$$

We will start by examining the 1^{st} *serial extraction system* of the parallel system. We will show that we can construct a string $h \in \Sigma_{IH}^*$ with the property that $P_1(s)h \in \mathcal{H}_m(1) \cap \mathcal{I}_m(1)$.

We first note that from **(2)**, we have $s \in \mathcal{H} \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_j \cap \mathcal{I}_j)]$

From **Proposition 23**, we thus have:
$P_1(s) \in P_1(\mathcal{H}) \cap [\cap_{j\in\{2,\ldots,n\}}P_1(\mathcal{I}_j)] \cap P_1(\mathcal{L}_{1)} \cap P_1(\mathcal{I}_1) = \mathcal{H}(1) \cap \mathcal{L}(1) \cap \mathcal{I}(1)$ **(3)**

We start be noting that *system*(1) is *serial level-wise nonblocking*, as the parallel system is *level-wise nonblocking* (by **(1)**). Combining with **(3)**, we can thus apply **Point I** of the *serial level-wise nonblocking* definition and conclude:

$$(\exists h' \in \Sigma(1)^*) \text{ s.t. } P_1(s)h' \in \mathcal{H}_m(1) \cap \mathcal{I}_m(1)$$

We next note that:

$$\begin{aligned} P_{IH}(1)\,(P_1(s)h') &= P_{IH}(1)(P_1(s))P_{IH}(1)(h') \\ &= P_{IH}(1)(P_1(s))P_{IH}(1)(P_{IH}(1)(h')) \text{ as the natural projection} \\ &\quad \text{is idempotent.} \\ &= P_{IH}(1)(P_1(s)P_{IH}(1)(h')) \qquad (4) \end{aligned}$$

We can now apply **Proposition 8**, **point b**, and conclude:

$$P_1(s)P_{IH}(1)(h') \in \mathcal{H}_m(1) \qquad \textbf{(5)}$$

As $\Sigma_I(1) \subseteq \Sigma_{IH}(1)$, we can conclude by **(4)** that $P_I(1)(P_1(s)h') = P_I(1)(P_1(s)P_{IH}(1)(h'))$

We can now apply **Proposition 8**, **point f**, and conclude:

$$P_1(s)P_{IH}(1)(h') \in \mathcal{I}_m(1) \qquad \textbf{(6)}$$

We next note that as $\Sigma(1) \subseteq \Sigma$ and $\Sigma_{IH} = \Sigma_{IH}(1)$ (by **Proposition 23**), we can conclude $P_{IH}(h') = P_{IH}(1)(h')$.

Combining with **(5)** and **(6)**, we can conclude:

$$P_1(s)P_{IH}(h') \in \mathcal{H}_m(1) \cap \mathcal{I}_m(1)$$

We then take $h = P_{IH}(h')$ and we have:

$$h \in \Sigma_{IH}^* \text{ and } P_1(s)h \in \mathcal{H}_m(1) \cap \mathcal{I}_m(1) \qquad \textbf{(7)}$$

We will now show that this implies: $sh \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}\mathcal{I}_{m_j}]$

From **(7)**, substituting for $\mathcal{H}_m(1)$ and $\mathcal{I}_m(1)$ (by **Proposition 23**) and noting $P_1(h) = h$ as $h \in \Sigma_{IH} \subseteq \Sigma(1)$, we can conclude that:

$$P_1(sh) \in P_1 \cdot P_{IH}^{-1}(L_m(G_H)) \cap [\cap_{j\in\{2,\ldots,n\}} P_1 \cdot P_{I_j}^{-1}(L_m(G_{I_j}))] \cap P_1 \cdot P_{I_1}^{-1}(L_m(G_{I_1}))$$

From **Proposition 23**, we have $\Sigma_{IH} \subseteq \Sigma(1)$. We can now apply **Corollary 2** by taking $\Sigma_a = \Sigma(1)$, $\Sigma_b = \Sigma_{IH}$, and $L_b = L_m(G_H)$ and thus conclude:

$$sh \in P_{IH}^{-1}(L_m(G_H)) = \mathcal{H}_m \qquad \textbf{(8)}$$

Similarly, we have $\Sigma_{I_j} \subseteq \Sigma(1)$ for $j \in \{1, \ldots, n\}$. We can now apply **Corollary 2** by taking $\Sigma_a = \Sigma(1)$, $\Sigma_b = \Sigma_{I_j}$, and $L_b = L_m(G_{I_j})$ and thus conclude:

$$sh \in P_{I_j}^{-1}(L_m(G_{I_j})) = \mathcal{I}_{m_j}$$

Combining with **(8)**, we have:

$$sh \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}\mathcal{I}_{m_j}]$$

Combining with **(7)**, we thus have $h \in \Sigma_{IH}^*$ with the required property that $(sh \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}\mathcal{I}_{m_j}])$

QED

5.5.6 Proof of Proposition 26

Proof for **Proposition 26** on page 69: *If the* n^{th} *degree* $(n \geq 1)$ parallel interface system *composed of DES* G_H, $G_{L_1}, \ldots, G_{L_n}$, $G_{I_1}, \ldots, G_{I_n}$, *is* level-wise nonblocking *and* interface consistent *with respect to the alphabet partition* $\Sigma := \dot\cup_{k\in\{1,\ldots,n\}}(\Sigma_{L_k} \,\dot\cup\, \Sigma_{R_k} \,\dot\cup\, \Sigma_{A_k}) \;\dot\cup\, \Sigma_H$, *then*

$(\forall s \in \mathcal{H} \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})])(\forall h \in \Sigma_{IH}^*)$

$\quad (sh \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}\mathcal{I}_{m_j}]) \Rightarrow (\exists u \in \Sigma^*)$ *s.t.* $(su \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})]$

Proof:

Assume system is *level-wise nonblocking* and *interface consistent.* **(1)**

Let $s \in \mathcal{H} \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})]$, $h \in \Sigma_{IH}^*$, and $sh \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}\mathcal{I}_{m_j}]$ **(2)**

We will now show this implies $(\exists u \in \Sigma^*)$ s.t. $(su \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})]$

To do this, we will use string h and apply **Proposition 15** iteratively, to construct n strings labelled $u_i \in \Sigma(i)^*$, $i \in \{1,\ldots,n\}$, with the properties $su_i \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}\mathcal{I}_{m_j}] \cap \mathcal{L}_{m_i}$ (i.e. string u_i takes the *high level* and the i^{th} *low level system* to a marked state) and $P_{IH}(u_i) = h$ (string h is the high level image of each string u_i). We will then use these n strings to construct the desired string u.

Iterative step:

For each $i \in \{1,\ldots,n\}$, construct u_i as follows:

To apply **Proposition 15** to *system*(i), the i^{th} *serial extraction system* of our parallel system, we must first show that $h \in \Sigma_{IH}^*$ (auto from **(2)**), $P_i(s) \in \mathcal{H}_m(i) \cap \mathcal{I}_m(i)$, and that $P_i(s) \in \mathcal{H}(i) \cap \mathcal{L}(i) \cap \mathcal{I}_m(i)$.

From **(2)**, we have $sh \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}\mathcal{I}_{m_j}]$

As $P_i(h) = h$ since $h \in \Sigma_{IH}^* \subseteq \Sigma(i)$ (by **Proposition 23**), we can conclude:

$$P_i(sh) = P_i(s)h \in P_i(\mathcal{H}_m) \cap [\cap_{j\in\{1,\ldots,i-1,i+1,\ldots,n\}}P_i(\mathcal{I}_{m_j})] \cap P_i(\mathcal{I}_{m_i})$$

$\Rightarrow P_i(s)h \in \mathcal{H}_m(i) \cap \mathcal{I}_m(i)$ (by **Proposition 23**) **(3)**

Our last step before we can apply **Proposition 15** is to show that $P_i(s) \in \mathcal{H}(i) \cap \mathcal{L}(i) \cap \mathcal{I}_m(i)$.

From **(2)**, we have $s \in \mathcal{H} \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_j \cap \mathcal{I}_j)]$ **(4)**

From **Proposition 23**, we thus have:
$P_i(s) \in P_i(\mathcal{H}) \cap [\cap_{j\in\{1,\ldots,i-1,i+1,\ldots,n\}}P_i(\mathcal{I}_j)] \cap P_i(\mathcal{L}_{i)} \cap P_i(\mathcal{I}_i) = \mathcal{H}(i) \cap \mathcal{L}(i) \cap \mathcal{I}(i)$ **(5)**

We have $P_i(s) \in \mathcal{H}(i) \cap \mathcal{L}(i)$ from **(15)**, so all that remains is to show that $P_i(s) \in \mathcal{I}_m(i)$.

From **(2)**, we have $s \in \mathcal{I}_{m_i}$. This implies $P_i(s) \in P_i(\mathcal{I}_{m_i}) = \mathcal{I}_m(i)$

Combining with **(2) (3)**, and **(5)**, we now apply **Proposition 15** by taking $P_i(s)$ to be string s in that proposition and conclude:

$$(\exists u_i \in \Sigma(i)^*) \text{ s.t. } P_i(s)u_i \in (\mathcal{H}_m(i) \cap \mathcal{L}_m(i) \cap \mathcal{I}_m(i)) \wedge (P_{IH}(i)(u_i) = h) \quad \textbf{(6)}$$

We next note that as $\Sigma(i) \subseteq \Sigma$ and $\Sigma_{IH} = \Sigma_{IH}(i)$ (by **Proposition 23**), we can conclude

$$P_{IH}(u_i) = P_{IH}(i)(u_i) = h. \quad \textbf{(7)}$$

We now note that $u_i \in \Sigma(i)^*$ implies that $P_i(u_i) = u_i$. Combining with **(6)** and substituting for $\mathcal{H}_m(i)$, $\mathcal{L}_m(i)$ (using **Proposition 23**), and $\mathcal{I}_m(i)$, we have:

$$P_i(su_i) \in P_i{\cdot}P_{IH}^{-1}(L_m(G_H)) \cap [\cap_{j\in\{1,\ldots,n\}} P_i{\cdot}P_{I_j}^{-1}(L_m(G_{I_j}))] \cap P_i{\cdot}P_{IL_i}^{-1}(L_m(G_{L_i})) \quad \textbf{(8)}$$

From **Proposition 23**, we have $\Sigma_{IH} \subseteq \Sigma(i)$. We can now apply **Corollary 2** by taking $\Sigma_a = \Sigma(i)$, $\Sigma_b = \Sigma_{IH}$, and $L_b = L_m(G_H)$ and thus conclude:

$$su_i \in P_{IH}^{-1}(L_m(G_H)) = \mathcal{H}_m \quad \textbf{(9)}$$

Similarly, we have $\Sigma_{I_j} \subseteq \Sigma(i)$ for $j \in \{1,\ldots,n\}$. We can now apply **Corollary 2** by taking $\Sigma_a = \Sigma(i)$, $\Sigma_b = \Sigma_{I_j}$, and $L_b = L_m(G_{I_j})$ and thus conclude:

$$su_i \in P_{I_j}^{-1}(L_m(G_{I_j})) = \mathcal{I}_{m_j} \quad \textbf{(10)}$$

Similarly, we have $\Sigma_{IL_i} \subseteq \Sigma(i)$. We can now apply **Corollary 2** by taking $\Sigma_a = \Sigma(i)$, $\Sigma_b = \Sigma_{IL_i}$, and $L_b = L_m(G_{L_i})$ and thus conclude:

$$su_i \in P_{IL_i}^{-1}(L_m(G_{L_i})) = \mathcal{L}_{m_i}$$

Combining with **(9)** and **(10)**, we have:

$su_i \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}} \mathcal{I}_{m_j}] \cap \mathcal{L}_{m_i}$, as required.

Iterative step complete.

Now that we have completed the **iterative step**, we have shown the following:

$$(\forall i \in \{1,\ldots,n\})(\exists u_i \in \Sigma(i)^*)\ \ (su_i \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}} \mathcal{I}_{m_j}] \cap \mathcal{L}_{m_i}) \wedge (P_{IH}(u_i) = h) \quad \textbf{(11)}$$

We will now use this information to construct a string $u \in \Sigma^*$ with the property:
$su \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}} (\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})]$

We take u to be any string in set $\cap_{i\in\{1,\ldots,n\}} P_i^{-1}(u_i)$ **(12)**

We know that the set is non-empty for the following reasons:

- For each $i \in \{1,\ldots,n\}$, we have $u_i \in \Sigma(i)^*$ where:
  $\Sigma(i) := \Sigma_{IH} \dot\cup \Sigma_{L_i} = \Sigma - (\dot\cup_{j\in\{1,\ldots,i-1,i+1,\ldots,n\}} \Sigma_{L_j})$.
- The only events strings u_i have in common are $\sigma \in \Sigma_{IH}$.
- All strings u_i agree on common events as $P_{IH}(u_i) = h$

From **(12)**, we have $(\forall i \in \{1,\ldots,n\})P_i(u) = u_i$. As $\Sigma_{IH} \subseteq \Sigma(i)$ (by **Proposition 23**) and $h \in \Sigma_{IH}^*$ (by **(22)**), we can conclude:

$$P_{IH}(u) = P_{IH}(u_i) = h = P_{IH}(h). \quad \textbf{(13)}$$

From **(2)**, we have: $sh \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}\mathcal{I}_{m_j}]$

We can now apply **Proposition 27**, **point b**, and conclude:

$$su \in \mathcal{H}_m \quad \textbf{(14)}$$

As $\Sigma_{I_j} \subseteq \Sigma_{IH}$ for $j \in \{1,\ldots,n\}$, we can conclude by **(13)** that $P_{I_j}(u) = P_{I_j}(h)$. We can now apply **Proposition 27**, **point f**, and conclude:

$$su \in \mathcal{I}_{m_j}$$

Combining with **(14)**, we can conclude:

$$su \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}\mathcal{I}_{m_j}] \quad \textbf{(15)}$$

All that remains is to show $su \in \cap_{j\in\{1,\ldots,n\}}\mathcal{L}_{m_j}$

From **(11)**, we have $su_j \in \mathcal{L}_{m_j}$ for $j \in \{1,\ldots,n\}$. From **(12)**, we have $P_j(u) = u_j = P_j(u_j)$ as $u_j \in \Sigma(j)^*$. As $\Sigma_{IL_j} \subseteq \Sigma(j)$ (by **Proposition 23**), we can conclude $P_{IL_j}(u) = P_{IL_j}(u_j)$. We can now apply **Proposition 27**, **point d**, and conclude:

$$su \in \mathcal{L}_{m_j}$$

Combining with **(15)**, we have: $su \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})] = L_m(G)$

QED

Chapter 6

Parallel Case: Controllability

Now that we have discussed nonblocking in the parallel interface setting, we now consider controllability. In the remainder of this chapter, we will define our setting and notation and then present some supporting propositions, followed by the serial controllability theorem.

6.1 Definitions and Notation

We now present some definitions and notation that will be useful in simplifying proofs. As in the serial case, we need to decompose the n^{th} degree ($n \geq 1$) *parallel interface system* into its plant and supervisor components. For the remainder of this section, the index j is defined to be $j \in \{1, \ldots, n\}$.

We now define the *high level plant* to to be $\mathcal{G}_H$, and the *high level supervisor* to be $\mathcal{S}_H$ (both defined over Σ_{IH}). Similarly, the j^{th} *low level plant* and *supervisor* are $\mathcal{G}_{L_j}$ and $\mathcal{S}_{L_j}$(defined over Σ_{IL_j}). To be consistent with our definitions in Chapter 5, we define the following identities for the *high level subsystem* and j^{th} *low level subsystem* as follows:

$$G_H := \mathcal{G}_H||_s\mathcal{S}_H \qquad\qquad G_{L_j} := \mathcal{G}_{L_j}||_s\mathcal{S}_{L_j}$$

We now have two ways to describe our system for the parallel case, depending on the level of detail required. We will call the original method described in Chapter 5 in terms of an *interface* and *high* and *low subsystems*, the *parallel subsystem based form.* This form is useful as it simplifies nonblocking definitions and proofs. We call the above method, given in terms of an interface and plants and supervisors, the *parallel general form* as the *parallel subsystem based form* can be recovered by applying the above identities. When we refer terms applicable to both forms (e.g. the *high level*), we

will simply state the term, allowing the type of the system to make our meaning clear.

Our next step is to define the *flat supervisor* and *plant* for our system.

$$\textbf{Plant} := \mathcal{G}_H||_s\mathcal{G}_{L_1}||_s \ldots ||_s\mathcal{G}_{L_n} \qquad \textbf{Sup} := \mathcal{S}_H||_s\mathcal{S}_{L_1}||_s \ldots ||_s\mathcal{S}_{L_n}||_sG_{I_1}||_s \ldots ||_sG_{I_n}$$

We next want to express the languages of **Plant** and **Sup** in terms of their components. To do this, we need to first define the following useful languages::

$$\mathbf{H} := P_{IH}^{-1}L(\mathcal{G}_H), \qquad \mathbf{H}_\mathcal{S} := P_{IH}^{-1}L(\mathcal{S}_H), \qquad \subseteq \Sigma^*$$
$$\mathbf{L}_j := P_{IL_j}^{-1}L(\mathcal{G}_{L_j}), \qquad \mathbf{L}_{\mathcal{S}_j} := P_{IL_j}^{-1}L(\mathcal{S}_{L_j}), \qquad \subseteq \Sigma^*$$

We can now express the languages of **Plant** and **Sup** as follows:

$$L(\textbf{Plant}) = \mathbf{H} \cap [\cap_{k\in\{1,\ldots,n\}}\mathbf{L}_k] \qquad L(\textbf{Sup}) = \mathbf{H}_\mathcal{S} \cap [\cap_{k\in\{1,\ldots,n\}}(\mathbf{L}_{\mathcal{S}_k} \cap \mathcal{I}_k)]$$

This allows us to present the proposition below that collects together several similar propositions. As it will be common in the proofs in this report to show that membership in languages such as **H** are dependent only on events in specific subsets (for **H**, events in subset Σ_{IH}), this proposition will be very useful.

Proposition 27

(a) $(\forall s, s' \in \Sigma^*)\ s \in \mathbf{H}$ *and* $P_{IH}(s) = P_{IH}(s') \Rightarrow s' \in \mathbf{H}$

(b) $(\forall s, s' \in \Sigma^*)\ s \in \mathbf{H}_\mathcal{S}$ *and* $P_{IH}(s) = P_{IH}(s') \Rightarrow s' \in \mathbf{H}_\mathcal{S}$

(c) $(\forall k \in \{1, \ldots, n\})(\forall s, s' \in \Sigma^*)\ s \in \mathbf{L}_k$ *and* $P_{IL_k}(s) = P_{IL_k}(s') \Rightarrow s' \in \mathbf{L}_k$

(d) $(\forall k \in \{1, \ldots, n\})(\forall s, s' \in \Sigma^*)\ s \in \mathbf{L}_{\mathcal{S}_k}$ *and* $P_{IL_k}(s) = P_{IL_k}(s') \Rightarrow s' \in \mathbf{L}_{\mathcal{S}_k}$

Proof:

Points a-b:

Identical to the proof of **point a** of **Proposition 8**, after substitution.

Points c-d:

Let $k \in \{1, \ldots, n\}$, then identical to the proof of **point a** of **Proposition 8**, after substitution.

QED

6.2 Serial System Extraction: General Form

We now extend the definition of the j^{th} *serial system extraction*, defined in Section 5.2, to operate on the general form of an n^{th} degree ($n \geq 1$) *parallel interface system*. The subsystem form of the definition can be recovered by using the identities $G_H(j) = \mathcal{G}_H(j)||_s\mathcal{S}_H(j)$, $G_L(j) = \mathcal{G}_L(j)||_s\mathcal{S}_L(j)$. Normally, we will simply refer to the j^{th} *serial system extraction,* as the type of the parallel system will make clear which definition is intended.

j^{th} Serial System Extraction: General Form For the n^{th} degree ($n \geq 1$) *parallel interface system* composed of DES $\mathcal{G}_H$, $\mathcal{G}_{L_1}, \ldots, \mathcal{G}_{L_n}$, $\mathcal{S}_H$, $\mathcal{S}_{L_1}, \ldots, \mathcal{S}_{L_n}$, $G_{I_1}, \ldots, G_{I_n}$, with alphabet partition $\Sigma := \dot{\cup}_{k\in\{1,\ldots,n\}}(\Sigma_{L_k} \,\dot{\cup}\, \Sigma_{R_k} \,\dot{\cup}\, \Sigma_{A_k}) \,\dot{\cup}\, \Sigma_H$, the j^{th} *serial system extraction,* denoted by *system*(j), is composed of the following elements:

$$
\begin{aligned}
\mathcal{G}_H(j) &:= \mathcal{G}_H||_sG_{I_1}||_s \ldots ||_sG_{I_{(j-1)}}||_sG_{I_{(j+1)}}||_s \ldots ||_sG_{I_n} \\
\mathcal{S}_H(j) &:= \mathcal{S}_H \\
\mathcal{G}_L(j) &:= \mathcal{G}_{L_j} \\
\mathcal{S}_L(j) &:= \mathcal{S}_{L_j} \\
G_I(j) &:= G_{I_j}
\end{aligned}
$$

$$
\begin{aligned}
\Sigma_H(j) &:= \dot{\cup}_{k\in\{1,\ldots,(j-1),(j+1),\ldots,n\}}\Sigma_{I_k} \,\dot{\cup}\, \Sigma_H \\
\Sigma_L(j) &:= \Sigma_{L_j} \\
\Sigma_R(j) &:= \Sigma_{R_j} \\
\Sigma_A(j) &:= \Sigma_{A_j}
\end{aligned}
$$

$$
\begin{aligned}
\Sigma(j) &:= \Sigma_H(j) \,\dot{\cup}\, \Sigma_L(j) \,\dot{\cup}\, \Sigma_R(j) \,\dot{\cup}\, \Sigma_A(j) \\
&= \Sigma - \dot{\cup}_{k\in\{1,\ldots,(j-1),(j+1),\ldots,n\}}\Sigma_{L_k}
\end{aligned}
$$

6.3 Controllability Properties

The goal in this chapter is to develop a means of verifying that our system's *flat supervisor* is controllable for the *flat plant* that uses only local checks. To achieve this, we will extend the *serial level-wise*

controllability definition to the parallel system case by using the *serial system extraction* definition.

Level-wise Controllable: The n^{th} degree ($n \geq 1$) *parallel interface system* composed of DES $\mathcal{G}_H$, $\mathcal{G}_{L_1}$, $\ldots, \mathcal{G}_{L_n}$, $\mathcal{S}_H$, $\mathcal{S}_{L_1}, \ldots, \mathcal{S}_{L_n}$, $G_{I_1}, \ldots, G_{I_n}$, is *level-wise controllable* with respect to alphabet partition $\Sigma := \dot{\cup}_{k\in\{1,\ldots,n\}}(\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, if:

$(\forall j \in \{1, \ldots, n\})$ The j^{th} *serial system extraction* of the system is *serial level-wise controllability.*

Now that we have the above definitions, we can present several related propositions that establish properties about the parallel system that will be useful in later proofs.

Our first proposition uses the *level-wise controllable* definition to establish the event set that the DES that make up an n^{th} degree ($n \geq 1$) *parallel interface system* are defined over. This is useful for defining the languages of a DES created by the synchronous product of one or more of these DES.

Proposition 28 *If n^{th} degree ($n \geq 1$)* parallel interface system *composed of DES $\mathcal{G}_H$, $\mathcal{G}_{L_1}, \ldots, \mathcal{G}_{L_n}$, $\mathcal{S}_H$, $\mathcal{S}_{L_1}, \ldots, \mathcal{S}_{L_n}$, $G_{I_1}, \ldots, G_{I_n}$, is* level-wise controllable *with respect to the alphabet partition $\Sigma := \dot{\cup}_{k\in\{1,\ldots,n\}}(\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$ then DES $\mathcal{G}_H$ and $\mathcal{S}_H$ are defined over event set Σ_{IH}, DES G_{I_j} is defined over event set Σ_{I_j}, and DES $\mathcal{G}_{L_j}$ and $\mathcal{S}_{L_j}$ are defined over event set Σ_{IL_j}, where $j \in \{1, \ldots, n\}$.*

Proof: See page 94

We are now ready to state the proposition below which establishes useful properties for often used languages.

Proposition 29 *If n^{th} degree ($n \geq 1$)* parallel interface system *composed of DES $\mathcal{G}_H$, $\mathcal{G}_{L_1}, \ldots, \mathcal{G}_{L_n}$, $\mathcal{S}_H$, $\mathcal{S}_{L_1}, \ldots, \mathcal{S}_{L_n}$, $G_{I_1}, \ldots, G_{I_n}$, is* level-wise controllable *with respect to the alphabet partition $\Sigma := \dot{\cup}_{k\in\{1,\ldots,n\}}(\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$ then, for all $j \in \{1, \ldots, n\}$, languages $\mathbf{H}$, $\mathbf{H}_{\mathcal{S}}$, $\mathbf{L}_j$, $\mathbf{L}_{\mathcal{S}_j}$, $\mathcal{I}_j$, $L(\mathbf{Plant})$, and $L(\mathbf{Sup})$ are closed.*

Proof: See page 96.

We now present a proposition that will aid in the use of the general form of *serial system extractions* in proofs. The proposition will interpret terminology for the j^{th} *serial system extraction* (a *serial interface system*) in terms of the original parallel system.

Before we can present the proposition, we need to first define (for use in the proposition) a new natural projection, P_j, to map strings from Σ^* (the event set of a given parallel system) to strings from $\Sigma(j)^*$ (the event set of the j^{th} extracted system of the given parallel system). It is as defined as follows:

$$P_j : \Sigma^* \rightarrow \Sigma(j)^*$$

Proposition 30 *If the* n^{th} *degree* ($n \geq 1$) parallel interface system *composed of DES* $\mathcal{G}_H, \mathcal{G}_{L_1}, \ldots, \mathcal{G}_{L_n}, \mathcal{S}_H, \mathcal{S}_{L_1}, \ldots, \mathcal{S}_{L_n}, G_{I_1}, \ldots, G_{I_n}$, *is* level-wise controllable *with respect to the alphabet partition* $\Sigma := \dot{\cup}_{k\in\{1,\ldots,n\}}(\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, *then for the* j^{th} serial system extraction, system(j), *the following is true:*

(i) The flat plant *is* $\mathbf{Plant}(j) = \mathcal{G}_H||_s G_{I_1}||_s \ldots ||_s G_{I_{(j-1)}}||_s G_{I_{(j+1)}}||_s \ldots ||_s G_{I_n}||_s \mathcal{G}_{L_j}$ *and the* flat supervisor *is* $\mathbf{Sup}(j) = \mathcal{S}_H||_s \mathcal{S}_{L_j}||_s G_{I_j}$

(ii) The following event sets are: $\Sigma_I(j) = \Sigma_{I_j}$, $\Sigma_{IH}(j) = \Sigma_{IH}$, *and* $\Sigma_{IL}(j) = \Sigma_{IL_j}$

(iii) The following inverse natural projections are: $P_{IH}(j)^{-1} = P_j \cdot P_{IH}^{-1}$, $P_{IL}(j)^{-1} = P_j \cdot P_{IL_j}^{-1}$, *and* $P_I(j)^{-1} = P_j \cdot P_{I_j}^{-1}$

(iv) The alphabet of $\mathcal{G}_H(j)$ *and* $\mathcal{S}_H(j)$ *is* $\Sigma_{IH}(j)$, *the alphabet of* $\mathcal{G}_L(j)$ *and* $\mathcal{S}_L(j)$ *is* $\Sigma_{IL}(j)$, *and the alphabet of* $G_I(j)$ *is* $\Sigma_I(j)$

(v) The following languages are:

$$\begin{aligned}
\mathbf{H}(j) &= P_j(\mathbf{H}) \cap [\cap_{k\in\{1,\ldots,(j-1),(j+1),\ldots,n\}} P_j(\mathcal{I}_k)] \\
\mathbf{H}_{\mathcal{S}}(j) &= P_j(\mathbf{H}_{\mathcal{S}}) \\
\mathbf{L}(j) &= P_j(\mathbf{L}_j) \\
\mathbf{L}_{\mathcal{S}}(j) &= P_j(\mathbf{L}_{\mathcal{S}_j}) \\
\mathcal{I}(j) &= P_j(\mathcal{I}_j) \\
L(\mathbf{Plant}(j)) &:= P_j(\mathbf{H}) \cap [\cap_{k\in\{1,\ldots,(j-1),(j+1),\ldots,n\}} P_j(\mathcal{I}_k)] \cap P_j(\mathbf{L}_j) \\
L(\mathbf{Sup}(j)) &:= P_j(\mathbf{H}_{\mathcal{S}}) \cap P_j(\mathbf{L}_{\mathcal{S}_j}) \cap P_j(\mathcal{I}_j)
\end{aligned}$$

(vi) Languages $\mathbf{H}(j)$, $\mathbf{H}_{\mathcal{S}}(j)$, $\mathbf{L}(j)$, $\mathbf{L}_{\mathcal{S}}(j)$, $\mathcal{I}(j)$, $L(\mathbf{Plant})(j)$, *and* $L(\mathbf{Sup})(j)$ *are closed.*

Proof: See page 97.

We close this section by noting that after examining the definition of the j^{th} *serial system extraction: general form* and the above proposition, we see that for $n = 1$, a general form *parallel interface system* reduces to a single general form *serial interface system.* We thus see that a *serial interface system* is a special case of a *parallel interface systems.* We will now talk of *parallel interface systems* and their definitions as the general case for bi-level interface systems.

6.4 Theorem and Propositions

We are now ready to present our main results for this chapter. We will first present two supporting propositions, followed by our parallel case controllability theorem. The following propositions are analogous to the serial controllability propositions.

6.4.1 Parallel Low level Controllability Proposition

We start with the parallel low level controllability proposition. It asserts that if the system is *level-wise controllable*, then each pair of low level supervisor and *interface* is controllable for the *flat plant.*

Proposition 31 *If the* n^{th} *degree* ($n \geq 1$) parallel interface system *composed of plant components* $\mathcal{G}_H, \mathcal{G}_{L_1}, \ldots, \mathcal{G}_{L_n}$, *supervisors* $\mathcal{S}_H, \mathcal{S}_{L_1}, \ldots, \mathcal{S}_{L_n}$, *and interfaces* $G_{I_1}, \ldots, G_{I_n}$, *is* level-wise controllable *with respect to the alphabet partition* $\Sigma := \dot{\cup}_{k\in\{1,\ldots,n\}}(\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, *then*

$$(\forall j \in \{1, \ldots, n\})\,(\forall s \in L(\mathbf{Plant}) \cap \mathbf{L}_{\mathcal{S}_j} \cap \mathcal{I}_j) \quad \mathit{Elig}_{L(\mathbf{Plant})}(s) \cap \Sigma_u \subseteq \mathit{Elig}_{\mathbf{L}_{\mathcal{S}_j}\cap\mathcal{I}_j}(s)$$

where $\mathbf{Plant} := \mathcal{G}_H||_s\mathcal{G}_{L_1}||_s \ldots ||_s\mathcal{G}_{L_n}$ *is the system's* flat plant.

Proof: See page 99.

6.4.2 Parallel High level Controllability Proposition

We now present the parallel high level controllability proposition. It asserts that if the system is *level-wise controllable*, then $\mathcal{S}_H$ is controllable for *flat plant* when the *flat Plant* is already under the control of the *interfaces.*

Proposition 32 *If the n^{th} degree ($n \geq 1$)* parallel interface system *composed of plant components $\mathcal{G}_H, \mathcal{G}_{L_1}, \ldots, \mathcal{G}_{L_n}$, supervisors $\mathcal{S}_H, \mathcal{S}_{L_1}, \ldots, \mathcal{S}_{L_n}$, and interfaces $G_{I_1}, \ldots, G_{I_n}$, is* level-wise controllable *with respect to the alphabet partition $\Sigma := \dot{\cup}_{k\in\{1,\ldots,n\}}(\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, then*

$$(\forall s \in L(\mathbf{Plant}) \cap \mathbf{H}_{\mathcal{S}} \cap [\cap_{k\in\{1,\ldots,n\}} \mathcal{I}_k]) \quad Elig_{L(\mathbf{Plant})\cap[\cap_{k\in\{1,\ldots,n\}}\mathcal{I}_k]}(s) \cap \Sigma_u \subseteq Elig_{\mathbf{H}_{\mathcal{S}}}(s)$$

where $\mathbf{Plant} := \mathcal{G}_H||_s\mathcal{G}_{L_1}||_s \ldots ||_s\mathcal{G}_{L_n}$ *is the system's* flat plant.

Proof: See page 101.

6.4.3 Parallel Controllability Theorem

We now present our main result for this chapter, the parallel controllability theorem. It states that, to verify if a parallel system is controllable, it is sufficient to check that each of its *serial system extractions* is *serial level-wise controllable.* As the *level-wise controllable* definition can be evaluated by examining only one level of our system at a time (the *high level* or one of the *low levels*), we now have a means to verify controllability of our system using local checks.

Theorem 4 *If the n^{th} degree ($n \geq 1$)* parallel interface system *composed of plant components $\mathcal{G}_H$, $\mathcal{G}_{L_1}, \ldots, \mathcal{G}_{L_n}$, supervisors $\mathcal{S}_H, \mathcal{S}_{L_1}, \ldots, \mathcal{S}_{L_n}$, and interfaces $G_{I_1}, \ldots, G_{I_n}$, is* level-wise controllable *with respect to the alphabet partition $\Sigma := \dot{\cup}_{k\in\{1,\ldots,n\}}(\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, then*

$$(\forall s \in L(\mathbf{Plant}) \cap L(\mathbf{Sup})) \quad Elig_{L(\mathbf{Plant})}(s) \cap \Sigma_u \subseteq Elig_{L(\mathbf{Sup})}(s)$$

where $\mathbf{Plant} := \mathcal{G}_H||_s\mathcal{G}_{L_1}||_s \ldots ||_s\mathcal{G}_{L_n}$ *is the system's* flat plant, *and* $\mathbf{Sup} := \mathcal{S}_H||_s\mathcal{S}_{L_1}||_s \ldots ||_s\mathcal{S}_{L_n}||_s G_{I_1}||_s \ldots ||_s G_{I_n}$ *is the system's* flat supervisor.

Proof:

Assume that the n^{th} degree ($n \geq 1$) *parallel interface system* is *level-wise controllable.* **(1)**

Let $s \in L(\mathbf{Plant}) \cap L(\mathbf{Sup})$, and $\sigma \in \text{Elig}_{L(\mathbf{Plant})}(s) \cap \Sigma_u$ **(2)**

We will now show this implies $\sigma \in \text{Elig}_{L(\mathbf{Sup})}(s)$. It's sufficient to show $s\sigma \in L(\mathbf{Sup}) = \mathbf{H}_{\mathcal{S}} \cap [\cap_{k\in\{1,\ldots,n\}}(\mathbf{L}_{\mathcal{S}_k} \cap \mathcal{I}_k)]$

We first note that $s, s\sigma \in \mathbf{H} \cap [\cap_{k\in\{1,\ldots,n\}}\mathbf{L}_k] = L(\mathbf{Plant})$ and $s \in \mathbf{H}_{\mathcal{S}} \cap [\cap_{k\in\{1,\ldots,n\}}(\mathbf{L}_{\mathcal{S}_k} \cap \mathcal{I}_k)]$ by **(2)**. **(3)**

We next note that, by **(1)**, we can apply **Proposition 31** and conclude:

$$s\sigma \in \cap_{k\in\{1,\ldots,n\}}(\mathbf{L}_{\mathcal{S}_k} \cap \mathcal{I}_k) \qquad \textbf{(4)}$$

All that remains is to show that $s\sigma \in \mathbf{H}_{\mathcal{S}}$

From **(3)**, we have $s \in L(\mathbf{Plant}) \cap \mathbf{H}_{\mathcal{S}} \cap [\cap_{k\in\{1,\ldots,n\}}\mathcal{I}_k]$

From **(3)** and**(4)**, we have $s\sigma \in L(\mathbf{Plant}) \cap [\cap_{k\in\{1,\ldots,n\}}\mathcal{I}_k]$

$\Rightarrow \sigma \in \text{Elig}_{L(\mathbf{Plant})\cap[\cap_{k\in\{1,\ldots,n\}}\mathcal{I}_k]}(s) \cap \Sigma_u$

We can now apply **Proposition 32**, and conclude $\sigma \in \text{Elig}_{\mathbf{H}_{\mathcal{S}}}(s)$

$\Rightarrow s\sigma \in \mathbf{H}_{\mathcal{S}}$

From **(4)**, we can now conclude $s\sigma \in \mathbf{H}_{\mathcal{S}} \cap [\cap_{k\in\{1,\ldots,n\}}(\mathbf{L}_{\mathcal{S}_k} \cap \mathcal{I}_k)]$, as required

QED

6.5 Proofs of Selected Propositions

In order to make this work more readable, the proofs of some propositions in this chapter were not given as the propositions were introduced. They are now presented in the following sections.

6.5.1 Proof of Proposition 28

Proof for **Proposition 28** on page 90: *If n^{th} degree ($n \geq 1$)* parallel interface system *composed of DES $\mathcal{G}_H$, $\mathcal{G}_{L_1}, \ldots, \mathcal{G}_{L_n}$, $\mathcal{S}_H$, $\mathcal{S}_{L_1}, \ldots, \mathcal{S}_{L_n}$, $G_{I_1}, \ldots, G_{I_n}$, is* level-wise controllable *with respect to the alphabet partition*
$\Sigma := \dot\cup_{k\in\{1,\ldots,n\}}(\Sigma_{L_k}\dot\cup\Sigma_{R_k}\dot\cup\Sigma_{A_k}) \dot\cup \Sigma_H$ *then DES $\mathcal{G}_H$ and $\mathcal{S}_H$ are defined over event set Σ_{IH}, DES G_{I_j} is defined over event set Σ_{I_j}, and DES $\mathcal{G}_{L_j}$ and $\mathcal{S}_{L_j}$ are defined over event set Σ_{IL_j}, where $j \in \{1, \ldots, n\}$.*

Proof:

Assume that the n^{th} degree ($n \geq 1$) *parallel interface system* is *level-wise controllable* with respect to the alphabet partition. **(1)**

We will now show this implies that DES $\mathcal{G}_H$ and $\mathcal{S}_H$ are defined over event set Σ_{IH}, DES G_{I_j} is defined

over event set Σ_{I_j}, and DES $\mathcal{G}_{L_j}$ and $\mathcal{S}_{L_j}$ are defined over event set Σ_{IL_j}, where $j \in \{1, \ldots, n\}$.

We first note that **(1)** implies that $(\forall j \in \{1, \ldots, n\})$ the j^{th} *serial system extraction*, labelled *system*(j), of the system is *serial level-wise controllable.*

This allows us to conclude that: $(\forall j \in \{1, \ldots, n\})$ DES $\mathcal{G}_H(j)$ and $\mathcal{S}_H(j)$ are defined over $\Sigma_{IH}(j)$, $\mathcal{G}_L(j)$ and $\mathcal{S}_L(j)$ are defined over $\Sigma_{IL}(j)$, and that $G_I(j)$ is defined over $\Sigma_I(j)$.

From **(1)**, we can now apply **Proposition 30**, **point ii** and conclude:[1]

$$\Sigma_I(j) = \Sigma_{I_j},\ \Sigma_{IH}(j) = \Sigma_{IH}, \text{ and } \Sigma_{IL}(j) = \Sigma_{IL_j}$$

We can now conclude that: $(\forall j \in \{1, \ldots, n\})$ DES $\mathcal{G}_H(j)$ and $\mathcal{S}_H(j)$ are defined over Σ_{IH}, $\mathcal{G}_L(j)$ and $\mathcal{S}_L(j)$ are defined over Σ_{IL_j}, and that $G_I(j)$ is defined over Σ_{I_j}. **(2)**

This implies: $(\forall j \in \{1, \ldots, n\})$ DES $\mathcal{S}_H = \mathcal{S}_H(j)$ is defined over Σ_{IH}, $\mathcal{G}_{L_j} = \mathcal{G}_L(j)$ and $\mathcal{S}_{L_j} = \mathcal{S}_L(j)$ are defined over Σ_{IL_j} and that $G_{I_j} = G_I(j)$ is defined over Σ_{I_j}. **(3)**

All that remains is to show that DES $\mathcal{G}_H$ is defined over alphabet Σ_{IH}. To do this, we first need to prove the following claim.

Claim: $\Sigma_{\mathcal{G}_H} \subseteq \Sigma_{IH}$ and $(\forall j \in \{1, \ldots, n\})\ \Sigma_{\mathcal{G}_H} \supseteq (\Sigma_H \cup \Sigma_{I_j})$

Let $j \in \{1, \ldots, n\}$. We will now show this implies $\Sigma_{\mathcal{G}_H} \subseteq \Sigma_{IH}$ and $\Sigma_{\mathcal{G}_H} \supseteq (\Sigma_H \cup \Sigma_{I_j})$.

We start by noting $\mathcal{G}_H(j) := \mathcal{G}_H||_s G_{I_1}||_s \ldots ||_s G_{I_{(j-1)}}||_s G_{I_{(j+1)}}||_s \ldots ||_s G_{I_n}$. By the definition of the $||_s$ operator, we know $\Sigma_{\mathcal{G}_H(j)} = \Sigma_{\mathcal{G}_H} \cup [\cup_{k \in \{1, \ldots, (j-1), (j+1), \ldots, n\}} \Sigma_{G_{I_k}}]$. This implies that $\Sigma_{\mathcal{G}_H} \subseteq \Sigma_{\mathcal{G}_H(j)}$. As $\Sigma_{\mathcal{G}_H(j)} = \Sigma_{IH}$ (from **(2)**), we immediately have $\Sigma_{\mathcal{G}_H} \subseteq \Sigma_{IH}$.

From **(3)**, we have $\Sigma_{\mathcal{G}_H(j)} = \Sigma_{\mathcal{G}_H} \cup [\cup_{k \in \{1, \ldots, (j-1), (j+1), \ldots, n\}} \Sigma_{I_k}]$.

We now note that $\Sigma_{I_j} \subseteq \Sigma_{IH}$ but $\Sigma_{I_j} \cap [\cup_{k \in \{1, \ldots, (j-1), (j+1), \ldots, n\}} \Sigma_{I_k}] = \emptyset$ because of our event partition.

This implies: $\Sigma_{I_j} \subseteq \Sigma_{\mathcal{G}_H}$

We next note that $\Sigma_H \subseteq \Sigma_{IH}$ but $\Sigma_H \cap [\cup_{k \in \{1, \ldots, (j-1), (j+1), \ldots, n\}} \Sigma_{I_k}] = \emptyset$ because of our event partition.

This implies: $\Sigma_H \subseteq \Sigma_{\mathcal{G}_H}$

We thus have $\Sigma_{\mathcal{G}_H} \supseteq (\Sigma_H \cup \Sigma_{I_j})$ as required.

Claim proven.

From the claim, we have $\Sigma_{\mathcal{G}_H} \subseteq \Sigma_{IH}$. To show that $\Sigma_{\mathcal{G}_H} = \Sigma_{IH}$ we now only have to show $\Sigma_{\mathcal{G}_H} \supseteq \Sigma_{IH}$.

[1] **Proposition 30**, **point iv** requires the proposition we are currently proving, but **point ii** of **Proposition 30** is independent of **point iv** and does not.

From the claim, we also have $(\forall j \in \{1, \ldots, n\})\ \Sigma_{\mathcal{G}_H} \supseteq (\Sigma_H \cup \Sigma_{I_j})$.

This implies $\Sigma_{\mathcal{G}_H} \supseteq \Sigma_H \cup [\cup_{k\in\{1,\ldots,n\}}\Sigma_{I_k}] = \Sigma_{IH}$. We thus have $\Sigma_{\mathcal{G}_H} = \Sigma_{IH}$.

We can now conclude that DES $\mathcal{G}_H$ is defined over Σ_{IH}, as required.

QED

6.5.2 Proof of Proposition 29

Proof for **Proposition 29** on page 90: If n^{th} degree ($n \geq 1$) *parallel interface system* composed of DES $\mathcal{G}_H$, $\mathcal{G}_{L_1}, \ldots,\ \mathcal{G}_{L_n}$, $\mathcal{S}_H$, $\mathcal{S}_{L_1},\ \ldots, \mathcal{S}_{L_n}$, $G_{I_1}, \ldots, G_{I_n}$, is *level-wise controllable* with respect to the alphabet partition $\Sigma := \dot\cup_{k\in\{1,\ldots,n\}}(\Sigma_{L_k} \dot\cup \Sigma_{R_k} \dot\cup \Sigma_{A_k}) \dot\cup \Sigma_H$ then, for all $j \in \{1, \ldots, n\}$, languages $\mathbf{H}$, $\mathbf{H}_\mathcal{S}$, $\mathbf{L}_j$, $\mathbf{L}_{\mathcal{S}_j}$, $\mathcal{I}_j$, $L(\mathbf{Plant})$, and $L(\mathbf{Sup})$ are closed.

Proof:

Assume system is *level-wise controllable.* Let $j \in \{1, \ldots, n\}$. **(1)**

Will now show this implies that the indicated languages are closed.

We first note that by **(1)**, the system is *level-wise controllable.* This allows us to apply **Proposition 28** and conclude:

$$L(\mathcal{G}_H), L(\mathcal{S}_H) \subseteq \Sigma^*_{IH},\ L(\mathcal{G}_{L_j}), L(\mathcal{S}_{L_j}) \subseteq \Sigma^*_{IL_j}, \text{ and } L(G_{I_j}) \subseteq \Sigma^*_{I_j}.$$

This tells us that languages $\mathbf{H} = P^{-1}_{IH}(L(\mathcal{G}_H))$, $\mathbf{H}_\mathcal{S} = P^{-1}_{IH}L(\mathcal{S}_H)$, $\mathbf{L}_j = P^{-1}_{IL}(L(\mathcal{G}_{L_j}))$, $\mathbf{L}_{\mathcal{S}_j} = P^{-1}_{IL}L(\mathcal{S}_{L_j})$, $\mathcal{I}_j = P^{-1}_I(L(G_{I_j}))$ are defined.

We will start by showing that languages $\mathbf{H}$, $\mathbf{H}_\mathcal{S}$, $\mathbf{L}_j$, $\mathbf{L}_{\mathcal{S}_j}$, and $\mathcal{I}_j$ are closed.

We now note that languages $L(\mathcal{G}_H)$, $L(\mathcal{S}_H)$, $L(\mathcal{G}_{L_j})$, $L(\mathcal{S}_{L_j})$, and $L(G_{I_j})$ are closed by the definition of the closed behaviour of a DES.

We can now apply **Proposition 1** repeatedly and conclude that $\mathbf{H}$, $\mathbf{H}_\mathcal{S}$, $\mathbf{L}_j$, $\mathbf{L}_{\mathcal{S}_j}$, and $\mathcal{I}_j$ are closed, as required. **(2)**

We will now show that languages $L(\mathbf{Plant})$, and $L(\mathbf{Sup})$ are closed.

We next note that $L(\mathbf{Plant}) = \mathbf{H} \cap [\cap_{k\in\{1,\ldots,n\}}\mathbf{L}_k]$ and $L(\mathbf{Sup}) = \mathbf{H}_\mathcal{S} \cap [\cap_{k\in\{1,\ldots,n\}}(\mathbf{L}_{\mathcal{S}_k} \cap \mathcal{I}_k)]$.

Combining with **(2)**, we can now apply **Proposition 2** repeatedly and conclude that $L(\mathbf{Plant})$, and $L(\mathbf{Sup})$ are closed, as required.

QED

6.5.3 Proof of Proposition 30

Proof for **Proposition 30** on page 91: *If the* n^{th} *degree* ($n \geq 1$) parallel interface system *composed of DES* $\mathcal{G}_H, \mathcal{G}_{L_1}, \ldots, \mathcal{G}_{L_n}, \mathcal{S}_H, \mathcal{S}_{L_1}, \ldots, \mathcal{S}_{L_n}, G_{I_1}, \ldots, G_{I_n}$, *is* level-wise controllable *with respect to the alphabet partition* $\Sigma := \dot{\cup}_{k\in\{1,\ldots,n\}}(\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, *then for the* j^{th} serial system extraction, system(j), *the following is true:*

(i) The flat plant *is* $\mathbf{Plant}(j) = \mathcal{G}_H||_s G_{I_1}||_s \ldots ||_s G_{I_{(j-1)}}||_s G_{I_{(j+1)}}||_s \ldots ||_s G_{I_n}||_s \mathcal{G}_{L_j}$ *and the* flat supervisor *is* $\mathbf{Sup}(j) = \mathcal{S}_H||_s \mathcal{S}_{L_j}||_s G_{I_j}$

(ii) The following event sets are: $\Sigma_I(j) = \Sigma_{I_j}$, $\Sigma_{IH}(j) = \Sigma_{IH}$, *and* $\Sigma_{IL}(j) = \Sigma_{IL_j}$

(iii) The following inverse natural projections are: $P_{IH}(j)^{-1} = P_j \cdot P_{IH}^{-1}$, $P_{IL}(j)^{-1} = P_j \cdot P_{IL_j}^{-1}$, *and* $P_I(j)^{-1} = P_j \cdot P_{I_j}^{-1}$

(iv) The alphabet of $\mathcal{G}_H(j)$ *and* $\mathcal{S}_H(j)$ *is* $\Sigma_{IH}(j)$, *the alphabet of* $\mathcal{G}_L(j)$ *and* $\mathcal{S}_L(j)$ *is* $\Sigma_{IL}(j)$, *and the alphabet of* $G_I(j)$ *is* $\Sigma_I(j)$

(v) The following languages are:

$$
\begin{aligned}
\mathbf{H}(j) &= P_j(\mathbf{H}) \cap [\cap_{k\in\{1,\ldots,(j-1),(j+1),\ldots,n\}} P_j(\mathcal{I}_k)] \\
\mathbf{H}_{\mathcal{S}}(j) &= P_j(\mathbf{H}_{\mathcal{S}}) \\
\mathbf{L}(j) &= P_j(\mathbf{L}_j) \\
\mathbf{L}_{\mathcal{S}}(j) &= P_j(\mathbf{L}_{\mathcal{S}_j}) \\
\mathcal{I}(j) &= P_j(\mathcal{I}_j) \\
L(\mathbf{Plant}(j)) &:= P_j(\mathbf{H}) \cap [\cap_{k\in\{1,\ldots,(j-1),(j+1),\ldots,n\}} P_j(\mathcal{I}_k)] \cap P_j(\mathbf{L}_j) \\
L(\mathbf{Sup}(j)) &:= P_j(\mathbf{H}_{\mathcal{S}}) \cap P_j(\mathbf{L}_{\mathcal{S}_j}) \cap P_j(\mathcal{I}_j)
\end{aligned}
$$

(vi) Languages $\mathbf{H}(j)$, $\mathbf{H}_{\mathcal{S}}(j)$, $\mathbf{L}(j)$, $\mathbf{L}_{\mathcal{S}}(j)$, $\mathcal{I}(j)$, $L(\mathbf{Plant})(j)$, *and* $L(\mathbf{Sup})(j)$ *are closed.*

Proof:

Assume that the n^{th} degree ($n \geq 1$) *parallel interface system* is *level-wise controllable* with respect to the alphabet partition. **(1)**

Let *system*(j) be the j^{th} *serial system extraction* of our parallel system. **(2)**

We will now show this implies *system*(j) satisfies **points i-vi.**

Point i: Show that the *flat plant* is $\mathbf{Plant}(j) = \mathcal{G}_H||_s G_{I_1}||_s \ldots ||_s G_{I_{(j-1)}}||_s G_{I_{(j+1)}}||_s \ldots ||_s G_{I_n}||_s \mathcal{G}_{L_j}$ and the *flat supervisor* is $\mathbf{Sup}(j) = \mathcal{S}_H||_s \mathcal{S}_{L_j}||_s G_{I_j}$

$$\begin{aligned}
\mathbf{Plant}(j) &:= \mathcal{G}_H(j)||_s \mathcal{G}_L(j), \text{ by definition.} \\
&= \mathcal{G}_H||_s G_{I_1}||_s \ldots ||_s G_{I_{(j-1)}}||_s G_{I_{(j+1)}}||_s \ldots ||_s G_{I_n}||_s \mathcal{G}_{L_j}, \text{ by } \mathbf{(2)}. \\
\mathbf{Sup}(j) &:= \mathcal{S}_H(j)||_s \mathcal{S}_L(j)||_s G_I(j), \text{ by definition.} \\
&= \mathcal{S}_H||_s \mathcal{S}_{L_j}||_s G_{I_j}, \text{ by } \mathbf{(2)}.
\end{aligned}$$

Point ii: Show that the following event sets are: $\Sigma_I(j) = \Sigma_{I_j}$, $\Sigma_{IH}(j) = \Sigma_{IH}$, and $\Sigma_{IL}(j) = \Sigma_{IL_j}$

Proof is identical to the proof of **point ii** of **Proposition 23**.

Point iii: Show that $P_{IH}(j)^{-1} = P_j \cdot P_{IH}^{-1}$, $P_{IL}(j)^{-1} = P_j \cdot P_{IL_j}^{-1}$, and $\Sigma_I(j) = P_j \cdot P_{I_j}^{-1}$

Proof is identical to the proof of **point iii** of **Proposition 23**.

Point iv: Show that the alphabet of $\mathcal{G}_H(j)$ and $\mathcal{S}_H(j)$ is $\Sigma_{IH}(j)$, the alphabet of $\mathcal{G}_L(j)$ and $\mathcal{S}_L(j)$ is $\Sigma_{IL}(j)$, and
the alphabet of $G_I(j)$ is $\Sigma_I(j)$

From **(1)**, we have that the system is *level-wise controllable*. This implies that *system*(j) is *serial level-wise controllable*. The result follows immediately.

Point v:

First, we must show that $\mathbf{H}(j) = P_j(\mathbf{H}) \cap [\cap_{k \in \{1,\ldots,(j-1),(j+1),\ldots,n\}} P_j(\mathcal{I}_k)]$

Proof is identical to the proof for $\mathcal{H}(j)$ of **point v** of **Proposition 23** after relabelling and substituting **Proposition 28** for **Proposition 21**. **(3)**

The proofs for the remaining languages for **point v** are straightforward, and are presented together below.

$$\begin{aligned}
\mathbf{H}_{\mathcal{S}}(j) &:= P_{IH}(j)^{-1} L(\mathcal{S}_H(j)), \text{ by definition.} \\
&= P_j \cdot P_{IH}^{-1}(L(\mathcal{S}_H)), \text{ by } \mathbf{(2)} \text{ and } \mathbf{point\ iii}. \\
&= P_j(\mathbf{H}_{\mathcal{S}}) \qquad (4) \\
\mathbf{L}(j) &:= P_{IL}(j)^{-1}(L(\mathcal{G}_L(j))), \text{ by definition.} \\
&= P_j \cdot P_{IL_j}^{-1}(L(\mathcal{G}_{L_j})), \text{ by } \mathbf{(2)} \text{ and } \mathbf{point\ iii}. \\
&= P_j(\mathbf{L}_j) \qquad (5) \\
\mathbf{L}_{\mathcal{S}}(j) &:= P_{IL}(j)^{-1}(L(\mathcal{S}_L(j))), \text{ by definition.}
\end{aligned}$$

$$
\begin{aligned}
&= P_j \cdot P_{IL_j}^{-1}(L(\mathcal{S}_{L_j})), \text{ by } \textbf{(2)} \text{ and } \textbf{point iii}. \\
&= P_j\left(\mathbf{L}_{\mathcal{S}_j}\right) && (\textbf{6}) \\
\mathcal{I}(j) &:= P_I(j)^{-1}(L(G_I(j))), \text{ by definition}. \\
&= P_j \cdot P_{I_j}^{-1}(L(G_{I_j})), \text{ by } \textbf{(2)} \text{ and } \textbf{point iii}. \\
&= P_j\left(\mathcal{I}_j\right) && (\textbf{7}) \\
L(\textbf{Plant}(j)) &:= \mathbf{H}(j) \cap \mathbf{L}(j), \text{ by definition}. \\
&= P_j\left(\mathbf{H}\right) \cap \left[\cap_{k\in\{1,\ldots,(j-1),(j+1),\ldots,n\}} P_j\left(\mathcal{I}_k\right)\right] \cap P_j\left(\mathbf{L}_j\right), \text{ by } \textbf{(3)} \text{ and } \textbf{(5)}. \\
L(\textbf{Sup}(j)) &:= \mathbf{H}_{\mathcal{S}}(j) \cap \mathbf{L}_{\mathcal{S}}(j) \cap \mathcal{I}(j), \text{ by definition}. \\
&= P_j(\mathbf{H}_{\mathcal{S}}) \cap P_j\left(\mathbf{L}_{\mathcal{S}_j}\right) \cap P_j\left(\mathcal{I}_j\right), \text{ by } \textbf{(4)}, \textbf{(6)}, \text{ and } \textbf{(7)}.
\end{aligned}
$$

Point vi: Show that the languages $\mathbf{H}(j)$, $\mathbf{H}_{\mathcal{S}}(j)$, $\mathbf{L}(j)$, $\mathbf{L}_{\mathcal{S}}(j)$, $\mathcal{I}(j)$, $L(\textbf{Plant})(j)$, and $L(\textbf{Sup})(j)$ are closed.

From **(2)**, we know that $\mathcal{G}_H(j) = \mathcal{G}_H||_s G_{I_1}||_s \ldots ||_s G_{I_{(j-1)}}||_s G_{I_{(j+1)}}||_s \ldots ||_s G_{I_n}$, $\textbf{Plant} = \mathcal{G}_H||_s G_{I_1} ||_s \ldots ||_s G_{I_{(j-1)}}||_s G_{I_{(j+1)}}||_s \ldots ||_s G_{I_n}||_s \mathcal{G}_{L_j}$, and $\textbf{Sup} = \mathcal{S}_H||_s \mathcal{S}_{L_j}||_s G_{I_j}$.

We can now apply **Proposition 5** and conclude that languages $L(\mathcal{G}_H(j))$, $L(\textbf{Plant})$, and $L(\textbf{Sup})$ is closed.

We next note that languages $\mathcal{S}_H(j)$, $\mathcal{G}_L(j)$, $\mathcal{S}_L(j)$, and $G_I(j)$ are closed as $\mathcal{S}_H(j) = \mathcal{S}_H$, $\mathcal{G}_L(j) = \mathcal{G}_{L_j}$, $\mathcal{S}_L(j) = \mathcal{S}_{L_j}$, and $G_I(j) = G_{I_j}$ (by **(2)**), and by the definition of the closed behaviour of a DES.

We now apply **Proposition 1** repeatedly and conclude that $\mathbf{H}(j) = P_{IH}(j)^{-1}L(\mathcal{G}_H(j))$, $\mathbf{H}_{\mathcal{S}}(j) = P_{IH}(j)^{-1}L(\mathcal{S}_H(j))$, $\mathbf{L}(j) = P_{IL}(j)^{-1}L(\mathcal{G}_L(j))$, $\mathbf{L}_{\mathcal{S}}(j) = P_{IL}(j)^{-1}L(\mathcal{S}_L(j))$, and $\mathcal{I}(j) = P_I(j)^{-1}L(G_I(j))$ are closed.

QED

6.5.4 Proof of Proposition 31

Proof for **Proposition 31** on page 92: *If the* n^{th} *degree* ($n \geq 1$) parallel interface system *composed of plant components* $\mathcal{G}_H, \mathcal{G}_{L_1}, \ldots, \mathcal{G}_{L_n}$, *supervisors* $\mathcal{S}_H, \mathcal{S}_{L_1}, \ldots, \mathcal{S}_{L_n}$, *and interfaces* $G_{I_1}, \ldots, G_{I_n}$, *is* level-wise controllable *with respect to the alphabet partition* $\Sigma := \dot{\cup}_{k\in\{1,\ldots,n\}}(\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, *then*

$$
(\forall j \in \{1,\ldots,n\})\,(\forall s \in L(\textbf{Plant}) \cap \mathbf{L}_{\mathcal{S}_j} \cap \mathcal{I}_j) \quad \mathit{Elig}_{L(\textbf{Plant})}(s) \cap \Sigma_u \subseteq \mathit{Elig}_{\mathbf{L}_{\mathcal{S}_j}\cap\mathcal{I}_j}(s)
$$

where $\textbf{Plant} := \mathcal{G}_H||_s\mathcal{G}_{L_1}||_s \ldots ||_s\mathcal{G}_{L_n}$ *is the system's* flat plant.

Proof:

Assume that the n^{th} degree ($n \geq 1$) *parallel interface system* is *level-wise controllable.* **(1)**

Let $j \in \{1, \ldots, n\}$, $s \in L(\mathbf{Plant}) \cap \mathbf{L}_{\mathcal{S}_j} \cap \mathcal{I}_j$, and $\sigma \in \text{Elig}_{L(\mathbf{Plant})}(s) \cap \Sigma_u$ **(2)**

We will now show that this implies $\sigma \in \text{Elig}_{\mathbf{L}_{\mathcal{S}_j} \cap \mathcal{I}_j}(s)$

It's sufficient to show that $s\sigma \in \mathbf{L}_{\mathcal{S}_j} \cap \mathcal{I}_j$

We first note that $s, s\sigma \in \mathbf{H} \cap [\cap_{k \in \{1,\ldots,n\}} \mathbf{L}_k] = L(\mathbf{Plant})$ by **(2)**. **(3)**

We have two cases: **I)** $\sigma \notin \Sigma_{IL_j}$ and **II)** $\sigma \in \Sigma_{IL_j}$

case I)

Assume $\sigma \notin \Sigma_{IL_j}$. This implies: $P_{IL_j}(\sigma) = \epsilon$, where ϵ is the empty string.

$\Rightarrow$ $P_{IL_j}(s\sigma) = P_{IL_j}(s)P_{IL_j}(\sigma) = P_{IL_j}(s)$, as the natural projection is concatenative. Similarly, we have $P_{I_j}(s\sigma) = P_{I_j}(s)$ as $\sigma \notin \Sigma_{I_j}$ since $\Sigma_{I_j} \subseteq \Sigma_{IL_j}$ **(4)**

From **(2),** we have $s \in \mathbf{H}_{\mathcal{S}} \cap [\cap_{k \in \{1,\ldots,n\}} (\mathbf{L}_{\mathcal{S}_k} \cap \mathcal{I}_k)] = L(\mathbf{Sup})$

As $s \in \mathbf{L}_{\mathcal{S}_j}$ and $P_{IL_j}(s\sigma) = P_{IL_j}(s)$, we can apply **Proposition 27**, **point c**, and conclude $s\sigma \in \mathbf{L}_{\mathcal{S}_j}$

Similarly, we can apply **Proposition 20**, **point e**,and conclude $s\sigma \in \mathcal{I}_j$

We thus have $s\sigma \in \mathbf{L}_{\mathcal{S}_j} \cap \mathcal{I}_j$

Case I complete.

case II)

Assume $\sigma \in \Sigma_{IL_j}$.

We now examine *system*(j), the j^{th} *serial system extraction* of our parallel system.

We first note that we have $\sigma \in \Sigma_{IL}(j)$ as $\Sigma_{IL}(j) = \Sigma_{IL_j}$ by **Proposition 30**.

$\Rightarrow$ $\sigma \in \Sigma(j) \supseteq \Sigma_{IL}(j)$

$\Rightarrow P_j(\sigma) = \sigma$. See Section 6.3 for the definition of the natural projection P_j.

$\Rightarrow$ $P_j(s\sigma) = P_j(s)\sigma$

From **(1),** we can conclude that *system*(j) is *serial level-wise controllable.*

We will use **point II** of this definition to show that $P_j(s)\sigma \in \mathbf{L}_{\mathcal{S}}(j) \cap \mathcal{I}(j)$

To do this, we first need to show that $P_j(s), P_j(s)\sigma \in \mathbf{L}(j)$.

As $s, s\sigma \in \mathbf{H} \cap [\cap_{k\in\{1,\ldots,n\}}\mathbf{L}_k]$ by **(3),** we have $s, s\sigma \in \mathbf{L}_j$

$\Rightarrow$ $P_j(s) \in P_j\mathbf{L}_j$ and $P_j(s\sigma) = P_j(s)\sigma \in P_j\mathbf{L}_j$

$\Rightarrow$ $P_j(s), P_j(s)\sigma \in \mathbf{L}(j)$, by **Proposition 30**.

As we have $\sigma \in \Sigma_u$ from **(2),** we can conclude $\sigma \in \text{Elig}_{\mathbf{L}(j)}(P_j(s)) \cap \Sigma_u$

We now only need to show $P_j(s) \in \mathbf{L}_{\mathcal{S}}(j) \cap \mathcal{I}(j)$

From **(2),** we have $s \in L(\mathbf{Sup})$ and thus $s \in \mathbf{L}_{\mathcal{S}_j} \cap \mathcal{I}_j$.

$\Rightarrow$ $P_j(s) \in P_j\mathbf{L}_{\mathcal{S}_j} \cap P_j\mathcal{I}_j$

$\Rightarrow$ $P_j(s) \in \mathbf{L}_{\mathcal{S}}(j) \cap \mathcal{I}(j)$, by **Proposition 30**.

We now have $P_j(s) \in \mathbf{L}(j) \cap \mathbf{L}_{\mathcal{S}}(j) \cap \mathcal{I}(j)$ and $\sigma \in \text{Elig}_{\mathbf{L}(j)}(P_j(s)) \cap \Sigma_u$ and can conclude by **point II** of the *serial level-wise controllable* definition that:

$$\sigma \in \text{Elig}_{\mathbf{L}_{\mathcal{S}}(j)\cap\mathcal{I}(j)}(P_j(s)) \text{ and thus } P_j(s)\sigma = P_j(s\sigma) \in \mathbf{L}_{\mathcal{S}}(j) \cap \mathcal{I}(j)$$

Substituting in for $\mathbf{L}_{\mathcal{S}}(j)$ and $\mathcal{I}(j)$ (by **Proposition 30**) gives: $P_j(s\sigma) \in P_jP_{IL_j}^{-1}L(\mathcal{S}_{L_j}) \cap P_jP_{I_j}^{-1}L(G_{I_j})$

We note that since $\Sigma_{IL}(j) = \Sigma_{IL_j}$ and $\Sigma_I(j) = \Sigma_{I_j}$ we have $\Sigma_{IL_j} \subseteq \Sigma(j)$ and $\Sigma_{I_j} \subseteq \Sigma(j)$. We can thus apply **Corollary 2** twice, taking first $\Sigma_a = \Sigma(j)$, $\Sigma_b = \Sigma_{IL_j}$, and $L_b = L(\mathcal{S}_{L_j})$ and then $\Sigma_a = \Sigma(j)$, $\Sigma_b = \Sigma_{I_j}$, and $L_b = L(G_{I_j})$. We can thus conclude:

$$s\sigma \in P_{IL_j}^{-1}L(\mathcal{S}_{L_j}) \cap P_{I_j}^{-1}L(G_{I_j}) = \mathbf{L}_{\mathcal{S}_j} \cap \mathcal{I}_j$$

Case II complete.

By **Cases I** and **II**, we have $s\sigma \in \mathbf{L}_{\mathcal{S}_j} \cap \mathcal{I}_j$, as required.

QED

6.5.5 Proof of Proposition 32

Proof for **Proposition 32** on page 93: *If the* n^{th} *degree* ($n \geq 1$) parallel interface system *composed of plant components* $\mathcal{G}_H, \mathcal{G}_{L_1}, \ldots, \mathcal{G}_{L_n}$, *supervisors* $\mathcal{S}_H, \mathcal{S}_{L_1}, \ldots, \mathcal{S}_{L_n}$, *and interfaces* $G_{I_1}, \ldots, G_{I_n}$, *is* level-wise controllable *with respect to the alphabet partition* $\Sigma := \dot{\cup}_{k\in\{1,\ldots,n\}}(\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$,

then

$$(\forall s \in L(\mathbf{Plant}) \cap \mathbf{H}_{\mathcal{S}} \cap [\cap_{k\in\{1,\ldots,n\}}\mathcal{I}_k]) \quad \mathit{Elig}_{L(\mathbf{Plant})\cap[\cap_{k\in\{1,\ldots,n\}}\mathcal{I}_k]}(s) \cap \Sigma_u \subseteq \mathit{Elig}_{\mathbf{H}_{\mathcal{S}}}(s)$$

where $\mathbf{Plant} := \mathcal{G}_H||_s\mathcal{G}_{L_1}||_s \ldots ||_s\mathcal{G}_{L_n}$ *is the system's* flat plant.

Proof:

Assume that the n^{th} degree ($n \geq 1$) *parallel interface system* is *level-wise controllable.* **(1)**

Let $s \in L(\mathbf{Plant}) \cap \mathbf{H}_{\mathcal{S}} \cap [\cap_{k\in\{1,\ldots,n\}}\mathcal{I}_k]$, and $\sigma \in \text{Elig}_{L(\mathbf{Plant})\cap[\cap_{k\in\{1,\ldots,n\}}\mathcal{I}_k]}(s) \cap \Sigma_u$ **(2)**

We will now show that this implies $\sigma \in \text{Elig}_{\mathbf{H}_{\mathcal{S}}}(s)$

It's sufficient to show that $s\sigma \in \mathbf{H}_{\mathcal{S}}$

We first note that $s, s\sigma \in \mathbf{H} \cap [\cap_{k\in\{1,\ldots,n\}}(\mathbf{L}_k \cap \mathcal{I}_k)] = L(\mathbf{Plant}) \cap [\cap_{k\in\{1,\ldots,n\}}\mathcal{I}_k]$ by **(2)**. **(3)**

By examining the definition of $\Sigma(j)$ for some $j \in \{1, \ldots, n\}$ (see definition of j^{th} serial system extraction: general form on page 89), we see that $\Sigma = \cup_{k\in\{1,\ldots,n\}}\Sigma(k)$

$\Rightarrow$ $(\exists j \in \{1, \ldots, n\})\ \sigma \in \Sigma(j)$ **(4)**

We use this j and note that by **(1)**, we can conclude that *system*(j), the j^{th} *serial system extraction* of our parallel system, is *serial level-wise controllable.* **(5)**

We will use **point III** of the *serial level-wise controllable* definition to show that $P_j(s)\sigma \in \mathbf{H}_{\mathcal{S}}(j)$. See Section 6.3 for the definition of the natural projection P_j.

We first need to show that $P_j(s) \in \mathbf{H}(j) \cap \mathcal{I}(j) \cap \mathbf{H}_{\mathcal{S}}(j)$ and $\sigma \in \text{Elig}_{\mathbf{H}(j)\cap\mathcal{I}(j)}(s) \cap \Sigma_u$

From **(2)** and **(3)**, we have $s \in \mathbf{H} \cap \mathbf{H}_{\mathcal{S}} \cap [\cap_{k\in\{1,\ldots,n\}}\mathcal{I}_k]$

$\Rightarrow P_j(s) \in P_j(\mathbf{H}) \cap P_j(\mathbf{H}_{\mathcal{S}}) \cap [\cap_{k\in\{1,\ldots,n\}}P_j(\mathcal{I}_k)]$

$\Rightarrow P_j(s) \in \mathbf{H}(j) \cap \mathcal{I}(j) \cap \mathbf{H}_{\mathcal{S}}(j)$, by **Proposition 30**.

Similarly, from **(3)** we can conclude $P_j(s\sigma) \in \mathbf{H}(j) \cap \mathcal{I}(j)$

We next note that $\sigma \in \Sigma(j)$ (from **(4)**) implies that $P_j(s\sigma) = P_j(s)\sigma$.

$\Rightarrow \sigma \in \text{Elig}_{\mathbf{H}(j)\cap\mathcal{I}(j)}(P_j(s)) \cap \Sigma_u$

We can now conclude by **point III** of the *serial level-wise controllable* definition that:

$\sigma \in \text{Elig}_{\mathbf{H}_\mathcal{S}(j)}(P_j(s))$ and thus $P_j(s)\sigma = P_j(s\sigma) \in \mathbf{H}_\mathcal{S}(j)$

$\Rightarrow P_j(s\sigma) \in P_j(\mathbf{H}_\mathcal{S})$, by **Proposition 30**.

$\Rightarrow P_j(s\sigma) \in P_j P_{IH}^{-1} L(\mathcal{S}_H)$

As $\Sigma_{IH} = \Sigma_{IH}(j)$ by **Proposition 30**, we have $\Sigma_{IH} \subseteq \Sigma(j)$. We can thus apply **Corollary 2** by taking $\Sigma_a = \Sigma(j)$, $\Sigma_b = \Sigma_{IH}$, and $L_b = L(\mathcal{S}_H)$ and thus conclude:

$s\sigma \in P_{IH}^{-1} L(\mathcal{S}_H) = \mathbf{H}_\mathcal{S}$, as required.

QED

Chapter 7

Parallel Manufacturing Example

To illustrate the parallel case, we will look at the simple manufacturing system shown in Figure 7.1. The system is composed of three manufacturing units running in parallel, a testing unit, material feedback, a packaging unit, plus three buffers to insure a proper flow of material.

For the manufacturing units (indexed by $j = \mathrm{I}, \mathrm{II}, \mathrm{III}$), we will reuse the systems developed in Chapter 4 with the packaging unit removed and treating the system as a flat model (i.e. ignoring for now the system's own interface structure). Figure 7.2 shows the plant models for each manufacturing unit.

For the source, sink, test unit, and packaging unit, we introduced new plant models. They are shown in Figure 7.3. For the three buffers, they will be implemented as supervisors.

7.1 Design Details

For this example, we want to design a parallel case interface system with the structure shown in Figure 7.4. We will treat the three independent manufacturing units as our *low levels.* Our first step is to define which plant model exists at which level. This is shown in Figure 7.5.

We now need to define *interfaces* between the *high level* and each of the three *low levels.* Normally, each *interface* would be quite different, but since each *low level* is an instance of the same manufacturing unit, it makes sense that the *interfaces* are also all of the same form. Figure 7.6 shows the *interface* to *low level* j. For the remainder of this chapter, we will take $j = \mathrm{I}, \mathrm{II}, \mathrm{III}$.

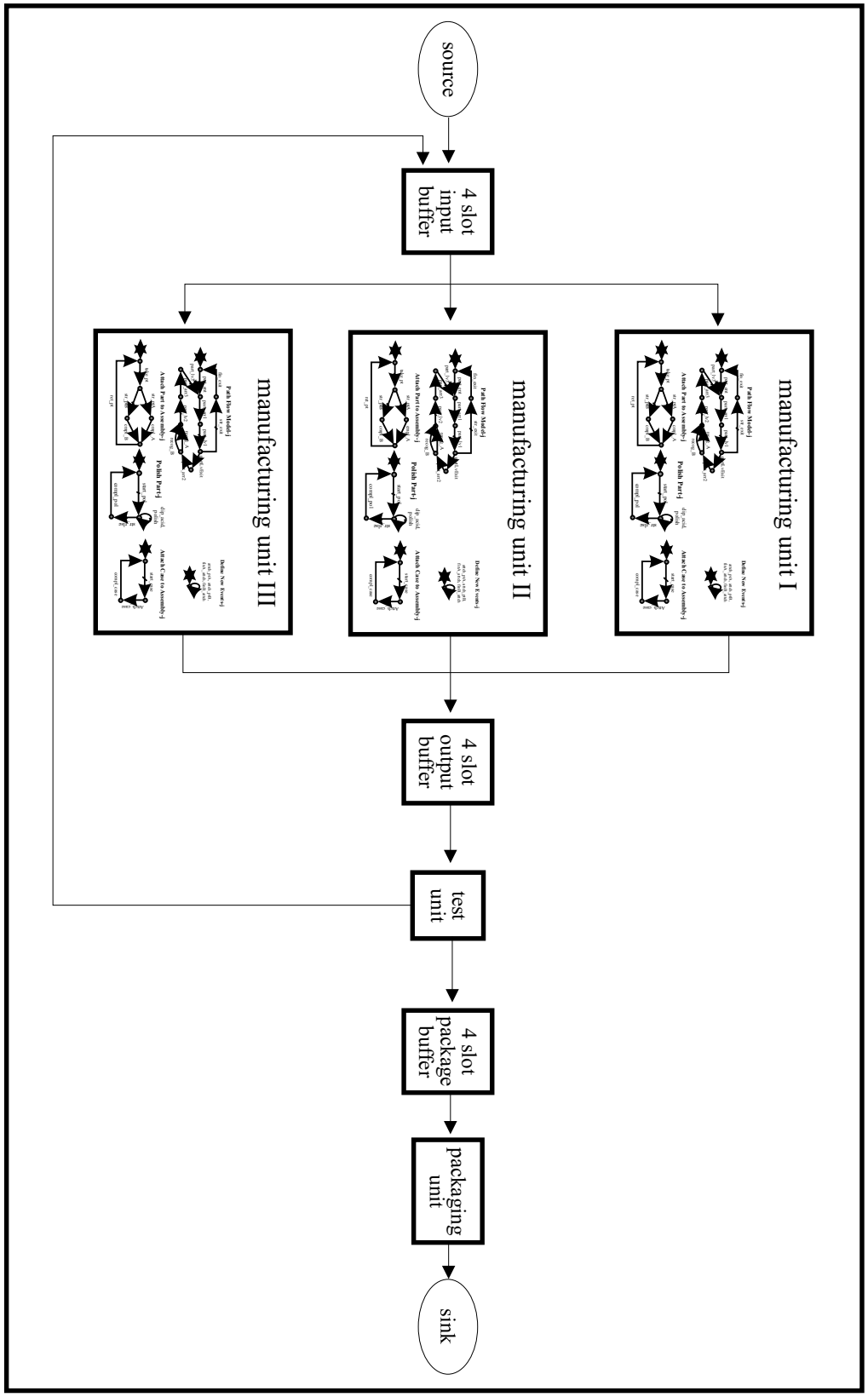


Figure 7.1: Block Diagram of Parallel Plant

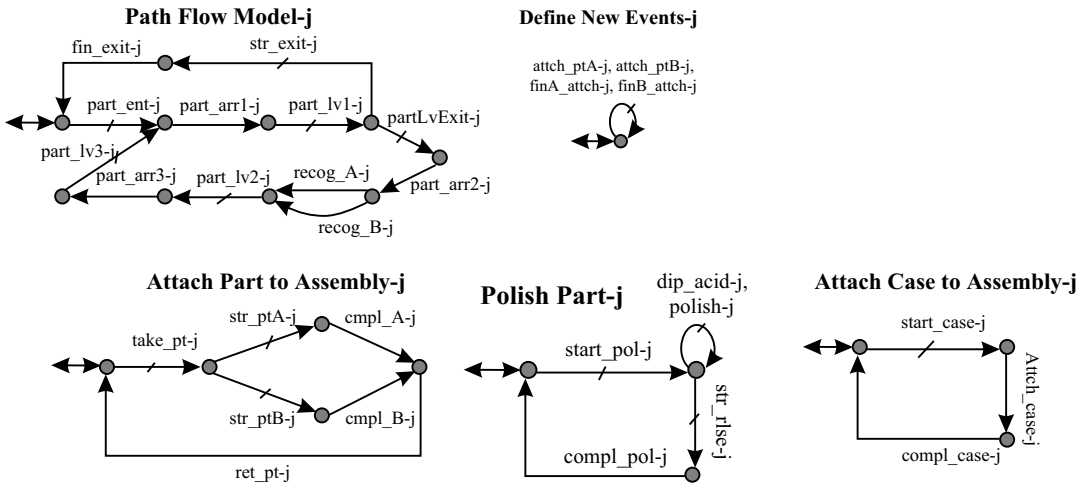


Figure 7.2: Plant Models for Manufacturing Unit j

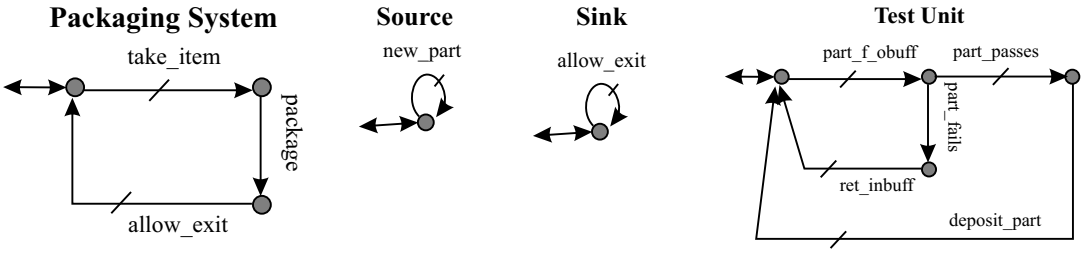


Figure 7.3: New Plant Models

We can now define the alphabet partition $\Sigma := [\dot{\cup}_{k\in\{\mathrm{I,II,III}\}}(\Sigma_{L_k}\dot{\cup}\Sigma_{R_k}\dot{\cup}\Sigma_{A_k})]\ \dot{\cup}\ \Sigma_H$ as below:

$$
\begin{aligned}
\Sigma_H &= \{\textit{take\_item}, \textit{package}, \textit{allow\_exit}, \textit{new\_part}, \textit{part\_f\_obuff}, \textit{part\_passes}, \textit{part\_fails},\\
&\quad \textit{ret\_inbuff}, \textit{deposit\_part}\}\\
\Sigma_{R_j} &= \{\textit{part\_ent-j}\}\\
\Sigma_{A_j} &= \{\textit{fin\_exit-j}\}\\
\Sigma_{L_j} &= \{\textit{start\_pol-j}, \textit{attch\_ptA-j}, \textit{attch\_ptB-j}, \textit{start\_case-j}, \textit{comp\_pol-j}, \textit{finA\_attch-j}, \textit{finB\_attch-j},\\
&\quad \textit{compl\_case-j}, \textit{part\_arr1-j}, \textit{part\_lv1-j}, \textit{partLvExit-j}, \textit{str\_exit-j}, \textit{part\_arr2-j}, \textit{recog\_A-j}, \textit{recog\_B-j},\\
&\quad \textit{part\_lv2-j}, \textit{part\_arr3-j}, \textit{part\_lv3-j}, \textit{take\_pt-j}, \textit{str\_ptA-j}, \textit{str\_ptB-j}, \textit{compl\_A-j}, \textit{compl\_B-j},\\
&\quad \textit{ret\_pt-j}, \textit{dip\_acid-j}, \textit{polish-j}, \textit{str\_rlse-j}, \textit{attch\_case-j}\}
\end{aligned}
$$

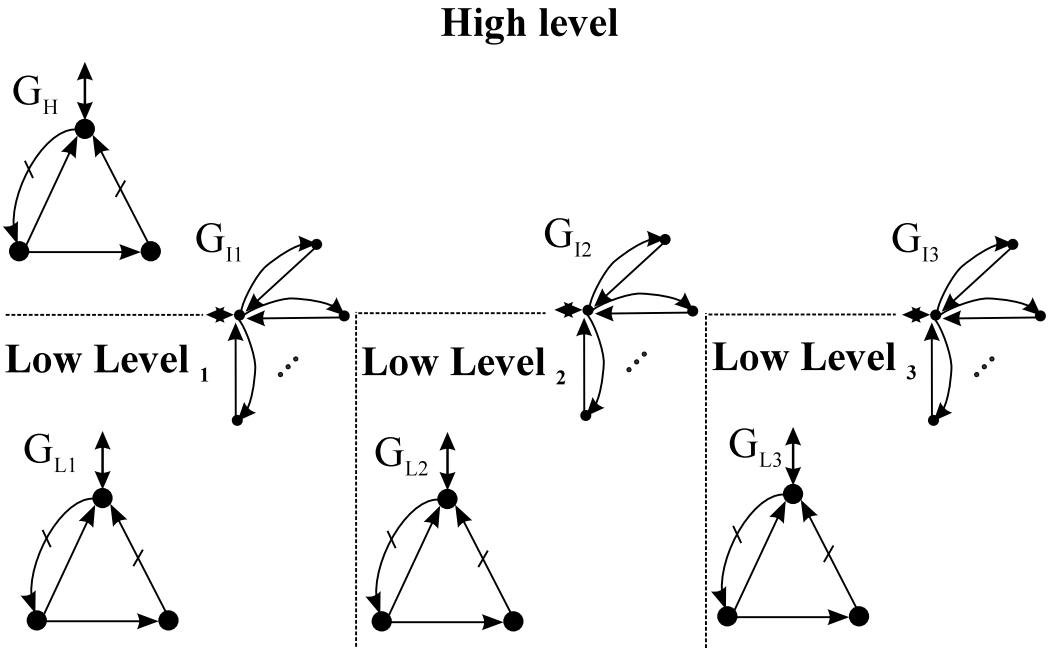


Figure 7.4: Desired Interface Structure

Our next step is to design new supervisors for our *low levels.* As we are reusing the manufacturing unit designed in Chapter 4, we already have a system that is designed to accept a new part, process it appropriately, and then allow the part to leave the unit; thus we can simply reuse supervisors from that chapter. They are shown in Figure 7.7 for *low level* j.

For the *high level,* we need to design supervisors to implement the input buffer, the output buffer, and the package buffer. Each buffer should have four slots and should never underflow or overflow. The corresponding supervisors are shown in Figure 7.8. Finally, we note that the above supervisors were designed by hand, but we could have also employed synthesis methods.

7.2 The Final System

Now that we have defined the individual components of the *system,* it is time to put everything together. We start by examining the j^{th} *low level subsystem.* This is shown in Figure 7.9, where we have labelled which DES belong to the j^{th} *low level subsystem,* the j^{th} low level plant, and the j^{th} low level supervisor (each formed by the synchronous product of the indicated DES). We can now assemble the complete parallel system shown in Figure 7.10, minus DES **Ensure_matFb** which we will introduce in Section 7.3. In addition to the *low level subsystems,* Figure 7.10 shows which DES belong to the *high level*

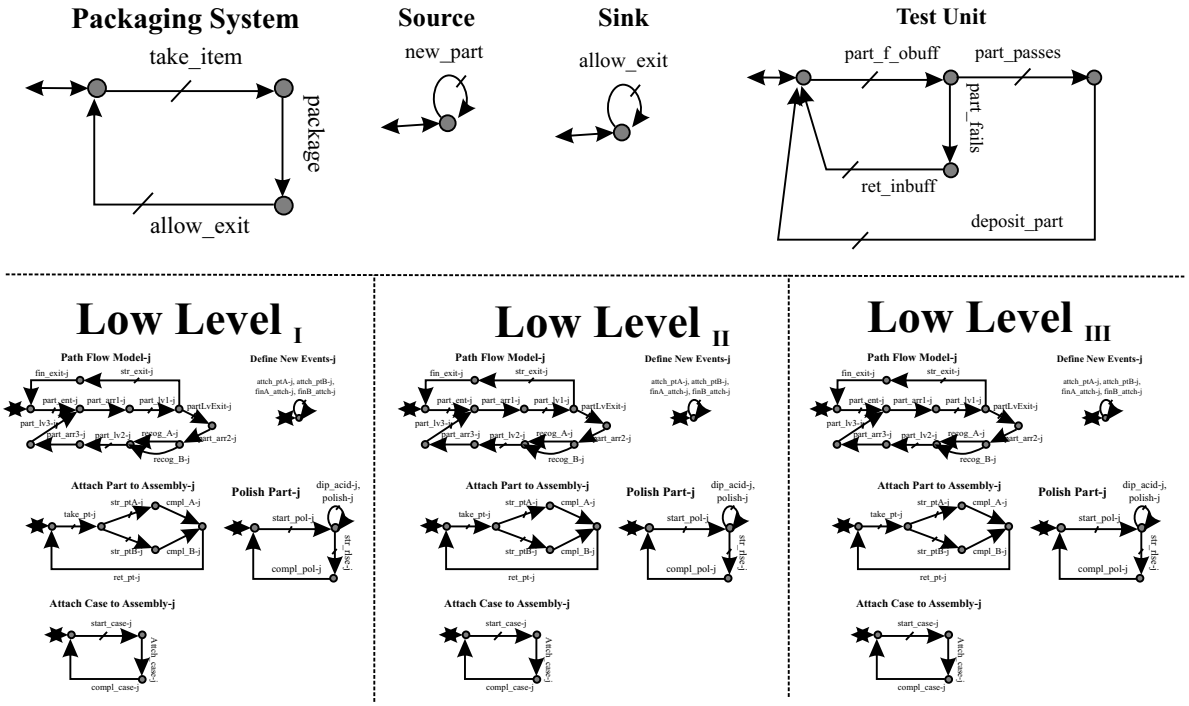


Figure 7.5: Plant Models for Parallel System

subsystem, high level plant, and the high level supervisor (each formed by the synchronous product of the indicated DES).

We now define the *flat system,* the *flat plant,* and the *flat supervisor* as follows:

$$
\begin{aligned}
G &= G_H||_s G_{L_{\mathrm{I}}}||_s G_{L_{\mathrm{II}}}||_s G_{L_{\mathrm{III}}}||_s G_{I_{\mathrm{I}}}||_s G_{I_{\mathrm{II}}}||_s G_{I_{\mathrm{III}}} \\
\textbf{Plant} &:= \mathcal{G}_H||_s \mathcal{G}_{L_{\mathrm{I}}}||_s \mathcal{G}_{L_{\mathrm{II}}}||_s \mathcal{G}_{L_{\mathrm{III}}} \\
\textbf{Sup} &:= \mathcal{S}_H||_s \mathcal{S}_{L_{\mathrm{I}}}||_s \mathcal{S}_{L_{\mathrm{II}}}||_s \mathcal{S}_{L_{\mathrm{III}}}||_s G_{I_{\mathrm{I}}}||_s G_{I_{\mathrm{II}}}||_s G_{I_{\mathrm{III}}}
\end{aligned}
$$

7.3 Evaluating Properties

Our next step is to verify that the *flat system* is nonblocking and that the *flat supervisor* is controllable for the *flat plant.* To achieve this, we will show that the system is *level-wise nonblocking and controllable,* and *interface consistent.* Our first step is to show that sets Σ_H, Σ_{R_j}, Σ_{A_j}, and Σ_{L_j} are pairwise disjoint. This can be seen by inspection of their definitions.

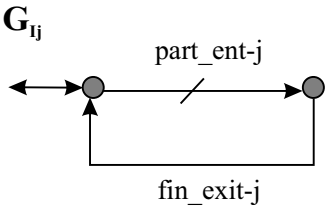


Figure 7.6: Interface Model for *Low Level j*

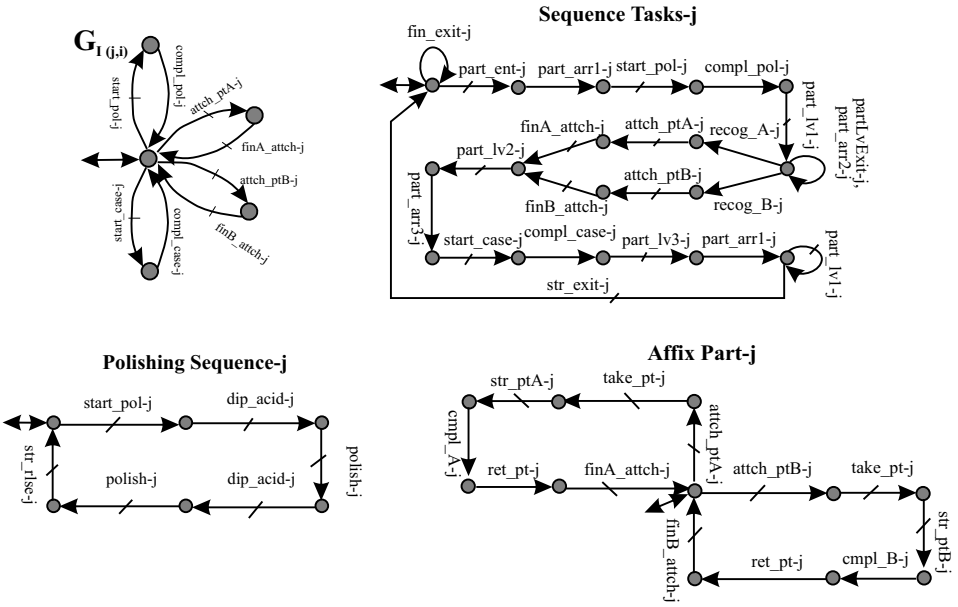


Figure 7.7: Supervisors for *Low Level j*

As we have a parallel system of degree $n = 3$, we must verify that the j^{th} *serial system extraction: subsystem form* are *serial level-wise nonblocking,* and *serial interface consistent,* and that the j^{th} *serial system extraction: general form* is *serial level-wise controllable.* We start by defining the *serial extraction systems: system*(I), *system*(II), and *system*(III). *System*(I) is defined below, with the DES definitions shown in Figure 7.11, minus DES **Ensure_matFb** which we will introduce later in section. The remaining systems are left as an exercise.

Parallel Subsystem Based Form:

$$\begin{aligned} G_H(I) &:= G_H||_s G_{I_{\rm II}}||_s G_{I_{\rm III}} \\ G_L(I) &:= G_{L_{\rm I}} \end{aligned}$$

$$
\begin{aligned}
G_I(I) &:= G_{I_{\mathrm{I}}} \\
\Sigma_H(I) &:= [\dot{\cup}_{k\in\{\mathrm{II},\mathrm{III}\}}\Sigma_{I_k})]\,\dot{\cup}\,\Sigma_H \\
\Sigma_L(I) &:= \Sigma_{L_{\mathrm{I}}} \\
\Sigma_R(I) &:= \Sigma_{R_{\mathrm{I}}} \\
\Sigma_A(I) &:= \Sigma_{A_{\mathrm{I}}} \\
\Sigma(I) &:= \Sigma_H(I)\,\dot{\cup}\,\Sigma_L(I)\,\dot{\cup}\,\Sigma_R(I)\,\dot{\cup}\,\Sigma_A(I)
\end{aligned}
$$

Parallel General Form:

$$
\begin{aligned}
\mathcal{G}_H(I) &:= \mathcal{G}_H||_s G_{I_{\mathrm{II}}}||_s G_{I_{\mathrm{III}}} \\
\mathcal{S}_H(I)) &:= \mathcal{S}_H \\
\mathcal{G}_L(I) &:= \mathcal{G}_{L_{\mathrm{I}}} \\
\mathcal{S}_L(I) &:= \mathcal{S}_{L_{\mathrm{I}}}
\end{aligned}
$$

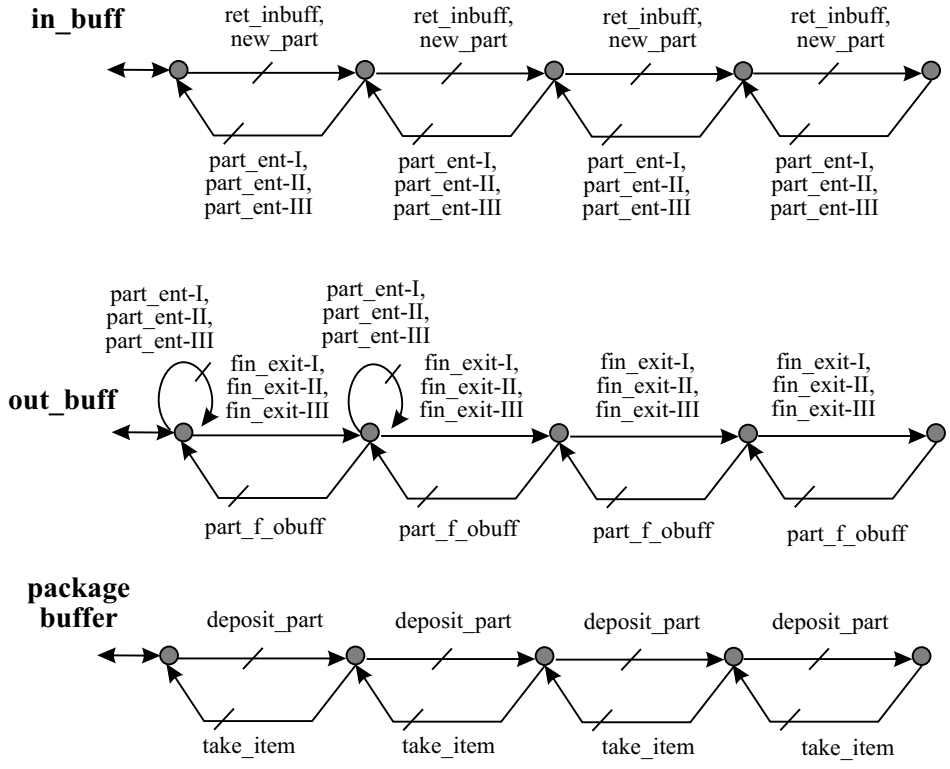


Figure 7.8: Supervisors for *High Level*

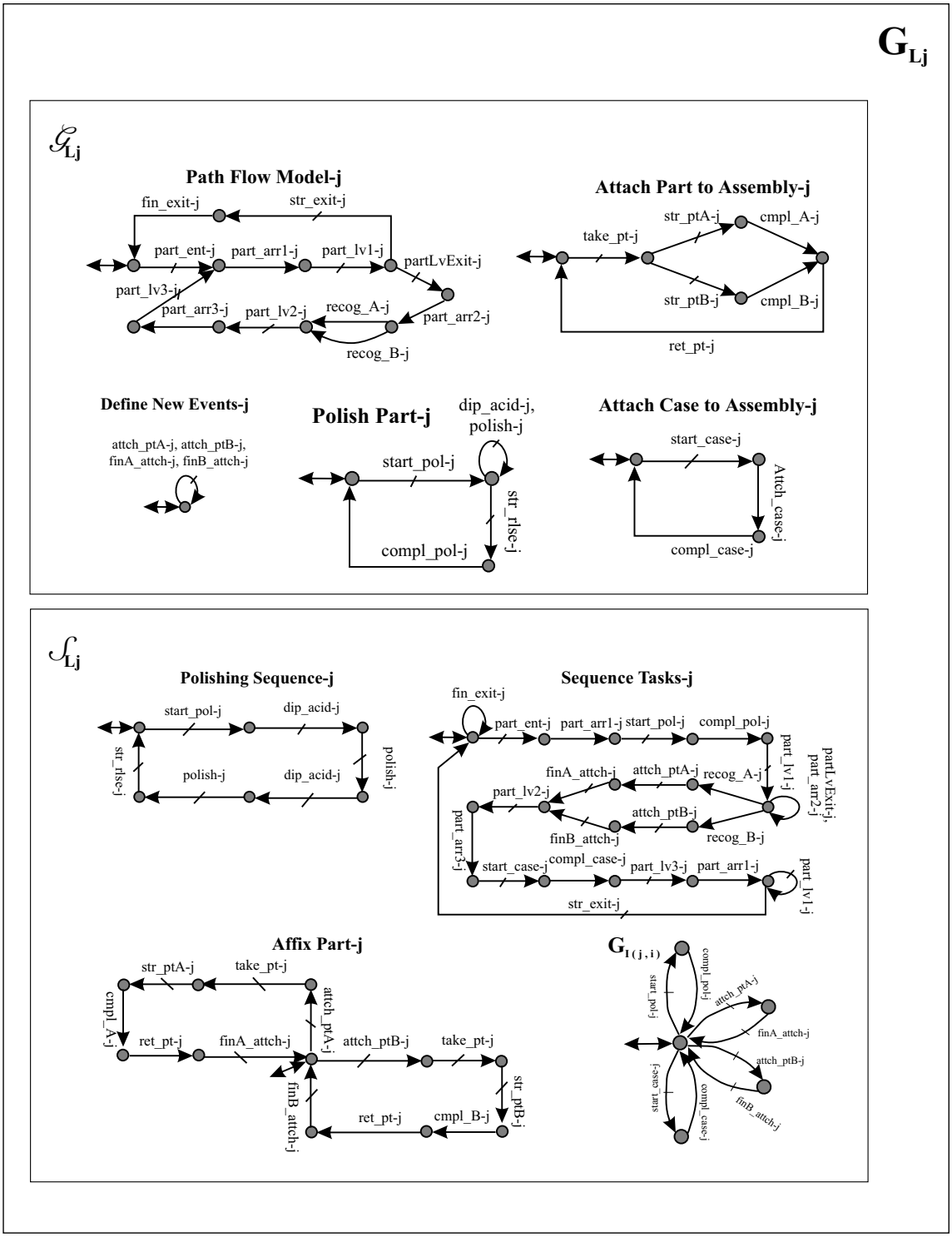


Figure 7.9: *Low Level Subsystem j*

High level Subsystem

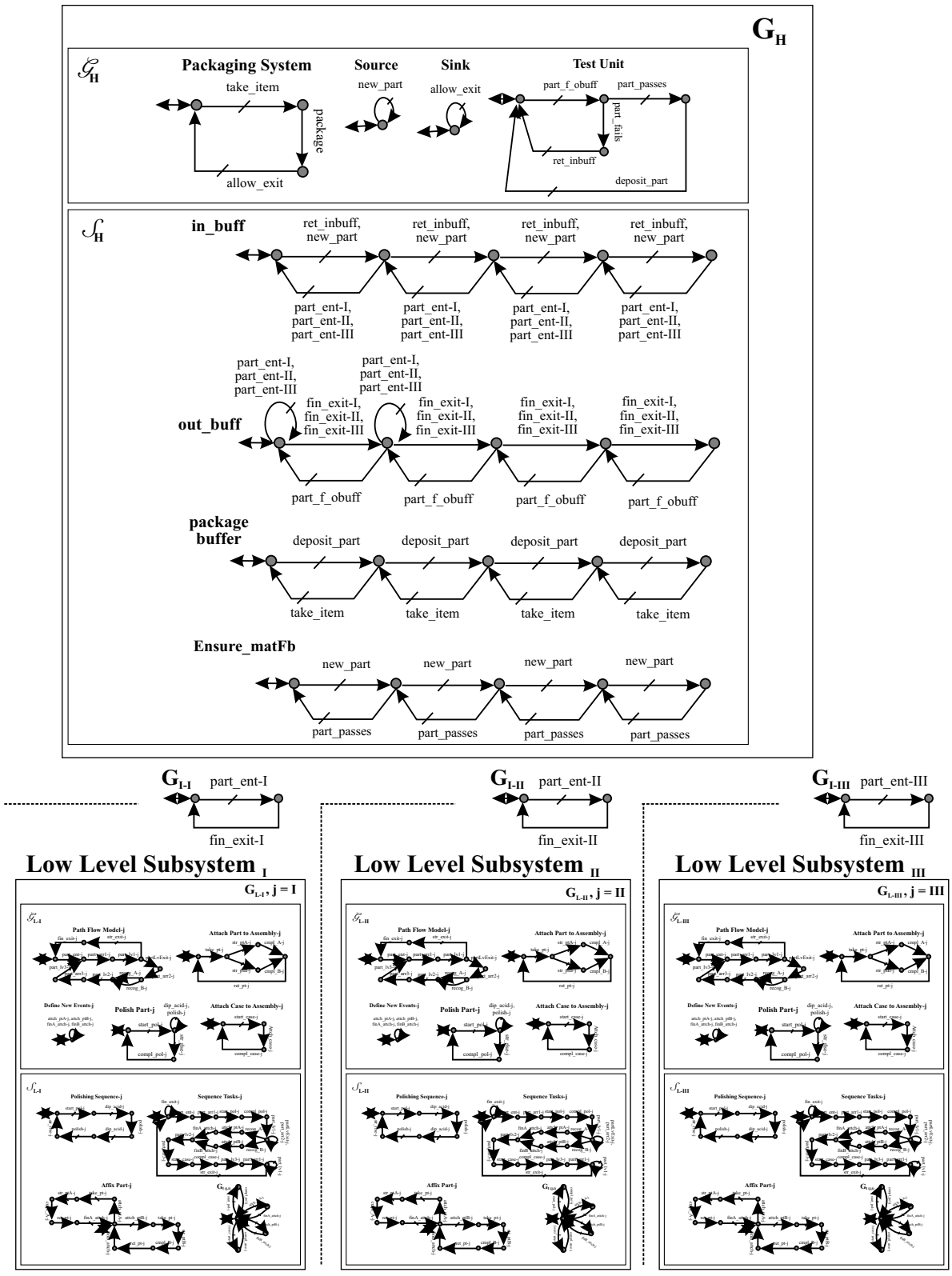
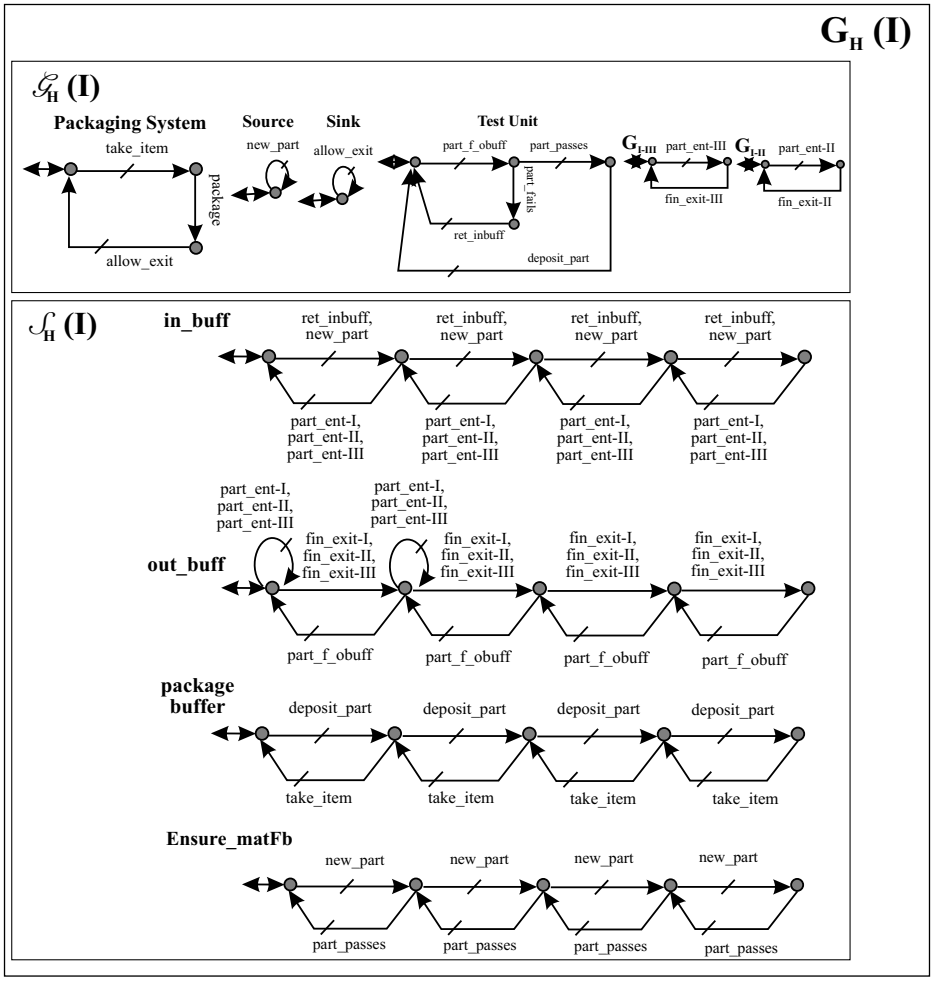
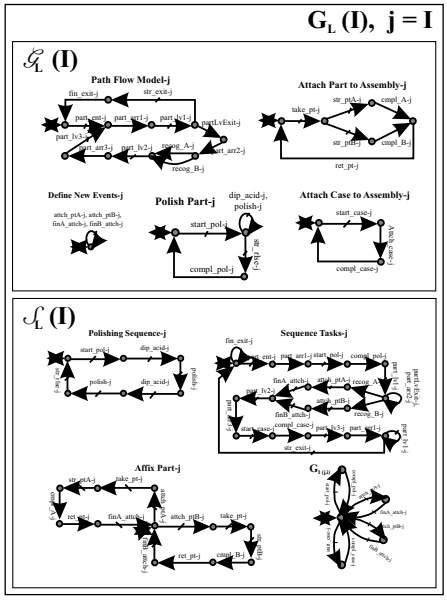


Figure 7.10: Complete Parallel System



High level Subsystem (I)

Low Level Subsystem (I)



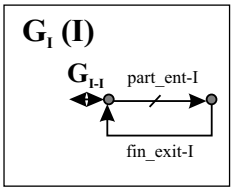


Figure 7.11: *Serial Extraction System I*

The reader should note that the *interfaces* and the three *low level subsystems,* plants, and supervisors are identical up to event relabelling, and thus isomorphic. This means that the *serial level-wise nonblocking and controllability,* and *serial interface consistency* verifications for them will be identical. We thus need to only evaluate one *low level,* and the results will apply to all three.

We now apply our software tool to the *serial extraction systems* and we find that the *high level* is blocking. The reason for this is that the supervisor for the input buffer does not take into account material feedback. This can be seen by analysing the sequence of events shown in Figure 7.12, and seeing how it leads DES **Test Unit, in_buff,** and **out_buff** to a blocking state.[1] To properly handle

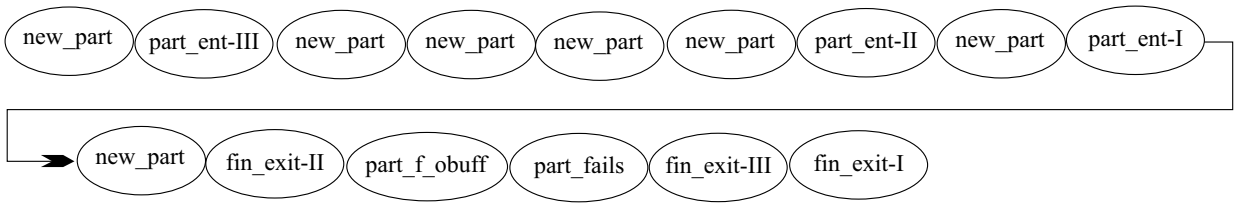


Figure 7.12: Deadlock Sequence

material feedback, we add the supervisor shown in Figure 7.13. This supervisor would be added to the *flat high level supervisor*, as shown in Figure 7.10. Figure 7.11 shows the new *system*(I). We now apply our research tool to the *serial extraction systems* and we find that each system is *serial level-wise nonblocking and controllable,* and *serial interface consistent.* We can thus conclude by Theorems 3 and 4, that the *flat system* is nonblocking and that the *flat supervisor* is controllable for the *flat plant.*

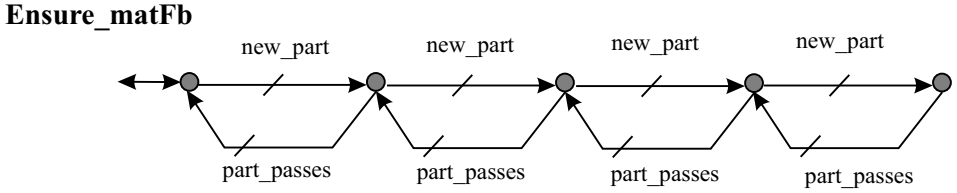


Figure 7.13: Material Feedback Supervisor

[1]The material feedback oversight was left in purposely to show that the interface structure alone does not guarantee nonblocking.

7.4 Comparison to Standard Method

The above computation was run on a 750MHz Athlon system, with 512MB of RAM, 2GB of swap, and running Redhat Linux 6.2. The *high level* consisted of 3120 states and each *low level* of 35 states. The computation took 0.15s to run and required 5MB of memory.

A standard nonblocking verification was done on the *flat system* which consists of 5,702,550 states. The computation ran for 40 minutes, required 850MB of memory, and found the system to be nonblocking.[2] In short, the standard method took 16438 times longer, and required 170 times more memory.

From this example, we can see that the *interface method* not only can offer a significant reduction in verification time and required resources, but that it also greatly improves the re-usability of the system. For example, a system designer would only need to design supervisors for the manufacturing unit once, and could then use them in each instantiation. Treating the manufacturing system as a *low level subsystem,* we only have to verify the subsystem once even though we use it in multiple places. In addition, we can change the *low level subsystem* without affecting the *high level* as long as the *interface* remains the same, and the *low level* remains *serial nonblocking, controllable,* and *interface consistent.*[3] This offers a potentially great savings in design and verification time.

7.5 AIP Example

We present a full example application of the theory based on the automated manufacturing system of the Atelier Inter-établissement de Productique (AIP) [4, 7] in the report [10]. The AIP system is broken down into a *high level* and seven *low levels* corresponding to the three assembly stations and four transport Units. In total, the example contains 181 DES, with an estimated closed-loop state space of 7×10^{21}.

The analysis in [10] finds the system to be *interface consistent, level-wise nonblocking,* and *level-wise controllable.* Thus we can conclude by Theorems 3 and 4, that the *flat system* is nonblocking and that the system's *flat supervisor* is controllable for the *flat plant.* For further details of the application, we refer the reader to [10].

[2] A controllability check was also run using standard methods and the *flat supervisor* was found to be controllable for the *flat plant.*

[3] More correctly, the *low level* continues to satisfy its portion of these properties.

Chapter 8

Conclusions

Hierarchical interface-based supervisory control offers an effective method to model systems with a natural client-server architecture. The method offers an intuitive way to model and design the system. Using multiple *low level subsystems* allows the subsystems to be independently modelled and verified, but still allowing a high degree of concurrent operation. As each requirement can be verified using only one subsystem, the entire plant model never needs to be constructed or traversed (in computer memory), offering potentially significant savings in computation.

It is clear from the definitions in Chapters 2, 3, 5, and 6 that once we have defined our *interface* and event partition, evaluating our *high* and *low level subsystems* for compliance can be done independently of each other. This means we can evaluate one *high (low) level subsystem* and use it with any *low (high) level subsystem* that satisfies the low (high) level portion of our definitions for the given *interface* and event partition. This provides us with the infrastructure required for component reuse.

In this report, we present two complete examples that illustrate the method, and then we discuss a large example based on the automated manufacturing system of the Atelier Inter-établissement de Productique (AIP) which we describe in detail in [10]. As the example contains 181 DES with an estimated closed-loop state space of 7×10^{21}, it demonstrates that Hierarchical Interface-based Supervisory Control can be applied to interesting systems of realistic complexity that were previously far beyond the means of previous monolithic, modular, or hierarchical supervisor design techniques.

Appendix A

Localization of Nonblocking Conditions

In this appendix we derive equivalent conditions for the serial interface consistency of Section 2.2. that only involve "local" information - properties of the component and its interface. The result is that these conditions can now be verified at the component level for arbitrary systems satisfying the alphabet partitioning criteria, $\Sigma := \Sigma_H \dot{\cup} \Sigma_L \dot{\cup} \Sigma_R \dot{\cup} \Sigma_A$.

Recall definitions from Section 2.1.2

As we will often be referring to different groupings of events, we define the following subsets:

$$\begin{aligned} \Sigma_I &:= \quad \Sigma_R \dot{\cup} \Sigma_A \qquad \textit{Interface Events} \\ \Sigma_{IH} &:= \quad \Sigma_H \dot{\cup} \Sigma_R \dot{\cup} \Sigma_A \qquad \textit{Interface \& High Level Events} \\ \Sigma_{IL} &:= \quad \Sigma_L \dot{\cup} \Sigma_R \dot{\cup} \Sigma_A \qquad \textit{Interface \& Low Level Events} \end{aligned}$$

$$\begin{aligned} P_{IH} &: \Sigma^* \rightarrow \quad \Sigma_{IH}^* \\ P_{IL} &: \Sigma^* \rightarrow \quad \Sigma_{IL}^* \\ P_I &: \Sigma^* \rightarrow \quad \Sigma_I^* \end{aligned}$$

and the following useful languages:

$$\begin{aligned} \mathcal{H} := P_{IH}^{-1}(L(G_H)), \quad \mathcal{H}_m := P_{IH}^{-1}(L_m(G_H)) \quad \subseteq \Sigma^* \\ \mathcal{L} := P_{IL}^{-1}(L(G_L)), \quad \mathcal{L}_m := P_{IL}^{-1}(L_m(G_L)) \quad \subseteq \Sigma^* \end{aligned}$$

$$\mathcal{I} := P_I^{-1}(L(G_I)), \qquad \mathcal{I}_m := P_I^{-1}(L_m(G_I)) \quad \subseteq \Sigma^*$$

We begin with a simple result to aid in the development of the equivalent definitions.

Proposition 33 *With the symbols as defined above,*

$$P_{IH}(P_{IH}^{-1}(L(G_H))\Sigma_A \cap P_I^{-1}(L(G_I))) = P_{IH}(P_{IH}^{-1}(L(G_H))\Sigma_A) \cap P_{IH}(P_I^{-1}(L(G_I)))$$

and

$$P_{IL}(P_{IL}^{-1}(L(G_L))\Sigma_R \cap P_I^{-1}(L(G_I))) = P_{IL}(P_{IL}^{-1}(L(G_L))\Sigma_R) \cap P_{IL}(P_I^{-1}(L(G_I)))$$

Proof:

Below we provide a proof of only the first equality. The proof of the second is identical with the appropriate substitutions.

($\subseteq$) Follows immediately from the fact that for $L_1, L_2 \subseteq \Sigma^*$, we have $P_{IH}(L_1 \cap L_2) \subseteq P_{IH}(L_1) \cap P_{IH}(L_2)$.

Taking L_1 to be $P_{IH}^{-1}(L(G_H))\Sigma_A$ and L_2 to be $P_I^{-1}(L(G_I))$, we have the desired result.

($\supseteq$) Suppose $s \in P_{IH}(P_{IH}^{-1}(L(G_H))\Sigma_A) \cap P_{IH}(P_I^{-1}(L(G_I)))$

If $s \in P_{IH}(P_I^{-1}(L(G_I)))$ then $s \in P_I^{-1}(L(G_I))$ since $\Sigma_I \subseteq \Sigma_{IH}$.

Also, if $s \in P_{IH}(P_{IH}^{-1}(L(G_H))\Sigma_A)$ then $s \in P_{IH}(P_{IH}^{-1}(L(G_H)))\Sigma_A$ since $\Sigma_A \subseteq \Sigma_{IH}$.

Thus $s = s'\alpha$ for some $s' \in P_{IH}(P_{IH}^{-1}(L(G_H)))$ and $\alpha \in \Sigma_A$. But then it also follows that $s' \in P_{IH}^{-1}(L(G_H))$ since $P_{IH}(s') = s'$. So $s \in P_{IH}^{-1}(L(G_H))\Sigma_A$.

Therefore

$$s \in P_{IH}^{-1}(L(G_H))\Sigma_A \cap P_I^{-1}(L(G_I))$$

and $P_{IH}(s) = P_{IH}(s'\alpha) = P_{IH}(s')P_{IH}(\alpha) = s'\alpha = s$, so

$$s \in P_{IH}(P_{IH}^{-1}(L(G_H))\Sigma_A \cap P_I^{-1}(L(G_I)))$$

as required.

QED

As a finally preliminary we need to define restrictions of a pair of projection functions. Because $\Sigma_I \subseteq \Sigma_I \cup \Sigma_H = \Sigma_{IH}$, we can restrict $P_I : \Sigma^* \to \Sigma_I*$, the natural projection from Σ^* to Σ_I^*, to Σ_{IH}, denoted $P_I|\Sigma_{IH}^* : \Sigma_{IH}^* \to \Sigma_I^*$. This function takes strings over Σ_{IH} and returns a string over Σ_I in the

obvious way by deleting all occurrences of events from $\Sigma_{IH} - \Sigma_I$ in a string. Similarly we can define $P_I|\Sigma^*_{IL} : \Sigma^*_{IL} \rightarrow \Sigma^*_I$.

We now turn our attention to the *High Level Task Completion Agreement* Property

3. $(\forall s \in \mathcal{H} \cap \mathcal{I})\ \text{Elig}_{\mathcal{I}}(s)\ \cap\ \Sigma_A \subseteq \text{Elig}_{\mathcal{H}}(s)$ *High level task completion agreement*

and the *Low level task request agreement* Property.

4. $(\forall s \in \mathcal{L} \cap \mathcal{I})\ \text{Elig}_{\mathcal{I}}(s)\ \cap\ \Sigma_R \subseteq \text{Elig}_{\mathcal{L}}(s)$ *Low level task request agreement*

of the *serial interface consistent* definition.

The following theorem provides equivalent definitions for these two properties that are formulated solely in terms of events that are local to a level.

Theorem 5

(i) $(\forall s \in \mathcal{H} \cap \mathcal{I})\ Elig_{\mathcal{I}}(s)\ \cap\ \Sigma_A \subseteq Elig_{\mathcal{H}}(s)$ *iff* $L(G_H)\Sigma_A \cap P_I|\Sigma^*_{IH}{}^{-1}(L(G_I)) \subseteq L(G_H)$

(ii) $(\forall s \in \mathcal{L} \cap \mathcal{I})\ Elig_{\mathcal{I}}(s)\ \cap\ \Sigma_R \subseteq Elig_{\mathcal{L}}(s)$ *iff* $L(G_L)\Sigma_R \cap P_I|\Sigma^*_{IL}{}^{-1}(L(G_I)) \subseteq L(G_L)$

Proof:

Once again we will only provide proof of the *high level* result since the proof of the *low level* result can be obtained by the obvious substitutions.

Condition 3 above can be reformulated in the more standard notation of [25] as:

$$(\mathcal{H} \cap \mathcal{I})\Sigma_A \cap \mathcal{I} \subseteq H$$

Due to the fact that $\mathcal{I}$ is prefix closed, $(\mathcal{H} \cap \mathcal{I})\Sigma_A \cap \mathcal{I} = \mathcal{H}\Sigma_A \cap \mathcal{I}$. Thus we can simplify the above to:

$$\mathcal{H}\Sigma_A \cap \mathcal{I} \subseteq \mathcal{H}$$

Replacing $\mathcal{H}$ and $\mathcal{I}$ by their definitions we obtain

$$P^{-1}_{IH}(L(G_H))\Sigma_A \cap P^{-1}_I(L(G_I)) \subseteq P^{-1}_{IH}(L(G_H)) \tag{A.1}$$

We will use this equivalent formulation in place of the left hand side of (i) of the theorem in the remainder of the proof.

($\Rightarrow$) Applying P_{IH} to both sides of (A.1) we obtain

$$P_{IH}(P^{-1}_{IH}(L(G_H))\Sigma_A \cap P^{-1}_I(L(G_I))) \subseteq P_{IH}(P^{-1}_{IH}(L(G_H))) \tag{A.2}$$

By Proposition 33, we can distribute P_{IH} over the intersection on the left side of (A.2) to obtain:

$$P_{IH}(P_{IH}^{-1}(L(G_H))\Sigma_A) \cap P_{IH}(P_I^{-1}(L(G_I))) \subseteq P_{IH}(P_{IH}^{-1}(L(G_H)))$$

We can then use the fact because $\Sigma_A \subseteq \Sigma_{IH}$ we have $P_{IH}(P_{IH}^{-1}(L(G_H))\Sigma_A) = P_{IH}(P_{IH}^{-1}(L(G_H)))\Sigma_A = L(G_H)\Sigma_A$ so the above simplifies to:

$$L(G_H)\Sigma_A \cap P_{IH}(P_I^{-1}(L(G_I))) \subseteq L(G_H)$$

Then it follows that $P_{IH} \circ P_I^{-1} = (P_I|\Sigma_{IH}^*)^{-1}$ this simplifies to:

$$L(G_H)\Sigma_A \cap P_I|\Sigma_{IH}{}^{-1}(L(G_I)) \subseteq L(G_H) \tag{A.3}$$

($\Leftarrow$) Suppose (A.3) holds. We can reverse the preceding argument to obtain (A.2). By Proposition 3, subset inclusion is preserved under inverse projection so applying P_{IH}^{-1} to both sides of (A.2) we obtain:

$$P_{IH}^{-1} \cdot P_{IH}(P_{IH}^{-1}(L(G_H))\Sigma_A \cap P_I^{-1}(L(G_I))) \subseteq P_{IH}^{-1} \cdot P_{IH}(P_{IH}^{-1}(L(G_H)))$$

We then have

$$\begin{aligned} P_{IH}^{-1}(L(G_H))\Sigma_A \cap P_I^{-1}(L(G_I)) &\subseteq P_{IH}^{-1} \cdot P_{IH}(P_{IH}^{-1}(L(G_H))\Sigma_A \cap P_I^{-1}(L(G_I))) \\ &\subseteq P_{IH}^{-1} \cdot P_{IH}(P_{IH}^{-1}(L(G_H))) \\ &= P_{IH}^{-1}(L(G_H)) \end{aligned}$$

where the first inequality follow from the fact that for any $L \subseteq \Sigma^*, L \subseteq P_{IH}^{-1} \cdot P_{IH}(L)$ and the final equality follows from Proposition 6.

QED

Part (i) of Theorem 5 is just stating that language generated by the *high level subsystem* is controllable with respect to the *interface* when *answer events* are treated as uncontrollable. Similarly, part (ii) of Theorem 5 states that the language generated by the *low level subsystem* is controllable with respect to the *interface* when *request events* are treated as uncontrollable. We thus see that the interface consistency definitions require that the *interface* can control communications between the components.

The two remaining conditions of the *serial interface consistent* definition are:

5. $(\forall s \in \Sigma^*.\Sigma_R \cap \mathcal{L} \cap \mathcal{I})$

$\text{Elig}_{\mathcal{L}\cap\mathcal{I}}(s\Sigma_L^*) \cap \Sigma_A = \text{Elig}_{\mathcal{I}}(s) \cap \Sigma_A$ *Low level task completion agreement*

where $\text{Elig}_{\mathcal{L}\cap\mathcal{I}}(s\Sigma_L^*) := \cup_{l\in\Sigma_L^*}\text{Elig}_{\mathcal{L}\cap\mathcal{I}}(sl)$

6. $(\forall s \in \mathcal{L}\cap\mathcal{I})$

   $s \in \mathcal{I}_m \Rightarrow (\exists l \in \Sigma_L^*)\ sl \in \mathcal{L}_m \cap \mathcal{I}_m$ *Low level marking agreement*

Using the fact that $P_{IL}(\mathcal{L}\cap\mathcal{I}) = L(G_L||_sG_I)$ it follows directly from the definitions that these two properties can be reformulated locally as:

5′. $(\forall s \in \Sigma_{IL}^*.\Sigma_R \cap L(G_L||_sG_I))$

   $\text{Elig}_{L(G_L||_sG_I)}(s\Sigma_L^*) \cap \Sigma_A = \text{Elig}_{L(G_I)}(P_I(s)) \cap \Sigma_A$ *Low level task completion agreement*

   where $\text{Elig}_{L(G_L||_sG_I)}(s\Sigma_L^*) := \cup_{l\in\Sigma_L^*}\text{Elig}_{L(G_L||_sG_I)}(sl)$

6′. $(\forall s \in L(G_L||_sG_I))$

   $P_I(s) \in L_m(G_I) \Rightarrow (\exists l \in \Sigma_L^*)\ sl \in L_m(G_L||_sG_I)$ *Low level marking agreement.*

Bibliography

[1] N. Alsop. *Formal Techniques for the Procedural Control of Industrial Processes*. PhD thesis, Department of Chemical Engineering and Chemical Technology, Imperial College of Science, Technology and Medicine, London, 1996.

[2] Rajeev Alur and Thomas A. Henzinger. Local liveness for compositional modelling of fair reactive systems. In *Proc. of seventh Int. Conf. on Computer-aided Verification, Lecture Notes in Computer Science*, pages 166–179, 1995.

[3] Adnan Aziz, Vigyan Singhal, and Gitanjali M. Swamy. Minimizing interacting finite state machines: A compositional approach to language containment. In *Proc. of IEEE Int. Conf. on Computer Design: VLSI in Computers and Processors*, pages 255–261, Cambridge, Massachusetts, Oct 1994.

[4] Bertil Brandin and François Charbonnier. The supervisory control of the automated manufacturing system of the AIP. In *Proc. Rensselaer's 1994 Fourth International Conference on Computer Integrated Manufacturing and Automation Technology*, pages 319–324, Troy, Oct 1994.

[5] Y. Brave and M. Heymann. Control of discrete event systems modeled as hierarchical state machines. *IEEE Trans. on Automatic Control*, 38(12):1803–1819, Dec 1993.

[6] P.E. Caines and Y.J. Wei. The hierarchical lattices of a finite machine. *Systems Control Letters*, 25:257–263, July 1995.

[7] F. Charbonnier. Commande par supervision des systèmes à événements discrets: application à un site expérimental l'Atelier Inter-établissement de Productique. Technical report, Laboratoire d'Automatique de Grenoble, Grenoble, France, 1994.

[8] E. W. Endsley, M. R. Lucas, and D. M. Tilbury. Modular design and verification of logic control for reconfigurable machining systems. Submitted to Discrete Event Dynamic Systems: Theory and Applications.

[9] Peyman Gohari-Moghadam. A linguistic framework for controlled hierarchical DES. Master's thesis, Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 1998.

[10] R. Leduc, M. Lawford, and W. Murray Wonham. Hierarchical interface based supervisory control: AIP example for parallel case. Technical Report No. 2, Software Quality Research Laboratory, Dept. of Computing and Software, McMaster University, Hamilton, ON, Canada, Nov 2001.

[11] R.J. Leduc, B.A. Brandin, and W. Murray Wonham. Hierarchical interface-based non-blocking verification. In *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, pages 1–6, May 2000.

[12] R.J. Leduc, B.A. Brandin, W. Murray Wonham, and M. Lawford. Hierarchical interface-based supervisory control: Serial case. In *Proc. of 40th Conf. Decision Contr.*, pages 4116–4121, Orlando, USA, December 2001.

[13] Ryan Leduc. PLC implementation of a DES supervisor for a manufacturing testbed: An implementation perspective. Master's thesis, Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 1996.

[14] F. Lin and W. Murray Wonham. Decentralized control and coordination of discrete-event systems with partial observations. In *Proc. 27th IEEE Conf. Decision Contr.*, pages 1125–1130, Dec 1988.

[15] Hong Liu, Jun-Cheol Park, and Raymond E. Miller. On hybrid synthesis for hierarchical structured petri nets. Technical report, Department of Computer Science, University of Maryland, College Park, MD, 1996.

[16] Chuan Ma. A computational approach to top-down hierarchical supervisory control of DES. Master's thesis, Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 1999.

[17] D. L. Parnas. Use of abstract interfaces in the development of software for embedded computer systems. NRL Report 8047, Naval Research Laboratory, 1977.

[18] David L. Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, December:1053–1058, December 1972.

[19] David Lorge Parnas, Paul C. Clements, and David M. Weiss. The modular structure of complex systems. *IEEE Transactions on Software Engineering*, SE-11(3):259–66, March 1985.

[20] M.H. de Queiroz and J.E.R. Cury. Modular supervisory control of large scale discrete event systems. In *Proceedings of WODES 2000*, pages 103–110, Ghent, Belgium, Aug 2000.

[21] P. Ramadge and W. Murray Wonham. Supervisory control of a class of discrete-event processes. *SIAM J. Control Optim*, 25(1):206–230, 1987.

[22] Karen Rudie and W. Murray Wonham. Think globally, act locally: decentralized supervisory control. *IEEE Trans. on Automatic Control*, 37(11):1692–1708, Nov 1992. Reprinted in F.A. Sadjadi (Ed.), Selected Papers on Sensor and Data Fusion, 1996; ISBN 0-8194-2265-7.

[23] Bing Wang. Top-down design for RW supervisory control theory. Master's thesis, Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 1995.

[24] K.C. Wong. *Discrete-Event Control Architecture: An Algebraic Approach*. PhD thesis, Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 1994.

[25] W. Murray Wonham. *Notes on Control of Discrete-Event Systems*. Department of Electrical and Computer Engineering, University of Toronto, 1999. Notes and CTCT software can be downloaded at http://odin.control.toronto.edu/DES/.

[26] W. Murray Wonham and P. Ramadge. On the supremal controllable sublanguage of a given language. *SIAM J. Control Optim*, 25(3):637–659, 1987.

[27] T. Yoo and S. Lafortune. A general architecture for decentralized supervisory control of discrete-event systems. In *Proc. of WODES 2000*, pages 111–118, Ghent, Belgium, Aug 2000.

[28] Z.H. Zhang. Smart TCT: an efficient algorithm for supervisory control design. Master's thesis, Dept. of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 2001.

[29] Z.H. Zhang and W. Murray Wonham. STCT: an efficient algorithm for supervisory control design. In *Proc. of SCODES 2001*, INRIA, Paris, July 2001.

[30] H. Zhong and W. Murray Wonham. On the consistency of hierarchical supervision in discrete-event systems. *IEEE Trans. on Automatic Control*, 35(10):1125–1134, Oct 1990.

[31] Meng Chu Zhou, David T. Wang, and Israel Mayk. Using petri nets for object-oriented design of command and control systems. *International Journal of Intelligent Control and Systems*, 2(2):287–300, 1998.