Modular Completeness for Communication Closed Layers *

Maarten Fokkinga, Mannes Poel & Job Zwiers
University of Twente†‡

Abstract

The Communication Closed Layers law is shown to be modular complete for a model related to that of Mazurkiewicz. It is shown that in a *modular* style of program development the CCL rule cannot be derived from simpler ones. Within a non-modular set-up the CCL rule can be *derived* however from a simpler independence rule and an analog of the expansion rule for process algebras.

1 Introduction

In an earlier paper [JPZ91] a formulation of the principle of *communication closed layers* [EF82] by means of an algebraic rule was proposed. This *Communication Closed Layers law* (CCL) has been applied thereafter in the construction and verification of various algorithms and protocols [JZ92b, JZ92a, JPSZ91, Zwi91], among which are a version of the Two Phase Commit protocol [JZ92b] and a rather complicated minimum weight spanning tree algorithm by Gallager, Humblet and Spira [JZ92a, GHS83, SdR89, CG88]. The versatility of the CCL rule, especially within a modular style of program development, stems from the possibility to rewrite a distributed process into a so-called layered form which can often be analyzed by means of simpler techniques for sequential programs. Parallel composition is denoted $P \parallel Q$ and *layered composition*, being a weakened version of sequential composition, is denoted $P \bullet Q$. The use of a partial order model, and exploitation of *independence information* enable transformations of concurrent systems into more sequential versions *without the expense of introducing extra nondeterminism*. A similar methodology of program construction underlies work on ISTL (interleaving set temporal logic) [KP87, KP89, KP90, KP92]. The CCL law allows one to prove equalities between processes based on *independence information*. In general the rule states that whenever P is independent of S and moreover Q is independent of R then the following equality holds:

$$(P \bullet Q) \parallel (R \bullet S) \;=\; (P \parallel R) \bullet (Q \parallel S) \qquad \text{(CCL)}$$

The notion of independence relies on the particular model of concurrency under concern; e.g. for shared variable models P and S are said to be independent whenever they access

*Part of this work has been supported by Esprit/BRA Project 6021 (REACT).
†Department of Computer Science, P.O. Box 217, 7500 AE Enschede, The Netherlands. E-mail: {fokkinga,mpoel,zwiers}@cs.utwente.nl
‡The authors thank Wil Janssen for carefully reading the manuscript.

disjoint sets of shared variables, whereas for communication based models independence of P and S means that they communicate along disjoint sets of channels.

Note that the CCL law does not rely on the internal structure of P, Q, R and S. In fact, P, Q, R and S are best regarded as typed process variables where the typing information allows one to decide whether the processes denoted by variables are independent or not, but anymore detailed information is not available. This is important for a *modular* style of process derivation, as we will explain now. If P, Q, R and S in the CCL law are *instantiated* (i.e. substituted) by concrete process terms that are built up from atomic actions, then we show that the law can be derived from simpler axioms among which are an expansion law for parallelism, and the *independence axiom:*

$$P \bullet Q \; = \; P \parallel Q \; = \; Q \bullet P \text{ if } P \text{ is independent from } Q.$$

Yet, in order to show equality between $(P \bullet Q) \parallel (R \bullet S)$ and $(P \parallel R) \bullet (Q \parallel S)$ based on these simpler laws involves *transforming the internal structure* of P, Q, R and S. For modular reasoning this is exactly what one would like to avoid.

Modular completeness [Zwi89] refers to the following completeness property. Let "P *sat spec*" denote that process P satisfies some specification *spec*. (Simple) completeness of a proposed axiomatization means that whenever P *sat spec* is a *valid* formula, then it is also (formally) derivable. Now assume that $P(X_0, \ldots, X_n)$ is a process term containing free process variables $X_0, \ldots, X_n$. *Modular* completeness means that whenever a formula of the form

$$spec'(X_0, \ldots, X_n) \rightarrow spec(P(X_0, \ldots, X_n))$$

is valid, then it is formally derivable within the axiomatization under consideration. For the present context, the specifications on the left hand side ("$spec'(X_0, \ldots, X_n)$") take on the form of a list of independence relations between process variables, of the form $X_i \# X_j$. The right hand side has the form of an equality "$P(X_0, \ldots, X_n) \; = \; Q(X_0, \ldots, X_n)$".

The main result of this paper states that a few basic axioms together with the CCL law form a *modular complete* system. Moreover, we argue that such a completeness result is not possible with rules that are any simpler than the CCL rule. For instance, the following simplified versions of the CCL rule do not lead to a complete system:

Provided that P and S are independent and Q and R are independent:

$$\begin{array}{rcll} P \parallel S & = & P \bullet S & \text{(Independence)} \\ (P \bullet Q) \parallel S & = & P \bullet (Q \parallel S) & \text{(CCL-L)} \\ (P \bullet Q) \parallel R & = & (P \parallel R) \bullet Q & \text{(CCL-R)} \end{array}$$

In fact it can be shown that if a system has rules only in the form of equalities between process terms that have less than four variables, the CCL rule is not derivable, and therefore such a system cannot be complete.

The completeness result is with respect to a partial order model of concurrency, related to Mazurkiewicz' trace model [Maz89]. It differs from simple partial order models in that it is based on both an partial order relation and a separate concurrency relation.

For a comparison we also consider processes that are built up from concrete atomic actions, rather than from typed process variables. Within this setting the CCL law still makes sense as it allows for a modular style of reasoning. But a non-modular style that avoids the CCL rule becomes also a possibility. This can be achieved for instance by introducing an analog of the expansion law for CCS [Mil80] that allows one to replace parallel composition by layered composition and nondeterministic choice. A few basic axioms together with the analog of the expansion law and the simple "Independence" axiom mentioned above turn out to form already a complete axiomatization. Thus the CCL law is seen to be a *derived law* within this context.

Finally we mention here that when *no* knowledge concerning independence is taken into account or, equivalently, when we assume that no two process variables are independent, process equality is already axiomatized completely [Gis84] by only a few simple axioms concerning commutativity of $\|$ and associativity of $\|$ and $\bullet$.

2 Two process languages

We introduce two process languages that both are tiny subsets of the languages discussed in e.g. [JPZ91, JPZ93, Zwi91]. Their sole purpose is to have a framework to discuss our completeness results in later sections.

Let $(\mathcal{A}ct, \leftrightsquigarrow)$ be a given (countable) set of atomic actions $(a \in)\mathcal{A}ct$, together with a symmetric irreflexive *dependence relation* "$\leftrightsquigarrow$". (Within specific applications, such as database serializability theory, dependency between transactions is also called *conflict*. The term conflict relation instead of dependence relation is also used in [JPZ91, JPZ93, Zwi91].) In many cases we use the *in*dependence relation "$\sharp$", which is just the complement of "$\leftrightsquigarrow$". An *alphabet* α is defined here as a subset of $\mathcal{A}ct$. Two alphabets α and β are independent iff all actions $a \in \alpha$ are independent of all $b \in \beta$. We assume that there is a set $(X_\alpha \in)\mathcal{P}var$ of *process variables,* each typed by means of some alphabet α. The alphabet $\alpha(P)$ of process terms P is defined as the set of all actions occurring in P. Two processes P and Q or two process variables X_α and X_β are said to be independent iff their associated alphabets are independent. This is denoted by $P \sharp Q$ (i.e. not $P \leftrightsquigarrow Q$) and $X_\alpha \sharp X_\beta$, respectively. The two languages L_{pvar} and L_{atomic} are defined thus:

$$P \in L_{pvar},$$

$$P ::= X_\alpha \quad | \quad P \bullet Q \quad | \quad P \parallel Q \quad | \quad \mathbf{skip}$$

$$P \in L_{atomic},$$

$$P ::= a \quad | \quad P \bullet Q \quad | \quad P \parallel Q \quad | \quad P \;\mathbf{or}\; Q \quad | \quad \mathbf{skip} \quad | \quad \mathbf{empty}$$

A partial order model for L_{atomic} is given below in section 3.3. Here we provide some intuition.

A basic assumption is that a (single) computation or *run* of a system can be modeled as a partially ordered multiset (pomset) of actions. (The usual distinction between actions and events, i.e., instances of actions, need not be made here). Actions within a given run remain unordered iff they are independent. Two processes P and Q are regarded as equal, denoted by $P = Q$, iff their sets of pomsets are equal.

Parallel composition $P \parallel Q$ executes P and Q with dependent actions of P and Q ordered nondeterministically. Thus $P \parallel Q$ denotes a set of runs. The nondeterministic choices for different pairs of dependent actions are of course subject to the condition that the order must be acyclic.

For *layer composition* $P \bullet Q$ the situation is somewhat like parallel composition, the difference being that when a P action a and a Q action b are dependent then the a can only precede b. (For parallel composition b could also precede a.) Layer composition should be distinguished from classical sequential composition $P\,\mathbf{;}\,Q$, which would require that all P actions precede all Q actions, irrespective of the dependence relation. One could view $P\,\mathbf{;}\,Q$ as a degenerate case where the independence relation is *empty*, i.e. no two actions are independent.

The process **skip** performs no action and acts as the unit element for layer and parallel composition.

Nondeterministic choice P **or** Q is a straightforward construct that executes either P or Q. The related process **empty** acts as the unit for the choice construct and as "zero" for parallel and layer composition. We use the abbreviation $\Sigma_{i \in I}\, P_i$ for choices of the form P_{i_0} **or** P_{i_1} **or** $\cdots$, where $i_j \in I$.

3 Axiomatization

In this section we present some groups of axioms for the two process languages L_{pvar} and L_{atomic} and give sound models for both.

3.1 Axioms for L_{pvar}

Axioms A1

$$\begin{array}{rcll}
P \parallel Q & = & Q \parallel P & \text{(COM1)} \\
P \parallel (Q \parallel R) & = & (P \parallel Q) \parallel R & \text{(ASSOC1)} \\
P \bullet (Q \bullet R) & = & (P \bullet Q) \bullet R & \text{(ASSOC2)} \\
P \parallel \mathbf{skip} & = & P & \text{(SKIP1)} \\
P \bullet \mathbf{skip} & = & P & \text{(SKIP2)} \\
\mathbf{skip} \bullet P & = & P & \text{(SKIP3)}
\end{array}$$

Axioms A2
Provided that $P \sharp S$, and $Q \sharp R$:

$$(P \bullet Q) \parallel (R \bullet S) \quad = \quad (P \parallel R) \bullet (Q \parallel S) \quad \text{(CCL)}$$

Remark.
From the Communication Closed Layers law (CCL) and the axioms for **skip** from **A1**, the following laws can be derived:
Provided that $P \sharp S$, and $Q \sharp R$:

$$\begin{array}{rcll}
(P \bullet Q) \parallel S & = & P \bullet (Q \parallel S) & \text{(CCL-L)} \\
(P \bullet Q) \parallel R & = & (P \parallel R) \bullet Q & \text{(CCL-R)} \\
P \parallel S & = & P \bullet S & \text{(Independence)}
\end{array}$$

(End of remark)

3.2 Axioms for L_{atomic}

Axioms A3

$$
\begin{array}{rcll}
P \textbf{ or } Q & = & Q \textbf{ or } P & \text{(COM2)} \\
P \textbf{ or } (Q \textbf{ or } R) & = & (P \textbf{ or } Q) \textbf{ or } R & \text{(ASSOC3)} \\
P \textbf{ or } P & = & P & \text{(IDEMPOT)} \\
P \parallel (Q \textbf{ or } R) & = & (P \parallel Q) \textbf{ or } (P \parallel R) & \text{(DIST1)} \\
P \bullet (Q \textbf{ or } R) & = & (P \bullet Q) \textbf{ or } (P \bullet R) & \text{(DIST2)} \\
(P \textbf{ or } Q) \bullet R & = & (P \bullet R) \textbf{ or } (Q \bullet R) & \text{(DIST3)} \\
P \textbf{ or } \textbf{empty} & = & P & \text{(EMPTY1)} \\
P \parallel \textbf{empty} & = & \textbf{empty} & \text{(EMPTY2)} \\
P \bullet \textbf{empty} & = & \textbf{empty} & \text{(EMPTY3)} \\
\textbf{empty} \bullet P & = & \textbf{empty} & \text{(EMPTY4)}
\end{array}
$$

In order to formulate the expansion law for L_{atomic} we introduce the notion of (syntactic) traces:

Definition 3.1 *Syntactic Traces*
A (syntactic) trace is a process term of the form $a_1 \bullet a_2 \bullet \cdots \bullet a_n$ where all the a_i are atomic actions. The case $n = 0$ is allowed, and is identified with the **skip** process. □

Definition 3.2 *Minimal Elements*
Let $t = t_1 \bullet \cdots \bullet t_n$ be a trace. The set of minimal elements of t is:

$$Minimal(t) \stackrel{\text{def}}{=} \{t_i \mid \forall j < i,\ t_j \sharp t_i\}$$

Let $t^{(i)}$ denote the trace t with t_i removed, i.e.

$$t^{(i)} = t_1 \bullet \cdots \bullet t_{i-1} \bullet t_{i+1} \bullet \cdots \bullet t_n \quad (1 \leq i \leq n)$$

□

Axioms A4
Let $P = p_1 \bullet p_2 \bullet \cdots \bullet p_n$ and $Q = q_1 \bullet q_2 \bullet \cdots \bullet q_m$ be (syntactic) traces, where $n, m \geq 1$.

$$
\begin{array}{rcl}
P \parallel Q & = & \Sigma_{p_i \in Minimal(P)}\ p_i \bullet (P^{(i)} \parallel Q) \\
& & \textbf{or} \\
& & \Sigma_{q_i \in Minimal(Q)}\ q_i \bullet (P \parallel Q^{(i)})
\end{array}
$$

Provided that for actions a and b, $a \sharp b$

$$a \parallel b \quad = \quad b \bullet a \quad \text{(Act-Independence)}$$

Remark.

- Laws like the Independence axiom for processes (rather than for actions), or the CCL laws are all valid for the model for L_{atomic} that we discuss below. Here we have included the minimum for the completeness result.

- Note that for the degenerated case where we have an *empty* independence relation, the above expansion law boils down to an expansion law for interleaving semantics. For in that case, only p_1 and q_1 are minimal elements.

- If a and b are independent actions, i.e. $a \sharp b$, then such actions can be interchanged:

$$a \bullet b = b \bullet a$$

- The expansion axiom presented here is a slight simplification of a similar law in [PZ92].

(End of remark)

3.3 The Partial Order Model for L_{atomic}

We construct a model for the language L_{atomic} for which all of the axioms **(A1), (A2), (A3)** and **(A4)** are valid. It is shown in the corresponding report, [PZ93b], that axioms **(A1), (A3)** and **(A4)** form a *complete* system with respect to this model.

A *partially ordered action set* is defined as a structure $(E, \longrightarrow)$ where E is a multiset of actions, and where $\longrightarrow$ is an irreflexive partial order on E. A partially ordered action multiset $(E, \longrightarrow)$ is *conflict closed* if for any two $a_0, a_1 \in E$ it is the case that they are ordered if they are dependent:

$$a_0 \longrightarrow a_1 \text{ or } a_1 \longrightarrow a_0 \text{ if } a_0 \leftrightsquigarrow a_1.$$

A *run* or *computation* is defined to be a partially ordered action set that is conflict closed. The set of all such runs is denoted by $\mathcal{PO}_0$.

Definition 3.3 *Semantic operators*
Let as usual R^+ denote the *transitive closure* of binary relation R.
For runs $H_0 = (E_0, \longrightarrow_0)$ and $H_1 = (E_1, \longrightarrow_1)$ in $\mathcal{PO}_0$ define

- the parallel composition by

$$\begin{aligned} H_0 \parallel H_1 = \{ & H \in \mathcal{PO}_0 \mid H = (E_0 \cup E_1, (\longrightarrow_0 \cup \longrightarrow_1 \cup \longrightarrow_C)^+), \text{ where} \\ & \longrightarrow_C \subseteq ((E_0 \times E_1) \cup (E_1 \times E_0)) \text{ such that} \\ & ((e,e') \in \longrightarrow_C \vee (e',e) \in \longrightarrow_C) \text{ iff } e \leftrightsquigarrow e' \} \end{aligned}$$

- and layer composition by

$$\begin{aligned} H_0 \bullet H_1 = \{ & H \in \mathcal{PO}_0 \mid H = (E_0 \cup E_1, (\longrightarrow_0 \cup \longrightarrow_1 \cup \longrightarrow_C)^+), \text{ where} \\ & \longrightarrow_C \subseteq (E_0 \times E_1) \text{ such that } ((e,e') \in \longrightarrow_C) \text{ iff } e \leftrightsquigarrow e' \} \end{aligned}$$

  Observe that $H_0 \bullet H_1$ consists of a single run.

□

All unions in the above definition should be understood as disjoint unions. Note that the only difference between parallel and layered composition is in the restrictions on $\longrightarrow_C$, hence $H_0 \bullet H_1 \subseteq H_0 \parallel H_1$. We define the semantics of the language L_{atomic} by means of a semantic meaning function of the form

$$[\![\cdot]\!]_{at} : L_{atomic} \rightarrow \mathcal{P}(\mathcal{PO}_0).$$

Definition 3.4 *Semantics of L_{atomic}*

$$[\![a]\!]_{at} = \{(a, \emptyset)\}$$

$$[\![P \parallel Q]\!]_{at} = \bigcup \{H_P \parallel H_Q \mid H_P \in [\![P]\!]_{at}, H_Q \in [\![Q]\!]_{at}\}$$

$$
\begin{aligned}
[\![P \bullet Q]\!]_{at} &= \bigcup \{H_P \bullet H_Q \mid H_P \in [\![P]\!]_{at},\, H_Q \in [\![Q]\!]_{at}\} \\
[\![P \textbf{ or } Q]\!]_{at} &= [\![P]\!]_{at} \cup [\![Q]\!]_{at} \\
[\![\ \textbf{skip}\ ]\!]_{at} &= \{(\emptyset, \emptyset)\} \\
[\![\ \textbf{empty}\ ]\!]_{at} &= \emptyset
\end{aligned}
$$

□

Theorem 3.5 *Soundness*
The axioms **(A1), (A2), (A3)** and **(A4)** are sound with respect to the partial order model defined above. □

The proof is omitted. Soundness of these axioms for a related (more complicated) model can be found in an earlier paper, [JPZ93].

Theorem 3.6 *Completeness*
The axioms **(A1)**, **(A3)**, and **(A4)** form a complete system for L_{atomic}. □

A proof of the completeness theorem can be found in [PZ93b]. It is shown in [JPZ91, PZ92, PZ93a] that the CCL law holds for (a slightly more complicated model than) L_{atomic}. Hence

Corollary 3.7
The CCL law is a derived law in the model L_{atomic} □

3.4 The Graph Model for L_{pvar}

We present a more complicated partial order model for the L_{pvar} language. Runs in the form of pomsets of *actions* don't suffice here, as the basic terms in L_{pvar} are variables, not actions. Rather we use runs that take on the form of *graphs* $G = (V, \rightarrow, —)$ with two kind of arcs, namely the *concurrency* arc — and the *precedence* arc $\rightarrow$. Formally, $\rightarrow$ is an irreflexive partial order on the set V of vertices, whereas the — is just a symmetric binary relation on V. The precedence relation $\rightarrow$ has the same role as in the simple model of the previous section. We shall now explain the reason for introducing the concurrency relation —.

The intended meaning of $X_\alpha \rightarrow Y_\beta$ and $X_\alpha — Y_\beta$ is as follows. Let a, b be dependent actions. Then after refining the variables into concrete processes with action a occurring in X_α and b in Y_β, we shall have $a \rightarrow b$ in case of $X_\alpha \rightarrow Y_\beta$, but either $a \rightarrow b$ or $b \rightarrow a$ in case of $X_\alpha — Y_\beta$. So, if both a, a' eventually occur in place of X_α, and b, b' in place of Y_β, then in the case of $X_\alpha — Y_\beta$ we may have $a \rightarrow b$ and $b' \rightarrow a'$ at the same time. This is not expressible by either $X_\alpha \rightarrow Y_\beta$ nor $Y_\beta \rightarrow X_\alpha$.

We will now give the semantics for L_{pvar} terms. P. Informally the semantics $[\![P]\!]_[var$ is the graph where the vertices V is the set of variables occurring in P. The binary relation $\rightarrow$ on V is defined as follows: $X_\alpha \rightarrow Y_\beta$ if and only if $X_\alpha \rightsquigarrow Y_\beta$ and there is a syntactic subexpression $P_1 \bullet P_2$ of P such that X_α is a variable of P_1 and Y_β is a variable of P_2. Similarly the symmetric binary relation — on V is defined as: $X_\alpha — Y_\beta$ if and only if $X_\alpha \leftrightsquigarrow Y_\beta$ and there is a syntactic subexpression $P_1 \parallel P_2$ of P such that X_α is a variable of P_1 and Y_β is a variable of P_2, or X_α in P_2 and Y_β in P_1.

Formally the semantics is given as follows. For two graphs G_Q and G_P define

$$G_Q \parallel G_R \stackrel{\text{def}}{=} (V_Q \cup V_R, \rightarrow_Q \cup \rightarrow_R, —_Q \cup —_R \cup —_{\parallel})$$

where

$$—_{\parallel} \stackrel{\text{def}}{=} \{(S,T) \mid ((S \in V_Q \wedge T \in V_R) \vee (S \in V_R \wedge T \in V_Q)) \ \wedge \ S \leftrightsquigarrow T\}$$

and

$$G_Q \bullet G_R \stackrel{\text{def}}{=} (V_Q \cup V_R, \rightarrow_Q \cup \rightarrow_R \cup \rightarrow_{\bullet}, —_Q \cup —_R)$$

where

$$\rightarrow_{\bullet} \stackrel{\text{def}}{=} \{(S,T) \mid (S \in V_Q \wedge T \in V_R) \ \wedge \ S \leftrightsquigarrow T\}$$

Then

$$\begin{aligned}
[\![X_\alpha]\!]_{var} &= (X_\alpha, \emptyset, \emptyset) \\
[\![P \parallel Q]\!]_{var} &= [\![P]\!]_{var} \parallel [\![Q]\!]_{var} \\
[\![P \bullet Q]\!]_{var} &= [\![P]\!]_{var} \bullet [\![Q]\!]_{var} \\
[\![\ \mathbf{skip}\ ]\!]_{var} &= (\emptyset, \emptyset, \emptyset)
\end{aligned}$$

3.4.1 Relation between the partial order model and the graph model

Let $P \in L_{atomic}$ with alphabet α and $Q \in L_{atomic}$ with alphabet β, and let $H_P \stackrel{\text{def}}{=} (E_P, \rightarrow_P) \in [\![P]\!]_{at}$ and $H_Q \stackrel{\text{def}}{=} (E_Q, \rightarrow_Q) \in [\![Q]\!]_{at}$ Observe that H_P and H_Q can be seen as graphs in the above sense, with empty $—$ relation. Thus consider H_P and H_Q as graphs, say $\mathcal{H}_P$ and $\mathcal{H}_Q$ respectively. Then $\mathcal{H}_P \parallel \mathcal{H}_Q$ is the graph $(E_P \cup E_Q, \rightarrow_P \cup \rightarrow_Q, —_{PQ})$ where the set of vertices $E_P \cup E_Q$ consists of atomic actions, and for each pair of atomic actions a, b

$$(a,b) \in —_{PQ} \ \text{ iff } \ (a \leftrightsquigarrow b) \ \wedge \ (a \in E_P \ \leftrightarrow \ b \in E_Q)$$

In other words, $(a,b) \in —_{PQ}$ if they are dependent and one them is contained in E_P and the other in E_Q. The intuitive meaning of $a—b$ is that in actual run a must precede b, i.e. $a \rightarrow b$, or conversely b must precede a, i.e. $b \rightarrow a$. Thus in order to transform the graph into an actual run one must make for each pair of atomic actions a, b with $\{(a,b),(b,a)\} \subseteq —_{PQ}$ a choice, namely $a \rightarrow b$ or $b \rightarrow a$. If one transforms each $a—b$ into $a \rightarrow b$ or $b \rightarrow$ one gets a graph of the form.

$$(E_P \cup E_Q, \rightarrow_P \cup \rightarrow_Q \cup \rightarrow_C, \emptyset)$$

where

$$\rightarrow_C \ \subseteq \ (E_P \times E_Q) \cup (E_Q \times E_P) \text{ such that } ((a,b) \in \rightarrow_C \ \vee \ (b,a) \in \rightarrow_C) \text{ iff } a \leftrightsquigarrow b$$

Observe that the above formula closely resembles the definition of parallel composition in Definition 3.3. If $\rightarrow_P \cup \rightarrow_Q \cup \rightarrow_C$ is acyclic then

$$(E_P \cup E_Q, (\rightarrow_P \cup \rightarrow_Q \cup \rightarrow_C)^+)$$

is a run of $H_P \parallel H_Q$. In the above construction there was choice, namely transform $a—b$ to $a \rightarrow b$ or $b \rightarrow a$. If one takes the union over all such choices the above procedure will generate exactly $H_P \parallel H_Q$.
Hence if one applies the above transformation to all $H_P \in [\![P]\!]_{at}$, and all $H_Q \in [\![Q]\!]_{at}$ one precisely obtains all runs in $[\![P \parallel Q]\!]_{at}$.

For $\mathcal{H}_P \bullet \mathcal{H}_Q$ the situation is not so complex

$$\mathcal{H}_P \bullet \mathcal{H}_Q = (E_P \cup E_Q, \rightarrow_P \cup \rightarrow_Q \cup \rightarrow_C, \emptyset)$$

where

$$\longrightarrow_C \;\subseteq\; (E_P \times E_Q) \text{ such that } ((a,b) \in \longrightarrow_C \text{ iff } a \rightsquigarrow b)$$

Hence

$$H_P \bullet H_Q = \{(E_P \;\cup\; E_Q, (\rightarrow_P \;\cup\; \rightarrow_Q \;\cup\; \rightarrow_C)^+)\}$$

and the union over all $H_P \in [\![P]\!]_{at}$, and $H_Q \in [\![Q]\!]_{at}$ in the above construction will precisely generate all runs in $[\![P \bullet Q]\!]_{at}$.

Thus in general the graph model can be seen as an intermediate stage in computing the partial order semantics of a process term in L_{atomic}.

3.5 Soundness

Theorem 3.8 *Soundness*
The graph model is a model for axioms **(A1), (A2)**. □

Proof. We will only sketch the soundness proof for the CCL axiom. First consider the case where P, Q, R and S are process variables. Then an easy calculation shows that both $(P \bullet Q) \;\|\; (R \bullet S)$ and $(P \;\|\; R) \bullet (Q \;\|\; S)$ denote the graph from figure 1. Next, assume

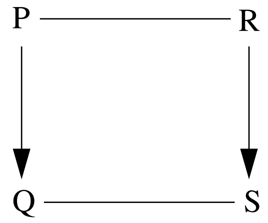


Figure 1: The graph of both $(P \bullet Q) \;\|\; (R \bullet S)$ and $(P \;\|\; R) \bullet (Q \;\|\; S)$

that P, Q, R and S are arbitrary process terms, not just variables. In that case the graph in figure 1 can be obtained as follows. First compute the graphs of P, Q, R and S. Then connect each vertex of P with — to each dependent vertex of R . Similarly for Q and S. Finally connect each vertex of P with a $\rightarrow$ to each dependent vertex of Q . Similarly for R and S. From this construction it is seen that the graphs for $(P \bullet Q) \;\|\; (R \bullet S)$ and $(P \;\|\; R) \bullet (Q \;\|\; S)$ are, again, the same. □

The next section is dedicated to the proof of

Theorem 3.9 *Modular Completeness*
The axioms **(A1)** and **(A2)** form a *complete* system for L_{pvar} □

3.6 Completeness of the CCL Rule

The aim of this section is to prove that the axioms **(A1)** together with **(A2)** (the CCL law) are modular complete. More precisely we will show that when two syntactic terms P and P' represent the same graph $\mathcal{G}$ in the graph model, P can be transformed into P', using only the axioms **(A1)** and **(A2)**.

3.6.1 Notational conventions

Letters P, Q, R range over terms in L_{pvar}, and $\mathcal{P}, \mathcal{Q}, \mathcal{R}$ range over graphs. Whenever a term is called P, its graph is called $\mathcal{P}$. Conversely, if there is a term denoting graph $\mathcal{P}$, then the term is called P. Similarly for $Q, \mathcal{Q}$ and $R, \mathcal{R}$. Graph $\mathcal{S}kip$ is the graph denoted by *skip*, and $\mathcal{X}$ is the graph denoted by X.

For graphs $\mathcal{P}, \mathcal{Q}$ the expression $\mathcal{P} \cap \mathcal{Q}$ denotes the graph whose vertex set is the intersection of those of $\mathcal{P}$ and $\mathcal{Q}$, and for which there is an $\rightarrow / \text{—}$-arc between two vertices if, and only if, there is an $\rightarrow / \text{—}$-arc between those vertices in both $\mathcal{P}$ and $\mathcal{Q}$.

Definition 3.10 *Representable Graph*
A graph $\mathcal{P} = (V, \rightarrow, \text{—})$ is representable if and only if there exists a term $P \in L_{pvar}$ such that $\mathcal{P}$ is the denotation of P. P is called a (syntactic) representation of $\mathcal{P}$. $\square$

Not all graphs are representable. This was observed for a related class of graphs (so-called series-parallel graphs) by Valdes, Tarjan and Lawler [VTL82]. As an example, adapted to our framework, consider the so-called "N-graph" in figure 2, which is not representable as a L_{pvar} term.

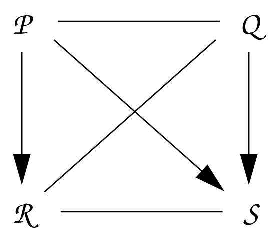


Figure 2: A non-representable graph "$\mathcal{N}$"

Lemma 3.11
Let $\mathcal{G} = (V, \rightarrow, \text{—})$ be a representable graph. Then $\rightarrow$ is acyclic and

$$\rightarrow \cap \text{ — } = \emptyset$$

$\square$

The proof can be given by induction on the structure of a representation of $\mathcal{G}$.

As an example consider the graph $\mathcal{G}$ in figure 3.

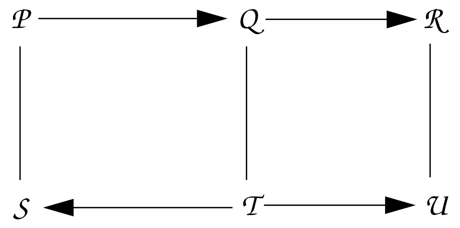


Figure 3: The graph $\mathcal{G}$

Assume that $\mathcal{P}, \mathcal{Q}, \mathcal{R}, \mathcal{S}, \mathcal{T}$ and $\mathcal{U}$ are representable, say by P, Q, R, S, T and U respectively, and that there are no dependencies between processes where there is no $\rightarrow$ or — edge in

$\mathcal{G}$, such as for example between P and T, or between S and U, then $\mathcal{G}$ can be represented as a parallel composition of the form:

$$(P \bullet Q \bullet R) \parallel (T \bullet (S \parallel U)).$$

But $\mathcal{G}$ has also a representation with layered composition as top operator:

$$((P \bullet Q) \parallel (T \bullet S)) \bullet (R \parallel U)$$

This layered representation can be transformed into the parallel version by applying the CCL law, the ASSOC2 law, and the (from the CCL law derived) Independence law. As has become clear from this example, a (representable) graph can have several representations which, so we claim, can be transformed into each other.

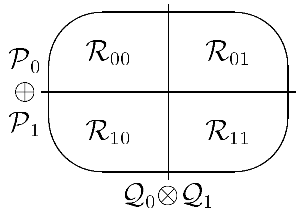
3.6.2 Completeness

Let us first introduce some more notation. Symbol $\stackrel{def}{=}$ denotes equality by syntactic definition, symbol $\stackrel{ax}{=}$ denotes axiomatically provable equality using the axioms **(A1), (A2)** only, and symbol $=$ denotes semantic (graph) equality.

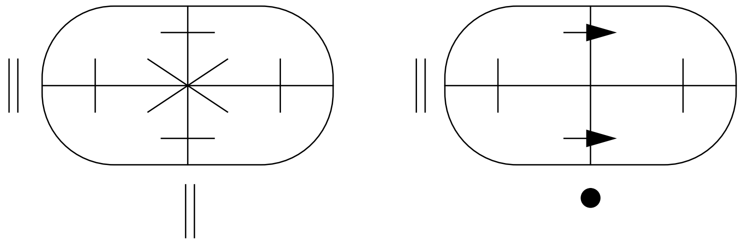
Throughout the sequel $\oplus$ and $\otimes$ range over $\{\parallel, \bullet\}$ (both the semantic graph operations and the syntactic operation symbols), and indices i, j range over $\{0, 1\}$. In the Decomposition Lemma and the Term Existence Lemma we use i, j exclusively in the names $\mathcal{P}_i, \mathcal{Q}_j, \mathcal{R}_{ij}$; in the latter the i relates to $\mathcal{P}_i$ and the j relates to $\mathcal{Q}_j$.

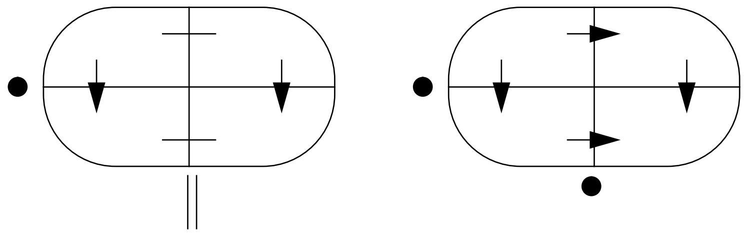
The Decomposition Lemma tells that various graph decompositions *abide* with each other, and that such abidances can also be proved axiomatically. The Term Existence Lemma tells that parallel and layer decompositions of denotable graphs can also be denoted and proved axiomatically.

Lemma 3.12 *Decomposition*
Let $\oplus, \otimes$ be arbitrary in $\{\parallel, \bullet\}$. Let $\mathcal{P}_i, \mathcal{Q}_j$ be graphs such that $\mathcal{P}_0 \oplus \mathcal{P}_1 = \mathcal{Q}_0 \otimes \mathcal{Q}_1$. Define $\mathcal{R}_{ij} = \mathcal{P}_i \cap \mathcal{Q}_j$.



Then, first, for vertices taken from different $\mathcal{R}_{ij}$, an arc between the vertices can only be of the type as indicated in the figure below, meaning also that no arc exists if no one is indicated:





Second, an arc between the vertices exists iff the vertices are dependent.
Third, the $\mathcal{P}_i, \mathcal{Q}_j$ can be decomposed as $\mathcal{P}_i = \mathcal{R}_{i0} \otimes \mathcal{R}_{i1}$ and $\mathcal{Q}_j = \mathcal{R}_{0j} \oplus \mathcal{R}_{1j}$. By the way, this together with $\mathcal{P}_0 \oplus \mathcal{P}_1 \quad = \quad \mathcal{Q}_0 \otimes \mathcal{Q}_1$ gives:

$$(\mathcal{R}_{00} \otimes \mathcal{R}_{01}) \oplus (\mathcal{R}_{10} \otimes \mathcal{R}_{11}) \quad = \quad (\mathcal{R}_{00} \oplus \mathcal{R}_{10}) \otimes (\mathcal{R}_{01} \oplus \mathcal{R}_{11}).$$

Fourth, if R_{ij} are terms that denote the $\mathcal{R}_{ij}$, then:

$$(R_{00} \otimes R_{01}) \oplus (R_{10} \otimes R_{11}) \quad \stackrel{ax}{=} \quad (R_{00} \oplus R_{10}) \otimes (R_{01} \oplus R_{11}).$$

□

Proof.
The first three claims are immediate by the definition of the graph operations $\|$ and $\bullet$, and the wel-formedness constraints. For the fourth claim we distinguish between the four choices for $\oplus, \otimes$:

Case $\oplus, \otimes = \|, \|$. Associativity and commutativity of $\|$ suffice; these are expressed by axiom **(A1)**.

Case $\oplus, \otimes = \|, \bullet$ or $\bullet, \|$. Since R_{ij} denotes $\mathcal{R}_{ij}$, we conclude from the definition of $\mathcal{R}_{ij}$ and the first part of the lemma that the independence condition of the CCL law is satisfied. So one application of axiom **(A2)** does the job.

Case $\oplus, \otimes = \bullet, \bullet$. Since R_{ij} denotes $\mathcal{R}_{ij}$, we conclude from the definition of $\mathcal{R}_{ij}$ and the first part of the lemma that R_{01} and R_{10} are independent. Now associativity of $\bullet$ (expressed by axiom **(A1)**) and the Independence Law (derivable from the axioms **(A1,A2)**) suffice. □

Lemma 3.13 *Term Existence*
Let P be a term, and $\mathcal{P}$ its graph. Let $\oplus$ be parallel or layered composition.

1. Suppose that $\mathcal{P} = \mathcal{S}kip$ or $\mathcal{P} = \mathcal{X}$. Then $P \stackrel{ax}{=} skip$ or $P \stackrel{ax}{=} X$, respectively.
2. Suppose there exist graphs $\mathcal{P}_i$ such that $\mathcal{P} = \mathcal{P}_0 \oplus \mathcal{P}_1$. Then there exist terms P_i denoting $\mathcal{P}_i$ such that $P \stackrel{ax}{=} P_0 \oplus P_1$.

□

Proof.
By induction on the structure of P. Part 1 is almost trivial. We apply case analysis on the syntactic form of P:

Case $P \stackrel{def}{=} skip$. Then apparently $\mathcal{P} = \mathcal{S}kip$, and trivially $P \stackrel{ax}{=} skip$ by reflexivity of axiomatic equality.

Case $P \stackrel{def}{=} Y$. Then apparently $X \stackrel{def}{=} Y$ and $\mathcal{P} = \mathcal{X}$, and again trivially $P \stackrel{ax}{=} X$.

Case $P \stackrel{def}{=} Q_0 \oplus Q_1$, where $\oplus$ is either $\|$ or $\bullet$. Now one of Q_j, say Q_0, denotes $\mathcal{S}kip$, and the other one, Q_1, denotes $\mathcal{P}$. By induction $Q_0 \stackrel{ax}{=} skip$, and $Q_1 \stackrel{ax}{=} skip$ or $\stackrel{ax}{=} X$. So by neutrality of *skip* for $\oplus$ (expressed by **(A1)**) we have either:

$$\begin{array}{lllllll} P & \stackrel{def}{=} & Q_0 \oplus Q_1 & \stackrel{ax}{=} & skip \oplus skip & \stackrel{ax}{=} & skip, \quad \text{or:} \\ P & \stackrel{def}{=} & Q_0 \oplus Q_1 & \stackrel{ax}{=} & skip \oplus X & \stackrel{ax}{=} & X. \end{array}$$

The proof of Part 2 is more interesting. Again, we apply case analysis on the syntactic form of P:

Case $P \stackrel{def}{=} skip$ or $P \stackrel{def}{=} X$. Then at least one of $\mathcal{P}_i$, say $\mathcal{P}_0$, is $\mathcal{S}kip$, and the other, $\mathcal{P}_1$, is $\mathcal{P}$. So take $P_0 \stackrel{def}{=} skip$ and $P_1 \stackrel{def}{=} P$. Then P_i denotes $\mathcal{P}_i$, and by neutrality of *skip* for $\oplus$ we have:

$$P \quad \stackrel{ax}{=} \quad skip \oplus P \quad \stackrel{ax}{=} \quad P_0 \oplus P_1.$$

Case $P \stackrel{def}{=} Q_0 \otimes Q_1$ where $\otimes$ is one of $\|$ or $\bullet$. Let $\mathcal{Q}_j$ be the graph denoted by Q_j. Put $\mathcal{R}_{ij} = \mathcal{P}_i \cap \mathcal{Q}_j$. By the Decomposition Lemma 3.12 we have:

$$\begin{array}{lll} (\mathbf{a0}) & \mathcal{P}_i = \mathcal{R}_{i0} \otimes \mathcal{R}_{i1} \\ (\mathbf{a1}) & \mathcal{Q}_j = \mathcal{R}_{0j} \oplus \mathcal{R}_{1j} \\ (\mathbf{a2}) & (R_{00} \oplus R_{10}) \otimes (R_{01} \oplus R_{11}) \stackrel{ax}{=} (R_{00} \otimes R_{01}) \oplus (R_{10} \otimes R_{11}), \end{array}$$

for all terms R_{ij} that happen to denote the graphs $\mathcal{R}_{ij}$. By induction (applied to terms Q_j with graph decomposition (a1)) there exist:

(**b**) terms R_{0j}, R_{1j} denoting $\mathcal{R}_{0j}, \mathcal{R}_{1j}$, such that

(**c**) $Q_j \stackrel{ax}{=} R_{0j} \oplus R_{1j}$.

Now define the required terms P_i by:

(**d**) $P_i \stackrel{def}{=} R_{i0} \otimes R_{i1}$.

It remains to verify the two claims about the P_i. First, term P_i denotes $\mathcal{P}_i$:

$$\begin{array}{ll} & P_i \\ \stackrel{def}{=} & \quad \{ \text{(d): definition } P_i \} \\ & R_{i0} \otimes R_{i1} \\ = & \quad \{ \text{(b): definition } R_{ij} \} \\ & \mathcal{R}_{i0} \otimes \mathcal{R}_{i1} \\ = & \quad \{ \text{(a0): Decomposition Lemma 3.12} \} \\ & \mathcal{P}_i \end{array}$$

Second, the equality $P = P_0 \oplus P_1$ is axiomatically provable:

$$\begin{array}{ll} & P \\ \stackrel{def}{=} & \quad \{ \text{case assumption} \} \\ & Q_0 \otimes Q_1 \\ \stackrel{ax}{=} & \quad \{ \text{(c): induction} \} \end{array}$$

$$
\begin{aligned}
& (R_{00} \oplus R_{10}) \otimes (R_{01} \oplus R_{11}) \\
\overset{ax}{=} \quad & \{ \text{ (b,a2): Decomposition Lemma 3.12 } \} \\
& (R_{00} \otimes R_{01}) \oplus (R_{10} \otimes R_{11}) \\
\overset{def}{=} \quad & \{ \text{ (e): definition } P_i \} \\
& P_0 \oplus P_1 .
\end{aligned}
$$

This completes the proof of Part 2, hence of the entire Term Existence Lemma. □

One may notice that precisely all axioms in **(A1)**, **(A2)** have been used in the proofs of the two lemmas. Now we can state and proof the main result.

Theorem 3.14 *Modular Completeness*
For arbitrary terms P, Q we have: $P = Q \quad \Longrightarrow \quad P \overset{ax}{=} Q$. □

As a corollary we obtain the following result which was first proven by Gischer [Gis84] in his thesis.

Corollary 3.15
Let $\leftrightsquigarrow\ = \mathcal{A}ct^2$, i.e. all process variables are dependent, then

$$P = Q \quad \Longrightarrow \quad P \overset{ax}{=} Q,$$

where in proving $P \overset{ax}{=} Q$ we only need the axioms **(A1)**. □

Proof of the Completeness Theorem.
By induction on the size of the smaller one of P, Q; the *size* of a term P, denoted $|P|$, is the number of occurrences of *skip*, process variables, and parallel and layer operation symbols in P.

Without loss of generality assume that $|P| \leq |Q|$. We apply case analysis on the syntactic form of P:

Case $P \overset{def}{=} \mathit{skip}$ or $P \overset{def}{=} X$. Then Q denotes $\mathcal{S}kip$ or $\mathcal{X}$, respectively. So by the Term Existence Lemma: $Q \overset{ax}{=} \mathit{skip} \overset{def}{=} P$ or $Q \overset{ax}{=} X \overset{def}{=} P$, respectively.

Case $P \overset{def}{=} P_0 \oplus P_1$ where $\oplus$ is one of $\|$ or $\bullet$. Take $\mathcal{Q}_j$ to be the graph denoted by P_j. So by the Term Existence Lemma, reading $Q, \mathcal{Q}, j$ instead of $P, \mathcal{P}, i$, there exist terms Q_j such that:

$$
\begin{array}{lll}
(\mathbf{a}) & Q_j \text{ denotes } \mathcal{Q}_j, \text{ hence } Q_j = P_j \text{ (semantically)}, & \\
(\mathbf{b}) & Q \quad \overset{ax}{=} \quad Q_0 \oplus Q_1 . &
\end{array}
$$

Observe that:

$$
\begin{array}{ccccccccc}
|P_i| & < & |P_0| + 1 + |P_1| & = & |P_0 \oplus P_1| & = & |P|, & & \text{so} \\
& \min(|P_i|, |Q_i|) & \leq & |P_i| & < & |P| & = & \min(|P|, |Q|), &
\end{array}
$$

hence, by induction applied to (a):

$$(\mathbf{c}) \qquad P_i \quad \overset{ax}{=} \quad Q_i .$$

Now, by the case assumption, (c), and (b):

$$P \quad \overset{def}{=} \quad P_0 \oplus P_1 \quad \overset{ax}{=} \quad Q_0 \oplus Q_1 \quad \overset{ax}{=} \quad Q .$$

This completes the proof of the Completeness Theorem. $\square$

Remark. Consider the process term

$$(P \bullet Q) \parallel (R \bullet S)$$

with $P \sharp S$ and $Q \sharp R$. Then none of the axioms in **(A1)** and neither (CCL-L) and (CCL-R) are applicable to this term. Hence the CCL law can not be derived from these laws.

Moreover a case analysis shows that every law of the form

$$P \oplus (Q \otimes R) = P \otimes (Q \oplus R)$$

can be derived from the axioms **(A1)** or is equal to (CCL-L) or (CCL-R).

Hence the CCL rule cannot be derived from simpler laws. **(End of remark)**

References

[CG88] C. Chou and E. Gafni. Understanding and verifying distributed algorithms using stratified decomposition. In *Proc. 7th ACM Symposium on Principles of Distributed Computing*. ACM, 1988.

[EF82] Elrad and N. Francez. Decomposition of distributed programs into communication closed layers. *Science of Computer Programming*, 2, 1982.

[GHS83] R.T. Gallager, P.A. Humblet, and P.M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM TOPLAS*, 5(1):66–77, Jan 1983.

[Gis84] J. L. Gischer. *Partial Orders and the Axiomatic Theory of Shuffle*. PhD thesis, Stanford University, 1984.

[JPSZ91] W. Janssen, M. Poel, K. Sikkel, and J. Zwiers. The primordial soup algorithm: A systematic approach to the specification and design of parallel parsers. In *Proc. Computing Science in the Netherlands*, pages 298–314, 1991.

[JPZ91] W. Janssen, M. Poel, and J. Zwiers. Action systems and action refinement in the development of parallel systems. In *Proc. of CONCUR '91*, pages 298–316. Springer-Verlag, LNCS 527, 1991.

[JPZ93] W. Janssen, M. Poel, and J. Zwiers. Action systems and action refinement in the development of parallel systems. Technical Report 93-14, University of Twente, 1993.

[JZ92a] W. Janssen and J. Zwiers. From sequential layers to distributed processes, deriving a distributed minimum weight spanning tree algorithm, (extended abstract). In *Proc. 11th ACM Symposium on Principles of Distributed Computing*, pages 215–227. ACM, 1992.

[JZ92b] W. Janssen and J. Zwiers. Protocol design by layered decomposition, a compositional approach. In J.Vytopil, editor, *Proc. Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 307–326. Springer-Verlag, LNCS 571, 1992.

[KP87] S. Katz and D. Peled. Interleaving set temporal logic. In *Proc. of the 6th ACM Symposium on Principles of Distributed Computing, Vancouver*, pages 178–190, 1987.

[KP89] S. Katz and D. Peled. An efficient verification method for parallel and distributed programs. In *Proc. of the REX workshop on Linear Time, Branching Time and Partial order in Logics and Models for Concurrency, Noordwijkerhout, Springer LNCS 354*, pages 489–507. Springer-Verlag, 1989.

[KP90] S. Katz and D. Peled. Interleaving set temporal logic. *Theoretical Computer Science*, 75(2), 1990.

[KP92] S. Katz and D. Peled. Verification of distributed programs using representative interleaving sequences. *Distributed Computing*, 6(2), 1992.

[Maz89] A. Mazurkiewicz. Basic notions of trace theory. In J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Proc. of the REX workshop on Linear Time, Branching Time and Partial order in Logics and Models for Concurrency, Noordwijkerhout 1988, Springer LNCS 354*, pages 285–363, 1989.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.

[PZ92] M. Poel and J. Zwiers. Layering techniques for development of parallel systems. In *Proc. CAV*, 1992.

[PZ93a] M. Poel and J. Zwiers. Closed layers in the presence of conspiracy and cascading. 1993.

[PZ93b] M. Poel and J. Zwiers. Modular completeness for communication closed layers. Technical report, University of Twente, 1993.

[SdR89] F.A. Stomp and W.P. de Roever. Designing distributed algorithms by means of formal sequentially phased reasoning. In J.-C. Bermond and M. Raynal, editors, *Proc. of the 3rd International Workshop on Distributed Algorithms, Nice, LNCS 392*, pages 242–253. Springer-Verlag, 1989.

[VTL82] J. Valdes, R.E. Tarjan, and E.L. Lawler. The recognition of series parallel digraphs. *SIAM Journal of Computing*, 2(11):298–313, 1982.

[Zwi89] J. Zwiers. *Compositionality, Concurrency and Partial Correctness.* Springer LNCS 321, 1989.

[Zwi91] J. Zwiers. Layering and action refinement for timed systems. In *Proc. of the REX Workshop on Real Time: Theory and Practice*, Mook, the Netherlands, June 3-7 1991. Springer-Verlag.