An Algebraic Semantics for Hierarchical P/T Nets
(extended abstract)

Twan Basten and Marc Voorhoeve

Department of Computing Science, Eindhoven University of Technology, The Netherlands
email: {tbasten,wsinmarc}@win.tue.nl

Abstract. The first part of this paper gives an algebraic semantics for Place/Transition nets in terms of an algebra which is based on the process algebra ACP. The algebraic semantics is such that a P/T net and its term representation have the same operational behavior. As opposed to other approaches in the literature, the actions in the algebra do not correspond to the firing of a transition, but to the consumption or production of tokens. Equality of P/T nets can be determined in a purely equational way.
The second part of this paper extends the results to hierarchical P/T nets. It gives a compositional algebraic semantics for both their complete operational behavior and their high-level, observable behavior. By means of a non-trivial example, the Alternating-Bit Protocol, it is shown that the notions of abstraction and verification in the process algebra ACP can be used to verify in an equational way whether a hierarchical P/T net satisfies some algebraic specification of its observable behavior. Thus, the theory in this paper can be used to determine whether two hierarchical P/T nets have the same observable behavior. As an example, it is shown that the Alternating-Bit Protocol behaves as a simple one-place buffer. The theory forms a basis for a modular, top-down design methodology based on Petri nets.

Key words: Place/Transition nets – hierarchical Petri nets – process algebra – abstraction – verification – top-down design

1 Introduction

Motivation. The theory of Petri nets (see for example [27]) has been developed to design and analyze distributed systems. In order to support the design of large, complex systems, high-level Petri nets [17, 19] have been defined, which include hierarchy, data, and time. Based on high-level Petri nets, automated tools, such as Design/CPN [22] and ExSpect [1], have been developed. The most important reasons for the widespread use of Petri nets in the area of system design, are their intuitive graphical representation and the simplicity of the main concepts of the theory. Unfortunately, the class of Place/Transition nets, which underlies all other classes of Petri nets, also lacks one important property, which is essential to top-down, modular design: *compositionality*.

Several other theories for describing concurrent systems do have this property. For example, process algebras, such as CCS [23], CSP [18], and ACP [4], all support compositionality. Therefore, it is not surprising that several attempts have been made to integrate P/T nets and CCS-like calculi. Some approaches give a net semantics for CCS-like calculi; others describe an algebraic semantics for (some subclass of) P/T nets. All approaches have in common that there is a one-to-one correspondence between actions in the calculus and transitions in the P/T nets. Usually, only flat P/T nets are considered. Below, a brief survey is given of some recent results described in the literature.

This paper presents a different approach. First, it gives an algebraic semantics for flat P/T nets in terms of an ACP-like algebra [4] in which atomic actions correspond to the consumption or production of a single token. This correspondence is the essential idea that allows for a straightforward extension to hierarchical nets. The algebraic semantics is such that the two transition systems which form the operational semantics of a P/T net and its algebraic representation are equivalent. The process algebra ACP is chosen, because it emphasizes equational reasoning as opposed to model-based reasoning. An equational theory is given which can be used to determine equality of P/T nets, without referring to their operational semantics. Second, this paper gives an algebraic semantics for both the complete behavior and the observable behavior of hierarchical P/T nets. The complete behavior of a hierarchical P/T net is the behavior of the flat net which is obtained when the hierarchical net is unfolded; the observable behavior of a hierarchical net is the behavior after hiding the internal behavior. The algebraic semantics for the observable behavior of hierarchical P/T nets can be used to verify whether a net satisfies some algebraic specification of its behavior, and, as a result, to determine whether two hierarchical nets have the same observable behavior. Everything can be done in a purely equational and compositional way. The theory thus forms the basis for a top-down design methodology based on hierarchical Petri nets. Figure 1 gives an example, which is used to further explain and motivate the research described in this paper.

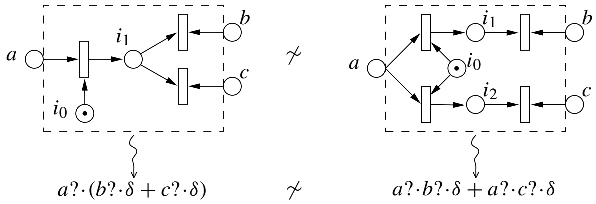

Fig. 1. Motivating example.

The two nets shown in Figure 1 look like ordinary P/T nets. However, the dashed boxes divide the set of places into *pins* and *internal places*. The idea is that pins are connectors to an environment, which can remove tokens from or add tokens to the pins. The internal structure of the net is hidden as in a black box. Thus P/T nets with pins are a very simple form of hierarchical nets. It is straightforward to extend such P/T nets with pins to more general hierarchical P/T nets. Besides places and transitions, the internal structure of a high-level, hierarchical net can also contain subnets, whose pins are connected to internal places or pins from the high-level net. Essentially, this is the hierarchy construct underlying the high-level nets described in [17]. It is also one of the constructs used to build hierarchical nets as described in [19]. Furthermore, it is supported by tools as Design/CPN and ExSpect.

The main objective of this paper is to give an algebraic semantics for the observable behavior of hierarchical nets, that is, their behavior projected onto pins. Therefore, it seems most appropriate to define the behavior of a net in terms of production and consumption of tokens. Consider, for example, the left net in Figure 1. Assuming that the environment provides sufficiently many tokens, it is easy to see that it first consumes a token from a, then, either consumes a token from b or from c, after which it deadlocks. In an ACP-like algebra, one could express this kind of behavior by the term $a?\cdot(b?\cdot\delta+c?\cdot\delta)$, where a question mark denotes the consumption of a token and δ denotes deadlock.

In order to compare hierarchical nets, some suitable notion of equivalence is needed. In the context of this paper, two hierarchical nets are considered equivalent if and only if their observable behavior is the same. For example, consider the right net in Figure 1. From the environment, it looks the same as the other net. However, it does not have the same observable behavior. Obviously, like the other net, it first consumes a token from a. But after that, its behavior is different. If the uppermost transition consumed the token from a after that, it will only consume a token from b but not from c. The consumption of a token from a implies a choice between b and c. For the left net, this is not the case. After consuming a token from a, it is still able to consume a token from either b or c. The behavior of the nets is different, because their *moments of choice* are different.

So, an appropriate notion of equivalence should capture the moments of choice in a process, often called the *branching structure of the process*. In [14], Van Glabbeek formally defines the branching structure of a process. He shows that an equivalence notion captures the branching structure if and only if it distinguishes more processes than *bisimulation equivalence*. Two processes are *bisimilar* if and only if, at any time, they can copy, or simulate, each others actions. Therefore, in this paper, two hierarchical P/T nets are considered equivalent if and only if their observable behavior is bisimilar. Consequently, the two nets of Figure 1 are not equivalent. In their survey on refinement of Petri nets, Brauer, Gold, and Vogler [10] propose bisimulation equivalence for similar reasons as explained above. It also appears in the survey on equivalence notions for Petri nets by Pomello, Rozenberg, and Simone [26]. Note that in this paper bisimulation is not used explicitly to determine equivalence of nets. Instead, an equational theory is given which can be used for this purpose. Since it is not the main subject of this paper to investigate equivalences on Petri nets, it is left for future work to investigate other notions of equivalence which might be of interest in the context of this paper. True concurrency equivalences seem to be interesting candidates.

Note that the notion of equivalence of hierarchical nets introduced above has an interesting consequence. For example, adding an *internal* output place to any of the transitions of a net in Figure 1 does not change its observable behavior. This behavior is even independent of the number of initial tokens in such an additional place. This means that nets with different reachable states can have equivalent observable behavior.

Summarizing, this paper gives an algebraic semantics for hierarchical P/T nets, in which atomic actions correspond to the consumption or production of tokens. An equational theory is given which can be used to determine equivalence of hierarchical nets.

Related work. One way to integrate P/T nets with CCS-like calculi is to give a net semantics for terms in the calculus, thus providing the calculus with a true concurrency semantics. Examples of this approach are Best, Devillers, and Hall [8], Degano, De Nicola, and Montanari [11], Van Glabbeek and Vaandrager [15], Goltz [16], Montanari and Yankelevich [24], Olderog [25], and Taubner [28]. As explained, this paper does the converse. It gives an algebraic semantics for P/T nets. Examples of this approach are Baeten and Bergstra [2], Boudol, Roucairol, and De Simone [9], and Dietz and Schreibert [12]. All three approaches are discussed briefly.

Dietz and Schreibert [12] give an algebraic semantics of P/T nets which reflects the parallelism in their dynamic behavior. The parallel components in the algebraic representation of a net do not correspond to its structural components. Such a relationship does exist in the other two papers. Boudol, Roucairol, and De Simone [9] give an algebraic term for each place and each transition of a P/T net. The complete behavior of the net is the parallel composition of all these terms. The communication between these terms corresponds to the flow of tokens. A similar approach is taken by Baeten and Bergstra [2]. Atomic actions in the algebra correspond to transitions in the P/T nets. So-called input and output causes are added to these actions, corresponding to input and output places. The behavior of a net is the parallel composition of all actions corresponding to its transitions. A so-called causal state operator is used to restrict the behavior in such a way that it corresponds to the flow of tokens in the net. As opposed to Boudol, Roucairol, and De Simone, Baeten and Bergstra emphasize equational reasoning.

The approach pursued in this paper is most closely related to the work of Baeten and Bergstra. The algebraic semantics given is very similar to theirs. However, as mentioned before, an important difference between this paper and all other approaches is that, in this paper, actions in the algebraic semantics correspond to the consumption or production of tokens, whereas, in the other approaches, there is a correspondence between actions and transitions. In the latter case, there is no straightforward extension to hierarchical nets. As this paper shows, there is a straightforward extension when actions correspond to the consumption or production of tokens.

Organization. The paper is organised as follows. Section 2 introduces a framework of transition systems which serve as the operational semantics for both P/T nets and terms in the process algebra. In Section 3, P/T nets with pins and their operational behavior are defined. Section 4 introduces the equational theory PTNA, for *Place/Transition-Net Algebra*, and its operational semantics. The algebraic semantics of a P/T net with pins is a closed PTNA term. It has the same operational semantics as the P/T net. Section 5 introduces the distinction between internal and observable behavior. It extends the algebraic semantics to general hierarchical nets. In Section 6, the theory is applied to the example of the Alternating-Bit Protocol. Finally, Section 7 ends with some concluding remarks and a discussion of future work. Due to lack of space, proofs are omitted. They can be found in [5].

Notation. For any set X, the notation $\mathbb{P}\,X$ denotes the powerset of X and $\mathbb{B}\,X$ denotes the set of all bags over X, where a bag is a finite multi-set. The standard operators minus ($-$) and union ($\cup$) on sets are also used on bags. Minus binds stronger than union. Set inclusion ($\subseteq$) is also extended to bags. Furthermore, restriction of some set or bag X to some domain D is denoted $X\lceil D$. Restriction binds stronger than minus and union. A bag is written as a sequence of its elements in arbitrary order, where each element appears only once and a superscript denotes its cardinality. Furthermore, for an *associative* binary operator $\otimes$, some function f, some $n \in \mathbb{N}$, and operands $x_0, \ldots, x_n$, the quantifier notation $(\otimes i \ : \ 0 \le i \le n \ : \ f(x_i))$ is used as a shorthand notation for $f(x_0) \otimes \ldots \otimes f(x_n)$. For an associative binary operator $\oplus$ that is also *commutative*, the notation $(\oplus\, x \ : \ x \in X \ : \ f(x))$, where X is some bag of operands, is sometimes used.

2 Processes

This section introduces a general framework of labeled transition systems. It serves as the process domain in which the operational semantics of both P/T nets and algebraic terms are defined. In this way, the behavior of P/T nets and algebraic terms can be compared in an unambiguous way. For now, there is no distinction between internal and observable behavior.

Definition 2.1. (Process space) A *process space* over some set of *actions* $\mathcal{A}$ is a pair $(\mathcal{P}, \longrightarrow)$, where $\mathcal{P}$ is a set of processes, and $\_ \xrightarrow{\_} \_ : \mathbb{P}(\mathcal{P} \times \mathcal{A} \times (\mathcal{P} \cup \{\surd\}))$ a ternary transition relation.

Intuitively, for any $p, p' \in \mathcal{P}$ and $\alpha \in \mathcal{A}$, the predicate $p \xrightarrow{\alpha} p'$ means that process p can perform an action α, thus transiting into a process p'. The predicate $p \xrightarrow{\alpha} \surd$ means that process p terminates successfully upon executing an action α. In P/T-net theory, no distinction is made between successful and unsuccessful termination (deadlock). However, in the process algebra ACP, such a distinction does exist. Hence, the distinction is necessary in the process domain. Of course, P/T nets should represent processes which cannot terminate successfully.

An equivalence on processes which captures their branching structure is defined as follows. Let $(\mathcal{P}, \longrightarrow)$ be some process space over $\mathcal{A}$.

Definition 2.2. (Bisimulation) A binary relation $\mathcal{R} : \mathbb{P}(\mathcal{P} \times \mathcal{P})$ is called a *bisimulation* if and only if, for any $p, p', q, q' \in \mathcal{P}$ and $\alpha \in \mathcal{A}$,

$i)$ $p\mathcal{R}q \wedge p \xrightarrow{\alpha} p' \Rightarrow (\exists\, q' : q' \in \mathcal{P} : q \xrightarrow{\alpha} q' \wedge p'\mathcal{R}q')$.

$ii)$ $p\mathcal{R}q \wedge q \xrightarrow{\alpha} q' \Rightarrow (\exists\, p' : p' \in \mathcal{P} : p \xrightarrow{\alpha} p' \wedge p'\mathcal{R}q')$.

$iii)$ $p\mathcal{R}q \Rightarrow p \xrightarrow{\alpha} \surd \Leftrightarrow q \xrightarrow{\alpha} \surd$.

Two processes p and q are called *bisimilar*, denoted $p \sim q$, if and only if there exists a bisimulation $\mathcal{R}$ such that $p\mathcal{R}q$.

3 P/T Nets with Pins

This section formalizes the notion of P/T nets with pins and their operational semantics. No distinction is made between observable and internal behavior. This means that the dashed box in the graphical representation of P/T nets with pins merely is a *glass box* instead of a black box. Let $\mathsf{L_p}$ be some universe of place labels and $\mathsf{L_t}$ a universe of transition labels.

Definition 3.1. (P/T net with pins) A *P/T-net structure with pins* is a 5-tuple $(P, T, \mathbf{i}, \mathbf{o}, I)$, where $P \subseteq \mathsf{L_p}$ is a finite, non empty set of places, $T \subseteq \mathsf{L_t}$ is a finite, non empty set of transitions, $\mathbf{i} : T \to \mathbb{B}\, P$ a function which gives the input places for each transition, $\mathbf{o} : T \to \mathbb{B}\, P$ a function which gives the output places for each transition, and $I \subseteq P$ the set of internal places. The set $P - I$ is the set of *pins*. The functions $\mathbf{i}$ and $\mathbf{o}$ must satisfy the following two conditions: (i) the union of the range of $\mathbf{i}$ and the range of $\mathbf{o}$ is equal to P, which means that there are no isolated places; (ii) for any $t \in T$, $\mathbf{i}t \cup \mathbf{o}t$ is not empty, which means that there are no isolated transitions. A *P/T net with pins*, in the remainder simply called P/T net or net, is a pair (N, s), where N is its structure as defined above and $s : \mathbb{B}\, I$ is its *state* or *marking*.

Note that, when the set of pins is empty, a P/T net with pins is just an ordinary P/T net. The state of a P/T net with pins is a bag of *internal* places. As usual, an element a of the state of a P/T net is often referred to as a *token* residing in place a. The reason for not considering *pins* in the state of a net is that we want to determine the behavior of a P/T net under the assumption that the environment is responsible for producing tokens on and consuming tokens from pins.

The dynamic behavior of a P/T net is a process space in which the P/T nets are the processes and the transition relation determines what actions a P/T net can perform. To formalize this definition, some terminology and definitions are given first.

Let (N, s) be a P/T net, where $N = (P, T, \mathbf{i}, \mathbf{o}, I)$. A transition $t \in T$ is *enabled* if and only if, for each *internal* place $a \in I$ with positive cardinality n in $\mathbf{i}t$, there are at least n tokens in a available in s. More concisely, a transition t is enabled if and only if $\mathbf{i}t \lceil I \subseteq s$. If a transition is enabled, it can *fire*. Upon firing, a transition t removes n tokens from each of its input places a, where n is again the cardinality of a in $\mathbf{i}t$; it adds m tokens to each of its output places b, where m is the cardinality of b in $\mathbf{o}t$. This means that upon firing t, the P/T net (N, s) evolves into another P/T net $(N, s - \mathbf{i}t \cup \mathbf{o}t \lceil I)$. Note that it follows from the standard definition of "$-$" that it is not necessary to restrict $\mathbf{i}t$ to I. The tokens that are removed from the net when firing a transition are often referred to as *consumed* tokens or the *consumption* of a transition; tokens that are added are referred to as the *production* of a transition. If I is chosen equal to P, that is, all places are internal, the definitions above are the usual ones for P/T nets without pins. Or, from a different viewpoint, the definitions given here are the usual ones *provided that the environment supplies sufficiently many tokens on the input pins*. It is assumed that transitions cannot fire simultaneously. However, as explained in Section 7, this is not a real restriction. All results can be extended to P/T nets with an operational semantics that allows transitions to fire simultaneously. The reason for not doing so, is that it unnecessarily complicates the theory and examples that follow, and thus distracts the reader from the essential points of this paper.

The definitions given so far are sufficient to formalize the operational semantics of P/T nets. Let $\mathcal{PTN}$ be the set of all P/T nets. A single action of a net, which is the firing of a single transition, is determined by two bags, the consumption and the production of the transition. Therefore, let $\mathcal{A}$ be $\mathbb{B}\,\mathsf{L}_\mathsf{p} \times \mathbb{B}\,\mathsf{L}_\mathsf{p}$. The transition relation $\_ \xrightarrow{\_} \_ : \mathbb{P}(\mathcal{PTN} \times \mathcal{A} \times (\mathcal{PTN} \cup \{\surd\}))$ is the smallest relation satisfying, for any net structure $N = (P, T, \mathbf{i}, \mathbf{o}, I)$, bags $s, s' \in \mathbb{B}\,I$, and transition $t \in T$,

$$(N, s \cup \mathbf{i}t \lceil I) \xrightarrow{(\mathbf{i}t, \mathbf{o}t)} (N, s \cup \mathbf{o}t \lceil I).$$

Note that, according to this definition, P/T nets have no successful termination. If a P/T net cannot perform any actions anymore, it is deadlocked. This conforms to the usual semantics for nets, where no distinction is made between successful and unsuccessful termination.

Example 3.2. Let $PTN_0 = (N_0, i_0)$ and $PTN_1 = (N_1, i_0)$ be the left and right P/T net in Figure 1 respectively. Figure 2 visualizes the transition relations of both nets. Since internal activity is visible, and hence the two nets perform different actions, they are obviously not bisimilar.

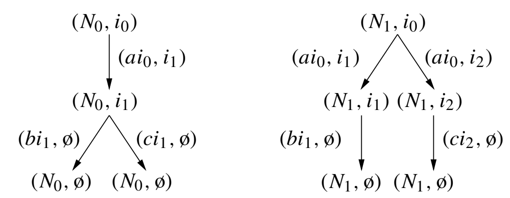

Fig. 2. The transition relations of PTN_0 and PTN_1.

4 An Algebraic Semantics for P/T Nets

This section introduces an ACP-like equational theory and its operational semantics. It gives an algebraic semantics for P/T nets such that a P/T net and its term representation have the same operational behavior.

The theory PTNA. An *equational theory* consists of a *signature* and a set of *axioms*. The signature defines the *sorts* of a theory and its *functions*. A 0-ary function is often called a *constant*.

The equational theory used in this paper is PTNA, *Place/Transition-Net Algebra*. The signature and the axioms are given in Table 1. The theory is parameterized by a set of constants $\mathsf{L_p}$, which is the set of place labels introduced in the previous section. The first part of Table 1 lists the sorts of PTNA; the second part defines the functions and the third part the axioms. An informal explanation is given below.

PTNA($\mathsf{L_p}$)

$\mathsf{A},\ \mathsf{AC},\ \mathsf{P};\ \mathsf{A} \subseteq \mathsf{AC} \subseteq \mathsf{P}$

$\delta : \mathsf{P}$	$\_?, \_! : \mathsf{L_p} \to \mathsf{A}$	
$\_+\_ : \mathsf{P} \times \mathsf{P} \to \mathsf{P}$	$\_\cdot\_ : \mathsf{P} \times \mathsf{P} \to \mathsf{P}$	$\_^*\_ : \mathsf{P} \times \mathsf{P} \to \mathsf{P}$
$\_\parallel\_ : \mathsf{P} \times \mathsf{P} \to \mathsf{P}$	$\_\lfloor\!\lfloor\_ : \mathsf{P} \times \mathsf{P} \to \mathsf{P}$	$\_\mid\_ : \mathsf{AC} \times \mathsf{AC} \to \mathsf{AC}$
$\lambda_-^- : (\mathbb{P}\,\mathsf{L_p} \times \mathbb{B}\,\mathsf{L_p}) \to (\mathsf{P} \to \mathsf{P})$	$\mathbf{c}\_, \mathbf{p}\_ : \mathsf{AC} \to \mathbb{B}\,\mathsf{L_p}$	

$d : \mathsf{AC} \cup \{\delta\};\ e, f, g : \mathsf{AC};\ x, y, z : \mathsf{P};\ I : \mathbb{P}\,\mathsf{L_p};\ s : \mathbb{B}\,\mathsf{L_p}$

$x + y = y + x$	$A1$	$e \mid f = f \mid e$	$S1$
$(x + y) + z = x + (y + z)$	$A2$	$(e \mid f) \mid g = e \mid (f \mid g)$	$S2$
$x + x = x$	$A3$		
$(x + y)\cdot z = x\cdot z + y\cdot z$	$A4$	$x \parallel y = x \lfloor\!\lfloor y + y \lfloor\!\lfloor x$	$M1$
$(x\cdot y)\cdot z = x\cdot(y\cdot z)$	$A5$	$d \lfloor\!\lfloor x = d\cdot x$	$M2$
$x + \delta = x$	$A6$	$d\cdot x \lfloor\!\lfloor y = d\cdot(x \parallel y)$	$M3$
$\delta\cdot x = \delta$	$A7$	$(x + y) \lfloor\!\lfloor z = x \lfloor\!\lfloor z + y \lfloor\!\lfloor z$	$M4$
$\lambda_s^I(\delta) = \delta$	$CSO1$	$(x \lfloor\!\lfloor y) \lfloor\!\lfloor z = x \lfloor\!\lfloor (y \parallel z)$	$ASC1$
$\mathbf{c}e \lceil I \subseteq s \ \Rightarrow\ \lambda_s^I(e) = e$	$CSO2$	$(x \parallel y) \parallel z = x \parallel (y \parallel z)$	$ASC2$
$\mathbf{c}e \lceil I \not\subseteq s \ \Rightarrow\ \lambda_s^I(e) = \delta$	$CSO3$		
$\lambda_s^I(e\cdot x) = \lambda_s^I(e)\cdot\lambda^I_{s-\mathbf{c}e\cup\mathbf{p}e\lceil I}(x)$	$CSO4$	$x^* y = x\cdot(x^* y) + y$	$BKS1$
$\lambda_s^I(x + y) = \lambda_s^I(x) + \lambda_s^I(y)$	$CSO5$	$x^*(y\cdot z) = (x^* y)\cdot z$	$BKS2$
		$x^*(y\cdot((x + y)^* z) + z) = (x + y)^* z$	$BKS3$

Table 1. Place/Transition-Net Algebra.

Intuitively, A is the set of *atomic actions*, AC the set of *actions*, and P the set of *processes*. Each atomic action is either the consumption of a token or the production of a token. A consumption is denoted by "?" and a production by "!." An action is the simultaneous consumption and/or production of one or more tokens. Actions are constructed by the synchronous-merge operator $\mid$. In an equational theory, nothing is an element of a subsort unless explicitly stated. This yields the following property.

Property 4.1. *For any* $a \in \mathsf{A}$, *there exists a* $b \in \mathsf{L_p}$, *such that* $a = b?$ *or* $a = b!$. *For any* $b \in \mathsf{L_p}$, $b? \in \mathsf{A}$ *and* $b! \in \mathsf{A}$. *For any* $e \in \mathsf{AC}$, *there exist* $a_0, \ldots, a_n \in \mathsf{A}$, *for some* $n \in \mathbb{N}$, *such that* $e = (\mid i :\ 0 \le i \le n :\ a_i)$. *For any* $a_0, \ldots, a_n \in \mathsf{A}$, *where* $n \in \mathbb{N}$, $(\mid i :\ 0 \le i \le n :\ a_i) \in \mathsf{AC}$.

The synchronous merge is a very simple form of the communication merge as defined in [7]. There, the axioms $S1$ and $S2$ appear as $C1$ and $C2$ respectively. The reason for changing the names is that there is no communication in PTNA. The operators $+$ and $\cdot$ denote choice and sequential composition respectively. Axiom $A4$ states the *right distributivity* of sequential composition over choice. To be able to distinguish between processes with different moments of choice, the converse, left distributivity, is not an axiom of the theory. As expressed by axioms $A6$ and $A7$, the special process δ can be interpreted as *inaction* or *deadlock*. The merge operator $\|$ can be interpreted as parallel execution. It is axiomatized using an auxiliary operator $\lfloor\!\lfloor$, called the left merge. It has the same meaning as the merge except that the left process must execute the first action. Axioms $ASC1$ and $ASC2$ are the so-called *axioms of standard concurrency*. Often, they are derivable for *closed* terms and omitted from the theory. However, in combination with the binary Kleene star (*) they are not derivable for closed terms and hence included. The binary Kleene star adds a simple form of recursion to the theory. It is the original star operator as introduced by Kleene [20]. In [6], where the axioms $BKS1$–3 are given, it was introduced into process algebra. Because of its simplicity, the binary Kleene star is preferred over general recursion (see for example [4]). The remainder of this paper shows that it is powerful enough to capture the behavior of P/T nets. Finally, the *causal state operator* λ is a special version of the state operator as described in [4]. It is very similar to the causal state operator as defined in [2]. A state operator has a parameter, the superscript, and a certain state space, the subscript. The state space of the causal state operator can be interpreted as the state or marking of some P/T net and its parameter as the set of internal places. For I, s and x as in Table 1, the term $\lambda_s^I(x)$ can be thought of as the P/T net x with internal places I and state s. The auxiliary functions $\mathbf{c}, \mathbf{p} : \ \mathsf{AC} \to \mathbb{B}\,\mathsf{L_p}$, for consumption and production respectively, are defined as follows. For all $a \in \mathsf{L_p}$ and $e \in \mathsf{AC}$, $\mathbf{c}a? = a$, $\mathbf{c}a! = \emptyset$, $\mathbf{c}(a?\,|\,e) = a \cup \mathbf{c}e$, and $\mathbf{c}(a!\,|\,e) = \mathbf{c}e$; $\mathbf{p}a? = \emptyset$, $\mathbf{p}a! = a$, $\mathbf{p}(a?\,|\,e) = \mathbf{p}e$, and $\mathbf{p}(a!\,|\,e) = a \cup \mathbf{p}e$.

The binding precedence of the above operators is as follows. Unary operators bind stronger than binary operators. Sequential composition and Kleene star bind stronger than all other binary operators. Choice binds weaker than all other operators.

The main purpose of a theory as PTNA is that it can be used to reason about processes in a purely equational way. For any processes x and y, PTNA $\vdash x = y$ denotes that $x = y$ can be derived from the axioms. In order to formalize the intuitive notions given above and to be able to compare processes defined by PTNA terms to processes defined by P/T nets, the next paragraph gives an operational semantics for PTNA.

Operational semantics for PTNA. A *semantics* or *model* of a theory is an interpretation in a, usually well known and well understood, mathematical domain, such that the axioms are valid in the interpretation. An *operational* semantics is a model obtained by giving a process space as defined in Section 2.

To obtain a model of the theory PTNA, interpretations of the sorts A, AC, and P must be given. Define the interpretation of the set of processes P as the set of *closed* PTNA terms, denoted $\mathcal{C}$(PTNA). Since it follows from Property 4.1 that A is a subset of all closed terms, namely the set of all closed terms in which either "?" or "!" appear, the interpretation of A is simply A itself. For the same reason, the interpretation of AC is AC itself. Obviously, these definitions satisfy $\mathsf{A} \subseteq \mathsf{AC} \subseteq \mathsf{P}$.

It remains to define the transition relation for processes in $\mathcal{C}(\text{PTNA})$. First, the set $\mathcal{A}$ must be defined. Intuitively, processes in $\mathcal{C}(\text{PTNA})$ can execute an action in AC, thus transiting into another process in $\mathcal{C}(\text{PTNA})$. Therefore, elements in $\mathcal{A}$ should be interpretations of actions which serve as the labels of the transition relation. Let $\phi : \mathsf{AC} \to \mathcal{A}$ be a function that maps actions in AC to actions in $\mathcal{A}$. The semantics given in the previous section for P/T nets suggests that the semantics of an algebra of P/T nets should have pairs of bags as transition labels, where the first element is the consumption and the second element the production of an action. The auxiliary functions $\mathbf{c}$ and $\mathbf{p}$ exactly define the consumption and production of each action in AC. Therefore, let $\mathcal{A}$ be equal to $\mathbb{B}\,\mathsf{L_p} \times \mathbb{B}\,\mathsf{L_p}$ and let for any $e \in \mathsf{AC}$, $\phi(e) = (\mathbf{c}e, \mathbf{p}e)$. The transition relation $\_ \xrightarrow{\_} \_ : \mathbb{P}(\mathcal{C}(\text{PTNA}) \times \mathcal{A} \times (\mathcal{C}(\text{PTNA}) \cup \{\surd\}))$ can now be defined as the smallest relation satisfying the derivation rules in Table 2.

$\alpha : \mathcal{A};\ e : \mathsf{AC};\ s : \mathbb{B}\,\mathsf{L_p};\ I : \mathbb{P}\,\mathsf{L_p};\ p, p', q, q' : \mathcal{C}(\text{PTNA})$

$e \xrightarrow{\phi(e)} \surd$	$\dfrac{p \xrightarrow{\alpha} p'}{p \cdot q \xrightarrow{\alpha} p' \cdot q}$	$\dfrac{p \xrightarrow{\alpha} \surd}{p \cdot q \xrightarrow{\alpha} q}$	
$\dfrac{p \xrightarrow{\alpha} p'}{p + q \xrightarrow{\alpha} p'}$	$\dfrac{q \xrightarrow{\alpha} q'}{p + q \xrightarrow{\alpha} q'}$	$\dfrac{p \xrightarrow{\alpha} \surd}{p + q \xrightarrow{\alpha} \surd}$	$\dfrac{q \xrightarrow{\alpha} \surd}{p + q \xrightarrow{\alpha} \surd}$
$\dfrac{p \xrightarrow{\alpha} p'}{p \parallel q \xrightarrow{\alpha} p' \parallel q}$	$\dfrac{q \xrightarrow{\alpha} q'}{p \parallel q \xrightarrow{\alpha} p \parallel q'}$	$\dfrac{p \xrightarrow{\alpha} \surd}{p \parallel q \xrightarrow{\alpha} q}$	$\dfrac{q \xrightarrow{\alpha} \surd}{p \parallel q \xrightarrow{\alpha} p}$
$\dfrac{p \xrightarrow{\alpha} p'}{p \mathbin{⫡} q \xrightarrow{\alpha} p' \parallel q}$		$\dfrac{p \xrightarrow{\alpha} \surd}{p \mathbin{⫡} q \xrightarrow{\alpha} q}$	
$\dfrac{p \xrightarrow{\alpha} p'}{p^* q \xrightarrow{\alpha} p' \cdot (p^* q)}$	$\dfrac{q \xrightarrow{\alpha} q'}{p^* q \xrightarrow{\alpha} q'}$	$\dfrac{p \xrightarrow{\alpha} \surd}{p^* q \xrightarrow{\alpha} p^* q}$	$\dfrac{q \xrightarrow{\alpha} \surd}{p^* q \xrightarrow{\alpha} \surd}$
$\dfrac{p \xrightarrow{\phi(e)} p'}{\lambda^I_{s \cup \mathbf{c}e \lceil I}(p) \xrightarrow{\phi(e)} \lambda^I_{s \cup \mathbf{p}e \lceil I}(p')}$		$\dfrac{p \xrightarrow{\phi(e)} \surd}{\lambda^I_{s \cup \mathbf{c}e \lceil I}(p) \xrightarrow{\phi(e)} \surd}$	

Table 2. The transition relation for PTNA.

It remains to show that the process space as defined above is indeed a model of the theory PTNA. Recall that a process space is a model if and only if all equations that can be derived from the axioms are valid in the process space. The notion of validity is formalized as follows. In Section 2, bisimulation is defined, which is an equivalence relation on processes. Therefore, it is possible to look at equivalence classes of closed PTNA terms modulo bisimulation, denoted $\mathcal{C}(\text{PTNA})/\!\sim$. For closed terms p and q, the equation $p = q$ is *valid*, denoted $\mathcal{C}(\text{PTNA})/\!\sim\ \models p = q$, if and only if $p \sim q$. That is, if and only if p and q are bisimilar and thus elements of the same equivalence class. It follows from the format of the derivation rules in Table 2 that bisimulation equivalence is a *congruence* on $\mathcal{C}(\text{PTNA})$ [3]. Therefore, it suffices to verify the validity of each axiom of PTNA to prove the following theorem. It states that if equality of two processes can be derived from the axioms, then the processes are bisimilar. This means that one can indeed use equational reasoning instead of model-based reasoning.

Theorem 4.2. *The set of closed* PTNA *terms modulo bisimulation is a model for* PTNA. *That is, for any* $p, q \in \mathcal{C}(\text{PTNA})$*,* $\text{PTNA} \vdash p = q \Rightarrow \mathcal{C}(\text{PTNA})/\!\sim\ \models p = q$.

An algebraic semantics for P/T nets. The following definition associates a closed PTNA term to each P/T net. The idea is to define first the unrestricted behavior of a net. That is, its behavior when every transition is always enabled. Then, the causal state operator instantiated with the initial marking is used to restrict the behavior to all possible firing sequences. The unrestricted behavior of a single transition is the infinite iteration of its consumption and production of tokens. The unrestricted behavior of a net is the parallel execution of all its transitions.

Definition 4.3. (Algebraic semantics for P/T nets) Let $PTN = (N, s)$ be a P/T net, where $N = (P, T, \mathbf{i}, \mathbf{o}, I)$. The algebraic semantics of *PTN*, denoted $\mathcal{N}[\![PTN]\!]$, is defined as follows:

$\mathcal{N}[\![PTN]\!] = \lambda_s^I(\parallel t : t \in T : \mathcal{T}[\![t]\!])$,

where, for any $t \in T$,

$\mathcal{T}[\![t]\!] = ((\mid i : i \in \mathbf{i}t : i?) \mid (\mid o : o \in \mathbf{o}t : o!))^* \delta$.

Empty quantifications should be simply omitted.

The following theorem states that a net and its algebraic representation have the same operational behavior. Recall that $(\mathcal{PTN}, \longrightarrow)$ and $(\mathcal{C}(\text{PTNA}), \longrightarrow)$ are the operational semantics for P/T nets and PTNA respectively. The set $\mathcal{A}$ is equal to $\mathbb{B}\,\mathsf{L_p} \times \mathbb{B}\,\mathsf{L_p}$.

Theorem 4.4. *For any P/T nets (N, s) and (N, s') and some $\alpha \in \mathcal{A}$,*

$(N, s) \xrightarrow{\alpha} (N, s') \Leftrightarrow \mathcal{N}[\![(N, s)]\!] \xrightarrow{\alpha} \mathcal{N}[\![(N, s')]\!]$.

Theorem 4.4 states that, in the union of the two process spaces $\mathcal{PTN}$ and $\mathcal{C}(\text{PTNA})$, the P/T net (N, s) is bisimilar to its algebraic representation $\mathcal{N}[\![(N, s)]\!]$. Note that neither the P/T net nor its term representation can terminate successfully.

Expansion. The following results are useful for simplifying many calculations.

Theorem 4.5. (Expansion) *For any $x_0 \ldots x_n \in \mathsf{P}$, $n \in \mathbb{N} - \{0\}$*

$(\parallel i : 0 \le i \le n : x_i) = (+ i : 0 \le i \le n : x_i \mathbin{⫡} (\parallel j : 0 \le j \le n \wedge j \ne i : x_j))$.

Property 4.6. *For any non empty $E \subseteq \mathsf{AC}$,*

$(\parallel e : e \in E : e^* \delta) = (+ e : e \in E : e \cdot (\parallel e : e \in E : e^* \delta))$.

Example 4.7. Let $PTN_0 = (N_0, i_0)$ be the left P/T net in Figure 1. Let X_0 be the algebraic representation of PTN_0, that is, $X_0 = \mathcal{N}[\![PTN_0]\!]$. This example shows how a simple expression can be derived for X_0, which is an algebraic term for the behavior of PTN_0. In the derivation below, the causal state operator is written without the superscript $\{i_0, i_1\}$. Let $X = (a? \mid i_0? \mid i_1!)^* \delta \parallel (b? \mid i_1?)^* \delta \parallel (c? \mid i_1?)^* \delta$. Use Property 4.6 plus the axioms *CSO*2–5 and *A*6–7 to obtain the following result.

$$\begin{aligned}
X_0 &= \lambda_{i_0}((a? \mid i_0? \mid i_1!)^* \delta \parallel (b? \mid i_1?)^* \delta \parallel (c? \mid i_1?)^* \delta) \\
&= \lambda_{i_0}((a? \mid i_0? \mid i_1!) \cdot X + (b? \mid i_1?) \cdot X + (c? \mid i_1?) \cdot X) \\
&= \lambda_{i_0}(a? \mid i_0? \mid i_1!) \cdot \lambda_{i_1}(X) + \lambda_{i_0}(b? \mid i_1?) \cdot \lambda_{i_0}(X) + \lambda_{i_0}(c? \mid i_1?) \cdot \lambda_{i_0}(X) \\
&= (a? \mid i_0? \mid i_1!) \cdot \lambda_{i_1}(X) + \delta \cdot \lambda_{i_0}(X) + \delta \cdot \lambda_{i_0}(X) \\
&= (a? \mid i_0? \mid i_1!) \cdot X_1,
\end{aligned}$$

where $X_1 = \lambda_{i_1}(X)$. In a similar way, the following result is obtained for X_1.

$X_1 = (b? \mid i_1?) \cdot \delta + (c? \mid i_1?) \cdot \delta$.

It is straightforward to verify that the transition relation of X_0 is the same as the transition relation of PTN_0 that is directly obtained from the operational semantics for nets given in Section 3 (See Figure 2). It is left to the reader to verify that also for the right P/T net in Figure 1, the transition relations for the net and its representation are the same.

5 An Algebraic Semantics for Hierarchical P/T Nets

In this section, hierarchical P/T nets are defined and an algebraic semantics for their complete operational behavior is given. Then, the notion of internal behavior is introduced in the process domain as well as in the theory PTNA. The abstraction mechanism from the process algebra ACP is adapted to give an algebraic semantics for the observable behavior of hierarchical P/T nets. This algebraic semantics indirectly specifies an operational semantics for the observable behavior of hierarchical nets.

5.1 Hierarchical P/T nets

In addition to places and transitions, a hierarchical P/T net has subnets which, in turn, are hierarchical P/T nets.

Definition 5.1. (Hierarchical P/T nets) A *hierachical P/T net* is a 7-tuple $(P, T, S, \mathbf{i}, \mathbf{o}, I, s)$, where $P \subseteq \mathsf{L}_\mathsf{p}$ is a finite, non empty set of places, $T \subseteq \mathsf{L}_\mathsf{t}$ is a finite set of transitions and S a finite set of hierarchical P/T nets, such that $T \cup S$ is not empty; function $\mathbf{i} : (T \cup S) \to \mathbb{B}\, P$ gives the input places for each transition and each subnet, $\mathbf{o} : (T \cup S) \to \mathbb{B}\, P$ gives the output places, $I \subseteq P$ is the set of internal places, and $s : \mathbb{B}\, I$ is the marking of the hierarchical net. It is assumed that there are no isolated places, transitions, or subnets. Set S must be such that for each subnet the set of input and output places is equal to the set of pins of the subnet. The sets of internal places of a hierarchical P/T net and all its subnets must be mutually disjoint. A hierarchical P/T net can be unfolded in the usual way. The unfolding of a hierarchical net must be finite.

Figure 4 in Section 6 shows an example of a hierarchical net. The two nets in Figure 1 are also examples of simple hierarchical nets. A hierarchical net without any subnets is a P/T net as defined in Definition 3.1. The *complete* operational behavior of a hierarchical P/T net is the operational semantics of its unfolding. This semantics can be obtained algebraically as follows.

Definition 5.2. (Complete behavior of hierarchical P/T nets) Let $HN = (P, T, S, \mathbf{i}, \mathbf{o}, I, s)$. Extend the function $\mathcal{N}[\![.]\!]$, as defined in Definition 4.3, inductively as follows. As before, empty quantifications should be omitted.

$$\mathcal{N}[\![HN]\!] = \lambda_s^I((\| \; t : \; t \in T : \; \mathcal{T}[\![t]\!]) \parallel (\| \; sn : \; sn \in S : \; \mathcal{N}[\![sn]\!])).$$

Note that the unfolding of a net must be finite, because only then the net has a finite algebraic representation.

5.2 Processes with Silent Actions.

In Section 2, a process is defined as a labeled transitions system over some set of actions. A process can execute actions, thus transiting into some other process. An action that is executed by a process is part of its observable behavior. To be able to distinguish between observable and internal behavior, *silent actions* are introduced. Usually, silent actions are denoted τ. Only a single symbol is needed, since all internal actions are equal in the sense that they do not have any visible, external effects. The notion of silent actions in an algebraic setting was first introduced by Milner [23].

The definition of a process space given in Section 2 can still be used in a context with silent actions. However, since bisimulation does not distinguish between observable and silent actions, the notion of equality on processes needs to be changed. Processes with the same observable behavior, but with different internal behavior should be equal. As before, the equivalence relation on processes should distinguish processes with different moments of choice. In [14], Van Glabbeek shows that *(rooted) branching bisimulation* is exactly the equivalence that satisfies these two requirements. A formal definition of branching bisimulation can be found in [4, 5]. It is omitted here, since it is fairly complicated and not really essential to the results given in this extended abstract. Branching bisimulation is a slightly finer equivalence than the better known observation equivalence [23]. That is, it distinguishes more processes than observation equivalence.

5.3 Place/Transition-Net Algebra with Silent Actions.

The theory PTNA$^\tau$**.** In this paragraph, the silent action τ and an abstraction operator are added to the theory PTNA, yielding the theory PTNA$^\tau$, for PTNA *with silent actions*. Table 3 gives a definition of PTNA$^\tau$. Recall that $\mathsf{L_p}$ is the set of place labels. The first entry of Table 3 means that PTNA$^\tau$ is a modular extension of PTNA. That is, PTNA$^\tau$ has all sorts, functions, and axioms given in Tables 1 and 3. The auxiliary functions $\mathbf{c}, \mathbf{p}\ :\ \mathsf{AC} \rightarrow \mathbb{B}\,\mathsf{L_p}$ that appear in Table 1 are extended to τ as follows: $\mathbf{c}\tau = \mathbf{p}\tau = \emptyset$.

PTNA$^\tau$($\mathsf{L_p}$)			
PTNA($\mathsf{L_p}$)			
$\tau\ :\ \mathsf{AC}$		$\tau_{\_}\ :\ \mathbb{P}\,\mathsf{L_p} \rightarrow (\mathcal{P} \rightarrow \mathcal{P})$	
$a:\ \mathsf{L_p};\ e, f:\ \mathsf{AC};\ x, y, z:\ \mathsf{P};\ I:\ \mathbb{P}\,\mathsf{L_p}$			
$e \mid \tau = e$	*AT*	$x \cdot \tau = x$	*B*1
		$x \cdot (\tau \cdot (y + z) + y) = x \cdot (y + z)$	*B*2
$a \in I\ \Rightarrow\ \tau_I(a?) = \tau$	*TAC*1	$a \in I\ \Rightarrow\ \tau_I(a!) = \tau$	*TAP*1
$a \notin I\ \Rightarrow\ \tau_I(a?) = a?$	*TAC*2	$a \notin I\ \Rightarrow\ \tau_I(a!) = a!$	*TAP*2
$\tau_I(\delta) = \delta$	*TAD*	$\tau_I(e \mid f) = \tau_I(e) \mid \tau_I(f)$	*TA*1
$\tau_I(\tau) = \tau$	*TAT*	$\tau_I(x + y) = \tau_I(x) + \tau_I(y)$	*TA*2
		$\tau_I(x \cdot y) = \tau_I(x) \cdot \tau_I(y)$	*TA*3
		$\tau_I(x^* y) = \tau_I(x)^* \tau_I(y)$	*TA*4

Table 3. Place/Transition-Net Algebra with silent actions.

For any set of place labels I, the abstraction operator τ_I simply renames actions from I to τ. The axioms $B1$ and $B2$ are an axiomatization of branching bisimulation [13]. Axiom *AT* states that only the visible part of the simultaneous execution of some action and τ is observed. It is different from the normal axioms for τ in ACP with silent actions. There, for any action e, $e \mid \tau$ is equal to δ. The reasoning behind this is that $\mid$ means communication. Since an invisible action cannot communicate, every attempt to communicate with τ results in deadlock.

Operational semantics for PTNA$^\tau$**.** Define the set of processes as the set of closed PTNA$^\tau$ terms, $\mathcal{C}(\text{PTNA}^\tau)$. As before, let $\mathcal{A}$ be equal to $\mathbb{B}\,\mathsf{L_p} \times \mathbb{B}\,\mathsf{L_p}$; let $\phi :\ \mathsf{AC} \longrightarrow \mathcal{A}$, for any $e \in \mathsf{AC}$, be defined as $\phi(e) = (\mathbf{c}e, \mathbf{p}e)$. Note that $\phi(\tau) = (\emptyset, \emptyset)$. This means that, in the process domain, the action $(\emptyset, \emptyset)$ is the silent action. As mentioned in the previous subsection, the silent action in the process domain is usually called τ as well. The reason for this is that actions in the theory often coincide with actions in the process domain. In the remainder, τ always refers to the silent action in the theory. The transition relation $\_ \xrightarrow{\_} \_ :\ \mathbb{P}(\mathcal{C}(\text{PTNA}^\tau) \times \mathcal{A} \times (\mathcal{C}(\text{PTNA}^\tau) \cup \{\surd\}))$ is the smallest relation satisfying the rules in Tables 2 and 4. In Table 2, let p, p', q, q' range over $\mathcal{C}(\text{PTNA}^\tau)$.

$$e:\ \mathsf{AC};\ I:\ \mathbb{P}\,\mathsf{L_p};\ p, p':\ \mathcal{C}(\text{PTNA}^\tau)$$

$$\frac{p \xrightarrow{\phi(e)} p'}{\tau_I(p) \xrightarrow{\phi(\tau_I(e))} \tau_I(p')} \qquad \frac{p \xrightarrow{\phi(e)} \surd}{\tau_I(p) \xrightarrow{\phi(\tau_I(e))} \surd}$$

Table 4. The transition relation for the abstraction operator.

The following theorem states that if equality of two processes can be derived from the axioms of PTNA$^\tau$, then the two processes are rooted branching bisimilar. Hence, equational reasoning can be used instead of model-based reasoning.

Theorem 5.3. *The set of closed* PTNA$^\tau$ *terms modulo rooted branching bisimulation is a model for* PTNA$^\tau$.

An algebraic semantics for hierarchical P/T nets. The algebraic semantics for the observable behavior of a hierarchical P/T net strongly resembles the algebraic semantics which describes its complete behavior. The essential difference is that the abstraction operator is used to hide the internal behavior of the net itself and its components.

Definition 5.4. (Observable behavior of hierarchical P/T nets) Let $HN = (P, T, S, \mathbf{i}, \mathbf{o}, I, s)$ be a hierarchical P/T net. The algebraic semantics for its observable behavior, $\mathcal{H}[\![HN]\!]$, is defined as follows. As before, omit empty quantifications.

$\mathcal{H}[\![HN]\!] = \tau_I \circ \lambda_s^I((\|\ t:\ t \in T:\ \mathcal{T}[\![t]\!]) \parallel (\|\ sn:\ sn \in S:\ \mathcal{H}[\![sn]\!]))$,
where, $\circ$ denotes function composition and, $\mathcal{T}[\![.]\!]$ is as in Definition 4.3.

Example 5.5. Again, let $PTN_0 = (N_0, i_0)$ be the left P/T net in Figure 1. In Example 4.7, the following expression is derived for its complete behavior $X_0 = \mathcal{N}[\![PTN_0]\!]$:

$X_0 = (a? \mid i_0? \mid i_1!) \cdot ((b? \mid i_1?) \cdot \delta + (c? \mid i_1?) \cdot \delta)$.

Since PTN_0 is a flat P/T net, its observable behavior $\mathcal{H}[\![PTN_0]\!]$ is equal to $\tau_{I_0}(X_0)$, where $I_0 = \{i_0, i_1\}$. Use the axioms $TA1$–3, AT and TAD to derive the following result.

$\tau_{I_0}(X_0) = (a? \mid \tau \mid \tau) \cdot ((b? \mid \tau) \cdot \tau_{I_0}(\delta) + (c? \mid \tau) \cdot \tau_{I_0}(\delta)) = a? \cdot (b? \cdot \delta + c? \cdot \delta)$.

The transition relation of $\tau_{I_0}(X_0)$ is shown in Figure 3. The expression $a?\cdot(b?\cdot\delta + c?\cdot\delta)$ describes the behavior of PTN_0 projected onto its pins. It corresponds to the expression already given in the motivating example (Figure 1). In a similar way, it is possible to derive the expression $a?\cdot b?\cdot\delta + a?\cdot c?\cdot\delta$ for the observable behavior of the right P/T net in Figure 1, PTN_1. The transition relation of this term is also shown in Figure 3. Obviously, the two processes are not bisimilar, which is the desired result.

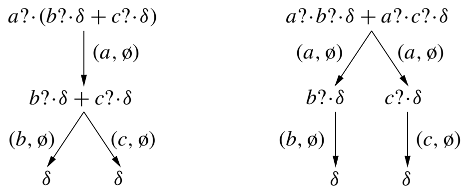

Fig. 3. The transition relations of $\mathcal{H}[\![PTN_0]\!]$ and $\mathcal{H}[\![PTN_1]\!]$.

6 Example: The Alternating-Bit Protocol

In this section, the theory developed in this paper is applied to a non-trivial example, namely the Alternating-Bit Protocol (ABP). The version of the ABP considered here consists of four components: a sender, a receiver, a message channel, and an acknowledgement channel. Both messages and acknowledgments can be corrupted. In order to guarantee that every message which is sent from one side is received by the other side, the sender marks messages alternatingly with a zero and a one bit. Each time it sends a message, it waits for an acknowledgment from the receiver.

The example of the ABP is used two show two applications of the theory developed in this paper. First, it can be used to verify the behavior of a hierarchical P/T net against an algebraic specification. At each hierarchical level, the algebraic terms describing the observable behavior of the subnets can, on the one hand, be seen as the specification of the level below, and, on the other hand, as the implementation of the level above. The theory of this paper can be used to verify such implementations against their specifications in a purely equational and compositional way. Second, the theory can be used to show that different hierarchical nets have the same observable behavior. In a hierarchical P/T net, one can exchange subnets with the same observable behavior, without influencing

$j \in \{0, 1\}$	
Specification:	
i, o	: input/output pin for messages from/to the environment;
abp	: the system that implements the ABP.
System abp:	
sen, rec	: the sender and receiver;
mc, ac	: the message channel and acknowledgement channel;
$jm1/2$	: places for messages with bit j;
$\perp m$	: corrupted messages;
$ja1/2$, $\perp a$	: idem for acknowledgements.
System sen:	
js	: the sender is ready to send a j message.
jw	: the sender has sent a j message and is waiting for an ack.;
jt, $jrt1/2$	: (re)transmit a j message;
ja	: receive an acknowledgement of a j message;
System rec:	
jr	: the receiver is ready to receive a j message;
ja, $jna1/2$	: acknowledge a j message; send a negative j ack.;
Systems mc, ac:	
fjm/a, cjm/a	: forward resp. corrupt message/acknowledgement.

Table 5. Informal explanation of the Alternating-Bit Protocol

the observable behavior at higher levels of abstraction.

Figure 4 gives a three-level hierarchical P/T net of the ABP which conforms to the informal description given above. Since for each net it is clear what are pins and what are internal places, dashed boxes are omitted. Table 5 explains the names of the subnets, transitions, and places.

To demonstrate the first application of the theory, a bottom-up verification of the ABP is given, which consists of four steps. First, simple algebraic expressions are derived for the behavior of the four nets at the most detailed level. Second, the abstraction operator is used to hide their internal behavior. Third, the results of these two steps

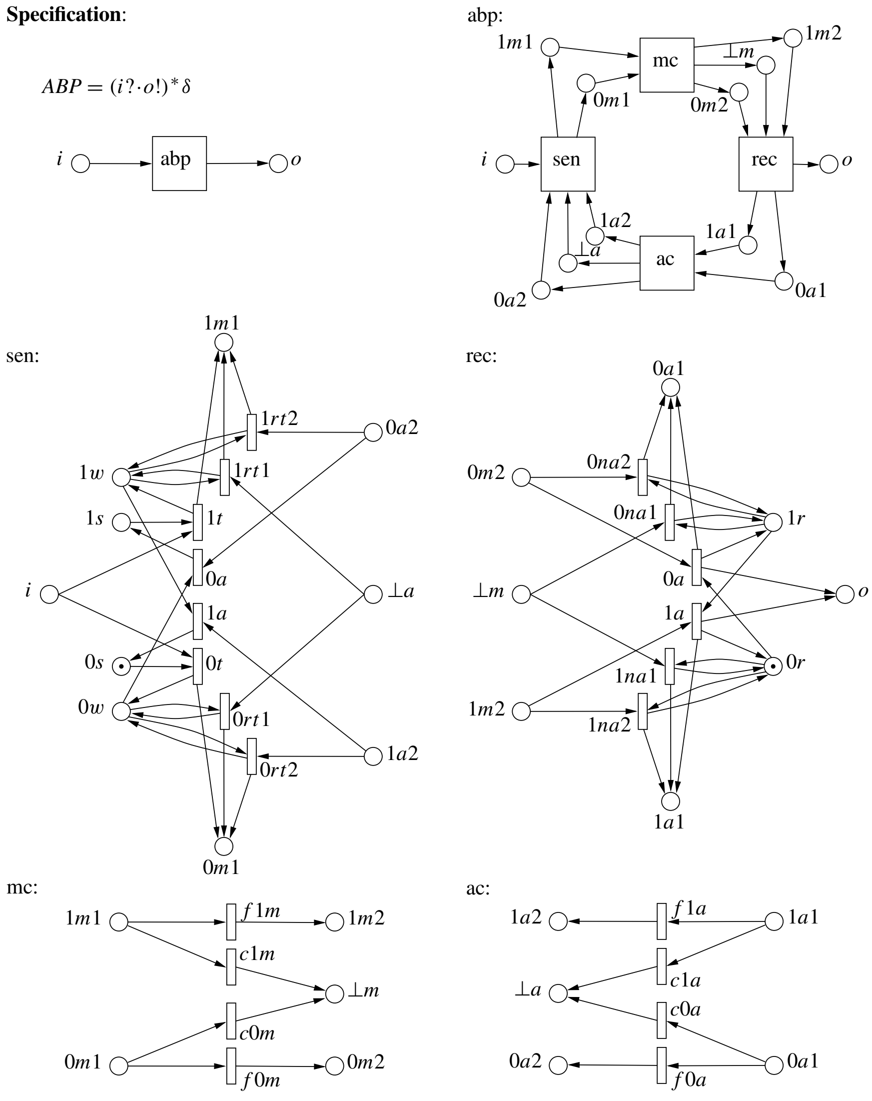

Fig. 4. A hierarchical P/T net of the alternating-bit protocol.

are used to derive an expression for the behavior of the net at the intermediate level. Finally, by hiding the internal behavior of the intermediate net, it is shown that the subnet "abp" satisfies its specification given in Figure 4. To demonstrate the other application of the theory, it is shown that, on the highest level of abstraction, the ABP behaves as a one-place buffer. A P/T net of the one-place buffer is given in Figure 5. The observable behavior of the ABP is the same as the observable behavior of this net.

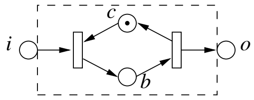

Fig. 5. A one-place buffer.

To be able to do all calculations, one new axiom is needed and one derivation rule. The *Fair Iteration Rule* (*FIR*) states that the binary Kleene star is *fair*. That is, a sequence of silent steps cannot be infinitely long. In terms of P/T nets, it means that, in an internal conflict situation, it is not possible that one transition is always chosen. In terms of the ABP, it means that the channels once in a while forward messages and acknowledgements uncorrupted.

$$x : \mathsf{P}$$
$$\tau^* x = x + \tau \cdot x \qquad \mathit{FIR}$$

The *Recursive Specification Principle* for the binary Kleene star (*RSP**) is a derivation rule which gives for some restricted set of recursive equations a solution in terms of the binary Kleene star. This derivation rule is necessary since many processes, such as the ABP, are inherently recursive. The requirement "x guarded in $y \cdot x$" means that y cannot terminate successfully without executing at least one visible action. A formal definition of guardedness is omitted here and can be found in [5].

$$x, y, z : \mathsf{P}$$
$$\frac{x = y \cdot x + z, \qquad x \text{ guarded in } y \cdot x}{x = y^* z} \qquad \mathit{RSP}^*$$

Although *FIR* and *RSP**, for arbitrary open terms, cannot be derived from the axioms of PTNA$^\tau$, they are valid in the model of closed PTNA$^\tau$ terms modulo rooted branching bisimulation. The reason for not including *FIR* and *RSP** in the theory PTNA$^\tau$ is that for other applications it might be desirable to extend PTNA$^\tau$ with different fairness principles and derivation rules.

The verification of the ABP. In the following, a transition name t is used as an abbreviation of $(|\ i : i \in \mathbf{i}t : i?) \mid (|\ o : o \in \mathbf{o}t : o!)$. For example, $0t$ is an abbreviation of $(i? \mid 0s? \mid 0w! \mid 0m1!)$. Furthermore, the following abbreviations are introduced:

$$
\begin{aligned}
S &= 0t^*\delta \parallel 0rt1^*\delta \parallel 0rt2^*\delta \parallel 0a^*\delta \parallel 1t^*\delta \parallel 1rt1^*\delta \parallel 1rt2^*\delta \parallel 1a^*\delta,\\
R &= 0a^*\delta \parallel 0na1^*\delta \parallel 0na2^*\delta \parallel 1a^*\delta \parallel 1na1^*\delta \parallel 1na2^*\delta,\\
M &= f0m^*\delta \parallel c0m^*\delta \parallel f1m^*\delta \parallel c1m^*\delta,\\
A &= f0a^*\delta \parallel c0a^*\delta \parallel f1a^*\delta \parallel c1a^*\delta.
\end{aligned}
$$

The first part of the verification is to derive expressions for $\mathcal{H}[\![\text{sen}]\!]$, $\mathcal{H}[\![\text{rec}]\!]$, $\mathcal{H}[\![\text{mc}]\!]$, and $\mathcal{H}[\![\text{ac}]\!]$. It consists of two steps. First, expressions are derived for the complete behavior of each component; second, their internal behavior is hidden. To start with, expressions are calculated for the two channels. Applying Property 4.6 and the axioms for the causal state operator yields the following result.

$$M_0 \;=\; \lambda^{\emptyset}_{\emptyset}(M) \;=\; f0m \cdot M_0 + c0m \cdot M_0 + f1m \cdot M_0 + c1m \cdot M_0.$$

RSP^* yields:

$$M_0 \;=\; (f0m + c0m + f1m + c1m)^* \delta.$$

Similarly,

$$A_0 \;=\; \lambda^{\emptyset}_{\emptyset}(A) \;=\; (f0a + c0a + f1a + c1a)^* \delta.$$

Next, an expression is calculated for the sender. The state operator is written without the superscript $\{0s, 0w, 1s, 1w\}$.

$$\begin{aligned}
S_0 &= \lambda_{0s}(S) &&= 0t \cdot S_1,\\
S_1 &= \lambda_{0w}(S) &&= 0rt1 \cdot S_1 + 0rt2 \cdot S_1 + 0a \cdot S_2,\\
S_2 &= \lambda_{1s}(S) &&= 1t \cdot S_3, \text{ and}\\
S_3 &= \lambda_{1w}(S) &&= 1rt1 \cdot S_3 + 1rt2 \cdot S_3 + 1a \cdot S_0.
\end{aligned}$$

Applying RSP^* and $BKS2$ on the last equation yields:

$$S_3 \;=\; (1rt1 + 1rt2)^* (1a \cdot S_0) \;=\; ((1rt1 + 1rt2)^* 1a) \cdot S_0$$

Substituting this result and repeatedly applying RSP^* and $BKS2$ gives:

$$S_0 \;=\; (0t \cdot ((0rt1 + 0rt2)^* 0a) \cdot (1t \cdot ((1rt1 + 1rt2)^* 1a)))^* \delta$$

Observe that this equation conforms to the intuitive notion of what the sender should do. First, it sends a zero message; if necessary, it retransmits this message until it reseives an acknowledgement; then, it repeats this behavior for a one message, after which it starts all over again. Similar to the calculations above, the following equation can be derived for the reciever:

$$R_0 \;=\; \lambda^{\{0r,1r\}}_{0r}(R) \;=\; (0a \cdot ((0na1 + 0na2)^* 1a) + 1na1 + 1na2)^* \delta$$

The second step is to hide the internal behavior of the sender and receiver in order to obtain their observable behavior. Since the two channels do not have internal places, their observable behavior is already given by the expressions above. The axioms for the abstraction operator plus AT yield the following results:

$$\begin{aligned}
TS_0 &= \tau_{\{0s,0w,1s,1w\}}(S_0)\\
&= ((i? \mid 0m1!) \cdot ((\bot a? \mid 0m1! + 1a2? \mid 0m1!)^* 0a2?) \cdot\\
&\qquad (i? \mid 1m1!) \cdot ((\bot a? \mid 1m1! + 0a2? \mid 1m1!)^* 1a2?))^* \delta\\
TR_0 &= \tau_{\{0r,1r\}}(R_0)\\
&= ((0m2? \mid 0a1! \mid o!) \cdot ((\bot m? \mid 0a1! + 0m2? \mid 0a1!)^* (1m2? \mid 1a1! \mid o!))\\
&\qquad\quad + \bot m? \mid 1a1! + 1m2? \mid 1a1!)^* \delta
\end{aligned}$$

This completes the first part of the verification. Summarizing, $\mathcal{H}[\![\text{sen}]\!] = TS_0$, $\mathcal{H}[\![\text{rec}]\!] = TR_0$, $\mathcal{H}[\![\text{mc}]\!] = M_0$, and $\mathcal{H}[\![\text{ac}]\!] = R_0$.

The second part of the verification is to use the expressions derived for $\mathcal{H}[\![\text{sen}]\!]$, $\mathcal{H}[\![\text{rec}]\!]$, $\mathcal{H}[\![\text{mc}]\!]$, and $\mathcal{H}[\![\text{ac}]\!]$ to determine an expression for $\mathcal{H}[\![\text{abp}]\!]$, the observable behavior of the subnet "abp." The result should satisfy the specification given in Figure 4. As the first part, $\mathcal{H}[\![\text{abp}]\!]$ is calculated in two steps. First, an expression is calculated for $X_0 = \lambda^{I}_{\emptyset}(TS_0 \parallel M_0 \parallel TR_0 \parallel A_0)$, where I is equal to $\{0m1, 1m1, 0m2, \bot m, 1m2, 0a1, 1a1, 0a2, \bot a, 1a2\}$. Second, the internal behavior of X_0 is hidden. The calculations of the first step are tedious, but not very complicated. In principle, one just repeatedly applies the expansion theorem (Theorem 4.5) and the axioms of PTNA. Figure 6 shows the transition relation of X_0. With RSP^* and $BKS2$, the following equations are obtained for X_0–X_9:

$$
\begin{aligned}
X_0 &= (i? \mid 0m1!)\cdot X_1\\
X_1 &= (((0m1? \mid \bot m!)\cdot(\bot m? \mid 1a1!)\cdot((1a1? \mid 1a2!)\cdot(1a2? \mid 0m1!)\\
&\qquad +(1a1? \mid \bot a!)\cdot(\bot a? \mid 0m1!)))^{*}(0m1? \mid 0m2!))\cdot X_2\\
X_2 &= (0m2? \mid 0a1! \mid o!)\cdot X_3\\
X_3 &= (((0a1? \mid \bot a!)\cdot(\bot a? \mid 0m1!)\cdot((0m1? \mid 0m2!)\cdot(0m2? \mid 0a1!)\\
&\qquad +(0m1? \mid \bot m!)\cdot(\bot m? \mid 0a1!)))^{*}(0a1? \mid 0a2!))\cdot X_4\\
X_4 &= 0a2?\cdot X_5\\
X_5 &= (i? \mid 1m1!)\cdot X_6\\
X_6 &= (((1m1? \mid \bot m!)\cdot(\bot m? \mid 0a1!)\cdot((0a1? \mid 0a2!)\cdot(0a2? \mid 1m1!)\\
&\qquad +(0a1? \mid \bot a!)\cdot(\bot a? \mid 1m1!)))^{*}(1m1? \mid 1m2!))\cdot X_7\\
X_7 &= (1m2? \mid 1a1! \mid o!)\cdot X_8\\
X_8 &= (((1a1? \mid \bot a!)\cdot(\bot a? \mid 1m1!)\cdot((1m1? \mid 1m2!)\cdot(1m2? \mid 1a1!)\\
&\qquad +(1m1? \mid \bot m!)\cdot(\bot m? \mid 1a1!)))^{*}(1a1? \mid 1a2!))\cdot X_9\\
X_9 &= 1a2?\cdot X_0
\end{aligned}
$$

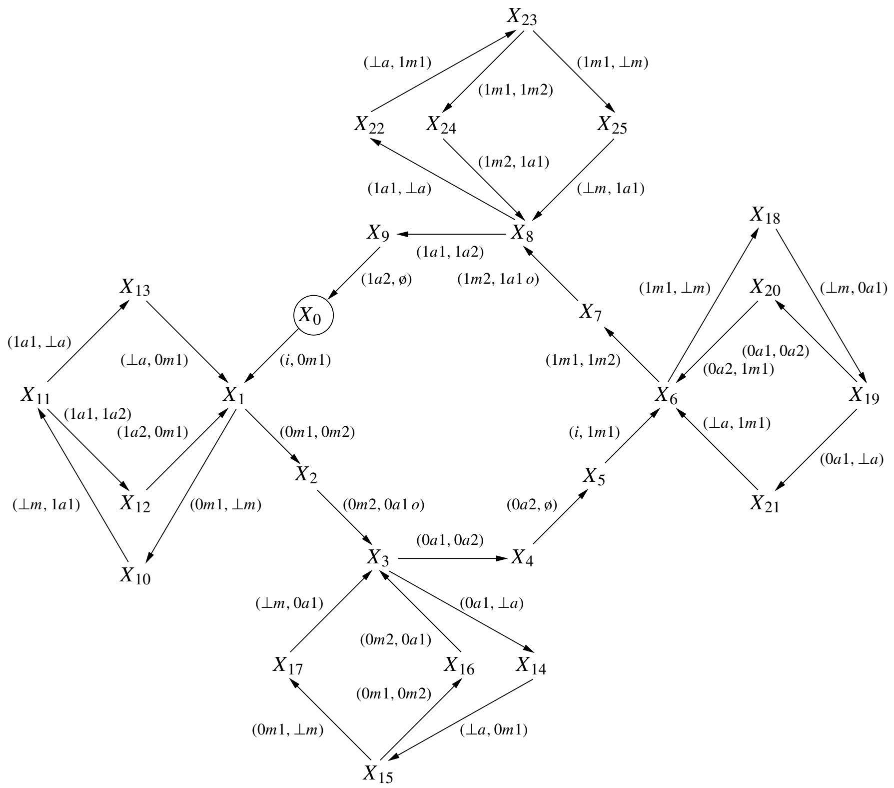

Fig. 6. The transition relation of the Alternating-Bit Protocol.

The final step is to hide the internal places I. Use the axioms for the abstraction operator plus AT, $B1$, $A3$, and FIR to obtain the following result:

$$\begin{aligned} ABP &= \tau_I(X_0) \\ &= i?\cdot((\tau\cdot\tau\cdot(\tau\cdot\tau+\tau\cdot\tau))^*\tau)\cdot\tau_I(X_2) \\ &= i?\cdot(\tau^*\tau)\cdot\tau_I(X_2) \\ &= i?\cdot(\tau+\tau\cdot\tau)\cdot\tau_I(X_2) \\ &= i?\cdot\tau_I(X_2) \end{aligned}$$

Repeating this derivation for X_2–X_9 and substituting the result in the equation above yields:

$$ABP = i?\cdot o!\cdot i?\cdot o!\cdot ABP$$

So, by RSP^*,

$$ABP = (i?\cdot o!\cdot i?\cdot o!)^*\delta$$

From the observation that, for any process x, $x^*\delta = (x\cdot x)\cdot(x^*\delta)$, which implies $x^*\delta = (x\cdot x)^*\delta$, it follows that

$$ABP = (i?\cdot o!)^*\delta.$$

This completes the verification of the ABP. The algebraic term which is derived for its observable behavior satisfies the specification given in Figure 4.

The ABP and the one-place buffer. As mentioned, the theory developed in this paper can be used to determine whether two hierarchical P/T nets have the same observable behavior. In the previous paragraph, an algebraic expression has been derived for the observable behavior of the ABP. In this paragraph, it is shown that the one-place buffer of Figure 5 has the same observable behavior. Thus, on a high level of abstraction, the one-place buffer and the ABP are equivalent.

Let BUF denote the observable behavior of the one-place buffer, $\mathcal{H}[\![\text{buf}]\!]$, where "buf" is the net shown in Figure 5. Let $I = \{b, c\}$ and let $B = (i?|c?|b!)^*\delta \,\|\, (b?|c!|o!)^*\delta$. As usual, the state operator is written without superscript I. Use Property 4.6 and the axioms for the causal state operator.

$$B_0 = \lambda_c(B) = (i?\,|\,c?\,|\,b!)\cdot\lambda_b(B) = (i?\,|\,c?\,|\,b!)\cdot(b?\,|\,c!\,|\,o!)\cdot B_0$$

RSP^*yields:

$$B_0 = ((i?\,|\,c?\,|\,b!)\cdot(b?\,|\,c!\,|\,o!))^*\delta$$

Hiding the internal places gives:

$$BUF = \tau_I(B_0) = ((i?\,|\,\tau\,|\,\tau)\cdot(\tau\,|\,\tau\,|\,o!))^*\delta = (i?\cdot o!)^*\delta$$

It follows that BUF equals ABP as derived in the previous paragraph. It means that the ABP and the one-place buffer have indeed the same observable behavior, and are thus equivalent.

High-level Petri nets. So far, only P/T nets have been considered. However, in practice, high-level nets, or colored nets, extended to data are used. In colored nets, tokens have values. As long as the values of tokens range over a finite domain, the results presented in this paper can be simply extended to data. For example, if messages in the ABP are taken from some finite data domain D, one could specify its behavior as follows: $ABP = (+\, d :\ d \in D :\ (i(d)?\cdot o(d)!)^*\delta)$, where $i(d)?$ means the consumption of a token (message) with value d from place i, and $o(d)!$ means the production of a token with value d. All calculations given above can be easily adapted to incorporate data. Furthermore, the P/T net of the ABP can be simplified by adding the bit which is used to mark messages and acknowledgements to the value of the tokens. Thus, the explicit distinction made in the current net is not necessary anymore.

In case of infinite data domains, the results of this paper must be adapted to an algebraic formalism which supports data, such as for example PSF [21].

7 Concluding Remarks and Future Work

The first part of this paper gives an algebraic semantics for P/T nets which is consistent with their usual interleaving semantics. The second part gives an algebraic semantics for the complete operational behavior of hierarchical P/T nets, as well as a semantics for their high-level, observable behavior. The latter can be used to determine whether a hierarchical net satisfies some algebraic specification of its observable behavior and, thus, to determine whether two hierarchical nets can be considered equivalent.

Although the first results appear to be promising, it is necessary to further investigate the theoretical foundation and applicability of the approach presented in this paper. It is interesting to study the meaning of an arbitrary PTNA$^{(\tau)}$ term in P/T-net theory. Furthermore, it is worthwile to investigate the meaning of results from Petri-net theory, such as place and transition invariants, in the theories PTNA$^{(\tau)}$.

There are several interesting ways to extend the results presented in this paper. It has already briefly been mentioned how they can be extended to colored nets. Furthermore, it seems worthwile to investigate other hierarchical constructs than the one presented in this paper, and maybe time or stochastic aspects.

Finally, it is interesting to look at other semantics than the interleaving semantics. It is straightforward to extend the results to a step semantics, in which multiple transitions can fire simultaneously. It is only necessary to define the synchronous-merge operator on processes in the same way as the communication merge is defined in [2]. A true concurrency semantics appears to be another interesting candidate for future investigation.

Acknowledgements. The authors want to thank Wil van der Aalst, Jos Baeten, Roland Bol, Kees van Hee, Sjouke Mauw, and Paul Rambags for the many fruitful discussions and their valuable suggestions. Special thanks go to Paul Rambags for his careful reading of an earlier version of this paper.

References

1. ASPT Foundation, Eindhoven University of Technology. *ExSpect 4.2 User Manual*, 1994.
2. J.C.M. Baeten and J.A. Bergstra. Non Interleaving Process Algebra. In *proc. CONCUR '93*, *LNCS* 715, pages 308–323. Springer–Verlag, 1993.
3. J.C.M. Baeten and C. Verhoef. A Congruence Theorem for Structured Operational Semantics with Predicates. In *proc. CONCUR '93*, *LNCS* 715, pages 477–492. Springer–Verlag, 1993.
4. J.C.M. Baeten and W.P. Weijland. *Process Algebra*, *Cambridge Tracts in Theoretical Computer Science* 18. Cambridge University Press, 1990.
5. T. Basten and M. Voorhoeve. An Algebraic Semantics for Hierarchical P/T Nets. Computing Science Report, Eindhoven University of Technology, 1995. To appear.
6. J.A. Bergstra, I. Bethke, and A. Ponse. Process Algebra with Iteration and Nesting. *The Computer Journal*, 37(4):241–258, 1994.
7. J.A. Bergstra and J.W. Klop. The Algebra of Recursively Defined Processes and the Algebra of Regular Processes. In *proc. ICALP '84*, *LNCS* 172, pages 82–95. Springer–Verlag, 1984.
8. E. Best, R. Devillers, and J.G. Hall. The Box Calculus: A New Causal Algebra with Multi-label Communication. In *APN '92*, *LNCS* 609, pages 21–69. Springer–Verlag, 1992.

9. G. Boudol, G. Roucairol, and R. de Simone. Petri Nets and Algebraic Calculi of Processes. In *APN '85*, *LNCS* 222, pages 41–58. Springer–Verlag, 1985.
10. W. Brauer, R. Gold, and W. Vogler. A Survey of Behaviour and Equivalence Preserving Refinements of Petri Nets. In *APN '90*, *LNCS* 483, pages 1–46. Springer–Verlag, 1990.
11. P. Degano, R. De Nicola, and U. Montanari. A Distributed Operational Semantics for CCS Based on Condition/Event Systems. *Acta Informatica*, 26(1/2):59–91, October 1988.
12. C. Dietz and G. Schreibert. A Term Representation of P/T Systems. In *proc. ATPN '94*, *LNCS* 815, pages 239–257. Springer–Verlag, 1994.
13. R. van Glabbeek. *Comparative Concurrency Semantics and Refinements of Actions*. PhD thesis, Free University, Amsterdam, The Netherlands, 1990.
14. R. van Glabbeek. What is Branching Time Semantics and Why to Use It? In *Bulletin of the EATCS* 53, pages 191–198. June 1994.
15. R.J. van Glabbeek and F.W. Vaandrager. Petri Net Models for Algebraic Theories of Concurrency. In *proc. PARLE '87, vol. II*, *LNCS* 259, pages 224–242. Springer–Verlag, 1987.
16. U. Goltz. On Representing CCS Programs by Finite Petri Nets. In *proc. MFCS '88*, *LNCS* 324, pages 339–350. Springer–Verlag, 1988.
17. K.M. van Hee. *Information Systems Engineering: A Formal Approach*. Cambridge University Press, 1994.
18. C.A.R. Hoare. *Communicating Sequential Processes*. Prentice–Hall International, 1985.
19. K. Jensen. *Coloured Petri Nets. Basic Consepts, Analysis Methods and Practical Use*, *EATCS monographs on Theoretical Computer Science* 28. Springer–Verlag, 1992.
20. S.C. Kleene. Representation of Events in Nerve Nets and Finite Automata. In *Automata Studies*, *Annals of Mathematics Studies* 34, pages 3–41. Princeton University Press, 1956.
21. S. Mauw and G.J. Veltink, editors. *Algebraic Specification of Communication Protocols*, *Cambridge Tracts in Theoretical Computer Science* 36. Cambridge University Press, 1993.
22. Meta Software Corporation, Cambridge, Massachusetts, USA. *Design/CPN Manual*, 1991.
23. R. Milner. *A Calculus of Communcating Systems*, *LNCS* 92. Springer–Verlag, 1980.
24. U. Montanari and D. Yankelevich. Combining CCS and Petri Nets via Structural Axioms. *Fundamenta Informaticae*, 20(1–3):193–229, May 1994.
25. E.-R. Olderog. Petri Nets and Algebraic Calculi of Processes. In *APN '87*, *LNCS* 266, pages 196–223. Springer–Verlag, 1987.
26. L. Pomello, G. Rozenberg, and C. Simone. A Survey of Equivalence Notions for Net Based Systems. In *APN '92*, *LNCS* 609, pages 410–472. Springer–Verlag, 1992.
27. W. Reisig. *Petri Nets: An Introduction*, *EATCS monographs on Theoretical Computer Science* 4. Springer–Verlag, 1985.
28. D. Taubner. *Finite Representations of CCS and TCSP Programs by Automata and Petri Nets*, *LNCS* 369. Springer–Verlag, 1989.