Structural characterizations of sound workflow nets

Eindhoven University of Technology
Department of Mathematics and Computing Science

Structural Characterizations of Sound Workflow Nets

by

W.M.P. van der Aalst

96/23

ISSN 0926-4515

editors: prof.dr. R.C. Backhouse
prof.dr. J.C.M. Baeten

Reports are available at:
http://www.win.tue.nl/win/cs

Computing Science Reports 96/23
Eindhoven, December 1996

Structural Characterizations of Sound Workflow Nets

W.M.P. van der Aalst
Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands, telephone: -31 40 2474295, e-mail: wsinwa@win.tue.nl

Workflow management systems facilitate the everyday operation of business processes by taking care of the logistic control of work. In contrast to traditional information systems, they attempt to support frequent changes of the workflows at hand. Therefore, the need for analysis methods to verify the correctness of workflows is becoming more prominent. In this paper we present a method based on Petri nets. This analysis method exploits the structure of the Petri net to find potential errors in the design of the workflow. Moreover, the analysis method allows for the compositional verification of workflows.

Keywords: Petri nets; free-choice Petri nets; workflow management systems; analysis of workflows; business process reengineering; analysis of Petri nets; compositional analysis.

1 Introduction

Workflow management systems (WFMS) are used for the modeling, analysis, enactment, and coordination of structured business processes by groups of people. Business processes supported by a WFMS are *case-driven*, i.e., tasks are executed for specific cases. Approving loans, processing insurance claims, billing, processing tax declarations, handling traffic violations and mortgaging, are typical case-driven processes which are often supported by a WFMS. These case-driven processes, also called *workflows*, are marked by three dimensions: (1) the process dimension, (2) the resource dimension, and (3) the case dimension (see Figure 1). The process dimension is concerned with the partial ordering of tasks. The tasks which need to be executed are identified and the routing of cases along these tasks is determined. Conditional, sequential, parallel and iterative routing are typical structures specified in the process dimension. Tasks are executed by resources. Resource are human (e.g. employee) and/or non-human (e.g. device, software, hardware). In the resource dimension these resources are classified by identifying roles (resources classes based on functional characteristics) and organizational units (groups, teams or departments). Both the process dimension and the resource dimension are generic, i.e., they are not tailored towards a specific case. The third dimension of a workflow is concerned with individual cases which are executed according to the process definition (first dimension) by the proper resources (second dimension).

Managing workflows is not a new idea. Workflow control techniques have existed for decades and many management concepts originating from production and logistics are also

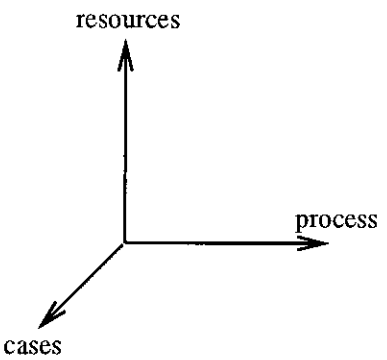

Figure 1: The three dimensions of workflow.

applicable in a workflow context. However, just recently, commercially available generic WFMS's have become a reality. Although these systems have been applied successfully, contemporary WFMS's have at least two important drawbacks. First of all, today's systems do not scale well, have limited fault tolerance and are inflexible in terms of interoperating with other systems. Secondly, a solid theoretical foundation is missing. Most of the more than 250 commercially available WFMS's use a vendor-specific ad-hoc modeling technique to design workflows. In spite of the efforts of the Workflow Management Coalition ([20]) real standards are missing. The absence of formalized standards hinders the development of tool-independent analysis techniques. As a result, contemporary WFMS's do not facilitate advanced analysis methods to determine the correctness of a workflow.

As many researchers have indicated ([9, 14, 21]), Petri nets constitute a good starting point for a solid theoretical foundation for workflow management. In this paper we focus on the process dimension. We use Petri nets to specify the partial ordering of tasks. Based on a Petri-net-based representation of the workflow process, we tackle the problem of verification. We will provide techniques to verify the so-called *soundness property*. A workflow process is sound if, for any case, the process terminates properly, i.e., termination is guaranteed, there are no dangling references, and deadlock and livelock are absent. We will show that in general this dynamic property can be checked in polynomial time. Moreover, we identify suspicious constructs which may endanger the correctness of a workflow process. Finally, we show that the approach presented in this paper allows for the compositional verification of workflow processes, i.e., the correctness of a process can be decided by partitioning it into sound subprocesses.

2 Petri nets

This section introduces the basic Petri net terminology and notations. Readers familiar with Petri nets can skip this section.[1]

Historically speaking, Petri nets originate from the early work of Carl Adam Petri ([17]). Since then the use and study of Petri nets has increased considerably. For a review of the history of Petri nets and an extensive bibliography the reader is referred to Murata [15].

The classical Petri net is a directed bipartite graph with two node types called *places* and *transitions*. The nodes are connected via directed *arcs*. Connections between two nodes of the same type are not allowed. Places are represented by circles and transitions by rectangles.

Definition 1 (Petri net) *A Petri net is a triple* (P, T, F)*:*

- P *is a finite set of places,*
- T *is a finite set of transitions* $(P \cap T = \emptyset)$*,*
- $F \subseteq (P \times T) \cup (T \times P)$ *is a set of arcs (flow relation)*

A place p is called an *input place* of a transition t iff there exists a directed arc from p to t. Place p is called an *output place* of transition t iff there exists a directed arc from t to p. We use $\bullet t$ to denote the set of input places for a transition t. The notations $t\bullet$, $\bullet p$ and $p\bullet$ have similar meanings, e.g. $p\bullet$ is the set of transitions sharing p as an input place. Note that we restrict ourselves to arcs with weight 1. In the context of workflow procedures it makes no sense to have other weights, because places correspond to conditions.

At any time a place contains zero of more *tokens*, drawn as black dots. The *state*, often referred to as marking, is the distribution of tokens over places, i.e., $M \in P \rightarrow \mathbf{N}$. We will represent a state as follows: $1p_1 + 2p_2 + 1p_3 + 0p_4$ is the state with one token in place p_1, two tokens in p_2, one token in p_3 and no tokens in p_4. We can also represent this state as follows: $p_1 + 2p_2 + p_3$. To compare states we define a partial ordering. For any two states M_1 and M_2, $M_1 \leq M_2$ iff for all $p \in P$: $M_1(p) \leq M_2(p)$

The number of tokens may change during the execution of the net. Transitions are the active components in a Petri net: they change the state of the net according to the following *firing rule*:

(1) A transition t is said to be *enabled* iff each input place p of t contains at least one token.

(2) An enabled transition may *fire*. If transition t fires, then t *consumes* one token from each input place p of t and *produces* one token for each output place p of t.

[1] Note that states are represented by weighted sums and note the definition of (elementary) (conflict-free) paths.

Given a Petri net (P, T, F) and a state M_1, we have the following notations:

- $M_1 \xrightarrow{t} M_2$: transition t is enabled in state M_1 and firing t in M_1 results in state M_2
- $M_1 \rightarrow M_2$: there is a transition t such that $M_1 \xrightarrow{t} M_2$
- $M_1 \xrightarrow{\sigma} M_n$: the firing sequence $\sigma = t_1 t_2 t_3 \ldots t_{n-1}$ leads from state M_1 to state M_n, i.e., $M_1 \xrightarrow{t_1} M_2 \xrightarrow{t_2} \ldots \xrightarrow{t_{n-1}} M_n$

A state M_n is called *reachable* from M_1 (notation $M_1 \xrightarrow{*} M_n$) iff there is a firing sequence $\sigma = t_1 t_2 \ldots t_{n-1}$ such that $M_1 \xrightarrow{t_1} M_2 \xrightarrow{t_2} \ldots \xrightarrow{t_{n-1}} M_n$. Note that the empty firing sequence is also allowed, i.e., $M_1 \xrightarrow{*} M_1$.

We use (PN, M) to denote a Petri net PN with an initial state M. A state M' is a *reachable state* of (PN, M) iff $M \xrightarrow{*} M'$. Let us define some properties for Petri nets.

Definition 2 (Live) *A Petri net (PN, M) is live iff, for every reachable state M' and every transition t there is a state M'' reachable from M' which enables t.*

Definition 3 (Bounded, safe) *A Petri net (PN, M) is bounded iff, for every reachable state and every place p the number of tokens in p is bounded. The net is safe iff for each place the maximum number of tokens does not exceed 1.*

Definition 4 (Well-formed) *A Petri net PN is well-formed iff there is a state M such that (PN, M) is live and bounded.*

Paths connect nodes by a sequence of arcs.

Definition 5 (Path, Elementary, Conflict-free) *Let PN be a Petri net. A path C from a node n_1 to a node n_k is a sequence $\langle n_1, n_2, \ldots, n_k \rangle$ such that $\langle n_i, n_{i+1} \rangle \in F$ for $1 \leq i \leq k-1$. C is elementary iff, for any two nodes n_i and n_j on C, $i \neq j \Rightarrow n_i \neq n_j$. C is conflict-free iff, for any transition n_i on C, $j \neq i-1 \Rightarrow n_j \notin {}^\bullet n_i$.*

For convenience, we introduce the alphabet operator α on paths. If $C = \langle n_1, n_2, \ldots, n_k \rangle$, then $\alpha(C) = \{n_1, n_2, \ldots, n_k\}$.

Definition 6 (Strongly connected) *A Petri net is strongly connected iff, for every pair of nodes (i.e. places and transitions) x and y, there is a path leading from x to y.*

3 WF-nets

In Figure 1 we indicated that a workflow has (at least) three dimensions. The process dimension is the most prominent one, because the core of any workflow system is formed by the processes it supports. In the process dimension building blocks such as the AND-split, AND-join, OR-split and OR-join are used to model sequential, conditional, parallel and iterative routing (WFMC [20]). Clearly, a Petri net can be used to specify the routing of cases. *Tasks* are modeled by transitions and causal dependencies are modeled by places.

In fact, a place corresponds to a *condition* which can be used as pre- and/or post-conditions for tasks. An AND-split corresponds to a transition with two or more output places, and an AND-join corresponds to a transition with two or more input places. OR-splits/OR-joins correspond to places with multiple outgoing/ingoing arcs. Moreover, in [2, 3] it is shown that the Petri net approach also allows for useful routing constructs absent in many WFMS's.
A Petri net which models the process dimension of a workflow, is called a *WorkFlow net* (WF-net). It should be noted that a WF-net specifies the dynamic behavior of a single case in isolation.

Definition 7 (WF-net) *A Petri net* $PN = (P, T, F)$ *is a WF-net (Workflow net) if and only if:*

(i) *PN has two special places:* i *and* o*. Place* i *is a source place:* $\bullet i = \emptyset$*. Place* o *is a sink place:* $o\bullet = \emptyset$*.*

(ii) *If we add a transition* t^* *to PN which connects place* o *with* i *(i.e.* $\bullet t^* = \{o\}$ *and* $t^*\bullet = \{i\}$*), then the resulting Petri net is strongly connected.*

A WF-net has one input place (i) and one output place (o) because any case handled by the procedure represented by the WF-net is created if it enters the WFMS and is deleted once it is completely handled by the WFMS, i.e., the WF-net specifies the life-cycle of a case. The second requirement in Definition 7 (the Petri net extended with t^* should be strongly connected) states that for each transition t (place p) there should be a path from place i to o via t (p). This requirement has been added to avoid 'dangling tasks and/or conditions', i.e., tasks and conditions which do not contribute to the processing of cases.

Figure 2 shows a WF-net which models the processing of complaints. First the complaint is registered (task *register*), then in parallel a questionnaire is sent to the complainant (task *send_questionnaire*) and the complaint is evaluated (task *evaluate*). If the complainant returns the questionnaire within two weeks, the task *process_questionnaire* is executed. If the questionnaire is not returned within two weeks, the result of the questionnaire is discarded (task *time_out*). Based on the result of the evaluation, the complaint is processed or not. The actual processing of the complaint (task *process_complaint*) is delayed until condition *c5* is satisfied, i.e., the questionnaire is processed or a time-out has occurred. The processing of the complaint is checked via task *check_processing*. Finally, task *archive* is executed. Note that sequential, conditional, parallel and iterative routing are present in this example.

The WF-net shown in Figure 2 clearly illustrates that we focus on the process dimension. We abstract from resources, applications and technical platforms. Moreover, we also abstract from *case variables* and *triggers*. Case variables are used to resolve choices (OR-split), i.e., the choice between *processing_required* and *no_processing* is (partially) based on case variables set during the execution of task *evaluate*. The choice between *processing_OK* and *processing_NOK* is resolved by testing case variables set by *check_processing*.

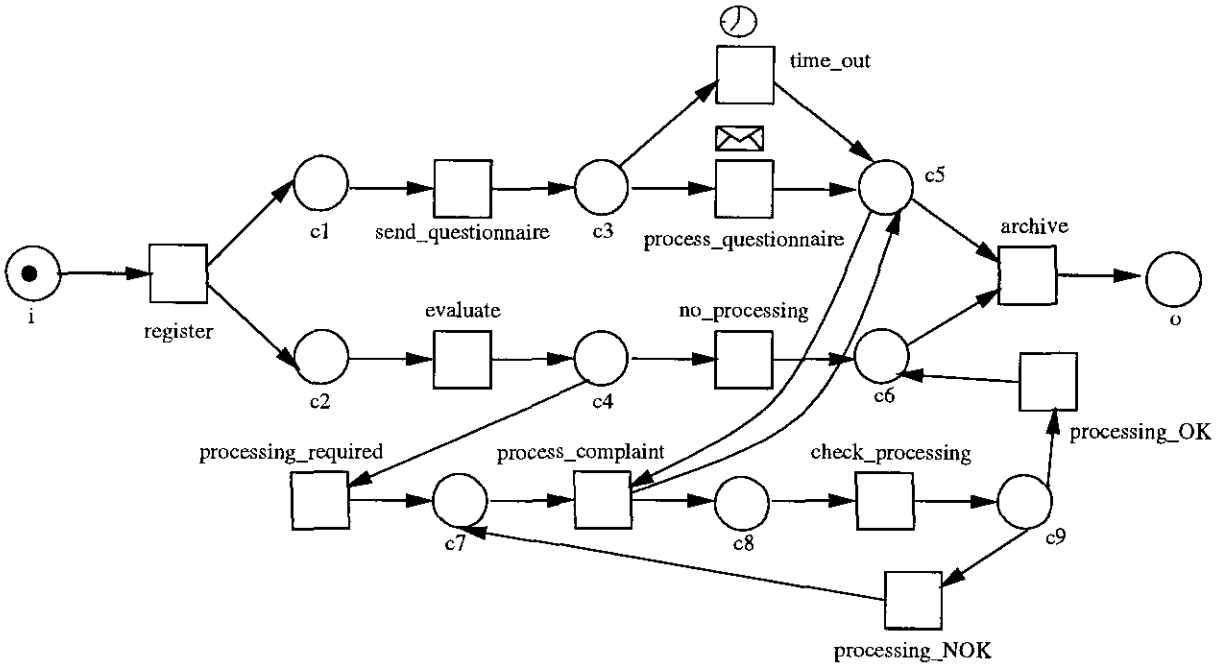

Figure 2: A WF-net for the processing of complaints.

In the WF-net we abstract from case variables by introducing non-deterministic choices in the Petri-net. If we don't abstract from this information, we would have to model the (unknown) behavior of the applications used in each of the tasks and analysis would become intractable. In Figure 2 we have indicated that *time_out* and *process_questionnaire* require triggers. The clock symbol denotes a time trigger and the envelope symbol denotes an external trigger. Task *time_out* requires a time trigger ('two weeks have passed') and *process_questionnaire* requires a message trigger ('the questionnaire has been returned'). A trigger can be seen as an additional condition which needs to be satisfied. In the remainder of this paper we abstract from these trigger conditions. We assume that the environment behaves fairly, i.e., the liveness of a transition is not hindered by the continuous absence of a specific trigger. As a result, every trigger condition will be satisfied eventually (if needed).

4 Soundness

The two requirements stated in Definition 7 can be verified statically, i.e., they only relate to the structure of the Petri net. However, there is a third requirement which should be satisfied:

> *For any case, the procedure will terminate eventually and the moment the procedure terminates there is a token in place o and all the other places are empty.*

Moreover, there should be no dead tasks, i.e., it should be possible to execute an arbitrary task by following the appropriate route though the WF-net. These two additional constraints correspond to the so-called *soundness property*.

Definition 8 (Sound) *A procedure modeled by a WF-net $PN = (P, T, F)$ is sound if and only if:*

(i) For every state M reachable from state i, there exists a firing sequence leading from state M to state o. Formally:[2]

$$\forall_M (i \xrightarrow{*} M) \Rightarrow (M \xrightarrow{*} o)$$

(ii) State o is the only state reachable from state i with at least one token in place o. Formally:

$$\forall_M (i \xrightarrow{*} M \ \wedge \ M \geq o) \Rightarrow (M = o)$$

(iii) There are no dead transitions in (PN, i). Formally:

$$\forall_{t \in T} \ \exists_{M,M'} \ i \xrightarrow{*} M \xrightarrow{t} M'$$

Note that the soundness property relates to the dynamics of a WF-net. The first requirement in Definition 8 states that starting from the initial state (state i), it is always possible to reach the state with one token in place o (state o). If we assume fairness (i.e. a transition that is enabled infinitely often will fire eventually), then the first requirement implies that eventually state o will be reached. The fairness assumption is reasonable in the context of workflow management; all choices are made (implicitly en explicitly) by applications, humans or external actors. Clearly, they should not introduce an infinite loop. The second requirement states that the moment a token is put in place o, all the other places should be empty. Sometimes the term *proper termination* is used to describe the first two requirements [12]. The last requirement states that there are no dead transitions (tasks) in the initial state i.

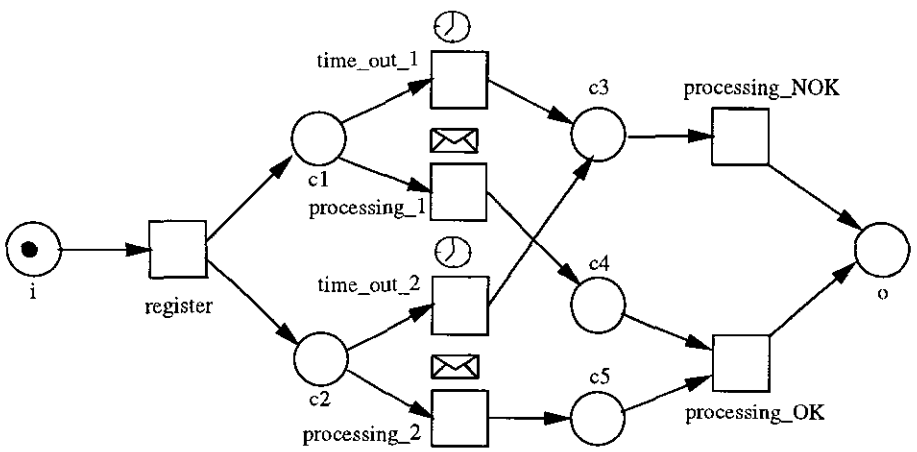

Figure 3: Another WF-net for the processing of complaints.

Figure 3 shows a WF-net which is not sound. There are several deficiencies. If *time_out_1*

[2]Note that there is an overloading of notation: the symbol i is used to denote both the *place* i and the *state* with only one token in place i (see Section 2).

and *processing_2* fire or *time_out_2* and *processing_1* fire, the WF-net will not terminate properly because a token gets stuck in *c4* or *c5*. If *time_out_1* and *time_out_2* fire, then the task *processing_NOK* will be executed twice and because of the presence of two tokens in o the moment of termination is not clear.

5 A necessary and sufficient condition for soundness

Given WF-net $PN = (P, T, F)$, we want to decide whether PN is sound. For this purpose we define an extended net $\overline{PN} = (\overline{P}, \overline{T}, \overline{F})$. $\overline{PN}$ is the Petri net that we obtain by adding an extra transition t^* which connects o and i. The extended Petri net $\overline{PN} = (\overline{P}, \overline{T}, \overline{F})$ is defined as follows:

$$\overline{P} = P$$

$$\overline{T} = T \cup \{t^*\}$$

$$\overline{F} = F \cup \{\langle o, t^* \rangle, \langle t^*, i \rangle\}$$

Figure 4 illustrates the relation between PN and $\overline{PN}$.

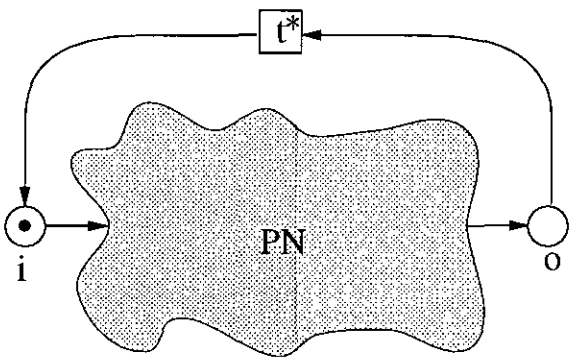

Figure 4: $\overline{PN} = (P, T \cup \{t^*\}, F \cup \{\langle o, t^* \rangle, \langle t^*, i \rangle\})$.

For an arbitrary WF-net PN and the corresponding extended Petri net $\overline{PN}$ we will prove the following result:

PN is sound if and only if $(\overline{PN}, i)$ is live and bounded.

First, we prove the 'if' direction.

Lemma 1 *If $(\overline{PN}, i)$ is live and bounded, then PN is a sound WF-net.*

Proof.
$(\overline{PN}, i)$ is live, i.e., for every reachable state M there is a firing sequence which leads to a state in which t^* is enabled. Since o is the input place of t^*, we find that for any state M reachable from state i it is possible to reach a state with at least one token in place o, i.e., requirement (i) holds. Consider an arbitrary reachable state $M' + o$, i.e., a state with at least one token in place o. In this state t^* is enabled. If t^* fires, then the state $M' + i$ is

reached. Since $(\overline{PN}, i)$ is also bounded, $M' + i \geq i$ implies $M' + i = i$, i.e., M' should be equal to the empty state. Hence requirement (ii) also holds and proper termination is guaranteed. Requirement (iii) follows directly from the fact that $(\overline{PN}, i)$ is live. Hence, PN is a sound WF-net. □

To prove the 'only if' direction, we first show that the extended net is bounded.

Lemma 2 *If PN is sound, then $(\overline{PN}, i)$ is bounded.*

Proof.
Assume that PN is sound and (PN, i) not bounded. Since PN is not bounded there are two states M_i and M_j such that $i \xrightarrow{*} M_i$, $M_i \xrightarrow{*} M_j$ and $M_j > M_i$. (See for example the proof that the coverability tree is finite in Peterson [16] (Theorem 4.1).) However, since PN is sound we know that there is a firing sequence σ such that $M_i \xrightarrow{\sigma} o$. Therefore, there is a state M such that $M_j \xrightarrow{\sigma} M$ and $M > o$. Hence, it is not possible that PN is both sound and not bounded. So if PN is sound, then (PN, i) is bounded.
From the fact that PN is sound and (PN, i) is bounded, we can deduce that $(\overline{PN}, i)$ is bounded. If transition t^* in $\overline{PN}$ fires, the net returns to the initial state i. □

Now we can prove that $(\overline{PN}, i)$ is live.

Lemma 3 *If PN is sound, then $(\overline{PN}, i)$ is live.*

Proof.
Assume PN is sound. By Lemma 2 we know that $(\overline{PN}, i)$ is bounded. Because PN is sound we know that state i is a so-called home-marking of $\overline{PN}$. So for every state M' reachable from $(\overline{PN}, i)$ it is possible to return to state i. In the original net (PN, i), it is possible to fire an arbitrary transition t (requirement (iii)). This is also the case in the modified net. Therefore, $(\overline{PN}, i)$ is live because for every state M' reachable from $(\overline{PN}, i)$ it is possible to reach a state which enables an arbitrary transition t. □

Theorem 1 *A WF-net PN is sound if and only if $(\overline{PN}, i)$ is live and bounded.*

Proof.
It follows directly from Lemma 1, 2 and 3. □

We can use standard Petri-net-based analysis tools to verify that the WF-net shown in Figure 2 is live and bounded. Therefore, the workflow process specified by this WF-net is guaranteed to behave properly (cf. Definition 8). Theorem 1 is an extension of the results presented in [1, 19]. In [1] we restrict ourselves to free-choice WF-nets. Independently, Straub and Hurtado [19] found necessary and sufficient conditions for soundness of COPA nets. (COPA nets correspond to a subclass of free-choice Petri nets.)

Perhaps surprisingly, the verification of the soundness property boils down to checking whether the extended Petri net is live and bounded! This means that we can use standard Petri-net-based analysis tools to decide soundness.

6 Structural characterization of soundness

Theorem 1 gives a useful characterization of the quality of a workflow process definition. However, there are a number of problems:

- For a complex WF-net it may be intractable to decide soundness. (For arbitrary WF-nets liveness and boundedness are decidable but also EXPSPACE-hard, cf. Cheng, Esparza and Palsberg [6].)
- Soundness is a minimal requirement. Readability and maintainability issues are not addressed by Theorem 1.
- Theorem 1 does not show how a non-sound WF-net should be modified, i.e., it does not identify constructs which invalidate the soundness property.

These problems stem from the fact that the definition of soundness relates to the dynamics of a WF-net while the workflow designer is concerned with the static structure of the WF-net. Therefore, it is interesting to investigate structural characterizations of sound WF-nets. For this purpose we introduce two interesting subclasses of WF-nets: free-choice WF-nets and well-structured WF-nets.

6.1 Free-choice WF-nets

Most of the WFMS's available at the moment, abstract from states between tasks, i.e., states are not represented explicitly. These WFMS's use building blocks such as the AND-split, AND-join, OR-split and OR-join to specify workflow procedures. The AND-split and the AND-join are used for parallel routing. The OR-split and the OR-join are used for conditional routing. Because these systems abstract from states, every choice is made *inside* an OR-split building block. If we model an OR-split in terms of a Petri net, the OR-split corresponds to a number of transitions sharing the same set of input places. This means that for these WFMS's, a workflow procedure corresponds to a *free-choice Petri net.*

Definition 9 (Free-choice) *A Petri net is a free-choice Petri net iff, for every two transitions* t_1 *and* t_2, $\bullet t_1 \cap \bullet t_2 \neq \emptyset$ *implies* $\bullet t_1 = \bullet t_2$.

It is easy to see that a process definition composed out of AND-splits, AND-joins, OR-splits and OR-joins is free-choice. If two transitions t_1 and t_2 share an input place ($\bullet t_1 \cap \bullet t_2 \neq \emptyset$), then they are part of an OR-split, i.e., a 'free choice' between a number of alternatives. Therefore, the sets of input places of t_1 and t_2 should match ($\bullet t_1 = \bullet t_2$). Figure 3 shows a free-choice WF-net. The WF-net shown in Figure 2 is not free-choice; *archive* and *process_complaint* share an input place but the two corresponding input sets differ.

We have evaluated many WFMS's and just one of these systems (COSA [18]) allows for a construction which is comparable to a non-free choice WF-net. Therefore, it makes sense to consider free-choice Petri nets. Clearly, parallelism, sequential routing, conditional routing and iteration can be modeled without violating the free-choice property. Another reason for restricting WF-nets to free-choice Petri nets is the following. If we allow non-free-choice Petri nets, then the choice between conflicting tasks *may* be influenced by the order

in which the preceding tasks are executed. The routing of a case should be independent of the order in which tasks are executed. A situation where the free-choice property is violated is often a mixture of parallelism and choice. Figure 5 shows such a situation. Firing transition *t1* introduces parallelism. Although there is no real choice between *t2* and *t5* (*t5* is not enabled), the parallel execution of *t2* and *t3* results in a situation where *t5* is not allowed to occur. However, if the execution of *t2* is delayed until *t3* has been executed, then there is a real choice between *t2* and *t5*. In our opinion parallelism itself should be separated from the choice between two or more alternatives. Therefore, we consider the non-free-choice construct shown in Figure 5 to be improper. In literature, the term *confusion* is often used to refer to the situation shown in Figure 5.

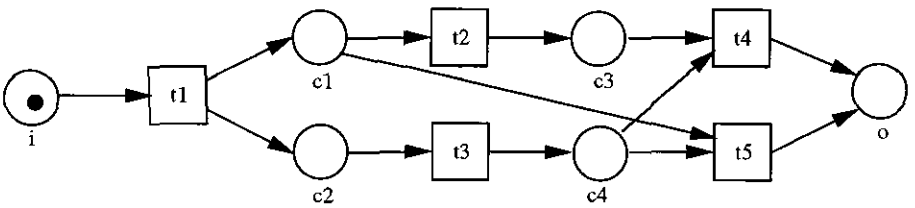

Figure 5: A non-free-choice WF-net containing a mixture of parallelism and choice.

Free-choice Petri nets have been studied extensively (cf. Best [5], Desel and Esparza [8, 7, 10], Hack [13]) because they seem to be a good compromise between expressive power and analyzability. It is a class of Petri nets for which strong theoretical results and efficient analysis techniques exist. For example, the well-known Rank Theorem (Desel and Esparza [8]) allows us to formulate the following corollary.

Corollary 1 *The following problem can be solved in polynomial time.*
Given a free-choice WF-net, to decide if it is sound.

Proof.
Let PN be a free-choice WF-net. The extended net $\overline{PN}$ is also free-choice. Therefore, the problem of deciding whether $(\overline{PN}, i)$ is live and bounded can be solved in polynomial time (Rank Theorem [8]). By Theorem 1, this corresponds to soundness. □

Corollary 1 shows that, for free-choice nets, there are efficient algorithms to decide soundness. Moreover, a sound free-choice WF-net is guaranteed to be safe.

Lemma 4 *A sound free-choice WF-net is safe.*

Proof.
Let PN be a sound free-choice WF-net. $\overline{PN}$ is the Petri net PN extended with a transition connecting o and i. $\overline{PN}$ is free-choice and well-formed. Hence, $\overline{PN}$ is covered by state-machines (S-components). Each place is part of such a state-machine component. Clearly, i and o are nodes of any state-machine component. Hence, for each place p there is a

semi-positive invariant with weights 0 or 1 which assigns a positive weight to p, i and o. Therefore, $\overline{PN}$ is safe and so is PN. □

Safeness is a desirable property, because it makes no sense to have multiple tokens in a place representing a condition. A condition is either true (1 token) or false (no tokens).

Although most WFMS's only allow for free-choice workflows, free-choice WF-nets are not a completely satisfactory structural characterization of 'good' workflows. On the one hand, there are non-free-choice WF-nets which correspond to sensible workflows (cf. Figure 2). On the other hand there are sound free-choice WF-nets which make no sense. Nevertheless, the free-choice property is a desirable property. If a workflow can be modeled as a free-choice WF-net, one should do so. A workflow specification based on a free-choice WF-net can be enacted by most workflow systems. Moreover, a free-choice WF-net allows for efficient analysis techniques and is more easy to understand. Non-free-choice constructs such as the construct shown in Figure 5 are a potential source of anomalous behavior (e.g. deadlock) which is difficult to trace.

6.2 Well-structured WF-nets

Another approach to obtain a structural characterization of 'good' workflows, is to balance AND/OR-splits and AND/OR-joins. Clearly, two parallel flows initiated by an AND-split, should not be joined by an OR-join. Two alternative flows created via an OR-split, should not be synchronized by an AND-join. As shown in Figure 6, an AND-split should be complemented by an AND-join and an OR-split should be complemented by an OR-join.

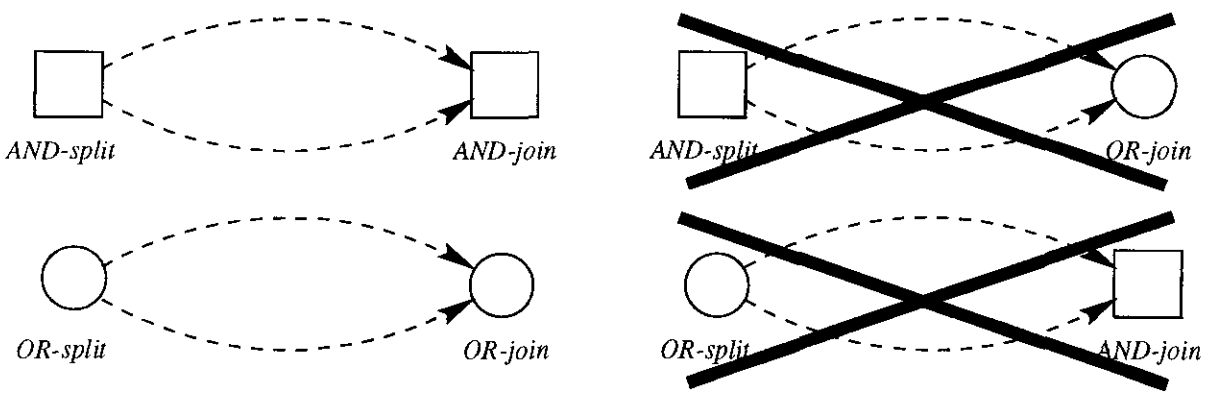

Figure 6: Good and bad constructions.

One of the deficiencies of the WF-net shown in Figure 3 is the fact that the AND-split *register* is complemented by the OR-join *c3* or the OR-join o. To formalize the concept illustrated in Figure 6 we give the following definition.

Definition 10 (Well-handled) *A Petri net PN is well-handled iff, for any pair of nodes x and y such that one of the nodes is a place and the other a transition and for any pair of elementary paths C_1 and C_2 leading from x to y, $\alpha(C_1) \cap \alpha(C_2) = \{x, y\} \Rightarrow C_1 = C_2$.*

Note that the WF-net shown in Figure 3 is not well-handled. A Petri net which is well-handled has a number of nice properties, e.g. strong connectedness and well-formedness coincide.

Lemma 5 *A strongly connected well-handled Petri net is well-formed.*

Proof.
Let PN be a strongly connected well-handled Petri net. Clearly, there are no circuits that have PT-handles nor TP-handles ([11]). Therefore, the net is structurally bounded (See Theorem 3.1 in [11]) and structurally live (See Theorem 3.2 in [11]). Hence, PN is well-formed. □

Clearly, well-handledness is a desirable property for any WF-net PN. Moreover, we also require the extended $\overline{PN}$ to be well-handled. We impose on this additional requirement for the following reason. Suppose we want to use PN as a part of a larger WF-net PN'. PN' is the original WF-net extended with an 'undo-task'. See Figure 7. Transition *undo* corresponds to the undo-task, transitions $t1$ and $t2$ have been added to make PN' a WF-net. It is undesirable that transition *undo* violates the well-handledness property of the original net. However, PN' is well-handled iff $\overline{PN}$ is well-handled. Therefore, we require $\overline{PN}$ to be well-handled. We use the term *well-structured* to refer to WF-nets whose extension is well-handled.

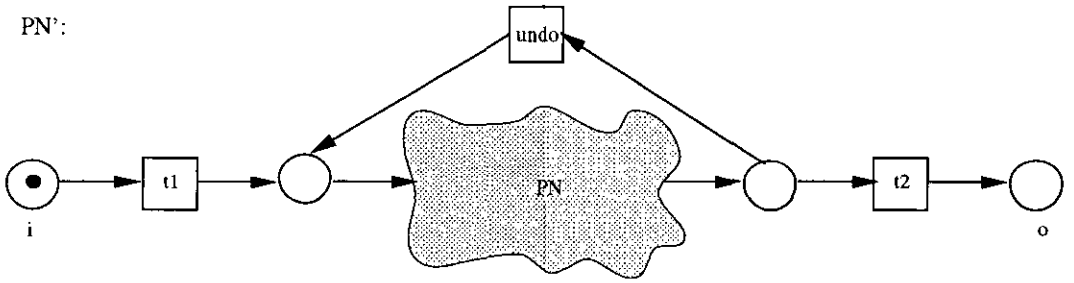

Figure 7: The WF-net PN' is well-handled iff $\overline{PN}$ is well-handled.

Definition 11 (Well-structured) *A WF-net PN is well-structured iff $\overline{PN}$ is well-handled.*

Well-structured WF-nets have a number of desirable properties. Soundness can be verified in polynomial time and a sound well-structured WF-net is safe. To prove these properties we use some of the results obtained for *elementary extended non-self controlling nets.*

Definition 12 (Elementary extended non-self controlling) *A Petri net PN is elementary extended non-self controlling (ENSC) iff, for every pair of transitions t_1 and t_2 such that $\bullet t_1 \cap \bullet t_2 \neq \emptyset$, there does not exist an elementary path C leading from t_1 to t_2 such that $\bullet t_1 \cap \alpha(C) = \emptyset$.*

Theorem 2 *Let PN be a WF-net. If PN is well-structured, then $\overline{PN}$ is elementary extended non-self controlling.*

Proof.
Assume that $\overline{PN}$ is not elementary extended non-self controlling. This means that there is a pair of transitions t_1 and t_k such that $\bullet t_1 \cap \bullet t_k \neq \emptyset$ and there exist an elementary path $C = \langle t_1, p_2, t_2, \ldots, p_k, t_k \rangle$ leading from t_1 to t_k and $\bullet t_1 \cap \alpha(C) = \emptyset$. Let $p_1 \in \bullet t_1 \cap \bullet t_k$. $C_1 = \langle p_1, t_k \rangle$ and $C_2 = \langle p_1, t_1, p_2, t_2, \ldots, p_k, t_k \rangle$ are paths leading from p_1 to t_k. (Note that C_2 is the concatenation of $\langle p_1 \rangle$ and C.) Clearly, C_1 is elementary. We will also show that C_2 is elementary. C is elementary, and $p_1 \notin \alpha(C)$ because $p_1 \in \bullet t_1$. Hence, C_2 is also elementary. Since C_1 and C_2 are both elementary paths, $C_1 \neq C_2$ and $\alpha(C_1) \cap \alpha(C_2) = \{p_1, t_k\}$, we conclude that $\overline{PN}$ is not well-handled. □

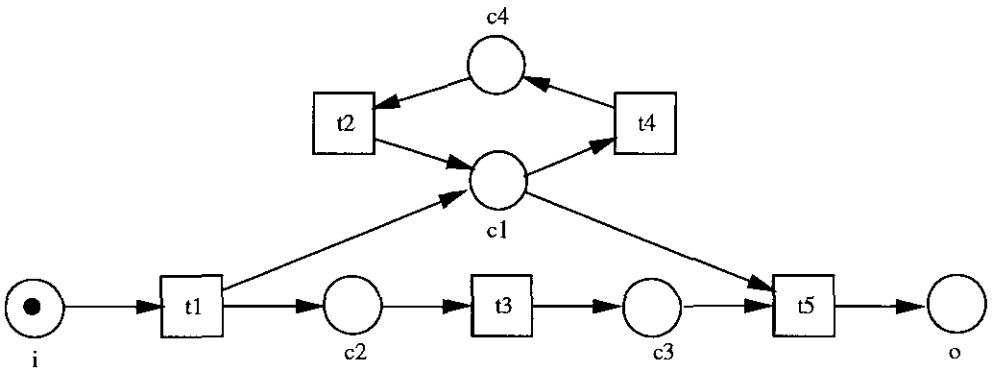

Figure 8: A well-structured WF-net.

Consider for example the WF-net shown in Figure 8. The WF-net is well-structured and, therefore, also elementary extended non-self controlling. However, the net is not free-choice. Nevertheless, it is possible to verify soundness for such a WF-net very efficiently.

Corollary 2 *The following problem can be solved in polynomial time.*
Given a well-structured WF-net, to decide if it is sound.

Proof.
Let PN be a well-structured WF-net. The extended net $\overline{PN}$ is elementary extended non-self controlling (Theorem 2) and structurally bounded (see proof of Lemma 5). For bounded elementary extended non-self controlling nets the problem of deciding whether a given marking is live, can be solved in polynomial time (See [4]). Therefore, the problem of deciding whether $(\overline{PN}, i)$ is live and bounded can be solved in polynomial time. By Theorem 1, this corresponds to soundness. □

Lemma 6 *A sound well-structured WF-net is safe.*

Proof.
Let $\overline{PN}$ be the net PN extended with a transition connecting o and i. $\overline{PN}$ is extended non-self controlling. $\overline{PN}$ is covered by state-machines (S-components), see Corollary 5.3 in [4]. Hence, each place is part of such a state-machine component. Clearly, i and o are nodes of any state-machine component. Hence, for each place p there is a semi-positive invariant with weights 0 or 1 which assigns a positive weight to p, i and o. Hence, $\overline{PN}$ is safe and so is PN. □

Well-structured WF-nets and free-choice WF-nets have similar properties. In both cases soundness can be verified very efficiently and soundness implies safeness. In spite of these similarities, there are sound well-structured WF-nets which are not free-choice (Figure 8) and there are sound free-choice WF-nets which are not well-structured. In fact, it is possible to have a sound WF-net which is neither free-choice nor well-structured (Figures 2 and 5).
Notwithstanding these observations, the two structural characterizations turn out to be very useful for the analysis of workflow process definitions. Both well-structuredness and the free-choice property correspond to desirable properties of a workflow. A WF-net satisfying one of these properties can be analyzed very efficiently. Moreover, most of today's WFMS's only allow for the enactment of workflows satisfying both properties.

What about the sound WF-nets shown in Figure 2 and Figure 5? The WF-net shown in Figure 5 can be transformed into a free-choice well-structured WF-net by separating choice and parallelism. The WF-net shown in Figure 2 cannot be transformed into a free-choice or well-structured WF-net without yielding a much more complex WF-net. Place *c5* acts as some kind of milestone which is tested by the task *process_complaint*. Traditional WFMS's which do not make the state of the case explicit, will not be able to handle the workflow specified by Figure 2. Only workflow management systems such as COSA ([18]) have the capability to enact such a state-based workflow. Even if one is able to use state-based workflows allowing for constructs which violate well-structuredness and the free-choice property, then the structural characterizations are still useful. If a WF-net is not free-choice or not well-structured, one should locate the source which violates one of these properties and check whether it is really necessary to use a non-free-choice or a non-well-structured construct. If the non-free-choice or non-well-structured construct is really necessary, then the correctness of the construct should be double-checked because it is a potential source of error.

7 Composition of WF-nets

The WF-nets in this paper are very simple compared to the workflows encountered in practise. For example, in the Dutch Customs Department there are workflows consisting of more than 80 tasks with a very complex interaction structure (cf. [3]). For the designer of such a workflow the complexity is overwhelming and communication with end-users using one huge diagram is difficult. In most cases hierarchical (de)composition is used to tackle

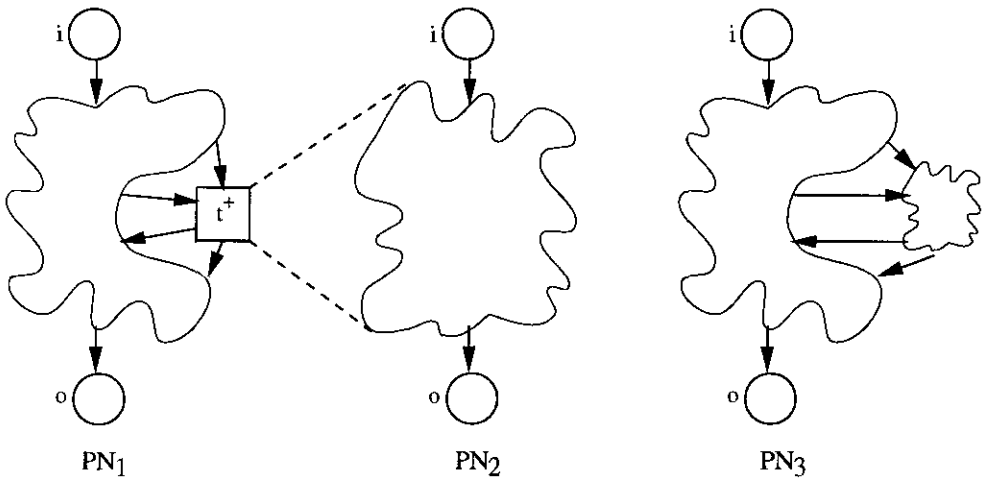

Figure 9: Task refinement: WF-net PN_3 is composed out of PN_1 and PN_2.

this problem. A complex workflow is decomposed into subflows and each of the subflows is decomposed into smaller subflows until the desired level of detail is reached. Many WFMS's allow for such a hierarchical decomposition. In addition, this mechanism can be utilized for the reuse of existing workflows. Consider for example multiple workflows sharing a generic subflow. Some WFMS-vendors also supply reference models which correspond to typical workflow processes in insurance, banking, finance, marketing, purchase, procurement, logistics and manufacturing.

Reference models, reuse and the structuring of complex workflows require a hierarchy concept. The most common hierarchy concept supported by many WFMS's is *task refinement*, i.e., a task can be refined by a subflow. This concept is illustrated in Figure 9. The WF-net PN_1 contains a task t^+ which is refined by another WF-net PN_2, i.e., t^+ is no longer a task but a reference to a subflow. A WF-net which represents a subflow should satisfy the same requirements as an ordinary WF-net (see Definition 7). The semantics of the hierarchy concept are straightforward; simply replace the refined transition by the corresponding subnet. Figure 9 shows that the refinement of t^+ in PN_1 by PN_2 yields a WF-net PN_3.

The hierarchy concept can be exploited to establish the correctness of a workflow. Given a complex hierarchical workflow model, it is possible to verify soundness by analyzing each of the subflows separately. The following theorem shows that the soundness property defined in this paper allows for modular analysis.

Theorem 3 (Compositionality) *Let* $PN_1 = (P_1, T_1, F_1)$ *and* $PN_2 = (P_2, T_2, F_2)$ *be two WF-nets such that* $T_1 \cap T_2 = \emptyset$, $P_1 \cap P_2 = \{i, o\}$ *and* $t^+ \in T_1$. $PN_3 = (P_3, T_3, F_3)$ *is the WF-net obtained by replacing transition* t^+ *in* PN_1 *by* PN_2, *i.e.,* $P_3 = P_1 \cup P_2$, $T_3 = (T_1 \setminus \{t^+\}) \cup T_2$ *and*

$$\begin{aligned} F_3 = {} & \{(x, y) \in F_1 \mid x \neq t^+ \wedge y \neq t^+\} \cup \{(x, y) \in F_2 \mid \{x, y\} \cap \{i, o\} = \emptyset\} \cup \\ & \{(x, y) \in P_1 \times T_2 \mid (x, t^+) \in F_1 \wedge (i, y) \in F_2\} \cup \\ & \{(x, y) \in T_2 \times P_1 \mid (t^+, y) \in F_1 \wedge (x, o) \in F_2\}. \end{aligned}$$

For PN_1, PN_2 *and* PN_3 *the following statements hold:*

1. *If* PN_3 *is free-choice, then* PN_1 *and* PN_2 *are free-choice.*
2. *If* PN_3 *is well-structured, then* PN_1 *and* PN_2 *are well-structured.*
3. *If* (PN_1, i) *is safe and* PN_1 *and* PN_2 *are sound, then* PN_3 *is sound.*
4. (PN_1, i) *and* (PN_2, i) *are safe and sound iff* (PN_3, i) *is safe and sound.*
5. PN_1 *and* PN_2 *are free-choice and sound iff* PN_3 *is free-choice and sound.*
6. *If* PN_3 *is well-structured and sound, then* PN_1 *and* PN_2 *are well-structured and sound.*
7. *If* $\bullet t^+$ *and* $t^+\bullet$ *are both singletons, then* PN_1 *and* PN_2 *are well-structured and sound iff* PN_3 *is well-structured and sound.*

Proof.

1. The only transitions that may violate the free-choice property are t^+ (PN_1) and $\{t \in T_2 \mid (i, t) \in F_2\}$ (PN_2). Transition t^+ has the same input set as any of the transitions $\{t \in T_2 \mid (i, t) \in F_2\}$ in PN_3 if we only consider the places in $P_3 \cap P_1$. Hence, t^+ does not violate the free-choice property in PN_1. All transitions t in PN_2 such that $(i, t) \in F_2$ respect the free-choice property; the input places in $P_3 \setminus P_2$ are replaced by i.
2. $\overline{PN_1}$ $(\overline{PN_2})$ is well-handled because any elementary path in $\overline{PN_1}$ $(\overline{PN_2})$ corresponds to a path in $\overline{PN_3}$.
3. Let (PN_1, i) be safe and let PN_1 and PN_2 be sound. We need to prove that $(\overline{PN_3}, i)$ is live and bounded. The subnet in $\overline{PN_3}$ which corresponds to t^+ behaves like a transition which may postpone the production of tokens for $t^+\bullet$. It is essential that the input places of t^+ in $(\overline{PN_3}, i)$ are safe. This way it is guaranteed that the states of the subnet correspond to the states of $(\overline{PN_2}, i)$. Hence, the transitions in $T_3 \cap T_2$ are live (t^+ is live) and the places in $P_3 \setminus P_1$ are bounded. Since the subnet behaves like t^+, the transitions in $T_3 \cap (T_1 \setminus \{t^+\})$ are live and the places in $P_3 \cap P_1$ are bounded. Hence, PN_3 is sound.

4. Let (PN_1, i) and (PN_2, i) be safe and sound. Clearly, PN_3 is sound (see proof of 3.). (PN_3, i) is also safe because every reachable state corresponds to a combination of a safe state of (PN_1, i) and a safe state of (PN_2, i).
Let (PN_3, i) be safe and sound. Consider the subnet in PN_3 which corresponds to t^+. X is the set of transitions in $T_3 \cap T_2$ consuming from $P_3 \cap {}^\bullet t^+$ and Y is the set of transitions in $T_3 \cap T_2$ producing tokens for $P_3 \cap t^+{}^\bullet$. If a transition in X fires, then it should be possible to fire a transition in Y because of the liveness of the original net. If a transition in Y fires, the subnet should become empty. If the subnet is not empty after firing a transition in Y, then there are two possibilities: (1) it is possible to move the subnet to a state such that a transition in Y can fire (without firing transitions in $T_3 \cap T_1$) or (2) it is not possible to move to such a state. In the first case, the places $t^+{}^\bullet$ in PN_3 are not safe. In the second case, a token is trapped in the subnet or the subnet is not safe the moment a transition in X fires. (PN_2, i) corresponds to the subnet bordered by X and Y and is, as we have just shown, sound and safe. Remains to prove that (PN_1, i) is safe and sound. Since the subnet which corresponds to t^+ behaves like a transition which may postpone the production of tokens, we can replace the subnet by t^+ without changing dynamic properties such as safeness and soundness.

5. Let PN_1 and PN_2 be free-choice and sound. Since $(\overline{PN_1}, i)$ is safe (see Lemma 4), PN_3 is sound (see proof of 3.). Remains to prove that PN_3 is free-choice. The only transitions in PN_3 which may violate the free-choice property are the transitions in $T_3 \cap T_2$ consuming tokens from $P_3 \cap {}^\bullet t^+$. Because PN_2 is sound, these transitions need to have an input set identical to t^+ in PN_1 (if this is not the case at least on of the transitions is dead). Since PN_1 is free-choice, PN_3 is also free-choice.
Let PN_3 be free-choice and sound. PN_1 and PN_2 are also free-choice (see proof of 1.). Since (PN_3, i) is safe (see Lemma 4), PN_1 and PN_2 are sound (see proof of 4.).

6. Let PN_3 be well-structured and sound. PN_1 and PN_2 are also well-structured (see proof of 2.). Since (PN_3, i) is safe (see Lemma 6), PN_1 and PN_2 are sound (see proof of 4.).

7. Remains to prove that if PN_1 and PN_2 are well-structured, then PN_3 is also well-structured. Suppose that PN_3 is not well-structured. There are two disjunct elementary paths leading from x to y in $\overline{PN_3}$. Since PN_1 is well-structured, at least one of these paths is enabled via the refinement of t^+. However, because t^+ has precisely one input and one output place and PN_2 is also well-structured, this is not possible.

□

Figure 10 shows a hierarchical WF-net. Both of the subflows (*handle_questionnaire* and *processing*) and the main flow are safe and sound. Therefore, the overall workflow represented by the hierarchical WF-net is also safe and sound. Moreover, the free-choice property and well-structuredness are also preserved by the hierarchical composition. Theorem 3 is of particular importance for the reuse subflows. For the analysis of a complex

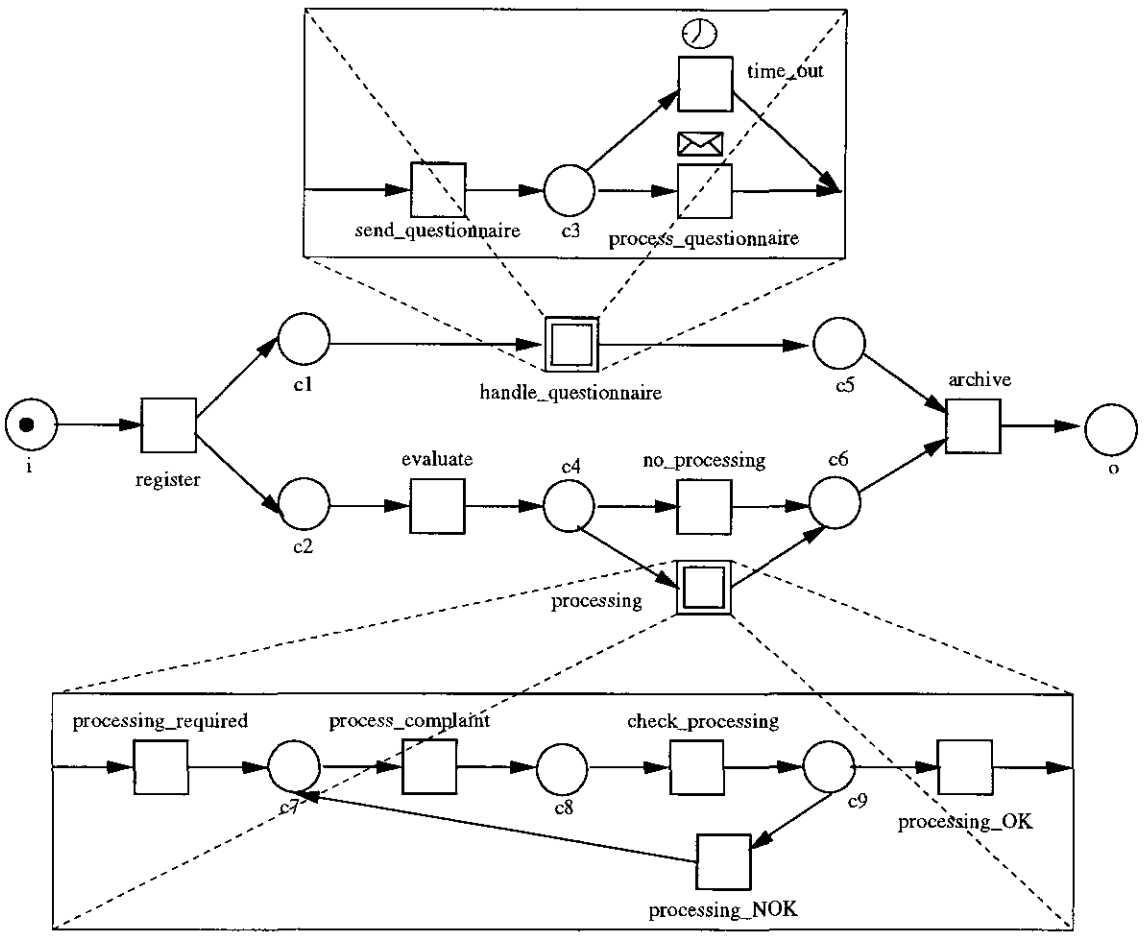

Figure 10: A hierarchical WF-net for the processing of complaints.

workflow, every safe and sound subflow can be considered to be a single task. This allows for an efficient modular analysis of the soundness property. Moreover, the statements embedded in Theorem 3 can help a workflow designer to construct correct workflow process definitions.

8 Conclusion

In this paper we have investigated a basic property that any workflow process definition should satisfy: the soundness property. For WF-nets this property coincides with liveness and boundedness. In our quest for a structural characterization of WF-nets satisfying the soundness property, we have identified two important subclasses: free-choice WF-nets and well-structured WF-nets. These subclasses have desirable properties and allow for efficient analysis methods. Moreover, most WFMS's only support workflows characterized by these two subclasses. Figure 11 illustrates the relationships between soundness and the two subclasses.

If a workflow process is specified by a hierarchical WF-net, then the modular analysis of the soundness property is often possible. A workflow composed out of correct subflows can be verified without incorporating the specification of each subflow.

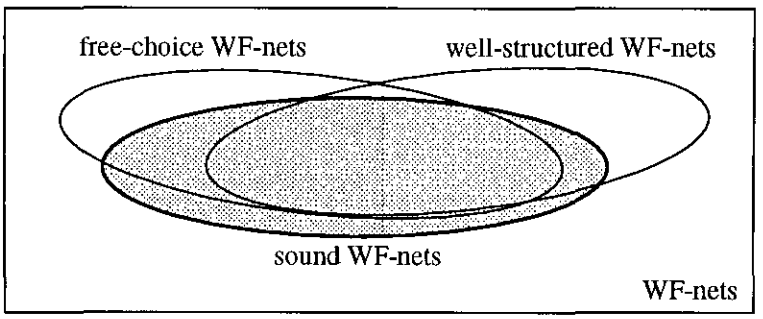

Figure 11: The relationships between soundness, well-formedness and the free-choice property.

The results presented in this paper give workflow designers a handle to construct correct workflows. Although it is possible to use standard Petri-net-based analysis tools, we are developing a workflow-analyzer which can be used by people not familiar with Petri-net theory. This workflow-analyzer will interface with existing workflow products such as COSA, Protos, Income and StructWare.

Acknowledgements

The author would like to thank Dr. M. Voorhoeve, Ir. T. Basten and Dr.Ir. G.J. Houben for their valuable suggestions.

References

[1] W.M.P. van der Aalst. A class of Petri net for modeling and analyzing business processes. Computing Science Reports 95/26, Eindhoven University of Technology, Eindhoven, 1995.

[2] W.M.P. van der Aalst. Petri-net-based Workflow Management Software. In A. Sheth, editor, *Proceedings of the NFS Workshop on Workflow and Process Automation in Information Systems*, pages 114–118, Athens, Georgia, May 1996.

[3] W.M.P. van der Aalst. Three Good reasons for Using a Petri-net-based Workflow Management System. In S. Navathe and T. Wakayama, editors, *Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC'96)*, pages 179–201, Camebridge, Massachusetts, Nov 1996.

[4] K. Barkaoui, J.M. Couvreur, and C. Hutheillet. On liveness in Extended Non Self-Controlling Nets. In G. De Michelis and M. Diaz, editors, *Application and Theory of Petri Nets 1995*, volume 935 of *Lecture Notes in Computer Science*, pages 25–44. Springer-Verlag, Berlin, 1995.

[5] E. Best. Structure theory of Petri nets: the free choice hiatus. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Advances in Petri Nets 1986 Part I: Petri Nets, central*

models and their properties, volume 254 of *Lecture Notes in Computer Science*, pages 168–206. Springer-Verlag, Berlin, 1987.

[6] A. Cheng, J. Esparza, and J. Palsberg. Complexity results for 1-safe nets. In R.K. Shyamasundar, editor, *Foundations of software technology and theoretical computer science*, volume 761 of *Lecture Notes in Computer Science*, pages 326–337. Springer-Verlag, Berlin, 1993.

[7] J. Desel. A proof of the Rank theorem for extended free-choice nets. In K. Jensen, editor, *Application and Theory of Petri Nets 1992*, volume 616 of *Lecture Notes in Computer Science*, pages 134–153. Springer-Verlag, Berlin, 1992.

[8] J. Desel and J. Esparza. *Free choice Petri nets*, volume 40 of *Cambridge tracts in theoretical computer science*. Cambridge University Press, Cambridge, 1995.

[9] C.A. Ellis and G.J. Nutt. Modelling and Enactment of Workflow Systems. In M. Ajmone Marsan, editor, *Application and Theory of Petri Nets 1993*, volume 691 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, Berlin, 1993.

[10] J. Esparza. Synthesis rules for Petri nets, and how they can lead to new results. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings of CONCUR 1990*, volume 458 of *Lecture Notes in Computer Science*, pages 182–198. Springer-Verlag, Berlin, 1990.

[11] J. Esparza and M. Silva. Circuits, Handles, Bridges and Nets. In G. Rozenberg, editor, *Advances in Petri Nets 1990*, volume 483 of *Lecture Notes in Computer Science*, pages 210–242. Springer-Verlag, Berlin, 1990.

[12] K. Gostellow, V. Cerf, G. Estrin, and S. Volansky. Proper Termination of Flow-of-control in Programs Involving Concurrent Processes. *ACM Sigplan*, 7(11):15–27, 1972.

[13] M.H.T. Hack. Analysis production schemata by Petri nets. Master's thesis, Massachusetts Institute of Technology, Cambridge, Mass., 1972.

[14] G. De Michelis, C. Ellis, and G. Memmi, editors. *Proceedings of the second Workshop on Computer-Supported Cooperative Work, Petri nets and related formalisms*, Zaragoza, Spain, June 1994.

[15] T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[16] J.L. Peterson. *Petri net theory and the modeling of systems*. Prentice-Hall, Englewood Cliffs, 1981.

[17] C.A. Petri. *Kommunikation mit Automaten*. PhD thesis, Institut für instrumentelle Mathematik, Bonn, 1962.

[18] Software-Ley. *COSA User Manual*. Software-Ley GmbH, Pullheim, 1996.

[19] P.A. Straub and C. Hurtado. The Simple Control Property of Business Process Models. In *XV International Conference of the Chilean Computer Science Society*, 1995.

[20] WFMC. Workflow Management Coalition Terminology and Glossary (WFMC-TC-1011). Technical report, Workflow Management Coalition, Brussels, 1996.

[21] M. Wolf and U. Reimer, editors. *Proceedings of the International Conference on Practical Aspects of Knowledge Management (PAKM'96), Workshop on Adaptive Workflow*, Basel, Switzerland, Oct 1996.

Computing Science Reports

Department of Mathematics and Computing Science
Eindhoven University of Technology

In this series appeared:

93/01	R. van Geldrop	Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36.
93/02	T. Verhoeff	A continuous version of the Prisoner's Dilemma, p. 17
93/03	T. Verhoeff	Quicksort for linked lists, p. 8.
93/04	E.H.L. Aarts J.H.M. Korst P.J. Zwietering	Deterministic and randomized local search, p. 78.
93/05	J.C.M. Baeten C. Verhoef	A congruence theorem for structured operational semantics with predicates, p. 18.
93/06	J.P. Veltkamp	On the unavoidability of metastable behaviour, p. 29
93/07	P.D. Moerland	Exercises in Multiprogramming, p. 97
93/08	J. Verhoosel	A Formal Deterministic Scheduling Model for Hard Real-Time Executions in DEDOS, p. 32.
93/09	K.M. van Hee	Systems Engineering: a Formal Approach Part I: System Concepts, p. 72.
93/10	K.M. van Hee	Systems Engineering: a Formal Approach Part II: Frameworks, p. 44.
93/11	K.M. van Hee	Systems Engineering: a Formal Approach Part III: Modeling Methods, p. 101.
93/12	K.M. van Hee	Systems Engineering: a Formal Approach Part IV: Analysis Methods, p. 63.
93/13	K.M. van Hee	Systems Engineering: a Formal Approach Part V: Specification Language, p. 89.
93/14	J.C.M. Baeten J.A. Bergstra	On Sequential Composition, Action Prefixes and Process Prefix, p. 21.
93/15	J.C.M. Baeten J.A. Bergstra R.N. Bol	A Real-Time Process Logic, p. 31.
93/16	H. Schepers J. Hooman	A Trace-Based Compositional Proof Theory for Fault Tolerant Distributed Systems, p. 27
93/17	D. Alstein P. van der Stok	Hard Real-Time Reliable Multicast in the DEDOS system, p. 19.
93/18	C. Verhoef	A congruence theorem for structured operational semantics with predicates and negative premises, p. 22.
93/19	G-J. Houben	The Design of an Online Help Facility for ExSpect, p.21.
93/20	F.S. de Boer	A Process Algebra of Concurrent Constraint Programming, p. 15.
93/21	M. Codish D. Dams G. Filé M. Bruynooghe	Freeness Analysis for Logic Programs - And Correctness, p. 24
93/22	E. Poll	A Typechecker for Bijective Pure Type Systems, p. 28.
93/23	E. de Kogel	Relational Algebra and Equational Proofs, p. 23.
93/24	E. Poll and Paula Severi	Pure Type Systems with Definitions, p. 38.
93/25	H. Schepers and R. Gerth	A Compositional Proof Theory for Fault Tolerant Real-Time Distributed Systems, p. 31.
93/26	W.M.P. van der Aalst	Multi-dimensional Petri nets, p. 25.
93/27	T. Kloks and D. Kratsch	Finding all minimal separators of a graph, p. 11.
93/28	F. Kamareddine and R. Nederpelt	A Semantics for a fine λ-calculus with de Bruijn indices, p. 49.
93/29	R. Post and P. De Bra	GOLD, a Graph Oriented Language for Databases, p. 42.
93/30	J. Deogun T. Kloks D. Kratsch H. Müller	On Vertex Ranking for Permutation and Other Graphs, p. 11.

93/31	W. Körver	Derivation of delay insensitive and speed independent CMOS circuits, using directed commands and production rule sets, p. 40.
93/32	H. ten Eikelder and H. van Geldrop	On the Correctness of some Algorithms to generate Finite Automata for Regular Expressions, p. 17.
93/33	L. Loyens and J. Moonen	ILIAS, a sequential language for parallel matrix computations, p. 20.
93/34	J.C.M. Baeten and J.A. Bergstra	Real Time Process Algebra with Infinitesimals, p.39.
93/35	W. Ferrer and P. Severi	Abstract Reduction and Topology, p. 28.
93/36	J.C.M. Baeten and J.A. Bergstra	Non Interleaving Process Algebra, p. 17.
93/37	J. Brunekreef J-P. Katoen R. Koymans S. Mauw	Design and Analysis of Dynamic Leader Election Protocols in Broadcast Networks, p. 73.
93/38	C. Verhoef	A general conservative extension theorem in process algebra, p. 17.
93/39	W.P.M. Nuijten E.H.L. Aarts D.A.A. van Erp Taalman Kip K.M. van Hee	Job Shop Scheduling by Constraint Satisfaction, p. 22.
93/40	P.D.V. van der Stok M.M.M.P.J. Claessen D. Alstein	A Hierarchical Membership Protocol for Synchronous Distributed Systems, p. 43.
93/41	A. Bijlsma	Temporal operators viewed as predicate transformers, p. 11.
93/42	P.M.P. Rambags	Automatic Verification of Regular Protocols in P/T Nets, p. 23.
93/43	B.W. Watson	A taxomomy of finite automata construction algorithms, p. 87.
93/44	B.W. Watson	A taxonomy of finite automata minimization algorithms, p. 23.
93/45	E.J. Luit J.M.M. Martin	A precise clock synchronization protocol,p.
93/46	T. Kloks D. Kratsch J. Spinrad	Treewidth and Patwidth of Cocomparability graphs of Bounded Dimension, p. 14.
93/47	W. v.d. Aalst P. De Bra G.J. Houben Y. Kornatzky	Browsing Semantics in the "Tower" Model, p. 19.
93/48	R. Gerth	Verifying Sequentially Consistent Memory using Interface Refinement, p. 20.
94/01	P. America M. van der Kammen R.P. Nederpelt O.S. van Roosmalen H.C.M. de Swart	The object-oriented paradigm, p. 28.
94/02	F. Kamareddine R.P. Nederpelt	Canonical typing and II-conversion, p. 51.
94/03	L.B. Hartman K.M. van Hee	Application of Marcov Decision Processe to Search Problems, p. 21.
94/04	J.C.M. Baeten J.A. Bergstra	Graph Isomorphism Models for Non Interleaving Process Algebra, p. 18.
94/05	P. Zhou J. Hooman	Formal Specification and Compositional Verification of an Atomic Broadcast Protocol, p. 22.
94/06	T. Basten T. Kunz J. Black M. Coffin D. Taylor	Time and the Order of Abstract Events in Distributed Computations, p. 29.
94/07	K.R. Apt R. Bol	Logic Programming and Negation: A Survey, p. 62.
94/08	O.S. van Roosmalen	A Hierarchical Diagrammatic Representation of Class Structure, p. 22.
94/09	J.C.M. Baeten J.A. Bergstra	Process Algebra with Partial Choice, p. 16.

94/10	T. verhoeff	The testing Paradigm Applied to Network Structure. p. 31.
94/11	J. Peleska C. Huizing C. Petersohn	A Comparison of Ward & Mellor's Transformation Schema with State- & Activitycharts, p. 30.
94/12	T. Kloks D. Kratsch H. Müller	Dominoes, p. 14.
94/13	R. Seljée	A New Method for Integrity Constraint checking in Deductive Databases, p. 34.
94/14	W. Peremans	Ups and Downs of Type Theory, p. 9.
94/15	R.J.M. Vaessens E.H.L. Aarts J.K. Lenstra	Job Shop Scheduling by Local Search, p. 21.
94/16	R.C. Backhouse H. Doornbos	Mathematical Induction Made Calculational, p. 36.
94/17	S. Mauw M.A. Reniers	An Algebraic Semantics of Basic Message Sequence Charts, p. 9.
94/18	F. Kamareddine R. Nederpelt	Refining Reduction in the Lambda Calculus, p. 15.
94/19	B.W. Watson	The performance of single-keyword and multiple-keyword pattern matching algorithms, p. 46.
94/20	R. Bloo F. Kamareddine R. Nederpelt	Beyond β-Reduction in Church's $\lambda\rightarrow$, p. 22.
94/21	B.W. Watson	An introduction to the Fire engine: A C++ toolkit for Finite automata and Regular Expressions.
94/22	B.W. Watson	The design and implementation of the FIRE engine: A C++ toolkit for Finite automata and regular Expressions.
94/23	S. Mauw and M.A. Reniers	An algebraic semantics of Message Sequence Charts, p. 43.
94/24	D. Dams O. Grumberg R. Gerth	Abstract Interpretation of Reactive Systems: Abstractions Preserving $\forall$CTL*, $\exists$CTL* and CTL*, p. 28.
94/25	T. Kloks	$K_{1,3}$-free and W_4-free graphs, p. 10.
94/26	R.R. Hoogerwoord	On the foundations of functional programming: a programmer's point of view, p. 54.
94/27	S. Mauw and H. Mulder	Regularity of BPA-Systems is Decidable, p. 14.
94/28	C.W.A.M. van Overveld M. Verhoeven	Stars or Stripes: a comparative study of finite and transfinite techniques for surface modelling, p. 20.
94/29	J. Hooman	Correctness of Real Time Systems by Construction, p. 22.
94/30	J.C.M. Baeten J.A. Bergstra Gh. Ştefanescu	Process Algebra with Feedback, p. 22.
94/31	B.W. Watson R.E. Watson	A Boyer-Moore type algorithm for regular expression pattern matching, p. 22.
94/32	J.J. Vereijken	Fischer's Protocol in Timed Process Algebra, p. 38.
94/33	T. Laan	A formalization of the Ramified Type Theory, p.40.
94/34	R. Bloo F. Kamareddine R. Nederpelt	The Barendregt Cube with Definitions and Generalised Reduction, p. 37.
94/35	J.C.M. Baeten S. Mauw	Delayed choice: an operator for joining Message Sequence Charts, p. 15.
94/36	F. Kamareddine R. Nederpelt	Canonical typing and Π-conversion in the Barendregt Cube, p. 19.
94/37	T. Basten R. Bol M. Voorhoeve	Simulating and Analyzing Railway Interlockings in ExSpect, p. 30.
94/38	A. Bijlsma C.S. Scholten	Point-free substitution, p. 10.
94/39	A. Blokhuis T. Kloks	On the equivalence covering number of splitgraphs, p. 4.

94/40	D. Alstein	Distributed Consensus and Hard Real-Time Systems, p. 34.
94/41	T. Kloks D. Kratsch	Computing a perfect edge without vertex elimination ordering of a chordal bipartite graph, p. 6.
94/42	J. Engelfriet J.J. Vereijken	Concatenation of Graphs, p. 7.
94/43	R.C. Backhouse M. Bijsterveld	Category Theory as Coherently Constructive Lattice Theory: An Illustration, p. 35.
94/44	E. Brinksma J. Davies R. Gerth S. Graf W. Janssen B. Jonsson S. Katz G. Lowe M. Poel A. Pnueli C. Rump J. Zwiers	Verifying Sequentially Consistent Memory, p. 160
94/45	G.J. Houben	Tutorial voor de ExSpect-bibliotheek voor "Administratieve Logistiek", p. 43.
94/46	R. Bloo F. Kamareddine R. Nederpelt	The λ-cube with classes of terms modulo conversion, p. 16.
94/47	R. Bloo F. Kamareddine R. Nederpelt	On Π-conversion in Type Theory, p. 12.
94/48	Mathematics of Program Construction Group	Fixed-Point Calculus, p. 11.
94/49	J.C.M. Baeten J.A. Bergstra	Process Algebra with Propositional Signals, p. 25.
94/50	H. Geuvers	A short and flexible proof of Strong Normalazation for the Calculus of Constructions, p. 27.
94/51	T. Kloks D. Kratsch H. Müller	Listing simplicial vertices and recognizing diamond-free graphs, p. 4.
94/52	W. Penczek R. Kuiper	Traces and Logic, p. 81
94/53	R. Gerth R. Kuiper D. Peled W. Penczek	A Partial Order Approach to Branching Time Logic Model Checking, p. 20.
95/01	J.J. Lukkien	The Construction of a small CommunicationLibrary, p.16.
95/02	M. Bezem R. Bol J.F. Groote	Formalizing Process Algebraic Verifications in the Calculus of Constructions, p.49.
95/03	J.C.M. Baeten C. Verhoef	Concrete process algebra, p. 134.
95/04	I. Hidders	An Isotopic Invariant for Planar Drawings of Connected Planar Graphs, p. 9.
95/05	P. Severi	A Type Inference Algorithm for Pure Type Systems, p.20.
95/06	T.W.M. Vossen M.G.A. Verhoeven H.M.M. ten Eikelder E.H.L. Aarts	A Quantitative Analysis of Iterated Local Search, p.23.
95/07	G.A.M. de Bruyn O.S. van Roosmalen	Drawing Execution Graphs by Parsing, p. 10.
95/08	R. Bloo	Preservation of Strong Normalisation for Explicit Substitution, p. 12.
95/09	J.C.M. Baeten J.A. Bergstra	Discrete Time Process Algebra, p. 20
95/10	R.C. Backhouse R. Verhoeven O. Weber	Math$\int$pad: A System for On-Line Prepararation of Mathematical Documents, p. 15
95/11	R. Seljée	Deductive Database Systems and integrity constraint checking, p. 36.
95/12	S. Mauw and M. Reniers	Empty Interworkings and Refinement

		Semantics of Interworkings Revised, p. 19.
95/13	B.W. Watson and G. Zwaan	A taxonomy of sublinear multiple keyword pattern matching algorithms, p. 26.
95/14	A. Ponse, C. Verhoef, S.F.M. Vlijmen (eds.)	De proceedings: ACP'95, p.
95/15	P. Niebert and W. Penczek	On the Connection of Partial Order Logics and Partial Order Reduction Methods, p. 12.
95/16	D. Dams, O. Grumberg, R. Gerth	Abstract Interpretation of Reactive Systems: Preservation of CTL*, p. 27.
95/17	S. Mauw and E.A. van der Meulen	Specification of tools for Message Sequence Charts, p. 36.
95/18	F. Kamareddine and T. Laan	A Reflection on Russell's Ramified Types and Kripke's Hierarchy of Truths, p. 14.
95/19	J.C.M. Baeten and J.A. Bergstra	Discrete Time Process Algebra with Abstraction, p. 15.
95/20	F. van Raamsdonk and P. Severi	On Normalisation, p. 33.
95/21	A. van Deursen	Axiomatizing Early and Late Input by Variable Elimination, p. 44.
95/22	B. Arnold, A. v. Deursen, M. Res	An Algebraic Specification of a Language for Describing Financial Products, p. 11.
95/23	W.M.P. van der Aalst	Petri net based scheduling, p. 20.
95/24	F.P.M. Dignum, W.P.M. Nuijten, L.M.A. Janssen	Solving a Time Tabling Problem by Constraint Satisfaction, p. 14.
95/25	L. Feijs	Synchronous Sequence Charts In Action, p. 36.
95/26	W.M.P. van der Aalst	A Class of Petri nets for modeling and analyzing business processes, p. 24.
95/27	P.D.V. van der Stok, J. van der Wal	Proceedings of the Real-Time Database Workshop, p. 106.
95/28	W. Fokkink, C. Verhoef	A Conservative Look at term Deduction Systems with Variable Binding, p. 29.
95/29	H. Jurjus	On Nesting of a Nonmonotonic Conditional, p. 14
95/30	J. Hidders, C. Hoskens, J. Paredaens	The Formal Model of a Pattern Browsing Technique, p.24.
95/31	P. Kelb, D. Dams and R. Gerth	Practical Symbolic Model Checking of the full μ-calculus using Compositional Abstractions, p. 17.
95/32	W.M.P. van der Aalst	Handboek simulatie, p. 51.
95/33	J. Engelfriet and JJ. Vereijken	Context-Free Graph Grammars and Concatenation of Graphs, p. 35.
95/34	J. Zwanenburg	Record concatenation with intersection types, p. 46.
95/35	T. Basten and M. Voorhoeve	An algebraic semantics for hierarchical P/T Nets, p. 32.
96/01	M. Voorhoeve and T. Basten	Process Algebra with Autonomous Actions, p. 12.
96/02	P. de Bra and A. Aerts	Multi-User Publishing in the Web: DreSS, A Document Repository Service Station, p. 12
96/03	W.M.P. van der Aalst	Parallel Computation of Reachable Dead States in a Free-choice Petri Net, p. 26.
96/04	S. Mauw	Example specifications in phi-SDL.
96/05	T. Basten and W.M.P. v.d. Aalst	A Process-Algebraic Approach to Life-Cycle Inheritance Inheritance = Encapsulation + Abstraction, p. 15.
96/06	W.M.P. van der Aalst and T. Basten	Life-Cycle Inheritance A Petri-Net-Based Approach, p. 18.
96/07	M. Voorhoeve	Structural Petri Net Equivalence, p. 16.
96/08	A.T.M. Aerts, P.M.E. De Bra, J.T. de Munk	OODB Support for WWW Applications: Disclosing the internal structure of Hyperdocuments, p. 14.
96/09	F. Dignum, H. Weigand, E. Verharen	A Formal Specification of Deadlines using Dynamic Deontic Logic, p. 18.
96/10	R. Bloo, H. Geuvers	Explicit Substitution: on the Edge of Strong Normalisation, p. 13.
96/11	T. Laan	AUTOMATH and Pure Type Systems, p. 30.
96/12	F. Kamareddine and T. Laan	A Correspondence between Nuprl and the Ramified Theory of Types, p. 12.
96/13	T. Borghuis	Priorean Tense Logics in Modal Pure Type Systems, p. 61
96/14	S.H.J. Bos and M.A. Reniers	The I^2C-bus in Discrete-Time Process Algebra, p. 25.
96/15	M.A. Reniers and J.J. Vereijken	Completeness in Discrete-Time Process Algebra, p. 139.
96/16	P. Hoogendijk and O. de Moor	What is a data type?, p. 29.

96/17	E. Boiten and P. Hoogendijk	Nested collections and polytypism, p. 11.
96/18	P.D.V. van der Stok	Real-Time Distributed Concurrency Control Algorithms with mixed time constraints, p. 71.
96/19	M.A. Reniers	Static Semantics of Message Sequence Charts, p. 71
96/20	L. Feijs	Algebraic Specification and Simulation of Lazy Functional Programs in a concurrent Environment, p. 27.
96/21	L. Bijlsma and R. Nederpelt	Predicate calculus: concepts and misconceptions, p. 26.
96/22	M.C.A. van de Graaf and G.J. Houben	Designing Effective Workflow Management Processes, p. 22.