A Backward/Forward Strategy for Verifying Safety Properties of Infinite–State Systems

Michael Baldamus, Richard Mayr and Gerardo Schneider

Department of Information Technology
Uppsala University
Box 337, SE-751 05 Uppsala, Sweden

Technical report 2003-065
December 2003
ISSN 1404-3203

A Backward/Forward Strategy for Verifying Safety Properties of Infinite–State Systems

Michael Baldamus[1,2], Richard Mayr[3,4], Gerardo Schneider[5,6]

[1,5]Uppsala University, Department of Computer Systems, Box 337, 751 05 Uppsala, Sweden
E–Mail: {Michael.Baldamus,Gerardo.Schneider}@it.uu.se
[3]University of Freiburg, Department of Computer Science, Georges–Köhler–Allee 51, 79110 Freiburg, Germany
E–Mail: mayrri@informatik.uni-freiburg.de

Abstract. This paper has two main contributions: For one, we describe a general method for verifying safety properties of non–well–quasi–ordered infinite–state systems for which reachability is undecidable in general, the question being whether a set U of configurations is reachable. In many cases this problem can be solved as follows: First, one constructs a well–quasi–ordered overapproximation of the system in question. Thereby one can compute an overapproximation of the set $Pre^*(U)$ of all predecessors of U. Second, one performs an exact bounded forward search for U (starting at the initial state) which always stays inside the already computed overapproximation of $Pre^*(U)$, thus curbing the search space. This restricted forward search is more efficient than a normal forward search, yielding answers of the form YES, NO, or UNKNOWN, where the YES and NO answers are always correct.
As our second main contribution herein, we apply our method to relabelling–free CCS with finite sums, which is already a process calculus for which reachability is undecidable. To our knowledge, this part is actually the first application of well–structered systems to verifying safety properties in process calculi. The application is done via a special Petri nets semantics for the calculus that we consider.

1 Introduction

Automatic techniques for reachability analysis have been developed for well–structured infinite–state systems for which there exists a well–quasi–order on the set of configurations (e.g., [2, 9]). The main idea is that for these systems one can effectively construct a representation of $Pre^*(U)$, the set of all predecessors of any given upward–closed set U of configurations.

Our aim is to make this technique applicable also to non–well–structured systems, such as CCS–style process calculi, for which reachability is undecidable in general. Our idea is to first construct a well–quasi–ordered overapproximation of the given system in the form of a (generalised) Petri net. Then one can compute an overapproximation of $Pre^*(U)$, given an upward–closed set U of configurations. If the initial configuration is not in this overapproximation of $Pre^*(U)$, then the answer to the reachability problem is surely NO. Otherwise, one can, in a second step, do an exact forward search (possibly using acceleration methods) which always stays inside the already computed overapproximation of $Pre^*(U)$, thus curbing the search–space. Due to the restricted search space, this forward search is more efficient than a normal forward search. It still does not terminate in general, however, and thus one restricts the depth of the forward search. It follows that the forward search is (unavoidably, in general) incomplete and therefore yields answers of the

[2]Work supported by European Union project PROFUNDIS, Contract No. IST–2001–33100.
[4]Work supported by Landesstiftung Baden–Württemberg, Grant No. 21–655.023.
[6]Work supported by European Union project ADVANCE, Contract No. IST–1999–29082.

form YES or UNKNOWN. Altogether, our method thus yields answers of the form YES, NO, or UNKNOWN.

In Section 2 we recall well–structured transition systems and some central results about them. In Section 3 we describe our method of combined backwards and forwards search in general terms. In the rest of the paper we apply it to relabelling–free CCS with finite summation, which is recalled in Section 4. In Section 5 we show how to encode the process calculus into generalised Petri nets and in Section 6 we show that these nets are well–structured with respect to an appropriate well–quasi–ordering on markings. In Section 7 we describe the application of the backward/forward search technique to these systems. Section 8 contains conclusions and topics for future work.

2 Preliminaries on Well–Structured Systems

For our purposes, we regard a *transition system* as a pair $\langle S, \rightarrow\rangle$ where $S = Q \times D$ is a set of *states* formed by a finite set Q of *control states* and a possibly infinite set D of *data values*, and $\rightarrow \subseteq S \times S$ is a set of *transitions* between states. A transition system is *labelled* if the transitions carry annotations from a finite set L of *labels*. We write $\langle s, s'\rangle \in \rightarrow$ as $s \rightarrow s'$, and $\langle s, s'\rangle \in \rightarrow$ with a label of α as $s \xrightarrow{\alpha} s'$. We denote the reflexive and transitive closure of $\rightarrow$ by $\rightarrow^*$. For any $S' \subseteq S$ and $\alpha \in L$ let $Pre_\alpha(S') := \{s \in S \mid \exists s' \in S'. s \xrightarrow{\alpha} s'\}$ and $Post_\alpha(S') := \{s \in S \mid \exists s' \in S'. s' \xrightarrow{\alpha} s\}$

Next, we recall *well–structured systems*, as well as some decidability results for such systems [2].

A *preorder* is a binary relation that is reflexive and transitive. A preorder $\preceq$ on a set D is (a) *decidable* if it is possible to effectively determine whether $a \preceq b$ for any a, b and (b) a *well–quasi–ordering* (wqo) if there is no infinite sequence $a_0, a_1, \ldots$, so that $a_i \not\preceq a_j$ for any $i < j$. A set $M \subseteq D$ is said to be *canonical* if $a, b \in M$ implies $a \not\preceq b$. M is a *minor set* of A if, for all $a \in A$, there is a $b \in M$ so that $b \preceq a$ and, at the same time, M is canonical.

Let D be a set. A subset $U \subseteq D$ is *upward closed* if whenever $a \in U, b \in D$ and $a \preceq b$, then $b \in U$. The *upward closure* of a set $A \subseteq D$ is the upward closed set $C(A) := \{b \in D \mid \exists a \in A. a \preceq b\}$. We say that $C(A)$ is *generated* by A. The following lemmas can be found proven in [2]:

Lemma 1. *If a preorder $\preceq$ is a well–quasi–ordering, then for each set A there exists at least one finite minor set of A.* □

Given a set A, we use Min to denote a function that returns a minor set of A. From the above lemma and the fact that $C(Min(U)) = U$ for each upward–closed set U, it follows that $Min(U)$ is a finite characterisation of U. Thus, a set $Min(U)$ can be used as a representative of U.

Lemma 2. *For a preorder $\preceq$ on a set A, $\preceq$ is a wqo if, and only if, for each infinite sequence $U_0 \subseteq U_1 \subseteq \ldots$ of upward–closed sets in A, there is a k so that $U_k = U_{k+1}$.* □

The forward direction of this lemma is important to prove termination of some verification algorithms based on well quasi–orderings.

Given a preorder $(D, \preceq)$ we define the preorder $(D^*, \preceq^*)$ on the set D^* of finite strings over D so that $x_1 \ldots x_m \preceq y_1 \ldots y_n$ if there is a strictly monotonic injection $f : \{1, \ldots, m\} \rightarrow \{1, \ldots, n\}$ where $x_i \preceq y_{f(i)}$ for $1 \leq i \leq m$. In a similar manner we define the preorder $(\mathcal{M}(D), \preceq^{\mathcal{M}})$ on the set $\mathcal{M}(D)$ of multisets over D: $A \preceq B$ if there is a multiset injection $f : A \Rightarrow B$ so that $x \preceq f(x)$ for all $x \in A$.

Theorem 1. *(Higman [10]) If $\preceq$ is a wqo on D, then $\preceq^*$ is a wqo on D^* and $\preceq^{\mathcal{M}}$ is a wqo on $\mathcal{M}(D)$.* □

For well–structured systems, we require that the set of data values be equipped with a decidable preorder $\preceq$. We also assume that we have a minor set of D, which we denote by D_{min}. We extend the preorder on D to a decidable preorder $\preceq$ on the set of states S of a transition system by $\langle q,d\rangle \preceq \langle q',d'\rangle$ if and only if $q=q'$ and $d \preceq d'$.

A labelled transition system $\langle S,\rightarrow\rangle$ is *monotonic* if, whenever $s \preceq t$ and $s \stackrel{\alpha}{\rightarrow} s'$, then $t \stackrel{\alpha}{\rightarrow} t'$ for some t' so that $t \preceq t'$.

Definition 1. *A transition system $\mathcal{L} = \langle S,\rightarrow\rangle$ with a decidable preorder $\preceq$ on the set of data values, is said to be* well–structured *if*

1. *$\preceq$ is a well–quasi–ordering and*
2. *$\langle S,\rightarrow\rangle$ is monotonic with respect to $\preceq$ and*
3. *for all $s \in S$ and $\alpha \in L$, the set $Min(Pre_\alpha(C(\{s\})))$ is computable.*

As emphasised in [2], $Min(Pre_\alpha(C(\{s\})))$ is finite if $\preceq$ is a well–quasi–ordering. We use $Minpre_\alpha(s)$ as shorthand for $Min(Pre_\alpha(C(\{s\})))$. The following theorem is the reason why we apply the theory of well–structured systems to infinite state processes:

Theorem 2. *Let $\langle S,\rightarrow\rangle$ be a well–structured transition system, $\langle q,d\rangle$ a state and U an upward–closed subset of the set of data values. Then it is decidable whether it is possible to reach, from $\langle q,d\rangle$, any state $\langle q',d'\rangle$ with $d' \in U$.* □

3 The Method in General Terms

Well–structured infinite–state transition systems have been studied intensively in recent years and several algorithms for reachability analysis have been developed (e.g., [2, 9, 7, 3, 5, 1]). Often the idea is that there exists a well–quasi–ordering on the set of configurations and thus any upward–closed set is characterised by its finitely many minimal elements. Using this fact, one can effectively compute a symbolic representation of the $Pre^*(U)$ (set of all predecessors) of any upward closed set of configurations. Checking if some element of an upward–closed set U is reachable from the set I of initial configurations can then be done by simply checking if $I \cap Pre^*(U) \neq \emptyset$.

Our aim is to make this technique also applicable to some classes of non–well–structured infinite–state systems, like CCS–like process calculi. Since the reachability problem is undecidable for these systems, one cannot expect to obtain any decision procedure. However, our method will yield answers of the form YES, NO, or UNKNOWN, where the YES and NO answers are always correct.

First we describe our approach in general terms. The problem we consider is the following. Let T be some non–well–structured transition system with set of initial states I. Let a be some atomic action and A the set of all configurations where a is enabled. The question is if, starting at some state in I, one can reach some state in A.

Our method works in several steps:

1. First one constructs a well–structured system $W(T)$, which is an overapproximation of T. $W(T)$ has the same underlying set of (potential) configurations as T, although the set of reachable configurations may be larger. As $W(T)$ is an overapproximation of T we must have that $\forall t_1, t_2 . (t_1 \rightarrow_T t_2 \;\Rightarrow\; t_1 \rightarrow_{W(T)} t_2)$. As $W(T)$ is well–structured, there exists a well–quasi–ordering on the set of configurations. In $W(T)$ the corresponding set $W(A)$ (with $A \subseteq W(A)$) is upward closed.
2. By backward search, one can now effectively construct the set $Pre^*_{W(T)}(W(A))$, the set of predecessors of $W(A)$ in $W(T)$. Since we consider an overapproximation we obtain $Pre^*_T(A) \subseteq Pre^*_{W(T)}(W(A))$. Now there are two cases.

If $I \cap Pre^*_{W(T)}(W(A)) = \emptyset$ then also $I \cap Pre^*_T(A) = \emptyset$ and thus in T the set A is not reachable from I. In this case our procedure gives the correct answer NO.
If $I \cap Pre^*_{W(T)}(W(A)) \neq \emptyset$ then nothing is decided yet. Maybe the set A is really reachable from I in T, but maybe it is just the case in the coarser overapproximation $W(T)$. So we continue with the next step.

3. One now does a forward search of reachable states in the system T, starting with the set I. However, we make the forward search more efficient by restricting it to those configurations that are in $Pre^*_{W(T)}(W(A))$. (Nothing is lost by this restriction, since we have $x \notin Pre^*_{W(T)}(W(A)) \Rightarrow x \notin Pre^*_T(A) \Rightarrow x \not\to^*_T A$). So we iteratively construct sets of configurations I_k s.t. $I_0 := I$ and $I_{k+1} := (I_k \cup Post_T(I_k)) \cap Pre^*_{W(T)}(W(A))$. Additionally one can combine this with acceleration methods to speed up the search. In this case one constructs $I_{k+1} := (I_k \cup H_k) \cap Pre^*_{W(T)}(W(A))$, for some H_k with $Post_T(I_k) \subseteq H_k \subseteq Post^+_T(I_k)$.
   One constructs these sets I_k up to some some fixed bound b for k. If $I_k \cap A \neq \emptyset$ for some $k \leq b$ then our procedure gives the correct answer YES. Otherwise the procedure returns the answer UNKNOWN.
   One can obtain a lower bound for the required b from the backwards search, but no upper bound, since the reachability problem is undecidable in general. The bound b must be at least as large as the minimal number s s.t. $I \cap Pre^s_{W(T)}(W(A)) \neq \emptyset$. This is because the shortest path from I to A in T can only be longer than the shortest path from I to $W(A)$ in $W(T)$, since $W(T)$ is an overapproximation of T (see Section 7). Thus, if one chose $b < s$ then the forward search could not possibly find A.

Remark 1. Alternatively, one might consider to abort the backwards–search at the smallest level s where $I \cap Pre^s_{W(T)}(W(A)) \neq \emptyset$, instead of computing the whole $Pre^*_{W(T)}(W(A))$. However, this has the big disadvantage that the following forward–search cannot be restricted. Restricting the forward–search to $Pre^*_{W(T)}(W(A))$ is safe (as shown above), but restricting it to $Pre^s_{W(T)}(W(A))$ is too restrictive. So, in this case, one would have to do an unrestricted forward–search, which is much less efficient. Therefore, we compute the whole $Pre^*_{W(T)}(W(A))$ as described above.

Our method has several advantages over a normal or even accelerated forward search where there is no prior backward search:

1. It can produce not only definitive YES but also definitive NO answers.
2. The forward search is more efficient, since the search space is restricted (but without any loss of information). This allows to use higher bounds b and thus a deeper and more complete forward search. Acceleration methods can still be used in the forward search.

In the following, we apply this method to a particular class of non–well–structured systems, which are described by a (still Turing-powerful) fragment of Milner's CCS [11]. The idea is to encode the CCS-processes into generalised coloured Petri nets and then to use a modified weaker variant of these nets as a well-structured overapproximation.

4 Agents

We consider a calculus that closely resembles Milner's CCS [11], except for (a) the lack of an operator for relabelling and (b) a binary operator for summation instead of one with generic — and potentially infinite — arity. We do not consider relabelling since it is not obvious how we could accommodate it within our Petri net semantics, where this semantics is, however, the link that allows us to apply well–quasi–orderings to the calculus; we do not consider infinite summation since our ultimate interest here lies in automated verification. As is well known, the state reachability problem for the calculus is undecidable, even though there is no relabelling. One can prove that by encoding a two–counter machine.

We assume a set *Chan* $:= \{c, \dots\}$ of *channels* and a set *Const* $:= \{A, \dots\}$ of *agent constants* to be given. The set *Act* $:= \{a, \dots\}$ of *visible actions* is derived via $Act = Chan \cup \{\overline{c} \mid c \in Chan\}$, where $c \neq \overline{d}$ and, at the same time, $c \neq d$ implies $\overline{c} \neq \overline{d}$ for all c, d. The set $Act_\tau := \{\alpha, \dots\}$ of *actions* is derived by $Act_\tau = Act \cup \{\tau\}$, $\tau \notin Act$. The $\overline{\cdot}$–operation is extended to a self–inverse *complementation operator* on this set via $\overline{\overline{c}} = c$ and $\overline{\tau} = \tau$.

The syntax of the language is given by the following BNF–like grammar clause:

$$P ::= \mathbf{0} \mid \alpha.P \mid P + Q \mid P|Q \mid P\backslash c \mid A \tag{1}$$

We use a standard interleaving SOS for agents, which is shown below:

$$\alpha.P \xrightarrow{\alpha} P \quad \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} \quad \frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'} \quad \frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q} \quad \frac{Q \xrightarrow{\alpha} Q'}{P|Q \xrightarrow{\alpha} P|Q'}$$

$$\frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\overline{a}} Q'}{P|Q \xrightarrow{\tau} P'|Q'} \quad \frac{P \xrightarrow{\alpha} P'}{P\backslash c \xrightarrow{\alpha} P'\backslash c}\ c \notin \{\alpha, \overline{\alpha}\} \quad \frac{P \xrightarrow{\alpha} P'}{A \xrightarrow{\alpha} P'}\ A \triangleq P$$

5 P–Nets: Finite Petri Nets for Finite– and Infinite–State Agents

We define p–nets as an extension of Petri nets, where the individual tokens carry information about their firing history. We show how the process calculus from the previous section can be encoded into p–nets. There are two different conditions on when p–net transitions are enabled to fire: First, there is *strong enabledness*, the standard notion for p–nets, where the firing history in the tokens is taken into account. Second, there is a weaker condition called *weak enabledness*, which ignores the token-histories and is more like the standard Petri net firing condition. Strong enabledness always implies weak enabledness, but not conversely. The precise definitions are given below.

P–nets with the strong enabledness condition for p–net transitions faithfully simulate the behaviour of the processes defined in the process calculus that we consider herein, but they are not well–structured systems. P–nets with the weak enabledness condition, however, are well–structured and form an overapproximation of the p–nets with the strong enabledness condition.

A p–net always depends on an environment of agent constant definitions. Hence, from now on, we fix an arbitrary such environment, defining the places, transitions and firings of the specific p–net arising from it.

Places The set *Place* $:= \{p, q, \dots\}$ contains all agent constants together with all agents and sub–agents that occur on the right–hand side of any defining equation within the environment.

Transitions Every p–net transition t is a pair of the form $\langle Pre, Post \rangle$, where *Pre* and *Post* are sets of places, which are called the *pre*– or *postset* of t. We denote *Pre* also by $Pre(t)$ and *Post* by $Post(t)$.

Then, first of all, a set $Trans(p)$ of preliminary p–net transitions for any $p \in Place$ is given as defined below. The "$\mapsto$"–symbol signifies labellings that are associated with some p–net transitions or arcs within p–net transitions.

$$
\begin{aligned}
\mathit{Trans}(\alpha.P) &= \left\{ \left\langle \{\alpha.P\}, \{P\} \right\rangle \mapsto \alpha \right\} \\
\mathit{Trans}(P+Q) &= \left\{ \left\langle \{P+Q\}, \{P\} \right\rangle, \left\langle \{P+Q\}, \{Q\} \right\rangle \right\} \\
\mathit{Trans}(P|Q) &= \left\{ \left\langle \{P|Q\}, \{P \mapsto \mathrm{l}, Q \mapsto \mathrm{r}\} \right\rangle \right\} \\
\mathit{Trans}(P \backslash c) &= \left\{ \left\langle \{P \backslash c\}, \{P\} \right\rangle \mapsto \backslash c \right\} \\
\mathit{Trans}(A) &= \left\{ \left\langle \{A\}, \{P\} \right\rangle \right\}, \text{ given that } A \stackrel{\Delta}{=} P
\end{aligned}
$$

What is left is to take possible synchronisation into account. That is done by the following definition of the overall set *Trans* of p–net transitions:

$$
\begin{aligned}
\mathit{Trans} = \left(\bigcup\nolimits_{p \in \mathit{Place}} \mathit{Trans}(p) \right) \cup \Big\{ & \left\langle \mathit{Pre}_1 \cup \mathit{Pre}_2, \mathit{Post}_1 \cup \mathit{Post}_2 \right\rangle \mapsto \langle \tau, c \rangle \mid \\
& \langle \mathit{Pre}_1, \mathit{Post}_1 \rangle \mapsto a \in \mathit{Trans}(p) \text{ for some } p \in \mathit{Place} \text{ and} \\
& \langle \mathit{Pre}_2, \mathit{Post}_2 \rangle \mapsto \overline{a} \in \mathit{Trans}(q) \text{ for some } q \in \mathit{Place}, \\
& \text{where } c \text{ is the channel on which } a \text{ and } \overline{a} \text{ occur} \qquad \Big\}.
\end{aligned}
$$

Later on we will use the fact that all p–net transitions have either one place or at most two places in their pre– and post–sets. In the first case we denote these places by $\mathit{pre}(t)$ and $\mathit{post}(t)$, where t is the transition; in the second case we denote these places by $\mathit{pre}_i(t)$ and $\mathit{post}_i(t)$, $i = 1, 2$, where the specific association between indexes and places does not matter as long as both $\mathit{pre}_i(t)$ and $\mathit{post}_i(t)$ are in the pre– or postset of the same transition labelled with a or $\overline{a}$. Also, we sum up what labels are used:

- α: On p–net transitions that correspond to action prefixes.
- l/r: On arcs leading to places in the postsets of p–net transitions that fork between concurrent components.
- $\langle \tau, c \rangle$: On p–transitions that correspond to synchronisations
- $\backslash c$: On p–net transitions that correspond to restrictions.

One might take note of the fact that agents of the form $P|P$ will lead to p–net transitions whose outgoing edge is labelled with both l and r.

Example 1. In Figure 1 we depict the p–net associated with an agent constant definition of the form $A \stackrel{\Delta}{=} c.\mathbf{0}|((\overline{c}.\mathbf{0}|A)\backslash c)$.

Tokens A *token* is a string from the set $(\mathit{Chan} \cup \{\mathrm{l}, \mathrm{r}\})^*$, assuming $\mathit{Chan} \cap \{\mathrm{l}, \mathrm{r}\} = \emptyset$. Let $\mathit{Tok} := \{\eta, \dots\}$ be the set of tokens. We denote the empty token, that is, the empty string by ϵ. Tokens carry history information with them in the sense that they keep track of in which concurrent threads and in which scopes with respect to restrictions they are. This information can then be used to guarantee that any firing of a synchronisation transition (a) only uses tokens that belong to concurrent threads and (b) the channel on which the synchronisation occurs is either free or restricted within the same scope. Moreover, any firing of a prefix transition with a visible label can be guaranteed to not use any token that is within a scope where the underlying channel is restricted.

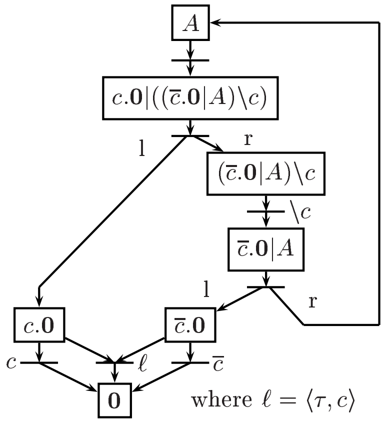

Fig. 1.

Markings A *marking* (usually denoted by m) of a p–net is a mapping from the places of the net to the set of all finite multisets of tokens. Every marking must be *non–empty* in the sense that at least one of the multi–sets in its range must be non–empty.

Firings As explained at the beginning of this section, we define two different conditions how transitions can be enabled for a given marking in our p–nets:

- A p–net transition is called *weakly enabled* for a given marking if there are tokens present on all its input places, just like in normal Petri nets.
- A p–net transition is *strongly enabled* if it is weakly enabled and, at the same time, the additional conditions explained below hold.

As we are dealing with coloured Petri nets whose tokens can change their colour, we need to put some machinery in place to express this discipline formally: We denote by $Pre(t) \Subset m$ the fact that each place p in the preset of t is such that $m(p) \neq \{\!|\,|\!\}$. Moreover, if $Pre(t) \Subset m$, then a *precondition* of t is a function from sets to multisets, usually denoted by pc, that associates an element of $m(p)$ with every $p \in Pre(t)$.

Then, given a marking m, a transition t and a precondition pc of t, t is weakly enabled if $Pre(t) \Subset m$, regardless of the choice of input tokens given by pc. If $|Pre(t)| = 1$ and t is not labelled with some a, that is, t is not a visible prefix transition, then t is strongly enabled too, also regardless of the choice of input tokens given by pc. If $|Pre(t)| = 1$ and t <u>is</u> labelled with some a, then t is strongly enabled if $m(pre(t))$ does not contain the underlying channel of a, that is, t must not be in a scope in which that channel is restricted. If $|Pre(t)| = 2$ then t is strongly enabled if the condition explained at the end of this paragraph is true. As a preliminary, we recall that $|Pre(t)| = 2$ means that t must be labelled with $\langle \tau, c \rangle$, for some c. As another preliminary, if $\eta \in Tok$, then we denote by $\eta \setminus Chan$ the substring of η that results from deleting all occurrences of letters in $Chan$. As still another preliminary, if $c \in Chan$, then $maxpref_c(\eta)$ is ϵ if c does not occur in η and, otherwise, the longest prefix η' of η so that $\eta' = \eta'' \bullet c$ for some η''. The condition that is to hold in the case of $|Pre(t)| = 2$, states that (i) $pc(pre_i(t)) \setminus Chan \neq \epsilon$, $i = 1, 2$, while $pc(pre_1(t)) \setminus Chan \neq pc(pre_2(t)) \setminus Chan$ and (ii) $maxpref_c(pc(pre_1(t))) = maxpref_c(pc(pre_2(t)))$. Part (i) means that the tokens that participate must belong to different concurrent threads; part (ii) means that the tokens must be in the same scope with respect to c. In the proof of Theorem 3 below we will need to refer to this condition again, and there we will denote it by $Sync(pc(pre_1(t)), pc(pre_2(t)), c)$.

Let t be strongly enabled for m and let pc be a precondition pc. Then a *firing* is a triple of the form $\langle m, \lambda, m' \rangle$, usually written as $m \xrightarrow{\lambda} m'$, where

- m is the *source marking*,
- λ is the *label* and
- m' is the *target marking* of the firing.

The label and the target marking are derived from the source marking and the choice of the $pc(\cdot)$–function, as explained below. By $M \oplus x$ and $M \ominus x$ we respectively denote the inclusion of x in or the exclusion of x from a multiset M.

- Predictably, the firing leaves places outside the pre– and postsets of t unaffected, so $m'(p) = m(p)$ for each $p \notin \mathit{Pre}(t) \cup \mathit{Post}(t)$.
- From places in the preset of t, the token specified by the $pc(\cdot)$–function is removed, so $m'(p) = m(p) \ominus pc(p)$ for each $p \in \mathit{Pre}(t)$.
- The label of the firing is a if that is t's label and τ if t is labelled with $\langle \tau, c \rangle$, for some c. Otherwise, the label is ϵ — The context will always distinguish this ϵ from the ϵ denoting the empty string.
- On places that belong to the postset of t, m' is specified as follows:
  - If t is not labelled with $\backslash c$ or $\langle \tau, c \rangle$, for some c, and if none of the arcs leading to the place(s) in $\mathit{Post}(t)$ is labelled with l or r, (which means that $|\mathit{Post}(t)| = 1$) then $m'(\mathit{post}(t)) = m(\mathit{post}(t)) \oplus pc(\mathit{pre}(t))$.
  - If t is labelled with $\backslash c$, for some c, (which means that $|\mathit{Post}(t)| = 1$) then $m'(\mathit{post}(t)) = m(\mathit{post}(t)) \oplus pc(\mathit{pre}(t)) \bullet c$.
  - If t is labelled with $\langle \tau, c \rangle$, for some c, (which means that $1 \leq |\mathit{Post}(t)| \leq 2$), then:
    - If $|\mathit{Post}(t)| = 2$, then $m'(\mathit{post}_i(t)) = m(\mathit{post}_i(t)) \oplus pc(\mathit{pre}_i(t))$, $i = 1, 2$.
    - If $|\mathit{Post}(t)| = 1$, then $m'(\mathit{post}(t)) = m(\mathit{post}(t)) \oplus pc(\mathit{pre}_1(t)) \oplus pc(\mathit{pre}_2(t))$,
  - If the arcs leading to $\mathit{post}_i(t)$ are labelled with d_i, $i = 1, 2$ and $d_i \in \{\mathrm{l}, \mathrm{r}\}$, (which means that $|\mathit{Post}(t)| \leq 2$ and $\{d_1, d_2\} = \{\mathrm{l}, \mathrm{r}\}$) then:
    - If $|\mathit{Post}(t)| = 2$, then $m'(\mathit{post}_i(t)) = m(\mathit{post}_i(t)) \oplus pc(\mathit{pre}(t)) \bullet d_i$, $i = 1, 2$.
    - If $|\mathit{Post}(t)| = 1$, then $m'(\mathit{post}(t)) = m(\mathit{post}(t)) \oplus pc(\mathit{pre}(t)) \bullet d_1 \oplus pc(\mathit{pre}(t)) \bullet d_2$.

The intuition behind p–net firings is to represent what could perhaps be called a kind of sub–atomic behaviour, that is, behaviour one level below the operational transition semantics. Specifically, net markings correspond to sub–atomic states and net firings to sub–atomic steps, where there are branches due to summation, forks due to parallel composition, backward jumps due to agent constants and so on. The behaviour can be regarded as sub–atomic since in general several firings of the eventual p–net are required to represent one SOS transition. What is crucial for us is that p–net firings coincide with the standard transition semantics in the sense that both generate the same observable traces up to firings labelled with ϵ. Theorem 3 states that formally. As a preliminary, given any agent P, let init_P be the marking that maps (i) P to the (singleton) multiset containing the empty token and (ii) all other places/agents to the empty multiset. Moreover:

1. The set $\mathit{TTrace}(P)$ of (possibly infinite) *transition traces* of P is given by

$$\mathit{TTrace}(P) = \{\alpha_1 \alpha_2 \ldots \mid \text{there are } P_0, P_1, \ldots \text{ so that } P_0 = P \text{ and } P_0 \xrightarrow{\alpha_1} P_1 \xrightarrow{\alpha_2} \ldots\}.$$

2. The set $\mathit{FTrace}(P)$ of (possibly infinite) *firing traces* of P is given by

$$\mathit{FTrace}(P) = \{\lambda_1 \lambda_2 \ldots \mid \text{there are } m_0, m_1, \ldots \text{ so that } m_0 = \mathit{init}_P \text{ and } m_0 \xrightarrow{\lambda_1} m_1 \xrightarrow{\lambda_2} \ldots\}.$$

Theorem 3. *For any agent P, the sets $\mathit{TTrace}(P)$ and $\mathit{FTrace}(P)$ coincide up-to the removal of all τ's from the traces in $\mathit{TTrace}(P)$ and of all τ's and all ϵ's from the traces in $\mathit{FTrace}(P)$.*

As the proof of Theorem 3 is somewhat long, we present it under its own sub–heading below.

We strongly conjecture that $TTrace(P)$ and $FTrace(P)$ in fact coincide up to the removal of all ϵ's from the traces in $FTrace(P)$, leaving all τ's in place. The reason why we nevertheless formulate Theorem 3 as above has to do with its proof, where it turns out to be convenient to appeal to a classical theorem about traces and may–testing. This result holds in general only if one strips the traces under consideration of all labels that are invisible. For our purposes that is of course irrelevant, as the safety properties we are interested in are only concerned with actions that are visible.

Proving Theorem 3

The proof of Theorem 3 employs two main auxiliary tools: CCS–like agents with an operator for internal choice and *decomposition sets*, a notion that effectively helps analysing p–net firings. Then there are four steps: Steps 1 to 3 are concerned with proving successive trace equivalences, considering agent transitions, decomposition set transitions as we are going to define them below and p–net firings; Step 4 just puts these equivalences together so that the validity of Theorem 3 is established. Step 2, being concerned with a trace equivalence of agents and decomposition sets, is in fact the one that bridges the widest gap. One could regard the entire setup as directed at making this step as easy as possible.

CCS–like Agents with an Operator for Internal Choice We refer to [8] as for background on CCS–like process calculi with an operator for internal choice. For our purposes, we employ the language below, where we denote internal choice, as it is common practice, by $\oplus$.

$$U ::= \mathbf{0} \mid \alpha.U \mid U \oplus V \mid U|V \mid U\backslash c \mid A \tag{2}$$

We should stress that our language has τ–prefixes, although one topic of the line of work to which [8] belongs is how to do without them while retaining all expressive power. It is easy to see that the result that we need — Theorem 4 — is not affected by that. Likewise, it is not affected by the presence of agent constants instead of the *rec*–operator, which is the usual means of expressing recursion in *CCS without τ's*, as it is sometimes called. We employ the language from (2) since its transition semantics is closer to how p–nets operate than the transition semantics of the language from (1) on page 5.

The transition semantics is given in SOS–style by the following axioms and clauses:

$$\alpha.U \xrightarrow{\alpha} U \qquad U \oplus V \xrightarrow{\epsilon} U \qquad U \oplus V \xrightarrow{\epsilon} V \qquad \frac{U \xrightarrow{\lambda} U'}{U|V \xrightarrow{\lambda} U'|V} \qquad \frac{V \xrightarrow{\lambda} V'}{U|V \xrightarrow{\lambda} U|V'}$$

$$\frac{U \xrightarrow{a} U' \quad V \xrightarrow{\overline{a}} V'}{U|V \xrightarrow{\tau} U'|V'} \qquad \frac{U \xrightarrow{\alpha} U'}{U\backslash c \xrightarrow{\alpha} U'\backslash c}\ c \notin \{\alpha, \overline{\alpha}\} \qquad A \xrightarrow{\epsilon} U \ \ A \stackrel{\Delta}{=} U$$

As another slight deviation from the usual ways in connection with CCS without τ's, we employ transitions with τ–labels. Again, it is easy to see that Theorem 4 is not affected by that. As compared to labelling all invisible transitions with ϵ, the benefit of retaining the τ–transitions is a clearer correspondence to decomposition set transitions as they are defined below.

To relate agents over (1) to agents over (2), we introduce a bijective *non–determinisation operator*. It is defined by $n(op(P_1, \ldots, P_{arity(op)})) = op(n(P_1), \ldots, n(P_{arity(op)}))$ for $op \neq +$ and $n(P + Q) = n(P) \oplus n(Q)$. Then we have the following classical result:

Theorem 4. *For any P, P and $n(P)$ are* may–testing equivalent *within the canonical framework that arises by admitting both $+$ and $\oplus$.*

Step 1: Equivalence of Traces of the Different Kinds of Agents By another classical result, may–testing equivalence is characterised by equivalent traces of visible actions, so we can state Corollary 1, our first step towards proving Theorem 3. As a preliminary, for any agent U, the set $\mathit{TTrace}(U)$ of (possibly infinite) transition traces of U is given by

$$\mathit{TTrace}(U) = \{\lambda_1\lambda_2\ldots \mid \text{there are } U_0, U_1, \ldots \text{ so that } U_0 = U \text{ and } U_0 \xrightarrow{\lambda_1} U_1 \xrightarrow{\lambda_2} \ldots\}.$$

Corollary 1. *For any agent P, the sets $\mathit{TTrace}(P)$ and $\mathit{TTrace}(n(P))$ coincide up-to the removal of all τ's from the traces in $\mathit{TTrace}(P)$ and both all τ's and all ϵ's from the traces in $\mathit{TTrace}(n(P))$.*

Decomposition Sets and their Transitions A *decomposition set* is a finite set of pairs of tokens and agents over (2), where we write $\langle\eta, U\rangle$ as $\eta{:}\,U$. A decomposition set D is *regular* if (i)–(iii) holds, where $\mathit{Tok}(D)$ is the multiset of tokens in D: (i) $\mathit{Tok}(D)$ is actually a set, that is, the tokens in D are unique; (ii) for every token of the form $\eta \bullet \mathrm{l} \bullet \theta_1$ in $\mathit{Tok}(D)$ there is also a token of the form $\eta \bullet \mathrm{r} \bullet \theta_2$ in $\mathit{Tok}(D)$, and conversely; (iii) there are not any two tokens η and θ in $\mathit{Tok}(D)$ so that η is a prefix of θ.

A labelled transition relation on decomposition sets is given as shown below, denoting by $X \dot\cup Y$ the disjoint union of any two sets X and Y and by $ch(a)$ the underlying channel of any $a \in \mathit{Act}$.

$$\begin{array}{l}
\{\eta{:}\,\alpha.U\} \,\dot\cup\, D \xrightarrow{\alpha} \{\eta{:}\,U\} \,\dot\cup\, D \text{ if } \alpha \neq \tau \text{ implies that } ch(\alpha) \text{ does not occur in } \eta \\
\{\eta{:}\,U \oplus V\} \,\dot\cup\, D \xrightarrow{\epsilon} \{\eta{:}\,U\} \,\dot\cup\, D \\
\{\eta{:}\,U \oplus V\} \,\dot\cup\, D \xrightarrow{\epsilon} \{\eta{:}\,V\} \,\dot\cup\, D \\
\{\eta{:}\,U|V\} \,\dot\cup\, D \xrightarrow{\epsilon} \{\eta \bullet \mathrm{l}{:}\,U, \eta \bullet \mathrm{r}{:}\,V\} \,\dot\cup\, D \\
\{\eta{:}\,U\backslash c\} \,\dot\cup\, D \xrightarrow{\epsilon} \{\eta \bullet c{:}\,U\} \,\dot\cup\, D \\
\{\eta{:}\,A\} \,\dot\cup\, D \xrightarrow{\epsilon} \{\eta{:}\,U\} \,\dot\cup\, D \text{ if } A \triangleq U \\
\{\eta{:}\,a.P, \theta{:}\,\overline{a}.V\} \,\dot\cup\, D \xrightarrow{\tau} \{\eta{:}\,U, \theta{:}\,V\} \,\dot\cup\, D \text{ if } \mathit{Sync}(\eta, \theta, ch(a))
\end{array} \tag{3}$$

It can easily be seen that decomposition set transitions preserve regularity.

Step 2: Equivalence of Agent and Decomposition Set Traces We introduce next a way of extracting agents over (2) from decomposition sets. That is done via a family $(\lhd_\eta)_\eta$ of binary predicates on those agents and decomposition sets, which is defined as follows:

$$X \lhd_\rho D \text{ if } \rho{:}\,X \in D \qquad \frac{X \lhd_{\rho\bullet\mathrm{l}} D \quad Y \lhd_{\rho\bullet\mathrm{r}} D}{X|Y \lhd_\rho D} \qquad \frac{X \lhd_{\rho\bullet c} D}{X\backslash c \lhd_\rho D}$$

There is, then, an operational correspondence via these predicates, that is, an operational correspondence between transitions of agents over (2) and decomposition set transitions. This lemma is the central and most difficult part of the proof of Theorem 3.

Lemma 3. *Suppose $U \lhd_\epsilon D$, where D is regular. Then:*

1. i. *Whenever $U \xrightarrow{\epsilon} U'$ then, for some regular D', $D \xrightarrow{\epsilon} D'$ and $U' \lhd_\epsilon D'$.*
   ii. *Whenever $U \xrightarrow{\alpha} U'$ then, for some regular D', $D \xrightarrow{\epsilon}{}^* \xrightarrow{\alpha} D'$ and $U' \lhd_\epsilon D'$.*
2. i. *Whenever $D \xrightarrow{\epsilon} D'$, then $U \lhd_\epsilon D'$ or, for some U', $U \xrightarrow{\epsilon} U'$ and $U' \lhd_\epsilon D'$.*
   ii. *Whenever $D \xrightarrow{\alpha} D'$ then, for some U', $U \xrightarrow{\alpha} U'$ and $U' \lhd_\epsilon D'$.*

Proof. We consider only Property 2.ii, as the other ones can be treated essentially along the same lines or complementarily. So suppose $U \lhd_\epsilon D$, where D is regular, and $D \xrightarrow{\alpha} D'$. First, we observe that, since D is regular, every pair $\theta{:}\,V$ in D must be referred to at some unique leaf of the derivation tree for the property $U \lhd_\epsilon D$. Now suppose $D \xrightarrow{\alpha} D'$ has been established via the second axiom in (3) — We consider only this case as the other ones are easier. So $D \xrightarrow{\alpha} D'$ is of the form $D \xrightarrow{\tau} D'$ and has been derived from the presence of some $\theta{:}\, a.V$ and some $\kappa{:}\, \overline{a}.W$ in D, where the predicate $Sync(\theta, \kappa, ch(a))$ holds. By our initial observation, $\theta{:}\, a.V$ and $\kappa{:}\, \overline{a}.W$ occur at unique leafs of the derivation tree for $U \lhd_\epsilon D$. We continue by moving down the derivation tree for $U \lhd_\epsilon D$, collecting further derivable transitions as follows:

– At any node where the rule

$$\frac{X \lhd_{\rho\bullet \mathrm{l}} D \quad Y \lhd_{\rho\bullet \mathrm{r}} D}{X|Y \lhd_\rho D}$$

is applied there are the following sub–cases:
 - There is a derivable transition $X \xrightarrow{\alpha} X'$ but none for Y, or a derivable transition $Y \xrightarrow{\alpha} Y'$ and none for X. In this case the next derivable transition is $X|Y \xrightarrow{\alpha} X'|Y$ or $X|Y \xrightarrow{\alpha} X|Y'$, respectively.
 - There is a derivable transition $X \xrightarrow{a} X'$ and a derivable transition $Y \xrightarrow{\overline{a}} Y'$, or a derivable transition $X \xrightarrow{\overline{a}} X'$ and a derivable transition $Y \xrightarrow{a} Y'$. In this case the next derivable transition is $X|Y \xrightarrow{\tau} X'|Y'$.
 In fact this situation occurs exactly once, and it is also not possible that two transitions with the same labels meet at the same node, the reason for both being twofold:
 a. Having started out at exactly two leafs, where the labels of the initial transitions are complementary to each other, we are moving down a tree.
 b. It must hold that ρ is a prefix of both θ and κ. Hence, by $Sync(\theta, \kappa, ch(a))$, it holds that $maxpref_{ch(a)}(\theta)$ — which is the same as $maxpref_{ch(a)}(\kappa)$ — is a prefix of ρ, so we can be sure that we never have to propagate a transition labelled with a or $\overline{a}$ over a restriction in $ch(a)$ before we switch over to transitions labelled with τ, where these transitions can then be propagated over any restriction.

– At any node where the rule

$$\frac{X \lhd_{\rho\bullet c} D}{X\backslash c \lhd_\rho D}$$

is applied, there is a derivable transition $X \xrightarrow{\alpha} X'$. As we have already shown, it must hold that $c \notin \{\alpha, \overline{\alpha}\}$, so the next derivable transition is $X\backslash c \xrightarrow{\alpha} X'\backslash c$.

So we arrive at a transition $U \xrightarrow{\tau} U'$. It remains to show $U' \lhd_\epsilon D'$ but that is a straightforward matter of going through the derivation tree again, this time considering the transition residuals.

Taking into account that $\{\epsilon{:}\, U\}$ is trivially regular for any U, it is now straightforward to conclude:

Corollary 2. *For any agent U, the sets $TTrace(U)$ and $DTrace(U)$ coincide up to the removal of all ϵ's.*

Step 3: Equivalence of Decomposition Set and P–Net Traces For any agent U, the set $DTrace(U)$ of (possibly infinite) *decomposition traces* of U is given by

$$DTrace(U) = \{\lambda_1\lambda_2 \ldots \mid \text{there are } D_0, D_1, \ldots \text{ so that } D_0 = \{\epsilon{:}\, U\} \text{ and } D_0 \xrightarrow{\lambda_1} D_1 \xrightarrow{\lambda_2} \ldots\}.$$

Lemma 4. *For any agent P, the sets $FTrace(P)$ and $DTrace(n(P))$ coincide.*

Proof. (Outline) To begin with, a marking m is *set–based* if $m(P)$ is a set for all P. It can readily be seen that p–net firings preserve this property. Then, for any set–based marking m, let a decomposition set $D(m)$ be given by $D(m) = \{\eta{:}\, n(P) \mid \eta \in m(P)\}$, and let m be *regular* if $D(m)$ is regular.

Because m is set–based, m and $D(m)$ contain the same information, so one can then easily show that $m \xrightarrow{\lambda} m'$ implies $D(m) \xrightarrow{\lambda} D(m')$ and, conversely, $D(m) \xrightarrow{\lambda} D'$ implies $m \xrightarrow{\lambda} m'$ for some m' so that $D' = D(m')$. Moreover, as $init_P$ is always regular, and as p–net firings preserve regularity in the same way as decomposition set transitions, all reachable markings are regular. These properties together entail Lemma 4.

Step 4: Concluding the Proof of Theorem 3

Proof. (Theorem 3) As a direct consequence of Corollary 1, Corollary 2 and the previous lemma.

6 P–Nets as Well–Structured Systems

In this section we introduce a preorder $\sqsubseteq$ on markings and prove the prerequisite for applying the method described in Section 3 to p–nets, namely that the firing discipline from the previous section gives rise to a well–structured system with respect to $\sqsubseteq$.

First of all, a preorder on tokens is given by $\eta \preceq \theta$ if η is a (not necessarily contiguous) substring of θ. By Higman's theorem (cf. Section 2), this preorder is a wqo if the underlying alphabet is finite, a property that holds as far as our purposes are concerned since we always consider the safety properties of a single p–net with finitely many transition and arc labels. A preorder $\preceq^{\mathcal{M}}$ on multisets of tokens is then given as described for general multisets over wqo's in Section 2. Again by Higman's theorem, this preorder is a wqo since the preorder on tokens is a wqo. Finally, a preorder $\sqsubseteq$ on the markings of any specific p–net N or, equivalently, a preorder on P–indexed families of token multisets, where P is the set of places of N, is given by $m \sqsubseteq m'$ if $m(p) \preceq^{\mathcal{M}} m'(p)$ for all $p \in P$. By a corollary of Higman's theorem, this preorder is also a wqo. The intuition is that $m \sqsubseteq m'$ if m' represents a (not necessarily strictly) longer firing history than m.

It is not difficult to see that basing a preorder on markings on the prefix relationship between strings over $Chan \cup \{\mathrm{l}, \mathrm{r}\}$ would not make sense in the sense of representing firing histories and deriving a sensible wqo. In fact we believe that our wqo on markings is canonical and, moreover, the reachability problem will of course remain undecidable regardless of any other wqo one might choose. It is therefore not possible to improve our backward/forward strategy as applied to the agents from Section 4 qualitatively. It might only be possible to improve it merely quantitatively in the sense that (definitive) YES or NO answers are produced in higher numbers. It is, however, not obvious how that could be achieved on the basis of a different preorder on markings.

Before we prove that the wqo on markings gives rise to a well–structured system, we still need to see why there is no well–structuredness if one considers only strongly enabled firings (*strong firings*). Of course well–structuredness can not hold already in the light of Theorems 2 and 3 in combination with the fact that the reachability problem over CCS without relabelling is undecidable. Still, more elementary arguments lead to the same result: For one, if t is a synchronisation transition, then we may have tokens η_i on $pre_i(t)$, $i = 1, 2$, that may fire strongly whereas, in a bigger marking, tokens θ_i that can not fire strongly may be on $pre_i(t)$, where $\eta_i \preceq \theta_i$; similarly, if t is a visible prefix transition, we may have a token η on $pre(t)$ that may fire strongly whereas, in a bigger marking, a token θ that can not fire strongly may be on $pre(t)$, where $\eta \preceq \theta$. In other words, strong firings are not monotonic, so well–structuredness does not hold even disregarding the computability of $Minpre_\lambda$.

Lemma 5. *P–nets with weakly enabled firings (*weak firings*) are well–structured systems, given that the underlying environment of agent constant definitions is finite.*

Proof. We know already that the preorder on markings is a wqo and, since a finite environment of agent constant definitions entails finite sets of channels, tokens and markings, this preorder is also decidable. So it remains to show that transitions are monotonic and that $Minpre_\lambda$ is computable.

Monotonicity of weak p–net firings: Suppose $m_1 \sqsubseteq m_2$ and $m_1 \xrightarrow{\lambda} m_1'$. We have to find an m_2 so that $m_2 \xrightarrow{\lambda} m_2'$ and $m_1' \sqsubseteq m_2'$. Every weak firing involves the firing of exactly one p–net transition, so we can conduct a case analysis on what kinds of p–net transitions there are and in what ways m_1' can depend on m_1, to prove that there is always a weak firing under m_2 so that the subsequent marking, m_2', is bigger than m_1' under $\sqsubseteq$. All of that is straightforward.

Computability of Minpre_λ: To show that we can compute the set of minimal predecessor markings under λ of the upward closure $C(\{m\})$ of any set $\{m\}$, we can again conduct a case analysis on what kinds of transitions there are, distinguishing as to whether λ is equal to a, for some a , ϵ or τ. We skip the case of $\lambda = a$ for some a, since it is simpler than the other ones. The idea is to compute a finite initial contribution to an overapproximation of $\mathit{Minpre}_\lambda(m)$ for each net transition t. That overapproximation itself is finite since the p–net is finite, so it is possible to compute the minimal markings in it, obtaining $\mathit{Minpre}_\lambda(m)$.

$\lambda = \tau$: In this case we need to consider all invisible prefix transitions and all synchronisation transitions that belong to the p–net. The case of any arbitrary synchronisation transition t is more difficult, so we consider only this one. We know that $|\mathit{Pre}(t)| = 2$ and $1 \leq |\mathit{Post}(t)| \leq 2$. The idea is to derive a finite set K of markings so that $m \preceq k$ for each $k \in K$ and, for every marking h with $m \preceq h$ and every firing $h' \xrightarrow{\tau} h$ via t, there is a k' so that $k' \preceq h'$ and $k' \xrightarrow{\tau} k$ via t for some $k \in K$. Then the set of all k' with $k' \xrightarrow{\tau} k$ via t for some $k \in K$ is t's contribution to the above–mentioned overapproximation of $\mathit{Minpre}_\tau(m)$. There can only be finitely many such k' since K is finite and, at the same time, the p–net is finite.

Notation. For any function f, we denote by $f[x := y]$ the function given by $f[x := y](z) = f(z)$ if $z \neq x$ and $f[x := y](z) = y$ if $z = x$.

If $|\mathit{Post}(t)| = 2$, then $K = \{m, m[\mathit{post}_1(t) := m(\mathit{post}_1(t)) \oplus \epsilon], m[\mathit{post}_2(t) := m(\mathit{post}_2(t)) \oplus \epsilon], m[\mathit{post}_i(t) := m(\mathit{post}_i(t)) \oplus \epsilon]_{i=1,2}\}$; if $|\mathit{Post}(t)| = 1$, then $K = \{m, m[\mathit{post}(t) := m(\mathit{post}(t)) \oplus \epsilon], m[\mathit{post}(t) := m(\mathit{post}(t)) \oplus \epsilon \oplus \epsilon]\}$. Obviously it holds that $m \preceq k$ for each $k \in K$. Then, whenever $m \preceq h$ and $h' \xrightarrow{\tau} h$ is a firing via t, recall that $m \preceq h$ entails that, for each $p \in \mathit{Place}$, there is a multiset injection $f_p : m(p) \Rightarrow h(p)$ so that $\eta \preceq f_p(\eta)$ for each $\eta \in m(p)$. In the light of this observation, it is easy to see that, by the way we have derived K, there is always a $k \in K$ with the desired properties, since $\epsilon \preceq \eta$ for every token η.

$\lambda = \epsilon$: In this case there are various kinds of transitions to be considered but we restrict ourselves to those that are labelled with $\backslash c$, for some c, or whose outgoing arcs are labelled with l and r. In fact we always derive a set of markings K with the same properties as in the above–described case of a synchronisation transition: If t is labelled with $\backslash c$, then $K = \{m, m[\mathit{post}(t) := m(\mathit{post}(t)) \oplus c]\}$; if the outgoing arcs of t are labelled with l and r, then (a) $K = \{m, m[\mathit{post}_1(t) := m(\mathit{post}_1(t)) \oplus \mathrm{l}], m[\mathit{post}_2(t) := m(\mathit{post}_2(t)) \oplus \mathrm{r}], m[\mathit{post}_1(t) := m(\mathit{post}_1(t)) \oplus \mathrm{l}][\mathit{post}_2(t) := m(\mathit{post}_2(t)) \oplus \mathrm{r}]\}$ if $|\mathit{Post}(t)| = 2$, assuming without loss of generality that the arc leading to $\mathit{post}_1(p)$ is labelled with l and that the arc leading to $\mathit{post}_2(t)$ is labelled with r, and (b) $K = \{m, m[\mathit{post}(t) := m(\mathit{post}(t)) \oplus \mathrm{l}], m[\mathit{post}(t) := m(\mathit{post}(t)) \oplus \mathrm{r}], m[\mathit{post}(t) := m(\mathit{post}(t)) \oplus \mathrm{l} \oplus \mathrm{r}]\}$ if $|\mathit{Post}(t)| = 1$.

We can now conclude, from Theorem 2 and Lemma 5:

Theorem 5. *The control state reachability problem is decidable for p–nets with weak firings (in the sense of Theorem 2).*

7 Safety Properties Verification

What is called the decidability of control-state reachability over p–nets translates to a strategy for curbing search spaces in verifying safety properties of finite and infinite-state agents as we have introduced in Subsection 4. It is only possible to deal with those safety properties of any agent A that can be expressed as upward–closed sets of markings of the p–net associated with A, but there are no other restrictions. Of course, even these kinds of safety properties are undecidable in general. However, our method can still solve many instances of them. Furthermore, our method is more efficient than normal forward–search techniques, since it is possible to curb the search space for the forward search by using the information computed during the backwards search phase. Altogether, our method produces answers of the form YES, NO, or UNKNOWN, where the YES and NO answers are always correct.

In the following we thus show how to apply the general principle described in Section 3 to our fragment of CCS and p–nets. For describing the strategy more formally, we stick to the problem of verifying whether some agent A may ever execute any given action a or not. We assume that the underlying channel of a is un–restricted.

The problem:

Instance: An agent A with initial state ini and an atomic action a.
Question: Can agent A ever execute action a?

Preparatory phase:

1. Build the p–net N associated with A.
2. Let M^a be the set of minimal markings where a is immediately executable. For every transition t labelled with a there is a minimal marking m_t that enables t. It is given by an ϵ–token on all places in $pre(t)$. Then $M^a = \{m_t \,|\, t \text{ labelled by } a\}$.

Backward search phase: Solve the control-state reachability problem for the upward closure of the set M^a. If any marking in this upward closure can be reached from the initial marking, then A can reach a state in which a is immediately executable. Conversely, action a will never be executed if the upward closure of M^a turns out not to be reachable from the initial state. This outcome is synonymous with the initial state not being contained in $Pre^*(M^a)$, the last state set generated in performing the backward search. Hence, if the initial state does belong to that set, then the answer is still open and forward search will commence.

The function *Reachability* takes as input the p–net, the initial state ini and the set of markings M^a. It returns YES, NO, or UNKNOWN.

The function $Search_{backward}(M^a, ini)$ searches backwards in p–net with the weak firing rule, i.e., in the overapproximation. It returns a result of the form (OB, s) where $OB = Pre^*_{\text{overapprox}}(M^a)$ and $s \in \mathbb{N}$ is the minimal number of steps that the backwards search needed to reach ini in the overapproximation of the system (or -1 if it never did). Termination is guaranteed by the fact that p–nets with the weak firing rule are well–structured systems (see Lemma 5 and Lemma 5).

If the initial state ini is **not** in $OB = Pre^*_{overapprox}(M^a)$ then surely no marking in the upward-closure of M^a is reachable either in the agent A and thus the answer is NO. Otherwise one starts the forward search.

$$
\begin{array}{l}
\textbf{function } Reachability(N, M^a, ini): \\
\quad (OB, s) := Search_{backward}(M^a, ini) \\
\quad \textbf{if } ini \notin OB \\
\quad \textbf{then} \longleftarrow \text{NO} \\
\quad \textbf{else} \longleftarrow Search_{forward}(ini, M^a, OB, b(s))
\end{array}
$$

In this algorithm $b(\_)$ is some fixed function that is used to determine the search–depth of the forward search. We must have $\forall x.\, b(x) \geq x$, otherwise the forward-search cannot possibly return a positive answer.

function $Search_{backward}(M^a, ini)$
$M_0 := M^a$; $k := 0$; $s := -1$
repeat
  if $(s = -1 \wedge ini \in M_k)$ **then** $s := k$
  $M_{k+1} := M_k \cup Minpre(M_k)$
  $k := k + 1$
until $M_{k+1} = M_k$
$\longleftarrow (M_k, s)$

Forward search phase: In the forward search phase we do no longer use the overapproximation, but the regular model (the p–net with the strong firing condition), in order to avoid getting false positives. We search forward from the initial marking ini to find out whether at least one marking that enables action a is reachable. That means that we are searching for a reachable marking that is bigger than (or equal to) some marking in M^a. We use only the condition of strong enabledness for transitions, in order to stay regular by avoiding irregular synchronisations. We restrict our forward search to those configurations that are in our already computed overapproximation OB of $Pre^*(M^a)$, since no marking bigger than some marking in M^a can possibly be reached from any configurations that are not in this set. This curbs the search space and makes the search more efficient and thus allows us to use larger numbers $b(s)$ than it would otherwise be practically possible.

function $Search_{forward}(ini, M^a, OB, b)$
$j := b$;
$I := \{ini\}$;
while $j \geq 0$
  $I := (I \cup Post_{strong}(I)) \cap OB$
  $j := j - 1$
  if $\exists m' \in I, m \in M^a. m \preceq m'$ **then** $\longleftarrow$ YES
$\longleftarrow$ UNKNOWN

In this algorithm the test $\exists m' \in I, m \in M^a. m \preceq m'$ is computable, because M^a is finite, and I is finite since global transitions on p–nets generate a finitely branching transition system. The function $Post_{strong}(I)$ returns the set of immediate successors of set I in the regular system (not the overapproximation), with the strong firing rule.

The algorithm presented here is not a semi–decision procedure, since there is no guarantee that the algorithm will give always the answer YES (or NO) if the action is (not) reachable. Because the forward search is restricted to depth $b(s)$, it is incomplete in the sense that the true answer YES might not be found in some cases, and instead answer UNKNOWN is produced. On the other hand, a true answer NO might also be missed. The overapproximation might be too coarse and thus the backward search might miss the correct NO answer and hand the problem over to the forward search which will produce the answer UNKNOWN. However, if our algorithm produces answer YES or NO then these are always correct.

Finally, it is possible to combine our method with acceleration techniques in the forward search. Thus one would replace the assignment $I := (I \cup Post_{strong}(I)) \cap OB$ in the algorithm above by $I := (I \cup Accelerate(I)) \cap OB$, where the function $Accelerate(I)$ computes a partial acceleration of the forward search, for example by looking for repeatable sequences of transitions and computing their least fixpoint. This can be done in many different ways, provided that it satisfies the following condition, which is needed for the progress and correctness of the acceleration:

$$Post_{strong}(I) \subseteq Accelerate(I) \subseteq Post^{+}_{strong}(I)$$

8 Conclusion and Further Work

We have described a general technique for reachability analysis of non–well–structured systems. It combines a complete backward reachability analysis in a well–structured over–approximation of the system with an incomplete forward search. This forward search is restricted to the previously computed backward reachable set. Our method produces answers of the form YES, NO, or UNKNOWN, where the YES and NO answers are always correct. We have also applied this general method to relabelling–free CCS with finite sums and we have shown how to construct a well–structured overapproximation by what we have called p–nets. Possible topics for future work include the following ones: (1) More detailed studies of acceleration methods that can be used in the forward search; (2) generalising our method to other classes of non–well–structured systems such as the π–calculus; (3) Implementing our algorithm in a tool.

Related Work As for related work, we have to mention [6], where well–structured systems are employed in deciding termination in a restricted π–calculus. The way well–structured systems are brought to bear in [6] is not Petri–net–based and therefore difficult to compare to our approach. Moreover, as safety properties of the kind we have considered are undecidable, any attack on the lines of [6] can never improve on our results qualitatively.

Petri net semantics for CCS–like process calculi appeared approximately from the mid–eighties onwards (see, for example, [12]) and more recently there has been a resurgence of this topic [4]. The semantics presented here is set apart by its heavy reliance on coloured tokens in combination with the fact that every p–net is finite, where that finiteness is obviously crucial for applying our method for verifying safety properties of infinite–state systems. In [4] infinite nets may occur. Moreover, other work on Petri net semantics for process calculi including [4] generally deals with relabelling whereas it is not obvious how we could modify our framework to do that.

Acknowledgement We would like to thank Parosh Aziz Abdulla and Frank Valencia for discussions related to the topics of this paper.

References

1. P. Abdulla, A. Bouajjani, and B. Jonsson. On-the-fly Analysis of Systems with Unbounded, Lossy Fifo Channels. In *10th Intern. Conf. on Computer Aided Verification (CAV'98)*. LNCS 1427, 1998.
2. P. Abdulla, K. Cerans, B. Jonsson, and T. Yih-Kuen. Algorithmic analysis of programs with well quasi-ordered domains. *Information and Computation*, 160:109–127, 2000.
3. P. Abdulla and B. Jonsson. Verifying Programs with Unreliable Channels. In *LICS'93*. IEEE, 1993.
4. M. Bernardo, N. Busi, and M. Ribaudo. Compact net semantics for process algebras. In *Formal Description Techniques for Distributed Systems and Communication Protocols and Protocol Specification, Testing and Verification*, pages 319–334. Kluwer, 2000.
5. A. Bouajjani and R. Mayr. Model checking lossy vector addition systems. In *Proc. of STACS'99*, volume 1563 of *LNCS*. Springer Verlag, 1999.
6. N. Busi, M. Gabbrielli, and G. Zavattaro. Replication vs. recursive definitions in channel–based calculi. In *ICALP 2003 proceedings*, LNCS 2719, pages 133–144. Springer-Verlag, 2003.
7. G. Cécé, A. Finkel, and S.P. Iyer. Unreliable Channels Are Easier to Verify Than Perfect Channels. *Information and Computation*, 124(1):20–31, 1996.
8. R. De Nicola and M. Hennessy. Testing Equivalences for Processes. *Theoretical Computer Science*, 34:83–133, 1983.
9. A. Finkel and Ph. Schnoebelen. Well structured transition systems everywhere! *Theoretical Computer Science*, 256(1-2):63–92, 2001.
10. G. Higman. Ordering by divisibility in abstract algebras. *Proc. London Math. Soc.*, 2:326–336, 1952.

11. R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.
12. E.-R. Olderog. *Nets, Terms and Formulas: Three Views of Concurrent Processes and Their Relationship*. Cambridge University Press, 1991.

Recent technical reports from the Department of Information Technology

2003-045 Sven-Olof Nyström: *A Polyvariant Type Analysis for Erlang*

2003-046 Martin Karlsson: *A Power-Efficient Alternative to Highly Associative Caches*

2003-047 Jimmy Flink: *Simuleringsmotor för tågtrafik med stöd för experimentell konfiguration*

2003-048 Timour Katchaounov and Tore Risch: *Interface Capabilities for Query Processing in Peer Mediator Systems*

2003-049 Martin Nilsson: *A Parallel Shared Memory Implementation of the Fast Multipole Method for Electromagnetics*

2003-050 Alexandre David: *Hierarchical Modeling and Analysis of Timed Systems*

2003-051 Pavel Krcal and Wang Yi: *Decidable and Undecidable Problems in Schedulability Analysis Using Timed Automata*

2003-052 Magnus Svärd and Jan Nordström: *Well Posed Boundary Conditions for the Navier-Stokes Equations*

2003-053 Erik Bängtsson and Maya Neytcheva: *Approaches to Reduce the Computational Cost when Solving Linear Systems of Equations Arising in Boundary Element Method Discretizations*

2003-054 Martin Nilsson: *Stability of the Fast Multipole Method for Helmholtz Equation in Three Dimensions*

2003-055 Martin Nilsson: *Rapid Solution of Parameter-Dependent Linear Systems for Electromagnetic Problems in the Frequency Domain*

2003-056 Parosh Aziz Abdulla, Johann Deneux, Pritha Mahata, and Aletta Nylén: *Forward Reachability Analysis of Timed Petri Nets*

2003-057 Erik Berg: *Low-Overhead Spatial and Temporal Data Locality Analysis*

2003-058 Erik Berg: *StatCache: A Probabilistic Approach to Efficient and Accurate Data Locality Analysis*

2003-059 Jonas Persson and Lina von Sydow: *Pricing European Multi-asset Options Using a Space-time Adaptive FD-method*

2003-060 Pierre Flener: *Realism in Project-Based Software Engineering Courses: Rewards, Risks, and Recommendations*

2003-061 Lars Ferm and Per Lötstedt: *Space-Time Adaptive Solution of First Order PDEs*

2003-062 Emilio Tuosto, Björn Victor, and Kidane Yemane: *Polyadic History-Dependent Automata for the Fusion Calculus*

2003-063 Michael Baldamus, Joachim Parrow, and Björn Victor: *Spi Calculus Translated to π-Calculus Preserving May-Testing*

2003-064 Arnim Brüger, Bertil Gustafsson, Per Lötstedt, and Jonas Nilsson: *High Order Accurate Solution of the Incompressible Navier-Stokes Equations*

UPPSALA
UNIVERSITET

December 2003
ISSN 1404-3203
http://www.it.uu.se/