On Combining the Stubborn Set Method with the Sleep Set Method

Kimmo Varpaaniemi

Helsinki University of Technology, Digital Systems Laboratory
Otakaari 1, SF-02150 Espoo, Finland
Kimmo.Varpaaniemi@hut.fi

Abstract. *Reachability analysis* is a powerful formal method for analysis of concurrent and distributed finite state systems. It suffers from the *state space explosion problem*, however: the state space of a system can be far too large to be completely generated. This paper considers two promising methods, Valmari's *stubborn set method* and Godefroid's *sleep set method*, to avoid generating all of the state space when searching for *undesirable reachable terminal states*, also called *deadlocks*. These methods have been *combined* by Godefroid, Pirottin, and Wolper to further reduce the number of inspected states. However, the combination presented by them places assumptions on the stubborn sets used. This paper shows that at least in *place/transition nets*, the stubborn set method can be combined with the sleep set method in such a way that all reachable terminal states are found, without having to place any assumption on the stubborn sets used. This result is shown by showing a more general result which gives a sufficient condition for a method to be compatible with the sleep set method in the detection of reachable terminal states in place/transition nets.
Topics: system verification using nets, analysis and behaviour of nets

1 Introduction

Reachability analysis, also known as *exhaustive simulation* or *state space generation*, is a powerful formal method for detecting errors in such concurrent and distributed systems that have a finite state space. It suffers from the so called *state space explosion problem*, however: the state space of the system can be far too large with respect to the time and other resources needed to inspect all states in the space. Fortunately, errors such as *undesirable reachable terminal states*, also called *deadlocks*, can be detected in a variety of cases without inspecting all reachable states of the system. What makes deadlocks especially interesting is the fact that the verification of a *safety property* can often be reduced to the detection of deadlocks, as shown by Godefroid and Wolper [8] among others.

This paper concentrates on the problem of detecting *reachable terminal states* in *place/transition nets* [15], a class of Petri nets. Two promising methods are studied: Valmari's *stubborn set method* [16, 17, 18, 19, 20] and Godefroid's *sleep set method* [4, 5, 6, 7, 8, 24, 25]. Both methods utilize the *independence of transitions* to cut down on the number of states inspected during the search. These methods have also been *combined* by Wolper and Godefroid [24], Godefroid and Pirottin [7], and Wolper, Godefroid, and Pirottin [25] to further reduce the number of inspected states. All of these methods guarantee that all reachable terminal states are found if the complete state space is finite.

The application of the stubborn set method and the sleep set method is not limited to Petri nets. The methods have been applied to several models of concurrency by Valmari [17, 20], Godefroid and Pirottin [7], Wolper and Godefroid [24], and Peled [13] among others. Valmari [16], and Godefroid and Kabanza [6] have presented how the stubborn set method and the sleep set method can be used to improve the *graph search methods* used for artificial intelligence. The independence of rules in production systems of artificial intelligence resembles the independence of actions in concurrent and distributed systems in many senses. The essential point in the stubborn set method and the sleep set method is that they utilize the independence of actions or rules. *Refined independence relations* are important since the more refined is the independence relation the less states usually have to be inspected. Best and Lengauer [2], and Katz and Peled [10] among others have studied refined independence relations and developed general concepts of independence.

The application of the stubborn set method and the sleep set method is not limited to the detection of reachable terminal states either. Both methods can be extended to verify properties expressed as linear temporal logic formulae as shown by Valmari [18, 20, 21], Wolper and Godefroid [24], and Peled [13].

The stubborn set method is closely related to, though not necessarily based on Overman's algorithms [12]. These, according to Valmari [16, 20], are somewhat limited and not so efficient as the stubborn set algorithms. The stubborn set method can also be considered a dynamic priority method in contrary to the static priority method mentioned by Valmari and Tiusanen [22], Rauhamaa [14], and Valmari [20] among others. The static priority method is in turn a generalization of the virtual coarsening of atomic actions presented by Ashcroft and Manna [1].

The sleep set method was originally inspired by Mazurkiewicz's *trace theory* [11]. An early version of the sleep set method [4] was essentially faithful to Mazurkiewicz's trace semantics. Later, inspired by Katz's and Peled's work [10], the method has been refined to take into account *conditional independence* [7]. As suggested in [24, 25] and seen in this work, representing traces is sometimes not necessary at all.

Katz and Peled have, independently of Valmari, developed verification algorithms that use *faithful decompositions* [9]. Peled [13] states that faithful decompositions are similar to stubborn sets. Peled has recently improved and extended [13] Valmari's and Godefroid's linear temporal logic verification algorithms.

The contribution of this paper can be described as follows: The combination of the stubborn set method and the sleep set method presented by Godefroid, Pirottin, and Wolper [7, 24, 25] places assumptions on the stubborn sets used. This paper shows that at least in place/transition nets, the stubborn set method can be combined with the sleep set method in such a way that all reachable terminal states are found, without having to place any assumption on the stubborn sets used. This result is shown by showing a more general result which gives a sufficient condition for a method to be compatible with the sleep set method in the detection of reachable terminal states in place/transition nets.

The rest of this paper has been organized as follows: in Section 2, we introduce place/transition nets. The presentation does not go beyond what is necessary for the remaining sections. In Section 3, dynamically stubborn sets [14, 19] are shown to be a useful generalization of stubborn sets. Section 4 considers the sleep set method and its combination with the stubborn set method. We conclude in Section 5 by summarizing the results obtained and briefly discussing possible directions for

future research.

2 Place/Transition Nets

In this section we give definitions of *place/transition nets* [15] that will be used in later sections.

We shall use "iff" to denote "if and only if". The *power set* (the set of subsets) of a set A is denoted by 2^A. The set of *(total) functions* from a set A to a set B is denoted by $(A \to B)$. The set of natural numbers, including 0, is denoted by N. We shall use ω to denote a formal infinite number, and N_ω to denote $N \cup \{\omega\}$. Relation $\leq$ over N is extended to N_ω by defining

$$\forall n \in N_\omega \; n \leq \omega.$$

Addition and subtraction are extended similarly by defining

$$\forall n \in N \; \omega + n = \omega \wedge \omega - n = \omega.$$

Clearly, $\omega \notin N$ since no natural number can be substituted for ω in these conditions in such a way that the conditions would hold.

Definition 1. A *place/transition net* is a 6-tuple $\langle S, T, F, K, W, M_0 \rangle$ such that

- S is the set of *places*,
- T is the set of *transitions*, $S \cap T = \emptyset$,
- F is the set of *arcs*, $F \subseteq (S \times T) \cup (T \times S)$,
- K is the *capacity function*, $K \in (S \to N_\omega)$,
- W is the *arc weight function*, $W \in (F \to (N \setminus \{0\}))$, and
- M_0 is the *initial marking (initial state)*, $M_0 \in \mathcal{M}$ where $\mathcal{M}$ is the set of *markings (states)*, $\mathcal{M} = \{M \in (S \to N) \mid \forall s \in S \; M(s) \leq K(s)\}$.

If $x \in S \cup T$, then the set of *input elements* of x is

$${}^\bullet x = \{y \mid \langle y, x \rangle \in F\},$$

the set of *output elements* of x is

$$x^\bullet = \{y \mid \langle x, y \rangle \in F\},$$

and the set of *adjacent elements* of x is $x^\bullet \cup {}^\bullet x$. The function W is extended to a function in $(((S \times T) \cup (T \times S)) \to N)$ by defining $W(x, y) = 0$ iff $\langle x, y \rangle \notin F$. The net is *finite* iff $S \cup T$ is finite. □

Unlike Reisig [15], we do not accept $M(s) = \omega$. Such markings would be redundant in finite place/transition nets.

Definition 2. Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net. A transition t is *enabled at a marking* M iff

$$\forall s \in {}^\bullet t \; M(s) \geq W(s, t)$$

and

$$\forall s \in t^\bullet \ M(s) - W(s,t) + W(t,s) \leq K(s).$$

A transition t *leads (can be fired) from a marking* M *to a marking* M' ($M[t\rangle M'$ for short) iff t is enabled at M and

$$\forall s \in S \ M'(s) = M(s) - W(s,t) + W(t,s).$$

A transition t is *disabled at a marking* M iff t is not enabled at M. A marking M is *terminal* iff no transition is enabled at M. A marking M is *nonterminal* iff M is not terminal. □

Our enabledness condition is weaker than Reisig's enabledness condition [15] that requires $M(s) + W(t,s) \leq K(s)$ instead of $M(s) - W(s,t) + W(t,s) \leq K(s)$.

Finite transition sequences and reachability are introduced in Definition 3. We shall use ε to denote the empty sequence.

Definition 3. Let $\langle S, T, F, K, W, M_0\rangle$ be a place/transition net. For any $T_s \subseteq T$,

$$\begin{aligned} T_s^0 &= \{\varepsilon\}, \\ (\forall n \in N \ T_s^{n+1} &= \{\sigma t \mid \sigma \in T_s^n \wedge t \in T_s\}), \text{ and} \\ T_s^* &= \{\sigma \mid \exists n \in N \ \sigma \in T_s^n\}. \end{aligned}$$

The set T_s^* is called the *set of finite sequences of transitions in* T_s, and the set T^* is called the *set of finite transition sequences of the net*. A finite transition sequence σ' is a *prefix* of a finite transition sequence σ iff there exists a finite transition sequence σ'' such that $\sigma = \sigma'\sigma''$. A finite transition sequence σ *leads (can be fired) from a marking* M *to a marking* M' iff $M[\sigma\rangle M'$ where

$$\begin{aligned} &\forall M \in \mathcal{M} \ M[\varepsilon\rangle M, \text{ and} \\ &\forall M \in \mathcal{M} \ \forall M' \in \mathcal{M} \ \forall \delta \in T^* \ \forall t \in T \\ &M[\delta t\rangle M' \Leftrightarrow (\exists M'' \in \mathcal{M} \ M[\delta\rangle M'' \wedge M''[t\rangle M'). \end{aligned}$$

A finite transition sequence σ is *enabled at a marking* M ($M[\sigma\rangle$ for short) iff σ leads from M to some marking. A finite transition sequence σ is *disabled at a marking* M iff σ is not enabled at M. A marking M' is *reachable from a marking* M iff some finite transition sequence leads from M to M'. A marking M' is a *reachable marking* iff M' is reachable from M_0. A marking M' is *globally unreachable* iff M' is not reachable from any other marking in $\mathcal{M}$ than M'. The *(full) reachability graph* of the net is the pair $\langle V, A\rangle$ such that the set of vertices V is the set of reachable markings, and the set of edges A is

$$\{\langle M, t, M'\rangle \mid M \in V \wedge M' \in V \wedge t \in T \wedge M[t\rangle M'\}. \ \square$$

A finite transition sequence is merely a string. It can be thought of as occurring as a path in the full reachability graph iff it is enabled at some reachable marking.

Definition 4. Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net. Let f be a function from $\mathcal{M}$ to 2^T. A finite transition sequence σ *f-leads (can be f-fired) from a marking M to a marking M'* iff $M[\sigma\rangle_f M'$, where

$$\forall M \in \mathcal{M}\ M[\varepsilon\rangle_f M, \text{ and}$$
$$\forall M \in \mathcal{M}\ \forall M' \in \mathcal{M}\ \forall \delta \in T^*\ \forall t \in T$$
$$M[\delta t\rangle_f M' \Leftrightarrow (\exists M'' \in \mathcal{M}\ M[\delta\rangle_f M'' \wedge t \in f(M) \wedge M''[t\rangle M').$$

A finite transition sequence σ is *f-enabled at a marking M* ($M[\sigma\rangle_f$ for short) iff σ f-leads from M to some marking. A marking M' is *f-reachable from a marking M* iff some finite transition sequence f-leads from M to M'. A marking M' is an *f-reachable marking* iff M' is f-reachable from M_0. The *f-reachability graph* of the net is the pair $\langle V, A \rangle$ such that the set of vertices V is the set of f-reachable markings, and the set of edges A is

$$\{\langle M, t, M' \rangle \mid M \in V \wedge M' \in V \wedge t \in f(M) \wedge M[t\rangle M'\}. \quad \square$$

Definition 4 is like a part of Definition 3 except that a transition selection function f determines which transitions are fired. If f is clear from the context or is implicitly assumed to exist and be of a kind that is clear from the context, then the f-reachability graph of the net is called the *reduced reachability graph* of the net. Note that the reduced reachability graph of the net can even be the full reachability graph of the net, e.g. in the case where $f(M) = T$ for each $M \in \mathcal{M}$.

Definition 5. Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net. A transition sequence δ is an *alternative sequence of a finite transition sequence σ at a marking M* iff δ is a finite transition sequence, σ is enabled at M, and δ leads from M to the same marking as σ. A transition sequence δ is a *length-secure alternative sequence of a finite transition sequence σ at a marking M* iff δ is an alternative sequence of σ at M and not longer than σ. The functions η and ϑ from $T^* \times \mathcal{M}$ to $2^{(T^*)}$ are defined as follows: for each finite transition sequence σ and marking M, $\eta(\sigma, M)$ is the *set of alternative sequences of σ at M*, and $\vartheta(\sigma, M)$ is the *set of length-secure alternative sequences of σ at M*. $\square$

Clearly, for each finite transition sequence σ and marking M, $\vartheta(\sigma, M) \subseteq \eta(\sigma, M)$. Also, $\eta(\sigma, M)$ is empty iff σ is not enabled at M.

Definition 6. Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net. A transition sequence δ is a *permutation of a finite transition sequence σ* iff δ is a finite transition sequence and for each transition t, the number of t's in δ is equal to the number of t's in σ. A transition sequence δ is an *enabled permutation of a finite transition sequence σ at a marking M* iff δ is a permutation of σ and enabled at M. The function π from $T^* \times \mathcal{M}$ to $2^{(T^*)}$ is defined as follows: for each finite transition sequence σ and marking M, $\pi(\sigma, M)$ is the set of enabled permutations of σ at M. $\square$

Clearly, if finite transition sequences are enabled permutations of each other at a marking M, they lead to the same marking from M. So, if a finite transition sequence σ is enabled at a marking M, then $\pi(\sigma, M) \subseteq \vartheta(\sigma, M)$. The set $\pi(\sigma, M)$ can be nonempty even if σ is not enabled at M since some permutation of σ can be enabled at M. The set of length-secure alternative sequences, as well as the set of

alternative sequences, of an enabled finite transition sequence σ at a marking can always be partitioned into sets of enabled permutations of sequences at the marking. Of course, only one of those sets is the set of enabled permutations of σ.

Figure 1 presents the functions η, ϑ, and π, in a nutshell.

$\eta(\sigma, M)$	the set of alternative sequences of a finite transition sequence σ at M
$\vartheta(\sigma, M)$	the set of length-secure alternative sequences of a finite transition sequence σ at M
$\pi(\sigma, M)$	the set of enabled permutations of a finite transition sequence σ at M

Fig. 1. The functions η, ϑ, and π.

Definition 7. Let $\langle S, T, F, K, W, M_0\rangle$ be a place/transition net. Let f be a function from $\mathcal{M}$ to 2^T. Then we say that f *represents all sets of alternative sequences to terminal markings* iff

$$\forall \sigma \in T^* \ \forall M \in \mathcal{M} \ (M[\sigma\rangle \wedge \forall t \in T \ \neg M[\sigma t\rangle) \Rightarrow (\exists \delta \in \eta(\sigma, M) \ M[\delta\rangle_f).$$

Correspondingly, f *represents all sets of length-secure alternative sequences to terminal markings* iff

$$\forall \sigma \in T^* \ \forall M \in \mathcal{M} \ (M[\sigma\rangle \wedge \forall t \in T \ \neg M[\sigma t\rangle) \Rightarrow (\exists \delta \in \vartheta(\sigma, M) \ M[\delta\rangle_f).$$

Respectively, f *represents all sets of enabled permutations to terminal markings* iff

$$\forall \sigma \in T^* \ \forall M \in \mathcal{M} \ (M[\sigma\rangle \wedge \forall t \in T \ \neg M[\sigma t\rangle) \Rightarrow (\exists \delta \in \pi(\sigma, M) \ M[\delta\rangle_f). \ \square$$

The following can clearly be seen from the above.

- A function representing all sets of enabled permutations to terminal markings represents all sets of length-secure alternative sequences to terminal markings.
- A function representing all sets of length-secure alternative sequences to terminal markings represents all sets of alternative sequences to terminal markings.

Definition 8. Let $\langle S, T, F, K, W, M_0\rangle$ be a place/transition net. Transitions t and t' *commute at a marking* M iff $M[tt'\rangle$ and $M[t't\rangle$. Transitions t and t' are *independent at a marking* M iff

$$(M[tt'\rangle \wedge M[t't\rangle) \vee ((\neg M[t\rangle) \wedge (\neg M[t'\rangle)) \vee$$
$$(M[t\rangle \wedge (\neg M[t'\rangle) \wedge (\neg M[tt'\rangle)) \vee (M[t'\rangle \wedge (\neg M[t\rangle) \wedge (\neg M[t't\rangle)). \ \square$$

Our definition of independence corresponds to Godefroid's and Pirottin's [7] definition of conditional independence which in turn is based on Katz's and Peled's [10] corresponding definition. Our definition of independence can be obtained from Godefroid's and Pirottin's definition of valid conditional dependency relations, Definition 5 in [7], by taking the necessary conditions for a triple of two transitions and one state to be in the complement of a valid dependency relation, and substituting terms of place/transition nets for the terms of the model of concurrency in [7] in an obvious way.

The following can clearly be seen from the above.

- Different transitions are independent at a marking iff neither of them can be fired at the marking making the other transition turn from enabled to disabled or from disabled to enabled.
- A transition t commutes with itself at a marking iff tt is enabled at the marking.
- A transition t is independent of itself at a marking iff tt is enabled or t is disabled at the marking.
- Transitions commute at a marking iff they are enabled and independent at the marking.

3 Dynamically Stubborn Sets

This section is concentrated on *dynamically stubborn* sets [14, 19]. All the stubborn sets that have been defined in the literature are known to be dynamically stubborn. Dynamically stubborn sets seem to have all the nice properties of (statically) stubborn sets except that the definition of dynamic stubbornness does not seem to imply a practical algorithm for computing dynamically stubborn sets. We define dynamic stubbornness on the basis of Rauhamaa's principles [14].

Definition 9. Let $\langle S, T, F, K, W, M_0\rangle$ be a place/transition net. Let M be a marking of the net. A set $T_s \subseteq T$ *fulfils the first principle of dynamic stubbornness* (*D1* for short) *at* M iff

$$\forall \sigma \in (T \setminus T_s)^*\ \forall t \in T_s\ M[\sigma t\rangle \Rightarrow M[t\sigma\rangle.$$

A transition t is a *key transition of a set* $T_s \subseteq T$ *at* M iff $t \in T_s$ and

$$\forall \sigma \in (T \setminus T_s)^*\ M[\sigma\rangle \Rightarrow M[\sigma t\rangle.$$

A set $T_s \subseteq T$ *fulfils the second principle of dynamic stubbornness* (*D2* for short) *at* M iff T_s has a key transition at M. A set $T_s \subseteq T$ *fulfils the first principle of strong dynamic stubbornness* (*SD1* for short) *at* M iff

$$\forall \sigma \in (T \setminus T_s)^*\ \forall t \in T_s\ M[\sigma t\rangle \Rightarrow M[t\rangle.$$

A set $T_s \subseteq T$ *fulfils the second principle of strong dynamic stubbornness* (*SD2* for short) *at* M iff

$$\forall \sigma \in (T \setminus T_s)^*\ \forall t \in T_s\ (M[t\rangle \wedge M[\sigma\rangle) \Rightarrow (M[\sigma t\rangle \wedge M[t\sigma\rangle).$$

A set $T_s \subseteq T$ is *dynamically stubborn at* M iff T_s fulfils D1 and D2 at M. A set $T_s \subseteq T$ is *strongly dynamically stubborn at* M iff T_s fulfils SD1 and SD2 at M and $\exists t \in T_s\ M[t\rangle$. □

The principles D1, D2, SD1, and SD2 are illustrated in Figure 2. The principles D1, D2, SD1, and SD2 are Rauhamaa's Principles 1*, 2*, 1, and 2, respectively [14]. Clearly, a key transition of a set at a marking is enabled at the marking. Our key transitions are similar to Valmari's key transitions [17]. The difference is that Valmari's key transitions satisfy a condition that can be checked easily and is sufficient but not necessary for a transition to be a key transition in the sense of our definition. As shown in [23], our strong dynamic stubbornness is weaker than Valmari's strong dynamic stubbornness [19].

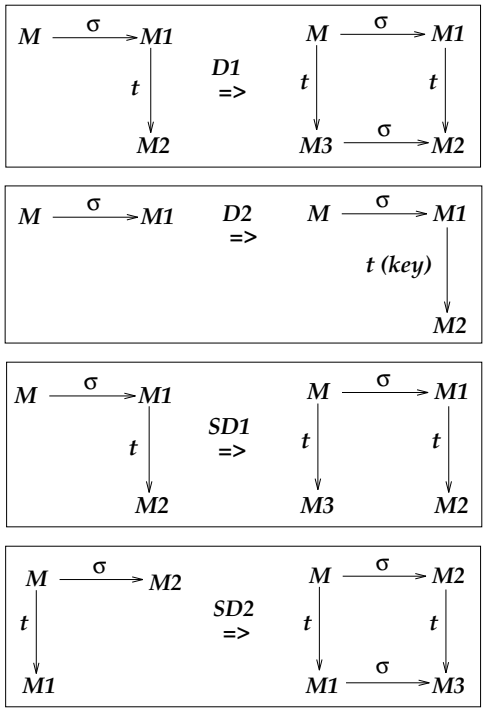

Fig. 2. The principles of dynamic and strong dynamic stubbornness.

Lemma 10. *A set is strongly dynamically stubborn at a marking iff the set is dynamically stubborn at the marking and each enabled transition in the set is a key transition of the set at the marking.*

Proof. The result follows trivially from Definition 9. □

The result in Lemma 10 is due to Valmari [17] but has missed explicit treatment. We shall soon see that dynamic stubbornness alone is sufficient as far as the detection of reachable terminal markings is concerned. While all stubborn sets are known to be dynamically stubborn, quite efficient stubborn set computation algorithms exist that can compute both strongly dynamically stubborn sets and such sets that are not strongly dynamically stubborn [16, 17, 23]. Strongly dynamically stubborn sets are useful when one wants to eliminate the *ignoring phenomenon* [17, 23]. A transition is ignored at a marking iff the transition is enabled at the marking but not fired at any marking that is reachable from the marking. The existence of ignored transitions is called the ignoring phenomenon.

Definition 11. Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net. Let f be a function from $\mathcal{M}$ to 2^T. Then we say that f is *dynamically stubborn* iff for each nonterminal marking M, $f(M)$ is dynamically stubborn. Correspondingly, f is *strongly dynamically stubborn* iff for each nonterminal marking M, $f(M)$ is strongly dynamically stubborn. □

Theorem 12. *Let $\langle S, T, F, K, W, M_0\rangle$ be a place/transition net. Let f be a dynamically stubborn function from $\mathcal{M}$ to 2^T. Then f represents all sets of enabled permutations to terminal markings.*

Proof. We show that

$$\forall \sigma'' \in T^*\ \forall M \in \mathcal{M}\ (M[\sigma''\rangle \wedge \forall t \in T\ \neg M[\sigma''t\rangle) \Rightarrow (\exists \delta \in \pi(\sigma'', M)\ M[\delta\rangle_f).$$

We use induction on the length of σ''. The claim holds trivially when restricted to $\sigma'' = \varepsilon$. Our induction hypothesis is that the claim holds when restricted to any σ'' of length $n \geq 0$. Let $\sigma \in T^*$, $M \in \mathcal{M}$, and $M' \in \mathcal{M}$ be such that σ is of length $n+1$, $M[\sigma\rangle M'$, and $\forall t \in T\ \neg M'[t\rangle$. The set $f(M)$ is dynamically stubborn at M since some transition is enabled at M. The sequence σ must contain a transition in $f(M)$ since otherwise some transition in $f(M)$ would be enabled at M' by D2. Let $\delta \in (T \setminus f(M))^*$, $t \in f(M)$, and $\delta' \in T^*$ be such that $\sigma = \delta t \delta'$. By D1 we have $M[t\delta\rangle$, so $M[t\delta\delta'\rangle$. Let $M'' \in \mathcal{M}$ be such that $M[t\rangle M''$. Now $M[t\rangle_f M''$. By the induction hypothesis, $\exists \sigma' \in \pi(\delta\delta', M'')\ M''[\sigma'\rangle_f$. We thus have $t\sigma' \in \pi(\sigma, M)$ and $M[t\sigma'\rangle_f$. □

The result in Theorem 12 is due to Valmari [16, 17] but has missed explicit treatment. Theorem 12 has the consequence that if a finite transition sequence leads from a marking M to a terminal marking, and M occurs in the reduced reachability graph, then an enabled permutation of the sequence occurs in the graph. A dynamically stubborn set selective search thus certainly finds all reachable terminal markings if the net and the set of reachable markings are finite. If the set of reachable markings is infinite but the net is finite and a dynamically stubborn set selective search is performed in a breadth-first order for some time, then reachable terminal markings "near the initial marking" can be found. As we shall see in Section 4, the permutation preserving property makes the stubborn set method compatible with the sleep set method in the detection of reachable terminal markings though a weaker property would suffice.

We define persistence and conditional stubbornness in such a way that the definitions correspond to the definitions given by Godefroid and Pirottin [7]. Our definitions can be obtained from Godefroid's and Pirottin's Definitions 7 and 8 in [7] by substituting terms of place/transition nets for the terms of the model of concurrency in [7] in an obvious way.

Definition 13. Let $\langle S, T, F, K, W, M_0\rangle$ be a place/transition net. Let $M \in \mathcal{M}$. A set $T_s \subseteq T$ *fulfils the principle of persistence and conditional stubbornness* (*PE* for short) *at* M iff

$$\forall \sigma \in (T \setminus T_s)^*\ \forall t \in T_s\ \forall t' \in T \setminus T_s\ \forall M' \in \mathcal{M}$$
$$(M[t\rangle \wedge M[\sigma\rangle M' \wedge M'[t'\rangle) \Rightarrow (t \text{ and } t' \text{ are independent at } M').$$

A set $T_s \subseteq T$ is *persistent at* M iff T_s fulfils PE at M and $\forall t \in T_s\ M[t\rangle$. A set $T_s \subseteq T$ is *conditionally stubborn at* M iff T_s fulfils SD1 and PE at M and $\exists t \in T_s\ M[t\rangle$. □

Clearly, the "$M[t\rangle\wedge$" in PE is redundant in the definition of persistence since all transitions in persistent sets are enabled. Our goal in the rest of this section is to reexpress persistence and conditional stubbornness in terms of dynamic stubbornness.

Lemma 14. *Let $\langle S, T, F, K, W, M_0\rangle$ be a place/transition net. Let M be a marking of the net. A set $T_s \subseteq T$ fulfils SD2 at M iff*

$$\forall \sigma \in (T \setminus T_s)^* \ \forall \delta \in (T \setminus T_s)^* \ \forall t \in T_s \ (M[t\rangle \wedge M[\sigma\delta\rangle) \Rightarrow M[\sigma t\delta\rangle.$$

Proof. The "if"-part is obvious. Let's prove the "only if" -part. Let a set $T_s \subseteq T$ fulfil SD2 at M. Let $\sigma \in (T \setminus T_s)^*$, $\delta \in (T \setminus T_s)^*$, $t \in T_s$, $M[t\rangle$, and $M[\sigma\delta\rangle$. Using SD2 for both $\sigma\delta$ and σ, we get $M[t\sigma\delta\rangle$ and $M[\sigma t\rangle$. As σt and $t\sigma$ lead to the same marking, we have $M[\sigma t\delta\rangle$. □

The result in Lemma 14 is due to Valmari [17] but has missed explicit treatment.

Lemma 15. *Let $\langle S, T, F, K, W, M_0\rangle$ be a place/transition net. Let M be a marking of the net, and T_s and T_e subsets of T such that*

$$\{t \in T_s \mid M[t\rangle\} \subseteq T_e, \ \text{and } T_e \subseteq T_s.$$

If T_s is dynamically stubborn at M, T_e is dynamically stubborn at M. If T_s is strongly dynamically stubborn at M, T_e is strongly dynamically stubborn at M.

Proof. (i) Let T_s be dynamically stubborn at M. We show that

$$\forall \sigma \in (T \setminus T_e)^* \ M[\sigma\rangle \Rightarrow \sigma \in (T \setminus T_s)^*.$$

Let $\sigma \in (T \setminus T_e)^*$ and $\delta \in (T \setminus T_s)^*$ be such that $M[\sigma\rangle$ and δ is the longest prefix of σ not containing any transition in T_s. If $\delta \neq \sigma$, the first transition after δ in σ is enabled at M by D1 for T_s. Since no transition in $T_s \setminus T_e$ is enabled at M, we conclude that $\delta = \sigma$, so $\sigma \in (T \setminus T_s)^*$.

(ii) Since strongly dynamically stubborn sets are dynamically stubborn by Lemma 10, the result of part (i) holds for them, too. Then D1 for T_s implies D1 for T_e, D2 for T_s implies D2 for T_e, SD1 for T_s implies SD1 for T_e, and SD2 for T_s, implies SD2 for T_e. □

The result in Lemma 15 is new though inspired by Godefroid and Pirottin [7]. Lemma 15 states that if we remove disabled transitions from a dynamically stubborn (strongly dynamically stubborn) set, the remaining set is dynamically stubborn (strongly dynamically stubborn). For example, if a dynamically stubborn set is minimal with respect to set inclusion, by Lemma 15 the set consists of enabled transitions only.

Lemma 16. *A set fulfils PE at a marking iff the set fulfils SD2 at the marking. A set is conditionally stubborn at a marking iff the set is strongly dynamically stubborn at the marking.*

Proof. We show that PE is equivalent to SD2. The second statement then follows directly from Definitions 9 and 13. Let $\langle S, T, F, K, W, M_0\rangle$ be a place/transition net. Let $M \in \mathcal{M}$ and $T_s \subseteq T$.

(i) We prove that SD2 implies PE. Let T_s fulfil SD2 at M. Let $\sigma \in (T \setminus T_s)^*$, $t \in T_s$, $t' \in T \setminus T_s$, $M' \in \mathcal{M}$, $M[t\rangle$, $M[\sigma\rangle M'$, and $M'[t'\rangle$. By Lemma 14 we have both $M'[t't\rangle$ and $M'[tt'\rangle$. The transitions t and t' are thus independent at M'.

(ii) We prove that PE implies SD2. Let T_s fulfil PE at M. We use induction on the length of finite transition sequences to show that T_s fulfils SD2 at M. The principle SD2 is fulfilled trivially when restricted to ε. Our induction hypothesis is that SD2 is fulfilled when restricted to finite transition sequences of length $n \geq 0$. We show that SD2 is then fulfilled when restricted to finite transition sequences of length $n+1$. Let $\delta \in (T \setminus T_s)^*$, $t' \in T \setminus T_s$, and $t \in T_s$ be such that $M[t\rangle$, $M[\delta t'\rangle$, and δ is of length n. Let $M' \in \mathcal{M}$ be such that $M[\delta\rangle M'$. The transition t' is then enabled at M'. By the induction hypothesis we have $M[\delta t\rangle$ and $M[t\delta\rangle$. The transition t is thus enabled at M'. The principle PE then implies that t and t' are independent at M'. Transitions commute at a marking iff they are enabled and independent at the marking. So t and t' commute at M'. Thus $M'[tt'\rangle$ and $M'[t't\rangle$, and consequently $M[\delta tt'\rangle$ and $M[\delta t't\rangle$. As already mentioned, we have $M[t\delta\rangle$, so $t\delta$ leads from M to the same marking as δt. We thus have $M[t\delta t'\rangle$. □

The result in Lemma 16 is new.

Lemma 17. *A set is a nonempty persistent set at a marking iff the set is a conditionally stubborn set at the marking and does not contain any transition that is disabled at the marking. The set of enabled transitions of any conditionally stubborn set is a nonempty persistent set.*

Proof. The first statement follows from the fact that a persistent set fulfils SD1 trivially since all its transitions are enabled. The second statement follows trivially from the first statement and Lemmata 15 and 16. □

The result in Lemma 17 is due to Godefroid and Pirottin [7] but the proof is new.

4 Sleep Set Method

In this section we present Godefroid's *sleep set method* [4, 5, 6, 7, 8, 24, 25]. The plain sleep set method preserves at least one sequence from each *conditional trace* [7, 10, 23] leading from the initial state to a terminal state. To prevent a transition from firing, it is put into a so called sleep set.

Wolper and Godefroid [24], Godefroid and Pirottin [7], and Wolper, Godefroid, and Pirottin [25] have combined the sleep set method with the stubborn set method. The combination is justified by the fact that the stubborn set method alone is sometimes bound to fire independent transitions at a state. The combination presented in [7, 24, 25] is such that at each encountered nonterminal state a nonempty persistent set is computed. Let us recall from Lemmata 16 and 17 that a set is a nonempty persistent set iff the set is a strongly dynamically stubborn set consisting of enabled transitions only. In [24, 25], persistence is defined on the basis of global independence but since global independence implies independence at each reachable state, the persistent sets in [24, 25] are persistent in the sense defined by Godefroid and Pirottin [7]. As mentioned immediately above Definition 13, our definition of persistence in Definition 13 corresponds to Godefroid's and Pirottin's definition [7]. The plain sleep set method can be thought of as a special case of the combined method: a simple heuristic for computing a persistent set is used. We shall not consider the plain sleep set method further.

We concentrate on a generalized version of Wolper's and Godefroid's terminal state detection algorithm [24]. The generalized version is in Figure 3. The intuitive idea of the algorithm is to eliminate such redundant *interleavings of transitions* that are not eliminated by the transition selection function f. We show that the algorithm is guaranteed to find all reachable terminal markings of any finite place/transition net with a finite set of reachable markings. From the finiteness of the net, from the finiteness of the set of reachable markings, and from the fact that no transition is fired twice at one marking it follows that the execution of the algorithm takes a finite time only. Any dynamically stubborn function is valid for f, but the algorithm is not limited to dynamically stubborn sets. It suffices that f represents all sets of length-secure alternative sequences to terminal markings. The set T_0 can be any subset of transitions that are disabled at the initial marking. The φ in Figure 3 can be any truth-valued function on $\mathcal{M} \times T \times T \times 2^T$ that satisfies: if $\varphi(M, t, t', T_s)$, then either t and t' commute at M and $t' \in T_s$, or tt' is disabled at M. For example, $\varphi(M, t, t', T_s)$ could be

- "either t and t' commute at M and $t' \in T_s$, or tt' is disabled at M",
- "t and t' are independent at M and $t' \in T_s$",
- "t and t' commute at M and $t' \in T_s$", or simply
- "false".

Note that if $M[t\rangle M'$, then tt' is disabled at M iff t' is disabled at M'. So the first alternative in the above list has the effect that if t is fired from M to M' in the algorithm in Figure 3, then the sleep set pushed onto the stack with M' contains all those transitions that are disabled at M'. We shall consider the practicalities related to φ and T_0 later in this section.

The algorithm in Figure 3 is similar to Wolper's and Godefroid's algorithm [24]. The only essential differences are that Wolper and Godefroid assume that the set corresponding to $f(M)$ is persistent, the set corresponding to T_0 is empty, and the condition corresponding to $\varphi(M, t, t', T_s)$ is "t and t' are globally independent and $t' \in T_s$".

Theorem 18. *Let $\langle S, T, F, K, W, M_0\rangle$ be a finite place/transition net such that the set of markings reachable from M_0 is finite. Let f be a function from $\mathcal{M}$ to 2^T such that f represents all sets of length-secure alternative sequences to terminal markings. Let T_0 be a subset of transitions that are disabled at M_0. Let φ be a truth-valued function on $\mathcal{M} \times T \times T \times 2^T$ such that for each marking M, for all transitions t and t', and for each $T_s \subseteq T$, if $\varphi(M, t, t', T_s)$, then either t and t' commute at M and $t' \in T_s$, or tt' is disabled at M. Then the algorithm in Figure 3 finds all terminal markings that are reachable from M_0.*

Proof. Let M_d be a terminal marking that is reachable from M_0.

(i) We first prove that if $X \subseteq T$, a finite transition sequence σ leads from a marking M to M_d, and for each δ in $\vartheta(\sigma, M)$, the first transition of δ is not in X, then, if $\langle M, X\rangle$ is pushed onto the stack, some element having M_d as the first component will be or has already been popped from the stack.

The proof proceeds by induction on the length of σ. For $\sigma = \varepsilon$, the result is immediate. Now, assume the proposition holds for finite transition sequences of length less than or equal to n, where $n \geq 0$, and let us prove that it holds for a

```
make Stack empty; make H empty;
push ⟨M_0, T_0⟩ onto Stack;
while Stack is not empty do {
    pop ⟨M, Sleep⟩ from Stack;
    if M is not in H then {
        Fire = {t ∈ f(M) \ Sleep | M[t⟩};
        if Fire and {t ∈ Sleep | M[t⟩} are both empty then print "Terminal state!";
        enter ⟨M, a copy of Sleep⟩ in H;
    }
    else {
        let hSleep be the set associated with M in H;
        Fire = {t ∈ hSleep \ Sleep | M[t⟩};
        Sleep = hSleep ∩ Sleep;
        substitute a copy of Sleep for the set associated with M in H;
    }
    for each t in Fire do {
        let M[t⟩M';
        tSleep = {t' ∈ T | φ(M, t, t', Sleep)};
        push ⟨M', a copy of tSleep⟩ onto Stack;
        Sleep = {t} ∪ Sleep;
    }
}
```

Fig. 3. A terminal marking detection algorithm.

finite transition sequence σ of length $n+1$. Let X be a subset of T, σ lead from a a marking M to M_d, and $\langle M, X\rangle$ have been pushed onto the stack. Let it also be the case that for each δ in $\vartheta(\sigma, M)$, the first transition of δ is not in X. Let us consider the actions immediately following the popping of $\langle M, X\rangle$ from the stack.

We first consider the case where M is not already in H. Since M is a non-terminal marking and f represents all sets of length-secure alternative sequences to terminal markings, at least one transition in $f(M)$ is the first transition of some sequence in $\vartheta(\sigma, M)$. Moreover, every such transition is in $T \setminus X$ and is thus fired at M. Let t_1 be the first of such transitions in the firing order. Then there exists a finite transition sequence σ' such that $t_1\sigma'$ is in $\vartheta(\sigma, M)$. From the definition of ϑ it follows that $M[t_1\sigma'\rangle M_d$ and $t_1\sigma'$ is not longer than σ. The length of σ' is thus less than or equal to n. Let t_1 lead from M to a marking M'. Then $M'[\sigma'\rangle M_d$. Let $\langle M', X'\rangle$ be pushed onto the stack when firing t_1 at M. We show that for each δ in $\vartheta(\sigma', M')$, the first transition of δ is not in X'.

Indeed, assume the opposite, i.e., there exists some transition t' in X' such that for some finite transition sequence δ', $t'\delta'$ is in $\vartheta(\sigma', M')$. Clearly, then $t_1t'\delta'$ is in $\vartheta(\sigma, M)$. From the condition satisfied by φ it follows that t_1 and t' commute at M and t' is in Sleep at the time of pushing $\langle M', X'\rangle$ onto the stack. The transition t' cannot be equal to t_1 since t_1 is not in Sleep at the time of the pushing. Since t_1 and t' commute at M, t_1t' and $t't_1$ lead to the same marking from M. Consequently, $t't_1\delta'$ is in $\vartheta(\sigma, M)$. From the condition satisfied by X it thus follows that t' is not

in X. There is then only the possibility that t' has been inserted into Sleep in the "for-loop" before firing t_1. We have obtained a contradiction since if a transition in $f(M) \setminus X$ is the first transition of some sequence in $\vartheta(\sigma, M)$, the transition is either t_1 itself or fired after t_1. The inductive hypothesis can thus be used to establish that some element having M_d as the first component will be or has already been popped from the stack.

We now consider the case where M already appears in H. Let $Y \subseteq T$ be such that $\langle M, Y \rangle$ is in H. All those transitions in $Y \setminus X$ that are enabled at s are fired. There are two situations: either some transition in Y is the first transition of some sequence in $\vartheta(\sigma, M)$, or no such transition exists. In the first situation, we can choose a transition analogous to the above t_1 and proceed as above.

Let us now turn to the second situation in which no transition in Y is the first transition of any sequence in $\vartheta(\sigma, M)$. This can be the case either because no transition in Y_0 is the first transition of any sequence in $\vartheta(\sigma, M)$ where Y_0 is the sleep set entered in H with M when M was inserted into H, or because there are some Y' and Z such that $\langle M, Z \rangle$ was popped from the stack before popping $\langle M, X \rangle$ from the stack, $\langle M, Y' \rangle$ was in H at the time of the popping of $\langle M, Z \rangle$ from the stack, some transition in Y' is the first transition of some sequence in $\vartheta(\sigma, M)$, and no transition in $Y' \cap Z$ is the first transition of any sequence in $\vartheta(\sigma, M)$. In the former case, we can proceed as above with Y_0 in the place of X. In the latter case, we can proceed as above with Z in the place of X, taking into account the fact that Sleep$= Y' \cap Z$ when the "for-loop" is entered.

(ii) The algorithm in Figure 3 starts by pushing $\langle M_0, T_0 \rangle$ onto an empty stack. Every transition in T_0 is disabled at M_0. From the result shown in part (i) it thus follows that some element having M_d as the first component will be popped from the stack. □

The result in Theorem 18 is new though inspired by Wolper and Godefroid [24], Godefroid and Pirottin [7], and Wolper, Godefroid, and Pirottin [25]. Let us recall from Theorem 12 that dynamically stubborn functions represent all sets of enabled permutations to terminal markings. So they represent all sets of length-secure alternative sequences to terminal markings, too. From Theorem 18 it thus follows that the algorithm in Figure 3 is compatible with all dynamically stubborn sets.

Let's consider an example which shows that the statement obtained from Theorem 18 by removing the word "length-secure" is not valid. Let $\varphi(M, t, t', T_s)$ iff t and t' commute at M and $t' \in T_s$. Let $T_0 = \emptyset$. Let $M_0[a\rangle M_1$, $M_1[d\rangle M_2$, and $M_2[bc\rangle M_3$ in the net in Figure 4. Let f be defined by $f(M_0) = \{a\}$ and $f(M) = T$ when $M \neq M_0$. For each $M \in \mathcal{M}$, if M_0 is reachable from M, then $M = M_0$ or $M = M_1$. The marking M_3 is the only terminal marking that is reachable from M_0. The function f thus represents all sets of alternative sequences to terminal markings. However, f does not represent all sets of length-secure alternative sequences to terminal markings since $M_0[cd\rangle M_3$ but for each $\sigma \in T^*$ of length less than or equal to 2, $\neg M_0[\sigma\rangle_f M_3$.

During the first visit to M_0, the algorithm in Figure 3 inserts $\langle M_0, \emptyset \rangle$ into H and pushes $\langle M_1, \emptyset \rangle$ onto the stack. The algorithm then visits M_1. The transitions b and d are the enabled transitions in $f(M_1) = T$ at M_1. Let b be fired before d at M_1. The algorithm pushes $\langle M_0, \emptyset \rangle$ and $\langle M_2, \{b\} \rangle$ onto the stack since b and d commute at M_1. The algorithm then visits M_2 but does not fire the sleeping b which is the

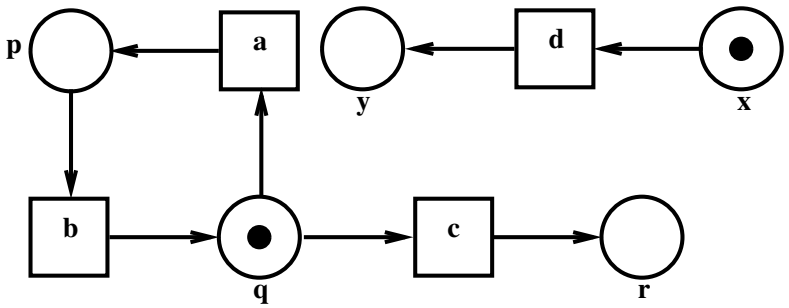

Fig. 4. A deceptive f makes the algorithm in Figure 3 fail.

only enabled transition at M_2. No transition is fired during the second visit to M_0 since the sleep set associated with M_0 in H is empty. The execution of the algorithm is then over. No terminal marking was found though M_3 is a terminal marking that is reachable from M_0.

Lemma 19. *Let $\langle S, T, F, K, W, M_0\rangle$ be a finite place/transition net. Let φ be a truth-valued function on $\mathcal{M} \times T \times T \times 2^T$ such that for each marking M, for all transitions t and t', and for each $T_s \subseteq T$, if $\varphi(M, t, t', T_s)$, then t and t' are independent at M and $t' \in T_s$. Let $T_0 = \emptyset$. Then in the algorithm in Figure 3, each sleep set associated with a marking contains only transitions that are enabled at the marking.*

Proof. If a transition t' is in the sleep set pushed onto the stack with a marking M' when a transition t is fired from a marking M to M', then $t' \in$ Sleep at the time of the push, and each transition in Fire is enabled at M. The sleep set associated with the initial marking at the beginning of the execution of the algorithm is empty. If transitions t and t'' are enabled and independent at a marking M, and $M[t\rangle M''$, then t'' is enabled at M''. The result thus follows by a trivial induction. □

The result in Lemma 19 is due to Wolper and Godefroid [24] despite the differences between the algorithm in Figure 3 and their terminal state detection algorithm.

In Figure 5, an implementation of the algorithm in Figure 3 with respect to φ and T_0 is presented. The algorithm in Figure 5 can be obtained from the algorithm in Figure 3 by making T_0 empty, removing the checking of enabledness of transitions in sleep sets, and defining: $\varphi(M, t, t', T_s)$ iff t and t' commute at M and $t' \in T_s$. We know that transitions commute at a marking iff they are enabled and independent at the marking. Checking commutation should be easier than checking independence. Lemma 19 implies that the algorithm in Figure 3 is equivalent to the algorithm in Figure 5 when T_0 is empty and φ is defined: $\varphi(M, t, t', T_s)$ iff t and t' commute at M and $t' \in T_s$. Lemma 5 thus also implies that in the algorithm in Figure 5, each sleep set associated with a marking contains only transitions that are enabled at the marking.

Lemma 20. *Let $\langle S, T, F, K, W, M_0\rangle$ be a finite place/transition net such that the set of markings reachable from M_0 is finite. Let T_0 be a subset of transitions that are disabled at M_0. Let φ be a truth-valued function on $\mathcal{M} \times T \times T \times 2^T$ such that for*

```
make Stack empty; make H empty;
push ⟨M0, ∅⟩ onto Stack;
while Stack is not empty do {
    pop ⟨M, Sleep⟩ from Stack;
    if M is not in H then {
        Fire = {t ∈ f(M) \ Sleep | M[t⟩};
        if Fire and Sleep are both empty then print "Terminal state!";
        enter ⟨M, a copy of Sleep⟩ in H;
    }
    else {
        let hSleep be the set associated with M in H;
        Fire = hSleep \ Sleep;
        Sleep = hSleep ∩ Sleep;
        substitute a copy of Sleep for the set associated with M in H;
    }
    for each t in Fire do {
        let M[t⟩M';
        tSleep = {t' ∈ Sleep | t and t' commute at M};
        push ⟨M', a copy of tSleep⟩ onto Stack;
        Sleep = {t} ∪ Sleep;
    }
}
```

Fig. 5. A practical implementation of the algorithm in Figure 3 with respect to φ and T_0.

each marking M, for all transitions t and t', and for each $T_s \subseteq T$, if $\varphi(M, t, t', T_s)$, then either t and t' commute at M and $t' \in T_s$, or tt' is disabled at M. Let's further require that for each marking M, for all transitions t and t', and for each $T_s \subseteq T$, if t and t' commute at M and $t' \in T_s$, then $\varphi(M, t, t', T_s)$. Let's assume that the sets, the set operations (insertion, union, intersection, and difference), the stack, the stack operations, the "for-loop", and the computation of $f(M)$ in the algorithms in Figure 3 and 5 are implemented exactly in the same way. Then the algorithms visit exactly the same markings and fire exactly the same transitions in exactly the same order.

Proof. Let's assume that a transition t is being fired from a marking M to a marking M' in the algorithm in Figure 3. If a transition t' is enabled at M' and is in the sleep set pushed onto the stack with M' when t is fired at M, then $\varphi(M, t, t', \text{Sleep})$ holds at the time of the push but tt' is enabled at M, so t and t' commute at M, $t' \in$ Sleep at the time of the push, and t' is enabled at M. The result now follows by a trivial induction. □

The result in Lemma 20 is new though inspired by Wolper and Godefroid [24].

Lemma 20 states that there is no more refined implementation of the algorithm in Figure 3 with respect to φ and T_0 than the algorithm in Figure 5 if φ and T_0 are required to satisfy the assumptions in Theorem 18. Lemmata 19 and 20 suggest the heuristic that each sleep set associated with a marking should only contain transitions that are enabled at the marking.

Let's consider the complexity of the algorithm in Figure 5. The time taken by a check of whether two transitions commute at a marking is at most proportional to ν, where ν is the maximum number of adjacent places of a transition. The cumulative time per marking spent in the "for-loop" is at most proportional to $\nu\rho^2$, where ρ is the maximum number of enabled transitions of a marking, and all visits to the marking are counted. This is based on the fact that each sleep set associated with a marking contains only transitions that are enabled at the marking. The time per visit to a marking spent in the operations related to H is the time of the search for the marking plus a time that is at most proportional to ρ. The searches in H are something that cannot be avoided easily whether or not we use sleep sets at all. It depends much on the net how many times a marking is visited and how many simultaneous occurrences of a marking there are in the stack. One stack element requires space for the marking and at most ρ transitions. It is not necessary to store copies of markings and transitions since pointers suffice. More clever ways to cut down on space consumption in sleep set algorithms have been presented by Godefroid, Holzmann, and Pirottin [5].

The combination of the sleep set method and the stubborn set method can really be better than the plain stubborn set method as far as the number of inspected markings is concerned. More precisely, there can be a dynamically stubborn function f such that $f(M)$ can be computed by using a feasible algorithm such as the *incremental algorithm* [16], and for each dynamically stubborn function g, the number of vertices in the g-reachability graph is greater than the number of markings that are inspected by the algorithm in Figure 5 that uses f. The net in Figure 6 is a simple example showing this. The example is essentially the same as can be found in [25].

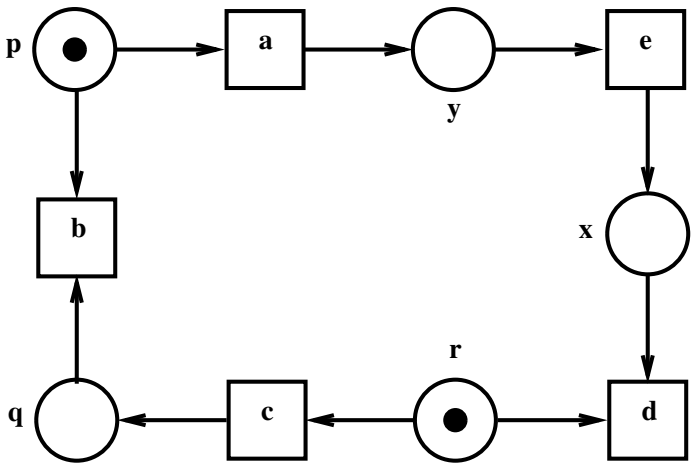

Fig. 6. A net showing some of the power of the algorithm in Figure 5.

An exhaustive investigation shows that at each reachable nonterminal marking of this net, there is one and only one dynamically stubborn set that is minimal with respect to set inclusion. By Lemma 15 we know that a dynamically stubborn set that

is minimal with respect to set inclusion only contains enabled transitions. Another exhaustive investigation shows that at each reachable nonterminal marking of this net, the set of enabled transitions of any stubborn set computed by the incremental algorithm, using any of the definitions of stubbornness in [16, 17, 19], is a dynamically stubborn set that is minimal with respect to set inclusion. If M is a nonterminal marking, let $f(M)$ be the set of enabled transitions of a stubborn set computed by the incremental algorithm. Then $f(M)$ is the only dynamically stubborn set at M that is minimal with respect to set inclusion. Thus, for each dynamically stubborn function g, the f-reachability graph is a subgraph of the g-reachability graph.

We have $f(M_0) = \{a, c\}$. Let a be fired before c at M_0 in the algorithm in Figure 5. Let $M_0[c\rangle M'$. Since a and c commute at M_0 and a is fired before c, $\langle M', \{a\}\rangle$ is pushed onto the stack. Let's consider the visit to M' where $\langle M', \{a\}\rangle$ is popped from the stack. We have $f(M') = \{a, b\}$, but the sleeping a is not fired. Let $M'[a\rangle M''$. By executing the algorithm in Figure 5 completely, we see that M'' is never encountered, and no nonterminal marking is visited more than once. The latter observation is important since it guarantees that all transitions that are fired at a marking M are in $f(M)$. The set of inspected markings is thus a proper subset of the markings of the f-reachability graph.

The statistics in [7, 24, 25] concerning some analyzed protocols do not give direct information for comparing the stubborn set method with the combination of the sleep set method and the stubborn set method.

5 Conclusions

The stubborn set method alone is sometimes bound to fire independent transitions at a state, so the sleep set method can further be used to eliminate redundant interleavings of transitions. We have generalized Wolper's and Godefroid's terminal state detection algorithm [24] and shown that the generalized version detects all reachable terminal markings of any finite place/transition net with a finite full reachability graph, given that the transition selection function represents all sets of length-secure alternative sequences to terminal markings. As already known, dynamically stubborn functions represent all sets of enabled permutations to terminal markings. They thus also represent all sets of length-secure alternative sequences to terminal markings.

Wolper and Godefroid [24], and Wolper, Godefroid, and Pirottin [25] suggest that the stubborn set method and the sleep set are compatible in a broad area of verification. The compatibility should certainly be studied further since all available means should be utilized in attacking the state space explosion problem, and in our opinion, only the detection of reachable terminal states has obtained more than cursory treatment so far. Linear temporal logics seem to form the most central area of research since they have a great expressive power, and the stubborn set method alone as well as the sleep set method alone can be extended to verify properties expressed as linear temporal logic formulae without a next state -operator [13, 18, 20, 21, 24]. The combination of the stubborn set method and the sleep set method should be studied in all those models of concurrency where each of these two methods alone are applicable. Finally, we have the problem of how efficient the algorithms are in practice and what could be done to improve their efficiency.

Acknowledgements

This work has been carried out in the Digital Systems Laboratory of Helsinki University of Technology. This paper is essentially an abridged version of a part of the research report [23]. I am grateful to Professor Leo Ojala for his continuous support, and Acting Associate Professor Mikko Tiusanen and Associate Professor Antti Valmari for their helpful comments. In addition, I would like to thank Lic.Tech. Marko Rauhamaa for fruitful discussions on the subject of this research.

The financial support received from the Emil Aaltonen Foundation is also gratefully acknowledged.

References

1. Ashcroft, E. and Manna, Z.: *Formalization of Properties of Parallel Programs.* Meltzer, B. and Michie, D. (Eds.), Machine Intelligence 6. Edinburgh University Press, Edinburgh 1971, pp. 17–42.
2. Best, E. and Lengauer, C.: *Semantic Independence.* Arbeitspapiere der GMD 250, Sankt Augustin 1987, 32 p.
3. Courcoubetis, C. (Ed.): Proceedings of the 5th International Conference on Computer-Aided Verification, Elounda, Greece, June/July 1993. Lecture Notes in Computer Science 697, Springer-Verlag, Berlin 1993, 504 p.
4. Godefroid, P.: *Using Partial Orders to Improve Automatic Verification Methods.* Clarke, E.M. and Kurshan, R.P. (Eds.), Proceedings of the 2nd International Workshop on Computer-Aided Verification, New Brunswick NJ, June 1990. Lecture Notes in Computer Science 531, Springer-Verlag, Berlin 1991, pp. 176–185.
5. Godefroid, P., Holzmann, G.J., and Pirottin, D.: *State Space Caching Revisited.* von Bochmann, G. and Probst, D.K. (Eds.), Proceedings of the 4th International Workshop on Computer-Aided Verification, Montreal, June 1992. Lecture Notes in Computer Science 663, Springer-Verlag, Berlin 1993, pp. 178–191.
6. Godefroid, P. and Kabanza, F.: *An Efficient Reactive Planner for Synthesizing Reactive Plans.* Proceedings of AAAI-91, Anaheim CA, July 1991, Vol. 2, pp. 640–645.
7. Godefroid, P. and Pirottin, D.: *Refining Dependencies Improves Partial-Order Verification Methods.* In [3], pp. 438–449.
8. Godefroid, P. and Wolper, P.: *Using Partial Orders for the Efficient Verification of Deadlock Freedom and Safety Properties.* Formal Methods in System Design 2 (1993) 2, pp. 149–164.
9. Katz, S. and Peled, D.: *Verification of Distributed Programs Using Representative Interleaving Sequences.* Distributed Computing 6 (1992) 2, pp. 107–120.
10. Katz, S. and Peled, D.: *Defining Conditional Independence Using Collapses.* Theoretical Computer Science 101 (1992) 2, pp. 337–359.
11. Mazurkiewicz, A.: *Trace Theory.* Brauer, W., Reisig, W., and Rozenberg, G. (Eds.), *Petri Nets: Applications and Relationships to Other Models of Concurrency. Advances in Petri Nets 1986, Part II, Proceedings of an Advanced Course, Bad Honnef, September 1986.* Lecture Notes in Computer Science 255, Springer-Verlag, Berlin 1987, pp. 279–324.
12. Overman, W.T.: *Verification of Concurrent Systems: Function and Timing.* PhD thesis, University of California Los Angeles, Los Angeles CA 1981, 174 p.
13. Peled, D.: *All from One, One for All: on Model Checking Using Representatives.* In [3], pp. 409–423.

14. Rauhamaa, M.: *A Comparative Study of Methods for Efficient Reachability Analysis.* Helsinki University of Technology, Digital Systems Laboratory Report A 14, Espoo, September 1990, 61 p.
15. Reisig, W.: *Petri Nets: An Introduction.* EATCS Monographs on Theoretical Computer Science 4, Springer-Verlag, Berlin 1985, 161 p.
16. Valmari, A.: *State Space Generation: Efficiency and Practicality.* Doctoral thesis, Tampere University of Technology Publications 55, Tampere 1988, 170 p.
17. Valmari, A.: *Stubborn Sets for Reduced State Space Generation.* Rozenberg, G. (Ed.), Advances in Petri Nets 1990. Lecture Notes in Computer Science 483, Springer-Verlag, Berlin 1991, pp. 491–515.
18. Valmari, A.: *A Stubborn Attack on State Explosion.* Formal Methods in System Design 1 (1992) 4, pp. 297–322.
19. Valmari, A.: *Stubborn Sets of Coloured Petri Nets.* Proceedings of the 12th International Conference on Application and Theory of Petri Nets, Gjern, Denmark, June 1991, pp. 102–121.
20. Valmari, A.: *Alleviating State Explosion during Verification of Behavioural Equivalence.* University of Helsinki, Department of Computer Science, Report A-1992-4, Helsinki 1992, 57 p.
21. Valmari, A.: *On-the-Fly Verification with Stubborn Sets.* In [3], pp. 397–408.
22. Valmari, A. and Tiusanen, M.: *A Graph Model for Efficient Reachability Analysis of Description Languages.* Proceedings of the 8th European Workshop on Application and Theory of Petri Nets, Zaragoza, June 1987, pp. 349–366.
23. Varpaaniemi, K.: *Efficient Detection of Deadlocks in Petri Nets.* Helsinki University of Technology, Digital Systems Laboratory Report A 26, Espoo, October 1993, 56 p.
24. Wolper, P. and Godefroid, P.: *Partial-Order Methods for Temporal Verification.* Best, E. (Ed.), Proceedings of the 4th International Conference on Concurrency Theory, Hildesheim, August 1993. Lecture Notes in Computer Science 715, Springer-Verlag, Berlin 1993, pp. 233–246.
25. Wolper, P., Godefroid, P., and Pirottin, D.: *A Tutorial on Partial-Order Methods for the Verification of Concurrent Systems.* Tutorial material of the 5th International Conference on Computer-Aided Verification, Elounda, Greece, June/July 1993, 85 p.