HELSINKI UNIVERSITY OF TECHNOLOGY
DIGITAL SYSTEMS LABORATORY
Series **B:** Technical Reports
No. **12**; April 1994

ISSN 0783–540X
ISBN 951–22–2126–8

ON COMPUTING SYMMETRIES AND STUBBORN SETS

Kimmo Varpaaniemi

Digital Systems Laboratory
Department of Computer Science
Helsinki University of Technology
Otaniemi, FINLAND

Helsinki University of Technology
Department of Computer Science
Digital Systems Laboratory
Otaniemi, Otakaari 1
FIN–02150 ESPOO, FINLAND

HELSINKI UNIVERSITY OF TECHNOLOGY
DIGITAL SYSTEMS LABORATORY
Series **B:** Technical Reports
No. **12**; April 1994

ISSN 0783–540X
ISBN 951–22–2126–8

On Computing Symmetries and Stubborn Sets

Kimmo Varpaaniemi

Abstract: In this report, we consider two promising methods for alleviating the state space explosion problem in reachability analysis: the *symmetry method* and the *stubborn set method*. We concentrate on the computation of symmetries and stubborn sets. The new algorithms presented in this report help us to save time and space in the symmetry method and to avoid inspecting redundant states in the so called *CFFD-preserving stubborn set method*.

Keywords: reachability analysis, symmetry method, stubborn set method, CFFD-semantics, place/transition nets

Printing: TKK Monistamo; Otaniemi 1994

Helsinki University of Technology
Department of Computer Science
Digital Systems Laboratory
Otaniemi, Otakaari 1
FIN–02150 ESPOO, FINLAND

Phone: $\frac{90}{+358\text{–}0}$ 4511
Telex: 125 161 htkk fi
Telefax: +358–0–465 077
E-mail: lab@saturn.hut.fi

Contents

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, many properties can often be verified without inspecting all reachable states of the system.

In this report, we consider two promising methods for alleviating the state space explosion problem: the *symmetry method* [1, 4, 5] and the *stubborn set method* [6, 7, 8]. We concentrate on the computation of symmetries and stubborn sets. We use *place/transition nets* [3] as our formalism. This is a tolerable restriction since many models of concurrency can be transformed into behaviourally equivalent place/transition nets.

The first contribution of this report can be described as follows. The algorithm of Schmidt [4] computes all symmetries of a net, i.e. all such bijections from places and transitions to places and transitions that "respect places, transitions, and arcs". In practice, *place symmetries,* i.e. the place mappings corresponding to symmetries, often suffice. We give an algorithm for computing the place symmetries. The algorithm consumes at most as much time and space as the algorithm of Schmidt [4]. Despite the similarity between our algorithm and the algorithm of Schmidt, we think that our algorithm deserves presentation since the proofs given by Schmidt [4] do not trivially imply the correctness of our algorithm.

The second contribution of this report is a new algorithm for computing stubborn sets for the so called *CFFD-preserving stubborn set method* [8]. The new algorithm computes stubborn sets that are minimal with respect to enabled transitions and certain other conditions. Such minimality is good since the number of directions inspected usually affects the number of states inspected during the verification of a given property. The algorithm has been derived from the algorithms presented by Valmari [6, 8], but as above, we think that the algorithm deserves presentation since the performed derivation is not quite trivial.

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. We consider the computation of symmetries in Section 3, and the CFFD-preserving stubborn set method in Section 4. We then end in conclusions in Section 5.

2 Place/Transition Nets

In this section we give definitions of *place/transition nets* [3] 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 *partial functions* from a set A to a set B is

$$\{R \subseteq A \times B \mid \forall x \in A\ \forall y \in B\ \forall z \in B\ (\langle x,y\rangle \in R \wedge \langle x,z\rangle \in R) \Rightarrow y = z\}.$$

The set of *total relations* from a set A to a set B is

$$\{R \subseteq A \times B \mid \forall x \in A\ \exists y \in B\ \langle x,y\rangle \in R\}.$$

The set of *functions* from a set A to a set B, denoted by $(A \rightarrow B)$, is the set of those partial functions from A to B that are total relations from A to B. So there is always an empty function from an empty set to any set but there is no function from a nonempty set to an empty set. 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 2.1 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*, $K \in (S \rightarrow N_\omega)$,
- W is the *arc weight function*, $W \in (F \rightarrow (N \setminus \{0\}))$, and
- M_0 is the *initial marking (initial state)*, $M_0 \in \mathcal{M}$ where $\mathcal{M}$ is the set of *markings (states)*, $\mathcal{M} = \{M \in (S \rightarrow N) \mid \forall s \in S\ M(s) \leq K(s)\}$.

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 $(((S \times T) \cup (T \times S)) \rightarrow N)$ by defining $W(x,y) = 0$ iff $\langle x,y\rangle \notin F$. The net is *finite* iff $S \cup T$ is finite. Arcs $\langle x,y\rangle \in F$ and $\langle x',y'\rangle \in F$ form a *self-loop* iff $x' = y$ and $y' = x$. The net has a self-loop iff some arcs of the net form a self-loop. A place s has an *infinite capacity* iff $K(s) = \omega$. The net is a *net with infinite capacities* iff each place has an infinite capacity. For any $k \in N$, the net is *k-safe* iff

$$\forall s \in S\ K(s) \leq k \wedge (\forall t \in s^\bullet\ W(s,t) \leq K(s)) \wedge (\forall t \in {}^\bullet s\ W(t,s) \leq K(s)).\ \square$$

If s is a place of a net and M is a marking of the net, then $M(s)$ is called the number of *tokens* in s at M. If each place has an infinite capacity, then the set of markings of the net is simply $(S \to N)$.

In our figures, places are circles, transitions are rectangles, and the initial marking is shown by the distribution of tokens, black dots, onto places. The weight of an arc is shown iff it is not 1. The capacity of a place is shown iff it is not ω, by the inscription $K = n$.

Definition 2.2 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. □

If each place has an infinite capacity, then the output places of a transition do not affect the enabledness of the transition. Our enabledness condition is weaker than Reisig's enabledness condition [3] that requires $M(s) + W(t,s) \leq K(s)$ instead of $M(s) - W(s,t) + W(t,s) \leq K(s)$. We have chosen this enabledness condition because we want to have meaningful self-loops even in 1-safe place/transition nets.

Finite transition sequences and reachability are introduced in Definition 2.3. We shall use ε to denote the empty sequence.

Definition 2.3 Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net. For any $T_s \subseteq T$,

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

T_s^* is called the *set of finite sequences of transitions in* T_s, and 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

$$\forall M \in \mathcal{M} \; M[\varepsilon\rangle M, \text{ and}$$

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

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. For any $k \in N$, the net is *k-bounded* iff for each reachable marking M and for each place $s \in S$, $M(s) \leq k$. 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

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

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. Clearly, every k-safe place/transition net is k-bounded, but the converse does not hold. In fact, each place/transition net can be transformed into a behaviourally equivalent net with infinite capacities by adding so called *complement places* [3]. For each k-safe place/transition net, there is thus a behaviourally equivalent k-bounded place/transition net with infinite capacities.

Definition 2.4 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 2.4 is like a part of Definition 2.3 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 2.5 Let $\langle S, T, F, K, W, M_0\rangle$ be a place/transition net. The *set of infinite transition sequences of the net* is the set of functions from N to T, $(N \to T)$. The function ς from $(N \to T) \times N$ to T^* is defined by

$$(\forall \sigma \in (N \to T)\ \varsigma(\sigma, 0) = \varepsilon), \text{ and}$$
$$(\forall \sigma \in (N \to T)\ \forall n \in N\ \varsigma(\sigma, n+1) = \varsigma(\sigma, n)\sigma(n)).$$

If σ is an infinite transition sequence and $n \in N$, $\varsigma(\sigma, n)$ is called the *prefix of length n of σ*. An infinite transition sequence σ is *enabled at a marking M* ($M[\sigma\rangle$ for short) iff for each $n \in N$, the prefix of length n of σ is enabled at M. An infinite transition sequence σ is *disabled at a marking M* iff σ is not enabled at M. Let f be a function from $\mathcal{M}$ to 2^T. An infinite transition sequence σ is *f-enabled at a marking M* ($M[\sigma\rangle_f$ for short) iff for each $n \in N$, the prefix of length n of σ is f-enabled at M. □

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

3 Symmetries

The *symmetry method* [1, 4, 5] produces a virtual reachability graph containing for each reachable marking one and only one member of the equivalence class of the marking, the corresponding equivalence relation being determined by the *symmetries* of the net.

Definition 3.1 Let $\langle S, T, F, K, W, M_0\rangle$ be a place/transition net. Let f be a function f from $S \cup T$ to $S \cup T$. Then we say that f *respects places and transitions* iff for each place s and for each transition t, $f(s)$ is a place and $f(t)$ is a transition. Respectively, f *respects arcs* iff $\forall\langle x, y\rangle \in (S \times T) \cup (T \times S)\ W(f(x), f(y)) = W(x, y)$. Finally, f is a *symmetry* iff f is a bijection from $S \cup T$ to $S \cup T$ and respects places, transitions, and arcs. A function g from $S \cup T$ to $S \cup T$ is a *completion* of a function h from S to S iff $\forall s \in S\ g(s) = h(s)$. A function from S to S is a *place symmetry* iff some of its completions is a symmetry. A function g from $\mathcal{M}$ to $\mathcal{M}$ is a *marking symmetry* iff there exists some place symmetry h such that for each place s, $g(M)(h(s)) = M(s)$. A marking M is *symmetric to* a marking M' iff there exists some marking symmetry h such that $h(M) = M'$. □

It is straightforward to show that the symmetries of a net form a group under substitution and that the same holds for place and marking symmetries. Consequently, the symmetricity relation between markings is an equivalence relation.

Schmidt [4] has presented an algorithm that computes the symmetries of a given net. The symmetries can then be used for finding all those markings that are symmetric to a given marking. For example, in reduced reachability graph generation we can search for such marking in the generated part of the graph that is symmetric to a given marking. However, we do not actually need the set of all symmetries for reduced reachability graph generation. It is completely sufficient to compute the set of all place symmetries. In this section, we derive from the algorithm of Schmidt [4] an algorithm that computes place symmetries instead of symmetries. The derived algorithm is particularly useful when there are many symmetries corresponding to one place symmetry. On the other hand, the algorithm always consumes at most as much time and space as the algorithm of Schmidt.

We call a collection of sets a *partition of A* iff the sets in the collection are nonempty and pairwise disjoint, and the union of all the sets in the collection is A.

Definition 3.2 Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net. A set $C \subseteq (2^{S \cup T} \times 2^{S \cup T})$ is a *constraint* iff $\{A \mid \exists B \ \langle A, B \rangle \in C\}$ and $\{B \mid \exists A \ \langle A, B \rangle \in C\}$ are partitions of $S \cup T$. A function f from $S \cup T$ to $S \cup T$ is *consistent with a constraint* C iff

$$\forall \langle A, B \rangle \in C \ \{f(a) \mid a \in A\} = B.$$

A function from S to S is consistent with a constraint iff some of the completions of the function is consistent with the constraint. □

Note that the set of symmetries consistent with $\{\langle S, S \rangle, \langle T, T \rangle\}$ is the set of all symmetries of the net if we assume that S and T are nonempty. It is easy to see that if a function from $S \cup T$ to $S \cup T$ is consistent with a constraint, the function is a bijection from $S \cup T$ to $S \cup T$ as well as a bijection from A to B for every $\langle A, B \rangle$ in the constraint.

We now define the operation REFINE in the same way as Schmidt [4].

Definition 3.3 Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net. For any subset A of $S \cup T$ and for any natural number i, the set $\{x \in S \cup T \mid \exists y \in A \ W(x, y) = i\}$ is denoted by $F_i A$ whereas the set $\{x \in S \cup T \mid \exists y \in A \ W(y, x) = i\}$ is denoted by $A F_i$. Let C be a constraint having the element $\langle A, B \rangle$ and the element $\langle A', B' \rangle$. Let i be a positive natural number. The *REFINE operation on C and i* produces the set

$$\begin{array}{l} (\ (C \setminus \{\langle A, B \rangle, \langle A', B' \rangle\}) \\ \quad \cup \{\ \langle A \cap F_i A', B \cap F_i B' \rangle, \langle A \setminus F_i A', B \setminus F_i B' \rangle, \\ \qquad \langle A' \cap A F_i, B' \cap B F_i \rangle, \langle A' \setminus A F_i, B' \setminus B F_i \rangle\}) \\ \setminus \{\langle \emptyset, \emptyset \rangle\}. \ \square \end{array}$$

Clearly, the result of any REFINE operation is a constraint. Schmidt [4] has shown that for any constraint C and for any positive natural number i, the set of symmetries consistent with the result of the REFINE operation on C and i is equal to the set of symmetries consistent with C. The REFINE operation can thus be used for removing some non-symmetries from the set of functions consistent with a constraint. If S and T are nonempty, we can start from the constraint $\{\langle S, S \rangle, \langle T, T \rangle\}$ and perform REFINE operations until we obtain a constraint that cannot be changed by any REFINE operation. The set of symmetries consistent with such constraint is then equal to the set of symmetries of the net. Note that such final constraint is unique since the result of two REFINE operations does not depend on the order of the operations. Unfortunately, the final constraint is $\langle S, S \rangle, \langle T, T \rangle$ if the underlying graph of the net is strongly connected. If we want a better approximation for the set of symmetries, we should start from a better initial constraint. Such initial constraint can be computed by classifying the places and transitions with respect to the arcs connected to them. We shall not go into the details of such classification in this report.

Lemma 3.4 *Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net. Let C be a constraint such that every pair in C contains two sets of same cardinality, no pair in C contains both places and transitions, no REFINE operation on C changes C, and all sets of places in C are singletons. Then all such functions from $S \cup T$ to $S \cup T$ that are consistent with C are symmetries.*

Proof. Let f be a function from $S \cup T$ to $S \cup T$ and consistent with C. Clearly, f is a bijection from $S \cup T$ to $S \cup T$ and respects places and transitions. We show that f respects arcs.

Let s_1 be a place and t_1 a transition such that $W(s_1, t_1) \neq 0$. Let $\langle A', B' \rangle \in C$ such that $t_1 \in A'$. Then $f(t_1) \in B'$ since f is consistent with C. The set $B' \cap \{f(s_1)\}F_{W(s_1,t_1)}$ must be equal to B' since f is consistent with C, no REFINE operation changes C, and $A' \cap \{s\}F_{W(s_1,t_1)} \neq \emptyset$. Consequently, $W(f(s_1), f(t_1)) = W(s_1, t_1)$.

Let s_2 be a place and t_2 a transition such that $W(f(s_2), f(t_2)) \neq 0$. Let $\langle A'', B'' \rangle \in C$ such that $t_2 \in A''$. Then $f(t_2) \in B''$ since f is consistent with C. The set $A'' \cap \{s_2\}F_{W(f(s_2),f(t_2))}$ must be equal to A'' since f is consistent with C, no REFINE operation changes C, and $B'' \cap \{f(s_2)\}F_{W(f(s_2),f(t_2))} \neq \emptyset$. Consequently, $W(s_2, t_2) = W(f(s_2), f(t_2))$.

Let s' be a place and t' a transition such that $W(t', s') \neq 0$. Let $\langle A_1, B_1 \rangle \in C$ such that $t' \in A_1$. Then $f(t') \in B_1$ since f is consistent with C. The set $B_1 \cap F_{W(t',s')}\{f(s')\}$ must be equal to B_1 since f is consistent with C, no REFINE operation changes C, and $A_1 \cap F_{W(t',s')}\{s'\} \neq \emptyset$. Consequently, $W(f(t'), f(s')) = W(t', s')$.

Let s'' be a place and t'' a transition such that $W(f(t''), f(s'')) \neq 0$. Let $\langle A_2, B_2 \rangle \in C$ such that $t'' \in A_2$. Then $f(t'') \in B_2$ since f is consistent with C. The set $A_2 \cap F_{W(f(t''),f(s''))}\{s''\}$ must be equal to A_2 since f is consistent with C, no REFINE operation changes C, and $B_2 \cap F_{W(f(t''),f(s''))}\{f(s'')\} \neq \emptyset$. Conseqently, $W(t'', s'') = W(f(t''), f(s''))$. □

Lemma 3.4 tells us that we can extract a place symmetry from a constraint even if some sets of transitions in the constraint are not singletons.

We now present the modified version of the algorithm of Schmidt [4]. We assume that the reader has the article of Schmidt [4] available so that we can keep close to the presentation of Schmidt [4] without having to repeat everything.

The computation of the set of place symmetries of a given net is started by computing an initial constraint such that every symmetry of the net is consistent with the constraint. Then REFINE operations are performed until such constraint is obtained that cannot be changed by any REFINE operation. Let C_* be the resulting constraint. If all sets of places in C_* are singletons, there is exactly one place symmetry, namely the identity function from S to S. Otherwise the set PlaceSymmetries is made empty and the procedure DefineSym in Figure 1 is called with the argument $C = C_*$. When the execution of the procedure ends, the set PlaceSymmetries contains a set of generators that generate the group of all place symmetries of the net.

The DEFINE operation is defined as follows. Let C be a constraint and x a place. Let $\langle A, B \rangle$ be the unique pair in C satisfying $x \in A$. Let $y \in B$. The result of the DEFINE operation on C, x, and y is the set

$$(C \setminus \{\langle A, B \rangle\}) \cup \{\langle A \setminus \{x\}, B \setminus \{y\} \rangle, \langle \{x\}, \{y\} \rangle\}.$$

The function AllInGroup has a constraint as an argument and returns true iff there is a place symmetry σ in the group generated by the set PlaceSymmetries in such a way that each pair of singleton sets of places in the constraint is $\langle \{s\}, \{\sigma(s)\} \rangle$ for

```
PROCEDURE DefineSym(C:constraint)
VAR
    x,y:     place;
    σ:       place symmetry;
    C_new:   constraint;
    K,L,M:set of places;
BEGIN
    x := the least place which isn't a member of
        a singleton first component of C;
    [K,L] := the corresponding pair;
    M := L;
    WHILE M ≠ ∅ DO
        y := the least element of M; M := M - {y};
        C_new := DEFINE on C, x, and y;
        IF NOT AllInGroup(C_new) THEN (* ! *)
            REPEAT
                REFINE C_new;
            UNTIL nothing changes;
            IF every pair in C_new contains two sets of same cardinality
                IF AllSetsOfPlacesAreSingletons(C_new) THEN
                    σ := MakePlaceSymmetry(C_new);
                    PlaceSymmetries := PlaceSymmetries ∪ { σ };
                ELSE
                    DefineSym(C_new);
                END;
            END;
        END; (* ! *)
    END;
END DefineSym;
```

Figure 1: A procedure for computation of a set of generators for the group of place symmetries of a net.

some place s. The implementation of AllInGroup seems to require computing the generated place symmetries and keeping them in memory or in some device.

The function MakePlaceSymmetry has a constraint as an argument and returns a function from S to S that is consistent with the constraint. It follows from the context that one and only one such function exists.

Our algorithm always consumes at most as much time and space as the algorithm of Schmidt [4]. This can be shown by comparing the procedure DefineSym in Figure 1 to the procedure DefineSym in [4]. The essential difference between the procedures is the fact that our procedure does not investigate the different possibilities to map transitions to each other.

Theorem 3.5 *Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net. Let C_* be as above. Then the procedure DefineSym called with the argument $C = C_*$ computes a set of generators that generate the group of all place symmetries of the net.*

Proof. The proof of Theorem 1 in [4] suffices with the following modifications. Instead of Lemma 2 of [4], our Lemma 3.4 is used. The symmetries $\sigma, \sigma', \sigma^*$ and the identity symmetry occurring in the proof of Theorem 1 in [4] are replaced by place symmetries having the same names. □

As mentioned by Schmidt [4], the algorithm in [4] can be modified to compute subgroups of symmetries simply by choosing a suitable initial constraint. Our algorithm can similarly be modified to compute subgroups of place symmetries.

4 Stubborn Sets and CFFD-Equivalence

A *stubborn set* consists of some transitions of a net such that the set is in some sense independent of the complement of the set. Stubbornness is a marking-dependent property, i.e. a set can be stubborn at some marking while not stubborn at some other marking. In *stubborn set selective reachability graph generation,* a stubborn set is computed at each encountered marking, and only the transitions in the set are fired from that marking. As shown by Valmari [6], all reachable terminal markings occur in the reduced reachability graph. Also, if there is an infinite path in the full reachability graph, then the reduced reachability graph has an infinite path, too [7].

CFFD-equivalence [2, 8] is a behavioural equivalence which has a close relationship to linear time temporal logic. The expression "CFFD" is an abbreviation of the expression "chaos-free failures divergences". The preservation of *CFFD-semantics* means that the reduced reachability graph is CFFD-equivalent to the full reachability graph. Valmari has presented an advanced version of the stubborn set method which preserves CFFD-semantics [8]. In this section, we present an algorithm that computes stubborn sets for the CFFD-preserving stubborn set method in such a way that the computed sets are minimal with respect to enabled transitions and certain other conditions. Before considering CFFD-semantics, we present the basic version of the stubborn set method.

Various definitions of stubbornness have been given in the literature. Here we have chosen a rather weak definition of stubbornness. The weakness is good since the weaker is the definition of stubbornness, the better are the chances to find stubborn sets having a small number of enabled transitions. We do not know any better simple heuristic for minimizing the number of markings in the reduced reachability graph than the minimization of the number of transitions fired at a marking.

Definition 4.1 Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net with infinite capacities. The function E_1 from $\mathcal{M} \times S$ to 2^T, the functions E_2 and E_3 from $\mathcal{M} \times T \times S$ to 2^T, and the function E_4 from S to 2^T are defined as follows: let $M \in \mathcal{M}$, $t \in T$, and $s \in S$. Then

$$\begin{aligned}
E_1(M,s) &= \{t' \in {}^\bullet s \mid M(s) \geq W(s,t') \wedge W(t',s) > W(s,t')\},\\
E_2(M,t,s) &= E_4(s) \cup \{t' \in s^\bullet \mid\ & W(s,t) > W(t,s) \wedge\\
& & W(s,t') > M(s) - W(s,t) + W(t,s)\},\\
E_3(M,t,s) &= E_1(M,s) \cup \{t' \in {}^\bullet s \mid M(s) \geq W(s,t') \wedge W(t',s) > W(t,s)\}, \text{ and}\\
E_4(s) &= \{t' \in s^\bullet \mid W(s,t') > W(t',s)\}. \quad \square
\end{aligned}$$

Intuitively, $E_1(M,s)$ is the set of transitions that could increase the number of tokens in s and are not disabled by s at M. Correspondingly, $E_2(M,t,s)$ is the set of transitions that could decrease the number of tokens in s or get disabled because of the firing of t at M. Respectively, $E_3(M,t,s)$ is the set of transitions that are not disabled by s at M and could increase the number of tokens in s or deposit more tokens to s than t. Finally, $E_4(s)$ is the set of transitions that could decrease the number of tokens in s.

Definition 4.2 Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net with infinite capacities. A transition t is a *key transition of a set* $T_s \subseteq T$ *at a marking* M iff $t \in T_s$, t is enabled at M, and

$$\forall s \in {}^\bullet t \ E_4(s) \subseteq T_s.$$

A set $T_s \subseteq T$ is *stubborn at a marking* M iff some transition is a key transition of T_s at M and each transition t in T_s satisfies

$$\begin{aligned}
&(\exists s \in {}^\bullet t \ M(s) < W(s,t) \wedge E_1(M,s) \subseteq T_s) \vee\\
&(M[t\rangle \wedge (\forall s \in {}^\bullet t \ W(s,t) \leq W(t,s) \vee E_2(M,t,s) \subseteq T_s \vee E_3(M,t,s) \subseteq T_s)). \quad \square
\end{aligned}$$

In [9] it is shown that using stubborn sets of Definition 4.2 in a stubborn set selective reachability graph generation guarantees that all reachable terminal markings are found and the existence of infinite paths is detected if the net and the set of reachable markings are finite.

We now present an algorithm that computes stubborn sets that are minimal with respect to enabled transitions. This algorithm has earlier been presented in [9]. We call it the *deletion algorithm,* according to the algorithm of Valmari [6] from which it was derived.

Definition 4.3 Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net with infinite capacities and M a marking of the net. The *and/or-graph at* M is a triple

$\langle V_{\mathrm{AND}}, V_{\mathrm{OR}}, A\rangle$ such that the set of and-vertices V_{AND} is

$$\begin{array}{l}\{s \mid \exists t \in T\ \neg M[t\rangle \wedge s \in {}^\bullet t\}\cup\\ \{t \in T \mid M[t\rangle\}\cup\\ \{\langle t,s,i\rangle \mid t \in T \wedge M[t\rangle \wedge s \in {}^\bullet t \wedge W(s,t) > W(t,s) \wedge i \in \{2,3\}\},\end{array}$$

the set of or-vertices V_{OR} is

$$\begin{array}{l}\{t \in T \mid \neg M[t\rangle\}\cup\\ \{\langle t,s\rangle \mid t \in T \wedge M[t\rangle \wedge s \in {}^\bullet t \wedge W(s,t) > W(t,s)\},\end{array}$$

and the set of edges A is

$$\begin{array}{l}\{\langle s,t'\rangle \mid \exists t \in T\ \neg M[t\rangle \wedge s \in {}^\bullet t \wedge t' \in E_1(M,s)\}\cup\\ \{\langle t,\langle t,s\rangle\rangle \mid t \in T \wedge M[t\rangle \wedge s \in {}^\bullet t \wedge W(s,t) > W(t,s)\}\cup\\ \{\langle\langle t,s,i\rangle,t'\rangle \mid\ t \in T \wedge M[t\rangle \wedge s \in {}^\bullet t \wedge W(s,t) > W(t,s) \wedge\\ \qquad\qquad i \in \{2,3\} \wedge t' \in E_i(M,t,s)\}\cup\\ \{\langle t,s\rangle \mid t \in T \wedge \neg M[t\rangle \wedge s \in {}^\bullet t\}\cup\\ \{\langle\langle t,s\rangle,\langle t,s,i\rangle\rangle \mid t \in T \wedge M[t\rangle \wedge s \in {}^\bullet t \wedge W(s,t) > W(t,s) \wedge i \in \{2,3\}\}.\end{array}$$

A set $V_s \subseteq V_{\mathrm{AND}} \cup V_{\mathrm{OR}}$ is *legal* iff

$$\begin{array}{l}(\forall x \in V_s \cap V_{\mathrm{AND}}\ \forall y \in V_{\mathrm{AND}} \cup V_{\mathrm{OR}}\ \langle x,y\rangle \in A \Rightarrow y \in V_s),\\ (\forall x \in V_s \cap V_{\mathrm{OR}}\ \exists y \in V_s\ \langle x,y\rangle \in A),\ \text{and}\end{array}$$

some transition is a key transition of $V_s \cap T$ at M. □

It is straightforward to show that the set of transitions of any legal set is stubborn. Also, for each stubborn set, there exists a legal set such that the set of transitions of the legal set is the stubborn set. Clearly, the set of vertices of the and/or-graph is legal iff the marking is nonterminal.

The deletion algorithm can be described as follows. Let the net be finite and the marking nonterminal. Let's use the terms of Definition 4.3 though it is not necessary to construct any explicit and/or-graph. Only such vertex that is both accessible from some enabled transition and backwards accessible from some enabled transition by the reflexive-transitive closure of A has to be explicitly presented if the and/or-graph is constructed. The set of vertices of the and/or-graph is a legal set, maximal with respect to set inclusion, that contains all enabled transitions. Let then V_s be a legal set, maximal with respect to set inclusion, such that the set of transitions in V_s is a stubborn set T_s. Let $t \in T_s$ be enabled. An attempt to remove t from V_s is performed by starting from t, moving backwards in the and/or-graph, and attempting to remove those vertices that must be removed to get a legal subset, maximal with respect to set inclusion, of $V_s \setminus \{t\}$. When such vertex is encountered that does not have to be removed, the backward search is not continued from the vertex. The special condition related to E_4 (the last condition in the definition of legal sets) is not checked until the cumulative removal attempt is over. If the set of vertices that were not attempted to remove from V_s is legal, the cumulative removal attempt is realized. The set of transitions in the new legal set is then a stubborn subset, maximal with respect to set inclusion, of $T_s \setminus \{t\}$. On the other hand, if the attempt to remove t from V_s fails, it is not possible to remove t from any subset of V_s either. The deletion algorithm proceeds

by repeatedly attempting to remove a new enabled transition from the current legal set and realizing each successful attempt. The set of vertices of the and/or-graph is the initial legal set. The algorithm stops when it is no longer possible to remove any enabled transition from the current legal set. The set of transitions of the final legal set is minimal in the sense that no proper subset of its enabled transitions can be the set of enabled transitions of any stubborn set. This result is based on what is stated above about the set of transitions of the current legal set before and after removing an enabled transition.

The time taken by an execution of this algorithm is at most proportional to $\mu\nu|T|^2$, where μ is the maximum number of input places of a transition, and ν is the maximum number of adjacent transitions of a place. Our deletion algorithm has no factor that would make it more than a constant times slower than the deletion algorithm in [6].

We now turn to CFFD-semantics. After that we shall return to the above deletion algorithm and modify it in such a way that the preservation of CFFD-semantics can be guaranteed.

Definition 4.4 Let $\langle S, T, F, K, W, M_0\rangle$ be a place/transition net. The function $\Re$ from $2^T \times T^*$ to T^* is defined by requiring that for each $T_s \subseteq T$,

$$\begin{array}{l}\Re(T_s, \varepsilon) = \varepsilon,\\ (\forall \sigma \in T^*\ \forall t \in T_s\ \Re(T_s, \sigma t) = \Re(T_s, \sigma)t), \text{ and}\\ (\forall \sigma \in T^*\ \forall t \in T \setminus T_s\ \Re(T_s, \sigma t) = \Re(T_s, \sigma)).\end{array}$$

We call $\Re(T_s, \sigma)$ the *restriction of σ to T_s*. A marking M' is *virtually reachable from a marking M by a finite transition sequence σ with respect to a transition set T_s* ($M[\sigma\rangle^{T_s} M'$ for short) iff there exists some finite transition sequence δ such that σ is the restriction of δ to T_s and δ leads from M to M'. A finite transition sequence σ is *virtually enabled at a marking M with respect to a transition set T_s* ($M[\sigma\rangle^{T_s}$ for short) iff some marking is virtually reachable from M by σ with respect to T_s. Let f be a function from $\mathcal{M}$ to 2^T. A marking M' is *virtually f-reachable from a marking M by a finite transition sequence σ with respect to a transition set T_s* ($M[\sigma\rangle_f^{T_s} M'$ for short) iff there exists some finite transition sequence δ such that σ is the restriction of δ to T_s and δ f-leads from M to M'. A finite transition sequence σ is *virtually f-enabled at a marking M with respect to a transition set T_s* ($M[\sigma\rangle_f^{T_s}$ for short) iff some marking is virtually f-reachable from M by σ with respect to T_s. □

Definition 4.5 Let $\langle S, T, F, K, W, M_0\rangle$ be a finite place/transition net. Let T_s be a subset of T and f a function from $\mathcal{M}$ to 2^T. A marking M is *f-stable with respect to T_s* iff no transition in $T \setminus T_s$ is f-enabled at M. A marking M is *f-instable with respect to T_s* iff M is not f-stable with respect to T_s. The pair of a finite transition sequence σ and a transition set A, $\langle \sigma, A\rangle$, is an *f-stable failure of a marking M with respect to T_s* iff there exists some marking M' such that $M[\sigma\rangle_f^{T_s} M'$ and no transition in $A \cup (T \setminus T_s)$ is f-enabled at M'. A finite transition sequence σ is an *f-divergence trace of a marking M with respect to T_s and f* iff there exist some marking M' and some infinite transition sequence δ such that $M[\sigma\rangle_f^{T_s} M'$, δ is f-enabled at M', and $\forall n \in N\ \delta(n) \in T \setminus T_s$. Let g be a function from $\mathcal{M}$ to 2^T. The functions f and the g are *chaos-free failures divergences equivalent* (*CFFD-equivalent* for short) *with respect to T_s* iff the set of f-stable failures of M_0 with respect to T_s is equal to the

set of g-stable failures of M_0 with respect to T_s, the set of f-divergence traces of M_0 with respect to T_s is equal to the set of g-divergence traces of M_0 with respect to T_s, and M_0 is either both f-stable and g-stable or both f-instable and g-instable with respect to T_s. □

We can extend the definition of the CFFD-equivalence by saying that two reduced reachability graphs are CFFD-equivalent with respect to T_s iff for all functions f and g such that one of the graphs is the f-reachability graph and the other graph is the g-reachability graph, f and g are CFFD-equivalent with respect to T_s. This definition is motivated by the fact that functions f' and g' are CFFD-equivalent with respect to any transition set if the f'-reachability graph is equal to the g'-reachability graph. As mentioned immediately after Definition 2.4, we include the full reachability graph in the class of reduced reachability graphs. We can thus talk about the CFFD-equivalence between the full reachability graph and some reduced reachability graph.

Let's consider an arbitrary linear time temporal logic formula that does not contain the "next state" operator but may have the subformula "the future is infinite". Let T_s be a set of transitions that contains all those transitions that can affect the truth values of the atomic subformulae of the formula. Kaivola and Valmari [2] have shown that if a reduced reachability graph is CFFD-equivalent to the full reachability graph with respect to T_s, then the formula is satisfiable in M_0 with respect to the reduced reachability graph iff the formula is satisfiable in M_0 with respect to the full reachability graph.

From now on, we assume that the transitions of the net have been divided into two disjoint sets, called *visible transitions* and *invisible transitions.* Whenever we say that two graphs are CFFD-equivalent, we actually mean that they are CFFD-equivalent with respect to the set of visible transitions.

The *CFFD-preserving stubborn set method* [8] produces a reduced reachability graph which is CFFD-equivalent to the full reachability graph. Valmari has shown that CFFD-preservation can be guaranteed by requiring that the following conditions hold [8].

A. The net and the set of reachable markings are finite.

B. The stubborn set contains all visible transitions or no enabled visible transition.

C. If an enabled invisible transition exists, at least one enabled invisible transition of the stubborn set remains enabled, whatever happens in the environment. This condition was obtained from Valmari in a private discussion. Valmari uses a stronger condition in [8], but it is easy to modify the proofs in [8] in such a way that this refined condition suffices.

D. Every elementary cycle in the reduced reachability graph contains at least one such marking where the chosen stubborn set contains all visible transitions.

We now present such version of the deletion algorithm that produces stubborn sets for the CFFD-preserving stubborn set method. As before, we assume that we are

in a nonterminal marking. The and/or-graph is still the same as in Definition 4.3 and determines which vertices must be removed if a given vertex is removed. We say that a set of vertices V_s of the and/or-graph is *invisibly legal* iff V_s is legal and some invisible transition is a key transition of $V_s \cap T$. Clearly, the set of transitions of an invisibly legal set is stubborn and satisfies the above condition C. There may be stubborn sets satisfying the condition C and not corresponding to any invisibly legal set, but it is hard if not impossible to re-express the condition C by means of simple static conditions. The modified deletion algorithm is as follows.

1. If there is no enabled invisible transition, return the set of all transitions.
2. Start from the set of vertices of the and/or-graph and remove all visible transitions if you can do it in such a way that the set of remaining vertices in the and/or-graph is invisibly legal. If you cannot do it, go to step 3. Otherwise remove enabled invisible transitions as long as the set of remaining vertices of the and/or-graph is invisibly legal, and return the set of transitions in the final set of vertices.
3. Start from the set of vertices of the and/or-graph and remove enabled invisible transitions as long as the set of remaining vertices of the and/or-graph is invisibly legal and contains all visible transitions. Then return the set of transitions of the final set of vertices.

The resulting set is a stubborn set which satifies the above conditions B and C and is minimal in the sense that no proper subset of its enabled transitions can be the set of all enabled transitions of any stubborn set satisfying the condition B and corresponding to an invisibly legal set.

When we presented the deletion algorithm for the basic version of the stubborn set method, we said that only such vertex that is both accessible from some enabled transition and backwards accessible from some enabled transition by the reflexive-transitive closure of the binary edge relation has to be explicitly presented if an explicit and/or-graph is constructed. To be able to execute the above step 3, we should also present all those vertices that are both accessible from some disabled visible transition and backwards accessible from some enabled transition.

To satisfy the above condition D, loops are detected during reduced reachability graph generation. At each encountered nonterminal marking, a preliminary stubborn set is computed by starting from the above step 1. If the preliminary stubborn set does not contain enabled visible transitions but would cause a nonempty sequence of invisible transitions leading from the current marking back to the marking, the preliminary stubborn set is replaced by a new stubborn set which is computed by the above step 3. As shown by Valmari [8], the reduced reachability graph can be generated in a depth-first order without visiting any marking more than once.

The definition of CFFD-equivalence can be generalized to concern arbitrary *labelled transition systems* (*LTS's* for short) [8]. If we want to compute an LTS that is smaller than but CFFD-equivalent to the *parallel composition* of given LTS's [8], we can construct a 1-bounded place/transition net that simulates the parallel composition, generate a reduced reachability graph of the net using the above algorithm, and

transform the reduced reachability graph into an LTS. The transitions corresponding to *invisible internal actions* and *hidden synchronization actions* are invisible.

5 Conclusions

We have presented an algorithm for computing the place symmetries of a given place/transition net. Compared to the symmetry computation algorithm of Schmidt [4], it may save both time and space. Place symmetries are sufficient in ordinary reachability analysis, whereas the different ways to map transitions to each other seem to be of more theoretical than practical interest.

We have extended the so called deletion algorithm, familiar from the basic stubborn set method [6], to compute stubborn sets for the CFFD-preserving stubborn set method [8]. The computed sets are minimal with respect to enabled transitions and a couple of static conditions which guarantee CFFD-preservation. The minimality is useful since the less enabled transitions there are in the stubborn sets used during reduced reachability graph generation, the less states we usually have to take into the reduced reachability graph. The deletion algorithm does not compute a stubborn set as fast as the so called *incremental algorithm* [6, 8], but the incremental algorithm has no practical minimality guarantee and may sometimes produce fatally large stubborn sets [9]. Though we presented the extended deletion algorithm for place/transition nets, it should be easy to derive algorithms from it for other models of concurrency where stubbornness and CFFD-semantics have been defined.

Acknowledgements

This work has been funded by the Technology Development Centre of Finland (TEKES). I am grateful to Doctor Karsten Schmidt, Acting Associate Professor Mikko Tiusanen, and Associate Professor Antti Valmari for fruitful discussions on the subject of this research.

References

[1] Huber, P., Jensen, A.M., Jepsen, L.O., and Jensen, K.: *Towards Reachability Trees for High-Level Petri Nets.* Aarhus University, Department of Computer Science, Technical report DAIMI PB-174, Aarhus 1985, 19+42 p.

[2] Kaivola, R., and Valmari, A.: *The Weakest Compositional Semantic Equivalence Preserving Nexttime-less Linear Temporal Logic.* Cleaveland, W.R. (Ed.), Proceedings of the 3rd International Conference on Concurrency Theory, Stony Brook NY, August 1992. Lecture Notes in Computer Science 630, Springer-Verlag, Berlin 1992, pp. 207–221.

[3] Reisig, W.: *Petri Nets: An Introduction.* EATCS Monographs on Theoretical Computer Science 4, Springer-Verlag, Berlin 1985, 161 p.

[4] Schmidt, K.: *Symmetries of Petri Nets.* Petri Net Newsletter 43 (1993), pp. 9–25.

[5] Starke, P.H.: *Reachability Analysis of Petri Nets Using Symmetries.* Systems Analysis – Modelling – Simulation 8 (1991) 4/5, pp. 293–303.

[6] Valmari, A.: *State Space Generation: Efficiency and Practicality.* Doctoral thesis, Tampere University of Technology Publications 55, Tampere 1988, 170 p.

[7] Valmari, A.: *Eliminating Redundant Interleavings during Concurrent Program Verification.* Proceedings of Parallel Architectures and Languages Europe '89 Vol. 2. Lecture Notes in Computer Science 366, Springer-Verlag, Berlin 1989, pp. 89–103.

[8] 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.

[9] Varpaaniemi, K.: *Efficient Detection of Deadlocks in Petri Nets.* Helsinki University of Technology, Digital Systems Laboratory Report A 26, Espoo, October 1993, 56 p.