Computing Reachability Properties Hidden in Finite Net Unfoldings

Burkhard Graves

Universität Hildesheim, Institut für Informatik,
Marienburger Platz 22, D-31141 Hildesheim, Germany
Fax: +49 5121 860475, email: graves@informatik.uni-hildesheim.de
(May 1997)

Abstract. It is commonly known that every reachable marking of a finite-state Petri net system is represented in its finite unfolding according to McMillan. Also the reachability of markings from each other is represented in the finite unfolding, but it is almost unknown that this information can be hidden very deep. This paper presents an efficient method for gaining this information, which is of course of great importance for potential modelcheckers working on finite unfoldings. All results presented in this paper also hold for a recently proposed optimized unfolding method.

1 Introduction and Motivation

A major drawback of interleaving semantics and modelcheckers based upon them is the so called *state explosion problem*. One among other approaches to cope with this problem is the use of partial order semantics instead of interleaving semantics [8]. Like many other works done in this area, this paper uses *finite 1-safe Petri net systems* to represent finite-state concurrent systems. Partial order semantics describes the behaviour of a net system by the set of its processes [3] or by its maximal branching process [5], also called *maximal unfolding* of the system, which can be seen as the union of all processes of the given system. However, if a system can exhibit an infinite behaviour, then at least one of its processes and consequently its maximal unfolding is infinite and therefore unsuitable for verification purposes. McMillan proposed in [8] an elegant algorithm for the computation of a finite initial part of the maximal branching process, called *finite unfolding*, in which every reachable marking of the system is represented. This work was refined and optimized by Esparza, Römer and Vogler in [7]; the finite unfolding calculated by their method is never bigger and often much smaller (in terms of orders of magnitude) than McMillan's finite unfolding, while still representing every reachable marking. However, in this paper we neglect the difference between these two unfolding methods. All results hold for both unfolding methods and the systems serving as examples have been choosen in a way, such that both unfolding methods yield the same finite unfolding (up to isomorphism). As already mentioned above, every reachable marking of a given system is represented in its finite unfolding. However,

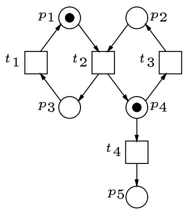

Fig. 1. A finite 1-safe net system.

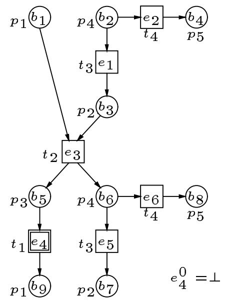

Fig. 2. Its finite unfolding β_f.

the reachability of markings from each other is deeply embedded in the finite unfolding. Consider for example the system and its finite unfolding displayed in Fig.1 and Fig.2. The reachable marking $\{p_2, p_3\}$ is represented by a process contained in the finite unfolding corresponding to the configuration $\{e_1, e_3, e_5\}$ (describing the occurrence sequence $t_3t_2t_3$). Obviously, the deadlock marking $\{p_1, p_5\}$ is reachable from $\{p_2, p_3\}$ (by the occurrence sequence $t_1t_2t_1t_4$, for example), but how can this information be gained from the finite unfolding? Now, imagine all processes describing a run of a system to a given marking. In general, only a few of these processes are totally contained in the finite unfolding, but it is always the case that some initial part of every such process is contained in the finite unfolding. In our example, there are infinitely many processes describing a run of the system into the deadlock marking $\{p_1, p_5\}$, but only two of them are totally contained in the finite unfolding. One corresponds to the configuration $\{e_2\}$, the other one corresponds to the configuration $\{e_1, e_3, e_4, e_6\}$. Each of the remaining processes is prefixed by a process corresponding to the configuration $\{e_1, e_3, e_4, e_5\}$. Since $\{e_1, e_3, e_5\}$ is a subset of $\{e_1, e_3, e_4, e_5\}$, it can be concluded that $\{p_1, p_5\}$ is reachable from $\{p_2, p_3\}$. This example demonstrates that a specific classification of configurations contained in the finite unfolding is of great importance for potential modelcheckers. Three types of configurations can be distinguished wrt. a given marking: Configurations of the first type correspond to processes describing runs to the given marking, configurations of the second type correspond to processes which cannot be extended to processes describing runs to the given marking, and configurations of the third type are configurations which are neither of type one nor of type two. Since this classification is a disjoint partitioning of all configurations, the knowledge of two classes yields the third one.

The configurations of type one can be easily calculated; two different algorithms for this task can be found in [6]. However, the classification of the remaining configurations is a problem. Esparza tries to solve this problem in [6] by introducing a 'shift' operator working on configurations. Configurations of the third type should be calculated by repeated applications of this operator on configurations of the first type. Unfortunately, this does not work in some cases, as we will see in Sect.4. One could say, that the finite unfolding as defined by

McMillan or by Esparza/Römer/Vogler is 'too small', because the problem can be fixed by creating a sufficiently large finite unfolding which does not contain these special cases mentioned above. But this 'brute force' method would significantly slow down potential model checking algorithms, e.g. the one proposed in [6]. This paper presents another solution, namely a modification of the 'shift' operator, such that it works as the old operator was supposed to do on the McMillan unfolding as well as on the Esparza/Römer/Vogler unfolding.

2 Basic Definitions

Following is a series of definitions, notions and theorems (without proofs) in a very brief form. More details can be found in the referenced literature.

Petri Nets. A triple $N = (S, T, F)$ is a *net* if $S \cap T = \emptyset$ and $F \subseteq (S \times T) \cup (T \times S)$. The elements of S are called *places*, the elements of T *transitions*. Places and transitions are generically called *nodes*. N is *finite* if $|S \cup T| \in \mathbb{N}$. We identify the *flow relation* F with its characteristic function on the set $(S \times T) \cup (T \times S)$. The *preset* of a node x, denoted by ${}^\bullet x$, is the set $\{y \in S \cup T \mid F(y,x) = 1\}$. The *postset* of x, denoted by $x^\bullet$, is the set $\{y \in S \cup T \mid F(x,y) = 1\}$. Pre- and postsets are generalized to sets of nodes $X \subseteq S \cup T$ in the following way: ${}^\bullet X = \bigcup_{x \in X} {}^\bullet x$, $X^\bullet = \bigcup_{x \in X} x^\bullet$ (notice ${}^\bullet\emptyset = \emptyset^\bullet = \emptyset$). A *marking* M of a net (S, T, F) is a mapping $S \to \mathbb{N}$. A 4-tuple $\Sigma = (S, T, F, M^0)$ is a *net system* if (S, T, F) is a net and M^0 is a marking of (S, T, F); M^0 is called *initial marking* of Σ. Σ is *finite* if the underlying net is finite.

A marking M *enables* a transition t if $\forall s \in S : M(s) \geq F(s,t)$. A marking enabling no transition is a *deadlock marking*. If a transition t is enabled at M, then it can *occur*, and its occurrence leads to a new marking M', denoted by $M[t\rangle M'$, such that $\forall s \in S : M'(s) = M(s) - F(s,t) + F(t,s)$. A sequence of transitions $\sigma = t_1 \ldots t_n$ ($n \in \mathbb{N}$) is an *occurrence sequence* if there exist markings $M_0, \ldots, M_n$ such that $M_0[t_1\rangle M_1[t_2\rangle \ldots [t_n\rangle M_n$. M_n is the marking reached from M_0 by the occurrence of σ, denoted by $M_0[\sigma\rangle M_n$. M' is *reachable* from M if there exists an occurrence sequence σ such that $M[\sigma\rangle M'$. The set of all markings which can be reached from M is denoted by $[M\rangle$.

A marking M of a net (S, T, F) is *1-safe* if $\forall s \in S : M(s) \leq 1$. We identify 1-safe markings with the set of places s such that $M(s) = 1$. A system is 1-safe if all its reachable markings are 1-safe.

Figure 1 (page 2) shows a finite 1-safe system; its initial marking is $\{p_1, p_4\}$.

Branching Processes. A branching process of a system is a special kind of net, called occurrence net, together with a certain homomorphism showing that this net can be interpreted as an unfolding of the system containing information about both concurrency and conflicts. In order to avoid confusions coming from the fact that the semantics of a (marked) net is again a (labelled) net, different names are used for the nodes of the net system and for those of the occurrence net which describes the system's semantics: the places of occurrence nets are called *conditions*, and their transitions are called *events*. We quickly review the main definitions and results of [5], where the notion 'branching process' was introduced first:

Let (S, T, F) be a net. The transitive closure of F, denoted by $\prec$, is called *causal relation*. The symbol $\preceq$ denotes the reflexive and transitive closure of F. $Min(N)$ equals $\{x \in S \cup T \mid \neg\exists y \in S \cup T : y \prec x\}$. For $x \in S \cup T$ and $X \subseteq S \cup T$, we say $x \prec X$ if $\exists y \in X : x \prec y$ (analogously for $\preceq$, $\succ$ and $\succeq$). Two nodes $x_1, x_2 \in S \cup T$ are in *conflict*, denoted by $x_1 \# x_2$, if $\exists t_1, t_2 \in T, t_1 \neq t_2, {}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset : t_1 \preceq x_1 \wedge t_2 \preceq x_2$. A node $x \in S \cup T$ is in *self-conflict* if $x \# x$. We say $x_1\, co\, x_2$ if neither $x_1 \prec x_2$ nor $x_2 \prec x_1$ nor $x_1 \# x_2$ holds.

An *occurrence net* is a net $N = (B, E, F)$ such that

(i) $\forall b \in B : |{}^\bullet b| \leq 1$ (iii) $\neg\exists e \in E : e \# e$

(ii) $\neg\exists x \in B \cup E : x \prec x$ (iv) $\forall x \in B \cup E : |\{y \in B \cup E \mid y \prec x\}| \in \mathbb{N}$.

If moreover $|b^\bullet| \leq 1$ holds for every $b \in B$, then N is called *causal net*.

Let $N_1 = (S_1, T_1, F_1)$ and $N_2 = (S_2, T_2, F_2)$ be two nets. A *homomorphism* from N_1 to N_2 is a mapping $h : S_1 \cup T_1 \to S_2 \cup T_2$ with $h(S_1) \subseteq S_2$ and $h(T_1) \subseteq T_2$ such that for every $t \in T_1$ the restriction of h to ${}^\bullet t$ is a bijection between ${}^\bullet t$ and ${}^\bullet h(t)$, analogously for $t^\bullet$ and $h(t)^\bullet$.

A *branching process* of a net system $\Sigma = (N, M^0)$ is a pair $\beta = (N', p)$ where $N' = (B, E, F)$ is an occurrence net and p is a homomorphism from N' to N such that the restriction of p to $Min(N')$ is a bijection between $Min(N')$ and M^0 and $\forall e_1, e_2 \in E : ({}^\bullet e_1 = {}^\bullet e_2 \wedge p(e_1) = p(e_2)) \Rightarrow e_1 = e_2$. If N' is a causal net, then β is a *process* of Σ.

Let $\beta_1 = (N_1, p_1)$ and $\beta_2 = (N_2, p_2)$ be two branching processes of a net system. A *homomorphism* from β_1 to β_2 is a homomorphism h from N_1 to N_2, such that $p_2 \circ h = p_1$ and the restriction of h to $Min(N_1)$ is a bijection between $Min(N_1)$ and $Min(N_2)$. β_1 and β_2 are *isomorphic* if there is a bijective homomorphism from β_1 to β_2. Intuitively, two isomorphic branching processes differ only in the names of their conditions and events. It is shown in [5] that a net system has an unique maximal branching process up to isomorphism. We call it *maximal unfolding* of the system and denote it by $\beta_m = (B_m, E_m, F_m, p_m)$.

β_1 is a *prefix* of β_2 if N_1 is a subnet of N_2 and, moreover, there exists an injective homomorphism from β_1 to β_2. Figure 2 (page 2) shows a prefix of the maximal unfolding β_m of the finite 1-safe system displayed in Fig.1. It should be clear, that a repeated continuation with four events and five conditions labelled and 'arranged' like $e_3, \ldots, e_6$ and $b_5, \ldots, b_9$, respectively, yields the maximal unfolding[1].

Configurations, Cuts and more. A *configuration* of an occurrence net $N = (B, E, F)$ is a causally closed, conflict-free set of events $C \subseteq E$, which means $\forall e, e' \in E : (e \preceq e' \wedge e' \in C) \Rightarrow e \in C$ and $\forall e, e' \in C : \neg(e \# e')$. Given $e \in E$, the set $[e] = \{e' \in E \mid e' \preceq e\}$ is a configuration, called *local configuration* of e.

A set of conditions $B' \subseteq B$ is a *co-set* if its elements are pairwise in the *co* relation. A maximal co-set wrt. set inclusion is a *cut*. A marking M of a

[1] For example: $p(e_7) = t_2$, ${}^\bullet e_7 = \{b_7, b_9\}$, $e_7^\bullet = \{b_{10}, b_{11}\}$, $p(b_{10}) = p_3$, $p(b_{11}) = p_4$, etc.

system Σ is *represented* in a branching process $\beta = (N, p)$ of Σ if β contains a cut c such that, for each place s of Σ, c contains exactly $M(s)$ conditions b with $p(b) = s$. Every marking represented in a branching process is reachable, and every reachable marking is represented in the maximal unfolding of the net system.

Finite configurations and cuts are tightly related: Let C be a finite configuration of a branching process $\beta = (N, p)$. Then $Cut(C) = (Min(N) \cup C^\bullet) \setminus {}^\bullet C$ is a cut representing the marking $Mark(C) = p(Cut(C))$. Two configurations C_1 and C_2 of a branching process *correspond to each other* if $Mark(C_1) = Mark(C_2)$. A pair (C_1, C_2) of corresponding configurations is called *cc-pair*.

Let $\beta = (B, E, F, p)$ be a branching process of a net system $\Sigma = (N, M^0)$ and let c be a cut of β. The set $\{x \in B \cup E \mid x \succeq c \wedge \forall y \in c : \neg(x \# y)\}$ is denoted by $\uparrow c$. Identifying F and p with their restrictions to $\uparrow c$, $\Uparrow c = (B \cap \uparrow c, E \cap \uparrow c, F, p)$ is a branching process of $(N, p(c))$; moreover, if $\beta = \beta_m$ then $\Uparrow c$ is the maximal branching process of $(N, p(c))$. It follows that $\Uparrow Cut(C_1)$ and $\Uparrow Cut(C_2)$ are isomorphic, provided (C_1, C_2) is a cc-pair; in this case we denote the (unique) isomorphism from $\Uparrow Cut(C_1)$ to $\Uparrow Cut(C_2)$ by $I_{(C_1, C_2)}$.

McMillan's Finite Unfolding. Here we only present McMillan's unfolding method. The refined method of Esparza, Römer and Vogler is more complicated; interested readers are referred to [7]. As already mentioned, the differences between these two unfolding methods are not relevant for this paper.

Let $\beta = (B, E, F, p)$ be a branching process of a net system Σ. We say that β is *complete* if every reachable marking of Σ is represented in β and, moreover, β contains an event labelled by t if a transition t can occur in Σ. The maximal unfolding of a net system is always complete. Since a finite 1-safe net system has only finitely many reachable markings, its maximal unfolding contains at least one complete finite prefix.

An event $e \in E$ is a *cut-off event* if there exists a local configuration $[e']$ such that $Mark([e']) = Mark([e])$ and $|[e']| < |[e]|$. To achieve that e is a cut-off event if $Mark([e]) = M^0$, a new 'virtual' event $\perp$ is introduced and $[\perp]$ is defined as the empty configuration: $Mark([\perp]) = M^0 = Mark([e])$ and $|[\perp]| = 0 < |[e]|$. Given a cut-off event e, there may exist several e' such that $Mark([e']) = Mark([e])$ and $|[e']| < |[e]|$. We assume in the sequel that for each cut-off event e one of these e' is fixed, call it *corresponding event* of e and denote it by e^0. Moreover, we assume without loss of generality that e^0 is *not* a cut-off event.

Let $E_f \subseteq E_m$ defined by: $e \in E_f$ iff no event $e' \prec e$ is a cut-off event. The (unique) prefix of β_m having E_f as set of events is called *McMillan's finite unfolding* and denoted by $\beta_f = (B_f, E_f, F_f, p_f)$. In [6], β_f is proved to be always complete. *Off* denotes the set of cut-off events of β_f. $\mathcal{F}_f$ denotes the set of all configurations of β_f. $\mathcal{D}$ denotes the set of all maximal configurations of β_f wrt. set inclusion. The set of all configurations contained in the maximal unfolding β_m is denoted by $\mathcal{F}_m$.

Figure 2 (page 2) shows the finite unfolding β_f of the finite 1-safe system displayed in Fig.1. e_4 is the only cut-off event, $e_4^0 = \perp$ is its corresponding event. Note, that indeed $Mark([e_4]) = Mark([\perp]) = \{p_1, p_4\}$. $\mathcal{D}$ contains three maximal configurations: $D_1 = \{e_2\}$, $D_2 = \{e_1, e_3, e_4, e_6\}$ and $D_3 = \{e_1, e_3, e_4, e_5\}$.

3 Classification of Configurations

Remark. To simplify subsequent definitions, we assume a finite 1-safe net system $\Sigma = (S, T, F, M^0)$ together with its (possibly infinite) maximal unfolding $\beta_m = (B_m, E_m, F_m, p_m)$ and its finite unfolding $\beta_f = (B_f, E_f, F_f, p_f)$ to be given throughout the rest of this paper. The prefix and the postfix of a (set of) node(s) and the operators $\uparrow$ and $\Uparrow$ always refer to the maximal unfolding β_m.

It is beyond the scope of this paper to explain the ideas of [6], where a model checking algorithm is introduced, which should be able to check formulas of a simple branching time logic. However, the following is similar to what can be found in [6], but due to the fact that we are only interested in recognizing the mutual reachability of markings, we are able to simplify some definitions.

Definition 1. Let C be a configuration and let $\mathcal{C}$, $\mathcal{C}_1$ and $\mathcal{C}_2$ be sets of configurations. We say

$$C \leq \mathcal{C} \text{ iff } \exists C' \in \mathcal{C} : C \subseteq C' \qquad \text{and} \qquad \mathcal{C}_1 \leq \mathcal{C}_2 \text{ iff } \forall C \in \mathcal{C}_1 : C \leq \mathcal{C}_2.$$

The restriction onto the finite unfolding β_f is denoted by

$$\triangledown C = C \cap E_f \qquad \text{and} \qquad \triangledown\, \mathcal{C} = \{\triangledown C \mid C \in \mathcal{C}\}.$$

The set of the maximal elements contained in $\mathcal{C}$ wrt. set inclusion is denoted by

$$max(\mathcal{C}) = \{C \in \mathcal{C} \mid \neg\exists C' \in \mathcal{C} : C \subset C'\}.$$

Notice that $max(\mathcal{C})$ may equal the empty set if $|\mathcal{C}| \notin \mathbb{N}$.

Definition 2. Let M be a marking. We define

$$\begin{aligned} Sat_m(M) &= \{C \mid C \in \mathcal{F}_m \wedge Mark(C) = M\}, \\ Sat_f(M) &= \{C \mid C \in \mathcal{F}_f \wedge Mark(C) = M\}, \\ Last(M) &= max(Sat_f(M)). \end{aligned}$$

In terms of Sect.1, the set $Sat_m(M)$ contains configurations of the first type, which correspond to processes describing runs to the marking M. The same holds for $Sat_f(M)$, but wrt. the finite unfolding β_f. $Last(M)$ can be seen as a 'compact representation' of $Sat_f(M)$, because every configuration contained in $Sat_f(M)$ is (at least) a subset of a configuration contained in $Last(M)$. Due to its compactness, the set $Last(M)$ can be calculated easily (provided that $\mathcal{D}$ is known, but this set can be calculated in advance by the unfolding mechanism); two different algorithms for this task can be found in [6]. In our first example, $Last(\{p_1, p_5\})$ equals $\{\{e_2\}, \{e_1, e_3, e_4, e_6\}\}$. But as we have seen, the knowledge of $Last(\{p_1, p_5\})$ is not enough to detect that $\{p_1, p_5\}$ is reachable from $\{p_2, p_3\}$. The following proposition shows that we are interested in $max(\triangledown Sat_m(M))$, the compact representation of $\triangledown Sat_m(M)$.

Proposition 3. *Let M_1 and M_2 be two markings.*
$M_2 \in [M_1\rangle$ *iff* $max(\bigtriangledown Sat_m(M_1)) \leq max(\bigtriangledown Sat_m(M_2))$.

Unfortunately, the sets $Sat_m(M_1)$ and $Sat_m(M_2)$ may be infinite. Fortunately, the following section shows that $max(\bigtriangledown Sat_m(M))$ is equal to the maximum of a set which can be calculated by some finite applications of a 'shift' operator on the finite set $Last(M)$.

4 The Shift Operators

In the following, we present two shift operators which are generalizations of (slightly modified[2]) operators originally defined in [6]. The first (basic) operator shifts a configuration wrt. a cc-pair, the second (complex) operator shifts a set of configurations wrt. a set of cc-pairs. By choosing these cc-pairs, the operators may be tuned. If they are tuned in such a way that they correspond to the ones defined in [6], we obtain a problem, as we will show. Hence we will propose a somewhat different tuning and we will show, as a result, that the problem disappears.

Let (C_1, C_2) be a cc-pair. As in [6], the branching process $\Uparrow Cut(C_2)$ can be thought to be $\Uparrow Cut(C_1)$ 'shifted forward' (or 'shifted backwards' if $|C_2| < |C_1|$). Accordingly, if C_1 is a subset of some configuration C of β_m then $C \setminus C_1$ is a configuration of $\Uparrow Cut(C_1)$, $I_{(C_1,C_2)}(C \setminus C_1)$ is a configuration of $\Uparrow Cut(C_2)$ and $C_2 \cup I_{(C_1,C_2)}(C \setminus C_1)$ is again a configuration of β_m, which can be thought to be C 'shifted forward' wrt. the cc-pair (C_1, C_2). The following is a formal definition of this operation.

Definition 4. Let $\Upsilon \subseteq \mathcal{F}_m \times \mathcal{F}_m$ be a set of cc-pairs and $(C_1, C_2) \in \Upsilon$. The *basic shift operator* is defined by

$$S_{(C_1,C_2)} : \begin{cases} \{C \in \mathcal{F}_m \mid C_1 \subseteq C\} \to \{C \in \mathcal{F}_m \mid C_2 \subseteq C\} \\ \qquad\qquad\qquad C \mapsto C_2 \cup I_{(C_1,C_2)}(C \setminus C_1). \end{cases}$$

The *complex shift operator* is defined by

$$S_\Upsilon : \begin{cases} 2^{\mathcal{F}_m} \to 2^{\mathcal{F}_m} \\ \quad \mathcal{C} \mapsto \mathcal{C} \cup \{S_{(C_1,C_2)}(C) \mid (C_1, C_2) \in \Upsilon \wedge C_1 \subseteq C \in \mathcal{C}\}. \end{cases}$$

The least fixpoint of S_Υ containing $\mathcal{C}$ is given by

$$\mu S_\Upsilon.\mathcal{C} = \bigcup_{n \geq 0} S^n_\Upsilon(\mathcal{C}).$$

For convenience and in accordance with [6], we fix the following

Notation 5. We abbreviate $S_\Upsilon(\mathcal{C})$ by $S(\mathcal{C})$ if $\Upsilon = \{([e^0], [e]) \mid e \in \mathit{Off}\}$. Moreover, if $(C_1, C_2) = ([e^0], [e])$ holds for some $e \in \mathit{Off}$, we abbreviate $I_{([e^0],[e])}(C)$ by $I_e(C)$ and $S_{([e^0],[e])}(C)$ by $S_e(C)$; the latter is called *elementary shift*.

[2] Merely for technical reasons, in order to get more algebraic properties.

In Fig.2 we have for example $S_{e_4}(\{e_2\}) = [e_4] \cup I_{e_4}(\{e_2\}) = \{e_1, e_3, e_4\} \cup \{e_6\}$. The following is a first step towards the aim formulated at the end of Sect.3.

Theorem 6. $max(\bigtriangledown Sat_m(M)) = max(\bigtriangledown \mu S.Sat_f(M)) = max(\bigtriangledown \mu S.Last(M))$.

Of course, there are still some problems: The calculation of $\mu S.Last(M)$ requires infinite applications of S, $\mu S.Last(M)$ itself may be infinite and, in particular, potential modelcheckers have only the finite unfolding β_f at their disposal. Therefore, they can only work with finite versions of the shift operators.

Definition 7. Let Υ be a set of cc-pairs, $(C_1, C_2) \in \Upsilon$, $\mathcal{C} \subseteq \mathcal{F}_f$ and $C \in \mathcal{C}$. We identify the finite versions of the shift operators by overbarring:

$$\begin{aligned} \bar{S}_{(C_1,C_2)}(C) &= \bigtriangledown S_{(C_1,C_2)}(C) \ , \\ \bar{S}_\Upsilon(\mathcal{C}) &= \bigtriangledown S_\Upsilon(\mathcal{C}) \qquad \text{and} \\ \mu\bar{S}_\Upsilon.\mathcal{C} &= \textstyle\bigcup_{n\geq 0} \bar{S}^n_\Upsilon(\mathcal{C}) \quad . \end{aligned}$$

Observe that $\bar{S}_\Upsilon(\mathcal{C}) = \mathcal{C} \cup \{\bar{S}_{(C_1,C_2)}(C) \mid (C_1, C_2) \in \Upsilon \wedge C_1 \subseteq C \in \mathcal{C}\}$ and that the calculation of $\bar{S}_{(C_1,C_2)}(C)$ does not require the maximal branching process β_m; the knowledge of the pairs $(e', I_{(C_1,C_2)}(e')) \in E_f^2$ for every $(C_1, C_2) \in \Upsilon$ is sufficient (an algorithm for calculating these pairs can be found in [6]). Moreover, since $\bar{S}^n_\Upsilon(\mathcal{C})$ is a set of configurations of the finite unfolding for every n, and the finite unfolding contains only finitely many configurations, there exists some $k \in \mathbb{N}$ such that $\bar{S}^k_\Upsilon(\mathcal{C}) = \bar{S}^{k+1}_\Upsilon(\mathcal{C})$. We have then $\mu\bar{S}_\Upsilon.\mathcal{C} = \bigcup_{0\leq n\leq k} \bar{S}^n_\Upsilon(\mathcal{C})$. Now, consider

$$\bigtriangledown\mu S.\mathcal{C} \stackrel{?}{=} \mu\bar{S}.\mathcal{C} \text{ for every } \mathcal{C} \subseteq \mathcal{F}_f \qquad \text{(cf. [6] p.177).} \tag{1}$$

If this was true, then $\bigtriangledown\mu S.Last(M)$ would equal $\mu\bar{S}.Last(M)$, which would solve all our remaining problems, because the set $\mu\bar{S}.Last(M)$ is finite and its calculation requires only a finite number of applications of $\bar{S}$. Unfortunately, (1) is not true; Fig.3 gives a counterexample. Consider the (always) reachable deadlock marking $\{p_7\}$, $Last(\{p_7\}) = \{\{e_2, e_5, e_8\}\}$. We have[3]

$$\begin{aligned} &S_{e_6}(S_{e_4}(\{e_2, e_5, e_8\})) = S_{e_6}(\{e_1, e_4, e_5', e_8'\}) = \{e_1, e_3, e_6, e_7, e_9, e_{10}, e_8''\}, \\ &\text{therefore } \{e_1, e_3, e_6, e_7, e_9, e_{10}, e_8''\} \in \mu S.\{\{e_2, e_5, e_8\}\} \\ &\text{and } \{e_1, e_3, e_6, e_7, e_9, e_{10}\} \in \bigtriangledown\mu S.\{\{e_2, e_5, e_8\}\}. \end{aligned}$$

Notice that e_5 was shifted out of β_f and then returned (as e_{10}). However, $\{e_1, e_3, e_6, e_7, e_9, e_{10}\} \notin \mu\bar{S}.Last(\{p_7\})$, because once the event e_5 is shifted out of the finite unfolding, the finite version $\bar{S}$ of the shift function S forgets about it. The problem is caused by the event e_4. This event belongs to a class of events, which is characterized in the next definition.

Definition 8 (Tricky events). Let $e \in \mathit{Off}$ and let (C_1, C_2) be a cc-pair. e *tricks* (C_1, C_2) iff $e \in \uparrow Cut(C_1) \wedge (I_{(C_1,C_2)}(e))^{\bullet\bullet} \cap \uparrow Cut(C_2) \cap E_f \neq \emptyset$.

[3] The (double) primed events are not contained in the finite unfolding, because they lie after some cut-off event. But it should be clear, which events are meant in the maximal branching process, e.g. ${}^\bullet e_5' = \{b_{14}\}$.

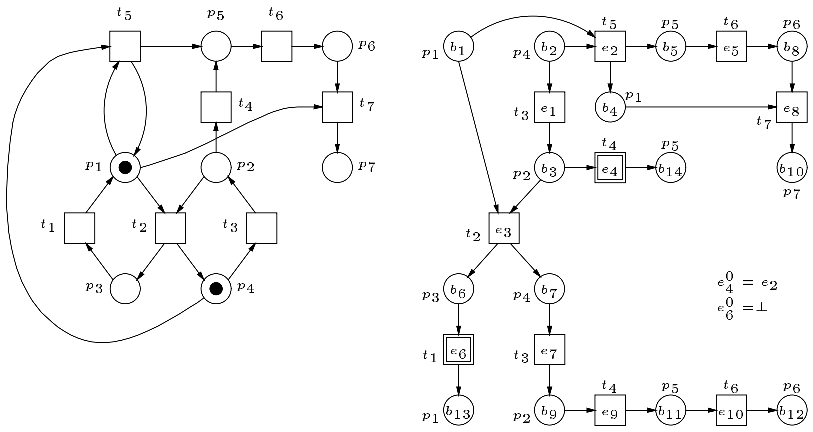

Fig. 3. A finite 1-safe net system and its finite unfolding β_f.

Tricky events are recognized when calculating[4] the pairs $(e', I_{(C_1,C_2)}(e')) \in E_f^2$ for a cc-pair (C_1, C_2). In our example, e_4 is recognized as tricky (it tricks $(\emptyset, [e_6])$) because $e_4 \in \uparrow \mathit{Cut}(\emptyset) \wedge (I_{(\emptyset,[e_6])}(e_4))^{\bullet\bullet} \cap \uparrow \mathit{Cut}([e_6]) \cap E_f = \{e_{10}\} \neq \emptyset$.

Equation (1) is only true for finite unfoldings containing no cut-off events e_1 and e_2, such that e_1 tricks $([e_2^0], [e_2])$. This can be achieved in a 'brute force' way by enlarging the finite unfolding in an appropriate way, e.g. by introducing the additional requirement that for every cut-off event e its corresponding event e^0 must be an element of its local configuration ($e^0 \in [e]$). But this method would significantly slow down potential model checking algorithms, e.g. the one proposed in [6]. Keeping all events which fall out of the finite unfolding during repeated applications of $\bar{S}$ 'in mind' has a similar additional space and time complexity and is therefore out of the question. The approach we follow is to consider tricky events by combining sequences of shifts disturbed by them into 'clean' single shifts. In our example, instead of shifting the configuration $\{e_2, e_5, e_8\}$ two times, first wrt. e_4 and then wrt. e_6 (losing the event e_5 or, respectively, e_{10}), we circumvent the tricky event e_4 by shifting $\{e_2, e_5, e_8\}$ wrt. $(\{e_2\}, \{e_1, e_3, e_6, e_7, e_9\})$ in order to get the desired configuration $\{e_1, e_3, e_6, e_7, e_9, e_{10}\}$.

Therefore, by adding (as few as possible) appropriate cc-pairs (V^0, V) to the set $\{([e^0], [e]) \mid e \in \mathit{Off}\}$, we now construct a set Ω of cc-pairs taking all tricky events into account such that $\mathit{max}(\bigtriangledown \mu S.\mathit{Last}(M)) = \mathit{max}(\mu \bar{S}_\Omega.\mathit{Last}(M))$ holds. This solves our remaining problems, because the set $\mu \bar{S}_\Omega.\mathit{Last}(M)$ is finite and its calculation requires only a finite number of applications of $\bar{S}$. The construction of Ω does not establish $\bigtriangledown \mu S.\mathit{Last}(M) = \mu \bar{S}_\Omega.\mathit{Last}(M)$ (it can be checked later on that Fig.3 is a counterexample) nor $\mathit{max}(\bigtriangledown \mu S.\mathcal{C}) = \mathit{max}(\mu \bar{S}_\Omega.\mathcal{C})$ for an arbitrary

[4] Remember, this is necessary for performing a finite shift of a configuration.

$\mathcal{C} \subseteq \mathcal{F}_f$. A counterexample concerning the second equation is omitted due to its complexity; interested readers should contact the author.

In general, each event e tricking a cc-pair (V_1^0, V_1) is taken into account by introducing a cc-pair (V_2^0, V_2), whereby V_2 can be easily calculated by shifting $V_1^0 \cup [e]$ wrt. (V_1^0, V_1). Observe that $V_2 = \bar{S}_{(V_1^0,V_1)}(V_1^0 \cup [e])$ contains only the cut-off events which are already contained in V_1. The calculation of V_2^0 is a little bit harder, because two unpleasant things can happen.

First, it would be nice to calculate V_2^0 by shifting $V_1^0 \cup [e]$ backwards wrt. e. This would work in our example, because $\bar{S}_{e_4}^{-1}(\emptyset \cup [e_4]) = \{e_2\}$. Unfortunately, if there is another event e' tricking $([e^0], [e])$, it can happen that $S_e^{-1}(V_1^0 \cup [e]) \not\subseteq E_f$. This would mean that $\bar{S}_e^{-1}(V_1^0 \cup [e])$ does not exist, or in other words, $\bigtriangledown S_e^{-1}(V_1^0 \cup [e])$ is probably not a corresponding configuration of V_2. Therefore, it can happen that the calculation of V_2^0 depends on the existence of another cc-pair (V_3^0, V_3) already contained in Ω, which takes the tricky event e' into account. This shows that the elements of Ω must be calculated in a certain order. An example concerning this 'first unpleasant thing' can be found in App.A.

Second, suppose there is some 'appropriate' cc-pair (V_3^0, V_3) and let

$$(V_2^0, V_2) = (S^{-1}_{(V_3^0,V_3)}(V_1^0 \cup [e]), S_{(V_1^0,V_1)}(V_1^0 \cup [e])).$$

In this case, it can happen that V_2^0 contains some cut-off event e'. Let $b' \in e'^{\bullet}$ and $b \in \mathit{Cut}(V_2)$ with $p(b) = p(b')$. If there is some $e'' \in b^{\bullet} \cap \uparrow \mathit{Cut}(V_2) \cap E_f$ then we are again confronted with the problem that there is no $e''' \in \uparrow \mathit{Cut}(V_2^0) \cap E_f$ with $I_{(V_2^0,V_2)}(e''') = e''$. To solve this problem, we shift $V_1^0 \cup [e]$ backwards (starting wrt. some appropriate $(V_3^0, V_3) \in \Omega$) until it contains no cut-off events anymore[5]. Notice that this sequence of reverse shifts may require a sequence of appropriate cc-pairs already contained in Ω. We call Ω *shift complete wrt.* $V_1^0 \cup [e]$ if such cc-pairs exist.

For simplification reasons we fix the following notation.

Notation 9. Let $C \in \mathcal{F}_m$ and let $a = (V_1^0, V_1) \ldots (V_n^0, V_n)$ be a sequence of cc-pairs ($n \in \mathbb{N}$, $n = 0 \Rightarrow a = \epsilon$). We define $S_a(C) = X_n$ and $S_a^{-1}(C) = Y_n$ with $X_0 = Y_0 = C$ and, for $0 \le i < n$,

$$X_{i+1} = \begin{cases} S_{(V_{i+1}^0, V_{i+1})}(X_i) & \text{, if } X_i \text{ is defined} \\ \text{undefined} & \text{, otherwise} \end{cases}$$

$$\text{and} \qquad Y_{i+1} = \begin{cases} S^{-1}_{(V_{n-i}^0, V_{n-i})}(Y_i) & \text{, if } Y_i \text{ is defined} \\ \text{undefined} & \text{, otherwise.} \end{cases}$$

[5] An example concerning this 'second unpleasant thing' is omitted due to its complexity; interested readers should contact the author.

Definition 10. Let Υ be a set of cc-pairs. We call Υ *shift complete with respect to a configuration* $C \in \mathcal{F}_f$ if $C \cap \mathit{Off} = \emptyset$ or if there exist some cc-pairs $(W_1^0, W_1), \ldots, (W_n^0, W_n) \in \Upsilon$ $(n \in \mathbb{N} \setminus \{0\})$ such that

$$\begin{array}{ll} & \forall i, 1 \leq i \leq n : S^{-1}_{(W_i^0, W_i) \ldots (W_n^0, W_n)}(C) \subseteq E_f \\ \text{and} & \quad S^{-1}_{(W_1^0, W_1) \ldots (W_n^0, W_n)}(C) \cap \mathit{Off} = \emptyset. \end{array}$$

In this case, we denote the sequence $(W_1^0, W_1) \ldots (W_n^0, W_n)$ by Υ^C. If Υ is shift complete wrt. all $C \in \mathcal{F}_f$ then we call Υ *shift complete*.

Observe that Υ^C does not have to be unique.

Definition 11. The set Ω is defined by $\Omega = \bigcup_{i \geq 0} X_i$ with

$$\begin{aligned} X_0 &= \{([e^0], [e]) \mid e \in \mathit{Off}\} \quad \text{and} \\ X_{i+1} &= X_i \cup \{(V_2^0, V_2) \mid \ \exists e \in \mathit{Off} \ \ \exists (V_1^0, V_1) \in X_i : \\ &\qquad\qquad e \text{ tricks } (V_1^0, V_1) \\ &\qquad\qquad \wedge X_i \text{ is shift complete wrt. } V_1^0 \cup [e] \\ &\qquad\qquad \wedge V_2 = S_{(V_1^0, V_1)}(V_1^0 \cup [e]) \\ &\qquad\qquad \wedge V_2^0 = S_D^{-1}(V_1^0 \cup [e]) \text{ with } D = X_i^{V_1^0 \cup [e]}\}. \end{aligned}$$

Since $\Omega \subseteq \mathcal{F}_f$, there exists some $k \in \mathbb{N}$ such that $\Omega = \bigcup_{0 \leq i \leq k} X_i$.

Proposition 12. *Ω is shift complete.*

Proof. Assume Ω not to be shift complete wrt. some $C' \in \mathcal{F}_f$. This means, that there is no sequence $\Omega^{C'}$, which implies that there is some C obtained by shifting C' backwards as often as possible without falling out of β_f such that

$$\forall (V^0, V) \in \Omega, V \subseteq C \ \exists e \in \mathit{Off} \ : e \text{ tricks } (V^0, V) \wedge I_{(V^0, V)}(e)^{\bullet\bullet} \cap C \neq \emptyset. \quad (2)$$

If (V_1^0, V_1) is one of these (V^0, V) and e_1 is the corresponding e then Ω is not shift complete wrt. $V_1^0 \cup [e_1]$ (otherwise the tricky event e_1 would have been taken into account due to the construction of Ω) and $|V_1^0 \cup [e_1]| < |C|$. This means, that there is no sequence $\Omega^{V_1^0 \cup [e_1]}$, which implies that there is some C obtained by shifting $V_1^0 \cup [e_1]$ backwards as often as possible without falling out of β_f such that (2) holds again. If (V_2^0, V_2) is one of these (V^0, V) and e_2 is the corresponding e then Ω is not shift complete wrt. $V_2^0 \cup [e_2]$ and $|V_2^0 \cup [e_2]| < |V_1^0 \cup [e_1]|$.

This argumentation can be repeated ad infinitum. Since there cannot exist an infinite sequence of configurations of decreasing size, the assumption must be false. □

A repeated application of the following Lemma shows that each non-elementary shift corresponds to a sequence of elementary shifts.

Lemma 13. *As in Definition 11, let* $(V_2^0, V_2) \in X_{i+1}$ *with* $V_2^0 = S_D^{-1}(V_1^0 \cup [e])$ *and* $V_2 = S_{(V_1^0,V_1)}(V_1^0 \cup [e])$ $(i \in \mathbb{N}, D = X_i^{V_1^0 \cup [e]})$ *and let* $C \in \mathcal{F}_m$.

$$S_{(V_2^0,V_2)}(C) = S_{(V_1^0,V_1)}(S_D(C)).$$

Finally we have

Theorem 14. $max(\bigtriangledown \mu S.Last(M)) = max(\mu \bar{S}_\Omega.Last(M))$.

Proof. Lemma 13 already yields

$$max(\bigtriangledown \mu S.Last(M)) \supseteq max(\mu \bar{S}_\Omega.Last(M)).$$

Let $lc(\mathcal{C}) = \{C \in \mathcal{C} \mid \neg\exists C' \in \mathcal{C} : |C'| < |C|\}$ denote the set of configurations of *l*owest *c*ardinality contained in some $\mathcal{C} \subseteq \mathcal{F}_m$ and let $C_1 \in max(\bigtriangledown \mu S.Last(M))$. Since Ω is shift complete, we can compute (by a finite amount of reverse shifts) C_1^0 with $C_1^0 \cap Off = \emptyset$ and $Mark(C_1^0) = Mark(C_1)$. Suppose $C_1^0 \not\leq Last(M)$ (otherwise we are done). Since $M \in [Mark(C_1^0)\rangle$, $\mathcal{C} = lc\{C \in \mathcal{F}_m \mid Mark(C) = M \wedge C_1^0 \subset C\} \neq \emptyset$. Let $C_2 \in \bigtriangledown\mathcal{C}$. Now we compute C_2^0 with $C_2^0 \cap Off = \emptyset$ and $Mark(C_2^0) = Mark(C_2)$. If again $C_2^0 \not\leq Last(M)$ we iterate the above procedure until we get some C_n^0 $(n > 2)$ with $C_n^0 \leq Last(M)$; this will happen sooner or later, because $|C' \setminus C_i^0|$ with $C' \in lc\{C \in \mathcal{F}_m \mid Mark(C) = M \wedge C_i^0 \subset C\}$ gets smaller in each iteration $(i = 1, 2, \ldots)$. By reverting all finite shifts that have been done, we can re-compute C_1 which shows

$$max(\bigtriangledown \mu S.Last(M)) \subseteq max(\mu \bar{S}_\Omega.Last(M)).$$

□

5 Summary, Conclusion and Outlook

This paper can be seen as a continuation of the work done in [6]. Its main contribution is a generalization and correction of the shift operators presented there. The necessity of this generalization is shown by uncovering several subtleties of the finite unfolding (e.g. the existence of tricky events). Finally, it is shown that with the new operators the computing of reachability properties hidden in the finite unfolding of a given system is an easy job (Prop.3, Theorem 6 and 14).

A properly working shift operator does not come for free. But since the construction of the shift complete set Ω is not very expensive[6] and has to be done only once and, in particular, before the potential start of a modelchecker working on the finite unfolding, it costs nearly nothing compared with the potentially numerous and expensive modelchecker runs. On the contrary, by summarizing sequences of shifts into single shifts, there may be chances for significant reductions of the time complexity of potential modelcheckers, e.g. the one proposed in [6]. This should be examined in future work.

Acknowledgments. I would like to thank Eike Best and Javier Esparza for reading an early draft version of this paper. Special thanks go to Michaela Huhn; she gave me a major hint for the proof of the shift completeness of Ω.

[6] A lot of examples have been checked; in general, even complex unfoldings contain only a few tricky events.

References

1. Bernardinello, L., De Chindio, F.: A survey of basic net models and modular net classes. G. Rozenberg, ed., Advances in Petri nets 1992, Lecture Notes in Computer science 609, (Springer, Berlin, 1992) 304–351.
2. Best, E., Devillers, R.: Sequential and Concurrent Behaviour in Petri Net Theory. Theoretical Computer Science, 55(1), (1987) 299–323.
3. Best, E., Fernández, C.: Nonsequential processes – a Petri net view. EATCS Monographs on Theoretical Computer Science 13 (1988).
4. Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Transactions on Programming Languages and Systems, 8(2), (1986) 244–263.
5. Engelfriet, J.: Branching processes of Petri nets. Acta Informatica 28, (1991) 575–591.
6. Esparza, J.: Model checking using net unfoldings. Science of Computer Programming 23, (1994) 151-195.
7. Esparza, J., Römer, S., Vogler, W.: An Improvement of McMillan's Unfolding Algorithm. T. Margaria, B. Steffen, ed., Proceedings of TACAS'96, LNCS 1055, (1996) 87–106.
8. McMillan, K.L.: Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. Proceedings of the 4th Workshop on Computer Aided Verification, (Montreal, 1992) 164–174
9. Petri, C.A.: Kommunikation mit Automaten. Schriften des Institutes für Instrumentelle Mathematik, (Bonn, 1962).
10. Queille, J.P., Sifakis, J.: Specification and verification of concurrent systems in CESAR. Proceedings of the 5th International Symposium on Programming, LNCS 137, (1981) 337–351.

A An Example Concerning the 'First Unpleasant Thing'

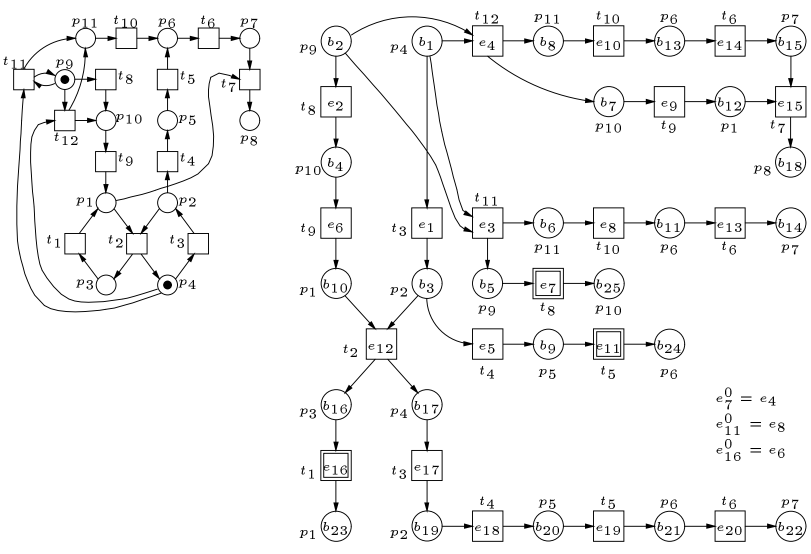

Fig. 4. A finite 1-safe net system and its finite unfolding β_f.

e_{11} is a tricky event; it tricks $([e_6], [e_{16}])$.

$$S_{e_{16}}([e_6] \cup [e_{11}]) = S_{e_{16}}(e_2, e_6, e_1, e_5, e_{11}) = \{e_1, e_2, e_6, e_{12}, e_{16}, e_{17}, e_{18}, e_{19}\}.$$

Notice that $\mathit{Mark}(\{e_1, e_2, e_6, e_{12}, e_{16}, e_{17}, e_{18}, e_{19}\}) = \{p_1, p_6\}$, but

$$\bigtriangledown S_{e_{11}}^{-1}([e_6] \cup [e_{11}]) = \bigtriangledown\{e_3, e_8, e_7, e_6'\} = \{e_3, e_8, e_7\}$$

and $\mathit{Mark}(\{e_3, e_8, e_7\}) = \{p_{10}, p_6\}$. This is caused by another tricky event, namely e_7 tricking $([e_8], [e_{11}])$, which can be taken into account by $(S_{e_7}^{-1}([e_8] \cup [e_7]), S_{e_{11}}([e_8] \cup [e_7])) = (\{e_4, e_{10}\}, \{e_1, e_5, e_{11}, e_2\})$. Observe, for example, that

$$S_{e_{11}}(S_{e_7}(\{e_4, e_9, e_{10}\})) = S_{e_{11}}(\{e_3, e_7, e_8, e_9'\}) = \{e_1, e_5, e_{11}, e_2, e_6\}$$
$$\text{and } \bar{S}_{e_{11}}(\bar{S}_{e_7}(\{e_4, e_9, e_{10}\})) = \{e_1, e_5, e_{11}, e_2\},$$
$$\text{but } S_{(\{e_4, e_{10}\}, \{e_1, e_5, e_{11}, e_2\})}(\{e_4, e_9, e_{10}\}) = \{e_1, e_5, e_{11}, e_2, e_6\}.$$

With $(\{e_4, e_{10}\}, \{e_1, e_5, e_{11}, e_2\}) \in \Omega$, the first component of the cc-pair taking the tricky event e_{11} into account can be calculated via

$$S_{(\{e_4, e_{10}\}, \{e_1, e_5, e_{11}, e_2\})}^{-1}(\{e_2, e_6, e_1, e_5, e_{11}\}) = \{e_4, e_{10}, e_9\}.$$

B Two Lemmas and Some More Proofs

Lemma 15. *Let $\mathcal{C}$ and $\mathcal{C}_i$ be sets of configurations ($i \in I$, I is a set of indices).*

$$
\begin{aligned}
&(i)\ \ max(max(\mathcal{C})) = max(\mathcal{C}), \quad \triangledown \triangledown \mathcal{C} = \triangledown \mathcal{C} \\
&(ii)\ \ max(\textstyle\bigcup_{i\in I} \mathcal{C}_i) = max(\textstyle\bigcup_{i\in I} max(\mathcal{C}_i)) \\
&(iii)\ \forall j \in J \subseteq I : |\mathcal{C}_j| \in \mathbb{N} \\
&\qquad \Longrightarrow max(\textstyle\bigcup_{i\in I} \triangledown \mathcal{C}_i) = max((\textstyle\bigcup_{j\in J} \triangledown max(\mathcal{C}_j)) \cup (\textstyle\bigcup_{i\in I\setminus J} \triangledown \mathcal{C}_i)) \\
&(iv)\ \ max(\mathcal{C}_i) \le max(\mathcal{C}_j) \wedge max(\mathcal{C}_j) \le max(\mathcal{C}_i) \\
&\qquad \Longleftrightarrow max(\mathcal{C}_i) = max(\mathcal{C}_j), \quad (i, j \in I).
\end{aligned}
$$

Notice that (iv) does not hold without the *max* operator.

Lemma 16. *Let $C \in \mathcal{F}_m$ and let (C_1, C_2) be a cc-pair with $C_1 \subseteq C$.*

$$
\begin{aligned}
&(i)\ \ Mark(C) = Mark(S_{(C_1,C_2)}(C)), \\
&(ii)\ \ |C_1| < |C_2| \Longleftrightarrow |S_{(C_1,C_2)}(C)| > |C| \ \textit{and} \\
&(iii)\ S_{(C_1,C_2)} \ \textit{is bijective}\ (S^{-1}_{(C_1,C_2)} = S_{(C_2,C_1)}) \ \textit{and monotonic wrt.}\ \subseteq.
\end{aligned}
$$

Moreover, let (D_1, D_2) be a cc-pair with $C_1 \subseteq D_1 \subseteq C$.

$$
\begin{aligned}
&(iv)\ \ I_{(C_1,C_2)}(C \setminus D_1) = I_{(D_1, S_{(C_1,C_2)}(D_1))}(C \setminus D_1), \\
&(v)\ \ I_{(D_1,D_2)}(C \setminus D_1) = I_{(S_{(C_1,C_2)}(D_1), D_2)}(S_{(C_1,C_2)}(C) \setminus S_{(C_1,C_2)}(D_1)).
\end{aligned}
$$

Proof of (v).

$$
\begin{aligned}
& I_{(S_{(C_1,C_2)}(D_1),D_2)}(S_{(C_1,C_2)}(C) \setminus S_{(C_1,C_2)}(D_1)) \\
= & \{\text{definition of } S_{(C_1,C_2)}\} \\
& I_{(C_2 \cup I_{(C_1,C_2)}(D_1\setminus C_1),D_2)}((C_2 \cup I_{(C_1,C_2)}(C \setminus C_1)) \setminus (C_2 \cup I_{(C_1,C_2)}(D_1 \setminus C_1))) \\
= & \{\text{set theory}\} \\
& I_{(C_2 \cup I_{(C_1,C_2)}(D_1\setminus C_1),D_2)}(I_{(C_1,C_2)}(C \setminus C_1) \setminus I_{(C_1,C_2)}(D_1 \setminus C_1)) \\
= & \{C_1 \subseteq D_1 \Rightarrow (C \setminus C_1) \setminus (D_1 \setminus C_1) = C \setminus D_1\} \\
& I_{(C_2 \cup I_{(C_1,C_2)}(D_1\setminus C_1),D_2)}(I_{(C_1,C_2)}(C \setminus D_1)) \\
= & \{\text{Lemma 16}(iv)\} \\
& I_{(C_2 \cup I_{(C_1,C_2)}(D_1\setminus C_1),D_2)}(I_{(D_1, C_2 \cup I_{(C_1,C_2)}(D_1\setminus C_1))}(C \setminus D_1)) \\
= & \{I_{(B,C)}(I_{(A,B)}) = I_{(A,C)}\} \\
& I_{(D_1,D_2)}(C \setminus D_1).
\end{aligned}
$$

□

Proof of Theorem 6. The first equality follows directly from

$$Sat_m(M) = \mu S.Sat_f(M)$$

which is proven in [6]. The second equality is proven indirectly in [6], but due to a problem in the proof, we give a direct one:

$$\begin{array}{l} \quad Last(M) = max(Sat_f(M)) \\ \Rightarrow \{\text{definition of } \leq \text{ and } max\} \\ \quad Sat_f(M) \leq Last(M) \\ \Rightarrow \{\text{monotonicity of } S_{(V^0,V)}\} \\ \quad \mu S.Sat_f(M) \leq \mu S.Last(M) \\ \Rightarrow \{\text{set theory}\} \\ \quad \bigtriangledown \mu S.Sat_f(M) \leq \bigtriangledown \mu S.Last(M) \\ \Rightarrow \{\text{definition of } max\} \\ \quad max(\bigtriangledown \mu S.Sat_f(M)) \leq max(\bigtriangledown \mu S.Last(M)) \end{array}$$

Together with $max(\bigtriangledown \mu S.Last(M)) \leq max(\bigtriangledown \mu S.Sat_f(M))$, which follows directly from $Last(M) \subseteq Sat_f(M)$, Lemma 15($iv$) yields the requested equality. □

Proof of Lemma 13. We proceed by induction over the length of $D = X_i^{V_1^0 \cup [e]} = (V_n^0, V_n) \ldots (V_3^0, V_3)$ $(n \geq 3)$.

- $n = 3 \quad (D = X_i^{V_1^0 \cup [e]} = (V_3^0, V_3))$

$$\begin{array}{ll} & S_{(V_2^0,V_2)}(C) \\ = & \{\text{by definition}\} \\ & V_2 \cup I_{(V_2^0,V_2)}(C \setminus V_2^0) \\ = & \{V_2^0 = S^{-1}_{(V_3^0,V_3)}(V_1^0 \cup [e])\} \\ & V_2 \cup I_{((V_3^0 \cup I^{-1}_{(V_3^0,V_3)}((V_1^0 \cup [e]) \setminus V_3)),V_2)}(C \setminus (V_3^0 \cup I^{-1}_{(V_3^0,V_3)}((V_1^0 \cup [e]) \setminus V_3))) \\ = & \{(V_1^0 \cup [e]) \setminus V_3 = V_1^0 \setminus V_3, \text{Lemma 16}(v) \text{ with (1)}\} \\ & V_2 \cup I_{((V_3 \cup V_1^0),V_2)}((V_3 \cup I_{(V_3^0,V_3)}(C \setminus V_3^0)) \setminus (V_3 \cup V_1^0)) \\ = & \{V_3 \notin I_{(V_3^0,V_3)}(C \setminus V_3^0)\} \\ & V_2 \cup I_{(V_3 \cup V_1^0,V_2)}(I_{(V_3^0,V_3)}(C \setminus V_3^0) \setminus V_1^0) \\ = & \{V_2 = S_{(V_1^0,V_1)}(V_1^0 \cup [e])\} \\ & V_2 \cup I_{(V_3 \cup V_1^0, V_1 \cup I_{(V_1^0,V_1)}((V_1^0 \cup [e]) \setminus V_1^0))}(I_{(V_3^0,V_3)}(C \setminus V_3^0) \setminus V_1^0) \\ = & \{(V_1^0 \cup [e]) \setminus V_1^0 = (V_3 \cup V_1^0) \setminus V_1^0, \text{Lemma 16}(iv) \text{ with (2)}\} \\ & V_2 \cup I_{(V_1^0,V_1)}(I_{(V_3^0,V_3)}(C \setminus V_3^0) \setminus V_1^0) \\ = & \{V_2 = S_{(V_1^0,V_1)}(V_1^0 \cup [e])\} \\ & V_1 \cup I_{(V_1^0,V_1)}((V_1^0 \cup [e]) \setminus V_1^0) \cup I_{(V_1^0,V_1)}(I_{(V_3^0,V_3)}(C \setminus V_3^0) \setminus V_1^0) \end{array}$$

$$= \quad \{(V_1^0 \cup [e]) \setminus V_1^0 = V_3 \setminus V_1^0\}$$

$$V_1 \cup I_{(V_1^0, V_1)}((V_3 \cup I_{(V_3^0, V_3)}(C \setminus V_3^0)) \setminus V_1^0)$$

$$= S_{(V_1^0, V_1)}(S_{(V_3^0, V_3)}(C)).$$

(1) $(C_1, C_2) = (V_3^0, V_3), (D_1, D_2) = (V_3^0 \cup I^{-1}_{(V_3^0, V_3)}((V_1^0 \cup [e]) \setminus V_3), V_2)$
(2) $(C_1, C_2) = (V_1^0, V_1), D_1 = V_3 \cup V_1^0$

- $n > 3 \quad (D = X_i^{V_1^0 \cup [e]} = a(V_3^0, V_3)$ with $a = (V_n^0, V_n) \ldots (V_4^0, V_4))$

$$S_{(V_1^0, V_1)}(S_{(V_3^0, V_3)}(S_a(C)))$$

$$= \quad \{\text{by induction hypothesis}\}$$

$$S_{(S^{-1}_{(V_3^0, V_3)}(V_1^0 \cup [e]), S_{(V_1^0, V_1)}(V_1^0 \cup [e]))}(S_a(C))$$

$= \quad \{$by repeated application of

$$C_1 \subseteq D_1 \Rightarrow S_{(D_1, D_2)}(S^{-1}_{(C_1, C_2)}(C)) = S_{(S_{(C_1, C_2)}(D_1), D_2)}(C)$$

which is an easy consequence of Lemma $16(v)\}$

$$S_{(S_a^{-1}(S^{-1}_{(V_3^0, V_3)}(V_1^0 \cup [e])), S_{(V_1^0, V_1)}(V_1^0 \cup [e]))}(C)$$

$$= S_{(V_2^0, V_2)}(C)$$

$\square$