Black Box Unfolding with Local First Search

Sebastien Bornot, Remi Morin, Peter Niebert, and Sarah Zennou

Laboratoire d'Informatique Fondamentale de Marseille (LIF)
Université de Provence – CMI
39, rue Joliot-Curie / F-13453 Marseille Cedex 13
[bornot,morin,niebert,zennou]@cmi.univ-mrs.fr

Abstract. McMillan's unfolding approach to the reachability problem in 1-safe Petri nets and its later improvements by Esparza-Römer-Vogler have proven in practice as a very effective method to avoid state-explosion. This method computes a *complete finite prefix* of the infinite branching process of a net. On the other hand, the Local First Search approach (LFS) was recently introduced as a new partial order reduction technique which characterizes a restricted subset of configurations that need to be explored to check *local properties*. In this paper we amalgamate the two approaches: We combine the reduction criterion of LFS with the notions of an adequate order and cutoff events essential to the unfolding approach. As a result, our new LFS method computes a reduced transition system without the problem of state duplication (present in the original LFS). Since it works for any transition system with an independence relation, this *black box partial unfolding* remains more general than the unfolding of Petri nets. Experiments show that the combination gives improved reductions compared to the original LFS.

1 Introduction

Model checking as an automatic method for proving simple system properties or finding witness executions of faulty systems suffers from the well known state explosion problem: Typically, the number of states explored by naive algorithms is exponential in the size of the system description, so that often this automatic approach is limited to very small systems. However, the explosion of the number of states is typically due to redundancies in the exploration of the whole global state space and in the interleaving semantics of parallel systems. Both can be circumvented in certain cases, in particular by means of *partial order methods.* There are two prominent approaches:

– *Partial order reduction techniques* (see e.g. [God96,Pel93,Val89]), which try to exploit "diamond" properties to make savings in verification. They are based on a notion of equivalent executions, called Mazurkiewicz traces [DR95], and aim to cut redundant branches (and whole sub state spaces). Partial order reduction techniques have been applied with success notably to deadlock detection and model checking of certain (equivalence robust) linear time temporal properties [Pel93].

– *Unfolding based methods* (see e.g. [McM92,ERV96,ER99]): Rather than partially exploring an interleaving transition system, these methods directly construct partial order representations of executions by means of event structures [NPW81] or, equivalently, occurrence nets [Eng91]. Instead of computing successor states, the unfolding approach is based on computing possible event extensions. Using an *adequate order* among events, a *complete finite prefix* of the set of all events is defined and can be computed.

Recently, a new partial order reduction technique has been introduced specifically for the verification of local properties. Such properties depend on a single component of the system so that one should be able to identify equivalent classes of global states and tackle the state explosion while checking their possible reachability. The *Local First Search* approach [NHZL01] gives a combinatorial criterion that shows how to explore a reduced and yet locally complete subset of states in an efficient way. Based on observations linked to the Strahler number of trees [Str52], this new technique characterizes concurrent executions which may be cut off from the exploration while checking *any* local property.

While the first practical experiments indicate a very strong reduction potential for Local First Search in the detection of counter-examples (finding paths leading to local states), the approach of the method to take a part of the past into account when comparing states leads to a costly need to explore certain states more than once (state duplication problem). In practice, this means that the original LFS cannot be used to prove the *absence* of a state, because state copying blows up the explored state space more than the reduction criterion reduces it.

In this paper, we solve the state duplication problem that appears in the original LFS method. Motivated by a strong relationship between Mazurkiewicz traces and event structures [Bed87,NW95], we apply the technique of adequate orders from event structures to trace systems in order to define and construct a *locally complete finite unfolding*.

An essential technical difference between our approach and the classical computation of the complete finite prefix is that the latter uses an event structure as essential data structure whereas the LFS based approach computes on configurations (traces). More precisely, we define a subset of configurations that respect the LFS criterion (which contains all prime configurations). The use of an adequate order [ERV96] gives us an additional *cutoff* criterion that allows us to avoid multiple explorations of the same state.

Moreover, the complicated computation of *possible extensions* for the construction of the complete finite prefix (see [KK01 for an extended discussion) is fully avoided by our approach, at the price of a bigger result (but, as explained in [NHZL01], not necessarily higher computational cost). In addition, it relies solely on abstract characteristics of concurrency within the system, not on a particular representation like Petri nets. In short, it allows *black box unfolding*.

The paper is structured as follows: In Section 2, we introduce the technical framework for the presentation, notably Mazurkiewicz traces, asynchronous transition systems and an unfolding semantics of asynchronous transition systems into trace systems. In Section 3, we formalize the notion of a local property in the context of asynchronous transition systems. In Section 4, we rephrase the main theorem of [NHZL01] in terms of traces and give a summary of its application to the verification of local properties. The main contribution is in Section 5, the development of a *locally complete finite subsystem* and the proof of its completeness for the local reachability problem. In Section 6, we give an actual algorithm for the computation of the locally complete finite subsystem and discuss first experimental results. In Section 7, we conclude and give an outlook for the continuation of this line of research.

2 Basics

In this section, we develop the formal framework for the description of our method.

The description of parallelism in this work is based on Mazurkiewicz trace theory [DR95], of which we recall the notions important to our work. The framework is thus kept as a level of generality so as to apply to a wide variety of system descriptions, not just Petri nets or products of automata. For further motivating examples of this choice, see [NHZL01].

Traces and partial orders. In this paper, we fix a finite alphabet Σ together with an *independence relation* $\parallel \subseteq \Sigma \times \Sigma$ which is symmetric and irreflexive. Intuitively, this relation represents concurrency between actions occurring on distinct processes in a distributed system. The *trace equivalence* associated to the independence alphabet $(\Sigma, \parallel)$ is the least congruence $\sim$ over $\Sigma^\star$ such that $ab \sim ba$ for any pair of independent actions $a \parallel b$. A *trace* $[u]$ is the equivalence class of a word $u \in \Sigma^\star$. We denote by $\mathbb{M}(\Sigma, \parallel)$ the set of all traces w.r.t. $(\Sigma, \parallel)$. Traces are partially ordered according to the *prefix relation* defined as follows: We put $[u] \preccurlyeq [v]$ whenever there exists a word $z \in \Sigma^\star$ such that $u.z \sim v$.

It is a basic observation of Mazurkiewicz trace theory that traces can be viewed as partial orders of events labelled by actions in Σ. We shall here often focus on the number of maximal events in a trace seen as a labelled partial order. For simplification, we can formalize this as follows:

Definition 1. *For a trace $[w] \in \mathbb{M}(\Sigma, \parallel)$, the subset of last actions $Last([w])$ consists of all actions that can appear at the end of some sequential view of $[w]$; i.e. $Last([w]) = \{a \in \Sigma \mid \exists v \in \Sigma^\star, v.a \in [w]\}$. Further, the* span *$\#_{Last}([w])$ is the number of last actions of $[w]$: $\#_{Last}([w]) = |Last([w])|$.*

It is clear that $Last([w])$ consists of pairwise independent actions. Now a key notion for our development concerns *prime traces.* The latter admit a single maximal event.

Definition 2. *A trace $[w]$ with $\#_{Last}([w]) = 1$ is a* prime trace.

In other words, a trace $[w]$ is prime if, and only if, for all words v_1, v_2 and all actions a_1 and a_2, $v_1.a_1 \sim w \sim v_2.a_2$ implies $a_1 = a_2$.

Asynchronous transition systems. A transition system is a triple $T = (S, \rightarrow, s_0)$ with S a set of states, $s_0 \in S$ the initial state, and $\rightarrow \subseteq S \times \Sigma \times S$ a transition relation. In this paper, we require that transition systems are *deterministic*[1], i.e. $\rightarrow$ is a partial function from $S \times \Sigma$ to S. The *language L(T)* of execution sequences of a transition system T is the set of words $w = a_1 a_2 \dots a_n$ of $\Sigma^\star$ such that there exist states $s_i \in S$, $i = 0, \dots, n$ such that $s_0 \overset{a_1}{\rightarrow} s_1 \overset{a_1}{\rightarrow} \dots \overset{a_n}{\rightarrow} s_n$. Due to determinism, for any execution sequence $w = a_1 a_2 \dots a_n \in L(T)$ there exists a unique state $s \in S$ such that $s_0 \overset{a_1}{\rightarrow} s_1 \overset{a_1}{\rightarrow} \dots \overset{a_n}{\rightarrow} s_n = s$. We refer to this state as $\sigma(w)$. Of course, there may be several paths leading to the same state, so $\sigma(u) = \sigma(v)$ does not imply $u = v$.

When modelling concurrent machines by transition systems with independence relations, some *diamond properties* frequently appear [Bed87,God96, Shi85]:

Definition 3. *A transition system T is called* asynchronous[2] *w.r.t. the independence alphabet* $(\Sigma, \|)$ *if for all pairs of independent actions* $a \| b$*,*

ID: $s \overset{a}{\rightarrow} s_1 \overset{b}{\rightarrow} s_2$ *implies* $s \overset{b}{\rightarrow} s_1' \overset{a}{\rightarrow} s_2$ *for some state* s_1' [Independent Diamond]

FD: $s \overset{a}{\rightarrow} s_1$ *and* $s \overset{b}{\rightarrow} s_1'$ *implies* $s_1 \overset{b}{\rightarrow} s_2$ *for some state* s_2 [Forward Diamond]

Axioms ID and FD formalise an intuitive notion of independence: If two independent actions can occur one immediately after the other then they can occur in the opposite order (ID); moreover if two independent actions can occur in a common state, the occurrence of one of them cannot rule out the other one (FD). We remark also that if $u \in L(T)$ and $u \sim u'$ then $u' \in L(T)$ and $\sigma(u) = \sigma(u')$. Therefore we extend the map σ from words in $L(T)$ to the traces in $L(T)/\!\sim$ as follows: For all $u \in L(T)$, we denote by $\sigma([u])$ the state $\sigma(u)$.

Many examples of independence relations in various modeling frameworks exist: In process algebras, dependency results from communication over shared channels; in Petri nets, transitions sharing places in the presets or postsets may be dependent. In [Pel93], it is pointed out that independence (in use for partial order reduction) need not have concurrency as only source. For instance, two operations “X:=X+1” and “X:=X+2” do also satisfy the diamond properties and can hence be considered independent, although they touch the same variable. In contrast, “X:=X+1” and “X:=X*2” will not satisfy these properties.

Unfoldings. Typically, an asynchronous transition system T is an abstraction for a 1-safe Petri net or a synchronized product of automata; it describes the

[1] By introducing new action names, we can transform a non-deterministic system into a deterministic one. Such a transformation doesn't modify the properties (mainly the reachability problem) we are interested in.

[2] This naming in the literature is a potential source of confusion: Asynchronous here refers to the independent progress of the components of a parallel system, not to the communication discipline. In fact, our examples use synchronous communication.

behaviour and the global states (or markings) reached by its components. Although it has usually finitely many states, one aims at avoiding the exploration of all these states or all its execution sequences. For this, we shall construct a *representative part* of its unfolding — which is also called trace system.

Definition 4 (Trace system). *Let $T = (S, \to, s_0)$ be an asynchronous transition system w.r.t. $(\Sigma, \|)$. Then the* trace system *of T is the transition system $\mathcal{TS}(T)$ whose states are the traces associated to an execution sequence, with the empty trace $[\varepsilon]$ as initial state and such that the transition relation is $\to = \{([w], a, [w.a]) \mid w.a \in L(T)\}$.*

Thus, we have $[u] \xrightarrow{a} [v]$ in $\mathcal{TS}(T)$ iff $u.a$ is an execution sequence of T and $u.a \sim v$. It follows that $\mathcal{TS}(T)$ is an acyclic (deterministic) transition system which is asynchronous w.r.t. $(\Sigma, \|)$. Furthermore $L(T) = L(\mathcal{TS}(T))$ and the unfolding of $\mathcal{TS}(T)$ is $\mathcal{TS}(T)$ itself. We observe also that the map σ from the traces in $L(T)/\sim$ to the states S induces a homomorphism from $\mathcal{TS}(T)$ to T, since $[u] \xrightarrow{a} [v]$ in $\mathcal{TS}(T)$ implies $\sigma(u) \xrightarrow{a} \sigma(v)$ in T. This map is surjective if T contains no unreachable states. The fact that the original transition system T is a homomorphic image of its trace system justifies the use of the trace system as a semantic model.

In [Bed87], the concurrent executions of an asynchronous transition system are described by a prime event structure. Generalizing the unfolding of 1-safe nets in occurrence nets [NPW81,Eng91], there is a one-to-one correspondence between the set of traces $L(T)/\sim$ and the finite configurations of the associated prime event structure. In this view, $\mathcal{TS}(T)$ appears as an abstract representation of the configuration structure of T. A key observation in [Bed87, chap. 5] is that one can identify the events of the underlying unfolding as the configurations having a single predecessor, that is, the prime traces. For this reason, prime traces are good candidates to check *local properties*, as explained in the next section.

3 Local Properties

In this section, we consider an asynchronous transition system $T = (S, \to, s_0)$ w.r.t. the independence alphabet $(\Sigma, \|)$.

Definition 5. *We say that $(\Sigma, \|)$ has* parallel degree *m if m is the maximal number of pairwise independent actions in Σ, i.e.*

$$m = \max\{|A| \mid A \subseteq \Sigma \text{ and } a, b \in A, a \neq b \implies a \| b\}.$$

For Petri nets, m corresponds to the maximal number of transitions that can be fired concurrently. In a system of processes, m is an upper bound for the number of sequential components that can work in parallel.

Definition 6 (Local properties). *For a given set $P \subseteq S$ (called* property*), the set of* visible actions *$V_P \subseteq \Sigma$ is the set of all actions $a \in \Sigma$ such that there* *exist $s_1, s_2 \in S$ with $(s_1, a, s_2) \in \rightarrow$ and either $s_1 \in P$ and $s_2 \notin P$, or $s_2 \in P$ and $s_1 \notin P$.*

A property P has parallel degree *m if the restricted independence alphabet $(V_P, \| \cap (V_P \times V_P))$ has parallel degree m. A property is called* local *if it has parallel degree 1.*

The idea of visible actions [Pel93] is that of actions that may affect a property of interest. The naming of local properties is due to the typical case of properties of one process in a network: These are properties that depend only on the local state of the process in question, and this state only changes by transitions involving this process, thus mutually dependent transitions.

Proposition 1. *A property $P \subseteq S$ of parallel degree m is reachable (i.e. there exists $[u]$ such that $\sigma([u]) \in P$) if, and only if, there exists an execution sequence $w = a_1 \dots a_k$ leading to a state $\sigma(w) \in P$ with $\#_{Last}([w]) \leqslant m$, where all last actions in the trace $[w]$ are visible actions.*

Proof. Consider an execution sequence $w = a_1 \dots a_n$ with $s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \dots \xrightarrow{a_n} s_n$, $s_n \in P$ and moreover $|w|$ *minimal*, i.e. $|w| \leqslant |w'|$ for any other sequence w' leading to a state $s' \in P$. Then each last action of $[w]$ is a *visible* action. Since these actions are pairwise independent, $\#_{Last}([w])$ is at most equal to the parallel degree of P. ■

In the sequel, we aim at checking whether a given *local* property is reachable in T. Then, Proposition 1 ensures that we need only to explore the *prime* traces $[w]$ of the unfolding $\mathcal{TS}(T)$ and check whether $\sigma([w]) \in P$. Since there are in general infinitely many prime traces, we will need a criterion to explore a finite part of the unfolding, only. In the next section, we describe an efficient strategy to construct prime traces.

4 Local First Search

Since traces are equivalence classes of sequential executions, there are generally several ways to reach the state $\sigma([w])$ of a trace $[w]$. Among all the sequential views $v \in [w]$, the LFS approach tries to minimize the number of last actions seen along v. More formally, the *beam* of a word $v = a_1 \dots a_k \in \Sigma^\star$ is the maximal span $\#_{Last}([u])$ among the traces $[u] = [a_1 \dots a_j]$ with $j \leqslant k$. Then the *LFS-number* of a trace $[w]$ is the minimal beam of $v \in [w]$. Equivalently, we have:

Definition 7 (LFS-number). *The* LFS-number *of a trace $[w]$ is the least number l such that there exists a representative $v = a_1 \dots a_k \in [w]$ such that for each $1 \leqslant j \leqslant k$ we have $\#_{Last}([a_1 \dots a_j]) \leqslant l$.*

As explained above, we aim at exploring prime traces. For this, the LFS approach exhibits an upper bound for the LFS-numbers of prime traces. This is based on a combinatorial aspect of the independence alphabet. For any action

$c \in \Sigma$, Σ_c denotes the subset of actions $b \in \Sigma$ which are dependent with c. Then the *communication degree of* $(\Sigma, \|)$ is the maximal parallel degree of the restricted independence alphabet $(\Sigma_c, \| \cap (\Sigma_c \times \Sigma_c))$ when $c \in \Sigma$. In other words:

Definition 8. *The* communication degree *of* $(\Sigma, \|)$ *is the maximal number* n *of pairwise independent actions which all depend on a common action, i.e.* $n = \max\{|B| \mid B \subseteq \Sigma, \exists c \in \Sigma, (\forall b \in B,\ c \not\| b\})$ *and* $(\forall b, b' \in B, b \neq b' \implies b \| b')\}$.

Obviously the communication degree n is smaller than the parallel degree m of $(\Sigma, \|)$. Actually, in many formalisms for concurrent systems, we observe that n tends to be small compared to m. For instance, many process algebras restrict communication to pairs of *send* and *receive* actions, leading to a communication degree 2. This bound holds for message sequence charts as well. The dependency relations resulting from Petri-nets are bounded by the number of presets and postsets of transitions, which are often very small compared to the size of the entire net.

The main technical result of [NHZL01] can be summarized as follows:

Theorem 1 (LFS-bound). *For all* prime *traces* $[w] \in \mathbb{M}(\Sigma, \|)$*, the LFS-number of* $[w]$ *is at most* $\lfloor (n-1)\log_n(m) \rfloor + 1$*, where* m *and* n *are respectively the parallel degree and the communication degree of* $(\Sigma, \|)$.

We will refer to $\lfloor (n-1)\log_n(m) \rfloor + 1$ as the *LFS-bound* of $(\Sigma, \|)$. Note that for the case of $n = 2$, the LFS-bound simplifies to $\lfloor \log_2(m) \rfloor + 1$ which compares favorably to the naive upper bound m.

Let us discuss the meaning of this theorem concerning the trace system of an asynchronous transition system and how this was exploited in [NHZL01] for the original version of local first search.

The LFS-number yields a partition of the states in the trace system, from 0 (for $[\varepsilon]$) potentially up to m. Then the theorem implies that the prime traces are reachable from the initial state via paths avoiding traces with a span exceeding the LFS-bound, and thus by traces with LFS-number smaller than this bound. On the other hand, *local properties* can be analyzed with attention restricted to prime traces.

The original LFS then exploited these facts via the construction of an extended transition system with states (s, M), where s is a state of the original transition system and $M \subseteq \Sigma$ is the subset of last actions of a trace $[w]$ leading to s. However, the original LFS may have to explore a single state s several times (with different sets of last actions). While the experiments showed that the strategy LFS gives good results if a state searched for exists with a low LFS-number, the construction of the state space up to the LFS-bound typically produced state spaces exceeding the size of the original transition system, i.e. with growing LFS-number the size of the class of states grows quickly and the state doubling phenomenon produces more overhead than can be possibly avoided by the reduction. The aim of this work is to combine notions from LFS with notions known from McMillan unfoldings to overcome this state copying problem.

5 Locally Complete Finite Subsystem of a Trace System

Similar to the *complete finite prefix* of the maximal branching process of a Petri net, we now want to define a subsystem of the trace system with the following properties:

- It should be computable and be no bigger than the state space of the original asynchronous transition system.
- It should be complete in the sense that it preserves reachability of local properties with respect to the unreduced trace system.
- It should not contain traces with LFS-number exceeding the LFS-bound.

The construction is modular and relies on three steps:

- The first step is to define a reduced trace system according to some reduction strategy, that preserves prime traces and thus reachability of local properties. In the present work, we will use the LFS-bound for this purpose but other reductions may work as well. The resulting reduced trace system is typically still infinite.
- The definition of an *adequate order* on the states of the (reduced) trace system that leads the construction of a finite prefix of the reduced trace system (in this particular order).
- The definition of a *cutoff criterion*, which will eliminate states explored "before" according to the adequate order.

In the rest of this paper, we consider a finite asynchronous transition system $T = (S, \rightarrow, s_0)$.

Definition 9. *A (reachable)* subsystem *of the trace system $\mathcal{TS}(T)$ is a subset of traces $R \subseteq L(T)/{\sim}$ such that for every trace $[w] \in R$ there exists a (predecessor) trace $[v] \in R$ with $[w] = [v.a]$ for some $a \in \Sigma$.*

In other words, the restriction of the trace system $\mathcal{TS}(T)$ to a subsystem R provides us with a new transition system whose states are reachable from the initial empty trace. Since we want to avoid a complete construction of $\mathcal{TS}(T)$ we will actually build such a subsystem only. However, this subsystem must keep enough information in order to check local properties.

Definition 10. *A subsystem R is* locally complete *if for all local properties P and for all traces $[w] \in L(T)/{\sim}$ such that $\sigma([w]) \in P$ there exists $[w'] \in R$ such that $\sigma([w']) \in P$.*

Thus, given a locally complete subsystem R, a local property P is reachable in the asynchronous transition system T if, and only if, there is a trace $[w] \in R$ that satisfies P, i.e. $\sigma([w]) \in P$. Therefore, what we need essentially is to build a *finite* and locally complete subsystem of the unfolding $\mathcal{TS}(T)$.

A basic method to stop the construction of traces while exploring a trace system is to fix a set F of "forbidden" traces. Clearly, for any subset $F \subseteq \mathbb{M}(\Sigma, \|)$, there exists a *largest* subsystem R_F that contains no trace of F, i.e.

$R_F \cap F = \emptyset$. This subsystem is called *the subsystem that forbids* F. Note, that conceptually forbidding traces can be done hierarchically, because forbidding $F' \subseteq R_F$ in R_F can be understood to yield the subsystem $R_{F \cup F'}$.

Definition 11. *We call* LFS-excessive *any trace whose LFS-number is greater than the LFS-bound of* $(\Sigma, \|)$. *The* LFS-subsystem *of* $\mathcal{TS}(T)$ *is the subset of all traces that are not LFS-excessive.*

One can easily show that the subsystem R_{LFS} that forbids the set LFS-excessive traces is precisely the *LFS-subsystem* of $\mathcal{TS}(T)$.

Due to Theorem 1, *the LFS-subsystem of* $\mathcal{TS}(T)$ *contains all prime traces.* It is therefore locally complete (Prop. 1). However it is in general still infinite. So we need to forbid some more traces to get a finite subsystem. A key ingredient we borrow from the theory of Petri nets unfoldings is that of an adequate order [ERV96] on traces:

Definition 12 (Adequate order). *A partial order* $\sqsubseteq$ *on the whole set of traces* $\mathbb{M}(\Sigma, \|)$ *is called* adequate *if*

(Ad_1) *it is well-founded;*
(Ad_2) *it refines the prefix order, i.e.* $[u] \preccurlyeq [v]$ *implies* $[u] \sqsubseteq [v]$*;*
(Ad_3) *it is a right congruence, i.e.* $[u] \sqsubseteq [v]$ *implies* $[u.z] \sqsubseteq [v.z]$ *for any* $z \in \Sigma^\star$.

Noteworthy, the last condition implies that if $[u] \sqsubset [v]$ then $[u.z] \sqsubset [v.z]$ for any word $z \in \Sigma^\star$. We will use some adequate order to specify how traces are explored and which additional traces should be forbidden. We will discuss some examples of adequate orders in the next section.

Definition 13 (Cutoff trace with respect to a subsystem). *Given an adequate order* $\sqsubseteq$ *on traces and a subsystem* R*, we say that a trace* $[v] \in R$ *is a* cutoff trace with respect to R *if there exists a trace* $[u] \in R$ *such that* $[u] \sqsubset [v]$ *and* $\sigma(u) = \sigma(v)$.

Now we can state our main theorem.

Theorem 2. *For all adequate orders, the subsystem* R *that forbids both the LFS-excessive traces and the cutoff traces with respect to the LFS-subsystem, is locally complete and finite.*

Proof. We first show by contradiction that R is locally complete. Let P be a local property of T. Let $[w] \in L(T)/\sim$ be a trace such that $\sigma([w]) \in P$ but for all traces $[w']$ in the subsystem R it holds that $\sigma([w']) \notin P$. Since $\sqsubseteq$ is well-founded (Ad_1), we can choose $[w]$ to be $\sqsubseteq$-minimal among the traces $[v]$ with $\sigma([v]) \in P$.

We observe first that $[w]$ is a prime trace. Otherwise we would have $v_1.a_1 \sim w \sim v_2.a_2$ with $a_1 \| a_2$. Since P is local, either a_1 or a_2 is invisible for P and $[v_1]$ or $[v_2]$ satisfies P. Since $\sqsubseteq$ refines the prefix order of traces (Ad_2), $[w]$ would not be $\sqsubseteq$-minimal.

Consequently, $[w]$ is a trace of the LFS-subsystem of T (Theorem 1). The reason for it not to make part of the subsystem R must thus rely on an ancestor

$[v]$ of $[w]$ in the LFS-subsystem that is a cutoff trace. So $[w] = [v.v']$. The fact that $[v]$ is a cutoff trace implies the existence of some trace $[u] \sqsubset [v]$ in the LFS-subsystem with $\sigma(u) = \sigma(v)$. Consequently, $[u.v'] \in L(T)/\!\sim$ and Ad_3 yields $[u.v'] \sqsubset [v.v']$. Since $\sigma(u.v') = \sigma(v.v') = \sigma(w)$, $[u.v']$ satisfies P and $[u.v'] \sqsubset [w]$. This contradicts the assumption that $[w]$ is $\sqsubseteq$-minimal.

We now prove that R is finite. Consider the subsystem R' that consists of all traces $[u]$ with $|u| \leqslant |S|$. Clearly R' is finite. We need just to check that $R \subseteq R'$, by contradiction. Assume $[u] \in R \setminus R'$. For all linear extensions $a_1 \dots a_k \in [u]$, we have $\sigma(a_1 \dots a_i) = \sigma(a_1 \dots a_j)$ for some $1 \leqslant i < j \leqslant k$ because $k > |S|$. Since R forbids cutoff traces, $[u]$ is not reachable in R. ∎

Corollary 1. *If the adequate order $\sqsubseteq$ is* total *then* $|R| \leqslant |S|$.

Proof. By contradiction, assume $|R| > |S|$. Then there are two distinct traces $[u], [v] \in R$ such that $\sigma(u) = \sigma(v)$. We may assume $[u] \sqsubset [v]$ because $\sqsubseteq$ is total. Then $[v]$ is a cutoff trace hence $[v] \notin R$. ∎

6 Algorithmics

In this section, we show how to apply Theorem 2 to obtain a reachability algorithm for local properties.

Theorem 2 relies on an adequate order under which it gives a definition of a locally complete finite subsystem of the trace system, but it does not immediately give an algorithm for computing this subsystem. The crucial step towards an algorithm is the choice of an adequate order with good algorithmic properties. Indeed, so far we did not even require the adequate order to be decidable.

The literature proposes a number of adequate orders that have been used in implementations of McMillan's unfolding method:

1. McMillan's original order was induced by $|w|$, the "number of events" in a trace $[w]$. Let us call this order of $\mathbb{M}(\Sigma, \|)$ the *length order*. Its advantage is its simplicity and low (logarithmic) complexity and that it corresponds closely to breadth first search.
2. Of course, the prefix order itself also is adequate, as is mentioned in some sources. It is however of no practical interest.
3. In [ERV96], an adequate order based on a lexicographic order on the sequences of sets of labels of the Foata normal form of traces was proposed. An important aspect of this order is that it is total. The advantage of a total order is that it results in a subsystem with at most one trace for each state of the unreduced transition system (Corollary 1). This is in contrast to the simpler order of McMillan, which can result in subsystems exponentially larger than the transition system. The price to be paid is that the order based on Foata normal form takes a worst case linear time effort to compute.
4. In [ER99], another *total* adequate order was proposed that – while also having worst case linear complexity – has algorithmic advantages and seems to be faster on average than the one based on the Foata normal form. Since this is the order we currently use, we will introduce it below.

Apart of the complexity of decision and the discrimination (totality), there is another important aspect for the choice of an adequate order: While Theorem 2 uses only the fact that the adequate is well-founded, an algorithm should intuitively construct the subsystem "in that order", i.e. the adequate order should permit to enumerate the traces, otherwise said be of ordinal type ω.

The total orders mentioned above have precisely this property, as they refine the length order and for a given length there is only a finite number of traces. Another interesting property of adequate orders that refine the length order is that they are compatible with *breadth first search*.

A particular adequate order. In order to introduce the adequate order of [ER99], we rely on a concrete representation of independence [Zie87]: Given a finite set of *locations Loc*, a *distributed alphabet* is a family $(\Sigma_l)_{l\in Loc}$ of (finite) local alphabets (which may overlap).

A distributed alphabet $(\Sigma_l)_{l\in Loc}$ induces an independence alphabet $(\Sigma, \|)$, where $\Sigma := \bigcup_{l\in Loc} \Sigma_l$ and $a \parallel b$ if there does *not* exist $l \in Loc$ such that both $a \in \Sigma_l$ and $b \in \Sigma_l$. Conversely, it is easy to see that for any independence alphabet $(\Sigma, \|)$ there exists a distributed alphabet $(\Sigma_l)_{l\in Loc}$ inducing it[3].

From now on, let $(\Sigma_l)_{l\in Loc}$ be a fixed distributed alphabet and $(\Sigma, \|)$ the induced independence alphabet. Moreover, let $\pi_l : \Sigma^\star \to \Sigma_l^\star$ denote the *projecting homomorphism* with $\pi_l(a) = a$ for $a \in \Sigma_l$ and $\pi_l(a) = \varepsilon$ for $a \notin \Sigma_l$.

It is a well known fact that for $u, v \in \Sigma^\star$ we have $u \sim v$ iff $\pi_l(u) = \pi_l(v)$ for all locations $l \in Loc$ [DR95]. This allows us to call $(\pi_l(w))_{l\in Loc}$ the *distributed representation* of the trace $[w]$. This is a very useful data structure for the manipulation of traces as it allows efficient tests for $\sim$, but also easy (componentwise) concatenation.

Definition 14 (Esparza-Römer order).
Let $\leqslant$ denote a total order on Σ and $Loc = \{1, \ldots, h\}$ be an enumeration of the locations. Let $\leqslant_{lex}$ denote the induced (total) lexicographic order on $\Sigma^\star$ induced by $(\Sigma, \leqslant)$.

For $l \in Loc$ and $u, v \in \Sigma_l^\star$, let $u \sqsubset_l v$ iff $|u| < |v|$, or $|u| = |v|$ and $u <_{lex} v$.

The Esparza-Römer order $\sqsubset_{ER}$ on traces is defined by $[u] \sqsubset_{ER} [v]$ if either $|u| < |v|$, or $|u| = |v|$ *and there exists $l \in Loc$ such that $\pi_l(u) \sqsubset_l \pi_l(v)$ and for all l' with $1 \leqslant l' < l$ we have $\pi_{l'}(u) = \pi_{l'}(v)$.*

It is easy to verify that $\sqsubseteq_{ER}$ is an adequate order and moreover total. By definition, it refines the length order. For an extended discussion of this order, see [ER99].

An algorithm. Based on a total *adequate order* $\sqsubseteq$ *that refines the length order* as parameter, we give an abstract algorithm for computing the associate locally complete finite subsystem (c.f. Algorithm 1). It computes the non-cutoff traces

[3] It is sufficient to take a collection of cliques $(\Sigma_l, \Sigma_l \times \Sigma_l)$ in the graph $(\Sigma, \not\parallel)$ such that they cover the graph.

Algorithm 1 Computation of a finite locally complete subsystem

```
Require: $|u| < |v|$ implies $[u] \sqsubset [v]$.
  Table $\leftarrow \{(s_0, [\varepsilon])\}$
  Previous_Level $\leftarrow \{(s_0, [\varepsilon])\}$
  while Previous_Level $\neq \emptyset$ do
    Current_Level $\leftarrow \emptyset$
    for all $(s, [u]) \in$ Previous_Level do
      for all $a \in \Sigma$, $s' \in S$ such that $s \xrightarrow{a} s'$ do
        if $\#_{Last}([u.a]) \leqslant$ LFS-bound then
          if $(s', [v]) \in$ Table then
            if $[u.a] \sqsubset [v]$ then {We have $|u.a| = |v|$ and $(s', [v]) \in$ Current_Level}
              Table $\leftarrow$ (Table $\setminus \{(s', [v])\}) \cup \{(s', [u.a])\}$
              Current_Level $\leftarrow$ (Current_Level $\setminus \{(s', [v])\}) \cup \{(s', [u.a])\}$
            end if
          else
            Table $\leftarrow$ Table $\cup \{(s', [u.a])\}$
            Current_Level $\leftarrow$ Current_Level $\cup \{(s', [u.a])\}$
          end if
        end if
      end for
    end for
    Previous_Level $\leftarrow$ Current_Level
  end while
  Return Table
```

level by level (in terms of the number of events) and stores them together with the corresponding state in a set (in practice, in a hash table). Once a level is empty, we stop.

While many algorithmic improvements on the level of detail are possible, we integrate one explicitly into the description of the algorithm: On a given level, we do not explore the traces in the adequate order but in any order. As a consequence, we may have to remove certain states on a given level if we find the same state on the same level with a smaller trace. The advantage is that we only have to test for the adequate order whenever we reach the same state several times, but no (inefficient) enumeration of the traces of one level is required.

First experimental results. Currently, only an early prototype written in Caml exists, which does neither allow insights on the runtime of the procedure, nor the exploration of big examples. However, it allows to measure the size of the state spaces explored and thus it gives hints on the *potential* of the *unfolding LFS* method, also in comparison to the original LFS procedure [NHZL01].

We consider two well known series of examples, that are known to have relative small Petri net unfoldings, the *asynchronous n-token buffer* and the *dining philosophers.* We have not tried to verify any properties on these examples, we only measure the number of states explored with different methods.

Table 1. Experimental results: Number of states with different methods

asynchronous buffers (m)	1	2	3	4	5	6	7	8	12	15	32
no reduction	2	4	8	16	32	64	128	256	4096	32768	2^{32}
original LFS $\#_{\text{Last}} \leqslant \lfloor \log_2 m \rfloor + 1$	1	3	8	18	43	100	22	830	24475	165993	–
unfolding LFS $\#_{\textbf{Last}} \leqslant \lfloor \log_2 m \rfloor + 1$	1	3	6	12	24	48	95	192	2829	17006	–
original LFS $\#_{\text{Last}} \leqslant 2$	1	3	8	16	30	53	88	138	548	1160	13638
unfolding LFS $\#_{\textbf{Last}} \leqslant 2$	1	3	6	12	23	42	72	116	492	1068	13172
original LFS $\#_{\text{Last}} \leqslant 3$	1	3	8	18	43	100	222	484	4850	17406	–
unfolding LFS $\#_{\textbf{Last}} \leqslant 3$	1	3	6	12	24	48	95	184	1865	7359	–

philosophers (m)	1	2	3	4	5	6	7	8	9	10	12
no reduction	2	8	26	80	242	728	2186	6560	19682	59048	531440
original LFS $\#_{\text{Last}} \leqslant \lfloor \log_2 m \rfloor + 1$	2	8	37	202	1006	4195	13981	206421	–	–	–
unfolding LFS $\#_{\textbf{Last}} \leqslant \lfloor \log_2 m \rfloor + 1$	2	8	25	79	226	598	1450	5347	13372	31286	142295
original LFS $\#_{\text{Last}} \leqslant 2$	2	8	37	129	343	738	1363	2270	3511	5138	9758
unfolding LFS $\#_{\textbf{Last}} \leqslant 2$	2	8	25	67	156	319	582	969	1504	2211	4237
original LFS $\#_{\text{Last}} \leqslant 3$	2	8	40	202	1006	4195	13981	38759	–	–	–
unfolding LFS $\#_{\textbf{Last}} \leqslant 3$	2	8	25	79	226	598	1450	3229	6655	12806	39875

The upper parts of the tables expose the number of buffer cells (philosophers respectively), the size of the state space without reduction, then the sizes of state spaces explored with the original LFS procedure of [NHZL01] and for the *unfolding LFS procedure* of this work.

As explained in [NHZL01], the bound $\lfloor \log_2 m \rfloor + 1$ yields a blowup rather than a reduction for the original LFS procedure, as the reduction gains are weaker than the blowup induced by the need to separate states with different sets of maximal events. However, the theoretical bound of $\lfloor \log_2 m \rfloor + 1$ is a sufficient upper bound for a worst case. For our examples, the actual LFS-number of any prime trace is bounded by 2, so that LFS applied with bound 2 is already exhaustive for local properties. Moreover, we have recently established dynamic criteria allowing to determine the absence of any prime trace with an LFS-number exceeding a certain level [LZ02]. For our examples, exploration up to LFS-bound 3 is sufficient to check the absence of local traces with a higher LFS-number.

Hence, we also show the numbers of states explored with bounds 2 and 3 in the lower parts of the tables. With exploration up to these levels, original LFS (with some heuristic improvements explained in [NHZL01]) already gives very good reductions.

The results for unfolding LFS show additional reductions compared to original LFS. As with original LFS, we observe a rapid explosion of the number of states with rising LFS-number. For the examples considered, the theoretical bound of $\lfloor \log_2 m \rfloor + 1$ does give a reduction, but initially not a strong one: Apparently, for a small number of philosphers, a significant fraction of all states have relatively low LFS-numbers.

However, the reduction increases with a growing number of philosophers and thus a growing difference[4] between the LFS-bound and the degree of parallelism (for 12 philosophers, the reduction is already at 75%). Although it is difficult to predict reductions for a big number of philosophers, they can be expected to be stronger and stronger. Indeed, LFS is designed to give reductions for a large number of components.

On the other hand, the reduction using the dynamic LFS-bound (2 and 3, see discussion above) is enormous on the whole. Even for the case 2 (sufficient in this case for the detection of counter-examples) the reduction significantly improves over the already impressing numbers for the original LFS.

Summarizing, we consider the additional reduction a sufficient justification for further exploration of our approach. Of course, further experiments are necessary for a full assessment of the method.

7 Conclusions

In this work, we have conceived a hybrid reduction method combining Local First Search [NHZL01] with important notions from Petri net unfoldings [McM92, ERV96]. This *unfolding LFS* eliminates an essential drawback of the original LFS, the need to explore certain states several times. The result is a method for searching and proving local properties with reductions significantly improving over Local First Search. In contrast to the Petri net unfolding approach, LFS only relies on the dependency relation and is thus *black box*, applicable to any kind of transition system with diamond properties.

On the practical side, there is a lot of work to be done: Apart of improving the complexity of our prototype implementation, heuristic improvements for further cropping of the state space should result in a wider applicability.

An important question concerns potential combination with other partial order reduction methods: Indeed, LFS (original or unfolding based) does eliminate diamonds of big dimensions but not the diamonds of small dimensions (below the LFS-bound), so the reduced systems still expose some potential for further partial order reductions. First steps in this direction are explored in [LZ02].

Acknowledgements. We thank Denis Lugiez for discussions, valuable comments, and human support during the preparation of this work.

References

[Bed87] M. Bednarczyk, *Categories of asynchronous systems*, Ph.D. thesis, Computer Science, University of Sussex, Brighton, 1987.

[DR95] V. Diekert and G. Rozenberg (eds.), *The book of traces*, World Scientific, 1995.

[4] Note, that the reduction percentage may temporarily increase at jumps in the LFS-bound at powers of 2 in the degree of parallelism, from 7 to 8, for example.

[Eng91] J. Engelfriet, *Branching processes of Petri nets*, Acta Informatica **28** (1991), no. 6, pp. 575–591.

[ER99] J. Esparza and S. Römer, *An unfolding algorithm for synchronous products of transition systems*, International Conference on Concurrency Theory (CONCUR), LNCS 1664, 1999, invited paper, pp. 2–20.

[ERV96] J. Esparza, S. Römer, and W. Vogler, *An improvement of McMillan's unfolding algorithm*, TACAS, LNCS 1055, 1996, pp. 87–106.

[God96] P. Godefroid, *Partial-order methods for the verification of concurrent systems: an approach to the state-explosion problem*, LNCS 1032, Springer-Verlag, 1996.

[Hol99] G.J. Holzmann, *The engineering of a model checker: the Gnu i-protocol case study revisited*, Proc. of the 6th Spin Workshop, LNCS, no. 1680, 1999.

[KK01] V. Khomenko and M. Koutny, *Towards an efficient algorithm for unfolding Petri nets*, International Conference on Concurrency Theory (CONCUR), LNCS 2154, 2001, pp. 366–381.

[LZ02] D. Lugiez, P. Niebert and S. Zennou, *Dynamic bounds and transition merging for local first search*, SPIN Workshop 2002, LNCS, 2002.

[McM92] K.L. McMillan, *Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits*, Computer Aided Verification (CAV), 1992, pp. 164–174.

[NHZL01] P. Niebert, M. Huhn, S. Zennou, and D. Lugiez, *Local first search – a new paradigm in partial order reductions*, International Conference on Concurrency Theory (CONCUR), LNCS 2154, 2001, pp. 396–410.

[NPW81] M. Nielsen, G. Plotkin, and G. Winskel, *Petri nets, event structures and domains, part I*, Theoretical Computer Science **13** (1981), no. 1, 85–108.

[NW95] M. Nielsen and G. Winskel, *Models for concurrency*, Handbook of Logic and the Foundations of Computer Science (S. Abramsky, Dov M. Gabbay, and T.S.E. Maibaum, eds.), vol. IV, Oxford Science Publications, Clarendon Press, 1995.

[Pel93] D. Peled, *All from one, one for all: On model checking using representatives*, International Conference on Computer Aided Verification (CAV), LNCS, vol. 697, 1993, pp. 409–423.

[Shi85] M. W. Shields, *Concurrent machines*, The Computer Journal **28** (1985), no. 5, 449–465.

[Str52] A.N. Strahler, *Hypsometric (area-altitude) analysis of erosonal topology*, Bull. Geol. Soc. of America **63** (1952), 1117–1142.

[Val89] A. Valmari, *Stubborn sets for reduced state space generation*, 10th International Conference on Application and Theory of Petri Nets, vol. 2, 1989, pp. 1–22.

[Zie87] Wi. Zielonka, *Notes on finite asynchronous automata*, R.A.I.R.O. – Informatique Théoretique et Applications **21** (1987), 99–135.