Deciding true concurrency equivalences on safe, finite nets

Lalita Jategaonkar *, Albert R. Meyer

AT&T Bell Laboratories, 1000 E. Warrenville Road, Naperville, IL 60566, USA
MIT Lab. for Computer Science, 545 Technology Square, NE43-315, Cambridge, MA 02139, USA

Abstract

We show that the pomset-trace equivalence problem for 1-safe, finite Petri nets is decidable; in fact it is complete for EXPSPACE. We also show that history-preserving bisimulation between such nets is complete for DEXPTIME. Our methods also yield tight complexity bounds for several other “true concurrency” and interleaving equivalences. The results are independent of the presence of hidden transitions.

1. Introduction

The computational complexity of the equivalence problem for nondeterministic finite-state automata under a variety of standard process semantics has been tightly characterized. In particular, Kannelakis and Smolka [13] have shown that trace equivalence and failure equivalence [4] are PSPACE-complete, while bisimulation [16] is PTIME-complete [1, 13]. It has been shown recently that these equivalence problems are exponentially harder for automata presented as finite “Mazurkiewicz nets” of synchronized state-machines [19]: namely, Rabinovich [18] and Mayer and Stockmeyer [15] have shown that trace equivalence and failure equivalence of these nets are EXPSPACE-complete, and Stockmeyer [20] has shown that bisimulation of these nets is DEXPTIME-complete.

The known results for “true” concurrency equivalences are much more limited. Vogler [27, 29] has shown the decidability of history-preserving bisimulation [2, 19, 23, 31, 27] and maximality-preserving bisimulation [7, 31] for finite 1-safe Petri nets; however, their complexity remained open. Decidability of such a basic true concurrency property as pomset-trace equivalence [23] appears not to have been known. (An ordinary trace is a linear sequence of visible actions; pomset-traces generalize these to multisets of actions partially ordered to reflect causality and concurrency.)

* Corresponding author. E-mail: lalita@research.att.com, meyer@theory.lcs.mit.edu
In contrast to trace equivalence, the decidability of pomset-trace equivalence for finite nets does not obviously reduce to equivalence of finite automata. The difficulty is that if a run of a net has a pomset-trace isomorphic to the pomset-trace of a run of another net, then whether a transition firable after one run yields the "same" pomset extension as a transition firable after the other run depends a priori on the entire pomset trace, which may be unboundedly large. Hence, instead of searching for a suitable equivalence relation on the finite set of net markings, one has to consider equivalence relations on a potentially infinite set of pomset traces and final markings.

A similar difficulty appears in deciding whether finite nets are history-preserving bisimilar, which Vogler [27, 29] overcomes by maintaining, instead of an entire pomset history, a partial order on the fixed set of places of the nets that reflects "most-recent" firings. We use a similar partial order, but instead of places, we find it technically smoother to keep track of the partial ordering between the most-recent firings of transitions. This idea leads to a decision procedure for pomset-trace equivalence, and a simple analysis of this procedure yields an \text{EXPSPACE} upper bound. The same approach also gives a \text{DUPTIME} decision procedure for history-preserving bisimulation.

Our lower bounds for these true concurrency equivalences follow easily from reductions from the corresponding interleaving equivalences, whose lower bounds in turn essentially follow from the results of Mayer and Stockmeyer [15, 20] and Rabinovich [18]. We thus obtain a tight bound of \text{EXPSPACE-completeness} for pomset-trace equivalence. Likewise, we obtain \text{DUPTIME-completeness} for history-preserving bisimulation and maximality-preserving bisimulation, settling questions left open by Vogler [27, 29].

Our methods also yield tight complexity bounds for several other true concurrency equivalences, summarized in Table 1. In particular, our \text{EXPSPACE-completeness} results for ST-traces and ST-failures [22, 24] solve problems left open by Vogler [30], who had earlier proved the decidability of these equivalences. Furthermore, our decidability results for pomset-bisimulation [3] and pomset-ST-bisimulation [31] settle questions alluded to by Vogler [26]. To keep this paper relatively self-contained, the definitions of all of these equivalences are included in this paper, with the exception of the interval pomset equivalences and the ST and pomset-ST equivalences. The reader is referred to [11, 28] and [22, 24, 31], respectively, for those definitions.

This paper is organized as follows. Section 2 describes our alternate characterization of pomset-trace equivalence, together with an \text{EXPSPACE} decision procedure. Similar analyses of history-preserving bisimulation and pomset bisimulation are given in Section 3, while Section 4 describes decision procedures for the other equivalences. Section 5 gives lower bounds for all these equivalences. A discussion of some open problems appears in Section 6.

\[1\] For expository purposes, we refer to bounds of the form \(2^{O(n^k)}\) for fixed \(k\) as exponential in \(n\). In the results presented here, \(k\) is at most 4.
Table 1
Complexity results for finite 1-safe Petri Nets

<table>
<thead>
<tr>
<th>Class</th>
<th>Equivalence</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traces</td>
<td>Traces</td>
<td>EXPSPACE-complete</td>
</tr>
<tr>
<td></td>
<td>Step-traces [12, 21, 23]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ST-traces [22, 24]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Interval-pomset-traces [11, 28]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pomset-traces [11, 23, 28]</td>
<td></td>
</tr>
<tr>
<td>Failures</td>
<td>Failures</td>
<td>EXPSPACE-complete</td>
</tr>
<tr>
<td>divergences</td>
<td>ST-failures [22, 24]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Interval-pomset-failures [11, 28]</td>
<td></td>
</tr>
<tr>
<td>Bisimulation</td>
<td>Bisimulation [16]</td>
<td>DEXPTIME-complete</td>
</tr>
<tr>
<td></td>
<td>Delay bisimulation [25]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Branching bisimulation [25]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Step-bisimulation [12, 23]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ST-bisimulation [22, 24]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>History-preserving bisimulation [2, 19, 23, 31, 27]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Maximality-preserving-bisimulation [7]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pomset-bisimulation [3]</td>
<td>DEXPTIME-hard and in EXPSPACE</td>
</tr>
<tr>
<td></td>
<td>Pomset-ST-bisimulation [31]</td>
<td>DEXPTIME-hard and in EXPSPACE</td>
</tr>
</tbody>
</table>

2. Deciding Pomset-Trace Equivalence

Throughout this paper, we use the standard definitions (cf. [27]) of Petri nets and their operational behavior. In order to keep this paper relatively self-contained, we repeat them here.

**Definition 2.1.** A labeled Petri net, $N$, is a triple $(S_N, T_N, Start_N)$, where $S_N$ is the set of places, $T_N$ is the set of transitions, and $Start_N$ is the set of initially marked places (which contain "tokens"). Every place $s \in S_N$ has a preset, $pre_N(s)$, and a post-set, $post_N(s)$. Every transition, $t$, in $T_N$ has a label, $l_N(t)$, a preset, $pre_N(t)$, and a post-set, $post_N(t)$. Labels are over a fixed set $Act \cup \{\tau\}$, where $Act$ is a set of "visible actions" and $\tau \notin Act$ is the "hidden action." A transition is visible (hidden) iff its label is visible (hidden). A net is finite iff it has a finite number of places and transitions; the size of a net is the total number of its places and transitions.

Transitions are represented graphically as horizontal bars, places are represented as circles, and tokens are represented as dots in these circles. The preset of a transition is the set of places from which there is an arrow to the transition; the post-set of a transition is the set of places to which there is an arrow from the transition. Dually, the preset (post-set) of a place is the set of transitions from (to) which there is an arrow to (from) the place.

A marking of a net is an assignment of a nonnegative number of "tokens" to each place in the net. A transition, $t$, is enabled under a marking iff every place in the preset
of \( t \) contains at least one token. If a transition \( t \) is enabled in a marking \( M \), then \( t \) can fire by removing a token from each place in its preset and placing a token into each place in its post-set. We write \( M[t)M' \), where \( M' \) is the resulting marking.

A firing sequence of a net, \( N \), is a possibly empty sequence, \( t_1 \ldots t_k \), of transitions of \( N \) such that \( t_1 \) is enabled under the initial marking of \( N \), and each \( t_i \) is successively enabled in the marking resulting from firing \( t_1 \ldots t_{i-1} \). A run is a finite firing sequence. The reachable markings of a net are exactly those markings that result from firing some run. A net is 1-safe iff every place contains at most one token under every reachable marking. Rather than being represented as a function from places to non-negative integers, a marking of a 1-safe net can be written as the set of places that contain a token.

Throughout the remainder of this paper, we use the term nets to refer to marked, 1-safe Petri nets whose transitions have labels over a fixed set \( \text{Act} \cup \{\tau\} \), where \( \tau \not\in \text{Act} \).

In order to define various interleaving semantics on nets, we will find it useful to represent the behavior of nets as labeled transition systems. The following definition is standard and is essentially taken verbatim from [9].

**Definition 2.2.** A labeled transition system (LTS) is a triple \( \langle S, \text{Act} \cup \{\tau\}, \to, s_{\text{init}} \rangle \), where
- \( S \) is a set of states containing \( s_{\text{init}} \).
- \( \text{Act} \cup \{\tau\} \) is a set of labels, such that \( \tau \not\in \text{Act} \).
- \( \to \) is a relation in \( S \times \text{Act} \times S \).
- \( s_{\text{init}} \) is designated as the "initial state" in \( S \).

We write \( s \xrightarrow{a} s' \) in place of \( (s, a, s') \in \to \). The relations \( \xrightarrow{a} \) are extended to relations \( \xrightarrow{v} \), for every \( v \in \text{Act}^* \), in the obvious way:
1. \( s \xrightarrow{\varepsilon} s' \) iff \( s' = s \),
2. \( s \xrightarrow{a\tau} s' \) iff \( s \xrightarrow{a} s'' \) for some \( s'' \) such that \( s'' \xrightarrow{\tau} s' \).

This means \( s \xrightarrow{v} s' \) if \( s \) can evolve to \( s' \) by performing the sequence of actions \( v \).

We also write \( s \xrightarrow{\tau} \) to mean that there exists a \( s' \) such that \( s \xrightarrow{v} s' \). We say that an action, \( a \), is enabled at a state, \( s \), iff \( s \xrightarrow{a} \).

These relations are generalized as follows, for every \( v \in \text{Act} \cup \{\tau\}^* \):
1. \( s \xrightarrow{a} s' \) iff \( s \xrightarrow{\tau} s_1 \xrightarrow{a} s_2 \xrightarrow{\tau} \ldots \xrightarrow{\tau} s' \) for some states \( s_1, s_2 \) and some \( i, j \geq 0 \),
2. \( s \xrightarrow{\tau} s' \) iff \( s \xrightarrow{\tau} s_1 \xrightarrow{\tau} s_2 \xrightarrow{\tau} \ldots s' \) for some \( k \geq 0 \),
3. \( s \xrightarrow{a\tau} s' \) iff \( s \xrightarrow{a} s'' \) for some \( s'' \) such that \( s'' \xrightarrow{\tau} s' \).

This means \( s \xrightarrow{v} s' \) if \( s \) can evolve to \( s' \) by performing the sequence of actions \( v \), possibly interspersed with \( \tau \)-actions. We also write \( s \xrightarrow{v} \) to mean that there exists a \( s' \) such that \( s \xrightarrow{v} s' \).

The following definition is essentially standard (cf. [17]).

**Definition 2.3.** The labeled transition system of a net \( N \), written \( \text{Lts}(N) \), is the labeled transition system over \( \text{Act} \cup \{\tau\} \) whose states are the reachable markings of \( N \) and
whose labeled transitions correspond to firings of single transitions of $N$. In particular, state $M$ goes to state $M'$ via an $a$-labeled transition in $lts(N)$ iff marking $M'$ of $N$ can be reached from marking $M$ by firing exactly some $a$-labeled transition of $N$. The initial state of $lts(N)$ is defined to be the initial marking of $N$.

We now consider "true concurrency" semantics.

**Definition 2.4.** A **pomset** is a labeled partial order. Formally, a pomset, $p$, consists of a set $\text{Events}_p$ whose elements are called **events**, a set $\text{Labels}_p$ whose elements are called **labels**, a function $\text{label}_p : \text{Events}_p \rightarrow \text{Labels}_p$, and a partial order relation $<_p$ on $\text{Events}_p$. A function $f$ is an isomorphism between pomset $p$ and pomset $q$ iff it is a label-preserving order-isomorphism, namely,

- $f : \text{Events}_p \rightarrow \text{Events}_q$ is a bijection,
- $\text{label}_p = \text{label}_q \circ f$,
- $e <_p e'$ iff $f(e) <_q f(e')$ for all $e, e' \in \text{Events}_p$.

An event $e$ is maximal in $p$ iff there is no event $e'$ in $p$ such that $e <_p e'$.

The **places** of a transition $t$ of a net $N$ are the places directly connected to it, i.e., the union of the preset and postset of $t$. Let $t_1, t_2$ be transitions of a net $N$. We say that $t_1$ and $t_2$ are **statically concurrent** in $N$ iff the places of $t_1$ are disjoint from the places of $t_2$.

A transition-sequence $r = t_1 \ldots t_n$ is a sequence of transitions of a net $N$. We write $|r|$ for the length of $r$, and for any $1 \leq i \leq |r|$, we write $r[i]$ to denote the $i$th element, $t_i$, of $r$. For any transition $t$, we write $r.t$ for the sequence $t_1 \ldots t_n t$.

The **transition-pomset** of $r$ has as events the integers from 1 to $n$, where the label of event $i$ is $t_i$ and the partial ordering is the transitive closure of the following "proximate cause" relation: event $i$ proximately causes event $j$ iff $i < j$ and $t_i$ and $t_j$ are not statically concurrent in $N$, cf. Fig. 1.

The **visible-pomset** of $r$ is the transition-pomset of $r$, restricted to events labeled with visible transitions; moreover, in the visible-pomset, the label of event $i$ is the label of $t_i$ (rather than $t_i$ itself), cf. Fig. 1. The **pomset-traces** of $N$ are the visible-pomsets of runs of $N$.

---

**Fig. 1.** An example of a transition-pomset and pomset-trace.
For transition-pomsets and visible-pomsets, it is traditional to say that event \( e \) causes event \( e' \) iff \( e < e' \) in the partial order.

**Definition 2.5.** Let \( N \) and \( N' \) be nets. Then \( N \) pomset-trace approximates \( N' \), written \( N \sqsubseteq_{pt} N' \), iff every pomset-trace of \( N \) is isomorphic to some pomset-trace of \( N' \). \( N \) and \( N' \) are pomset-trace equivalent iff each is \( \sqsubseteq_{pt} \) the other.

The runs of a finite net are clearly recognizable by a finite state automaton, namely, the “global state” automaton of the net itself. We represent an ordered pair \( r = t_1 \ldots t_n \), \( r'' = t_1'' \ldots t_n'' \), of transition-sequences of the same length as an input string \((t_1, t_1'') \ldots (t_n, t_n'')\) for an automaton whose alphabet is ordered pairs of transitions. So an “obvious” solution to the pomset-trace equivalence problem would be to define an effective procedure that, given any two finite nets as input, computes a finite-state automaton whose language consists of all the pairs of runs of the respective nets that have isomorphic pomset-traces. Such an automaton would easily yield a decision procedure for pomset-trace equivalence, since we could project the language it accepts onto the components of the pairs and check that the resulting languages include the set of runs of the respective nets.

However, such a finite-state automaton does not exist; the difficulty is that pairs of runs with isomorphic pomset-traces may generate the pomset-traces in different order, one getting unboundedly behind the other before catching up at the end. For example, let \( N \) be the net pictured in Fig. 2. Then two runs of \( N \) have the same pomset-trace iff they have the same number of occurrences of \( a \)- and \( b \)-labeled transitions, and the set of such pairs of runs is obviously not finite-state recognizable.

We will show in this section that it suffices to consider pairs of runs that are “synchronous” in the sense that their behavior corresponds at each pair of transitions.

**Definition 2.6.** Let \( r \) and \( r' \) be runs of nets \( N \) and \( N' \), respectively. Then \( r' \) and \( r'' \) are equivalent up to concurrency iff they have isomorphic transition-pomsets.

We will show that

- for all pairs of runs \( r \) and \( r' \) with isomorphic pomset-traces, there is a run \( r'' \) that is equivalent to \( r' \) up to concurrency, and \( r \) and \( r'' \) are “synchronous”;
- the set of pairs of synchronous runs is recognizable by a finite automaton with size bounded by an exponential in the sizes of the nets.

![Fig. 2. An example.](image-url)
Our decision procedure for pomset-trace equivalence is based on constructing such a finite-state automaton. To simplify the exposition, we consider first the case without hidden transitions. Our proofs will use the following fact about transition-pomsets, where we write \(|\mathcal{S}|\) for the size of any set \(\mathcal{S}\).

**Definition 2.7.** A pomset \(p'\) is a *linearization* of a pomset \(p\) iff it has the same events and labels as \(p\) and \(<_p\) is a total ordering that contains \(<_p\). Let \(q\) be a pomset such that \(<_q\) is a total ordering. Then for any \(1 \leq i \leq |\text{Events}_q|\), the \(i\)th largest event of \(q\) is the (necessarily unique) event \(e \in \text{Events}_q\) such that the longest chain \(e_1 <_q \cdots <_q e_k <_q e\) in \(q\) is of length \(i\).

**Proposition 2.8.** Let \(r\) be a run of a net \(N\), let \(p'\) be a linearization of the transition-pomset of \(r\), and let \(r'\) be the transition-sequence corresponding to \(p'\), i.e., \(r' = t_1 \ldots t_{|r'|}\), where each \(t_i\) is the label of the \(i\)th largest event of \(p'\). Then \(r'\) is a run of \(N\) reaching the same final marking as \(r\).

The proposition is easily proved by induction on the number of pairs \((i,j)\) such that \(i < j\) but the \(i\)th event of \(p'\) is larger (in the standard integer ordering) than the \(j\)th event of \(p'\). The details are omitted.

### 2.1. Nets without hidden transitions

In this section, we assume that nets do not contain hidden transitions.

**Definition 2.9.** Let \(r\) and \(r'\) be transition-sequences of nets \(N\) and \(N'\), respectively. We say that \(r\) and \(r'\) are *synchronous* iff the identity function on \(\{1, 2, \ldots, |r|\}\) is an isomorphism between the visible-pomset of \(r\) and the visible-pomset of \(r'\).

In particular, if \(r\) and \(r'\) are synchronous, then they are of the same length.

We then have:

**Lemma 2.10.** Let \(r\) and \(r'\) be runs of nets \(N\) and \(N'\), respectively. If the pomset-traces of \(r\) and \(r'\) are isomorphic, then there is some run \(r''\) of \(N'\) such that
- \(r'\) and \(r''\) are equivalent up to concurrency, and
- \(r\) and \(r''\) are synchronous.

**Proof.** Let \(I\) be the isomorphism between the pomset-trace of \(r\) and the pomset-trace of \(r'\). Since in this section we assume that nets do not contain hidden transitions, clearly \(r\) and \(r'\) are of the same length. Let \(r''\) be the transition-sequence obtained from \(r'\) by applying \(I\) elementwise to \(r\); that is, \(r''[i] = r'[I(i)]\) for all \(1 \leq i \leq |r'|\).

It follows easily from the definition of \(r''\) that \(I\) is a label-preserving bijection between the transition-pomsets of \(r''\) and \(r'\). To show that \(I\) is an order-isomorphism, it clearly suffices to show that \(I\) and \(I^{-1}\) preserve proximate causes. Let event \(i\) be a
proximate cause of event \( j \) in the transition-pomset of \( r'' \). Then \( i < j \), and transition \( r''[i] \) and transition \( r''[j] \) are not statically concurrent in \( N' \); hence, transition \( r'[I(i)] \) and transition \( r'[I(j)] \) are not statically concurrent in \( N' \). \( I(j) < I(i) \) would imply that event \( I(j) \) is a proximate cause of event \( I(i) \) in the pomset-trace of \( r' \); since \( I \) is an isomorphism between the pomset-trace of \( r \) and the pomset-trace of \( r' \), it would follow that event \( j \) causes event \( i \) in the pomset-trace of \( r \), and therefore that \( j < i \), a contradiction. Hence \( I(i) < I(j) \), and so event \( I(i) \) is a proximate cause of \( I(j) \) in the transition-pomset of \( r' \), proving this direction. The proof of the other direction is similar and omitted. This completes the proof that \( r' \) and \( r'' \) are equivalent up to concurrency; that is, they have isomorphic transition-pomsets.

Every transition-sequence corresponds to a linearization of its transition-pomset, by definition. Since \( r' \) is a run, and \( r' \) and \( r'' \) have isomorphic transition-pomsets, Proposition 2.8 immediately implies that \( r'' \) is a run of \( N' \).

Clearly, \( I^{-1} \) is an isomorphism between the pomset-trace of \( r' \) and the pomset-trace of \( r'' \). Pomset isomorphisms are closed under function composition; thus \( I^{-1} \circ I \), i.e., the identity function on \( \{1, \ldots, |r|\} \), is an isomorphism between the pomset-trace of \( r \) and the pomset-trace of \( r'' \). This implies that \( r \) and \( r'' \) are synchronous, completing the proof of the lemma. □

An important property of synchronous transition-sequences is that their equal-length prefixes are also synchronous.

**Definition 2.11.** Let \( p \) be a pomset and \( e, e' \in \text{Events}_p \). Event \( e' \) is a **maximal cause** of event \( e \) in \( p \) providing \( e' <_p e \) and there is no event \( e'' \in \text{Events}_p \) such that \( e' <_p e'' <_p e \).

**Proposition 2.12.** Let \( r \) and \( r' \) be transition-sequences of length \( n \geq 0 \) and let \( t \) and \( t' \) be transitions of nets \( N \) and \( N' \), respectively. Then \( r.t \) and \( r'.t' \) are synchronous if

- \( r \) and \( r' \) are synchronous,
- \( t \) and \( t' \) have the same label, and
- the maximal causes of event \( n + 1 \) are the same in the transition-pomsets of \( r.t \) and \( r'.t' \).

The proof is completely straightforward and is omitted.

Thus, in determining whether two pomset-traces "grow" synchronously, it suffices to keep track of the correspondence between maximal causes. We now observe that all maximal causes will necessarily be the most-recent firings of the corresponding transitions.

**Definition 2.13.** Let \( r = t_1 \ldots t_n \) be a transition-sequence of a net \( N \). Event \( i \) is a **most recent firing of transition \( t \)** in \( r \) iff \( t_i = t \) and \( t_j \neq t \) for \( i < j \leq n \). Let \( \text{growth-sites}(r) \) be the transition-pomset of \( r \), restricted to the most-recent firings of the transitions in \( r \), cf. Fig. 3.
Proposition 2.14. Let \( r = t_1 \ldots t_n \) be a transition-sequence and \( t \) be a transition of a net \( N \). Then the maximal causes of event \( n + 1 \) in the visible-pomset of \( r.t \) are a subset of the events of growth-sites(\( r \)).

Proof. Suppose event \( i \) of the visible-pomset of \( r.t \) is a maximal cause of event \( n + 1 \). Then by the definition of the causal partial ordering, event \( i \) must be a proximate cause of event \( n + 1 \), and hence transition \( t_i \) must not be statically concurrent with \( t \). Therefore any later firing of \( t_i \), that is, any event \( j \) with \( i < j \leq n \) and \( t_j = t_i \), would also be a proximate cause of \( t \). But since event \( i \) proximately causes any such event \( j \), this would contradict event \( i \) being a maximal cause of event \( n + 1 \).

We also make the simple observation that the growth-sites of transition-sequence \( r.t \) are fully determined by \( t \) and the growth-sites of \( r \).

Proposition 2.15. Let \( r \) be a transition-sequence and \( t \) a transition of a net \( N \). Then
\[
growth-sites(r.t) = \{ i \in growth-sites(r) : r[i] \neq t \} \cup \{|r.t|\}
\]

Proof. Clearly, event \(|r.t|\) is the most-recent firing of transition \( t \) in \( r.t \). Furthermore, the most recent firing of any other transition \( t' \) is the same in \( r \) and \( r.t \).
It now follows that whether two synchronous runs remain synchronous after firing another pair of transitions depends solely on the labels of these transitions, and on whether the causes of these transitions are the same in the growth-sites of the respective runs. It will be helpful to define a more general growth-site correspondence ($gsc$) between causes in growth-sites. To avoid confusion, we introduce the following terminology.

**Definition 2.16.** Let $p$ and $q$ be pomsets and let $f : p \rightarrow q$ be a partial function from $\text{Events}_p$ to $\text{Events}_q$. Then $p$ is the source of $f$, written $\text{source}(f)$, and $q$ is the target of $f$, written $\text{target}(f)$. Furthermore, the domain-of-definition of $f$ is the subset of $\text{Events}_p$ given by $\{ e \in \text{Events}_p : f(e) \text{ is defined} \}$, and the image of $f$ is the subset of $\text{Events}_q$ given by $\{ e' \in \text{Events}_q : f(e) = e' \text{ for some } e \in \text{Events}_p \}$.

**Definition 2.17.** Let $r = t_1 \ldots t_n$ and $r' = t'_1 \ldots t'_m$ be transition-sequences of nets $N$ and $N'$, respectively. Then $gsc(r, r')$ is defined iff $r$ and $r'$ are synchronous. Furthermore, if $r$ and $r'$ are synchronous, then $gsc(r, r')$ is the partial identity function $\beta : \text{growth-sites}(r) \rightarrow \text{growth-sites}(r')$ such that $\beta(i) = j$ iff $i = j$ and $i \in \text{Events}_{\text{growth-sites}(r)} \cap \text{Events}_{\text{growth-sites}(r')}$, cf. Fig. 3. In particular, $\text{growth-sites}(r)$ is the source of $gsc(r, r')$, and $\text{growth-sites}(r')$ is the target of $gsc(r, r')$.

We now state the key observation underlying our decision procedure: the growth-site correspondence of a pair of runs $r.t$ and $r'.t'$ is determined up to isomorphism by the isomorphism class of the growth-site correspondence between $r$ and $r'$.

**Definition 2.18.** Let $\beta$ and $\gamma$ be partial functions whose source and target are pomsets. We say that $\beta$ and $\gamma$ are isomorphic, written $\beta \approx \gamma$, iff there is a pair of functions $(I, J)$ such that

- $I$ is an isomorphism between $\text{source}(\beta)$ and $\text{source}(\gamma)$,
- $J$ is an isomorphism between $\text{target}(\beta)$ and $\text{target}(\gamma)$, and
- $\gamma \circ I = J \circ \beta$.

**Lemma 2.19.** Let $r_1, r_2$ be transition-sequences and $t$ a transition of net $N$; likewise for $r'_1, r'_2, t'$ of net $N'$. If $gsc(r_1, r'_1) \approx gsc(r_2, r'_2)$, then $gsc(r_1.t, r'_1.t') \approx gsc(r_2.t, r'_2.t')$.

**Proof.** Let $(I, J)$ be the isomorphism between $gsc(r_1, r'_1)$ and $gsc(r_2, r'_2)$, noting that both $gsc(r_1, r'_1)$ and $gsc(r_2, r'_2)$ are defined.

We define the function $I'$ to be

$$I'(i) = \begin{cases} |r_2.t| & \text{if } i = |r_1.t|, \\
I(i) & \text{if } i \in \text{Events}_{\text{growth-sites}(r_1,t)} \text{ and } i \neq |r_1.t|, \end{cases}$$

and define the function $J'$ to be

$$J'(j) = \begin{cases} |r'_2.t'| & \text{if } j = |r'_1.t'|, \\
J(j) & \text{if } j \in \text{Events}_{\text{growth-sites}(r'_1,t')} \text{ and } j \neq |r'_1.t'|. \end{cases}$$
By Proposition 2.15, $I'$ and $J'$ are total functions on $\text{Events}_{\text{growth-sites}(r_1,t)}$ and $\text{Events}_{\text{growth-sites}(r_2,t')}$, respectively. Definition 2.13, Proposition 2.15, and the properties of $I$ and $J$ imply that $I'$ is an isomorphism between $\text{growth-sites}(r_1,t)$ and $\text{growth-sites}(r_2,t)$, and $J'$ is an isomorphism between $\text{growth-sites}(r_1',t')$ and $\text{growth-sites}(r_2',t')$. The details are omitted.

In order to prove that $gsc(r_2,t,r_2,t') \circ I' = J' \circ gsc(r_1,t,r_1,t')$, we first show that $gsc(r_1,t,r_1',t')$ is defined iff $gsc(r_2,t,r_2',t')$ is defined. For one direction, suppose that $gsc(r_1,t,r_1',t')$ is defined; thus, $r_1,t$ and $r_1',t'$ are synchronous and $t$ and $t'$ have the same label. Furthermore, since $gsc(r_2,r_2')$ is defined, we have that $r_2$ and $r_2'$ are synchronous and $|r_2| = |r_2'|$. By Proposition 2.12, it remains to show that the maximal causes of event $|r_2,t|$ are the same in the transition-pomsets of $r_2,t$ and $r_2',t'$. For one direction, let event $k$ be a maximal cause of event $|r_2,t|$ in the transition-pomset of $r_2,t$; Proposition 2.14 implies that $k \in \text{growth-sites}(r_2)$. Since $I$ is an isomorphism between $\text{growth-sites}(r_1)$ and $\text{growth-sites}(r_2)$, it follows that $I^{-1}(k) \in \text{growth-sites}(r_1,t)$ and that event $I^{-1}(k)$ is a maximal cause of event $|r_1,t|$ in the transition-pomset of $r_1,t$; the details are straightforward but slightly tedious and are omitted. Since $r_1,t$ and $r_1',t'$ are synchronous, Proposition 2.12 implies that event $I^{-1}(k)$ is also a maximal cause of event $|r_1',t'|$ in the transition-pomset of $r_1',t'$, $r_1'[I^{-1}(k)]$ and $t'$ are not statically concurrent, and $I^{-1}(k) \in \text{growth-sites}(r_1')$. Definitions 2.13, 2.17, and 2.18 and our definition of $(I,J)$ then imply that $r_1'[I^{-1}(k)] = r_2'[J(I^{-1}(k))] = r_2'[k]$, and so $r_2'[k]$ and $t'$ are not statically concurrent; hence, event $k$ must cause event $|r_2',t'|$ in the transition-pomset of $r_2',t'$. The other direction is analogous, and so the maximal causes of event $|r_2,t|$ are the same in the transition-pomsets of $r_2,t$ and $r_2',t'$. Thus, by Proposition 2.12, $r_2,t$ and $r_2',t'$ are synchronous, proving that $gsc(r_2,t,r_2',t')$ is defined. The proof of the other direction, namely that $gsc(r_1,t,r_1',t')$ is defined whenever $gsc(r_2,t,r_2',t')$ is defined, is analogous and omitted.

We now show that $gsc(r_2,t,r_2',t') \circ I' = J' \circ gsc(r_1,t,r_1',t')$. For one direction, let $i$ be some event on which $gsc(r_2,t,r_2',t') \circ I'$ is defined. It then follows by Definition 2.17 and the definition of $I'$ that $i \in \text{growth-sites}(r_1,t)$, $I'(i) \in \text{growth-sites}(r_2,t) \cap \text{growth-sites}(r_2',t')$, and $gsc(r_2,t,r_2',t')$ is defined; thus, by the above proof, $gsc(r_1,t,r_1',t')$ is defined, $|r_1,t| \neq |r_1',t'|$, and $|r_2,t| \neq |r_2',t'|$. For one case, suppose that $i \neq |r_1,t|$; then $I'(i) = I(i) \neq |r_2',t'|$ and thus by Proposition 2.15, $i \in \text{growth-sites}(r_1)$, $I'(i) \in \text{growth-sites}(r_2) \cap \text{growth-sites}(r_2')$, and $r_2'[I'(i)] \neq t'$. Since by assumption, $gsc(r_1,r_1')$ and $gsc(r_2,r_2')$ are defined and $gsc(r_2,r_2') \circ I = J \circ gsc(r_1,r_1')$, it follows that $(J \circ gsc(r_1,r_1')(i) = I'(i)).$ Thus, $i \in \text{growth-sites}(r_1')$, $I'(i) = J(i)$, and $r_1'[i] = r_2'[J(i)] = r_2'[I'(i)]$, and so $r_1'[i] \neq t'$. Proposition 2.15 then implies that $i \in \text{growth-sites}(r_2',t')$, and so $J' \circ gsc(r_1,t,r_1',t')$ is defined on $i$. Furthermore,

$$(gsc(r_2,t,r_2',t') \circ I'(i) = (gsc(r_2,r_2') \circ I)(i) = (J \circ gsc(r_1,r_1'))(i) = (J' \circ gsc(r_1,t,r_1',t'))(i)$$

proving this case. The other case is similar and are omitted. The proof of the other direction is analogous. □
The size of the growth-sites of any transition-sequence of a net is obviously bounded by the number of transitions in that net. We can thus easily conclude that the number of isomorphism classes of growth-site correspondences between transition-sequences of nets $N$ and $N'$ is bounded by an exponential in the maximum of the number of transitions in $N$ and $N'$.

We thus have:

**Theorem 2.20.** For any finite nets $N$ and $N'$, there is a deterministic finite-state automaton recognizing the set of pairs of synchronous transition-sequences of $N$ and $N'$. If $m$ and $m'$ are the number of transitions in $N$ and $N'$, respectively, then the number of states in the automaton is bounded by $c^{\max\{m,m'\}^2}$ for some fixed constant $c > 1$.

**Proof.** The states of the automaton are the isomorphism classes of growth-site correspondences between transition-sequences of $N$ and $N'$. A state $\beta$ moves to a state $\gamma$ via a pair $(t,t')$ of transitions iff $\beta$ is the isomorphism class of $gsc(r,r')$ and $\gamma$ is the isomorphism class of $gsc(r,t,r',t')$ for some transition-sequences $r$ and $r'$ of $N$ and $N'$, respectively. The start state is the isomorphism class of the empty function, and all states are accepting. By Lemma 2.19, this automaton is deterministic.

If $(t_1,t'_1)\ldots(t_k,t'_k)$ is in the language of the automaton, then by Lemma 2.19, the final state reached must be the isomorphism class of $gsc(t_1\ldots t_k,t'_1\ldots t'_k)$. Hence, this growth-site correspondence is defined, and so $t_1\ldots t_k$ and $t'_1\ldots t'_k$ are synchronous. Conversely, if $t_1\ldots t_k$ and $t'_1\ldots t'_k$ are synchronous, then all their equal-length prefixes are synchronous, and so $gsc(t_1\ldots t_i,t'_1\ldots t'_i)$ is defined for all $0 \leq i \leq k$. Hence, by Lemma 2.19 and the definition of the automaton, $(t_1,t'_1)\ldots(t_k,t'_k)$ is in its language.

Since the runs of a finite net are finite-state recognizable by the (necessarily deterministic) transition system of the net itself, and since finite-state recognizable sets are closed under intersection and renaming input symbols, we conclude:

**Corollary 2.21.** For any finite nets $N$ and $N'$, there is a finite-state automaton whose language is the set of runs $r$ of $N$ for which there is some run $r'$ of $N'$ such that $r$ and $r'$ are synchronous. If $m$ and $m'$ are the number of transitions in $N$ and $N'$, respectively, and $n$ and $n'$ are the number of places in $N$ and $N'$, respectively, then the number of states in the automaton is bounded by $d^{\max\{m,m'\}^2+\max\{n,n'\}}$ for some fixed constant $d > 1$.

**Proof.** The number of states in the deterministic automaton that recognizes the set of pairs of runs of $N$ and $N'$ is $b^{\max\{n,n'\}}$ for some fixed constant $b > 1$. The intersection of this automaton with that of Theorem 2.20 has number of states bounded by $d^{\max\{m,m'\}^2+\max\{n,n'\}}$ for some fixed constant $d > 1$. Then renaming each input symbol $(t,t')$ by symbol $t$ does not change the number of states and yields the desired automaton.
It is fairly straightforward to show that such an automaton can in fact be constructed in space proportional to the size of its transition table. The desired decidability result then follows as a corollary.

Theorem 2.22. The pomset-trace equivalence problem for finite nets without hidden transitions can be decided in space exponential in the number of places and transitions in the nets.

Proof. By Lemma 2.10 and Corollary 2.21, \( N \subseteq_{pt} N' \) iff the language of the finite-state automaton given in Corollary 2.21 is the set of all runs of \( N \). It is easy to construct another finite-state automaton, of essentially the same size, recognizing the runs of \( N \). So \( N \subseteq_{pt} N' \) iff these automata recognize the same language. But language equivalence is checkable in space proportional to the size of the automata [10]. \( \square \)

2.2. Nets with hidden transitions

We now show how the results above extend to nets which may contain hidden transitions. We begin by modifying our definition of “synchronous” to take account of hidden transitions. This new definition will coincide with Definition 2.9 for nets without hidden transitions.

Definition 2.23. Let \( r = t_1 \ldots t_n \) and \( r' = t'_1 \ldots t'_m \) be transition-sequences of nets \( N \) and \( N' \), respectively. Let \( \alpha_{r,r'} \) be the partial function on the integers such that \( \alpha_{r,r'}(i) = j \) iff \( t_i \) is the \( k \)th transition of \( r \) with a visible label and \( t'_j \) is the \( k \)th transition of \( r' \) with a visible label, for some (necessarily unique) \( k \). Then \( r \) and \( r' \) are synchronous iff \( \alpha_{r,r'} \) is an isomorphism between the visible-pomset of \( r \) and the visible-pomset of \( r' \). In particular, if \( r \) and \( r' \) are synchronous, then they have the same number of occurrences of visible transitions.

Lemma 2.10 continues to hold for this generalized notion of synchronous:

Lemma 2.24. Let \( r \) and \( r' \) be runs of nets \( N \) and \( N' \), respectively. If the pomset-traces of \( r \) and \( r' \) are isomorphic, then there is some run \( r'' \) of \( N' \) such that

- the transition-pomsets of \( r' \) and \( r'' \) are isomorphic, and
- \( r \) and \( r'' \) are synchronous.

Proof. The proof extends that of Lemma 2.10. Let \( I \) be the isomorphism between the pomset-trace of \( r \) and the pomset-trace of \( r' \), and let \( q \) and \( q' \), respectively, be the transition-pomsets of \( r \) and \( r' \). Clearly, \( r \) and \( r' \) must contain the same number, \( k \), of occurrences of transitions with visible labels. For \( 1 \leq i \leq k \), we define \( \text{vis}_r(i) \) to be the index of the \( i \)th visible transition-occurrence in \( r \); that is, \( \text{vis}_r(i) = m \), where \( r[m] \) is the (necessarily unique) \( i \)th transition of \( r \) with a visible label. We let \( v \) be the sequence of visible transition-occurrences obtained from \( r' \) by applying \( I \) elementwise to visible transitions of \( r \); that is, \( v[i] = r'[I(\text{vis}_r(i))] \) for all \( 1 \leq i \leq k \). We then obtain \( r'' \) by “padding” \( v \) with sequences \( w_i \) of hidden transition-occurrences of \( r' \); each composite
sequence $w_1 \ldots w_i$ will contain exactly the hidden transition-occurrences of $r'$ that are necessary for the $v[1], \ldots, v[i]$ to fire. In order to define the $w_i$, we first define $z_i$, for $1 \leq i \leq k$, to be the ascending sequence of indices of the "remaining" hidden transition-occurrences that causally precede $r'[I(vis_r(i))]$. Furthermore, we define $z_{k+1}$ to be the sequence of indices of "left-over" hidden transition-occurrences of $r'$.

$$z_i = \text{the ascending sequence over the set}$$

\[ \{ j < q : I(vis_r(i)) : r'[j] \text{ is a hidden transition and } j \not\in I(vis_r(n)) \text{ for all } n < i \} \]

$$z_{k+1} = \text{the ascending sequence over the set}$$

\[ \{ j \leq |r'| : r'[j] \text{ is a hidden transition and } j \not\in I(vis_r(n)) \text{ for all } n \leq k \} \]

We then define $r''$ to be the sequence $w_1 v[1]w_2 v[2] \ldots v[k]w_{k+1}$, where each $w_i$ is the sequence of transition-occurrences of $r'$ corresponding to $z_i$; that is, $|w_i| = |z_i|$ and $w_i[n] = r'[z_i[n]]$ for all $1 \leq n \leq |z_i|$. Hence, for all $1 \leq i \leq k$, $r''[vis_r(i)] = v[i] = r'[I(vis_r(i))]$.

Let

$$C(i) = \begin{cases} I(vis_r(vis_r^{-1}(i))) & \text{if } r''[i] \text{ is a visible transition,} \\ m & \text{if for some (necessarily unique) } n \text{ and hidden transition } t \\
& r''[i] \text{ is the } n \text{th occurrence of } t \text{ in } r'', \text{ and} \\
& r'[m] \text{ is the } n \text{th occurrence of } t \text{ in } r'. \end{cases}$$

It is straightforward but tedious to show that $C$ is a label-preserving bijection between the transition-pomsets of $r''$ and $r'$; the details are omitted.

To show that $C$ is an order-isomorphism, it clearly suffices to show that $C$ and $C^{-1}$ preserve proximate causes. Suppose that event $i$ is a proximate cause of event $j$ in the transition-pomset of $r''$; then $i < j$ and transition $r''[i]$ and transition $r''[j]$ are not statically concurrent in $N'$. Then by definition of $r''$ and $C$, transition $r'[C(i)]$ and transition $r'[C(j)]$ are not statically concurrent in $N'$. For one case, suppose that both $r''[i]$ and $r''[j]$ are visible transitions. $C(j) < C(i)$ would imply that event $C(j)$ is a proximate cause of event $C(i)$ in the pomset-trace of $r'$; since $I$ is an isomorphism between the pomset-trace of $r$ and the pomset-trace of $r'$, it would follow that event $I^{-1}(C(j))$ causes event $I^{-1}(C(i))$ in the pomset-trace of $r$, and so $I^{-1}(C(j)) < I^{-1}(C(i))$. Clearly, $vis_r$ and $vis_r^{-1}$ are monotone functions, implying that $j < i$, a contradiction. Hence $C(i) < C(j)$, and so event $C(i)$ is a proximate cause of event $C(j)$ in the transition-pomset of $r'$, proving this case.

For another case, suppose that $r''[i]$ is a hidden transition $t$, and $r''[j]$ is a visible transition. Then for some $n$, $r''[i]$ is the $n$th occurrence of $t$ in $r''$ and $r'[C(i)]$ is the $n$th occurrence of $t$ in $r'$. Let $n'$ be the number of occurrences of $t$ preceding $r''[j]$ in $r''$; clearly, $n' \geq n$ since $i < j$. By definition of $r''$, $r''[j] = v[vis_r^{-1}(j)]$; hence by definition of the $z_i$, there are distinct $l_1, \ldots, l_{n'}$ in $z_0 \ldots z_{vis_r^{-1}(j)}$ such that $r'[l_1], \ldots, r'[l_{n'}]$ is each an occurrence of $t$. Let $l$ be the maximum of $l_1, \ldots, l_{n'}$; from
the definition of $C$ and the $z_i$, $l > C(j)$ would imply that there is some $j' < j$ such that $r'[C(j')]$ is a visible transition and $l < q_i C(j')$. Then, clearly, $C(j) < q_i l < q_i C(j')$, and so $I(vis_r(vis_r^{-1}(j))) < q_i I(vis_r(vis_r^{-1}(j')))$. Since $I$ is an isomorphism between the pomset-traces of $r$ and $r'$, it would follow that $vis_r(vis_r^{-1}(j)) < q_i vis_r(vis_r^{-1}(j'))$, and so $vis_r(vis_r^{-1}(j)) < vis_r(vis_r^{-1}(j'))$. The monotonicity of $vis_r$ and $vis_r^{-1}$ would then imply that $j < j'$, a contradiction. Thus, $l < C(j)$ after all; now, $C(j) < C(i)$ would imply that there are $n' > n$ occurrences of $t$ preceding $r'[C(i)]$ in $r'$, contradicting the fact that $r'[C(i)]$ is the $n$th occurrence of $t$ in $r'$. Hence $C(i) < C(j)$, and so event $C(i)$ is a proximate cause of event $C(j)$ in the transition-pomset of $r'$, proving this case.

The proofs of the other cases and the other direction are similar, and are omitted.

The proof that $r''$ is a run of $N'$ is identical to that for Lemma 2.10. Clearly, $vis_{r''} \circ vis_{r}^{-1} \circ I^{-1}$ is an isomorphism between the pomset-trace of $r'$ and the pomset-trace of $r''$. Pomset isomorphisms are closed under function composition; thus, $vis_{r''} \circ vis_{r}^{-1} \circ I^{-1} \circ I$ is an isomorphism between the pomset-trace of $r$ and the pomset-trace of $r''$. It follows easily from the definitions of $\alpha_r,r''$, $vis_r$, and $vis_{r''}$ that $\alpha_r,r'' = vis_{r''} \circ vis_{r}^{-1}$, proving that $r$ and $r''$ are synchronous, and completing the proof of the lemma. □

The notion of maximal cause must now be sharpened to be a maximal visible cause.

**Definition 2.25.** Let $N$ be a net, let $p$ be a transition-pomset of $N$, and let $e,e' \in$ Events$_p$. Event $e'$ is a maximal visible cause of event $e$ in $p$ providing $I_p(e')$ is a visible transition of $N$, $e' < p e$ and there is no event $e'' \in$ Events$_p$ such that $I_p(e'')$ is a visible transition of $N$ and $e' < p e'' < p e$.

Then Proposition 2.12 generalizes as follows.

**Proposition 2.26.** Let $r,r'$ be transition-sequences and let $t,t'$ be visible transitions of nets $N,N'$, respectively. Then $r.t$ and $r'.t'$ are synchronous iff

- $r$ and $r'$ are synchronous,
- $t$ and $t'$ have the same label, and
- $\alpha_r,r'$ restricted to the maximal visible causes of event $|r| + 1$ in the transition-pomset of $r.t$ is a bijection onto the maximal visible causes of event $|r'| + 1$ in the transition-pomset of $r'.t'$.

Also, if $t$ is a hidden transition, then $r.t$ and $r'$ are synchronous iff $r$ and $r'$ are synchronous.

The proof is completely straightforward and omitted.

The notion of growth-sites extends to hidden transitions as follows.

**Definition 2.27.** Let $r$ be a transition-sequence of a net $N$. Let most-recent($r$) be the set of most recent firings in $r$ of each transition. Let max-visible-causes($t,r$) be the maximal visible causes (in the transition-pomset of $r$) of the most recent firing in $r$ of
transition \( t \). Then \( \text{growth-sites}(r) \) is the restriction of the transition-pomset of \( r \) to

\[
\text{most-recent}(r) \cup \bigcup \{ \text{max-visible-causes}(t,r) : t \text{ is a hidden transition} \}.
\]

As before, the maximal causes will necessarily be a subset of the events in the growth-sites.

**Proposition 2.28.** Let \( r = t_1 \ldots t_n \) be a transition-sequence and \( t \) be a visible transition of a net \( N \). Then the maximal causes of event \( n + 1 \) in the visible-pomset of \( r.t \) are a subset of the events of \( \text{growth-sites}(r) \).

**Proof.** Suppose event \( i \) of the visible-pomset of \( r.t \) is a maximal cause of event \( n + 1 \). For one case, suppose that event \( i \) is also a maximal cause of \( n + 1 \) in the transition-pomset of \( r.t \); then \( i \in \text{most-recent}(r) \) by a proof identical to that of Proposition 2.14. For the other case, there must be some event \( k \) in the transition-pomset of \( r.t \) such that \( t_k \) is a hidden transition, event \( i \) causes event \( k \), and event \( k \) is a maximal cause of event \( n + 1 \). It follows by the same reasoning as in the proof of Proposition 2.14 that event \( k \) must be the most-recent firing of transition \( t_k \) in \( r \). Therefore, event \( i \) not being in \( \text{growth-sites}(r) \) would imply that event \( i \) is not a maximal visible cause of event \( k \). There would thus be some event \( j \) in the transition-pomset of \( r \) such that \( t_j \) is a visible transition, event \( i \) causes event \( j \), and event \( j \) causes event \( k \). But this would contradict event \( i \) being a maximal cause of \( n + 1 \) in the visible-pomset of \( r.t \).

We now observe that the growth-sites of transition-sequence \( r.t \) are fully determined by \( t \), the growth-sites of \( r \), and the static concurrency relation of \( N \).

**Proposition 2.29.** Let \( r \) be a transition-sequence and \( t \) a transition of a net \( N \). Then an event \( i \) is a visible cause of event \( |r.t| \) in the transition-pomset of \( r.t \) iff \( i \in \text{growth-sites}(r) \), \( r[i] \) is a visible transition, and there is some event \( j \in \text{growth-sites}(r) \) such that transition \( r[j] \) and \( t \) are not statically concurrent, and either event \( i \) causes event \( j \) in \( \text{growth-sites}(r) \) or \( i = j \). Furthermore, an event \( i \) is a maximal visible cause of event \( |r.t| \) in the transition-pomset of \( r.t \) iff event \( i \) is a visible cause of event \( |r.t| \) in the transition-pomset of \( r.t \) and there is no event \( k \in \text{growth-sites}(r) \) such that event \( i \) causes event \( k \) in \( \text{growth-sites}(r) \) and event \( k \) is a visible cause of event \( |r.t| \) in the transition-pomset of \( r.t \).

The proposition is a straightforward consequence of Proposition 2.14; the details are omitted.

**Proposition 2.30.** Let \( r = t_1 \ldots t_n \) be a transition-sequence of a net \( N \). Then

\[
\text{most-recent}(r) = \{ i \in \text{growth-sites}(r) : \text{there is no event } j \in \text{growth-sites}(r) \text{ such that } j > i \text{ and } l_{\text{growth-sites}(r)}(i) = l_{\text{growth-sites}(r)}(j) \}.
\]
Furthermore,

\[ \text{max-visible-causes}(t_k, r) = \{ i \in \text{growth-sites}(r) : \text{there is some event } j \in \text{most-recent}(r) \text{ such that } l_{\text{growth-sites}(r)}(j) = t_k \text{ and event } i \text{ is a maximal visible cause of event } j \text{ in } \text{growth-sites}(r) \}. \]

The proposition is a simple consequence of Definition 2.27; the details are omitted.

**Proposition 2.31.** Let \( r \) be a transition-sequence and \( t \) a transition of a net \( N \). Then

\[ \text{growth-sites}(r.t) = \{|r.t|\} \cup \{ i \in \text{growth-sites}(r) : \text{either } i \in \text{most-recent}(r) \text{ and } r[i] \neq t \text{ or } i \in \text{max-visible-causes}(t', r) \text{ for some hidden transition } t' \neq t \text{ or } i \in \text{max-visible-causes}(t, r.t) \text{ and } t \text{ is a hidden transition} \}. \]

**Proof.** Clearly, event \( |r.t| \) is the most-recent firing of transition \( t \) in \( r.t \), and the most-recent firing of any other transition is the same in \( r \) and \( r.t \). Furthermore, the maximal visible causes of the most-recent occurrence of any hidden transition other than \( t \) are the same in the transition-pomsets of \( r \) and \( r.t \), from which the highlighted equality immediately follows. \( \square \)

As an immediate consequence of the preceding three propositions, we have:

**Proposition 2.32.** Let \( r \) be a transition-sequence and \( t \) a transition of a net \( N \). Then \( \text{growth-sites}(r.t) \) is fully determined by \( t \), \( \text{growth-sites}(r) \), and the static concurrency relation of \( N \).

Our definition of growth-site correspondences is also modified accordingly; this new definition will coincide with Definition 2.17 for nets without hidden transitions.

**Definition 2.33.** Let \( r \) and \( r' \) be transition-sequences of nets \( N \) and \( N' \), respectively. Then \( gsc(r, r') \) is defined iff \( r \) and \( r' \) are synchronous. Furthermore, if \( r \) and \( r' \) are synchronous, then \( gsc(r, r') \) is the 1-1 partial function \( \beta : \text{growth-sites}(r) \rightarrow \text{growth-sites}(r') \) such that

\[ \text{graph}(\beta) = \text{graph}(\alpha_r, r') \cap (\text{Events}_{\text{growth-sites}(r)} \times \text{Events}_{\text{growth-sites}(r')}). \]

In particular, \( \text{growth-sites}(r) \) is the source of \( gsc(r, r') \), and \( \text{growth-sites}(r') \) is the target of \( gsc(r, r') \).

Again, the growth-site correspondences are significant only up to isomorphism.
Lemma 2.34. Let $r_1, r_2$ be transition-sequences of net $N$ and let $r'_1, r'_2$ be transition-sequences of net $N'$. If $gsc(r_1, r'_1) \approx gsc(r_2, r'_2)$, then

- $gsc(r_1.t, r'_1.t') \approx gsc(r_2.t, r'_2.t')$ for any pair of visible transitions $t$ and $t'$ of $N$ and $N'$, respectively.
- $gsc(r_1.t, r'_1) \approx gsc(r_2.t, r'_2)$ for any hidden transition $t$ of $N$.
- $gsc(r_1.t', r'_1) \approx gsc(r_2.t', r'_2)$ for any hidden transition $t'$ of $N'$.

The proof is a straightforward but tedious adaptation of the proof of Lemma 2.19 and uses Definitions 2.4, 2.23, 2.27, and 2.33, and Propositions 2.26, 2.32, and 2.28, instead of the corresponding definitions and propositions in the previous section. The details are omitted.

We note that it follows from Definition 2.27 that the size of the growth-sites of any transition-sequence of a net is bounded by the square of the number of transitions in that net.

We remark that, in order to allow hidden transitions to move independently, the alphabet of the automaton of Theorem 2.20 is generalized to pairs $(u, u')$, where either $u$ and $u'$ are both visible transitions of the respective nets, or exactly one of $u$ and $u'$ is a hidden transition of the respective net and the other is a special symbol $\bullet$. We refer to any sequence $w$ of such pairs as a $\bullet$-pair-sequence, and for $i = 1, 2$, we write $proj_i(w)$ to denote the projection of $w$ onto its $i$th component alphabet, with all occurrences of $\bullet$ omitted.

Theorem 2.35. For any finite nets $N$ and $N'$, there is a deterministic finite-state automaton recognizing the set of pairs of synchronous transition-sequences of $N$ and $N'$. If $m$ and $m'$ are the number of transitions in $N$ and $N'$, respectively, then the number of states in the automaton is bounded by $c^{\max\{m, m'\}^4}$ for some fixed constant $c > 1$.

Proof. The states of the automaton are the isomorphism classes of growth-site correspondences between transition-sequences of $N$ and $N'$. A state $\beta$ moves to a state $\gamma$ via a pair $(t, t')$ of transitions iff $\beta$ is the isomorphism class of $gsc(r, r')$ and $\gamma$ is the isomorphism class of $gsc(r.t, r'.t')$ for some transition-sequences $r$ and $r'$ of $N$ and $N'$, respectively. A state $\beta$ moves to a state $\gamma$ via a pair $(t, \bullet)$ iff $\beta$ is the isomorphism class of $gsc(r, r')$ and $\gamma$ is the isomorphism class of $gsc(r.t, r'.t')$ for some transition-sequences $r$ and $r'$ of $N$ and $N'$, respectively; a similar definition applies to pairs $(\bullet, t')$. The start state is the isomorphism class of the empty function, and all states are accepting. By Lemma 2.34, this automaton is deterministic.

If $w = (u_1, u'_1) \ldots (u_k, u'_k)$ is in the language of the automaton, then by Lemma 2.34, the final state reached must be the isomorphism class of $gsc(proj_1(w), proj_2(w))$. Hence, this growth-site correspondence is defined, and so $proj_1(w)$ and $proj_2(w)$ are synchronous. Conversely, if $proj_1(w)$ and $proj_2(w)$ are synchronous, then $gsc(proj_1(w'), proj_2(w'))$ is defined for all prefixes $w'$ of $w$. Hence, by Lemma 2.34 and the definition of the automaton, $w$ is in its language. □

As before, we conclude:
Corollary 2.36. For any finite nets $N$ and $N'$, there is a finite-state automaton whose language is the set of runs $r$ of $N$ for which there is some run $r'$ of $N'$ such that $r$ and $r'$ are synchronous. If $m$ and $m'$ are the number of transitions in $N$ and $N'$, respectively, and $n$ and $n'$ are the number of places in $N$ and $N'$, respectively, then the number of states in the automaton is bounded by $d^{\max\{m,m'\}}^{\max\{n,n'\}}$ for some fixed constant $d > 1$.

Proof. The number of states in the deterministic automaton whose alphabet consists of $\bullet$-pairs and that recognizes the set of pairs of runs of $N$ and $N'$ is $b^{\max\{n,n'\}}$ for some fixed constant $b > 1$. The intersection of this automaton with that of Theorem 2.35 has number of states bounded by $d^{\max\{m,m'\}}^{\max\{n,n'\}}$ for some fixed constant $d > 1$. Then renaming each input symbol $(t, t')$ by symbol $t$, renaming each input symbol $(t, \bullet)$ by $t$, and renaming each input symbol $(\bullet, t')$ by $\varepsilon$ does not change the number of states and yields the desired automaton. \qed

The earlier argument without hidden transitions now carries over.

Theorem 2.37. The pomset-trace equivalence problem for finite nets that may contain hidden transitions can be decided in space exponential in the number of places and transitions in the nets.

Proof. Since, language equivalence of automata with $\varepsilon$-moves is decidable in space proportional to the size of the automata [10], the proof of the theorem is identical to that of Theorem 2.22, except that it uses Lemma 2.24 and Corollary 2.36. \qed

3. History-preserving bisimulation and pomset-bisimulation

In this section, we assume that all nets may contain $\tau$-labeled transitions. We begin by defining history-preserving bisimulation on nets. Our definition induces the same equivalence as that of [2, 19, 23, 31, 27].

Definition 3.1. A set $\mathcal{H}$ of triples of the form $(r, r', f)$ is a history-preserving bisimulation between nets $N$ and $N'$ iff

1. If $(r, r', f) \in \mathcal{H}$, then $r$ and $r'$ are runs of $N$ and $N'$, respectively, and $f$ is an isomorphism between pomset-trace$(r)$ and pomset-trace$(r')$.

2. $(\varepsilon, \varepsilon, \emptyset) \in \mathcal{H}$, where $\varepsilon$ is the empty transition-sequence.

3. If $(r, r', f) \in \mathcal{H}$ and $r.t$ is a run of $N$, then there is some, possibly empty, sequence of transitions $t_1 \ldots t_k$ and some function $f'$ such that $((r.t), (r'.t_1 \ldots t_k), f') \in \mathcal{H}$ and $f'$ restricted to pomset-trace$(r)$ equals $f$.

4. If $(r, r', f) \in \mathcal{H}$ and $r'.t'$ is a run of $N'$, then there is some, possibly empty, sequence of transitions $t_1 \ldots t_k$ and some function $f'$ such that $((r.t_1 \ldots t_k), (r'.t'), f') \in \mathcal{H}$ and $f'$ restricted to pomset-trace$(r)$ equals $f$. 
We say that \( N \) and \( N' \) are **history-preserving bisimilar** iff there exists a history-preserving bisimulation relating them.

Vogler [27, 29] has given an alternate characterization of history-preserving bisimulation based on partially ordered sets of places, together with a decidability result. We give an alternate proof based on the approach presented in Section 2. We recall that the finite automaton described in Theorem 2.35 is deterministic, and we let \( \text{update} \) refer to its state-transition function. Furthermore, for any \( \bullet \)-pair-sequence \( w \) and any \( \text{gsc} \) \( \beta \), we write \( \text{update}(\beta, w) \) to mean the successive application of \( \text{update} \) to each of the pairs in \( w \). For any net \( N \), we write \( \text{init}(N) \) to denote the initial marking of \( N \).

**Definition 3.2.** A set \( \mathcal{G} \) of triples of the form \((M, M', \beta)\) is a **gsc-bisimulation** between nets \( N \) and \( N' \) iff

1. If \((M, M', \beta) \in \mathcal{G}\), then \( M \) and \( M' \) are markings of \( N \) and \( N' \), respectively, and \( \beta \) is an isomorphism class of growth-site correspondences between \( N \) and \( N' \).
2. \((\text{init}(N), \text{init}(N'), \emptyset) \in \mathcal{G}\).
3. If \((M, M', \beta) \in \mathcal{G}\) and \( M[t]M_1 \) for some transition \( t \) and some marking \( M_1 \), then there is some marking \( M'_1 \) and some \( \bullet \)-pair-sequence \( w \) such that \( \text{proj}_1(w) = t \), \( M'[\text{proj}_2(w)]M'_1 \) and \((M_1, M'_1, \text{update}(\beta, w)) \in \mathcal{G}\).
4. Vice versa; if \((M, M', \beta) \in \mathcal{G}\) and \( M'[t']M'_1 \) for some transition \( t' \) and some marking \( M'_1 \), then there is some marking \( M_1 \) and some \( \bullet \)-pair-sequence \( w \) such that \( \text{proj}_2(w) = t' \), \( M[\text{proj}_1(w)]M_1 \) and \((M_1, M'_1, \text{update}(\beta, w)) \in \mathcal{G}\).

We say that \( N \) and \( N' \) are **gsc-bisimilar** iff there exists a gsc-bisimulation relating them.

**Lemma 3.3.** Nets are history-preserving bisimilar iff they are gsc-bisimilar.

**Proof.** For one direction, let \( \mathcal{H} \) be a history-preserving bisimulation between nets \( N \) and \( N' \). Let

\[ \mathcal{G} = \{(M, M', \text{gsc}(r, r')) : (r, r', \text{gsc}(r, r')) \in \mathcal{H}, \text{init}(N)[r]M \text{ and } \text{init}(N')[r']M'\}. \]

Property (1) and (2) of Definition 3.2 follow easily from Definitions 2.33 and 3.1; the details are omitted. To prove property (3), let \((M, M', \beta) \in \mathcal{G}\) and let transition \( t \) and marking \( M_1 \) be such that \( M[t]M_1 \). Clearly, there must be some \((r, r', \text{gsc}(r, r')) \in \mathcal{H}\) such that \( \beta = \text{gsc}(r, r') \), \( \text{init}(N)[r]M \), and \( \text{init}(N')[r']M' \). By Definition 3.1, \( r.t \) is a run of \( N \), and so property (3) of Definition 3.1 implies the existence of some, possibly empty, sequence of transitions \( t_1' \ldots t_k' \) and some function \( f' \) such that \((r.t, (r'.t'_1 \ldots t'_k), f') \in \mathcal{H} \) and \( f' \) restricted to \( \text{pomset-trace}(r) \) equals \( \text{gsc}(r, r') \). Definition 3.1 implies that \( f' \) is an isomorphism between the pomset-traces of \( r.t \) and \( r'.t'_1 \ldots t'_k \), from which it then follows easily from Definition 2.33 that \( f' = \text{gsc}(r.t, r'.t'_1 \ldots t'_k) \). The definition of \( \bullet \)-sequences, the definition of \( \text{update} \), and the definition of \( \mathcal{G} \) then immediately imply that property (3) of Definition 3.2 must hold for \( \mathcal{G} \). A similar proof holds for property (4), and hence \( \mathcal{G} \) is a gsc-bisimulation.
For the other direction, let $\mathcal{G}$ be a gsc-bisimulation between nets $N$ and $N'$. We define the set of triples $\mathcal{H}$ inductively as follows. For the basis step, let $\mathcal{H} = \{(e, e, 0)\}$. For one inductive step, if $(r, r', \alpha) \in \mathcal{H}$, and for some $t, t_1 \ldots t_k$,

1. $r.t$ is a run of $N$, $r'.t'_1 \ldots t'_k$ is a run of $N'$, and
2. $(M, M', gsc(r.t, r'.t'_1 \ldots t'_k)) \in \mathcal{G}$, where $init(N)[r.t]M$ and $init(N')[r'.t'_1 \ldots t'_k]M'$, then $(r.t, r'.t'_1 \ldots t'_k, \alpha_{r,t,r'.t'_1 \ldots t'_k}) \in \mathcal{H}$.

For the other inductive step, if $(r, r', \alpha) \in \mathcal{H}$, and for some $t_1 \ldots t_k, t'$,

1. $r.t_1 \ldots t_k$ is a run of $N$, $r'.t'$ is a run of $N'$, and
2. $(M, M', gsc(r.t_1 \ldots t_k, r'.t')) \in \mathcal{G}$, where $init(N)[r.t_1 \ldots t_k]M$ and $init(N')[r'.t']M'$, then $(r.t_1 \ldots t_k, r'.t', \alpha_{r,t,r'.t'_1 \ldots t'_k}) \in \mathcal{H}$.

By the definition of $gsc$ and the $\alpha$, it is clear that properties (1) and (2) of Definition 3.1 hold for $\mathcal{H}$. To prove (3), suppose that $(r, r', f) \in \mathcal{H}$ and $r.t$ is a run of $N$. Then $(M, M', gsc(r, r')) \in \mathcal{G}$, where $init(N)[r.t]M$ and $init(N')[r'.t']M'$. Let $M_1$ be the marking such that $init(N)[r.t]M_1$, and by the definition of $gsc$-bisimulations, there is some marking $M_1'$ and some $\bullet$-pair-sequence $w$ such that $proj_1(w) = t, M'[\text{proj}_2(w)]M_1'$ and $(M_1, M_1', \text{update}(gsc(r, r'), w)) \in \mathcal{G}$. Let $proj_2(w) = t_1' \ldots t_k'$; then by definition, $\text{update}(gsc(r, r'), w)$ is isomorphic to $gsc(r.t, r'.t'_1 \ldots t'_k)$, so $(r.t, r'.t'_1 \ldots t'_k, \alpha_{r,t,r'.t'_1 \ldots t'_k}) \in \mathcal{H}$. It is easy to see by the definition of $\alpha$ that $\alpha_{r,t,r'.t'_1 \ldots t'_k}$ restricted to the pomset-trace of $r$ is equal to $\alpha_{r', t'}$, which is in turn equal to $f$, proving this case. The proof of (4) is analogous.

As in Section 2.2, it is easy to see that for any finite net, the number of triples $(M, M', \beta)$ is bounded by an exponential in the sizes of the nets. We use this fact in our decision procedure.

**Theorem 3.4.** For finite nets that may contain hidden transitions, history-preserving bisimulation can be decided in deterministic time exponential in the number of places and transitions in the nets.

**Proof.** The algorithm to decide history-preserving bisimulation of nets $N$ and $N'$ is similar to the decision procedure for (interleaving) bisimulation by successive refinement. We start with a set $\mathcal{G}_0$ that contains all possible triples, and each step, we shrink this set. Specifically, we define inductively:

$$\mathcal{G}_0 = \{(M, M', \beta) : M, M' \text{ are markings of } N, N', \text{ and } \beta \text{ is a gsc-isomorphism class between } N \text{ and } N'\}$$

$$\mathcal{G}_{i+1} = \{(M, M', \beta) \in \mathcal{G}_i : \text{ for every transition } t \text{ and marking } M_1 \text{ with } M[t]M_1, \text{ there is some marking } M'_1 \text{ and some } \bullet \text{-pair-sequence } w \text{ such that } proj_1(w) = t, M'[\text{proj}_2(w)]M'_1, \text{ and } (M_1, M'_1, \text{update}(\beta, w)) \in \mathcal{G}_i \}$$

and vice-versa.
We now show that $N$ and $N'$ are gsc-bisimilar iff

$$(\text{init}(N), \text{init}(N'), \emptyset) \in \mathcal{G}_k$$

for any $k$ that exceeds the number of possible triples $(M,M',\beta)$. For one direction, let $\mathcal{G}'$ be a gsc-bisimulation between $N$ and $N'$. Using Definition 3.2, a simple induction on $i$ shows that $\mathcal{G}' \subseteq \mathcal{G}_i$ for all $i \geq 0$. Since Definition 3.2 implies that $(\text{init}(N), \text{init}(N'), \emptyset) \in \mathcal{G}'$, we have that $(\text{init}(N), \text{init}(N'), \emptyset) \in \mathcal{G}_k$, as desired. For the other direction, we observe that for all $i$, $\mathcal{G}_{i+1}$ is either a strict subset of $\mathcal{G}_i$ or $\mathcal{G}_i = \mathcal{G}_j$ for all $j > i$. Since $k$ is greater than the number of triples, it immediately follows that $\mathcal{G}_k = \mathcal{G}_{k+1}$. Thus, by Definition 3.2 and the definition of the $\mathcal{G}_i$, $\mathcal{G}_k$ is a gsc-bisimulation whenever it contains $(\text{init}(N), \text{init}(N'), \emptyset)$.

We observe that $k$ is easily bounded by an exponential in the sizes of $N$ and $N'$. It is also easy to check that $\mathcal{G}_k$ can be computed in $\text{DEXPTIME}$ in the size of $N$ and $N'$ (using a transitive closure technique as in [13] to calculate the existence of a $\bullet$-pair-sequence $w$). Thus, it can be checked in deterministic time exponential in the number of places and transitions in $N$ and $N'$ whether $(\text{init}(N), \text{init}(N'), \emptyset) \in \mathcal{G}_k$, and hence the theorem follows easily from Lemma 3.3.

We now define pomset-bisimulation. Our definition induces the same equivalence as that of [3, 23, 31].

**Definition 3.5.** A set $\mathcal{P}$ of pairs of the form $(M,M')$ is a pomset-bisimulation between nets $N$ and $N'$ iff

1. If $(M,M') \in \mathcal{P}$, then $M$ and $M'$ are markings of $N$ and $N'$, respectively.
2. $(\text{init}(N), \text{init}(N')) \in \mathcal{P}$.
3. If $(M,M') \in \mathcal{P}$ and $M[r]M_1$ for some transition-sequence $r$ and some marking $M_1$, then there is some transition-sequence $r'$ and some marking $M'_1$ such that the pomset-traces of $r$ and $r'$ are isomorphic, $M'[r']M'_1$, and $(M_1,M'_1) \in \mathcal{P}$.
4. Vice versa; if $(M,M') \in \mathcal{P}$ and $M'[r']M'_1$ for some transition-sequence $r'$ and some marking $M'_1$, then there is some transition-sequence $r$ and some marking $M_1$ such that the pomset-traces of $r$ and $r'$ are isomorphic, $M[r]M_1$, and $(M_1,M'_1) \in \mathcal{P}$.

We say that $N$ and $N'$ are pomset-bisimilar iff there exists a pomset-bisimulation relating them.

**Theorem 3.6.** For finite nets that may contain hidden transitions, pomset-bisimulation can be decided in space exponential in the number of places and transitions in the nets.

**Proof.** The algorithm to decide pomset-bisimulation of nets $N$ and $N'$ is also by successive refinement. We start with a set $\mathcal{P}_0$ that contains all possible pairs, and each step, we shrink this set. Specifically, we define inductively:

$$\mathcal{P}_0 = \{(M,M') : M,M' \text{ are markings of } N,N'\}$$
\[ P_{i+1} = \{(M, M') \in P_i : \text{ for every transition-sequence } r \text{ and marking } M_1 \text{ with } M[r]M_1, \text{ there is some transition-sequence } r' \text{ and some marking } M'_1 \text{ such that the pomset-traces of } r \text{ and } r' \text{ are isomorphic, } M'[r']M'_1, \text{ and } (M_1, M'_1) \in P_i \text{ and vice-versa}\} \]

It is straightforward to show that \( N \) and \( N' \) are pomset-bisimilar iff
\[ (\text{init}(N), \text{init}(N')) \in P_k \]
for any \( k \) that exceeds the number of pairs, and this number is easily bounded by an exponential in the sizes of \( N \) and \( N' \). To compute each \( P_{i+1} \), we use the following straightforward modification of the decision procedure for pomset-trace equivalence. For each pair \( (M, M') \in P_i \), let \( N_M \) be \( N \), except that the initial marking of \( N_M \) is \( M \) (rather than \( \text{init}(N) \)); net \( N'_M \) is defined similarly. As in the proof of Corollary 2.36, we intersect the automaton that recognizes the set of pairs of runs of \( N_M \) and \( N'_M \) with the automaton of Theorem 2.35 constructed for \( N_M \) and \( N'_M \). Each state of the resulting automaton is a pair of the form \((\beta, (M_1, M'_1))\), where \( M_1 \) is a state of \( N_M \) and \( M'_1 \) is a state of \( N'_M \). For each state \( (\beta, (M_1, M'_1)) \), we now add a new \( M_1 \)-labeled transition iff \((M_1, M'_1) \in P_i \); all such transitions lead to a single new, accepting state. All other states of the automaton are defined to be non-accepting. We then relabel the other transitions \((u, u')\) as in the proof of Corollary 2.36. Thus, the language of this automaton is all pairs \((r, M_r)\) of runs \( r \) and corresponding final marking \( M_r \) of \( N_M \) for which there is some run \( r' \) and corresponding final marking \( M'_r \) of \( N'_M \) such that \( r \) and \( r' \) are synchronous and \((M_r, M'_r) \in P_i \). It is easy to see that the transition table of this modified automaton remains exponential in the sizes of \( N \) and \( N' \). (An similar automaton is also constructed whose language is all pairs \((r', M'_r)\) of runs \( r' \) and corresponding final marking \( M'_r \) of \( N'_M \) for which there is some run \( r \) and corresponding final marking \( M_r \) of \( N_M \) such that \( r \) and \( r' \) are synchronous and \((M_r, M'_r) \in P_i \).)

By Proposition 2.8, Definition 2.23, and Lemma 2.24, it is then straightforward to show that \((M, M') \in P_{i+1} \) iff (1) the language of the finite-state automaton given above is the set of all pairs \((r, M_r)\) such that \( r \) is a run of \( N_M \) and \( M[r]M_r \), and (2) the language of the similar automaton constructed for \( N'_M \) is the set of all pairs \((r', M'_r)\) such that \( r' \) is a run of \( N'_M \) and \( M'[r']M'_r \). It is easy to construct other finite-state automata of essentially the same size, recognizing the set of such pairs \((r, M_r)\) or the set of such pairs \((r', M'_r)\). So \((M, M') \in P_{i+1} \) iff each of the two appropriate pairs of automata recognize the same language. Since language equivalence is checkable in space proportional to the size of the automata [10], each \( P_i \) can be computed in space exponential in the size of \( N \) and \( N' \), and hence so can \( P_k \).

4. Deciding other true concurrency equivalences

We begin with the standard interleaving equivalences [5, 16, 25] for processes represented as labeled transition systems. These equivalences apply straightforwardly to nets.
Definition 4.1. Let $TS = (S, \text{Act}, \rightarrow, s_{\text{init}})$ and $TS' = (S', \text{Act}', \rightarrow', s'_{\text{init}})$ be labeled transition systems. We say that a state $s$ is divergent iff $s$ can perform an infinite sequence of $\tau$-actions. A failure set of a state $s$ is any set of visible actions, $a$, that are not enabled at $s$, even after further performing any finite sequence of $\tau$-labeled actions; that is, $s \xrightarrow{a}$. Then:

$$traces(TS) \overset{\text{def}}{=} \{ v \in \text{Act}^* : s_{\text{init}} \xrightarrow{v} \},$$

$$F(TS) \overset{\text{def}}{=} \{ (v,F) : v \in \text{Act}, F \subseteq \text{Act}, \text{ and there is some state } s \text{ such that } s_{\text{init}} \xrightarrow{v} s \text{ and } F \text{ is a failure set of } s \}$$

$$\cup \{ (v,F) : v \in \mathcal{D}(TS) \text{ and } F \subseteq \text{Act} \}$$

$$\mathcal{D}(TS) \overset{\text{def}}{=} \{ v \cdot v' : v,v' \in \text{Act}^* \text{ and } s_{\text{init}} \xrightarrow{v} s \text{ for some divergent state } s \}.$$  

We say that $TS$ and $TS'$ are trace-equivalent iff $traces(TS) = traces(TS')$, and are failures-equivalent iff $\mathcal{D}(TS) = \mathcal{D}(TS')$ and $F(TS) = F(TS').$

We say that $TS$ and $TS'$ are strongly bisimilar iff there exists a relation $R \subseteq S \times S'$ such that

1. $(s_{\text{init}},s'_{\text{init}}) \in R$. 
2. If $(s,s') \in R$ and $s \xrightarrow{a} s_1$ for some $a \in \text{Act} \cup \{\tau\}$, then there is some $s'_1$ such that $s' \xrightarrow{a} s'_1$ and $(s_1,s'_1) \in R$. 
3. If $(s,s') \in R$ and $s' \xrightarrow{a} s'_1$ for some $a \in \text{Act} \cup \{\tau\}$, then there is some $s_1$ such that $s \xrightarrow{a} s_1$ and $(s_1,s'_1) \in R$.

We say that $TS$ and $TS'$ are weakly bisimilar iff there exists a relation $W \subseteq S \times S'$ such that

1. $(s_{\text{init}},s'_{\text{init}}) \in W$. 
2. If $(s,s') \in W$ and $s \xrightarrow{a} s_1$ for some $a \in \text{Act} \cup \{\varepsilon\}$, then there is some $s'_1$ such that $s' \xrightarrow{a} s'_1$ and $(s_1,s'_1) \in W$. 
3. If $(s,s') \in W$ and $s' \xrightarrow{a} s'_1$ for some $a \in \text{Act} \cup \{\varepsilon\}$, then there is some $s_1$ such that $s \xrightarrow{a} s_1$ and $(s_1,s'_1) \in W$.

We say that $TS$ and $TS'$ are delay bisimilar iff there exists a relation $D \subseteq S \times S'$ such that

1. $(s_{\text{init}},s'_{\text{init}}) \in D$. 
2. If $(s,s') \in D$ and $s \xrightarrow{a} s_1$ for some $a \in \text{Act} \cup \{\tau\}$, then either
   (a) $a = \tau$ and $(s_1,s') \in D$, or
   (b) there exists a path $s' \xrightarrow{\varepsilon} u' \xrightarrow{a} v' \xrightarrow{\varepsilon} s_1'$ such that $(s_1,v') \in D$ and $(s_1,s_1') \in D$. 
3. If $(s,s') \in D$ and $s' \xrightarrow{a} s_1'$ for some $a \in \text{Act} \cup \{\tau\}$, then either
   (a) $a = \tau$ and $(s_1,s_1') \in D$, or
   (b) there exists a path $s \xrightarrow{\varepsilon} u \xrightarrow{a} v \xrightarrow{\varepsilon} s_1$ such that $(v,s_1') \in D$ and $(s_1,s_1') \in D$. 

We say that \(TS\) and \(TS'\) are branching bisimilar iff there exists a relation \(\mathcal{R} \subseteq S \times S'\) such that

1. \((s_{\text{init}}, s'_{\text{init}}) \in \mathcal{R}\).
2. If \((s, s') \in \mathcal{R}\) and \(s \xrightarrow{a} s_1\) for some \(a \in \text{Act} \cup \{\tau\}\), then either
   (a) \(a = \tau\) and \((s_1, s') \in \mathcal{R}\), or
   (b) there exists a path \(s' \xrightarrow{\epsilon} u' \xrightarrow{a} v' \xrightarrow{\epsilon} s'_1\) such that \((s, u') \in \mathcal{R}\), \((s_1, v') \in \mathcal{R}\), and \((s_1, s'_1) \in \mathcal{R}\).
3. If \((s, s') \in \mathcal{R}\) and \(s' \xrightarrow{a} s'_1\) for some \(a \in \text{Act} \cup \{\tau\}\), then either
   (a) \(a = \tau\) and \((s, s'_1) \in \mathcal{R}\), or
   (b) there exists a path \(s \xrightarrow{\epsilon} u \xrightarrow{a} v \xrightarrow{\epsilon} s_1\) such that \((u, s') \in \mathcal{R}\), \((v, s'_1) \in \mathcal{R}\), and \((s_1, s'_1) \in \mathcal{R}\).

These equivalences apply directly to nets.

**Definition 4.2.** Any two nets \(N\) and \(N'\) are trace equivalent, failures equivalent, weakly, strongly, delay and/or branching bisimilar iff their labeled transition systems are.

Since the transition system of a net is a finite-state automaton, the decision procedures for the interleaving trace, failure and bisimulation equivalences for nets follow directly from the results of Kanellakis and Smolka [13] for finite-state automata.

**Theorem 4.3.** For finite nets that may contain hidden transitions, the trace equivalence problem and the failure equivalence problem can be decided in space which is a product of an exponential in the number of places in the nets and a polynomial in the number of transitions in the nets. Furthermore, the strong and weak bisimulation problems, the delay bisimulation problem, and the branching bisimulation problem can be decided in deterministic time which is a product of an exponential in the number of places in the nets and a polynomial in the number of transitions in the nets.

**Proof.** The transition system of a finite net is a deterministic finite-state automaton whose states correspond to the reachable markings of the net and whose transitions correspond to transitions of the net. Let \(m\) and \(m'\) be the number of transitions in \(N\) and \(N'\), respectively, and let \(n\) and \(n'\) be the number of places in \(N\) and \(N'\), respectively. Then the maximum of the number of transitions in these automata is bounded by \(m \cdot 2^{\max\{n, n'\}}\), and the maximum of the number of states in these automata is bounded by \(2^{\max\{n, n'\}}\). Clearly, relabeling each visible transition \(t\) with the label of \(t\) and relabeling each hidden transition \(t'\) with \(\epsilon\) does not change the sizes of the automata. (For strong, delay, and branching bisimulation, hidden transitions are relabeled with \(\tau\).)

By definition, the finite nets are trace, failures, or bisimulation equivalent iff these finite-state automata with \(\epsilon\)-moves are respectively trace, failures, or bisimulation equivalent. Trace equivalence of finite-state automata is checkable in space proportional to
the size of the automata [13], while bisimulation equivalence is checkable in \textsc{ptime} [13], as are delay bisimulation and branching bisimulation [8]. The decision procedure for divergence-respecting failures equivalence [5] of finite-state automata is a straightforward generalization of Kannelakis and Smolka's \textsc{pspace} decision procedure for divergence-blind failures equivalence.

The decision procedures for most of the other true concurrency equivalences in Table 1 then follow from reductions to the corresponding interleaving equivalences, which are part of known full abstraction proofs [11, 12, 28, 30]. We begin with perhaps the most basic of the true concurrency equivalences, the "step" equivalences.

**Definition 4.4.** Let \( N \) be a net and let \( T \subseteq T_N \). We say that \( T \) is a step iff all distinct \( t_i, t_j \) in \( T \) are statically concurrent, and we write \( \text{vis}(T) \) for the multiset of labels of the visible transitions in \( T \).

\( T \) is step-enabled under a marking \( M \) iff every \( t_i \) in \( T \) is enabled under \( M \). The result of firing \( T \) is the marking \( M' \) resulting from successively firing each of the \( t_i \) in \( T \), and we write \( M[T]M' \). A step-run of \( N \) is a sequence \( T_1 \ldots T_n \) of steps such that \( T_1 \) is step-enabled under the initial marking of \( N \), and each \( T_i \) is successively enabled under the marking resulting from firing \( T_1 \ldots T_{i-1} \).

The step-trace of a step-run \( r = T_1 \ldots T_n \) of \( N \) is the sequence of multisets \( \text{vis}(T_1) \ldots \text{vis}(T_n) \). A step-failure of a step-run \( r = T_1 \ldots T_n \) of \( N \) is a pair \(<\text{vis}(T_1) \ldots \text{vis}(T_n), F>\), where \( F \subseteq \text{Act} \) is a failure set of the marking reached after firing \( r \): that is, for all \( a \in \text{Act} \), no \( a \)-labeled transition is enabled in this marking, even after possibly firing any finite number of \( \tau \)-transitions. A step-divergence is a step-trace of a step-run after which an infinite sequence of \( \tau \)-labeled transitions is enabled. The step-traces of \( N \) is the set of step-traces of all step-runs of \( N \). The step-D of \( N \) is the set of step-divergences, closed under "step-extension"; this is the obvious analogue to the set \( D \) in interleaving failures semantics, cf. Definition 4.1. The step-F of \( N \) is the union of the sets of step-failures over all step-runs of \( N \), closed under step-D; again, this is the obvious step analogue to Definition 4.1.

**Definition 4.5.** Let \( N \) and \( N' \) be nets. Then \( N \) and \( N' \) are step-trace equivalent iff they have the same set of step-traces, and are step-failure equivalent iff they have the same step-F and step-D. We say that \( N \) and \( N' \) are step-bisimilar iff there is some relation \( \mathcal{R} \subseteq \text{markings}(N) \times \text{markings}(N') \) such that

1. If \((M, M') \in \mathcal{R}\), then \( M \) and \( M' \) are markings of \( N \) and \( N' \), respectively.
2. \((\text{init}(N), \text{init}(N')) \in \mathcal{R} \).
3. If \((M, M') \in \mathcal{R} \) and \( M[T]M_1 \) for some step \( T \) and some marking \( M_1 \), then there exists some step \( T' \) and some marking \( M'_1 \) such that \( \text{vis}(T) = \text{vis}(T'), M'[T']M'_1 \), and \((M_1, M'_1) \in \mathcal{R} \).
4. If \((M, M') \in \mathcal{R} \) and \( M'[T']M'_1 \) for some step \( T' \) and some marking \( M'_1 \), then there exists some step \( T \) and some marking \( M_1 \) such that \( \text{vis}(T) = \text{vis}(T'), M[T]M_1 \), and \((M_1, M'_1) \in \mathcal{R} \).
Theorem 4.6. For finite nets that may contain hidden transitions, the step-trace equivalence problem and the step-failure equivalence problem can be decided in space exponential in the number of places and transitions in the nets. Furthermore, the step-bisimulation problem can be decided in deterministic time exponential in the number of places and transitions in the nets.

Proof. By a known full abstraction result [12], there is a context $C[]$ involving only a self-synchronization operator [12] such that nets $N$ and $N'$ are step-trace, step-failures, or step-bisimulation equivalent iff the nets $C[N]$ and $C[N']$ are, respectively, trace equivalent, failures equivalent, or bisimulation equivalent. In particular, $C[]$ adds a new transition for every set of pairwise statically concurrent transitions, and does not add any new places.

Let $m$ and $m'$ be the number of transitions in $N$ and $N'$, respectively, and let $n$ and $n'$ be the number of places in $N$ and $N'$, respectively. Then the maximum of the number of transitions in $C[N]$ and $C[N']$ is bounded by $2\max{m, m'}$, and the maximum of the number of places in $C[N]$ and $C[N']$ is bounded by $\max{n, n'}$. The proof then follows easily by Theorem 4.3. ⊓⊔

We now consider interval-pomset-trace and interval-pomset-failures equivalence [11, 28], which have been shown there to be fully abstract for action refinement. The main idea behind these equivalences is that nets are first "split", so that every visible transition is split into two transitions. The pomsets-traces of these split nets are then closed with respect to augmentation of the partial orderings. Finally, this set is restricted to only those pomsets with a particular "interval ordering". Pomset-failures are similar, except that failure sets and a causal version of divergences are also tracked. We omit the precise definitions of these equivalences here.

The decision procedure for interval-pomset-trace equivalence and interval-pomset-failure equivalence relies on a full abstraction result involving action refinement.

Theorem 4.7. For finite nets that may contain hidden transitions, the interval-pomset-trace equivalence problem and the interval-pomset-failures equivalence problem can be decided in space exponential in the number of places and transitions in the nets.

Proof. By known full abstraction results [11, 28], there is a context $C[.]$ built from split and choice refinements such that nets $N$ and $N'$ are interval-pomset-trace equivalent or interval-pomset-failures equivalent iff the nets $C[N]$ and $C[N']$ are, respectively, trace equivalent or failures equivalent. In particular, $C[.]$ refines every visible transition by the net $a^+_1 a^+_{i1} + \cdots + a^+_k a^-_{ik}$, where $a$ is the label of the visible transition and $k$ is bounded by the maximum of the number of transitions in $N$ and $N'$.

Let $m$ and $m'$ be the number of transitions in $N$ and $N'$, respectively, and let $n$ and $n'$ be the number of places in $N$ and $N'$, respectively. Then the maximum of the number of transitions in $C[N]$ and $C[N']$ is bounded by $2 \cdot \max{m, m'}^2 + 1$, and the maximum
of the number of places in $C[N]$ and $C[N']$ is bounded by $\max\{n, n'\} + \max\{m, m'\}^2$. The proof then follows easily by Theorem 4.3. □

We now consider the ST-equivalences [22, 24], which Vogler [30] has shown coincide with the interval-pomset equivalences. Rather than "splitting" nets, the ST-equivalences keep track of possible "half-fired" transitions that are maximal in transition-pomsets of runs; we omit the precise definitions here. As an immediate consequence of Vogler's results, we have:

**Theorem 4.8.** For finite nets that may contain hidden transitions, the ST-trace equivalence problem and the ST-failure equivalence problem can be decided in space exponential in the number of places and transitions in the nets. Furthermore, the ST-bisimulation problem can be decided in deterministic time exponential in the number of places and transitions in the nets.

**Proof.** The proofs for ST-traces and ST-failures is identical to that of Theorem 4.7, while the proof for ST-bisimulation uses the same context $C[\cdot]$ to yield a reduction to bisimulation. The desired upper bound then follows by Theorem 4.3. □

Using the decision procedure for history-preserving bisimulation, a similar result holds for maximality-preserving bisimulation [7].

**Definition 4.9.** A set $\mathcal{M}$ of triples of the form $(r, r', f)$ is a *maximality-preserving bisimulation* between nets $N$ and $N'$ iff

1. If $(r, r', f) \in \mathcal{M}$, then $r$ and $r'$ are runs of $N$ and $N'$, respectively, and $f$ is an isomorphism between $\text{pomset-trace}(r)$ and $\text{pomset-trace}(r')$.
2. $(\epsilon, \epsilon, \theta) \in \mathcal{M}$, where $\epsilon$ is the empty transition-sequence.
3. If $(r, r', f) \in \mathcal{M}$ and $r.t$ is a run of $N$, then there is some, possibly empty, sequence of transitions $t'_1 \ldots t'_k$ and some function $f'$ such that
   (a) $((r.t), (r'.t'_1 \ldots t'_k), f') \in \mathcal{M}$ and $f'$ restricted to $\text{pomset-trace}(r)$ equals $f$,
   (b) if $t$ is visible and is maximal in the transition-pomset of $r.t$, then $f'(t)$ is maximal in the transition-pomset of $r'.t'_1 \ldots t'_k$,
   (c) for all visible $t_i \in r$ that are maximal in the transition-pomset of $r.t$, either $f(t_i)$ is maximal in the transition-pomset of $r'.t'_1 \ldots t'_k$ or $f(t_i)$ is not maximal in the transition-pomset of $r'$.
4. If $(r, r', f) \in \mathcal{M}$ and $r'.t'$ is a run of $N'$, then there is some, possibly empty, sequence of transitions $t_1 \ldots t_k$ and some function $f'$ such that
   (a) $((r.t_1 \ldots t_k), (r'.t'), f') \in \mathcal{M}$ and $f'$ restricted to $\text{pomset-trace}(r)$ equals $f$,
   (b) if $t'$ is visible and is maximal in the transition-pomset of $r'.t'$, then $f'^{-1}(t')$ is maximal in the transition-pomset of $r.t_1 \ldots t_k$,
   (c) for all visible $t'_i \in r'$ that are maximal in the transition-pomset of $r'.t'$, either $f'^{-1}(t'_i)$ is maximal in the transition-pomset of $r.t_1 \ldots t_k$ or $f'^{-1}(t'_i)$ is not maximal in the transition-pomset of $r$. 

We say that $N$ and $N'$ are maximality-preserving bisimilar iff there exists a maximality-preserving bisimulation relating them.

**Theorem 4.10.** For finite nets that may contain hidden transitions, the maximality-preserving bisimulation problem can be decided in deterministic time exponential in the number of places and transitions in the nets.

**Proof.** Let $C[\cdot]$ be the net context involving split and choice refinements given in the proof of Theorem 4.7. Then by a proof similar to that of [26], nets $N$ and $N'$ are maximality-preserving bisimilar iff the nets $C[N]$ and $C[N']$ are history-preserving bisimilar. The theorem is then a simple consequence of Theorem 3.4. □

Lastly, our decision procedure for pomset-bisimulation yields one for pomset-ST-bisimulation [31]. As with the interleaving ST-equivalences, "half-fired" transitions in the transition-pomsets of runs are additionally tracked; we omit the precise definition here.

**Theorem 4.11.** For finite nets that may contain hidden transitions, the pomset-ST bisimulation problem can be decided in space exponential in the number of places and transitions in the nets.

**Proof.** Let $C[\cdot]$ be the net context involving split and choice refinements given in the proof of Theorem 4.7. Then by a proof similar to that of [26], nets $N$ and $N'$ are pomset-ST-bisimilar iff the nets $C[N]$ and $C[N']$ are pomset-bisimilar. The theorem is then a simple consequence of Theorem 3.6. □

### 5. Lower bounds

The lower bounds for trace equivalence and bisimulation essentially follow from previous results of Mayer and Stockmeyer on Mazurkiewicz nets and regular expressions with interleaving. In particular, Mayer and Stockmeyer [15] have shown the \textsc{ExpSpace}-hardness of deciding whether the language of a regular expression with interleaving is $\Sigma^*$. Our \textsc{ExpSpace} lower bound for trace equivalence of finite 1-safe Petri nets follows by a polynomial-time reduction. For expository simplicity, we first give the proof for nets that may contain hidden transitions.

**Theorem 5.1.** The problem of deciding whether the language of a regular expression with interleaving is $\Sigma^*$ is polynomial-time reducible to trace equivalence of finite nets that may contain hidden transitions.

**Proof.** Let $\Sigma$ be a finite alphabet consisting only of visible labels, and let $\sqrt{ } \notin \Sigma$ be a visible label. For any regular expression $r$ over $\Sigma$ built from $\{\cup, *, \cdot, ||\}$, we give an inductive translation to finite 1-safe nets with labels from $\Sigma \cup \{\tau, \sqrt{ }\}$. Each of these nets will have exactly one $\sqrt{ }$-labeled transition, and the post-set of this transition will be empty.
The translation, \( \text{net} \), uses net operators defined in [11]; we do not repeat the definitions here. However, we slightly modify the internal choice operator presented there to ensure that the resulting nets always have exactly one \( \sqrt{-} \)-labeled transition. This in turn guarantees that the translation \( \text{net} \) can be performed in polynomial-time; that is, for any regular expression \( r \) with interleaving, the net \( \text{net}(r) \) can be constructed in deterministic time polynomial in the number of symbols in \( r \).

For every \( a \in \Sigma \), \( a \) is the net corresponding to \( a \cdot \sqrt{-} \). The \( \cdot \) operator is modeled by the sequencing operator on nets. The \( * \) operator applied to a net \( N \) adds the initially marked places of \( N \) to the post-set of its \( \sqrt{-} \)-labeled transition, relabels the \( \sqrt{-} \)-transition with \( \tau \), and hooks up a single new \( \sqrt{-} \)-labeled transition to the set of initially marked places of \( N \). The union operator applied to nets \( N \) and \( N' \) is modeled by the internal choice operator on nets except that in addition, the \( \sqrt{-} \)-labeled transitions of \( N \) and \( N' \) are relabeled by \( \tau \), one common new place is added to the post-set of both of these relabeled transitions, and this new place feeds into a new \( \sqrt{-} \)-labeled transition. The interleaving operator applied to nets \( N \) and \( N' \) is modeled by the noncommunicating parallel composition operator on nets, in which \( N \) and \( N' \) are simply placed side by side but required to synchronize on \( \sqrt{-} \)-labeled transitions. We note that since all nets in the target of \( \text{net} \) have exactly one \( \sqrt{-} \)-labeled transition, the noncommunicating parallel composition operator takes only a trivial cross-product of the \( \sqrt{-} \)-labeled transitions and hence adds no extra transitions (or places). This ensures that \( \text{net} \) is a polynomial-time translation in the length of \( r \).

It is straightforward to show by induction that each of the nets in the target of \( \text{net} \) will immediately reach a deadlocked state whenever its (necessarily unique) \( \sqrt{-} \)-labeled transition fires. Furthermore, this \( \sqrt{-} \)-labeled can be fired from any reachable marking, after first performing a finite, possibly empty, sequence of other transitions. For any regular expression \( r \) with interleaving, it follows by a simple induction that

\[
L(r) = \{ v \in \Sigma^* | \exists \sqrt{-} \text{ is a trace of } \text{net}(r) \},
\]

where \( L(r) \) is the language of \( r \).

Let \( N_{\sqrt{-}} \) be the finite net with exactly \( |\Sigma| + 1 \) transitions, each uniquely labeled from \( \Sigma \cup \{ \sqrt{-} \} \), and exactly one place, which is initially marked and is in the preset of all the transitions and in the post-set of all the transitions not labeled with \( \sqrt{-} \). The set of traces of \( N_{\sqrt{-}} \) is the prefix closure of \( \Sigma^* \cdot \sqrt{-} \). We show that for any regular expression \( r \) with interleaving, \( L(r) = \Sigma^* \) if and only if \( \text{net}(r) \) and \( N_{\sqrt{-}} \) are trace equivalent. One direction follows immediately from the equality highlighted above. For the other direction, suppose \( L(r) = \Sigma^* \). Since firing the \( \sqrt{-} \)-labeled transition immediately puts \( \text{net}(r) \) in a deadlocked state, clearly the traces of \( \text{net}(r) \) are contained in the traces of \( N_{\sqrt{-}} \). For the reverse containment, it follows immediately from the highlighted equality that the set \( \Sigma^* \cdot \sqrt{-} \) is contained in the traces of \( \text{net}(r) \). Since traces are prefix-closed, the set \( \Sigma^* \) is also contained in the traces of \( \text{net}(r) \), and so \( \text{net}(r) \) and \( N_{\sqrt{-}} \) are trace-equivalent.

This is a polynomial-time reduction from deciding whether the language of a regular expressions with interleaving is \( \Sigma^* \) to trace equivalence of finite nets with hidden transitions. \( \Box \)
We then have as a corollary:

**Theorem 5.2.** For finite nets that may contain hidden transitions, trace equivalence is EXSPACE-hard.

We now modify the proof of Theorem 5.1 to yield the lower bound for trace equivalence of finite nets without hidden transitions.

**Theorem 5.3.** The problem of deciding whether the language of a regular expression with interleaving is \( \Sigma^* \) is polynomial-time reducible to trace equivalence of finite nets without hidden transitions.

**Proof.** Let \( net \) be the translation defined in the proof of Theorem 5.1, and let \( 1 \not\in (\Sigma^* \cup \{\sqrt{\cdot}\}) \) be a visible label. For any regular expression \( r \) with interleaving, we define a new translation \( Net \) from \( net(r) \) as follows: first, we relabel all \( \tau \)-labeled transitions in \( net(r) \) with the label 1, then for every place \( s \) in \( net(r) \), we add a new 1-labeled transition and put it in the preset and postset of the place \( s \) (i.e., in a self-loop under \( s \)). \( Net(r) \) is defined to be the resulting net, and clearly can be constructed in polynomial time in the length of \( r \). Furthermore, \( Net(r) \) satisfies all the properties of \( net(r) \) specified in the proof of Theorem 5.1 concerning markings and \( \sqrt{\cdot} \)-labeled transitions. The labeled transition system of \( Net(r) \) is identical to that of \( net(r) \), except that all \( \tau \)-labeled transitions are replaced by 1-labeled transitions, and every state has a 1-labeled transition trivially looping back to itself.

It is straightforward to show by induction that for any regular expression \( r \) with interleaving, \( net(r) \) can perform at most \( 4 \cdot |r| \) consecutive \( \tau \)-moves, where \( |r| \) is the number of symbols in \( r \). By construction of \( Net(r) \), it then follows that

\[
L(r) = \{a_1 \ldots a_k \in \Sigma^* | 1^{4|r|}a_11^{4|r|} \ldots a_k1^{4|r|}\sqrt{\cdot} \text{ is a trace of } Net(r)\}.
\]

For any regular expression \( r \) with interleaving, let \( N_r \) be the finite net with \( 4 \cdot |r| + |\Sigma| + 1 \) transitions and \( 4 \cdot |r| + 1 \) places, whose set of traces is the prefix-closure of \((1^{4|r|} \cdot \Sigma)^* \cdot 1^{4|r|} \cdot \sqrt{\cdot}; \) the intended definition of \( N_r \) is obvious and omitted. By reasoning similar to that of the proof of Theorem 5.1, it follows that \( L(r) = \Sigma^* \) iff the set of traces of \( Net(r) \) contains the set of the traces of \( N_r \). The details are omitted.

To reduce trace-containment to trace equivalence, we observe that for any nets \( N_1 \) and \( N_2 \), the set of traces of \( N_1 \) contains the set of traces of \( N_2 \) iff the net \( (N_1 \parallel_{\Sigma \cup \{\sqrt{\cdot}\}} N_2) \) and the net \( N_2 \) are trace equivalent, where \( \parallel_{\Sigma \cup \{\sqrt{\cdot}\}} \) is a parallel composition operator which requires synchronization on (visible) labels and hence corresponds to trace intersection. Furthermore, the size of \( N_1 \parallel_{\Sigma \cup \{\sqrt{\cdot}\}} N_2 \) is polynomial in the sizes of \( N_1 \) and \( N_2 \), giving a polynomial-time reduction from trace containment to trace equivalence, and proving the theorem. \( \Box \)

We then have as a corollary:
Theorem 5.4. For finite nets without hidden transitions, trace equivalence is \textsc{ExpSpace}-hard.

Using these results, we obtain a lower-bound for failures equivalence; the proof is very similar to that of Kanellakis and Smolka [13] for finite-state automata.

Theorem 5.5. For finite nets without hidden transitions, trace equivalence is polynomial-time reducible to failures equivalence.

Proof. For any finite nets \( N_1 \) and \( N_2 \) without hidden transitions, let \( N'_1 \) be constructed by adding to \( N_i \) a single new, initially marked place, \( s_{\text{new}} \), which is placed in the preset and post-set of every transition of \( N_i \). The labeled transition system of \( N'_1 \) is isomorphic to that of \( N_i \). Now, \( N''_1 \) is constructed by adding to \( N'_1 \) a new \( a \)-labeled transition \( t_a \), for every visible label \( a \), and hooking up each \( t_a \) so that its post-set is empty and its preset contains only the place \( s_{\text{new}} \). All of the \( t_a \) are enabled under every reachable marking of \( N'_1 \), and firing any one of them puts \( N''_1 \) in a deadlocked state.

\( N_1 \) and \( N_2 \) are trace equivalent iff \( N''_1 \) and \( N''_2 \) are failures equivalent; the proof is identical to that of Kanellakis and Smolka [13] and is omitted. This is a polynomial-time reduction from trace equivalence to failures equivalence. \( \square \)

We then have as a corollary:

Theorem 5.6. Failures equivalence of finite nets is \textsc{ExpSpace}-hard.

Our proof of a \textsc{DExpTime} lower bound for strong bisimulation is a simple adaptation of Stockmeyer's result [20] for Mazurkiewicz nets: namely, we reduce the acceptance problem for polynomial-space alternating Turing machines to the bisimulation problem for finite 1-safe Petri nets. In particular, we simulate the tape and finite-state control of polynomial-space alternating Turing machines by polynomial-time constructible 1-safe Petri nets, and our reduction to bisimulation is essentially identical to that of Stockmeyer. Since Mazurkiewicz nets are somewhat more succinct than 1-safe Petri nets, our lower bound for bisimulation is a minor technical improvement of the results of Stockmeyer.

Theorem 5.7. The acceptance problem for polynomial-space alternating Turing machines is polynomial-time reducible to strong bisimulation of finite nets.

Proof. Let \( A \) be an alternating Turing machine that, for some polynomial \( p \), uses \( p(n) \) space on input of size \( n \). A well-known property of polynomial-space alternating Turing machines is that every computation halts in deterministic time exponential in the size of the input [6, 14]. Let \( p'(n) \) be so large that \( 2^{p'(n)} \) exceeds the time bound of \( A \) on input of size \( n \), and let \( \Sigma \) be the finite tape alphabet of \( A \). We can assume, without loss of generality, that \( A \) begins in an existential state, existential and universal states alternate at every step, and when \( A \) enters an accepting state it continues to take steps while
staying in accepting states. Furthermore, we can assume that \( A \) has exactly two possible
moves at every step, every existential state has at least one immediate successor that
is a rejecting universal state, every universal state has at least one immediate successor
that is an accepting existential state, and the final state of every computation is an
existential state.

For any input \( x \), we first construct a polynomial-size Petri net \( \text{net}(A_x) \) that "simulates"
the computation of \( A \) on \( x \). Each tape square \( i \) of \( A \) is represented as a group of places
\( \{s(i,a_1), \ldots, s(i,a_k)\} \cup \{s(i,q_0), \ldots, s(i,q_l)\} \), where \( \Sigma = \{a_1, \ldots, a_k\} \) and \( \{q_0, \ldots, q_l\} \) are
the control states of \( A \). The idea is that for each tape square \( i \), exactly one of the places
in \( \{s(i,a_1), \ldots, s(i,a_k)\} \) will be marked under every reachable marking, indicating which
tape symbol is currently written on tape square \( i \). Furthermore, over all \( 1 \leq i \leq p(n) \)
and all \( 0 \leq j \leq l \), exactly one of \( s(i,q_j) \) is marked, indicating which tape square holds
the head and which control state \( A \) is currently in. Let \( x = a_{i_1} \ldots a_{i_p}; \) then exactly the
places \( \{s(1,a_{i_1}), \ldots, s(n,a_{i_p})\} \cup \{s(1,q_0)\} \) are initially marked.

The net \( \text{net}(A_x) \) is wired up as follows: for every tape square \( i \), every control state \( q \),
every symbol \( a_j \in \Sigma \), and every control transition \( (q', a_j, D) \in \delta(q, a_j) \) in \( A \), where \( D \)
is either \( L \) or \( R \), \( \text{net}(A) \) contains a transition \( t_{(q,a_j)}(q',a_j,D) \), labeled with some common
label 1. The idea is that this transition fires iff \( A \) is currently in control state \( q \) and
tape square \( i \) holds the head and contains \( a_j \). Firing this transition puts \( A \) in control
state \( q' \), writes \( a_j \) on tape square \( i \), and moves the head to tape square \( i - 1 \) if \( D = L \)
and to tape square \( i + 1 \) if \( D = R \). In particular, the preset of transition \( t_{(q,a_j)}(q',a_j,D) \)
is \( \{s(i,q), s(i,a_j)\} \) and the post-set is \( \{s(i-1,q'), s(i,a_j)\} \) or \( \{s(i+1,q'), s(i,a_j)\} \) depending on
whether \( D \) is \( L \) or \( R \). Finally, for every accepting existential control state \( q \) and tape
square \( i \), we introduce a transition \( X_{(i,q)} \) with preset \( \{s(i,q)\} \), empty postset, and label \text{acc}. For every rejecting existential control state \( q \) and tape
square \( i \), we introduce a transition \( Y_{(i,q)} \) with preset \( \{s(i,q)\} \), empty postset, and label \text{acc}, and a transition \( Y_{(i,q)} \) with preset \( \{s(i,q)\} \), empty postset, and label \text{rej}. Clearly, \( \text{net}(A_x) \) contains \( (k + l) \cdot p(n) \)
places and at most \( (2l + m) \cdot p(n) \) transitions, where \( k \) is the size of the tape alphabet of
\( A \), \( l \) is the number of control states of \( A \), and \( m \) is the number of control transitions of \( A \).

It is straightforward to show that \( \text{net}(A_x) \) is 1-safe, sequential (i.e., no transitions can
fire concurrently under any reachable marking), and that its labeled transition system
is isomorphic to that of \( A \) on input \( x \), ignoring the labels of the control transitions, and
ignoring the \text{acc}-labeled and \text{rej}-labeled transitions altogether.

Let \( T \) be the deterministic Turing machine which, started with a string of 0's on its
tape, successively adds 1 to the binary number on its tape until the original string of 0's
is changed into a string of 1's (of the same length). Then \( T \) enters an accepting state
and halts. So, when started on a string on \( m \) 0's, it runs for at least \( 2^m \) steps and halts.

The polynomial-time translation \( \text{net} \) given above for alternating Turing machines also
holds for any deterministic polynomial-space Turing machine, except that we add both
\text{acc}-labeled and \text{rej}-labeled transitions for every pair \( (i, q) \). Hence, if "started" on input
consisting of a string of \( p'(|x|) \) 0's, this net is of size bounded by some polynomial in
\( |x| \), and has the sole behaviors that it fires at most some fixed \( m' \) > \( 2^{p'(|x|)} \) number of
1's, and each point along the way it nondeterministically chooses between firing either acc or rej and exiting, or firing a 1. Furthermore, after firing \( m' \) 1's followed by a single acc or rej, it reaches a deadlocked state. We call this net Count\((m')\). We can assume, without loss of generality that \( m' \) is odd, and since \( m' \) exceeds the time bound of \( A \) on input \( x \), we can assume, without loss of generality, that every computation path of \( A \) on input \( x \) is exactly of length \( m' \).

To finish the construction, let \( N_F \) be a finite 1-safe net of constant size with the labeled transition system pictured in Fig. 4, and let \( N_x \coloneqq N_F \parallel_{1, \text{acc, rej}} \text{Count}(m') \), where synchronization is required on the symbols 1, acc, and rej. \( N_x \) is of size polynomial in \(|x|\), and its labeled transition system is bisimilar to the transition system pictured in Fig. 5.

We now show that \( \text{net}(A_x) \) is bisimilar to the net \( N_x \) iff \( A \) accepts input \( x \). For one direction, suppose that \( \text{net}(A_x) \) is bisimilar to \( N_x \); then \( \text{net}(A_x) \) must have some \( m' \)-length path bisimilar to \( \exists(a) \forall(a) \exists(a) \forall(a) \ldots \exists(a) \) after which it fires an acc-labeled transition. Thus, all the states of \( \text{net}(A_x) \) that are reached along the way must be accepting. Since the labeled transition system of \( \text{net}(A_x) \) is essentially isomorphic to the labeled transition system of \( A \) on \( x \), \( A \) must accept \( x \). Recalling our assumptions on \( A \), the other direction follows by a simple induction on \( \approx_i \), where \( \approx_i \) is an \( i \)-step bisimulation (cf. [16]). This is a polynomial-time reduction from the acceptance problem for polynomial-space alternating Turing machines to bisimulation of finite nets.

It is well-known that the class of problems decidable in polynomial space by alternating Turing machines is the same as the class of problems decidable in deterministic exponential time by ordinary Turing machines [6, 14]. We then have as a simple corollary of this fact and Theorem 5.7.

**Theorem 5.8.** Strong bisimulation of finite nets is \textsc{dexpptime-hard}.

We now show the lower bounds for the remaining equivalences listed in Table 1.

![Fig. 4. Labeled transition system of \( N_F \).](image-url)
Theorem 5.9. For finite nets,
1. trace equivalence is polynomial-time reducible to step-trace equivalence, ST-trace equivalence, interval pomset-trace equivalence, and pomset-trace equivalence;
2. failures equivalence is polynomial-time reducible to step-failures equivalence, ST-failures equivalence, and interval pomset-failures equivalence; and
3. strong bisimulation is polynomial-time reducible to weak bisimulation, delay bisimulation, branching bisimulation, step-bisimulation, ST-bisimulation, history-preserving bisimulation, maximality-preserving bisimulation, pomset-bisimulation, and pomset-ST-bisimulation.

Proof. For the true concurrency equivalences, we give the proof only for pomset-trace equivalence, as the other cases are completely analogous. For any finite nets $N_1, N_2$ without hidden transitions, let $N'_1$ be constructed by adding to $N_i$ a single new, initially marked place which is placed in the preset and post-set of every transition of $N_i$. Clearly, $N'_1$ is trace equivalent to $N_i$. Since no transitions in $N'_1$ are statically concurrent, it is easy to see that $N'_1$ and $N'_2$ are trace equivalent iff they are pomset-trace equivalent; hence, $N_1$ and $N_2$ are trace equivalent iff $N'_1$ and $N'_2$ are pomset-trace equivalent. This is a polynomial-time reduction from trace equivalence to pomset-trace equivalence.

For the interleaving bisimulations, we note that weak bisimulation, delay bisimulation and branching bisimulation coincide with strong bisimulation for nets without hidden
transitions. Thus, relabeling all $\tau$-labeled transitions with the same "fresh" visible label completes the reduction. □

We then have as a simple corollary of Theorems 5.4, 5.6, 5.8, and 5.9.

**Theorem 5.10.** For finite nets, the decision problems for

1. step-trace equivalence, ST-trace equivalence, interval pomset-trace equivalence, and pomset-trace equivalence are EXPSPACE-hard,
2. step-failures equivalence, ST-failures equivalence, and interval pomset-failures equivalence are EXPSPACE-hard,
3. weak bisimulation, delay bisimulation, branching bisimulation, step-bisimulation, ST-bisimulation, history-preserving bisimulation, maximality-preserving bisimulation, pomset-bisimulation, and pomset-ST-bisimulation are DEEXPSPACE-hard.

We remark that all the lower bound results in this section are independent of the presence of hidden transitions, except as specifically stated in the lower bound proofs for trace equivalence.

6. Conclusions

We remark that all these complexity results apply equally to process approximation as well as equivalence. An open problem is the decidability and complexity of augmentation-closed pomset-trace equivalence. Another open problem that we regard as especially significant is the decidability and complexity of our earlier general pomset-failures semantics [11], which keeps track of concurrent divergences. We are currently working to extend our methods to handle these cases.

Acknowledgements

We are grateful to Alex Rabinovich for some very helpful discussions, especially of lower bound proofs, and to Alain Mayer and Larry Stockmeyer for related discussions. We also thank Rob van Glabbeek for useful suggestions and information regarding previously known results, and Carl Gunter, John Mitchell, Mogens Nielsen, and Paul Taylor for helpful advice about terminology.

References