Article

Depth-First Net Unfoldings and Equivalent Reduction

Xu Yang 1,2,*, Chen Ye 2,*, and Yijun Chen 3,*

1 Department of Computer Science and Technology, College of Electronic and Information, Tongji University, Shanghai 201804, China
2 The Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai 201804, China
3 National Maglev Transportation Engineering R&D Center, Tongji University, Shanghai 201804, China
* Correspondence: 1610482@tongji.edu.cn (X.Y.); yechen@tongji.edu.cn (C.Y.); chenyijun@tongji.edu.cn (Y.C.);
Tel.: +86-18616815297 (X.Y.); +86-21-69580135 (Y.C.)

Abstract: In Petri net unfolding, according to the strategies of breadth first and depth first, the biggest problem lies in the potential explosion of the state space. Unfolding generates either accessible trees or branch processes. Making marking reduction or branch cutting accessible proves to be an effective approach to mitigating the state space expansion. In this paper, we propose three reduction rules based on similarity equivalence, conduct state space reduction, present three theorems supported by a case study, and propose a new unfolding algorithm for the unfolding process. In both the new case and the experiments, the completeness, optimality, completeness, and memory and time consumption are reduced by about 60%.

Keywords: Petri net; unfolding; partial order reduction; depth first; breadth first

1. Introduction

Exploring the path of a concurrent system [1–4] may lead to a state space explosion. Partial order semantics [2,5–11] and prime event structure have proven to be efficacious techniques in alleviating state space explosions during traversal. The concurrency of the Petri net is a valuable tool for concurrent system modeling. Petri net unfolding technology effectively combines partial sequence semantics and prime event structure. However, it can produce state space explosions during the unfolding process and reach a level of proximity at the factorial (n!) level when traversing paths, with n representing the number of changes. The tireless pursuit of scholars in this field is finding ways to reduce the state space, decrease the storage space, and simplify computational complexity. This paper analyzes the state space reduction and the similarity of change to induce a substantial reduction in the state space. Exploring all conceivable paths of a system that are not exhaustive, as well as executing equivalent path reduction, comprise a viable solution to the problem.

Some scholars have made contributions in terms of state space reduction and similarity equivalence. For instance, Jensen [12] applied symmetry to reduce the state space of the CPN (color Petri net) using limited conditions, especially considering only fully symmetrical properties. Chiola [13] made a similar reduction, but also according to the symmetry factor of the emission. Schmidt [14] studied the symmetry reduction in the context of the Petri network. Junttila [15] studied the complexity of the question. However, how to reduce the state space during the unfolding process remains a persisting problem that remains to be solved.

Petri net unfolding [12–20] is a concurrency theory that is based on semantics proposed by McMillan. This theory has been extended to support nets with read arcs and applied in the context of both model checking and path traversal in concurrent systems. Prime event structures are utilized to analyze the partial ordering relationship between events, thereby facilitating efficient model checking or path analysis. The starting point of this work is in devising strategies to reduce state space expansion and conserve resources in the process.
This study explores a transition system (in the context of a bounded net), which unfolds into a tree exhibiting the same behavior. The unfolding of the net can be equivalently conceptualized in terms of true concurrency. The deployment process involves non-deterministic selection or marking a cutout event when the same transition rule is encountered, until no further transitions are possible. The unfolding follows a partial order, and as each new transition may be added, the order of selection changes, leading to a different maximum expansion.

To determine whether a specific place $p$ in the net $N$ can be marked, the unfolding process can be extended. This involves adding a marker $Mt$ to each new transition $t$ as it unfolds. When triggering transition $t$, the process checks whether $Mt$ has already occurred. If it has, $Mt$ is considered a cut-off node and is not executed. The enabled transitions are repeatedly triggered in a cyclic manner until no further transitions are possible in the net, indicating successful and complete deployment.

Otherwise, the algorithm examines whether an earlier added transition $t'$ fulfils $Mt' = Mt$; in such an instance, it designates $t$ as a cut-off node, signifying that this search branch will no longer be explored. The search concludes without success when no additional transition can be incorporated. In the process of unfolding, it is crucial to implement effective measures for state space reduction. The similarity equivalence can effectively reduce the state space while lowering the computational complexity and memory consumption.

As has been widely acknowledged in the literature, the correctness of the search—i.e., the assurance that the search will always terminate with the correct result—is highly dependent on the strategy employed [6]. Several papers [6,7,12] have proposed breadth-first strategies that have been shown to be correct.

Various strategies can be used to conduct the search, with breadth-first and depth-first strategies being the most commonly employed. Figure 1 depicts the paths of the deployment net using depth-first and breadth-first strategies, respectively.

![Figure 1. (a) Depth-First Event Sequence. (b) Breadth-First Event Sequence.](image-url)
by a case study of the unfolding net, thereby establishing the correctness of the depth-first search algorithm. In Section 4, three reduction strategies are proposed and implemented to reduce the unfolded net, resulting in the final reduced equivalent net. Finally, Section 5 provides a detailed discussion on the results.

2. Unfolding of Petri Net

The following section serves to introduce the net and fundamental concepts of unfolding [1,12,16,18,21–30]. Readers acquainted with this subject matter may proceed to the subsequent section.

Definition 1. (Net) A triple \( N = (P, T, F) \) is called a net if and when:

1. \( P \) and \( T \) represent finite sets of places and transitions, respectively;
2. \( F \subseteq (P \times T) \cup (T \times P) \) arcs set;
3. \( P \cup T \neq \emptyset \) and \( P \cap T = \emptyset \).

If \( N = (P, T, F) \) is a network, node \( x \in P \cup T \), the predecessor \( x \) and successor \( x \) can be defined as:

1. \( x = \{ y \in P \cup T \mid (y, x) \in F \} \);
2. \( x = \{ y \in P \cup T \mid (x, y) \in F \} \).

Among them, \( \forall x \in P \cup T : x \cup x \neq \emptyset \).

Definition 2. (Reachability) \( \sum = (P, T, F, M_0) \) as a Petri net:

1. If there is a transition \( t \) and the markings \( M \) and \( M' \) satisfy \( M t M' \), then the marking \( M' \) from \( M \) is directly reachable;
2. If there is a sequence of transition occurrence \( \sigma = t_1 t_2 t_3 \ldots t_n \), \( M t_1 M_2 t_2 \ldots M_{n-1} t_n M'' \), that is, \( M(\sigma) M'' \), then the marking \( M'' \) from \( M \) is reachable. All sets of identifiers reachable from the marking \( M \) are denoted as \( R(M) \).

In a net \( N \), nodes \( x, y \in P \cup T \):

1. \( x \) and \( y \) belong to a causal relationship, denoted as \( x \leq y \), if and only if there is a path from \( x \) to \( y \) in \( N \); if \( x \neq y \), it is recorded as \( x < y \);
2. \( x \) and \( y \) belong to a conflicting relationship, denoted as \( x \# y \), if and only if \( \exists t_1, t_2 \in T : t_1 \cap t_2 \neq \emptyset \land t_1 \leq x \land t_2 \leq y \);
3. \( x \) and \( y \) belong to a concurrent relationship, denoted as \( x \co y \), if and only if they satisfy \( \neg (x < y \land y < x \land x \# y) \), that is, \( x \) and \( y \) are neither causal nor conflicting.

Definition 3. (Occurrence net) A net \( N = (P, T, F) \) is called an occurrence net if and when:

1. \( \forall x, y \in P \cup T : x < y \Rightarrow y \not< x \);
2. \( \forall p \in P : |p| \leq 1 \);
3. There is no self-conflicting change, that is, \( \forall t \in T : \neg(t \# t) \).

Definition 4. (Branch process) Let Petri net \( \sum = (P, T, F, M_0) \) and occurrence net \( O = (B, E, G) \) and the homomorphic function \( h: B \cup E \rightarrow P \cup T \) simultaneously satisfy:

1. \( h(B) \subseteq P \land h(E) \subseteq T \);
2. For any event \( e \in E \), the \( h \) function acting on \( e \) (resp., \( e^* \)) satisfies the bijection of \( e \) to \( h(e) \) (resp., \( e^* \) and \( h(e^*) \));
3. The \( h \) function acting on \( \text{Min}(O) \) is also limited to the bijection between \( \text{Min}(O) \) and \( M_0 \);
4. For any event \( e_1, e_2 \in E \), if \( e_1 = e_2 \) and \( h(e_1) = h(e_2) \), then \( e_1 = e_2 \).

Definition 5. (Configuration) The configuration set \( C \) of a branch process can represent the possible set of events that a Petri net may run, which requires the following conditions to be met:

1. Causal closure: \( e \in C \Rightarrow \forall e' \leq e : e' \in C \);
2. No conflict: \( \forall e, e' \in C : \neg(e \# e') \).
Definition 6. (Possible extensions) For configuration $C$ and event set $E$, $C \oplus E$ represents an extension of $C$, if and when $C \cup E$ is a configuration and $C \cap E = \emptyset$.

Definition 7. (Completeness) The branching process $\beta$ of a Petri net is complete, and there exists a configuration $C \cup \{e\}$ for any reachable identifier $M$ and its enabling transition $t$ (i.e., $M \mid t$), and event $e$ satisfies $\text{Marking}(C) = M \wedge e \notin C \wedge t = h(e)$.

Definition 8. (Total order) If the partial order relationship $<_e$ acting on a finite configuration satisfies total order, then the following conditions must be met:

1. $<_e$ satisfies the order relationship;
2. $C_1 \subset C_2 \Rightarrow C_1 <_e C_2$;
3. If $C_1 <_e C_2$ and $\text{Marking}(C_1) = \text{Marking}(C_2)$, then there is an isomorphic relationship $I_1$ and an extended event set $E'$ to satisfy $C_1 \oplus E' < C_2 \oplus I_1(E')$.

Definition 9. (Cut-off event) Let $\beta$ be a prefix for network expansion, with $e_1$ and $e_2$ being two of the events, and satisfy $\text{Marking}(\{e_1\}) = \text{Marking}(\{e_2\})$. If $[e_1] < [e_2]$, then $e_2$ is called a cut-off event.

3. Net Unfolding
   In this section, we provide an informal but precise definition of unfolding. For formal definitions, readers may refer to [7].

   Classic unfolding algorithm (Algorithm 1):

   **Algorithm 1: ERV Net Unfolding Algorithm** [12]

   ```
   input: Petri net $\Sigma = (P, T, F, M_0)$
   output: Finite complete prefix $\text{Fin}$.
   1: $\text{Fin} = \text{Min}(\text{Linf}(\Sigma))$;
   2: $\text{Poe} = \text{PE}(\text{Fin}); /*\text{Possible Extension Set}*/$
   3: $\text{Cut-off} = \emptyset; /*\text{Cut-off Event Set}*/$
   4: while $\text{Poe} \neq \emptyset$ do
   5:     Select the event $e$ with the smallest $<_e$ relationship from $\text{Poe}$;
   6:     if $[e] \cap \text{Cut-off} = \emptyset$, then
   7:         Expanding $e$ and $e^* \cup e$ into $\text{Fin}$;
   8:         $\text{Poe} = \text{PE}(\text{Fin})$;
   9:     end if
   10:    if $e$ is cut-off event, then
   11:         $\text{Cut-off} = \text{Cut-off} \cup \{e\}$
   12:    end if
   13:    else $\text{Poe} = \text{Poe} \setminus \{e\}$;
   14: end if
   15: end while
   ```

   The net used in this article is shown in Figure 2 as a secure net.

   ![Figure 2. A Petri net $\text{SecN}$](image-url)
4. Depth-First Unfolding

Upon net expansion, the process commences with a place for each element of MI, originating from a state. For the Petri net N depicted in Figure 2, MI = {a, b, c, d}.

We proceed to generalize the concept of potential extension: if the current labeled net permits reaching a marking m, labeled by a marking M of the original net, and M enables a transition t culminating in a marking M', then the unfolding is augmented with a new event labeled by t, and for each output place p of t in N, a new place labeled by p is added.

Depth-first unfolding algorithm (Algorithm 2; conditions are mapped to places, and events to transitions):

Algorithm 2: The Depth-First Unfolding Algorithm

| input: a Petri net $\Sigma = (N, M_0)$ |
| output: $\pi$ – a complete prefix of $\text{Unf}^{\max}_{\Sigma}$ |
| 1. $\pi := \text{the empty branching processes}$ |
| 2. Push all init of the places from $M_0$ to $\pi$ |
| 3. $\text{cut} \_\text{off} := M_0$ |
| 4. $i = 0$; // Record the subscript of the event |
| 5. $pe := \text{PoTEXT}(\pi, \text{cut} \_\text{off})$ //Enabled events |
| 6. while $pe \neq \emptyset$, do |
| 7. pop $e \in pe$ //using stack for data structure; |
| 8. $i = i + 1$; |
| 9. fire $e$ title with $e_i$ (Trigger transition event subscript plus 1)) |
| 10. add new instances from $q(e)^* \Sigma$ to $\pi$ |
| 11. (If similarity (new instances) does not exist in $\pi$, new instances of places Superscript $j++$; |
| 12. else, the superscript $j$ is 0 (0 means there is no superscript) // Counting of newly |
| generated library superscripts starting from 0) |
| 13. if $I(C(\pi \cup \{e\}) \text{then cut} \_\text{off} := \text{cut} \_\text{off} \cup \{e\}$ // Become a new node and cut it the |
| next time it appears |
| 14. else |
| $pe := pe + \text{PoTEXT}(\pi, \text{cut} \_\text{off})$ |
| // Become a cutting event (if it already exists, it is a cutting node); |
| 15. End while |

In this unfolding algorithm, $\pi$ represents the complete prefix of expansion. When line 2 is a place with a token, line 5 calculates the possible changes. When line 6 is not empty, one of the changes occurs, using the stack data structure. Line 9 triggers the enabled event, line 10 adds a new marking, and line 11 determines whether a new marking is added. Meanwhile, markings that are deemed equivalent in terms of similarity are considered identical, otherwise line 12 becomes a cut-off node. Line 13 adds a new cut-off node, and line 14 continues to judge the change events that can be started under the new marking until line 15 has no event to be enabled. When unfolding, it is necessary to realize the state space reduction, and this can be accomplished through the state similarity equivalence method. We perform the state similarity equivalence method to reduce the number of traversed paths. To illustrate the concept of similarity equivalence, as shown in Figure 3, we treat the two identifications, M1 and M2, as equivalent in the algorithm.

The complexity analysis for Algorithm 2 (the depth-first unfolding algorithm) achieves a polynomial complexity with $N$ nodes and $m$ places; at most, the complexity is at the level of $n \times m$. Using the depth-first unfolding algorithm to unfold the Petri net in Figure 2 is detailed in Table 1.

Figure 4 demonstrates a graphical representation of the unfolding process of SecN. These unfolding prefixes are referred to as such. The initial marking bearing with label {a, b, c, d} enables transitions A and B, resulting in two feasible extensions. We opt to incorporate event e1 first, labeled by A. The prefix now possesses a novel reachable marking, corresponding to marking {c, d, i, k} in SecN, which paves the way for a new potential extension labeled C. Let us assume event e2, labeled by B, is added subsequently.
The prefix now comprises two additional reachable markings: \([a, b, j, l]\) and \([i, j, k, l]\). These markings facilitate extensions labeled with \(D\) and \(T\), respectively. After event \(e_3\), labeled by \(C\), a new marking labeled by \([c, h, i]\) emerges, enabling possible extensions with labels \(E\) and \(H\), etc.

These algorithms construct progressively larger prefixes of the unfolding of \(S\). Similar to transition systems, they examine one event at a time, with some events designated as cut-offs, beyond which successors are no longer explored.

The challenge lies in generalizing the definition of cut-off events for Petri nets. McMillan [14] proposed a solution. The crux is to associate each event \(e\) with an appropriate reachable marking \(Me\) of the original Petri net \(S\), accomplished in three steps, exemplified by event \(e_3\) in Figure 4:

- Determine the set \([e_n]\) comprising all predecessors of \(e_n\), i.e., the set of all events \(e_m\) such that the unfolding includes a path from \(e_m\) to \(e\). In this case, \([e_3] = [e_1, e_3]\);
- Select any occurrence sequence \(s_r\) containing each element of \([e]\) exactly once (which is guaranteed to exist) and allow it to occur. Here, \(s_r = e_1 e_3\);
- Let \(m\) denote the marking of the unfolding reached by firing \(s_r\) (which can be proven to be independent of the choice of \(s_r\)) and define \(Me\) as the label of \(m\). In this instance, \(m = [i, e, h]\) and \(Me3 = [i, e, h]\).
For each novel event $e_{xf}$, the algorithms compute and store the marking $M_{ext}$. It is worth noting that these are the sole markings of SecN known to be reachable by the algorithms.

Conditions (1) and (2) can now be effortlessly generalized. An event $e_{cut}$ is designated as a cut-off if marking $Me$ satisfies one of these conditions:

(a) $Me(pT) = 1$; the algorithm concludes with the result “reachable”;
(b) $Me'$ is already known to be reachable: either $Me = MI$ or $Me' = Me$ for some other event $e'$. In such a case, $e'$ is referred to as the corresponding event of $e$.

The unfolding prefix in Table 1 exhibits a reachable marking, corresponding to $\{a, c, b, d\}$. As this is equivalent to $MeI$, such an extension would constitute a cut-off.

![Figure 4. A prefix of the unfolding of the net in Figure 2.](image)

5. Equivalence Reduction of Unfolded Nets

Table 1, presented above, has not been reduced, and following unfolding, it remains excessively large. In Table 1, distinct colors represent different states, and this article employs various colors for differentiation. For instance, initial states $a$, $b$, $c$, and $d$ are represented in black, while newly generated states $c$, $d$, $i$, and $k$ are depicted in green. Reduction of a net utilizing equivalence [31–33] is explored.

For this unfolding net, research has determined that the unfolding net remains overly extensive. This paper employs equivalence-based reduction. The reduction process presents as follows:

For the initial reduction, four rectangles of different colors are reduced based on the same transition (e.g., $G$, $G'$; $H$, $H'$; $E$, $E'$; $F$, $F'$ represent identical transitions) guarantee.

Reduction rule: If the same marking $M$ undergoes the same transformation and yields the same marking $M'$, it is considered a cut-off node. After one reduction, the results are shown in Figure 5.
Figure 5. The same transition reduction guarantee.

The reduced graph is as follows in Figure 6.

Figure 6. Reduction result of the same transition and next guarantee.

Following the second reduction, when only a single transition remains (i.e., no branching path), we implement the reduction. After two alterations have transpired, the subsequent changes that can occur are all \( t_j \). For instance, enabling transitions \( t_i \) and \( t_j \), regardless of whether \( t_i \) occurs or \( t_j \) occurs first, the transition that can be triggered after occurrence is \( t_k \). In lines 9 and 10 in Table 2, both of which are transitions \( T \) that occur, the resulting states are the same; hence, reduction can be performed.

Table 1, presented above, has not been reduced, and following unfolding, it remains excessively large. In Table 1, distinct colors represent different states, and this article emphasizes that these are the sole markings of SecN known to be reachable by the algorithms. As this is equivalent to \( Me = \{i, e, h\} \) and \( Me = 1 \), the algorithm concludes with the result "reachable".
Table 2. Results of the first reduction in the expansion of the net in Figure 2.

<table>
<thead>
<tr>
<th>No</th>
<th>Marking</th>
<th>Enabled Transfer</th>
<th>Selected</th>
<th>New Marking</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a, b, c, d</td>
<td>A, B</td>
<td>A/ε1(a, b-&gt;i, k)</td>
<td>c, d, i, k</td>
<td>new</td>
</tr>
<tr>
<td>2</td>
<td>a, b, c, d</td>
<td>B</td>
<td>B/ε2(b-&gt;l, j)</td>
<td>a, b, l, j</td>
<td>new</td>
</tr>
<tr>
<td>3</td>
<td>c, d, i, k</td>
<td>C, B</td>
<td>C/ε3(c, d-&gt;e, h)</td>
<td>i, e, h</td>
<td>new</td>
</tr>
<tr>
<td>4</td>
<td>c, d, i, k</td>
<td>B</td>
<td>B'/ε4(c, d-&gt;l, j)</td>
<td>i, k, l, j</td>
<td>new</td>
</tr>
<tr>
<td>5</td>
<td>a, b, l, j</td>
<td>A, D</td>
<td>A'/ε5(b, c-&gt;e, h)</td>
<td>i, e, h</td>
<td>new</td>
</tr>
<tr>
<td>6</td>
<td>a, b, l, j</td>
<td>B</td>
<td>B'/ε6(b, c-&gt;l, j)</td>
<td>i, k, l, j</td>
<td>new</td>
</tr>
<tr>
<td>7</td>
<td>i, e, h</td>
<td>E, H</td>
<td>E/ε7(e, i-&gt;a, c)</td>
<td>h, a', c'</td>
<td>new</td>
</tr>
<tr>
<td>8</td>
<td>i, e, h</td>
<td>H</td>
<td>H/ε8(h-&gt;b, d)</td>
<td>i, e, b', d'</td>
<td>new</td>
</tr>
<tr>
<td>9</td>
<td>l, k, l', j'</td>
<td>T</td>
<td>T'/ε9(l, k, l-&gt;p)</td>
<td>P</td>
<td>end</td>
</tr>
<tr>
<td>10</td>
<td>i', k', l, j</td>
<td>T'</td>
<td>T'/ε10(l, k, l-&gt;p)</td>
<td>P</td>
<td>end</td>
</tr>
<tr>
<td>11</td>
<td>f, g, j</td>
<td>E, G</td>
<td>F/ε11(f-&gt;a, c)</td>
<td>a'', c'', g, j</td>
<td>new</td>
</tr>
<tr>
<td>12</td>
<td>f, g, j</td>
<td>G</td>
<td>G/ε12(g-&gt;b, d)</td>
<td>f, b'', d''</td>
<td>new</td>
</tr>
</tbody>
</table>

Upon the completion of the two transitions \( t_i \) and \( t_j \), the subsequent transition that can occur is \( t_k \), that is, enabling transitions \( t_i \) and \( t_j \), irrespective of whether \( t_i \) occurs first or \( t_j \) occurs first, the transition that can be triggered.

The third step of the reduction rule: if the same marking \( M \) undergoes corresponding transformations and produces an equivalent marking \( M' \), it is considered a cut-off node. As illustrated in Figure 7, distinct transitions yield different identifiers, but analogous identifiers are generated. The unfolding paragraph after the final reduction is displayed in Figure 8.

Theorem 1. If marking \( A \) reaches marking \( C \) after transition \( t_1 \), and marking \( B \) reaches marking \( C \) after transition \( t_2 \); \( t_1 \) is equivalent to \( t_2 \).

Proof. If Marking \( C = Marking C \), i.e., Mark \( (t_1) = Mark \ (t_2) \), and then \( t_1^* = t_2^* \), \( h(t_1^*) = h(t_2^*) \), namely \( h(t_1^*) = h(t_2^*) \). □

Theorem 2. If marking \( A \) reaches marking \( C \) after transition \( t_1 \), and marking \( B \) reaches marking \( D \) after transition \( t_2 \), and marking \( C \) is equivalent to marking \( D \), then \( t_1 \) is equivalent to \( t_2 \).
Proof. If $\text{Marking } C = \text{Marking } D$, i.e., $\text{Mark } ([t_1]) = \text{Mark } ([t_2])$, and then $t_1^* = t_2^*$, $h(t_1^*) = h(t_2^*)$, namely $h(t_1^*) = h(t_2^*)$. □

Definition 10. $\text{Marking } C$ is approximately equivalent to $\text{marking } D$, that is, the transitions that can occur under $\text{marking } C$ are equal to the transitions that can occur under $\text{marking } D$.

Theorem 3. $\text{Marking } A$ reaches $\text{marking } C$ after the transition $t_1$, and $\text{marking } B$ reaches $\text{marking } D$ after the transition $t_2$, and $\text{marking } C$ is approximately equivalent to $\text{marking } D$, so $t_1$ is equivalent to $t_2$.

Proof. If $\text{Marking } C \approx \text{Marking } D$, i.e., $\text{Mark } ([t_1]) = \text{Mark } ([t_2])$, and then $t_1^* = t_2^*$, $h(t_1^*) = h(t_2^*)$, namely $h(t_1^*) = h(t_2^*)$. □

Figure 7. Second reduction result and the third Reduction Rule.

Figure 8. Unfolding paragraph after the final reduction.

6. Case Study and Experiments Result

This paper employs a case study to scrutinize the process of net reduction through unfolding, predicated on the three reduction rules posited herein. Figure 9a illustrates a Petri net, while Figure 9b portrays the unfolded net. Subsequently, this article puts forth a case study to delineate three reduction rules applicable to the unfolding net reduction procedure. Table 3 shows the result of net reduced unfolding.

Table 3. Results of Net unfolding and Reduction in Figure 2.

<table>
<thead>
<tr>
<th>No</th>
<th>Init</th>
<th>Enabled</th>
<th>Select</th>
<th>New</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a, b, c, d</td>
<td>A, B</td>
<td>$A/e_{1(a, b-&gt;i, k)}$</td>
<td>c, d, i, k</td>
<td>New</td>
</tr>
<tr>
<td>2</td>
<td>a, b, c, d</td>
<td>B</td>
<td>$B/e_{2(c, d-&gt;l, j)}$</td>
<td>a, b, l, j</td>
<td>New</td>
</tr>
<tr>
<td>3</td>
<td>c, d, i, k</td>
<td>C, B</td>
<td>$C/e_{3(k, c, d-&gt;e, h)}$</td>
<td>i, e, h</td>
<td>New</td>
</tr>
<tr>
<td>4</td>
<td>a, b, l, j</td>
<td>D</td>
<td>$D/e_{6(a, b, l-&gt;f, g)}$</td>
<td>f, g, j</td>
<td>New</td>
</tr>
<tr>
<td>5</td>
<td>i, e, h</td>
<td>H</td>
<td>$H/e_{8(i-&gt;b, d)}$</td>
<td>i, e, b', d'</td>
<td>New</td>
</tr>
<tr>
<td>6</td>
<td>l, k, i', j'</td>
<td>T</td>
<td>$T/e_{9(k, l, i')-&gt;p}$</td>
<td>P</td>
<td>end</td>
</tr>
<tr>
<td>7</td>
<td>f, g, j</td>
<td>F, G</td>
<td>$F/e_{11(f-&gt;a, c)}$</td>
<td>a'', c'', g, j</td>
<td>Init</td>
</tr>
</tbody>
</table>
The first step of reduction is premised on (1) identical marking \( M \), following the same transition, yielding the same marking \( M' \), regarded as the cut node or, more precisely, the cut-off node.

The second step of reduction is based on (2) the similar transition of marking \( M \), engendering a marking \( M' \) with an identical structure and functioning as a cut-off node, as demonstrated in Figure 10a.

![Figure 9. (a) Original Net. (b) unfolded Net and Reduction Rule 1.](image)

![Figure 10. (a) First step of reduction and reduction rule 2. (b) Second step of reduction and reduction rule 3.](image)
The third step of reduction adheres to the subsequent criterion: (3) identical marking \( M \), upon corresponding alterations, generates an equivalent marking \( M' \), deemed as a cut-off node. The final equivalent unfolding net is exhibited in Figure 11.

![Figure 11. Final equivalent unfolding net.](image)

We added more experiments to show the scalability of the proposed scheme and a comparison with related state-of-the-art works. Otherwise, it is difficult to evaluate the contributions of this work.

The algorithm satisfies the following three properties:

(i) **Soundness**: each complete path explored by the algorithm is a consistent path of the Petri net;

(ii) **Completeness**: the algorithm explores all consistent paths of the Petri net;

(iii) **Optimality**: each path is explored exactly once.

Based on the open-source tool PIPE (Platform Independent Petri Net Editor), both experiments in this section are conducted on machines with an Intel Core i7-6400 CPU (5.10 GHz) and 64.0 G memory. Each test case set requires 10 experiments with the average of the total number. As shown in Table 4, “Exe” represents the number of executed paths, “M” represents memory consumption, “Time” indicates time consumption in seconds, and “Accuracy” denotes the acceleration ratio.

<table>
<thead>
<tr>
<th></th>
<th>ERV Unfolding</th>
<th>Similarity Unfolding</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Exe</td>
<td>M</td>
<td>Time</td>
</tr>
<tr>
<td>SecN (10)</td>
<td>3328</td>
<td>46</td>
<td>0</td>
</tr>
<tr>
<td>SecN (15)</td>
<td>147,456</td>
<td>46</td>
<td>9</td>
</tr>
<tr>
<td>SecN (20)</td>
<td>6,029,312</td>
<td>46</td>
<td>455</td>
</tr>
<tr>
<td>PeN (8)</td>
<td>10,080</td>
<td>182</td>
<td>1</td>
</tr>
<tr>
<td>PeN (16)</td>
<td>oom</td>
<td>oom</td>
<td>oom</td>
</tr>
<tr>
<td>exp-p (9)</td>
<td>oom</td>
<td>oom</td>
<td>oom</td>
</tr>
<tr>
<td>exp-p (4)</td>
<td>20,736</td>
<td>55</td>
<td>1</td>
</tr>
<tr>
<td>exp-p (5)</td>
<td>705,600</td>
<td>326</td>
<td>58</td>
</tr>
<tr>
<td>exp-p (6)</td>
<td>oom</td>
<td>33</td>
<td>177</td>
</tr>
</tbody>
</table>
7. Conclusions

In the verification of the concurrent system based on Petri nets, it is inevitable to encounter a state space explosion during the unfolding of the net, and during path exploration, the exponential level of the path trace is increased. How to conduct path exploration to reduce the state space and the equivalence merging of paths are important research directions. In this paper, three kinds of equivalence marking are proposed, but three kinds of equivalence transfers are also based on the three equivalence rules, so as to reduce the equivalence path of exploration. After the corresponding case analysis and experimental results, it is shown that the acceleration ratio in memory consumption and time complexity reaches about 60%.

Depth-first search algorithms in this paper for place reachability in Petri nets are accurate and, expressed otherwise, depth-first search algorithms construct an unfolding signifying all reachable markings. Both depth-first and breadth-first search algorithms produce a complete prefix. In the full prefix of the size of the construction process, in the secure bounded net, the existence of linear algorithm complexity and linear memory consumption in the model checking of Petri nets is likely.

Similar depth-first algorithms and breadth-first search algorithms can select subsequent events, not only in the potential extensions enabled by the last added event, but also in the possible implementation extensions concurrent with this event.

This study proposes three types of reduction: (1) Identical marking M, after the same transition, produces the same marking M′, which is considered the cut node or, more specifically, the cut-off node. (2) The corresponding transition of marking M generates a marking M′ with the same structure, functioning as a cut-off node. (3) Identical marking M, following corresponding modifications, produces an equivalent marking M′, considered as a cut-off node. Utilizing a case study, we managed to diminish the net size by approximately 50%.

In future work, we will propose a theoretical system and whether the reduction algorithm can be applied to the equivalence reduction of color network expansion and the reduction of data-centric values of Petri net unfolding, reducing the traversal path number.

Author Contributions: Conceptualization, X.Y. and C.Y.; methodology, X.Y.; software, X.Y.; validation, X.Y., C.Y. and Y.C.; formal analysis, X.Y.; investigation, C.Y.; resources, C.Y.; data curation, Y.C.; writing—original draft preparation, Y.C.; writing—review and editing, Y.C.; visualization, Y.C.; supervision, C.Y.; project administration, C.Y.; funding acquisition, Y.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (62172166), the Shanghai Maglev and Rail Transit Collaborative Innovation Center, and the Science Research Project of Shanghai Science and Technology Committee (Grant No. 18DZ1205803).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References


33. Bernardinello, L.; Lomazova, I.; Nesterov, R.; Pomello, L. *Property-Preserving Transformations of Elementary Net Systems Based on Morphisms*; Koutny, M., Kordon, F., Moldt, D., Eds.; Transactions on Petri Nets and Other Models of Concurrency XVI; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2022; Volume 13220. [CrossRef]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.