Model checking Petri nets with MSVL

Ya Shi\textsuperscript{a}, Cong Tian\textsuperscript{a,\textdagger}, Zhenhua Duan\textsuperscript{a,\textdagger}, Mengchu Zhou\textsuperscript{b}

\textsuperscript{a}ICTT and ISN Laboratory, Xidian University, Xi’an 710071, China
\textsuperscript{b}Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07032, USA

1. Introduction

Petri nets have been widely used in the modeling, analysis and verification of concurrent systems \cite{1,7,16,29}. Many interesting properties of Petri nets, such as deadlock, liveness and reachability, can be analyzed by means of minimal siphons \cite{19,21,22}, abstraction \cite{5}, unfolding \cite{6} and observable liveness \cite{9}. In addition, various model checking approaches have been proposed for Petri nets \cite{2,3,13,14,17,18,34} to verify the properties expressed in Linear Temporal Logic (LTL) \cite{26} or Computation Tree Logic (CTL) \cite{8}.

Although LTL and CTL are able to describe many temporal properties, their expressiveness is not powerful enough, and actually weaker than full regular language \cite{28}. For instance, neither of them can describe the property shown in \textbf{Fig. 1}, where if a marking sequence is finite then it can be divided into multiple sub-sequences such that each one contains five markings and the atomic proposition \(q\) holds eventually over each sub-sequence. Intuitively, this is a periodic property. Since loop structures are very common in Petri nets, it is desired and sometimes necessary to verify these properties. Furthermore, the semantics of Petri nets falls into three categories: interleaving, concurrency and max-concurrency. However, most of the existing model checking methods focus only on the first one. Therefore, we are motivated to propose a method to verify full regular properties \cite{28} of Petri nets under each semantics.

Propositional Projection Temporal Logic (PPTL) is useful for verifying the properties of concurrent systems \cite{10}. It provides full regular expression capabilities \cite{28}, and is well suited for describing periodic and interval-related properties. For example, formula \(\diamond \epsilon \rightarrow (1 \land 4 \land \diamond q)^*\) elegantly expresses the property mentioned in \textbf{Fig. 1}. Here, \(\diamond \epsilon\) means that a marking sequence is finite, and \((1 \land 4 \land \diamond q)^*\) describes that the sequence is divided into multiple sub-sequences, each of which
satisfies $1 \pi 4 \land \Box q$. Further, $1 \pi 4$ states that each sub-sequence consists of 5 markings with its length being 4; and $\Diamond q$ indicates that the atomic proposition $q$ holds eventually over each sub-sequence.

Modeling, Simulation and Verification Language (MSVL) [10,11] is an executable subset of Projection Temporal Logic (PTL) [33]. It provides not only common statements of C and Java, such as assignment, sequential, conditional and loop statements, but also concurrent statements like projection and parallel statements. As a formal programming language, MSVL can be executed with an interpreter. Furthermore, it is able to model concurrent systems and verify properties of concurrent systems specified by PPTL formulas through the unified model checking methods [11]. A supporting tool, named MSV, has been developed for the purpose of modeling, simulation and verification of MSVL programs.

In this paper, we present three translations from Petri nets to MSVL programs, each of which is directed by a semantics of Petri nets. Although some translations from Petri nets to other programming languages have been proposed [15,17,20,23,25,27], the equivalence relations between Petri nets and the generated programs are hard to proved. The operational semantics of MSVL has been presented in [33] which serves as the basis for reasoning about properties of MSVL programs. Consequently, for each translation proposed in this paper, an equivalence relation between Petri nets and the generated MSVL programs is established. As a result, the tool MSV can be utilized to verify the properties of Petri nets expressed by PPTL formulas under each semantics.

The rest of this paper is organized as follows. In Section 2, the preliminaries about Petri nets, MSVL and PPTL are introduced. Three translations from Petri nets to MSVL programs are presented in Sections 3–5, respectively. Section 6 demonstrates a translating tool PN4MSVL and presents case studies. The conclusions are drawn in Section 7.

2. Preliminaries

This section briefly presents Petri nets, MSVL and PPTL.

2.1. Net systems

A net is a 3-tuple $N = (P, T, W)$, where $P$ and $T$ are disjoint sets of places and transitions, respectively, and $W : (P \times T) \cup (T \times P) \rightarrow \mathcal{N}_0$ is a mapping indicating the multiplicity of directed edges between places and transitions. Here, $\mathcal{N}_0$ stands for the set of non-negative integers. For a node $z \in P \cup T$, $^*z = \{y \in P \cup T \mid W(y, z) \geq 1\}$ and $^z = \{y \in P \cup T \mid W(z, y) \geq 1\}$ are the sets of input and output nodes, respectively, and $^*z = ^z \cup ^*z$.

A marking of a net $N = (P, T, W)$ is a multiset of places $P \rightarrow \mathcal{N}_0$. A transition $t$ is enabled at $M$ if $\forall p \in ^*t, M(p) \geq W(p, t)$. If $t$ is enabled at $M$, then it can occur and the occurrence will lead to the marking $M'$, where for each $p \in P$, $M'(p) = M(p) - W(p, t) + W(t, p)$. A marking is dead if it enables no transitions. A step is a non-empty multiset of transitions $U : T \rightarrow \mathcal{N}_0$. A step $U$ is enabled at $M$, denoted by $M[U]$, if for each $p \in P$, $M(p) \geq \sum_{t \in U} W(t, p)$. If $U$ is enabled at $M$, then it can execute and the execution will lead to the marking $M'$, denoted by $M[U]M'$, where for each $p \in P$, $M'(p) = M(p) + \sum_{t \in U} W(t, p) - W(p, t)$. $U$ is a min-step at $M$ if $M[U]$ and $|U| = 1$; it is a max-step at $M$ if $M[U]$ and there is no step $U'$ such that $U \subset U'$ and $M[U']$.

A net system is a 4-tuple $Z = (P, T, W, M_0)$ where $(P, T, W)$ is a net and $M_0$ is the initial marking. A sequence of steps $\lambda = U_1 U_2 \ldots U_n$ is a step sequence if there exist markings $M_1, M_2, \ldots, M_n$ with $M_0 \preceq U_1 \preceq M_1 \preceq U_2 \preceq \ldots \preceq U_n \preceq M_n$. $M_0(\lambda)M$ means that $M$ can be reached from $M_0$ through the execution of a finite step sequence $\lambda$. $M$ is reachable if there is a finite step sequence $\lambda$ such that $M_0(\lambda)M$. A net system is $k$-bounded if every reachable marking puts at most $k$ tokens into each place. A net system is called $1$-of-if it is 1-bounded.

A finite word of $Z$ is a finite marking sequence $\alpha = M_0 M_1 \ldots M_n$ such that there is a finite step sequence $\lambda = U_1 U_2 \ldots U_n$ with $M_0 \preceq U_1 \preceq M_1 \preceq U_2 \preceq \ldots \preceq U_n \preceq M_n$, and $M_n$ is dead. An infinite word of $Z$ is an infinite marking sequence $\alpha = M_0 M_1 \ldots$ such that there is an infinite step sequence $\lambda = U_1 U_2 \ldots$ with $M_0 \preceq U_1 \preceq U_2 \ldots$. In both cases, we call that $\alpha$ is induced by $\lambda$. A word is a min word (resp. max word) if it can be induced by a step sequence consisting of only min-steps (resp. max-steps). The concurrency semantics of $Z$ is the set of all words, while the interleaving semantics (resp. max-concurrency semantics) is composed of all min words (resp. max words).

The concurrent reachability graph of $Z$ is $\text{CRG}_Z = (RM_Z, \{(M, U, M') \mid M \in RM_Z \land M[U]M'\})$ where $RM_Z$ is the set of all reachable markings indicated by the nodes in the graph, $M_0$ is the initial node, and the arcs between nodes are labeled by steps.
2.2. Modeling, Simulation and Verification Language (MSVL)

As a modeling, simulation and verification language, MSVL provides an executable subset of PTL with more intuitive and practical elements [10,11]. First of all, we present intervals over which MSVL programs are interpreted.

2.2.1. Interval

Let \( \mathcal{P} \) be a set of atomic propositions, \( \mathcal{B} = \{ \text{true}, \text{false} \} \) the Boolean domain, \( \mathcal{V} \) a countable set of typed variables, and \( \mathcal{D} \) all data types we need such as integers and lists. We assume that the variables are partitioned into static and dynamic ones.

A state \( s \) over \( \mathcal{V} \) and \( \mathcal{P} \) is a pair of assignments \( (l_{\text{var}}, l_{\text{prop}}) \) where \( \forall v \in \mathcal{V}, s[v] = l_{\text{var}}[v] \in \mathcal{D} \cup \{ \text{nill} \} \), and \( \forall \pi \in \mathcal{P}, s[\pi] = l_{\text{prop}}[\pi] \in \mathcal{B} \). Here, \text{nill} means undefined. An interval \( \sigma = (s_0, s_1, \ldots) \) is a state sequence. The length of \( \sigma \) is the number of states minus 1. To give a unified notation for both finite and infinite intervals, the extended integers in \( \mathbb{N}_o = \mathbb{N}_0 \cup \{ \omega \} \) are used as indices. Further, the comparison operators \( =, < \) and \( \leq \) are extended to \( \mathbb{N}_o \) by considering \( \omega = \omega \) and \( \forall i \in \mathbb{N}_o, i < \omega \). For convenience, we use \( \langle s_0, \ldots, s_0 \rangle \) to denote a finite or infinite interval \( \sigma \), where \( s_0[\pi] \) is undefined if \( |\sigma| = \omega \). Further, the \( \text{i} \)th prefix of \( \sigma, \sigma_i, \) is utilized to denote the sub-sequence \( (s_0, \ldots, s_i) \) for all \( 0 \leq i \leq |\sigma| \), and \( \sigma_{-1} = \varepsilon \). The concatenation of \( \sigma = (s_0, \ldots, s_i) \) and \( \sigma' = (s_{i+1}, s_{i+2}, \ldots) \) is defined as \( \sigma \cdot \sigma' = (s_0, \ldots, s_i, s_{i+1}, s_{i+2}, \ldots) \).

2.2.2. Syntax

MSVL contains two kinds of expressions: arithmetic expressions \( e \) and Boolean expressions \( b \), which are inductively defined as follows:

\[
e ::= n \mid x \mid \bigcirc x \mid \exists x \mid e_1 \text{ op } e_2 \quad \text{where} \quad \text{op} ::= + \mid - \mid * \mid / \mid \text{mod}
\]

\[
b ::= \text{true} \mid \text{false} \mid e_0 = e_1 \mid \neg b \mid b_0 \land b_1
\]

where \( n \) is an integer, \( x \) a variable, \( \bigcirc x \) (resp. \( \exists x \)) the value of \( x \) at the next (resp. the previous) state.

The elementary statements of MSVL are shown in Fig. 2, where \( b_i \) is a Boolean expression, \( p_i \) a statement, \( 1 \leq i \leq m \), \( x \) a variable, and \( e \) an arithmetic expression. Here, \( \varepsilon \) means that the current state is the final one of the interval over which a program is executed. \( \text{skip} \) specifies the interval of a unit length. The \( \text{sequential} \), \( \text{if} \) and \( \text{while} \) statements are the same as those in the conventional imperative languages. \( p_1 || p_2 \) indicates that \( p_1 \) and \( p_2 \) start simultaneously, execute in parallel, and can end asynchronously. \( p_1 \) or \( p_2 \) means that either \( p_1 \) or \( p_2 \) is executed. \( \lceil p \rceil_{\text{while}}(b_1 \rightarrow b_i) \) denotes that if no condition is \( \text{true} \), then the program terminates; otherwise an arbitrary program \( p_i \) with the condition \( b_i \) being \( \text{true} \) is selected for execution. \( x := e \) implies that, at the current state, \( x \) is assigned by the current value of \( e \) and a proposition \( p_k \) associated with \( x \) is assigned by \( \text{true} \). \( \lbf(x) \) means that \( x \) is framed at the current state. That is, the value of \( x \) remains unchanged if no assignment to \( x \) is encountered at the state. \( \text{fr}(x) \) implies that \( x \) is framed at all states over an interval except the first one. For convenience, we denote \( \bigwedge_{v \in \mathcal{V}} \lbf(v) \) and \( \bigwedge_{v \in \mathcal{V}} \text{fr}(v) \) by \( \lbf(V') \) and \( \text{fr}(V') \) with \( V' \subseteq \mathcal{V} \), respectively.

A state program is the conjunction of state components. Here, a state component \( ps(x) \) is defined by:

\[
A ::= e \mid x := e \\
ps(x) ::= A \mid \lbf(x) \mid A \land \lbf(x)
\]

2.2.3. Operational semantics

The operational semantics of MSVL is described in [33] by means of reduction rules which specify how the statements are executed and how the expressions are evaluated. The rules are defined over configurations which fall into two types: one for programs and the other for expressions. A configuration w.r.t. a program \( p \) is a 4-tuple \( (p, \sigma, s_i, 0) \) where \( \sigma \) is an interval, \( s_i \) the current state at which \( p \) is being executed, and \( i \) a variable counting the number of states in \( \sigma \). The initial configuration is \((p, \varepsilon, s_0, 0)\). When a program terminates, it is reduced to \( \text{true} \) and the state is changed to \( \emptyset \). Thus, the final configuration is \((\text{true}, \sigma, \emptyset, |\sigma| + 1)\). Similarly, the configuration w.r.t. an expression \( \exp \) is \((\exp, \sigma, s_i, 1)\).

To access values of the variables at a state, we need the following notations. Let \( x \) and \( y \) be variables, \( m \) an integer, \( s_i \) a state, and \( p_k \) denote \( p_k \) or \( \neg p_k \). \( s_i[m, p_k]/x(y) = (m, p_k) \) if \( y = x \), otherwise \( s_i[m, p_k]/x(y) = s_i(y) \). Further, let \( w \equiv \bigwedge_{k \in X}(x_k = m_k \land p_{s_k}). s_i[w](y) = (m_k, p_{s_k}) \) if \( y = x_k \in X \), otherwise \( s_i[w](y) = s_i(y) \). Function \( \Pi_j \) is used to obtain the \( j \)th
component of a tuple. For instance, \( \Pi_1(s_i(x)) = m \) where \( s_i(x) = (m, p_i) \). For convenience, \( \bigwedge \{p_1, \ldots, p_m\} \) and \( \bigvee \{p_1, \ldots, p_m\} \) are employed to denote \( p_1 \wedge \cdots \wedge p_m \) and \( p_1 \vee \cdots \vee p_m \), respectively, where \( p_i \) is a program for all \( 1 \leq i \leq m \).

\( \rightarrow \) means a binary relation over the set of configurations within a single state, while \( \rightarrow \) is a transition relation between two configurations at different states. Evaluation rules of expressions are given in Table A.1, where \( \downarrow \) denotes the evaluation relation concerning how the results of expressions are obtained. Semantic equivalence rules are shown in Table A.2, where \( w \) is a state program or a Boolean expression. Table A.3 provides transition rules within a state to handle concurrent assignments within a state. These rules are used to perform assignments and capture the minimal models of programs. Here, minimal models are the canonical models describing exactly the execution paths of the program [10]. Interval transition rules in Table A.4 make the reduction of the program move from the current state to the next one or stop. Rule Non-Or in Table A.4 is necessary to reduce non-deterministic programs. For simplicity, we define the following auxiliary symbols, where Conf denotes the set of all configurations.

\[
\begin{align*}
0 & \mapsto \{(c, c) \mid c \in \text{Conf}\} \\
0 & \mapsto \{(a, b) \mid a, b \in \text{Conf} \text{ and } \exists c \in \text{Conf}, a \rightarrow c \text{ and } c \rightarrow b, k \geq 0\} \\
0 & \mapsto \{c, c' \in \text{Conf} \text{ and } c \rightarrow c'\} \\
0 & \mapsto \{(a, b) \mid a, b \in \text{Conf} \text{ and } \exists c \in \text{Conf}, a \rightarrow c \text{ and } c \rightarrow b, k \geq 0\} \mapsto \{c, c' \in \text{Conf} \text{ and } c \rightarrow c'\}
\end{align*}
\]

**Lemma 1.** For a finite interval \( \sigma \) and a program \( \phi \), \( \sigma \) is a finite minimal model of \( \phi \) iff \( (\phi, \epsilon, s_0, 0) \mapsto^{[\sigma] + 1} (\text{true}, \emptyset, n, [\sigma] + 1) \).

This lemma can be easily obtained from Theorems 12 and 17 in [33]. Intuitively, it states that each finite minimal model of a program can be obtained by reducing the program through finite steps.

### 2.3. Propositional Projection Temporal Logic (PPTL)

PPTL is the propositional subset of PTL. Its expressive power is full regular [28]. PPTL formulas are constructed by the following grammar:

\[
p = \pi \mid \neg p \mid p_1 \land p_2 \mid p_1 \lor p_2 \mid (p_1, \ldots, p_m) \text{ proj } p_0 \mid p^+
\]

where \( \pi \in \mathcal{P} \), and \( p_1 \) and \( p_i \) are PPTL formulas, \( 0 \leq i \leq m \). A state \( s \) over \( \mathcal{P} \) is a mapping from \( \mathcal{P} \) to \( \mathcal{B} \). The following are some derived formulas of PPTL:

- **Empty**: \( \varepsilon \equiv \neg \top \)
- **Sometimes**: \( \Diamond p \equiv (\text{true}, p) \text{ proj } \varepsilon \)
- **Star**: \( p^* \equiv \varepsilon \lor p^+ \)
- **Length**: \( \text{len } n \equiv \Diamond^n \varepsilon \), \( n \geq 0 \)

Obviously, \( \varepsilon \) is not only a PPTL formula but also an MSVL statement. In both cases, it has the same meaning, i.e., the current state is the final one. \( \top \) means that from now on \( p \) holds eventually over the interval. \( p^* \) denotes \( \varepsilon \) or \( p \) holds repeatedly for a finite number of times. \( \text{len } n \) means the length of an interval is \( n \).

A net system satisfies a PPTL formula under some semantics iff each word of the net system under this semantics satisfies the formula.

### 3. Interleaving semantics based translation

We first present the translation from net systems to MSVL programs which is directed by the interleaving semantics of net systems. Let \( Z = (P, T, W, M_0) \) be a net system. For each transition \( t \in T \), let \( g_t \equiv \bigwedge_{q \in \tau} v_q \geq w_q \) and \( \phi_t \equiv \bigwedge_{q \in \tau} v_q := v_q - w_q + w_q, \) where for each place \( q \in \tau \), \( w_q = W(q, t) \) and \( w_q = W(t, q) \) are natural numbers. Furthermore, we assume \( g_t \equiv \bigvee_{t \in T} g_t \). An MSVL program \( \phi_Z \) is generated as follows:

\[
\phi_Z \equiv \bigwedge_{q \in \Pi_1(s_0(x))} (v_q \leq m_q) \wedge \text{while}(\phi_t)\left[\bigcup_{t \in T} (g_t \rightarrow \phi_t)\right]
\]

where for each place \( q \in P \), \( m_q = M_0(q) \) is a natural number.

Within \( \phi_Z \), for each place \( q \in P \), there is a variable \( v_q \) counting the number of tokens in \( q \). Naturally, \( v_q \) is initialized with the initial marking \( m_q \) of \( q \). For each transition \( t \in T \), a Boolean expression \( g_t \) and a statement \( \phi_t \) are defined according to its enabling and firing rules, respectively. Additionally, a Boolean expression \( g_t \) is introduced. It holds iff some transitions are enabled at the current marking. Intuitively, \( \phi_Z \) simulates the interleaving semantics of \( Z \). That is, its execution repeats the process in which one transition enabled at the current marking is randomly selected for occurrence each time, until a dead marking is encountered.

Let \( \phi \) be an MSVL program containing a variable \( v_q \) for each place \( q \) of \( Z \). Formally, \( Z \) is interleaving equivalent (concurrently equivalent, or max-concurrently equivalent) to \( \phi \) iff for each min word (word, or max word) \( M_0 M_1 \ldots \) of \( Z \), there exists a minimal model \( \sigma = (s_0, s_1, \ldots) \) of \( \phi \) such that for all \( 0 \leq h \leq |\sigma| \) and all \( q \in P \), \( \Pi_1(s_h(v_q)) = M_h(q) \), and vice versa.
Next, we show that Z is interleaving equivalent to $\varphi_2$. For convenience, let $\varphi_Z \equiv (\land_{q \in P}(fr(v_q) \land v_q \leftarrow m_q)) \land \delta$ where $\delta = \text{while}(g_t) \left[ \bigwedge_{t \in T} (g_t \rightarrow \varphi_t) \right]$. Furthermore, $\varphi_t$ and $\delta$ are reduced to the semantically equivalent formulas $\bigcirc \land \{\gamma, \varepsilon\}$ and $\lor \left[ \bigwedge_{t \in T} \{g_t \land \delta\} \right]$, respectively:

$$\varphi_t \equiv \bigwedge_{q \in \cal T} v_q = v_{q \varepsilon} + W_{q t}$$

$$\equiv \bigwedge_{q \in \cal T} \bigcirc (v_q \leftarrow \bigcirc (v_q - W_{q t} + W_{q t}) \land \varepsilon)$$

(UAass)

$$\equiv \bigwedge_{q \in \cal T} \bigcirc (v_q \leftarrow \bigcirc v_q - W_{q t} + W_{q t} \land \varepsilon)$$

(Next1)

$$\equiv \bigcirc \bigwedge_{q \in \cal T} (v_q \leftarrow \bigcirc v_q - W_{q t} + W_{q t} \land \varepsilon)$$

$$\equiv \bigcirc \bigwedge_{q \in \cal T} \{\gamma, \varepsilon\}$$

where for each transition $t \in T$, $\gamma_t \equiv \bigwedge_{q \in \cal T} v_q \leftarrow \bigcirc v_q - W_{q t} + W_{q t}$.

$$\delta = \text{while}(g_t) \left[ \bigwedge_{t \in T} (g_t \rightarrow \varphi_t) \right]$$

$$\equiv \text{while}(g_t) \left\{ \lor \bigwedge_{t \in T} \{ \text{if}(g_t) \land \varphi_t \}, \lor \{ \text{false} \} \right\}$$

(Guard)

$$\equiv \text{while}(g_t) \left\{ \lor \bigwedge_{t \in T} \{ \text{if}(g_t) \land \varphi_t \} \right\} \lor \{ \text{false} \}$$

(If)

$$\equiv \text{while}(g_t) \left\{ \lor \bigwedge_{t \in T} \{ \text{if}(g_t) \land \varphi_t \} \right\}$$

(F1-2)

$$\equiv \text{if}(g_t) \land \left\{ \lor \bigwedge_{t \in T} \{ \text{if}(g_t) \land \varphi_t \}, \lor \{ \text{false} \} \right\} \lor \{ \text{false} \}$$

(Wil)

$$\equiv \lor \left\{ \land \{ \text{if}(g_t) \land \varphi_t \}, \lor \{ \text{false} \} \right\} \lor \{ \text{false} \}$$

(If)

$$\equiv \lor \left\{ \land \{ \text{if}(g_t) \land \varphi_t \}, \lor \{ \text{false} \} \right\} \lor \{ \text{false} \}$$

(Next2)

$$\equiv \text{Chop1}$$

$$\equiv \text{Chop2}$$

$$\equiv \text{Chop1}$$
\[\equiv \bigvee \left\{ \bigwedge_{t \in T} (\mathbf{gr} \cdot \bigvee \{ \mathbf{gr} \cdot \bigwedge \{ \gamma_t, \delta \} \bigwedge \{ \neg \mathbf{gr}, \varepsilon \} \bigwedge \{ \gamma_t, \delta \} \bigwedge \{ \neg \mathbf{gr}, \varepsilon \} \right\} \]  

(Chop3)

Theorems 2 and 3 demonstrate the relationship between finite min words of \(Z\) and finite minimal models of \(\psi_z\).

**Theorem 2.** For each finite min word \(M_0M_1 \ldots M_n\) of \(Z\), there is a finite minimal model \(\langle s_0[w_0], \ldots, s_n[w_n] \rangle\) of \(\psi_z\) such that for all \(0 \leq h \leq n\) and all \(q \in P, \Pi_1(s_h[w_h])(v_q) = M_h(q)\).

**Proof.** We assume that there is a finite min word \(M_0M_1 \ldots M_n\) of \(Z\) where \(n \geq 0\). Let \(t_h \in T\) be the transition with \(M_{h-1}[t_h])M_h\) for all \(0 < h \leq n\), and \(\sigma = \langle s_0[w_0], \ldots, s_n[w_n] \rangle\) where

- for all \(0 < h \leq n\) and all \(q \in P, \Pi_1(s_h[w_h])(v_q) = M_h(q)\);
- \(\forall q \in P, \Pi_1(s_0[w_0])(v_q) = p^q_0\); for all \(0 < h \leq n\), \(\forall q \in \tau_{t_h}, \Pi_2(s_h[w_h])(v_q) = p^q_{v_q}\), and \(\forall q \in P \\neg \tau_{t_h}, \Pi_2(s_h[w_h])(v_q) = \neg p^q_{v_q}\).

We need to prove that \(\sigma\) is a minimal model of \(\psi_z\). Let \(V_P = \{v_q \mid q \in P\}\) for each subset \(P \subseteq P\).

1. We first prove that for all \(0 \leq m \leq n\), \((\varphi_z, \varepsilon, s_0, 0) \models (\bigwedge_{q \in P} \varphi(V_P), \delta, s_m[w_m], m, \sigma_m, s_m[w_m], m)\). The proof proceeds by induction on \(m\).

**Base:** \(m = 0\). We have

1. \((\varphi_z, \varepsilon, s_0, 0) \models (\bigwedge_{q \in P} \varphi(V_P), \delta, s_0[w_0], 0 = n - 1)\)  
   (Hypothesis)
2. \(\varphi(V_P), \bigwedge_{t \in T} (\mathbf{gr} \cdot \bigwedge \{ \gamma_t, \delta \} \bigwedge \{ \neg \mathbf{gr}, \varepsilon \} \bigwedge \{ \gamma_t, \delta \} \bigwedge \{ \neg \mathbf{gr}, \varepsilon \} = (B1, 4-5)
3. 
4. \(\varphi(V_P), \bigwedge_{t \in T} (\mathbf{gr} \cdot \bigwedge \{ \gamma_t, \delta \} \bigwedge \{ \neg \mathbf{gr}, \varepsilon \} \bigwedge \{ \gamma_t, \delta \} \bigwedge \{ \neg \mathbf{gr}, \varepsilon \} = (T1, F1-2)
5. 
6. \(\varphi(V_P), \bigwedge_{t \in T} (\mathbf{gr} \cdot \bigwedge \{ \gamma_t, \delta \} \bigwedge \{ \neg \mathbf{gr}, \varepsilon \} \bigwedge \{ \gamma_t, \delta \} \bigwedge \{ \neg \mathbf{gr}, \varepsilon \} = (Non-Or)
7. 
8. \(\varphi(V_P), \bigwedge_{t \in T} (\mathbf{gr} \cdot \bigwedge \{ \gamma_t, \delta \} \bigwedge \{ \neg \mathbf{gr}, \varepsilon \} \bigwedge \{ \gamma_t, \delta \} \bigwedge \{ \neg \mathbf{gr}, \varepsilon \} = (T1)
9. 
10. \(\varphi(V_P), \bigwedge_{t \in T} (\mathbf{gr} \cdot \bigwedge \{ \gamma_t, \delta \} \bigwedge \{ \neg \mathbf{gr}, \varepsilon \} \bigwedge \{ \gamma_t, \delta \} \bigwedge \{ \neg \mathbf{gr}, \varepsilon \} = (Next2)
11. 
12. \(\varphi(V_P), \bigwedge_{t \in T} (\mathbf{gr} \cdot \bigwedge \{ \gamma_t, \delta \} \bigwedge \{ \neg \mathbf{gr}, \varepsilon \} \bigwedge \{ \gamma_t, \delta \} \bigwedge \{ \neg \mathbf{gr}, \varepsilon \} = (Fr2)
13. 
14. \(\varphi(V_P), \bigwedge_{t \in T} (\mathbf{gr} \cdot \bigwedge \{ \gamma_t, \delta \} \bigwedge \{ \neg \mathbf{gr}, \varepsilon \} \bigwedge \{ \gamma_t, \delta \} \bigwedge \{ \neg \mathbf{gr}, \varepsilon \} = (Next1)\)
(11) \[ \rightarrow \left( \bigwedge \left\{ \text{bf}(V_p), \text{fr}(V_p), \gamma_t, \delta \right\}, \sigma_{n-1}, s_n, n \right) \] (Tr1)

(12) \[ \rightarrow \left( \bigwedge \left\{ \text{bf}(V_p), \text{fr}(V_p), \bigwedge_{q \in \text{tr}^*} \nu_q \leq d_q, \delta \right\}, \sigma_{n-1}, s_n, n \right) \] (A4)

where \( \forall q \in \text{tr}^*; d_q = \Pi_1(s_{n-1}[w_{n-1}](v_q)) - w_{q_1} + w_{t_1q} = M_{n-1}(q) - w_{q_1} + w_{t_1q} = M_n(q) \) is a natural number

(13) \[ \rightarrow \left( \bigwedge \left\{ \text{fr}(V_p), \text{bf}(V_{D1}), \bigwedge_{q \in \text{tr}^*} \nu_q \leq d_q, \delta \right\}, \sigma_{n-1}, s_n, n \right) \] (Ass2)

(14) \[ \rightarrow^* \left( \bigwedge \left\{ \text{fr}(V_p), \sigma_{n-1}, s_n[w_n], n \right\} \right) \] (Min1, 3)

(2) Next, we show \((\bigwedge \{ \text{fr}(V_p), \delta \}, \sigma_{n-1}, s_n[w_n], n) \rightarrow (true, \phi, n + 1)\).

(1) \((\bigwedge \{ \text{fr}(V_p), \delta \}, \sigma_{n-1}, s_n[w_n], n) \)

(2) \((\bigwedge \{ \text{fr}(V_p), \bigwedge \{ g_t, \bigwedge_{q \in \text{tr}^*} \nu_q \leq d_q, \delta \right\}, \bigwedge \{ \neg g_t, \epsilon \right\}, \sigma_{n-1}, s_n[w_n], n) \)

(3) \[ \rightarrow^* \left( \bigwedge \left\{ \text{fr}(V_p), \bigwedge \left\{ \text{false, } \bigwedge_{t \in T} \bigwedge \left\{ g_t, \bigwedge_{q \in \text{tr}^*} \nu_q \leq d_q, \delta \right\}, \bigwedge \left\{ \text{true, } \epsilon \right\}, \sigma_{n-1}, s_n[w_n], n \right\} \right) \] (B2, 4-5)

since \( M_n \) is a dead marking, i.e., \( \forall t \in T, \exists q \in \text{tr}, \Pi_1(s_n[w_n](v_q)) = M_n(q) < w_{q_1} \)

(4) \[ \rightarrow^* \left( \bigwedge \left\{ \text{fr}(V_p), \epsilon, \sigma_{n-1}, s_n[w_n], n \right\} \right) \] (T1, F1-2)

(5) \[ \rightarrow (\epsilon, \sigma_{n-1}, s_n[w_n], n) \] (Fr1)

(6) \[ \rightarrow (true, \sigma_n, \phi, n + 1) \] (Tr2)

By Lemma 1, \( \sigma = \sigma_n \) is a minimal model of \( \varphi_Z \). \( \square \)

Theorem 3. For each finite minimal model \( \langle s_0[w_0], \ldots, s_n[w_n] \rangle \) of \( \varphi_Z \), there is a finite min word \( M_0, \ldots, M_n \) of \( Z \) such that for all \( 0 \leq h \leq n \) and all \( q \in P, M_h(q) = \Pi_1(s_n[w_n](v_q)) \).

Proof. We assume that there is a finite minimal model \( \sigma = \langle s_0[w_0], \ldots, s_n[w_n] \rangle \) of \( \varphi_Z \). By Lemma 1, \( (\varphi_Z, \epsilon, s_0, 0) \rightarrow (true, \phi, \psi, n + 1) \). Let \( \alpha = M_0, \ldots, M_n \) be a finite minimal sequence of \( Z \), where for all \( 0 \leq h \leq n \) and all \( q \in P, M_h(q) = \Pi_1(s_n[w_n](v_q)) \). We prove that \( \alpha \) is a min word of \( Z \) by considering two cases: \( n = 0 \) and \( n > 0 \).

(1) \( n = 0 \). In this case, \( \sigma = \langle s_0[w_0] \rangle \). We prove that \( M_0 \) is a dead marking. Suppose that \( M_0 \) is not dead. That is, there are some transitions enabled at \( M_0 \). Let \( T_1 \) be the set of transitions enabled at \( M_0 \) and \( t_1 \in T_1 \). We have

(2) \( (\varphi_Z, \epsilon, s_0, 0) \rightarrow (\bigwedge \{ \text{fr}(V_p), \bigwedge_{t \in T} \bigwedge \left\{ g_t, \bigwedge \left\{ \text{true, } \epsilon \right\}, \bigwedge \left\{ \text{false, } \epsilon \right\}, s_0[w_0], 0 \right\} \}) \) (B4, 6)

since \( \forall t \in T_1, \forall q \in \text{tr}^*, \Pi_1(s_0[w_0](v_q)) = M_0(q) \geq w_{q_1} \); and \( \forall t \in \text{tr}^*, \exists q \in \text{tr}, \Pi_1(s_0[w_0](v_q)) = M_0(q) < w_{q_1} \)

(3) \[ \rightarrow^* \left( \bigwedge \left\{ \text{fr}(V_p), \bigwedge \left\{ \text{false, } \epsilon \right\}, s_0[w_0], 0 \right\} \right) \] (T1, F1-2)
where Rule Non-Or is involved only at the last step. Thus, for any other transition in \( T_1 \), we can obtain a configuration similar to \( c_0 \). Furthermore, according to the proof of Theorem 2, each finite minimal model generated by the reduction from \( c_0 \) contains more than one state. Hence, each minimal model of \( \varphi_Z \) contains more than one state. This contradicts the assumption that \( \sigma = (s_0[w_0]) \) is a minimal model of \( \varphi_Z \). Hence, \( M_0 \) is dead, and there must exist a finite minimal word \( M_0 \) of \( Z \).

(2) \( n > 0 \). We prove that for all \( 0 < m \leq n \), there exists a transition \( t_m \) such that \( M_{m-1}[\{t_m\}]M_m \). The proof proceeds by induction on \( m \).

**Base**: \( m = 1 \). According to (1) in the proof of Theorem 2, we obtain \((\varphi_Z, \epsilon, s_0, 0) \rightarrow (\bigwedge \{\varphi(Vp), \delta\}, \epsilon, s_0[w_0], 0) = c_0' \).

Suppose that \( M_1 \) is not reachable from \( M_0 \) through the occurrence of a min-step. We consider the following two cases:

(a) \( M_0 \) is a dead marking. By (2) in the proof of Theorem 2, \( c_0' \rightarrow (\text{true}, \sigma_0, \emptyset, 1) \). Since Rule Non-Or is not involved in the reduction, \((\text{true}, \sigma_0, \emptyset, 1) \) is the only final configuration which can be obtained by reducing \((\varphi_Z, \epsilon, s_0, 0) \).

By Lemma 1, \( \sigma_0 \) is the only minimal model of \( \varphi_Z \). This contradicts the assumption that \( \sigma \) is a minimal model of \( \varphi_Z \) and \( |\sigma| = n > |\sigma_0| = 0 \). Thus, \( M_0 \) is not dead.

(b) Let \( T_1 \) be the set of transitions enabled at \( M_0 \). Suppose that no transition \( t \in T_1 \) can satisfy \( M_0[[t]]M_1 \). According to (1) in the proof of Theorem 2, \( c_0' \rightarrow (\bigwedge \{\varphi(Vp), \delta\}, \sigma_0, s_1[w'_1], 1) \) where \( \forall q \in P, \Pi_1(s_1[w'_1](v_q)) = M'_i(q) \). Here, \( M'_i \) is the marking reachable from \( M_0 \) through the occurrence of a min-step \( \{t'_i\} \), namely \( M_0[[t'_i]]M'_i \), where \( t'_i \in T_1 \). Since no transition \( t \in T_1 \) satisfies \( M_0[[t]]M_1 \), we have \( M_1 \neq M'_i \). We can further obtain \( w'_i \neq w_i \). Thus, no minimal model of \( \varphi_Z \) has the prefix \( \sigma_1 \). This contradicts the assumption that \( \sigma \) is a minimal model of \( \varphi_Z \). Hence, there must exist a transition \( t_1 \in T_1 \) such that \( M_0[[t_1]]M_1 \).

**Induction**: Suppose that the conclusion holds when \( m = n - 1 \). By (1) in the proof of Theorem 2, \((\varphi_Z, \epsilon, s_0, 0) \overset{\text{n-1}}{\rightarrow} (\bigwedge \{\varphi(Vp), \delta\}, \sigma_n, s_{n-1}[w_{n-1}], n - 1) = c_{n-1} \). When \( m = n \), suppose that \( M_n \) is not reachable from \( M_{n-1} \) through the occurrence of a min-step. The two cases are considered:

(c) \( M_{n-1} \) is a dead marking. According to (2) in the proof of Theorem 2, we have \( c_{n-1} \rightarrow (\text{true}, \sigma'_{n-1}, \emptyset, 1) \). Since \( |\sigma'_{n-1}| = n \) and \( |\sigma| = n + 1 \), each minimal model of \( \varphi_Z \) with \( \sigma_{n-2} \) being the prefix is shorter than \( \sigma \). Hence, \( M_{n-1} \) is not dead.

(d) Let \( T_n \) be the set of transitions enabled at \( M_{n-1} \). Suppose no transition \( t \in T_n \) can satisfy \( M_{n-1}[\{t\}]M_n \). We can obtain a contradiction similar to that in (b). Hence, there must exist a transition \( t_n \in T_n \) such that \( M_{n-1}[\{t_n\}]M_n \).

Next, we show that \( M_n \) is a dead marking. Suppose that there are transitions enabled at \( M_n \). Let \( t_{n+1} \) be a transition enabled at \( M_n \). For \( 0 < m \leq n + 1 \), there is a transition \( t_m \) such that \( M_{m-1}[\{t_m\}]M_m \). By (1) in the proof of Theorem 2, \((\varphi_Z, \epsilon, s_0, 0) \overset{\text{n+1}}{\rightarrow} (\bigwedge \{\varphi(Vp), \delta\}, \sigma_n, s_{n+1}[w_{n+1}], n + 1) = c_{n+1} \). According to the proof of Theorem 2, each finite minimal model generated by \( M_0 \) contains more than \( n + 1 \) states. Thus, each finite minimal model of \( \varphi_Z \) with \( \sigma_n \) being the prefix has more than \( n + 1 \) states. This contradicts the assumption that \( \sigma \) is a minimal model of \( \varphi_Z \). Hence, \( M_n \) is dead and \( \alpha = M_0 \ldots M_n \) is a finite minimal word of \( Z \). \( \square \)

To prove the relationship between infinite min words of \( Z \) and infinite minimal models of \( \varphi_Z \), we need the fixed-point theorem and the fixed-point induction shown below [32]. A subset \( D \) of a complete partial order relation \((B, \sqsubseteq)\) is inclusive if for all \( \omega \)-chains \( d_0 \sqsubseteq d_1 \sqsubseteq \cdots \sqsubseteq d_n \sqsubseteq \cdots \) in \( B \), if \( d_i \in D \) for all \( i \in \omega \) then \( \bigcup_{i < \omega} d_i \in D \). \( j \geq 0 \).

**Theorem 4** (Kleene fixed-point theorem). Let \((B, \sqsubseteq)\) be a complete partial order relation with the bottom being \( \perp \). Every continuous function \( f \) over \((B, \sqsubseteq)\) has a fixed point \( \text{fix}(f) = \bigcup_{i \in \omega} f^i(\perp) \) [32].

**Theorem 5** (Scott fixed-point induction). Let \((B, \sqsubseteq)\) be a complete partial order relation with the bottom being \( \perp \), \( f \) a continuous function over \((B, \sqsubseteq)\), and \( D \) an inclusive subset of \( B \). If \( \perp \in D \) and \( \forall x \in B, x \in D \rightarrow \text{fix}(F)(x) \in D \), then \( \text{fix}(F) \in D \) [32].

**Theorems 6** and **7** indicate the relationship between infinite min words of \( Z \) and infinite minimal models of \( \varphi_Z \).

**Theorem 6.** For each infinite min word \( M_0M_1 \ldots \) of \( Z \), there is an infinite minimal model \( \langle s_0[w_0], s_1[w_1], \ldots \rangle \) of \( \varphi_Z \) such that for all \( h \geq 0 \) and all \( q \in P, \Pi_1(s_h[w_h](v_q)) = M_h(q) \).

**Proof.** We assume that there exists an infinite min word \( M_0M_1 \ldots \) of \( Z \). Let \( t_h \in T \) be the transition with \( M_{h-1}[\{t_h\}]M_h \) for all \( h \geq 1 \), and \( \sigma = (s_0[w_0], s_1[w_1], \ldots) \) an infinite interval where

- for all \( h \geq 0 \) and all \( q \in P, \Pi_1(s_h[w_h](v_q)) = M_h(q) \); and
- \( \forall q \in P, \Pi_2(s_h[w_h](v_q)) = p_0[q]; \forall q \in \tau_q', \Pi_2(s_h[w_h](v_q)) = p_{\theta_q} \) and \( \forall q \in P \setminus \tau_q', \Pi_2(s_h[w_h](v_q)) = -p_{\theta_q} \). for all \( h > 0 \).

We prove that \( \sigma \) is a minimal model of \( \varphi_Z \). Let \( B = \{\sigma_i \mid i \in \omega \} \). A binary relation \( \leq \) and a combination operator \( \sqcup \) are defined over \( B \) : \( \sigma_i \leq \sigma_j \) if \( \sigma_i \) is a prefix of \( \sigma_j \), and \( \sigma_i \sqcup \sigma_j = \sigma_j \) if \( \sigma_i \leq \sigma_j \). Obviously, \( \sigma_i \leq \sigma_j \) iff \( i \leq j \). It is observed that \((B, \leq)\) is a partial order relation:
Reflexivity: For each $\sigma_{i-1}$, it is trivial that $\sigma_{i-1} \leq \sigma_{i-1}$.
Transitivity: If $\sigma_{i-1} \leq \sigma_{j-1}$ and $\sigma_{j-1} \leq \sigma_{k-1}$, then $i \leq j \leq k$. Thus, $\sigma_{i-1} \leq \sigma_{k-1}$.
Anti-symmetry: If $\sigma_{i-1} \leq \sigma_{j-1}$ and $\sigma_{j-1} \leq \sigma_{i-1}$, then $i \leq j$ and $i \leq j$. Hence, $i = j$ leading to $\sigma_{i-1} = \sigma_{j-1}$.
Further, we obtain that for any $\omega$-chain $\sigma_{0} \leq \sigma_{1} \leq \cdots \leq \sigma_{n}$ in $(\mathcal{B}, \leq)$, there exists a least upper bound $\bigcup_{k \in \mathbb{N}_0} \sigma_{k-1} \leq \sigma$.
Hence, $(\mathcal{B}, \leq)$ is a complete partial order relation with the bottom being $\sigma_{-1} = \epsilon$.
We prove that for all $j \in \mathbb{N}_0$, $\sigma_{j-1}$ is a prefix of a minimal model $\sigma'$ of $\varphi_Z$.
Base: It is trivial that $\sigma_{-1} = \epsilon$ is a prefix of $\sigma'$.
Induction: Suppose that $\sigma_{i-1}$ is a prefix of $\sigma'$ where $i \in \mathbb{N}_0$. According to (1) in the proof of Theorem 2, we have $(\varphi_Z, \epsilon, 0, 0, 0, 0, 0, 0, 0, 0) \rightarrow (\bigwedge_{j \in \mathbb{N}_0} \sigma_{j-1}, \sigma_{j-1})$, where $\sigma_{j-1}$ is the minimal interpretation on the propositions selected by Rules Min 1-3 at $s_i$. Hence, $\sigma_{i-1}$ is a prefix of $\sigma'$.
Let $f : B \rightarrow B$ be a function such that $f(\sigma_{i-1}) = \sigma_{i}$ for all $i \in \mathbb{N}_0$. Since $\sigma_{i-1} \leq \sigma_{j-1}$ implies $f(\sigma_{i-1}) = \sigma_{i} \leq f(\sigma_{j-1}) = \sigma_{j}$, we obtain that $f$ is monotonic. For any $\omega$-chain $\sigma_{0} \leq \sigma_{1} \leq \cdots \leq \sigma_{n}$ in $(\mathcal{B}, \leq)$ with $\bigcup_{k \in \mathbb{N}_0} \sigma_{k-1} = \sigma_{n-1}$, we have $f(\bigcup_{k \in \mathbb{N}_0} \sigma_{k-1}) = f(\sigma_{n-1}) = \sigma_{n} = \bigcup_{k \in \mathbb{N}_0} \sigma_{k} = \bigcup_{k \in \mathbb{N}_0} f(\sigma_{k-1})$. Therefore, $f$ is continuous. Further, by Kleene fixed-point theorem, $fix(f) = \lim_{n \rightarrow \infty} f^n(\epsilon) = \sigma'$.
Let $D = \{\sigma_{i-1} | \sigma_{i-1} \in B$ and it is a finite prefix of $\sigma'\}$. We assume that for any $\omega$-chain $\sigma_{0} \leq \sigma_{1} \leq \cdots$ of $(\mathcal{B}, \leq)$, each element $\sigma_{k-1}$ belongs to $D$. The least upper bound $\bigcup_{k \in \mathbb{N}_0} \sigma_{k-1}$ is a finite prefix $\sigma_{m-1}$ or the infinite interval $\sigma$. In both cases, $\bigcup_{k \in \mathbb{N}_0} \sigma_{k-1} \subseteq D$. Thus, $D$ is an inclusive subset of $B$. Since $\epsilon \in D$ and for all $i \in D$, $\sigma_{i-1} \in D \Rightarrow f(\sigma_{i-1}) = \sigma_{i} \in D$. By Scott fixed-point induction, we have $fix(f) = \sigma' \in D$. Since there exists only one infinite interval $\sigma$ in $D$, $\sigma = \sigma'$ is an infinite minimal model of $\varphi_Z$. □

Theorem 7. For each infinite minimal model $\langle s_0[w_0], s_1[w_1], \ldots \rangle$ of $\varphi_Z$, there exists an infinite min word $M_0M_1 \ldots$ of $Z$ such that for all $h \geq 0$ and all $q \in P$, $M_h(q) = s_{h}(s_{h}(w_h)\uparrow q_{h})$.

Proof. We assume that there is an infinite minimal model $\sigma = \langle s_0[w_0], s_1[w_1], \ldots \rangle$ of $\varphi_Z$. Let $\alpha = M_0M_1 \ldots$ be an infinite marking sequence of $Z$ such that for all $h \geq 0$ and all $q \in P$, $M_h(q) = s_{h}(s_{h}(w_h)\uparrow q_{h})$. We prove that $\alpha$ is a min word of $Z$.
Here $\alpha_i$ denotes the sequence $M_0M_1 \ldots M_i$, $i \in \mathbb{N}_0$. Let $R = \{\alpha_i | i \in \mathbb{N}_0\}$. A binary relation $\leq$ and a combination operator $\sqcup$ are defined over $R$: $\alpha_i \leq \alpha_j$ iff $\alpha_i$ is a prefix of $\alpha_j$, and $\alpha_i \sqcup \alpha_j = \alpha_j$ iff $\alpha_i \leq \alpha_j$. Similar to the proof of Theorem 6, we can prove that $(R, \leq)$ is a complete partial order relation with the bottom being $\alpha_0 = M_0$.
We prove that for all $j \in N_0$, $\alpha_j$ is a prefix of a min word $\alpha'$. Base: Since $M_0 = s_0$ is the initial marking of $Z$, $\alpha_0 = M_0$ is a prefix of $\alpha'$.
Induction: Suppose that $\alpha_i$ is a prefix of $\alpha'$ where $i \in N_0$. By (2) of Theorem 3, there is a transition $t_{i+1}$ such that $M_0(t_{i+1})M_{i+1}$. Thus, $\alpha_{i+1} = M_{i+1}$ is a prefix of $\alpha'$.

By Theorems 2, 3, 6 and 7, the following corollary can be obtained.

Corollary 1. $Z$ is interleaving equivalent to $\varphi_Z$.

4. Concurrency semantics based translation

This section presents the translation directed by concurrency semantics of net systems. As an additional condition, $Z$ is restricted to a 1-safe net system. Let $g_1 \equiv \sum_{t \in \mathcal{T}} p_t > 0$, $g_2 \equiv \bigwedge_{q \in P} q \geq \sum_{t \in \mathcal{T}} v_t \cdot w_q$ and $t_p \equiv \bigwedge_{q \in P} q \geq v_q := v_q + \sum_{t \in \mathcal{T}} v_t \cdot (w_q - w_q)$. An MSVL program $\kappa_0^Z$ is generated below:

$$\kappa_0^Z \equiv \left\{ \begin{array}{ll} \bigwedge_{q \in P} v_q := M_q & \text{while}(g_T) \\ \left( \bigwedge_{t \in \mathcal{T}} (v_t := 1 \vee v_t := 0) \right) \land \text{if}(g_1 \land g_2) \text{then}(t_p) \text{else} \{\text{false} \} \end{array} \right\}$$

where $g_T$ is the same as that of $\varphi_Z$.

Similar to the interleaving semantics based translation, for each place $q \in P$, a variable $v_q$ is employed to record the number of tokens in $q$. Note that for each transition $t \in \mathcal{T}$, a variable $v_t$ is introduced to mark whether $t$ is selected for occurrence at the current marking. This is different from the interleaving semantics based translation. Since the original models are restricted to 1-safe net systems, each transition occurs at most once at each reachable marking. If $v_t$ is set to 1, then $t$ will fire at the current marking; otherwise $t$ will not. Therefore, transitions with $v_t$ being 1 constitute the step selected to be executed at the current marking. Boolean expression $g_1$ guarantees that the step is not empty, while $g_2$ ensures that
each place can offer enough tokens for the step's execution. Statement \( \tau_p \) describes the change of each place's marking after executing the selected step. Intuitively, \( \kappa^0_Z \) simulates the concurrency semantics of \( Z \). That is, its execution repeats the process in which a step enabled at the current marking is randomly selected to be executed, until a dead marking is encountered.

Next, we prove that \( Z \) is concurrently equivalent to \( \kappa^0_Z \). For convenience, let \( \kappa^0_Z \equiv (\land_{q \in P} \nu_q \leftarrow \nu_q) \land \theta \). where \( \theta \equiv \text{while}(\gamma_T \land \text{if}(g_1 \land g_2)\text{then}(\nu_p)\text{else}(\text{false})) \) and \( \gamma_T \equiv (\land_{t \in T}(v_t = 1 \lor v_t = 0)) \). Furthermore, \( \tau_p \) and \( \theta \) are reduced into the semantically equivalent formulas \( \bigcirc \land \{\gamma_p, \epsilon\} \) and \( \bigvee (\land_{q \in P} \gamma_T \land g_1 \land g_2 \land \tau_p \land \theta) \). \( \land \{\neg g_T, \epsilon\} \). respectively.

\[
\tau_p \equiv \bigcap_{q \in P} \nu_q := \nu_q + \sum_{t \in \tilde{q}} (W_{tq} - W_{tq})
\]

\[
\equiv \bigcap_{q \in P} \bigcirc \left( \nu_q \leftarrow \left( \nu_q + \sum_{t \in \tilde{q}} (W_{tq} - W_{tq}) \right) \land \epsilon \right)
\]

\[
\equiv \bigcirc \left( \nu_q \leftarrow \bigcirc \nu_q + \sum_{t \in \tilde{q}} (W_{tq} - W_{tq}) \land \epsilon \right)
\]

\[
\equiv \bigcirc \left( \nu_q \leftarrow \bigcirc \nu_q + \sum_{t \in \tilde{q}} (W_{tq} - W_{tq}) \land \epsilon \right)
\]

\[
\equiv \bigcirc \left( \nu_q \leftarrow \bigcirc \nu_q + \sum_{t \in \tilde{q}} (W_{tq} - W_{tq}) \land \epsilon \right)
\]

where \( \gamma_T \equiv \bigcap_{q \in P} \nu_q \leftarrow \bigcirc \nu_q + \sum_{t \in \tilde{q}} (W_{tq} - W_{tq}) \).

\( \theta \equiv \text{while}(\gamma_T \land \text{if}(g_1 \land g_2)\text{then}(\nu_p)\text{else}(\text{false})) \)

\[
= \text{while}(\gamma_T) \left( \bigcap \{\gamma_T, \bigvee \{gL_{g_1, g_2, \tau_p}, \bigwedge \{\neg g_1, g_2\}\}\} \right)
\]

\[
= \text{while}(\gamma_T) \left( \bigcap \{\gamma_T, \bigvee \{gL_{g_1, g_2, \tau_p}, \bigwedge \{\neg g_1, g_2\}\}\} \right)
\]

\[
= \text{if}(\gamma_T) \left( \bigvee \{gL_{g_1, g_2, \tau_p}, \bigwedge \{\neg g_1, g_2\}\}\right)
\]

\[
= \text{if}(\gamma_T) \left( \bigvee \{gL_{g_1, g_2, \tau_p}, \bigwedge \{\neg g_1, g_2\}\}\right)
\]

The following two theorems present the relationship between finite words of \( Z \) and finite minimal models of \( \kappa^0_Z \).

**Theorem 8.** For each finite word \( M_0M_1 \ldots M_n \) of \( Z \), there exists a finite minimal model \( \langle s_0[w_0], \ldots, s_n[w_n] \rangle \) of \( \kappa^0_Z \) such that for all \( 0 \leq h \leq n \) and all \( q \in P \), \( \Pi_1(s_h[w_h](v_q)) = M_h(q) \).

**Proof.** We assume that there is a finite word \( M_0M_1 \ldots M_n \) of \( Z \) where \( n \geq 0 \). Let \( U_h \subseteq T \) be the step where \( M_{h-1}[U_h] \subseteq M_h \). \( 0 < h < n \), and \( \sigma = \langle s_0[w_0], \ldots, s_n[w_n] \rangle \) where

- for all \( 0 \leq h \leq n \) and all \( q \in P \), \( \Pi_1(s_h[w_h](v_q)) = M_h(q) \) and \( \Pi_2(s_h[w_h](v_q)) = p_{w_h} \);
- for all \( 0 < h \leq n \) and all \( t \in T \), \( \Pi_1(s_{h-1}[w_{h-1}](v_t)) = U_h(t) \), \( \Pi_2(s_{h-1}[w_{h-1}](v_t)) = p_{w_h} \), and \( \forall t \in T \), \( \Pi_1(s_n[w_n](v_t)) = n1 \) and \( \Pi_2(s_n[w_n](v_t)) = p_{w_h} \.

We prove that \( \sigma \) is a minimal model of \( \kappa^0_Z \). For all \( 0 \leq h \leq n \), let \( W^p_{h} \) be the statement obtained from \( w_h \) with the assignments to variables in \( V_T \) removed. Here, \( V_p = \{v_q \mid q \in P\} \) and \( V_T = \{v_t \mid t \in T\} \).

(1) We first prove that for all \( 0 \leq m \leq n \), \( (\kappa^0_Z, \epsilon, s_0, 0) \rightarrow_m (\theta, \Pi_{m-1}, s_m[w^p_{m}]) \). The proof proceeds by induction on \( m \).

**Base:** \( m = 0 \). We have

(1) \( (\kappa^0_Z, \epsilon, s_0, 0) \rightarrow (\land_{q \in q} \nu_q \leftarrow \nu_q) \land \epsilon, s_0, 0) \)

(2) \( \rightarrow (\theta, \epsilon, s_0[w^p_{0}], 0) \) (Min1)

**Induction:** Suppose that the conclusion holds for \( m = n - 1 \). When \( m = n \), we have

(1) \( (\kappa^0_Z, \epsilon, s_0, 0) \rightarrow (\theta, \Pi_{n-2}, s_{n-1}[w^p_{n-1}], n-1) \) (Hypothesis)
(2) \( \rightarrow (\forall \{ g_T, g_2, \tau_P: \theta \}, \bigwedge_{i \in T} [\neg g_T, \varepsilon], \sigma_{n-2}, s_{n-1}[w_{n-1}^V], n-1) \)

(3) \( \rightarrow^* \left( \bigvee \left\{ \bigwedge \{ \text{true, } g_1, g_2, \tau_P: \theta \}, \bigwedge \{ \text{false, } \varepsilon \}, \sigma_{n-2}, s_{n-1}[w_{n-1}^V], n-1 \right\} \right) \)

since \( M_{n-1}(U_n)M_n \), i.e., \( \forall t \in U_n, \forall q \in T, \Pi_1(s_{n-1}[w_{n-1}^V](v_q)) = M_{n-1}(q) \geq w_q \)

(4) \( \rightarrow^* \left( \bigwedge \{ g_1, g_2, \tau_P: \theta \}, \sigma_{n-2}, s_{n-1}[w_{n-1}^V], n-1 \right) \)

(5) \( \rightarrow^* \left( \bigwedge \{ v_t = d_t, g_1, g_2, \tau_P: \theta \}, \sigma_{n-2}, s_{n-1}[w_{n-1}^V], n-1 \right) \)

(6) \( \rightarrow \left( \bigwedge \{ g_1, g_2, \tau_P: \theta \}, \sigma_{n-2}, s_{n-1}[w_{n-1}], n-1 \right) \)

(7) \( \rightarrow \left( \bigwedge \{ \text{true, true, } \tau_P: \theta \}, \sigma_{n-2}, s_{n-1}[w_{n-1}], n-1 \right) \)

since \( U_n \) is non-empty and enabled at \( M_{n-1} \), i.e., \( \sum_{t \in T} \Pi_1(s_{n-1}[w_{n-1}](v_t)) = \sum_{t \in T} U_n(t) > 0 \), and \( \forall q \in P, \Pi_1(s_{n-1}[w_{n-1}](v_q)) = M_{n-1}(q) \geq \sum_{t \in T} \Pi_1(s_{n-1}[w_{n-1}](v_t)) \cdot w_q = \sum_{t \in T} U_n(t) \cdot w_q \)

(8) \( \rightarrow (\tau_P: \theta, \sigma_{n-2}, s_{n-1}[w_{n-1}], n-1) \)

(9) \( \rightarrow (\bigcirc (\bigwedge \{ g_T, \varepsilon \}; \theta), \sigma_{n-2}, s_{n-1}[w_{n-1}], n-1) \)

(10) \( \rightarrow (\bigwedge \{ g_T, \varepsilon \}; \theta), \sigma_{n-1}, s_n, n \)

(11) \( \rightarrow (\bigwedge \{ v_q = d_q, \varepsilon \}; \theta), \sigma_{n-1}, s_n, n \)

(12) \( \rightarrow (\bigwedge \{ v_q = d_q, \varepsilon \}; \theta), \sigma_{n-1}, s_n, n \)

(13) \( \rightarrow (\bigwedge \{ v_q = d_q, \theta \}, \sigma_{n-1}, s_n, n \)

(14) \( \rightarrow^* (\theta, \sigma_{n-1}, s_n[w_n^V], n) \)

(2) Next, we show \( (\theta, \sigma_{n-1}, s_n[w_n^V], n) \overset{1}{\rightarrow} (\text{true, } \emptyset, n + 1). \)

(1) \( (\theta, \sigma_{n-1}, s_n[w_n^V], n) \)

(2) \( \rightarrow (\forall \{ g_T, g_2, \tau_P: \theta \}, \bigwedge_{i \in T} [\neg g_T, \varepsilon], \sigma_{n-1}, s_n[w_n^V], n) \)
Theorem 9. For each finite minimal model \(\langle s_0[w_0], \ldots, s_n[w_n]\rangle\) of \(\kappa_2^0\), there exists a word \(M_0, \ldots, M_n\) of \(Z\) such that for all \(0 \leq h \leq n\) and all \(q \in P\), \(M_h(q) = \Pi_1(s_h[w_h](v_q))\).

Proof. We assume that there is a finite minimal model \(\sigma = \langle s_0[w_0], \ldots, s_n[w_n]\rangle\) of \(\kappa_2^0\). By Lemma 1, \((\kappa_2^0, \epsilon, s_0, 0)^{n+1}\) is a word. Let \(\alpha = M_0, \ldots, M_n\) be the finite marking sequence of \(Z\), where for all \(0 \leq h \leq n\) and all \(q \in P\), \(M_h(q) = \Pi_1(s_h[w_h](v_q))\). We prove that \(\alpha\) is a word by considering two cases: \(n = 0\) and \(n > 0\).

(1) \(n = 0\). In the case, \(\sigma = \langle s_0[w_0]\rangle\). We prove that \(M_0\) is a dead marking. Suppose \(M_0\) is not dead, i.e., there are some transitions enabled at \(M_0\). Let \(U_0 \subseteq T\) be a set of transitions. According to Lines 1–6 of Induction in the proof of Theorem 8, \((\kappa_2^0, \epsilon, s_0, 0) \rightarrow (\langle g_1, g_2, \tau, \theta \rangle, \epsilon, s_0[w_0], 0) = c_0\) where \(\forall \tau \in T\), \(\Pi_1(s_0[w_0](v_\tau)) = U_0(t)\). We can obtain \(c_0 \rightarrow (\text{false, } \epsilon, s_0[w_0], 0)\) by B4, B6 and F1 if \(U_1\) is empty or not enabled at \(M_0\). Then, \(\sum_{\tau \in T} \Pi_1(s_0[w_0](v_\tau)) = \sum_{\tau \in T} U_1(t) = 0\). This contradicts the assumption that \(\sigma = \langle s_0[w_0]\rangle\) is a minimal model of \(\kappa_2^0\). Therefore, \(M_0\) is dead, and there must exist a finite word \(M_0\) of \(Z\).

(2) \(n > 0\). We prove that for all \(0 < m \leq n\), there is a step \(U_m\) with \(M_{m-1}[U_m]M_m\). The proof proceeds by induction on \(m\).

Base: \(m = 1\). By (1) in the proof of Theorem 8, \((\kappa_2^0, \epsilon, s_0, 0) \rightarrow (\theta, \epsilon, s_0[w_0], 0) = c_0\). Suppose that \(M_1\) is not reachable from \(M_0\) through the execution of a step. The following two cases are considered:

- (a) \(M_0\) is dead. By (2) in the proof of Theorem 8, \(c_0 \rightarrow (\text{true, } \sigma_0, 0, 1)\). Since Rule NON-Or is not involved in the reduction, \((\text{true, } \sigma_0, 0, 1)\) is the only final configuration which can be generated by reducing \(c_0\). According to Lemma 1, \(\sigma_0\) is the only minimal model of \(\kappa_2^0\). This contradicts the assumption that \(\sigma\) is a minimal model of \(\kappa_2^0\), and \(|\sigma| = n > |\sigma_0| = 0\). Thus, \(M_0\) is not a dead marking.

- (b) Let \(T_0\) be the set of transitions enabled at \(M_0\). Suppose that there is no step \(U \subseteq T_0\) with \(M_0[U]M_1\). According to (1) in the proof of Theorem 8, \(c_0 \rightarrow (\theta, \sigma_0, s_1[w_1^s], 0, 1)\) where \(\forall \tau \in T_1, \Pi_1(s_1[w_1^s](v_\tau)) = M_1(q)\). Here \(M_1\) is the marking reachable from \(M_0\) through the execution of a step \(U \subseteq T_1\), namely \(M_0[U_1]M_1\). Since there is no step \(U \subseteq T_1\) with \(M_0[U]M_1\), we obtain that \(M_0' \neq M_1\) and \(w_1^s \neq w_1^f\). Thus, no minimal model of \(\kappa_2^0\) has the prefix \(\sigma_1\). This contradicts the assumption that \(\sigma\) is a minimal model of \(\kappa_2^0\). Hence, there must exist a step \(U \subseteq T_1\) such that \(M_0[U_1]M_1\).

Induction: Suppose that the conclusion holds for \(m = n - 1\). By (1) in the proof of Theorem 8, \((\kappa_2^0, \epsilon, s_0, 0)^{n-1} \rightarrow (\theta, \sigma_{n-2}, s_{n-1}[w_{n-1}], 0, 1) = c_{n-1}\). When \(m = n\), suppose that \(M_n\) is not reachable from \(M_{n-1}\) through the execution of a step. We consider the following two cases:

- (c) \(M_{n-1}\) is a dead marking. According to (2) in the proof of Theorem 8, \(c_{n-1} \rightarrow (\text{true, } \sigma_{n-1}, 0, 1, 0)\). Since \(|\sigma_{n-1}| = n\) and \(|\sigma_{n-1}| = n + 1\), each minimal model of \(\kappa_2^0\) with \(\sigma_{n-2}\) being the prefix is shorter than \(\sigma\). Therefore, \(M_{n-1}\) is not dead.

- (d) Let \(T_n\) be the set of transitions enabled at \(M_{n-1}\). Suppose that there is no step \(U \subseteq T_n\) such that \(M_{n-1}[U]M_n\). We can obtain a contradiction similar to that in (b). Hence, there must exist a step \(U \subseteq T_n\) such that \(M_{n-1}[U]M_n\).

Finally, we show that \(M_n\) is dead. Suppose that there are transitions enabled at \(M_n\). Let \(U_{n+1}\) be a step enabled at \(M_n\). For all \(0 < m \leq n + 1\), there is a step \(U_m\) with \(M_{m-1}[U_m]M_m\). By (1) in the proof of Theorem 8, \((\kappa_2^0, \epsilon, s_0, 0)^{n+1} \rightarrow (\theta, \sigma_n, s_{n+1}[w_{n+1}], 0, 1) = c_{n+1}\). According to the proof of Theorem 8, each finite minimal model generated by reducing \(c_{n+1}\) contains more than \(n + 1\) states. Hence, each finite minimal model of \(\kappa_2^0\) with \(\sigma_n\) being the prefix has more than \(n + 1\) states. This contradicts the assumption that \(\sigma\) is a minimal model of \(\kappa_2^0\). Therefore, \(M_n\) is dead, and \(\alpha\) is a finite word of \(Z\).

Theorems 10 and 11 present the relationship between infinite words of \(Z\) and infinite minimal models of \(\kappa_2^0\). The two theorems can be proved similarly to Theorems 6 and 7, respectively. By Theorems 8–11, Corollary 2 can be obtained.

Theorem 10. For each infinite word \(M_0M_1 \ldots \) of \(Z\), there is an infinite minimal model \(\langle s_0[w_0], s_1[w_1], \ldots \rangle\) of \(\kappa_2^0\) such that for all \(h \geq 0\) and all \(q \in P\), \(\Pi_1(s_h[w_h](v_q)) = M_h(q)\).
Theorem 11. For each infinite minimal model \(s_0[w_0], s_1[w_1], \ldots\) of \(\kappa_2^0\), there exists an infinite word \(M_0M_1\ldots\) of \(Z\) such that for all \(h \geq 0\) and all \(q \in P\), \(M_h(q) = \Pi_1(s_h[w_h](v_q))\).

Corollary 2. \(Z\) is concurrently equivalent to \(\kappa_2^0\).

5. Max-concurrency semantics based translation

This section presents the translation directed by the max-concurrency semantics of net systems. The original model \(Z\) is still a 1-safe net system. By adding the Boolean expression \(g_3 \equiv \bigwedge_{t \in T} \bigvee_{q \in \mathbb{P}} v_q < w_{qt} + \sum_{t' \in T} v_{t'} \cdot w_{t'q}\) to the if statement of \(\kappa_2^0\), another MSVL program \(\kappa_2^1\) is obtained as follows:

\[
\kappa_2^1 = \left( \bigwedge_{q \in \mathbb{P}} v_q = m_q \right) \land \text{while}(g_T) \land \left( \bigwedge_{t \in T} (v_t = 1 \lor v_t = 0) \right) \land \text{if}(g_1 \land g_2 \land g_3) \text{then}\{t\} \text{else}\{f\}
\]

As mentioned above, \(g_1\) and \(g_2\) guarantee that the randomly selected step is non-empty and enabled at the current marking, while the added Boolean expression \(g_3\) ensures that the step is a max-step. Intuitively, \(\kappa_2^1\) simulates the max-concurrency semantics of \(Z\). That is, its execution repeats the process that a max-step enabled at the current marking is randomly selected for execution until a dead marking is encountered. The following corollary can be obtained similarly to Corollary 2.

Corollary 3. \(Z\) is max-concurrently equivalent to \(\kappa_2^1\).

6. Implementation and case studies

The three translations from net systems to MSVL programs have been implemented in PN4MSVL (http://pan.baidu.com/s/1c0D1660). As a result, the supporting tool MSV developed for MSVL can be utilized to verify properties of net systems described by PPTL formulas. We have integrated PN4MSVL into MSV as illustrated in Fig. 3. To analyze and verify a net system with MSV, we first build the net system by Workcraft [24] or TINA [4], then we translate it into MSVL programs by PN4MSVL, and finally we use MSV to model, simulate and verify the generated programs.

The 1-safe net system \(Z_0 = (P, T, W_0, M_0)\) in Fig. 4, built by Workcraft, will be used to show an equivalence relation between net systems and the generated MSVL programs for each translation. Furthermore, it will be utilized to depict how to perform model checking of net systems with MSV under each semantics. The concurrent reachability graph \(Z_0\) is shown in Fig. 5, where \(M_0 = \{p_0\}, M_1 = \{p_1, p_2\}, M_2 = \{p_1, p_4\}, M_3 = \{p_3, p_4\}, M_4 = \{p_3\}\) and \(M_5 = \{p_2, p_3\}\).

By PN4MSVL, \(Z_0\) is translated into programs \(\varphi_{Z_0}, \kappa_2^0, \text{and} \kappa_2^1\). Here, only \(\varphi_{Z_0}\) is presented at the bottom of Fig. 3. For convenience, a variable \(q\) (resp. \(t\)) rather than \(v_q\) (resp. \(v_t\)) is introduced in the actual programs for each place \(q\) (resp. transition \(t\)) of the net system.

Next, MSV generates the normal form graphs (NFGs) [12] in Figs. 6-8 from the generated programs, respectively. As claimed in [12], a finite path from the root node (i.e., the node with double circles) to the terminating node (i.e., the black dot) in the NFG is a finite minimal model of the relative program, while an infinite path emanating from the root node in the NFG is an infinite minimal model of the relative program, and vice versa. In Figs. 6-8, each edge \(\text{Edge}_q\) (abbr. \(e_q\)), labeled
by a state formula $w_l$, indicates a state of the minimal models of the programs. Since we focus on the equivalence relations between the words of $Z_0$ and the minimal models of the programs, only the values of the variables introduced for places are presented in the NFGs.

By the interleaving equivalence between $Z_0$ and $\varphi_{Z_0}$, we obtain that for each path $\langle e_{i0}, \ldots, e_{in} \rangle$ in Fig. 6, there is a min word $M_{i0} \ldots M_{in}$ of $Z_0$ such that for all $0 \leq h \leq n$ and all $p \in P_0$, $M_{ih}(p) = \Pi_1(e_{ih}[w_{ih}](p))$, and vice versa. For instance, for the path $e_0e_1e_2e_3e_4$, we can find a min word $M_0M_1M_2M_3$ of $Z_0$; and for the min word $M_0(M_1M_2M_3)^{\omega}$ of $Z_0$, a path $e_0(e_1e_2e_3)^{\omega}$ can be found.

According to the concurrency equivalence (resp. max-concurrency equivalence) between $Z_0$ and $\kappa_0^Z$ (resp. $\kappa_1^Z$), we obtain that for each path $\langle e_{i0}, \ldots, e_{in} \rangle$ in Fig. 7 (resp. Fig. 8), there is a word (resp. a max word) $M_{i0} \ldots M_{in}$ of $Z_0$ such that for all $0 \leq h \leq n$ and all $p \in P_0$, $M_{ih}(p) = \Pi_1(e_{ih}[w_{ih}](p))$, and vice versa. For example, for the path $e_0e_6e_7e_8e_9e_{10}$ in Fig. 7, a word $M_0M_1M_2M_3(M_0M_1M_2)^{\omega}$ of $Z_0$ can be found; and for the word $M_0M_1M_2M_3M_1M_2M_3M_4$ of $Z_0$, we can find a path $e_0e_1e_4e_5e_6e_7e_8e_9$ in Fig. 7. Note that neither of the two words is a max word or a min word. Moreover, for the path $e_0(e_1e_4e_5)^{\omega}$ in Fig. 8, we can find a max word $M_0(M_1M_2M_3)^{\omega}$ of $Z_0$; and for the max word $M_0M_1M_2M_3M_4$ of $Z_0$, a path $e_0e_1e_2e_3$ in Fig. 8 can be found.

The property to be verified is the PPTL formula mentioned in Section 1, i.e., $\odot \varphi \rightarrow (\odot 4 \land \odot q)^*$. Here the atomic proposition $q$ is defined by $p_0 = 1$. MSV is used to check whether $\varphi_{Z_0}$, $\kappa_0^Z$, and $\kappa_1^Z$ satisfy the property, respectively. Note that MSV provides the counterexamples iff an MSVL program does not satisfy a PPTL formula. Since no counterexample is generated by MSV in the verification of $\varphi_{Z_0}$, we have that $\varphi_{Z_0}$ satisfies the property. Hence, $Z_0$ satisfies the property under the interleaving semantics. In the verifications of $\kappa_0^Z$ and $\kappa_1^Z$, MSV provides the same counterexample $e_0e_1e_2e_3$ in Figs. 9 and 10, respectively. This states that neither of them satisfies the property. Consequently, $Z_0$ does not satisfy the property under the concurrency or max-concurrency semantics.
Fig. 7. NFG of $\kappa_2^0$.

\begin{align*}
\text{Edge}_0: w_0 &= p_0 = 1 \land p_1 = 0 \land p_2 = 0 \land p_3 = 0 \land p_4 = 0 \land p_5 = 0 \\
\text{Edge}_1: w_1 &= p_0 = 0 \land p_1 = 1 \land p_2 = 1 \land p_3 = 0 \land p_4 = 0 \land p_5 = 0
\end{align*}

Fig. 8. NFG of $\kappa_{1,0}^1$.

\begin{align*}
\text{Edge}_0: w_0 &= p_0 = 1 \land p_1 = 0 \land p_2 = 0 \land p_3 = 0 \land p_4 = 0 \land p_5 = 0 \\
\text{Edge}_1: w_1 &= p_0 = 0 \land p_1 = 1 \land p_2 = 1 \land p_3 = 0 \land p_4 = 0 \land p_5 = 0 \\
\text{Edge}_2: w_2 &= p_0 = 0 \land p_1 = 0 \land p_2 = 0 \land p_3 = 1 \land p_4 = 1 \land p_5 = 0 \\
\text{Edge}_3: w_3 &= p_0 = 0 \land p_1 = 0 \land p_2 = 0 \land p_3 = 0 \land p_4 = 0 \land p_5 = 0 \\
\text{Edge}_4: w_4 &= p_0 = 0 \land p_1 = 0 \land p_2 = 0 \land p_3 = 1 \land p_4 = 1 \land p_5 = 0 \\
\text{Edge}_5: w_5 &= p_0 = 0 \land p_1 = 0 \land p_2 = 0 \land p_3 = 1 \land p_4 = 1 \land p_5 = 0 \\
\text{Edge}_6: w_6 &= p_0 = 0 \land p_1 = 0 \land p_2 = 0 \land p_3 = 1 \land p_4 = 1 \land p_5 = 0 \\
\text{Edge}_7: w_7 &= p_0 = 0 \land p_1 = 0 \land p_2 = 0 \land p_3 = 1 \land p_4 = 1 \land p_5 = 0 \\
\text{Edge}_8: w_8 &= p_0 = 0 \land p_1 = 0 \land p_2 = 0 \land p_3 = 1 \land p_4 = 1 \land p_5 = 0 \\
\text{Edge}_9: w_9 &= p_0 = 0 \land p_1 = 0 \land p_2 = 0 \land p_3 = 1 \land p_4 = 1 \land p_5 = 0 \\
\text{Edge}_{10}: w_{10} &= p_0 = 0 \land p_1 = 0 \land p_2 = 1 \land p_3 = 0 \land p_4 = 0 \land p_5 = 0 \\
\text{Edge}_{11}: w_{11} &= p_0 = 0 \land p_1 = 0 \land p_2 = 1 \land p_3 = 1 \land p_4 = 0 \land p_5 = 0 \\
\text{Edge}_{12}: w_{12} &= p_0 = 0 \land p_1 = 0 \land p_2 = 0 \land p_3 = 1 \land p_4 = 1 \land p_5 = 0 \\
\text{Edge}_{13}: w_{13} &= p_0 = 0 \land p_1 = 0 \land p_2 = 0 \land p_3 = 1 \land p_4 = 1 \land p_5 = 0 \\
\text{Edge}_{14}: w_{14} &= p_0 = 0 \land p_1 = 0 \land p_2 = 0 \land p_3 = 1 \land p_4 = 1 \land p_5 = 0 \\
\text{Edge}_{15}: w_{15} &= p_0 = 0 \land p_1 = 0 \land p_2 = 0 \land p_3 = 1 \land p_4 = 1 \land p_5 = 0
\end{align*}

Fig. 9. Verification result of $\kappa_9^0$. 

\begin{align*}
\text{Edge}_0: w_0 &= p_0 = 1 \land p_1 = 0 \land p_2 = 0 \land p_3 = 0 \land p_4 = 0 \land p_5 = 0 \\
\text{Edge}_1: w_1 &= p_0 = 0 \land p_1 = 1 \land p_2 = 1 \land p_3 = 0 \land p_4 = 0 \land p_5 = 0 \\
\text{Edge}_2: w_2 &= p_0 = 0 \land p_1 = 0 \land p_2 = 0 \land p_3 = 1 \land p_4 = 1 \land p_5 = 0 \\
\text{Edge}_3: w_3 &= p_0 = 0 \land p_1 = 0 \land p_2 = 0 \land p_3 = 0 \land p_4 = 0 \land p_5 = 0 \\
\text{Edge}_4: w_4 &= p_0 = 0 \land p_1 = 0 \land p_2 = 0 \land p_3 = 1 \land p_4 = 1 \land p_5 = 0 \\
\text{Edge}_5: w_5 &= p_0 = 0 \land p_1 = 0 \land p_2 = 0 \land p_3 = 1 \land p_4 = 1 \land p_5 = 0 \\
\text{Edge}_6: w_6 &= p_0 = 0 \land p_1 = 0 \land p_2 = 0 \land p_3 = 1 \land p_4 = 1 \land p_5 = 0 \\
\text{Edge}_7: w_7 &= p_0 = 0 \land p_1 = 0 \land p_2 = 0 \land p_3 = 1 \land p_4 = 1 \land p_5 = 0 \\
\text{Edge}_8: w_8 &= p_0 = 0 \land p_1 = 0 \land p_2 = 0 \land p_3 = 1 \land p_4 = 1 \land p_5 = 0 \\
\text{Edge}_9: w_9 &= p_0 = 0 \land p_1 = 0 \land p_2 = 0 \land p_3 = 1 \land p_4 = 1 \land p_5 = 0 \\
\text{Edge}_{10}: w_{10} &= p_0 = 0 \land p_1 = 0 \land p_2 = 1 \land p_3 = 0 \land p_4 = 0 \land p_5 = 0 \\
\text{Edge}_{11}: w_{11} &= p_0 = 0 \land p_1 = 0 \land p_2 = 1 \land p_3 = 1 \land p_4 = 0 \land p_5 = 0 \\
\text{Edge}_{12}: w_{12} &= p_0 = 0 \land p_1 = 0 \land p_2 = 0 \land p_3 = 1 \land p_4 = 1 \land p_5 = 0 \\
\text{Edge}_{13}: w_{13} &= p_0 = 0 \land p_1 = 0 \land p_2 = 0 \land p_3 = 1 \land p_4 = 1 \land p_5 = 0 \\
\text{Edge}_{14}: w_{14} &= p_0 = 0 \land p_1 = 0 \land p_2 = 0 \land p_3 = 1 \land p_4 = 1 \land p_5 = 0 \\
\text{Edge}_{15}: w_{15} &= p_0 = 0 \land p_1 = 0 \land p_2 = 0 \land p_3 = 1 \land p_4 = 1 \land p_5 = 0
\end{align*}
7. Conclusion

Three translations and the supporting tool PN4MSVL from Petri nets to MSVL programs have been presented in this paper. Directed by one semantics of Petri nets, each translation accurately establishes an equivalence relation between the original models and generated programs. These translations make it possible to apply the supporting tool MSV for verifying properties of Petri nets specified by PPTL formulas under each semantics. In the near future, we intend to compare the efficiency of the proposed verification method with the model checking methods available for Petri nets. We also plan to explore abstract and bounded model checking approaches with MSVL to overcome the state space explosion problem. In addition, how the approach can be applied to unbounded Petri nets [30,31] remains to be performed.

Appendix A. Reduction rules

Table A1
Evaluation rules of expressions [33].

| A | (1) \((n, \sigma_{-1}, s_i, i) \downarrow n \) (2) \((x, \sigma_{-1}, s_i, i) \downarrow \Pi_1(s_i(x)) \) (3) \((\sqcup^n \sqcap^n x, \sigma_{-1}, s_i, i) \downarrow \Pi_1(s_i,m(x)) \) if \(1 \leq n \leq m \leq i \) (4) \((\sqcup^n \sqcap^n x, \sigma_{-1}, s_i, i) \downarrow \Pi_1(s_i,m(x)) \) if \(1 \leq n - m \leq i \) |
| B | (1) \((true, \sigma_{-1}, s_i, i) \downarrow true \) (2) \((false, \sigma_{-1}, s_i, i) \downarrow false \) (3) \((\sigma_{-1}, s_i, i, 0 < n, (s_i, \sigma_{-1}, s_i, i) \downarrow n^f) \) \(\Rightarrow \) \(true\) if \(n = n_1\) \(false\) otherwise (4) \((\sigma_{-1}, s_i, i, 0 < n, (s_i, \sigma_{-1}, s_i, i) \downarrow n^f) \) \(\Rightarrow \) \(true\) if \(n = n_1\) \(false\) otherwise (5) \((\sigma_{-1}, s_i, i, 0 < n, (s_i, \sigma_{-1}, s_i, i) \downarrow n^f) \) \(\Rightarrow \) \(true\) if \(n \geq 0\) \(false\) otherwise (6) \((\sigma_{-1}, s_i, i, 0 < n, (s_i, \sigma_{-1}, s_i, i) \downarrow n^f) \) \(\Rightarrow \) \(true\) if \(n = n_1\) \(false\) otherwise |

Table A2
Semantic equivalence rules of programs [33].

| ASS | \(\{1\text{bit}(x) ; x = e\} \equiv x = e\) (2) \(\{1\text{bit}(x) ; x = e\} \equiv x = \overline{e}\) |
| UAss | \(x := e \equiv \bigcirc(x \leftrightarrow \bigcirc x \wedge e)\) |
| NEXT | (1) \(\bigcirc(p \wedge q) \equiv \bigcirc(p \land q)\) (2) \(\bigcirc p \equiv \land\bigcirc p, \text{more}\) |
| CHOP | (1) \(\Lambda\{w, p ; q = \Lambda\{w, p ; q\}\} (2) \bigcirc p ; q = \bigcirc(p ; q) (3) \epsilon ; q = q\) |
| IF | if \(b\) then \(p\) else \(q\) \(\epsilon\) |
| PAR | \(p \parallel q = \bigvee\{p, q, true, q\}, \land\{p, q, true\}\) |
| Guard | \(\bigvee\Pi^n_b(p, \epsilon) \equiv \bigvee\Pi^n_b(p, \epsilon) \) (1) \(b \rightarrow p_i\) \equiv \(\bigvee\Pi^n_b(p, \epsilon) \) (2) \(b \rightarrow p_i\) \equiv \(\bigvee\Pi^n_b(p, \epsilon) \) (3) \(b \rightarrow p_i\) \equiv \(\bigvee\Pi^n_b(p, \epsilon) \) |
| FR | (1) \(\Lambda\{true, \epsilon\} = \epsilon\) (2) \(\Lambda\{true, \epsilon\} = \epsilon\) (3) \(\Lambda\{true, \epsilon\} = \epsilon\) |
| CONG | (1) \(\bigvee\Pi^n_b(p, \epsilon) \equiv \bigvee\Pi^n_b(p, \epsilon)\) |

| | FFT | (1) \(\Lambda\{false, p\} = false\) (2) \(\Lambda\{false, p\} = p\) (3) \(\Lambda\{false, \epsilon\} = false\) |
| | | (1) \(\Lambda\{true, p\} = p\) (2) \(\Lambda\{true, p\} = true\) (3) \(\Lambda\{true, \epsilon\} = true\) |
Table A3

<table>
<thead>
<tr>
<th>Transition rules within a state [33].</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) If $e, \sigma_{i-1}, s_i, i \notin n$ and there is no state component $1\sigma(x)$ or $x \equiv e$ in $p$, (\land (p, x \Rightarrow e), \sigma_{i-1}, s_i, i) \rightarrow (p, \sigma_{i-1}, s_i)(n, p_i)x, l, i))</td>
</tr>
<tr>
<td>Min (1\sigma_{i-1}, s_i, i \notin n) \rightarrow {p, \sigma_{i-1}, s_i)(n, p_i)x, l, i))</td>
</tr>
<tr>
<td>(3) If $x \equiv e$ in $p$, (\land (p, x \Rightarrow e), \sigma_{i-1}, s_i, i) \rightarrow (p, \sigma_{i-1}, s_i)(n, p_i)x, l, i))</td>
</tr>
<tr>
<td>CONG II (p \rightarrow p_{i+1}, s_i, i \notin n) \rightarrow {p, \sigma_{i-1}, s_i)(n, p_i)x, l, i))</td>
</tr>
</tbody>
</table>

Table A4

<table>
<thead>
<tr>
<th>Interval transition rules [33].</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tr (p, \sigma_{i-1}, s_i, i \in {p, \sigma_{i-1}, s_i}(n, p_i)x, l, i))</td>
</tr>
<tr>
<td>(1) $\big((\lor p, \sigma_{i-1}, s_i, i) \rightarrow (p, \sigma_{i-1}, s_i)(n, p_i)x, l, i))</td>
</tr>
<tr>
<td>Non-Or (p_{i-1}, s_i, i \notin n) \rightarrow {p, \sigma_{i-1}, s_i)(n, p_i)x, l, i))</td>
</tr>
</tbody>
</table>

References