Theoretical Computer Science 554 (2014) 169–190

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

A practical decision procedure for Propositional Projection Temporal Logic with infinite models ☆

Zhenhua Duan, Cong Tian *

ICTT and ISN Laboratory, Xidian University, Xi'an, 710071, PR China

ARTICLE INFO

Available online 18 February 2014

Keywords:
Decision procedure
Propositional Projection Temporal Logic
Büchi accepting condition
Automata
Algorithm

ABSTRACT

This paper presents a practical decision procedure for Propositional Projection Temporal Logic with infinite models. First, a set $Prop_l$ of labels l_i, $0 \leqslant i \leqslant n \in N_0$, is used to mark nodes of an LNFG of a formula, and a node with l_i is treated as an accepting state as in an automaton. Further, the generalized Büchi accepting condition for automata is employed to identify a path (resulting a word) in an LNFG as a model of the formula. In addition, the implementation details of the decision procedure and relevant algorithms including pre-processing, LNFG, circle finding algorithms are presented; as a matter of fact, all algorithms are implemented by C++ programs.

1. Introduction

A decision procedure for Propositional Projection Temporal Logic (PPTL) with infinite models was given in [1], and further improved in [2]. The decision procedure can also be applied to Propositional Interval Temporal Logic (PITL) [5] and Propositional Linear Temporal Logic (PLTL) [9] with minor modification. With this decision procedure, Normal Form (NF), Normal Form Graph (NFG) and Labeled NFG (LNFG) play important roles for constructing models of a formula. Generally, given a formula P, all models of P are contained in its NFG, and determined by its LNFG. That is, an NFG can be treated as an automaton structure while an LNFG can be viewed as an omega automaton with a specified accepting condition. Nevertheless, accepting conditions on LNFG was only simply expressed in an informal way. Also, the decision procedure and the relative algorithms are only written in pseudo code. When implementing the decision procedure, we found that to precisely define accepting conditions is neither a trivial nor an intuitive work. This paper focuses on the practical aspects of the implementation of the decision procedure such as the acceptance condition, data structures, implementation details of the relevant algorithms. The main contributions of the papers are as follows: (1) A set $Prop_l$ of labels l_i, $0 \leqslant i \leqslant n \in N_0$, is used to mark nodes of NFG of a formula R, producing LNFG of R, and a node with l_i is treated as an accepting state as in an automaton. Further, the generalized Büchi accepting condition for an automaton is employed to identify a path (resulting a word) in an LNFG of R as a model of the formula R. (2) The decision procedure and relevant algorithms are all implemented by C++ programs.

The paper is organized as follows: in the following section, Propositional Projection Temporal Logic is briefly presented. Section 3 reviews the Normal Form (NF) and Normal Form Graph (NFG); a set of labels is introduced to mark the nodes

☆ This research is supported by the National Program on Key Basic Research Project of China (973 Program) Grant No. 2010CB328102, National Natural Science Foundation of China under Grant Nos. 61133001, 61202038, 61272117, 61272118, 61322202 and 61373043.

* Corresponding author.
E-mail addresses: zhhduan@mail.xidian.edu.cn (Z. Duan), ctian@mail.xidian.edu.cn (C. Tian).

http://dx.doi.org/10.1016/j.tcs.2014.02.011
0304-3975/

of NFG; thus, Labeled NFG (LNFG) with many labels are constructed. In Section 4, accepting conditions with LNFG are generalized, and an improved and more practical decision procedure is formalized. In Section 5, some implementation details are explained. Further, examples are given to illustrate how the decision procedure works. Conclusions are drawn in Section 6.

2. Propositional Project Temporal Logic

Propositional Projection Temporal Logic (PPTL) [8,3] is an extension of Propositional ITL (PITL) [5] with infinite models and a new projection construct [4]. Let *Prop* be a countable set of atomic propositions and $B = \{true, false\}$ the boolean domain. Usually, we use small letters, possibly with subscripts, like p, q, r to denote atomic propositions and capital letters, possibly with subscripts, like P, Q, R to represent general PPTL formulas. Then the formulas of PPTL are defined by the following grammar:

$$P ::= p \mid \neg P \mid P_1 \wedge P_2 \mid \bigcirc P \mid (P_1, \ldots, P_m)\,\mathsf{prj}\,P \mid P^+$$

where $p \in Prop$, $\bigcirc$(next), $+$ (chop-plus) and prj (projection) are temporal operators, and $\neg, \wedge$ are similar as that in the classical propositional logic.

We define a *state* s over *Prop* to be a mapping from *Prop* to B, $s : Prop \to B$. We write $s[p]$ to denote the valuation of p at state s. An *interval* $\sigma = \langle s_0, s_1, \ldots, \rangle$ is a non-empty sequence of states, which can be finite or infinite. The length of σ, $|\sigma|$, is the number of states in σ minus one if σ is finite; otherwise it is ω. Let N_0 denote the set of non-negative integers. To have a uniform notation for both finite and infinite intervals, we will use *extended integers* as indices, that is $N_\omega = N_0 \cup \{\omega\}$, and extend the comparison operators, $=, <, \leqslant$, to N_ω by considering $\omega = \omega$ and for all $i \in N_0, i < \omega$. Moreover, we write $\preceq$ as $\leqslant -\{(\omega, \omega)\}$. To simplify definitions, we will denote σ by $\langle s_0, \ldots, s_{|\sigma|}\rangle$, where $s_{|\sigma|}$ is undefined if σ is infinite. With such a notation, $\sigma_{(i..j)}$ $(0 \leqslant i \preceq j \leqslant |\sigma|)$ denotes the sub-interval $\langle s_i, \ldots, s_j\rangle$.

To formalize the semantics of the projection construct, we need an auxiliary operator $\downarrow$. Let $\sigma = \langle s_0, s_1, \ldots\rangle$ be an interval and $r_1, \ldots, r_h$ be integers $(h \geqslant 1)$ such that $0 \leqslant r_1 \leqslant \cdots \leqslant r_h \preceq |\sigma|$. The projection of σ onto $r_1, \ldots, r_h$ is the *projected interval*, $\sigma \downarrow (r_1, \ldots, r_h) \overset{\text{def}}{=} \langle s_{t_1}, s_{t_2}, \ldots, s_{t_l}\rangle$, where $t_1, \ldots, t_l$ are attained from $r_1, \ldots, r_h$ by deleting all duplicates. In other words, $t_1, \ldots, t_l$ is the longest strictly increasing subsequence of $r_1, \ldots, r_h$. For instance, $\langle s_0, s_1, s_2, s_3\rangle \downarrow (0, 2, 2, 2, 3) = \langle s_0, s_2, s_3\rangle$. The concatenation $(\cdot)$ of a finite interval $\sigma = \langle s_0, s_1, \ldots, s_{|\sigma|}\rangle$ with another interval $\sigma' = \langle s'_0, s'_1, \ldots, s'_{|\sigma|}\rangle$ is represented by $\sigma \cdot \sigma' = \langle s_0, s_1, \ldots, s_{|\sigma|}, s'_0, s'_1, \ldots, s'_{|\sigma|}\rangle$ (not sharing any states).

An *interpretation* is a tuple $\mathcal{I} = (\sigma, k, j)$, where $\sigma = \langle s_0, s_1, \ldots\rangle$ is an interval, k is a non-negative integer, and j is an integer or ω, such that $0 \leqslant k \preceq j \leqslant |\sigma|$. We write (σ, k, j) to mean that a formula is interpreted over a subinterval $\sigma_{(k..j)}$ with the current state being s_k. We utilize I^k_{prop} to stand for the state interpretation at state s_k. The satisfaction relation $\models$ for formulas is given as follows:

$$
\begin{array}{ll}
\mathcal{I} \models p & \text{iff } s_k[p] = I^k_{prop}[p] = true \\
\mathcal{I} \models \neg P & \text{iff } \mathcal{I} \not\models P \\
\mathcal{I} \models P_1 \wedge P_2 & \text{iff } \mathcal{I} \models P_1 \text{ and } \mathcal{I} \models P_2 \\
\mathcal{I} \models \bigcirc P & \text{iff } k < j \text{ and } (\sigma, k+1, j) \models P \\
\mathcal{I} \models (P_1, \ldots, P_m)\,\mathsf{prj}\,P & \text{iff there exist integers } r_0, \ldots, r_m, \text{ and } k = r_0 \leqslant \cdots \leqslant r_{m-1} \preceq r_m \leqslant j \\
& \quad \text{such that } (\sigma, r_{l-1}, r_l) \models P_l \text{ for all} \\
& 1 \leqslant l \leqslant m \text{ and } (\sigma', 0, |\sigma'|) \models P \text{ for } \sigma' \text{ given by:} \\
& (1)\, r_m < j \text{ and } \sigma' = \sigma \downarrow (r_0, \ldots, r_m) \cdot \sigma_{(r_m+1, \ldots, j)} \\
& (2)\, r_m = j \text{ and } \sigma' = \sigma \downarrow (r_0, \ldots, r_h) \text{ for some } 0 \leqslant h \leqslant m
\end{array}
$$

$$
\begin{array}{ll}
\mathcal{I} \models P^+ & \text{iff there are finitely many integers } r_0, \ldots, r_n \text{ and} \\
& k = r_0 \leqslant r_1 \leqslant \cdots \leqslant r_{n-1} \preceq r_n = j \ (n \geqslant 1) \text{ such that} \\
& (\sigma, r_{l-1}, r_l) \models P \text{ for all } 1 \leqslant l \leqslant n; \text{ or } j = \omega \text{ and there are} \\
& \ \text{infinitely many integers } k = r_0 \leqslant r_1 \leqslant r_2 \leqslant \cdots \text{ such that} \\
& \lim\limits_{i \to \infty} r_i = \omega \text{ and } (\sigma, r_{l-1}, r_l) \models P \text{ for all } l \geqslant 1.
\end{array}
$$

For convenience, some derived formulas from elementary PPTL formulas are shown below, which are explained in [3,8]. The abbreviations true, false, $\vee$, $\to$ and $\leftrightarrow$ are defined as usual.

$$
\begin{array}{llll}
\varepsilon \stackrel{\text{def}}{=} \neg \bigcirc \text{true} & & \text{more} \stackrel{\text{def}}{=} \neg\varepsilon \\
\Diamond P \stackrel{\text{def}}{=} (\text{true}, P)\,\text{prj}\,\varepsilon & & \Box P \stackrel{\text{def}}{=} \neg\Diamond\neg P \\
\text{fin}(P) \stackrel{\text{def}}{=} \Box(\varepsilon \to P) & & \text{halt}(P) \stackrel{\text{def}}{=} \Box(\varepsilon \leftrightarrow P) \\
\text{keep}(P) \stackrel{\text{def}}{=} \Box(\neg\varepsilon \to P) & & \text{rem}(P) \stackrel{\text{def}}{=} \Box(\text{more} \to \bigcirc P) \\
P\,;Q \stackrel{\text{def}}{=} (P, Q)\,\text{prj}\,\varepsilon & & P\,;_{\text{w}} Q \stackrel{\text{def}}{=} (P\,;Q) \vee (P \wedge \Box\text{more}) \\
\text{fin} \stackrel{\text{def}}{=} \Diamond\varepsilon & & \text{inf} \stackrel{\text{def}}{=} \Box\text{more} \\
P^* \stackrel{\text{def}}{=} P^+ \vee \varepsilon & & \text{len}(n) \stackrel{\text{def}}{=} \begin{cases} \varepsilon & \text{if } n = 0 \\ \bigcirc\text{len}(n-1) & \text{if } n > 1 \end{cases}
\end{array}
$$

Usually, $\models \Box(P \leftrightarrow Q)$ is represented by $P \equiv Q$ (*strong equivalence*), meaning that P and Q have the same truth value at all states of any models while $\models \Box(P \to Q)$ is denoted by $P \supset Q$ (*strong implication*), stating that $P \to Q$ is true at all states of any models. The following are some useful logic laws. Here w is a state formula. The proofs of the logic laws can be found in [3].

$$
\begin{array}{lll}
L_1 & \Box(P \wedge Q) & \equiv \Box P \wedge \Box Q \\
L_2 & \Diamond(P \vee Q) & \equiv \Diamond P \vee \Diamond Q \\
L_3 & \bigcirc(P \vee Q) & \equiv \bigcirc P \vee \bigcirc Q \\
L_4 & \bigcirc(P \wedge Q) & \equiv \bigcirc P \wedge \bigcirc Q \\
L_5 & R\,;(P \vee Q) & \equiv (R\,;P) \vee (R\,;Q) \\
L_6 & (P \vee Q)\,;R & \equiv (P\,;R) \vee (Q\,;R) \\
L_7 & \Diamond P & \equiv P \vee \bigcirc\Diamond P \\
L_8 & \Box P & \equiv P \wedge \odot\Box P \\
L_9 & \text{more} \wedge \neg\bigcirc P & \equiv \text{more} \wedge \bigcirc\neg P \\
L_{10} & \neg\odot P & \equiv \bigcirc\neg P \\
L_{11} & \bigcirc P\,;Q & \equiv \bigcirc(P\,;Q) \\
L_{12} & w \wedge (P\,;Q) & \equiv (w \wedge P)\,;Q \\
L_{13} & Q\ prj\ \varepsilon & \equiv Q \\
L_{14} & \varepsilon\ prj\ Q & \equiv Q \\
L_{15} & (P_1, \ldots P_m)\ prj\ \varepsilon & \equiv P_1; \ldots; P_m \\
L_{16} & (P, \varepsilon)\ prj\ Q & \equiv (P \wedge \Diamond\varepsilon)\ prj\ Q \\
L_{17} & (P_1, \ldots, w \wedge \varepsilon, P_t, \ldots, P_m)\ prj\ Q & \equiv (P_1, \ldots, w \wedge P_t, \ldots, P_m)\ prj\ Q \\
L_{18} & (P_1, \ldots, P_i \vee P'_i, \ldots, P_m)\ prj\ Q & \equiv (P_1, \ldots, P_i, \ldots, P_m)\ prj\ Q \vee (P_1, \ldots, P'_i, \ldots, P_m)\ prj\ Q \\
L_{19} & (P_1, \ldots, P_m)\ prj\ (P \vee Q) & \equiv (P_1, \ldots, P_m)\ prj\ P \vee (P_1, \ldots, P_m)\ prj\ Q \\
L_{20} & (P_1, \ldots, P_m)\ prj\ \bigcirc Q & \equiv \big(P_1 \wedge \text{more}\,;(P_2, \ldots, P_m)\ prj\ Q\big) \vee \big(P_1 \wedge \varepsilon\,;(P_2, \ldots, P_m)\ prj\ \bigcirc Q\big) \\
L_{21} & (\bigcirc P_1, \ldots, P_m)\ prj\ \bigcirc Q & \equiv \bigcirc\big(P_1\,;(P_2, \ldots, P_m)\ prj\ Q\big) \\
L_{22} & (w \wedge P_1, \ldots, P_m)\ prj\ Q & \equiv w \wedge \big((P_1, \ldots, P_m)\ prj\ Q\big) \\
L_{23} & (P_1, \ldots, P_m)\ prj\ (w \wedge Q) & \equiv w \wedge \big((P_1, \ldots, P_m)\ prj\ Q\big)
\end{array}
$$

By the derived formulas and logic laws, we can further prove the following conclusions [1,3]:

$$\begin{array}{llll} \mathsf{fin}(P) & \equiv P \wedge \varepsilon \vee \bigcirc \mathsf{fin}(P) & \mathsf{keep}(P) & \equiv \varepsilon \vee P \wedge \bigcirc \mathsf{keep}(P) \\ \mathsf{halt}(P) & \equiv P \wedge \varepsilon \vee \neg P \wedge \bigcirc \mathsf{halt}(P) & \mathsf{rem}(P) & \equiv \varepsilon \vee \bigcirc\big(P \wedge \mathsf{rem}(P)\big) \\ \mathsf{inf} & \equiv \mathsf{true}\,;_{\mathsf{w}}\mathsf{false} & \mathsf{true} & \equiv \varepsilon \vee \bigcirc \mathsf{true} \end{array}$$

3. Normal Form and Normal Form Graph

Normal Form (NF) and Normal Form Graph (NFG) are useful in constructing LNFGs of PPTL formulas. The details of these concepts can be found in [1]. In the following, only a brief introduction is given.

Definition 1 *(Normal Form).* Let Q_p be the set of atomic propositions appearing in a PPTL formula Q. The normal form of Q can be defined as follows.

$$Q \equiv \bigvee_{j=0}^{n_0} (Q_{ej} \wedge \varepsilon) \vee \bigvee_{i=0}^{n_1} \big(Q_{ci} \wedge \bigcirc Q_i'\big)$$

where $Q_{ej} \equiv \bigwedge_{k=1}^{m_0} \dot{q}_{jk}$, $Q_{ci} \equiv \bigwedge_{h=1}^{m} \dot{q}_{ih}$, $q_{jk}, q_{ih} \in Q_p$, for any $r \in Q_p$, $\dot{r}$ denotes r or $\neg r$; Q_i' is a PPTL formula without "$\vee$" being the main operator. □

According to the definition, in a normal form, $p \wedge q \wedge \bigcirc(\Box p \vee q)$ must be written as $p \wedge q \wedge \bigcirc(\Box p) \vee p \wedge q \wedge \bigcirc q$ since "$\vee$" is the main operator of $\Box p \vee q$. Implicitly, Q_i' is also not permitted to be the form of $\neg \bigwedge_{k=1}^{n \geqslant 2} P_k$. Further, for convenience, we call $\bigvee_{j=0}^{n_0}(Q_{ej} \wedge \varepsilon)$ the terminating part whereas $\bigvee_{i=0}^{n_1}(Q_{ci} \wedge \bigcirc Q_i')$ the non-terminating part of the normal form. The reduction process for obtaining a normal form of a formula is known as normal form reduction.

Definition 2 *(Complete Normal Form).* Let Q_p be the set of atomic propositions appearing in a PPTL formula Q. The complete normal form of Q is defined by,

$$Q \equiv \bigvee_{j=0}^{n_0} (Q_{ej} \wedge \varepsilon) \vee \bigvee_{i=0}^{n_1} \big(Q_{ci} \wedge \bigcirc Q_i'\big)$$

where like in the normal form, $Q_{ej} \equiv \bigwedge_{k=1}^{m_0} \dot{q}_{jk}$, $Q_{ci} \equiv \bigwedge_{h=1}^{m} \dot{q}_{ih}$, $q_{jk}, q_{ih} \in Q_p$, for any $r \in Q_p$, $\dot{r}$ denotes r or $\neg r$; further $\bigvee_i Q_{ci} \equiv \mathit{true}$ and $\bigvee_{i \neq j}(Q_{ci} \wedge Q_{cj}) \equiv \mathit{false}$; Q_i' is an arbitrary PPTL formula. □

Note that a complete normal form may be not a normal form, since "$\vee$" is possibly the main operator of Q_i'.

For a PPTL formula P, Normal Form Graph (NFG for short) of P is a directed graph, $G = (CL(P), EL(P), V_0)$, where $CL(P)$ denotes the set of nodes, $EL(P)$ the set of edges, and $V_0 \subseteq CL(P)$ the set of root nodes in the graph. Each node in $CL(P)$ is specified by a formula in PPTL, while each edge in $EL(P)$ is a directed arc from a node such as Q to another node such as R, labeled with a state formula such as Q_e, and identified by a triple, (Q, Q_e, R). Accordingly, for convenience sometimes, a node in an NFG or Labeled NFG is called a formula. The NFG of a PPTL formula is inductively defined in Definition 3.

Definition 3 *(Normal Form Graph, NFG).* For a PPTL formula P, the set $CL(P)$ of nodes and the set $EL(P)$ of edges connecting nodes in $CL(P)$ are inductively defined as follows:

1. Initially, let $V_0 = CL(P) = EL(P) = \emptyset$;
2. Let $P \equiv \bigvee_i P_i$. For each i $P_i \in V_0$, $P_i \in CL(P)$;
3. For all $Q \in CL(P) \setminus \{\varepsilon, \mathit{false}\}$, if Q is rewritten into its normal form $\bigvee_{j=0}^{h}(Q_{ej} \wedge \varepsilon) \vee \bigvee_{i=0}^{k}(Q_{ci} \wedge \bigcirc Q_i')$, then $\varepsilon \in CL(P)$, $(Q, Q_{ej}, \varepsilon) \in EL(P)$ for each j, $1 \leqslant j \leqslant h$; $Q_i' \in CL(P)$, $(Q, Q_{ci}, Q_i') \in EL(P)$ for all i, $1 \leqslant i \leqslant k$;

The NFG of formula P is the directed graph $G = (CL(P), EL(P), V_0)$.

In an NFG, any root node in V_0 is denoted by a circle with an incoming edge without a source, ε node is marked by a small black dot, and each of other nodes by a single circle. Each edge is denoted by a directed arc connecting two nodes. A finite path is a finite alternating sequence of nodes and edges, $\pi = \langle n_0, e_0, n_1, e_1, \ldots, \varepsilon\rangle$ from a root node to the ε node, while an infinite path is an infinite alternating sequence of nodes and edges, $\pi = \langle n_0, e_0, n_1, e_1, \ldots, n_i, e_i, \ldots, n_j, e_j, n_i, e_i, \ldots, n_j, e_j, \ldots\rangle$ departing from the root node with some nodes, e.g. $n_i, \ldots, n_j$, occurring for infinitely many times. For convenience, we use $\mathsf{Inf}(\pi)$ to denote the set of nodes which infinitely often occur in the infinite path π. In some circumstances, in a path of NFG of formula Q, a node n_i can be replaced by a formula $Q_i \in CL(Q)$ and an edge e_i can be replaced by a state formula $Q_{ie} \in EL(Q)$.

Intuitively, all models of a formula are implicitly contained in its NFG. For easily expressing the relationship between models of a formula Q and paths of NFG G of Q, functions $P2M(\pi, G)$ and $M2P(\sigma, G)$ are formally defined below. Let Π_{pptl} be the set of all pptl formulas, Δ the set of NFGs of all formulas in Π_{pptl}, Σ_G the set of all paths in an NFG G, and Γ the set of all intervals. Given a path $\pi = \langle Q, Q_{0e}, Q_1, Q_{1e}, \ldots\rangle$ in an NFG G, an interval σ_π can be obtained by function $P2M : \Sigma_G \times \Delta \longrightarrow \Gamma$ defined as follows.

$$\sigma_\pi = P2M(\pi, G) = \begin{cases} \langle s_0, s_1, \ldots, s_n\rangle, \text{ for } 1 \leqslant i \leqslant n \\ \quad s_i[q] = \mathit{true} \text{ if } q \text{ is in } Q_{ie}, \text{ and } s_i[q] = \mathit{false} \text{ if } \neg q \text{ is in } Q_{ie}, \\ \quad \text{if } \pi = \langle Q, Q_{0e}, Q_1, Q_{1e}, \ldots, Q_{ne}, \varepsilon\rangle \text{ is a finite path} \\ \langle s_0, s_1, \ldots, (s_i, \ldots, s_j)^\omega\rangle, \text{ for } 1 \leqslant k \leqslant j \\ \quad s_k[q] = \mathit{true} \text{ if } q \text{ is in } Q_{ke}, \text{ and } s_k[q] = \mathit{false} \text{ if } \neg q \text{ is in } Q_{ke}, \\ \quad \text{if } \pi = \langle Q, Q_{0e}, Q_1, Q_{1e}, \ldots, (Q_i, Q_{ie}, \ldots, Q_j, Q_{je})^\omega\rangle \text{ is an infinite path.} \end{cases}$$

Correspondingly, for a model $\sigma = \langle s_0, s_1, \ldots\rangle \models Q$, a path π_σ w.r.t. the NFG G of Q can be obtained by function $M2P$: $\Gamma \times \Delta \longrightarrow \Sigma_G$ defined as follows:

$$\pi_\sigma = M2P(\sigma, G) = \begin{cases} \langle Q, Q_{0e}, Q_1, Q_{1e}, \ldots, Q_{ne}, \varepsilon\rangle, \text{ for } 1 \leqslant i \leqslant n \\ \quad Q_{ie} = \bigwedge_k \dot{q_k}, \dot{q_k} \equiv q_k, \text{ if } s_i[q_k] = \mathit{true}, \\ \quad \text{and } \dot{q_k} \equiv \neg q_k, \text{ if } s_i[q_k] = \mathit{false}, Q_i \in CL(Q) \\ \quad \text{if } \sigma = \langle s_0, s_1, \ldots, s_n\rangle \text{ is a finite interval} \\ \langle Q, Q_{0e}, Q_1, Q_{1e}, \ldots, (Q_i, Q_{ie}, \ldots, Q_j, Q_{je})^\omega\rangle, \text{ for } 1 \leqslant h \leqslant j \\ \quad Q_{he} = \bigwedge_k \dot{q_k}, \dot{q_k} \equiv q_k, \text{ if } s_h[q_k] = \mathit{true}, \\ \quad \text{and } \dot{q_k} \equiv \neg q_k, \text{ if } s_h[q_k] = \mathit{false}, Q_h \in CL(Q) \\ \quad \text{if } \sigma = \langle s_0, s_1, \ldots, (s_i, \ldots, s_j)^\omega\rangle \text{ is an infinite interval.} \end{cases}$$

Although in the above an interval σ_π can be defined for a given path π of the NFG of formula Q, whether or not $\sigma_\pi \models Q$ needs to be proved (see Lemmas 2, 6 and 9). Similarly, given a model σ of formula Q, $\sigma \models Q$, although a path π_σ can be constructed, however, whether or not the path can be found in G also needs to be proved (see Lemmas 3 and 5). Before the proofs of theorems and lemmas, we need the following notations: given an interval $\sigma = \langle s_0, s_1, \ldots\rangle$, the ith prefix of σ denoted by σ^i is the subsequence $\langle s_0, \ldots, s_i\rangle$ while the ith suffix of σ denoted by $\sigma^{(i)}$ is the subsequence $\langle s_i, \ldots, s_{|\sigma|}\rangle$. In a similar way, for a given path π, we can define the ith prefix π^i and the ith suffix $\pi^{(i)}$ of π.

Theorem 1. *Finite paths in the NFG of PPTL formula Q precisely characterize finite models of Q.*

Proof. It is a consequence of Lemmas 2 and 3. □

Lemma 2. *For a finite path π in the NFG G of Q, $\sigma_\pi \models Q$.*

Proof. Let $\pi = \langle Q, Q_{0e}, Q_1, Q_{1e}, \ldots Q_{n-1}, Q_{(n-1)e}, Q_n, Q_{ne}, \varepsilon\rangle$ be a finite path of the NFG G of Q. $\sigma_\pi = \langle s_0, s_1, \ldots, s_{n-1}, s_n\rangle$ $(n \geqslant 0)$ can be obtained by function $P2M(\pi, G)$, where for each $i \in N_0$, $s_i[q] = \mathit{true}$ if proposition q in Q_{ie}, otherwise $s_i[q] = \mathit{false}$ if $\neg q$ in Q_{ie}. We need to prove σ_π is a model of formula Q.

Case 1: $n = 0$, the path is $\pi = \langle Q, Q_{0e}, \varepsilon\rangle$ and the interval is $\sigma_\pi = \langle s_0\rangle$. By the construction of s_0, $\langle s_0\rangle \models Q_{0e}$. Thus, σ_π is a model of Q.

Case 2: $n > 0$, in this case, the proof proceeds by induction on the length of the prefix σ_π^i of σ_π.

Base: $\sigma_\pi^0 = \langle s_0\rangle$. By the construction of s_0, $\langle s_0\rangle \models Q_{0e}$. That is σ_π^0 is a prefix of a model of Q.

Induction: Suppose for all $j < n$, prefix σ_π^j of σ_π is a prefix of a model. We prove σ_π^n is a prefix of a model of Q. By hypothesis, $\sigma_\pi^{n-1} = \langle s_0, \ldots, s_{n-1}\rangle$ is a prefix of the model. According to Algorithm NFG, there must be $Q_n \equiv Q_{ne} \wedge \varepsilon$ such that a new node ε and a new edge $(Q_n, Q_{ne}, \varepsilon)$ are added to the NFG of Q. By the construction of s_n, $\langle s_n\rangle \models Q_{ne} \wedge \varepsilon$. Since σ_π^{n-1} is a prefix of the model, so $\sigma_\pi^n = \sigma_\pi$ is a model of Q. □

Lemma 3. *For a finite interval $\sigma \models Q$, π_σ can be found in the NFG G of Q.*

Proof. For a finite interval $\sigma = \langle s_0, s_1, \ldots, s_{n-1}, s_n\rangle$, $(\sigma, 0, |\sigma|) \models Q$, a path $\pi_\sigma = \langle Q, Q_0, Q_1', Q_1, \ldots Q_{n-1}', Q_{n-1}, Q_n', Q_n, \varepsilon\rangle$ can be obtained by function $M2P(\sigma, G)$, where $Q_i = \bigwedge_k \dot{q_k}$, $\dot{q_k} \equiv q_k$, if $s_i[q_k] = \mathit{true}$, and $\dot{q_k} \equiv \neg q_k$ if $s_i[q_k] = \mathit{false}$ for each i, $0 \leqslant i \leqslant n$. We need to prove that π_σ can be found in the NFG of Q.

Case 1: $n=0$, the model $\sigma=\langle s_0\rangle$, and the path $\pi_\sigma=\langle Q, Q_0, \varepsilon\rangle$. By the construction of NFG, there must exist a normal form of Q, $Q\equiv(Q_0\wedge\varepsilon)\vee\bigvee_{j=0}^{k}(Q_{0j}\wedge\bigcirc Q'_{0j})$ such that there exists an edge from root Q to node ε and labeled by Q_0 in G, and $(\sigma,0,|\sigma|)\models Q_0\wedge\varepsilon$.

Case 2: $n>0$, in this case, the proof proceeds by induction on the length of the prefix π_σ^i of π_σ.

Base: $\pi_\sigma^0=\langle Q, Q_0, Q'_1\rangle$. By the construction of NFG, there must exist a normal form of Q, $Q\equiv(Q_{0e}\wedge\varepsilon)\vee\bigvee_{j=0}^{k}(Q_{0j}\wedge\bigcirc Q'_{0j})$ such that there exists an edge from root Q to node $Q'_1\equiv Q'_{0h}$ and labeled by $Q_0\equiv Q_{0h}$ $(0\leqslant h\leqslant k)$ in G, and $(\sigma,0,|\sigma|)\models Q_{0h}\wedge\bigcirc Q'_{0h}$. Here $Q_{0h}\equiv\bigwedge_k\dot{q}_k$, $\dot{q_k}\equiv q_k$, if $s_i[q_k]=\mathit{true}$, and $\dot{q_k}\equiv\neg q_k$ if $s_i[q_k]=\mathit{false}$ for each i, $0\leqslant i\leqslant n$. So, prefix π_σ^0 of π_σ has been found in G.

Induction: Suppose we have found a prefix $\pi_\sigma^{n-1}=\langle Q, Q_0, Q'_1, Q_1,\dots Q_{n-1}, Q'_n\rangle$ of π_σ in G w.r.t. the prefix $\sigma^{n-1}=\langle s_0,\dots,s_{n-1}\rangle$ of model σ. At this point, we can rewrite Q'_n to its normal form, $Q'_n\equiv(Q_{ne}\wedge\varepsilon)\vee\bigvee_{j=0}^{k}(Q_{nj}\wedge\bigcirc Q'_{(n+1)j})$, such that there exists an edge from Q'_n to ε and labeled by $Q_n\equiv Q_{ne}$ in G and $(\sigma,n,|\sigma|)\models Q_{ne}\wedge\varepsilon$. Here $Q_{ne}\equiv\bigwedge_l\dot{r}_l$, and $\dot{r}_l$ is an atomic proposition r_l or $\neg r_l$. Thus, $\langle s_n\rangle\models Q_{ne}\wedge\varepsilon$. That is, $s_n[r_l]=\mathit{true}$ if $\dot{r}_l$ is r_l, and $s_n[r_l]=\mathit{false}$ if $\dot{r}_l$ is $\neg r_l$. So we have found a path $\pi_\sigma=\langle Q, Q_0, Q'_1, Q_1,\dots Q'_{n-1}, Q_{n-1}, Q'_n, Q_n,\varepsilon\rangle$ in G w.r.t. the model σ. $\square$

For infinite models, the relationship between models of a formula and paths of the NFG of the formula is much more intricate because of the involvement of chop and projection operators. In fact, not all of infinite paths in the NFG of a formula are infinite models of the corresponding formula. We investigate the case carefully in the following.

A formula R is called a chop formula if $R\equiv P\wedge Q$ and $P\equiv P_1;P_2$, where P_1, P_2 and Q are any PPTL formulas. Further, $P_1;P_2$ is called a chop component of chop formula R. Note that a chop component is also a chop formula but the reverse may be not true. Formally, a chop formula R_c can be defined as follows

$$R_c ::= P;Q \mid R_c\wedge R$$

where P, Q and R are any PPTL formulas.

For instance, $(\bigcirc p;q)$, $(p;p^*)$, $(\bigcirc p;q)\wedge\bigcirc r$ are chop formulas while $\bigcirc(\bigcirc p;q)$, $\neg(\bigcirc p;q)$ and p^* are not, where p, q, and r are atomic propositions. Note that $P;Q$ is a chop formula but $\neg(P;Q)$ is not. This will be formally analyzed later.

For chop construct $P;Q$, an infinite model $\sigma=\langle s_0,s_1,\dots,s_k,\dots\rangle\models P;Q$ if and only if there exists $i\in N_0$, such that $\sigma^i\models P$ and $\sigma^{(i)}\models Q$. This implies that if an infinite model $\sigma=\langle s_0,s_1,\dots,s_k,\dots\rangle\models P$ but there are no any finite prefixes $\sigma^i\models P$ $(i\in N_0)$, it fails to satisfy $P;Q$. Note that in the weak version of the chop construct, $P;Q$ is still satisfied in the case. In some contexts, for convenience, we say this type of requirement for P is the finiteness (or terminating) property of P. Formally, the finiteness of P, called FSC_Property, in strong chop construct $P;Q$ means that $P;Q$ can be reduced to some formula $P_i\wedge\varepsilon;Q$ by means of repeatedly using normal form reduction and $P_i\wedge Q$ is satisfiable. More precisely, FSC_Property of P in $P;Q$ is satisfiable over an infinite model σ if function $\mathit{fsc}'(\sigma, P;Q)=\mathit{true}$. Formally, function fsc' is defined as $\mathit{fsc}':\Gamma\times\Pi_{pptl}\longrightarrow\{\mathit{true},\mathit{false}\}$, where Π_{pptl} is the set of all PPTL formulas, and $\mathit{fsc}'(\sigma, P;Q)=\mathit{true}$ if there exists $i\in N_0$ such that $\sigma^i\models P$ and $\sigma^{(i)}\models Q$. Actually, an infinite path of the NFG of $P;Q$ presents an infinite model of either $P;Q$ or P. So, FSC_Property of P needs to be considered if a node (formula) is a chop formula.

Correspondingly, in NFGs, when constructing NFG of $P;Q$, initially, $P;Q$ is transformed into its normal form,

$$\begin{aligned}
P;Q &\equiv\left(\bigvee_{j=0}^{n_0}P_{ej}\wedge\varepsilon\vee\bigvee_{i=0}^{n_1}\bigl(P_{ci}\wedge\bigcirc P'_i\bigr)\right);Q\\
&\equiv\bigvee_{j=0}^{n_0}(\underline{P_{ej}\wedge\varepsilon;Q})\vee\bigvee_{i=0}^{n_1}P_{ci}\wedge\bigcirc\bigl(P'_i;Q\bigr)\\
&\equiv\bigvee_{j=0}^{n_0}(P_{ej}\wedge Q)\vee\bigvee_{i=0}^{n_1}P_{ci}\wedge\bigcirc\bigl(P'_i;Q\bigr)
\end{aligned}$$

Subsequently, the new generated formula $P'_i;Q$ needs to be repeatedly transformed into its normal form in the same way. Whenever a final state of P is reached, i.e. $P_e\wedge\varepsilon;Q$ is encountered, where P_e is a state formula, FSC_Property of $P;Q$ is satisfied over the path π departing from the root node.

To formally define FSC_Property of $P;Q$ over a path π by means of function fsc' we could transfer π into a model σ_π by function $P2M$. However, for clarity and simplicity, we here formally define functions FSC and fsc. Let $\pi=\langle R, R_{0e}, R_1, R_{1e},\dots\rangle$ be an infinite path in the NFG G of a formula R, and $\pi^{(k)}=\langle R_k, R_{ke},\dots\rangle$ be the kth suffix of π. Whether or not FSC_Property of R is satisfiable over π, denoted by function $\mathit{FSC}(\pi, R)$, FSC (also fsc) : $\Sigma\times\Pi_{pptl}\longrightarrow\{\mathit{true},\mathit{false}\}$, can be defined based on $\mathit{fsc}(\pi, R)$ as follows, where Π_{pptl} stands for all PPTL formulas while Σ denotes all paths in NFGs of all formulas of Π_{pptl}.

Definition 4. $fsc(\pi, R) = true$ if

1. R is not a chop formula, or
2. $R \equiv P;Q$ and there exists $i \in N_0$ such that $R_i \equiv P_{ie} \wedge \varepsilon;Q$, or
3. $R \equiv P \wedge Q$ and $fsc(\pi, P) = true$ and $fsc(\pi, Q) = true$, or
4. $R \equiv P \vee Q$ and $fsc(\pi, P) = true$ or $fsc(\pi, Q) = true$.

Accordingly, $FSC(\pi, R) = true$ if $fsc(\pi^{(k)}, R_k) = true$ for all $k \in N_0$.

In particular, notice that $fsc(\pi, \neg(P;Q)) = true$ since $\neg(P;Q)$ is not a chop formula.

The functions FSC and fsc facilitate us to check whether or not a node which is a chop formula in particular involved in a circle in a path of NFG of a formula satisfies FSC_property. As a matter of fact, when a chop formula Q_i is involved in a circle in a path π in NFG G of Q the FSC_property of Q_i can be checked by means of $fsc(\pi, G)$ and the whole path, i.e. all nodes of the path, can be checked by means of $FSC(\pi, G)$. Although for any chop formula $X;Y$ involved in a circle we consider its FSC_property, however, for the negation of a chop formula like $\neg(X;Y)$ involved in a circle in a path we do not need to consider its finiteness property since an infinite path containing $\neg X$ could generate an infinite model of $\neg(X;Y)$ (see Lemma 4). This case can happen when we construct the NFG of a formula such as $\neg(P;Q;R)$.

Lemma 4. *Let π be an infinite path in the NFG G of a formula Q with $FSC(\pi, Q) = true$, and $R \equiv \neg(X;Y)$ be the ith node in π. For the infinite model $\sigma_\pi = P2M(\pi, G)$, $\sigma_\pi^{(i)} \models \neg(X;Y)$ iff $\forall k(i \leqslant k \in N_0 \rightarrow (\sigma_\pi^k \models \neg X \vee \sigma_\pi^{(k)} \models \neg Y)$.*

Proof. Since σ_π be an infinite model, and

$$\sigma_\pi^{(i)} \models X;Y \quad \Leftrightarrow \quad \exists k i \leqslant k \in N_0 \text{ and } \sigma_\pi^k \models X \text{ and } \sigma_\pi^{(k)} \models Y$$

We have,

$$\begin{aligned}
& \sigma_\pi^{(i)} \models \neg(X;Y) \\
\Leftrightarrow\ & \neg\big(\sigma_\pi^{(i)} \models (X;Y)\big) \\
\Leftrightarrow\ & \neg\big(\exists k i \leqslant k \in N_0 \wedge \sigma_\pi^k \models X \wedge \sigma_\pi^{(k)} \models Y\big) \\
\Leftrightarrow\ & \forall k\big(\neg(i \leqslant k \in N_0) \vee \sigma^k \models \neg X \vee \sigma^{(k)} \models \neg Y\big) \\
\Leftrightarrow\ & \forall k\big(i \leqslant k \in N_0 \rightarrow \big(\sigma^k \models \neg X \vee \sigma^{(k)} \models \neg Y\big)\big) \qquad \square
\end{aligned}$$

Note that, in Lemma 4, if $k = \omega$ the implication of the right hand side is trivially satisfied. This means that an infinite model $\sigma^\omega \models \neg X$ might hold.

For projection construct, $(P_1, \ldots, P_m)\, prj\, Q$, it is eventually treated as a chop formula when constructing NFGs according to the following transforming rule. Suppose

$$P_1 \equiv P_{1e} \wedge \varepsilon \vee \bigvee_{i=1}^{r} (P_{1i} \wedge \bigcirc P'_{1i})$$

$$P_2 \equiv P_{2e} \wedge \varepsilon \vee \bigvee_{l=1}^{s} (P_{2l} \wedge \bigcirc P'_{2l})$$

$$Q \equiv Q_e \wedge \varepsilon \vee \bigvee_{j=1}^{k} (Q_j \wedge \bigcirc Q'_j)$$

then,

$$\begin{aligned}
(P_1, P_2)\, \text{prj}\, Q \equiv\ & Q_e \wedge P_{1e} \wedge P_{2e} \wedge \varepsilon \\
& \vee \bigvee_{l=1}^{s} (Q_e \wedge P_{1e} \wedge P_{2l} \wedge \bigcirc P'_{2l}) \\
& \vee \bigvee_{j=1}^{k} (P_{1e} \wedge P_{2e} \wedge Q_j \wedge \bigcirc Q'_j)
\end{aligned}$$

$$\vee \bigvee_{j=1}^{k} \bigvee_{l}^{s} \big(P_{1e} \wedge Q_j \wedge P_{2l} \wedge \bigcirc\big(P'_{2l}\,;Q'_j\big)\big)$$
$$\vee \bigvee_{i=1}^{r} \big(Q_e \wedge P_{1i} \wedge \bigcirc\big(P'_{1i}\,;P_2\big)\big)$$
$$\vee \bigvee_{i=1}^{r} \bigvee_{j}^{k} \big(Q_j \wedge P_{1i} \wedge \bigcirc\big(P'_{1i}\,;\big(P_2\ prj\ Q'_j\big)\big)\big)$$

The above explanation shows that the finiteness property for projection constructs and $\neg(P\,;Q)$ needs not to be considered, and only chop formulas require to be treated in a special way. To do so, we need the fixed point theory and Scott's fixed point induction.

Fixed point theory. Every monotonic function F over a complete lattice $\langle A, \subseteq\rangle$ has a unique least fixed point $\bigcup_i F^i(\bot)$ and a unique greatest fixed point $\bigcap_i F^i(\top)$ [7]. □

Scott's fixed-point induction. Suppose D is a complete lattice with bottom $\bot$, $F : D \to D$ a continuous function, and B an inclusion subset of D. If $\bot \in B$ and $\forall x \in D.x \in B \Rightarrow F(x) \in B$, then $\mathit{fix}(F) \in B$ [7]. □

The inclusion subset is defined as follows:

Definition 5 *(Inclusion subset).* Let D be a complete partial order. A subset B of D is inclusive iff for all ω-chains $d_0 \sqsubseteq d_1 \cdots \sqsubseteq d_n \sqsubseteq \cdots$ in D if $d_n \in B$ for all $n \in N_0$ then $\bigsqcup_{n\in N_0} d_n \in B$. □

We now turn to prove some conclusions about the relationship between paths in the NFG and models of a formula Q.

Lemma 5. *For an infinite model σ and NFG G of a formula Q, if $\pi_\sigma = M2P(\sigma, G)$ then π_σ with $FSC(\pi_\sigma, Q) = true$ can be found in G.*

Proof. Let $\sigma = \langle s_0, s_1, \ldots\rangle$ and $\pi_\sigma = \langle Q, Q_{0e}, Q_1, Q_{1e}, \ldots\rangle$. We first prove that π_σ can be found in G. By the construction of π_σ, for $(\sigma, 0, |\sigma|) \models Q$, there must exist a normal form of Q, $Q \equiv (Q_{e0} \wedge \varepsilon) \vee \bigvee_{j=0}^{k}(Q_{0j} \wedge \bigcirc Q'_{1j})$ such that there exists an edge from root Q to node $Q'_{1t} \equiv Q_1$ and labeled by $Q_{0t} \equiv Q_{0e}$ for some $0 \leqslant t \leqslant k$ in G and $(\sigma, 0, |\sigma|) \models Q_{0t} \wedge \bigcirc Q'_{1t}$. Here $Q_{0t} \equiv \bigwedge_i \dot{q}_i$, and $\dot{q}_i$ is an atomic proposition q_i or $\neg q_i$. Thus, $\langle s_0\rangle \models Q_{0t}$, and $(\sigma, 1, |\sigma|) \models Q'_{1t}$. That is, $s_0[q_i] = \mathit{true}$ if $\dot{q}_i$ is q_i, and $s_0[q_i] = \mathit{false}$ if $\dot{q}_i$ is $\neg q_i$. So we have found a prefix $\pi_\sigma^0 = \langle Q, Q_{0e}, Q_1\rangle = \langle Q, Q_{0t}, Q'_{1t}\rangle$ of π_σ in G w.r.t. $\langle s_0\rangle$.

Suppose we have found a prefix $\pi_\sigma^i = \langle Q, Q_{0e}, Q_1, \ldots, Q_{ie}, Q_{i+1}\rangle$ of π_σ w.r.t. the prefix $\langle s_0, \ldots, s_i\rangle$ of model σ such that, for $0 \leqslant k \leqslant i$, $(\sigma, k, |\sigma|) \models Q_{ke} \wedge \bigcirc Q_{k+1}$; here $Q_{ke} \equiv \bigwedge_j \dot{p}_{kj}$, and $\dot{p}_{kj}$ is an atomic proposition p_{kj} or $\neg p_{kj}$, and $\langle s_k\rangle \models Q_{ke}$, and $(\sigma, k+1, |\sigma|) \models Q_{k+1}$. That is, $s_k[p_{kj}] = \mathit{true}$ if $\dot{p}_{kj}$ is p_{kj}, and $s_k[p_{kj}] = \mathit{false}$ if $\dot{p}_{kj}$ is $\neg p_{kj}$.

At this point, we can rewrite Q_{i+1} to its normal form, $Q_{i+1} \equiv (Q_{e(i+1)} \wedge \varepsilon) \vee \bigvee_{j=0}^{k}(Q_{(i+1)j} \wedge \bigcirc Q'_{(i+2)j})$, such that there exists an edge from Q_{i+1} to $Q'_{(i+2)t} \equiv Q_{i+2}$ and labeled by $Q_{(i+1)t} \equiv Q_{i+1e}$ in G for some $0 \leqslant t \leqslant k$ and $(\sigma, i+1, |\sigma|) \models Q_{(i+1)e} \wedge \bigcirc Q_{(i+2)}$. Here $Q_{(i+1)e} \equiv \bigwedge_l \dot{r}_l$, and $\dot{r}_l$ is an atomic proposition r_l or $\neg r_l$. Thus, $\langle s_{i+1}\rangle \models Q_{(i+1)e}$ and $(\sigma, (i+2), |\sigma|) \models Q_{(i+2)}$. That is, $s_{i+1}[r_l] = \mathit{true}$ if $\dot{r}_l$ is r_l, and $s_{i+1}[r_l] = \mathit{false}$ if $\dot{r}_l$ is $\neg r_l$. So we have found a prefix $\pi_\sigma^{i+1} = \langle Q, Q_{0e}, Q_1, Q_{1e}, Q_2, \ldots Q_{ie}, Q_i, Q_{(i+1)e}, Q_{(i+2)}\rangle$ of π_σ w.r.t. the prefix $\langle s_0, \ldots, s_i, s_{i+1}\rangle$ of model σ.

Now we need to prove $\pi_\sigma = \langle Q, Q_{0e}, Q_1, Q_{1e}, Q_2, \ldots, Q_{ie}, Q_{i+1}, Q_{(i+1)e}, Q_{(i+2)}, \ldots\rangle$ is an infinite path in the NFG G of Q w.r.t. σ. To do so, we define, $\pi_\sigma^{-1} = \epsilon$, $\pi_\sigma^0 = \langle Q, Q_{0e}, Q_1\rangle$, $\pi_\sigma^i = \langle Q, Q_{0e}, Q_1\rangle, \ldots, \langle Q_{(i)}, Q_{ie}, Q_{(i+1)}\rangle$ ($i \in N_0$). Then we define a set $D = \{\pi_\sigma^{-1}, \pi_\sigma^0, \ldots, \pi_\sigma^i, \ldots\} \cup \{\pi_\sigma\}$. It is readily to prove that $(D, \subseteq)$ is a complete partial order set with bottom $\pi_\sigma^{-1} = \epsilon$, so $(D, \subseteq)$ is a complete lattice. We further define a function $\mathcal{R}$ over D as follows:

$$\mathcal{R}\big(\pi_\sigma^k\big) = \pi_\sigma^{k+1}, \quad \text{for all } k \in \{-1\} \cup N_0$$

It is easily to prove that, for $\pi_\sigma^i, \pi_\sigma^j \in D$, $\pi_\sigma^i \subseteq \pi_\sigma^j$ iff $i \leqslant j$. Furthermore,

$$\mathcal{R}\big(\pi_\sigma^i\big) = \pi_{\sigma s}^{i+1} \qquad \mathcal{R}\big(\pi_\sigma^j\big) = \pi_{\sigma s}^{j+1}$$

Since $i \leqslant j$, so $i+1 \leqslant j+1$. Thus, we have $\pi_\sigma^{i+1} \subseteq \pi_\sigma^{j+1}$. Hence $\mathcal{R}$ is monotonic. By Tarsky's fixed point theorem,

$$\mathit{fix}(\mathcal{R}) = \bigcup_i \mathcal{R}^i\big(\pi_\sigma^{-1}\big) = \pi_\sigma$$

Further, we have

$$\mathcal{R}\Big(\bigcup_{i\in N_0}\pi_\sigma^i\Big)=\pi_\sigma=\bigcup_{i\in N_0}\mathcal{R}\big(\pi_\sigma^i\big)$$

So, $\mathcal{R}$ is also continuous. We can now construct a subset B of D as follows:

$$B=\big\{\pi_\sigma^i \bigm| \pi_\sigma^i\in D \text{ and } \pi_\sigma^i \text{ is the } i\text{th prefix of a path } \pi_1 \text{ of } G\big\}$$

In the previous step, we have shown that, for $i\in N_0$, ith prefix π_σ^i of π_σ is ith prefix of a path in G. Hence, we can reasonably assume that this path in G is π_1. So, for any ω-chain $\pi_\sigma^0\sqsubseteq\pi_\sigma^1\sqsubseteq\cdots\sqsubseteq\pi_\sigma^i\sqsubseteq\cdots$ in D, this ω-chain is in fact in B since $\pi_\sigma^i\in B$. Moreover, we have $\bigcup_{i\in N_0}\pi_\sigma^i=\pi_\sigma\in B$, namely, $\bigsqcup_{i\in N_0}\pi_{\sigma s}^i\in B$. Therefore, B is an inclusion subset of D.

Since $\pi_\sigma^i=\pi_1^i$ for $i\in N_0$, so, $B=\{\epsilon\}\cup\{\pi_1^i\mid\pi_1^i$ is the ith prefix of $\pi_1\}\cup\{\pi_\sigma\}=D$. Further, F is a monotonic and continuous function over D. Hence, F is also a monotonic and continuous function over B with $F(\pi_1^i)=\pi_1^{i+1}$ for $i\in N_0$. Now by using Tarsky's theorem, we obtain $\mathit{fix}(F)=\pi_1$. Further, $\epsilon\in B$, by using Scott's Fix-Point Induction, $\mathit{fix}(F)=\pi_1\in B$. However, B has only one element π_σ with infinite length. It turns out that $\pi_\sigma=\pi_1$ is an infinite path of G.

Now we prove $FSC(\pi_\sigma,Q)=\mathit{true}$. This can equivalently be proved in terms of $\mathit{fsc}(\pi_\sigma^{(i)},Q_i)=\mathit{true}$, for all $i\geqslant 0$. Let Q_i be an arbitrary node in π_σ. We need only to prove that for any chop component $P;V$ appearing in Q_i, $\sigma^{(i)}\models P;V$. That is, $\exists j\in N_0$, $j\geqslant i$, $\sigma_{(i..j)}\models P$ and $\sigma_{(j..|\sigma|)}\models V$. As the reduction proceeds, a final state of P can be reached, i.e. $P_{e(j-i)}\wedge\varepsilon;V$ can be encountered, where $P_{e(i-j)}$ is a state formula, FSC_Property of $P;V$ can be satisfied over the path π_σ. Hence, at state s_j over path π_σ $Q_j\equiv P_{(ej-i)}\wedge\varepsilon;V$. By Definition 4, $\mathit{fsc}(\pi_\sigma^{(i)},Q_i)=\mathit{true}$. It turns out that $FSC(\pi_\sigma,Q)=\mathit{true}$.

Lemma 6. *For an infinite path π in the NFG G of Q, if there exist no chop formulas over π, $\sigma_\pi=P2M(\pi,G)\models Q$.*

Proof. Let $\pi=\langle Q,Q_{0e},Q_1,Q_{1e},\ldots,Q_i,Q_{ie},\ldots\rangle$ be an infinite path in the NFG G of Q. Then we can construct an interval $\sigma_\pi=\langle s_0,s_1,\ldots,s_i,\ldots\rangle$ by function $P2M(\pi,G)$. For each $i\in N_0$, if proposition q in Q_{ie} then $s_i[q]=\mathit{true}$, otherwise if $\neg q$ in Q_{ie} then $s_i[q]=\mathit{false}$. We need to prove σ_π is a model of formula Q. To do so, we need first to prove that a prefix σ_π^i of σ_π is a prefix of a model. The proof proceeds by induction on the length of the prefix of the interval.

Base: $\sigma_\pi^0=\langle s_0\rangle$. By construction of s_0, $\langle s_0\rangle\models Q_{0e}$.

Induction: Suppose for all $j<i$, prefix σ_π^j of σ_π is a prefix of a model. We prove σ_π^i is also a prefix of the model. By hypothesis, $\sigma_\pi^{i-1}=\langle s_0,\ldots,s_{i-1}\rangle$ is a prefix of the model. According to the construction of NFG, there must be $Q_{i-1}\equiv Q_{ie}\wedge\bigcirc Q_i$ such that a new node Q_i and a new edge (Q_{i-1},Q_{ie},Q_i) are added to G. By the construction of s_i, $s_i\models Q_{ie}$. Since σ_π^{i-1} is a prefix of the model, σ_π^i is also a prefix of the model. We now need to prove that σ_π is a model of Q.

Let $\sigma_\pi^{-1}=\epsilon$ and $\sigma_\pi^i=\langle s_0,s_1,\ldots,s_i\rangle$ for $i\in N_0$. Then we defined a set $A=\{\sigma_\pi^{-1},\ldots,\sigma_\pi^i,\ldots\}\cup\{\sigma_\pi\}$. It is readily to prove that $(A,\subseteq)$ is a complete partial order set with bottom $\sigma_{\pi s}^{-1}=\epsilon$, so $(A,\subseteq)$ is a complete lattice. We define a function $\mathcal{R}$ over A as follows:

$$\mathcal{R}\big(\sigma_\pi^k\big)=\sigma_\pi^{k+1},\quad k\geqslant -1$$

So, for all σ_π^i, $\sigma_\pi^j\in A$, $i\leqslant j$, we have $\sigma_\pi^i\subseteq\sigma_\pi^j$, and

$$\mathcal{R}\big(\sigma_\pi^i\big)=\sigma_\pi^{i+1}\qquad\mathcal{R}\big(\sigma_\pi^j\big)=\sigma_\pi^{j+1}$$

Since $i\leqslant j$, so $i+1\leqslant j+1$. Then we have $\sigma_\pi^{i+1}\subseteq\sigma_\pi^{j+1}$. So, $\mathcal{R}$ is monotonic. Thus, by fixed point theorem,

$$\mathit{fix}(\mathcal{R})=\bigcup_i\mathcal{R}^i\big(\sigma_\pi^{-1}\big)=\sigma_\pi$$

Further, we have

$$\mathcal{R}\Big(\bigcup_{i\in N_0}\sigma_\pi^i\Big)=\bigcup_{i\in N_0}\mathcal{R}\big(\sigma_\pi^i\big)$$

So $\mathcal{R}$ is also continuous. We can now construct a subset B of A as follows:

$$B=\{\epsilon\}\cup\big\{\sigma_\pi^i\bigm|\sigma_\pi^i\in A \text{ and } \sigma_\pi^i \text{ is the } i\text{th prefix of a model } \sigma_1 \text{ of } Q\big\}$$

In the previous step, we have shown that the ith prefix σ_π^i of σ_π is the ith prefix of a model of Q. Hence, we can assume that this model is σ_1. Further, for any ω-chain $\sigma_\pi^0\sqsubseteq\sigma_\pi^1\sqsubseteq\cdots\sqsubseteq\sigma_\pi^i\sqsubseteq\cdots$ in A, this ω-chain is in fact in B since $\sigma_\pi^i\in B$. Moreover, we have $\bigcup_{i\in N_0}\sigma_\pi^i=\sigma_\pi\in B$, namely, $\bigsqcup_{i\in N_0}\sigma_\pi^i\in B$. Therefore, B is an inclusion subset of A.

Since $\sigma_\pi^i=\sigma_1^i$ for $i\in N_0$, so, $B=\{\epsilon\}\cup\{\sigma_1^i|\sigma_1^i$ is the ith prefix of $\sigma_1\}\cup\{\sigma_\pi\}=A$. Further, F is a monotonic and continuous function over A. Hence, F is also a monotonic and continuous function over B with $F(\sigma_1^i)=\sigma_1^{i+1}$ for $i\in N_0$. Now by using Tarsky's theorem, we obtain $fix(F) = \sigma_1$. Further, $\epsilon \in B$, by using Scott's Fix-Point Induction, $fix(F) = \sigma_1 \in B$. However, B has only one element σ_π with infinite length. It turns out that $\sigma_\pi = \sigma_1$ is an infinite model of Q. $\square$

In the following, we further discuss the general case of path π for a given formula R. The final conclusion is presented in Theorem 10.

Lemma 7. *For an infinite path $\pi = \langle R, R_{e0}, R_1, R_{e1}, R_2, \ldots\rangle$ in the NFG G of formula R, if $\sigma_\pi^{(i)} \models R_i$, then $\sigma_\pi^{(i-1)} \models R_{i-1}$.*

Proof. By function $P2M(\pi, G)$, we can obtain an interval $\sigma_\pi = \langle s_0, s_1, \ldots\rangle$. Thus, for $i > 0$, $\langle s_{i-1}\rangle \models R_{ei-1}$. Hence, $\langle s_{i-1}, s_i\rangle \models R_{ei-1} \wedge skip$, and $\sigma_\pi^{(i-1)} = \langle s_{i-1}, s_i\rangle \circ \sigma_\pi^{(i)} \models R_{ei-1} \wedge skip ; R_i$. So, $\sigma_\pi^{(i-1)} \models R_{ei-1} \wedge (skip ; R_i)$. That is, $\sigma_\pi^{(i-1)} \models R_{ei-1} \wedge \bigcirc R_i$. Further, by the construction of NFG, $R_{i-1} \equiv R_{ei-1} \wedge \bigcirc R_i \vee X$, X being a PPTL formula in its normal form. So, $\models R_{e\ i-1} \wedge \bigcirc R_i \rightarrow R_{i-1}$, leading to $\sigma_\pi^{(i-1)} \models R_{i-1}$. $\square$

Corollary 8. *For an infinite path $\pi = \langle R, R_{e0}, R_1, R_{e1}, R_2, \ldots\rangle$ in the NFG of formula R, $\sigma_\pi \models R$ if there exists $i \in N_0$ such that $\sigma_\pi^{(i)} \models R_i$.*

Lemma 9. *If π is an infinite path in the NFG G of formula R with $FSC(\pi, R) = true$, and $\sigma_\pi = P2M(\pi, G)$, then $\sigma_\pi \models R$.*

Proof. Let $\pi = \langle R_0, R_{0e}, R_1, R_{1e}, \ldots\rangle$, where $R_0 \equiv R$, and $Nod_\pi = \{R_0, R_1, R_2, \ldots\}$ be the set of all nodes of π, and $CNod_\pi = \{R_l \mid R_l \in Nod_\pi$ and R_l is a chop formula $\}$, and $CCop_\pi = \{P_l ; Q_l \mid R_l \in CNod_\pi$ and $R_l \equiv (P_l ; Q_l) \wedge R'\}$ be the set of all chop components over π. The proof proceeds by induction on the number of chop components over π.

Base: there are no any chop components over π. This implies $CNod_\pi = \emptyset$. By Lemma 6, $\sigma_\pi \models R$.

Induction: Suppose for any path π, $\mid CCop_\pi \mid \leqslant k \in N_0$, $\sigma_\pi \models R$. When $\mid CCop_\pi \mid = k+1$, let R_l be the first chop formula over π. By the definition of chop formulas, $R_l = (P_l ; Q_l) \wedge R'$. Since $FSC(\pi, R) = true$, we have $fsc(\pi^{(l)}, R_l) = true$. That is, there exists $i_l \in N_0$ such that $R_{i_l} = P_{i_l e} \wedge \varepsilon ; Q_l \equiv P_{i_l e} \wedge Q_l$, $i_l - l \geqslant 1$. This means one chop component is eliminated in at least one reduction step. Therefore, $|CCop_{\pi^{(i_l)}}| \leqslant k$, according to the hypothesis, $\sigma_{\pi^{(i_l)}} \models R_{i_l}$. Further, by Lemma 7 and Corollary 8, $\sigma_\pi \models R$. $\square$

Theorem 10. *The set of infinite path π with $FSC(\pi, R) = true$ in the NFG G of PPTL formula R precisely characterizes the set of infinite models of R.*

Proof. The theorem is a direct consequence of Lemmas 5 and 9. $\square$

4. Decision procedure based on LNFG

To explicitly display whether or not the FSC_Property of a chop formula is satisfied, extra propositions l_k, $k \in N_0$ and $k > 0$, are introduced. Let $Prop_l = \{\mathsf{l}_1, \mathsf{l}_2, \ldots\}$ be the set of extra propositions with $Prop \cap Prop_l = \emptyset$. Note that these extra propositions are merely employed to mark nodes and are not allowed to appear in a PPTL formula. When constructing NFGs by normal form reductions, for any chop formula $P ; Q$, we equivalently rewrite it as $P \wedge fin(\mathsf{l}_k)|CHOPQ$. Here, $fin(P)$ is defined by,

$$fin(P) \equiv \square(\varepsilon \rightarrow P)$$

Equivalently,

$$fin(P) \equiv P \wedge \varepsilon \vee \bigcirc fin(P)$$

Thus, we have,

$$\begin{aligned}
& P \wedge fin(\mathsf{l}_k) ; Q \\
\equiv & \left(\bigvee_{j=0}^{n_0} P_{ej} \wedge \varepsilon \vee \bigvee_{i=0}^{n_1} \left(P_{ci} \wedge \bigcirc P'_i\right)\right) \wedge \left(\mathsf{l}_k \wedge \varepsilon \vee \bigcirc fin(\mathsf{l}_k)\right) ; Q \\
\equiv & \left(\bigvee_{j=0}^{n_0} P_{ej} \wedge \mathsf{l}_k \wedge \varepsilon \vee \bigvee_{i=0}^{n_1} \left(P_{ci} \wedge \bigcirc\left(P'_i \wedge fin(\mathsf{l}_k)\right)\right)\right) ; Q \\
\equiv & \bigvee_{j=0}^{n_0} (\underline{P_{ej} \wedge \mathsf{l}_k \wedge \varepsilon ; Q}) \vee \bigvee_{i=0}^{n_1} \left(P_{ci} \wedge \bigcirc\left(P'_i \wedge fin(\mathsf{l}_k) ; Q\right)\right)
\end{aligned}$$

$$\equiv \bigvee_{j=0}^{n_0} \underline{(P_{ej} \wedge \mathsf{l_k} \wedge Q)} \vee \bigvee_{i=0}^{n_1} \big(P_{ci} \wedge \bigcirc\big(P'_i \wedge fin(\mathsf{l_k})\,;Q\big)\big)$$

As a result, by using $fin(\mathsf{l_k})$, FSC_Property of $P\,;Q$ is satisfied if there exists an edge where $\mathsf{l_k}$ holds. Furthermore, $fin(\mathsf{l_k})$ occurring in a node $P \wedge fin(\mathsf{l_k})\,;Q$ means that FSC_Property of $P\,;Q$ has not been satisfied at this node. For convenience, for a node in the form of $P \wedge fin(\mathsf{l_k})\,;Q$ or $\bigwedge_{i=1}^{n} R_i$ with some $R_i \equiv P_i \wedge fin(\mathsf{l_k})\,;Q_i$, we add an extra label $\tilde{\mathsf{l_k}}$ in this node to mean that the finiteness of some chop formula has not been satisfied at this node.

Accordingly, Labeled Normal Form Graph (LNFG) is defined based on NFG by means of $\mathsf{l_k}$ propositions.

Definition 6 *(Labeled Normal Form Graph, LNFG).* For a PPTL formula P, its LNFG is a tuple $G = (CL(P), EL(P), V_0, \mathbb{L} = \{\mathbb{L}_1, \dots, \mathbb{L}_m\})$, where $CL(P)$, $EL(P)$ and V_0 are identical to the ones in the NFG, while each $\mathbb{L}_k \subseteq CL(P)$, $1 \leqslant k \leqslant m$, is the set of nodes with $\tilde{\mathsf{l_k}}$ labels.

Algorithm LNFG based on Algorithm NFG is formalized by further rewriting any chop component $P\,;Q$ as $P \wedge fin(\mathsf{l_k})\,;Q$ for some $k \in N_0$ whenever it is encountered. Basically, for a given PPTL formula P, the number of notes of its LNFG G is at most double of number of notes of NFG G' since, in the worst case, a note of G' can be placed in both $CL(P)$ and $CL'(P)$. The key difference between G and G' is that some extra labels $\mathsf{l_k}$s appear in some edges and $\tilde{\mathsf{l_k}}$s appear in some nodes of G. In the following, we use $P \overset{\text{fin}}{\equiv} Q$ for convenience if $P \equiv Q$ without considering $fin(l_x)$ and l_x labels appearing in P and Q. Note that $P\,;Q$ does not need to be replaced by $P \wedge fin(l_k)\,;Q$ if $P \equiv P' \wedge len(n)$ for $n \in N_0$ since, in this case, the finiteness of P can certainly be satisfied. However, for simplicity, we replace all $P\,;Q$ by $P \wedge fin(l_k)\,;Q$ in Algorithm LNFG.

In the algorithm LNFG, an auxiliary algorithm PRE is employed (see Appendix A for details). The functions of PRE are as follows:

1. replace $P \to Q$, $P \leftrightarrow Q$, by $\neg P \vee Q$ and $P \wedge Q \vee \neg P \wedge \neg Q$ respectively;
2. replace $\varepsilon prj\ \varepsilon$ by ε, $(P_1, \dots, P_m)\ prj\ \varepsilon$ by $(P_1\,; \dots ;P_m)$, $(P_1, \dots, P_t \vee P'_t, \dots, P_m)\ prj\ Q$ by $(P_1, \dots, P_t, \dots, P_m)\ prj\ Q \vee (P_1, \dots P'_t, \dots, P_m)\ prj\ Q$, $(P_1, \dots, P_m)\ prj\ (Q \vee Q')$ by $(P_1, \dots, P_m)\ prj\ Q \vee (P_1, \dots, P_m)\ prj\ Q'$, $(w \wedge P_1, \dots, P_m)\ prj\ Q$ and $(P_1, \dots, P_m)\ prj\ Q \wedge w$ by $w \wedge (P_1, \dots, P_m)\ prj\ Q$, where w is a state formula;
3. replace $\varepsilon\,;P$ by P, and $\varepsilon \wedge w\,;P$ by $w \wedge P$, where w is a state formula;
4. replace $true\,;P$ by $P \vee \bigcirc\Diamond P$;
5. move negation operators to the front of atomic propositions or chop or projection formulas as possible as we can by means of distributive laws on conjunction and disjunction operations;
6. simplify the formula by using logic laws: $\neg\neg P \equiv P$, $P \wedge \neg P \equiv false$ ($\varepsilon \wedge more \equiv false$), $P \vee \neg P \equiv true$ ($\varepsilon \vee more \equiv true$), $P \wedge false \equiv false$, $P \vee false \equiv P$, $P \wedge true \equiv P$, $P \vee true \equiv true$, $false\,;P \equiv false$, $P\,;false \equiv false$, $(p_1, \dots, false, \dots, P_m)\ prj\ Q \equiv false$, $(P_1, \dots, P_m)\ prj\ false \equiv false$;

   Note that rules 2, 3 and 4 enable us to reduce the number of $fin(l_x)$ labels.

Lemma 11. *Let $\pi_c = \langle (R_i, R_{(ei+1)}, \dots, R_j, R_{(ej+1)})^{\omega}\rangle$ be a circle of an infinite path $\pi = \langle R, R_{e0}, \dots, (R_i, R_{(ei+1)}, \dots, R_j, R_{(ej+1)})^{\omega}\rangle$ in the LNFG $G = (CL(R), EL(R), V_0, \mathbb{L} = \{\mathbb{L}_1, \dots, \mathbb{L}_m\})$ of a formula R, and $\mathbb{L}^c_i = \inf(\pi_c) \cap \mathbb{L}_i (1 \leqslant i \leqslant m)$. We have the following conclusions:*

(1) $inf(\pi) = inf(\pi_c)$,
(2) $\mathbb{L}^c_i \subseteq \mathbb{L}_i$,
(3) *$FSC(\pi, R) = true$ iff $FSC(\pi_c, R_i) = true$,*
(4) *$inf(\pi) \nsubseteq \mathbb{L}_i$ iff $inf(\pi_c) \nsubseteq \mathbb{L}^c_i$.*

Proof. The conclusions (1) and (2) are obvious.

The proof of (3): $FSC(\pi, R) = true$ implies trivially $FSC(\pi_c, R_i) = true$ since π_c is a part of π. Conversely, if $FSC(\pi_c, R_i) = true$, then for all nodes of π_c, in particular, the joining node R_i from path π entering into the circle π_c, can be reduced to a node $R_h \wedge l_x \wedge \varepsilon\,;R'_h$ such that $fsc(\pi_c, R_i) = true$. Thus, any node R_k $(k < i)$ in the path π is in fact out of π_c. As a matter of fact, either R_k can be reduced to a node $R_l \wedge l_y \wedge \varepsilon\,;R'_l$ out of π_c or R_k can be reduced to R_i which can be further reduced to node $R_h \wedge l_x \wedge \varepsilon\,;R'_h$. Hence, in both cases, $fsc(\pi^{(k)}, R_k) = true$, leading to $FSC(\pi, R) = true$.

The proof of (4): First of all, note that any node R_x in π but out of π_c is not identical to any node R_y in π_c. Hence, any node $R_x \notin \mathbb{L}^c_i$ in π_c implies $R_x \notin \mathbb{L}_i$, and any node $R_y \notin \mathbb{L}_i$ in π_c implies $R_x \notin \mathbb{L}^c_i$. Therefore, $inf(\pi) \nsubseteq \mathbb{L}_i$ iff $inf(\pi_c) \nsubseteq \mathbb{L}^c_i$. □

Lemma 12. *For a circle $\pi_c = \langle (R_i, R_{(ei+1)}, \dots, R_j, R_{(e\ j+1)})^{\omega}\rangle$ of an infinite path π in the LNFG G of a formula R, if $FSC(\pi_c, R_i) = true$, then for any label $\mathsf{l_k} \in Prop_l$, not all of nodes $R_h \in inf(\pi_c)$ are marked with $\tilde{\mathsf{l_k}}$.*

Proof. Suppose the conclusion is false. Then there is a label $\mathsf{l_k} \in Prop_l$, all of nodes $R_h \in inf(\pi_c)$ are marked with $\tilde{\mathsf{l_k}}$. As a result, $fsc(\pi_c^{(h)}, R_h) = false$ since there is no any node in the circle can be reduced to the form $R_{h+m} \wedge \mathsf{l_k} \wedge \varepsilon\,;R'_h$. This contradicts with $FSC(\pi_c, R_i) = true$. □

Algorithm LNFG: Constructing LNFG of a PPTL formula.

Function LNFG(P)
/* precondition: $P \equiv \bigvee_i P_i$ $(0 \leqslant i)$ is a PPTL formula */
/* postcondition: LNFG of P, $G = (CL(P), EL(P), V_0, \mathbb{L} = \{\mathbb{L}_1, \ldots, \mathbb{L}_m\})$ */
begin function
$V_0 = CL(P) = \{P_i \mid P_i$ appears in $\bigvee_i P_i\}$; $Mark[P_i] = 0$ for each i;
$EL(P) = \emptyset$; $\mathsf{AddE} = \mathsf{AddN} = 0$; $k = 1$; $\mathbb{L} = \emptyset$; $CL'(P) = \emptyset$; /* initialization */
while $\exists R \in (CL(P) \cup CL'(P)) \backslash \{\varepsilon\}$ and $mark[R] == 0$
$R = PRE(R)$;
if $R \not\equiv false$ **then** /* R is false if $r \wedge \neg r$ appears in it */
if $R = (\bigwedge_{i=1}^{n} R_i) \wedge Z$ with $R_i = P_i; Q_i$ and Z is not a chop formula **then**
for $i = 1$ to n /* adding labels to chop formulas */
if R_i has been rewritten with a $\mathrm{fin}(l_j)$ $(1 \leqslant j < k)$
then $\mathbb{L}_j = \mathbb{L}_j \cup \{R\}$/* incorporating R to corresponding $\mathbb{L}_j$ */
else Rewrite R_i as $P_i \wedge \mathrm{fin}(l_k)$; Q_i; $\mathbb{L}_k = \mathbb{L}_k \cup \{R\}$; $k = k + 1$;
end for
end if
$Q = NF(R)$; $mark[R] = 1$; /* rewriting R into its normal form */
case /* deciding whether future products or terminal products are contained in NF */
Q is $\bigvee_{t=1}^{m} Q_{et} \wedge \varepsilon$: $\mathsf{AddE} = 1$;
Q is $\bigvee_{j=1}^{h} (Q_{cj} \wedge \bigcirc Q_{fj})$: $\mathsf{AddN} = 1$;
Q is $\bigvee_{t=1}^{m} Q_{et} \wedge \varepsilon \vee \bigvee_{j=1}^{h} (Q_{cj} \wedge \bigcirc Q_{fj})$: $\mathsf{AddE} = \mathsf{AddN} = 1$;
end case
if $\mathsf{AddE} == 1$ **then**
for $t = 1$ **to** m /* dealing with each Q_{et} */
if $Q_{et} \not\equiv false$ **then**
if $\varepsilon \notin CL(P)$ **then** $CL(P) = CL(P) \cup \{\varepsilon\}$; **end if**
$EL(P) = EL(P) \cup (R, Q_e, \varepsilon)$;
end if
end for
$\mathsf{AddE} = 0$;
end if
if $\mathsf{AddN} == 1$ **then**
for $j = 1$ **to** h /* dealing with each Q_{cj} */
if $Q_{cj} \not\equiv \mathsf{false}$ **then**
if $Q_{fj} \notin CL(P)$ **then**
if $\exists P_x \in CL(P)$ such that $Q_{fj} \overset{\mathrm{fin}}{=} P_x$ **then**
if $\exists P_s \in CL'(P)$ such that $Q_{fj} \overset{\mathrm{fin}}{=} P_s$ **then** $EL(P) = EL(P) \cup \{(R, Q_{cj}, P_x)\}$
/* a node added with third $fin(l_x)$ is forced to point to the first one in $CL(P)$ */
else $CL'(P) = CL'(P) \cup \{Q_{fj}\}$; $EL(P) = EL(P) \cup \{(R, Q_{cj}, Q_{fj})\}$; $mark[Q_{fj}] = 0$;
/* a node added with second $fin(l_x)$ is placed in $CL'(P)$ */
end if
else $CL(P) = CL(P) \cup \{Q_{fj}\}$; $EL(P) = EL(P) \cup \{(R, Q_{cj}, Q_{fj})\}$; $mark[Q_{fj}] = 0$;
/* a node added with first $fin(l_x)$ is placed in $CL(P)$ */
end if
else $EL(P) = EL(P) \cup \{(R, Q_{cj}, Q_{fj})\}$
end if
end for
$\mathsf{AddN} = 0$;
end if
end if
end while
$\mathbb{L} = \bigcup_{i=1}^{k-1} \{\mathbb{L}_i\}$; $CL(P) = CL(P) \cup CL'(P)$;
while $\exists R \in CL(P)$, such that R is not ε and has no edges departing from
$CL(P) = CL(P) \backslash \{R\}$; $EL(P) = EL(P) \backslash \bigcup_i (R_i, R_c, R)$; $\mathbb{L} = \{\mathbb{L}_1 \backslash \{R\}, \ldots, \mathbb{L}_m \backslash \{R\}\}$;
end while
return $G = (CL(P), EL(P), V_0, \mathbb{L})$;
end function

Corollary 13. *For a circle $\pi_c = \langle (R_i, R_{(ei+1)}, \ldots, R_j, R_{(ej+1)})^{\omega} \rangle$, with at least one label $\tilde{\mathsf{l}}_\mathsf{k}$ ($\mathsf{l}_\mathsf{k} \in Prop_l$), of an infinite path π in the LNFG G of a formula R, if $FSC(\pi_c, R_i) = true$, then $|inf(\pi_c)| \geqslant 2$.*

Lemma 14. *For an infinite path π in the LNFG, $G = (CL(R), EL(R), V_0, \mathbb{L} = \{\mathbb{L}_1, \ldots, \mathbb{L}_m\})$, of a PPTL formula R, $FSC(\pi, R) = true$ iff $\mathsf{Inf}(\pi) \nsubseteq \mathbb{L}_i$ for all i, $1 \leqslant i \leqslant m$.*

Proof. Since π is an infinite path in the LNFG G of a formula R, there is a circle $\pi_c = \langle (R_i, R_{(e\ i+1)}, \ldots, R_j, R_{(ej+1)})^{\omega} \rangle$ of π. By Lemma 11, we have, $FSC(\pi, R) = true$ iff $FSC(\pi_c, R_i) = true$, and $inf(\pi) \nsubseteq \mathbb{L}_i$ iff $inf(\pi_c) \nsubseteq \mathbb{L}_i^c$. We need only to prove $FSC(\pi_c, R_i) = true$ iff $inf(\pi_c) \nsubseteq \mathbb{L}_k^c$ for all $i \leqslant k \leqslant j$.

Algorithm CHECK: Checking whether or not P is satisfiable.

Function CHECK(P)
/* precondition: P is a PPTL formula */
/* postcondition: CHECK(P) checks whether formula P is satisfiable or not. */
begin function
  G =LNFG(P);
  if there exists ε node in $CL(P)$,
  return P is satisfiable with finite models;
  if there exists infinite path π with $\mathrm{Inf}(\pi) \not\subseteq \mathbb{L}_i$, for all $1 \leqslant i \leqslant m$
  return P is satisfiable with infinite models;
  else return unsatisfiable;
end function

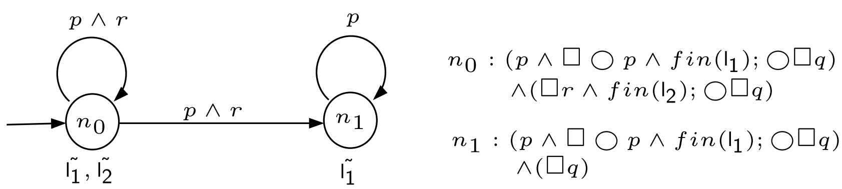

Fig. 1. LNFG of $(p \wedge \Box \bigcirc p; \bigcirc \Box q) \wedge (\Box r; \bigcirc \Box q)$.

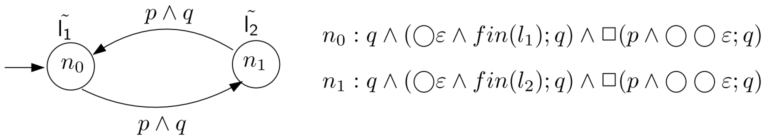

Fig. 2. LNFG of $q \wedge (\bigcirc\varepsilon; q) \wedge \Box(p \wedge \bigcirc\bigcirc \varepsilon; q)$.

$\Rightarrow$: Case 1: Suppose for all $R_h \in inf(\pi_c)$, R_h has no label $\tilde{l_k}$. In this case, $fsc(\pi_c^{(h)}, R_h) = true (i \leqslant h \leqslant j)$ and $\mathbb{L}_k^c = \emptyset$ leading to $inf(\pi_c) \not\subseteq \mathbb{L}_k^c$ for all $1 \leqslant k \leqslant m$.

Case 2: Suppose $m \geqslant 1$ labels $\tilde{l_1}, \tilde{l_2}, \ldots \tilde{l_m}$ are used in π_c. By Lemma 12, for all i $(1 \leqslant i \leqslant m)$, at least one node $R_h \in inf(\pi_c)$ is not marked by $\tilde{l_i}$. Thus, $R_h \notin \mathbb{L}_i^c$, leading to $inf(\pi_c) \not\subseteq \mathbb{L}_i^c$ for all $1 \leqslant i \leqslant m$.

$\Leftarrow$: Suppose $inf(\pi_c) \not\subseteq \mathbb{L}_k^c$ for all $1 \leqslant k \leqslant m$. This guarantees that not all of nodes can be marked by a single label $\tilde{l_k}$. This means that any node $R_h \in inf(\pi_c)$ with label $\tilde{l_k}$ can be reduced to a state $R_h^i \wedge l_k \wedge \varepsilon; R_h'$. Therefore, $fsc(\pi_c^{(h)}, R_h) = true$. Of course, for any node $R_h \in inf(\pi_c)$ without any labels, we also have $fsc(\pi_c^{(h)}, R_h) = true$. Therefore $FSC(\pi_c, R_i) = true$. □

Theorem 15. *In the LNFG G of a formula P, finite paths precisely characterize finite models of P; infinite paths with* $\mathrm{Inf}(\pi) \not\subseteq \mathbb{L}_i$, *for all* $1 \leqslant i \leqslant m$ *precisely characterize infinite models of P.*

Proof. For finite case, it has been proved in Theorem 1. For infinite case, it is a direct consequence of Theorem 10 and Lemma 14. □

Consequently, a decision procedure for checking the satisfiability of a PPTL formula P can be constructed based on the LNFG of P. In the following, a sketch of the procedure, Algorithm CHECK in pseudo code, is demonstrated.

Example 1 *(Checking the satisfiability of formula* $P \equiv (p \wedge \Box \bigcirc p; \bigcirc\Box q) \wedge (\Box r; \bigcirc\Box q)$*).* By Algorithm LNFG, LNFG $G = (CL(P), EL(P), V_0, \mathbb{L} = \{\mathbb{L}_1, \ldots, \mathbb{L}_m\})$ of formula $(p \wedge \Box \bigcirc p; \bigcirc\Box q) \wedge (\Box r; \bigcirc\Box q)$ is constructed as depicted in Fig. 1, where $CL(P) = \{n_0, n_1\}$, $EL(P) = \{(n_0, p \wedge r, n_0), (n_0, p \wedge r, n_1), (n_1, p, n_1)\}$, $V_0 = \{n_0\}$, $\mathbb{L} = \{\mathbb{L}_1, \mathbb{L}_2\}$, $\mathbb{L}_1 = \{n_0, n_1\}$ and $\mathbb{L}_2 = \{n_0\}$. There are two infinite paths in the LNFG G of P: $\pi_1 = \langle (n_0, p \wedge r)^\omega \rangle$ and $\pi_2 = \langle n_0, (p \wedge r, n_0)^k, p \wedge r, (n_1, p)^\omega \rangle$, $0 \leqslant k \in N_0$. As a result, $inf(\pi_1) = \{n_0\} \subseteq \mathbb{L}_2 = \{n_0\} \subseteq \mathbb{L}_1 = \{n_0, n_1\}$ and $inf(\pi_2) = \{n_1\} \subseteq \mathbb{L}_1 = \{n_0, n_1\}$. Hence, the formula P is unsatisfiable. In fact, there exists no node without $\tilde{l_1}$ label which is reachable from n_0 and n_1. □

Example 2 *(Checking the satisfiability of the formula,* $R \equiv q \wedge (\bigcirc\varepsilon; q) \wedge \Box(p \wedge \bigcirc\bigcirc \varepsilon; q)$*).* The new LNFG of formula $R \equiv q \wedge (\bigcirc\varepsilon; q) \wedge \Box(p \wedge \bigcirc\bigcirc \varepsilon; q)$ is illustrated in Fig. 2. As a result, $\mathbb{L}_1 = \{n_0\}$ and $\mathbb{L}_2 = \{n_1\}$. The formula is satisfiable since for the only infinite path $\pi = \langle (n_0, p \wedge q, n_1, p \wedge q)^\omega \rangle$, $inf(\pi) = \{n_0, n_1\} \not\subseteq \mathbb{L}_i$, $i = 1, 2$. □

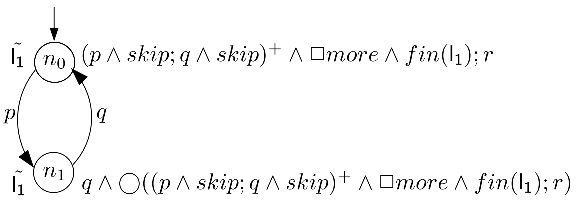

Fig. 3. LNFG of $(p \wedge skip\,; q \wedge skip)^+ \wedge \Box more\,; r$.

Example 3 *(Checking the satisfiability of formula $(p \wedge skip\,; q \wedge skip)^+ \wedge \Box more\,; r$).* The formula is obviously unsatisfiable since $(p \wedge skip\,; q \wedge skip)^+ \wedge \Box more$ cannot be satisfied with finite models. With the new decision procedure, the LNFG is depicted in Fig. 3 (2). As you can see, $\mathbb{L}_1 = \{n_0, n_1\}$, and the only infinite path $\pi = \langle (n_0, p, n_1, q)^\omega \rangle$. This indicates the formula is unsatisfiable since $inf(\pi) = \{n_0, n_1\} \subseteq \mathbb{L}_1$. $\Box$

5. Implementation of the decision procedure

This section focuses on the implementation aspects of the decision procedure. To do so, implementation details of relevant algorithms including pre-processing algorithms, the circle founding algorithm, the strong connected components algorithm i.e. Tarjan algorithm etc. are provided; the generalized Büchi accepting condition for LNFG is formalized. To show how the program works, three examples are presented.

5.1. Programs related issues

It has been proved that infinite paths with $\mathrm{Inf}(\pi) \not\subseteq \mathbb{L}_i$ for all $1 \leqslant i \leqslant m$ precisely characterize infinite models of P. This can be equivalently expressed by "infinite paths with $\mathrm{Inf}(\pi) \cap \overline{\mathbb{L}_i} \neq \emptyset$ for all $1 \leqslant i \leqslant m$ precisely characterize infinite models of P", where $\overline{\mathbb{L}_i}$ denotes $CL(P) \setminus \mathbb{L}_i$. Thus, the generalized Büchi accepting set can be defined by $F = \{F_1, \ldots, F_m\}$ where $F_i = \overline{\mathbb{L}_i}$ for each i.

In the LNFG algorithm, when we partition the set of vertices based on $\stackrel{\text{fin}}{=}$ relation, there are at most 2 nodes in each equivalence class, where

$$p \stackrel{\text{fin}}{=} q \quad \textbf{iff} \quad p \equiv q \text{ without considering } fin(l_x) \text{ and } l_x; \quad l_x \notin prop$$

We have proved that the number of nodes of an NFG of a formula is finite [1]. In the same way, we can also prove the number of nodes of an LNFG of a formula is finite. In fact, when an LNFG is produced based on its NFG of a formula, each node of the NFG is at most split into two nodes. Since each node of an LNFG is marked by same label at most twice, so the set of labels is also finite.

To facilitate for producing LNFGs, some pre-processes are needed. We replace $P \to Q$, $P \leftrightarrow Q$ by $\neg P \vee Q$ and $P \wedge Q \vee \neg P \wedge \neg Q$ respectively; we also replace $\varepsilon\, prj\, \varepsilon$ by ε, $(P_1, \ldots, P_m)\, prj\, \varepsilon$ by $P_1; \ldots; P_m$, $(P_1, \ldots, P_t \vee P'_t, \ldots, P_m)$ prj Q by $(P_1, \ldots, P_t, \ldots, P_m)$ prj $Q \vee (P_1, \ldots, P'_t, \ldots, P_m)$ prj Q, $(P_1, \ldots, P_m)$ prj $(Q \vee Q')$ by $(P_1, \ldots, P_m)$ prj $Q \vee (P_1, \ldots, P_m)$ prj Q', $(w \wedge P_1, \ldots, P_m)$ prj Q and $(P_1, \ldots, P_m)$ prj $Q \wedge w$ by $w \wedge (P_1, \ldots, P_m)$ prj Q; further, we replace $\varepsilon\,; P$ by P, and $\varepsilon \wedge w\,; P$ by $w \wedge P$, respectively, where w is a state formula; finally, we replace $true; P$ by $P \vee \bigcirc \Diamond P$.

We move negation operators to the front of atomic propositions or chop or projection formulas as possible as we can by means of distributive laws on conjunction and disjunction operations; we also simplify the formula by using logic laws: $\neg\neg P \equiv P$, $P \wedge \neg P \equiv false$ $(\varepsilon \wedge more \equiv false)$, $P \vee \neg P \equiv true$ $(\varepsilon \vee more \equiv true)$, $P \wedge false \equiv false$, $P \vee false \equiv P$, $P \wedge true \equiv P$, $P \vee true \equiv true$, $false\,; P \equiv false$, $P\,; false \equiv false$, $(P_1, \ldots, false, \ldots, P_m)$ prj $Q \equiv false$, $(P_1, \ldots, P_m)$ prj $false \equiv false$.

A loop without duplicated vertices is called a simple loop. To find out all simple loops in an LNFG, we use a DFS algorithm to search the LNFG. To do so, we use a global stack S to store the visited vertices in DFS. For a vertice v, if its successor node w is not in S, w is pushed into S and recursively DFS w; otherwise, we find a loop and record it, then continue depth first searching. If all successor nodes of v are visited, v is popped and we return to DFS of the previous node.

Tarjan algorithm is a strongly connected components searching algorithm based on DFS. Define $\mathrm{DFN}(u)$ as searching number (timestamp) of u and $\mathrm{Low}(u)$ as searching number of the earliest vertice that is reachable for u or subtrees of u. $\mathrm{Low}(u) = \min\{\mathrm{DFN}(u), \mathrm{Low}(v), \mathrm{DFN}(v)\}$, where $u \to v$ is an edge.

Fig. 4. LNFG of formula (1).

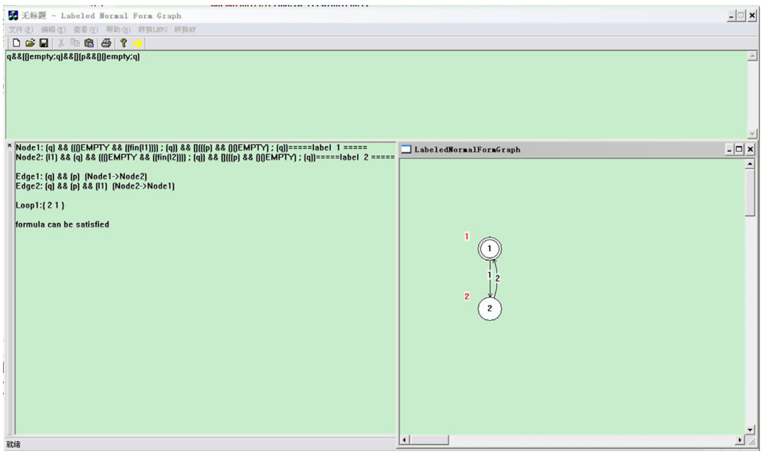

Fig. 5. LNFG of formula (2).

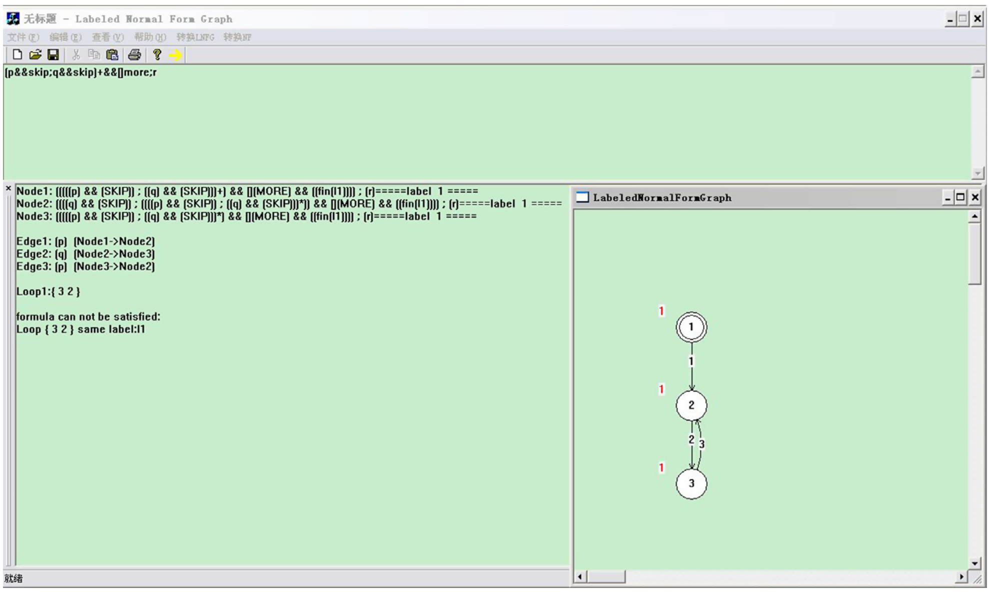

Fig. 6. LNFG of formula (3).

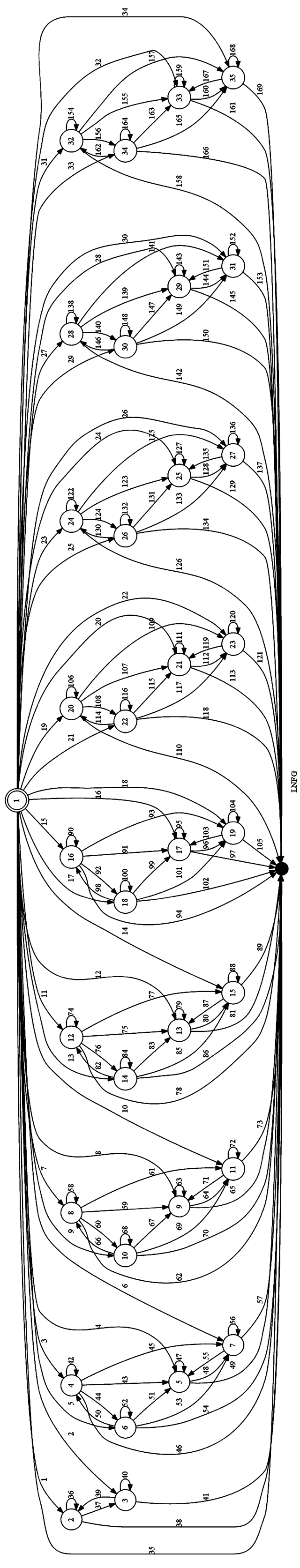

Fig. 7. LNFG of formula (4).

5.2. Examples

The LNFGs of the first three examples are artificially presented in the previous section. In this section, we run the program to produce the LNFGs and the satisfaction results of the formulas automatically.

(1) $P \equiv (p \wedge \Box \bigcirc p; \bigcirc\Box q) \wedge (\Box r; \bigcirc\Box q)$.
The LNFG of formula (1) is constructed in Fig. 4.
(2) $R \equiv q \wedge (\bigcirc\varepsilon; q) \wedge \Box(p \wedge \bigcirc\bigcirc\varepsilon; q)$
The LNFG of formula (2) is shown in Fig. 5.
(3) $(p \wedge skip; q \wedge skip)^+ \wedge \Box more; r$
The LNFG of formula (3) is shown in Fig. 6.
(4) $((\Box\Diamond p_0 \to \Box\Diamond p_1) \wedge (\Box\Diamond p_2 \to \Box\Diamond p_0) \wedge (\Box\Diamond p_3 \to \Box\Diamond p_2) \wedge (\Box\Diamond p_4 \to \Box\Diamond p_2) \wedge (\Box\Diamond p_5 \to \Box\Diamond p_3) \wedge (\Box\Diamond p_6 \to \Box\Diamond(p_5 \vee p_4)) \wedge (\Box\Diamond p_7 \to \Box\Diamond p_6) \wedge (\Box\Diamond p_1 \to \Box\Diamond p_7)) \to \Box\Diamond p_8$
This formula is a typical example given in [6]. Our decision procedure can produce the LNFG of the formula as shown in Fig. 7.

6. Conclusion

We have presented a practical decision procedure for PPTL with infinite models. As a matter of fact, this decision procedure can also be used as decision procedures for PITL [5] and PLTL [9] since PPTL subsumes PITL and PLTL. Based on the decision procedure, a model checker for PPTL (can also be used for PITL and PLTL) have also been developed [10]. In the future, we will further investigate the techniques such as symbolic, bounded, abstract, probabilistic model checking for taking over the state space exploring problem. We will also develop practical model checkers for PPTL.

Acknowledgements

We are grateful to Miss Xiaofang Liu for her assistance in implementing the decision procedure in C++.

Appendix A. Algorithm PRE

This algorithm is recursive and basically uses case statements to deal with different formulas. Function PRE(R), ImpEquPre(R), PreRecur(R), not_PRE(R), con_PRE(R), dis_PRE(R), cchop_PRE(R), prj_PRE(R).

Algorithm PRE(R): preprocessing of a PPTL formula R.

```
Function PRE(R)
/* precondition : R is a PPTL formula as a syntax tree */
/* postcondition: PRE (R) returns the reduced form of R */
begin function
  //reduce operator → and ↔
  ImpEquPre(R);
  return PreRecur(R);
end function
```

Algorithm ImpEquPre(R): preprocessing → and ↔ of a PPTL formula R.

```
Function ImpEquPre(R)
/* precondition: R is a PPTL formula as a syntax tree */
/* postcondition: ImpEquPre(R) returns the → and ↔ reduced form of R */
begin function
  for i = 1 to m /* assume R has m subtrees R_1, ..., R_m */
    if R_i is P_i → Q_i
    then ImpEquPre(P_i); ImpEquPre(Q_i);R_i is rewritten as ¬P_i ∨ Q_i;
    else if R_i is P_i ↔ Q_i
        then ImpEquPre(P_i); ImpEquPre(Q_i); R_i is rewritten as P_i ∧ Q_i ∨ ¬P_i ∧ ¬Q_i;
        endif
    endif
  endfor
  return R;
end function
```

Algorithm PreRecur(R): preprocessing a PPTL formula R without $\rightarrow$ and $\leftrightarrow$.

Function
/* precondition: R is a PPTL formula as a syntax tree without operator $\rightarrow$ and $\leftrightarrow$ */
/* postcondition: PreRecur(R) returns the reduced form of R */
begin function
case
R is $\neg P$: **return** not_PRE(R);
R is $P \wedge Q$: **return** con_PRE(R);
R is $P \vee Q$: **return** dis_PRE(R);
R is $P_1; \ldots; P_m$: **return** chop_PRE(R);
R is $(P_1, \ldots, P_m)\ prj\ Q$: **return** prj_PRE($R$);
R is $\Box P$: **return** $\Box$PreRecur(P);
R is $\Diamond P$: **return** $\Diamond$PreRecur(P);
R is $\bigcirc P$: **return** $\bigcirc$PreRecur(P);
R is $len(x)$: **if** x=0: **then return** ε; **else return** $\bigcirc^x \varepsilon$; **endif**
R is $fin(P)$: **return** fin(PreRecur(P));
R is $keep(P)$: **return** $keep$(PreRecur(P));
R is $halt(P)$: **return** $halt$(PreRecur(P));
R is $true$: **return** R;
R is $false$: **return** R;
R is $p(p \in prop)$: **return** R;
endcase
end function

Algorithm not_PRE(R): preprocessing a PPTL formula $R = \neg P$.

Function
/* precondition: R is a PPTL formula, R=$\neg P$ */
/* postcondition: not_PRE(R) returns the reduced form of R */
begin function
case
R is $\neg\varepsilon$: **return** $more$;
R is $\neg more$: **return** ε;
R is $\neg true$: **return** $false$;
R is $\neg false$: **return** $true$;
R is $\neg\neg P$: **return** PreRecur(P);
R is $\neg P$: **break**;
endcase
PreRecur(P);
case
R is $\neg\varepsilon$: **return** $more$;
R is $\neg more$: **return** ε;
R is $\neg true$: **return** $false$;
R is $\neg false$: **return** $true$;
R is $\neg\neg P$: **return** PreRecur(P);
R is $\neg P$: **return** R;
endcase
end function

Algorithm con_PRE(R): preprocessing a PPTL formula $R = P \wedge Q$.

Function
/* `precondition`: R is a PPTL formula, $R = P \wedge Q$ */
/* `postcondition`: con_PRE(R) returns the reduced form of R */
begin function
  case
    R is $P \wedge \neg P$: **return** *false*;
    R is $\varepsilon \wedge more$: **return** *false*;
    R is $P \wedge false$: **return** *false*;
    R is $false \wedge Q$: **return** *false*;
    R is $P \wedge true$: **return** PreRecur(P);
    R is $true \wedge Q$: **return** PreRecur(P);
    R is $P \wedge Q$: **break**;
  endcase
  PreRecur(P);
  if P is *true* **then return** PreRecur(Q); **endif**
  if P is *false* **then return** *false*; **endif**
  PreRecur(Q);
  case
    R is $P \wedge \neg P$: **return** *false*;
    R is $\varepsilon \wedge more$: **return** *false*;
    R is $P \wedge false$: **return** *false*;
    R is $P \wedge true$: **return** PreRecur(P);
    R is $P \wedge Q$: **return** R;
  endcase
end function

Algorithm dis_PRE(R): preprocessing a PPTL formula $R = P \vee Q$.

Function
/* `precondition`: R is a PPTL formula, $R = P \vee Q$ */
/* `postcondition`: dis_PRE(R) returns the reduced form of R */
begin function
  case
    R is $P \vee \neg P$: **return** *true*;
    R is $\varepsilon \vee more$: **return** *true*;
    R is $P \vee false$: **return** PreRecur(P);
    R is $false \vee Q$: **return** PreRecur(Q);
    R is $P \vee true$: **return** *true*;
    R is $true \vee Q$: **return** *true*;
    R is $P \vee Q$: **break**;
  endcase
  PreRecur(P);
  if P is *true* **then return** *true*; **endif**
  if P is *false* **then return** PreRecur(Q); **endif**
  PreRecur(Q);
  case
    R is $P \vee \neg P$: **return** *true*;
    R is $\varepsilon \vee more$: **return** *true*;
    R is $P \vee false$: **return** PreRecur(P);
    R is $P \vee true$: **return** *true*;
    R is $P \vee Q$: **return** R;
  endcase
end function

Algorithm chop_PRE(R): preprocessing a PPTL formula $R = P ; Q$.

Function
/∗ `precondition`: R is a PPTL formula, $R = P ; Q$ ∗/
/∗ `postcondition`: chop_PRE(R) returns the reduced form of R ∗/
begin function
  if R is $P_1 ; \dots ; P_m$ **and** any of $(P_1 ; \dots ; P_m)$ is false
  then return *false*
  endif
  case
    R is $\varepsilon ; P$: **return** PreRecur(P);
    R is $\varepsilon \wedge w ; P$: **return** PreRecur($w \wedge P$);
    R is $true ; P$: **return** PreRecur($P \wedge \bigcirc\Diamond P$);
    R is $P ; Q$: **break**;
  endcase
  PreRecur(P);
  if P is *true* **then** PreRecur($Q \vee \bigcirc\Diamond Q$); **endif**
  if P is *false* **then return** *false*; **endif**
  PreRecur(Q);
  case
    R is $\varepsilon ; P$: **return** PreRecur(P);
    R is $\varepsilon \wedge w ; P$: **return** PreRecur($w \wedge P$);
    R is $P ; false$: **return** *false*;
    R is $P ; true$: **return** PreRecur(P);
    R is $P ; Q$: **return** R;
  endcase
end function

Algorithm prj_PRE(R): preprocessing a PPTL formula $R = (P_1, \dots, P_m)\ prj\ Q$.

Function
/* `precondition`: R is a PPTL formula, $R = (P_1, \dots, P_m)\,\mathrm{prj}\,Q$ */
/* `postcondition`: prj_PRE(R) returns the reduced form of R */
begin function
  case
    R is $\varepsilon\,\mathrm{prj}\,\varepsilon$: **return** ε;
    R is $(P_1, \dots, P_m)\,\mathrm{prj}\,\varepsilon$: **return** PreRecur($P_1 ; \dots ; P_m$);
    R is $(P_1, \dots, P_i \vee P_i', \dots, P_m)\ prj\ Q$:
      return PreRecur($(P_1, \dots, P_i, \dots, P_m)\,\mathrm{prj}\,Q \vee (P_1, \dots, P_i' \dots, P_m)\ prj\ Q$);
    R is $(P_1, \dots, P_m)\ prj\ (Q \vee Q')$:
      return PreRecur($(P_1, \dots, P_m)\ prj\ Q \vee (P_1, \dots, P_m)\ prj\ Q'$);
    R is $(w \wedge P_1, \dots, P_m)\ prj\ Q$: **return** PreRecur($w \wedge (P_1, \dots, P_i, \dots, P_m)\ prj\ Q$);
      /* where w is a state formula */
    R is $(P_1, \dots, P_m)\ prj\ Q \wedge w$: **return** PreRecur($w \wedge (P_1, \dots, P_i, \dots, P_m)\ prj\ Q$);
      /* where w is a state formula */
    R is $(P_1, \dots, false, \dots, P_m)\ prj\ Q$: **return** *false*;
    R is $(P_1, \dots, P_m)\ prj\ false$: **return** *false*;
    R is $(P_1, \dots, P_m)\ prj\ Q$: **break**;
  endcase
  for i=1 **to** m
    PreRecur(P_i);
    if P_i is *false* **then return** *false*; **endif**
  endfor
  PreRecur(Q);
  if Q is *false* **then return** *false*; **endif**
  case
    R is $\varepsilon\ prj\ \varepsilon$: **return** ε;
    R is $(P_1, \dots, P_m)\ prj\ \varepsilon$: **return** PreRecur($P_1 ; \dots ; P_m$);
    R is $(P_1, \dots, P_i \vee P_i', \dots, P_m)\ prj\ Q$:
      return PreRecur($(P_1, \dots, P_i, \dots, P_m)\ prj\ Q \vee (P_1, \dots, P_i' \dots, P_m)\ prj\ Q$);
    R is $(P_1, \dots, P_m)\ prj\ (Q \vee Q')$:
      return PreRecur($(P_1, \dots, P_m)\ prj\ Q \vee (P_1, \dots, P_m)\ prj\ Q'$);
    R is $(w \wedge P_1, \dots, P_m)\ prj\ Q$: **return** PreRecur($w \wedge (P_1, \dots, P_i, \dots, P_m)\ prj\ Q$);
      /* where w is a state formula */
    R is $(P_1, \dots, P_m)\ prj\ Q \wedge w$: **return** PreRecur($w \wedge (P_1, \dots, P_i, \dots, P_m)\ prj\ Q$);
      /* where w is a state formula */
    R is $(P_1, \dots, P_m)\ prj\ Q$: **return** R;
  endcase
end function

Appendix B. DFSLoopSearch Algorithm

Algorithm DFSLoopSearch(G): search loops in LNFG G.

```
Function
/* precondition: G an LNFG G = {CL, EL, V_0, ..., L = {L_1, ..., L_m}} */
/* postcondition: DFSLoopSearch(G) returns a boolean variable
  indicating whether there exist loops or not */
begin function
  //initialization
  dfsStack = Φ;/ * global stack of DFS search */
  loopSet = Φ;/ * global set of simple loops in LNFG */
  existLoop = false;/ * indicate whether there exist loops in LNFG * /
  //kernal
  deleteIrrelevantNodes(G);
  if CL = Φ then return existLoop; else existLoop = true; endif
  for i = 1 to n
    if v_i.DFN undefined then Tarjan(G, v_i);endif
  endfor
  for i = 1 to m/ * assume there are m subgraphs */
    choose a random vertice v in CL of G_i, and dfsStack.push(v);
    dfsOneSubgraph(G_i);
  endfor
end function
```

Algorithm deleteIrrelevantNodes(G): delete irrelevant nodes.

```
Function
/* precondition: G an LNFG G = {CL, EL, V_0, ..., L = {L_1, ..., L_m}} */
/* postcondition: DFSLoopSearch(G) delete nodes whose in degree or
   out degree is zero */
begin function
  for i=1 to m
    v_i.indgree = 0; v_i.outdegree = 0;
  endfor
  isEmpty = false;
  nodeDeleted = false;
  for i=1 to n
    w_i.outdgree++;v_i.indgree++;/ * assume e_i is w_i → v_i */
  endfor
  for i=1 to l/ * assume there l nodes in CL * /
    if v_i.indegree = 0
    then delete v_i and delete edges start with v_i;nodeDeleted = true;
    endif
    if v_i.outdegree=0
    then delete v_i and delete edges end with v_i;nodeDeleted = true;
    endif
  endfor
  if CL = Φ then return;
  else if nodeDeleted = true thendeleteIrrelevantNodes(G); endif
  endif
end function
```

Algorithm dfsOneSubgraph(G): DFS a connected graph and obtain its loops.

Function
/* precondition: G a strongly connected graph */
/* postcondition: DFSLoopSearch(G) obtains loops in G */
begin function
  $w =$ dfsStack.front();
  for i=1 **to** m
    /* assume there are m edges starting from w and $e_i = w \to v_i$ */
    if v_i is in dfsStack
    then loopSet+=$\{v_i, \ldots, w\}$;
    else dfsStack.push(v_i);dfsOneSubgraph(G);
    endif
  endfor
  dfsStack.pop();
end function

Appendix C. Tarjan Algorithm

Algorithm Tarjan(G, v): obtain strongly connected graphs associated with v.

Function
/* precondition: G a directed graph,v is a vertice in G */
/* postcondition: Tarjan(G) obtains strongly connected
  components associated with v */
begin function
  v.DFN=v.Low=++Index;/* Index is numeral order in Tarjan */
  Stack.push(v);
  foreach $v \to w$ **in** EL
    if w is not visited
    then Tarjan(G, w);v.Low $=$ min{v.Low,w.Low};
    else
      if w is in Stack
      then v.Low $=$ min{v.Low,w.DFN};
      endif
    endif
  endfor
  if v.DFN $=$ v.Low
  then
    subGraph CG;
    do u $=$ Stack.pop(); CG.add(u);
    until (u $=$ v)
  endif
end function

References

[1] Z. Duan, C. Tian, Li Zhang, A decision procedure for propositional projection temporal logic with infinite models, Acta Inform. 45 (1) (2008) 43–78.
[2] Z. Duan, C. Tian, An improved decision procedure for propositional projection temporal logic with infinite models, in: Proceedings of ICFEM 2010, LNCS, vol. 6447, pp. 90–105.
[3] Z. Duan, Temporal Logic and Temporal Logic Programming, Science Press, Beijing, 2006.
[4] Z. Duan, M. Koutny, C. Holt, Projection in temporal logic programming, in: F. Pfenning (Ed.), Proceedings of Logic Programming and Automatic Reasoning, in: Lecture Notes in Artificial Intelligence, vol. 822, Springer-Verlag, 1994, pp. 333–344.
[5] B. Moszkowski, Executing Temporal Logic Programs, Cambridge University Press, 1986.
[6] Moshe Y. Vardi, The Büchi complementation saga, in: STACS 2007, in: LNCS, vol. 4393, 2007, pp. 12–22.
[7] Glynn Winskel, The Formal Semantics of Programming Languages, Foundations of Computing, The MIT Press, Cambridge, MA.
[8] Z. Duan, An extended interval temporal logic and a framing technique for temporal logic programming, Ph.D. thesis, University of Newcastle upon Tyne, 1996.
[9] A. Pnueli, The temporal logic of programs, in: Proceedings of the 18th IEEE Symposium on Foundations of Computer Science, IEEE, New York, 1977, pp. 46–57.
[10] Cong Tian, Zhenhua Duan, Model checking propositional projection temporal logic based on SPIN, in: ICFEM, in: LNCS, vol. 4789, Springer-Verlag, Nov 2007, pp. 246–265.