# Characterizing Petri Nets with the Temporal Logic CTL

Liu Zhifeng
School of Computer Science and Telecommunication Engineering, Jiangsu University
Zhenjiang, China
liuzf@ujs.edu.cn

Xing Zhihu
School of Computer Science and Telecommunication Engineering, Jiangsu University
Zhenjiang, China

***Abstract*—Model checking is a powerful technique for verifying systems and detecting errors at early stages of the design process. When model checking is used to check properties of Petri net, the specification has to be expressed in temporal logics. In this paper we will focus on how to characterize some important properties of Petri net such as reachability, liveness et al. with the computation tree logic *CTL*. Under the characterization Petri net can be verified automatically with the help of a model checker.**

***Keywords-model checking; petri net; computation tree logic***

## I. INTRODUCTION

Model Checking [1, 2] is a powerful technique for verifying systems and detecting errors at early stages of the design process, which is obtaining wide acceptance in industrial setting. In Model Checking, the specification is expressed in temporal logic-either Computation Tree Logic(*CTL*)[3] or Linear Temporal Logic(*LTL*)[4]-and the system is modeled as a finite state machine(*FSM*). Petri net[5] is a state machine, which is widely used for modeling reactive systems such as communication protocols, workflow[6].

Petri nets are a graphical and mathematical modeling tool applicable to many systems. They are a promising tool for describing and studying information processing systems that are characterized as being concurrent asynchronous, distributed, parallel, nondeterministic, and stochastic. As a graphical tool, Petri nets can be used as a visual communication aid similar to flow charts, block diagrams, and networks. In addition, token are used in these nets to simulate the dynamic and concurrent activities of systems. As a mathematical tool, it is possible to set up state equations, algebraic equations, and other mathematical models governing the behavior of systems. Petri nets can be used by both practitioners and theoreticians. Thus, they provide a powerful medium of communication between them: practitioners can learn from theoreticians how to make their models more methodical, and theoreticians can learn from practitioners how to make their models more realistic.

When model checking is applied to check Petri nets, the properties are needed to be expressed with temporal logic. Some important properties include reachability[7], liveness, boundedness[8], reversibility, home state, coverability, and persistence. In this paper we exploited how to describe these properties with computation tree logic *CTL*. Under the description, a model checker such as NuSMV can be used to check Petri nets automatically.

## II. PETRI NET

Historically speaking, Petri nets originate from the early work of Carl Adam Petri. Since then the use and study of Petri nets have increased considerably. The classical petri net is a directed bipartite graph with two node types called places and transitions. The nodes are connected via directed arcs. Connections between two nodes of the same type are not allowed.

**Definition 2.1.** A petri net is a four-tuple: $PN = (P, T, F, M_0)$

1) $P$ is a finite set of places.
2) $T$ is a finite set of transitions
3) $P \cap T = \varnothing, P \cup T \neq \varnothing$
4) $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs.
5) $M_0 : P \rightarrow N$ is an initial state.

$\bullet u = \{v \mid v \in (P \cup T) \wedge (v,u) \in F\}$ is called the preset of $u$, $u\bullet = \{v \mid v \in (P \cup T) \wedge (u,v) \in F\}$ is called the postset of $u$.

**Definition 2.2.** A transition $t$ is enabled in the state $M$ if and only if $\forall p \in \bullet t, M(p) \geq 1$.

If the transition $t$ is enabled in the state $M$, it can be fired. When $t$ is fired, the new state $M'$ is computed as follows:

$$M'(p) = \begin{cases} M(p) - 1 : p \in \bullet t - t\bullet \\ M(p) + 1 : p \in t\bullet - \bullet t \\ M(p) : otherwise \end{cases} \qquad (1)$$

Following is some notations used in the paper.

*a)* $M \xrightarrow{t} M'$ *:* The transition $t$ is fired at the state $M$ and a new state $M'$ is computed as (1).

*b)* $M_1 \xrightarrow{\sigma} M_k$ *:* For the transition sequence $\sigma = t_1 t_2 ... t_{k-1}$ there exists states $M_2 M_3 ... M_{k-1}$ such that $M_i \xrightarrow{t_i} M_{i+1}$ for $1 \leq i \leq k-1$. $M_k$ is reachable from $M_1$ if there exists a transition sequence $\sigma = t_1 t_2 ... t_{k-1}$ such that

$M_1 \xrightarrow{\sigma} M_k$. The empty transition is allowed, i.e., the state is reachable from itself.

*c)* $M_1 \xrightarrow{*} M_k$ *:* There exists a transition sequence $\sigma = t_1 t_2 ... t_{k-1}$ such that $M_1 \xrightarrow{\sigma} M_k$.

*d)* $[M)$ *is the set of states reachable from M.*

**Definition 2.3.** A petri net $PN = (P, T, F, M_0)$ is bounded iff for each place $p$ there is a natural number *n* such that for every reachable state the number of tokens in $p$ is no more than *n*, i.e. , for every $M \in [M_0), M(p) \le n$.

**Definition 2.4.** A petri net $PN = (P, T, F, M_0)$ is *n*-bounded iff for every reachable state $M, \forall p \in P, M(p) \le n$.

## III. The Computation Tree Logic CTL

**Definition 3.1.** The Kripke structure of a petri net $PN = (P, T, F, M_0)$ is a four-tuple $(S, R, L, s_0)$, where *S* is the set of states, *R* is the transition relation, and $s_0$ is the initial state. *S* and *R* are defined inductively as follows.

*1)* $s_0 = M_0 \in S$

*2)* If $M \in S$ *then* $(M, M) \in R$

*3)* If $M \in S$ and $\exists t \in T$, $M \xrightarrow{t} M'$ then $M' \in S$ and $(M, M') \in R$

*4)* $L : S \to 2^{AP}$ is a function that labels each state with the set of atomic propositions true in that state.

*5)* *S* and *R* have no other elements.

In the following of this section K is the Kripke structure of a petri net $PN = (P, T, F, M_0)$ $(P = \{p_1, p_2, ..., p_m\})$ and $s_0$ is the initial state in *K*. Computation Tree Logic *CTL* are composed of path quantifiers and temporal operators. The path quantifiers are used to describe the branching structurein the computation tree. There are two such quantifiers *A*(for all computation paths) and *E*( for some computation path). The temporal operators describe properties of a path through the tree. There are four basic operators:

- • *X* ("next time") requires that a property holds in the second state of the path.
- • *F* ("eventually" or "in the future") operator is used to assert that a property will hold at some state on the path.
- • *G* ("always" or "globally") specifies that a property holds at every state on the path.
- • *U* operator is used to combine two properties. It holds if there is a state on the path where the second property holds, and at every preceding state on the path, the first property holds.

The syntax of *CTL* formulas is given by the following rules:

*1)* If $p \in AP$, then $p$ is a *CTL* formula,

*2)* If $f$ is a *CTL* formula, then $\neg f, \mathrm{AX}f, \mathrm{EX}f, \mathrm{AF}f$, $\mathrm{EF}f, \mathrm{AG}f, \mathrm{EG}f$ are *CTL* formulas.

*3)* If $f$ and g are *CTL* formulas, then $f \wedge g$, $f \vee g$, $\mathrm{A}f\mathrm{U}g$, $\mathrm{E}f\mathrm{U}g$ are *CTL* formulas.

We define the semantics of *CTL* with respect to a *Kripke* structure. ( $f_1$, $f_2$ are *CTL* formulas and $p$ is an atomic proposition.)

*1)* $K, s \models p \Leftrightarrow p \in L(s)$.

*2)* $K, s \models \neg f_1 \Leftrightarrow K, s \not\models f_1$.

*3)* $K, s \models f_1 \vee f_2 \Leftrightarrow K, s \models f_1$ *or* $K, s \models f_2$.

*4)* $K, s \models f_1 \wedge f_2 \Leftrightarrow K, s \models f_1$ *and* $K, s \models f_2$.

*5)* $K, s \models EXf_1 \Leftrightarrow$ there is a path $\pi = s_0 s_1 ...$ from $s(s = s_0)$ such that $K, s_1 \models f_1$.

*6)* $K, s \models AXf_1 \Leftrightarrow$ for every path $\pi = s_0 s_1 ...$ from $s(s_0 = s)$ such that $K, s_1 \models f_1$.

*7)* $K, s \models EFf_1 \Leftrightarrow$ there exists a path $\pi = s_0 s_1 ...$ from $s(s_0 = s)$ on which there exists a state $s_k (k \ge 0)$ such that $K, s_k \models f_1$.

*8)* $K, s \models AFf_1 \Leftrightarrow$ for every path $\pi = s_0 s_1 ...$ from $s(s_0 = s)$, there exists a $s_k (k \ge 0)$ on the path $\pi$ such that $K, s_k \models f_1$.

*9)* $K, s \models EGf_1 \Leftrightarrow$ there exists a path $\pi = s_0 s_1 ...$ from $s(s_0 = s)$ such that $\forall i \ge 0,\ K, s_i \models f_1$.

*10)* $K, s \models AGf_1 \Leftrightarrow$ for every path $\pi = s_0 s_1 ...$ from $s(s_0 = s)$ such that $\forall i \ge 0,\ K, s_i \models f_1$.

*11)* $K, s \models Ef_1Uf_2 \Leftrightarrow$ there exists a path $\pi = s_0 s_1 ...$ from $s(s_0 = s)$ on which there exists a $k \ge 0$ such that $K, s_k \models f_2$ and $\forall 0 \le j \le k,\ K, s_j \models f_1$.

*12)* $K, s \models Af_1Uf_2 \Leftrightarrow$ *for every path* $\pi = s_0 s_1 ...$ *from* $s(s_0 = s)$ *there exists a* $k \ge 0$ *on the path* $\pi$ *such that* $K, s_k \models f_2$ *and* $\forall 0 \le j \le k,\ K, s_j \models f_1$.

## IV. Characterizing Petri Net's Properties with CTL

### *A. Reachability*

Reachability is a fundamental basis for studying the dynamic properties of any system. The firing of an enabled transition will change the token distribution in a net according to the transition rule described in Section 2. A sequence of firing will result in a sequence of marking. The reachability problem for petri nets is the problem of finding if $M$ is reachable from $M_0$ for a given state $M$ .We have the following theorem.

**Theorem 4.1.** *M* is reachable from the initial state $M_0$ iff $K, s_0 \models EF(M'(p_1) = M(p_1) \wedge M'(p_2) = M(p_2) \wedge ... \wedge M'(p_m) = M(p_m))$.

### *B. Boundedness*

Boundedness is a very important property for Petri nets. By verifying that the net is bounded or safe, it is guaranteed that there will be no overflows in the buffers or registers, no matter what firing sequence is taken.

**Theorem 4.2.** A petri net $PN = (P, T, F, M_0)$ is *n*−bounded iff $K, s_0 \models AG(\prod_{i=1}^{m} M(p_i) \le n)$.

**Theorem 4.3.** A petri net $PN = (P, T, F, M_0)$ is bounded iff there is a positive integer $k$ such that $K, s_0 \models AG(\prod_{i=1}^{m} M(p_i) \leq k)$.

*C. Liveness*

The concept of liveness is closely related to the complete absence of deadlocks in operating systems. A Petri net $PN = (P, T, F, M_0)$ is said to be live if no matter what states has been reached from $M_0$, it is possible to ultimately fire any transition of the net by progressing through some further firing sequence.This means that a live Petri net guarantees deadlock-free operation, no matter what firing sequence is chosen.

**Theorem 4.4.** A petri net $PN = (P, T, F, M_0)$ is live iff for all *t*, $K, s_0 \models AGEF(\prod_{p \in \bullet t} M(p) \geq 1)$.

Liveness is an ideal property for many systems. However it is impractical and too costly to verify this strong property for some systems such as the operating systems of a large computer. Thus we relax the liveness condition and define different levels of liveness as follows[5][8].

**Definition 4.5.** A transition t in a petri net $PN = (P, T, F, M_0)$ is said to be dead ($L_0$-live) if *t* can never be fired in any firing sequence.

**Theorem 4.6.** A transition t in a petri net $PN = (P, T, F, M_0)$ is dead ($L_0$-live) iff $K, s_0 \models AG(\neg(\prod_{p \in \bullet t} M(p) \geq 1))$.

**Definition 4.7.** A transition t in a petri net $PN = (P, T, F, M_0)$ is said to be $L_1$-live if *t* can be fired at least once in some firing sequence.

**Theorem 4.8.** A transition t in a petri net $PN = (P, T, F, M_0)$ is said to be $L_1$-live iff $K, s_0 \models EF(\prod_{p \in \bullet t} M(p) \geq 1)$.

**Definition 4.9.** A transition t in a petri net $PN = (P, T, F, M_0)$ is said to be $L_3$-live if *t* can be fired infinitely often in some firing sequence.

**Theorem 4.10.** A transition t in a petri net $PN = (P, T, F, M_0)$ is said to be $L_3$-live iff $K, s_0 \models EGF(\prod_{p \in \bullet t} M(p) \geq 1)$.

*D. Reversibility and Home State*

**Definition 4.11.** A Petri net $PN = (P, T, F, M_0)$ is said to be reversible if for each state $M$ in $[M_0)$, $M_0$ is reachable from $M$. Thus in a reversible net one can always get back to the initial state.

**Theorem 4.12.** A Petri net $PN = (P, T, F, M_0)$ is reversible iff $K, s_0 \models AGEF(\prod_{i=1}^{m} M(p_i) = M_0(p_i))$.

In many applications , it is not necessary to get back to the initial state as along as one can get back to some state. Therefore, we relax the reversibility condition and define a home state.

**Definition 4.13.** A state $M'$ is said to be a home state if for each state $M$ in $[M_0)$, $M'$ is reachable from $M$.

**Theorem 4.14.** A state $M'$ is a home state iff $K, s_0 \models AGEF(\prod_{i=1}^{m} M(p_i) = M'(p_i))$.

*E. Coverability*

Coverability is closely related to $L_1$−liveness. Let $M$ be the minimum marking needed to enable a transition *t*. Then *t* is dead if and only if $M$ is not coverable.That is, *t* is $L_1$−live if and only if $M$ is coverable.

**Definition 4.15.** A state $M$ in a petri net $PN = (P, T, F, M_0)$ is said to be coverable if there exists a state $M'$ in $[M_0)$ such that $M'(p) \geq M(p)$ for each $p$ in the petri net.

**Theorem 4.16.** A state $M$ in a petri net $PN = (P, T, F, M_0)$ is said to becoverable iff $K, s_0 \models EF(\prod_{i=1}^{m} M'(p_i) \geq M(p_i))$.

*F. Persistence*

**Definition 4.17.** A Petri net $PN = (P, T, F, M_0)$ is said to be persistent if for any two enabled transitions, the firing of one transition will not disable the other.

A transition in a persistent net, once it is enabled, will stay enabled until it fires. The notion of persistence is useful in the context of parallel program schemata and speed-independent asynchronous circuits. Persistency is closely related to conflict free nets, and a safe persistent net can be transformed into a marked graph by duplicating some transitions and places. Note that all marked graphs are persistent, but not all persistent nets are marked graphs.

**Theorem 4.18.** A Petri net $PN = (P, T, F, M_0)$ is persistent if and only if for any two transitions $t_1$, $t_2$, $K, s_0 \models AG((\prod_{p \in \bullet t_1} M(p) \geq 1) \wedge \prod_{p \in \bullet t_2} M(p) \geq 1) \rightarrow \sum_{p \in \bullet t_1 \cap \bullet t_2} M(p) \geq 2)$.

## V. Conclusions and Future work

Model checking is an automatical technique and widely used in industries. When model checking is applied to check properties of Petri net, the specification has to be expressed in temporal logic such as *CTL*, *LTL*. In this paper we focused on how to characterize some important properties of Petri net such as reachability, liveness et al. with the computation tree logic *CTL*. The characterization makes Petri net be verified automatically with the help of a model checker such as NuSMV. There are several directions in which further work is needed. Firstly we will explore how to express more properties with *CTL*. Second work is to characterize other models with *CTL*.

## ACKNOWLEDGEMENTS

This work was supported by the National Natural Science Foundation of China (Grant Nos. 60773049, 61003288), the People with Ability Foundation of Jiangsu University (07JDG014), the Fundamental Research Project of the Natural Science in Colleges of Jiangsu Province (08KJD520015), and the Ph.D. Programs Foundation of Ministry of Education of China (20093227110005).

## REFERENCES

[1] E. M. Clarke, O. Grumberg, and D. Peled. Model Checking, MIT Press, 1999.

[2] Christel Baier,Joost-Pieter Katoen, Principles of Model Checking, MIT Press,2008.

[3] M. Ben-Ari. Z. Manna. and A. Pnueli, The temporal logic of branching time. Acta Information, 20(1983), 207-226.

[4] A. Pnueli. A temporal logic of concurrent programs. Theoretical Computer Science, 13, 45-60.

[5] T. Murata. Petri Nets: Properties, analysis and application. Proc of the IEEE, Vol. 77(4):541-574, April 1989.

[6] W.M. P. van der Aalst. Verification of Workflow Nets. In P. Azema and G. Balbo, editors, Application and Theory of Petri Nets 1997, volume 1248 of LNCS, pages 407-426. 1997.

[7] M. Praveen, K. Lodaya, Analyzing Reachability for Some Petri Nets With Fast Growing Markings. Electronic Notes in Theoretical Computer Science, Volume 223, 26 December 2008, Pages 215-237.

[8] Li Jiaoa, To-Yat Cheunga, Weiming Lu. On liveness and boundedness of asymmetric choice nets. Theoretical Computer Science, Volume 311, Issues 1-3, 23 January 2004, Pages 165-197.