IEICE TRANS. INF. & SYST., VOL.E103–D, NO.3 MARCH 2020

LETTER

Symbolic Representation of Time Petri Nets for Efficient Bounded Model Checking

Nao IGAWA[†], ***Nonmember***, **Tomoyuki YOKOGAWA**[†a)], ***Member***, **Sousuke AMASAKI**[†], ***Nonmember***, **Masafumi KONDO**[††], **Yoichiro SATO**[†], ***and*** **Kazutami ARIMOTO**[†], ***Members***

SUMMARY Safety critical systems are often modeled using Time Petri Nets (TPN) for analyzing their reliability with formal verification methods. This paper proposed an efficient verification method for TPN introducing bounded model checking based on satisfiability solving. The proposed method expresses time constraints of TPN by Difference Logic (DL) and uses SMT solvers for verification. Its effectiveness was also demonstrated with an experiment.

key words: *formal verification, Time Petri Net, SMT Solver, difference logic*

1. Introduction

Industries strive for assuring their safety critical systems, failure of which exposes man-life and social infrastructures to serious danger. For assurance, the safety critical systems are often modeled using Time Petri Net (TPN) [1], [2], which is one of the main extensions of Petri Nets with time constraints [3]. The reliability of systems is then analyzed with the TPN models applying formal verification methods. A formal verification approach based on model checking methodology expresses possibly tremendous states of a system as assignments to propositional variables and real ones. Thus, tools based on this approach such as TINA [4] and Romeo [5] are suffered from a complex and space consuming problem. Representing the state space symbolically such as [6]–[8] is considered promising. These studies applied SAT-based bounded model checking [9] to the verification of Timed Automata.

In [10], we proposed a verification method for TPN based on bounded model checking that extended an SAT-based bounded model checking method for Petri Nets [11] and showed the advantage of our method over existing model checker. The method expressed time constraints in linear arithmetic (LA) and enabled to use an SMT (Satisfiability Modulo Theories) solver for verification. Its symbolic representation adopted a similar encoding of timed systems in [7] such that firing of transitions and elapsing place delays were encoded as boolean expressions separately. We also extended the symbolic representation to express time constraints in difference logic (DL) [12] in a similar manner to [8], and an efficient satisfiability solving algorithm for formulas described in DL [13] could be applicable.

In this paper, we conduct a comparison experiment and show the advantage of the encoding based on DL compared to the existing encoding based on LA. We prepare different-sized TPN examples to evaluate the scalability of the encoding. We also demonstrate that the formula size reduction [10] has a greater effect on the proposed encoding.

Manuscript received May 6, 2019.
Manuscript revised September 11, 2019.
Manuscript publicized December 20, 2019.

[†]The authors are with Okayama Prefectural University, Sojashi, 719–1197 Japan.
[††]The author is with Kawasaki University of Medical Welfare, Kurashiki-shi, 701–0193 Japan.
a) E-mail: t-yokoga@cse.oka-pu.ac.jp

DOI: 10.1587/transinf.2019EDL8086

2. Bounded Model Checking for TPN

2.1 TPN

We introduce P-Time Petri Nets (P-TPN) [14], a subclass of TPN that extend Petri Nets (PN) so that it can represent a place delay. P-TPN is defined as 6-tuple P-TPN $= (P, T, F, F_{in}, M_0, X)$, where P is a set of *places*, T is a set of *transitions*, $F \subseteq (P \times T) \cup (T \times P)$ is a set of *arcs*, $F_{in} \subset (P \times T)$ is a set of *inhibitor arcs* and $M_0 \subseteq P$ is an initial marking. $X : P \to (\mathbb{Z}^+) \times (\mathbb{Z}^+ \cup +\inf)$ is a function mapping places to place delays ($\mathbb{Z}^+$ denotes a set of integers which are greater than or equal to zero). We focus on a *safe* P-TPN where each place has at most one token, and no arc weight is considered.

A place can have a *token*. A place delay represents a time required to enable a token of the place and have lower bound l_i and upper bound u_i ($l_i \leq u_i$) for $p_i \in P$. A token in place p_i can be enabled on and after l_i elapsed and must be enabled until u_i elapsed. A transition must fire immediately when all of its input places have an *enabled* token, and the tokens move to its output places through the transition. An inhibitor arc represents a condition of inhibiting a transition firing. Transitions connected with input places through an inhibitor arc can fire only if those places have no enabled token. Input and output places of a transition $t \in T$ are described as $\bullet t$ and $t\bullet$, and input and output transitions of a place $p \in P$ are described as $\bullet p$ and $p\bullet$, respectively. Places with an inhibitor arc to transition t are represented as $\circ t$.

2.2 Bounded Model Checking

A state s of TPN which has l places ($l = |P|$) is defined by a variable c and two l - vectors $\mathbf{m} = (m_1, \ldots, m_l)$ and $\mathbf{z} = (z_1, \ldots, z_l)$ for the vector of places $\mathbf{p} = (p_1, \ldots, p_l)$ where $p_i \in P$. c denotes the global time of the TPN and

z_i represents the time when p_i gets a token. By using the variables, the elapsed time of the token in p_i is expressed as $c - z_i$. m_i is a boolean variable which evaluates to true when p_i has a token. Here, a boolean function over variables of s which holds iff a state s belongs to a state set S is called a *characteristic function* of S. Similarly, a transition relation can be specified as a characteristic function over s and s' which holds iff s can change to s'.

To carry out SAT-based bounded model checking, it is necessary to encode two characteristic functions $\mathcal{N}_k$ and $\mathcal{R}_k$. $\mathcal{N}_k(s_0, \cdots, s_k)$ denotes that the initial state s_0 can reach s_k by k-steps through $s_1, s_2, \cdots, s_{k-1}$. $\mathcal{R}_k(s_0, \cdots, s_k)$ denotes that a desired property to be checked is satisfied in any one of the states $s_0, \cdots, s_k$. Here, a *step* represents the changing of states caused by time elapsing and firing of transitions. A step is defined as a changing of states s and s' through a state s'', where s changes to s'' by elapsing of some time interval and s'' changes to s' by firing of some transitions. Bounded model checking can check whether a system can satisfy the desirable property within k-steps by determining the satisfiability of $\mathcal{N}_k \wedge \mathcal{R}_k$ using SMT-solver. If $\mathcal{N}_k \wedge \mathcal{R}_k$ is satisfiable, the property can be satisfied in k-steps from the initial state. That is, the system satisfies the reachability of the states where the property holds, and the verification succeeds. If not, it means that the system does not satisfy the property at least within k-steps.

2.3 Symbolic Representation

In our encoding, time constraints are expressed in difference logic (DL), which is a sub logic of linear arithmetic and has a form restricted as $x - y \bowtie c$ for variables x, y and constant c, where $\bowtie$ represent equality or inequality. Constraints expressed in DL can be solved efficiently by searching a negative cycle of a weighted directed graph.

Two step types are supposed in our encoding: an *elapsing sub-step* caused by time elapsing and a *firing sub-step* caused by transition firing discretely. $C(s, s')$ and $\mathcal{F}(s, s')$ denote characteristic functions of an elapsing sub-step and firing sub-steps. A characteristic function of one step $\mathcal{T}(s, s')$ is defined as $\mathcal{T}(s, s') \stackrel{def}{=} C(s, s'') \wedge \mathcal{F}(s'', s')$ for some intermediate state s'' between the steps.

A transition t must fire immediately when all $p \in {}^\bullet t$ have enabled tokens and any $p \in t^\circ$ has no enabled token; otherwise, t can not fire. Here, we introduce two characteristic functions $En_t(s)$ and $Ds_t(s)$. $En_t(s)$ denotes t must fire in a state s and $Ds_t(s)$ denotes t can not fire in a state s as follows:

$$En_t(s) \stackrel{def}{=} \bigwedge_{p_i \in {}^\bullet t} (m_i \wedge u_i \leq c - z_i) \wedge \bigwedge_{p_i \in t^\circ} \neg(m_i \wedge l_i \leq c - z_i),$$

$$Ds_t(s) \stackrel{def}{=} \bigvee_{p_i \in {}^\bullet t} \neg(m_i \wedge l_i \leq c - z_i) \vee \bigvee_{p_i \in t^\circ} (m_i \wedge u_i \leq c - z_i).$$

Note that $En_t(s)$ and $Ds_t(s)$ are defined so that there exist states where neither $En_t(s)$ nor $Ds_t(s)$ holds. This is because whether the token in p_i is enabled or not is decided non-deterministically when $l_i \leq c - z_i < u_i$. In such states, whether t fires or not is also decided non-deterministically.

As an elapsing sub-step is performed when there is no transition which must fire, $C(s, s')$ can be defined as follows:

$$C(s, s') \stackrel{def}{=} (c' - c > 0) \wedge \bigwedge_{t \in T} \neg En_t(s) \wedge \bigwedge_{p_i \in P} (z_i' = z_i \wedge m_i' \leftrightarrow m_i)$$
$$\vee (c' - c = 0) \wedge \bigwedge_{p_i \in P} (z' = z_i \wedge m_i' \leftrightarrow m_i)$$

where m_i', z_i' and c' represent m_i, z_i and c on state s' respectively. Note that $c' - c > 0$ holds only when $En_t(s)$ does not hold for all $t \in T$. As you can see, $C(s, s')$ is satisfied even if some t can fire on the global time x between c and c' ($c < x < c'$). Thus $C(s, s')$ provides an over-approximation of the time elapsing behavior of TPN and $\mathcal{T}(s, s')$ permits sequences of states which can not occur in actuality. However, $\mathcal{T}(s, s')$ is satisfied for all sequences of states from s to s' which the TPN allows and our encoding can check reachability to states where desired properties hold.

$\mathcal{F}(s, s')$ can be defined as a conjunction of characteristic functions $F_t(s_i, s_j)$ for all transitions ($n = |T|$), which holds iff s_i changes to s_j by a firing of t or $s_i = s_j$:

$$\mathcal{F}(s, s') \stackrel{def}{=} F_{t_1}(s, s_1) \wedge F_{t_2}(s_1, s_2) \wedge \cdots \wedge F_{t_n}(s_{n-1}, s').$$

$\mathcal{F}(s, s')$ holds if s can reach s' by firing of some sequence of transitions following the order in $t_1, \ldots, t_n$. $F_t(s, s')$ can be defined as follows:

$$\begin{aligned}\mathcal{F}_t(s, s') \stackrel{def}{=} &(c' - c = 0) \wedge \neg Ds_t(s) \\ &\wedge \bigwedge_{p_i \in t^\bullet} (m_i' \wedge z_i' = c) \wedge \bigwedge_{p_i \in {}^\bullet t \setminus t^\bullet} (\neg m_i' \wedge z_i' = z_i) \\ &\wedge \bigwedge_{p_i \in P \setminus ({}^\bullet t \cup t^\bullet)} (m' \leftrightarrow m_i \wedge z' = z_i) \\ &\vee (c' - c = 0) \wedge \bigwedge_{p_i \in P} (m_i' \leftrightarrow m_i \wedge z_i' = z_i).\end{aligned}$$

Since $\mathcal{N}_k$ represents that the initial state s_0 can reach the state s_k by k-steps, $\mathcal{N}_k$ can be obtained as follows:

$$\mathcal{N}_k \stackrel{def}{=} \mathcal{I}(s_0) \wedge \mathcal{T}(s_0, s_1) \wedge \mathcal{T}(s_1, s_2) \wedge \cdots \wedge \mathcal{T}(s_{k-1}, s_k)$$

where $\mathcal{I}(s)$ denotes a characteristic function of the initial states and is defined as follows:

$$\mathcal{I}(s) \stackrel{def}{=} \bigwedge_{p_i \in M_0} m_i \wedge \bigwedge_{p_i \notin M_0} \neg m_i \wedge \bigwedge_{p_i \in P} z_i = 0.$$

Suppose a characteristic function $R(s)$ which holds iff a given property is satisfied in s. When $\mathcal{N}_k$ holds for the states $s_0, \cdots, s_k$ and $R(s_i)$ holds in some state s_i ($0 \leq i \leq k$), $\mathcal{N}_k$ also holds by assigning s_i to $s_{i+1}, \ldots, s_k$. Because $\mathcal{T}(s, s')$ is true when $s = s'$. Thus we can obtain $\mathcal{R}_k \stackrel{\text{def}}{=} R(s_k)$.

For example, deadlock freeness of TPN can be checked by expressing deadlock states as a characteristic function. A

Table 1 Effects of variables replacing.

	Step	# of variables				# of constraints			
		before replacing		after replacing		before replacing		after replacing	
		LA	DL	LA	DL	LA	DL	LA	DL
#1	1	292,469	292,859	3,815	3,442	300,670	300,325	12,016	10,908
	5	1,459,353	1,461,299	16,083	14,214	1,490,770	1,490,485	47,500	43,400
	10	2,917,958	2,921,849	31,418	27,679	2,978,395	2,978,185	91,855	84,015
#2	1	2,461,701	2,462,895	11,479	10,450	2,486,925	2,486,084	36,703	33,639
	5	12,300,265	12,306,231	49,155	44,006	12,397,525	12,397,336	146,415	135,111
	10	24,598,470	24,610,401	96,250	85,951	24,785,775	24,786,401	283,555	261,951
#3	1	25,975,013	25,979,063	38,423	35,218	26,060,356	26,058,043	123,766	114,198
	5	129,849,417	129,869,663	166,467	150,438	130,180,260	130,181,319	497,310	462,094
	10	259,692,422	259,732,913	326,522	294,463	260,330,140	260,335,414	964,240	896,964

Table 2 Comparison of execution time (sec.).

	Step	MathSAT		Z3		SMTInterpol		yices		CVC4		result
		LA	DL	LA	DL	LA	DL	LA	DL	LA	DL	
#1	1	0.043	0.031	0.031	0.031	0.479	0.420	0.012	0.015	0.149	0.148	unsat
	2	0.069	0.061	0.071	0.071	0.629	0.570	0.023	0.027	0.277	0.270	unsat
	3	0.118	0.116	0.138	0.137	0.730	0.690	0.035	0.041	0.421	0.405	unsat
	4	0.219	0.186	0.250	0.238	0.941	0.891	0.048	0.057	0.544	0.524	unsat
	5	4.182	1.097	0.965	0.785	6.158	8.586	6.397	0.209	1.113	1.008	unsat
	6	5.325	2.876	2.989	1.813	11.633	11.916	16.283	0.544	2.860	1.406	unsat
	7	18.612	3.956	8.524	3.264	17.235	39.502	12.725	0.590	3.395	2.454	unsat
	8	19.148	10.185	14.632	5.260	78.673	58.682	14.222	1.393	11.403	4.746	unsat
	9	31.172	17.919	33.772	11.692	201.717	187.003	32.164	3.885	49.292	14.752	unsat
	10	78.780	42.962	61.003	32.251	> 600	258.586	91.310	7.780	69.900	184.923	unsat
#2	1	0.103	0.108	0.107	0.104	0.888	0.843	0.038	0.046	0.620	0.549	unsat
	2	0.353	0.283	0.384	0.333	2.880	1.499	0.089	0.101	1.304	1.143	unsat
	3	0.878	0.557	1.040	0.717	2.187	2.130	0.343	0.173	1.987	1.754	unsat
	4	1.799	1.082	1.775	1.343	10.130	6.396	0.854	0.291	2.762	2.418	unsat
	5	6.991	4.115	4.165	4.024	16.678	35.643	4.230	0.427	3.969	3.582	unsat
	6	10.310	6.721	8.226	6.014	55.335	62.101	9.524	1.229	11.929	5.239	unsat
	7	40.479	46.537	29.248	14.886	148.538	294.035	42.590	4.308	20.572	10.864	unsat
	8	65.158	68.318	77.255	37.106	591.304	320.213	71.035	8.127	65.388	29.093	unsat
	9	88.706	53.858	118.414	84.530	> 600	> 600	212.159	19.547	208.174	73.094	unsat
	10	296.510	173.050	224.211	135.838	—	—	340.504	48.888	389.901	204.180	unsat
#3	1	0.371	0.384	0.437	0.415	1.867	1.775	0.129	0.160	1.837	1.827	unsat
	2	1.727	1.520	2.706	1.715	10.087	10.064	0.468	0.460	4.804	4.819	unsat
	3	5.398	3.688	7.790	4.658	23.477	19.291	2.835	1.011	7.868	7.157	unsat
	4	11.624	7.474	16.579	9.238	130.407	104.760	26.152	2.488	10.562	9.784	unsat
	5	49.279	21.836	63.565	25.299	91.997	215.889	89.636	5.140	14.849	13.539	unsat
	6	173.166	89.663	156.303	96.474	> 600	> 600	150.506	16.581	31.128	28.166	unsat
	7	335.292	337.237	> 600	188.084	—	—	> 600	32.838	304.836	81.453	unsat
	8	> 600	> 600	—	> 600	—	—	—	139.254	451.156	404.832	unsat
	9	—	—	—	—	—	—	—	240.595	> 600	> 600	unsat
	10	—	—	—	—	—	—	—	> 600	—	—	N/A

deadlock occurs when all tokens of places are enabled and no transition can fire. Thus a characteristic function $R_d(s)$ which denotes deadlock states can be defined as follows:

$$R_d(s) \stackrel{def}{=} \bigwedge_{t \in T} \neg En_t(s) \wedge \bigwedge_{p_i \in P} (m_i \rightarrow u_i \leq c - z_i).$$

2.4 Size of Formula to be Solved

To express time constraints in DL, we introduce the additional variable c which represents the global time for each state and the additional constraints over the value of c. Thus the formula to be solved increases in the number of variables. In [10], we provided the method for reducing formula size by replacing unchanged variables. Since the value of z_i will be unchanged until the place p_i gets a new token in the proposed encoding, there are many terms formed as $(z_i' = z_i)$ which can be removed by replacing z_i' with z_i. This makes it possible for the proposed encoding to reduce the formula size compared to the existing encoding.

3. Experiments

For evaluation, the proposed encoding was compared with the existing encoding based on Linear Arithmetic (LA) shown in [10]. This experiment was performed on the computer with Ubuntu 16.04 LTS OS, Intel Core i7 7700 3.6 GHz CPU, and 64 GB Memory. We supplied three TPNs #1, #2 and #3, whose numbers of places/transitions are 374/389,

1030/1193 and 3206/4049. We used five SMT solvers at the top of SMT-COMP 2018 [15]: MathSAT [16], Z3 [17], SMTInterpol [18], yices [19], and CVC4 [20].

Table 1 shows the comparison of effects of the formula size reduction by variable replacing. "Variable" denotes the variables which are declared in the formula, and "constraint" denotes the logical constraints in the formula. Here the logical constraint denotes x, $\neg x$, or $x \leftrightarrow y$ for boolean variable x and y, or constraints in LA. As shown in the Table, the number of variables and constraints can be significantly reduced by the variable replacing, and the reduction works well on the proposed encoding compared to the existing one.

Table 2 shows execution times of bounded model checking from 1 step to 10 steps for the three TPNs. The column "LA" and "DL" describes the execution times by the existing encoding and proposed encoding, respectively. We set the timeout to 600 seconds. As shown in the "result" column, deadlock cannot be detected for the TPNs within 10 steps. For all of the SMT solvers, the proposed encoding performed better than the existing encoding. In particular, yices showed a good performance for the formula in DL even if the size of the target TPN becomes large. For some cases, a performance for the formula in LA is better than the formula in DL. Since the size of formula is not so large, the differences of structure of the formula seems to affect the execution time rather than the effectiveness of the fast algorithm.

4. Summary and Future Work

A symbolic representation for efficient bounded model checking is proposed, and its effectiveness is demonstrated. An experiment with practical TPNs and a comparison with other tools such as TINA and Romeo are on our future research direction. We will extend this work to apply interpolation based unbounded model checking (UBMC) [12], [21]. Future work is an application the effective UBMC algorithm [22] to our encoding.

Acknowledgments

This research was partially supported by JST CREST Grant Number JPMJCR1531, Japan. A part of the experiments was conducted with the support of the 2017 research grant of the Okawa Foundation for Information and Telecommunications.

References

[1] S. Bernardi, J. Campos, and J. Merseguer, "Timing-Failure Risk Assessment of UML Design Using Time Petri Net Bound Techniques," IEEE Trans. Industr. Inform., vol.7, no.1, pp.90–104, 2011.
[2] Y.H. Kacem, A. Magdich, A. Mahfoudhi, C. Mraidha, and W. Karamti, "Using MDE and Priority Time Petri Nets for the schedulability analysis of Embedded Systems modeled by UML activity diagrams," Proc. IEEE 19th Int'l Conf. and Workshops on Eng. Comp. Based Syst. (ECBS 2012), pp.316–323, 2012.
[3] P. Merlin and D. Farber, "Recoverability of Communication Protocols–Implications of a Theoretical Study," IEEE Trans. Comm., vol.24, no.9, pp.1036–1043, Sept. 1976.
[4] B. Berthomieu, P.-O. Ribet, and F. Vernadat, "The tool TINA – Construction of abstract state spaces for Petri nets and Time Petri nets," International Journal of Production Research, vol.42, no.14, pp.2741–2756, 2007.
[5] D. Lime, O.H. Roux, C. Seidner, and L.-M. Traonouez, "Romeo: A parametric model-checker for petri nets with stopwatches," Proc. 15th Int'l Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2009), pp.54–57, 2009.
[6] G. Audemard, A. Cimatti, A. Kornilowicz, and R. Sebastiani, "SAT-Based bounded model checking for timed systems," Proc. 22nd IFIP WG 6.1 Int'l Conf. Houston on Formal Tech. for Networked and Distributed Syst. (FORTE 2002), pp.243–259, 2002.
[7] M. Sorea, "Bounded Model Checking for Timed Automata," Electron. Notes Theor. Comput. Sci., vol.68, no.5, pp.116–134, May 2003.
[8] Z. Chen, Z. Xu, J. Du, M. Mei, and J. Guo, "Efficient Encoding for Bounded Model Checking of Timed Automata," IEEJ Transactions on Electrical and Electronic Engineering, vol.12, no.5, pp.710–720, 2017.
[9] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu, "Symbolic Model Checking without BDDs," Proc. 5th Int'l Conf. Tools and Algorithms for Construction and Analysis of Syst. (TACAS '99), vol.1579, pp.193–207, 1999.
[10] T. Yokogawa, M. Kondo, H. Miyazaki, S. Amasaki, Y. Sato, and K. Arimoto, "Bounded model checking of Time Petri Nets using SAT solver," IEICE Electronics Express, vol.12, no.2, p.20141112, Jan. 2015.
[11] S. Ogata, T. Tsuchiya, and T. Kikuno, "SAT-Based Verification of Safe Petri Nets," Proc. Second Int'l Conf. Automated Technology for Verification and Analysis (ATVA 2004), vol.3299, pp.79–92, 2004.
[12] N. Igawa, T. Yokogawa, S. Amasaki, K. Komoku, Y. Sato, and K. Arimoto, "Interpolation Based Unbounded Model Checking for Time Petri Nets," Proc. 2018 IEEE 7th Global Conf. Consumer Electronics (GCCE), pp.587–591, 2018.
[13] S. Cotton and O. Maler, "Fast and Flexible Difference Constraint Propagation for DPLL(T)," Proc. 9th Int'l Conf. Theory and Applications of Satisfiability Testing (SAT 2006), vol.4121, pp.170–183, 2006.
[14] J. Sifakis, "Performance evaluation of systems using nets," Net Theory and Applications, LNCS, vol.84, pp.307–319, 1980.
[15] "SMT-COMP 2018," http://smtcomp.sourceforge.net/2018/
[16] R. Bruttomesso, A. Cimatti, A. Franzén, A. Griggio, and R. Sebastiani, "The MathSAT$_4$ SMT Solver," Proc. 20th Int'l Conf. Computer Aided Verification (CAV 2008), vol.5123, pp.299–303, 2008.
[17] L. De Moura and N. Bjørner, "Z3: An efficient SMT Solver," Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol.4963, LNCS, pp.337–340, 2008.
[18] J. Christ, J. Hoenicke, and A. Nutz, "SMTInterpol: an Interpolating SMT Solver," Proc. 19th Int'l SPIN Workshop on Model Checking of Softw. (SPIN 2012), vol.7385, pp.248–254, 2012.
[19] B. Dutertre, "Yices 2.2," Proc. 26th Int'l Conf.n Computer Aided Verification (CAV 2014), vol.8559, pp.737–744, 2014.
[20] C. Barrett, C.L. Conway, M. Deters, L. Hadarean, D. Jovanović, T. King, A. Reynolds, and C. Tinelli, "CVC4," Proc. 23rd Int'l Conf. Computer Aided Verification (CAV 2011), pp.171–177, 2011.
[21] K.L. McMillan, "Interpolation and SAT-Based Model Checking," Proc. 15th Int'l Conf. Computer Aided Verification (CAV 2003), vol.2725, pp.1–13, 2003.
[22] B. Li and F. Somenzi, "Efficient Abstraction Refinement in Interpolation-Based Unbounded Model Checking," Proc. 12th Int'l Conf. Tools and Algorithms for the Construction and Analysis of Syst. (TACAS 2006), vol.3920, pp.227–241, 2006.