Optimization Techniques for Model Checking Leads-to Properties in a Stratified Way

Canh Minh Do, Yati Phyo, Adrian Riesco, Kazuhhiro Ogata

canhdo@jaist.ac.jp

Japan Advanced Institute of Science and Technology (JAIST)

Present physically at Software Engineering Symposium 2024 (invited paper)

September 19, 2024

Outline

- Introduction
- Background
- $L + 1$-layer divide & conquer approach to leads-to model checking
- All counterexample state generation at once
- Layer configuration selection
- Conclusion

Outline

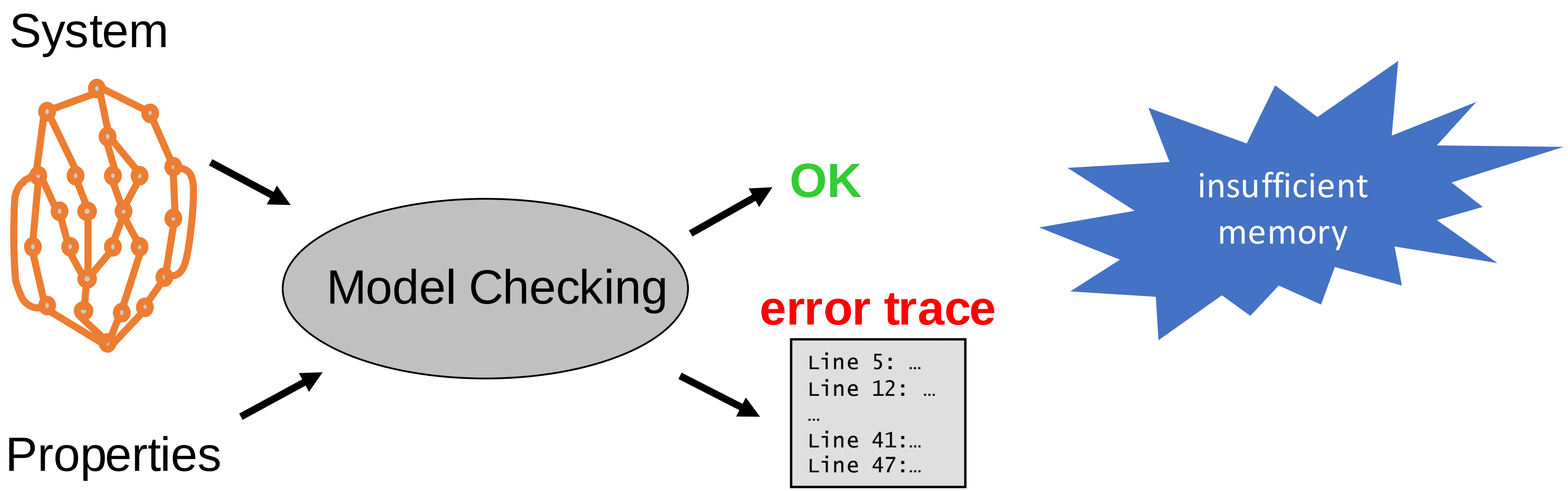
- Introduction
- Background
- $L + 1$-layer divide & conquer approach to leads-to model checking
- All counterexample state generation at once
- Layer configuration selection
- Conclusion

Introduction

- Model checking is one of the most successful formal verification techniques for verifying that a finite state system satisfies its desired properties.
- There are still some challenges in model checking:

  (1) The state space explosion problem (space challenge).

  (2) Improving the running performance of model checking (time challenge).



Outline

- Introduction
- Background
- $L + 1$-layer divide & conquer approach to leads-to model checking
- All counterexample state generation at once
- Layer configuration selection
- Conclusion

Kripke structure

A Kripke structure $\boldsymbol{K}$ is a tuple $\langle \boldsymbol{S}, \boldsymbol{I}, \boldsymbol{T}, \boldsymbol{A}, \boldsymbol{L} \rangle$, where

- $\boldsymbol{S}$ is a set of states.
- $\boldsymbol{I} \subseteq \boldsymbol{S}$ is the set of initial states.
- $\boldsymbol{T} \subseteq \boldsymbol{S} \times \boldsymbol{S}$ is a left-total binary relation over $\boldsymbol{S}$. $(s, s') \in \boldsymbol{T}$ is called a state transition from s to s' denoted $s \rightarrow_{\boldsymbol{K}} s'$ or $s \rightarrow s'$.
- $\boldsymbol{A}$ is a set of atomic propositions.
- $\boldsymbol{L}$ is a labeling function whose type is $\boldsymbol{S} \rightarrow 2^{\boldsymbol{A}}$. For $s \in \boldsymbol{S}$, $\boldsymbol{L(s)}$ is the set of atomic propositions that hold in s.

Kripke structure

A path π is an infinite sequence $s_0, \ldots, s_i, s_{i+1}, \ldots$ such that $s_i \rightarrow_K s_{i+1}$ for each i. Some notations are used for paths as follows:

- $\pi(i) \triangleq s_i$
- $\pi^i \triangleq s_i, s_{i+1}, \ldots$

A path π is called a computation of $\boldsymbol{K}$ if and only if $\pi(0) \in \boldsymbol{I}$.

Linear temporal logic (LTL)

The syntax of linear temporal logic (LTL) is as follows:

$$\varphi ::= \top \mid a \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\, \varphi \mid \varphi_1\, \mathcal{U}\, \varphi_2$$

where $a \in A$.

next connective (or operator)

until connective (or operator)

Linear temporal logic (LTL)

Some other connectives (or operators) are defined as follows:

$$\bot \triangleq \neg\top$$

$$\varphi_1 \wedge \varphi_2 \triangleq \neg((\neg\varphi_1) \vee (\neg\varphi_2))$$

$$\varphi_1 \Rightarrow \varphi_2 \triangleq (\neg\varphi_1) \vee \varphi_2$$

$$\varphi_1 \Leftrightarrow \varphi_2 \triangleq (\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1)$$

$$\Diamond \varphi_1 \triangleq \top\, \mathcal{U}\, \varphi_1 \qquad \Box \varphi_1 \triangleq \neg(\Diamond \neg\varphi_1)$$

eventually connective   always connective

$$\varphi_1 \leadsto \varphi_2 \triangleq \Box(\varphi_1 \Rightarrow \Diamond \varphi_2)$$

leads-to connective

Linear temporal logic (LTL)

For any Kripke structure $\boldsymbol{K}$, any path π of $\boldsymbol{K}$ and any LTL formulas φ, $\boldsymbol{K}, \pi \models \varphi$ is inductively defined as follows:

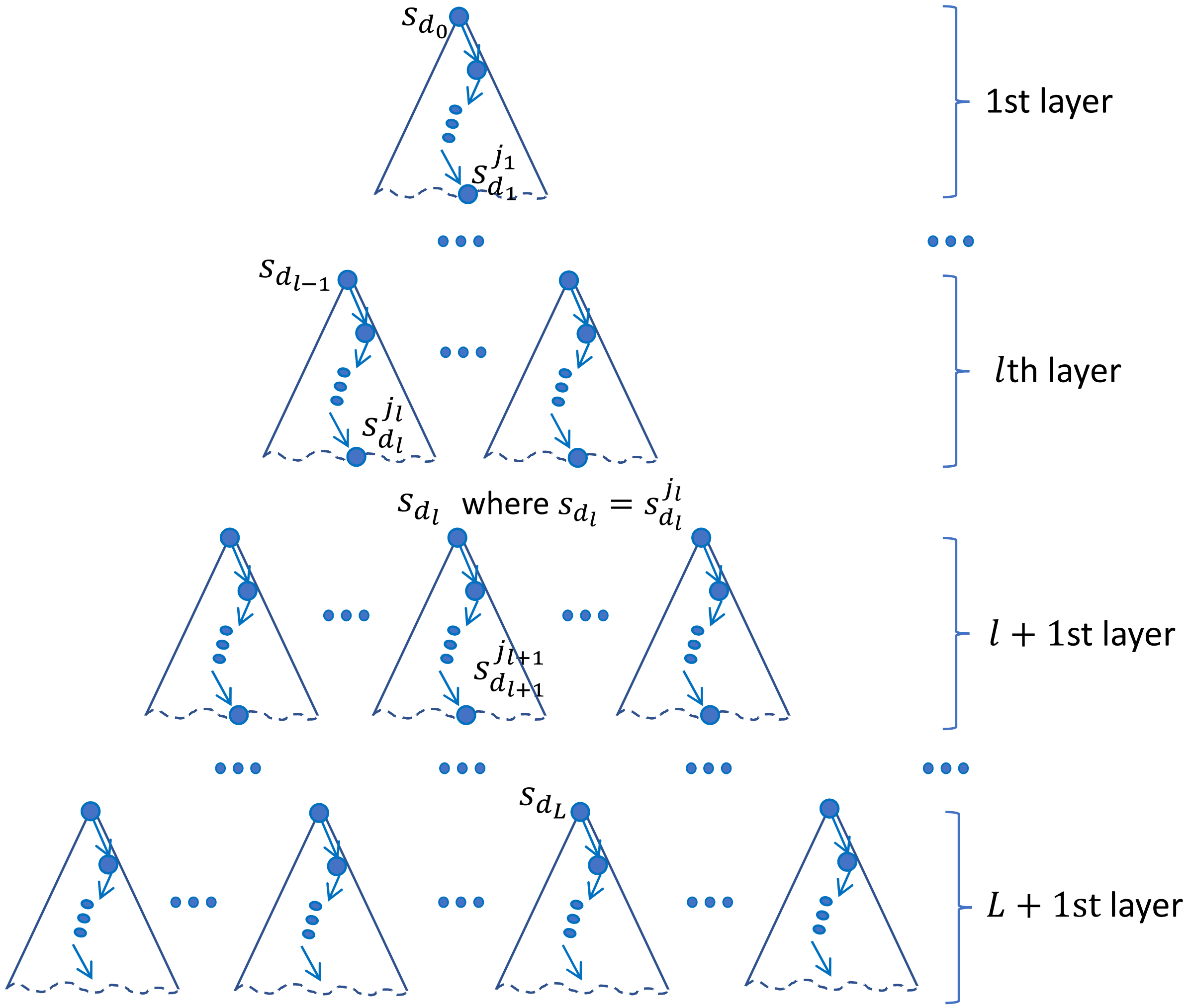
- $\boldsymbol{K}, \pi \models a$ if and only if $a \in \boldsymbol{L}(\pi(0))$
- $\boldsymbol{K}, \pi \models \top$
- $\boldsymbol{K}, \pi \models \neg\varphi_1$ if and only if $\boldsymbol{K}, \pi \not\models \varphi_1$
- $\boldsymbol{K}, \pi \models \varphi_1 \vee \varphi_2$ if and only if $\boldsymbol{K}, \pi \models \varphi_1$ and/or $\boldsymbol{K}, \pi \models \varphi_2$
- $\boldsymbol{K}, \pi \models \bigcirc \varphi_1$ if and only if $\boldsymbol{K}, \pi^1 \models \varphi_1$
- $\boldsymbol{K}, \pi \models \varphi_1 \,\mathcal{U}\, \varphi_2$ if and only if there exists a natural number i such that $\boldsymbol{K}, \pi^i \models \varphi_2$ and for each natural number $j < i$, $\boldsymbol{K}, \pi^j \models \varphi_1$

where φ_1 and φ_2 are LTL formulas. Then, $\boldsymbol{K} \models \varphi$ if and only if $\boldsymbol{K}, \pi \models \varphi$ for all computations π of $\boldsymbol{K}$.

Outline

- Introduction
- Background
- $L + 1$-layer divide & conquer approach to leads-to model checking
- All counterexample state generation at once
- Layer configuration selection
- Conclusion

Core idea



$L+1$-DCA2L2MC is a new technique to mitigate the state space explosion dedicated to leads-to properties $\varphi_1 \leadsto \varphi_2$, where φ_1, φ_2 are state propositions.

The core idea is to divide the reachable state space from each initial state into multiple layers, generating multiple sub-state spaces, and checking a smaller model checking problem for each sub-state space.

If each sub-state space is much smaller than the original reachable state space, the state space explosion problem may be mitigated.

$L+1$-layer divide & conquer approach to leads-to model checking

For each initial state $s_{d_0} \in I$, an infinite tree is made by using T in Kripke structure K.

Let us suppose that the infinite tree is split into $L+1$ layers.

A function d is used to express layer configuration such that:

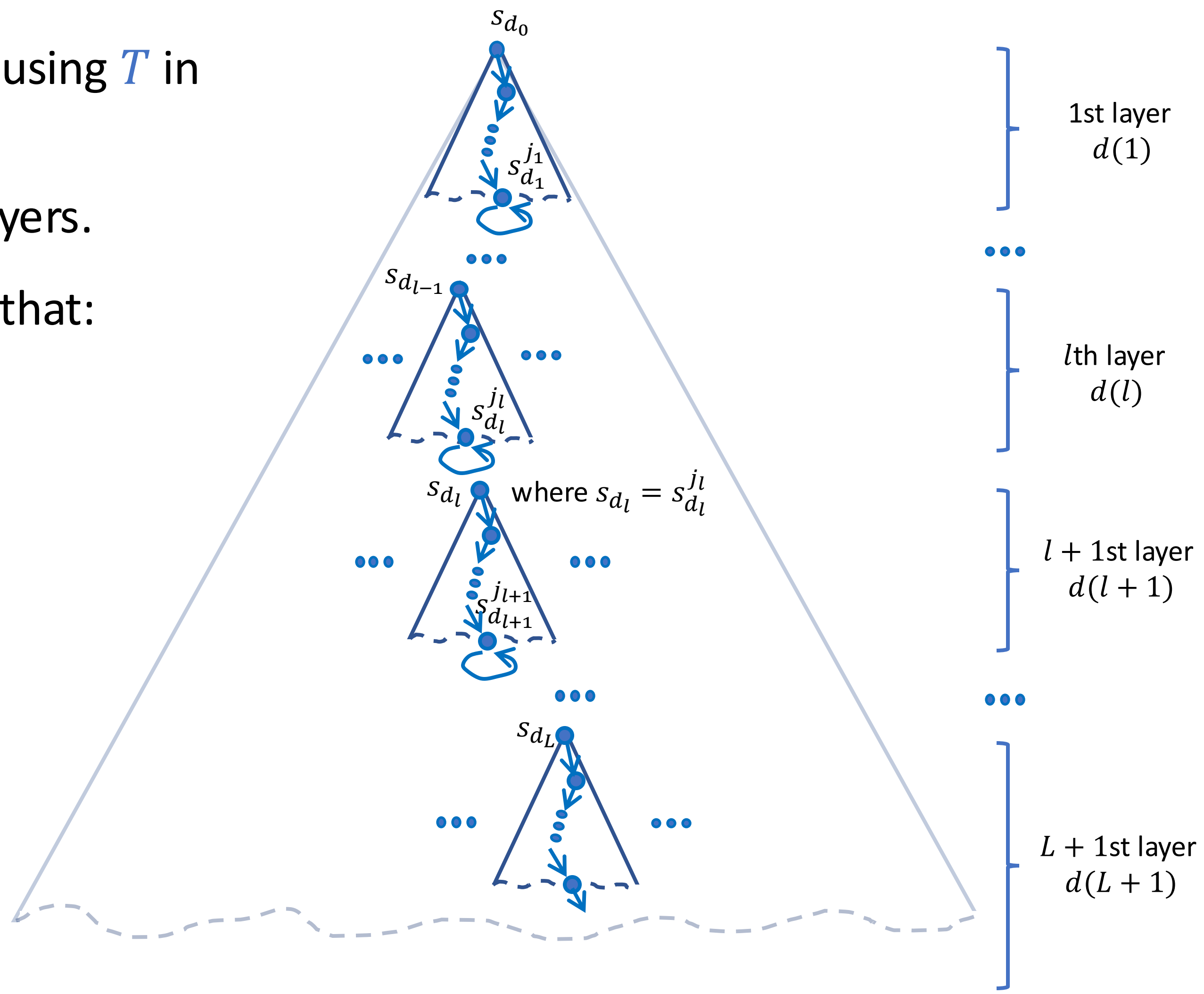
- $d(0) = 0$
- $d(i)$ is a non-zero natural number for $i = 1, \ldots, L$
- $d(L+1) = \infty$

$d_i = d(0) + \cdots + d(i)$ is the depth of states located the bottom at layer i for $i = 0, 1, \ldots, L$.

K_i is the Kripke structure obtained from K by

- deleting all transitions from each state at the depth d_i
- adding the self-transition to each state at depth d_i

for $i = 1, \ldots, L$.



$L + 1$-layer divide & conquer approach to leads-to model checking

Each path π of K_i is in the form $s_0^{(i)}, \dots, s_{j-1}^{(i)}, s_j^{(i)}, s_j^{(i)}, \dots$ where $s_j^{(i)}$ is a state located at the bottom of layer i (or the beginning of layer $i + 1$) for $i = 1, \dots, L$.

A property φ is either a leads-to property $\varphi_1 \leadsto \varphi_2$ or an eventual property $\Diamond\varphi_2$ where φ_1, φ_2 are state propositions.

If $K_i, \pi \nvDash \varphi$, then $s_j^{(i)}$ is called a counterexample state at layer i and last(π) = $s_j^{(i)}$.

$L+1$-layer divide & conquer approach to leads-to model checking

Let us consider a leads-to property $\varphi_1 \rightsquigarrow \varphi_2$, where φ_1, φ_2 are state propositions.

Let $K, s \vDash \varphi$ denote $K, \pi \vDash \varphi$ for all paths π of K that start with state s.

Let $AllS_L$ be the set of all states at depth d_L.

Let CxS_L be the set of all counterexample states at depth d_L for $K_L, s_{d_0} \vDash \varphi_1 \rightsquigarrow \varphi_2$.

$$K \vDash \varphi_1 \rightsquigarrow \varphi_2 \text{ iff } \forall s_{d_0} \in I, K, s_{d_0} \vDash \varphi_1 \rightsquigarrow \varphi_2$$

$$\approx$$

$$K, s_{d_0} \vDash \varphi_1 \rightsquigarrow \varphi_2 \text{ iff } \forall s \in AllS_L, K, s \vDash \varphi_1 \rightsquigarrow \varphi_2 \text{ and } \forall s \in CxS_L, K, s \vDash \Diamond\, \varphi_2.$$

$L + 1$-layer divide & conquer approach to leads-to model checking

Initially, $AllS_0 = \{s_{d_0}\}$ and $CxS_0 = \emptyset$.

• • •

For each layer i, we suppose to have $AllS_{i-1}$ and CxS_{i-1} from layer $i-1$ for $i = 1, \dots, L$.

$\forall s \in AllS_{i-1}$, the states are reachable from s with exactly $d(i)$ transitions stored in $AllS_i$ — generating states

$\forall s \in AllS_{i-1}$, let $Cxs1$ be the set of all counterexample states for $K_i, s \vDash \varphi_1 \leadsto \varphi_2$

$\forall s \in CxS_{i-1}$, let $Cxs2$ be the set of all counterexample states for $K_i, s \vDash \lozenge\, \varphi_2$

$CxS_i = \{s \mid \pi \in Cxs1 \cup Cxs2, s = last(\pi)\}$

generating counterexample states

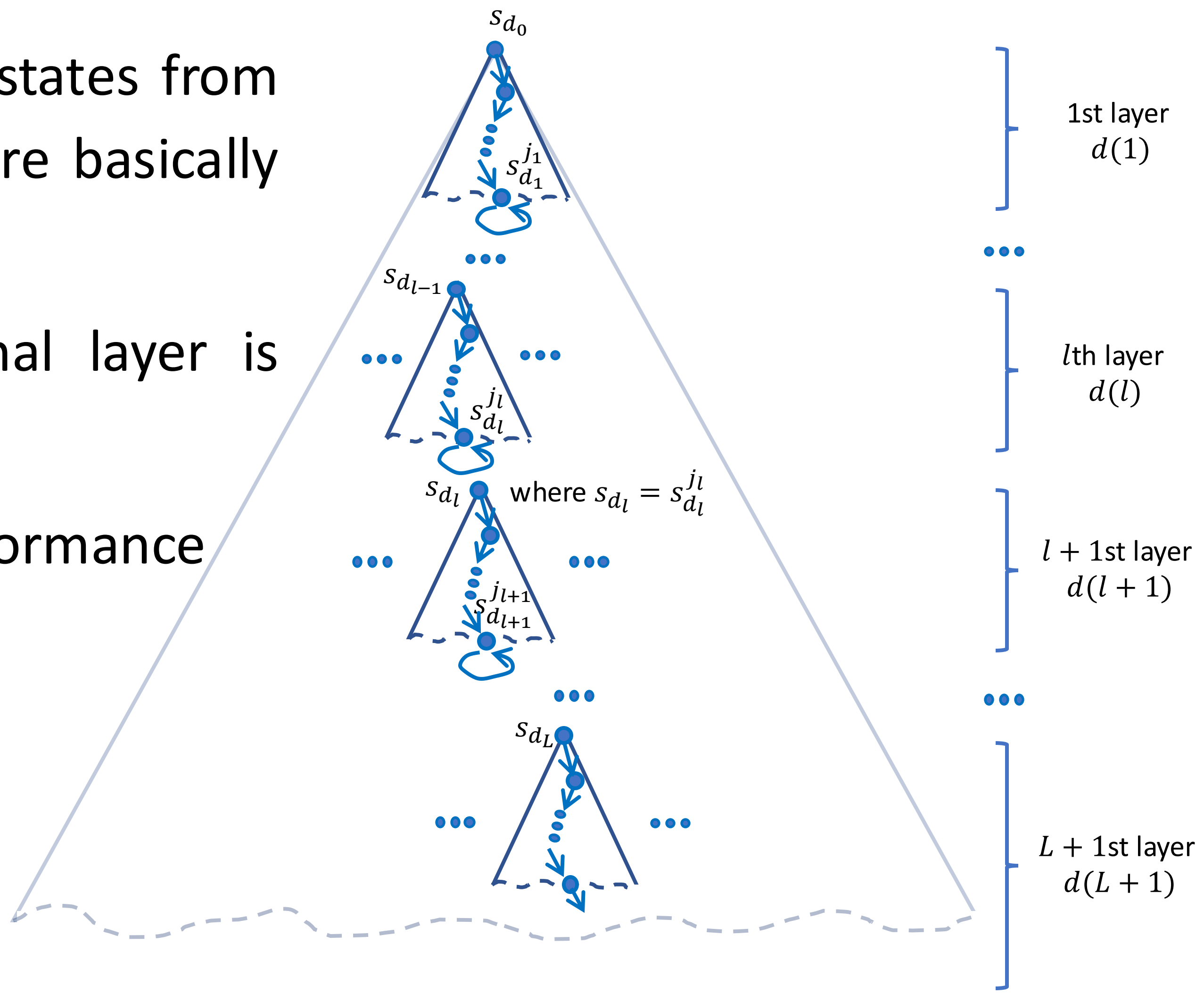
• • •

Finally, $AllS_L$ and CxS_L.

$L+1$-layer divide & conquer approach to leads-to model checking

- Generating states and counterexample states from each state at each intermediate layer are basically independent.
- Model checking each state at the final layer is completely independent.
- Hence, we can improve the running performance

  of $L+1$-DCA2L2MC by parallelization.



A support tool

- We develop a tool for sequential and parallel $L + 1$-DCA2L2MC in Maude by using its meta-programming facilities.
- Our tool can be used as an ordinary model checker in which users are supposed to provide:
  (1) A formal systems specification
  (2) A formal property specification
  (3) A layer configuration
  (4) A number of workers if parallelization is used
- Our tool automates model checking experiments.

Experiments

- We use four mutual exclusion protocols:

  (1) TAS is a simple mutual exclusion where test&set instruction is used.

  (2) Qlock is an abstract verson of the Dijkstra binary semaphore.

  (3) Anderson is an array-based mutual exclusion protocol invented by Anderson.

  (4) MCS is a list-based queuing mutual exclusion protocol whose variants have been used in Java virtual machines.

- We model check inWs1 $\leadsto$ inCs1 for all case studies.
- We use a MacPro that has 28 cores and 1.5 TB memory to conduct our experiments.

Experimental data for $L + 1$-DCA2L2MC with Maude LTL, Spin, LTSmin model checkers

Protocol	**Maude**	**Layers**	**L + 1-DCA2L2MC**	**SPIN**	**LTSmin**
Qlock (10 processes)	37d 1h 23m	2 2	5h 8m 22s	OOM (after 9h 28m 41s)	OOM (after 5h 0m 50s)
Anderson (9 processes)	12d 16h 42m	2 2	1h 15m 49s	OOM (after 3h 4m 32s)	OOM (after 6h 24m 4s)
MCS (6 processes)	25d 15h 46m	4 4 4 4 2	2d 14h 38m	62s	30m 54s
TAS (13 processes)	20h 18m	3 3 3	6d 13h 44m	33s	7m 39s

Experimental data for Parallel $L + 1$-DCA2L2MC with LTSmin + CNDFS and LTSmin + UFSCC

Protocol	**Layers**	**Parallel L + 1-DCA2L2MC (32 workers)**	**LTSmin + CNDFS (32 threads)**	**LTSmin + UFSCC (32 threads)**
Qlock (10 processes)	2 2	12m 5s	OOM (after 31m 9s)	OOM (after 27m 49s)
Anderson (9 processes)	2 2	3m 49s	OOM (after 25m 39s)	OOM (after 21m 41s)
MCS (6 processes)	4 4 4 4 2	13h 24m 35s	23s	47s
TAS (13 processes)	3 3 3	10h 11m	17s	33s

Outline

- Introduction
- Background
- $L + 1$-layer divide & conquer approach to leads-to model checking
- All counterexample state generation at once
- Layer configuration selection
- Conclusion

Motivation

- Maude LTL model checker returns only one counterexample at one time.
- Generating all counterexample states one by one from a state at each intermediate layer in Maude is expensive because we need to update the specification of a system under verification on the fly frequently.
- Moreover, this is one main task in our tool that can be optimized to improve the running performance of our tool.

All counterexample state generation at once

- We propose a technique to generate all counterexample states at once in model checking.
- The technique is based on Tarjan algorithm to find all Strongly Connected Components (SCCs) in the product automaton of a system automaton and a property automaton.
- We develop a new model checker to support the technique in Maude by using facilities existing in Maude LTL model checker.

Experiments with our new model checker

Protocol	Layers	Parallel $L+1$-DCA2L2MC (8 workers)	Parallel $L+1$-DCA2L2MC (New) (8 workers)	Percentage Improvement (%)
Qlock (10 processes)	2 2	1h 14m	56m	24%
Anderson (9 processes)	2 2	21m	17m	19%
MCS (6 processes)	8 8	5d 6h 34m	5d 3h 26m	2.5 %
TAS (13 processes)	3 3	3d 8h 36m	2d 11h 31m	26 %

Experiments with our new model checker

Protocol	Layers	$L+1$-DCA2L2MC	$L+1$-DCA2L2MC (New)	Percentage Improvement (%)
Qlock (10 processes)	2 2	50.35s	2.26s	95.5%
Anderson (9 processes)	2 2	20.99s	2.26s	89.2%
MCS (6 processes)	8 8	17h 11m	65m	93.7 %
TAS (13 processes)	3 3	25m 11s	30.64s	97.95 %

Sequential $L+1$-DCA2L2MC with our new model checker for generating states and counterexample states only.

Outline

- Introduction
- Background
- $L + 1$-layer divide & conquer approach to leads-to model checking
- All counterexample state generation at once
- Layer configuration selection
- Conclusion

Layer configuration selection

We propose an approach to finding a good layer configuration and an analysis tool supporting the approach as follows:

- We use a small depth for each layer (e.g., in a range of 1 - 4), start with a 2-layer configuration, and measure the time to generate states up to the final layer.
- Randomly select some states and cx-states at the current final layer to conduct some trial model checking experiments in parallel and roughly estimate the verification in total.
- If the estimated verification is not good enough and the time to generate states up to the current final layer is not large, we will use one more layer and keep on doing it until a good layer configuration is found. Otherwise, the current one is a good layer configuration candidate.

Parallel $L + 1$-DCA2L2MC with different layer configurations

Protocol	**Layers**	**Parallel $L + 1$-DCA2L2MC (8 workers)**
Qlock (10 processes)	2 2	42m 36s
	2 2 2	2d 11h 31m
Anderson (9 processes)	2 2	12m 58s
	2 2 2	1h 36m 13s
MCS (6 processes)	4 4 4 4 2	22h 13m 45s
	4 4 4 4 2 2	2d 12h 47m
TAS (13 processes)	3 3 3	1d 12h 33m
	3 3 3 3	1d 13h 53m

Parallel $L + 1$-DCA2L2MC with different layer configurations

Protocol	Layers	Parallel $L + 1$-DCA2L2MC (8 workers)
Qlock (10 processes)	2 2	42m 36s
	2 2 2	2d 11h 31m
Anderson (9 processes)	2 2	12m 58s
	2 2 2	1h 36m 13s
MCS (6 processes)	4 4 4 4 2	22h 13m 45s
	4 4 4 4 2 2	2d 12h 47m
TAS (13 processes)	3 3 3	1d 12h 33m
	3 3 3 3	1d 13h 53m

Limitations

- If there are long lasso loops in the specification, the sub-state spaces at the final layer are not much smaller than the original state space.
  - We need to come up with a way to handle long lasso loops in the specification but it is a non-trivial task.
- If the specification has a symmetry in the graph search, many duplicated parts are likely to be shared by many sub-state spaces at the final layer as in the TAS case study.
  - We may use the symmetric reduction technique to remove the symmetric parts in the graph search on the fly with our technique/tool.

Outline

- Introduction
- Background
- $L + 1$-layer divide & conquer approach to leads-to model checking
- All counterexample state generation at once
- Layer configuration selection
- Conclusion

Conclusion

- We have proposed some techniques to mitigate the state space explosion and improve the running performance of model checking for leads-to properties to some extent by parallelization.
- For the reader of interest, we have extended our technique to handle not only leads-to properties but also any LTL properties in the following work.

  A Tableau-based Approach to Model Checking Linear Temporal Properties (just accepted for publication at ICFEM 2024)

Thank you for your attention!