INF5140 – Specification and Verification of Parallel Systems

Spring 2017

Institutt for informatikk, Universitetet i Oslo

April 7, 2017

Sat-based & Bounded model checking

Model checking

$$S \models^? \varphi$$

- origin [Clarke and Emerson, 1982][1] & [Queille and Sifakis, 1982]
- S (model of the) system,
- φ: formula in a suitable logic
  - LTL
  - CTL, CTL*, modal μ-calculus
  - ...
- ultimately a fancy "graph exploration problem" (with *big* graphs)

[1] the conference was 1981, the book was published 1982

Advantages of MC

- no proofs, “push button”
- diagnostic counterexamples
- logics used for MC can express many concurrency problems

Main “disadvantage”

- state space explosion problem (aka state explosion problem)
- problem “solution” space grows *exponential* is the problem “description” space
  - notably reachable state space exponential in the number of processes

The 4 big breakthroughs combatting the SSEP

Apart from

- advances in data structures,
- software engineering,
- tricks, optimizations, heuristics and
- general advances in processing power/memory.

Clarke identifies the following

"big 4" breakthroughs

1. symbolic techniques (notably using BDDs)[a]
2. partial order reduction
3. bounded model checking
4. CEGAR, localisation reduction [Kurshan, 1993] [Clarke et al., 2010] [Clarke et al., 2000]

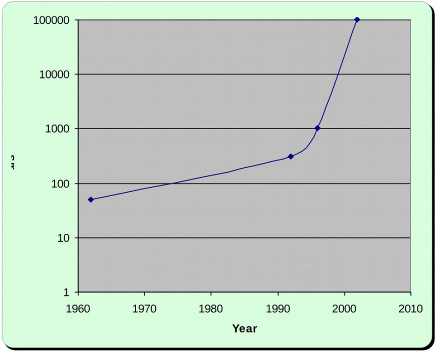
[a] See later presentations

SAT

- (boolean) satisfiability
- famous, prototypical NP-complete problem
-

SAT solver progress

- highly competitive field
- yearly "SAT-competition"[2]



taken from [Clarke, 2017]

[2] http://www.satcompetition.org/

Bounded model checking

- Origin: [Biere et al., 2009] (see also [Biere et al., 2003])

BMC starting point

Leverage sat-solving, a powerful a successful technique, to do model checking

Cf.: Symbolic model checking and BDDs

- See separate presentation
- successful technique
- used (most prominently for HW) in industrial uses of MC
- Two ingredients of SMC
  - operating symbolically on representation of sets of states
  - use *BDDs* (= specific kind of graph representation of *boolean* functions) to represent and operate on them
- like SMC/BDD-based MC: BMC based on “boolean encodings”

Bad news: the MC problem/reachability is *not* a SAT problem :-(

- MC *here*:[3]
  - models are kind of transition systems/Kripke structures ...
  - spec's are "temporal logic" formulas

solving an MC problem

It all boils down to some form of fancy graph reachability

- "reachability", however:
  - a form of "fixpoint" calculation[4]
  - fixpoints are emphatically not part of boolean logic.[5]

[3] The term "model checking", i.e., solving $M \models^? \varphi$ can be applied in different settings as well. A boolean assignment can be seen as *model* of a propositional formula, for instance. That *is* of course a SAT problem. But we are interested transition systems satisfying a TL formula.

[4] see also the presentation about μ-calculus.

[5] They are not even part of first-order logic. Implicitly they are part in temporal logics, though (eventually, until etc.)

Good news: *bounded* MC can be seen as SAT :-)

less ambitious goal

Can I find an error (conterexample) in the behavior of the system considering up-to k steps from the initial states

- price to pay: no more "verification"[6]
- bug-hunting
- simple core idea

[6] but MC is typically verification of a model/abstraction anyhow and/or verification up until the MC runs out of time/memory.

LTL and "existential" LTL

- remember: LTL (linear time temporal logic) and definition of

$$S \models \varphi$$

- φ must hold for all paths of S
- If $S \not\models \varphi$ (error), then exists a paths π such that $\pi \not\models \varphi$

For explicitness' sake

path quantifiers[a]

$$\forall\varphi \quad \text{and} \quad \exists\varphi$$

[a] one single quantifier as prefix to an LTL formula.

- assume NNF

Terminology: witnesses

counterexample for

$$S \models \Box p \quad \text{corresponding to} \quad S \models \forall \Box p$$

corresponds the question if there exists a witness[7]

$$\Diamond \neg p$$

- Goal: find finite (fixed bound) prefixes as witness to an existential model checking problem (LTL)
- conceptually easy if original $\forall \varphi$ is a safety prop.
- complications due to loops

[7] in logics in general, a witness is a thing (here a path) that gives (constructive) evidence to an existential formula

Paths with and without loops

No loop

$$\begin{tikzcd} \bullet \arrow[r] & \bullet_{s_i} \arrow[r] & \bullet \arrow[r] & \bullet_{s_k} \end{tikzcd}$$

- only prefix with back loop can be witness for $\Box p$

(k, l)-loop

$$\begin{tikzcd} \bullet \arrow[r] & \bullet_{s_l} \arrow[r] & \bullet \arrow[r] & \bullet_{s_i} \arrow[r] & \bullet_{s_k} \arrow[lll, bend left] \end{tikzcd}$$

Given: TS/Kripke-structure. transition relation $\longrightarrow$.

Definition

Assume $l \leq k$. A path π is a (k, l)-loop if $\pi_k \longrightarrow \pi_l$ and

$$\pi = u \cdot v^\omega$$

with

$$u = \pi_0 \ldots \pi_{l-1} \quad \text{and} \quad v = \pi_l \ldots \pi_k$$

A path π is a k-loop if there exists an l with $0 \leq l \leq k$ s.t. π is a (k, l)-loop

- remember: paths π are (infinite) sequences of "states" (worlds)[8]
- loops here is about those states (not "edges" of the picture)

[8] Earlier we also used σ as symbol

Bounded semantics

- remember the "normal" semantics of LTL from before, relating formulas and paths
- $[\![\varphi]\!]$ or $\pi \models \varphi$
- now: the new "looping paths" (k-loops) as basis for bounded semantics, i.e., basis for BMC
- note: "finite" prefixes (loops) can give information for infinite paths, thus serve as witnesses
- boundes semantics for path

  with loop: "unchanged"

  without loop: be aware of the cut-off and be pessimistic

Bounded semantics: for loops

Definition

Let π be a k-loop. A formula φ is *valid* along π *with bound* k, written

$$\pi \models_k \varphi \ ,$$

iff $\pi \models \varphi$.

Bounded semantics: without loops

Definition

Let π be a path which is *not* a k-loop. Then an LTL formula φ is *valid along π with bound k*, written

$$\pi \models_k \varphi \ ,$$

iff $\pi \models_k^0 \varphi$, given below.

- earlier $\pi \models \varphi$, corresponding here to $\models^0$
- k is treated as "cut-off":
- what comes *afterward*: unknown
- if in doubt: "false", i.e., the path is not valid/does not satisfy the formula in the bounded manner
  - for $\bigcirc$: don't "look" beyond k
  - for $\Box$: be pessimistic
  - for $\Diamond$: positive answer at least possible within the bound

Bounded semantics: without loops ($\models_k^i$)

$$
\begin{array}{lll}
\pi \models_k^i p & \text{iff} & p \in L(\pi_i) \\
\pi \models_k^i \neg p & \text{iff} & p \notin L(\pi_i) \\
\pi \models_k^i \varphi_1 \wedge \varphi_2 & \text{iff} & \pi \models_k^i \varphi_1 \quad \text{and} \quad \pi \models_k^i \varphi_2 \\
\pi \models_k^i \varphi_1 \vee \varphi_2 & \text{iff} & \pi \models_k^i \varphi_1 \quad \text{or} \quad \pi \models_k^i \varphi_2
\end{array}
$$

$$
\begin{array}{lll}
\pi \models_k^i \Box\varphi & & \text{is always false} \\
\pi \models_k^i \Diamond\varphi & \text{iff} & \exists j.i \leq j \leq k. \quad \pi \models_k^j \varphi \\
\pi \models_k^i \bigcirc\varphi & \text{iff} & i < k \quad \text{and} \quad \pi \models_k^{i+1} \varphi \\
\pi \models_k^i \varphi_1 U \varphi_2 & \text{iff} & \exists j, i \leq j \leq k. \pi \models_k^j \varphi_2 \text{ and } \forall n, i \leq n < j. \pi \models_k^n \varphi_1 \\
\pi \models_k^i \varphi_1 R \varphi_2 & \text{iff} & \exists j, i \leq j \leq k. \pi \models_k^j \varphi_1 \text{ and } \forall n, i \leq n < j. \pi \models_k^n \varphi_2
\end{array}
$$

Bounded → unbounded semantics

- Note, the connection is done for existential LTL (formulas of the form $\exists\varphi$, not like $\forall\varphi$)
- unbounded semantics as limit of the bounded ones (for all/arbitrary bounds k)

Lemma (Easy direction (per path))

$$\pi \models_k \varphi \quad \textit{implies} \quad \pi \models \varphi$$

Lemma (For TSs/KSs)

$$S \models \exists\varphi \quad \textit{implies} \quad S \models_k \exists\varphi \qquad \textit{for some } k \geq 0$$

Theorem

$$S \models \exists\varphi \quad \textit{iff} \quad S \models_k \exists\varphi \qquad \textit{for some } k \geq 0$$

BMC via SAT

- so far:
  - definition of the bounded MC problem
  - we convinced ourself: BMC approximates MC (at least for existential path formulas)
- Now: reduce to sat-solving

Goal

$[\![S, \varphi]\!]_k$ is *satisfiable* iff π is a witness for φ

  - sat-problems: formula with (propositional variables)
- encoding given in 3 parts. given k
  1. valid initial path for S and
  2. satisfaction of formula if
     - there's a loop or
     - there's no loop

Kripke-structure/transition system

In the modal logics chapter, mild variation in terminology and choice of symbols. [Baier and Katoen, 2008] called (basically) the same things *transition systems* avoiding the word Kripke structure

Definition (Kripke structure)

A Kripke structure is a tuple $(S, I, \longrightarrow, L)$ where S is the set of states, $I \subseteq S$ the set of initial states, $\longrightarrow \subseteq S \times S$ the transition relation, and $L : S \to 2^{AP}$ the labelling function.

- transition relation: a predicate:[9] $\longrightarrow : S^2 \to Bool$
- initial states: a predicate $I : S \to Bool$

[9][Biere et al., 2003] write $T(s_1, s_2)$ for our infix relational notation $s_1 \longrightarrow s_2$, where T is the transition relation predicate.

1^{st} component: Translating S

- remember transition system/Kripke stuctures
  - states s_i. Consider s_i as *variables*
  - transition relation: as predicate $T(s_k, s_l)$, we write still infix $s_k \longrightarrow s_l$
- unfolding of the transition relation

$$[\![S]\!]_k \triangleq I(s_0) \wedge \bigwedge_{i=0}^{k-1} s_i \longrightarrow s_{i+1} \tag{1}$$

- states in KS: *propositional variables* s_k
- beware of the role of index/subscript in equation (1)

Loop condition

- Remember the def. of (k, l)-loop

$$\begin{tikzcd} \bullet \arrow[r] & \bullet_{s_l} \arrow[r] & \bullet \arrow[r] & \bullet_{s_i} \arrow[r] & \bullet_{s_k} \arrow[lll, bend left] \end{tikzcd}$$

- simple abbreviation

$$_{l}L_k \triangleq s_k \longrightarrow s_l$$

- loop condition holds[10] iff there is a back loop from a state s_k back to a previous state s_l (which can be s_k)

Definition (Loop condition)

$$L_k \triangleq \bigvee_{l=0}^{k} {}_{l}L_k$$

[10] resp. it will hold when applied to a path consisting of a sequence of states s_i, which are considered as propositional variables, as said. the word "back" makes sense only if one interprets the variables to be "in a sequence".

a rather unsurprising definition: define "successor"

$succ(i)$ of i in a (k, l)-loop as

- $succ(i) = i + 1$ for $i < k$
- $succ(i) = l$ for k

2^{nd} component: translating formula with a loop

propositional part: boring

$$
\begin{array}{rcl}
{}_l[\![p]\!]_k^i & \triangleq & p(s_i) \\
{}_l[\![\neg p]\!]_k^i & \triangleq & \neg p(s_i) \\
{}_l[\![\varphi_1 \wedge \varphi_2]\!]_k^i & \triangleq & {}_l[\![\varphi_1]\!]_k^i \wedge {}_l[\![\varphi_2]\!]_k^i \\
{}_l[\![\varphi_1 \vee \varphi_2]\!]_k^i & \triangleq & {}_l[\![\varphi_1]\!]_k^i \vee {}_l[\![\varphi_2]\!]_k^i
\end{array}
$$

Cont'd

Actually straightforward

- loop $\rightarrow$ no cut-off $\rightarrow$ "standard semantics"
- remember *unrolling* of fixpoints[11]

temporal part: a bit more interesting

$$
\begin{array}{rcl}
{}_I[\![\Box\varphi]\!]^i_k & \triangleq & {}_I[\![\varphi]\!]^i_k \wedge {}_I[\![\Box\varphi]\!]^{succ(i)}_k \\
{}_I[\![\Diamond\varphi]\!]^i_k & \triangleq & {}_I[\![\varphi]\!]^i_k \vee {}_I[\![\Diamond\varphi]\!]^{succ(i)}_k \\
{}_I[\![\bigcirc\varphi]\!]^i_k & \triangleq & {}_I[\![\varphi]\!]^{succ(i)}_k \\
{}_I[\![\varphi_1 U \varphi_2]\!]^i_k & \triangleq & {}_I[\![\varphi_1]\!]^i_k \vee {}_I[\![\varphi_1 U \varphi_2]\!]^{succ(i)}_k \\
{}_I[\![\varphi_1 R \varphi_2]\!]^i_k & \triangleq & {}_I[\![\varphi_2]\!]^i_k \wedge {}_I[\![\varphi_1 R \varphi_2]\!]^{succ(i)}_k
\end{array}
$$

[11] Cf. also the presentation about the μ-calculus. Also in the construction of the Büchi-automaton from an LTL formula, that unrolling played a role (for U).

Translation without a loop

- same principles
- “index” I not needed
- instead of the more complex $succ(i)$: simply $i + 1$.
- otherwise: the definition stays “the same”)

3^{rd} component: translating formula without a loop

Inductive case $\forall i \leq k$:

propositional part: boring again

$$
\begin{array}{rcl}
[\![p]\!]_k^i & \triangleq & p(s_i) \\
[\![\neg p]\!]_k^i & \triangleq & \neg p(s_i) \\
[\![\varphi_1 \wedge \varphi_2]\!]_k^i & \triangleq & [\![\varphi_1]\!]_k^i \wedge [\![\varphi_2]\!]_k^i \\
[\![\varphi_1 \vee \varphi_2]\!]_k^i & \triangleq & [\![\varphi_1]\!]_k^i \vee [\![\varphi_2]\!]_k^i
\end{array}
$$

Loop-case (cont'd)

Inductive case $\forall i \leq k$:

temporal part: a bit more interesting

$$
\begin{aligned}
[\![\Box\varphi]\!]_k^i &\triangleq [\![\varphi]\!]_k^i \wedge [\![\Box\varphi]\!]_k^{i+1} \\
[\![\Diamond\varphi]\!]_k^i &\triangleq [\![\varphi]\!]_k^i \vee [\![\Diamond\varphi]\!]_k^{i+1} \\
[\![\bigcirc\varphi]\!]_k^i &\triangleq [\![\varphi]\!]_k^{i+1} \\
[\![\varphi_1 U \varphi_2]\!]_k^i &\triangleq [\![\varphi_1]\!]_k^i \vee [\![\varphi_1 U \varphi_2]\!]_k^{i+1} \\
[\![\varphi_1 R \varphi_2]\!]_k^i &\triangleq [\![\varphi_2]\!]_k^i \wedge [\![\varphi_1 R \varphi_2]\!]_k^{i+1}
\end{aligned}
$$

- base case: $[\![\varphi]\!]_k^{k+1} \triangleq \mathit{false}$

Putting it together

$$
\begin{aligned}
[\![S,\varphi]\!]_k \triangleq\ & [\![S]\!]_k \wedge \\
& (\ (\neg L_k \wedge [\![\varphi]\!]_k^0) \\
& \quad \vee \quad (\bigvee_{i=0}^{k} (\neg_l L_k \wedge\ {}_l[\![\varphi]\!]_k^0) \quad)
\end{aligned}
\tag{2}
$$

Theorem

$$[\![S,\varphi]\!]_k \mathit{satisfiable} \quad \mathit{iff} \quad S \models_k \exists\varphi\ .$$

Further info

- The technical slides here recap parts of the journal article [Biere et al., 2003] by the inventors of BMC
- BMC for software [Kroening et al., 2004]
- Survey [Prasad et al., 2005]
-

References I

[Baier and Katoen, 2008] Baier, C. and Katoen, J.-P. (2008).
Principles of Model Checking.
MIT Press.

[Biere et al., 2009] Biere, A., Cimatti, A., Clarke, E. M., Fujita, M., and Zhu, Y. (2009).
Symbolic model checking using SAT procedures instead of BDDs.
In *Proceedings of DAC'09: Design Automation Conference*, pages 317–320. ACM.

[Biere et al., 2003] Biere, A., Cimatti, A., Clarke, E. M., Strichman, O., and Zhu, Y. (2003).
Bounded model checking.
Advances in Computers, 58.

[Clarke et al., 2000] Clarke, E., Grumberg, O., Jha, S., Lu, Y., and Veith, H. (2000).
Counterexample-guided abstraction refinement.
In Emerson, E. A. and Sistla, A. P., editors, *Proceedings of the 12th International Conference on Computer-Aided Verification (CAV '00)*, volume 1855 of *Lecture Notes in Computer Science*, pages 154–169. Springer Verlag.

[Clarke et al., 2010] Clarke, E. C., Kurshan, R. P., and Veith, H. (2010).
The localization reduction and counter-example guided abstraction refinement.
In Manna, Z. and Peled, D., editors, *Pnueli Festschrift*, volume 6200 of *Lecture Notes in Computer Science*, pages 61–71. Springer Verlag.

[Clarke, 2008] Clarke, E. M. (2008).
Model checking – my 27-year quest to overcome the state explosion problem.
In Cervesato, I., Veith, H., and Voronkov, A., editors, *Logic for Programming, Artificial Intelligence, and Reasoning: 15th International Conference, LPAR 2008, Doha, Qatar, November 22-27, 2008. Proceedings*, Lecture Notes in Artificial Intelligence, pages 182–182. Springer Verlag.

References II

[Clarke, 2017] Clarke, E. M. (2017).
SAT-based bounded and unbounded model checking.
Available electronically on the net.
Data of publication unknown.

[Clarke and Emerson, 1982] Clarke, E. M. and Emerson, E. A. (1982).
Design and synthesis of synchronisation skeletons using branching time temporal logic specifications.
In Kozen, D., editor, *Proceedings of the Workshop on Logic of Programs 1981*, volume 131 of *Lecture Notes in Computer Science*, pages 244–263. Springer Verlag.

[Kroening et al., 2004] Kroening, D., Lerda, F., and Clarke, E. (2004).
Bounded model checking for software.
In Jensen, K. and Podelski, A., editors, *Proceedings of TACAS 2004*, volume 2988 of *Lecture Notes in Computer Science*. Springer Verlag.

[Kurshan, 1993] Kurshan, R. P. (1993).
Automata Theoretic Verification of Coordinating Processes.
Princeton University Press.

[Prasad et al., 2005] Prasad, M. R., Biere, A., and Gupta, A. (2005).
A survey of recent advances in sat-based formal verification.
International Journal on Software Tools for Technology Transfer, 7(2):156–173.

[Queille and Sifakis, 1982] Queille, J. P. and Sifakis, J. (1982).
Specification and verification of concurrent systems in CESAR.
In Dezani-Ciancaglini, M. and Montanari, U., editors, *Proceedings of the 5th International Symposium on Programming 1981*, volume 137 of *Lecture Notes in Computer Science*, pages 337–351. Springer Verlag.