Hoare logic and Model checking

Part II: Model checking

Lecture 11: Implementing model checking

Jean Pichon-Pharabod
University of Cambridge

CST Part II – 2020/21

Definite temporal models

For the model checker to be effective, the input temporal model needs to be effective.

A **definite temporal model**:

$$\begin{aligned}
&\mathsf{DTModel} \in \mathsf{Set}\\
&\mathit{DM}, \ldots \in \mathsf{DTModel} \overset{\mathit{def}}{=}\\
&\quad (S \in \mathsf{Set}) \times\\
&\qquad (F \in \mathsf{Fintype}\ S) \times\\
&\qquad\quad (S_0 \in S \to \mathbb{B}) \times\\
&\qquad\qquad (\textcircled{\scriptsize 1}\ T\ \textcircled{\scriptsize 2} \in S \to S \to \mathbb{B}) \times\\
&\qquad\qquad\quad (\ell \in S \to \mathit{AP} \to \mathbb{B}) \times\\
&\qquad\qquad\qquad (\forall s \in S.\, \exists s' \in S.\, s\ T\ s' = \top_{\mathbb{B}})
\end{aligned}$$

Specifying a CTL model checker

We will see how to implement the world's worst CTL model checker:

$$\mathsf{mc} \in (\mathit{AP} \in \mathsf{Set}) \to \mathsf{DTModel}\ \mathit{AP} \to \mathsf{StateProp}^{\mathsf{CTL}}\ \mathit{AP} \to \mathbb{B}$$

which has the following specification:

$$\begin{aligned}
&\forall \mathit{AP} \in \mathsf{Set}, \mathit{DM} \in \mathsf{DTModel}\ \mathit{AP}, \psi^{\mathsf{CTL}} \in \mathsf{StateProp}^{\mathsf{CTL}}\ \mathit{AP}.\\
&\quad \mathsf{reflect}\ (\mathsf{mc}\ \mathit{AP}\ \mathit{DM}\ \psi^{\mathsf{CTL}})\ \ (\mathit{DM} \vDash^{\mathsf{WI}}_{\mathit{AP}} \psi^{\mathsf{CTL}})
\end{aligned}$$

where satisfaction in a definite model is as expected.

Defining a CTL model checker

To check whether the model satisfies a property, it suffices to check whether all the initial states satisfy the property, which we check using an auxiliary function *mca* that checks whether a state satisfies a given property.

$$\begin{aligned}
&\mathsf{mc}\ \mathit{AP}\ \mathit{DM}\ \psi^{\mathsf{CTL}} \overset{\mathit{def}}{=}\\
&\quad \mathsf{forall\text{-}fin}\ \mathit{DM}.S\ (s \mapsto \mathit{DM}.S_0\ s \to_{\mathbb{B}} \mathsf{mca}\ \mathit{DM}\ \psi^{\mathsf{CTL}}\ s)
\end{aligned}$$

$$\begin{aligned}
&\mathsf{mca} \in (\mathit{AP} \in \mathsf{Set}) \to (\mathit{DM} \in \mathsf{DTModel}\ \mathit{AP}) \to\\
&\qquad\qquad \mathsf{StateProp}^{\mathsf{CTL}}\ \mathit{AP} \to (\mathit{DM}.S \to \mathbb{B})
\end{aligned}$$

This *mca* function works by recursion on the proposition, calling itself on the sub-propositions.

CTL model checker: propositional fragment

$$\begin{aligned}
\text{mca } \mathit{AP}\ \mathit{DM}\ p &\overset{\mathit{def}}{=} s \mapsto \mathit{DM}.\ell\ s\ p \\
\text{mca } \mathit{AP}\ \mathit{DM}\ (\hat{\neg}\phi^{\text{CTL}}) &\overset{\mathit{def}}{=} \text{let } V = \text{mca } \mathit{AP}\ \mathit{DM}\ \phi^{\text{CTL}} \text{ in} \\
&\phantom{\overset{\mathit{def}}{=}}\ s \mapsto \neg_{\mathbb{B}}(V\ s) \\
\text{mca } \mathit{AP}\ \mathit{DM}\ (\phi_1^{\text{CTL}}\ \hat{\wedge}\ \phi_2^{\text{CTL}}) &\overset{\mathit{def}}{=} \text{let } V_1 = \text{mca } \mathit{AP}\ \mathit{DM}\ \phi_1^{\text{CTL}} \text{ in} \\
&\phantom{\overset{\mathit{def}}{=}}\ \text{let } V_2 = \text{mca } \mathit{AP}\ \mathit{DM}\ \phi_2^{\text{CTL}} \text{ in} \\
&\phantom{\overset{\mathit{def}}{=}}\ s \mapsto V_1\ s \wedge_{\mathbb{B}} V_2\ s \\
\text{mca } \mathit{AP}\ \mathit{DM}\ (\phi_1^{\text{CTL}}\ \hat{\vee}\ \phi_2^{\text{CTL}}) &\overset{\mathit{def}}{=} \text{let } V_1 = \text{mca } \mathit{AP}\ \mathit{DM}\ \phi_1^{\text{CTL}} \text{ in} \\
&\phantom{\overset{\mathit{def}}{=}}\ \text{let } V_2 = \text{mca } \mathit{AP}\ \mathit{DM}\ \phi_2^{\text{CTL}} \text{ in} \\
&\phantom{\overset{\mathit{def}}{=}}\ s \mapsto V_1\ s \vee_{\mathbb{B}} V_2\ s \\
\text{mca } \mathit{AP}\ \mathit{DM}\ (\phi_1^{\text{CTL}}\ \hat{\to}\ \phi_2^{\text{CTL}}) &\overset{\mathit{def}}{=} \text{let } V_1 = \text{mca } \mathit{AP}\ \mathit{DM}\ \phi_1^{\text{CTL}} \text{ in} \\
&\phantom{\overset{\mathit{def}}{=}}\ \text{let } V_2 = \text{mca } \mathit{AP}\ \mathit{DM}\ \phi_2^{\text{CTL}} \text{ in} \\
&\phantom{\overset{\mathit{def}}{=}}\ s \mapsto V_1\ s \to_{\mathbb{B}} V_2\ s
\end{aligned}$$

CTL model checker: next

If we know in which states ϕ^{CTL} holds, then we know in which states X ϕ^{CTL} holds: their predecessors:

$$\begin{aligned}
&\text{mca } \mathit{AP}\ \mathit{DM}\ (\text{A X } \phi^{\text{CTL}}) \overset{\mathit{def}}{=} \\
&\quad \text{let } V = \text{mca } \mathit{AP}\ \mathit{DM}\ \phi^{\text{CTL}} \text{ in} \\
&\quad s \mapsto \text{forall-fin } \mathit{DM}.S\ (s' \mapsto (s\ \mathit{DM}.T\ s' \to_{\mathbb{B}} V\ s'))
\end{aligned}$$

$$\begin{aligned}
&\text{mca } \mathit{AP}\ M\ (\text{E X } \phi^{\text{CTL}}) \overset{\mathit{def}}{=} \\
&\quad \text{let } V = \text{mca } \mathit{AP}\ \mathit{DM}\ \phi^{\text{CTL}} \text{ in} \\
&\quad s \mapsto \text{exists-fin } \mathit{DM}.S\ (s' \mapsto s(\mathit{DM}.T\ s' \wedge_{\mathbb{B}} V\ s'))
\end{aligned}$$

CTL model checker: small paths 1/2

The remaining temporal operators talk about infinite paths.
But it is sufficient to consider paths smaller than the diameter of the model[1]:

$$\begin{aligned}
\text{IsSmallPathFrom} \in\ & (\mathit{AP} \in \text{Set}) \to (\mathit{DM} \in \text{DTModel } \mathit{AP}) \to \mathit{DM}.S \to \\
& \quad \text{list } \mathit{DM}.S \to \text{Prop}
\end{aligned}$$

$$\begin{aligned}
&\text{IsSmallPathFrom } \mathit{AP}\ \mathit{DM}\ s\ \Pi \overset{\mathit{def}}{=} \\
&\quad (\text{length } \Pi \leq \text{size } \mathit{DM}.F) \wedge (\text{nth } \Pi\ 0 = \text{some } s) \wedge \\
&\quad (\text{nth } \Pi\ (\text{length } \Pi - 1) = \text{some } s') \wedge (s'\ \mathit{DM}.T\ s) \wedge \\
&\quad \left(\forall n \in \mathbb{N}, s', s''.\ \begin{pmatrix} \text{nth } \Pi\ n = \text{some } s' \wedge \\ \text{nth } \Pi\ (n+1) = \text{some } s'' \end{pmatrix} \to s'\ \mathit{DM}.T\ s'' = \top_{\mathbb{B}}\right)
\end{aligned}$$

[1] reminiscent of the pumping lemma for automata.

CTL model checker: small paths 2/2

And we can obtain all these paths:

$$\begin{aligned}
\text{small-paths-from} \in\ & (\mathit{AP} \in \text{Set}) \to (\mathit{DM} \in \text{DTModel } \mathit{AP}) \to \\
& \quad (s \in \mathit{DM}.S) \to \\
& \qquad \text{Fintype } (\text{SmallPathFrom } \mathit{AP}\ \mathit{DM}\ s) \\
\text{small-paths-from} &\overset{\mathit{def}}{=} \ldots
\end{aligned}$$

CTL model checker: generally

For the ‘generally’ temporal operator, we need to look at lasso-shaped paths that are made up of a loop and a (possibly empty) path that leads to that loop, and check that all the states of this lasso satisfy the sub-property:

$$
\begin{array}{l}
\text{mca } \mathit{AP}\ \mathit{DM}\ (\text{A G } \phi^{\text{CTL}}) \stackrel{\mathit{def}}{=} \\
\quad \text{let } V = \text{mca } \mathit{AP}\ \mathit{DM}\ \phi^{\text{CTL}} \text{ in} \\
\quad s \mapsto \ \ \text{forall-fin} \\
\qquad\qquad (\text{small-paths-from } \mathit{AP}\ \mathit{DM}\ s) \\
\qquad\qquad (\Pi \mapsto \text{forall-list } \Pi\ (s' \mapsto V\ s')) \\
\text{mca } \mathit{AP}\ \mathit{DM}\ (\text{E G } \phi^{\text{CTL}}) \stackrel{\mathit{def}}{=} \\
\quad \text{let } V = \text{mca } \mathit{AP}\ \mathit{DM}\ \phi^{\text{CTL}} \text{ in} \\
\quad s \mapsto \ \ \text{exists-fin} \\
\qquad\qquad (\text{small-paths-from } \mathit{AP}\ \mathit{DM}\ s) \\
\qquad\qquad (\Pi \mapsto \text{forall-list } \Pi\ (s' \mapsto V\ s'))
\end{array}
$$

CTL model checker: future

$$\text{mca } \mathit{AP}\ \mathit{DM}\ (\text{A F } \phi^{\text{CTL}}) \stackrel{\mathit{def}}{=} \ldots$$

$$\text{mca } \mathit{AP}\ \mathit{DM}\ (\text{E F } \phi^{\text{CTL}}) \stackrel{\mathit{def}}{=} \ldots$$

Left as an exercise.

CTL model checker: until

$$
\begin{array}{l}
\text{mca } \mathit{AP}\ \mathit{DM}\ (\text{A } (\phi_1^{\text{CTL}} \text{ U } \phi_2^{\text{CTL}})) \stackrel{\mathit{def}}{=} \\
\quad \text{let } V_1 = \text{mca } \mathit{AP}\ \mathit{DM}\ \phi_1^{\text{CTL}} \text{ in} \\
\quad \text{let } V_2 = \text{mca } \mathit{AP}\ \mathit{DM}\ \phi_2^{\text{CTL}} \text{ in} \\
\quad s \mapsto \left(\begin{array}{l}
\text{forall-fin (small-paths-from } \mathit{AP}\ \mathit{DM}\ s) \\
\quad \left(\begin{array}{l}
\Pi \mapsto \\
\quad \left(\begin{array}{l}
\text{existi } \Pi \\
\quad \left(\begin{array}{l}
j\ s'' \mapsto \\
\quad \left(\begin{array}{l}
(\text{foralli } \Pi\ (i\ s' \mapsto j <_{\mathbb{B}} i \to_{\mathbb{B}} V_1\ s') \\
\wedge_{\mathbb{B}} V_2\ s''
\end{array}\right)
\end{array}\right)
\end{array}\right)
\end{array}\right)
\end{array}\right)
\end{array}
$$

$$\text{mca } \mathit{AP}\ \mathit{DM}\ (\text{E } (\phi_1^{\text{CTL}} \text{ U } \phi_2^{\text{CTL}})) \stackrel{\mathit{def}}{=} \ldots$$

Left as an exercise.

Actually implementing model checking

This is not very efficient!

In practice,

- the Vs are memoised;
- “symbolic model checking” uses binary decision diagrams (IB Logic and proof) to represent sets of states, and performs operations on sets-as-BDDs, instead of explicitly manipulating the sets;
- the states can be computed lazily;
- “partial order reduction” tries to not enumerate redundant interleavings;
- ...
- 40+ years of tricks!

Counterexamples

Generating counterexamples

Adapted from “Tree-Like Counterexamples in Model Checking”.

If the specification is not satisfied, and is in ACTL, then we can do better than just say “no”: we can produce a counterexample.

The idea is that $M \nvDash_{AP} \psi^{\text{ACTL}}$ is equivalent to $M \vDash_{AP} \neg\psi^{\text{ACTL}}$, which is itself equivalent to nf-model $M \vDash_{AP}$ nf-neg^{s} $AP\ \psi^{\text{ACTL}}$, where the latter formula is (the embedding of a proposition) in ECTL: it suffices to find a witness of that ECTL proposition.

Shape of ECTL witnesses

The shape of an ECTL witness:

$W, \ldots \in$ data Witness $(AP \in \text{Set})\ (M \in \text{TModel}\ AP) \in$ Set $:=$
wap $\in M.S \to$ Witness $AP\ M$
| wand $\in$ Witness $AP\ M \to$ Witness $AP\ M \to$ Witness $AP\ M$
| winjl $\in$ Witness $AP\ M \to$ Witness $AP\ M$
| winjr $\in$ Witness $AP\ M \to$ Witness $AP\ M$
| wX $\in M.S \to M.S \to$ Witness $AP\ M \to$ Witness $AP\ M$
| wF $\in$ list $M.S \to$ Witness $AP\ M \to$ Witness $AP\ M$
| wG $\in$ list $(M.S \times$ Witness $AP\ M) \to$ Witness $AP\ M$
| wU $\in$ list $(M.S \times$ Witness $AP\ M) \to M.S \to$ Witness $AP\ M \to$
Witness $AP\ M$

Being an ECTL witness: atomic propositions

$= \vDash_{-} \equiv$ wit-by $\equiv$: $(AP \in \text{Set}) \to (M \in \text{TModel}\ AP) \to M.S \to$
$(\psi \in \text{StateProp}^{\text{CTL}}\ AP) \to$ Witness $AP\ M\ s \to$
Prop

There are (on purpose) no cases for A . . . .

A witness for an atomic proposition is just that the atomic proposition holds immediately:

$s \vDash_{AP,M} p$ wit-by $W \overset{\text{def}}{=} M.\ell\ s\ p \wedge W =$ wap $AP\ M\ s$

Being an ECTL witness: next

A witness for next is a transition from the current state, and a witness that the sub-property holds from the end state:

$$s \models_{AP,M} \text{E X } \psi \text{ wit-by } W \overset{\text{def}}{=}$$

$$\exists s' \in M.S, W' \in \text{Witness } AP\ M. \begin{pmatrix} s\ M.T\ s' \wedge \\ s' \models_{AP,M} \psi \text{ wit-by } W' \wedge \\ W = \text{wX } AP\ M\ s\ s'\ W' \end{pmatrix}$$

Being an ECTL witness: future

A witness for the ‘future’ temporal operator is a path that leads to a state for which we have a witness that it satisfies the sub-property:

$$\begin{array}{l} s \models_{AP,M} \text{E F } \psi \text{ wit-by } W \overset{\text{def}}{=} \\ \quad \exists s' \in M.S, \Pi \in \text{list } M.S, W' \in \text{Witness } AP\ M. \\ \quad \begin{pmatrix} \text{IsSmallPathFrom } AP\ M\ s\ \Pi \wedge \\ \text{last } \Pi = \text{some } s' \wedge \\ s' \models_{AP,M} \psi \text{ wit-by } W' \wedge \\ W = \text{wF } AP\ M\ s\ \Pi\ W' \end{pmatrix} \end{array}$$

Being an ECTL witness: generally

A witness for the ‘generally’ temporal operator is a lasso, for all the states of which we have a witness that they satisfy the sub-property:

$$\begin{array}{l} s \models_{AP,M} \text{E G } \psi \text{ wit-by } W \overset{\text{def}}{=} \\ \quad \text{let } T = (M.S \times \text{Witness } AP\ M) \text{ in} \\ \quad \exists X \in \text{list } T. \\ \quad \begin{pmatrix} \text{IsSmallPathFrom } AP\ M\ s\ X \wedge \\ (\exists i. (\text{last } T\ X)\ M.T\ (\text{nth } T\ X\ i)) \wedge \\ \begin{pmatrix} \forall i \in \mathbb{N}, s' \in M.S, W' \in \text{Witness } AP\ M\ s'. \\ \quad \begin{pmatrix} \text{nth } T\ X\ i = \text{some } \langle s', W' \rangle \to \\ \quad s' \models_{AP,M} \psi \text{ wit-by } W' \end{pmatrix} \end{pmatrix} \wedge \\ W = \text{wG } AP\ M\ X) \end{pmatrix} \end{array}$$

Being an ECTL witness: until

$$\begin{array}{l} s \models_{AP,M} \text{E } \psi_1 \text{ U } \psi_2 \text{ wit-by } W \overset{\text{def}}{=} \\ \quad \text{let } T = (M.S \times \text{Witness } AP\ M) \text{ in} \\ \quad \exists X \in \text{list } T, s' \in M.S, W' \in \text{Witness } AP\ M. \\ \quad \begin{pmatrix} \text{IsSmallPathFrom } AP\ M\ s\ (X +\!\!+ [\langle s', W' \rangle]) \wedge \\ \begin{pmatrix} \forall i \in \mathbb{N}, s'' \in M.S, W'' \in \text{Witness } AP\ M\ s'. \\ \quad \begin{pmatrix} \text{nth } T\ X\ i = \text{some } \langle s'', W''' \rangle \to \\ \quad s'' \models_{AP,M} \psi_1 \text{ wit-by } W'' \end{pmatrix} \end{pmatrix} \wedge \\ (s' \models_{AP,M} \psi_2 \text{ wit-by } W') \wedge \\ W = \text{wU } AP\ M\ X\ s'\ W') \end{pmatrix} \end{array}$$

Being an ECTL witness: conjunction

$$\begin{array}{l} s \vDash_{AP,M} \psi_1 \mathbin{\hat{\wedge}} \psi_2 \text{ wit-by } W \overset{\mathit{def}}{=} \\ \quad \exists W_1 \in \text{Witness } AP\ M, W_2 \in \text{Witness } AP\ M. \\ \quad \left(\begin{array}{l} s \vDash_{AP,M} \psi_1 \text{ wit-by } W_1 \wedge s \vDash_{AP,M} \psi_2 \text{ wit-by } W_2 \wedge \\ W = \text{wand } AP\ M\ W_1\ W_2 \end{array} \right) \end{array}$$

Being an ECTL witness: disjunction

$$\begin{array}{l} s \vDash_{AP,M} \psi_1 \mathbin{\hat{\vee}} \psi_2 \text{ wit-by } W \overset{\mathit{def}}{=} \\ \quad \left(\begin{array}{l} \exists W_1 \in \text{Witness } AP\ M. \\ \quad \left(\begin{array}{l} s \vDash_{AP,M} \psi_1 \text{ wit-by } W_1 \wedge \\ W = \text{winjl } AP\ M\ W_1 \end{array} \right) \end{array} \right) \vee \\ \quad \left(\begin{array}{l} \exists W_2 \in \text{Witness } AP\ M. \\ \quad \left(\begin{array}{l} s \vDash_{AP,M} \psi_2 \text{ wit-by } W_2 \wedge \\ W = \text{winjr } AP\ M\ W_2 \end{array} \right) \end{array} \right) \end{array}$$

Satisfiability and existence of witnesses

The requirement for a DTModel is just a brutal way to require M to be finite (otherwise, the witness could be infinite, and we would need a coinductive definition of a witness)

$$\begin{array}{l} \forall AP \in \text{Set}, M \in \text{TModel } AP, DM \in \text{DTModel } AP, \\ \quad s \in M.S, \psi \in \text{StateProp}^{\text{CTL}}\ AP. \\ \quad\quad \text{es } \psi \to \text{reflect-model } AP\ M\ DM \to \\ \quad\quad\quad \left(\begin{array}{l} (s \vDash^{\text{WI-S}}_{AP,M} \psi) \leftrightarrow \\ \left(\begin{array}{l} \exists W \in \text{Witness (split } AP\text{) (nf-model } AP\ M\text{)}. \\ \quad s \vDash_{(\text{split } AP),(\text{nf-model } AP\ M)} (\text{nf}^{s}\ AP\ \psi) \text{ wit-by } W \end{array} \right) \end{array} \right) \end{array}$$

Now, if we have $M \nvDash_{AP} \psi^{\text{ACTL}}$, there exists a corresponding W — and we can effectively find it by tweaking our model checking algorithm above (details elided).

Witnesses beyond ECTL

Can we have witnesses for more than just ECTL?

Yes, for example, one of the nice things about LTL is that counterexamples are just paths.

But if we look at fragments of CTL* that are to expressive, then these witnesses are often difficult to understand and use.

Instead, focus has been mostly on making better counterexamples for common subsets of ECTL.

Model checking LTL and CTL*

Requires a bit of machinery to check whether a state is visited infinitely often: Büchi automata.

We will not consider this further.

Summary

We saw a model checking algorithm for CTL, and sketched how it could be modified to generate counterexamples for ACTL formulas.

CEGAR
not examinable

CEGAR

Assume that we have a way to automatically generate abstract models. Then we can take the following approach: recursively:
pick an abstraction of the model
check the property in the abstract model
if it is true, happy
if it is false, is it a genuine counterexample?
try it on the base model: if it works, we have found a genuine counterexample
if it does not work, build an abstraction.

Model checking hybrid systems

Modelling physical systems is often best done with continuous variables. Is it possible apply model checking to these?

Yes! It has been done for example for ACAS X, the Next-Generation Airborne Collision Avoidance System `https://doi.org/10.1007/s10009-016-0434-1`

Summary

- How temporal models can be used to describe systems that evolve in time.
- How temporal logics (CTL*, etc.) can be used to specify those systems.
- How to use model checking in practice.
- How to relate a concrete temporal model to an abstract temporal model with simulation.
- How to implement model-checking for CTL, and counterexample generation for ACTL.