Hoare Logic and Model Checking

Model Checking
Lecture 12: Loose ends

Dominic Mulligan
Based on previous slides by Alan Mycroft and Mike Gordon

Programming, Logic, and Semantics Group
University of Cambridge

Academic year 2016–2017

Learning outcomes

By the end of this lecture, you should:

- Understand the relative expressive power of LTL and CTL
- Understand the relation between LTL, CTL, and CTL⋆
- Understand the state-space explosion problem
- Know a little about some common model-checking optimisations

LTL and CTL: a perspective

Relative expressivity of LTL and CTL

We have seen two widely used temporal logics

How do they compare?

Both have very different models of time

But perhaps there's a clever way of "compiling" one into the other?

Relative expressivity of LTL and CTL

Simple answer: LTL and CTL are **incomparable**

LTL is not a superset of CTL, nor is CTL a superset of LTL

There exist temporal properties:

- that can be expressed in LTL but not CTL,
- that can be expressed in CTL but not LTL,
- that can be expressed in both,
- that can be expressed in neither.

We will now examine these four cases

Properties expressible in LTL but not CTL

Consider the LTL formula $\Diamond p \rightarrow \Diamond q$

Intuitively: “all paths that have a p along them also have a q”

Property is inexpressible in CTL as all path formulae are “guarded”

Consider the candidate CTL formula: $\forall\Diamond\ p \rightarrow \forall\Diamond\ q$

Intuitively: “if all paths have a p, then all paths have a q”

Properties expressible in CTL but not LTL

Consider the CTL formula $\forall\Box\exists\Diamond\ p$

Intuitively: “in all future states, it’s always possible to reach p”

Property is inexpressible in LTL

Cannot express existence of states in LTL (generally)

A common subset of LTL and CTL

Consider the CTL formula $\forall\Box(p \rightarrow \forall\Diamond\ q)$

Consider the LTL formula $\Box(p \rightarrow \Diamond\ q)$

These express the same property

Intuitively: “any p is eventually followed by a q”

Warning:

- $\Box(p \rightarrow \Diamond\ q)$ obtained from $\forall\Box(p \rightarrow \forall\Diamond\ q)$ by dropping $\forall$,
- doing this does not always lead to equivalent formulae,
- $\forall\Diamond\forall\Box\Phi$ in CTL and $\Diamond\Box\phi$ in LTL are not equivalent!

Properties expressible in neither LTL and CTL

Consider the formula $\exists\Box\Diamond\, p$

Intuitively: “there is a path with infinitely many p”

This is not expressible in CTL or LTL

LTL cannot quantify over paths

Complex proof for this not being expressible in CTL

Question

If last formula not expressible in LTL or CTL, what is it expressible in?

CTL$\star$

Both LTL and CTL are fragments of another temporal logic: CTL$\star$

CTL$\star$ has both path and state formulae like CTL

But path formulae can refer to state formulae, and vice versa

CTL$\star$ grammar

Define CTL$\star$ state formulae by:

$$\begin{aligned}\Phi, \Psi &::= \top \mid \bot \mid p \\ &::= \Phi \wedge \Psi \mid \Phi \vee \Psi \mid \neg\Phi \\ &::= \forall\phi \mid \exists\psi\end{aligned}$$

Define CTL$\star$ path formulae by:

$$\begin{aligned}\phi, \psi &::= \Phi \mid \neg\phi \\ &::= \phi \wedge \psi \mid \phi \vee \psi \\ &::= \phi \;\texttt{UNTIL}\; \psi \mid \Box\phi \mid \Diamond\phi \mid \bigcirc\phi\end{aligned}$$

Similar to CTL: but Φ embedded in path formulae

LTL and CTL as fragments of CTL⋆

LTL formulae are:

- CTL⋆ path formulae ϕ where all state subformulae are atomic
- Preceded by a universal quantifier

CTL formulae are obtained by restricting CTL⋆ path formulae:

$$\phi, \psi ::= \bigcirc\Phi \mid \Box\Phi \mid \Diamond\Phi \mid \Phi \ \mathtt{UNTIL}\ \Psi$$

for Φ, Ψ state formulae

When to use LTL, when to use CTL?

CTL and LTL (and CTL⋆) have different expressive powers

So: when to use CTL, when to use LTL?

Answer: “it depends”:

- Depends on system being modelled
- Depends on properties we are interested in
- Our particular biases (debate similar to editor wars)

CTL advantages

Some say model checking CTL is computationally more efficient than LTL

Complexity of CTL model checking is linear, versus exponential for LTL

CTL model checking has been applied successfully in industry

If required to assert existence of states, or similar, use CTL

LTL advantages

More modern model checkers use linear time model

LTL often seen as more “intuitive” than CTL:

> *We found only simple CTL equations to be comprehensible; nontrivial equations are hard to understand and prone to error*
>
> Formal verification made easy, 1997

> *CTL is difficult to use for most users and requires a new way of thinking about hardware*
>
> On the fly model checking of RCTL formulas, 1998

Complexity, revisited

Note following theorem (Clarke and Draghicescu):

> *Let Φ be a CTL formula and ϕ an LTL formula obtained from Φ by deleting all path quantifiers.*
> *Then $\Phi \equiv \phi$ or there does not exist any LTL formula equivalent to Φ.*

Equivalent properties are expressed by shorter LTL formulae (if they exist)

In fact, formulae may be exponentially shorter in LTL than equivalents

So direct comparison of running times may be misleading

Debate rages on...

Abstraction

State space explosion problem

A key problem with model checking is "state space explosion" problem

As systems become larger, size of models can grow exponentially

Puts a limit on what systems are feasible to verify with today's computers

Motivates many optimisations to reduce size of models

Abstraction

Suppose we have a huge model—can we simplify it somehow?

One way of doing this is to use abstraction

Write $\mathcal{M} \preceq \mathcal{M}'$ when:

- To each step of $\mathcal{M}$ there is a corresponding step of $\mathcal{M}'$
- Atomic properties of $\mathcal{M}$ correspond to atomic properties of $\mathcal{M}'$

Intuitively, if $\mathcal{M} \preceq \mathcal{M}'$ then $\mathcal{M}'$ is "simpler" view of $\mathcal{M}$

If $\mathcal{M} \preceq \mathcal{M}'$ say $\mathcal{M}'$ is abstraction of $\mathcal{M}$

Example

Suppose $\mathcal{M}^1 = \langle S^1, S^1_0, \rightarrow^1, \mathcal{L}^1 \rangle$ and $\mathcal{M}^2 = \langle S^2, S^2_0, \rightarrow^2, \mathcal{L}^2 \rangle$ where:

- $S^2 \subseteq S^1$
- $S^2_0 = S^1_0$
- $s \rightarrow^2 t$ iff $s \rightarrow^1 t$ for all s, t in S^2
- $\mathcal{L}^1(s) = \mathcal{L}^2(s)$ for all s in S^2

and S^2 contains all reachable states in $\mathcal{M}$:

$$\text{for all } s \in S^2, \text{ for all } t \in S^1, s \rightarrow^1 t \text{ implies } t \in S^2$$

Then $\mathcal{M}^1 \preceq \mathcal{M}^2$

Note

Note in last example all $\mathcal{M}^1$ paths from initial states are $\mathcal{M}^2$ paths

Hence $\mathcal{M}^2 \models \phi$ implies $\mathcal{M}^1 \models \phi$

But now $\mathcal{M}^2 \models \phi$ is a simpler problem, as $\mathcal{M}^2$ is smaller model

Can this observation be generalised?

Simulations between models

Fix $\mathcal{M}^1 = \langle S^1, S^1_0, \rightarrow^1, \mathcal{L}^1 \rangle$ and $\mathcal{M}^2 = \langle S^2, S^2_0, \rightarrow^2, \mathcal{L}^2 \rangle$

Assume models are over same set of atomic propositions

Call a relation $H \subseteq S^1 \times S^2$ a simulation between $\mathcal{M}^1$ and $\mathcal{M}^2$ when:

- To each step of $\rightarrow^1$ there is a corresponding step of $\rightarrow^2$
- Steps lead to H-related states
- If $H\ s\ t$ then $s \rightarrow^1 s'$ implies there exists t' where $t \rightarrow^2 t'$ and $H\ s'\ t'$

Simulation preorder on models

Fix $\mathcal{M}^1 = \langle S^1, S^1_0, \rightarrow^1, \mathcal{L}^1 \rangle$ and $\mathcal{M}^2 = \langle S^2, S^2_0, \rightarrow^2, \mathcal{L}^2 \rangle$

Then $\mathcal{M}^1 \preceq \mathcal{M}^2$ if:

- There exists a simulation H between $\mathcal{M}^1$ and $\mathcal{M}^2$
- For all s in S^1_0, exists $s' \in S^2_0$ such that $H\ s\ s'$
- $\mathcal{L}^1(s) = \mathcal{L}^2(s')$ whenever $H\ s\ s'$

This is a preorder on models (i.e. reflexive and transitive)

ACTL subset of CTL

Define ACTL as the existential-free subset of CTL

Useful fragment of CTL

Example: $\forall\Box\forall\Diamond\, p$ — “can reach p from anywhere”

Theorem: if $\mathcal{M}^1 \preceq \mathcal{M}^2$ then $\mathcal{M}^2 \models \phi$ implies $\mathcal{M}^1 \models \phi$

Note: if $\mathcal{M}^2 \models \phi$ fails then not necessarily the case $\mathcal{M}^1 \models \phi$ fails

May be a “spurious” counterexample that does not hold in $\mathcal{M}^1$

Counter-example guided abstract refinement

CEGAR is a technique to automatically develop refinements

Suppose we are trying to show $\mathcal{M} \models \phi$

Then:

- Generate an initial abstraction of $\mathcal{M}$ called $\mathcal{M}'$
- Check whether $\mathcal{M}' \models \phi$
- If so, we are done per the above theorem
- Otherwise refine the model abstraction, and repeat

Microsoft’s SLAM device driver verifier uses CEGAR

Other optimisations

Symbolic model checking

Recall CTL model checking algorithm

Relies on *enumerative* presentation of transition system

System explicitly given by predecessor/successor set of each state

When models are huge, this is not efficient

How to make this more efficient?

Symbolic models

Idea: represent states, transition relations, and so on symbolically

Recall Ordered Binary Decision Diagrams from *Logic and Proof*:

- Canonical way of representing boolean functions
- Efficient operations over boolean functions

Represent model as a boolean formula, represented as an OBDD

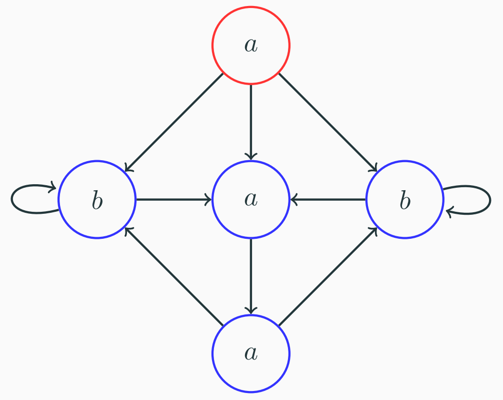
Alter CTL model checking algorithm to work with OBDDs

Model is now never explicitly enumerated, but represented implicitly

Very common optimisation in modern model checkers

Symmetry reduction example

Consider a model:



Note how model is symmetric vertically...

Symmetry reduction

More generally models may:

- Have repeated subcomponents
- Be symmetric
- Have similar repeated structures

If we could detect these, then could reduce size of model

Symmetries can be reduced in two ways:

- Requiring explicit annotations by user to spot symmetries
- Trying to use insights from group theory to automatically spot symmetries

Automatically spotting symmetries very hard, under active research

Other optimisations

Lots of active work making model checking:

- Faster
- Feasible on ever-larger models

Check out the Model Checking Competition and the Hardware Model Checking Competition

Represent state-of-the-art in model checking

Cutting edge optimisations implemented there, first

Course conclusion

Lecture 1: overview

You should know:

- importance of formal methods
- when to use model checking over other formal methods,
- the importance of temporal properties in system specification
- how to model systems as a transition system

LTL: overview

You should know:

- the linear model of time,
- LTL syntax and semantics
- LTL model checking is a language problem, and basics of automata-based model checking
- exponential time complexity for LTL model checking

CTL: overview

You should know:

- the branching model of time,
- CTL syntax and semantics,
- important CTL equivalences, and normal forms,
- CTL model checking is a reachability problem
- naïve recursive labelling algorithm for CTL
- linear time complexity for CTL model checking

Loose ends: overview

You should know:

- the incomparable expressiveness of LTL and CTL,
- LTL and CTL's relation to CTL$\star$
- CEGAR, abstraction, and a high-level view of other optimisations in model checking

The End!

(Good luck with the exam...)