Hoare Logic and Model Checking

Model Checking
Lecture 10: Computation Tree Logic (CTL)

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:

- Be familiar with the branching model of time
- Be familiar with CTL syntax and semantics
- Understand CTL semantic equivalence, and why it is important
- Be familiar with important CTL equivalences
- Be familiar with Existential Normal Form

Branching model of time

Branching time

CTL's conception of time:

- At each moment in time exactly potentially multiple futures
- Time "branches" into multiple futures at each state
- Quantify over possible futures

CTL therefore describes "state properties" of systems

CTL formulae describe states in transition system

A note on models

Note: by changing model of time, not changed underlying model

CTL models are based on right-serial transition systems, same as LTL

Changing conception of time:

- Affects properties that can be expressed by formulae
- Affects what CTL formulae describe (states, not paths)

CTL syntax

Atomic propositions

Like in LTL, we fix a set AP of **atomic propositions**

We continue to use p, q, r, and so on to range over AP

CTL state and path formulae

Define **state formulae** with the following grammar:

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

and **path formulae** with the following grammar:

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

In semantics of CTL:

- Path formulae are evaluated relative to a path
- State formulae are evaluated relative to a state

Intuitive explanation of CTL formulae

First line (of state formula grammar):

$$\top \mid \bot \mid p$$

$\top$, $\bot$, and p for p atomic are all primitive CTL state formulae

- $\top$ is the **logical truth** constant (or "true"),
- $\bot$ is the **logical falsity** constant (or "false"),
- p is the embedding of **atomic propositions** into CTL formulae

The last should now be familiar too!

Intuitive explanation of CTL formulae

Second line (of state formula grammar):

$$\neg\Phi$$

If Φ is a CTL state formula, then $\neg\Phi$ is a CTL state formula

- $\neg\Phi$ is **negation** of ϕ (or "not Φ")

Intuitive explanation of CTL formulae

Third line (of state formula grammar):

$$\Phi \wedge \Psi \mid \Phi \vee \Psi \mid \Phi \Rightarrow \Psi$$

If Φ and Ψ are CTL state formulae, then so are $\Phi \wedge \Psi$, $\Phi \vee \Psi$, $\Phi \Rightarrow \Psi$

- $\Phi \wedge \Psi$ is **conjunction** (or "Φ and Ψ")
- $\Phi \vee \Psi$ is **disjunction** (or "Φ or Ψ")
- $\Phi \Rightarrow \Psi$ is **implication** (or "if Φ then Ψ", or "ψ whenever ϕ")

Intuitive explanation of CTL formulae

Last line (of state formula grammar):

$$\forall\phi \mid \exists\phi$$

If ϕ and ψ are CTL path formulae, then $\forall\phi$ and $\exists\phi$ are CTL state formulae

- $\forall\phi$ is "ϕ along every path that starts here"
- $\exists\phi$ is "ϕ along at least one path that starts here", or "there exists a path where ϕ holds"

Specific to CTL!

Intuitive explanation of CTL formulae

Path formula grammar:

$$\bigcirc\Phi \mid \Box\Phi \mid \Diamond\Phi \mid \Phi \texttt{ UNTIL } \Psi$$

If Φ and Ψ are CTL state formulae, then $\bigcirc\Phi$, $\Box\Phi$, $\Diamond\Phi$, and $\Phi \texttt{ UNTIL } \Psi$ are CTL path formulae

- $\Box\Phi$ is "henceforth Φ", or "from now, always Φ"
- $\Diamond\Phi$ is "at some future point Φ"
- $\bigcirc\Phi$ is "immediately after Φ", or "in the next state Φ"
- $\Phi \texttt{ UNTIL } \Psi$ is "at some future point Ψ, but until then Φ"

Alternative syntax for modalities

Grammar above enforces path formula be "covered" by quantifier

Impossible to construct $\Box\forall\phi$ or $\exists\phi \texttt{ UNTIL } \Psi$

Effect is to have

$$\forall\Box\Phi \quad \exists\Box\Phi \quad \forall\Diamond\Phi \quad \exists\Diamond\Phi$$

$$\forall\bigcirc\Phi \quad \exists\bigcirc\Phi \quad \forall(\Phi \texttt{ UNTIL } \Psi) \quad \exists(\Phi \texttt{ UNTIL } \Psi)$$

$\forall\Box$, $\exists\bigcirc$, and so on, are "derived modalities"

Alternative syntax for modalities

Some collapse grammar of CTL into a single grammar of "formulae"

Less clear (to me, anyway) what is going on:

- $\forall$ and $\exists$ are instructions: "go off and examine paths"
- Path formulae evaluated relative to paths
- State formulae relative to states
- Grammar closer to grammar of CTL$\star$

Might also see (e.g. in "Logic in Computer Science"):

- A and E instead of $\forall$ and $\exists$
- X, G, F, and U instead of $\bigcirc$, $\Box$, $\Diamond$, and $\texttt{UNTIL}$

Operator precedence

We add parentheses freely to disambiguate

Assign precedence to reduce number of parentheses needed:

- Unary $\neg$, $\forall$, $\exists$, $\Box$, $\Diamond$, and $\bigcirc$ bind most tightly
- After that $\texttt{UNTIL}$
- After that $\vee$ and $\wedge$
- Finally $\Rightarrow$ binds least tightly

Precedence examples

So:

$$\Phi \Rightarrow \forall \bigcirc \Psi \quad \text{means} \quad \Phi \Rightarrow (\forall(\bigcirc\Psi))$$

$$\Phi \Rightarrow \Psi \vee \exists\Box\Psi \quad \text{means} \quad \Phi \Rightarrow (\Phi \vee (\exists(\Box\Psi)))$$

$$\forall \bigcirc \Phi \vee \Xi \Rightarrow \Psi \;\texttt{UNTIL}\; \Xi \quad \text{means} \quad ((\forall(\bigcirc\Phi)) \vee \Xi) \Rightarrow (\Psi \;\texttt{UNTIL}\; \Xi)$$

and so on...

Example CTL formulae

Suppose **`started`** and **`ready`** are atomic propositions, then:

$$\exists\Diamond(\texttt{started} \wedge \neg\texttt{ready})$$

can be read as:

> *it is possible to get to a state where "started" holds but "ready" does not*

Example CTL formulae

Suppose **`started`** and **`ready`** are atomic propositions, then:

$$\forall\Box\neg(\texttt{started} \wedge \neg\texttt{ready})$$

can be read as:

> *it is not possible to get to a state where "started" holds but "ready" does not*

Example CTL formulae

Suppose **`deadlock`** is an atomic proposition, then:

$$\forall\Diamond\forall\Box\texttt{deadlock}$$

can be read as:

> *the system will always progress to a state where it is henceforth permanently "deadlocked"*

Example CTL formulae

Suppose floor2, floor5, direction_up, and button_pressed_5 are atomic propositions, then:

$$\forall\Box(\texttt{floor2} \land \texttt{direction\_up} \land \texttt{button\_pressed\_5} \Rightarrow \forall(\texttt{direction\_up}\ \texttt{UNTIL}\ \texttt{floor5}))$$

can be read as:

> *A lift on the second floor travelling upwards will always continue to travel upwards until reaching level 5 whenever it contains passengers wishing to reach that floor*

Semantics of CTL

Making intuition precise

Previous examples:

- Showed examples of properties expressible in CTL,
- Provided intuition for meaning of CTL formulae

Time to make that intuition precise...

Models for CTL

Recall $\mathcal{M} = \langle S, S_0, \rightarrow, \mathcal{L} \rangle$, where:

- S set of **states**
- $S_0 \subseteq S$ set of **initial states**
- $\rightarrow \subseteq S \times S$ (right-serial) **transition relation** on S
- $\mathcal{L} : S \rightarrow \mathbb{P}(AP)$ **labelling function**

"Right serial" means $\forall s \in S . \exists s' \in S . s \rightarrow s'$

Infinite paths of states

Fix a CTL model $\mathcal{M} = \langle S, S_0, \rightarrow, \mathcal{L} \rangle$

Write $Paths(s)$ for set of infinite **paths** of S starting at s

Write $\pi[i]$ for i^{th} state of π (“indexing”)

Write π^i for suffix of π starting position i

Satisfaction at a state

Suppose $\mathcal{M}$ is a model, s is a state in $\mathcal{M}$, and Φ is a state formula

Define the **satisfaction relation** $s \models \Phi$ recursively by:

$s \models \top$	always
$s \models \bot$	never
$s \models p$	iff $p \in \mathcal{L}(s)$
$s \models \neg\Phi$	iff not $s \models \Phi$

Satisfaction at a state

$s \models \Phi \vee \Psi$	iff $s \models \Phi$ or $s \models \Psi$
$s \models \Phi \wedge \Psi$	iff $s \models \Phi$ and $s \models \Psi$
$s \models \Phi \Rightarrow \Psi$	iff not $s \models \Phi$ or if $s \models \Phi$ and $s \models \Psi$

Satisfaction at a state

$s \models \forall\phi$	iff $\pi \models \phi$ for every $\pi \in Paths(s)$
$s \models \exists\phi$	iff $\pi \models \phi$ for some $\pi \in Paths(s)$

$\pi \models \phi$ is the evaluation of path formula ϕ relative to a path π

Satisfaction along a path

Suppose $\mathcal{M}$ is a model, π is a path in $\mathcal{M}$, and ϕ is a path formula

Define the **satisfaction relation** $\pi \models \phi$ by:

$$\begin{aligned}
\pi &\models \bigcirc\Phi && \text{iff } \pi[1] \models \Phi \\
\pi &\models \Box\Phi && \text{iff } \pi[i] \models \Phi \text{ for all } i \\
\pi &\models \Diamond\Phi && \text{iff } \pi[i] \models \Phi \text{ for some } i \\
\pi &\models \Phi \texttt{ UNTIL } \Psi && \text{iff } \pi[i] \models \Psi \text{ for some } i \text{ and } \pi[j] \models \Phi \text{ for all } j < i
\end{aligned}$$

Notes on satisfaction relations

Two relations are mutually recursive—mutually recursive grammar

Satisfaction relation for path formulae similar to LTL relation

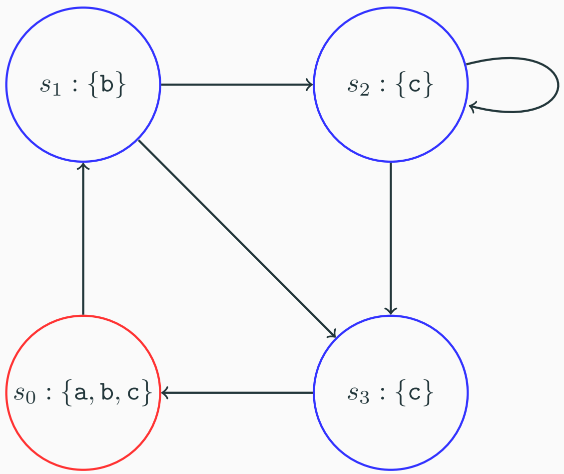
BUT:

- In LTL modality $\Box\phi$ uses all *suffixes* of path π
- In CTL modality $\Box\Phi$ uses all *indexes* of path π
- Similar for other modalities

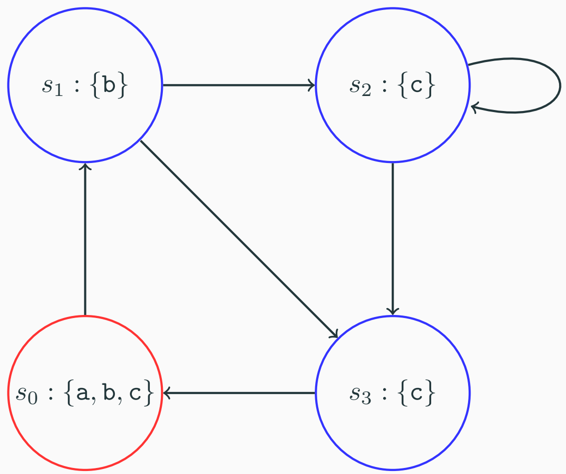
Tip: imagine types of π^i, $\pi[i]$ and satisfaction relations

Examples

CTL model as a picture:

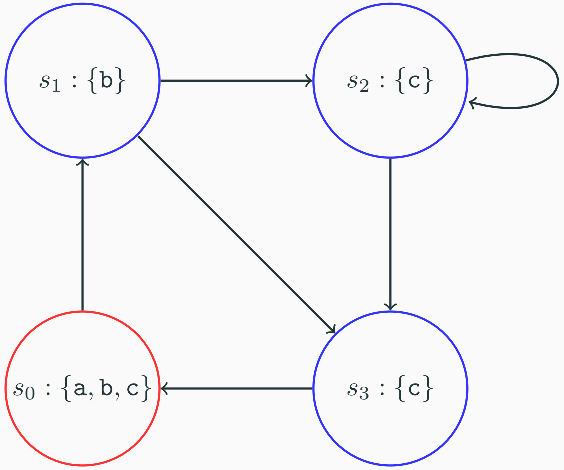


Examples



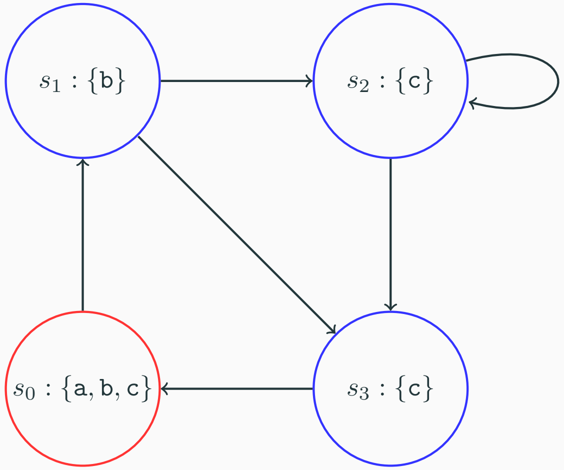
We have $s_0 \models \mathtt{a} \land \mathtt{b} \land \mathtt{c}$

Examples



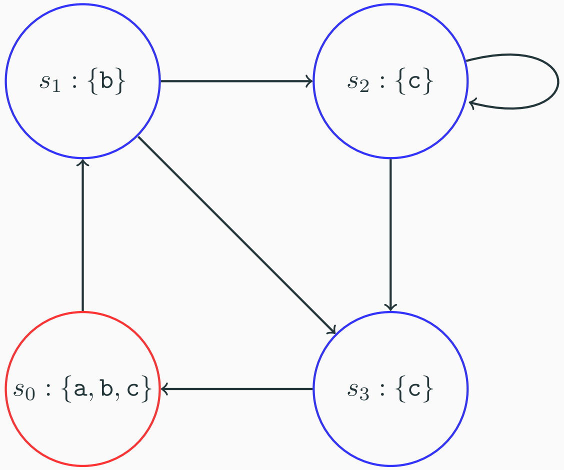
We have $s_0 \models \forall(\mathtt{b}\ \mathtt{UNTIL}\ \mathtt{c})$

Examples



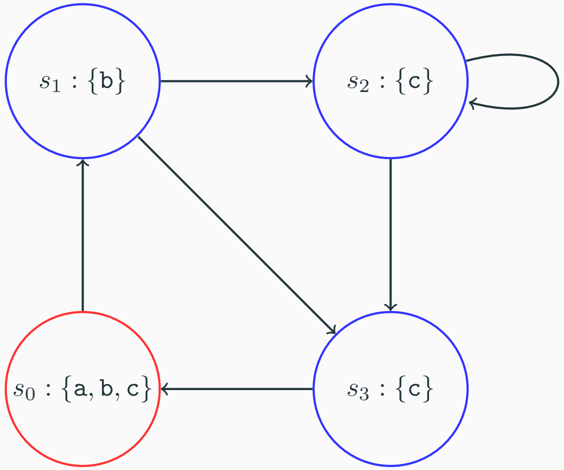
We have $s_1 \models \forall\bigcirc\mathtt{c}$

Examples



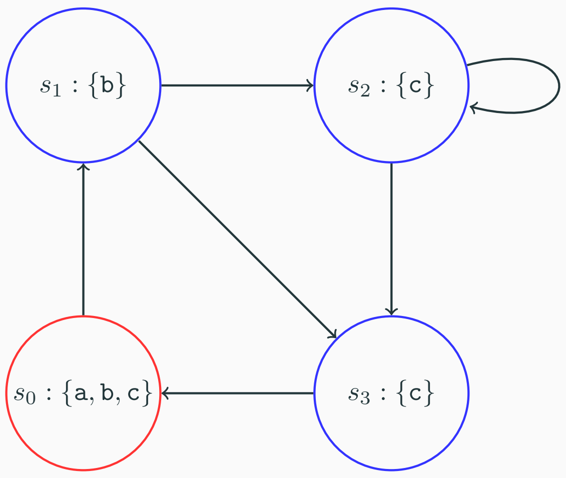
We have $s_1 \models \forall\bigcirc\forall\bigcirc\mathtt{c}$

Examples



We have $s_1 \models \exists\Diamond\mathtt{a}$

Examples



We have $s_2 \models \exists\Box\mathtt{c}$

Semantic equivalence

Satisfaction in model

Write $\mathcal{M} \models \Phi$ when $s \models \Phi$ for all states s in $\mathcal{M}$

Read $\mathcal{M} \models \Phi$ as "model $\mathcal{M}$ satisfies Φ"

Holds whenever all states of $\mathcal{M}$ satisfy Φ

Semantic equivalence

Say Φ and Ψ are semantically equivalent ($\Phi \equiv \Psi$) when:

$$\mathcal{M} \models \Phi \text{ if and only if } \mathcal{M} \models \Psi \text{ for all models } \mathcal{M}$$

Intuitively $\Phi \equiv \Psi$ asserts that:

- Φ and Ψ have same "semantic content"
- Safe to replace Φ with Ψ (and vice versa) in any context
- Quantifying over $\mathcal{M}$ means can't distinguish models

Properties of semantic equivalence

Semantic equivalence:

- Is reflexive ($\phi \equiv \phi$)
- Is symmetric ($\phi \equiv \psi$ implies $\psi \equiv \phi$)
- Is transitive ($\phi \equiv \psi$ and $\psi \equiv \xi$ implies $\phi \equiv \xi$)

Also is congruent with structure of formulae

Example: $\phi_1 \equiv \phi_2$ implies $\neg\phi_1 \equiv \neg\phi_2$ and $\exists \bigcirc \phi_1 \equiv \exists \bigcirc \phi_2$

Important semantic equivalences

$$\top \equiv \neg\bot$$

$$\Phi \Rightarrow \Psi \equiv \neg\Phi \vee \Psi$$

$$\Phi \vee \Psi \equiv \neg(\neg\Phi \wedge \neg\Psi)$$

Important semantic equivalences

$$\forall \bigcirc \Phi \equiv \neg\exists \bigcirc \neg\Phi$$

$$\forall\Box\Phi \equiv \neg\exists(\top \texttt{ UNTIL } \neg\Phi)$$

$$\forall\Diamond\Phi \equiv \forall(\top \texttt{ UNTIL } \Phi)$$

$$\forall(\Phi \texttt{ UNTIL } \Psi) \equiv \neg\exists(\neg\Psi \texttt{ UNTIL } (\neg\Phi \wedge \neg\Psi)) \wedge \neg\exists\Box\neg\Psi$$

$$\exists\Diamond\Phi \equiv \exists(\top \texttt{ UNTIL } \Phi)$$

Example proof

Task: show $\Phi \vee \Psi \equiv \neg(\neg\Phi \wedge \neg\Psi)$

Fix arbitrary model $\mathcal{M}$ and state s in $\mathcal{M}$

Need to show $s \models \Phi \vee \Psi$ if and only if $s \models \neg(\neg\Phi \wedge \neg\Psi)$

One direction

Assume $s \models \Phi \vee \Psi$

Then $s \models \Phi$ or $s \models \Psi$

Assume without loss of generality $s \models \Phi$

Then not $s \models \neg\Phi$

Hence not $s \models \neg\Phi \wedge \neg\Psi$

Therefore $s \models \neg(\neg\Phi \wedge \neg\Psi)$, as required

T'other

Assume $s \models \neg(\neg\Phi \wedge \neg\Psi)$

Then not $s \models \neg\Phi$ and $s \models \neg\Psi$

Then not (not $s \models \Phi$ and not $s \models \Psi$)

Hence either $s \models \Phi$ or $s \models \Psi$

Without loss of generality, assume $s \models \Phi$

Then $s \models \Phi \vee \Psi$, as required

Therefore $\Phi \vee \Psi \equiv \neg(\neg\Phi \wedge \neg\Psi)$

Existential Normal Form

Define formulae in Existential Normal Form (ENF) by:

$$\begin{aligned}\Phi, \Psi &::= \top \mid p \\ &::= \Phi \wedge \Psi \mid \neg\Phi \\ &::= \exists \bigcirc \Phi \mid \exists(\Phi \texttt{ UNTIL } \Psi) \mid \exists\Box\Phi\end{aligned}$$

Theorem:

Every state formula has an equivalent ENF formula

Proof: by structural induction, using previous semantic equivalences and congruences

Note proof is constructive: describes an algorithm

Summary

- CTL uses a branching model of time
- CTL state formulae express “state properties” of systems
- CTL semantics with respect to states in model
- Equivalence when formulae have same “semantic content”
- Can use equivalences to rewrite a formula into ENF