Computation Tree Logic (CTL)

Fazle Rabbi

University of Oslo, Oslo, Norway
Bergen University College, Bergen, Norway
fazlr@student.matnat.uio.no, Fazle.Rabbi@hib.no

May 30, 2015
Finite-state systems are modeled by labeled state transition graphs, called *Kripke Structures*.

**Example**

![State Transition Graph or Kripke Model](image)

**Figure**: State Transition Graph or Kripke Model

Formally, a *Kripke structure* is a triple $M = \langle S, R, L \rangle$, where

- $S$ is the set of states,
- $R \subseteq S \times S$ is the transition relation, and
- $L : S \rightarrow \mathcal{P}(AP)$ gives the set of atomic propositions true in each state.
If some state is designated as the initial state, the structure can be unwound into an infinite tree with that state as the root.

**Example (Unwind State Graph to obtain Infinite Tree)**

A path in $M$ is an infinite sequence of states, $\pi = s_0, s_1, \ldots$ such that for $i \geq 0, (s_i, s_{i+1}) \in R$. 
Motivation of using CTL

LTL formulas are evaluated on paths. A state of a system satisfies an LTL formula if all paths from the given state satisfy it. Thus, LTL implicitly quantifies universally over paths. Properties which assert the existence of a path cannot be expressed in LTL.

Example

From any state it is possible to get to the Restart state.
Motivation of using CTL

Computation Tree Logic (CTL) is a branching-time logic, meaning that its model of time is a tree-like structure in which the future is not determined; there are different paths in the future, any one of which might be the ‘actual path that is realised.

In CTL, as well as the temporal operators $X$, $F$, $G$ and $U$ of LTL we also have quantifiers $A$ and $E$ which express ‘all paths’ and ‘exists a path’, respectively.

Example (Each computation tree has the state $s_0$ as its root)

\[
\begin{align*}
M, s_0 &\models EF\ g \\
M, s_0 &\models AF\ g \\
M, s_0 &\models EG\ g \\
M, s_0 &\models AG\ g
\end{align*}
\]
Syntax of CTL

Definition

CTL formulas are inductively defined via a Backus Naur form

\[ \phi ::= \top | \bot | p | (\neg \phi) | (\phi \land \phi) | (\phi \lor \phi) | (\phi \rightarrow \phi) | AX\phi | EX\phi | AF\phi | EF\phi | AG\phi | EG\phi | A[\phi U \phi] | E[\phi U \phi] \]

where \( p \) ranges over a set of atomic formulas.

Notice that each of the CTL temporal connectives is a pair of symbols.

there exists an execution \( E \) for all execution \( A \)

Q

T

X (next)

F (finally)

G (globally)

U (until)

(and possibly others)
Semantics of computation tree logic

Let $M = (S, \rightarrow, L)$ be a model. For a CTL formula $\phi$, the relation $M, s \models \phi$ is defined by structural induction on $\phi$:

1. $M, s \models \top$ and $M, s \not\models \bot$
2. $M, s \models p$ iff $p \in L(s)$
3. $M, s \models \neg \phi$ iff $M, s \not\models \phi$
4. $M, s \models \phi_1 \land \phi_2$ iff $M, s \models \phi_1$ and $M, s \models \phi_2$
5. $M, s \models \phi_1 \lor \phi_2$ iff $M, s \models \phi_1$ or $M, s \models \phi_2$
6. $M, s \models \phi_1 \rightarrow \phi_2$ iff $M, s \not\models \phi_1$ or $M, s \models \phi_2$
7. $M, s \models AX \phi$ iff for all $s_1$ such that $s \rightarrow s_1$ we have $M, s_1 \models \phi$
8. $M, s \models EX \phi$ iff for some $s_1$ such that $s \rightarrow s_1$ we have $M, s_1 \models \phi$
9. $M, s \models AG \phi$ holds iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$, where $s_1$ equals $s$, and all $s_i$ along the path, we have $M, s_i \models \phi$
10. $M, s \models EG \phi$ holds iff there is a path $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$, where $s_1$ equals $s$, and all $s_i$ along the path, we have $M, s_i \models \phi$
11. $M, s \models AF \phi$ holds iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$, where $s_1$ equals $s$, there is some $s_i$ such that $M, s_i \models \phi$
12. $M, s \models EF \phi$ holds iff there is a path $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$, where $s_1$ equals $s$, and for some $s_i$ along the path, we have $M, s_i \models \phi$
13. $M, s \models A[\phi_1 U \phi_2]$ holds iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$, where $s_1$ equals $s$, that path satisfies $\phi_1 U \phi_2$
14. $M, s \models E[\phi_1 U \phi_2]$ holds iff there is path $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$, where $s_1$ equals $s$, and that path satisfies $\phi_1 U \phi_2$
Example of CTL formula

Example

- There is a reachable state satisfying \( q \): this is written \( EF\; q \).
- From all reachable states satisfying \( p \), it is possible to maintain \( p \) continuously until reaching a state satisfying \( q \): \( AG(p \rightarrow E[p \lor q]) \).
- Whenever a state satisfying \( p \) is reached, the system can exhibit \( q \) continuously for evermore: \( AG(p \rightarrow EG\; q) \).
- There is a reachable state from which all reachable states satisfy \( p \): \( EF\; AG\; p \).
LTL and CTL have different expressive powers. The choice between LTL and CTL depends on the application and the personal preferences.

- For example, there is no CTL formula that is equivalent to the LTL formula $\Diamond \Box p$.
- Likewise, there is no LTL formula that is equivalent to the CTL formula $\text{AG}(\text{EF } p)$.
- The disjunction $\Diamond \Box p \lor \text{AG}(\text{EF } p)$ is a CTL* formula that is not expressible in either CTL or LTL.
Expressive Power

There is no CTL formula that is equivalent to the LTL formula $\Diamond \Box p$.

$M, s_0 \models_{\text{LTL}} \Diamond \Box a$

$M, s_0 \not\models_{\text{CTL}} \text{AF (AG} \ a\text{)}$
Expressive Power

There is no LTL formula that is equivalent to the CTL formula $\text{AG(}\text{EF } p)$. This is shown by contradiction: assume $\varphi \equiv \text{AG(}\text{EF } p)$; let:

- $M \models \text{AG(}\text{EF } p)$, and thus- by assumption- $M \models \varphi$
- Paths($M'$) $\subseteq$ Paths($M$), Thus $M' \models \varphi$
- But $M' \not\models \text{AG(}\text{EF } p)$ as path $s^w \not\in \text{G(}\text{EF } p)$
Equivalences between CTL formulas

Definition

Two CTL formulas $\phi$ and $\psi$ are said to be semantically equivalent if any state in any model which satisfies one of them also satisfies the other; we denote this by $\phi \equiv \psi$.

Example

- $\neg AF\phi \equiv EG\neg\phi$
- $\neg EF\phi \equiv AG\neg\phi$
- $\neg AX\phi \equiv EX\neg\phi$
- $AF\phi \equiv A[T \ U \ \phi]$
- $EF\phi \equiv E[T \ U \ \phi]$
Adequate sets of CTL connectives

There are ten basic CTL operators:

- $AX$ and $EX$
- $AF$ and $EF$
- $AG$ and $EG$
- $AU$ and $EU$
- $AR$ and $ER$

Each of the ten operators can be expressed in terms of three operators $EX$, $EG$, and $EU$:

- $AX \phi = \neg EX(\neg \phi)$
- $EF \phi = E[\top U \phi]$
- $AG \phi = \neg EF(\neg \phi)$
- $AF \phi = \neg EG(\neg \phi)$
- $A[\phi U \psi] \equiv \neg E[\neg \psi U (\neg \phi \land \neg \psi)] \land \neg EG \neg \psi$
- $A[\phi R \psi] \equiv \neg E[\neg \phi U \neg \psi]$
- $E[\phi R \psi] \equiv \neg A[\neg \phi U \neg \psi]$
INPUT: a model $M = (S, \rightarrow, L)$ and a CTL formula $\phi$.
OUTPUT: the set of states of $M$ which satisfy $\phi$.

First, we convert $\phi$ with the adequate sets of CTL connectives (i.e., $\neg$, $\lor$, $EX$, $EU$, $EG$).
Next, label the states of $M$ with the subformulas of $\phi$ that are satisfied there, starting with the smallest subformulas and working outwards towards $\phi$.
Suppose $\psi$ is a subformula of $\phi$ and states satisfying all the immediate subformulas of $\psi$ have already been labelled. We determine by a case analysis which states to label with $\psi$. 
If $\psi$ is

- $\bot$: then no states are labelled with $\bot$.
- $p$: then label $s$ with $p$ if $p \in L(s)$.
- $\psi_1 \land \psi_2$: label $s$ with $\psi_1 \land \psi_2$ if $s$ is already labelled both with $\psi_1$ and with $\psi_2$.
- $\neg \psi_1$: label $s$ with $\neg \psi_1$ if $s$ is not already labelled with $\psi_1$.
- $EX\psi_1$: label any state with $EX\psi_1$ if one of its successors is labelled with $\psi_1$. 


If \( \psi \) is

- \( E[\psi_1 U \psi_2] \):
  - If any state \( s \) is labelled with \( \psi_2 \), label it with \( E[\psi_1 U \psi_2] \).
  - Repeat: label any state with \( E[\psi_1 U \psi_2] \) if it is labelled with \( \psi_1 \) and at least one of its successors is labelled with \( E[\psi_1 U \psi_2] \), until there is no change. This step is illustrated in Figure
If $\psi$ is $\text{EG}\psi_1$:

- Label all the states with $\text{EG}\psi_1$.
- If any state $s$ is not labelled with $\psi_1$, delete the label $\text{EG}\psi_1$.
- Repeat: delete the label $\text{EG}\psi_1$ from any state if none of its successors is labelled with $\text{EG}\psi_1$; until there is no change.
The complexity of the above mentioned algorithm is $O(|f| \times |S| \times (|S| + |R|))$, where $|f|$ is the number of connectives in the formula, $|S|$ is the number of states and $|R|$ is the number of transitions; the algorithm is linear in the size of the formula and quadratic in the size of the model.
A variant which is more efficient

For $EX$ and $EU$ we do as before (but take care to search the model by backwards breadth-first search, for this ensures that we wont have to pass over any node twice). For the $EG\psi$ case:

- Restrict the graph to states satisfying $\psi$, i.e., delete all other states and their transitions;
- Find the maximal strongly connected components (SCCs); these are maximal regions of the state space in which every state is linked with (= has a finite path to) every other one in that region.
- Use backwards breadth-first search on the restricted graph to find any state that can reach an SCC.

The complexity of this algorithm is $O(|f| \times (|S| + |R|))$, i.e., linear both in the size of the model and in the size of the formula.
Consider the dining philosopher problem. It is not very reasonable to assume that “One philosopher keeps eating forever.”

How can we rule out this behavior?
One solution is to use ‘Fairness Constraints’

**Fairness constraints rule out unrealistic executions**
**Fiarness (cont..)**

**Fairness Constraint**

- Sets of states (constraint) that must occur infinitely often along a computation path to be considered
- Usually described by a formula of the logic
- Fairness constraints restrict the path quantifiers (E and A) to fair paths
- **EF** $\phi$ holds at state $s$ only if there exists a fair path from $s$ along which $\phi$ holds
- **AG** $\phi$ holds at $s$ if $\phi$ holds in all states reachable from $s$ along fair paths.

If fairness constraints are interpreted as sets of states, then a fair path must contain an element of each fairness constraint infinitely often.
If fairness constraints are interpreted as CTL formulas, then a path is fair if each constraint is true infinitely often along the path.
In CTL fairness constraints cannot be expressed!

**Solution:**
- Impose Fairness Constraints on top of the Kripke Model.
- We call Fair Computation Paths those paths verifying a fairness constraint infinitely often.
- We call Fair Kripke Models those models restricted to fair paths.

Formally, a *fair Kripke structure* is a 4-triple $M = \langle S, R, L, F \rangle$, where $S, R, L$ are defined as before and $F \subseteq 2^S$ is a set of fairness constraints. Let $\pi = s_0, s_1, \ldots$ be a path in $M$. defined

$$\text{inf}(\pi) = \{ s \mid s = s_i \text{ for infinitely many } i \}$$

We say that $\pi$ is *fair* if and only if for every $P \in F$, $\text{inf}(\pi) \cap P \neq \emptyset$. 
Edmund M. Clarke Jr. and Orna Grumberg. 
*Model Checking.* 

Michael Huth and Mark Ryan 
Logic in Computer Science: Modelling and Reasoning about Systems. 
ISBN: 9780521543101
Thank You