Computation Tree Logic

Computation tree logic (CTL) is a branching-time logic that includes the propositional connectives as well as *temporal connectives* AX, EX, AU, EU, AG, EG, AF, and EF.

The syntax of CTL formulas is defined by the following recursive rules:

$$\begin{aligned}
\phi \quad ::= \quad & \top \mid \bot \mid P \mid (\neg\phi) \\
& \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi) \\
& \mid AX\phi \mid EX\phi \\
& \mid A[\phi U \phi] \mid E[\phi U \phi] \\
& \mid AG\phi \mid EG\phi \\
& \mid AF\phi \mid EF\phi \\
P \quad ::= \quad & p_1 \mid p_2 \mid p_3 \mid \ldots
\end{aligned}$$

Informally, A and E mean "along all paths" and "along at least one path," respectively; whereas X, F, G, and U refer to "next state," "some future state," "all future states," and "Until."

Semantics of CTL

We use structural induction on CTL formulas to define a satisfaction relation

$$\mathcal{M}, s \models \phi$$

that depends on a transition system $\mathcal{M} = (S, \rightarrow, L)$, a state s of $\mathcal{M}$, and a CTL formula ϕ:

1. $\mathcal{M}, s \models \top$ and $\mathcal{M}, s \not\models \bot$ for all states s.

2. $\mathcal{M}, s \models p$ iff $p \in L(s)$.

3. $\mathcal{M}, s \models \neg\phi$ iff $\mathcal{M}, s \not\models \phi$.

4. $\mathcal{M}, s \models \phi_1 \wedge \phi_2$ iff $\mathcal{M}, s \models \phi_1$ and $\mathcal{M}, s \models \phi_2$.

5. $\mathcal{M}, s \models \phi_1 \vee \phi_2$ iff $\mathcal{M}, s \models \phi_1$ or $\mathcal{M}, s \models \phi_2$.

6. $\mathcal{M}, s \models \phi_1 \rightarrow \phi_2$ iff $\mathcal{M}, s \not\models \phi_1$ or $\mathcal{M}, s \models \phi_2$.

7. $\mathcal{M}, s \models AX\phi$ iff $\mathcal{M}, s' \models \phi$ for *all* states s' with $s \rightarrow s'$.

8. $\mathcal{M}, s \models EX\phi$ iff $\mathcal{M}, s' \models \phi$ for *some* state s' with $s \rightarrow s'$.

9. $\mathcal{M}, s \models AG\phi$ iff for *all* paths $s = s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \cdots$ and *all* states s_i along the path, we have $\mathcal{M}, s_i \models \phi$.

10. $\mathcal{M}, s \models EG\phi$ iff there is *some* path $s = s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \cdots$ such that for *all* states s_i along the path, we have $\mathcal{M}, s_i \models \phi$.

11. $\mathcal{M}, s \models AF\phi$ iff for *all* paths $s = s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \cdots$ there is *some* state s_i along the path, for which $\mathcal{M}, s_i \models \phi$.

12. $\mathcal{M}, s \models EF\phi$ iff for *some* path $s = s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \cdots$ and *some* state s_i along the path, we have $\mathcal{M}, s_i \models \phi$.

13. $\mathcal{M}, s \models A[\phi_1 U \phi_2]$ iff for *all* paths $s = s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \cdots$ there is *some* state s_i along the path, such that $\mathcal{M}, s_i \models \phi_2$ and $\mathcal{M}, s_j \models \phi_1$, for all $j < i$.

14. $\mathcal{M}, s \models E[\phi_1 U \phi_2]$ iff for *some* path $s = s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \cdots$ there is *some* state s_i along the path, such that $\mathcal{M}, s_i \models \phi_2$ and $\mathcal{M}, s_j \models \phi_1$, for all $j < i$.

CTL Equivalences

Two CTL formulas ϕ and ψ are said to be *(semantically) equivalent*, written $\phi \equiv \psi$, if and only if

$$\mathcal{M}, s \models \phi \text{ iff } \mathcal{M}, s \models \psi$$

for all models $\mathcal{M}$ and all states s in $\mathcal{M}$.

The following basic equivalences illuminate the relation between temporal connectives:

$$\begin{aligned}
\neg AF\phi &\equiv EG\neg\phi && (1)\\
\neg EF\phi &\equiv AG\neg\phi && (2)\\
\neg AX\phi &\equiv EX\neg\phi && (3)\\
AF\phi &\equiv A[\top U\phi] && (4)\\
EF\phi &\equiv E[\top U\phi] && (5)
\end{aligned}$$

The following equivalences provide the theoretical basis for a model checking algorithm:

$$\begin{aligned}
AG\phi &\equiv \phi \wedge AX\, AG\phi\\
EG\phi &\equiv \phi \wedge EX\, EG\phi\\
AF\phi &\equiv \phi \vee AX\, AF\phi\\
EF\phi &\equiv \phi \vee EX\, EF\phi\\
A[\phi U\psi] &\equiv \psi \vee (\phi \wedge AX\, A[\phi U\psi])\\
E[\phi U\psi] &\equiv \psi \vee (\phi \wedge EX\, E[\phi U\psi])
\end{aligned}$$

Adequate Sets of Connectives

The basic equivalences for CTL formulas listed above, and the fact that propositional equivalences carry over to CTL formulas in general, imply that all CTL formulas can be expressed in equivalent form via a few selected connectives.

Theorem

> The set of connectives $\{\bot, \neg, \wedge, AU, EU, EX\}$ is adequate for CTL.

Another useful equivalence, to be be discussed in more detail later, is

$$A[pUq] \equiv \neg(E[\neg qU(\neg p \wedge \neg q)] \vee EG\neg q).$$

Theorem

> The sets of connectives $\{\bot, \neg, \wedge, EG, EU, EX\}$ and $\{\bot, \neg, \wedge, AF, EU, EX\}$ are adequate for CTL.

There are also other adequate sets for CTL.

Mutual Exclusion Example

We next formalize the properties expected for the mutual exlcusion model in CTL.

Safety:

$$AG\neg(c_1 \wedge c_2)$$

Liveness:

$$AG(t_1 \rightarrow AFc_1)$$

Non-blocking:

$$AG(n_1 \rightarrow EXt_1)$$

No strict sequencing:

$$EF(c_1 \wedge E[c_1 U(\neg c_1 \wedge E[\neg c_2 U c_1])])$$

CTL*

If we allow nested modalities and propositional connectives before applying path quantifiers, we obtain a logic called CTL* in which there are two classes of syntactic expressions.

State formulas are evaluated in states:

$$\phi \quad ::= \quad P \mid (\neg\phi) \mid (\phi \wedge \phi)$$

$$\mid A[\alpha] \mid E[\alpha]$$

whereas *path formulas* are evaluated along paths:

$$\alpha \quad ::= \quad \phi \mid (\neg\alpha) \mid (\alpha \wedge \alpha)$$

$$(\alpha U\, \alpha) \mid (G\alpha) \mid (F\alpha) \mid (X\alpha)$$

The variable P ranges over atomic descriptions, the variable α over path formulas; and the variable ϕ over state formulas.

Examples

$A[(pU\, r) \vee (qU\, r)]$: along all paths, either p is true until r, or q is true until r,

$A[Xp \vee XXp]$: along all paths, p is true in the next state, or the next but one.

$E[GFp]$: there is a path along which p is true infinitely often.

Note that the second and third formula can not be expressed in CTL.

Semantics of CTL*

In the case of CTL* we define satisfaction relations for both state and path formulas:

1. (a) $\mathcal{M}, s \models p$ iff $p \in L(s)$.
   (b) $\mathcal{M}, s \models \neg\phi$ iff $\mathcal{M}, s \not\models \phi$.
   (c) $\mathcal{M}, s \models \phi_1 \wedge \phi_2$ iff $\mathcal{M}, s \models \phi_1$ and $\mathcal{M}, s \models \phi_2$.
   (d) $\mathcal{M}, s \models A[\alpha]$ iff for *all* paths τ beginning at s we have $\mathcal{M}, \tau \models \alpha$.
   (e) $\mathcal{M}, s \models E[\alpha]$ iff for *some* path τ beginning at s we have $\mathcal{M}, \tau \models \alpha$.

2. (a) $\mathcal{M}, \pi \models \phi$ iff $\mathcal{M}, s_1 \models \phi$.
   (b) $\mathcal{M}, \pi \models \neg\alpha$ iff $\mathcal{M}, \pi \not\models \alpha$.
   (c) $\mathcal{M}, \pi \models \alpha_1 \wedge \alpha_2$ iff $\mathcal{M}, \pi \models \alpha_1$ and $\mathcal{M}, \pi \models \alpha_2$.
   (d) $\mathcal{M}, \pi \models X\alpha$ iff $\mathcal{M}, \pi^2 \models \alpha$.
   (e) $\mathcal{M}, \pi \models G\alpha$ iff, for all $i \geq 1$, $\mathcal{M}, \pi^i \models \alpha$.
   (f) $\mathcal{M}, \pi \models F\alpha$ iff, for some $i \geq 1$, $\mathcal{M}, \pi^i \models \alpha$.
   (g) $\mathcal{M}, \pi \models \alpha_1 U \alpha_2$ iff for some $i \geq 1$, $\mathcal{M}, \pi^i \models \alpha_2$ while $\mathcal{M}, \pi^j \models \alpha_1$ for all $j < i$.

where p ranges over atomic descriptions, α over path formulas, and ϕ over state formulas.

CTL, LTL, and CTL*

The language of CTL is a subset of CTL* (in which the definition of path formulas is more restricted).

Semantically in LTL one considers all paths, and hence an LTL formula α can be considered to be equivalent to $A[\alpha]$. In this sense LTL may also be viewed as a subset of CTL*

The following examples show the relation between these logics.

In CTL and LTL:

$AG(p \rightarrow AFq)$ in CTL or $G(p \rightarrow Fq)$ in LTL

In CTL but not in LTL:

$AG(EFp)$

In LTL but not in CTL:

$A[GFp \rightarrow Fq]$

Neither in CTL nor in LTL:

$E[GFp]$

Weak Until Operator in CTL

The until operator in CTL has been defined in such a way that whenever $A[p\,U\,q]$ is true in a state s then q must be true somewhere along each path from s.

Sometimes it is useful to consider a modified until operator W such that $p\,W\,q$ is deemed to be true even for paths along which q never holds, provided however that then p is always true.

In LTL and CTL* this *weak until* operator can be defined as follows:

$$p\,W\,q \equiv (p\,U\,q) \vee Gp.$$

A direct encoding in CTL is not possible, but we have the following equivalences for EW

$$E[p\,W\,q] \equiv E[p\,U\,q] \vee EGp.$$

and AW

$$A[p\,W\,q] \equiv \neg E[\neg q\,U\,\neg(p \vee q)].$$