Lecture 12

CTL and First-order Predicate Logic

February 17, 1998
Lecturer: Richard Ladner
Notes: Todd Millstein

12.1. Computational Tree Logic

12.1.1. CTL and CTL*

CTL includes two kinds of formulas. The *path* formulas talk about paths in a model. These formulas are closed under the linear-time logic operators X, U, and G. The *state* formulas talk about states in the model. These formulas are closed under the ordinary boolean connectives as well as two new operators, E ("exists") and A ("for all"). The semantics of these operators were given in the previous lecture.

In CTL, there is a restriction that each path operator in a formula must be immediately preceded by a state operator. This ensures that path formulas can never be evaluated with respect to a state, but only with respect to a path. CTL* drops this restriction, providing a semantics for evaluating a path formula with respect to a state.

As an example, consider the formula $AGEF\alpha$. This is in CTL because each path operator (G and F) is directly preceded by a state operator (A and E, respectively). The formula is true in model M at state s if from every state on every path from s (AG), there exists a path on which α is eventually true ($EF\alpha$).

On the other hand, consider the formula $AGF\alpha$. This is not in CTL because G directly precedes F, but it is in CTL*. The formula is true in model M at state s if from every state on every path from s (AG), α is eventually true ($F\alpha$) *on that same path*.

12.1.2. Complexity Results

The following table summarizes some complexity results for the temporal logics we have discussed.

	model checking	*satisfiability*
LTL	PSPACE-complete	PSPACE-complete
CTL	P	EXPTIME-complete
CTL*	PSPACE-complete	2-EXPTIME-complete

As we can see, the syntactic restriction imposed by CTL makes it tractable, as model-checking takes polynomial time instead of polynomial space. We will informally discuss this algorithm. Like other model-checking algorithms we've seen, it uses the *tableau* method.

We are given a formula α to check, a model $M = (S, \pi, R)$ (defined as in previous lectures), and a state s_0 in the model. Each state $s \in S$ has a set T_s of true formulas and F_s of false formulas. These sets initially contain the true and false variables at that state, respectively, and we use this information to derive truth values for all subformulas of α at each state.

As an example, here are the rules for deriving the truth value of $E(\beta U \gamma)$ in some state s:

- $\gamma \in T_s \longrightarrow E(\beta U \gamma) \in T_s$
- $\beta \in F_s, \gamma \in F_s \longrightarrow E(\beta U \gamma) \in F_s$
- $(\beta \in T_s, \exists s'((s, s') \in R, E(\beta U \gamma) \in T_{s'})) \longrightarrow E(\beta U \gamma) \in T_s$
- $(\gamma \in F_s, \forall s'((s, s') \in R \Rightarrow E(\beta U \gamma) \in F_{s'})) \longrightarrow E(\beta U \gamma) \in F_s$

There are similar rules for the ordinary boolean connectives and for operators of the form $EX\beta$, $AX\beta$, and $A(\beta U \gamma)$ (all other operators can be defined in terms of these operators). The algorithm applies the rules until no more can be applied and then checks whether $\alpha \in T_{s_0}$. This can be done in polynomial time.

There are two final comments about modal and temporal logics, regarding areas that we have not explored. First, although we have stressed algorithms such as the above one, there are also classical proof systems for K and CTL (and LTL? and CTL*?). Second, there is a close connection between modal and temporal logics and automata on infinite objects.

12.2. First-order Predicate Logic

12.2.1. Syntax

We will be studying first-order predicate logic with equality. That is, the symbol "=" always denotes equality. The basic elements of the logic are:

- individual variables $x, y, z, \ldots$ These can take on arbitrary values in some domain.
- function symbols $f, g, h, \ldots$ Each function has an associated arity. Note that a 0-ary function is simply a constant.
- relation symbols $R, S, T, \ldots$ Each relation has an associated arity. Note that the two 0-ary relations are simply the booleans *true* and *false*. Further, a unary relation is just a set.

¿From these primitives, we can build the syntax of the language inductively. *Terms* are defined as follows:

- Any variable is a term.

- If $t_1, t_2, \ldots, t_k$ are terms and f is a k-ary function symbol, then $f(t_1, t_2, \ldots, t_k)$ is a term.

From terms, we can define *formulas* in the logic:

- If $t_1, t_2, \ldots, t_k$ are terms and R is a k-ary relation symbol, then $R(t_1, t_2, \ldots, t_k)$ is a formula.
- If t_1 and t_2 are terms, then $t_1 = t_2$ is a formula.
- If α is a formula, then so is $\neg\alpha$.
- If α and β are formulas, then so is $\alpha \vee \beta$.
- If α is a formula, then so is $\exists x.\alpha$.

Finally, a *sentence* is a formula containing no free variables.

12.2.2. Semantics

A *language* L is a set of function and relation symbols. A *structure* (or *model*, or *interpretation*) for a language L is a pair $M = (U^M, I^M)$, where $U^M \neq \phi$. U^M defines the domain over which variables range, and I^M gives semantics for each of the function and relation symbols in L.

For example, consider the language $L = \{+, <, 0\}$. We can define a model $\mathbb{Z} = (U^{\mathbb{Z}}, I^{\mathbb{Z}})$, where $U^{\mathbb{Z}} = \{\ldots, -1, 0, 1, \ldots\}$ and $I^{\mathbb{Z}}$ defines $+^{\mathbb{Z}}$ to be integer addition, $<^{\mathbb{Z}}$ to be integer less-than, and $0^{\mathbb{Z}}$ to be the integer constant 0. Clearly $\mathbb{Z}$ is a model for the integers.

Because formulas can contain free variables, we need an associated *assignment* σ in a model M, which maps each variable to an element of U^M. The assignment σ can then be inductively extended in the obvious way to interpret all terms:

$$\sigma(f(t_1, t_2, \ldots, t_k)) = f^M(\sigma(t_1), \sigma(t_2), \ldots, \sigma(t_k))$$

Now we can inductively define the semantics of a formula α under a given model M and assignment σ:

- $M, \sigma \models R(t_1, t_2, \ldots, t_k)$ if $R^M(\sigma(t_1), \sigma(t_2), \ldots, \sigma(t_k))$
- $M, \sigma \models t_1 = t_2$ if $\sigma(t_1) = \sigma(t_2)$
- $M, \sigma \models \neg\beta$ if $M, \sigma \not\models \beta$
- $M, \sigma \models \beta \vee \gamma$ if $M, \sigma \models \beta$ or $M, \sigma \models \gamma$
- $M, \sigma \models \exists x.\beta$ if there is an assignment τ such that $\tau(y) = \sigma(y)$ for all $y \neq x$, and $M, \tau \models \beta$

If α is a sentence, then σ is irrelevant since α has no free variables, so we simply write $M \models \alpha$. A sentence α is *valid* if for all models M, $M \models \alpha$. A sentence α is *satisfiable* if for some model M, $M \models \alpha$.

Let Γ be a set of sentences. We write $M \models \Gamma$ if for all $\alpha \in \Gamma, M \models \alpha$. We write $\Gamma \models \alpha$ if for all models M such that $M \models \Gamma$, $M \models \alpha$. Define the *consequences* $CN(\Gamma)$ of a set of sentences Γ as $\{\alpha : \Gamma \models \alpha\}$. If $\Gamma = CN(\Gamma)$, then Γ is called a *theory.* A theory Γ is *consistent* if it has a model. A theory Γ is *complete* if for all α, either $\alpha \in \Gamma$ or $\neg\alpha \in \Gamma$.

Define the theory $Th(M)$ of a model M as $\{\alpha : M \models \alpha\}$.

Proposition 12.1: Γ is consistent and complete if and only if $\Gamma = Th(M)$ for some model M.

Proof: $\Rightarrow$: Suppose Γ is consistent and complete. Since it's consistent, it has some model M. Therefore, $\Gamma \subseteq Th(M)$. Suppose $\Gamma \neq Th(M)$, so there is some sentence α such that $\alpha \in Th(M)$ but $\alpha \notin \Gamma$. Since Γ is complete, this means that $\neg\alpha \in \Gamma$, but then $\neg\alpha \in Th(M)$. So both α and $\neg\alpha$ are in $Th(M)$, which cannot occur according to the semantics presented above. Therefore, we have a contradiction, so it must be the case that $\Gamma = Th(M)$.

$\Leftarrow$: Suppose $\Gamma = Th(M)$. Then by the definition of $Th(M)$, $M \models \Gamma$, so Γ is consistent. Further, by the semantics presented above, we know that for all sentences α, either $M \models \alpha$ or $M \models \neg\alpha$. Therefore $Th(M)$ is complete, so Γ is complete as well. $\square$

In the next lecture, we will discuss the following three theorems:

Theorem 12.2 (Completeness): If Γ is recursively enumerable, then so is $CN(\Gamma)$.

Theorem 12.3 (Compactness): If Γ is inconsistent, then some finite subset of Γ is inconsistent.

Theorem 12.4 (Lowenheim-Skolem): If Γ is countable and consistent, then Γ has a countable model.

These theorems will be proven by the *Herbrand universe* method.