Introduction to Logic in Computer Science: Autumn 2006

Ulle Endriss
Institute for Logic, Language and Computation
University of Amsterdam

Plan for Today

One of the grand success stories of logic in mainstream computer science has been the application of various logics to the specification and verification of both hardware and software systems.

This lecture will introduce several *temporal logics* that are being used for this purpose, and it will outline a couple of *model checking* algorithms that can be used to check whether a given model (representing a system) satisfies certain properties (expressed as temporal logic formulas).

- LTL (linear-time); CTL (branching-time); CTL$*$ (both, sort of)
- Model checking in CTL and LTL

Remark: Temporal logics are generally considered useful for specifying and verifying programs that run continuously (such as operating systems), as opposed to input-output programs (which are better tackled using Hoare logic or PDL).

Linear-time Temporal Logic

LTL is the original temporal logic (also known as "tense logic"), going back to work in philosophy by Arthur Prior (mid 1950s). Early pioneers: Hans Kamp, Dov Gabbay, Johan van Benthem.

Amir Pnueli's seminal paper (1977) started the field of program specification and verification using temporal logic.

Syntax: $\varphi \ ::= \ p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \mathrm{X}\varphi \mid \mathrm{F}\varphi \mid \mathrm{G}\varphi \mid \varphi \,\mathrm{U}\, \varphi$

Intuitive reading of the temporal operators:

- $\mathrm{X}\varphi$: φ is true at the next state (alternative syntax: $\bigcirc\varphi$)
- $\mathrm{F}\varphi$: φ is true at *some* future state ($\Diamond\varphi$)
- $\mathrm{G}\varphi$: φ is true at *all* future states, globally ($\Box\varphi$)
- $\varphi \,\mathrm{U}\, \psi$: φ is true until ψ is true

A. Pnueli. *The Temporal Logic of Programs.* Proc. 18th Annual Symposium on Foundations of Computer Science, 1977.

Semantics of LTL

A *transition system* $\mathcal{M} = (S, R, V)$ consists of a set of states S, a binary relation $R \subseteq S \times S$, and a valuation function V mapping atomic propositions to sets of states ($\leadsto$ Kripke model).

For ease of presentation, we assume that R is *serial* (not a serious restriction: we could always add an explicit "deadlock state").

A *path* $\pi = s_0, s_1, s_2, \ldots$ is an infinite sequence of R-successors. Write π^i for the subpath starting at s_i. Define *truth on a path* $\pi = s_0, \ldots$ like this:

- $\pi \models p$ iff $s_0 \in V(p)$ for atomic propositions p;
- $\pi \models \neg\varphi$ iff not $\pi \models \varphi$, and similarly for $\wedge$, $\vee$ and $\rightarrow$;
- $\pi \models \mathrm{X}\varphi$ iff $\pi^1 \models \varphi$;
- $\pi \models \mathrm{F}\varphi$ iff there exists an $i \geq 0$ such that $\pi^i \models \varphi$;
- $\pi \models \mathrm{G}\varphi$ iff $\pi^i \models \varphi$ for all $i \geq 0$;
- $\pi \models \varphi \,\mathrm{U}\, \psi$ iff there is an $i \geq 0$ s.t. $\pi^i \models \psi$ and $\pi^j \models \varphi$ for all $j < i$.

Write $\mathcal{M}, s \models \varphi$ iff $\pi \models \varphi$ for *every* path π in $\mathcal{M}$ starting at state s.

Alternative Interpretations

Our definitions follow the standard way of defining LTL in the context of program verification. Classically, models of LTL are based on just a single linear time line $(\mathbb{N}, <)$: just one "path" in our terminology here. Hence, the name *linear*-time temporal logic.

People have also considered LTL with respect to other *flows of time*, such as the rational numbers ...

Remark: Hans Kamp (1968) showed that our set of operators is *expressively complete:* anything expressible in the first-order theory of $(\mathbb{N}, <)$ can also be said in the temporal language ...

J.A.W. Kamp. *Tense Logic and the Theory of Linear Order.* PhD thesis, Department of Philosophy, UCLA, 1968.

Examples

Some properties that we can express in LTL:

- If a `request` occurs then it will eventually be `acknowledged`: $\mathrm{G}(\texttt{request} \rightarrow \mathrm{F}\,\texttt{acknowledge})$
- Some process is `enabled` infinitely often: $\mathrm{GF}\,\texttt{enabled}$
- $\mathrm{G}(\texttt{floor2} \wedge \texttt{going-up} \wedge \texttt{pressed5} \rightarrow (\texttt{going-up}\,\mathrm{U}\,\texttt{floor5}))$

Some properties that we cannot express in LTL:

- From any state it is possible to get to a `restart` state.
- The lift can remain idle on the third floor with its doors closed (from that state, there is a path where it stays on the third floor ...).

LTL cannot express these properties, because we cannot freely quantify over paths (universal quantification is implicit).

The examples on this slide are taken from Huth and Ryan (2004).

M. Huth and M. Ryan. *Logic in Computer Science.* Cambridge University Press, 2nd edition, 2004.

Computation Tree Logic

CTL is a branching-time temporal logic allowing for quantification over paths. However, as formulas will be evaluated over states, all quantifiers now have to come *in pairs:* first we quantify over paths and then over states on the chosen path.

Syntax: $\varphi \ ::= \ p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \mathrm{AX}\varphi \mid \mathrm{EX}\varphi \mid \mathrm{AF}\varphi \mid \mathrm{EF}\varphi \mid \mathrm{AG}\varphi \mid \mathrm{EG}\varphi \mid \mathrm{A}(\varphi\,\mathrm{U}\,\varphi) \mid \mathrm{E}(\varphi\,\mathrm{U}\,\varphi)$

Intuitive reading of some of the temporal operators:

- $\mathrm{AX}\varphi$: on *all* paths starting here, φ will be true next.
- $\mathrm{EF}\varphi$: there *exists* a path emanating from the current state on which φ will be true at *some* future state.

CTL has been introduced by Clarke and Emerson (1981), together with a suitable model checking algorithm.

E.M. Clarke and E.A. Emerson. *Design and Synthesis of Synchronization Skeletons Using Branching Time Temporal Logic.* Logics of Programs, 1981.

Semantics of CTL

Truth at a state s_0 in a transition system $\mathcal{M} = (S, R, V)$:

- $\mathcal{M}, s_0 \models p$ iff $s_0 \in V(p)$ for atomic propositions p;
- $\mathcal{M}, s_0 \models \neg\varphi$ iff not $\mathcal{M}, s_0 \models \varphi$, and similarly for $\wedge$, $\vee$ and $\rightarrow$;
- $\mathcal{M}, s_0 \models \mathrm{AX}\varphi$ iff $\mathcal{M}, s_1 \models \varphi$ for all $s_1 \in S$ with $s_0 R s_1$;
- $\mathcal{M}, s_0 \models \mathrm{EX}\varphi$ iff $\mathcal{M}, s_1 \models \varphi$ for some $s_1 \in S$ with $s_0 R s_1$;
- $\mathcal{M}, s_0 \models \mathrm{AF}\varphi$ iff for all paths $s_0, s_1, \ldots \ \exists\, s_i$ s.t. $\mathcal{M}, s_i \models \varphi$;
- $\mathcal{M}, s_0 \models \mathrm{EF}\varphi$ iff for some path $s_0, s_1, \ldots \ \exists\, s_i$ s.t. $\mathcal{M}, s_i \models \varphi$;
- $\mathcal{M}, s_0 \models \mathrm{AG}\varphi$ iff for all paths $s_0, s_1, \ldots$, $\mathcal{M}, s_i \models \varphi$ for all s_i;
- $\mathcal{M}, s_0 \models \mathrm{EG}\varphi$ iff for some path $s_0, s_1, \ldots$, $\mathcal{M}, s_i \models \varphi$ for all s_i;
- $\mathcal{M}, s_0 \models \mathrm{A}(\varphi\,\mathrm{U}\,\psi)$ iff for all paths $s_0, s_1, \ldots$ there exists an s_i s.t. $\mathcal{M}, s_i \models \psi$ and $\mathcal{M}, s_j \models \varphi$ for all $j < i$;
- $\mathcal{M}, s_0 \models \mathrm{E}(\varphi\,\mathrm{U}\,\psi)$ iff for some path $s_0, s_1, \ldots$ there exists an s_i s.t. $\mathcal{M}, s_i \models \psi$ and $\mathcal{M}, s_j \models \varphi$ for all $j < i$.

Some Equivalences

The following equivalences show that we can eliminate most of the temporal operators:

$$\mathrm{AX}\varphi \equiv \neg\mathrm{EX}\neg\varphi \tag{1}$$

$$\mathrm{EG}\varphi \equiv \neg\mathrm{AF}\neg\varphi \tag{2}$$

$$\mathrm{AG}\varphi \equiv \neg\mathrm{EF}\neg\varphi \tag{3}$$

$$\mathrm{EF}\varphi \equiv \mathrm{E}(\top\,\mathrm{U}\,\varphi) \tag{4}$$

$$\mathrm{A}(\varphi\,\mathrm{U}\,\psi) \equiv \mathrm{AF}\psi \wedge \neg\mathrm{E}[\neg\psi\,\mathrm{U}\,(\neg\varphi\wedge\neg\psi)] \tag{5}$$

Examples

Again, these are taken from Huth and Ryan (2004).

- We are now able to express that the lift can remain idle on the third floor with the doors closed:
  $\mathrm{AG}(\texttt{floor3} \wedge \texttt{idle} \wedge \texttt{closed} \rightarrow \mathrm{EG}(\texttt{floor3} \wedge \texttt{idle} \wedge \texttt{closed}))$
- In CTL, we cannot express "all paths which make p true at one of their states also make q true at one of their states".
  But in LTL this is easy: $\mathrm{F}\,p \rightarrow \mathrm{F}\,q$

CTL∗: The Best of Both Worlds

CTL∗ has been introduced by Emerson and Halpern (1983). It allows you to freely mix all operators and thereby generalises both LTL and CTL (think of LTL formulas as being preceded by an A).

State formulas φ are evaluated over states:

$$\varphi ::= p \mid \neg\varphi \mid \varphi\wedge\varphi \mid \varphi\vee\varphi \mid \varphi\rightarrow\varphi \mid \mathrm{A}\alpha \mid \mathrm{E}\alpha$$

Path formulas α are evaluated over paths:

$$\alpha ::= \varphi \mid \neg\alpha \mid \alpha\wedge\alpha \mid \alpha\vee\alpha \mid \alpha\rightarrow\alpha \mid \mathrm{X}\alpha \mid \mathrm{F}\alpha \mid \mathrm{G}\alpha \mid \alpha\,\mathrm{U}\,\alpha$$

The semantics of CTL∗ is defined in the obvious manner . . . (note that if a path formula is also a state formula, then it is evaluated at the first state of the path in question).

E.A. Emerson and J.Y. Halpern. *"Sometimes" and "not never" Revisited: On Branching versus Linear Time.* Proc. POPL-1983.

CTL Model Checking

Given a model $\mathcal{M}$, a state s, and a formula φ, the *model checking* problem is the problem of determining whether $\mathcal{M}, s \models \varphi$ holds.

Alternative formulation: Given $\mathcal{M}$ and φ, find all s s.t. $\mathcal{M}, s \models \varphi$.

Recall that we can restrict ourselves to the temporal operators EX, AF and EU (and $\neg$ and $\wedge$) . . .

Labelling Algorithm

Recursively go through all subformulas ψ of φ (starting with the shortest formulas) and decide which states to label with ψ. If ψ is of the form ...

- p (atomic proposition), then label s with p if $s \in V(p)$.
- $\neg\psi_1$, then label s with $\neg\psi_1$ if s is not labelled with ψ.
- $\psi_1 \wedge \psi_2$, then label s with $\psi_1 \wedge \psi_2$ if s is labelled with ψ_1 and ψ_2.
- $\mathrm{EX}\psi_1$: label any state with $\mathrm{EX}\psi_1$ if one of its immediate successor states is labelled with ψ_1.
- $\mathrm{AF}\psi_1$: (1) label any state labelled with ψ_1 also with $\mathrm{AF}\psi_1$;
  (2) repeat: label any state with $\mathrm{AF}\psi_1$ if all its immediate successor states are labelled with $\mathrm{AF}\psi_1$.
- $\mathrm{E}(\psi_1 \,\mathrm{U}\, \psi_2)$: (1) label any state labelled with ψ_2 with $\mathrm{E}(\psi_1 \,\mathrm{U}\, \psi_2)$;
  (2) repeat: label any state with $\mathrm{E}(\psi_1 \,\mathrm{U}\, \psi_2)$ if it is labelled with ψ_1 and one of its immediate successors is labelled with $\mathrm{E}(\psi_1 \,\mathrm{U}\, \psi_2)$.

Finally, return the set of states s labelled with the full formula φ.

LTL Model Checking

First of all, observe that the simple approach taken for CTL won't work for LTL: now subformulas are evaluated over *paths*.

The general idea for checking whether $\mathcal{M}, s \models \varphi$ is the following:

(1) Construct an *automaton* $A_{\neg\varphi}$ for the formula $\neg\varphi$. States are maximally consistent sets of subformulas of φ (or their complements) and the transition relation is arranged "appropriately" (see next slide).

(2) Combine $A_{\neg\varphi}$ and $\mathcal{M}$ to get another transition system. The paths in that system are both paths in $A_{\neg\varphi}$ and $\mathcal{M}$. Model checking succeeds iff there is no path in this transition system.

We assume that φ only uses negation, conjunction, X and U (F and G are definable in terms of U).

Constructing the Automaton

We construct $A_{\neg\varphi}$ as follows. Introduce a state q for each maximally consistent subset of the set of subformulas of φ and their complements:

- For all non-negated ψ, either $\psi \in q$ or $\neg\psi \in q$ (but not both).
- $(\psi_1 \wedge \psi_2) \in q$ iff both $\psi_1 \in q$ and $\psi_2 \in q$.
- If $(\psi_1 \,\mathrm{U}\, \psi_2) \in q$ then $\psi_1 \in q$ or $\psi_2 \in q$.
- If $\neg(\psi_1 \,\mathrm{U}\, \psi_2) \in q$ then $\neg\psi_2 \in q$.

The initial states are those including $\neg\varphi$. For the transition relation δ, we define $(q, q') \in \delta$ iff all of the following conditions hold:

- $\psi \in q'$ whenever $(\mathrm{X}\psi) \in q$; and $\neg\psi \in q'$ whenever $(\neg\mathrm{X}\psi) \in q$.
- $(\psi_1 \,\mathrm{U}\, \psi_2) \in q'$ whenever $(\psi_1 \,\mathrm{U}\, \psi_2) \in q$ and $\psi_2 \not\in q$.
- $\neg(\psi_1 \,\mathrm{U}\, \psi_2) \in q'$ whenever $\neg(\psi_1 \,\mathrm{U}\, \psi_2) \in q$ and $\psi_1 \in q$.

Acceptance condition: the run has infinitely many states satisfying $\neg(\psi_1 \,\mathrm{U}\, \psi_2) \vee \psi_2$ for each subformula of the form $\psi_1 \,\mathrm{U}\, \psi_2$.

Complexity Results

Our intuitions about the simplicity of various model checking algorithms are confirmed by formal complexity results:

Theorem 1 (Clarke and Emerson, 1981) *CTL model checking is in* **P**.

Theorem 2 (Sistla and Clarke, 1982) *LTL (and CTL∗) model checking are both* **PSPACE***-complete.*

But note that the main parameter determining the problem size is the number of states—and this number may itself be exponential in some other, more interesting parameter.

E.M. Clarke and E.A. Emerson. *Design and Synthesis of Synchronization Skeletons Using Branching Time Temporal Logic.* Logics of Programs, 1981.

A.P. Sistla and E.M. Clarke. *The Complexity of Propositional Linear Temporal Logics.* Proc. STOC-1982.

References

The main reference in the area of model checking:

• E.M. Clarke, O. Grumberg, D.A. Peled. *Model Checking*. MIT Press, 1999.

Chapter 3 of the following undergraduate textbook is about model checking and has been my main reference for this lecture:

• M. Huth and M. Ryan. *Logic in Computer Science*. Cambridge University Press, 2nd edition, 2004.

The following are excellent books on temporal logic *per se*, without paying special attention to applications in computer science:

• D.M. Gabbay, I. Hodkinson, and M. Reynolds. *Temporal Logic*, Vol. 1. Oxford University Press, 1994.

• R. Goldblatt. *Logics of Time and Computation*. CSLI, 2nd edition, 1992.

Summary

- We have introduced three temporal logics:
  - LTL for linear time
  - CTL for branching time
  - CTL∗: a generalisation of CTL that allows for arbitrary combinations of path and state quantification
- The general idea of using temporal logic and model checking for program verification is that a transition system can serve as a model of the system to be checked and formulas of the temporal logic of choice can express desirable properties.
- We have seen algorithms for CTL and LTL model checking.
- Much work has gone into optimising model checking algorithms by finding suitable forms of representation for very large models: *symbolic model checking*