Overview of the Temporal Logic CTL*: Part 2

by
Edmund M. Clarke, Orna Grumberg, and
Doron A. Peled

February 7, 2003

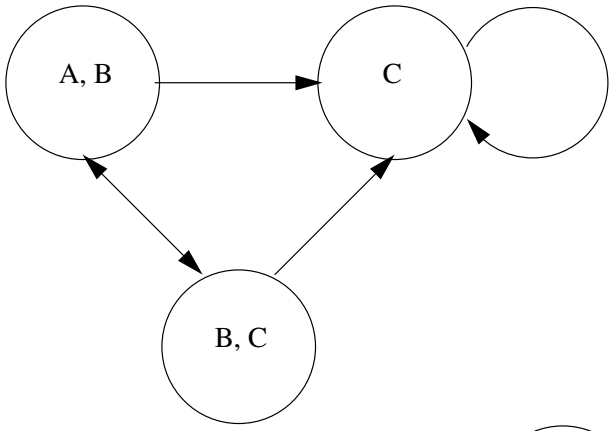
Okay, so what do we do with a Kripke structure?

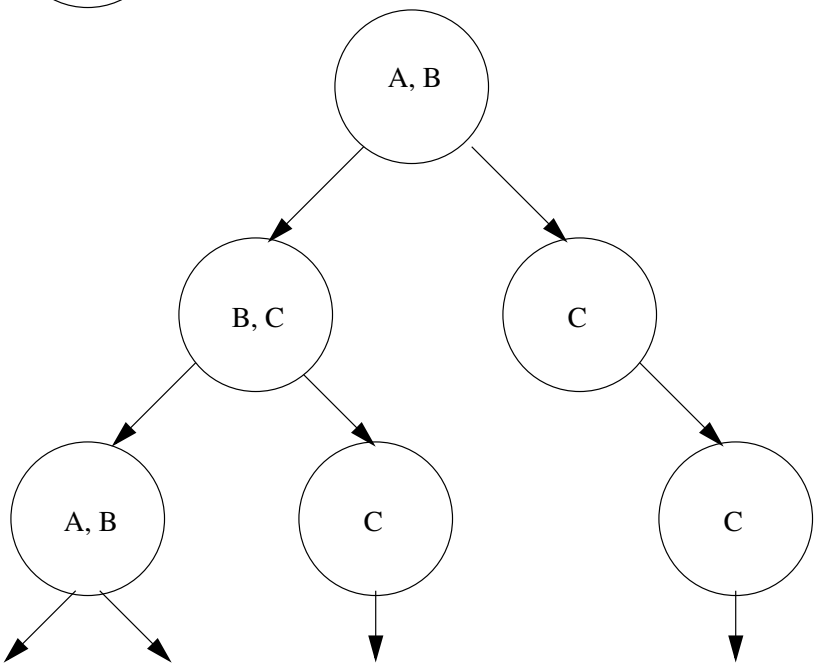
Model checking is used to verify finite state concurrent systems. Once we have constructed a Kripke structure for a system, we represent properties of this system in a temporal logic. While several temporal logics may be used, the one we will discuss is the temporal logic **CTL***.

Computation Trees

Recall that a Kripke structure can be viewed as a transition diagram with a finite set of initial states. A Kripke structure is often represented as a forest of computation trees. For each initial state, such a tree can be constructed by unravelling the graph into an infinite tree.

Example:





Note: In this example, our Kripke structure had only one initial state, and consequently the forest consists of only one tree. If we would have had multiple initial states, we would add the corresponding trees to the forest.

Syntax of CTL*

There are two types of formulas in **CTL***: *state formulas* and *path formulas*. State formulas represent the properties of a specific state, whereas path formulas specify the properties of a specific path.

CTL* formulas are composed of *path quantifiers*, *temporal operators*, and *atomic propositions*.

Path Quantifiers

CTL* introduces the following two path quantifiers:

- **A** - the universal path quantifier
- **E** - the existential path quantifier

Examples: Let p_1 and p_2 denote various properties of paths. The path quantifiers are used as follows:

- **A** p_1 - all paths beginning at some state s satisfy the property p_1.

- **E** p_2 - property p_2 is satisfied for some paths beginning at some state s.

Temporal Operators

CTL* introduces the following five temporal operators:

- **X** - the next time operator
- **F** - the eventually/future time operator
- **G** - the global operator
- **U** - the until operator
- **R** - the release operator

The next time operator (**X**) is used to specify that some property holds in the second state of a path.

Example:

$powerOn \wedge \mathbf{X}\ lightOn$ - the power is on, and in the next state of our path, the light is on.

The future time operator (**F**) is used to specify that some property eventually holds at some state in our path.

Example:

F $lightOn$ - eventually the light is on in some state of our path.

The global operator (**G**) is used to specify that some property holds for all states of a particular path.

Example:

G $possibleToChangeBulb$ - in all states of our path, it is always possible to change a bulb.

The until operator (**U**) is a binary operator, and is used to specify that the first property holds in all states preceding the one where the second property is satisfied.

Example:

$lightOff$ **U** $lightOn$ - the light is off in all states of our path preceding the state where the light is on.

The release operator (**R**) is also a binary operator, and is used to specify that the second property holds in all states along a path up to and including the first state that satisfies the first property. The first property is not required to be eventually satisfied.

Example:

$changeBulb$ **R** $brokenBulb$ - the bulb is broken until we change it.

We are now ready to formally define the syntax of **CTL***.

Let AP be the set of atomic proposition names. We define the syntax of **CTL*** as follows:

- If $p \in AP$ then p is a state formula.
- If f is a state formula, then f is also path formula.
- If f and g are state formulas, then $\neg f$, $f \wedge g$, and $f \vee g$ are state formulas.

- If f is a path formula, then **E** f, and **A** f are state formulas.

- If f and g are path formulas, then $\neg f$, $f \vee g$, $f \wedge g$, **X** f, **F** f, **G** f, f **U** g, and f **R** g are path formulas.

Notation

Before defining the semantics of **CTL***, we introduce the following notation:

- π is used to denote a path in a Kripke structure.
- π^i is used to dentoe the *suffix* of π beginning with state s_i.
- $M, s \models f_s$ means that the state formula f_s holds at state s in the Kripke structure M.
- $M, \pi \models f_p$ means that the path formula f_p holds along path π in the Kripke structure M.

Semantics

The entailment relation ($\models$), is defined recursively as follows:

Let f_1 and f_2 be state formulas, and g_1 and g_2 be path formulas.

- $M, s \models p \Leftrightarrow p \in s$
- $M, s \models \neg f_1 \Leftrightarrow M, s \not\models f_1$
- $M, s \models f_1 \vee f_2 \Leftrightarrow M, s \models f_1$ or $M, s \models f_2$
- $M, s \models f_1 \wedge f_2 \Leftrightarrow M, s \models f_1$ and $M, s \models f_2$
- $M, s \models \mathbf{E}\, g_1 \Leftrightarrow$ there is a path π from s such that $M, \pi \models g_1$

- $M, s \models \mathbf{A}\ g_1 \Leftrightarrow$ for every path π from s such that $M, \pi \models g_1$

- $M, \pi \models f_1 \Leftrightarrow s$ is the first state in π and $M, s \models f_1$

- $M, \pi \models \neg g_1 \Leftrightarrow M, \pi \not\models g_1$

- $M, \pi \models g_1 \vee g_2 \Leftrightarrow M, \pi \models g_1$ or $M, \pi \models g_2$

- $M, \pi \models g_1 \wedge g_2 \Leftrightarrow M, \pi \models g_1$ and $M, \pi \models g_2$

- $M, \pi \models \mathbf{X}\ g_1 \Leftrightarrow M, \pi^1 \models g_1$

- $M, \pi \models \mathbf{F}\ g_1 \Leftrightarrow$ there exists a $k \geq 0$ such that $M, \pi^k \models g_1$

- $M, \pi \models \mathbf{G}\ g_1 \Leftrightarrow$ for every $i \geq 0$, $M, \pi^i \models g_1$

- $M, \pi \models g_1\ \mathbf{U}\ g_2 \Leftrightarrow$ there exists a $k \geq 0$ such that $M, \pi^k \models g_1$, and for all $0 \leq j < k$, $M, \pi^j \models g_2$

- $M, \pi \models g_1\ \mathbf{R}\ g_2 \Leftrightarrow$ for all $j \geq 0$, if for every $i < j$ $M, \pi^i \not\models g_1$ then $M, \pi^j \models g_2$

Equivalences

The following **CTL*** formulas are equivalent:

- $f \wedge g \equiv \neg(\neg f \vee \neg g)$
- $f \ \mathbf{R} \ g \equiv \neg(\neg f \ \mathbf{U} \ \neg g)$
- $\mathbf{F} g \equiv True \ \mathbf{U} \ g$
- $\mathbf{G} f \equiv \neg \mathbf{F} \ \neg f$
- $\mathbf{A} \ (f) \equiv \neg \mathbf{E} \ (\neg f)$

Final Remarks

Our temporal logic formulas specify the properties of computation trees. Once we have our specification, we can use the various model checking algorithms to verify whether or not our Kripke structure satisfies the specification or not.

The various model checking algorithms will be discussed in a future talk.