CTL - syntax and semantics

Franco Raimondi

Department of Computer Science
School of Science and Technology
Middlesex University
http://www.rmnd.net

CTL syntax and semantics

Model checking: definition

(this material is taken from Chapter 5 and from Huth and Ryan)

- Given a model M and a formula ϕ, model checking is the problem of verifying whether or not ϕ is true in M (written $M \models \phi$).
- Mainly for temporal logics.
- We have seen how to perform LTL model checking.
- Another approach: CTL model checking.
- CTL example: *There exists an execution of the system such that, if the proposition p is true, then in the next computation step q is true*

CTL Syntax

We start from a set of *atomic propositions* $AP = \{p, q, \dots\}$.
Atomic propositions stand for atomic facts which may hold in a system, e.g. *"Printer ps706 is busy"*, *"Process 1486 is idle"*, *"The value of x is 5"*, etc.
The Backus-Naur form form CTL formulae is the following:

$$\phi \quad ::= \quad \top \mid \bot \mid p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \mid AX\phi \mid EX\phi \mid AF\phi \mid EF\phi \mid AG\phi \mid EG\phi \mid A[\phi U \phi] \mid E[\phi U \phi]$$

CTL Syntax

Each CTL operator is a pair of symbols. The first one is either A ("for All paths"), or E ("there Exists a path"). The second one is one of X ("neXt state"), F ("in a Future state"), G ("Globally in the future") or U ("Until").

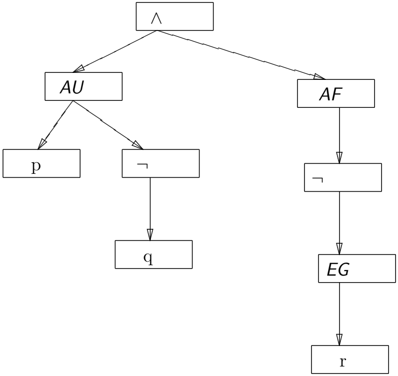
NOTICE: U is a *binary* operator, it could be written $EU(\phi, \psi)$ or $AU(\phi, \psi)$. Notice that the quantifier is graphically separated (e.g., $E[pUq]$), but it is in fact a single operator EU, which could be written $EU(p, q)$.

Example: $AG(p \rightarrow (EFq))$ is read as "It is Globally the case that, if p is true, then there Exists a path such that at some point in the Future q is true".

CTL Syntax: parse trees

Parse trees are very useful to understand CTL formulas. For instance:
Build the parse tree of the following formula:

$$A[pU\neg q] \wedge (AF(\neg EGr))$$



CTL Syntax: EXERCISE

Is it a wff? Why?

1. $EFGr$
2. $A\neg G\neg p$
3. $A[pU(EFr)]$
4. $F[rUq]$
5. $EF(rUq)$
6. $AEFr$
7. $A[rUA[pUq]]$
8. $A[(rUq) \wedge (pUr)]$

CTL Syntax: EXERCISE

Answers

1. $EFGr$ NO
2. $A\neg G\neg p$ NO
3. $A[pU(EFr)]$ YES
4. $F[rUq]$ NO
5. $EF(rUq)$ NO
6. $AEFr$ NO
7. $A[rUA[pUq]]$ YES
8. $A[(rUq) \wedge (pUr)]$ NO

CTL Semantics

You should be able to identify well-formed CTL formulae. Now: how to evaluate formulae, i.e., how to decide whether or not a formula is true.

You should know the meaning of *tautology* and *unsatisfiable formulae*:

- $AG(p \vee \neg p)$: tautology
- $AG(p \wedge \neg p)$: unsatisfiable

But what about EFp? it may be true or not, depending on how we evaluate formulae.

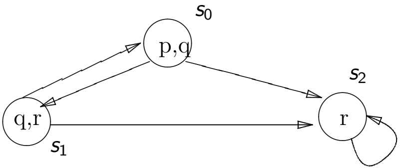
CTL Semantics: transition systems

We evaluate formulae in *transition systems*. A transition system model a system by means of *states* and *transitions* between states. Formally:

A transition system $M = (S, R_t, I, L)$ is a set of states S with a binary relation $R_t \subseteq S \times S$, a set of initial states and a labelling function $L : S \to 2^{AP}$ (AP is a set of atomic propositions, see above). The relation R_t is *serial*, i.e., for every state $s \in S$, there exists a state s' s.t. sR_ts'.

CTL Semantics: transition systems

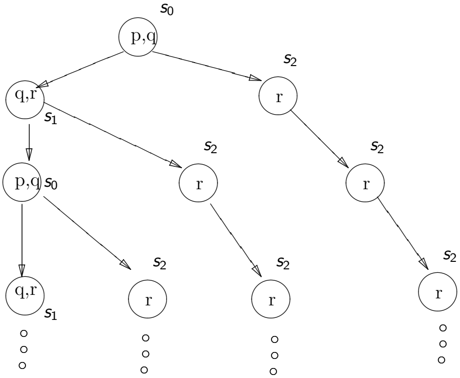
An example $M = (S, R_t, L)$



Here $S = \{s_0, s_1, s_2\}$,
$R_t = \{(s_0, s_1), (s_0, s_2), (s_1, s_0), (s_1, s_2), (s_2, s_2)\}$, and
$L(s_0) = \{p, q\}$, $L(s_1) = \{q, r\}$, $L(s_2) = \{r\}$.

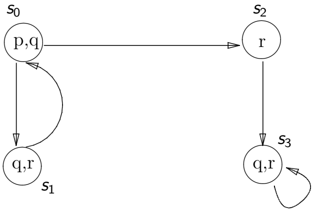
CTL semantics: from transition systems to computation paths

It is useful to visualise all possible computation paths by *unwinding* the transition system (given an initial state):

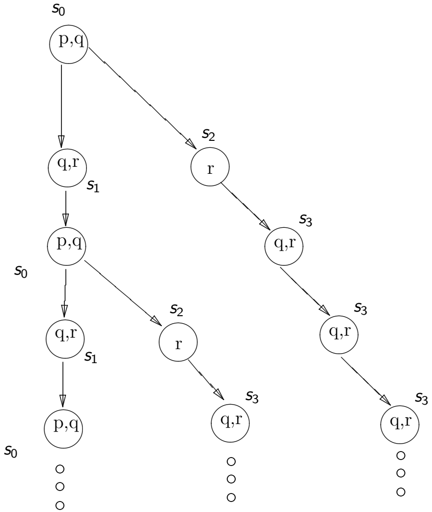


CTL semantics: computation paths EXERCISE

Unwind the following transition systems from s_0:



CTL semantics: computation paths EXERCISE SOLUTION



Short summary

- You should be able to recognise well-formed CTL formulas.
- You know what a transition system is ($M = (S, R_t, L)$).
- You know how to unwind a transition system and obtain computation paths.

Next: Given a CTL formula ϕ and a transition system M, establish whether or not ϕ is true at a given state s in M, written as:

$$M, s \models \phi$$

CTL semantics (finally!)

Let $M = (S, R_t, I, L)$ be a transition system (also called a *model* for CTL). Let ϕ be a CTL formula and $s \in S$. $M, s \models \phi$ is defined inductively on the structure of ϕ, as follows (I'm using the first transition system of today as an example on the board):

$M, s \models \top$		
$M, s \not\models \bot$		
$M, s \models p$	iff	$p \in L(s)$
$M, s \models \neg\phi$	iff	$M, s \not\models \phi$
$M, s \models \phi \wedge \psi$	iff	$M, s \models \phi$ and $M, s \models \phi$
$M, s \models \phi \vee \psi$	iff	$M, s \models \phi$ or $M, s \models \phi$

CTL Semantics (temporal operators)

$M, s \models AX\phi$	iff	$\forall s'$ s.t. sR_ts', $M, s' \models \phi$
$M, s \models EX\phi$	iff	$\exists s'$ s.t. sR_ts' and $M, s' \models \phi$
$M, s \models AG\phi$	iff	for all paths $(s, s_2, s_3, s_4, \dots)$ s.t. $s_i R_t s_{i+1}$ and for all i, it is the case that $M, s_i \models \phi$
$M, s \models EG\phi$	iff	*there is a path* $(s, s_2, s_3, s_4, \dots)$ s.t. $s_i R_t s_{i+1}$ and for all i it is the case that $M, s_i \models \phi$
$M, s \models AF\phi$	iff	for all paths $(s, s_2, s_3, s_4, \dots)$ s.t. $s_i R_t s_{i+1}$, there is a state s_i s.t. $M, s_i \models \phi$
$M, s \models EF\phi$	iff	*there is a path* $(s, s_2, s_3, s_4, \dots)$ s.t. $s_i R_t s_{i+1}$, and there is a state s_i s.t. $M, s_i \models \phi$

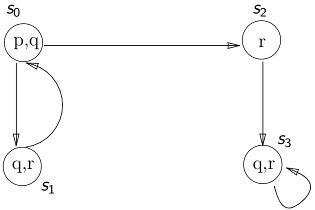
CTL Semantics (temporal operators)

$M, s \models A[\phi U \psi]$	iff	for all paths $(s, s_2, s_3, s_4, \dots)$ s.t. $s_i R_t s_{i+1}$ there is a state s_j s.t. $M, s_j \models \psi$ and $M, s_i \models \psi$ for all $i < j$.
$M, s \models E[\phi U \psi]$	iff	there exists a path $(s, s_2, s_3, s_4, \dots)$ s.t. $s_i R_t s_{i+1}$ and th a state s_j s.t. $M, s_j \models \psi$ and $M, s_i \models \psi$ for all $i < j$.

We write $M \models \phi$ if a formula is true in all the initial states of a model.

CTL semantics: EXERCISE

Consider the following transition system:



Verify whether or not: (1) $M, s_0 \models EX(\neg p)$; (2) $M, s_0 \models EXEG(r)$; (3) $M, s_1 \models AG(q \vee r)$; (4) $M, s_2 \models A[rUq]$; (5) $M, s_1 \models A[qUAG(r)]$; (6) $M, s_1 \models E[qUEG(r)]$; (7) $M, s_0 \models \neg EG(q)$; (8) $M, s_1 \models EFAG(q)$.

CTL semantics: EXERCISE SOLUTIONS

(1) YES; (2) YES; (3) YES ; (4) YES; (5) NO (because $AG(r)$ is never true if you keep looping between s_0 and s_1); (6) YES; (7) NO; (8) YES.

Equivalences between CTL formulae

In the syntax of CTL we introduced all the operators AX, EX, AF, EF, AG, EG, AU, and EU. However, some formulas are equivalent:

$$
\begin{aligned}
AX\phi &\equiv \neg EX\neg\phi \\
AG\phi &\equiv \neg EF\neg\phi \\
AF\phi &\equiv \neg EG\neg\phi
\end{aligned}
$$

Moreover, $EF\phi \equiv E[\top U\phi]$. Therefore, only three operators are required to express all the remaining: EX, EG, EU (this is called an *adequate set of operators*. This is useful when developing algorithms for model checking.

Specification patterns

Temporal logics are useful to express requirements of systems. Typically, requirements have *common and recurring patterns*. For instance, two example of patterns:

- **Liveness**: "Something good will eventually happen". For instance: "Whenever any process requests to enter its critical section, it will eventually be permitted to do so". In CTL:

$$AG(request \rightarrow AF(critical))$$

- **Safety**: "Nothing bad will happen". For instance, "Only one process is in its critical section at any time". In CTL (with 2 processes only):

$$AG(\neg(critical_1 \wedge critical_2))$$

Specification patterns: EXERCISE

Write in CTL the following requirements:

1. "From any state it is possible to get a `reset` state"
2. "Event p precedes s and t on all computation paths".
3. "On all computation paths, after p, q is never true".