Problem:
Given a finite state environment $E$, and a formula $\phi$, determine if $E,(r,0) \models \phi$ for all runs $r$ of $E$.

CTL - a restricted fragment of branching time temporal logic
We defined branching time temporal logic so that if $\phi$ is a formula then $A\phi$ and $E\phi$ are formulas
So, e.g., $A((\bigcirc p) \lor p \bigvee q)$ is a formula
CTL is the logic in which the branching operators $A, E$ apply only to formulas in which the outermost operator is a temporal (not boolean) operator:
E.g. $A((\bigcirc p) \lor p \bigvee q)$ is not a CTL formula
But $A(p \bigvee q), E \bigcirc p A\Box E p$ are CTL formulas

Model Checking CTL + Knowledge, Observational View

Theorem: Model Checking of $\phi \in \mathcal{L}(K_1,\ldots,K_n,C,\forall \bigcirc, \forall \bigvee, \exists \bigvee)$ in $E$ with respect to $\mathrm{obs}$ is in PTIME.
Let $S'$ be the set of reachable states of $E$

label states of $M = \langle S', K_1, \ldots, K_n, \pi_e \rangle$ by subformulas of $\phi$

1. label $s$ by $K_i \psi$ ($C\psi$) if $sK_it$ ($sK_tC t$) implies $t$ labelled $\psi$

2. label $s$ by $\forall \psi$ if $sTt$ implies $t$ labelled by $\psi$

3. label $s$ by $\exists (\psi_1 U \psi_2)$ if there exists a sequence $s = s_0, s_1, \ldots, s_k$
   such that $s_k$ labelled $\psi_2$ and for $l < k$ $s_lTs_{l+1}$ and $s_l$ is labelled $\psi_1$

4. label $s$ by $\neg \forall (\psi_1 U \psi_2)$ if there exists a sequence $s = s_0, s_1, \ldots, s_k, \ldots s_m$
   of states such that
   (a) $s_lTs_{l+1}$ for all $l < m$
   (b) $s_k = s_m$
   (c) $\{s_k, \ldots, s_m\} \cap \alpha \neq \emptyset$
   (d) either $s_l$ is not labelled $\psi_2$ for all $l \leq m$, or, for the least $l$
      such that $s_l$ is labelled $\psi_2$ there exists $l' < l$ such that $s_{l'}$ is not labelled $\psi_1$

It’s not necessary to construct every state to run this algorithm, it can be done symbolically.

Step 1: represent each state $s$ as a Boolean assignment to a set of state variables $V = \{v_1, \ldots, v_n\}$: $s: V \rightarrow \{0, 1\}$

Step 2: represent a set $X$ of states as a Boolean function $f_X : \{0, 1\} \times \ldots \times \{0, 1\} \rightarrow \{0, 1\}$ with $n$ arguments $x_1, \ldots, x_n$, so that $f_X(x_1, \ldots, x_n) = 1$ iff $s \in X$ where $s$ is the state with $s(v_1) = x_1, \ldots s(v_n) = x_n$.

Step 3: Compute the set $[\phi] = \{s \in S | E, s \models \phi\}$ using the above rule setwise, using this representation: e.g., $f_{[\phi_1 \land \phi_2]} = f_{[\phi_1]} \land f_{[\phi_2]}$.

Step 4: represent these functions as binary decision diagrams....
Operations on BDD’s
Given BDD’s representing boolean functions \( f, g \), we can compute BDD’s representing

1. the functions \( \neg f, (f \land g), (f \lor g) \), defined pointwise, e.g.
   \( (f \land g)(v) = f(v) \land g(v) \).

2. the function \( \exists v(f) \). If the arguments of \( f \) are \( u, v, w \), this is defined by
   \( (\exists v(f))(u, w) = f(u, 0, w) \lor f(u, 1, w) \).

Model Checking with respect to Perfect Recall
Theorem [van der Meyden & Shilov 99]: Model Checking \( \mathcal{L}_{\{\bigcirc, u, K_1, \ldots, K_n\}} \) with respect to perfect recall is decidable, but non-elementary in complexity.

This lecture: Focus on implementation for formulas of the form \( \bigcirc^k \phi \) where \( \phi \in \mathcal{L}_{\{K_1\}} \).
Let $o_j$ be a sequence of boolean variables representing an observation of agent 1.

Let $s$ be a sequence of variables representing a state of the environment.

Define

$$f_k(o_0, \ldots, o_k, s) = 1 \text{ iff there exists a run } r \text{ such that}$$

1. agent 1’s observations during $[0, \ldots, k]$ in $r$ are $(o_0, \ldots, o_k)$
2. $r(k) = s$
Computing $f_k$ recursively

$$f_{k+1}(o_0, \ldots, o_{k+1}, s) = \exists t(f_k(o_0, \ldots, o_k, t) \land T(t, s) \land O_i(s) = o_{k+1})$$

\[ Sat_p(o_0, \ldots, o_k, s) = \pi_e(p, s) \]
\[ Sat_{\neg \phi}(o_0, \ldots, o_k, s) = \neg Sat_{\phi}(o_0, \ldots, o_k, s) \]
\[ Sat_{\alpha \land \beta}(o_0, \ldots, o_k, s) = Sat_{\alpha}(o_0, \ldots, o_k, s) \land Sat_{\beta}(o_0, \ldots, o_k, s) \]
\[ Sat_{K_1\alpha}(o_0, \ldots, o_k, s) = \forall s'(f_k(o_0, \ldots, o_k, s) \Rightarrow Sat_{\alpha}(o_0, \ldots, o_k, s')) \]

Model checking $\bigcirc^k \phi$:

$$\forall o_0 \ldots o_k s(f_k(o_0, \ldots, o_k, s) \Rightarrow Sat_{\phi}(o_0, \ldots, o_k, s))$$

MCK: a model checker for the logic of knowledge and time

A system developed at UNSW. It can be downloaded from

http://www.cse.unsw.edu.au/~mck

You can also run it (preferably on williams) from

/import/kamen/1/peteg/bin/mck-cudd

Clock View

If $\rho = s_0, s_1, \ldots$ is a run of an environment, the clock view is the (synchronous) local state assignment defined by

$$\rho^\text{clock}_i(m) = (m, O_i(\rho(m)))$$
### MCK: Current Capability

<table>
<thead>
<tr>
<th></th>
<th>Observational</th>
<th>Clock</th>
<th>Perfect Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>(X^\phi, \phi \in L{K_i})</td>
<td>(\text{spec}_{\text{ltl}})</td>
<td>(\text{spec}_{\text{ltl}})</td>
<td>(\text{spec}_{\text{ltl}})</td>
</tr>
<tr>
<td>(L(X, K_1, \ldots, K_n))</td>
<td>(\text{spec}_{\text{ltl}})</td>
<td>(\text{spec}_{\text{ltl}})</td>
<td>(\text{spec}_{\text{ltl}})</td>
</tr>
<tr>
<td>(L(\text{CTL}, K_1, \ldots, K_n, C))</td>
<td>(\text{spec}_{\text{ltl}})</td>
<td>(\text{spec}_{\text{ltl}})</td>
<td>(\text{spec}_{\text{ltl}})</td>
</tr>
<tr>
<td>(L(\text{LTL}, K_1, \ldots, K_n, C))</td>
<td>(\text{spec}_{\text{ltl}})</td>
<td>(\text{spec}_{\text{ltl}})</td>
<td>(\text{spec}_{\text{ltl}})</td>
</tr>
</tbody>
</table>

(sp = synchronous perfect recall)