Slide 1

COMP3152/9152
Lecture 6
Model Checking Knowledge and Time
Ron van der Meyden

Slide 2

Problem:

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

Slide 3

CTL - a restricted fragment of branching time temporal logic

We defined branching time temporal logic so that if ϕ is a formula then $A\phi$ and $E\phi$ are formulas

So, e.g., $A((\bigcirc p) \vee p\,\mathcal{U}\,q)$ 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. $A((\bigcirc p) \vee p\,\mathcal{U}\,q)$ is not a CTL formula

But $A(p\,\mathcal{U}\,q)$, $E \bigcirc p$ $A\Box E\Diamond p$ are CTL formulas

Slide 4

Model Checking CTL + Knowledge, Observational View

Theorem: Model Checking of $\phi \in \mathcal{L}_{\{K_1,\ldots,K_n,C,\forall\bigcirc,\forall\,\mathcal{U},\exists\,\mathcal{U}\}}$ in E with respect to `obs` is in PTIME.

Slide 5

Let S' be the set of reachable states of E

label states of $M = \langle S', \mathcal{K}_1, \ldots, \mathcal{K}_n, \pi_e \rangle$ by subformulas of ϕ

1. label s by $K_i\psi$ ($C\psi$) if $s\mathcal{K}_i t$ ($s\mathcal{K}_C t$) implies t labelled ψ
2. label s by $\forall \bigcirc \psi$ if sTt implies t labelled by ψ
3. label s by $\exists(\psi_1 \; \mathcal{U} \; \psi_2)$ if there exists a sequence $s = s_0, s_1, \ldots, s_k$ such that s_k labelled ψ_2 and for $l < k$ $s_l T s_{l+1}$ and s_l is labelled ψ_1

Slide 6

4. label s by $\neg\forall(\psi_1 \; \mathcal{U} \; \psi_2)$ if there exists a sequence $s = s_0, s_1, \ldots, s_k, \ldots s_m$ of states such that
   (a) $s_l T s_{l+1}$ 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 ψ_2 for all $l \leq m$, or, for the least l such that s_l is labelled ψ_2 there exists $l' < l$ such that $s_{l'}$ is not labelled ψ_1

Slide 7

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 $V = \{v_1, \ldots, v_n\}$: $s : V \rightarrow \{0, 1\}$

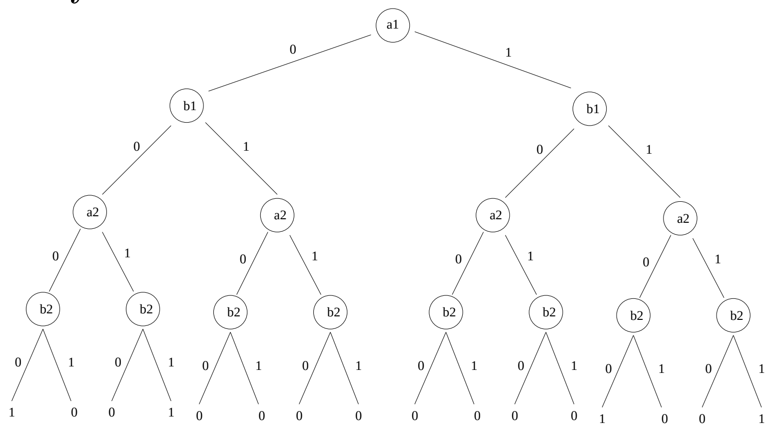
Step 2: represent a set X of states as a Boolean function $f_X : \{0,1\} \times \ldots \times \{0,1\} \rightarrow \{0,1\}$ with n arguments $x_1, \ldots, x_n$, so that $f_X(x_1, \ldots, x_n) = 1$ iff $s \in X$ where s is the state with $s(v_1) = x_1, \ldots s(v_n) = x_n$.

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

Step 4: represent these functions as binary decision diagrams....

Slide 8

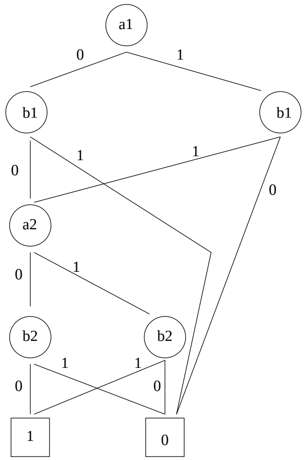
Binary Decision Trees



$f(a1, a2, b1, b2) = (a1 = b1) \wedge (a2 = b2)$

Slide 9

Binary Decision Diagrams



lide 10

Operations on BDD's

Given BDD's representing boolean functions f, g, we can compute BDD's representing

1. the functions $(\neg f)$, $(f \wedge g)$, $(f \vee g)$, defined pointwise, e.g. $(f \wedge g)(\mathbf{v}) = f(\mathbf{v}) \wedge g(\mathbf{v})$.
2. the function $\exists v(f)$. If the arguments of f are $\mathbf{u}, v, \mathbf{w}$, this is defined by

$$(\exists v(f))(\mathbf{u}, \mathbf{w}) = f(\mathbf{u}, 0, \mathbf{w}) \vee f(\mathbf{u}, 1, \mathbf{w})$$

Slide 11

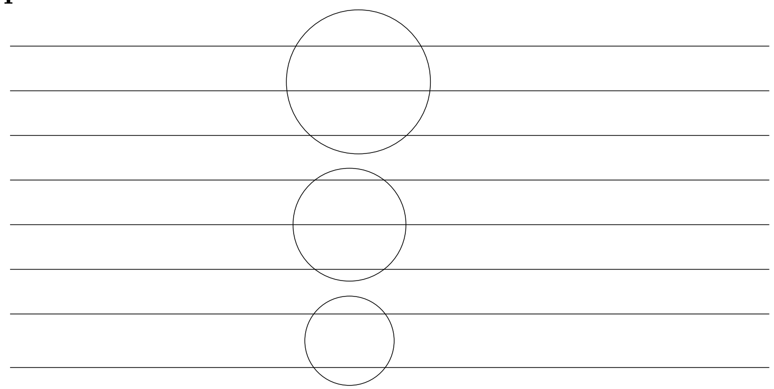
Model Checking with respect to Perfect Recall

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

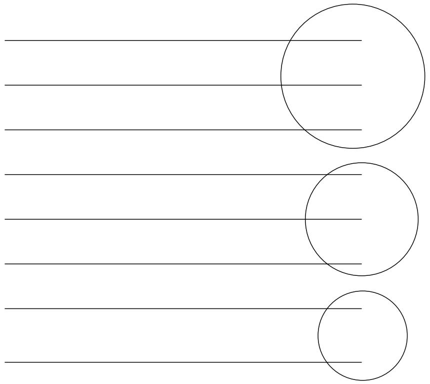
This lecture: Focus on implementation for formulas of the form $\bigcirc^k \phi$ where $\phi \in \mathcal{L}_{\{K_1\}}$

Slide 12

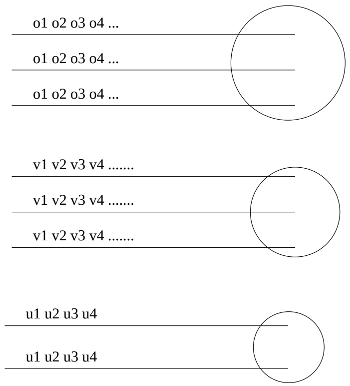
Reducing Model Checking Knowledge to BDD operations



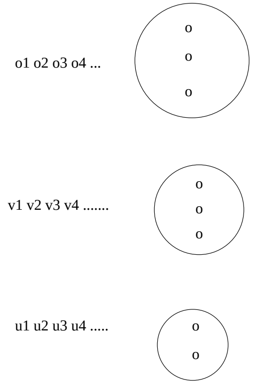
lide 13



lide 14



Slide 15



Slide 16

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

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

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

lide 17

Computing f_k recursively

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

lide 18

$Sat_p(o_0, \ldots, o_k, s) = \pi_e(p, s)$

$Sat_{\neg\phi}(o_0, \ldots, o_k, s) = \neg Sat_\phi(o_0, \ldots, o_k, s)$

$Sat_{\alpha\wedge\beta}(o_0, \ldots, o_k, s) = Sat_\alpha(o_0, \ldots, o_k, s) \wedge Sat_\beta(o_0, \ldots, o_k, s)$

$Sat_{K_1\alpha}(o_0, \ldots, o_k, s) = \forall s'(f_k(o_0, \ldots, o_k, s) \Rightarrow Sat_\alpha(o_0, \ldots, o_k, s'))$

Model checking $\bigcirc^k \phi$:

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

Slide 19

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

Slide 20

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

$\rho_i^{\texttt{clock}}(m) = (m, O_i(\rho(m)))$

lide 21

MCK: Current Capability

	Observational	Clock	Perfect Recall
$\mathbf{X}^n\phi$, $\phi \in \mathcal{L}_{\{K_i\}}$	**spec_obs_ltl**	**spec_clk_xn**	**spec_spr_xn**
$\mathcal{L}_{\{\mathbf{X},K_1,\ldots,K_n\}}$	**spec_obs_ltl**		**spec_spr_ltl_nested**
$\mathcal{L}_{\{\mathbf{X},K_1,\ldots,K_n\}}$	**spec_obs_ctl**	**spec_clk_ctl_nested**	
$\mathcal{L}_{\{CTL,K_1,\ldots,K_n,C\}}$	**spec_obs_ctl**		
$\mathcal{L}_{\{LTL,K_n,\ldots,K_n,C\}}$	**spec_obs_ltl**		

(spr = synchronous perfect recall)