CTL Model Checking

Lecture #19 of Model Checking

Joost-Pieter Katoen

Lehrstuhl 2: Software Modeling & Verification

E-mail: `katoen@cs.rwth-aachen.de`

June 13, 2007

Overview Lecture #19

$\Rightarrow$ Existential normal form

- Basic CTL model-checking algorithm
- Algorithms for $\exists(\Phi \, \mathsf{U} \, \Psi)$ and $\exists\Box\,\Phi$
- Time complexity

Existential normal form (ENF)

The set of CTL formulas in *existential normal form* (ENF) is given by:

$$\Phi \quad ::= \quad \text{true} \;\Big|\; a \;\Big|\; \Phi_1 \;\wedge\; \Phi_2 \;\Big|\; \neg\Phi \;\Big|\; \exists\bigcirc\,\Phi \;\Big|\; \exists(\Phi_1 \,\mathsf{U}\, \Phi_2) \;\Big|\; \exists\Box\,\Phi$$

For each CTL formula, there exists an equivalent CTL formula in ENF

$$\forall\bigcirc\,\Phi \quad \equiv \quad \neg\exists\bigcirc\,\neg\Phi$$

$$\forall(\Phi \,\mathsf{U}\, \Psi) \quad \equiv \quad \neg\exists(\neg\Psi \,\mathsf{U}\, (\neg\Phi \wedge \neg\Psi)) \;\wedge\; \neg\exists\Box\,\neg\Psi$$

Overview Lecture #19

- Existential normal form

⇒ Basic CTL model-checking algorithm

- Algorithms for $\exists(\Phi \,\mathsf{U}\, \Psi)$ and $\exists\Box\,\Phi$

- Time complexity

Model checking CTL

- How to check whether state *TS* satisfies CTL formula $\widehat{\Phi}$?
  - convert the formula $\widehat{\Phi}$ into the equivalent Φ in ENF
  - compute *recursively* the set $Sat(\Phi) = \{ s \in S \mid s \models \Phi \}$
  - $TS \models \Phi$ if and only if each initial state of *TS* belongs to $Sat(\Phi)$

- Recursive bottom-up computation of $Sat(\Phi)$:
  - consider the parse-tree of Φ
  - start to compute $Sat(a_i)$, for all leafs in the tree
  - then go one level up in the tree and determine $Sat(\cdot)$ for these nodes

$$\text{e.g.,:} \quad Sat(\underbrace{\Psi_1 \wedge \Psi_2}_{\text{node at level } i}) = Sat(\underbrace{\Psi_1}_{\substack{\text{node at} \\ \text{level } i+1}}) \cap Sat(\underbrace{\Psi_2}_{\substack{\text{node at} \\ \text{level } i+1}})$$

  - then go one level up and determine $Sat(\cdot)$ of these nodes
  - and so on....... until the root is treated, i.e., $Sat(\Phi)$ is computed

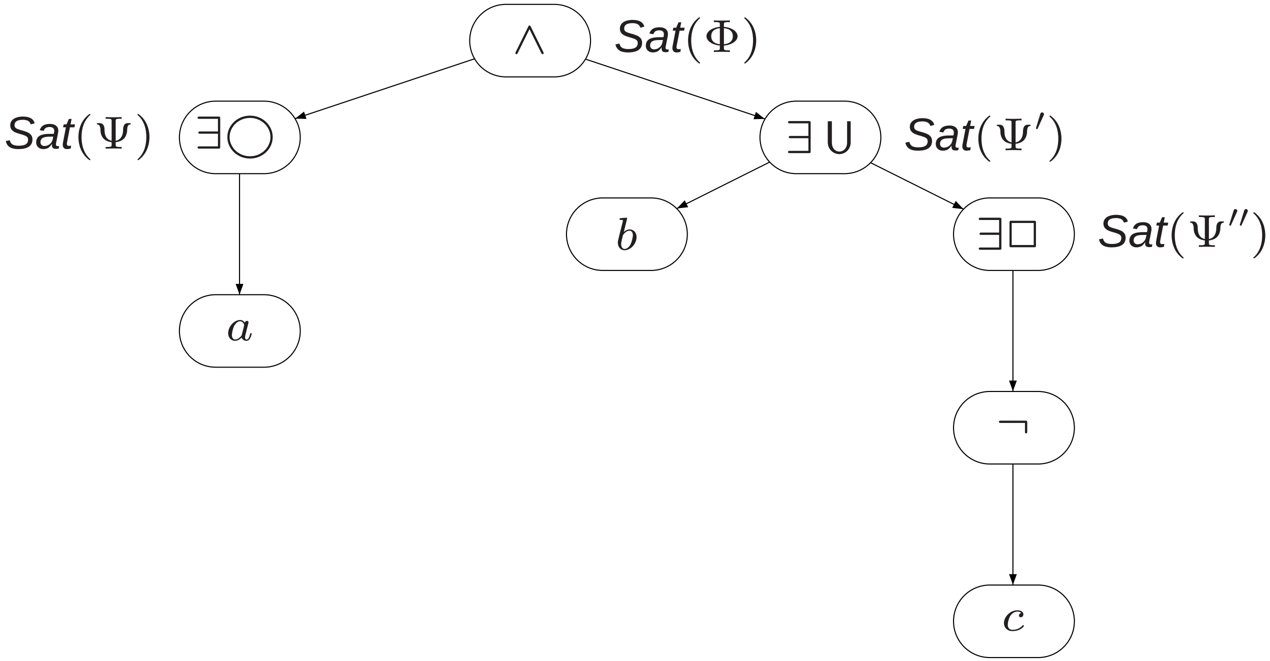
Basic algorithm

Input: finite transition system *TS* and CTL formula Φ (both over *AP*)
Output: *TS* $\models \Phi$

(* compute the sets *Sat*$(\Phi) = \{ s \in S \mid s \models \Phi \}$ *)

for all $i \leqslant |\Phi|$ **do**
  for all $\Psi \in$ *Sub*(Φ) with $|\Psi| = i$ **do**
    compute *Sat*(Ψ) from *Sat*(Ψ') (* for maximal proper $\Psi' \in$ *Sub*(Ψ) *)
  od
od
return $I \subseteq$ *Sat*(Φ)

Example



$$\Phi = \underbrace{\exists\bigcirc a}_{\Psi} \wedge \underbrace{\exists(b\,\mathsf{U}\,\underbrace{\exists\Box\neg c}_{\Psi''})}_{\Psi'} .$$

Characterization of *Sat* (1)

For all CTL formulas Φ, Ψ over *AP* it holds:

$$
\begin{aligned}
\mathit{Sat}(\text{true}) &= S \\
\mathit{Sat}(a) &= \{\, s \in S \mid a \in L(s) \,\}, \text{ for any } a \in \mathit{AP} \\
\mathit{Sat}(\Phi \wedge \Psi) &= \mathit{Sat}(\Phi) \cap \mathit{Sat}(\Psi) \\
\mathit{Sat}(\neg \Phi) &= S \setminus \mathit{Sat}(\Phi) \\
\mathit{Sat}(\exists \bigcirc \Phi) &= \{\, s \in S \mid \mathit{Post}(s) \cap \mathit{Sat}(\Phi) \neq \varnothing \,\}
\end{aligned}
$$

where $\mathit{TS} = (S, \mathit{Act}, \rightarrow, I, \mathit{AP}, L)$ is a transition system without terminal states

Characterization of *Sat* (2)

- $Sat(\exists(\Phi \, \mathsf{U} \, \Psi))$ is the <u>smallest</u> subset T of S, such that:

$$(1)\ Sat(\Psi) \subseteq T \quad \text{and} \quad (2)\ (s \in Sat(\Phi) \text{ and } Post(s) \cap T \neq \varnothing) \quad \Rightarrow \quad s \in T$$

- $Sat(\exists \Box \, \Phi)$ is the <u>largest</u> subset T of S, such that:

$$(3)\ T \subseteq Sat(\Phi) \quad \text{and} \quad (4)\ s \in T \text{ implies } Post(s) \cap T \neq \varnothing$$

where $TS = (S, Act, \rightarrow, I, AP, L)$ is a transition system without terminal states

Proof

Computation of *Sat*

switch(Φ):

a	:	**return** $\{ s \in S \mid a \in L(s) \}$;
...	:	....
$\exists \bigcirc \Psi$	:	**return** $\{ s \in S \mid Post(s) \cap Sat(\Psi) \neq \varnothing \}$;

$\exists(\Phi_1 \mathsf{U} \Phi_2)$	:	$T := Sat(\Phi_2)$; (* compute the smallest fixed point *)
		while $\{ s \in Sat(\Phi_1) \setminus T \mid Post(s) \cap T \neq \varnothing \} \neq \varnothing$ **do**
		  let $s \in \{ s \in Sat(\Phi_1) \setminus T \mid Post(s) \cap T \neq \varnothing \}$;
		  $T := T \cup \{ s \}$;
		od;
		return T;

$\exists \Box \Phi$	:	$T := Sat(\Phi)$; (* compute the greatest fixed point *)
		while $\{ s \in T \mid Post(s) \cap T = \varnothing \} \neq \varnothing$ **do**
		  let $s \in \{ s \in T \mid Post(s) \cap T = \varnothing \}$;
		  $T := T \setminus \{ s \}$;
		od;
		return T;

end switch

Overview Lecture #19

- Existential normal form
- Basic CTL model-checking algorithm

$\Rightarrow$ Algorithms for $\exists(\Phi \, \mathsf{U} \, \Psi)$ and $\exists\Box\,\Phi$

- Time complexity

Computing $\mathit{Sat}(\exists(\Phi \,\mathsf{U}\, \Psi))$ (1)

- $\mathit{Sat}(\exists(\Phi \,\mathsf{U}\, \Psi))$ is the smallest set $T \subseteq S$ such that:

$$(1)\ \mathit{Sat}(\Psi) \subseteq T \quad \text{and} \quad (2)\ (s \in \mathit{Sat}(\Phi) \text{ and } \mathit{Post}(s) \cap T \neq \varnothing) \;\Rightarrow\; s \in T$$

- This suggests to compute $\mathit{Sat}(\exists(\Phi \,\mathsf{U}\, \Psi))$ iteratively:

$$T_0 = \mathit{Sat}(\Psi) \quad \text{and} \quad T_{i+1} = T_i \cup \{ s \in \mathit{Sat}(\Phi) \mid \mathit{Post}(s) \cap T_i \neq \varnothing \}$$

- T_i = states that can reach a Ψ-state in at most i steps via a Φ-path

- By induction on j it follows:

$$T_0 \subseteq T_1 \subseteq \ldots \subseteq T_j \subseteq T_{j+1} \subseteq \ \ldots \ \subseteq \mathit{Sat}(\exists(\Phi \,\mathsf{U}\, \Psi))$$

Computing *Sat*$(\exists(\Phi \, \mathsf{U} \, \Psi))$ (2)

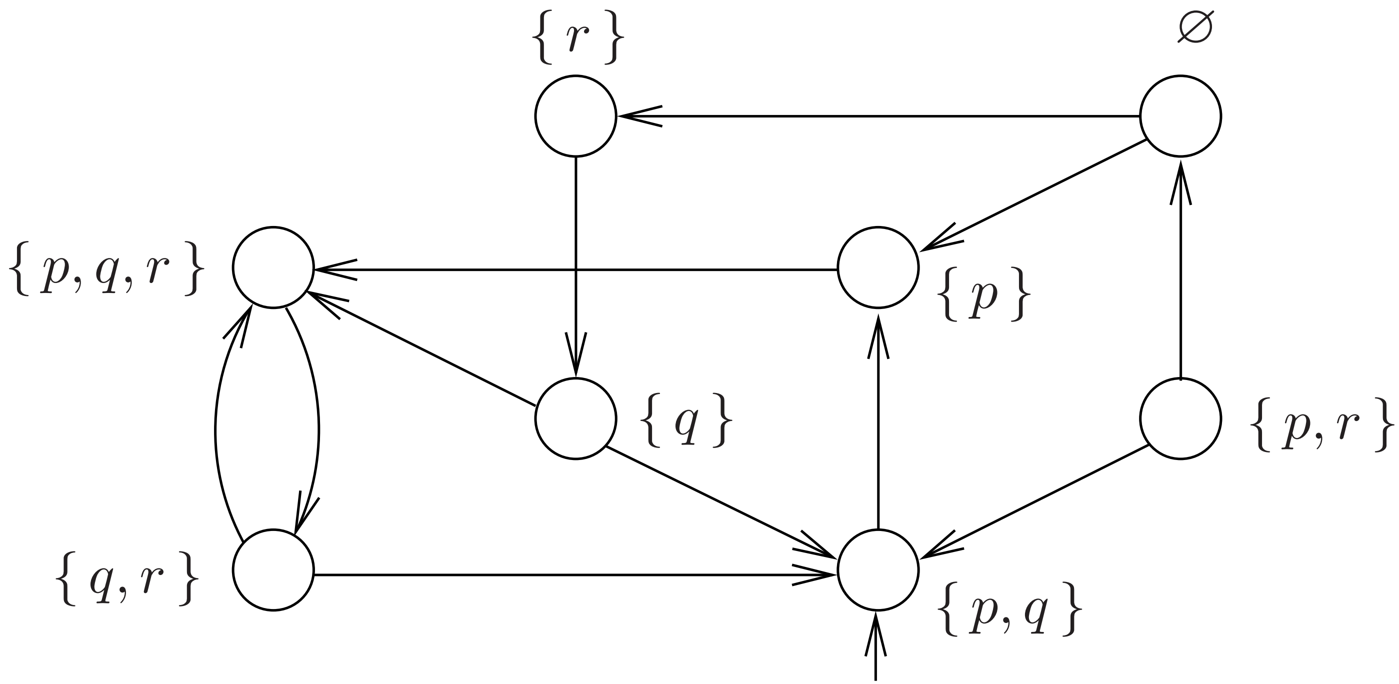
- *TS* is finite, so for some $j \geqslant 0$ we have: $T_j = T_{j+1} = T_{j+2} = \ldots$
- Therefore: $T_j = T_j \cup \{ s \in \mathit{Sat}(\Phi) \mid \mathit{Post}(s) \cap T_j \neq \varnothing \}$
- Hence: $\{ s \in \mathit{Sat}(\Phi) \mid \mathit{Post}(s) \cap T_j \neq \varnothing \} \subseteq T_j$
  - hence, T_j satisfies (2), i.e., $(s \in \mathit{Sat}(\Phi)$ and $\mathit{Post}(s) \cap T_j \neq \varnothing) \Rightarrow s \in T_j$
  - further, $\mathit{Sat}(\Psi) = T_0 \subseteq T_j$ so, T_j satisfies (1), i.e. $\mathit{Sat}(\Psi) \subseteq T_j$
- As *Sat*$(\exists(\Phi \, \mathsf{U} \, \Psi))$ is the *smallest* set satisfying (1) and (2):
  - $\mathit{Sat}(\exists(\Phi \, \mathsf{U} \, \Psi)) \subseteq T_j$ and thus $\mathit{Sat}(\exists(\Phi \, \mathsf{U} \, \Psi)) = T_j$
- Hence: $T_0 \subsetneqq T_1 \subsetneqq T_2 \subsetneqq \ldots \subsetneqq T_j = T_{j+1} = \ldots = \mathit{Sat}(\exists(\Phi \, \mathsf{U} \, \Psi))$

Computing *Sat*$(\exists(\Phi \,\mathsf{U}\, \Psi))$ (3)

Input: finite transition system *TS* with state-set S and CTL-formula $\exists(\Phi \,\mathsf{U}\, \Psi)$
Output: *Sat*$(\exists(\Phi \,\mathsf{U}\, \Psi)) = \{ s \in S \mid s \models \exists(\Phi \,\mathsf{U}\, \Psi) \}$

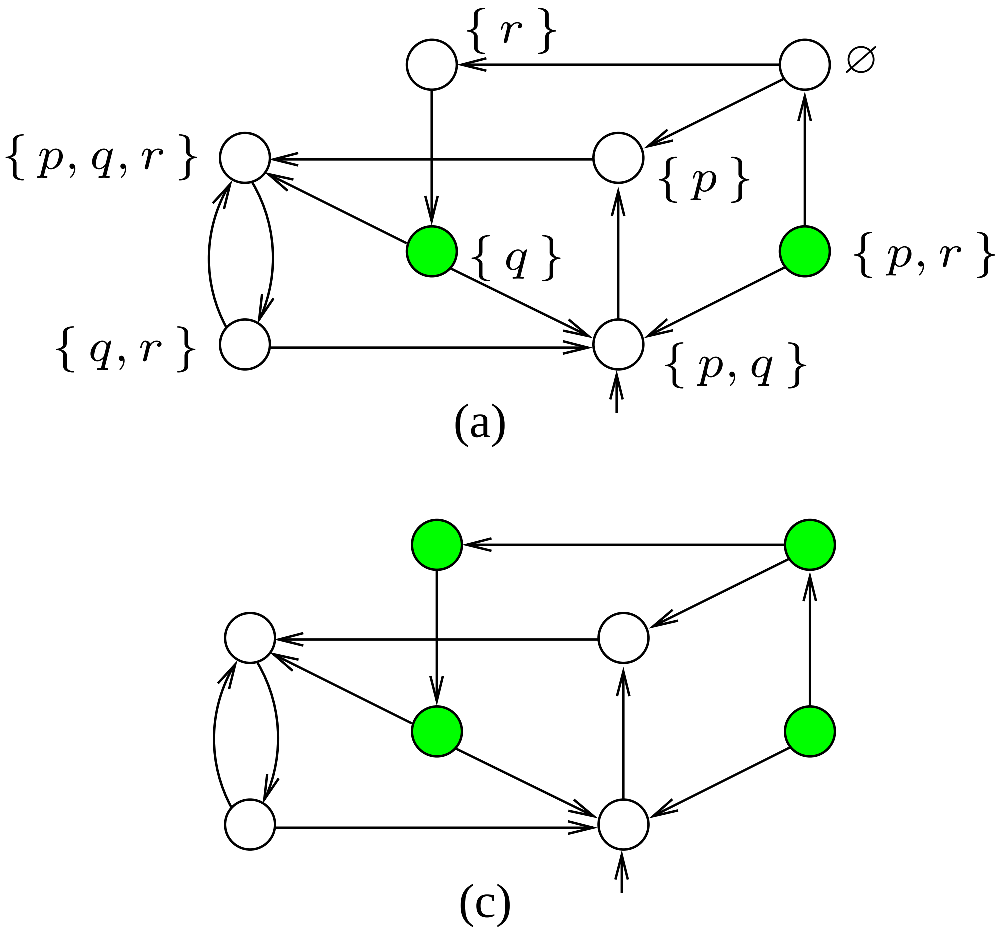
$E :=$ *Sat*(Ψ); (* E administers the states s with $s \models \exists(\Phi \,\mathsf{U}\, \Psi)$ *)
$T := E$; (* T contains the already visited states s with $s \models \exists(\Phi \,\mathsf{U}\, \Psi)$ *)
while $E \neq \varnothing$ **do**
  let $s' \in E$;
  $E := E \setminus \{ s' \}$;
  for all $s \in$ *Pre*(s') **do**
    if $s \in$ *Sat*$(\Phi) \setminus T$ **then** $E := E \cup \{ s \}; T := T \cup \{ s \}$; **endif**
  od
od
return T

Example



let's check the CTL-formula $\exists\Diamond\,((p = r) \wedge (p \neq q))$

The computation in snapshots



Computing $Sat(\exists\Box\,\Phi)$

Computing $\mathit{Sat}(\exists\Box\,\Phi)$

```
E := S \ Sat(Φ);                    (* E contains any not visited s' with s' ⊭ ∃□Φ *)

T := Sat(Φ);                        (* T contains any s for which s ⊨ ∃□Φ has not yet been disproven *)

for all s ∈ Sat(Φ) do c[s] := | Post(s) |; od                    (* initialize array c *)

while E ≠ ∅ do
                                    (* loop invariant: c[s] = | Post(s) ∩ (T ∪ E) | *)
  let s' ∈ E;                                                    (* s' ⊭ Φ *)
  E := E \ { s' };                                               (* s' has been considered *)
  for all s ∈ Pre(s') do
    if s ∈ T then
      c[s] := c[s] − 1;             (* update counter c[s] for predecessor s of s' *)
      if c[s] = 0 then
        T := T \ { s }; E := E ∪ { s };       (* s does not have any successor in T *)
      fi
    fi
  od
od
return T
```

Example

Alternative algorithm for *Sat*$(\exists\Box\,\Phi)$

1. Consider only state s if $s \models \Phi$, otherwise *eliminate* s

- change *TS* into $TS[\Phi] = (S', Act, \rightarrow', I', AP, L')$ with $S' = Sat(\Phi)$,
- $\rightarrow' = \rightarrow \cap (S' \times Act \times S')$, $I' = I \cap S'$, and $L'(s) = L(s)$ for $s \in S'$

$\Rightarrow$ all removed states will not satisfy $\exists\Box\,\Phi$, and thus can be safely removed

2. Determine all *non-trivial strongly connected components* in $TS[\Phi]$

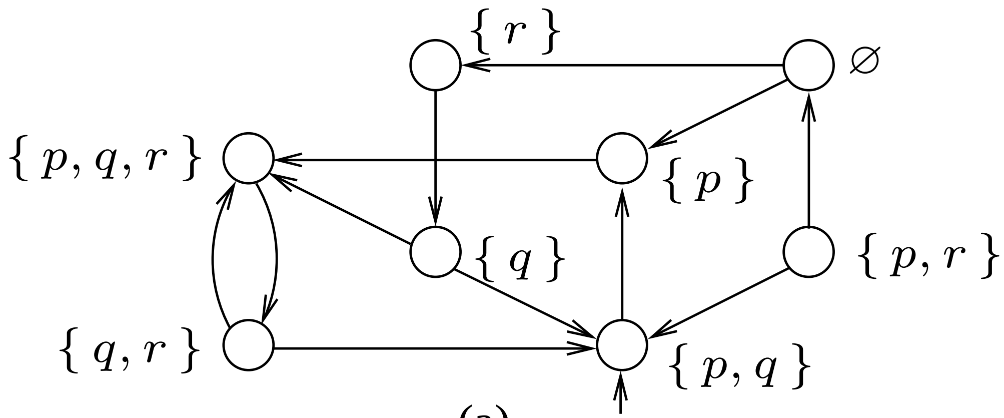
- non-trivial SCC = maximal, connected subgraph with at least one transition

$\Rightarrow$ any state in such SCC satisfies $\exists\Box\,\Phi$

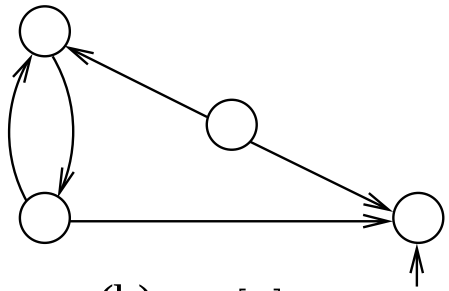
3. $s \models \exists\Box\,\Phi$ is equivalent to “some *SCC is reachable* from s”

- this search can be done in a backward manner

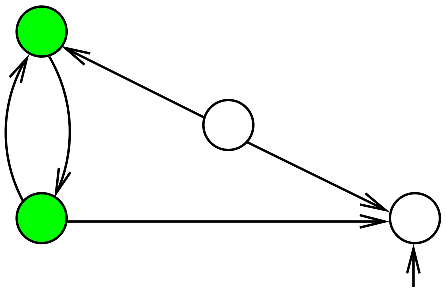
Example



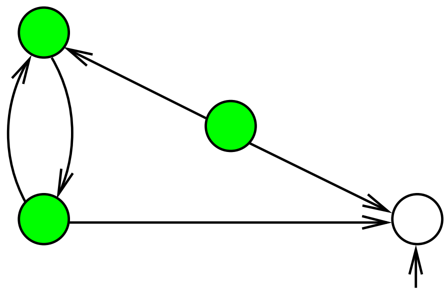
(a)



(b) $\mathit{TS}[q]$



(c) SCC



(d)

Overview Lecture #19

- Existential normal form
- Basic CTL model-checking algorithm
- Algorithms for $\exists(\Phi \, \mathsf{U} \, \Psi)$ and $\exists\Box\,\Phi$

$\Rightarrow$ Time complexity

Time complexity

For transition system *TS* with N states and K transitions,

and CTL formula Φ, the CTL model-checking problem *TS* $\models \Phi$

can be determined in time $\mathcal{O}(|\ \Phi\ |\cdot(N + M))$

this applies to both algorithms for $\exists\Box\ \Phi$

Model-checking LTL versus CTL

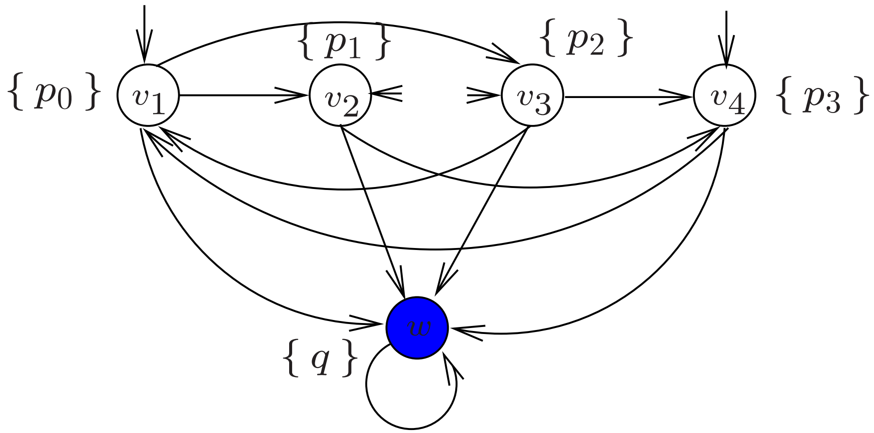
- Let *TS* be a transition system with N states and M transitions
- Model-checking LTL-formula Φ has time-complexity $\mathcal{O}((N+M) \cdot 2^{|\,\Phi\,|})$
  - linear in the state space of the system model
  - exponential in the length of the formula
- Model-checking CTL-formula Φ has time-complexity $\mathcal{O}((N+M) \cdot |\,\Phi\,|)$
  - linear in the state space of the system model and the formula
- Is model-checking CTL more efficient?

Model-checking LTL versus CTL

- Let *TS* be a transition system with N states and M transitions
- Model-checking LTL-formula Φ has time-complexity $\mathcal{O}((N+M) \cdot 2^{|\Phi|})$
  - linear in the state space of the system model
  - exponential in the length of the formula
- Model-checking CTL-formula Φ has time-complexity $\mathcal{O}((N+M) \cdot |\Phi|)$
  - linear in the state space of the system model and the formula
- Is model-checking CTL more efficient? No!

Hamiltonian path problem (1)

$\Rightarrow$ LTL-formulae can be *exponentially shorter* than their CTL-equivalent



- Existence of Hamiltonian path in LTL: $\bigwedge_i \Big(\Diamond p_i \;\wedge\; \Box (p_i \rightarrow \bigcirc \Box \neg p_i) \Big)$
- In CTL, all possible (= 4!) routes need to be encoded