Explicit State Model Checking Algorithm for CTL

CTL Model Checking Problem

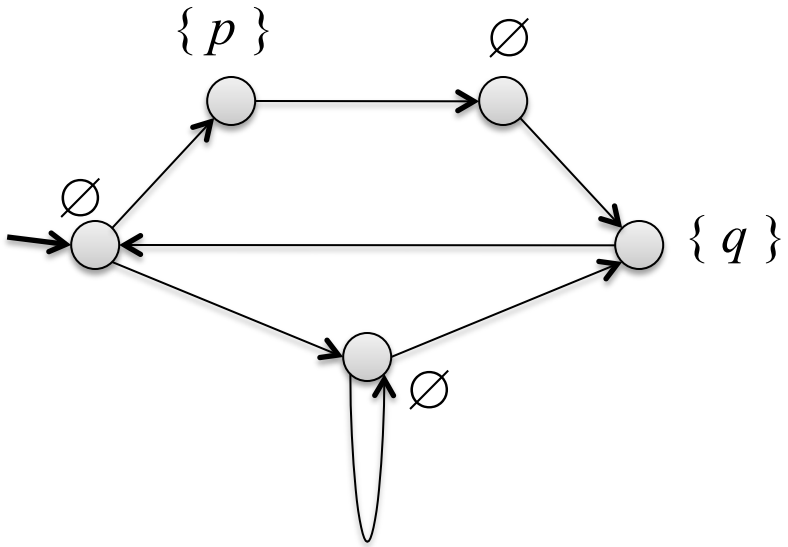
- Given
  - A model describing the behaviors of a system
  - A set of specifications expressed in CTL
- Algorithmically
  - Check that every behavior satisfies the specifications

Explicit-State MC Algorithm

- Operates on a FSM *M* of a system and a CTL specification *f* that has been put in a special form :
  - Uses only the operators: ¬, ∧, *EX*, *EU*, and *EG*
- The algorithm proceeds in stages
  - Each stage considers a sub-formula *f'* of *f*
  - Based on the Boolean structure of *f'* and knowing the states at which the sub-formulas of *f'* are true, determines the states at which *f'* is true
  - Starts with the "smallest" sub-formulas of *f* and works up to *f*

Example

System model:

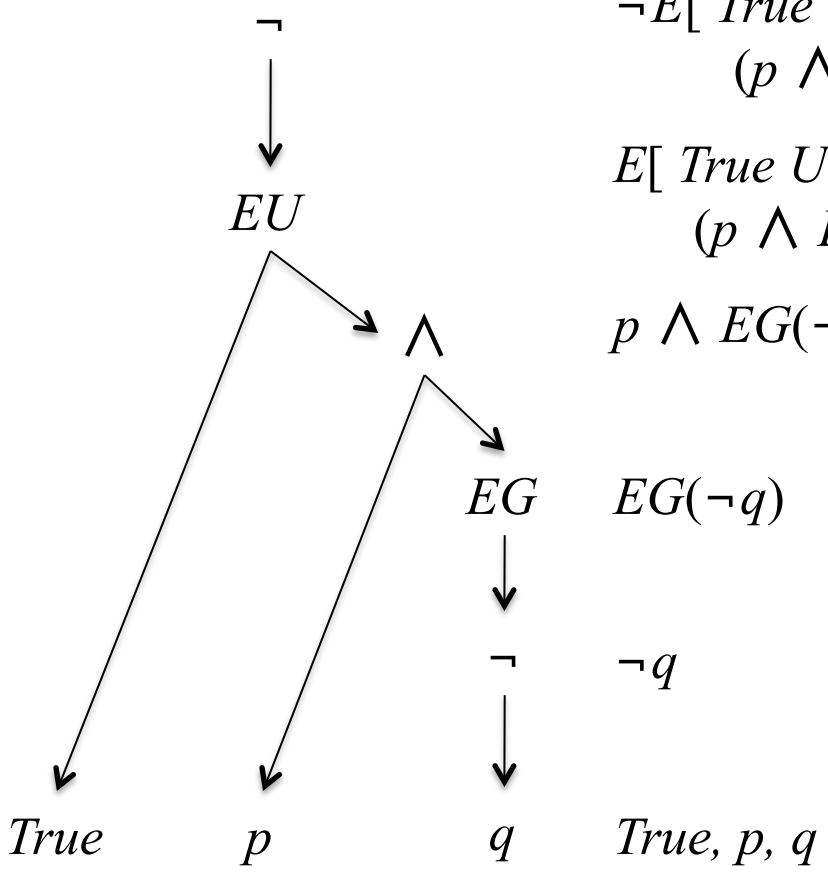


CTL Specification: $AG(\, p \Rightarrow AF(q)\,)$

Example

Rewrite spec to use only $\neg, \wedge, EX, EU, EG$:

$$AG(p \Rightarrow AF(q))$$
$$\equiv \neg EF[\neg(p \Rightarrow AF(q))]$$
$$\equiv \neg EF[p \wedge \neg AF(q)]$$
$$\equiv \neg E[True\ U\ (p \wedge \neg AF(q))]$$
$$\equiv \neg E[True\ U\ (p \wedge EG(\neg q))]$$



Sub-formulas considered, bottom up:

$\neg E[\ True\ U\ (p \wedge EG(\neg q))\ ]$

$E[\ True\ U\ (p \wedge EG(\neg q))\ ]$

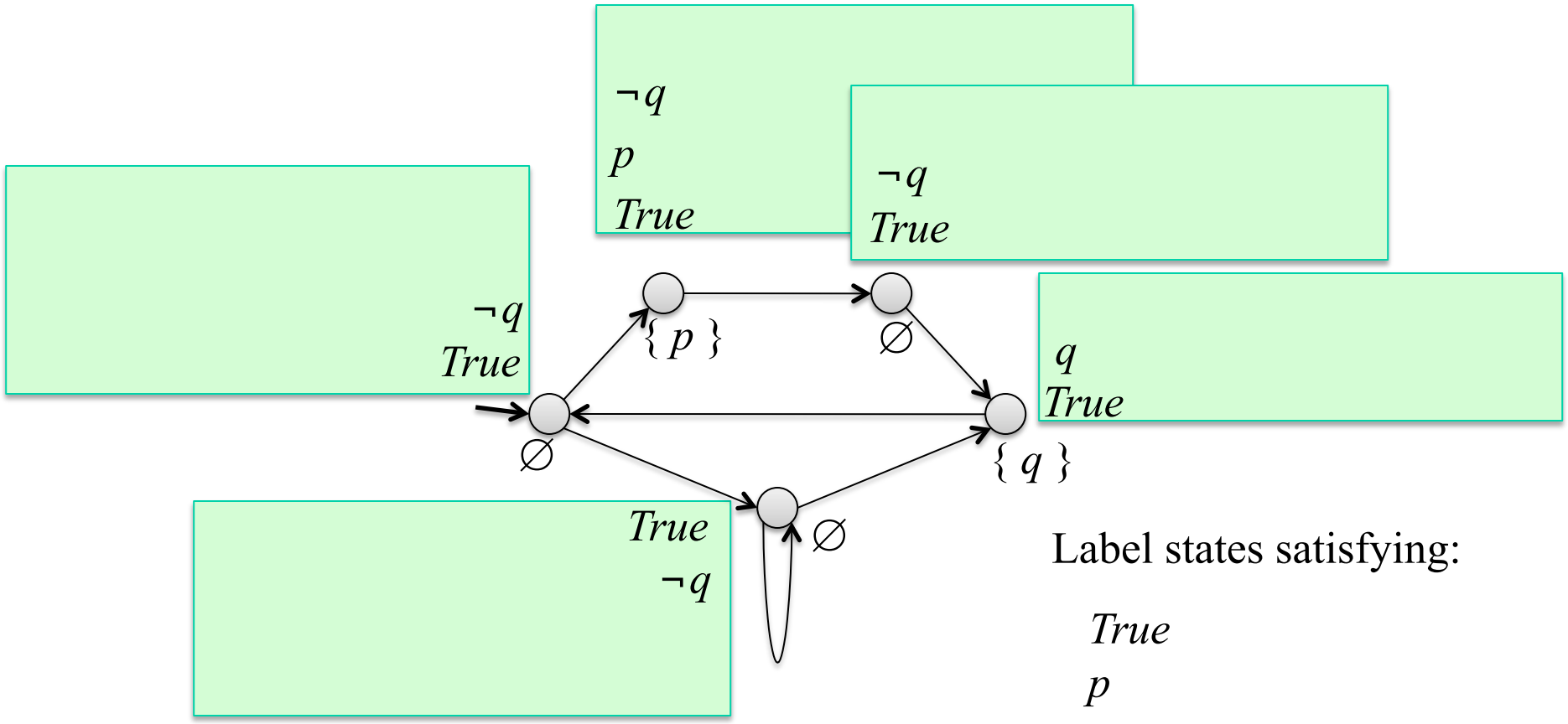
$p \wedge EG(\neg q)$

$EG(\neg q)$

$\neg q$

$True, p, q$

Example

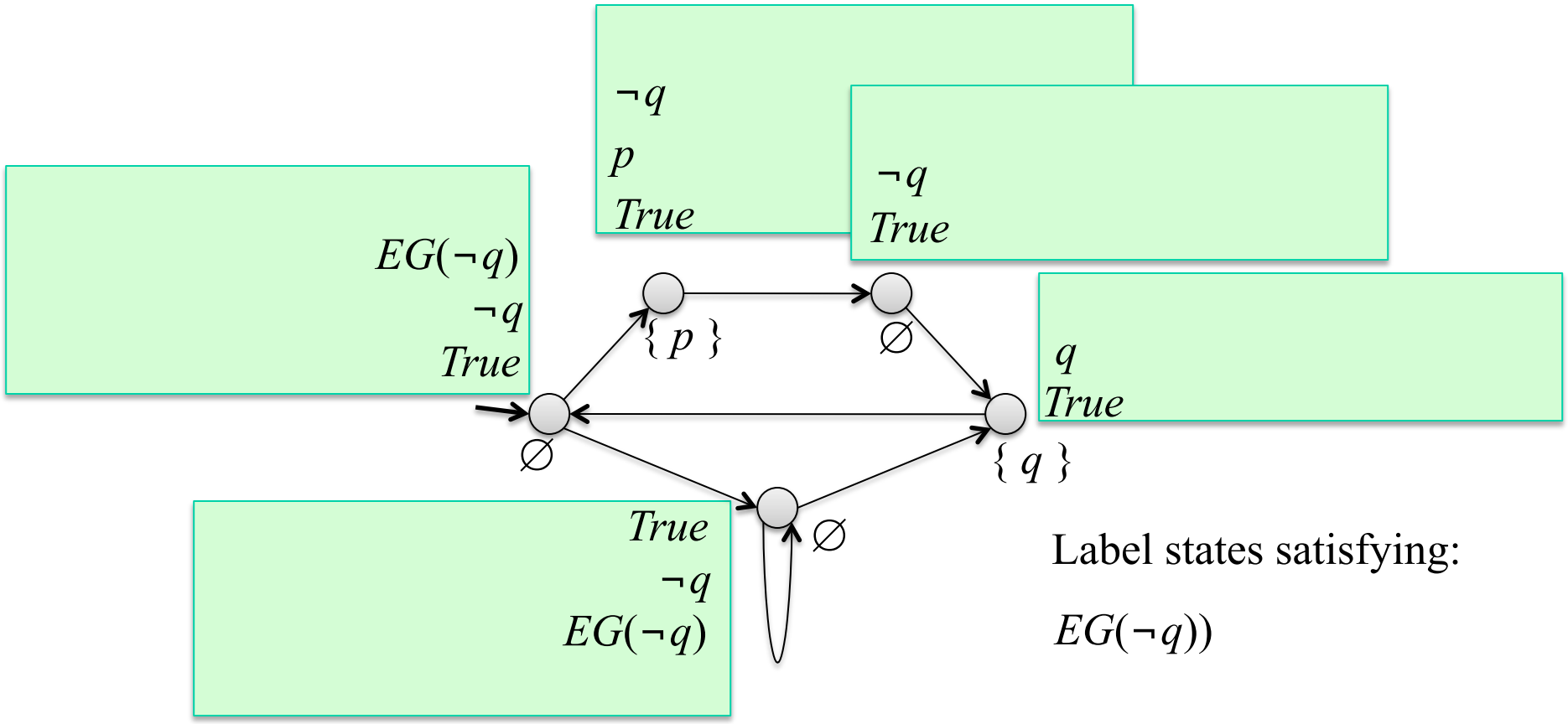


Label states satisfying:

True
p
q
¬*q*

Specification: $\neg E[\ True\ U\ (p\ \wedge\ EG(\neg q))\ ]$

Example

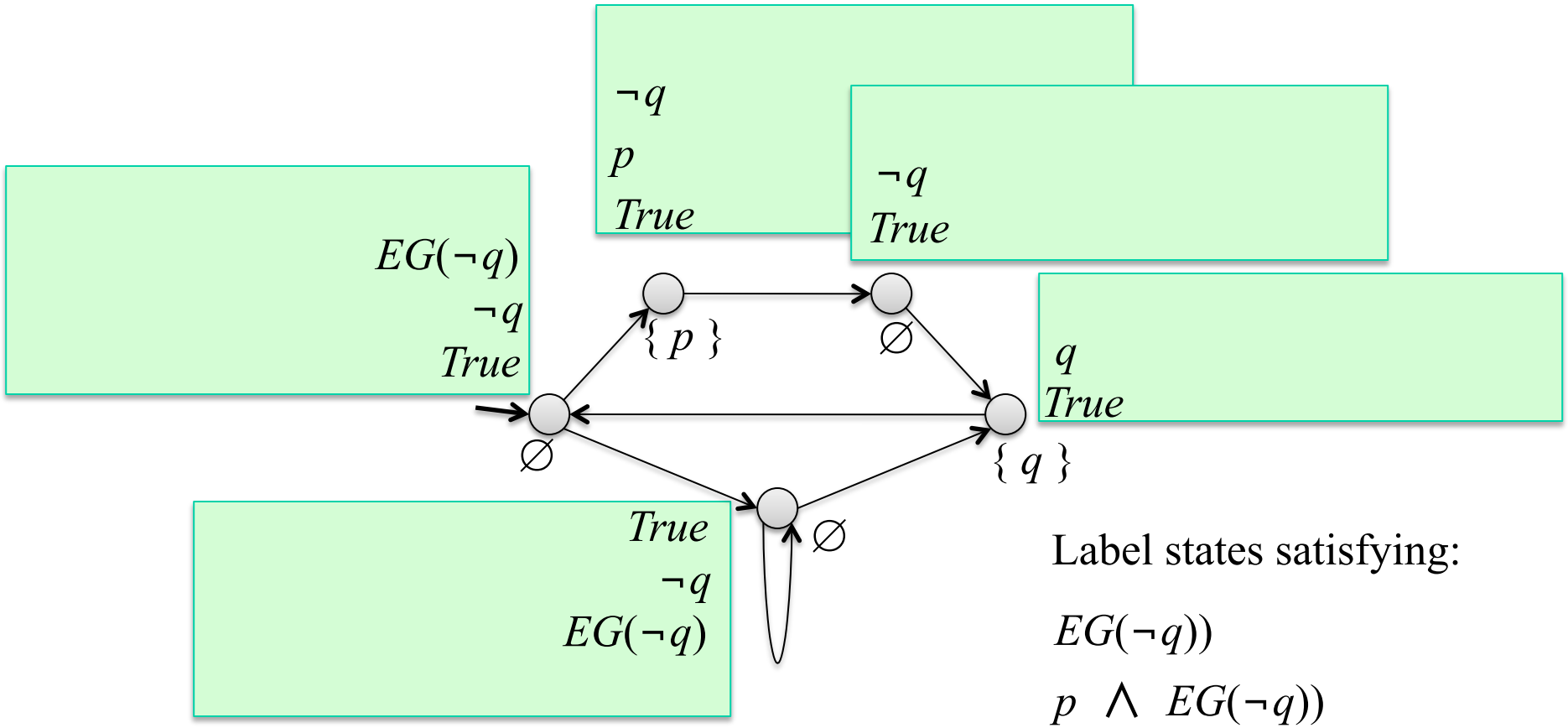


Label states satisfying:

$EG(\neg q))$

Specification: $\neg E[\ True\ U\ (p\ \wedge\ EG(\neg q))\ ]$

Example



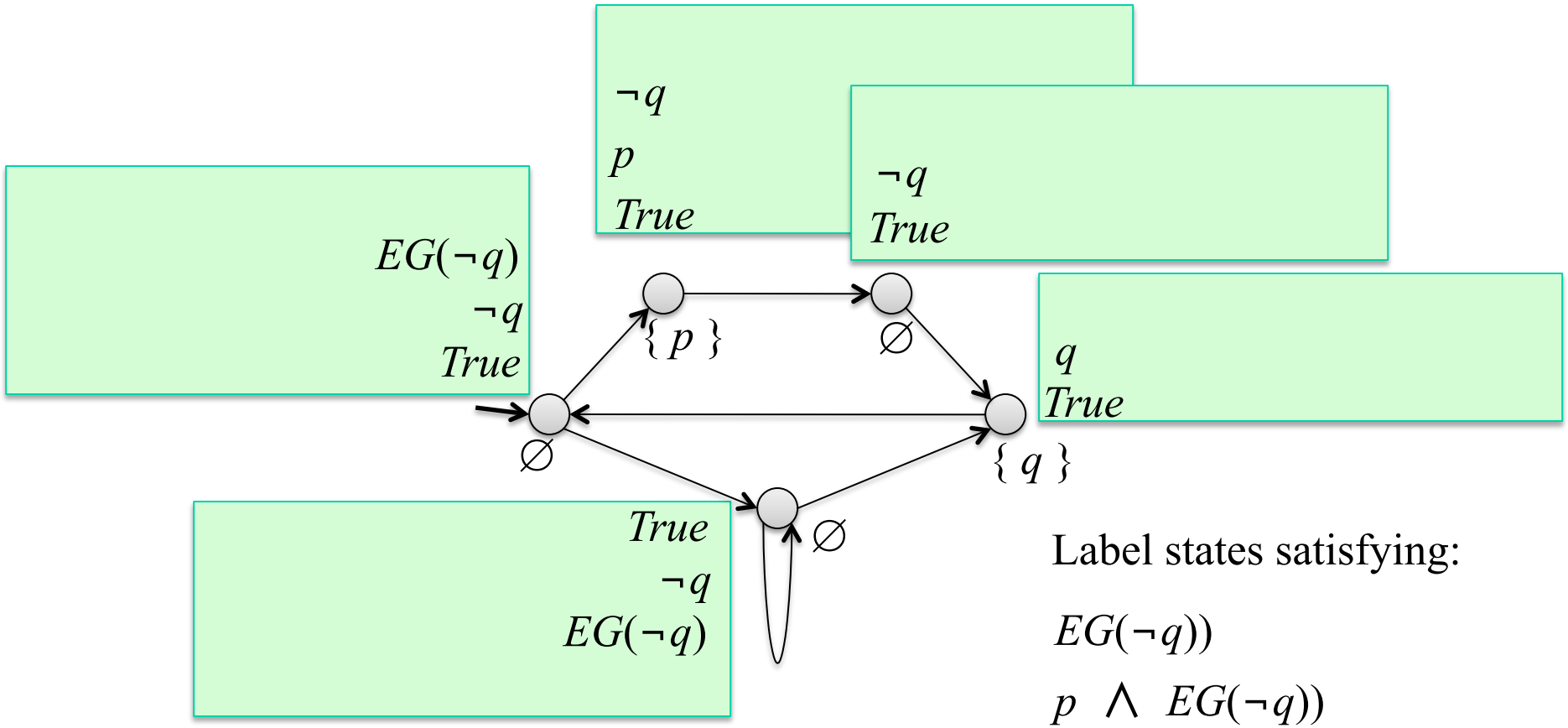
Label states satisfying:

$EG(\neg q))$

$p \wedge EG(\neg q))$

Specification: $\neg E[\ True\ U\ (p \wedge EG(\neg q))\ ]$

Example



Label states satisfying:

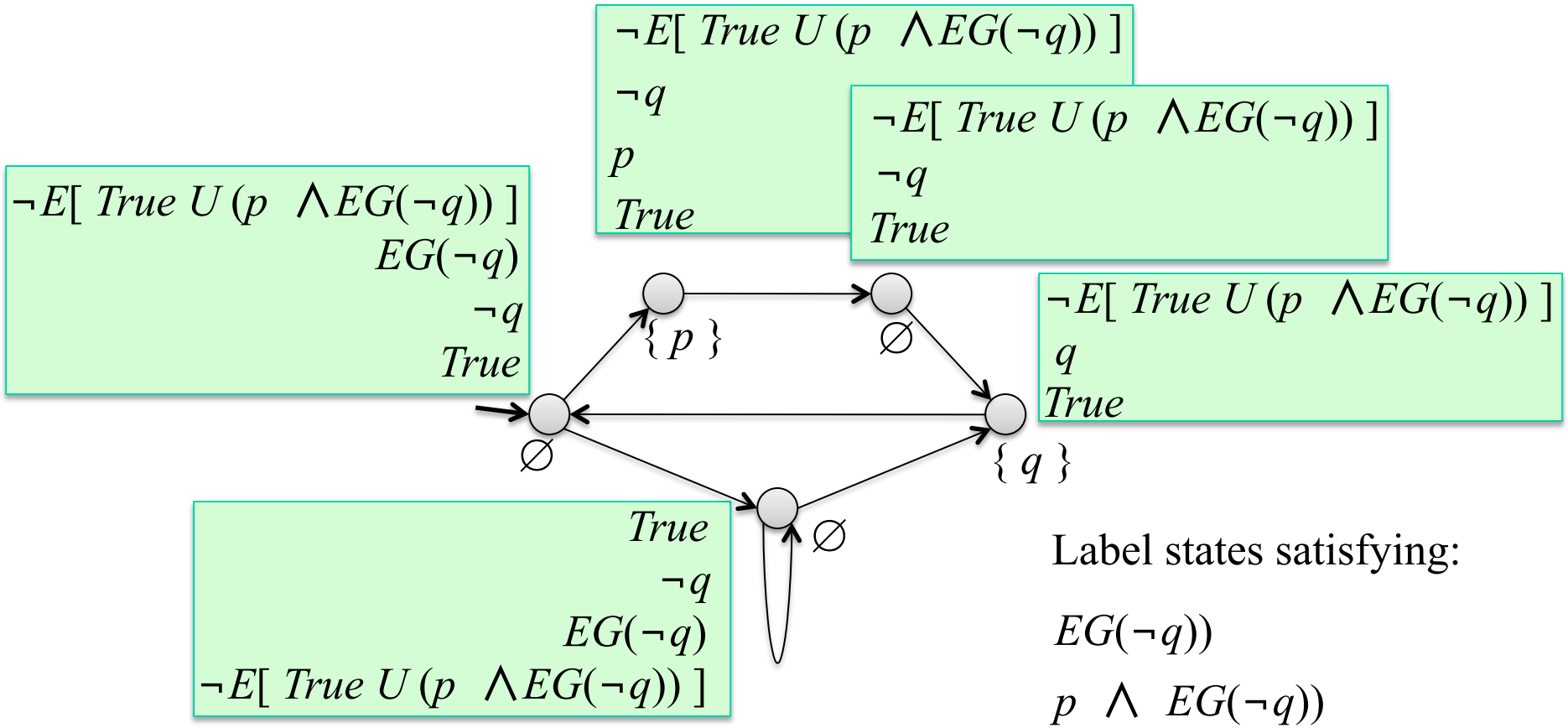
$EG(\neg q))$

$p \wedge EG(\neg q))$

$E[\ True\ U\ (p \wedge EG(\neg q))\ ]$

Specification: $\neg E[\ True\ U\ (p \wedge EG(\neg q))\ ]$

Example



Label states satisfying:

$EG(\neg q))$

$p \wedge EG(\neg q))$

$E[\ True\ U\ (p \wedge EG(\neg q))\ ]$

$\neg E[\ True\ U\ (p \wedge EG(\neg q))\ ]$

Specification: $\neg E[\ True\ U\ (p \wedge EG(\neg q))\ ]$

CTL MC Algorithm

Input: FSM $M = (S, R, L)$ and CTL spec f

Algorithm: Compute $Labels(s)$ for each $s \in S$:

- For each $s \in S$, initialize $Labels(s) = L(s)$
- For $i = 1$ .. d, where d is the depth of the parse tree of f :
  - For each sub-formula g at depth $d - i$, perform the "stage computation"
- On termination, $(M, s) \models f$ iff $f \in Labels(s)$

Stage Computation for CTL formulas

Five cases: $\neg$, $\wedge$, *EX*, *EU*, *EG*

- Case g has the form $\neg f$:
  If $f \notin Labels(s)$, then add g to $Labels(s)$
- Case g has the form $f \wedge g$:
  If $f \in Labels(s)$ and $g \in Labels(s)$, then add g to $Labels(s)$
- Case g has the form EXf :
  If there is some some t such that $(s, t) \in R$ and
  $f \in Labels(t)$, then add g to $Labels(s)$

Stage Computation for CTL formulas

Five cases: ¬, ∨, *EX*, *EU*, *EG*

- Case g has the form E[f *U* h]
  - If $h \in Labels(s)$, then add g to $Labels(s)$
  - Repeat until a fixed point is reached:
    If there is some some t such that
    $(s, t) \in R$ and $f \in Labels(s)$ and $g \in Labels(t)$,
    then
    add g to $Labels(s)$

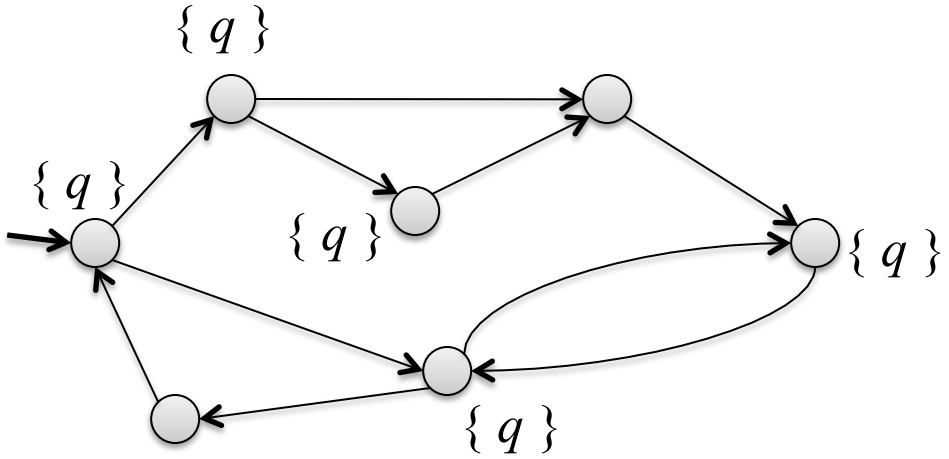
Stage Computation for CTL formulas

Five cases: ¬, ∨, *EX*, *EU*, *EG*

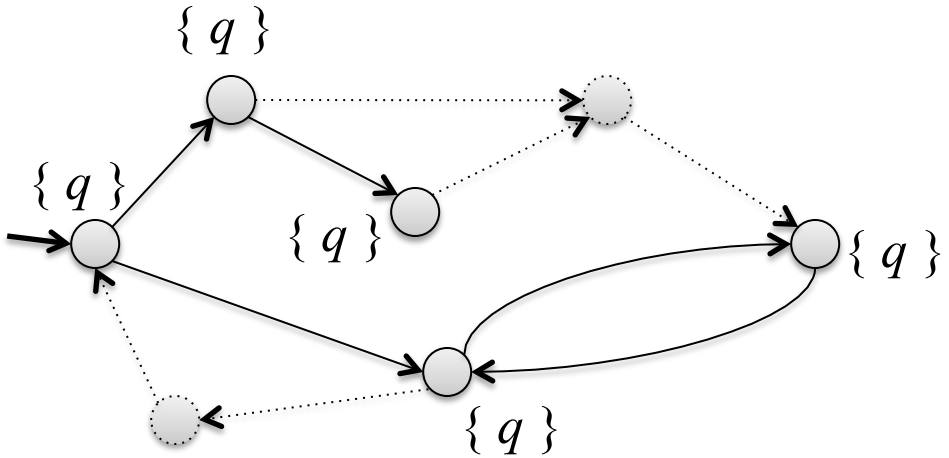
- Case *g* has the form *EG*(*f*):
  - Compute the sub-FSM *M'* of *M* produced by deleting all states for which $f \notin Labels(s)$.
  - Find the maximal strongly connected components (SCCs) of *M'*
  - For each non-trivial SCC of *M'* and every *s* in *M'*, if *s* reaches the SCC, add *g* to *Labels*(*s*)

Example: $EG(q)$

Given FSM M :

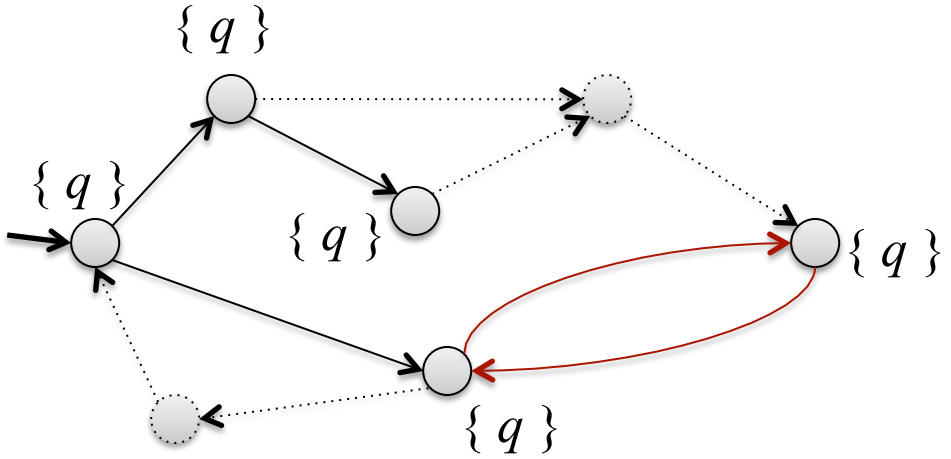


Compute sub-FSM M':

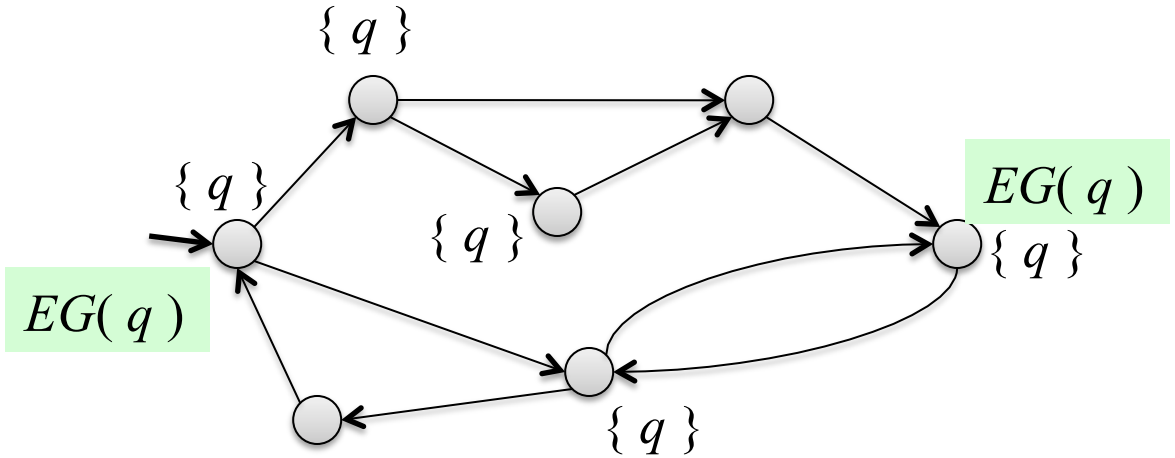


Example: $EG(q)$

Compute max'l SCCs of M':



Compute states that reach the SCC :



Summary

- Stage computation labels each state of the FSM with sub-formulas that are true at that state
  - sub-formulas considered bottom-up
  - based on Boolean structure of sub-formulas
- Complexity is linear in the size of the formula and the size of the FSM
- Explicit-state because it needs to examine every state in a FSM, which is typically a source of state explosion