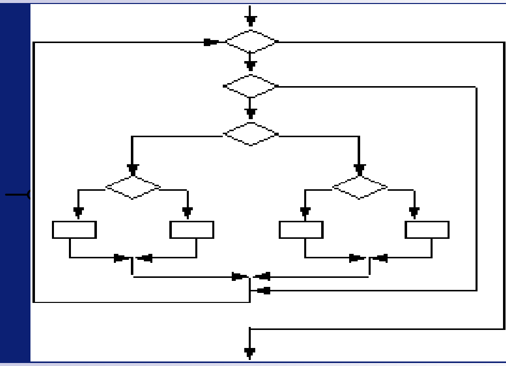
Spring 2007



Program correctness

Model-checking CTL

Marcello Bonsangue

Formal Verification

Verification techniques comprise

- a modelling framework M, Γ

  to describe a system
- a specification language ϕ

  to describe the properties to be verified
- a verification method $M \models \phi, \Gamma \vdash \phi$

  to establish whether a model satisfies a property

Today for CTL

Model Checking

- Question: does a given transition system satisfies a temporal formula?
- Simple answer: use definition of $\models$!
  - We cannot implement it as we have to unwind the transition system in a possibly infinite tree

Can we do better?

and most probably!

The problem

- We need efficient algorithms to solve the problems

  [1] $M,s \overset{?}{\models} \phi$

  [2] $M,\overset{?}{s} \models \phi$

  where M should have finitely many states, and ϕ is a CTL formula.

- We concentrate to solution of [2], as [1] can be easily derived from it.

The solution

- Input: A CTL model M and CTL formula ϕ
- Output: The set of states of M which satisfy ϕ
- Basic principles:
  - Translate any CTL formula ϕ in terms of the connectives AF, EU,EX, $\wedge$,$\neg$, and $\perp$.
  - Label the states of M with sub-formulas of ϕ that are satisfied there, starting from the smallest sub-formulas and working outwards towards ϕ
  - Output the states labeled by ϕ

The labelling

- An immediate sub-formula of a formula ϕ is any maximal-length formula ψ other than ϕ itself
- Let ψ be a sub-formula of ϕ and assume the states of M have been already labeled by all immediate sub-formulas of ψ.
- Which states have to be labeled by ψ?

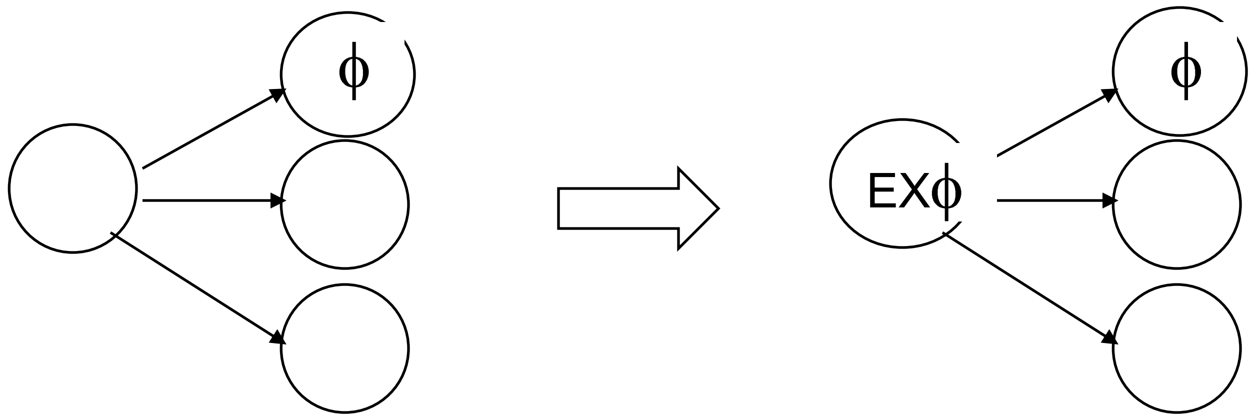
We proceed by case analysis

The basic labeling

- $\perp$ no states are labeled
- p label a state s with p if $p \in l(s)$
- $\phi_1 \wedge \phi_2$ label a state s with $\phi_1 \wedge \phi_2$ if s is already labeled with ϕ_1 and ϕ_2
- $\neg\phi$ label a state s with $\neg\phi$ if s is not already labeled with ϕ

The EX labeling

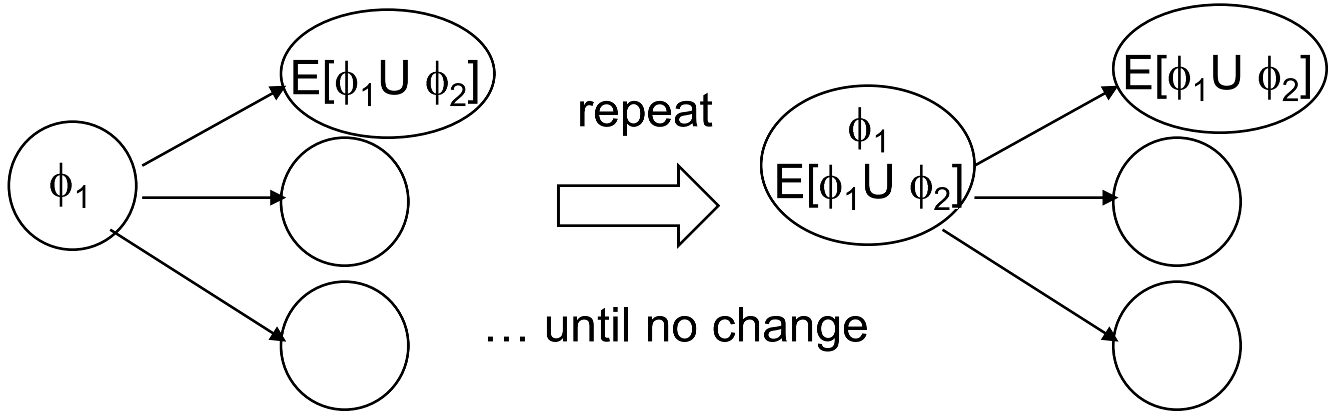
- EXϕ Label with EXϕ any state s with one of its successors already labeled with ϕ



The EU labeling

- $E[\phi_1 U \phi_2] \equiv \phi_2 \vee (\phi_1 \wedge EXE[\phi_1 U \phi_2])$

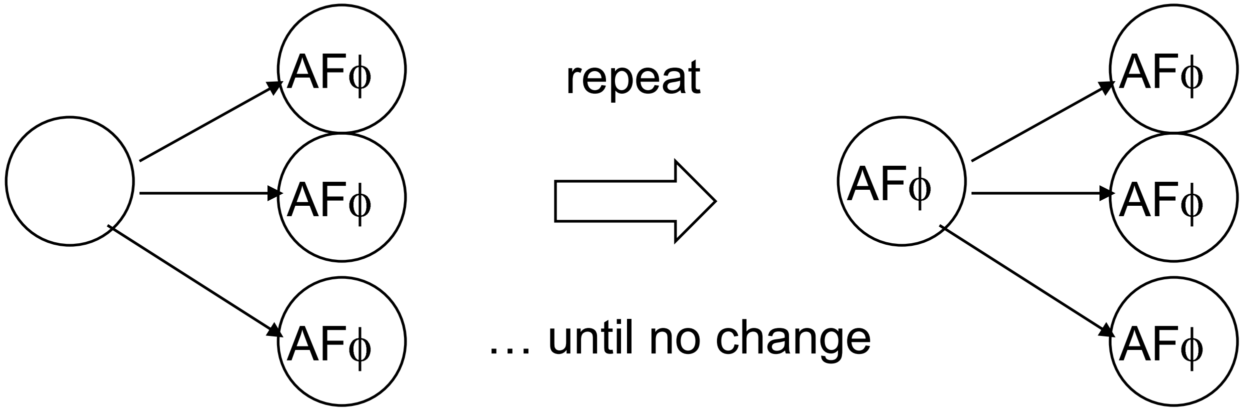
1. Label with $E[\phi_1 U \phi_2]$ any state s already labeled with ϕ_2
2. Repeat until no change: label any state s with $E[\phi_1 U \phi_2]$ if s is labeled with ϕ_1 and at least one of its successor is already labeled with $E[\phi_1 U \phi_2]$



The AF labeling

- $AF\phi \equiv \phi \vee AXAF\phi$

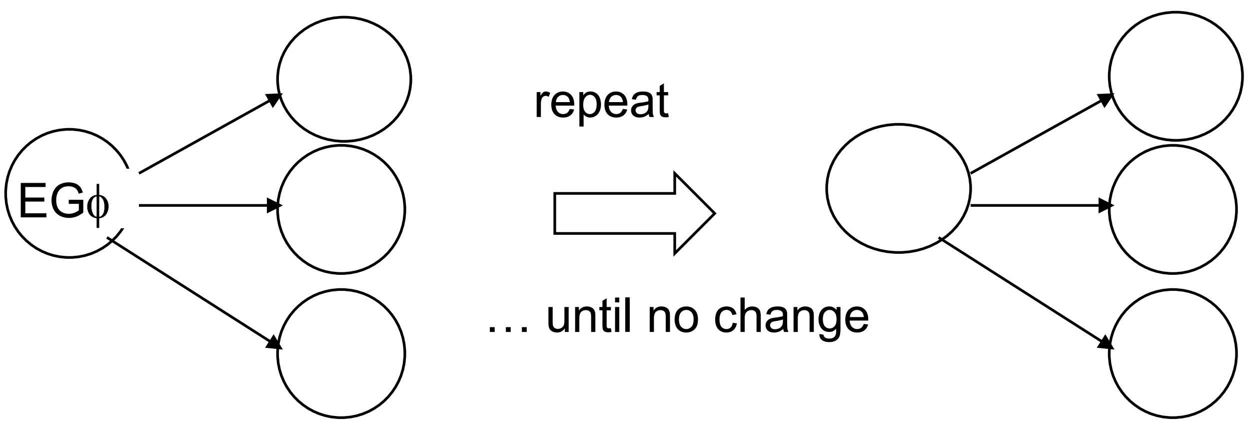
1. Label with $AF\phi$ any state s already labeled with ϕ
2. Repeat until no change: label any state s with $AF\phi$ if all successors of s are already labeled with $AF\phi$



The EG labeling (direct)

- $EG\phi \equiv \phi \wedge EXEG\phi \equiv \neg AF\neg\phi$

1. Label all the states with $EG\phi$
2. Delete the label $EG\phi$ from any state s not labeled with ϕ
3. Repeat until no change: delete the label $EG\phi$ from any state s if none of its successors is labeled with $EG\phi$



Complexity

The complexity of the model checking algorithm is

$$\mathbf{O}(f * V * (V+E))$$

where f = number of connectives in ϕ

V = number of states of M

E = number of transitions of M

It can be easily improved to an algorithm linear both in the size of the formula and of the model

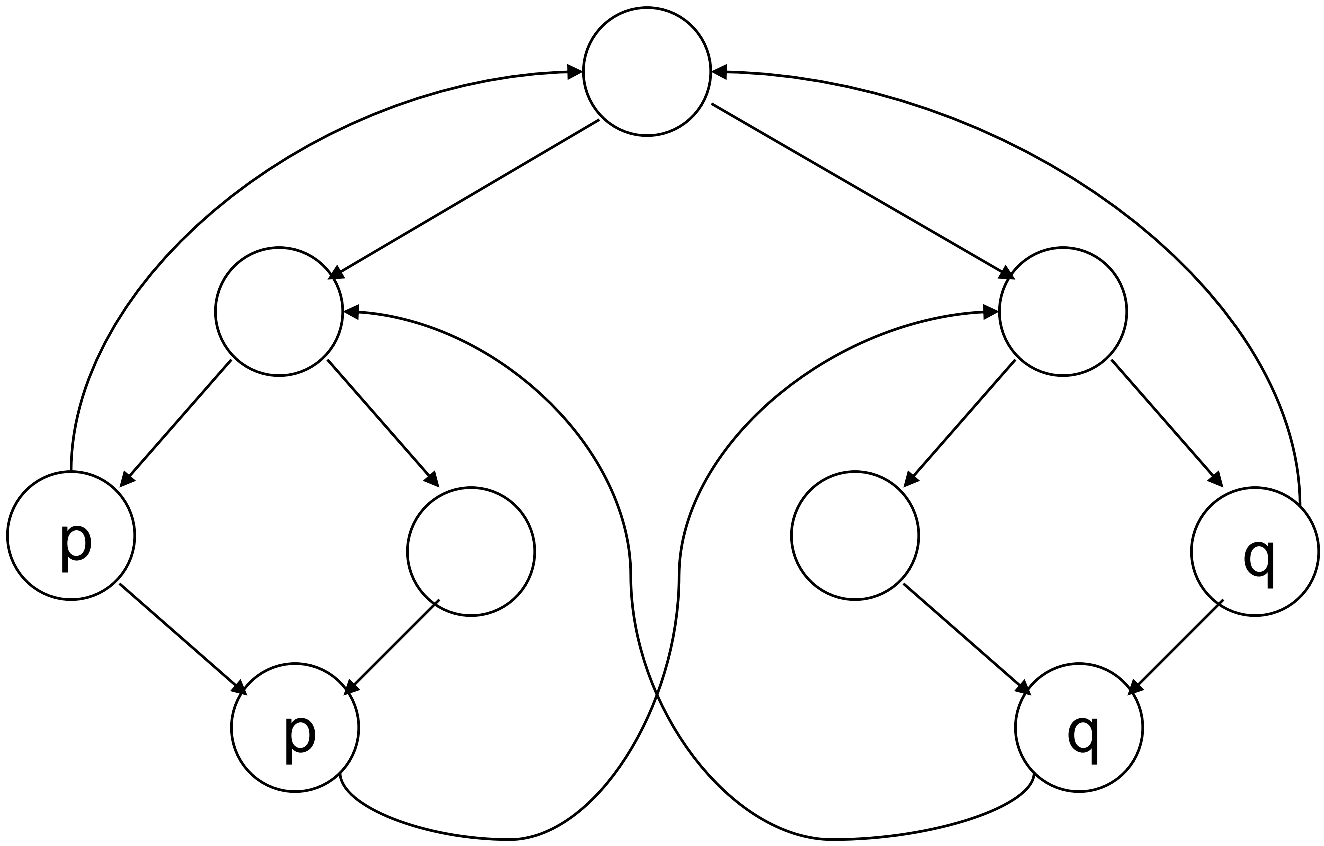
State explosion

- The algorithm is linear in the size of the model but the size of the model is exponential in the number of variables, components, etc.

Can we reduce state explosion?

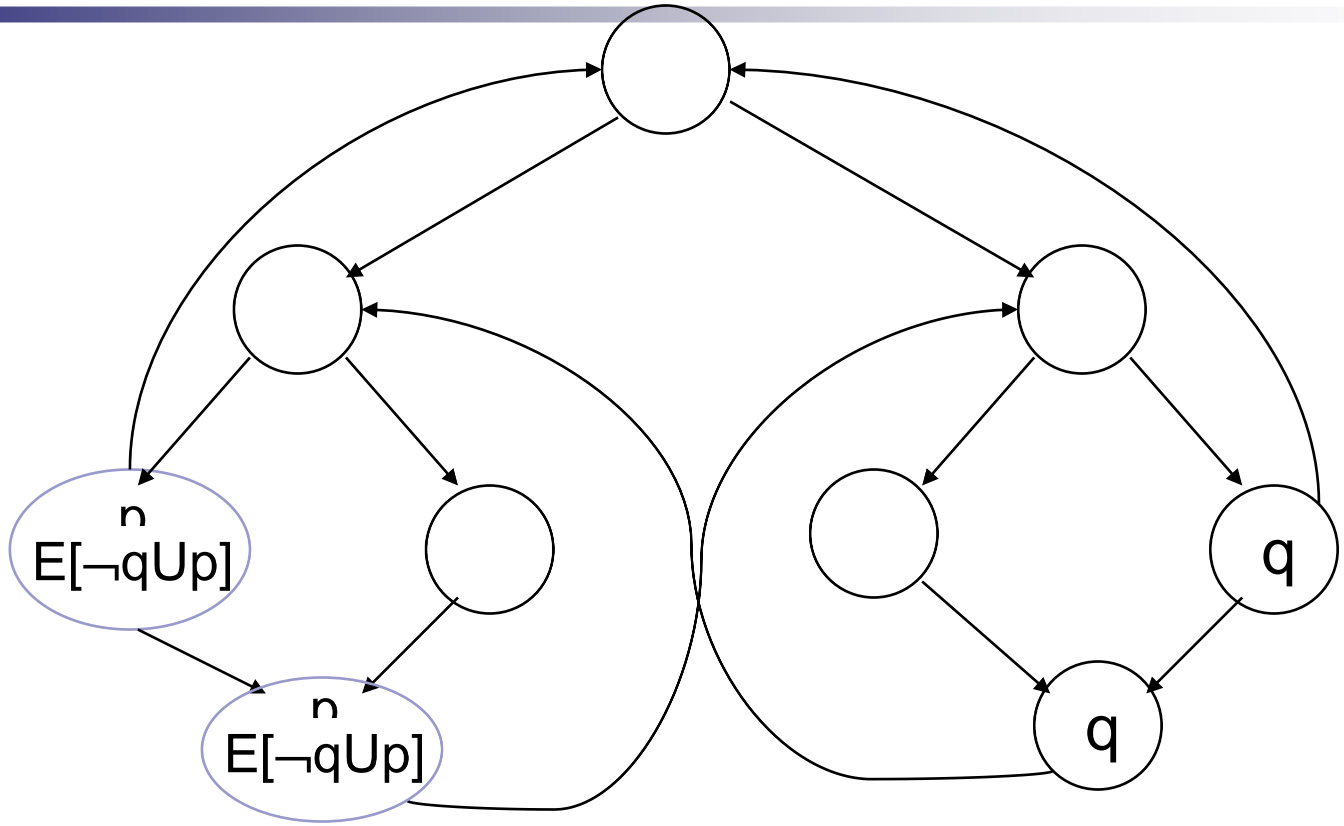
- Abstraction (what is relevant?)
- Induction (for 'similar' components)
- Composition (divide and conquer)
- Reduction (prove semantic equivalence)
- Ordered binary decision diagrams

Example: Input



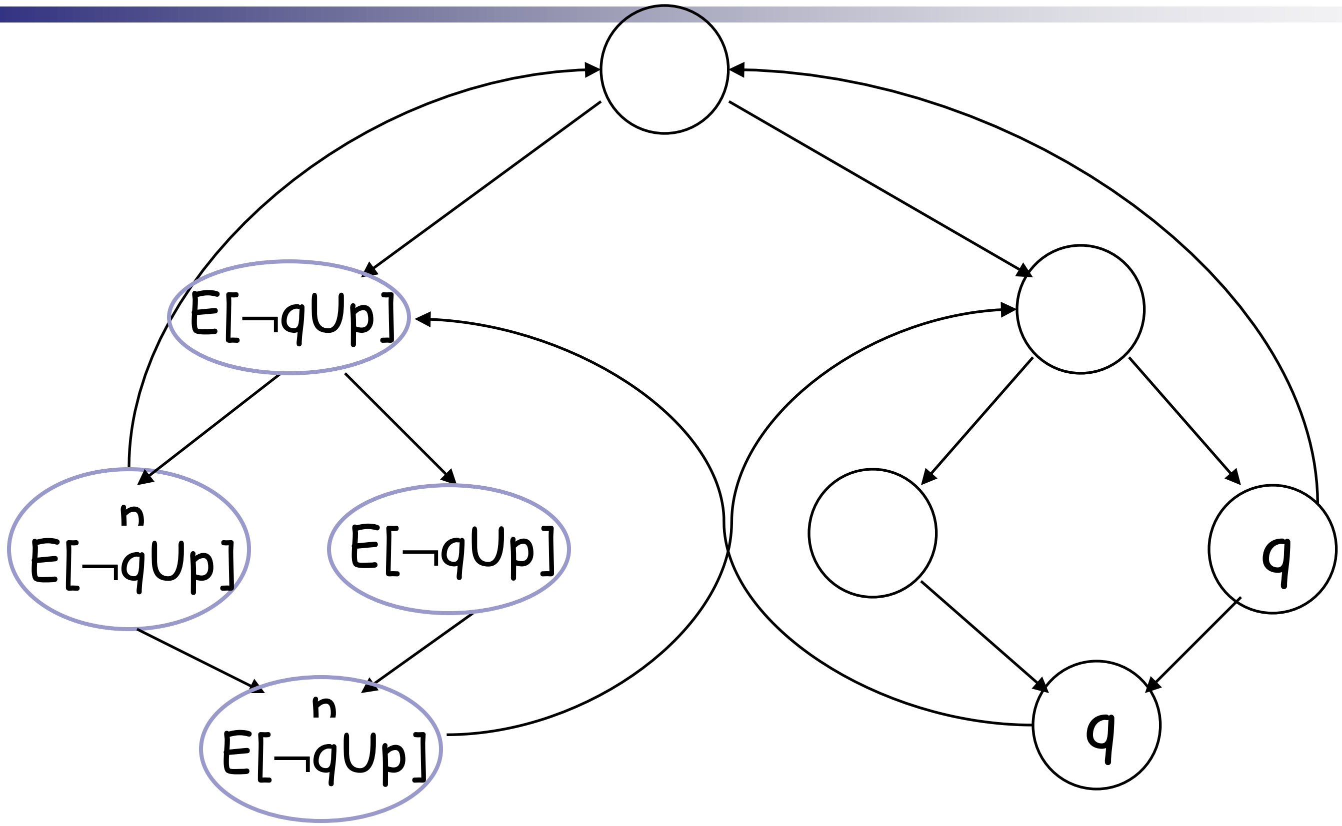
$$\phi = AF(E[\neg q \; U \; p] \vee EXq)$$

Example: EU - step 1



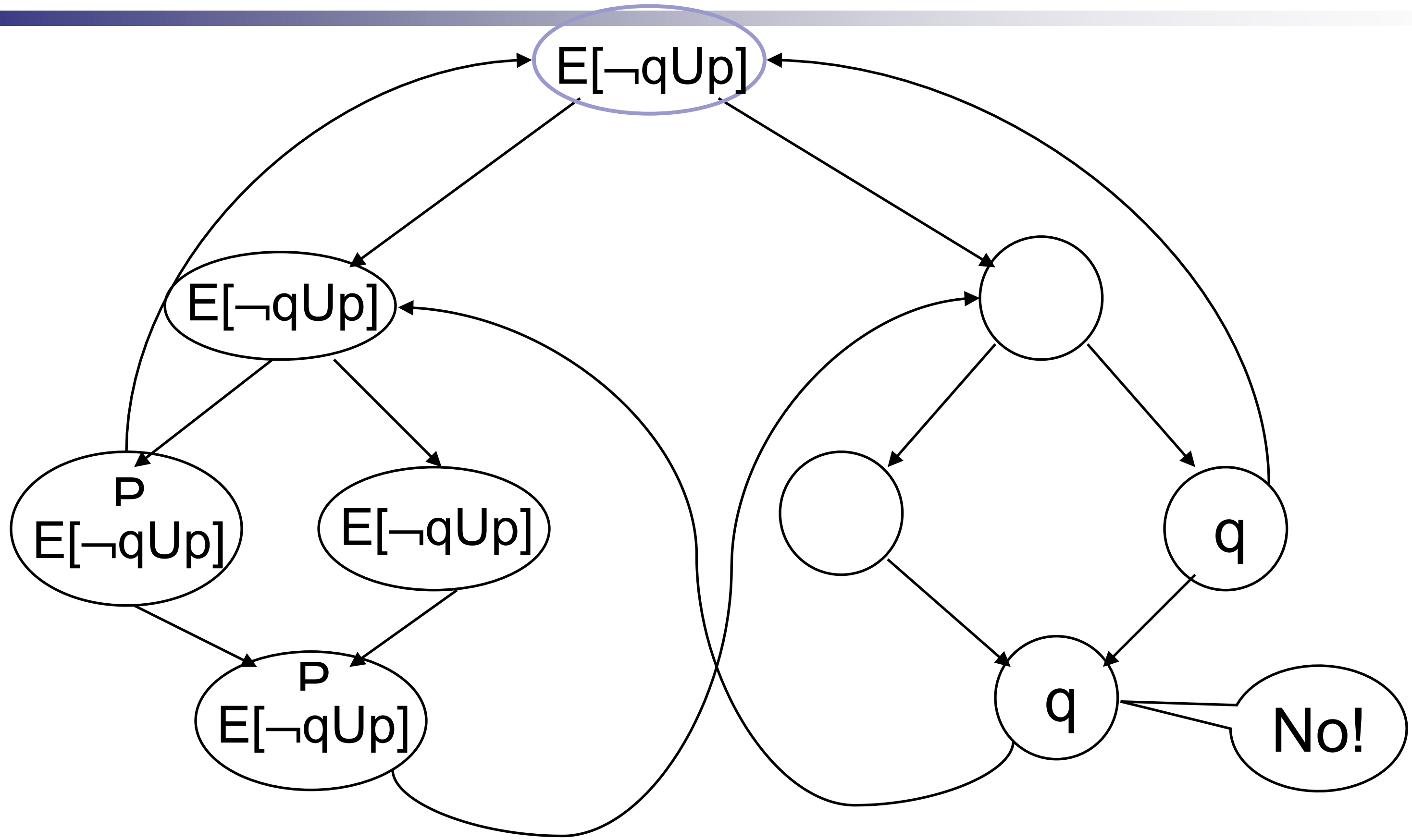
1. Label with E[¬qUp] all states which satisfy p

Example: EU-step 2.1



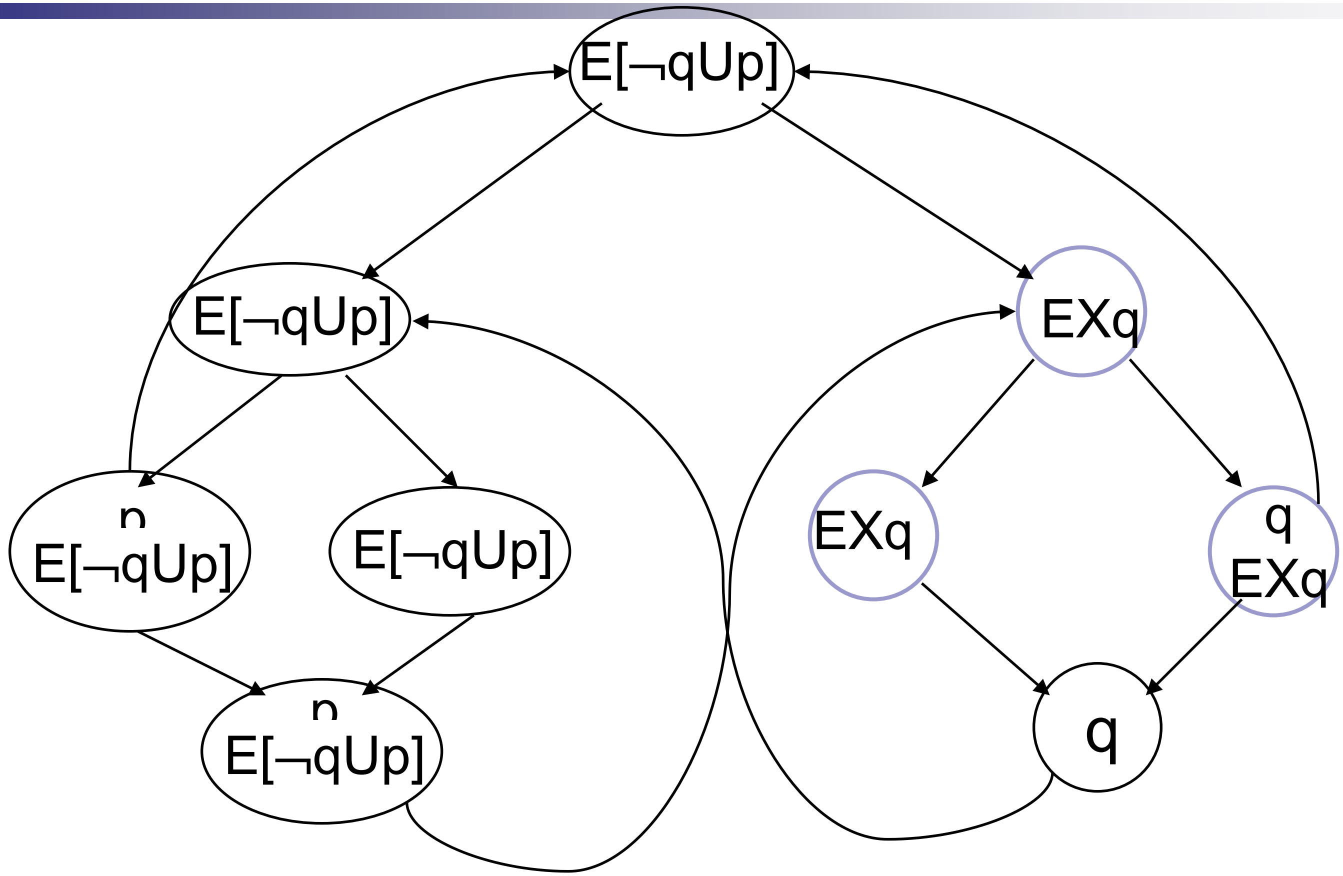
2.1 label with E[¬qUp] any state that is already labeled with ¬q and with one of its successor already labeled by E[¬qUp]

Example: EU-step 2.2



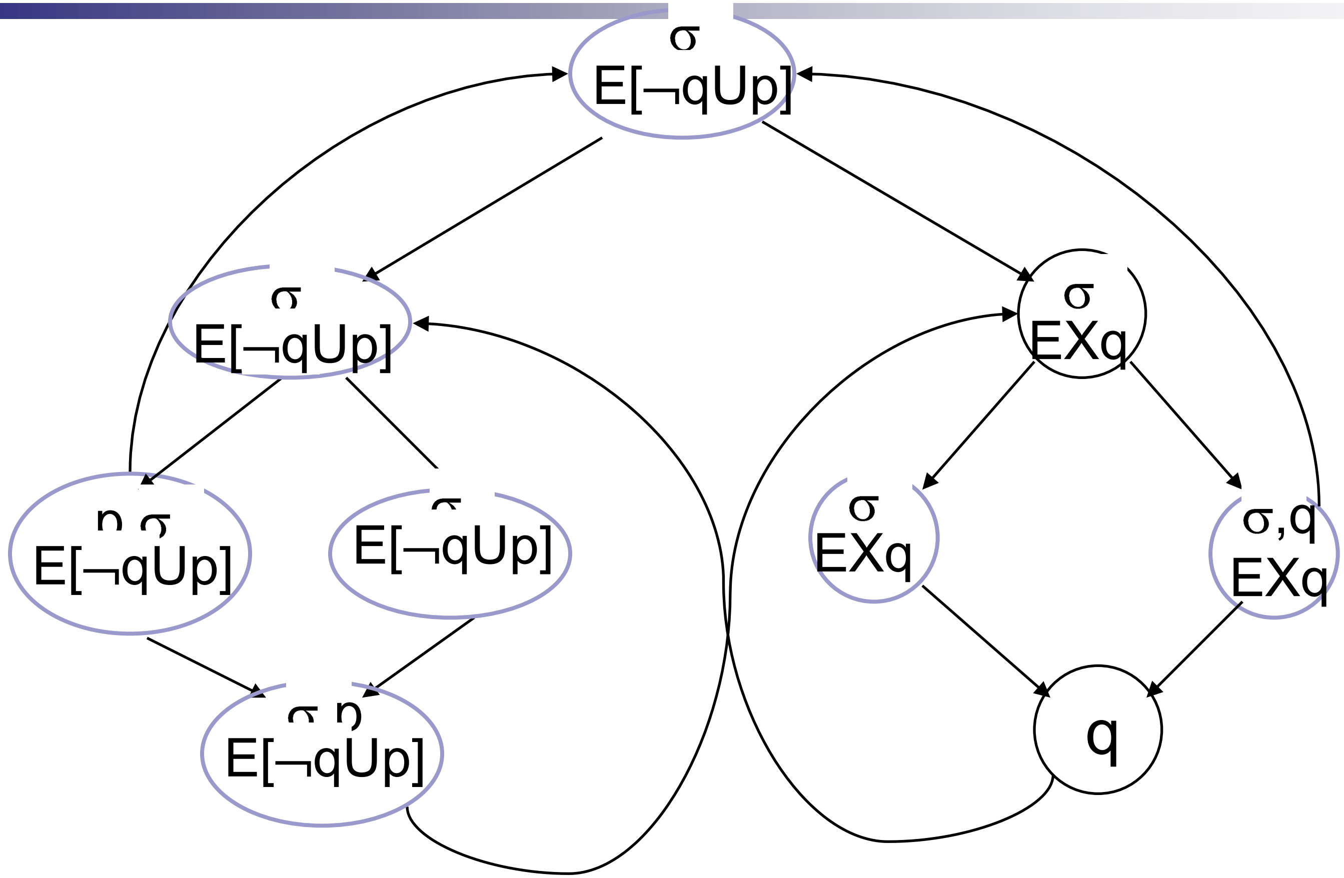
2.2 label with E[¬qUp] any state that is already labeled with ¬q and with one of its successor already labeled by E[¬qUp]

Example: EX-step 3



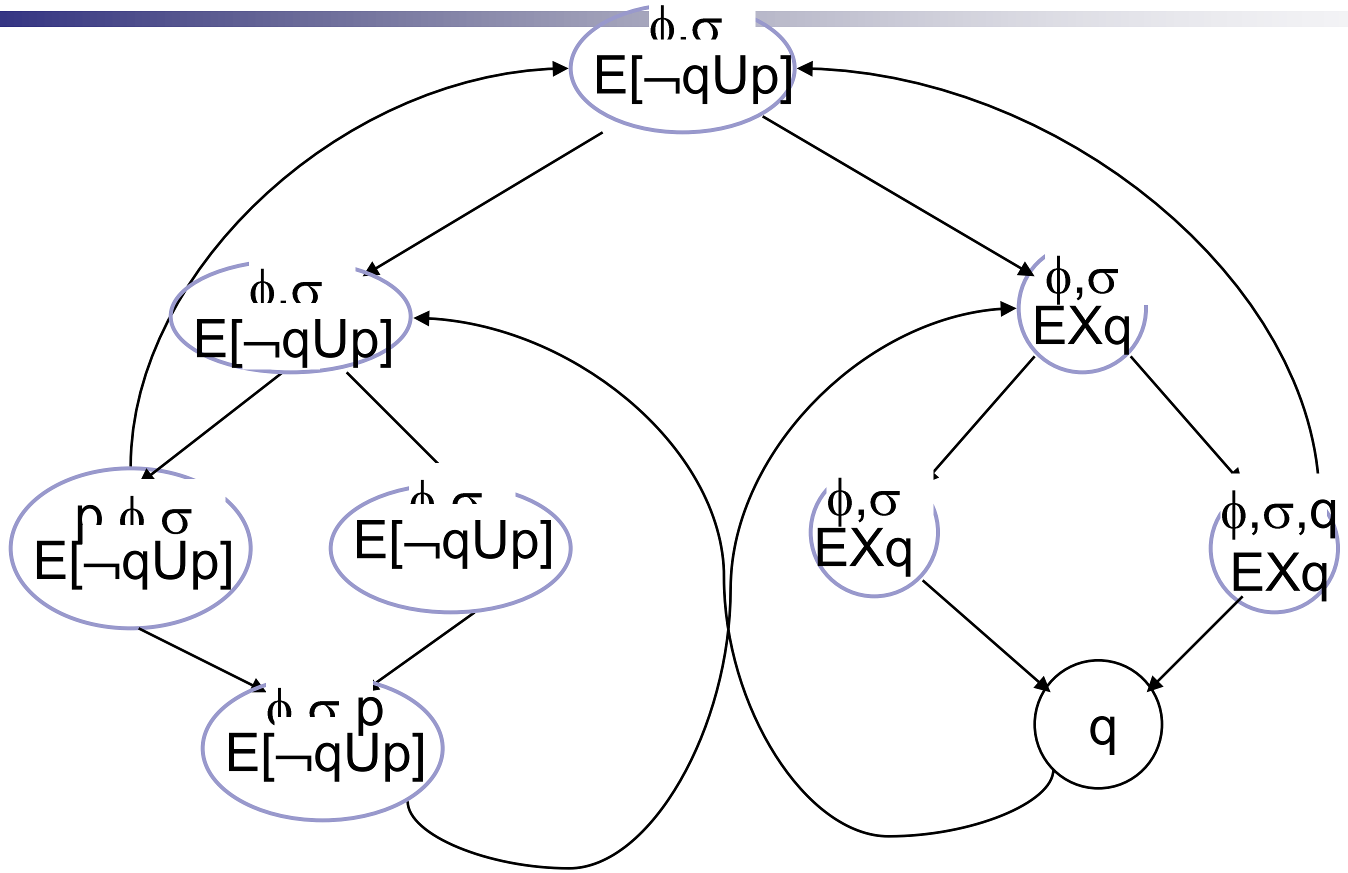
3. Label with EXq any state with one of it successors already labeled by q

Example: ∨-step 4



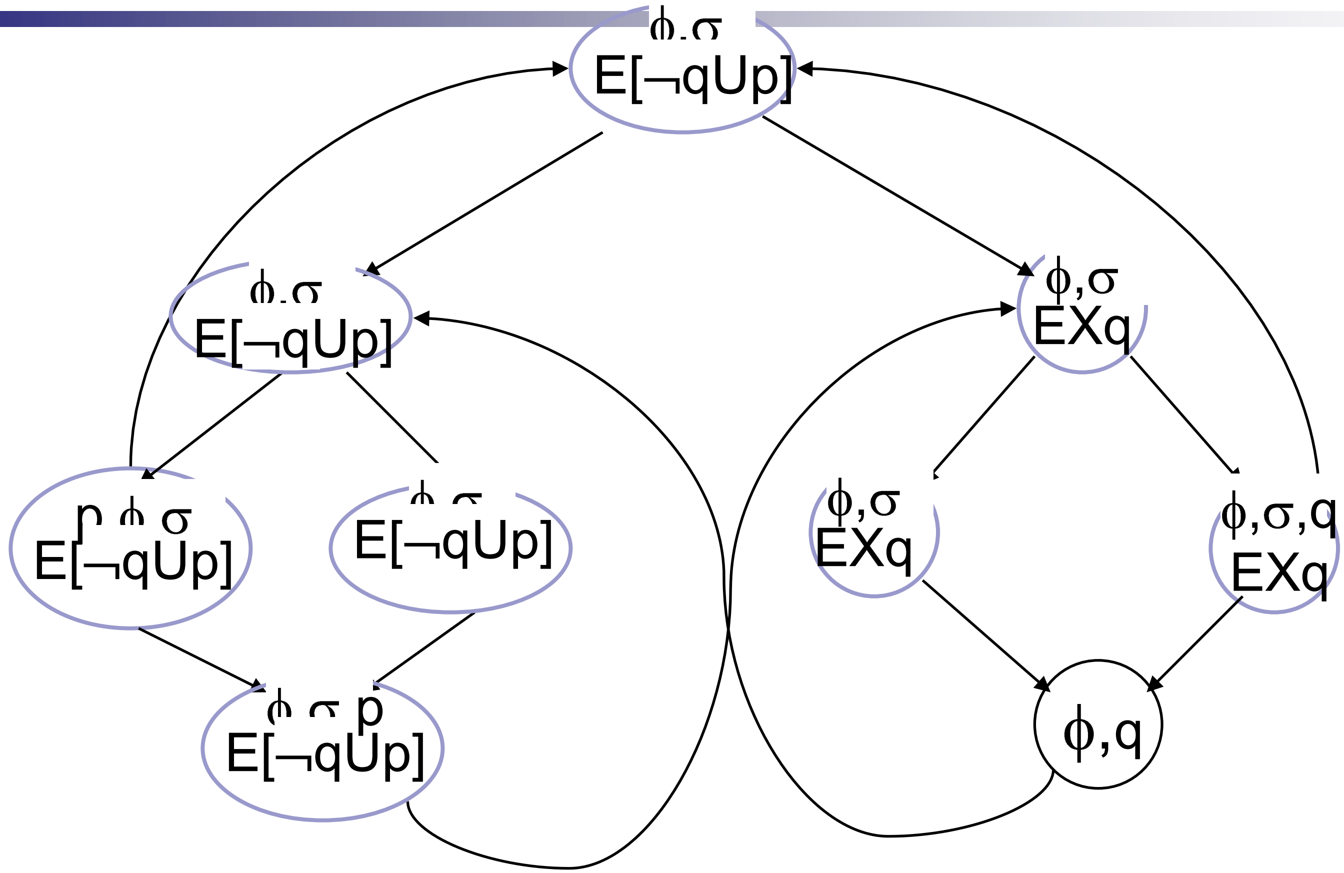
4. Label with σ = E[¬qUp] v EXq any state s already labeled by E[¬qUp] or EXq

Example: AF-step 5.1



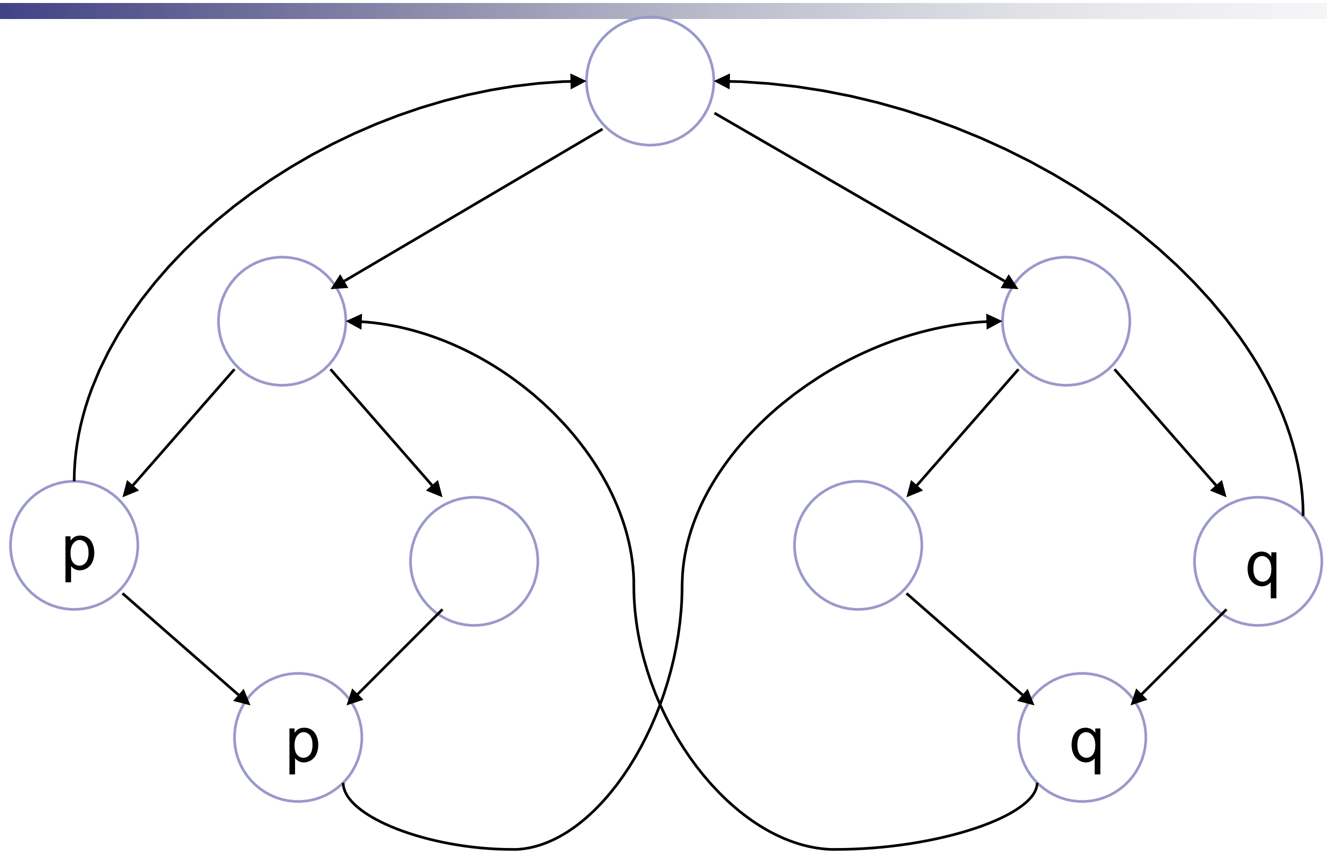
5.1 Label with ϕ = AF(E[$\neg$qUp]vEXq) any state already labeled by σ = E[$\neg$qUp]vEXq

Example: AF-step 5.2



5.2 Label with ϕ any state with all successor already labeled by ϕ.

Example: Output



- All states satisfy AF(E[¬q U p] v EXq)