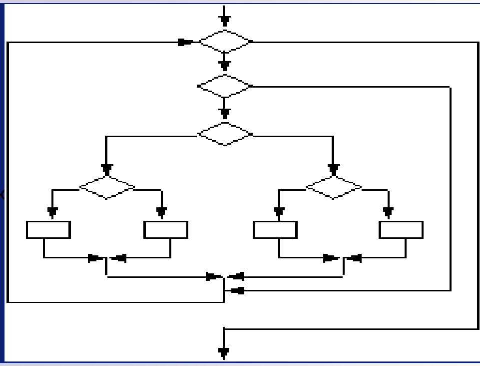
Spring 2007



Program correctness

Model checking LTL

Marcello Bonsangue

Leiden Institute of Advanced Computer Science
Research & Education

Context

- Model checking CTL was relatively easy because the truth of formulas depends

on the current state (CTL)

and not

on an execution path (LTL)

and not

on the tree of all executions (CTL*)

- Next we concentrate on model checking LTL

LTL: a recap

- Syntax

  $\phi ::= \top \mid p \mid \neg\phi \mid \phi\vee\phi \mid X\phi \mid \phi U\phi$

  All other connectives can be written in the above syntax

LTL formulas as languages (I)

- ϕ = GFp (infinitely often p)
  - The execution $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \ldots$ satisfies ϕ if it contains infinitely many $s_{n_1}, s_{n_2}, \ldots$ at which p holds. In between there can be an arbitrary but finite number of state at which $\neg$p holds.

As a language $((\neg p)^*.p)^\omega$

ω-regular expressions

$*$ = an arbitrary but finite number of repetitions

$^\omega$ = an infinite number of repetitions

LTL formulas as languages(II)

- ϕ = FGp (Eventually always p)
- The execution $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \ldots$ satisfies ϕ if from a certain state onwards at all states p holds.

- As ω-regular expression $(p + \neg p)^*.p^\omega$

Automata on finite words: a recap

- A non-deterministic finite automaton is a special kind of transition systems for recognizing languages on finite words

- <u>NF-automaton</u> $A = <\Sigma, S, \rightarrow, I, F>$
  - Σ finite alphabet
  - S finite set of states
  - $\rightarrow \subseteq S \times \Sigma \times S$ transition relation
  - $I \subseteq S$ initial states
  - $F \subseteq S$ accepting states

- The language of an automaton A is

$$L(A) = \{a_1a_2\ldots a_n \in \Sigma^* \mid \exists\, s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \ldots \xrightarrow{a_3} s_n \in F \text{ with } s_1 \in I\}$$

Properties of finite languages

- Theorem: $L(A_1 x A_2) = L(A_1) \cap L(A_2)$

  $A_1 x A_2 = <\Sigma, S_1 x S_2, \rightarrow, I_1 x I_2, F_1 x F_2>$ where

  $<s,t> \xrightarrow{a} <s',t'>$ iff $s \xrightarrow{a}_1 s'$ and $t \xrightarrow{a}_2 t'$

- Theorem: $L(A) = \varnothing$ is decidable

  It is enough to find a path from an initial state in I to a final state in F.

Automata on infinite words: Buchi

- A Buchi automaton is a special kind of transition systems for recognizing languages on infinite words

- Buchi automaton $A = < \Sigma, S, \rightarrow, I, F >$
  - Σ finite alphabet
  - S finite set of states
  - $\rightarrow \subseteq S \times \Sigma \times S$ transition relation
  - $I \subseteq S$ initial states
  - $F \subseteq S$ accepting states

Buchi automata

An infinite execution of a Buchi automaton A

$$s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_3 \xrightarrow{a_3} s_4 \ldots$$

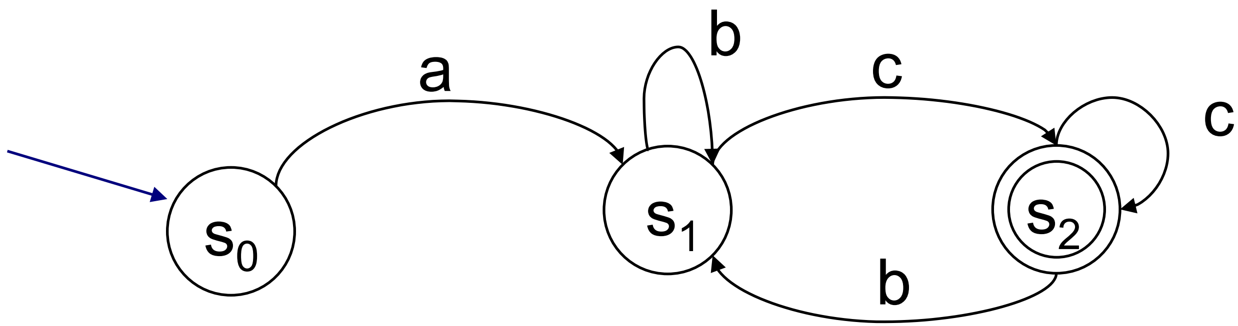
is accepted by A if

- $s_1 \in I$
- there exists infinitely many $i > 0$ such that $s_i \in F$

- The language of a Buchi automaton A is

$L_\omega(A) = \{a_1 a_2 \ldots \in \Sigma^\omega \mid \exists\ s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \ldots$ accepted by A$\}$

Example



- abccccccc... accepted
- abcbcbcbcb... accepted
- abcbbbbbb… rejected

Properties of infinite languages

- **Theorem**: $L_\omega(A_1 \otimes A_2) = L_\omega(A_1) \cap L_\omega(A_2)$

  $A_1 \otimes A_2 = \langle \Sigma, S_1 x S_2 x \{1,2\}, \rightarrow, I_1 x I_2 x \{1\}, F_1 x S_2 x \{1\} \rangle$

  where $\langle s,t,i \rangle \xrightarrow{a} \langle s',t',j \rangle$ iff

  - $s \xrightarrow{a}_1 s'$ and $t \xrightarrow{a}_2 t'$ and $i=j$ unless
  - $i=1$ and $s \in F_1$ in which case $j = 2$, or
  - $i=2$ and $t \in F_2$ in which case $j = 1$.

- **Theorem**: $L_\omega(A) = \varnothing$ is decidable

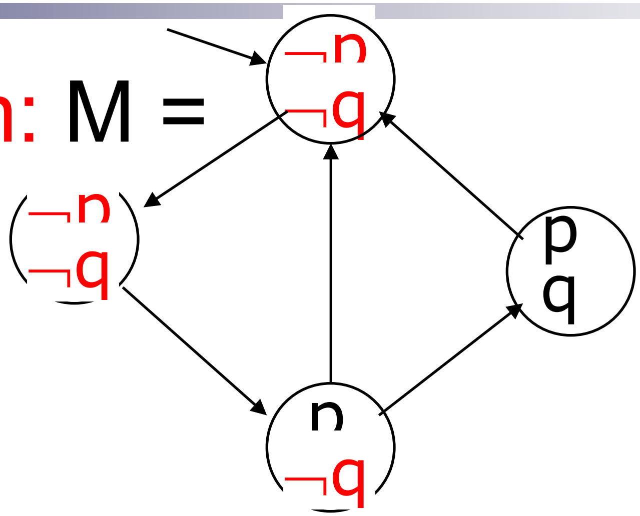
  It is enough to find a path from an initial state $s \in I$ to a final state $t \in F$ such that t has a path to t itself.

Transition systems and Buchi automata

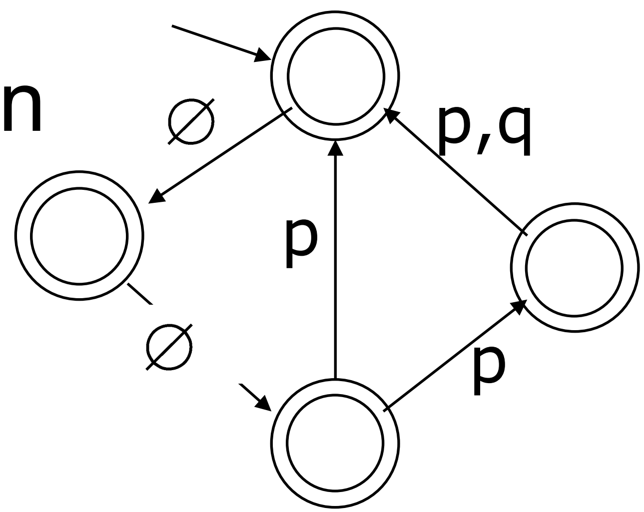
- Any transition systems $M = <S, \rightarrow_M, s_0>$ with a labelling function $\ell : S \rightarrow 2^{Prop}$ can be seen as a Buchi automata $A_M = < \Sigma, S, \rightarrow, I, F>$ where
  - $\Sigma = 2^{Prop}$ assignment of truth values to propositions (i.e. valuations)
  - S same states
  - $s \xrightarrow{a} t$ iff $s \rightarrow_M t$ and $a = \ell(s)$ transition relation
  - $I = \{s_0\}$ same initial state
  - $F = S$ every state is final

Example

- The system: M =



becomes the Buchi automaton



LTL and Buchi automata

- An LTL formula denotes a set of infinite traces which satisfy that formula

- A Buchi automaton accepts a set of infinite traces

- Theorem: Given an LTL formula ϕ, we can build a Buchi automaton

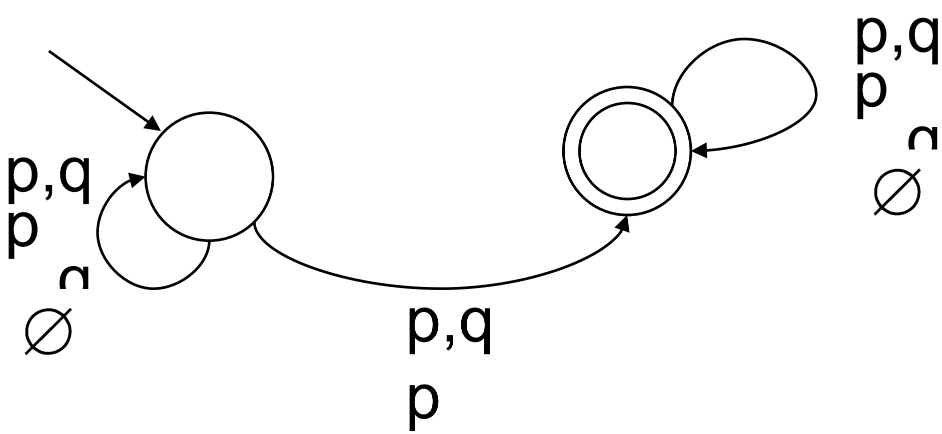
  $$A_\phi = < \Sigma,S,\rightarrow,I,F>$$

  where $\Sigma = 2^{Prop}$ consists of the subsets of (possibly negated) atomic propositions (i.e. valuations), which accepts only and all the executions satisfying the formula ϕ.

Example (1)

- ϕ = Fp eventually p

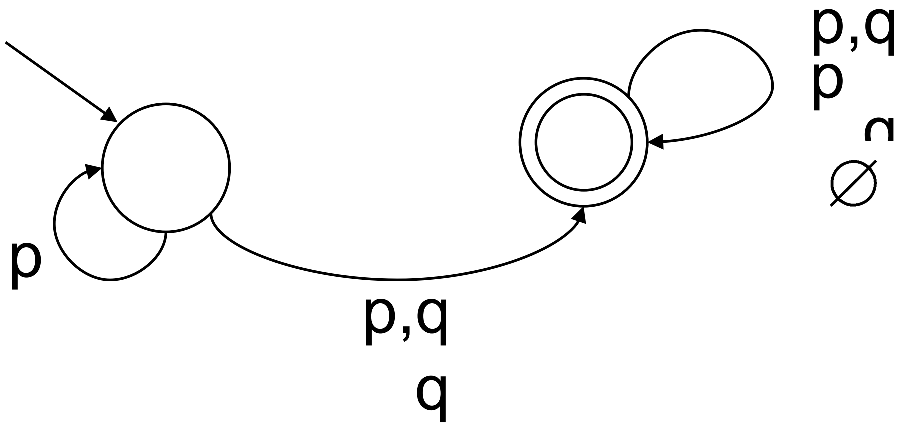
A_ϕ =



Example (2)

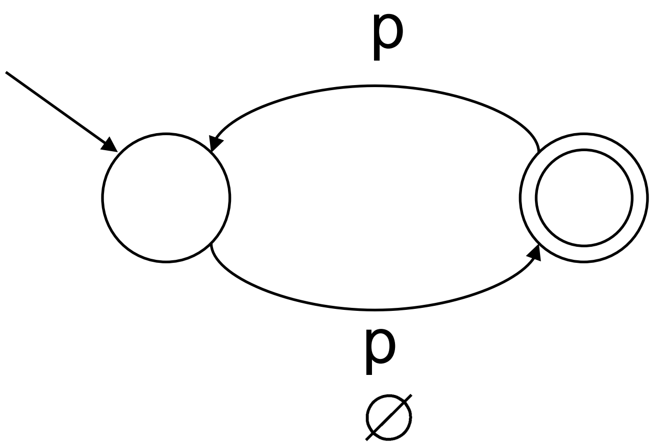
- ϕ = p U q   p until q

A_ϕ =



LTL and Buchi automata

- Not every Buchi automaton is an LTL formula:



"p holds on every odd step"

Model checking LTL:the idea

- Let ϕ be an LTL formula and M,s be a transition system specifying the behavior of a system
  - A_ϕ corresponds to all allowable behavior of the system
  - A_M corresponds to all possible behavior of the system (all infinite paths of M that are potentially interesting)

  To see whether a system satisfies a specification we need to check if every path of A_M is in A_ϕ

$$L_\omega(A_M) \subseteq L_\omega(A_\phi)$$

Model checking LTL

- To check set inclusion note that

$$B \subseteq A \Leftrightarrow B \cap \bar{A} = \emptyset$$

- Further, $\overline{L_\omega(A_\phi)} = L_\omega(A_{\neg\phi})$ thus

Every possible path is allowable
is equivalent to say that
there is no path that is possible and not allowable

that is $M,s \vDash \phi$ if and only if $L_\omega(A_M) \cap L_\omega(A_{\neg\phi}) = \emptyset$

The method

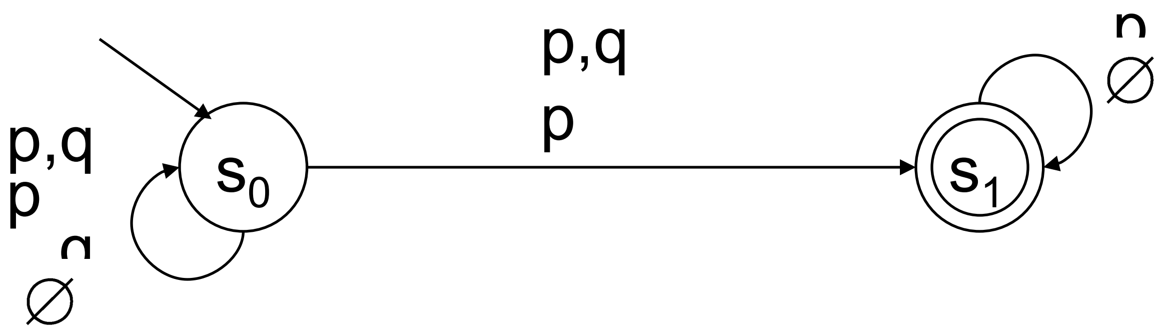
- Problem: $M,s \models \phi$?
  1. Construct a Buchi automaton $A_{\neg\phi}$ representing the negation of the desired LTL specification ϕ
  2. Construct the automaton A_M representing the system behavior
  3. Construct the automaton $A_M \otimes A_{\neg\phi}$
  4. Check if $L_\omega(A_M \otimes A_{\neg\phi}) = \varnothing$
  5. If yes then $M,s \models \phi$

Example (1)

- Specification: $\phi = G(p \Rightarrow XFq)$
  Any occurence of p must be followed (later) by an occurrence of q

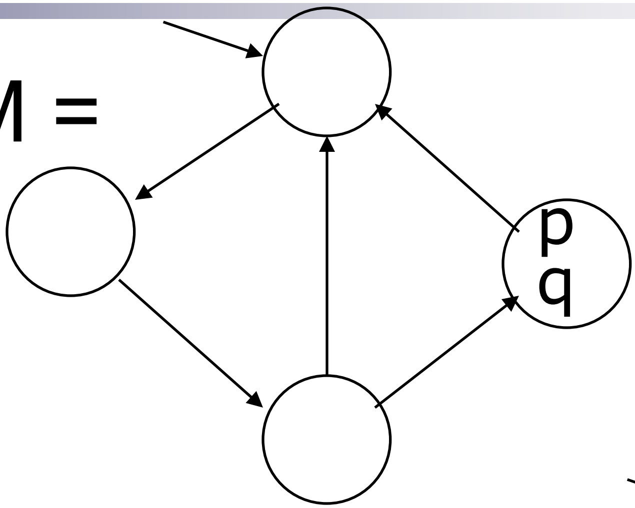
- $\neg\phi = F(p \wedge XG\neg q)$
  there exist an occurrence of p after which q will never be encountered again

- $A_{\neg\phi}$ =

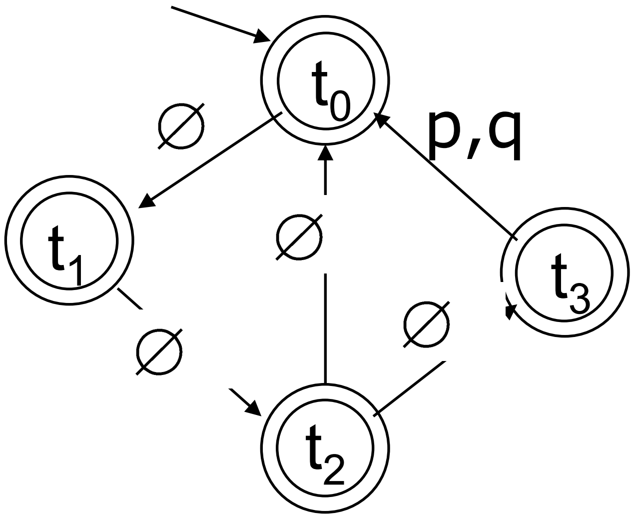


Example (2)

- The system: M =

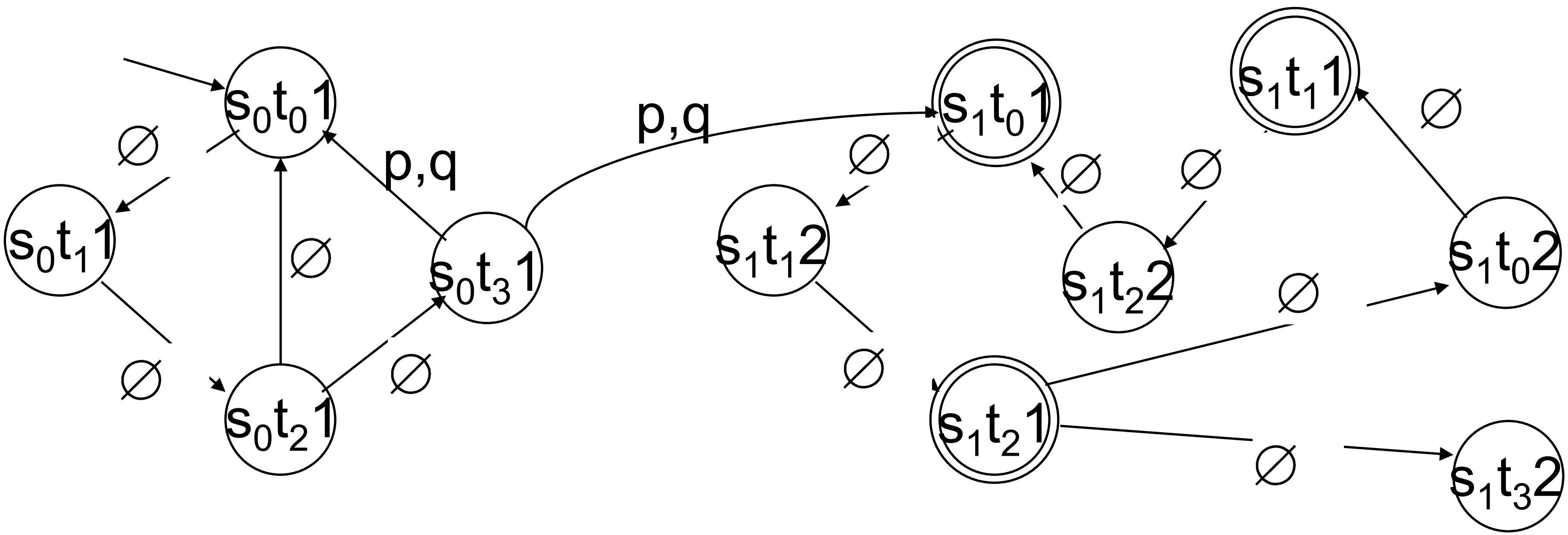


and its Buchi automaton A_M



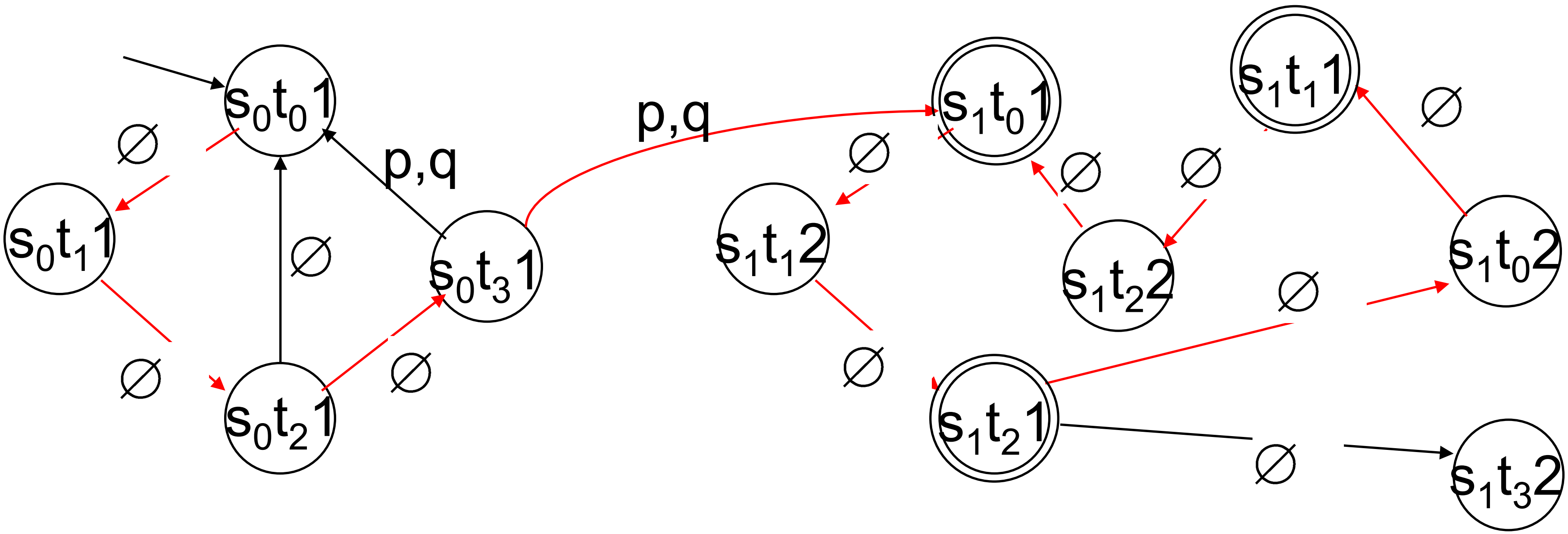
Example: (3)

- The product $A_{\neg\phi} \otimes A_M$



Example: (4)

- $L(A_{\neg\phi} \otimes A_M) = \emptyset$?



There is a path starting from <s_0t_01> that passes infinitely often through the final states

Example: (5)

- Since $L(A_{\neg\phi} \otimes A_M)$ is not empty

$$M,s \not\models G(p \Rightarrow XFq)$$

The counterexample is given by the path

$$t_0t_1t_2t_3t_0t_1t_2t_0t_1t_2t_0\ldots$$

From LTL to Buchi automata

- General approach:
  - Rewrite formula in normal form
  - Translate formula into generalized Buchi automata
  - Turn generalized Buchi automata into ordinary Buchi automata

Normal form

- LTL formulas with the until operator U that may contains also the next operators X

- Every formula ϕ can be converted into an equivalent formula ψ in normal form expressing an infinite behavior using equivalences such as:
  - $T = T \; U \; T$
  - $p = p \wedge XT$
  - $F\phi = T \; U \; \phi$ $\qquad$ $G \; \phi = \perp R \; \phi$
  - $\phi_1 R \phi_2 = \neg(\neg\phi_1 U \neg\phi_2)$

Additional simplifications

- Use extra equivalences to reduce size of the formula. For example:
  - $\neg\neg\phi = \phi$
  - $X\phi_1 \vee X\phi_2 = X(\phi_1 \vee \phi_2)$
  - $X\phi_1 \wedge X\phi_2 = X(\phi_1 \wedge \phi_2)$
  - $X\phi_1 U X\phi_2 = X(\phi_1 U \phi_2)$

Example:

- $G(Fp \Rightarrow q) = G(\neg Fp \vee q)$
  $= \bot \; R \; (\neg Fp \vee q)$
  $= \neg \; (\neg \; \bot \; U \; \neg(\neg \; (T \; U \; p) \vee q))$

- $p \wedge \neg q = (p \wedge \neg q) \wedge T$
  $= (p \wedge \neg q) \wedge XT$
  $= (p \wedge \neg q) \wedge XGT$
  $= (p \wedge \neg q) \wedge X(T \; U \; T)$

Generalized Buchi Automata

- They differ from (normal) Buchi automata only in the acceptance condition, which is a 'set of acceptance sets', i.e. $\mathcal{F} \subseteq 2^S$

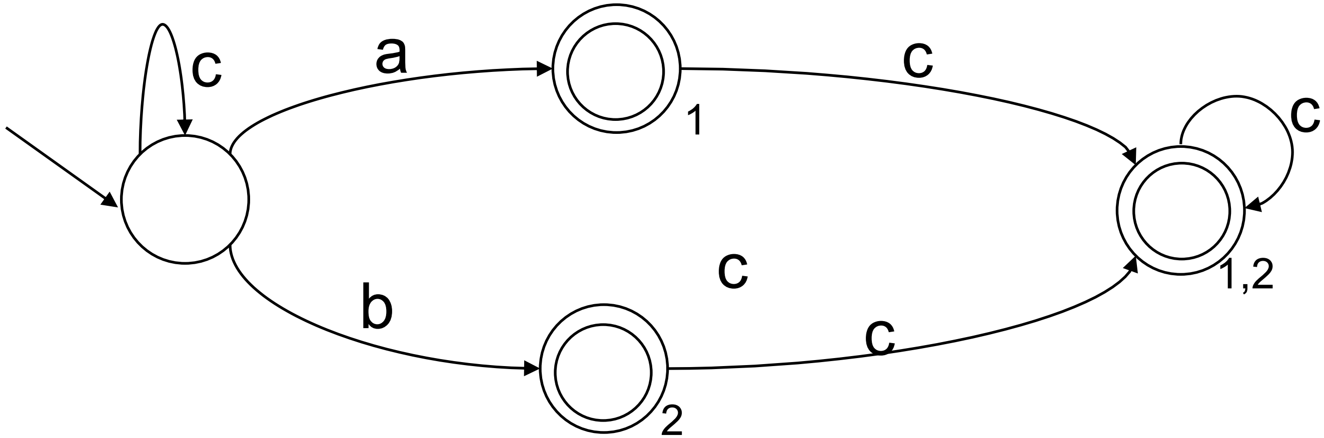
- The language of a generalized Buchi automaton $A = < \Sigma, S, \rightarrow, I, \mathcal{F} >$ is

$$L(A) = \cap \{ L(A_F) \mid F \in \mathcal{F} \text{ and } A_F = < \Sigma, S, \rightarrow, I, F > \}$$

that is, a path has to visit for each set of final states $F \in \mathcal{F}$ infinitely many times states from *F*.

Example

- A generalized Buchi automaton:



- Every path of c's with either eventually one a or eventually one b is accepted

Generalized Buchi Automata

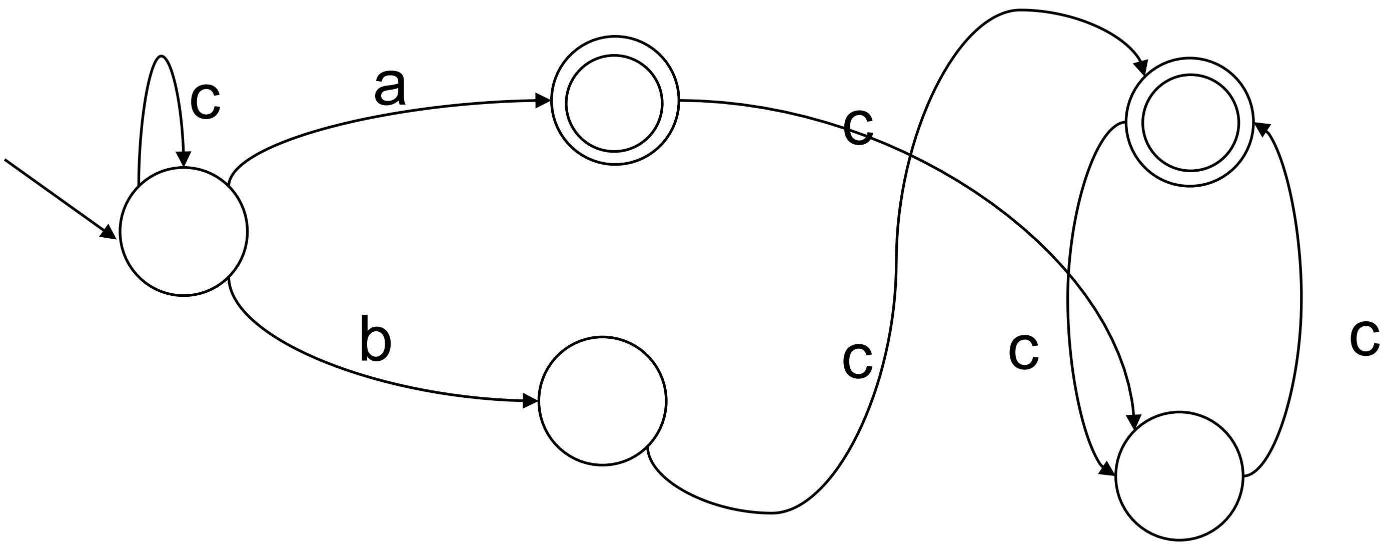
- A generalised Buchi automaton A = < Σ,S,$\rightarrow$, I, $\mathcal{F}$ > can be translated back into an ordinary Buchi automata by taking the intersection of the automata A_F = < Σ,S,$\rightarrow$, I,F> for each F $\in \mathcal{F}$.

- If $\mathcal{F} = \emptyset$ then every infinite path is accepted.
- The ordinary Buchi automata of < Σ,S,$\rightarrow$, I, $\emptyset$> is

< Σ,S,$\rightarrow$, I, S >

Example (cont'd)

- The translation of the previous automaton into an ordinary Buchi automaton is



Closure of a formula

- Given an LTL formula ϕ define its closure $Cl(\phi)$ to be the set of subformulas ψ of ϕ and of their complement.
  - $\phi \in Cl(\phi)$
  - $\psi \in Cl(\phi)$ implies $\neg\psi \in Cl(\phi)$
  - $\psi_1 \vee \psi_2 \in Cl(\phi)$ implies $\psi_1, \psi_2 \in Cl(\phi)$
  - $X\psi \in Cl(\phi)$ implies $\psi \in Cl(\phi)$
  - $\psi_1 U \psi_2 \in Cl(\phi)$ implies $\psi_1, \psi_2 \in Cl(\phi)$

Constructing the automata A_ϕ:states

- The states $Sub(\phi)$ of the automata are the maximal subsets S of $Cl(\phi)$ that have no propositional inconsitency

1. For all $\psi \in Cl(\phi)$, $\psi \in S$ iff $\neg\psi \notin S$
2. If $T \in Cl(\phi)$ then $T \in S$
3. $\psi_1 \vee \psi_2 \in S$ iff $\psi_1 \in S$ or $\psi_2 \in S$, whenever $\psi_1 \vee \psi_2 \in Cl(\phi)$
4. $\neg(\psi_1 \vee \psi_2) \in S$ iff $\neg\psi_1 \in S$ and $\neg\psi_2 \in S$, whenever $\neg(\psi_1 \vee \psi_2) \in Cl(\phi)$
5. If $\psi_1 U \psi_2 \in S$ then $\psi_1 \in S$ or $\psi_2 \in S$
6. If $\neg(\psi_1 U \psi_2) \in S$ then $\neg\psi_2 \in S$

Intuition: $\psi \in S$ implies that ψ holds in S

- The initial states are those states containing ϕ

Example

- $Cl(pUq) = \{p, q, \neg p, \neg q, pUq, \neg(pUq)\}$

- $Sub(pUq) = \{\{p, q, pUq\},$
  $\{p, \neg q, pUq\},$
  $\{p, \neg q, \neg(pUq)\}$
  $\{\neg p, q, pUq\}$
  $\{\neg p, \neg q, \neg(pUq)\}\}$

Constructing the automata: transitions

Define the transition relation by setting $s \xrightarrow{a} s'$ iff

1. $X\psi \in s$ implies $\psi \in s'$
2. $\neg X\psi \in s$ implies $\neg\psi \in s'$
3. $\psi_1 U \psi_2 \in s$ and $\psi_2 \notin s$ implies $\psi_1 U \psi_2 \in s'$
4. $\neg(\psi_1 U \psi_2) \in s$ and $\psi_1 \in s$ implies $\neg(\psi_1 U \psi_2) \in s'$
5. a = set of all atomic propositions that hold in s

N.B.: Conditions 3. and 4. are there because

$$\psi_1 U \psi_2 \equiv \psi_2 \vee (\psi_1 \wedge X(\psi_1 U \psi_2))$$

$$\psi_1 R \psi_2 \equiv \psi_2 \wedge (\psi_1 \vee X(\psi_1 R \psi_2))$$

Constructing the automata: acceptance

- For each $\chi_i U \psi_i \in Cl(\phi)$ define the set of accepting states F_i by
  - $s \in F_i$ iff $\neg(\chi_i U \psi_i) \in s$ or $\psi_i \in s$
  - The above means that we only accept executions for which infinitely many time $\neg(\chi_i U \psi_i) \vee \psi_i$ holds

- Intuition:

  For each $\chi_i U \psi_i \in Cl(\phi)$ we have to guarantee that eventually ψ_i holds.

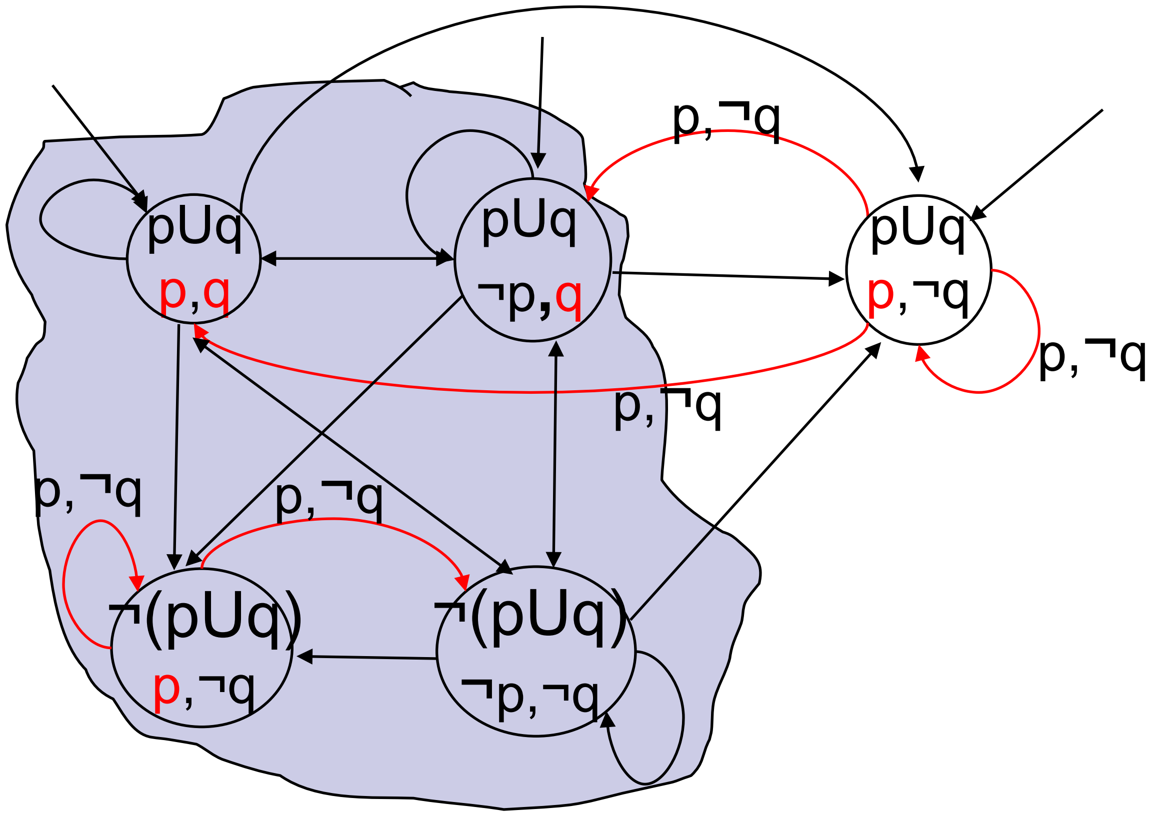
  1. Suppose we accept an execution for which only finitely many time $\neg(\chi_i U \psi_i) \vee \psi_i$ holds.
  2. Then we can find a suffix such that $\neg(\chi_i U \psi_i) \vee \psi_i$ will never hold, that is $(\chi_i U \psi_i) \wedge \neg\psi_i$ will always hold.
  3. Thus we have an execution for which our goal is not guaranteed

Complexity

- $A_{\neg\phi}$ has size $O(2^{|\phi|})$ in the worst case
- The product $A \otimes B$ has size $O(|A| x |B|)$
- We can determine if there no acceptable path in $A \otimes B$ in $O(|A \otimes B|)$ time
- Thus, model checking $M,s \vDash \phi$ can be done in $O(|M| x\ 2^{|\phi|})$ time

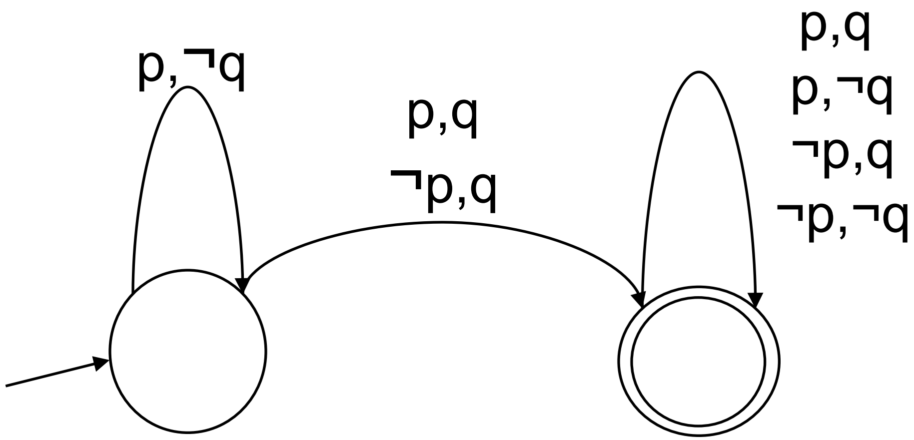
Example: pUq

- Cl(pUq) = { p, ¬p, q, ¬q, pUq, ¬(pUq) }



Example: pUq

- The previous automata is equivalent to



Example II

- Buchi automaton for atomic proposition p
  - p = $p \wedge X(T\ U\ T) = \phi$
  - $Cl(\phi) = \{ p, \neg p, T, \neg T, TUT, \neg(T\ U\ T), X(TUT), \neg X(TUT), \phi, \neg\phi\}$
  - $Sub(\phi) = \{1,2,3\}$ with
    - $1 = \{p, T, TUT, X(TUT), \phi \}$,
    - $2 = \{\neg p, T, TUT, X(TUT), \neg\phi\}$
    - $3 = \{p, T, TUT, \neg X(TUT), \neg\phi\}\}$

Example II

- Buchi automaton for atomic proposition p