An Automata-Theoretic Approach to Automatic Program Verification

Moshe Y. Vardi Pierre Wolper
Presentation: Carl Leonardsson

Reading Group Seminar 24/2 - 2012

Purpose

- Goal: Verifying programs against temporal formulae
- Same as last time
- This time in LTL

Purpose

- Goal: Verifying programs against temporal formulae
- Same as last time
- This time in LTL

LTL

LTL formulae

- p, $\neg\phi$, $\phi \wedge \psi$ - Propositional logic as usual
- $X\phi$ - **neXt**: ϕ holds in the next state
- $\phi U \psi$ - **Until**: ψ will happen sooner or later, ϕ holds until then

- LTL quantifies universally over paths.
- Comparison: CTL picks quantification at each computation tree branch

- How do we model check against LTL formulae?

LTL

LTL formulae

- p, $\neg\phi$, $\phi \wedge \psi$ - Propositional logic as usual
- $X\phi$ - **neXt**: ϕ holds in the next state
- $\phi U \psi$ - **Until**: ψ will happen sooner or later, ϕ holds until then

- LTL quantifies universally over paths.
- Comparison: CTL picks quantification at each computation tree branch
- How do we model check against LTL formulae?

LTL

LTL formulae

- p, $\neg\phi$, $\phi \wedge \psi$ - Propositional logic as usual
- $X\phi$ - **neXt**: ϕ holds in the next state
- $\phi U \psi$ - **Until**: ψ will happen sooner or later, ϕ holds until then

- LTL quantifies universally over paths.
- Comparison: CTL picks quantification at each computation tree branch

- How do we model check against LTL formulae?

Automata accepting languages of infinite words.

Definition

A Büchi automaton is $(\Sigma, S, \rho, S_0, F)$

- Σ an alphabet
- S a set of states
- $\rho : S \times \Sigma \to 2^S$ the transition function
- S_0 the set of initial states
- F the set of accepting states

- An infinite word w is accepted by a Büchi automaton A if *there is* a run of A, following w, which passes through accepting states an *infinite* number of times.

Preliminary: Büchi Automata

Automata accepting languages of infinite words.

Definition

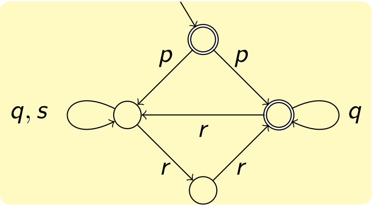
A Büchi automaton is $(\Sigma, S, \rho, S_0, F)$

- Σ an alphabet
- S a set of states
- $\rho : S \times \Sigma \to 2^S$ the transition function
- S_0 the set of initial states
- F the set of accepting states

- An infinite word w is accepted by a Büchi automaton A if *there is* a run of A, following w, which passes through accepting states an *infinite* number of times.

Büchi Automata: Example

- An infinite word w is accepted by a Büchi automaton A if *there is* a run of A, following w, which passes through accepting states an *infinite* number of times.



p

$pqqqqqqq\cdots$

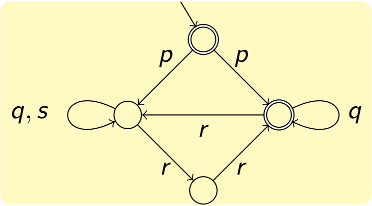
$psssssss\cdots$

$pppppppp\cdots$

$prrrrrrr\cdots$

Büchi Automata: Example

- An infinite word w is accepted by a Büchi automaton A if *there is* a run of A, following w, which passes through accepting states an *infinite* number of times.



p Not infinite!

$pqqqqqq\cdots$

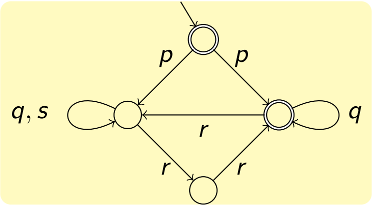
$psssssss\cdots$

$pppppppp\cdots$

$prrrrrr\cdots$

Büchi Automata: Example

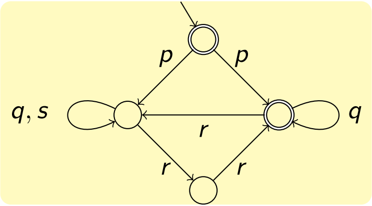
- An infinite word w is accepted by a Büchi automaton A if *there is* a run of A, following w, which passes through accepting states an *infinite* number of times.



p	Not infinite!
$pqqqqqqq\cdots$	Accepted
$psssssss\cdots$	
$pppppppp\cdots$	
$prrrrrrr\cdots$	

Büchi Automata: Example

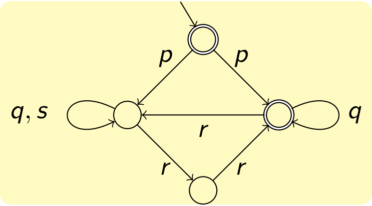
- An infinite word w is accepted by a Büchi automaton A if *there is* a run of A, following w, which passes through accepting states an *infinite* number of times.



p	Not infinite!
$pqqqqqq\cdots$	Accepted
$psssssss\cdots$	Not accepted
$pppppppp\cdots$	
$prrrrrr\cdots$	

Büchi Automata: Example

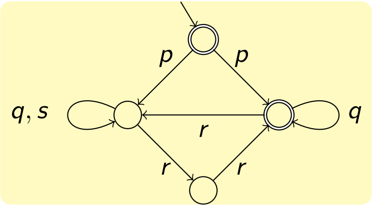
- An infinite word w is accepted by a Büchi automaton A if *there is* a run of A, following w, which passes through accepting states an *infinite* number of times.



p	Not infinite!
$pqqqqqq\cdots$	Accepted
$psssssss\cdots$	Not accepted
$pppppppp\cdots$	Not accepted
$prrrrrr\cdots$	

Büchi Automata: Example

- An infinite word w is accepted by a Büchi automaton A if *there is* a run of A, following w, which passes through accepting states an *infinite* number of times.



p	Not infinite!
$pqqqqqq\cdots$	Accepted
$pssssss\cdots$	Not accepted
$pppppppp\cdots$	Not accepted
$prrrrrr\cdots$	Accepted

Outline

Goal: Verification of a program P against an LTL formula ϕ
I.e. Check that all computations of P satisfy ϕ.

1. Represent P as a Büchi automaton A^P
2. Represent $\neg\phi$ as a Büchi automaton $A^{\neg\phi}$
3. Compute the Büchi automaton $A = A^P \bigcap A^{\neg\phi}$
4. Check that the language of A is empty.

Complexity

- Linear in $|P|$ (size of model)
- Exponential in $|\phi|$ (number of symbols)

Outline

Goal: Verification of a program P against an LTL formula ϕ
I.e. Check that all computations of P satisfy ϕ.

1. Represent P as a Büchi automaton A^P
2. Represent $\neg\phi$ as a Büchi automaton $A^{\neg\phi}$
3. Compute the Büchi automaton $A = A^P \bigcap A^{\neg\phi}$
4. Check that the language of A is empty.

Complexity

- Linear in $|P|$ (size of model)
- Exponential in $|\phi|$ (number of symbols)

Outline

Goal: Verification of a program P against an LTL formula ϕ
I.e. Check that all computations of P satisfy ϕ.

1. Represent P as a Büchi automaton A^P
2. Represent $\neg\phi$ as a Büchi automaton $A^{\neg\phi}$
3. Compute the Büchi automaton $A = A^P \bigcap A^{\neg\phi}$
4. Check that the language of A is empty.

Complexity

- Linear in $|P|$ (size of model)
- Exponential in $|\phi|$ (number of symbols)

Outline

Goal: Verification of a program P against an LTL formula ϕ
I.e. Check that all computations of P satisfy ϕ.

1. Represent P as a Büchi automaton A^P
2. Represent $\neg\phi$ as a Büchi automaton $A^{\neg\phi}$
3. Compute the Büchi automaton $A = A^P \bigcap A^{\neg\phi}$
4. Check that the language of A is empty.

Complexity

- Linear in $|P|$ (size of model)
- Exponential in $|\phi|$ (number of symbols)

Outline

Goal: Verification of a program P against an LTL formula ϕ
I.e. Check that all computations of P satisfy ϕ.

1. Represent P as a Büchi automaton A^P
2. Represent $\neg\phi$ as a Büchi automaton $A^{\neg\phi}$
3. Compute the Büchi automaton $A = A^P \bigcap A^{\neg\phi}$
4. Check that the language of A is empty.

Complexity

- Linear in $|P|$ (size of model)
- Exponential in $|\phi|$ (number of symbols)

Outline

Goal: Verification of a program P against an LTL formula ϕ
I.e. Check that all computations of P satisfy ϕ.

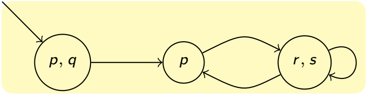
1. Represent P as a Büchi automaton A^P
2. Represent $\neg\phi$ as a Büchi automaton $A^{\neg\phi}$
3. Compute the Büchi automaton $A = A^P \bigcap A^{\neg\phi}$
4. Check that the language of A is empty.

Complexity

- Linear in $|P|$ (size of model)
- Exponential in $|\phi|$ (number of symbols)

Program → Büchi Automaton

A program (model) is an automaton with states labeled by propositions.



A computation of P is an infinite sequence of propositional valuations.

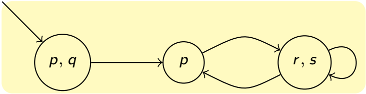
$$\{p, q\}\{p\}\{r, s\}\{r, s\}\{r, s\}\cdots$$

Compute Büchi Automaton which accepts precisely the computations of P.

Just move propositions out of the states and make every state accepting.

Program $\rightarrow$ Büchi Automaton

A program (model) is an automaton with states labeled by propositions.



A computation of P is an infinite sequence of propositional valuations.

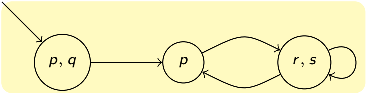
$$\{p,q\}\{p\}\{r,s\}\{r,s\}\{r,s\}\cdots$$

Compute Büchi Automaton which accepts precisely the computations of P.

Just move propositions out of the states and make every state accepting.

Program $\rightarrow$ Büchi Automaton

A program (model) is an automaton with states labeled by propositions.



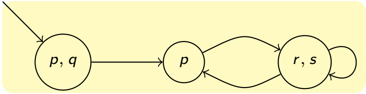
A computation of P is an infinite sequence of propositional valuations.

$$\{p, q\}\{p\}\{r, s\}\{r, s\}\{r, s\}\cdots$$

Compute Büchi Automaton which accepts precisely the computations of P.

Just move propositions out of the states and make every state accepting.

A program (model) is an automaton with states labeled by propositions.

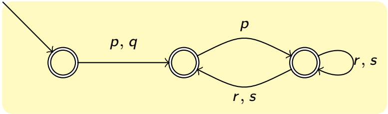


A computation of P is an infinite sequence of propositional valuations.

$$\{p,q\}\{p\}\{r,s\}\{r,s\}\{r,s\}\cdots$$

Compute Büchi Automaton which accepts precisely the computations of P.

Just move propositions out of the states and make every state accepting.



LTL $\rightarrow$ Büchi Automaton

For a given LTL formula ϕ, construct Büchi automaton A^ϕ such that $L(A^\phi) = \{w | w \models \phi\}$.[1]

1. Construct automaton L^ϕ that checks *local* conformance with formulae.
2. Construct automaton E^ϕ that checks that for $\phi_0 U \phi_1$, *eventually* ϕ_1 holds.
3. Compute $A^\phi = L^\phi \times E^\phi$

[1] For more detail see Appendix B in VW85.

LTL $\rightarrow$ Büchi Automaton

For a given LTL formula ϕ, construct Büchi automaton A^{ϕ} such that $L(A^{\phi}) = \{w | w \models \phi\}$.[1]

1. Construct automaton L^{ϕ} that checks *local* conformance with formulae.
2. Construct automaton E^{ϕ} that checks that for $\phi_0 U \phi_1$, *eventually* ϕ_1 holds.
3. Compute $A^{\phi} = L^{\phi} \times E^{\phi}$

[1] For more detail see Appendix B in VW85.

LTL $\rightarrow$ Büchi Automaton

For a given LTL formula ϕ, construct Büchi automaton A^ϕ such that $L(A^\phi) = \{w | w \models \phi\}$.[1]

1. Construct automaton L^ϕ that checks *local* conformance with formulae.
2. Construct automaton E^ϕ that checks that for $\phi_0 U \phi_1$, *eventually* ϕ_1 holds.
3. Compute $A^\phi = L^\phi \times E^\phi$

[1] For more detail see Appendix B in VW85.

LTL → Büchi Automaton

For a given LTL formula ϕ, construct Büchi automaton A^ϕ such that $L(A^\phi) = \{w | w \models \phi\}$.[1]

1. Construct automaton L^ϕ that checks *local* conformance with formulae.
2. Construct automaton E^ϕ that checks that for $\phi_0 U \phi_1$, *eventually* ϕ_1 holds.
3. Compute $A^\phi = L^\phi \times E^\phi$

[1] For more detail see Appendix B in VW85.

LTL $\rightarrow$ Büchi Automaton: L^{ϕ}

Idea: Construct an automaton which keeps track of how we can transition between valuations of subformulae of ϕ.

Let $cl(\phi)$ denote the set of subformulae of ϕ.

Construct an automaton where the states are *consistent* valuations of $cl(\phi)$. A *consistent* valuation s of cl satisfies

- s is propositionally consistent
- $\phi U \psi \in s \rightarrow \phi \in s \vee \psi \in s$
- $\psi \in s \rightarrow \phi U \psi \in s$

Transitions are of the form $s \xrightarrow{s} t$ where

- $X\phi \in s$ iff $\phi \in t$
- $\phi U \psi \in s$ iff
  - $\psi \in s$ or
  - $\phi \in s \wedge \phi U \psi \in t$

LTL $\rightarrow$ Büchi Automaton: L^{ϕ}

Idea: Construct an automaton which keeps track of how we can transition between valuations of subformulae of ϕ.

Let $cl(\phi)$ denote the set of subformulae of ϕ.

Construct an automaton where the states are *consistent* valuations of $cl(\phi)$. A *consistent* valuation s of cl satisfies

- s is propositionally consistent
- $\phi U \psi \in s \rightarrow \phi \in s \vee \psi \in s$
- $\psi \in s \rightarrow \phi U \psi \in s$

Transitions are of the form $s \xrightarrow{s} t$ where

- $X\phi \in s$ iff $\phi \in t$
- $\phi U \psi \in s$ iff
  - $\psi \in s$ or
  - $\phi \in s \wedge \phi U \psi \in t$

LTL $\rightarrow$ Büchi Automaton: L^{ϕ}

Idea: Construct an automaton which keeps track of how we can transition between valuations of subformulae of ϕ.

Let $cl(\phi)$ denote the set of subformulae of ϕ.

Construct an automaton where the states are *consistent* valuations of $cl(\phi)$. A *consistent* valuation s of cl satisfies

- s is propositionally consistent
- $\phi U \psi \in s \rightarrow \phi \in s \vee \psi \in s$
- $\psi \in s \rightarrow \phi U \psi \in s$

Transitions are of the form $s \xrightarrow{s} t$ where

- $X\phi \in s$ iff $\phi \in t$
- $\phi U \psi \in s$ iff
  - $\psi \in s$ or
  - $\phi \in s \wedge \phi U \psi \in t$

LTL $\rightarrow$ Büchi Automaton: L^{ϕ}

Idea: Construct an automaton which keeps track of how we can transition between valuations of subformulae of ϕ.

Let $cl(\phi)$ denote the set of subformulae of ϕ.

Construct an automaton where the states are *consistent* valuations of $cl(\phi)$. A *consistent* valuation s of cl satisfies

- s is propositionally consistent
- $\phi U \psi \in s \rightarrow \phi \in s \vee \psi \in s$
- $\psi \in s \rightarrow \phi U \psi \in s$

Transitions are of the form $s \xrightarrow{s} t$ where

- $X\phi \in s$ iff $\phi \in t$
- $\phi U \psi \in s$ iff
  - $\psi \in s$ or
  - $\phi \in s \wedge \phi U \psi \in t$

Example: $\phi = pUq$

Consistent state s:

- s is propositionally consistent
- $\phi U \psi \in s \rightarrow \phi \in s \vee \psi \in s$
- $\psi \in s \rightarrow \phi U \psi \in s$

Transition $s \xrightarrow{s} t$:

- $X\phi \in s$ iff $\phi \in t$
- $\phi U \psi \in s$ iff
  - $\psi \in s$ or
  - $\phi \in s \wedge \phi U \psi \in t$

Subformulae: p, q, pUq

States:

$\{\}$ $\{p\}$

$\{p, pUq\}$

$\{q, pUq\}$

$\{p, q, pUq\}$

Nice automaton! ... but what is it good for?

Example: $\phi = pUq$

Consistent state s:

- s is propositionally consistent
- $\phi U \psi \in s \rightarrow \phi \in s \vee \psi \in s$
- $\psi \in s \rightarrow \phi U \psi \in s$

Transition $s \xrightarrow{s} t$:

- $X\phi \in s$ iff $\phi \in t$
- $\phi U \psi \in s$ iff
  - $\psi \in s$ or
  - $\phi \in s \wedge \phi U \psi \in t$

Subformulae: p, q, pUq

States:

$\{\}$ $\{p\}$

$\{pUq\}$ $\{p, pUq\}$

$\{q, pUq\}$ $\{p, q\}$

$\{q\}$ $\{p, q, pUq\}$

Example: $\phi = pUq$

Consistent state s:

- s is propositionally consistent
- $\phi U \psi \in s \to \phi \in s \vee \psi \in s$
- $\psi \in s \to \phi U \psi \in s$

Transition $s \xrightarrow{s} t$:

- $X\phi \in s$ iff $\phi \in t$
- $\phi U \psi \in s$ iff
  - $\psi \in s$ or
  - $\phi \in s \wedge \phi U \psi \in t$

Subformulae: $p,\ q,\ pUq$

Inconsistent States:

$\{\}$ $\{p\}$

$\{pUq\}$ $\{p, pUq\}$

$\{q, pUq\}$ $\{p, q\}$

$\{q\}$ $\{p, q, pUq\}$

Example: $\phi = pUq$

Consistent state s:

- s is propositionally consistent
- $\phi U \psi \in s \rightarrow \phi \in s \vee \psi \in s$
- $\psi \in s \rightarrow \phi U \psi \in s$

Transition $s \xrightarrow{s} t$:

- $X\phi \in s$ iff $\phi \in t$
- $\phi U \psi \in s$ iff
  - $\psi \in s$ or
  - $\phi \in s \wedge \phi U \psi \in t$

Subformulae: p, q, pUq

Consistent States:

$\{\}$ $\{p\}$

$\{p, pUq\}$

$\{q, pUq\}$

$\{p, q, pUq\}$

Nice automaton! ... but what is it good for?

LTL $\rightarrow$ Büchi Automaton: L^{ϕ}: Example

Example: $\phi = pUq$

Consistent state s:

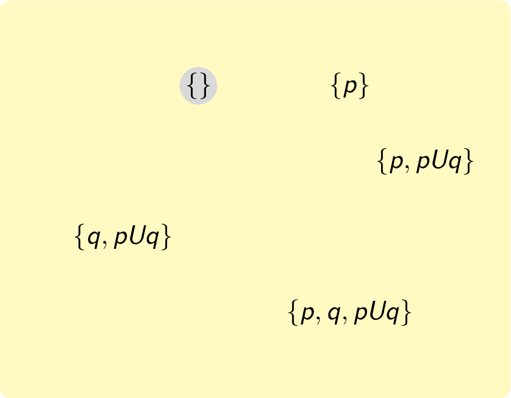
- s is propositionally consistent
- $\phi U \psi \in s \rightarrow \phi \in s \vee \psi \in s$
- $\psi \in s \rightarrow \phi U \psi \in s$

Transition $s \xrightarrow{s} t$:

- $X\phi \in s$ iff $\phi \in t$
- $\phi U \psi \in s$ iff
  - $\psi \in s$ or
  - $\phi \in s \wedge \phi U \psi \in t$

Subformulae: p, q, pUq

Transitions



Nice automaton! ... but what is it good for?

LTL $\to$ Büchi Automaton: L^{ϕ}: Example

Example: $\phi = pUq$

Consistent state s:

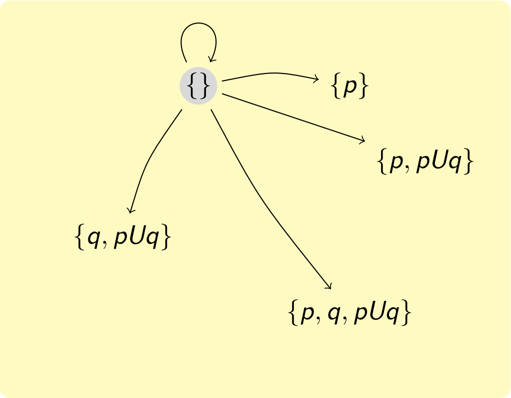
- s is propositionally consistent
- $\phi U \psi \in s \to \phi \in s \lor \psi \in s$
- $\psi \in s \to \phi U \psi \in s$

Transition $s \xrightarrow{s} t$:

- $X\phi \in s$ iff $\phi \in t$
- $\phi U \psi \in s$ iff
  - $\psi \in s$ or
  - $\phi \in s \land \phi U \psi \in t$

Subformulae: p, q, pUq

Transitions



Nice automaton! ... but what is it good for?

Example: $\phi = pUq$

Consistent state s:

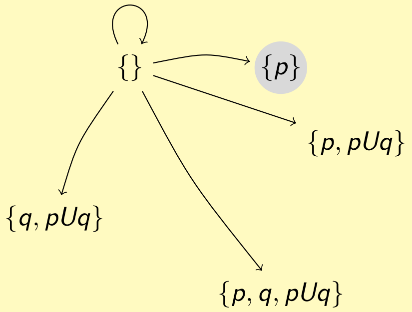
- s is propositionally consistent
- $\phi U \psi \in s \to \phi \in s \vee \psi \in s$
- $\psi \in s \to \phi U \psi \in s$

Transition $s \xrightarrow{s} t$:

- $X\phi \in s$ iff $\phi \in t$
- $\phi U \psi \in s$ iff
  - $\psi \in s$ or
  - $\phi \in s \wedge \phi U \psi \in t$

Subformulae: p, q, pUq

Transitions



Nice automaton! ... but what is it good for?

LTL $\to$ Büchi Automaton: L^{ϕ}: Example

Example: $\phi = pUq$

Consistent state s:

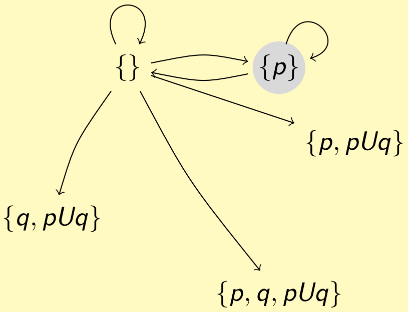
- s is propositionally consistent
- $\phi U \psi \in s \to \phi \in s \vee \psi \in s$
- $\psi \in s \to \phi U \psi \in s$

Transition $s \xrightarrow{s} t$:

- $X\phi \in s$ iff $\phi \in t$
- $\phi U \psi \in s$ iff
  - $\psi \in s$ or
  - $\phi \in s \wedge \phi U \psi \in t$

Subformulae: p, q, pUq

Transitions



Nice automaton! ... but what is it good for?

LTL $\rightarrow$ Büchi Automaton: L^{ϕ}: Example

Example: $\phi = pUq$

Consistent state s:

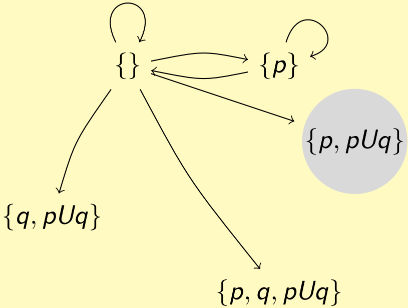
- s is propositionally consistent
- $\phi U \psi \in s \rightarrow \phi \in s \vee \psi \in s$
- $\psi \in s \rightarrow \phi U \psi \in s$

Transition $s \xrightarrow{s} t$:

- $X\phi \in s$ iff $\phi \in t$
- $\phi U \psi \in s$ iff
  - $\psi \in s$ or
  - $\phi \in s \wedge \phi U \psi \in t$

Subformulae: p, q, pUq

Transitions



Nice automaton! ... but what is it good for?

LTL $\rightarrow$ Büchi Automaton: L^{ϕ}: Example

Example: $\phi = pUq$

Consistent state s:

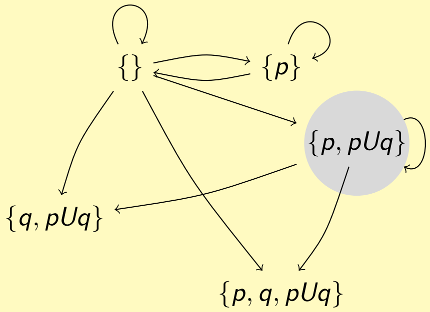
- s is propositionally consistent
- $\phi U \psi \in s \rightarrow$ $\phi \in s \vee \psi \in s$
- $\psi \in s \rightarrow \phi U \psi \in s$

Transition $s \xrightarrow{s} t$:

- $X\phi \in s$ iff $\phi \in t$
- $\phi U \psi \in s$ iff
  - $\psi \in s$ or
  - $\phi \in s \wedge \phi U \psi \in t$

Subformulae: p, q, pUq

Transitions



Nice automaton! ... but what is it good for?

LTL $\rightarrow$ Büchi Automaton: L^{ϕ}: Example

Example: $\phi = pUq$

Consistent state s:

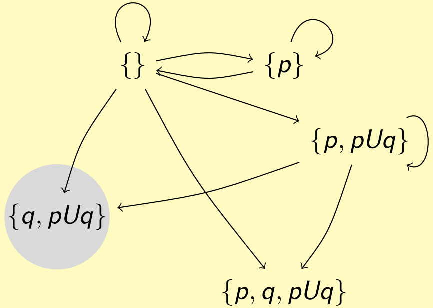
- s is propositionally consistent
- $\phi U \psi \in s \rightarrow \phi \in s \vee \psi \in s$
- $\psi \in s \rightarrow \phi U \psi \in s$

Transition $s \xrightarrow{s} t$:

- $X\phi \in s$ iff $\phi \in t$
- $\phi U \psi \in s$ iff
  - $\psi \in s$ or
  - $\phi \in s \wedge \phi U \psi \in t$

Subformulae: p, q, pUq

Transitions



Nice automaton! ... but what is it good for?

Example: $\phi = pUq$

Consistent state s:

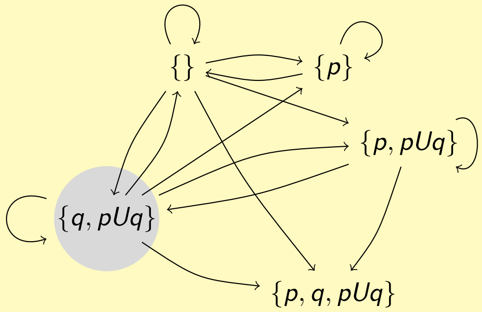
- s is propositionally consistent
- $\phi U \psi \in s \rightarrow \phi \in s \vee \psi \in s$
- $\psi \in s \rightarrow \phi U \psi \in s$

Transition $s \xrightarrow{s} t$:

- $X\phi \in s$ iff $\phi \in t$
- $\phi U \psi \in s$ iff
  - $\psi \in s$ or
  - $\phi \in s \wedge \phi U \psi \in t$

Subformulae: p, q, pUq

Transitions



Nice automaton! ... but what is it good for?

LTL $\to$ Büchi Automaton: L^{ϕ}: Example

Example: $\phi = pUq$

Consistent state s:

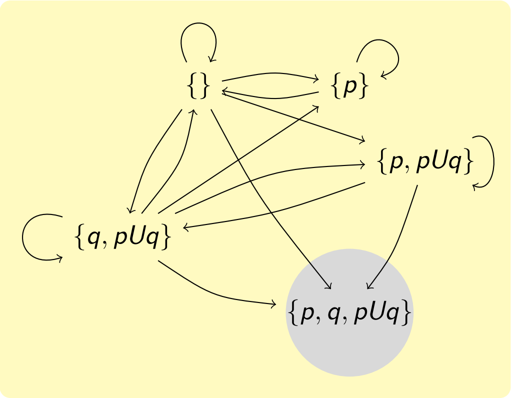
- s is propositionally consistent
- $\phi U \psi \in s \to \phi \in s \vee \psi \in s$
- $\psi \in s \to \phi U \psi \in s$

Transition $s \xrightarrow{s} t$:

- $X\phi \in s$ iff $\phi \in t$
- $\phi U \psi \in s$ iff
  - $\psi \in s$ or
  - $\phi \in s \wedge \phi U \psi \in t$

Subformulae: p, q, pUq

Transitions



Nice automaton! ... but what is it good for?

LTL $\rightarrow$ Büchi Automaton: L^{ϕ}: Example

Example: $\phi = pUq$

Consistent state s:

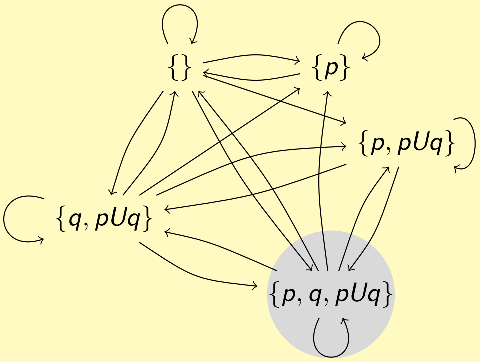
- s is propositionally consistent
- $\phi U \psi \in s \rightarrow \phi \in s \vee \psi \in s$
- $\psi \in s \rightarrow \phi U \psi \in s$

Transition $s \xrightarrow{s} t$:

- $X\phi \in s$ iff $\phi \in t$
- $\phi U \psi \in s$ iff
  - $\psi \in s$ or
  - $\phi \in s \wedge \phi U \psi \in t$

Subformulae: p, q, pUq

Transitions



Nice automaton! ... but what is it good for?

Example: $\phi = pUq$

Consistent state s:

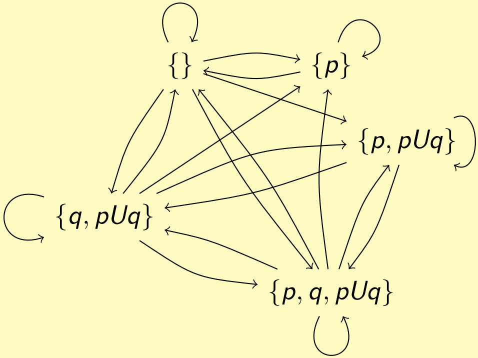
- s is propositionally consistent
- $\phi U \psi \in s \rightarrow \phi \in s \vee \psi \in s$
- $\psi \in s \rightarrow \phi U \psi \in s$

Transition $s \xrightarrow{s} t$:

- $X\phi \in s$ iff $\phi \in t$
- $\phi U \psi \in s$ iff
  - $\psi \in s$ or
  - $\phi \in s \wedge \phi U \psi \in t$

Subformulae: p, q, pUq

Transitions



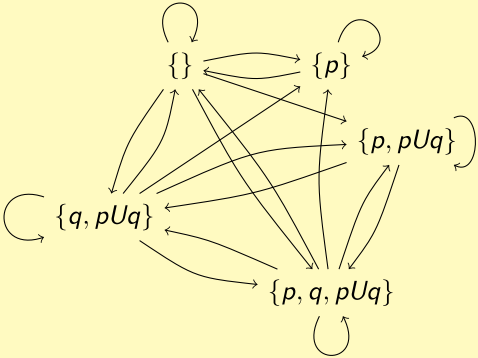
Nice automaton! ... but what is it good for?

LTL $\rightarrow$ Büchi Automaton: L^{ϕ}: Example

Example: $\phi = pUq$

Subformulae: p, q, pUq

Transitions



Consistent state s:

- s is propositionally consistent
- $\phi U \psi \in s \rightarrow \phi \in s \vee \psi \in s$
- $\psi \in s \rightarrow \phi U \psi \in s$

Transition $s \xrightarrow{s} t$:

- $X\phi \in s$ iff $\phi \in t$
- $\phi U \psi \in s$ iff
  - $\psi \in s$ or
  - $\phi \in s \wedge \phi U \psi \in t$

Nice automaton! ... but what is it good for?

LTL $\rightarrow$ Büchi Automaton: E^{ϕ}

L^{ϕ} is not sufficient!

Consider the following scenario:

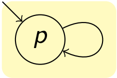
L^{pUq}:

Model:

The model never performs an illegal transition, but still it does not satisfy the assumption that pUq.

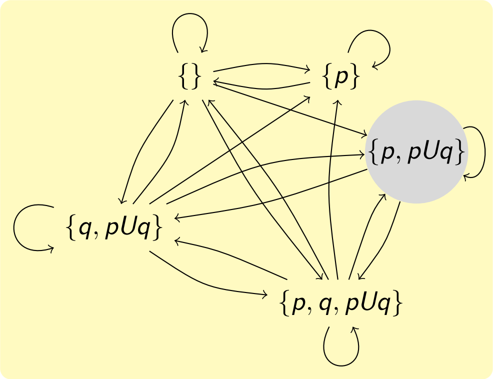
LTL $\rightarrow$ Büchi Automaton: E^{ϕ}

L^{ϕ} is not sufficient!
Consider the following scenario:

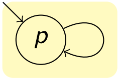
Model: 

The model never performs an illegal transition, but still it does not satisfy the assumption that pUq.

L^{pUq}:

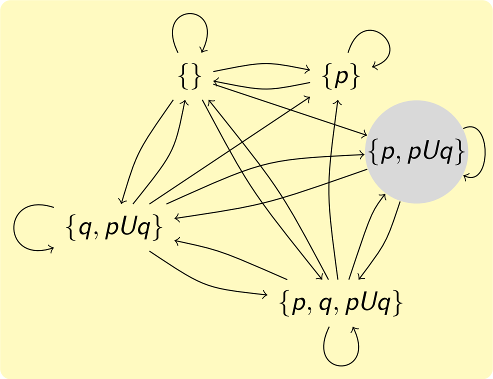


L^{ϕ} is not sufficient!
Consider the following scenario:

Model: 

The model never performs an illegal transition, but still it does not satisfy the assumption that pUq.

L^{pUq}:



Construction of E^{ϕ}

- Let the states be sets of formulae $\phi_0 U \phi_1 \in cl(\phi)$.
  - $\phi_0 U \phi_1 \in u$ means that in state u we are waiting for ϕ_1.
- Label transitions with consistent valuations of $cl(\phi)$.
- For a transition $u \xrightarrow{a} v$
  - If $u = \emptyset \wedge \phi_0 U \phi_1 \in a \wedge \phi_1 \notin a$ then $\phi_0 U \phi_1 \in v$
  - If $\phi_0 U \phi_1 \in u \wedge \phi_1 \notin a$ then $\phi_0 U \phi_1 \in v$

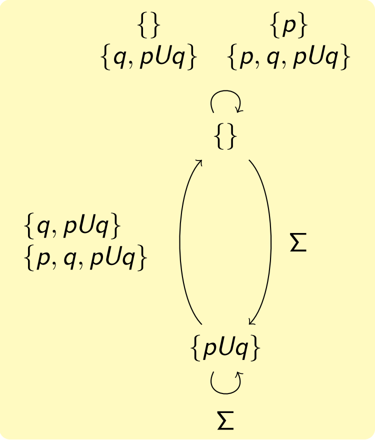
Construction of E^{ϕ}

- Let the states be sets of formulae $\phi_0 U \phi_1 \in cl(\phi)$.
  - $\phi_0 U \phi_1 \in u$ means that in state u we are waiting for ϕ_1.
- Label transitions with consistent valuations of $cl(\phi)$.
- For a transition $u \xrightarrow{a} v$
  - If $u = \emptyset \wedge \phi_0 U \phi_1 \in a \wedge \phi_1 \notin a$ then $\phi_0 U \phi_1 \in v$
  - If $\phi_0 U \phi_1 \in u \wedge \phi_1 \notin a$ then $\phi_0 U \phi_1 \in v$

LTL $\rightarrow$ Büchi Automaton: E^{ϕ}

Example: $\phi = pUq$

$$\Sigma = \left\{ \begin{array}{l} \{\} \\ \{p\} \\ \{p, pUq\} \\ \{q, pUq\} \\ \{p, q, pUq\} \end{array} \right\}$$



LTL $\rightarrow$ Büchi Automaton: $(L^\phi, E^\phi) \rightarrow A^\phi$

Combine L^ϕ and E^ϕ into a Büchi automaton A^ϕ:

1. States: cross product
2. $(l, e) \xrightarrow{a} (l', e')$ iff $l \xrightarrow{a} l'$ and $e \xrightarrow{a} e'$.
   - Intuition: Require both satisfaction of both L^ϕ and E^ϕ
3. Starting states: $(l, \emptyset)$ where $\phi \in l$
   - Intuition: Assume that ϕ holds for the start of computation.
4. Accepting states: $(l, \emptyset)$
   - Intuition: Any state is fine as long as we are not still waiting for ϕ_1 in $\phi_0 U \phi_1$.

Combine L^ϕ and E^ϕ into a Büchi automaton A^ϕ:

1. States: cross product
2. $(l, e) \xrightarrow{a} (l', e')$ iff $l \xrightarrow{a} l'$ and $e \xrightarrow{a} e'$.
   - Intuition: Require both satisfaction of both L^ϕ and E^ϕ
3. Starting states: $(l, \emptyset)$ where $\phi \in l$
   - Intuition: Assume that ϕ holds for the start of computation.
4. Accepting states: $(l, \emptyset)$
   - Intuition: Any state is fine as long as we are not still waiting for ϕ_1 in $\phi_0 U \phi_1$.

LTL $\rightarrow$ Büchi Automaton: $(L^\phi, E^\phi) \rightarrow A^\phi$

Combine L^ϕ and E^ϕ into a Büchi automaton A^ϕ:

1. States: cross product
2. $(l, e) \xrightarrow{a} (l', e')$ iff $l \xrightarrow{a} l'$ and $e \xrightarrow{a} e'$.
   - Intuition: Require both satisfaction of both L^ϕ and E^ϕ
3. Starting states: $(l, \emptyset)$ where $\phi \in l$
   - Intuition: Assume that ϕ holds for the start of computation.
4. Accepting states: $(l, \emptyset)$
   - Intuition: Any state is fine as long as we are not still waiting for ϕ_1 in $\phi_0 U \phi_1$.

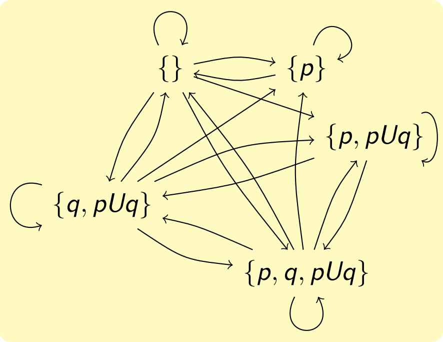
LTL $\rightarrow$ Büchi Automaton: $(L^\phi, E^\phi) \rightarrow A^\phi$

Combine L^ϕ and E^ϕ into a Büchi automaton A^ϕ:

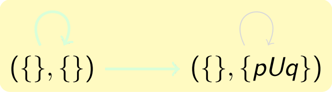
1. States: cross product
2. $(l, e) \xrightarrow{a} (l', e')$ iff $l \xrightarrow{a} l'$ and $e \xrightarrow{a} e'$.
   - Intuition: Require both satisfaction of both L^ϕ and E^ϕ
3. Starting states: $(l, \emptyset)$ where $\phi \in l$
   - Intuition: Assume that ϕ holds for the start of computation.
4. Accepting states: $(l, \emptyset)$
   - Intuition: Any state is fine as long as we are not still waiting for ϕ_1 in $\phi_0 U \phi_1$.

LTL $\rightarrow$ Büchi Automaton: $(L^\phi, E^\phi) \rightarrow A^\phi$: Example

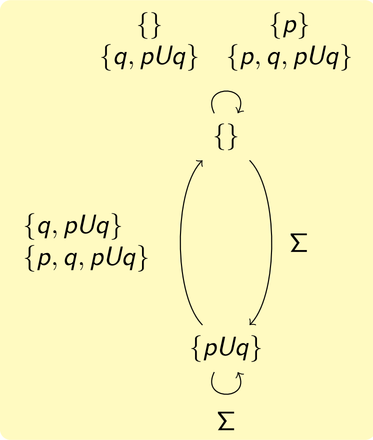
L^{pUq}



A^{pUq} (partial)

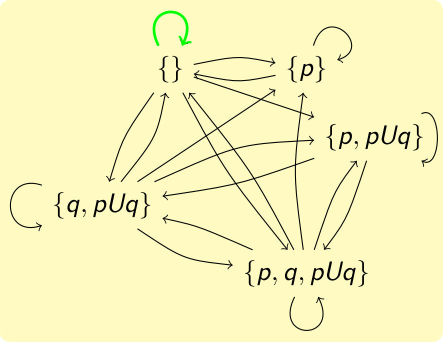


E^{pUq}

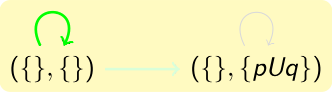


LTL $\rightarrow$ Büchi Automaton: $(L^{\phi}, E^{\phi}) \rightarrow A^{\phi}$: Example

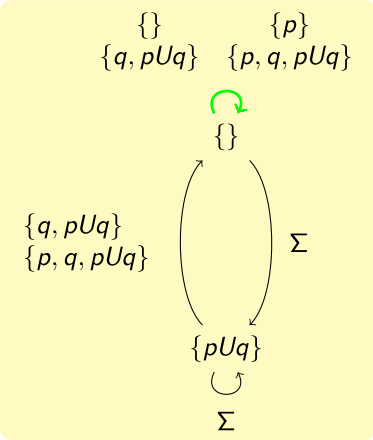
L^{pUq}



A^{pUq} (partial)

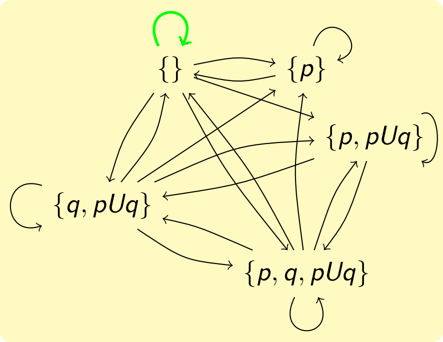


E^{pUq}

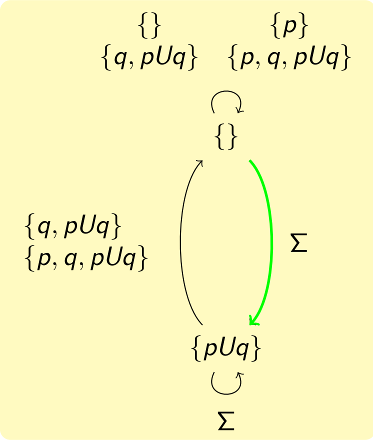


LTL $\to$ Büchi Automaton: $(L^{\phi}, E^{\phi}) \to A^{\phi}$: Example

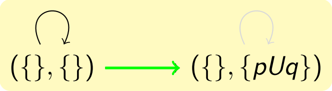
L^{pUq}



E^{pUq}

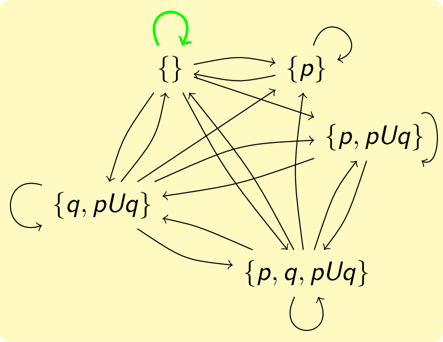


A^{pUq} (partial)

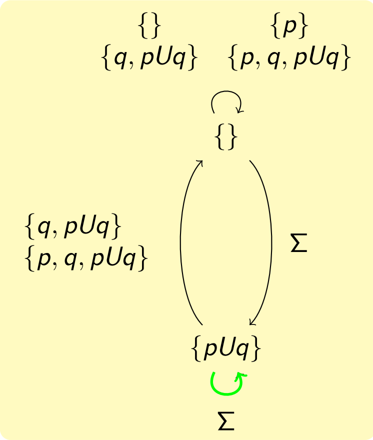


LTL $\rightarrow$ Büchi Automaton: $(L^{\phi}, E^{\phi}) \rightarrow A^{\phi}$: Example

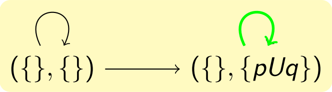
L^{pUq}



E^{pUq}

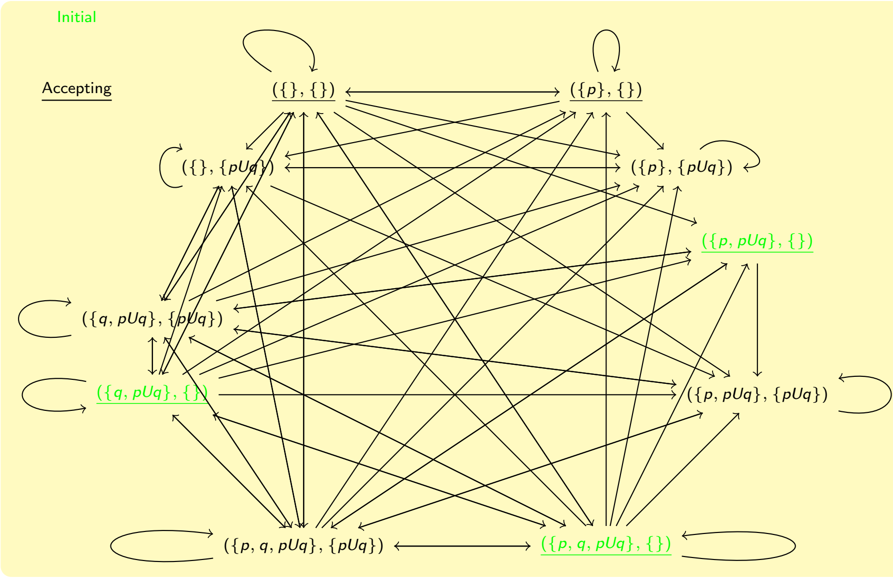


A^{pUq} (partial)



LTL → Büchi Automaton: $(L^{\phi}, E^{\phi}) \to A^{\phi}$: Example

A^{pUq}



How to use $A^{\neg\phi}$ to verify a model

Using $A^{\neg\phi}$ to verify a model P:

- Compute the model automaton A^P.
- Compute the intersection $A = A^{\neg\phi} \bigcap A^P$
  - Basically run $A^{\neg\phi}$ and A^P together and accept whenever $A^{\neg\phi}$ accepts.
- Check emptiness for A.
  - DFS

How to use $A^{\neg\phi}$ to verify a model

Using $A^{\neg\phi}$ to verify a model P:

- Compute the model automaton A^P.
- Compute the intersection $A = A^{\neg\phi} \bigcap A^P$
  - Basically run $A^{\neg\phi}$ and A^P together and accept whenever $A^{\neg\phi}$ accepts.
- Check emptiness for A.
  - DFS

How to use $A^{\neg\phi}$ to verify a model

Using $A^{\neg\phi}$ to verify a model P:

- Compute the model automaton A^P.
- Compute the intersection $A = A^{\neg\phi} \bigcap A^P$
  - Basically run $A^{\neg\phi}$ and A^P together and accept whenever $A^{\neg\phi}$ accepts.
- Check emptiness for A.
  - DFS