Büchi Automata and their Application to Software Verification

Finite Automata Theory and Formal Languages

Wolfgang Ahrendt

22nd April 2013

Motivating Temporal Logic?

But How to Express Properties Involving State Changes?

In any run of a program P

- n will become greater than 0 eventually?
- n changes its value infinitely often

etc.

Motivating Temporal Logic?

But How to Express Properties Involving State Changes?

In any run of a program P

- n will become greater than 0 eventually?
- n changes its value infinitely often

etc.

Linear Temporal Logic: talks about (infinite) traces of states

Semantics of Propositional Logic

Interpretation $\mathcal{I}$

Assigns a truth value to each propositional variable

$$\mathcal{I} : \mathcal{P} \rightarrow \{T, F\}$$

Semantics of Propositional Logic

Interpretation $\mathcal{I}$

Assigns a truth value to each propositional variable

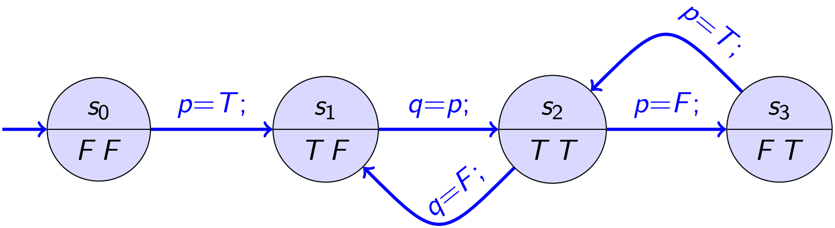
$$\mathcal{I} : \mathcal{P} \to \{T, F\}$$

Example

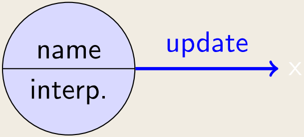
Let $\mathcal{P} = \{p, q\}$

	p	q
$\mathcal{I}$	F	F
$\mathcal{I}'$	F	T
$\mathcal{I}''$	T	F
$\mathcal{I}'''$	T	T

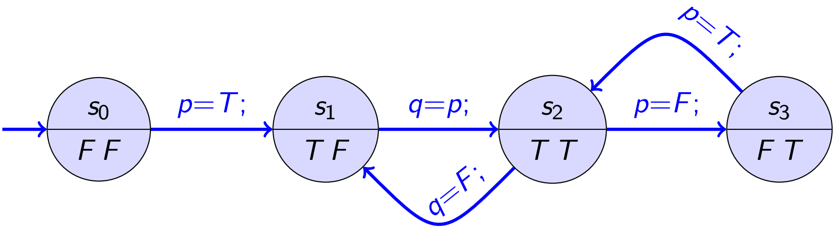
Transition systems (aka Kripke Structures)



Notation



Transition systems (aka Kripke Structures)



- Each state s_i has its own propositional interpretation I_i
  - Convention: list values of variables in ascending lexicographic order
- Computations, or runs, are *infinite* paths through states
  - Intuitively ‘finite’ runs modelled by looping on final states
- In general, infinitely many different runs possible
- How to express (for example) that p changes its value infinitely often in each run?

(Linear) Temporal Logic

An extension of propositional logic that
allows to specify properties of all runs

(Linear) Temporal Logic—Syntax

An extension of propositional logic that
allows to specify properties of all runs

Syntax

Based on propositional signature and syntax

Extension with three connectives:

Always If ϕ is a formula then so is $\Box\phi$

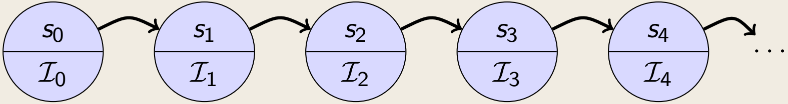
Eventually If ϕ is a formula then so is $\Diamond\phi$

Concrete Syntax

	text book	SPIN
Always	$\Box$	[]
Eventually	$\Diamond$	<>

Temporal Logic—Semantics

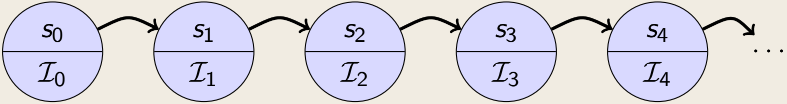
A run σ is an infinite chain of states



$\mathcal{I}_j$ propositional interpretation of variables in j-th state
Write more compactly $s_0\, s_1\, s_2\, s_3 \dots$

Temporal Logic—Semantics

A run σ is an infinite chain of states



$\mathcal{I}_j$ propositional interpretation of variables in j-th state
Write more compactly $s_0\, s_1\, s_2\, s_3 \ldots$

If $\sigma = s_0\, s_1 \cdots$, then $\sigma|_i$ denotes the suffix $s_i\, s_{i+1} \cdots$ of σ.

Temporal Logic—Semantics (Cont'd)

Valuation of temporal formula relative to run: infinite sequence of states

Temporal Logic—Semantics (Cont'd)

Valuation of temporal formula relative to run: infinite sequence of states

Definition (Validity Relation)

Validity of temporal formula depends on runs $\sigma = s_0\, s_1 \ldots$

$\sigma \models p \qquad$ iff $\quad \mathcal{I}_0(p) = T$, for $p \in \mathcal{P}$.

Temporal Logic—Semantics (Cont'd)

Valuation of temporal formula relative to run: infinite sequence of states

Definition (Validity Relation)

Validity of temporal formula depends on runs $\sigma = s_0\, s_1 \ldots$

$\sigma \models p$	iff	$\mathcal{I}_0(p) = T$, for $p \in \mathcal{P}$.
$\sigma \models \neg\phi$	iff	not $\sigma \models \phi$ (write $\sigma \not\models \phi$)
$\sigma \models \phi \land \psi$	iff	$\sigma \models \phi$ and $\sigma \models \psi$
$\sigma \models \phi \lor \psi$	iff	$\sigma \models \phi$ or $\sigma \models \psi$
$\sigma \models \phi \to \psi$	iff	$\sigma \not\models \phi$ or $\sigma \models \psi$

Temporal Logic—Semantics (Cont'd)

Valuation of temporal formula relative to run: infinite sequence of states

Definition (Validity Relation)

Validity of temporal formula depends on runs $\sigma = s_0\, s_1 \ldots$

$$
\begin{array}{lll}
\sigma \models p & \text{iff} & \mathcal{I}_0(p) = T\text{, for } p \in \mathcal{P}. \\
\sigma \models \neg\phi & \text{iff} & \text{not } \sigma \models \phi \quad (\text{write } \sigma \not\models \phi) \\
\sigma \models \phi \land \psi & \text{iff} & \sigma \models \phi \text{ and } \sigma \models \psi \\
\sigma \models \phi \lor \psi & \text{iff} & \sigma \models \phi \text{ or } \sigma \models \psi \\
\sigma \models \phi \to \psi & \text{iff} & \sigma \not\models \phi \text{ or } \sigma \models \psi
\end{array}
$$

Temporal connectives?

Temporal Logic—Semantics (Cont'd)

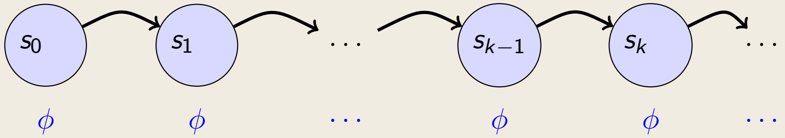
Run σ

Definition (Validity Relation for Temporal Connectives)

Given a run $\sigma = s_0\, s_1 \cdots$

Temporal Logic—Semantics (Cont'd)

Run σ



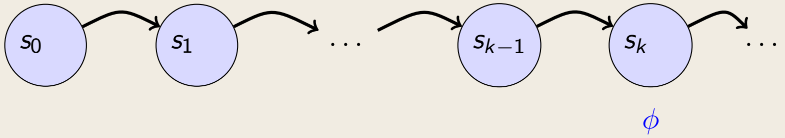
Definition (Validity Relation for Temporal Connectives)

Given a run $\sigma = s_0\, s_1 \cdots$

$\sigma \models \Box\phi \quad \text{iff} \quad \sigma|_k \models \phi$ for all $k \geq 0$

Temporal Logic—Semantics (Cont'd)

Run σ



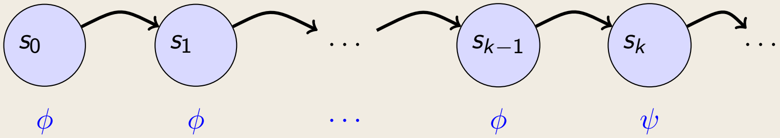
Definition (Validity Relation for Temporal Connectives)

Given a run $\sigma = s_0\, s_1 \cdots$

$$\begin{array}{lll} \sigma \models \Box\phi & \text{iff} & \sigma|_k \models \phi \text{ for all } k \geq 0 \\ \sigma \models \Diamond\phi & \text{iff} & \sigma|_k \models \phi \text{ for some } k \geq 0 \end{array}$$

Temporal Logic—Semantics (Cont'd)

Run σ



Definition (Validity Relation for Temporal Connectives)

Given a run $\sigma = s_0\, s_1 \cdots$

$\sigma \models \Box\phi \quad \text{iff} \quad \sigma|_k \models \phi$ for all $k \geq 0$

$\sigma \models \Diamond\phi \quad \text{iff} \quad \sigma|_k \models \phi$ for some $k \geq 0$

Transition Systems: Formal Definition

Definition (Transition System)

A transition system $\mathcal{T} = (S, \mathit{Ini}, \delta, \mathcal{I})$ is composed of a set of states S, a set $\emptyset \neq \mathit{Ini} \subseteq S$ of initial states, a transition relation $\delta \subseteq S \times S$, and a labeling $\mathcal{I}$ of each state $s \in S$ with a propositional interpretation $\mathcal{I}_s$.

Definition (Run of Transition System)

A run of $\mathcal{T}$ is a sequence of states $\sigma = s_0\, s_1 \cdots$ such that
$s_0 \in \mathit{Ini}$ and for all i is $s_i \in S$ as well as $(s_i, s_{i+1}) \in \delta$.

ω-Languages

Given a finite alphabet (vocabulary) Σ

A word $w \in \Sigma^*$ is a finite sequence

$$w = a_o \cdots a_n$$

with $a_i \in \Sigma, i \in \{0, \ldots, n\}$

$\mathcal{L} \subseteq \Sigma^*$ is called a language

ω-Languages

Given a finite alphabet (vocabulary) Σ

An ω-word $w \in \Sigma^\omega$ is an infinite sequence

$$w = a_o \cdots a_k \cdots$$

with $a_i \in \Sigma, i \in \mathbb{N}$

$\mathcal{L}^\omega \subseteq \Sigma^\omega$ is called an ω-language

Büchi Automaton

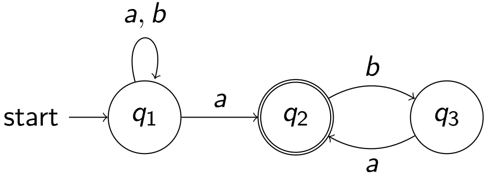
Definition (Büchi Automaton)

A (non-deterministic) Büchi automaton over an alphabet Σ consists of a

- finite, non-empty set of locations Q
- a non-empty set of initial/start locations $I \subseteq Q$
- a set of accepting locations $F = \{F_1, \ldots, F_n\} \subseteq Q$
- a transition relation $\delta \subseteq Q \times \Sigma \times Q$

Example

$\Sigma = \{a, b\}, Q = \{q_1, q_2, q_3\}, I = \{q_1\}, F = \{q_2\}$



Büchi Automaton—Executions and Accepted Words

Definition (Execution)

Let $\mathcal{B} = (Q, I, F, \delta)$ be a Büchi automaton over alphabet Σ.
An execution of $\mathcal{B}$ is a pair (w, v), with

- $w = a_o \cdots a_k \cdots \in \Sigma^\omega$
- $v = q_o \cdots q_k \cdots \in Q^\omega$

where $q_0 \in I$, and $(q_i, a_i, q_{i+1}) \in \delta$, for all $i \in \mathbb{N}$

Büchi Automaton—Executions and Accepted Words

Definition (Execution)

Let $\mathcal{B} = (Q, I, F, \delta)$ be a Büchi automaton over alphabet Σ.
An execution of $\mathcal{B}$ is a pair (w, v), with

- $w = a_o \cdots a_k \cdots \in \Sigma^\omega$
- $v = q_o \cdots q_k \cdots \in Q^\omega$

where $q_0 \in I$, and $(q_i, a_i, q_{i+1}) \in \delta$, for all $i \in \mathbb{N}$

Definition (Accepted Word)

A Büchi automaton $\mathcal{B}$ accepts a word $w \in \Sigma^\omega$, if there exists an execution (w, v) of $\mathcal{B}$ where some accepting location $f \in F$ appears infinitely often in v

Büchi Automaton—Language

Let $\mathcal{B} = (Q, I, F, \delta)$ be a Büchi automaton, then

$$\mathcal{L}^\omega(\mathcal{B}) = \{w \in \Sigma^\omega | w \in \Sigma^\omega \text{ is an accepted word of } \mathcal{B}\}$$

denotes the ω-language recognised by $\mathcal{B}$.

Büchi Automaton—Language

Let $\mathcal{B} = (Q, I, F, \delta)$ be a Büchi automaton, then

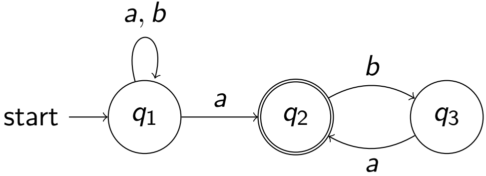
$$\mathcal{L}^\omega(\mathcal{B}) = \{w \in \Sigma^\omega | w \in \Sigma^\omega \text{ is an accepted word of } \mathcal{B}\}$$

denotes the ω-language recognised by $\mathcal{B}$.

An ω-language for which an accepting Büchi automaton exists is called ω-regular language.

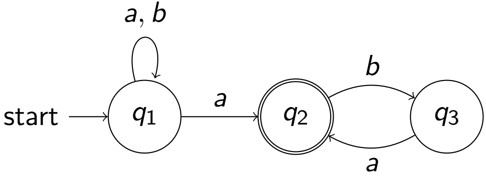
Example, ω-Regular Expression

Which language is accepted by the following Büchi automaton?



Example, ω-Regular Expression

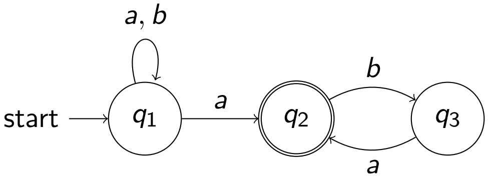
Which language is accepted by the following Büchi automaton?



Solution: $(a+b)^*(ab)^\omega$ [NB: $(ab)^\omega = a(ba)^\omega$]

Example, ω-Regular Expression

Which language is accepted by the following Büchi automaton?



Solution: $(a+b)^*(ab)^\omega$ [NB: $(ab)^\omega = a(ba)^\omega$]

ω-regular expressions like standard regular expression

ab a **then** b

$a+b$ a **or** b

a^* arbitrarily, but finitely often a

new: a^ω infinitely often a

Decidability, Closure Properties

Many properties for regular finite automata hold also for Büchi automata

Theorem (Decidability)

It is decidable whether the accepted language $\mathcal{L}^\omega(\mathcal{B})$ of a Büchi automaton $\mathcal{B}$ is empty.

Decidability, Closure Properties

Many properties for regular finite automata hold also for Büchi automata

Theorem (Decidability)

It is decidable whether the accepted language $\mathcal{L}^\omega(\mathcal{B})$ of a Büchi automaton $\mathcal{B}$ is empty.

Theorem (Closure properties)

The set of ω-regular languages is closed with respect to intersection, union and complement:

- *if $\mathcal{L}_1, \mathcal{L}_2$ are ω-regular then $\mathcal{L}_1 \cap \mathcal{L}_2$ and $\mathcal{L}_1 \cup \mathcal{L}_2$ are ω-regular*
- *$\mathcal{L}$ is ω-regular then $\Sigma^\omega \backslash \mathcal{L}$ is ω-regular*

Decidability, Closure Properties

Many properties for regular finite automata hold also for Büchi automata

Theorem (Decidability)

It is decidable whether the accepted language $\mathcal{L}^\omega(\mathcal{B})$ of a Büchi automaton $\mathcal{B}$ is empty.

Theorem (Closure properties)

The set of ω-regular languages is closed with respect to intersection, union and complement:

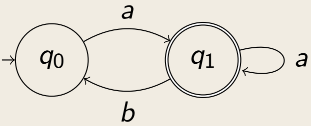
- *if $\mathcal{L}_1, \mathcal{L}_2$ are ω-regular then $\mathcal{L}_1 \cap \mathcal{L}_2$ and $\mathcal{L}_1 \cup \mathcal{L}_2$ are ω-regular*
- *$\mathcal{L}$ is ω-regular then $\Sigma^\omega \backslash \mathcal{L}$ is ω-regular*

But in contrast to regular finite automata

Non-deterministic Büchi automata are strictly more expressive than deterministic ones

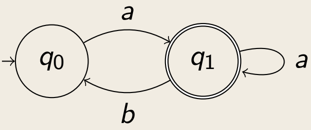
Büchi Automata—More Examples

Language:



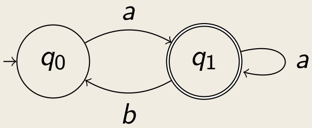
Büchi Automata—More Examples

Language: $a(a + ba)^\omega$

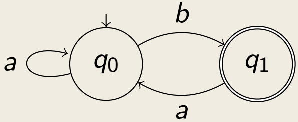


Büchi Automata—More Examples

Language: $a(a + ba)^\omega$

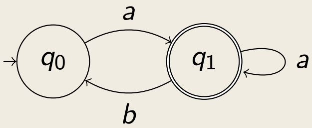


Language:

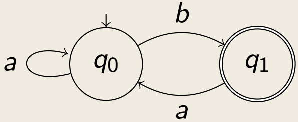


Büchi Automata—More Examples

Language: $a(a + ba)^\omega$



Language: $(a^* ba)^\omega$



Linear Temporal Logic and Büchi Automata

LTL and Büchi Automata are connected

Definition (Validity Relation)

Given a transition system $\mathcal{T} = (S, Ini, \delta, \mathcal{I})$, a temporal formula ϕ is valid in $\mathcal{T}$ (write $\mathcal{T} \models \phi$) iff $\sigma \models \phi$ for all runs σ of $\mathcal{T}$.

A run of the transition system is an infinite sequence of interpretations I

Linear Temporal Logic and Büchi Automata

LTL and Büchi Automata are connected

Definition (Validity Relation)

Given a transition system $\mathcal{T} = (S, \mathit{Ini}, \delta, \mathcal{I})$, a temporal formula ϕ is valid in $\mathcal{T}$ (write $\mathcal{T} \models \phi$) iff $\sigma \models \phi$ for all runs σ of $\mathcal{T}$.

A run of the transition system is an infinite sequence of interpretations I

Intended Connection

Given an LTL formula ϕ:

Construct a Büchi automaton accepting exactly those runs (infinite sequences of interpretations) that satisfy ϕ

Encoding an LTL Formula as a Büchi Automaton

$\mathcal{P}$ set of propositional variables, e.g., $\mathcal{P} = \{r, s\}$

Alphabet Σ of Büchi automaton

A state transition of Büchi automaton must represent an interpretation

Let Σ (i.e., the alphabet of the automata) be set of all interpretations over $\mathcal{P}$, i.e., $\Sigma = 2^{\mathcal{P}}$

Example

$\Sigma = \big\{\emptyset, \{r\}, \{s\}, \{r, s\}\big\}$

$$I_{\emptyset}(r) = F, I_{\emptyset}(s) = F, I_{\{r\}}(r) = T, I_{\{r\}}(s) = F, \ldots$$

Büchi Automaton for LTL Formula By Example

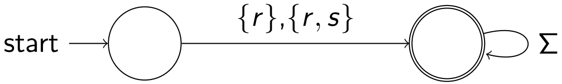
Example (Büchi automaton for formula r over $\mathcal{P} = \{r, s\}$)

A Büchi automaton $\mathcal{B}$ accepting exactly those runs σ satisfying r

Büchi Automaton for LTL Formula By Example

Example (Büchi automaton for formula r over $\mathcal{P} = \{r, s\}$)

A Büchi automaton $\mathcal{B}$ accepting exactly those runs σ satisfying r

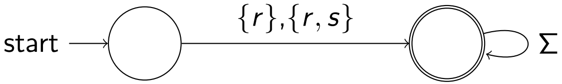


In the first state s_0 (of σ) at least r must hold, the rest is arbitrary

Büchi Automaton for LTL Formula By Example

Example (Büchi automaton for formula r over $\mathcal{P} = \{r, s\}$)

A Büchi automaton $\mathcal{B}$ accepting exactly those runs σ satisfying r



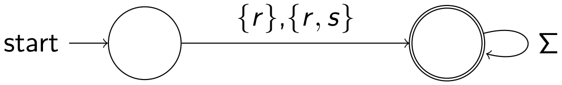
In the first state s_0 (of σ) at least r must hold, the rest is arbitrary

Example (Büchi automaton for formula $\Box r$ over $\mathcal{P} = \{r, s\}$)

Büchi Automaton for LTL Formula By Example

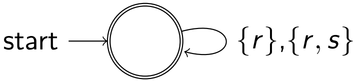
Example (Büchi automaton for formula r over $\mathcal{P} = \{r, s\}$)

A Büchi automaton $\mathcal{B}$ accepting exactly those runs σ satisfying r



In the first state s_0 (of σ) at least r must hold, the rest is arbitrary

Example (Büchi automaton for formula $\Box r$ over $\mathcal{P} = \{r, s\}$)

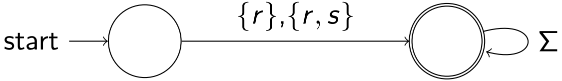


In **all** states s (of σ) at least r must hold

Büchi Automaton for LTL Formula By Example

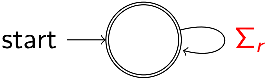
Example (Büchi automaton for formula r over $\mathcal{P} = \{r, s\}$)

A Büchi automaton $\mathcal{B}$ accepting exactly those runs σ satisfying r



In the first state s_0 (of σ) at least r must hold, the rest is arbitrary

Example (Büchi automaton for formula $\Box r$ over $\mathcal{P} = \{r, s\}$)



$$\Sigma_r := \{I | I \in \Sigma, r \in I\}$$

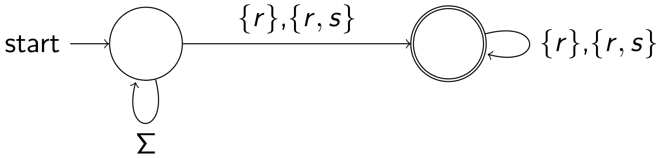
In all states s (of σ) at least r must hold

Büchi Automaton for LTL Formula By Example

Example (Büchi automaton for formula $\Diamond\Box r$ over $\mathcal{P} = \{r, s\}$)

Büchi Automaton for LTL Formula By Example

Example (Büchi automaton for formula $\Diamond\Box r$ over $\mathcal{P} = \{r, s\}$)



Model Checking

Check whether a formula is valid in all runs of a transition system

Given a transition system $\mathcal{T}$ (e.g., derived from a Promela program)

Verification task: is the LTL formula ϕ satisfied in all runs of $\mathcal{T}$, i.e.,

$$\mathcal{T} \models \phi \quad ?$$

Model Checking

Check whether a formula is valid in all runs of a transition system

Given a transition system $\mathcal{T}$ (e.g., derived from a PROMELA program)

Verification task: is the LTL formula ϕ satisfied in all runs of $\mathcal{T}$, i.e.,

$$\mathcal{T} \models \phi \quad ?$$

In the following: Basic principle behind SPIN model checking

SPIN Model Checking—Overview

$$\mathcal{T} \models \phi \quad ?$$

1. Represent transition system $\mathcal{T}$ as Büchi automaton $\mathcal{B}_{\mathcal{T}}$ such that $\mathcal{B}_{\mathcal{T}}$ accepts exactly those words corresponding to runs through $\mathcal{T}$

SPIN **Model Checking—Overview**

$$\mathcal{T} \models \phi \quad ?$$

1. Represent transition system $\mathcal{T}$ as Büchi automaton $\mathcal{B}_{\mathcal{T}}$ such that $\mathcal{B}_{\mathcal{T}}$ accepts exactly those words corresponding to runs through $\mathcal{T}$
2. Construct Büchi automaton $\mathcal{B}_{\neg\phi}$ for negation of formula ϕ

Spin Model Checking—Overview

$$\mathcal{T} \models \phi \quad ?$$

1. Represent transition system $\mathcal{T}$ as Büchi automaton $\mathcal{B}_{\mathcal{T}}$ such that $\mathcal{B}_{\mathcal{T}}$ accepts exactly those words corresponding to runs through $\mathcal{T}$
2. Construct Büchi automaton $\mathcal{B}_{\neg\phi}$ for negation of formula ϕ
3. If
$$\mathcal{L}^{\omega}(\mathcal{B}_{\mathcal{T}}) \cap \mathcal{L}^{\omega}(\mathcal{B}_{\neg\phi}) = \emptyset$$
then ϕ holds.

SPIN Model Checking—Overview

$$\mathcal{T} \models \phi \quad ?$$

1. Represent transition system $\mathcal{T}$ as Büchi automaton $\mathcal{B}_{\mathcal{T}}$ such that $\mathcal{B}_{\mathcal{T}}$ accepts exactly those words corresponding to runs through $\mathcal{T}$
2. Construct Büchi automaton $\mathcal{B}_{\neg\phi}$ for negation of formula ϕ
3. If

$$\mathcal{L}^{\omega}(\mathcal{B}_{\mathcal{T}}) \cap \mathcal{L}^{\omega}(\mathcal{B}_{\neg\phi}) = \emptyset$$

then ϕ holds.

If

$$\mathcal{L}^{\omega}(\mathcal{B}_{\mathcal{T}}) \cap \mathcal{L}^{\omega}(\mathcal{B}_{\neg\phi}) \neq \emptyset$$

then each element of the set is a counterexample for ϕ.

SPIN **Model Checking—Overview**

$$\mathcal{T} \models \phi \quad ?$$

1. Represent transition system $\mathcal{T}$ as Büchi automaton $\mathcal{B}_{\mathcal{T}}$ such that $\mathcal{B}_{\mathcal{T}}$ accepts exactly those words corresponding to runs through $\mathcal{T}$
2. Construct Büchi automaton $\mathcal{B}_{\neg\phi}$ for negation of formula ϕ
3. If

   $$\mathcal{L}^{\omega}(\mathcal{B}_{\mathcal{T}}) \cap \mathcal{L}^{\omega}(\mathcal{B}_{\neg\phi}) = \emptyset$$

   then ϕ holds.

   If

   $$\mathcal{L}^{\omega}(\mathcal{B}_{\mathcal{T}}) \cap \mathcal{L}^{\omega}(\mathcal{B}_{\neg\phi}) \neq \emptyset$$

   then each element of the set is a counterexample for ϕ.

   To check $\mathcal{L}^{\omega}(\mathcal{B}_{\mathcal{T}}) \cap \mathcal{L}^{\omega}(\mathcal{B}_{\neg\phi})$ construct intersection automaton and search for cycle through accepting state

Representing a Model as a Büchi Automaton

First Step: Represent transition system $\mathcal{T}$ as Büchi automaton $\mathcal{B}_{\mathcal{T}}$ accepting exactly those words representing a run of $\mathcal{T}$

Example

```
active proctype P () {
do
  :: atomic {
      !wQ; wP = true
     };
     Pcs = true;
     atomic {
      Pcs = false;
      wP = false
     }
od }
```

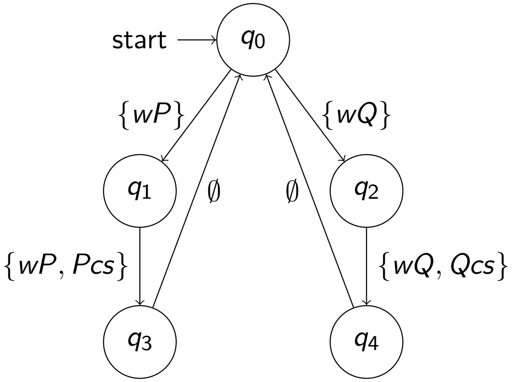
First location skipped and second made `atomic` just to keep automaton small; similar code for process `Q`

Representing a Model as a Büchi Automaton

First Step: Represent transition system $\mathcal{T}$ as Büchi automaton $\mathcal{B}_{\mathcal{T}}$ accepting exactly those words representing a run of $\mathcal{T}$

Example

```
active proctype P () {
do
  :: atomic {
      !wQ; wP = true
     };
     Pcs = true;
     atomic {
      Pcs = false;
      wP = false
     }
od }
```



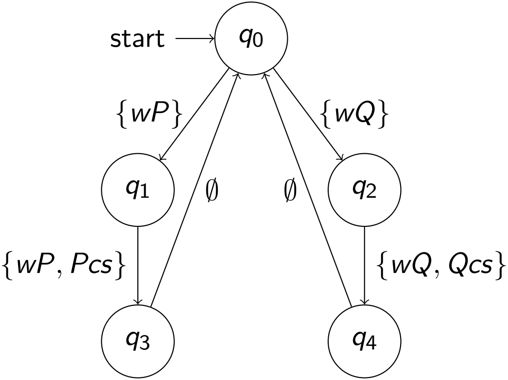
First location skipped and second made `atomic` just to keep automaton small; similar code for process `Q`

Representing a Model as a Büchi Automaton

First Step: Represent transition system $\mathcal{T}$ as Büchi automaton $\mathcal{B}_{\mathcal{T}}$ accepting exactly those words representing a run of $\mathcal{T}$

Example

```
active proctype P () {
do
  :: atomic {
      !wQ; wP = true
     };
     Pcs = true;
     atomic {
      Pcs = false;
      wP = false
     }
od }
```



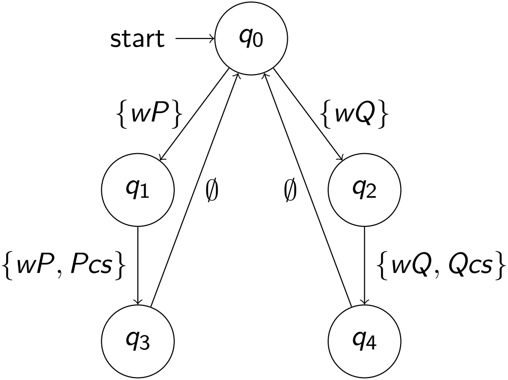
Which are the accepting locations?

Representing a Model as a Büchi Automaton

First Step: Represent transition system $\mathcal{T}$ as Büchi automaton $\mathcal{B}_{\mathcal{T}}$ accepting exactly those words representing a run of $\mathcal{T}$

Example

```
active proctype P () {
do
  :: atomic {
      !wQ; wP = true
     };
     Pcs = true;
     atomic {
      Pcs = false;
      wP = false
     }
od }
```



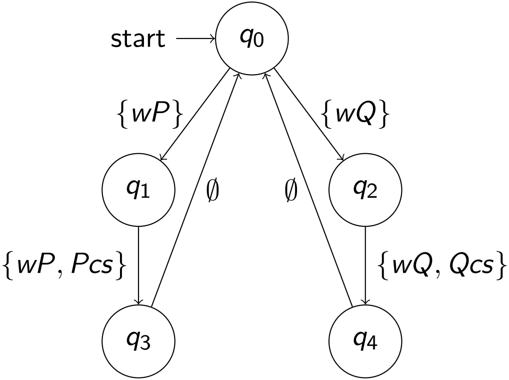
Which are the accepting locations? All!

Representing a Model as a Büchi Automaton

First Step: Represent transition system $\mathcal{T}$ as Büchi automaton $\mathcal{B}_{\mathcal{T}}$ accepting exactly those words representing a run of $\mathcal{T}$

Example

```
active proctype P () {
do
  :: atomic {
      !wQ; wP = true
     };
     Pcs = true;
     atomic {
      Pcs = false;
      wP = false
     }
od }
```



The property we want to check is $\phi = \Box\neg Pcs$ (which does not hold)

Büchi Automaton $B_{\neg\phi}$ for $\neg\phi$

Second Step:

Construct Büchi Automaton corresponding to negated LTL formula

$\mathcal{T} \models \phi$ holds iff there is no accepting run of $\mathcal{T}$ for $\neg\phi$

Simplify $\neg\phi = \neg\Box\neg Pcs = \Diamond Pcs$

Büchi Automaton $B_{\neg\phi}$ for $\neg\phi$

Second Step:

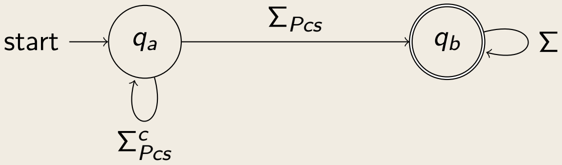
Construct Büchi Automaton corresponding to negated LTL formula

$\mathcal{T} \models \phi$ holds iff there is no accepting run of $\mathcal{T}$ for $\neg\phi$

Simplify $\neg\phi = \neg\Box\neg Pcs = \Diamond Pcs$

Büchi Automaton $\mathcal{B}_{\neg\phi}$

$$\mathcal{P} = \{wP, wQ, Pcs, Qcs\},\ \Sigma = 2^{\mathcal{P}}$$



$$\Sigma_{Pcs} = \{I | I \in \Sigma, Pcs \in I\}, \quad \Sigma^c_{Pcs} = \Sigma - \Sigma_{Pcs}$$

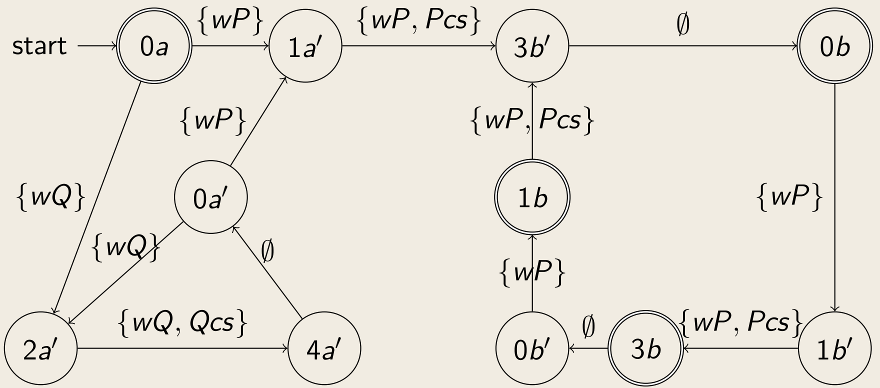
Checking for Emptiness of Intersection Automaton

Third Step: $\mathcal{L}^\omega(\mathcal{B}_\mathcal{T}) \cap \mathcal{L}^\omega(\mathcal{B}_{\neg\phi}) = \emptyset$?

Checking for Emptiness of Intersection Automaton

Third Step: $\mathcal{L}^{\omega}(\mathcal{B}_{\mathcal{T}}) \cap \mathcal{L}^{\omega}(\mathcal{B}_{\neg\phi}) = \emptyset$?

Intersection Automaton



Checking for Emptiness of Intersection Automaton

Third Step: $\mathcal{L}^\omega(\mathcal{B}_\mathcal{T}) \cap \mathcal{L}^\omega(\mathcal{B}_{\neg\phi}) \neq \emptyset$

Counterexample

Intersection Automaton