

Introduction

Modelling parallel systems

Linear Time Properties

**Regular Properties**

Linear Temporal Logic

Computation-Tree Logic

Equivalences and Abstraction



*Idea:* define regular LT properties to be those languages of finite words over the alphabet  $2^A$  that have a representation by a finite automata

*Idea:* define **regular LT properties** to be those languages of **finite words** over the alphabet  $2^{AP}$  that have a representation by a **finite automata**

~~For~~ regular safety properties:

**NFA**-representation for the **bad prefixes**

~~Idea:~~ ~~defn~~ **regular LT properties** to be those languages of **finite words** over the alphabet  **$2^A$**  that have a representation by a **finite automata**

~~reg~~ regular safety properties:

**NFA**-representation for the **bad prefixes**

~~reg~~ other regular LT properties:

representation by **-automata**, i.e.,  
acceptors for **finite words**

Introduction

Modelling parallel systems

Linear Time Properties

**Regular Properties**

regular safety properties

-regular properties

model checking with Buchi automata

Linear Temporal Logic

Computation-Tree Logic

Equivalences and Abstraction



# Recall: definition of safety properties

is2.5-15b

Let  $E$  be a LT property over  $AP$ , i.e.,  $E \subseteq 2^{AP}$ .

$E$  is called a **safety property** if for all words

$$w = A_0 A_1 A_2 \dots \in 2^{AP} \setminus E$$

there exists a **finite prefix**  $A_0 A_1 \dots A_n$  of  $w$  such that *none* of the words  $A_0 A_1 \dots A_n B_{n+1} B_{n+2} B_{n+3} \dots$  belongs to  $E$ , i.e.,

$$E \subseteq 2^{AP} : A_0 \dots A_n \text{ is a prefix of } w \implies w \in E$$

Such words  $A_0 A_1 \dots A_n$  are called **bad prefixes** for  $E$ .

$$\text{BadPref}(E) \stackrel{\text{def}}{=} \text{set of bad prefixes for } E \subseteq 2^{AP}$$





Let  $E \subseteq 2^{AP}$  be a safety property.

$E$  is called regular if the language

$BadPref = \{ \text{set of all bad prefixes for } E \}$

is regular.

Let  $E \subseteq 2^{AP}$  be a safety property.

$E$  is called regular if the language

$BadPref = \text{set of all bad prefixes for } E \subseteq 2^{AP}$

is regular.

Let  $E \subseteq 2^{AP}$  be a safety property.

$E$  is called regular iff the language

$BadPref = \text{set of all bad prefixes for } E \subseteq 2^{AP}$

$BadPref = L(A)$  for some NFA  $A$   
over the alphabet  $2^{AP}$

is regular.

# Nondeterministic finite automata (NFA)

is2.5-15

NFA  $A = (Q, \Sigma, \delta, Q_0, F)$

$Q$  finite set of states

$\Sigma$  alphabet

$\delta : Q \times \Sigma \rightarrow 2^Q$  transition relation

$Q_0 \subseteq Q$  set of initial states

$F \subseteq Q$  set of final states, also called accept states

NFA  $A = (Q, \Sigma, \delta, q_0, F)$

$Q$  finite set of states

$\Sigma$  alphabet

$\delta : Q \times \Sigma \rightarrow 2^Q$  transition relation

$q_0 \in Q$  set of initial states

$F \subseteq Q$  set of final states, also called accept states

run for a word  $A_0 A_1 \dots A_{n-1}$  :

state sequence  $= q_0 q_1 \dots q_n$  where  $q_0 \in Q_0$

and  $q_{i+1} \in \delta(q_i, A_i)$  for  $0 \leq i < n$

NFA  $A = (Q, \Sigma, \delta, q_0, F)$

$Q$  finite set of states

$\Sigma$  alphabet

$\delta : Q \times \Sigma \rightarrow 2^Q$  transition relation

$q_0 \in Q$  set of initial states

$F \subseteq Q$  set of final states, also called accept states

run for a word  $A_0 A_1 \dots A_{n-1}$  :

state sequence  $= q_0 q_1 \dots q_n$  where  $q_0 \in Q_0$

and  $q_{i+1} \in \delta(q_i, A_i)$  for  $0 \leq i < n$

run is called accepting if  $q_n \in F$

NFA  $A = (Q, \Sigma, Q_0, F)$

$Q$  finite set of states

$\Sigma$  alphabet

$\delta : Q \times \Sigma \rightarrow 2^Q$  transition relation

$Q_0 \subseteq Q$  set of initial states

$F \subseteq Q$  set of final states, also called accept states

accepted language  $L(A)$  is given by:

$L(A) = \{ w \mid w \text{ is a word over } \Sigma \text{ that has an accepting run in } A \}$



NFA  $A = (Q, \Sigma, \delta, Q_0, F)$

$Q$  finite set of states

$\Sigma$  alphabet      here:  $\Sigma = 2^A$

$\delta : Q \times \Sigma \rightarrow 2^Q$  transition relation

$Q_0 \subseteq Q$  set of initial states

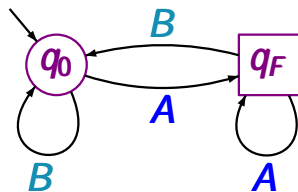
$F \subseteq Q$  set of final states, also called accept states

accepted language  $L(A)$  is given by:

$L(A) = \{ \text{set of words over } \Sigma \text{ that have an accepting run in } A \}$

# Notations in pictures for NFA

is2.5-15a



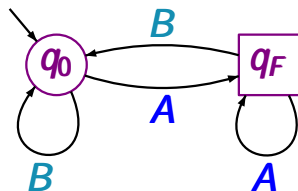
↘ initial state

○ non-final state

□ final state

# Notations in pictures for NFA

is2.5-15a



↘ initial state

○ non-final state

□ final state

NFA  $A$  with state space  $\{q_0, q_F\}$

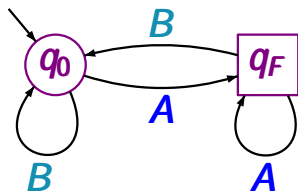
$q_0$  initial state

$q_F$  final state

alphabet =  $\{A, B\}$

# Notations in pictures for NFA

is2.5-15a



↘ initial state

○ non-final state

□ final state

accepted language  $L(A)$ :

set of all words over  $\{A, B\}$   
ending with letter  $A$

for transitions in **NFA** over the alphabet  $= 2^{AP}$

NFA  $A = (Q, \delta, Q_0, F)$  over the alphabet  $\Sigma = 2^{AP}$   
*symbolic notation* for the labels of transitions:

If  $\phi$  is a propositional formula over  $AP$  then

$\delta^{-1}(\phi)$  stands for the set of transitions  $\delta^{-1}(\phi)$   
 where  $A \models \phi$  such that  $A \models \phi$

NFA  $A = (Q, \delta, Q_0, F)$  over the alphabet  $\Sigma = 2^{AP}$   
*symbolic notation* for the labels of transitions:

If  $\phi$  is a propositional formula over  $AP$  then

$q \xrightarrow{\phi} p$  stands for the set of transitions  $q \xrightarrow{A} p$   
 where  $A \models AP$  such that  $A \models \phi$

Example: if  $AP = \{a, b, c\}$  then

$q \xrightarrow{a} p = q \xrightarrow{A} p : A = \{a, c\} \text{ or } A = \{a\}$

NFA  $A = (Q, \delta, Q_0, F)$  over the alphabet  $\Sigma = 2^{AP}$   
*symbolic notation* for the labels of transitions:

If  $\phi$  is a propositional formula over  $AP$  then

$q \xrightarrow{\phi} p$  stands for the set of transitions  $q \xrightarrow{A} p$   
 where  $A \subseteq AP$  such that  $A \models \phi$

Example: if  $AP = \{a, b, c\}$  then

$q \xrightarrow{a \vee c} p = q \xrightarrow{A} p : A = \{a, c\} \text{ or } A = \{a\}$

$q \xrightarrow{\text{true}} p = q \xrightarrow{A} p : A \subseteq AP$



A safety property  $E \subseteq 2^{AP}$  is called regular if

$BadPref =$  set of all bad prefixes for  $E \subseteq 2^{AP}$

$BadPref = L(A)$  for some NFA  $A$   
over the alphabet  $2^{AP}$

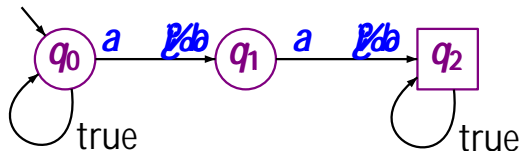
is regular.

A safety property  $E \subseteq 2^{AP}$  is called regular if

$BadPref =$  set of all bad prefixes for  $E \subseteq 2^{AP}$

$BadPref = L(A)$  for some NFA  $A$   
over the alphabet  $2^{AP}$

is regular.



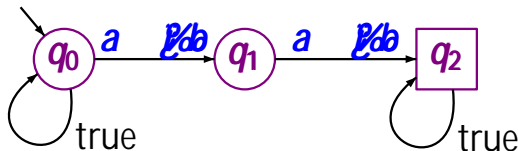
$AP = \{a, b\}$

A safety property  $E$  over  $2^{AP}$  is called regular if

$BadPref =$  set of all bad prefixes for  $E$  over  $2^{AP}$  is

$BadPref = L(A)$  for some NFA  $A$  over the alphabet  $2^{AP}$

is regular.



$AP = \{a, b\}$

symbolic notation:

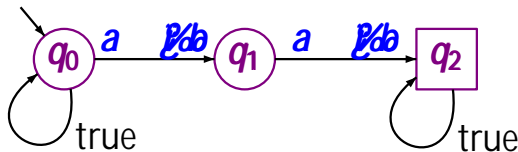
$a \text{ } b = \{a\}$

A safety property  $E$  is called regular if

$BadPref =$  set of all bad prefixes for  $E$

$BadPref = L(A)$  for some NFA  $A$  over the alphabet

is regular.



$AP = \{a, b\}$

symbolic notation:

$a \text{ } \overline{b} = \{a\}$

safety property  $E$ :  $a \text{ } \overline{b}$  never holds twice in a row

*Every red phase is preceded by a yellow phase*

*Every red phase is preceded by a yellow phase*

set of all infinite words  $A_0 A_1 A_2 \dots$  s.t. for all  $i \geq 0$ :

*red*  $A_i = i + 1$  and *yellow*  $A_{i+1}$

# Example: regular safety property

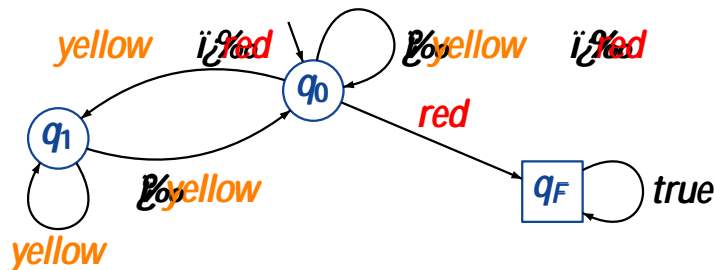
is2.5-16

*Every red phase is preceded by a yellow phase*

set of all infinite words  $A_0 A_1 A_2 \dots$  s.t. for all  $i \geq 0$ :

**red**  $A_i = 1$  and **yellow**  $A_{i+1} = 1$

**DFA** for all (possibly non-minimal) bad prefixes



# Example: regular safety property

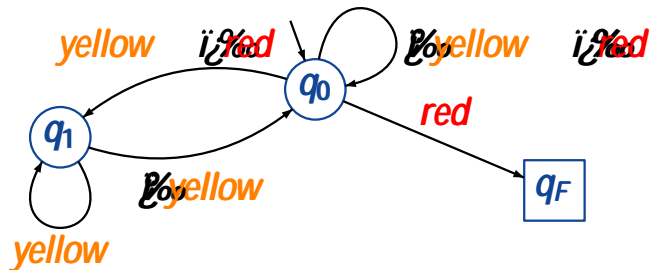
is2.5-16

*Every red phase is preceded by a yellow phase*

set of all infinite words  $A_0 A_1 A_2 \dots$  s.t. for all  $i \geq 0$ :

$\text{red } A_i = i + 1$  and  $\text{yellow } A_{i+1}$

DFA for minimal bad prefixes





# Bad prefixes vs minimal bad prefixes

is2.5-14a

Let  $E \subseteq 2^{AP}$  be a safety property.

*BadPref* = set of all bad prefixes for  $E$

*MinBadPref* = set of minimal bad prefixes for  $E$

Claim: *BadPref* is regular      *MinBadPref* is regular

# Bad prefixes vs minimal bad prefixes

is2.5-14a

Let  $E \subseteq 2^{AP}$  be a safety property.

*BadPref* = set of all bad prefixes for  $E$

*MinBadPref* = set of minimal bad prefixes for  $E$

Claim: *BadPref* is regular      *MinBadPref* is regular

Let  $A$  be an NFA for *MinBadPref*.

# Bad prefixes vs minimal bad prefixes

is2.5-14a

Let  $E \subseteq 2^{AP}$  be a safety property.

**BadPref** = set of all bad prefixes for  $E$

**MinBadPref** = set of minimal bad prefixes for  $E$

Claim: **BadPref** is regular      **MinBadPref** is regular

Let  $A$  be an NFA for **MinBadPref**.

An NFA  $A$  for **BadPref** is obtained from  $A$  by adding self-loops  $p \xrightarrow{\text{true}} p$  to all **bad** states  $p$ .

# Bad prefixes vs minimal bad prefixes

is2.5-14a

Let  $E \subseteq 2^{AP}$  be a safety property.

**BadPref** = set of all bad prefixes for  $E$

**MinBadPref** = set of minimal bad prefixes for  $E$

Claim: **BadPref** is regular      **MinBadPref** is regular

Let  $A$  be an NFA for **MinBadPref**.

An NFA  $A$  for **BadPref** is obtained from  $A$  by adding self-loops  $p \xrightarrow{\text{true}} p$  to all  $\text{Val}$  states  $p$ .

Let  $A$  be a DFA for **BadPref**.

# Bad prefixes vs minimal bad prefixes

is2.5-14a

Let  $E \in 2^{AP}$  be a safety property.

**BadPref** = set of all bad prefixes for  $E$

**MinBadPref** = set of minimal bad prefixes for  $E$

Claim: **BadPref** is regular      **MinBadPref** is regular

~~Let~~ Let  $A$  be an NFA for **MinBadPref**.

An NFA  $A$  for **BadPref** is obtained from  $A$  by adding self-loops  $p \xrightarrow{\text{true}} p$  to all ~~bad~~ states  $p$ .

~~Let~~ Let  $A$  be a DFA for **BadPref**.

A DFA  $A$  for **MinBadPref** is obtained from  $A$  by removing all outgoing transitions of ~~bad~~ states.

Every **invariant** is regular.

Every **invariant** is regular.

correct.

Every **invariant** is regular.

**correct.**

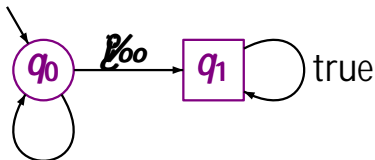
Let **E** be an invariant with invariant condition



Every **invariant** is regular.

correct.

Let **E** be an invariant with invariant condition

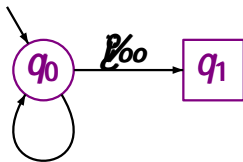


is a DFA for the language of all bad prefixes

Every invariant is regular.

correct.

Let  $E$  be an invariant with invariant condition



is a DFA for the language of all minimal bad prefixes

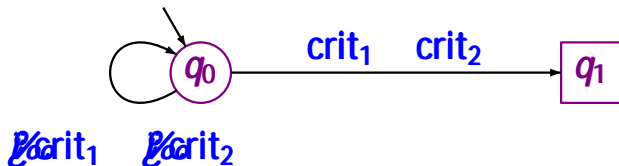
The two processes are never simultaneously  
in their critical sections.

## Example: DFA for *MUTEX*

is2.5-19

The two processes are **never simultaneously**  
in their **critical sections**

**DFA** for minimal bad prefixes over the alphabet  
 $2^{AP}$  where  $AP = \{\text{crit}_1, \text{crit}_2\}$



Every **safety property** is regular.

Every **safety property** is regular.

**wrong.**

Every **safety property** is regular.

wrong. e.g.,  $AP = \{\text{pay}, \text{drink}\}$

$E$  = set of all infinite words  $A_0 A_1 A_2 \dots$  ( $2^{AP}$ )  
 such that for all  $j \in \mathbb{N}$ :

$\{i \in \mathbb{N} : \text{pay} \text{ at } A_i\}$        $\{i \in \mathbb{N} : \text{drink} \text{ at } A_i\}$

Every **safety property** is regular.

wrong. e.g.,  $AP = \{\text{pay}, \text{drink}\}$

$E$  = set of all infinite words  $A_0 A_1 A_2 \dots$  ( $2^{AP}$ )  
 such that for all  $j \in \mathbb{N}$ :  
 $\{i \in \mathbb{N} : \text{pay}(A_i)\} \neq \emptyset$        $\{i \in \mathbb{N} : \text{drink}(A_i)\} \neq \emptyset$

~~No~~  $E$  is a safety property, but

~~No~~ the language of (minimal) bad prefixes is *not* regular





*given:*      ~~No~~ TS  $T$   
                 regular safety property  $E$   
                 (represented by an **NFA** for its bad prefixes)

*question:*   does  $T \models E$  hold ?

*given:* ~~No~~ TS  $T$   
regular safety property  $E$   
(represented by an **NFA** for its bad prefixes) ~~Yes~~

*question:* does  $T \models E$  hold ?

*method:* relies on an analogy between the tasks:

~~Yes~~ checking language inclusion for **NFA**

~~Yes~~ model checking regular safety properties

language inclusion  
for NFA

$L(A_1)$     $L(A_2)$  ?

verification of regular  
safety properties

$Traces(T)$     $E$  ?

language inclusion  
for NFA

$L(A_1) \subseteq L(A_2) ?$

check whether

$L(A_1) \cap (L(A_2)^c)$

is empty

verification of regular  
safety properties

$Traces(T) \subseteq E ?$

language inclusion  
for NFA

verification of regular  
safety properties

$L(A_1) \subseteq L(A_2) ?$

$Traces(T) \subseteq E ?$

check whether  
 $L(A_1) \cap (L(A_2))^c$   
is empty

1. complement  $A_2$ , i.e.,  
construct NFA  $\overline{A_2}$  with  
 $L(\overline{A_2}) = L(A_2)^c$

language inclusion  
for NFA

verification of regular  
safety properties

$L(A_1) \subseteq L(A_2) ?$

$Traces(T) \subseteq E ?$

check whether

$L(A_1) \cap \overline{L(A_2)}$

is empty

1. complement  $A_2$ , i.e.,  
construct NFA  $\overline{A_2}$  with  
 $L(\overline{A_2}) = \overline{L(A_2)}$
2. construct NFA  $A$  with  
 $L(A) = L(A_1) \cap L(\overline{A_2})$

language inclusion  
for NFA

verification of regular  
safety properties

$L(A_1) \subseteq L(A_2) ?$

$Traces(T) \subseteq E ?$

check whether

$L(A_1) \cap \overline{L(A_2)}$

is empty

1. complement  $A_2$ , i.e.,  
construct NFA  $\overline{A_2}$  with  
 $L(\overline{A_2}) = \overline{L(A_2)}$
2. construct NFA  $A$  with  
 $L(A) = L(A_1) \cap \overline{L(A_2)}$
3. check if  $L(A) = \emptyset$



language inclusion  
for NFA

$L(A_1) \subseteq L(A_2) ?$

check whether

$L(A_1) \cap \overline{L(A_2)}$

is empty

1. complement  $A_2$ , i.e.,  
construct NFA  $\overline{A_2}$  with  
 $L(\overline{A_2}) = \overline{L(A_2)}$
2. construct NFA  $A$  with  
 $L(A) = L(A_1) \cap L(\overline{A_2})$
3. check if  $L(A) = \emptyset$

verification of regular  
safety properties

$Traces(T) \subseteq E ?$

check whether

$Traces(T) \cap \overline{BadPref}$

is empty

language inclusion  
for NFA

$L(A_1) \subseteq L(A_2) ?$

check whether  
 $L(A_1) \cap (L(A_2))^c$   
is empty

1. complement  $A_2$ , i.e.,  
construct NFA  $\overline{A_2}$  with  
 $L(\overline{A_2}) = L(A_2)^c$
2. construct NFA  $A$  with  
 $L(A) = L(A_1) \cap L(\overline{A_2})$
3. check if  $L(A) = \emptyset$

verification of regular  
safety properties

$Traces(T) \subseteq E ?$

check whether  
 $Traces(T) \cap BadPref$   
is empty

1. construct NFA  $A$   
for the bad prefixes  
 $L(\overline{A}) = BadPref$

language inclusion  
for NFA

$L(A_1) \subseteq L(A_2) ?$

check whether  
 $L(A_1) \cap \overline{L(A_2)}$   
is empty

1. complement  $A_2$ , i.e.,  
construct NFA  $\overline{A_2}$  with  
 $L(\overline{A_2}) = \overline{L(A_2)}$
2. construct NFA  $A$  with  
 $L(A) = L(A_1) \cap \overline{L(A_2)}$
3. check if  $L(A) = \emptyset$

verification of regular  
safety properties

$Traces(T) \subseteq E ?$

check whether  
 $Traces(T) \cap \overline{BadPref}$   
is empty

1. construct NFA  $A$   
for the bad prefixes  
 $L(\overline{A}) = \overline{BadPref}$
2. construct TS  $T$  with  
 $Traces(T) = \dots$

language inclusion  
for NFA

$L(A_1) \subseteq L(A_2) ?$

check whether  
 $L(A_1) \cap \overline{L(A_2)}$   
is empty

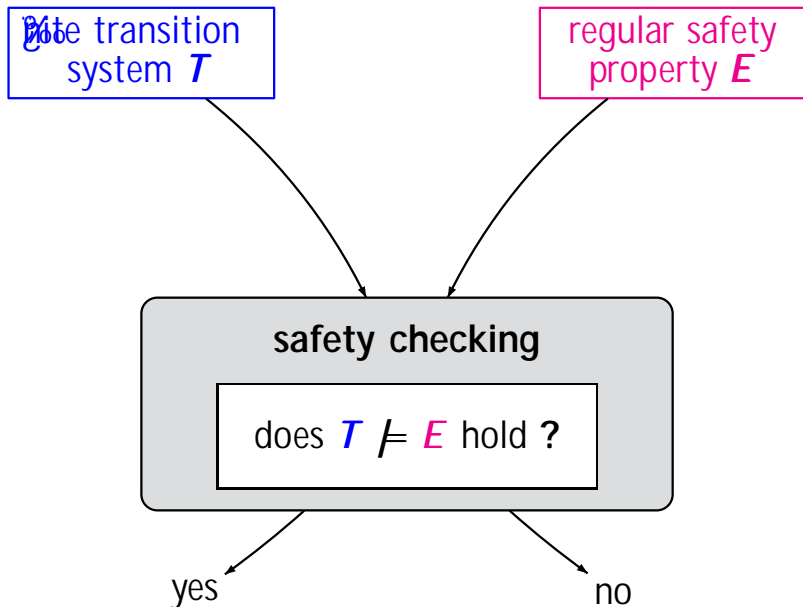
1. complement  $A_2$ , i.e.,  
construct NFA  $\overline{A_2}$  with  
 $L(\overline{A_2}) = \overline{L(A_2)}$
2. construct NFA  $A$  with  
 $L(A) = L(A_1) \cap \overline{L(A_2)}$
3. check if  $L(A) = \emptyset$

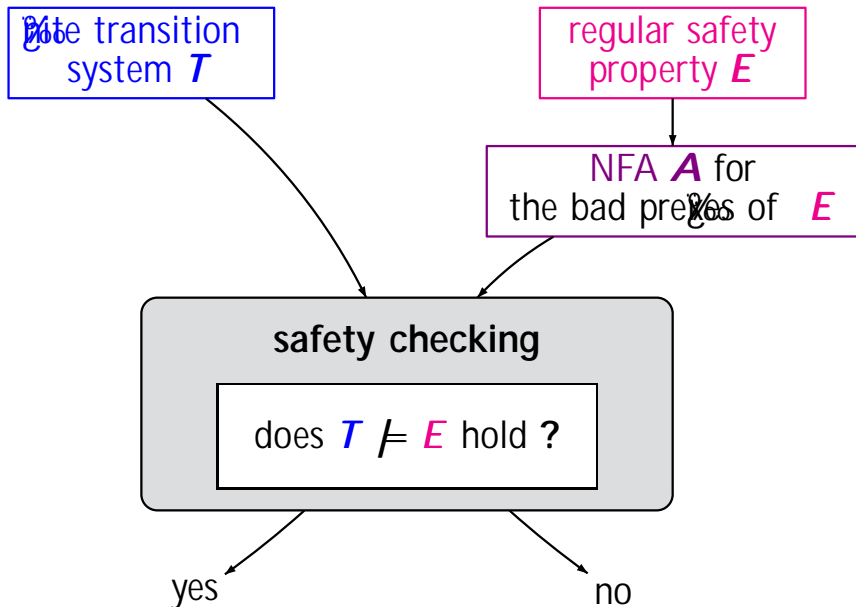
verification of regular  
safety properties

$Traces(T) \subseteq E ?$

check whether  
 $Traces(T) \cap \overline{BadPref}$   
is empty

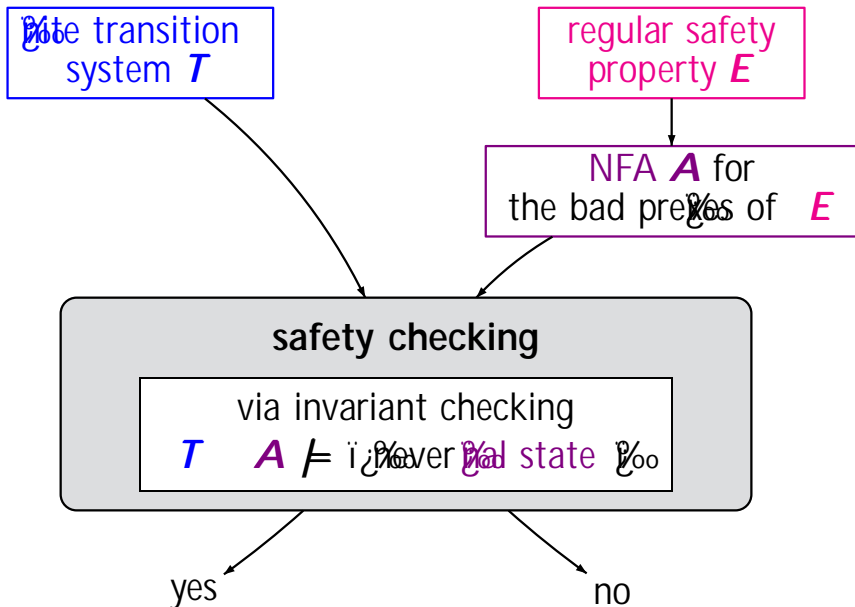
1. construct NFA  $A$   
for the bad prefixes  
 $L(\overline{A}) = \overline{BadPref}$
2. construct TS  $T$  with  
 $Traces(T) = \dots$
3. invariant checking  
for  $T$





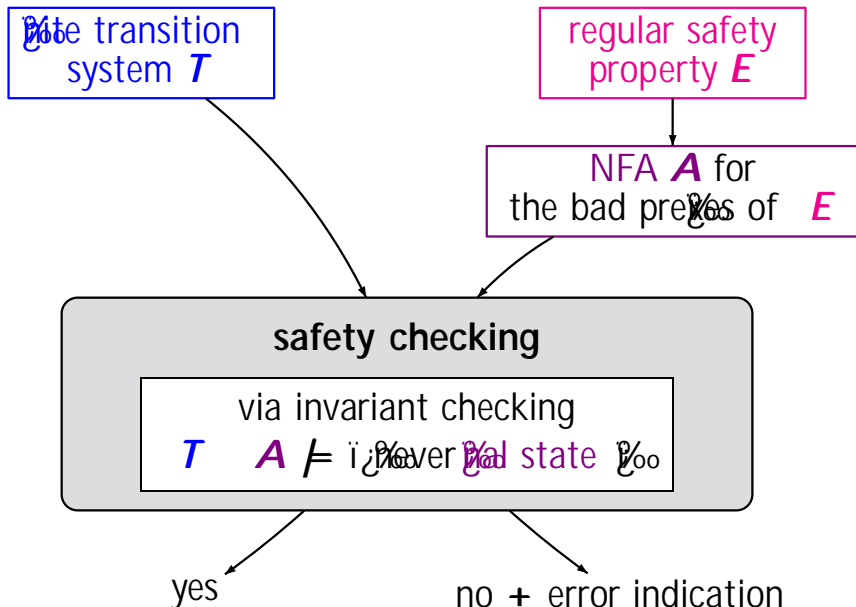
# Checking regular safety properties

is2.5-21



# Checking regular safety properties

is2.5-21







# Product of a TS and an NFA

is2.5-22

Note transition system

$$T = (S, Act, \rightarrow, s_0, AP, L)$$

NFA for bad prefixes

$$A = (Q, 2^{AP}, \rightarrow, q_0, F)$$

$s_0$



$s_1$



$s_2$



$\vdots$



$s_n$

path  
fragment

# Product of a TS and an NFA

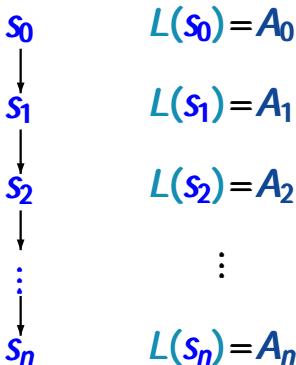
is2.5-22

Note transition system

$$T = (S, Act, \rightarrow, s_0, AP, L)$$

NFA for bad prefixes

$$A = (Q, 2^{AP}, \rightarrow, q_0, F)$$



path  
fragment

trace

# Product of a TS and an NFA

is2.5-22

Finite transition system

$$T = (S, Act, s_0, AP, L)$$

$s_0$



$s_1$



$s_2$



$\vdots$



$s_n$

$$L(s_0) = A_0$$

$$L(s_1) = A_1$$

$$L(s_2) = A_2$$

$\vdots$

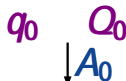
$$L(s_n) = A_n$$

path  
fragment

trace

NFA for bad prefixes

$$A = (Q, 2^{AP}, Q_0, F)$$



$q_1$



$q_2$



$\vdots$



$q_n$



$q_{n+1}$

run for  $trace()$

# Product of a TS and an NFA

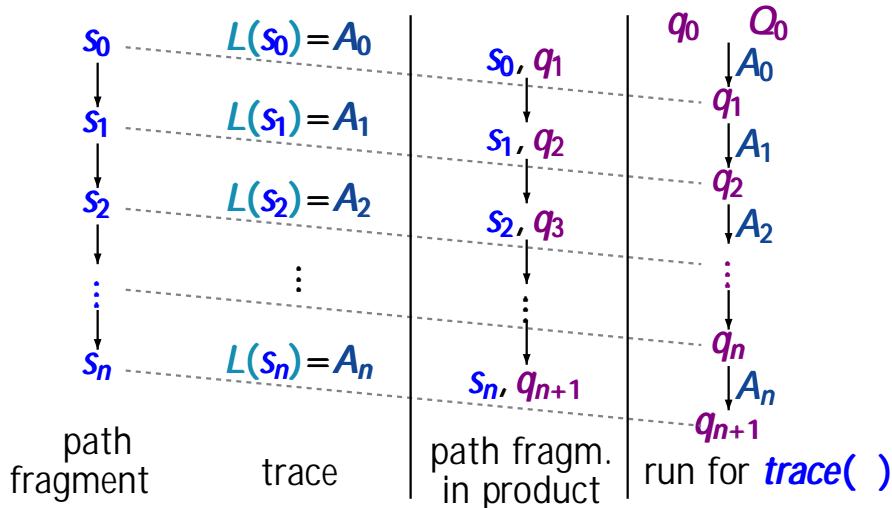
is2.5-22

Finite transition system

$$T = (S, Act, \rightarrow, s_0, AP, L)$$

NFA for bad prefixes

$$A = (Q, 2^{AP}, \rightarrow, q_0, F)$$





$T = (S, Act, \rightarrow, S_0, AP, L)$  transition system

$A = (Q, 2^{AP}, \rightarrow, Q_0, F)$  NFA

$T = (S, Act, \rightarrow, S_0, AP, L)$  transition system

$A = (Q, 2^{AP}, \rightarrow, Q_0, F)$  NFA

product-TS  $T \times A \stackrel{\text{def}}{=} (\cancel{S} \times \cancel{Q}, Act, \cancel{\rightarrow}, S_0, AP, L)$



$T = (S, Act, \rightarrow, S_0, AP, L)$  transition system

$A = (Q, 2^{AP}, \rightarrow, Q_0, F)$  NFA

product-TS  $T \times A \stackrel{\text{def}}{=} (S \times Q, Act, \rightarrow, S_0, AP, L)$

$$\frac{s \times q \quad s \quad q \quad (q, L(s))}{s, q \times s, q}$$

$T = (S, Act, \rightarrow, S_0, AP, L)$  transition system

$A = (Q, 2^{AP}, \rightarrow, Q_0, F)$  NFA

product-TS  $T \times A \stackrel{\text{def}}{=} (S \times Q, Act, \rightarrow, S_0, AP, L)$

$$\frac{s \times q \rightarrow s' \quad (q, L(s))}{s, q \rightarrow s', q}$$

initial states:  $S_0 = \{s_0, q : s_0 \in S_0, q \in Q_0, L(s_0)\}$

# Product transition system

is2.5-25

$T = (S, Act, \rightarrow, S_0, AP, L)$  transition system

$A = (Q, 2^{AP}, \rightarrow, Q_0, F)$  NFA

product-TS  $T \times A \stackrel{\text{def}}{=} (S \times Q, Act, \rightarrow, S_0, AP, L)$

$$\frac{s \times q \quad s \quad q \quad (q, L(s))}{s, q \times s, q}$$

initial states:  $S_0 = s_0, q : s_0 \quad S_0, q \quad Q_0, L(s_0)$

for  $P \subseteq Q$  and  $A \subseteq AP$ :  $(P, A) = \{ (p, A) \mid p \in P \}$

$T = (S, Act, \rightarrow, S_0, AP, L)$  transition system

$A = (Q, 2^{AP}, \rightarrow, Q_0, F)$  NFA

product-TS  $T \times A \stackrel{\text{def}}{=} (S \times Q, Act, \rightarrow, S_0, AP, L)$

$$\frac{s \times q \quad s \quad q \quad (q, L(s))}{s, q \rightarrow s, q}$$

initial states:  $S_0 = \{s_0, q : s_0 \mid S_0, q \mid Q_0, L(s_0)\}$

set of atomic propositions:  $AP = Q$

$T = (S, Act, \rightarrow, S_0, AP, L)$  transition system

$A = (Q, 2^{AP}, \rightarrow, Q_0, F)$  NFA

product-TS  $T \times A \stackrel{\text{def}}{=} (S \times Q, Act, \rightarrow, S_0, AP, L)$

$$\frac{s \times q \quad (q, L(s))}{s, q \rightarrow s, q}$$

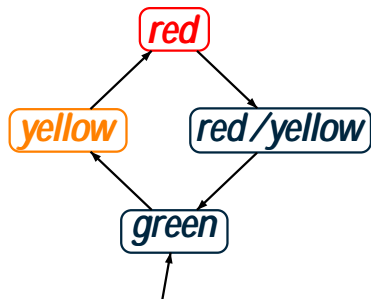
initial states:  $S_0 = \{s_0, q : s_0 \in S_0, q \in Q_0, L(s_0)\}$

set of atomic propositions:  $AP = Q$

labeling function:  $L(s, q) = \{q\}$

# Example: product-TS

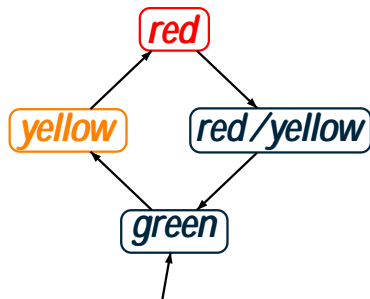
is2.5-26



transition system  $T$  over  
 $AP = \{\text{red}, \text{yellow}\}$

# Example: product-TS

is2.5-26



transition system  $T$  over

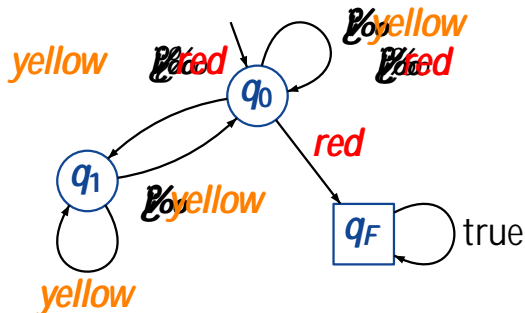
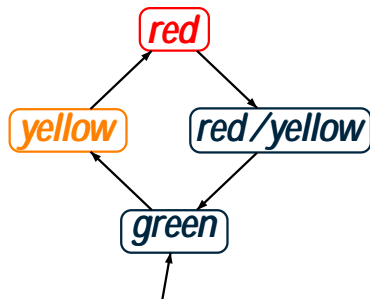
$AP = \{\text{red}, \text{yellow}\}$

$T$  satisfies the safety property  $E$

*Every red phase is preceded by a yellow phase*

# Example: product-TS

is2.5-26



transition system  $T$  over  
 $AP = \{\text{red}, \text{yellow}\}$

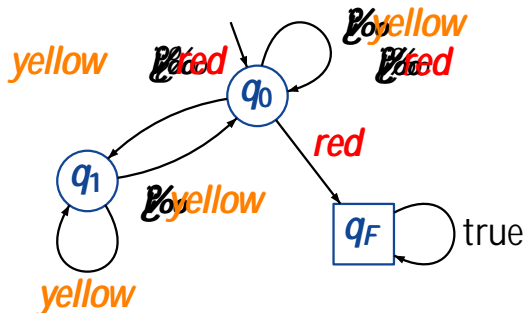
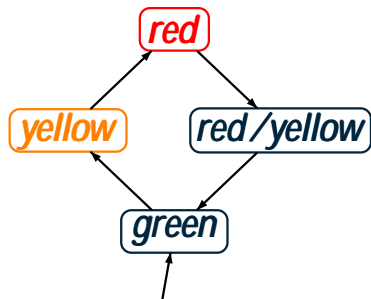
DFA  $A$  for the  
 bad prefix property  $E$

$T$  satisfies the safety property  $E$   
*Every red phase is preceded by a yellow phase*



# Example: product-TS

is2.5-26



green  $q_0$

red/yellow  $q_0$

yellow  $q_1$

red  $q_0$

...

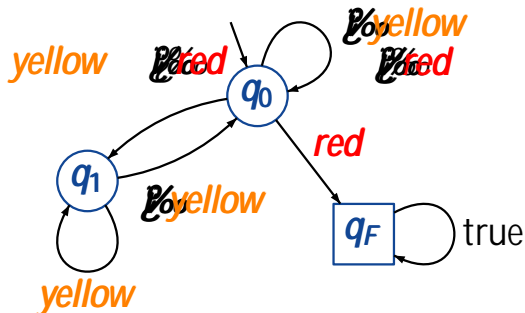
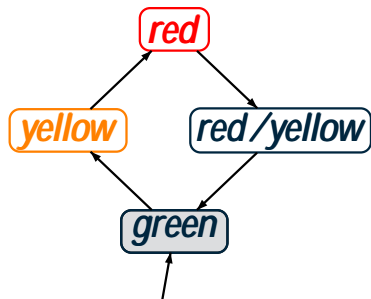
product-TS

$T \quad A$

(4 3 = 12 states)

# Example: product-TS

is2.5-26



green  $q_0$

red/yellow  $q_0$

yellow  $q_1$

red  $q_0$

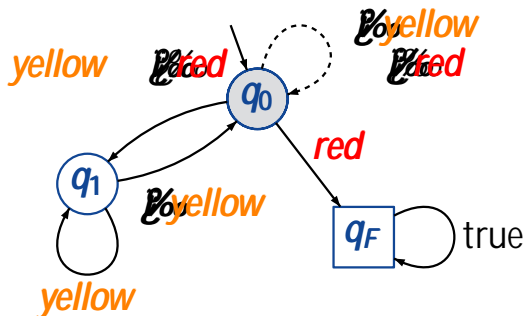
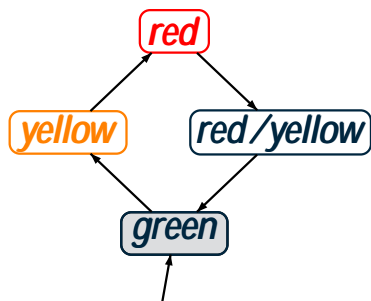
...

initial state  
green, ( $q_0$ , )

$L(\text{green}) =$

# Example: product-TS

is2.5-26



green  $q_0$

red/yellow  $q_0$

yellow  $q_1$

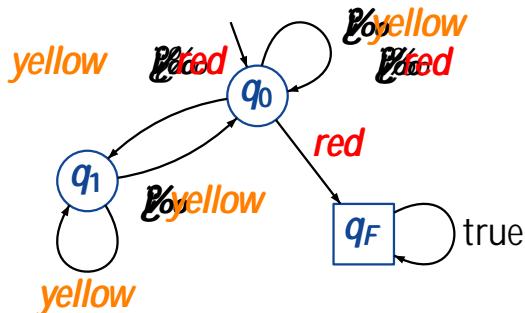
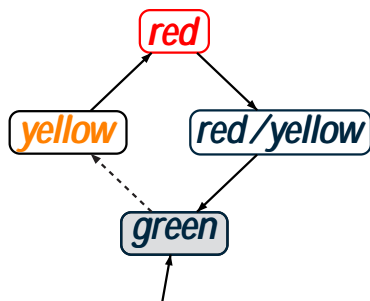
red  $q_0$

...

initial state  
green,  $(q_0, \_)$   
 $\underline{\quad} = q_0$

# Example: product-TS

is2.5-26



**green q<sub>0</sub>**

**red/yellow q<sub>0</sub>**

**yellow q<sub>1</sub>**

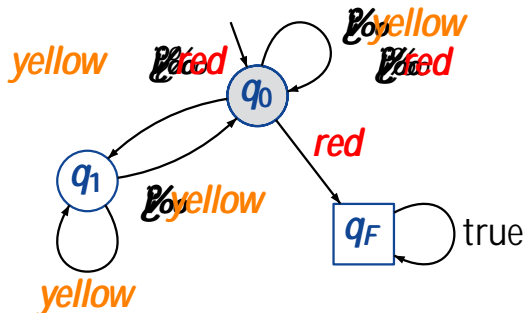
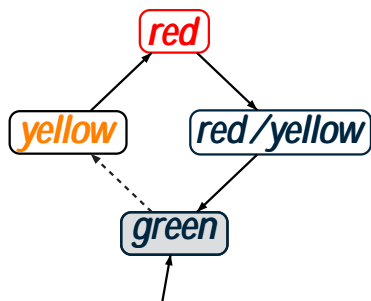
**red q<sub>0</sub>**

...

lifting the transition  
**green q<sub>0</sub>** **yellow**

# Example: product-TS

is2.5-26



green  $q_0$

red/yellow  $q_0$

yellow  $q_1$

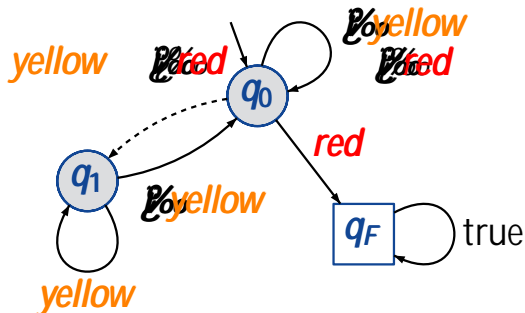
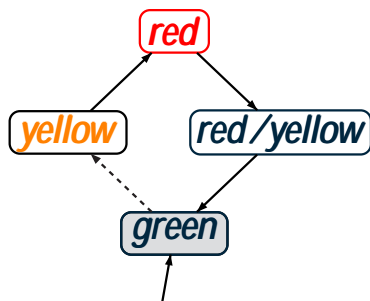
red  $q_0$

...

lifting the transition  
~~green~~  $q_0$  yellow  
 green,  $q_0$   
 yellow, ?

# Example: product-TS

is2.5-26



green  $q_0$

red/yellow  $q_0$

yellow  $q_1$

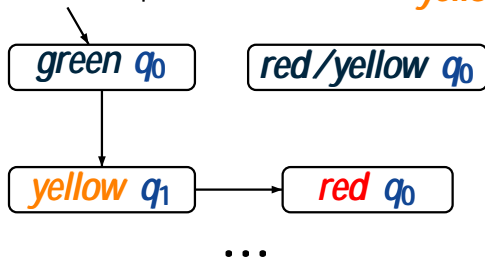
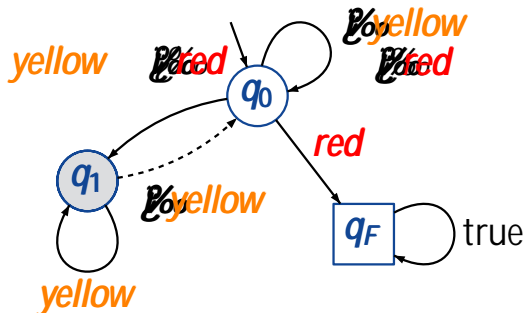
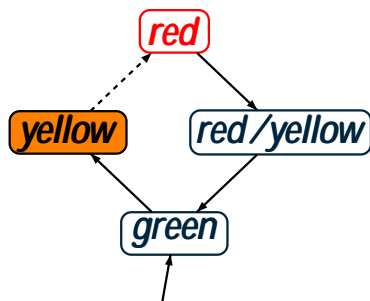
red  $q_0$

...

lifting the transition  
~~green~~ $q_0$   $\xrightarrow{\text{yellow}}$  ~~green~~ $q_0$   
 $\text{green}, q_0$   
 $\text{yellow}, (q_0, \{\text{yellow}\})$   
 $\underline{\hspace{1cm}} \quad \underline{\hspace{1cm}}$   
 $= q_1$

# Example: product-TS

is2.5-26



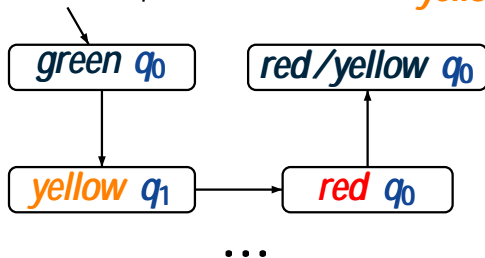
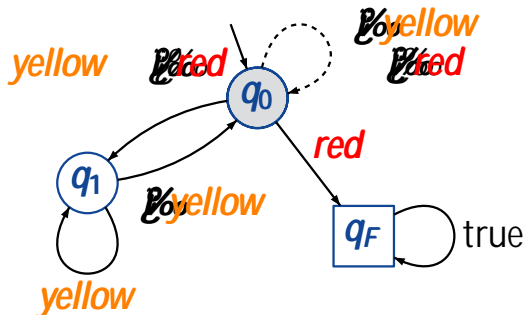
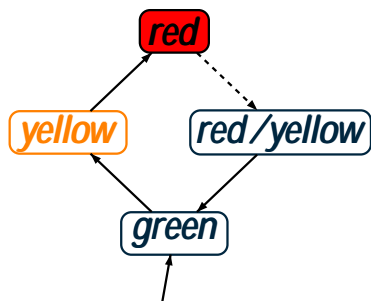
lifting the transition

~~yellow~~ ~~q0~~ ~~red~~  
yellow, q1

~~red, (q1, {red})~~  
= q0

# Example: product-TS

is2.5-26



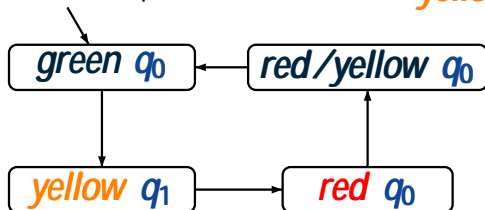
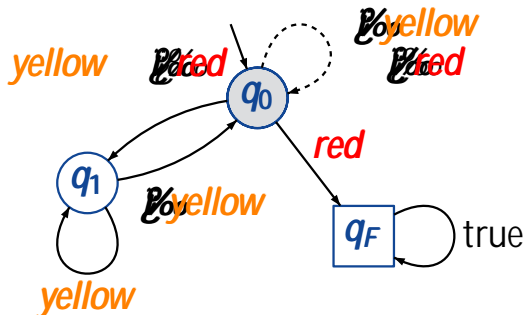
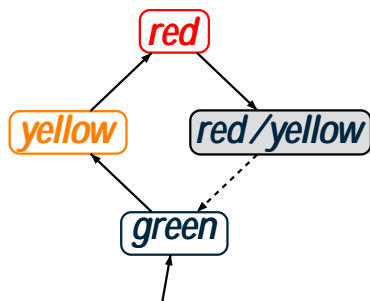
lifting the transition  
~~red~~  $q_0$  ~~red/yellow~~  
 $red, q_0$

$$red/yellow, (q_0, \underline{\quad}) \\ = q_0$$



# Example: product-TS

is2.5-26

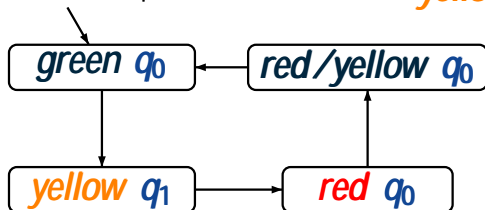
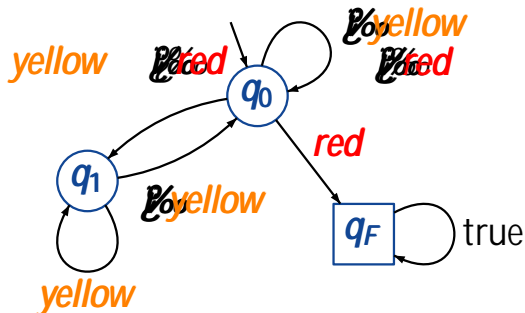
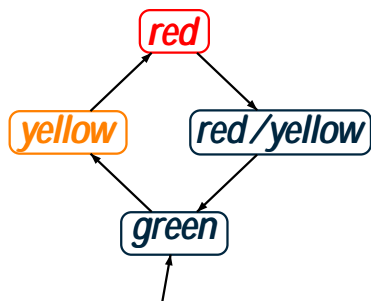


lifting the transition  
 $\text{red/yellow } q_0 \rightarrow \text{green}$   
 $\text{red/yellow}, q_0$

$$\text{green}, (q_0, \_) \\ \hline = q_0$$

# Example: product-TS

is2.5-26



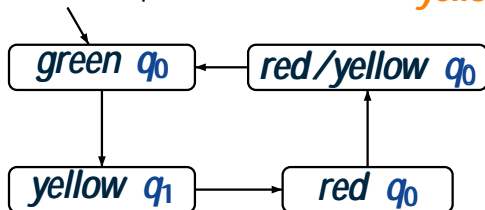
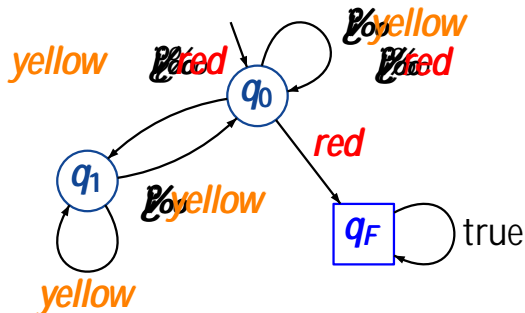
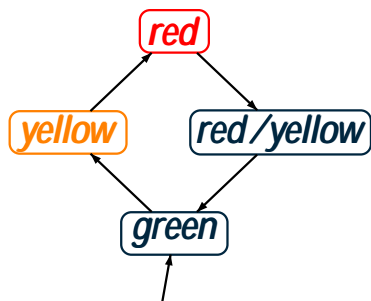
product-TS

*T* *A*

4 3 = 12 states, but  
just 4 reachable states

# Example: product-TS

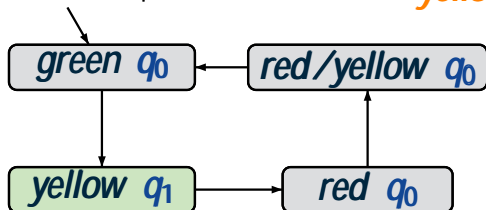
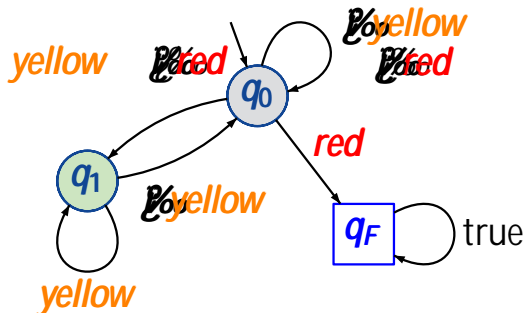
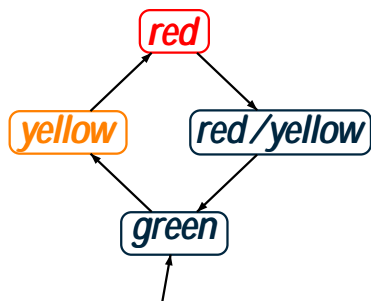
is2.5-26



set of propositions  
 $AP = \{q_0, q_1, q_F\}$

# Example: product-TS

is2.5-26



set of propositions  
 $AP = \{q_0, q_1, q_F\}$

invariant condition  ~~$q_F$~~  holds  
 for all reachable states

definition of the product of

two transition systems  $T = (S, Act, \rightarrow, s_0, AP, L)$

and an NFA  $A = (Q, 2^{AP}, \rightarrow, Q_0, F)$

then the product  $T \times A = (S \times Q, Act, \rightarrow, \dots)$  is a TS

definition of the product of

of a transition system  $T = (S, Act, \rightarrow, s_0, AP, L)$

without terminal states

of an NFA  $A = (Q, 2^{AP}, \rightarrow, Q_0, F)$

then the product  $T \times A = (S \times Q, Act, \rightarrow, \dots)$  is a TS

definition of the product of

two transition systems  $T = (S, Act, \rightarrow, s_0, AP, L)$

without terminal states

and an NFA  $A = (Q, 2^{AP}, \rightarrow, Q_0, F)$

then the product  $T \times A = (S \times Q, Act, \rightarrow, \dots)$  is a TS

without terminal states

definition of the product of

two transition systems  $T = (S, Act, \rightarrow, s_0, AP, L)$

without terminal states

and an NFA  $A = (Q, 2^{AP}, \rightarrow, Q_0, F)$

then the product  $T \times A = (S \times Q, Act, \rightarrow, \dots)$  is a TS

without terminal states

assumptions on the NFA  $A$ :



definition of the product of

an transition system  $T = (S, Act, \rightarrow, s_0, AP, L)$

without terminal states

an NFA  $A = (Q, 2^{AP}, \rightarrow, Q_0, F)$

then the product  $T \times A = (S \times Q, Act, \rightarrow, \dots)$  is a TS

without terminal states

assumptions on the NFA  $A$ :

$A$  is non-blocking, i.e.,

$$Q_0 = \{q \in Q \mid A \models 2^{AP}. (q, A) = \dots\}$$

definition of the product of

two transition systems  $T = (S, Act, \rightarrow, s_0, AP, L)$

without terminal states

and an NFA  $A = (Q, 2^{AP}, \rightarrow, Q_0, F)$

then the product  $T \times A = (S \times Q, Act, \rightarrow, \dots)$  is a TS

without terminal states

assumptions on the NFA  $A$ :

$A$  is non-blocking, i.e.,

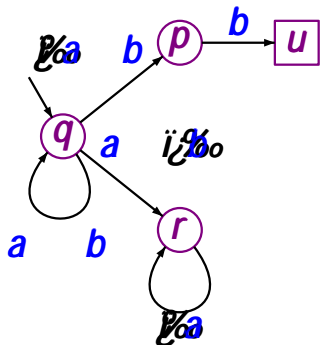
$Q_0 = \{q \in Q \mid A \models 2^{AP}\}$ .  $(q, A) =$

no initial state of  $A$  is final, i.e.,  $Q_0 \cap F = \emptyset$

# Non-blocking NFA

is2.5-23

NFA **A**

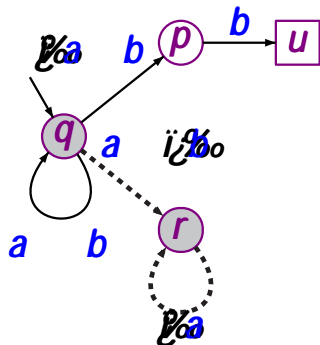


alphabet =  $2^{AP}$  where  $AP = \{a, b\}$

# Non-blocking NFA

is2.5-23

NFA  $\mathcal{A}$



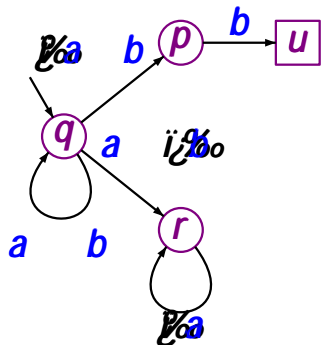
blocks for input  
 $\{a\}$   $\{a\}$

alphabet  $= 2^{AP}$  where  $AP = \{a, b\}$

# Non-blocking NFA

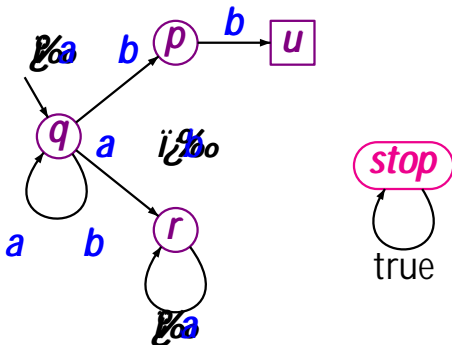
is2.5-23

NFA **A**



blocks for input  
 $\{a\}$   $\{a\}$

equivalent NFA **A**

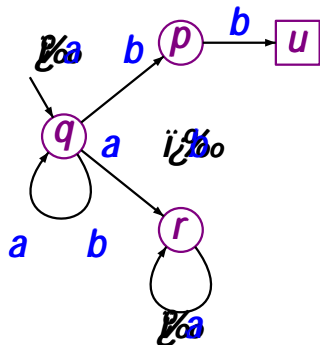


add a trap state **stop**

# Non-blocking NFA

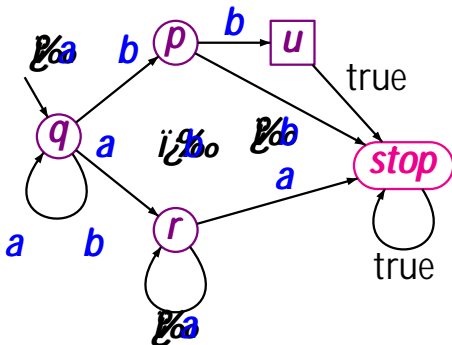
is2.5-23

NFA **A**



blocks for input  
**{a}** **{a}**

equivalent NFA **A**

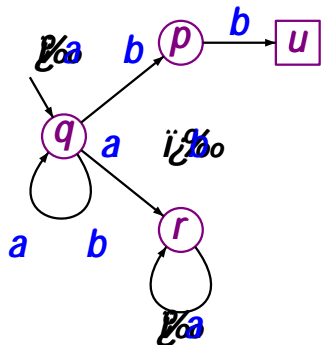


add a trap state **stop**

# Non-blocking NFA

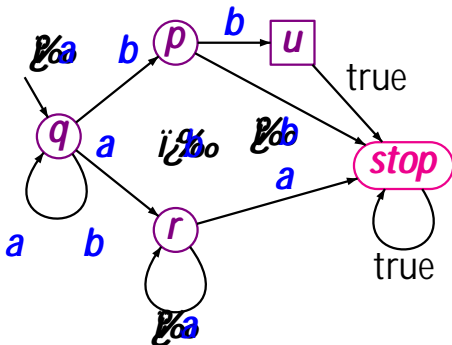
is2.5-23

NFA **A**



blocks for input  
 $\{a\}$   $\{a\}$

equivalent NFA **A**

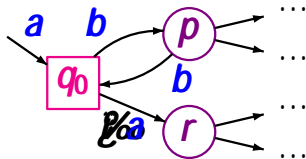


non-blocking

# NFA where no initial state is final

is2.5-24

NFA  $A$  with  $Q_0$   $F =$

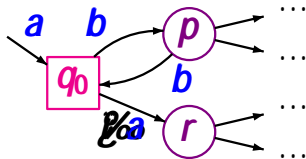




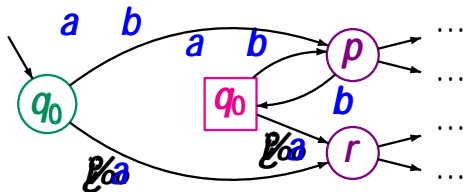
# NFA where no initial state is final

is2.5-24

NFA  $A$  with  $Q_0$   $F =$



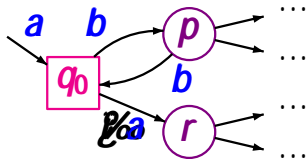
NFA  $A$  with  $Q_0$   $F =$



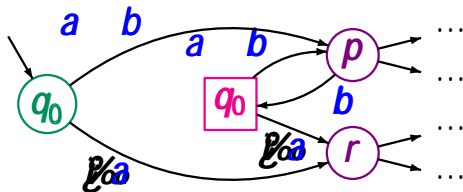
# NFA where no initial state is final

is2.5-24

NFA  $A$  with  $Q_0$   $F =$



NFA  $A$  with  $Q_0$   $F =$

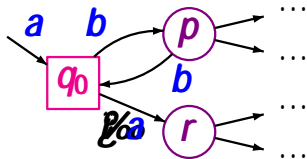


$$L(A) = L(A) \setminus \{ \}$$

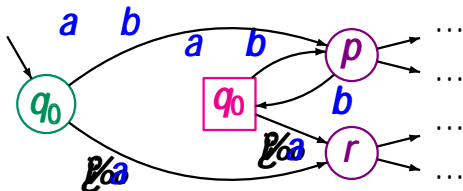
# NFA where no initial state is final

is2.5-24

NFA  $A$  with  $Q_0$   $F =$



NFA  $A$  with  $Q_0$   $F =$



$$L(A) = L(A) \setminus \{ \}$$

note: if  $A$  is an NFA for the  
bad prefixes of a safety property then

$$L(A) = \text{BadPref}$$

# Model checking regular safety properties

is2.5-26a

... via a reduction to invariant checking .....

# Model checking regular safety properties

is2.5-26a

Let  $T = (S, Act, \rightarrow, S_0, AP, L)$  be a transition system

$A = (Q, 2^{AP}, \rightarrow, Q_0, F)$  be an NFA  
for the bad prefixes of a regular safety property  $E$

# Model checking regular safety properties

is2.5-26a

Let  $T = (S, Act, \rightarrow, S_0, AP, L)$  be a transition system  
(without terminal states)

$A = (Q, 2^{AP}, \rightarrow, Q_0, F)$  be an NFA  
for the bad prefixes of a regular safety property  $E$

# Model checking regular safety properties

is2.5-26a

Let  $T = (S, Act, \rightarrow, S_0, AP, L)$  be a transition system  
(without terminal states)

$A = (Q, 2^{AP}, \rightarrow, Q_0, F)$  be an NFA  
for the bad prefixes of a regular safety property  $E$   
(non-blocking and  $Q_0 \cap F = \emptyset$ )



Let  $T = (S, Act, \rightarrow, S_0, AP, L)$  be a transition system  
(without terminal states)

$A = (Q, 2^{AP}, \rightarrow, Q_0, F)$  be an NFA  
for the bad prefixes of a regular safety property  $E$   
(non-blocking and  $Q_0 \cap F = \emptyset$ )

The following statements are equivalent:

- (1)  $T \models E$
- (2)  $Traces_A(T) \cap L(A) = \emptyset$

# Model checking regular safety properties

is2.5-26a

Let  $T = (S, Act, \rightarrow, S_0, AP, L)$  be a transition system  
(without terminal states)

$A = (Q, 2^{AP}, \rightarrow, Q_0, F)$  be an NFA  
for the bad prefixes of a regular safety property  $E$   
(non-blocking and  $Q_0 \cap F = \emptyset$ )

The following statements are equivalent:

- (1)  $T \models E$
- (2)  $Traces_A(T) \cap L(A) = \emptyset$
- (3)  $T \models A$  invariant always  $\neg F$

# Model checking regular safety properties

is2.5-26a

Let  $T = (S, Act, \rightarrow, S_0, AP, L)$  be a transition system  
(without terminal states)

$A = (Q, 2^{AP}, \rightarrow, Q_0, F)$  be an NFA  
for the bad prefixes of a regular safety property  $E$   
(non-blocking and  $Q_0 \cap F = \emptyset$ )

The following statements are equivalent:

- (1)  $T \models E$
- (2)  $Traces(T) \cap L(A) = \emptyset$
- (3)  $T \models A$  invariant always  $\neg F$

where  $\neg F$  denotes  $\bigcup_{q \in F} q$

# Product transition system

is2.5-25a

$T = (S, Act, \rightarrow, S_0, AP, L)$  transition system

$A = (Q, 2^{AP}, \rightarrow, Q_0, F)$  NFA

product-TS  $T \times A \stackrel{\text{def}}{=} (S \times Q, Act, \rightarrow, S_0, AP, L)$

$$\frac{s \times q \quad s \quad q \quad (q, L(s))}{s, q \times s, q}$$

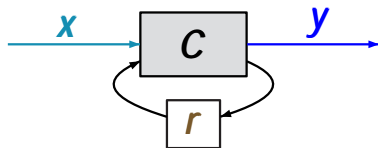
initial states:  $S_0 = s_0, q : s_0 \quad S_0, q \quad Q_0, L(s_0)$

set of atomic propositions:  $AP = Q$

labeling function:  $L(s, q) = \{q\}$

# Example: sequential circuit

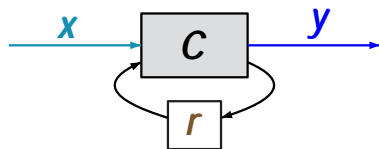
is2.5-27



$$y = r = x \quad r$$

# Example: sequential circuit

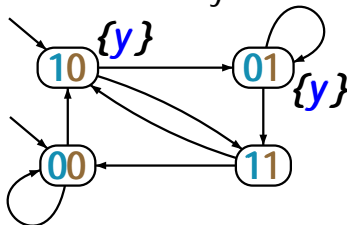
is2.5-27



$$y = r = x \quad r$$

initially  $r = 0$

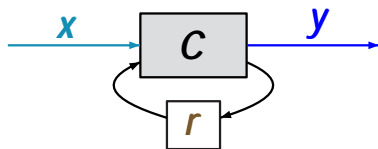
transition system  $T$



over  $AP = \{y\}$

# Example: sequential circuit

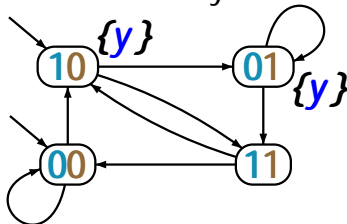
is2.5-27



$$y = r = x \quad r$$

initially  $r = 0$

transition system  $T$



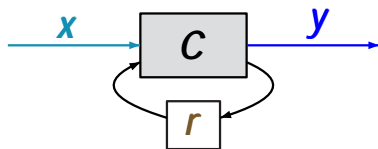
over  $AP = \{y\}$

safety property  $E$

*The circuit will never  
output two ones  
after each other*

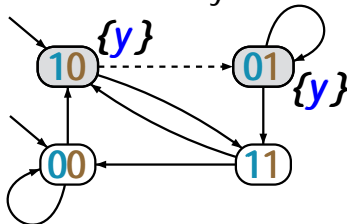
# Example: sequential circuit

is2.5-27



$y = r = x$   
initially  $r = 0$

transition system  $T$



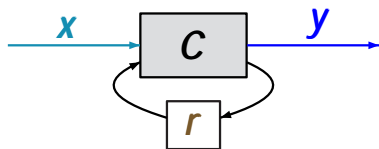
$T \not\models E$

safety property  $E$   
*The circuit will never  
output two ones  
after each other*

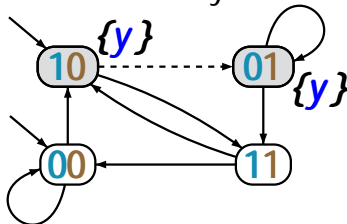


# Example: sequential circuit

is2.5-27



transition system  $T$



$$y = r = x \quad r$$

initially  $r = 0$

$$T \not\models E$$

error indication, e.g.,

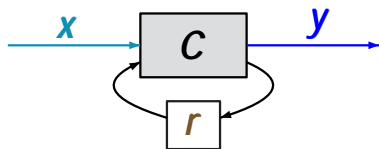
10 01

safety property  $E$

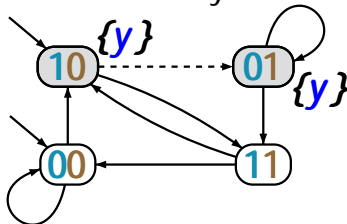
*The circuit will never  
output two ones  
after each other*

# Example: sequential circuit

is2.5-27



transition system  $T$



$$y = r = x \quad r$$

initially  $r = 0$

$$T \not\models E$$

error indication, e.g.,

10 01

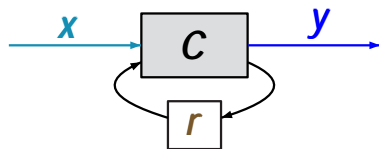
bad prefix  $\{y\}\{y\}$

safety property  $E$

*The circuit will never  
output two ones  
after each other*

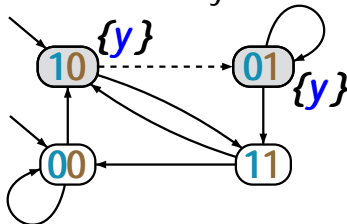
# Example: sequential circuit

is2.5-27



$y = r = x$  initially  $r = 0$

transition system  $T$



$T \not\models E$

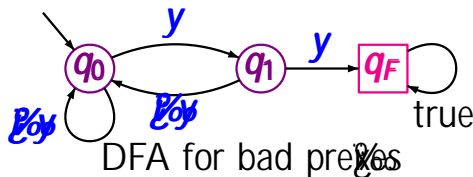
error indication, e.g.,

10 01

bad prefixes  $\{y\} \{y\}$

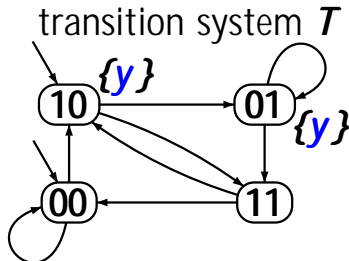
safety property  $E$

*The circuit will never output two ones after each other*

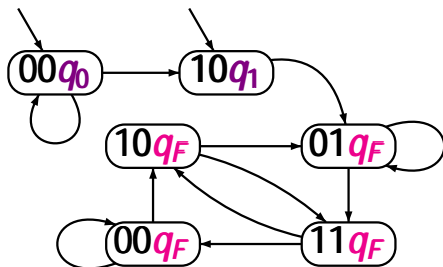
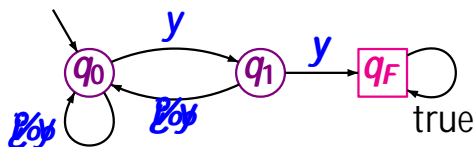


# Example: product-TS

is2.5-28



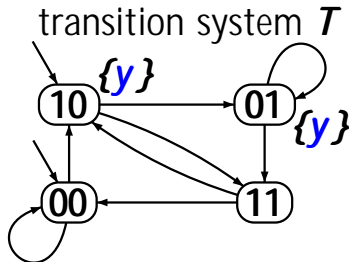
safety property  $E$   
... never two ones in a row ...



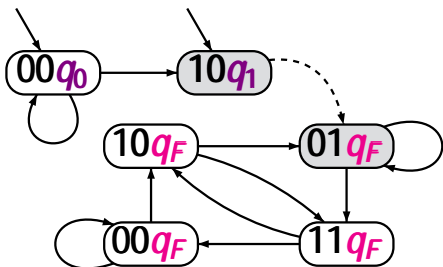
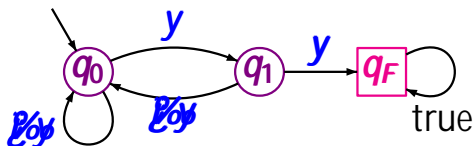
product-TS  $T \quad A$

# Example: product-TS

is2.5-28



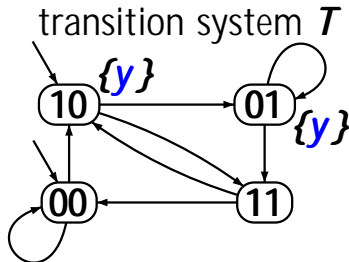
safety property  $E$   
*... never two ones in a row ...*



$T \times A \models \neg \text{never } q_F$

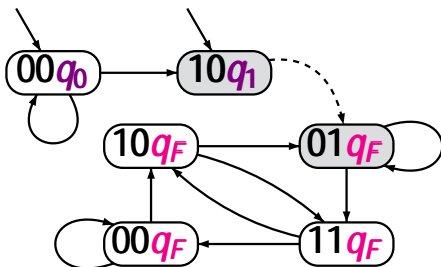
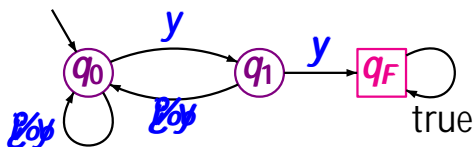
# Example: product-TS

is2.5-28



safety property  $E$

... never two ones in a row ...

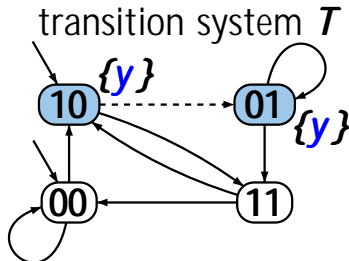


error indication for  
 $T \times A \models \neg \text{never } q_F$

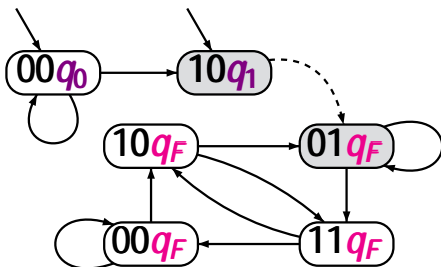
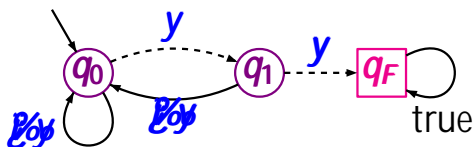
$10q_1$      $01q_F$

# Example: product-TS

is2.5-28

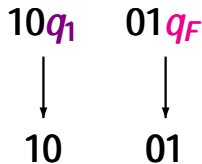


safety property  $E$   
*... never two ones in a row ...*



error indication for  $T \models E$

error indication for  
 $T \models E$  : never  $q_F$







# Model checking regular safety properties

is2.5-29

*input:*     $\varphi \in \text{TS}$   
          NFA  $A$  for the bad prefixes of  $E$

*output:*     $\varphi \in \text{TS}$   
          Yes  $\varphi \in E$   
          otherwise No

# Model checking regular safety properties

is2.5-29

input:  $\langle T, TS \rangle$   
NFA  $A$  for the bad prefixes of  $E$

output:  $\langle T, A \rangle$   
otherwise  $\langle T, \emptyset \rangle$

construct product transition system  $T \times A$

check whether  $T \times A \models \neg \phi$

where  $F$  = set of bad states in  $A$

# Model checking regular safety properties

is2.5-29

input:  $\langle T, TS \rangle$   
 NFA  $A$  for the bad prefixes of  $E$

output:  $\langle T, A \rangle$   
 otherwise  $\langle \emptyset, \emptyset \rangle$

construct product transition system  $T \times A$

check whether  $T \times A \models \neg \phi$

if so, then return  $\langle \emptyset, \emptyset \rangle$

if not, then return  $\langle T, A \rangle$

where  $F$  = set of bad states in  $A$

# Model checking regular safety properties

is2.5-29

input:  $\langle T, TS \rangle$   
NFA  $A$  for the bad prefixes of  $E$

output:  $\langle T, A \rangle$   
otherwise  $\perp$  + error indication

construct product transition system  $T \times A$

check whether  $T \times A \models \neg \phi$

if so, then return  $\perp$

if not, then return  $\langle T, A \rangle$

and an error indication

where  $F$  = set of bad states in  $A$

construct product transition system  $T \quad A$

IF  $T \quad A \models \varphi$  always  $\%F\%$

THEN return  $\%Yes\%$

ELSE

FI

construct product transition system  $T \quad A$

IF  $T \quad A \models \text{always } \text{Prop}$

THEN return Yes

ELSE compute a counterexample for  $T \quad A$  and  
the invariant  $\text{always } \text{Prop}$

FI

construct product transition system  $T \quad A$

IF  $T \quad A \models \text{always } \textcolor{red}{P} \textcolor{red}{F}$

THEN return Yes

ELSE compute a counterexample for  $T \quad A$  and  
the invariant  $\text{always } \textcolor{red}{P} \textcolor{red}{F}$

i.e., an initial path fragment in the product

$s_0, p_0 \quad s_1, p_1 \quad \dots \quad s_n, p_n$  where  $p_n \textcolor{red}{F}$

FI

construct product transition system  $T \quad A$

IF  $T \quad A \models \text{always } \neg F$

THEN return Yes

ELSE compute a counterexample for  $T \quad A$  and  
the invariant  $\text{always } \neg F$

i.e., an initial path fragment in the product

$s_0, p_0 \quad s_1, p_1 \quad \dots \quad s_n, p_n$  where  $p_n \quad F$

return No and  $s_0 \quad s_1 \quad \dots \quad s_n$

FI



construct product transition system  $T \quad A$

IF  $T \quad A \models \text{always } \text{No}$

THEN return Yes

ELSE compute a counterexample for  $T \quad A$  and  
the invariant  $\text{always } \text{No}$

i.e., an initial path fragment in the product

$s_0, p_0 \quad s_1, p_1 \quad \dots \quad s_n, p_n$  where  $p_n \quad F$

return No and  $s_0 \quad s_1 \quad \dots \quad s_n$

FI

time complexity:  $O \quad \text{size}(T) \quad \text{size}(A)$

If  $T$  is a finite transition system then  
 $Traces(T)$  is regular.

If  $T$  is a finite transition system then  
 $Traces(T)$  is regular.

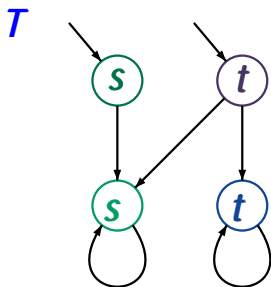
correct.

If  $T$  is a finite transition system then  
 $Traces(T)$  is regular.

correct.  $T$  can be transformed into an **NFA**.

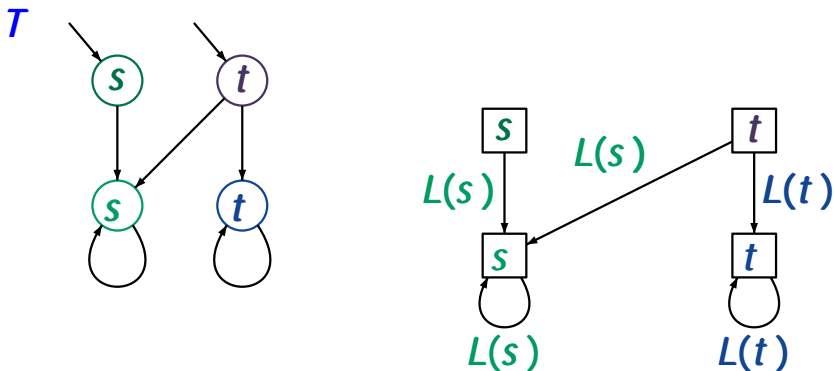
If  $T$  is a ~~finite~~ transition system then  
 $Traces(T)$  is regular.

correct.  $T$  can be transformed into an **NFA**.



If  $T$  is a ~~finite~~ transition system then  
 $Traces(T)$  is ~~regular~~.

correct.  $T$  can be transformed into an **NFA**.

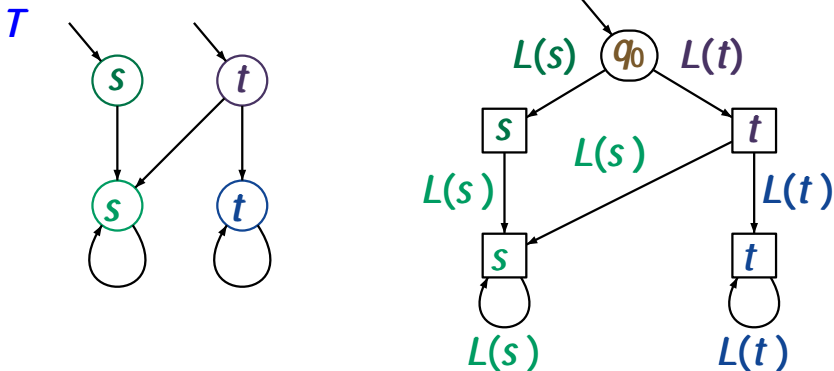


# Correct or wrong?

is2.5-35

If  $T$  is a finite transition system then  
 $Traces(T)$  is regular.

correct.  $T$  can be transformed into an NFA.



# Correct or wrong?

is2.5-35

If  $T$  is a ~~finite~~ transition system then  
 $Traces_{\Delta}(T)$  is regular.

correct.  $T$  can be transformed into an NFA.

