11. Automata-based Model Checking

Computer-Aided Verification

Dave Parker

University of Birmingham

2017/18

Mid-term questionnaires

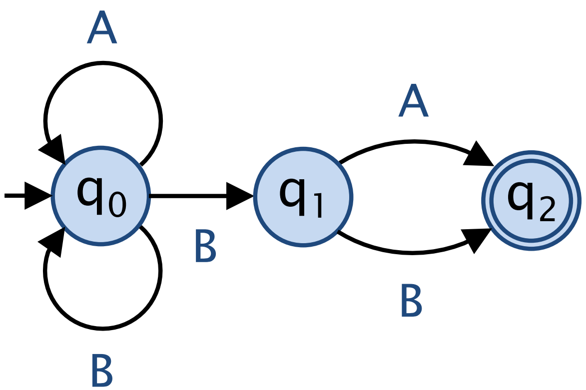
- Generally, all happy
  - lectures, tutorials, feedback, etc.
- Comments/suggestions
  - don't split tutorial sessions
  - more unassessed exercises
  - assignment timing for extended version
  - additional (practical) context for material

Overview

- Regular safety properties
  - nondeterministic finite automata (NFAs)
- Model checking regular safety properties
  - LTS-NFA products
- See [BK08] Sections 4–4.2

Recap: Regular languages

- A set of finite words $\mathcal{L} \subseteq \Sigma^*$ is a regular language…
  - iff $\mathcal{L} = \mathcal{L}(E)$ for some regular expression E
  - iff $\mathcal{L} = \mathcal{L}(\mathcal{A})$ for some finite automaton $\mathcal{A}$



(A+B)*B(A+B)

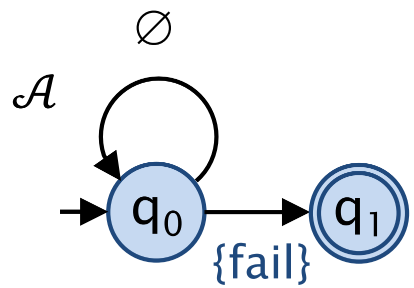
(i.e. penultimate symbol is B)

Languages/automata as properties

- Recall:
  - a linear-time property is a set of infinite words $P \subseteq (2^{AP})^\omega$
  - (we assume non-terminating systems with infinite paths/traces)

- But we could represent a set of finite traces/words
  - as a regular language over alphabet 2^{AP}
  - or an NFA over alphabet 2^{AP}

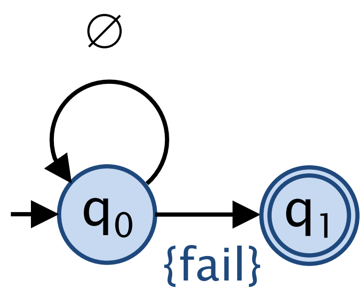
- Simple example
  - finite traces where a failure eventually occurs
  - $AP = \{fail\}$
  - $2^{AP} = \{\varnothing, \{fail\}\}$
  - $\mathcal{L}(\mathcal{A}) = \{ \{fail\}, \varnothing \{fail\}, \varnothing \varnothing \{fail\}, \varnothing \varnothing \varnothing \{fail\}, \ldots \}$



Regular safety properties

- A regular safety property is
  - a safety property for which the set of “bad prefixes” (finite violations) forms a regular language
- Example:
  - "a failure never occurs" – $\Box\neg$fail
  - AP = {fail}
  - $2^{AP} = \{\varnothing, \{fail\}\}$

NFA $\mathcal{A}$:



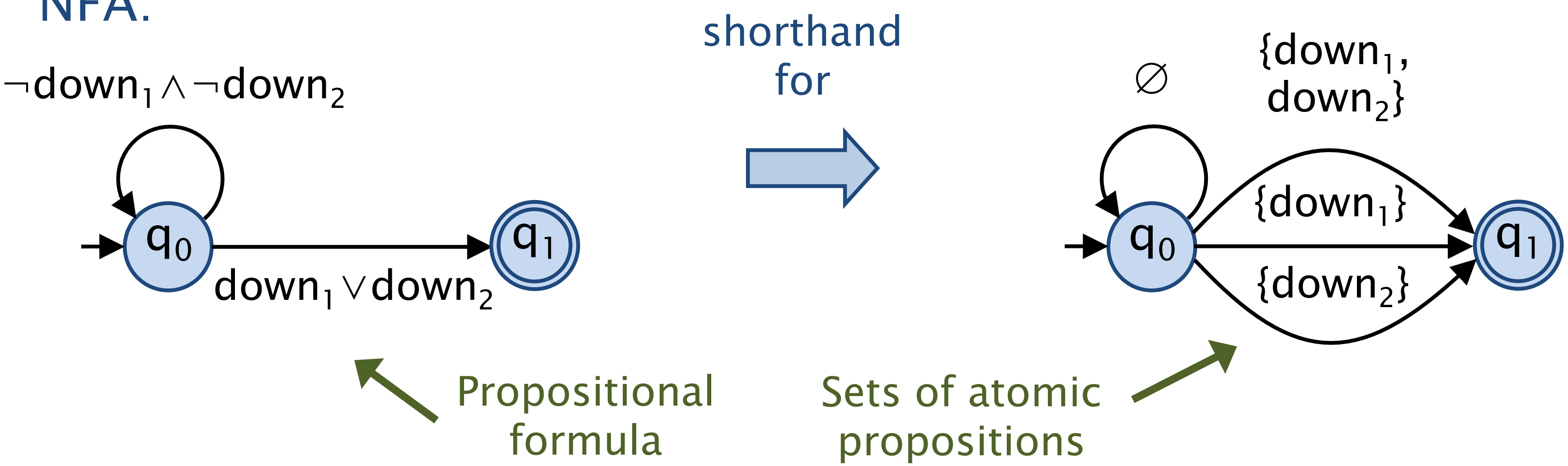
Regexp:

$\varnothing^*$ {fail}

- The bad prefixes are represented by an NFA over 2^{AP}
  - $\mathcal{L}(\mathcal{A})$ = { {fail}, ∅ {fail}, ∅ ∅ {fail}, ∅ ∅ ∅ {fail}, … }
- Note: we actually represent just minimal bad prefixes here
  - we could also represent all bad prefixes

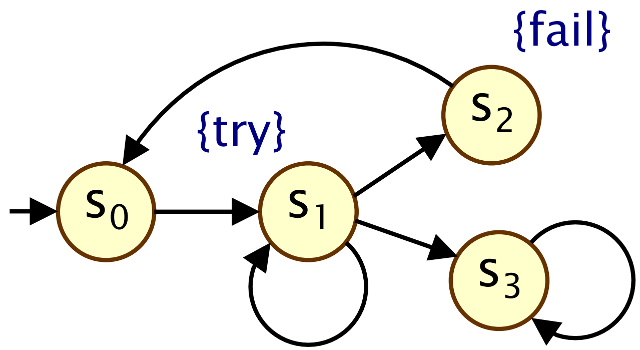
Example 2

- Regular safety property
  - "both servers are always up" – $\square(\neg down_1 \wedge \neg down_2)$
  - $AP = \{down_1, down_2\}$
  - $2^{AP} = \{\varnothing, \{down_1\}, \{down_2\}, \{down_1, down_2\}\}$
- Bad prefixes
  - any finite word ending in $\{down_1\}$, $\{down_2\}$ or $\{down_1, down_2\}$
- NFA:

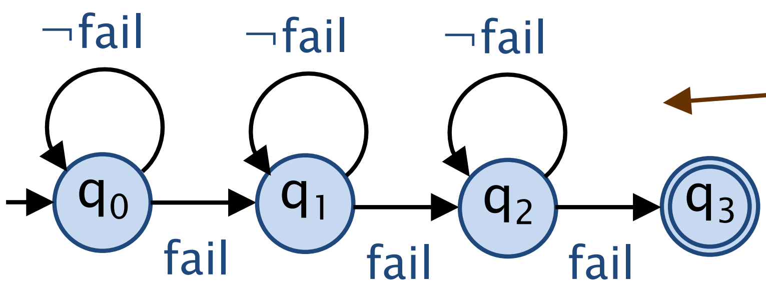


Example 3

- Regular safety property:
  - "at most 2 failures occur"
  - AP = {try, fail}
  - 2^{AP} = { ∅, {fail}, {try}, {fail,try} }



- NFA for bad prefixes:



fail denotes:
{fail}, {fail,try}
¬fail denotes:
∅, {try}

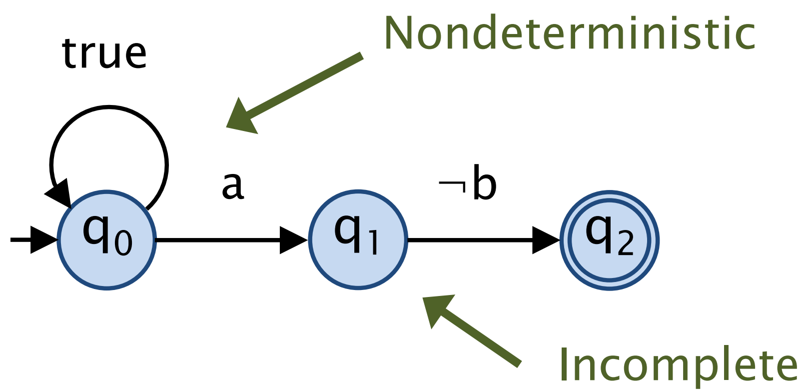
- Regular expression for bad prefixes:

(¬fail)*.fail.(¬fail)*.fail.(¬fail)*.fail

fail denotes:
({fail} + {fail,try})
¬fail denotes:
(∅ + {try})

Example 4

- NFA for regular safety property
  - AP = {a,b}

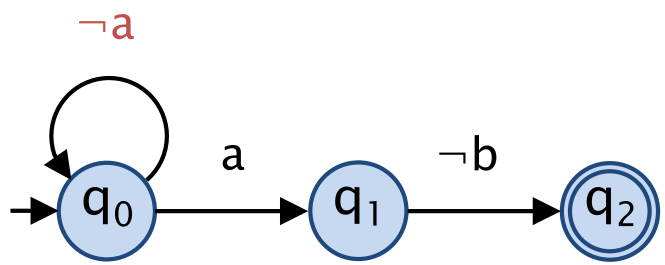


- Bad prefixes
  - any finite word where a appears and then b does not appear immediately afterwards

- Regular safety property
  - "b always immediately follows a" – $\Box(a \rightarrow \bigcirc b)$

Example 5

- NFA for regular safety property
  - AP = {a,b}



- Bad prefixes
  - any finite word where b does not appear immediately after the first a (if there is an a)

- Regular safety property
  - "b always immediately follows the first occurrence of a"

Model checking

- Given an LTS M and regular safety property P_{safe}
  - $M \models P_{safe} \iff \mathrm{Traces}(M) \subseteq P_{safe}$

    $\iff \mathrm{Traces}_{fin}(M) \cap \mathrm{BadPref}(P_{safe}) = \varnothing$

- Given also an NFA $\mathcal{A}$ representing the bad prefixes of P_{safe}
  - $M \models P_{safe} \iff \mathrm{Traces}_{fin}(M) \cap \mathcal{L}(\mathcal{A}) = \varnothing$

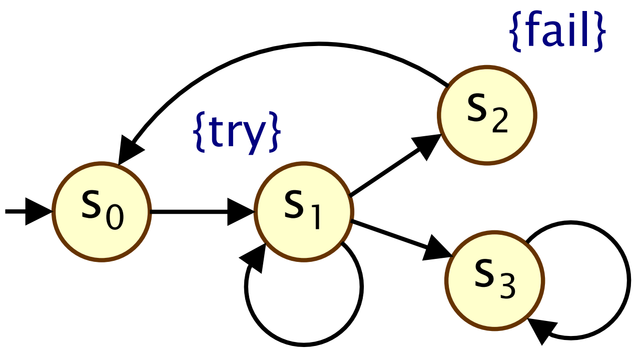
Finite traces

Bad prefixes

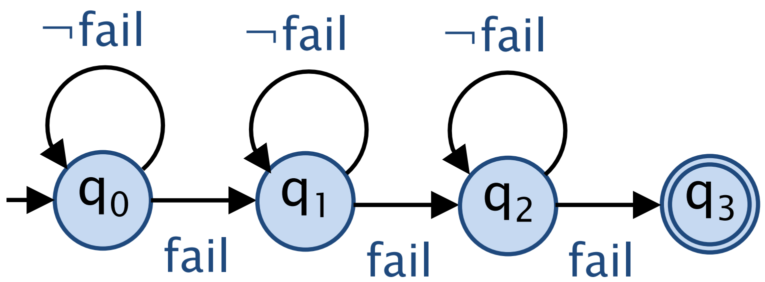
- Model checking M against regular safety property P_{safe}
  - check if any finite behaviour of M intersects with P_{safe}
  - which we do by constructing a product of M and $\mathcal{A}$

Example 3 revisited

LTS M



NFA $\mathcal{A}$



"at most 2 failures occur"

Product of an LTS and an NFA

- For an LTS M and an NFA $\mathcal{A}$
  - we construct the product LTS of M and $\mathcal{A}$, denoted $M \otimes \mathcal{A}$

- Synchronous parallel composition
  - transitions of NFA $\mathcal{A}$ synchronise with state labels of LTS M
  - allows finite traces/words that are in <u>both</u> M and $\mathcal{A}$

- Product definition (informal)
  - states are pairs (s,q) of states from M and $\mathcal{A}$
  - transitions go from (s,q) to (s',q') if
    $s \xrightarrow{\alpha} s'$ in M <u>and</u> $q \xrightarrow{L(s')} q'$ in $\mathcal{A}$
  - initial states are (s_0,q) where s_0 is some initial state of M and $q_0 \xrightarrow{L(s_0)} q$ for some initial state q_0 of $\mathcal{A}$
  - states (s,q) are labelled with "accept" if q is accepting in $\mathcal{A}$

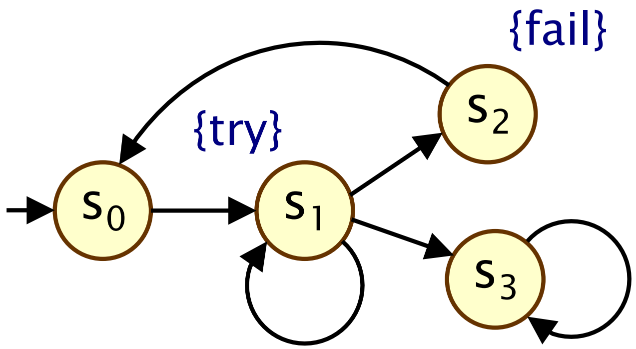
Product of an LTS and an NFA

- Formally:
  - for LTS $M = (S, Act, \rightarrow, I, AP, L)$ and NFA $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$
- The product $D \otimes A$ is:
  - the LTS $(S \times Q, Act, \rightarrow', I', \{accept\}, L')$
- where:
  - $I' = \{(s_0,q) \mid s_0 \in I$ and $q_0 \text{ -}L(s_0)\rightarrow q$ for some $q_0 \in Q_0\}$
  - $L'((s,q)) = \{ accept \}$ if $q \in F$ and $L'((s,q)) = \emptyset$ otherwise
  - $\rightarrow'$ is defined as follows:

$$\frac{s \text{ -}\alpha\rightarrow s' \wedge q \text{ -}L(s')\rightarrow q'}{(s,q) \text{ -}\alpha\rightarrow (s',q')}$$

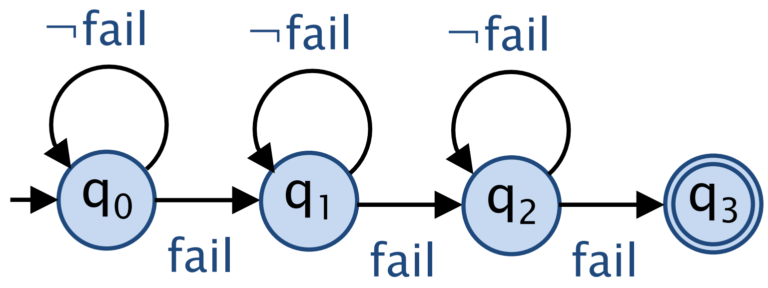
Example 3 revisited

LTS M

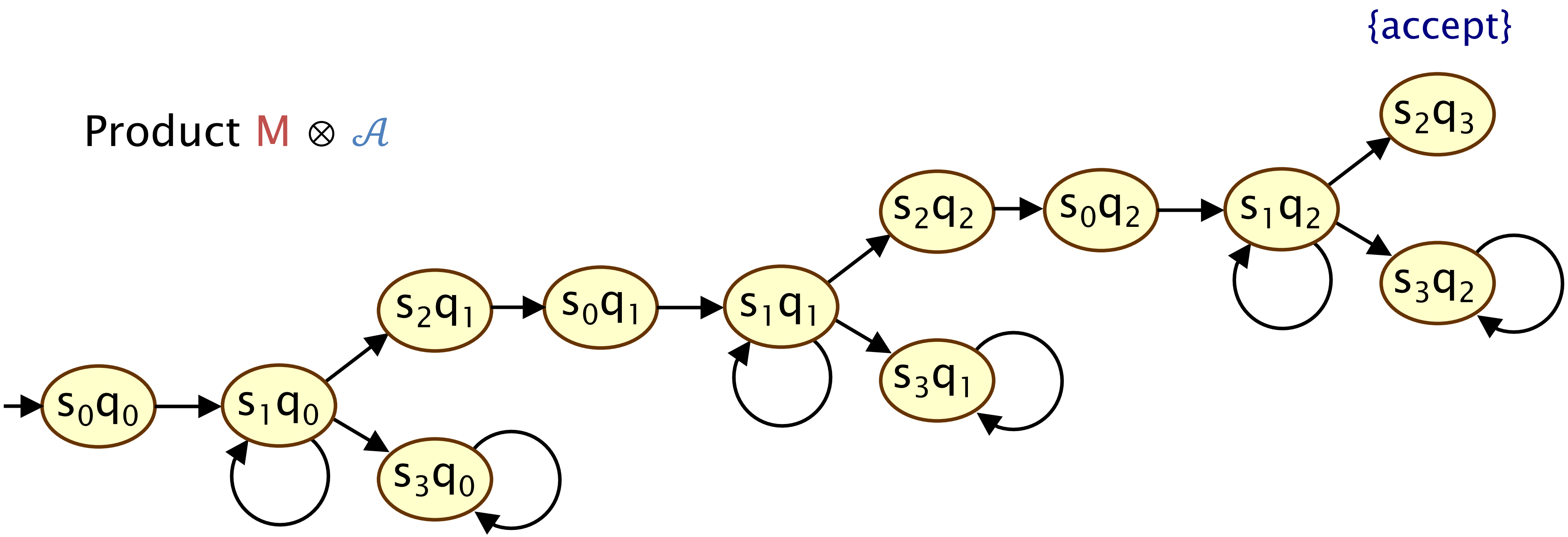


$\otimes$

NFA $\mathcal{A}$



Product $M \otimes \mathcal{A}$



Model checking

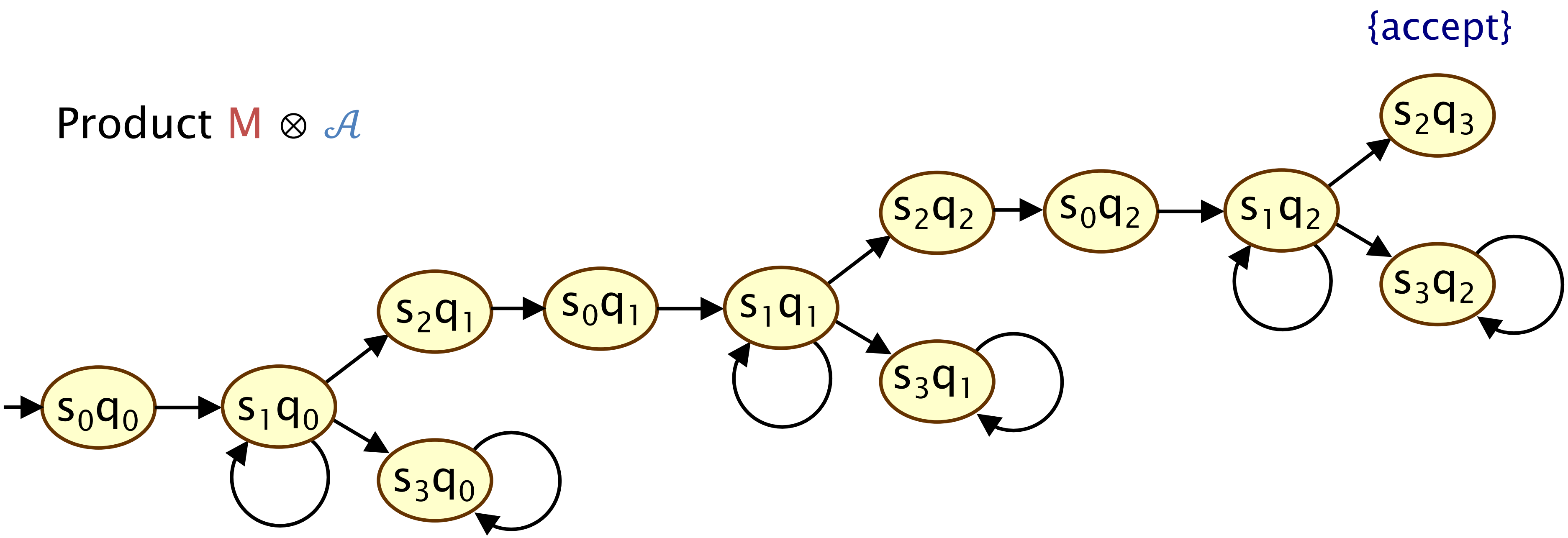
- Given an LTS M and regular safety property P_{safe}
  - and NFA $\mathcal{A}$ representing the bad prefixes of P_{safe}
  - $M \models P_{safe} \Leftrightarrow Traces_{fin}(M) \cap \mathcal{L}(\mathcal{A}) = \varnothing$

$$\boxed{M \models P_{safe} \Leftrightarrow M \otimes \mathcal{A} \models \Box\neg accept}$$

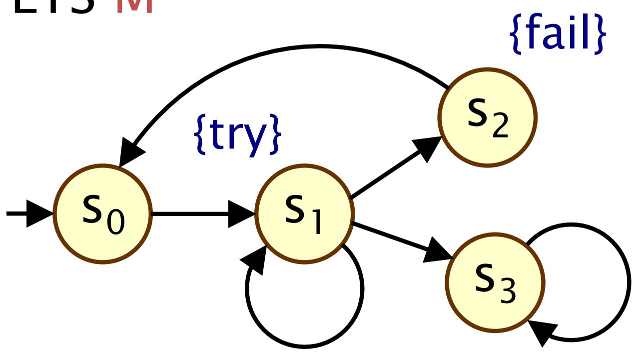
- In other words
  - $M \models P_{safe}$ iff no "accept" state is reachable in $M \otimes \mathcal{A}$
  - so model checking P_{safe} reduces to checking an invariant
- i.e., can be checked through reachability
  - see earlier discussion of invariants
  - also equivalent to checking CTL $M \otimes \mathcal{A} \models \neg\exists(\text{true U accept})$

Example 3 revisited

Product $M \otimes \mathcal{A}$



LTS M

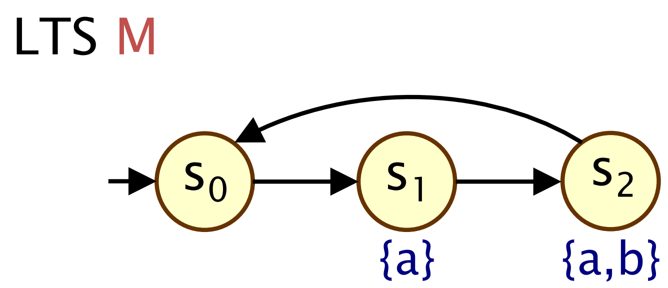


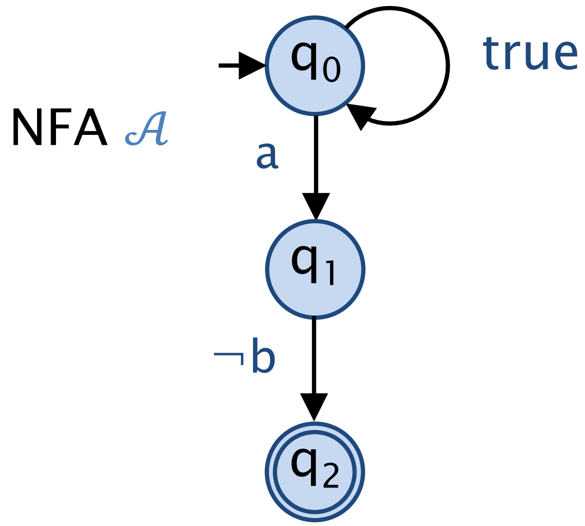
P_{safe} = "at most 2 failures occur"

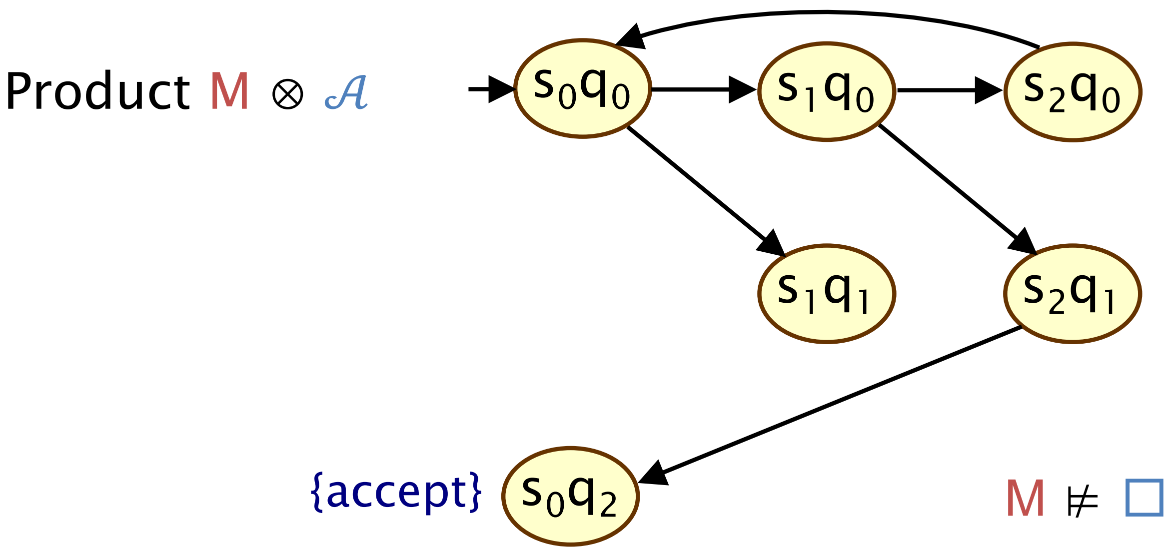
$M \not\models P_{safe}$ (since $M \otimes \mathcal{A} \not\models \Box\neg\text{accept}$)

Example

- Model check $\Box(a \rightarrow \bigcirc b)$ on LTS M





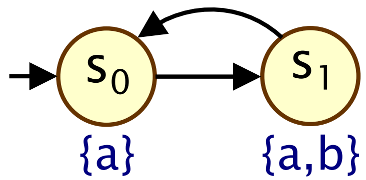


$M \not\models \Box(a \rightarrow \bigcirc b)$

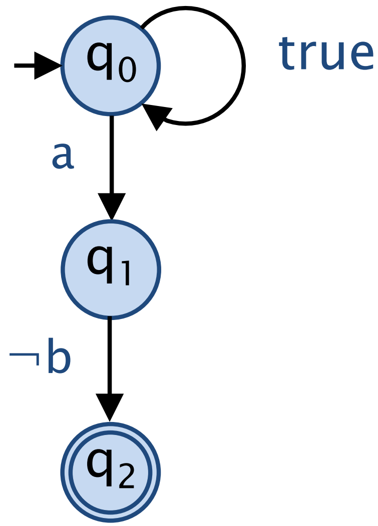
Example

- Model check $\Box(a \rightarrow \bigcirc b)$ on LTS M

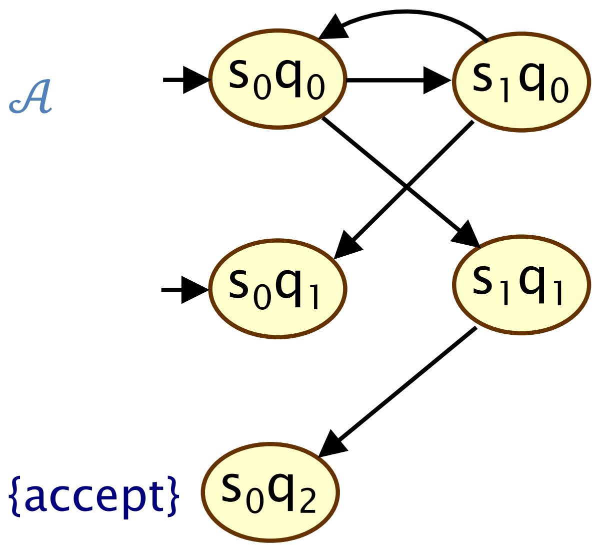
LTS M



NFA $\mathcal{A}$



Product $M \otimes \mathcal{A}$



$M \not\models \Box(a \rightarrow \bigcirc b)$

Summary

- Regular safety properties as finite automata
  - bad prefixes represented by an NFA
  - transitions labelled with propositional formulae

- Product of LTS and NFA
  - synchronise state labels of LTS with transitions of NFA
  - represents intersection of possible paths/runs

- Model checking regular safety properties
  - construct product LTS
  - reduces to checking an invariant, i.e. reachability

Next lecture

- Automata on infinite words
  - see Chapter 4.3,4.4 of [BK08]