Lecture 16
Automata-based properties

Dr. Dave Parker

Department of Computer Science
University of Oxford
Property specifications

• 1. Reachability properties, e.g. in PCTL
  – \( F a \) or \( F^{\leq t} a \) (reachability)
  – \( a U b \) or \( a U^{\leq t} b \) (until – constrained reachability)
  – \( G a \) (invariance) (dual of reachability)
  – probability computation: graph analysis + solution of linear equation system (or linear optimisation problem)

• 2. Long-run properties, e.g. in LTL
  – \( GF a \) (repeated reachability)
  – \( FG a \) (persistence)
  – probability computation: BSCCs + probabilistic reachability

• This lecture: more expressive class for type 1
Overview

- Nondeterministic finite automata (NFA)
- Regular expressions and regular languages
- Deterministic finite automata (DFA)
- Regular safety properties
- DFAs and DTMCs
Some notation

- Let $\Sigma$ be a finite alphabet

- A (finite or infinite) word $w$ over $\Sigma$ is
  - a sequence of $\alpha_1 \alpha_2 \ldots$ where $\alpha_i \in \Sigma$ for all $i$

- A prefix $w'$ of word $w = \alpha_1 \alpha_2 \ldots$ is
  - a finite word $\beta_1 \beta_2 \ldots \beta_n$ with $\beta_i = \alpha_i$ for all $1 \leq i \leq n$

- $\Sigma^*$ denotes the set of finite words over $\Sigma$

- $\Sigma^\omega$ denotes the set of infinite words over $\Sigma$
Finite automata

• A nondeterministic finite automaton (NFA) is...

  - a tuple $A = (Q, \Sigma, \delta, Q_0, F)$ where:

    - $Q$ is a finite set of states
    - $\Sigma$ is an alphabet
    - $\delta : Q \times \Sigma \rightarrow 2^Q$ is a transition function
    - $Q_0 \subseteq Q$ is a set of initial states
    - $F \subseteq Q$ is a set of “accept” states
Language of an NFA

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

• A run of $A$ on a finite word $w = \alpha_1 \alpha_2 \ldots \alpha_n$ is:
  – a sequence of automata states $q_0 q_1 \ldots q_n$ such that:
    – $q_0 \in Q_0$ and $q_{i+1} \in \delta(q_i, \alpha_{i+1})$ for all $0 \leq i < n$

• An accepting run is a run with $q_n \in F$

• Word $w$ is accepted by $A$ iff:
  – there exists an accepting run of $A$ on $w$

• The language of $A$, denoted $L(A)$ is:
  – the set of all words accepted by $A$

• Automata $A$ and $A'$ are equivalent if $L(A) = L(A')$
Example – NFA

\( q_0 \) \( \xrightarrow{\alpha} q_1 \) \( \xrightarrow{\beta} q_2 \)

\( q_1 \) \( \xrightarrow{\alpha} q_0 \) \( \xrightarrow{\beta} q_2 \)

\( q_2 \) \( \xrightarrow{\beta} q_0 \) \( \xrightarrow{\beta} q_1 \)
Regular expressions

- Regular expressions $E$ over a finite alphabet $\Sigma$
  - are given by the following grammar:
    
    $E ::= \emptyset \mid \varepsilon \mid \alpha \mid E + E \mid E.E \mid E^*$
    
    - where $\alpha \in \Sigma$

- Language $L(E) \subseteq \Sigma^*$ of a regular expression:
  
  - $L(\emptyset) = \emptyset$ (empty language)
  - $L(\varepsilon) = \{ \varepsilon \}$ (empty word)
  - $L(\alpha) = \{ \alpha \}$ (symbol)
  - $L(E_1 + E_2) = L(E_1) \cup L(E_2)$ (union)
  - $L(E_1.E_2) = \{ w_1.w_2 \mid w_1 \in L(E_1) \text{ and } w_2 \in L(E_2) \}$ (concatenation)
  - $L(E^*) = \{ w^i \mid w \in L(E) \text{ and } i \in \mathbb{N} \}$ (finite repetition)
Regular languages

• A set of finite words \( L \) is a regular language...

\[ \text{iff } L = L(E) \text{ for some regular expression } E \]

\[ \text{iff } L = L(A) \text{ for some finite automaton } A \]

\[ (\alpha + \beta)^* \beta (\alpha + \beta) \]

(i.e. penultimate symbol is \( \beta \))
Operations on NFA

- Can construct NFA from regular expression inductively
  - includes addition (and then removal) of $\varepsilon$-transitions

- Can construct the intersection of two NFA
  - build (synchronised) product automaton
  - cross product of $A_1 \otimes A_2$ accepts $L(A_1) \cap L(A_2)$
Deterministic finite automata

• A finite automaton is deterministic if:
  – \(|Q_0| = 1\)
  – \(|\delta(q, \alpha)| \leq 1\) for all \(q \in Q\) and \(\alpha \in \Sigma\)
  – i.e. one initial state and no nondeterministic successors

• A deterministic finite automaton (DFA) is total if:
  – \(|\delta(q, \alpha)| = 1\) for all \(q \in Q\) and \(\alpha \in \Sigma\)
  – i.e. unique successor states

• A total DFA
  – can always be constructed from a DFA
  – has a unique run for any word \(w \in \Sigma^*\)
Determinisation: NFA $\rightarrow$ DFA

- Determinisation of an NFA $A = (Q, \Sigma, \delta, Q_0, F)$
  - i.e. removal of choice in each automata state

- Equivalent DFA is $A_{\text{det}} = (2^Q, \Sigma, \delta_{\text{det}}, q_0, F_{\text{det}})$ where:
  - $\delta_{\text{det}}(Q', \alpha) = \bigcup_{q \in Q'} \delta(q, \alpha)$
  - $F_{\text{det}} = \{ Q' \subseteq Q \mid Q' \cap F \neq \emptyset \}$

- Note exponential blow-up in size...
Example

NFA $A$

regexp: $(\alpha + \beta)^* \beta (\alpha + \beta)$
Example

\[
\begin{align*}
\text{NFA } A & \quad \text{DFA } A_{\text{det}} \\
\begin{array}{c}
\text{regexp:} \\
(\alpha + \beta)^*\beta(\alpha + \beta)
\end{array}
\end{align*}
\]
Other properties of NFA/DFA

• NFA/DFA have the same expressive power
  – but NFA can be more efficient (up to exponentially smaller)

• NFA/DFA are closed under complementation
  – build total DFA, swap accept/non-accept states

• For any regular language L, there is a unique minimal DFA that accepts L (up to isomorphism)
  – efficient algorithm to minimise DFA into equivalent DFA
  – partition refinement algorithm (like for bisimulation)

• Language emptiness of an NFA reduces to reachability
  – $L(A) \neq \emptyset$ iff can reach a state in F from an initial state in $Q_0$
Languages as properties

- Consider a model, i.e. an LTS/DTMC/MDP/…
  - e.g. DTMC $D = (S, s_{\text{init}}, P, \text{Lab})$
  - where labelling Lab uses atomic propositions from set $AP$
  - let $\omega \in \text{Path}(s)$ be some infinite path

- Temporal logic properties
  - for some temporal logic (path) formula $\psi$, does $\omega \models \psi$?

- Traces and languages
  - $\text{trace}(\omega) \in (2^{AP})^\omega$ denotes the projection of state labels of $\omega$
  - i.e. $\text{trace}(s_0s_1s_2s_3\ldots) = \text{Lab}(s_0)\text{Lab}(s_1)\text{Lab}(s_2)\text{Lab}(s_3)\ldots$
  - for some language $L \subseteq (2^{AP})^\omega$, is $\text{trace}(\omega) \in L$?
Example

- **Atomic propositions**
  - $\text{AP} = \{ \text{fail, try} \}$
  - $2^\text{AP} = \{ \emptyset, \{\text{fail}\}, \{\text{try}\}, \{\text{fail, try}\} \}$

- **Paths and traces**
  - e.g. $\omega = s_0 s_1 s_1 s_2 s_0 s_1 s_2 s_0 s_1 s_3 s_3 s_3 \ldots$
  - $\text{trace}(\omega) = \emptyset \{\text{try}\} \{\text{try}\} \{\text{fail}\} \emptyset \{\text{try}\} \{\text{fail}\} \emptyset \{\text{try}\} \emptyset \emptyset \emptyset \ldots$

- **Languages**
  - e.g. “no failures”
  - $L = \{ \alpha_1 \alpha_2 \ldots \in (2^\text{AP})^\omega \mid \alpha_i \text{ is } \emptyset \text{ or } \{\text{try}\} \text{ for all } i \}$
Regular safety properties

• **A safety property** \( P \) is a language over \( 2^{AP} \) such that
  - for any word \( w \) that violates \( P \) (i.e. is not in the language), \( w \) has a prefix \( w' \), all extensions of which, also violate \( P \)

• **A regular safety property** is
  - safety property for which the set of “bad prefixes” (finite violations) forms a regular language

• **Formally…**
  - \( P \subseteq (2^{AP})^\omega \) is a safety property if:
    1. \( \forall w \in ((2^{AP})^\omega \setminus P) \cdot \exists \) finite prefix \( w' \) of \( w \) such that:
    2. \( P \cap \{ w'' \in (2^{AP})^\omega | w' \text{ is a prefix of } w'' \} = \emptyset \)
  - \( P \) is a regular safety property if:
    1. \( \{ w' \in (2^{AP})^* | \forall w'' \in (2^{AP})^\omega . w'.w'' \notin P \} \) is regular
Regular safety properties

- A safety property $P$ is a language over $2^{AP}$ such that
  - for any word $w$ that violates $P$ (i.e. is not in the language), $w$ has a prefix $w'$, all extensions of which, also violate $P$

- A regular safety property is
  - safety property for which the set of “bad prefixes” (finite violations) forms a regular language

- Examples:
  - “at least one traffic light is always on”
  - “two traffic lights are never on simultaneously”
  - “a red light is always preceded immediately by an amber light”
Example

- Regular safety property:
  - “at most 2 failures occur”
  - language over:
    \[2^{AP} = \{ \emptyset, \{\text{fail}\}, \{\text{try}\}, \{\text{fail,try}\} \}\]

\[
\begin{array}{c}
\text{s}_0 \\
\text{s}_1 \\
\text{s}_2 \\
\text{s}_3 \\
\end{array}
\]

\[
\begin{array}{c}
\{\text{try}\} \\
0.01 \quad 0.98 \\
0.01 \\
\{\text{fail}\} \\
1 \\
\end{array}
\]
Example

• Regular safety property:
  – “at most 2 failures occur”
  – language over:
    \[ 2^{\text{AP}} = \{ \emptyset, \{\text{fail}\}, \{\text{try}\}, \{\text{fail,try}\} \} \]

• Bad prefixes (regexp):
  \[ (\neg \text{fail})^* \cdot \text{fail} \cdot (\neg \text{fail})^* \cdot \text{fail} \cdot (\neg \text{fail})^* \cdot \text{fail} \]

• Bad prefixes (DFA):

  \( \neg \text{fail} \) denotes: \[ (\{\text{fail}\} + \{\text{fail,try}\}) \]
  \( \neg \text{fail} \) denotes: \[ (\emptyset + \{\text{try}\}) \]

  fail denotes:
  \[ \{\text{fail}\}, \{\text{fail,try}\} \]
  \( \neg \text{fail} \) denotes:
  \[ \emptyset, \{\text{try}\} \]
Regular safety properties + DTMCs

- Consider a DTMC $D$ (with atomic propositions from $AP$) and a regular safety property $P \subseteq (2^AP)^\omega$

- Let $\text{Prob}^D(s, P)$ denote the probability of $P$ being satisfied
  - i.e. $\text{Prob}^D(s, P) = \Pr^D_s\{ \omega \in \text{Path}(s) \mid \text{trace}(\omega) \in P \}$
  - where $\Pr^D_s$ is the probability measure over $\text{Path}(s)$ for $D$
  - this set is always measurable (see later)

- Example (safety) specifications
  - “the probability that at most 2 failures occur is $\geq 0.999$”
  - “what is the probability that at most 2 failures occur?”

- How to compute $\text{Prob}^D(s, P)$?
Product DTMC

- We construct the product of
  - a DTMC $D = (S, s_{\text{init}}, P, L)$
  - and a (total) DFA $A = (Q, \Sigma, \delta, q_0, F)$
  - intuitively: records state of $A$ for path fragments of $D$

- The product DTMC $D \otimes A$ is:
  - the DTMC $(S \times Q, (s_{\text{init}}, q_{\text{init}}), P', L')$ where:
    - $q_{\text{init}} = \delta(q_0, L(s_{\text{init}}))$
    - $P'((s_1, q_1), (s_2, q_2)) = \begin{cases} P(s_1, s_2) & \text{if } q_2 = \delta(q_1, L(s_2)) \\ 0 & \text{otherwise} \end{cases}$
    - $L'(s, q) = \{ \text{accept} \}$ if $q \in F$ and $L'(s, q) = \emptyset$ otherwise
Example

DTMC D

DFA A

fail denotes: \{\text{fail}\}, \{\text{fail,try}\}

\neg \text{fail} \text{ denotes: } \emptyset, \{\text{try}\}
Example

Product DTMC $D \otimes A$

states beyond “accept” state unimportant

{s_0, \varepsilon(q_0, L(s_0))}
Product DTMC

- **One interpretation of** $D \otimes A$:
  - unfolding of $D$ where $q$ for each state $(s,q)$ records state of automata $A$ for path fragment so far

- **In fact, since** $A$ **is deterministic…**
  - for any $\omega \in \text{Path}(s)$ of the DTMC $D$:
    - there is a unique run in $A$ for $\text{trace}(\omega)$
    - and a corresponding (unique) path through $D \otimes A$
  - for any path $\omega' \in \text{Path}^{D \otimes A}(s,q_{\text{init}})$ where $q_{\text{init}} = \delta(q_0,L(s))$
    - there is a corresponding path in $D$ and a run in $A$

- **DFA has no effect on probabilities**
  - i.e. probabilities preserved in product DTMC
Regular safety properties + DTMCs

- Regular safety property $P \subseteq (2^{AP})^\omega$
  - “bad prefixes” (finite violations) represented by DFA $A$

- Probability of $P$ being satisfied in state $s$ of $D$
  - $\Pr_D^{s} \{ \omega \in \text{Path}(s) \mid \text{trace}(\omega) \in P \} = 1 - \Pr_D^{s} \{ \omega \in \text{Path}(s) \mid \text{trace}(\omega) \notin P \} = 1 - \Pr_D^{s} \{ \omega \in \text{Path}(s) \mid \text{pref}(\text{trace}(\omega)) \cap L(A) \neq \emptyset \}$
  - where $\text{pref}(w)$ = set of all finite prefixes of infinite word $w$

\[
\Pr_D^{s}(P) = 1 - \Pr_D^{s\otimes A}((s,q_s), F \text{ accept})
\]

- where $q_s = \delta(q_0,L(s))$
Example

• $\text{Prob}^D(s_0, \text{“at most 2 failures occur”})$
  
  $= 1 - \text{Prob}^{D \otimes A}((s_0,q_0), \text{F accept})$
  
  $= 1 - (1/99)^3$
  
  $\approx 0.9999989694$
Summing up…

- **Nondeterministic finite automata (NFA)**
  - can represent any regular language, regular expression
  - closed under complementation, intersection, ...
  - (non-)emptiness reduces to reachability

- **Deterministic finite automata (DFA)**
  - can be constructed from NFA through determinisation
  - equally expressive as NFA, but may be larger

- **Regular safety properties**
  - language representing set of possible traces
  - bad (violating) prefixes form a regular language

- **Probability of a regular safety property on a DTMC**
  - construct product DTMC
  - reduces to probabilistic reachability