Pattern Matching and Regular Sets

Finite Representations of a language

- (a) Finite Automaton structure, (b) **Pattern matching**.

Finite Representations of a language

- (a) Finite Automaton structure, (b) **Pattern matching**.
- Example: When we type *.ext on a console we are pattern matching with any file with the same extension.

Finite Representations of a language

- (a) Finite Automaton structure, (b) **Pattern matching**.
- Example: When we type *.ext on a console we are pattern matching with any file with the same extension.
- Note: Pattern matching is an important application of finite automata. Grep, fgrep, egrep are pattern matching commands and they use finite automata in their implementation.

What is Pattern Matching?

- Σ is the finite alphabet. A *pattern* is a single string of symbols that represents a subset of strings in Σ^*. Eg. *.ext

What is Pattern Matching?

- Σ is the finite alphabet. A *pattern* is a single string of symbols that represents a subset of strings in Σ^*. Eg. *.ext
- Two kinds – *atomic and compound*.

What is Pattern Matching?

- Σ is the finite alphabet. A *pattern* is a single string of symbols that represents a subset of strings in Σ^*. Eg. *.ext
- Two kinds – *atomic and compound*.
- Notational Convention: Denoted by Greek letters α, β etc.

Atomic Patterns

- a for each $a \in \Sigma$,
  ϵ,
  $\emptyset$,
  $\#$,
  $@$.

Atomic Patterns

- a for each $a \in \Sigma$,
  ϵ,
  $\emptyset$,
  $\#$,
  $@$.
- Given a pattern α, $L(\alpha) = \{x | x \text{ matches the pattern } \alpha\}$.

Atomic Patterns

- a for each $a \in \Sigma$,
  ϵ,
  $\emptyset$,
  $\#$,
  $@$.
- Given a pattern α, $L(\alpha) = \{x | x \text{ matches the pattern } \alpha\}$.
- What are the strings that match to these atomic patterns?
  $\{a\}, \{\epsilon\}, \emptyset, \Sigma, \Sigma^*$, respectively.

Compound Patterns

- Inductively defined from atomic patterns using **binary operators** $+, \cap, \cdot$, and **unary operators** $^*, ^+, \sim$ (or $\neg$).

Compound Patterns

- Inductively defined from atomic patterns using **binary operators** $+, \cap, \cdot$, and **unary operators** $^*, ^+, \sim$ (or $\neg$).
- If α and β are patterns then so are $\alpha + \beta$, $\alpha \cap \beta$, $\alpha \cdot \beta$, α^*, α^+, $\sim \alpha$ (or $\neg\alpha$).

Strings matching to Compound Patterns

- $L(\alpha + \beta) = L(\alpha) \cup L(\beta)$

Strings matching to Compound Patterns

- $L(\alpha + \beta) = L(\alpha) \cup L(\beta)$
- $L(\alpha \cap \beta) = L(\alpha) \cap L(\beta)$

Strings matching to Compound Patterns

- $L(\alpha + \beta) = L(\alpha) \cup L(\beta)$
- $L(\alpha \cap \beta) = L(\alpha) \cap L(\beta)$
- $L(\alpha \cdot \beta) = L(\alpha)L(\beta)$

Strings matching to Compound Patterns

- $L(\alpha + \beta) = L(\alpha) \cup L(\beta)$
- $L(\alpha \cap \beta) = L(\alpha) \cap L(\beta)$
- $L(\alpha \cdot \beta) = L(\alpha)L(\beta)$
- $L(\alpha^*) = L(\alpha)^*$, concatenation of strings of length ≥ 0

Strings matching to Compound Patterns

- $L(\alpha + \beta) = L(\alpha) \cup L(\beta)$
- $L(\alpha \cap \beta) = L(\alpha) \cap L(\beta)$
- $L(\alpha \cdot \beta) = L(\alpha)L(\beta)$
- $L(\alpha^*) = L(\alpha)^*$, concatenation of strings of length ≥ 0
- $L(\alpha^+) = L(\alpha)^+$, concatenation of length ≥ 1 strings

Strings matching to Compound Patterns

- $L(\alpha + \beta) = L(\alpha) \cup L(\beta)$
- $L(\alpha \cap \beta) = L(\alpha) \cap L(\beta)$
- $L(\alpha \cdot \beta) = L(\alpha)L(\beta)$
- $L(\alpha^*) = L(\alpha)^*$, concatenation of strings of length ≥ 0
- $L(\alpha^+) = L(\alpha)^+$, concatenation of length ≥ 1 strings
- $L(\sim \alpha) = \sim L(\alpha) = \Sigma^* - L(\alpha)$

Patterns and Properties of Matching

- So patterns are strings over symbols $\Sigma^\dagger = \Sigma \cup \{\epsilon, \emptyset, \#, \texttt{@}, +, \cap, \cdot, ^+, ^*, \sim, (,)\}$

Patterns and Properties of Matching

- So patterns are strings over symbols $\Sigma^{\dagger} = \Sigma \cup \{\epsilon, \emptyset, \#, @, +, \cap, \cdot, ^{+}, ^{*}, \sim, (,)\}$
- Meaning of $\#$, $@$, $\sim$ depends on Σ.

Patterns and Properties of Matching

- So patterns are strings over symbols $\Sigma^\dagger = \Sigma \cup \{\epsilon, \emptyset, \#, \text{@}, +, \cap, \cdot, ^+, ^*, \sim, (,)\}$
- Meaning of $\#$, @, $\sim$ depends on Σ.
- Eg. $x \in \Sigma^*$ is a pattern, $L(x)$ is $\{x\}$.
  What is $L(x_1 + x_2 + x_3)$?

Patterns and Properties of Matching

- So patterns are strings over symbols $\Sigma^\dagger = \Sigma \cup \{\epsilon, \emptyset, \#, \mathtt{@}, +, \cap, \cdot, ^+, ^*, \sim, (,)\}$
- Meaning of $\#$, $\mathtt{@}$, $\sim$ depends on Σ.
- Eg. $x \in \Sigma^*$ is a pattern, $L(x)$ is $\{x\}$.
  What is $L(x_1 + x_2 + x_3)$?
- Note: $+$ is associative. So is $\cdot$.

Patterns and Properties of Matching

- So patterns are strings over symbols $\Sigma^\dagger = \Sigma \cup \{\epsilon, \emptyset, \#, @, +, \cap, \cdot, ^+, ^*, \sim, (,)\}$
- Meaning of $\#$, $@$, $\sim$ depends on Σ.
- Eg. $x \in \Sigma^*$ is a pattern, $L(x)$ is $\{x\}$.
  What is $L(x_1 + x_2 + x_3)$?
- Note: $+$ is associative. So is $\cdot$.
- $@a@a$ means: Set of strings with at least 2 a's and ending in a.

Patterns and Properties of Matching

- So patterns are strings over symbols $\Sigma^\dagger = \Sigma \cup \{\epsilon, \emptyset, \#, @, +, \cap, \cdot, ^+, ^*, \sim, (,)\}$
- Meaning of $\#$, $@$, $\sim$ depends on Σ.
- Eg. $x \in \Sigma^*$ is a pattern, $L(x)$ is $\{x\}$.
  What is $L(x_1 + x_2 + x_3)$?
- Note: $+$ is associative. So is $\cdot$.
- $@a@a$ means: Set of strings with at least 2 a's and ending in a.
- Language L over Σ where every a has at least one b after it (two a's may have the same b after them): is there a pattern α s.t $L = L(\alpha)$?
  $(\# \cap \sim a)^* + @b(\# \cap \sim a)^*$

Patterns and Properties of Matching

- So patterns are strings over symbols $\Sigma^\dagger = \Sigma \cup \{\epsilon, \emptyset, \#, @, +, \cap, \cdot, ^+, ^*, \sim, (,)\}$
- Meaning of $\#$, $@$, $\sim$ depends on Σ.
- Eg. $x \in \Sigma^*$ is a pattern, $L(x)$ is $\{x\}$.
  What is $L(x_1 + x_2 + x_3)$?
- Note: $+$ is associative. So is $\cdot$.
- $@a@a$ means: Set of strings with at least 2 a's and ending in a.
- Language L over Σ where every a has at least one b after it (two a's may have the same b after them): is there a pattern α s.t $L = L(\alpha)$?
  $(\# \cap \sim a)^* + @b(\# \cap \sim a)^*$
- Above language L if $\Sigma = \{a, b\}$: what is the pattern?
  $\epsilon + @b$.

Questions

1. How hard is it to check if a string matches a given pattern?

Questions

1. How hard is it to check if a string matches a given pattern?
2. If I give you any set of strings, can it be matched by a pattern? *A pattern is a string over some $\Sigma^{\dagger}$. There can be countably infinite such patterns – think of it as a k-ary representation where k is the number of symbols in $\Sigma^{\dagger}$; each pattern then maps to a unique natural number. If a set matches to the pattern, the strings are over Σ. The number of possible subsets is a power set of Σ^*. So uncountable. So there will be sets that cannot be matched to a pattern.*

Questions

1. How hard is it to check if a string matches a given pattern?
2. If I give you any set of strings, can it be matched by a pattern? *A pattern is a string over some $\Sigma^{\dagger}$. There can be countably infinite such patterns – think of it as a k-ary representation where k is the number of symbols in $\Sigma^{\dagger}$; each pattern then maps to a unique natural number. If a set matches to the pattern, the strings are over Σ. The number of possible subsets is a power set of Σ^*. So uncountable. So there will be sets that cannot be matched to a pattern.*
3. Patterns α and β are equivalent ($\alpha \equiv \beta$) if $L(\alpha) = L(\beta)$. How can you find out equivalence?

Questions

1. How hard is it to check if a string matches a given pattern?
2. If I give you any set of strings, can it be matched by a pattern? *A pattern is a string over some* $\Sigma^{\dagger}$*. There can be countably infinite such patterns – think of it as a* k*-ary representation where* k *is the number of symbols in* $\Sigma^{\dagger}$*; each pattern then maps to a unique natural number. If a set matches to the pattern, the strings are over* Σ*. The number of possible subsets is a power set of* Σ^**. So uncountable. So there will be sets that cannot be matched to a pattern.*
3. Patterns α and β are equivalent ($\alpha \equiv \beta$) if $L(\alpha) = L(\beta)$. How can you find out equivalence?
4. Which operators are redundant?

Which operators are redundant?

- Eg. ϵ is equivalent to $\sim(\#@)$ or $\emptyset^*$.
  @ is same as $\#^*$.
  $^+$ not necessary: $a^+ = aa^*$
  $\#$ not necessary: $\Sigma = \{a, b, ..z\}$ means $\# = a + b + \ldots z$
  $\cap$ is redundant – $a \cap b = \sim(\sim a + \sim b)$
  Can be shown that $\sim$ is also redundant.

Which operators are redundant?

- Eg. ϵ is equivalent to $\sim(\#@)$ or $\emptyset^*$.
  $@$ is same as $\#^*$.
  $^+$ not necessary: $a^+ = aa^*$
  $\#$ not necessary: $\Sigma = \{a, b, ..z\}$ means $\# = a + b + \ldots z$
  $\cap$ is redundant – $a \cap b = \sim(\sim a + \sim b)$
  Can be shown that $\sim$ is also redundant.
- Thus, each pattern is equivalent to one with only atomic patterns $a \in \Sigma, \epsilon, \emptyset$ and operators $+, \cdot, ^*$.

Which operators are redundant?

- Eg. ϵ is equivalent to $\sim (\#@)$ or $\emptyset^*$.
  @ is same as $\#^*$.
  $^+$ not necessary: $a^+ = aa^*$
  $\#$ not necessary: $\Sigma = \{a, b, ..z\}$ means $\# = a + b + \ldots z$
  $\cap$ is redundant – $a \cap b = \sim (\sim a + \sim b)$
  Can be shown that $\sim$ is also redundant.
- Thus, each pattern is equivalent to one with only atomic patterns $a \in \Sigma, \epsilon, \emptyset$ and operators $+, \cdot, ^*$.
- Note: Atomic pattern ϵ is also redundant but we keep it for notational simplicity.

Which operators are redundant?

- Eg. ϵ is equivalent to $\sim(\#@)$ or $\emptyset^*$.
  $@$ is same as $\#^*$.
  $^+$ not necessary: $a^+ = aa^*$
  $\#$ not necessary: $\Sigma = \{a, b, ..z\}$ means $\# = a + b + \ldots z$
  $\cap$ is redundant – $a \cap b = \sim(\sim a + \sim b)$
  Can be shown that $\sim$ is also redundant.
- Thus, each pattern is equivalent to one with only atomic patterns $a \in \Sigma, \epsilon, \emptyset$ and operators $+, \cdot, ^*$.
- Note: Atomic pattern ϵ is also redundant but we keep it for notational simplicity.
- A pattern that only uses the above atomic patterns and operators is called a **regular expression**.

Notational Conventions for Patterns

- $\cdot$ given preference over $+$.
  Eg: $\alpha + \beta\gamma$ is $\alpha + (\beta\gamma)$ and not $(\alpha + \beta)\gamma$.

Notational Conventions for Patterns

- $\cdot$ given preference over $+$.
  Eg: $\alpha + \beta\gamma$ is $\alpha + (\beta\gamma)$ and not $(\alpha + \beta)\gamma$.
- * given preference over $+$ or $\cdot$.
  Eg: $\alpha + \beta^*$ is $\alpha + (\beta^*)$ and not $(\alpha + \beta)^*$

Notational Conventions for Patterns

- $\cdot$ given preference over $+$.
  Eg: $\alpha + \beta\gamma$ is $\alpha + (\beta\gamma)$ and not $(\alpha + \beta)\gamma$.
- * given preference over $+$ or $\cdot$.
  Eg: $\alpha + \beta^*$ is $\alpha + (\beta^*)$ and not $(\alpha + \beta)^*$
- Or use parenthesis properly!

Regular Expressions and Regular Sets

- **Theorem**: Equivalent statements:

Regular Expressions and Regular Sets

- **Theorem**: Equivalent statements:
- A. A is a regular set

Regular Expressions and Regular Sets

- **Theorem**: Equivalent statements:
- A. A is a regular set
- B. $A = L(\alpha)$ for a pattern α

Regular Expressions and Regular Sets

- **Theorem**: Equivalent statements:
- A. A is a regular set
- B. $A = L(\alpha)$ for a pattern α
- C. $A = L(\alpha)$ for a regular expression α.

Theorem: $C \implies B$

$A = L(\alpha)$ for a regular expression $\alpha \implies A = L(\alpha)$ for a pattern α.
Proof: $C \implies B$ from definition.

Theorem: $B \implies A$

$A = L(\alpha)$ for a pattern $\alpha \implies A$ is a regular set

- Proof :

Theorem: $B \implies A$

$A = L(\alpha)$ for a pattern $\alpha \implies A$ is a regular set

- Proof :
- Singleton set $\{a\}$ is regular (How?)

Theorem: $B \implies A$

$A = L(\alpha)$ for a pattern $\alpha \implies A$ is a regular set

- Proof :
- Singleton set $\{a\}$ is regular (How?)
- $\{\epsilon\}$ is regular (How?)

Theorem: $B \implies A$

$A = L(\alpha)$ for a pattern $\alpha \implies A$ is a regular set

- Proof :
- Singleton set $\{a\}$ is regular (How?)
- $\{\epsilon\}$ is regular (How?)
- $\emptyset$ is regular (How?)

Theorem: $B \implies A$ contd.

- We have shown regular sets are closed under $\cap, \cup, \sim$ (or $\neg$), $\cdot, ^*$.

Theorem: $B \implies A$ contd.

- We have shown regular sets are closed under $\cap, \cup, \sim$ (or $\neg$), $\cdot,^*$.
- Closure under $^+$ (Final states make an ϵ transition to one new final state which goes to start state by an ϵ transition)

Theorem: $B \implies A$ contd.

- We have shown regular sets are closed under $\cap, \cup, \sim$ (or $\neg$), $\cdot, ^*$.
- Closure under $^+$ (Final states make an ϵ transition to one new final state which goes to start state by an ϵ transition)
- Now we induct on the length of the pattern. What is the form of the pattern?

Theorem: $B \implies A$ contd.

- We have shown regular sets are closed under $\cap, \cup, \sim$ (or $\neg$), $\cdot, ^*$.
- Closure under $^+$ (Final states make an ϵ transition to one new final state which goes to start state by an ϵ transition)
- Now we induct on the length of the pattern. What is the form of the pattern?
- Base case:
  1. a for some $a \in \Sigma$: $L(a) = \{a\}$ a regular set
  2. ϵ : $L(\epsilon) = \{\epsilon\}$ a regular set
  3. $\emptyset$: $L(\emptyset) = \emptyset$ a regular set
  4. $\#$ - redundant
  5. @ – redundant

Theorem: $B \implies A$ contd.

- Induction: For compound pattern, induction on the number of operators.

  6. β^+ – redundant

  7. $\beta + \gamma$: $L(\beta + \gamma) = L(\beta) \cup L(\gamma)$. By induction, β and γ give regular sets. Closure under $\cup$ gives regular set.

  8. $L(\beta \cap \gamma) = L(\beta) \cap L(\gamma)$: regular set

  9. $L(\beta \cdot \gamma) = L(\beta) \cdot L(\gamma)$: regular set

  10. $L(\beta^*) = L(\beta)^*$: regular set

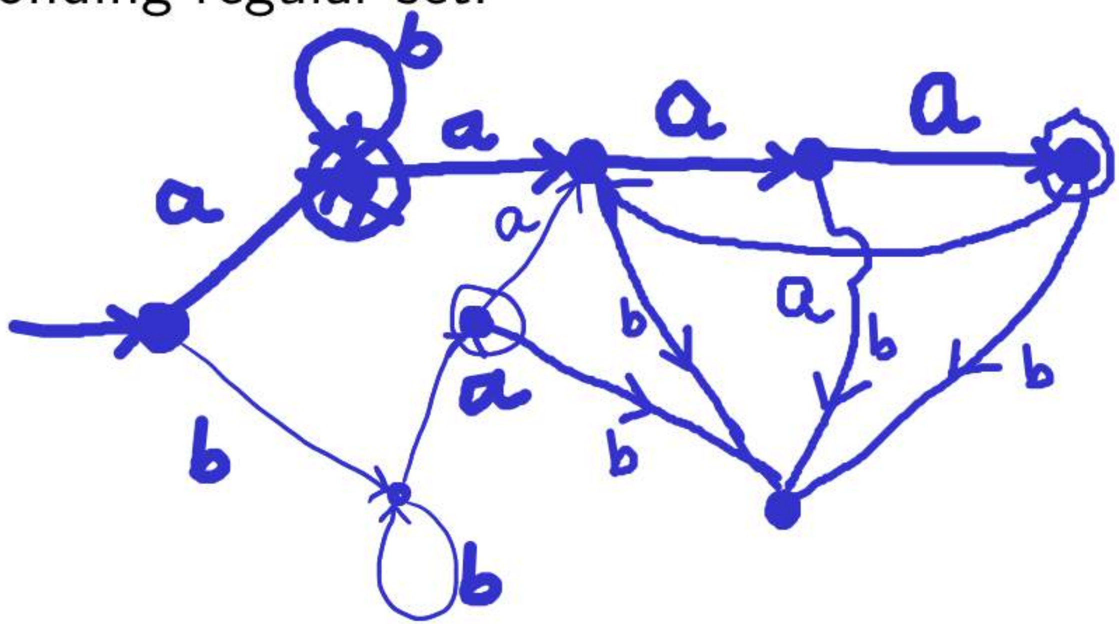
  11. $L(\sim \beta)$ or $L(\neg\beta) = \sim L(\beta)$: regular set

Theorem: $B \implies A$ contd.

- Induction: For compound pattern, induction on the number of operators.
  6. β^+ – redundant
  7. $\beta + \gamma$: $L(\beta + \gamma) = L(\beta) \cup L(\gamma)$. By induction, β and γ give regular sets. Closure under $\cup$ gives regular set.
  8. $L(\beta \cap \gamma) = L(\beta) \cap L(\gamma)$: regular set
  9. $L(\beta \cdot \gamma) = L(\beta) \cdot L(\gamma)$: regular set
  10. $L(\beta^*) = L(\beta)^*$: regular set
  11. $L(\sim \beta)$ or $L(\neg\beta) = \sim L(\beta)$: regular set
- Thus, done.

Intermezzo

- Eg: Convert the regular expression $(ab^* + b^*a)(aaa)^*$ to the corresponding regular set.



Intermezzo

- Eg: Convert the regular expression $(ab^* + b^*a)(aaa)^*$ to the corresponding regular set.
- Eg. Try to convert $\{a^n b^n | n \geq 0\}$ to a pattern. Is this regular? Is this a pattern, or does there exist a pattern? - Will eventually lead to answer to Q2.

Intermezzo

- Eg: Convert the regular expression $(ab^* + b^*a)(aaa)^*$ to the corresponding regular set.
- Eg. Try to convert $\{a^n b^n | n \geq 0\}$ to a pattern. Is this regular? Is this a pattern, or does there exist a pattern? - Will eventually lead to answer to Q2.
- Going back to the Questions: Q1 – How will you match a string to a given pattern? $[B \implies A]$

Intermezzo

- Eg: Convert the regular expression $(ab^* + b^*a)(aaa)^*$ to the corresponding regular set.
- Eg. Try to convert $\{a^n b^n | n \geq 0\}$ to a pattern. Is this regular? Is this a pattern, or does there exist a pattern? - Will eventually lead to answer to Q2.
- Going back to the Questions: Q1 – How will you match a string to a given pattern? [$B \implies A$]
- (Q3 – We will have a look later).

Theorem: $A \implies C$

A is a regular set $\implies$ $A = L(\alpha)$ for a regular expression α.

- Proof:

Theorem: $A \implies C$

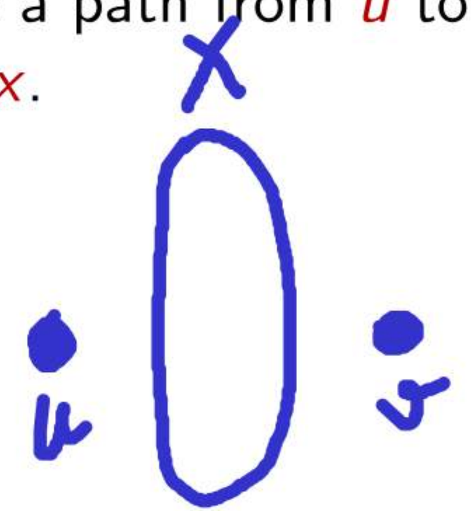
A is a regular set $\implies$ $A = L(\alpha)$ for a regular expression α.

- Proof:
- Suppose I have an NFA $M = (Q, \Sigma, \Delta, S, F)$.
  We will be defining an equivalent regular expression for $L(M)$.

Theorem: $A \implies C$

A is a regular set $\implies$ $A = L(\alpha)$ for a regular expression α.

- Proof:
- Suppose I have an NFA $M = (Q, \Sigma, \Delta, S, F)$.
  We will be defining an equivalent regular expression for $L(M)$.
- **Aim**: For a subset $X \subseteq Q$ and states u, v, let α_{uv}^{X} be a regular expression for all strings x that have a path from u to v with all internal vertices in X labelled by x.



Theorem: $A \implies C$

A is a regular set $\implies$ $A = L(\alpha)$ for a regular expression α.

- Proof:
- Suppose I have an NFA $M = (Q, \Sigma, \Delta, S, F)$. We will be defining an equivalent regular expression for $L(M)$.
- **Aim**: For a subset $X \subseteq Q$ and states u, v, let α_{uv}^{X} be a regular expression for all strings x that have a path from u to v with all internal vertices in X labelled by x.
- **Implication**: If we did this for all u, v and all subsets X, then $\Sigma_{s \in S} \Sigma_{f \in F} \alpha_{sf}^{Q}$ would be a regular expression for all strings in $L(M)$. We will be done.

Theorem: $A \implies C$ contd.

- Proving the Aim by induction on size of X.

Theorem: $A \implies C$ contd.

- Proving the Aim by induction on size of X.
- Base case: X is $\emptyset$.

Theorem: $A \implies C$ contd.

- Proving the Aim by induction on size of X.
- Base case: X is $\emptyset$.
- If $u \neq v$, $\alpha_{uv}^{\emptyset}$
  $=$ Sum over all elements in Σ' , $\Sigma' = \{$alphabets that are labels on outgoing edges of $u\}$
  $= \emptyset$ if $\Sigma' = \emptyset$

Theorem: $A \implies C$ contd.

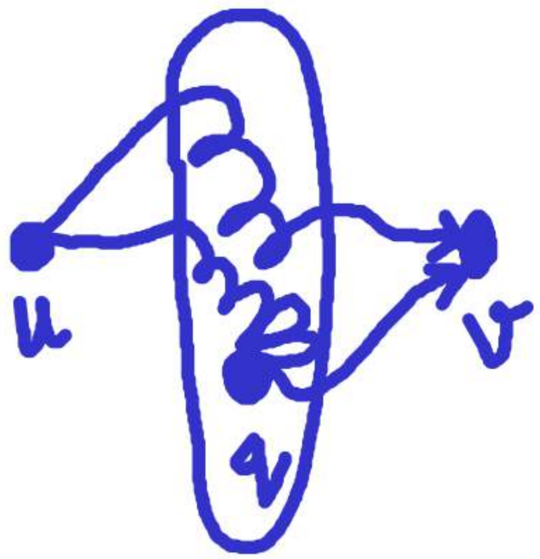
- Proving the Aim by induction on size of X.
- Base case: X is $\emptyset$.
- If $u \neq v$, $\alpha_{uv}^{\emptyset}$
  $=$ Sum over all elements in Σ', Σ' = {alphabets that are labels on outgoing edges of u}
  $= \emptyset$ if $\Sigma' = \emptyset$
- If $u = v$, then $\alpha_{uv}^{\emptyset}$
  $=$ Sum over all elements in $\Sigma' + \epsilon$ [all possible labelled loops plus staying in the same state means no input read]
  $= \epsilon$ if $\Sigma' = \emptyset$

Theorem: $A \implies C$ contd.

- Now inductive definition of α_{uv}^{X}. Take some $q \in X$

Theorem: $A \implies C$ contd.

- Now inductive definition of α_{uv}^{X}. Take some $q \in X$
- $\alpha_{uv}^{X} = \alpha_{uv}^{X-q} + \alpha_{uq}^{X-q}(\alpha_{qq}^{X-q})^{*}\alpha_{qv}^{X-q}$.

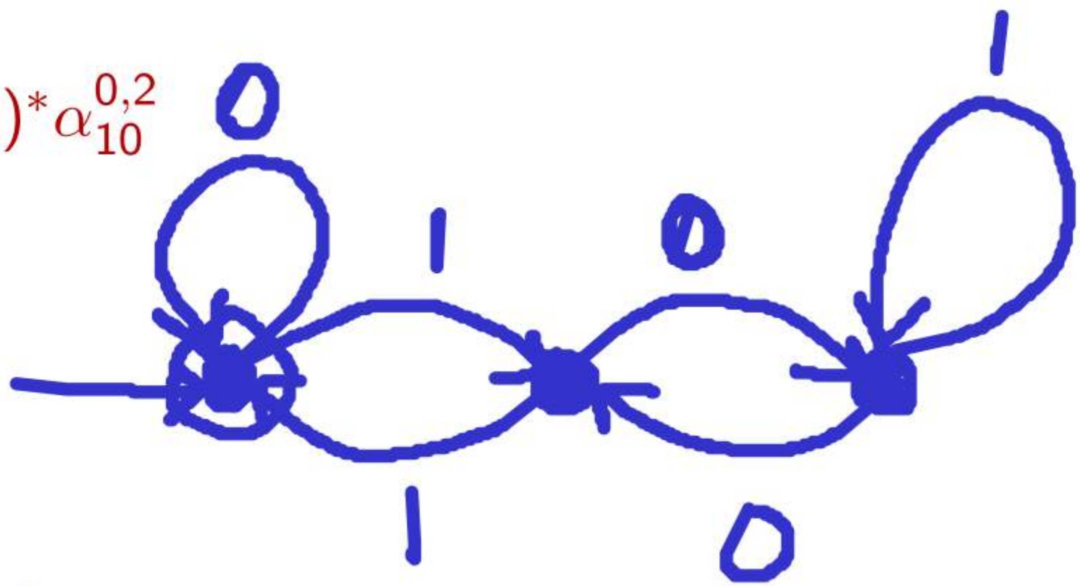


Theorem: $A \implies C$ contd.

- Now inductive definition of α_{uv}^{X}. Take some $q \in X$
- $\alpha_{uv}^{X} = \alpha_{uv}^{X-q} + \alpha_{uq}^{X-q}(\alpha_{qq}^{X-q})^{*}\alpha_{qv}^{X-q}$.
- By IH on size of X, RHS combines to form a regular expression. So, we are done with proving Aim, and therefore Implication.

Example: Regular Expressions for DFA for binary strings divisible by 3

- $\alpha_{00}^{0,1,2} = \alpha_{00}^{0,2} + \alpha_{01}^{0,2}(\alpha_{11}^{0,2})^*\alpha_{10}^{0,2}$
- $\alpha_{00}^{0,2} = 0^*$
- $\alpha_{01}^{0,2} = 0^*1$
- $\alpha_{11}^{0,2} = 01^*0 + 10^*1$
- $\alpha_{10}^{0,2} = 10^*$
- So $0^* + 0^*1(01^*0 + 10^*1)^*01^*$



$0^* + 0^*1(01^*0 + 10^*1)^*01^*$!!

- Can we get a simpler expression? Recall Q3. Is there an equivalent expression that is simpler?

$$0^* + 0^*1(01^*0 + 10^*1)^*01^* \text{ !!}$$

- Can we get a simpler expression? Recall Q3. Is there an equivalent expression that is simpler?
- Equivalence ($\equiv$): reflexive, symmetric and transitive.
  So if two expressions are equivalent one can substitute the other.

$0^* + 0^*1(01^*0 + 10^*1)^*01^*$!!

- Can we get a simpler expression? Recall Q3. Is there an equivalent expression that is simpler?
- Equivalence ($\equiv$): reflexive, symmetric and transitive.
  So if two expressions are equivalent one can substitute the other.
- Q3 is asking to solve an NP-hard problem!

Laws of Simplification

- $\alpha + (\beta + \gamma) \equiv (\alpha + \beta) + \gamma$
- $\alpha + \beta \equiv \beta + \alpha$
- $\alpha + \emptyset \equiv \alpha$
- $\alpha + \alpha \equiv \alpha$
- $\alpha(\beta \cdot \gamma) \equiv (\alpha \cdot \beta)\gamma$
- $\epsilon \cdot \alpha \equiv \alpha \cdot \epsilon \equiv \alpha$
- $\alpha(\beta + \gamma) \equiv \alpha \cdot \beta + \alpha \cdot \gamma$

Laws of Simplification

- $(\alpha + \beta)\gamma \equiv \alpha \cdot \gamma + \beta \cdot \gamma$
- $\emptyset \cdot \alpha \equiv \alpha \cdot \emptyset \equiv \emptyset$
- $\epsilon + \alpha \cdot \alpha^* \equiv \alpha^* \equiv \epsilon + \alpha^* \alpha$

Notation: $\alpha \leq \beta \iff L(\alpha) \subseteq L(\beta) \iff L(\alpha + \beta) = L(\beta)$, or $\alpha + \beta \equiv \beta$

- $\beta + \alpha \cdot \gamma \leq \gamma \implies \alpha^* \beta \leq \gamma$ (show set theoretically)
- $\beta + \gamma . \alpha \leq \gamma \implies \beta . \alpha^* \leq \gamma$ (show set theoretically)

Other Equations

- $(\alpha \cdot \beta)^* \alpha \equiv \alpha(\beta \cdot \alpha)^*$: Argue that $(\alpha \cdot \beta)^i \cdot \alpha \equiv \alpha(\beta \cdot \alpha)^i$
- $(\alpha^* \beta)^* \alpha^* \equiv (\alpha + \beta)^*$
- $\alpha^*(\beta \cdot \alpha^*)^* \equiv (\alpha + \beta)^*$: Same as above if α^* is taken as γ and the first equation is applied.
- $(\epsilon + \alpha)^* \equiv \alpha^*$: Substitute appropriately in 2nd equation
- $\alpha \cdot \alpha^* \equiv \alpha^* \cdot \alpha$: assume α is not ϵ as otherwise it trivially follows.
  Neither LHS nor RHS matches with the ϵ string. Add ϵ to both sides and this gives α^* to both sides, so LHS must be same as RHS.

Example of Simplification

- $(1 + 01 + 001)^*(\epsilon + 0 + 00)$

Example of Simplification

- $(1 + 01 + 001)^*(\epsilon + 0 + 00)$
- $\equiv ((\epsilon + 0 + 00)1)^*(\epsilon + 0 + 0 + 00)$ [as $1 \equiv \epsilon \cdot 1$, $\alpha \equiv \alpha + \alpha$]

Example of Simplification

- $(1 + 01 + 001)^*(\epsilon + 0 + 00)$
- $\equiv ((\epsilon + 0 + 00)1)^*(\epsilon + 0 + 0 + 00)$ [as $1 \equiv \epsilon \cdot 1$, $\alpha \equiv \alpha + \alpha$]
- $\equiv ((\epsilon + 0)(\epsilon + 0)1)^*(\epsilon + 0)(\epsilon + 0)$

Example of Simplification

- $(1 + 01 + 001)^*(\epsilon + 0 + 00)$
- $\equiv ((\epsilon + 0 + 00)1)^*(\epsilon + 0 + 0 + 00)$ [as $1 \equiv \epsilon \cdot 1$, $\alpha \equiv \alpha + \alpha$]
- $\equiv ((\epsilon + 0)(\epsilon + 0)1)^*(\epsilon + 0)(\epsilon + 0)$
- This defines a set of strings that do not have more than 2 consecutive 0's in any substring.