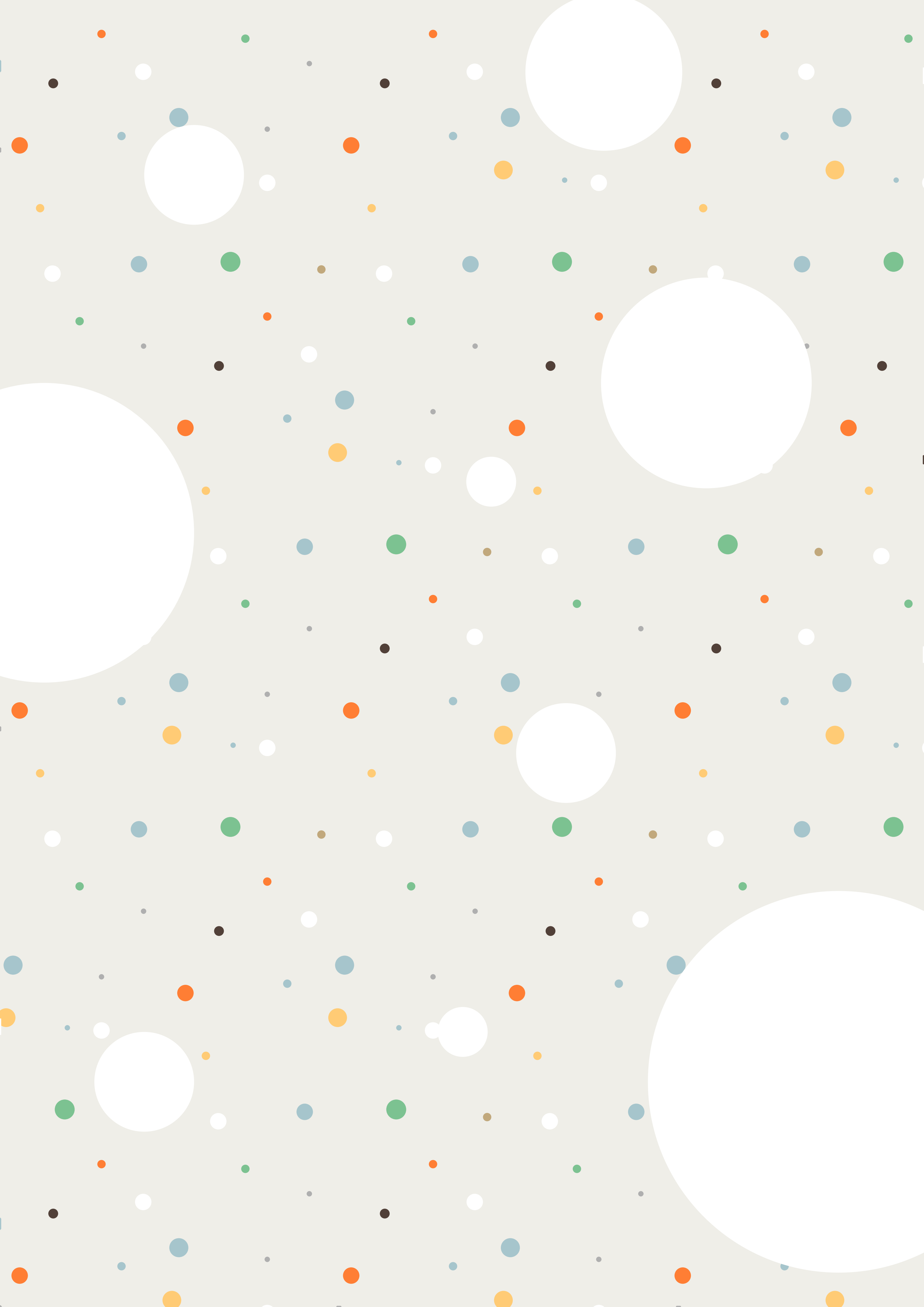

Pre-lecture Video

Alphabet : finite set of symbols e.g. $\Sigma = \{0,1\}$
Σ^* : set of all finite strings using only symbols from Σ
Language (over Σ) : a subset of Σ^* e.g. $L_1 = \phi$, $L_2 = \Sigma^*$.
Convention: Strings are written without quotes. e.g. $x = 011$, $y = 00$, $z = xy = 01100$
ε denotes the empty string.

String reversal : symbols in reverse order . e.g. $x^R = (011)^R = 110$

Language Operations:

Complementation : $\overline{L} = \Sigma^* - L = \{x : x \in \Sigma^*, x \notin L\}$
Union : $L_1 \cup L_2$
Intersection : $L_1 \cap L_2$
Concatenation : $L_1 L_2 = \{xy : x \in L_1, y \in L_2\}$
Kleene star : $L^* = \{x : x = \varepsilon$ or $x = y_1 y_2 \cdots y_k$ for some $k > 0$ and $y_1, \dots, y_k \in L\}$
Exponentation : $L^k = \begin{cases} \{\varepsilon\} & \text{if } k = 0 \\ L^{k-1}L & \text{if } k > 0 \end{cases}$

Reversal : $L^R = \{x^R : x \in L\}$

Remark: $\phi \cdot L = \phi$, $L \cdot \phi = \phi$, $\phi^* = \{\varepsilon\}$

Regular expressions (regex)

A way to describe a language

Given an alphabet Σ , a regex (over Σ) is a string in $(\Sigma \cup \{\phi, \varepsilon, *, +, (,)\})^*$ e.g. $((0+1)(00))^*$

Definition:

The set of regexes (over Σ), called RE , is the smallest set s.t.

Basis: $\phi, \varepsilon \in RE$ and $a \in RE$ for any $a \in \Sigma$.

I.S: If $R, S \in RE$, then $(R+S)$, (RS), $R^* \in RE$

We define $\mathcal{L}(R)$, the language denoted by R (the set of strings that R matches)

$\mathcal{L}(\phi) = \phi$
$\mathcal{L}(\varepsilon) = \{\varepsilon\}$
$\mathcal{L}(a) = \{a\}$, for any $a \in \Sigma$
$\mathcal{L}(R+S) = \mathcal{L}(R) \cup \mathcal{L}(S)$
$\mathcal{L}((RS)) = \mathcal{L}(R)\mathcal{L}(S)$
$\mathcal{L}(R^*) = (\mathcal{L}(R))^*$

R	$\mathcal{L}(R)$
$(0+1)^*$	All strings in $\{0,1\}^*$
$(0(0+1)^*)$	All strings that start with 0

Convention: drop outer most parantheses

e.g. $(0+1) \Rightarrow 0+1$

Precendences (high to low)

(i) star

(ii) concatenation

(iii) + (Union)

Definition:

We say 2 regexes R and S are equivalent iff $\mathcal{L}(R) = \mathcal{L}(S)$

Definition:

Let L be a language. We say L is regular iff there's a regex R s.t. $L = \mathcal{L}(R)$.

Closure Properties for Regular Languages

Let f be a language operation, i.e. $f : P(\Sigma^*) \to P(\Sigma^*)$.

Put another way, f maps a language to a language.

We say f preserves regular languages iff for every regular language L, $f(L)$ is regular.

We also say regular languages are closed under f.

To prove that f preserves regular languages, we can define a predicate (on regexes)

$P(R)$: There exists regex R' s.t. $\mathcal{L}(R') = f(\mathcal{L}(R))$ then prove that $P(R)$ holds for all regexes R.

Ex 1:

Let $\Sigma = \{0,1\}$. Consider this language operation

$$\text{Ins}0(L) = \{ u0v : u,v \in \Sigma^* \text{ and } uv \in L \}$$

Prove that Ins0 preserves regular languages.

Proof:

$P(R)$: $\exists R'$ s.t. $\mathcal{L}(R') = \text{Ins}0(\mathcal{L}(R))$

Prove $P(R)$ holds for all regexes R.

Basis: (3 cases)

(i) If $R = \phi$, then let $R' = \underline{\ \phi\ }$

(ii) If $R = \varepsilon$, then let $R' = \underline{\ 0\ }$

(iii) If $R = b$, where $b \in \Sigma = \{0,1\}$, then let $R' = \underline{0b + b0}$

I.S.: Let S, T be regexes

Suppose $P(S), P(T)$ hold [I.H.]

i.e. there are regexes S', T' s.t. $\mathcal{L}(S') = \text{Ins}0(\mathcal{L}(S))$ and $\mathcal{L}(T') = \text{Ins}0(\mathcal{L}(T))$

WTP: $P(R)$ for 3 cases. $R = S+T$, $R = ST$, $R = S^*$

Case 1: If $R = S+T$, then let $R' = \underline{S' + T'}$

Want: $\mathcal{L}(R') = \text{Ins}0(\mathcal{L}(S+T))$

$= \text{Ins}0(\mathcal{L}(S) \cup \mathcal{L}(T))$

$= \text{Ins}0(\mathcal{L}(S)) \cup \text{Ins}0(\mathcal{L}(T))$

$= \mathcal{L}(S') \cup \mathcal{L}(T')$

$= \mathcal{L}(S' + T')$

Case 2: If $R = ST$, then let $R' = \underline{S'T + ST'}$

Want: $\mathcal{L}(R') = \text{Ins}0(\mathcal{L}(ST))$

$= \text{Ins}0(\mathcal{L}(S)\mathcal{L}(T))$

$= \text{Ins}0(\mathcal{L}(S)) \cdot \mathcal{L}(T) \cup \text{Ins}0(\mathcal{L}(T)) \cdot \mathcal{L}(S)$

$= \mathcal{L}(S')\mathcal{L}(T) \cup \mathcal{L}(S)\mathcal{L}(T')$ [I.H]

$= \mathcal{L}(S'T + ST')$

Case 3: If $R = S^*$, then let $R' = \underline{0 + S^*S'S^*}$

Want: $\mathcal{L}(R') = \text{Ins}0(\mathcal{L}(S^*))$

$= \text{Ins}0(\mathcal{L}(S)^*)$

$= \text{Ins}0(\{\varepsilon\}) \cup \mathcal{L}(S)^* \, \text{Ins}0(\mathcal{L}(S)) \, \mathcal{L}(S)^*$

$= \mathcal{L}(0 + S^*S'S^*)$ ☑

B36 Oct 6 Lec 1 Notes

1. Pre: L is a list of distinct integers
   Post: Return the set of all peaks in L.

Get Peaks(L)

```
if len(L) < 3: return {}    # empty set
m = MaxIndex(L); S = {}
if m > 0:
    S = Getpeaks(L[0:m])
    if m < len(L)-1:
        S = S union {m} union Get Peaks(L[m+1:len(L)])
return S
```

0

[6, 4, 2, 5, 1]

len(L) = 5
m = 0
return ∅.

0 1 2 3 4

[1, 5, 2, 4, 6]

0 1 2 3 4

GP([1, 5, 2, 4, 6])
m = 4

0 1 2 3

↳ GP([1, 5, 2, 4])

Corrected:

GP(L)

```
if len(L) < 3: return {}
m = MaxIndex(L);
if m > 0:
    S1 = GP(L[0:m])
if m < len(L)-1
    S2 = GP(L[m+1:len(L)])
    add m+1 to each element of S2
if 0 < m < len(L)-1: return S1 union {m} union S2
else: return S1 union S2
```

$Q(n)$: If L is a list of distinct integers and $n = \text{len}(L)$
then Get peaks(L) returns the set of all peaks in L.

Basis: $n = 0$, $n = 1$, $n = 2$
$Q(n)$ holds by C1

I.S.: Let $n \geq 3$

Assume $Q(j)$ holds for $0 \leq j < n$. [I.H.]

WTP: $Q(n)$ holds