CSE 396 Lecture Thu 1 Feb 2018 Spr 18

Languages are sets of strings. From an OOP perspective

Basic type: char character over an alphabet Σ.

We will think of string as a basic type, but it's "really": capital sigma.

$$\text{string} = \underbrace{\text{list<char>}}_{\text{"First Order" Object}}$$

A string is a (finite) list of chars (in Σ).

Examples:

$\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$ $x = $ "27" is the list $(2,7)$

13,596 is not a string over Σ, but is over $\Sigma' = \Sigma \cup \{,\}$.

Often we can say Σ is "big enough for all the chars we're using."

Or we can assume $\Sigma = \{0,1\}$. (!) (or $\{a,b\}$.
or $\{\alpha,\beta\}$)

ASCII = {96 printable vs typewriter chars
plus 32 control codes
plus 128 other "upper ASCII" chars}.

We can re-code a char in ASCII {as a string over $\{0,1\}^8$, viz.
as a byte in $\{0,1\}$.

UNICODE uses 16 bits per "wide" char. UTF-8 blends ASCII & UNICODE.

$$\text{language} = \text{Set<string>} = \underbrace{\text{Set<list<char>>}}_{\text{Two uses of < --->}}$$

"Second Order" Object

Class = Set<language> "Third Order" - (Star Wars 17 will feature Jedis vs. the Third Order).

Examples of Languages:

$\Sigma = \{0,1\}$: $L_p = \{10, 11, 101, 111, 1011, 1101\}$

$DIG = \{0, \ldots 9\}$: $L_{PD} = \{\text{"2"}, \text{"3"}, \text{"5"}, \text{"7"}, \text{"11"}, \text{"13"}\}$.

This language expresses the property of being a prime number ≤ 13, but the idea of "prime number" is simpler and yields an infinite language.

$L_{PRIMES} = \{10, 11, 101, 111, 1011, 1101, 10001, \ldots\}$.

Numbers and Strings (and languages of them) will often be interchangeable

"first order"
Natural Numbers $\mathbb{N} = 0, 1, 2, 3, 4, \ldots$ ($\mathbb{N}^+$ starts with 1)

Standard Binary Notation: $0, 1, 10, 11, 100, 101, 110, 111, 1000, \ldots$
(skips strings other than 0 that have leading 0s)

Standard Order on Σ^*: $\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots$

Under the latter, PRIMES could be encoded as $\{0, 1, 01, 11, 011 \ldots\}$

To represent Rational Numbers like $17/355$, use strings like "17/355" or "10001/10101111"

Encoding Other Objects as Strings. E.g. Graphs.

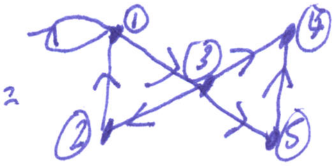
A graph $G = (V, E)$ consists of a set V of nodes and a set $E \subseteq V \times V$ of edges.

A typical edge looks like (a,b) where $a \in V$ and $b \in V$. We can also write $E(a,b)$ as a relation $(a,b) \in E$.

G is undirected if E is symmetric, i.e. $(a,b) \in E \iff (b,a) \in E$.

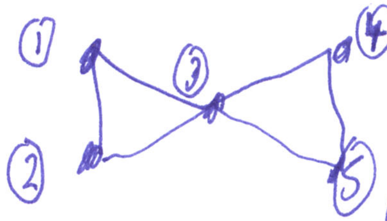
This space accidentally left blank. Fill it by *adding*: 3

By convention, undirected graphs generally don't have self-loops $(a,a) \in E$, but directed graphs often do. E.g.

$G =$ 

has $E = \{(1,1),$ ← self-loop at 1
$(1,3), (3,2), (2,1),$ } A directed triangle
$(3,4), (5,4), (3,5)\}$ } not a directed triangle

We can also label edges by changing e.g. $(1,3)$ to $(1,\ell,3)$ where ℓ is the label.
Using chars (and later ε's) as labels defines finite automata (ch. 1).

Example: $G =$ 

"$V = \{1,2,3,4,5\}$
$E = \{(1,2), (1,3), (2,3), (3,4), (4,5), (3,5)\}$ "

And i.e. $\cup \{(2,1), (3,1), (3,2), (4,3), (5,4), (5,3)\}$
[but for an undirected graph we can ignore this]

To encode G as a string, we can use an

Edge List

If I want a binary string encoding of G, I could convert "..." from ASCII to bytes, or I could do an adjacency matrix

$A_G =$

	1	2	3	4	5
1	0	1	1	0	0
2	1	0	1	0	0
3	1	1	0	1	1
4	0	0	1	0	1
5	0	0	1	1	0

Finally "unroll" A_G (in row-major order) as the string $x_G = 01100 \cdot 10100 \cdot 11011 \cdot 00101 \cdot 00110$

$A_G[i,j] = 1$ if $(i,j) \in E$, 0 otherwise

Adjacency Encoding

We can now specify properties of graphs via languages of strings
e.g.

$L_\Delta = \{ x_G :$ G (as an undirected graph) has at least one triangle
i.e. subgraph of 3 vertices with 3 edges between them.

G above is in L_Δ.

Operations on Languages:

Some are for sets in general. Others are specific to strings.

Cartesian Product: $A \times B = \{(a,b) : a \in A \;\&\; b \in B\}$

Example

$A = \{\text{"0"}, \text{"01"}, \text{"10"}\}$ $B = A$ $A \times A = \{(a,b) : a \in A \;\&\; \underline{b \in A}\}$

$A \times A = \{(0,0), (0,01), (0,10), (01,0), (01,01), (01,10), (10,0), (10,01), (10,10)\}$

Always, $|A \times B| = |A| \cdot |B|$ so $|A \times A|$ here $= 3 \times 3 = 9$.

Specific to sets of strings: Concatenation of Languages

Defn: $A \bullet B = \{x \cdot y : x \in A \;\&\; y \in B\}$.

Convention: a, b = chars or other basic objects. x, y, z, w... strings or numbers...

So $A \bullet A = \{x \cdot y : x \in A \;\&\; y \in A\}$

NOT $\{x \cdot x : x \in A\}$!

For the above A, $A \cdot A =$

$\{0\cdot0, 0\cdot01, 0\cdot10, 01\cdot0, 01\cdot01, 01\cdot10, 10\cdot0, 10\cdot01, 10\cdot10\}$. so far like $A \times A$ BUT.

$\{00, 001, \underbrace{010, 010}, 0101, 0110, 100, 1001, 1010\}$

Unlike lists, sets don't allow repeats. So as a language, it is

$= \{00, 001, 010, 0101, 0110, 100, 1001, 1010\}$ $|C| = 8$ not 9.

he lecture was also meant to include these other set operations:

nion $A \cup B = \{x : x \in A \;\underline{\text{or}}\; x \in B\}$ **Difference of sets:** $A \setminus B = \{x : x \in A \text{ but } x \notin B\}$

tersection $A \cap B = \{x : x \in A \;\underline{\text{and}}\; x \in B\}$. **Symmetric Difference:** $A \Delta B = \{x : x \in A \;\underline{\text{XOR}}\; x \in B\}$

The text uses — for difference and avoids $A \Delta B$ except to say $(A \setminus B) \cup (B \setminus A)$.