Best-Effort Program Synthesis

Hila Peleg, UCSD

Based on work with Nadia Polikarpova

Users make mistakes

Users make mistakes

Task: Extract the last name from a full name variable.

Target program: x.substr(x.indexOf(' ') + 1, x.length – (x.indexOf(' ') + 1))

Users make mistakes

Task: Extract the last name from a full name variable.

Target program: x.substr(x.indexOf(' ') + 1, x.length – (x.indexOf(' ') + 1))

User provides examples:

$$\mathcal{E}_1 \begin{cases} \{x \mapsto \text{"John Doe"}\} \rightarrow \text{"Doe"} \\ \{x \mapsto \text{"Jane Roe"}\} \rightarrow \text{"Doe"} \\ \{x \mapsto \text{"Joe Public"}\} \rightarrow \text{"Public"} \end{cases} \mathcal{E}_2$$

Users make mistakes

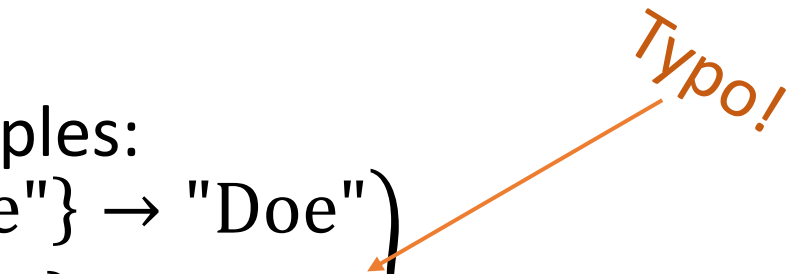
Task: Extract the last name from a full name variable.

Target program: x.substr(x.indexOf(' ') + 1, x.length – (x.indexOf(' ') + 1))

User provides examples:

$$\mathcal{E}_1 \begin{cases} \{x \mapsto \text{"John Doe"}\} \rightarrow \text{"Doe"} \\ \{x \mapsto \text{"Jane Roe"}\} \rightarrow \text{"Doe"} \end{cases} \mathcal{E}_2$$

$$\{x \mapsto \text{"Joe Public"}\} \rightarrow \text{"Public"}$$



Users make mistakes

Task: Extract the last name from a full name variable.

Target program: x.substr(x.indexOf(' ') + 1, x.length – (x.indexOf(' ') + 1))

User provides examples:

$$\mathcal{E}_1 \begin{cases} \{x \mapsto \text{"John Doe"}\} \rightarrow \text{"Doe"} \\ \{x \mapsto \text{"Jane Roe"}\} \rightarrow \text{"Doe"} \\ \{x \mapsto \text{"Joe Public"}\} \rightarrow \text{"Public"} \end{cases} \mathcal{E}_2$$

Typo!

Why don't we try every $\mathcal{E}' \in \mathcal{P}(\mathcal{E}_2)$?

What happens in an OE enumeration

$i_0 = \{x \mapsto$"John Doe"$\}$
$i_1 = \{x \mapsto$"Jane Roe"$\}$

Enumeration

Equivalence classes

What happens in an OE enumeration

$i_0 = \{x \mapsto$"John Doe"$\}$
$i_1 = \{x \mapsto$"Jane Roe"$\}$

Enumeration

Programs being constructed

Equivalence classes

Values we've already seen

What happens in an OE enumeration

$i_0 = \{x \mapsto$"John Doe"$\}$
$i_1 = \{x \mapsto$"Jane Roe"$\}$

Enumeration

h=0: 0, 1, ' ', x

Equivalence classes

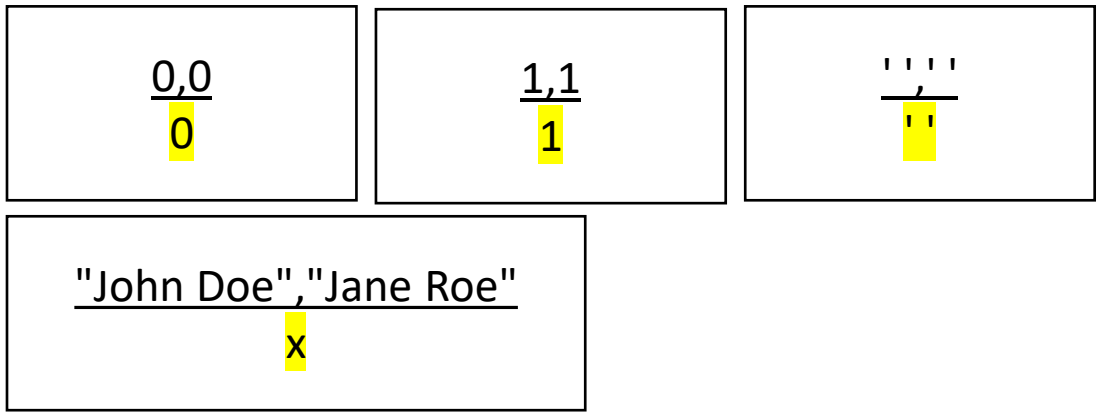
What happens in an OE enumeration

$i_0 = \{x \mapsto$"John Doe"$\}$
$i_1 = \{x \mapsto$"Jane Roe"$\}$

Enumeration

h=0: 0, 1, ' ', x

Equivalence classes



What happens in an OE enumeration

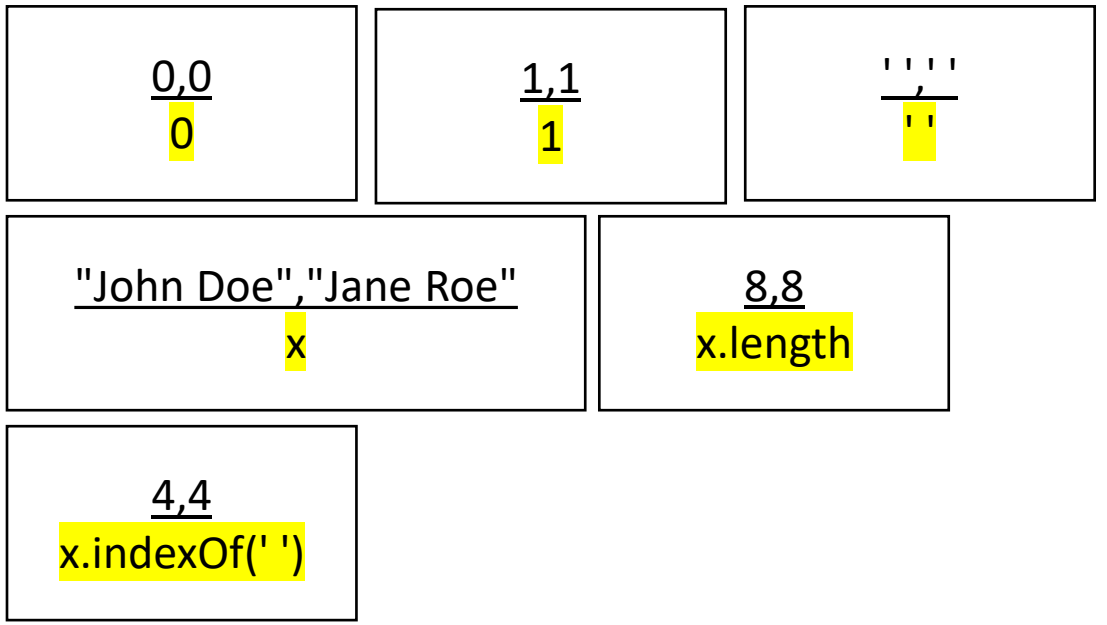
$i_0 = \{x \mapsto$"John Doe"$\}$
$i_1 = \{x \mapsto$"Jane Roe"$\}$

Enumeration

h=0: 0, 1, ' ', x

h=1: x.length, x.indexOf(' '), …

Equivalence classes



What happens in an OE enumeration

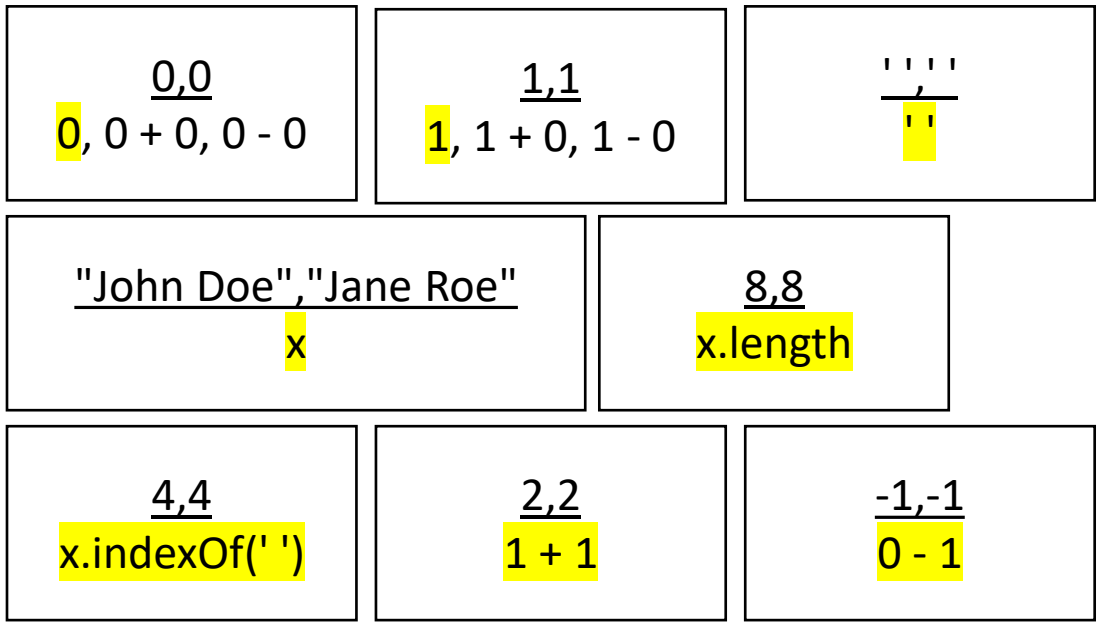
$i_0 = \{x \mapsto$"John Doe"$\}$
$i_1 = \{x \mapsto$"Jane Roe"$\}$

Enumeration

h=0: 0, 1, ' ', x

h=1: x.length, x.indexOf(' '), …
0+0, 1+0, 0+1, 1+1, 0-0, 0-1, 1-0,…

Equivalence classes



What happens in an OE enumeration

$i_0 = \{x \mapsto \text{"John Doe"}\}$
$i_1 = \{x \mapsto \text{"Jane Roe"}\}$

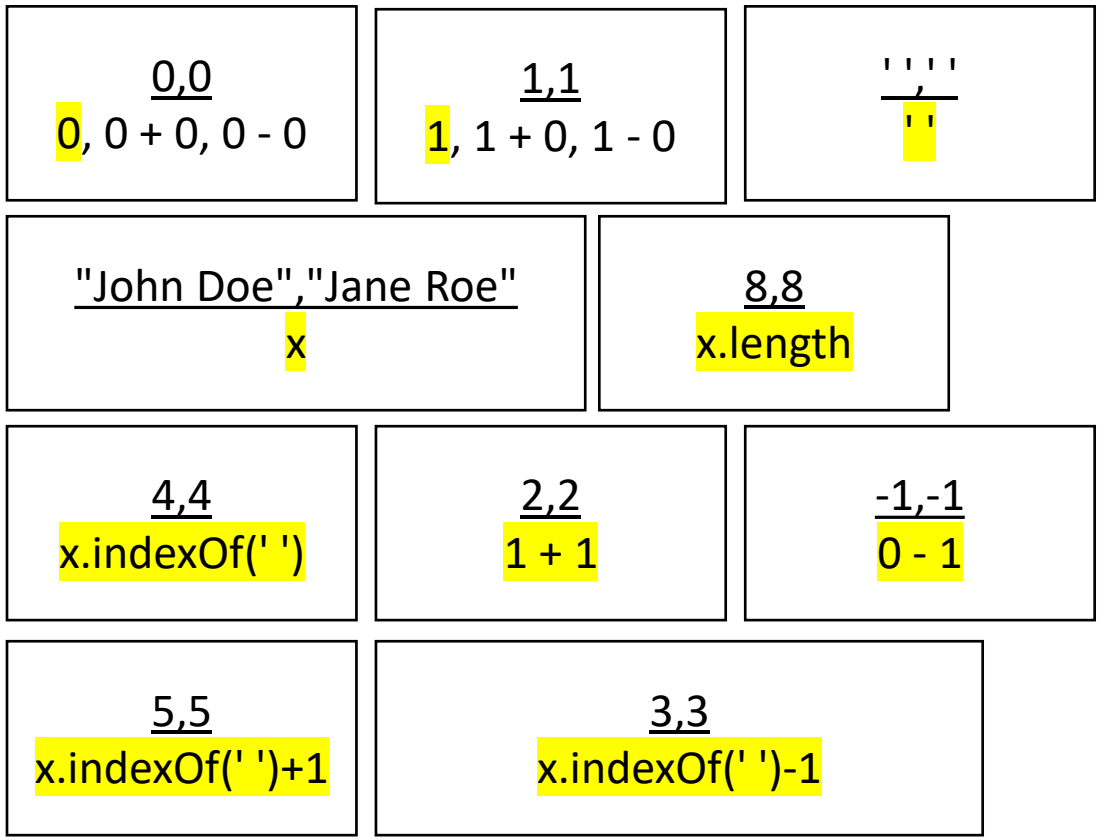
Enumeration

h=0: 0, 1, ' ', x

h=1: x.length, x.indexOf(' '), …
0+0, 1+0, 0+1, 1+1, 0-0, 0-1, 1-0,…

h=2: x.indexOf(' ')+1,
x.indexOf(' ')-1

Equivalence classes



What happens in an OE enumeration

$i_0 = \{x \mapsto \text{"John Doe"}\}$
$i_1 = \{x \mapsto \text{"Jane Roe"}\}$

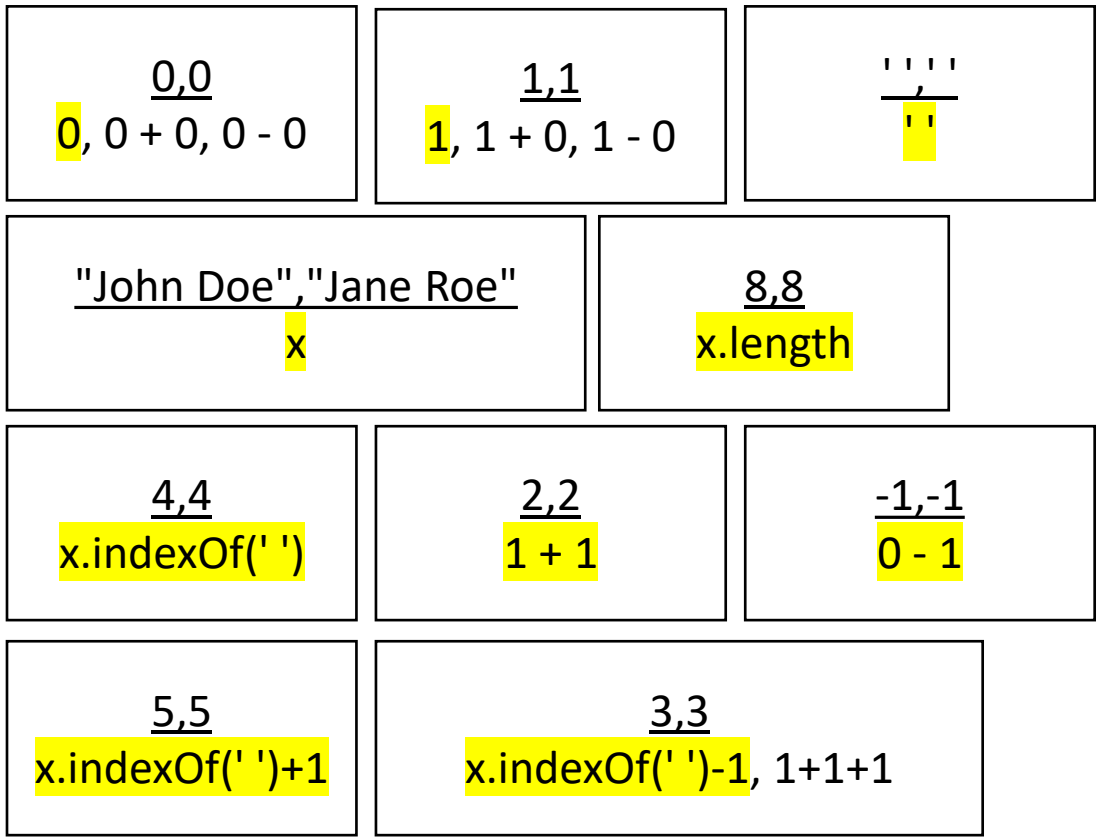
Enumeration

h=0: 0, 1, ' ', x

h=1: x.length, x.indexOf(' '), …
0+0, 1+0, 0+1, 1+1, 0-0, 0-1, 1-0,…

h=2: x.indexOf(' ')+1,
x.indexOf(' ')-1, 1+1+1,…

Equivalence classes



What happens in an OE enumeration

$i_0 = \{x \mapsto$"John Doe"$\}$
$i_1 = \{x \mapsto$"Jane Roe"$\}$

Enumeration

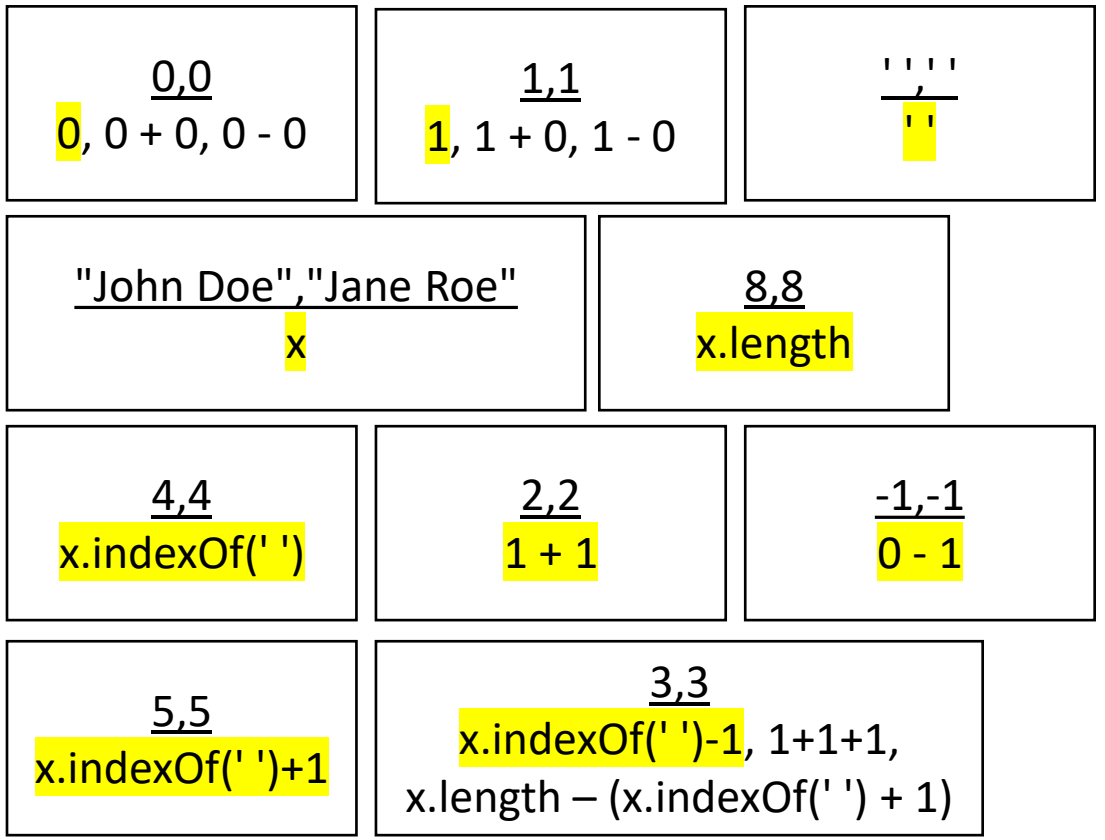
h=0: 0, 1, ' ', x

h=1: x.length, x.indexOf(' '), …
0+0, 1+0, 0+1, 1+1, 0-0, 0-1, 1-0,…

h=2: x.indexOf(' ')+1,
x.indexOf(' ')-1, 1+1+1,…

h=3: x.length – (x.indexOf(' ') + 1)

Equivalence classes



A problematic equivalence class

3,3

```
x.indexOf(' ')-1
1+1+1
x.length-(x.indexOf(' ')+1)
```

A problematic equivalence class

Refine with additional example: $\{x \mapsto \text{"Joe Public"}\} \rightarrow \text{"Public"}$

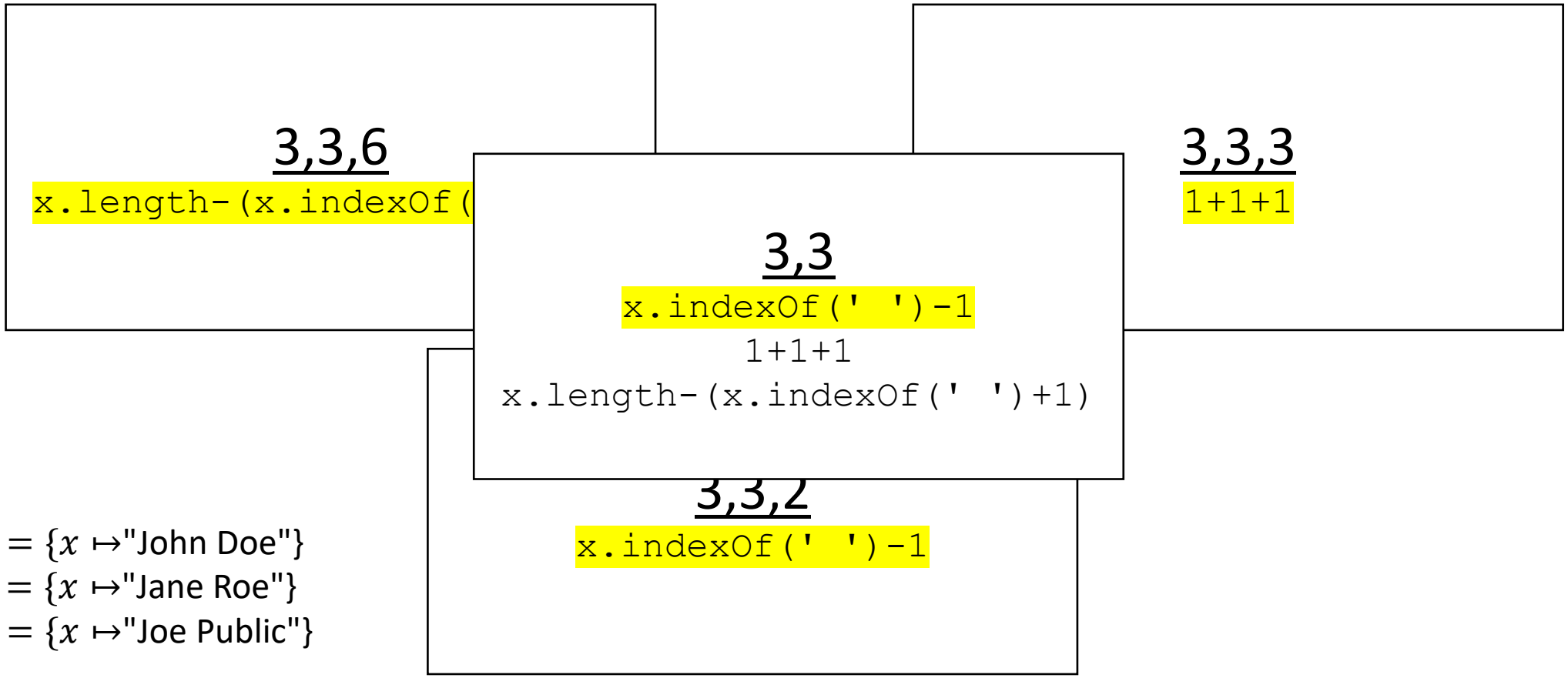
3,3

```
x.indexOf(' ')-1
```

```
1+1+1
```

```
x.length-(x.indexOf(' ')+1)
```

Refining equivalence from $\mathcal{E}_1$ to $\mathcal{E}_2$



$i_0 = \{x \mapsto$"John Doe"$\}$
$i_1 = \{x \mapsto$"Jane Roe"$\}$
$i_2 = \{x \mapsto$"Joe Public"$\}$

Looking for $\mathcal{P}(\mathcal{E}_2)$ on the equivalence classes

$o_0 =$ "Doe"
$o_1 =$ "Doe"
$o_2 =$ "Public"

Looking for $\mathcal{P}(\mathcal{E}_2)$ on the equivalence classes

$o_0 =$ "Doe"
$o_1 =$ "Doe"
$o_2 =$ "Public"

"Doe", "Roe", "Pub"

```
x.subsr(
 x.indexOf(' ')+1 ,
 1+1+1)
```

Looking for $\mathcal{P}(\mathcal{E}_2)$ on the equivalence classes

$o_0 =$ "Doe"
$o_1 =$ "Doe"
$o_2 =$ "Public"

✓

"Doe", "Roe", "Pub"

```
x.subsr(
 x.indexOf(' ')+1 ,
 1+1+1)
```

Looking for $\mathcal{P}(\mathcal{E}_2)$ on the equivalence classes

$o_0 =$ "Doe"
$o_1 =$ "Doe"
$o_2 =$ "Public"

✓

"Doe", "Roe", "Pub"

```
x.subsr(
 x.indexOf(' ')+1 ,
 1+1+1)
```

...

"Doe", "Roe", "Public"

```
x.subsr(
 x.indexOf(' ')+1 ,
 x.length-(x.indexOf(' ')+1))
```

Looking for $\mathcal{P}(\mathcal{E}_2)$ on the equivalence classes

$o_0 =$ "Doe"
$o_1 =$ "Doe"
$o_2 =$ "Public"

✓

"Doe", "Roe", "Pub"

```
x.subsr(
 x.indexOf(' ')+1 ,
 1+1+1)
```

...

✓ ✓

"Doe", "Roe", "Public"

```
x.subsr(
 x.indexOf(' ')+1 ,
 x.length-(x.indexOf(' ')+1))
```

Summary

What I talked about

- If the spec has errors we want to satisfy as much of it as we can
- We want to synthesize every subset of the examples
- In a bottom-up observational equivalence enumeration, we get that for free

Summary

What I talked about

- If the spec has errors we want to satisfy as much of it as we can
- We want to synthesize every subset of the examples
- In a bottom-up observational equivalence enumeration, we get that for free

What I didn't talk about

- Deciding between multiple programs that satisfy the same amount of examples
- The connection between errors in the spec and hard problems
- Evaluation