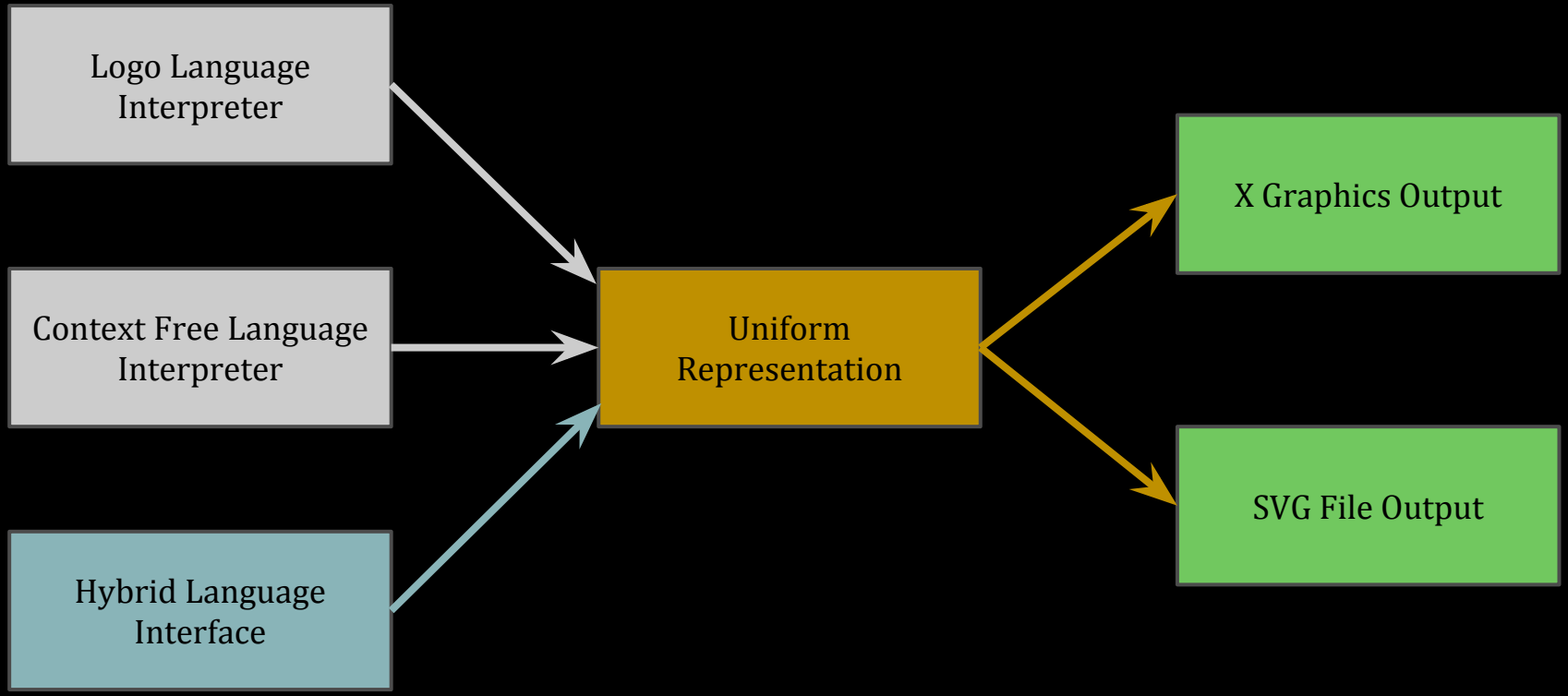
Unifying the Best Features of Graphics Languages

Jake Barnwell & Miles Steele

6.905/6.945 Large-scale Symbolic Systems

6 May 2015

System Design



Uniform Representation (UR)

List of Instructions:

```
(instruction-1
 instruction-2
 …
 instruction-n)
```

Instruction Primitives:

```
(line x1 y1 x2 y2)
(point x y)
(color “color-name”)
```

Example of a UR:

```
((color “red”)
 (line 0 0 0 0)
 (line 1 1 0.5 0.5)
 (line 1 1 -0.5 -0.5)
 (color “blue”)
 (point -1 -1)
 (point -1 0.5)
 ... )
```

Logo Language

Primitives

```
(forward 10) (rotate 90) (pen-up) (pen-down) (color "blue")
```

Repetition

```
(repeat 4 statements...)
```

Procedure Call

```
(square 100)
```

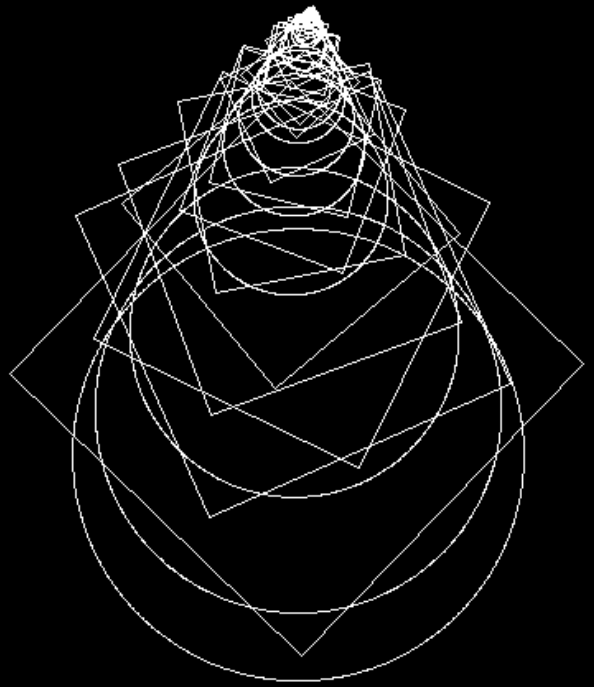
Procedure Definition

```
(to (square size)
  (repeat 4
    (fd size)
    (rt 90)))
```

Logo Language

(demo)

Context Free Language



Example Usage:

```
(ctxf '(
    (startshape r)
    (shape r
        (rule (
                (square (dr (random 90)))
                (r (dr -2 y 0.1 s 0.9 0.9))))
        (rule (
                (circle (s 0.9 0.9))
                (r (dr 2  y 0.1 s 0.9 0.9))))
        )))
```

Transformation Stack

```
(ctxf '(
  (startshape S (x 5))
  (shape S ( (square (y 1))
             (S (dr 5 s 0.5 0.5))
           ))))
```

```
(square (x 5 y 1)
(square (x 5 dr 5 s 0.5 0.5 y 1)
...
(square (x 5 dr 5 s 0.5 0.5 ... y 1)
when too-small? HALT
```

Translation:

$$\begin{pmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{pmatrix}$$

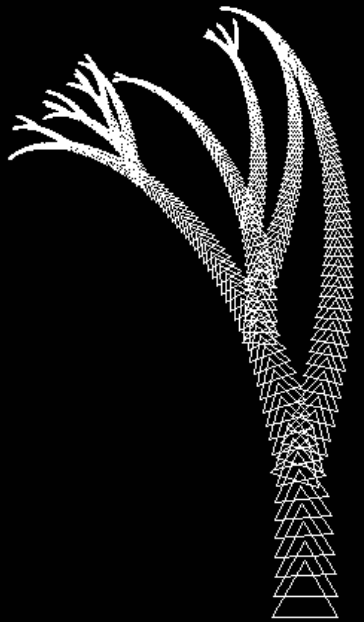
Scaling:

$$\begin{pmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Rotation:

$$\begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Context Free Language



```
(ctxf '(
  (startshape branch)
  (shape branch
    (rule 7 (
      (branch (dr 5 s 0.95 0.95))
      (branch (dr -25 s 0.95 0.95))))
    (rule 99 (
      (stem ())
      (branch (dr 1 y 0.13 s 0.92 0.92)))))
  (shape stem (
    (TRIANGLE (s 0.4 0.4))))))
```

Hybrid Language

Primitives

```
(line! x1 y1 x2 y2)
(color! "green")
```

Recursion

```
(guard thunk)
(repeat times thunk)
```

Transforms

```
(save-excursion thunk)
(translate dx dy [thunk])
(rotate deg [thunk])
(scale x [y [thunk]])
(flip deg [thunk])
```

Hybrid Language

(demo)

End