

Ragged Blocks: Rendering Structured Text With Style



Sam Cohen
PAT Seminar
Friday, Nov 21, 2024

Design Space for Structured Visualizations

Where can structure be expressed?

Horizontally and Vertically

Horizontally

Nowhere

Sandblocks^{1†}

```
if expression :
  pass
```

Hass² (Prior Work)

```
length
. filter (isPrefixOf "--")
. lines
```

Rocks (This System)

```
length
. filter (not .isPrefixOf "--")
. lines
```

Fructure^{3†}

```
define exs ess
begin if zero? ●
```

```
2 + 3 4
```

Tylr^{4†}



Text Selection

```
quam erat volutpat. Vest:
ic mauris sem, imperdiet
ipit orci. Donec ultrici
```

Deuce^{5†}

```
(def rectangle
  (rect fill x y w h))
```

What is the layout primitive?

Boxes

S-Blocks

Ragged Blocks (Rocks)

¹Tom Beckmann, Patrick Rein, Stefan Ramson, Joana Bergsiek, and Robert Hirschfeld. 2023. Structured Editing for All: Deriving Usable Structured Editors from Grammars.

²Cohen and Chugh

³Andrew Blinn, 2019. Fructure: A Structured Editing Engine in Racket

⁴D. Moon, A. Blinn and C. Omar. Gradual Structure Editing with Obligations.

⁵Brian Hempel, Justin Lubin, Grace Lu, and Ravi Chugh. 2018. Deuce: a Lightweight User Interface for Structured Editing.

†These images were modified from their original publication to use a consistent font and line width.

A Working Example (S-Blocks)

Input

```
-- Anonymous Authors
module Main where

-- Print a greeting
main =
  putStrLn "Hello, OOPSLA!"
```

a *Syntax Highlighting Style Sheet*

```
main =
  getContents
  >>= print
    . length
    . filter $ isPrefixOf "--"
    . lines
```

b *Type Error Style Sheet*

```
main =
  getContents
  >>= print
    . length
    . filter $ isPrefixOf "--"
    . lines
```

c *Blocks Style Sheet*

```
main =
  getContents
  >>= print
    . length
    . filter $ isPrefixOf "--"
    . lines
```

d *Blocks Style Sheet*

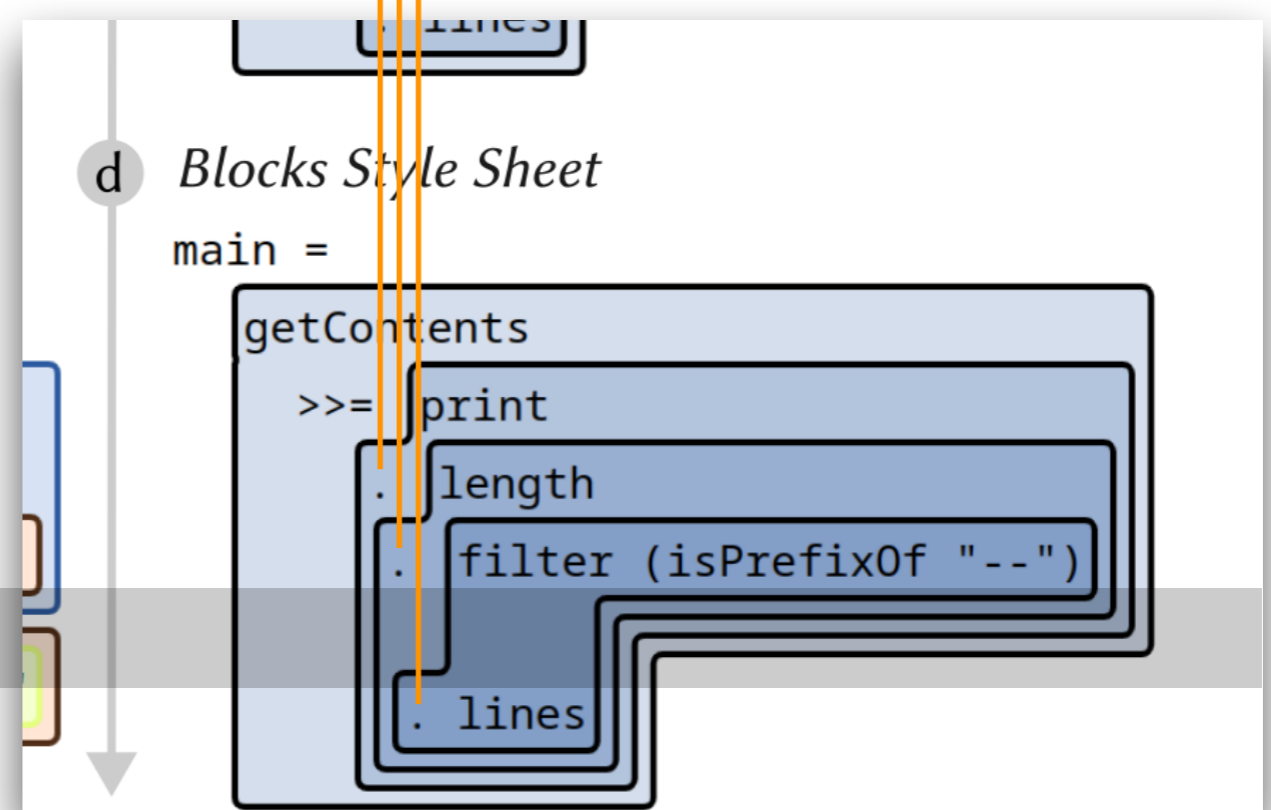
```
main =
  getContents
  >>= print
    . length
    . filter (isPrefixOf "--")
    . lines
```

A Working Example (S-Blocks)

- ✓ Text layout looks “natural”
- ✓ Structure is clear
- ✓ Polygon outlines are simple
- ✗ Column alignment is destroyed
- ✗ Not very compact

Columns no longer aligned

Wasted Vertical Space

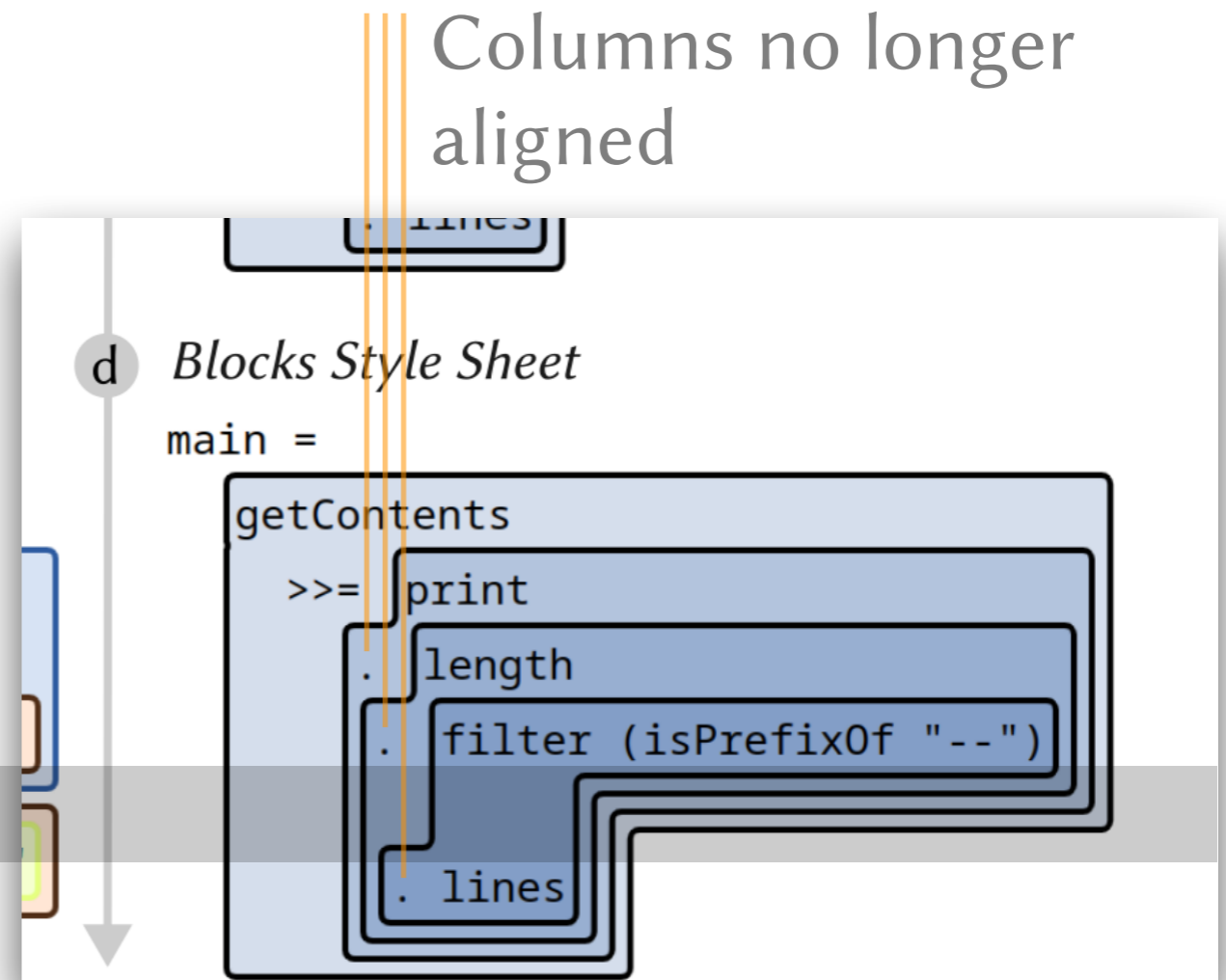


A Working Example (S-Blocks)

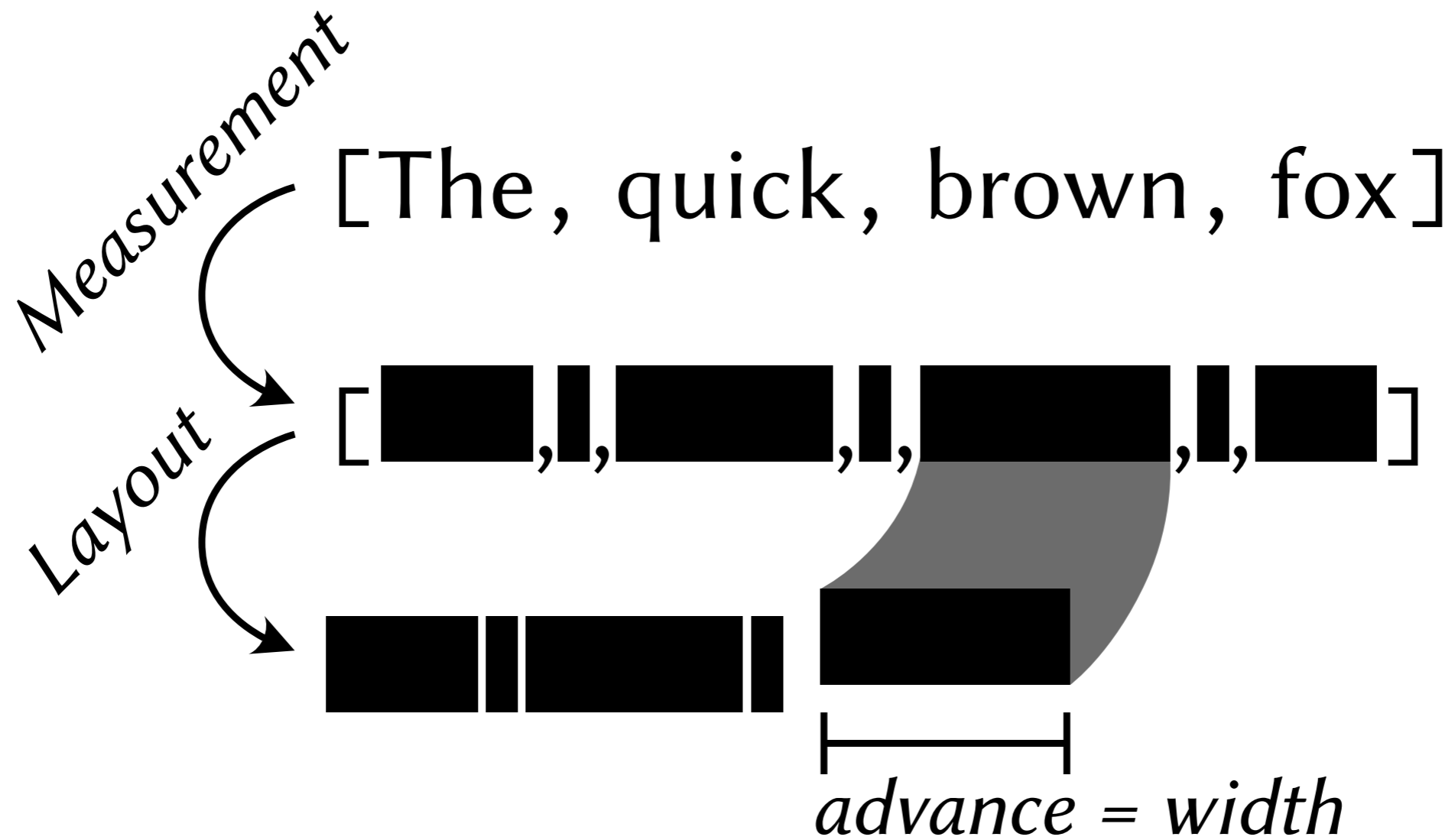
- ✓ Text layout looks “natural”
- ✓ Structure is clear
- ✓ Polygon outlines are simple
- ✗ Column alignment is destroyed
- ✗ Not very compact

Columns no longer aligned

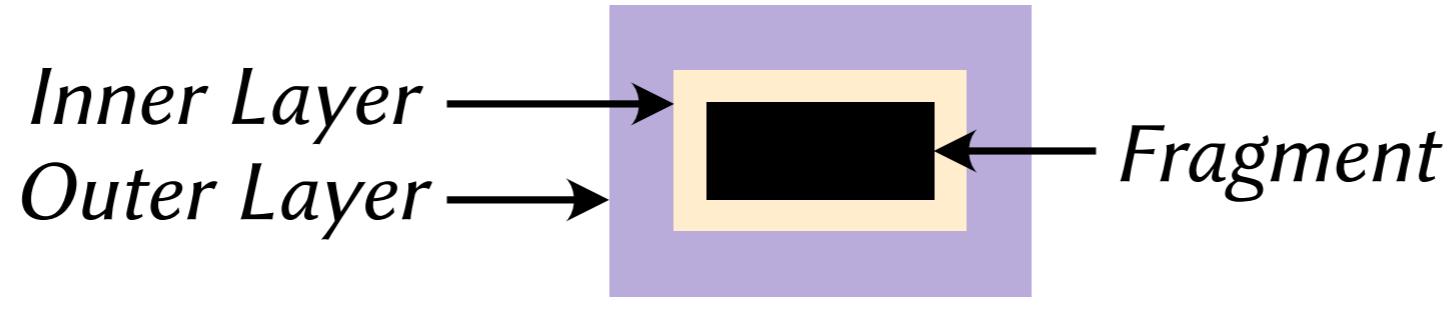
Wasted Vertical Space



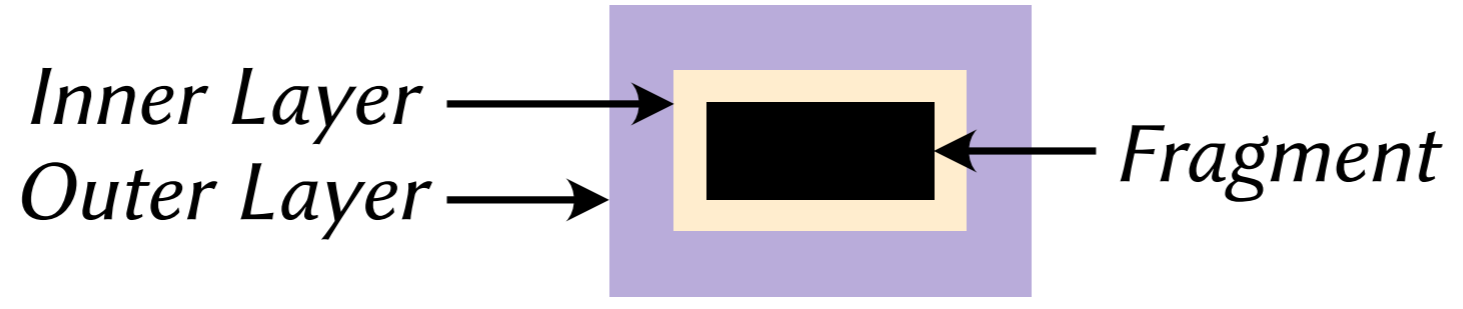
How Does Text Layout Normally Work?



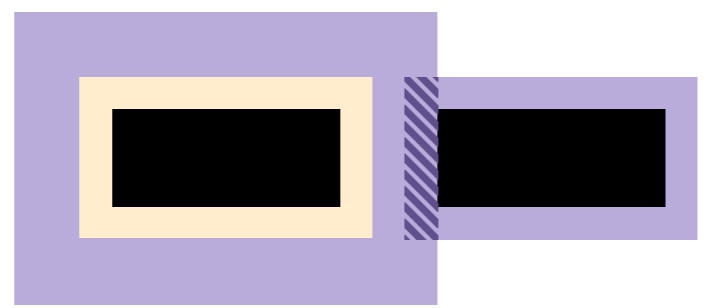
Big Idea: Regions



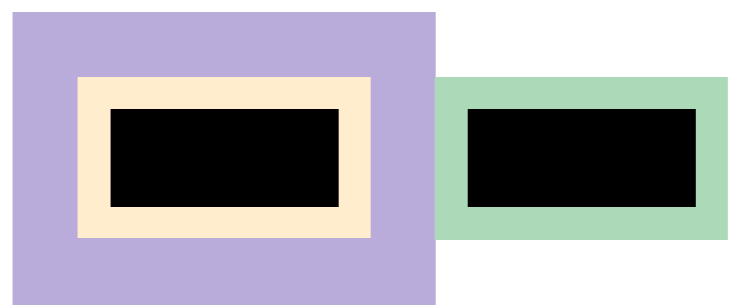
Big Idea: Regions



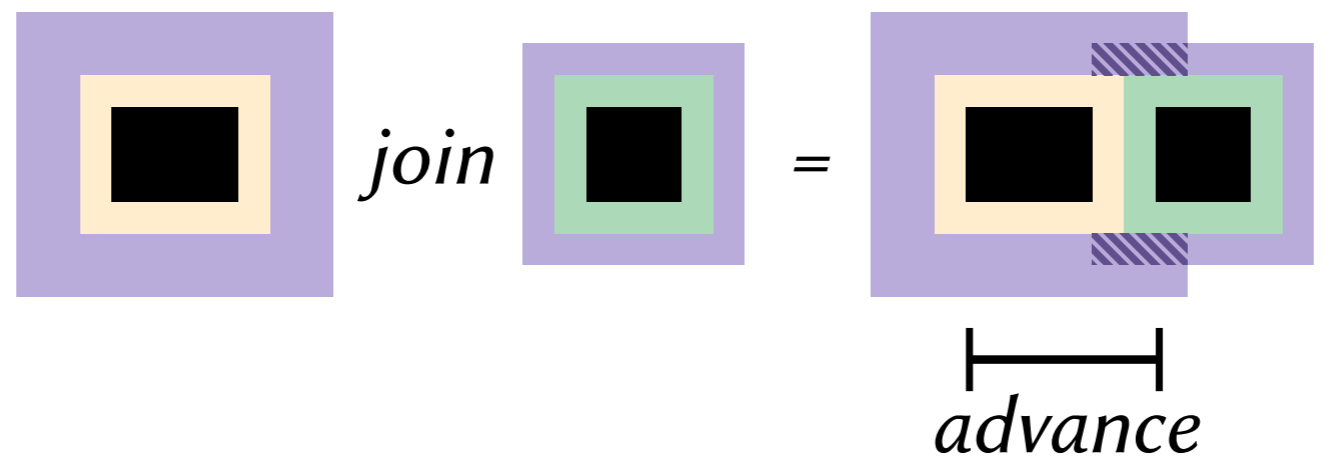
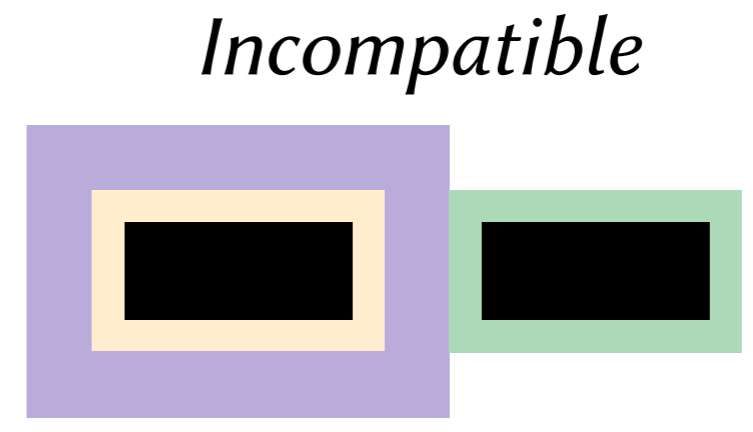
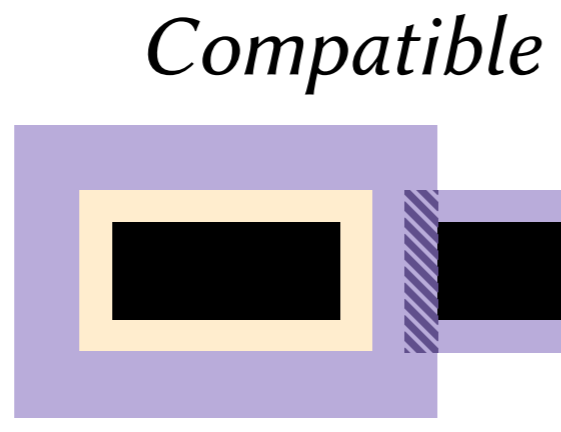
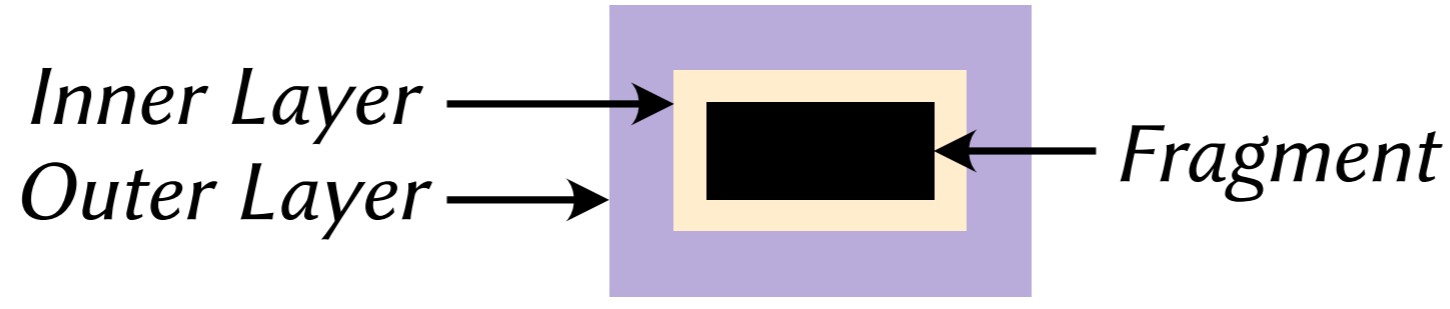
Compatible



Incompatible



Big Idea: Regions



Using Regions to Implement Layout

```
fact 0 = 1
```

```
fact n = n * fact (n - 1)
```

Using Regions to Implement Layout

```
fact 0 = 1
```

```
fact n = n * fact (n - 1)
```

```
[ [ fact 0 = 1 ]
```

```
, [ fact n = , n * , fact , n - 1 ] ]
```

Using Regions to Implement Layout

```
fact 0 = 1
```

```
fact n = n * fact (n - 1)
```

```
[ [ fact 0 = 1 ]
```

```
, [ fact n = , n * , fact , n - 1 ] ]
```

```
[ fact 0 = 1
```

```
, fact n = n * fact n - 1 ]
```

Using Regions to Implement Layout

```
fact 0 = 1
```

```
fact n = n * fact (n - 1)
```

```
[ [ fact 0 = 1 ]
```

```
, [ fact n = , n * , fact , n - 1 ] ]
```

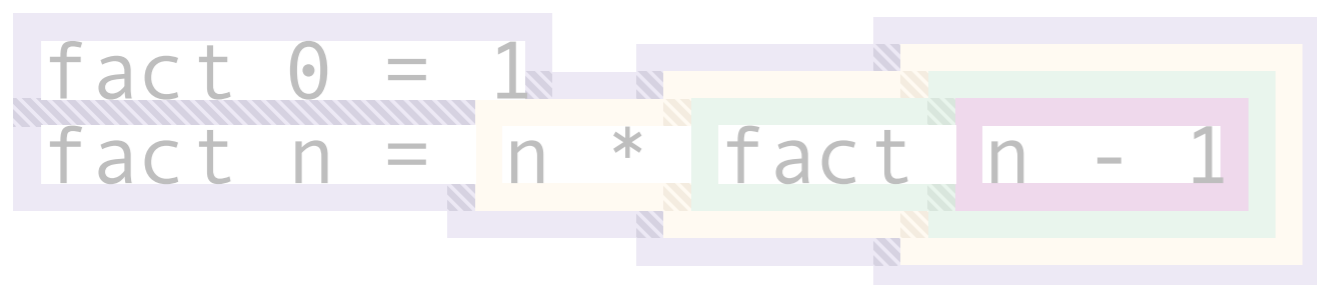
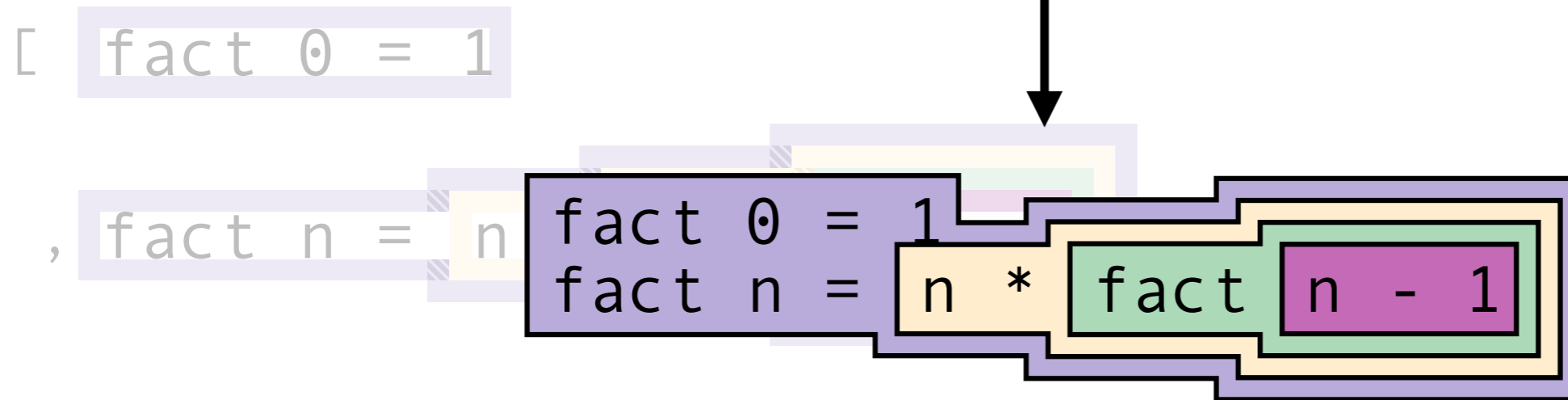
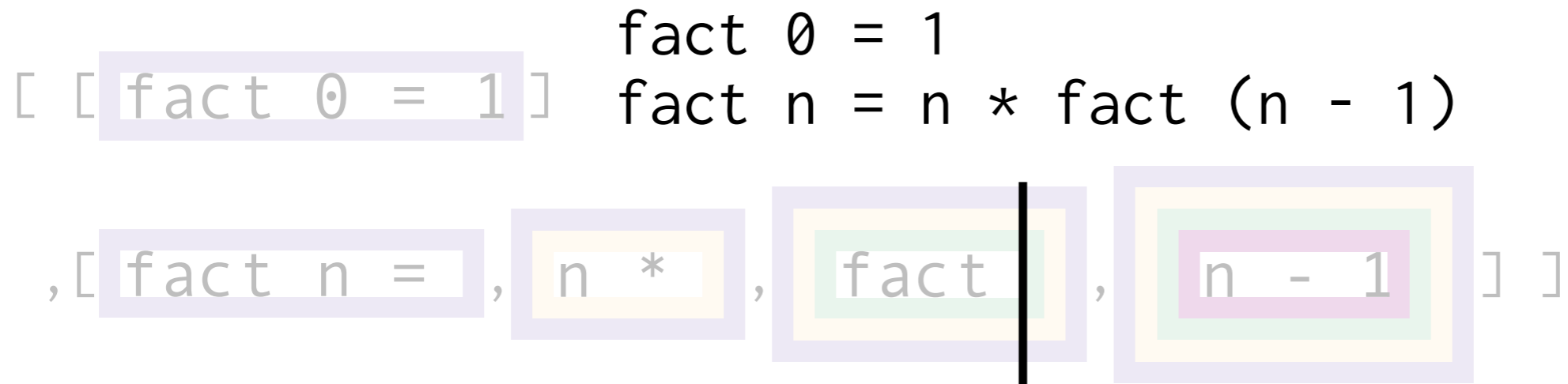
```
[ fact 0 = 1
```

```
, fact n = n * fact n - 1 ]
```

```
fact 0 = 1  
fact n = n * fact n - 1
```

Using Regions to Implement Layout

```
fact 0 = 1  
fact n = n * fact (n - 1)
```



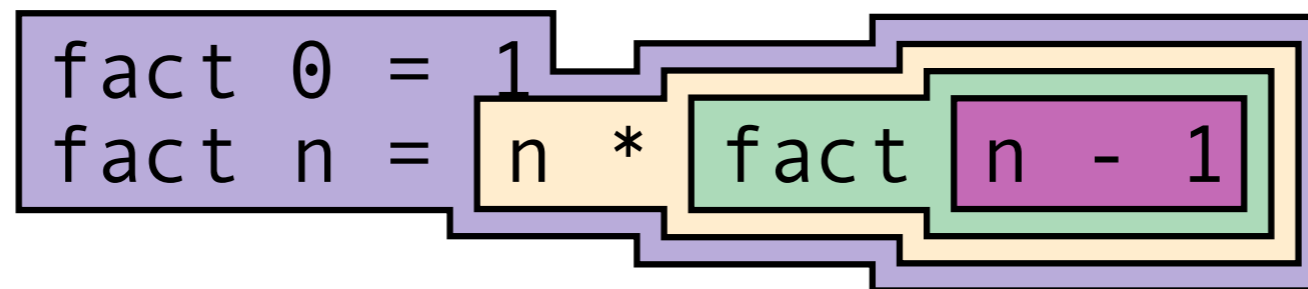
✓ Text layout looks “natural”

✓ Structure is clear

✗ Polygon outlines are simple

✗ Columns can be aligned (you’ll have to take my word for it)

✓ Compact!



Simplification

```
fact 0 = 1
fact n = n * fact (n - 1)
```

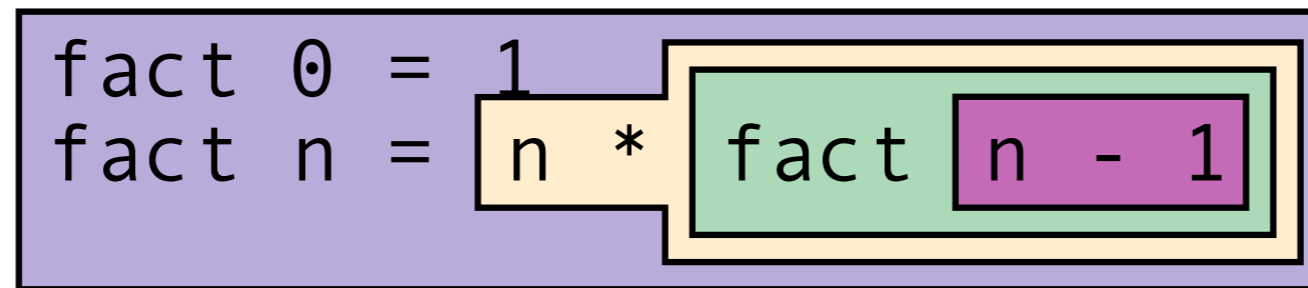
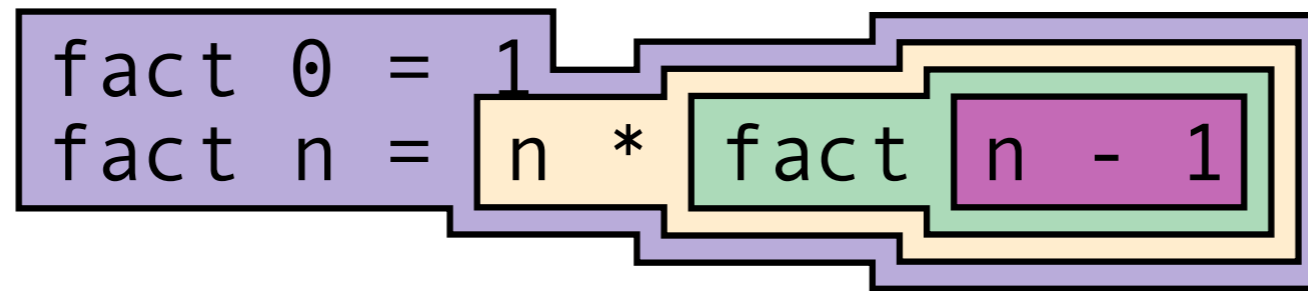
The diagram shows the initial recursive definition of factorial. The first line is `fact 0 = 1` and the second line is `fact n = n * fact (n - 1)`. The text is contained within a purple box. The `1` in the first line is highlighted in a yellow box. The `n *` in the second line is highlighted in a yellow box. The `fact (n - 1)` in the second line is highlighted in a green box. The `n - 1` in the second line is highlighted in a purple box. The entire structure is enclosed in a larger purple box with a stepped right edge.



```
fact 0 = 1
fact n = n * fact (n - 1)
```

The diagram shows the simplified recursive definition of factorial. The first line is `fact 0 = 1` and the second line is `fact n = n * fact (n - 1)`. The text is contained within a single purple box. The `1` in the first line is highlighted in a yellow box. The `n *` in the second line is highlighted in a yellow box. The `fact (n - 1)` in the second line is highlighted in a green box. The `n - 1` in the second line is highlighted in a purple box. The entire structure is enclosed in a single purple box.

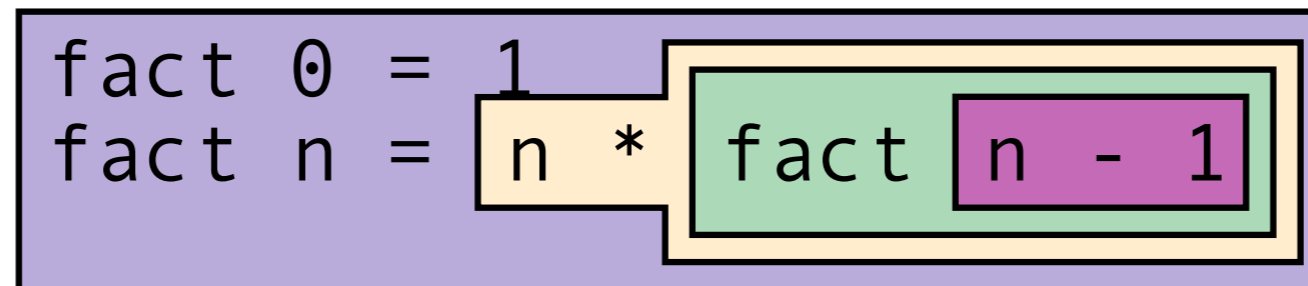
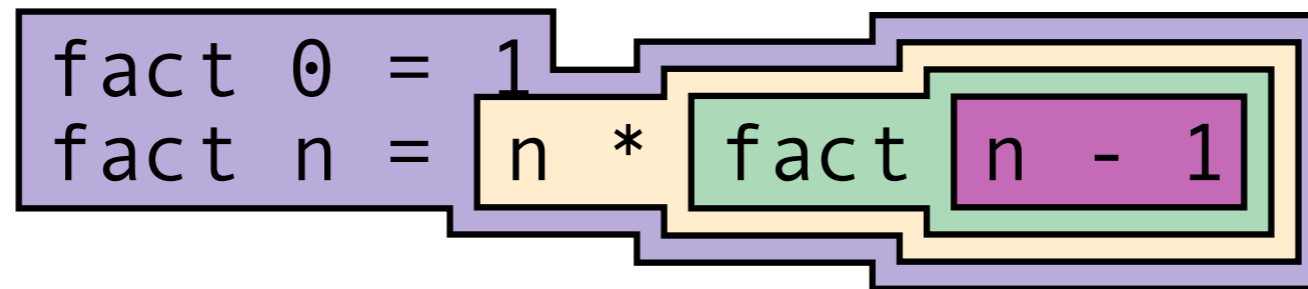
Simplification



*Still a lot to think about in this area.

Simplification

Thank You!



*Still a lot to think about in this area.