Walking through an example with the stencil kernel SEJITS specializer

**Stencil example: input**

```python
from stencil_kernel import *

class ExampleKernel(StenciKernel):
    def kernel(self, in_grid, out_grid):
        for x in in_grid.interior_points():
            for y in in_grid.neighbors(x, 1):
                out_grid[x] = out_grid[x] + in_grid[y]

in_grid = StenciGrid((5,5))
in_grid.data = numpy.ones((5,5))
out_grid = StenciGrid((5,5))
ExampleKernel().kernel(in_grid, out_grid)
```

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 4 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**3. Bind platform**

- Autotuning and other platform-specific optimizations
- Optimizations can be reused across “similar” platforms (e.g., emit a generic SIMD representation, let compiler map final instruction set)

**Opportunities for improvement:**
- Target new platforms (multicore, GPU, cloud)
- New platform-specific optimizations

**Summary**

**App author (PLL)**
- Specializer author (ELL)
- SEJITS team
- 3rd party library

**Platform-Specific**
- CodePy
- Templates

**Low-level language source code**
- Machine code

**Try SEJITS yourself at:**
http://aspsejits.pbworks.com

**2. Optimize Domain AST**

- Optimizations are platform-independent
- Optimizations can exploit domain knowledge

**Opportunities for improvement:**
- New domain-specific optimizations

**4. Emit code**