Game of Life in Concrete Python

Benoit Chevallier-Mames, Umut Şahin

Concrete Framework

Write code in pure Python, that you can compile to its FHE equivalent, without taking care of the cryptography

Game of Life Rules

1. active cell with 2 or 3 active neighbors: it survives
2. inactive cell with 3 active neighbors: the cell borns
3. else, the cell is dead

Basic Implementation

```python
def update_grid_basic(grid):
    # Method which follows the naive approach
    weights_method_basic = np.array(
        [
            [1, 1, 1],
            [1, 0, 1],
            [1, 1, 1],
        ]
    )
    table_next_cell_basic_a = [int(i in [3]) for i in range(9)]
    table_next_cell_basic_b = [int(i in [2, 3]) for i in range(9)]

    table_cp_next_cell_basic_a = fhe.LookupTable(table_next_cell_basic_a)
    table_cp_next_cell_basic_b = fhe.LookupTable(table_next_cell_basic_b)

    # This is to workaround the fact that we have no pad option in fhe.conv
    do_padded_fix = True

    convoluted_grid = conv_with_hand_padding(grid, weights_method_basic,
                                             do_padded_fix)

    grid = table_cp_next_cell_basic_a[convoluted_grid] | (
        table_cp_next_cell_basic_b[convoluted_grid] & (grid == 1)
    )

    return grid
```

Open Source Code

Start building.

5b-PBS implementation

```python
def update_grid_method_5b(grid):
    # Method which uses a single TLU of 5 bits
    weights_method_5b = np.array(
        [
            [1, 1, 1],
            [1, 9, 1],
            [1, 1, 1],
        ]
    )
    table_next_cell_5b = [int(i in [3, 9 + 2, 9 + 3]) for i in range(18)]

    table_cp_next_cell_5b = fhe.LookupTable(table_next_cell_5b)

    # This is to workaround the fact that we have no pad option in fhe.conv
    do_padded_fix = True

    convoluted_grid = conv_with_hand_padding(grid, weights_method_5b,
                                             do_padded_fix)

    grid = table_cp_next_cell_5b[convoluted_grid]

    return grid
```

4b-PBS implementation

```python
def update_grid_method_4b(grid):
    # Method which uses a first TLU of 4 bits and a second TLU of 2 bits
    weights_method_4b = np.array(
        [
            [1, 1, 1],
            [1, 0, 1],
            [1, 1, 1],
        ]
    )
    table_next_cell_4b_a = [i - 1 if i in [2, 3] else 0 for i in range(9)]
    table_next_cell_4b_b = [int(i in [2, 3]) for i in range(4)]

    table_cp_next_cell_4b_a = fhe.LookupTable(table_next_cell_4b_a)
    table_cp_next_cell_4b_b = fhe.LookupTable(table_next_cell_4b_b)

    # This is to workaround the fact that we have no pad option in fhe.conv
    do_padded_fix = True

    convoluted_grid = conv_with_hand_padding(grid, weights_method_4b,
                                             do_padded_fix)

    grid_a = table_cp_next_cell_4b_a[convoluted_grid]
    grid = grid_a + grid
    grid = table_cp_next_cell_4b_b[grid]

    return grid
```

Results

On an m6i AWS machine: you can reproduce yourself!

	update_grid_basic	update_grid_3b	update_grid_4b	update_grid_5b
PBS count	three 4b + two 2b	two 3b + one 2b	one 4b + one 2b	one 5b
Execution time for a (6, 6) grid	0.25 s[a]	**0.12 s**	0.18 s	0.18 s
Execution time for a (200, 200) grid	65 s[a]	35 s	**32 s**	38 s

ZAMA

Resources

zama.ai/blog
github.com/zama-ai/concrete