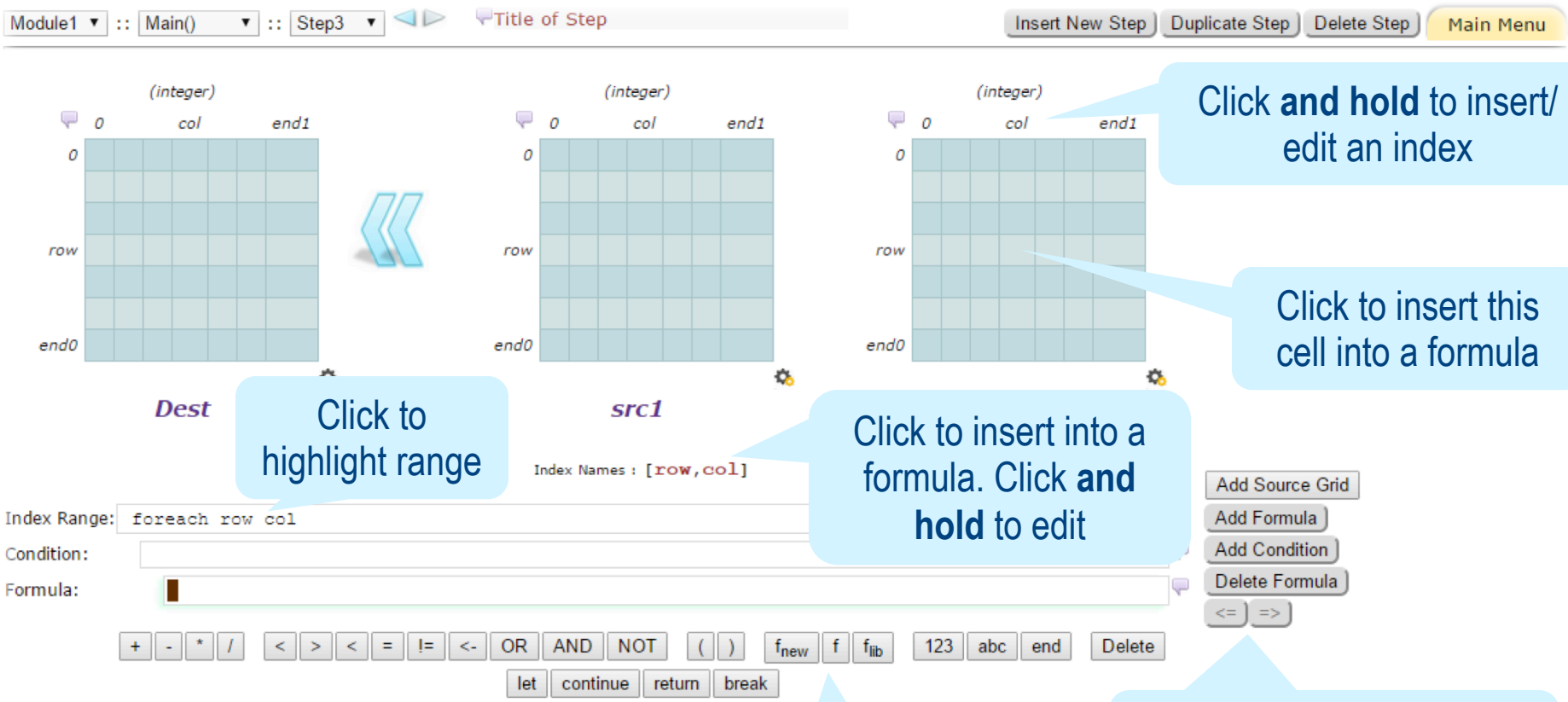
GLAF

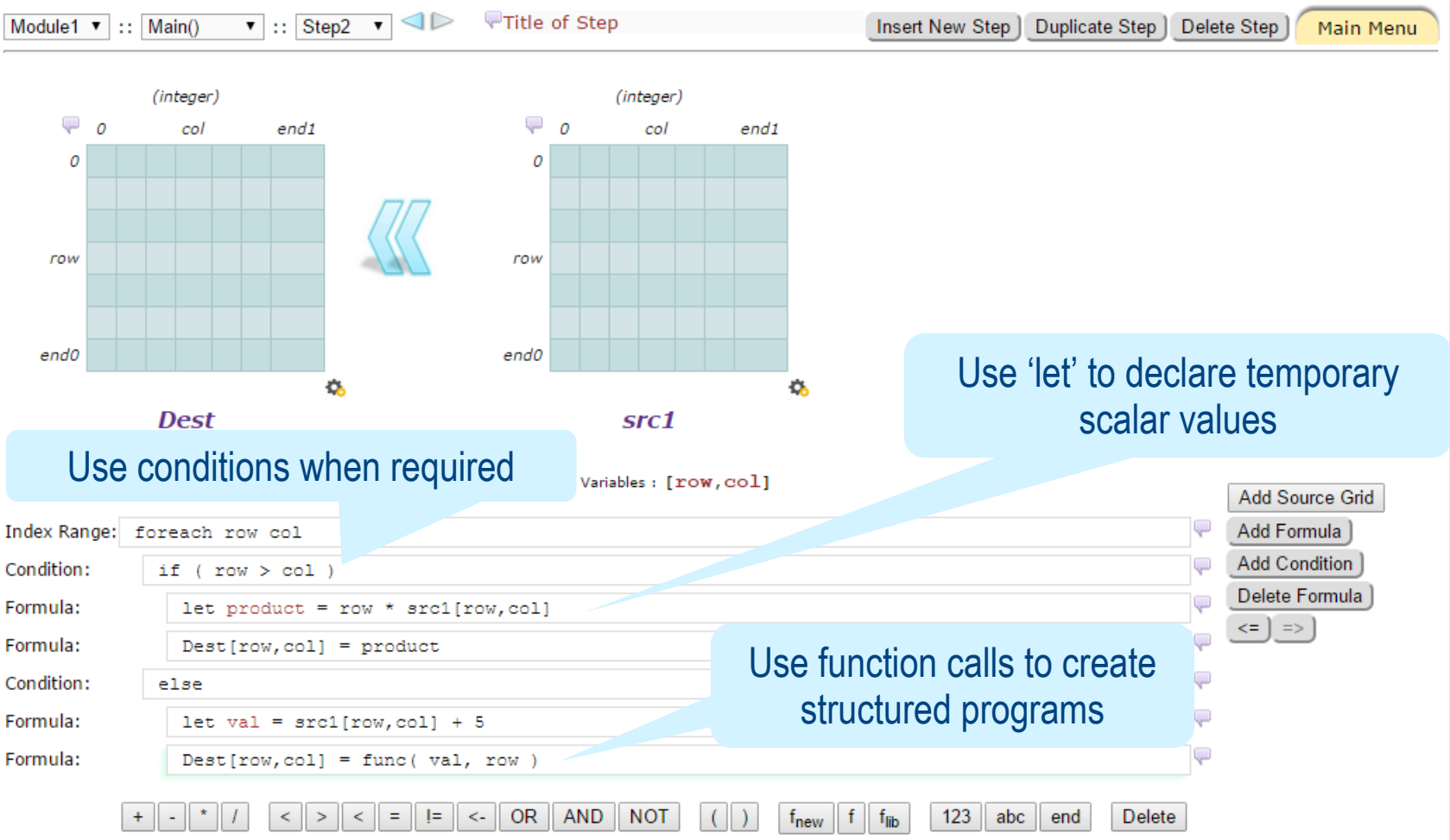
Quick Guide and Best Practices:
Writing Automatically Parallelizable Programs

Ruchira Sasanka

Getting Started: Mouse Actions



Structured Programming Example



Fill Output Grid Using Read-Only Source Grid(s)

- **Organize a step so that we …**
  - Fill an output grid using read-only source grid(s)
    - i.e., do **not** read and write to the same grid
  - Examples:

```
Out[row] = src1[row] + src2[row];
```

```
Out[row] = src1[row] * myFunc(src2, row);
```

Use function calls to create structured programs.

Simplest Rule to Follow to Obtain Parallelism

Avoid Read + Writes to the Same Grid

- **If reads AND writes are necessary to the same grid**
  - Consider using two steps
    - (i) Step 1: Read and calculate a temporary grid
    - (ii) Step 2: Write to the output grid using temporary grid

Avoid Writing to Multiple Grid Locations in Same Step

- **Don't write to multiple grid locations, like:**

```
foreach row
    Out[row] = 5;
    Out[10] = 12;
```

- **Instead separate out to two steps** *OR…*
- **Use an if-condition:**
  - E.g.,

```
foreach row
    if (row == 10)
        Out[row] = 12;
    else
        Out[row] = 5;
```

Always write to the same location, Out[row]

Minimize Passing Writable Grids to Functions

- **If only a single location is written, don't pass an entire grid**
  - E.g., Instead of func(out, in1, in2, row), which writes to out[row]
    - use:

```
foreach row
        Out[row] = func(in1, in2, row);
        // func has read-only arguments
```

- **If an entire grid is written, it is OK to pass the entire grid**

```
initGrid(out);
// initializes whole grid
```

Use UniqInd as Data Type of Indexed Grids

- **UniqInd (Unique Index) means every index value is unique**

- **A[B[row]] = ….**
  - Can be parallelized only if B[row] contains unique values
    - Use **data type UniqInd for Grid B**
    - Required only if the ***output*** grid is accessed indirectly

How to Avoid Serial Counters

- **Traditional counting** (**serial**)

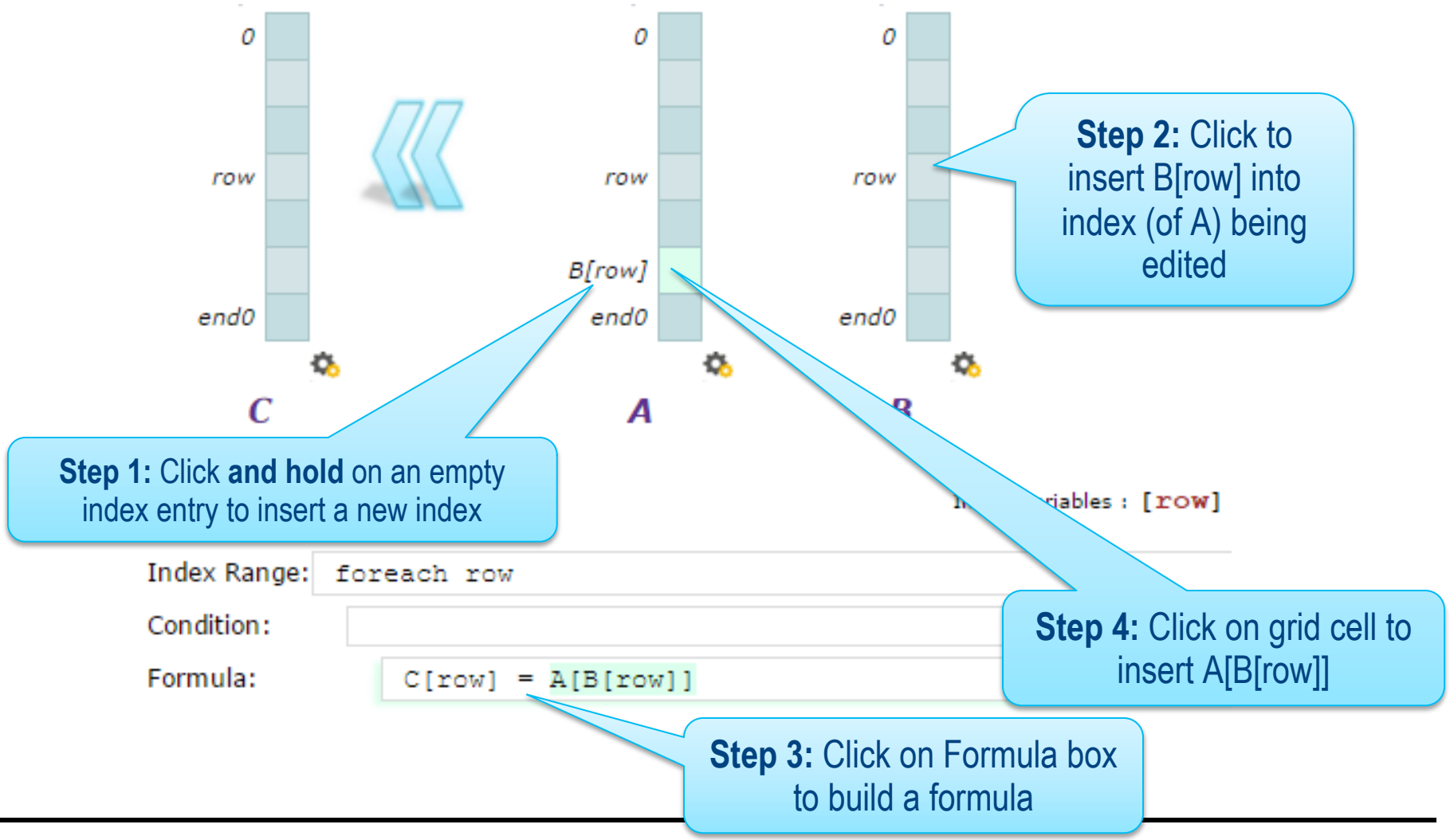
```
for (j=0; j < 10; j++) print(j);
```

- **Parallel counting** (**parallel**)

```
foreach row
        A[row] = row;
Print A
```

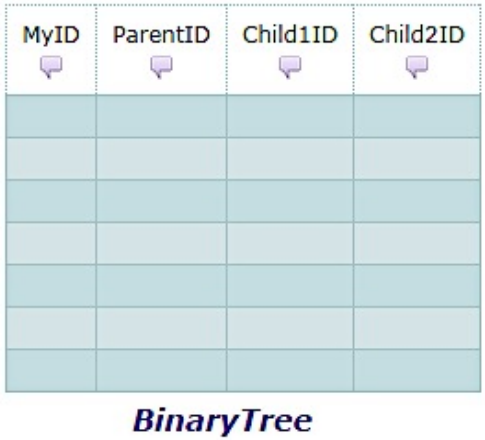
A[] can be filled in any order; hence parallel

How to Insert Indirect Accesses

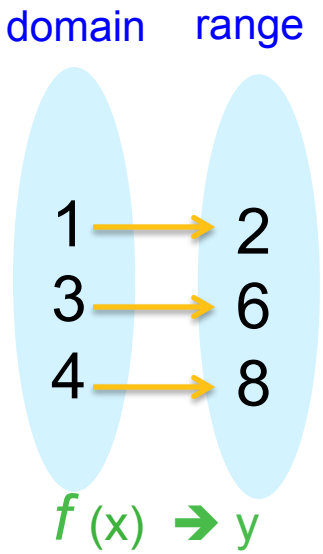


How to Think About Data Representation

- E.g., how do we represent trees, graphs, …



- Think in terms of relationships
  - Any mathematical relation is:
    - A mapping from ***domain*** to a ***range***
    - Can be represented with a table/matrix
      - If domain/range is *discrete* and *finite*



  - If there is a mathematical relation to your algo → represent as grid
    - E.g., graphs, trees, maps, GUI elements, objects, …
      - Ex. relationships: parent_of, child_of, neighbor_of, sibling_of

User is encouraged think in terms of relationships