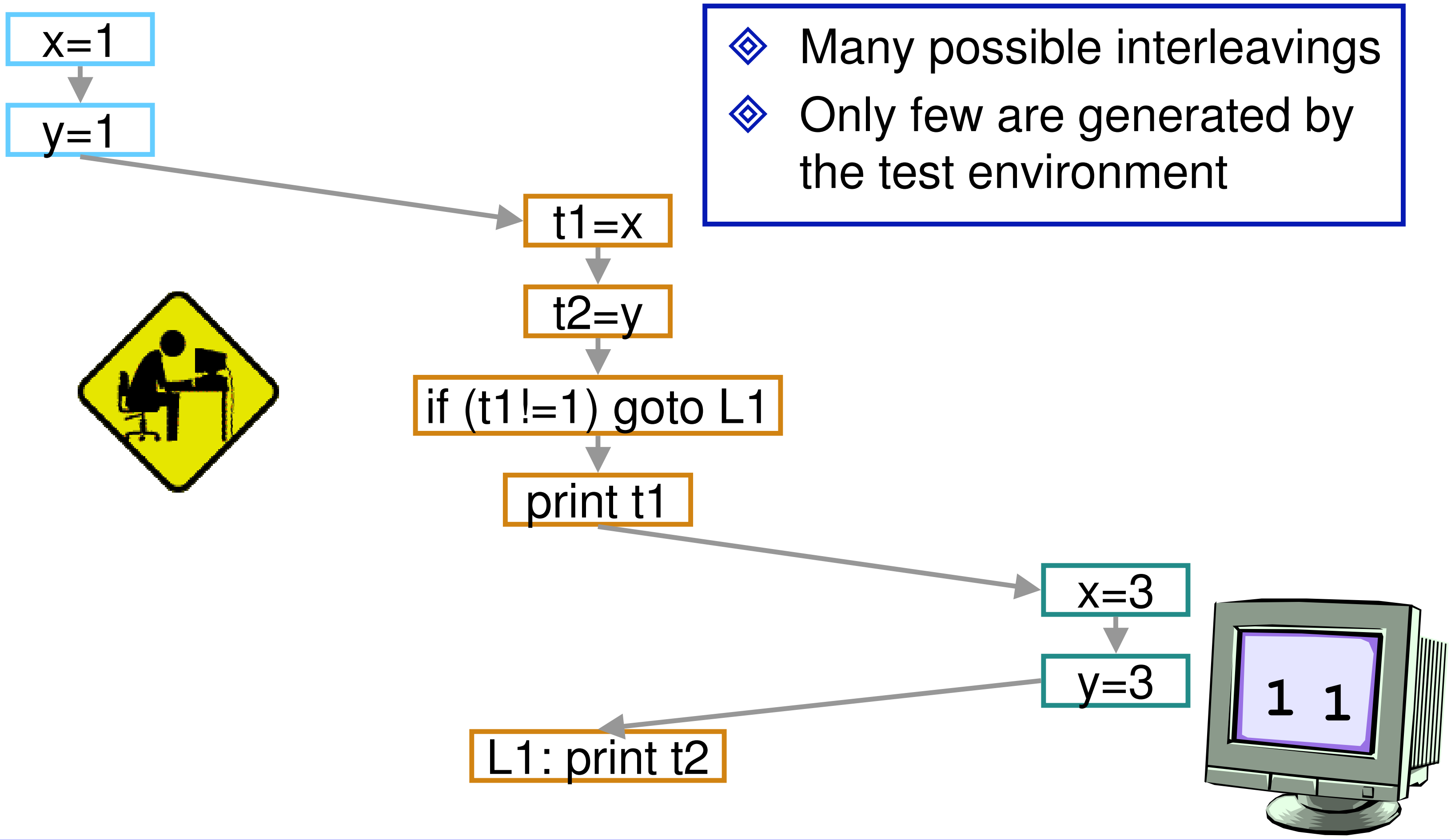
Fidgeting Till The Point Of No Return

- Marina Biberstein
  Eitan Farchi
  Shmuel Ur

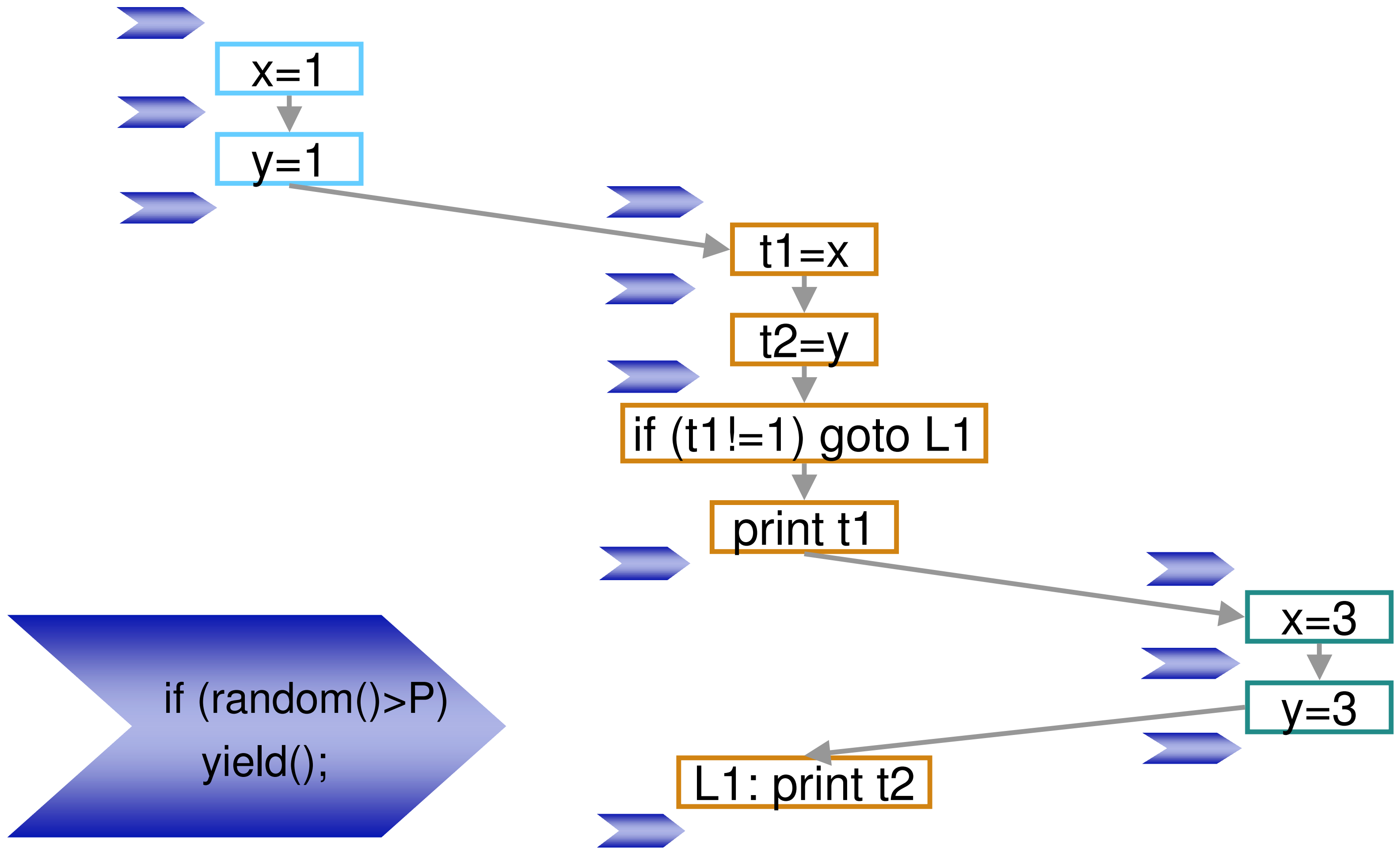
Table of contents

A sample program

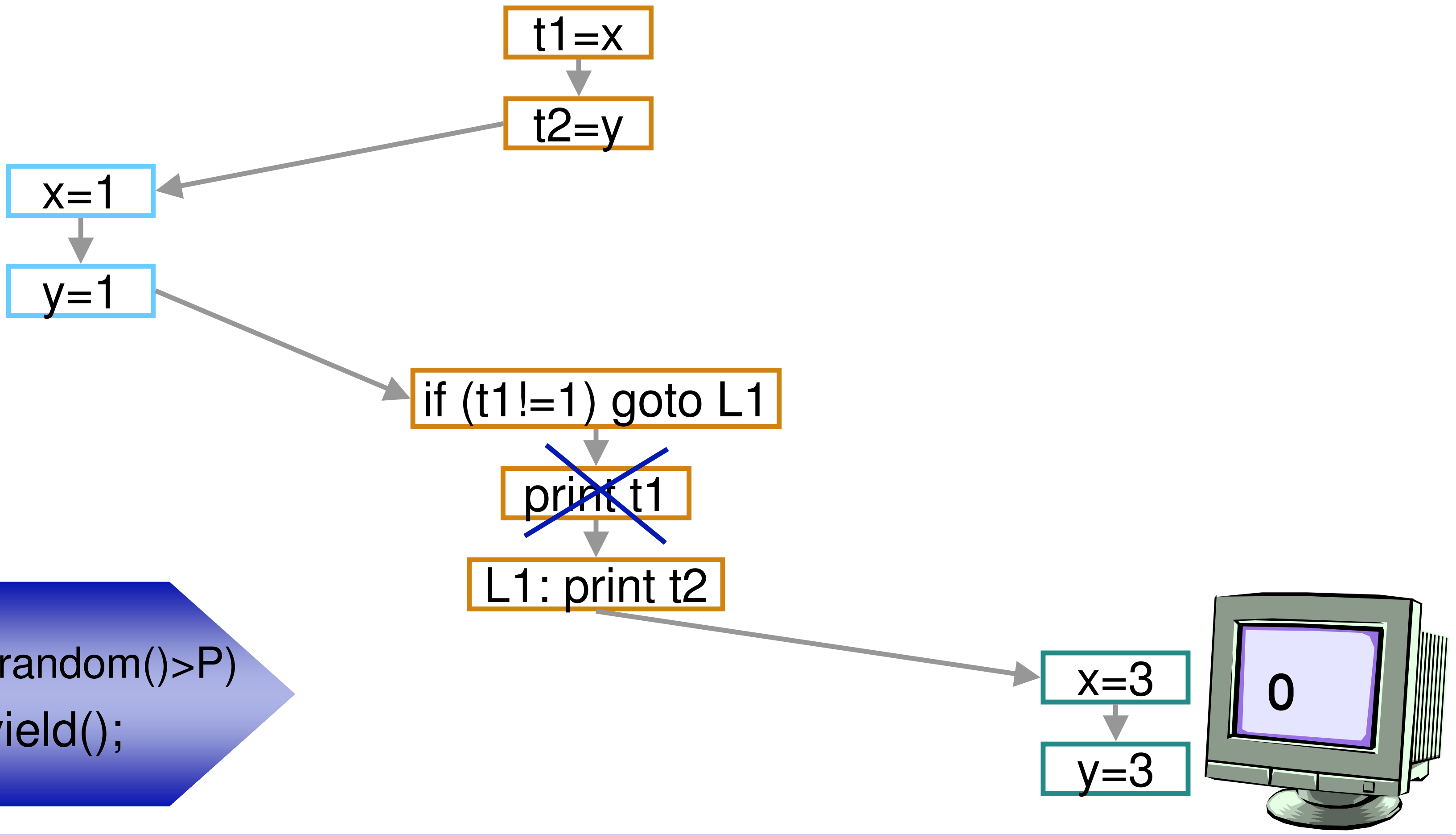


- Many possible interleavings
- Only few are generated by the test environment

Making things happen – the noise-making tools

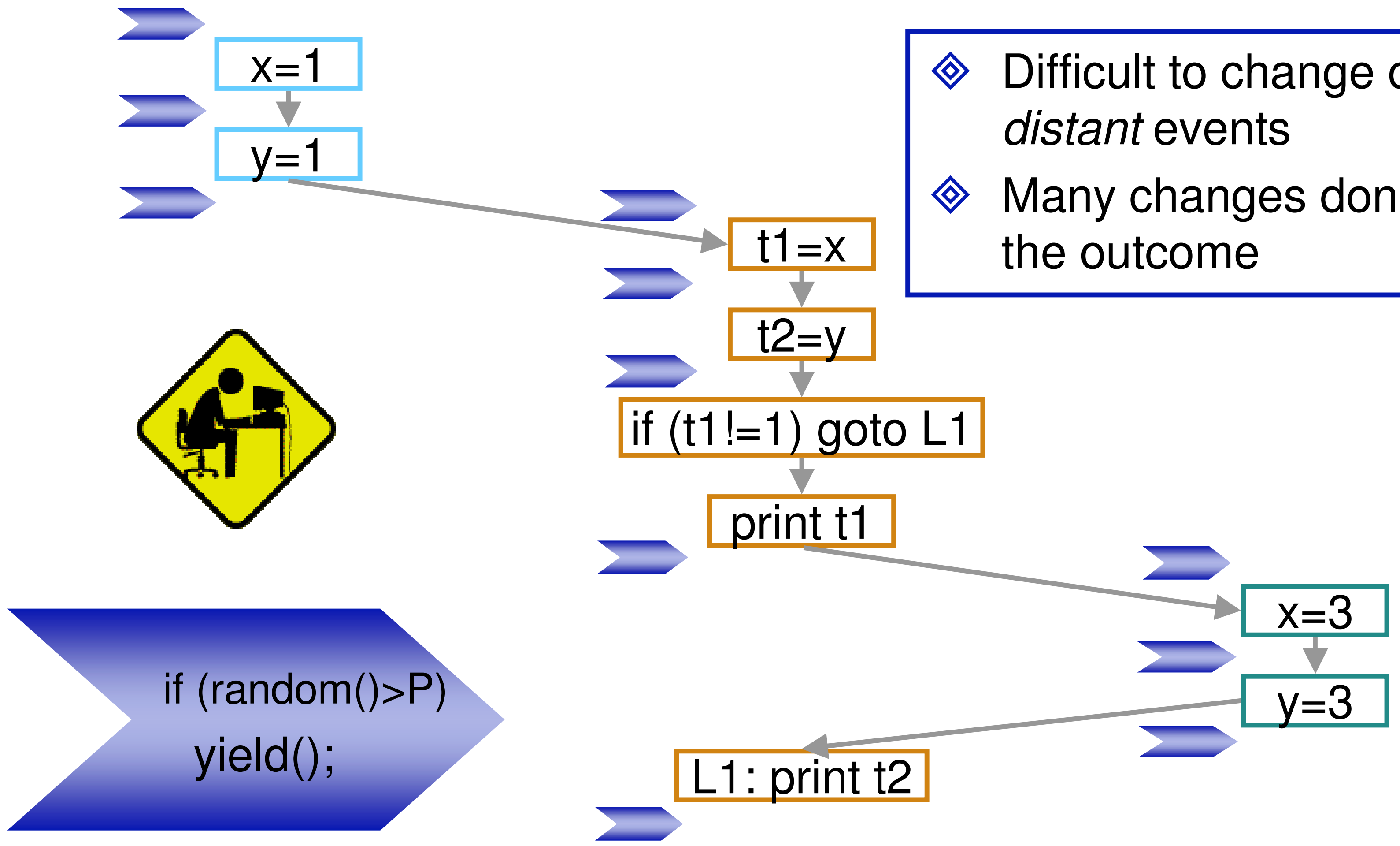


Making things happen – the noise-making tools



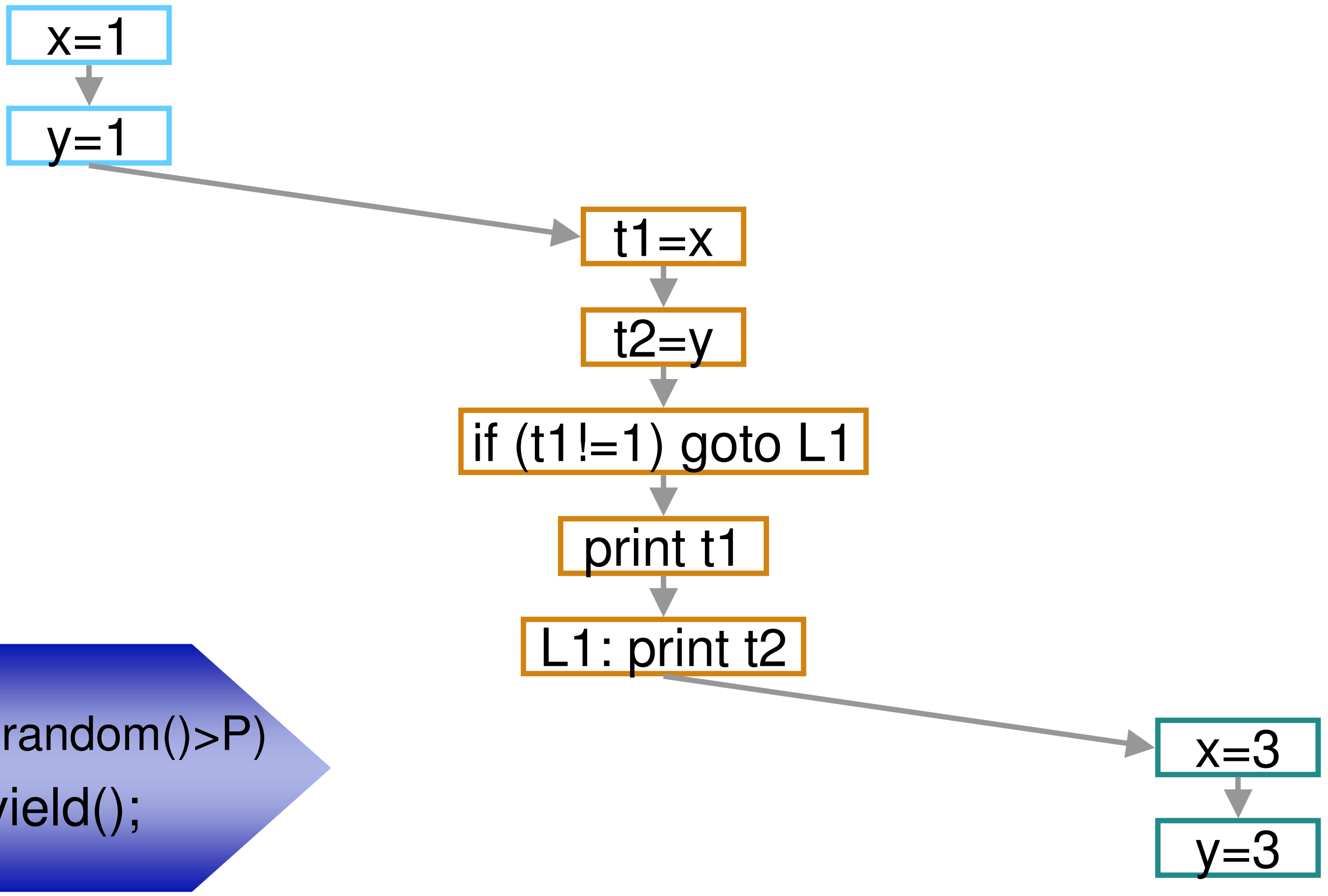
if (random()>P)
yield();

Making things happen – the noise-making tools



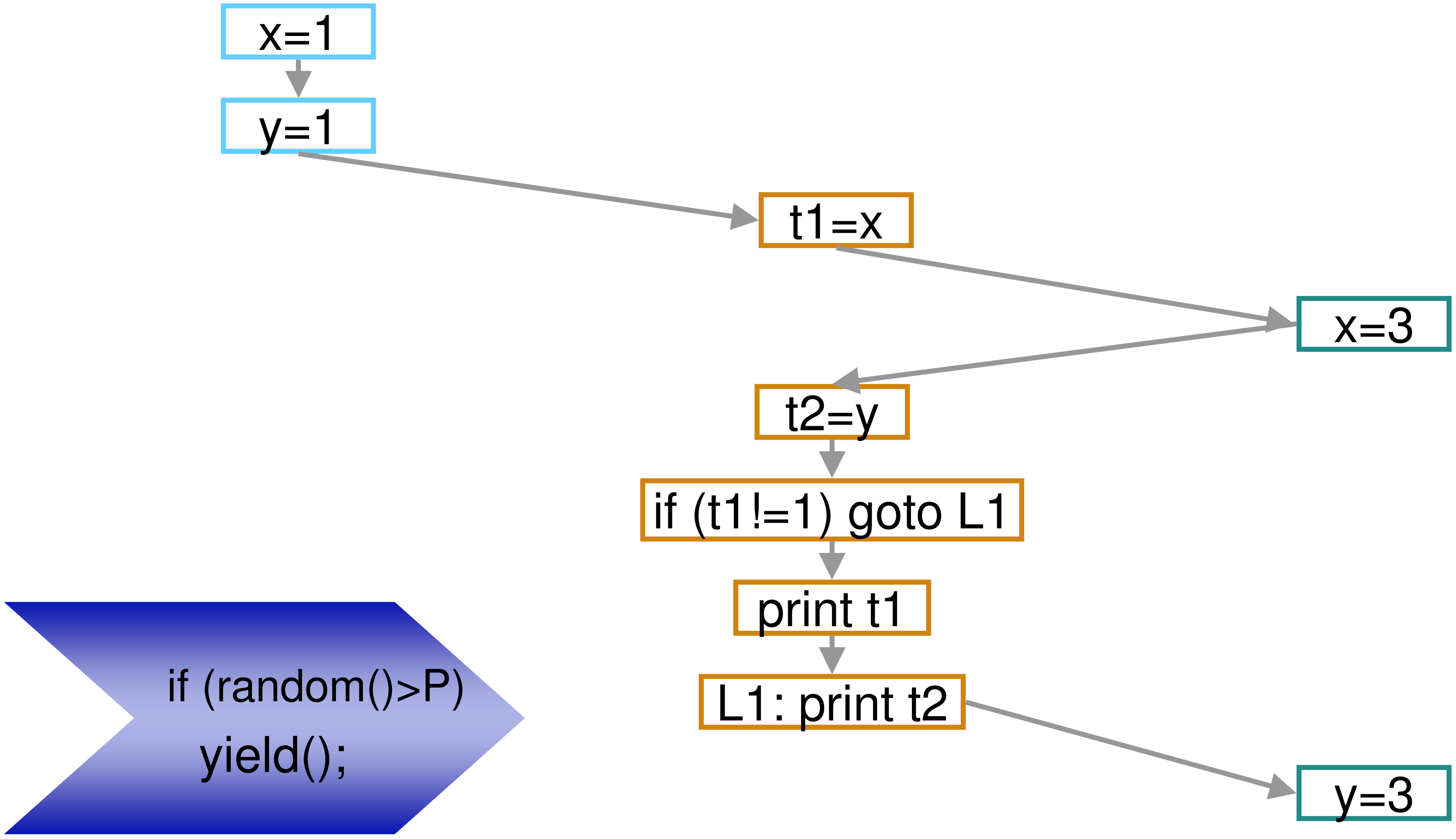
- Difficult to change order of *distant* events
- Many changes don't affect the outcome

Noise-making tools: equivalent schedules

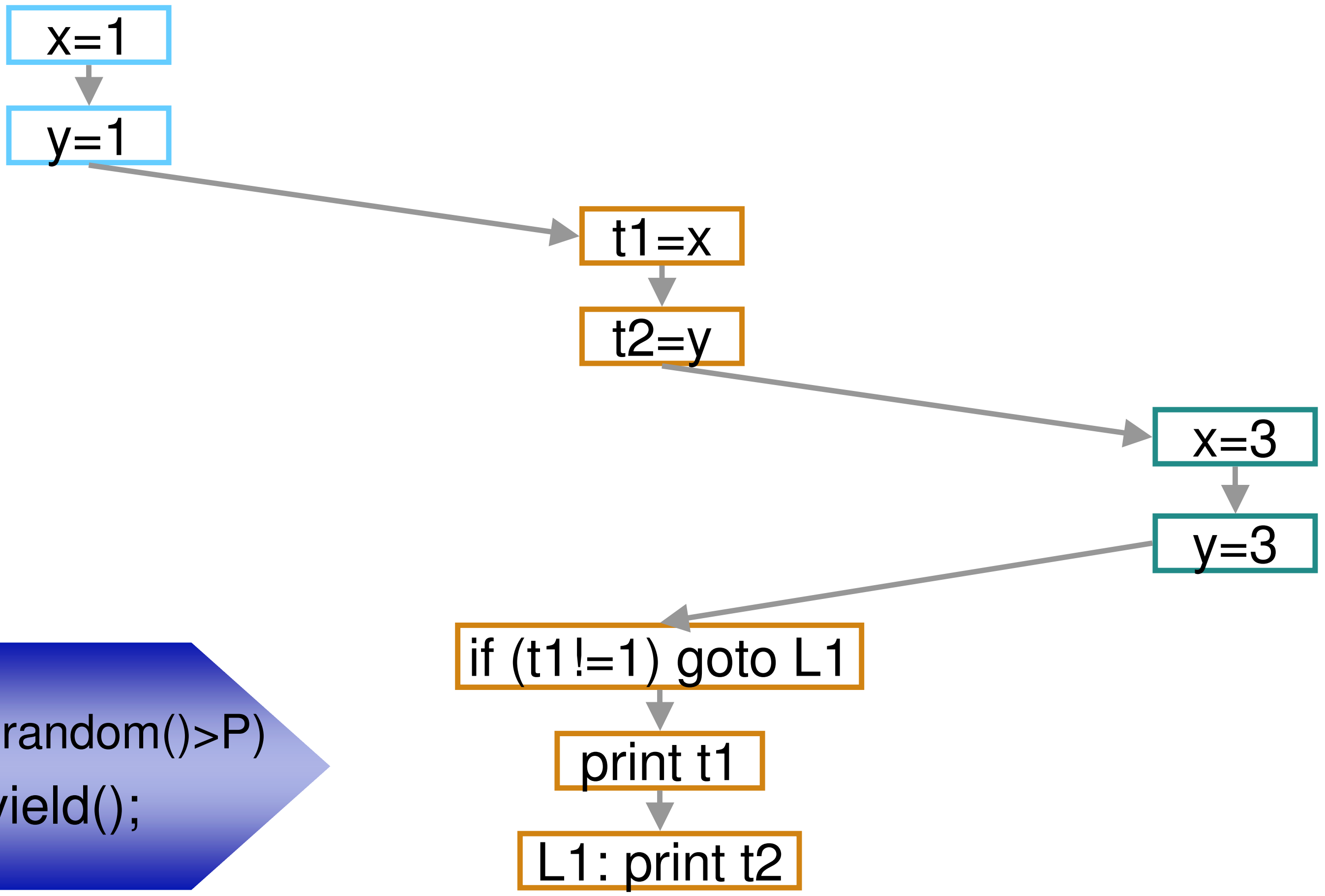


```
if (random()>P)
  yield();
```

Noise-making tools: equivalent schedules

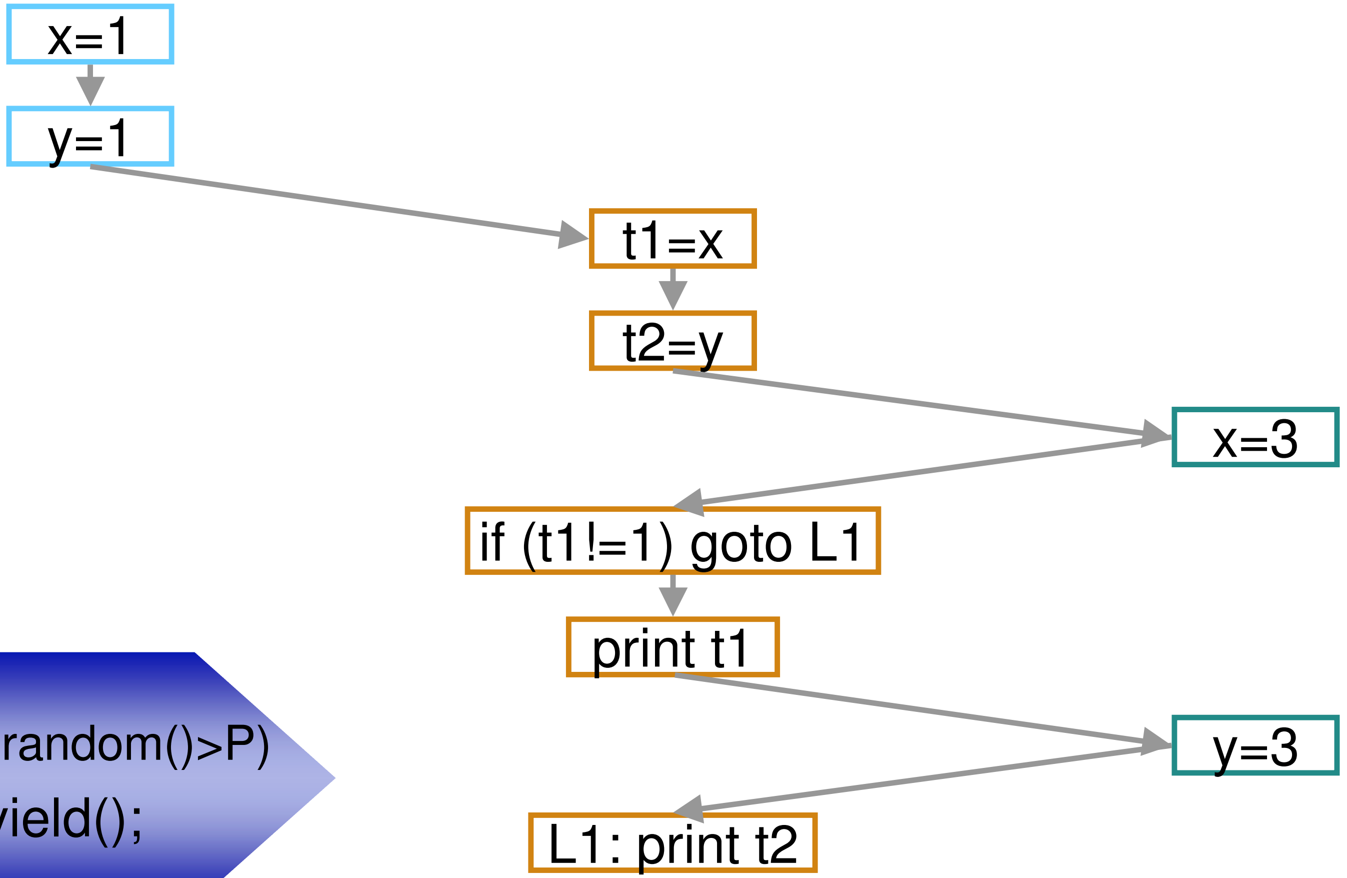


Noise-making tools: equivalent schedules



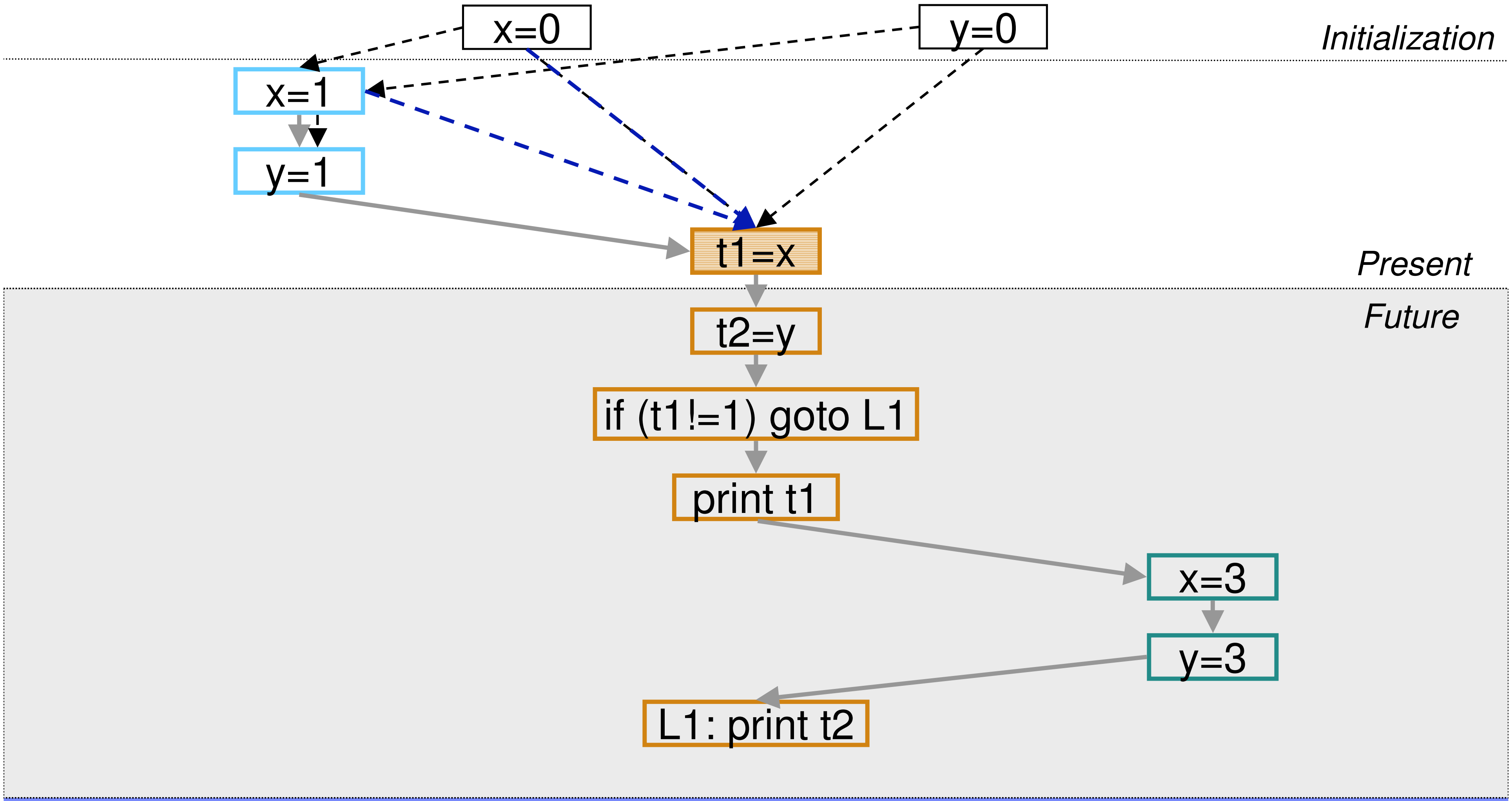
if (random()>P)
yield();

Noise-making tools: equivalent schedules

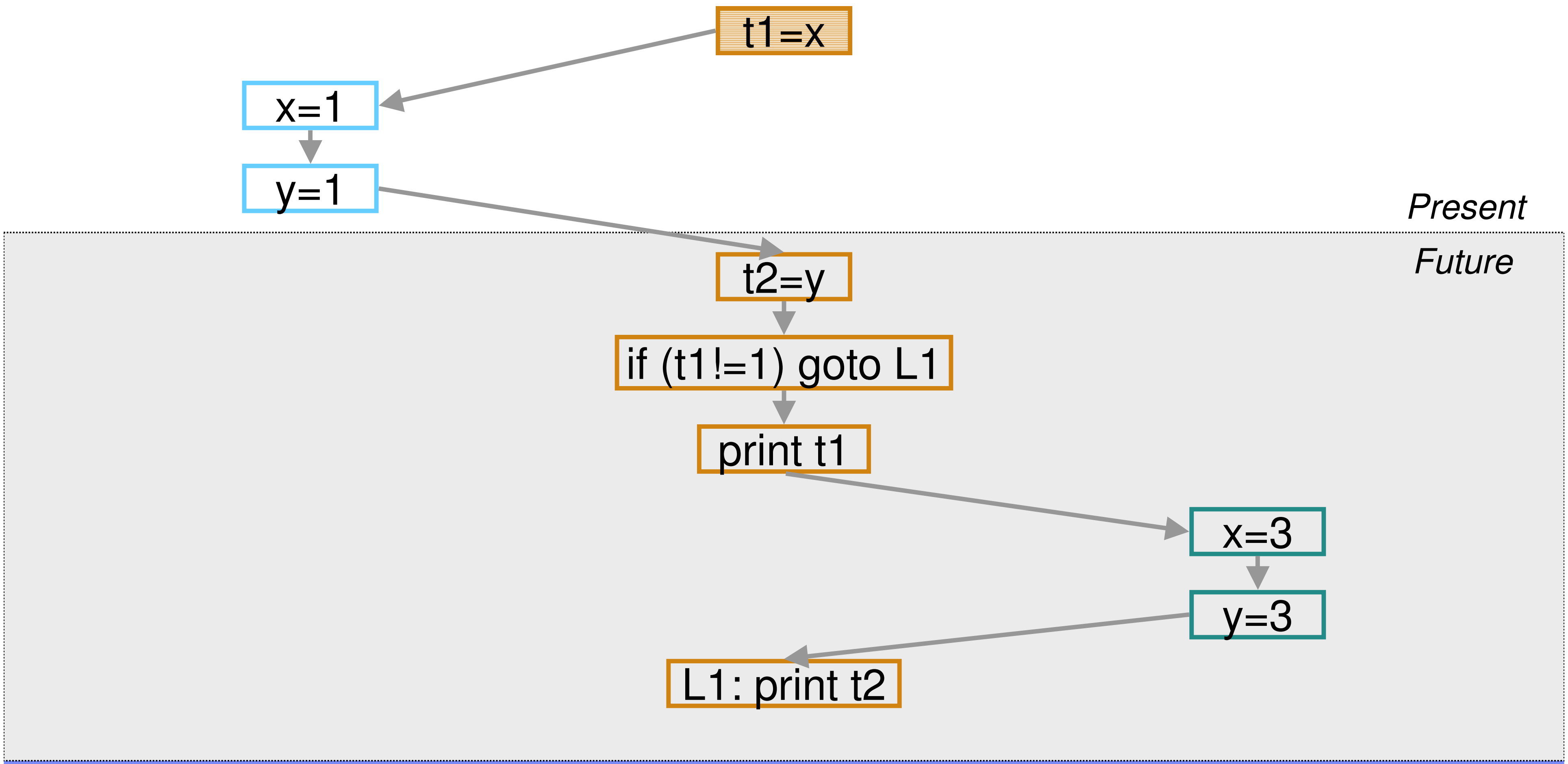


```
if (random()>P)
  yield();
```

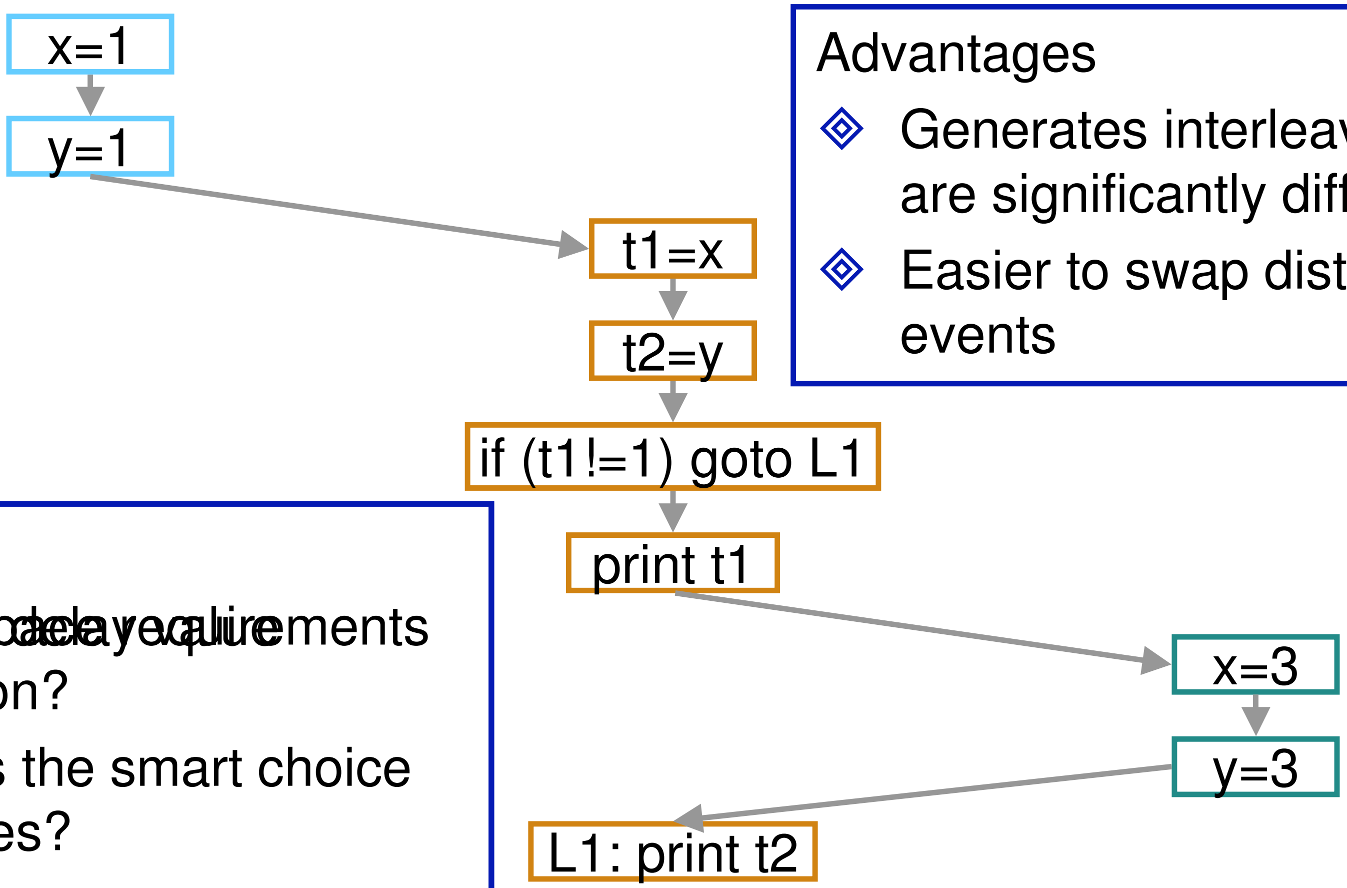
Alternative Pasts: generating interesting things



Alternative Pasts: generating interesting things



Alternative Pasts: generating interesting things



Advantages

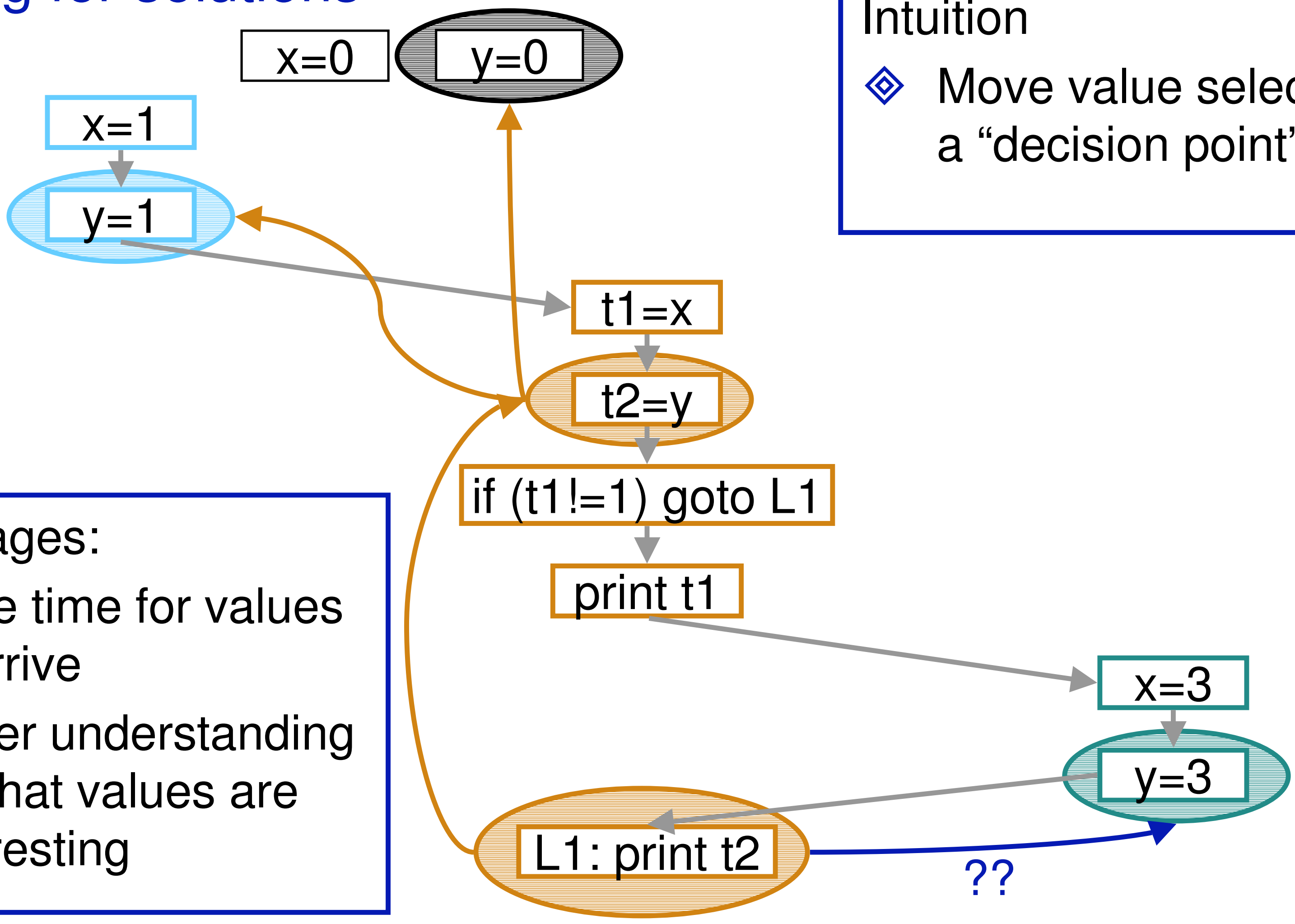
- Generates interleavings that are significantly different
- Easier to swap distant events

Issues

- Time/space requirements
- How to delay value selection?
- What is the smart choice of values?

Table of contents

Looking for solutions



Intuition

- Move value selection to a "decision point"

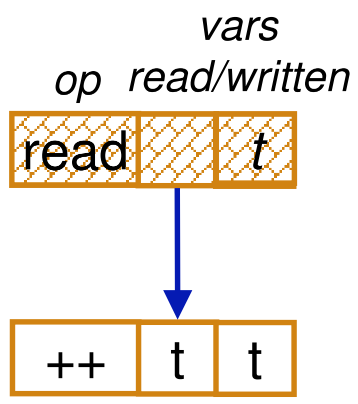
Advantages:

- More time for values to arrive
- Better understanding of what values are interesting

Fidgeting: the basic concepts

- Instructions: broken into two groups
  - Can be re-executed: **=, +, -, …**
  - Can't be re-executed: **if, print**
- Events:
  - Instruction
  - Variables read
  - Variable written
- Visibility graph:
  - Timing restrictions on events
  - Nodes:
    - Event
    - Event state (*raw* or *processed*)
  - Edges: timing precedence

```
t = read();
t++;
```



☐	*raw*
▨	*processed*

Visibility: When can a value be used?

- Problem:
  - Node r reads variable λ
  - Node w writes variable λ
  - Can r use the value produced by w?
- Answer: Yes, unless timing restrictions in visibility graph imply that
  - r precedes w, or
  - Another node that writes λ intervenes between w and r
- In graph terms:
  - There is a path from r to w, or
  - There is a path from w to r that passes through a node writing λ

Hiding nodes

- Situation:
  - Node r reads variable λ
  - Nodes w, w' write variable λ and are visible from r
- Problem: make w' invisible
- Solution:
  - Add edge (r, w'), or
  - Add edges (w', w) and (w, r)
- Exists a method that doesn't introduce cycles

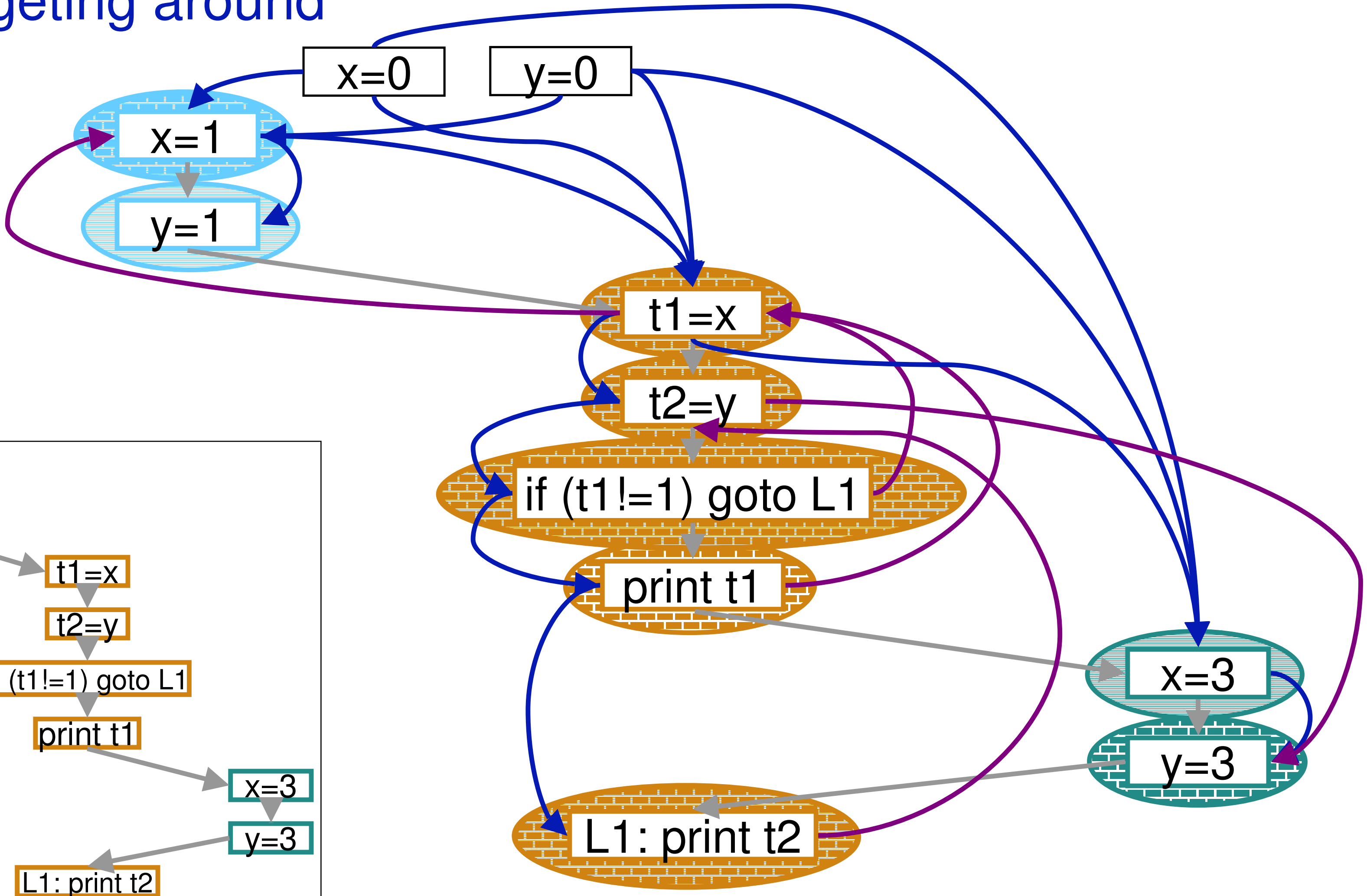
Processing node

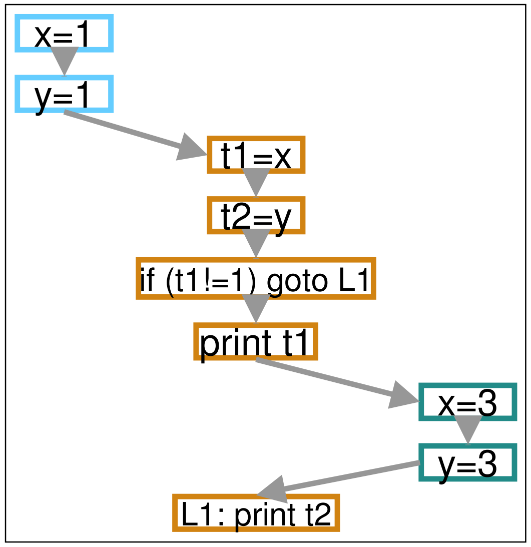
- Goal: Select the values to be used by node *n*
- Processing node n:
  - If node state is *processed* – done
  - Set node state to *processed*
  - For every variable λ read by *n*
    - Select a visible node *w* that writes λ
    - Hide all other visible nodes that write λ
    - Process *w*

Fidgeting: An outline

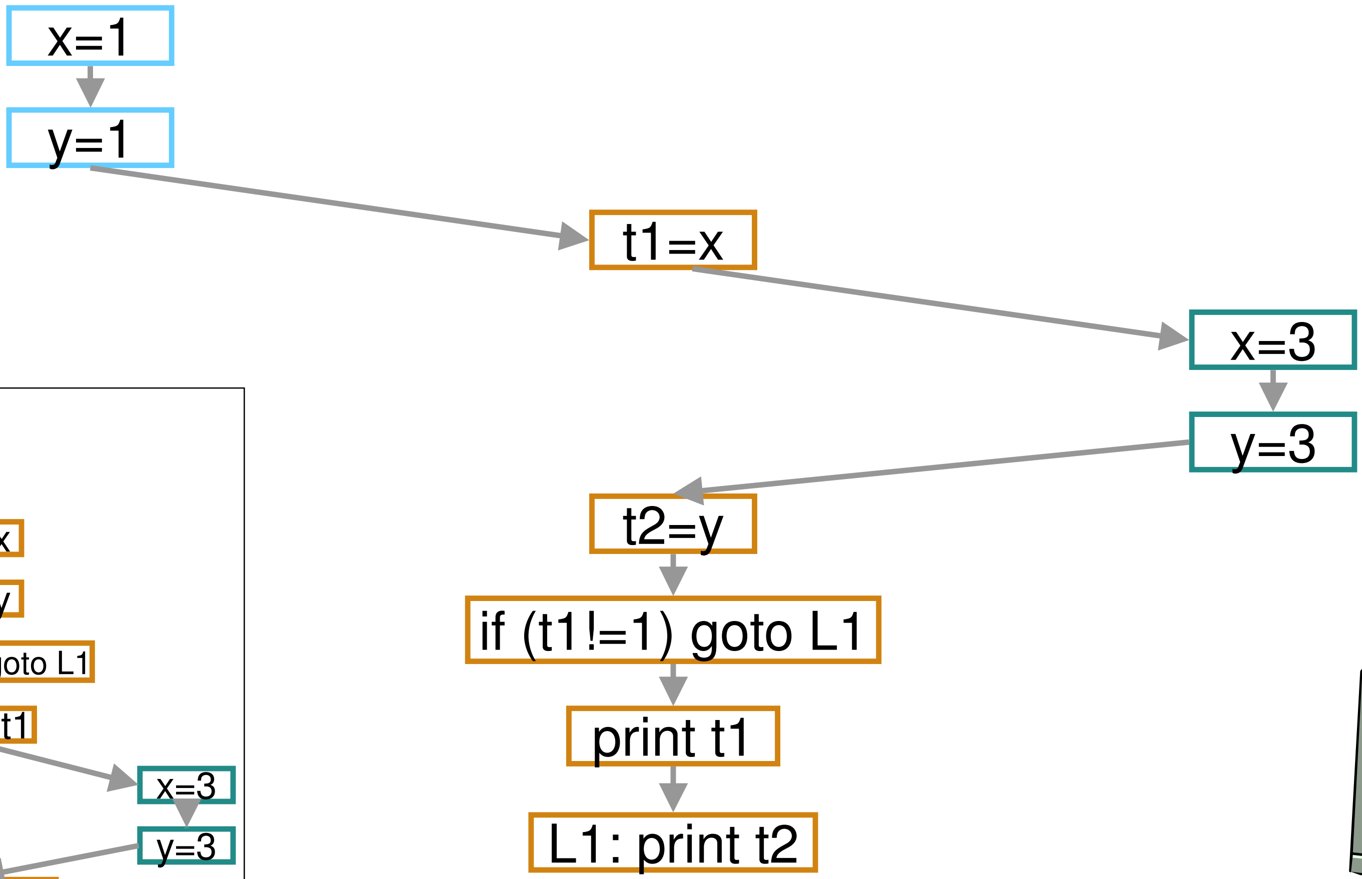
- Start executing the tested program
- At each event:
  - Create a new *raw* node
  - Add it to graph
    - First event in thread:
      - Add edge from *create* in the parent thread
      - Add edges from initialization events
    - Otherwise: add edge from the previous event in the thread
  - If the instruction cannot be replayed: process the node
  - Execute the event,
    - Raw: no intervention
    - Processed: for each read variable, use its value as produced by the visible write event

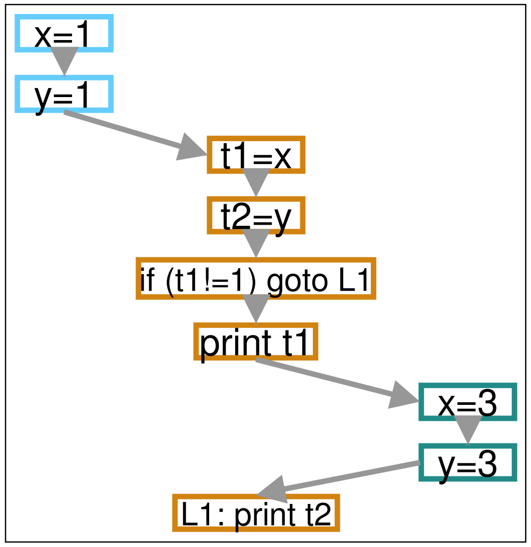
Fidgeting around





Fidgeting around





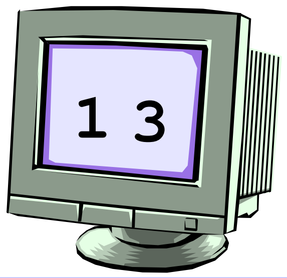
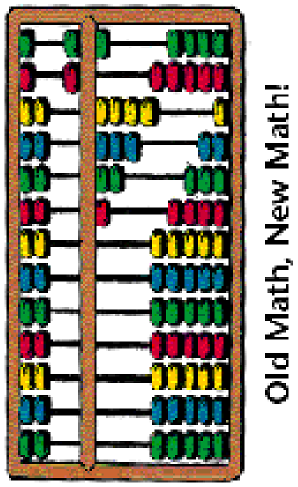
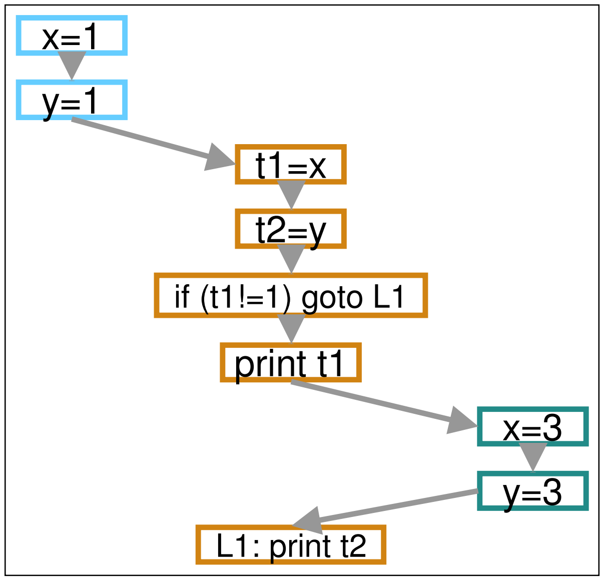


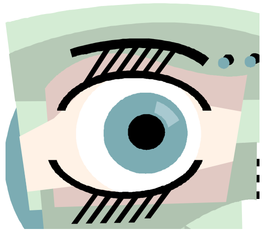
Table of contents

Summing up

- A new algorithm for generating interesting interleavings
- More aggressive delays that with alternative pasts
- More informed choice of values at decision points
  - Especially useful for achieving coverage
- Noise-makers can help delay decision points
- Complexity issues remain to be addressed
  - Some optimizations available and should be evaluated

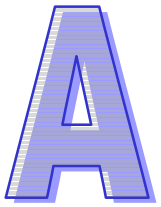


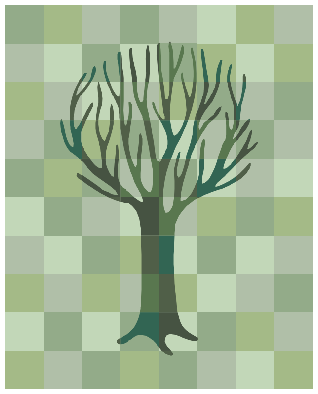




There once was a man who said, “God
Must think it exceedingly odd
    If He finds that this tree
    Continues to be
When there’s no one about in the Quad.”

“Dear Sir:
    Your astonishment’s odd:
I am always about in the Quad
    And that’s why the tree
    Will continue to be,
Since observed by,
                Yours faithfully,
                    God.”





Q&A