Refinement Correction Strategy

for Invalid XML Documents and Regular Tree Grammars

Martin Svoboda and Irena Holubova (Mlynkova)

svoboda@ksi.mff.cuni.cz

DEXA 2014

Munich, Germany

September 2, 2014

XML and Web Engineering Research Group

Charles University in Prague

Outline

- Introduction
- Corrections
- Algorithms
- Experiments
- Conclusion

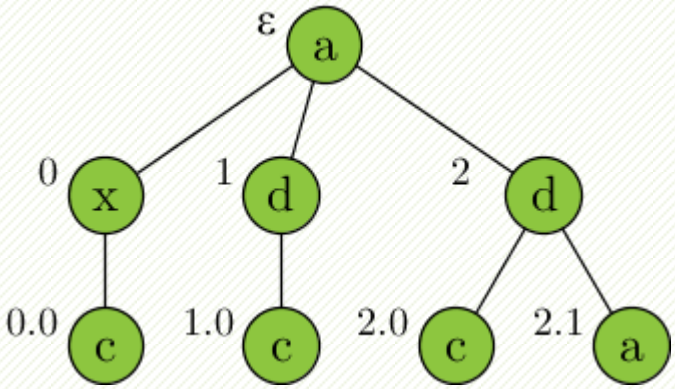
Introduction

- Motivation
  - **Incorrect XML data**
    - Well-formedness, schema validity, data consistency
- Input
  - **One XML document**
    - Well-formed but (potentially) invalid
  - **DTD or XSD schema**
- Goal
  - **Structural corrections of elements**

Sample Correction

- **Document**

```
<a>
  <x><c/></x>
  <d><c/></d>
  <d><c/><a/></d>
</a>
```



- **Grammar**

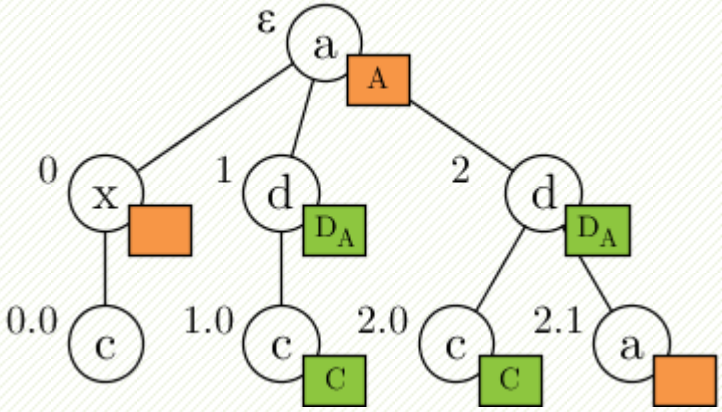
[a, C.D_A* → A]

[b, D_B* → B]

[c, ε → C]

[d, C* → D_A]

[d, A|B|C → D_B]



Edit Operations

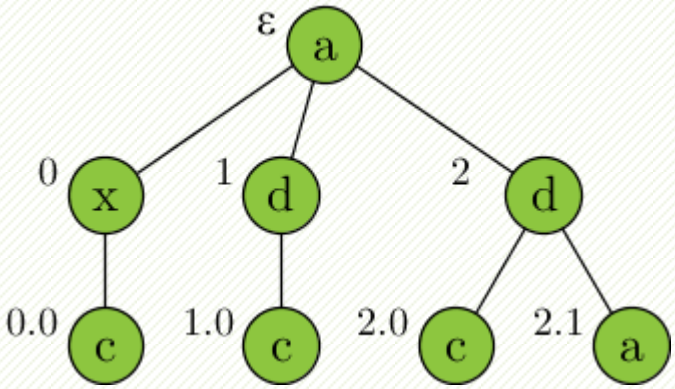
- **Edit operations**
  - Add leaf node
  - Remove leaf node
  - Rename node
- **Edit sequences**
  - Insert new subtree
  - Delete existing subtree
  - Repair existing subtree
    - With an option of node renaming

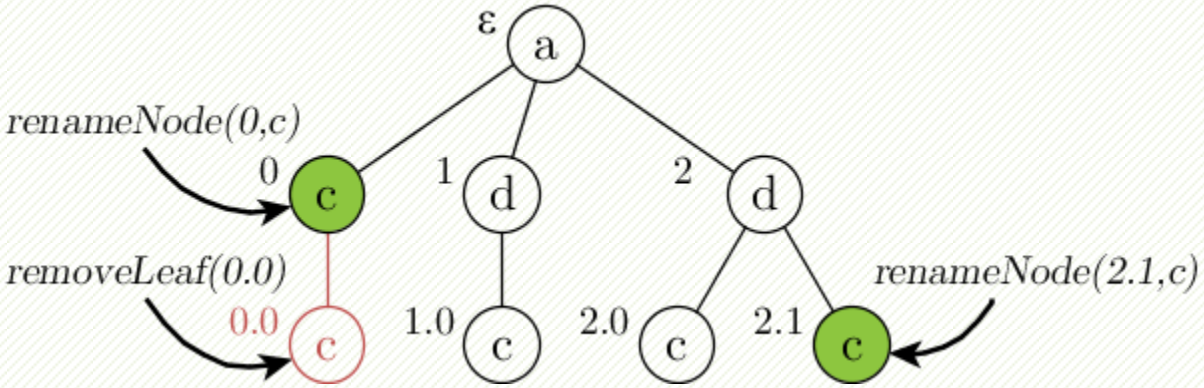
Edit Operations

- **Example**

  ```
  renameNode(0,c),
  removeLeaf(0.0),
  renameNode(2.1,c)
  ```

- **Cost**

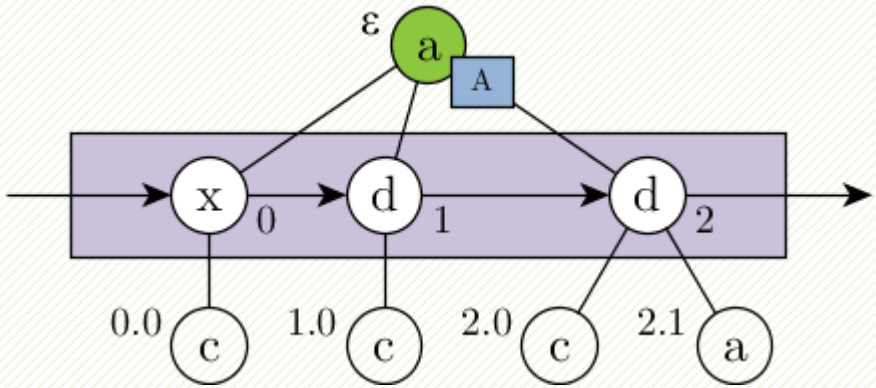




Algorithm Idea

- **Recursive processing**
  - From the root node towards leaf nodes…
  - … and at each particular data tree node…
  - … correct a sequence of its child nodes
- **Example**

$C.D_A*$

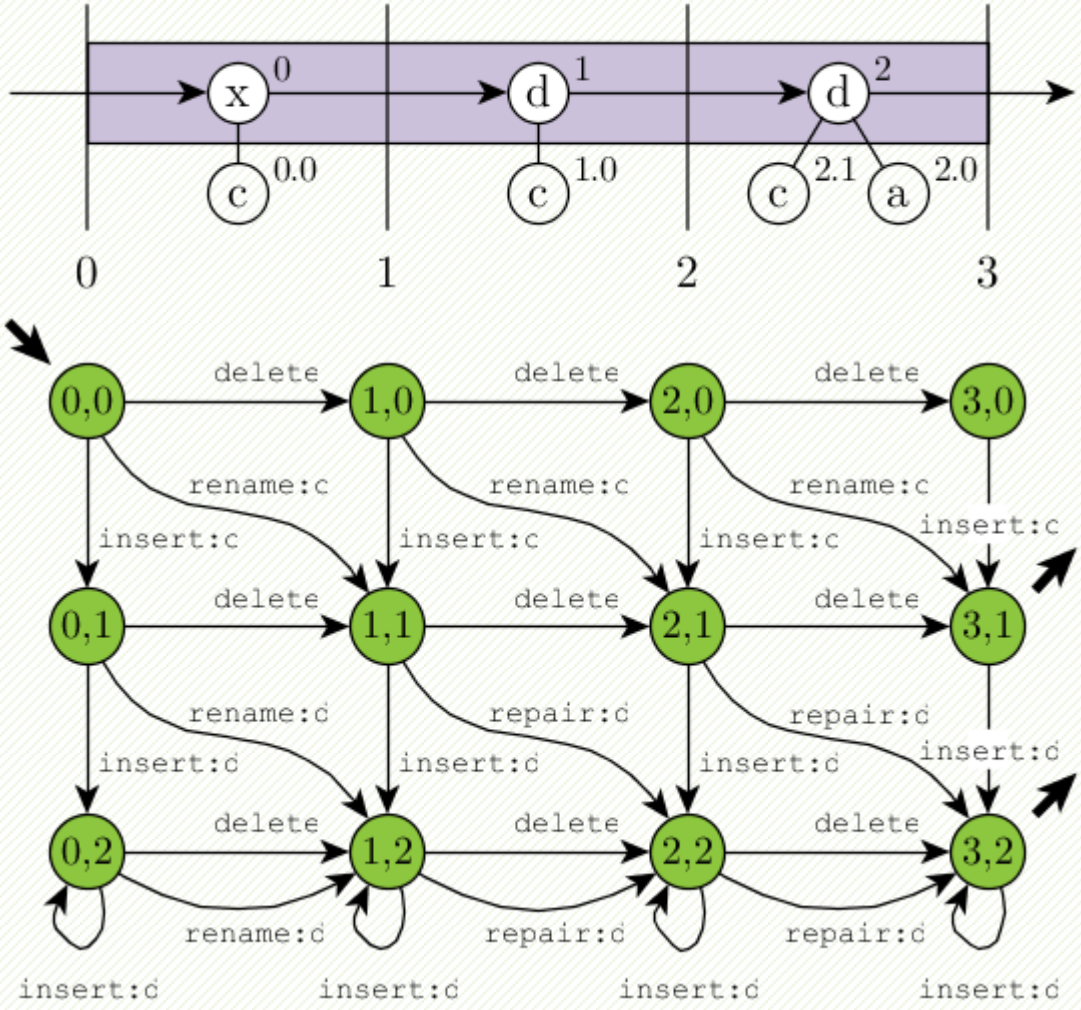


Horizontal Correction

- Automaton traversal
  - **Start**
    - **Before the entire node sequence**
    - **At the initial automaton state**
  - Step
    - Before some particular node (if any)
    - At some particular automaton state
  - **End**
    - **After the entire node sequence**
    - **At one of the accepting states**

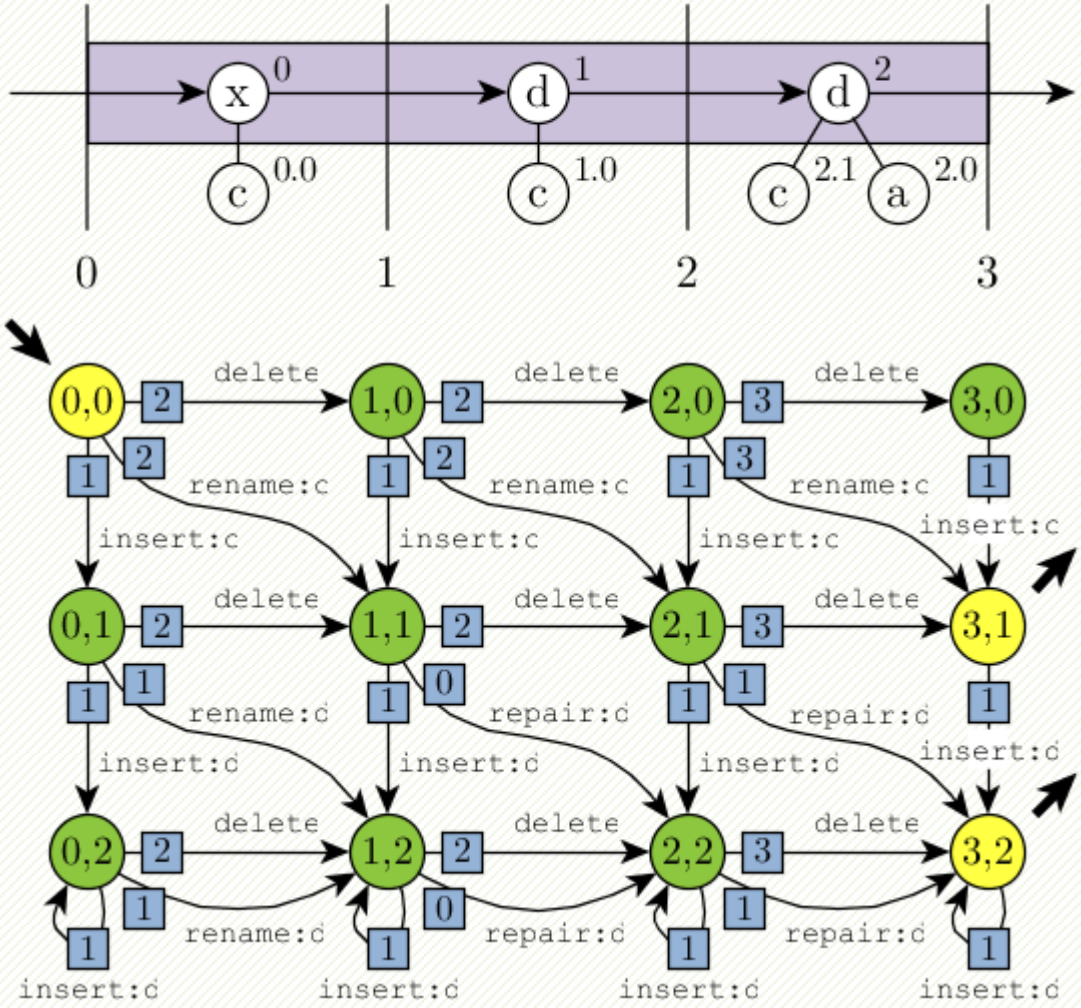
Correction Multigraphs

- **Structure**
  - Vertices
  - Edges



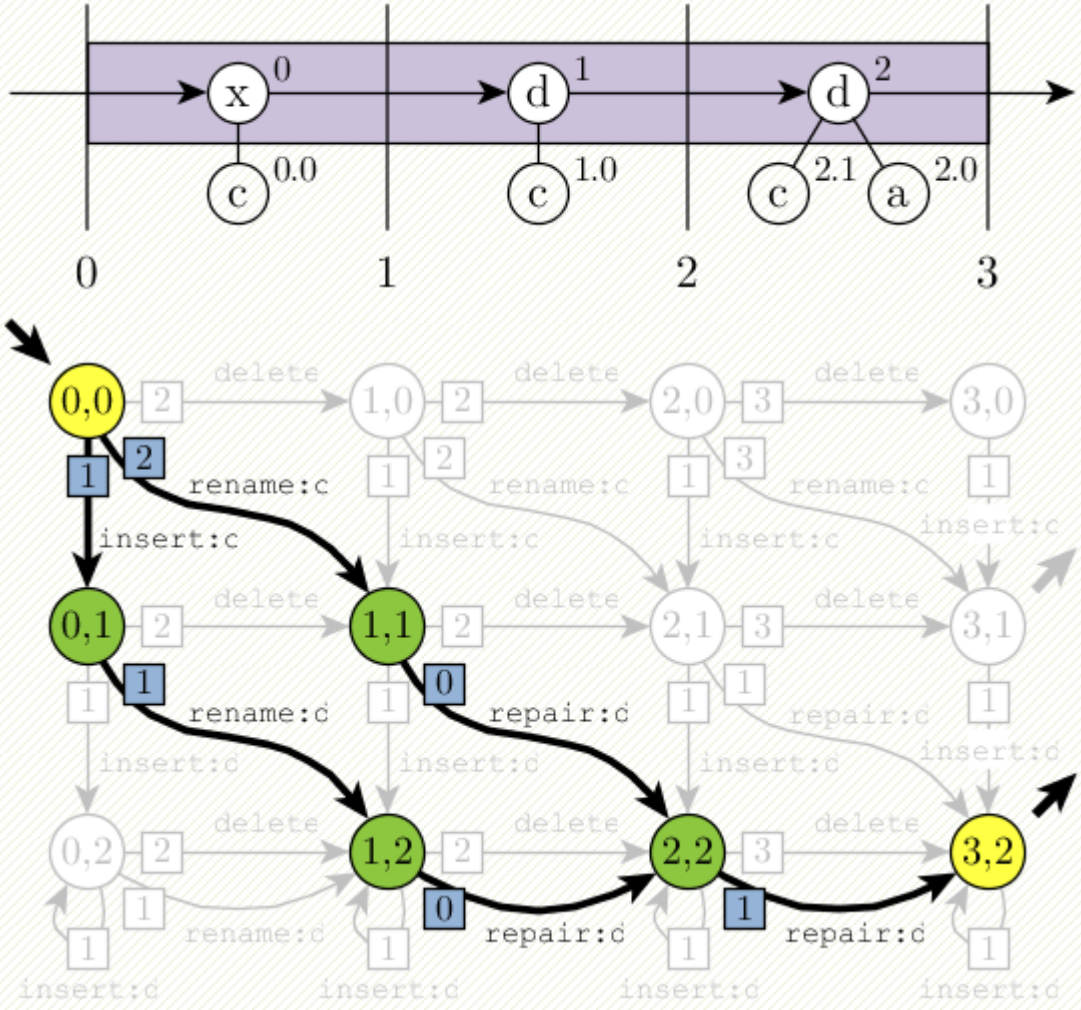
Shortest Paths

- **Paths**
  - Source
  - Targets



Intent Repair

- Structure

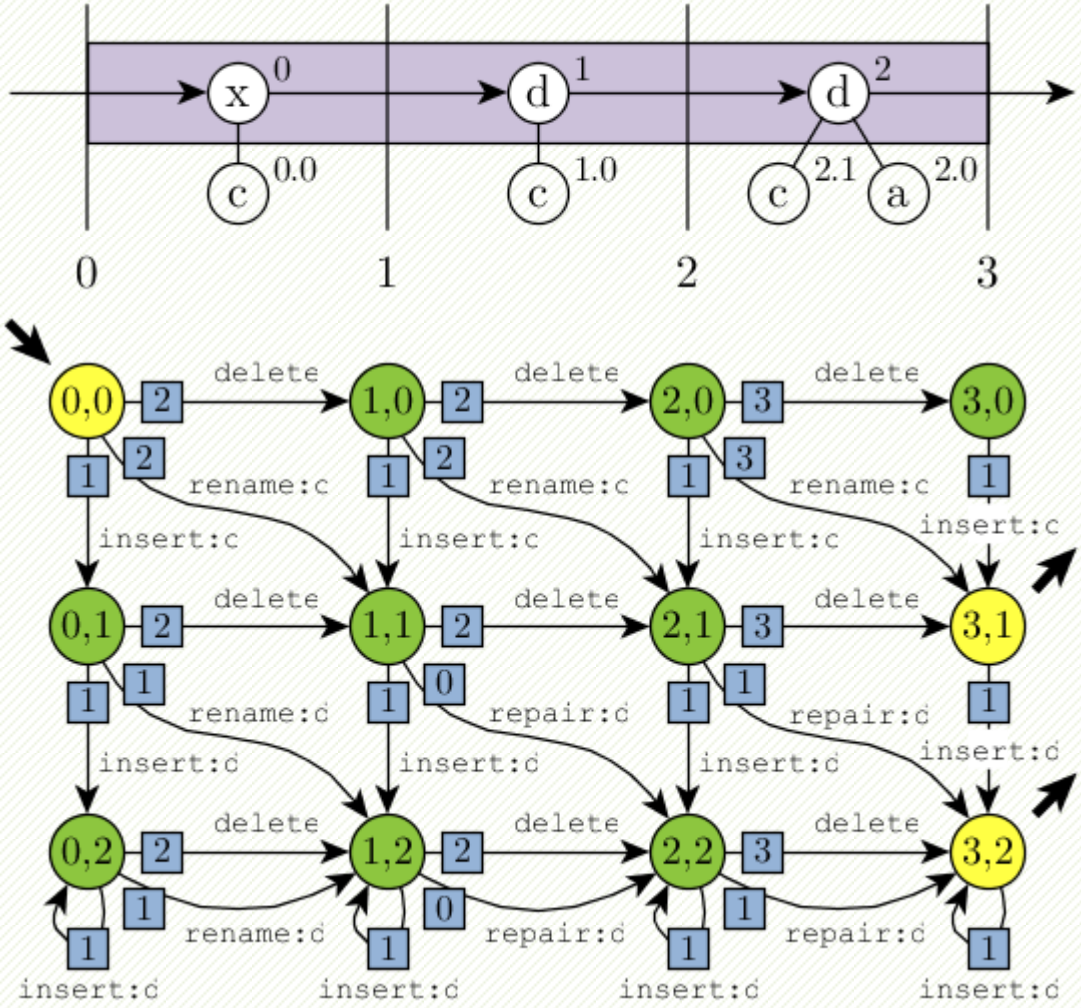


Intent Signatures

- Observation
  - **Different intents** may lead to **identical repairs**
    - We do not need to evaluate them repeatedly
- Solution
  - Intent signatures
  - Repairs caching

Correction Strategies

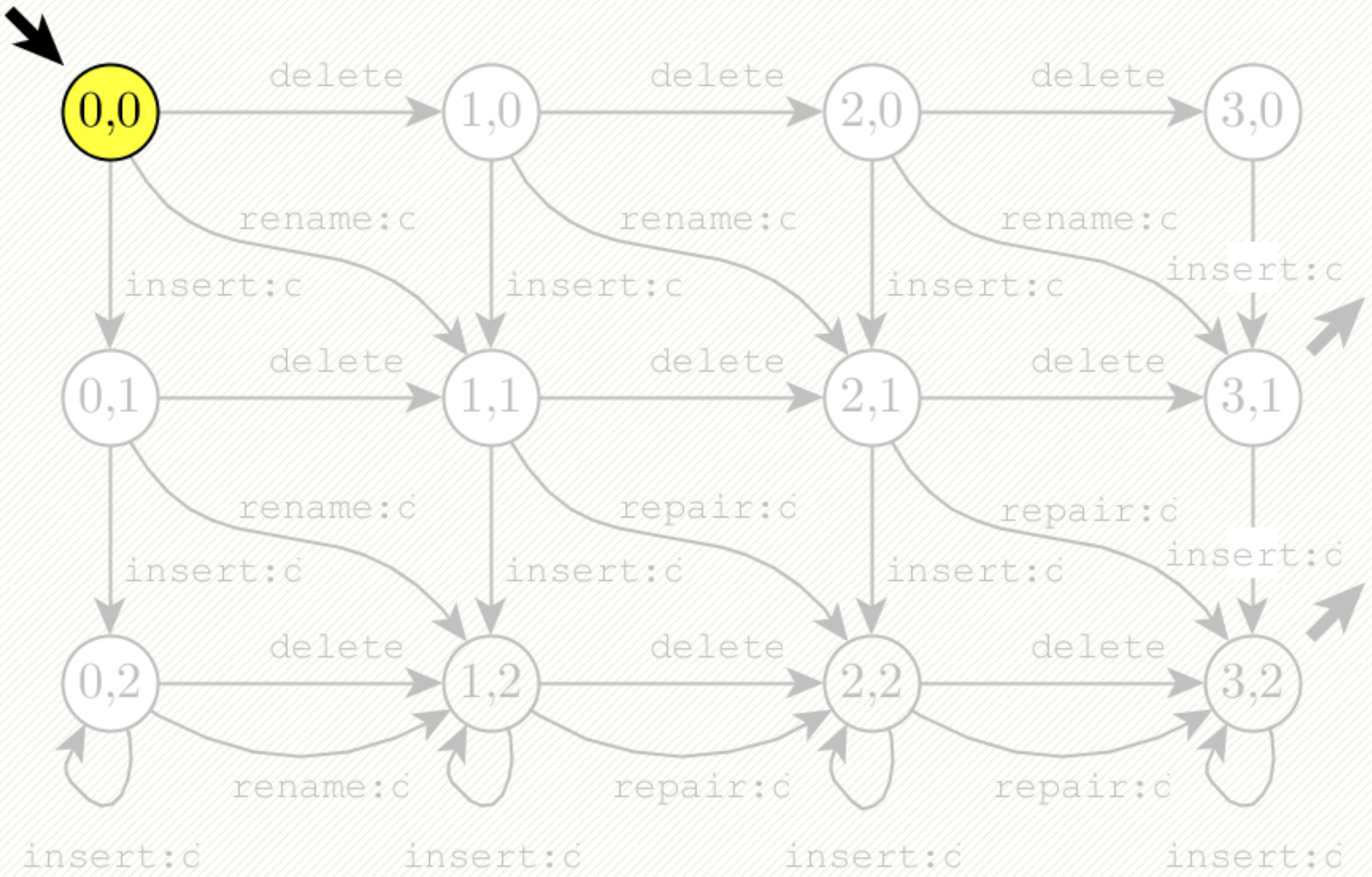
- **Strategies**
  - Default
  - Exploring
  - Refinement



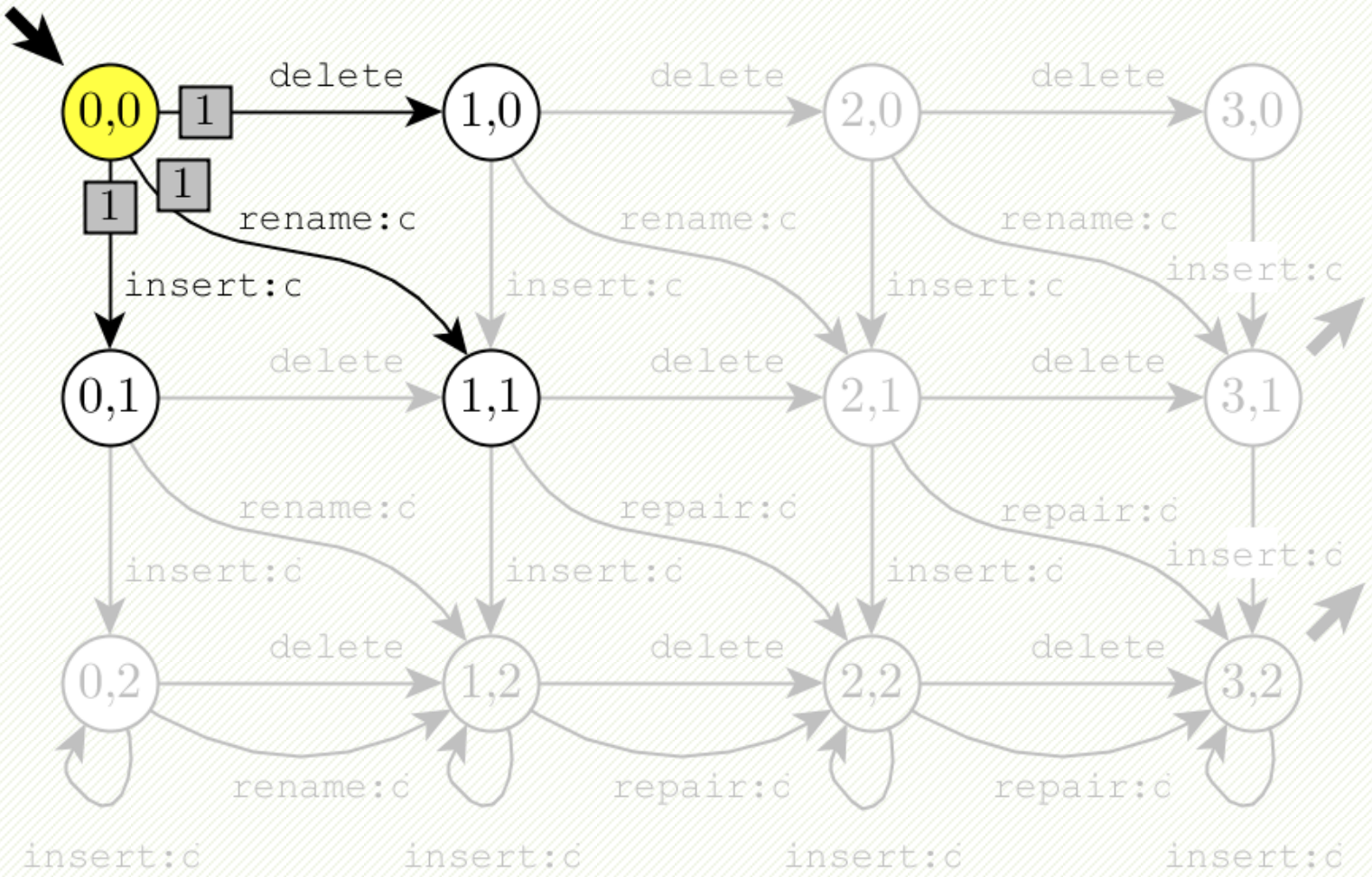
Refinement Strategy

- Observation
  - **Until now we always worked with…**
    - **… fully evaluated nested intents**
    - **… and therefore their final costs**
- Idea
  - **Refinement exploration based on estimations**

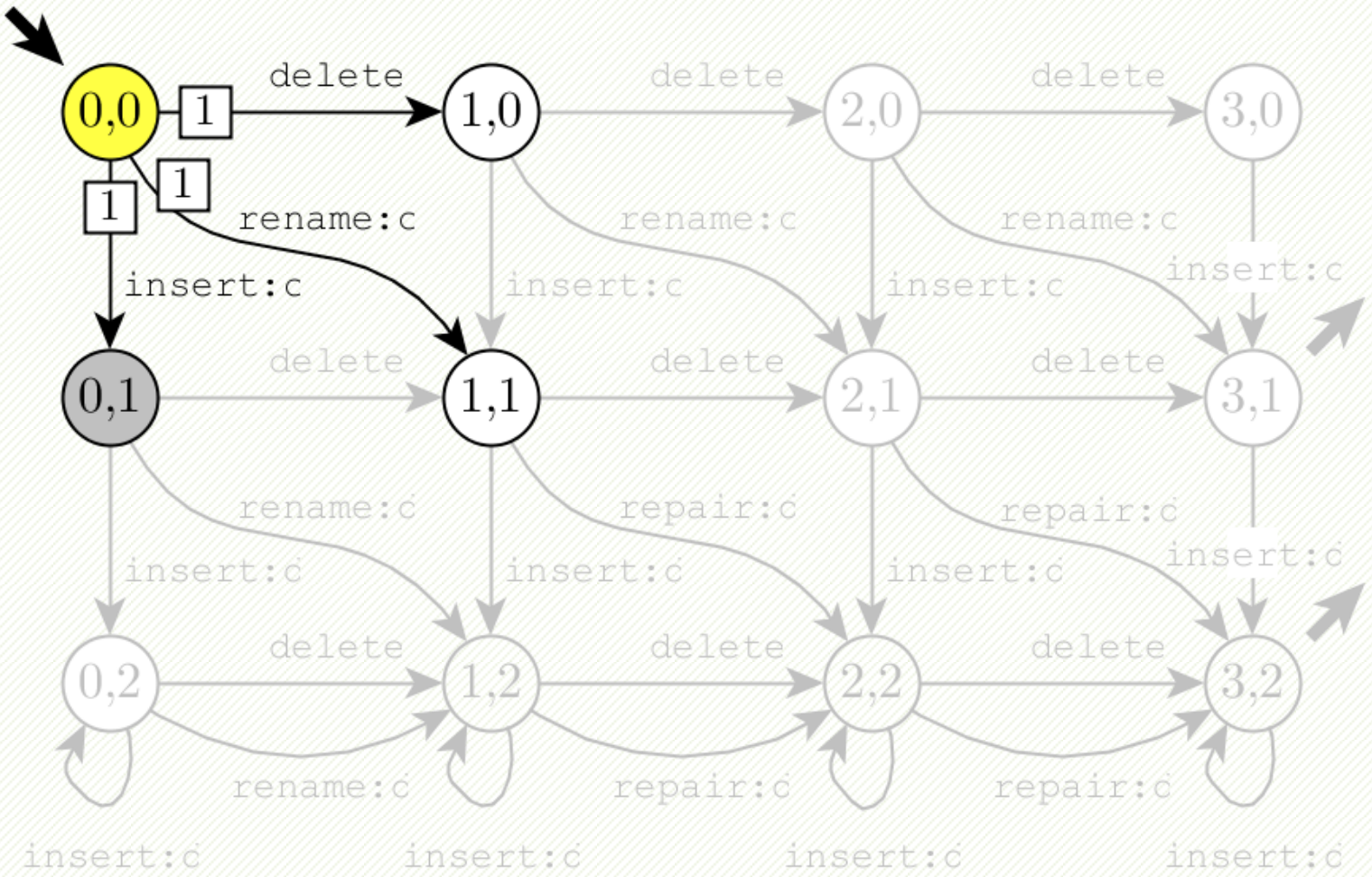
Refinement Strategy



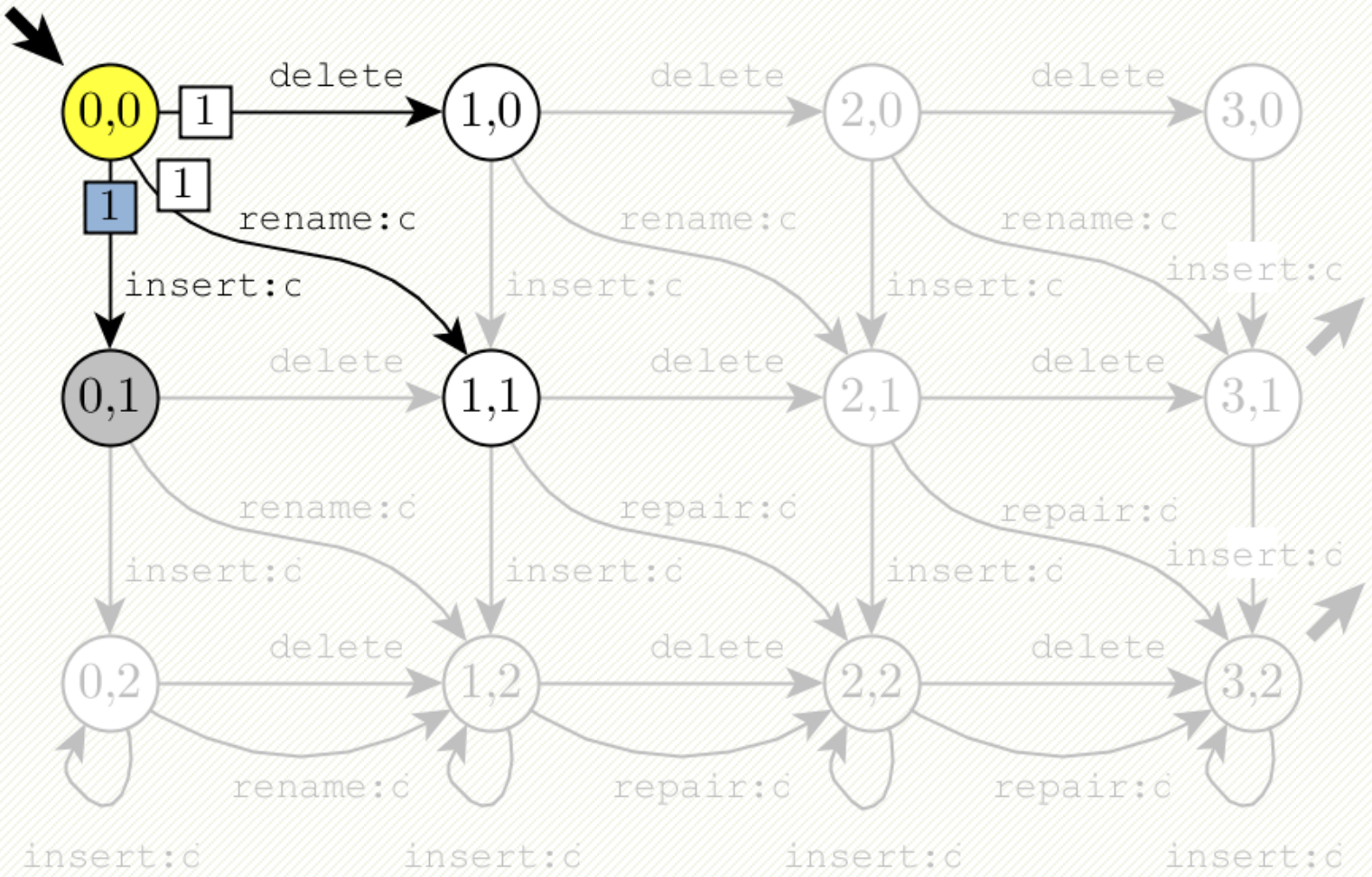
Refinement Strategy



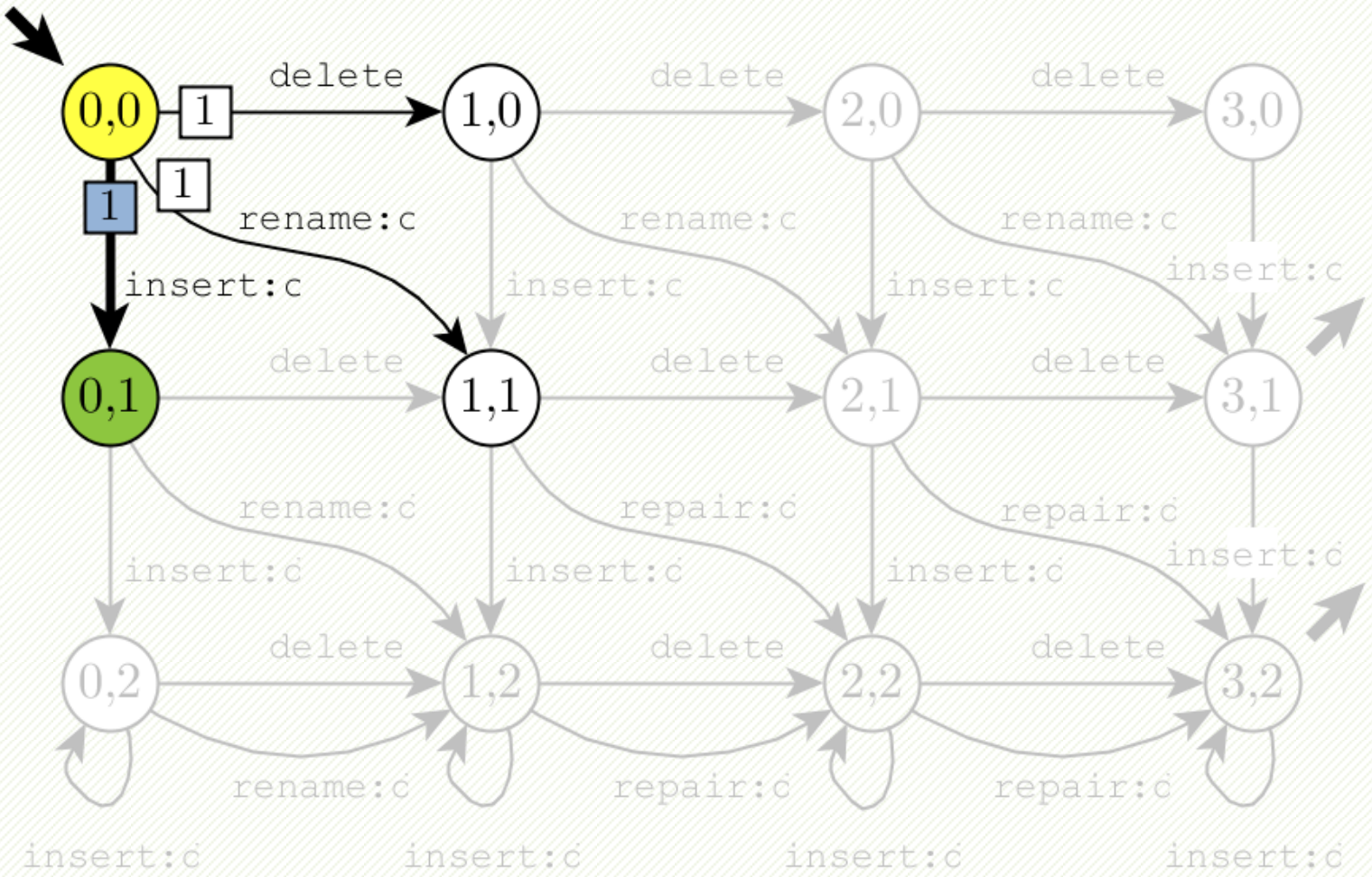
Refinement Strategy



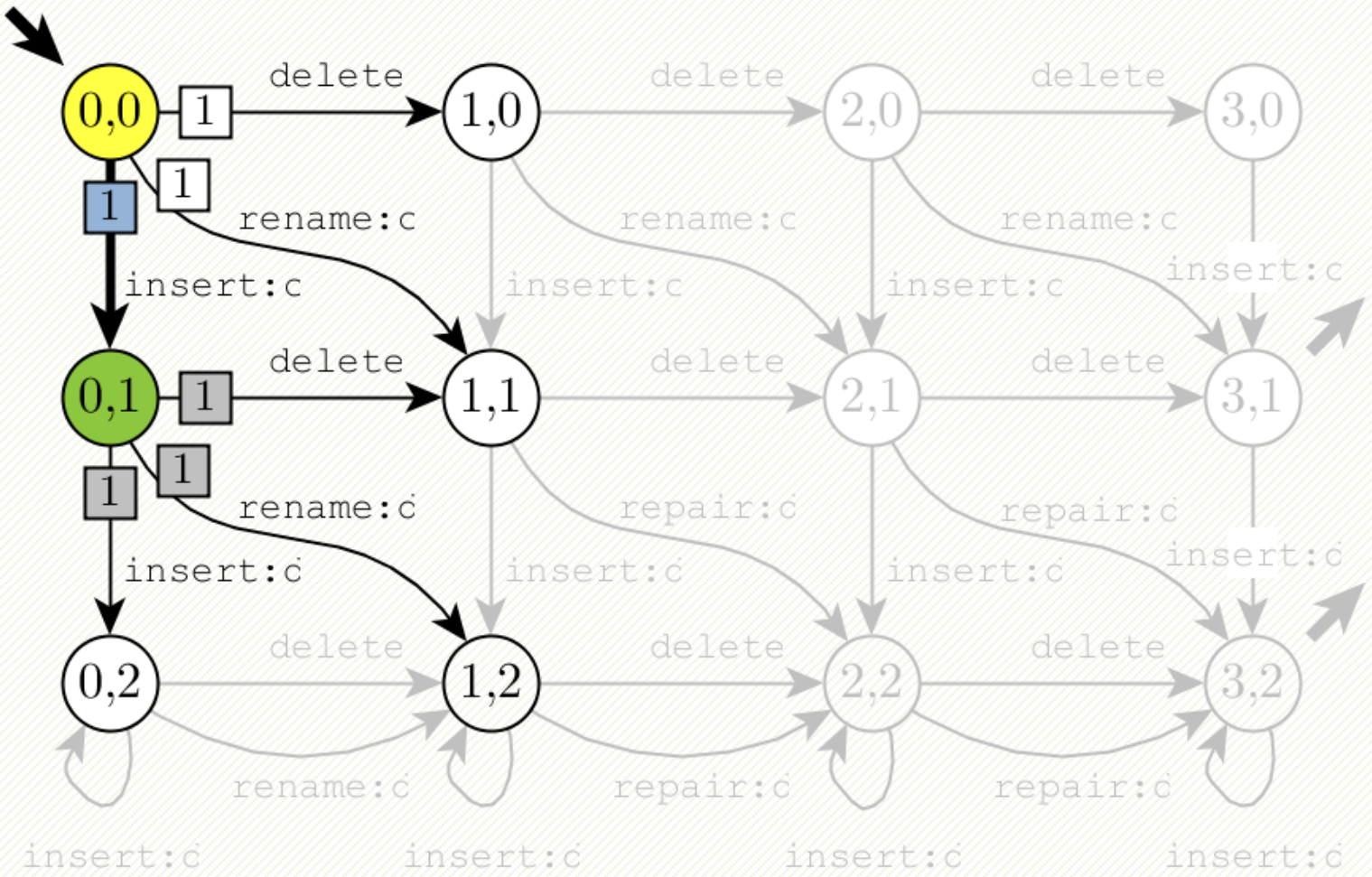
Refinement Strategy



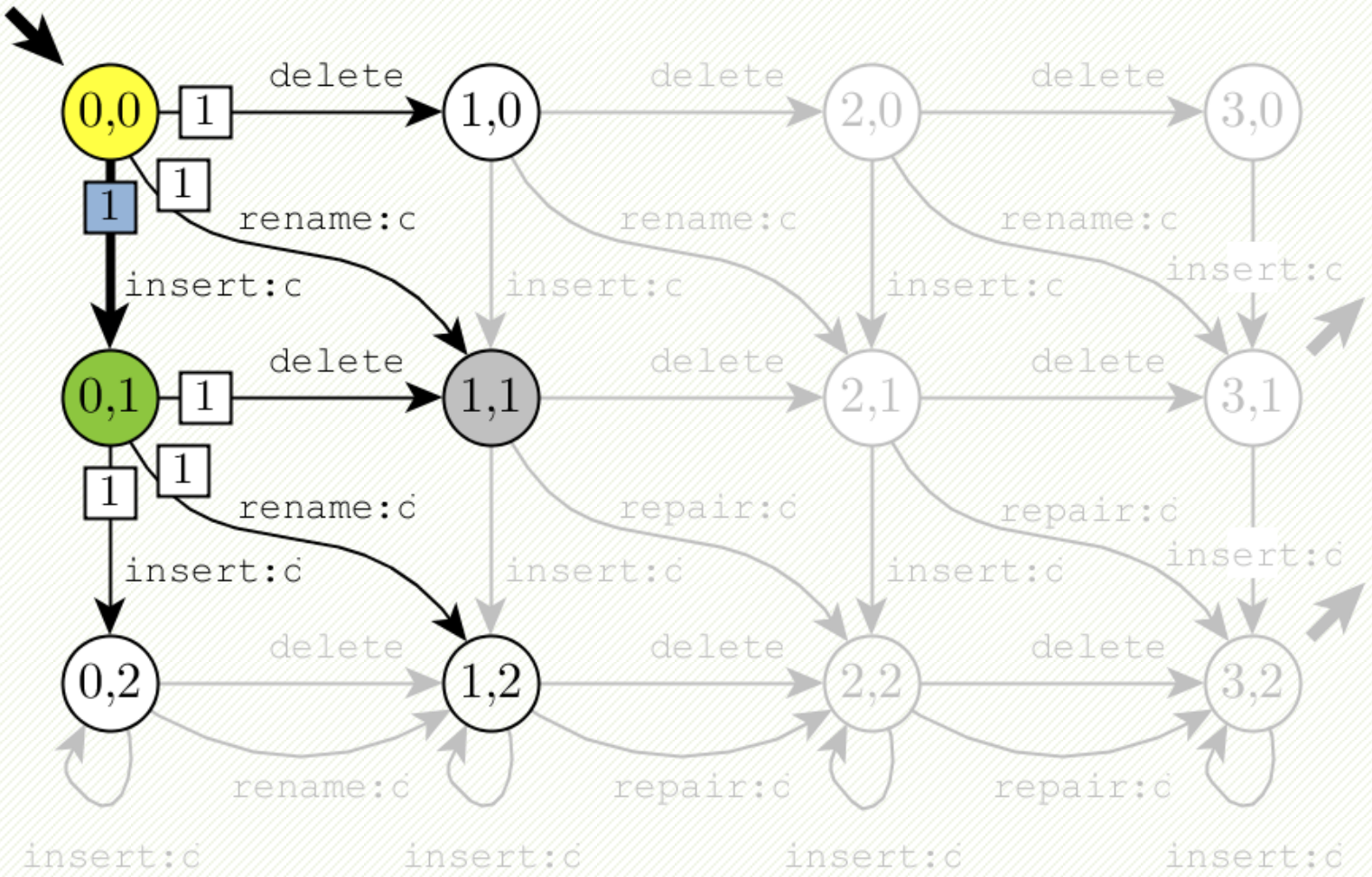
Refinement Strategy



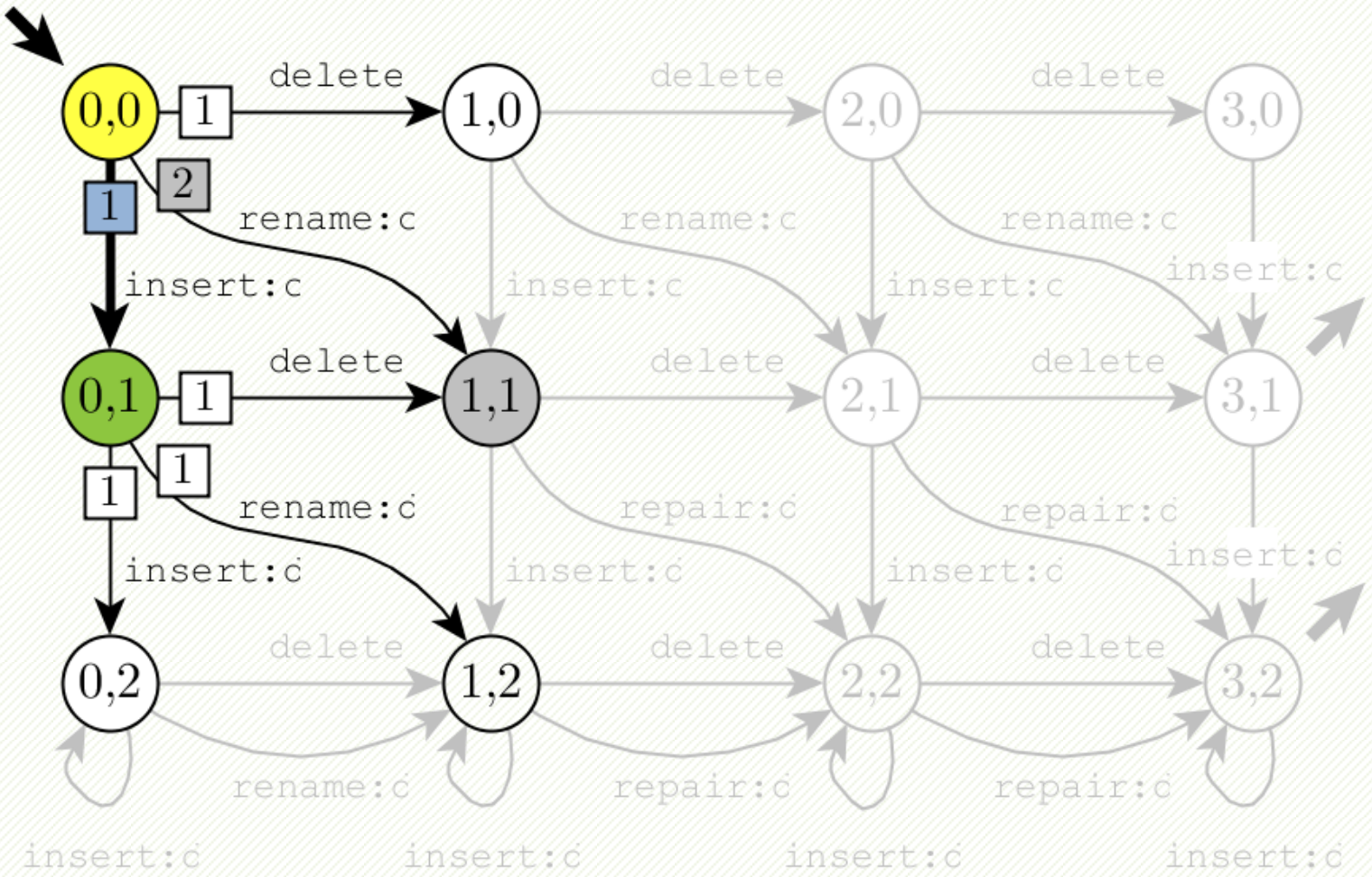
Refinement Strategy



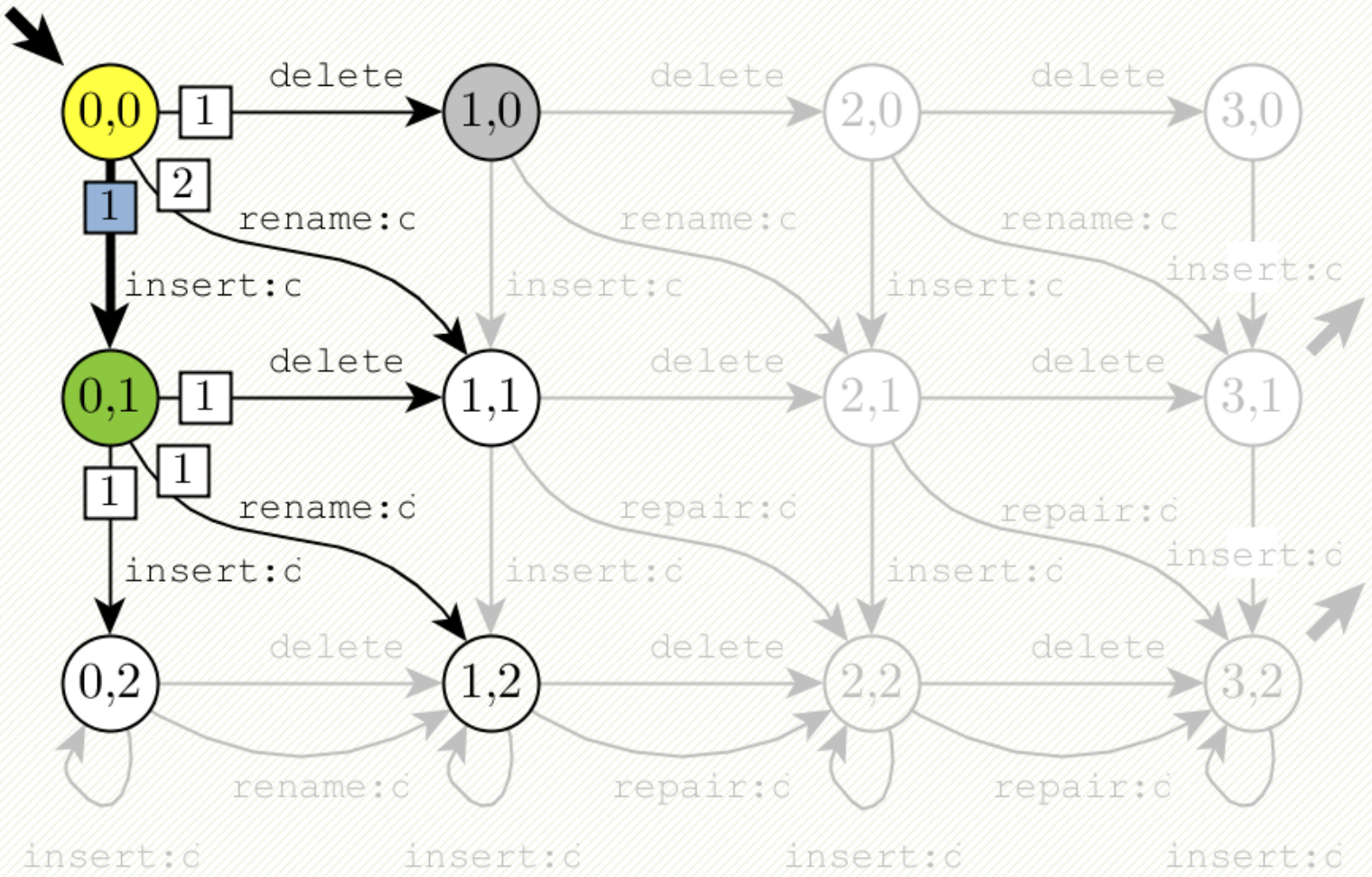
Refinement Strategy



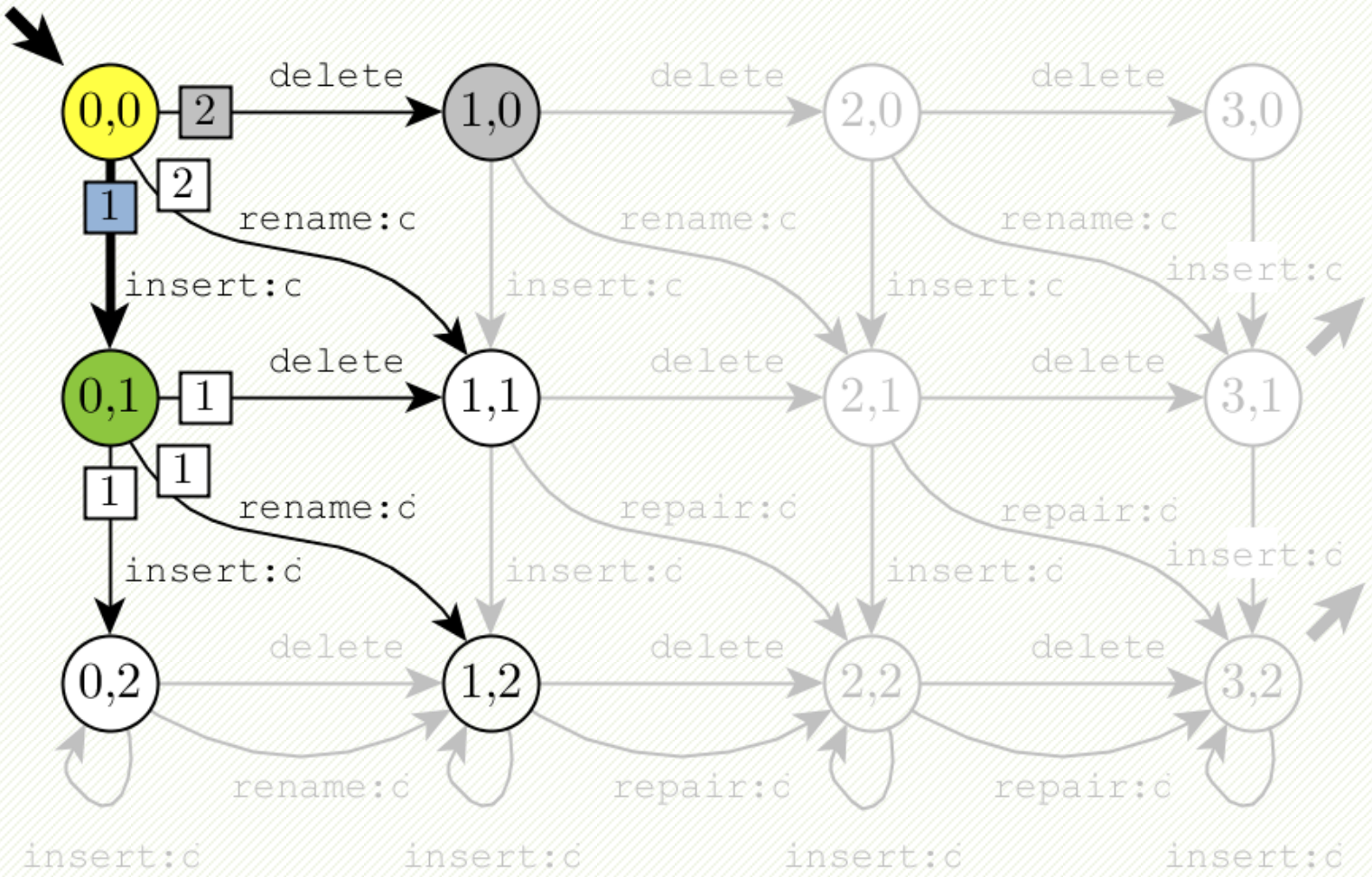
Refinement Strategy



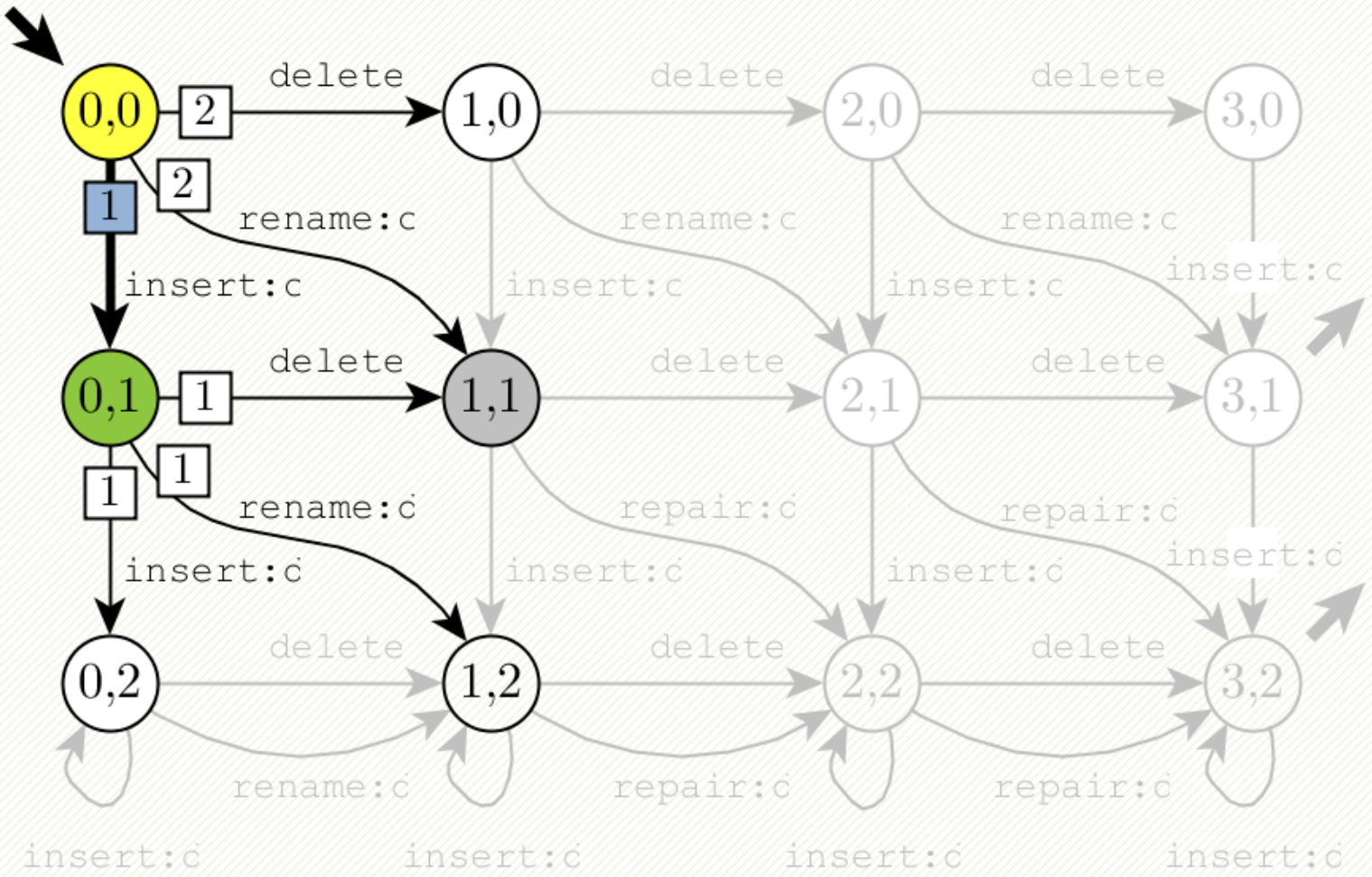
Refinement Strategy



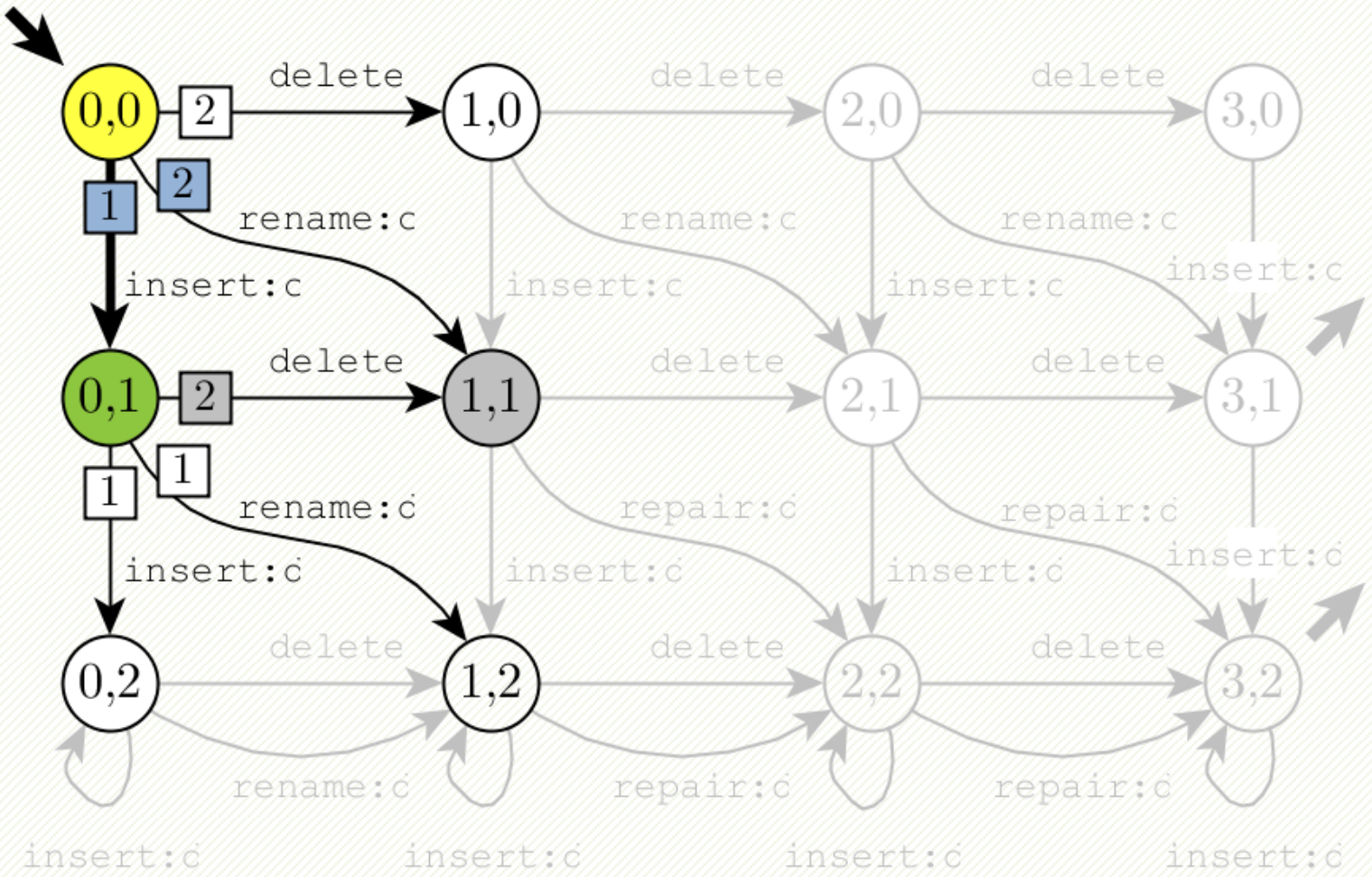
Refinement Strategy



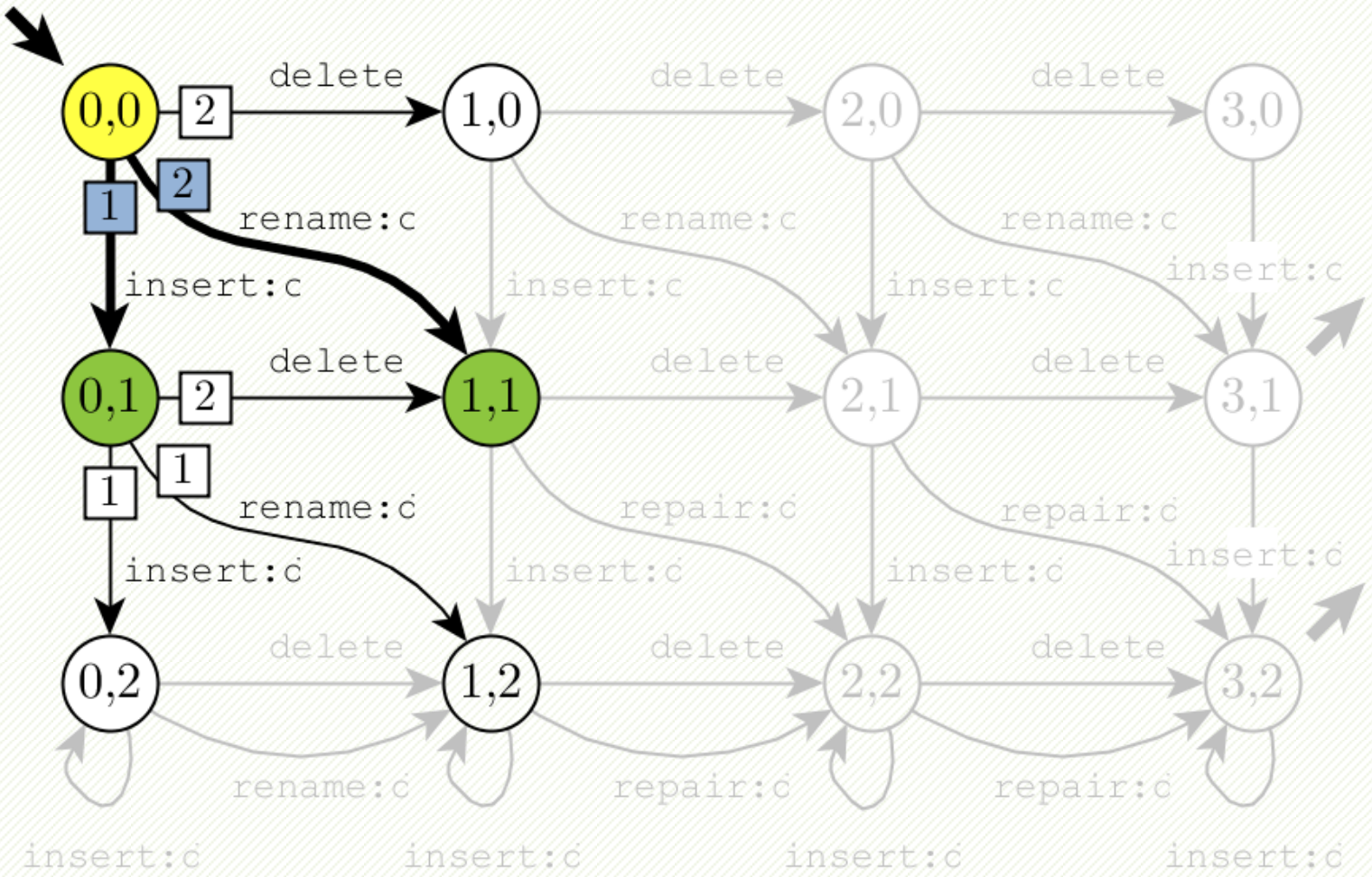
Refinement Strategy



Refinement Strategy



Refinement Strategy

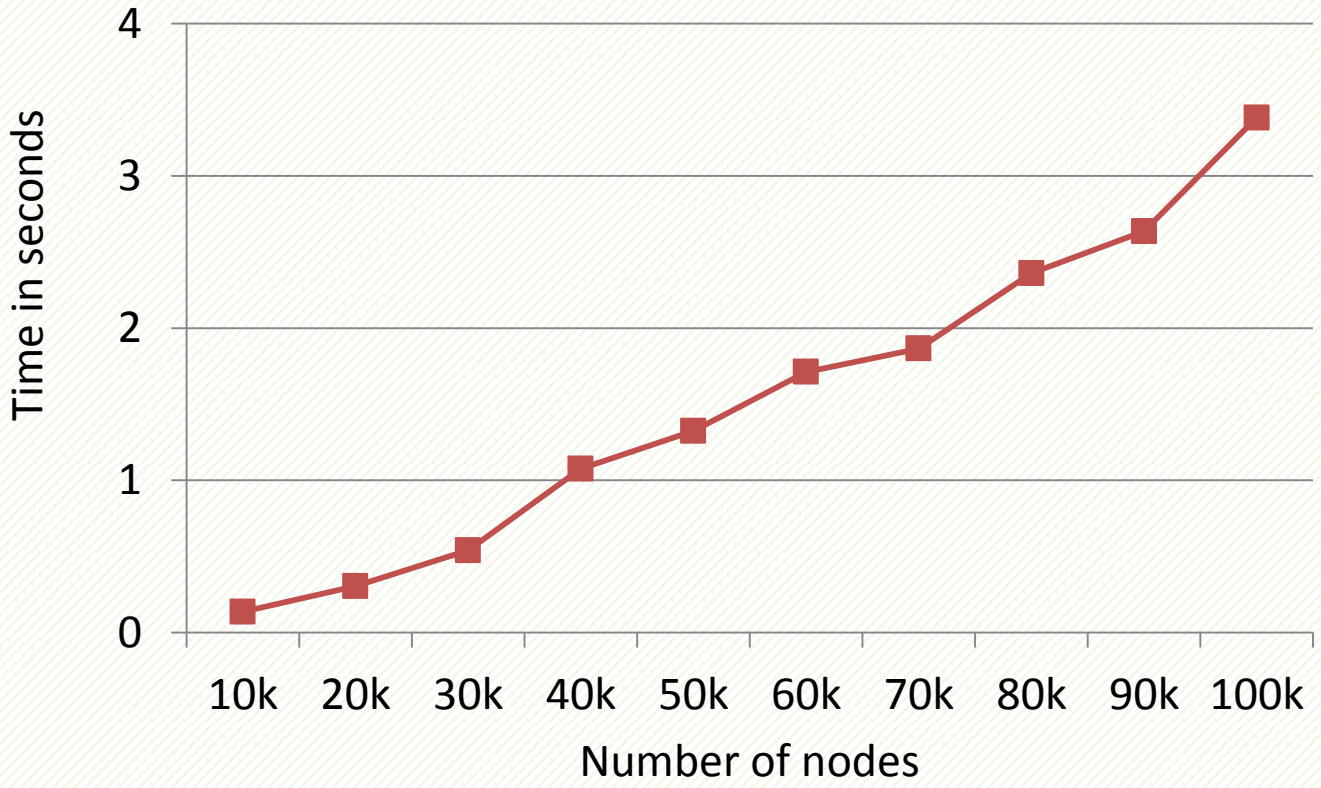


Refinement Strategy

- Exploration loop
  - **Complete vertex**
    - Explore outgoing edges
    - Obtain first cost estimations
    - Update current distances
  - **Incomplete vertex**
    - **Request refinement of open perspective ingoing edges**
      - Assign a quota to limit the allowed refinement progress

Execution times

- **Refinement strategy**



Conclusion

- **Features**
  - Regular tree grammars
  - Compact repair structure
  - All minimal corrections
  - No parameters required
  - Nearly linear algorithms

Thank you for your attention…

Faculty of Mathematics and Physics

Charles University in Prague