Recomputation

Constraint Programming, lecture 7

Marco Kuhlmann, Guido Tack

Recapitulation

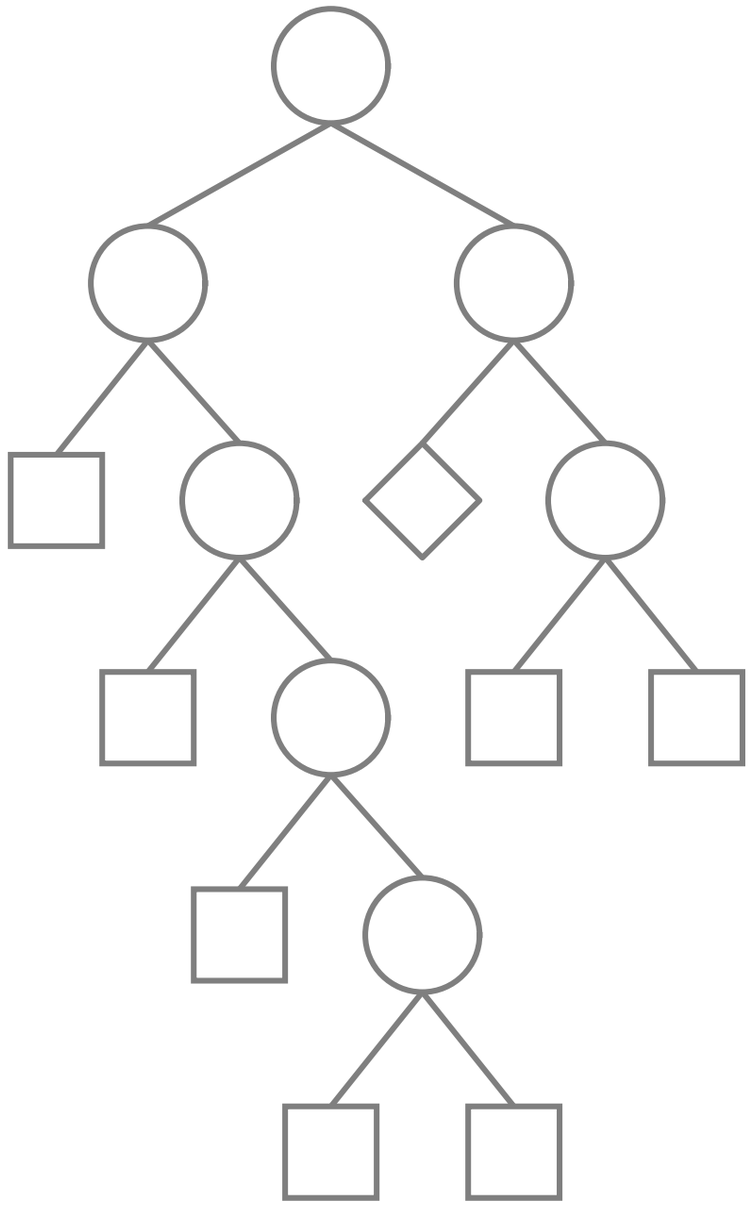
- separate propagation and branching from search
- architecture for search:

  computation spaces
- simple primitives, complex search engines

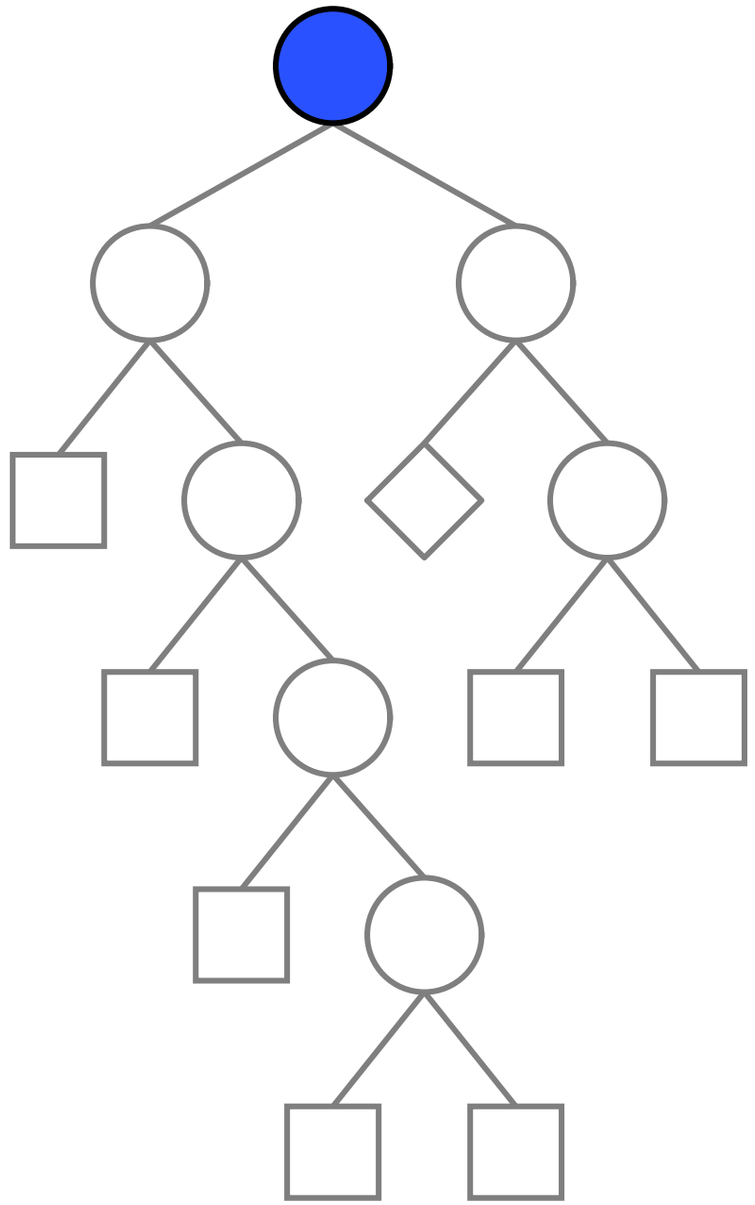
Recapitulation

- search needs backtracking

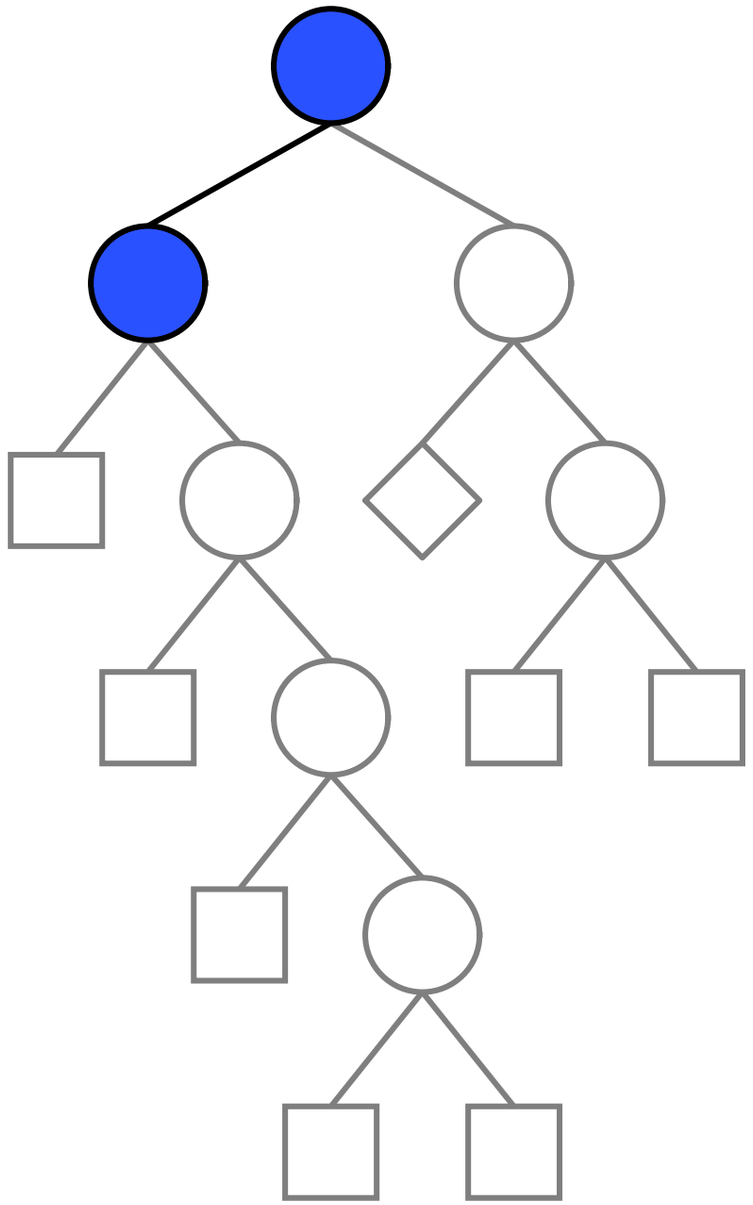
Backtracking



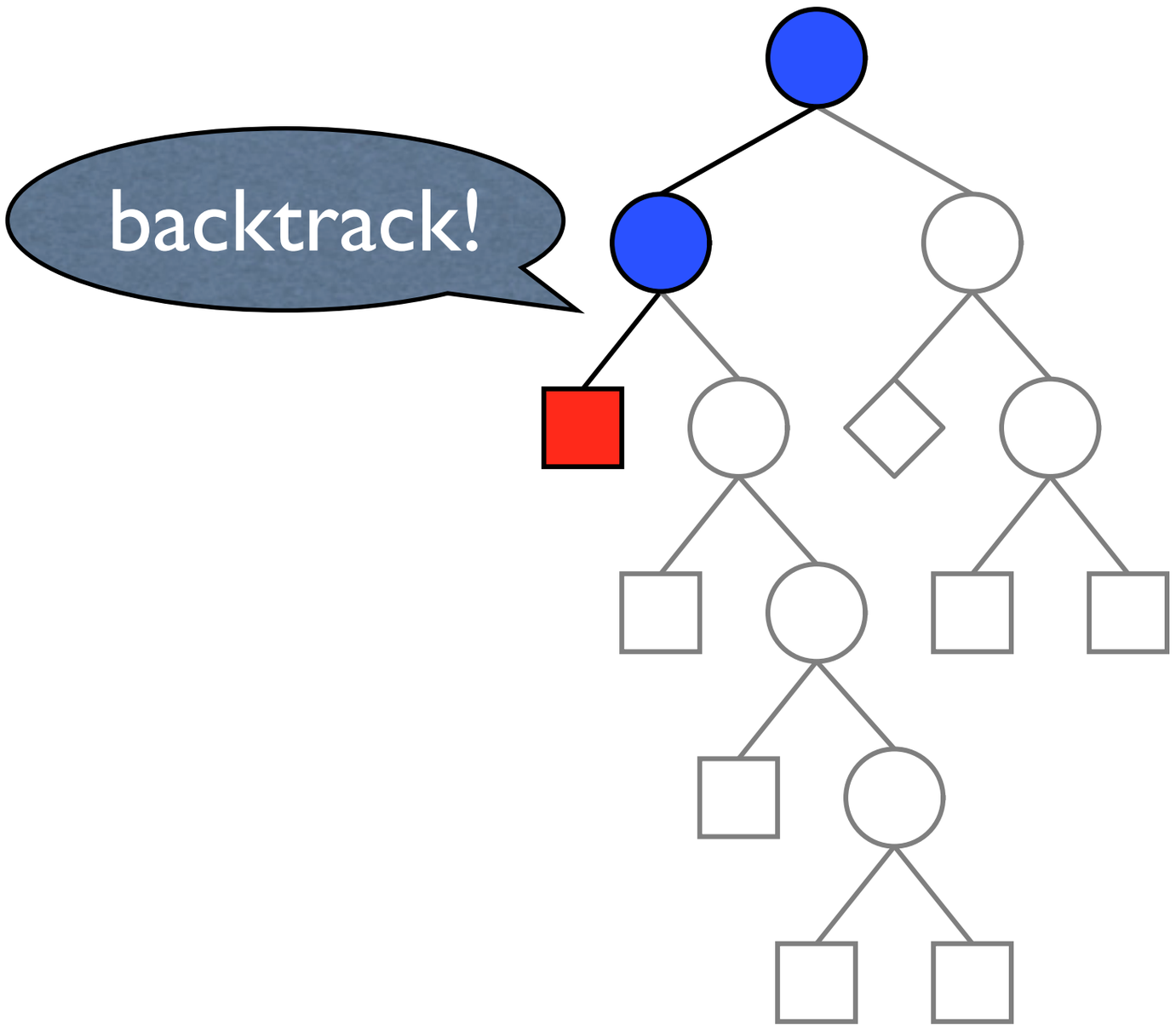
Backtracking



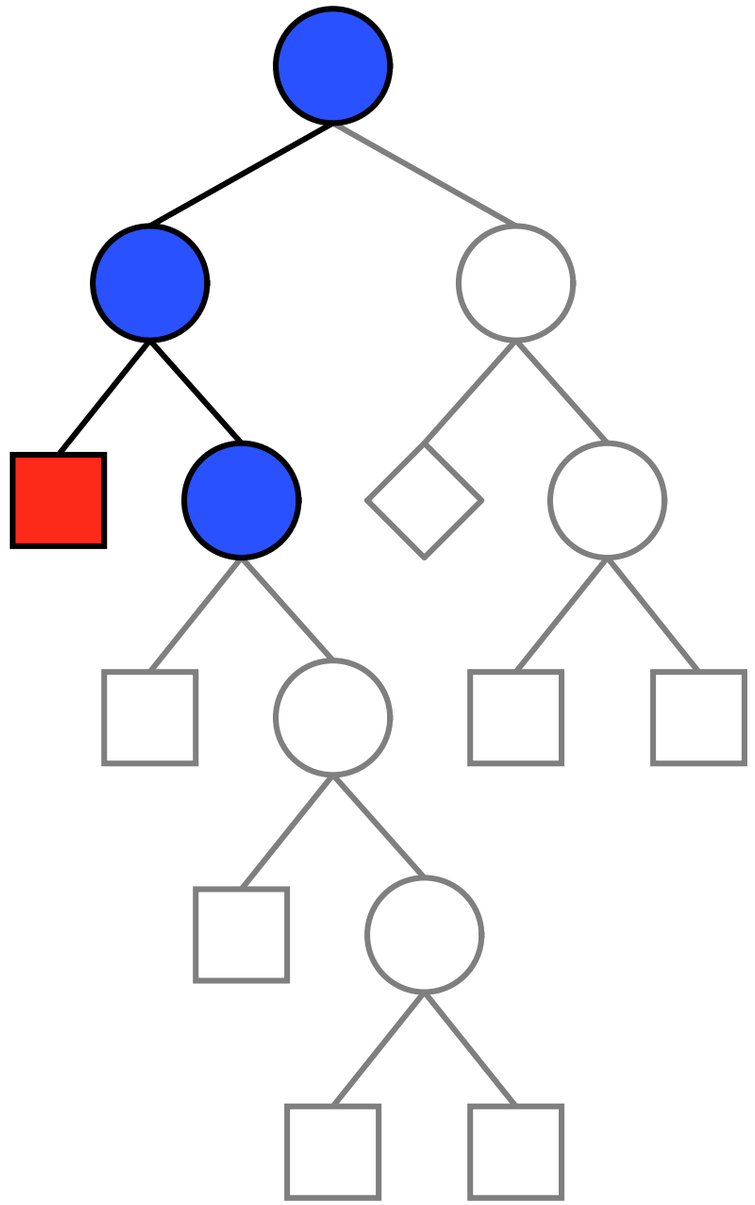
Backtracking



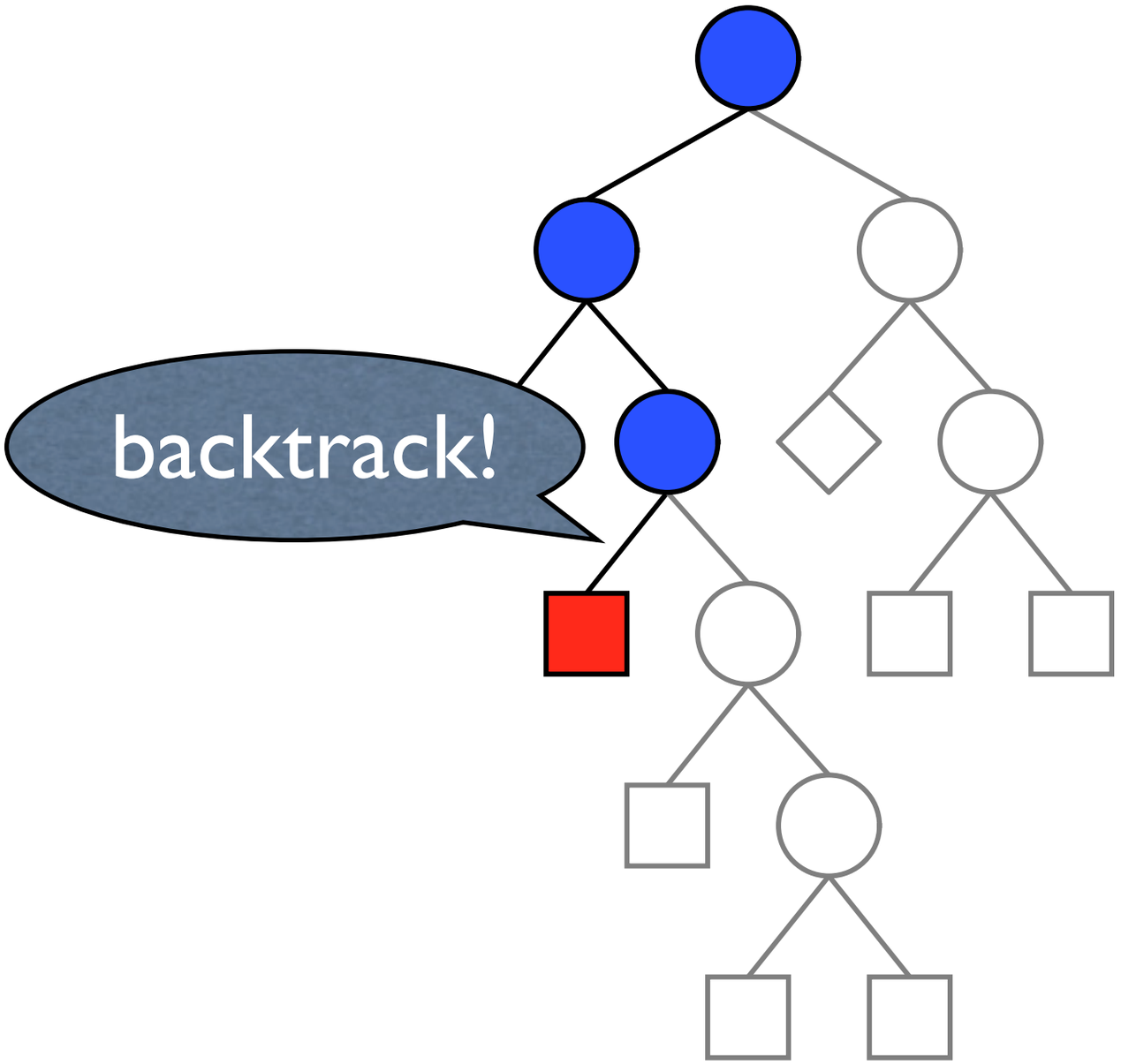
Backtracking



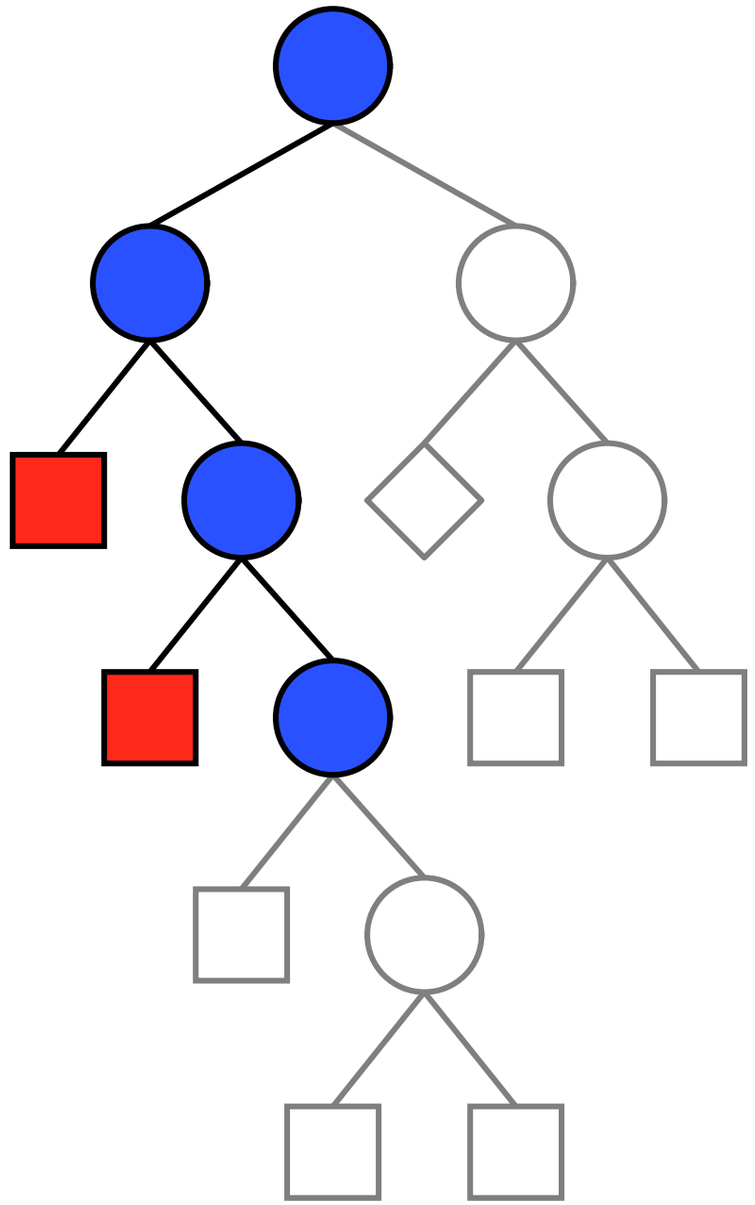
Backtracking



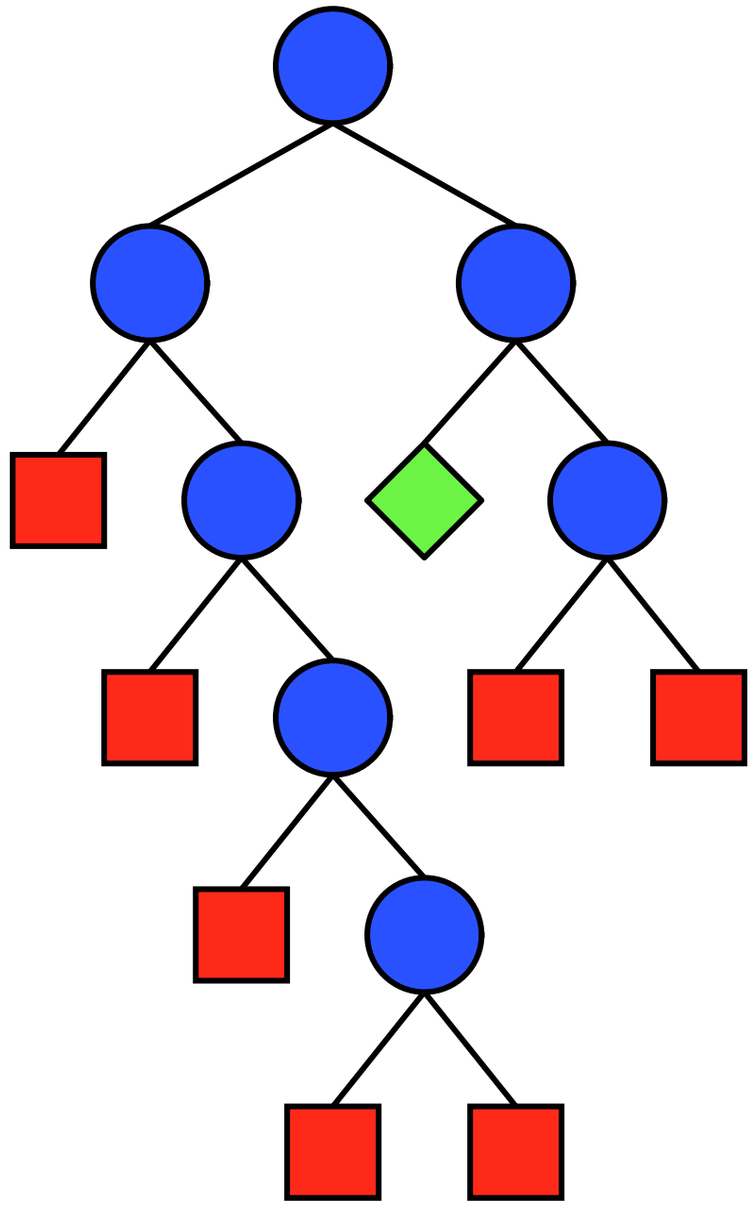
Backtracking



Backtracking



Backtracking



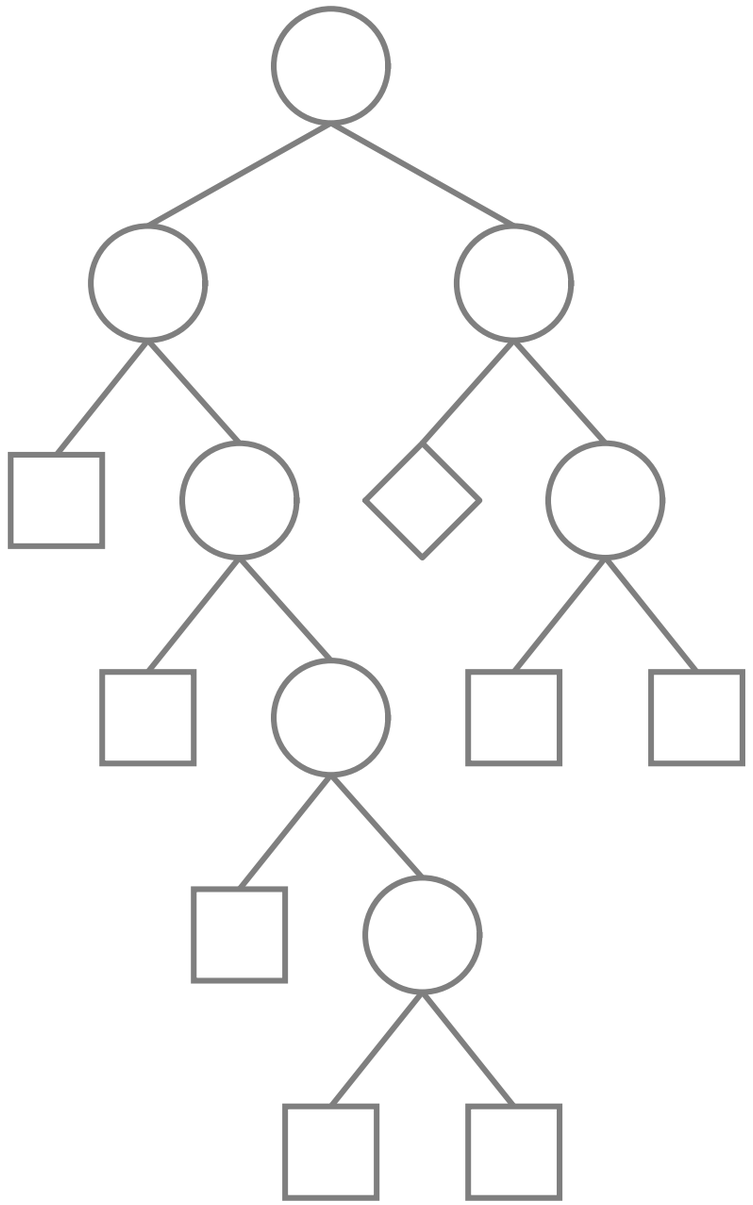
Recapitulation

- search needs backtracking
- strategies:
  - copying
  - trailing
  - recomputation

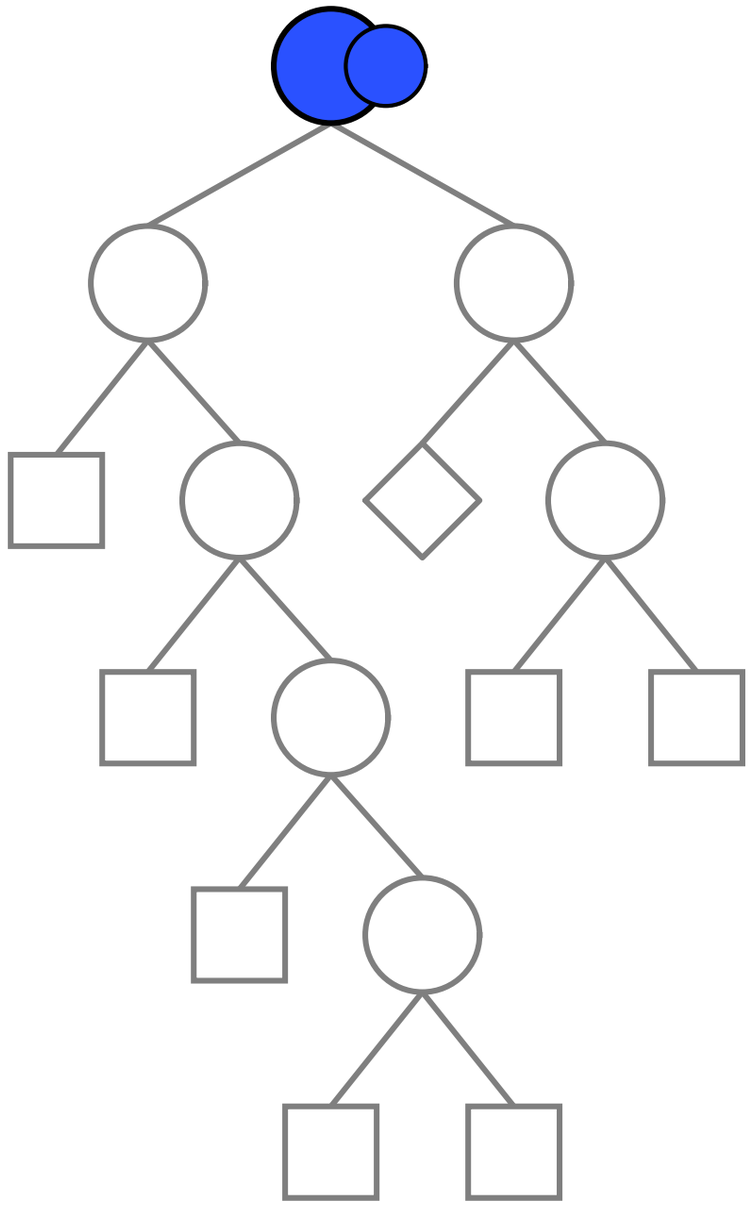
Recapitulation

```
fun dfs (s) =
  case status s of
    FAILED => nil
  | SOLVED => [s]
  | BRANCH =>
      let val c = clone s in
        commit (s, 1);
        commit (c, 2);
        dfs s @ dfs c
      end
```

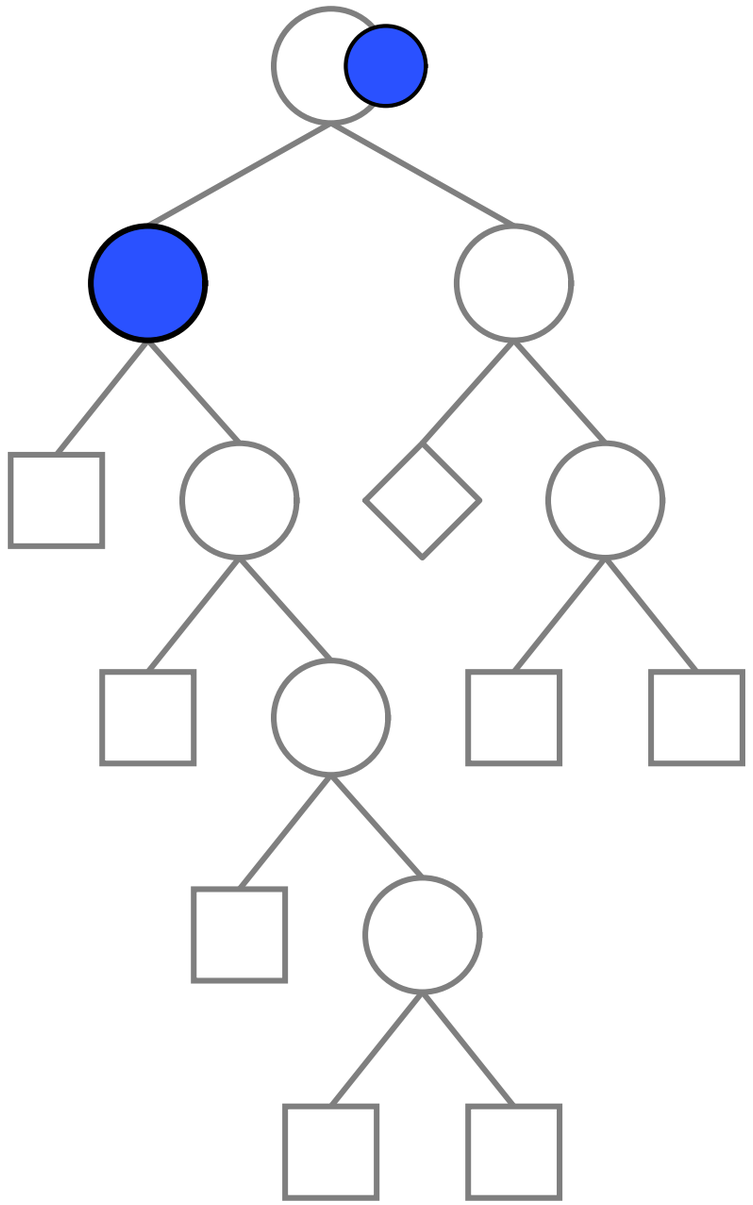
Copying



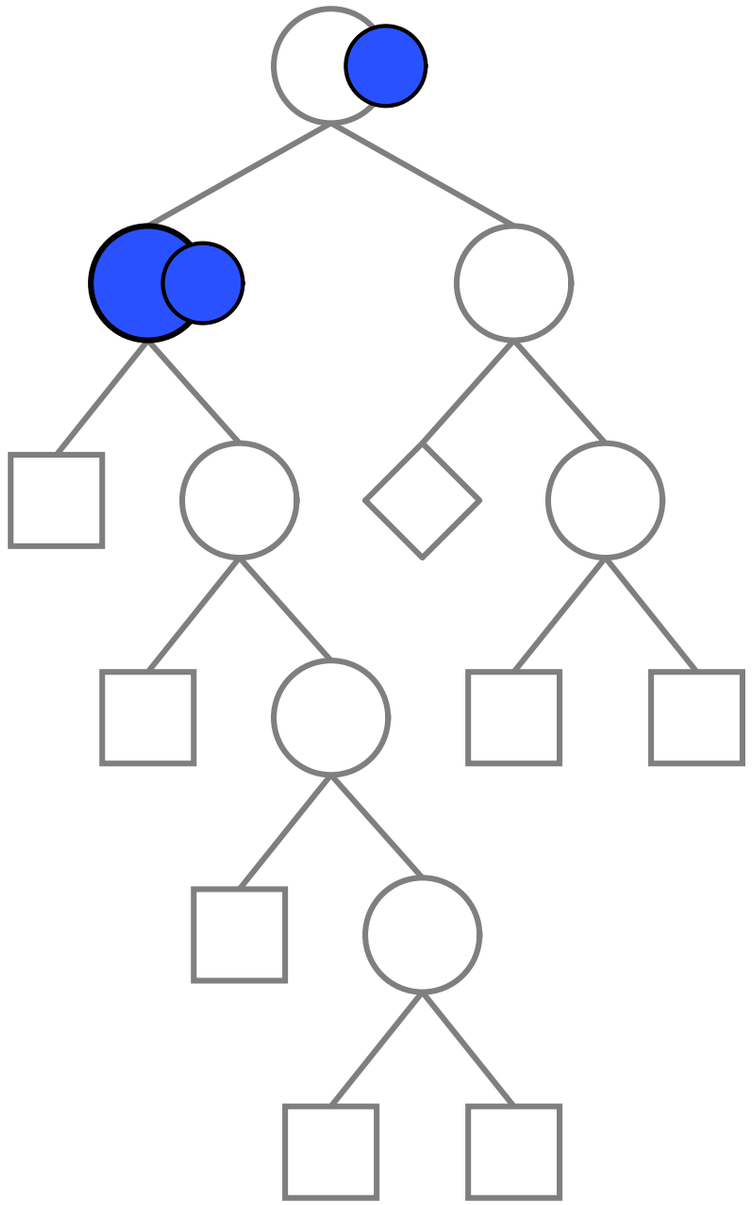
Copying



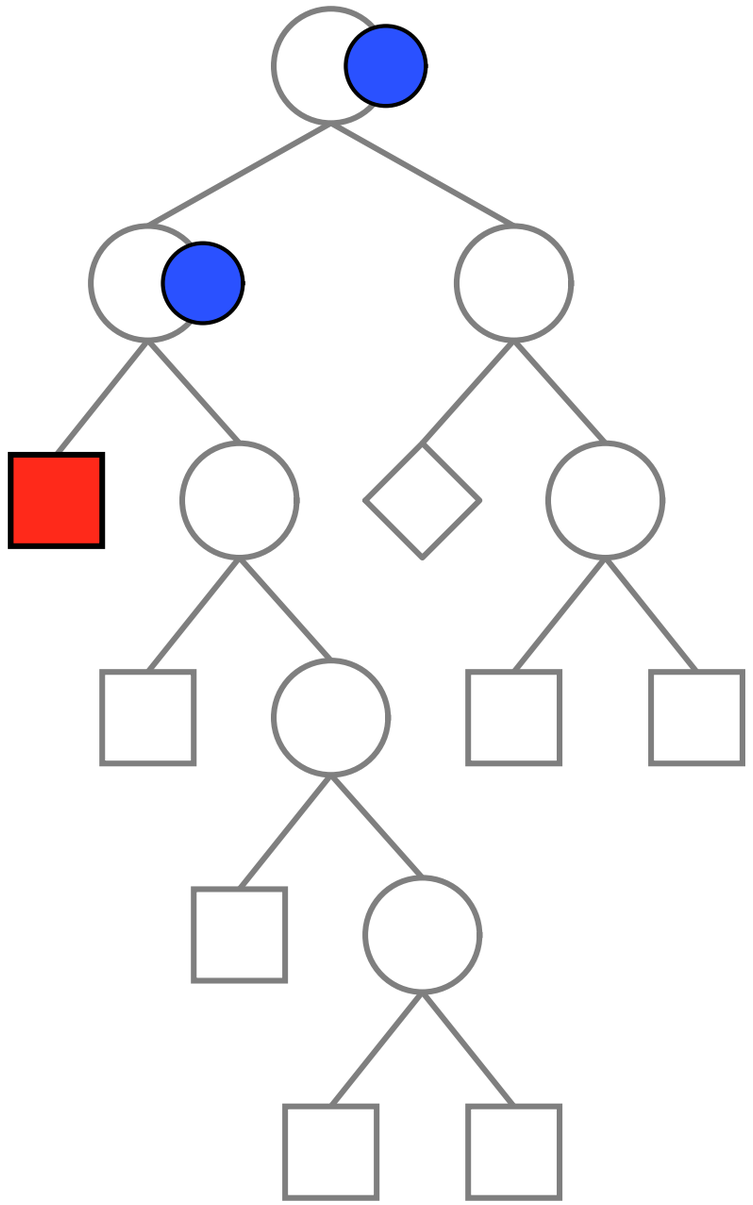
Copying



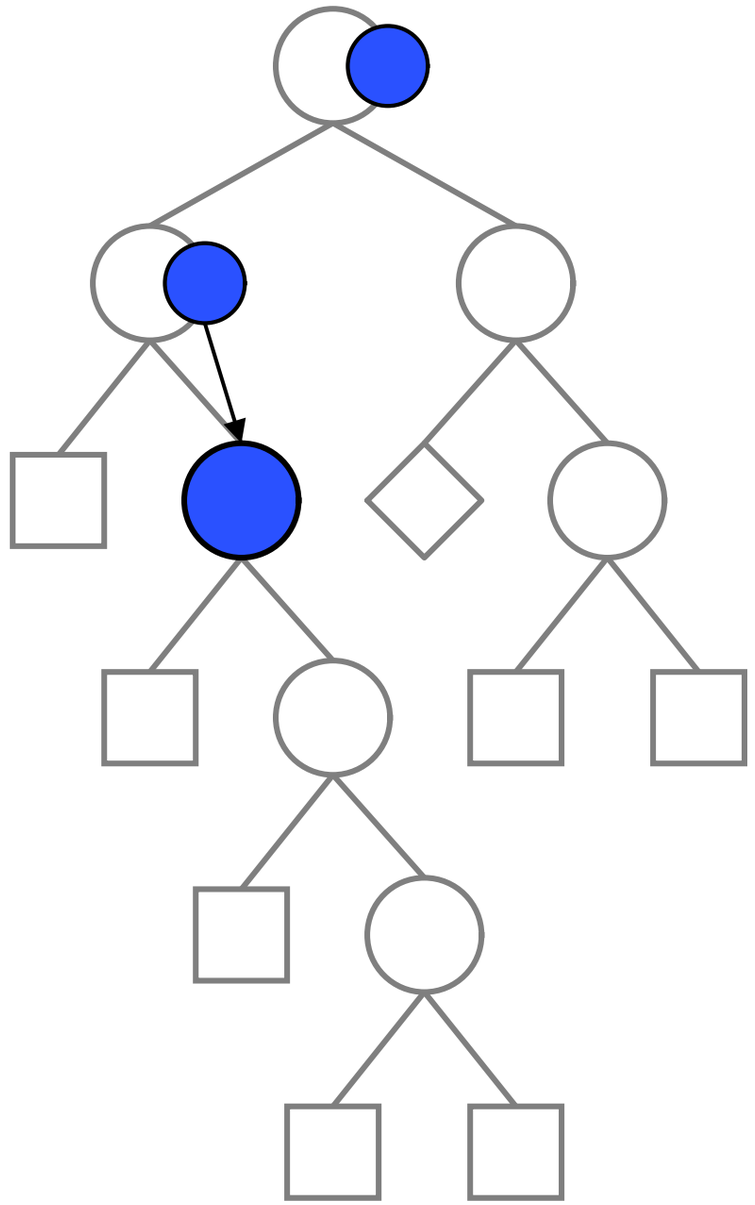
Copying



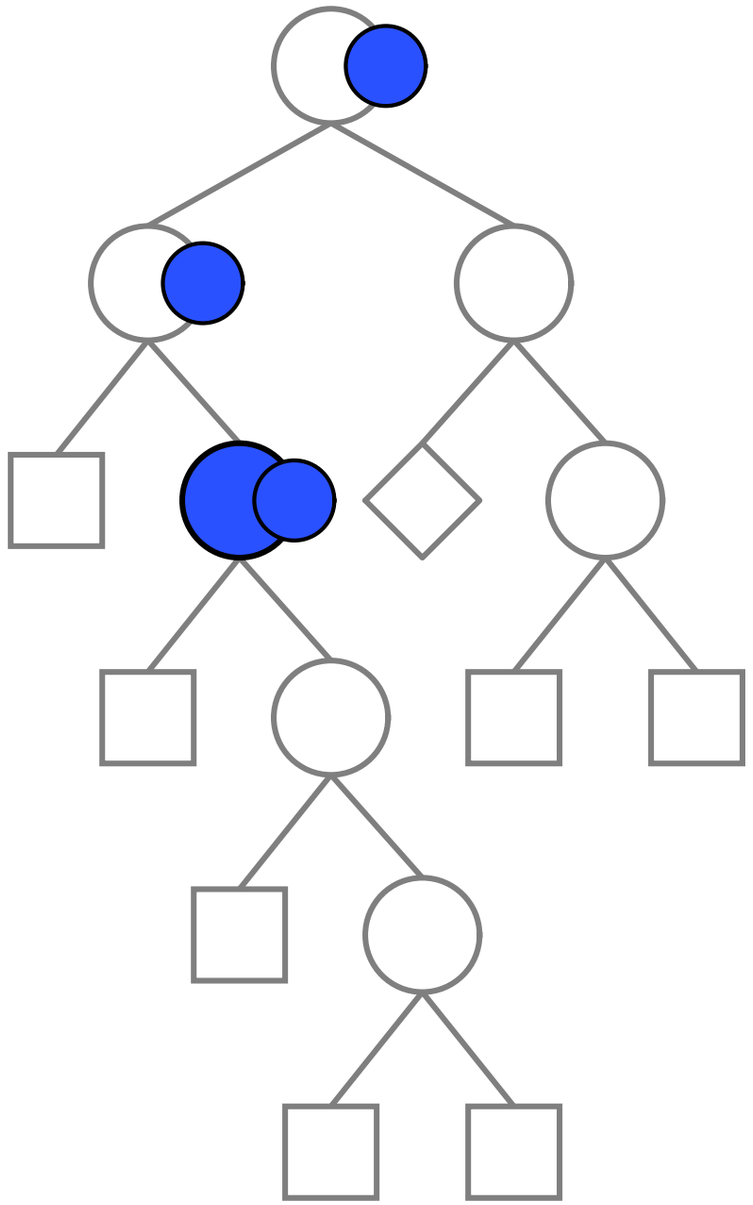
Copying



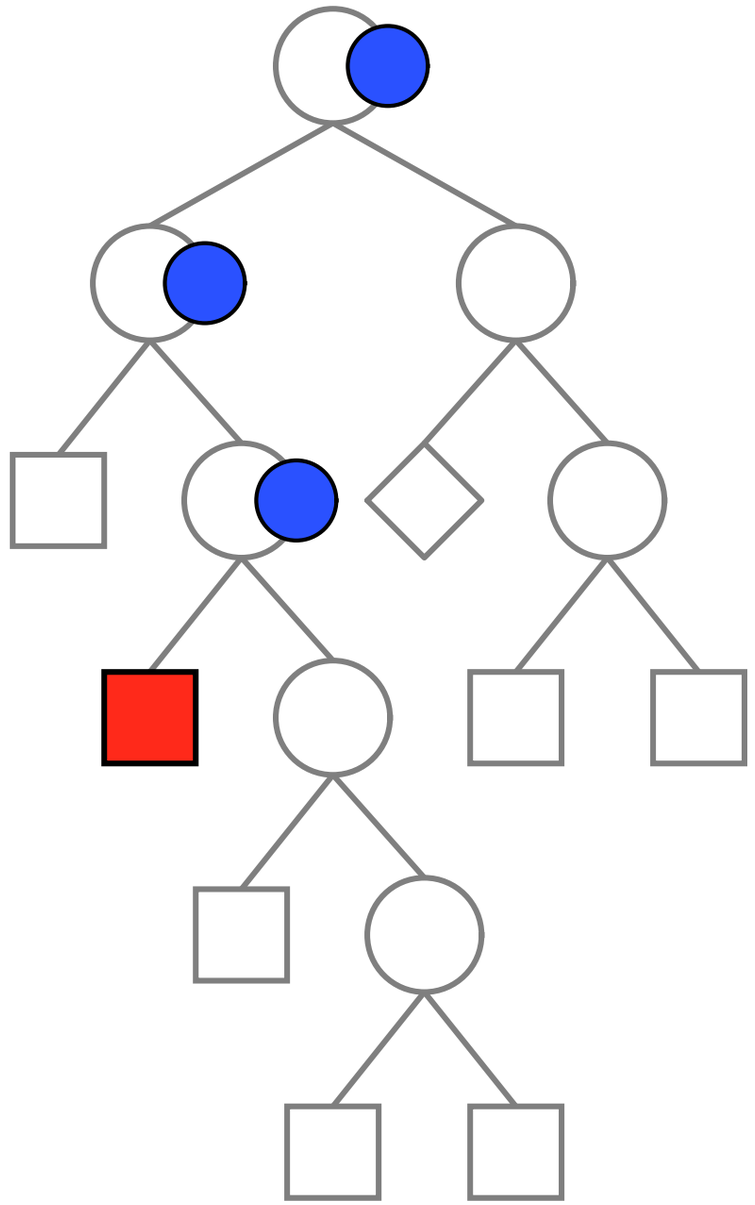
Copying



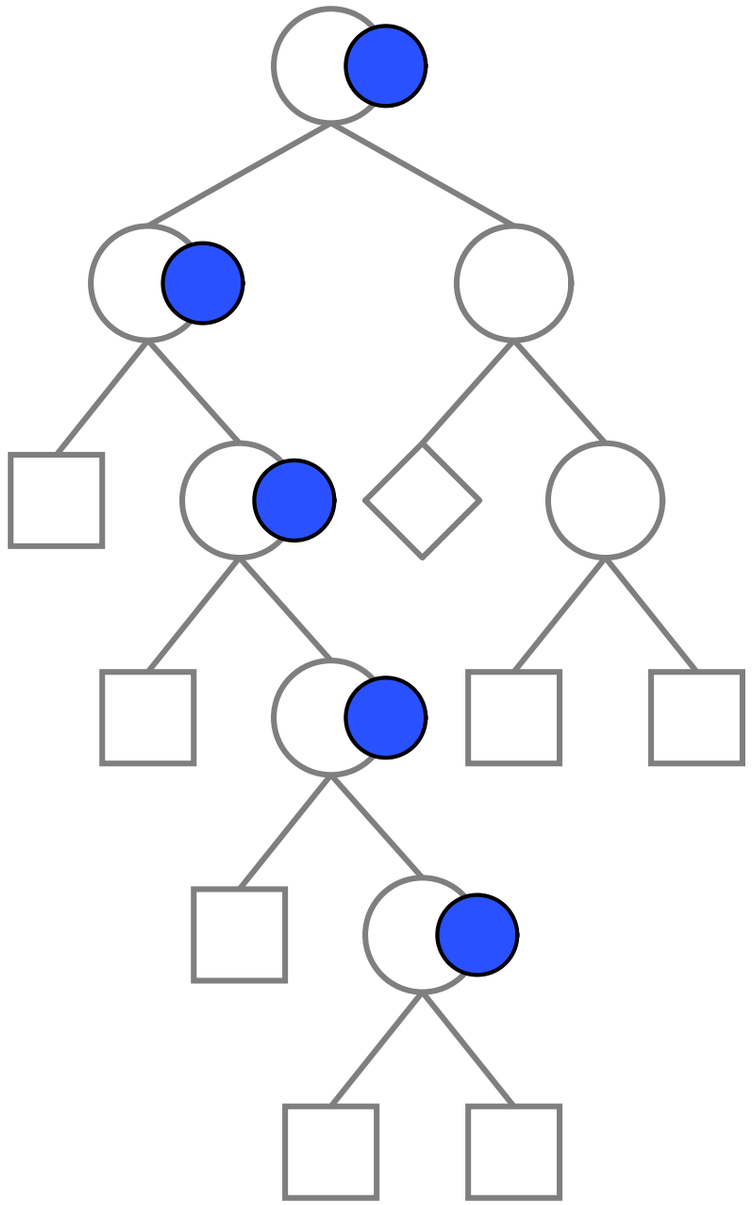
Copying



Copying



Copying



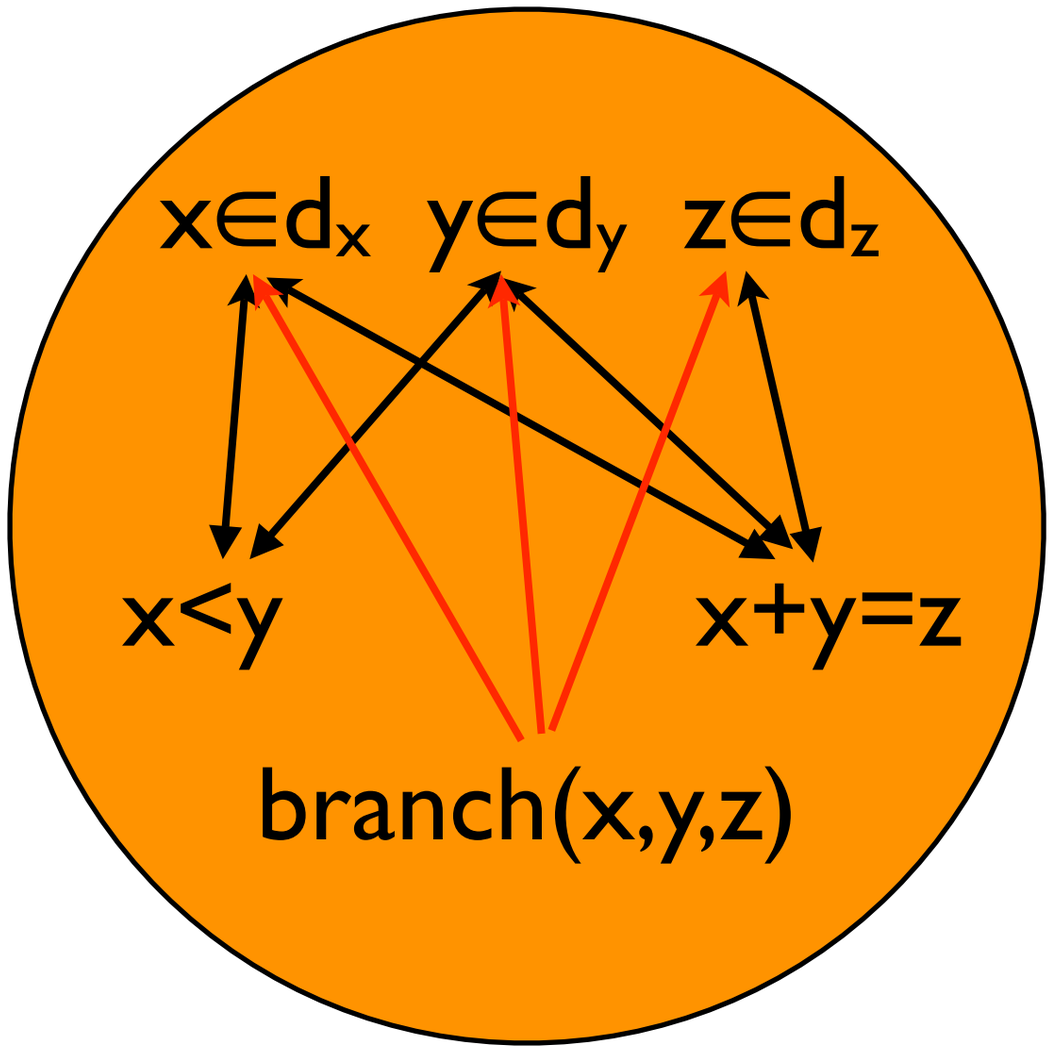
Recapitulation

- search needs backtracking
- strategies:
  - copying
  - trailing
  - recomputation

Today

- what's in a space?
- recomputation

What's in a space?



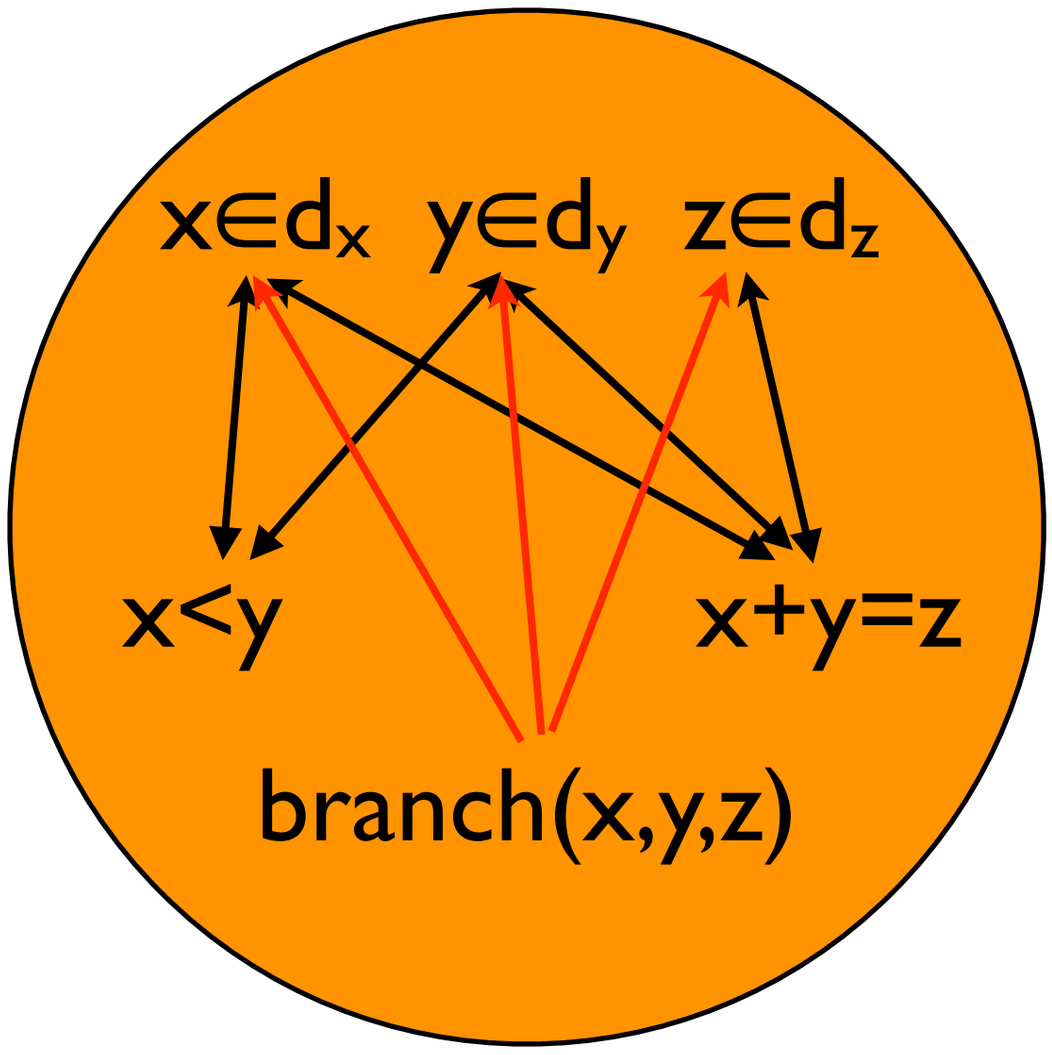
Space

Modified Variables

Queue

Control

What's in a space?



Space

Memory consumption:

$\sum$domains+
$\sum_{var}$ #prop(var)+
#propagators +
#branchings

~1000 variables

~10000 propagators

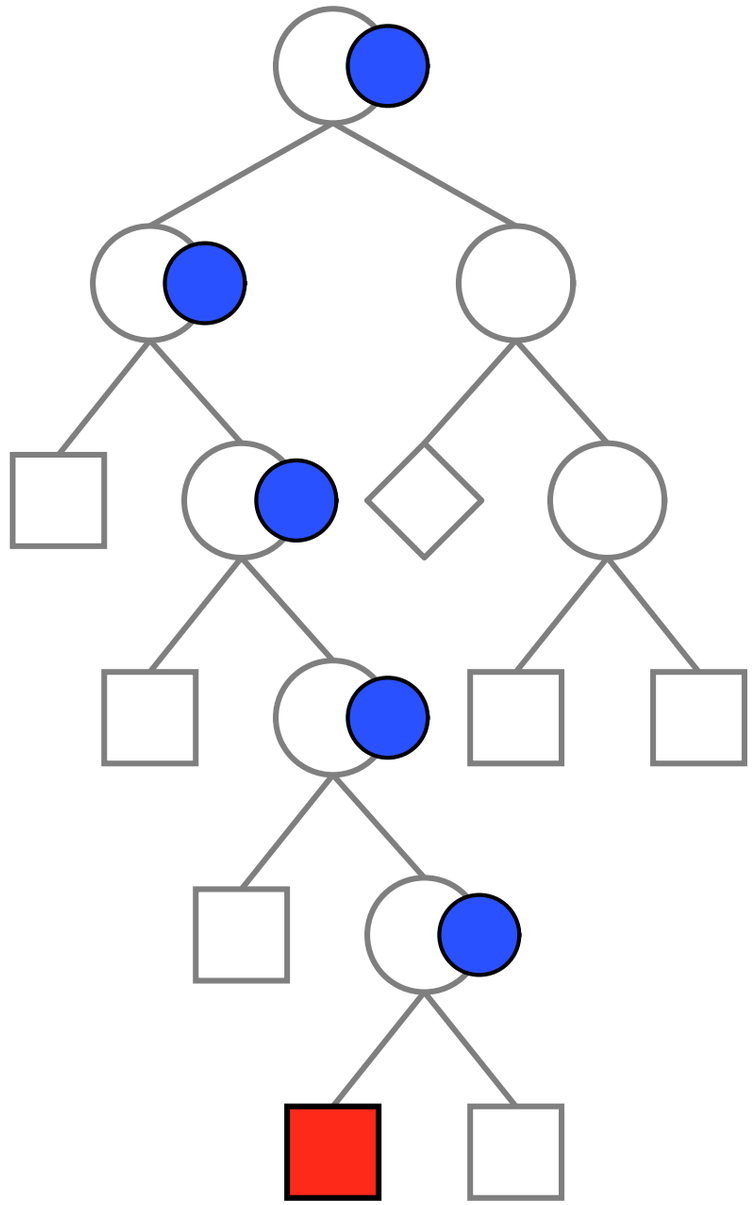
⇒ several MByte *per space*

Recomputation

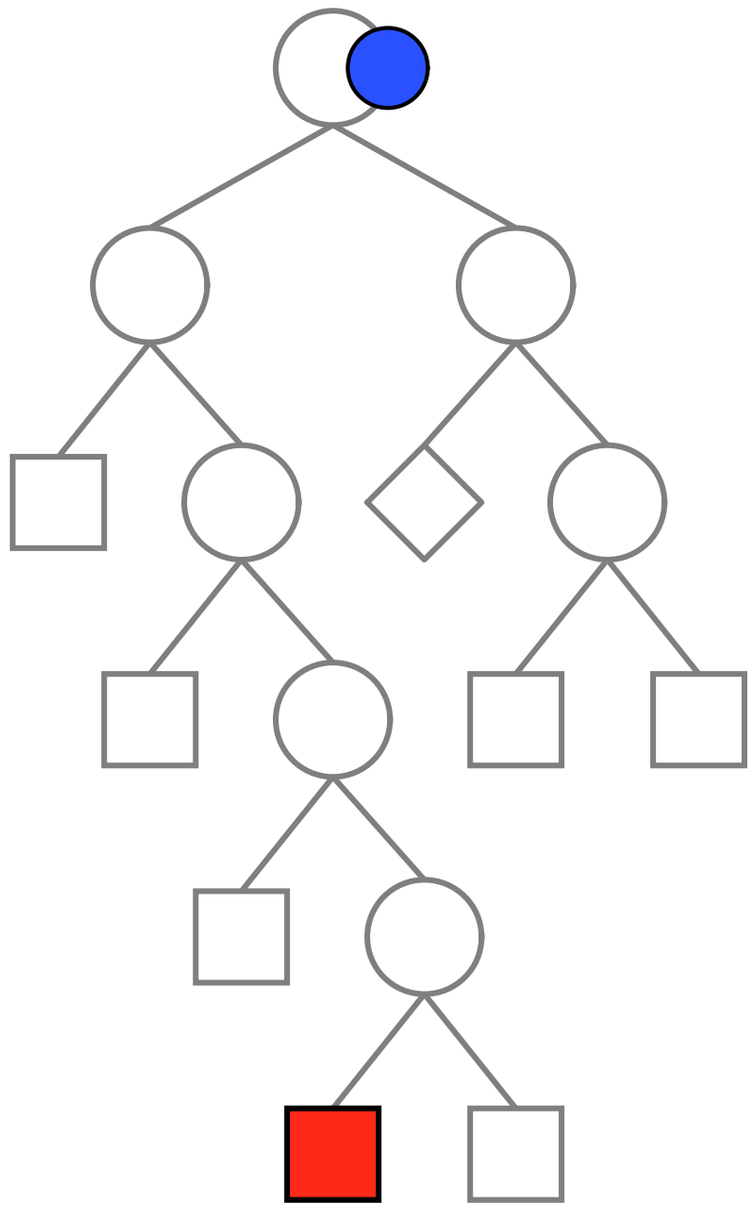
- Fundamental idea:

  trade space for time

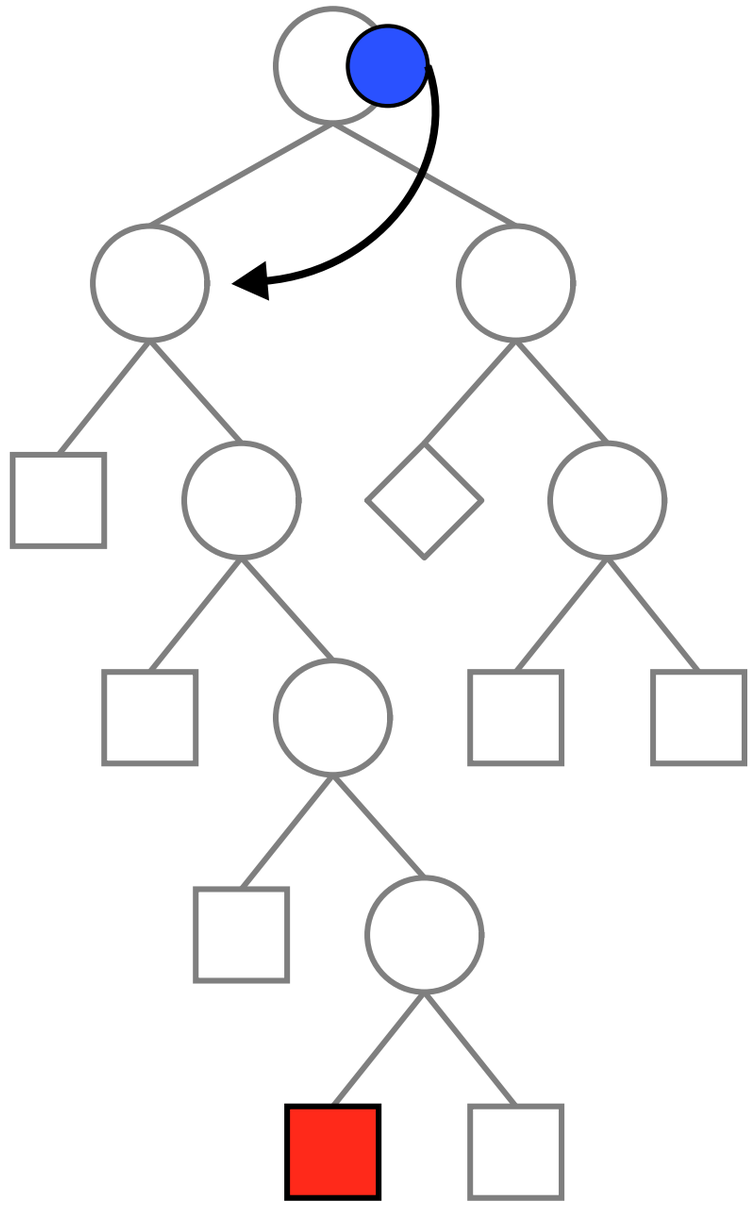
Full recomputation



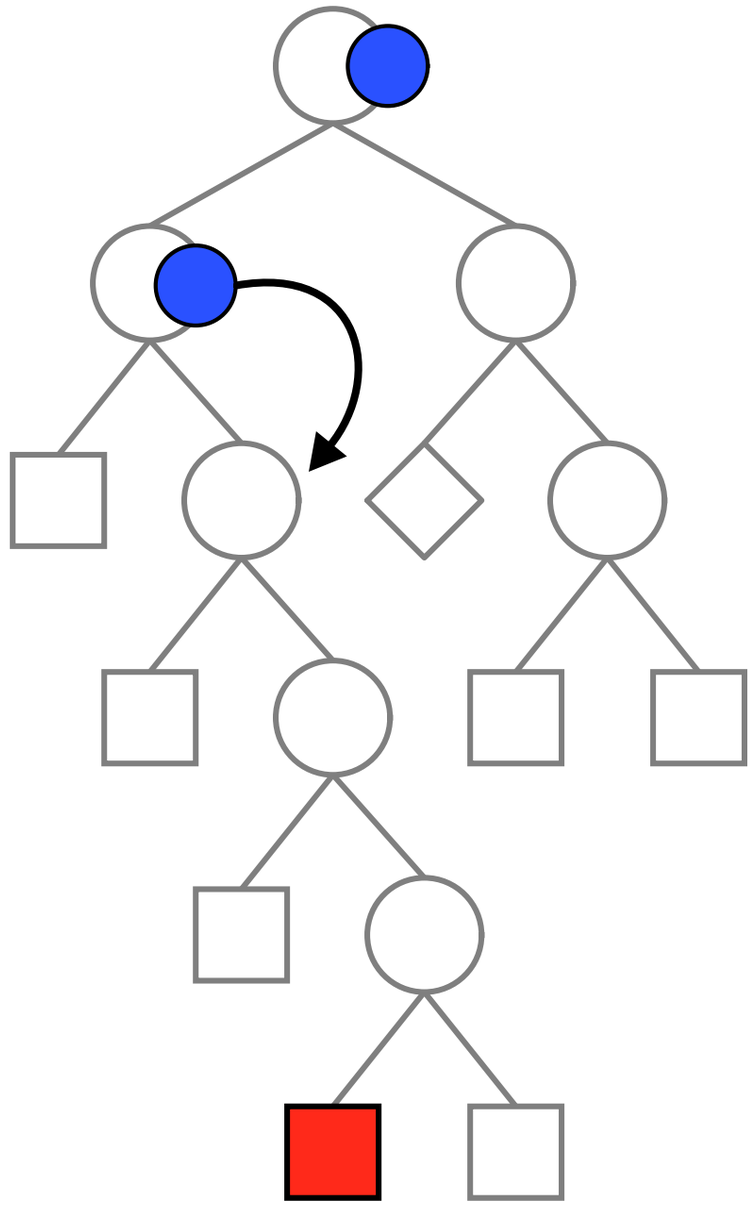
Full recomputation



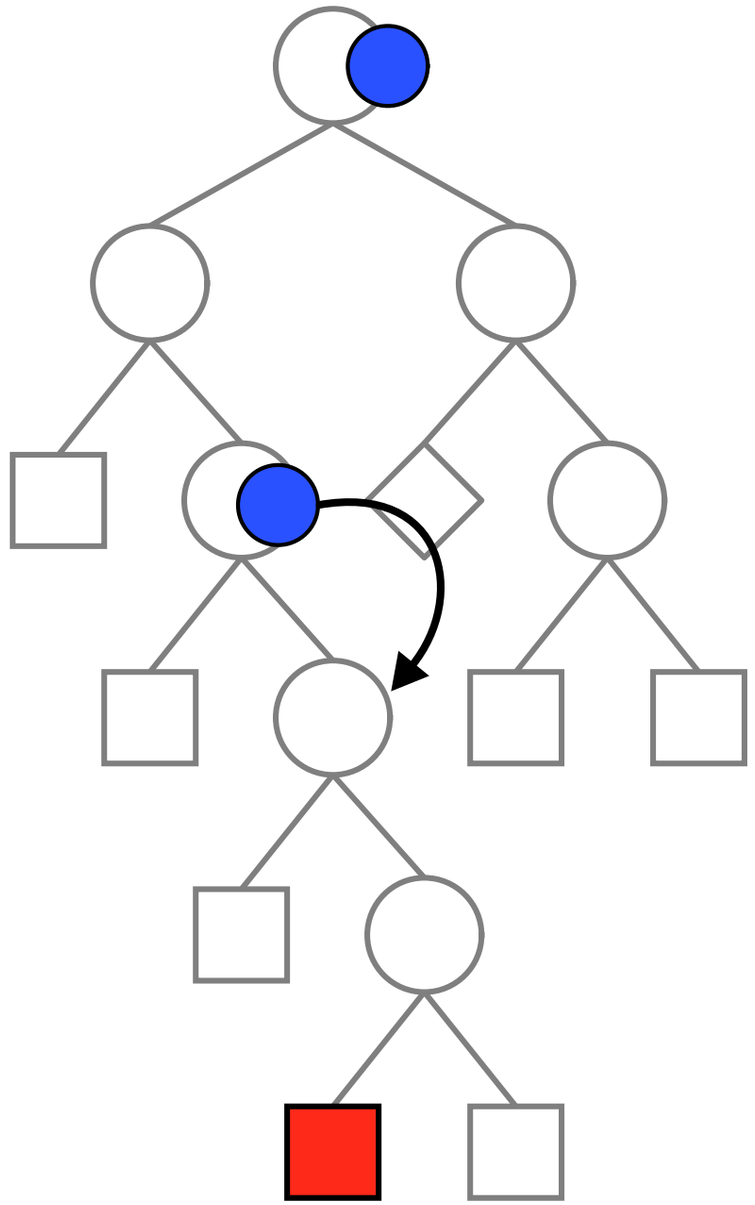
Full recomputation



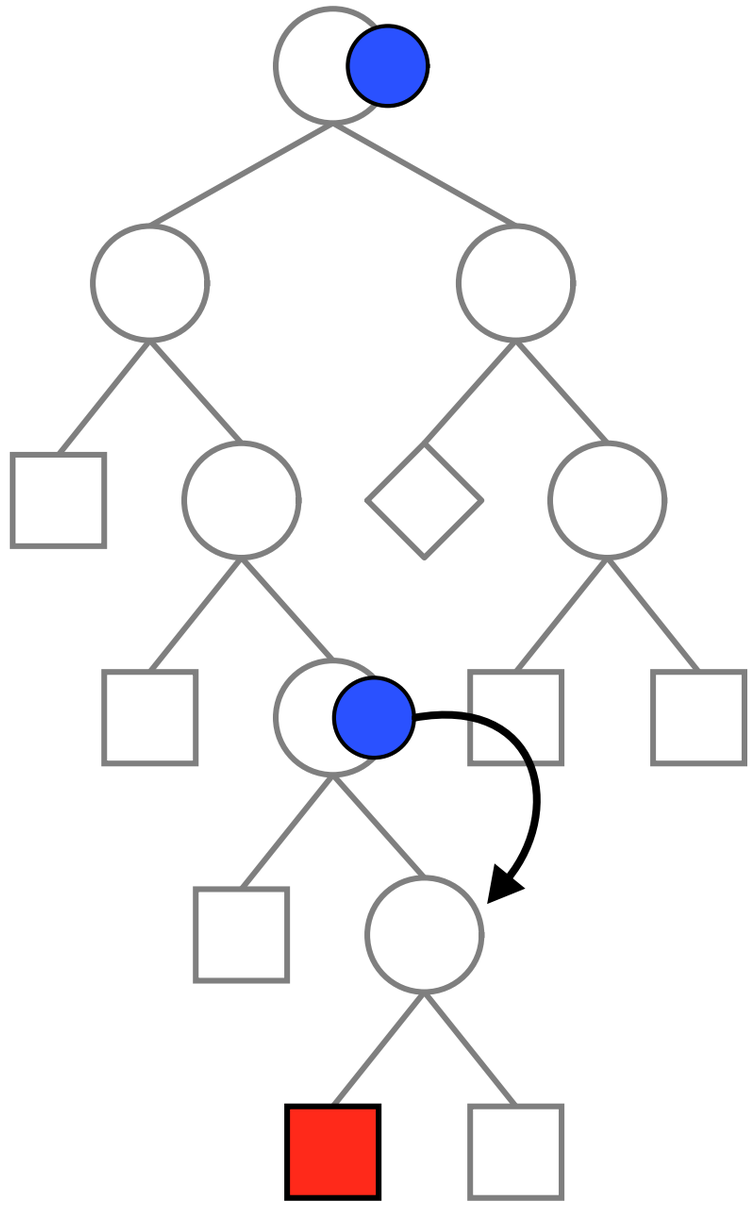
Full recomputation



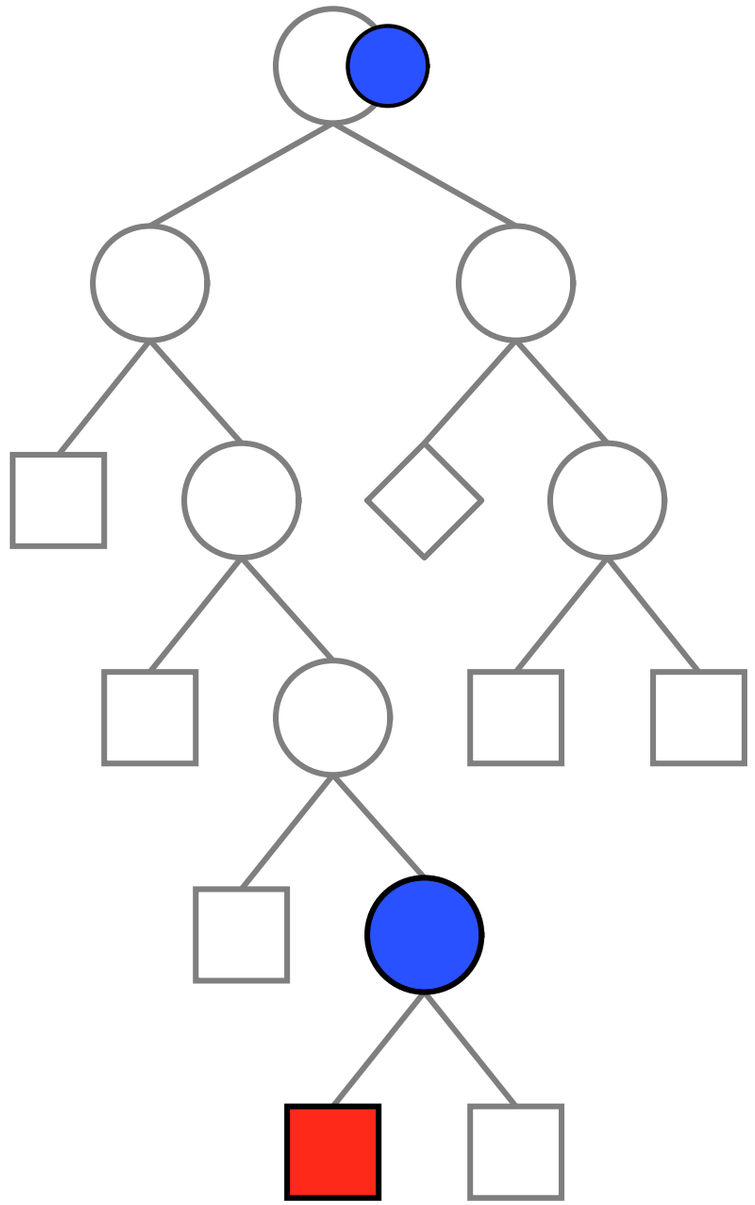
Full recomputation



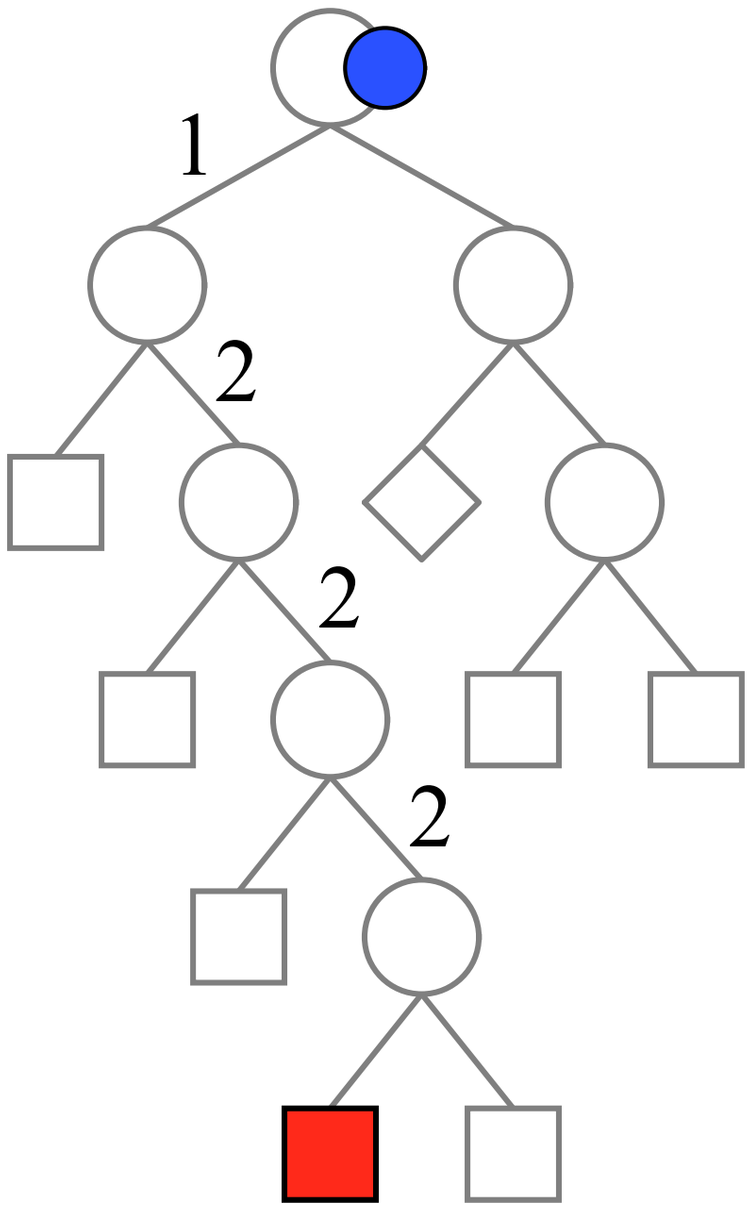
Full recomputation



Full recomputation



How do we recompute?




```
fun recompute s nil = clone s
  | recompute s (i::ir) =
    let
        val s’ = recompute s ir
    in
       commit(s’, i)
       s’
    end
```

DFS with full recomputation

```
fun dfs (s,root,path) =
  case status s of
    FAILED => ()
  | SOLVED => raise Solution s
  | BRANCH =>
     (commit(s, 1);
      dfs (s,root,1::path);
      let val c = recompute(root, path) in
       commit (c, 2);
       dfs (c,root,2::path)
      end)
```

Recomputation

- fundamental idea:

  trade space for time

- full recomputation:

  pro: no backup copies, constant space

  con: time overhead

- can we do better?

Recomputation strategies

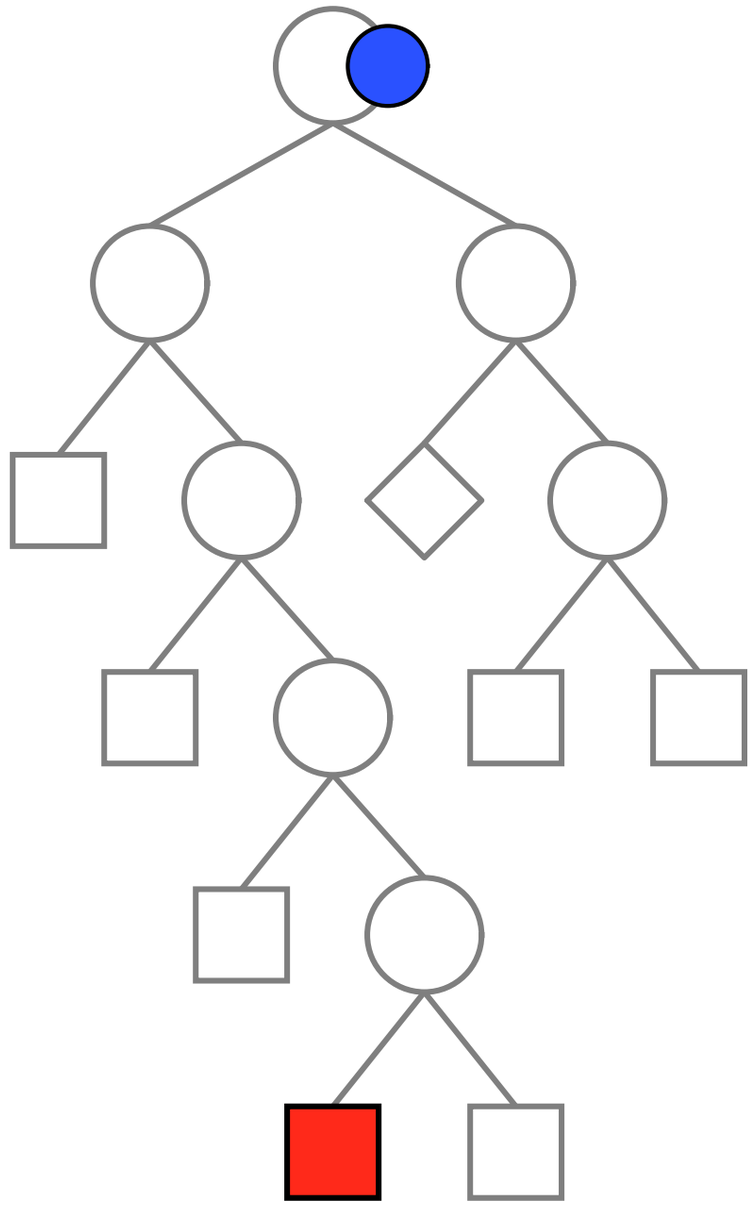
- fixed:

  keep a copy every *n* nodes

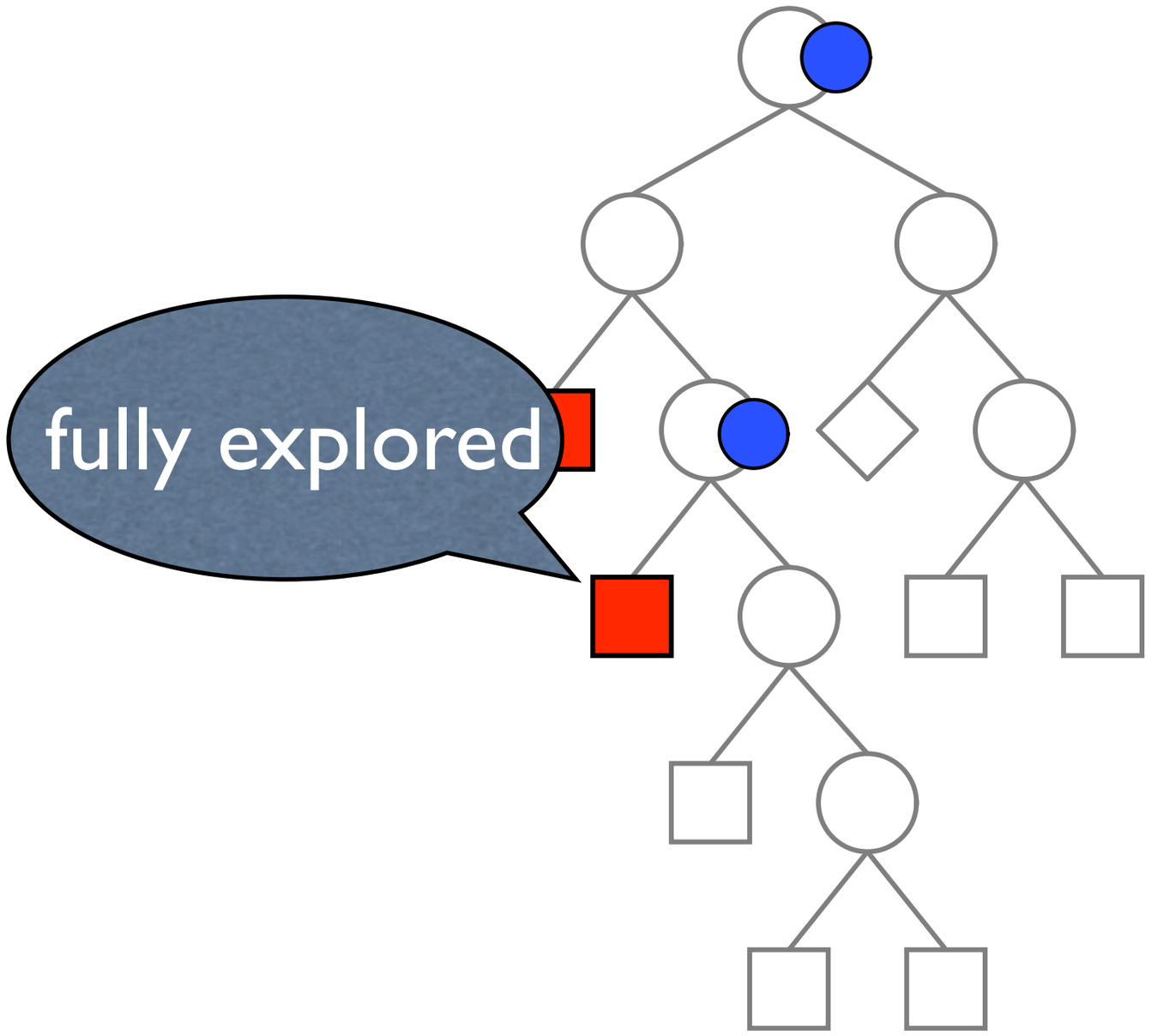
- adaptive:

  during recomputation, place new copy in the middle of the path to the last copy

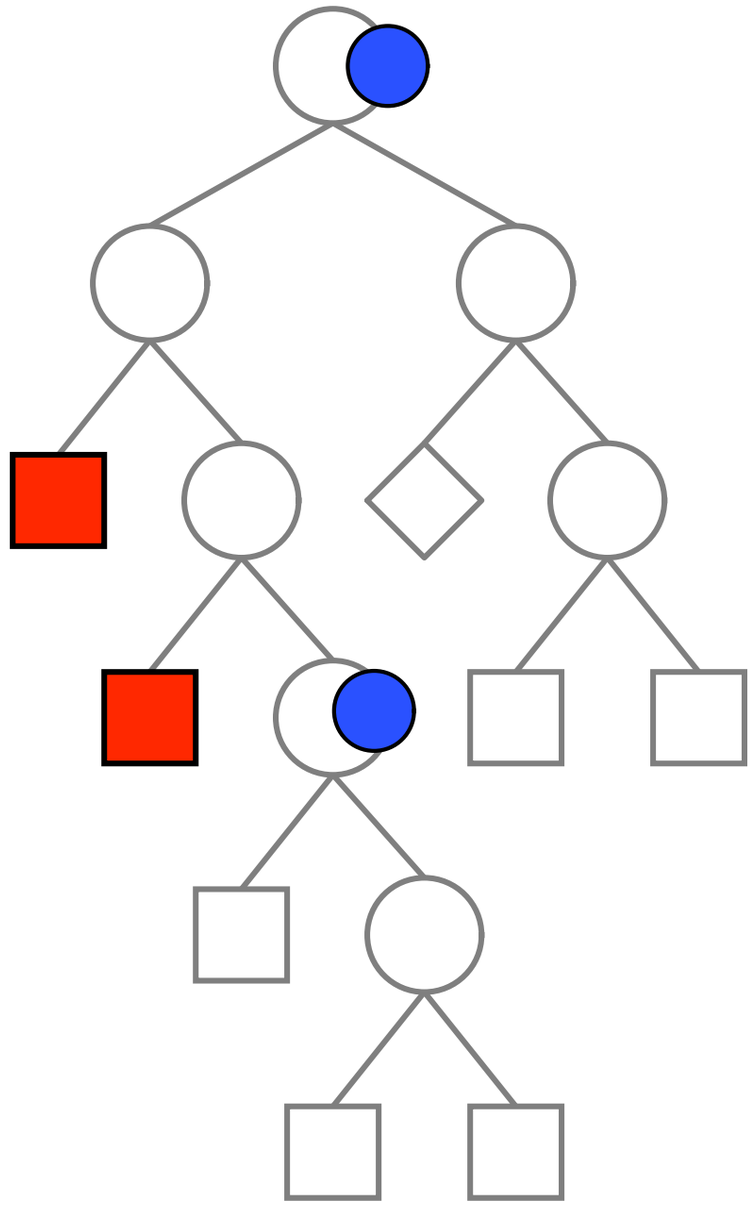
Recomputation strategies



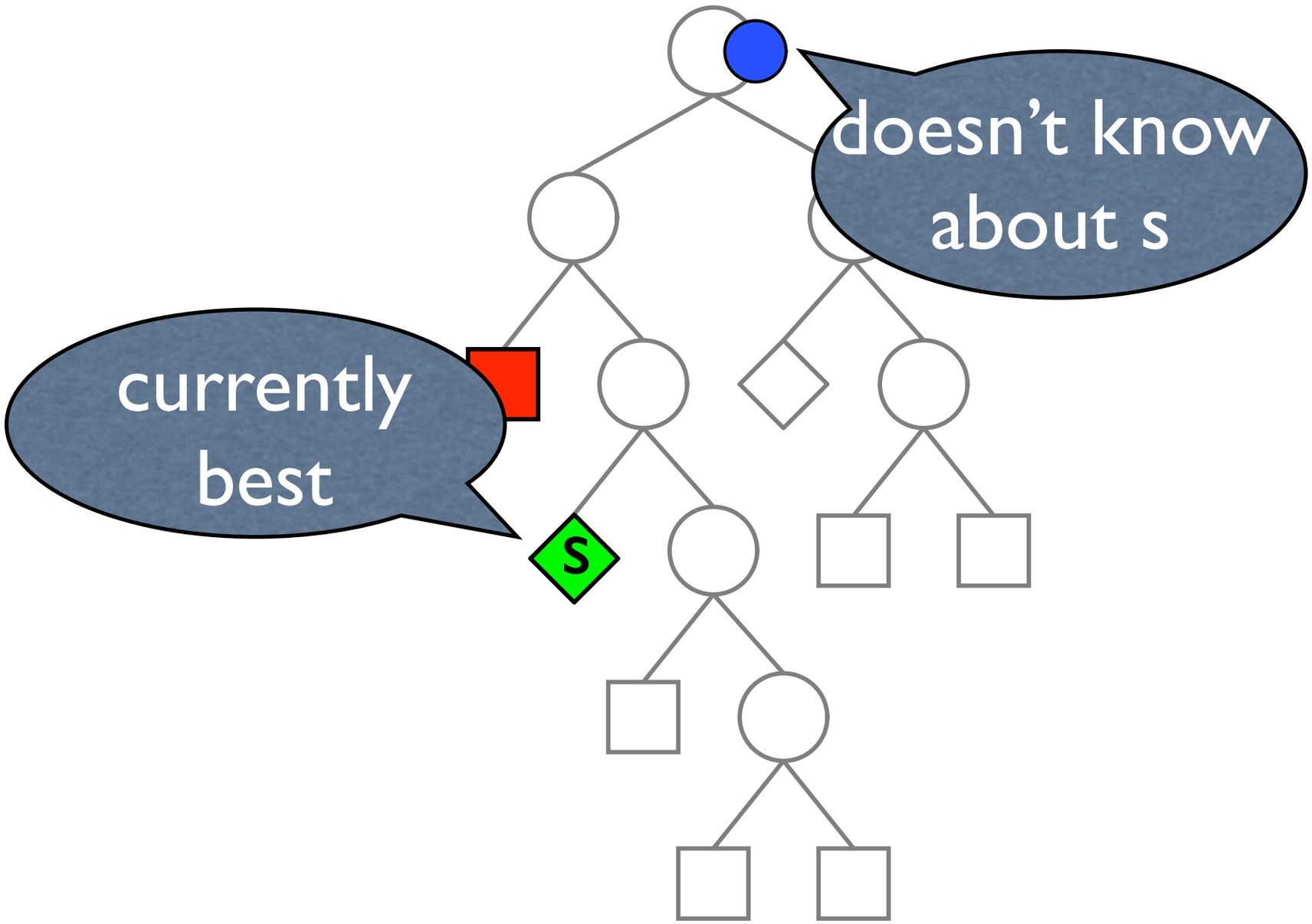
Last Alternative Optimization



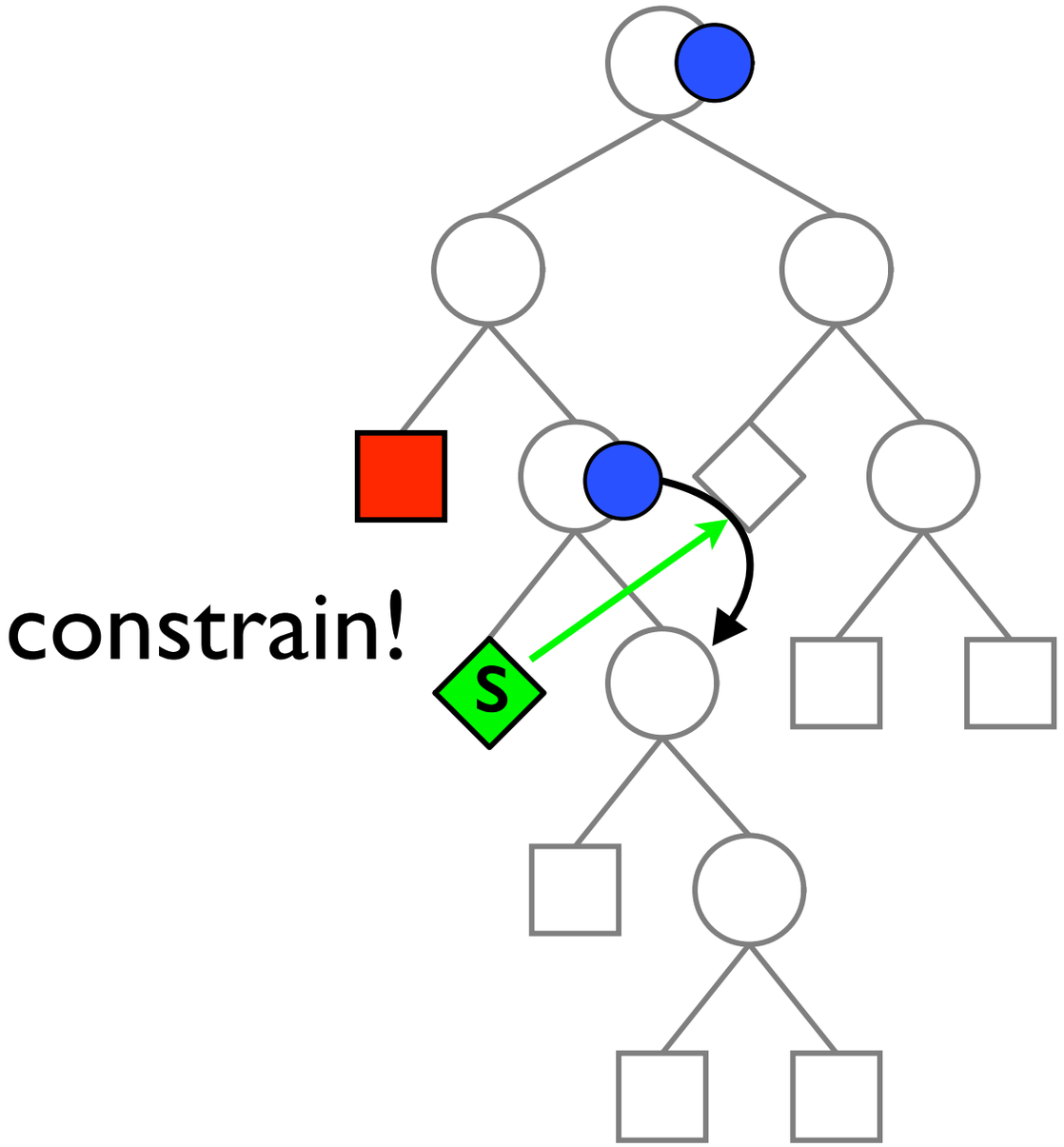
Last Alternative Optimization



BAB with recomputation



BAB with recomputation



Batch Recomputation

- observation:

  recomputing from a space *s* with a path of length *n* computes *n* fixpoints

Fixpoints

- `commit` chooses alternative of a branching
- alternative depends on state of the space
- state of the space determined by fixpoint

  ⇒ commit must propagate before actually committing!

Batch Recomputation

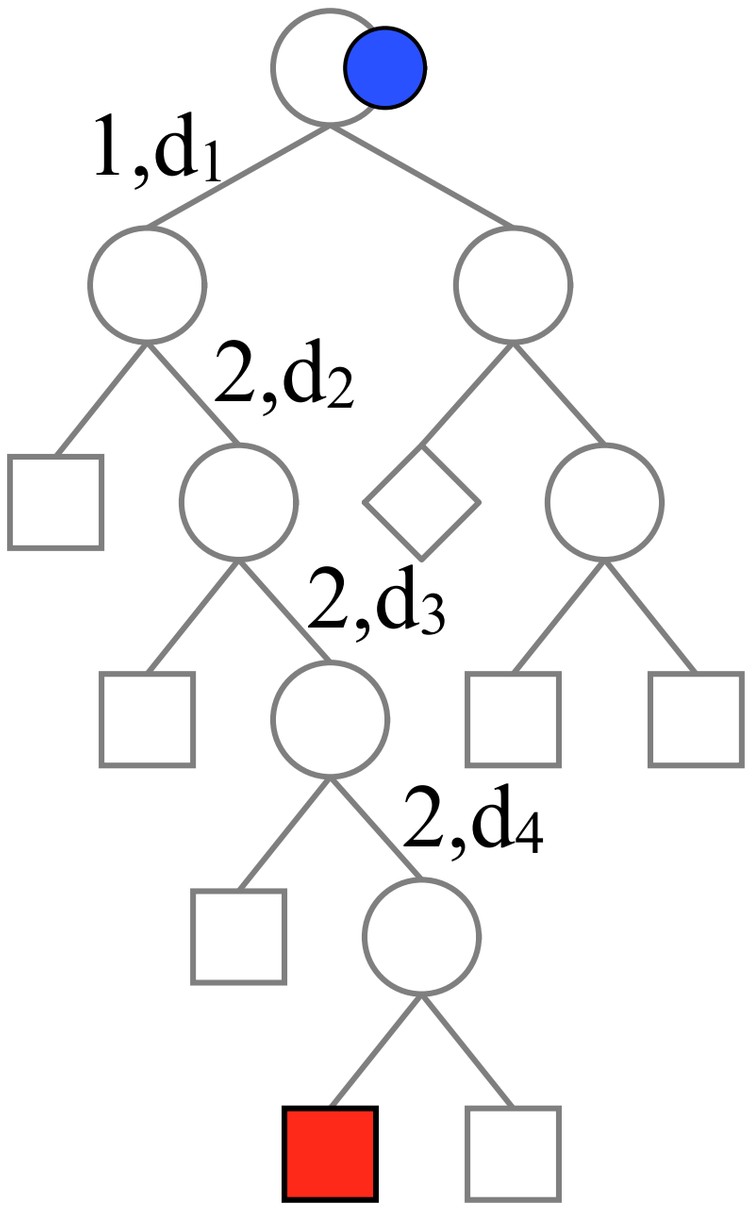
- observation:

  recomputing from a space *s* with a path of length *n* computes *n* fixpoints

- idea:

  more information on paths, compute only one fixpoint!

Batch Recomputation



- Store *descriptions* of the choices:

  represent constraints added by `commit`

- Commit with descriptions, post constraints

- Compute only one fixpoint!

DFS with batch recomputation

```
fun dfs (s,root,path) =
  case status s of
    FAILED => ()
  | SOLVED => raise Solution s
  | BRANCH d =>
      (commit(s,d,1);
       dfs (s,root,(1,d)::path);
       let val c = recompute(root, path) in
        commit (c,d,2);
        dfs (c,root,(2,d)::path)
       end)
```

Summary

- spaces can be big
- copying is not feasible for large search trees
- store only some copies, redo commits on path from such a copy
- batch recomputation: additional information on path ⇒ one FP instead of n

Exercises

- Graded!
- Implement search engines with recomputation

Next week

Set constraints