# Efficiently Evaluating Complex Boolean Expressions

Yahoo! Research
Marcus Fontoura, Suhas Sadanadan, Jayavel Shanmugasundaram, Sergei Vassilvitski, Erik Vee, Srihari Venkatesan and Jason Zien

# Agenda

- Motivation and problem definition
- Algorithms
- Experiments

# Agenda

- Motivation and problem definition
- Algorithms
- Experiments

# Simple example

- Display advertising
  - Ads: Boolean expressions (contracts)

    ```
    age IN {young}
    age IN {old} AND income IN {high, veryHigh}
    income IN {high} AND browser NOT_IN {ie}
    ```

  - Publishers: assignments

    ```
    age = old; income = high; browser = firefox
    ```

# Simple example

- Display advertising
  - Ads: Boolean expressions (contracts)

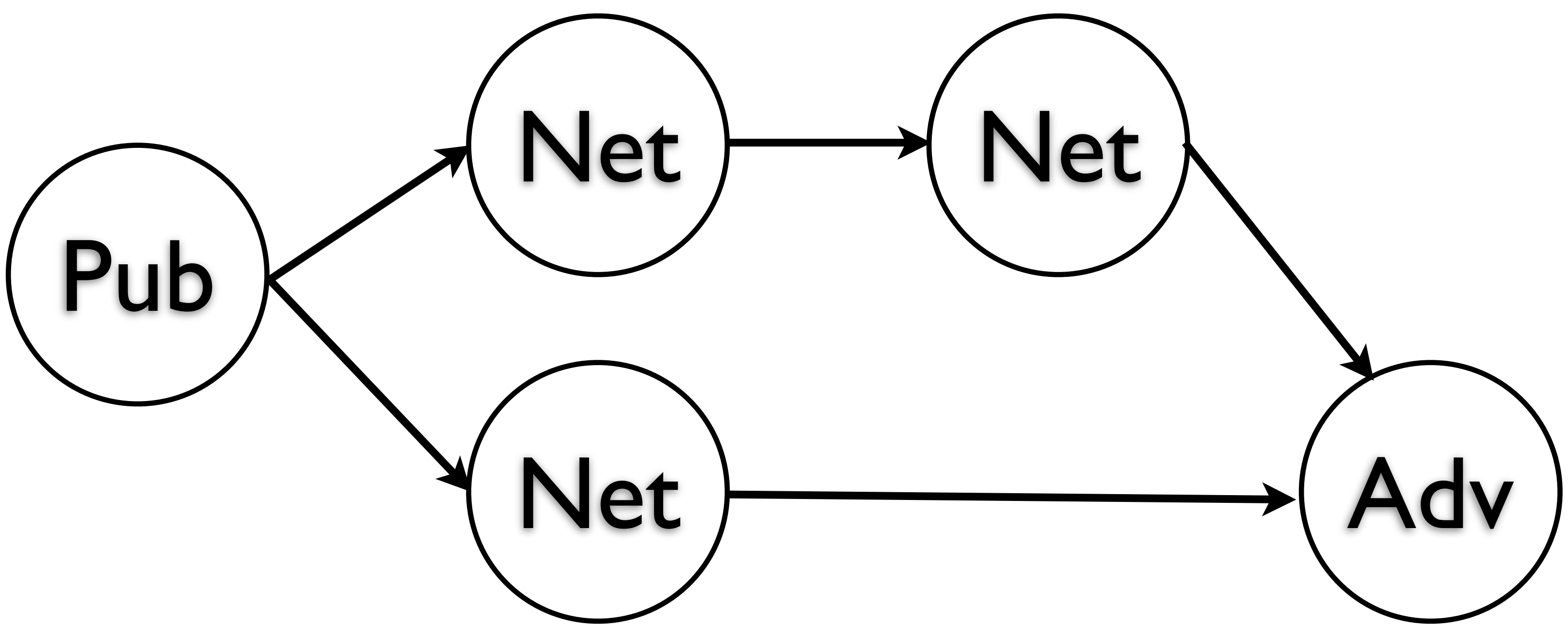
    ~~`age IN {young}`~~
    `age IN {old} AND income IN {high, veryHigh}`
    `income IN {high} AND browser NOT_IN {ie}`

  - Publishers: assignments

    `age = old; income = high; browser = firefox`

# More complex example

- Display advertising exchange

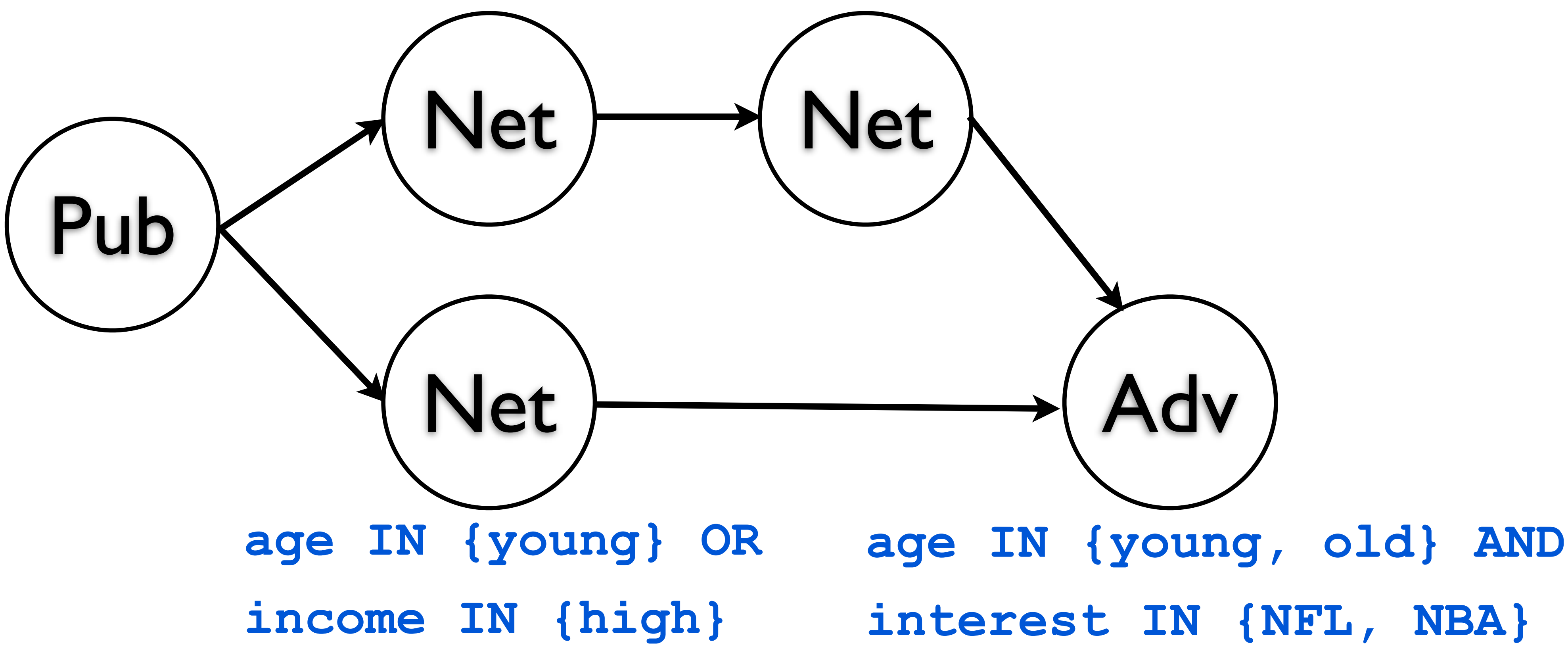



```
age IN {young} OR
income IN {high}
```

```
age IN {young, old} AND
interest IN {NFL, NBA}
```

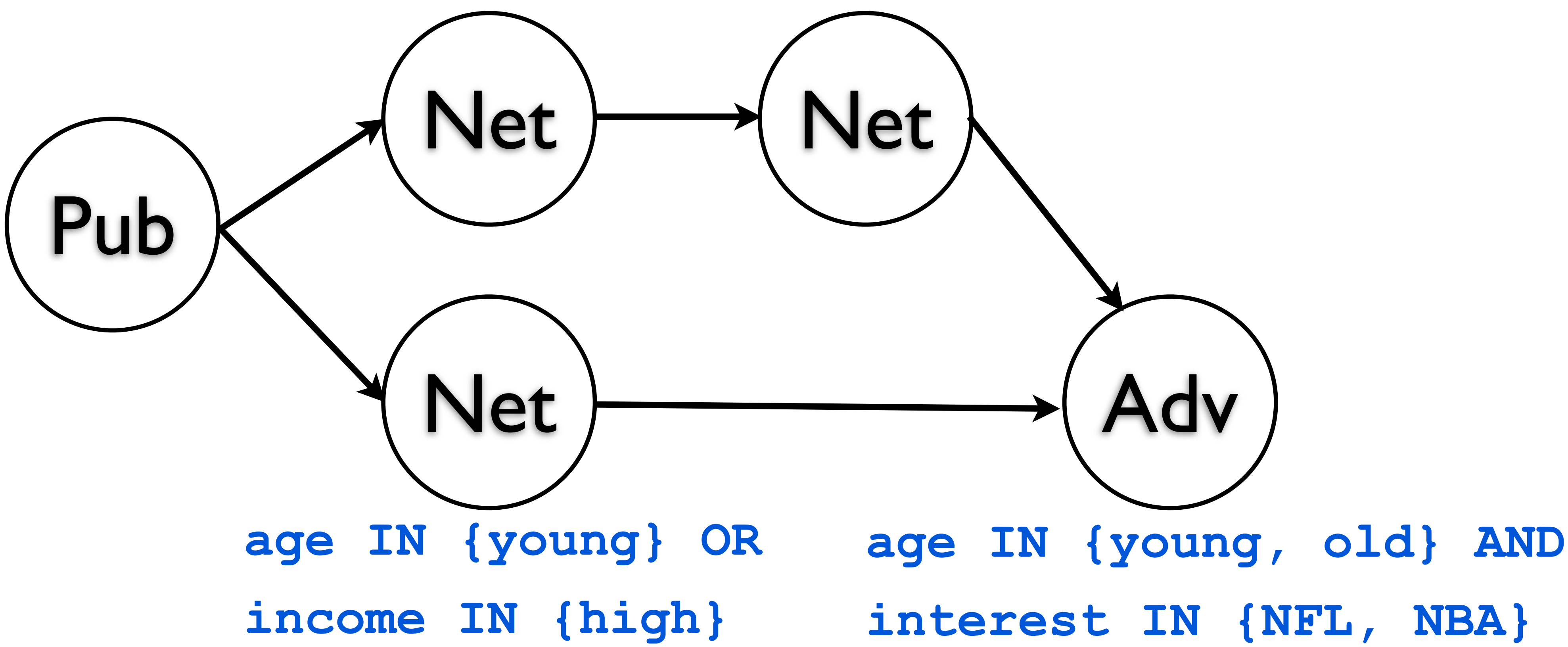
# More complex example

- Boolean expressions model the type of inventory sold by each node



# More complex example

- Each Boolean expression can be a DNF/CNF
- Contracts for the publisher are "complex" expressions



# Other examples

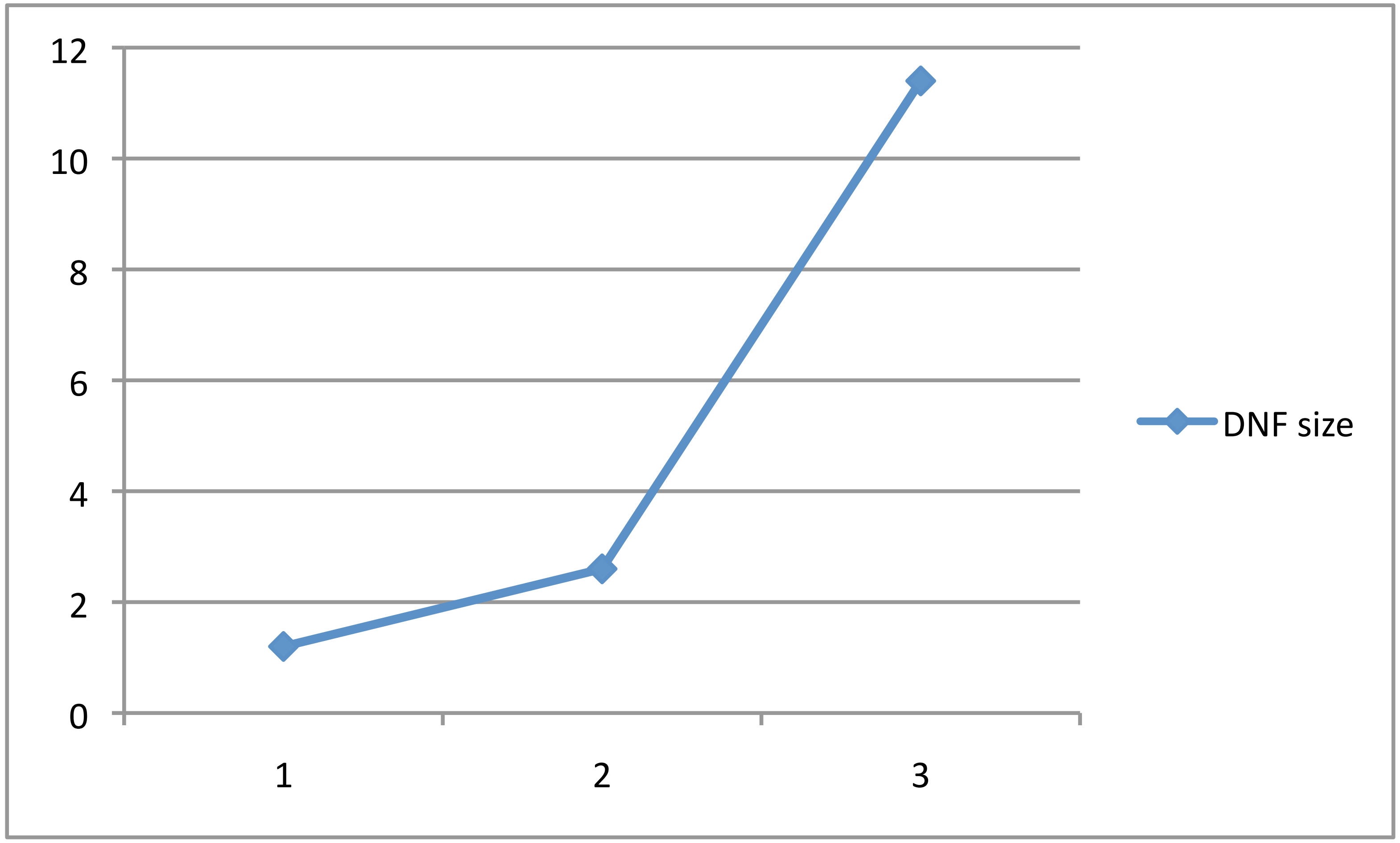
- Automatic targeting in display advertising
  - e.g. machine generated expressions to maximize click-through
- Information dissemination in social network graphs

# State-of-the-art

- There are existing solutions for efficiently evaluating CNF and DNF expressions
  - Content-based publish-subscribe systems
- Normalizing complex Boolean expressions into DNF incurs in an exponential blow-up in size

# DNF growth

- In KB, averaged over 20 DNFs of each size
- Data set is realistic



# Problem definition

- Evaluate complex Boolean expressions (e.g. AND of DNFs)
  - Modeled as a tree of AND/OR nodes, where leafs are conjunctions of IN and NOT_IN operators
- Given an assignment, retrieve all valid expressions

# Intuition

- (Offline) Annotate the conjunctions with their position on the complex Boolean expression tree
- Evaluate conjunctions (leafs) using a state-of-the-art algorithm
- Evaluate the trees bottom-up, using the retrieved conjunctions and their positions

# Agenda

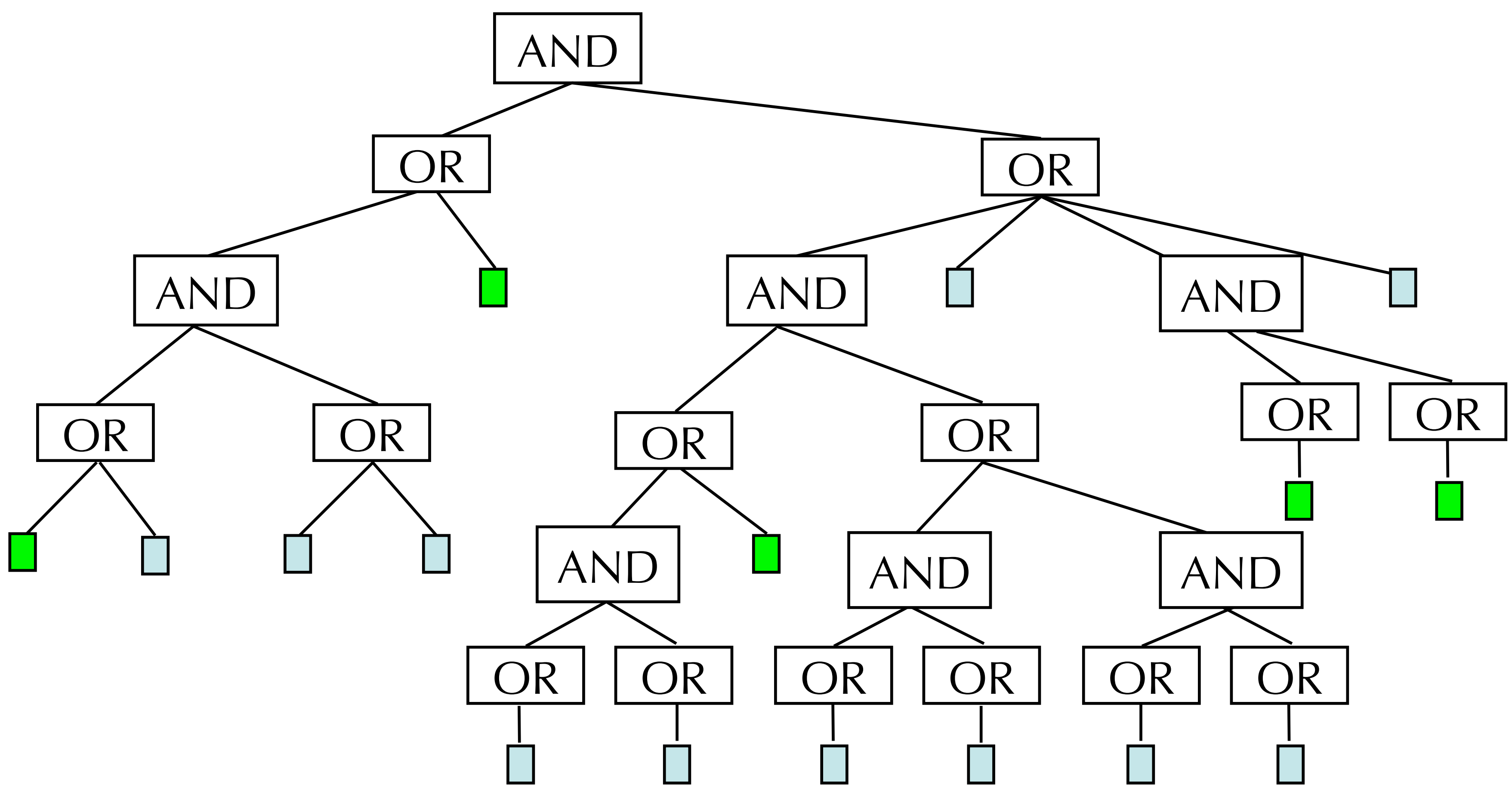
- Motivation and problem definition
- Algorithms
- Experiments

# Agenda

- Motivation and problem definition
- Algorithms
- Experiments

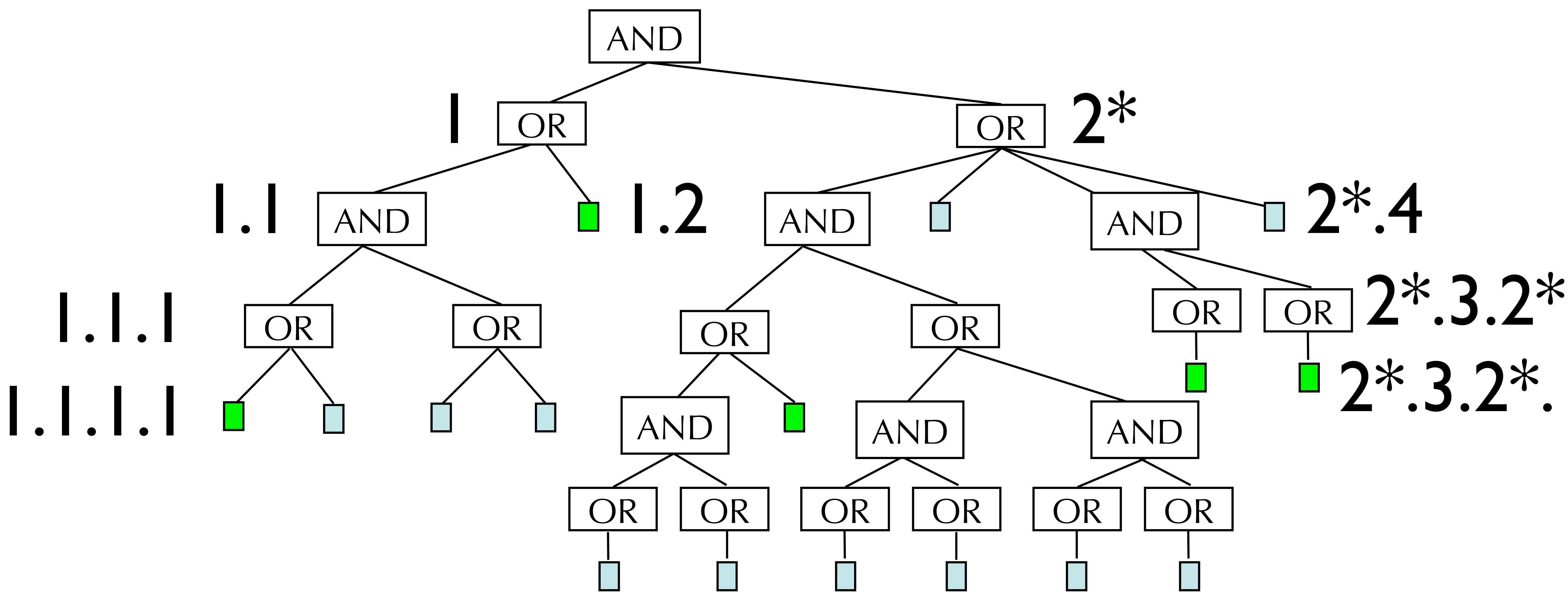
# Online problem

- Given a set of valid conjunctions, is the Boolean expression satisfied
- Tree is never explicitly represented



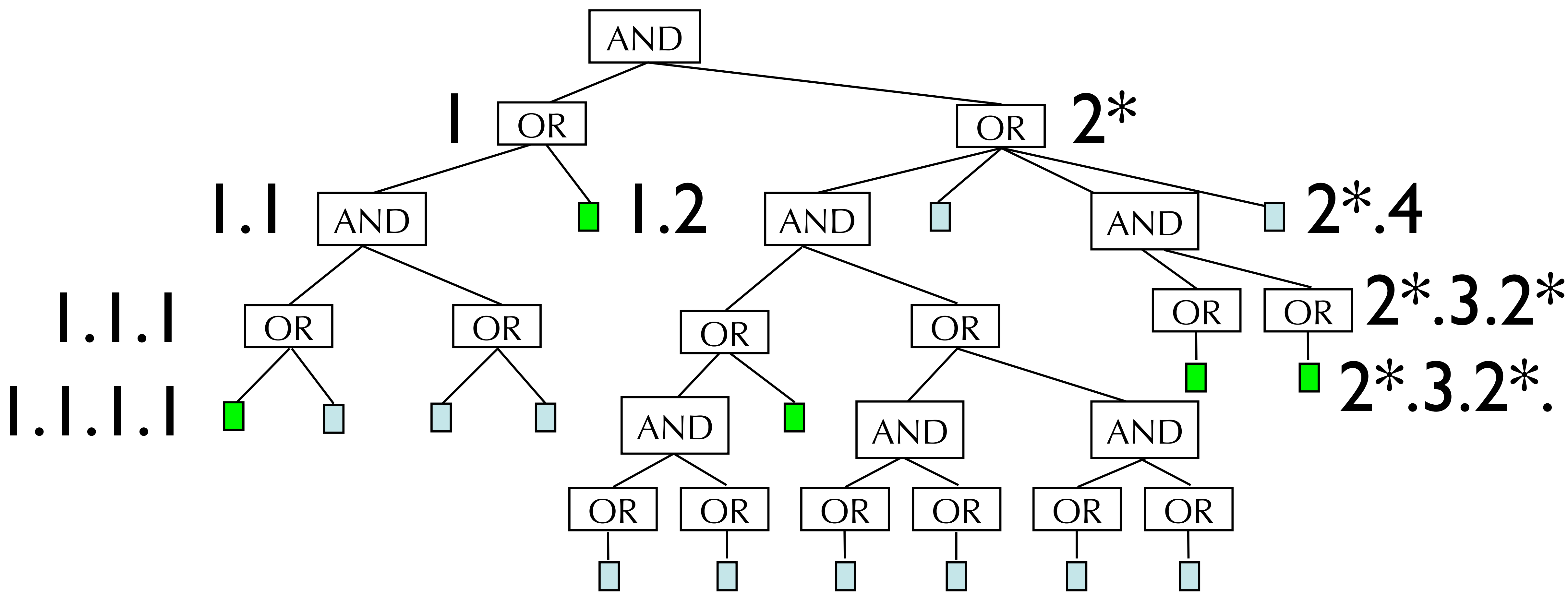
# Algorithm 1: Dewey ids

- Assign Dewey ids for every node in the expression tree
- Ordering children of a node



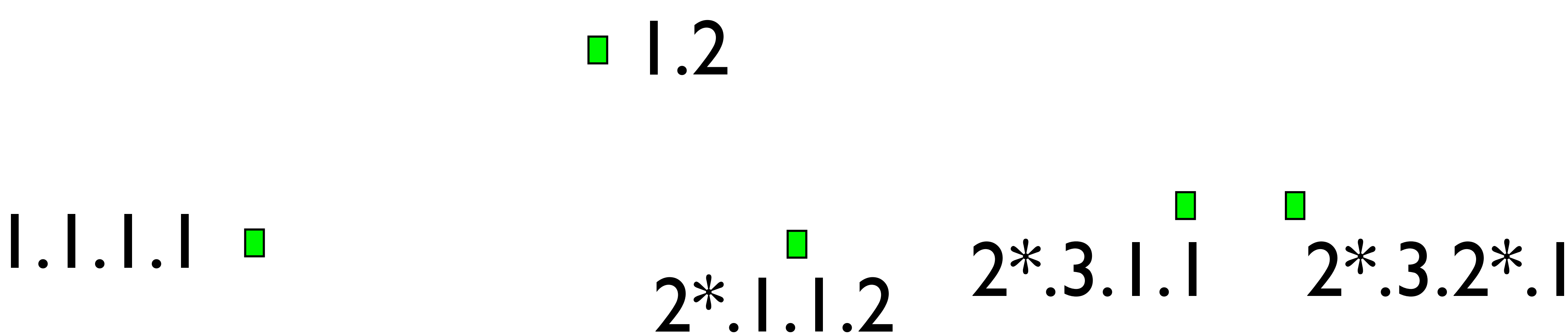
# Algorithm 1: Dewey ids

- Alternating AND/OR trees
- * denotes last child of an AND node



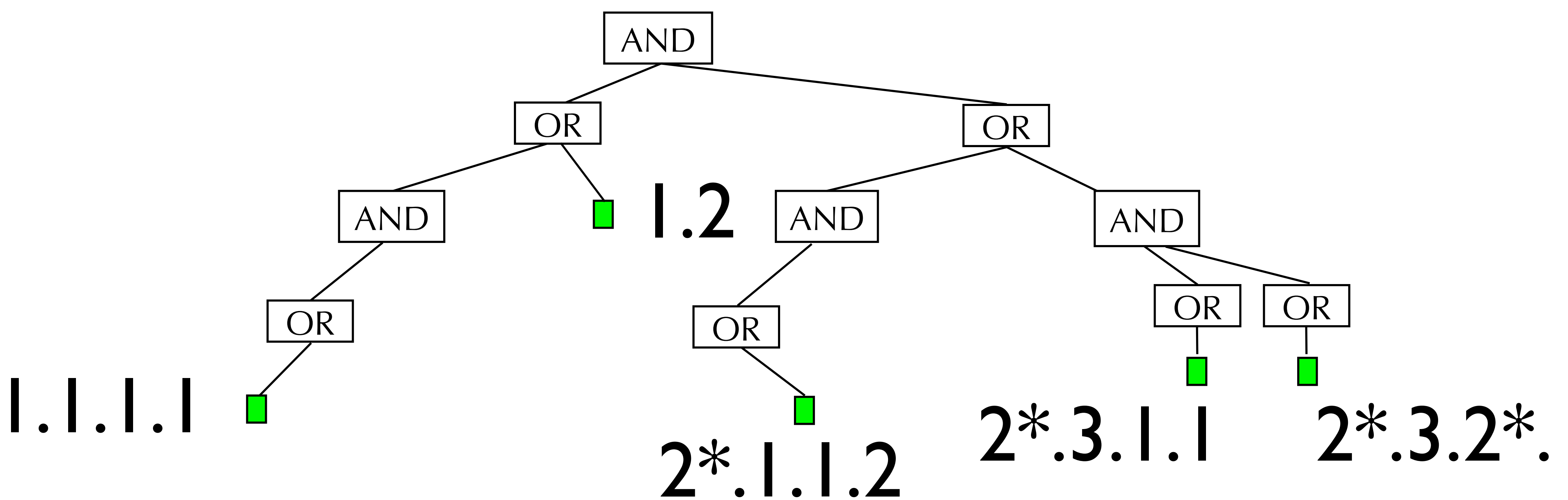
# Algorithm 1: Dewey ids

- Index evaluator will return the leaf nodes, which are the matching conjunctions



# Algorithm 1: Dewey ids

- Index evaluator will return the leaf nodes, which are the matching conjunctions

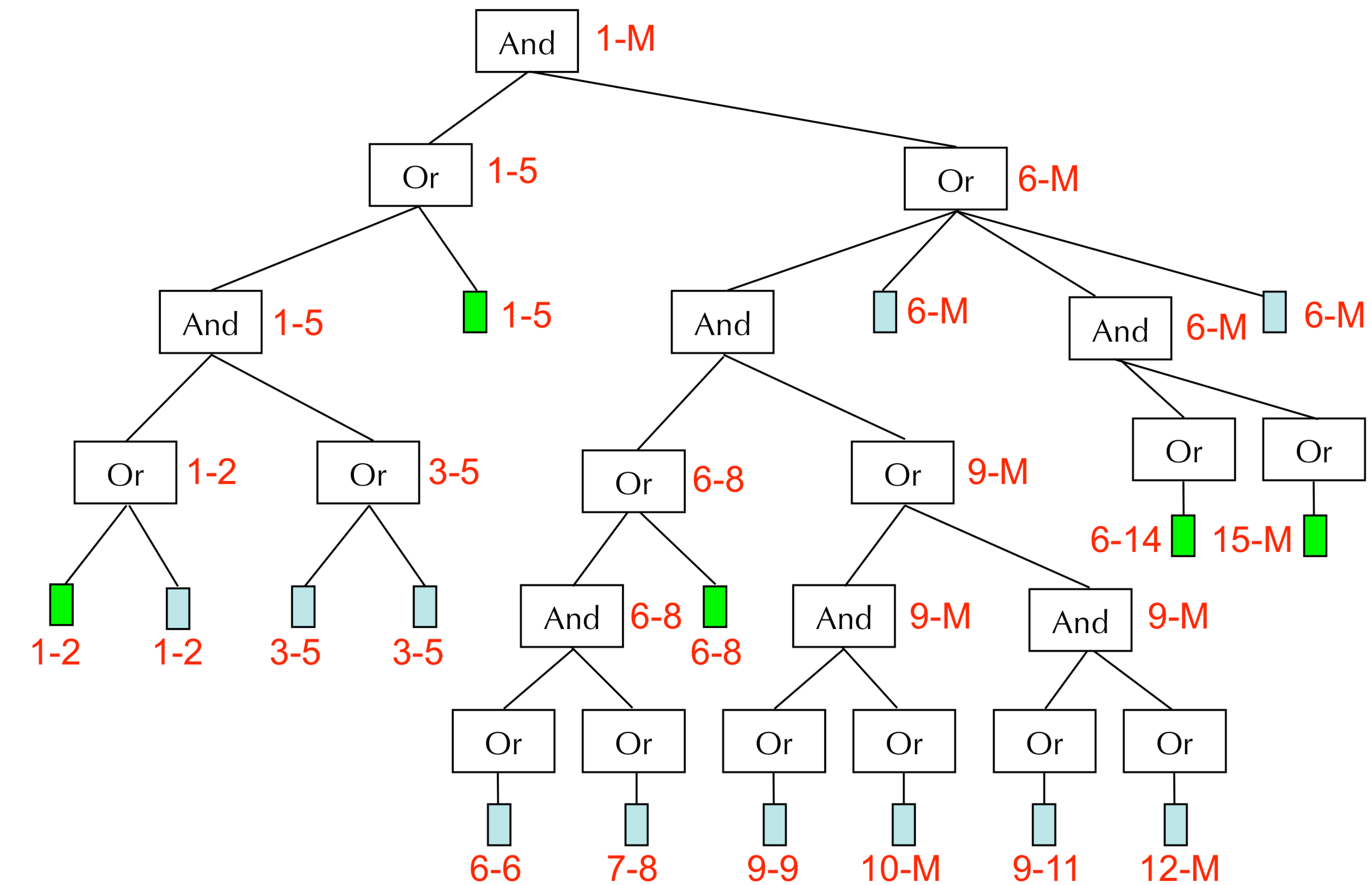


# Algorithm 2: Interval ids

- We map each Boolean tree to a one dimensional interval [1,M]
- M is the maximum number of leaves
- Tree is satisfied if there is a subset of intervals that cover all integer points on [1,M] without overlap

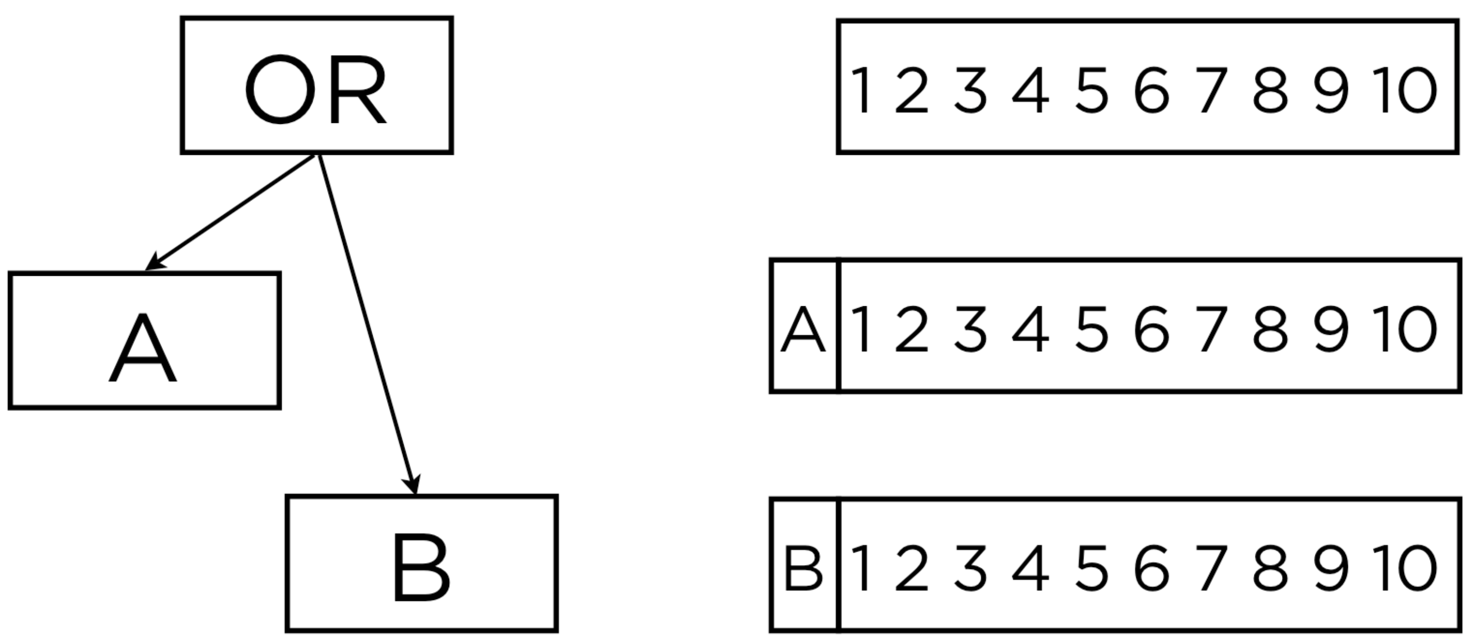
# Algorithm 2: Interval ids

- Look at: [1-5] [6-14] [15-M] : all integer points covered without overlap



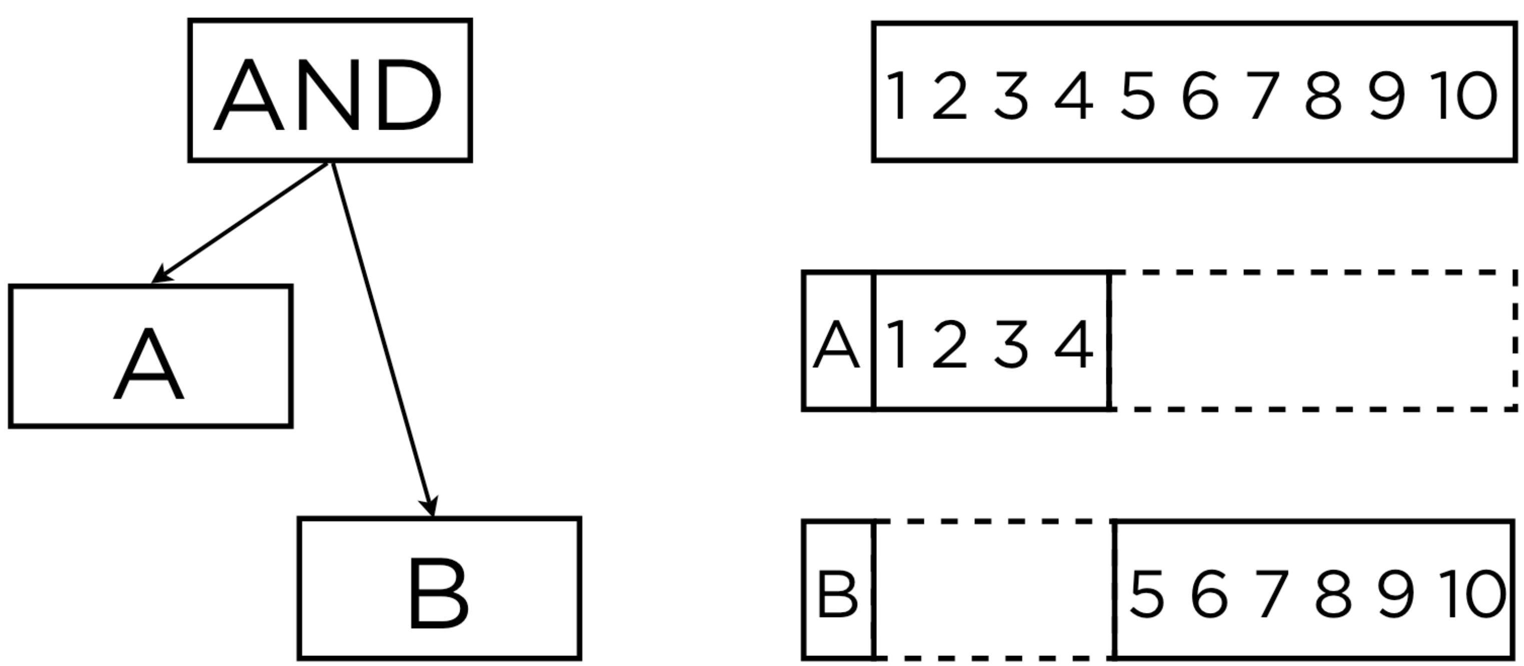
# Assigning intervals

- Recursive procedure
- Children of an OR inherit the parent interval



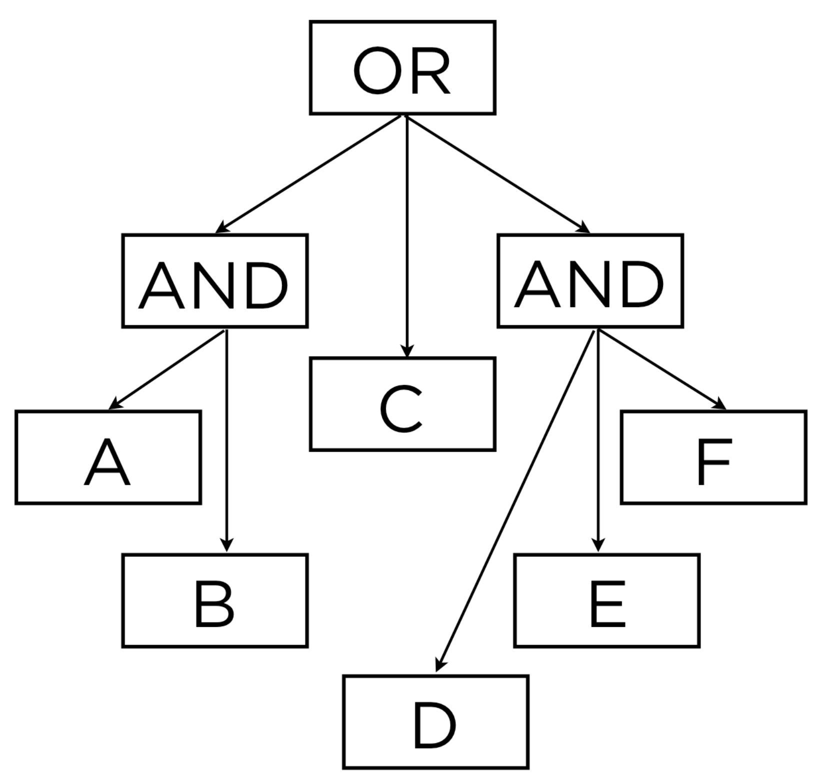
# Assigning intervals

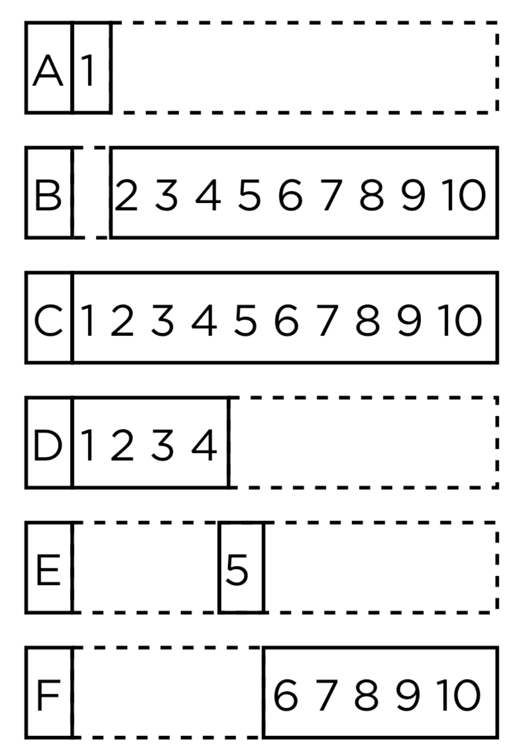
- Recursive procedure
- Children of an AND partition the interval



# Slightly more complex example

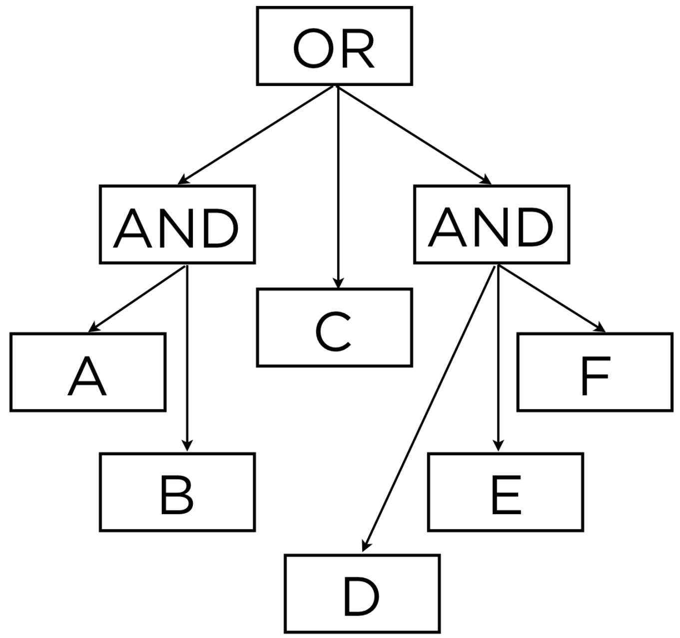
- B & D are not enough to satisfy, since intervals overlap
- D & E & F are OK, since intervals don't overlap

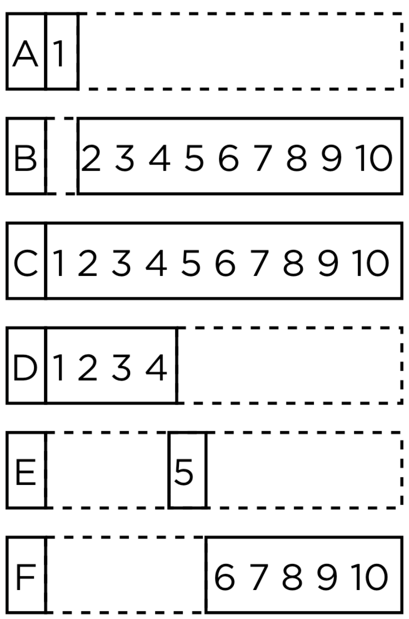


# Example

- Suppose intervals returned are
  - [1,1], [1,4], [5,5], [6,10]
- Final matched array: 1 1 0 0 1 1 0 0 0 1



# Agenda

- Motivation and problem definition
- Algorithms
- Experiments

# Agenda

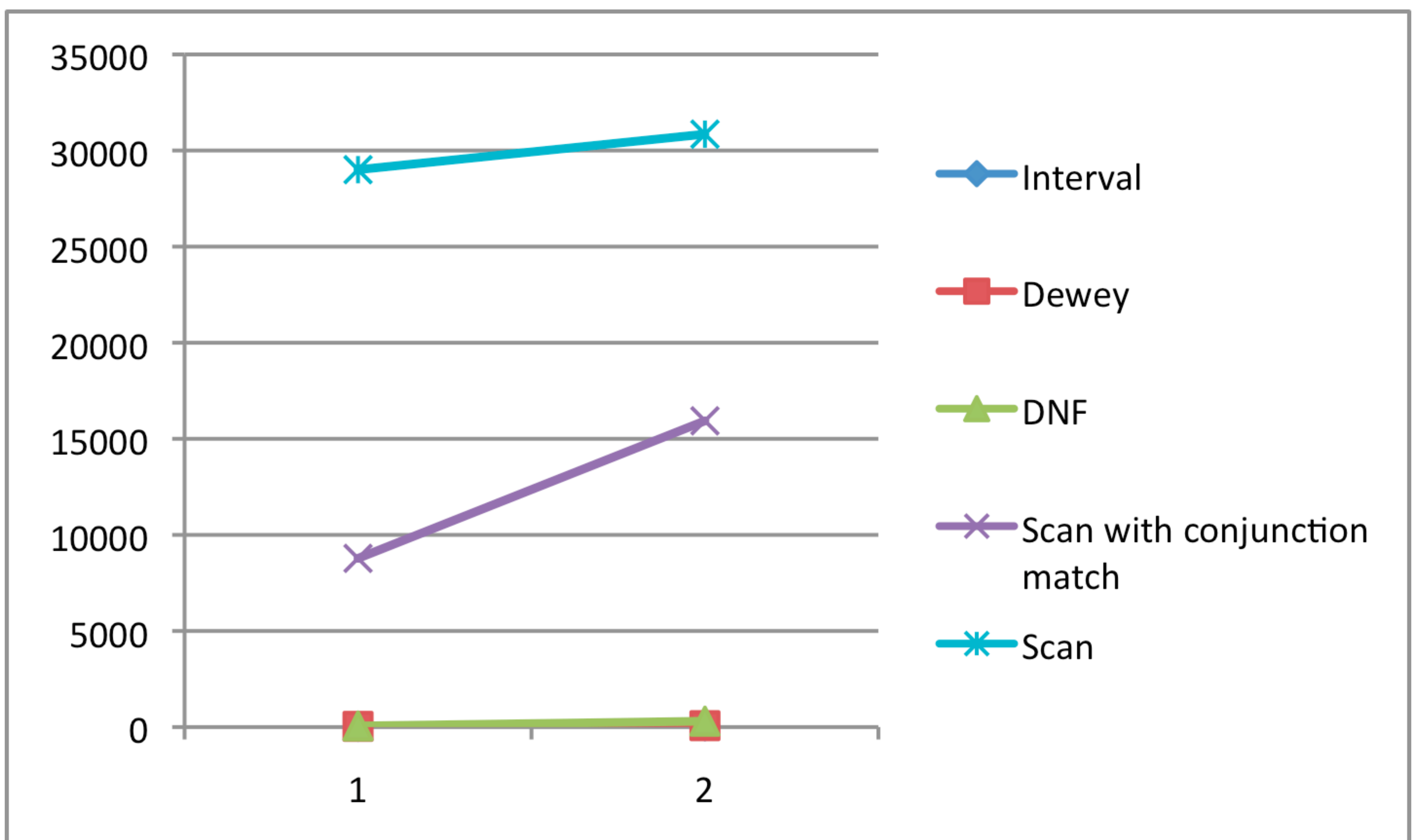
- Motivation and problem definition
- Algorithms
- Experiments

# Data

- Generated a synthetic data set of expressions based on real logs
- Depth of the tree between 1 and 4
- Typical number of children of nodes between 1 and 4

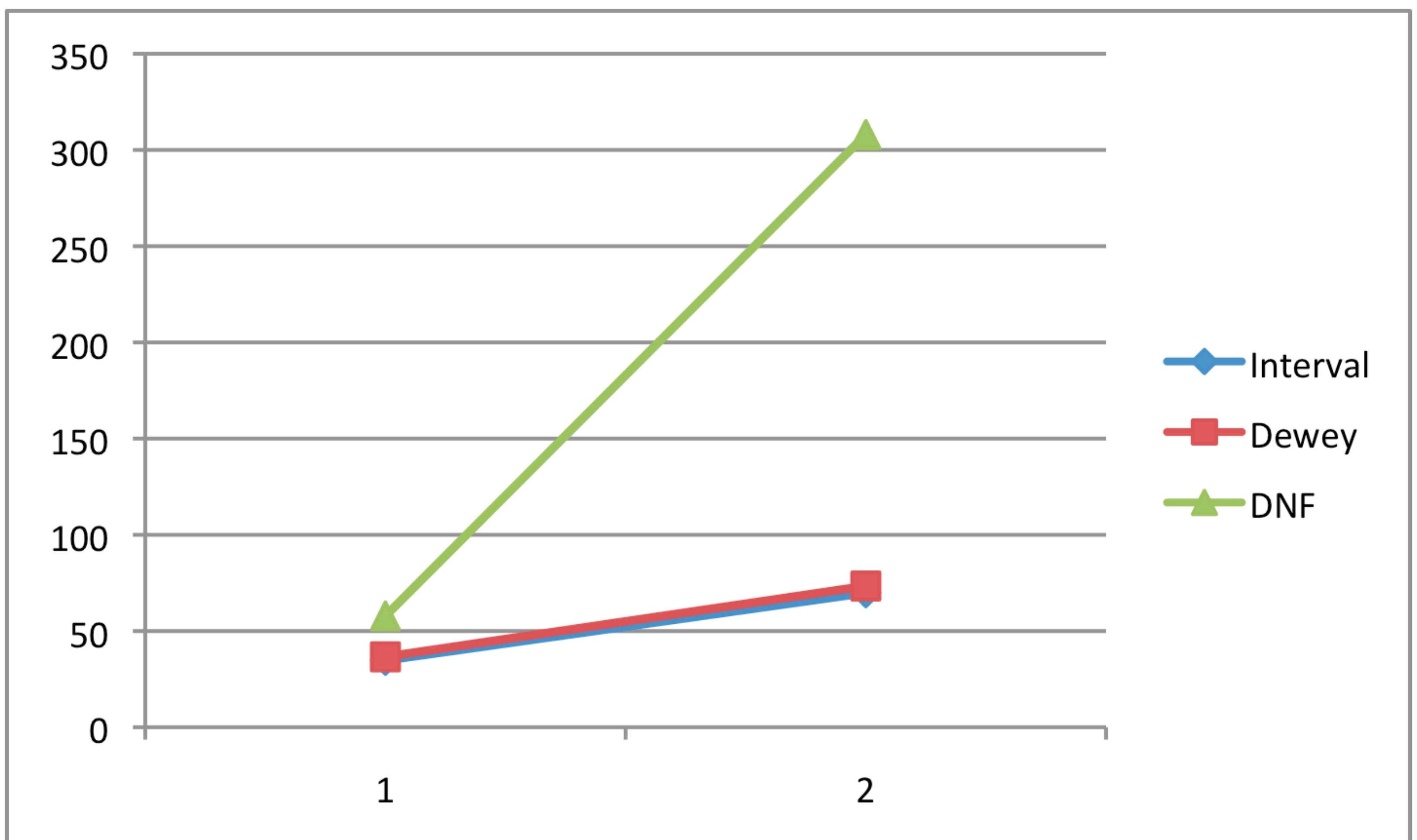
# Performance of different methods

- Running time in ms (y axis) vs. tree depth (x axis). Scan does not scale wrt time



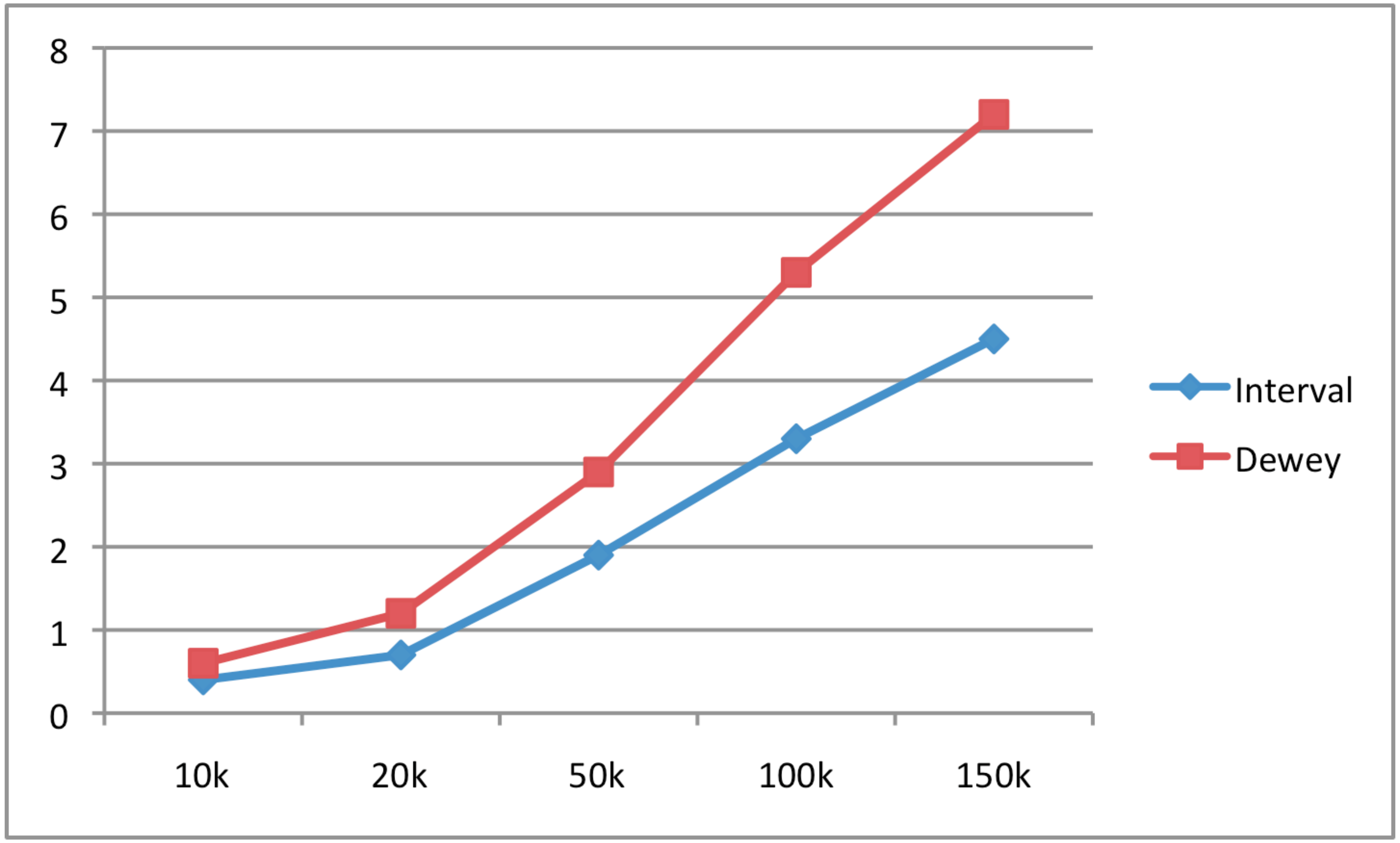
# DNF performance

- Running time  in ms (y axis) vs. tree depth (x axis)



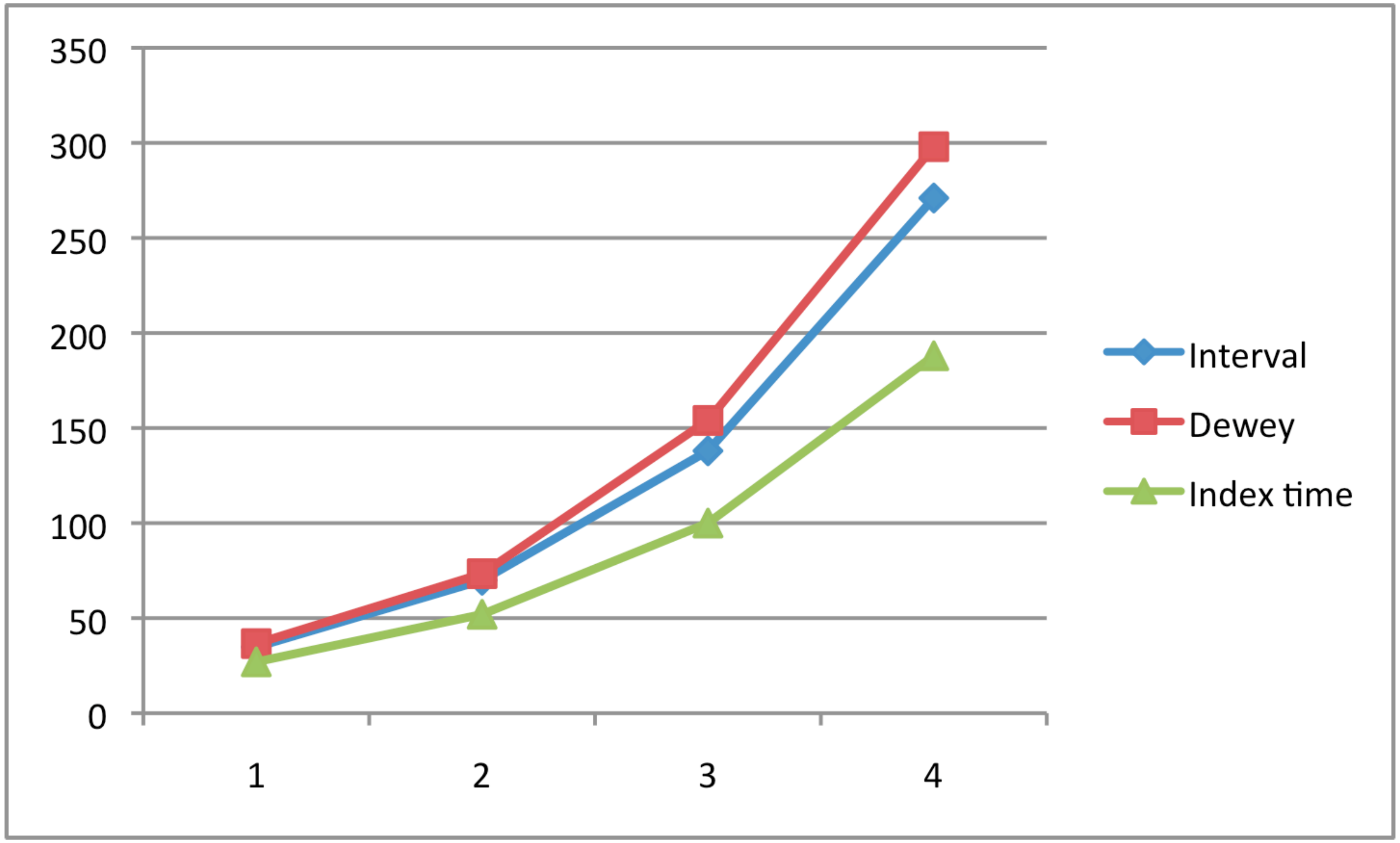
# Interval and Dewey

- Running time of the tree evaluation in ms (y axis) vs. #boolean expressions in test



# Conjunction matching time

- Running time of the tree evaluation in ms (y axis) vs. tree depth



# Excluding conjunction matching