Graph mining for technology-enhanced learning

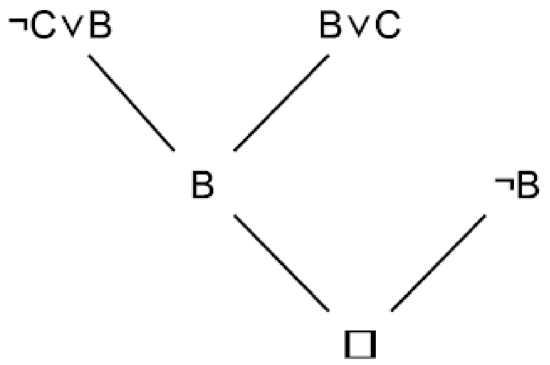
Master's Thesis
FI MU Brno 2013

Author: Karel Vaculík
Thesis supervisor: doc. RNDr. Lubomír Popelínský, Ph.D.

Motivation

- Constructive tasks (resolution proofs in logic, tableau proofs, ...)
  - Large amount of tasks solved by students (automated processing is an advantage)
  - Task solutions can be represented as graphs, some solutions (e.g. resolution proofs) even as trees.

⇒ Usage of graph mining methods

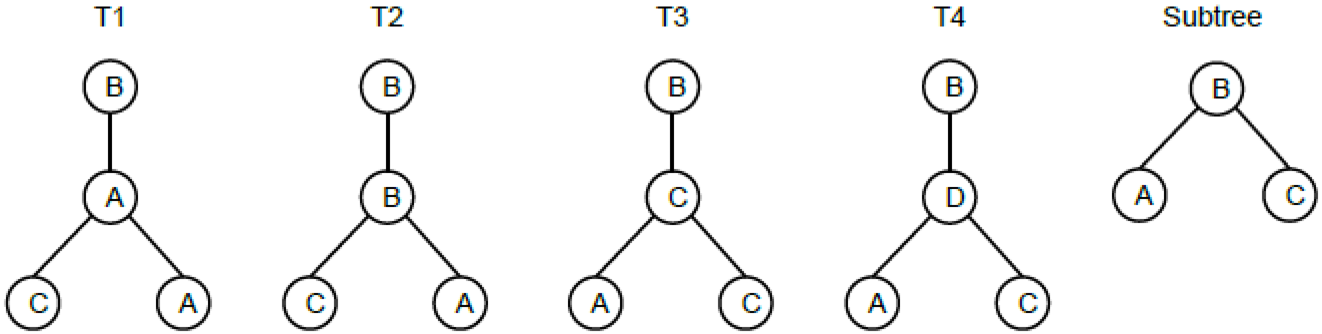


Thesis goals

- Overview of graph mining methods with focus on trees
- Design and implementation of a tree mining system for classification of solved tasks in logic, specifically resolution proofs in propositional calculus
- System verification on data set from logic courses at FI MU
- Discussion and further improvements

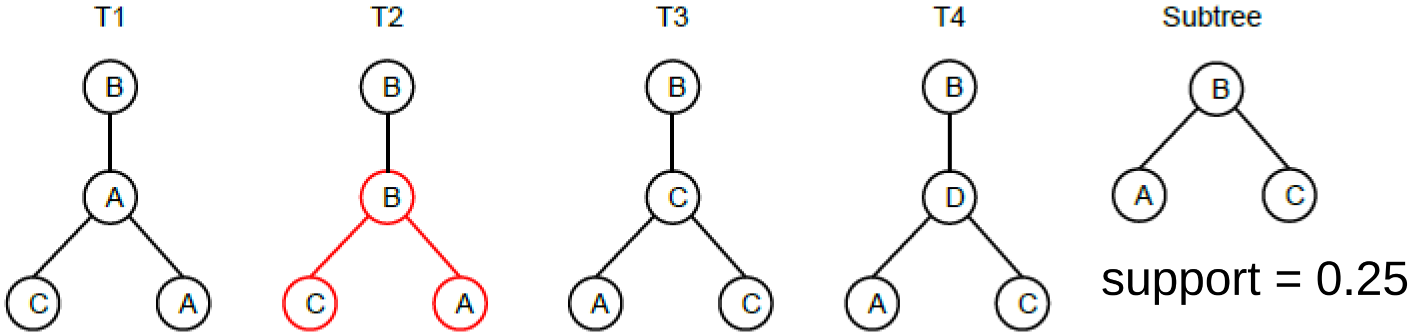
Tree mining

- Main focus on frequent tree mining
- Trees: free, rooted (ordered, unordered);
- Subtrees (rooted trees): induced, embedded



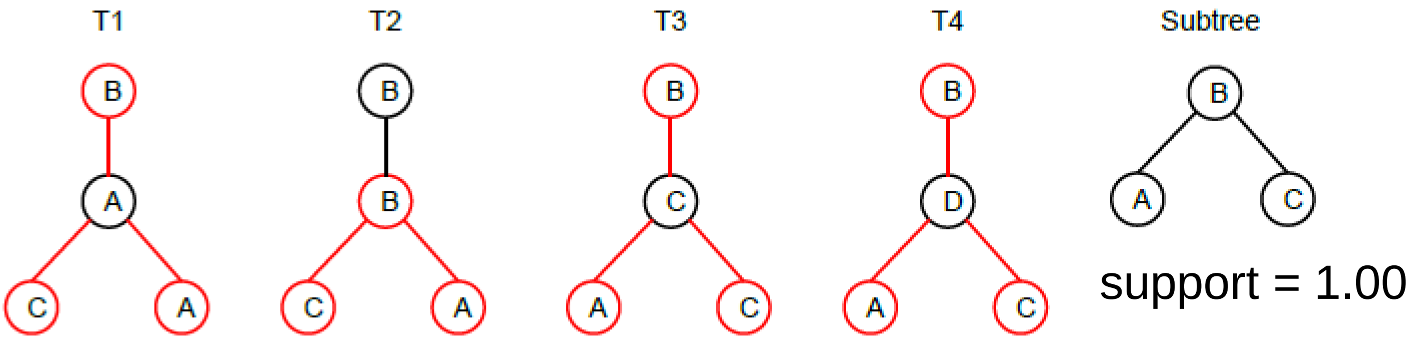
Tree mining

- Main focus on frequent tree mining
- Trees: free, rooted (ordered, unordered);
- Subtrees (rooted trees): induced, embedded



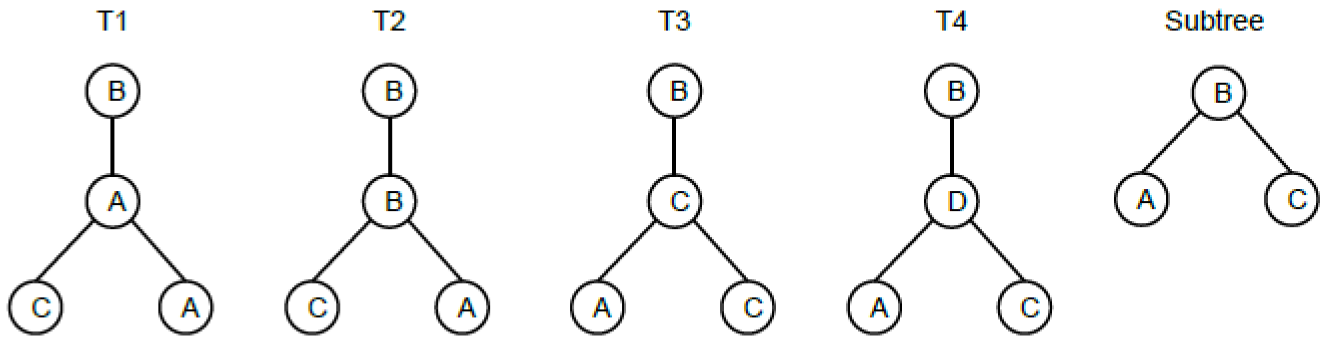
Tree mining

- Main focus on frequent tree mining
- Trees: free, rooted (ordered, unordered);
- Subtrees (rooted trees): induced, embedded



Tree mining

- Main focus on frequent tree mining
- Trees: free, rooted (ordered, unordered);
- Subtrees (rooted trees): induced, embedded



- Task: find all frequent subtrees satisfying specified minimum support

Tree mining algorithms

- FreeTreeMiner
- TreeMiner
- Freqt
- uFreqt
- Unot
- PathJoin
- HybridTreeMiner
- Sleuth

Tree mining algorithms

- FreeTreeMiner
- TreeMiner
- Freqt
- uFreqt
- Unot
- PathJoin
- HybridTreeMiner
- Sleuth

Only for free trees

Only for ordered trees

Implementation not available

Unsuitable output

Data

- 393 solved resolution proofs; in GraphML format
- Source: tests from course IB101 – Introduction to Logic
- 2 assignments (183 + 210 trees)
- Trees (proofs) classified as:
  - Positive – correct solution (322 instances)
  - Negative – incorrect solution (71 instances)
- Other attributes: number of obtained points, type of resolution, numbers of occurences for particular types of error, ...

Created system

New system which consists of modules for:

- Data preprocessing (from general graphs in GraphML to trees in convenient format)
- Frequent subtree mining (using SLEUTH)
- Visualization of trees with subtres and decision trees
- **Classification of resolution proofs**

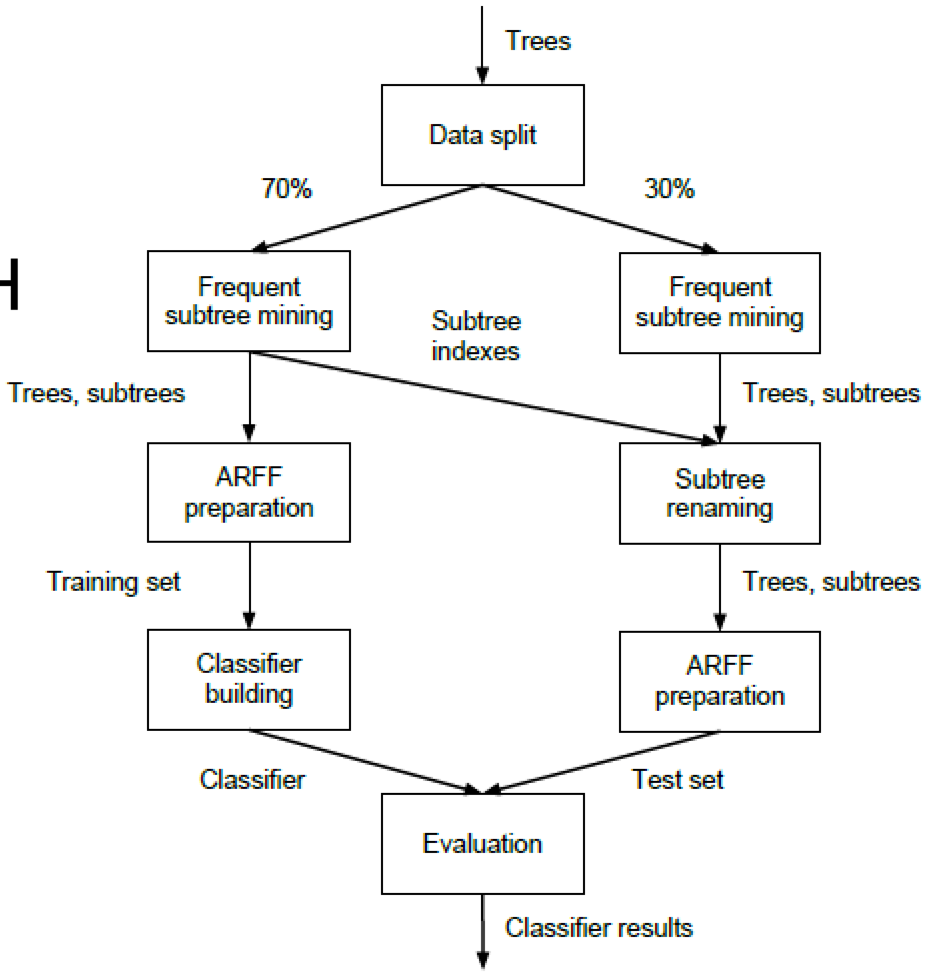
Classification

- Classes: correct or incorrect proof (values *positive* and *negative*)
- Every tree (proof) is represented by a set of its frequent subtrees according to a given minimum support value:

pattern_1	pattern_2	...	pattern_m	class
true	false	...	false	negative
...	...	...	...	
false	true	...	true	positive

Classification – basic scheme

- Evaluation method:
  - Using test set
  - Cross validation
- Subtrees by SLEUTH
- Classifiers from Weka

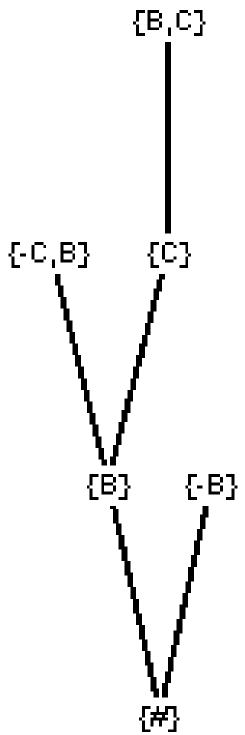


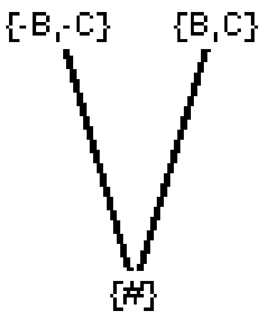
Classification – emerging patterns

- Emerging pattern: A pattern with a substantial support in data that belongs to one particular class (GrowthRate metrics)
- For each class: create a lexicographical ordering among *all* patterns on *GrowthRate* × *Support* × *PatternSize*
- Take patterns from beginning of those orderings to get N desired features for classification
  - More patterns can be taken from ordering for a particular class

Classification – emerging patterns

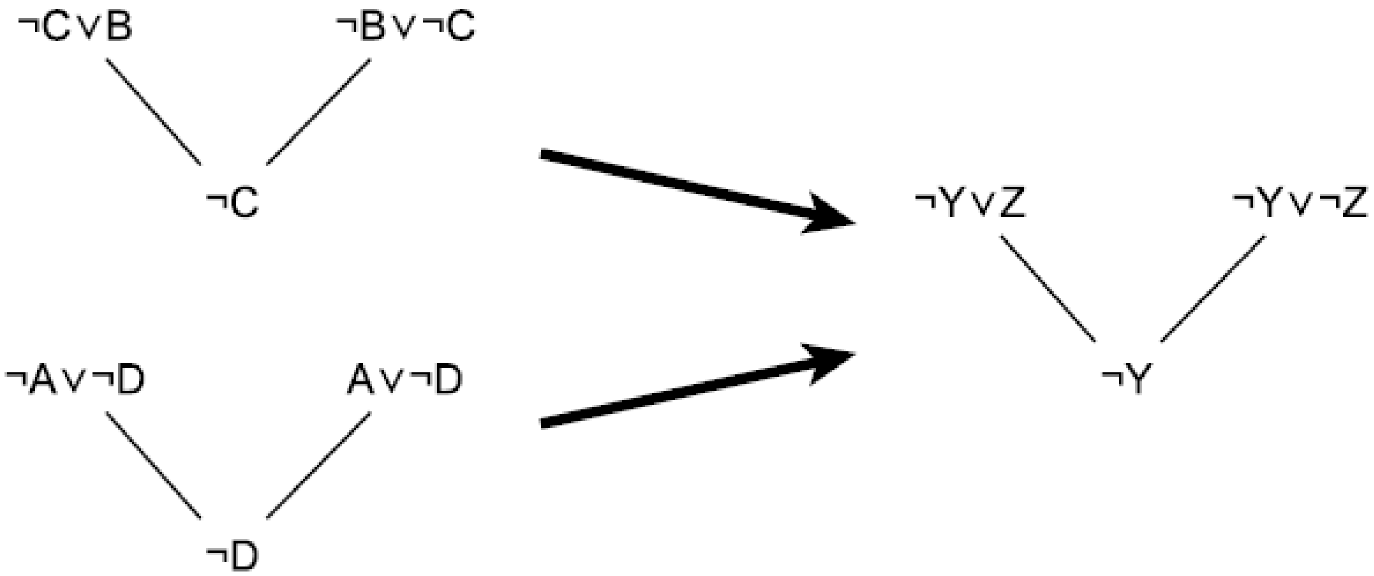
- Examples of most significant emerging patterns for classes (visualized by the system):
  a) positive b) negative





Classification – generalized patterns

- Goal: perform generalization on the set of patterns



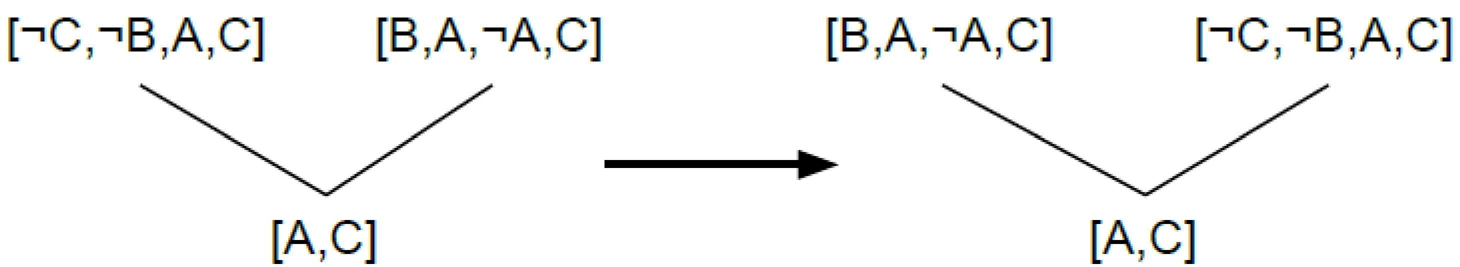
Classification – generalized patterns

- Only for the 3-node patterns (application of the resolution rule)
- Lexicographical ordering on list of literals based on number of negative and positive literals: *NegLiteral* × *PosLiteral*
  - E.g. $\neg C, \neg B, A, C \Rightarrow \mathrm{A} \leq \mathrm{B} \leq C \;\; ((0,1) \leq (1,0) \leq (1,1))$;
    $B, A, \neg A, C \Rightarrow B \leq \mathrm{C} \leq \mathrm{A} \;\; ((0,1) \leq (0,1) \leq (1,1))$
- Lexicographical ordering on the previous ordering – for node (clause) comparison:
  - $((0,1), (1,0), (1,1)) \leq ((0,1),(0,1), (1,1))$

Classification – generalized patterns

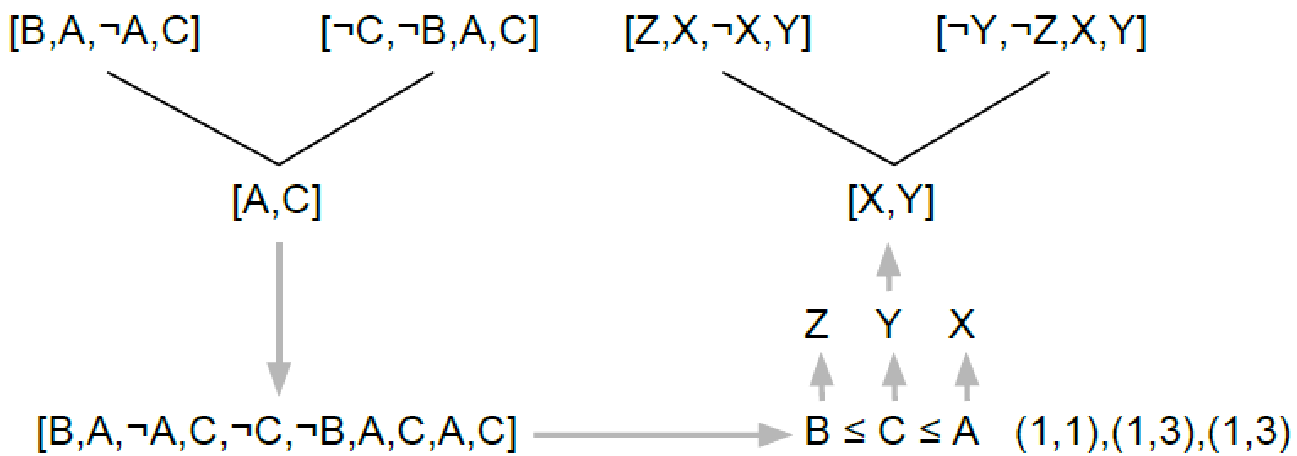
- Procedure:
  1. Compare parent nodes, smaller node will be first.

     E.g.:



Classification – generalized patterns

- Procedure:
  1. Compare parent nodes, smaller node will be first.
  2. Merge literals from all nodes and create ordering among them (in case of a tie check ordering on nodes). Then assing variables to literal letters according to ordering. E.g.:



Classification – generalized patterns

- Procedure:
  1. Compare parent nodes, smaller node will be first.
  2. Merge literals from all nodes and create ordering among them (in case of a tie check ordering on nodes). Then assing variables to literal letters according to ordering.
  3. Lexicographically reorder literals in each node (as we want: $Z, \neg Y \sim \neg Y, Z$).

Classification – three classes

- To increase reliability of a classifier, it is used a third class *UNKNOWN* for cases in which the classifier is not very confident
- J48, NaiveBayes and IBk can output probability of classifying an example $\Rightarrow$ when probability is lower than a given *threshold*, use *UNKNOWN*

Experiments

- Classification on generalized frequent patterns and emerging generalized patterns; used cross-validation
- Generalized frequent patterns:
  - Min. support (%): 0, 1, 2, 5, 10, 15, 20
- Emerging generalized patterns:
  - Min. support (%): 1
  - Number of used emerging patterns: 10, 50, 100, 200, 500
  - Proportion of patterns for classes negative / positive: 50:50, 65:35, 80:20

Experiments

- Generalized frequent patterns:

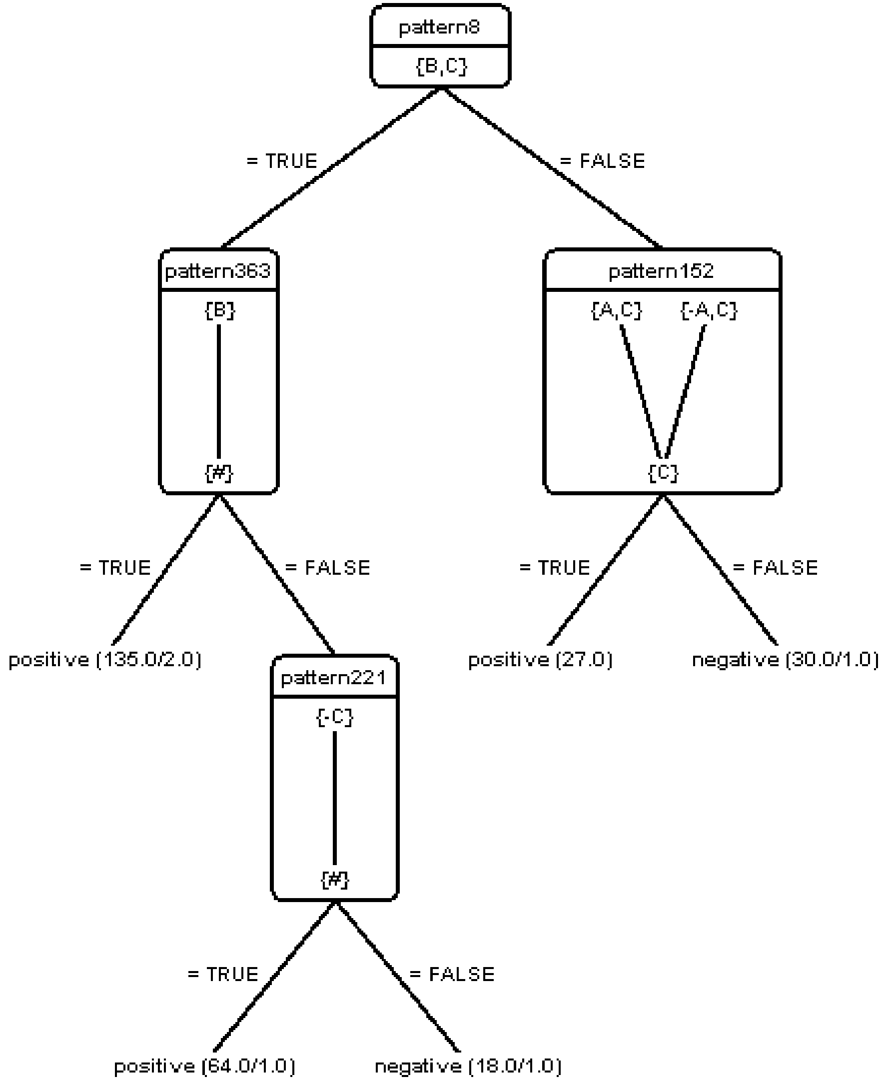
Algorithm	Min. support (%)	Accuracy (%)	Precision (positive)	Recall (positive)	Precision (negative)	Recall (negative)
J48	0	97.2	0.970	0.997	0.986	0.862
Naive Bayes	1	96.7	0.965	0.997	0.986	0.832
SMO	0	97.5	0.973	0.997	0.988	0.873
IBk	5	96.7	0.970	0.991	0.955	0.862

- Emerging generalized patters, best result:
  J48, 100 patterns (proportion 65:35), accuracy 97.5%

Experiments

- Classification into 3 classes:
  - Same values for parameters + threshold 0.5–0.9
  - Best result: IBk on generalized frequent patterns (min. support 5%), threshold 0.8, accuracy 97.97% (but negative recall only 0.816)

Experiments – decision tree example



Conclusion and future work

- Created new system for tree mining
- Main part of the system is module for classification which uses several techniques; on real data set from logic course reached accuracy 97%
- System is going to be extended for new kinds of constructive tasks (such as tableau proofs)

Thank you