Graph Mining for Automatic Classification of Logical Proofs

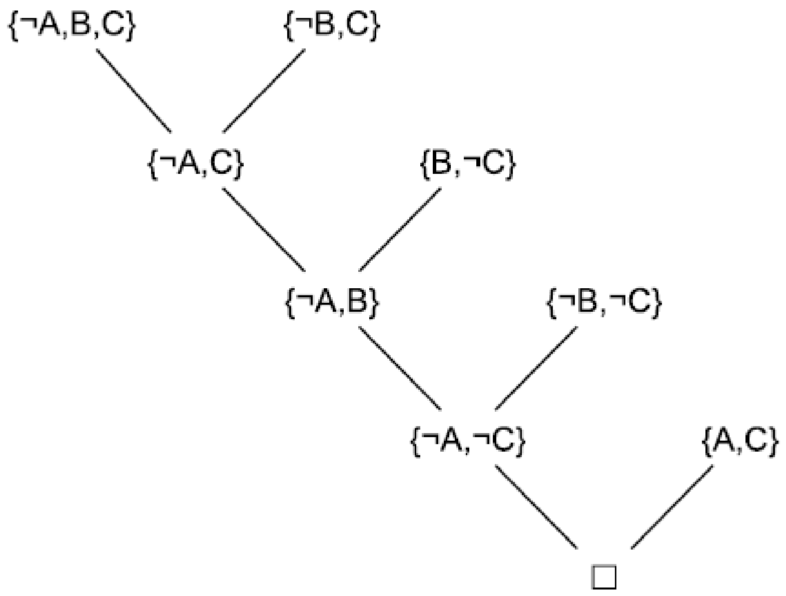
Karel Vaculík, Luboš Popelínský

FI MU

PV056 13. 5. 2014

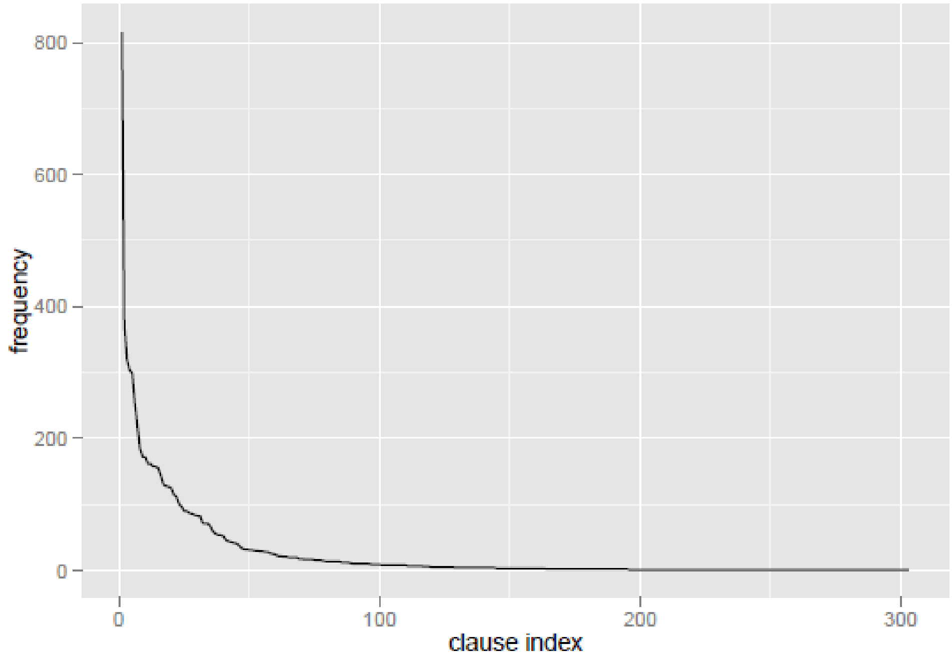
Data and data pre-processing

- 393 resolution proofs, correctness assigned by a teacher
  - 322 correct proofs
  - 71 incorrect proofs
- Common errors found by specialized scripts:
  - Resolving on two literals (see Figure)
  - Repetition of the same literal in a clause
  - Resolving on same literals



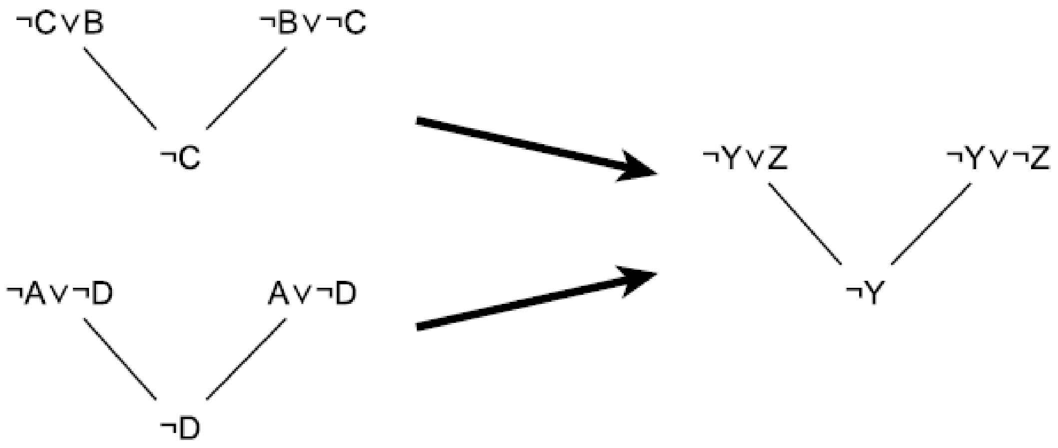
Mining subgraphs – first try

- Used algorithm: Sleuth
- Mining frequent subtrees with min. support 1% (infrequency of errors)
- Problems:
  - Inefficient on large datasets and/or large graphs
  - Different assignments of tasks (different propositional letters)



Mining subgraphs – new method

1. Extract all 3-node subgraphs (parents with the resolvent)
2. Perform generalization on these subgraphs
3. (Remove infrequent patterns)



Mining subgraphs – new method

- Ordering on list of literals based on number of negative and positive literals: NegLiteral × PosLiteral

$\neg C, \neg B, A, C \longrightarrow (0,1)_A \leq (1,0)_B \leq (1,1)_C \longrightarrow A \leq B \leq C$

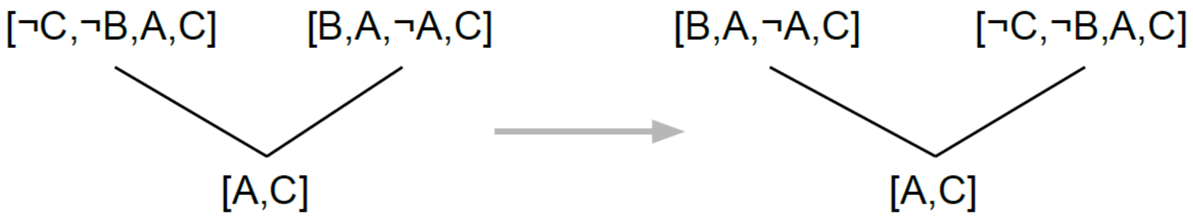
- Lexicographical ordering on the previous ordering – for node (clause) comparison

$\neg C, \neg B, A, C$		$(0,1) \leq (1,0) \leq (1,1)$		$\neg C, \neg B, A, C$
	→	II VI	→	VI
$B, A, \neg A, C$		$(0,1) \leq (0,1) \leq (1,1)$		$B, A, \neg A, C$

Mining subgraphs – new method

Procedure:

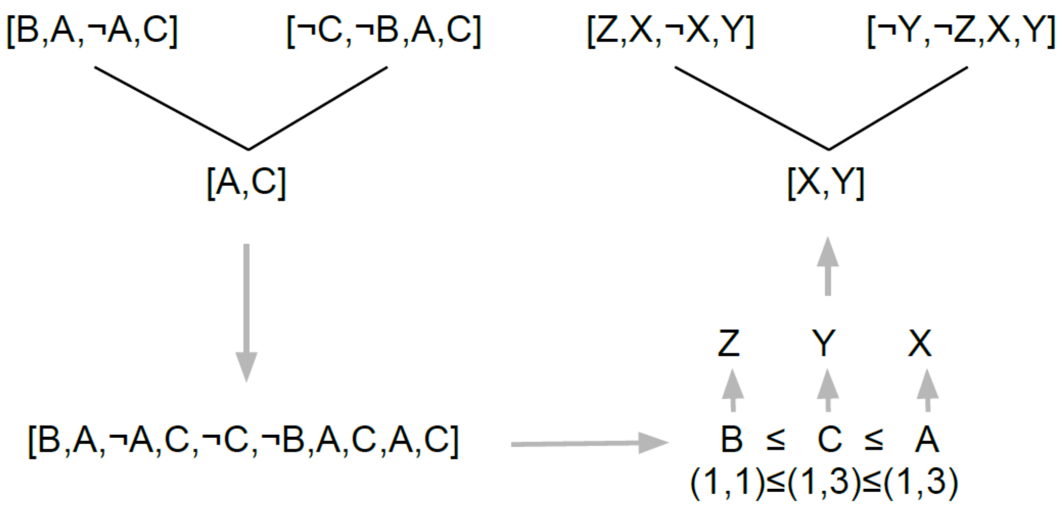
1. Compare parent nodes, smaller node will be first. For example:



Mining subgraphs – new method

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. For example:



Mining subgraphs – new method

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. For example $Z, \neg Y$ and $\neg Y, Z$ should be same.

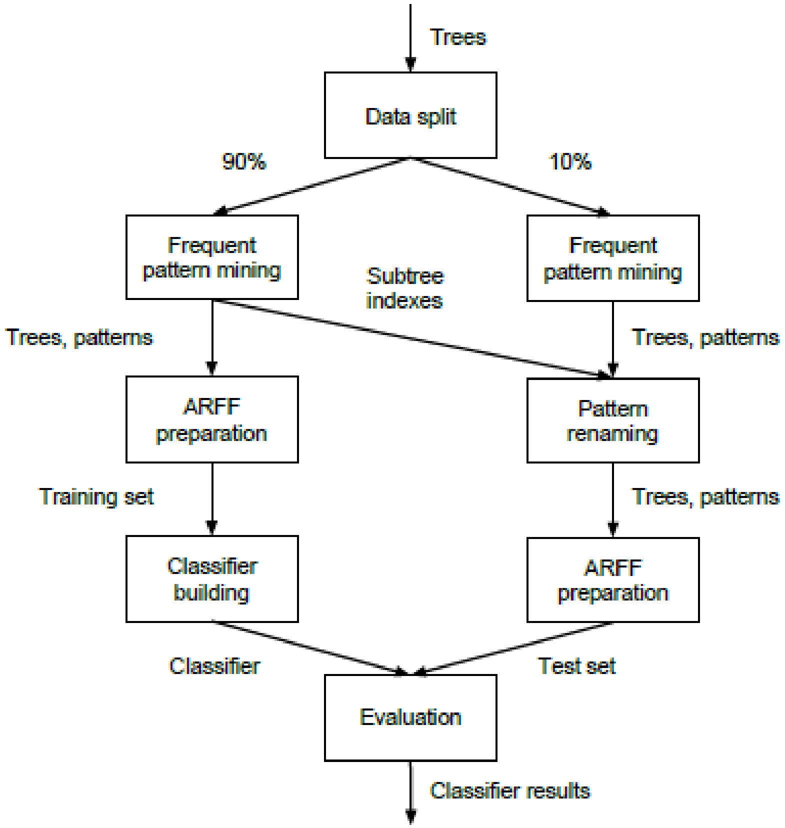
Experiments - classification

- Classes: *correct* or *incorrect* proof
- 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	incorrect
...	...	...	...	
false	true	...	true	correct

Experiments - classification

- Evaluation method:
  - 10-fold cross validation
- Classifiers: J48, SVM, …



Experiments - classification

- Results:

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

Conclusion

- Generalized subgraphs provide a useful representation of resolution proofs
- It is possible to classify resolution proofs on the basis of this representation
  - This is appropriate if the classes are not assigned clearly
  - For precise specification of classes it is better to use some exact algorithm

Current work

- New data for analysis
- Explanation of errors in proofs
- Extension of generalization method
- Exploitation of temporal information
- Outlier detection

Thank you.