Exploiting PDT in practical applications: experience of developers

Miloš Jakubíček and Vojtěch Kovář

NLP Centre
Faculty of Informatics, Masaryk University
Brno, Czech Republic
{xjakub, xkovar3}@fi.muni.cz

ÚFAL MFF UK, 15. 11. 2010

Outline

Background: what we do

- Natural Language Processing Centre
- about 15 employees
- main areas of interest:
  - corpus linguistics (Sketch Engine)
  - Czech morphology (`ajka`, `majka`)
  - parsing of Czech (`synt`, `SET`)
  - logical analysis (TIL)
  - computer lexicography (DEB platform)
- focus on practical deliverable applications rather than theoretical linguistics

Background: what we do

- Natural Language Processing Centre
- about 15 employees
- main areas of interest:
  - **corpus linguistics (Sketch Engine)**
  - Czech morphology (`ajka`, `majka`)
  - **parsing of Czech (`synt`, `SET`)**
  - **logical analysis (TIL)**
  - computer lexicography (DEB platform)
- focus on practical deliverable applications rather than theoretical linguistics

Background: what we do

- Natural Language Processing Centre
- about 15 employees
- main areas of interest:
  - **corpus linguistics (Sketch Engine)**
  - Czech morphology (`ajka`, `majka`)
  - **parsing of Czech (`synt`, `SET`)**
  - **logical analysis (TIL)**
  - computer lexicography (DEB platform)
- focus on practical deliverable applications rather than theoretical linguistics

Relevant projects

- the `synt` system
  - phrase-structure grammar, chart parser
- developing the new SET parser
  - rule-based pattern matching system
  - constituent, dependency, hybrid trees
- phrases annotation
  - NPs, PPs, VPs, clauses
  - just deciding correctness of automatic output
  - mapping to complex valency frames
  - $\longrightarrow$ semantic classes of phrases
- anaphora resolution

What PDT means for us

- **the biggest source of Czech syntactically annotated data**
  - created by skilled domain experts
  - reference corpus – looking for the „right way”
  - $\longrightarrow$ testing parsing results
  - $\longrightarrow$ extracting statistical models
  - in general all kinds of useful syntactic data
- markup of anaphoric expressions on the t-layer
  - training and testing automatic anaphora resolution

Using PDT in particular projects

- `synt` parser
  - measuring coverage, testing
- SET parser development
  - „training" and testing
- phrases annotation
  - extracting phrases and clauses
  - „gold standard" phrases
- anaphora resolution
  - extracting anaphora information
  - training and testing of the tools

PDT Problems

- with regard to the particular applications
- from the technical point of view (usability)
  - $\longrightarrow$ we expect as straightforward usage of the resource as possible
  - $\longrightarrow$ ideally for all particular applications
- we do not want to infirm the theoretical background
- but provide feedback from a different point of view
  - for discussion
  - to help identifying problems
  - for easier use of PDT in the future

Problem #1 – extensive annotation manuals

- m-layer: 50 pages; a-layer: 300 pages; t-layer: 1200 pages
- lots of exceptions and special cases
- to understand/use the data, one has to read it through and memorize it
  - → this is not easy at all
- description granularity
  - often goes beyond what most people are able to distinguish
  - e.g. „zakladatel a prezident firmy" vs. „prezident a zakladatel firmy" (t-layer manual, 5.6.1.1)
- the same holds for annotators
  - → errors, inconsistencies

Problem #2 – sentence selection

- „0:0, 1:1, 2:1, 5:3, ...”
- „Dítě 0 - 15: 4”
- → more exceptions and special cases
- → more pages in manual
- → more errors
- for discussion
  - do we really need to have such sentences in annotated treebank?
  - are the dependency structures meaningful for such sentences?
  - distinction between „Czech sentences” and „technical data”?
- our opinion
  - such „technical data” do not really represent natural language
  - parser should handle them in a separate mode

Problem #3 – strong formal requirements

- dependency formalism is simple (which is great), but...
- each token needs to have dependency
  - → complicated structures on numbers and punctuation
- problem with marking coordinations...
  - ... and common expansions of their members
  - to be able to read them, one needs to have the edge labels
  - → complicates reading the structures

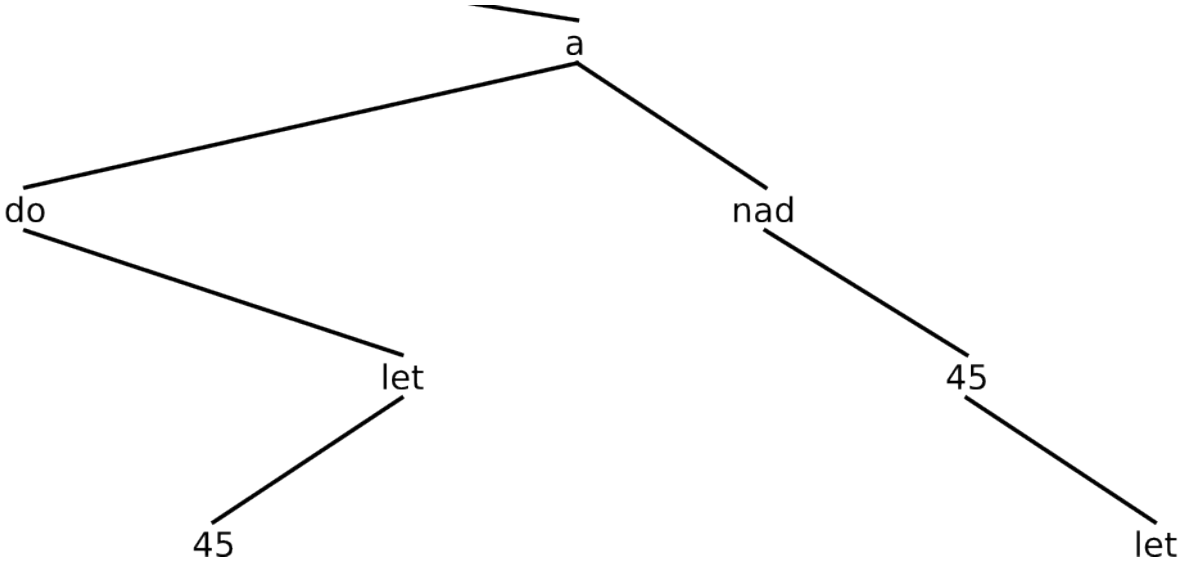
Problem #4 – phrases extraction

- syntactically annotated corpus
  - it should be straightforward to extract flat phrases
  - $\rightarrow$ NPs, PPs, verb groups, clauses
- we spent 2 days working on algorithm for NP extraction
  - still not completely OK
  - precision ca. 97.5 %
  - (some of the rest are annotation errors)
  - problems mainly with coordination structures
- clauses extraction
  - crucial for some AR algorithms
  - are being annotated separately

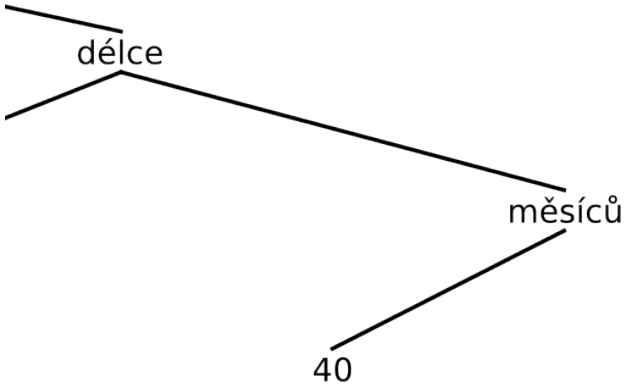
Problem #5 – errors, inconsistencies

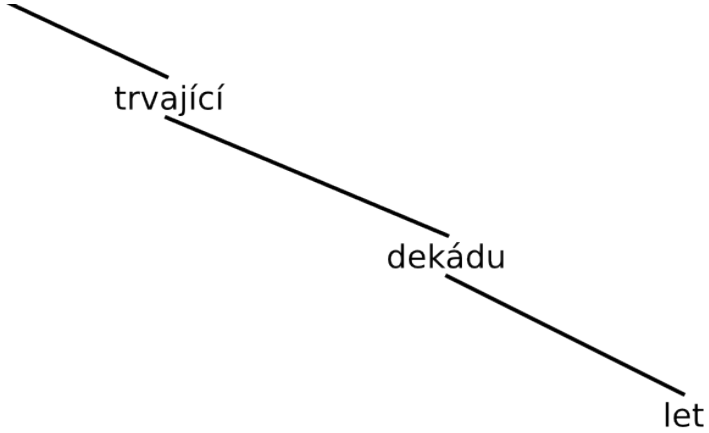
- partly connected with previous issues
- random errors
- systematic inconsistencies
  - number – unit
  - passive verb phrases
  - punctuation
- very very rough estimation of error rate
  - $\rightarrow$ 5 – 10 % of edges
- however, proper analysis is needed

Number – unit I

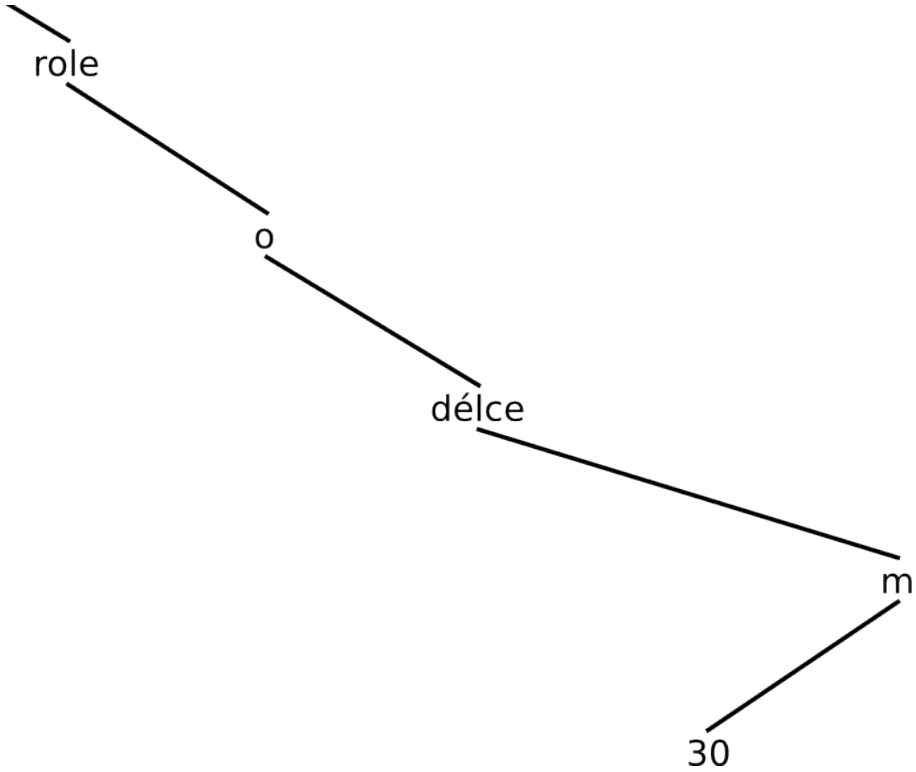


Number – unit II

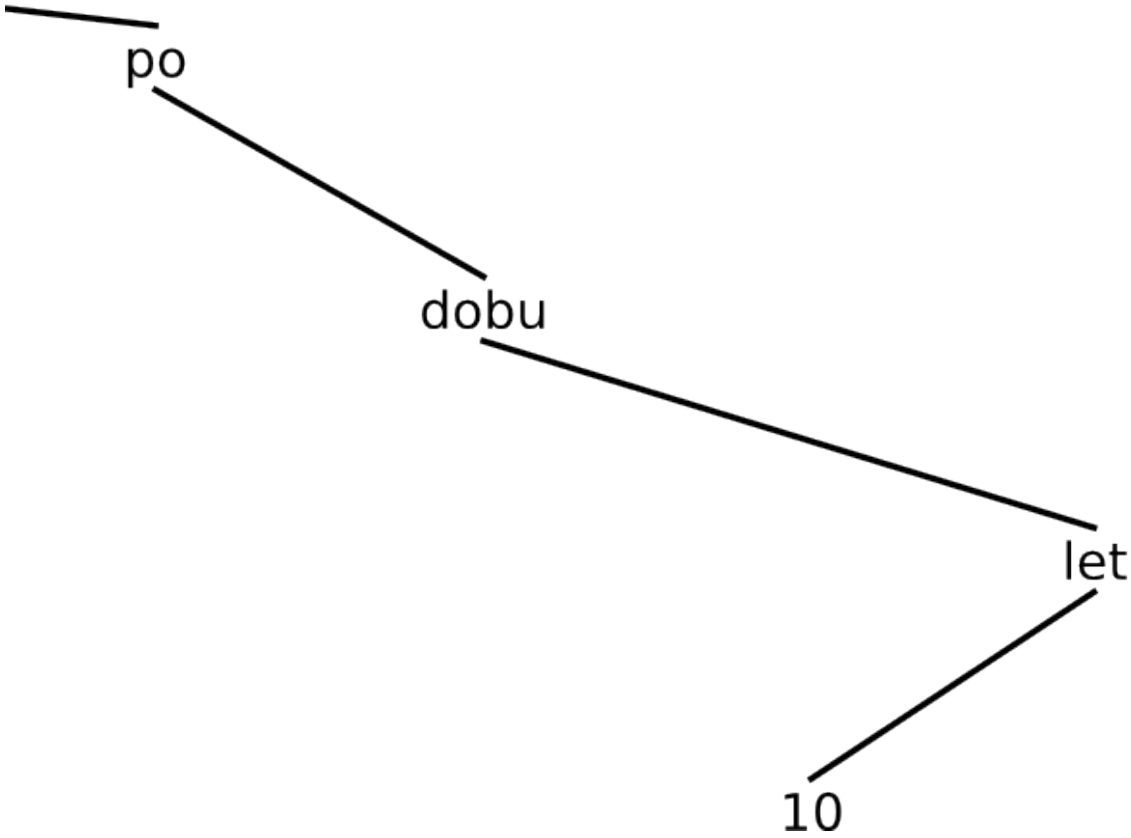




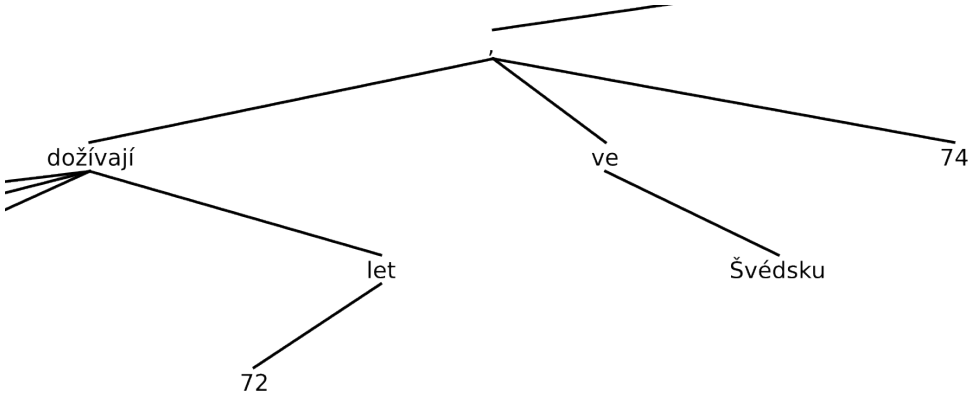
Number – unit III

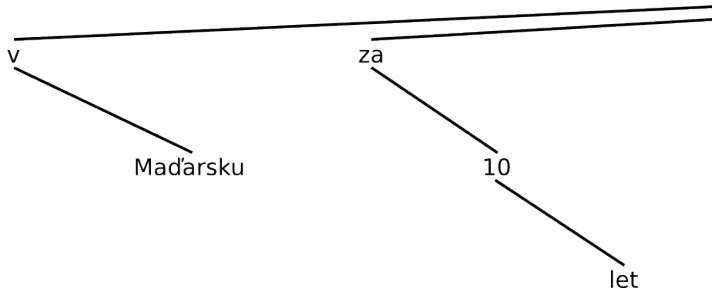


Number – unit IV

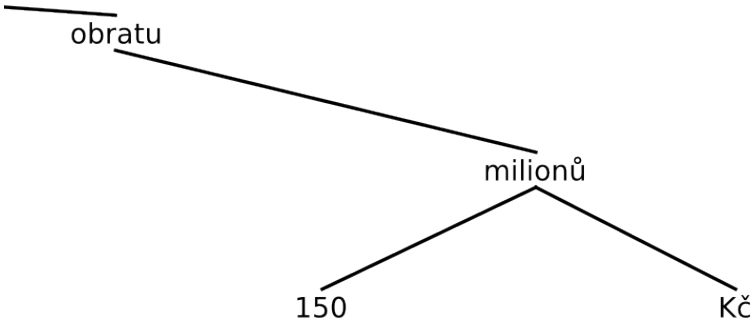


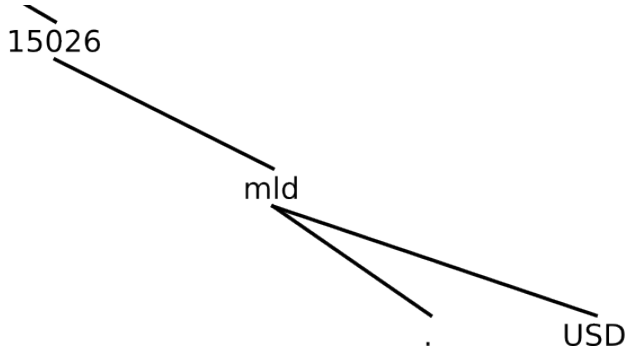
Number – unit V



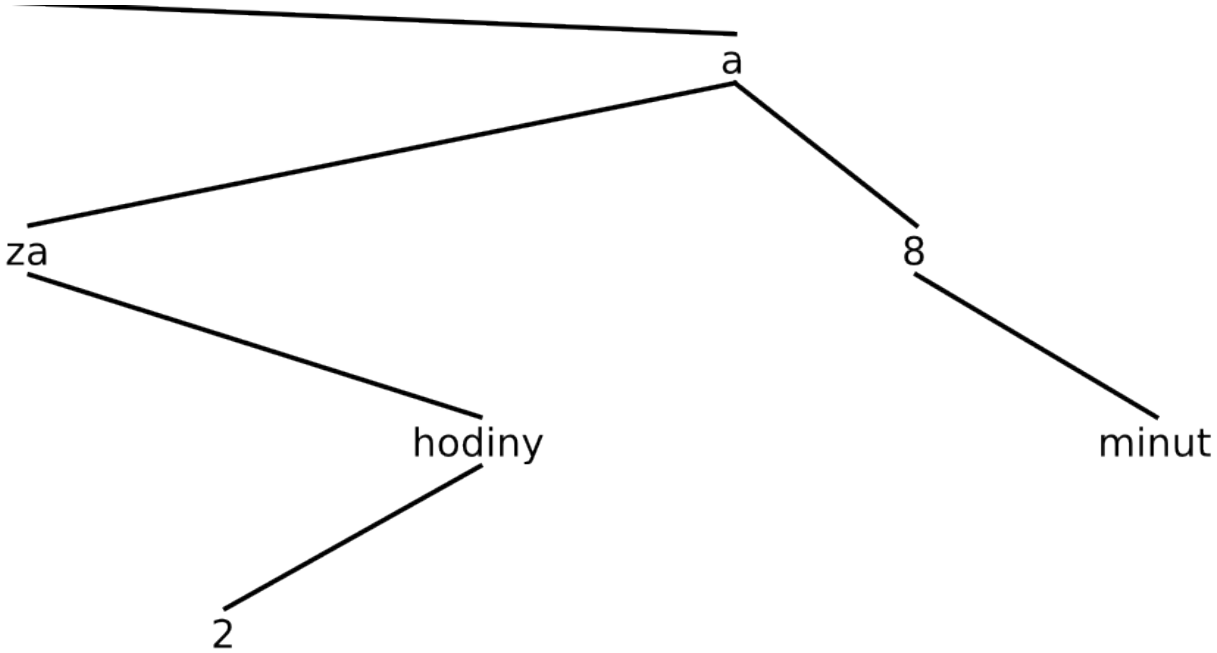


Number – unit VI

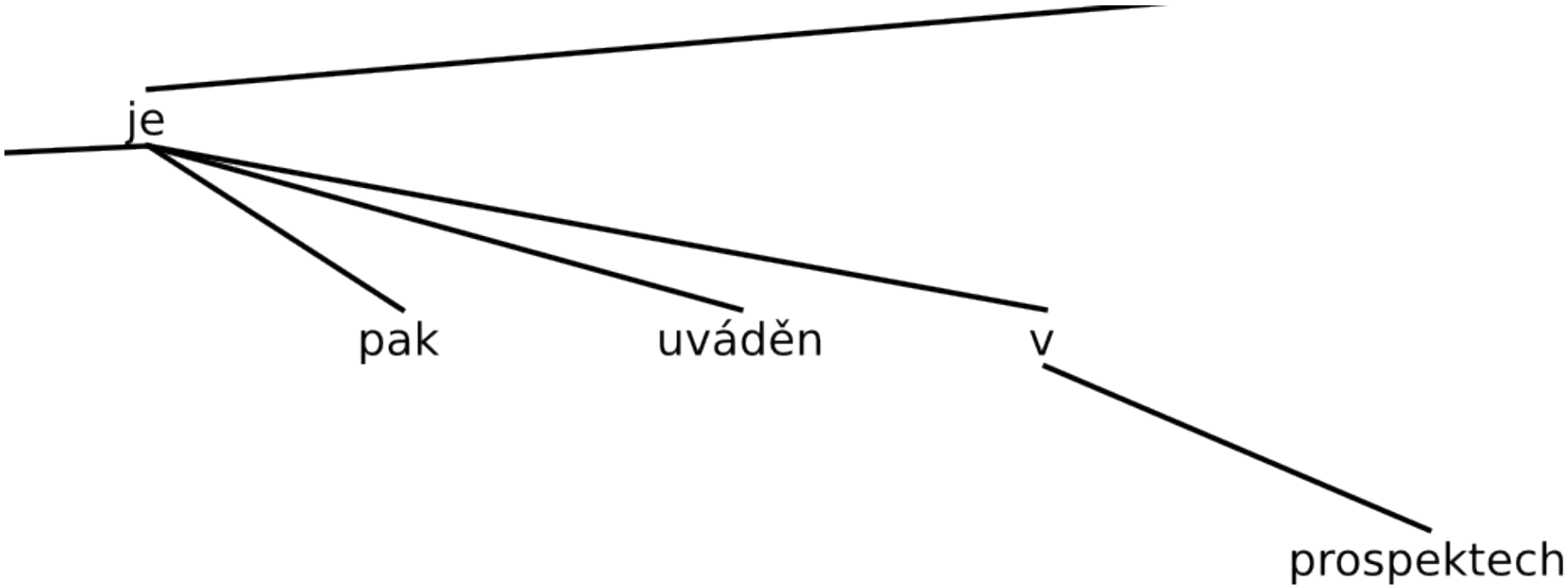




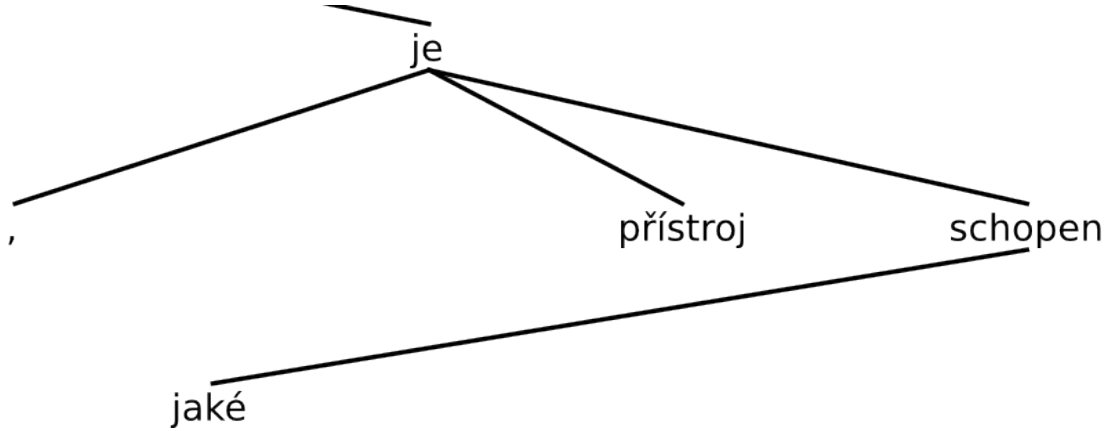
Number – unit VII



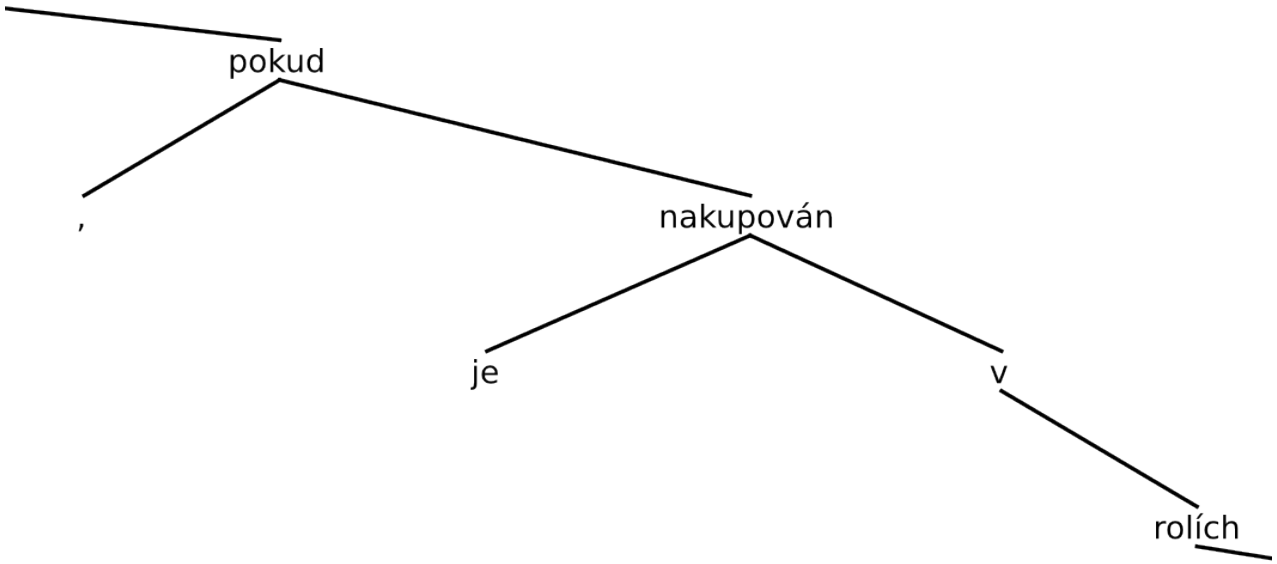
Passive verb constructions I



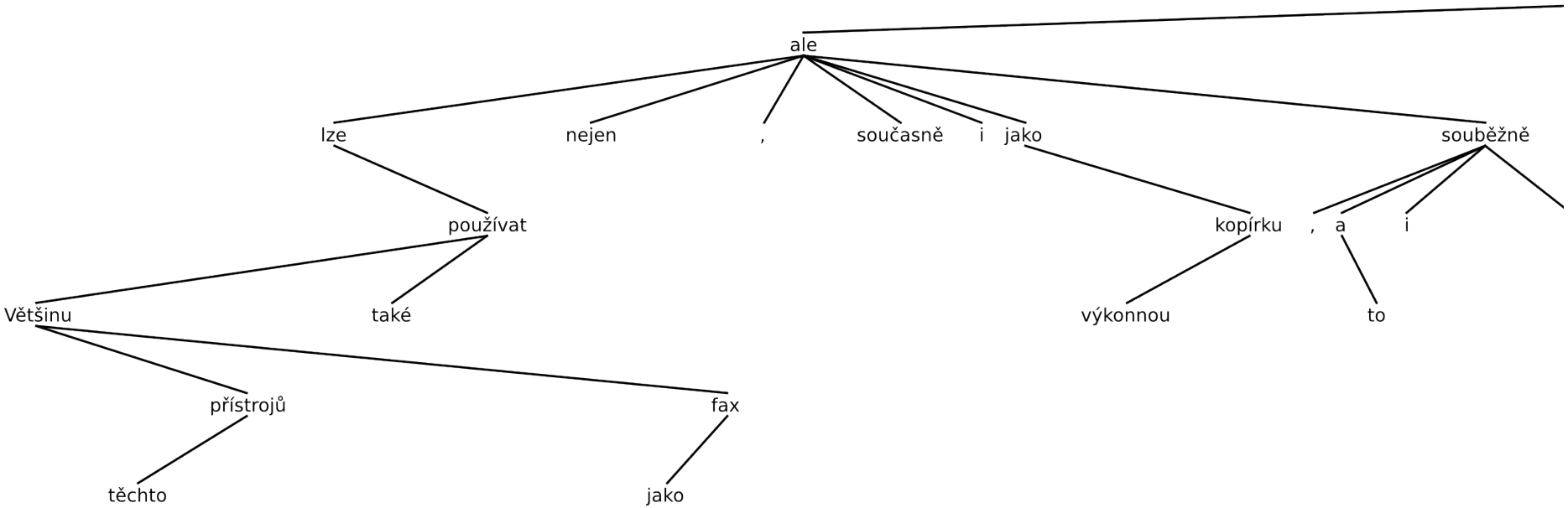
Passive verb constructions II



Passive verb constructions III



Other issues



Solutions?

- errors, inconsistencies
  - semi-automatic heuristic checks
  - extracting simple information (phrases) for checking
- complexity
  - revise some of the annotation principles
  - divide usual Czech sentences from technical data
  - „worse is better" principle in software development:
    - simplicity
    - consistency
    - correctness
    - completeness

Thank you

- Thank you for your attention
  - looking forward to the discussion