Treex: Modular NLP Framework

Martin Popel
ÚFAL (Institute of Formal and Applied Linguistics)
Charles University in Prague

LATE Lunch, DFKI
November 4th 2010, Saarbrücken, Germany

Outline

Treex

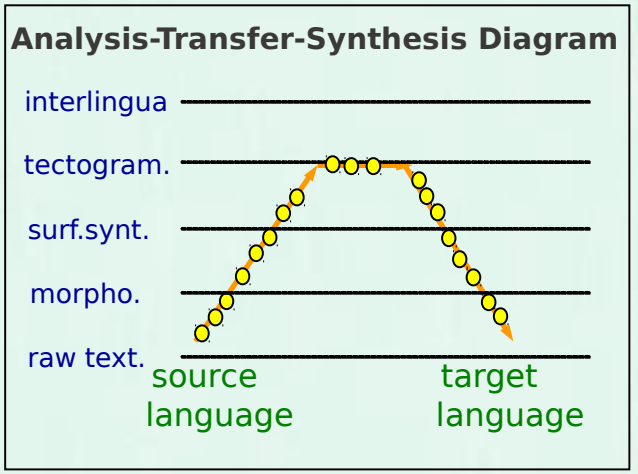
- Motivation
- Layers of language description in PDT
- Treex architecture
- Treex internals
- Future plans
- Conclusion and examples

Motivation for creating Treex

Treex

Originally a framework for a linguistically motivated MT system

- called TectoMT (both the MT system and framework)
- deep syntactic (tectogrammatical) transfer
- started with English to Czech direction
- translation process divided to ~ 90 “blocks“
- combining statistical and rule-based blocks



Goals:

- elegant integration of in-house and third-party NLP tools
- modularity, reusability, cooperation
- ability to easily modify and add code in full-fledged programming language (Perl)

Motivation for creating Treex

Treex

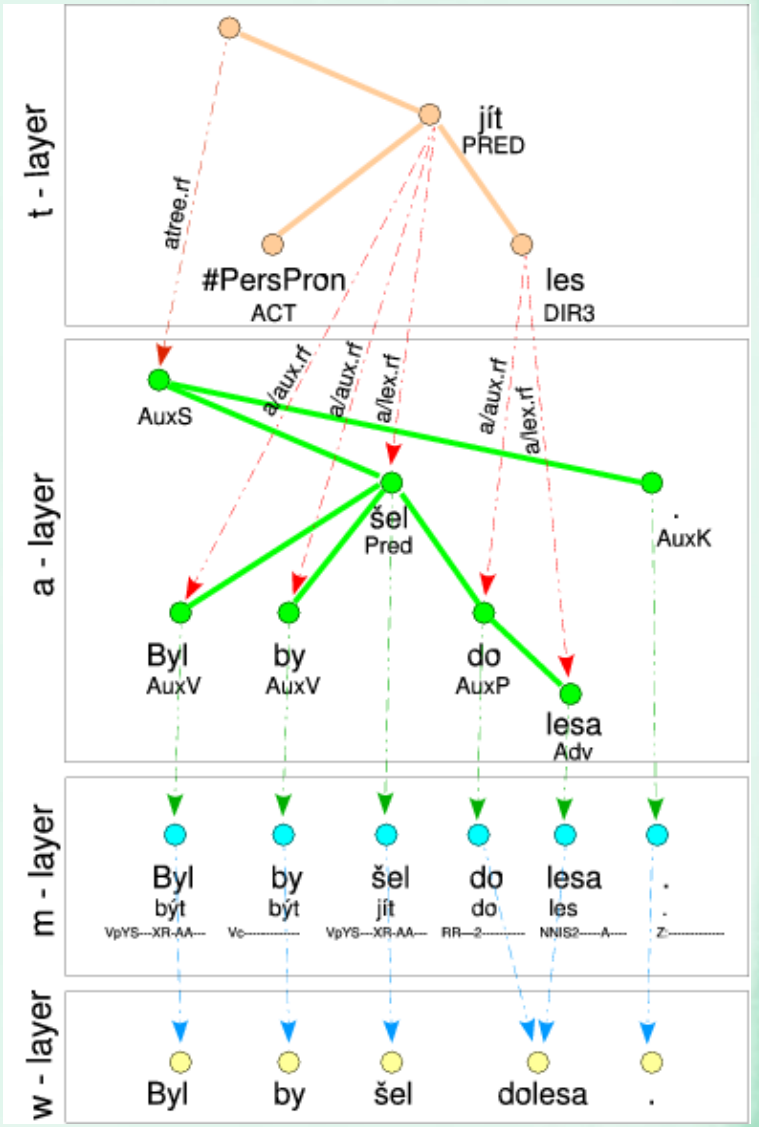
Now used for many other projects,
not limited to MT nor tectogrammatics:

- automatic alignment & annotation of a parallel treebank (CzEng)
- support for manual annotations (PEDT)
- lemmatization, tagging, parsing
- named entity recognition, information retrieval, coreference
- preprocessing for phrase-base MT
  - change word order, append determiners to nouns,...
  - add deep-syntactic features as an input for factored translation
- conversions, evaluations, etc.

4 layers of language description
implemented in Prague Dependency Treebank (PDT)

Treex

- word layer
  raw (tokenized) text

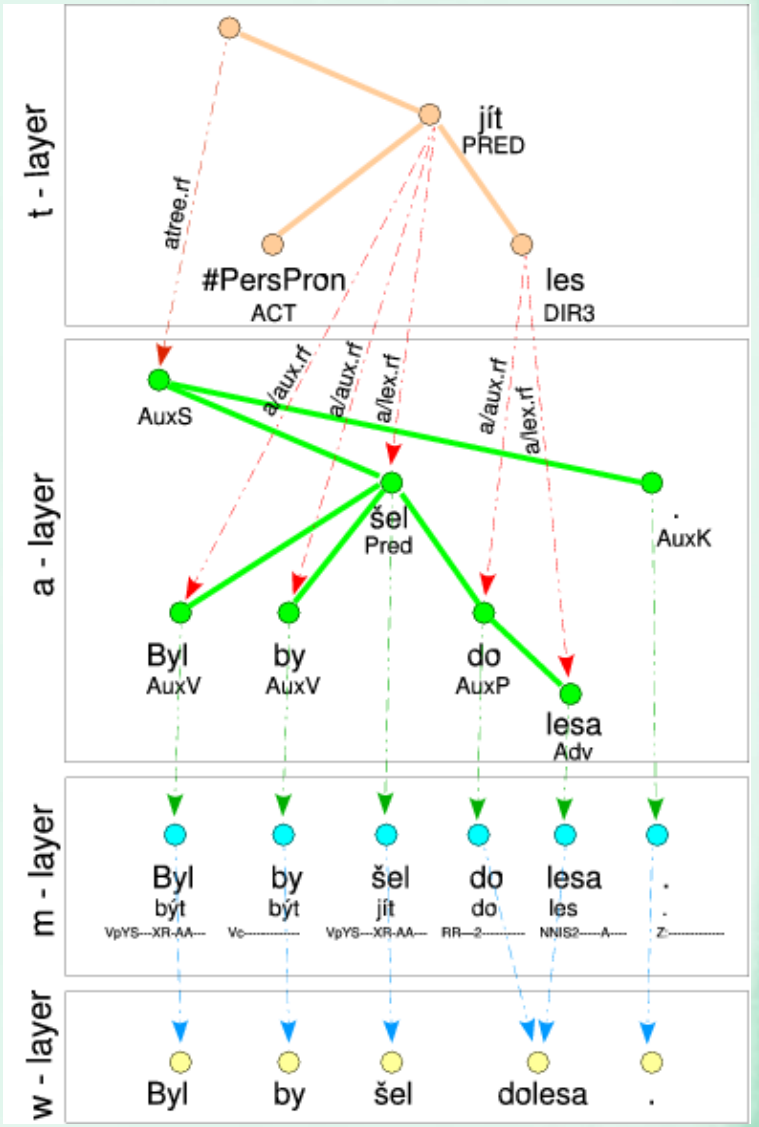


4 layers of language description
implemented in Prague Dependency Treebank (PDT)

Treex

- **morphological layer**
  lemma & POS tag for each word

- word layer
  raw (tokenized) text

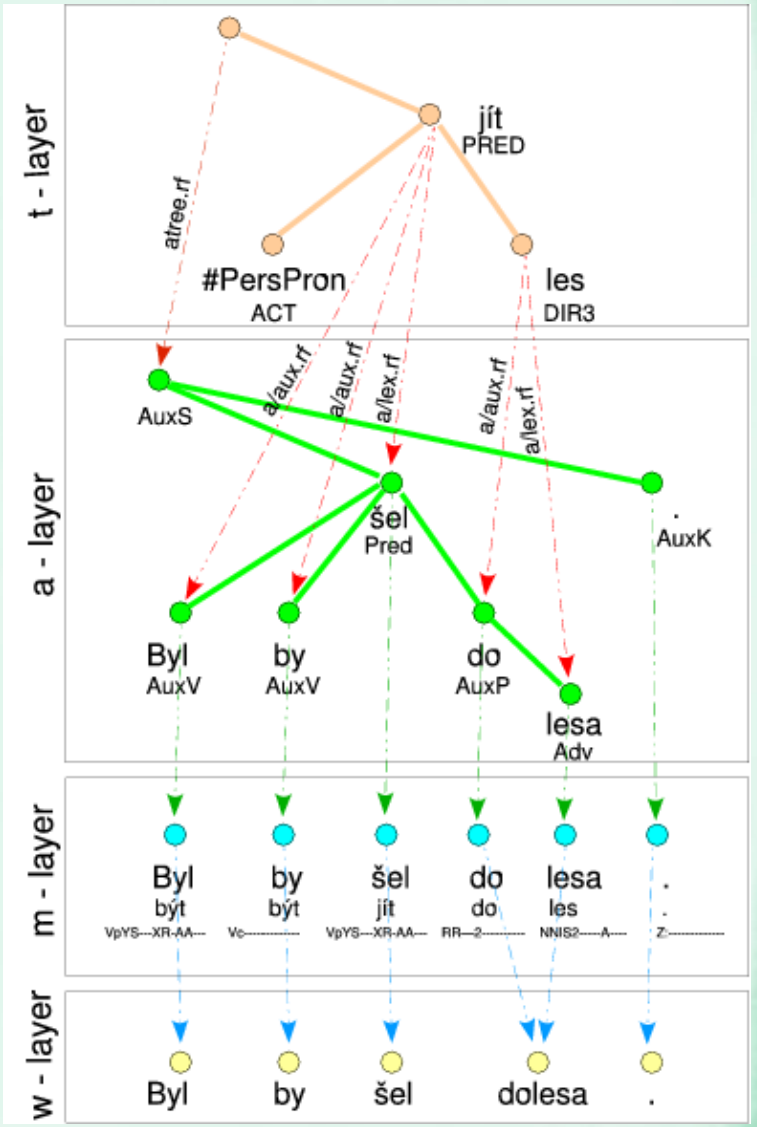


4 layers of language description

implemented in Prague Dependency Treebank (PDT)

Treex

- **analytical layer**
  surface-syntactic dependency trees, labeled edges
- **morphological layer**
  lemma & POS tag for each word
- word layer
  raw (tokenized) text

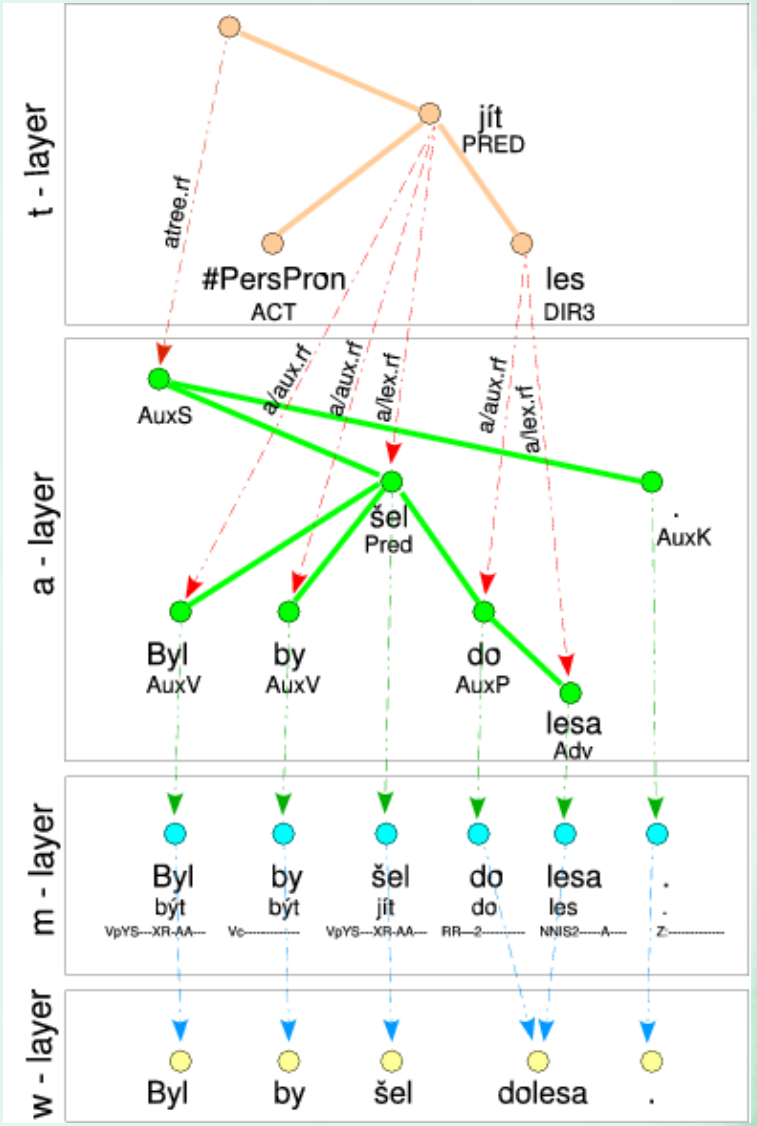


4 layers of language description

implemented in Prague Dependency Treebank (PDT)

Treex

- **tectogrammatical layer**
  deep-syntactic dependency trees
- **analytical layer**
  surface-syntactic dependency trees, labeled edges
- **morphological layer**
  lemma & POS tag for each word
- word layer
  raw (tokenized) text



4 layers of language description
implemented in Prague Dependency Treebank (PDT)

Treex

- **tectogrammatical layer**
  deep-syntactic dependency trees

- abstraction from many language-specific phenomena

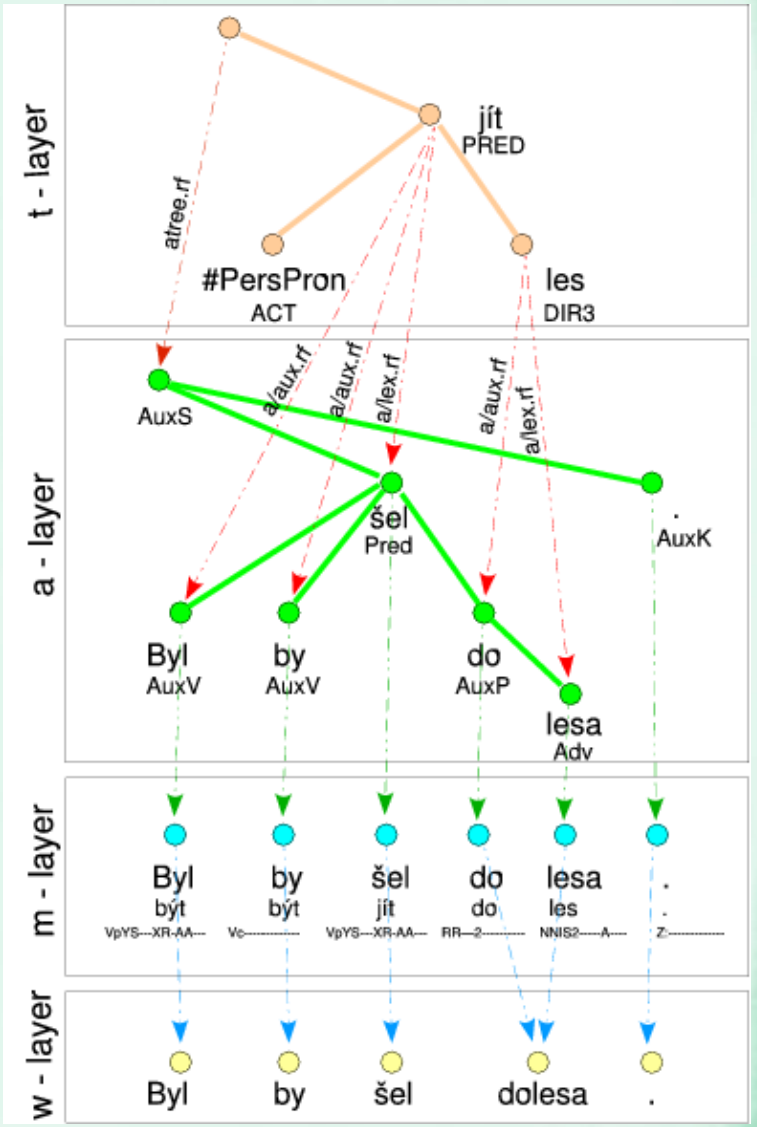
- autosemantic (meaningful) words
  ~ **nodes**

- functional words (prepositions, auxiliaries)
  ~ **attributes**

- syntactic-semantic relations (dependecies)
  ~ **edges**

- added nodes (e.g. becasue of pro-drop)

- ...



Treex architecture processing units

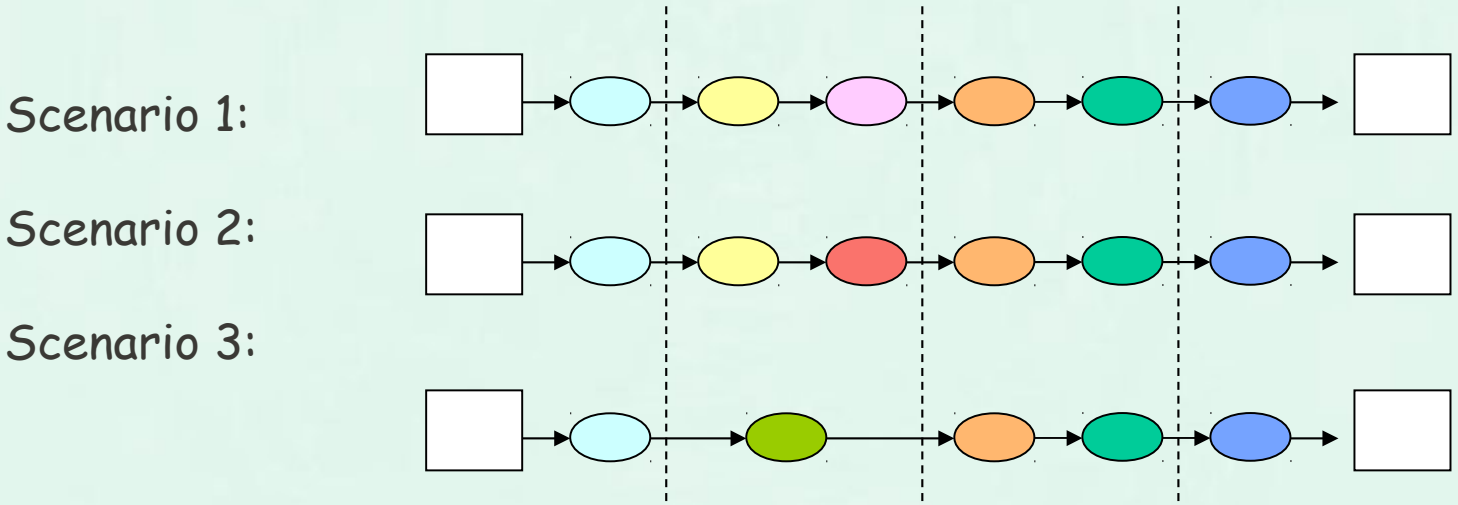
Treex

- **block** – elementary processing unit in Treex
  - corresponding to a given NLP subtask
  - one Perl class, saved in one file
- **scenario** – a sequence of blocks
  - saved in plain text files
  - just a list of the blocks' names and their parameters
- **application** – represents an end-to-end NLP task
  - conversion of the input to Treex internal format (XML)
  - possibly split into more files
  - applying a scenario to the files (loaded in memory)
  - conversion to the desired output format

Treex architecture processing units

Treex

Blocks can be easily substituted with an alternative solution.



Treex architecture processing units

Treex

Blocks can be easily substituted with an alternative solution.

Scenario A

```
Sentence_segmentation_simple
Penn_style_tokenization
TagMxPost
Lemmatize_mtree
McD_parser
```

Scenario B

```
Each_line_as_sentence
Tokenize_and_tag
Lemmatize_mtree
Malt_parser
```

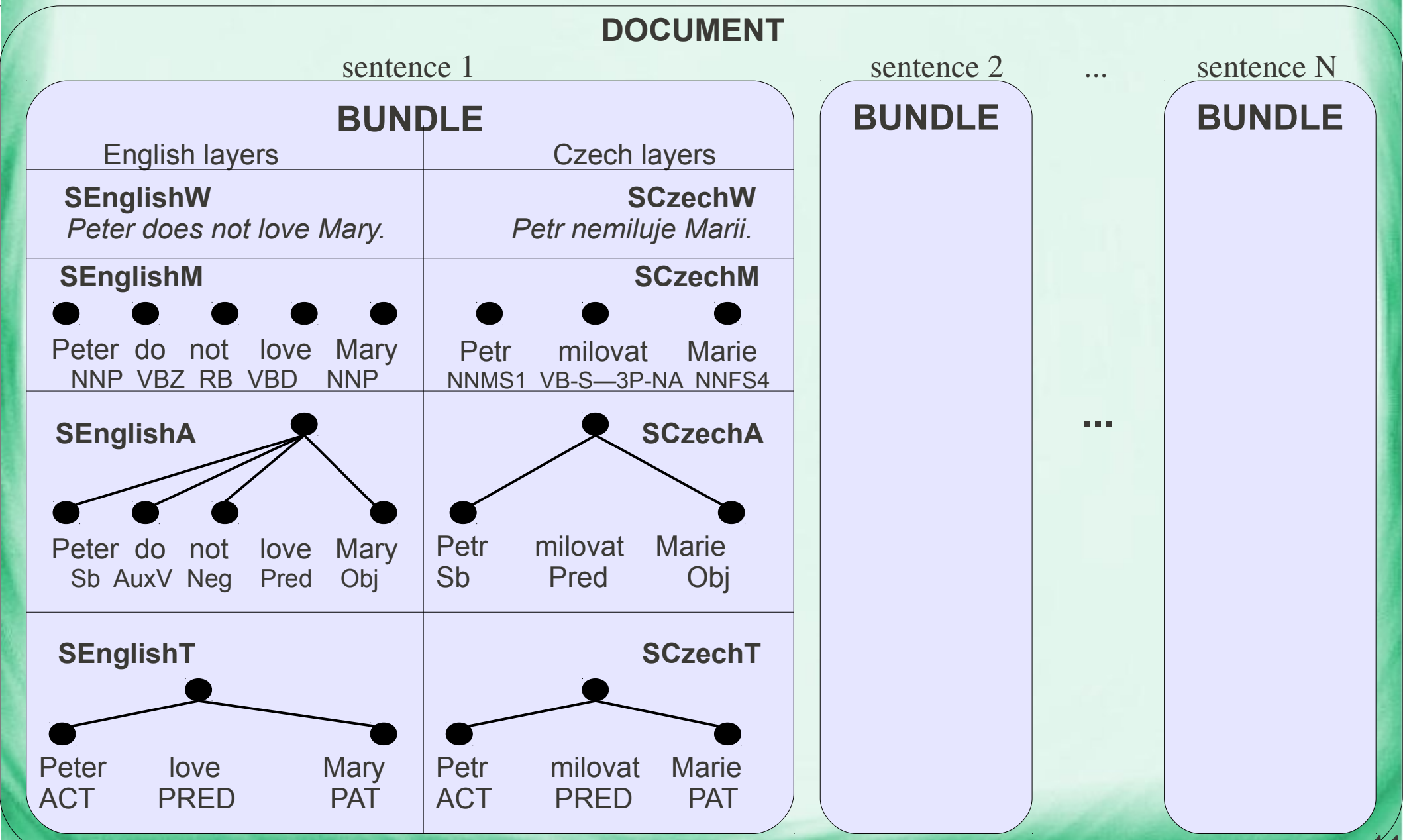
Treex architecture
data units

Treex

- **Document**
  - stored in one file
  - sequence of sentences
- **Bundle**
  - corresponds to one sentence
  - “bundle of trees”
- **Tree**
  - direction (S=source, T=target)
  - language (Arabic, Czech, English, German,...)
  - layer of language description (M, A, T)

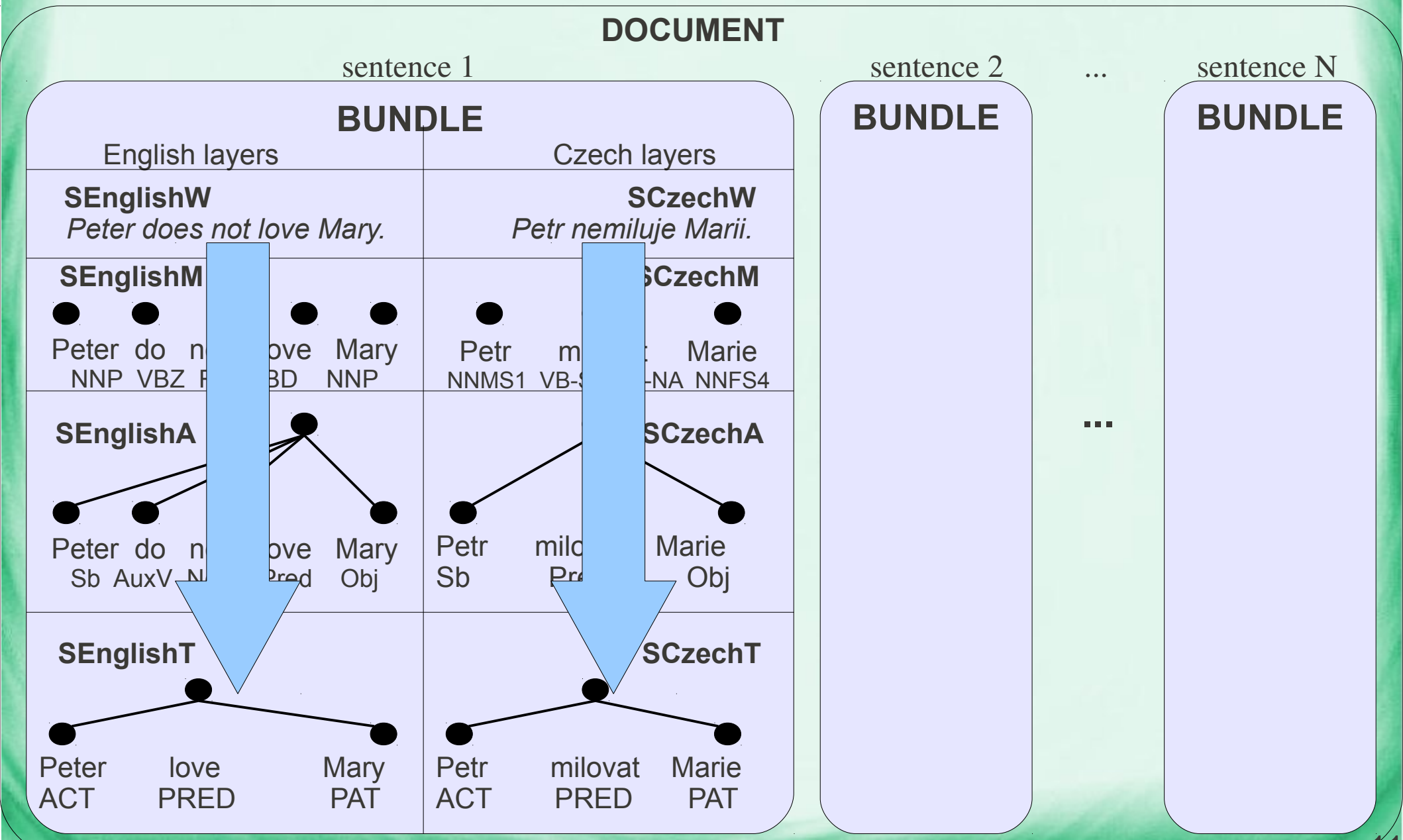
Treex architecture data units

Treex



Treex architecture data units

Treex



Treex architecture data units

Treex

DOCUMENT

sentence 1 | sentence 2 | ... | sentence N

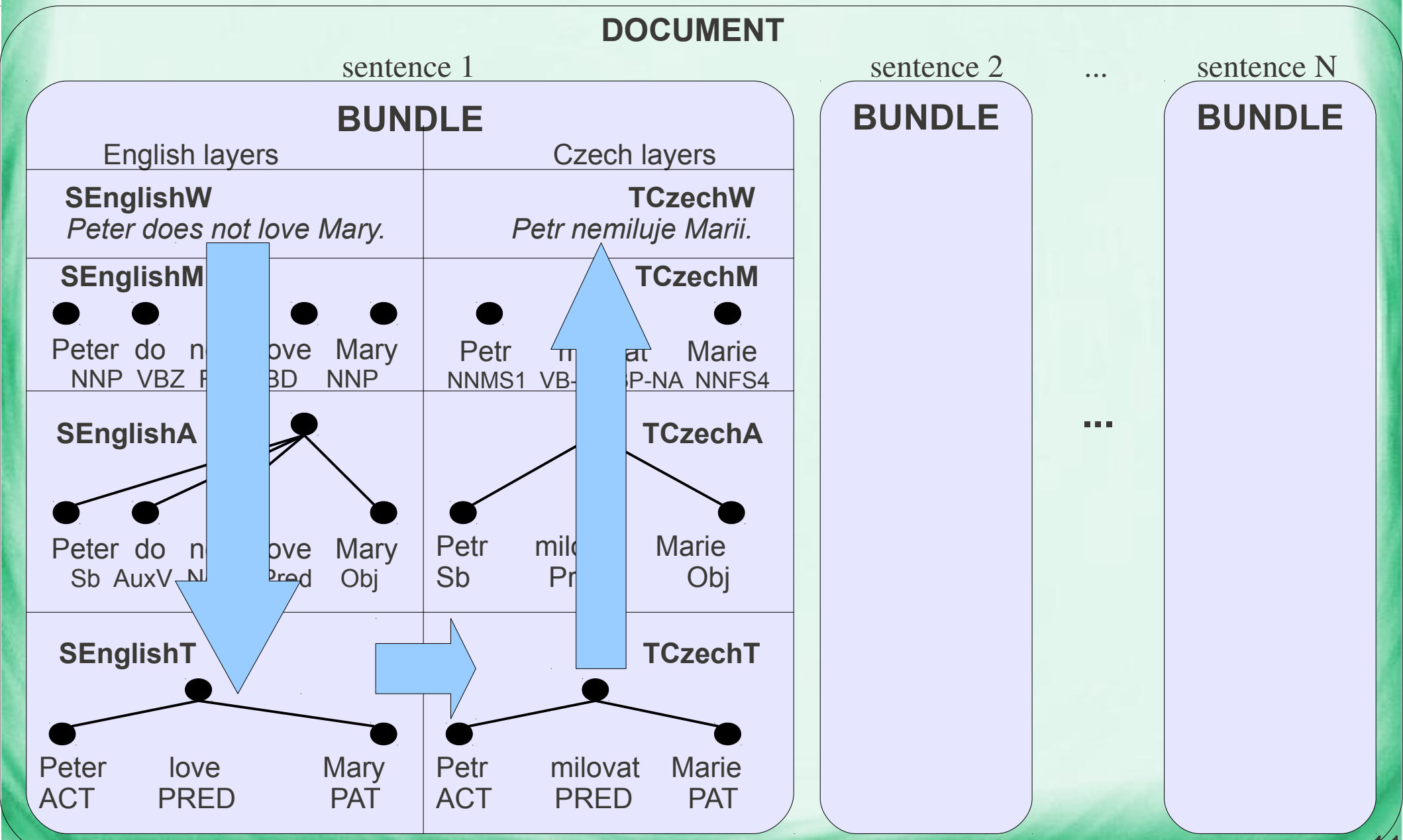
BUNDLE | **BUNDLE** | **BUNDLE**

English layers	Czech layers
SEnglishW *Peter does not love Mary.*	**TCzechW** *Petr nemiluje Marii.*
SEnglishM Peter do not love Mary NNP VBZ RB VBD NNP	**TCzechM** Petr milovat Marie NNMS1 VB-S—3P-NA NNFS4
SEnglishA Peter do not love Mary Sb AuxV Neg Pred Obj	**TCzechA** Petr milovat Marie Sb Pred Obj
SEnglishT Peter love Mary ACT PRED PAT	**TCzechT** Petr milovat Marie ACT PRED PAT

...

Treex architecture data units

Treex



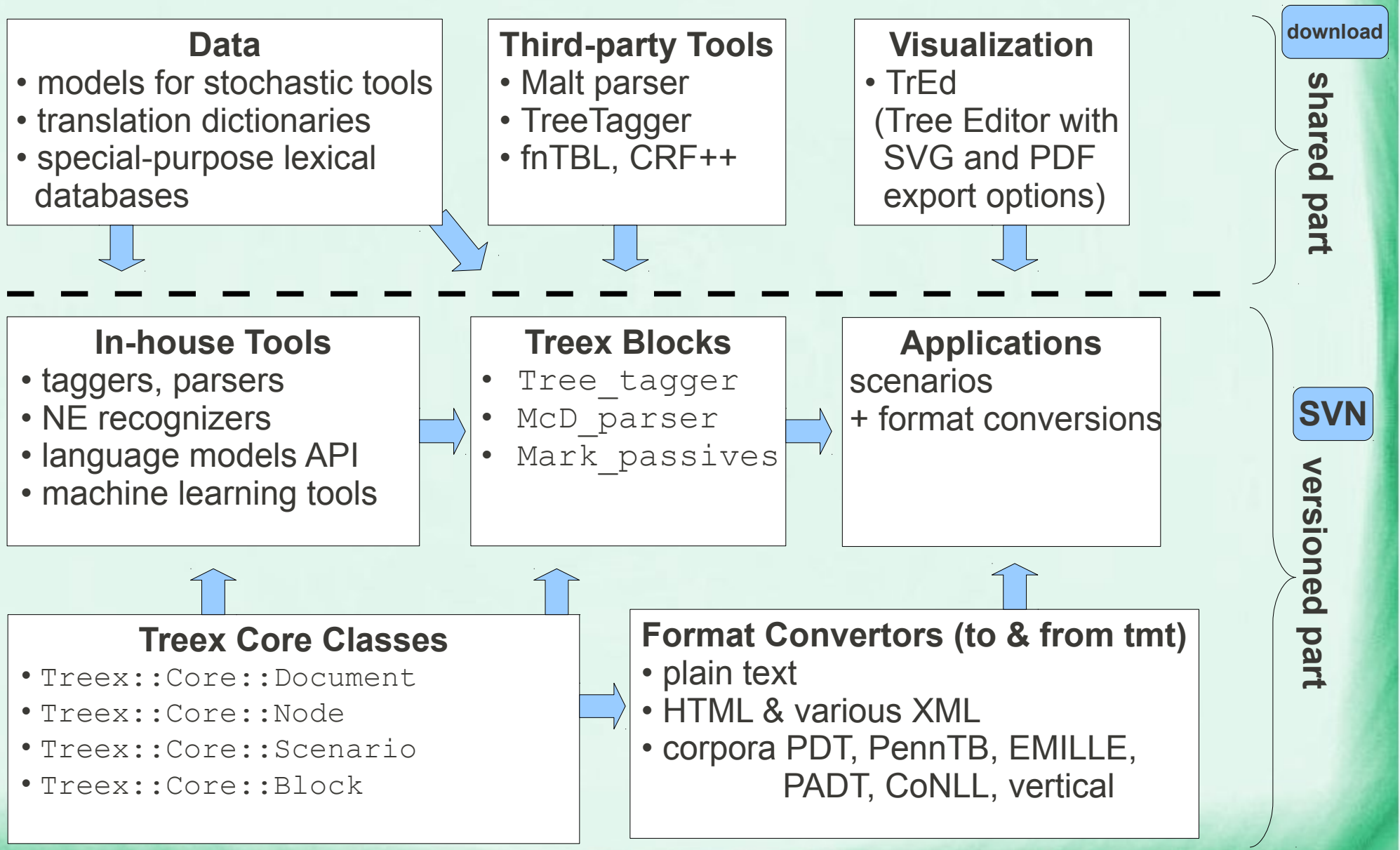
Internals – Design decisions

Treex

- Perl (wrappers for binaries, Java,...)
- Linux (some applications platform-independent)
- OOP (ClassStd, Moose)
- Open source (GNU GPL for the versioned part)
- Neutral w.r.t. methodology (statistical, rule-based)
- Multilingual
- Open standards (Unicode, XML)

Internals – Components

Treex



Internals – Statistics

Treex

- Developed since 2005, over ten developers
- Over 400 blocks (140 English, 120 Czech, 60 English-to-Czech, 30 other languages, 50 language independent)
- Taggers (5 English, 3 Czech, 1 German and Russian)

  Parsers (Dep. 2 English, 3 Czech, 2 German; Const. 2 English)

  Named Entity Recognizers (2 Czech,1 English)
- Speed example: Best version of English-to-Czech MT

  1.2 seconds per sentence plus 90 seconds loading,

  with 20 computers in cluster: 2000 sentences in 4 min

Internals – Statistics

Treex

- Developed since 2005, over ten developers
- Over 400 blocks (140 English, 120 Czech, 60 English-to-Czech, 30 other languages, 50 language independent)
- Taggers (5 English, 3 Czech, 1 German and Russian, Tamil)

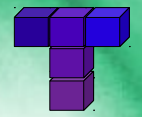
  Parsers (Dep. 2 English, 3 Czech, 2 German; Const. 2 English)

  Named Entity Recognizers (2 Czech,1 English)
- Speed example: Best version of English-to-Czech MT

  1.2 seconds per sentence plus 90 seconds loading,

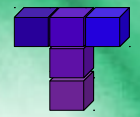
  with 20 computers in cluster: 2000 sentences in 4 min

Future plans



- Reimplementation of core components
- CPAN release
- Adding new languages more easily
- Improved parallelization support
- Faster code, smaller files,...

Conclusion
Treex main properties

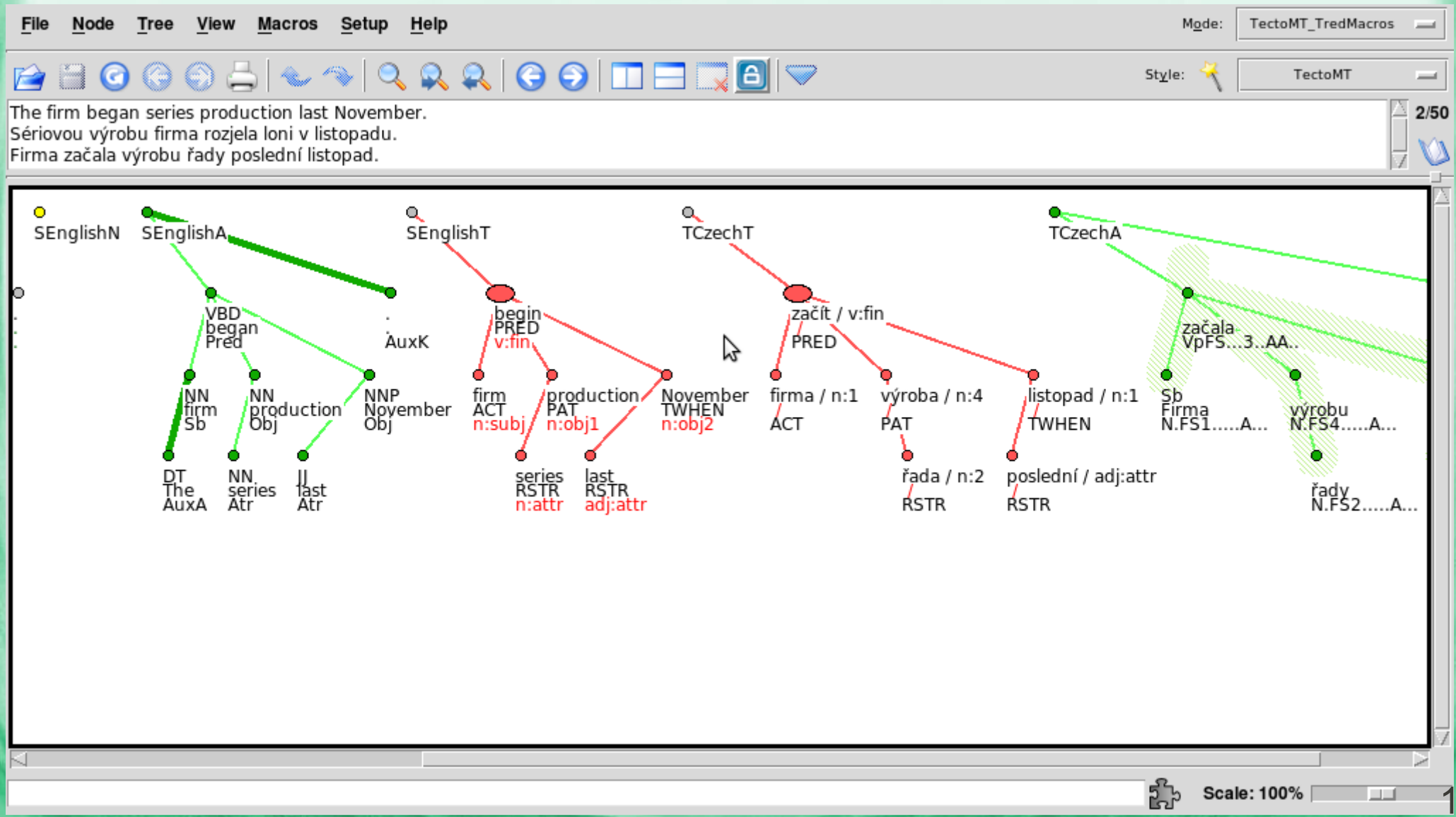


- emphasized efficient development, modular design and reusability
- stratificational approach to the language
- unified object-oriented interface for accessing data structures
- comfortable development

TrEd visualization

Treex

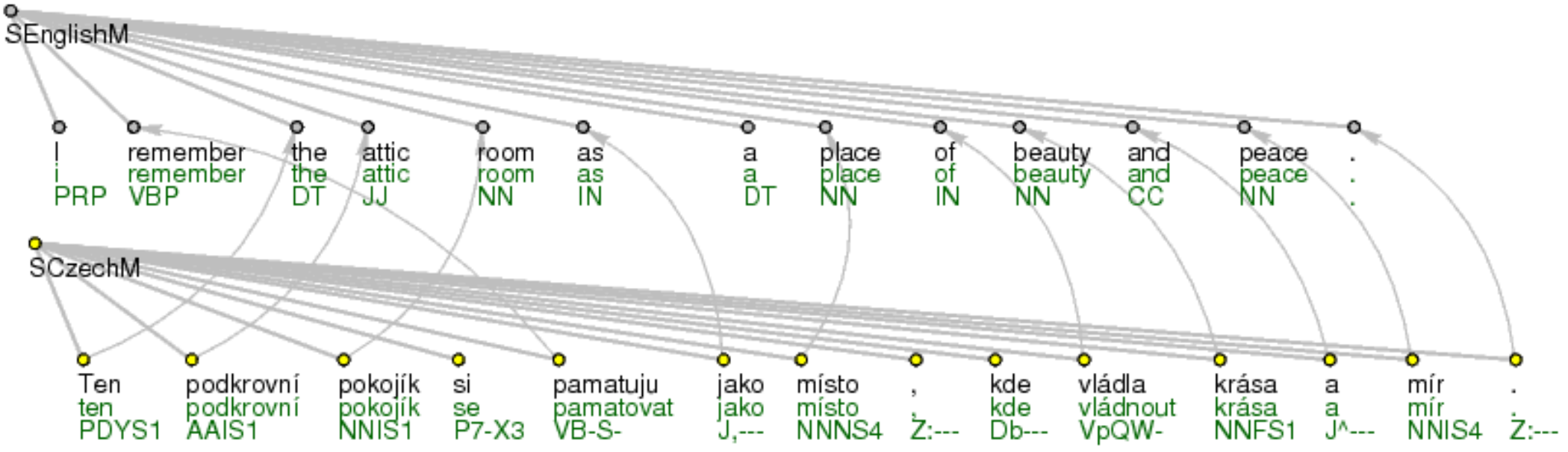
translation



TrEd visualization

Treex

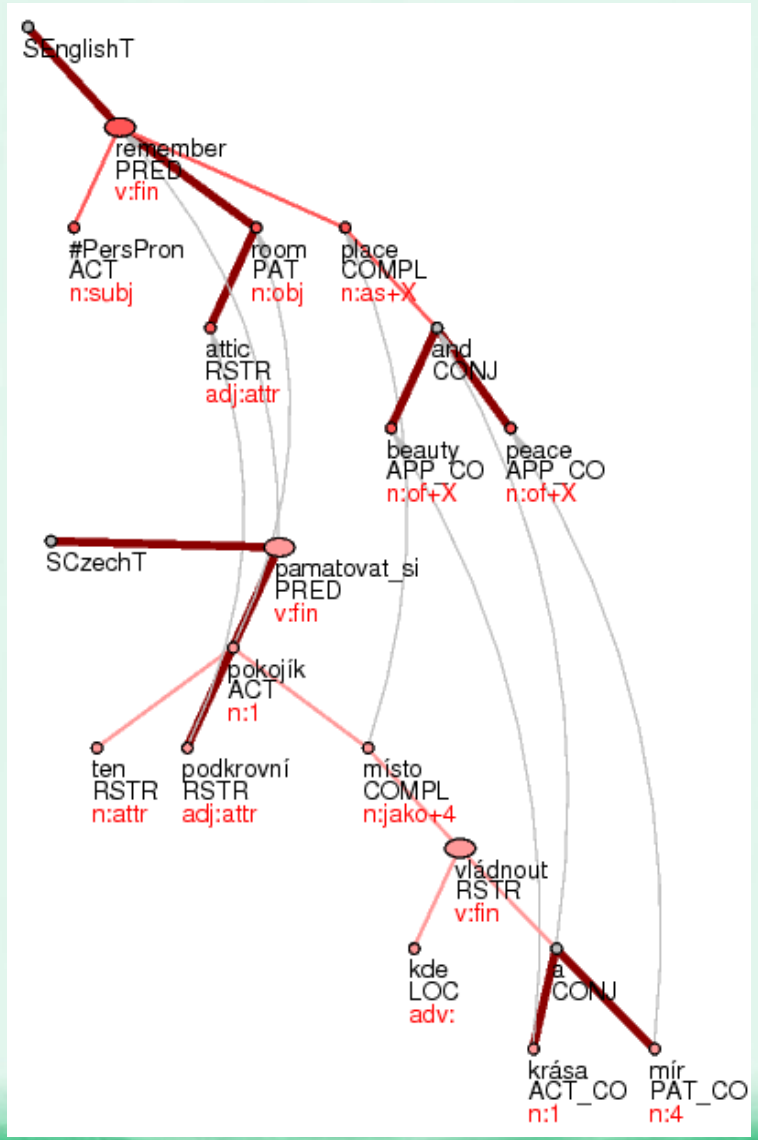
word alignment on the morphological layer



TrEd visualization

Treex

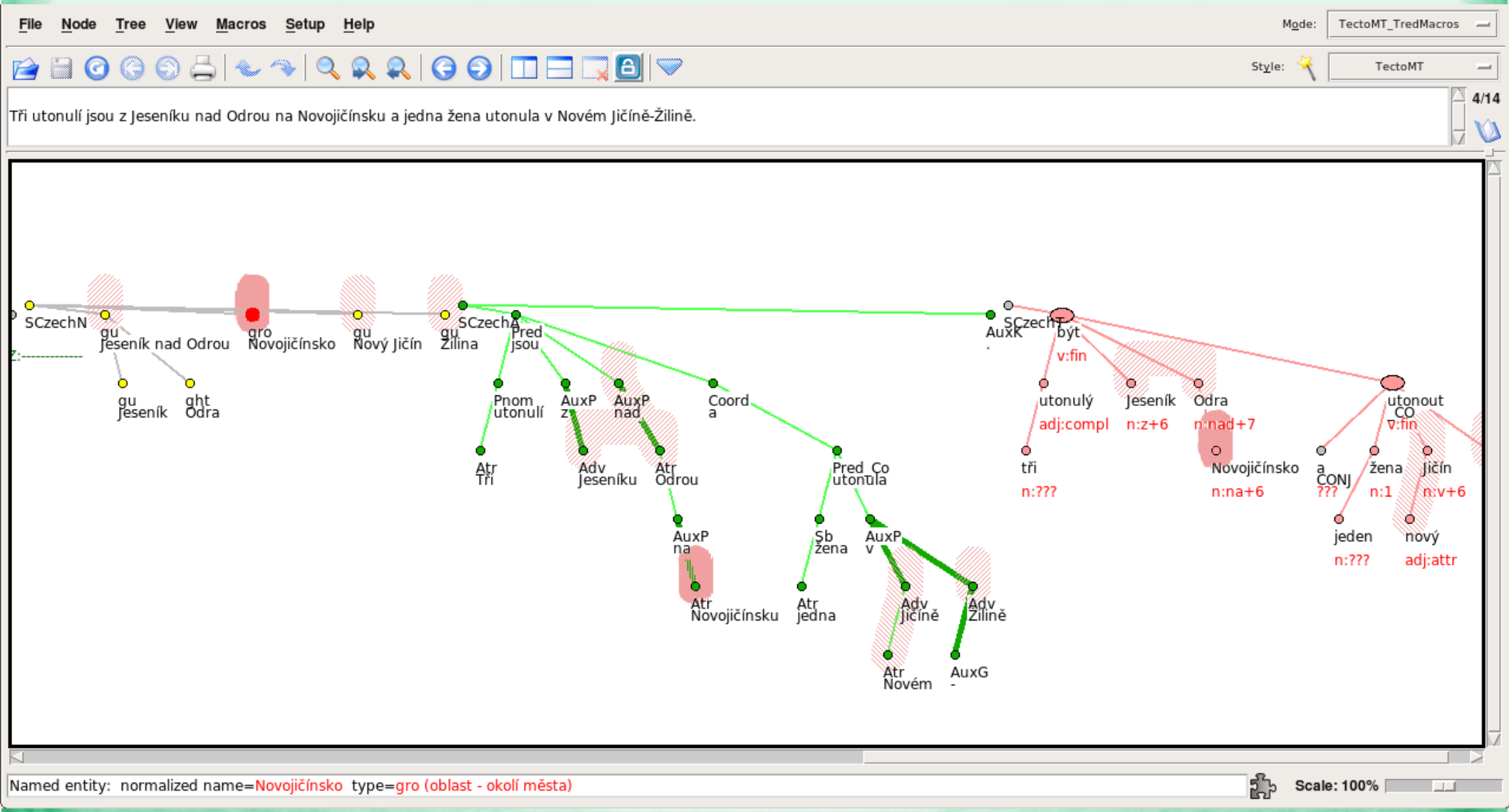
word alignment on the tectogrammatical layer



TrEd visualization

Treex

named entities



Block example – SVO to SOV code

Treex

Treex core
Treex convention
Perl keyword/convention

```
package Tutorial::SVO_to_SOV_solution;
use Moose;
extends 'Treex::Core::Block';

sub process_bundle {
  my ( $self, $bundle ) = @_;
  my $a_root = $bundle->get_tree('SEnglishA');

  foreach my $a_node ( $a_root->get_descendants() ) {
    if ( $a_node->get_attr('m/tag') =~ /^V/ ) {       # verb found
      foreach my $child ( $a_node->get_echildren() ) {
        if ( $child->get_attr('afun') eq 'Obj' ) {    # object found
          # Move the object and its subtree so it precedes the verb
          $child->shift_before_node($a_node);
        }
      }
    }
  }
}
1;
```

Thank you

Treex

Cooperation is welcomed.

http://ufal.mff.cuni.cz/tectomt