

Treeex NLP Framework

Martin Popel

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



JHU/CLSP Fred Jelinek Memorial PIRE Workshop
July 11, 2014, Prague

Outline

Applications

Background

**multi-purpose
NLP framework**
Treeex

Architecture

Treebanks

Applications

lemmatization

tagging

parsing

deep parsing

**machine
translation**
TectoMT

**online
interface**
Treex::Web

named entity r.

coreference

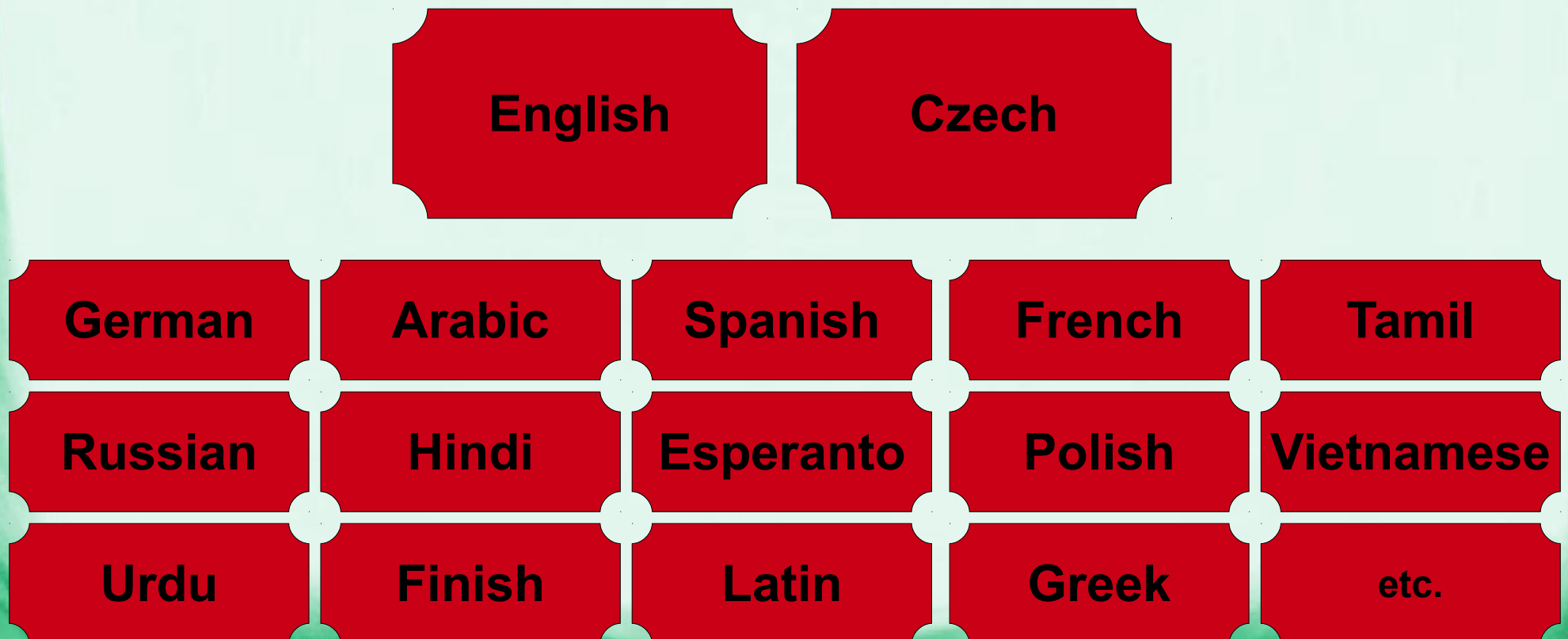
alignment

SMT preproc.

**multi-purpose
NLP framework**
Treex

Applications: analysis tools

lemmatization	base forms of words (linguistically adequate, finer than stemming)
tagging	part-of-speech tags, morphological analysis, disambiguation
parsing	dependency or constituency (phrase-based) trees
deep parsing	tectogrammatical trees, meaning of a sentence



Applications: TectoMT

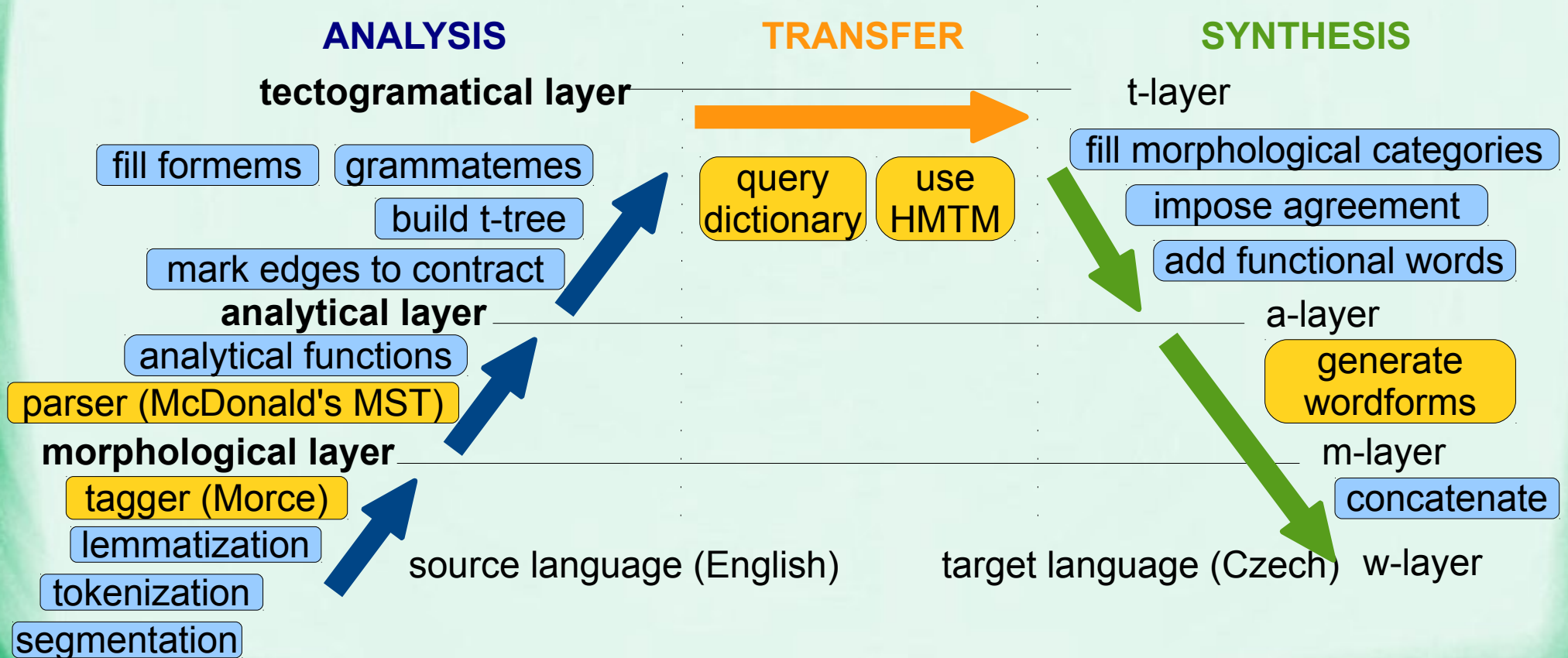


**machine
translation
*TectoMT***

- linguistically motivated machine translation
- developed since 2005 at ÚFAL by Zdeněk Žabokrtský et al.
- so far only English to Czech
- WMT 2012: evaluated as equally good as Moses system
- WMT 2013: in combination with Moses better than Google Translate
- speed: 0.8 seconds per sentence (after loading models)
- parallelization support (SGE with 100 cores, 68 M sentences, 10 days)

Applications: TectoMT

- deep-syntactic (tectogrammatical) transfer
- translation process divided to more than 140 “blocks”
- combining **statistical** and **rule based** blocks



Applications: TectoMT



MORPHOLOGY:

ResegmentSentences

Tokenize

NormalizeForms

FixTokenization

TagMorce

FixTags

Lemmatize

NAMED ENTITIES:

StanfordNamedEntities

DistinguishPersonalNames

A-LAYER:

MarkChunks

ParseMST

SetIsMemberFromDeprel

RehangConllToPdtStyle

FixNominalGroups

FixIsMember

FixAtree

FixMultiwordPrepAndConj

FixDicendiVerbs

SetAfunAuxCPCoord

SetAfun

T-LAYER:

MarkEdgesToCollapse

MarkEdgesToCollapseNeg

BuildTtree

SetIsMember

MoveAuxFromCoordToMembers

FixTlemmas

SetCoapFunctors

FixEitherOr

FixIsMember

MarkClauseHeads

MarkPassive

SetFunctors

MarkInfin

MarkRelClauseHeads

MarkRelClauseCoref

MarkDspRoot

MarkParentheses

SetNodetype

SetGrammatemes

SetFormeme

RehangSharedAttr

SetVoice

FixImperatives

SetIsNameOfPerson

SetGenderOfPerson

AddCorAct

FindTextCoref

TRANSFER:

CopyTtree

TrLFPPhrases

TrLFJointStatic

DeleteSuperfluousTnodes

TrFTryRules

TrFAddVariants

TrFRerank

TrLTryRules

TrLAddVariants

TrLFNumeralsByRules

TrLFilterAspect

TransformPassiveConstructions

PrunePersonalNameVariants

RemoveUnpassivizableVariants

TrLFCompounds

CutVariants

RehangToEffParents

TrLFTreeViterbi

RehangToOrigParents

CutVariants

FixTransferChoices

ReplaceVerbWithAdj

DeletePossPronBeforeVlastni

TrLFemaleSurnames

AddNounGender

MarkNewRelClauses

AddRelpronBelowRc

ChangeCorToPersPron

AddPersPronBelowVfin

AddVerbAspect

FixDateTime

FixGrammatemesAfterTransfer

FixNegation

MoveAdjsBeforeNouns

MoveGenitivesRight

MoveRelClauseRight

MoveDicendiCloserToDsp

MovePersPronNextToVerb

MoveEnoughBeforeAdj

MoveJesteBeforeVerb

FixMoney

OverridePpWithPhraseTr

FindGramCorefForRefIPron

NeutPersPronGenderFromAntec

ValencyRelatedRules

SetClauseNumber

TurnTextCorefToGramCoref

SYNTHESIS TO A-LAYER:

CopyTtree

DistinguishHomonymous.

ReverseNumberNounDep.

InitMorphcat

FixPossessiveAdjs

MarkSubject

ImposePronZAgr

ImposeRelPronAgr

ImposeSubjpredAgr

ImposeAttrAgr

ImposeComplAgr

DropSubjPersProns

AddPrepos

AddSubconjs

AddReflexParticles

AddAuxVerbCompoundPassive

AddAuxVerbModal

AddAuxVerbCompoundFuture

AddAuxVerbConditional

AddAuxVerbCompoundPast

AddClausalExpletivePronouns

ResolveVerbs

ProjectClauseNumber

AddParentheses

AddSentFinalPunct

AddSubordClausePunct

AddCoordPunct

AddAppositionPunct

ChooseMlemmaForPersPron

GenerateWordforms

MoveCliticsToWackernagel

DeleteSuperfluousPrepos

DeleteEmptyNouns

VocalizePrepos

CapitalizeSentStart

CapitalizeNamedEntities.

FillTagFromMorphcat

SYNTHESIS TO TEXT:

ConcatenateTokens

ApplySubstitutions

DetokenizeUsingRules

RemoveRepeatedTokens

NormalizePunctuationForWMT

Applications: Treex::Web



online interface

quest.ms.mff.cuni.cz/treex-web/result/180u77gRxPyo6x0oKYx

Treex::Web Home Run Treex Results Scenarios Logout

Treex Result

[« Go back to all results](#)

[Download result](#) [Download all](#) [Run again](#)

Previous 1 2 3 4 5 6 7 Next

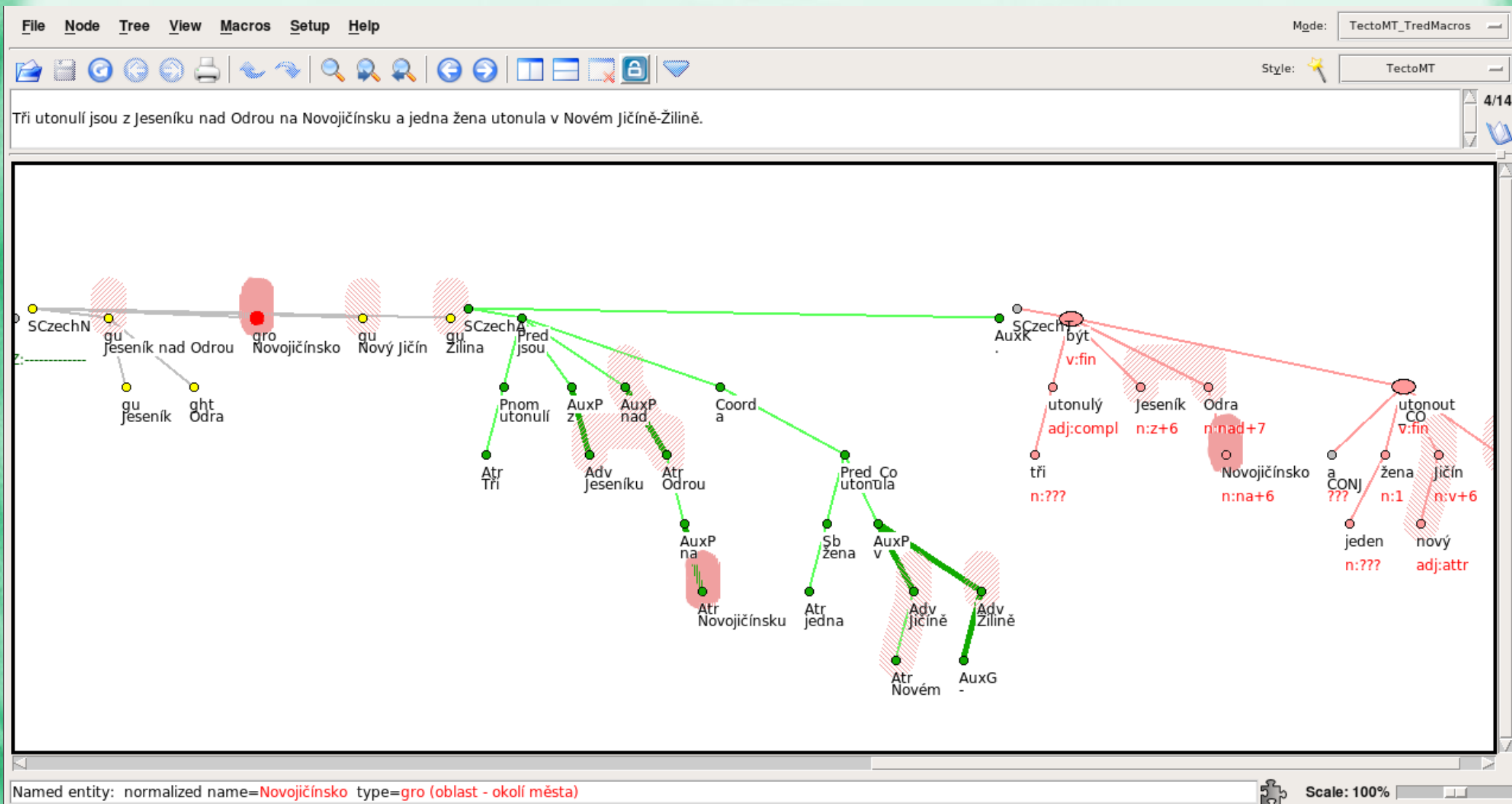
[cs] Český jazyk neboli čeština je západoslovanský jazyk, nejvíce příbuzný se slovenštinou, poté polštinou a lužickou srbštinou.

The image displays two syntactic trees for the sentence: "Český jazyk neboli čeština je západoslovanský jazyk, nejvíce příbuzný se slovenštinou, poté polštinou a lužickou srbštinou." The left tree is an 'a-tree' (zone=cs) and the right tree is a 't-tree' (zone=cs). Both trees show the hierarchical structure of the sentence, with nodes representing grammatical functions and parts of speech. The 'a-tree' is more detailed, showing the internal structure of the noun phrase 'se slovenštinou, poté polštinou a lužickou srbštinou'.

Applications: NER

named entity recognition

- Support for nested named entities (“Jeseník nad Odrou”: city vs. river)
- Hierarchical classes (g=geographical entity, gr=region, gu=city, gs=street,...)



Background

**PDT layers
of
annotation**

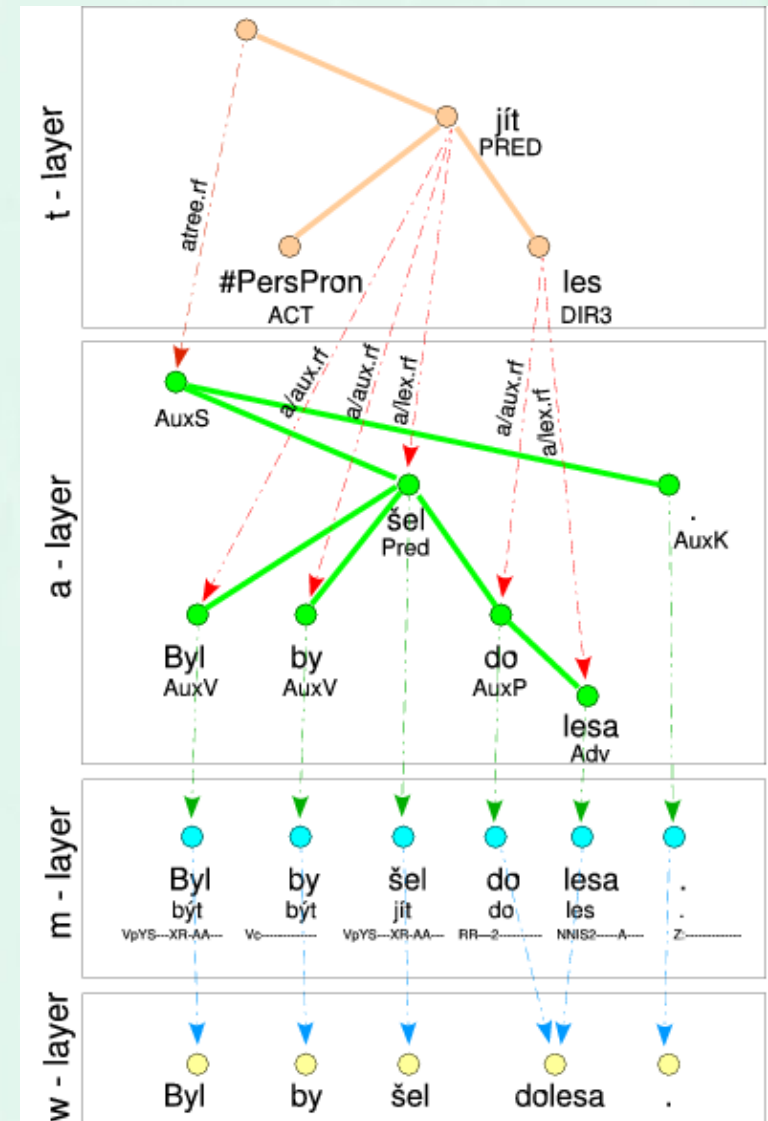
**multi-purpose
NLP framework**
Treex

**Prague
Markup
Language**

PDT layers of language description

implemented in Prague Dependency Treebank

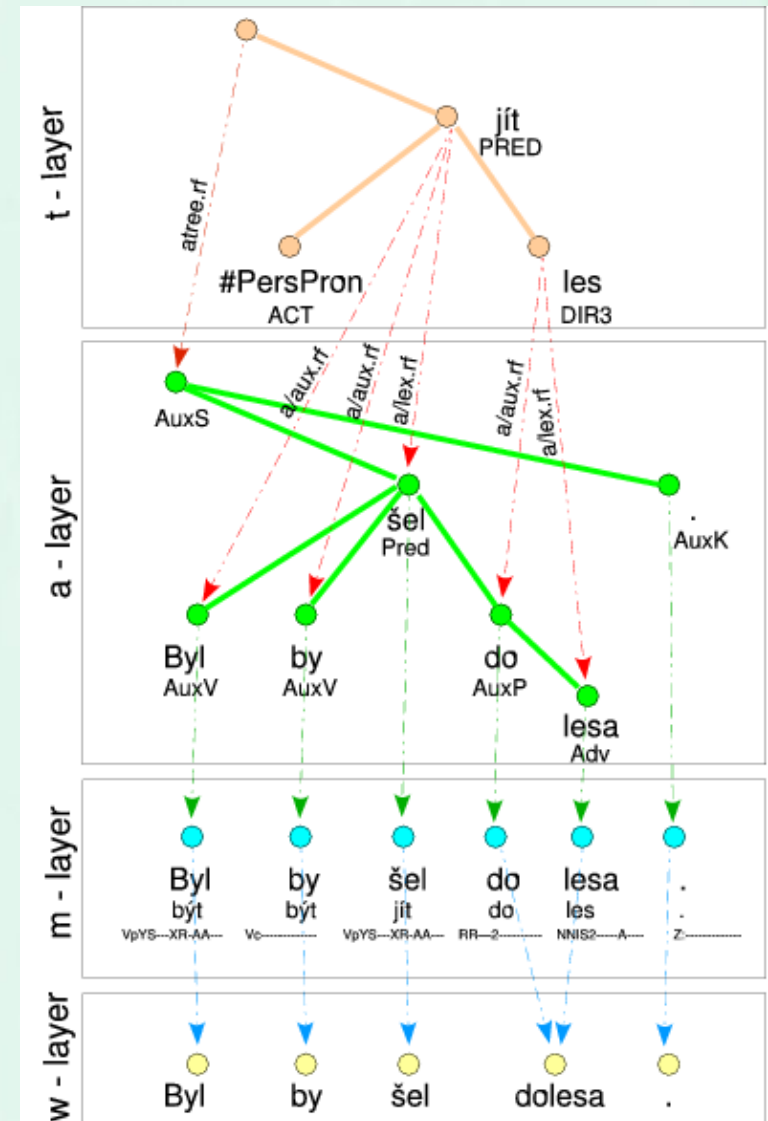
- **tectogrammatical layer**
deep-syntactic dependency trees
- **analytical layer**
surface-syntactic dependency trees, labeled edges
- **morphological layer**
lemma & POS tag for each word
- **word layer**
raw (tokenized) text



PDT layers of language description

implemented in Prague Dependency Treebank

- **tectogrammatical layer**
deep-syntactic dependency trees
 - abstraction from many language-specific phenomena
 - autosemantic (meaningful) words
~ **nodes**
 - functional words (prepositions, auxiliaries)
~ **attributes**
 - syntactic-semantic relations (dependencies)
~ **edges**
 - added nodes (e.g. because of pro-drop)
 - ...



layers of language description

implemented in Treex

- Mostly backward compatible adaptations (adding attributes)
 - **formeme** (n:2, n:k+3, v:že+vfin, v:rc, adj:attr)
 - attributes for clauses, is_passive ($\rightarrow$ diathesis),...
- All layers stored in **one file**
- A-layer and m-layer merged into one
- Two more layers:
 - P-layer phrase-structure trees
 - N-layer named entities

Prague Markup Language

- developed since 2005, used for PDT 2.0 (2006)
- many similarities with NIF: universal NLP,...
- XML-based (no RDF), PML Schema ($\rightarrow$ Relax NG)
- special support for trees and roles
- one schema for each layer of PDT, one for Treex
- used in **TrEd** (tree editor) and **PML-TQ** (tree query)
- much less popular than simple CoNLL format
- too universal, too complex, only Perl implementation
- not enough advertising?
- standard: format vs. API?

Treebanks

**multi-purpose
NLP framework**
Treeex

**Czech-English
deep treebank**
CzEng

**Czech-English
deep treebank**
PCEDT

**treebanks for
30 languages**
HamleDT



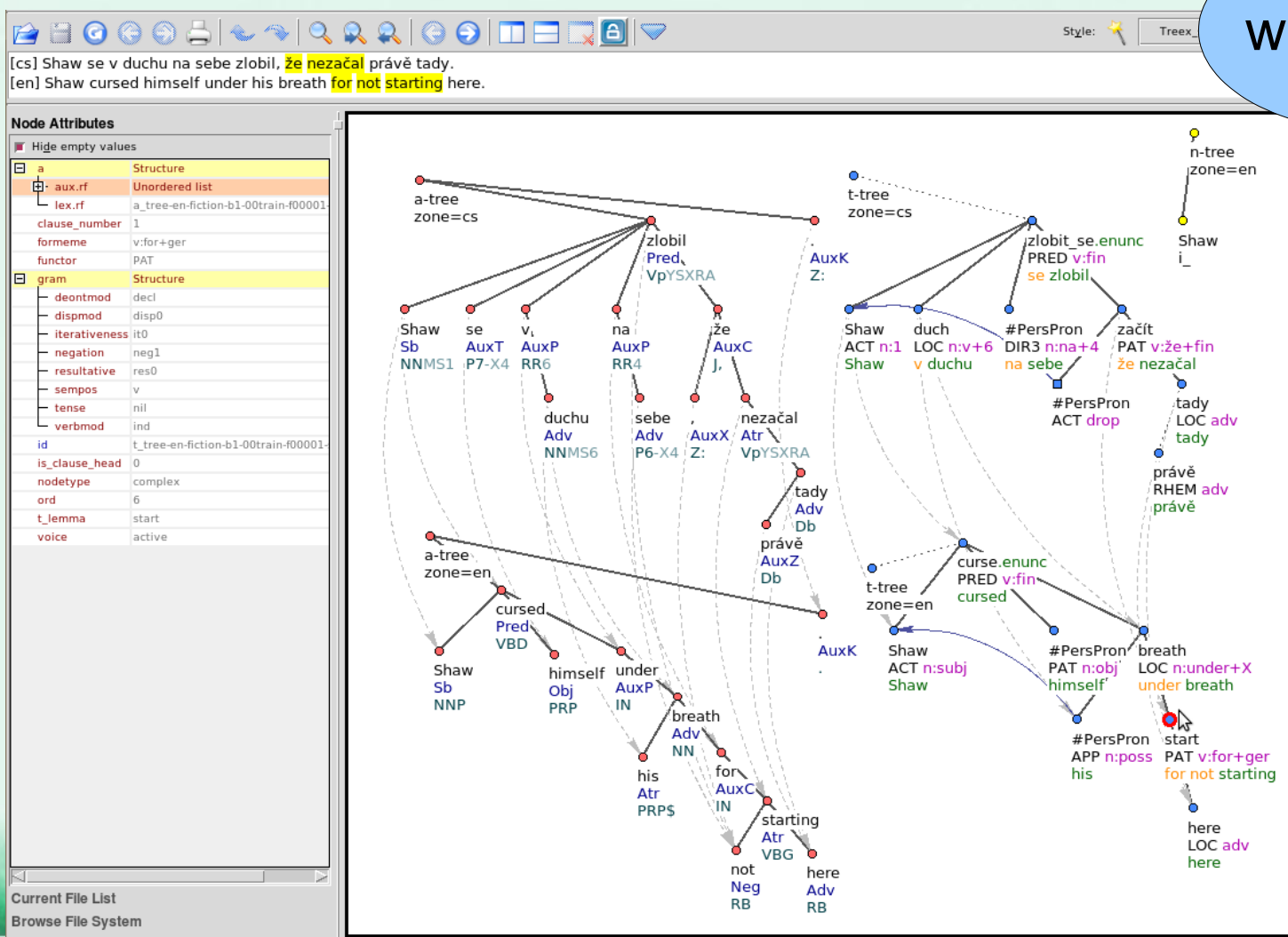
CzEng

(Czech-English treebank)



- automatically parsed deep parallel treebank
- 15 M sentences; 200 M tokens

with Treex

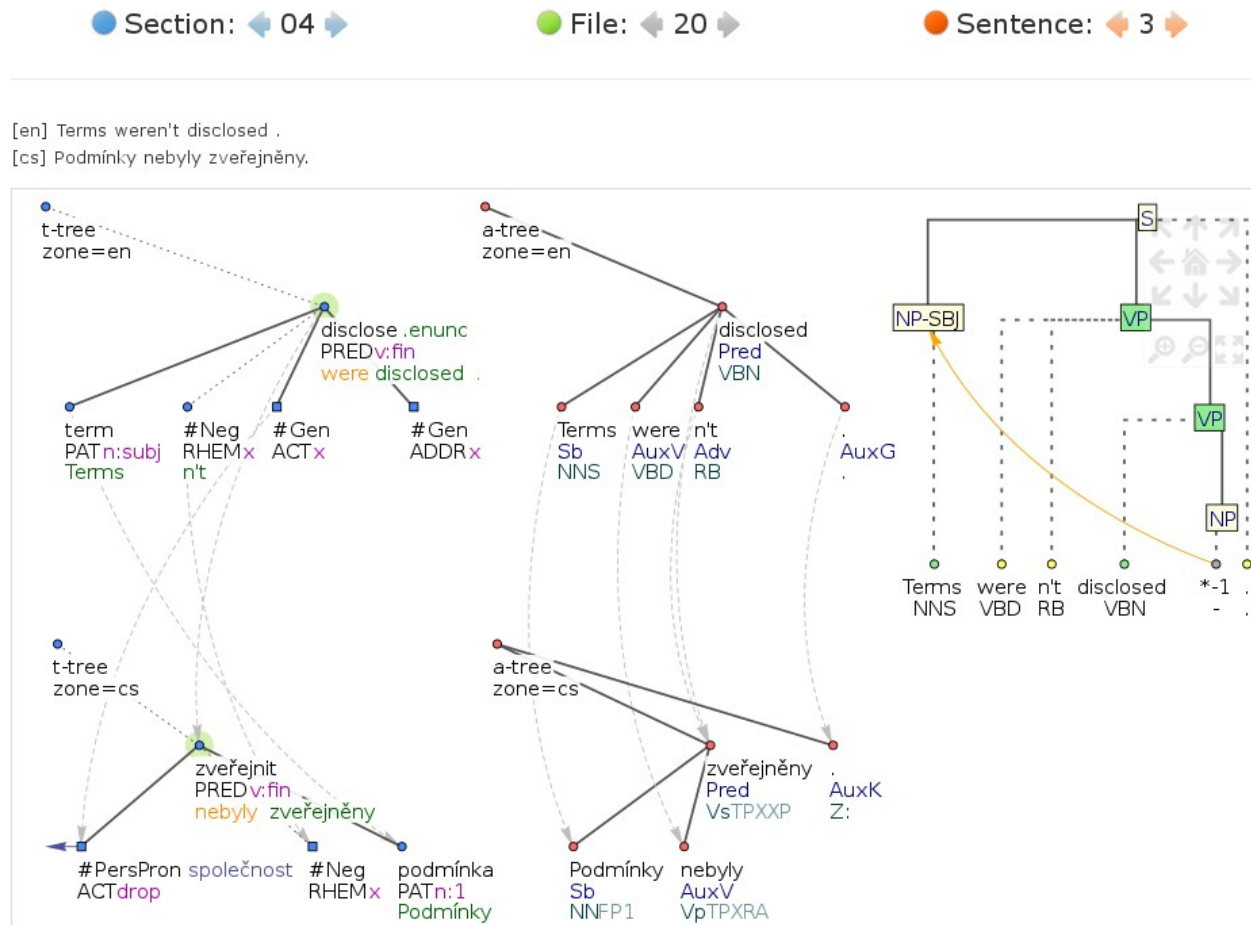


PCEDT

(Prague Czech-English Dependency Treebank)

- manually parsed deep parallel treebank
- 50 k sentences; 1.2 M tokens

pre-annotated
with Treex

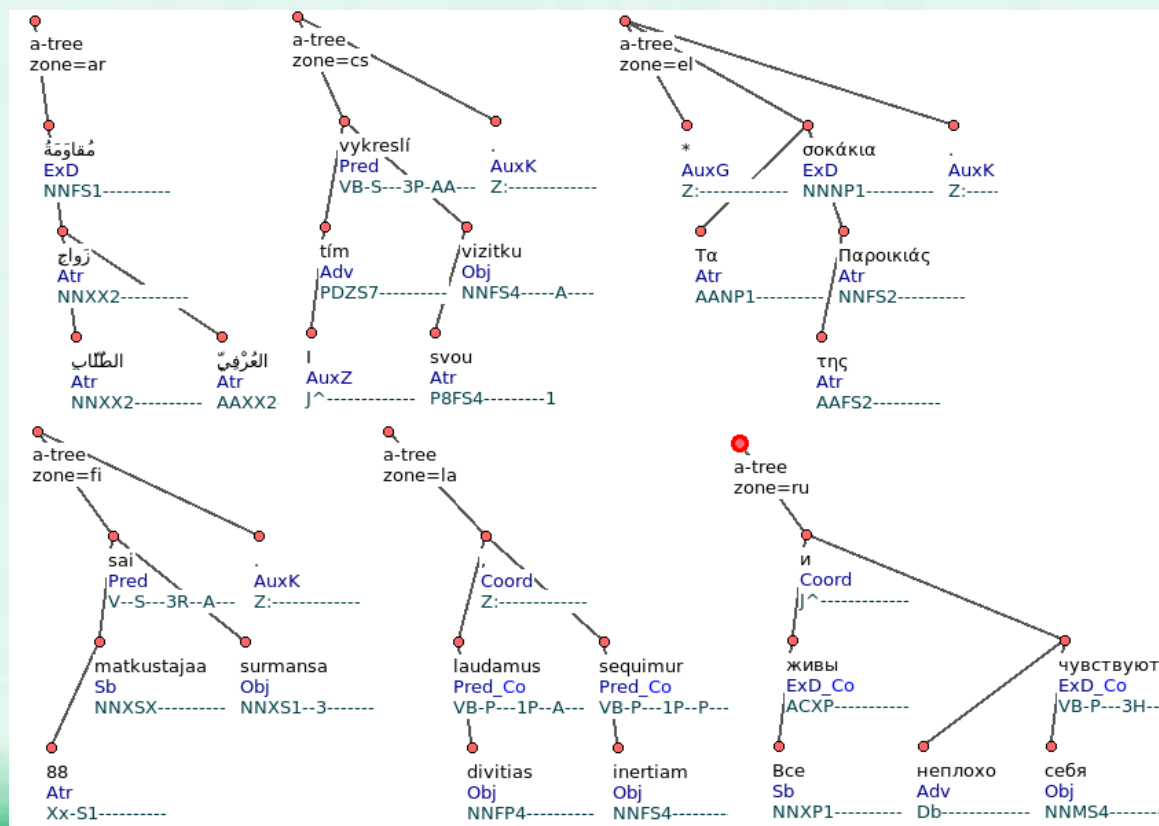


HamleDT



- 30 manually annotated dependency treebanks
- 30 languages; 400 k sentences; 6 M tokens
- harmonized into a common style (guidelines)

using
Treeex



Treex architecture

**multi-purpose
NLP framework**
Treex

motivation

architecture

internals

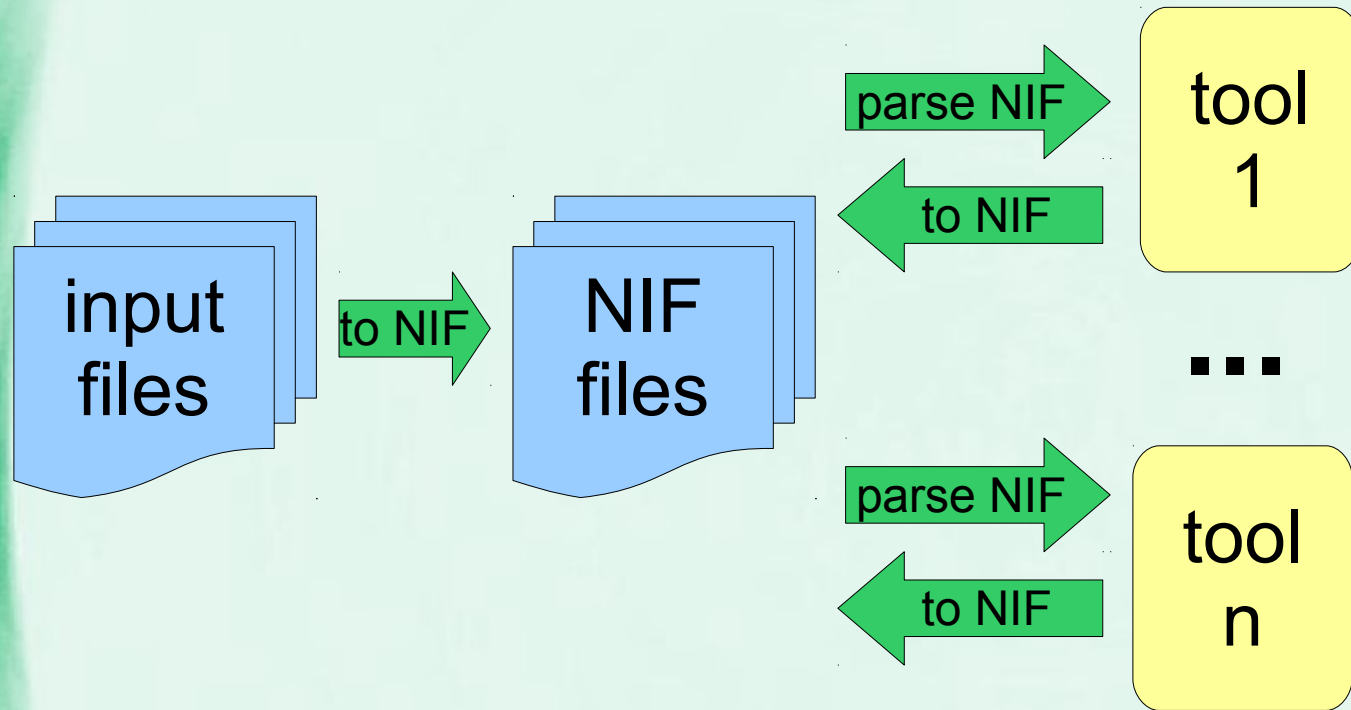
examples

Motivation

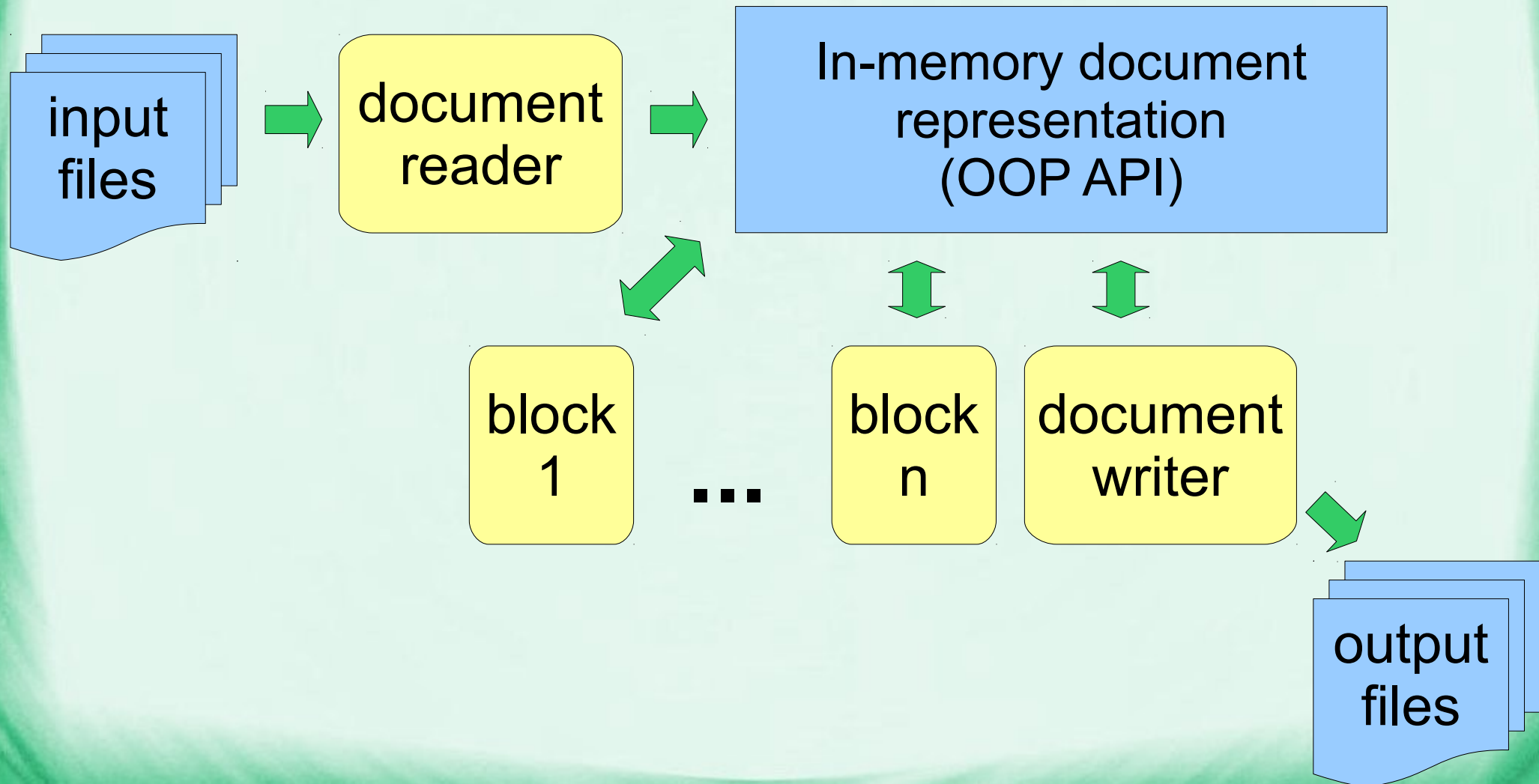
Goals of Treex

- elegant integration of in-house and third-party NLP tools
- modularity, reusability, cooperation, efficient development
- ability to easily modify and add code in a full-fledged programming language (Perl)
- unified object-oriented interface for accessing data structures

My idea about NIF pipeline



Treex architecture



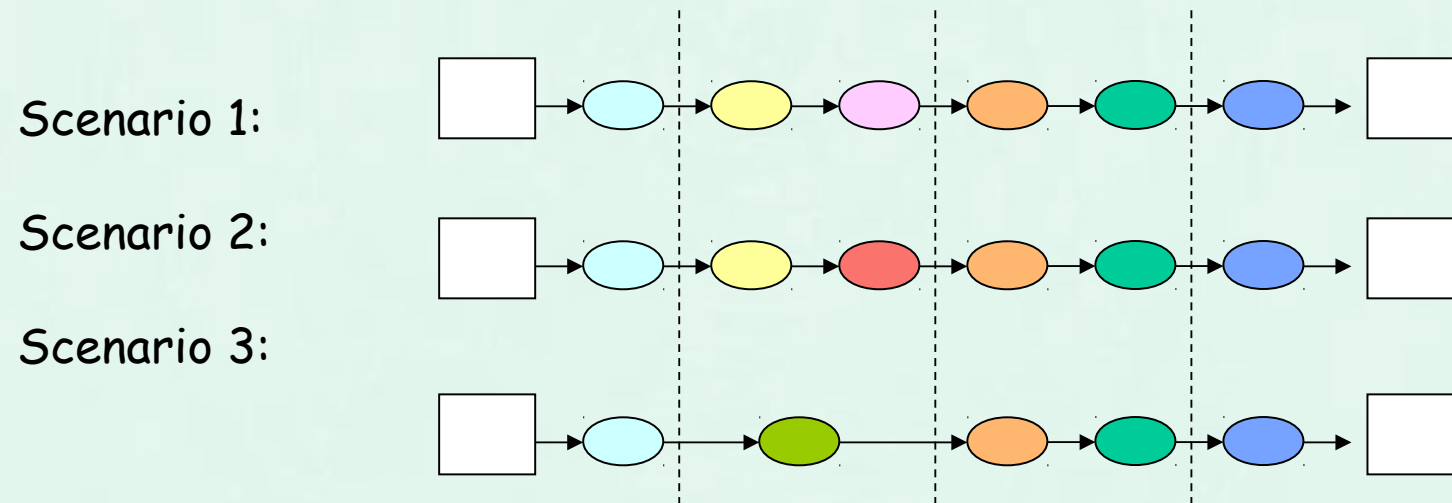
Treex architecture processing units

- **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
 - described by a scenario that
 - starts with a **reader** (input conversion)
 - ends with a **writer** (output conversion)
 - Readers can split the input file into more in-memory docs.
 - There are readers&writers for a number of popular formats: plain text, CoNLL, PDT PML, Penn MRG, Tiger...

***.treex.gz**

Treex architecture processing units

Blocks can be easily substituted with an alternative solution.



Treeex architecture processing units

Blocks can be easily substituted with an alternative solution.

Scenario A

W2A::EN::Segment

W2A::EN::Tokenize

W2A::EN::TagMorce

W2A::EN::Lemmatize

W2A::EN::ParseMST

Scenario B

W2A::SegmentOnNewLines

W2A::EN::TagLinguaEn

W2A::EN::Lemmatize

W2A::EN::ParseMalt

Treeex architecture

data units

- **Document**
 - stored in one file
 - sequence of sentences
- **Bundle** (“bundle of trees”)
 - corresponds to one sentence
- **Zone**
 - one for each language (Arabic, Czech, English,...)
 - and optionally a variant (“selectors” src, trg, ref,...)
- **Tree**
 - layer of language description: A, T (plus P, N)
 - m-layer is stored with the a-layer in one tree

Treex architecture data units

DOCUMENT

sentence 1

BUNDLE

Zone en_src

W-layer

Peter does not love Mary.

M-layer

● ● ● ● ●
Peter do not love Mary
NNP VBZ RB VBD NNP

A-layer

● ● ● ● ●
Peter do not love Mary
Sb AuxV Neg Pred Obj

T-layer

● ● ●
Peter love Mary
ACT PRED PAT

Zone cs_src

W-layer

Petr nemiluje Marii.

M-layer

● ● ●
Petr milovat Marie
NNMS1 VB-S—3P-NA NNFS4

A-layer

● ● ●
Petr milovat Marie
Sb Pred Obj

T-layer

● ● ●
Petr milovat Marie
ACT PRED PAT

sentence 2

BUNDLE

...

sentence N

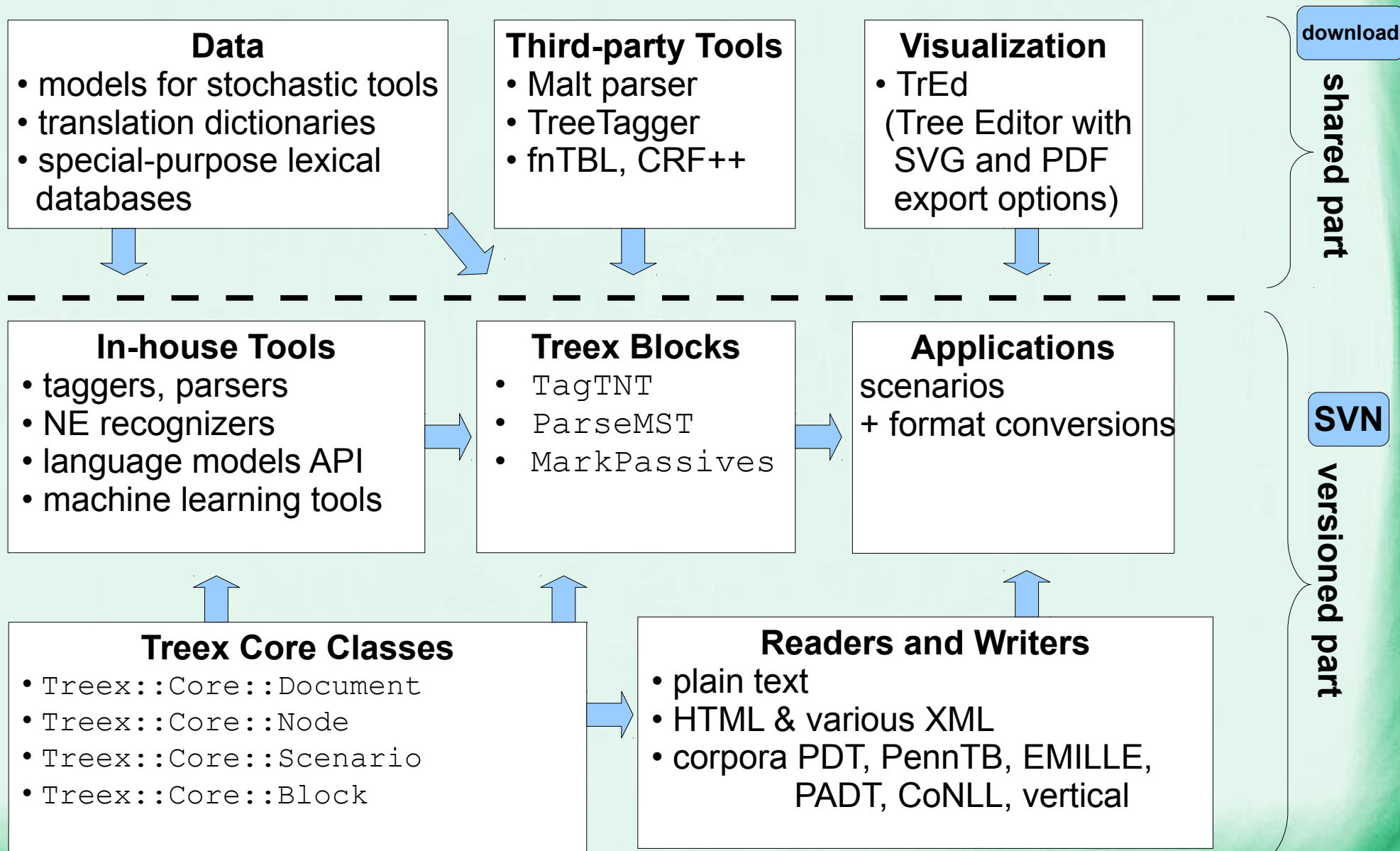
BUNDLE

...

Internals – Design decisions

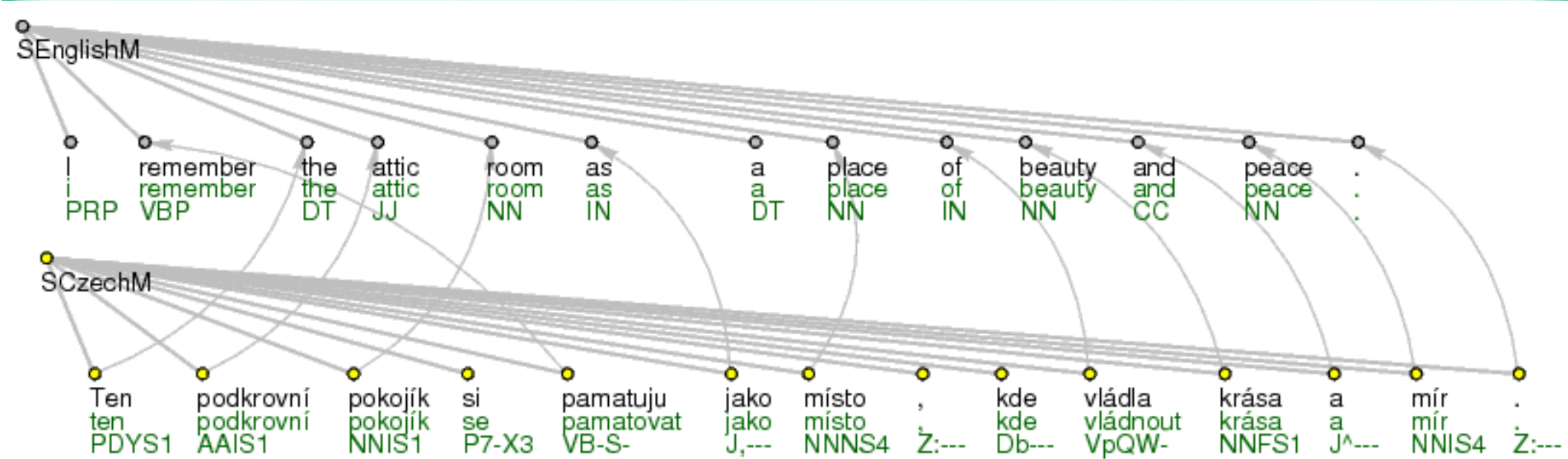
- Perl (wrappers for binaries, Java,...)
- Linux (some applications platform-independent)
- OOP (Moose)
- Open source (GNU GPL for the versioned part)
- Neutral w.r.t. methodology (statistical, rule-based)
- Multilingual
- Open standards (Unicode, XML)

Internals – Components



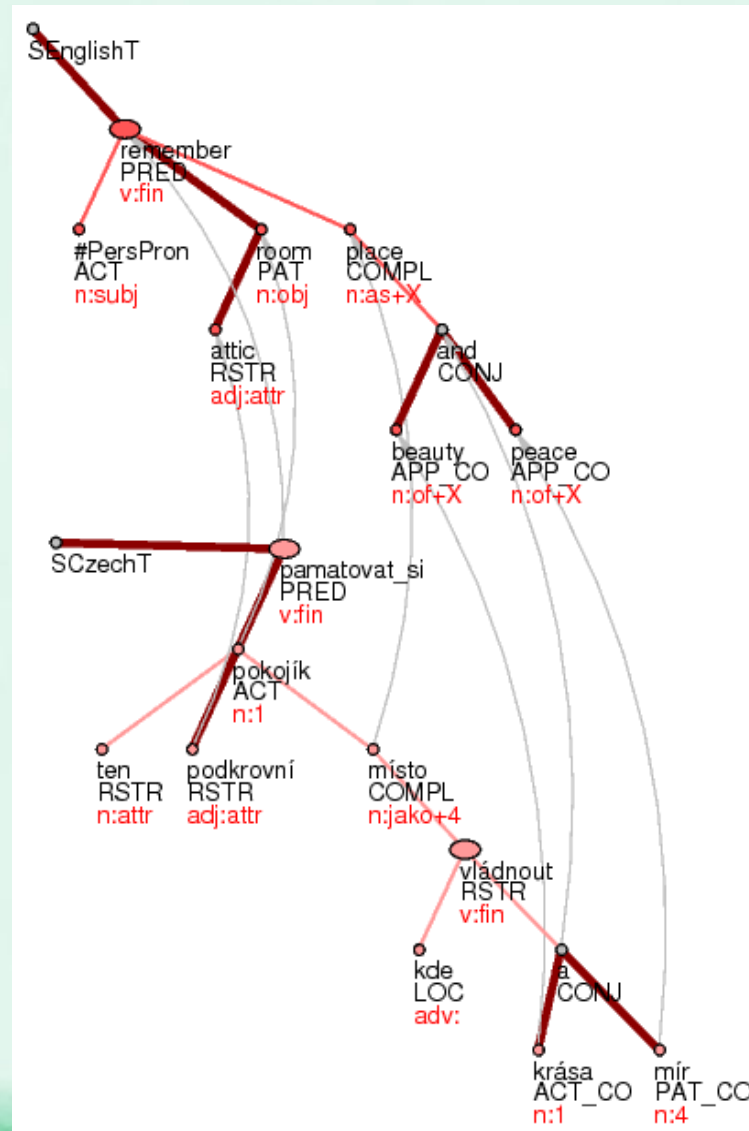
TrEd visualization

word alignment on the morphological layer



TrEd visualization

word alignment on the tectogrammatical layer



Block example – SVO to SOV code

```
package Tutorial::Svo2SovSolution;  
use Moose;  
use Treex::Core::Common;  
extends 'Treex::Core::Block';
```

Treex core
Treex convention
Perl keyword/convention

```
sub process_anode {  
    my ( $self, $a_node ) = @_  
    if ( $a_node->tag =~ /^V/ ) {          # verb found  
        foreach my $child ( $a_node->get_echildren() ) {  
            if ( $child->afun eq 'Obj' ) {   # object found  
                # Move the object and its subtree so it precedes the verb  
                $child->shift_before_node($a_node);  
            }  
        }  
    }  
    return;  
}  
1;
```

John loves Mary more than cherry.
SUBJ VERB OBJECT

John Mary loves more than cherry.
SUBJ OBJECT VERB

SVO → SOV

Thank you

Cooperation is welcomed.



<http://ufal.mff.cuni.cz/treex>

References

<http://ufal.mff.cuni.cz/treex>

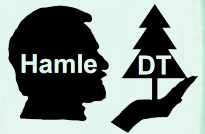


<http://quest.ms.mff.cuni.cz/treex-web>

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



<http://ufal.mff.cuni.cz/hamledt>



<http://ufal.mff.cuni.cz/czeng>



<http://ufal.mff.cuni.cz/pcedt2.0>

<http://ufal.mff.cuni.cz/jazz/PML>

<http://euler.ms.mff.cuni.cz/>

PML-TQ