Searching in Discourse-Annotated Treebanks

Jiří Mírovský

Charles University

Institute of Formal and Applied Linguistics

Prague, November 12, 2018

Searching in Discourse-Annotated Treebanks

outline

- Prague Dependency Treebank (PDT) & Discourse relations
- Prague Markup Language (PML)
- PML-Tree Query
- PDT and PML-Tree Query
- PDTB and PML-Tree Query

Searching in Discourse-Annotated Treebanks

outline

- Prague Dependency Treebank (PDT) & Discourse relations
- Prague Markup Language (PML)
- PML-Tree Query
- PDT and PML-Tree Query
- PDTB and PML-Tree Query

PDT

Prague **D**ependency **T**reebank

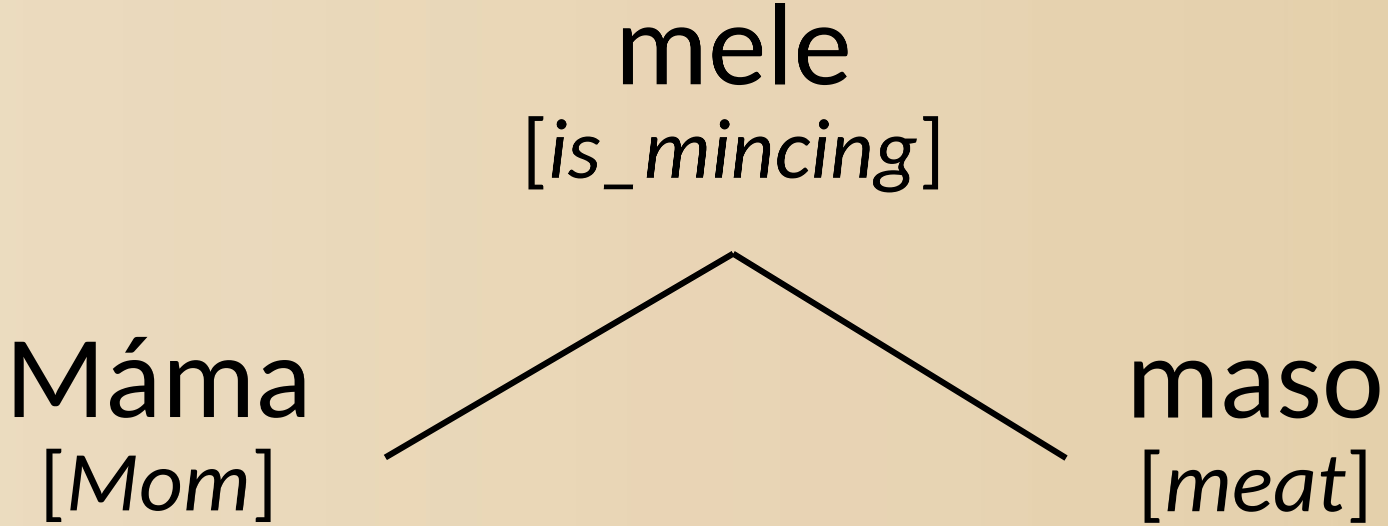
- **Czech journalistic texts** from 1990's
- **50 thousand** sentences annotated manually on **several layers**
  - morfological layer (part of speech, case, ...)
  - analytical layer (surface syntax)
  - tectogrammatical layer (deep syntax)

Prague Dependency Treebank

versions and availability

- **PDT 1.0** – published in 2001 (LDC)
- **PDT 2.0** – published in 2006 (LDC)
  - tectogrammatical layer in large scale
- **PDT 2.5** – published in 2011 (Lindat/Clarin, Creative Commons License)
  - multiword expressions (“named entities”)
- **PDiT 1.0** – published in 2012 (Lindat/Clarin, ...)
  - discourse relations, bridging anaphora, extended textual coreference
- **PDT 3.0** – published in 2013 (Lindat/Clarin, ...)
- **PDiT 2.0** – published in 2016 (Lindat/Clarin, ...)
  - secondary discourse connectives, further extended coreference
- **PDT 3.5** – published in 2018 (Lindat/Clarin, ...)

PDT – Analytical layer



mele
[*is_mincing*]

Máma
[*Mom*]

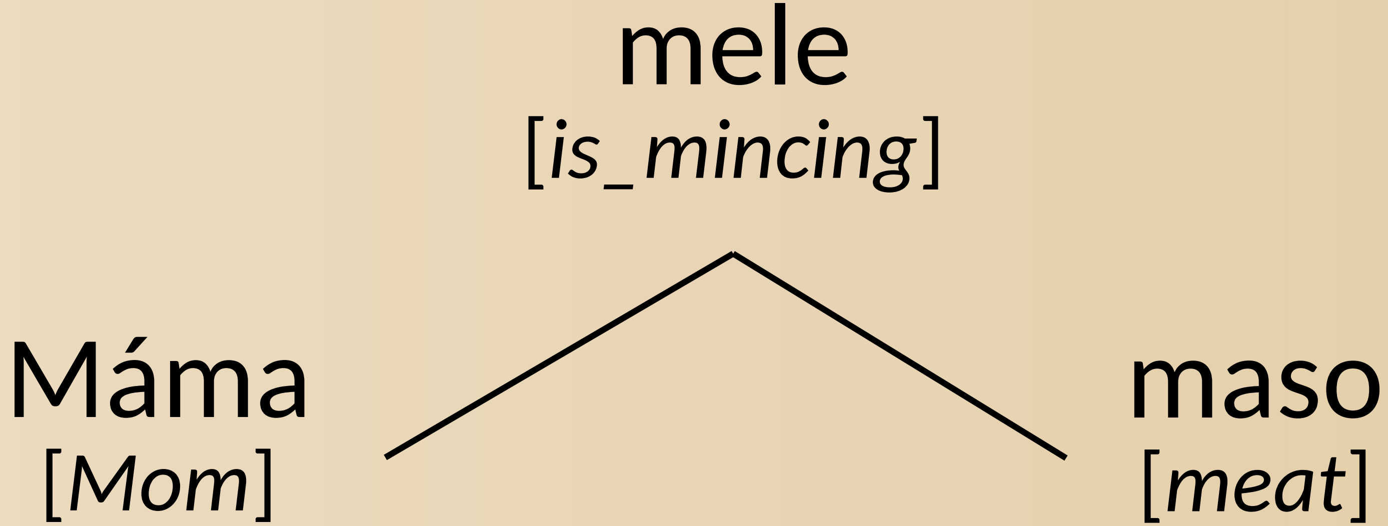
maso
[*meat*]

Máma [*Mom*] – Subject

mele [*is_mincing*] – Predicate

maso [*meat*] – Object

PDT – Tectogrammatical layer



mele

[*is_mincing*]

Máma

[*Mom*]

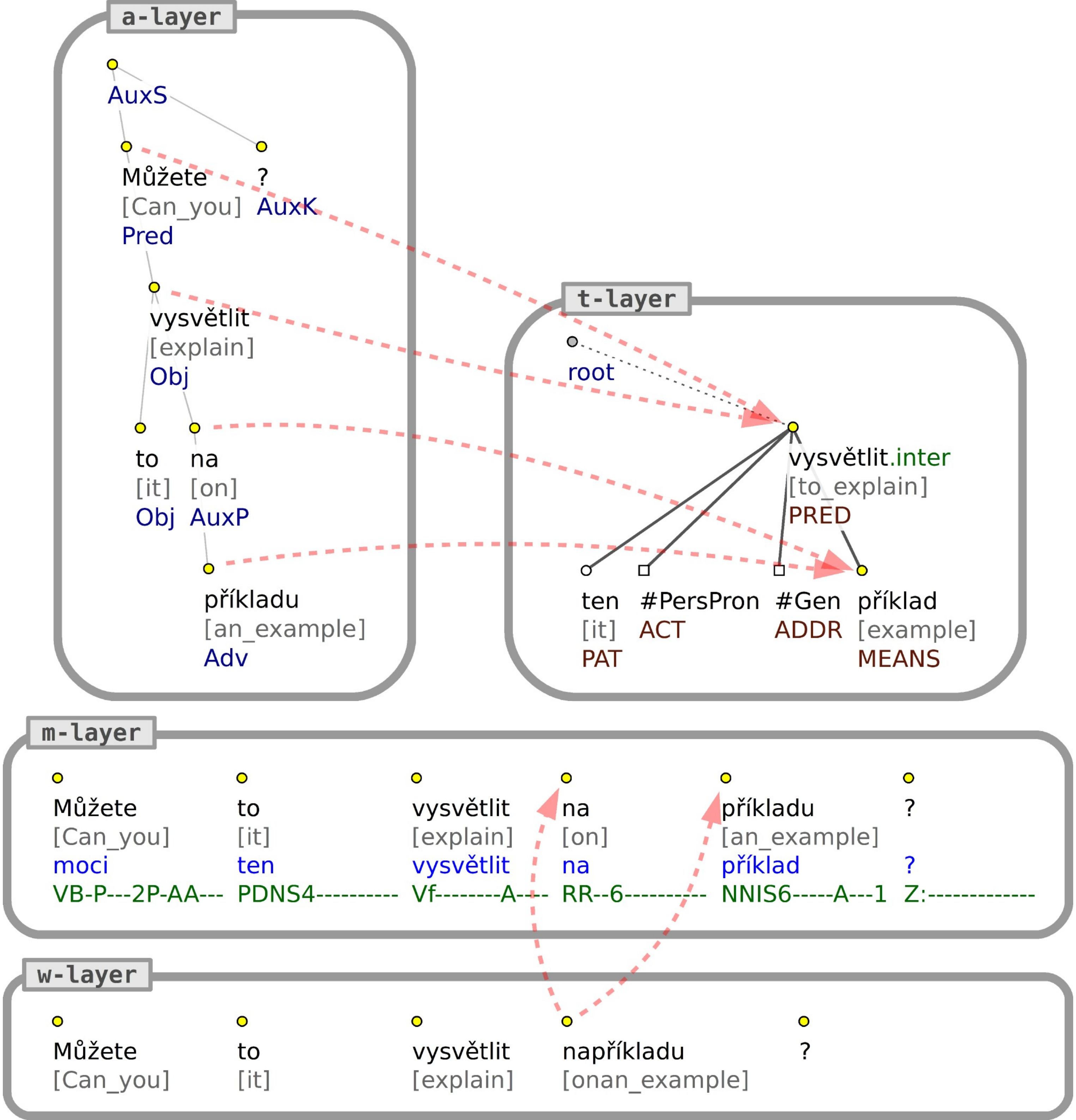
maso

[*meat*]

Máma [*Mom*] – Actor

mele [*is_mincing*] – Predicate

maso [*meat*] – Patiens

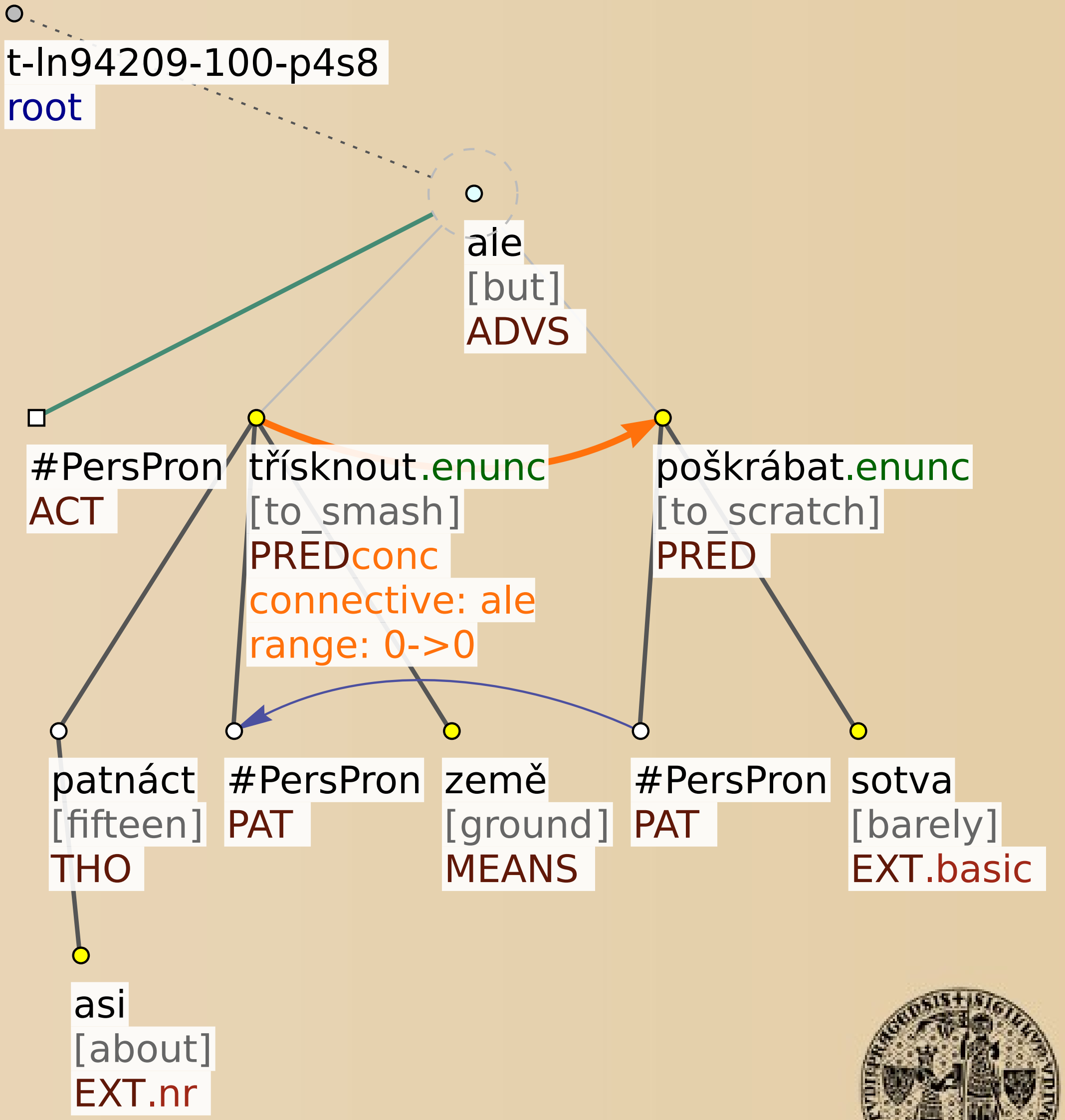

a-layer
AuxS
Můžete
[Can_you]
Pred
?
AuxK
vysvětlit
[explain]
Obj
to
[it]
Obj
na
[on]
AuxP
příkladu
[an_example]
Adv
t-layer
root
vysvětlit.inter
[to_explain]
PRED
ten
[it]
PAT
#PersPron
ACT
#Gen
ADDR
příklad
[example]
MEANS
m-layer
Můžete
[Can_you]
moci
VB-P---2P-AA---
to
[it]
ten
PDNS4----------
vysvětlit
[explain]
vysvětlit
Vf--------A----
na
[on]
na
RR--6----------
příkladu
[an_example]
příklad
NNIS6-----A---1
?
?
Z:-------------
w-layer
Můžete
[Can_you]
to
[it]
vysvětlit
[explain]
například
[onan_example]
?

Prague Discourse Treebank 1.0

An example

Asi patnáctkrát jsem jí třísknul o zem, ale **sotva jsem ji poškrábal**.

[Lit.: *I smashed it against the ground about fifteen times* but **I barely scratched it**.]

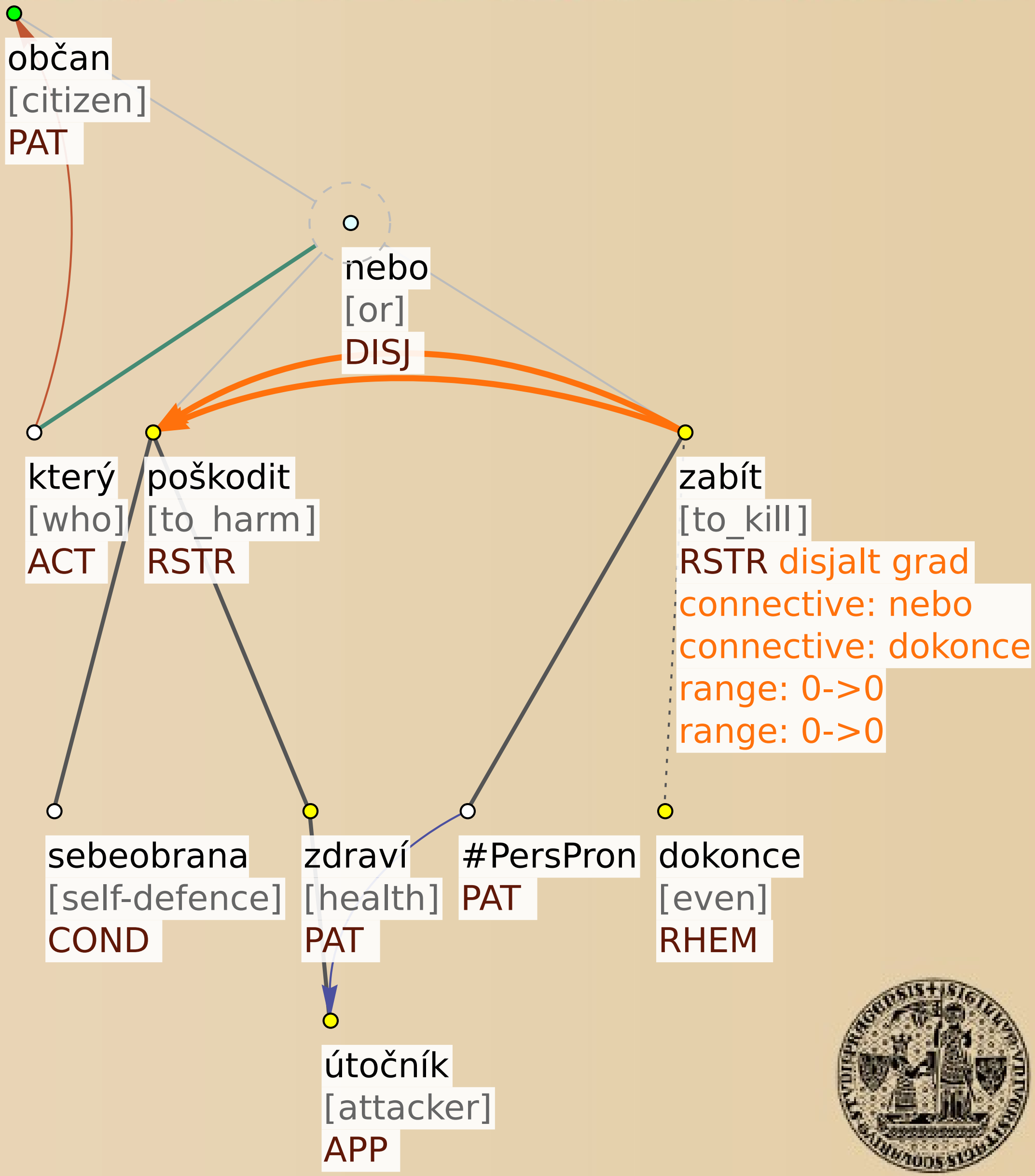


Prague Dependency Treebank 3.0

Second relations

Občané, *kteří v sebeobraně poškodili zdraví útočníka* nebo **ho** dokonce **zabili**, bývají za své jednání často nespravedlivě stíháni.

[Lit.: Citizens *who in self-defence harmed health of the attacker* or even **killed him**, are for their actions often unfairly prosecuted.]



Prague Dependency Treebank 3.0
(PDT **3.0**)

In the **whole** PDT 3.0 (**50 th.** sentences), there are

- **20,556** discourse relations
  - **6,226** inter-sentential
  - **14,330** intra-sentential

(plus **83** list structures)

- **95,302** relations of textual coreference
- **23,312** relations of grammatical coreference
- **34,367** bridging relations

Searching in Discourse-Annotated Treebanks

outline

- Prague Dependency Treebank (PDT) & Discourse relations
- Prague Markup Language (PML)
- PML-Tree Query
- PDT and PML-Tree Query
- PDTB and PML-Tree Query

PML

PML – **P**rague **M**arkup **L**anguage

PML is a **general** **XML-based** **format** for **all kinds** of linguistically annotated **treebanks**.

Prague Markup Language

three components for your data

- **PML-schema**
  - data structure
- **Stylesheet**
  - data appearence
- **Macros**
  - data manipulation

Prague Markup Language

PML-schema

Description of the **structure** of the data

- **types of nodes** in the data (root, node, terminal, non-terminal, …)
- **relations** among nodes (child relation between non-terminal → non-terminal, non-terminal → terminal, coreference, discourse relations, …)
- **names** and **types** (and special roles) **of attributes**
- **values** of enumerative attributes

Prague Markup Language

PML-schema

```
<type name="t-node.type"> <!-- simplified! -->
  <structure role="#NODE" name="t-node">
    <member as_attribute="1" name="id" role="#ID" required="1">
      <cdata format="ID"/>
    </member>
    <member name="is_generated" type="bool.type"/>
    <member name="t_lemma" required="1">
      <cdata format="any"/>
    </member>
    <member name="functor" required="1">
      <alt type="func.type"/>
    </member>
    <member name="deepord" role="#ORDER" required="1">
      <cdata format="nonNegativeInteger"/>
    </member>
    <member name="discourse" required="0">
      <list ordered="0" type="t-discourse-link.type"/>
    </member>
    ...
  </structure>
</type>
```

Prague Markup Language

PML-schema

```
<type name="t-discourse-link.type"> <!-- simplified! -->
  <structure>
    <member name="target_node.rf" required="0">
      <cdata format="PMLREF"/>
    </member>
    <member name="start_range" required="1"> ... </member>
    <member name="target_range" required="0"> ... </member>
    <member name="discourse_type"  type="t-discourse-type.type" required="0"/>
    <member name="a-connectors.rf" required="0">
      <list ordered="0"> <cdata format="PMLREF"/> </list>
    </member>
    <member name="t-connectors.rf" required="0">
      <list ordered="0"> <cdata format="PMLREF"/> </list>
    </member>
    <member name="connective" required="0"> <!-- for searching in PML-TQ only (not in the distributed data) -->
      <cdata format="any"/>
    </member>
    ...
  </structure>
</type>
```

Prague Markup Language

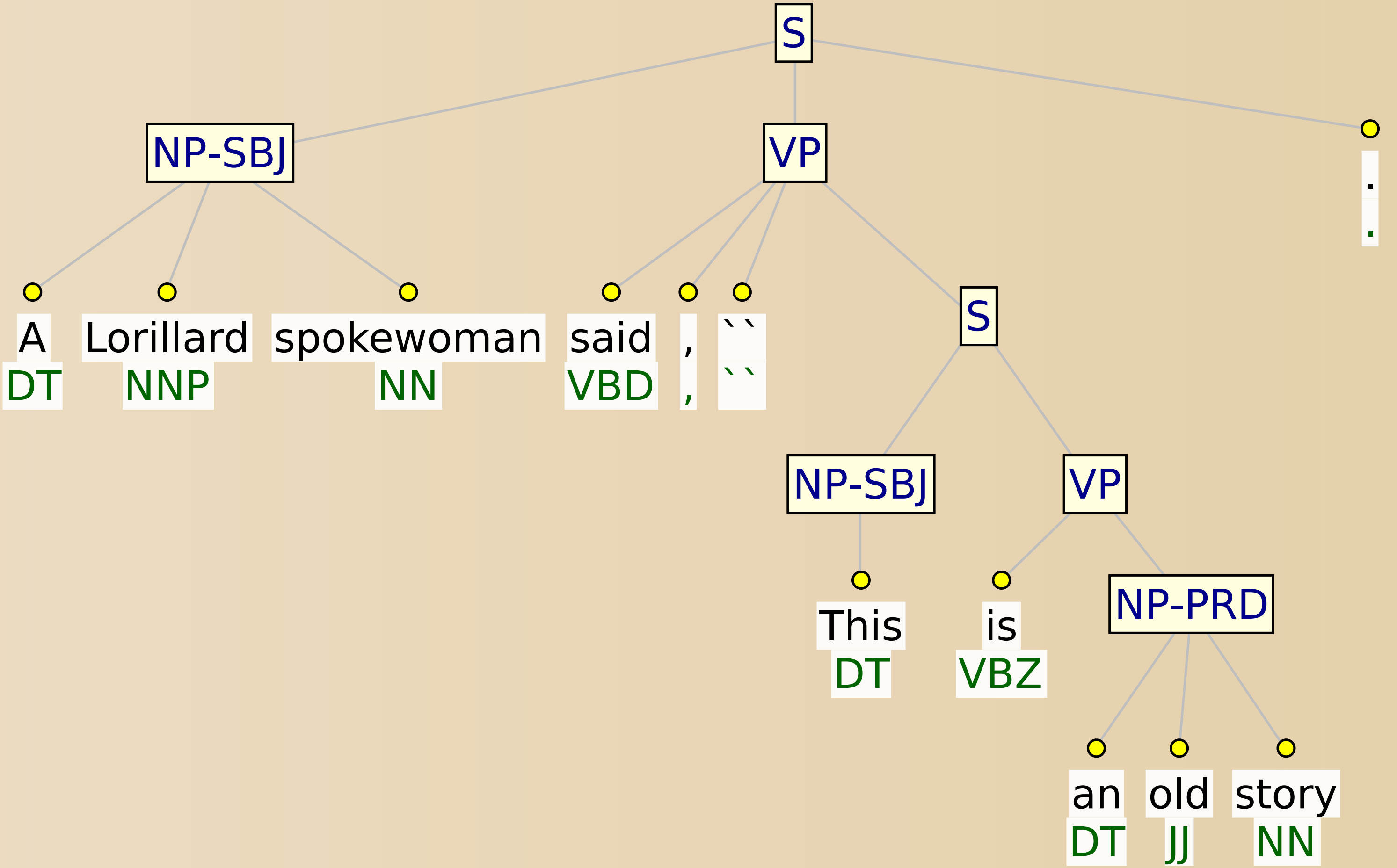
Stylesheet

How to **present** the data to the user

- **attributes** displayed at nodes
- **relations** displayed between nodes
- **shape** of nodes and edges
- **position** of nodes
- …

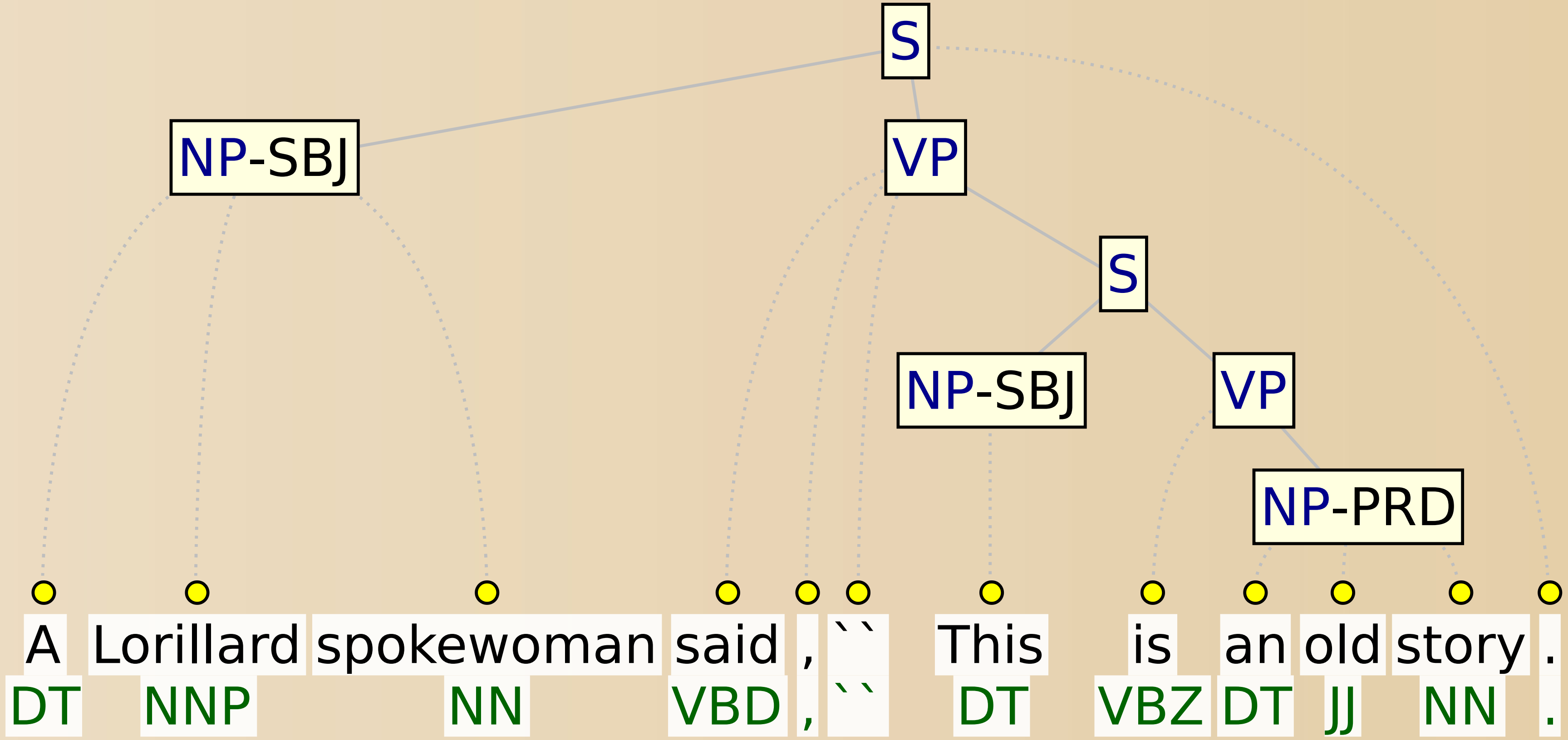
Prague Markup Language

Stylesheet



Prague Markup Language

Stylesheet



Prague Markup Language

Macros

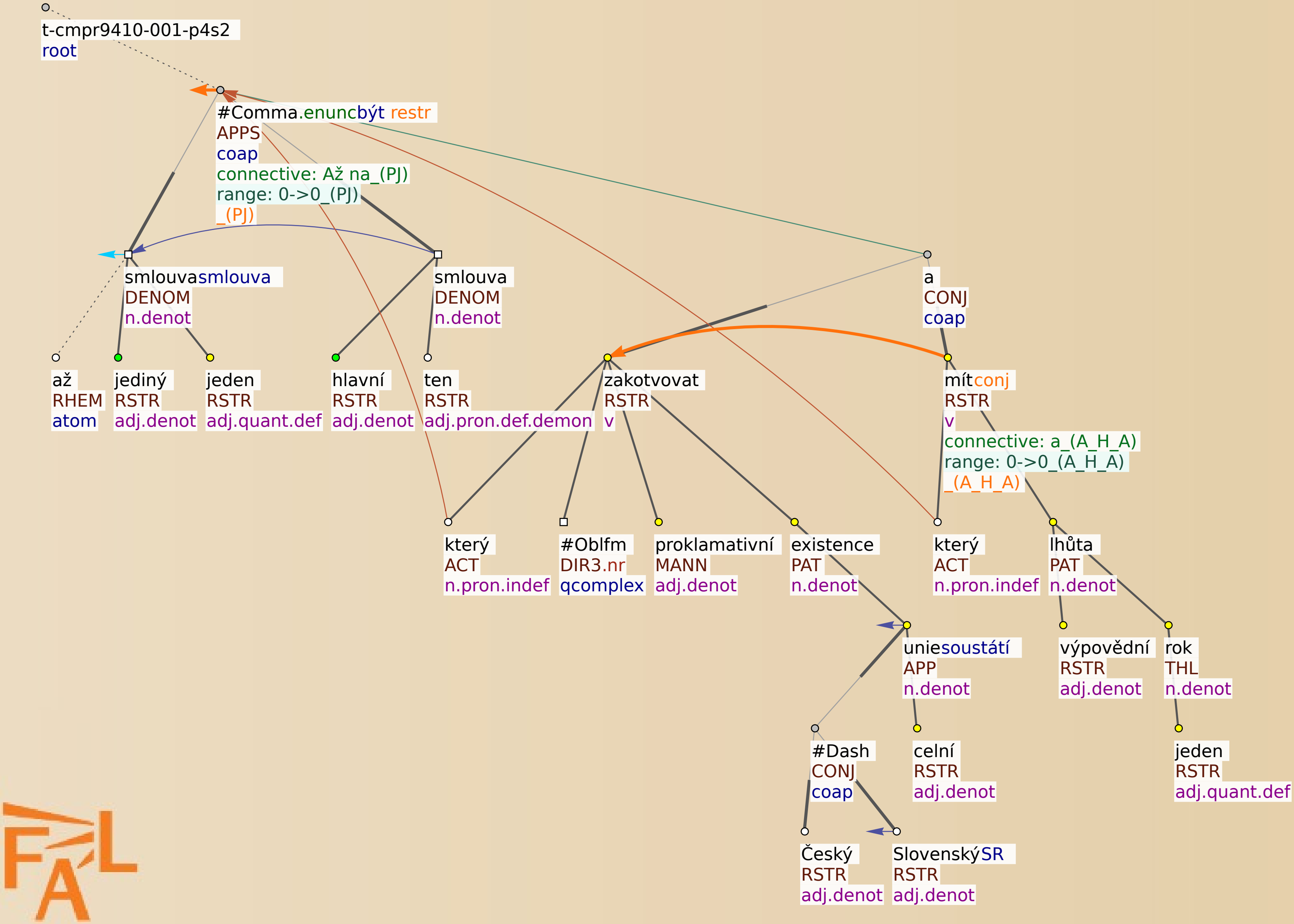
Perl code to **change** the **data or** their **appearence**

- run by a key stroke

- **annotation** of the data
- various possibilities to **present** the same data
- …

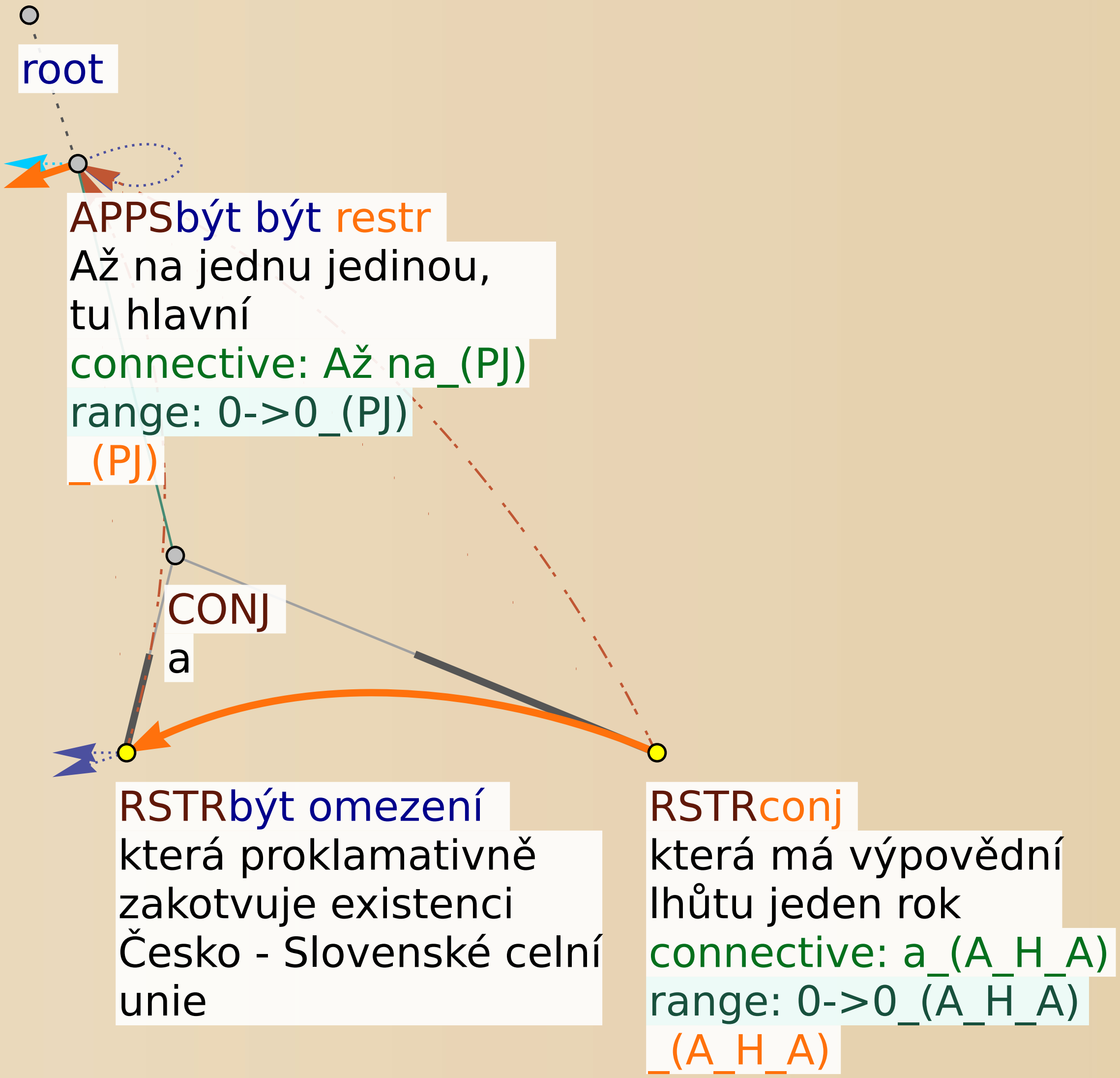
Prague Markup Language

Macros



Prague Markup Language

Macros



root

APPSbýt být restr
Až na jednu jedinou,
tu hlavní
connective: Až na_(PJ)
range: 0->0_(PJ)
_(PJ)

CONJ
a

RSTRbýt omezení
která proklamativně
zakotvuje existenci
Česko - Slovenské celní
unie

RSTRconj
která má výpovědní
lhůtu jeden rok
connective: a_(A_H_A)
range: 0->0_(A_H_A)
_(A_H_A)

Prague Markup Language
treebanks

Which treebanks we have in PML?

- Prague family of treebanks
  (PDT, PCEDT, PDTSC, CzEng, ...)
- HamleDT
- Tiger Corpus, BNC, Penn Treebank, Penn Discourse Treebank, ...

Prague Markup Language

application framework

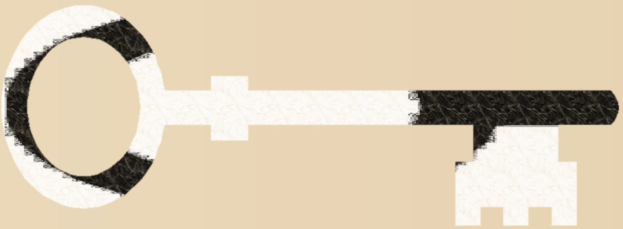
Once the data are in PML, you can

- use Tree Editor **TrEd** to **open**, **browse** and **manually edit** the data
- use **btred** to **process** the data from the command line – **apply** perl/btred **scripts** to the data
- use **PML-Tree Query** to **search** in the data

Prague Markup Language
application framework

Once the data are in PML, you can

- use Tree Editor **TrEd** to **open**, **browse** and **manually edit** the data
- use **btred** to **process** the data from the command line – **apply** perl/btred **scripts** to the data
- use **PML-Tree Query** to **search** in the data



Searching in Discourse-Annotated Treebanks

outline

- Prague Dependency Treebank (PDT) & Discourse relations
- Prague Markup Language (PML)
- PML-Tree Query
- PDT and PML-Tree Query
- PDTB and PML-Tree Query

PML-Tree Query

PML-TQ is a **powerful** **open-source** **user-friendly** **search tool** for **all kinds** of linguistically annotated **treebanks.**

PML – Prague Markup Language (XML)

TQ – Tree Query

PML-Tree Query

PML-TQ (2009): Petr Pajas, Jan Štěpánek

Pajas Petr, Štěpánek Jan: **System for Querying Syntactically Annotated Corpora**, in *Proceedings of the **ACL-IJCNLP 2009** Software Demonstrations*, Association for Computational Linguistics, Suntec, Singapore, pp. 33-36, 2009

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

Currently maintained and developed by:

Matyáš Kopp

PML-Tree Query

Client-server architecture

- **3** clients
- **2** backends (servers)

PML-TQ: Servers

2 backends (servers):

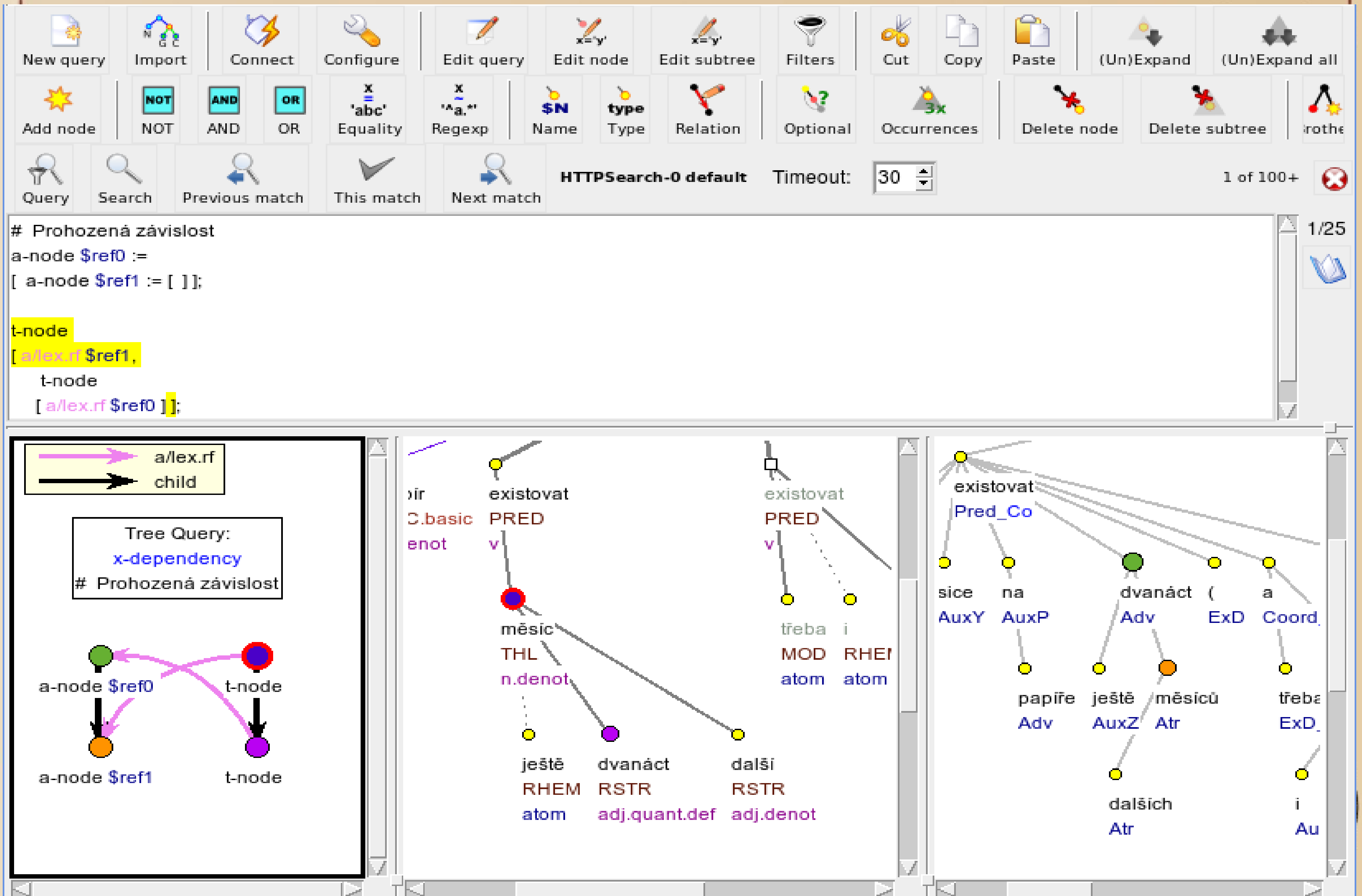
- **database** (PostgreSQL, Oracle)
  - suitable for **large**(!?), **static** treebanks
- **Tree Editor TrEd**
  - **small**, **changing** data (up to ~10k trees)

PML-TQ: Clients

3 clients:

- **Web browser** (SVG, CSS, Javascript)
  - portable, limited functionality
- **TrEd**
  - requires installation, full power of TrEd environment
- **command-line** (simple, text-based)

PML-Tree Query in TrEd



HTTPSearch-0 default   Timeout: 30   1 of 100+

1/25

```
#  Prohozená závislost
a-node $ref0 :=
[ a-node $ref1 := [ ] ];

t-node
[ a/lex.rf $ref1,
    t-node
   [ a/lex.rf $ref0 ] ];
```

PML-Tree Query
outline

- PDT – Prague Dependency Treebank
- Discourse relations in PDT
- PML – Prague Markup Language
- PML-Tree Query
- PDT and PML-Tree Query
- PDTB and PML-Tree Query

PML-Tree Query

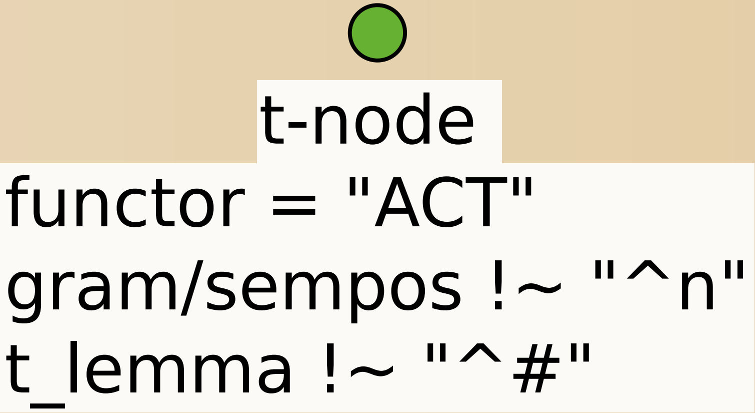
A single node (query)

A query searching for a single node that:

- is an ACTor
- its semantic part of speech is not noun
- it does not have a substitute t_lemma

Textual form of the query:

```
 t-node
  [ functor = "ACT", gram/sempos !~
"^n",
   t_lemma !~ "^#" ]
```



t-node
functor = "ACT"
gram/sempos !~ "^n"
t_lemma !~ "^#"

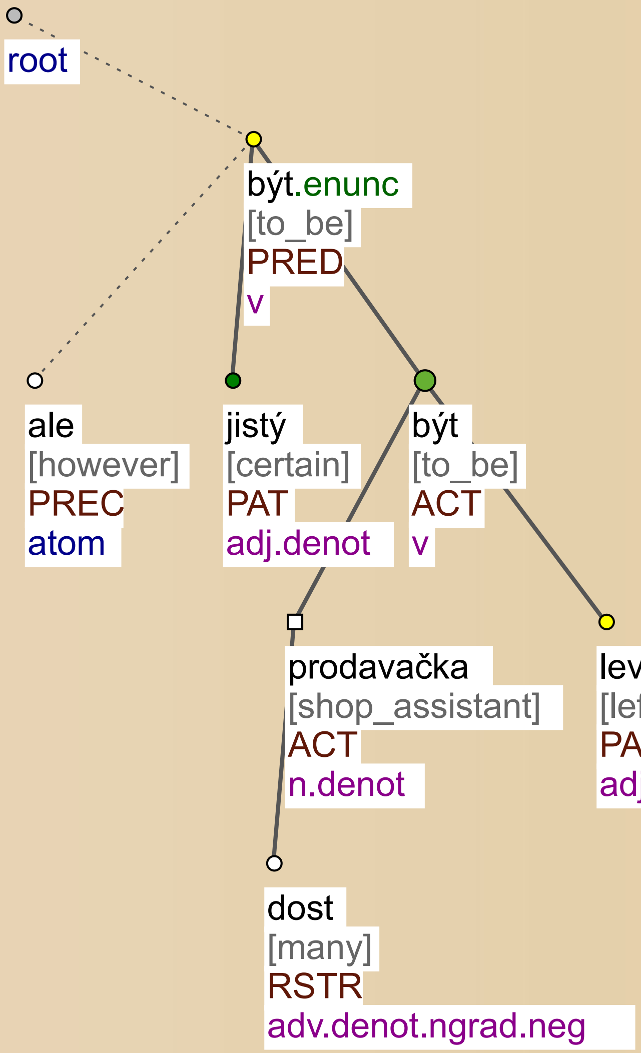
PML-Tree Query

A single node (result)

A result:

Jisté ale je, že **je** dost levorukých [prodavaček]. (PDT)

[It is, however, certain that many [shop-assistants] **are** left-handed.]



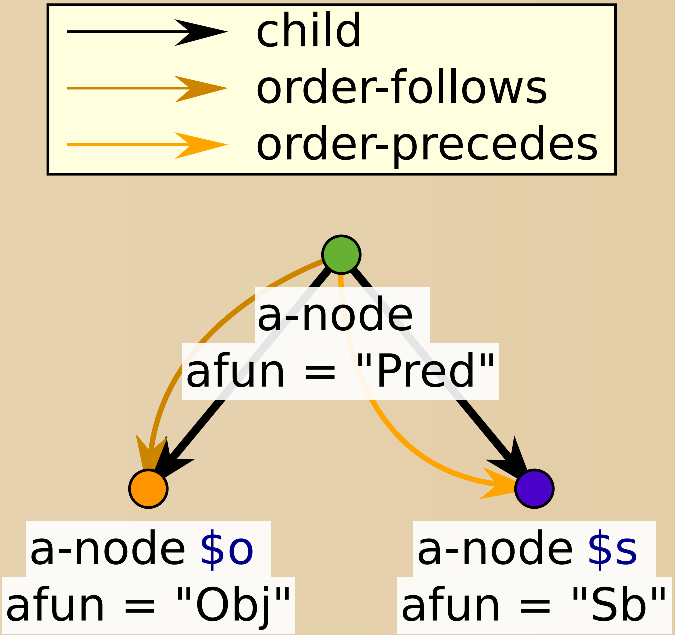
PML-Tree Query

Relations among nodes (query)

A query searching (on the analytical layer) for a Predicate governing a Subject and an Object with the surface order
Object – Predicate – Subject

Textual form of the query:

```
 a-node
  [ afun = "Pred", order-follows $o, order-
precedes $s,
   a-node $o =
    [ afun = "Obj" ],
   a-node $s =
    [ afun = "Sb" ] ]
```



PML-Tree Query

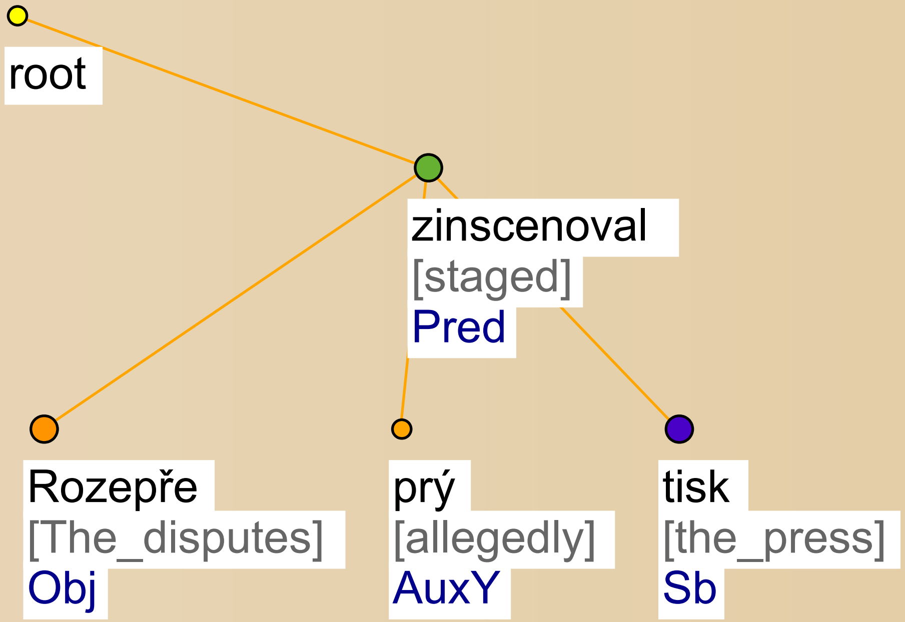
Relations among nodes (result)

A result:

Rozepře prý zinscenoval tisk. (PDT)

[lit. The_disputes$_{\text{.Acc-Obj}}$ allegedly staged the_press.$_{\text{Nom-Sb}}$]

[The disputes were allegedly staged by the press.]



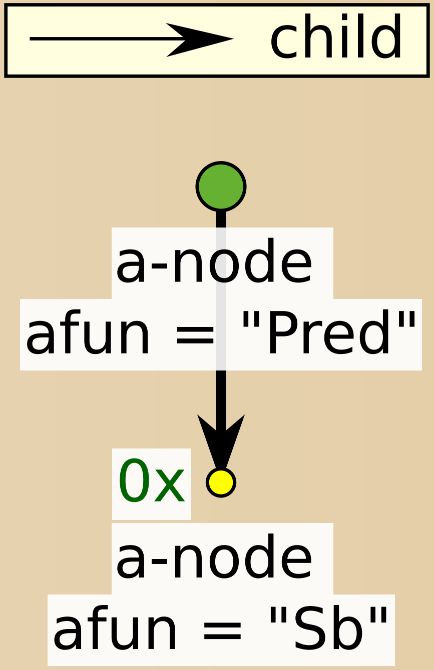
PML-Tree Query

Non-existence (query)

A query searching (on the analytical layer) for a Predicate not governing a Subject

Textual form of the query:

```
a-node
 [afun = "Pred",
  0x a-node
     [afun = "Sb"]]
```



PML-Tree Query

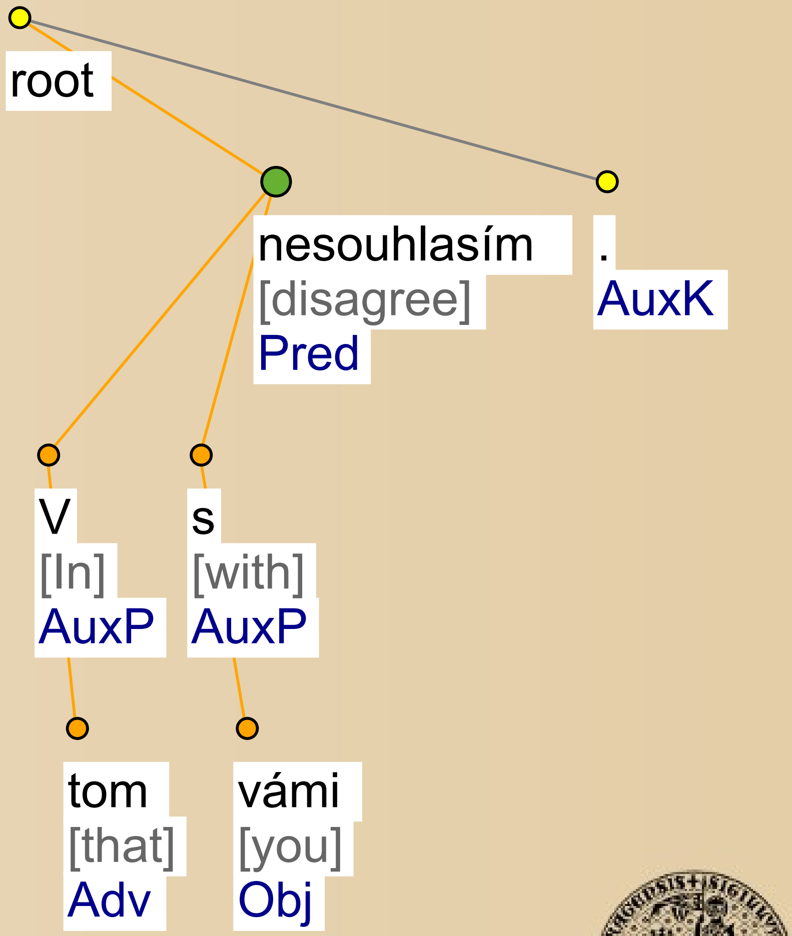
Non-existence (result)

A result:

V tom s vámi nesouhlasím. (PDT)

[lit. In that with you [I] disagree.]

[In that [I] do not agree with you.]



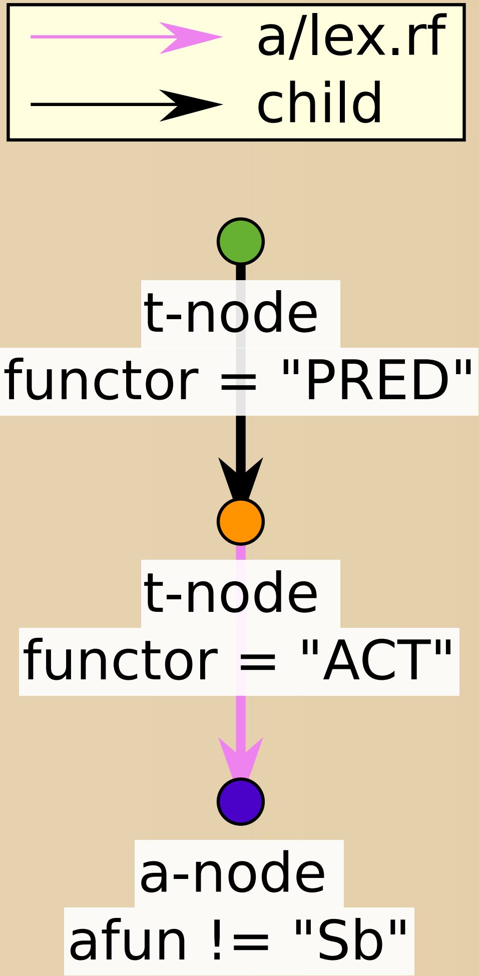
PML-Tree Query

Across layers (query)

A query searching for a PREDicate governing an ACTor that is **not** (on the analytical layer) represented by a Subject

Textual form of the query:

```
t-node
 [ functor = "PRED",
  t-node
   [ functor = "ACT",
    a/lex.rf a-node
          [ afun != "Sb" ] ] ];
```



PML-Tree Query

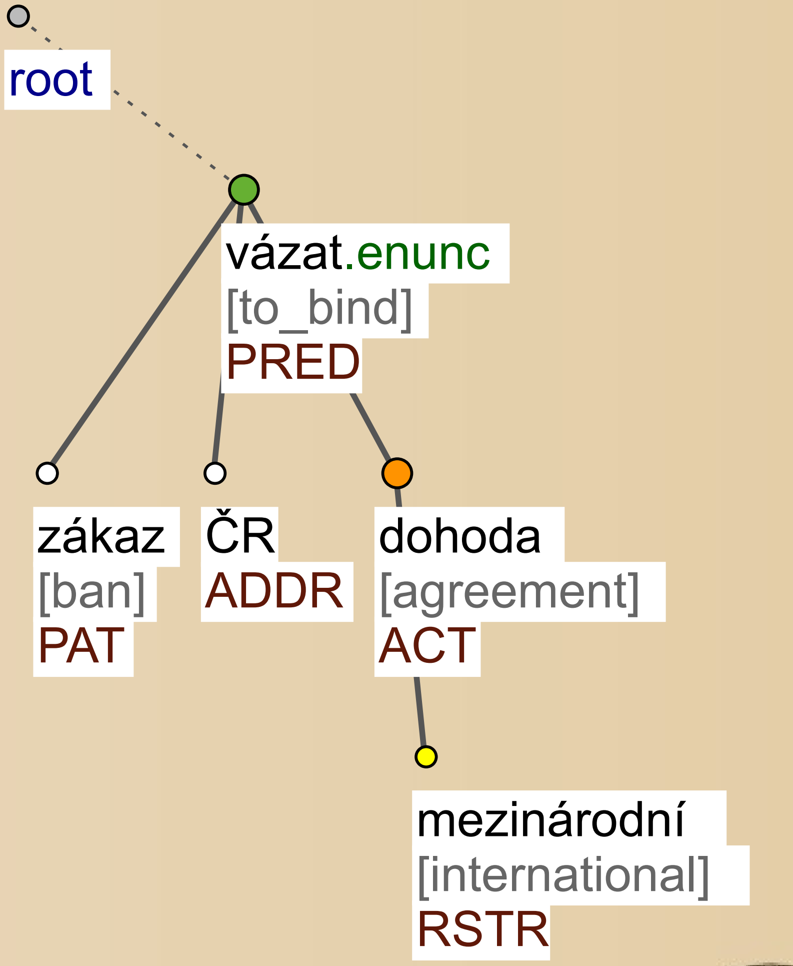
Across layers (result)

A result:

K zákazu je ČR vázána mezinárodními dohodami. (PDT)

[Lit.: To the_ban is ČR bound by international agreements$_{\text{.Instr-Obj}}$.]

[The Czech Republic is bound to [implement] the ban by international agreements.]



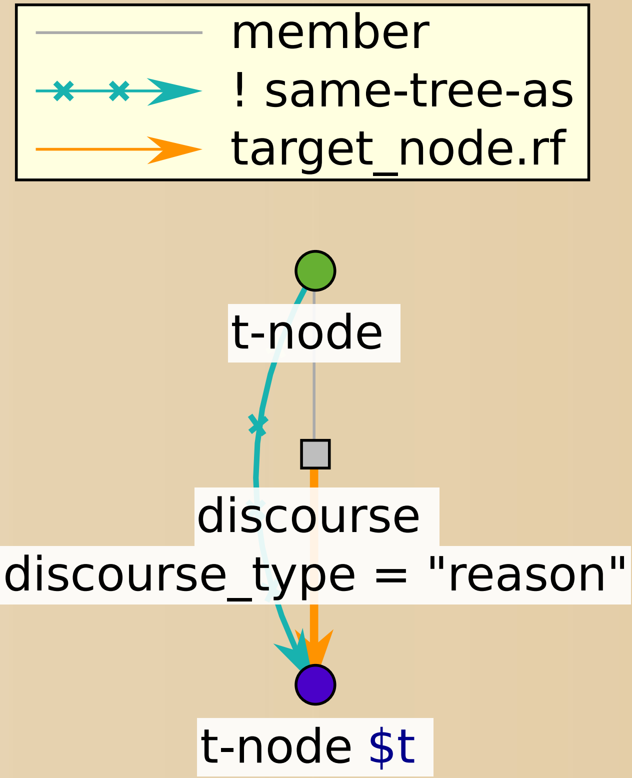
PML-Tree Query

Non-dependency relations (query)

A query searching for an inter-sentential discourse relation (technically, two nodes representing the two arguments, connected by a discourse arrow)

Textual form of the query:

```
t-node
 [ !same-tree-as $t,
  member discourse
   [ discourse type = "reason",
    target_node.rf t-node $t = [ ] ] ];
```



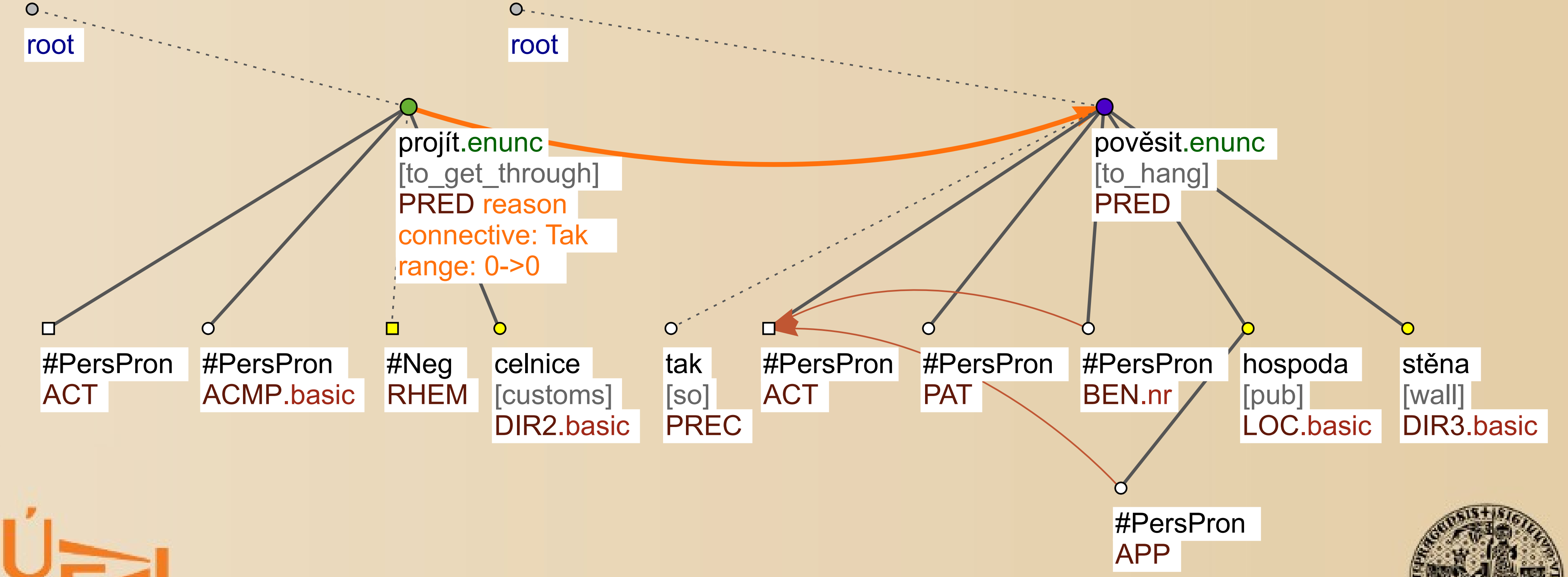
PML-Tree Query

Non-dependency relations (result)

Neprošel s ní celnicí. **Tak.**$_{\text{reason–result}}$ si ji pověsil ve své hospodě na stěnu. (PDT)

[lit.: He_did_not_get with it through_customs. **So** REFL it hung in his pub on the_wall.]

[He could not get through the customs with it. **So** he has hanged it in his pub on the wall.]



(The story is about climbers talking about a perfect prosthesis.)

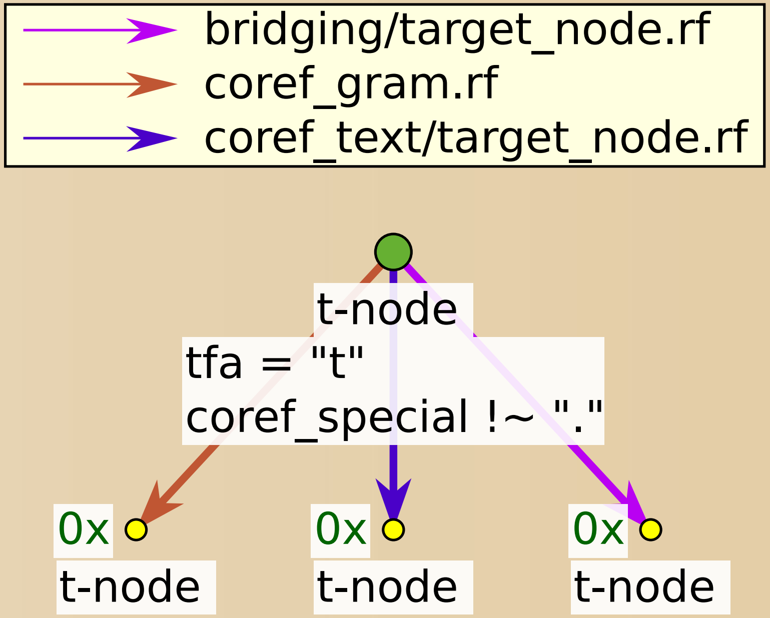
PML-Tree Query

Topic–focus articulation and anaphora (query)

A query searching for non-contrastively contextually bound nodes from which there is no anaphoric reference to the previous context

Textual form of the query:

```
t-node
 [ tfa = "t",
   coref_special !~ ".",
   0x coref_gram.rf t-node [ ],
   0x coref_text/target_node.rf t-node [ ],
   0x bridging/target_node.rf t-node [ ] ];
```



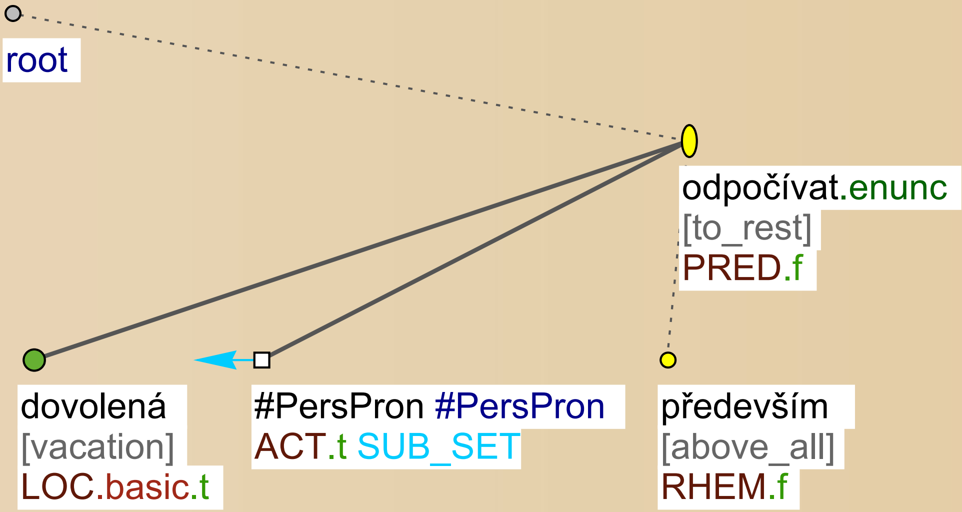
PML-Tree Query

Topic–focus articulation and anaphora (result)

A result:

Na dovolené chceme především odpočívat. (PDT)

[On vacation, we want above all to rest.



Previous context:

Pojedete do zahraničí s cestovkou? (PDT)

[Will you go abroad with a travel agency?]

PML-Tree Query

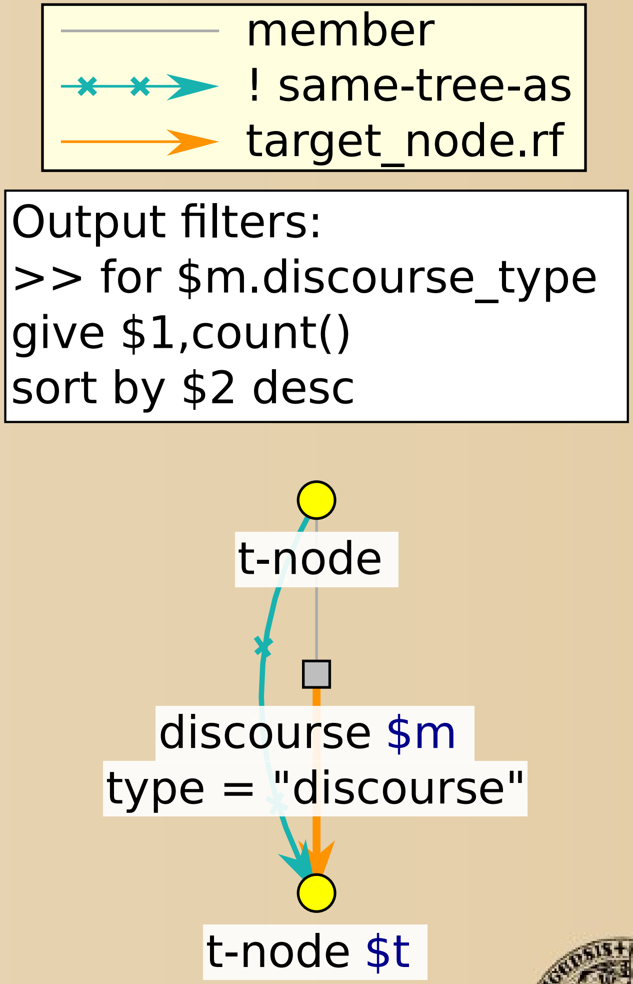
Output filters (query)

A query searching for inter-sentential discourse relations; the output filter provides a distribution of discourse types

Textual form of the query:

```
t-node
 [ !same-tree-as $t,
  member discourse $m =
   [ type = "discourse",
    target_node.rf t-node $t = [ ] ] ];
```

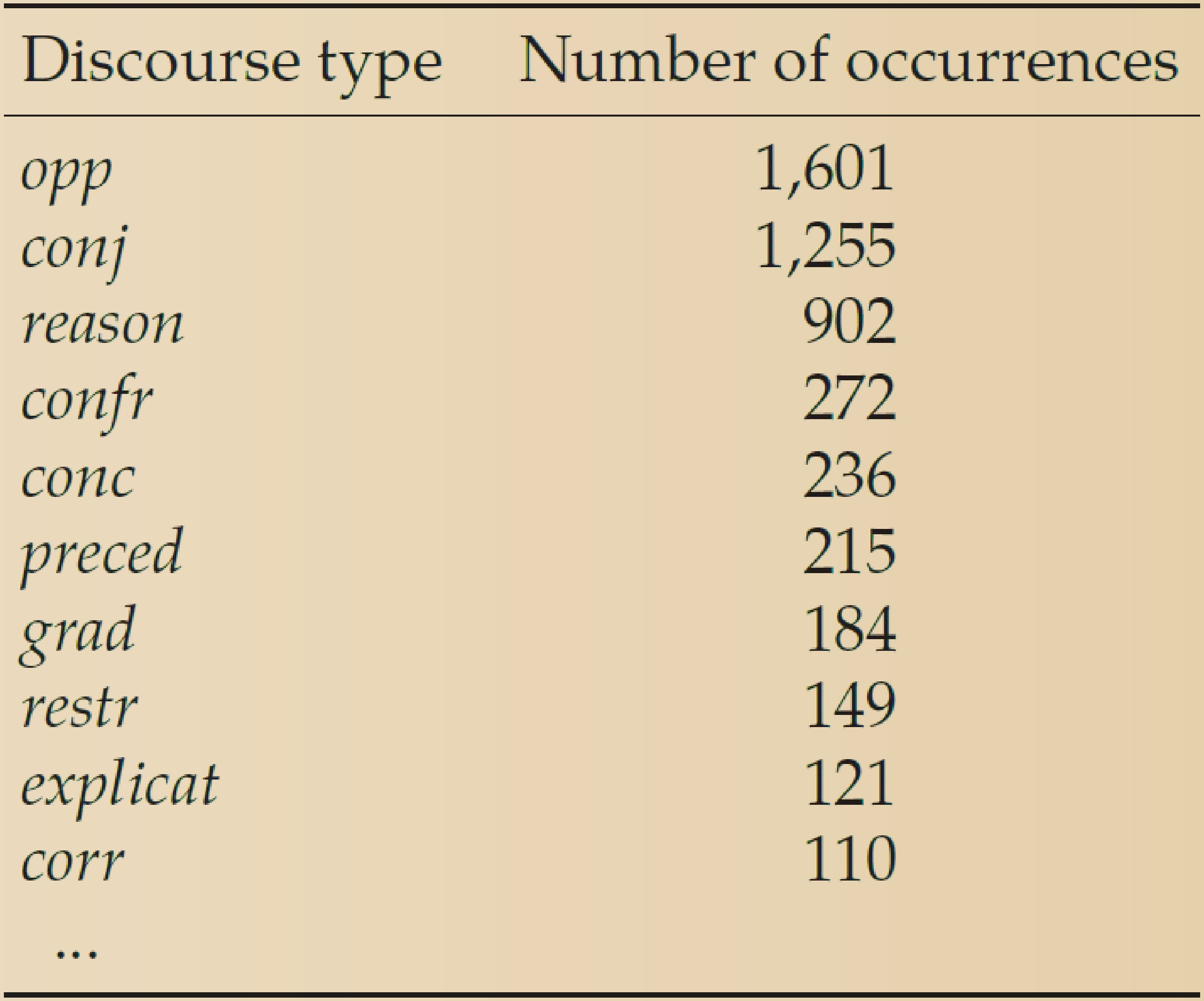
```
>> for $m.discourse_type give $1, count() sort
by $2 desc
```



Output filters:
>> for $m.discourse_type
give $1,count()
sort by $2 desc

PML-Tree Query

Output filters (result)



Discourse type	Number of occurrences
opp	1,601
conj	1,255
reason	902
confr	272
conc	236
preced	215
grad	184
restr	149
explicat	121
corr	110
...	

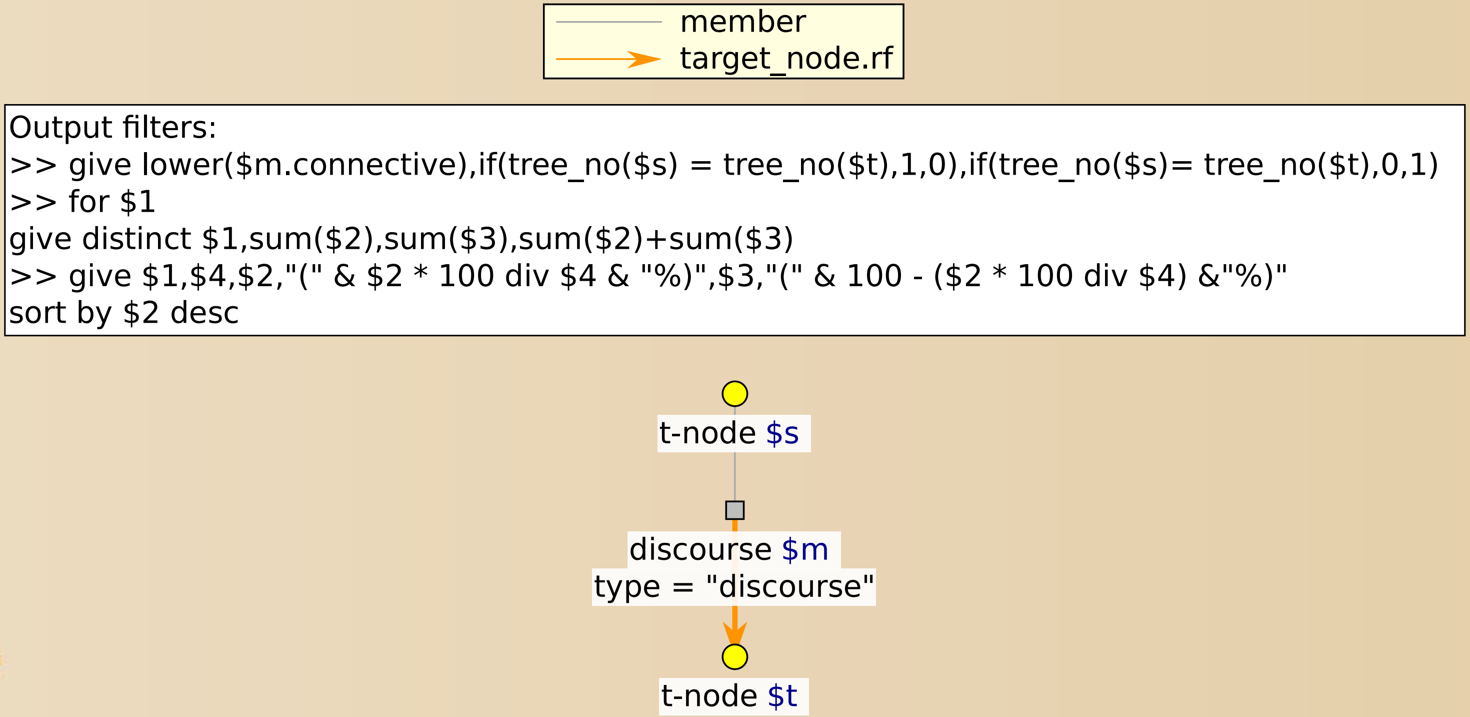
PML-Tree Query

Output filters (query #2)

A query searching for all discourse relations; the output filter gives a distribution of connectives and their intra- and inter-sentential usages

Textual form of the query:

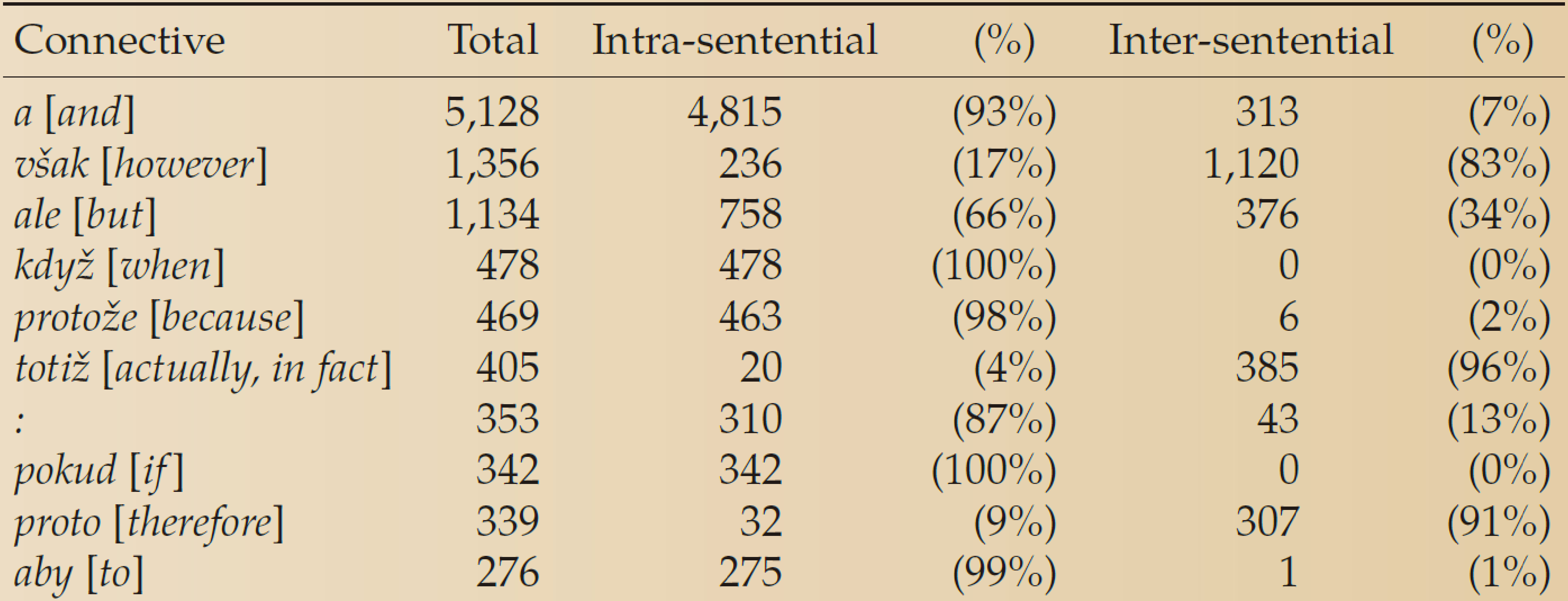
```
t-node $s :=
 [ member discourse $m :=
   [ type = "discourse", target_node.rf t-node $t := [ ] ] ];
>> give lower($m.connective), if(tree_no($s) = tree_no($t),1,0), if(tree_no($s)
= tree_no($t),0,1)
>> for $1 give distinct $1, sum($2), sum($3), sum($2)+sum($3)
>> give $1,$4,$2,"(" & $2 * 100 div $4 & "%)", $3, "(" & 100 - ($2 * 100 div $4) &
"%)" sort by $2 desc
```



```
Output filters:
>> give lower($m.connective),if(tree_no($s) = tree_no($t),1,0),if(tree_no($s)= tree_no($t),0,1)
>> for $1
give distinct $1,sum($2),sum($3),sum($2)+sum($3)
>> give $1,$4,$2,"(" & $2 * 100 div $4 & "%)",$3,"(" & 100 - ($2 * 100 div $4) &"%)"
sort by $2 desc
```

PML-Tree Query

Output filters (result #2)



Connective	Total	Intra-sentential	(%)	Inter-sentential	(%)
a [and]	5,128	4,815	(93%)	313	(7%)
však [however]	1,356	236	(17%)	1,120	(83%)
ale [but]	1,134	758	(66%)	376	(34%)
když [when]	478	478	(100%)	0	(0%)
protože [because]	469	463	(98%)	6	(2%)
totiž [actually, in fact]	405	20	(4%)	385	(96%)
:	353	310	(87%)	43	(13%)
pokud [if]	342	342	(100%)	0	(0%)
proto [therefore]	339	32	(9%)	307	(91%)
aby [to]	276	275	(99%)	1	(1%)

Searching in Discourse-Annotated Treebanks

outline

- Prague Dependency Treebank (PDT) & Discourse relations
- Prague Markup Language (PML)
- PML-Tree Query
- PDT and PML-Tree Query
- PDTB and PML-Tree Query

PDTB 2.0

Penn **D**iscourse **T**reebank 2.0 (2008, LDC)

- WSJ part of the Penn Treebank
  - **50 thousand** sentences annotated (among others) on the surface syntax layer
- discourse relations annotated on raw texts

→ We use a combination of both annotations and a transformation to the PML

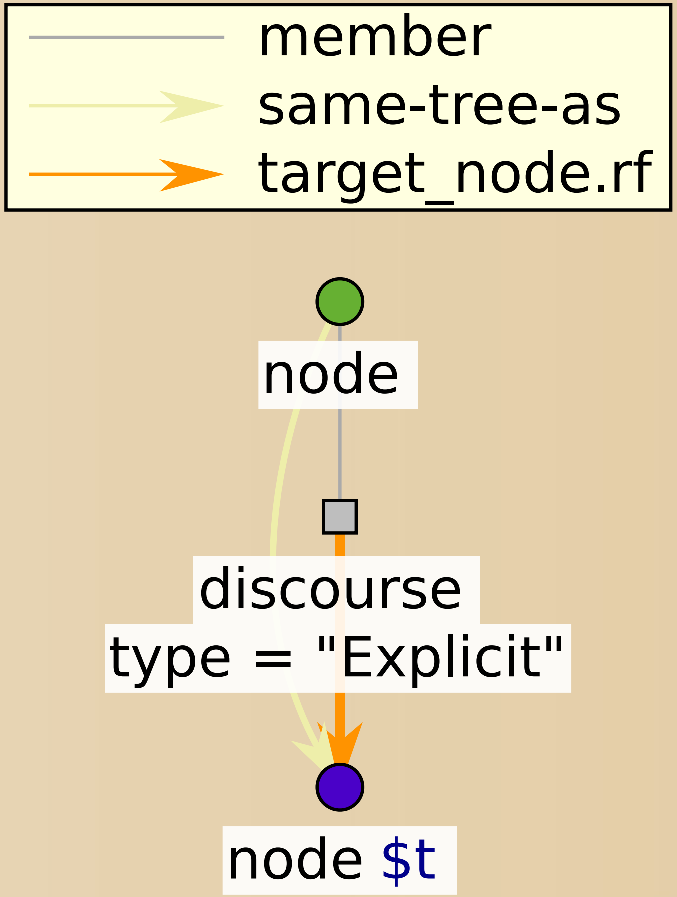
PML-Tree Query

A simple PDTB example (query)

A query searching for an intra-sentential discourse relation with an explicit connective

Textual form of the query:

```
node
 [ same-tree-as $t,
  member discourse
   [ type = "Explicit",
    target_node.rf node $t := [ ] ] ];
```

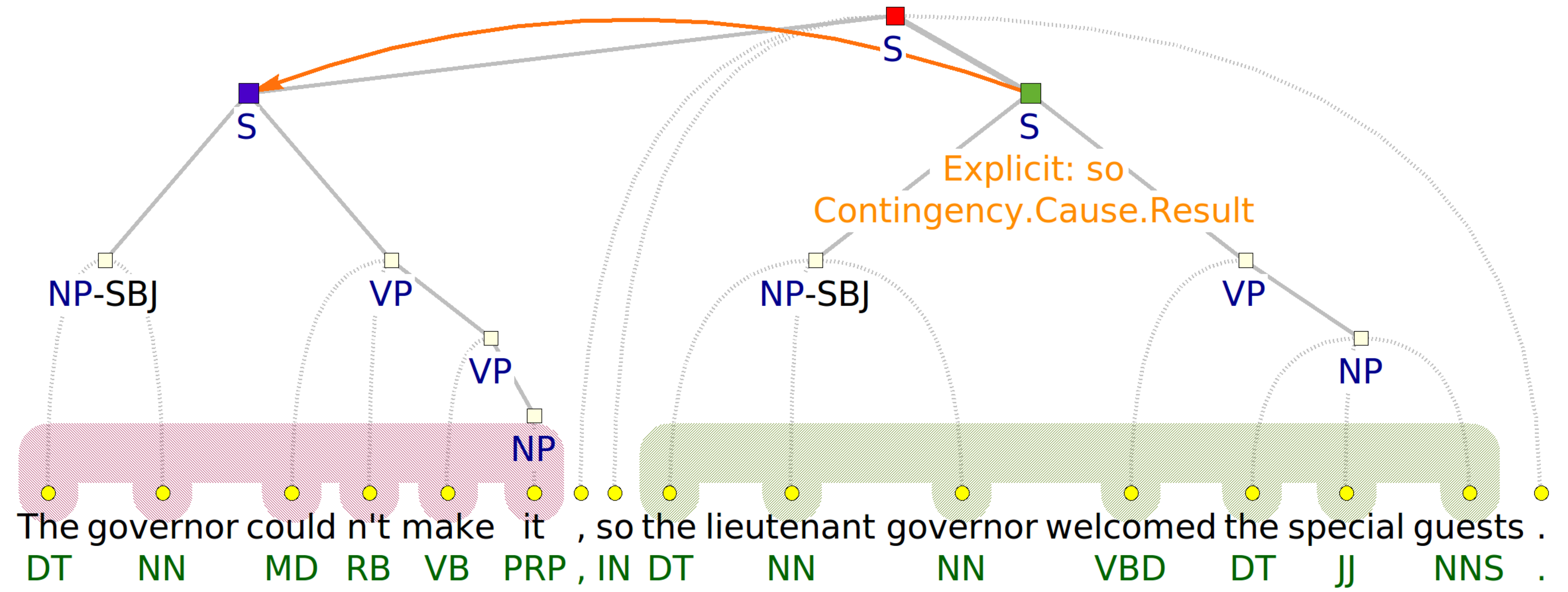


PML-Tree Query

A simple PDTB example (result)

A result:

The governor couldn't make it, so **the lieutenant governor welcomed the special guests.** (PDTB-2.0)


S
S
S
Explicit: so
Contingency.Cause.Result
NP-SBJ
VP
VP
NP
NP-SBJ
VP
NP
The governor could n't make it , so the lieutenant governor welcomed the special guests .
DT NN MD RB VB PRP , IN DT NN NN VBD DT JJ NNS .

PML-Tree Query

A PDTB example with an output filter (query)

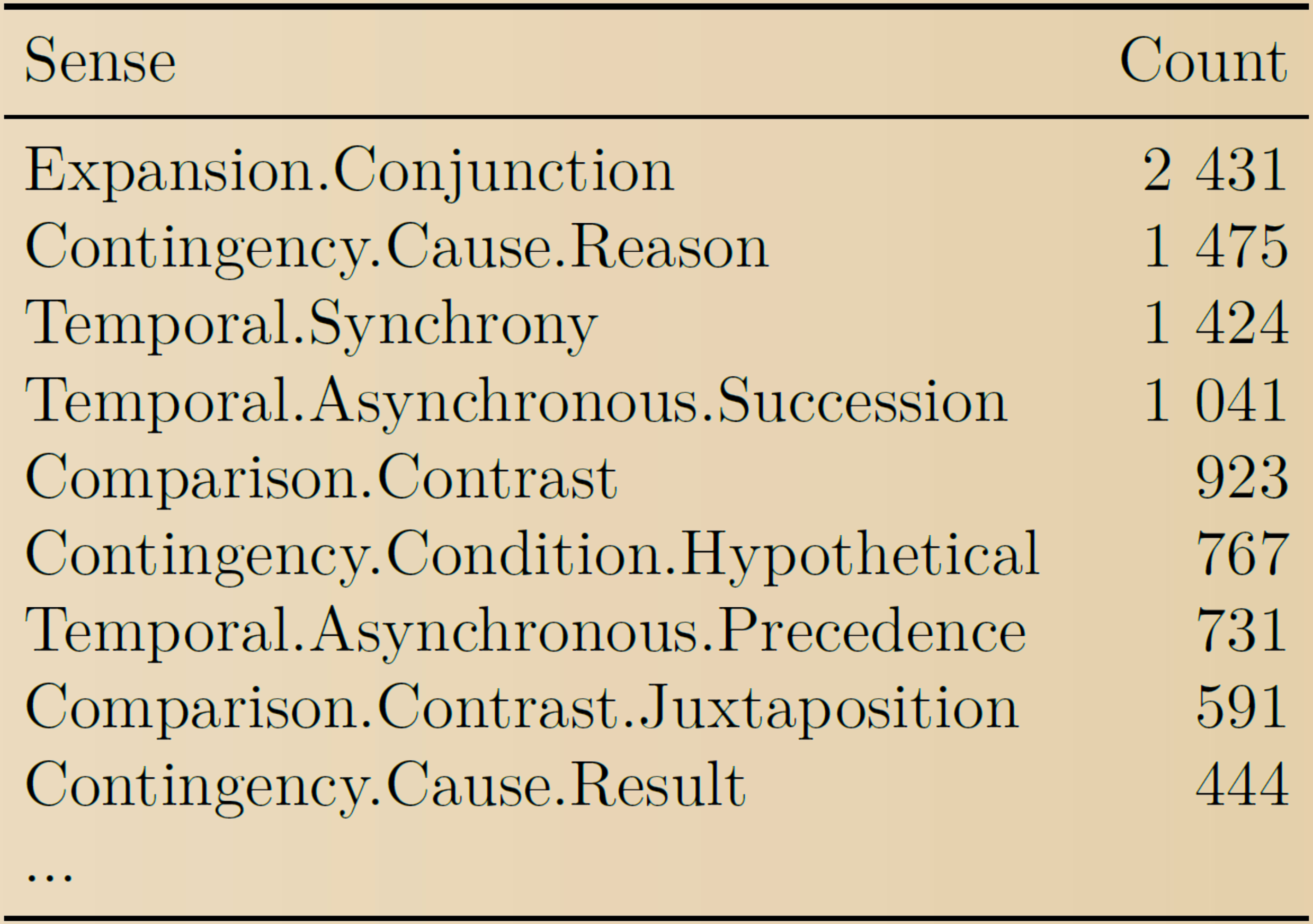
A query searching for all explicit intra-sentential discourse relations and – thanks to the **output filter** – produces a distribution table of the senses of these relations, sorted in the descending order by the number of occurrences

Textual form of the query:

```
node
 [ same-tree-as $t,
  member discourse
    [ type = "Explicit", target_node.rf node $t := [ ],
     member conn
      [ member sem $s := [ ] ] ] ];
>> for $s.sense give $1,count() sort by $2 desc
```

PML-Tree Query

A PDTB example with an output filter (result)



Sense	Count
Expansion.Conjunction	2 431
Contingency.Cause.Reason	1 475
Temporal.Synchrony	1 424
Temporal.Asynchronous.Succession	1 041
Comparison.Contrast	923
Contingency.Condition.Hypothetical	767
Temporal.Asynchronous.Precedence	731
Comparison.Contrast..Juxtaposition	591
Contingency.Cause.Result	444
...	

PML-Tree Query

A PDTB example with genres (query)

A query searching for all senses annotated at all discourse relations in the data and produces distributions of the four semantic classes for each individual genre

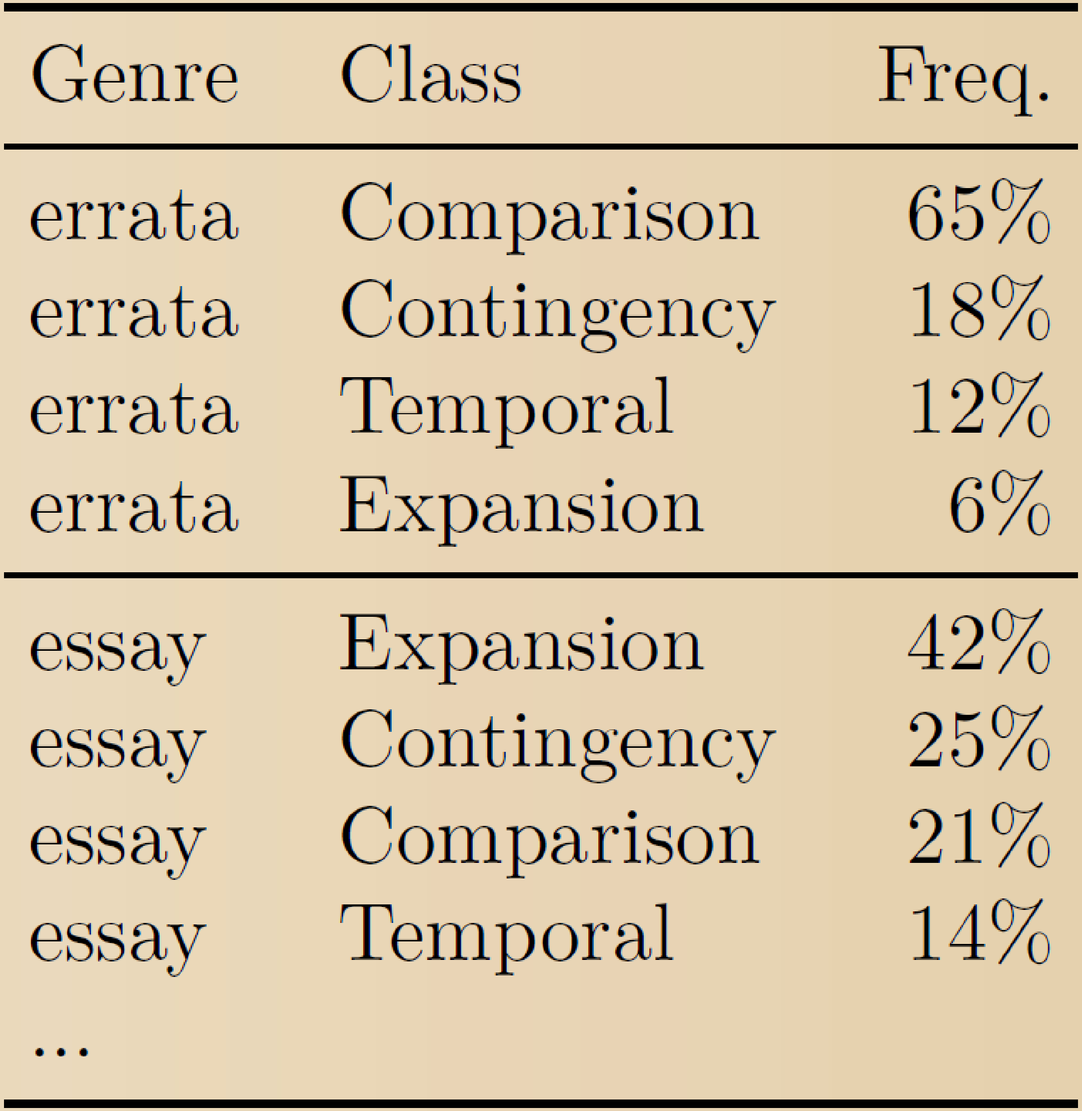
Textual form of the query:

```
root $r :=
 [ descendant node
   [ member discourse
     [ member conn
       [ member sem $s := [ ] ] ] ] ];
```

```
>> for $r.genre_ad,match($s.sense,'^[^.]+') give $1,$2,count() sort by $1,$3 desc
>> give $1,$2,ceil($3 * 100 div sum($3 over $1)) & '%'
```

PML-Tree Query

A PDTB example with genres (result)



Genre	Class	Freq.
errata	Comparison	65%
errata	Contingency	18%
errata	Temporal	12%
errata	Expansion	6%
essay	Expansion	42%
essay	Contingency	25%
essay	Comparison	21%
essay	Temporal	14%
…		

PML-Tree Query

A PDTB example: Appendix A of the PDTB manual (query)

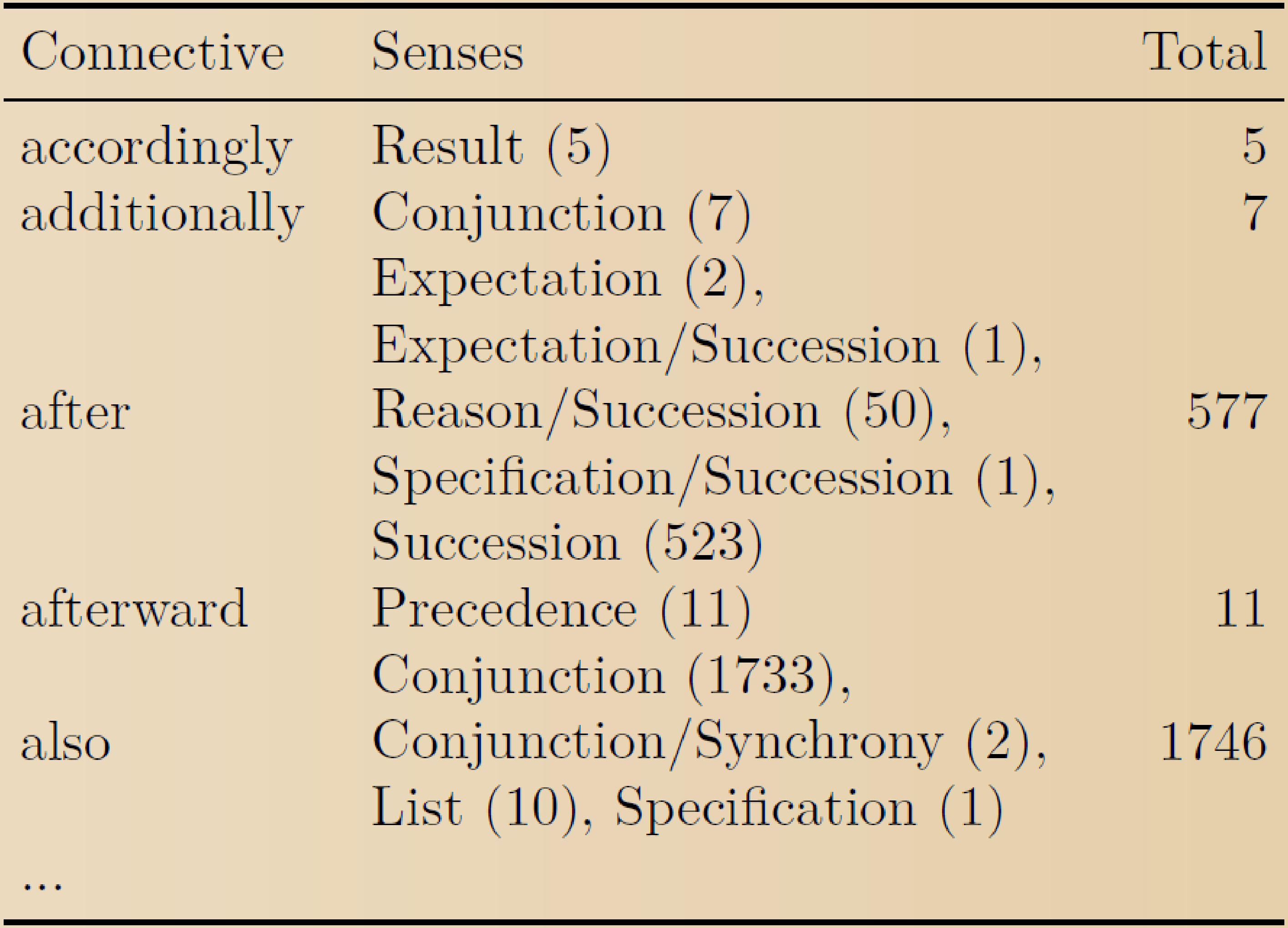
A query searching for all explicit discourse relations; the output filter produces a distribution of senses for each connective

Textual form of the query:

```
node
 [ member discourse
   [ type = "Explicit",
     member conn $c :=
      [ member sem $s := [ ] ] ] ];
>> give $c,lower($c.head),match($s.sense,'[^\.]+$')
>> give distinct $1,$2,concat($3,'/' over $1 sort by $3)
>> for $2,$3 give $1,$2,count()
>> for $1,$2,$3 give $1,$2 & ' (' & $3 & ')', sum($3 over $1) sort by $1,$2
>> give distinct $1,concat($2,', ' over $1),$3 sort by $1
```

PML-Tree Query

A PDTB example: Appendix A of the PDTB manual (result)



Connective	Senses	Total
accordingly	Result (5)	5
additionally	Conjunction (7)	7
after	Expectation (2), Expectation/Succession (1), Reason/Succession (50), Specification/Succession (1), Succession (523)	577
afterward	Precedence (11)	11
also	Conjunction (1733), Conjunction/Synchrony (2), List (10), Specification (1)	1746
…		

PML-Tree Query

Query Language Highlights

- **queries** can span **over all layers** of annotation (including annotation dictionaries) and **over all sentences in one document**
- allows **arbitrary logical constraints**
- supports **output filters** (generate custom text output, compute statistics, ...)
- offers **graphical query representation** with **relations** (links) between nodes **depicted as arrows**
- understands **PML data model** (no conversion, no information loss)

PML-Tree Query

Thank you for your attention!