PML-Tree Query

Jiří Mírovský

Charles University in Prague

Institute of Formal and Applied Linguistics

PML-Tree Query

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

PML – Prague Markup Language (XML)

TQ – Tree Query

PML-Tree Query

Before PML-TQ

Before PML-TQ

Manatee/Bonito (Rychlý 2000)

for searching in **huge** **linear** linguistic data (such as morphologically annotated texts)

[lemma="jaro" & tag="N...6.+" & word="j.+"]

Used e.g. for **Czech National Corpus** (hundreds of millions of words)

Before PML-TQ

TGrep (Pito 1994)

developed primarily **for the Penn Treebank**; usable for any treebank where each node is evaluated with **only one symbol** – either a non-terminal or a token

S <1 /^NP/ < (VP < (NP $.. NP))

Get all Ss that start with an NP and that dominate a VP that in turn has two NP sons. The predicates used in this example mean:

<1 immediate dominance, first child

< immediate dominance

$.. brotherhood, precedence

Before PML-TQ

TGrep2 (Rohde 2001-2005)

A sequel to TGrep, many enhancements of the query language, e.g. **Boolean expressions in relations between nodes**

A [< B | ![. C !, F]] | ![< D !.. E]

means: (A has son B **or** it does **not** (immediately precede C **and** **not** immediately follow F)) **or** (A does **not** (have son D **and** is **not** followed by E))

Before PML-TQ

TigerSearch (Lezius 2002)

graphical search tool for the Tiger Treebank

```
(#n:[cat="S"] > [pos="PRELS"]) &
(#n > [word="lacht" & pos="VVFIN"])
```

> immediate dominance

all node expressions in the query are **existentially quantified**

Before PML-TQ

Other search tools:

Oraculum (Ljubopytnov et al. 2002) - PDT

Viqtorya (Steiner, Kallmeyer 2002) - Tübingen Treebanks

Finite structure query (fsq, Kepser 2003) - Tübingen Treebanks

Netgraph 1.0 (Ondruška 1998) – PDT

Before PML-TQ

Netgraph 2.0 (Mírovský 2000-2008)

client-server based search tool for PDT and other treebanks

graphically oriented **creation** and representation **of the query**

graphical representation of the result

powerful but **easy-to-use** query language – aimed at **linguists** without programming skills

Before PML-TQ

Netgraph 2.0 query language

determined by the requirements set **by the annotated data**

e.g. to study:

word order – a way to control **left-right order of nodes**

coreference – a way to establish the **non-dependency relation** between nodes and **set attributes** of both nodes

across layers – a way to access lower layers **with non-1:1 relation** among nodes

PDT Requirements

Complex Evaluation of a Node

multiple attributes evaluation (an ability to set values of several attributes at one node)

alternative values (e.g. to define that functor of a node is either a disjunction or a conjunction)

alternative nodes (alternative evaluation of the whole set of attributes of a node)

wild cards (regular expressions) in values of attributes

negation (e.g. to express “this node is not an Actor”)

relations less than (<) , greater than (>) (for numerical attributes)

PDT Requirements

Dependencies Between Nodes (Vertical Relations)

immediate, transitive dependency (existence, non-existence)

vertical distance (from root, from one another)

number of sons (zero for leaves)

Horizontal Relations

precedence, immediate precedence (positive, negative)

horizontal distance

secondary edges (secondary dependencies, coreferences, long-range relations)

PDT Requirements

Other Features

multi-tree queries (combined with general OR relation)

skipping a node of a given type (for skipping simple types of coordination, apposition etc.)

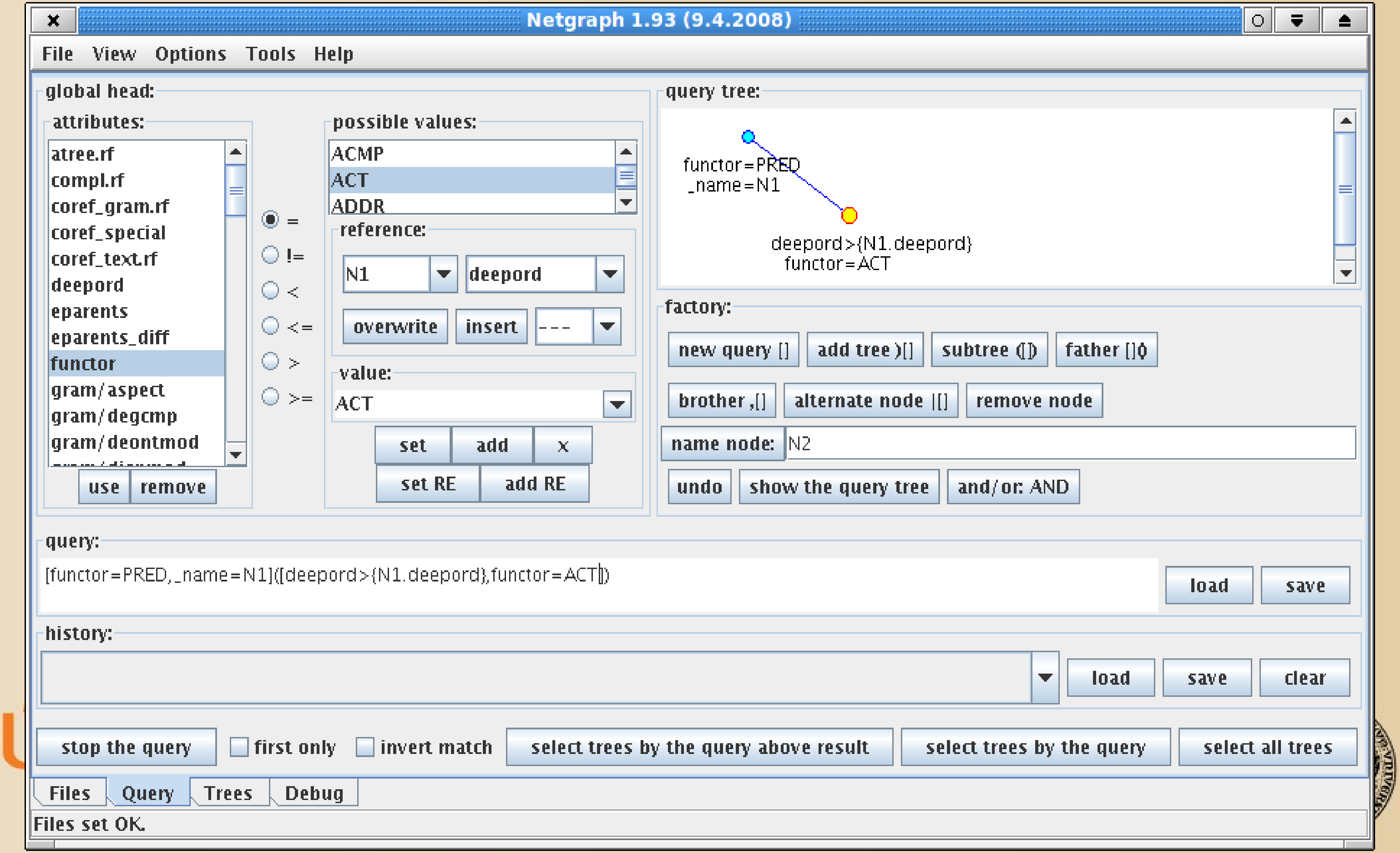
skipping multiple nodes of a given type (e.g. for recognizing the rightmost path)

references (for matching values of attributes unknown at the time of creating the query)

accessing several layers of annotation at the same time with **non-1:1** relation (for studying relation between layers)

searching in the surface form of the sentence

Netgraph



attributes: functor

possible values: ACT

reference: N1 deepord

value: ACT

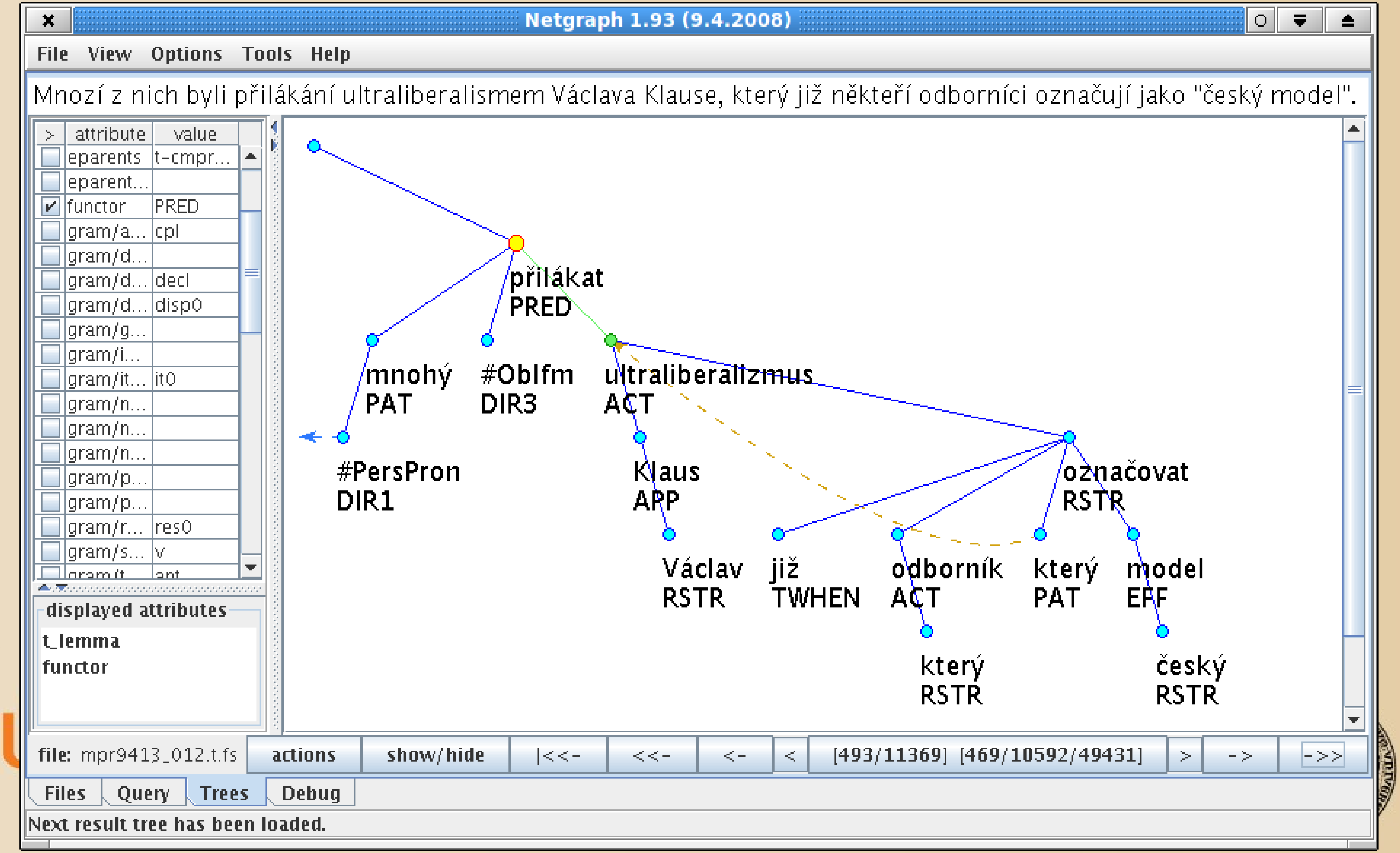
name node: N2

query:

```
[functor=PRED,_name=N1]([deepord>{N1.deepord},functor=ACT])
```

Files set OK.

Netgraph



Mnozí z nich byli přilákáni ultraliberalismem Václava Klause, který již někteří odborníci označují jako "český model".

>	attribute	value
☐	eparents	t-cmpr...
☐	eparent...	
☑	functor	PRED
☐	gram/a...	cpl
☐	gram/d...	
☐	gram/d...	decl
☐	gram/d...	disp0
☐	gram/g...	
☐	gram/i...	
☐	gram/it...	it0
☐	gram/n...	
☐	gram/n...	
☐	gram/n...	
☐	gram/p...	
☐	gram/p...	
☐	gram/r...	res0
☐	gram/s...	v
☐	gram/t...	ant

displayed attributes

t_lemma
functor

file: mpr9413_012.t.fs

Next result tree has been loaded.

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: **Michal Sedlák**

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:

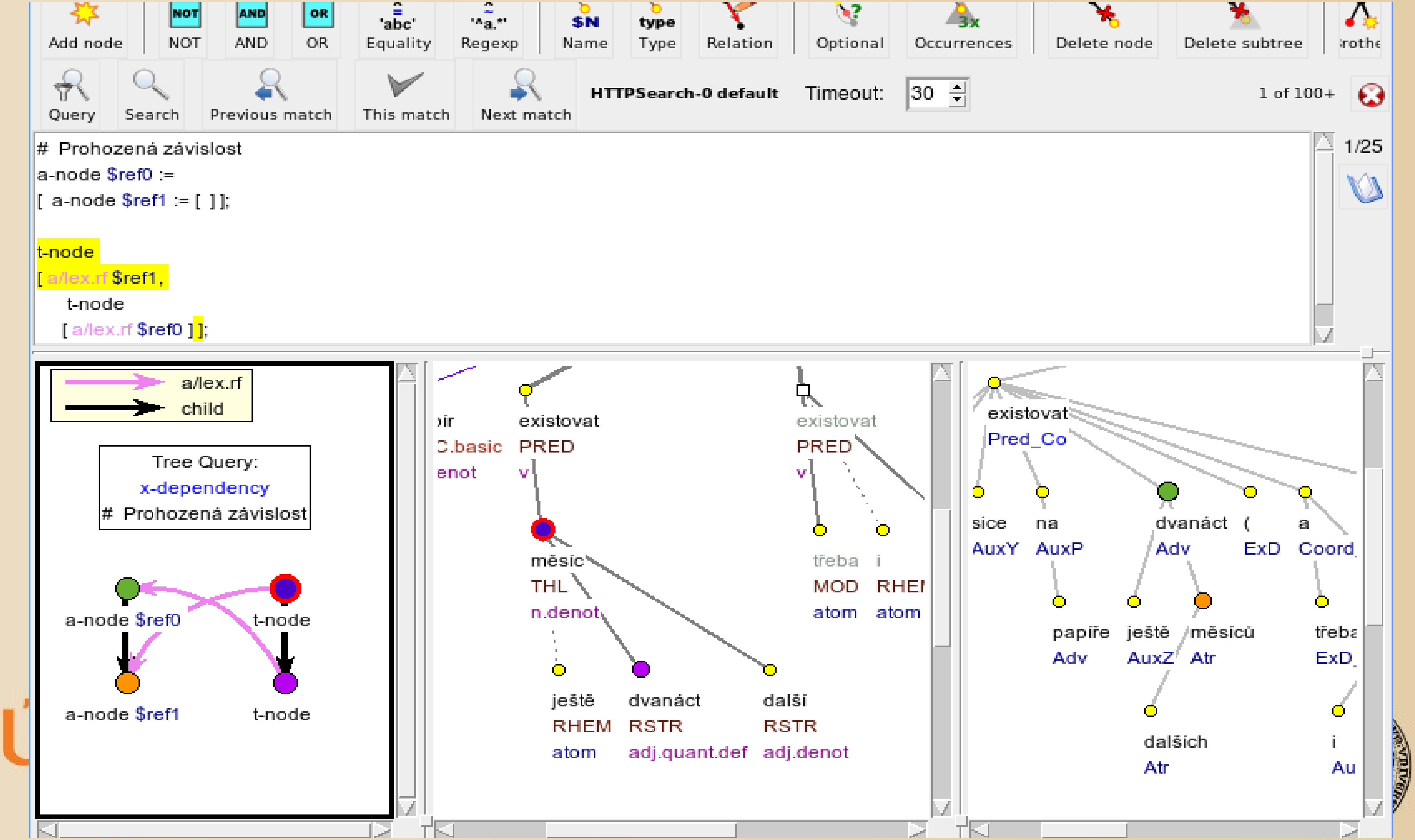
- **Web browser** (SVG, CSS, Javascript)
  - portable, limited functionality
- **TrEd**
  - requires installation, full power of TrEd environment
- **command-line** (simple, text-based)

PML-Tree Query

Query Language Highlights

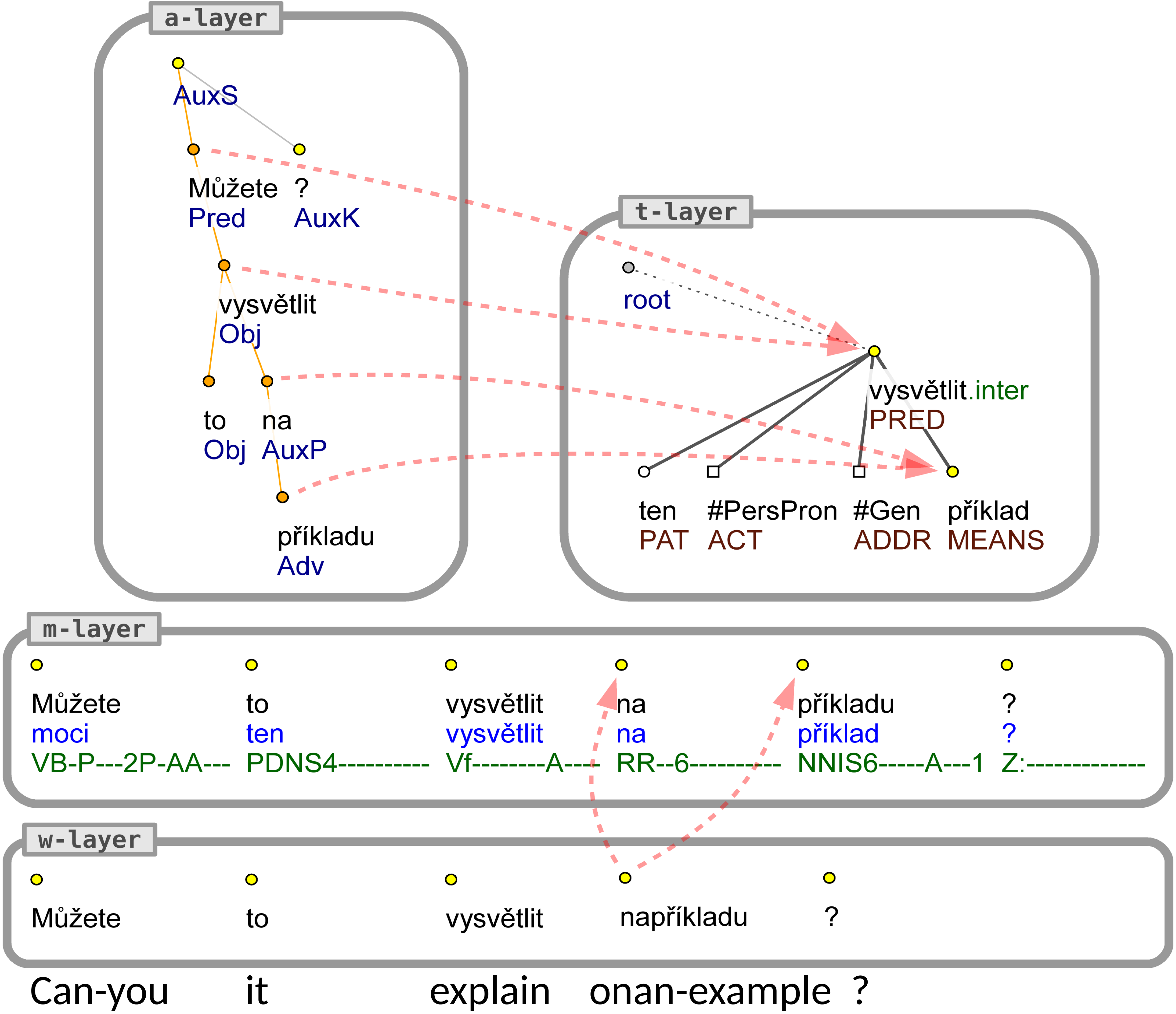
- **queries** can span **over all layers** of annotation (including annotation dictionaries) and **over all sentences in one document**
- allows **arbitrary logical constraints**
- supports **output filters** (generate custom text output, compute statistics, ...)
- offers **graphical query representation**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

PML-Tree Query in TrEd



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

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


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