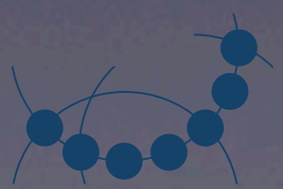
PML-TQ

Tree Query Language and software stack

PML Tree Query

- Query and report language
- Two engines
- SQL
- Perl ([bnj]tred)
- Several clients
- Browser
- TrEd
- CLI
- The SQL engine requires the data to be converted to SQL and loaded into a database → stable datasets.
- The Perl engine is slow, but works directly with the data files → data in progress.

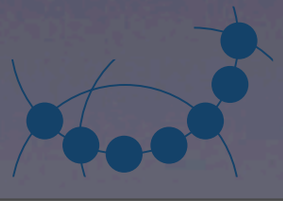
LINDAT CLARIN



PML Framework

- Multi-layered typed XML data format
- OS independent tree editor TrEd
- PML-TQ Tree Query Language and Engine(s)
- Used in development of Prague Dependency Treebank, but tested on more than ten different treebanks

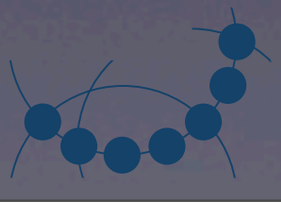
LINDAT CLARIN



Tree Query Language

- Selecting all occurrences of nodes with given properties and in given relations (tree topology, cross-referencing, surface ordering, etc.)
- (Un)bounded iteration of relations (e.g. descendant{1,3}, coref_gram.rf{1,}, sibling{-1,1}, etc.)
- Multi-layered or aligned treebanks with structured attribute values
- Quantified or negated subqueries (e.g. "find all clauses with exactly three objects but no subject")

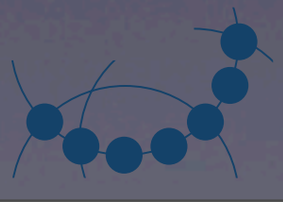
LINDAT CLARIN



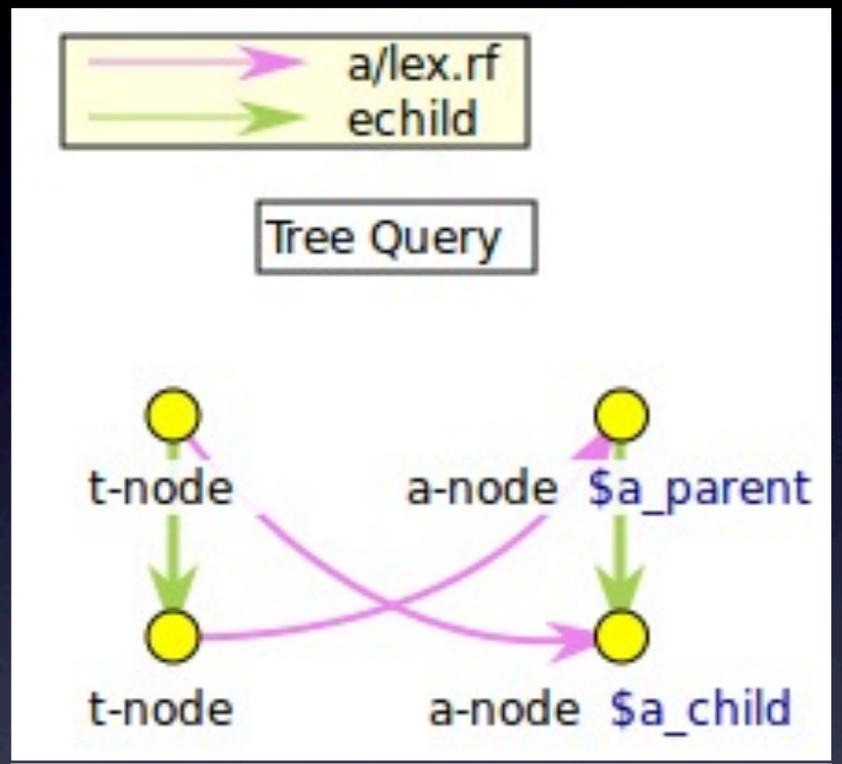
Tree Query Language II

- References (e.g. "find a parent and child that have the same case but different number")
- Natural textual and graphical representation of the query
- Sublanguage for postprocessing and generating reports (filtering, grouping, aggregating, and sorting)
- Regular expressions, basic arithmetic and string operations

LINDAT CLARIN

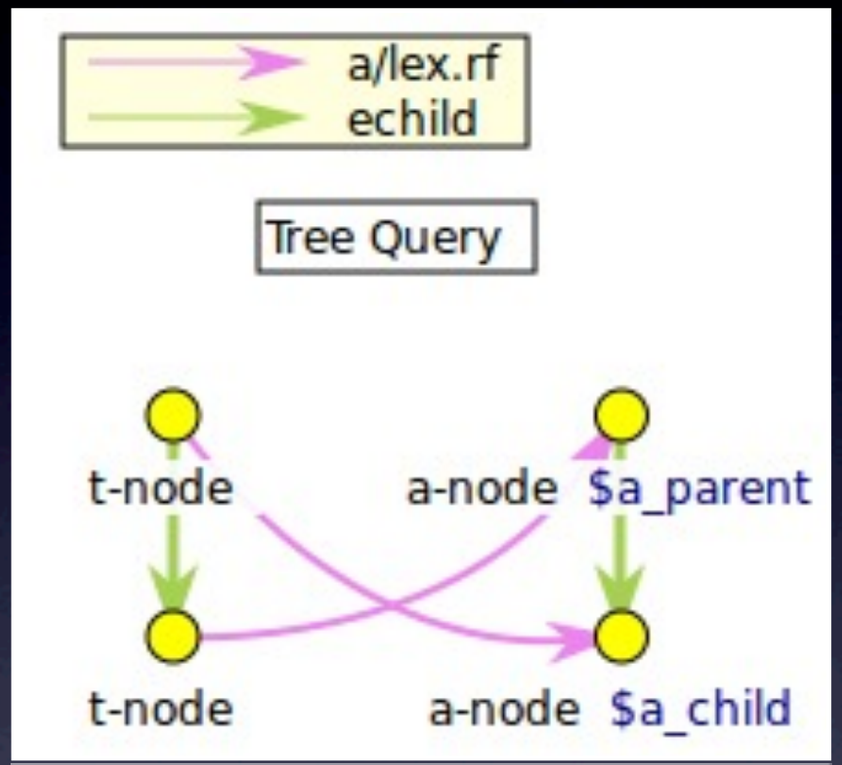


Reversed Dependency



Reversed Dependency

```
    t-node [
     a/lex.rf $a_child,
     echild t-node [ a/lex.rf
$a_parent ]
];
a-node $a_parent := [
     echild a-node $a_child := [   ]
];
```



Distribution of MWEs by their types

```
t-root $r := [ member mwes [] ]
>> for $r.mwes/type
give $1, count() sort by $2 desc
```

Distribution of MWEs by their types

```
t-root $r := [ member mwes [] ]
>> for $r.mwes/type
give $1, count() sort by $2 desc
```

lexeme	44304
person	16708
institution	12243
number	9018
object	7519
time	6549
location	4434
foreign	256
address	230
biblio	43

LINDAT CLARIN

Grammar extraction

Tiger corpus

```
nonterminal $p := [ * $ch := [   ] ]
>> give $p, $p.cat,
        first_defined($ch.cat,$ch.pos),
        lbrothers($ch)
>> give $2 & " → "
        & concat($3," " over $1 sort by $4)
>> for $1 give count(),$1 sort by $1 desc
```

Grammar extraction

Tiger corpus

```
nonterminal $p := [ * $ch := [  ] ]
>> give $p, $p.cat,
        first_defined($ch.cat,$ch.pos),
        lbrothers($ch)
>> give $2 & " → "
        & concat($3," " over $1 sort by $4)
>> for $1 give count(),$1 sort by $1 desc
```

39126	NP → ART NN
20796	PP → APPR ART NN
19248	NP → ART ADJA NN
13432	PN → NE NE
12897	NP → ART NN PP
12282	PP → APPR NN
12018	NP → ART NN NP
11384	PP → APPR ART ADJA NN
10318	PP → APPRART NN
10244	PP → APPR NE
9488	PP → APPR ART NN NP
9072	NP → ADJA NN
8790	PP → APPR ADJA NN

LINDAT CLARIN

Word order Typology

CoNLL 2009 German training data (Tiger corpus)

```
   node $p := [
    substr(pos, 0, 1) = "V",
    ? child node $ch := [
        deprel in { "SB", "OA", "OC", "OA2", "OP" }
] ]
>> give $p.xml:id,
        if($p = $ch,
           if($p.deprel = "ROOT", "V", "v"),
           substr($ch.deprel, 0, 1)),
        $ch.order
>> give distinct $1, concat($2, "" over $1 sort by $3)
>> give substitute($2, "([OS])\\1+", "\\1", "g")
>> filter ($1 ~ "O" and $1 ~ "S")
>> for $1
   give $1, if($1 ~ "V", count(), 0), if($1 ~ "V", 0, count())
>> give $1, if($1 ~ "V", 0, 1),
        percnt(ratio($2 over all) + ratio($3 over all), 3)
   sort by $2, $3 desc
>> give $1, $3 & " %"
```

Main clause	Number of occurrences	Dependent clause	Number of occurrences
SVO	52.336 %	SOv	63.942 %
VSO	33.031 %	SvO	19.235 %
OVS	10.261 %	vSO	9.419 %
VOS	2.903 %	OSv	5.128 %
SOV	0.511 %	OvS	0.922 %
OVSO	0.423 %	vOS	0.542 %
VOSO	0.297 %	SOvO	0.313 %

Parallel Treebanks

Prague Czech-English Dependency Treebank 2.0

```
t-node [
    t_lemma = "force",
    alignment/counterpart.rf t-node [
        t_lemma != "síla"
    ]
]
```