Nfickle: Improved Scanner Generators for Tcl

Detlef Groth
Potsdam University,
Germany

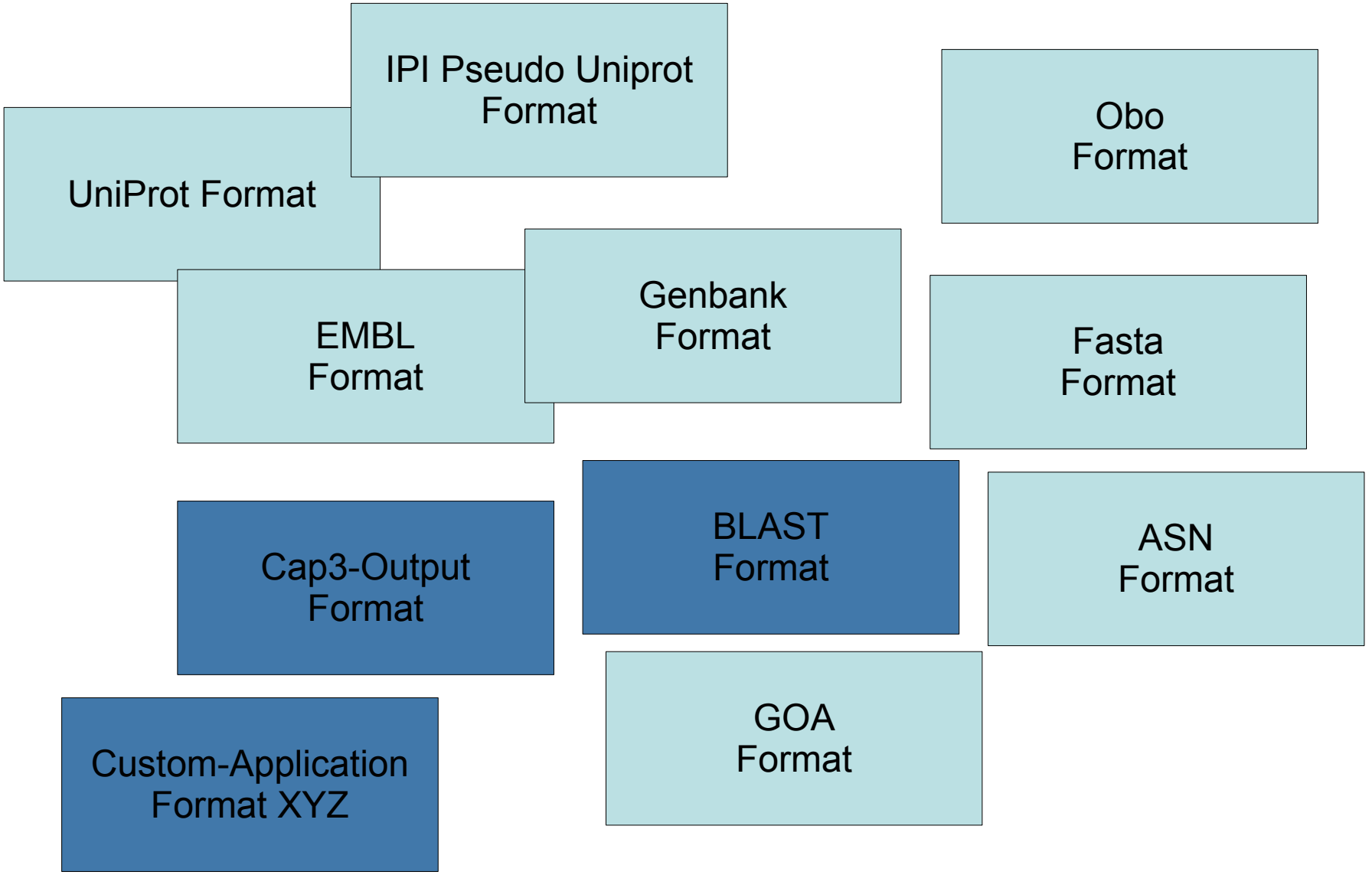
EuroTcl 2008
Strasbourg, France

Outline

- Bioinformatics data problem, Speed matters
- Scanners and Parsers in Bioinformatics
- Bioscanners project
- Tcl Scanner Zoo
  - Fickle
  - Ifickle
  - Nfickle
- Outlook, Conclusions

Bioinformatics Data

Ever increasing amount on data and data formats ….



BLAST-SAMPLE

```
BLASTP 2.2.17 [Aug-19-2007]
Reference: Altschul, Stephen F., Thomas L. Madden, Alejandro A. Schaffer,
Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman (1997),
"Gapped BLAST and PSI-BLAST: a ...
...
Query= tr|Q5C7W3|Q5C7W3_SCHJA SJCHGC09430 protein (Fragment) -
Schistosoma japonicum (Blood fluke).
         (118 letters)
Database: sprot_human.dat.go.fasta
           10,218 sequences; 5,961,085 total letters

Searching.....................................................done
...                                                              Score    E
Sequences producing significant alignments:                     (bits) Value

MTR1L_HUMAN  Melatonin-related receptor (G protein-coupled recep...    32   0.048
CS062_HUMAN  Uncharacterized protein C19orf62.  Homo sapiens (Hu...    29   0.28
ZN174_HUMAN  Zinc finger protein 174 (AW-1) (Zinc finger and SCA...    28   0.76
RAGE_HUMAN  Advanced glycosylation end product-specific receptor...    27   1.9
```

Bio*Zoo

- BioPerl
- BioJava
- BioSQL (only (postgresql and | mysql)
- BioPython
- (BioTcl3.kit)

- quite complex and quite large for easy tasks
- Sometimes behind new formats and new features
- Sometimes difficult to maintain different versions

Parsers / Scanners

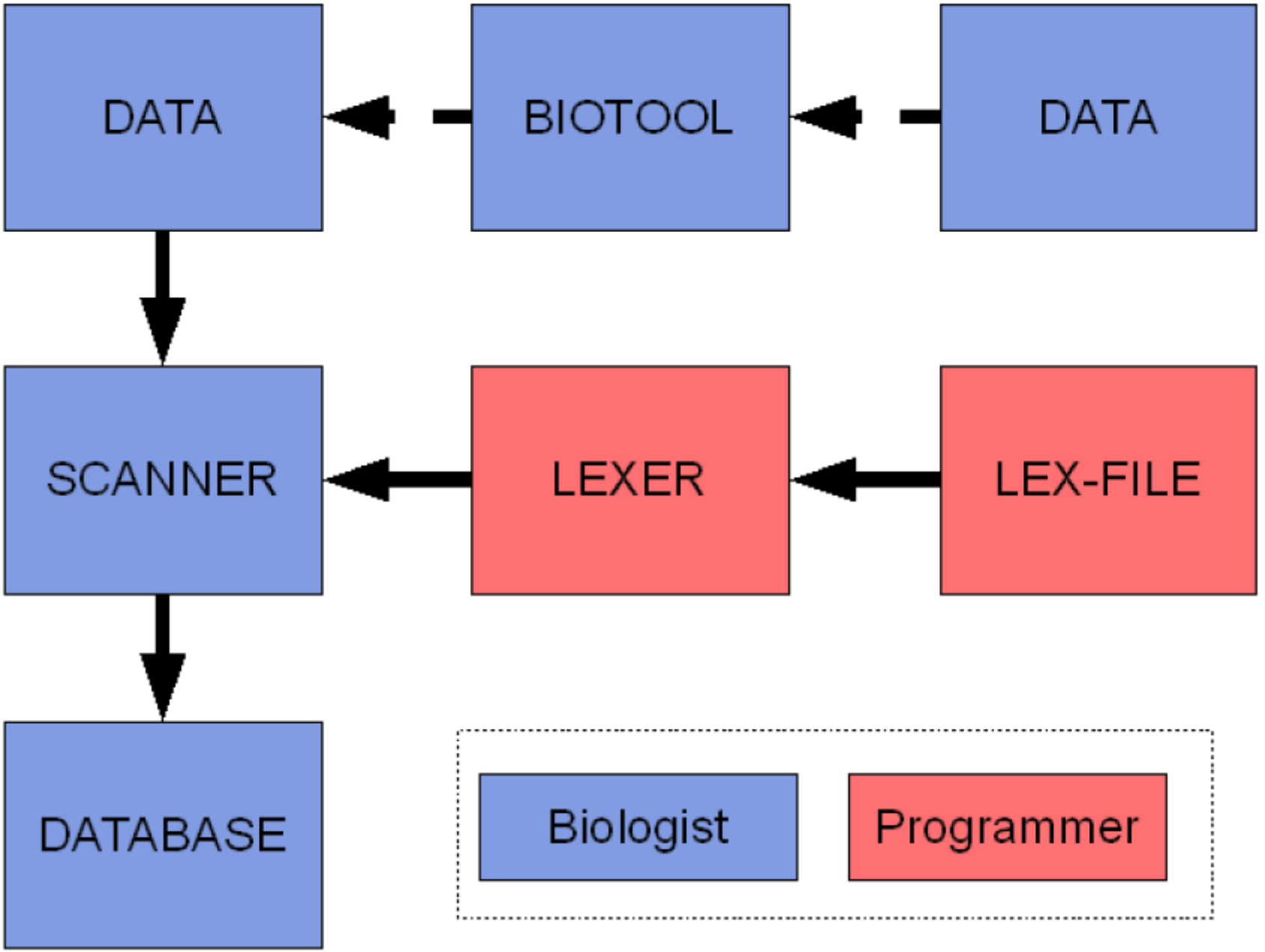
- Data parsers has been invented many times
- Large libs for programmers only

```
use strict;
use Bio::SearchIO;
my $in = new Bio::SearchIO(-format => 'blast',
                           -file   => 'report.bls');
while( my $result = $in->next_result ) {
  while( my $hit = $result->next_hit ) {
   while( my $hsp = $hit->next_hsp ) {
    if( $hsp->length('total') > 50 ) {
     if ( $hsp->percent_identity >= 75 ) {
      print "Hit= ",        $hit->name,
            ",Length=",     $hsp->length('total'),
            ",Percent_id=", $hsp->percent_identity, "\n";
     }
    }
   }
  }
}
```

- Bioscanners parse data and emit SQL code
- Small applications for biologist
- Development is for developers

```
$ BlastScanner report.bls |
sqlite3 report.sqlite3
$ sqlite3 “select hit_id,
identity, from hits where identity
>= 75 and hit_id like '%_HUMAN'”
report.sqlite3
```

Scanner Pipeline



Scanner Generators

- Advantages
  - Apps are easier to code and maintain
  - Standalone applications with simple installation
  - Easy to keep different versions
  - Better performance
- Compiled:
  - C: flex, re2c
  - Pascal: plex
  - Java: jflex
- Scripted:
  - Tcl: yeti, fickle, ifickle
  - Perl: Parse::Flex, Parse::Lex

Scanner-Skeleton

```
%{
package loading, declarations
variable declarations
%}

%option value
%x STATE

%%
<STATE>regexp { action }
regexp { action }
%%
user code, might be main function
```

Sample Wc-Scanner

```
$ ifickle.tcl wc.fcl
$ wc.tcl wc.fcl
      25     102     577 wc.fcl
```

```
%{
#!/usr/bin/tclsh8.4
public variable nline 0
public variable nword 0
public variable nchar 0
%}
%buffersize 1024
%%
\n          { incr nline; incr nchar ; }
[^ \t\n]+ { incr nword; incr nchar $yyleng ;}
.           { incr nchar;}
%%
```

Wc-user code

```
if {[llength $argv] == 0} {
   puts stderr "usage wc.tcl inputfile"
   exit 0
}
if {[catch {open [lindex $argv 0] r} yyin]} {
    puts stderr "Could not open [lindex $argv 0]"
    exit 0
}

set sc [wc #auto -yyin $yyin]
$sc yylex
puts [format "%7d %7d %7d %s" [$sc cget -nline]  \
   [$sc cget -nword] [$sc cget -nchar] \
   [lindex $argv 0]]
close $yyin
```

wc.tcl - code

```
# verbatim code
itcl::class scannername {
    # special verbatim code
    method yylex {} {}
}
itcl::body scannername::yylex {
    while {1} {
       if {[string length $buffer] - $yyindex < $buffersize]} {
          set buffer [string range $buffer $yyindex end]
          append buffer [read $yyin $buffersize]
       }
       set rule -1
       set yyleng 1
       if {[regexp -start $yyindex pattern1 $buffer yytext]} {
           set rule 1
           set yyleng [string length $yytext]
       }
       incr yyindex $yyleng
       switch -- $rule {
           1 {
             action 1
           }
       }
    }
}
# user code
```

Performance considerations BlastScanner

Mode	small	medium	large	memory(Mb)
blast-biojava	2.019	8.055	err	1054
blast-bioperl	4.026	47.822	nd	21
blast-flex	0.051	0.567	4.237	20
blast-jflex	0.607	1.906	8.532	863
blast-re2c	0.027	0.283	2.094	19.7
blast-zerg	0.017	0.185	1.331	6.5
blast-tclkit851	35.480	nd	nd	10.1

Parsing time in seconds
Small: 1 Mb Blastfile
Medium: 14 Mb Blastfile
Large: 140 Mb Blastfile

Re2c scanners

- **Easy compilation, standalone C code, no library required**

```
$ re2c BlastScanner.re2c > BlastScanner.re2c.c
$ gcc -o BlastScanner-Linux-x86 BlastScanner.re2.c
$ ./BlastScanner-Linux-x86
usage: BlastScanner-Linux-x86 --infile=<infile> [--
prefix=<prefix>]
```

Re2c BlastScanner

- ~20kb executable, crossplatform C-code
- translates ~20Mb Blastfile into cross database sql code in ~2 seconds

```
$ ls -l BlastScanner*
-rw-r--r--    1 dgroth   Administ    29956 May  8 13:27
BlastScanner-Darwin-x86
-rw-r--r--    1 dgroth   Administ    22296 May  8 12:52
BlastScanner-Linux-x86
-rw-r--r--    1 dgroth   Administ    24312 Apr 25 10:17
BlastScanner-Linux-x86_64
-rwxr-xr-x    1 dgroth   Administ    26624 May 19 04:27
BlastScanner-Windows-x86.exe
```

```
$ ./BlastScanner-Windows-x86.exe --infile
../sample_human_sprot.blastp --prefix none | head
begin;
create table blastdata (bkey VARCHAR(50), bvalue VARCHAR(256));
create table queries (query_id VARCHAR(50),query_length
INT,position BIGINT);
create table hits (hit INT,query_id VARCHAR(50),hit_id
VARCHAR(50),score INT,evalue DOUBLE,identities
ositives INT,gaps INT,frame INT,query_start INT,query_end
INT,subject_start INT,subject_end INT);
create table subjects (hit_id VARCHAR(50),description
VARCHAR(256),length INT);
insert into queries (query_id,query_length,position) values ('tr|
Q5DDY0|Q5DDY0_SCHJA',221,0);
insert into blastdata (bkey,bvalue) values
('db','sprot_human.dat.go.fasta');
insert into blastdata (bkey,bvalue) values ('typ','BLASTP');
insert into blastdata (bkey,bvalue) values ('version','2.2.17');
insert into queries (query_id,query_length,position) values ('tr|
Q5C7W3|Q5C7W3_SCHJA',118,974);
```

Why not pure C

- Coding in C can be a pain (at least for me), code change needs recompilation on all platforms
- Self updating starkits using critcl can automatically do this for us
- Starkits can contain different applications at once
- Tcl starkits can embed a webserver, javascript code, ...

Why Tcl for Scanner Generation

- Most dynamic language
- Most mature language
- Most stable language (stubs!!)
- Almost most cross platform
- Easy to integrate C-code
- Clean API with sub commands

Current Tcl-Scanner generators

- Tclish-declaration
  - Tclex – C code unmaintained
  - Yeti – On the fly generation, but all input is scanned at once
- Flex-like input, eating up char by char
  - Fickle (Jason Tang) 2002-2004, proc only standalone
  - Ifickle Port by me 2006, Itcl-class

Java/Jflex -> Tcl/Ifickle

- Porting a complete BlastScanner from Java/Jflex to Tcl/Ifickle has taken less than 2 hours
- But code was 70 times slower
  - Stdout buffering, buffersize ?b
- As shown two years ago original fickle is not faster than ifickle
- How to achieve performance improvements?

Ifickle Problems

- Itcl-classes only
- Very low performance
- Does traditional parsing for scanner generation and directly puts out the code, thereof making extensions difficult
- Fickle / Ifickle two branches with the same goal
- A lot of duplicated code

Next Fickle (nfickle) Solutions

- Should support any class system, proc or namespaces
- Easier scanner generation using templates, inheritence
- Scanner must be able to generate itself (?)
- Possible future:
  - Better performance by rewrite and integration of C-code via critcl
  - Easier C-code writing by creating C-code

Without-Templates (ifickle)

```
    if $::headers {
        append ::output "# ${::P}_FLUSH_BUFFER flushes the
scanner's internal buffer so that the
# next time the scanner attempts to match a token, it will
first
# refill the buffer using ${::P}_INPUT.
#   -- from the flex(1) man page\n"
    }
    append ::output "itcl::body
${::classname}::${::P}_FLUSH_BUFFER \{\} \{
    set ${::p}_buffer \"\"
    set ${::p}index 0
    set ${::p}_done 0
\}
"
```

With-Templates (nfickle)

```
<% set scanner(verbatim) %>
namespace eval <% set scanner(name) %> {
    variable state INITIAL
    variable states [list INITIAL <% join $scanner(states) %>]
}
proc <% set scanner(name) %>::BEGIN {nstate} {
    variable state
    variable states
    if {[lsearch $states $nstate] == -1} {
        return -code error "unknown state $nstate"
    }
    set state $nstate
}
proc <% set scanner(name) %>::ECHO {} {
    uplevel 1 {
        puts -nonewline $yytext
    }
}
```

Scanner should generate itself ?

- Chicken and Egg problem
- Difficult to build Checiken and Egg at the same time -> nfickle1 failed ($yytext vs \$yytext problem)
- Nfickle2 classical scanner / parser
  - Good, focus should be development of scanners not the scanner generator !
  - 277 TCL-LOC
  - Samples
    - Namespace scanner
    - Itcl scanner

Ifickle/Nfickle

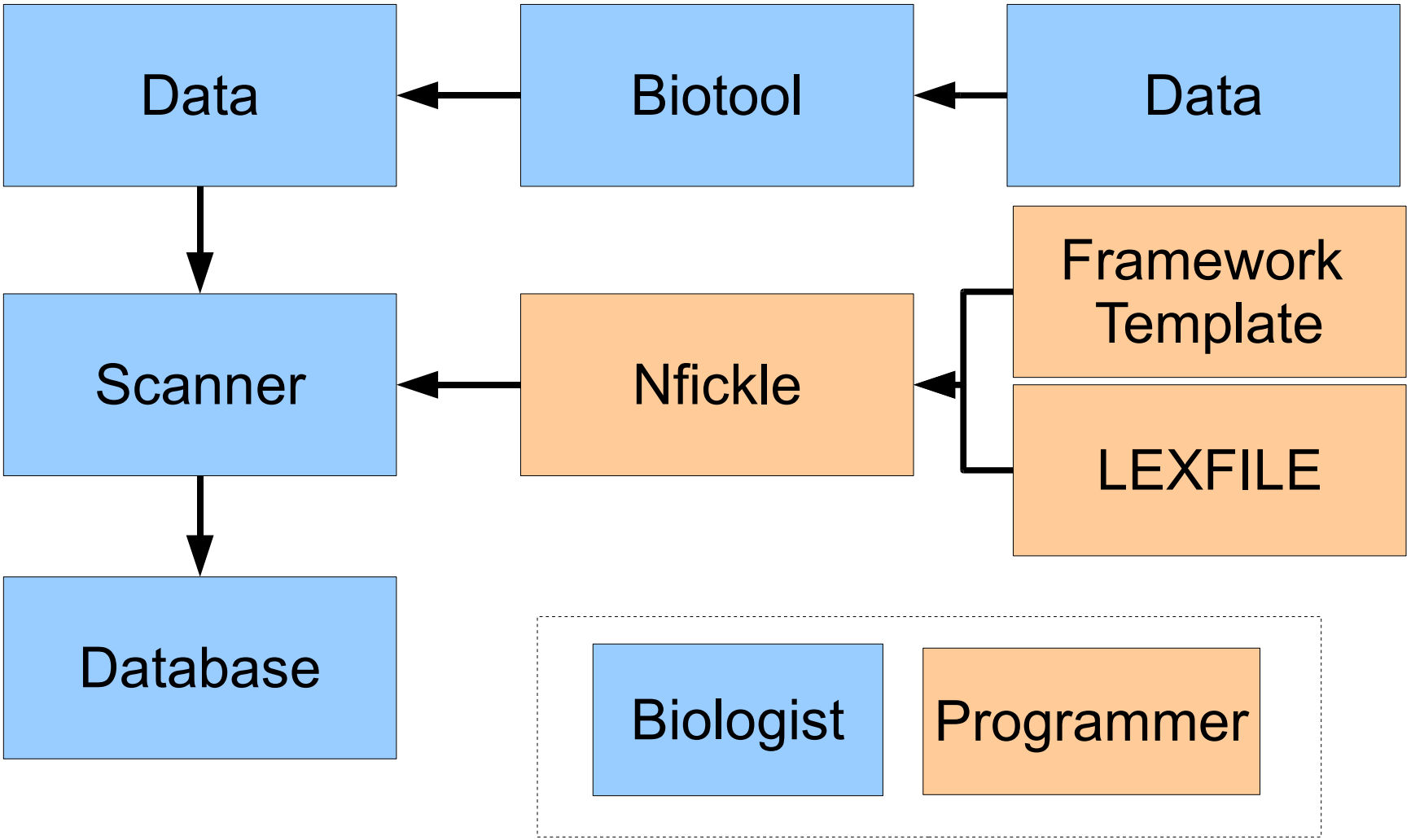
- Ifickle (GPL)
  - 777 LOC
  - More features
  - Only itcl classes
  - Slower
  - Difficult to extend
- Nfickle (BSD)
  - 264 LOC
  - Twice as fast as Ifickle
  - Most important features
    - Stack, yyless, BEGIN
    - Buffersize, name
  - Extra features: init, eof
  - Itcl, Nsp, ...

Nfickle WC-scanner (nsp and itcl)

```
%{
#!/usr/bin/tclsh8.4
%}
%init{
variable nline 0
variable nword 0
variable nchar 0
%init}
%buffersize 1024
%%
\n          { incr nline; incr nchar ; }
[^ \t\n]+ { incr nword; incr nchar $yyleng ;}
.           { incr nchar;}
<<EOF>> {
    puts "$nline $nword $nchar [lindex $::argv 0]"
}
%%
```

User Code belongs to the template !!

Nfickle-Pipeline



Speed issus

- **Still to slow, requires C speedup with Critcl**

```
$ /c/Tclkit/tclkitsh850.exe nfickle2.tcl wc2.fcl  -template
ScannerTemplate.exp > wc2.tcl

dgroth@GLAUKE /c/home/dgroth/mycvs/mytcl/goblet1/scanner
$ date; /c/Tclkit/tclkitsh850.exe wc2.tcl
../sample_human_sprot_trembl.blastp ; date
Fri Jun  6 04:09:00 GMT 2008
113631 565866 5063231 ../sample_human_sprot_trembl.blastp
required 26 seconds
Fri Jun  6 04:09:26 GMT 2008
```

Question: Could we not generate re2c-C code directly with Tcl ?

```
yy2:   yyaccept = 0;
    yych = *(YYMARKER = ++YYCURSOR);
    switch(yych){
    case 'B': goto yy172;
    case 'T': goto yy173;
    default:  goto yy3;
    }
yy3:
#line 96 "CBlastScanner-make.re"
{ goto std ; }
#line 49 "<stdout>"
yy4:   yyaccept = 0;
    yych = *(YYMARKER = ++YYCURSOR);
    switch(yych){
    case 'L': goto yy162;
    default:  goto yy3;
    }
```

Conclusions

- Nfickle is a from scratch rewritten scanner generator
- Some speed improvements
- By using templates it is easier to maintain and to extend
- Homepage: will be at http://bioscanners.sf.net/
- Todo: Critcl/C speed improvements

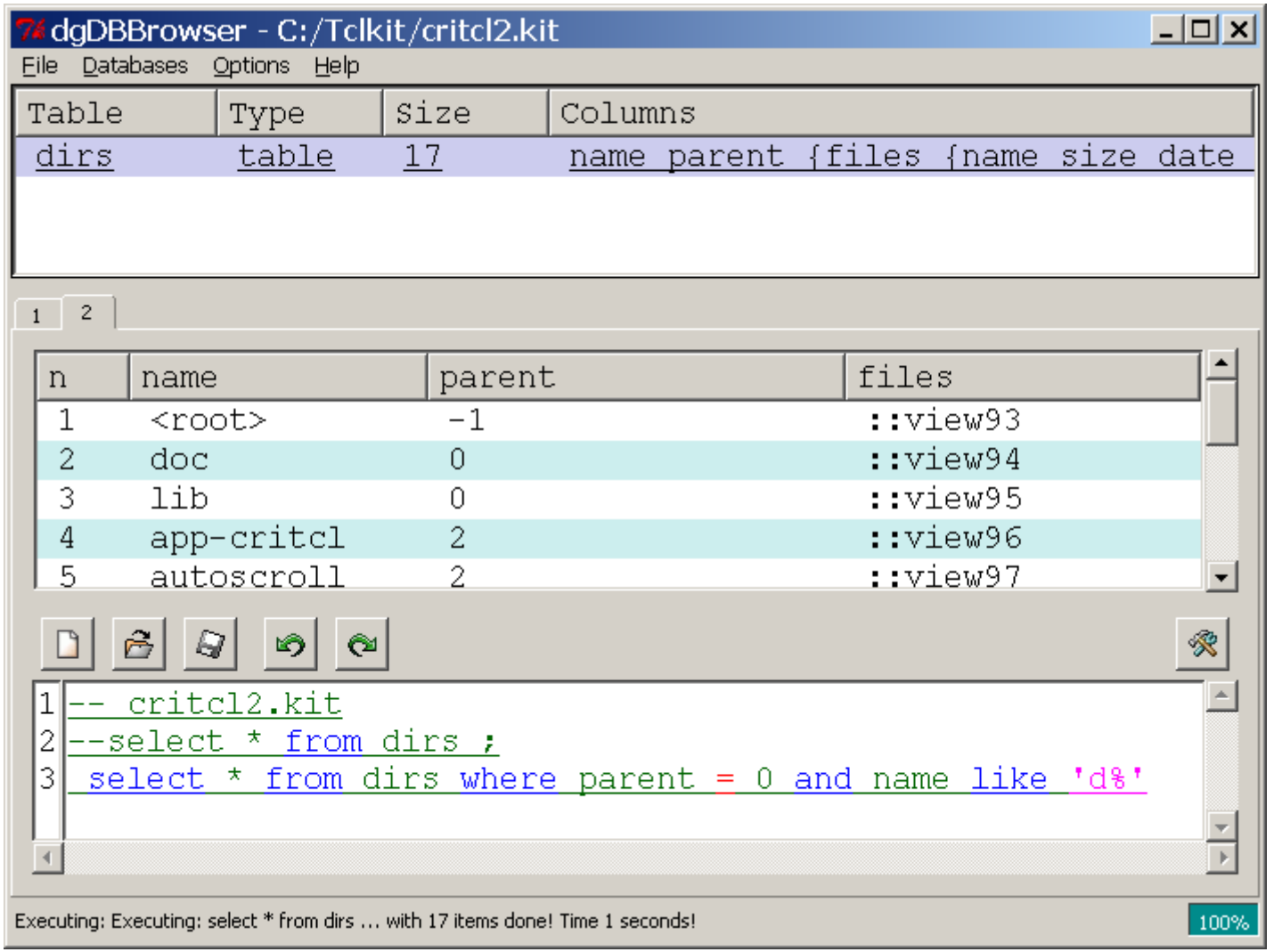
Links

- http://goblet.molgen.mpg.de
  - Tcl based web application for annotating anonymous sequence data, currently uses Re2c-BlastScanner in the bg
- http://bioscanners.sf.net
  - Home of code for various scanners and scanner generators nfickle, ifickle, pllex, biotcl
  - Might be misused for GOblet and dgDBBrowser code

Other Areas

- Web-Development with Tcl, JavaScript
- Statistics with R (MicroArrays)
- Biotcl used internally
- Teaching activities Statistics, Unix and Perl
- Documentation for Tcl8.5.2 dgHelpBrowser
- Databases (dgDBBrowser)
- 800m, 1500m
- Look for DDG in the wiki

dgDBBrowser



Acknowledgements

- Jean Claude Wippler, Equi4
- Joachim Selbig, Uni Potsdam
- Steffen Hennig, Imagenes GmbH
- EuroTcl 2008 organizers