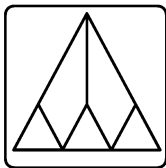


Introducing the Stratego Libraries

Martin Bravenboer



Stratego/XT

Delft University of Technology
Department of Software Technology

Stratego User Days 2006
November 30

current practice: xtc for sglr, ast2abox, ast2text, etc

```
xtc-transform(!"sglri",  
  !["-p", <xtc-find> "parse-testsuite.tbl"])
```

```
xtc-transform(!"Vis-amb")  
; xtc-transform(!"ast2abox",  
  !["-p", <xtc-find> "visamb.pp.af",  
    "-p", <xtc-find> "Sdf2.pp.af"])  
; xtc-transform(!"abox2text")
```

```
xtc-transform(!"stratego-parenthesize")  
xtc-transform(!"stratego2abox")  
xtc-transform(!"abox2text")
```

current practice: xtc for sglr, ast2abox, ast2text, etc

- various sources of overhead
 - file access
 - process creation
 - aterm (de)serialization
 - aterm allocation
 - parse tables
- command execution not portable
 - `fork` and `exec` not ANSI C
 - `system` undefined
 - `mkstemp` source of various issues
- packages not relocatable
 - programs and libraries not self-contained
 - option: relocater for xtc repositories



well, yes and no

- good: late binding, repositories
- good: compose transformation tools

yes, switching to libraries is a compromise

- **libstratego-lib**
stratego standard library
- **libstratego-xtc**
invoking xtc tools
- **libstratego-sglr**
for parsing and implode-asfix
- **libstratego-gpp**
for pretty-printing
- **libstratego-rtg**
for format checking using rtg
- **libstratego-tool-doc**
for attractive `-help`, `-version`, `-about`

available as separate package

- self-contained, portable
- native Microsoft Windows binaries (not just Cygwin)
- possible to implement portable transformation tools



Source distribution

- [stratego-libraries-0.17M1pre15941.tar.gz](#) (285K)

This source distribution requires that the following packages are installed:

- [aterm-2.4.2.tar.gz](#)



Binary archive for Microsoft Windows

- [stratego-libraries.zip](#) (5578139 bytes; MD5 hash: 40101010101010101010101010101010)



RPM for Fedora Core 3

- `php-front-0.1pre287-1.i386.rpm` (2664874 bytes; MD5 hash: 2d6a14c0957d7c39d)
- `php-front-0.1pre287-1.src.rpm` (3746241 bytes; MD5 hash: 42e0e43aa5efd5c95fc)

This RPM requires that the following packages are also installed:

- `aterm-2.4.2-1.i386.rpm`
- `sdf2-bundle-2.3.4pre15345-1.i386.rpm`
- `strategox-0.17M3pre15898-1.i386.rpm`



Binary archive for Microsoft Windows

- `php-front.zip` (9742091 bytes; MD5 hash: a110807c8dff0e9f987fbe2dbff18b05)

```
parse-php4.exe, parse-php5.exe, libphp-front.dll,  
libstratego-sglr.dll, libstratego-lib.dll,  
libstrateg-runtime.dll, libATerm.dll
```



great, but how do I use the stratego libraries?

example module

```
module sample
imports libstratego-lib libstratego-sglr
```

compile using standalone strc

```
$ strc -i sample.str -la stratego-sglr -la stratego-lib
```

compile using Makefile.am using AutoXT

```
sample_LDADD = \  
  $(STRATEGO_SGLR_LIBS) $(STRATEGO_LIB_LIBS) $(STRATEGO_RUNTIME_LIBS)  
  
STRCFLAGS = \  
  $(STRATEGO_SGLR_STRCFLAGS) $(STRATEGO_LIB_STRCFLAGS)
```

```
main =
  <parse-java-string> "class Foo {}"

parse-java-string =
  where(tbl := <java-tbl>)
  ; finally(
    parse-string(|tbl, "TypeDec")
    , <close-parse-table> tbl
    )

java-tbl =
  <ReadFromFile; open-parse-table> "Java-15.tbl"
```

⇒ parse table can be included in programs and libraries

```
main =
  where(tbl := <java-tbl>)
    ; finally(parse(|tbl), <close-parse-table> tbl)

parse(|tbl) =
  ![ <parse-string(|tbl, "TypeDec")> "class Foo {}"
    , <parse-string(|tbl, "TypeDec")> "interface Bar {}"
    , <parse-string(|tbl, "TypeDec")> "enum Fred {}"
  ]

java-tbl =
  <ReadFromFile; open-parse-table> "Java-15.tbl"
```

```
parse-string(|tbl)
parse-string(|tbl, start-symbol)
parse-string(|tbl, start-symbol, path)

parse-string-pt(|...)

parse-stream(|...)
parse-stream-pt(|...)

parse-file(|...)
parse-file-pt(|...)

parse-xtc-file(|...)
parse-xtc-file-pt(|...)
```

```
module pp-sample
imports libstratego-lib libstratego-gpp
strategies

main =
  io-stream-wrap(
    ?(<read-from-stream>, fout)
    ; pretty-print(|fout)
  )

pretty-print(|stream) =
  ast2box(|<parse-pptable-file> "Exp.pp")
  ; box2text-stream(|80, stream)
```

```
[ Plus -- H hs=1 [_1 "+" _2],
  Var -- _1 ]
```

- pp-table parse table in libstratego-gpp
- pretty-print table can be included in programs and libraries

using an external pretty-printer

```
module pp-sample
imports libstratego-lib libstratego-gpp libphp-front
strategies

main =
  <parse-php-string> "<?php echo 'Hello world'; ?>"
  ; pretty-print

pretty-print =
  pp-php5-to-abox
  ; box2text-string(|80)
```

in combination with Stratego-Box

```
module pp-sample
imports libstratego-lib libstratego-gpp
strategies

main =
  io-stream-wrap(
    ?(<read-from-stream>, fout)
    ; pretty-print(|fout)
  )

pretty-print(|stream) =
  topdown(try(PrettyPrint))
  ; box2text-stream(|80, stream)

PrettyPrint :
  Plus(e1, e2) -> H hs=1 [ ~e1 "+" ~e2 ]

PrettyPrint :
  Var(s) -> H hs=0 [ s ]
```

check that current term has the format of the given RTG

```
module rtg-sample
imports libstratego-lib libstratego-rtg
strategies

main =
  io-wrap(
    where(rtg := <parse-rtg-file; rtg-normalize> "Exp.rtg")
      ; rtg-format-check(true, true | rtg)
    )
```

```
rtg-format-check(|rtg) =
  rtg-format-check(false, true | rtg)

rtg-format-check(report-errors, fail-on-error | rtg ) = ...

parse-rtg-file = ...
```

```
module tool-doc-sample
imports libstratego-lib libstratego-tool-doc
strategies

main =
  io-wrap(fail, usage, about, id)

usage =
  <tool-doc>
  [ Usage("tool-doc-sample [OPTIONS]")
  , Summary("Does something very cool.")
  , OptionUsage()
  , AutoReportBugs()
  ]

about =
  <tool-doc>
  [ AutoProgram()
  , Author(Person("Martin Bravenboer", "martin@cs.uu.nl"))
  , GNU_LGPL("2002-2006", "Martin Bravenboer <martin@cs.uu.nl>")
  ]
```

```
$ parse-php4 --help
Usage: parse-php4 [OPTIONS]
```

Summary:

Parses a PHP4 source file to an abstract syntax tree in the ATerm format.

Options:

<code>-r --release r</code>	Use a specific release, either 4 or 5. [4]
<code>-s --start-symbol s</code>	Start parsing with symbol s [Document]
<code>--preserve-comments</code>	Preserve source code comments as annotations of the abstract syntax tree. [off]
<code>--preserve-positions</code>	Preserve source code positions in the input file as annotations of the abstract syntax tree. [off]
<code>-i f --input f</code>	Read input from f
<code>-o f --output f</code>	Write output to f
<code>-b</code>	Write binary output
<code>-S --silent</code>	Silent execution (same as <code>--verbose 0</code>)
<code>-h -? --help</code>	Display usage information
<code>--about</code>	Display information about this program
<code>--version</code>	Same as <code>--about</code>

```
$ parse-php4 --about
parse-php4
  Package   php-front
  Version   0.1
  Revision  288
```

Author:

* Eric Bowers <eric@bowers.info>

License:

Copyright (C) 2006 Eric Bowers <eric@bowers.info>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. ...



fabulous, but how do I create my own library?

- **library**

```
module foo
  imports libstratego-lib

  strategies

  add2 = <add> (1, 2)
```

- **program**

```
module bar
  imports libfoo

  strategies

  main = add2
```

standalone strc from command-line

```
$ strc -i foo.str --library -la stratego-lib
```

lots of commands are being invoked

```
$ ls only useful files
```

```
foo.str libfoo.rtree libfoo.c libfoo.la libfoo.so libfoo.a
```

```
$ strc -i bar.str -la libfoo.la
```

lots of commands are being invoked

```
$ ./bar
```

```
3
```

more advanced use

- want to install at a different location

```
$ strc -i foo.str --library -la stratego-lib --libdir /usr/lib  
lots of commands
```

- don't want strc to run as root

```
cp: cannot create regular file '/usr/lib/libfoo.so': Permission denied
```

```
$ strc -i foo.str --library --libdir /usr/lob --disable-install  
lots of commands
```

```
[ strc | info ] Don't forget to install the library using:  
  libtool --mode=install cp libfoo.la /usr/lib/libfoo.la  
some more commands
```

- strc knows nothing about platform-specific linking idiosyncrasies: powered by libtool

Makefile.am: Automake + Makefile.xt

```
phpfrontlib = list str files

lib_LTLIBRARIES = libphp-front.la

libphp_front_la_SOURCES = libphp-front.c

libphp_front_la_CPPFLAGS = \
    $(STRATEGO_LIB_CFLAGS) $(STRATEGO_RUNTIME_CFLAGS) $(ATERM_CFLAGS)

libphp_front_la_LIBADD = \
    $(STRATEGO_SGLR_LIBS) $(STRATEGO_GPP_LIBS) \
    $(STRATEGO_LIB_LIBS) $(STRATEGO_RUNTIME_LIBS) $(ATERM_LIBS)

libphp_front_la_LDFLAGS = -avoid-version -no-undefined

libphp-front.rtree libphp-front.c : $(phpfrontlib)
    @$(STRC)/bin/strc -c --library -i $< -o libphp-front.rtree $(STRINCLUDES)
    rm libphp-front.str
```



hmmm, but how does that work on my OS?

goal: make it possible to write very portable transformation tools and libraries using Stratego

- three standards: ANSI C, POSIX, POSIX+XSI
- very portable usually means ANSI C
- Stratego/XT uses POSIX and XSI specific functions at many places
- Stratego/XT is not ANSI C and will not be Real Soon Now
- new Stratego Libraries are portable
 - only depend on aterm library
 - variants: `--with-std={C99|POSIX|POSIX+XSI}`
 - C99 variant can be installed virtually everywhere

restrictions

- don't invoke any tools, only libraries
 - maybe portable xtc variant in the future
- don't use strategies not supported in C99 variant
 - don't use `system/posix` and `system/posix-xsi`
 - for variability, import `libstratego-lib`
- don't use `xtc-transform`
 - `xtc i/o` is ok
- don't use any tools on installation
 - depend on `aterm` and `stratego-libraries` only
 - disable `xtc` registration
- don't use undefined symbols in libraries

AutoXT support

- `XT_USE_BOOTSTRAP_XT_PACKAGES`
- adds configure option `--enable-bootstrap`
- adds configure option `--enable-xtc`
- disables xtc register in `Makefile.xt`
- checks for different packages, depending on config

compile using MinGW

- MinGW: GNU toolchain for Microsoft Windows
- runtime uses Microsoft Windows runtime libraries
 - no POSIX layer
- does not require GPL (or even open source)
- use MinGW from MSYS or Cygwin
- Nix packages supports MinGW + MSYS
 - used in Nix buildfarm
 - solves all kinds of PATH issues
- very confusing packaging
 - maybe we will start MinGW buildfarm
 - just contact us if you need help

stratego-shell supports loading of native libraries

```
$ stratego-shell
stratego> :showloaded
/pkg/strategoxt/2006-11-15-23-13/lib/libstratego-lib.so

stratego> :load lib/libphp-front.la

stratego> :showloaded
/pkg/php-front/lib/libphp-front.so
/pkg/strategoxt/2006-11-15-23-13/lib/libstratego-lib.so

stratego> import share/php-front/libphp-front.rtree

stratego> <parse-php-string> "<?php echo 'hello world' ?>"
Document( ... )
```

dynamic loading of stratego libraries

- based on libtool's libltdl
- more generally useful
 - php-sat analysis plugins
 - compiler plugins
 - separate deployment of language extensions
- future: make dynamic loading available to all stratego applications