
XTC Support in the Stratego Shell

Ad Hoc Transformation Tool Composition



Martin Bravenboer

`martin@cs.uu.nl`

Institute of Information and Computing Sciences, University Utrecht, The Netherlands

Component-Based Transformation Systems

- *separate tools* performing clear and identifiable task
- *goals* of component-based software development
 - reuse
 - control complexity
 - improve independently
 - composition without detailed knowledge of internals
- extensively *applied in Stratego/XT*
 - separate transformation tools
 - exchanging structure data
 - standard invocation interface (`stdin/out` and `-i/o`)
 - lots of reuse

Transformation Tool Composition

- *ad hoc*
 - for a specific situation
 - composing all tools ad hoc is not user-friendly
 - typically entered in a shell
- *anticipated*
 - for a likely situation
 - must be deployable
 - typically implemented in a shell script

XTC Concepts

- abstraction from *tool location*
 - invoke external tools: configuration issue
 - solution: bind tools to identifiers
 - invocation of tools using identifier
 - XTC repository stores bindings
- abstraction from *tool invocation interface*
 - XTC library handles actual invocation
- abstraction from *wiring*
 - XTC library connects the tools

Extended XTC Concepts

- new XTC: full abstraction of invocation
 - interpreted programs
 - (java, class, jar, python, sglr tbl)
 - network launch protocol (jnlp)
 - web services (grammar base?, packages?)
 - exchange format (aterm, xml)
- XTC repository: everything is meta data
 - run descriptors
 - contracts (syntax defs or arbitrary)
 - version
 - language of parse tables and syntax defs
 - license, manual
 - ...

XTC Trip

- creating an XT component
`main = io-wrap(s)`
- creating an XT composition
`main = xtc-io-wrap(s)`
- exit an XT composition
`xtc-exit` or `xtc-io-exit`
- just handling input/output
`xtc-io(s)`
- find location
`<xtc-find> "toolid"`
- invoking a tool
`<xtc-command(!"toolid")> args`
- invoking an XT component
`xtc-transform(!"toolid", !args)`

XTC Trip, Scoped Temporary Files

- new XTC temporary file scope
`xtc-temp-files(s)`
- create XTC temporary file
`xtc-new-file`
- current term to new XTC file
`write-to`
- list of strings to new XTC textfile
`print-to`
- read a term from an XTC file
`read-from`
- file operations
`rename-to(!"filename")`
`copy-to(!"filename")`

AutoXT build support for XTC

- default location of XTC repository is in executable
 - defined by Autoconf (`AC_DEFINE`)
- registered by including `Makefile.xt`:
 - `pkgdata_DATA`
 - `bin_PROGRAMS`
 - `libexec_PROGRAMS`
 - `sdfdata_DATA`

XTC Example

```
pp-stratego =
  xtc-io-wrap(pp-stratego-options <+ io-options,

    (where(<get-config> "--abstract")
      <+ xtc-transform(!"parse-stratego")))

; xtc-transform(!"stratego-ensugar")

; (where(<get-config> "--annotations")
  <+ xtc-transform(!"annos-to-term"))

; xtc-transform(!"ast2abox", !["-p",
  <xtc-find> "Stratego-pretty.pp" | <pass-verbose>])
; xtc-transform(!"abox2text", !<pass-verbose>)
)
```

Ad Hoc XTC

- current XTC library targets anticipated compositions
- return to good old shell for ad hoc composition
- shell problems return
 - tool and resource location
 - tool invocation interface
 - no ad hoc rewriting
 - limited to executable files

XTC Shell

- solution: apply XTC concepts in a shell
- solution? `xtc call/get` and existing shell

```
cat foo.str
| xtc call sglr -p `xtc get Stratego.tbl`
| xtc call implode-asfix
```
- solution? Stratego Shell and XTC library
 - XTC library syntax is too verbose

Solution: XTC Support in the Stratego Shell

- *syntax* for XTC tool invocations
 - embed same syntax in Stratego
 - syntax for composition and rewriting
- environment of tool bindings
 - bind individual tools, not all in a directory
 - *store and load an environment* of tool bindings
- by extending the Stratego Shell we get for free
 - ad hoc rewriting in ad hoc compositions
 - a nice shell

Ideas for the XTC Syntax

- value of tool variable `sglr`
 - `&sglr`
 - `&sglr[version = "3.10.2"]`
- tool invocation
 - `&sglr -p &Stratego.tbl -i foo.str`
- tool invocation
 - `&sglr -p &Stratego.tbl -i foo.str`
- tool invocation combinators
 - Stratego strategy combinators
 - parallel execution `invocation || invocation`

Ideas for the XTC Syntax

- import tool bindings stored in an XTC repository

```
import /usr/share/strategoxt/XTC
```

```
import http://services.strategoxt.org/xml-tools
```

- export tool bindings to an XTC repository

```
export /usr/share/strategoxt/XTC
```

- define a tool binding (register)

```
automake : /usr/bin/automake
```

```
sgr-old : &sgr[version = "3.8.0"]
```

```
dir : &ls
```

```
ll : &ls -l
```

XTC Example

```
pp-stratego =
  xtc-io-wrap(pp-stratego-options <+ io-options,

    (where(<get-config> "--abstract")
      <+ xtc-transform(!"parse-stratego"))

    ; xtc-transform(!"stratego-ensugar")

    ; (where(<get-config> "--annotations")
      <+ xtc-transform(!"annos-to-term"))

    ; xtc-transform(!"ast2abox", !["-p",
      <xtc-find> "Stratego-pretty.pp" | <pass-verbose>])
    ; xtc-transform(!"abox2text", !<pass-verbose>)
  )
```

XTC Example with Embedded XTC Syntax

```
pp-stratego =
  xtc-io-wrap(pp-stratego-options <+ io-options,

    (where(<get-config> "--abstract")
      <+ &parse-stratego

; &stratego-ensugar

; (where(<get-config> "--annotations")
  <+ &annos-to-term

; &ast2abox -p &Stratego-pretty.pp <pass-verbose>
; &abox2text <pass-verbose>

)
```


XTC Shell Future

- improve integration with Stratego part of the Stratego Shell
- system for ad hoc tool invocation and rewriting
- aterm output for system tools like `ls`?
- full abstraction of tool invocation
- <insert your idea here>