$U{\cdot}(TP)^2$ Hacker's Guide

Andrew Butterfield

November 21, 2013

Contents

Chapter 1

Introduction

1.1 What is $U{\cdot}(TP)^2$?

$U{\cdot}(TP)^2$ is a theorem-proving assistant for Hoare and He's Unifying Theories of Programming (UTP) [HH98]. It was developed as a tool to support foundational work in the UTP, that is, the development of UTP theories. A user-friendly graphical user-interface (GUI) has been designed into the tool from the start.

1.2 Structure of This Document

This is the Hacker's Guide for $U{\cdot}(TP)^2$. It gives an overview of how the program and documentation are written and organised.

At present this document is a very rough draft.

1.3 Gotchas

This at the moment is a unstructured list of things we feel you really ought to know before you start hacking.

- The datatypes `Expr` and `Pred` in `src/Datatypes.lhs` have "focus" variants used in proofs. These no longer use a "knot-tying" style (see `src/Focus.lhs`). So this gotcha in earlier versions of this guide is no longer an issue.

Chapter 2

Program Structure

2.1 Overview

The sources for $U{\cdot}(TP)^2$ comprise a very large collection of LaTeX sources, and Haskell Literate Scripts, which are themselves also valid LaTeX.

The directory/folder structure has been designed to support ease of development for the main development environment, namely WinEdt/MikTeX/ghci on Windows (currently Windows 8).

2.2 Directory Structure

The top-level contains all the LaTeX master documents, currently including:

UTP2-MAIN.tex $U{\cdot}(TP)^2$ sources and documentation

UTP2-Hacking-MAIN.tex This document

UTP2-Reference-MAIN.tex Reference manual

UTP2-User-Manual-MAIN.tex User Guide.

Almost all other files present at this level should be considered as junk, even if tracked by Mercurial. This will be tidied up at some future date.

Subdirectories are organised as follows:

src
: Haskell source files, as well as MS-DOS batch files (.bat) for building under Windows.

doc
: Mainly LaTeX files giving documentation of various forms, as well as text files to do with installation. It has a couple of sub-directories to manage certain types of documentation:

  formal mainly formal definitions of aspects of the logic

  images images (obviously!)

  papers sources for conference/journal papers about $U{\cdot}(TP)^2$

  styles LaTeX style files (currently ignored).

  screenshots screen shots of the tool in action, arranged in topic sub-directories

batch
: Created by someone from a unix background, to hold .bat files. (Deprecated, unused, will probably disappear).

licence Licensing files.

orphans unwanted and unloved — also likely to vanish.

resource mainly sound and help files.
: The help file (with the long unpronounceable name) has been subsumed into the relevant code, and is no longer required.

test
: test stuff, currently unused, but we will probably flesh this out at some stage.

thlib
: This is where we build $U{\cdot}(TP)^2$ theories to drive and test the development, and most of which will become part of a standard theory library release.

www
: Stuff for the (release) website.

2.3 Literate File Structure

All the Haskell source files are literate scripts (.lhs extension) that are themselves valid LaTeX files, in which the Haskell source is enclosed in `\begin{code}` ... `\end{code}` environments. The `code` environment is defined in the style file `doc/saoithin.sty`.

We do not use “bird-tracks” or `lhs2tex`, nor do we use Hackage/Haddock in any way.

An example of some .lhs source is below:

```
\subsection{\UTP2 Source Example}

\begin{code}
module Example where
import Utilities
\end{code}

\subsubsection{Intro}

We can have a suitably mathematical comment:
$\sigma \circ \sigma = \mathsf{id}$
and then some code:
\begin{code}
sigma = negate
\end{code}
```

When typeset, this results in:

2.3.1 $U{\cdot}(TP)^2$ Source Example

```
module Example where
import Utilities
```

Intro

We can have a suitably mathematical comment: $\sigma \circ \sigma = \mathsf{id}$ and then some code:

```
sigma = negate
```

2.4 $U{\cdot}(TP)^2$ Distribution Structure

At present Unix and Mac OS X users have to build from source, and at present we do not have proper makefiles or install scripts.

For windows users we package up a binary release.

Below are listing of all the relevant installation text files.

2.4.1 README.txt

```
UTP2 is a Theorem Proving Assistant for Unifying Theories of
Programming

Copyright (C) Andrew Butterfield 2007-2012
School of Computer Science and Statistics,
Trinity College, University of Dublin,
Ireland.

See COPYING.txt for licence and warranty information.
```

2.4.2 COPYING.txt

```
This work is released under GPL version 2, see GPL2.txt for
details of the licence and warranty

It incorporates material from Mark Utting's jaza animator,
licensed under GPL2.

It also uses the Parsec library distributed with Haskell,
see PARSEC-LICENCE.txt for details, and the relevant warranty.

It also uses the wxHaskell library distributed with Haskell,
see WXWINDOWS-LICENCE.txt for details, and the relevant warranty.

The software includes sounds from freesound.org, all
distributed under the Creative Commons Sampling Plus 1.0
license, viewable at
 http://creativecommons.org/licenses/sampling+/1.0/

Saoithin-note.wav: derived from Chip116.wav by HardPCM
Saoithin-alert.wav: derived from Chip073 by HardPCM
Saoithin-cheer.wav: derived from dramatic_drum_roll.wav by ingsey101
Saoithin-scream.wav: derived from crash.wav by sageturtle
```

2.4.3 INSTALL.txt

```
INSTALLATION INSTRUCTIONS:

Binary install for Windows

1. Unpack ZIP archive to where you want application to live.

2. Run executable - it should set-up required directories and
   files.

LaTeX packages required to render code + documentation:

saoithin (included)

amssymb,amsmath,verbatim,tikz,graphicx
(not included, but should be in any standard distribution)

mathpartir
(not included, available from http://pauillac.inria.fr/~remy/latex/ )
```

2.4.4 MANIFEST.txt

```
Windows Manifest

UTP2.exe
saoithin.sty
wxc-msw2.8.10-0.11.1.2.dll
README.txt
INSTALL.txt
MANIFEST.txt
COPYING.txt
WXWINDOWS-LICENCE.txt
GPL2.txt
PARSEC-LICENCE.txt
jaza-COPYING.txt
Saoithin-alert.wav
Saoithin-note.wav
Saoithin-cheer.wav
Saoithin-scream.wav
UTP2-Reference-DRAFT.pdf
UTP2-User-Manual-DRAFT.pdf
```

Chapter 3

Odds and Ends

3.1 Parser Implementation

3.1.1 Type/Expression/Predicate Parser Grammar

LXCHARACTERS

$$
\begin{aligned}
\textit{visible} &= \text{all visible (ASCII) characters} \\
\textit{white} &= \text{all invisible (ASCII) characters} \\
\textit{alpha} &= \{\text{`}\mathtt{a}\text{'} \ldots \text{`}\mathtt{z}\text{'}, \text{`}\mathtt{A}\text{'} \ldots \text{`}\mathtt{Z}\text{'}\} \\
\textit{digit} &= \{\text{`}\mathtt{0}\text{'} \ldots \text{`}\mathtt{9}\text{'}\} \\
\textit{sym} &= \textit{visible} \setminus (\textit{alpha} \cup \textit{digit})
\end{aligned}
$$

LXTOKENS

$$
\begin{aligned}
\textit{Tok} &= \textit{Name} \mid \textit{Ident} \mid \textit{Num} \mid \textit{Symbol} \\
\textit{Name} &= \textit{alpha}\ \textit{AlfDig}^* \\
\textit{AlfDig} &= \textit{alpha} \mid \textit{digit} \\
\textit{Ident} &= \textit{Name}\ \textit{IdPost} \mid (\text{`}\mathtt{O}\text{'} \mid \text{`}\mathtt{M}\text{'} \mid \text{`}\mathtt{S}\text{'})[\textit{LstSuffix}] \\
\textit{IdPost} &= \textit{Decor} \mid \text{`}\mathtt{\$}\text{'}[\textit{LstSuffix}] \\
\textit{Decor} &= \epsilon \mid \text{`}\,\mathtt{'}\,\text{'} \mid \text{`}\_\text{'}\textit{AlfDig}^* \mid \text{`}\mathtt{?}\text{'} \\
\textit{LstSuffix} &= \textit{Decor}[\text{`}\backslash\text{'}\textit{Roots}] \\
\textit{Roots} &= (\textit{Name}[\text{`}\mathtt{\$}\text{'}], \text{`}\mathtt{:}\text{'})^+ \\
\textit{Num} &= [\text{`}\mathtt{-}\text{'}]\textit{digit}^+[\text{`}\mathtt{.}\text{'}\textit{digit}^+] \\
\textit{Symbol} &= \text{`}\,\mathtt{`}\,\text{'} \mid (\textit{sym} \setminus \text{`}\,\mathtt{`}\,\text{'})^+ \\
\textit{White} &= \text{`}\,\mathtt{`}\,\text{'} \mid (\textit{sym} \setminus \text{`}\,\mathtt{`}\,\text{'})^+
\end{aligned}
$$

SNWORDS

$$\begin{array}{rcl}
\oplus \in \mathit{Binop} & \supset & \{\text{`\texttt{/\textbackslash}'}, \text{`\texttt{+}'}, \text{`\texttt{<=}'}, \ldots\} \\
n \in \mathit{Name} & \supset & \{\text{`\texttt{a}'}, \text{`\texttt{ab}'}, \text{`\texttt{a1}'}, \ldots\} \\
v \in \mathit{Variable} & \supset & \{\text{`\texttt{y'}'}, \text{`\texttt{z\$}'}, \text{`\texttt{0\$\textbackslash x:y}'}, \ldots\} \\
c \in \mathit{Constant} & \supset & \{\text{`\texttt{true}'}, \text{`\texttt{false}'}, \text{`}\sim\text{'}, \text{`\texttt{0}'}, \text{`\texttt{1}'}, \text{`\texttt{2}'}, \ldots\} \\
Q \in \mathit{Binder} & ::= & \text{`\texttt{\textbackslash}'} \mid \text{`\texttt{the}'} \mid \text{`\texttt{forall}'} \mid \text{`\texttt{exists}'} \mid \text{`\texttt{exists1}'} \\
 & & \mid \text{`\texttt{Forall}'} \mid \text{`\texttt{Exists}'} \mid \text{`\texttt{\textbackslash!}'} \mid \text{`\texttt{!!}'}
\end{array}$$

TEPSYNTAX

$$\begin{array}{rcl}
pe \in \mathit{PredExpr} & ::= & tm\,[\text{`\texttt{<|}'}\,pe\,\text{`\texttt{|>}'}\,tm] \\
tm \in \mathit{Term} & ::= & f\,[\oplus\,tm]\ \text{(with precedences)} \\
f \in \mathit{Factor} & ::= & b^{+} \mid Q\ qs\,[\text{`\texttt{|}'}\,pe]\text{`\texttt{@}'}\,pe \\
b \in \mathit{Base} & ::= & c \mid n \mid v \mid se \mid le \mid de \mid he \mid \ell \mid \text{`\texttt{|:}'}\,T\,\text{`\texttt{:|}'} \\
se \in \mathit{SExpr} & ::= & \text{`}\{\text{'}(pe^{*}_{,}\text{`}\}\text{'} \mid \text{`}\{\text{'}qs\,[\text{`\texttt{|}'}\,pe]\,[\text{`\texttt{@}'}\,pe\,]\,\text{`}\}\text{'} \mid \text{`}\{\text{'}me^{*}_{,}\text{`}\}\text{'} \\
he \in \mathit{HExpr} & ::= & \text{`}\{\{\text{'}pe^{*}_{,}\text{`}\}\}\text{'} \mid \text{`}\{\{\text{'}v^{*}\,[\text{`\texttt{|}'}\,pe]\,[\text{`\texttt{@}'}\,pe\,]\,\text{`}\}\}\text{'} \\
le \in \mathit{LExpr} & ::= & \text{`\texttt{[}'}pe^{*}_{,}\text{`\texttt{]}'} \mid \text{`\texttt{[}'}sb\text{`\texttt{]}'} \\
de \in \mathit{DExpr} & ::= & \text{`\texttt{[[}'}pe\text{`\texttt{]]}'} \mid \text{`\texttt{(}'}pe^{+}_{,}\text{`\texttt{)}'} \\
\ell \in \mathit{Lang} & ::= & \text{`\,`'user-specified`\,`'} \\
T \in \mathit{Type} & ::= & \text{see elsewhere} \\
qs \in \mathit{QVars} & ::= & (n \mid v)^{*}_{,} \\
me \in \mathit{MElem} & ::= & pe\text{`\texttt{|->}'}pe \\
sb \in \mathit{Subs} & ::= & e^{*}_{,}(\text{`\texttt{//}'} \mid \text{`\texttt{///}'})\ qs, \quad n \geqslant 1
\end{array}$$

TEPLSYNTAX

$$
\begin{array}{rcl}
pe \in \mathit{PredExpr} & ::= & tm\,[\text{`}\texttt{<|}\text{'}\,pe\,\text{`}\texttt{|>}\text{'}\,tm] \\
tm \in \mathit{Term} & ::= & f\,[\oplus\,tm\,]\ (\text{ with precedences}) \\
f \in \mathit{Factor} & ::= & b\,[\,f\,] \\
b \in \mathit{Base} & ::= & c \mid n \mid v \mid \text{`}\texttt{[[}\text{'}\,pe\,\text{`}\texttt{]]}\text{'} \mid \text{`}\texttt{(}\text{'}\,pe^{+}_{,}\,\text{`}\texttt{)}\text{'} \\
 & & \mid \text{`}\{\text{'}\,se \mid \text{`}\texttt{[}\text{'}\,le \mid \text{`}\{\{\text{'}\,he \\
 & & \mid Q\ qs\,[\text{`}\texttt{|}\text{'}\,pe]\,\text{`}\texttt{@}\text{'}\,pe \\
 & & \mid \text{`}\texttt{|:}\text{'}\,te\,\text{`}\texttt{:|}\text{'} \\
 & & \mid \ell \\
se \in \mathit{SExpr} & ::= & \text{`}\}\text{'} \mid pe\ sec \mid q\,[\text{`}\texttt{|}\text{'}\,pe]\,[\text{`}\texttt{@}\text{'}\,pe\,]\,\text{`}\}\text{'} \\
sec \in \mathit{SExprCont} & ::= & \text{`}\}\text{'} \mid \text{`}\texttt{,}\text{'}\,pe^{+}_{,}\,\text{`}\}\text{'} \mid \text{`}\texttt{|->}\text{'}\,pe\ mec \\
mec \in \mathit{MExprCont} & ::= & \text{`}\}\text{'} \mid \text{`}\texttt{,}\text{'}\,pe\,\text{`}\texttt{|->}\text{'}\,pe\ mec \\
le \in \mathit{LExpr} & ::= & pe^{*}_{,}\ lec \\
lec \in \mathit{LExprCont} & ::= & \text{`}\texttt{]}\text{'} \mid \text{`}\texttt{//}\text{'}\,qs\,\text{`}\texttt{]}\text{'} \mid \text{`}\texttt{///}\text{'}\,qs\,\text{`}\texttt{]}\text{'} \\
qs \in \mathit{QVars} & ::= & (n \mid v)^{*}_{,} \\
he \in \mathit{HExpr} & ::= & \text{`}\}\}\text{'} \mid pe\ hec \mid v^{*}\,[\text{`}\texttt{|}\text{'}\,pe]\,[\text{`}\texttt{@}\text{'}\,pe\,]\,\text{`}\}\}\text{'} \\
hec \in \mathit{HExprCont} & ::= & \text{`}\}\}\text{'} \mid \text{`}\texttt{,}\text{'}\,pe^{+}_{,}\,\text{`}\}\}\text{'} \\
\ell \in \mathit{Lang} & ::= & \text{`‘’user-specified‘‘’}
\end{array}
$$

3.2 Mac OS X ScreenShot Renaming

This is standalone code, intended for use on Mac OS X to rename screenshots from that platform.

```
module OSXRename where
import Data.Char
import System.IO
import System.Directory
import System.FilePath.Posix

what = putStrLn "doit :: IO ()"
doit = ssRenameFiles isOSXNewScreenShot

isOSXNewScreenShot path
               = takeExtension path == ".png" && take 11 path == "Screen Shot"

ssRenameFiles :: (FilePath -> Bool) -> IO ()
ssRenameFiles newShot
 = do paths <- getDirectoryContents "."
      putStrLn "\nBEFORE:"
      putStrLn $ unlines paths
      putStr "Starting Number ? (negative to abort) : "
      txt <- getLine
      let firstNo = (read txt) :: Int
      if firstNo < 0
       then putStrLn "No files renamed"
       else do putStr "Screenshot Series Title (filename characters only) : "
               title <- getLine
               createDirectoryIfMissing True title
               mapM_ (doRename title) $ zip [firstNo..] $ filter newShot paths
               putStrLn ("Files renamed")
               paths' <- getDirectoryContents ("./"++title)
               putStrLn "\nAFTER:"
               putStrLn $ unlines paths'

doRename :: String -> (Int, FilePath) -> IO ()
doRename title (n,oldpath)  =  renameFile oldpath (title++"/"++title++display 4 n++".png")

display w n =  replicate (w - length s) '0' ++ s where s = show n
```

Bibliography

[HH98] C. A. R. Hoare and Jifeng He. *Unifying Theories of Programming.* Prentice-Hall, 1998.