Concepts tool - Manual

Jonathan Beaumont
j.r.beaumont@ncl.ac.uk
School of Electrical and Electronic Engineering, Newcastle University,UK

Last updated September 4, 2016

Contents

1 Introduction

Concepts are a formal method of specifying asynchronous circuits. With concepts, one can describe behaviours of a circuit and its environment at various levels, with the aim of defining the behaviours in terms of gates, protocols or simple signal interactions. Using this method, one can split a design into scenarios, based on operational modes, individual functions, or any other method a user decides. These can be specified separately, using concepts, and then combined to provide a full system specification. Through this, reuse is promoted, allowing concepts to be reused between scenarios and even entire designs. More information on the theory of concepts can be found in [1], the latest version of which can be found here.

In this document, we will discuss the associated tool for concepts. This open source tool is written in *Haskell*, and contains both the library for concepts, including abstract concepts and circuit concepts built upon the abstract, and a tool for translating concepts into *Signal Transition Graphs* (STGs) [2] [3]. These are commonly used for the specification, verification and synthesis of asynchronous control circuits in the academic community, and they are supported by multiple EDA tools, such as Petrify [4],Mpsat [5], Versify [6],Workcraft [7][8], and others. The aim of this tool is to design and debug concepts, and then translate them to STGs, where they can be used with these tools.

The latest version of this tool, and this manual, can be found in the GitHub repo. This tool is also distributed as a back-end tool for Workcraft. This version of the concepts tool will be the latest version that works correctly with Workcraft. The latest version of Workcraft, which also features tools such as Petrify and Mpsat, can be downloaded from `http://workcraft.org/`.

2 Installation

The concepts tool, as well as Workcraft are available for *Windows*, *Linux* and *Mac OS X*. The installation instructions for all of these operating systems are the same. We will be referring to directories using the forward-slash character ('/') as a separator, however for *Windows*, replace this with a back-slash character ('\').

If choosing to use the concepts tool on its own, this must be downloaded from the GitHub repo. If you choose to use this tool as part of Workcraft, download this from the Workcraft website. Once downloaded, extract the contents of the folder, and move them to a directory you wish to run them from.

2.1 Concepts tool requirements

The concepts tool is written in *Haskell*, and as such, the *Glasgow Haskell Compiler* (GHC) is required for installation as well as the *Cabal* library builder. Cabal builds and installs packages for Haskell, and is necessary for the concepts tool. GHC and Cabal can be downloaded from `https://haskell.org/downloads`. We recommend the "Minimal installers", as this contains GHC and Cabal.

If needed, download the installers for GHC and Cabal, and follow the instructions to install these.

2.2 Workcraft requirements

Workcraft is written in Java, and the latest version of the *Java Runtime Environment* (JRE) needs to be installed to run. This can be downloaded from `http://java.com/en/download/`.

If needed, download the JRE installer, and follow the instructions to install this.

While the concepts tool is distributed with Workcraft, it still needs to be built by Cabal in order for it to be run. See Section 2.1 for requirements for the concepts tool.

2.3 Installing the tool

Once either the concepts tool or WORKCRAFT has been extracted and moved to the desired directory, using command line, navigate to the concepts tool directory, or if using WORKCRAFT, navigate to this directory, and then navigate to the concepts tool directory, found in `tools/concepts` (for *OS X*, it is located within the `Workcraft.app` contents folder `Contents/Resources/tools/concepts`.

Now, the process of installing the tool is the same, regardless of how you aim to use the concepts tool. First of all, let's update the Cabal package list. To do this, run:

```
$ cabal update
```

When this has finished, the package list will be updated, and all dependencies of the concepts tool will be able to be automatically downloaded, built and installed during the process of installing the concepts tool.

Now, to build and install the concepts tool, simply run:

```
$ cabal install
```

If this process completes successfully, the tool will now be installed and ready to be used.

3 Using the tool from command line

With the tool installed, we can now start to use it to translate concepts to STGs. We will begin by discussing how it is used from command line. Usage as a back-end tool in WORKCRAFT is discussed in Section 4.

The standard command for the tool is as follows:

```
$ runghc <path-to-translate> <path-to-concepts-file>
```

The three parts of this are as follows:

- `runghc` - This runs the translate file
- `<path-to-translate>` - This is file path pointing to the translate code file, which performs the necessary operations to translate concepts to STGs.
- `<path-to-concepts-file>` - This is the path pointing to the file containing the concepts to be translated.

When running the concepts tool from command line, it does not necessarily matter in which directory you currently are, as long as the paths to the translate code, and the concepts file are correct.

Now, let's use one of the examples included with the concepts tool to show the usage. These can be found in the `examples` directory. For this section, we will assume that we are currently in the concepts tool directory. The example we will use is the file titled "`Celement_with_env_1.hs`". This has the ".hs" file extension, as it is in fact a file using Haskell code, and all files containing concepts should feature this file extension. To translate this concepts file to an STG, the following command must be run:

```
$ runghc translate/Main.hs examples/Celement_with_env_1.hs
```

When the translation is complete, the tool will output the following:

```
.model out
.inputs A B
.outputs C
.internals
.graph
A0 A+
A+ A1
A1 A-
A- A0
B0 B+
B+ B1
B1 B-
```

```
B- B0
C0 C+
C+ C1
C1 C-
C- C0
A1 C+
C+ A1
B1 C+
C+ B1
C1 A-
A- C1
C1 B-
B- C1
A0 C-
C- A0
B0 C-
C- B0
C0 A+
A+ C0
C0 B+
B+ C0
.marking {A0 B0 C0}
.end
```

This output is the STG representation in *.g* format. *.g* files are a standard type used as input to tools, such as PETRIFY, MPSAT, and WORKCRAFT. Therefore, this output can by copy-and-pasted into a file, and saved with the file etension *.g*, and then used as input to these tools.

The concepts tool can be used in a similar way, ensuring that the file paths to the translate code file, and the concepts input file, are correct. Section 6 contains information on how to layout a concepts file, to avoid as many errors as possible.

Any errors that occur during the translation process will produce errors referring to the problematic areas, usually lines, of the concepts that are problematic.

4 Using the tool from Workcraft

This section will discuss how to use the concepts tool from within WORKCRAFT. There are many other features of WORKCRAFT, both as part of the STG plug in, some of which I will discuss in the context of concepts here, and as part of other modelling formalisms. More information on these can be found at `http://workcraft.org/`.

4.1 Translating and authoring concepts

First of all, WORKCRAFT must be started. This can be done by running the start up script, located in the WORKCRAFT directory in *Windows* and *Linux*. In *Windows*, this script is named "`workcraft.bat`". In *Linux*, it is simply "`workcraft`". In *OS X*, WORKCRAFT can be started instead by double clicking the WORKCRAFT icon, which is the app container for the necessary files.

When workcraft starts, you will be greeted by blank screen, as seen in Figure 1.

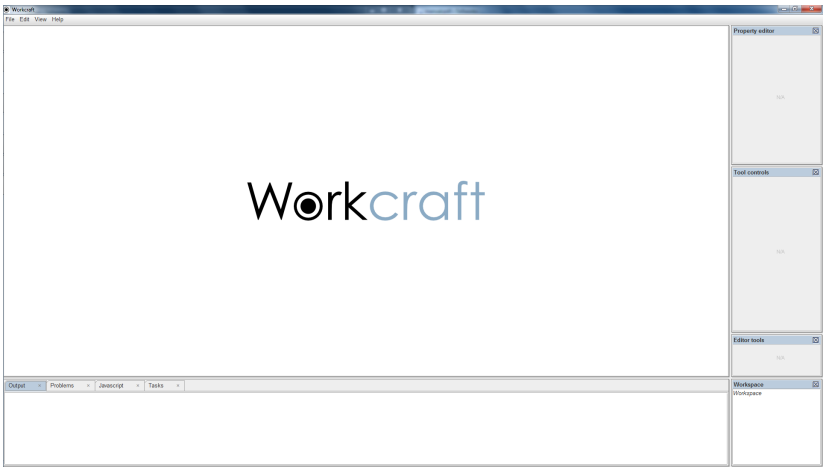

Figure 1: Workcraft immediately after starting.

Now, we need to open a new work, specifically a new STG work. Open the "New work" dialog using the menu bar, `File -> Create work...`, or by pressing `Ctrl-N` (`CMD-N` on *OS X*). This will bring up a menu as seen in Figure 2.

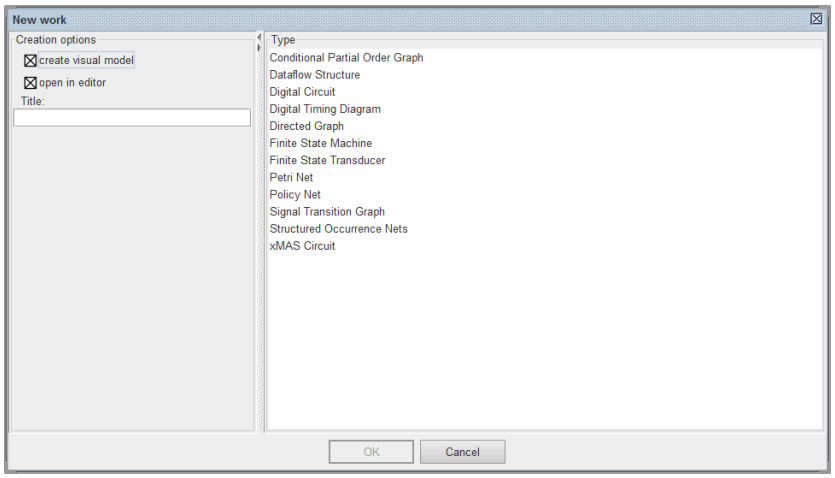

Figure 2: The create work window.

In this window, select "*Signal Transition Graph*' abd cick the "OK" button at the bottom of the window. This will the open a blank workspace in which we can create an STG, which will look similar to Figure 3.

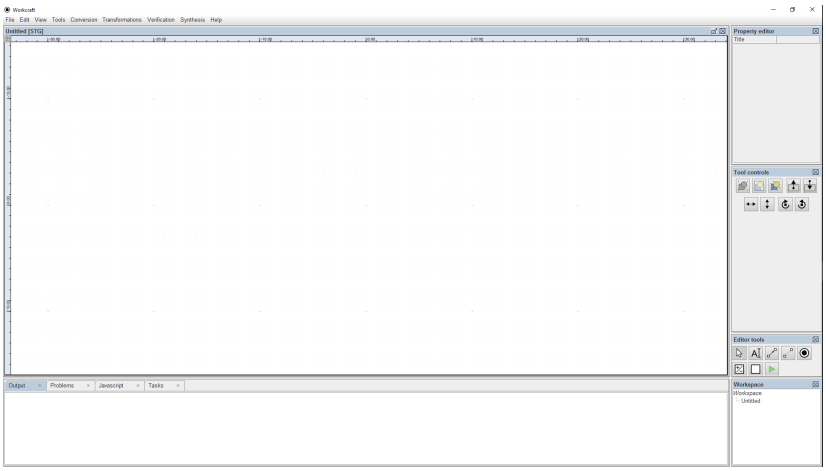

Figure 3: A new STG workspace

Now, we can start translating concepts. To do this, first we need to open the concepts dialog. This is done from the menu bar, by selecting the "*Conversion*" menu, and then the "*Translate concepts...*" option. The concepts dialog will look as shown in Figure 4.

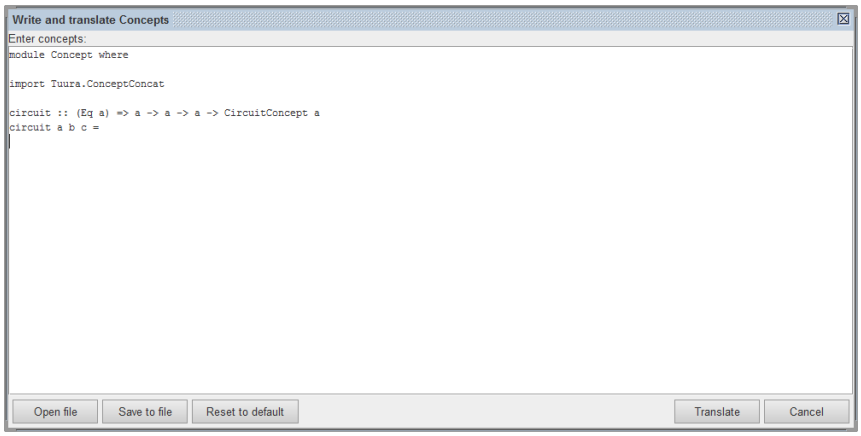

Figure 4: The concepts dialog.

From within this dialog, one can write their own concepts, from the default template as shown in Figure 4, or open an existing concepts file, with the *.hs* extension. When satisfied with the concepts written, a user can choose to save the file, if not already saved, and then translate these concepts.

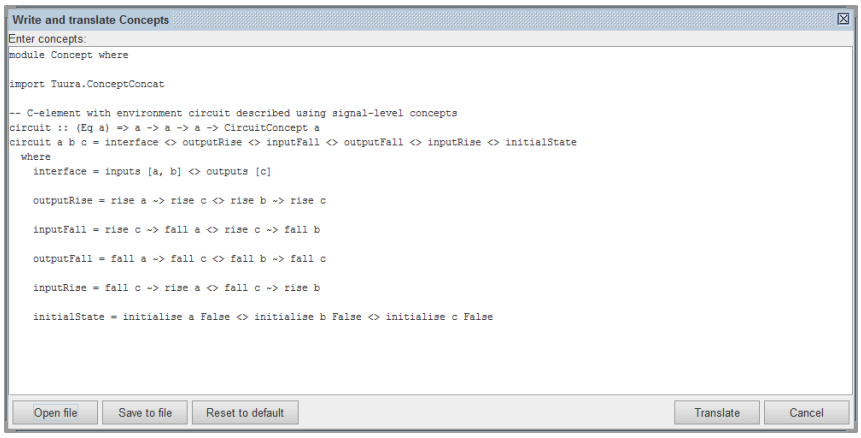

Figure 5: The concepts dialog with a concept file opened.

Figure 5 is the concepts dialog after we have opened the Celement with environment example, named "*Celement_with_env1.hs*", from the concepts tool examples directory. Clicking translate at this point will produce an STG representation of these concepts in the workspace.

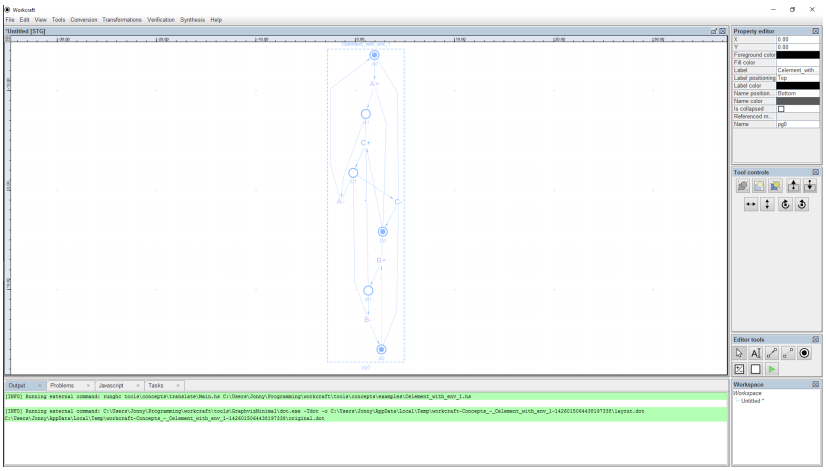

Figure 6: The STG produced from translating the concepts.

The translated concepts will look similar to Figure 6

Now, a user can choose to insert more concepts, make changes to this STG, and once they are satisfied with it, can then perform various functions on this STG. One can perform transformations, verifications, simulations and synthesis on this STG using the menus within this workspace now. Any further changes to this STG, based on the results of these operations can be made to this STG or to the concepts file.

5 Importing concepts directly

It is also possible to import concepts directly from a file, without having to view the concepts first. This can be done from the "*File*" menu, by selecting the "*Import...*" option.

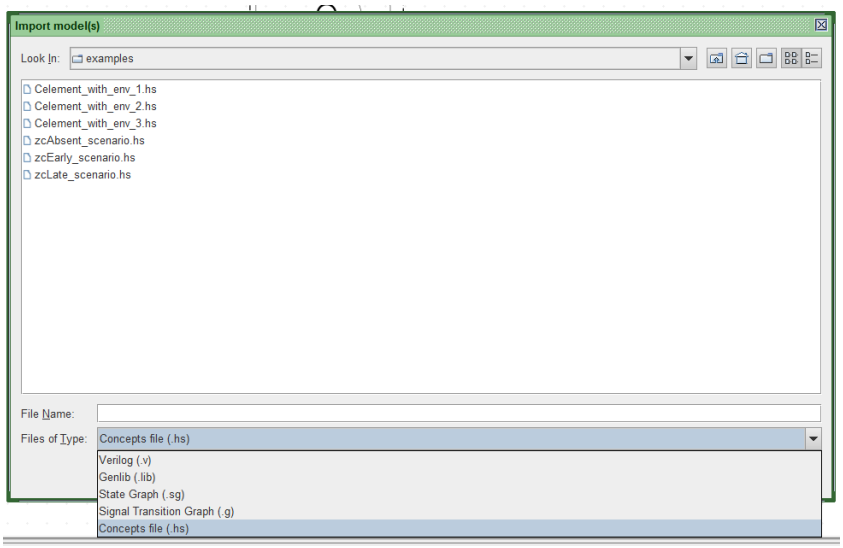

Figure 7: The STG produced from translating the concepts.

When importing concepts using this menu, ensure to set the "*Files of Type*" option to "*Concepts file (.hs)*", as shown in Figure 7

5.1 Errors

If any errors are encountered during the translation process, WORKCRAFT will produce a helpful error message. This usually can tell you with more detail what the issue that is causing the error is, but will ask you to refer to WORKCRAFT's console window for specific line numbers. These errors will include whether a signal has not been declared as an input or output, a signal has not had it's initial state given, or even that the concepts tool has not been installed correctly.

6 Concepts file layout

References

[1] J Beaumont A Mokhov D Sokolov A Yakovlev. High-level asynchronous concepts at the interface between analogue and digital worlds. 2016.

[2] T.-A. Chu. *Synthesis of self-timed VLSI circuits from graph-theoretic specifications*. PhD thesis, Massachusetts Institute of Technology, 1987.

[3] A. Yakovlev L. Rosenblum. Signal graphs: from self-timed to timed ones. *International Workshop on Timed Petri Nets*, pages 199–206.

[4] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev. *Logic Synthesis for Asynchronous Controllers and Interfaces*. Springer, 2002.

[5] Victor Khomenko, Maciej Koutny, and Alex Yakovlev. Detecting state encoding conflicts in stg unfoldings using sat. *Fundamenta Informaticae*, 62(2):221–241, 2004.

[6] Oriol Roig i Mansill. *Formal Verification and Testing of Asynchronous Circuits*. PhD thesis, Citeseer, 1997.

[7] I. Poliakov, D. Sokolov, and A. Mokhov. Workcraft: a static data flow structure editing, visualisation and analysis tool. In *Petri Nets and Other Models of Concurrency*, pages 505–514. 2007.

[8] Workcraft. www.workcraft.org.