Andrew J. Cook
May 14, 2004

# Using Upwards

`upwards` is a program for coverting rules used in the SPARK proof checker into a form that is usable by CLAM. This document is intended to provide instuctions on how to use the collection of `upwards` scripts. For more information on the workings of `upwards` see [?].

## 1 `upwards`

`upwards` consists of several scripts which use the prolog program contained in the `upwards` directory. These wrapper scripts calculate the absolute paths the program needs to be able to perform the transformation.

### 1.1 `uplift`

The `uplift` script performs the transformation of all the rules that may be relevant to a particular sub-program. As well as transforming the sub-program specific rules generated by spark it will also convert all the general user defined rules and the builtin rules.

### 1.2 `genuplift`

The `genuplift` script will only convert the rules in the general rules directory, though because of the way the uplift program works it need to be given the path of a project directory.

The general rules directory `gen_rules` should be at the same level as directories containing the programs or up a level from the path given for the project directory. So if given `examples/sqrt` as the project directory it will expect to find the general rule files in `examples/gen_rules`.

### 1.3 `spcuplift`

The `genuplift` script only converts the sub-program specific generated rules. So it requires the relevant project directory as an argument.

### 1.4 `uplifter`

Due to changes to the `uplift` script the `uplifter` scripts are now obsolete.

## 2 Using Uplift

The files RM(gen|spc)uplift contain instructions for using the uplift scripts.

Each of the scripts (`uplift`, `genuplift`, and `spcuplift`) take two arguments, the path of a project directory and the path of the spade rules directory in the clam library.

`uplift` and `spcuplift` will convert the sub-program specific rules from this directory and insert them into the clam library. Even though `genuplift` doesn't convert the sub-program specific rules it still requires a project directory(any project directory) to enable it to find the `gen_rules` directory.

`uplift` and `genuplift` will convert the general rules from the standard `gen_rules` directory found by using the input project directory. The `upwards` program expects the directory to be at the same level as the project directories.

### 2.1 A use of `uplift`

A use of the `uplift` script might look like this:

```
nuspade/spade2clam/uplift nuspade/examples/filters nuspade/lib/srule
```

*Notes in this series are for $\epsilon$-baked ideas, for $1 \geq \epsilon \geq 0$. Only exceptionally should they be cited or distributed outwith the NuSPADE project (EPSRC GR/R24081).

The script (in `nuspade/spade2clam/`) will take the generated rules from the subdirectory `nuspade/examples/filters/filter_sum` and convert them(it should find the subdirectory `filter_sum` automaticly). It will also converts any general rules it finds in `nuspade/examples/gen_rules`. The built in rules are also converted.

The resulting translated rules are output into the directory `nuspade/lib/srule/`.

The `genuplift` and `spcuplift` scripts take the same arguments as `uplift` but will only perform the general rule and subprogram specific rule transformations respectively.

# 3 vcs-to-clam

The `vc2clam` script uses the program in the vcs-to-clam directory to transform generated vcs in the sub-program directory to clam planning nodes.

## 3.1 vc2clam

`vc2clam` takes three arguments, the path of the sub-program directory, the sub-program base name and the clam library directory. The sub-program directory is the directory generated by SPARK within the project directory for the specific sub-program. The base name is the name, without any extension, given to the files in the sub-program directory. The converted vcs will be stored in the subdirectory vcs within the clam library directory.

## 3.2 A use of vc2clam

A use of the `uplift` script might look like this:

```
nuspade/spade2clam/vc2clam nuspade/examples/filters/filter_sum/ filter_s nuspade/lib/
```

The script (`nuspade/spade2clam/vc2clam`) will take the generated vcs found in `nuspade/examples/filters/filter_sum/filter_s.vcg` (*sub-program directory*/*base name*`.vcg`) and output them into the `nuspade/lib/vcs/` directory.