Pre Processing VDM++ models for the Automatic Proof Support

Miguel Ferreira
miguel@di.uminho.pt
Samuel Silva
samuel.silva@gmail.com

April 10, 2008

Abstract

The Automatic Proof Support (APS) main goal is to discharge the proof of VDM++ integrity properties, known as proof obligations, in the theorem prover HOL4. As automatic the proof can be, there is many manual work to be done before a VDM++ model can be translated to HOL4 syntax. This contribution is a preliminary step of the translation, automating the processes that prepare a VDM++ model to be fed to the VdmHolTranslator.

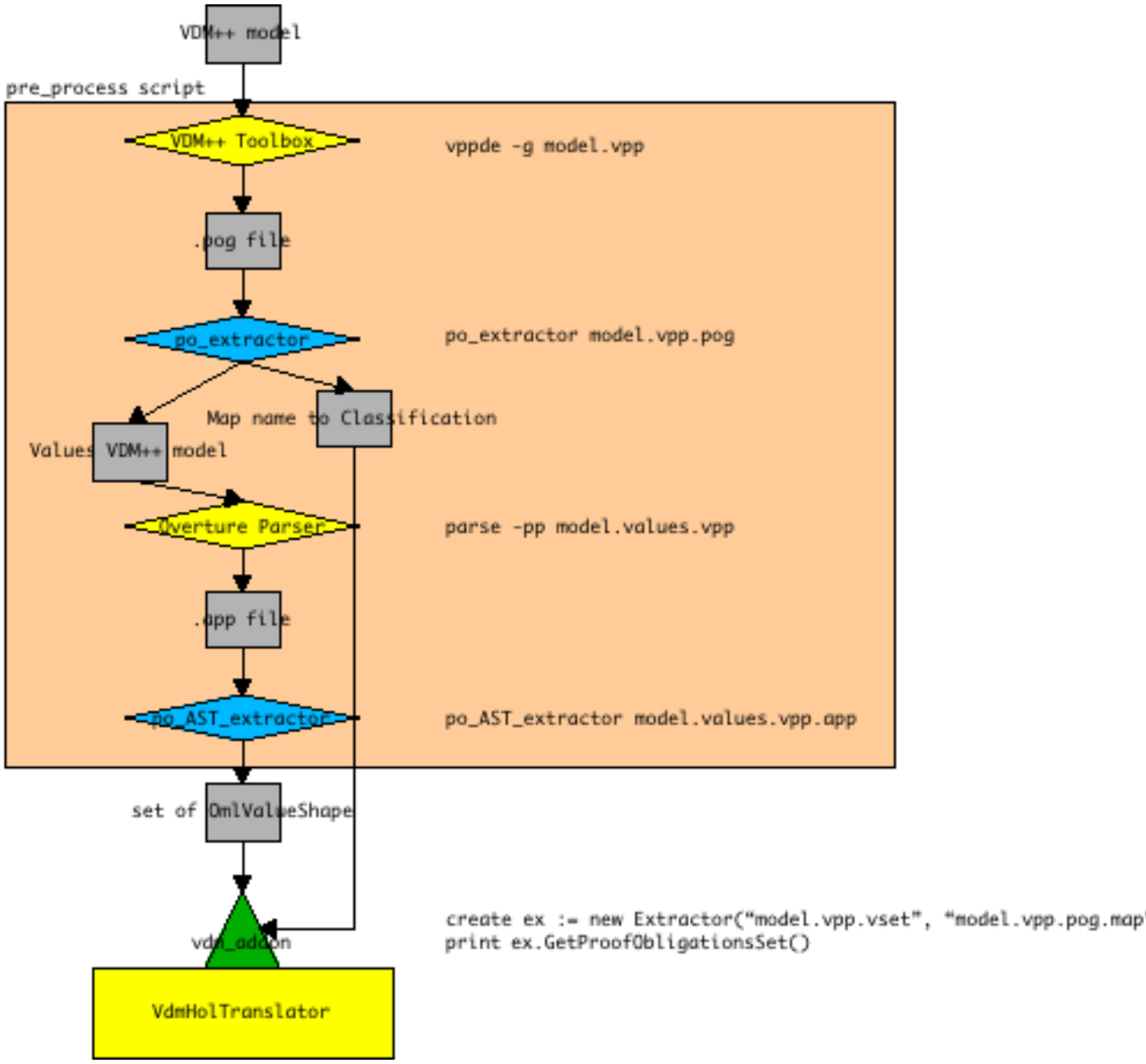
1 Introduction

It is assumed that the reader has used the APS, and all tools involved in it: The VDM++ Toolbox, Overture Parser, VdmHolTranslator, HOl.

...

A VDM++ model can be type checked and tested in The VDM++ Toolbox, but, in matters of correctness, the toolbox can only generate the proof obligations. The Integrity Checker is responsible for checking the model for invariants preservation, function application, mapping application, termination, and other properties that assure the model correctness and integrity. These properties, or proof obligations, must then be proved to be true by the user.

The APS uses the VdmHolTranslator and HOL to achieve such proofs.



2 Preparing a VDM++ model for the VdmHolTranslation

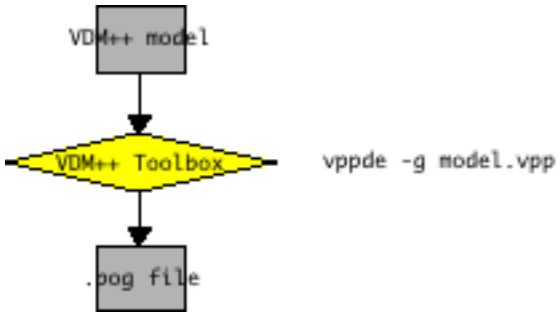
Five different and sequential steps must be performed to prepare a model and it's proof obligations to be discharged in HOL.

2.1 Step 1: Type check and proof obligation generation

Type checking a VDM++ model can be done with The VDM++ Toolbox or the Overture Parser. As for the proof obligation generation it can only be achieved in the toolbox.

In the command line passing the `-g` option to `vppde`, it generates a file with the .pog extension, containing all the proof obligations. The file also contains additional

information, from which the proof obligations classification are needed in later steps.



2.2 Step 2: Process the proof obligations for the next step

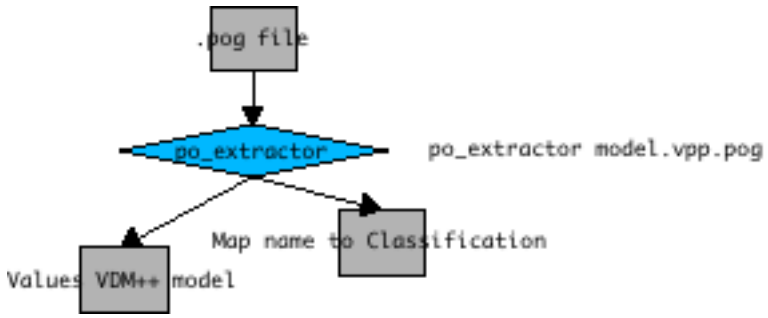
The goal of the next step is to generate an Oml Abstract Syntax Tree (AST) for each proof obligation. This step processes the .pog file to generate a new VDM++ class that contains the proof obligation's expressions assigned to `bool` values.

The last step will need the proof obligation's classification, that is also stated in the .pog file, and must be extracted.

One of the automations presented here is the processing of the .pog file, that is done based on ad hoc observation of it's syntax. The parser, `po_extractor`, is generated by the open-source tool JFLEX. Its grammar is very simple and unsophisticated, making the parser consume lines that are analyzed in a Java implemented object.

The output of the parser is a VDM++ class, which only has declared values, and a VDM++ mapping of proof obligation names to `Classification` (type defined in the VdmHolTranslator) in a file with the extension .map.

A better way of processing the .pog file would be to specify its syntax grammar, and write a "real" parser.

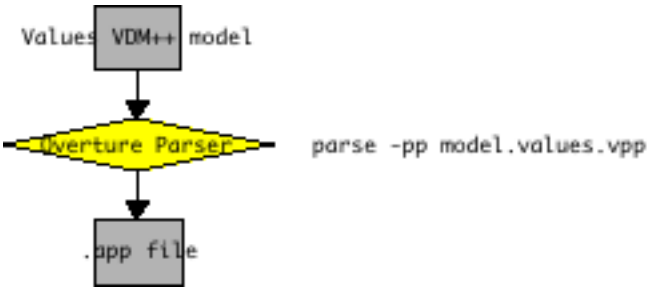


2.3 Step 3: Generate the Oml AST of each proof obligation

Passing the class obtained in the last step to the Overture Parser, is all that takes to generate the ASTs.

The Overture Parser has an option `-pp` that produces as output the AST, and some other output lines that must be filtered before dumping the AST to a .app file. Filtering the output is, for the time being, done in a bash script that glues all the steps together.

A Perl script would be a better option for portability reasons.



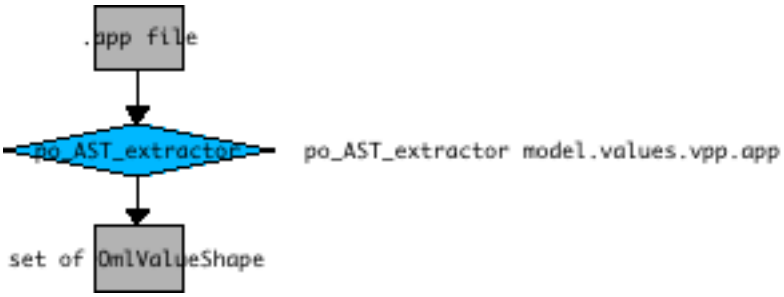
2.4 Step 4: Processing the Oml AST

Having the .app file with the Oml AST of the class that contains the proof obligation expressions, it is necessary to extract the portions os the AST that actually represent the expressions.

Another parser was developed, `po_AST_extractor`, that automates this step. This one is, also very simple, and reads line by line looking for specific Oml AST class constructors.

The output is a VDM++ set of `OmlValueShape` (class defined in the Oml AST specification).

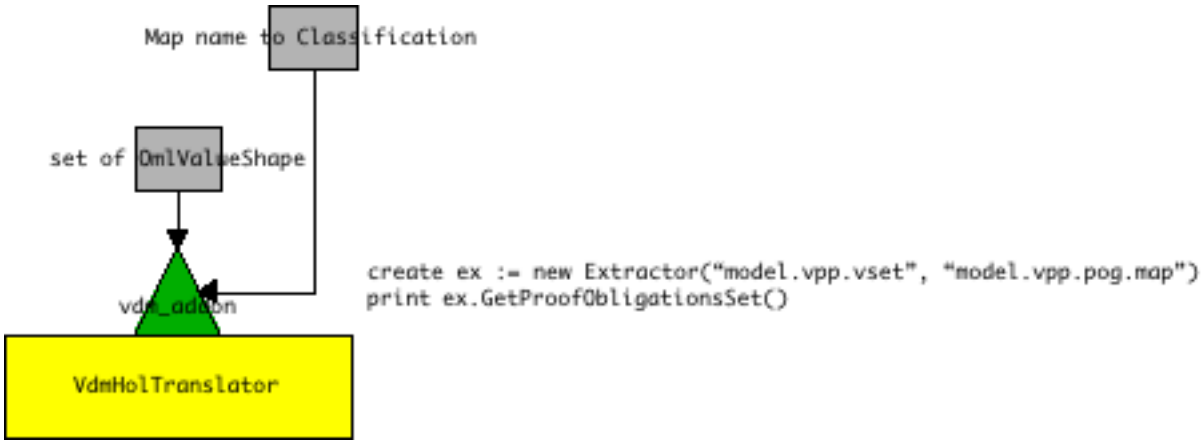
As in step 2, this parser should also be based on the grammar of the input files syntax.



2.5 Step 5: Generate a set of ProofObligation objects

The VdmHolTranslator needs the Oml AST of the model and a set of `ProofObligation`. The `ProofObligation` objects can be constructed from the Oml AST portion that represents the proof obligation expression, and a `Classification` type inhabitant.

The goal of this step it to generate the mentioned set. This can be done using the VDM++ class `Extractor`, developed for this purpose. Its constructor takes a OmlValueShape set (generated in step 4), and a mapping of proof obligation value name to Classification (generated in step 2), which are consumed by the operation `GetProofObligations`.



3 Press the button

Performing the five steps presented in the last section on a VDM++ model, prepares it and its proof obligations to be translated to HOL.

The automation of the first four steps is done in a bash script, `pre_process`, to which a model is passed as input. The script calls all the steps sequentially, in the presented order, collecting intermediate outputs for inputing to subsequent steps.

Step 5, as it is done inside The VDM++ Toolbox with the VdmHolTranslator project loaded, could be included in the translation operations.