ExSpec—Executable Specifications

Stefan Ratschan

June 27, 2023

1 Introduction

This is a program for executing algorithm specifications written in first-order predicate language with variable ranging over the integers and functions from closed integer intervals to integers. The solver searches for a satisfying assignment of a given input formula. Hence the input to an algorithm specification simply has to be part of the given formula. Examples are part of the distribution.

The solver proceeds by enumerating all possibilities, and hence—in theory, for formulas without universal quantifiers—will eventually hit upon a solution. However, in practice, efficiency of the solver depends essential on bounds that the user provides for all variables. Moreover, universal quantifiers can only be proved if their variables range over closed intervals. The smaller these intervals are, the faster the algorithm will terminate.

2 Usage

To run a first example do:

```
./exspec <examples/maximum.spec
```

To pretty print the example do:

```
./pp <examples/maximum.spec
```

Note that the output of pretty printing is valid input for `exspec`, as well.

3 Syntax

$$
\begin{array}{rcl}
\langle\text{spec}\rangle & ::= & \langle\text{declBlock}\rangle.\langle\text{formula}\rangle \\
\langle\text{formula}\rangle & ::= & \langle\text{atomic}\rangle\wedge\langle\text{formula}\rangle \ \mid \\
 & & \langle\text{atomic}\rangle\vee\langle\text{formula}\rangle \ \mid \\
 & & \langle\text{atomic}\rangle\Rightarrow\langle\text{formula}\rangle \ \mid
\end{array}
$$

$$\begin{array}{rcl}
 & & \neg\langle\text{formula}\rangle \mid \\
 & & \forall\ \langle\text{declBlock}\rangle\,\texttt{.}\,\langle\text{formula}\rangle \mid \\
 & & \exists\ \langle\text{declBlock}\rangle\,\texttt{.}\,\langle\text{formula}\rangle \mid \\
 & & \texttt{[}\langle\text{formula}\rangle\texttt{]} \\
\langle\text{atomic}\rangle & ::= & \langle\text{term}\rangle\langle\text{predSym}\rangle\langle\text{term}\rangle \\
\langle\text{predSym}\rangle & ::= & = \mid \neq \mid > \mid \geq \mid < \mid \leq \\
\langle\text{declBlock}\rangle & ::= & \langle\text{decl}\rangle\texttt{,}\langle\text{declBlock}\rangle \mid \langle\text{decl}\rangle \\
\langle\text{decl}\rangle & ::= & \langle\text{variable}\rangle\texttt{:}\langle\text{type}\rangle \\
\langle\text{type}\rangle & ::= & \langle\text{intType}\rangle \mid \langle\text{funcType}\rangle \\
\langle\text{intType}\rangle & ::= & \mathbb{Z} \\
\langle\text{funcType}\rangle & ::= & \mathbb{Z} \to \mathbb{Z}
\end{array}$$

Terms are built in the usual way, allowing addition, subtraction, and unary plus and minus as function symbols.

Alternatives for some terminal symbols are listed in the following table:

$\mathbb{Z}$	`integer`
$\mathbb{Z} \to \mathbb{Z}$	`integer -> integer`
$\Rightarrow$	`==>`
$\wedge$	`/\`
$\vee$	`\/`
$\neg$	`not`
$\forall$	`forall`
$\exists$	`exists`
$\to$	`->`
$\neq$	`<>`
$\geq$	`>=`
$\leq$	`<=`