Sentinel - Proof Checker - documentation

Konrad Werbliński

January 16, 2018

Code is divided into four separate files:

1. `Sentinel.hs`.
2. `Parser.hs`
3. `AST.hs`
4. `ProofChecker.hs`

Compilation: `ghc Sentinel`

1 Sentinel.hs

Handles input and output, launches parsing and proof checking.

2 Parser.hs

Parses input file into abstract syntax tree built using types defined in `AST.hs`. Made using parsec 2 library.

3 AST.hs

Definition of types (and basic operations on that types) used to describe abstract syntax tree. The file containts definitions of following types:

1. `Block` represents a proof or an axiom.
2. `Proof` represents part of a proof which is either a sequence, a frame or a formula.
3. `Formula` represents first order logic formula.

All three types are instances of Show class. Formula is also instance of Eq class. Operations defined in `AST.hs`:

1. `getName` - returns the name of the proof (value of type Block).
2. `getGoal` - retuuns a goal of the proof (value of type Block).
3. `getProofPos` - returns position of the proof element (value of type Proof)
4. `substitute` - substitutes every occurrence of a specified variable in the first formula with a second formula.

4 ProofChecker.hs

File responisble for checking correctness of the proofs. It contains, among others, following functions:

1. `takeAllProofs`, `takeAllAxioms` - takes all proofs / axioms from `Block` list.
2. `checkAllProofs` - checks all the proofs in the list using `checkProof`. Handles errors and produces text output.
3. `checkProof` - checks single proof, using `tryToProve`.
4. `match` - patern matching used in searching for an axiom and in reasoning rules for quantifiers.
5. `tryToProve` - tries to prove single proof part (sequence, frame, formula).
6. `checkIfProved` - checks if formula was proved already.
7. `checkIfAxiom` - tries to find axiom, matching to a currently checked formula.
8. `introduce` - tries to prove formula, using previously proved proof parts and introduction rule.
9. `eliminate___` - set of functions (e.g. `eliminateImp`, `eliminateOr`) which try o prove formula, using previously proved proof parts and elimination rule.