APPENDIX B

Concrete Syntax

Predefined symbols

These symbols have a predefined meaning in Version 2.6. Note that they are not reserved words. For instance, they could also be used in principle as user-defined sort or function symbols in scripts.

`Bool continued-execution error false immediate-exit incomplete logic memout sat success theory true unknown unsupported unsat`

Predefined keywords

These keywords have a predefined meaning in Version 2.6.

`:all-statistics :assertion-stack-levels :authors :category :chainable :definition :diagnostic-output-channel :error-behavior :extensions :funs :funs-description :global-declarations :interactive-mode :language :left-assoc :license :name :named :notes :pattern :print-success :produce-assignments :produce-models :produce-proofs :produce-unsat-assumptions :produce-unsat-cores :random-seed :reason-unknown :regular-output-channel :reproducible-resource-limit :right-assoc :smt-lib-version :sorts :sorts-description :source :status :theories :values :verbosity :version`

Auxiliary Lexical Categories

⟨*white_space_char*⟩	::=	9_{dec} | 10_{dec} | 13_{dec} | 32_{dec}
⟨*printable_char*⟩	::=	32_{dec} | ⋯ | 126_{dec} | 128_{dec} | ⋯ | 255_{dec}
⟨*digit*⟩	::=	`0` | ⋯ | `9`
⟨*letter*⟩	::=	`A` | ⋯ | `Z` | `a` | ⋯ | `z`

Tokens

Reserved Words

General: `!` `_` `as` `BINARY` `DECIMAL` `exists` `HEXADECIMAL` `forall` `let` `match` `NUMERAL` `par` `STRING`

Command names: `assert` `check-sat` `check-sat-assuming` `declare-const` `declare-datatype` `declare-datatypes` `declare-fun` `declare-sort` `define-fun` `define-fun-rec` `define-sort` `echo` `exit` `get-assertions` `get-assignment` `get-info` `get-model` `get-option` `get-proof` `get-unsat-assumptions` `get-unsat-core` `get-value` `pop` `push` `reset` `reset-assertions` `set-info` `set-logic` `set-option`

Other tokens

```
(
)
⟨numeral⟩         ::=  0 | a non-empty sequence of digits not starting with 0
⟨decimal⟩         ::=  ⟨numeral⟩.0*⟨numeral⟩
⟨hexadecimal⟩     ::=  #x followed by a non-empty sequence of digits and letters
                       from A to F , capitalized or not
⟨binary⟩          ::=  #b followed by a non-empty sequence of 0 and 1 characters
⟨string⟩          ::=  sequence of whitespace and printable characters in double quotes
                       with escape sequence ""
⟨simple_symbol⟩   ::=  a non-empty sequence of letters, digits and the characters
                       + - / * = % ? ! . $ _ ~ & ^ < > @ that does not start
                       with a digit
⟨symbol⟩          ::=  ⟨simple_symbol⟩
                    |  a sequence of whitespace and printable characters that
                       starts and ends with | and does not otherwise include | or \
⟨keyword⟩         ::=  :⟨simple_symbol⟩
```

Members of the ⟨*symbol*⟩ category starting with the character `@` or `.` are reserved for solver use. Solvers can use them respectively as identifiers for abstract values and solver generated function symbols other than abstract values.

S-expressions

```
⟨spec_constant⟩  ::=  ⟨numeral⟩ | ⟨decimal⟩ | ⟨hexadecimal⟩ | ⟨binary⟩ | ⟨string⟩
⟨s_expr⟩         ::=  ⟨spec_constant⟩ | ⟨symbol⟩ | ⟨reserved⟩ | ⟨keyword⟩
                  |   ( ⟨s_expr⟩* )
```

Identifiers

```
⟨index⟩       ::=  ⟨numeral⟩ | ⟨symbol⟩
⟨identifier⟩  ::=  ⟨symbol⟩ | ( _ ⟨symbol⟩ ⟨index⟩+ )
```

Sorts

```
⟨sort⟩  ::=  ⟨identifier⟩  |  ( ⟨identifier⟩ ⟨sort⟩+ )
```

Attributes

```
⟨attribute_value⟩  ::=  ⟨spec_constant⟩  |  ⟨symbol⟩  |  ( ⟨s_expr⟩* )
⟨attribute⟩        ::=  ⟨keyword⟩  |  ⟨keyword⟩ ⟨attribute_value⟩
```

Terms

```
⟨qual_identifier⟩  ::=  ⟨identifier⟩  |  ( as ⟨identifier⟩ ⟨sort⟩ )
⟨var_binding⟩      ::=  ( ⟨symbol⟩ ⟨term⟩ )
⟨sorted_var⟩       ::=  ( ⟨symbol⟩ ⟨sort⟩ )
⟨pattern⟩          ::=  ⟨symbol⟩  |  ( ⟨symbol⟩ ⟨symbol⟩+ )
⟨match_case⟩       ::=  ( ⟨pattern⟩ ⟨term⟩ )
⟨term⟩             ::=  ⟨spec_constant⟩
                     |  ⟨qual_identifier⟩
                     |  ( ⟨qual_identifier⟩ ⟨term⟩+ )
                     |  ( let ( ⟨var_binding⟩+ ) ⟨term⟩ )
                     |  ( forall ( ⟨sorted_var⟩+ ) ⟨term⟩ )
                     |  ( exists ( ⟨sorted_var⟩+ ) ⟨term⟩ )
                     |  ( match ⟨term⟩ ( ⟨match_case⟩+ ) )
                     |  ( ! ⟨term⟩ ⟨attribute⟩+ )
```

Theories

```
⟨sort_symbol_decl⟩        ::=  ( ⟨identifier⟩ ⟨numeral⟩ ⟨attribute⟩* )
⟨meta_spec_constant⟩      ::=  NUMERAL | DECIMAL | STRING
⟨fun_symbol_decl⟩         ::=  ( ⟨spec_constant⟩ ⟨sort⟩ ⟨attribute⟩* )
                           |   ( ⟨meta_spec_constant⟩ ⟨sort⟩ ⟨attribute⟩* )
                           |   ( ⟨identifier⟩ ⟨sort⟩+ ⟨attribute⟩* )
⟨par_fun_symbol_decl⟩     ::=  ⟨fun_symbol_decl⟩
                           |   ( par ( ⟨symbol⟩+ ) ( ⟨identifier⟩ ⟨sort⟩+ ⟨attribute⟩* ) )
⟨theory_attribute⟩        ::=  :sorts ( ⟨sort_symbol_decl⟩+ )
                           |   :funs ( ⟨par_fun_symbol_decl⟩+ )
                           |   :sorts-description ⟨string⟩
                           |   :funs-description ⟨string⟩
                           |   :definition ⟨string⟩
                           |   :values ⟨string⟩
                           |   :notes ⟨string⟩
                           |   ⟨attribute⟩
⟨theory_decl⟩             ::=  ( theory ⟨symbol⟩ ⟨theory_attribute⟩+ )
```

Logics

```
⟨logic_attribute⟩    :=   :theories ( ⟨symbol⟩+ )
                      |   :language ⟨string⟩
                      |   :extensions ⟨string⟩
                      |   :values ⟨string⟩
                      |   :notes ⟨string⟩
                      |   ⟨attribute⟩
⟨logic⟩              ::=  ( logic ⟨symbol⟩ ⟨logic_attribute⟩+ )
```

Info flags

```
⟨info_flag⟩  ::=  :all-statistics | :assertion-stack-levels | :authors
               |  :error-behavior | :name | :reason-unknown
               |  :version | ⟨keyword⟩
```

Command options

```
⟨b_value⟩  ::=  true | false

⟨option⟩   ::=  :diagnostic-output-channel ⟨string⟩
            |   :global-declarations ⟨b_value⟩
            |   :interactive-mode ⟨b_value⟩
            |   :print-success ⟨b_value⟩
            |   :produce-assertions ⟨b_value⟩
            |   :produce-assignments ⟨b_value⟩
            |   :produce-models ⟨b_value⟩
            |   :produce-proofs ⟨b_value⟩
            |   :produce-unsat-assumptions ⟨b_value⟩
            |   :produce-unsat-cores ⟨b_value⟩
            |   :random-seed ⟨numeral⟩
            |   :regular-output-channel ⟨string⟩
            |   :reproducible-resource-limit ⟨numeral⟩
            |   :verbosity ⟨numeral⟩
            |   ⟨attribute⟩
```

Commands

```
⟨sort_dec⟩          ::=  ( ⟨symbol⟩ ⟨numeral⟩ )
⟨selector_dec⟩      ::=  ( ⟨symbol⟩ ⟨sort⟩ )
⟨constructor_dec⟩   ::=  ( ⟨symbol⟩ ⟨selector_dec⟩* )
⟨datatype_dec⟩      ::=  ( ⟨constructor_dec⟩+ )  | ( par ( ⟨symbol⟩+ ) ( ⟨constructor_dec⟩+ ) )
⟨function_dec⟩      ::=  ( ⟨symbol⟩ ( ⟨sorted_var⟩* ) ⟨sort⟩ )
⟨function_def⟩      ::=  ⟨symbol⟩ ( ⟨sorted_var⟩* ) ⟨sort⟩ ⟨term⟩
⟨prop_literal⟩      ::=  ⟨symbol⟩  |  ( not ⟨symbol⟩ )
⟨command⟩           ::=  ( assert ⟨term⟩ )
                     |   ( check-sat )
                     |   ( check-sat-assuming ( ⟨prop_literal⟩* ) )
                     |   ( declare-const ⟨symbol⟩ ⟨sort⟩ )
                     |   ( declare-datatype ⟨symbol⟩ ⟨datatype_dec⟩)
                     |   ( declare-datatypes ( ⟨sort_dec⟩^{n+1} ) ( ⟨datatype_dec⟩^{n+1} ) )
                     |   ( declare-fun ⟨symbol⟩ ( ⟨sort⟩* ) ⟨sort⟩ )
                     |   ( declare-sort ⟨symbol⟩ ⟨numeral⟩ )
                     |   ( define-fun ⟨function_def⟩ )
                     |   ( define-fun-rec ⟨function_def⟩ )
                     |   ( define-funs-rec ( ⟨function_dec⟩^{n+1} ) ( ⟨term⟩^{n+1} ) )
                     |   ( define-sort ⟨symbol⟩ ( ⟨symbol⟩* ) ⟨sort⟩ )
                     |   ( echo ⟨string⟩ )
                     |   ( exit )
                     |   ( get-assertions )
                     |   ( get-assignment )
                     |   ( get-info ⟨info_flag⟩ )
                     |   ( get-model )
                     |   ( get-option ⟨keyword⟩ )
                     |   ( get-proof )
                     |   ( get-unsat-assumptions )
                     |   ( get-unsat-core )
                     |   ( get-value ( ⟨term⟩+ ) )
                     |   ( pop ⟨numeral⟩ )
                     |   ( push ⟨numeral⟩ )
                     |   ( reset )
                     |   ( reset-assertions )
                     |   ( set-info ⟨attribute⟩ )
                     |   ( set-logic ⟨symbol⟩ )
                     |   ( set-option ⟨option⟩ )
⟨script⟩            ::=  ⟨command⟩*
```

Command responses

⟨error-behavior⟩	::=	`immediate-exit` | `continued-execution`
⟨reason-unknown⟩	::=	`memout` | `incomplete` | ⟨s_expr⟩
⟨model_response⟩	::=	`(` `define-fun` ⟨function_def⟩ `)` | `(` `define-fun-rec` ⟨function_def⟩ `)`
	|	`(` `define-funs-rec` `(` ⟨function_dec⟩$^{n+1}$ `)` `(` ⟨term⟩$^{n+1}$ `)` `)`
⟨info_response⟩	::=	`:assertion-stack-levels` ⟨numeral⟩
	|	`:authors` ⟨string⟩
	|	`:error-behavior` ⟨error-behavior⟩
	|	`:name` ⟨string⟩
	|	`:reason-unknown` ⟨reason-unknown⟩
	|	`:version` ⟨string⟩
	|	⟨attribute⟩
⟨valuation_pair⟩	::=	`(` ⟨term⟩ ⟨term⟩ `)`
⟨t_valuation_pair⟩	::=	`(` ⟨symbol⟩ ⟨b_value⟩ `)`
⟨check_sat_response⟩	::=	`sat` | `unsat` | `unknown`
⟨echo_response⟩	::=	⟨string⟩
⟨get_assertions_response⟩	::=	`(` ⟨term⟩* `)`
⟨get_assignment_response⟩	::=	`(` ⟨t_valuation_pair⟩* `)`
⟨get_info_response⟩	::=	`(` ⟨info_response⟩$^{+}$ `)`
⟨get_model_response⟩	::=	`(` ⟨model_response⟩* `)`
⟨get_option_response⟩	::=	⟨attribute_value⟩
⟨get_proof_response⟩	::=	⟨s_expr⟩
⟨get_unsat_assump_response⟩	::=	`(` ⟨symbol⟩* `)`
⟨get_unsat_core_response⟩	::=	`(` ⟨symbol⟩* `)`
⟨get_value_response⟩	::=	`(` ⟨valuation_pair⟩$^{+}$ `)`
⟨specific_success_response⟩	::=	⟨check_sat_response⟩ | ⟨echo_response⟩
	|	⟨get_assertions_response⟩ | ⟨get_assignment_response⟩
	|	⟨get_info_response⟩ | ⟨get_model_response⟩
	|	⟨get_option_response⟩ | ⟨get_proof_response⟩
	|	⟨get_unsat_assumptions_response⟩
	|	⟨get_unsat_core_response⟩ | ⟨get_value_response⟩
⟨general_response⟩	::=	`success` | ⟨specific_success_response⟩
	|	`unsupported` | `(` `error` ⟨string⟩ `)`