Grammar

The Logtalk grammar is here described using Backus-Naur Form syntax. Non-terminal symbols in *italics* have the definition found in the ISO Prolog Standard. Terminal symbols are represented in a `fixed width font` and between "".

Compilation units

entity ::=
    object |
    category |
    protocol

Object definition

object ::=
    begin_object_directive [object_directives] [clauses] end_object_directive.

begin_object_directive ::=
    "`:- object(`" object_identifier ["`,`" object_relations] "`).`"

end_object_directive ::=
    "`:- end_object.`"

object_relations ::=
    prototype_relations |
    non_prototype_relations

prototype_relations ::=
    prototype_relation |
    prototype_relation "`,`" prototype_relations

prototype_relation ::=
    implements_protocols |
    imports_categories |
    extends_objects

non_prototype_relations ::=
    non_prototype_relation |
    non_prototype_relation "`,`" non_prototype_relations

non_prototype_relation ::=
    implements_protocols |
    imports_categories |
    instantiates_classes |
    specializes_classes

Category definition

category ::=
    begin_category_directive [category_directives] [clauses] end_category_directive.

begin_category_directive ::=
    ":- category(" category_identifier ["," category_relations] ")."

end_category_directive ::=
    ":- end_category."

category_relations ::=
    category_relation |
    category_relation "," category_relations

category_relation ::=
    implements_protocols |
    imports_categories

Protocol definition

protocol ::=
    begin_protocol_directive [protocol_directives] end_protocol_directive.

begin_protocol_directive ::=
    ":- protocol(" protocol_identifier ["," extends_protocols] ")."

end_protocol_directive ::=
    ":- end_protocol."

Entity relations

implements_protocols ::=
    "implements(" implemented_protocols ")"

extends_protocols ::=
    "extends(" extended_protocols ")"

imports_categories ::=
    "imports(" imported_categories ")"

extends_objects ::=
    "extends(" extended_objects ")"

instantiates_classes ::=
    "instantiates(" instantiated_objects ")"

specializes_classes ::=
    "specializes(" specialized_objects ")"

Implemented protocols

implemented_protocols ::=
    implemented_protocol |
    implemented_protocol_sequence |
    implemented_protocol_list

implemented_protocol ::=
    protocol_identifier |
    scope "::" protocol_identifier

implemented_protocol_sequence ::=
    implemented_protocol |
    implemented_protocol "," implemented_protocol_sequence

implemented_protocol_list ::=
    "[" implemented_protocol_sequence "]"

Extended protocols

extended_protocols ::=
    extended_protocol |
    extended_protocol_sequence |
    extended_protocol_list

extended_protocol ::=
    protocol_identifier |
    scope "::" protocol_identifier

extended_protocol_sequence ::=
    extended_protocol |
    extended_protocol "," extended_protocol_sequence

extended_protocol_list ::=
    "[" extended_protocol_sequence "]"

Imported categories

imported_categories ::=
    imported_category |
    imported_category_sequence |
    imported_category_list

imported_category ::=
    category_identifier |
    scope "::" category_identifier

imported_category_sequence ::=
    imported_category |
    imported_category "," imported_category_sequence

imported_category_list ::=
    "[" imported_category_sequence "]"

Extended objects

extended_objects ::=
    extended_object |
    extended_object_sequence |
    extended_object_list

extended_object ::=
    object_identifier |
    scope "::" object_identifier

extended_object_sequence ::=
    extended_object |
    extended_object "," extended_object_sequence

extended_object_list ::=
    "[" extended_object_sequence "]"

Instantiated objects

instantiated_objects ::=
    instantiated_object |
    instantiated_object_sequence |
    instantiated_object_list

instantiated_object ::=
    object_identifier |
    scope "::" object_identifier

instantiated_object_sequence ::=
    instantiated_object
    instantiated_object "," instantiated_object_sequence |

instantiated_object_list ::=
    "[" instantiated_object_sequence "]"

Specialized objects

specialized_objects ::=
    specialized_object |
    specialized_object_sequence |
    specialized_object_list

specialized_object ::=
    object_identifier |
    scope "::" object_identifier

specialized_object_sequence ::=
    specialized_object |
    specialized_object "," specialized_object_sequence

specialized_object_list ::=
    "[" specialized_object_sequence "]"

Entity scope

scope ::=
"public" |
"protected" |
"private"

Entity identifiers

entity_identifiers ::=
entity_identifier |
entity_identifier_sequence |
entity_identifier_list

entity_identifier ::=
object_identifier |
protocol_identifier |
category_identifier

entity_identifier_sequence ::=
entity_identifier |
entity_identifier "," entity_identifier_sequence

entity_identifier_list ::=
"[" entity_identifier_sequence "]"

Object identifiers

object_identifiers ::=
object_identifier |
object_identifier_sequence |
object_identifier_list

object_identifier ::=
atom |
compound

object_identifier_sequence ::=
object_identifier |
object_identifier "," object_identifier_sequence

object_identifier_list ::=
"[" object_identifier_sequence "]"

Category identifiers

category_identifiers ::=
    category_identifier |
    category_identifier_sequence |
    category_identifier_list

category_identifier ::=
    atom

category_identifier_sequence ::=
    category_identifier |
    category_identifier "," category_identifier_sequence

category_identifier_list ::=
    "[" category_identifier_sequence "]"

Protocol identifiers

protocol_identifiers ::=
    protocol_identifier |
    protocol_identifier_sequence |
    protocol_identifier_list

protocol_identifier ::=
    atom

protocol_identifier_sequence ::=
    protocol_identifier |
    protocol_identifier "," protocol_identifier_sequence

protocol_identifier_list ::=
    "[" protocol_identifier_sequence "]"

Source file names

source_file_names ::=
    source_file_name |
    source_file_name_list

source_file_name ::=
    atom |
    library_source_file_name

library_source_file_name ::=
    library_name "`(`" *atom* "`)`"

library_name ::=
    atom

source_file_name_sequence ::=
    source_file_name |
    source_file_name "`,`" source_file_name_sequence

source_file_name_list ::=
    "`[`" source_file_name_sequence "`]`"

Directives

Source file directives

source_file_directives ::=
    source_file_directive |
    source_file_directive source_file_directives

source_file_directive ::=
    "`:- encoding(`" *atom* "`).`" |
    initialization_directive |
    operator_directive

Object directives

object_directives ::=
    object_directive |
    object_directive object_directives

object_directive ::=
    initialization_directive |
    "`:- threaded.`" |
    "`:- dynamic.`" |
    "`:- uses(`" object_identifier "`).`" |
    "`:- calls(`" protocol_identifiers "`).`" |
    "`:- info(`" info_list "`).`" |
    predicate_directives

Category directives

category_directives ::=
    category_directive |
    category_directive category_directives

category_directive ::=
    initialization_directive |
    ":- uses(" object_identifier ")." |
    ":- calls(" protocol_identifiers ")." |
    ":- dynamic." |
    ":- info(" info_list ")." |
    predicate_directives

Protocol directives

protocol_directives ::=
    protocol_directive |
    protocol_directive protocol_directives

protocol_directive ::=
    initialization_directive |
    ":- dynamic." |
    ":- info(" info_list ")." |
    predicate_directives

Predicate directives

predicate_directives ::=
    predicate_directive |
    predicate_directive predicate_directives

predicate_directive ::=
    alias_directive |
    atomic_directive |
    uses_directive |
    scope_directive |
    mode_directive |
    meta_predicate_directive |
    info_directive |
    dynamic_directive |
    discontiguous_directive |
    operator_directive

alias_directive ::=
    ":- alias(" entity_identifier "," predicate_indicator "," predicate_indicator ")." |
    ":- alias(" entity_identifier "," non_terminal_indicator "," non_terminal_indicator ")."

atomic_directive ::=
    ":- atomic(" predicate_indicator ")." |
    ":- atomic(" non_terminal_indicator ")."

uses_directive ::=
    ":- uses(" object_identifier "," predicate_indicator_alias_list ")."

scope_directive ::=
    ":- public(" predicate_indicator_term | non_terminal_indicator_term ")." |
    ":- protected(" predicate_indicator_term | non_terminal_indicator_term ")." |
    ":- private(" predicate_indicator_term | non_terminal_indicator_term ")."

mode_directive ::=
    ":- mode(" predicate_mode_term | non_terminal_mode_term "," number_of_solutions ")."

meta_predicate_directive ::=
    ":- meta_predicate(" meta_predicate_mode_indicator ")."

info_directive ::=
    ":- info(" predicate_indicator | non_terminal_indicator "," info_list ")."

dynamic_directive ::=
    ":- dynamic(" predicate_indicator_term | non_terminal_indicator_term ")." |

discontiguous_directive ::=
    ":- discontiguous(" predicate_indicator_term | non_terminal_indicator_term ")." |

predicate_indicator_term ::=
    predicate_indicator |
    predicate_indicator_sequence |
    predicate_indicator_list

predicate_indicator_sequence ::=
    predicate_indicator |
    predicate_indicator "," predicate_indicator_sequence

predicate_indicator_list ::=
    "[" predicate_indicator_sequence "]"

predicate_indicator_alias ::=
    predicate_indicator |
    predicate_indicator "::" *predicate_indicator*

predicate_indicator_alias_sequence ::=
    predicate_indicator_alias |
    predicate_indicator_alias "," predicate_indicator_alias_sequence

predicate_indicator_alias_list ::=
    "[" predicate_indicator_alias_sequence "]"

non_terminal_indicator_term ::=
    non_terminal_indicator |
    non_terminal_indicator_sequence |
    non_terminal_indicator_list

non_terminal_indicator_sequence ::=
    non_terminal_indicator |
    non_terminal_indicator "," non_terminal_indicator_sequence

non_terminal_indicator_list ::=
    "[" non_terminal_indicator_sequence "]"

non_terminal_indicator ::=
    functor "//" *arity*

predicate_mode_term ::=
    atom "(" mode_terms ")"

non_terminal_mode_term ::=
    atom "(" mode_terms ")"

mode_terms ::=
    mode_term |
    mode_term "," mode_terms

mode_term ::=
    "@" [type] | "+" [type] | "-" [type] | "?" [type]

type ::=
    prolog_type | logtalk_type | user_defined_type

prolog_type ::=
    "term" | "nonvar" | "var" |
    "compound" | "ground" | "callable" | "list" |
    "atomic" | "atom" |
    "number" | "integer" | "float"

logtalk_type ::=
    "object" | "category" | "protocol" |
    "event"

user_defined_type ::=
    atom |
    compound

number_of_solutions ::=
    "zero" | "zero_or_one" | "zero_or_more" | "one" | "one_or_more" | "error"

meta_predicate_mode_indicator ::=
    atom "(" meta_predicate_terms ")"

meta_predicate_terms ::=
    meta_predicate_term |
    meta_predicate_term "," meta_predicate_terms

meta_predicate_term ::=
    "::" | "*" | *integer*

info_list ::=
    "[]" |
    "[" info_item "is" *nonvar* "|" info_list "]"

info_item ::=
    "comment" | "remarks" |
    "author" | "version" | "date" |
    "copyright" | "license" |
    "parameters" | "parnames" |
    "arguments" | "argnames" |
    "definition" | "redefinition" | "allocation" |
    "examples" | "exceptions" |
    atom

Clauses and goals

goal ::=
    message_call |
    external_call |
    callable

message_call ::=
    message_to_object |
    message_to_self |
    message_to_super

message_to_object ::=
    receivers "::" messages

message_to_self ::=
    "::" messages

message_to_super ::=
    "^^" message

messages ::=
    message |
    "(" message "," messages ")" |
    "(" message ";" messages ")"

message ::=
    callable |
    variable

receivers ::=
    receiver |
    "(" receiver "," receivers ")" |
    "(" receiver ";" receivers ")"

receiver ::=
    object_identifier |
    variable

external_call ::=
    "{" *callable* "}"

Entity properties

category_property ::=
    "static" |
    "dynamic" |
    "built_in"

object_property ::=
    "static" |
    "dynamic" |
    "built_in"

protocol_property ::=
    "static" |
    "dynamic" |
    "built_in"

Predicate properties

predicate_property ::=
    "static" |
    "dynamic" |
    "private" |
    "protected" |
    "public" |
    "atomic" |
    "built_in" |
    "declared_in(" entity_identifier ")" |
    "defined_in(" object_identifier | category_identifier ")" |
    "meta_predicate(" meta_predicate_mode_indicator ")" |
    "alias(" callable ")" |
    "non_terminal(" non_terminal_indicator ")"

XHTML + CSS Last updated on: September 17, 2006