**Introduction**

Robotspeak is a programming language for controlling the actions and movements of a small robot, ‘Runner’, within a virtual 2D environment. The language defines only a small set of instructions and sensors, reflecting deliberate constraints intending to showcase programming design under limitations.

**Lexical Elements**

Character set: ASCII

Whitespace: The ASCII space, tab or new line characters; separates tokens, otherwise ignored.

Comments: The ASCII at character; start with @ and continues until the end of the line

Identifiers: The ASCII letters, digits and the ASCII underscore; must begin with a letter. Identifiers are variables of Boolean type. All variables are implicitly declared on first assignment.

**Keywords and Tokens**

Actions: MOVE\_FORWARD, TURN\_LEFT, TURN\_RIGHT, PICK\_KEY, OPEN\_DOOR

Control and Operators: LOAD, IF, OTHERWISE, WHILE, END, AND, OR

Literals: TRUE, FALSE

Sensors: FRONT\_IS\_CLEAR, ON\_KEY, AT\_DOOR, AT\_EXIT

**Grammar (EBNF)**

|  |
| --- |
| program ::= "LOAD" env\_id { stmt } "END" ;  stmt ::= action\_stmt  | if\_stmt  | while\_stmt  | assign\_stmt ;  action\_stmt ::= "MOVE\_FORWARD" | "TURN\_LEFT" | "TURN\_RIGHT"  | "PICK\_KEY" | "OPEN\_DOOR" ;  if\_stmt ::= "IF" expr { stmt }  [ "OTHERWISE" { stmt } ]  "END" ;  while\_stmt ::= "WHILE" expr { stmt } "END" ;  assign\_stmt ::= identifier ":=" expr ;  expr ::= term { "OR" term } ;  term ::= factor { "AND" factor } ;  factor ::= "(" expr ")"  | literal  | sensor  | identifier ;  literal ::= "TRUE" | "FALSE" ;  sensor ::= "FRONT\_IS\_CLEAR" | "ON\_KEY" | "AT\_DOOR" | "AT\_EXIT" ;  env\_id ::= "1" | "2" | "3" ; |

**Semantics**

**Program structure**

Programs MUST begin with a LOAD statement specifying the environment (1, 2 or 3) and MUST terminate with END. Statements are executed sequentially in the order they appear top to bottom.

**Actions**

**MOVE\_FORWARD:** Advances the runner by one tile in the current facing direction if FRONT\_IS\_CLEAR is true, otherwise no effect.

**TURN\_LEFT:** Rotates the Runner’s facing direction 90 ° counterclockwise; always succeeds.

**TURN\_RIGHT:** Rotates the Runner’s facing direction 90 ° clockwise; always succeeds.

**PICK\_KEY:** Picks up a key if ON\_KEY is true, otherwise no effect.

**OPEN\_DOOR:** Outputs “You escaped!” to the terminal and terminates the program if AT\_DOOR is true and a correct key has been previously picked up, or AT\_EXIT is true, otherwise no effect. If the program reaches the final END without OPEN\_DOOR ever successfully executing, the program silently terminates.

**Sensors**

**FRONT\_IS\_CLEAR:** Assigned true if the next tile in the Runner’s facing direction is not a wall, otherwise assigned false.

ON\_KEY: Assigned true if the tile the Runner is on has a key, otherwise assigned false.

AT\_DOOR: Assigned true if the Runner is on a door tile, otherwise assigned false.

AT\_EXIT: Assigned true if the Runner is on an exit tile, otherwise assigned false.

**Control**

**IF…OTHERWISE…END:** Executes the first block if the condition is true, otherwise executes the OTHERWISE block if present.

**WHILE…END:** Executes the code block if the condition is true, and perpetually does so until the condition is false after the end of an executed cycle.

**Operator precedence:** AND binds tighter than OR. Both operators are left-associative. Parenthesis override precedence.

**Error model**

**Syntax errors:** Detected during parsing. A statement with a syntax error MUST NOT be executed. Examples: Unknown or invalid token, missing or unmatched END, LOAD not appearing as the first statement or appearing more than once, malformed expressions.

A syntax error produces a diagnostic message of the form:

|  |
| --- |
| SyntaxError: <description> at line <n> |

**Runtime errors:** Detectedduring runtime. Examples: referencing a variable before it is assigned.

A runtime error produces a diagnostic message of the form:

|  |
| --- |
| RuntimeError: <description> at line <n> |