Appendix: Quick reference

The bare bones

Comments:

From %% to end of line.

Special characters:

#, $, @, ~, *, |, \, parentheses, brackets, and braces. Prefix by \ to print.

Rule definitions:

- The rule head is in the leftmost column.
- The rule body is a sequence of statements, not in the leftmost column. They form a conjunction.
- Multiple rule definitions for the same predicate are tried in order. They form a disjunction.
- Queries succeed or fail. Failure causes backtracking.

Statements:

- Queries (normal, negated, or multi-) to predicates,
- text to be printed, or
- special syntax.

Value syntax:

#abc

Object name.

@abc

Dictionary word. The @ is optional inside list expressions. Special words to represent non-printable keys: @\n, @\b, @\s, @\u, @\d, @\l, @\r.

$abc

Variable.

123

Number (the valid range is 0–16383).

[]

Empty list.

[*element1 element2 element3 ...*]

Complete list.

[*element1 element2 element3 ...* | *tail*]

Partial list.

{ ... }

Closure.

*

Current topic.

The current topic is set with an object name in the leftmost column.

Unification:

- Arguments are passed by unification.
- Simple values unify with themselves.
- Lists unify when each element unifies.
- Unbound variables unify (with values or unbound variables) by binding.

Special syntax

{ ... }

Conjunction.

~{ ... }

Negated conjunction.

... (or) ... (or) ...

Disjunction.

(if) ... (then) ... (elseif) ... (then) ... (else) ... (endif)

Conditions are evaluated at most once. Missing else-clause is assumed to be empty, i.e. succeeds.

(select) ... (or) ... (or) ... (stopping)

Branches entered one at a time, then the final branch repeats.

(select) ... (or) ... (or) ... (cycling)

Branches entered one at a time, then the cycle repeats.

(select) ... (or) ... (or) ... (at random)

Branches entered at random, avoiding repetition.

(select) ... (or) ... (or) ... (purely at random)

Branches entered at random, allowing repetition.

(select) ... (or) ... (or) ... (then at random)

First like (select) ... (stopping), then like (select) ... (at random).

(select) ... (or) ... (or) ... (then purely at random)

First like (select) ... (stopping), then like (select) ... (purely at random).

(exhaust) *statement*

Backtracks over all solutions to the statement (typically a block).

(collect $Element) ... (into $List)

Backtracks over all solutions to the inner expression. Values bound to $Element are collected in order and unified with $List.

(collect words) ... (into $List)

Backtracks over all solutions to the inner expression, grabbing all output. Printed words are diverted into $List, possibly out of order.

(accumulate $Element) ... (into $Sum)

Backtracks over all solutions to the inner expression. Values bound to $Element are added and their sum is unified with $Sum.

(determine object $Obj) ... (from words) ... (matching all of $List)

Backtracks over every object $Obj that makes the first inner expression succeed, and for which the second expression (when exhausted) emits at least every word in $List.

(stoppable) *statement*

The inner statement executes, succeeding at most once. The (stop) built-in breaks out of the innermost stoppable environment. The stoppable environment itself always succeeds.

(span $Class) *statement*

The inner statement executes, succeeding at most once. All output is rendered according to the given style class.

(div $Class) *statement*

The inner statement executes, succeeding at most once. All output is rendered into a rectangular area according to the given style class.

(status bar $Class) *statement*

Like (div $), but the output is rendered into the top status area, which is created if necessary.

(inline status bar $Class) *statement*

Like (div $), but the output is rendered into an inline status area. The previous inline status area, if any, is removed from display.

(link) *statement*

The inner statement executes, succeeding at most once. The interpreter displays the output, optionally in the form of a hyperlink. If the hyperlink is selected by the player, the output from the inner statement is appended to the current input buffer, which is submitted.

(link $Words) *statement*

The inner statement executes, succeeding at most once. The interpreter displays the output, optionally in the form of a hyperlink. If the hyperlink is selected by the player, the given $Words are appended to the current input buffer, which is submitted.

(link resource $Id) *statement*

The inner statement executes, succeeding at most once. The interpreter displays the output, optionally in the form of a hyperlink. The hyperlink leads to the resource identified by $Id.

(log) *statement*

If running in the debugger, execute the inner statement in a stoppable environment. The output appears between line breaks, in a distinct style.

(now) *pseudo-query*

Updates a dynamic predicate.

(just)

Removes choice-points created since the current predicate was queried.

(global variable (name of predicate $))

Declares a global variable.

(interface (name of predicate))

Declares an interface, i.e. the intended use of a predicate.

(generate $N (name of predicate $))

Creates $N anonymous objects, for which the predicate will succeed.

@(...) ...

Defines an access predicate. Queries or definitions matching the rule head are transformed into the rule body at compile-time.

Built-in predicates

The list is sorted alphabetically, considering just the non-parameter words.

($X = $Y)

Unifies $X with $Y.

($X < $Y)

Succeeds if $X and $Y are numbers, and $X is strictly less than $Y.

($X > $Y)

Succeeds if $X and $Y are numbers, and $X is strictly greater than $Y.

(append $A $B $AB)

Unifies $AB with the concatenation of $A (which must be bound) and $B.

(bold)

Enables bold text.

(bound $X)

Succeeds if $X is bound to a value.

(breakpoint)

Suspends execution (if running in the debugger).

(clear)

Clears the main window, but not the top status area.

(clear all)

Clears the screen and disables the top status area.

(clear div)

Clears or hides the current div.

(clear links)

Transforms all hyperlinks into plain text, except in the status areas.

(clear old)

Clears the screen from all text that the player has had a chance to read.

(compiler version)

Prints the name and version of the compiler.

(display memory statistics)

Displays a backend-specific line of memory usage statistics.

($X divided by $Y into $Z)

Unifies $Z with the quotient after dividing $X by $Y.

(embed resource $Id)

Displays the resource identified by $Id, embedded in the story text.

(empty $X)

Succeeds if $X is bound to an empty list.

(fail)

Fails. Equivalent in functionality to e.g. (1 = 2).

(fixed pitch)

Enables fixed-pitch text.

(fully bound $X)

Succeeds if $X is bound to a value, and—in case of a list—contains only fully bound elements.

(get input $)

Reads a line of input from the player. Returns a list of words.

(get key $)

Waits for the player to press a key. Returns a single-character word.

($X has parent $Y)

Dynamic predicate that succeeds when $X is a direct child of $Y in the object tree. Either parameter can be unbound.

(interpreter can embed $Id)

Succeeds if the current interpreter supports the resource identified by $Id, and is able to display it using (embed resource $Id) without falling back on just printing the alt-text.

(interpreter supports inline status bar)

Succeeds if the current interpreter supports inline status areas.

(interpreter supports links)

Succeeds if the current interpreter claims to support hyperlinks, and they are currently enabled. This can change at runtime, for instance if a game is saved and subsequently restored on a different interpreter.

(interpreter supports quit)

Succeeds if the current interpreter supports quit in a way that is meaningful to the player. This can change at runtime, for instance if a game is saved and subsequently restored on a different interpreter.

(interpreter supports status bar)

Succeeds if the current interpreter supports the top status area.

(interpreter supports undo)

Succeeds if the current interpreter claims to support undo. This can change at runtime, for instance if a game is saved and subsequently restored on a different interpreter.

($X is one of $Y)

Unifies $X with each element of $Y in turn.

(italic)

Enables italic text.

(join words $List into $Word)

Concatenates the dictionary words (or numbers) in $List into a new dictionary word (or number), and unifies the result with $Word.

(line)

Requests a line break.

(list $X)

Succeeds if $X is bound to a list (empty or non-empty).

($X minus $Y into $Z)

Unifies $Z with the difference between $X and $Y.

($X modulo $Y into $Z)

Unifies $Z with the remainder after dividing $X by $Y.

(nonempty $X)

Succeeds if $X is bound to an non-empty list.

(no space)

Inhibits automatic whitespace before the next word or punctuation mark.

(number $X)

Succeeds if $X is bound to a number.

(object $X)

Checks if $X is an object, or—in a multi-query—backtracks over every object.

(par)

Requests a paragraph break.

(progress bar $ of $)

Draws a progress bar scaled to fit the width of the current div.

($X plus $Y into $Z)

Unifies $Z with the sum of $X and $Y.

(quit)

Immediately terminates the interpreter.

(random from $X to $Y into $Z)

Unifies $Z with a random number in the range $X to $Y inclusive.

(repeat forever)

Provides an infinite supply of choice points. Generally invoked with a multi-query.

(restart)

Restarts the program.

(restore)

Restores a saved game (the interpreter asks for a filename).

(reverse)

Enables reverse-video text.

(roman)

Disables all text styles (bold, italic, reverse, and fixed pitch).

(save $ComingBack)

Saves the current game (the interpreter asks for a filename). Unifies $ComingBack with 0 after a successful save, 1 after a successful restore.

(save undo $ComingBack)

Saves the current program state in memory. Unifies $ComingBack with 0 after a successful save, 1 after a successful restore.

(script off)

Disables transcription.

(script on)

Enables transcription (the interpreter asks for a filename).

(serial number)

Prints the serial number (compilation date) of the current program.

(space)

Forces whitespace before the next word or punctuation mark.

(space $N)

Prints $N space characters.

(split $X by $Y into $Left and $Right)

Splits $X into two halves around each occurrence of $Y or any member of $Y.

(split word $Word into $List)

Converts the dictionary word (or number) $Word into a list of its constituent characters, and unifies the result with $List.

(stop)

Breaks out of the innermost (stoppable) environment.

($X times $Y into $Z)

Unifies $Z with the product of $X and $Y.

(trace off)

Disables query tracing.

(trace on)

Enables query tracing.

(undo)

Restores the program state at the time of the latest (save undo 0).

(unknown word $X)

Succeeds if $X is bound to a word that wasn't found in the game dictionary.

(unstyle)

Select the default text style for the current division.

(uppercase)

Convert the next printed character to uppercase.

(word $X)

Succeeds if $X is bound to a dictionary word.

Entry points and metadata predicates

(error $ErrorCode entry point)

Execution restarts here when a fatal error has occurred.

(program entry point)

Normal execution starts here.

(story author)

Metadata: Defines the author of the story.

(story blurb)

Metadata: Defines the blurb for the story.

(story ifid)

Metadata: Defines the IFID of the story.

(story noun)

Metadata: Defines the noun (also known as the headline) of the story.

(story release $N)

Metadata: Defines the release number of the story.

(story title)

Metadata: Defines the title of the story.

(library version)

Defines the library version and is used to identify the library source code file.

(removable word endings)

Defines one or more word endings that can be removed when parsing user input.

(style class $Name)

Associates one or more style attributes with the given class name.

(define resource $Id)

Defines the location (local filename or URL) and alt-text of a resource.

Back to the Table of Contents