SR5 - Phrase Definitions A/sr5

Purpose

The phrases making up the Inform language, and in terms of which all other phrases and rules are defined; and the final sign-off of the Standard Rules extension, including its minimal documentation.

A/sr5.§2-12 Say phrases; §13 Using the list-writer; §14 Text substitutions using the list-writer; §15 Grouping in the list-writer; §16 Lists written but not by the list-writer; §17 Filtering in the list-writer; §18-25 Values; §26-27 Tables; §28 Indexed text; §29-30 Matching text; §31 Replacing text; §32 Casing of text; §33-41 Lists; §42-43 Using external resources; §44-53 Control phrases; §54-60 Actions, activities and rules; §61-63 Rules; §64-76 The model world; §77-80 Understanding; §81-82 Message support; §83-95 Miscellaneous other phrases

---

§**1.** Our last task is to create the phrases: more or less all of them, but that does need a little qualification. NI has no phrase definitions built in, but it does contain assumptions about how "say ...", "repeat ...", "let ...", "otherwise ..." and "end ..." will behave when defined: we would not be allowed to call these something else, or redefine them in fundamentally different ways. Apart from that, we are more or less free.

Most of these phrases are defined in terms of I6 code, using the `(-` and `-)` notation – it would be too cumbersome to use the "... translates into I6 as ..." verb for this, too. The fact that phrases are not so much translated as transliterated was one source of early criticism of Inform 7. Phrases appeared to have very simplistic definitions, with the natural language simply being a verbose description of obviously equivalent I6 code. However, the simplicity is misleading, because the definitions below tend to conceal where the complexity of the translation process suddenly increases. If the preamble includes "(c - condition)", and the definition includes the expansion `{c}`, then the text forming c is translated in a way much more profound than any simple substitution process could describe. Type-checking also complicates the code produced below, since NI automatically generates the code needed to perform run-time type checking at any point where doubt remains as to the phrase definition which must be used.

§**2. Say phrases.** We begin with saying phrases: the very first phrase to exist is the one printing a single piece of static text, which seems only appropriate for an IF system. We then have a general way to say any value...

```
Part SR5 - Phrasebook

Section SR5/1/1 - Saying - Values

To say (something - text)
    (documented at ph_say):
    (- print (PrintText) {something}; -).
To say (value - sayable value of kind K)
    (documented at phs_value):
    (- print ({-printing-routine:K}) {-pointer-to:value}; -).
```

§**3.** ...but despite that we override this to give numbers their own definition:

```
To say (something - number)
    (documented at phs_value):
    (- print (say__n={something}); -).
```

§**4.** And similarly for Unicode characters. It would be tidier to abstract this with a function call, but it would cost a function call.

Note that emitting a Unicode character requires different code on the Z-machine to Glulx; we have to handle this within I6 conditional compilation blocks because neither syntax will compile when I6 is compiling for the other VM.

```
To say (ch - unicode character) -- running on
    (documented at phs_unicode):
    (- #ifdef TARGET_ZCODE; @push self; self = {ch}; @print_unicode self; @pull self;
    #ifnot; if (unicode_gestalt_ok) glk_put_char_uni({ch}); else print "?"; #endif; -).
```

§**5.** Three little grace-notes for printing values: we can have numbers or times of day "in words", rather than given as digits, and we can produce an optional "s" where a number not equal to 1 has recently been printed. This is how "You see [X] balloon[s]." is handled: the printing of the value X sets the I6 variable `say__n` as a side-effect (see definition above) and the routine handling "[s]" looks at this variable to see whether to print an "s" or not.

```
To say (something - number) in words
    (documented at phs_numwords):
    (- print (number) say__n=({something}); -).
To say (something - time) in words
    (documented at phs_timewords):
    (- print (PrintTimeOfDayEnglish) {something}; -).
To say s
    (documented at phs_s):
    (- STextSubstitution(); -).
```

§**6.** Now we come to the fourth and last of the "say (something - K)" definitions in the SR, followed by six close variations. Note that say phrases are case sensitive on the first word, so that "to say a something" and "to say A something" are different.

A curiosity of the original I6 design, arising I think mostly from the need to save property memory in *Curses* (1993), the work of IF for which Inform 1 had been created, is that it lacks the `print (A) ...` syntax to match the other forms. The omission is made good by using a routine in the I6 library instead.

```
Section SR5/1/2 - Saying - Names with articles

To say a (something - object)
    (documented at phs_a):
    (- print (a) {something}; -).
To say an (something - object)
    (documented at phs_a):
    (- print (a) {something}; -).
To say A (something - object)
    (documented at phs_A):
    (- CIndefArt({something}); -).
To say An (something - object)
    (documented at phs_A):
    (- CIndefArt({something}); -).
To say the (something - object)
    (documented at phs_the):
    (- print (the) {something}; -).
To say The (something - object)
    (documented at phs_The):
    (- print (The) {something}; -).
```

§**7.** Now for "[if ...]", which expands into a rather assembly-language-like usage of `jump` statements, I6's form of goto. For instance, the text "[if the score is 10]It's ten![otherwise]It's not ten, alas." compiles thus:

```
if (~~(score == 10)) jump L_Say3;
    ...
jump L_SayX2; .L_Say3;
    ...
.L_Say4; .L_SayX2;
```

Though labels actually have local namespaces in I6 routines, we use globally unique labels throughout the whole program: compiling the same phrase again would involve say labels 5 and 6 and "say exit" label 3. This example text demonstrates the reason we `jump` about, rather than making use of `if... else...` and bracing groups of statements: it is legal in I7 either to conclude with or to omit the "[end if]". (If statements in I6 compile to jump instructions in any event, and on our virtual machines there is no speed penalty for branches.) We also need the same definitions to accommodate what amounts to a switch statement. The trickier text "[if the score is 10]It's ten![otherwise if the score is 8]It's eight?[otherwise]It's not ten, alas." comes out as:

```
if (~~(score == 10)) jump L_Say5;
    ...
jump L_SayX3; .L_Say5; if (~~(score == 8)) jump L_Say6;
    ...
jump L_SayX3; .L_Say6;
    ...
.L_Say7; .L_SayX3;
```

In either form of the construct, control passes into at most one of the pieces of text. The terminal labels (the two on the final line) are automatically generated; often – when there is a simple "otherwise" or "end if" to conclude the construct – they are not needed, but labels are quick to process in I6, are soon discarded from I6's memory when not needed any more, and compile no code.

We assume in each case that the next say label number to be free is always the start of the next block, and that the next say exit label number is always the one at the end of the current construct. This is true because NI does not allow "say if" to be nested.

(The use of `{-erase}` below only tidies up the white space to set out the compiled I6 code neatly, and has no effect on the eventual story file.)

```
Section SR5/1/3 - Saying - Say if and otherwise

To say if (c - condition)
    (documented at phs_if): (- {-erase}
    if (~~({c})) jump {-next-label:Say};
        -).
To say unless (c - condition)
    (documented at phs_unless): (- {-erase}
    if ({c}) jump {-next-label:Say};
        -).
To say otherwise/else if (c - condition)
    (documented at phs_elseif): (- {-erase}
    jump {-next-label:SayX}; .{-label:Say}; if (~~({c})) jump {-next-label:Say};
        -).
To say otherwise/else unless (c - condition)
    (documented at phs_elseunless): (- {-erase}
    jump {-next-label:SayX}; .{-label:Say}; if ({c}) jump {-next-label:Say};
        -).
To say otherwise
    (documented at phs_otherwise): (- {-erase}
    jump {-next-label:SayX}; .{-label:Say};
```

```
        -).
To say else
    (documented at phs_otherwise): (- {-erase}
    jump {-next-label:SayX}; .{-label:Say};
        -).
To say end if
    (documented at phs_endif): (- {-erase}
    .{-label:Say}; .{-label:SayX};
        -).
To say end unless
    (documented at phs_endunless): (- {-erase}
    .{-label:Say}; .{-label:SayX};
        -).
```

§**8.** The other control structure: the random variations form of saying. This part of the Standard Rules was in effect contributed by the community: it reimplements a form of Jon Ingold's former extension Text Variations, which itself built on code going back to the days of I6.

The head phrase here has one of the most complicated definitions in the SR, but is actually documented fairly explicitly in the *Extensions* chapter of *Writing with Inform*, so we won't repeat all that here. Essentially it uses its own allocated cell of storage in an array to remember a state between uses, and compiles as a switch statement based on the current state.

```
Section SR5/1/4 - Saying - Say one of

To say one of -- beginning say_one_of (documented at phs_oneof):
    (- {-allocate-storage:say_one_of}I7_ST_say_one_of-->{-counter:say_one_of} =
    {-final-segment-marker}(I7_ST_say_one_of-->{-counter:say_one_of}, {-segment-count});
    switch((I7_ST_say_one_of-->{-advance-counter:say_one_of})%({-segment-count}+1)-1) {-open-brace}
        0: -).
To say or -- continuing say_one_of (documented at phs_or):
    (- @nop; {-segment-count}: -).
To say at random -- ending say_one_of with marker I7_SOO_RAN (documented at phs_random):
    (- {-close-brace} -).
To say purely at random -- ending say_one_of with marker I7_SOO_PAR (documented at phs_purelyrandom
  ... ):
    (- {-close-brace} -).
To say then at random -- ending say_one_of with marker I7_SOO_TRAN (documented at phs_thenrandom):
    (- {-close-brace} -).
To say then purely at random -- ending say_one_of with marker I7_SOO_TPAR (documented at phs_thenpu
  ... relyrandom):
    (- {-close-brace} -).
To say sticky random -- ending say_one_of with marker I7_SOO_STI (documented at phs_sticky):
    (- {-close-brace} -).
To say as decreasingly likely outcomes -- ending say_one_of with marker I7_SOO_TAP (documented at p
  ... hs_decreasing):
    (- {-close-brace} -).
To say in random order -- ending say_one_of with marker I7_SOO_SHU (documented at phs_order):
    (- {-close-brace} -).
To say cycling -- ending say_one_of with marker I7_SOO_CYC (documented at phs_cycling):
    (- {-close-brace} -).
To say stopping -- ending say_one_of with marker I7_SOO_STOP (documented at phs_stopping):
    (- {-close-brace} -).

To say first time -- beginning say_first_time (documented at phs_firsttime):
```

```
    (- {-allocate-storage:say_first_time}
    if ((I7_ST_say_first_time-->{-advance-counter:say_first_time})++ == 0) {-open-brace}
        -).
To say only -- ending say_first_time (documented at phs_firsttime):
    (- {-close-brace} -).
```

§**9.** For an explanation of the paragraph breaking algorithm, see the template file "Printing.i6t".

```
Section SR5/1/5 - Saying - Paragraph control

To say line break -- running on
    (documented at phs_linebreak):
    (- new_line; -).
To say no line break -- running on
    (documented at phs_nolinebreak): do nothing.
To say conditional paragraph break -- running on
    (documented at phs_condparabreak):
    (- DivideParagraphPoint(); -).
To say command clarification break -- running on
    (documented at phs_clarifbreak):
    (- CommandClarificationBreak(); -).
To say paragraph break -- running on
    (documented at phs_parabreak):
    (- DivideParagraphPoint(); new_line; -).
To say run paragraph on -- running on
    (documented at phs_runparaon):
    (- RunParagraphOn(); -).
To say run paragraph on with special look spacing -- running on
    (documented at phs_runparaonsls):
    (- SpecialLookSpacingBreak(); -).
To decide if a paragraph break is pending
    (documented at ph_breakpending):
    (- (say__p) -).
```

§**10.** Now some text substitutions which are the equivalent of escape characters. (In double-quoted I6 text, the notation for a literal quotation mark is a tilde ~.)

```
Section SR5/1/6 - Saying - Special characters

To say bracket -- running on
    (documented at phs_bracket):
    (- print "["; -).
To say close bracket -- running on
    (documented at phs_closebracket):
    (- print "]"; -).
To say apostrophe/' -- running on
    (documented at phs_apostrophe):
    (- print "'"; -).
To say quotation mark -- running on
    (documented at phs_quotemark):
    (- print "~"; -).
```

§**11.** Now some visual effects, which may or may not be rendered the way the user hopes: that's partly up to the virtual machine, unfortunately.

```
Section SR5/1/7 - Saying - Fonts and visual effects

To say bold type -- running on
    (documented at phs_bold):
    (- style bold; -).
To say italic type -- running on
    (documented at phs_italic):
    (- style underline; -).
To say roman type -- running on
    (documented at phs_roman):
    (- style roman; -).
To say fixed letter spacing -- running on
    (documented at phs_fixedspacing):
    (- font off; -).
To say variable letter spacing -- running on
    (documented at phs_varspacing):
    (- font on; -).
To display the boxed quotation (Q - text)
    (documented at ph_boxed):
    (- DisplayBoxedQuotation({-box-quotation-text:Q}); -).
```

§**12.** And now some oddball special texts which must sometimes be said.

```
Section SR5/1/8 - Saying - Some built-in texts

To say the/-- banner text
    (documented at phs_banner):
    (- Banner(); -).
To say the/-- list of extension credits
    (documented at phs_extcredits):
    (- ShowExtensionVersions(); -).
To say the/-- complete list of extension credits
    (documented at phs_compextcredits):
    (- ShowFullExtensionVersions(); -).
To say the/-- player's surroundings
    (documented at phs_surroundings):
    (- SL_Location(); -).
```

§**13. Using the list-writer.** The I7 list-writer resembles the old I6 library one, but has been reimplemented in a more general way: see the template file "ListWriter.i6t". The following is the main routine for listing:

```
Section SR5/1/9 - Saying - Saying lists of things

To list the contents of (O - an object),
    with newlines,
    indented,
    giving inventory information,
    as a sentence,
    including contents,
    including all contents,
    tersely,
    giving brief inventory information,
    using the definite article,
    listing marked items only,
    prefacing with is/are,
    not listing concealed items,
    suppressing all articles
    and/or with extra indentation
    (documented at ph_listcontents):
    (- WriteListFrom(child({O}), {phrase options}); -).

To say contents of (O - an object)
    (deprecated)
    (documented at phs_contents_dep):
    list the contents of O, as a sentence.

To say the contents of (O - an object)
    (deprecated)
    (documented at phs_contents_dep):
    list the contents of O, as a sentence, using the definite article.
```

§**14. Text substitutions using the list-writer.** These all look (and are) repetitive. We want to avoid passing a description value to some routine, because that's tricky if the description needs to refer to a value local to the current stack frame. (There are ways round that, but it minimises nuisance to avoid the need.) So we mark out the set of objects matching by giving them, and only them, the `workflag2` attribute.

```
To say a list of (OS - description of objects)
    (documented at phs_alistof):
    (- @push subst__v;
        objectloop (subst__v ofclass Object) if ({-bind-variable:OS})
        give subst__v workflag2; else give subst__v ~workflag2;
        WriteListOfMarkedObjects(ENGLISH_BIT);
        @pull subst__v; -).
To say A list of (OS - description of objects)
    (documented at phs_Alistof):
    (- @push subst__v;
        objectloop (subst__v ofclass Object) if ({-bind-variable:OS})
        give subst__v workflag2; else give subst__v ~workflag2;
        WriteListOfMarkedObjects(ENGLISH_BIT+CFIRSTART_BIT);
        @pull subst__v; -).
To say list of (OS - description of objects)
    (documented at phs_listof):
```

```
    (- @push subst__v;
        objectloop (subst__v ofclass Object) if ({-bind-variable:OS})
        give subst__v workflag2; else give subst__v ~workflag2;
        WriteListOfMarkedObjects(ENGLISH_BIT+NOARTICLE_BIT);
        @pull subst__v; -).
To say the list of (OS - description of objects)
    (documented at phs_thelistof):
    (- @push subst__v;
        objectloop (subst__v ofclass Object) if ({-bind-variable:OS})
        give subst__v workflag2; else give subst__v ~workflag2;
        WriteListOfMarkedObjects(ENGLISH_BIT+DEFART_BIT);
        @pull subst__v; -).
To say The list of (OS - description of objects)
    (documented at phs_Thelistof):
    (- @push subst__v;
        objectloop (subst__v ofclass Object) if ({-bind-variable:OS})
        give subst__v workflag2; else give subst__v ~workflag2;
        WriteListOfMarkedObjects(ENGLISH_BIT+DEFART_BIT+CFIRSTART_BIT);
        @pull subst__v; -).
To say is-are a list of (OS - description of objects)
    (documented at phs_isalistof):
    (- @push subst__v;
        objectloop (subst__v ofclass Object) if ({-bind-variable:OS})
        give subst__v workflag2; else give subst__v ~workflag2;
        WriteListOfMarkedObjects(ENGLISH_BIT+ISARE_BIT);
        @pull subst__v; -).
To say is-are list of (OS - description of objects)
    (documented at phs_islistof):
    (- @push subst__v;
        objectloop (subst__v ofclass Object) if ({-bind-variable:OS})
        give subst__v workflag2; else give subst__v ~workflag2;
        WriteListOfMarkedObjects(ENGLISH_BIT+ISARE_BIT+NOARTICLE_BIT);
        @pull subst__v; -).
To say is-are the list of (OS - description of objects)
    (documented at phs_isthelistof):
    (- @push subst__v;
        objectloop (subst__v ofclass Object) if ({-bind-variable:OS})
        give subst__v workflag2; else give subst__v ~workflag2;
        WriteListOfMarkedObjects(ENGLISH_BIT+DEFART_BIT+ISARE_BIT);
        @pull subst__v; -).
To say a list of (OS - description of objects) including contents
    (documented at phs_alistofconts):
    (- @push subst__v;
        objectloop (subst__v ofclass Object) if ({-bind-variable:OS})
        give subst__v workflag2; else give subst__v ~workflag2;
        WriteListOfMarkedObjects(ENGLISH_BIT+RECURSE_BIT+PARTINV_BIT+
            TERSE_BIT+CONCEAL_BIT);
        @pull subst__v; -).
```

§**15. Grouping in the list-writer.** See the specifications of `list_together` and `c_style` in the DM4, which are still broadly accurate.

```
Section SR5/1/10 - Saying - Group in and omit from lists

To group (OS - description of objects) together
    (documented at ph_group):
    (- @push subst__v;
        objectloop (subst__v provides list_together) if ({-bind-variable:OS})
        subst__v.list_together = {-list-together};
        @pull subst__v; -).
To group (OS - description of objects) together giving articles
    (documented at ph_groupart):
    (- @push subst__v;
        objectloop (subst__v provides list_together) if ({-bind-variable:OS})
        subst__v.list_together = {-articled-list-together};
        @pull subst__v; -).
To group (OS - description of objects) together as (T - text)
    (documented at ph_grouptext):
    (- @push subst__v;
        objectloop (subst__v provides list_together) if ({-bind-variable:OS})
        subst__v.list_together = {T};
        @pull subst__v; -).
To omit contents in listing
    (documented at ph_omit):
    (- c_style = c_style &~ (RECURSE_BIT+FULLINV_BIT+PARTINV_BIT); -).
```

§**16. Lists written but not by the list-writer.** These are lists in the sense of the "list of" kind of value constructor, and they might contain any values, not just objects. They're printed by code in "Lists.i6t".

```
Section SR5/1/11 - Saying - Lists of values

To say (L - a list of values) in brace notation
    (documented at phs_listbraced):
    (- LIST_OF_TY_Say({-pointer-to:L}, 1); -).
To say (L - a list of objects) with definite articles
    (documented at phs_listdef):
    (- LIST_OF_TY_Say({-pointer-to:L}, 2); -).
To say (L - a list of objects) with indefinite articles
    (documented at phs_listindef):
    (- LIST_OF_TY_Say({-pointer-to:L}, 3); -).
```

§**17. Filtering in the list-writer.** Something of a last resort, which is intentionally not documented. It's needed by the Standard Rules to tidy up an implementation and avoid I6, but is not an ideal trick and may be dropped in later builds. Recursion occurs when the list-writer descends to the contents of, or items supported by, something it lists. Here we can restrict to just those contents, or supportees, matching a description `D`.

```
Section SR5/1/12 - Saying - Filtering contents - Unindexed

To filter list recursion to (D - description of objects):
    (- list_filter_routine = {D}; -).
To unfilter list recursion:
    (- list_filter_routine = 0; -).
```

§**18. Values.** To begin with, the magic "now".

```
Section SR5/2/1 - Values - Making conditions true

To now (cn - now-condition)
    (documented at ph_now):
    (- {cn} -).
```

§**19.** Assignment is probably the most difficult thing the type-checker has to cope with, since "let" has to work when applied to both unknown names (it creates a new variable) and existing ones (kind of value permitting). There are also four different ways to create with "let", and two to use existing variables. Note that the "given by" forms are not strictly speaking assignments at all; the value placed in `t` is found by solving the equation `Q`. This does require special typechecking, but of a different kind to that requested by "(assignment operation)". All of which makes the "To let" section here only slightly shorter than John Galsworthy's Forsyte novel of the same name:

```
Section SR5/2/2 - Values - Giving values temporary names

To let (t - nonexisting variable) be (u - value)
    (assignment operation)
    (documented at ph_let):
    (- {-creationassignment}; -).
To let (t - nonexisting variable) be (u - name of kind of value)
    (assignment operation)
    (documented at ph_letdefault):
    (- {-creationdefault}; -).
To let (t - nonexisting variable) be (u - description of relations)
    (assignment operation)
    (documented at ph_letrelation):
    (- {-pointer-to:t} = {-allocate-storage-for:u}; {-adapt-relation}; -).
To let (t - nonexisting variable) be given by (Q - equation name)
    (documented at ph_letequation):
    (- {-solve-equation}; -).

To let (t - existing variable) be (u - value)
    (assignment operation)
    (documented at ph_let):
    (- {-assignment}; -).
To let (t - existing variable) be given by (Q - equation name)
    (documented at ph_letequation):
    (- {-solve-equation}; -).
```

§**20.** There are five sorts of storage in I7 at present: local variables (i.e., "let" and "called" variables, together with parameters in "To..." preambles), global variables, table entries, property values and list entries. Objects are not a form of storage in this sense. The reason we handle "change O to X" specially where O is an object is that it's legal to "change O to open" if O is a container or door, or to "change O to 20kg" if O has a weight, for instance. The meaning then is to do with the transference of a property, not an assignment. This requires careful disambiguation, since O may be, e.g., a global variable whose kind of value is "object".

```
Section SR5/2/3 - Values - Changing stored values

To change (S - storage) to (w - value)
    (deprecated)
    (assignment operation)
    (documented at ph_change_dep):
    (- {-assignment}; -).
To change (o - object) to (p - property)
    (deprecated)
    (assignment operation)
    (documented at ph_change_dep):
    (- SetEitherOrProperty({o}, {p}, false, {-adjective-definition:p}); -).
To change (o - object) to (w - enumerated value)
    (deprecated)
    (assignment operation)
    (documented at ph_change_dep):
    (- WriteGProperty(OBJECT_TY,{o},{-convert-adjectival-constants:w},{w}); -).
```

§**21.** It is not explicit in the following definitions that Inform should typecheck that the values held by these storage objects can be incremented or decremented (as a room, say, cannot, but a number or a height can): Inform nevertheless contains code which does this.

```
To increase (S - storage) by (w - value)
    (assignment operation)
    (documented at ph_increase):
    (- {-increase}; -).
To decrease (S - storage) by (w - value)
    (assignment operation)
    (documented at ph_decrease):
    (- {-decrease}; -).
To increment (S - storage)
    (documented at ph_increment):
    (- {-increment}; -).
To decrement (S - storage)
    (documented at ph_decrement):
    (- {-decrement}; -).
```

§**22.** There are eight arithmetic operations, internally numbered 0 to 7, and given verbal forms below. These are handled unusually in the type-checker because they need to be more polymorphic than most phrases: Inform uses dimension-checking to determine the kind of value resulting. (Thus a height times a number is another height, and so on.)

The totalling code (10) is not structly to do with arithmetic in the same way, but it's needed to flag the phrase for the Inform typechecker's special attention.

```
Section SR5/2/4 - Values - Arithmetic

To decide which arithmetic value is (X - arithmetic value) + (Y - arithmetic value)
    (arithmetic operation 0)
    (documented at ph_plus): (- ({X}+{Y}) -).
To decide which arithmetic value is (X - arithmetic value) plus (Y - arithmetic value)
    (arithmetic operation 0)
    (documented at ph_plus):
    (- ({X}+{Y}) -).
To decide which arithmetic value is (X - arithmetic value) - (Y - arithmetic value)
    (arithmetic operation 1)
    (documented at ph_minus):
    (- ({X}-{Y}) -).
To decide which arithmetic value is (X - arithmetic value) minus (Y - arithmetic value)
    (arithmetic operation 1)
    (documented at ph_minus):
    (- ({X}-{Y}) -).
To decide which arithmetic value is (X - arithmetic value) * (Y - arithmetic value)
    (arithmetic operation 2)
    (documented at ph_times):
    (- ({X}*{Y}{-rescale-times}) -).
To decide which arithmetic value is (X - arithmetic value) times (Y - arithmetic value)
    (arithmetic operation 2)
    (documented at ph_times):
    (- ({X}*{Y}{-rescale-times}) -).
To decide which arithmetic value is (X - arithmetic value) multiplied by (Y - arithmetic value)
    (arithmetic operation 2)
    (documented at ph_times):
    (- ({X}*{Y}{-rescale-times}) -).
To decide which arithmetic value is (X - arithmetic value) / (Y - arithmetic value)
    (arithmetic operation 3)
    (documented at ph_divide):
    (- (IntegerDivide({X}{-rescale-divide},{Y})) -).
To decide which arithmetic value is (X - arithmetic value) divided by (Y - arithmetic value)
    (arithmetic operation 3)
    (documented at ph_divide):
    (- (IntegerDivide({X}{-rescale-divide},{Y})) -).
To decide which arithmetic value is remainder after dividing (X - arithmetic value)
    by (Y - arithmetic value)
    (arithmetic operation 4)
    (documented at ph_remainder):
    (- (IntegerRemainder({X},{Y})) -).
To decide which arithmetic value is (X - arithmetic value) to the nearest (Y - arithmetic value)
    (arithmetic operation 5)
    (documented at ph_nearest):
    (- (RoundOffTime({X},{Y})) -).
To decide which arithmetic value is the square root of (X - arithmetic value)
```

```
    (arithmetic operation 6)
    (documented at ph_squareroot):
    (- (SquareRoot({X}{-rescale-root})) -).
To decide which arithmetic value is the cube root of (X - arithmetic value)
    (arithmetic operation 7)
    (documented at ph_cuberoot):
    (- (CubeRoot({X}{-rescale-cuberoot})) -).
To decide which arithmetic value is total (p - arithmetic value valued property)
    of (S - description of values)
    (arithmetic operation 11)
    (documented at ph_total):
    (- {-total-of:S} -).
```

§**23.** Some of the things we can do with enumerations, others being listed under randomness below.

```
Section SR5/2/5 - Values - Enumerations

To decide which number is number of (S - description of values)
    (documented at ph_numberof):
    (- {-number-of:S} -).
To decide which K is (name of kind of enumerated value K) after (X - K)
    (documented at ph_enumafter):
    (- A_{-printing-routine:K}({X}) -).
To decide which K is (name of kind of enumerated value K) before (X - K)
    (documented at ph_enumbefore):
    (- B_{-printing-routine:K}({X}) -).
To decide which K is the first value of (name of kind of enumerated value K)
    (documented at ph_enumfirst):
    decide on the default value of K.
To decide which K is the last value of (name of kind of enumerated value K)
    (documented at ph_enumlast):
    decide on K before the default value of K.
```

§**24.** The following exists only to convert a condition to a value, and is needed because I7 does not silently cast from one to the other in the way that C would.

```
Section SR5/2/6 - Values - Truth states

To decide what truth state is whether or not (C - condition)
    (documented at ph_whether):
    (- ({C}) -).
```

§**25.** Random numbers and random items chosen from sets of objects matching a given description ("a random closed door").

```
Section SR5/2/7 - Values - Randomness

To decide which K is a/-- random (S - description of values of kind K)
    (documented at ph_randomdesc):
    (- {-random-of:S} -).
To decide which K is a random (name of kind of arithmetic value K) between (first value - K) and (s
  ... econd value - K)
    (documented at ph_randombetween):
    (- R_{-printing-routine:K}({first value}, {second value}) -).
To decide which K is a random (name of kind of arithmetic value K) from (first value - K) to (secon
  ... d value - K)
    (documented at ph_randombetween):
    (- R_{-printing-routine:K}({first value}, {second value}) -).
To decide which K is a random (name of kind of enumerated value K) between (first value - K) and (s
  ... econd value - K)
    (documented at ph_randombetween):
    (- R_{-printing-routine:K}({first value}, {second value}) -).
To decide which K is a random (name of kind of enumerated value K) from (first value - K) to (secon
  ... d value - K)
    (documented at ph_randombetween):
    (- R_{-printing-routine:K}({first value}, {second value}) -).
To decide whether a random chance of (N - number) in (M - number) succeeds
    (documented at ph_randomchance):
    (- (GenerateRandomNumber(1, {M}) <= {N}) -).
To seed the random-number generator with (N - number)
    (documented at ph_seed):
    (- VM_Seed_RNG({N}); -).
```

§**26. Tables.** And off we go into the world of tables, the code for which is all in "Tables.i6t". Note that changing a table entry is not here: it's above, with the phrases for changing other storage objects.

```
Section SR5/2/8 - Values - Tables

To choose a/the/-- row (N - number) in/from (T - table name)
    (documented at ph_chooserow):
    (- {-require-ctvs}ct_0 = {T}; ct_1 = {N}; -).
To choose a/the/-- row with (TC - K valued table column) of (w - value of kind K)
    in/from (T - table name)
    (documented at ph_chooserowwith):
    (- {-require-ctvs}ct_0 = {T}; ct_1 = TableRowCorr(ct_0, {TC}, {w}); -).
To choose a/the/-- blank row in/from (T - table name)
    (documented at ph_chooseblankrow):
    (- {-require-ctvs}ct_0 = {T}; ct_1 = TableBlankRow(ct_0); -).
To choose a/the/-- random row in/from (T - table name)
    (documented at ph_chooserandomrow):
    (- {-require-ctvs}ct_0 = {T}; ct_1 = TableRandomRow(ct_0); -).
To decide which number is number of rows in/from (T - table name)
    (documented at ph_numrows):
    (- TableRows({T}) -).
To decide which number is number of blank rows in/from (T - table name)
    (documented at ph_numblank):
```

```
    (- TableBlankRows({T}) -).
To decide which number is number of filled rows in/from (T - table name)
    (documented at ph_numfilled):
    (- TableFilledRows({T}) -).
To decide if there is (TR - table-reference)
    (documented at ph_thereis):
    (- (Exists{-do-not-dereference:TR}) -).
To decide if there is no (TR - table-reference)
    (documented at ph_thereisno):
    (- (Exists{-do-not-dereference:TR} == false) -).
To blank out (tr - table-reference)
    (documented at ph_blankout):
    (- {tr}{-backspace},4); -).
To blank out the whole row
    (documented at ph_blankoutrow):
    (- {-require-ctvs}TableBlankOutRow(ct_0, ct_1); -).
To blank out the whole (TC - table column) in/from (T - table name)
    (documented at ph_blankoutcol):
    (- TableBlankOutColumn({T}, {TC}); -).
To blank out the whole of (T - table name)
    (documented at ph_blankouttable):
    (- TableBlankOutAll({T}); -).
To delete (tr - table-reference)
    (deprecated)
    (documented at ph_deleteentry_dep):
    (- {tr}{-backspace},4); -).
```

§27. Sorting.

```
Section SR5/2/9 - Values - Sorting tables

To sort (T - table name) in random order
    (documented at ph_sortrandom):
    (- TableShuffle({T}); -).
To sort (T - table name) in (TC - table column) order
    (documented at ph_sortcolumn):
    (- TableSort({T}, {TC}, 1); -).
To sort (T - table name) in reverse (TC - table column) order
    (documented at ph_sortcolumnreverse):
    (- TableSort({T}, {TC}, -1); -).
```

§**28. Indexed text.** As repetitive as the following is, it's much simpler and less prone to possible namespace trouble if we don't define kinds of value for the different structural levels of text (character, word, punctuated word, etc.).

```
Section SR5/2/10 - Values - Indexed text

To decide what number is the number of characters in (txb - indexed text)
    (documented at ph_numchars):
    (- IT_BlobAccess({-pointer-to:txb}, CHR_BLOB) -).
To decide what number is the number of words in (txb - indexed text)
    (documented at ph_numwords):
    (- IT_BlobAccess({-pointer-to:txb}, WORD_BLOB) -).
To decide what number is the number of punctuated words in (txb - indexed text)
    (documented at ph_numpwords):
    (- IT_BlobAccess({-pointer-to:txb}, PWORD_BLOB) -).
To decide what number is the number of unpunctuated words in (txb - indexed text)
    (documented at ph_numupwords):
    (- IT_BlobAccess({-pointer-to:txb}, UWORD_BLOB) -).
To decide what number is the number of lines in (txb - indexed text)
    (documented at ph_numlines):
    (- IT_BlobAccess({-pointer-to:txb}, LINE_BLOB) -).
To decide what number is the number of paragraphs in (txb - indexed text)
    (documented at ph_numparas):
    (- IT_BlobAccess({-pointer-to:txb}, PARA_BLOB) -).

To decide what indexed text is character number (N - a number) in (txb - indexed text)
    (documented at ph_charnum):
    (- IT_GetBlob({-pointer-to-new:indexed text}, {-pointer-to:txb}, {N}, CHR_BLOB) -).
To decide what indexed text is word number (N - a number) in (txb - indexed text)
    (documented at ph_wordnum):
    (- IT_GetBlob({-pointer-to-new:indexed text}, {-pointer-to:txb}, {N}, WORD_BLOB) -).
To decide what indexed text is punctuated word number (N - a number) in (txb - indexed text)
    (documented at ph_pwordnum):
    (- IT_GetBlob({-pointer-to-new:indexed text}, {-pointer-to:txb}, {N}, PWORD_BLOB) -).
To decide what indexed text is unpunctuated word number (N - a number) in (txb - indexed text)
    (documented at ph_upwordnum):
    (- IT_GetBlob({-pointer-to-new:indexed text}, {-pointer-to:txb}, {N}, UWORD_BLOB) -).
To decide what indexed text is line number (N - a number) in (txb - indexed text)
    (documented at ph_linenum):
    (- IT_GetBlob({-pointer-to-new:indexed text}, {-pointer-to:txb}, {N}, LINE_BLOB) -).
To decide what indexed text is paragraph number (N - a number) in (txb - indexed text)
    (documented at ph_paranum):
    (- IT_GetBlob({-pointer-to-new:indexed text}, {-pointer-to:txb}, {N}, PARA_BLOB) -).
```

§**29. Matching text.** A common matching engine is used for matching plain text...

```
Section SR5/2/11 - Values - Matching text

To decide if (txb - indexed text) exactly matches the text (ftxb - indexed text),
    case insensitively
    (documented at ph_exactlymatches):
    (- IT_Replace_RE(CHR_BLOB,{-pointer-to:txb},{-pointer-to:ftxb},0,{phrase options},1) -).
To decide if (txb - indexed text) matches the text (ftxb - indexed text),
    case insensitively
    (documented at ph_matches):
    (- IT_Replace_RE(CHR_BLOB,{-pointer-to:txb},{-pointer-to:ftxb},0,{phrase options}) -).
To decide what number is number of times (txb - indexed text) matches the text
    (ftxb - indexed text), case insensitively
    (documented at ph_nummatches):
    (- IT_Replace_RE(CHR_BLOB,{-pointer-to:txb},{-pointer-to:ftxb},1,{phrase options}) -).
```

§**30.** ...and for regular expressions, though here we also have access to the exact text which matched (not interesting in the plain text case since it's the same as the search text, up to case at least), and the values of matched subexpressions (which the plain text case doesn't have).

```
To decide if (txb - indexed text) exactly matches the regular expression (ftxb - indexed text),
    case insensitively
    (documented at ph_exactlymatchesre):
    (- IT_Replace_RE(REGEXP_BLOB,{-pointer-to:txb},{-pointer-to:ftxb},0,{phrase options},1) -).
To decide if (txb - indexed text) matches the regular expression (ftxb - indexed text),
    case insensitively
    (documented at ph_matchesre):
    (- IT_Replace_RE(REGEXP_BLOB,{-pointer-to:txb},{-pointer-to:ftxb},0,{phrase options}) -).
To decide what indexed text is text matching regular expression
    (documented at ph_matchtext):
    (- IT_RE_GetMatchVar({-pointer-to-new:indexed text}, 0) -).
To decide what indexed text is text matching subexpression (N - a number)
    (documented at ph_subexpressiontext):
    (- IT_RE_GetMatchVar({-pointer-to-new:indexed text}, {N}) -).
To decide what number is number of times (txb - indexed text) matches the regular expression
    (ftxb - indexed text),case insensitively
    (documented at ph_nummatchesre):
    (- IT_Replace_RE(REGEXP_BLOB,{-pointer-to:txb},{-pointer-to:ftxb},1,{phrase options}) -).
```

§**31. Replacing text.** The same engine, in "RegExp.i6t", handles replacement.

```
Section SR5/2/12 - Values - Replacing text

To replace the text (ftxb - indexed text) in (txb - indexed text) with (rtxb - indexed text),
    case insensitively
    (documented at ph_replace):
    (- IT_Replace_RE(CHR_BLOB, {-pointer-to:txb}, {-pointer-to:ftxb},
        {-pointer-to:rtxb}, {phrase options}); -).
To replace the regular expression (ftxb - indexed text) in (txb - indexed text) with
    (rtxb - indexed text), case insensitively
    (documented at ph_replacere):
    (- IT_Replace_RE(REGEXP_BLOB, {-pointer-to:txb}, {-pointer-to:ftxb},
        {-pointer-to:rtxb}, {phrase options}); -).
To replace the word (ftxb - indexed text) in (txb - indexed text) with
    (rtxb - indexed text)
    (documented at ph_replacewordin):
    (- IT_ReplaceText(WORD_BLOB, {-pointer-to:txb}, {-pointer-to:ftxb}, {-pointer-to:rtxb}); -).
To replace the punctuated word (ftxb - indexed text) in (txb - indexed text)
    with (rtxb - indexed text)
    (documented at ph_replacepwordin):
    (- IT_ReplaceText(PWORD_BLOB, {-pointer-to:txb}, {-pointer-to:ftxb}, {-pointer-to:rtxb}); -).

To replace character number (N - a number) in (txb - indexed text)
    with (rtxb - indexed text)
    (documented at ph_replacechar):
    (- IT_ReplaceBlob(CHR_BLOB, {-pointer-to:txb}, {N}, {-pointer-to:rtxb}); -).
To replace word number (N - a number) in (txb - indexed text)
    with (rtxb - indexed text)
    (documented at ph_replaceword):
    (- IT_ReplaceBlob(WORD_BLOB, {-pointer-to:txb}, {N}, {-pointer-to:rtxb}); -).
To replace punctuated word number (N - a number) in (txb - indexed text)
    with (rtxb - indexed text)
    (documented at ph_replacepword):
    (- IT_ReplaceBlob(PWORD_BLOB, {-pointer-to:txb}, {N}, {-pointer-to:rtxb}); -).
To replace unpunctuated word number (N - a number) in (txb - indexed text)
    with (rtxb - indexed text)
    (documented at ph_replaceupword):
    (- IT_ReplaceBlob(UWORD_BLOB, {-pointer-to:txb}, {N}, {-pointer-to:rtxb}); -).
To replace line number (N - a number) in (txb - indexed text) with (rtxb - indexed text)
    (documented at ph_replaceline):
    (- IT_ReplaceBlob(LINE_BLOB, {-pointer-to:txb}, {N}, {-pointer-to:rtxb}); -).
To replace paragraph number (N - a number) in (txb - indexed text) with (rtxb - indexed text)
    (documented at ph_replacepara):
    (- IT_ReplaceBlob(PARA_BLOB, {-pointer-to:txb}, {N}, {-pointer-to:rtxb}); -).
```

§32. Casing of text.

```
Section SR5/2/13 - Values - Casing of text

To decide what indexed text is (txb - indexed text) in lower case
    (documented at ph_lowercase):
    (- IT_CharactersToCase({-pointer-to-new:indexed text}, {-pointer-to:txb}, 0) -).
To decide what indexed text is (txb - indexed text) in upper case
    (documented at ph_uppercase):
    (- IT_CharactersToCase({-pointer-to-new:indexed text}, {-pointer-to:txb}, 1) -).
To decide what indexed text is (txb - indexed text) in title case
    (documented at ph_titlecase):
    (- IT_CharactersToCase({-pointer-to-new:indexed text}, {-pointer-to:txb}, 2) -).
To decide what indexed text is (txb - indexed text) in sentence case
    (documented at ph_sentencecase):
    (- IT_CharactersToCase({-pointer-to-new:indexed text}, {-pointer-to:txb}, 3) -).
To decide if (txb - indexed text) is in lower case
    (documented at ph_inlower):
    (- IT_CharactersOfCase({-pointer-to:txb}, 0) -).
To decide if (txb - indexed text) is in upper case
    (documented at ph_inupper):
    (- IT_CharactersOfCase({-pointer-to:txb}, 1) -).
```

§33. Lists.

The following are all for adding and removing values to dynamic lists:

```
Section SR5/2/14 - Values - Lists

To add (new entry - K) to (L - list of values of kind K), if absent
    (documented at ph_addtolist):
    (- LIST_OF_TY_InsertItem({-pointer-to:L}, {new entry}, 0, 0, {phrase options}); -).

To add (new entry - K) at entry (E - number) in (L - list of values of kind K), if absent
    (documented at ph_addatentry):
    (- LIST_OF_TY_InsertItem({-pointer-to:L}, {new entry}, 1, {E}, {phrase options}); -).

To add (LX - list of Ks) to (L - list of values of kind K), if absent
    (documented at ph_addlisttolist):
    (- LIST_OF_TY_AppendList({-pointer-to:L}, {-pointer-to:LX}, 0, 0, {phrase options}); -).

To add (LX - list of Ks) at entry (E - number) in (L - list of values of kind K)
    (documented at ph_addlistatentry):
    (- LIST_OF_TY_AppendList({-pointer-to:L}, {-pointer-to:LX}, 1, {E}, 0); -).

To remove (existing entry - K) from (L - list of values of kind K), if present
    (documented at ph_remfromlist):
    (- LIST_OF_TY_RemoveValue({-pointer-to:L}, {existing entry}, {phrase options}); -).

To remove (N - list of Ks) from (L - list of values of kind K), if present
    (documented at ph_remlistfromlist):
    (- LIST_OF_TY_Remove_List({-pointer-to:L}, {-pointer-to:N}, {phrase options}); -).

To remove entry (N - number) from (L - list of values), if present
    (documented at ph_rementry):
    (- LIST_OF_TY_RemoveItemRange({-pointer-to:L}, {N}, {N}, {phrase options}); -).

To remove entries (N - number) to (N2 - number) from (L - list of values), if present
    (documented at ph_rementries):
    (- LIST_OF_TY_RemoveItemRange({-pointer-to:L}, {N}, {N2}, {phrase options}); -).
```

§**34.** Searching a list is implemented in a somewhat crude way at present, and the following syntax may later be replaced with a suitable verb "to be listed in", so that there's no need to imitate.

```
To decide if (N - K) is listed in (L - list of values of kind K)
    (documented at ph_islistedin):
    (- (LIST_OF_TY_FindItem({-pointer-to:L}, {N})) -).
To decide if (N - K) is not listed in (L - list of values of kind K)
    (documented at ph_isnotlistedin):
    (- (LIST_OF_TY_FindItem({-pointer-to:L}, {N}) == false) -).
```

§**35.** The following are casts to and from other kinds or objects. Lists are not the only I7 way to hold list-like data, because sometimes the memory requirements for dynamic lists are beyond what the virtual machine can sustain.

(a) A description is a representation of a set of objects by means of a predicate (e.g., "open unlocked doors"), and it converts into a list of current members (in creation order), but there's no reverse process.

(b) The multiple object list is a data structure used in the parser when processing commands like TAKE ALL.

```
To decide what list of Ks is the list of (D - description of values of kind K)
    (documented at ph_listofdesc):
    (- LIST_OF_TY_Desc({-pointer-to-new:list of K}, {D}, {-strong-kind:K}) -).
To decide what list of objects is the multiple object list
    (documented at ph_multipleobjectlist):
    (- LIST_OF_TY_Mol({-pointer-to-new:list of objects}) -).
To alter the multiple object list to (L - list of objects)
    (documented at ph_altermultipleobjectlist):
    (- LIST_OF_TY_Set_Mol({-pointer-to:L}); -).
```

§**36.** Determining and setting the length:

```
Section SR5/2/15 - Values - Length of lists
To decide what number is the number of entries in/of (L - a list of values)
    (documented at ph_numberentries):
    (- LIST_OF_TY_GetLength({-pointer-to:L}) -).
To truncate (L - a list of values) to (N - a number) entries/entry
    (documented at ph_truncate):
    (- LIST_OF_TY_SetLength({-pointer-to:L}, {N}, -1, 1); -).
To truncate (L - a list of values) to the first (N - a number) entries/entry
    (documented at ph_truncatefirst):
    (- LIST_OF_TY_SetLength({-pointer-to:L}, {N}, -1, 1); -).
To truncate (L - a list of values) to the last (N - a number) entries/entry
    (documented at ph_truncatelast):
    (- LIST_OF_TY_SetLength({-pointer-to:L}, {N}, -1, -1); -).
To extend (L - a list of values) to (N - a number) entries/entry
    (documented at ph_extend):
    (- LIST_OF_TY_SetLength({-pointer-to:L}, {N}, 1); -).
To change (L - a list of values) to have (N - a number) entries/entry
    (documented at ph_changelength):
    (- LIST_OF_TY_SetLength({-pointer-to:L}, {N}, 0); -).
```

§**37.** Easy but useful list operations:

```
Section SR5/2/16 - Values - Reversing and rotating lists

To reverse (L - a list of values)
    (documented at ph_reverselist):
    (- LIST_OF_TY_Reverse({-pointer-to:L}); -).
To rotate (L - a list of values)
    (documented at ph_rotatelist):
    (- LIST_OF_TY_Rotate({-pointer-to:L}, 0); -).
To rotate (L - a list of values) backwards
    (documented at ph_rotatelistback):
    (- LIST_OF_TY_Rotate({-pointer-to:L}, 1); -).
```

§**38.** Sorting ultimately uses a common sorting mechanism, in "Sort.i6t", which handles both lists and tables.

```
Section SR5/2/17 - Values - Sorting lists

To sort (L - a list of values)
    (documented at ph_sortlist):
    (- LIST_OF_TY_Sort({-pointer-to:L}, 1); -).
To sort (L - a list of values) in reverse order
    (documented at ph_sortlistreverse):
    (- LIST_OF_TY_Sort({-pointer-to:L}, -1); -).
To sort (L - a list of values) in random order
    (documented at ph_sortlistrandom):
    (- LIST_OF_TY_Sort({-pointer-to:L}, 2); -).
To sort (L - a list of objects) in (P - property) order
    (documented at ph_sortlistproperty):
    (- LIST_OF_TY_Sort({-pointer-to:L}, 1, {P}, {-block-value:P}); -).
To sort (L - a list of objects) in reverse (P - property) order
    (documented at ph_sortlistpropertyreverse):
    (- LIST_OF_TY_Sort({-pointer-to:L}, -1, {P}, {-block-value:P}); -).
```

§**39.** To call use options a "data structure" is a stretch. Moreover, the following phrase is now deprecated: write "if O is active" in preference to "if using O".

```
Section SR5/2/18 - Values - Use options

To decide whether using the/-- (UO - use option)
    (deprecated)
    (documented at ph_testuseoption_dep):
    (- (TestUseOption({UO})) -).
```

§**40.** Relations are the final data structure given here. In some ways they are the most fundamental of all, but they're not either set or tested by procedural phrases – they lie in the linguistic structure of conditions. So all we have here are the route-finding phrases:

```
Section SR5/2/19 - Values - Relations

To show relation (R - relation)
    (documented at ph_showrelation):
    (- {-show-me}; RelationTest({-pointer-to:R}, RELS_SHOW); -).

To decide which object is next step via (R - relation of values to values)
    from (O1 - object) to (O2 - object)
    (documented at ph_nextstep):
    (- RelationRouteTo({-pointer-to:R},{O1},{O2},false) -).
To decide which number is number of steps via (R - relation of values to values)
    from (O1 - object) to (O2 - object)
    (documented at ph_numbersteps):
    (- RelationRouteTo({-pointer-to:R},{O1},{O2},true) -).

To decide which list of Ks is list of (name of kind of value K)
    that/which/whom (R - relation of Ks to values of kind L) relates
    (documented at ph_leftdomain):
    (- RelationTest({-pointer-to:R}, RELS_LIST, {-pointer-to-new:list of K}, RLIST_ALL_X) -).

To decide which list of Ls is list of (name of kind of value L)
    to which/whom (R - relation of values of kind K to Ls) relates
    (documented at ph_rightdomain):
    (- RelationTest({-pointer-to:R}, RELS_LIST, {-pointer-to-new:list of L}, RLIST_ALL_Y) -). [1]

To decide which list of Ls is list of (name of kind of value L)
    that/which/whom (R - relation of values of kind K to Ls) relates to
    (documented at ph_rightdomain):
    (- RelationTest({-pointer-to:R}, RELS_LIST, {-pointer-to-new:list of L}, RLIST_ALL_Y) -). [2]

To decide which list of Ks is list of (name of kind of value K) that/which/who
    relate to (Y - L) by (R - relation of Ks to values of kind L)
    (documented at ph_leftlookuplist):
    (- RelationTest({-pointer-to:R}, RELS_LOOKUP_ALL_X, {Y}, {-pointer-to-new:list of K}) -).

To decide which list of Ls is list of (name of kind of value L) to which/whom (X - K)
    relates by (R - relation of values of kind K to Ls)
    (documented at ph_rightlookuplist):
    (- RelationTest({-pointer-to:R}, RELS_LOOKUP_ALL_Y, {X}, {-pointer-to-new:list of L}) -). [1]

To decide which list of Ls is list of (name of kind of value L)
    that/which/whom (X - K) relates to by (R - relation of values of kind K to Ls)
    (documented at ph_rightlookuplist):
    (- RelationTest({-pointer-to:R}, RELS_LOOKUP_ALL_Y, {X}, {-pointer-to-new:list of L}) -). [2]

To decide whether (name of kind of value K) relates to (Y - L) by
    (R - relation of Ks to values of kind L)
    (documented at ph_ifright):
    (- RelationTest({-pointer-to:R}, RELS_LOOKUP_ANY, {Y}, RLANY_CAN_GET_X) -).

To decide whether (X - K) relates to (name of kind of value L) by
    (R - relation of values of kind K to Ls)
    (documented at ph_ifleft):
    (- RelationTest({-pointer-to:R}, RELS_LOOKUP_ANY, {X}, RLANY_CAN_GET_Y) -).

To decide which K is (name of kind of value K) that/which/who relates to
    (Y - L) by (R - relation of Ks to values of kind L)
    (documented at ph_leftlookup):
```

```
    (- RelationTest({-pointer-to:R}, RELS_LOOKUP_ANY, {Y}, RLANY_GET_X) -).
To decide which L is (name of kind of value L) to which/whom (X - K)
    relates by (R - relation of values of kind K to Ls)
    (documented at ph_rightlookup):
    (- RelationTest({-pointer-to:R}, RELS_LOOKUP_ANY, {X}, RLANY_GET_Y) -). [1]
To decide which L is (name of kind of value L) that/which/whom (X - K)
    relates to by (R - relation of values of kind K to Ls)
    (documented at ph_rightlookup):
    (- RelationTest({-pointer-to:R}, RELS_LOOKUP_ANY, {X}, RLANY_GET_Y) -). [2]
```

§**41.** The standard map and reduce operations found in most functional programming languages:

```
Section SR5/2/20 - Values - Functional programming
To decide whether (val - K) matches (desc - description of values of kind K)
    (documented at ph_valuematch):
    (- {-description-application} -).
To decide what K is (function - phrase nothing -> value of kind K) applied
    (documented at ph_applied0):
    (- {-function-application} -).
To decide what L is (function - phrase value of kind K -> value of kind L)
    applied to (input - K)
    (documented at ph_applied1):
    (- {-function-application} -).
To decide what M is (function - phrase (value of kind K, value of kind L) -> value of kind M)
    applied to (input - K) and (second input - L)
    (documented at ph_applied2):
    (- {-function-application} -).
To decide what N is (function - phrase (value of kind K, value of kind L, value of kind M) -> value
  ...    of kind N)
    applied to (input - K) and (second input - L) and (third input - M)
    (documented at ph_applied3):
    (- {-function-application} -).
To apply (function - phrase nothing -> nothing)
    (documented at ph_apply0):
    (- {-function-application}; -).
To apply (function - phrase value of kind K -> nothing)
    to (input - K)
    (documented at ph_apply1):
    (- {-function-application}; -).
To apply (function - phrase (value of kind K, value of kind L) -> nothing)
    to (input - K) and (second input - L)
    (documented at ph_apply2):
    (- {-function-application}; -).
To apply (function - phrase (value of kind K, value of kind L, value of kind M) -> nothing)
    to (input - K) and (second input - L) and (third input - M)
    (documented at ph_apply3):
    (- {-function-application}; -).
To decide what list of L is (function - phrase K -> value of kind L) applied to (original list - li
  ...  st of values of kind K)
    (documented at ph_appliedlist):
```

```
    let the result be a list of Ls;
    repeat with item running through the original list:
        let the mapped item be the function applied to the item;
        add the mapped item to the result;
    decide on the result.

To decide what K is the (function - phrase (K, K) -> K) reduction of (original list - list of value
  ...  s of kind K)
    (documented at ph_reduction):
    let the total be a K;
    let the count be 0;
    repeat with item running through the original list:
        increase the count by 1;
        if the count is 1, now the total is the item;
        otherwise now the total is the function applied to the total and the item;
    decide on the total.

To decide what list of K is the filter to (criterion - description of Ks) of
    (full list - list of values of kind K)
    (documented at ph_filter):
    let the filtered list be a list of K;
    repeat with item running through the full list:
        if the item matches the criterion:
            add the item to the filtered list;
    decide on the filtered list.

To showme (V - value)
    (documented at ph_showme):
    (- {-show-me} -).

To decide what K is the default value of (V - name of kind of value of kind K)
    (documented at ph_defaultvalue):
    (- {-default-value-for:V} -).
```

§**42. Using external resources.** The following all refer to "FileIO.i6t" and work only on Glulx.

```
Section SR5/2/21 - Values - Files (for Glulx external files language element only)

To read (filename - external file) into (T - table name)
    (documented at ph_readtable):
    (- FileIO_GetTable({filename}, {T}); -).
To write (filename - external file) from (T - table name)
    (documented at ph_writetable):
    (- FileIO_PutTable({filename}, {T}); -).
To decide if (filename - external file) exists
    (documented at ph_fileexists):
    (- (FileIO_Exists({filename}, false)) -).
To decide if ready to read (filename - external file)
    (documented at ph_fileready):
    (- (FileIO_Ready({filename}, false)) -).
To mark (filename - external file) as ready to read
    (documented at ph_markfileready):
    (- FileIO_MarkReady({filename}, true); -).
To mark (filename - external file) as not ready to read
    (documented at ph_markfilenotready):
    (- FileIO_MarkReady({filename}, false); -).
To write (T - text) to (FN - external file)
    (documented at ph_writetext):
    (- FileIO_PutContents({FN}, {-allow-stack-frame-access:T}, false); -).
To append (T - text) to (FN - external file)
    (documented at ph_appendtext):
    (- FileIO_PutContents({FN}, {-allow-stack-frame-access:T}, true); -).
To say text of (FN - external file)
    (documented at ph_saytext):
    (- FileIO_PrintContents({FN}); say__p = 1; -).
```

§**43.** Figures and sound effects. Ditto, but for "Figures.i6t".

```
Section SR5/2/22 - Values - Figures (for figures language element only)

To display (F - figure name), one time only
    (documented at ph_displayfigure):
    (- DisplayFigure(ResourceIDsOfFigures-->{F}, {phrase options}); -).
To decide which number is the Glulx resource ID of (F - figure name)
    (documented at ph_figureid):
    (- ResourceIDsOfFigures-->{F} -).

Section SR5/2/23 - Values - Sound effects (for sounds language element only)

To play (SFX - sound name), one time only
    (documented at ph_playsf):
    (- PlaySound(ResourceIDsOfSounds-->{SFX}, {phrase options}); -).
To decide which number is the Glulx resource ID of (SFX - sound name)
    (documented at ph_soundid):
    (- ResourceIDsOfSounds-->{SFX} -).
```

§**44. Control phrases.** While "unless" is supposed to be exactly like "if" but with the reversed sense of the condition, that isn't quite true. For example, there is no "unless ... then ...": logical it might be, English it is not.

```
Section SR5/3/1 - Control phrases - If and unless

To if (c - condition) , (ph - phrase)
    (documented at ph_if):
    (- if {c} {ph} -).
To if (c - condition) begin -- end
    (documented at ph_if):
    (- if {c}  -).
To if (c - condition) then (ph - phrase)
    (deprecated)
    (documented at ph_if_dep):
    (- if {c} {ph} -).
To unless (c - condition) , (ph - phrase)
    (documented at ph_unless):
    (- if (~~{c}) {ph} -).
To unless (c - condition) begin -- end
    (documented at ph_unless):
    (- if (~~{c})  -).
```

§**45.** It looks as if the definitions below could be abbreviated a little by making more use of the slash ("To else/otherwise if...", say), but in fact the slash isn't compatible with these built-in control structure definitions – or at any rate using it causes a handful of problem messages from the type-checker to be worded badly in some cases.

```
To otherwise if (c - condition)
    (documented at ph_otherwise):
    (- } else if {c} { -).
To otherwise unless (c - condition)
    (documented at ph_otherwise):
    (- } else if (~~{c}) { -).
To otherwise (ph - phrase)
    (documented at ph_otherwise):
    (- else {ph} -).
To else if (c - condition)
    (documented at ph_otherwise):
    (- } else if {c} { -).
To else unless (c - condition)
    (documented at ph_otherwise):
    (- } else if (~~{c}) { -).
To else (ph - phrase)
    (documented at ph_otherwise):
    (- else {ph} -).
```

§**46.** The switch form of "if" is subtly different, and here again "unless" is not allowed in its place.

The begin and end markers here are in a sense bogus, in that the end user isn't supposed to type them: they are inserted automatically in the sentence subtree maker, which converts indentation into block structure.

```
To if (V - word value) is begin -- end
    (documented at ph_switch):
    (- switch({V})  -).
```

§**47.** After all that, the while loop is simplicity itself. Perhaps the presence of "unless" for "if" argues for a similarly negated form, "until" for "while", but users haven't yet petitioned for this.

```
Section SR5/3/2 - Control phrases - While

To while (c - condition) repeatedly (ph - phrase)
    (deprecated)
    (documented at ph_while_dep):
    (- while {c} {ph} -).
To while (c - condition) , (ph - phrase)
    (deprecated)
    (documented at ph_while_dep):
    (- while {c} {ph} -).
To while (c - condition) begin -- end
    (documented at ph_while):
    (- while {c}  -).
```

§**48.** The repeat loop looks like a single construction, but isn't, because the range can be given in four fundamentally different ways (and the loop variable then has a different kind of value accordingly). First, the equivalents of BASIC's `for` loop and of Inform 6's `objectloop`, respectively:

```
Section SR5/3/3 - Control phrases - Repeat

To repeat with (loopvar - nonexisting K variable)
    running from (v - arithmetic value of kind K) to (w - K) begin -- end
    (documented at ph_repeat):
        (- for ({loopvar}={v}: {loopvar}<={w}: {loopvar}++)  -).
To repeat with (loopvar - nonexisting K variable)
    running from (v - enumerated value of kind K) to (w - K) begin -- end
    (documented at ph_repeat):
        (- for ({loopvar}={v}: {loopvar}<={w}: {loopvar}++)  -).
To repeat with (loopvar - nonexisting K variable)
    running through (OS - description of values of kind K) begin -- end
    (documented at ph_runthrough):
        (- {-loop-over:OS}  -).
To repeat with (loopvar - nonexisting object variable)
    running through (L - list of values) begin -- end
    (documented at ph_repeatlist):
    (- {-loop-over-list:L}  -).
```

§**49.** The following are all repeats where the range is the set of rows of a table, taken in some order, and the repeat variable – though it does exist – is never specified since the relevant row is instead the one selected during each iteration of the loop.

```
To repeat through (T - table name) begin -- end
    (documented at ph_repeattable):
    (- {-push-ctvs}
        for ({-ct-v0}={T},{-ct-v1}=1,ct_0={-ct-v0},ct_1={-ct-v1}:
            {-ct-v1}<=TableRows({-ct-v0}):{-ct-v1}++,ct_0={-ct-v0},ct_1={-ct-v1})
            if (TableRowIsBlank(ct_0,ct_1)==false)  -).
To repeat through (T - table name) in reverse order begin -- end
    (documented at ph_repeattablereverse):
    (- {-push-ctvs}
        for ({-ct-v0}={T},{-ct-v1}=TableRows({-ct-v0}),ct_0={-ct-v0},ct_1={-ct-v1}:
            {-ct-v1}>=1:{-ct-v1}--,ct_0={-ct-v0},ct_1={-ct-v1})
            if (TableRowIsBlank(ct_0,ct_1)==false)  -).
To repeat through (T - table name) in (TC - table column) order begin -- end
    (documented at ph_repeattablecol):
    (- {-push-ctvs}
        for ({-ct-v0}={T},{-ct-v1}=TableNextRow({-ct-v0},{TC},0,1),ct_0={-ct-v0},ct_1={-ct-v1}:
            {-ct-v1}~=0:
            {-ct-v1}=TableNextRow({-ct-v0},{TC},{-ct-v1},1),ct_0={-ct-v0},ct_1={-ct-v1})  -).
To repeat through (T - table name) in reverse (TC - table column) order begin -- end
    (documented at ph_repeattablecolreverse):
    (- {-push-ctvs}
        for ({-ct-v0}={T},{-ct-v1}=TableNextRow({-ct-v0},{TC},0,-1),ct_0={-ct-v0},ct_1={-ct-v1}:
            {-ct-v1}~=0:
            {-ct-v1}=TableNextRow({-ct-v0},{TC},{-ct-v1},-1),ct_0={-ct-v0},ct_1={-ct-v1})  -).
```

§**50.** The equivalent of `break` or `continue` in C or I6, or of `last` or `next` in Perl. Here "in loop" means "in any of the forms of while or repeat".

```
Section SR5/3/6 - Control phrases - Changing the flow of loops

To break -- in loop
    (documented at ph_break):
    (- break; -).
To next -- in loop
    (documented at ph_next):
    (- continue; -).
```

§**51.** The following certainly aren't loops and aren't quite conditionals either, but they reflect the use of rules within rulebooks as being a form of control structure, and they have to be indexed somewhere.

The antique forms "yes" and "no" are now somewhat to be regretted, with "decide yes" and "decide no" being clearer ways to write the same thing. But we seem to be stuck with them.

```
Section SR5/3/7 - Control phrases - Deciding outcomes

To yes
    (documented at ph_yes):
    (- rtrue; -) - in to decide if only.
To decide yes
    (documented at ph_yes):
    (- rtrue; -) - in to decide if only.
To no
    (documented at ph_no):
    (- rfalse; -) - in to decide if only.
To decide no
    (documented at ph_no):
    (- rfalse; -) - in to decide if only.
```

§**52.** Note that returning a value has to invoke the type-checker to ensure that the return value matches the kind of value expected. This certainly rejects the phrase if it's used in a definition which isn't meant to be deciding a value at all, so an "in... only" clause is not needed.

```
To decide on (something - value)
    (documented at ph_decideon):
    (- return {-check-return-type:something}; -).
```

§**53.** "Do nothing" is useful mainly when other syntax has backed us into something clumsy, but it can't be dispensed with. (In the examples, it used to be used when conditions were awkward to negate – if condition, do nothing, otherwise blah blah blah – but the creation of "unless" made it possible to remove most of the "do nothing"s.)

```
Section SR5/3/8 - Control phrases - Stop or go

To do nothing (documented at ph_nothing):
    (- ; -).
To stop (documented at ph_stop):
    (- return; -) - in to only.
```

§**54. Actions, activities and rules.** We begin with the firing off of new actions. The current action runs silently if the I6 global variable `keep_silent` is set, so the result of the definitions below is that one can go into silence mode, using "try silently", but not climb out of it again. This is done because many actions try other actions as part of their normal workings: if we want action X to be tried silently, then any action X itself tries should also be tried silently.

```
Section SR5/4/1 - Actions, activities and rules - Trying actions

To try (doing something - action)
    (documented at ph_try):
    (- {doing something}; -).
To silently try (doing something - action)
    (documented at ph_trysilently):
    (- @push keep_silent; keep_silent=1; {doing something}; @pull keep_silent; -).
To try silently (doing something - action)
    (documented at ph_trysilently):
    (- @push keep_silent; keep_silent=1; {doing something}; @pull keep_silent; -).
```

§**55.** The requirements of the current action can be tested. The following may be reimplemented using a verb *to require* at some future point.

```
Section SR5/4/2 - Actions, activities and rules - Action requirements

To decide whether the action requires a touchable noun
    (documented at ph_requirestouch):
    (- (NeedToTouchNoun()) -).
To decide whether the action requires a touchable second noun
    (documented at ph_requirestouch2):
    (- (NeedToTouchSecondNoun()) -).
To decide whether the action requires a carried noun
    (documented at ph_requirescarried):
    (- (NeedToCarryNoun()) -).
To decide whether the action requires a carried second noun
    (documented at ph_requirescarried2):
    (- (NeedToCarrySecondNoun()) -).
To decide whether the action requires light
    (documented at ph_requireslight):
    (- (NeedLightForAction()) -).
```

§**56.** Within the rulebooks to do with an action, returning `true` from a rule is sufficient to stop the rulebook early: there is no need to specify success or failure because that is determined by the rulebook itself. (For instance, if the check taking rules stop for any reason, the action failed; if the after rules stop, it succeeded.) In some rulebooks, notably "instead" and "after", the default is to stop, so that execution reaching the end of the I6 routine for a rule will run into an `rtrue`. "Continue the action" prevents this.

```
Section SR5/4/3 - Actions, activities and rules - Stop or continue

To stop the action
    (documented at ph_stopaction):
    (- rtrue; -) - in to only.
To continue the action
    (documented at ph_continueaction):
    (- rfalse; -) - in to only.
```

§**57.** Note that we define "try", "silently try" and "try silently" all over again here, but for stored actions rather than actions: NI's type-checking means that it will automatically use whichever definition is appropriate.

```
Section SR5/4/4 - Actions, activities and rules - Stored actions

To decide what stored action is the current action
    (documented at ph_currentaction):
    (- STORED_ACTION_TY_Current({-pointer-to-new:stored action}) -).
To decide what stored action is the action of (A - action)
    (documented at ph_actionof):
    (- {A}{-backspace}{-backspace}, STORED_ACTION_TY_Current({-pointer-to-new:stored action})) -).
To try (S - stored action)
    (documented at ph_trystored):
    (- STORED_ACTION_TY_Try({S}); -).
To silently try (S - stored action)
    (documented at ph_trystoredsilently):
    (- STORED_ACTION_TY_Try({S}, true); -).
To try silently (S - stored action)
    (documented at ph_trystoredsilently):
    (- STORED_ACTION_TY_Try({S}, true); -).
To decide if (act - a stored action) involves (X - an object)
    (documented at ph_involves):
    (- (STORED_ACTION_TY_Involves({-pointer-to:act}, {X})) -).
To decide what action name is the action name part of (act - a stored action)
    (documented at ph_actionpart):
    (- (STORED_ACTION_TY_Part({-pointer-to:act}, 0)) -).
To decide what object is the noun part of (act - a stored action)
    (documented at ph_nounpart):
    (- (STORED_ACTION_TY_Part({-pointer-to:act}, 1)) -).
To decide what object is the second noun part of (act - a stored action)
    (documented at ph_secondpart):
    (- (STORED_ACTION_TY_Part({-pointer-to:act}, 2)) -).
To decide what object is the actor part of (act - a stored action)
    (documented at ph_actorpart):
    (- (STORED_ACTION_TY_Part({-pointer-to:act}, 3)) -).
```

§**58.** Firing off activities:

```
Section SR5/4/5 - Actions, activities and rules - Carrying out activities

To carry out the (A - activity on nothing) activity
    (documented at ph_carryout):
    (- CarryOutActivity({A}); -).
To carry out the (A - activity on value of kind K) activity with (val - K)
    (documented at ph_carryoutwith):
    (- CarryOutActivity({A}, {val}); -).
```

§**59.** This is analogous to "continue the action":

```
To continue the activity
    (documented at ph_continueactivity):
    (- rfalse; -) - in to only.
```

§**60.** Advanced activity phrases: for setting up one's own activities structured around I7 source text. People tend not to use this much, and perhaps that's a good thing, but it does open up possibilities, and it's good for retro-fitting onto extensions to make the more customisable.

```
Section SR5/4/6 - Actions, activities and rules - Advanced activities

To begin the (A - activity on nothing) activity
    (documented at ph_beginactivity):
    (- BeginActivity({A}); -).
To begin the (A - activity on value of kind K) activity with (val - K)
    (documented at ph_beginactivitywith):
    (- BeginActivity({A}, {val}); -).
To decide whether handling (A - activity) activity
    (documented at ph_handlingactivity):
    (- (~~(ForActivity({A}))) -).
To decide whether handling (A - activity on value of kind K) activity with (val - K)
    (documented at ph_handlingactivitywith):
    (- (~~(ForActivity({A}, {val}))) -).
To end the (A - activity on nothing) activity
    (documented at ph_endactivity):
    (- EndActivity({A}); -).
To end the (A - activity on value of kind K) activity with (val - K)
    (documented at ph_endactivitywith):
    (- EndActivity({A}, {val}); -).
To abandon the (A - activity on nothing) activity
    (documented at ph_abandonactivity):
    (- AbandonActivity({A}); -).
To abandon the (A - activity on value of kind K) activity with (val - K)
    (documented at ph_abandonactivitywith):
    (- AbandonActivity({A}, {val}); -).
```

§**61. Rules.** Four different ways to invoke a rule or rulebook:

```
Section SR5/4/7 - Actions, activities and rules - Following rules

To follow (RL - a rule)
    (documented at ph_follow):
    (- FollowRulebook({RL}); -).
To follow (RL - value of kind K based rule producing a value) for (V - K)
    (documented at ph_followfor):
    (- FollowRulebook({RL}, {V}, true); -).
To consider (RL - a rule)
    (documented at ph_consider):
    (- ProcessRulebook({RL}); -).
To consider (RL - value of kind K based rule producing a value) for (V - K)
    (documented at ph_considerfor):
    (- ProcessRulebook({RL}, {V}, true); -).
To decide what K is the (name of kind K) produced by (RL - rule producing a value of kind K)
    (documented at ph_producedby):
    (- ResultOfRule({RL}, 0, false, {-strong-kind:K}) -).
To decide what L is the (name of kind L) produced by (RL - value of kind K based rule
    producing a value of kind L) for (V - K)
    (documented at ph_producedbyfor):
    (- ResultOfRule({RL}, {V}, true, {-strong-kind:L}) -).
To abide by (RL - a rule)
    (documented at ph_abide):
    (- if (ProcessRulebook({RL})) rtrue; -) - in to only.
To abide by (RL - value of kind K based rule producing a value) for (V - K)
    (documented at ph_abidefor):
    (- if (ProcessRulebook({RL}, {V}, true)) rtrue; -) - in to only.
To anonymously abide by (RL - a rule)
    (documented at ph_abideanon):
    (- if (temporary_value = ProcessRulebook({RL})) {
        if (RulebookSucceeded()) ActRulebookSucceeds(temporary_value);
        else ActRulebookFails(temporary_value);
        return 2;
    } -) - in to only.
To anonymously abide by (RL - value of kind K based rule producing a value) for (V - K)
    (documented at ph_abideanon):
    (- if (temporary_value = ProcessRulebook({RL}, {V}, true)) {
        if (RulebookSucceeded()) ActRulebookSucceeds(temporary_value);
        else ActRulebookFails(temporary_value);
        return 2;
    } -) - in to only.
```

§**62.** Rules return `true` to indicate a decision, which could be either a success or a failure, and optionally may also return a value. If they return `false`, there's no decision.

```
Section SR5/4/8 - Actions, activities and rules - Success and failure

To make no decision
    (documented at ph_nodecision): (- rfalse; -) - in to only.
To rule succeeds
    (documented at ph_succeeds):
    (- RulebookSucceeds(); rtrue; -) - in to only.
To rule fails
    (documented at ph_fails):
    (- RulebookFails(); rtrue; -) - in to only.
To rule succeeds with result (val - a value)
    (documented at ph_succeedswith):
    (- RulebookSucceeds({-weak-kind-to-be-produced:val},{-check-success-type:val}); rtrue; -) - in to
o
  ... nly.
To decide if rule succeeded
    (documented at ph_succeeded):
    (- (RulebookSucceeded()) -).
To decide if rule failed
    (documented at ph_failed):
    (- (RulebookFailed()) -).
To decide which rulebook outcome is the outcome of the rulebook
    (documented at ph_rulebookoutcome):
    (- (ResultOfRule()) -).
```

§**63.** And lastly the suite of procedural rule tricks, though happily these have been less and less needed as Inform has grown in flexibility. (They incur an appreciable speed penalty at run-time.)

```
Section SR5/4/9 - Actions, activities and rules - Procedural manipulation

To ignore (RL - a rule)
    (deprecated)
    (documented at ph_ignore_dep):
    (- SuppressRule({RL}); -).
To reinstate (RL - a rule)
    (deprecated)
    (documented at ph_reinstate_dep):
    (- ReinstateRule({RL}); -).
To reject the result of (RL - a rule)
    (deprecated)
    (documented at ph_reject_dep):
    (- DonotuseRule({RL}); -).
To accept the result of (RL - a rule)
    (deprecated)
    (documented at ph_accept_dep):
    (- DonotuseRule({RL}); -).
To substitute (RL1 - a rule) for (RL2 - a rule)
    (deprecated)
    (documented at ph_substitute_dep):
    (- SubstituteRule({RL1},{RL2}); -).
To restore the original (RL1 - a rule)
```

```
    (deprecated)
    (documented at ph_restore_dep):
    (- SubstituteRule({RL1},{RL1}); -).
To move (RL1 - a rule) to before (RL2 - a rule)
    (deprecated)
    (documented at ph_movebefore_dep):
    (- MoveRuleBefore({RL1},{RL2}); -).
To move (RL1 - a rule) to after (RL2 - a rule)
    (deprecated)
    (documented at ph_moveafter_dep):
    (- MoveRuleAfter({RL1},{RL2}); -).
```

§**64. The model world.** Phrase definitions with wordings like "the story has ended" are a necessary evil. The "has" here is parsed literally, not as the present tense of *to have*, so inflected forms like "the story had ended" are not available: nor is there any value "the story" for the subject noun phrase to hold... and so on. Ideally, we would word all conditional phrases so as to avoid the verbs, but natural language just doesn't work that way.

```
Section SR5/5/1 - Model world - Ending the story

To end the story
    (documented at ph_end):
    (- deadflag=3; story_complete=false; -).
To end the story finally
    (documented at ph_endfinally):
    (- deadflag=3; story_complete=true; -).
To end the story saying (finale - text)
    (documented at ph_endsaying):
    (- deadflag={finale}; story_complete=false; -).
To end the story finally saying (finale - text)
    (documented at ph_endfinallysaying):
    (- deadflag={finale}; story_complete=true; -).
To decide whether the story has ended
    (documented at ph_ended):
    (- (deadflag~=0) -).
To decide whether the story has ended finally
    (documented at ph_finallyended):
    (- (story_complete) -).
To decide whether the story has not ended
    (documented at ph_notended):
    (- (deadflag==0) -).
To decide whether the story has not ended finally
    (documented at ph_notfinallyended):
    (- (story_complete==false) -).
To resume the story
    (documented at ph_resume):
    (- resurrect_please = true; -).

To end the game in death (deprecated)
    (documented at ph_enddeath_dep):
    (- deadflag=1; story_complete=false; -).
To end the game in victory (deprecated)
    (documented at ph_endvictory_dep):
    (- deadflag=2; story_complete=true; -).
To end the game saying (finale - text) (deprecated)
    (documented at ph_endgamesaying_dep):
    (- deadflag={finale}; story_complete=false; -).
To resume the game (deprecated)
    (documented at ph_resumegame_dep):
    (- resurrect_please = true; -).
To decide whether the game is in progress (deprecated)
    (documented at ph_inprogress_dep):
    (- (deadflag==0) -).
To decide whether the game is over (deprecated)
    (documented at ph_over_dep):
    (- (deadflag~=0) -).
To decide whether the game ended in death (deprecated)
```

```
    (documented at ph_endeddeath_dep):
    (- (deadflag==1) -).
To decide whether the game ended in victory (deprecated)
    (documented at ph_endedvictory_dep):
    (- (deadflag==2) -).
```

§**65.** Times of day.

```
Section SR5/5/2 - Model world - Times of day

To decide which number is the minutes part of (t - time)
    (documented at ph_minspart):
    (- ({t}%ONE_HOUR) -).
To decide which number is the hours part of (t - time)
    (documented at ph_hourspart):
    (- ({t}/ONE_HOUR) -).
```

§**66.** Comparing times of day is inherently odd, because the day is circular. Every 2 PM comes after a 1 PM, but it also comes before another 1 PM. Where do we draw the meridian on this circle? The legal day divides at midnight but for other purposes (daylight savings time, for instance) society often chooses 2 AM as the boundary. Inform uses 4 AM instead as the least probable time through which play continues. (Modulo a 24-hour clock, adding 20 hours is equivalent to subtracting 4 AM from the current time: hence the use of `20*ONE_HOUR` below.) Thus 3:59 AM is after 4:00 AM, the former being at the very end of a day, the latter at the very beginning.

```
To decide if (t - time) is before (t2 - time)
    (documented at ph_timebefore):
    (- ((({t}+20*ONE_HOUR)%(TWENTY_FOUR_HOURS))<(({t2}+20*ONE_HOUR)%(TWENTY_FOUR_HOURS))) -).
To decide if (t - time) is after (t2 - time)
    (documented at ph_timeafter):
    (- ((({t}+20*ONE_HOUR)%(TWENTY_FOUR_HOURS))>(({t2}+20*ONE_HOUR)%(TWENTY_FOUR_HOURS))) -).
To decide if it is before (t2 - time) (deprecated)
    (documented at ph_itisbefore_dep):
    (- (((the_time+20*ONE_HOUR)%(TWENTY_FOUR_HOURS))<(({t2}+20*ONE_HOUR)%(TWENTY_FOUR_HOURS))) -).
To decide if it is after (t2 - time) (deprecated)
    (documented at ph_itisafter_dep):
    (- (((the_time+20*ONE_HOUR)%(TWENTY_FOUR_HOURS))>(({t2}+20*ONE_HOUR)%(TWENTY_FOUR_HOURS))) -).
To decide which time is (t - time) before (t2 - time)
    (documented at ph_shiftbefore):
    (- (({t2}-{t}+TWENTY_FOUR_HOURS)%(TWENTY_FOUR_HOURS)) -).
To decide which time is (t - time) after (t2 - time)
    (documented at ph_shiftafter):
    (- (({t2}+{t}+TWENTY_FOUR_HOURS)%(TWENTY_FOUR_HOURS)) -).
```

§**67.** Durations are in effect casts from "number" to "time".

```
Section SR5/5/3 - Model world - Durations

To decide which time is (n - number) minutes
    (documented at ph_durationmins):
    (- (({n})%(TWENTY_FOUR_HOURS)) -).
To decide which time is (n - number) hours
    (documented at ph_durationhours):
    (- (({n}*ONE_HOUR)%(TWENTY_FOUR_HOURS)) -).
```

§**68.** Timed events.

```
Section SR5/5/4 - Model world - Timed events

To (R - rule) in (t - number) turn/turns from now
    (documented at ph_turnsfromnow):
    (- SetTimedEvent({-mark-event-used:R}, {t}+1, 0); -).
To (R - rule) at (t - time)
    (documented at ph_attime):
    (- SetTimedEvent({-mark-event-used:R}, {t}, 1); -).
To (R - rule) in (t - time) from now
    (documented at ph_timefromnow):
    (- SetTimedEvent({-mark-event-used:R}, (the_time+{t})%(TWENTY_FOUR_HOURS), 1); -).
```

§**69.** Scenes.

```
Section SR5/5/5 - Model world - Scenes

To decide if (sc - scene) has happened
    (documented at ph_hashappened):
    (- (scene_endings-->({sc}-1)) -).
To decide if (sc - scene) has not happened
    (documented at ph_hasnothappened):
    (- (scene_endings-->({sc}-1) == 0) -).
To decide if (sc - scene) has ended
    (documented at ph_hasended):
    (- (scene_endings-->({sc}-1) > 1) -).
To decide if (sc - scene) has not ended
    (documented at ph_hasnotended):
    (- (scene_endings-->({sc}-1) <= 1) -).
```

§**70.** Timing of scenes.

```
Section SR5/5/6 - Model world - Timing of scenes

To decide which time is the time since (sc - scene) began
    (documented at ph_scenetimesincebegan):
    (- (SceneUtility({sc}, 1)) -).
To decide which time is the time when (sc - scene) began
    (documented at ph_scenetimewhenbegan):
    (- (SceneUtility({sc}, 2)) -).
To decide which time is the time since (sc - scene) ended
    (documented at ph_scenetimesinceended):
    (- (SceneUtility({sc}, 3)) -).
To decide which time is the time when (sc - scene) ended
    (documented at ph_scenetimewhenended):
    (- (SceneUtility({sc}, 4)) -).
```

§**71.** Player's identity and location.

```
Section SR5/5/7 - Model world - Player's identity and location

To change the/-- player to (O - an object)
    (deprecated)
    (documented at ph_changeplayer_dep):
    (- ChangePlayer({O}); -).
To decide whether in (somewhere - an object)
    (deprecated)
    (documented at ph_in_dep):
    (- (WhetherIn({somewhere})) -).
To decide whether in darkness
    (documented at ph_indarkness):
    (- (location==thedark) -).
```

§**72.** Moving and removing things.

```
Section SR5/5/8 - Model world - Moving and removing things

To move (something - object) to (something else - object),
    without printing a room description
    or printing an abbreviated room description
    (documented at ph_move):
    (- MoveObject({something}, {something else}, {phrase options}, false); -).
To remove (something - object) from play
    (documented at ph_remove):
    (- RemoveFromPlay({something}); -).
To move (O - object) backdrop to all (D - description of objects)
    (documented at ph_movebackdrop):
    (- MoveBackdrop({O}, {D}); -).
To update backdrop positions
    (documented at ph_updatebackdrop):
    (- MoveFloatingObjects(); -).
```

§**73.** The map.

```
Section SR5/5/9 - Model world - The map

To decide which room is location of (O - object)
    (documented at ph_locationof):
    (- LocationOf({O}) -).
To decide which room is room (D - direction) from/of (R1 - room)
    (documented at ph_roomdirof):
    (- MapConnection({R1},{D}) -).
To decide which door is door (D - direction) from/of (R1 - room)
    (documented at ph_doordirof):
    (- DoorFrom({R1},{D}) -).
To decide which object is the other side of (D - door) from (R1 - room)
    (documented at ph_othersideof):
    (- OtherSideOfDoor({D},{R1}) -).
To decide which object is the direction of (D - door) from (R1 - room)
    (documented at ph_directionofdoor):
    (- DirectionDoorLeadsIn({D},{R1}) -).
To decide which object is room-or-door (D - direction) from/of (R1 - room)
    (documented at ph_roomordoor):
    (- RoomOrDoorFrom({R1},{D}) -).
To change (D - direction) exit of (R1 - room) to (R2 - room)
    (documented at ph_changeexit):
    (- AssertMapConnection({R1},{D},{R2}); -).
To change (D - direction) exit of (R1 - room) to nothing/nowhere
    (documented at ph_changenoexit):
    (- AssertMapConnection({R1},{D},nothing); -).
To decide which room is the front side of (D - object)
    (documented at ph_frontside):
    (- FrontSideOfDoor({D}) -).
To decide which room is the back side of (D - object)
    (documented at ph_backside):
    (- BackSideOfDoor({D}) -).
```

§**74.** Route-finding.

```
Section SR5/5/10 - Model world - Route-finding

To decide which object is best route from (R1 - object) to (R2 - object),
    using doors or using even locked doors
    (documented at ph_bestroute):
    (- MapRouteTo({R1},{R2},0,{phrase options}) -).
To decide which number is number of moves from (R1 - object) to (R2 - object),
    using doors or using even locked doors
    (documented at ph_bestroutelength):
    (- MapRouteTo({R1},{R2},0,{phrase options},true) -).
To decide which object is best route from (R1 - object) to (R2 - object) through
    (RS - description of objects),
    using doors or using even locked doors
    (documented at ph_bestroutethrough):
    (- MapRouteTo({R1},{R2},{RS},{phrase options}) -).
To decide which number is number of moves from (R1 - object) to (R2 - object) through
    (RS - description of objects),
```

```
    using doors or using even locked doors
    (documented at ph_bestroutethroughlength):
    (- MapRouteTo({R1},{R2},{RS},{phrase options},true) -).
```

§**75.** The object tree.

```
Section SR5/5/11 - Model world - The object tree

To decide which object is holder of (something - object)
    (documented at ph_holder):
    (- (HolderOf({something})) -).
To decide which object is next thing held after (something - object)
    (documented at ph_nextheld):
    (- (sibling({something})) -).
To decide which object is first thing held by (something - object)
    (documented at ph_firstheld):
    (- (child({something})) -).
```

§**76.** Last, and since it's deprecated, least:

```
Section SR5/5/12 - Model world - Score

To award (some - number) point/points
    (deprecated)
    (documented at ph_awardpoints_dep):
    (- score=score+{some}; -).
```

§**77. Understanding.** First, asking yes/no questions.

```
Section SR5/6/1 - Understanding - Asking yes/no questions

To decide whether player consents
    (documented at ph_consents):
        (- YesOrNo() -).
```

§**78.** Support for snippets, which are substrings of the player's command.

```
Section SR5/6/2 - Understanding - The player's command

To decide if (S - a snippet) matches (T - a topic)
    (documented at ph_snippetmatches):
    (- (SnippetMatches({S}, {T})) -).
To decide if (S - a snippet) does not match (T - a topic)
    (documented at ph_snippetdoesnotmatch):
    (- (SnippetMatches({S}, {T}) == false) -).
To decide if (S - a snippet) includes (T - a topic)
    (documented at ph_snippetincludes):
    (- (matched_text=SnippetIncludes({T},{S})) -).
To decide if (S - a snippet) does not include (T - a topic)
    (documented at ph_snippetdoesnotinclude):
    (- (SnippetIncludes({T},{S})==0) -).
```

§**79.** Changing the player's command.

```
Section SR5/6/3 - Understanding - Changing the player's command

To change the text of the player's command to (txb - indexed text)
    (documented at ph_changecommand):
    (- SetPlayersCommand({-pointer-to:txb}); -).
To replace (S - a snippet) with (T - text)
    (documented at ph_replacesnippet):
    (- SpliceSnippet({S}, {T}); -).
To cut (S - a snippet)
    (documented at ph_cutsnippet):
    (- SpliceSnippet({S}, 0); -).
To reject the player's command
    (documented at ph_rejectcommand):
    (- RulebookFails(); rtrue; -) - in to only.
```

§**80.** Scope and pronouns.

```
Section SR5/6/4 - Understanding - Scope and pronouns

To place (O - an object) in scope, but not its contents
    (documented at ph_placeinscope):
    (- PlaceInScope({O}, {phrase options}); -).
To place the/-- contents of (O - an object) in scope
    (documented at ph_placecontentsinscope):
    (- ScopeWithin({O}); -).
To set pronouns from (O - an object)
    (documented at ph_setpronouns):
    (- PronounNotice({O}); -).
To set pronouns from possessions of the player
    (deprecated)
    (documented at ph_setpronouns_dep):
    (- PronounNoticeHeldObjects(); -).
```

§**81. Message support.** "Unindexed" here is a euphemism for "undocumented". This is where experimental or intermediate phrases go: things we don't want people to use because we will probably revise them heavily in later builds of Inform. For now, the Standard Rules do make use of these phrases, but nobody else should. They will change without comment in the change log.

```
Section SR5/8/1 - Message support - Issuance - Unindexed

To stop the action with library message (AN - an action name) number
    (N - a number) for (H - an object):
    (- return GL__M({AN},{N},{H}); -) - in to only.
To stop the action with library message (AN - an action name) number
    (N - a number):
    (- return GL__M({AN},{N},noun); -) - in to only.
To issue miscellaneous library message number (N - a number):
    (- GL__M(##Miscellany,{N}); -).
To issue miscellaneous library message number (N - a number) for (H - an object):
    (- GL__M(##Miscellany,{N}, {H}); -).
To issue library message (AN - an action name) number
    (N - a number) for (H - an object):
    (- GL__M({AN},{N},{H}); -).
To issue library message (AN - an action name) number
    (N - a number) for (H - an object) and (H2 - an object):
    (- GL__M({AN},{N},{H},{H2}); -).
To issue library message (AN - an action name) number (N - a number):
    (- GL__M({AN},{N},noun); -).
To issue actor-based library message (AN - an action name) number
    (N - a number) for (H - an object) and (H2 - an object):
    (- AGL__M({AN},{N},{H},{H2}); -).
To issue actor-based library message (AN - an action name) number
    (N - a number) for (H - an object):
    (- AGL__M({AN},{N},{H}); -).
To issue actor-based library message (AN - an action name) number (N - a number):
    (- AGL__M({AN},{N},noun); -).

To issue score notification message:
    (- NotifyTheScore(); -).
To say pronoun dictionary word:
    (- print (address) pronoun_word; -).
To say recap of command:
    (- PrintCommand(); -).
The pronoun reference object is an object that varies.
The pronoun reference object variable translates into I6 as "pronoun_obj".
The library message action is an action name that varies.
The library message action variable translates into I6 as "lm_act".
The library message number is a number that varies.
The library message number variable translates into I6 as "lm_n".
The library message amount is a number that varies.
The library message amount variable translates into I6 as "lm_o".
The library message object is an object that varies.
The library message object variable translates into I6 as "lm_o".
The library message actor is an object that varies.
The library message actor variable translates into I6 as "actor".
The second library message object is an object that varies.
The second library message object variable translates into I6 as "lm_o2".
```

§**82.** Intervention. These are hooks for the new library messages system coming in a later build; they won't last.

```
Section SR5/8/2 - Message support - Intervention - Unindexed

To decide if intervened in miscellaneous message:
    decide on false;

To decide if intervened in miscellaneous list message:
    decide on false;

To decide if intervened in action message:
    decide on false;
```

§**83. Miscellaneous other phrases.** Again, *these are not part of Inform's public specification.*

```
Section SR5/9/1 - Miscellaneous other phrases - Unindexed
```

§**84.** These are actually sensible concepts in the world model, and could even be opened to public use, but they're quite complicated to explain.

```
To decide which object is the component parts core of (X - an object):
    (- CoreOf({X}) -).
To decide which object is the common ancestor of (O - an object) with
    (P - an object):
    (- (CommonAncestor({O}, {P})) -).
To decide which object is the not-counting-parts holder of (O - an object):
    (- (CoreOfParentOfCoreOf({O})) -).
To decide which object is the visibility-holder of (O - object):
    (- VisibilityParent({O}) -).
To calculate visibility ceiling at low level:
    (- FindVisibilityLevels(); -).
```

§**85.** These are in effect global variables, but aren't defined as such, to prevent people using them. Their contents are only very briefly meaningful, and they would be dangerous friends to know.

```
To decide which number is the visibility ceiling count calculated:
    (- visibility_levels -).
To decide which object is the visibility ceiling calculated:
    (- visibility_ceiling -).
```

§**86.** This is a unique quasi-action, using the secondary action processing stage only. A convenience, but also an anomaly, and let's not encourage its further use.

```
To produce a room description with going spacing conventions:
    (- LookAfterGoing(); -).
```

§**87.** An ugly little trick needed because of the mismatch between I6 and I7 property implementation, and because of legacy code from the old I6 library. Please don't touch.

```
To print the location's description:
    (- PrintOrRun(location, description); -).
```

§**88.** This is a bit trickier than it looks, because it isn't always set when one thinks it is.

```
To decide whether the I6 parser is running multiple actions:
    (- (multiflag==1) -).
```

§**89.** Again, the following cries out for an enumerated kind of value.

```
To decide if set to sometimes abbreviated room descriptions:
    (- (lookmode == 1) -).
To decide if set to unabbreviated room descriptions:
    (- (lookmode == 2) -).
To decide if set to abbreviated room descriptions:
    (- (lookmode == 3) -).
```

§**90.** Action conversion is a trick used in the Standard Rules to simplify the implementation of actions: it allows one action to become another one mid-way, without causing spurious action failures. (There are better ways to make user-defined actions convert, and some of the examples show this.)

```
To convert to (AN - an action name) on (O - an object):
    (- return GVS_Convert({AN},{O},0); -) - in to only.
To convert to request of (X - object) to perform (AN - action name) with
    (Y - object) and (Z - object):
    (- TryAction(true, {X}, {AN}, {Y}, {Z}); rtrue; -).
To convert to special going-with-push action:
    (- ConvertToGoingWithPush(); rtrue; -).
```

§**91.** The "surreptitiously" phrases shouldn't be used except in the Standard Rules because they temporarily violate invariants for the object tree and the light variables; the SR uses them carefully in situations where it's known to work out all right.

```
To surreptitiously move (something - object) to (something else - object):
    (- move {something} to {something else}; -).
To surreptitiously move (something - object) to (something else - object) during going:
    (- MoveDuringGoing({something}, {something else}); -).
To surreptitiously reckon darkness:
    (- SilentlyConsiderLight(); -).
```

§**92.** This is convenient for debugging Inform, but for no other purpose. It toggles verbose logging of the type-checker.

```
To ***:
    (- {-verbose-checking} -).
```

§**93.** And so, at last...

```
The Standard Rules end here.
```

§**94.** ...except that this is not quite true, because like most extensions they then quote some documentation for Inform to weave into index pages: though here it's more of a polite refusal than a manual, since the entire system documentation is really the description of what was defined in this extension.

```
---- DOCUMENTATION ----
Unlike other extensions, the Standard Rules are compulsorily included
with every project. They define the phrases, kinds and relations which
are basic to Inform, and which are described throughout the documentation.
```