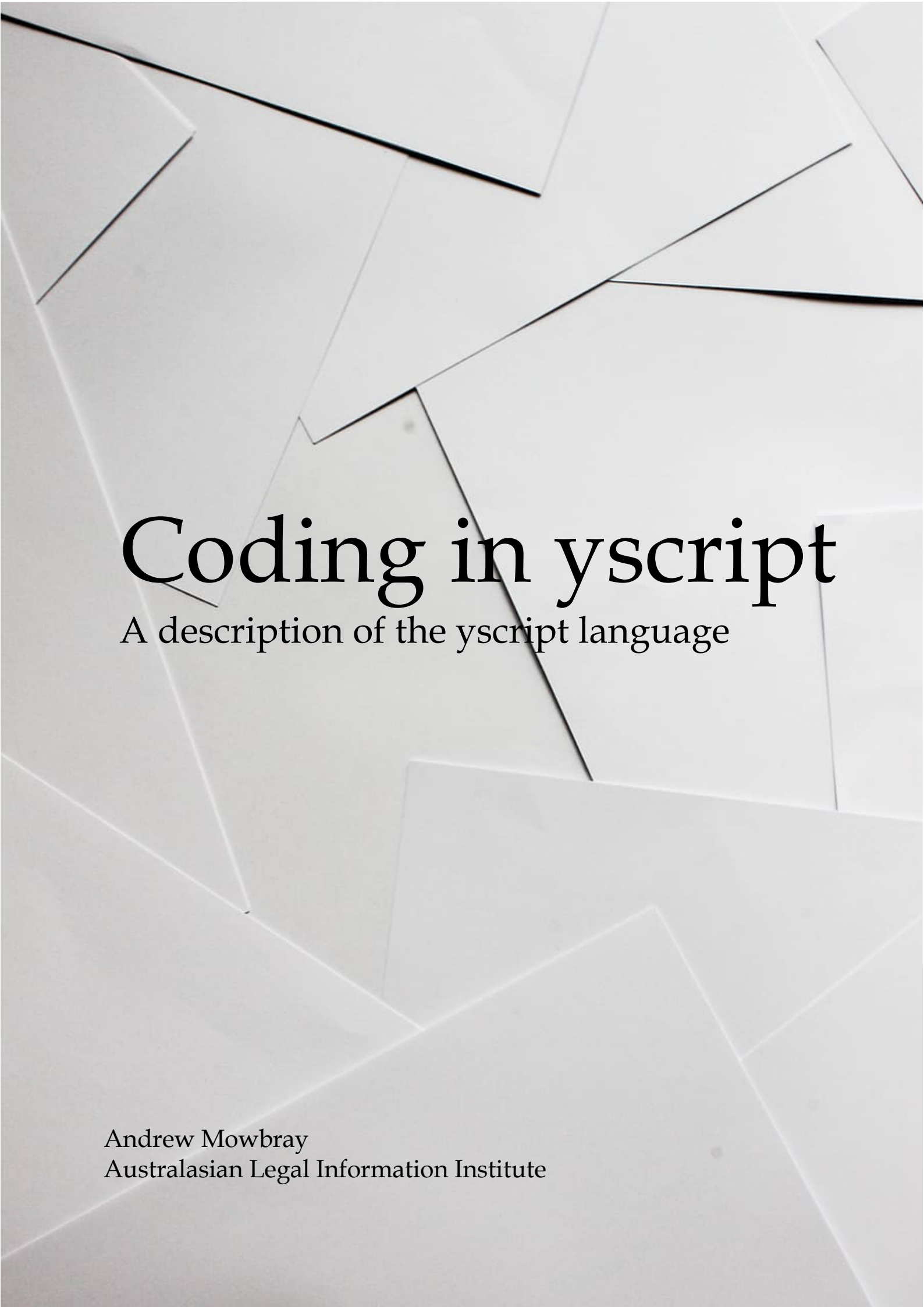

Coding in yscript

A description of the yscript language

Andrew Mowbray
Australasian Legal Information Institute

Coding with yscript – A description of the yscript language

Andrew Mowbray *is Professor of Law and Information Technology, University of Technology Sydney and Co-Director, AustLII*

Australasian Legal Information Institute (AustLII) Level 14, 61 Broadway
Ultimo NSW 2007 Australia

Tel: +61 2 9514 4918
Email: andrew@austlii.edu.au
Web: http://www.austlii.edu.au/

AustLII is a joint facility of UTS and UNSW Faculties of Law.

TABLE OF CONTENTS

TABLE OF FIGURES

Chapter 1: Introduction

yscript[1] (pronounced 'why-script') is a computer language for representing and manipulating propositions. It can be used to represent real-world rules such as legislation or codes of practice, as well as to create systems based around less formally defined procedures or knowledge.

When yscript code is executed, it results in a dialog or *consultation*. A series of questions are asked, and conclusions are made. Along the way, the user can interrogate the system as to why questions are being asked, to explain how conclusions have been reached and to check hypotheticals. When a session completes, a report is generated to give an answer to the original goals and to explain why this is the case.

The core language has been stable for a long time.[2] From the outset, one of the central aims was to develop a form of representation that looked as much like natural language as possible. The language syntax manages to almost entirely avoid the use of symbols which are the principle structural elements of most programming languages. This was done partly to make it easier to write code for non-programmers, but also to make the code more transparent. Even if you can't write code in yscript, you can probably understand what it is doing and possibly even comment upon whether it accurately encapsulates anything from the real world that it is meant to reflect.

In formal terms, yscript code consists of *rules* that deal with *facts*. Facts are expressed in their plain English-language form. Individual rules are imperative but often just declare relationships between facts. Once a rule is being evaluated, other rules that can help determine a value for required facts are automatically executed in a goal-oriented fashion. Each time that a new fact becomes known, rules are used to check if other fact values can be derived. When required, rules can be specifically called like procedures or functions in other languages.

yscript also supports *examples*[3]. An *example* is a set of propositions supporting a particular outcome. This was implemented in yscript to deal with case-based reasoning in law but probably has application in other fields. Finally, yscript applications can generate *documents*. A *document* is built procedurally but can take

---

[1] The *yscript* language was originally developed for the expert systems shell *ysh*. Prior to being integrated into AustLII's DataLex platform, yscript was also used as the language and code interpreter for a system called wysh (short for "web-ysh"). For a history of the project see Greenleaf, Mowbray & Chung, 'Building Sustainable Free Legal Advisory Systems: Experiences from the History of AI & Law' (2018) 34(1) *Computer Law & Security Review* 324

[2] The language has been substantially extended since inception, but the core elements have remained the same and, with very few minor exceptions code written for the earliest versions of *yscript* will still run under the current interpreter. A summary of changes between ysh version 1.x and yscript 2.1x are set out in Appendix 3 at page 139.

[3] The yscript library uses an analogous reasoning approach developed by Alan Tyree called PANNDA. This is described in his book Tyree, A, *Expert Systems in Law*, Prentice Hall, 1990.

advantage of rule-based determination of necessary facts. Document assembly can also use external templates.

One of the advantages of using a quasi-natural language form of coding is that it is often possible to directly adopt the wording of a set of rules that you are trying to represent. This is particularly useful in the legal domain, where it is desirable to use language taken from statutes and regulations. In legal documents, the exact form of words often matters. Specific words and phrases can imply underlying complex meaning or act as a point of reference to a body of interpretative documents and decisions which need to be considered.

The rule-based structure of yscript encourages and supports *isomorphism* (that is, one-to-mapping) of real-word rules into yscript code. This makes it a lot easier to build applications, but more importantly it allows for simpler maintenance of the code as source legislation or other rules change.

yscript can be used directly via the *cyscript* command-line interpreter or it can be embedded in larger systems and applications. A major environment that uses yscript is AustLII's *DataLex* platform. This uses yscript to answer legal problems within the AustLII web-based environment and provides links to and from yscript dialogs to primary and secondary legal materials that provide support and explanation. For other applications, yscript provides a library with as associated API that supports most programming languages including C, C++, Python, Ruby, and Perl.

The structure of the rest of this guide is as follows:

Chapter 2 gives a quick tutorial introduction designed to provide you with enough information to start writing code. Whilst yscript provides many features, you should be able to do 90% of the things you need to do with 10% of the language. It is possible to build quite large and sophisticated applications with little more information than is contained in the tutorial.

The following chapters describe the language in detail and are also meant to serve as a reference source.

Chapter 3 is a technical chapter describing the core elements of the language. It discusses the basic syntax, data types, constants and expressions and contains several tables enumerating fundamental language elements.

Chapter 4 deals with *facts*. At one level, facts are like variables in other languages. In yscript, however, facts are also the principal way in which an application communicates with the user. This chapter deals with how fact names are transformed into questions and statements and how you can change this if necessary. The chapter also deals with the use of *context*. This is important when building larger or collaborative applications which need to be divided into separate and independent parts.

Chapter 5 discusses *statements*. Most of the statements that are used in typical yscript code are just assignments and `IF–THEN–ELSE` statements. Whilst it is probably not

desirable if you are trying to write reusable declarative code, yscript also supports most of the basic statement constructs that can be found in conventional languages.

Chapter 6 is all about *rules*. Rules are generally invoked in a goal-driven (backward chaining) way. In some circumstances, it is necessary to use rules differently. Other types of rules such as *forward* rules, *daemons*, and procedures are discussed. This chapter also describes how rules can be declared as *goals*.

Chapter 7 introduces *examples*. Examples are used to support analogous reasoning in yscript. They were mainly introduced to deal with handling the application of legal case-based reasoning but can be used more generally.

Chapter 8 describes the *document assembly* facilities available. Document assembly is imperative by its nature but can make use of rules to determine surrounding issues and which parts of a document need to be created.

The final chapter – Chapter 9, makes various suggestions for how to write good yscript code. Some of the principles that are suggested are quite straight-forward (such as *not using a yscript feature just because it is there*) but are based on experience with building applications and teaching others to do so and are intended to be practical.

The appendices contain a detailed description of available units of measurement and currencies, the interactive command line environment – *cyscript*, a summary of recent changes to language features, a formal description of the yscript grammar, a description of the interpreter library API, a guide to using cyscript as a server and several extended examples.

Chapter 2: Getting Started

This chapter provides a quick tutorial introduction to using yscript. It is designed to present the essential elements of the language in a practical way. Most of the details are skipped in favour of giving enough information to start writing useful applications. All examples are complete pieces of code and can be run as is or modified to allow you to play with the language and to become comfortable with it.

2.1 Hello Rainy World

yscript code consists of a set of declarations of *rules* and *facts*. When evaluated, execution begins by firing a rule and generally proceeds by using other rules to determine the values of *facts* as they are needed.

Consider the following code:[4]

```
RULE PROVIDES
you should take an umbrella ONLY IF it is raining
```

The first thing you will notice is that some of the words ("`RULE PROVIDES`" and "`ONLY IF`") are in upper case. These are reserved words or *keywords* and are used to provide structure. Other phrases are in lower case, such as "you should take an umbrella" and "it is raining". These are English language[5] propositions or *facts*. Facts do not necessarily have to be propositional and can have others sorts of values, but we will leave discussing this until later.

The meaning of the rule is meant to be transparent, namely that it will be determined that you should take an umbrella if it is raining, and it will be concluded that you should not take an umbrella if it is not. The `ONLY IF` operator indicates a propositional (or *boolean*) assignment.

When evaluated, the yscript interpreter will execute (*fire*) the rule and attempt to assign the value of the fact "it is raining" to the fact "you should take an umbrella". Because there is no other rule that can help to determine a value for "it is raining", the system will ask the user. If the user answers "yes" then it will be determined that "you should take an umbrella", otherwise it will be concluded that "you should not take an umbrella". If the user answers "unknown", it will be concluded that "it is uncertain whether or not you should take an umbrella".

---

[4] The *umbrella* code is the equivalent of the "Hello, World" example that starts most computer language tutorials. It is not a very "*yscript*" sort of thing to want to do, but if you did want to write code to output "Hello, World", you would write: `RULE PROVIDES SAY "Hello, World"`.

[5] It is possible at some stage in the future that *yscript* may be adapted to work in other languages, but for the moment proposition transformations will only work in English.

yscript code is contained in one or more files, that usually have the suffix - `.ys`. Most legal users will be using yscript as part of the *DataLex* platform[6] (see http://datalex.org/dev/tools/>[7]), but if you are running it from the standard distribution under Unix or Linux, code is interpreted by the `ys` command line interpreter - *cyscript*.[8] Assuming the above code was saved in a file called - "`rain.ys`", the user session might go something like:

```
% ys rain.ys

Is it raining?
** why

This will help determine whether or not you should take an umbrella.

Is it raining?
** no

You should not take an umbrella because it is not raining.
```

In the dialog that occurs during the above session, facts are automatically turned into questions and expressed in the positive and negative form as necessary. Explanations for conclusions are also built automatically, as are explanations for "why" questions are being considered (so-called *explication*). The use of natural language propositions and descriptions for both identifying facts and for generating interaction with an application's user is one of yscript's distinguishing features. It is designed to make applications quick to build and easy to maintain.

2.2 Adding More Rules

The example can be expanded by adding additional rules and by using *expressions* to create more complicated propositions using logical operators as follows:

```
RULE PROVIDES
you should take and umbrella ONLY IF
    you have an umbrella AND
    you might need an umbrella

RULE PROVIDES
you might need an umbrella ONLY IF
    it is raining OR
    it looks like it might rain

RULE PROVIDES
it is raining ONLY IF
    water is falling from the sky
```

---

[6] *DataLex* is a legal applications development environment which is available on AustLII (see http://www.datalex.org/). It incorporates yscript in a system that integrates it with the primary and secondary legal materials on AustLII and facilitates cooperative code development.

[7] If you just want to test and run yscript code, skip the first section ("Import legislation section") and just type your code into the second box "Edit DataLex Code-base" and "Run consultation".

[8] The yscript standard distribution also includes a C-library API with examples of how to call this from C, C++, Python, Ruby, and Perl. It is also possible to run cyscript as a server on a Unix socket. See Appendices 5 and 6.

When more than one rule is present, by default execution begins with the first rule and the following interaction will result:

```
Do you have an umbrella?
** yes

Is water falling from the sky?
** no

Does it look like it might rain?
** yes

You should take an umbrella because you have an umbrella and
you might need an umbrella. You might need an umbrella because
it looks like it might rain.
```

2.3 A Slightly More Legal Example

As was said in the introduction, the main application of yscript has been to build applications in the legal domain, so let's now consider a slightly more legal example:

```
PERSON the client
PERSON the defendant

RULE Litigation Basics PROVIDES
the client should sue the defendant ONLY IF
    the client has a good cause of action AND
    the defendant has the capacity to pay damages

RULE Consequences of Insolvency PROVIDES
IF the defendant is insolvent THEN
    the defendant does not have the capacity to pay damages
```

This example introduces a few more elements of the yscript language. The first thing that is different is that the rules have names: "Litigation Basics" and "Consequences of Insolvency". Rule names are generally optional,[9] but it is good practice to include them as rule names provide documentation as to what a rule does and perhaps what it is based on. These descriptions can be used to describe the hierarchy of rules under consideration and for selecting goal rules for sessions.

The first rule uses the same sort of `ONLY IF` statement that has been used in the previous two examples. The second rule introduces the "`IF-THEN`"[10] statement. This statement allows you to specify that some conclusion should be made (in the case "`the defendant does not have the capacity to pay damages`") if some premise ("`the defendant is insolvent`") is true. If the premise is not true, then no conclusion will be made.

The code starts with the declarations "`PERSON the client`" and "`PERSON the defendant`". These statements declare the *named subjects* – "the client" and "the defendant". A

[9] Rules must be named if you specifically want to invoke them with the `CALL` statement or if they are to be used as `GOALS`. Rule names are also used by the verbose tracking facility.

[10] As we will see later, you can follow an `IF-THEN` with an `ELSE` clause as is the case for most programming languages. You can also nest `IF-THEN-ELSE` statements and specify that they should apply to more than one statement by using `BEGIN-END` blocks. See § 5.2 at page 64.

named subject is a string that forms part of the description (name) of other facts. In the example, both "the client" and "the defendant" appear as part of the fact "the client should sue the defendant". These named subjects will be replaced by a name or an appropriate pronoun when the fact is used in a question or explanation. Other types of named subjects include `THING`s (i.e., something that has a name but is inanimate like a company or a place) and `PERSON-THING`s (i.e., an entity that can be animate or inanimate).[11]

When executed, the session goes:

```
1) What is the name of the client?
   ** Maryanne Smithers

2) What is Maryanne Smithers's preferred gender?
   ** female

3) What is Maryanne Smithers's preferred form of address?
   ** Professor Smithers

4) What is the name of the defendant?
   ** George Dobinson

5) What is George Dobinson's preferred gender?
   ** male

6) What is George Dobinson's preferred form of address?
   ** Dr

7) Does Professor Smithers have a good cause of action?
   ** yes

8) Is Dr Dobinson insolvent?
   ** no

9) Does he have the capacity to pay damages?
   ** yes

   Professor Smithers should sue Dr Dobinson because she has
   a good cause of action and Dr Dobinson has the capacity
   to pay damages.
```

2.4 Rules as Code

The most common application of yscript has been to represent legislation, regulations, and formal rules (*Rules as Code*[12] or *RaC*). Using the elements introduced so far in this tutorial, it is possible to represent the provisions of most forms of legislative text. Because we generally want to be as legally correct as possible, it is generally desirable to use the exact language from source legislation as yscript facts. It is also good practice to map the structure and organisation of legislation to rules to make code easier to maintain and to facilitate better explanations.

---

[11] In the example, "the client" and "the defendant" probably should really be of type `PERSON-THING` as both parties could potentially be corporations or unincorporated associations.

[12] For a discussion of *Rules as Code* generally, see J Mohun and A Roberts, "Cracking the code: Rulemaking for humans and machines", *OECD Working Papers on Public Governance*, No. 42, OECD Publishing, Paris 2020.

Consider the following section from NSW *Crimes Act 1900* which deals with "*Setting trap etc*":

```
Setting trap etc
49 SETTING TRAP ETC
(1) Any person who--
(a) places or sets, or causes to be placed or set, any trap, device or thing
    (whether its nature be electronic, electric, mechanical, chemical or
    otherwise) capable of destroying human life or inflicting grievous bodily
    harm on any person, or
(b) knowingly permits any such trap, device or thing to continue to be placed
    or set,
    with intent to inflict grievous bodily harm shall be liable to imprisonment
    for five years.
(2) Nothing in subsection (1) shall extend to any gin or trap, placed with the
    intention of destroying vermin, or to any trap, device or thing placed in
    a dwelling-house for the protection thereof.
```

To represent this in yscript, we should reflect that the overall structure of the section has two subsections (1) and (2) and subsection (1) has two sub-subsections (a) and (b). We should also try to keep as much of the language as we can, but we will need to make some judgment calls about how requirements should be broken up and what the overall outcome of each yscript rule should be.

The main purpose of the section is to provide for an offence of "setting a trap" with a penalty of imprisonment for five years where either subsections (1)(a) or (1)(b) apply and where subsection (2) does not. The opening rules might go something like:

```
RULE Crimes Act 1900 Section 49 PROVIDES
the defendant has committed an offence under section 49 ONLY IF
    section 49(1) applies AND
    section 49(2) does not apply

RULE Crimes Act 1900 Section 49(1) PROVIDES
section 49(1) applies ONLY IF
    section 49(1)(a) applies AND/OR
    section 49(1)(b) applies AND
    the defendant had the intent to inflict grievous bodily harm
```

Note that we have had to introduce a new operator - `AND/OR`. This is the same as an `OR`, but it has higher precedence (that is, it binds more tightly) than `AND`, so that either (a) or (b) must apply, as well as the defendant having intent to inflict grievous bodily harm.

If we ran this code, it would go:

```
1) Does section 49(1)(a) apply?
   ** yes

2) Did the defendant have the intent to inflict grievous bodily harm?
   ** yes

3) Does section 49(2) apply?
   ** no

   The defendant has committed an offence under section 49 because section
  49(1) applies and section 49(2) does not apply. Section 49(1) applies
  because section 49(1)(a) applies and the defendant had the intent to inflict
  grievous bodily harm.
```

To complete the section translation and make the application more useful, we could add:

```
PERSON the defendant

RULE Crimes Act 1900 Section 49 PROVIDES
the defendant has committed an offence under section 49 ONLY IF
    section 49(1) applies AND
    section 49(2) does not apply

RULE Crimes Act 1900 Section 49(1) PROVIDES
section 49(1) applies ONLY IF
    section 49(1)(a) applies AND/OR
    section 49(1)(b) applies AND
    the defendant had the intent to inflict grievous bodily harm

RULE Crimes Act 1900 Section 49(1)(a) PROVIDES
section 49(1)(a) applies ONLY IF
    the defendant placed or set a trap, device or thing AND/OR
    the defendant caused a trap, device or thing to be placed or
    set AND
    the trap, device or thing was capable of destroying human
    life AND/OR
    the trap, device or thing was capable of inflicting grievous
    bodily harm on any person

RULE Crimes Act 1900 Section 49(1)(b) PROVIDES
section 49(1)(b) applies ONLY IF
    the defendant knowingly permitted a trap, device or thing to
    continue to be placed or set AND
    the trap, device or thing was capable of destroying human life
    or inflicting grievous bodily harm on any person

RULE Crimes Act 1900 Section 49(2) PROVIDES
section 49(2) applies ONLY IF
    the trap was placed with the intention of destroying vermin OR
    the trap was placed in a dwelling house to protect against
    vermin
```

One of the interesting features of the example is the amount of statutory interpretation involved in making the statute work as code. In section 49(2), for example, we have had to require that the exception for traps relating to protection of dwelling houses only operates in the context of vermin.

When run, the session might go:

```
1) What is the name of the defendant?
   ** Howard Jones

2) What is Howard Jones's preferred gender?
   ** male

3) What is Howard Jones's preferred form of address?
   ** Mr Jones

4) Did Mr Jones place or set a trap, device or thing?
   ** no

5) Did he cause a trap, device or thing to be placed or set?
   ** no

6) Did he knowingly permit a trap, device or thing to continue to
```

```
   be placed or set?
   ** yes

 7) Was the trap, device or thing capable of destroying human life
   or inflicting grievous bodily harm on any person?
   ** yes

 8) Did Mr Jones have the intent to inflict grievous bodily harm?
   ** yes

 9) Was the trap placed with the intention of destroying vermin?
   ** no

10) Was the trap placed in a dwelling house to protect against
   vermin?
   ** no

   Mr Jones has committed an offence under section 49 because
   section 49(1) applies and section 49(2) does not apply.

   Section 49(1) applies because section 49(1)(b) applies and
   Mr Jones had the intent to inflict grievous bodily harm.

   Section 49(1)(b) applies because Mr Jones knowingly
   permitted a trap, device or thing to continue to be placed
   or set and the trap, device or thing was capable of
   destroying human life or inflicting grievous bodily harm on
   any person.

   Section 49(2) does not apply because the trap was not placed
   with the intention of destroying vermin and the trap was not
   placed in a dwelling house to protect against vermin.
```

2.5 Review

Reviewing the information covered in this chapter in half a page, yscript code consists of rules which often have the form:

```
RULE rule-name PROVIDES
some conclusion is reached ONLY IF
    some premise applies AND
    some other premise applies
```

or sometimes:

```
RULE rule-name PROVIDES
IF some premise applies THEN
    some conclusion is reached
```

You can create more complicated conditions by replacing `AND` with the `OR` operator or the `AND/OR` operator.

Named subjects can be introduced with the keywords `PERSON`, `THING` and `PERSON-THING` as in:

```
PERSON description of person's role
THING description of non-person subject
PERSON-THING description of a subject that is animate or inanimate
```

If you have followed up until now, you should now be able to write yscript code and be able to create quite large and sophisticated applications. You can probably put off reading the rest of this manual until you have had the chance to experiment.

Chapter 3: Syntax, Types, Constants and Expressions

This chapter describes yscript syntax, the basic data *types* that it supports and *expressions* that can be used to combine *constants* and *facts*. The content includes a formal description of the fundamental elements of language, so you might just want to skim through it and use it as a reference source.

3.1 Keywords and Descriptors

The syntax of yscript code is different from most computer languages in that it makes very limited use of non-alphanumeric symbols. Generally, characters such as '=', '+' or '&' have no special meaning.[13] The code consists of *keywords* and *descriptors.* A *keyword* is a reserved term that is always written in upper case. The full list[14] of keywords is:

ALIAS	ALL	AND	AND/OR	AND/OR/WITH	AND/WITH
AS	ASSERT	ATTACH	AUTHORITY	BACKWARD	BEGIN
BOOLEAN	BY	CALL	CASE	CONTEXT	DAEMON
DATE	DAY	DAYS	DEFAULT	DEFINITE	DETERMINE
DISPLAYED	DIVIDED	DO	DOCUMENT	DOLLAR	DOLLARS
ELSE	END	END-SECTION	EQUAL	EQUALS	EXAMPLE
EXIT	EXPLAIN	FACT	FOR	FORGET	FORMAL
FORWARD	FROM	GENDER	GENDER-NEUTRAL	GENDERNEUTRAL	GOAL
GREATER	GREATEREQUAL	HOUR	HOURS	IF	IN
INCLUDE	INFO	INFORMAL	INTEGER	IS	LESS
LESSEQUAL	LEVEL	LINE	LINK	LISTED	MATCH
MATCHES	MINUS	MINUTE	MINUTES	MOD	MONEY
MONTH	MONTHS	NEXT	NOT	NUMBER	NUMBER
NUMBERED	ONLY	OR	OR/WITH	ORDER	PARAGRAPH
PERSON	PERSON-THING	PERSONTHING	PLUS	PROCEDURE	PROMPT
PROVIDES	RANGE	REPEAT	REPORT	RULE	SAY
SECOND	SECONDS	SECTION	SESSION	SEX	STRING
STYLE	SUB-RULE	SUBRULE	SYSTEM	TEMPLATE	TEXT
THAN	THEN	THING	TIME	TIMES	TO
TRANSLATE	UNCERTAIN	UNIT	UNKNOWN	UNLISTED	UNNAMED
UNREPORTED	UNTIL	VERB	VERBS	WEEK	WEEKS
WHEN	WHILE	YEAR	YEARS		

Figure 1 - List of Keywords

A *descriptor* is a sequence of letters and symbols that does not include a keyword and is used to refer to a *constant, fact* or *text*. Descriptors may include any Unicode character

---

[13] Apart from being used in the occasional keywords (such as `AND/OR`, `AND/OR/WITH` and `PERSON-THING`), other places where special characters are significant are in facts where tilde (~) is optionally used to divide a fact name into a subject and predicate and where curly brackets ({}) are used to embed fact values or to specify *generic rules* and in the `VERBS` declaration where pipe (|) is used to specify alternative word forms. Single and double quotes are used to indicate string constants. Other characters such as decimal points, plus or minus signs and unit abbreviations have special meaning within numerical constant values.

[14] This list includes keywords that have been deprecated or which are used to parse DataLex declarations that are simply passed through to the DataLex system.

in UTF-8 encoding.[15] Sequences of blanks (spaces, tabs and newlines) containing more than one newline character are treated as a single newline. All other blank sequences are processed as though they were a single space. Upper and lower case is significant. Code may be formatted however you wish[16] but it is recommended that standard indenting is used to indicate control flow.[17] *Text* may optionally be enclosed in single or double quotes.

3.2 Escape Sequences

The special meaning of a keyword and of any individual character may be escaped by preceding it with a backslash ('\') character. Newlines can be included in text with the sequence `\n`. For example:

```
\"THIS \IS A \STRING CONTAINING KEYWORDS SURROUNDED \BY QUOTES\"
this text is in \{ curly brackets \}
this is a fact the contains a real tilde – \~
this is the first line of text\nthis the second\nand this is the third
```

The complete list of escape sequences is:

Escape Sequence	Meaning
`\`*`KEYWORD`*	the keyword as a plain word
`\n`	newline
`\\`	backslash
`\~`	tilde
`\"`	double quote
`\'`	single quote
`\{`	left curly bracket
`\}`	right curly bracket
`\<`	Left angle bracket[18]
`\>`	Right angle bracket
`\x`	any other character 'x'

Figure 2 – Character Escape Sequences

3.3 Comments

Comments are used to describe parts of the code and to indicate things that are not obvious, limitations of the code or things that are yet to be done. Comments have no

---

[15] Normally, non-ASCII Unicode characters are simply included in descriptors. Where such characters are syntactically significant such as single and double quotes and non-breaking spaces in non-*text* blocks, these are transliterated to ASCII form.

[16] *Text* blocks may use Markdown formatting. See § 8.2 at page 91.

[17] `yscript` can also be called with the `-p` flag which will "pretty-print" code in suggested standard format.

[18] In the original version of yscript, angle brackets were used for embedded facts. To support legacy code, facts that are enclosed in angle brackets and used or declared elsewhere in the code are still treated as embedded facts. Apart from dealing with any problems that arise from this special case, angle brackets should not need to be escaped.

effect on the meaning of the code itself or of what it does when executed or evaluated. yscript code is intended to be transparent and so use of comments should be used when they really have something to say rather than just repeating what anyone can already read. yscript supports line and multi-line commenting styles.

Multi-line comments are introduced with the characters /* and finish with */. Unlike most languages, comments nest so that you can comment out code which already contains comments. For example:

```
/* this is a comment */

/*
 *   so is /* this */
 */
```

You can also use C++ style single line // comments. For example

```
// this is single-line comment in C++ style
this is code // but this is a comment
```

3.4 Code Syntax and Structure

yscript code consists of a set of declarations of *facts* and *rules*.[19] Each of these types of declarations is dealt with in a chapter of their own below, but we have already seen examples of each type of declaration in the previous section.

Facts are dynamically typed and only need to be formally declared if their *type* is not implicit from their usage. The examples of fact declarations that have already been used relate to a special type of facts called - *named subjects* which are declared as in:

```
PERSON the claimant
THING the destination
PERSON-THING the party
```

`BOOLEAN` facts can be associated with the values `true` and `false`.[20] Other facts can have values such as numbers, dates, or strings.

Where necessary, these can be formally declared as in:

```
MONEY the purchase price
INTEGER the number of properties purchased
DATE  the date of purchase
STRING the name of the client
```

Rule declarations start with a rule type (generally just `RULE`), followed by an optional name and the keyword `PROVIDES`, and finally one or more *statements*. We have already

---

[19] As we will see later, facts and rules can also be grouped in *contexts* which are designed to build larger applications. See § 4.18 on page 58. There are also several other miscellaneous declaration types including: *defaults, rule order, includes* and *verbs*.

[20] As an alternative, `BOOLEAN` facts can take the constant values `yes` and `no` which have the same meaning as `true` and `false`.

considered examples of two types of statements: *assignment* (`ONLY IF`) and the conditional `IF-THEN` statement. An assignment statement can take several forms, but the simplest is the `ONLY IF` form where a (`BOOLEAN`) fact is assigned the truth value of a fact or *expression*.

For example:

```
RULE PROVIDES
a proposition requiring two conditions is true ONLY IF
    the first condition is met AND
    the second condition is met
```

In this example, the value of the *expression* "`the first condition is met AND the second condition is met`" is assigned to the fact "`a proposition requiring two conditions is true`".

The other example of a *statement* that we have seen is the `IF-THEN` statement which will only evaluate a statement if a condition is true. For example:

```
RULE PROVIDES
IF the condition is true THEN
    the conclusion is true
```

will execute the assignment statement - "`the conclusion is true`", only if the expression containing just the fact "`the condition is true`", is true.

3.5 Types and Constants

There are eight basic *types* of *facts* and *constants*: boolean, currency, dates, genders, integers, numbers and strings and times.[21]

Constants are used to refer to values that do not change[22] such as numbers, dates, and strings. Each constant has an associated type and will be recognised as such based on the format of a descriptor and where it appears in the code.

---

[21] There are also three composite types for *named subjects*: `PERSON`, `THING` and `PERSON-THING`.

[22] Two exceptions to the rule that constants do not change are the `DATE` *constant* – today, and the `INTEGER` *constant* - `random`. The former will always have the value of today's date and the latter is a random number.

The basic types and associated constant forms are:

Type	Example	Description
BOOLEAN	true	true or false.
NUMBER	1000.123	A number with optional decimal point and fractional component and optionally preceded by a plus or minus sign.
INTEGER	1,000	A positive or negative whole number with no decimal point or fractional component.
MONEY	$100.00	A monetary value consisting of a currency abbreviation, then an integer followed by an optional decimal point and two digits.
STRING	"xyz"	Anything starting and ending with a single or double quote.
DATE	1 January 2021	A date in any commonly used format including the shorthand today. By default, day/month ordering is day and then month.
TIME	10:23am	A time in any commonly used format including: now, midday, noon and midnight.
GENDER	female	Any preferred gender or unspecified.

Figure 3 - Types and Constants

BOOLEAN facts and constants have a value of true or false. An INTEGER is a positive or negative whole number which can be in the range -10^{18} to $+10^{18}$. A NUMBER is a double precision floating point number with approximately 10^{16} digits of accuracy and is presented with up to 6 digits after the decimal point. A STRING is any series of characters contained in double or single quotes.[23]

GENDER facts may refer to any string value. The value can also be UNSPECIFIED where a user does not wish to provide their gender or where the value is otherwise inappropriate or unnecessary.

A MONEY value is identical to a NUMBER except that if a fact or constant is not associated with some other currency *unit* when its value is first used, then the fact or constant

[23] Unicode quotes are also recognised. A string can contain single and double quotes. Strings can also contain embedded facts enclosed in {curly brackets}. See § 4.13 below at page 56.

will be associated with the unit `dollars`. This can be changed by specifying a different monetary *unit*.[24] A `MONEY` value cannot be associated with a non-currency unit.

Date Constants

The `DATE` type is used to refer to calendar dates. Most standard date formats can be used as constants in code and during execution. Examples of some of these are:

`5th March 2020`	`MARCH 5, 2020`
`5th day of March, 2020`	`Mar 5 2020`
`5 MAR 20`	`5/3/2020`
`05.03.2020`	`05:03:20`
`2020-05-03`	`today`

Figure 4 - Common Date Formats

The default output date format[25] is: `21st March, 2021`. The day/month order in yscript code assumes that the day comes before the month. The day/month ordering for user input when yscript code is executed is determined by the locale.

The supported range of years is from `AD 1` to `9999`[26]. Where a year has only two digits, and if it is less than `50` it will be taken to be a shorthand for `2000–2050`, and abbreviated years from `50` to `99` will be taken to be referring to `1951–1999`. To avoid confusion introduced by US style month-day ordering,[27] it is good practice to use a date format with a non-numerical month. It is also a good idea to avoid short form years and include full four-digit years. The word `today` is interpreted as a date equivalent to today's date.

Time Constants

The `TIME` type is used to refer to time. Like dates, most common formats are recognised. The default output format is *`hh.mm`* `am/pm`. This can be changed by specifying a different `STYLE`.[28] Times are assumed to be local by default. This can be changed by adding a time zone.

---

[24] This can be done on a per fact basis (see § 4.5 at page 45) or the default change be changed for all monetary units (see § 4.8 at page 49).

[25] This can be changed by specifying a new default date format. See § 4.8 at page 49.

[26] Years before the Common Era (`BCE` or `BC`) can be dealt with using the `calendar year` unit. See page 48.

[27] Apart from ISO dates, yscript code uses the European convention of assuming dates are in *day-month-year* format. The day-month order for dates input by an application end-user is determined by the current locale.

[28] See § Figure 22 – Time Styles at page 48.

For example:

10 PM	10.00p.m.	12 noon	2300 hours
22:00	10 o'clock	12 midnight	now
20:00:02	10.20.30 a.m.	noon	14:20 AEST[29]

Figure 5 - Common Time Formats

3.6 Number Formats

Numbers can be written in a variety of formats. All numbers can be positive or negative and may always start with an optional plus or minus sign. For non-integers, the decimal point can be written as a full-stop or as a comma and this may appear as the first thing in a number without being preceded by a zero. Digits to the left of a decimal point can be grouped into sets of 3 or 4 digits with commas, full-stops or spaces. Where this is ambiguous, the number will be interpreted as though a comma is a separator rather than a decimal point. Non-integers can also be expressed in exponential form.

Some examples:

123	the integer, one-hundred-an-twenty-three
−1,000	minus one thousand using US/UK style comma grouping
200 000 000	two hundred million using ISO style digit grouping
10.200.000	ten point two million using European grouping
+1020,0000	ten point two million using Japanese 4 digit grouping
10’200’000	ten point two million using Swiss style grouping
10.5	ten and a half in US/UK style
10,50	ten and a half in European / international format
2.000,12	two thousand point twelve in European format
3,300	three thousand, three hundred in UK/US format
3,3	three-point three in European format
1.200e+3	one thousand, two hundred in exponential form

Figure 6 - Number Formats

The output numbering style can be set using the `STYLE` sub-declaration on a per fact basis[30] or can be set generally with the `DEFAULT NUMBER STYLE` declaration.

3.7 Currencies and Units of Measurement

Facts and constants with a numerical value can be associated with a unit of measurement or an international currency.

---

[29] For a list of supported time zones see Figure 29 - Time Zones at page 110.

[30] See § 4.6 at page 46.

A reference to a numerical constant can include a unit by either indicating a unit type after the constant value or, in the case of currency amounts, by preceding the constant value with the currency type such as a dollar sign or other currency symbol. For example:

```
50 km          30°C          100 MPH
1 µl           20 t          30 N m
€ 200          1000VT        GBP 150
```

Figure 7 - Example Units

The specification of a unit may sometimes result in a change to the value of an associated constant. The reference to € 200 in the example above will have no effect other than indicating in the code that this is intended to be an amount in Euros. The *euro* is a *base unit* and so € 200 has a value of 200. The second example - 50km, is expressed in terms of the *kilometre* which is derived from the base unit of length – the *metre*. A kilometre is 1000 metres and so 50 kms will have the value – 50000. The final example - 1 µl, refers to one microlitre which is one-millionth of the base-unit - the litre and accordingly will have the value of 0.000001.

Metric Base Units

Metric values are made up of an optional metric prefix followed by a *metric base-unit*. The metric base-units[31] are as follows:

amp	A	atmosphere	atm	bar	bar
becquerel	Bq	bit	b	byte	B
candela	cd	coulomb	C	dalton	Da
degrees Celsius	°C	electronvolt	eV	farad	F
gram[32]	g	gray	Gy	hectare	ha
henry	H	hertz	Hz	joule	J
katal	kat	kelvin	K	litre	l
lumen	lm	lux	lx	metre	m
mole	mol	newton	N	ohm	Ω
pascal	Pa	second	s	siemens	S
sievert	Sv	tesla	T	ton	ton
tonne	t	volt	V	watt	W
weber	Wb				

Figure 8 –Metric Base Units

[31] This is a list of all base units that can take a metric prefix and includes the seven SI base units, the official twenty named dimensioned SI derived units, the dimensioned non-SI units accepted for use with SI units and the unofficial non-SI units – bits, bytes, and tons.

[32] The SI unit of mass is the *kilogram* which is one of the seven SI base units. Whilst the *kilo* is technically part of the base SI unit name, yscript regards the base-unit for mass as the *gram*.

Non-Metric Base Units

Non-metric unit types such as non-dimensioned SI units, Imperial and US based measures are also supported. These include:

acre	acre	ångström	Å	astronomical unit	au
Bel	bel	British thermal unit	BTU	calorie	Cal
chain	ch	cup	cup	day	day
decibel	dB	degree	°	degree Fahrenheit	°F
fathom	ftm	fluid ounce	fl oz	foot	ft
furlong	fur	gallon	gal	gravity	*g*
hour	hr	inch	″	inch of mercury	inHg
knot	kn	mile	mi	minute	'
month	Mth	nautical mile	nmi	neper	Np
ounce	oz	percent	%	pint	pt
pound	lb	quart	qt	radian	rad
second	"	steradian	sr	stone	st
tablespoon	tbsp	torr	Torr	week	wk
yard	yd	year	yr		

Figure 9 – US, Imperial and other Units

Referring to a Unit

A unit can be referred to by either its full name or by its abbreviation.[33] Long names may be in plural or singular form. The following sets of examples all refer to the same unit and will result in a constant having the same value:

```
€ 200, 200 euros, 200 EUR, 200
10 ohms, 10 Ω, 10
90 degrees, 90°, 90
```

Metric Prefixes

Metric base units can be combined with a *metric prefix* to indicate multiple and submultiple values. Any metric base unit can be combined with the following prefixes:

[33] There are also some automatic mappings for commonly used alternative forms of both the long and abbreviated unit names. See Figure 27 - Unit Synonyms at page 105.

yotta	Y	10^{24}	1 000 000 000 000 000 000 000 000
yetta	Z	10^{21}	1 000 000 000 000 000 000 000
exa	E	10^{18}	1 000 000 000 000 000 000
peta	P	10^{15}	1 000 000 000 000 000
tera	T	10^{12}	1 000 000 000 000
giga	G	10^{9}	1 000 000 000
mega	M	10^{6}	1 000 000
kilo	k	10^{3}	1 000
milli	m	10^{-3}	0.001
micro	µ	10^{-6}	0.000 001
nano	n	10^{-9}	0.000 000 001
pico	p	10^{-12}	0.000 000 000 001
femto	f	10^{-15}	0.000 000 000 000 001
atto	a	10^{-18}	0.000 000 000 000 000 001
zepto	z	10^{-21}	0.000 000 000 000 000 000 001
yocto	y	10^{-24}	0.000 000 000 000 000 000 000 001

Figure 10 - Metric Prefixes

The following examples each have the same meaning:

```
1 tonne, 1 t, 1 megagram, 1000 kg
50 km, 50 kilometres, 50000 metres
```

Note that for metric values, it is possible to refer to the same value in different ways. A tonne is one million grams and hence the use of the metric prefix "mega" (which means million). Similarly, a kilogram (abbreviated to kg in second example) refers to 1000 grams (the "kilo" prefix meaning 1000). One thousand kilograms is a million grams or one tonne. Similarly, a kilometre is 1000 metres and so 50km is 50000 metres.

Base units can be combined to produce derived units. Only specific derived units are recognised.

For example:

```
9.81 m/s²
30 miles per gallon
22 m²·kg·s⁻²·K⁻¹·mol⁻¹
```

Composite Units

Some units may optionally be constructed from components. The most important of these is elapsed time which can be expressed in hours, minutes, and seconds. The other composite units are *feet and inches* and *pounds and ounces*.

For example:

```
2 hours 10 minutes 3 seconds          10 lbs 3 ounces
10 minutes                            10 feet
10 minutes 3.3 seconds                6' 6"
```

Figure 11 - Composite Units

Currency Units

Currency units are indicated by preceding a numerical value with a currency abbreviation or following a number with a currency description. Currencies can be generic or currency specific. Generic currencies are not necessarily associated with any country or economic group. For example:

Long Description	Abbreviation	Examples
dollars	$	$100, $10,230.00, 2000 dollars
pounds	£	£15, £22.50, 100 pounds
euros	€	€ 75, € 150.00,50, 3.000 euros
yen	¥	¥1000, ¥10,0000, 500 yen

Figure 12 - Generic Currencies

All currencies, also have a country or economic group specific unit. This is indicated by three-letter ISO-4217 currency code. Some currencies also use unique abbreviations that map to these codes. For example:

```
USD 20,000, US$20,000, USD$20,000, 20000 USD
AUD 5,000, A$5000, AUD$5000, 5000 AUD
GBP 3000, GBP£3,000, 3,000 GBP
EUR 75, 2.000,12 EUR
JPY 1000, JP¥10,0000, 500 JPY
```

3.8 Expressions

An *expression* consists of fact or constant references, connected by *operators*. Expressions are used as conditions in *statements* (such as `IF`, `WHILE` and `REPEAT`) and for building complex values to be assigned to facts. *Binary operators* describe the relationship between two facts or perform an operation that depends on two fact values. Expressions can be *logical* (combining truth values), *relational* (comparing two values) or *arithmetic* (mathematical). Unary operators operate over a single fact value and can appear before a fact or constant (*pre-unary*) or after a fact or constant (*post-unary).* Unary operators bind more tightly (that is, have higher precedence) than binary operators. The order of precedence for binary operators is (highest to lowest):

```
TIMES DIVIDED MOD
PLUS MINUS
EQUALS GREATER GREATEREQUAL LESS LESSEQUAL NOTEQUAL MATCHES IN
AND/OR AND/OR/WITH
AND AND/WITH
OR OR/WITH
```

Figure 13 - Operator Precedence

The order of evaluation of an expression can be changed by use of BEGIN–END pairs (which work like brackets in ordinary mathematical expressions). For example, the mathematical expression (2 + 3) × 4 could be written as:

```
BEGIN 2 PLUS 3 END TIMES 4
```

Logical Operators

Logical operators are used to combine propositions to form more complex propositions. The available logical operators are:

Logical Operator	Meaning
AND	logical conditional AND
OR	logical conditional OR
AND/OR	logical conditional OR with high precedence
AND/WITH	logical non-conditional AND
OR/WITH	logical non-conditional OR
AND/OR/WITH	logical non-conditional OR with high precedence

Figure 14 - Logical Operators

Conditional operators operate left-to-right but will stop when a result is known. Non-conditional operators evaluate **all** arguments from left to right and are only used in special circumstances where you want to evaluate all facts in an expression regardless of whether they effect the logical result.

Some operators are semantically equivalent and only differ in respect of their precedence. Normally, logical operators follow the same precedence found in most programming languages, such that (for example) AND will bind more tightly than OR. Like in mathematical expressions, this can be altered with BEGIN and END pairs as discussed above. It is often however more aesthetic to use the AND/OR operator which works exactly like the normal conditional OR operator but with a binding strength above that of AND.[34]

Relative Operators

Relative operators compare two values for equality or relativity. The EQUALS and NOTEQUALS operators will work with expressions returning any type of value. The IN-operator tests if the first string is a sub-string of the second string and only works for expressions returning type STRING. All other relative operators are only legal for values with a type that has a continuous range (that is, DATE, INTEGER, MONEY, and NUMBER). Relative operators may be preceded by the keyword IS and then optionally followed

[34] When using conditional logical operators, the slightly less aesthetic AND/OR/WITH can be used for the same purpose.

by the keywords `THAN` or `TO` where this makes the code read in a more English-like way.

The following relative operators are available:

Relative Operator	Meaning
`EQUAL` or `EQUALS` or `EQUAL TO` or `EQUALS TO`	Test for equality
`NOT EQUAL` or `NOT EQUALS` or `NOT EQUAL TO` or `NOT EQUALS TO`	Test for in-equality
`IN`	Test if the first string is a sub-string of the second
`LESS` or `LESS THAN` or `IS LESS THAN`	Test if the first expression is less than the second expression
`GREATER` or `GREATER THAN` or `IS GREATER THAN`	Test if the first expression is greater than the second expression
`LESSEQUAL` or `LESSEQUAL THAN` or `IS LESSEQUAL TO`	Test if the first expression is less than or equal to the second expression
`GREATEREQUAL` or `GREATEREQUAL TO` or `IS GREATEREQUAL TO`	Test if the first expression is greater than or equal to the second expression

Figure 15 - Relative Operators

Note that comparing facts of type `NUMBER` for equality or inequality will always be inexact to allow for rounding errors that may have been introduced by calculation.

Some examples:

```
the value of the property IS LESS THAN $1,000,000

the number of people IS GREATER THAN 5

the type of game IS NOT EQUAL TO "a lottery"

"Neptune" IN the list of planets
the value of the prize IS GREATER THAN $100 AND the value of the prize IS
LESS THAN $1,000
```

Arithmetic Operators

Arithmetic operators normally perform mathematical calculations on numbers and dates. The `MATCHES` operator returns a number between 0 and 100 indicating the extent to which two strings are similar. This is useful if you are attempting to deal with free form user responses. Whilst not strictly arithmetic, the `PLUS` operator will also concatenate two strings.

Arithmetical Operator	Meaning
TIMES	Multiply two numbers.
DIVIDED or DIVIDED BY	Divide the first number by the second number.
PLUS	Add two numbers together or concatenate two strings.
MINUS	Subtract the second number from the first number.
MOD	Return remainder after dividing first number by second number.
MATCHES	Return a percentage fuzzy match for two strings.

Figure 16 - Arithmetic Operators

Some examples:

```
1 PLUS 1
BEGIN 1 PLUS 2 END TIMES 3
"the cat" PLUS "in the hat"
the response MATCHES "something we're looking for" GREATER THAN 50
```

Unary Operators

Unary operators have a single operand which may appear before or after the operator. The following unary operators are available:

Pre-Unary Operator	Meaning
UNKNOWN	Test if a fact is unknown or unknowable.
NOT	Negate the truth value of the following expression.
DAY	Extract the (integral) day from the following date.
HOUR	Extract the hours value from the following time.
MINUTE	Extract the minutes value from the following time.
MONTH	Extract the (integral) month from the following date.
SECOND	Extract the seconds value from the following time.
YEAR	Extract a year from a date.

Figure 17 – Pre-Unary Operators

Post-Unary Operator	Meaning
SECOND or SECONDS	The previous expression is in seconds.
MINUTE or MINUTES	The previous expression is in minutes.
HOUR or HOURS	The previous expression is in hours.
DAY or DAYS	The previous expression is in days.
WEEK or WEEKS	The previous expression is in weeks.
MONTH or MONTHS	The previous expression is in months.
YEAR or YEARS	The previous expression is in years.

Figure 18 – Post-Unary Operators

The NOT operator negates the truth value of the expression that follows. It is not often strictly necessary, as usually you can write boolean fact in its negative form. For example, a reference to a fact in the form: `section 123 does not apply` is equivalent to `NOT section 123 applies`.

The UNKNOWN operator checks to see whether the value of a fact is known without attempting to find a value for it. All other unary operators relate to date and time arithmetic. [35]

Date Arithmetic and Comparison

Dates can be compared in expressions using regular equality and relational operators. They may only be compared with other facts and constants of type DATE. For example:

```
the start of Spring EQUALS 1 September 2020
the date something happened IS LESS THAN the date
    something else happened
```

The components of a date can be extracted using the pre-unary operators: DAY, MONTH and YEAR. The result in an integral (INTEGER) value. For example:

```
the month that Spring starts IS MONTH the start of Spring
the year it happened IS YEAR the date something happened
```

The PLUS and MINUS expression operators can be used to perform date arithmetic. Days can be simply added to or subtracted from a date. It is also possible to add and subtract weeks, months, and years by using the `week`, `month`, and `years` units. For example:

[35] This is discussed further in the next section. Note that for constant values the effect of using the post-unary keywords is equivalent to the use of an equivalent unit.

```
ninety days after this date IS this date PLUS 90
ninety days after this date IS this date PLUS 90 days
six months before this date IS this date MINUS 6 months
```

When adding or subtracting months or years to or from a date and where the day falls on the last day of the month, the resulting date will fall on the last day of the month for the new date. For example:

```
30 June 2020 PLUS 9 months EQUALS 31 March 2021
29 February 2020 PLUS 1 year EQUALS 28 February 2021
```

It is permissible to add or subtract days, months, and years to a date at the same time. For example:

```
the final date IS the signing date PLUS 1 month PLUS 1 day
```

or a longer example:

```
DATE my date
INTEGER my day
INTEGER my month
INTEGER my year

RULE PROVIDES
my date IS 1 January 1 PLUS my day DAYS[36] MINUS 1 DAY
        PLUS my month MONTHS MINUS 1 MONTH
                PLUS my year YEARS MINUS 1 YEAR
```

Time Arithmetic and Comparison

Like dates, fact values and constants of type `TIME` can also be compared and subject to arithmetic. Times are stored as the number of seconds since midnight in local time.

For example:

```
TIME the time

RULE PROVIDES
IF the time IS LESS THAN 12 noon THEN
    it is morning
ELSE IF the time IS LESS THAN 6 pm THEN
    it is afternoon
ELSE
    it is evening
```

Seconds can be simply added to or subtracted from times and they will continue to display correctly regardless of whether a time becomes negative or is greater than 24 hours. Times can be added and subtracted to `DATES`. For example:

[36] The units `DAY`, `WEEK`, `MONTH` and `YEAR` are also reserved words when capitalised. These have the same effect as using the corresponding units but can be applied to a fact as well as being part of a constant reference.

```
RULE PROVIDES
ASSERT the start date IS 31 December 2021
ASSERT the start time IS 11pm
ASSERT the finish time IS the start time PLUS 3 hours
ASSERT the finish date IS the start date PLUS the finish time
```

will produce:

```
The start date is 31st December, 2021. The start time is
11pm. The finish time is 2am because the start time is 11pm.
The finish date is 1st January, 2022 because the start date
is 31st December, 2021 and the finish time is 2am.
```

Times from different time zones[37] may be combined and will be automatically converted as necessary. For example:

```
TIME the time in Sydney
    UNIT AEST

RULE PROVIDES
the time in Sydney IS 5.16am UTC-7
```

will produce:

```
The time in Sydney is 10.16pm AEST.
```

Expressions Involving Numerical Units

The value of all facts and constants is stored in *base-unit* format.[38] For example, the value of `1000 m` and `1 km` will both be `1000`. *Composite units*[39] are also stored using the unit associated with the base form. Elapsed time expressed in `hours` and/or `minutes` is stored in `seconds`, `feet` are stored as `inches` and `pounds` are stored as `ounces`. For example, the constants - `2 hours` and `120 minutes`, will both have the value `7200`.

The effect of this is that generally units forming part of a base-unit or base form family can be compared or assigned to each other. These units can also be added together or subtracted from each other. For example:

```
RULE PROVIDES
ASSERT the length IS 750 m PLUS 1.25 km
ASSERT the other length IS 50 feet PLUS 10 feet 6 inches
```

will produce:

```
The length is 2 km. The other length is 60 ft 6 in.
```

Units from different systems of measurement are converted as necessary. So, for example:

---

[37] See Figure 29 - Time Zones on page 110 for a list of supported time zones.

[38] See § 3.7 above at page 29.

[39] See Composite Units at page 32.

```
NUMBER the imperial length
    UNIT yards

RULE PROVIDES
ASSERT the length IS 750 m PLUS 60 ft 6 in
ASSERT the imperial length IS the length
```

will produce:

```
The imperial length is 840.37664 yd because the length is 768.4404 m.
```

Finally, values associated with units can be multiplied and divided. For example:

```
RULE PROVIDES
ASSERT x IS 10 metres TIMES 10 metres
ASSERT y IS 10 feet TIMES 24 inches
```

will result in:

```
X is 100 m². Y is 20 sq ft.
```

Chapter 4: Facts

Facts are used to hold a value during execution and are referred to by a descriptor that is not a *constant* and that is not *text*. Generally, fact names should be written in lower case unless referring to a proper noun or an abbreviation. Fact names can contain any Unicode character in UTF-8 encoding. Case is significant. Like other descriptors, fact names can contain multiple spaces or tabs and can extend across more than one line. Blank space is treated as a single space.

4.1 Fact Declarations

Facts are always associated with a single *type*. Most facts will be of type `BOOLEAN`, that is they are propositional statements representing either a premise or a conclusion. Other valid fact types are `DATE`, `GENDER`, `INTEGER`, `MONEY`, `NUMBER`, `STRING` and `TIME`.[40] It is not generally necessary[41] to formally declare a fact unless its type is not clear from the way it is used or if you want to modify the way the name of the fact is used when communicating with the user. Fact declarations consist of an optional *qualifier* followed by a *type,* optionally followed by the keyword `FACT` and then the name of the fact. If only the keyword `FACT` appears then the fact is assumed to be `BOOLEAN`.

The formal syntax[42] for a fact declaration header is: [43]

```
[ GOAL ] [ SYSTEM | UNREPORTED ]
    [ BOOLEAN | GENDER | INTEGER | NUMBER | STRING | TIME ] [FACT]
        fact-name [ FROM context ]
```

The qualifiers `SYSTEM` and `UNREPORTED` serve to supress the reporting of fact values for facts that are being used in a mechanical way. The use of *contexts* allows for facts to have separate namespaces and is discussed at the end of this chapter.

Where the `GOAL` keyword is specified, this procedure will be marked as a goal rule.

Some examples of simple fact declarations include:

```
INTEGER the number of children of the testator
MONEY the value of the prize
STRING the name of the event
FACT the testator is survived by a legal or de facto spouse
```

---

[40] Facts may also refer to a *named subject* which is a composite fact discussed below. See § 4.12 at page 53.

[41] The other main reason for formally declaring facts is so that they can be shared between different *contexts* as discussed in § 4.18 at page 58.

[42] The convention used in this manual is that square brackets [] indicate that the contents are *optional*, curly brackets {} indicate *one or more occurrences* of their content and pipe (|) indicates an alternative. Non-literals are shown in italics. *boolean-fact* {`AND` *boolean-fact*} thus means a boolean fact (for example, `section 23 applies`) optionally followed by zero or more occurrences of the keyword `AND` followed by another boolean fact (for example, `section 23 applies AND section 24 applies`).

[43] Fact declarations can also be followed by a prompt, an alias, attachments, translations, explanations, and a range. Where a fact is part of an object, it can also contain *statements*. See § 6.8 at page 78.

Fact declarations should appear outside of rules and procedures. Facts cannot be declared as being local to a rule or procedure and are accessible from all rules in the context.[44]

4.2 Propositions

yscript code generally involves manipulating facts in propositional form. *Propositional logic* is a branch of logic that is concerned with combining statements which are true or false (*propositions*) to form or decompose more complex propositions. Unlike other forms of logic, propositional logic is not concerned with how the propositions themselves work. Propositions are regarded as indivisible units of expression. In yscript, propositions are represented as `BOOLEAN` facts.

The name of a fact operates much like a variable name in a conventional programming language. The thing that is different in yscript, however, is that the fact names are also used to build elements of English dialogs which are used to interact with application users. This is important because it means that information can be treated symbolically but with no requirement to provide a separate textual representation for each symbol. Whilst the composition of propositional (boolean) fact names is irrelevant to the symbolic operation of an application (that is, what conclusions are formed and for which reasons) it is very important from the point of view of communicating with an end-user.

yscript uses a light-weight natural language parser[45] to transform simple English language propositions into their positive or negative form and to turn them into questions when necessary. A proposition is regarded as a subject followed by a predicate. The subject is the person or thing that the proposition is about. The predicate modifies the subject. A predicate implies some action and should always start with a simple or compound verb and may optionally be followed by an object or modifying action phrase:

```
subject compound-or-simple-verb [ object ]
```

The most critical step for yscript in processing a proposition is to divide the proposition into a subject and a predicate. The current approach applies a set of heuristic rules that identify different parts of speech based around generally used word forms supplemented by a dictionary of commonly used verbs expressed in their present, plural, past and past-participle forms. The algorithm will prefer auxiliary verbs over all other verbs in a proposition. Where no auxiliary verb is present, it will look for a known verb in past tense or plural form, then for words that appear to be verbs (that is, which appear to be plurals or to be in past tense or which are preceded

---

[44] Facts may also be shared between contexts. See § 4.18 at page 58.

[45] If you are interested in how yscript parses propositions, the standard distribution includes a utility called `trans` which interactively prompts for a proposition and shows how and why the proposition is split into a subject, verb and object as well as displaying resultant translations.

by an adverb) and then a known verb in present tense. The algorithm considers parts of speech that appear to be nouns and several other factors.

The list of auxiliary verbs includes:

am	are	can	could	did	do	does
had	has	have	is	may	might	must
ought	shall	should	was	were	will	would

Figure 19 - List of Auxiliary Verbs

The practical upshot of all of this is that when choosing the name for a `BOOLEAN` fact, you should describe it as a proposition, that is it should start with a subject, then a verb (in the positive or negative and in compound or non-compound form) and, optionally an object. For example:

```
yscript is a computer language
propositional logic studies ways of joining statements
the rule applies
```

Provided that fact names appear in this format, yscript will normally be able to affect sensible prompts and translations that can be used during user sessions. For the above examples, the following automatic prompts and translations would be used:

```
Is yscript a computer language?
yscript is a computer language.
yscript is not a computer language.

Does propositional logic study ways of joining propositions?
Propositional logic studies ways of joining propositions.
Propositional logic does not study ways of joining propositions.

Does the rule apply?
The rule applies.
The rule does not apply.
```

Internally, all boolean fact names are stored in positive form. Contractions are expanded, and some grammatical forms simplified.

The following sets of alternatives all refer to the same fact:

```
the rule applies
the rule does not apply
the rule does apply
the rule doesn't apply

the person went overseas
the person did not go overseas
the person didn't go overseas
the person did go overseas

the defendant did twenty years in prison
the defendant did not do twenty years in prison

the testator had no children
the testator had children
the testator did not have children
the testator didn't have children
```

4.3 Dealing with Grammatical Errors

The natural language parsing techniques used by yscript to transpose and manipulate propositions are heuristic in nature and occasionally mistakes will be made.[46] When this happens, the first step is to try to rephrase the proposition in a more straightforward way. Provisos and exceptions are probably best expressed as separate facts. Passive statements should be expressed in active form. The parser will always prefer an auxiliary verb[47] over any other known verb or word that appears to be a verb. Inserting an auxiliary verb like "does" or "did" will often resolve the problem. If this is superfluous, it will serve just to identify a verb and will not appear in consultations.

If these techniques do not work, then there are three other approaches. The simplest is to divide the proposition manually by inserting a tilde (~) between the proposition subject and predicate (that is, before the compound or simple verb). This will become part of the name of the fact and must be included the first time a fact is used. For example:

```
the client~thinks he has a good case
the cost of the appeals~exceeded expectations
```

The parser currently contains a dictionary of around 6,000 commonly used verbs. For other verbs (particularly irregular verbs), it may either not recognise the word as a verb or transpose it incorrectly to and from its plural, past tense, or past participle form. In such cases, verbs can be added to the dictionary using the `VERBS` declaration.[48] The syntax is:

```
VERB|VERBS { present|plural|past|past-participle }
```

For example:

```
VERBS catch|catches|caught|caught break|breaks|broke|broken
```

Where none of these approaches works, the final resort is to declare specific prompts and translations for the fact.[49]

---

[46] Apart from the `trans` utility mentioned previously, the `yscript` interpreter has a `-t` flag which will display the translations for all facts in an application.

[47] See Figure 19 at page 43 for a list.

[48] Previously, yscript was much more reliant upon verbs being declared in code. The need for the `VERBS` declaration is now much reduced to the point that it is nearly always unnecessary. In older code, you may also see that verb forms were presented in condensed form and separated by tildes. This usage is deprecated and probably best forgotten as a bad idea.

[49] See § 4.11 at page 51 below.

4.4 Non-Boolean Facts

Non-boolean facts are introduced in one of two ways: by use or formally. If the first use of a fact requires that it have a type other than boolean, the fact is automatically associated with this type.

For non-boolean facts, choose a name that can be followed by an "is" or an "are" (where the subject is plural) and a value. For example, the fact declarations:

```
MONEY the purchase price
MONEY the proceeds of the raffle
DATE the date of the purchase
STRING the names of the children
```

will result in the following prompts and translations:

```
What is the purchase price?
The purchase price is $1,000,000.

What are the proceeds of the raffle?
The proceeds of the raffle are $200.

What is the date of purchase?
The date of purchase is 28th February 2020.

What are the names of the children?
The names of the children are Moe, Larry and Curly
```

4.5 Currencies and Units of Measure

Numerical facts may also be associated with a *unit* of measure and a *dimension*. The units associated with facts can be declared explicitly or can be inferred from assignments and expression usage of constants or other facts. For example, the following statements or expressions will result in the fact having the specified type, unit, and dimension:

Statement	Type	Unit	Dimension
`the weight IS 10 kg`	`INTEGER`	`grams`	`kilograms`
`the force IS LESS THAN 1.0 N`	`NUMBER`	`newtons`	`newtons`
`the temperature EQUALS 100 °C`	`INTEGER`	`degrees Celsius`	`degrees Celsius`
`the amount GREATER THAN $100.00`	`NUMBER`	`dollars`	`dollars`

A unit and dimension association can also be indicated as part of a formal fact declaration. This is done using the `UNIT` sub-declaration. The syntax is:

```
UNIT unit-name
```

The *unit-name* may be either the full name or the abbreviation for a unit. For example, the formal declarations for the facts from the previous examples would be:

```
INTEGER the weight UNIT kilograms
NUMBER the force UNIT newtons
INTEGER the temperature UNIT degrees Celsius
NUMBER the amount UNIT dollars
```

The *dimension* of the unit is indicated by a metric prefix to the unit name. It is used as the default when asking the end-user for any values during consultations. It will also sometimes affect the way that a value is displayed.[50]

Where a fact is assigned the value of an expression with a different unit type, the value and unit type will be converted where possible. This includes units with different dimensions as well as units from different systems of measurement.

For example:

```
INTEGER the temperature UNIT degrees Celsius
NUMBER the wind speed UNIT km/h
INTEGER the temperature in Fahrenheit UNIT °F
INTEGER the wind speed in miles per hour UNIT mph

RULE PROVIDES
ASSERT the temperature IS 25°C
ASSERT the wind speed IS 10 km/h
ASSERT the temperature in Fahrenheit IS the temperature
ASSERT the wind speed in miles per hour IS the wind speed
```

will produce:

```
The temperature is 25°C. The wind speed is 10 km/h. The
temperature in Fahrenheit is 77°F because the temperature
is 25°C. The wind speed in miles per hour is 6 mph because
the wind speed is 10 km/h.
```

4.6 Time Zones

Times are assumed to be local by default. This can be changed by specifying the time `UNIT` to a particular time zone. [51] For example:

```
TIME the time in New York City
UNIT EST
```

Note that time zones are not completely standardised and can be ambiguous. The time `UNIT` can also be set to an IANA country code.[52]

---

[50] For example, where the unit `millimetres` or `mm` is used, lengths will be displayed as millimetres for values less than 10m. Similarly, values using the unit/dimension `kilometres` will be displayed in kilometres even when greater than 1000.

[51] See Figure 29 - Time Zones at page 110 for a complete list of available time zones.

[52] See Figure 30 - Time Zone Country Names at page 112 for a list.

For example:

```
TIME the time in London
UNIT Europe/London
```

Apart from being unique, using IANA names has the advantage that it will select a time zone that accounts for any daylight savings changes. Where there is no available time zone, this will be set to local time.

The overall default time zone can be changed with the `DEFAULT TIME UNIT` declaration.[53]

4.7 Styles

Facts can also be associated with *styles* which can be used to specify how the value of a fact should be displayed. The style will normally be automatically selected based on the fact type and associated unit type. This can be changed by using the `STYLE` sub-declaration.

The syntax is:

```
STYLE style-description
```

For example:

```
NUMBER the length of the boat
UNIT metres
STYLE European
```

Numerical Styles

The valid numerical styles which are applicable to fact declarations of types: `NUMBER`, `INTEGER` and `MONEY` are:

default	use commas to separate 1,000s and a dot for the decimal point
european	use dots to separate 1.000s and a comma for the decimal point
si	use SI format – spaces to separate 1 000s and a dot for the decimal point
eastern-european	use SI format, but use a comma for the decimal point
japanese	use commas to separate groups of 1,0000s and dot for the decimal point
swiss	use quotes to separate groups of 3 digits and a comma for decimal point
roman	use roman numerals

Figure 20 – Numerical Styles

[53] See § 4.8 at page 49.

Calendar Years

Calendar years can be declared as the `INTEGER` unit – `calendaryear`. When output, calendar years will be expressed to be `BC / BCE` and `AD / CE` depending upon the year value. The output presentation can be changed with the following styles:

AD or BC	precede years after zero with AD and follow years before 0 with BC
A.D. or B.C.	precede years after zero with A.D. and follow years before 0 with B.C.
BCE or CE	follow years after zero with CE and follow years before 0 with BCE
B.C.E. or C.E.	precede years after zero with C.E. and follow years before 0 with B.C.E.

Figure 21 - Calendar Year Styles

Where the style is in lowercase, the era will only be included for non-contemporary dates (which are defined as prior to AD 1000 and after 2200). For example:

```
AD 300
300 CE
2000 B.C.E
2021
```

Time Style Formats

The `TIME` style format consists of a string with the following elements:

`HH`	zero padded hours (`00`–`23`)
`hh`	hour (`1`–`12`)
`MM`	zero padded minutes (`00`–`59`)
`mm`	minutes (`1`–`59`)
`SS`	Zero padded minutes (`00`–`59`)
`ss`	Seconds (`1`–`59`)
`am` or `pm`	`am`, `pm`, `noon` or `midnight`
`AM` or `PM`	`AM`, `PM`, `NOON` or `MIDNIGHT`
`A.M.` or `P.M.`	`A.M.`, `P.M.`, `NOON` or `MIDNIGHT`
`TZ or tz`	`international time zone`

Figure 22 – Time Styles

For example:

Format	**Example Output**
`hh.mm.ssam/pm`	`10.23pm`
`HH:mm:ss`	`22:23`
`HHmm:ss hours`	`2223:15 hours`
`HH:MM tz`	`14:25 AEST`
`hh.mm am/pm TZ`	`2.15 am UTC+11`

Date Style Formats

The output date format can be changed by specifying a `DATE` style which is a string containing the following elements:

`day` or `dd`	the day (`1`–`31`)
`DD`	zero padded day (`01`–`31`)
`month`	the month (`January`–`December`)
`MONTH`	the month in capitals
`MMM`	abbreviated month in capitals
`mmm`	abbreviated month in mixed case
`MM`	zero padded month as number
`mm`	the month as a number (`1`–`12`)
`th`	day suffix (`st`, `nd` or `th`)
`TH`	day suffix in capitals
`year` or `yyyy`	the year (`0`–`9999`)
`yy`	the year (`00`–`99`)

Figure 23 - Date Styles

For example, setting the date style to "`the dayth of month, year`" will result in dates displaying as in "`the 20th of December, 2021`" and "`mm/dd/yyyy`" will display the first of March 2021 as "`3/1/2021`".

There are no styles for fact values of type `BOOLEAN`, `GENDER` and `STRING`.

4.8 Default Styles, Numbering and Units

The default output style and numbering for all facts of a particular type can be changed using the `DEFAULT` declaration. The available default style declarations are:

Declaration Syntax	**Example**
`DEFAULT NUMBER`[54] `STYLE` *`number-style`*[55]	`DEFAULT NUMBER STYLE swiss`
`DEFAULT MONEY UNIT` *`currency-unit`*[56]	`DEFAULT MONEY UNIT euro`
`DEFAULT YEAR STYLE year-style`	`DEFAULT YEAR STYLE B.C.E.`
`DEFAULT DATE STYLE` *`date-format-string`*	`DEFAULT DATE STYLE dd/mm/yyyy`
`DEFAULT TIME STYLE` *`time-style`*[57]	`DEFAULT TIME STYLE hh:mm:ss`
`DEFAULT TIME UNIT time-zone`	`DEFAULT TIME UNIT Asia/Tokyo`

Figure 24 - Default Styles, Numbering and Units

---

[54] This will change the formatting for all output numerical values, that is it will apply to types: `NUMBER`, `INTEGER` and `MONEY`.

[55] See Figure 20 – Numerical Styles on page 47 for a list of valid numerical styles.

[56] Currency units are discussed on page 33.

[57] See Figure 22 – Time Styles on page 48.

4.9 Certainty

Before a fact is assigned a value or one is provided by the user, the fact value is `unknown`. If there is no way of otherwise determining a value for a fact and the user says that the value of the fact is `uncertain`, the fact is tagged as being `unknowable` (or `unspecified`). Once a fact becomes unknowable, the system will not attempt to use rules to find a value and it will not re-ask the user for a value.

The certainty of a fact can be tested with the `UNKNOWN` pre-unary operator. This will return `true` if the fact value is `uncertain` (that is, not value has yet to be determined) or `unknowable` (that is, it is not possible to know a value for this fact).

For example:

```
IF UNKNOWN the defendant is bankrupt THEN
    it is possible that the defendant is not solvent
```

4.10 Ranges

Where there is a need to further limit the range of acceptable responses from the user then a `FACT` declaration can include a `RANGE` restriction.

The syntax is:

```
RANGE arithmetic-term TO arithmetic-term
     { RANGE arithmetic-term TO arithmetic-term } |

RANGE arithmetic-term { RANGE arithmetic-term }
```

Continuous Ranges

The first format is used to restrict user responses to a continuous range of values. It is unusual, but you can also specify a split range of values by using more than one `RANGE` declaration. Some examples:

```
MONEY the maximum amount of money payable as a separate prize
    RANGE $0 TO the total value of all of the prizes

DATE the school holidays
    RANGE 28 March 2020 TO 11 April 2020
    RANGE 19 June 2020 TO 30 June 2020
    RANGE 20 September 2020 TO 3 October 2020
    RANGE 15 December 2020 TO 28 January 2021

DATE the notice period
    RANGE today TO today PLUS 6 MONTHS
```

Discrete Ranges

Ranges can also be used to specify that a user must select from a specific set of values. This can be used with facts of any type, but most typically this is used for facts of type `STRING`.

For example:

```
STRING the name of the security agency
     RANGE "ASIO" RANGE "ASD" RANGE "ASIS" RANGE "DIO"
     RANGE "AFP" RANGE "AIC"
```

Where a discrete range of responses is specified, the calling environment should present a question in relation to the associated fact in multiple choice format.

For example:

```
1) What is the name of the security agency?
   (a) ASIO
   (b) ASD
   (c) ASIS
   (d) DIO
   (e) AFP
   (f) AIC

Please select?
**
```

Range Errors

If the user attempts to provide a value for a fact that is out of range, an error message will be issued[58] and the question re-asked. For example:

```
MONEY the total value of all prizes
MONEY the maximum amount of money payable as a separate prize
    RANGE 0 TO the total value of all prizes

RULE PROVIDES
DETERMINE the maximum amount of money payable as a separate prize
```

will result in the consultation:

```
   1) What is the total value of all prizes?
      ** 500

   2) What is the maximum amount of money payable as a separate prize?
      ** 1000

      The value must be between $0 to $500.
```

4.11 Prompts and Translations

One of the features of yscript is its ability to automatically transpose propositions into questions, translations, and explanations so that you do not need to maintain a separate set of textual material for an otherwise symbolic form of representation. Nevertheless, there are occasions when you will want the representation as contained

[58] This message can be customised using the `RANGE` argument to the `TRANSLATE` declaration covered in the next section.

in facts in the code to be different from the text that is used to communicate with the user.

In addition to specifying `RANGE` restrictions following a formal fact declaration, you can also provide a prompt and translations. The syntax for specifying a prompt and translations is:

```
PROMPT text

TRANSLATE [constant | RANGE | UNKNOWN] AS text
```

For example:

```
FACT the testator died intestate[59]
PROMPT did the deceased die without leaving a will
TRANSLATE true AS the deceased died without a will
TRANSLATE false AS the deceased left a will
```

In the case of a non-boolean fact, the *translation-text* will precede the value when output. If you want to put the value somewhere else, include an empty pair of curly brackets. Where *constant* is omitted, the translation becomes the *default translation* and is used if none of the other translations apply. If *constant* is replaced by the keyword `UNKNOWN`,[60] the translation will be used where the fact value is unknown or unspecified. If *value* is replaced by the keyword `RANGE`, the translation will be used in place of the automatically generated out of range error message.

For example:

```
INTEGER the number of surviving children
RANGE 0 TO 100
PROMPT how many children survived the testator
TRANSLATE 0 AS no children survived the testator
TRANSLATE 1 AS one child survived the testator
TRANSLATE AS {} children survived the testator
TRANSLATE RANGE AS please enter the number of children (0 if none)
TRANSLATE UNKNOWN AS it is not known if the testator had any children
```

Apart from including the value of the fact itself in translations and prompts, you can also include the value of any other fact by including the fact name in curly brackets. If the value of the fact is not known, this will be determined before a value for the current fact is requested or set.[61]

---

[59] Prompts and translations will automatically be transformed into questions and sentences which start with a capital letter and finish with an appropriate punctuation character. These can also be specifically provided. Quotes surrounding text blocks will normally be removed and should be escaped if you really want a quote or translation to appear in quotes.

[60] You can also use the constant `unknown` or `unspecified` which in this context all have the same meaning.

[61] Named subjects (discussed in §4.12 on page 51) may also be used in translation text.

For example:

```
STRING the capital city
TRANSLATE AS {} is the capital of {the name of the country}
```

4.12 Named Subjects

Named subjects are used to dynamically replace references to subjects contained in facts with the names or appropriate pronouns and possessives for persons and things. There are three *types* of named subjects: `PERSON`, `THING` and `PERSON-THING`. A `PERSON` is animate with a name and optionally a gender and preferred form of address. This will be replaced with the preferred name and a pronoun. A `THING` is inanimate and is used to refer to such entities as companies and places. Where appropriate, a `THING` can be replaced by "it" or "its". A `PERSON-THING` may be either animate or inanimate such as a client, a buyer or seller, or a party to a legal action.

Named subject declarations

Named subjects must be specifically declared.[62] For example:

```
PERSON the testator
THING the agency
PERSON-THING the owner
```

During execution, when a fact containing a named subject is first used (that is, the fact becomes a goal or a value has been concluded), yscript will prompt for a name and ask questions about preferred gender or whether a `PERSON-THING` is animate or not.

For example:

```
1) What is the name of the owner?
   ** Malcolm Turnbull

2) Is Malcolm Turnbull a person?
   ** yes

3) What is Malcolm Turnbull's preferred gender?
   ** male

4) What is Malcolm Turnbull's preferred form of address?
   ** Mr Turnbull
```

Related Facts

The named subject itself is a fact with the underlying basic type `STRING`. When a named subject is declared, up to three other *related* facts may also be automatically created. These have the following types and format:

[62] Named subjects can be declared at any point in the code. They do not have to be declared before they are used.

```
STRING the name of the named subject
BOOLEAN the named subject is a person
GENDER the gender of the named subject
STRING the preferred form of address for the named subject
```

Named subjects of type PERSON-THING will generate all four of these related facts. THINGs only have a name and PERSONs have a name, gender, and preferred form of address.

The GENDER related fact, has the following default translations:

```
GENDER the gender of the named subject
PROMPT what is the named subject's preferred gender
TRANSLATE AS the named subject's preferred gender is {}
```

The *preferred form of address* related fact, has these default translations:

```
STRING the preferred form of address for the named subject
PROMPT what is the named subject's preferred form of address
TRANSLATE AS the named subject's preferred form of address is {}
```

These related facts can be used like any other facts.[63] Their values can be set and used, and new prompts and translations can be defined. For example, you could change the prompt and translation for `the named subject is a person` fact as follows:

```
PERSON-THING the taxpayer
BOOLEAN the taxpayer is a person
PROMPT the taxpayer is an individual
TRANSLATE "yes" AS the taxpayer is an individual
TRANSLATE "no" AS the taxpayer is a company or some other entity
```

Any of the facts related to a named subject can be unknown or unspecified (indicated with an "unknown", "unspecified" or *blank*) response from the user during execution. A non-blank response for preferred gender is also taken to be "unspecified". Where the name of a named subject is unknown, the description of the named subject will not be replaced in any dialogs. Where it is unknown whether a named subject refers to a person or not, or where it is known that the named subject is referring to a person, but the preferred gender is unspecified, then the name is always used to replace references to the named subject in other facts.

[63] Using a related fact that is a component of a named subject, may alter the order that the system asks for the value of other related facts to the named subject.

For example, assuming the consultation is asking about a fact named `the taxpayer` which is of type `PERSON`:

```
1) What is the name of the taxpayer?
   ** Amelia Smith

2) What is Amelia Smith's preferred gender?
   ** pansexual

3) What is Amelia Smith's preferred form of address?
   ** Amy

4) Is Amy an Australian citizen?
   ** yes

5) Does Amy earn income in Australia?
   ** what[64]

1) The name of the taxpayer is Amelia Smith.
2) Amelia Smith's preferred gender is pansexual.
3) Amelia Smith's preferred form of address is Amy.
4) Amy is an Australian citizen.
```

The intended user response a question related to the preferred form of address fact is a complete replacement for the name of the named subject.[65] Where the value appears to be a complete form of address, it will be used in place of the name. If however, the user responds with just an honorific (such as "Ms", "Dr", or "Professor"), this is used to construct a preferred name consisting of the honorific followed by a surname. If the name is a set of pronouns separated by slashes (e.g., zie/zim/zis), or if the name otherwise doesn't appear to be valid, the original name is used. In all cases, the value of the preferred form of address will be set to a complete name.

Named Subject Declaration Modifiers

Which related facts are used for a named subject and other named subject behaviour can be changed with the modifiers: `DEFINITE`, `GENDER-NEUTRAL`, `INFORMAL` and `UNNAMED`. They can be combined and are used as in:

```
DEFINITE the vehicle
PERSON the academic
GENEDER-NEUTRAL PERSON the applicant
INFORMAL PERSON the child
UNNAMED PERSON the defendant
UNNAMED GENDER-NEUTRAL PERSON the deceased
```

`DEFINITE` named subjects are used with `THING`s to automatically prepend the definite article – "the". This is often a more natural way to describe certain classes of inanimate objects that are referring to a class of things[66]. `GENDER-NEUTRAL` named subjects don't ask

[64] The "what" command is used in cyscript to display the known user supplies facts (or *premises*). See Appendix 2 for a complete list of cyscript interactive commands that are available.

[65] This may be more formal or less formal. For example, if the name is "George Brown", then the preferred form of address could be "Mr Brown" or "George".

[66] For example, the description of a group of things or the type or class of a group of persons or things.

for a preferred gender and do not use pronouns. INFORMAL named subjects don't ask for a preferred form of address. UNNAMED named subjects use a name only once, and then revert to using only the name of the named section itself or pronouns. UNNAMED named subjects never ask for a preferred form of addresses. GENDER-NEUTRAL INFORMAL named subjects only ask for a name and use this for all named subject replacements. GENDER-NEUTRAL UNNAMED named subjects are like UNNAMED subjects but do not use any pronouns.

4.13 Embedded Facts

Fact values may be embedded as part of the name of another fact by including the fact name within curly brackets. For example:

```
{the name of the child} is the child of {the name of the parent}
```

When executed, this code will lead to the following dialog:

```
1) What is the name of the child?
   ** Grace

2) What is the name of the parent?
   ** Henry

    REPORT

    Grace is the child of Henry.
```

Embedded facts can also be used in *text*, such as for document assembly, translations, and explanations. Where the value of an embedded fact is unknown prior to the use of the parent fact, a value will be determined.

4.14 Explanations

Explanations are text descriptions that add further explanation about a fact (the *default explanation*) or that provide information about the fact when it has a particular value (*valued explanations*). It is up to the application interface as to how these are used, but typically the default explanation will be displayed before the prompt seeking a value for a fact. *Valued explanations* are displayed after a user has provided a value for a fact or can also be used to give the user more information about the implications of answering a question in a particular way.

Explanations are declared in a similar way to translations. The syntax is:

```
EXPLAIN [ constant | UNKNOWN ] AS text
```

The following example code is from an application relating to community gaming:[67]

```
STRING the type of gaming activity
RANGE "Art union" RANGE "Draw Lottery"
EXPLAIN AS The Community Gaming Regulations 2020 regulate the
    Conduct of gambling for social, charitable and non-profit
    purposes in NSW. The Regulations provide for several categories
    of permitted gaming activities.
EXPLAIN "Art union" AS You will now be asked a series of questions
    to see whether or not your proposed activity is covered by the
    "Art Union gaming activity" provisions which are contained in
    regulation 4.
EXPLAIN "Draw lottery" AS
    You will now be asked a series of questions to see whether or
    not your proposed activity is covered by the "Draw Lottery"
    provisions which are contained in regulation 6.

RULE PROVIDES
IF the type of gaming activity EQUALS "Art union" THEN
    DETERMINE IF regulation 4 applies
IF the type of gaming activity EQUALS "Draw Lottery" THEN
    DETERMINE IF regulation 6 applies
```

When executed using cyscript, the session will go:

```
The Community Gaming Regulations 2020 regulate the Conduct of gambling for
social, charitable and non-profit purposes in NSW. The Regulations provide
for several categories of permitted gaming
activities.

1) What is the type of gaming activity?
   (a) Art union
   (b) Draw Lottery

    Please select?
    ** b

You will now be asked a series of questions to see whether or not
Your proposed activity is covered by the "Draw Lottery" provisions
Which are contained in regulation 6.

2) Does regulation 6 apply?
   **
```

4.15 Information

Apart from attaching explanations that always get displayed before a question is asked or when a particular answer is made or contemplated, it is also possible to provide further information about a question using the `INFO` sub-declaration.

The syntax is:

```
INFO text
```

[67] This type of application is probably better dealt with by use of the CASE-WHEN-THEN statement which automatically applies range restrictions. See § 5.3 at page 64.

For example:

```
FACT you are married
INFO If you are unsure about whether you are married, please see the
Marriage Act 1961 (Cth).
```

In cyscript, the `INFO` message will be displayed whenever the user gives a blank or illegal response to a question. Other calling environments can display this when a user hovers the mouse pointer over a question or clicks on an "information" type icon or symbol.

4.16 Attachments

Fact declarations may include one or more *attachments*. Attachments are used in document assembly and are more fully discussed in § 8.5. The syntax is:

```
ATTACH [DISPLAYED] [qualifier] REPORT|DOCUMENT|TEMPLATE name [AS
description]
```

4.17 Aliases

Aliases are also used in document assembly and provide a way to specify a short name for a fact as used in document templates.[68] The syntax is:

```
ALIAS name
```

For example:

```
STRING the name of the first mentioned party
ALIAS party_one
```

4.18 Context

For most small applications, all fact names and rules exist in the same namespace. Any rule can use and set a value for any fact. For larger projects, yscript code can be divided into *contexts*. A *context* provides a separate namespace for facts and rule names that are related, and which can operate independently or as part of some larger application. A file may contain more than one context and a context can be split across several files.

Context Declaration

At the start of each new file, rules and facts are in the *shared context* and are available in all other contexts. A new context is introduced with a declaration of the form:

[68] See § 8.5 at page 97.

```
CONTEXT context-name
```

where context-name is the name of a new of existing context. For example, to specify that the rules and facts following in a file are to do with a particular piece of legislation, you could say:

```
CONTEXT Modern Slavery Act 2018 (Cth)
```

Once declared, everything following the context declaration will be within the specified context until either another context declaration or end of file.

Namespace

Rules and facts within each context have their own separate namespace. If a fact or rule exists with same name as one in the shared context, the version of the fact in the current context will be used.[69] Similarly, if identically named facts or rules exist in two or more contexts, all are treated as referring to different facts.

Where you need to refer to a fact from a different context (other than the shared context), you can declare it within a given context as follows:

```
type fact FROM context
```

For example:

```
CONTEXT Modern Slavery Act 2018
FACT section 23 applies FROM Corporations Act 2001
NUMBER amount of proceeds FROM Gaming Regulations 2020
```

All references to the fact with the current context will then be taken to be to the fact *from* the external context. You can access the fact value and/or set a new value. Rules needing the fact will use it regardless of whether they are in the original context or the context that has imported the fact.

[69] This will result in a warning message. It is better to avoid this situation by renaming facts if possible.

Consider the example:

```
THING the entity

CONTEXT Modern Slavery Act 2018 (Cth)

FACT the entity carried on business in Australia FROM Corporations Act 2001
TRANSLATE true AS the entity carried on business in Australia under section
21 of the Corporations Act 2001
TRANSLATE false AS the entity did not carry-on business in Australia under
section 21 of the Corporations Act 2001

RULE Section 5(2) PROVIDES
the entity carried on business in Australia during the reporting period
ONLY IF
    the entity was a body corporate during the reporting period AND
    the entity carried on business in Australia

CONTEXT Corporations Act 2001

RULE Section 21 PROVIDES
the entity carried on business in Australia ONLY IF
    section 21(1) applies OR
    section 21(2) applies

RULE Corporations Act 2001 Section 21(1) PROVIDES
section 21(1) applies ONLY IF
    the entity has a place of business in Australia, or in a State
    or Territory
```

In this example, we declare two contexts: one for material relating to the *Modern Slavery Act 2018 (Cth)* and another for the *Corporations Act 2001*. The named subject - `the entity`, is declared in the shared context and so is available in both other defined contexts. The first of these Acts relies on the other for a defined term – `the entity carried on business in Australia`. The `Modern Slavery Act (Cth)` context specifically imports this fact making it shared between the two contexts. The rest of the facts have their own separate name spaces. When executed, the session will go:

```
1) What is the name of the entity?
   ** Telstra

2) Was Telstra a body corporate during the reporting period?
   ** yes

3) Does Telstra have a place of business in Australia, or in
   a State or Territory?
   ** yes

   REPORT

   Telstra carried on business in Australia during the
   reporting period because it was a body corporate during the
   reporting period and it carried on business in Australia
   under section 21 of the Corporations Act 2001.

   Telstra carried on business in Australia under section 21 of
   the Corporations Act 2001 because section 21(1) applies.
   Section 21(1) applies because Telstra has a place of
   business in Australia, or in a State or Territory.
```

Rule and Procedure References

If you want to expressly call[70] a rule or procedure from a remote context, you can use the syntax:

```
CALL rule-name FROM context
```

for example:

```
CALL section 23 FROM Corporations Act 2001
```

Rule, goal and fact names returned via the API do not reflect the context. The name of link facts should incorporate enough information to identify the fact independently. If you need to specify the name of a goal and the context is important, you can proceed the goal pattern with the name of the context followed by two colons, for example:

```
Modern Slavery Act 2018 (Cth)::section 23
```

[70] The CALL statement is discussed in the following chapter at § 5.6 on page 66.

Chapter 5: Statements

Rules are comprised of one or more *statements*. These include *assignments* and `IF–THEN–ELSE` control statements. Other statements can be used to perform multi-way branches (`CASE–WHEN`), `DETERMINE` the value of a fact, to specifically `CALL` a particular rule, to specify loops (`WHILE` and `REPEAT`) and to imperatively output text (`SAY`).

Multiple statements can be grouped as a single statement by enclosing them with `BEGIN` and `END`.

5.1 Assignments and Assertions

Assignments are used to set the value of a fact. There are two basic types of assignment: *assertions*, where a truth (boolean) value is set by asserting a fact in the positive or negative and *regular assignments* where any type of fact is explicitly assigned a value. The assignment syntax is:

```
[ASSERT] boolean-fact { AND boolean-fact } |
[ASSERT] fact IS | ONLY IF expression
```

Assertions

To assert a `BOOLEAN` fact, you simply express the fact in positive or negative form. Where necessary to separate the assertion from a previous statement, it can be preceded with the keyword `ASSERT`. Additional assertions can follow by adding the keyword `AND` then further assertions. For example:

```
section 123 applies

section 123 does not apply

the testator died intestate AND
the estate will be distributed according to the rules of intestacy

ASSERT the activity is illegal
ASSERT penalties will apply
```

Regular Assignments

The `IS` and `ONLY IF` operators are used to assign a specific value of an expression to a fact. The two operators are entirely equivalent, but generally `ONLY IF` is used for `BOOLEAN` assignments and `IS` for everything else. Some examples:

```
the applicant meets the requirements of section 49 ONLY IF
    section 49(1) is satisfied AND
    section 49(2) is satisfied

ASSERT the date of the agreement IS 31 January 2020

ASSERT the value of the estate IS £200,000 MINUS the cost of taxes
```

One of the side-effects of assignments is that they associate facts that are used to calculate the assigned value, as reasons for *how* the value of the target fact was determined. Internally, this information is stored as a *how list* for each fact. A fact can never be a justification for itself, and self-references are excluded from ever being added to how list for a fact.

5.2 IF-THEN-ELSE Statement

`IF-THEN-ELSE` statements provide for conditional evaluation of other statements. The syntax is:

```
IF expression THEN statement [ ELSE statement ]
```

Where *expression* is true, then the first *statement* is executed, otherwise (if present) the second *statement* following the `ELSE` is executed. For example:

```
IF the entity fails to comply with a request under section 16A THEN
     the Minister may publish the identity of the entity AND
     the Minister may publish the identities of all entities included by a
     joint modern slavery statement

IF the deadline for submission has been extended THEN
    the last day for submission IS
        the last day of the financial year PLUS 9 months
ELSE
    the last day for submission IS
         the last day of the financial year PLUS 3 months
```

`IF-THEN-ELSE` statements can be joined together to test for a range of conditions. For example:

```
IF the name of the entity EQUALS "DFAT" THEN
    the entity is a commonwealth government department
ELSE IF the name of the entity EQUALS "Telstra" THEN
    the entity is a public company
ELSE
    nothing much is known about the nature of the entity
```

`BEGIN-END` pairs can also be used to make the meaning and function of nested `IF-THEN-ELSE` statements clearer.

For example:

```
IF section 16(1) applies THEN BEGIN
    IF the exemption in section 16(2) applies THEN
        there is not a breach of section 16
END ELSE
    section 16 does not apply
```

5.3 CASE-WHEN Statement

The `CASE-WHEN` statement is a multi-way branch statement which compares the value of a fact against one or more constants.

The syntax is:

```
CASE fact-name[71] { WHEN constant-value | DEFAULT [ THEN ] statement }
```

The value of the fact referred to by *fact-name* is compared against each of the specified constant values. The first statement associated with a constant value which matches the fact value is executed. If no constant matches the fact value, then the statement associated with the DEFAULT instance is executed. The DEFAULT instance must be the last condition in a CASE statement.

Where *fact-name* refers to a fact of type STRING[72] and there is no default condition, the range of the fact will be restricted to these values. For example:

```
CASE you would like to immigrate
    WHEN "yes" THEN you should apply for an immigration visa
    WHEN "no" THEN CALL you should apply for a visitor visa
    WHEN "maybe" THEN CALL you should get advice about an appropriate visa
```

will result in the automatic declaration:

```
STRING FACT you would like to immigrate
RANGE "yes"
RANGE "no"
RANGE "maybe"
```

and result in a multi-choice question when executed:

```
Would you like to immigrate?
(a) yes
(b) no
(c) maybe

Please select:
**
```

5.4 WHILE-DO Statement

WHILE-DO statements allow for a statement to be executed whilst an expression is true. The syntax is:

```
WHILE expression DO statement
```

The statement is inherently procedural and is seldom used unless you are attempting to use yscript as a general-purpose programming language.[73] To illustrate what is possible, the following example will calculate the value of the mathematical constant *Pi*:

---

[71] Where a CASE-WHEN statement immediately follows the declaration of a fact (that is when the fact is an *object*), then CASE fact-name may be omitted. See § 6.8 at page 78 below.

[72] If the type is not declared, then the fact's type will be implicitly declared as STRING.

[73] See § 6.7 below for a discussion of the limitations on loops.

```
// Calculate the value of Pi

SYSTEM NUMBER i

PROCEDURE Pi PROVIDES
ASSERT i IS 2.0
ASSERT pi IS 3.0
WHILE i LESS THAN 128 DO BEGIN
     pi IS pi PLUS 4 DIVIDED BEGIN
            i TIMES
            BEGIN i PLUS 1 END TIMES
            BEGIN i PLUS 2 END
     END
     i IS i PLUS 2
     ASSERT pi IS pi MINUS 4 DIVIDED BEGIN
            i TIMES
            BEGIN i PLUS 1 END TIMES
            BEGIN i PLUS 2 END
     END
     i IS i PLUS 2
     SAY Pi is <pi>.
END
```

5.5 REPEAT-UNTIL Statement

The REPEAT-UNTIL statement executes a set of one or more statements until an expression is true. Its syntax is:

```
REPEAT statements UNTIL expression
```

Again, this is used infrequently in practice, but there is room for an example:

```
//      Using yscript as a general-purpose programming language
//
//     (aka "how not to use yscript" :-)
//

PROCEDURE main PROVIDES
CALL Squares

SYSTEM INTEGER x
SYSTEM INTEGER x2
SYSTEM INTEGER the largest number to show squares for

PROCEDURE Squares PROVIDES
x IS 1
REPEAT
        ASSERT x2 IS x TIMES x
        SAY {x} squared is {x2}
        ASSERT x IS x PLUS 1
UNTIL x IS GREATER THAN the largest number to show squares for
```

5.6 CALL / SUBRULE Statement

The CALL statement allows *rules* and *procedures* to be invoked explicitly. The syntax is:

```
CALL | SUBRULE [ GOAL ] rule-name [ FROM context ]
```

The statements for the named *rule* or *procedure* will be executed and control will be returned to the next statement. The CALL statement is procedural and should be used

with care.[74] Where it is more natural, the CALL keyword can be replaced by SUBRULE. Where the GOAL keyword is specified, the specified rule will become the current goal.

Where a rule is not in the current context or the shared context, you can still call it by specifying the *context* using the FROM *context* syntax.

For example:

```
RULE Section 45 PROVIDES
SUBRULE Section 45(1)
SUBRULE Section 45(2)
SUBRULE Section 45(3)
```

5.7 NEXT Statement

The NEXT statement is the same as the CALL statement but is does not return. When the rule or procedure completes, the session finishes. The syntax is:

```
NEXT [ GOAL] rule-name [ FROM context ]
```

For example:

```
DOCUMENT Non-Disclosure Agreement PROVIDES
CALL Preamble
IF the agreement is unilateral THEN
    NEXT Unilateral Agreement
ELSE
    NEXT Multi-party Agreement
```

5.8 DETERMINE Statement

The DETERMINE statement causes the value of a fact to be determined, that is the fact becomes the current goal fact. The syntax is:

```
DETERMINE [IF] fact
```

Unless you are intentionally writing imperative code, the use of this statement is probably best confined to goal rules. Otherwise, if facts are *forgotten* or if the system otherwise must recalculate everything, it may lead to unanticipated results.

One of the uses of including DETERMINE in a goal rule is that it has the effect of adding the fact to list of those that will be *reported* on. *Reports* are discussed in § 6.4.

[74] See § 6.7 *Writing Procedural Code* below.

An example:

```
GOAL RULE Competition and Consumer Act 2010 Section 52F – Application for
registration of news business and news business corporation PROVIDES
DETERMINE the News Business Corporation may apply to the ACMA for
    Registration of the news business under section 52F(1)(a)
DETERMINE the News Business Corporation may apply to the ACMA for
    registration of the News Business Corporation under section 52F(1)(b)
DETERMINE the News Business Corporation may apply to the ACMA for
    endorsement of the News Business Corporation as the registered news
    business corporation for the news business under section 52F(1)(c)
```

5.9 FORGET Statement

The `FORGET` statement causes the value of facts to be forgotten. The syntax is:

```
FORGET ALL | fact-name
```

Where followed by the keyword `ALL`, all fact values will be cleared, and the session will restart.

Where a *`fact-name`* is specified, the value for this fact and all facts that were subsequently determined or provided by the user after this fact will be set to `UNKNOWN`. See §6.7 *Writing Procedure Code* below for examples of how `FORGET` should be used.

An example:

```
RULE Contracts Formation PROVIDES
a contract has been formed ONLY IF
    there has been an offer AND
    the offer has been accepted AND
    there is consideration
IF you have made any mistakes THEN
    FORGET ALL
```

5.10 SAY Statement

The `SAY` statement sends a text message which will be displayed the next time that the controlling environment asks the user a question. The syntax is:

```
SAY text
```

for example:

```
SAY "Hello, World"
```

The statement is procedural and particularly if included in backward or forward-chaining rules will produce a lot of unwanted text. Even in procedural code, it will be invoked repeatedly as the system rebuilds between user questions. See § 4.14 Explanations for a mechanism to attach text to facts which is more controllable.

Normally, the SAY statement is only useful for debugging. Consider the following example:

```
RULE Giants PROVIDES
SAY "Fee"
IF you heard that THEN BEGIN
    SAY "Fi"
    IF you heard another sound THEN BEGIN
       SAY "Fo"
       IF it sounds like a giant is coming THEN BEGIN
           SAY "Fum"
           SAY "It is definitely a giant!"
       END
    END
END
```

This code will produce the following result when executed:

```
1) Did you hear that?
   ** y

   Fee
   Fi

2) Did you hear another sound?
   ** y

   Fee
   Fi
   Fo

3) Does it sound like a giant is coming?
   ** y

   Fee
   Fi
   Fo
   Fum
   It is definitely a giant!
```

5.11 EXIT Statement

The EXIT statement terminates a session. The syntax is:

```
EXIT [SESSION]
```

Without the SESSION argument, the statement will case the interpreter to terminate. With the SESSION argument, control will return to the calling environment. If you want to restart a session, use FORGET ALL.

5.12 INCLUDE Directive

The INCLUDE directive can be used to include the contents of another file[75] as part of the source. This is useful for organising large pieces of code and possibly for assisting with collaboration.[76] It can be used within a rule as a statement or outside rules as a declaration.

The syntax is:

```
INCLUDE filename
```

For example, where the file interpretation.ys includes some standard interpretation provisions such as:

```
// interpretation.ys

RULE Persons PROVIDES
IF the party is a company THEN
    the party is not a person
IF the party is a person THEN BEGIN
    the party is not a company AND
    the party is not a partnership
```

it could be included as:

```
INCLUDE "interpretation.ys"

RULE Section 56 PROVIDES
Section 57 applies ONLY IF
    the party is a company
```

[75] The DataLex environment extends INCLUDE to allow for URLs.

[76] Specifying multiple files when running code is probably more flexible.

Chapter 6: Rules

Statements are contained in *rules*. There must be at least one rule in any piece of yscript code. [77] Execution commences with the *goal rule*.[78]

6.1 Default Rule Behaviour

Once the goal rule has started execution (or has been *fired*), the rule statements are executed in order. By default, if a fact value is required as part of an expression for an assignment or as a condition for a control statement (such as `IF`), yscript will fire each of the rules in the code that potentially can determine a value for the fact until a value is available.[79] If a value can't be determined, the user will be prompted to provide one. If a value is found, each of the rules in code that make use of the fact with the newly determined value are fired but will block if a user response is ever required.

The default behaviour of rules in yscript can hence be said to be goal oriented or *backward chaining*. The behaviour is also *forward chaining* in the sense that rules will be used to determine every possible fact value that can be reached given the current facts which are known.[80] This behaviour can be altered by changing the rule type.

6.2 Rule Declarations and Types

A rule declaration consists of a *rule header* followed by one or more statements. The rule header contains an optional `GOAL` qualifier, a rule type, and an optional rule name. This is followed by the keyword `PROVIDES` and then the statements that make up the rule.

The syntax is:

```
        [GOAL] rule-type [RULE] [rule-name] PROVIDES statements
```

where *rule-type* is one of:

```
        BACKWARD|DAEMON|DOCUMENT|FORWARD|PROCEDURE|RULE
```

---

[77] This includes an object which is essentially a fact with statements. See § 6.8 at page 78.

[78] Goal rules are discussed in § 6.3 at page 72.

[79] When backward chaining, the system keeps track of which rules are already being evaluated. If a rule ever becomes reliant upon itself (directly or indirectly), the rule will *block* and consideration will move to the next available rule (if any).

[80] Another way of putting this is that the default rule type is `BACKWARD` chaining but that it also operates as a forward-chaining `DAEMON`.

For example:

```
RULE Common-sense about companies PROVIDES
IF the client is a company THEN
    the client is not a person
```

The rule name is optional, but it is strongly recommended that you include rule names as these are used to select goals and rules to be CALLED as well as for describing what the system is doing and why it is doing it. Rule names are in a different namespace from fact names.

You can change the way that rules are used, by declaring with a type other than just RULE. The following additional rule types are available:

Backward Rules

A rule that is declared to be BACKWARD will only ever backward chain, that is it will not be used to determine fact values except in a strictly goal driven fashion. This will have the effect of stopping conclusions being made that are strictly unnecessary.

Forward Rules

A rule that is declared to be FORWARD will be fired whenever the value of a fact that it uses becomes known. Forward rules are not used for backward chaining.

Daemons

A DAEMON is fired whenever a fact that it uses becomes known. Unlike a FORWARD rule, a DAEMON operates silently and will never cause a question to be asked of the user. Daemons are not used when backward chaining.

Procedures

PROCEDUREs are not used to determine any fact values unless they are specifically called with a CALL or NEXT statement.

Document Rules

DOCUMENT rules are used for document assembly which is discussed in Chapter 8: Document Assembly. They are like procedures but allow additional statements for generating text to be included in documents.

6.3 Goal Rules

If there is only one rule, or no goal rules are explicitly specified, then the *goal rule* will be the first rule that is declared. Goal rules can be declared by including the qualifier GOAL before the rule declaration.

For example, the goal rule for the Modern Slavery application is:

```
GOAL RULE Modern Slavery Act 2018 (Cth)
```

If only one goal rule is declared, this will be used in place of the first rule to start execution. If there is more than one goal rule, the user will be asked to select which goal rule they wish to use.

For example:

```
The following goals are defined:

1) Community Gaming Regulation 2020 Regulation 4 – Art union
   gaming activities
2) Community Gaming Regulation 2020 Regulation 5 – Housie or bingo
3) Community Gaming Regulation 2020 Regulation 6 – Draw lotteries
4) Community Gaming Regulation 2020 Regulation 7 – No-draw lotteries
5) Community Gaming Regulation 2020 Regulation 8 – Mini-numbers
   lotteries
6) Community Gaming Regulation 2020 Regulation 9 – Progressive
   lotteries
7) Community Gaming Regulation 2020 Regulation 10 – Free lotteries
8) Community Gaming Regulation 2020 Regulation 11 – Promotional
   raffles conducted by registered clubs
9) Community Gaming Regulation 2020 Regulation 12 – Other
   gaming activities for charitable purposes
10) Community Gaming Regulation 2020 Regulation 13 – Sweeps and
    calcuttas
11) Community Gaming Regulation 2020 Regulation 14 – Trade promotion
    gaming activities

Please select a goal?
**
```

6.4 Reports

Apart from being important in specifying a starting point, goal rules also have a direct effect upon *reports*. By default,[81] a report is generated at the end of each user session and is designed to give a summary of the outcomes and the reasons for them. All facts that are assigned a value or that are the subject of a `DETERMINE` statement in the goal rule are included in the final report. The report will include an explanation as to how each of these facts was determined and will go on to recursively explain each of sub-conclusions necessary to support this explanation.

[81] This behaviour can be changed by using attachments. See § 8.6 at page 97.

For example:

```
AustLII Bingo is not permitted "housie" or "bingo" under
regulation 5 because: it is not Charity housie under
regulation 5(2); it is not Social housie under regulation
5(3); and it is not Club bingo under regulation 5(4).

AustLII Bingo is not Charity housie under regulation 5(2)
because regulation 5(2)(c) is not satisfied. Regulation
5(2)(c) is not satisfied because the total value of the
expenses of conducting AustLII Bingo (excluding the cost of
prizes) is more than 12.5% of the gross proceeds of the
gaming activity. The total value of the expenses of
conducting AustLII Bingo (excluding the cost of prizes) is
more than 12.5% of the gross proceeds of the gaming activity
because the total value of the expenses of conducting it
(excluding the cost of prizes) is $1,000 and the amount of
the gross proceeds of AustLII Bingo are $5,000.
```

If you want to include a fact that would otherwise not be included, use `DETERMINE` in the goal rule. If you want to exclude a fact from the report, use the `UNREPORTED` qualifier when the fact is declared.

Where the goal rule does not determine the value of any facts, the report will explain all determined facts in reverse order to that in which they were determined.

6.5 Rule Order

When more than one rule can potentially help to determine the value for a fact, yscript will normally fire each rule in the order that they were declared, that is earlier rules in earlier files will go first. If you wish to change the order in which rules are evaluated, you can simply re-order them. The disadvantage of this approach is that it might mean that rule order is no longer reflective of a source rule set.

The order in which rules are fired can be explicitly set with the `ORDER` statement. The syntax is:

```
ORDER rule-name { THEN rule-name }
```

For example:

```
ORDER Section 23 THEN Section 3 THEN Section 10
```

6.6 Generic Rules

Generic rules allow for the declaration of rule templates which have a similar form but match a number of different sets of facts. The aim of generic rules is to save you having to rewrite essentially the same rule but with facts with slightly different names.

Within generic rules, facts may contain a variable element which is indicated by an empty pair of curly brackets. Prior to executing the code, yscript will automatically produce a new rule for each fact that it can find matching the specified pattern in the

rule template. The variable part of a fact name represented by {} must be the same in all facts in the rule. For example, the *Copyright Act 1968 (Cth)* contains a definition of "qualified person" in section 32(4) which can apply at "the time that the work was made" or "at the time the work was first published".[82]

The following generic rule will deal with all these facts:

```
RULE Copyright Act 1968 – Section 32(4) PROVIDES
    the author was a "qualified person" {} under the section 32(4)
    definition ONLY IF
        the author was an Australian citizen {} OR
        the author was an Australian protected person {} OR
        the author was a person resident in Australia {}
```

It is equivalent to writing two rules:

```
RULE Copyright Act 1968 – Section 32(4) PROVIDES
    the author was a "qualified person" at the time that the work was
    made under the section 32(4) definition ONLY IF
        the author was an Australian citizen at the time that the work
        was made OR
        the author was an Australian protected person at the time that
        the work was made OR
        the author was a person resident in Australia at the time that
        the work was made

RULE Copyright Act 1968 – Section 32(4) PROVIDES
    the author was a "qualified person" at the time the work was first
    published under the section 32(4) definition ONLY IF
        the author was an Australian citizen at the time the work was
        first published OR
        the author was an Australian protected person at the time the
        work was first published OR
        the author was a person resident in Australia at the time the
        work was first published
```

6.7 Writing Procedural Code

This section provides a slightly technical explanation of some issues relating to writing procedural code in yscript.

By its nature, yscript is a declarative language. The order of execution is normally driven by goal-directed backward chaining or result-directed forward firing of rules. Rules and expressions themselves, however, are imperative. Statements in rules are executed sequentially, and facts and constants in expressions are evaluated according to operator precedence then are evaluated left to right.

When an expression is being evaluated, each reference to an uninitialised fact may result in a rule firing to attempt to derive a value, or in a question being asked of the user. The rules engine will block attempts by rules to invoke themselves to stop loops and infinite recursion. Each time a question is asked, control returns to the calling environment. When control passes back to the yscript interpreter, the rules are re-

[82] There are also several other forms contained in the Act such as: s32(1)(b) *for a substantial part of the period of the making of the work*; and s32(2)(e) *immediately before the author's death*.

executed from the starting rule to re-establish the current state and to use any new information that has been provided.

yscript includes several statements which facilitate procedural programming. These include control statements such as `WHILE-DO` and `REPEAT-UNTIL`, as well as the `CALL` and `NEXT` statements which will specifically invoke a particular rule. The language also includes the `FORGET` statement which will de-initialise the value of a fact and all facts that were subsequently determined.

Most of the time, procedural code will execute as in any other imperative computer language. The main exceptions to this are when a rule directly or indirectly invokes itself via `CALL` or `NEXT` or where a `WHILE-DO` or `REPEAT-UNTIL` loop relies on a fact that needs to be re-initialised on each iteration of the loop.

Consider the latter case first. The following code will run as expected because it does not rely upon any interaction with the end-user (that is, no questions are asked):

```
ASSERT count IS 1
WHILE count LESS THAN 10 DO
    ASSERT count IS count PLUS 1
```

In this example, the fact – `count`, is first initialised to `1` and remains initialised throughout the execution of the while loop until it reaches the value of `10`.

Consider the following example however:

```
WHILE you wish to continue DO       // incorrect
    SAY do something
```

This will cause the user to be asked the question: "*Do you wish to continue?*". If the user answers "*no*", control will continue after the statement. If the user answers "*yes*", this will produce an infinite loop.[83]

An incorrect way of trying to address this would be to insert a `FORGET` statement before the loop as in:

```
FORGET you wish to continue        // incorrect
WHILE you wish to continue DO
    SAY do something
```

This example will not work and will result in the question "*Do you wish to continue?*" being asked indefinitely. Each time that the value of "`you wish to continue`" is forgotten, the interpreter will pass control back to the calling environment to ask the user for a value and then when restarted, will forget the value again.

[83] The infinite loop will be picked up by the interpreter and it will issue a fatal error message: `maximum recursion level reached`.

The correct way to write this code is to make sure that any FORGET statement comes after a reference to the fact to which it refers:

```
WHILE you wish to continue DO BEGIN
    SAY do something
    FORGET you wish to continue
END
```

In this case, the WHILE loop never actually iterates. When the value of "you wish to continue" is forgotten, the system restarts and stops when it encounters the reference to it in the guard for the loop. Arguably, the code is better written as:

```
IF you wish to continue THEN BEGIN
    SAY do something
    FORGET you wish to continue
END
```

In the same way that the interpreter's rules engine will block backward-chaining rules from calling themselves, attempts to CALL a rule, or pass control via NEXT to a rule, that is already on the rules stack will also be blocked. When a CALL or NEXT statement attempts to execute a rule that is already being executed, the interpreter will automatically call FORGET to bring the system back to the state that it was in when the rule was originally called. The effect of this is that it is always safe to use CALL or NEXT to invoke any rule, and if necessary, the system will backtrack to produce a sensible result.

Consider, for example, the following code:

```
PROCEDURE Options PROVIDES
CASE your preferred option
    WHEN "option 1" CALL Option 1
    WHEN "option 2" CALL Option 2

PROCEDURE Option 1 PROVIDES
SAY "Something about option 1"
FORGET your preferred option

PROCEDURE Option 2 PROVIDES
SAY "Something about option 2"
CALL Options
```

The aim of the code is to present the user with two options, call a procedure based on their answer and then ask for another option. The approach set out in "Option 1" is like that discussed above in relation to WHILE–DO loops. The "Option 1" procedure does what it needs to and then the call to FORGET will forget the value of "your preferred option", and the question "*What is your preferred option?*" will be asked.

"Option 2" will also work. When the user selects "option 2", the procedure "Option 2" is executed and the SAY message "*Something about option 2*" is sent to the message queue for display the next time the calling environment gets control and asks the user a question. When the CALL to "Options" is made, this will be blocked because the "Options" procedure is already on the goal rule stack. Instead, the interpreter will FORGET the value of "your preferred option" and only then do the CALL to "Options" (again, producing the question "What is your preferred option?".

You can see what is happening, by putting cyscript into verbose mode:

```
* FIRING Options
   1) What is your preferred option?
      (a) option 1
      (b) option 2
      Please select?
      ** b

 * DETERMINED VALUE FOR your preferred option
  * FIRING Options
    * BLOCKED Options
    * FORGOT your preferred option
      Something about option 2

   1) What is your preferred option?
      (a) option 1
      (b) option 2
      Please select?
      **
```

6.8 Rules as Objects

The namespace for rule names and facts is different and it is possible to declare a rule and a fact with the same name. When this is done, a fact effectively becomes a very simple *object* in the sense that it can hold a value and that it also has associated code.

To provide a convenient way to write procedural code to implement decision-trees and chatbots, yscript provides for a short-form way of associating a procedure with a fact. Fact declarations may be followed by a set of statements which will form a procedure with the same name. For example:

```
FACT it is raining
WHEN true NEXT take an umbrella
WHEN false NEXT it looks like it might rain
```

is equivalent to:

```
FACT it is raining

PROCEDURE it is raining PROVIDES
CASE it is raining
    WHEN true NEXT take an umbrella
    WHEN false NEXT it looks like it might rain
```

Where a `CASE-WHEN` statement immediately follows a fact declaration, the "`CASE` *`fact-name`*" part of the statement can be omitted, and the fact name will be used instead.

When implementing decision-trees, it is common to ask a question and then to branch depending upon a response. Permitting the `CASE` part of the `CASE-WHEN` statement is a convenience to reduce code repetition and to make the code more compact.

The following example illustrates how a simple decision-tree can be implemented using the object syntax:

```
FACT there has been an offer
WHEN true NEXT the offer has been accepted
WHEN false THEN a contract has not been formed

FACT the offer has been accepted
WHEN true NEXT there is consideration
WHEN false THEN a contract has not been formed

FACT there is consideration
WHEN true THEN a contract has been formed
WHEN false THEN a contract has not been formed
```

This will result in the session:

```
1) Has there been an offer?
   ** y

2) Has the offer been accepted?
   ** y

3) Is there consideration?
   ** y

   A contract has been formed because: there has been an offer;
   the offer has been accepted; and there is consideration.
```

Chapter 7: Examples

Apart from the rule-based approaches outlined above, yscript also supports simple analogous reasoning. This is based on a method of measuring similarity of examples called PANNDA, developed by Alan Tyree and described in his book *Expert Systems in Law*, Prentice Hall, 1990. The method is designed to support case-based reasoning in law but may be of application in other domains.

7.1 Example Declarations

When yscript is trying to infer a value for a fact and no further rules can be found to assist, it will look to see if the fact is contained in an *example set*. The syntax to declare an `EXAMPLE` is:

```
[GOAL] EXAMPLE [RULE] name PROVIDES [IF expression THEN] assignment
```

The *expression* component of either the `IF` guard or the *`assignment`* itself, should consist of a number of relative expressions separated by an `AND` operator. Each relative expression (normally just a fact name) should represent one significant facet of the example. The OR connector should not generally be used. The following example comes from the finding of chattels code:

```
EXAMPLE Armory v Delamirie PROVIDES
    the finder wins ONLY IF
        the finder was not the occupier of the premises AND
        the chattel was not attached AND
        the non-finder was not the owner of the real estate AND
        the non-finder was not the owner of the chattel AND
        there was a bailment of the chattel AND
        there was not a term in a lease which mentioned found items AND
        there was not a master-servant relationship between the parties AND
        the chattel was not hidden AND
        there was not an attempt to find the true owner of the chattel AND
        there was prior knowledge of the existence of the chattel
```

The `IF-THEN` form of the declaration should only be used where the fact about which the example relates is non-boolean.

It is important that each example is sensibly named. The name is used to construct three automatic facts of the form:

```
the situation is similar to name
the situation is on all-fours with name
name can be distinguished
```

7.2 Example Evaluation

When the system is about to attempt to infer a value for a fact using an example set, it first finds all examples which relate to it (that is, all examples where the fact appears

as the target of an assignment). It then infers (or asks the user for) a value for all facts used in the examples. Finally, it compares each example with the situation described by these fact values and finds the *nearest* and *furthest* example. The furthest example is the one with the closest facts but giving a different result to the nearest one.

The target fact is set to the same value as the nearest example. The *similar* or *all-fours* fact for the nearest example is set to true. If the example is not on all fours, the *distinguished* fact is also set for the furthest case. All facts (including the target fact itself) receive sensible explanatory associations (for *how/reporting*). Not all possible supporting facts are used for explanations. Rather, only *significant* ones are reported (*significant* facts are those which tend to, in themselves, divide the example set or in this instance have unusual values).

The underlying mechanism used to handle analogous reasoning is based on Tyree's PANNDA algorithm (where each matching fact is weighted according to how poorly it divides the example set as measured by its inverse variance). This approach has also been extended in several minor respects:

1. The original PANNDA algorithm dealt only in boolean facts and outcomes. There has never really been any good reason why the outcomes had to be boolean (they are not used in determining which case to follow or distinguish). Accordingly, this restriction has been dropped in the yscript implementation.

2. yscript also supports non-boolean facts. The variance for each of these is calculated in the context of the present fact value and so care should be taken with use of equality operators. These should only be used where the fact can only take one of a discrete number of values.

3. It is not necessary that each example contains all attributes used in other examples. This feature can be used to generalise the effect of an example. The missing attributes become, in effect, *wild*. Such examples, are, of course, much easier to match. Again, caution is called for.

7.3 Finders Example

This is the complete finding of chattels example which uses the cases from the original PANNDA Finders code:

```
EXAMPLE Armory v Delamirie PROVIDES
the finder wins ONLY IF
    the finder was not the occupier of the premises AND
    the chattel was not attached AND
    the non-finder was not the owner of the real estate AND
    the non-finder was not the owner of the chattel AND
    there was a bailment of the chattel AND
    there was not a term in a lease which mentioned found items AND
    there was not a master-servant relationship between the
    parties AND
    the chattel was not hidden AND
    there was not an attempt to find the true owner of the
    chattel AND
    there was prior knowledge of the existence of the chattel

EXAMPLE Bridges v Hawkesworth PROVIDES
the finder wins ONLY IF
    the finder was not the occupier of the premises AND
    the chattel was not attached AND
    the non-finder was the owner of the real estate AND
    the non-finder was not the owner of the chattel AND
    there was a bailment of the chattel AND
    there was not a term in a lease which mentioned found items AND
    there was not a master-servant relationship between the
    parties AND
    the chattel was not hidden AND
    there was an attempt to find the true owner of the chattel AND
    there was not prior knowledge of the existence of the chattel

EXAMPLE Elwes v Brigg Gas PROVIDES
the finder does not win ONLY IF
    the finder was the occupier of the premises AND
    the chattel was attached AND
    the non-finder was the owner of the real estate AND
    the non-finder was not the owner of the chattel AND
    there was not a bailment of the chattel AND
    there was a term in a lease which mentioned found items AND
    there was not a master-servant relationship between the
    parties AND
    the chattel was hidden AND
    there was an attempt to find the true owner of the chattel AND
    there was not prior knowledge of the existence of the chattel

EXAMPLE Hannah v Peel PROVIDES
the finder wins ONLY IF
    the finder was not the occupier of the premises AND
    the chattel was not attached AND
    the non-finder was the owner of the real estate AND
    the non-finder was not the owner of the chattel AND
    there was not a bailment of the chattel AND
    there was not a term in a lease which mentioned found items AND
    there was not a master-servant relationship between the
    parties AND
    the chattel was hidden AND
    there was an attempt to find the true owner of the chattel AND
    there was not prior knowledge of the existence of the chattel

EXAMPLE Corporation of London v Yorkwin PROVIDES
the finder does not win ONLY IF
    the finder was the occupier of the premises AND
    the chattel was attached AND
    the non-finder was the owner of the real estate AND
    the non-finder was not the owner of the chattel AND
    there was a bailment of the chattel AND
    there was a term in a lease which mentioned found items AND
    there was not a master-servant relationship between the
```

```
    parties AND
    the chattel was hidden AND
    there was an attempt to find the true owner of the chattel AND
    there was not prior knowledge of the existence of the chattel

EXAMPLE Moffatt v Kazana PROVIDES
the finder does not win ONLY IF
    the finder was the occupier of the premises AND
    the chattel was not attached AND
    the non-finder was not the owner of the real estate AND
    the non-finder was the owner of the chattel AND
    there was not a bailment of the chattel AND
    there was not a term in a lease which mentioned found items AND
    there was not a master-servant relationship between the
    parties AND
    the chattel was hidden AND
    there was an attempt to find the true owner of the chattel AND
    there was prior knowledge of the existence of the chattel

EXAMPLE South Staffordshire Water Co v Sharman PROVIDES
the finder does not win ONLY IF
    the finder was not the occupier of the premises AND
    the chattel was attached AND
    the non-finder was the owner of the real estate AND
    the non-finder was not the owner of the chattel AND
    there was not a bailment of the chattel AND
    there was not a term in a lease which mentioned found items AND
    there was a master-servant relationship between the parties AND
    the chattel was hidden AND
    there was an attempt to find the true owner of the chattel AND
    there was not prior knowledge of the existence of the chattel

EXAMPLE Yorkwin v Appleyard PROVIDES
the finder does not win ONLY IF
    the finder was not the occupier of the premises AND
    the chattel was attached AND
    the non-finder was not the owner of the real estate AND
    the non-finder was not the owner of the chattel AND
    there was not a bailment of the chattel AND
    there was not a term in a lease which mentioned found items AND
    there was a master-servant relationship between the parties AND
    the chattel was hidden AND
    there was an attempt to find the true owner of the chattel AND
    there was not prior knowledge of the existence of the chattel
```

An example session for the above code is as follows:[84]

```
 1) What is the name of the finder?
    ** Alan Parker

 2) Is Alan Parker a person?
    ** yes

 3) What is Alan Parker's preferred gender?
    ** male

 4) What is Alan Parker's preferred form of address?
    ** Mr

 5) Was Mr Parker the occupier of the premises?
    ** no

 6) Was the chattel attached?
    ** no

 7) What is the name of the non-finder?
    ** British Airways

 8) Is British Airways a person?
    ** no

 9) Was British Airways the owner of the real estate?
    ** yes

10) Was British Airways the owner of the chattel?
    ** no

11) Was there a bailment of the chattel?
    ** no

12) Was there a term in a lease which mentioned found items?
    ** no

13) Was there a master-servant relationship between the parties?
    ** no

14) Was the chattel hidden?
    ** no

15) Was there an attempt to find the true owner of the chattel?
    ** yes

16) Was there prior knowledge of the existence of the chattel?
    ** no

    REPORT

    Mr Parker wins because the situation is similar to Bridges v
    Hawkesworth and South Staffordshire Water Co v Sharman can
    be distinguished.

    The situation is similar to Bridges v Hawkesworth because:
    Mr Parker was not the occupier of the premises; the chattel
    was not attached; British Airways was the owner of the real
    estate; and the chattel was not hidden.

    South Staffordshire Water Co v Sharman can be distinguished
    because the chattel was not attached and there was not a
    master-servant relationship between the parties.
```

Chapter 8: Document Assembly

yscript provides two document assembly mechanisms: *documents* and *templates*. Using either approach, documents are generated in imperative fashion but can take advantage of the rest of yscript to determine what needs to be included or to select an appropriate document type or format. *Documents* are generated using statements which generate text, optionally in Markdown format.[85] *Templates* use yscript fact values to modify external documents which are generally in word-processing format such as `DOCX`.[86]

When a document has been assembled, it will be displayed at the end of the user session along with the report. Where there are several documents, these can be presented as *attachments*. An attachment is part of a fact declaration that indicates that a document, template, or report should be made available to the user whenever a value for the fact is concluded.

8.1 Documents

Documents are generated from rules of the type - `DOCUMENT`. Document rules are never automatically invoked and must either by a goal rule or called using a `CALL` or `NEXT` statement. The only difference between a `DOCUMENT` rule and a `PROCEDURE` is that document rules may use the statements - `LINE`, `PARAGRAPH` and `TEXT` to write text to a document. The `LINE` statement ends the output line with a new-line and the `PARAGRAPH` statement ends the line with two new-lines. The `TEXT` statement just outputs the raw text with no new line. The syntax of these statements is:

```
[NUMBERED] [LEVEL number] [PARAGRAPH|LINE|TEXT] text
```

where *`text`* is an arbitrary piece of text to be output as part of the document being generated.

For the most part, *`text`* will be used literally except that it may contain the values of facts by including these in curly brackets. If these facts are not used elsewhere, they will be automatically declared (as type `STRING`) and if it is not possible to determine a value for the fact then the system will ask the user. If you want a fact appearing in text to be of a different type to `STRING` (including for named subjects) then you need to formally declare it.

---

[85] Markdown is discussed in § 8.2 at page 91.

[86] The type of documents which can be used as templates depends on the template engine. The Jinja2 engine will work with DOCX and any text format such as RTF, Markdown and TEX. See § 8.7 at page 98.

For example:

```
INFORMAL PERSON the testator
DATE the date of execution of the Will

DOCUMENT Preamble PROVIDES
PARAGRAPH This will dated {the date of execution of the Will}
is made by me {the name of the testator}, of {the address of the testator},
{the occupation of the testator}.
```

will produce the following dialog and output:

```
1) What is the date of execution of the Will?
   ** today

2) What is the name of the testator?
   ** Gloria Huntingdale

3) What is Gloria Huntingdale's preferred gender?
   ** female

4) What is the address of Gloria Huntingdale?
   ** One, The Esplanade, Gold Coast, Australia

5) What is the occupation of Gloria Huntingdale?
   ** Mining Magnate

   This will dated 4th November, 2020 is made by me Gloria
   Huntingdale, of One, The Esplanade, Gold Coast, Australia,
   Mining Magnate.
```

Including the keyword `NUMBERED` before a `PARAGRAPH`, `LINE` or `TEXT` statement will number the paragraph. The `LEVEL` keyword will control the sub-level of numbering and will generate Markdown blockquote tags which are discussed in § 8.2.

If you want to include different text depending upon one or more fact values, you can use `IF-THEN` statements as in:

```
DOCUMENT Revocation PROVIDES
IF all former testamentary dispositions are to be revoked THEN
    NUMBERED PARAGRAPH I hereby revoke all former testamentary
    dispositions.
```

It is generally convenient to use multiple `DOCUMENT` rules to generate different parts of a document and then to `CALL` these as necessary.

For example:

```
DOCUMENT Last Will & Testament PROVIDES
    CALL Preamble
    CALL Revocation
    CALL Bequest
    CALL Execution
```

Following is a complete example of a simple Will generator:

```
INFORMAL PERSON the testator
STRING the name of the testator
PROMPT what is the name of the person making the Will
DATE the date of execution of the Will
DATE the date of the old Will

GOAL DOCUMENT Last Will & Testament PROVIDES
IF the testator should make a Will THEN BEGIN
    CALL Disclaimer
    CALL Preamble
    CALL Revocation
    CALL Contemplation of Marriage
    CALL Sole Beneficiary
    CALL Attestation
END

DOCUMENT Disclaimer PROVIDES
PARAGRAPH Disclaimer: This is not a real Will and must not be used as such.
This will does not purport to accurately represent the current or past law
of any jurisdictions.

DOCUMENT Preamble PROVIDES
PARAGRAPH This will dated {the date of execution of the Will} is made by me
{the name of the testator}, of {the testator's address},
{the testator's occupation}.

DOCUMENT Revocation PROVIDES
IF all former testamentary dispositions are to be revoked THEN
    NUMBERED PARAGRAPH I revoke all former testamentary dispositions.
ELSE
    NUMBERED PARAGRAPH I revoke all former testamentary dispositions except
    clause(s) {the list of clauses from the old will which are to be saved}
    of my testamentary instrument dated {the date of the old Will} which
    clause(s) I hereby confirm.

DOCUMENT Contemplation of Marriage PROVIDES
IF the Will is to be made in contemplation of marriage THEN BEGIN
    IF the Will is to be conditional on the marriage actually taking place
THEN
        NUMBERED PARAGRAPH This will is made in contemplation of my
        marriage with {the name of the testator's fiancée} and is
        conditional on the marriage taking place within
        {the maximum number of months within which the wedding must
        take place} months.
    ELSE
        NUMBERED PARAGRAPH This will is made in contemplation of my
marriage
        with {the name of the testator's fiancée} but is not conditional on
the
        marriage taking place.
END
```

```
DOCUMENT Sole Beneficiary PROVIDES
IF everything disposed of under the Will is to be left one person THEN
BEGIN
    IF the sole beneficiary is over 18 THEN
        NUMBERED PARAGRAPH I give the whole of my estate to
        {the name of the sole beneficiary} whom I appoint my sole executor.
    ELSE BEGIN
        NUMBERED PARAGRAPH I give the whole of my estate to
        {the name of the sole beneficiary}.
        NUMBERED PARAGRAPH I appoint {name of the executor and trustee} as
        my sole executor and sole trustee of my estate.
    END
END ELSE BEGIN
    NUMBERED PARAGRAPH I give the whole of my estate in equal shares to
    {the names of the joint beneficiaries}.
    NUMBERED PARAGRAPH I appoint the {the name of the executor} as my
    sole executor.
END

DOCUMENT Attestation PROVIDES
PARAGRAPH SIGNED by the testator in our presence and attested by us in the
presence of the testator and each other.
```

To sort out the issue as to whether a person should make a Will at all, you could add:

```
RULE Power to Make a Will PROVIDES
the testator should make a will ONLY IF
    the testator is over 18 years of age AND
    there is not any doubt about the testator's mental capacity to make a
    Will AND/OR
    the test set out in Banks v Goodfellow is met AND/OR
    the Will is to be approved by the Supreme Court under section 18 of the
    Succession Act 2006 OR
    the testator is not over 18 years of age AND
    the Will is to be made in contemplation of marriage AND
    there is not any doubt about the testator's mental capacity to make a
    Will AND/OR
    the test set out in Banks v Goodfellow is met OR
    the testator is not over 18 years of age AND
    the Will is to be approved by the Supreme Court under section 16 of the
    Succession Act 2006

RULE The Test in Banks v Goodfellow PROVIDES
the test set out in Banks v Goodfellow is met ONLY IF
    the testator can appreciate the effect of making a will AND
    the testator can recall the assets that make up the testator's estate
AND
    the testator can comprehend that there are people who are entitled to
    provisions from the deceased estate AND
    the testator does not suffer from a disorder that stops the testator
    from making rational decisions about the distribution of the testator's
```

When run, the following consultation will result:

```
 1) What is the name of the person making the Will?
    ** George Brown

 2) What is George Brown's preferred gender?
    ** m

 3) Is he over 18 years of age?
    ** y

 4) Is there any doubt about his mental capacity to make a Will?
    ** n

 5) What is the date of execution of the Will?
    ** today

 6) What is George Brown's address?
    ** 34 Pitt Street, Sydney

 7) What is his occupation?
    ** Nurse

 8) Are all former testamentary dispositions to be revoked?
    ** y

 9) Is the Will to be made in contemplation of marriage?
    ** n

10) Is everything disposed of under the Will to be left one person?
    ** y

11) Is the sole beneficiary over 18?
    ** y

12) What is the name of the sole beneficiary?
    ** Daisy Brown
```

The generated document will be:

```
Disclaimer: This is not a real Will and must not be used as
such. This Will does not purport to accurately represent the
current or past law of any jurisdictions.

This Will dated 20th June, 2021 is made by me George Brown,
of 34 Pitt Street, Sydney, Nurse.

1. I revoke all former testamentary dispositions.

2. I give the whole of my estate to Daisy Brown whom I
appoint my sole executor.

SIGNED by the testator in our presence and attested by us in
the presence of the testator and each other.
```

8.2 Markdown

Markdown is a lightweight text mark-up language which allows for simple text formatting. It was originally developed by John Gruber and Aaron Swartz in 2004 as a way of making documents "publishable as-is, as plain text, without looking like it's

been marked up with tags or formatting instructions".[87] A central aim is that text marked up in Markdown should remain humanly readable whilst being able to be converted into other formats.

yscript uses Markdown for all *text* blocks such as explanations, SAY statements and document output. You can use it to introduce basic formatting into documents. The following illustrates some of the basic Markdown elements:

```
Introduction to Markdown
========================

Paragraphs consist of one or more lines of text that
are separated by blank lines.

## Headings can be underlined as above or begin with between one and six
hashes to indicate the heading level.

Lists can be numbered or bulleted:
1. Numbered lists start with digits then a dot then a space;
2. Other list items continue. Words can be in *italics* or **bold**;
3. Until you are finished the list.

* Bullet lists are similar
* With items preceded by a star or a dash
> Blockquotes are preceded by a greater than.
> Like this. And can be nested:
>> Like this nested blockquote.
```

When rendered, this will look like:

Introduction to Markdown

Paragraphs consist of one or more lines of text that are separated by blank lines.

Headings can be underlined as above or begin with between one and six hashes to indicate the heading level.

Lists can be numbered or bulleted:

1. Numbered lists start with digits then a dot then a space;
2. Other list items continue. Words can be in *italics* or **bold**;
3. Until you are finished the list.

- Bullet lists are similar
- With items preceded by a star or a dash

> Blockquotes are preceded by a greater than. Like this. And can be nested:
>
> > Like this nested blockquote.

8.3 Including Markdown Text in yscript Code

Markdown text is inherently line-based. When including Markdown text in yscript code, you may still indent it with spaces or tabs to represent the format or control flow of the yscript code. Any left indent that follows the first newline in the block will

[87] John Grubber, *Markdown* < https://daringfireball.net/projects/markdown/>.

conceptually become a left margin for Markdown purposes. The first non-space character on this and subsequent lines in the text block will be treated as though this is the first character on the line.

For example, in the following code:

```
IF the sky is blue THEN
    PARAGRAPH
        1. it is a great day
        Today
```

the text block following the `PARAGRAPH` statement represents a Markdown numbered list. It is equivalent to:

```
IF the sky is blue THEN PARAGRAPH 1. it is a great day today
```

or

```
       IF the sky is blue THEN PARAGRAPH
1. It is a great day today
```

If you wish to indent some text to indicate (for example) a Markdown code block, you can do something like:

```
PARAGRAPH
This is an example of some code:
     This text will be indented.
This text will not be.
```

In yscript code, trailing spaces on a line (that is space characters after the last character of text but immediately before a newline) are ignored. The Markdown use of two trailing spaces to indicate a hard line-break is not supported.[88] If you want to introduce a hard end of line, finish the line with a slosh (`'\'`) or use the escape sequence `'\n'`.

For example:

```
PARAGRAPH
This is a line by itself\
So is this\n
This is the last line of the paragraph.

But all of
this text gets wrapped
as the next paragraph.
```

Apart from these subtleties, Markdown text will normally be passed through yscript unchanged and any idents and line-breaks that would cause changes to the meaning of the Markdown text are preserved.

[88] The reason for this is that giving meaning to two trailing spaces is likely to cause confusion for what is otherwise a free-form syntax.

8.4 Suggested Markdown Elements

It is up to the calling environment to render the Markdown text. There are many different extensions to the original Markdown format.[89] yscript is largely agnostic as to which of these you use and displays the Markdown text in its unchanged (but humanly readable) form. The DataLex environment uses a third-party *CommonMark* based library.

It is suggested that you confine your use to the following Markdown elements:

Paragraphs

A paragraph is a set of unindented lines. Paragraphs can be separated by a blank line (that is, a line containing only zero or more spaces or tabs followed by a newline). Normal paragraphs should not be indented. For example:

```
This is a paragraph. All
words in the paragraph will
be word-wrapped and left-aligned.
```

Lines

A line is a piece of text followed by a slosh ('\') and then a newline. As already stated, yscript does not support the two trailing spaces format from the original Markdown specification.

For example:

```
These are two\
separate lines.
```

Bold and Italics

Bold and italics can be indicated by surrounding text with two stars and one star respectively. For example, this is in **bold** and this is in *italics*. There cannot be a space between the stars and the text, i.e., this is ** incorrect and not in bold ** and * this is also incorrect and not in italics *.

Headings

Headings are introduced by preceding a line with between one and six hash characters. There should be a space after the last hash. For example:

---

[89] The most popular extensions are *GFM* (Git-Hub Flavoured Markdown), *CommonMark*, *Markdown Extra*, *MultiMarkdown* and *R Markdown*. These support extended syntax features such as tables, footnotes, definition lists, task lists and emojis.

```
# This is a level one heading
## This is a level two heading
###### This is a level six heading
```

Alternatively, headings can use underlining. Text underlined with equal signs ('=') represents a level one heading and text underlined with minus signs ('–') represents a level two heading. For example:

```
This is a level one heading
===========================

This is a level two heading
---------------------------
```

Numbered Lists

Numbered lists consist of one or more *paragraphs* beginning with a series of digits followed by a stop ('.') and then a space. The numbering of the paragraphs does not matter, but the first number should be one ('1. '). Where you want to include subsequent paragraphs at the same level as a numbered paragraph, this should be indented by four spaces.

For example:

```
1. This is the first paragraph in the numbered list.

    This paragraph follows on at the same level as the first
    numbered paragraph.

2. This is the second paragraph in the numbered list.
```

Unnumbered Lists

Unnumbered lists start with a star ('∗'), minus sign ('-') or plus sign ('+'), then a space and then text. Again, subsequent paragraphs can be included at the same level but indenting by four spaces. For example:

```
* This is the first item of an unnumbered list;
* this is the second item; and
* this is the third item.
```

Horizontal Rules

You can insert a horizontal rule but starting a line with three or more stars ('∗') or minus signs ('–'). Optionally, a space may be included between each star or minus sign. For example:

```
---

**************************************************

* * * * * * *
```

Blockquotes

A blockquote can be introduced by starting a line with one or more greater than signs ('>') followed by a space. The number of greater than signs indicate the level of nesting. A greater than sign may precede each line in a blockquote, or you can just place a greater than sign before the first line.[90] A blockquote can include other Markdown elements.

For example:

```
> This is an example of a blockquote
> that contains two lines and then a block quoted list:
>> 1. This is the first item in the numbered list.
>> 2. This is the second item in the numbered list.
> This is the last sentence of the original blockquote.
```

Code Blocks

Code blocks are used to represent pre-formatted (i.e., mono-spaced) text such as is used in computer code. In code blocks, all hard line-breaks and indenting are rendered. Code blocks can be introduced by indenting every line by at least four spaces or a tab or by *fencing* it within two lines starting with three back-ticks ('`').

For example:

```
```yscript
this yscript code will be pre-formatted ONLY IF
    it is not indented OR
    it is indented however you like
```

this yscript code will be pre-formatted ONLY IF
        it is indented by at least four spaces or a tab
```

Links

You can include HTML links by placing the link text in square brackets then the destination URL in round brackets. For example:

```
[link text](http://www.destination.url)
```

Images

Images can be included in text by starting a line with an exclamation mark ('!'), followed by a description of the image in square brackets and then the URL for the image in round brackets. For example:

```
![alt-image text](http://image.url.location/image.png)
```

---

[90] John Gruber refers to the latter approach as the *lazy* format. Placing a greater than sign before each line is designed to reflect the ordinary e-mail practice and is easier to read for raw (non-rendered) Markdown text.

8.5 Templates

Templates are files that are separate from any yscript code that are marked up in a templating language and processed by an external template processor. They are suitable for documents of any complexity and may make use of the word-processing or other features of their native format. Templates can use yscript facts to insert values such as names, addresses, and amounts as well as to determine whether to include material. When a template is to be generated, yscript passes the values of all relevant facts for use by the template processor.[91] The results can be *attached* to yscript facts as described in the next section.

8.6 Attachments

Attachments are fact sub-declarations that indicate that generated reports, documents, and templates should be brought to the users' attention when the value of a fact becomes known. Typically, this fact will be the ultimate or a major conclusion.

The attachment syntax is:

```
ATTACH [DISPLAYED] [qualifier] REPORT|DOCUMENT|TEMPLATE [name] [AS
description]
```

Where the `DISPLAYED` keyword is used, or where there is only a single attachment, the attachment will be immediately displayed when the fact becomes known. If it is not possible to display the attachment, the user will be prompted to save it.

If there is more than one non-displayed attachment, a list will be generated, and the user will be presented with a multi-choice question as to which documents they wish to view and potentially save.

The optional *`qualifier`* argument can be used to specify the template language. `The` permitted values are currently: `DataLex` or `Jinja2`.

The `REPORT`, `DOCUMENT` or `TEMPLATE` argument specifies what is being attached. Without a *name*, `REPORT` and `DOCUMENT` will display the default report and all documents respectively. Otherwise, a report or document will be produced for the specified rule name.

The *`description`* argument is used to describe the document in the multi-choice list or in any questions.

[91] The template processor can be invoked by either the yscript interpreter directly or by the calling environment.

For example:

```
FACT the disclosing party needs a non-disclosure agreement
ATTACH REPORT AS Record of Advice
ATTACH TEMPLATE nda.docx AS Draft Non-Disclosure Agreement
ATTACH DOCUMENT Letter to Client AS Letter to Client
```

will produce the necessary attached documents and present the dialog:

```
The following documents are available:
1) Record of Advice
2) Draft Non-Disclosure Agreement
3) Letter to Client

Please select:
**
```

8.7 Aliases

Fact names can be long and may not be convenient when used in templated documents. Such names can be *aliased* using the `ALIAS` declaration. The syntax is:

```
ALIAS name
```

8.8 Templating Languages and Formats

yscript currently only supports two templating languages: DataLex and Jinja2.

DataLex

The *DataLex* templating language has been developed specifically for the DataLex Development Environment by Philip Chung. It works with Microsoft Word `DOCX` format documents.

To insert yscript fact values into your document, copy and paste in the fact name, replace all spaces with underscores and then highlight it in yellow. Conditional text is highlighted in green and roughly follows yscript syntax.

Jinja2

Jinja is a template engine written by Armin Ronacher. The syntax is based on Django's[92] built-in web template engine. Jinja is used by several other systems including docassemble. [93] The most recent version of Jinja is *Jinja2* and this is how it is often referred to.

---

[92] Django is a is a Python-based free and open-source web framework. See < https://www.djangoproject.com>.

[93] docassemble is a free, open-source expert system for guided interviews and document assembly, based on Python, YAML, and Markdown written by Jonathan Pyle. See <http://docassemble.org>.

Jinja2 is an extremely flexible language,[94] and the processor can be used with any type of text-based file format (such as Markdown, TEX, XML or RTF) as well as Microsoft Word DOCX.

To include a yscript fact in a Jinja2 template document, replace all of the spaces in the fact name with underscores and include it between a pair of double curly braces.

For example:

```
Dear {{name_of_the_customer}},

I am writing to you in relation to your contract with
{{name_of_contracting_party}}. Please note that this agreement will finish
on {{the_contract_termination_date}}. Should you with to renew the
contract, please give us a call.

Yours sincerely,
```

To include conditional text, use the syntax:

```
{% if condition %}
text to be included if condition is met
{% else %}
Optional text to be included if condition is not met
{% endif %}
```

For example:

```
Dear {{name_of_the_customer}},

I am writing to you in relation to your contract with
{{name_of_contracting_party}}. Please note that this agreement will finish
on {{the_contract_termination_date}}.

{% if the_customer_is_a_valued_customer %}
We really value your business and we would like to set up a time to talk to
you about renewing the contract.
{% else %}
We thank you for your business and hope you find some other company to take
care of your needs in the future.
{% endif %}

Yours sincerely,
```

[94] A full description can be found at <https://jinja.palletsprojects.com/en/3.0.x/>.

Chapter 9: Style Guide

Writing *good* code can largely be a matter of personal taste, but like most languages yscript will naturally support some approaches to coding better than others. The following suggestions about style will make it easier to write code and make it simpler to maintain and extend the code. These approaches will also make it easier for other people to understand your code and to work collaboratively.

Simplicity

Try to aim for simplicity wherever possible. Complicated mechanical constructs and workarounds detract from the readability of the code and can have unexpected repercussions, particularly when code is later expanded or changed. Don't use language features simply because they are available.

Isomorphism

Where the code represents rules from a legal source document such as a piece of legislation, try to directly translate the statutory rules into yscript code, observing as far as possible the order and grouping of the rules, and adding as little interpretation as possible. Keep other rules, such as interpretation or 'common sense' which do not derive directly from the legislation, in a separate part of your code.

Small rules

Avoid large and complicated rules. Small rules are easier to understand and will assist with automatic explanations.

Fact Names

Include the basis for facts in descriptors, as in *the layout is in "material form" as defined in s.5*. This will make for more meaningful explanations. Avoid using unnecessarily long descriptors. These make for convoluted questions and explanations. Do not use the translation and prompt options unnecessarily. Try changing the fact name to get yscript to handle it properly, first. Avoid use of embedded facts.

Rule Types

Use only the default rule type unless you have a good reason for doing otherwise. Relying upon the order of rule execution can lead to code that is cryptic and difficult to understand.

Declarative Representation

Unless you are intentionally writing something that is procedural, avoid using `DETERMINE` and `CALL` statements except where unavoidable or when generating documents. When writing declarative code, avoid being concerned about the actual operation of rules and instead concentrate on *describing* what it is that you are trying to deal with.

Comments

Avoid using comments to repeat what is already in your code. The code should largely be transparent. Comments are helpful where there is something non-obvious or work remaining to be done.

Context

For large projects, divide the code into *context*s and avoid overly relying on the shared context for communication between contexts. As far as is possible, each context should be capable of running on a free-standing basis and its reliance or links to other contexts explicitly declared.

Appendix 1: Units, Currencies and Time Zones

This section contains a set of detailed tables outlining available units of measurement, currency units and time zones.

9.1 Numerical Units

Unit	Abbrev	Metric
acre	acre	
amp	A	*
angstrom	Å	
astronomical unit	au	
atmosphere	atm	*
bar	bar	*
barrel	bl	
becquerel	Bq	*
bel	B	
bit	b	*
British thermal unit	BTU	
bushel	bu	
byte	B	*
calorie	Cal	
candela	cd	*
chain	ch	
coulomb	C	*
cup	cup	
dalton	Da	*
day	day	
decibel	dB	
degree	°	
degree Celsius	°C	
degree Fahrenheit	°F	
dram	dr	
electronvolt	eV	*
farad	F	*
fathom	ftm	
fluid ounce	fl oz	
foot	ft	
furlong	fur	
gallon	gal	
grain	gr	
gram	g	
gravity	*g*	
gray	Gy	*
hectare	ha	*
henry	H	*
hertz	Hz	*
hour	hr	
imp cup	imp cup	
imp tablespoon	imp tbsp	
imp teaspoon	imp tsp	
imperial barrel	imp bl	
imperial bushel	imp bu	
imperial fluid ounce	imp fl oz	
imperial gallon	imp gal	
imperial pint	imp pt	
imperial quart	imp qt	
imperial ton	Imp ton	
inch	in	
joule	J	*
katal	kat	*
kelvin	K	
knot	knot	
litre	l	*
lumen	lm	*
lux	lx	*
metre	m	*
mile	mi	
minute	min	
mole	mol	*
month	mth	
nautical mile	nmi	
neper	Np	
newton	N	*
ohm	Ω	*
ounce	oz	
pascal	Pa	*
pennyweight	pwt	
percent	%	
pint	pt	
pound	lb	
quart	qt	
radian	rad	
second	s	*
siemens	S	*
sievert	Sv	*
steradian	sr	
stone	st	
tablespoon	tbsp	
teaspoon	tsp	
tesla	T	*
ton	ton	*
tonne	t	*
us cup	us cup	
us tablespoon	us tbsp	
us teaspoon	us tsp	
volt	V	*
watt	W	*
weber	Wb	*
week	wk	
yard	yd	
year	yr	

Figure 25 – Numerical Units

9.2 Derived Units

Derived Unit	Abbreviation	Metric
bits per second	bps	*
bytes per second	Bps	*
coulombs per cubic metre	C/m³	*
coulombs per kilogram	C/kg	*
coulombs per square metre	C/m²	*
cubic centimetres	cm³	
cubic feet	cu ft	
cubic inches	cu in	
cubic kilometres	km³	
cubic metres	m³	
cubic millimetres	mm³	
cubic yards	cu yd	
farads per metre	F/m	*
grams per metre square	g/m²	*
gray per second	Gy/s	*
hectopascals	hPa	*
henrys per metre	H/m	*
inches of mercury	inHg	
joules per cubic metre	J/m³	*
joules per kelvin	J/K	*
joules per kilogram	J/kg	*
joules per mole	J/mol	*
katals per cubic metre	kat/m³	*
metres per hour	m/h	*
metres per litre	m/l	*
metres per second	m/s	*
metres per second squared	m/s²	*
miles per gallon	mpg	
miles per hour	mph	
moles per cubic metre	mol/m³	*
newton metres	N m	*
newtons per metre	N/m	*
pascal seconds	Pa s	*
pounds per square inch	psi	
radians per second	rad/s	
radians per second squared	rad/s²	
reciprocal metres	m⁻¹	
revolutions per minute	rpm	
square centimetres	cm²	
square feet	sq ft	
square inches	sq in	
square kilometres	km²	
square metres	m²	
square miles	sq mi	
square millimetres	mm²	
square yards	sq yd	
torr	mmHg	
volts per metre	V/m	*
watt hours	W h	*
watts per square metre	W/m²	*
watts per steradian	W/sr	*

Figure 26 - Derived Units

9.3 Unit Synonyms

Units may be referred to by the following synonyms. By default, abbreviating and multiplicative dots and stops are ignored. Non-SI units are not case specific. Units with negative exponents will match equivalent units using a divisor. Exponents can be expressed as simple digits (that is, `10m2` is treated the same as `10m²`).

Synonym	Unit
baud	bits per second
candle	candela
candlepower	candela
cc	cm³
Celsius	degrees Celsius
centigrade	Celsius
cu cm	cm³
cu foot	cu ft
cu inch	cu in
cu m	m³
cu yard	cu yd
cui	cu in
degC	Celsius
degF	Fahrenheit
degK	Kelvin
degree Kelvin	Kelvin
displacement ton	imp ton
drachm	dram
dwt	pwt
Fahrenheit	degree Fahrenheit
fermi	femtometre
ft²	sq ft
ft³	cu ft
gamma	nanotesla
gramme	gram
in²	sq in
in³	cu in
inche	inch
J/s	W
kg m/s²	N
km per hour	km/h
km per hr	km/h
km/hour	km/h
km/hr	km/h
kmph	km/h
kms/hr	km/h
kph	km/h
KWh	kW h
kWh	kW h
liter	litre
long ton	imp ton
m/s/s	m/s²
meter	metre
metric ton	tonne
mho	siemen
mi/h	mph
mi/hr	mph
mi/hr	mph
mi²	sq mi
micron	micrometre
millimicron	nanometre
N m	J
N/m²	Pa
r/min	rpm
rev/min	rpm
revs/min	rpm
short ton	ton
sq cm	cm²
sq foot	sq ft
sq inch	sq in
sq km	km²
sq m	m²
sq mile	sq mi
sq yard	sq yd
us barrel	barrel
us bl	bl
us bu	bu
us bushel	bushel
us fl oz	fl oz
us fluid ounce	fluid ounce
us gal	gal
us gallon	gallon
us pint	pint
us pt	pt
us qt	qt
us quart	quart
us ton	ton
w/t	imp ton
weight ton	imp ton
yd2	sq yd
yd³	cu yd

Figure 27 - Unit Synonyms

9.4 Currency Units

Currency	Abb	Sym
Afghan afghani	AFN	؋
Albanian lek	ALL	L
Alderney pound		£
Algerian dinar	DZD	DA
Angloan kwanza	AOA	Kz
Argentine peso	ARS	$
Armenian dram	AMD	֏
Artsakh dram		դր.
Aruban florin	AWG	ƒ
Australian dollar	AUD	$
Azerbaijani manat	AZN	₼
Bahamian dollar	BSD	$
Bahraini dinar	BHD	د.ب.
Bangladeshi taka	BDT	৳
Barbadian dollar	BBD	BBD$
Belarusian ruble	BYN	Br
Belize dollar	BZD	$
Bermudian dollar	BMD	$
Bhutanese ngultrum	BTN	Nu.
Bitcoin	BTC	₿
Bolivian boliviano	BOB	Bs.
Botswana pula	BWP	P
Brazilian real	BRL	R$
British pound	GBP	£
Brunei dollar	BND	$
Bulgarian leva	BGN	лв.
Burmese kyat	MMK	Ks
Burundian franc	BIF	Fr
CFP franc	XPF	₣
Cambodian riel	KHR	៛
Canadian dollar	CAD	$
Cape Verdean escudo	CVE	Esc
Cayman Islands dollar	KYD	$
Central African franc	XAF	Fr
Chilean peso	CLP	$
Chinese yuan	CNY	元
Colombian peso	COP	$
Comorian franc	KMF	Fr
Congolese franc	CDF	Fr
Cook Islands dollar	CKD	$
Costa Rican colón	CRC	₡
Croatian kuna	HRK	kn
Cuban peso	CUP	$
Czech koruna	CZK	Kč
Danish kroner	DKK	kr
Djiboutian franc	DJF	Fr
Dominican peso	DOP	RD$
Eastern Caribbean dollar	XCD	$
Egyptian pound	EGP	£
Eritrean nakfa	ERN	Nfk
Ethiopian birr	ETB	Br
European Euro	EUR	€
Falkland Islands pound	FKP	£
Faroese króna	FOK	kr
Fijian dollar	FJD	$
Gambian dalasi	GMD	D
Georgia lari	GEL	₾
Ghanaian cedi	GHS	₵
Gibraltar pound	GIP	£
Guatemalan quetzal	GTQ	Q
Guernsey pound	GGP	£
Guinean franc	GNF	Fr
Guyanese dollar	GYD	$
Haitian gourde	HTG	G
Honduran lempira	HNL	L
Hong Kong dollar	HKD	HK$
Hungarian forint	HUF	Ft
Icelandic króna	ISK	kr
Indian rupee	INR	₹
Indonesian rupiah	IDR	Rp
Iranian rial	IRR	﷼
Iraqi dinar	IQD	د.ع
Israeli new shekel	ILS	₪
Jamaican dollar	JMD	$
Japanese yen	JPY	¥
Jersey pound	JEP	£
Jordanian dinar	JOD	د.ا
Kazakhstani tenge	KZT	₸
Kenyan shilling	KES	Sh
Kiribati dollar	KID	$
Kuwaiti dinar	KWD	د.ك
Kyrgyzstani som	KGS	c
Lao kip	LAK	₭
Lebanese pound	LBP	ل.ل
Lesotho loti	LSL	L
Liberian dollar	LRD	LRD$
Libyan dinar	LYD	ل.د
Macanese pataca	MOP	$
Macedonian denar	MKD	ден
Malagasy ariary	MGA	Ar
Malawian kwacha	MWK	MK
Malaysian ringgit	MYR	RM
Maldivian rufiyaa	MVR	.ރ
Manx pound	IMP	£
Mauritanian ouguiya	MRU	UM
Mauritian rupee	MUR	₨
Mexican peso	MXN	MXN$
Moldovan leu	MDL	L
Mongolian tögrög	MNT	₮
Moroccan dirham	MAD	د.م
Mozambican metical	MZN	MT
Namibian dollar	NAD	$
Nepalese rupee	NPR	₹
Netherlands Ant guilder	ANG	ƒ
New Taiwan dollar	TWD	$
New Zealand dollar	NZD	$
Nicaraguan córdoba	NIO	$
Nigerian naira	NGN	₦
Niue dollar		$
North Korean won	KPW	₩
Norwegian kroner	NOK	kr
Omani rial	OMR	ر.ع
Pakistani rupee	PKR	₨
Panamanian balboa	PAB	B/.
Papua New Guinean kina	PGK	K
Paraguayan guaraní	PYG	₲
Peruvian sol	PEN	S/.
Philippine peso	PHP	₱
Pitcairn Islands dollar	PND	$
Polish złoty	PLN	zł
Qatari riyal	QAR	ر.ق

RTGS dollar	ZWB	
Romanian leu	RON	lei
Russian ruble	RUB	₽
Rwandan franc	RWF	Fr
Sahrawi peseta		₧
Saint Helena pound	SHP	£
Samoan tālā	WST	T
Saudi riyal	SAR	﷼
Serbian dinar	RSD	дин
Seychellois rupee	SCR	₨
Sierra Leonean leone	SLL	Le
Singapore dollar	SGD	$
Solomon Islands dollar	SBD	$
Somali shilling	SOS	Sh
Somaliland shilling	SLS	Sl
South African rand	ZAR	R
South Korean won	KRW	₩
South Sudanese pound	SSP	£
Sri Lankan rupee	LKR	Rs
Sudanese pound	SDG	ج.س
Surinamese dollar	SRD	SRD$
Swazi lilangeni	SZL	L
Swedish krona	SEK	kr
Swiss franc	CHF	Fr.
Syrian pound	SYP	£
Tajikistani somoni	TJS	c.
Tanzanian shilling	TZS	Sh
Thai baht	THB	฿
Tongan paʻanga	TOP	$
Transnistrian ruble	PRB	p.
Trinidad & Tobago dollar	TTD	$
Tunisian dinar	TND	د ت
Turkish lira	TRY	₺
Turkmenistan manat	TMT	m.
Tuvaluan dollar	TVD	$
Ugandan shilling	UGX	Sh
Ukrainian hryvnia	UAH	₴
UAE dirham	AED	د.إ
United States dollar	USD	$
Uruguayan peso	UYU	$
West African CFA franc	XOF	Fr
Yemeni rial	YER	﷼
Zambian kwacha	ZMW	ZK

Figure 28 - Currency Units

9.5 Time Zone Styles

The following style names / time zone abbreviations can be used as the `STYLE` for facts of type `TIME`. You can also use specify a time zone relative to `UTC` (Coordinated Universal Time) or `GMT` (Greenwich Mean Time), for example: `UCT+10` or `GMT-1`. Note that time zones are not formally defined and that they sometimes can mean more than one thing.

Abbrev	Time Zone	Time Diff
ACDT	Australian Central Daylight-Saving Time	UTC+10:30
ACST	Australian Central Standard Time	UTC+09:30
ACT	Acre Time	UTC-05
ACWST	Australian Central Western Standard Time	UTC+08:45
ADT	Atlantic Daylight Time	UTC-03
AEDT	Australian Eastern Daylight Savings Time	UTC+11
AEST	Australian Eastern Standard Time	UTC+10
AFT	Afghanistan Time	UTC+04:30
AKDT	Alaska Daylight Time	UTC-08
AKST	Alaska Standard Time	UTC-09
ALMT	Alma-Ata Time	UTC+06
AMST	Amazon Summer Time (Brazil)	UTC-03
AMT	Amazon Time (Brazil)	UTC-04
AMT	Armenia Time	UTC+04
ANAT	Anadyr Time	UTC+12
AQTT	Aqtobe Time	UTC+05
ART	Argentina Time	UTC-03
AST	Atlantic Standard Time	UTC-04
AWST	Australian Western Standard Time	UTC+08
AZOST	Azores Summer Time	UTC±00
AZOT	Azores Standard Time	UTC-01
AZT	Azerbaijan Time	UTC+04
BNT	Brunei Time	UTC+08
BIOT	British Indian Ocean Time	UTC+06
BIT	Baker Island Time	UTC-12
BOT	Bolivia Time	UTC-04
BRST	Brasília Summer Time	UTC-02
BRT	Brasília Time	UTC-03
BST	British Summer Time[95]	UTC+01
BTT	Bhutan Time	UTC+06
CAT	Central Africa Time	UTC+02
CCT	Cocos Islands Time	UTC+06:30
CDT	Central Daylight Time	UTC-05
CEST	Central European Summer Time	UTC+02
CET	Central European Time	UTC+01
CHADT	Chatham Daylight Time	UTC+13:45
CHAST	Chatham Standard Time	UTC+12:45
CHOT	Choibalsan Standard Time	UTC+08
CHOST	Choibalsan Summer Time	UTC+09
CHST	Chamorro Standard Time	UTC+10
CHUT	Chuuk Time	UTC+10
CIST	Clipperton Island Standard Time	UTC-08
CKT	Cook Island Time	UTC-10
CLST	Chile Summer Time	UTC-03
CLT	Chile Standard Time	UTC-04
COST	Colombia Summer Time	UTC-04
COT	Colombia Time	UTC-05
CST	Central Standard Time	UTC-06
CVT	Cape Verde Time	UTC-01
CWST	Central Western Standard Time	UTC+08:45
CXT	Christmas Island Time	UTC+07
DAVT	Davis Time	UTC+07
DDUT	Dumont d'Urville Time	UTC+10
EASST	Easter Island Summer Time	UTC-05
EAST	Easter Island Standard Time	UTC-06
EAT	East Africa Time	UTC+03
ECT	Ecuador Time	UTC-05

[95] `BST` is also sometimes used to mean Bougainville Standard Time. Use `UCT+11` instead.

EDT	Eastern Daylight Time	UTC−04
EEST	Eastern European Summer Time	UTC+03
EET	Eastern European Time	UTC+02
EGST	Eastern Greenland Summer Time	UTC±00
EGT	Eastern Greenland Time	UTC−01
EST	Eastern Standard Time	UTC−05
FET	Further-eastern European Time	UTC+03
FJT	Fiji Time	UTC+12
FKST	Falkland Islands Summer Time	UTC−03
FKT	Falkland Islands Time	UTC−04
FNT	Fernando de Noronha Time	UTC−02
GALT	Galápagos Time	UTC−06
GAMT	Gambier Islands Time	UTC−09
GET	Georgia Standard Time	UTC+04
GFT	French Guiana Time	UTC−03
GILT	Gilbert Island Time	UTC+12
GIT	Gambier Island Time	UTC−09
GMT	Greenwich Mean Time	UTC±00
GST	South Georgia and the South Sandwich Islands Time	UTC−02
GST	Gulf Standard Time	UTC+04
GYT	Guyana Time	UTC−04
HDT	Hawaii–Aleutian Daylight Time	UTC−09
HAEC	Heure Avancée d'Europe Centrale	UTC+02
HST	Hawaii–Aleutian Standard Time	UTC−10
HKT	Hong Kong Time	UTC+08
HMT	Heard and McDonald Islands Time	UTC+05
HOVST	Hovd Summer Time	UTC+08
HOVT	Hovd Time	UTC+07
ICT	Indochina Time	UTC+07
IDLW	International Day Line West time zone	UTC−12
IDT	Israel Daylight Time	UTC+03
IOT	Indian Ocean Time	UTC+03
IRDT	Iran Daylight Time	UTC+04:30
IRKT	Irkutsk Time	UTC+08
IRST	Iran Standard Time	UTC+03:30
IST	Indian Standard Time[96]	UTC+05:30
JST	Japan Standard Time	UTC+09
KALT	Kaliningrad Time	UTC+02
KGT	Kyrgyzstan Time	UTC+06
KOST	Kosrae Time	UTC+11
KRAT	Krasnoyarsk Time	UTC+07
KST	Korea Standard Time	UTC+09
LHST	Lord Howe Standard Time	UTC+10:30
LHST	Lord Howe Summer Time	UTC+11
LINT	Line Islands Time	UTC+14
MAGT	Magadan Time	UTC+12
MART	Marquesas Islands Time	UTC−09:30
MAWT	Mawson Station Time	UTC+05
MDT	Mountain Daylight Time	UTC−06
MET	Middle European Time	UTC+01
MEST	Middle European Summer Time	UTC+02
MHT	Marshall Islands Time	UTC+12
MIST	Macquarie Island Station Time	UTC+11
MIT	Marquesas Islands Time	UTC−09:30
MMT	Myanmar Standard Time	UTC+06:30
MSK	Moscow Time	UTC+03
MST	Malaysia Standard Time	UTC+08
MST	Mountain Standard Time	UTC−07
MUT	Mauritius Time	UTC+04
MVT	Maldives Time	UTC+05
MYT	Malaysia Time	UTC+08
NCT	New Caledonia Time	UTC+11
NDT	Newfoundland Daylight Time	UTC−02:30
NFT	Norfolk Island Time	UTC+11
NOVT	Novosibirsk Time	UTC+07
NPT	Nepal Time	UTC+05:45
NST	Newfoundland Standard Time	UTC−03:30

[96] Note that `IST` can also mean Irish Standard Time and Israel Standard Time. Use instead `UTC+1` and `UTC+2` respectively.

NT	Newfoundland Time	UTC−03:30
NUT	Niue Time	UTC−11
NZDT	New Zealand Daylight Time	UTC+13
NZST	New Zealand Standard Time	UTC+12
OMST	Omsk Time	UTC+06
ORAT	Oral Time	UTC+05
PDT	Pacific Daylight Time	UTC−07
PET	Peru Time	UTC−05
PETT	Kamchatka Time	UTC+12
PGT	Papua New Guinea Time	UTC+10
PHOT	Phoenix Island Time	UTC+13
PHT	Philippine Time	UTC+08
PHST	Philippine Standard Time	UTC+08
PKT	Pakistan Standard Time	UTC+05
PMDT	Saint Pierre and Miquelon Daylight Time	UTC−02
PMST	Saint Pierre and Miquelon Standard Time	UTC−03
PONT	Pohnpei Standard Time	UTC+11
PST	Pacific Standard Time	UTC−08
PWT	Palau Time	UTC+09
PYST	Paraguay Summer Time	UTC−03
PYT	Paraguay Time	UTC−04
RET	Réunion Time	UTC+04
ROTT	Rothera Research Station Time	UTC−03
SAKT	Sakhalin Island Time	UTC+11
SAMT	Samara Time	UTC+04
SAST	South African Standard Time	UTC+02
SBT	Solomon Islands Time	UTC+11
SCT	Seychelles Time	UTC+04
SDT	Samoa Daylight Time	UTC−10
SGT	Singapore Time	UTC+08
SLST	Sri Lanka Standard Time	UTC+05:30
SRET	Srednekolymsk Time	UTC+11
SRT	Suriname Time	UTC−03
SST	Samoa Standard Time	UTC−11
SST	Singapore Standard Time	UTC+08
SYOT	Showa Station Time	UTC+03
TAHT	Tahiti Time	UTC−10
THA	Thailand Standard Time	UTC+07
TFT	French Southern and Antarctic Time	UTC+05
TJT	Tajikistan Time	UTC+05
TKT	Tokelau Time	UTC+13
TLT	Timor Leste Time	UTC+09
TMT	Turkmenistan Time	UTC+05
TRT	Turkey Time	UTC+03
TOT	Tonga Time	UTC+13
TVT	Tuvalu Time	UTC+12
ULAST	Ulaanbaatar Summer Time	UTC+09
ULAT	Ulaanbaatar Standard Time	UTC+08
UTC	Coordinated Universal Time	UTC±00
UYST	Uruguay Summer Time	UTC−02
UYT	Uruguay Standard Time	UTC−03
UZT	Uzbekistan Time	UTC+05
VET	Venezuelan Standard Time	UTC−04
VLAT	Vladivostok Time	UTC+10
VOLT	Volgograd Time	UTC+04
VOST	Vostok Station Time	UTC+06
VUT	Vanuatu Time	UTC+11
WAKT	Wake Island Time	UTC+12
WAST	West Africa Summer Time	UTC+02
WAT	West Africa Time	UTC+01
WEST	Western European Summer Time	UTC+01
WET	Western European Time	UTC±00
WIB	Western Indonesian Time	UTC+07
WIT	Eastern Indonesian Time	UTC+09
WITA	Central Indonesia Time	UTC+08
WGST	West Greenland Summer Time	UTC−02
WGT	West Greenland Time	UTC−03
WST	Western Standard Time	UTC+08
YAKT	Yakutsk Time	UTC+09

Figure 29 - Time Zones

9.6 Time Zone Country Names

Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmara
Africa/Bamako
Africa/Bangui
Africa/Banjul
Africa/Bissau
Africa/Blantyre
Africa/Brazzaville
Africa/Bujumbura
Africa/Cairo
Africa/Casablanca
Africa/Ceuta
Africa/Conakry
Africa/Dakar
Africa/Dar_es_Salaam
Africa/Djibouti
Africa/Douala
Africa/El_Aaiun
Africa/Freetown
Africa/Gaborone
Africa/Harare
Africa/Johannesburg
Africa/Juba
Africa/Kampala
Africa/Khartoum
Africa/Kigali
Africa/Kinshasa
Africa/Lagos
Africa/Libreville
Africa/Lome
Africa/Luanda
Africa/Lubumbashi
Africa/Lusaka
Africa/Malabo
Africa/Maputo
Africa/Maseru
Africa/Mbabane
Africa/Mogadishu
Africa/Monrovia
Africa/Nairobi
Africa/Ndjamena
Africa/Niamey
Africa/Nouakchott
Africa/Ouagadougou
Africa/Porto-Novo
Africa/Sao_Tome
Africa/Tripoli
Africa/Tunis
Africa/Windhoek
America/Adak
America/Anchorage
America/Anguilla
America/Antigua
America/Araguaina
America/Argentina/Buenos_Aries
America/Argentina/Catamarca
America/Argentina/Cordoba
America/Argentina/Jujuy
America/Argentina/La_Rioja
America/Argentina/Mendoza
America/Argentina/Rio_Gallegos
America/Argentina/Salta
America/Argentina/San_Juan
America/Argentina/San_Luis
America/Argentina/Tucuman
America/Argentina/Ushuaia
America/Aruba
America/Asuncion
America/Atikokan
America/Bahia
America/Bahia_Banderas
America/Barbados
America/Belem
America/Belize
America/Blanc-Sablon
America/Boa_Vista
America/Bogota
America/Boise
America/Cambridge_Bay
America/Campo_Grande
America/Cancun
America/Caracas
America/Cayenne
America/Cayman
America/Chicago
America/Chihuahua
America/Costa_Rica
America/Creston
America/Cuiaba
America/Curacao
America/Danmarkshavn
America/Dawson
America/Dawson_Creek
America/Denver
America/Detroit
America/Dominica
America/Edmonton
America/Eirunepe
America/El_Salvador
America/Fort_Nelson
America/Fortaleza
America/Glace_Bay
America/Goose_Bay
America/Grand_Turk
America/Grenada
America/Guadeloupe
America/Guatemala
America/Guayaquil
America/Guyana
America/Halifax
America/Havana
America/Hermosillo
America/Indiana/Indianapolis
America/Indiana/Knox
America/Indiana/Marengo
America/Indiana/Petersburg
America/Indiana/Tell_City
America/Indiana/Vevay
America/Indiana/Vincennes
America/Indiana/Winamac
America/Inuvik
America/Iqaluit
America/Jamaica
America/Juneau
America/Kentucky/Louisville
America/Kentucky/Monticello
America/Kralendijk
America/La_Paz
America/Lima
America/Los_Angeles
America/Lower_Princes
America/Maceio
America/Managua
America/Manaus
America/Marigot
America/Martinique
America/Matamoros
America/Mazatlan
America/Menominee
America/Merida
America/Metlakatla
America/Mexico_City
America/Miquelon
America/Moncton
America/Monterrey
America/Montevideo
America/Montserrat
America/Nassau
America/New_York
America/Nipigon
America/Nome
America/Noronha
America/North_Dakota/Beulah
America/North_Dakota/Center
America/North_Dakota/New_Salem
America/Nuuk
America/Ojinaga
America/Panama
America/Pangnirtung
America/Paramaribo
America/Phoenix
America/Port_of_Spain
America/Port-au-Prince
America/Porto_Velho
America/Puerto_Rico
America/Punta_Arenas
America/Rainy_River
America/Rankin_Inlet
America/Recife
America/Regina
America/Resolute
America/Rio_Branco
America/Santarem
America/Santiago
America/Santo_Domingo
America/Sao_Paulo
America/Scoresbysund
America/Sitka
America/St_Barthelemy
America/St_Johns
America/St_Kitts
America/St_Lucia
America/St_Thomas
America/St_Vincent
America/Swift_Current
America/Tegucigalpa
America/Thule
America/Thunder_Bay
America/Tijuana
America/Toronto
America/Tortola
America/Vancouver
America/Whitehorse
America/Winnipeg
America/Yakutat
America/Yellowknife
Antarctica/Casey
Antarctica/Davis
Antarctica/DumontDUrville
Antarctica/Macquarie
Antarctica/Mawson
Antarctica/McMurdo
Antarctica/Palmer
Antarctica/Rothera
Antarctica/Syowa
Antarctica/Troll
Antarctica/Vostok

Arctic/Longyearbyen	Asia/Tehran	Europe/Riga
Asia/Aden	Asia/Thimphu	Europe/Rome
Asia/Almaty	Asia/Tokyo	Europe/Samara
Asia/Amman	Asia/Tomsk	Europe/San_Marino
Asia/Anadyr	Asia/Ulaanbaatar	Europe/Sarajevo
Asia/Aqtau	Asia/Urumqi	Europe/Saratov
Asia/Aqtobe	Asia/Ust-Nera	Europe/Simferopol
Asia/Ashgabat	Asia/Vientiane	Europe/Skopje
Asia/Atyrau	Asia/Vladivostok	Europe/Sofia
Asia/Baghdad	Asia/Yakutsk	Europe/Stockholm
Asia/Bahrain	Asia/Yangon	Europe/Tallinn
Asia/Baku	Asia/Yekaterinburg	Europe/Tirane
Asia/Bangkok	Asia/Yerevan	Europe/Ulyanovsk
Asia/Barnaul	Atlantic/Azores	Europe/Uzhgorod
Asia/Beirut	Atlantic/Bermuda	Europe/Vaduz
Asia/Bishkek	Atlantic/Canary	Europe/Vatican
Asia/Brunei	Atlantic/Cape_Verde	Europe/Vienna
Asia/Chita	Atlantic/Faroe	Europe/Vilnius
Asia/Choibalsan	Atlantic/Madeira	Europe/Volgograd
Asia/Colombo	Atlantic/Reykjavik	Europe/Warsaw
Asia/Damascus	Atlantic/South_Georgia	Europe/Zagreb
Asia/Dhaka	Atlantic/St_Helena	Europe/Zaporozhye
Asia/Dili	Atlantic/Stanley	Europe/Zurich
Asia/Dubai	Australia/Adelaide	Indian/Antananarivo
Asia/Dushanbe	Australia/Brisbane	Indian/Chagos
Asia/Famagusta	Australia/Broken_Hill	Indian/Christmas
Asia/Gaza	Australia/Darwin	Indian/Cocos
Asia/Hebron	Australia/Eucla	Indian/Comoro
Asia/Ho_Chi_Minh	Australia/Hobart	Indian/Kerguelen
Asia/Hong_Kong	Australia/Lindeman	Indian/Mahe
Asia/Hovd	Australia/Lord_Howe	Indian/Maldives
Asia/Irkutsk	Australia/Melbourne	Indian/Mauritius
Asia/Istanbul	Australia/Perth	Indian/Mayotte
Asia/Jakarta	Australia/Sydney	Indian/Reunion
Asia/Jayapura	Europe/Amsterdam	Pacific/Apia
Asia/Jerusalem	Europe/Andorra	Pacific/Auckland
Asia/Kabul	Europe/Astrakhan	Pacific/Bougainville
Asia/Kamchatka	Europe/Athens	Pacific/Chatham
Asia/Karachi	Europe/Belgrade	Pacific/Chuuk
Asia/Kathmandu	Europe/Berlin	Pacific/Easter
Asia/Khandyga	Europe/Bratislava	Pacific/Efate
Asia/Kolkata	Europe/Brussels	Pacific/Enderbury
Asia/Krasnoyarsk	Europe/Bucharest	Pacific/Fakaofo
Asia/Kuala_Lumpur	Europe/Budapest	Pacific/Fiji
Asia/Kuching	Europe/Busingen	Pacific/Funafuti
Asia/Kuwait	Europe/Chisinau	Pacific/Galapagos
Asia/Macau	Europe/Copenhagen	Pacific/Gambier
Asia/Magadan	Europe/Dublin	Pacific/Guadalcanal
Asia/Makassar	Europe/Gibraltar	Pacific/Guam
Asia/Manila	Europe/Guernsey	Pacific/Honolulu
Asia/Muscat	Europe/Helsinki	Pacific/Kiritimati
Asia/Nicosia	Europe/Isle_of_Man	Pacific/Kosrae
Asia/Novokuznetsk	Europe/Istanbul	Pacific/Kwajalein
Asia/Novosibirsk	Europe/Jersey	Pacific/Majuro
Asia/Omsk	Europe/Kaliningrad	Pacific/Marquesas
Asia/Oral	Europe/Kiev	Pacific/Midway
Asia/Phnom_Penh	Europe/Kirov	Pacific/Nauru
Asia/Pontianak	Europe/Lisbon	Pacific/Niue
Asia/Pyongyang	Europe/Ljubljana	Pacific/Norfolk
Asia/Qatar	Europe/London	Pacific/Noumea
Asia/Qostanay	Europe/Luxembourg	Pacific/Pago_Pago
Asia/Qyzylorda	Europe/Madrid	Pacific/Palau
Asia/Riyadh	Europe/Malta	Pacific/Pitcairn
Asia/Sakhalin	Europe/Mariehamn	Pacific/Pohnpei
Asia/Samarkand	Europe/Minsk	Pacific/Port_Moresby
Asia/Seoul	Europe/Monaco	Pacific/Rarotonga
Asia/Shanghai	Europe/Moscow	Pacific/Saipan
Asia/Singapore	Europe/Nicosia	Pacific/Tahiti
Asia/Srednekolymsk	Europe/Oslo	Pacific/Tarawa
Asia/Taipei	Europe/Paris	Pacific/Tongatapu
Asia/Tashkent	Europe/Podgorica	Pacific/Wake
Asia/Tbilisi	Europe/Prague	Pacific/Wallis

Figure 30 - Time Zone Country Names

Appendix 2: The cyscript Interpreter

This appendix describes *cyscript* – the default command line interpreter. This is a Unix/Linux command line tool that provides a simple environment to run and test yscript code. The code for the interpreter interface itself (ys.c) is also an example of how to use the yscript API.[97]

1. Usage

cyscript is invoked with the command `ys`. The command takes zero or more files containing yscript code as arguments. Where no source file is provided, cyscript will read from standard input. The files conventionally end with the suffix .ys and `cyscript` will check for the presence of a file with this ending before looking for the bare filename.

The formal usage is:

```
ys [-cdgmpstvx] [--facts=facts] [--goal=rule] [ file[.ys] ] ...
```

Available Flags

The flags have the following meanings:

`-c`, `--check` — This option parses all input files but does not run them. This is useful for checking to see if there are any syntax errors or warnings.[98]

`-d`, `--debug` — The `-d` flag turns on verbose mode which reports system actions such as when rules as fired or blocked, when a fact is explicitly being determined or forgotten and when fact values are concluded or changed.

`-f file`, `--facts=file` — The `-f` indicates that once the code files have been parsed and loaded, that the facts and default goal should be set to those contained in the specified `file`. The format[99] of the `file` is the same as the one used by the interactive `load` and `save` commands.

---

[97] cyscript can also be placed on a socket and used as a server. This is covered in Appendix 6: Using cyscript as a Server on page 157.

[98] There is a list of all error and warning messages below on page 127.

[99] This file is stored in JSON format.

`-g rule`,
`--goal=rule`
: The `-g` flag sets the default goal rule to a rule either matching the specified `rule` name or (if there is no such rule) the first rule that contains `rule` as a substring.

`-h, --help`
: The help flag displays a quick summary of command usage and flags.

`-j, --json`
: This flag will cause all output to be in *JSON* format.[100] This is intended to support applications that use the interpreter in an automated way. It makes sure that all output is always flushed and changes the prompt to ∗−∗ on a line by itself.

`-n, --showfacts`
: The `-n` flag writes the names of all facts that are used in any of the input source files. These are written one per line. It is intended to support external applications that need to interact with yscript using fact values.

`-p, --pretty`
: The `-p` flag outputs a pretty-printed copy of the yscript code on standard output.

`-s, --statistics`
: The `-s` flag outputs some statistics about the yscript code. These include the total number of rules, facts, tokens and characters.

`-t, --translations`
: The `-t` flag outputs the translations used for all facts. This includes the prompt that will be used to get a value from the user and the fact in positive and negative form.

`-T, --templates=dir`
: This flag specifies the base directory for all document template[101] files. When set, only files located under this directory can be used as templates.

`-v, --version`
: The `-v` flag prints the version number for interpreter and library.

`-x, --xref`
: The `-x` flag prints a fact cross reference that shows which rules use the fact and which rules conclude a value for the fact.

---

[100] See Appendix 6: Using cyscript as a Server on page 157.

[101] For discussion of document templates, see § 8.5 on page 97.

`-y, --yaml` The `-y` flag causes all output to be in *YAML*[102] format. Like the `-j` flag, this also causes cyscript to operate in server mode without pagination and with regular output flushing.

Code Reformatting

The pretty printer (`-p`) will output code in standard recommended format. This is mainly intended for tidying up code and making it consistent, but it is also useful occasionally for tracking down obscure bugs. For example, the following input:

```
PERSON the person
RULE South African Citizenship Act 1995 PROVIDES
the person is a South African citizen ONLY IF
   the person is a citizen by birth under section 2 of
  the South African Citizenship Act 1995 OR the person is a citizen
by descent under section 3 of the South African Citizenship Act 1995 OR the
person is a citizen by naturalisation under section 4
                of the South African Citizenship Act 1995 RULE South
African Citizenship Act 1995 Section 2 PROVIDES
the person
is
a citizen by birth under section 2 of
the South African Citizenship Act 1995 ONLY IF section 2(1) of the
South African Citizenship Act 1995 applies OR section 2(2) of
the South African Citizenship Act 1995 applies OR section 2(3)
of the South African Citizenship Act 1995 applies
```

will be reformatted as:

```
PERSON the person

RULE South African Citizenship Act 1995 PROVIDES
the person is a South African citizen ONLY IF
    the person is a citizen by birth under section 2 of the South
    African Citizenship Act 1995 OR
    the person is a citizen by descent under section 3 of the South
    African Citizenship Act 1995 OR
    the person is a citizen by naturalisation under section 4 of the
    South African Citizenship Act 1995

RULE South African Citizenship Act 1995 Section 2 PROVIDES
the person is a citizen by birth under section 2 of the South African
Citizenship Act 1995 ONLY IF
    section 2(1) of the South African Citizenship Act 1995 applies OR
    section 2(2) of the South African Citizenship Act 1995 applies OR
    section 2(3) of the South African Citizenship Act 1995 applies
```

[102] Discussion of YAML output is in Appendix 6: Using cyscript as a Server at page 157.

Statistics

The `-s` flag produces simple statistics about yscript code. For example:

```
Rules:          93
Facts:         445
Tokens:        858
Characters:  32968
```

Translations

The `-t` flag produces a list of the translations used for all facts including the forms that will be used to ask for a value and, as appropriate, its positive and negative form or the form used to report a non-boolean value. Where the default generated translations have been overridden (via a `TRANSLATE` or `PROMPT` declaration), these translations are used.

This facility is useful because it allows you to see how every fact will be presented to the application user without having to run through all fact scenarios. The output format is first the fact as a question, then the fact in positive form and then the fact in negative form (or just the fact with a value for non-boolean facts).

For example:

```
- Does Part III of the Act apply?
- Part III of the Act applies.
- Part III of the Act does not apply.

- Is S8(1)(a) satisfied?
- S8(1)(a) is satisfied.
- S8(1)(a) is not satisfied.

- Did a person copy the circuit layout in a material form?
- A person copied the circuit layout in a material form.
- A person did not copy the circuit layout in a material form.

- Has the copy of the circuit layout or an integrated circuit made
  in accordance with the circuit layout been sold?
- The copy of the circuit layout or an integrated circuit made in
  accordance with the circuit layout has been sold.
- The copy of the circuit layout or an integrated circuit made in
  accordance with the circuit layout has not been sold.

- What is the date of infringement?
- The date of infringement is {}.
```

Note that the standard distribution also includes a utility called `trans` which can also be useful in understanding how propositions have been interpreted. It is a filter that reads a line which should contain a proposition and it outputs how the proposition has been divided, why and gives the default translations.

For example:

```
% trans
Aristotle remarks that falsity and truth require combination and separation

aristotle/remarks/that falsity and truth require combination and separation
[defined-verb]

- Does Aristotle remark that falsity and truth require combination and
separation?
- Aristotle remarks that falsity and truth require combination and
separation.
- Aristotle does not remark that falsity and truth require combination and
separation.

The notion of a proposition can also be found in the works of Medieval
philosophers

the notion of a proposition/can/also be found in the works of medieval
philosophers [skip-noun|auxiliary-verb]

- Can the notion of a proposition also be found in the works of Medieval
philosophers?
- The notion of a proposition can also be found in the works of Medieval
philosophers.
- The notion of a proposition cannot also be found in the works of Medieval
philosophers.
```

Cross-referencing

The –x flag produces a fact cross-reference. It shows a list of all facts along with the names of rules where a fact is used (preceded by a hyphen) and rules where a value for the fact is determined (preceded by an Asterix). Fact names are the glue that binds rules together and it is important that they match when they should. An example cross-reference follows:

```
section 2(1) of the South African Citizenship Act 1995 applies
    * South African Citizenship Act 1995 Section 2(1)
    - South African Citizenship Act 1995 Section 2
    - South African Citizenship Act 1995 Section 2(2)(a)

section 2(1)(a) of the South African Citizenship Act 1995 applies
    * South African Citizenship Act 1995 Section 2(1)(a)
    - South African Citizenship Act 1995 Section 2(1)

the person was a South African citizen immediately prior to the date of
commencement of the South African Citizenship Amendment Act, 2010
    * Citizenship Amendment Act, 2010 Commencement
    - South African Citizenship Act 1995 Section 2(1)(a)

the person was born in the republic of South Africa
    - South African Citizenship Act 1995 Section 2(2)(a)
    - South African Citizenship Act 1995 Section 2(3)
    - South African Citizenship Act 1995 Section 4(3)
```

Return Code

ys returns `0` on success, and non-zero on error. Possible errors include usage errors, missing or invalid file names, syntax errors and run-time errors. For a full list, see *Error and Warning Messages* below.

2. Interactive Sessions

When the cyscript is started, the first thing it does is to load and parse the yscript code file(s). If any syntax or other errors are encountered, one or more error messages or warnings are issued. If a fact file has been specified via the `-f` flag, the included goal is set, and facts are loaded. If a goal has been declared via the `-g` flag, this is set as the default goal.

Selecting a Goal

If no goal has been set and the code contains more than one listed `GOAL`, the user is asked which goal rule they wish to use.

For example:

```
The following goals are defined:

 1) Community Gaming Regulation 2020 Regulation 4 – Art union
    gaming activities
 2) Community Gaming Regulation 2020 Regulation 5 – Housie or bingo
 3) Community Gaming Regulation 2020 Regulation 6 – Draw lotteries
 4) Community Gaming Regulation 2020 Regulation 7 – No-draw lotteries
 5) Community Gaming Regulation 2020 Regulation 8 – Mini-numbers
    lotteries
 6) Community Gaming Regulation 2020 Regulation 9 – Progressive
    lotteries
 7) Community Gaming Regulation 2020 Regulation 10 – Free lotteries
 8) Community Gaming Regulation 2020 Regulation 11 – Promotional
    raffles conducted by registered clubs
 9) Community Gaming Regulation 2020 Regulation 12 – Other gaming
    activities for charitable purposes
10) Community Gaming Regulation 2020 Regulation 13 – Sweeps and
    calcuttas
11) Community Gaming Regulation 2020 Regulation 14 – Trade
    promotion gaming activities

    Please select a goal?
    **
```

Questions and Prompts

Execution begins by *firing* the goal rule. Rule evaluation proceeds until the value of a fact is needed from the application user. The system then issues the default explanation for the fact (if any) and prompts for a fact value. Ordinarily, a simple prompt is used, and the user is expected to respond with an answer of a type that is appropriate for the fact in issue. There are also several standard responses such as `quit` to end the session and `help` to get a list of available cyscript commands. If the user

types in something that can't be understood or that is otherwise invalid, the system issues an error and re-asks the question.

For example:

```
 1) What is the name of the nominee?
    ** Paul Dutton

 2) Is he an Australian citizen?
    ** no idea

    Please respond with yes or no.

 3) Is Peter Dutton an Australian citizen?
    **
```

Similarly, if the fact has an associated range and the answer is out of range, a warning will be issued and the question re-asked.

For example:

```
 9) What is the amount of the gross proceeds of the AustLII
    Christmas Lottery?
    ** 30000

10) What is the amount paid to AustLII?
    ** 40000

    The value must be between $0 to $30,000.
```

Where the fact has a set of discrete possible range values, the prompt will be in multiple choice form. For example:

```
1) What is the type of gaming activity that you are proposing?
   (a) Art union
   (b) Housie or bingo
   (c) Draw lottery (including raffles and guessing competitions)
   (d) No-draw lottery (including break-open lotteries, scratch
       lotteries and football doubles)
   (e) Mini-numbers lottery
   (f) Progressive lottery (including a hundred club, silver
       Circles and tipping competitions)
   (g) Free lottery (including lucky door or lucky seat promotions)
   (h) Promotional raffle conduct by a registered club
   (i) Other gaming activity (including a chocolate wheel or
       Lucky envelopes) for charitable purposes
   (j) Sweep or calcutta
   (k) Trade promotion gaming activity (including card jackpot games)

   Please select?
   **
```

Uncertain Responses

If the user is uncertain of an answer or where they specifically wish to say that the answer to a question is uncertain or they do not wish to specify, they can type `u`, `unknown`, `uncertain`, or `unspecified`.

A blank answer for a question involving facts of type `STRING` will also be interpreted as an unspecified response. This behaviour is important for the answering of questions related to named subjects. It means, for example, that if the user hits enter in response to the name of a named subject, follow-up questions will be supressed, and the description of the named subject will not be replaced in subsequent dialogs.

For example:

```
1) What is the name of the taxpayer?
   **

2) Is the taxpayer an Australian citizen?
   **
```

It also means that the user can skip answering gender questions or questions in relation to the preferred form of address. For example:

```
1) What is the name of the taxpayer?
   ** Gretta Simpson

2) What is Gretta Simpson's preferred gender?
   **

3) What is Gretta Simpson's preferred form of address?
   **

4) Is Gretta Simpson an Australian citizen?
   **
```

What If?

The user can test what will happen if they answer in a particular way by prefacing their answer with the term `whatif` or the words `what if`. Provided that the proposed answer is a legal response, the system will return the conclusions that would be reached or if none would be reached, what the next question will be.

For example:

```
7) Has Google volunteered to comply with the requirements of the Act?
   ** what if no

   The following conclusions will be drawn:
      - Google has not volunteered to comply with the requirements
        of the Act under section 6 for the reporting period.
      - Section 5(1)(d) does not apply.
      - Section 5(1) does not apply.
      - Google is not a reporting entity.
      - The conditions set out in section 13(1) are not met.
      - Google has not submitted a modern slavery statement under
        section 13.
      - Section 16(2)(a) applies.
      - Section 16(2) applies.
      - Google has complied with the Act.

7) Has Google volunteered to comply with the requirements of the Act?
   ** what if yes

   No immediate conclusions will be drawn. The system will go on to ask:
      Has Google given written notice to the Minister in a manner and
      form approved by the Minister?
```

Why?

The `why` command can be used to explain *why* a question is being asked. Where the current question is about a related fact such as a named subject or a string used in an associated explanation, translation, or document, then the response will be to say that answer will be used when referring to the fact or named subject.

For example:

```
1) What is the name of the client?
   ** why

   This will be used when referring to the client.

2) What is the client's address
   ** why

   This will be used when referring to the client's address.
```

Where the current question results from an attempt to infer a value for another fact, the first time the command is issued, the response will be to say that the fact on the top of the goal fact stack is being determined. If the command is re-issued, the system backs up through the stack to use the next goal fact and so on. The effect is to successively explain what a question is being ask and then why it is necessary to know the value of the fact forming the basis of the explanation.

For example:

```
3) Did Telstra have a consolidated revenue of at least $100 million
   for the reporting period?
   ** why

   This will help determine whether or not section 5(1) applies.

3) Did Telstra have a consolidated revenue of at least $100 million
   for the reporting period?
   ** why

   This will help determine whether or not Telstra is a reporting
   entity.

3) Did Telstra have a consolidated revenue of at least $100 million
   for the reporting period?
   ** why

   This will help determine whether or not Telstra has complied with
   the Act.

3) Did Telstra have a consolidated revenue of at least $100 million
   for the reporting period?
   ** why

   There are no other facts under consideration.
```

When the `why` command is followed by the word `this` (ie `why this`), only the top level goal will be displayed. For example:

```
3) Did Telstra have a consolidated revenue of at least $100 million
   for the reporting period?
   ** why this

   This will help determine whether or not section 5(1) applies.
```

What?

The `what` command displays a list of all facts which have been supplied by the user (that is, it displays a list of premises). Apart from being useful in and of itself, this also provides numbers that can be used with the `forget` command.

For example:

```
1) What is the name of the entity?
   ** Telstra

2) Is Telstra an individual?
   ** no

3) Did Telstra have a consolidated revenue of at least $100 million
   for the reporting period?
   ** yes

4) Was Telstra a company during the reporting period?
   ** yes

5) Was Telstra formed or incorporated in Australia?
   ** what

1) The name of the entity is Telstra.
2) Telstra is not an individual.
3) Telstra had a consolidated revenue of at least $100 million for
   the reporting period.
4) Telstra was a company during the reporting period.
```

So?

The `so` command displays a list of all conclusions that have been reached. The numbers for each fact can be used with the how command.

For example:

```
   Was Telstra formed or incorporated in Australia?
   ** so

1) Telstra was a body corporate during the reporting period.
2) Section 5(2)(b) does not apply.
3) Telstra was not a trust where the trust estate was a resident
   trust estate within the meaning of Division 6 of Part III of the
   Income Tax Assessment Act 1936 during the reporting period.
4) Subsection (b) of the definition of "Australian entity" does not
   apply.
5) Telstra was not a corporate limited partnership which was a
   resident within the meaning of section 94T of the Income Tax
   Assessment Act 1936 during the reporting period.
6) Subsection (c) of the definition of "Australian entity" does not
   apply.
7) Telstra was not a partnership during the reporting period.
8) Telstra was not a non-corporate Commonwealth entity within the
   meaning of the Public Governance, Performance and Accountability
   Act 2013.
```

How

The how command can be used to explain how a particular conclusion was reached. It lists the facts that were used in determining a value for the fact. For example:

```
5) Was Telstra formed or incorporated in Australia?
** how 2

Section 5(2)(b) does not apply because Telstra was a body
Corporate during the reporting period.
```

Forget

The `forget` command is used to *forget* the value of a user-supplied fact or premise. The value of the fact is set to `unknown` and the system is restarted. The effect of this is to usually to immediately re-ask for a value of the fact that was just forgotten. If instead of an individual fact number, the user type `forget all`, the value of all facts and conclusions will be forgotten and the session will restart.

For example:

```
1) What is the name of the entity?
   ** Australia Post

2) Is Australia Post an individual?
   ** no

3) Did Australia Post have a consolidated revenue of at least $100
   million for the reporting period?
   ** yes

4) Was Australia Post a company during the reporting period?
   ** what

1) The name of the entity is Australia Post.
2) Australia Post is not an individual.
3) Australia Post had a consolidated revenue of at least $100
   million for the reporting period.

4) Was Australia Post a company during the reporting period?
   ** forget 3

3) Did Australia Post have a consolidated revenue of at least $100
   million for the reporting period?
   ** what

1) The name of the entity is Australia Post.
2) Australia Post is not an individual.

3) Did Australia Post have a consolidated revenue of at least $100
   million for the reporting period?
   ** forget all

1) What is the name of the entity?
   **
```

Goals

At the start of a session, if there is more than one goal rule (that is, more than one rule is declared with the qualifier `GOAL`), cyscript will ask the user to choose from a list of the available goal rules. At any point in the session, the user can also type `goals`, and this will show the same set of goals and give the user an opportunity to keep the current facts, but to pursue a different goal rule. If there is only one goal rule, or if the user types `goals all`,[103] the system will display a list of all available rules regardless of whether or not they have been formally declared as a `GOAL`. If the user wishes to

[103] The user can also type `rules all` and this will have the same effect.

pursue a particular goal and they know the rule name, they can type `goal` *`rule-name`* where *rule-name* is an entire rule name or part thereof.

For example:

```
   The following goals are defined:

     1) Modern Slavery Act 2018 (Cth) Section 4 "Australian entity"
     2) Modern Slavery Act 2018 (Cth) Section 5
     3) Modern Slavery Act 2018 (Cth) Section 12
     4) Modern Slavery Act 2018 (Cth)

   Please select a goal?
   ** 2


1) What is the name of the entity?
   ** Queensland Rail

2) Is Queensland Rail an individual?
   ** n

3) Did Queensland Rail have a consolidated revenue of at least $100
   million for the reporting period?
   ** goals

   The following goals are defined:

     1) Modern Slavery Act 2018 (Cth) Section 4 "Australian entity"
     2) Modern Slavery Act 2018 (Cth) Section 5
     3) Modern Slavery Act 2018 (Cth) Section 12
     4) Modern Slavery Act 2018 (Cth)

   Please select a goal?
   **
```

Rules

The command `rules all` is equivalent to `goals all` (and will display a list of all rule names and allow the user to select one as the current goal). The command `rule` *`rule-name`* is like `goal` *`rule-name`* but searches all rules for one matching *rule-name*. Typing rule by itself will display a list of rules that are currently being considered.

For example:

```
1) What is the name of the entity?
   ** ANZ

2) Is ANZ an individual?
   ** n

3) Did ANZ have a consolidated revenue of at least $100 million for
   the reporting period?
   ** y

4) Was ANZ a company during the reporting period?
   ** rules

   Modern Slavery Act 2018 (Cth)
    Modern Slavery Act 2018 (Cth) Section 5
     Modern Slavery Act 2018 (Cth) Section 5(1)
```

```
Modern Slavery Act 2018 (Cth) Section 5(1)(a)
 Modern Slavery Act 2018 (Cth) Section 5(1)(a)(i)
  Modern Slavery Act 2018 (Cth) Section 4 "Australian entity"
   Income Tax Assessment Act 1936 Section 6 "resident" (1)(b)
```

Typing just the command `rule` will display the complete text of the current rule that is being evaluated. In a similar fashion to the `why` command, issuing more than once will step back through the rule stack.

For example:

```
3) Did ANZ have a consolidated revenue of at least $100 million for
   the reporting period?
   ** rule

   RULE Modern Slavery Act 2018 (Cth) Section 5(1)(a) PROVIDES
       section 5(1)(a) applies ONLY IF
           the entity had a consolidated revenue of at least $100
           million for the reporting period AND
           section 5(1)(a)(i) applies AND/OR
              section 5(1)(a)(ii) applies
```

Where you know the name of the rule you wish to display, you can also always display any rule with the command `show rule` *`rule-name`*, for example:

```
show rule section 5(1)
```

Verbose Mode

cyscript can be put into *verbose* mode either by invoking it with the `-d` (or `--debug`) flag or by typing the command `verbose`. Verbose mode can be toggled off and on by typing `verbose` again. The effect of verbose mode is to display system actions.[104] For example:

```
1) What is the name of the entity?
   ** verbose

1) What is the name of the entity?
   ** Sydney Water

* DETERMINED VALUE FOR the name of the entity
* FORWARD-CHAINING FOR the name of the entity
 * FIRING Entity is Commonwealth of Australia

 2) Is Sydney Water an individual?
    ** no

* DETERMINED VALUE FOR the entity is a person
* FORWARD-CHAINING FOR the entity is a person
 * FIRING Individuals and Companies are not trusts
```

[104] There is a list of verbose messages in the following section on page 135.

Load and Save

The save and load commands will save the current goal and facts to and from a file (respectively). For example:

```
2) Is BHP an individual?
   ** save myfacts

2) Is BHP an individual?
   ** load myfacts
```

3. Error and Warning Messages

yscript generates various types of messages: parsing errors, session messages, verbose messages, fatal errors, and internal errors. This section explains what each message means. Each group is sorted for ease of look-up.

Parsing Errors

Parsing errors can be produced when a new or changed piece of code is first read by the interpreter. They indicate that something is syntactically or semantically wrong in the source file. Each message is preceded by a file name and an approximate line number as to where the error was encountered. For example:

```
gaming.ys: 23: expected factor before end of expression
```

Only the first error for each block (that is, fact declaration or rule) is displayed. The complete list of error and warning messages is as follows:

`alias redeclared`

A fact has been declared with more than one document-assembly alias (via the `ALIAS` sub-declaration).

`AND or AND/WITH argument(s) not boolean`
`AND/OR argument(s) not boolean`

One or more of the arguments to an `AND` or `AND/OR` operator in an *expression* does not return a boolean result.

`arithmetic expression has incompatible units`
`arithmetic term has incompatible units`
`assignment to '%s' involves incompatible units`

An expression or an assignment involves values that have units that cannot be used together. For example, it is illegal to add a weight to a length or a time to volume.

`attempted assignment to constant`

The left-hand-side of an assignment is a constant and not a fact.

bad multi-assignment

This error occurs when parsing assertions forming part of a multiple assignment. It probably indicates a trailing AND keyword that is missing a following fact name.

bad statement

The parser was expecting the start of a new statement such as an assignment or an IF–THEN–ELSE. The error generally indicates a malformed expression on the previous line.

bad WEEKS/MONTHS/YEARS type

The WEEKS, MONTHS and YEARS post-unary operators can only be applied to a number. The factor preceding one of these keywords is producing a non-numerical result.

DAY/MONTH/YEAR applied to non-date

When used as a pre-unary operator, the WEEKS, MONTHS and YEARS operators can only be applied to a factor yielding a DATE.

declaration hides shared fact

This is only a warning message. It is warning you that you have formally declared a fact, which will hide access to a fact with the name in the shared context. If this was intended, then you can safely ignore this message.

declaration of boolean expressed in negative

This is warning message indicating that you have used a proposition that is expressed in the negative when declaring a fact. The actual fact name will be changed to convert this into positive form.

default not last CASE condition

The default case must be the last WHEN condition in a CASE statement.

expected ALL or fact name after FORGET

The FORGET keyword is not followed by the keyword ALL or the name of a fact to be forgotten.

expected block start

The parser was expecting to see a new rule or fact definition. This will generally appear as the first error message and usually indicates that you are using a word-processor and have saved the file in binary format or that the file is not yscript code at all.

expected CASE before WHEN

The parser has encountered a WHEN condition without being preceded by a CASE. It is only legal to skip the CASE/fact specification as the first statement in following a FACT (object) declaration.

expected constant after EXPLAIN
expected constant after TRANSLATE
expected constant after WHEN

This error generally indicates that an EXPLAIN, TRANSLATE or WHEN keyword has been followed by a fact name rather than a constant.

expected context
expected context name

This error occurs in either a fact declaration or a CALL statement where you have indicated that you are referring to something from another context with the FROM keyword, but you are missing the name of the context itself. Deleting the FROM keyword will fix this.

expected description after AS

This error indicates that you have not included a text description following the AS keyword.

expected example name not constant

You have attempted to use a constant instead of a name for an EXAMPLE.

expected fact name after CASE

The CASE keyword must be followed by a fact name. This error indicates either a missing fact name or attempted use of a constant.

expected fact name after DETERMINE

The DETERMINE statement requires the name of a fact. Again, this might indicate that you have attempted to determine the value of a constant.

expected factor

This error message indicates a malformed expression. The parser was expecting to see a factor (that is, a fact or a constant) and instead has found something else (for example, a keyword).

expected factor before end of expression

This error is like the previous message but indicates that the parser ran out of expression when it was trying to parse a factor. A typical way to generate this message is to put a trailing operator (for example, an AND) on the end of an otherwise valid expression.

expected GOAL, RULE, or rule type

This error indicates a malformed RULE declaration. It is normally generated if you attempt to make something other than a rule (e.g., a fact) a goal.

expected number after LEVEL

A PARAGRAPH, LINE or TEXT statement is malformed. You have used the LEVEL keyword without following it by a literal number in the range 1 to 7.

expected rule name not constant

A rule name cannot start with a number or be some other constant.

expected RULE or rule type

You have started introducing a new rule with the GOAL keyword, then failed to follow it with the RULE keyword or a rule type.

expected STYLE after DEFAULT DATE
expected STYLE after DEFAULT NUMBER
expected STYLE after DEFAULT TIME

The parser was expecting the keyword STYLE after a DEFAULT declaration.

expected text after SAY

The SAY keyword must be followed by the text to be displayed. This can be within or without quotes.

expected UNIT after DEFAULT MONEY

The parser was expected the sequence DEFAULT MONEY UNIT *unit*. You have possibly attempted to assign a MONEY STYLE which is currently unsupported.

expected WHEN after CASE

You have attempted to use a CASE statement with no WHEN conditions. This probably indicates a syntax error immediately after the WHEN fact name.

fact already has associated rule

You have attempted to add two or more sets of statements to a fact / object.

guard is not boolean

The condition for an IF statement does not return a boolean result. You may need to formally declare some of the facts involved in the conditional expression.

illegal assignment to 'fact-name'

The right-hand side of an assignment (that is the *expression* after the ONLY IF or IS) returns a value which is of an incompatible type with the target fact. Again, this only happens after an attempt has been made to coerce the type of facts involved in the expression and so you made need to formally declare and type facts.

illegal example body

Examples do not share the same syntax as other rules. They can only consist of a single IF–THEN–ELSE *statement* or an *assignment*. Read the documentation about examples in Chapter 7.

inappropriate terms for DIVIDE
inappropriate terms for MINUS
inappropriate terms for MOD
inappropriate terms for PLUS
inappropriate terms for TIMES

You have attempted to perform a mathematical function on a non-numeric value. The DIVIDE, MINUS, MOD, PLUS and TIMES binary operators can only be used with the following exceptions: STRINGs can be concatenated with PLUS; and numbers can be added to and subtracted from DATEs.

inappropriate terms for relational operator

A relational operator is a comparative operator such as LESS THAN, GREATER THAN or EQUALS. The message indicates that the arguments cannot sensibly be compared. You may need to formally declare fact types.

LEVEL argument too large

The number following the level portion of the PARAGRAPH, LINE or TEXT statement must be between 1 and 7.

missing ALIAS descriptor

You have used the ALIAS sub-declaration within a fact but have not followed it by a fact descriptor.

missing context name

You have started a new context with the CONTEXT keyword, but you have not given it a name.

missing fact description after DETERMINE

The DETERMINE keyword must be followed by the name of a fact.

missing default date format
missing default money unit
missing default number style
missing default time format

A DEFAULT declaration is missing the actual unit or style.

missing END

The parser was expecting the keyword END. This probably indicates a mis-matched BEGIN–END pair.

missing EXPLAIN text

The text block for an EXPLAIN sub-declaration is missing.

missing fact name

You have started formally declaring a fact by using a type name (e.g. BOOLEAN, STRING etc) and you have not followed it by a fact name.

missing factor

This error message indicates a malformed expression. The parser was expecting to see a *factor* (that is, a fact name or constant) and instead it found a keyword that is not a pre-unary operator.

missing include file name

You are missing the file name after an INCLUDE keyword. This may be optionally in quotes.

missing INFO text

An INFO sub-declaration is not followed by a text block.

missing keyword AS after EXPLAIN
missing keyword AS after TRANSLATE
missing keyword DO
missing keyword END
missing keyword IF after ONLY
missing keyword PROVIDES
missing keyword THEN
missing keyword UNTIL

The indicated keyword is missing.

missing PARAGRAPH, LINE or TEXT

You have used NUMBERED or LEVEL and have not then gone on to use PARAGRAPH, LINE or TEXT.

missing PROMPT text

You have used the PROMPT keyword within a formal fact declaration, but you have not followed it by the prompt text.

missing rule name after CALL
missing rule name after NEXT

The name of the rule to call has been omitted after a CALL or NEXT statement.

missing STYLE type

You have not included a style type following the STYLE keyword.

missing text

The parser has found a PARAGRAPH, TEXT, or LINE statement, but this is not followed by any text to write.

missing TRANSLATE text

You are missing the translation text after a TRANSLATE declaration.

missing UNIT type

You have not included a unit type following the UNIT keyword.

named subjects can't have translations or ranges

You have attempted to provide a translation or range for a named subject which is illegal. You can translate the related facts such as the name, gender, and preferred address of the named subject.

no rules!

You have passed yscript a syntactically correct (or empty) file which doesn't declare any rules. The interpreter has nothing to do.

non-boolean assignment to 'fact-name'

You have attempted to assign a non-boolean value to a boolean fact. This may indicate that the fact should be formally declared as something other than BOOLEAN.

NOT value not boolean
OR argument(s) not Boolean

The argument(s) to a NOT or OR operator are not boolean and cannot be coerced to be so.

out of expression space

You have used an *expression* which is too complex for the system to handle. Try dividing the expression into smaller units (and preferably distribute it between several different rules).

rule undefined

You have used CALL to call a rule that you have not declared.

rule/range too large/complex

The rule body or range has produced too many tokens for the code-generator to deal with at once. Try dividing the contents between more rules.

SECOND/MINUTE/HOUR applied to non-time

You have attempted to use one of the pre-unary operators – `SECOND`, `MINUTE` or `HOUR` to a value which is not of the type - `TIME`.

style redeclared

You have attempted to declare the style for a fact.

too many WHEN conditions for CASE

There is a hard limit of 4096 when conditions for any individual `CASE` statement.

unit redeclared

You have attempted to redeclare the unit for a fact.

unknown currency - '%s'

You have used an unknown currency unit. See § 3.7 at page 29 for a list of units.

unknown default number style - '%s'
unknown number style - '%s'

You have used an unknown numbering style. See Figure 20 – Numerical Styles on page 23.

unknown unit - '%s'

You have used an unknown unit name. See § 3.7 at page 29 for a list of units.

UNTIL guard is not boolean
WHILE guard is not Boolean

The condition to a `REPEAT` or `WHILE` statement is non-boolean. Try formally declaring some of the facts.

Session Errors

Session errors are produced in response to user interaction with the cyscript interpreter. Their meaning is intended to be straight-forward, but for the sake of completeness are listed and briefly explained here.

Bad parameter for forget.

You have tried to forget a fact that is not yet known. Use the - `what` command to see the valid fact numbers.

Bad parameter for how.

You have tried to show how a conclusion has been concluded before it has been concluded. Try using `so` to see how many conclusions have been reached.

`Load failed.`

You tried to load a set of facts from a file that either does not exist, is unreadable or is corrupt.

`No conclusions have been reached.`

You have issued a `so` command and no conclusions have yet been reached.

`No facts have been supplied.`

You have issued a `what` command and no facts have yet been supplied.

`Rule not found.`

The full or partial name of the rule that you have specified as an argument to the `rule` command could not be found.

`Save failed.`

You have tried to save the current facts to a file and the write failed.

`The value is out of range.`

You have entered a value that is out of range. Normally you will get a more helpful message that tells you what the valid range is!

`There are no other facts under consideration.`

You have issued a `why` command or, more probably a series of `why` commands and there is no further explanation available to explain why a question is being asked[105].

Verbose Messages

Verbose messages can be either turned on at invocation with the `-d` flag or toggled on and off with the `verbose` command during a user session. They are intended to let you know what the system is really doing with your rules.

`BACKWARD-CHAINING FOR`

The system is backward chaining to find a value for a new goal fact. It will invoke each rule that potentially can conclude a value for the fact until these are exhausted, or a value is found. If along the way, it needs the values of other facts these will recursively become goal facts.

---

[105] For many years this message was "Planet Earth is blue and there is nothing I can do" which was meant to express the same level of exasperation that parents often feel when a young child just keeps asking "why?".

BLOCKED

The system has blocked a rule from firing because the rule is already being executed as part of the current set of rules being executed. If rules did not block, then an infinite loop would result.

CALLING

A rule has been explicitly called with a `CALL` statement. It is fired regardless of whether it is needed to derive a value for any of the goal facts.

CHANGED VALUE FOR

The value of a fact has been changed. This is unusual and reflects some procedural or imperative element in the code.

DETERMINED VALUE FOR

The value of a fact has been determined. If it was a goal fact, it will be popped off the goal stack and its predecessor will become the current goal.

DETERMINING

A value for a fact is being determined. This is a result of it being a goal fact.

EFACT RETURNED VARIANCE OF

This indicates the overall variance of a fact that is involved as an attribute of an example set.

EXAMPLE RETURNED SCORE OF

This number reflects the overall score of a particular example given the current set of facts / attributes.

FIRING

A rule is being fired. This can be the result of backward or forward chaining or can reflect that a rule has been specifically called.

FORGOT

The value of a fact has been forgotten. This will usually result in an attempt by the system to immediately recalculate.

FORWARD-CHAINING FOR

The system is firing a rule because one of its premises has become known. This will normally only happen if a rule is specifically tagged as being `FORWARD`.

Fatal Errors

Fatal errors can occur on invocation, during parsing / code-generation and at run-time. They indicate that something is fundamentally wrong and cause the interpreter to exit.

`filename: can't open`

You have passed the interpreter the name of a code file (both with and without the `.ys` extension) that does not exist or that cannot be opened.

`popbuft: out of stack space`
`out of memory`

Unless you are trying to run an enormous piece of code, these errors probably indicate an internal error.

`maximum recursion level reached`

This error usually indicates some form of infinite loop such as a rule calling itself unconditionally or a set of rules that operate together to produce a loop.

`Usage: ys [-cdjnpstvxy] [--facts=facts] [--goal=rule-name] [--templates=dir]  [file[.ys] ...]`

You have possibly got one of the flags wrong. Try `ys --help` for current usage.

Appendix 3: yscript 2.x Language Changes

This appendix sets out the changes to the yscript language between versions 1.x and 2.x. Changes are backwardly compatible and yscript code written in previous versions revisions of the language should still work without needing to be modified.

Context [4.18]

The most fundamental change is the introduction of *contexts*. The aim of contexts is to support large applications which need to be split into smaller self-contained and largely independent pieces. yscript code can also now be contained in more than one file. Each file may contain one or more contexts and conversely a context can be split across files.

A context provides a separate namespace for rules and facts. Each context may be, for example, an individual Act.

At the start of each new file, rules and facts are in the *shared context* and are available in all other contexts in all files containing code. If you want to start a new context (or add to one from another file), you include the keyword `CONTEXT` and the name of the context, for example:

```
CONTEXT Electoral Act 1918 (Cth)
```

From this point, all new rule and fact names have their own namespace. If a fact exists in the shared context, then that fact will be used unless it is explicitly declared in the new context. Otherwise, you can have facts in two or more different contexts with the same name but referring to different values.

If you want to refer to a fact from a different context, you need to associate the fact with one in that context by declaring it as in:

```
CONTEXT Commonwealth Constitution
FACT the definition of "adult" under Schedule 1 FROM Acts Interpretation
Act 1901
```

The effect of this is that all references to the fact 'the definition of "adult" under Schedule 1' will be to the version of this fact in the Acts Interpretation Act 1901 context. References in both the `Commonwealth Constitution` context and the `Acts Interpretation Act 1901` context will refer to the same fact.

Embedded Facts [4.13]

Curly brackets are now used instead of angle brackets for embedded facts. This applies to fact names, constants, and text. The reason for the change is to better support document assembly and interfaces involving HTML or XML. The use of angle brackets is deprecated but will still work to specify the default fact (i.e. {}, previously <>), in templated rules and otherwise where the contents of the angle brackets refer to a fact

that is either explicitly or implicitly defined elsewhere. Otherwise, angle bracket characters will be output as literals.

Markdown [8.2]

Text may now be formatted in *Markdown*. Markdown is a lightweight text formatting language that allows for simple text markup to include things like ****bold****, **italics**, headings and lists. Output from the document assembly statements `PARAGRAPH`, `LINE` and `TEXT` will now generate output in Markdown compatible format. For example:

```
DOCUMENT List Example PROVIDES
NUMBERED LEVEL 1 PARAGRAPH This section applies only if:
NUMBERED LEVEL 2 PARAGRAPH some sub-condition applies; and
NUMBERED LEVEL 3 PARAGRAPH some sub-sub-condition applies; or
NUMBERED LEVEL 3 PARAGRAPH some other sub-sub-condition applies; and
NUMBERED LEVEL 2 PARAGRAPH some other sub-condition applies.
NUMBERED LEVEL 1 PARAGRAPH The effect of not complying with 1 is whatever.
```

will produce:

```
1. This section applies only if:
> (1) some sub-condition applies; and
>> (a) some sub-sub-condition applies; or
>> (b) some other sub-sub-condition applies; and
>
> (2) some other sub-condition applies.

2. The effect of not complying with 1 is whatever.
```

Attachments [8.6]

Document assembly has been extended by *templates* and *attachments*. A template is a file that has been marked up in the new *DataLex* format or with *Jinja2* and which uses facts from yscript to insert names, dates and places into documents. Templates, reports and documents can be attached to a fact and will be displayed when a value for the fact is determined. For example:

```
FACT the client has a cause of action
ATTACH REPORT AS "Legal Advice"
ATTACH DOCUMENT AS "Letter to Client"
ATTACH TEMPLATE form.docx AS "Statement of Claim"
```

Named Subjects [4.15]

The related facts for named subjects have changed. `The named subject is a natural person` related fact becomes just: `the named subject is a person`. The type and usage of the gender fact changes from `SEX` to `GENDER` and `the sex of the named subject` is now `the gender of the named subject`. Finally, for named subjects of type `PERSON` and `PERSON-`

THING[106], there is an additional related fact: `the form of address for the named subject`. The following table compares the old related facts to the new ones:

Old Related Fact	**New Related Fact**
`STRING the name of the named subject`	`STRING the name of the named subject`
`BOOLEAN the named subject is a natural person`	`BOOLEAN the named subject is a person`
`SEX the sex of the named subject`	`GENDER the gender of the named subject`
	`STRING the form of address for the named subject`

The *gender* and *form of address* related facts also have default prompts and translations as follows:

```
GENDER the gender of the named subject
PROMPT what is the named subject's preferred gender
TRANSLATE unspecified AS the named subject's preferred gender is <>

STRING the preferred form of address for the named subject
 PROMPT what is the named subject's preferred form of address
TRANSLATE AS the named subject's preferred form of address is {}
```

There is legacy support translate the old type SEX to be replaced by GENDER, and to map the old fact names to the new ones.

Three qualifiers have been added for named subjects: GENDER-NEUTRAL, INFORMAL and UNNAMED. GENDER-NEUTRAL named subjects don't ask for a preferred gender and do not use pronouns. INFORMAL named subjects don't ask for a preferred form of address. UNNAMED uses a name only once, and then reverts to using only the name of the named section itself or pronouns.

New Type Names [3.5]

The REAL type has been renamed NUMBER. DOLLAR has become MONEY. By default, a MONEY type behaves like the old DOLLAR type and presents amounts as dollars. This can be changed however by specifying a UNIT or by setting the default MONEY unit with the DEFAULT MONEY UNIT declaration. The GENDER type can hold any string value.

Currencies and Units of Measurement [3.7]

All numeric values (NUMBERS, INTEGERS and MONEY) can be associated with a unit of measurement or a currency. They can also have a numbering style. Dates and times may also have styles.

---

[106] PERSONTHING may now also be written as PERSON-THING (preferred).

New Fact Qualifiers [0]

Two qualifiers have been introduced to fact declarations: `UNREPORTED` and `SYSTEM`. An `UNREPORTED` fact will be excluded from the final report. A `SYSTEM` fact will not be included in the explanation of any conclusion and will not be returned on a list of premises or conclusions.

Explanations [4.14]

Facts may now have optional "explanations". An explanation is a piece of text that is associated with the fact generally (the "default explanation") or with particular values for the fact. Explanations are attached to facts in a similar fashion to translations. They should follow a fact declaration and have the form:

```
STRING the type of gaming activity that you are proposing
EXPLAIN AS The Community Gaming Regulations 2020 regulate the
Conduct of gambling for social, charitable, and non-profit purposes
in NSW. The Regulations provide for 11 types of permitted gaming activities.
EXPLAIN Art union AS You will now be asked a series of questions to see
see whether or not your proposed activity is covered by the "Art
Union gaming activity" provisions which are contained in regulation 4.
```

The default explanation is displayed when the system needs to prompt the user for a fact value or when a value for the fact is first determined. It is intended to add additional explanatory material to help the user answer the question. The value-based explanations are intended to be display either after a value has been provided by the user or (depending on the interface) when the user is considering answering a question in a particular way (such as by *hovering* over an option in a graphical presentation).

CASE-WHEN-THEN Statement [5.3]

A new CASE-WHEN-THEN statement has been added. The syntax is:

```
CASE fact-name { WHEN constant-value | DEFAULT [ THEN ] statement }
```

This is useful when dealing with multi-value user input. Where the *constant-values* are all strings, then *fact-name* will automatically be `RANGE` restricted to these values (which will also result in a multiple-choice type question).

Objects [6.8]

You can create a procedure with the same name as a fact by following a fact declaration with one or more statements. If this statement is a `CASE` statement, it will assume the name of the fact and the first part of the statement can be omitted. This is helpful when implementing decision-trees and chat-bots.

For example:

```
FACT you wish to get divorced
WHEN true NEXT you are married
WHEN false THEN you may get divorced at any time following 2 years
separation

FACT you are married
WHEN true you have been separated from your partner for 2 years
WHEN false THEN you cannot be divorced

FACT you have been separated from your partner for 2 years
WHEN true THEN you can apply for a divorce
WHEN false THEN you must be separated for 2 years to get a divorce
```

New Assignment Syntax [5.1]

The syntax for assignments has changed to allow for `AND` separated multiple assignments that will be treated as a single statement. In previous versions, the assignment statement syntax was:

```
[ ASSERT | AND ] fact-descriptor [ IS expression ]
```

The new assignment statement syntax is:

```
[ ASSERT ] fact-descriptor { AND fact-descriptor } |
[ ASSERT ] fact-descriptor IS expression
```

The reason why the keyword `AND` was previously allowed as a synonym for `ASSERT` was to permit usage such as:

```
you should take an umbrella AND
you should take your gumboots
```

This was confusing when used in conditionals that allowed control of a single statement (for example, `IF-THEN-ELSE` and `WHILE`) because the second assertion was the next statement after the conditional one.

For example:

```
IF it is raining THEN
   you should take an umbrella AND
   you should take your gumboots
```

was previously (and non-obviously) interpreted as:

```
IF it is raining THEN
     you should take an umbrella
ASSERT you should take your gumboots
```

The statement is now equivalent to:

```
IF it is raining THEN BEGIN
    you should take an umbrella AND
    you should take your gumboots
END
```

FORGET Statement [5.9]

The FORGET command can be used to forget the value of facts. The syntax is:

```
FORGET ALL | fact-name
```

When the `ALL` keyword is specified, the value of all facts will be forgotten, and the user consultation will restart. Where a *fact-name* is given, the value of that fact and all subsequently determined facts will be forgotten. Care needs to be taken to only apply this to facts that have already been determined.

EXIT Statement [5.11]

The `EXIT` statement causes the system to exit. The syntax is:

```
EXIT [ SESSION ]
```

When followed by the keyword `SESSION`, the current session is terminated, and control is returned to the calling environment. For cyscript, this will result in the `"Another problem?"` prompt. Without the `SESSION` keyword, the statement causes an immediate exit. All outstanding messages are ignored, and the calling environment is terminated.

SAY Statement [5.10]

The `SAY` statement causes an associated text message to be displayed before the next prompt. This is only useful in procedural code or for debugging. Otherwise, the command will be executed each time a `RULE` is fired.

INCLUDE Directive [5.12]

The new `INCLUDE` directive allows the contents of a file to be included at any point in the code. In the DataLex environment, this directive is pre-processed to extend what can be included to URIs.

Comments [3.3]

In addition to existing C-style multi-line comments, single line comments may now also be used. Anything after `//` on a line will be ignored.

Ranges [4.10]

cyscript (the name of the default command line interpreter) now displays a multiple-choice list of options when a fact has a range of a set of discrete values. The DataLex interface also does this. See `gaming.ys` for an example[107]. The range error messages have

[107] See page 187.

been improved to describe the range automatically, and a new facility to allow translation of the required range error message has been added via an extension to the `TRANSLATE` declaration.

For example:

```
TRANSLATE RANGE AS please enter a number from 1 to 100
```

Facts [4.3]

Facts may now have a tilde in them to specify where the verb is / how to split the proposition into a subject and a predicate. For example:

```
profits~exceed expectations
the government~reports a likely deficit
```

The tilde is part of the name of the fact but is not displayed (except in the code itself of course). Use of this approach should be used sparingly.

SUBRULE Alternative to CALL [5.6]

A new keyword – SUBRULE has been added as an alternative for CALL. This allows for more natural syntax when working with legislation.

VERBS Generally Unnecessary [4.3]

A much larger verbs dictionary (6,000 words in four forms)[108] has been added. The `VERBS` statement is now generally unnecessary. The old tilde (~) separated format is deprecated (but still available) and replaced with a more straightforward piped (|) format.

For example:

```
VERBS foresee|foresees|foresaw|foreseen
```

LISTED and UNLISTED Deprecated [6.3]

The `LISTED` and `UNLISTED` qualifiers to a `RULE` declarations have been deprecated. For backward compatibility, `LISTED` without being followed by `GOAL` is now interpreted as just `GOAL`. `UNLISTED` is totally ignored.

[108] The verbs list is contained in the source file "verbs.h".

Lexical Changes

Strings can now use single quotes as well as double quotes. Values and text for `TRANSLATIONs` may now be quoted or unquoted. Descriptors including fact names and constants may contain any Unicode characters in UTF-8 format.

Whilst not encouraged, the lexical analyser will translate UTF-8 double and single quotes, em and en dashes and non-breaking spaces to their ASCII equivalents where these symbols are lexically significant (for example, to designate a string).

DataLex Declarations

DataLex `LINK` and `DEFAULT` statements are parsed and ignored. It is a good idea to continue excluding non-yscript extensions with comment tags, for example:

```
/*
DEFAULT ACT Modern Slavery Act 2018
*/
```

Interactive Commands

The `goal` and `rule` commands in cyscript have been extended:

`goals`	show all defined goals or all rules if no goals defined
`goals all`	show all rules and allow session for any of these
`goal` *`pattern`*	change current goal to first goal with name including pattern
`rule` *`pattern`*	change current goal to first rule with name including pattern
`rule`	show the source of the rule under consideration
`rules`	show the hierarchy of all rules under consideration
`rules all`	same as "`goals all`"
`show fact`	show the name of the current fact
`show rule` *`pattern`*	show a specified rule
`why this`	display the first why explanation (only)

Appendix 4: Formal Grammar

```
code = block { block }

block = context | defaults | example | fact-declaration | include | order | rule |
verbs

context = CONTEXT context

defaults = DEFAULT generic-type STYLE text

fact-declaration = [ GOAL ] fact-type descriptor [ FROM context ] [ PROVIDES ]
             { attachment | explanation | info | prompt | range | translation }
                    [ statements ]

fact-type = [qualifier] [type FACT | FACT]

qualifier = SYSTEM | UNREPORTED | UNNAMED | INFORMAL | GENDER-NEUTRAL

attachment = ATTACH [DISPLAYED] [qualifier] REPORT|DOCUMENT|TEMPLATE name [AS
description]

explanation = EXPLAIN [ UNKNOWN | value ] AS text

info = INFO text

prompt = PROMPT text

range = RANGE arith-term [ TO arith-term ]

translation = TRANSLATE [ RANGE | UNKNOWN | value ] AS text

fact-type = generic-type | named-subject

generic-type = BOOLEAN|DATE|GENDER|INTEGER|MONEY|NUMBER|STRING

named-subject = PERSON | THING | PERSON-THING

verbs = VERB | VERBS descriptor

include = INCLUDE file-name

example = example-header example-body

example-header = [GOAL] EXAMPLE [RULE] [descriptor] PROVIDES

example-body = IF bool-expr THEN assignment

order = ORDER descriptor { THEN descriptor }

rule = rule-header statements

rule-header = [GOAL] rule-type [RULE] [descriptor] PROVIDES

rule-type = BACKWARD|DAEMON|DOCUMENT|FORWARD|PROCEDURE|RULE

statements = statement { statement }
```

```
statement =
assignment|call|case|determine|exit|forget|if|include|repeat|say|while|write|
             BEGIN statements END

assignment = [ ASSERT ] descriptor { AND descriptor } |

             descriptor IS|ONLY IF expression

call = CALL | SUBRULE | NEXT [GOAL] descriptor [ FROM descriptor ]

case = CASE descriptor { WHEN descriptor [ THEN ] statement }

determine = DETERMINE descriptor

exit = EXIT [SESSION]

forget = FORGET ALL | descriptor

if = IF expression THEN statement [ ELSE statement ]

say = SAY text

repeat = REPEAT statements UNTIL expression

write = [NUMBERED] [LEVEL number] PARAGRAPH|LINE|TEXT text

while = WHILE expression DO statement

expression = bool-expr { OR|OR/WITH bool-expr }

bool-expr = and-or-expr { AND|AND/WITH and-or-expr }

and-or-expr = rel-expr { AND/OR|AND/OR/WITH rel-expr }

rel-expr = arith-expr { [IS] rel-op [THAN|TO] arith-expr }

arith-expr = term { PLUS|MINUS } term

term = factor { TIMES|DIVIDED [BY] } factor }

factor = {pre-unary-op} descriptor [post-unary-op]

rel-op = LESS|GREATER|LESSEQUAL|GREATEREQUAL|EQUAL|EQUALS|NOT EQUAL|NOT EQUALS

pre-unary-op = NOT|MINUS|PLUS|DAY|MONTH|YEAR|HOUR|MINUTE|SECOND|UNKNOWN

post-unary-op = DAY|WEEK|MONTH|YEAR|DAYS|WEEKS|MONTHS|YEARS|HOURS|MINUTES|SECONDS
```

Appendix 5: The yscript API

yscript includes an API to a C language-based library. The standard distribution includes a static and shared object version of the library as well as a Perl module and separate library. The library/API should be usable from most other languages. Examples are provided in this appendix for Python, Ruby, Perl, C++ and C. More complete examples are provided for the `cyscript` code (contained in `src/ys.c`) and the Perl implementation of `cyscript – API/ys.pl`.

Fundamental Routines

To create a yscript-based application, you need to call at least three basic routines:

- `ys_load()` or `ys_load_files()` to load one or more yscript code files
- a loop to call `ys_evaluate()` to start execution and to give the next fact prompt
- `ys_get_report()` to get a copy of the final *report*

`ys_load()` loads a single file containing yscript code, whereas `ys_load_files()` takes a number and string vector containing the names of one or more files.

`ys_evaluate()` optionally specifies a starting goal, gives the value and known status of the last current fact, and passes back a prompt and a type for the next current fact.

`ys_report()` returns a pointer to the final report.

Applications should include the header file `ysapi.h`.

The core of an application using the API (in C) looks like:

```
ys_load(filename);
while (ys_evaluate(0, reply, YS_KNOWN_VALUE, prompt, &type) != 1) {
     fputs(prompt, stdout);
     fgets(reply, YS_MAXLINE, stdin);
}
ys_get_report(0);
```

Fuller examples of simple applications in C, C++, Perl, Python and Ruby are available as `ys-simple.c`, `ys-simple.cpp`, `ys-simple.pl`, `ys-simple.py` and `ys-simple.rb` respectively.

A complete alphabetical list of all API calls follows:

Alphabetical List of API Calls

*int ys_check_range(char * reply)*

Check that a user reply is in range for the current fact. Returns `0` if valid or `-1` if illegal.

*int ys_check_type(char * reply, int type)*

Check the *type* of a user *reply*. Returns `0` if valid or `-1` if illegal.

int ys_clear_messages()

Clear all messages on the message queue. This will delete any pending explanations, messages, and verbose messages.

int ys_conclusions_count()

Return the number of conclusions reached.

*int ys_copy_facts(FILE * stream, char ** mem, int flags)*

Copy facts to a file or memory. Flags are:

```
YS_FACTS_ALL           include conclusions
YS_FACTS_ALIASES       use alias if available
YS_FACTS_ALIASES_ONLY  only save facts with aliases
YS_FACTS_INCL_GOAL     include the current goal
YS_FACTS_EXCL_CONTEXT  don't include context info
YS_FACTS_NON_BOOLEAN   include only non-boolean and aliased facts
YS_FACTS_IDENTIFY      identify GOAL/FACT/VALUE lines
YS_FACTS_UNDERSCORE    replace spaces in fact names with underscores
YS_FACTS_JSON          output in JSON format
```

*int ys_evaluate(char *goal, char *val, int known, char *prompt,int * type)*

Start or continue a yscript session using "*goal*" as the goal rule (if not `NULL`). Returns `1` if the session is continuing and sets "*prompt*" to a string to ask the user for information of type "*type*". The next time, `ys_evaluate` is called you should pass it the user response as a string in "*val*" and its known status in "*known*". See below for value types and known constants. If `ys_set_askquestion()` has been used to set a call-back to get user answers, you should only have to call ys_evaluate once per session.

*char * ys_explain(char * value)*

Return an explanation associated with a fact for a "*value*". If "*value*" is `NULL`, return the default explanation for the fact. If there is no associated explanation, return `NULL`.

int ys_forget(int number)

Forget a fact (that is, make a fact uncertain). When called with `-1`, forget all facts. After a fact value is changed, it is important that the next call have a known value of `YS_KNOWN_UNKNOWN` to cause the system status to be recalculated.

*char * ys_get_document(char * rule-name)*

Return the document as a string (with newlines at paragraph ends). If rule-name is NULL or empty, then use the current goal or the first rule if none.

*char * ys_get_fact(int number, int premises)*

Get a descriptor for fact "number" or the current fact if number is 0. Use *premises*|YS_CONTEXT to return context information (i.e. *context*::*fact-description*).

*char * ys_get_goal()*

Return the name of the current goal rule or NULL if not set.

int ys_goal_count(int all)

Return the number of defined goal rules.

*char * ys_get_how(int which)*

Get an explanation as to how a conclusion was made.

*char * ys_get_info()*

Return hint/info for the current fact. Return NULL if not set.

*int ys_get_next_message(char **message, char **desc, char ***info, int *flags)*

Get the next "message" described as "desc" with optional "info" null terminated set of strings and flags set as "flags" from the message stream and return the message type. Returns -1 at end of list.

Valid flags are:

```
YS_ATTACH_DISPLAY       attachment should be displayed
```

*char * ys_get_prompt(int reset_pronouns)*

Return a prompt for the current fact. If "*reset_pronouns*" is non-zero, use name (not pronoun) for the first named subject in the fact.

*char * ys_get_range_trans()*

Get a translation for expected range. Returns NULL if there is none.

*char ** ys_get_range_options()*

Get a vector of individual valid range values (if in use else NULL)

*char * ys_get_report(char * rule-name)*

Return the final report as a string (with newlines at paragraph ends). If *rule-name* is NULL or empty, then use the current goal or the first rule if none.

*char * ys_get_rule(int level, int title_only, int indent)*

Return a preformatted copy of the source for a rule at "*level*" on the rule stack (use zero for the current rule). Use *title_only*|YS_CONTEXT to return information about context i.e. *context*::*rule-name*

*char * ys_get_rule_by_name(char * rule-name, int indent)*

Return the text of a rule matching rule-name or pattern.

*char * ys_get_why(int level)*

Get a string explanation as to why a fact value needs to be determined.

*int ys_load(char * filename)*

Load a single yscript code file. Note use ys_load_files() for more than one file. Return -1 on failure.

*int ys_load_files(int filec, char ** filev)*

Load "*filec*" yscript code files with names list in string vector "*filev*". Returns -1 on failure.

*int ys_load_facts(char * filename)*

Load the current user facts/state from a file

*char * ys_next_goal(int first, int all)*

Get the next rule name or goal rule name, depending on "*all*". If "first" is set, then start (again) at the first rule. Return NULL when at last rule or goal. "*all*" is 0 for the next GOAL rule or 1 for the next RULE regardless of whether it is a GOAL rule. Use *all*|YS_CONTEXT to return rule-name with context included (i.e. *context*::*rule-name*).

int ys_premises_count()

Return the number of premises that have been supplied.

*int ys_rule_exists(char * pattern)*

Check is a rule exists with a name containing "*pattern*". Returns non-zero on success.

int ys_rules_count()

Return number of rules under consideration (i.e., on rules stack).

*int ys_save_facts(char * filename)*

Save the current user facts/state to a file.

*void ys_set_askquestion(int(*p)(char* prompt,char* reply,int type, int x))*

Set call-back to get an answer in "*reply*" to a question with "*prompt*" of "*type*". "*x*" is set to non-zero for questions generated by yscript in case you want to use the routine for your own questions.

*int ys_set_goal(char * goal)*

Set the goal rule. Return –1 if the rule does not exist or 0 otherwise.

*char * ys_showall(FILE * stream, char ** string)*

Write a complete formatted (pretty-printed) version of the code to a stream or a dynamically allocated string.

*char * ys_showtrans(FILE * stream, char ** string)*

Write a list of all fact translations and prompts to a stream or string.

*char * ys_showstats(FILE * stream, char ** string)*

Write system statistics to a stream or a string.

*char * ys_xref(FILE * stream, char ** string)*

Write the fact cross-index to a stream or a string.

int ys_fpretty_print(FILE stream,char** str,char* s, int left, int right)*

Write a string "*s*" to a "*stream*" or a string ("*str*") formatted to have a left margin of "*left*" spaces and a right line length of "right" characters. If "stream" is NULL then write to "str" which should be a pointer to a NULL initialised character pointer. If "left" is less than zero then don't format (just do screen pagination). If "right" is less than "left" use screen width (80 characters).

*int ys_pretty_print(char * s, int left)*

Equivalent to `ys_fpretty_print(stdout, NULL, s, left, –1).`

int ys_pretty_no_pause(int on)

Turn pretty-print screen pagination on or off

API Constants

The constant YS_MAXTERM is defined in "ysapi.h" and is used for all internal string buffers.

Valid fact types are:

```
YS_TYPE_BOOLEAN        "yes" or "no"
YS_TYPE_DATE           date (in any format)
YS_TYPE_GENDER         a string
YS_TYPE_INTEGER        positive or negative integer
YS_TYPE_MONEY          monetary amount
YS_TYPE_NUMBER         positive negative real
YS_TYPE_STRING         any string
YS_TYPE_TIME           time (in any format)
```

Valid "known" values are:

```
YS_KNOWN_UNSPECIFIED   user "uncertain" response
YS_KNOWN_UNKNOWN       known to be unknown
YS_KNOWN_FALSE         known to be false (booleans only)
YS_KNOWN_TRUE          known to be true (booleans only)
YS_KNOWN_MALE          known to be male (gender only)
YS_KNOWN_FEMALE        known to be female (gender only)
YS_KNOWN_VALUE         value is known
```

The following sections include simple examples of using the API from various languages:

C / C++ API Example

```
/*
 * ys-simple.cpp / ys-simple.c -- simple C/C++ example using yscript API
 */

#include <stdio.h>
#include <string.h>
#include "ysapi.h"

int main(int argc, char* argv[])
{
   int   type;
   char  reply[YS_MAXLINE],
         prompt[YS_MAXLINE],
         full_prompt[YS_MAXLINE];

   ys_load_files(argc-1, &argv[1]);
   while (ys_evaluate(0, reply, YS_KNOWN_VALUE, prompt, &type) != 1) {
         snprintf(full_prompt, YS_MAXLINE, "\n%s\n** ", prompt);
         ys_pretty_print (full_prompt, 6);
         fgets(reply, YS_MAXLINE, stdin);
   }
   ys_get_report(0);
   return 0;
}
```

Perl API Example

```
#
#   ys-simple.pl -- simple perl example using the yscript API
#

use ys;                                         # API api module

ys::ys_load_files($#ARGV+1, \@ARGV);          # load source file(s)

do {
    ($retval, $prompt, $type) =
        ys::ys_evaluate("", $reply,           # evaluate / get next question
            $ys::YS_KNOWN_VALUE);             # until done
    if ($retval != 1) {
        ys::ys_pretty_print("\n$prompt\n** ", 6);
        $reply = <STDIN>;
    }
} while ($retval != 1);

$report = ys::ys_get_report("");              # show the report
ys::ys_pretty_print("$report\n", 6);
```

Python API Example

```
#
#    ys-simple.py -- simple yscript api example for python 2.x/3.x using ctypes
#

from sys import *
from ctypes import *                          # use ctypes

ys = CDLL("../libys.so")                             # load shared object library
and
                                              # set up api argtypes and restypes
 ys.ys_load.argtypes=[c_char_p]
ys.ys_evaluate.argtypes=[c_char_p, c_char_p, c_int, c_char_p, POINTER(c_int)]
ys.ys_get_report.argtypes=[c_char_p]
ys.ys_get_report.restype=c_char_p
ys.ys_pretty_print.argtypes=[c_char_p, c_int]

if len(argv) != 2:
    print("Usage: " + argv[0] + " filename[.ys]")
    exit(1)

ys.ys_load(argv[1].encode("utf-8"))         # load yscript code file

prompt = create_string_buffer(4096)
type = c_int(0)
reply = b""
retval = 0
known = 0

while retval != 1:                                  # evaluate and get user
answers
    retval = ys.ys_evaluate(b"", reply, known, prompt, byref(type));
    if retval != 1:
        myprompt = prompt.value.decode("utf-8")
        ys.ys_pretty_print(b"\n"+myprompt.encode("utf-8")+b"\n** ",6)
        reply = stdin.readline().encode("utf-8")

report = ys.ys_get_report(b"")              # show report
ys.ys_pretty_print(b"\nREPORT\n\n" + report + b"\n\n", 6)
```

Ruby API Example

```
#
#    ys-simple.rb -- simple yscript api example for ruby using ffi
#

require 'ffi'

module YS
    extend FFI::Library
    ffi_lib 'c'
    ffi_lib '../libys.so'
    attach_function :ys_load, [ :string ], :int
    attach_function :ys_evaluate,[:string,:string,:int,:pointer,:pointer],:int
    attach_function :ys_get_report, [ :string ], :pointer
    attach_function :ys_pretty_print, [ :pointer, :int ], :int
end

reply = ""
prompt = FFI::MemoryPointer.new(:char, 4096)
type = FFI::MemoryPointer.new(:long)

YS.ys_load(ARGV[0])
while YS.ys_evaluate("", reply, 0, prompt, type) != 1
    YS.ys_pretty_print("\n", 6)
    YS.ys_pretty_print(prompt, 6)
    YS.ys_pretty_print("\n** ", 6)
    reply = $stdin.readline
end
YS.ys_pretty_print(YS.ys_get_report(""), 6)
YS.ys_pretty_print("\n", 6)
```

Appendix 6: Using cyscript as a Server

The main purpose of cyscript is to provide an interactive command-line based user environment for running applications using the yscript interpreter. You can also use cyscript as a server by placing it on a socket. In this mode, cyscript reads commands on standard input and sends the results to a machine interface. To make things easier, cyscript can generate output in the data serialization formats - *JSON*[109] and *YAML*.[110]

The JSON and YAML interfaces have identical. Most computing languages have libraries that support both formats. YAML is the more modern of the two standards and is easier to read.

1. Invocation

To start cyscript as a server generating JSON, call it with the `-j` flag and to have it generate YAML, call it with the `-y` flag. Apart from changing the type of output that cyscript produces, either flag will also cause screen pagination to be supressed and for more regular flushing of output streams. If you are going to be using external document template files, you will probably also want to set the template directory with the `-T` or `--templates` flag. For example, to invoke cyscript as a server using YAML and setting the template directory:

```
ys -y --templates=/usr/local/ys/templates application.ys
```

2. Commands

Input is identical in server and interactive cyscript modes. Each line of input is in response to a question that is asked by the server. The same commands (such as `why`, `what`, and `so`) can be issued in response to any question.

3. Output

Output will either be in JSON or YAML format depending upon how cyscript is invoked. In both cases, the end of output is indicated by three dots ("`...`") on a line by

---

[109] JSON is an acronym for *JavaScript Object Notation* and is an open standard file and data-serialization and interchange format. JSON is a language-independent data format, but its syntax is derived from JavaScript object notation. See <www.json.org> for further information.

[110] YAML is a recursive acronym for *YAML Ain't Markup Language* and is also a data-serialization language that is commonly used for configuration files. It is a strict superset of JSON and includes syntax elements from Python, Perl and C. See <http://yaml.org> for information.

themselves. In a sense, this replaces the prompt in interactive mode to say that the server is waiting for input.[111]

JSON data consists of one or more objects which are enclosed by curly brackets. Each object consists of one or more name/value pairs which are separated by commas. Each name/value pair consists of a name in double quotes (sometimes called a *key*) followed by a colon (':') and then a value. A *value* can be a *null*, a *string*, a *number*, a *boolean*, an *array* or an *object*. *Strings* are in double quotes, *null* is just written as null, *numbers* are numbers and *booleans* are true or false. *Arrays* are in square brackets.

YAML is a more complicated format than JSON. YAML data is inherently line based and depends upon space-based indenting of text to indicate data structure. Fundamentally it is similar to JSON in that the basic units of representation are key/value pairs. Lines and indenting replace the need for curly and square brackets and obviate the requirement for comma separators. Key/Value pairs are separated by colons. YAML output is divided into *documents* which are indicated by a line containing three dashes ('---').

The following example, shows the same yscript output in both JSON and YAML format:

YAML

```
---
question:
  number: 3
  type: boolean
  text: Did Telstra have a consolidated
    revenue of at least $100 million
    for the reporting period?


...
yes
---
question:
  number: 4
  type: boolean
  text: Was Telstra a company during the
    reporting period?


...
why
---
why:
  text: This will help determine whether
    or not Telstra was a company which
    was a resident within the meaning
    of subsection 6(1) of the Income
    Tax Assessment Act 1936 during the
    reporting period.

---
question:
  number: 4
  type: boolean
  text: Was Telstra a company during the
```

JSON

```
{
  "question" : {
    "number" : 3,
    "type" : "boolean",
    "text" : "Did Telstra have a consolidated
revenue of at least $100 million for the
reporting period?"
  }
}
...
yes
{
  "question" : {
    "number" : 4,
    "type" : "boolean",
    "text" : "Was Telstra a company during the
reporting period?"
  }
}
...
why
{
  "why" : {
    "text" : "This will help determine whether
or not Telstra was a company which was a
resident within the meaning of subsection 6(1)
of the Income Tax Assessment Act 1936 during
the reporting period."
  }
}
{
  "question" : {
    "number" : 4,
    "type" : "boolean",
```

[111] Three dots formally indicate the end of a document in the YAML standard. Many parsers do not support this. It is not part of the JSON standard. It is best separately parsed and regarded as a prompt.

```
    reporting period?
...
```

```
   "text" : "Was Telstra a company during the
reporting period?"
  }
}
...
```

4. Objects

All output consists of one of twenty different types of objects. As can be seen in the above example, the most used object is "question" which indicates that the user is being asked for the value of the *numbered* premise of a particular *type* with the prompt indicated by *text*. The other object in the above example is "why" which contains a response to a "why" command indicating the reason for asking the current question as *text*.

The complete list of object values and their attributes is as follows:

```
attachments:
  text: intro-text
  list:
    - number: attach-number
      text: attach-desc

conclusions:
  list:
    - number: fact-number
      text: fact-description

document:
  text: |
    document-text

error:
  text: error-message

explanation:
  text: |
    explanation-text

fact:
  text: fact-name

file:
  name: file-name
  text: description

goals:
  text: text
  list:
    - number: goal-number
      text: rule-name
```

```
help:
  text: help-text

how:
  text: how-text

multi:
  text: prompt
  list:
    -  option: letter
       text: text

premises:
  list:
    number: fact-number
    text: fact-description

question:
  number: fact-number
  type: type
  info: info-text
  text: prompt

report:
  text: |
    report-text

rule:
  text: |
    rule-text

rules:
  list:
    - level: level-number
      text: rule-name
```

```
template:
  file: file-name
  format: template-language
  facts:
    - fact-name: value

verbose:
  level: number
  text: message

whatif:
  text: text
  list:
    - conclusions

why:
  text: why-text
```

5. Detailed Object Description

This section sets out a description of each object and provides a brief example.

attachments

Syntax:
```
attachments:
  text: intro-text
  list:
    - number: attach-number
      text: attach-description
```

Example:
```
attachments:
  text: The following documents are
    available:
  list:
    - number: 1
      text: Record of Advice
    - number: 2
      text: Draft Non-Disclosure
Agreement
    - number: 3
      text: Letter to Client
```

The `attachments` object describes a list of attachment messages relating to reports, documents and template documents that have been received because a value for a fact with `ATTACH` declarations has been set. The *`text`* attribute is an introductory comment to the list of attachments. The *list* itself consists of two or more objects each with a *number* (which is just an ordinal count) and *text* which contains a brief description of the attachments.

This object is only produced when there are two or more attachments which need to be drawn to the attention of the user. The object will be followed by a `question object to deal with the numbered` user response.

conclusions

Syntax:
```
conclusions:
  list:
    - number: fact-number
      text: fact-description
```

Example:
```
conclusions:
  list:
    - number: 1
      text: Google was a body corporate
        during the reporting period.
    - number: 2
      text: Section 5(2)(b) does apply.
    - number: 3
      text: Google was not a trust.
```

The `conclusions` object is produced in response to a `so` command. It includes a `list` of facts that the system has concluded. The `numbers` can be used as arguments to the `how command.`

document

Syntax:
```
document:
  text: |
    document-text
```

Example:
```
document:
  text: |
    1st July, 2021

    Henry Ford
    Ford Motor Co
    Detroit

    Dear Henry,

    Re: **Non-Disclosure Agreement**
```

The document object contains pre-formatted text that has been generated for a document. Markdown formatting is maintained.

error

Syntax:
```
error:
  text: error-message
```

Example:
```
error:
  text: Please respond with yes or no.
```

The `error` object is produced whenever there is an error.[112]

explanation

Syntax:
```
explanation:
  text: |
    explanation-text
```

Example:
```
explanation:
  text: |
    This application will help you to
    determine if your proposed gaming
    activity fits within the legislative
    framework and whether or not you
    need formal approval.
```

The `explanation` object is generated either because a question is being asked about a fact that has an `EXPLAIN` declaration or a fact has been determined for a fact which has an explain declaration. The associated `text` may have associated Markdown elements.

fact

Syntax:
```
fact:
  text: fact-name
```

Example:
```
fact:
  text: |
    the gaming activity is a game in
    which prizes are awarded by lot
```

[112] A full list of error messages is on page 127.

The fact object results from running the `show fact` command. The `text` attribute contains the unmodified name of the current fact (that is, the one being asked about).

file

Syntax:
```
file:
  name: file-name
  text: description
```

Example:
```
file:
  name: /tmp/ys.qJO2j.docx
  text: Draft Non-Disclosure Agreement
```

The `file` object results when a user has requested to view a file. The `name` attribute contains the file-name. This will be relative to the template directory if set by the `-T` flag. The `text tag` contains a short description of the file.

goals

Syntax:
```
goals:
  text: text
  list:
    - number: goal-number
      text: rule-name
```

Example:
```
goals:
  text: The following goals are defined:
  list:
    - number: 1
      text: Competition and Consumer Act
        2010 Section 52A "covered News
        content"
    - number: 2
      text: Competition and Consumer Act
        2010 Section 52A "designated
        digital platform corporation"
```

The `goals` object can be generated at the beginning of a user session if no goal has been set and there is more than one to choose from. It can also be generated in response to the `goals` and `rules all` commands. The `text` attribute contains a short introductory message. `list` is an array or list of objects consisting of a `number` which can be used to select the goal in response to the next question and a `text` description of the goal or rule.

help

Syntax:
```
help:
  text: help-text
```

Example:
```
help:
  text: Other valid responses are:
    forget, goals, how, quit, rule,
    so, uncertain, verbose, and why.
```

The `help` object results from running the `help` command. It just lists some of the available commands.

how

Syntax:
```
how:
  text: how-text
```

Example:
```
how:
  text: Section 156(1)(b) of the
    National Law (Qld) does not apply
    because Dr Smith is not a
    registered student.
```

The `how` object results from running the `how` command. Your interface should normally just display this for the user to see it.

multi

Syntax:
```
multi:
  text: prompt
  list:
    - option: letter
      text: text
```

Example:
```
multi:
  text: What is the type of gaming
    activity that you are proposing?
  list:
    - option: a
      text: Art union
    - option: b
      text: Housie or Bingo
    - option: c
      text: Draw lottery (including
        raffles and guessing
        competitions)
```

The `multi` object results from the need to ask the user a question with a set `RANGE` of discrete values (either explicitly using `RANGE` or via a `CASE-WITH-THEN` statement). The prompt is an over question prompt. `list` contains a set of objects with each `option` indicating a different `text` response. The `multi` object will be followed by a `question` object to gather the multi-choice response from the user.

premises

Syntax:
```
premises:
  list:
    number: fact-number
    text: fact-description
```

Example:
```
premises:
  list:
    - number: 1
      text: The name of the layout is
        Apple M1.
    - number: 2
      text: Apple M1 is a plan
        comprising a two dimensional
        representation.
    - number: 3
      text: Apple M1 is stored in a form
        from which it, or a substantial
        part of Apple M1, can be
        reproduced.
```

The `premises` object results from executing the `so` command. It lists all of the known user-supplied facts by `number` and description (`text`).

question

Syntax:
```
question:
  number: fact-number
  type: type
  info: info-text
  text: prompt
```

Example:
```
question:
  number: 1
  type: string
  info: You should include the full name.
  text: What is the name of the nominee?
```

The `question` object is generated each time the user needs to provide an answer to a question about the value of a fact or where some other value is necessary such as a multi-choice or a goal. `number` is the fact number that can be used with `forget` or -1 where the question does not refer to a fact. The `type` is the type of value required. It is

one of: `boolean`, `string`, `date`, `integer`, `money`, `number` or `gender`. `info` is extra information about the question that should be displayed when the user hovers over the question or asks for more information. Your interface should display the `prompt` and return the user response.

report

Syntax:
```
report:
  text: |
    report-text
```

Example:
```
report:
  text: |
    The Act does not apply because
    UTS/Fudan Exchange is not a "non-
    Core foreign arrangement" under the
    section 4 definition and it is not a
    "core foreign arrangement" under
    section 10(2).
```

The `report` object is generated at the end of a session or as a result of an attachment. It contains pre-formatted text which could be in Markdown format. This should just be displayed.

rule

Syntax:
```
rule:
  text: |
    rule-text
```

Example:
```
rule:
  text: |
    RULE Electoral Act 1918 - Section
      163(1)(b) PROVIDES
        section 163(1)(b) of the
        Electoral Act 1918 is satisfied
        ONLY IF
          the nominee is an
          Australian citizen.
```

The `rule` object is generated by the `rule` command. The `text` is pre-formatted and care should be taken not to re-format it.

rules

Syntax:
```
rules:
  list:
    - level: level-number
      text: rule-name
```

Example:
```
rules:
  list:
    - level: 2
      text: Check overall compliance
        with Act
    - level: 1
      text: Section 4 - Definitions
        "non-core foreign arrangement"
    - level: 0
      text: Section 6 - Foreign
        arrangements
```

The `rules` object lists the rules that are under consideration by `level` and rule name (`text`).

template

Syntax:
```
template:
  file: file-name
  format: template-language
  facts:
    - fact-name: value
```

Example:
```
template:
  file: nda.docx
  format: DataLex
  facts:
    the_governing_law: United States
      South Wales
    the_name_of_the_client: Elon Musk
    the_name_of_the_disclosing_party:
      Tesla
    address_of_the_disclosing_party:
      Palo Alto
```

The `template` object is generated where an attachment has been generated and the user has selected to view or copy it. File is the name of the file `template` and the name will be relative to the template directory if `-T` is specified. The `format` is one of `DataLex` or `Jinja2`. DataLex templates only work with Word documents and can only be processed with the DataLex environment. Jinja2 documents are generated automatically and do not generate `template` objects.

verbose

Syntax:
```
verbose:
  level: number
  text: message
```

Example:
```
verbose:
  level: 1
  text: FIRING Check overall compliance
    with Act
---
verbose:
  level: 2
  text: BACKWARD-CHAINING FOR the name
    of the arrangement
```

`Verbose` objects describe system actions. They should only be displayed when the user has specifically for this.

whatif

Syntax:
```
whatif:
  text: text
  list:
    - conclusions
```

Example:
```
whatif:
  text: The following conclusions will
    be drawn:
  list:
    - AustLII Lottery is not an art
      union gaming activity.
    - AustLII Lottery is not a permitted
      "art union gaming activity" under
      regulation 4.
```

The `whatif` object results from a `whatif` command. If only a single result will happen, then no `list` is generated. `text` and the `list` of conclusions should normally be displayed.

why

Syntax:
```
why:
  text: why-text
```

Example:
```
why:
  text: This will help determine whether
    or not section 5(1)(a) applies.
```

The `why` object results from the `why` command. `text` contains an explanation as to why the current fact is being asked. This should normally just be displayed.

Appendix 7: yscript Code Examples

1. Commonwealth Constitution s44

```
//
//    elect.ys – eligibility for Federal Parliament under s44
//                of the Australian Constitution and the Australian
//                Electoral Act


  PERSON the nominee

  GOAL RULE Eligibility for Nomination to Federal Parliament PROVIDES
       the nominee is entitled to be nominated for election to federal parliament
       ONLY IF
            the nominee is entitled to be nominated under section 163
           of the Electoral Act 1918 AND
            section 164 of the Electoral Act 1918 does not apply AND
            section 44 of the Constitution does not apply

  RULE Electoral Act 1918 – Section 163 PROVIDES
       the nominee is entitled to be nominated under section 163 of the
       Electoral Act 1918 ONLY IF
            section 163(1) of the Electoral Act 1918 is satisfied AND
            section 163(2) of the Electoral Act 1918 is not satisfied

  RULE Electoral Act 1918 – Section 163(1) PROVIDES
       section 163(1) of the Electoral Act 1918 is satisfied ONLY IF
            section 163(1)(a) of the Electoral Act 1918 is satisfied AND
            section 163(1)(b) of the Electoral Act 1918 is satisfied AND
            section 163(1)(c) of the Electoral Act 1918 is satisfied

  RULE Electoral Act 1918 – Section 163(1)(a) PROVIDES
       section 163(1)(a) of the Electoral Act 1918 is satisfied ONLY IF
            the age of the nominee IS GREATEREQUAL THAN 18 OR
            UNKNOWN the age of the nominee AND
            the nominee is 18 years of age or older

  RULE Electoral Act 1918 – Section 163(1)(b) PROVIDES
       section 163(1)(b) of the Electoral Act 1918 is satisfied ONLY IF
            the nominee is an Australian citizen

  RULE Electoral Act 1918 – Section 163(1)(c) PROVIDES
       section 163(1)(c) of the Electoral Act 1918 is satisfied ONLY IF
            section 163(1)(c)(i) of the Electoral Act 1918 is satisfied OR
            section 163(1)(c)(ii) of the Electoral Act 1918 is satisfied

  RULE Electoral Act 1918 – Section 163(1)(c)(i) PROVIDES
       section 163(1)(c)(i) of the Electoral Act 1918 is satisfied ONLY IF
            the nominee is an elector entitled to vote at a House of
                  Representatives election

  RULE Electoral Act 1918 – Section 163(1)(c)(ii) PROVIDES
       section 163(1)(c)(ii) of the Electoral Act 1918 is satisfied ONLY IF
            the nominee is qualified to become an elector entitled to vote
                  at a House of Representatives election

  RULE Electoral Act 1918 – Section 163(2) PROVIDES
       section 163(2) of the Electoral Act 1918 is satisfied ONLY IF
            section 163(1) of the Electoral Act 1918 is not satisfied
```

```
RULE Electoral Act 1918 - Section 164 PROVIDES
     section 164 of the Electoral Act 1918 applies ONLY IF
          section 164(a) of the Electoral Act 1918 applies OR
          section 164(b) of the Electoral Act 1918 applies OR
          section 164(c) of the Electoral Act 1918 applies

RULE Electoral Act 1918 - Section 164(a) PROVIDES
     section 164(a) of the Electoral Act 1918 applies ONLY IF
          the nominee is a member of the Parliament of a State

RULE Electoral Act 1918 - Section 164(b) PROVIDES
     section 164(b) of the Electoral Act 1918 applies ONLY IF
          the nominee is a member of the Legislative Assembly of
                the Northern Territory of Australia

RULE Electoral Act 1918 - Section 164(c) PROVIDES
     section 164(c) of the Electoral Act 1918 applies ONLY IF
          the nominee is a member of the Legislative Assembly for
                the Australian Capital Territory

RULE Constitution - Section 44 PROVIDES
     section 44 of the Constitution applies ONLY IF
          section 44(i) of the Constitution applies OR
          section 44(ii) of the Constitution applies OR
          section 44(iii) of the Constitution applies OR
          section 44(iv) of the Constitution applies OR
          section 44(v) of the Constitution applies

RULE Constitution - Section 44(i) PROVIDES
     section 44(i) of the Constitution applies ONLY IF
          the nominee is under an acknowledgment of allegiance,
                obedience, or adherence to a foreign power OR
          the nominee is a subject of a foreign power OR
          the nominee is a citizen of a foreign power OR
          the nominee is entitled to the rights or privileges of
                a subject or a citizen of a foreign power

RULE Constitution - Section 44(ii) PROVIDES
     section 44(ii) of the Constitution applies ONLY IF
               the nominee is attainted of treason OR
          the nominee has been convicted and is under sentence or
                the nominee is subject to be sentenced, for any
                      offence punishable under the law of the
                            Commonwealth or of a State by
                            imprisonment for one year or longer

RULE Constitution - Section 44(iii) PROVIDES
     section 44(iii) of the Constitution applies ONLY IF
                the nominee is an undischarged bankrupt OR
                the nominee is insolvent

RULE Constitution - Section 44(iv) PROVIDES
     section 44(iv) of the Constitution applies ONLY IF
          the nominee holds an office of profit under the Crown AND/OR
          the nominee holds a pension payable during the pleasure
          of the Crown out of any of the revenues of the
          Commonwealth AND
          the exception to section 44(iv) of the
                Constitution does not apply
```

```
RULE Constitution – Section 44(v) PROVIDES
     section 44(v) of the Constitution applies ONLY IF
          the nominee has any direct or indirect pecuniary interest in
          any agreement with the Public Service of the Commonwealth AND
          all such interests are not as a member and in common with the
          other members of an incorporated company consisting of more
          than twenty-five persons

RULE Constitution – Section 44(iv) (exception) PROVIDES
     the exception to section 44(iv) of the Constitution applies ONLY IF
          the nominee holds the office of any of the Queen's
                Ministers of State for the Commonwealth OR
          the nominee holds the office of any of the Queen's
                Ministers for a State OR
          the nominee is a part-time officer or member of the Queen's
                navy or army OR
          the nominee is a part-time officer or member of the naval
                or military forces of the Commonwealth

RULE Acts Interpretation Act 1901 Schedule 1 PROVIDES
     the nominee is an adult ONLY IF
          the definition of "adult" under Schedule 1 of the Acts
                Interpretation Act 1901 Schedule 1 is met

RULE Acts Interpretation Act 1901 Schedule 1 PROVIDES
     the definition of "adult" under Schedule 1 of the Acts
                Interpretation Act 1901 Schedule 1 is met ONLY IF
          the age of the nominee IS GREATEREQUAL THAN 18

EXAMPLE Sykes v Cleary PROVIDES
the nominee holds an office of profit under the Crown ONLY IF
     the nominee was employed as a teacher at the time of nomination AND
     the nominee was appointed or employed by executive government AND
     the nominee was on unpaid leave at the time of nomination AND
     the nominee was not a judicial member of a tribunal

EXAMPLE Nash (No 2) PROVIDES
the nominee holds an office of profit under the Crown ONLY IF
     the nominee was not employed as a teacher at the time of nomination AND
     the nominee was appointed or employed by executive government AND
     the nominee was not on unpaid leave at the time of nomination AND
     the nominee was a judicial member of a tribunal

EXAMPLE Lambie (2018) PROVIDES
the nominee does not hold an office of profit under the Crown ONLY IF
     the nominee was not employed as a teacher at the time of nomination AND
     the nominee was not appointed or employed by executive government AND
     the nominee was not on unpaid leave at the time of nomination AND
     the nominee was not a judicial member of a tribunal

RULE PROVIDES
IF the nominee was employed as a teacher at the time of nomination THEN
     the nominee was appointed or employed by executive government

RULE PROVIDES
IF the nominee was a judicial member of a tribunal THEN
     the nominee was appointed or employed by executive government
```

2. Modern Slavery Act 2018 (Cth)

```
PERSON-THING the entity
DATE the day the Act received Royal Assent
DATE the day the Act was proclaimed

RULE Modern Slavery Act 2018 (Cth) Section 1 PROVIDES
the short title of this Act is the Modern Slavery Act 2018

RULE Modern Slavery Act 2018 (Cth) Section 2 PROVIDES
the date of commencement of section 1 and section 2 IS
    the day the Act received Royal Assent
ASSERT the date of commencement for sections other than section 1
    and section 2 IS the day the Act was proclaimed

RULE Modern Slavery Act 2018 (Cth) Section 3 PROVIDES
this Act requires entities based, or operating, in Australia, with
    an annual consolidated revenue of more than $100 million, to
    report annually on the risks of modern slavery in their operations and
    supply chains, and actions to address those risks AND
other entities based, or operating, in Australia may report voluntarily AND
the Commonwealth is required to report on behalf of non-corporate
    Commonwealth entities AND
the reporting requirements also apply to Commonwealth corporate entities
    and companies with an annual consolidated revenue of more
than $100 million AND
reports are kept by the Minister in a public repository known as the
    Modern Slavery Statements Register AND
statements on the register may be accessed by the public, free of charge,
    on the internet

RULE Modern Slavery Act 2018 (Cth) Section 4 "accounting standards" PROVIDES
"accounting standards" has the same meaning as in the Corporations Act 2001

RULE Modern Slavery Act 2018 (Cth) Section 4 "Australia" PROVIDES
"Australia", when used in a geographical sense, does include the
external Territories

RULE Modern Slavery Act 2018 (Cth) Section 4 "Australian entity" PROVIDES
the entity was an Australian entity during the reporting period ONLY IF
    subsection (a) of the definition of "Australian entity" applies OR
    subsection (b) of the definition of "Australian entity" applies OR
    subsection (c) of the definition of "Australian entity" applies OR
    subsection (d) of the definition of "Australian entity" applies

RULE Modern Slavery Act 2018 (Cth) Section 4 "Australian entity" (a) PROVIDES
subsection (a) of the definition of "Australian entity" applies ONLY IF
    the entity was a company which was a resident within the meaning of
    subsection 6(1) of the Income Tax Assessment Act 1936 during
    the reporting period

RULE Modern Slavery Act 2018 (Cth) Section 4 "Australian entity" (b) PROVIDES
subsection (b) of the definition of "Australian entity" applies ONLY IF
    the entity was a trust where the trust estate was a resident trust
    estate within the meaning of Division 6 of Part III of the
    Income Tax Assessment Act 1936 during the reporting period

RULE Modern Slavery Act 2018 (Cth) Section 4 "Australian entity" (c) PROVIDES
subsection (c) of the definition of "Australian entity" applies ONLY IF
    the entity was a corporate limited partnership which was a resident
        within the meaning of section 94T of the Income Tax
            Assessment Act 1936 during the reporting period

RULE Modern Slavery Act 2018 (Cth) Section 4 "Australian entity" (d) PROVIDES
subsection (d) of the definition of "Australian entity" applies ONLY IF
```

```
    subsection (d)(i) of the definition of "Australian entity" applies OR
    subsection (d)(ii) of the definition of "Australian entity" applies

RULE Modern Slavery Act 2018 (Cth) Section 4 "Australian entity" (d)(i) PROVIDES
subsection (d)(i) of the definition of "Australian entity" applies ONLY IF
    the entity was formed or incorporated in Australia

RULE Modern Slavery Act 2018 (Cth) Section 4 "Australian entity" (d)(ii)
PROVIDES
subsection (d)(ii) of the definition of "Australian entity" applies ONLY IF
    the central management or control of the entity was in Australia

RULE Modern Slavery Act 2018 (Cth) Section 4 "carries on business in Australia"
PROVIDES
the entity carries on business in Australia under the
    section 4 definition ONLY IF section 5(2) is satisfied

RULE Modern Slavery Act 2018 (Cth) Section 4 "consolidated revenue"
PROVIDES
IF the entity does not control another entity or entities within the
meaning of the accounting standards THEN
    the "consolidated revenue" of the entity as defined in section 4 IS
    "the total revenue of the entity for the reporting period"
ELSE
    the "consolidated revenue" of the entity as defined in section 4 IS
    "the total revenue of the entity and all of the controlled entities,
    considered as a group, for the reporting period of the controlling entity"

RULE Modern Slavery Act 2018 (Cth) Section 4 "control" PROVIDES
"control", of an entity by another entity, means control of the
entity within the meaning of the accounting standards

RULE Modern Slavery Act 2018 (Cth) Section 4 "entity" PROVIDES
the entity is an "entity" under section 4 ONLY IF
    the entity satisfies the definition of "entity" contained in section
    960-100 of the Income Tax Assessment Act 1997

RULE Modern Slavery Act 2018 (Cth) Section 4 "modern slavery" PROVIDES
the conduct is modern slavery ONLY IF
    subsection (a) of the section 4 definition of "modern slavery" applies OR
    subsection (b) of the section 4 definition of "modern slavery" applies OR
    subsection (c) of the section 4 definition of "modern slavery" applies OR
    subsection (d) of the section 4 definition of "modern slavery" applies

RULE Modern Slavery Act 2018 (Cth) Section 4 "modern slavery" (a) PROVIDES
subsection (a) of the section 4 definition of "modern slavery" applies ONLY IF
    the conduct would constitute an offence under Division 270 or 271 of the
    Criminal Code

RULE Modern Slavery Act 2018 (Cth) Section 4 "modern slavery" (b) PROVIDES
subsection (b) of the section 4 definition of "modern slavery" applies ONLY IF
    the conduct would constitute an offence under Division 270 or 271
    of the Criminal Code if the conduct took place in Australia

RULE Modern Slavery Act 2018 (Cth) Section 4 "modern slavery" (c) PROVIDES
subsection (c) of the section 4 definition of "modern slavery" applies ONLY IF
    the conduct would constitute trafficking in persons, as defined in
    Article 3 of the Protocol to Prevent, Suppress and Punish Trafficking
    in Persons, Especially Women and Children, supplementing the United
    Nations Convention against Transnational Organized Crime, done at
    New York on 15 November 2000 ([2005] ATS 27)

RULE Modern Slavery Act 2018 (Cth) Section 4 "modern slavery" (d) PROVIDES
subsection (d) of the section 4 definition of "modern slavery" applies ONLY IF
    the conduct would constitute the worst forms of child labour, as
    defined in Article 3 of the ILO Convention (No. 182) concerning the
```

```
    Prohibition and Immediate Action for the Elimination of the Worst
    Forms of Child Labour, done at Geneva on 17 June 1999 ([2007] ATS 38)

RULE Modern Slavery Act 2018 (Cth) Section 4 "modern slavery statement"
PROVIDES
the statement is a "modern slavery statement" as defined in section 4 ONLY IF
    the statement is a "modern slavery statement" as defined in section 12

RULE Modern Slavery Act 2018 (Cth) Section 4 "principal governing body"
PROVIDES
IF the entity is a company THEN
    the principal governing body of an entity is the board of directors
ELSE IF the entity is a superannuation fund THEN
    the principal governing body of an entity is the board of trustees
ELSE IF the entity is of a kind prescribed by the rules THEN
    the principal governing body of an entity is the prescribed body
    or member(s)
ELSE
    the principal governing body of an entity is the body, or group of
    members of the entity, with primary responsibility for the governance
    of the entity

RULE Modern Slavery Act 2018 (Cth) Section 4 "register" PROVIDES
"register" means the Modern Slavery Statements Register established
under section 18

RULE Modern Slavery Act 2018 (Cth) Section 4 "reporting entity" PROVIDES
the entity is a "reporting entity" under the section 4 definition ONLY IF
    section 5(1) is satisfied

RULE Modern Slavery Act 2018 (Cth) Section 4 "reporting period" PROVIDES
"reporting period", of an entity, means a financial year, or another annual
accounting period applicable to the entity, which starts after the
commencement of this section

RULE Modern Slavery Act 2018 (Cth) Section 4 "responsible member" PROVIDES
IF subsection (a) of the section 4 definition of "responsible member" applies
THEN
    the responsible member IS "an individual member of the entity's
    principal governing body who is authorised to sign modern slavery
    statements for the purposes of this Act"
ELSE IF subsection (b) of the section 4 definition of "responsible member"
applies THEN
    the responsible member IS "the trustee"
ELSE IF subsection (c) of the section 4 definition of "responsible member"
applies THEN
    the responsible member IS "the individual constituting the corporation"
ELSE IF subsection (d) of the section 4 definition of "responsible member"
applies THEN
    the responsible member IS "the administrator"
ELSE IF subsection (e) of the section 4 definition of "responsible member"
applies THEN
    the responsible member IS "the prescribed member of the entity"

RULE Modern Slavery Act 2018 (Cth) Section 4 "responsible member" (a)
PROVIDES
subsection (a) of the section 4 definition of "responsible member" applies
ONLY IF
    subsection (b) of the section 4 definition of "responsible member" does
    not apply AND
    subsection (c) of the section 4 definition of "responsible member"
    does not apply AND
    subsection (d) of the section 4 definition of "responsible member"
    does not apply AND
    subsection (e) of the section 4 definition of "responsible member"
    does not apply
```

```
RULE Modern Slavery Act 2018 (Cth) Section 4 "responsible member" (b)
PROVIDES
subsection (b) of the section 4 definition of "responsible member" applies
ONLY IF
    the entity is a trust administered by a sole trustee

RULE Modern Slavery Act 2018 (Cth) Section 4 "responsible member" (c)
PROVIDES
subsection (c) of the section 4 definition of "responsible member" applies
ONLY IF
    the entity is a corporation sole

RULE Modern Slavery Act 2018 (Cth) Section 4 "responsible member" (d)
PROVIDES
subsection (d) of the section 4 definition of "responsible member" applies
ONLY IF
    the entity is under administration within the meaning of the
    Corporations Act 2001

RULE Modern Slavery Act 2018 (Cth) Section 4 "responsible member" (e)
PROVIDES
subsection (e) of the section 4 definition of "responsible member" applies
ONLY IF
    the entity is of a kind prescribed by the rules

RULE Modern Slavery Act 2018 (Cth) Section 4 "rules" PROVIDES
"rules" means rules made by the Minister under section 25

RULE Modern Slavery Act 2018 (Cth) Section 5 PROVIDES
the entity is a reporting entity ONLY IF
    section 5(1) applies

RULE Modern Slavery Act 2018 (Cth) Section 5(1) PROVIDES
section 5(1) applies ONLY IF
    section 5(1)(a) applies OR
    section 5(1)(b) applies OR
    section 5(1)(c) applies OR
    section 5(1)(d) applies

RULE Modern Slavery Act 2018 (Cth) Section 5(1)(a) PROVIDES
section 5(1)(a) applies ONLY IF
    the entity had a consolidated revenue of at least $100 million
    for the reporting period AND
        section 5(1)(a)(i) applies AND/OR
        section 5(1)(a)(ii) applies

RULE Modern Slavery Act 2018 (Cth) Section 5(1)(a)(i) PROVIDES
section 5(1)(a)(i) applies ONLY IF
    the entity was an Australian entity during the reporting period

RULE Modern Slavery Act 2018 (Cth) Section 5(1)(a)(ii) PROVIDES
section 5(1)(a)(ii) applies ONLY IF
    the entity carried on business in Australia during the
    reporting period

RULE Modern Slavery Act 2018 (Cth) Section 5(1)(b) PROVIDES
section 5(1)(b) applies ONLY IF
    the reporting entity is the Commonwealth

RULE Modern Slavery Act 2018 (Cth) Section 5(1)(c) PROVIDES
section 5(1)(c) applies ONLY IF
    the entity was a corporate Commonwealth entity within the meaning of
    section 11 of the Public Governance, Performance and
    Accountability Act 2013 during the reporting period OR
    the entity was a Commonwealth company within the meaning of
```

```
    section 89 of the Public Governance, Performance and
    Accountability Act 2013 during the reporting period

RULE Modern Slavery Act 2018 (Cth) Section 5(1)(d) PROVIDES
section 5(1)(d) applies ONLY IF
    the entity has volunteered to comply with the requirements
    of the Act under section 6 for the reporting period

RULE Modern Slavery Act 2018 (Cth) Section 5(2) PROVIDES
the entity carried on business in Australia during the reporting period
ONLY IF
    section 5(2)(a) applies OR
    section 5(2)(b) applies

RULE Modern Slavery Act 2018 (Cth) Section 5(2)(a) PROVIDES
section 5(2)(a) applies ONLY IF
    the entity was a body corporate during the reporting period AND
    the entity carried on business in Australia within the meaning of
    section 21 of the Corporations Act 2001

RULE Modern Slavery Act 2018 (Cth) Section 5(2)(b) PROVIDES
section 5(2)(b) applies ONLY IF
    the entity was not a body corporate during the reporting period AND
    the entity carried on business in Australia within the meaning of
    section 21 of the Corporations Act 2001

RULE Modern Slavery Act 2018 (Cth) Section 6 PROVIDES
the entity has volunteered to comply with the requirements
of the Act under section 6 for the reporting period ONLY IF
    the entity has volunteered to comply with the requirements of the Act AND
    the entity has given written notice to the Minister in a manner
    and form approved by the Minister AND
    section 6(3) does not apply AND
    section 6(2) applies

RULE Modern Slavery Act 2018 (Cth) Section 6(2) PROVIDES
section 6(2) applies ONLY IF
    section 6(2)(a) applies OR
    section 6(2)(b) applies

RULE Modern Slavery Act 2018 (Cth) Section 6(2)(a) PROVIDES
section 6(2)(a) applies ONLY IF
    the entity was an Australian entity during the reporting period

RULE Modern Slavery Act 2018 (Cth) Section 6(2)(b) PROVIDES
section 6(2)(b) applies ONLY IF
    the entity carried on business in Australia during
    reporting period

RULE Modern Slavery Act 2018 (Cth) Section 6(3) PROVIDES
section 6(3) applies ONLY IF
    the entity has revoked the notice made under section 6(1) by giving
    written notice to the Minister before the start of the reporting period

RULE Modern Slavery Act 2018 (Cth) Section 7 PROVIDES
the Act has a jurisdictional basis ONLY IF
    section 7(1)(a) applies OR
    section 7(1)(b) applies

RULE Modern Slavery Act 2018 (Cth) Section 7(1)(a) PROVIDES
section 7(1)(a) applies ONLY IF
    the Act relates to trade and commerce under paragraph 51(i) of the
    Constitution OR
    the Act relates to census and statistics under paragraph 51(xi) of the
    Constitution OR
    the Act relates to aliens under paragraph 51(xix) of the Constitution OR
```

```
    the Act relates to corporations under paragraph 51(xx) of
    the Constitution OR
    the Act relates to marriage under paragraph 51(xxi) of the Constitution OR
    the Act relates to immigration under paragraph 51(xxvii) of the
    Constitution OR
    the Act relates to external affairs under paragraph 51(xxix) of the
    Constitution OR
    the Act relates to incidental matters under paragraph 51(xxxix) of the
    Constitution OR
    the Act is covered the executive powers provisions contained in section 61
    of the Constitution

RULE Modern Slavery Act 2018 (Cth) Section 7(1)(b) PROVIDES
section 7(1)(b) applies ONLY IF
    the Act is made under the implied legislative powers of the Commonwealth

RULE Modern Slavery Act 2018 (Cth) Section 7(2) PROVIDES
IF the Act gives effect to the International Convention to Suppress the
    Slave Trade and Slavery, done at Geneva on 25 September 1926
    ([1927] ATS 11) OR
    the Act gives effect to the ILO Convention (No. 29) concerning Forced
    or Compulsory Labour, done at Geneva on 28 June 1930 ([1933] ATS 21) OR
    the Act gives effect to the Supplementary Convention on the Abolition
    of Slavery, the Slave Trade, and Institutions and Practices similar to
    Slavery, done at Geneva on 7 September 1956 ([1958] ATS 3) OR
    the Act gives effect to the International Covenant on Civil and
    Political Rights, done at New York on 16 December 1966 ([1980] ATS 23) OR
    the Act gives effect to the Convention on the Elimination of All Forms
    of Discrimination Against Women, done at New York on 18 December 1979
    ([1983] ATS 9) OR
    the Act gives effect to the Convention on the Rights of the Child, done
    at New York on 20 November 1989 ([1991] ATS 4) OR
    the Act gives effect to the Protocol to Prevent, Suppress and Punish
    Trafficking in Persons, Especially Women and Children, supplementing the
    United Nations Convention against Transnational Organized Crime, done at
    New York on 15 November 2000 ([2005] ATS 27) OR
    the Act gives effect to the Optional Protocol to the Convention on the
    Rights of the Child on the Sale of Children, Child Prostitution and
    Child Pornography, done at New York on 25 May 2000 ([2007] ATS 6) OR
    the Act gives effect to the ILO Convention (No. 182) concerning the
    Prohibition and Immediate Action for the Elimination of the Worst Forms
    of Child Labour, done at Geneva on 17 June 1999 ([2007] ATS 38)
THEN
    the Act relates to external affairs under paragraph 51(xxix) of
    the Constitution

RULE Modern Slavery Act 2018 (Cth) Section 8 PROVIDES
this Act binds the Crown in right of the Commonwealth AND
this Act does not bind the Crown in right of a State, the Australian Capital
Territory or the Northern Territory

RULE Modern Slavery Act 2018 (Cth) Section 9 PROVIDES
this Act extends to every external Territory

RULE Modern Slavery Act 2018 (Cth) Section 10 PROVIDES
this Act extends to acts, omissions, matters and things outside Australia

RULE Modern Slavery Act 2018 (Cth) Section 11 PROVIDES
slavery statements must be given annually to the Minister, describing
    the risks of modern slavery in the operations and supply chains of
    reporting entities and entities owned or controlled by those entities AND
the statements must also include information about actions taken to
address those risks AND
joint modern slavery statements may be given on behalf of one or more
    reporting entities AND
the Minister must prepare an annual modern slavery statement on behalf
```

```
    of all non-corporate Commonwealth entities AND
the Minister may request an explanation from an entity about the
    entity's failure to comply with a requirement in relation to modern
    slavery statements, and may also request that the reporting
    entity undertake remedial action in relation to that requirement AND
the Minister may publish information about the failure to comply on the
    register or elsewhere, including the identity of an entity if an entity
    fails to comply with the request

RULE Modern Slavery Act 2018 (Cth) Section 12 PROVIDES
the requirement to report under section 12 has been met ONLY IF
    the entity has submitted a modern slavery statement under section 13 OR
    a joint modern slavery statement has been submitted that covers the entity
    under section 14 OR
    the entity is covered by a Commonwealth modern slavery statement
    under section 15

RULE Modern Slavery Act 2018 (Cth) Section 13 PROVIDES
the entity has submitted a modern slavery statement under section 13 ONLY IF
    the conditions set out in section 13(1) are met AND
    the conditions set out in section 13(2) are met

RULE Modern Slavery Act 2018 (Cth) Section 13(1) PROVIDES
the conditions set out in section 13(1) are met ONLY IF
    the entity is a reporting entity AND
    a joint modern slavery statement has not been submitted that covers
    the entity under section 14 AND
    the entity has lodged a modern slavery statement as a single
    reporting entity AND
    the conditions set out in section 13(2) are met

RULE Modern Slavery Act 2018 (Cth) Section 13(2) PROVIDES
the conditions set out in section 13(2) are met ONLY IF
    section 13(2)(a) is satisfied AND
    section 13(2)(b) is satisfied AND
    section 13(2)(c) is satisfied AND
    section 13(2)(d) is satisfied AND
    section 13(2)(e) is satisfied

RULE Modern Slavery Act 2018 (Cth) Section 13(2)(a) PROVIDES
section 13(2)(a) is satisfied ONLY IF
    the modern slavery statement complies with section 16

RULE Modern Slavery Act 2018 (Cth) Section 13(2)(b) PROVIDES
section 13(2)(b) is satisfied ONLY IF
    the statement has been prepared in a form approved by the Minister

RULE Modern Slavery Act 2018 (Cth) Section 13(2)(c) PROVIDES
section 13(2)(c) is satisfied ONLY IF
    the statement has been approved by the principal governing
    body of the entity

RULE Modern Slavery Act 2018 (Cth) Section 13(2)(d) PROVIDES
section 13(2)(d) is satisfied ONLY IF
    the statement has been signed by a responsible member of the entity

RULE Modern Slavery Act 2018 (Cth) Section 13(2)(e) PROVIDES
section 13(2)(e) is satisfied ONLY IF
    the statement was given to the Minister in a manner approved by
    the Minister AND
    the statement was given to the Minister
    within the required time after the end
    of the reporting period for the entity

RULE Modern Slavery Act 2018 (Cth) Section 14 PROVIDES
a joint modern slavery statement has been submitted that covers the
```

```
entity under section 14 ONLY IF
    section 14(1) applies AND
    section 14(2) applies

RULE Modern Slavery Act 2018 (Cth) Section 14(1) PROVIDES
section 14(1) applies ONLY IF
    the reporting entity is not the Commonwealth AND
    the entity is covered by a joint modern slavery statement

RULE Modern Slavery Act 2018 (Cth) Section 14(2) PROVIDES
section 14(2) applies ONLY IF
    section 14(2)(a) is satisfied AND
    section 14(2)(b) is satisfied AND
    section 14(2)(c) is satisfied AND
    section 14(2)(d) is satisfied AND
    section 14(2)(e) is satisfied AND
    section 14(2)(f) is satisfied


RULE Modern Slavery Act 2018 (Cth) Section 14(2)(a) PROVIDES
section 14(2)(a) is satisfied ONLY IF
    the modern slavery statement complies with section 16

RULE Modern Slavery Act 2018 (Cth) Section 14(2)(b) PROVIDES
section 14(2)(b) is satisfied ONLY IF
    the joint statement has been prepared in a form approved by the Minister

RULE Modern Slavery Act 2018 (Cth) Section 14(2)(c) PROVIDES
section 14(2)(c) is satisfied ONLY IF
    the joint statement has been prepared in consultation with each
    reporting entity covered by the statement

RULE Modern Slavery Act 2018 (Cth) Section 14(2)(d) PROVIDES
section 14(2)(d) is satisfied ONLY IF
    section 14(2)(d)(i) is satisfied OR
    section 14(2)(d)(ii) is satisfied OR
    section 14(2)(d)(iii) is satisfied

RULE Modern Slavery Act 2018 (Cth) Section 14(2)(d)(i) PROVIDES
section 14(2)(d)(i) is satisfied ONLY IF
    the joint statement has been approved by the principal governing body of
    each reporting entity covered by the statement

RULE Modern Slavery Act 2018 (Cth) Section 14(2)(d)(ii) PROVIDES
section 14(2)(d)(ii) is satisfied ONLY IF
    the joint statement has been approved by the principal governing body of
    an entity which is in a position, directly or indirectly, to influence or
    control each reporting entity covered by the statement

RULE Modern Slavery Act 2018 (Cth) Section 14(2)(d)(iii) PROVIDES
section 14(2)(d)(iii) is satisfied ONLY IF
    it is not practicable to have all entities covered by the statement to
    approve it AND
    it is not practicable to have a higher entity approve the statement on
    behalf of all entities covered by the joint statement AND
    at least one reporting entity covered by the statement has approved it

RULE Modern Slavery Act 2018 (Cth) Section 14(2)(e) PROVIDES
section 14(2)(e) is satisfied ONLY IF
    section 14(2)(e)(i) is satisfied OR
    section 14(2)(e)(ii) is satisfied OR
    section 14(2)(e)(iii) is satisfied
```

```
RULE Modern Slavery Act 2018 (Cth) Section 14(2)(e)(i) PROVIDES
section 14(2)(e)(i) is satisfied ONLY IF
    section 14(2)(d)(i) is satisfied AND
    the joint statement is signed by a responsible member of each reporting
    entity

RULE Modern Slavery Act 2018 (Cth) Section 14(2)(e)(ii) PROVIDES
section 14(2)(e)(ii) is satisfied ONLY IF
    section 14(2)(d)(ii) is satisfied AND
    the joint statement is signed by a responsible member of the higher entity

RULE Modern Slavery Act 2018 (Cth) Section 14(2)(e)(iii) PROVIDES
section 14(2)(e)(iii) is satisfied ONLY IF
    section 14(2)(d)(iii) is satisfied AND
    the joint statement is signed by a responsible member of the relevant
    entities

RULE Modern Slavery Act 2018 (Cth) Section 14(2)(f) PROVIDES
section 14(2)(f) is satisfied ONLY IF
    section 14(2)(f)(i) is satisfied OR
    section 14(2)(f)(ii) is satisfied

RULE Modern Slavery Act 2018 (Cth) Section 14(2)(f)(i) PROVIDES
section 14(2)(f)(i) is satisfied ONLY IF
    the statement was given to the Minister within the required time after the
end
    of the reporting period for the entity AND
    the statement was given to the Minister in a manner approved by
    the Minister

RULE Modern Slavery Act 2018 (Cth) Section 14(2)(f)(ii) PROVIDES
section 14(2)(f)(ii) is satisfied ONLY IF
    the statement was given to the Minister within a period prescribed by
    the rules

RULE Modern Slavery Act 2018 (Cth) Section 15 PROVIDES
the entity is covered by a Commonwealth modern slavery statement
under section 15 ONLY IF
   the entity is a Commonwealth Department, agency or other Commonwealth owned
   or controlled corporate or non-corporate entity AND
    section 15(1) applies AND
    section 15(2) applies

RULE Modern Slavery Act 2018 (Cth) Section 15(1) PROVIDES
section 15(1) applies ONLY IF
    the entity was not a non-corporate Commonwealth entity within the meaning
    of the Public Governance, Performance and Accountability Act 2013 AND
    the entity is included in a modern slavery statement prepared by the
    Commonwealth

RULE Modern Slavery Act 2018 (Cth) Section 15(2) PROVIDES
section 15(2) applies ONLY IF
    section 15(2)(a) applies AND
    section 15(2)(b) applies

RULE Modern Slavery Act 2018 (Cth) Section 15(2)(a) PROVIDES
section 15(2)(a) applies ONLY IF
    the modern slavery statement complies with section 16

RULE Modern Slavery Act 2018 (Cth) Section 15(2)(b) PROVIDES
section 15(2)(b) applies ONLY IF
    the Commonwealth modern slavery statement was prepared
    within the required time after the end of the reporting period

RULE Modern Slavery Act 2018 (Cth) Section 16 PROVIDES
the modern slavery statement complies with section 16 ONLY IF
```

```
    section 16(1) applies AND
    section 16(2) applies

RULE Modern Slavery Act 2018 (Cth) Section 16(1) PROVIDES
section 16(1) applies ONLY IF
    section 16(1)(a) is satisfied AND
    section 16(1)(b) is satisfied AND
    section 16(1)(c) is satisfied AND
    section 16(1)(d) is satisfied AND
    section 16(1)(e) is satisfied AND
    section 16(1)(f) is satisfied AND
    section 16(1)(g) is satisfied

RULE Modern Slavery Act 2018 (Cth) Section 16(1)(a) PROVIDES
section 16(1)(a) is satisfied ONLY IF
    the modern slavery statement identifies the entity

RULE Modern Slavery Act 2018 (Cth) Section 16(1)(b) PROVIDES
section 16(1)(b) is satisfied ONLY IF
    the statement describes the structure, operations and supply chains
    of the entity

RULE Modern Slavery Act 2018 (Cth) Section 16(1)(c) PROVIDES
section 16(1)(c) is satisfied ONLY IF
    the statement describes the risks of modern slavery practices in the
    operations and supply chains of the entity, and any entities that
    the entity owns or controls

RULE Modern Slavery Act 2018 (Cth) Section 16(1)(d) PROVIDES
section 16(1)(d) is satisfied ONLY IF
    the statement describes the actions taken by the entity and
    any entity that the entity owns or controls, to assess and address
    those risks, including due diligence and remediation processes

RULE Modern Slavery Act 2018 (Cth) Section 16(1)(e) PROVIDES
section 16(1)(e) is satisfied ONLY IF
    the statement describes how the entity assesses the
    effectiveness of such actions

RULE Modern Slavery Act 2018 (Cth) Section 16(1)(f) PROVIDES
section 16(1)(f) is satisfied ONLY IF
    section 16(1)(f)(i) is satisfied AND
    section 16(1)(f)(ii) is satisfied

RULE Modern Slavery Act 2018 (Cth) Section 16(1)(f)(i) PROVIDES
section 16(1)(f)(i) is satisfied ONLY IF
    the statement describes the process of consultation with any entities
    that the entity owns or controls

RULE Modern Slavery Act 2018 (Cth) Section 16(1)(f)(ii) PROVIDES
section 16(1)(f)(ii) is satisfied ONLY IF
    a joint modern slavery statement has not been submitted that covers the
    entity under section 14 OR
    the statement describes the process of consultation with
    the reporting entity giving the statement

RULE Modern Slavery Act 2018 (Cth) Section 16(1)(g) PROVIDES
section 16(1)(g) is satisfied ONLY IF
    the statement includes all other information considered to be relevant

RULE Modern Slavery Act 2018 (Cth) Section 16(2) PROVIDES
section 16(2) applies ONLY IF
    the entity is covered by a Commonwealth modern slavery statement
    under section 15 OR
    section 16(2)(a) applies OR
    section 16(2)(b) applies
```

```
RULE Modern Slavery Act 2018 (Cth) Section 16(2)(a) PROVIDES
section 16(2)(a) applies ONLY IF
    the entity has not submitted a modern slavery statement under section 13 OR
    the statement includes details of approval by the principal governing
    body of the entity

RULE Modern Slavery Act 2018 (Cth) Section 16(2)(b) PROVIDES
section 16(2)(b) applies ONLY IF
    a joint modern slavery statement has not been submitted that covers the
    entity under section 14 OR
        section 16(2)(b)(i) applies AND
        section 16(2)(b)(ii) applies

RULE Modern Slavery Act 2018 (Cth) Section 16(2)(b)(i) PROVIDES
section 16(2)(b)(i) applies ONLY IF
    the joint modern statement includes details of approval by the
    relevant principal governing body or bodies

RULE Modern Slavery Act 2018 (Cth) Section 16(2)(b)(ii) PROVIDES
section 16(2)(b)(ii) applies ONLY IF
    section 14(2)(d)(iii) does not apply OR
    the statement does include an explanation of why it is not
    practicable to comply with subsections 14(2)(d)(i) or (ii)

RULE Modern Slavery Act 2018 (Cth) Section 16A PROVIDES
IF the entity has not submitted a modern slavery statement under
section 13 AND
a joint modern slavery statement has not been submitted that covers the
entity under section 14 THEN BEGIN
    the Minister may give a written request to the entity under section
    16A(1)(a) to provide an explanation for the failure to comply within
    a specified period of 28 days or longer after the request is given AND
    the Minister may give a written request to the entity under
    section 16A(1)(b) to undertake specified remedial action in accordance
    with the request within a specified period of 28 days or longer after
    the request is given AND
    pursuant to section 16A(2), the Minister may extend, or further extend,
    a period specified in a request under section 16A(1) by written notice
    given to the entity before or after the end of the specified
    period AND
    as required by section 16A(3), the request must include a statement
    of the effect of subsections (2) and (4) to (6)
END ELSE
    a request from the Minister to the entity in relation to compliance is
    not necessary

RULE Modern Slavery Act 2018 (Cth) Section 16A(4) PROVIDES
IF the entity has failed to comply with a request under section 16A(1) THEN
the Minister may publish the identity of the entity AND
the Minister may publish the identities of all entities included by a
joint modern slavery statement AND
the Minister may publish the date that a request under section 16A(1) was
made and details of any extensions under section 16A(2) AND
the Minister may publish details of the explanation or remedial action
requested, and the period or periods specified in the request AND
the Minister may publish the reasons why the Minister is satisfied that the
entity has failed to comply with the request

RULE Modern Slavery Act 2018 (Cth) Section 16A(5) PROVIDES
the entity has failed to comply with a request under section 16A(1) ONLY IF
    no explanation has been given in response to the request within the
    period specified OR
    no remedial action is undertaken in response to the request within
    the period specified
```

```
RULE Modern Slavery Act 2018 (Cth) Section 17 PROVIDES
this Part establishes the Modern Slavery Statements Register AND
the register is made available to the public on the internet AND
modern slavery statements are registered by the Minister AND
revised versions of registered modern slavery statements can be
registered in some circumstances

RULE Modern Slavery Act 2018 (Cth) Section 18 PROVIDES
the Minister must maintain a register of modern slavery statements, to be
known as the Modern Slavery Statements Register AND
the register must be made available for public inspection, without
charge, on the internet

RULE Modern Slavery Act 2018 (Cth) Section 19 PROVIDES
a modern slavery statement can be registered ONLY IF
    the Minister must register the modern slavery statement under
    section 19(1) OR
    the Minister may register the modern slavery statement under
    section 19(2)

RULE Modern Slavery Act 2018 (Cth) Section 19(1) PROVIDES
the Minister must register the modern slavery statement under
section 19(1) ONLY IF
    the entity has submitted a modern slavery statement under section 13 OR
    a joint modern slavery statement has been submitted that covers the
    entity under section 14 OR
    the entity is covered by a Commonwealth modern slavery statement
    under section 15

RULE Modern Slavery Act 2018 (Cth) Section 19(2) PROVIDES
the Minister may register the modern slavery statement under
section 19(2) ONLY IF
    the statement was given for the purposes of compliance with section
    13 or 14 (including a statement given in response to a request under
    section 16A) even if the entity giving the statement does not comply
    with the requirements of subsection 13(2) or 14(2)

RULE Modern Slavery Act 2018 (Cth) Section 20 PROVIDES
section 20 needs work

RULE Modern Slavery Act 2018 (Cth) Section 21 PROVIDES
this Part deals with things done by an unincorporated entity AND
this Part deals with the Minister's capacity to delegate powers and
functions under this Act AND
this Part deals with annual reports about the implementation of this Act
this Part deals with the 3-year review of this Act AND
this Part deals with the power to make rules

RULE Modern Slavery Act 2018 (Cth) Section 22 PROVIDES
IF this Act requires or allows a thing to be done by an entity that is an
unincorporated body THEN
    the thing must, or may, be done by a responsible member of the entity
    on the entity's behalf

RULE Modern Slavery Act 2018 (Cth) Section 23 PROVIDES
the Minister may, by writing, delegate all or any of the Minister's powers
and functions under this Act to an SES employee, or acting SES employee,
in the Department ONLY IF the power does not make, vary or revoke the rules
ASSERT the delegate must comply with any directions of the Minister in
exercising powers or functions under a delegation


RULE Modern Slavery Act 2018 (Cth) Section 24 PROVIDES
section 24 needs work

RULE Modern Slavery Act 2018 (Cth) Section 25 PROVIDES
```

```
section 25 needs work

/*
 *  other Acts, mechanics and common-sense
 */

RULE Income Tax Assessment Act 1936 Section 6 "resident" (1)(b) PROVIDES
the entity was a company which was a resident within the meaning of
subsection 6(1) of the Income Tax Assessment Act 1936 during
the reporting period ONLY IF
    the entity was a company during the reporting period AND BEGIN
    the entity was formed or incorporated in Australia OR
        the entity was not formed or incorporated in Australia AND
        the entity carried on business in Australia during the
        reporting period AND
            the central management or control of the entity was
            in Australia AND/OR
            the entity's voting power was controlled by shareholders who
            were residents of Australia
    END

RULE Corporations Act 2001 Section 21 PROVIDES
the entity carried on business in Australia within the meaning of
section 21 of the Corporations Act 2001 ONLY IF
    section 21(1) of the Corporations Act 2001 applies OR
    section 21(2) of the Corporations Act 2001 applies AND
    the business activities in Australia are not restricted to only
    those listed in section 21(3) of the Corporations Act
    2001

RULE Corporations Act 2001 Section 21(1) PROVIDES
section 21(1) of the Corporations Act 2001 applies ONLY IF
    the entity has a place of business in Australia, or in State or
    Territory

RULE Corporations Act 2001 Section 21(2) PROVIDES
section 21(2) of the Corporations Act 2001 applies ONLY IF
    section 21(2)(a) of the Corporations Act 2001 applies OR
    section 21(2)(b) of the Corporations Act 2001 applies

RULE Corporations Act 2001 Section 21(2)(a) PROVIDES
section 21(2)(a) of the Corporations Act 2001 applies ONLY IF
    the entity has established or used a share transfer office or a
    share registration office in Australia, or in a State or Territory

RULE Corporations Act 2001 Section 21(2)(b) PROVIDES
section 21(2)(b) of the Corporations Act 2001 applies ONLY IF
    the entity has administered, managed or otherwise dealt with property
    situated in Australia as an agent, legal representative or trustee,
    whether by employees or agents or otherwise

RULE Corporations Act 2001 Section 21(3) PROVIDES
    the business activities in Australia are restricted to only
    those listed in section 21(3) of the Corporations Act 2001 ONLY IF
    the business activities in Australia involve being a party to
    proceedings or effecting settlement of a proceeding,
    claim or dispute AND/OR
    the business activities in Australia involve holding meetings of
    directors or shareholders or conducting other activities relating to
    the entity's internal affairs AND/OR
    the business activities in Australia involve maintaining a bank
    account AND/OR
    the business activities in Australia involve effecting a sale through
    an independent contractor AND/OR
    the business activities in Australia involve soliciting or procuring an
    order that becomes a binding contract only if the order is accepted
```

```
    outside of Australia AND/OR
    the business activities in Australia involves the creation of evidence
    of a debt or creates a charge on property AND/OR
    the business activities in Australia involves securing or collecting
    debts or enforcing rights in regard to securities relating to such
    debts AND/OR
    the business activities in Australia involves conducting an isolated
    transaction that is completed with 31 days, not being one of a number
    of similar transactions repeated from time to time AND/OR
    the business activities in Australia involve investing funds or holding
    any property AND
    the entity does not have any other business activities in Australia
    other than those mentioned in section 21(3) of the Corporations Act
    2001

DATE the first day of the reporting period
DATE the last day of the relevant financial year for the entity
DATE the date of commencement for sections other than section 1 and section 2

RULE Modern Slavery Act 2018 (Cth) Royal Assent PROVIDES
the day the Act received Royal Assent IS 10 December 2018

RULE Modern Slavery Commencement Proclamation 2018 F2018N00189 PROVIDES
the day the Act was proclaimed IS 1 January 2019

RULE Reporting Period PROVIDES
the reporting period is covered by the Act ONLY IF
    the first day of the reporting period is after the commencement of the Act

RULE Reporting Period After Commencement of the Act PROVIDES
the first day of the reporting period is after the commencement of the Act
ONLY IF
    the first day of the reporting period GREATER THAN
        the date of commencement for sections other than section 1 and
        section 2

RULE First Day of the Reporting Period PROVIDES
the first day of the reporting period IS
    the last day of the relevant financial year for the entity
        MINUS 1 YEAR PLUS 1 DAY

RULE Last Day to Submit PROVIDES
IF the reporting period is covered by the Act THEN
IF the Australian Government has extended the deadline for submission of
modern slavery statements by 3 months due to the COVID-19 pandemic THEN
    the last day to provide the modern slavery statement for the entity IS
    the last day of the relevant financial year for the entity PLUS 9 MONTHS
ELSE
    the last day to provide the modern slavery statement for the entity IS
    the last day of the relevant financial year for the entity PLUS 3 MONTHS

RULE COVID-19 Extension PROVIDES
the Australian Government has extended the deadline for submission of
modern slavery statements by 3 months due to the COVID-19 pandemic

RULE Commonwealth Statement Within Required Time PROVIDES
the Commonwealth modern slavery statement was prepared
within the required time after the end of the reporting period ONLY IF
    the date the Commonwealth modern slavery statement was
    prepared IS LESSEQUAL THAN the last day to provide the modern
    slavery statement for the entity

RULE Statement Within Required Time PROVIDES
the statement was given to the Minister within the required time after the
end of the reporting period for the entity ONLY IF
    the date the modern slavery statement was given to the Minister
```

```
    IS LESSEQUAL THAN the last day to provide the modern slavery statement
    for the entity

RULE Time to Submit Has Not Lapsed PROVIDES
the entity still has time to submit a modern slavery statement that
meets the requirements of the Act ONLY IF
    the last day to provide the modern slavery statement for the entity
    IS GREATEREQUAL today

GOAL RULE Modern Slavery Act 2018 (Cth) PROVIDES
the entity has complied with the Act ONLY IF
    the entity is not a reporting entity OR
    the entity is a reporting entity AND
    the reporting period is not covered by the Act OR
    the entity is a reporting entity AND
    the reporting period is covered by the Act AND
    the requirement to report under section 12 has been met OR
    the entity is a reporting entity AND
    the reporting period is covered by the Act AND
    the requirement to report under section 12 has not been met AND
    the entity still has time to submit a modern slavery
            statement that meets the requirements of the Act

RULE Entity is Commonwealth of Australia PROVIDES
the reporting entity is the Commonwealth ONLY IF
    the name of the entity EQUALS "the Commonwealth" OR
    the name of the entity EQUALS "Commonwealth" OR
    the name of the entity EQUALS "the Commonwealth of Australia" OR
    the entity is a Commonwealth Department, agency or other Commonwealth
    owned or controlled corporate or non-corporate entity

RULE Companies Are Bodies Corporate PROVIDES
IF the entity was a company during the reporting period THEN
    the entity was a body corporate during the reporting period

RULE Individuals and Companies are not trusts PROVIDES
IF the entity was a company during the reporting period OR
the entity is a person THEN
    the entity was not a trust where the trust estate was a resident
    trust estate within the meaning of Division 6 of Part III of the
    Income Tax Assessment Act 1936 during the reporting period

RULE Individuals and Companies are not corporate limited partnerships PROVIDES
IF the entity was a company during the reporting period OR
the entity is a person THEN
    the entity was not a corporate limited partnership which was a
    resident within the meaning of section 94T of the Income Tax
    Assessment Act 1936 during the reporting period

RULE Individuals and Companies are not Partnerships PROVIDES
IF the entity was a company during the reporting period OR
the entity is a person THEN
    the entity was not a partnership during the reporting period

RULE Individuals and Companies not non-Corporate Commonwealth Entities PROVIDES
IF the entity was a company during the reporting period OR
the entity is a person THEN
    the entity was not a non-corporate Commonwealth entity within the meaning
    of the Public Governance, Performance and Accountability Act 2013

RULE Individuals and Private Companies are not Commonwealth Entities PROVIDES
IF the entity was a company during the reporting period OR
the entity is a person THEN
    the entity is not a Commonwealth Department, agency or other Commonwealth
    owned or controlled corporate or non-corporate entity
```

```
RULE Individuals are not Companies PROVIDES
IF the entity is a person THEN
    the entity was not a company during the reporting period

RULE Individuals are not Bodies Corporate PROVIDES
IF the entity is a person THEN
    the entity was not a body corporate during the reporting period

RULE Individuals are not Formed or Incorporated PROVIDES
IF the entity is a person THEN
    the entity was not formed or incorporated in Australia

RULE Individual does not have Central Management or Control PROVIDES
IF the entity is a person THEN
    the central management or control of the entity was not in Australia

RULE Individual is not a Corporate Commonwealth Entity PROVIDES
IF the entity is a person THEN
    the entity was not a corporate Commonwealth entity within the meaning of
    section 11 of the Public Governance, Performance and Accountability
    Act 2013 during the reporting period

RULE Individual is not a Commonwealth Company PROVIDES
IF the entity is a person THEN
    the entity was not a Commonwealth company within the meaning of section
    89 of the Public Governance, Performance and Accountability Act 2013
    during the reporting period

BOOLEAN the entity is a person
PROMPT is the entity an individual
TRANSLATE true AS the entity is an individual
TRANSLATE false AS the entity is not an individual
```

3. Community Gaming Regulations 2020 (NSW)

```
PERSON the person conducting the gaming activity
THING the benefiting organisation
THING the gaming activity
THING the organiser
THING the registered club
THING the fund-raising organisation

MONEY the amount of the gross proceeds of the gaming activity
MONEY the amount returned to participants
MONEY the total value of all of the prizes
MONEY the total cost of prizes and expenses
MONEY the amount paid to the benefiting organisation
MONEY the maximum amount of money payable as a separate prize


RULE Community Gaming Regulation 2020 Regulation 1 PROVIDES
this Regulation is the Community Gaming Regulation 2020

RULE Community Gaming Regulation 2020 Regulation 2 PROVIDES
this Regulation commences on 1 July 2020 AND
this Regulation is required to be published on the NSW legislation website

RULE Community Gaming Regulation 2020 Regulation 2 Commencement PROVIDES
the gaming activity is covered by the Regulations ONLY IF
    the date of the gaming activity is after the commencement of the
    Regulations on 1 July 2020

RULE Community Gaming Regulation 2020 Regulation 2 Commencement Calculation
PROVIDES
the date of the gaming activity is after the commencement of the Regulations
on 1 July 2020 ONLY IF
    the date of the gaming activity GREATEREQUAL THAN 1 July 2020

RULE Community Gaming Regulation 2020 Regulation 3 "art union" PROVIDES
the gaming activity is run by an "art union" under the regulation 3
definition ONLY IF
    the organiser is a voluntary association formed to purchase prizes to
    be awarded by lot among members of the association AND
    clause (a) of the definition of "art union" applies AND/OR
    clause (b) of the definition of "art union" applies AND/OR
    clause (c) of the definition of "art union" applies

RULE Community Gaming Regulation 2020 Regulation 3 "art union" (a) PROVIDES
clause (a) of the definition of "art union" applies ONLY IF
    the purpose of raising funds is to support a charitable organisation
    under the regulation 3 definition

RULE Community Gaming Regulation 2020 Regulation 3 "art union" (b) PROVIDES
clause (b) of the definition of "art union" applies ONLY IF
    the purpose of raising funds is to support a non-profit organisation
    under the regulation 3 definition

RULE Community Gaming Regulation 2020 Regulation 3 "art union" (c) PROVIDES
clause (c) of the definition of "art union" applies ONLY IF
    the purpose of raising funds is to support an object of genuinely public
    or charitable character

RULE Community Gaming Regulation 2020 Regulation 3 "art union gaming
activity" PROVIDES
    "art union gaming activity" has the meaning given by regulation 4

RULE Community Gaming Regulation 2020 Regulation 3 "benefiting organisation"
PROVIDES
"benefiting organisation" is defined as the person or body for whose benefit a
```

```
gaming activity is conducted

RULE Community Gaming Regulation 2020 Regulation 3 "bingo"
PROVIDES
"bingo" is defined in regulation 5(1)

RULE Community Gaming Regulation 2020 Regulation 3 "calcutta" PROVIDES
the gaming activity is a "calcutta" under the regulation 3 definition
ONLY IF
    clause (a) of the definition of "calcutta" in regulation 4 applies AND
    clause (b) of the definition of "calcutta" in regulation 4 applies AND
    clause (c) of the definition of "calcutta" in regulation 4 applies

RULE Community Gaming Regulation 2020 Regulation 3 "calcutta" (a) PROVIDES
clause (a) of the definition of "calcutta" in regulation 4 applies ONLY IF
    the gaming activity is a game in which each participant pays a fee for
    a chance to win by lot a right in respect of a competitor in a sporting
    or racing event

RULE Community Gaming Regulation 2020 Regulation 3 "calcutta" (b) PROVIDES
clause (b) of the definition of "calcutta" in regulation 4 applies ONLY IF
    the rights in respect of a competitor in a sporting or racing event are
    auctioned

RULE Community Gaming Regulation 2020 Regulation 3 "calcutta" (c) PROVIDES
clause (c) of the definition of "calcutta" in regulation 4 applies ONLY IF
    clause (c)(i) of the definition of "calcutta" in regulation 4 applies AND
    clause (c)(ii) of the definition of "calcutta" in regulation 4 applies

RULE Community Gaming Regulation 2020 Regulation 3 "calcutta" (c)(i)
PROVIDES
clause (c)(i) of the definition of "calcutta" in regulation 4 applies ONLY IF
    the holder of each right is entitled to elect to sell the right at the
    auction and receive half of the proceeds of sale

RULE Community Gaming Regulation 2020 Regulation 3 "calcutta" (c)(ii)
PROVIDES
clause (c)(ii) of the definition of "calcutta" in regulation 4 applies
ONLY IF
    the holder of each right is entitled to elect to retain the right by
    making (and paying half of) the highest bid

RULE Community Gaming Regulation 2020 Regulation 3 "charitable
organisation" (1) PROVIDES
the purpose of raising funds is to support a charitable organisation under
the regulation 3 definition ONLY IF
    the purpose of raising funds is to support an incorporated or
    unincorporated charitable body formed for, or to benefit, a benevolent,
    philanthropic or patriotic purpose

RULE Community Gaming Regulation 2020 Regulation 3 "charitable
organisation" (2) PROVIDES
the fund-raising organisation is a "charitable organisation" under the
regulation 3 definition ONLY IF
    the organisation is an incorporated or unincorporated charitable body
    formed for, or to benefit, a benevolent, philanthropic, or patriotic
    purpose

RULE Community Gaming Regulation 2020 Regulation 3 "charitable
organisation" (3) PROVIDES
the gaming activity is conducted by or on behalf of a charitable
organisation under the regulation 3 definition ONLY IF
    the gaming activity is conducted by or on behalf of an incorporated or
    unincorporated charitable body formed for, or to benefit, a benevolent,
    philanthropic or patriotic purpose
```

```
RULE Community Gaming Regulation 2020 Regulation 3 "charity housie" PROVIDES
"charity housie" means a gaming activity permitted under regulation 5(2)

RULE Community Gaming Regulation 2020 Regulation 3 "club bingo" PROVIDES
"club bingo" means a gaming activity permitted under regulation 5(4)

RULE Community Gaming Regulation 2020 Regulation 3 "chocolate wheel" PROVIDES
the gaming activity is a "chocolate wheel" under the regulation 3 definition
ONLY IF
    clause (a) of the definition of "chocolate wheel" applies AND
    clause (b) of the definition of "chocolate wheel" applies AND
    clause (c) of the definition of "chocolate wheel" applies

RULE Community Gaming Regulation 2020 Regulation 3 "chocolate wheel" (a)
PROVIDES
clause (a) of the definition of "chocolate wheel" applies ONLY IF
    numbered tickets are sold to participants

RULE Community Gaming Regulation 2020 Regulation 3 "chocolate wheel" (b)
PROVIDES
clause (b) of the definition of "chocolate wheel" applies ONLY IF
    a wheel is spun to determine the prize winners based on
    the numbers on the tickets corresponding to numbers on the wheel

RULE Community Gaming Regulation 2020 Regulation 3 "chocolate wheel" (c)
PROVIDES
clause (c) of the definition of "chocolate wheel" applies ONLY IF
    a participant wins a prize if the participant holds a ticket that
    corresponds to the number on which the wheel comes to rest after being
    spun

RULE Community Gaming Regulation 2020 Regulation 3 "draw lottery" PROVIDES
the gaming activity is a "draw lottery" under the regulation 3 definition
ONLY IF
    the gaming activity is a "lottery" under the regulation 3 definition AND
    clause (a) of the definition of "draw lottery" applies AND
    clause (b) of the definition of "draw lottery" applies AND
    clause (c) of the definition of "draw lottery" applies

RULE Community Gaming Regulation 2020 Regulation 3 "draw lottery" (a)
PROVIDES
clause (a) of the definition of "draw lottery" applies ONLY IF
    numbered tickets are sold to participants in the lottery

RULE Community Gaming Regulation 2020 Regulation 3 "draw lottery" (b)
PROVIDES
clause (b) of the definition of "draw lottery" applies ONLY IF
    a draw is held in which one or more numbers (corresponding to the
    numbers on the tickets) is or are selected at random

RULE Community Gaming Regulation 2020 Regulation 3 "draw lottery" (c)
PROVIDES
clause (c) of the definition of "draw lottery" applies ONLY IF
    prizes are distributed to the participants holding the tickets
    corresponding to the numbers selected

RULE Community Gaming Regulation 2020 Regulation 3 "free lottery" PROVIDES
the gaming activity is a "free lottery" under the regulation 3 definition
ONLY IF
    the gaming activity is a permitted free lottery under regulation 10

RULE Community Gaming Regulation 2020 Regulation 3 "gross proceeds" PROVIDES
"gross proceeds" of a gaming activity means the total receipts received
from the sale of tickets or from other payments by participants and
donations before deduction of expenses
```

```
RULE Community Gaming Regulation 2020 Regulation 3 "housie" PROVIDES
"housie" is defined in regulation 5(1)

RULE Community Gaming Regulation 2020 Regulation 3 "liquor component"
PROVIDES
"liquor component" means a part of a prize in a gaming activity consisting
of or including liquor

RULE Community Gaming Regulation 2020 Regulation 3 "lottery" PROVIDES
the gaming activity is a "lottery" under the regulation 3 definition ONLY IF
    the gaming activity is a game in which prizes are awarded by lot

RULE Community Gaming Regulation 2020 Regulation 3 "lucky envelopes" PROVIDES
the gaming activity is "lucky envelopes" under the regulation 3 definition
ONLY IF
    clause (a) of the definition of "lucky envelopes" applies AND
    clause (b) of the definition of "lucky envelopes" applies

RULE Community Gaming Regulation 2020 Regulation 3 "lucky envelopes" (a)
PROVIDES
clause (a) of the definition of "lucky envelopes" applies ONLY IF
    the player purchases a ticket that contains a concealed number

RULE Community Gaming Regulation 2020 Regulation 3 "lucky envelopes" (b)
PROVIDES
clause (b) of the definition of "lucky envelopes" applies ONLY IF
    the player wins a prize if the concealed number matches a number
    displayed at the point of sale of the tickets

RULE Community Gaming Regulation 2020 Regulation 3 "mini-numbers lottery"
PROVIDES
the gaming activity is a "mini-numbers lottery" under the regulation 3
definition ONLY IF
    the gaming activity is a game in which participants choose or attempt to
    forecast, from designated numbers, fewer numbers to be drawn on a random
    basis

RULE Community Gaming Regulation 2020 Regulation 3 "no-draw lottery" PROVIDES
the gaming activity is a "no-draw lottery" under the regulation 3
definition ONLY IF
    the gaming activity is a listed example of a "no-draw lottery" AND/OR
    the gaming activity is not a listed example of a "no-draw lottery" AND
    clause (a) of the definition of "no-draw lottery" applies AND
    clause (b) of the definition of "no-draw lottery" applies AND
    clause (c) of the definition of "no-draw lottery" applies AND
    clause (d) of the definition of "no-draw lottery" applies

RULE Community Gaming Regulation 2020 Regulation 3 "no-draw lottery" (a)
PROVIDES
clause (a) of the definition of "no-draw lottery" applies ONLY IF
    participants purchase a right to participate in the gaming activity

RULE Community Gaming Regulation 2020 Regulation 3 "no-draw lottery" (b)
PROVIDES
clause (b) of the definition of "no-draw lottery" applies ONLY IF
    each ticket, or card or board conferring the right to participate,
    has a hidden symbol (or a set of hidden symbols) that can be exposed
    by removing a covering of paper or other opaque material

RULE Community Gaming Regulation 2020 Regulation 3 "no-draw lottery" (c)
PROVIDES
clause (c) of the definition of "no-draw lottery" applies ONLY IF
     the hidden symbols (or sets of hidden symbols) including prizewinning
     symbols (or sets of prizewinning symbols) are randomly distributed
     among the tickets or cards
```

```
RULE Community Gaming Regulation 2020 Regulation 3 "no-draw lottery" (d)
PROVIDES
clause (d) of the definition of "no-draw lottery" applies ONLY IF
    a participant, on exposing a hidden symbol (or set of hidden symbols)
    that accords with another symbol (or set of symbols) specified in the
    rules of the gaming activity (whether or not displayed on the ticket or
    card), has a right under those rules to receive a specified prize

RULE Community Gaming Regulation 2020 Regulation 3 "no-draw lottery" examples
PROVIDES
the gaming activity is a listed example of a "no-draw lottery" ONLY IF
    the gaming activity is a break-open lottery, scratch lottery or football
    double

RULE Community Gaming Regulation 2020 Regulation 3 "no-profit
organisation" (1) PROVIDES
the purpose of raising funds is to support a non-profit organisation under
the regulation 3 definition ONLY IF
    the purpose of raising funds is to support a non-profit incorporated or
    unincorporated body not formed or conducted for private gain

RULE Community Gaming Regulation 2020 Regulation 3 "no-profit
organisation" (2) PROVIDES
the fund-raising organisation is a "non-profit organisation" under the
regulation 3
definition
ONLY IF
    the organisation is a non-profit incorporated or unincorporated body
    not formed or conducted for private gain

RULE Community Gaming Regulation 2020 Regulation 3 "no-profit
organisation" (3) PROVIDES
the gaming activity is conducted by or on behalf of a non-profit
organisation under the regulation 3 definition ONLY IF
    the gaming activity is conducted by or on behalf of a non-profit
    incorporated or unincorporated body not formed or conducted for
    private gain

RULE Community Gaming Regulation 2020 Regulation 3 "progressive lottery"
PROVIDES
the gaming activity is a "progressive lottery" under the regulation 3
definition ONLY IF
    clause (a) of the definition of progressive lottery applies OR
    clause (b) of the definition of progressive lottery applies OR
    the gaming activity is not a tipping competition or other gaming activity
    conducted along substantially similar lines AND
    the gaming activity is a progressive lottery where a number
    of draws may be conducted on various dates over a stipulated period of
    time AND
    the gaming activity is not a "sweep" under the regulation 3
    definition AND
    the gaming activity is not a "calcutta" under the regulation 3
    definition

RULE Community Gaming Regulation 2020 Regulation 3 "progressive lottery" (a)
PROVIDES
clause (a) of the definition of progressive lottery applies ONLY IF
    the gaming activity is a hundred club, silver circles or other
    activity conducted along substantially similar lines

RULE Community Gaming Regulation 2020 Regulation 3 "progressive lottery" (b)
PROVIDES
clause (b) of the definition of progressive lottery applies ONLY IF
    the gaming activity is a tipping competition or other gaming activity
    conducted along substantially similar lines AND
    clause (b)(i) of the definition of "tipping competition" applies AND
```

```
    clause (b)(ii) of the definition of "tipping competition" applies AND
    clause (b)(iii) of the definition of "tipping competition" applies AND
    clause (b)(iv) of the definition of "tipping competition" applies

RULE Community Gaming Regulation 2020 Regulation 3 "progressive
lottery" (b)(i) PROVIDES
clause (b)(i) of the definition of "tipping competition" applies ONLY IF
    the player predicts the outcome or results of a sporting or other
    contingency

RULE Community Gaming Regulation 2020 Regulation 3 "progressive
lottery" (b)(ii) PROVIDES
clause (b)(ii) of the definition of "tipping competition" applies ONLY IF
    points are awarded for successful predictions

RULE Community Gaming Regulation 2020 Regulation 3 "progressive
lottery" (b)(iii) PROVIDES
clause (b)(iii) of the definition of "tipping competition" applies ONLY IF
    the prizes are wholly distributed in accordance with the rules of the
    competition

RULE Community Gaming Regulation 2020 Regulation 3 "progressive
lottery" (b)(iv) PROVIDES
clause (b)(iv) of the definition of "tipping competition" applies ONLY IF
    periodical prizes may be awarded (in accordance with the rules of the
    competition

RULE Community Gaming Regulation 2020 Regulation 3 "publish"
PROVIDES
"publish" has the same meaning as in section 9 of the
Community Gaming Act 2018

RULE Community Gaming Regulation 2020 Regulation 3 "registered club"
PROVIDES
the fund-raising organisation is a "registered club" under the Regulation 3
definition
ONLY IF
    the organisation is a club that holds a club licence under the
    Liquor Act 2007

RULE Community Gaming Regulation 2020 Regulation 3 "session"
PROVIDES
"session" of a gaming activity means a number of games of the activity
played in succession on the same occasion at the same place

RULE Community Gaming Regulation 2020 Regulation 3 "sweep"
PROVIDES
the gaming activity is a "sweep" under the regulation 3 definition ONLY IF
    the gaming activity is a game in which each participant pays a fee for a
    chance to win by lot a right in respect of a competitor in a sporting
    or racing event AND
    the rights in respect of a competitor in a sporting or racing event are
    not auctioned AND
    the gaming activity is not a "calcutta" under the regulation 3 definition

RULE Community Gaming Regulation 2020 Regulation 3 "symbol"
PROVIDES
"symbol" includes amount, word or picture

RULE Community Gaming Regulation 2020 Regulation 3 "the Act"
PROVIDES
"the Act" means the Community Gaming Act 2018

RULE Community Gaming Regulation 2020 Regulation 3 "ticket"
PROVIDES
"ticket" includes a right to participate in a gaming activity
```

```
RULE Community Gaming Regulation 2020 Regulation 3 "trade promotion gaming
activity"
PROVIDES
the gaming activity is a "trade promotion gaming activity" under the
regulation 3 definition ONLY IF
    the gaming activity is conducted for the purpose of promoting goods or
    services provided by a business

RULE Community Gaming Regulation 2020 Part 2 - Permitted gaming
activities PROVIDES
the gaming activity is a permitted gaming activity under Part 2 ONLY IF
    the gaming activity is covered by the Regulations AND
    the gaming activity is a permitted "art union gaming activity" under
    regulation 4 AND/OR
    the gaming activity is permitted "housie" or "bingo"
    under regulation 5 AND/OR
    the gaming activity is a permitted "draw lottery" under
    regulation 6 AND/OR
    the gaming activity is a permitted no-draw lottery under
    regulation 7 AND/OR
    the gaming activity is a permitted mini-numbers lottery under
    regulation 8 AND/OR
    the gaming activity is a permitted progressive lottery under
    regulation 9 AND/OR
    the gaming activity is a permitted free lottery under
    regulation 10 AND/OR
    the gaming activity is a permitted promotional raffle conducted by a
    registered club under regulation 11 AND/OR
    the gaming activity is a permitted gaming activity for charitable
    purposes under regulation 12 AND/OR
    the gaming activity is a permitted sweep or calcutta under
    regulation 13 AND/OR
    the gaming activity is a permitted trade promotion gaming activity under
    regulation 14

RULE Community Gaming Regulation 2020 Regulation 4 - Art union gaming
activities PROVIDES
the gaming activity is a permitted "art union gaming activity" under
regulation 4 ONLY IF
    the gaming activity is an art union gaming activity AND
    regulation 4(d) applies AND
    regulation 4(a) applies AND
    regulation 4(b) applies AND
    regulation 4(c) applies

RULE Community Gaming Regulation 2020 Regulation 4 "art union gaming
activity" PROVIDES
the gaming activity is an art union gaming activity ONLY IF
    the gaming activity is run by an "art union" under the regulation 3
    definition AND
    the gaming activity is a game in which prizes are awarded by lot

RULE Community Gaming Regulation 2020 Regulation 4(a) PROVIDES
regulation 4(a) applies ONLY IF
    at least 30% of the gross proceeds of the gaming activity are paid
    to the benefiting organisation

RULE Community Gaming Regulation 2020 Regulation 4(a) Calculation PROVIDES
at least 30% of the gross proceeds of the gaming activity are paid to
the benefiting organisation ONLY IF
    the amount of the gross proceeds of the gaming activity TIMES 0.3
    LESSEQUAL THAN the amount paid to the benefiting organisation

RULE Community Gaming Regulation 2020 Regulation 4(b) PROVIDES
regulation 4(b) applies ONLY IF
```

```
    the total value of all of the prizes exceeds $30,000

RULE Community Gaming Regulation 2020 Regulation 4(b) Calculation PROVIDES
the total value of all of the prizes exceeds $30,000 ONLY IF
    the total value of all of the prizes GREATER THAN $30,000

RULE Community Gaming Regulation 2020 Regulation 4(c) PROVIDES
regulation 4(c) applies ONLY IF
    the maximum amount of money payable as a separate prize does not
    exceed $30,000

RULE Community Gaming Regulation 2020 Regulation 4(c) Calculation PROVIDES
the maximum amount of money payable as a separate prize does not
exceed $30,000 ONLY IF
    the maximum amount of money payable as a separate prize
    LESSEQUAL THAN $30,000

RULE Community Gaming Regulation 2020 Regulation 4(d) PROVIDES
regulation 4(d) applies ONLY IF
    the person conducting the gaming activity holds an authority for
    the gaming activity AND
    the authority is in force AND
    the gaming activity is conducted in accordance with the authority

RULE Community Gaming Regulation 2020 Regulation 5 - Housie or bingo
PROVIDES
the gaming activity is permitted "housie" or "bingo" under
regulation 5
ONLY IF
    the gaming activity meets the definition of "housie" or "bingo" under
    regulation 5(1) AND
    the gaming activity is Charity housie under regulation 5(2) AND/OR
    the gaming activity is Social housie under regulation 5(3) AND/OR
    the gaming activity is Club bingo under regulation 5(4)

RULE Community Gaming Regulation 2020 Regulation 5(1) PROVIDES
the gaming activity meets the definition of "housie" or "bingo" under
regulation 5(1) ONLY IF
    regulation 5(1)(a) is satisfied AND
    regulation 5(1)(b) is satisfied AND
    regulation 5(1)(c) is satisfied

RULE Community Gaming Regulation 2020 Regulation 5(1)(a) PROVIDES
regulation 5(1)(a) is satisfied ONLY IF
    the gaming activity is housie or bingo that is played by one or more
    participants using cards or a device with numbered spaces or symbols

RULE Community Gaming Regulation 2020 Regulation 5(1)(b) PROVIDES
regulation 5(1)(b) is satisfied ONLY IF
    numbered spaces or symbols identified randomly and announced during play
    are marked off by each participant who has a card or device on which the
    numbered space or symbol is displayed

RULE Community Gaming Regulation 2020 Regulation 5(1)(c) PROVIDES
regulation 5(1)(c) is satisfied ONLY IF
    the game is won by the participant who is first able to mark off all
    numbered spaces or symbols on the card or device that are required to be
    marked off for a win

RULE Community Gaming Regulation 2020 Regulation 5(2) PROVIDES
the gaming activity is Charity housie under regulation 5(2) ONLY IF
    regulation 5(2)(a) is satisfied AND
    regulation 5(2)(b) is satisfied AND
    regulation 5(2)(c) is satisfied AND
    regulation 5(2)(d) is satisfied AND
    regulation 5(2)(e) is satisfied
```

```
RULE Community Gaming Regulation 2020 Regulation 5(2)(a) PROVIDES
regulation 5(2)(a) is satisfied ONLY IF
    the gaming activity is conducted by or on behalf of a charitable
    organisation under the regulation 3 definition

RULE Community Gaming Regulation 2020 Regulation 5(2)(b) PROVIDES
regulation 5(2)(b) is satisfied ONLY IF
    at least 12.5% of the gross proceeds of the gaming activity are
    paid to the benefiting organisation

RULE Community Gaming Regulation 2020 Regulation 5(2)(b) Calculation PROVIDES
at least 12.5% of the gross proceeds of the gaming activity are
paid to the benefiting organisation ONLY IF
    the amount paid to the benefiting organisation GREATEREQUAL THAN
    the amount of the gross proceeds of the gaming activity TIMES 0.125

RULE Community Gaming Regulation 2020 Regulation 5(2)(c) PROVIDES
regulation 5(2)(c) is satisfied ONLY IF
    the total value of the expenses of conducting the gaming activity
    (excluding the cost of prizes) is not more than 12.5% of the gross
    proceeds of the gaming activity

RULE Community Gaming Regulation 2020 Regulation 5(2)(c) Calculation PROVIDES
the total value of the expenses of conducting the gaming activity
(excluding the cost of prizes) is not more than 12.5% of the gross
proceeds of the gaming activity ONLY IF
    the total value of the expenses of conducting the gaming activity
    (excluding the cost of prizes) LESSEQUAL THAN the amount of the
    gross proceeds of the gaming activity TIMES 0.125

RULE Community Gaming Regulation 2020 Regulation 5(2)(d) PROVIDES
regulation 5(2)(d) is satisfied ONLY IF
    the total value of all of the prizes for one session of the gaming
    activity is not more than $10,000 AND
    the total value of all of the prizes for one session is not more than 75%
    of the gross proceeds from the gaming activity

RULE Community Gaming Regulation 2020 Regulation 5(2)(d) Calculation 1
PROVIDES
the total value of all of the prizes for one session of the gaming activity
is not more than $10,000 ONLY IF
    the total value of all of the prizes for one session of the
    gaming activity LESSEQUAL THAN $10,000

RULE Community Gaming Regulation 2020 Regulation 5(2)(d) Calculation 2
PROVIDES
the total value of all of the prizes for one session is not more than 75%
of the gross proceeds from the gaming activity ONLY IF
    the total value of all of the prizes for one session of the
    gaming activity LESSEQUAL THAN the amount of the gross proceeds of
    the gaming activity TIMES 0.75

RULE Community Gaming Regulation 2020 Regulation 5(2)(e) PROVIDES
regulation 5(2)(e) is satisfied ONLY IF
    no more than 48 tickets are permitted to be sold to the same participant

RULE Community Gaming Regulation 2020 Regulation 5(2)(e) Calculation PROVIDES
no more than 48 tickets are permitted to be sold to the same participant
ONLY IF
    the maximum number of tickets that can be sold to the same participant
    LESSEQUAL THAN 48

RULE Community Gaming Regulation 2020 Regulation 5(3) PROVIDES
the gaming activity is Social housie under regulation 5(3) ONLY IF
    regulation 5(3)(a) is satisfied AND
```

```
    regulation 5(3)(b) is satisfied AND
    regulation 5(3)(c) is satisfied AND/OR
    regulation 5(3)(d) applies AND
    regulation 5(3)(e) is satisfied

RULE Community Gaming Regulation 2020 Regulation 5(3)(a) PROVIDES
regulation 5(3)(a) is satisfied ONLY IF
    the gaming activity is conducted solely for social purposes

RULE Community Gaming Regulation 2020 Regulation 5(3)(b) PROVIDES
regulation 5(3)(b) is satisfied ONLY IF
    the gaming activity is not conducted on premises to which a licence
    under the Liquor Act 2007 relates

RULE Community Gaming Regulation 2020 Regulation 5(3)(c) PROVIDES
regulation 5(3)(c) is satisfied ONLY IF
    the total value of all of the prizes for one session of the
    gaming activity is not more than $40 AND
    regulation 5(3)(d) does not apply

RULE Community Gaming Regulation 2020 Regulation 5(3)(c) Calculation PROVIDES
the total value of all of the prizes for one session of the
gaming activity is not more than $40 ONLY IF
    the total value of all of the prizes for one session of the
    gaming activity LESSEQUAL THAN $40

RULE Community Gaming Regulation 2020 Regulation 5(3)(d) PROVIDES
regulation 5(3)(d) applies ONLY IF
    the value of any jackpot prize is not more than $200

RULE Community Gaming Regulation 2020 Regulation 5(3)(d) Calculation PROVIDES
the value of any jackpot prize is not more than $200 ONLY IF
    the value of any jackpot prize LESSEQUAL THAN $200

RULE Community Gaming Regulation 2020 Regulation 5(3)(e) PROVIDES
regulation 5(3)(e) is satisfied ONLY IF
    the total amount invested by participants in a session of the gaming
    activity is returned to participants, after the costs of prizes and
    expenses of conducting the session are deducted

RULE Community Gaming Regulation 2020 Regulation 5(3)(e) Calculation PROVIDES
the total amount invested by participants in a session of the gaming
activity is returned to participants, after the costs of prizes and
expenses of conducting the session are deducted ONLY IF
    the total amount invested by participants in a session of the gaming
    activity MINUS the total cost of prizes and expenses EQUALS
    the amount returned to participants

RULE Community Gaming Regulation 2020 Regulation 5(4) PROVIDES
the gaming activity is Club bingo under regulation 5(4) ONLY IF
    regulation 5(4)(a) is satisfied AND
    regulation 5(4)(d) is satisfied AND
    regulation 5(4)(b) is satisfied AND/OR
    regulation 5(4)(c) applies

RULE Community Gaming Regulation 2020 Regulation 5(4)(a) PROVIDES
regulation 5(4)(a) is satisfied ONLY IF
    the gaming activity is conducted by or on the authority of a
    registered club on club premises for the purpose of attracting patronage
    to the club's facilities

RULE Community Gaming Regulation 2020 Regulation 5(4)(b) PROVIDES
regulation 5(4)(b) is satisfied ONLY IF
    the total value of all of the prizes for one game of the
    gaming activity is not more than $70
```

```
RULE Community Gaming Regulation 2020 Regulation 5(4)(b) Calculation PROVIDES
the total value of all of the prizes for one game of the gaming
activity is not more than $70 ONLY IF
    the total value of all of the prizes for one game of the gaming
    activity LESSEQUAL THAN $70

RULE Community Gaming Regulation 2020 Regulation 5(4)(c) PROVIDES
regulation 5(4)(c) applies ONLY IF
    a single bonus prize of more than $70 in value is offered at the end
    of a session of club bingo

RULE Community Gaming Regulation 2020 Regulation 5(4)(d) PROVIDES
regulation 5(4)(d) is satisfied ONLY IF
    the prizes do not consist of or include money

RULE Community Gaming Regulation 2020 Regulation 6 - Draw lotteries
PROVIDES
the gaming activity is a permitted "draw lottery" under regulation 6 ONLY IF
    the gaming activity is a "draw lottery" under the regulation 3
    definition AND
    regulation 6(a) is satisfied AND
    regulation 6(b) is satisfied AND
    regulation 6(c) is satisfied

RULE Community Gaming Regulation 2020 Regulation 6(a) PROVIDES
regulation 6(a) is satisfied ONLY IF
    the gaming activity is conducted by or on behalf of a charitable
    organisation under the regulation 3 definition OR
    the gaming activity is conducted by or on behalf of a non-profit
    organisation under the regulation 3 definition

RULE Community Gaming Regulation 2020 Regulation 6(b) PROVIDES
regulation 6(b) is satisfied ONLY IF
    at least 40% of the gross proceeds of the gaming activity are paid
    to the benefiting organisation

RULE Community Gaming Regulation 2020 Regulation 6(b) Calculation PROVIDES
at least 40% of the gross proceeds of the gaming activity are paid
to the benefiting organisation ONLY IF
    the amount of the gross proceeds of the gaming activity TIMES 0.4
    LESSEQUAL THAN the amount paid to the benefiting organisation

RULE Community Gaming Regulation 2020 Regulation 6(c) PROVIDES
regulation 6(c) is satisfied ONLY IF
    the total value of all of the prizes is not more than $30,000

RULE Community Gaming Regulation 2020 Regulation 6(c) CALCULATION PROVIDES
the total value of all of the prizes is not more than $30,000 ONLY IF
    the total value of all of the prizes LESSEQUAL THAN $30,000

RULE Community Gaming Regulation 2020 Regulation 7 - No-draw lotteries
PROVIDES
the gaming activity is a permitted no-draw lottery under regulation 7 ONLY IF
    the gaming activity is a "no-draw lottery" under the regulation 3
    definition AND
    regulation 7(a) is satisfied AND
    regulation 7(b) is satisfied AND
    regulation 7(c) is satisfied AND
    regulation 7(d) is satisfied

RULE Community Gaming Regulation 2020 Regulation 7(a) PROVIDES
regulation 7(a) is satisfied ONLY IF
    the gaming activity is conducted by or on behalf of a charitable
    organisation under the regulation 3 definition OR
    the gaming activity is conducted by or on behalf of a non-profit
    organisation under the regulation 3 definition
```

```
RULE Community Gaming Regulation 2020 Regulation 7(b) PROVIDES
regulation 7(b) is satisfied ONLY IF
    at least 40% of the gross proceeds of the gaming activity are
    paid to the benefiting organisation

RULE Community Gaming Regulation 2020 Regulation 7(c) PROVIDES
regulation 7(c) is satisfied ONLY IF
the total value of all of the prizes is not more than $5,000

RULE Community Gaming Regulation 2020 Regulation 7(c) Calculation PROVIDES
the total value of all of the prizes is not more than $5,000 ONLY IF
    the total value of all of the prizes LESSEQUAL THAN $5,000

RULE Community Gaming Regulation 2020 Regulation 7(d) PROVIDES
regulation 7(d) is satisfied ONLY IF
    the total number of tickets produced or obtained for sale for the
    lottery is not more than 3,000

RULE Community Gaming Regulation 2020 Regulation 7(d) Calculation PROVIDES
the total number of tickets produced or obtained for sale for the
lottery is not more than 3,000 ONLY IF
    the total number of tickets produced or obtained for sale
    LESSEQUAL THAN 3000

RULE Community Gaming Regulation 2020 Regulation 8 - Mini-numbers
lotteries PROVIDES
the gaming activity is a permitted mini-numbers lottery under
regulation 8 ONLY IF
    the gaming activity is a "mini-numbers lottery" under the regulation 3
    definition AND
    regulation 8(a) is satisfied AND
    regulation 8(b) is satisfied AND
    regulation 8(c) is satisfied AND
    regulation 8(d) is satisfied

RULE Community Gaming Regulation 2020 Regulation 8(a) PROVIDES
regulation 8(a) is satisfied ONLY IF
    the gaming activity is conducted by or on behalf of a charitable
    organisation under the regulation 3 definition OR
    the gaming activity is conducted by or on behalf of a non-profit
    organisation under the regulation 3 definition

RULE Community Gaming Regulation 2020 Regulation 8(b) PROVIDES
regulation 8(b) is satisfied ONLY IF
    at least 40% of the gross proceeds of the gaming activity are paid
    to the benefiting organisation

RULE Community Gaming Regulation 2020 Regulation 8(c) PROVIDES
regulation 8(c) is satisfied ONLY IF
    the total value of all of the prizes for one session of the gaming
    activity is not more than $20,000

RULE Community Gaming Regulation 2020 Regulation 8(c) Calculation PROVIDES
the total value of all of the prizes for one session of the gaming
activity is not more than $20,000 ONLY IF
    the total value of all of the prizes for one session of the gaming
    activity LESSEQUAL THAN $20,000

RULE Community Gaming Regulation 2020 Regulation 8(d) PROVIDES
regulation 8(d) is satisfied ONLY IF
    the total value of all of the prizes for one session of the gaming
    activity is at least 50% of the gross proceeds of the mini-numbers
    lottery

RULE Community Gaming Regulation 2020 Regulation 8(d) Calculation PROVIDES
```

```
the total value of all of the prizes for one session of the gaming
activity is at least 50% of the gross proceeds of the mini-numbers
lottery ONLY IF
    the total value of all of the prizes for one session of the gaming
    activity GREATEREQUAL THAN the amount of the gross proceeds of the gaming
    activity TIMES 0.5

RULE Community Gaming Regulation 2020 Regulation 9 - Progressive
lotteries PROVIDES
the gaming activity is a permitted progressive lottery under
regulation 9 ONLY IF
    the gaming activity is a "progressive lottery" under the regulation 3
    definition AND
    regulation 9(a) is satisfied AND
    regulation 9(b) is satisfied

RULE Community Gaming Regulation 2020 Regulation 9(a) PROVIDES
regulation 9(a) is satisfied ONLY IF
    the maximum amount of money payable as a prize is not more than $7,000

RULE Community Gaming Regulation 2020 Regulation 9(a) Calculation PROVIDES
the maximum amount of money payable as a prize is not more than $7,000
ONLY IF
    the maximum amount of money payable as a prize LESSEQUAL THAN $7,000

RULE Community Gaming Regulation 2020 Regulation 9(b) PROVIDES
regulation 9(b) is satisfied ONLY IF
    the total value of all of the prizes is not more than $30,000 OR
    the person conducting the gaming activity holds an authority to do so AND
    the authority is in force AND
    the gaming activity is conducted in accordance with the authority

RULE Community Gaming Regulation 2020 Regulation 9(b) Calculation PROVIDES
the total value of all of the prizes is not more than $30,000 ONLY IF
    the total value of all of the prizes LESSEQUAL THAN $30,000

RULE Community Gaming Regulation 2020 Regulation 10 - Free lotteries
PROVIDES
the gaming activity is a permitted free lottery under regulation 10 ONLY IF
    the gaming activity is a "lottery" under the regulation 3 definition AND
    regulation 10(b) is satisfied AND
    regulation 10(c) is satisfied AND
    regulation 10(a) is satisfied AND
    regulation 10(d) is satisfied

RULE Community Gaming Regulation 2020 Regulation 10(a) PROVIDES
regulation 10(a) is satisfied ONLY IF
    the gaming activity is not a "progressive lottery" under the regulation 3
    definition AND
    the gaming activity is not a "trade promotion gaming activity" under the
    regulation 3 definition

RULE Community Gaming Regulation 2020 Regulation 10(b) PROVIDES
regulation 10(b) is satisfied ONLY IF
    participation in the lottery is free

RULE Community Gaming Regulation 2020 Regulation 10(c) PROVIDES
regulation 10(c) is satisfied ONLY IF
    the prizes do not consist of or include money

RULE Community Gaming Regulation 2020 Regulation 10(d) PROVIDES
regulation 10(d) is satisfied ONLY IF
    the total value of all of the prizes for the gaming activity is not more
    than $30,000

RULE Community Gaming Regulation 2020 Regulation 11 - Promotional
```

```
raffles conducted by registered clubs PROVIDES
the gaming activity is a permitted promotional raffle conducted by a
registered club under regulation 11 ONLY IF
    the gaming activity is a raffle AND
    the raffle is conducted by or on the authority of a registered club AND
    regulation 11(a) is satisfied AND
    regulation 11(b) is satisfied AND
    regulation 11(e) is satisfied AND
    regulation 11(c) is satisfied AND
    regulation 11(d) is satisfied

RULE Community Gaming Regulation 2020 Regulation 11(a) PROVIDES
regulation 11(a) is satisfied ONLY IF
    the gaming activity is conducted on the registered club's premises

RULE Community Gaming Regulation 2020 Regulation 11(b) PROVIDES
regulation 11(b) is satisfied ONLY IF
    the gaming activity is conducted for the purpose of attracting
    patronage to the registered club's facilities

RULE Community Gaming Regulation 2020 Regulation 11(c) PROVIDES
regulation 11(c) is satisfied ONLY IF
    at least 90% of the gross proceeds of the gaming activity are used to
    meet the cost of the prizes in the gaming activity or other similar
    gaming activities

RULE Community Gaming Regulation 2020 Regulation 11(c) Calculation PROVIDES
at least 90% of the gross proceeds of the gaming activity are used to
meet the cost of the prizes in the gaming activity or other similar
gaming activities ONLY IF
    the cost of the prizes in the gaming activity GREATEREQUAL THAN
    the amount of the gross proceeds of the gaming activity TIMES 0.9

RULE Community Gaming Regulation 2020 Regulation 11(d) PROVIDES
regulation 11(d) is satisfied ONLY IF
    the total value of all of the prizes for one session of the gaming
    activity is not more than $5,000

RULE Community Gaming Regulation 2020 Regulation 11(d) Calculation PROVIDES
the total value of all of the prizes for one session of the gaming
activity is not more than $5,000 ONLY IF
    the total value of all of the prizes for one session of the gaming
    activity LESSEQUAL THAN $5,000

RULE Community Gaming Regulation 2020 Regulation 11(e) PROVIDES
regulation 11(e) is satisfied ONLY IF
    the prizes do not consist of or include money

RULE Community Gaming Regulation 2020 Regulation 12 - Other gaming
activities for charitable purposes PROVIDES
the gaming activity is a permitted gaming activity for charitable purposes
under regulation 12 ONLY IF
    the gaming activity is a "chocolate wheel" under the regulation 3
    definition AND/OR
    the gaming activity is "lucky envelopes" under the regulation 3
    definition AND/OR BEGIN
    the gaming activity is of some other type of gaming activity AND
    regulation 12(2) does not apply END AND
    regulation 12(1) is satisfied

RULE Community Gaming Regulation 2020 Regulation 12(1) PROVIDES
regulation 12(1) is satisfied ONLY IF
    regulation 12(1)(a) is satisfied AND
    regulation 12(1)(b) is satisfied

RULE Community Gaming Regulation 2020 Regulation 12(1)(a) PROVIDES
```

```
regulation 12(1)(a) is satisfied ONLY IF
    at least 40% of the gross proceeds of the gaming activity are
    paid to the benefiting organisation

RULE Community Gaming Regulation 2020 Regulation 12(1)(b) PROVIDES
regulation 12(1)(b) is satisfied ONLY IF
    the total value of all of the prizes for one session of the gaming
    activity is not more than $5,000

RULE Community Gaming Regulation 2020 Regulation 12(2) PROVIDES
regulation 12(2) applies ONLY IF
    the gaming activity is a "lottery" under the regulation 3 definition OR
    the gaming activity is a "sweep" under the regulation 3 definition OR
    the gaming activity is an art union gaming activity OR
    the gaming activity meets the definition of "housie" or "bingo" under
    regulation 5(1) OR
    the gaming activity is a "calcutta" under the regulation 3 definition

RULE Community Gaming Regulation 2020 Regulation 13 - Sweeps and
calcuttas PROVIDES
the gaming activity is a permitted sweep or calcutta under regulation 13
ONLY IF
    the gaming activity is a "sweep" under the regulation 3 definition AND/OR
    the gaming activity is a "calcutta" under the regulation 3 definition AND
    regulation 13(1) applies AND
    regulation 13(3) is satisfied AND
    regulation 13(4) is satisfied

RULE Community Gaming Regulation 2020 Regulation 13(1) PROVIDES
regulation 13(1) applies ONLY IF
    the gaming activity is conducted according to regulation 13(1)(a) AND
    regulation 13(1)(b) applies AND/OR
    regulation 13(2) applies AND
    regulation 13(1)(c) is satisfied

RULE Community Gaming Regulation 2020 Regulation 13(1)(a) PROVIDES
the gaming activity is conducted according to regulation 13(1)(a) ONLY IF
    regulation 13(1)(a)(i) applies OR
    regulation 13(1)(a)(ii) applies

RULE Community Gaming Regulation 2020 Regulation 13(1)(a)(i) PROVIDES
regulation 13(1)(a)(i) applies ONLY IF
    the gaming activity is conducted for or on behalf of a fund-raising
    organisation AND
    the fund-raising organisation is an "approved fund-raising
    organisation" as defined in regulation 13(6)

RULE Community Gaming Regulation 2020 Regulation 13(1)(a)(ii) PROVIDES
regulation 13(1)(a)(ii) applies ONLY IF
    the gaming activity is conducted for social purposes

RULE Community Gaming Regulation 2020 Regulation 13(1)(b) PROVIDES
regulation 13(1)(b) applies ONLY IF
    there is not a payment or benefit payable for the right to participate
    in the gaming activity, other than the stake money for the gaming
    activity

RULE Community Gaming Regulation 2020 Regulation 13(1)(c) PROVIDES
regulation 13(1)(c) is satisfied ONLY IF
    the total value of all of the prizes for the gaming activity is not more
    than $30,000 OR
    the person conducting the gaming activity holds an authority to do so AND
    the authority is in force AND
    the gaming activity is conducted in accordance with the authority

RULE Community Gaming Regulation 2020 Regulation 13(1)(c) Calculation
```

```
PROVIDES
the total value of all of the prizes for the gaming activity is not more
than $30,000 ONLY IF
    the total value of all of the prizes LESSEQUAL THAN $30,000

RULE Community Gaming Regulation 2020 Regulation 13(2) PROVIDES
regulation 13(2) applies ONLY IF
    the payment is a fee charged for entry to a venue or function at which
    the gaming activity is conducted AND
    the payment is not related to the gaming activity AND
    the payment is usually charged for entry to the venue or function

RULE Community Gaming Regulation 2020 Regulation 13(3) PROVIDES
regulation 13(3) is satisfied ONLY IF
    regulation 13(1)(a)(i) does not apply OR
    regulation 13(3)(a) is satisfied AND
    regulation 13(3)(b) is satisfied

RULE Community Gaming Regulation 2020 Regulation 13(3)(a) PROVIDES
regulation 13(3)(a) is satisfied ONLY IF
    a reasonable amount of the gross proceeds is paid to the
    approved fund-raising organisation

RULE Community Gaming Regulation 2020 Regulation 13(3)(b) PROVIDES
regulation 13(3)(b) is satisfied ONLY IF
    the amount is agreed in writing before the sweep or calcutta is conducted

RULE Community Gaming Regulation 2020 Regulation 13(4) PROVIDES
regulation 13(4) is satisfied ONLY IF
    regulation 13(1)(a)(i) applies OR
    the gross proceeds are distributed to the holders of the rights in
    respect of the successful participants in the event to which the sweep
    or calcutta relates

RULE Community Gaming Regulation 2020 Regulation 13(5) PROVIDES
the amount of proceeds remaining after payment of prize money and the
costs and expenses of the gaming activity may be paid for the purposes
of the fund-raising organisation, even if that amount exceeds the
agreed amount of the gross proceeds

RULE Community Gaming Regulation 2020 Regulation 13(6) PROVIDES
the fund-raising organisation is an "approved fund-raising organisation"
as defined in regulation 13(6) ONLY IF
    regulation 13(6)(a) applies OR
    regulation 13(6)(b) applies OR
    regulation 13(6)(c) applies OR
    regulation 13(6)(d) applies OR
    regulation 13(6)(e) applies OR
    regulation 13(6)(f) applies

RULE Community Gaming Regulation 2020 Regulation 13(6)(a) PROVIDES
regulation 13(6)(a) applies ONLY IF
    the fund-raising organisation is a "charitable organisation" under
    the regulation 3 definition OR
    the fund-raising organisation is a "non-profit organisation" under the
    regulation 3 definition

RULE Community Gaming Regulation 2020 Regulation 13(6)(b) PROVIDES
regulation 13(6)(b) applies ONLY IF
    the fund-raising organisation is a political party or trade union

RULE Community Gaming Regulation 2020 Regulation 13(6)(c) PROVIDES
regulation 13(6)(c) applies ONLY IF
    the fund-raising organisation is a "registered club" under the
    Regulation 3 definition
```

```
RULE Community Gaming Regulation 2020 Regulation 13(6)(d) PROVIDES
regulation 13(6)(d) applies ONLY IF
    the fund-raising organisation is a club registered under the Rules
    of Racing of Racing New South Wales

RULE Community Gaming Regulation 2020 Regulation 13(6)(e) PROVIDES
regulation 13(6)(e) applies ONLY IF
    the fund-raising organisation is a greyhound racing club within the
    meaning of the Greyhound Racing Act 2017

RULE Community Gaming Regulation 2020 Regulation 13(6)(f) PROVIDES
regulation 13(6)(f) applies ONLY IF
    the fund-raising organisation is a harness racing club within the
    meaning of the Harness Racing Act 2009

RULE Community Gaming Regulation 2020 Regulation 14 - Trade promotion
gaming activities PROVIDES
the gaming activity is a permitted trade promotion gaming activity under
regulation 14 ONLY IF
    the gaming activity is a "trade promotion gaming activity" under the
    regulation 3 definition AND
    regulation 14(a) is satisfied AND
    regulation 14(b) is satisfied AND
    regulation 14(c) is satisfied

RULE Community Gaming Regulation 2020 Regulation 14(a) PROVIDES
regulation 14(a) is satisfied ONLY IF
    an entry or other fee is not charged to participate in the gaming
    activity

RULE Community Gaming Regulation 2020 Regulation 14(b) PROVIDES
regulation 14(b) is satisfied ONLY IF
    written consent has been obtained to the conduct of the gaming
    activity from a person who is authorised by the business benefiting
    from the gaming activity to provide that consent

RULE Community Gaming Regulation 2020 Regulation 14(c) PROVIDES
regulation 14(c) is satisfied ONLY IF
    the total value of all of the prizes is not more than $10,000 OR
    the person conducting the gaming activity holds an authority for
    the activity AND
    the authority is in force AND
    the gaming activity is conducted in accordance with the authority

RULE Community Gaming Regulation 2020 Regulation 14(c) Calculation PROVIDES
the total value of all of the prizes is not more than $10,000 ONLY IF
    the total value of all of the prizes LESSEQUAL THAN $10,000

MONEY the amount paid to the benefiting organisation
    RANGE $0 TO the amount of the gross proceeds of the gaming activity

MONEY the maximum amount of money payable as a separate prize
    RANGE $0 TO the total value of all of the prizes

MONEY the total value of the expenses of conducting the gaming activity
(excluding the cost of prizes)
    RANGE $0 TO the amount of the gross proceeds of the gaming activity MINUS
    the amount paid to the benefiting organisation

STRING the type of gaming activity that you are proposing
    EXPLAIN AS The Community Gaming Regulations 2020 regulate the conduct
    of gambling for social, charitable and non-profit purposes in NSW. The
    Regulations provide for 11 categories of permitted gaming activities.
    EXPLAIN Art union AS You will now be asked a series of questions
    to see whether or not your proposed activity is covered by the "Art Union
    gaming activity" provisions which are contained in regulation 4.
```

```
    EXPLAIN Housie or bingo AS You will now be asked a series of
    questions to see whether or not your proposed activity is covered by the
    "housie or bingo" provisions which are contained in regulation 5.
    EXPLAIN Draw lottery (including raffles and guessing competitions) AS
    You will now be asked a series of questions to see whether or not your
    proposed activity is covered by the "Draw Lottery" provisions which are
    contained in regulation 6.
    EXPLAIN No-draw lottery (including break-open lotteries, scratch lotteries
    and football doubles) AS You will now be asked a series of questions to
    see whether or not your proposed activity is covered by the "No-Draw
    Lottery" provisions which are contained in regulation 7.
    EXPLAIN Mini-numbers lottery AS You will now be asked a series of
    questions to see whether or not your proposed activity is covered by the
    "Mini-numbers Lottery" provisions which are contained in regulation 8.
    EXPLAIN Progressive lottery (including a hundred club, silver circles and
    tipping competitions) AS You will now be asked a series of questions to
    see whether or not your proposed activity is covered by the
    "Progressive Lottery" provisions which are contained in regulation 9.
    EXPLAIN Free lottery (including lucky door or lucky seat promotions) AS
    You will now be asked a series of questions to see whether or not
    your proposed activity is covered by the "Free Lottery"
    provisions which are contained in regulation 10.
    EXPLAIN Promotional raffle conduct by a registered club AS
    You will now be asked a series of questions to see whether or not
    your proposed activity is covered by the "Registered Club Promotional
    Raffle" provisions which are contained in regulation 11.
    EXPLAIN Other gaming activity (including a chocolate wheel or lucky
    envelopes) for charitable purposes AS You will now be asked a series
    of questions to see whether or not your proposed activity is covered by
    the "Other gaming activities" provisions which are contained in
    regulation 12."
    EXPLAIN Sweep or calcutta AS You will now be asked a series of questions
    to see whether or not your proposed activity is covered by the "Sweep or
    Calcutta" provisions which are contained in regulation 13.
    EXPLAIN Trade promotion gaming activity (including card jackpot games) AS
    You will now be asked a series of questions to see whether
    your proposed activity is covered by the "Trade Promotion" provisions
    which are contained in regulation 14.
    RANGE "Art union"
    RANGE "Housie or bingo"
    RANGE "Draw lottery (including raffles and guessing competitions)"
    RANGE "No-draw lottery (including break-open lotteries, scratch lotteries
    and football doubles)"
    RANGE "Mini-numbers lottery"
    RANGE "Progressive lottery (including a hundred club, silver circles and
    tipping competitions)"
    RANGE "Free lottery (including lucky door or lucky seat promotions)"
    RANGE "Promotional raffle conduct by a registered club"
    RANGE "Other gaming activity (including a chocolate wheel or lucky
    envelopes) for charitable purposes"
    RANGE "Sweep or calcutta"
    RANGE "Trade promotion gaming activity (including card jackpot games)"

GOAL RULE Community Gaming Regulations 2020 (NSW) PROVIDES
IF the type of gaming activity that you are proposing EQUALS "Art union"
THEN BEGIN
    DETERMINE IF the gaming activity is a permitted "art union gaming
    activity" under regulation 4
END
IF the type of gaming activity that you are proposing EQUALS "Housie or
Bingo" THEN BEGIN
    DETERMINE IF the gaming activity is permitted "housie" or "bingo"
    under regulation 5
END
IF the type of gaming activity that you are proposing EQUALS "Draw lottery
(including raffles and guessing competitions)" THEN BEGIN
```

```
    DETERMINE IF the gaming activity is a permitted "draw lottery" under
    regulation 6
END
IF the type of gaming activity that you are proposing EQUALS "No-draw
lottery (including break-open lotteries, scratch lotteries and football
doubles)" THEN BEGIN
    DETERMINE IF the gaming activity is a permitted no-draw lottery under
    regulation 7
END
IF the type of gaming activity that you are proposing EQUALS "Mini-numbers
lottery" THEN BEGIN
    DETERMINE IF the gaming activity is a permitted mini-numbers lottery
    under regulation 8
END
IF the type of gaming activity that you are proposing EQUALS "Progressive
lottery (including a hundred club, silver circles and tipping competitions)"
THEN BEGIN
    DETERMINE IF the gaming activity is a permitted progressive lottery under
    regulation 9
END
IF the type of gaming activity that you are proposing EQUALS "Free lottery
(including lucky door or lucky seat promotions)" THEN BEGIN
    DETERMINE IF the gaming activity is a permitted free lottery under
    regulation 10
END
IF the type of gaming activity that you are proposing EQUALS "Promotional
raffle conduct by a registered club" THEN BEGIN
    DETERMINE IF the gaming activity is a permitted promotional raffle
    conducted by a registered club under regulation 11
END
IF the type of gaming activity that you are proposing EQUALS "Other gaming
activity (including a chocolate wheel or lucky envelopes) for charitable
purposes" THEN BEGIN
    DETERMINE IF the gaming activity is a permitted gaming activity for
    charitable purposes under regulation 12
END
IF the type of gaming activity that you are proposing EQUALS "Sweep or
calcutta" THEN BEGIN
    DETERMINE IF the gaming activity is a permitted sweep or calcutta under
    regulation 13
END
IF the type of gaming activity that you are proposing EQUALS "Trade
promotion gaming activity (including card jackpot games)" THEN BEGIN
    DETERMINE IF the gaming activity is a permitted trade promotion gaming
    activity under regulation 14
END
```

INDEX

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Y