PIP-0105
Options in write_term

PIP working group

Subject and motivation

ISO-standard defines

```
    write_term(@stream_or_alias, @term, @write_options_list)
```

with the 4 options:

- **quoted(Bool)**
- **ignore_ops(Bool)**
- **numbervars(Bool)**
- **variable_names(List)**

Many Prolog systems implement additional options.

Draft PIP-0105 reviews these and makes recommendations.

https://prolog-lang.org/ImplementersForum/0105-write-options.html

Proposed general improvements

Unimplemented Options

ISO requires error, but ignoring or warning is more convenient. Introduce a Prolog flag **`unknown_option`** with values **`error|warning|ignore`**. The flag may be module-specific.

Convenience

Allow **`option_name`** as abbreviation for **`option_name(true)`**, e.g.

```
write_term(Term, [quoted,ignore_ops])
```

Flexibility

Allow **defaults** to be inherited from context settings. These may be global, module-specific or stream-specific.

Compatibility

If **numbervars(true)** is in effect, and the argument of `'$VAR'(N)` is an atom representing a valid variable name, output this name unquoted. Provides backward compatibility with pre-ISO implementations and removes the need for workarounds such as provided in SICStus or GProlog.

New options - term layout

max_depth(N) (SP,SWI,ECL,GP,Ciao,XSB)

If N is a positive integer, print the term only up to a maximum nesting depth of N, and represent more deeply nested subterms as .... If 0, impose no depth limit.

portrayed(Bool) (SP,ECL,IF,GP,Ciao,SWI)

If true, call the user-defined predicate portray/1,2 in the way print/1,2 does.

spacing(Atom) (PIP based on SWI)

Where to print spaces, with the alternatives

- **compact**: when needed for correct parsing (with some implementation-specific allowance for redundancy)
- **next_argument**: also after the comma separating structure or list arguments
- **generous**: also after prefix, around infix and before postfix operators

cycles(Bool) (PIP based on SP,SWI)

Use a (so far implementation-defined) finite notation to print cyclic terms.

New options - terminating printed terms

fullstop(Bool) (ECL,SWI)

Terminate the term with a fullstop (a dot followed by blank space), so it can be read back. If necessary, an extra space will be inserted *before* the fullstop, in order to separate it from the end of the term.

nl(Bool) (ECL,SWI)

Print a newline sequence (as with nl/1) after the term. If this is used together with the fullstop(true) option, this newline serves as the blank space *after* the fullstop.

```
?- write_term(+++, [fullstop,nl]), write_term(atom, [fullstop,nl]).
+++ .
atom.
```

New options - printing partial terms

priority(Prec) (SP,SWI,GP,Ciao,ECL,XSB)

Prec is an integer between 0 and 1200 (default 1200), representing context operator precedence. The written term will be enclosed in parentheses if its precedence is higher than Prec.

partial(+Bool) (SWI)

If true, insert a single space ahead of the printed term, if this is necessary to ensure token separation from previously printed text.

RATIONALE: Needed to correctly print subterms in the context of larger terms, for example when implementing pretty-printers.

New options - functor-specific syntax

portable(Bool) (PIP)

Ignore operator declarations and output the corresponding compound terms in functional notation. This is like ISO ignore_ops, except that it retains list notation ([...], also for improper lists), brace-terms ({...}) and infix commas.

```
write_term(..., [])                 [a+b*c,{d},(e,f)]
write_term(..., [portable])         [+(a,*(b,c)),{d},(e,f)]
write_term(..., [ignore_ops])       .(+(a,*(b,c)),.({}(d),.(,(e,f),[])))
```

RATIONALE: Useful when exchanging text between different Prolog contexts, such as modules with different local operator declarations, or different Prolog systems. It leads to a more compact and readable representation than `ignore_ops(true)` and is therefore often preferable.

This single option was devised as a compromise to avoid the inclusion of a multitude of more specific options that exist in systems (such as `dotlists, operators, brace_terms`, ...).

New options - subterm type specific

float_precision(+Precision) (XSB)

Number of significant digits used in printing floats. **0** means “enough digits to read back accurately”.

integer_base(+Int) (PIP based on XSB)

Specify the base (radix) for printing integers. Range 2..36, default 10.

text_max(+Length) (PIP based on XSB)

Truncate text (atoms and strings) after Length characters. Don't truncate if `0` (default). Whether and how the abbreviation is indicated is left implementation-defined. This applies to both quoted and unquoted output.

atom_quoting(+When) (PIP based on SWI)

Specify how atoms are quoted if `quoted(true)`. When is one of

- **when_needed**: when necessary for correct parsing (default)
- **when_non_ascii**: in addition, quote atoms that contain non-ASCII characters
- **always**: quote all atoms, regardless of the characters they contain (useful for non-Prolog readers)

…

Summary

- This is a somewhat open-ended subject
- PIP lists a number of further reasonable options, and recommends some for deprecation
- Before adding new options in your system, please consult the PIP beforehand!