

SML Coding Standards

Coding standards

It is important that your code be easy to read and understand. To help you achieve that goal, here are some suggestions for coding conventions.

- Limit your line lengths to about 80 characters or less.
- Indent your code following the style used in provided example code.
- Use the module system to structure your program and have a separate file per top-level module. Files containing a module `Foo` should be named `foo.sml`. Signatures can either go into the file of the module they describe or in their own file. If a signature is referenced in multiple files, then it should be defined in its own file (*e.g.*, if it has multiple implementations). If a signature `FOO` is defined in its own file, the file should be named `foo.sig` or `foo-sig.sml`.
- Define signatures for all your structures.
- Document your code: every module and every top-level definition should have a comment explaining its purpose and invariants. Non-obvious implementation techniques should also be explained in comments.
- SML has a number of different classes of identifiers and there are common conventions for how they are named.

Signature names Signature names are all caps with underscores to separate words (*e.g.*, `FINITE_SET`).

Module names Structure and functor names are mixed case with a leading upper-case letter (*e.g.*, `FiniteSet`).

Type names Type names are lower case with underscores to separate words (*e.g.*, `finite_set`).

Type variables Type variables are lower-case (*e.g.* 'a).

Exception and data constructor names Data constructors (including exceptions) follow the rules for structures.

Variable names Variables are mixed-case with an initial lower-case letter (*e.g.*, `numElements`).

Field names Field names follow the rules for variable names.

- Do *not* use **open**; instead define a shorthand local alias for a module. For example:

```
structure S = FiniteSet
... S.numElements s ...
```