Custom Annotations for the text format

Proposal status update

Andreas Rossberg

Motivation

Binary format has custom sections

Want equivalent in text format

E.g. for hinting, debugging, custom use, etc.

…use cases keep popping up

Proposal Summary

Free-form format just like custom sections

Allow (@id ...) to be placed anywhere in text

No prescribed or implied syntax or semantics,
neither regarding form nor placement

Can be ignored by tools,
but some may assign meaning to some ids

Only affects text format!

Details

Allow annotation brackets anywhere in text format

annot ::= "(@"id annotelem* ")"

Headed by id that categorises extension

…by convention, corresponds to name of a custom section

Body is arbitrary token sequence, but must be well-bracketed

A tool may define more concrete structure for its annotations

Example: Names

```
(global (@name "äquidistanz") f64)

(func $sum (@name "∑")
    (local (@name "δ") i32)
    …
)
```

Example: Generic custom section

```
(module
  …
  (@custom "my-section" (after func)
    "contents-bytes"
  )
)
```

NB: placement “after func” only legal if a (non-empty) function section exists

Spec Appendix

Describes @name and @custom annotations

Analogous to “name” sections in binary format

Proposal Status

✅ Full spec (prose and formal)

✅ Interpreter implementation

✅ Test suite
(though no tests for Appendix)

✅ Implementation in wabt
(text-format only, does not affect engines)

Stage 3 (2020/03)

Discussion: Test suite

Custom sections and annotations are optional

…no implementation needs to recognise them

Thus, cannot put them into the regular test suite

Need infrastructure for optional tests

…fairly general problem, maybe solve separately?

Discussion

Move to stage 4?