Test sets

The test set expression DSL is used to get fine control over selecting tests for operations without requiring a lot of complicated options. Test sets binary or unary nested expressions of sets using set operators.

If no test set is specified by the user, but individual test identifiers are passed they are used as overrides for the `default` test set. More concretely given the invocation `typst-test list test1 test2 ...` the following test set is constructed: `default & (id(=test1) | id(=test2) | ...)`.

Special test sets

The following built in constants are given and can be used as regular test sets in compound expressions:

Name	Explanation
`none`	Includes no tests.
`all`	Includes all tests.
`ignored`	Includes all tests with an ignored annotation
`compile-only`	Includes all tests without references.
`ephemeral`	Includes all tests with ephemeral references.
`persistent`	Includes all tests with persistent references.
`default`	A shorthand for `!ignored`, this is used as a default if no test set is passed.

Matcher test sets

The following matchers are given and are used within `id(...)` to match on the identifiers of tests:

Name	Example	Explanation
`plain`	`id(plain)`	Defaults to contains matcher[1]
`=exact`	`=mod/name`	Matches exactly one test who's identifier is exactly the contained term.
`~contains`	`~plot`	Matches any test which contains the given term in its identifier.
`/regex/`	`/mod-[234]\/.*/`	Matches an tests who's identifier matches the given regex, literal `/` must be escaped using `\`.

Operators

The following binary operators exist and can operatate on any other test set expression:

Type	Prec.	Name	Symbols	Literal	Explanation
infix	1	union	`∪`, `|` or `+`	`or`	Includes all tests which are in either the left OR right test set expression.
infix	1	difference	`\` or `-`	—[2]	Includes all tests which are in the left but NOT in the right test set expression.
infix	2	intersection	`∩` or `&`	`and`	Includes all tests which are in both the left AND right test set expression.

[1]This may change in the future to allow different defaults for different matchers.

Type	Prec.	Name	Symbols	Literal	Explanation
infix	3	symmetric difference	`Δ` or `^`	`xor`	Includes all tests which are in either the left OR right test set expression, but NOT in both.
prefix	4	complement	`¬` or `!`	`not`	Includes all tests which are NOT in the test set expression.

Be aware of precedence when combining different operators, higher precedence means operators bind more strongly, e.g. `not a and b` is `(not a) and b`, not `not (a and b)` because `complement` has a higher precedence than `intersection`. Binary operators are left associative, e.g. `a - b - c` is `(a - b) - c`, not `a - (b - c)`.

Examples

Suppose you had a project with the following tests:

```
...
mod/sub/foo ephemeral
mod/sub/bar ephemeral
mod/sub/baz persistent
mod/foo     persistent
mod/bar     ephemeral
...
```

And you wanted to make your ephemeral tests in `mod/sub` persistent, you could construct a expression with the following steps:

1. Let's filter out all ignored test as typst-test does by default, this could be done with `!ignored`, but there is the also handy default test set for this.
   - `default`
2. We only want ephemeral tests so we add annother intersection with the ephemeral test set.
   - `default & ephemeral`
3. Now we finally restrict it to be only test which are in `mod` by adding an identifier matcher test set.
   - `default & ephemeral & id(/^mod\/sub/)`

You can iteratively test your results with `typst-test list -e '...'` until you're satisfied and then do `typst-test update --all -e '...'` with the given expression, the `--all` option is required if you're operating destructively (editing, updating, removing) on more than one test.

Notes on scripting

When building expressions programmatically it may serve simplicity to assign a default value to one operand of an n-ary expression. The `all` and `none` test sets can be used as identity sets for certain set operations.

Make sure to use your shell's non-interpreting quotes (often single quotes `'...'`) around the expression to avoid accidentally running.

[2] There is currently no literal operator for set difference.