This directory contains unit tests for the MIR-based dataflow analysis.

These unit tests check the dataflow analysis by embedding calls to a special `rustc_peek` intrinsic within the code, in tandem with an attribute `#[rustc_mir(rustc_peek_maybe_init)]` (*). With that attribute in place, `rustc_peek` calls are a signal to the compiler to lookup the computed dataflow state for the Lvalue corresponding to the argument expression being fed to `rustc_peek`. If the dataflow state for that Lvalue is a 1-bit at that point in the control flow, then no error is emitted by the compiler at that point; if it is a 0-bit, then that invocation of `rustc_peek` will emit an error with the message "rustc_peek: bit not set".

(*): Or `#[rustc_mir(rustc_peek_maybe_uninit)]`, and perhaps other variants in the future.

The end effect is that one can write unit tests for MIR dataflow that perform simple-queries of the computed dataflow state, and the tests should be able to be robust in the face of changes to how MIR is represented or constructed.

Sometimes understanding the dataflow results is difficult without looking at the actual MIR control-flow graph being processed with the corresponding GEN and KILL sets.

For a graphviz-rendering with dataflow annotations, add the attribute `#[rustc_mir(borrowck_graphviz_postflow="/path/to/suffix.dot")]` to the function in question. (You can change the content of `"suffix.dot"` to control the filenames used for the output). This will generate a separate file for each dataflow analysis, adding a prefix (corresponding to the name of the analysis) to the filename in each generated output path.

- For example, the above attribute will currently cause two files to be generated: `/path/to/maybe_init_suffix.dot` and `/path/to/maybe_uninit_suffix.dot`.
- The generated `.dot` file shows both the computed dataflow results on *entry* to each block, as well as the gen- and kill-sets that were so-called "transfer functions" summarizing the effect of each basic block.
- (In addition to the `borrowck_graphviz_postflow` attribute-key noted above, there is also `borrowck_graphviz_preflow`; it has the same interface and generates the same set of files, but it renders the dataflow state after building the gen- and kill-sets but *before* running the dataflow analysis itself, so each entry-set is just the initial default state for that dataflow analysis. This is less useful for understanding the error message output in these tests.)