From DFA to code

- Assuming we have produced a minimal DFA for the token types in our language (e.g. via Thompson's, subset construction, and Hopcroft's) we now need to produce actual scanner code
- There are three common approaches: table-driven scanners, direct-coded scanners, and hand-coded scanners
- Each involves producing code to simulate the DFA: reading input characters and carrying out transitions, then identifying the correct token type
- Differences are in how the simulation is carried out

Process (for all three approaches)

- In each case, reads and transitions are simulated until a point is reached at which no valid transition is available
- If current state is an accept the corresponding token type is output and the simulation begins anew for next token
- If current state is not an accept the simulation rolls back to most recent accept state (if any) and *that* state's token type is output, then the simulation begins anew
- If no accept state has been encountered then the input is invalid

Table-driven scanners

- The scanner code itself is relatively simple in this method, as the language-specific information is held in a set of tables: a Classifier table, a Transition table, and a Token Type table
- The scanner generator produces these tables based on the minimized DFA produced earlier
- A fixed algorithm is used to read the input and use the tables to conduct the scanning (the algorithm is independent of the language being scanned)

Direct coded scanners

- Rather than using lookup tables, language-specific code is used during the scanning process, with unique code segments corresponding to each DFA state
- The reduces the time/memory needed for table lookups, but increases the code size/complexity, and makes the scanner code specific to a single source language
- Depending on the size and classification techniques for the sets of input characters, this can be significantly better or worse than the table driven approach

Hand coded scanners

- Still widely used, relying on the skills of the developers to recognize and implement improvements on either of the other techniques (or to use hybrids of the other techniques)
- A number of common areas for improvement will be examined