CSE443
Compilers

Dr. Carl Alphonce
alphonce@buffalo.edu
343 Davis Hall
Announcements

- Will be posted no later than Monday
  - HW-01
  - PR-01
- Team formation
  - Take a few minutes to do it now
  - Make private Piazza post with UBITS
Phases of a compiler

Figure 1.6, page 5 of text
We can describe a regular language using a regular expression
Why do we care?

- We will be using a tool called FLEX to construct a lexical analyzer (a lexer) for the programming language we're constructing a compiler for.
- If we give FLEX a regular expression describing the lexical structure of our language, FLEX will produce a C program which acts as our lexer.
- The next step for us to understand (at a high level) how FLEX converts a regex to a C program.
A regular expression can be implemented using a finite state machine.

Finite state machines can be deterministic or non-deterministic:

- DFA: deterministic finite automaton
- NFA: non-deterministic finite automaton
Process of building lexical analyzer

1) spell out the language
Process of building lexical analyzer

2) formulate a regular expression
Process of building lexical analyzer

3) build an NFA

(language -> regex -> NFA)
Process of building lexical analyzer

4) transform NFA to DFA
Process of building lexical analyzer

5) transform DFA to a minimal DFA
Process of building lexical analyzer

5) The minimal DFA is our lexical analyzer

language → regex → NFA → DFA → character stream

lexical analyzer → token stream
Step 1: Construct NFA from regex
Nondeterministic Finite Automata (NFA)

- A finite set of states $S$
- An alphabet $\Sigma$, $\varepsilon \notin \Sigma$
- $\delta \subseteq S \times (\Sigma \cup \{\varepsilon\}) \times \mathcal{P}(S)$ (transition function)
- $s_0 \in S$ (a single start state)
- $F \subseteq S$ (a set of final or accepting states)
Deterministic Finite Automata (DFA)

- A finite set of states \( S \)
- An alphabet \( \Sigma \), \( \varepsilon \notin \Sigma \)
- \( \delta \subseteq S \times \Sigma \times S \) (transition function)
- \( s_0 \in S \) (a single start state)
- \( F \subseteq S \) (a set of final or accepting states)
A state is a circle with its state number written inside.
Initial state has an arrow from nowhere pointing in. State 0 is often the initial state.
A final state is drawn with a double circle.
Arrows are labeled with $\varepsilon$ ...

$\varepsilon$

... or $a \in \Sigma$.

$a$

for each $a \in \Sigma$
Regex $\rightarrow$ NFA

For each $a \in \Sigma$
Regex → NFA
Simple example

static
Simple example

static
Simple example

static

struct

![Diagram showing a simple example with nodes labeled from 0 to 13 and arrows indicating transitions between them.](image-url)
Step 2: Construct DFA equivalent to NFA
first we construct an NFA from this regular expression
$(a|b)^*abb$
(a|b)*abb

a

b

Diagram shows a nondeterministic automaton with transitions from 'a' to 'a' and from 'b' to 'b'.
$(a|b)^*abb$
(a|b)*abb
\[(a|b)^*abb\]
$(a|b)^*abb$
$(a|b)^*abb$
Operations

- $\varepsilon$-closure($t$) is the set of states reachable from state $t$ using only $\varepsilon$-transitions.

- $\varepsilon$-closure($T$) is the set of states reachable from any state $t \in T$ using only $\varepsilon$-transitions.

- move($T,a$) is the set of states reachable from any state $t \in T$ following a transition on symbol $a \in \Sigma$. 
NFA -> DFA algorithm
(set of states construction - page 153 of text)

Input: An NFA \( N = (S, \Sigma, \delta, s_0, F) \)

Output: A DFA \( D = (S', \Sigma, \delta', s_0', F') \) such that \( L(D) = L(N) \)

Algorithm:
Compute \( s_0' = \varepsilon\text{-closure}(s_0) \), an unmarked set of states
Set \( S' = \{ s_0' \} \)
while there is an unmarked \( T \in S' \)
mark \( T \)
for each symbol \( a \in \Sigma \)
let \( U = \varepsilon\text{-closure}(\text{move}(T, a)) \)
if \( U \notin S' \), add unmarked \( U \) to \( S' \)
add transition: \( \delta'(T, a) = U \)

\( F' \) is the subset of \( S' \) all of whose members contain a state in \( F \).
NFA -> DFA algorithm
(set of states construction - page 153 of text)

So' = \{ A = \{0,1,2,4,7\} \}

Pick an unmarked set from So', A, mark it, and \( \forall x \in \Sigma \) let \( U = \varepsilon\)-closure(move(A,x)), if \( U \notin S' \), add unmarked U to S' and add transition: \( \delta'(A,x) = U \)

S1' = \{ A' , B = \{1,2,3,4,6,7,8\} , C = \{1,2,4,5,6,7\}\}
\( \delta'(A,a) = B \)
\( \delta'(A,b) = C \)

Pick an unmarked set from S1', B, mark it, and \( \forall x \in \Sigma \) let \( U = \varepsilon\)-closure(move(B,x)), if \( U \notin S' \), add unmarked U to S' and add transition: \( \delta'(B,x) = U \)

S2' = \{ A' , B' , C , D = \{1,2,4,5,6,7,9\}\}
\( \delta'(B,a) = B \)
\( \delta'(B,b) = D \)

Pick an unmarked set from S2', C, mark it, and \( \forall x \in \Sigma \) let \( U = \varepsilon\)-closure(move(C,x)), if \( U \notin S' \), add unmarked U to S' and add transition: \( \delta'(C,x) = U \)

S3' = \{ A' , B' , C' , D \}
\( \delta'(C,a) = B \)
\( \delta'(C,b) = C \)
NFA -> DFA algorithm
(set of states construction - page 153 of text)

Pick an unmarked set from $S_3'$, $D$, mark it, and $\forall x \in \Sigma$ let $U = \varepsilon$-closure(move($D$, $x$)), if $U \not\in S'$, add unmarked $U$ to $S'$ and add transition: $\delta'(D, x) = U$

$S_4' = \{ A\checkmark, B\checkmark, C\checkmark, D\checkmark, E = \{1,2,4,5,6,7,10\} \}$

$\delta'(D,a) = B$

$\delta'(D,b) = E$

Pick an unmarked set from $S_4'$, $E$, mark it, and $\forall a \in \Sigma$ let $U = \varepsilon$-closure(move($E$, $a$)), if $U \not\in S'$, add unmarked $U$ to $S'$ and add transition: $\delta'(E, a) = U$

$S_5' = \{ A\checkmark, B\checkmark, C\checkmark, D\checkmark, E\checkmark \}$

$\delta'(E,a) = B$

$\delta'(E,b) = C$

Since there are no unmarked sets in $S_5'$ the algorithm has reached a fixed point. STOP.

$F'$ is the subset of $S'$ all of whose members contain a state in $F$: $\{E\}$
The original NFA
The resulting DFA

DFA = ( \{A, B, C, D, E\}, \{a, b\}, A, \delta', \{E\})

\begin{align*}
\delta'(A,a) &= B \\
\delta'(A,b) &= C \\
\delta'(B,a) &= B \\
\delta'(B,b) &= D \\
\delta'(C,a) &= B \\
\delta'(C,b) &= C \\
\delta'(D,a) &= B \\
\delta'(D,b) &= E \\
\delta'(E,a) &= B \\
\delta'(E,b) &= C
\end{align*}
Process of building lexical analyzer

5) The minimal DFA is our lexical analyzer
Step 3: DFA minimization
NFA for \((a|b)^{*}abb\)
DFA for \((a|b)^*abb\)
Minimization Algorithm
DFA -> minimal DFA algorithm

**INPUT:** An DFA \( D = (S, \Sigma, \delta, s_0, F) \)

**OUTPUT:** A DFA \( D' = (S', \Sigma, \delta', s'_0, F') \) such that
- \( S' \) is as small as possible, and
- \( \mathcal{L}(D) = \mathcal{L}(D') \)

**ALGORITHM:**
1. Let \( \pi = \{ F, S-F \} \)
2. Let \( \pi' = \pi \). For every group \( G \) of \( \pi \):
   - partition \( G \) into subgroups such that two states \( s \) and \( t \) are in the same subgroup iff for all input symbols \( a \), states \( s \) and \( t \) have transitions on \( a \) to states in the same group of \( \pi \)
   - Replace \( G \) in \( \pi' \) by the set of all subgroups formed
3. if \( \pi' = \pi \) let \( \pi'' = \pi \), otherwise set \( \pi = \pi' \) and repeat 2.
4. Choose one state in each group of \( \pi'' \) as a representative for that group.
   a) The start state of \( D' \) is the representative of the group containing the start state of \( D \)
   b) The accepting states of \( D' \) are the representatives of those groups that contain an accepting state of \( D \)
   c) Adjust transitions from representatives to representatives.
ORIGINAL DFA

\[ D = (S, \Sigma, s_0, \delta, F) \]

\( S = \{A, B, C, D, E\} \)
\( \Sigma = \{a, b\} \)
\( s_0 = A \)
\( \delta = \{(A,a)\rightarrow B, (A,b)\rightarrow C, \)
         \( (B,a)\rightarrow B, (B,b)\rightarrow D, \)
         \( (C,a)\rightarrow B, (C,b)\rightarrow C, \)
         \( (D,a)\rightarrow B, (D,b)\rightarrow E, \)
         \( (E,a)\rightarrow B, (E,b)\rightarrow C\} \)
\( F = \{E\} \)
Finding the minimal set of distinct sets of states

\[ \pi_0 = \{ F, S-F \} = \{ \{E\}, \{A,B,C,D\} \} \]

Pick a non-singleton set \( X = \{A,B,C,D\} \) from \( \pi_0 \) and check behavior of states on all transitions on symbols in \( \Sigma \) (are they to states in \( X \) or to other groups in the partition?)

\((A,a)\rightarrow B, (B,a)\rightarrow B, (C,a)\rightarrow B, (D,a)\rightarrow B\)
\((A,b)\rightarrow C, (B,b)\rightarrow D, (C,b)\rightarrow C, (D,b)\rightarrow E\)

\( D \) behaves differently, so put it in its own partition.
Finding the minimal set of distinct sets of states

\[ \pi_1 = \{ \{E\}, \{A, B, C\}, \{D\} \} \]

Pick a non-singleton set \( X = \{A,B,C\} \) from \( \pi_1 \) and check behavior of states on all transitions on symbols in \( \Sigma \) (are they to states in \( X \) or to other groups in the partition?)

\[ (A,a) \rightarrow B, \ (B,a) \rightarrow B, \ (C,a) \rightarrow B \]
\[ (A,b) \rightarrow C, \ (B,b) \rightarrow D, \ (C,b) \rightarrow C \]

B behaves differently, so put it in its own partition.
Finding the minimal set of distinct sets of states

\[ \pi_2 = \{ \{E\}, \{A, C\}, \{B\}, \{D\} \} \]

Pick a non-singleton set \( X = \{A, C\} \) from \( \pi_2 \) and check behavior of states on all transitions on symbols in \( \Sigma \) (are they to states in \( X \) or to other groups in the partition?)

\( (A, a) \rightarrow B, (C, a) \rightarrow B \)
\( (A, b) \rightarrow C, (C, b) \rightarrow C \)

A and C both transition outside the group on symbol a, to the same group (the one containing B). Therefore A and C are indistinguishable in their behaviors, so do not split this group.
Finding the minimal set of distinct sets of states

\[ \pi_3 = \{ \{E\}, \{A, C\}, \{B\}, \{D\} \} = \pi_2 \]

We have reached a fixed point! STOP
Pick a representative from each group

\[ \pi_{\text{final}} = \{ \{E\}, \{A, C\}, \{B\}, \{D\} \} \]
MINIMAL DFA

\[ D' = (S', \Sigma, s'_0, \delta', F') \]

\( S' = \{B, C, D, E\} \) -> the representatives
\( \Sigma = \{a, b\} \) -> no change
\( s'_0 = C \) -> the representative of the group that contained D's starting state, A
\( \delta = \) (on next slide)
\( F = \{E\} \) -> the representatives of all the groups that contained any of D's final states (which, in this case, was just \{E\})
The new transition function $\delta'$

- For each state $s \in S'$, consider its transitions in $\mathcal{D}$, on each $a \in \Sigma$.

- If $\delta(s,a) = t$, then $\delta'(s,a) = r$, where $r$ is the representative of the group containing $t$. 
\[ \delta = \{ (B,a) \rightarrow B, (B,b) \rightarrow D, (C,a) \rightarrow B, (C,b) \rightarrow C, (D,a) \rightarrow B, (D,b) \rightarrow E, (E,a) \rightarrow B, (E,b) \rightarrow C \} \]
Minimal DFA for $(a|b)^*abb$
DFA for (ab|b)*abb

Non-minimized