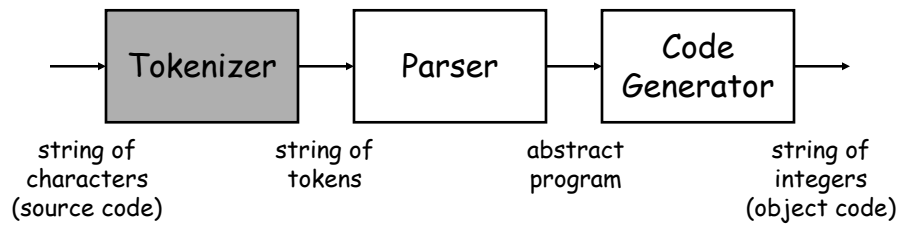
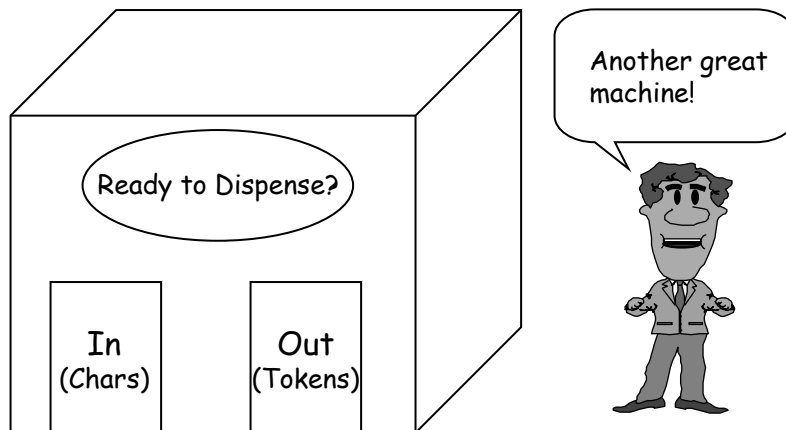


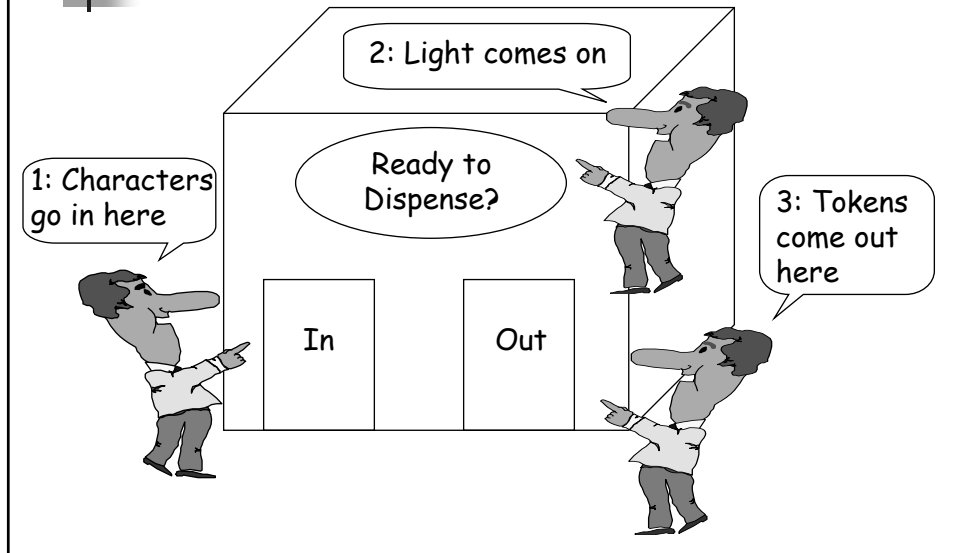
Translator Architecture



A Tokenizing Machine



Tokenizing Machine Continued...



Tokenizing Machine Continued...

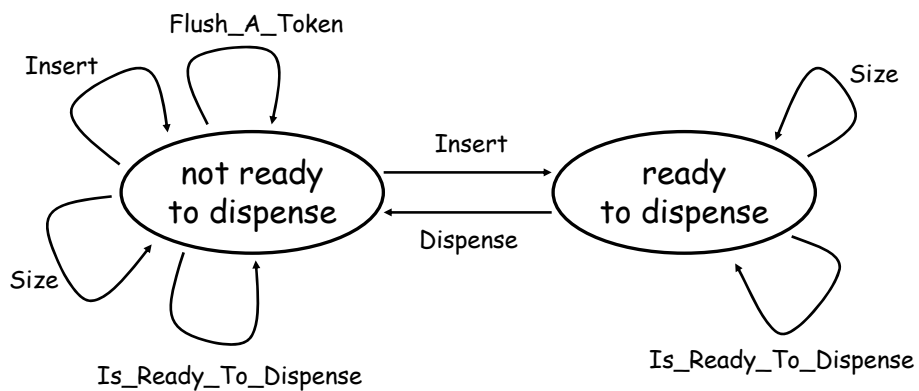
- **Type**
BL_Tokenizing_Machine_Kernel
is modeled by (
 buffer: string of character
 ready_to_dispense: boolean
)
 constraint ...
- **Initial Value**
 (empty_string, false)

Tokenizing Machine Continued...

■ Operations

- m.Insert (ch)
- m.Dispense (token_text, token_kind)
- m.Is_Ready_To_Dispense ()
- m.Flush_A-Token (token_text, token_kind)
- m.Size ()

A State-Transition View



The Specification of Insert

```
procedure Insert (  
  preserves Character ch  
) is_abstract;  
/*!  
  requires  
    self.ready_to_dispense = false  
  ensures  
    self.buffer = #self.buffer * <ch> and  
    self.ready_to_dispense =  
      IS_COMPLETE_TOKEN_TEXT (#self.buffer, ch)  
!*/
```

An Important Math Operation

```
math definition IS_COMPLETE_TOKEN_TEXT (  
  s: string of character  
  c: character  
) : boolean is  
  (s is in OK_STRINGS and  
   s * <c> is not in OK_STRINGS) or  
  (<c> is in PREFIX (OK_STRINGS) and  
   s * <c> is not in PREFIX (OK_STRINGS))
```

Other Math Definitions

- **OK_STRINGS =**
 - {s: string of character (IS_KEYWORD (s))} union
 - {s: string of character (IS_IDENTIFIER (s))} union
 - {s: string of character (IS_CONDITION_NAME (s))} union
 - {s: string of character (IS_WHITE_SPACE (s))} union
 - {s: string of character (IS_COMMENT (s))}
- **PREFIX (s_set) =**
 - {x: string of character
 - (there exists y: string of character
 - (x * y is in s_set))}

PREFIX Examples

- s_set = {"abc"}
- PREFIX (s_set) = ?

- s_set = {"abc", "de"}
- PREFIX (s_set) = ?

Tokenizing Machine: Implementation

- Obvious Representation
 - Text buffer_rep
 - Boolean token_ready
- Insert (ch)?
 - check if IS_COMPLETE_TOKEN_TEXT (self[buffer_rep], ch), and set self[token_ready] accordingly
 - append ch at end of self[buffer_rep]

Tokenizing Machine: Implementation Continued...

- Dispense (token_text, token_kind)?
 - set token_text to all but the last character of self[buffer_rep]
 - set token_kind to the value of WHICH_KIND (token_text)

Tokenizing Machine: Implementation Continued...

- How do we “check if `IS_COMPLETE_TOKEN_TEXT (self[buffer_rep], ch)`”?
- How do we determine “`WHICH_KIND (token_text)`”?
- How do we do these things quickly?

Making Decisions Quickly

- Keep track of the “state” of the buffer by adding one field to the representation:
 - Text `buffer_rep`
 - Boolean `token_ready`
 - Integer `buffer_state`



Possible Buffer States

- How many *interestingly* different buffer states do you think there may be?
- Let's start enumerating them...



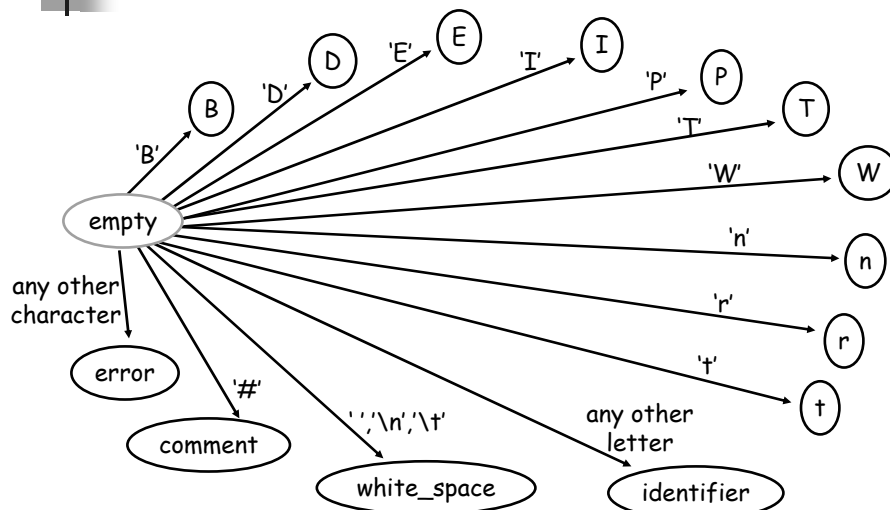
Buffer States Continued...

- Initial state (empty buffer)
- How many states after inserting the first character?

Buffer States Continued...

- How many states after inserting the second character?

A State Transition Diagram: Transitions Out of 'empty' Only



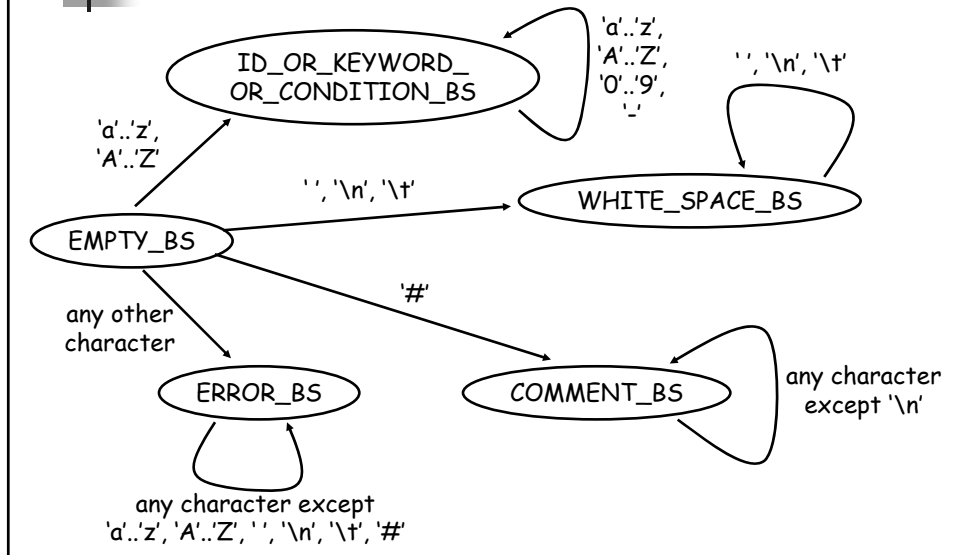
Structure of Body of Insert

```
case_select (self[buffer_state])
{
  case empty:
    // case for buffer = empty_string
  case B:
    // case for buffer = "B"
  case D:
    // case for buffer = "D"
  case E:
    // case for buffer = "E"
  ...
  case error:
    // case for buffer holding an error token
}
```

A Simplified View

- Buffer States
 - EMPTY_BS
 - ID_OR_KEYWORD_OR_CONDITION_BS
 - WHITE_SPACE_BS
 - COMMENT_BS
 - ERROR_BS

The State Transition Diagram



Useful Protected Functions

- `Is_White_Space_Character (ch)`
- `Is_Digit_Character (ch)`
- `Is_Alphabetic_Character (ch)`
- `Is_Identifier_Character (ch)`
- `Can_Start_Token (ch)`
- `Id_Or_Keyword_Or_Condition (t)`
- `Buffer_Type (ch)`
- `Token_Kind (bs, str)`