CSE216 Programming Abstractions

Parsing JSON

YoungMin Kwon

Recursive Descent Parsing

- How to write a recursive descent parser
  - Write a subroutine for each non-terminal
  - Input-token is the token available from the scanner
  - Match to consume and update the input-token

Parsing JSON file

- JSON format
  - A human-readable standard format for data interchange.
- JSON data types
  - Number: a positive integer
  - Boolean: true or false
  - null: an empty value, using the word null
  - String: a sequence of zero or more characters.
    - Strings are delimited with double-quotation marks.

```
{
    "name": "John Smith",
    "isAlive": true,
    "age": 27,
    "address": {
        "city": "New York",
        "state": "NY",
    },
    "phoneNumbers": [
        {
            "type": "home",
            "number": "212 555-1234"
        },
        {
            "type": "office",
            "number": "646 555-4567"
        }
    ],
    "children": [],
    "spouse": null
}
```

Parsing JSON file

- Array: an ordered list of zero or more elements of any type.
  - Arrays use square bracket notation ('[', ']') with **comma**-separated elements.
- Object: a collection of name–value pairs, where the names are strings.
  - objects are delimited with curly brackets ('{', '}') and use **commas** to separate each pair.
  - within each pair the colon '**:**' character separates the key or name from its value.

```
{
    "name": "John Smith",
    "isAlive": true,
    "age": 27,
    "address": {
        "city": "New York",
        "state": "NY",
    },
    "phoneNumbers": [
        {
            "type": "home",
            "number": "212 555-1234"
        },
        {
            "type": "office",
            "number": "646 555-4567"
        }
    ],
    "children": [],
    "spouse": null
}
```

Scanner for JSON

```
enum tokens {
    TOK_ALL = 256,  //match for the whole expr
    TOK_WS,         //white space
    TOK_STR,        //string
    TOK_NUM,        //num
    TOK_TRUE,       //true
    TOK_FALSE,      //false
    TOK_NULL,       //null
    TOK_EOF,        //end of file
    TOK_COUNT = (TOK_EOF - TOK_ALL + 1) //token count
};

#define TOK_REGEXP  "^([ \t\n\r]+)|"                /*whitespace*/\
                    /*TODO: add a regular expression for string*/\
                    "^(\"[^\"]*\")|"                /*string*/\
                    /*TODO: add a regular expression for number*/\
                    "^([0-9]+)|"                    /*number*/\
                    "^(true)|"                      /*true*/\
                    "^(false)|"                     /*false*/\
                    "^(null)"                      /*null*/
```

CFG for JSON

```
json
    -> value
value
    -> STRING
    |  NUMBER
    |  TRUE
    |  FALSE
    |  NULL
    |  object
    |  array
opt_value_list  //optional value list
    ->          //possibly empty
    |  value_list

object          //cannot be empty

array           //can be empty

pair

pair_list

value_list
```

```
{
    "name": "John Smith",
    "isAlive": true,
    "age": 27,
    "address": {
        "city": "New York",
        "state": "NY",
    },
    "phoneNumbers": [
        {
            "type": "home",
            "number": "212 555-1234"
        },
        {
            "type": "office",
            "number": "646 555-4567"
        }
    ],
    "children": [],
    "spouse": null
}
```

CFG for JSON

```
json
    -> value

value
    -> STRING
    |  NUMBER
    |  TRUE
    |  FALSE
    |  NULL
    |  object
    |  array

opt_value_list  //optional value list
    ->          //possibly empty
    |  value_list

object          //cannot be empty
    -> { pair_list }

array           //can be empty
    -> [ opt_value_list ]
```

```
pair
    -> STRING : value

pair_list
    -> pair
    |  pair_list , pair

value_list
    -> value
    |  value_list , value
```

Parser for JSON

```
static void parse_json() {
    parse_value();
}
static void parse_opt_value_list() {
    if(sc.token == ']') //empty list
        return;
    parse_value_list();
}

//TODO: implement parse_object
static void parse_object() {
}

//TODO: implement parse_array
static void parse_array() {
}

//TODO: implement parse_pair
static void parse_pair() {
}
```

```
//TODO: implement parse_pair_list
static void parse_pair_list() {
}

//TODO: implement parse_value
static void parse_value() {

    //when nothing matches
        ON_FALSE_EXIT(0, strmsg(
            "Syntax error: unexpected token %s in line %d",
            sc.text, sc.line));
}

//TODO: implement parse_value_list
static void parse_value_list() {
}

void parse(char *fname) {
    open_scan(&sc, fname);
    read_token(&sc);        //load the first token for the parser
    parse_json();
    close_scan(&sc);
    printf("Success.");
}
```

Thank you for your attention during the semester!

Any questions or comments?

Please provide your course evaluation and course outcome survey for ABET at
https://stonybrook.campuslabs.com/eval-home/
https://forms.gle/YAZag8ccNi1wgFyC6

SUNY Korea