AWSIM-Script Language Grammar

Duong Dinh Tran, Takashi Tomita, and Toshiaki Aoki
Japan Advanced Institute of Science and Technology, Ishikawa 923-1292, Japan

Listing 1. Part of AWSIM-Script grammar defined in ANTLR v4.

```
grammar AWSIMScriptGrammar;
positionExp
  : stringExp ('at' (numberExp | variableExp))?
  | variableExp ('back' | 'forward' | 'left' | 'right')
     (numberExp | variableExp)
  | positionExp ('back' | 'forward' | 'left' | 'right')
     (numberExp | variableExp);
roadExp
  : stringExp ('max-velocity' '(' (numberExp | variableExp) ')')?
  | ('change-lane' | 'cut-in' | 'cut-out') '(' argumentList? ')';
vector2Exp : ((numberExp | variableExp) '#'
              (numberExp | variableExp));
configExp
  : 'aggressive-driving'
  | ('acceleration' | 'deceleration' | 'speed') '(' (numberExp |
     variableExp) ')'
  | ('delay-spawn' | 'delay-move') '(' (numberExp | variableExp)
     ')'
  | ('delay-spawn-until-ego-move' | 'delay-move-until-ego-move')
     '(' (numberExp | variableExp) ')'
  | ('delay-spawn-until-ego-engaged' |
     'delay-move-until-ego-engaged') '(' (numberExp |
     variableExp) ')';
egoSettingExp : 'max-velocity' '(' (numberExp | variableExp) ')';
simulationSettingExp : 'saving-timeout' '(' numberExp ')';
functionExp : idExp '(' argumentList? ')' ;
arrayExp : '[' argumentList? ']';
argumentList : expression ( ',' expression )∗ ;
assignmentStm : variableExp '=' expression;
variableExp: idExp;
expression
  : stringExp | numberExp | vector2Exp | positionExp | roadExp |
     arrayExp | variableExp | configExp | egoSettingExp |
     simulationSettingExp | functionExp;
statement : (assignmentStm | functionExp) ';' ;
scenario : (statement)+ EOF;
```

We use the ANother Tool for Language Recognition (ANTLR) v4 framework [1] to define the grammar of AWSIM-Script and to implement the parser component. This is a powerful parser generator framework, widely used to build interpreters, compilers, and tools that require parsing formal languages such as programming languages or domain-specific languages. ANTLR supports the grammar that is close to Backus-Naur Form (BNF) but includes additional features that are not present in standard BNF.

Listing 1 shows part of the language grammar defined in ANTLR. The language supports constructs for defining positions, road elements (e.g., lane changes), and simulation configurations. For example, a `positionExp` denotes a location on a traffic lane, and a `roadExp` allows users to define routes and lane changes together with desired speeds.

Note that `stringExp`, `numberExp`, and `idExp` in the grammar denote strings (which are enclosed by a pair of double quotes), float numbers, and variable/function names (which are made from alphabets, numbers 0-9, and underscore character), respectively, although we omit their definitions in the figure. We use the ANTLR parser generator tool to generate the base parser from the defined grammar. Building on this base parser, we implement the module that parses the input script and invokes the appropriate simulation APIs of the extended AWSIM-Labs to perform the simulation. Note also that there exists the case in which an input script is accepted by the grammar but the parsing process produces an error, for example, using an undefined variable.

References

[1] T. Parr, *The Definitive ANTLR 4 Reference*. 2nd ed., 2013.