KI-Programmierung, Simple Scheduling

Roman Gerloff, Ahmed Attia

WS 2022

1 Description of Grammar

For a scheduling problem, the grammar is either a single task description or multiple task descriptions. A task description contains a name, duration, and dependencies. It is composed of an alphanumeric character followed by any number of alphanumeric characters. The duration of a task is a non-zero digit followed by any number of digits. The dependencies can be either “none” or an expression. An expression is either a single term or multiple terms connected by the word “or.” A term is either a factor or multiple factors connected by the word “and.” A factor can be a task name or an expression within parentheses. An alphanumeric character is either an alpha character or a digit. An alpha character is any letter of the alphabet (uppercase or lowercase). A non-zero digit is any number between 1 and 9, and a digit is any number between 0 and 9. The grammar defined that “and” has higher precedence than “or.” If an expression contains both “and” and “or” operators, the “and” operator is evaluated first, then the ”or” operator.

Let’s take ”A and B or C and D as an example.” The ”and” operator will be applied first, resulting in the expression ”(A and B) or (C and D). ” The ”or” operator will then be applied to the resulting terms, resulting in the final expression ”((A and B) or (C and D)).”

The grammar defines the precedence as follows:

1. Parentheses
2. Factors
3. Terms
4. Expression

The file ”Grammar.txt” contains the complete grammar in BNF syntax.

2 Design of task descriptions

Objects of the class "Task" are used to store task descriptions. It includes three parameters. The first parameter is the name, which is a String. The duration is an integer. A tree represents the dependencies.

3 Collaborate work

We worked together to define the grammar and design the task class. It was important to adjust the final output from the parser to fit into the z3 solver after the parser and z3-solver were done so that they could work without problems..

- The convert_tree_to_z3_expression() method in the class "scheduling_z3.py"
  - It is the method responsible for translating the dependencies into valid conditions for the solver
  - If no dependency is needed for this task, it will simply be skipped. Otherwise it will work recursively by parsing the 'And' and 'Or' operators correctly until there is only one task will be added to the solver after that.
- Implementation of "main.py"
  - The schedule() method, which combines the parser and the z3-solver.
  - The error handling of the parser and Z3 was implemented.
  - If the input is invalid, the program will terminate and print an Exception message.
  - The examples provided in the project description are included in doctests.
- Multiple test files were added for testing, such as:
  - simplest
  - vicious
  - standard
  - wrong_name including "Task 1" instead of Task1
  - invalid_symbols including "Task-1" which doesn't follow the grammar.
  - invalid_dependencies including (Task1,Task2,Task3)