A Programming Language for Designing Turing Machines

Frederic Hamelink, Sander Marinus,

Jens Leysen en Dries De Maeseneer

Features

Features

1. Een CFL voor single- en multi-tape Turing machines.
2. Een parsing algoritme uit de volgende lijst: LR, SLR, LALR, Early.
3. Aanmaken van een single-tape Turing machine.
4. Simulatie van een single-tape Turing machine.
5. Dynamische visualisatie van de werking van een single-tape Turing machine.
6. Statische visualisatie van een single-tape Turing machine.
7. Het inlezen van een multi-tape Turing machine.
8. Het simuleren van een multi-tape tape Turing machine.
9. Code genereren aan de hand van AST.
10. Error detection AST.

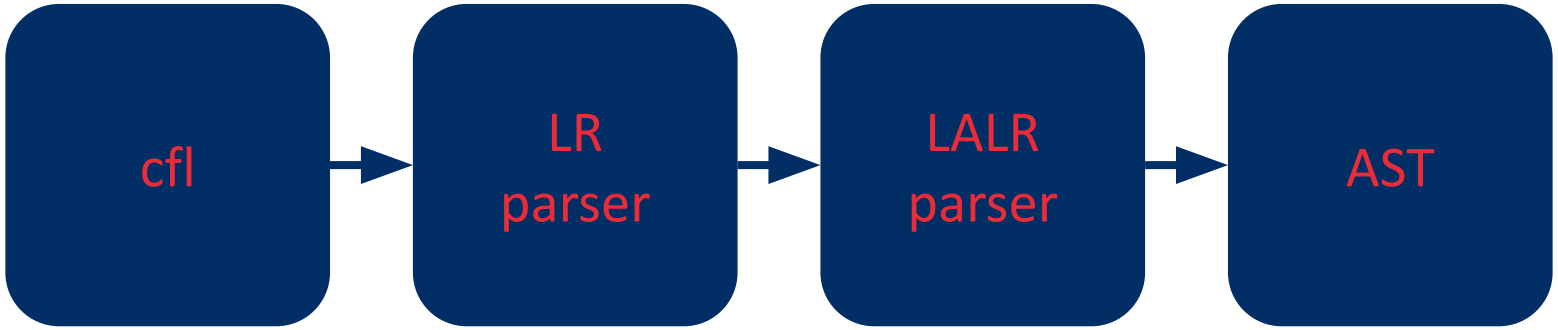
Features

1. Een CFL voor single- en multi-tape Turing machines.

```
Q0 -> Q1
Q1 -> Q2 * Q3 \n ' Q2 ' Q4 Q6 Q2 Q5
Q2 -> A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z|0|1|2|3|4|5|6|7|8|9|_|' '
Q3 -> ~ | e
Q4 -> ' Q2 ' Q4 Q6 | e
Q5 -> \n Q2 Q3 \n ' Q2 ' Q4 Q6 Q2 Q5 | e
Q6 -> { Q2 Q7 }
Q7 -> < | >
```

Features

2. Een parsing algoritme uit de volgende lijst: LR, SLR, LALR, Early.



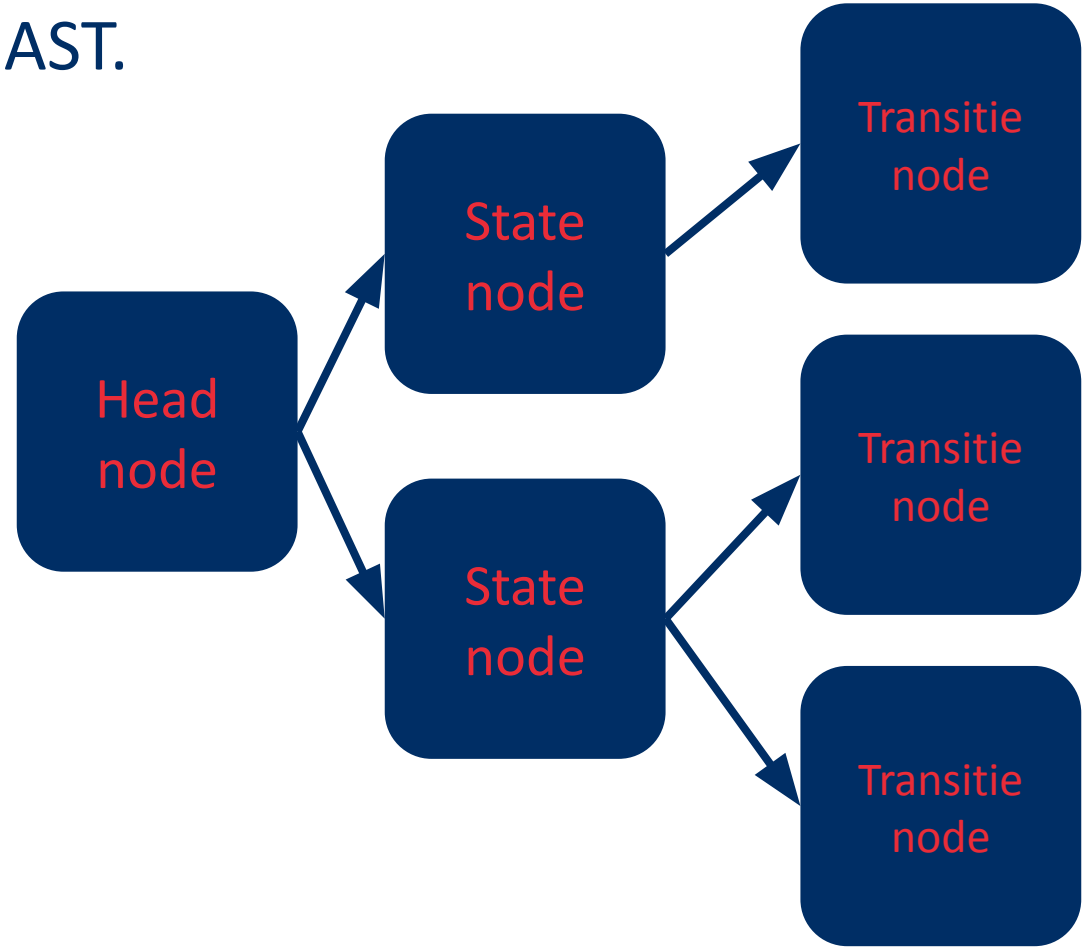
Features

	Actions			Goto	
	c	d	$	S	C
0	s3	s4		1	2
1			acc		
2	s6	s7			5
3	s3	s4			8
4	r3	r3			
5			r1		
6	s6	s7			9
7			r3		
8	r2	r2			
9			r2		

	Actions			Goto	
	c	d	$	S	C
0	s(3,6)	s(4,7)		1	2
1			acc		
2	s(3,6)	s(4,7)			5
(3,6)	s(3,6)	s(4,7)			(8,9)
(4,7)	r3	r3	r3		
5			r1		
(8,9)	r2	r2	r2		

Features

9. Code genereren aan de hand van AST.
10. Error detection AST.



Features

3. Aanmaken van een single-tape Turing machine.
4. Simulatie van een single-tape Turing machine.
7. Het inlezen van een multi-tape Turing machine.
8. Het simuleren van een multi-tape tape Turing machine.

```
A*
    'K' 'L' {C >} {D >} T
T
    'L' 'Q' {Z <} {G >} B
T
    'K' 'E' {K >} {L <} A
B~
    'E' 'G' {U >} {D <} A
```

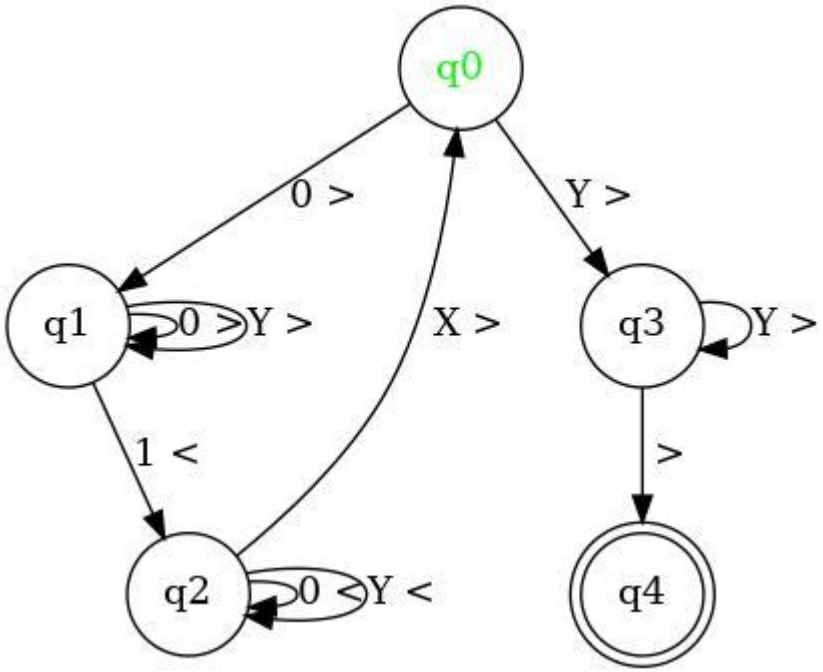

MultitapeTuringMachine
-states: vector<mtmState::ptr> -starting: mtmState::ptr -k: int
+MultitapeTuringMachine(k : int) +MultitapeTuringMachine(ast : const AbstractSyntaxTree&) +constructTape(in : const string&) : vector<vector<string>> +addState(nState : const mtmState::ptr&) : void +setStarting(sState : const mtmState::ptr&) : void +simulate(input : const string&) : bool +simulate_return(input : const string&) : vector<vector<string>> +simStep(mtmState::ptr& curState, vector<vector<string>>& tapes, vector<int>& tapeHeads) : bool +getStarting() : const mtmState::ptr& +getStates() : vector<mtmState::ptr>&

Features

5. Dynamische visualisatie van de werking van een single-tape Turing machine.

Features

6. Statische visualisatie van een single-tape Turing machine.



Tests

Tests

- Used the Catch2 framework
- **Implemented 18 tests**

Tests

```
TEST_CASE("TM_TESTS") {
    TuringMachine TM;

    std::vector<tmState::ptr> sV;
    sV.push_back(std::make_shared<tmState>("q0"));
    sV.push_back(std::make_shared<tmState>("q1"));
    sV.push_back(std::make_shared<tmState>("q2"));
    sV.push_back(std::make_shared<tmState>("q3"));
    sV.push_back(std::make_shared<tmState>("q4", true));

    for (auto state: sV) {
        TM.addState(state);
    }
    TM.setStarting(sV[0]);

    sV[0]->addTransition("0", tmTransition(sV[1], true, "X"));
    sV[1]->addTransition("0", tmTransition(sV[1], true, "0"));
    sV[2]->addTransition("0", tmTransition(sV[2], false, "0"));

    sV[1]->addTransition("1", tmTransition(sV[2], false, "Y"));

    sV[2]->addTransition("X", tmTransition(sV[0], true, "X"));

    sV[0]->addTransition("Y", tmTransition(sV[3], true, "Y"));
    sV[1]->addTransition("Y", tmTransition(sV[1], true, "Y"));
    sV[2]->addTransition("Y", tmTransition(sV[2], false, "Y"));
    sV[3]->addTransition("Y", tmTransition(sV[3], true, "Y"));

    sV[3]->addTransition("_", tmTransition(sV[4], true, "_"));

    SECTION("SIMULATION TEST 1") {
        REQUIRE(TM.simulate("0011") == true);
    }SECTION("SIMULATION TEST 2") {
        REQUIRE(TM.simulate("00011") == false);
    }
}
```

Q&A

Bedankt voor jullie aandacht!