Rule	Definition	Command Binding
1	`( ( Rule#66 | Rule#67 ) ( ';' )? )*`	
	Accepts any rules repeatably as described in the specifications.	
2	`DO BRAKE id`	`CommandBehavioralBrake`
	Applies the brakes to reference engine `id`.	
3	`@DO FORCE id DIRECTION ( FORWARD | BACKWARD )`	`CommandBehavioralForceDirection`
	Forces the direction of reference engine `id` to change instantaneously.	
4	`@DO FORCE id1 POSITION TRACK id2 DISTANCE number FROM ( START | END )`	`CommandBehavioralForcePosition`
	Forces the position of reference engine `id1` to track `id2` at `number` meters from the start or end.	
5	`@DO FORCE id SPEED number`	`CommandBehavioralForceSpeed`
	Forces the speed of reference engine `id` to change instantaneously.	
6*	`DO SELECT DRAWBRIDGE id POSITION ( UP | DOWN )`	`CommandBehavioralSelectBridge`
	Instructs drawbridge `id` to raise or lower.	
7*	`DO SELECT ROUNDHOUSE id POSITION angle ( CLOCKWISE | COUNTERCLOCKWISE )`	`CommandBehavioralSelectRoundhouse`
	Instructs the turntable of roundhouse `id` to rotate to `angle` in a particular direction.	
8	`DO SELECT SWITCH id PATH ( PRIMARY | SECONDARY )`	`CommandBehavioralSelectSwitch`
	Instructs switch `id` to change positions. The positions of primary and secondary depend on the type of switch and its configuration.	
9	`DO SELECT WATER TANK id FLOW ( ON | OFF )`	`CommandBehavioralSelectWaterTank`
	Instructs water tank `id` to start or stop its flow.	
10	`@DO SET COLLISIONS ( ENABLE | DISABLE )`	`CommandBehavioralSetCollisions`
	Enables or disables collision handling for rolling stock.	
11	`DO SET id DIRECTION ( FORWARD | BACKWARD )`	`CommandBehavioralSetDirection`
	Instructs reference engine `id` to change direction. The engine must not be in motion.	
12	`DO SET REFERENCE ENGINE id`	`CommandBehavioralSetReference`
	Makes engine `id` the reference engine, which is the one to communicate with behaviorially in a train. This automatically unsets any other engine that already was the reference.	
13	`DO SET SEMAPHORE id ( STOP | CAUTION | PROCEED )`	`CommandBehavioralSetSemaphore`
	Instructs semaphore `id` to change state.	
14	`DO SET SIGNAL LIGHT id ( STOP | PROCEED )`	`CommandBehavioralSetSignalLight`
	Instructs signal light `id` to change state.	
15	`DO SET id SPEED number`	`CommandBehavioralSetSpeed`
	Instructs reference engine `id` to change speed.	
16	`CREATE ACTUATOR id1 AS CROSSBUCK ON TRACK id2 DISTANCE number FROM ( START | END )`	`CommandCreateActuatorCrossbuck`
	Creates crossbuck `id1` on track `id2` at `number` meters from the start or end.	
17	`CREATE ACTUATOR id1 AS GATE ON TRACK id2 DISTANCE number FROM ( START | END )`	`CommandCreateActuatorGate`
	Creates gate `id1` on track `id2` at `number` meters from the start or end.	
18	`CREATE ACTUATOR id1 AS SEMAPHORE ON TRACK id2 DISTANCE number FROM ( START | END ) TOWARD ( START | END )`	`CommandCreateActuatorSemaphore`
	Creates semaphore `id1` on track `id2` at `number` meters from the start or end facing toward the start or end.	
19	`CREATE ACTUATOR id1 AS SIGNAL LIGHT ON TRACK id2 DISTANCE number FROM ( START | END ) TOWARD ( START | END )`	`CommandCreateActuatorSignalLight`
	Creates signal light `id1` on track `id2` at `number` meters from the start or end facing toward the start or end.	

20	`CREATE ACTUATOR id1 AS STATION ON TRACK id2 DISTANCE number FROM ( START | END )`	`CommandCreateActuatorStation`

Creates train station `id1` on track `id2` at `number` meters from the start or end.

21	`CREATE ACTUATOR id1 AS WATER TANK ON TRACK id2 DISTANCE number FROM ( START | END )`	`CommandCreateActuatorWaterTank`

Creates water tank `id1` on track `id2` at `number` meters from the start or end.

22	`CREATE POWER CATENARY id1 WITH POLES id`*n*`+`	`CommandCreatePowerCatenary`

Creates catenary array `id1` consisting of power poles `id`*n*.

23	`CREATE POWER POLE id1 ON TRACK id2 DISTANCE number FROM ( START | END )`	`CommandCreatePowerPole`

Creates power pole `id1` on track `id2` at `number` meters from the start or end.

24	`CREATE POWER STATION id1 REFERENCE (coordinates_world	('$' id2)) DELTA coordinates_delta WITH (SUBSTATION | SUBSTATIONS) id`*n*`+`	`CommandCreatePowerStation`

Creates power station `id1` at `coordinates_delta` meters from `coordinates_world` or `id2` with substation(s) `id`*n*. See command #66.

25	`CREATE POWER SUBSTATION id1 REFERENCE (coordinates_world	('$' id2)) DELTA coordinates_delta WITH CATENARIES id`*n*`+`	`CommandCreatePowerSubstation`

Creates power substation `id1` at `coordinates_delta` meters from `coordinates_world` or `id2` with catenaries `id`*n*.

26	`CREATE SENSOR id1 FOR OCCUPANCY ON TRACK id2 DISTANCE number1 FROM ( START | END ) RANGE number2`	`CommandCreateSensorOccupancy`

Creates occupancy sensor `id1` on track `id2` at `number1` meters from the start or end, becoming active when stock comes within `number2` meters.

27	`CREATE SENSOR id1 FOR SPEED ON TRACK id2 DISTANCE number1 FROM ( START | END ) RANGE number2`	`CommandCreateSensorSpeed`

Creates speed sensor `id1` on track `id2` at `number1` meters from the start or end, becoming active when stock comes within `number2` meters.

28	`CREATE STOCK CAR id AS BOX`	`CommandCreateStockCarBox`

Creates box car `id`.

29	`CREATE STOCK CAR id AS CABOOSE`	`CommandCreateStockCarCaboose`

Creates caboose `id`.

30*	`CREATE STOCK CAR id AS FLATBED`	`CommandCreateStockCarFlatbed`

Creates flatbed car `id`.

31*	`CREATE STOCK CAR id AS PASSENGER`	`CommandCreateStockCarPassenger`

Creates passenger car `id`.

32*	`CREATE STOCK CAR id AS TANK`	`CommandCreateStockCarTank`

Creates tank car `id`.

33*	`CREATE STOCK CAR id AS TENDER`	`CommandCreateStockCarTender`

Creates tender `id`.

34	`CREATE STOCK ENGINE id1 AS DIESEL ON TRACK id2 DISTANCE number FROM ( START | END ) FACING ( START | END )`	`CommandCreateStockEngineDiesel`

Creates diesel engine `id1` on track `id2` at `number` meters from the start or end facing the start or end.

35	`CREATE STOCK ENGINE id1 AS DIESEL ELECTRIC ON TRACK id2 DISTANCE number FROM ( START | END ) FACING ( START | END )`	`CommandCreateStockEngineDieselElectric`

Creates diesel-electric engine `id1` on track `id2` at `number` meters from the start or end facing the start or end.

36	`CREATE STOCK ENGINE id1 AS ELECTRIC ON TRACK id2 DISTANCE number FROM ( START | END ) FACING ( START | END )`	`CommandCreateStockEngineElectric`

Creates electric engine `id1` on track `id2` at `number` meters from the start or end facing the start or end.

37	`CREATE STOCK ENGINE id1 AS STEAM WITH WATER SUPPLY number2 RATE number3 ON TRACK id2 DISTANCE number1 FROM ( START | END ) FACING ( START | END )`	`CommandCreateStockEngineSteam`

Creates steam engine `id1` on track `id2` at `number1` meters from the start or end facing the start or end, initially with `number2` gallons of water on board and consuming `number3` gallons per kilometer per hour of speed.

38	`CREATE STOCK ENGINE id1 AS SWITCHER ON TRACK id2 DISTANCE number FROM ( START | END ) FACING ( START | END )`	`CommandCreateStockEngineSwitcher`

Creates switcher engine `id1` on track `id2` at `number` meters from the start or end facing the start or end.

| 39* | `CREATE TRACK BRIDGE DRAW id1 REFERENCE ( coordinates_world | ( '$' id2 ) ) DELTA START coordinates_delta1 END coordinates_delta2 ANGLE angle` | `CommandCreateTrackBridgeDraw` |

Creates drawbridge track `id1` starting at `coordinates_delta1` meters and ending at `coordinates_delta2` meters from `coordinates_world` or `id2` with maximum elevation `angle`.

| 40* | `CREATE TRACK BRIDGE id1 REFERENCE ( coordinates_world | ( '$' id2 ) ) DELTA START coordinates_delta1 END coordinates_delta2` | `CommandCreateTrackBridgeFixed` |

Creates fixed bridge track `id1` starting at `coordinates_delta1` meters and ending at `coordinates_delta2` meters from `coordinates_world` or `id2`.

| 41* | `CREATE TRACK CROSSING id1 REFERENCE ( coordinates_world | ( '$' id2 ) ) DELTA START coordinates_delta1 END coordinates_delta2` | `CommandCreateTrackCrossing` |

Creates grade-crossing track `id1` starting at `coordinates_delta1` meters and ending at `coordinates_delta2` meters from `coordinates_world` or `id2`.

| 42* | `CREATE TRACK CROSSOVER id1 REFERENCE ( coordinates_world | ( '$' id2 ) ) DELTA START coordinates_delta1 END coordinates_delta2 START coordinates_delta3 END coordinates_delta4` | `CommandCreateTrackCrossover` |

Creates crossover track `id1` with segment 1 starting at `coordinates_delta1` meters and ending at `coordinates_delta2` meters from `coordinates_world` or `id2`, and segment 2 starting at `coordinates_delta3` meters and ending at `coordinates_delta4` meters.

| 43 | `CREATE TRACK CURVE id1 REFERENCE ( coordinates_world | ( '$' id2 ) ) DELTA START coordinates_delta1 END coordinates_delta2 ( ( DISTANCE ORIGIN number ) | ( ORIGIN coordinates_delta3 ) )` | `CommandCreateTrackCurve` |

Creates curve track `id1` starting at `coordinates_delta1` meters and ending at `coordinates_delta2` meters from `coordinates_world` or `id2`, with either the orthogonal to the line connecting the start and end `number` meters long or the origin `coordinates_delta3` meters from the reference. Lecture will go over this coordinate configuration. The complexity is why Task 3 uses only straight segments.

| 44* | `CREATE TRACK END id1 REFERENCE ( coordinates_world | ( '$' id2 ) ) DELTA START coordinates_delta1 END coordinates_delta2` | `CommandCreateTrackEnd` |

Creates termination track `id1` starting at `coordinates_delta1` meters and ending at `coordinates_delta2` meters from `coordinates_world` or `id2`.

| 45 | `CREATE TRACK LAYOUT id1 WITH TRACKS id`*n*`+` | `CommandCreateTrackLayout` |

Creates track layout `id1` with tracks `id`*n*.

| 46 | `CREATE TRACK ROUNDHOUSE id1 REFERENCE ( coordinates_world | ( '$' id2 ) ) DELTA ORIGIN coordinates_delta1 ANGLE ENTRY angle1 START angle2 END angle3 WITH integer SPURS LENGTH number1 TURNTABLE LENGTH number2` | `CommandCreateTrackRoundhouse` |

Creates roundhouse track `id1` centered at `coordinates_delta1` meters from `coordinates_world` or `id2` with entry track angle `angle1` and `integer` spurs of length `number1` starting at `angle2` and ending at `angle3`. The turntable is `number2` meters in diameter.

| 47 | `CREATE TRACK STRAIGHT id1 REFERENCE ( coordinates_world | ( '$' id2 ) ) DELTA START coordinates_delta1 END coordinates_delta2` | `CommandCreateTrackStraight` |

Creates straight track `id1` starting at `coordinates_delta1` meters and ending at `coordinates_delta2` meters from `coordinates_world` or `id2`.

| 48 | `CREATE TRACK SWITCH TURNOUT id1 REFERENCE ( coordinates_world | ( '$' id2 ) ) STRAIGHT DELTA START coordinates_delta1 END coordinates_delta2 CURVE DELTA START coordinates_delta3 END coordinates_delta4 DISTANCE ORIGIN number` | `CommandCreateTrackSwitchTurnout` |

Creates turnout switch track `id1` with the straight segment starting at `coordinates_delta1` meters and ending at `coordinates_delta2` meters from `coordinates_world` or `id2` and the curved segment starting at `coordinates_delta3` meters and ending at `coordinates_delta4` meters with the tangent to the line connecting the start and end of the curved segment `number` meters long.

| 49 | `CREATE TRACK SWITCH WYE id1 REFERENCE ( coordinates_world | ( '$' id2 ) ) DELTA START coordinates_delta1 END coordinates_delta2 DISTANCE ORIGIN number1 DELTA START coordinates_delta3 END coordinates_delta4 DISTANCE ORIGIN number2` | `CommandCreateTrackSwitchWye` |

Creates wye switch track `id1` with the first segment starting at `coordinates_delta1` meters and ending at `coordinates_delta2` meters from `coordinates_world` or `id2` and the second segment starting at `coordinates_delta3` meters and ending at `coordinates_delta4` meters, both with the tangent to the line connecting the start and end of the curved segment `number1` and `number2` meters long, respectively.

| 50 | *This command left intentionally undefined* | |

| 51* | `@EXIT` | `CommandMetaDoMetaExit` |

Exits the system.

| 52 | `@RUN string` | `CommandMetaDoMetaRun` |

Loads and executes the script file in fully qualified filename `string`. Be careful with file paths because their handling varies by operating system.

| ~~53~~ | ~~`@SCHEDULE AT number Rule#67`~~ | ~~`CommandMetaDoMetaSchedule`~~ |

Schedules any command in `Rule#67` to execute at absolute simulation time `number`, which must be in the future.

| 54 | `@WAIT number` | `CommandMetaDoMetaWait` |

Waits `number` seconds before executing the next command.

| 55* | `CLOSE VIEW id` | `CommandMetaViewDestroy` |

Closes view `id`.

| 56 | OPEN VIEW id1 ORIGIN (coordinates_world \| ('$' id2)) WORLD WIDTH integer1 SCREEN WIDTH integer2 HEIGHT integer3 | CommandMetaViewGenerate |

Creates view `id1` at origin `coordinates_world` or `id2` where the world is `integer1` meters wide and the screen is `integer2` pixels wide and `integer3` high.

| 57 | SYNC VIEW id1 NORTH ON id2 | CommandMetaViewSync |

Centers view `id1` on stock `id2` with up representing north.

| 58 | SYNC VIEW id1 TRACK ON id2 | CommandMetaViewSync |

Centers view `id1` on stock `id2` with up representing the direction of `id2`.

| 59 | UNSYNC VIEW id | CommandMetaViewUnsync |

Uncenters view `id` and reverts it to free mode with up representing north.

| 60 | COMMIT | CommandStructuralCommit |

Commits creational and structural configurations and prevents any further changes.

| 61 | COUPLE STOCK id1 AND id2 | CommandStructuralCouple |

Couples stock `id1` to `id2`.

| 62 | LOCATE STOCK id1 ON TRACK id2 DISTANCE number FROM (START \| END) | CommandStructuralLocate |

Locates stock `id1` on track `id2` at `number` meters from the start or end of the track.

| 63 | MAP OCCUPANCY (SENSOR \| SENSORS) id$n$+ TO ACTUATOR id1 | CommandStructuralMapSensorsOccupancy |

Creates a sensor network that maps occupancy sensor(s) `id`n to actuator `id1`.

| 64 | MAP SPEED (SENSOR \| SENSORS) id$n$+ TO ACTUATOR id1 RESPOND (PROCEED \| STOP) WHEN SPEED (LESS \| GREATER) THAN number | CommandStructuralMapSensorsSpeed |

Creates a sensor network that maps speed sensor(s) `id`n to actuator `id1`, triggering a proceed or stop state when stock less or greater than speed `number`.

| 65* | UNCOUPLE STOCK id1 AND id2 | CommandStructuralUncouple |

Uncouples stock `id1` from `id2`.

| 66 | USE id AS REFERENCE coordinates_world | See MyParserHelper |

Sets `id` as a short-form substitution for `coordinates_world` in any of the commands with a $ field.

| 67 | Rule#2 through Rule#65 | |

| 68 | @CLOCK | CommandMetaDoClockShow |

Prints the current simulation time.

| 69 | @CLOCK integer number | CommandMetaDoClockSetRate |

Sets the simulation clock to `integer` clock ticks per real-world second (if possible) and `number` simulation seconds per tick.

| 70 | @CLOCK (PAUSE \| RESUME) | CommandMetaDoClockSetRun |

Pauses or resumes the simulation clock.

| 71 | @CLOCK UPDATE | CommandMetaDoClockUpdate |

Forces the simulation clock to update by one tick. This is valid only when the clock is paused.