Viability Checking and Requirements Completion: How Constraint Programming Helps State Machines in Performance Requirements Engineering

Gefei Zhang
Hochschule für Technik und Wirtschaft Berlin, Germany

September 2, 2025

Motivation

- State machines very popular for modeling reactive components

$\Rightarrow$ Integrate performance modeling into state machines

Motivation

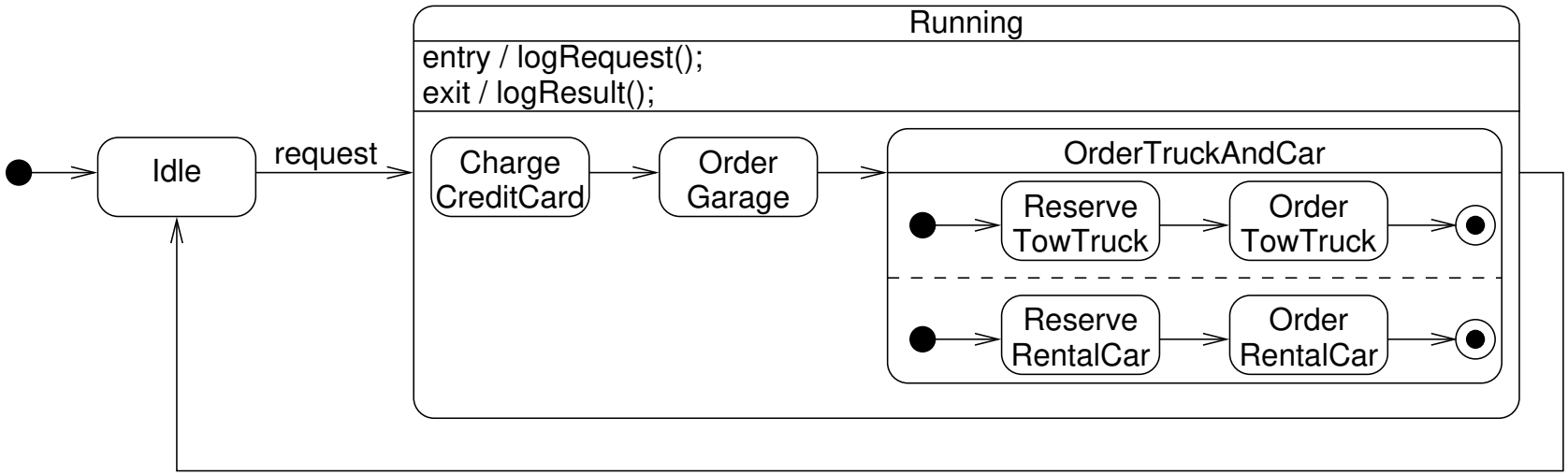
- State machines very popular for modeling reactive components

$\Rightarrow$ Integrate performance modeling into state machines

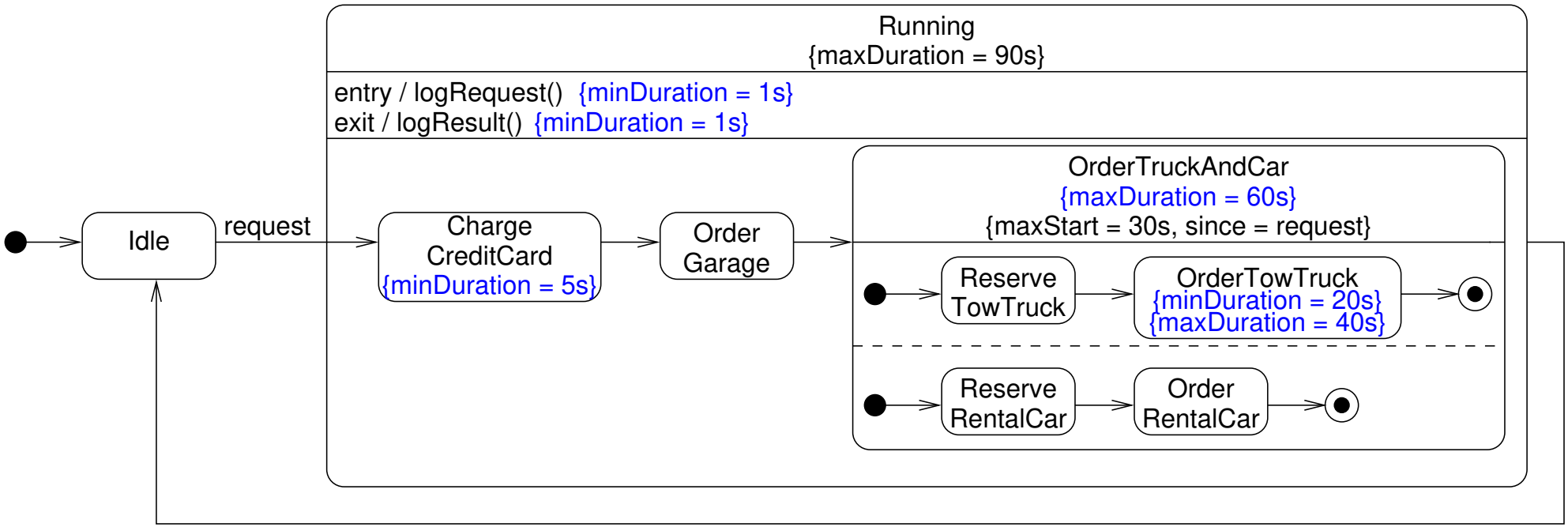
- Validate and complete the requirements

$\Rightarrow$ Use constraint programming

Running Example



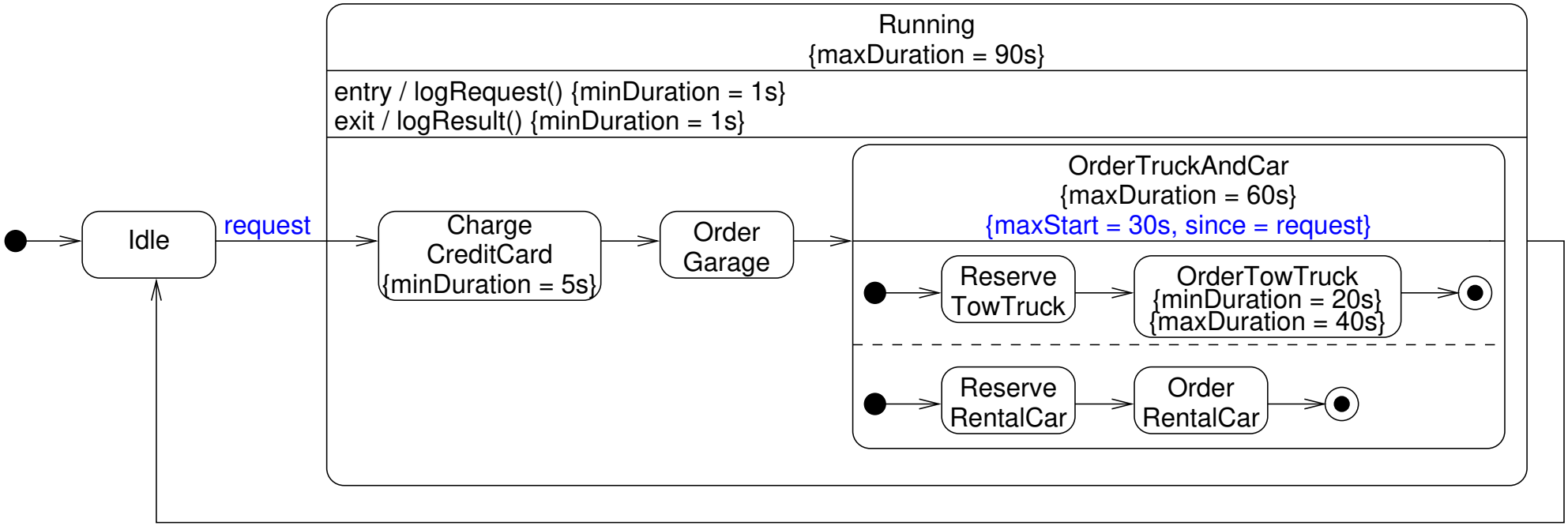
Performance Modeling in UML State Machines (1)



Assumptions

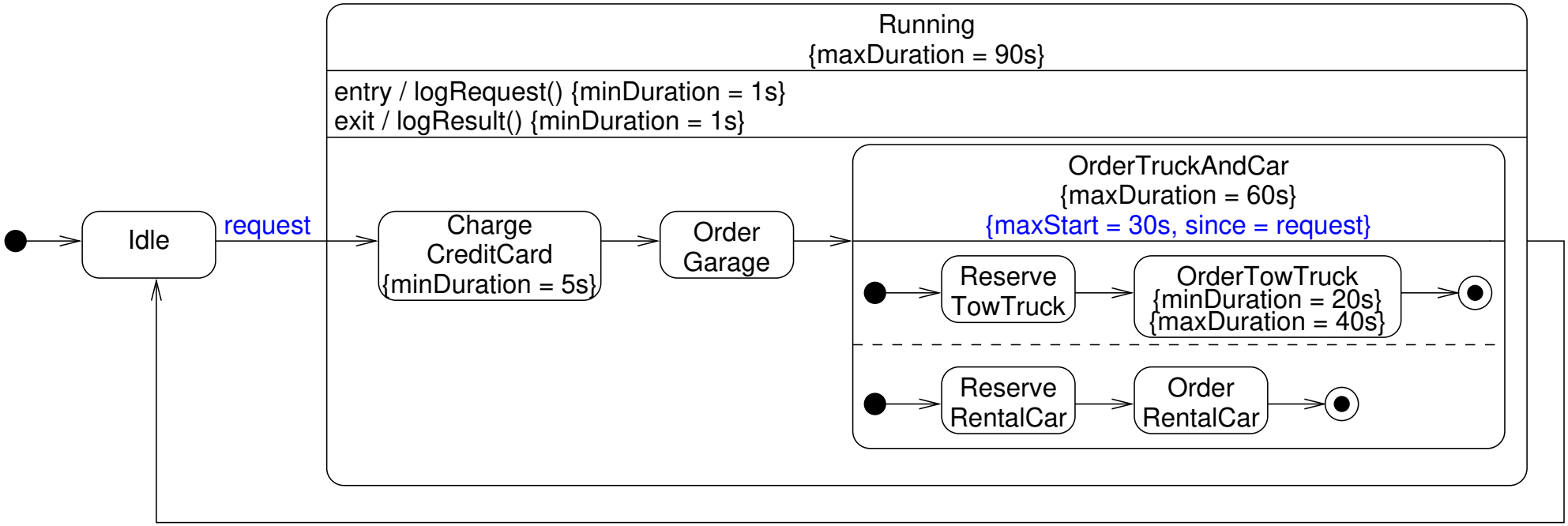
- A state may remain active for some time
- An action needs some time to finish
- A state transition succeeds immediately (unless it carries out an *action*)
- maxDuration: requirement; minDuration: constraint

Performance Modeling in UML State Machines (2)



- maxStart: latest starting time since some event

Performance Modeling in UML State Machines (2)



- maxStart: latest starting time since some event

Next

- Viability?
- Other states?

Constraint Programming

- Given domains of variables and a set of constraints, find possible valuations such that the constraints are satisfied

Constraint Programming

- Given domains of variables and a set of constraints, find possible valuations such that the constraints are satisfied

$$1 \leq x \leq 20$$
$$9 \leq y \leq 11$$
$$150 \leq z \leq 161$$

$$xy = z$$

Constraint Programming

- Given domains of variables and a set of constraints, find possible valuations such that the constraints are satisfied

$$1 \leq x \leq 20$$
$$9 \leq y \leq 11$$
$$150 \leq z \leq 161$$

$$xy = z$$

```
var 1..20:    x;
var 9..11:    y;
var 150..161: z;

constraint x*y = z;

solve satisfy;
```

Constraint Programming

- Given domains of variables and a set of constraints, find possible valuations such that the constraints are satisfied

$$1 \leq x \leq 20$$
$$9 \leq y \leq 11$$
$$150 \leq z \leq 161$$

$$xy = z$$

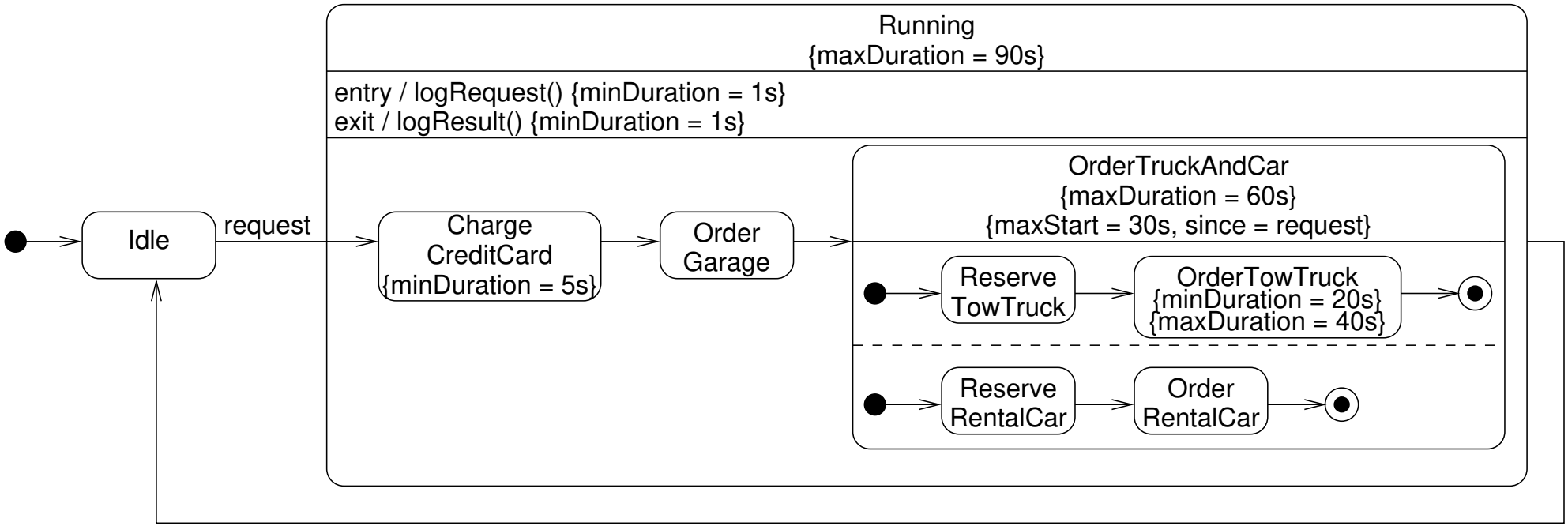
```
var 1..20:    x;
var 9..11:    y;
var 150..161: z;

constraint x*y = z;

solve satisfy;
```

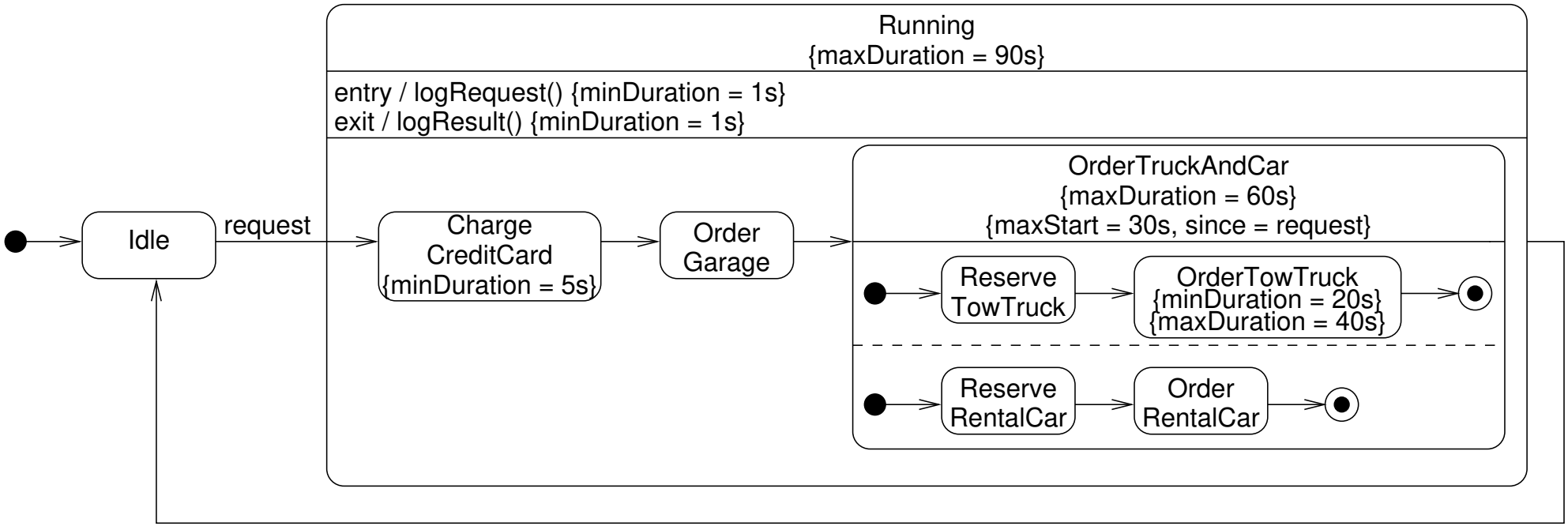
- Optimisation also possible
- For example: find the largest z satisfying the above constraint

Transforming State Machine to Constraint System



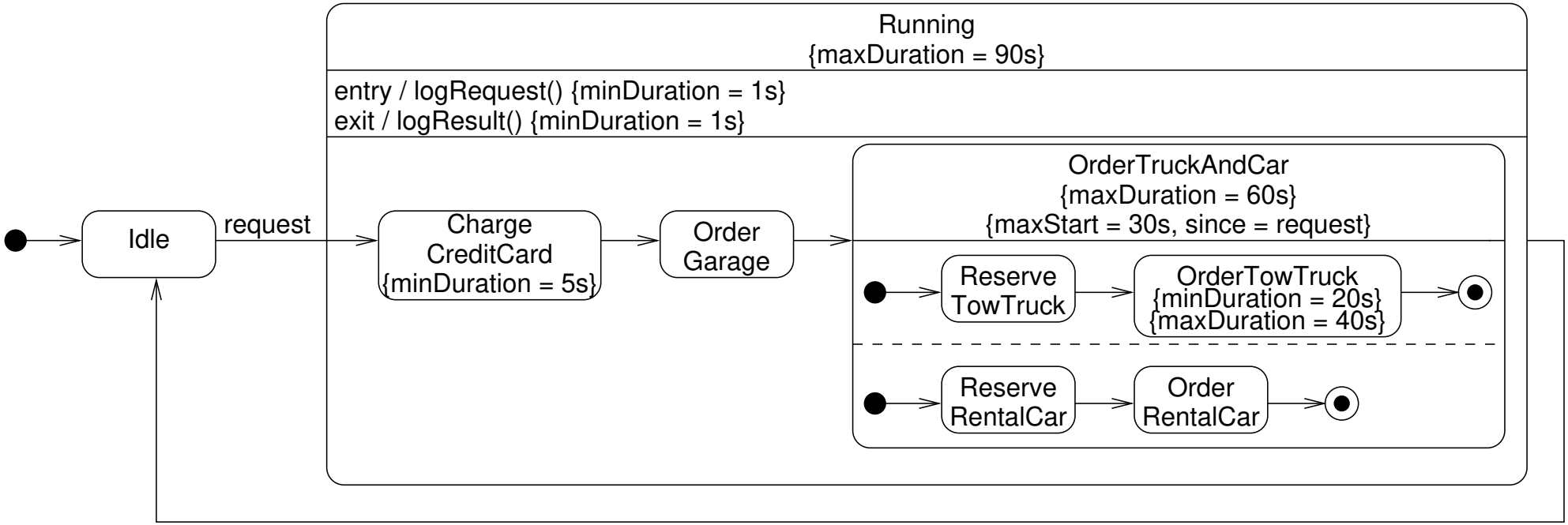

```
int: m = 90;
var 0..m: Running;
var 0..m: OrderTruckAndCar;
var 0..m: OrderTowTruck;
...
...
```

Transforming State Machine to Constraint System



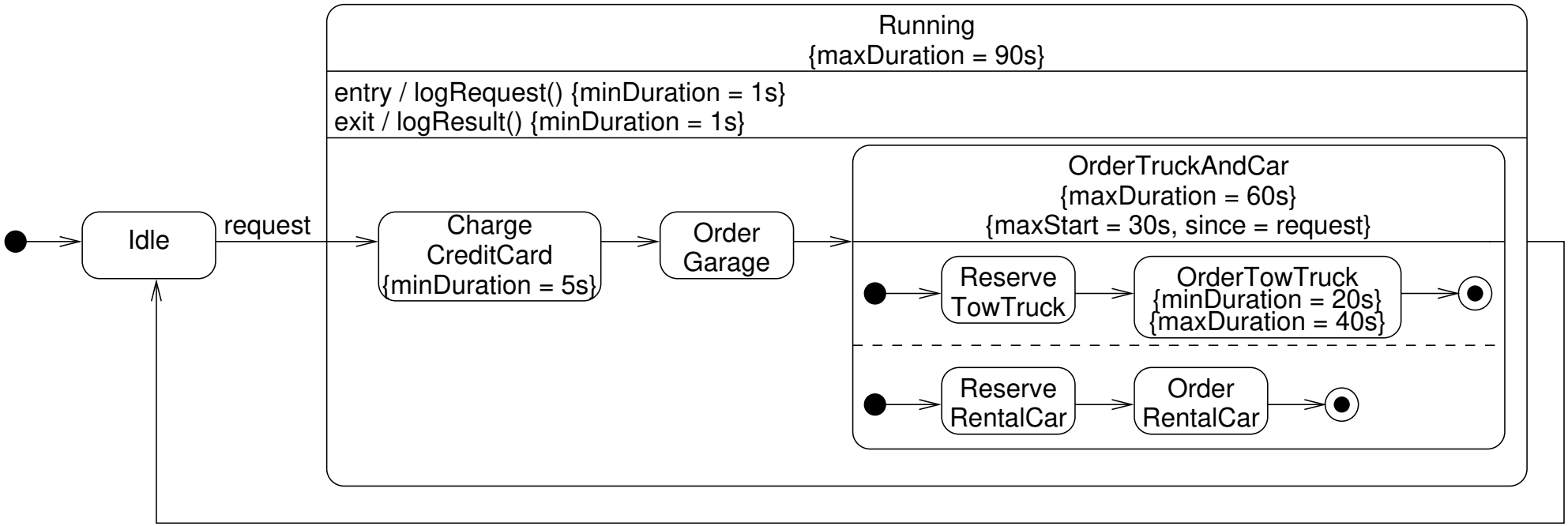

```
constraint logRequests >= 1;
constraint logResult >= 1;
constraint OrderTruckAndCar <= 60;
constraint OrderTowTruck >= 20;
constraint OrderTowTruck <= 40;
...
```

Transforming State Machine to Constraint System




```
constraint
    ReserveTowTruck + OrderTowTruck + Final1 <=
    OrderTruckAndCar;
constraint
    ReserveRentalCar + OrderRentalCar + Final2 <=
    OrderTruckAndCar;
```

Transforming State Machine to Constraint System




```
constraint
    ChargeCreditCard +
    OrderGarage +
    OrderTruckAndCar <=
    Running - logRequest - logResult;
```

Viability Checking and Requirements Completion

- Are the existing requirements satisfiable?

```
solve satisfy;
```

Viability Checking and Requirements Completion

- Are the existing requirements satisfiable?

```
solve satisfy;
```

- How much time can `OrderGarage` take at most?

```
solve maximize OrderGarage;
```

Conclusions and Future Work

- Simple notation to incorporate performance requirements into UML state machines
- Using constraint solving to check viability and complete requirements

Conclusions and Future Work

- Simple notation to incorporate performance requirements into UML state machines
- Using constraint solving to check viability and complete requirements

Future Work

- More complex state machines, including cycles
- Tool support