R5: Intersection Control in Ptolemy II

Felix Baumann
Mai Geiger
Johannes Spilka

Inhalt

Ptolemy II

- Einführung
- Funktionen
- Finite State Machine

Projekt: Intersection Control

Demo

Einführung

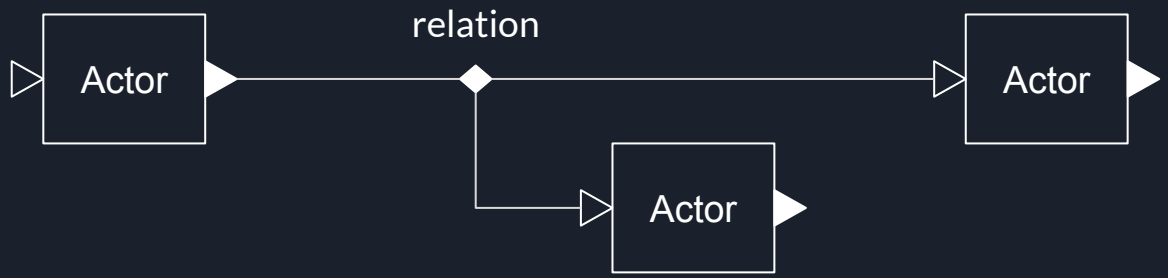
- Ptolemy II ist ein auf Java basierendes Open-Source Software Framework
- Besitzt eine GUI: Vergil
- Simulation von Modellen durch Softwarekomponenten
- Actors & Directors als Softwarekomponenten

Actor-oriented design

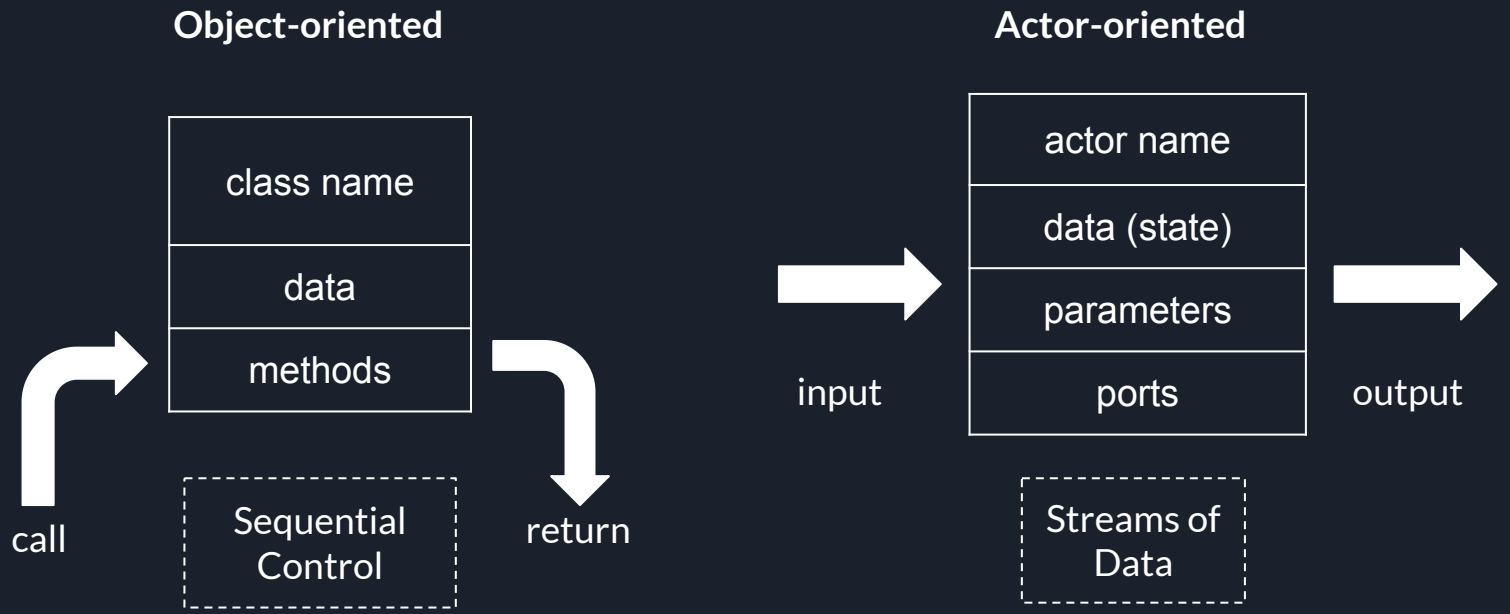
In Ptolemy II wird die Semantik eines Modells nicht durch das Framework bestimmt, sondern durch eine Softwarekomponente im Modell, die als Director bezeichnet wird und ein Berechnungsmodell implementiert.

Actors:

- verhalten sich “concurrent”
- kommunizieren untereinander via asynchroner Nachrichten
- können hierarchisch zusammengesetzt werden
- besitzen Ports, durch die sie mit anderen Actors verbunden sind

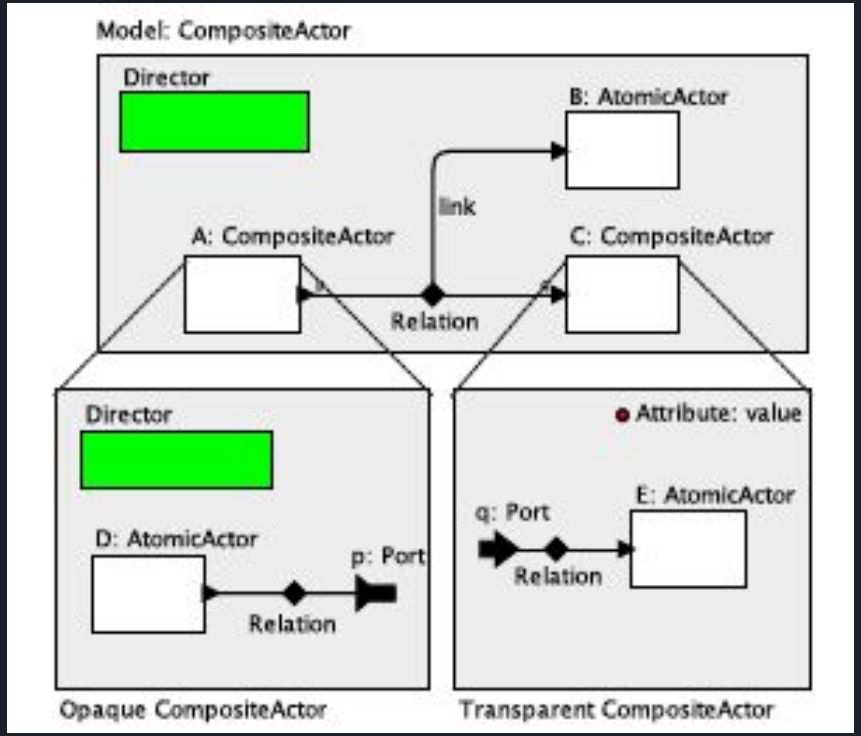


Actor Orientation vs. Object Orientation



Directors

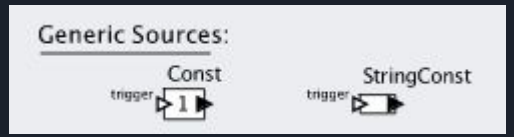
Direktoren geben den Modellen eine ganz bestimmte Bedeutung und definieren die Semantik des Modells.



Discrete Event Director (DE)

- DE Actors kommunizieren durch getimete Events
- Ein DE Director verwaltet eine Event Queue, in der Events zeitlich gespeichert werden
- Mit jeder Iteration wird das Event mit dem kleinsten Zeitstempel in der Queue zuerst verarbeitet.
- Events werden in die Queue aufgenommen,
  - wenn eine Actor die fireAt Methode des Directors aufruft
  - wenn der Input-Port einen Token erhält

Funktionen: Source Library

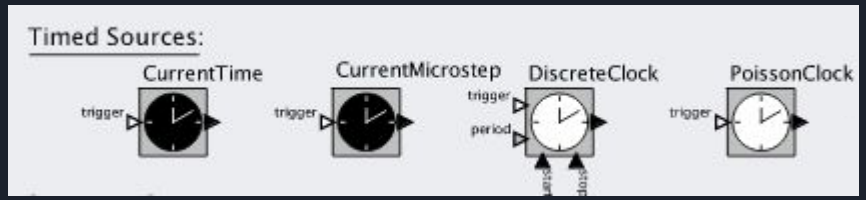


- **Const** erzeugt einen Wert (oder Ausdruck) der durch einen Parameter festgelegt wird
- **StringConst** erzeugt Strings und kann auf andere Parameter verweisen

Funktionen: Source Library

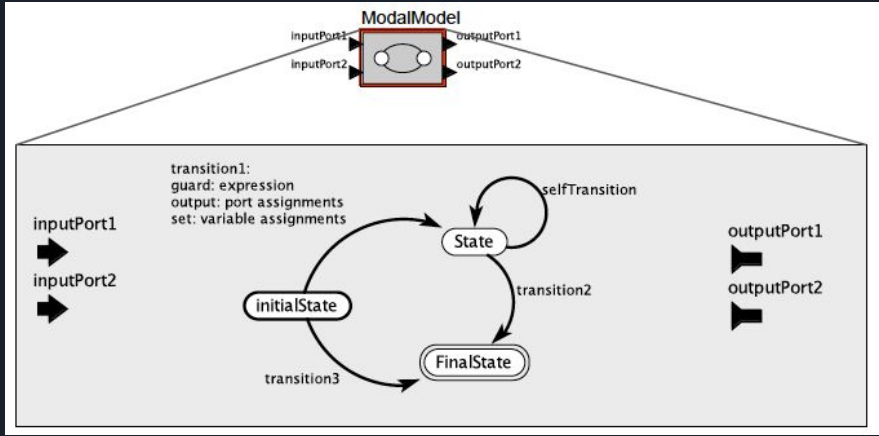
- **RecordAssembler** gibt einen Datensatz aus, der ein Feld für jeden Eingabeport enthält.
- **RecordDisassembler** extrahiert Felder aus einem Datensatz
- **Display** zeigt die Werte der Eingaben in einem Textfenster an, das geöffnet wird, wenn das Modell ausgeführt wird

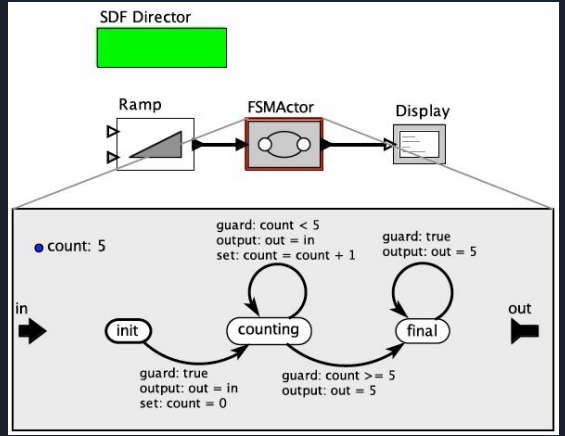
Funktionen: Source Library



- **CurrentTime** beobachtet die Modellzeit von Ereignissen
- **DiscreteClock** ist ein vielseitiger Ereignisgenerator, welcher eine Folge von Ereignissen in regelmäßigen Zeitabständen generiert
- **PoissonClock** erzeugt im Gegensatz dazu, Ereignisse zu zufälligen Zeiten

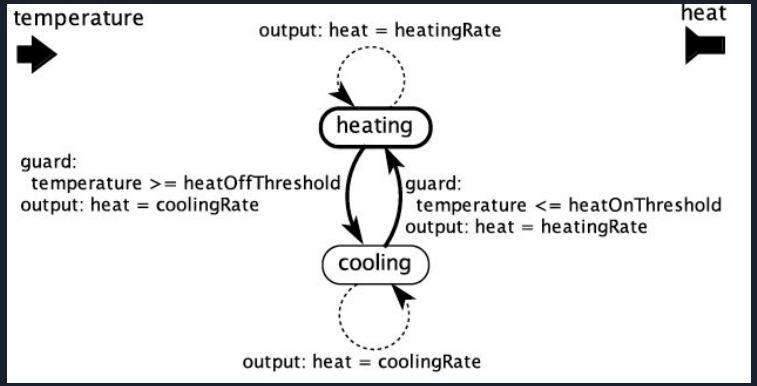
ModalModel (FSM)





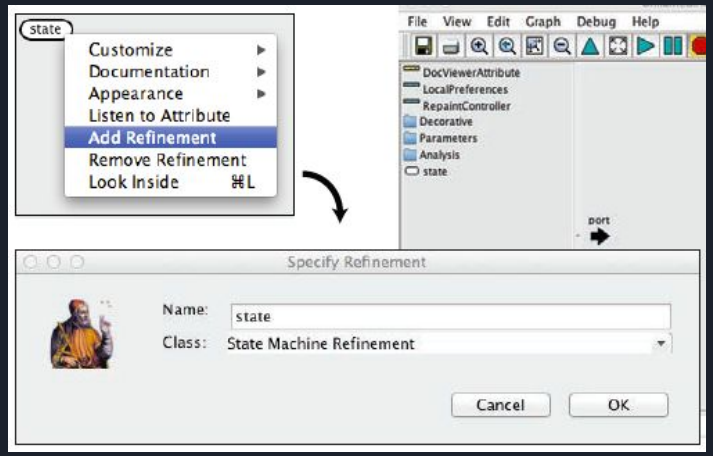
In Ptolemy II können Finite State Machine genannt **ModalModel erstellt werden.** Ein ModalModel besitzt **states**, wobei jede state als **initial state** oder **final state** markiert werden kann. Die Verbindungen zwischen den states nennt sich **transition.** Diese können verschiedene Parameter beinhalten, wie in etwa **guard expression** (Bedingung), **output data** oder **set values** der Parameter in der FSM.

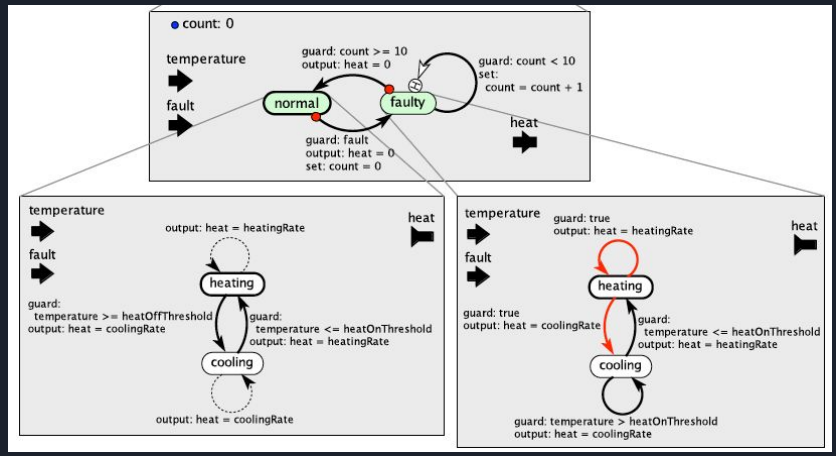
ModalModel (FSM)



Transition können sowohl eine **durchgehende** als auch eine **gepunktete Linie** zeigen. Die gepunktete Linie zeigt einen **default** an, das bedeutet, dass diese transition zum Tragen kommt, falls keine andere aktiviert wurde.

ModalModel (FSM)



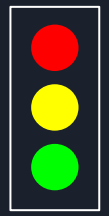


Eine Erweiterung zur regulären FSM bietet die Hierarchical FSM. Dadurch können states in einer FSM wiederum eine FSM beinhalten (oben grün eingefärbt). Dazu wählt man bei dem jeweiligen state die Option **Add Refinement** aus. Der rote Punkt markiert eine **preemptive transition** (präventiver Übergang), der dafür sorgt, dass die innere FSM nicht startet, sollte der guard der äußeren FSM aktiviert werden.

Projekt: Intersection Control

Ziel: Simulation einer Verkehrskreuzung durch Ampelregelung

Verkehrsteilnehmer: Autonome Fahrzeuge und reguläre von Menschen gesteuerte Fahrzeuge

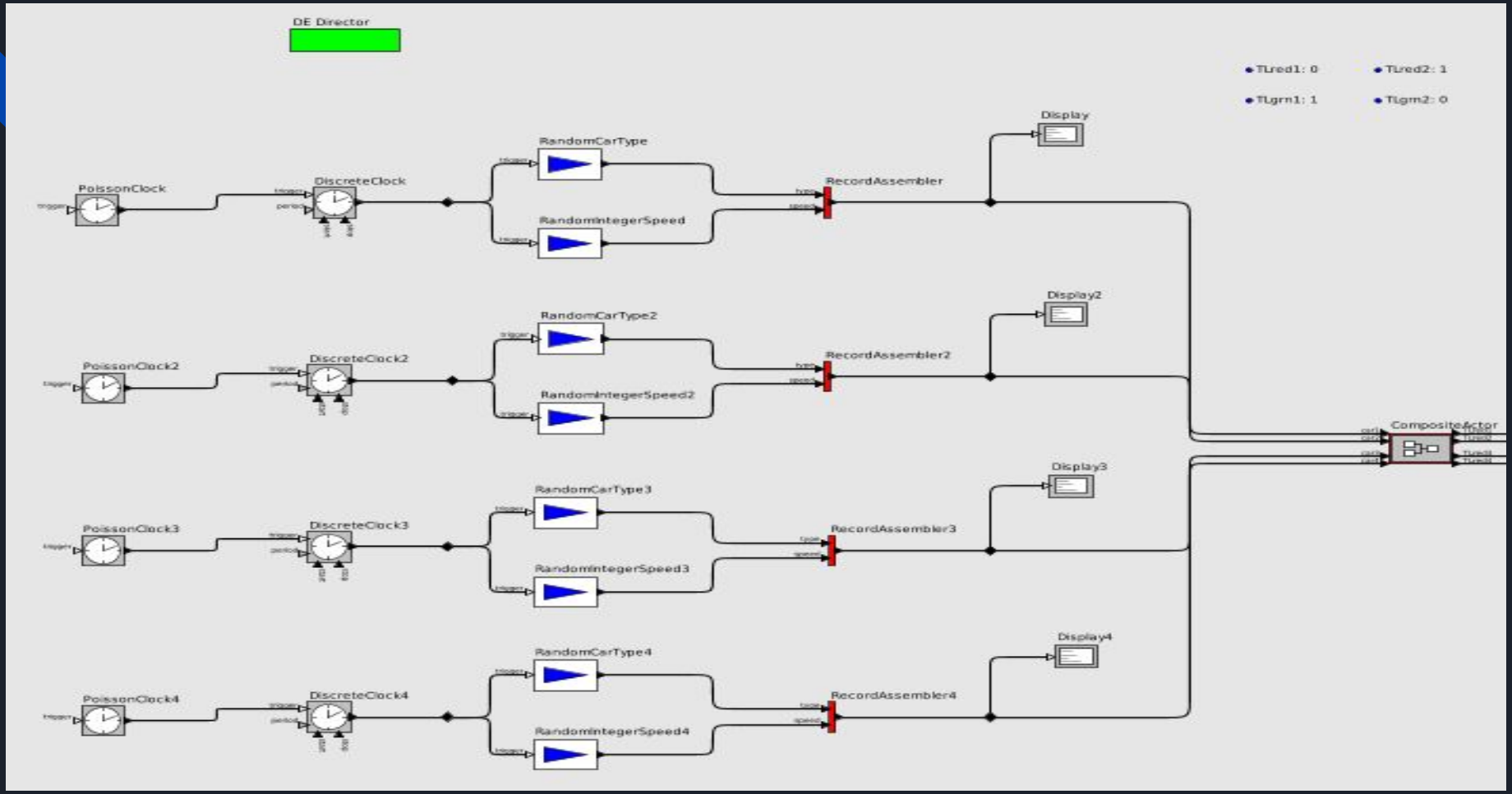


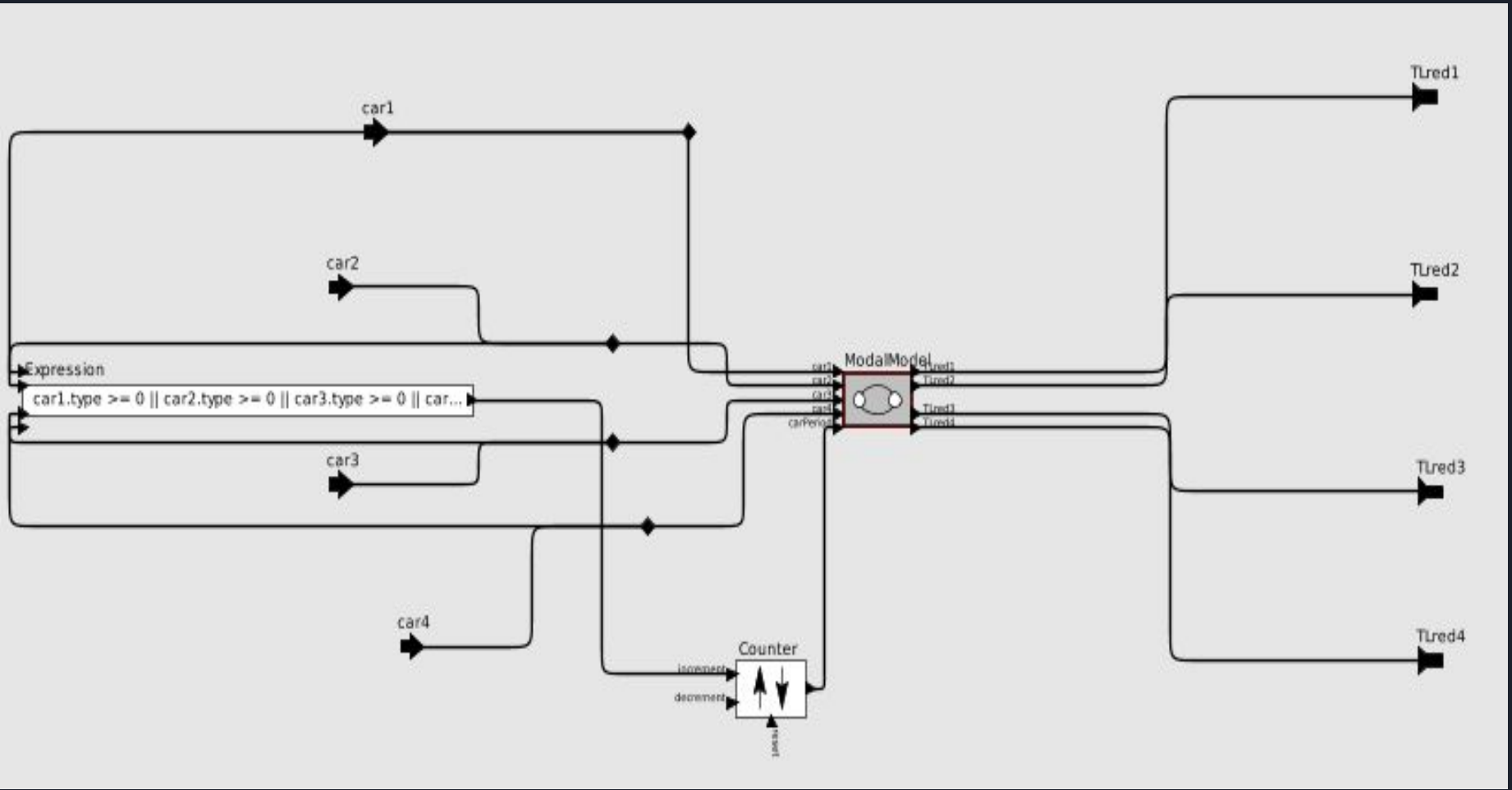
Strategie:

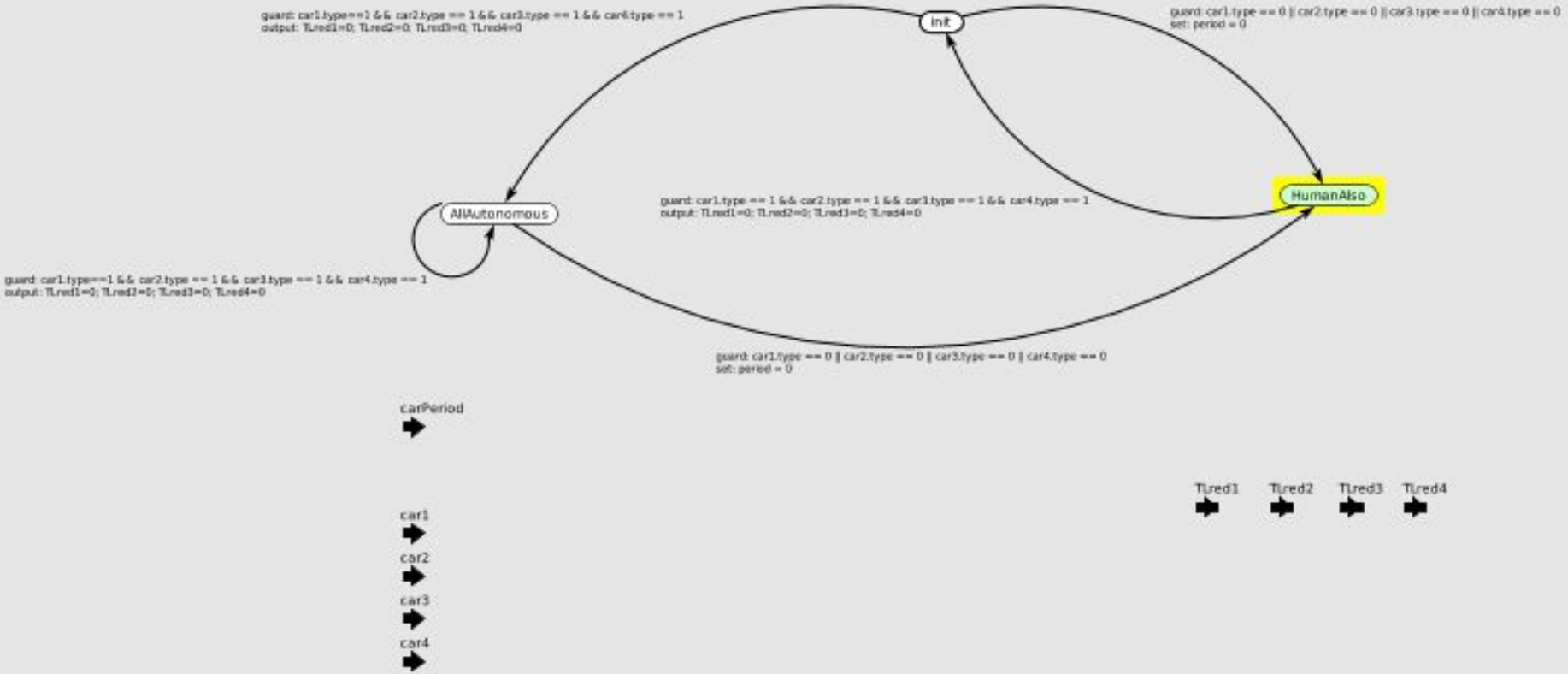
1) Falls sich nur autonome Fahrzeuge an der Kreuzung befinden,

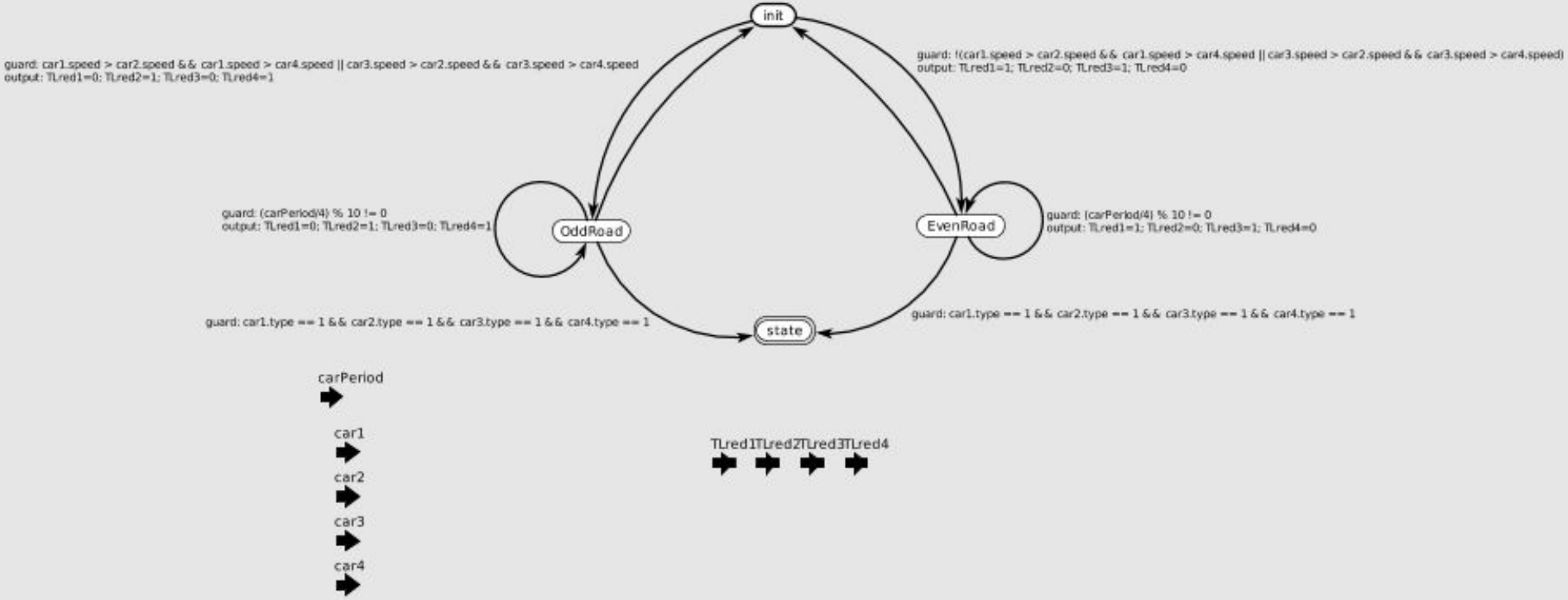
koordinieren diese ihren Zugang selbst → bedeutet, alle Ampeln sind grün.

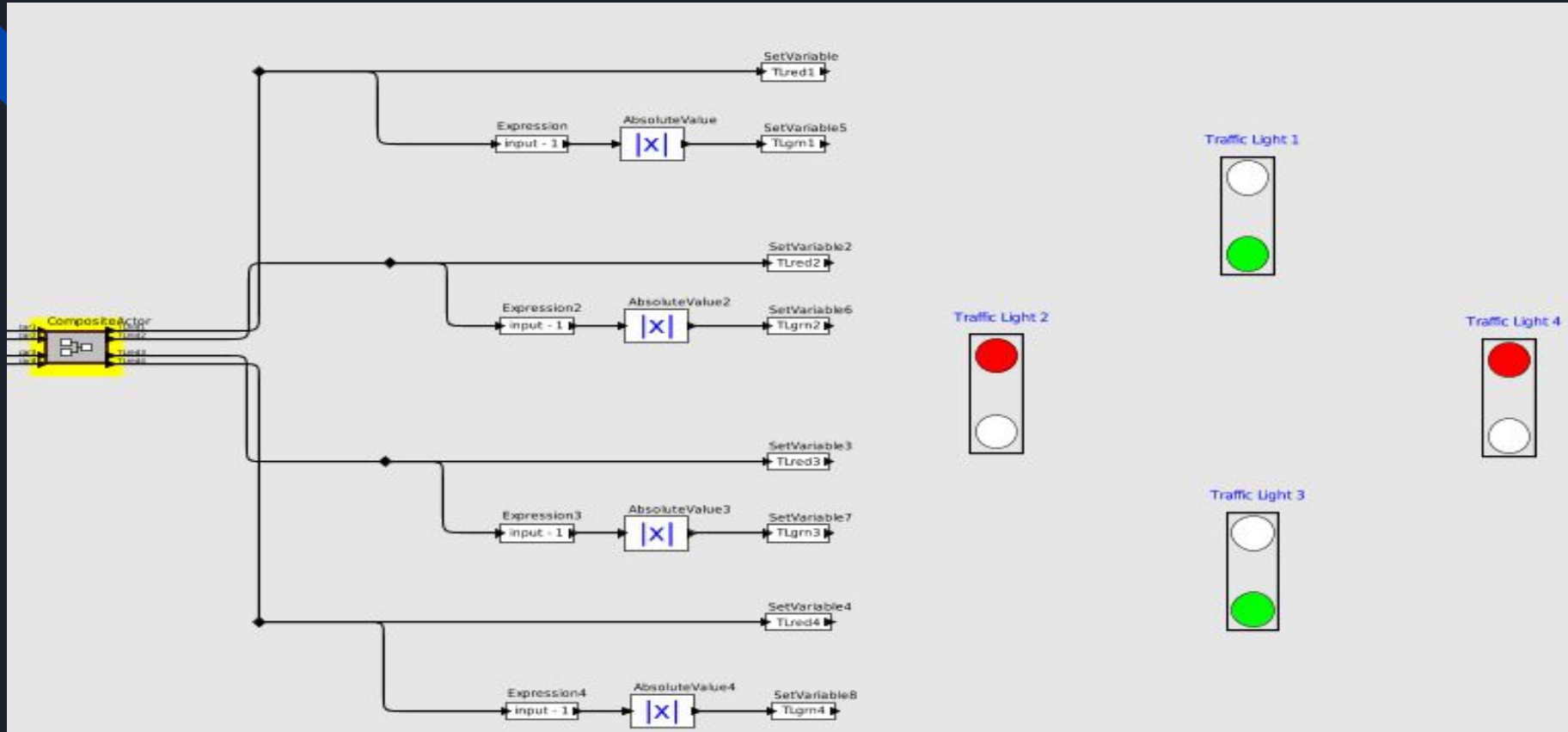
2) Human-driven Cars sind auf eine Ampelregelung angewiesen.


DE Director
TLred1: 0
TLred2: 1
TLgrn1: 1
TLgrn2: 0
PoissonClock
DiscreteClock
RandomCarType
RandomIntegerSpeed
RecordAssembler
Display
PoissonClock2
DiscreteClock2
RandomCarType2
RandomIntegerSpeed2
RecordAssembler2
Display2
PoissonClock3
DiscreteClock3
RandomCarType3
RandomIntegerSpeed3
RecordAssembler3
Display3
PoissonClock4
DiscreteClock4
RandomCarType4
RandomIntegerSpeed4
RecordAssembler4
Display4
CompositeActor


car1
car2
car3
car4
Expression
car1.type >= 0 || car2.type >= 0 || car3.type >= 0 || car...
ModalModel
TLred1
TLred2
TLred3
TLred4
Counter
increment
decrement
reset


guard: car1.type==1 && car2.type == 1 && car3.type == 1 && car4.type == 1
output: TLred1=0; TLred2=0; TLred3=0; TLred4=0
Init
guard: car1.type == 0 || car2.type == 0 || car3.type == 0 || car4.type == 0
set: period = 0
AllAutonomous
guard: car1.type == 1 && car2.type == 1 && car3.type == 1 && car4.type == 1
output: TLred1=0; TLred2=0; TLred3=0; TLred4=0
HumanAlso
guard: car1.type==1 && car2.type == 1 && car3.type == 1 && car4.type == 1
output: TLred1=0; TLred2=0; TLred3=0; TLred4=0
guard: car1.type == 0 || car2.type == 0 || car3.type == 0 || car4.type == 0
set: period = 0
carPeriod
car1
car2
car3
car4
TLred1
TLred2
TLred3
TLred4


init
guard: car1.speed > car2.speed && car1.speed > car4.speed || car3.speed > car2.speed && car3.speed > car4.speed
output: TLred1=0; TLred2=1; TLred3=0; TLred4=1
guard: !(car1.speed > car2.speed && car1.speed > car4.speed || car3.speed > car2.speed && car3.speed > car4.speed)
output: TLred1=1; TLred2=0; TLred3=1; TLred4=0
guard: (carPeriod/4) % 10 != 0
output: TLred1=0; TLred2=1; TLred3=0; TLred4=1
OddRoad
EvenRoad
guard: (carPeriod/4) % 10 != 0
output: TLred1=1; TLred2=0; TLred3=1; TLred4=0
guard: car1.type == 1 && car2.type == 1 && car3.type == 1 && car4.type == 1
state
guard: car1.type == 1 && car2.type == 1 && car3.type == 1 && car4.type == 1
carPeriod
car1
car2
car3
car4
TLred1TLred2TLred3TLred4


SetVariable
TLred1
Expression
input - 1
AbsoluteValue
|x|
SetVariable5
TLgrn1
SetVariable2
TLred2
Expression2
input - 1
AbsoluteValue2
|x|
SetVariable6
TLgrn2
CompositeActor
SetVariable3
TLred3
Expression3
input - 1
AbsoluteValue3
|x|
SetVariable7
TLgrn3
SetVariable4
TLred4
Expression4
input - 1
AbsoluteValue4
|x|
SetVariable8
TLgrn4
Traffic Light 1
Traffic Light 2
Traffic Light 3
Traffic Light 4

Live - Demo