Lecture 13: Quantitative Model Checking

So far we only answered Yes/No questions

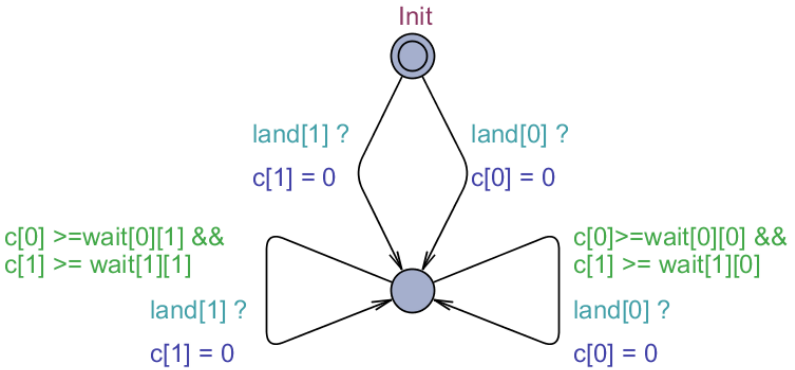
- There is need for quantitative verification
  - Quantify uncertainty
    - How often does bad events happen?
  - Quantify performance
    - What's the minimum battery consumption?
- There are tools available to evaluate
  - Probability
  - Cost/reward

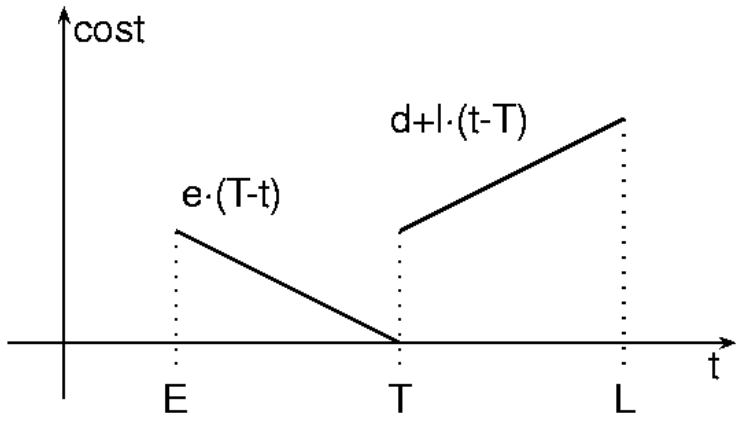
UPPAAL Tool Family

- UPPAAL CORA
  - Cost Optimal Reachability Analysis
- UPPAAL SMC
  - Statistical Model Checking
- UPPAAL TIGA
  - Controller Synthesis

UPPAAL CORA: Cost Optimal Reachability Analysis

- Linearly priced timed automata (LPTA)
- Add cost/reward to each location
- Calculates the path with minimum cost
- Can be used to model power consumption, etc





E earliest landing time

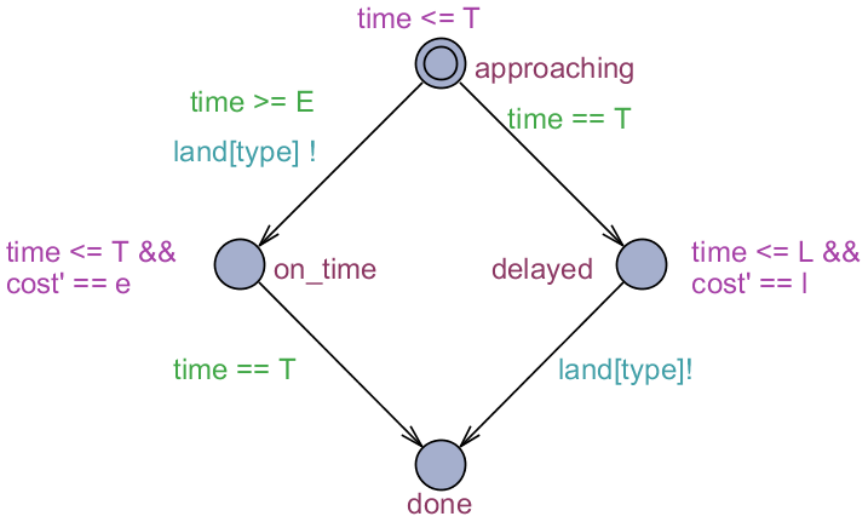
T target (cruise) landing time

L latest landing time

e early cost rate

l late cost rate

d late penalty



UPPAAL SMC

- Statistical Model Checking (SMC)
  - Non-exhaustive evaluation of the model's state space
  - Through statistical simulations within certain time bound

- Statistical Timed Automata
  - Resolve non-determinism with stochastic behaviors
  - Based on Monte Carlo Simulation

Monte Carlo Simulation

Suppose you timed 20 athletes running the 50m dash and tallied the information into the four time intervals below.

You then count the tallies and make a frequency distribution.

Then convert the frequencies into percentages.

Finally, use the percentages to develop the random number intervals.

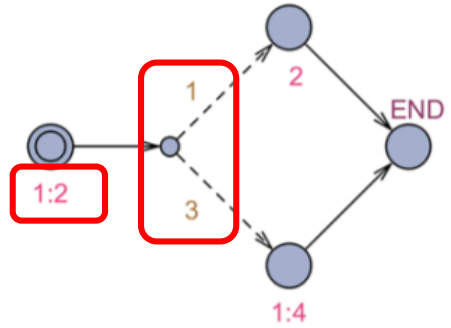
Seconds	Tallies	Frequency	%	RN Intervals
0-5.99	𝍸	4	20	01-20
6-6.99	𝍸 𝍸	10	50	21-70
7-7.99	𝍸	4	20	71-90
8 or more	||	2	10	91-100

Monte Carlo Simulation: NBA Draft

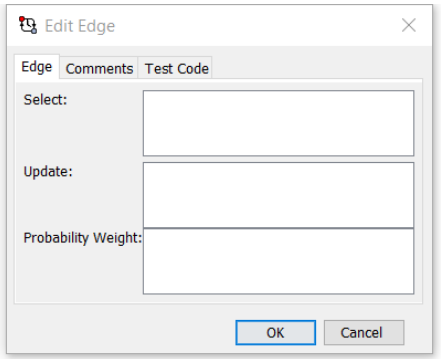
- 14 ping pong balls numbered 1 through 14 are placed in a drum.
  - C_{14}^{4}=1,001
- Prior to the Lottery, 1,000 combinations are assigned to the Lottery teams based on their order of finish during the regular season.
  - The worst team has 250 combinations (25% chance for No.1 pick)
- 4 balls are drawn from the drum with a combination
- The team that has been assigned that combination will receive the number one pick.
- The four balls are placed back in the drum and the process is repeated to determine the number two and three picks.

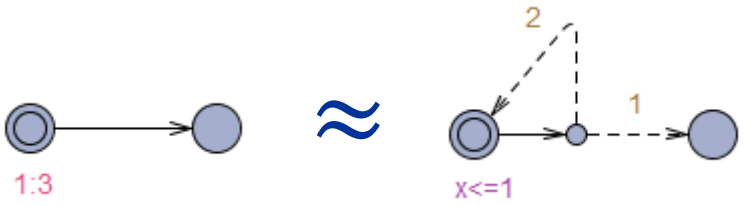
UPPAAL SMC: New Syntax

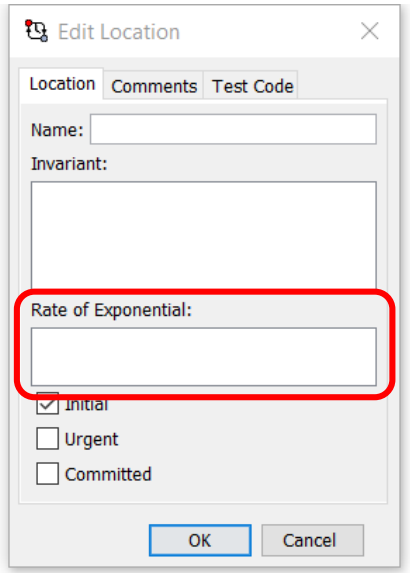
- Probabilistic transition
  - Resolves nondeterminism
  - i.e. ¼ chance going up, ¾ chance going down



- Rate of Exponential
  - "How eager you want to exit the state"

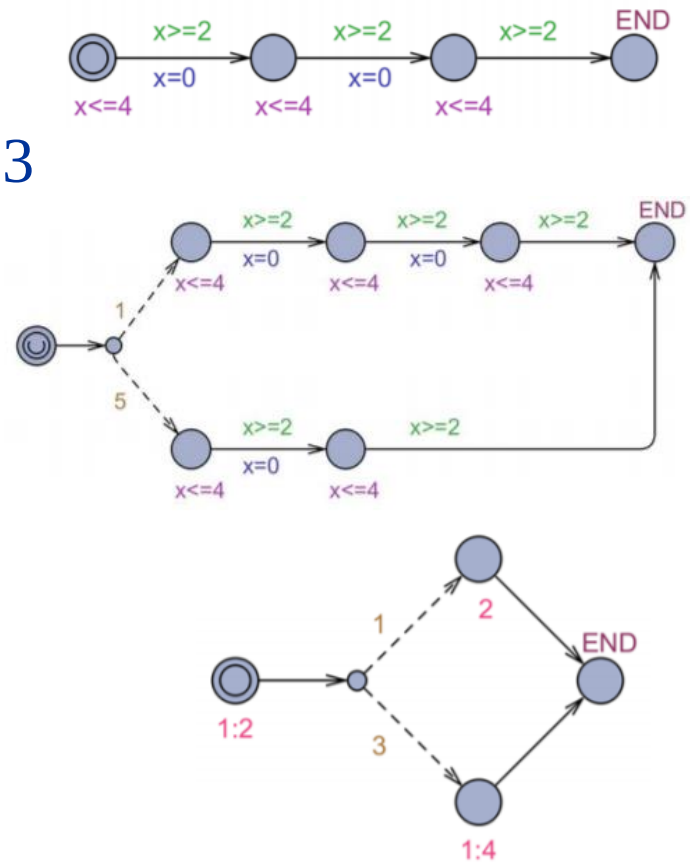


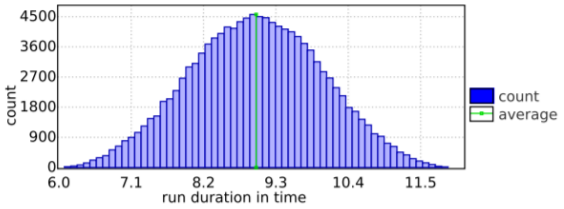




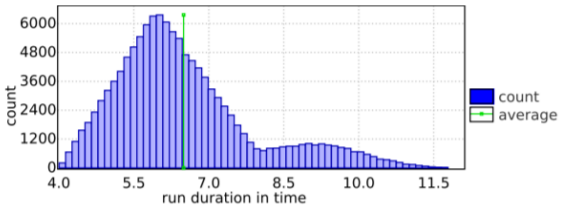
UPPAAL SMC: Semantics

- The time it takes to reach END
- Uniform distribution
  - Transition out at time 2 and time 3 are equal
- Probabilistic transition
- Exponential distribution

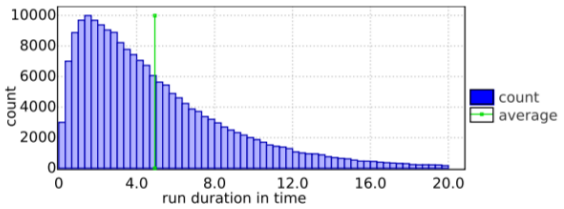




(a) A_1 arrival to END.



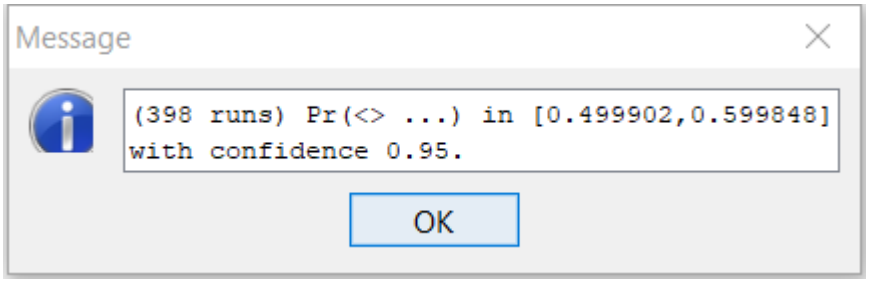
(b) A_2 arrival to END.



(c) A_3 arrival to END.

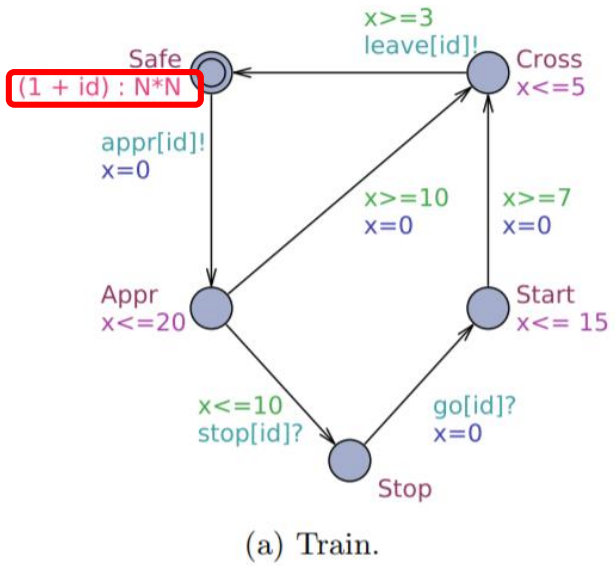
UPPAAL SMC New Queries

- Simulation
  - simulate N [<=bound] { E1,..,Ek }
- Probability Estimation
  - Pr[bound](<> psi)
- Hypothesis Testing
  - Pr[bound](psi) >= p_0
- Probability Comparison
  - Pr[bound1](psi1) >= Pr[bound2](psi2)
- Expected min/max for certain expression
  - E[bound ; N] (min/max: expr)

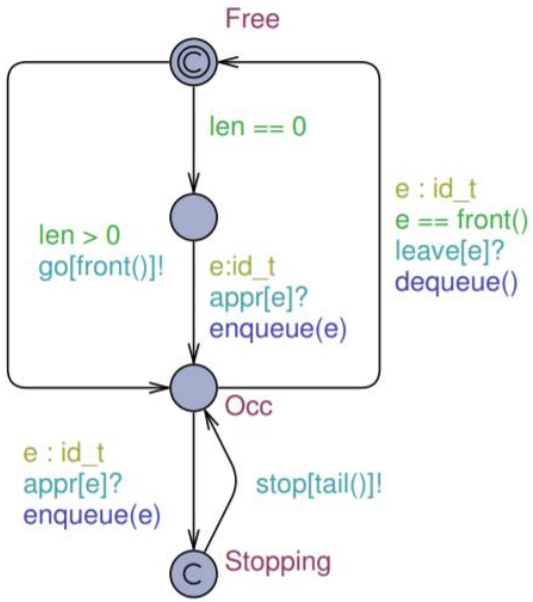


Example: Stochastic Train-Gate

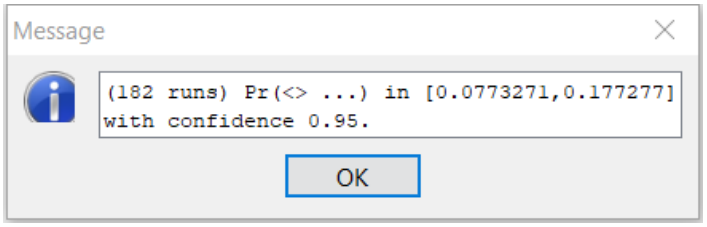
- Train with larger id is more eager to start the approach
- simulate 1 [<=300] { Train(0).Cross, Train(5).Cross, Gate.len}
- Pr[<=300](<> Gate.len < 3 and t > 20)

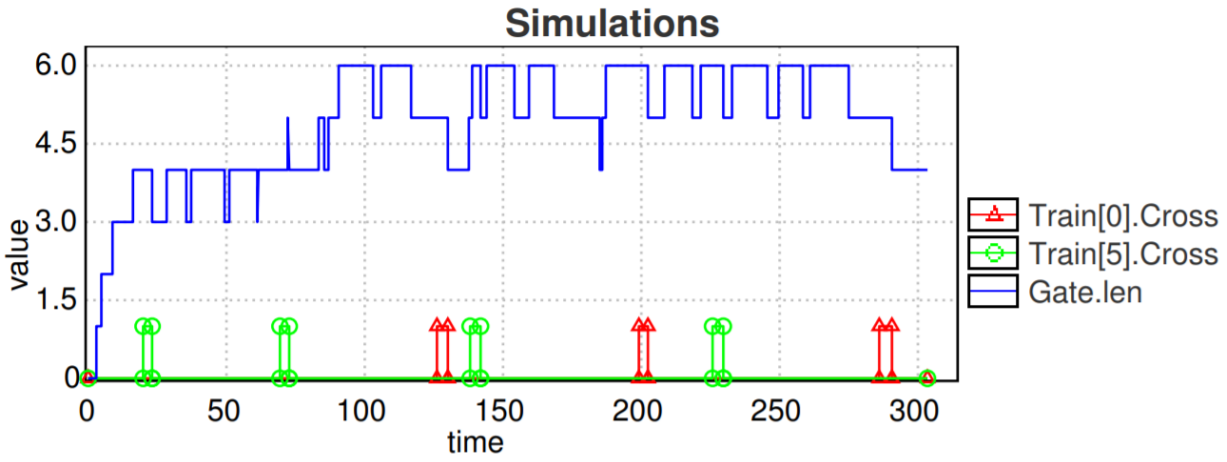


(a) Train.



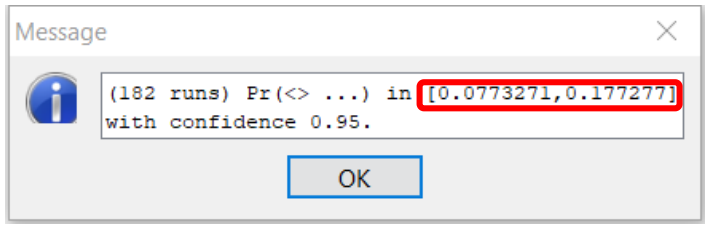
(b) Gate controller.

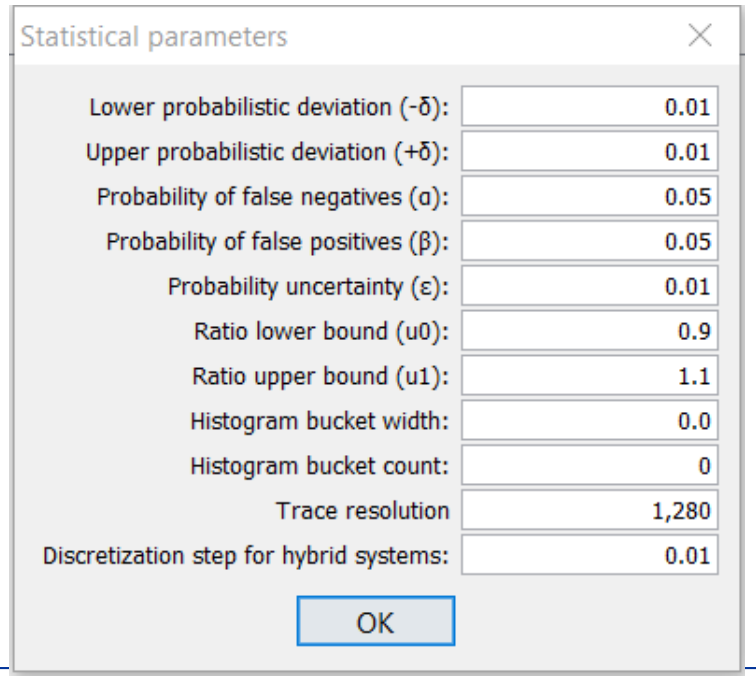




Stochastic Parameters

- δ, α, β: hypothesis testing
- ε: uncertainty for the output
  - The smaller the range, the more simulations needed
- $u0, u1$: for probability comparison

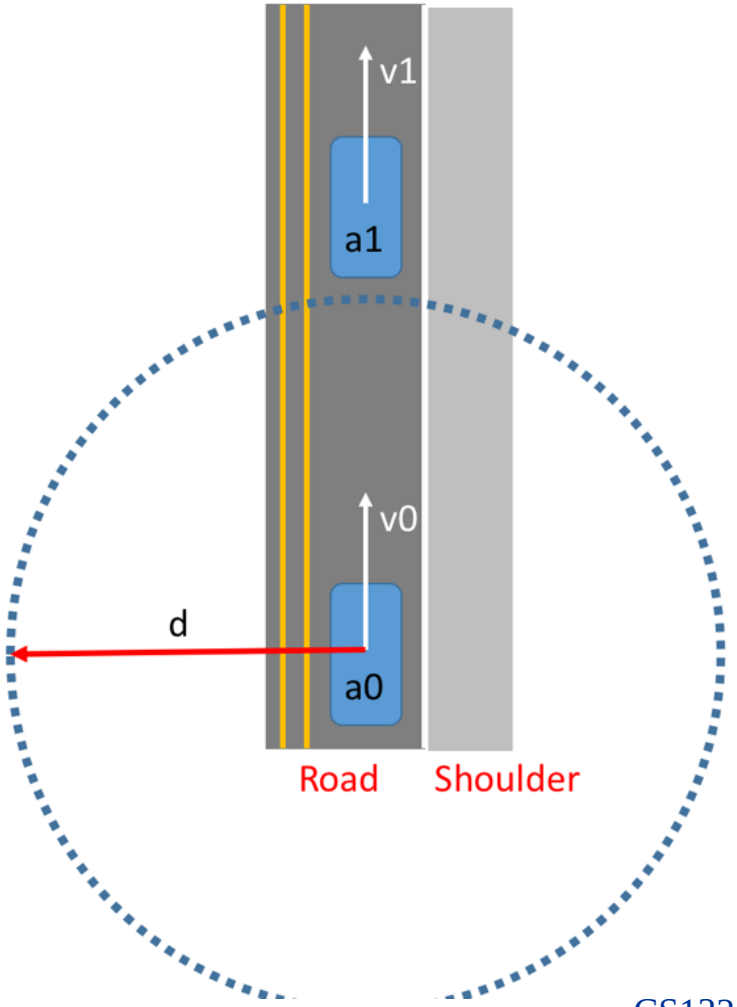




Controller Synthesis

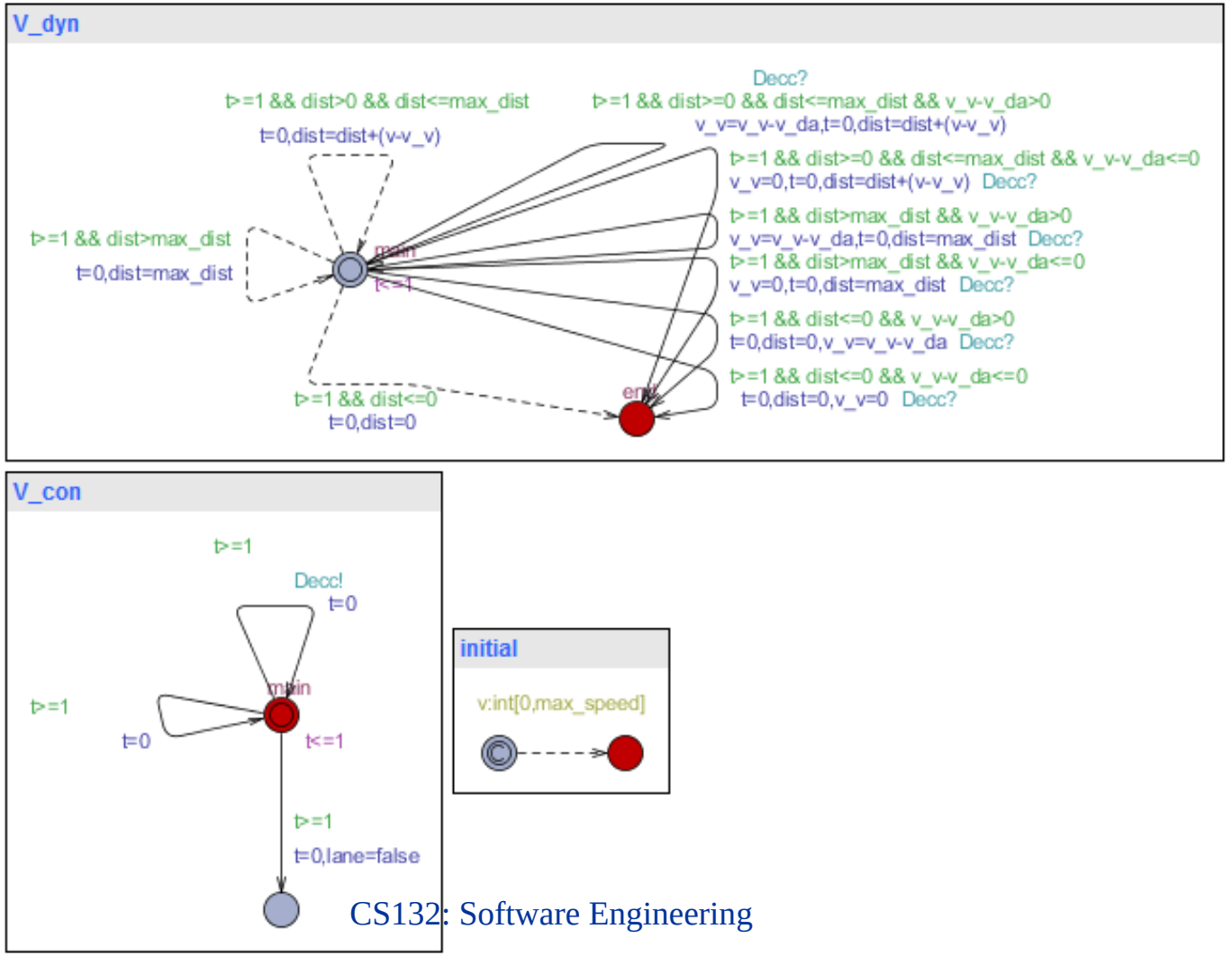
- Synthesize a controller that satisfy the requirement
- Two player game: Controller vs. Environment
- Return the winning strategy for controller

Toy Example

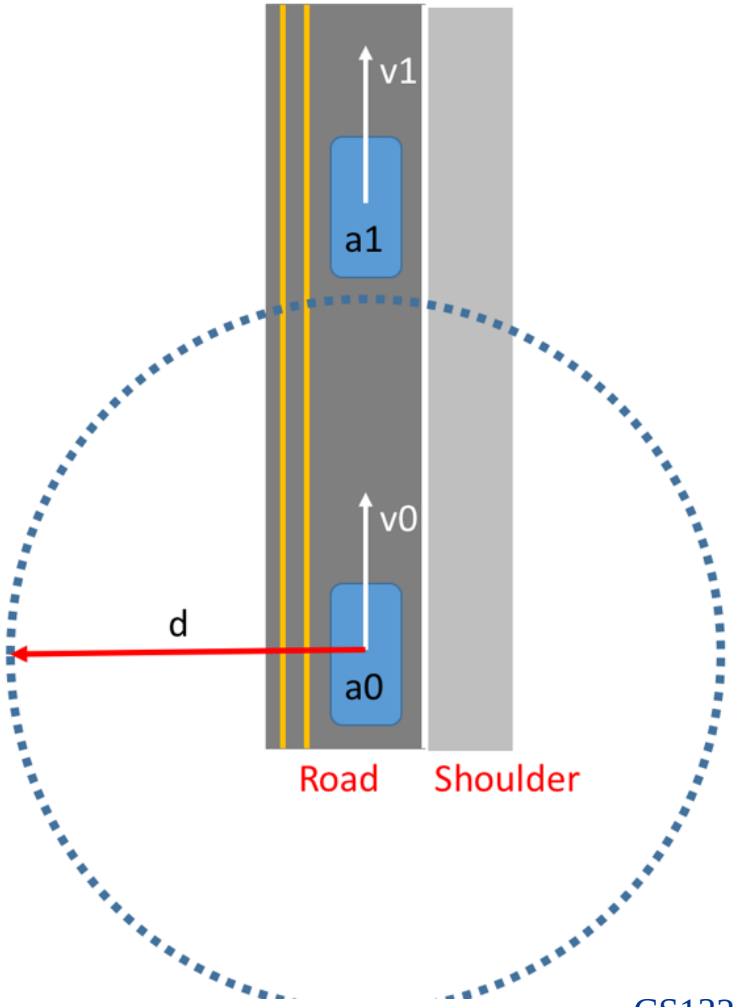


R1: No collision
R2: No driving on shoulder
R3: No hard braking

UPPAAL TIGA


V_dyn
Decc?
⊳=1 && dist>0 && dist<=max_dist
⊳=1 && dist>=0 && dist<=max_dist && v_v-v_da>0
t=0,dist=dist+(v-v_v)
v_v=v_v-v_da,t=0,dist=dist+(v-v_v)
⊳=1 && dist>=0 && dist<=max_dist && v_v-v_da<=0
v_v=0,t=0,dist=dist+(v-v_v) Decc?
⊳=1 && dist>max_dist
t=0,dist=max_dist
⊳=1 && dist>max_dist && v_v-v_da>0
v_v=v_v-v_da,t=0,dist=max_dist Decc?
⊳=1 && dist>max_dist && v_v-v_da<=0
v_v=0,t=0,dist=max_dist Decc?
main
t<=1
⊳=1 && dist<=0 && v_v-v_da>0
t=0,dist=0,v_v=v_v-v_da Decc?
⊳=1 && dist<=0 && v_v-v_da<=0
t=0,dist=0,v_v=0 Decc?
⊳=1 && dist<=0
t=0,dist=0
end
V_con
⊳=1
Decc!
t=0
main
⊳=1
t=0
t<=1
⊳=1
t=0,lane=false
initial
v:int[0,max_speed]

Naïve Solution



R1: No collision
R2: No driving on shoulder
R3: No hard braking

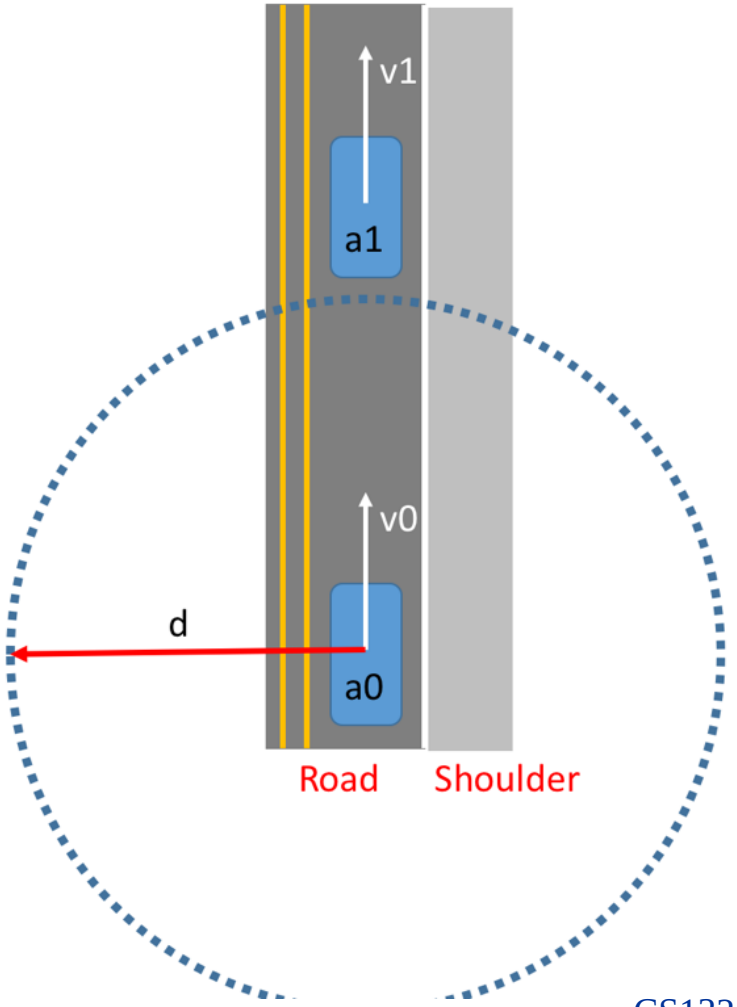
A:
v1=[0,v_max]
G: R1 R2 R3

Winning Strategy

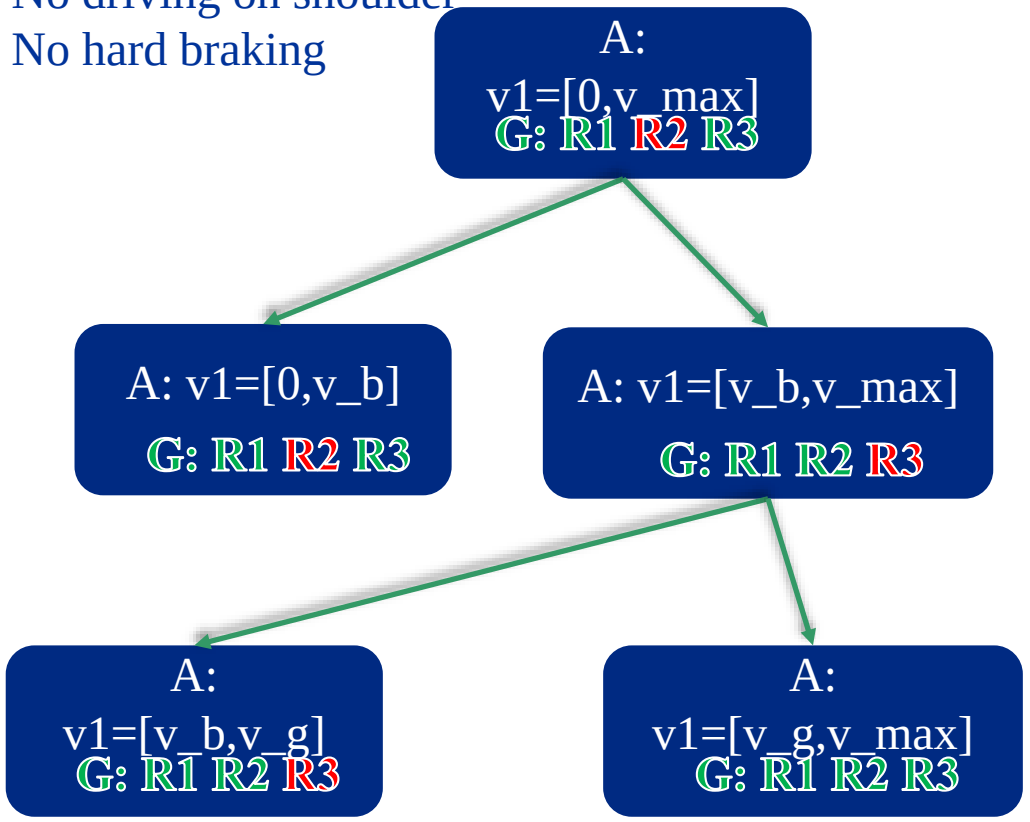
- Change lane to avoid collision

```
Strategy to avoid losing:

State: ( V_dyn.main V_con.main initial._id4 ) v_v=4 lane=1 dist=6 v=0
When you are in (V_dyn.t==1 && V_dyn.t==V_con.t && V_con.t==1), take transition
V_con.main->V_con._id0 { t >= 1, tau, t := 0, lane := 0 }

State: ( V_dyn.main V_con.main initial._id4 ) v_v=4 lane=1 dist=2 v=0
When you are in (V_dyn.t-V_con.t==-1 && V_con.t==1), take transition V_con.main-
>V_con._id0 { t >= 1, tau, t := 0, lane := 0 }
```

Model/Strategy Refinement



R1: No collision
R2: No driving on shoulder
R3: No hard braking



Reference

- Downlaod
  - www.uppaal.org
- Tutorials
  - On the same webpage
  - Recommended:
    - UPPAAL 4.0: Small Tutorial.
    - Uppaal SMC Tutorial