SELF-DRIVING CARS
PLANNING

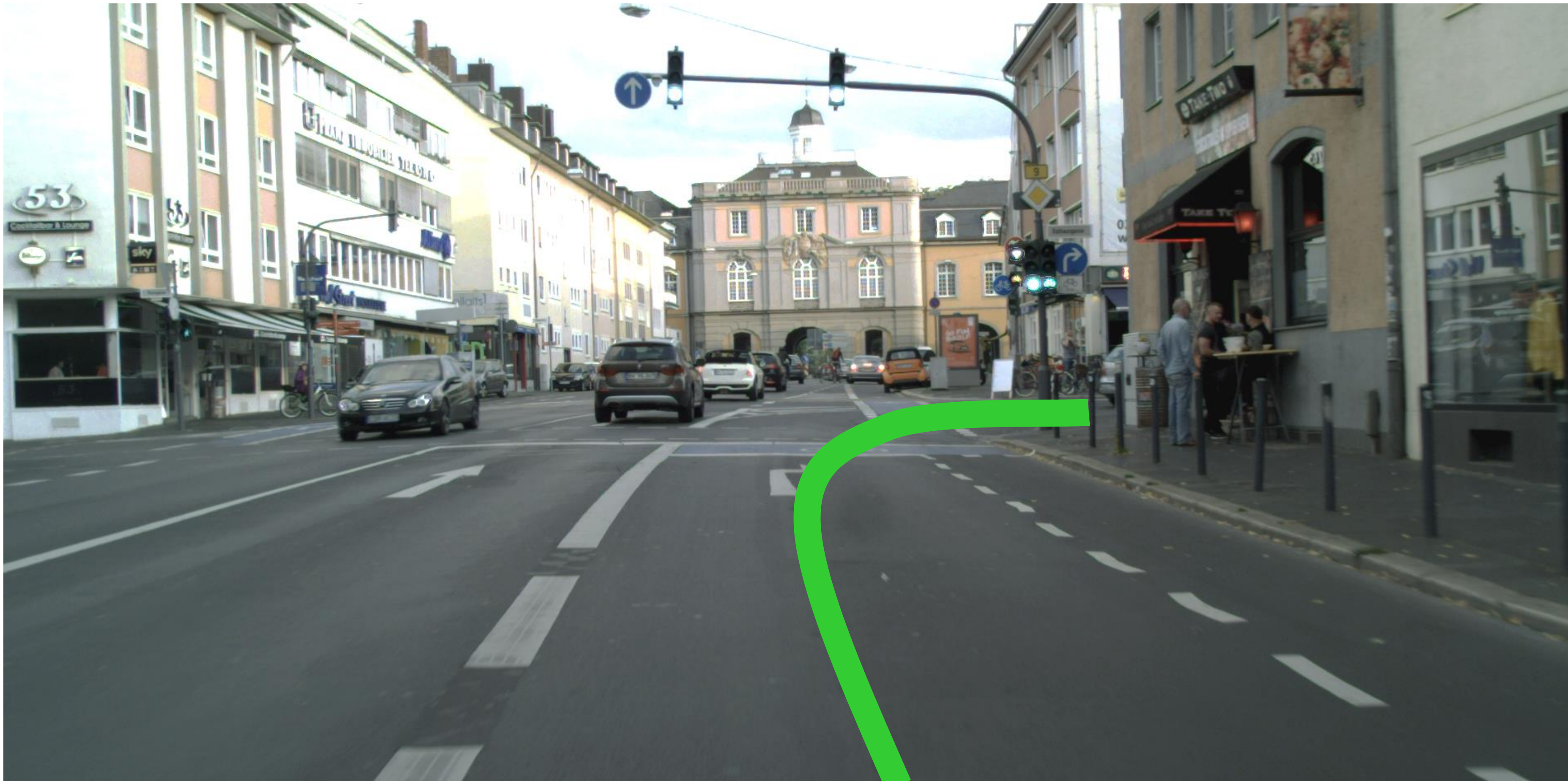
Planning for Self-Driving Cars

Benedikt Mersch

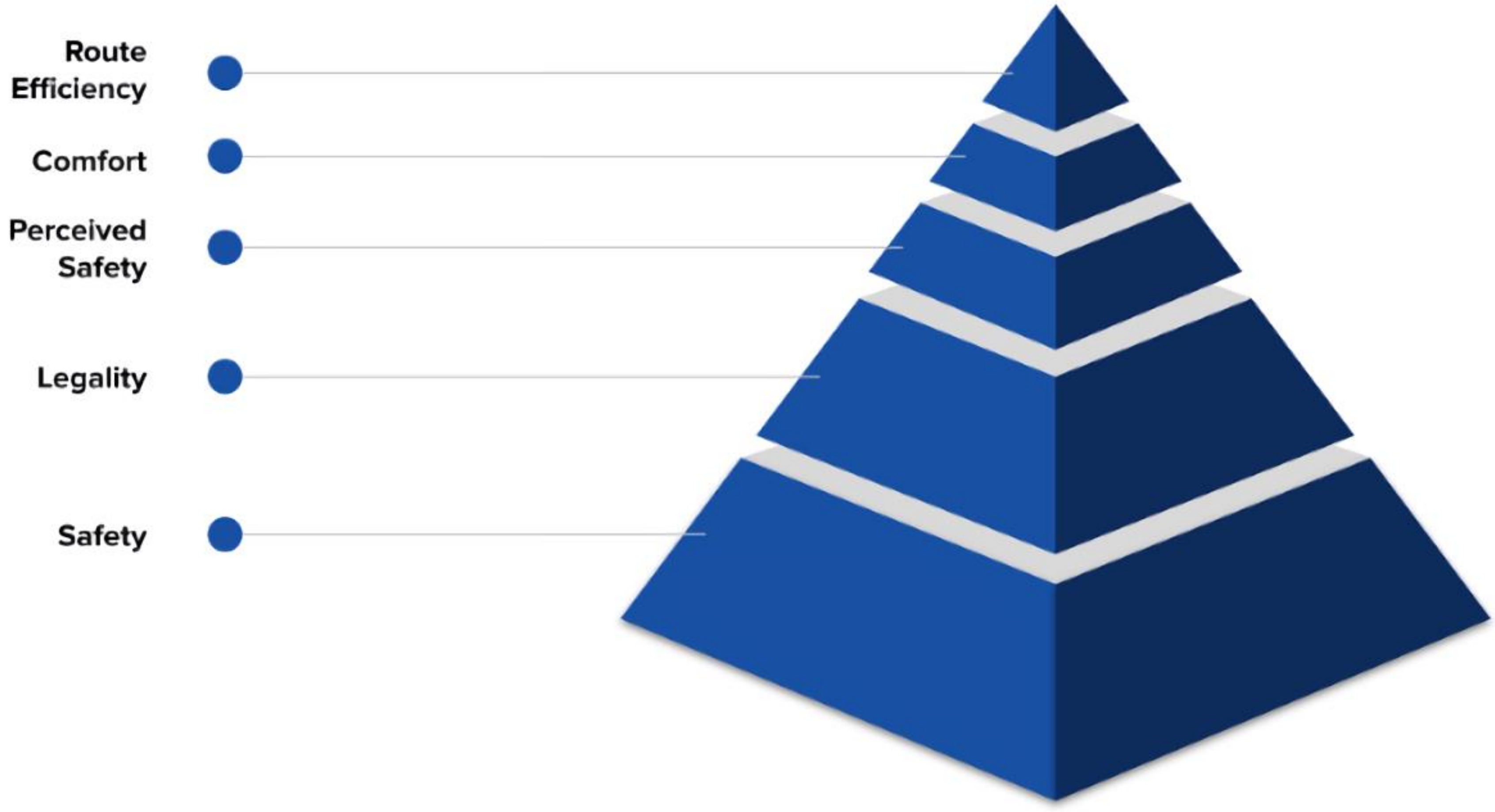
Part of the Course: Techniques for Self-Driving Cars by
C. Stachniss, J. Behley, N. Chebrolu, B. Mersch, I. Bogoslavskyi

Self-Driving Car Scenario

How to Get a Trajectory?



What Is Most Important?



[Courtesy of Lyft]

What is the Mission?

- Get from A to B on a path which is
  - safe
  - physically feasible
  - smooth
  - fast
  - short
  - …

- Task gets more complex for a larger planning horizon

[Courtesy of Waymo]

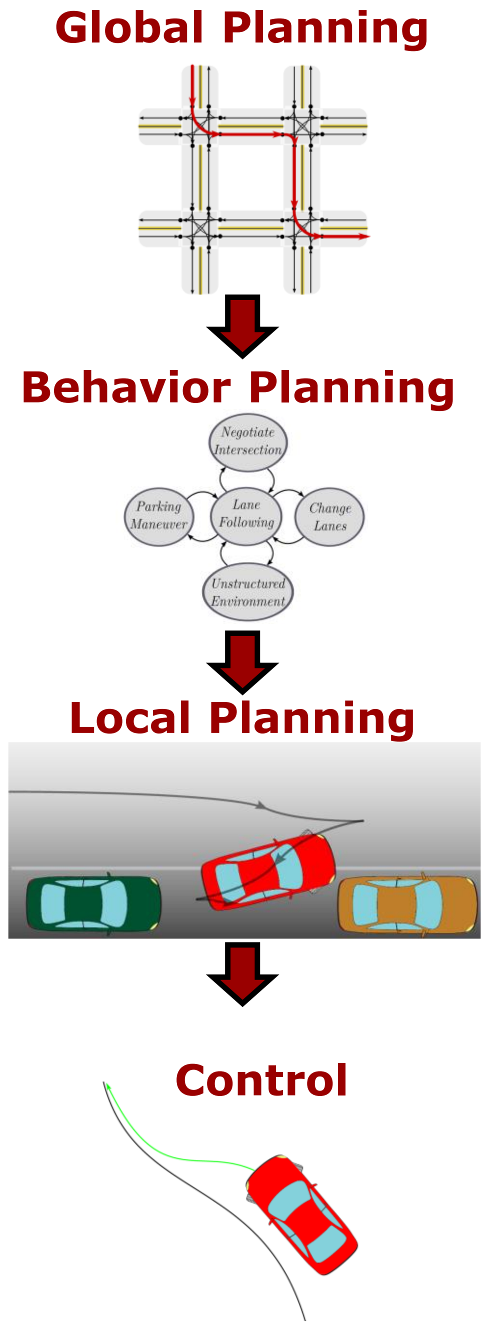
Challenges

- React according to **events** like
  - speed limits
  - red lights
  - jaywalking
  - slower cars
- Not all information is available in advance ➡ **online** planning
- Not all information is needed for each decision ➡ **hierarchical** planning

Hierarchical Planning

Hierarchical Planning

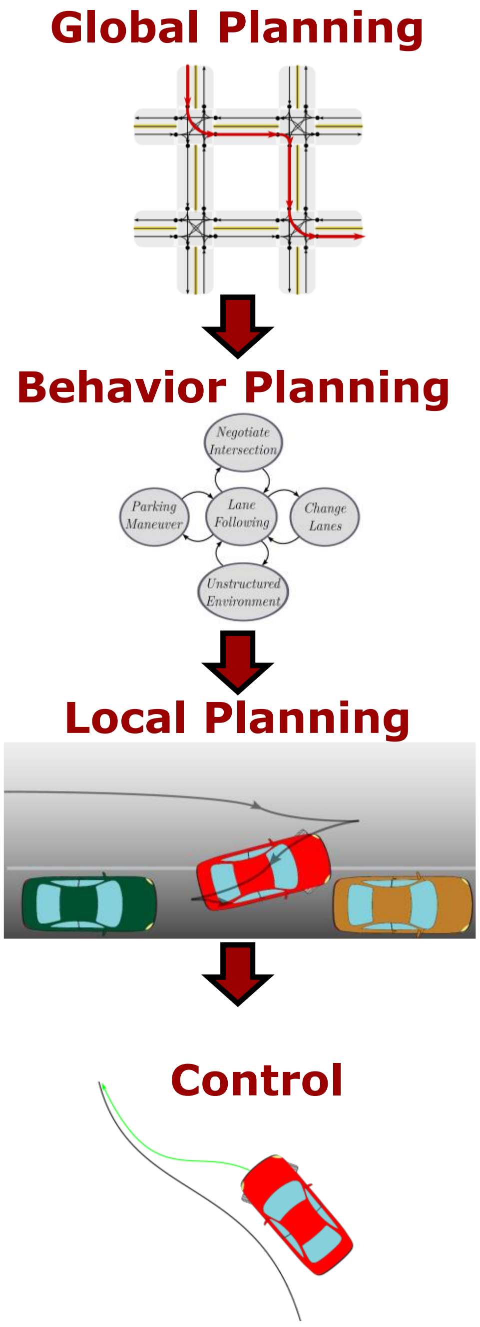
- Divide driving task into **subproblems**
- Hierarchy defines levels of abstraction and information access



[Paden et al.]

Hierarchical Planning

- Divide driving task into **subproblems**
- Hierarchy defines levels of abstraction and information access



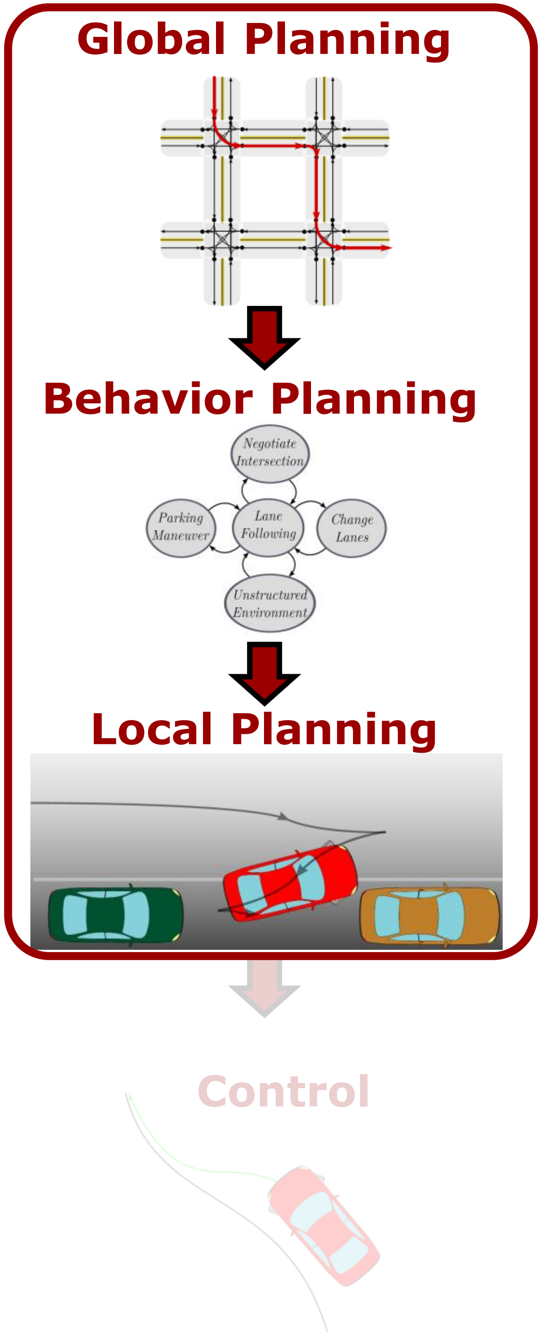
Recap: Low-level controller

- Assume a given reference path and current vehicle state
- Output control commands to follow path

[Paden et al.]

Hierarchical Planning

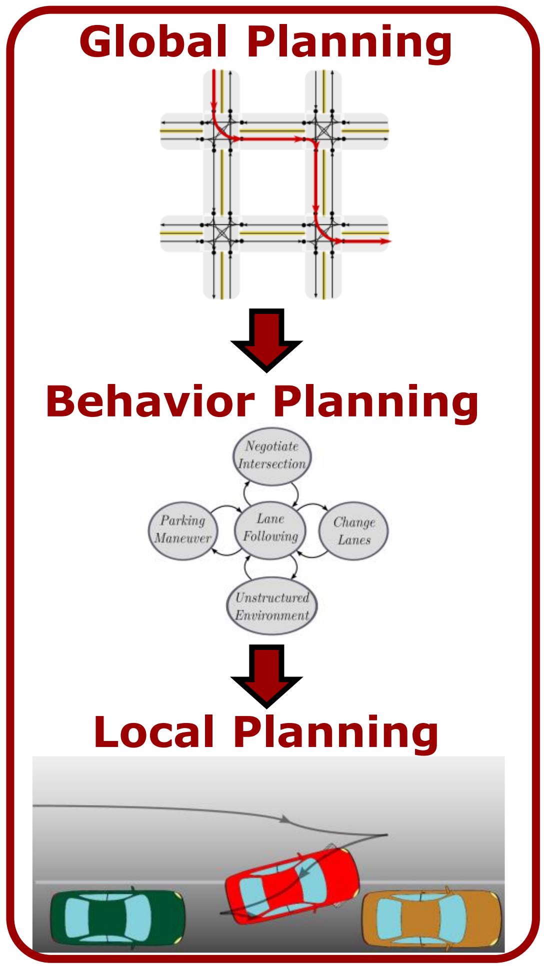
- Divide driving task into **subproblems**
- Hierarchy defines levels of abstraction and information access



[Paden et al.]

Hierarchy Properties

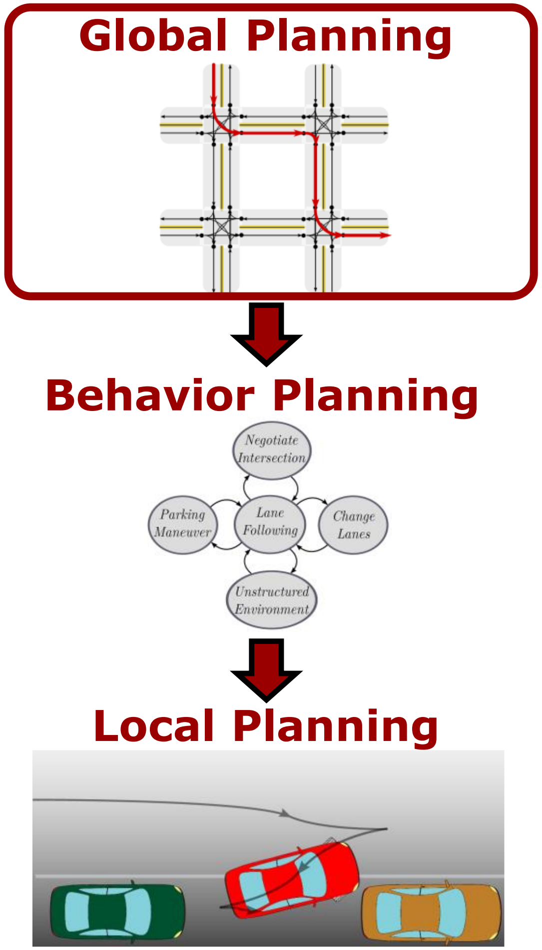
High abstraction → Low abstraction



Low frequency → High frequency

[Paden et al.]

Global Planning

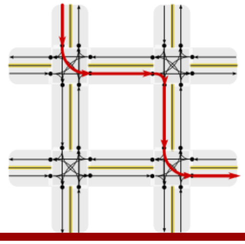


[Paden et al.]

Global Planning

User defined destination and
road network

Global Planning



Sequence of waypoints

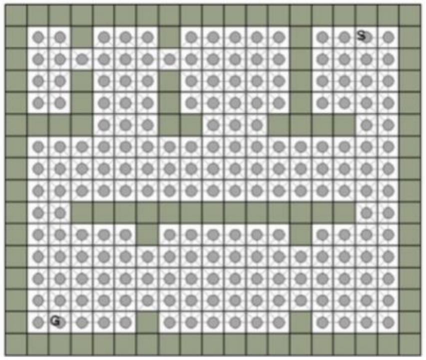
[Paden et al.]

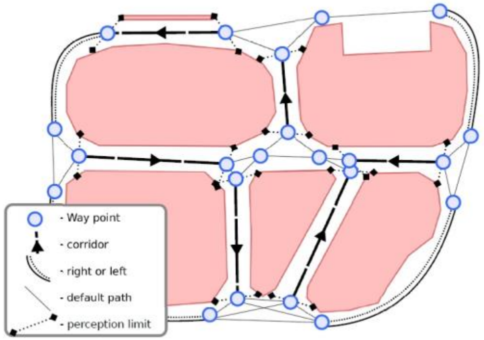
Global Planning

- Similar functionality as a satnav
- **Assumes** known poses and a map
- **Goal**: Find waypoints through the road network with minimum cost
- Can be done **offline** in advance
- **Replan** if new information is available (blocked road, traffic jam, speed limits, ...)

From High Definition Maps to Graphs

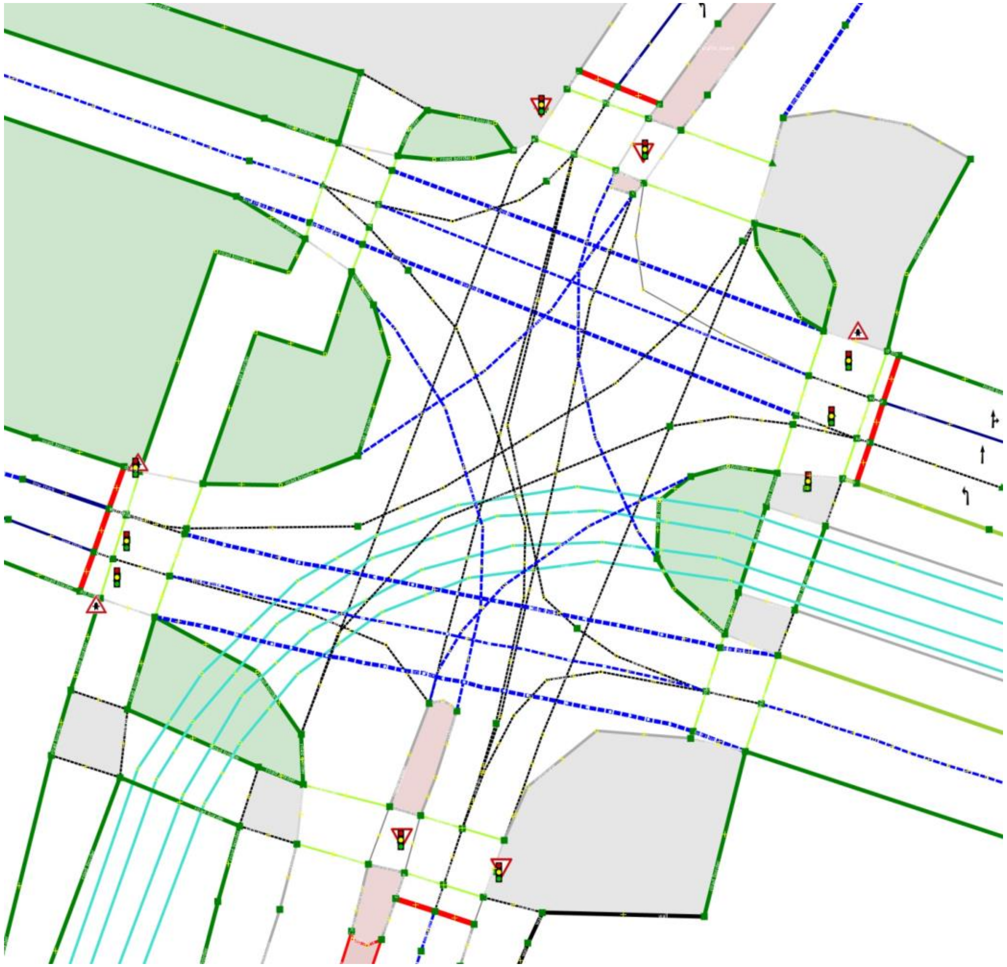
- **Often in robotics**: Discretize world (state and actions) resulting in a grid
- **Here**: Use a topological road map
- **HD maps** contain information about intersections, roundabouts, lanes, ...





[Lozenguez et al.]

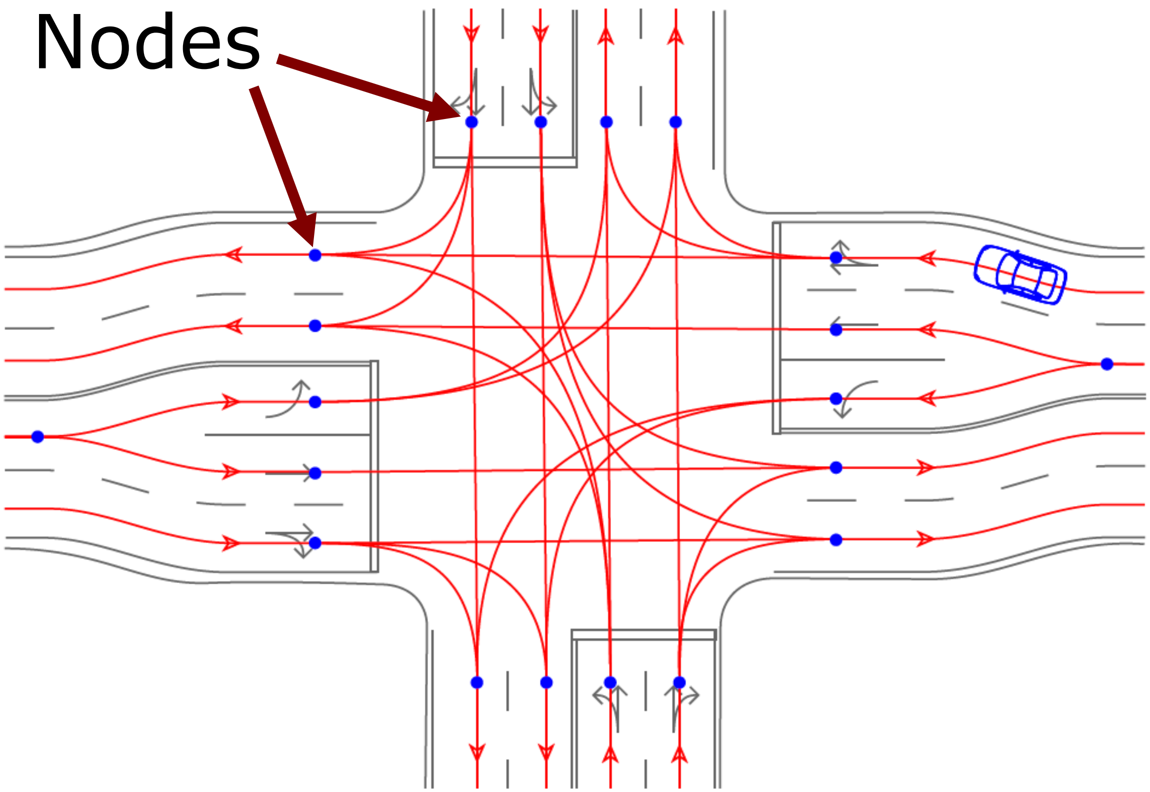
High Definition Map - Example



Generate a **graph** from the map

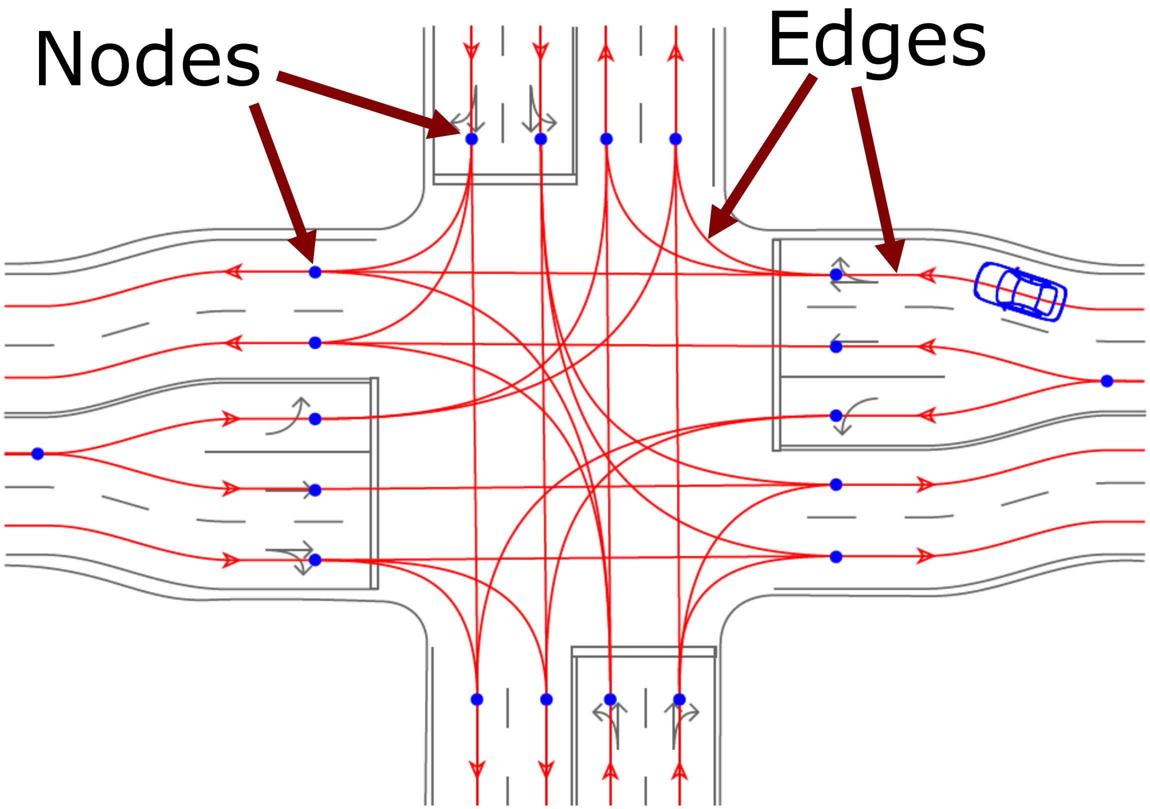
[Poggenhans et al.]

Graph - Example



Nodes represent start and end of road segments

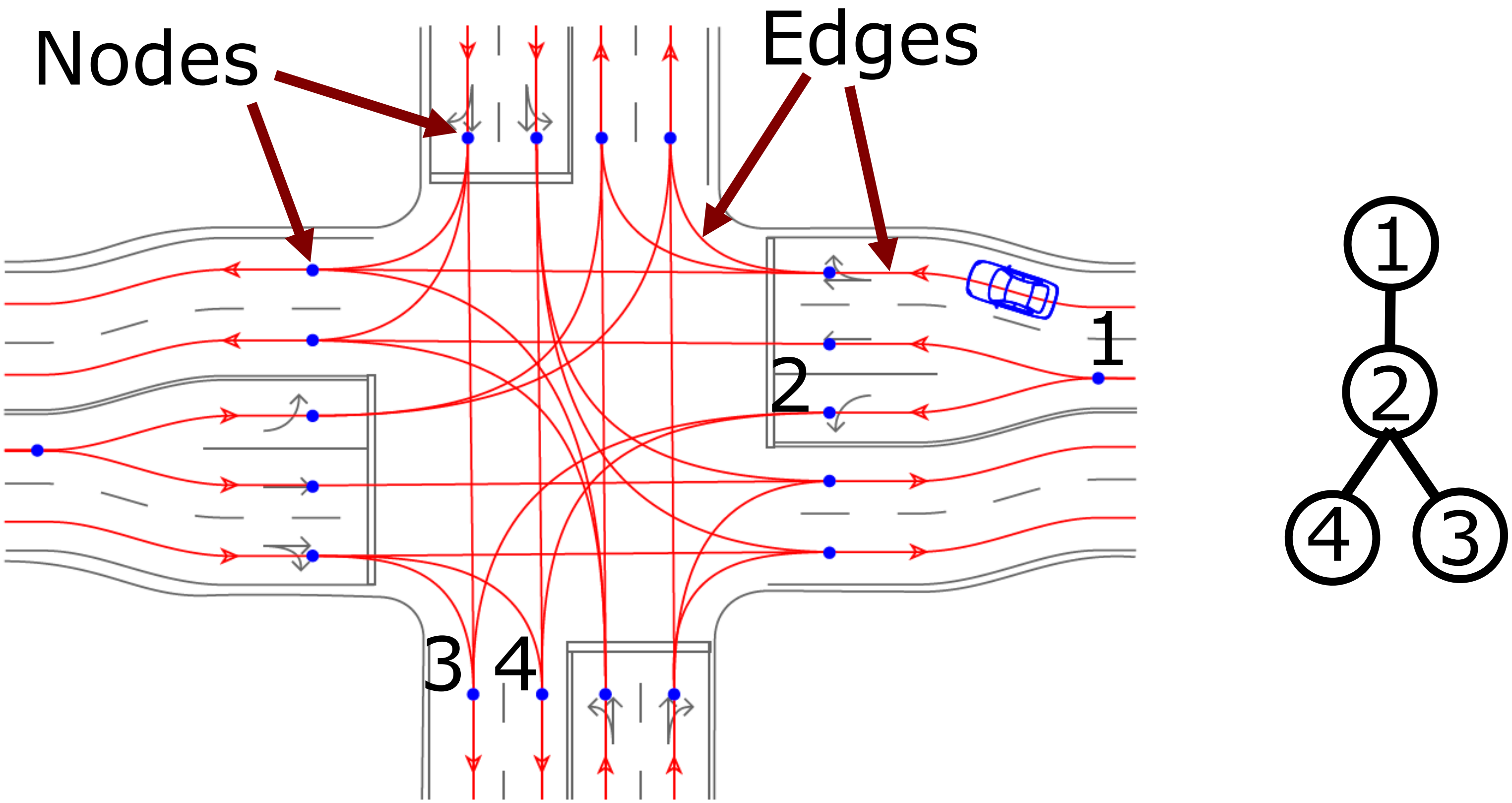
Graph - Example



Edges store and update road information like type, length, speed limits

[Paden et al.]

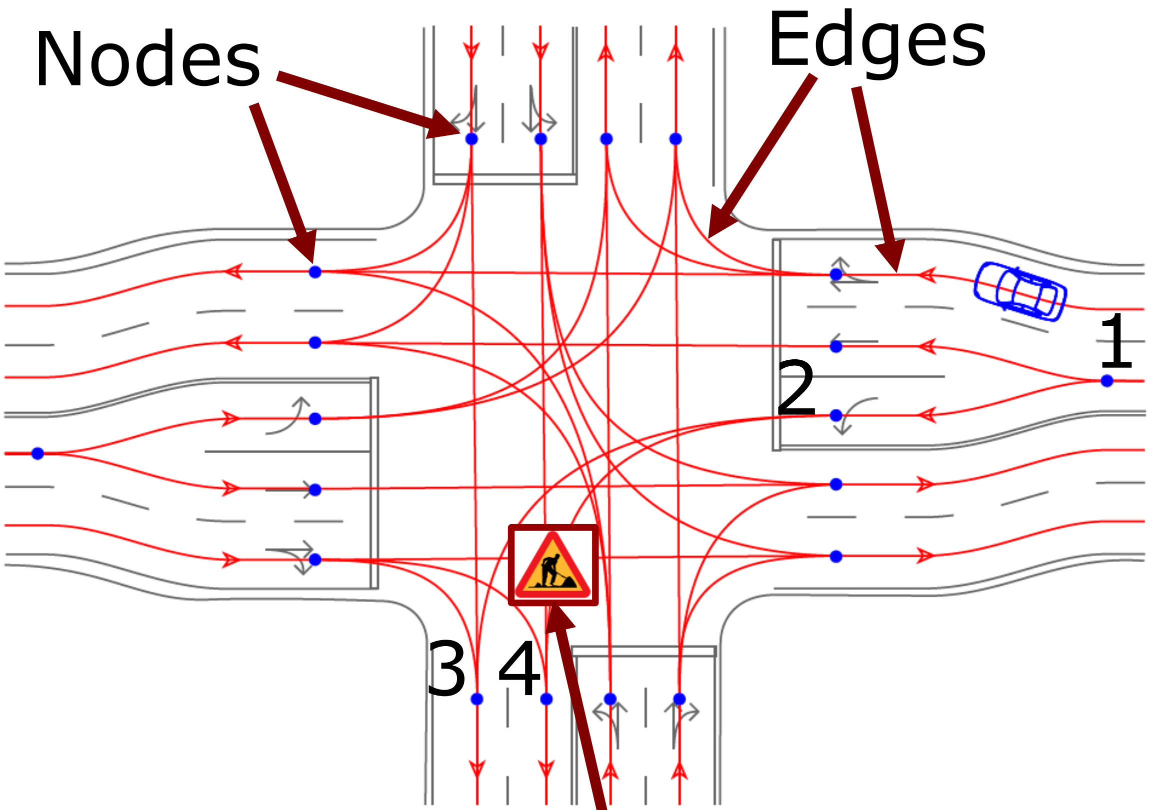
Graph - Example

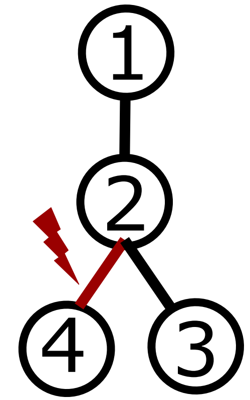


Edges store and update road information like type, length, speed limits

[Paden et al.]

Graph - Example

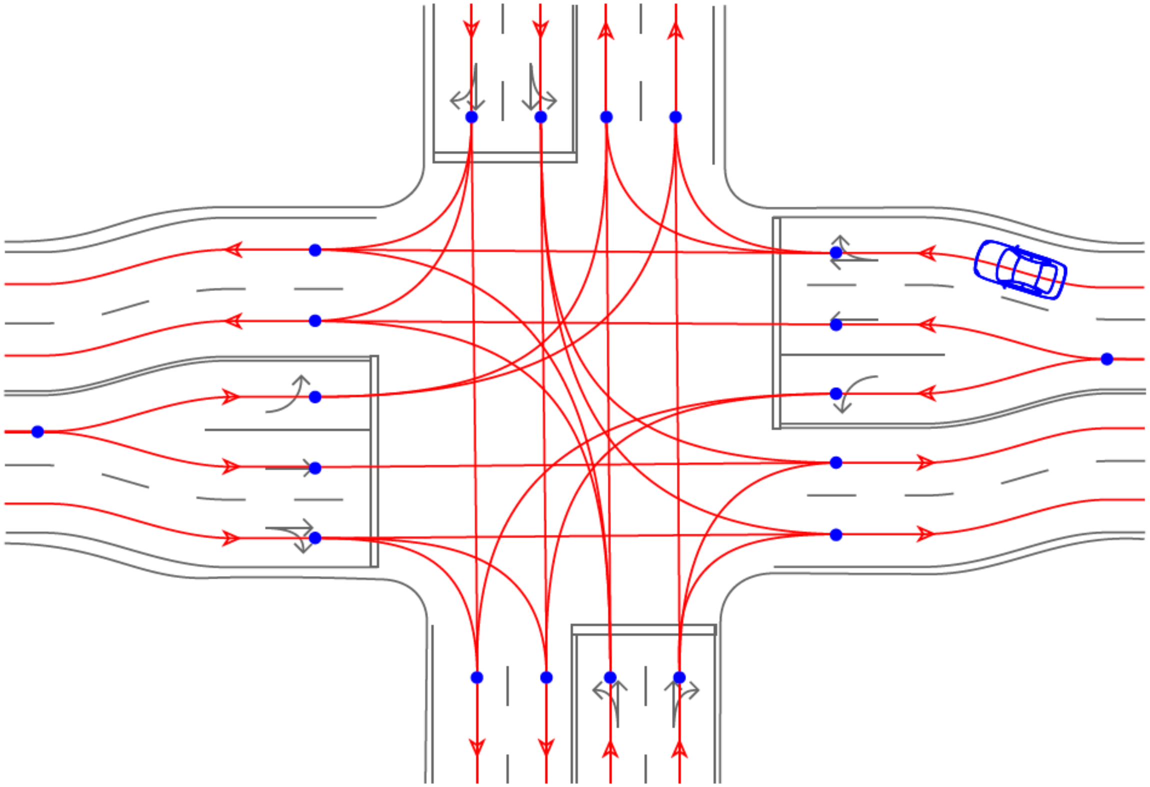




Edges store and update road information like type, length, speed limits

[Paden et al.]

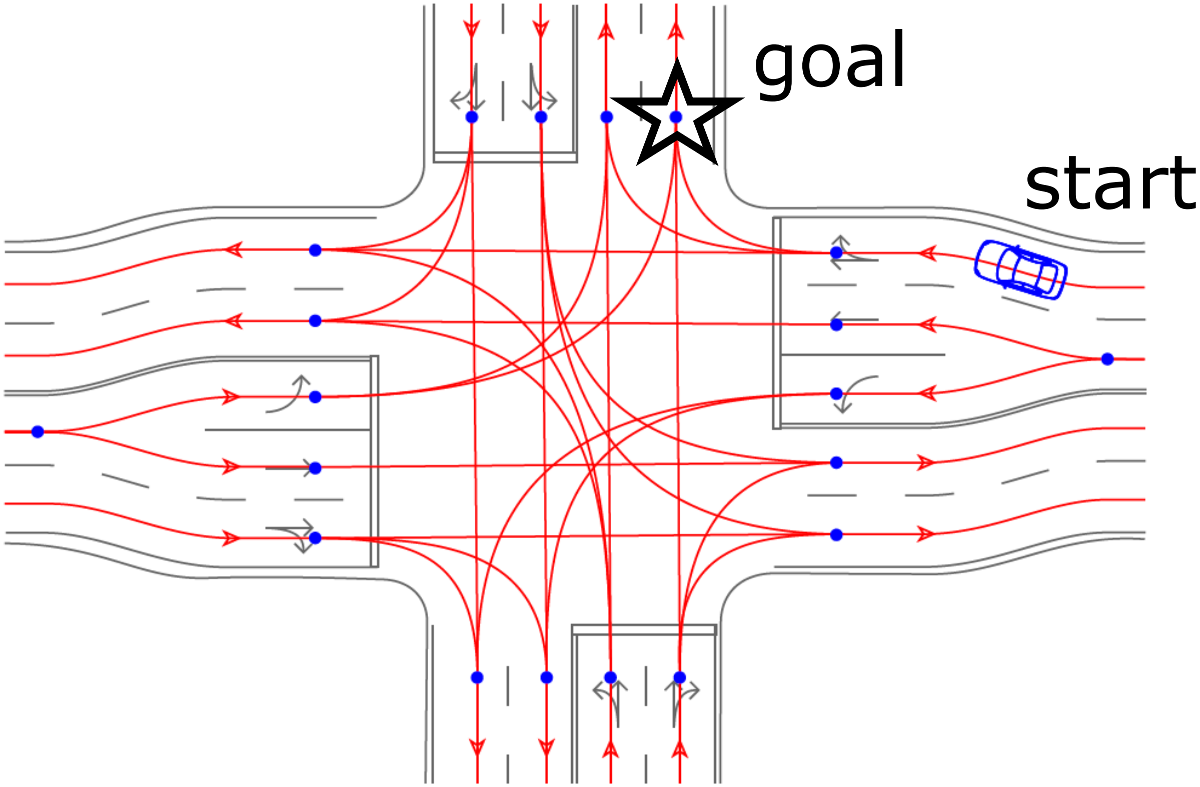
Graph - Example



Next step: Search for the best path from start to goal

[Paden et al.]

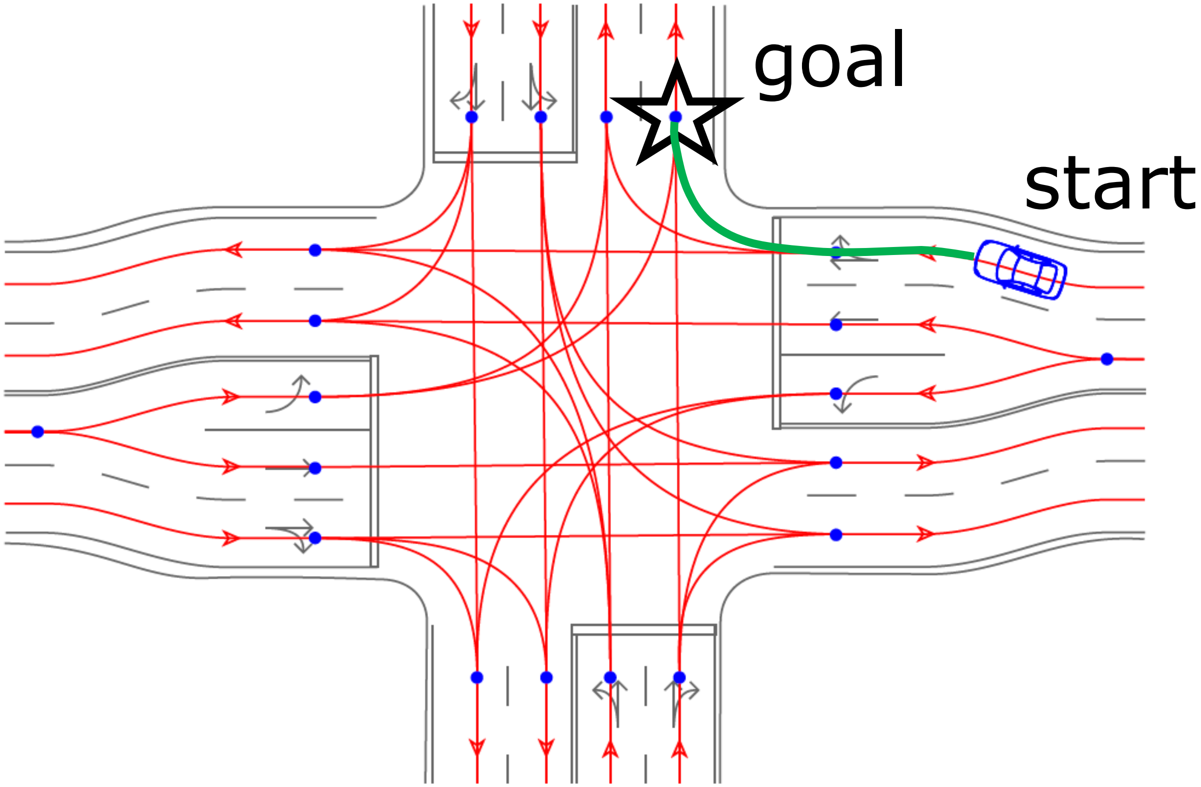
Graph - Example



Next step: Search for the best path from start to goal

[Paden et al.]

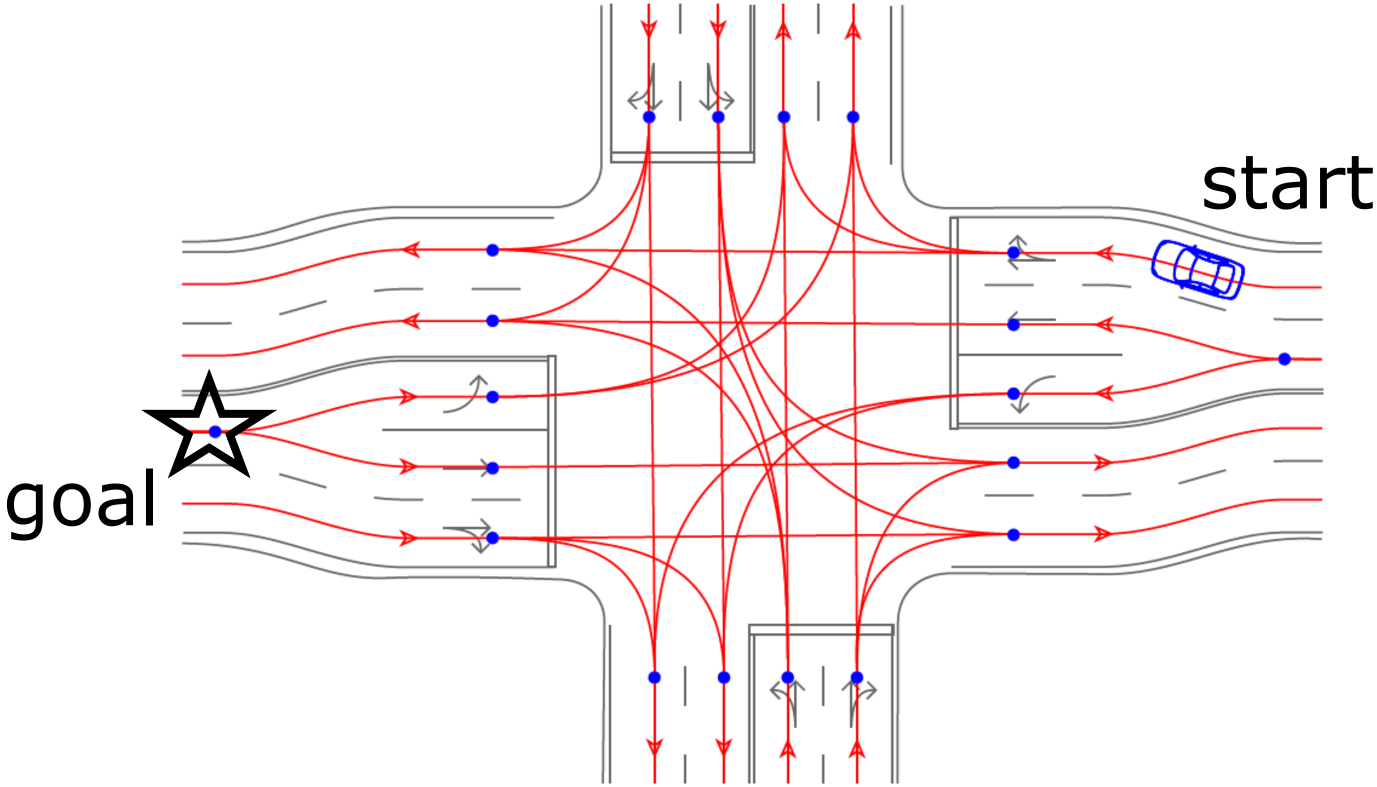
Graph - Example



Next step: Search for the best path from start to goal

[Paden et al.]

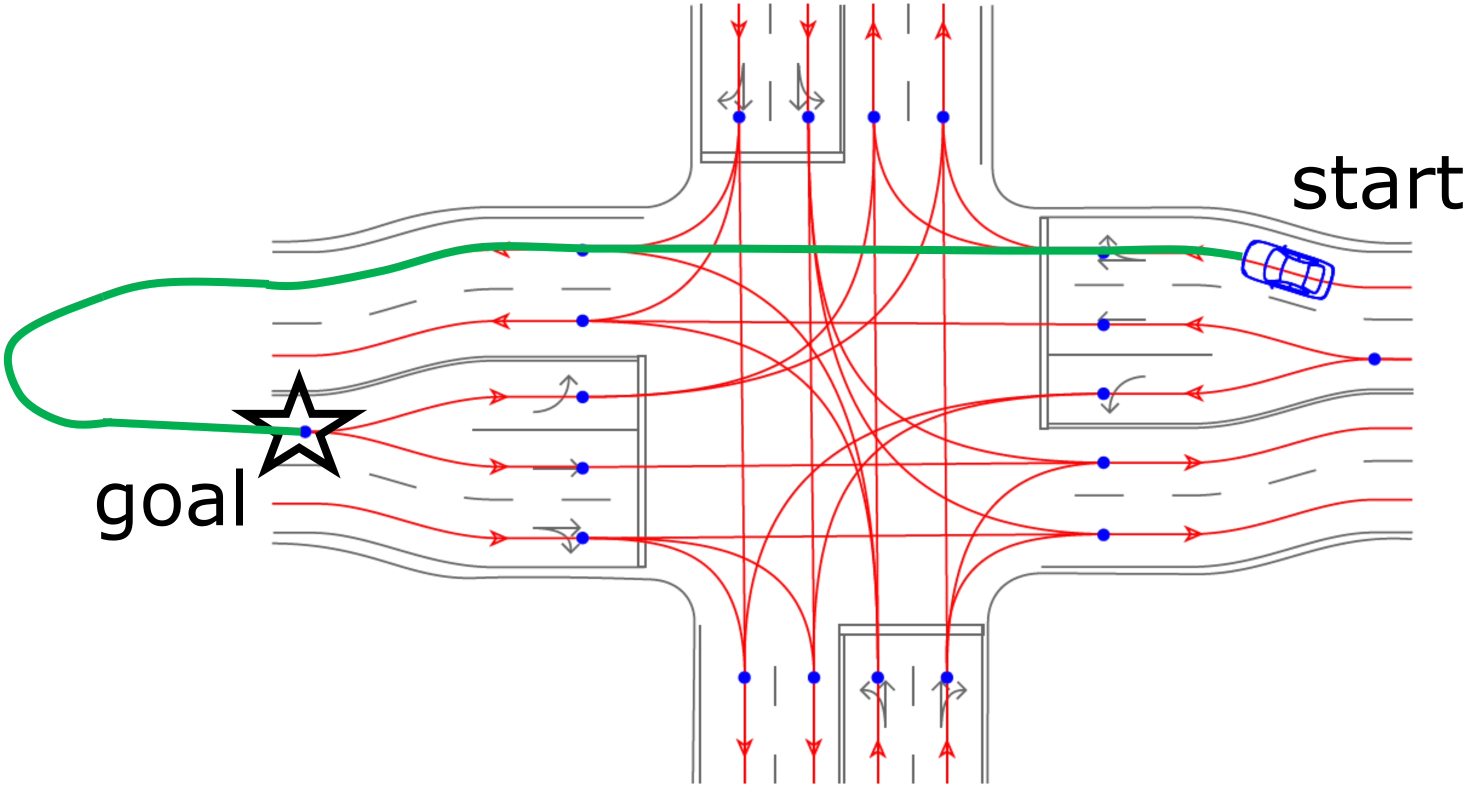
Graph - Example



Next step: Search for the best path from start to goal

[Paden et al.]

Graph - Example

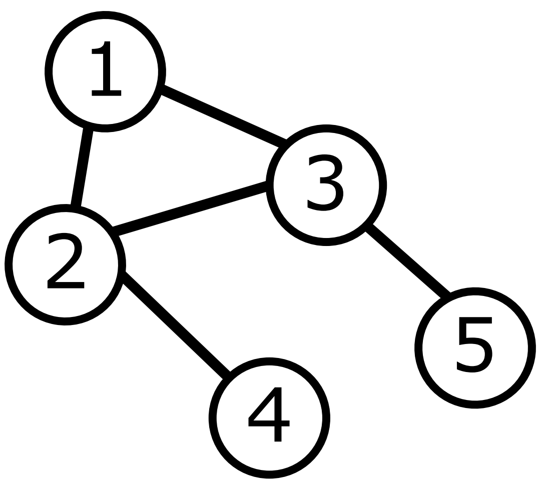


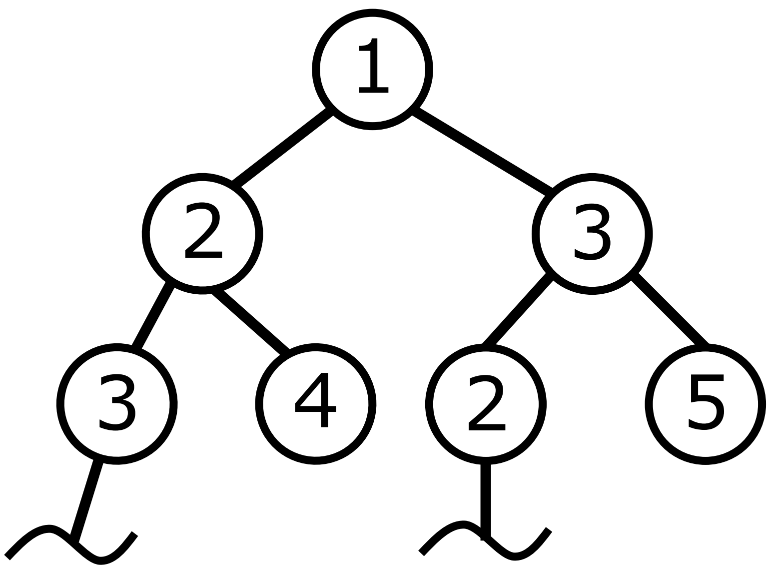
Next step: Search for the best path from start to goal

Tree Search

From Graph to Tree

- A path on the graph can be represented in a search tree
- Top node is the start pose
- At each level an action is taken which results in a new pose



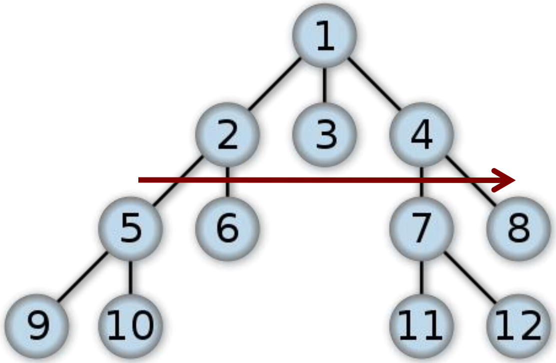


Tree Search

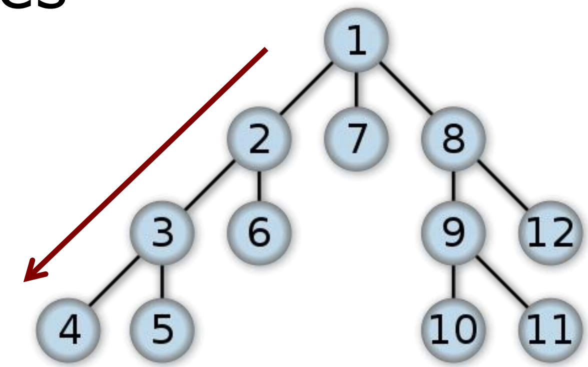
- Find the **best path** to the **goal node**
- How can we search the nodes in an **efficient** way?
- Search algorithms differ in
  - Completeness
  - Optimality
  - Time consumption
  - Memory consumption
- Uninformed or informed search

Uninformed Search

- **Breadth-first (BFS)**
  - Complete
  - Optimal if action costs equal
  - Time and space: $O(b^d)$



- **Depth-first (DFS)**
  - Not complete in infinite spaces
  - Not optimal
  - Time: $O(b^m)$
  - Space: $O(bm)$ (we can forget explored subtrees)

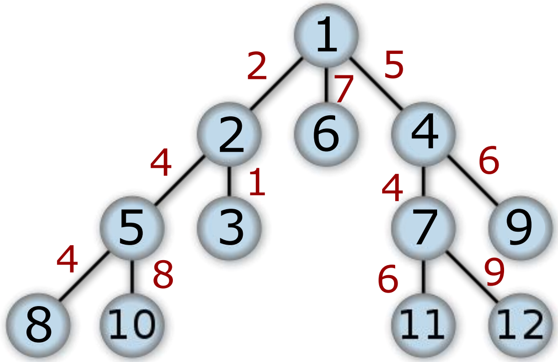


(*b:* branching factor, *d:* goal depth, *m:* max. tree depth)

Cost Sensitive Search

Uniform Cost Search (UCS)

- Nodes are treated uniformly but expansion has a **cost function**
- At each step expand the node with minimum **accumulated cost (g)**
- Complete & optimal
- Time: $O(b^{C/e})$
- Space: $O(b^{C/e})$



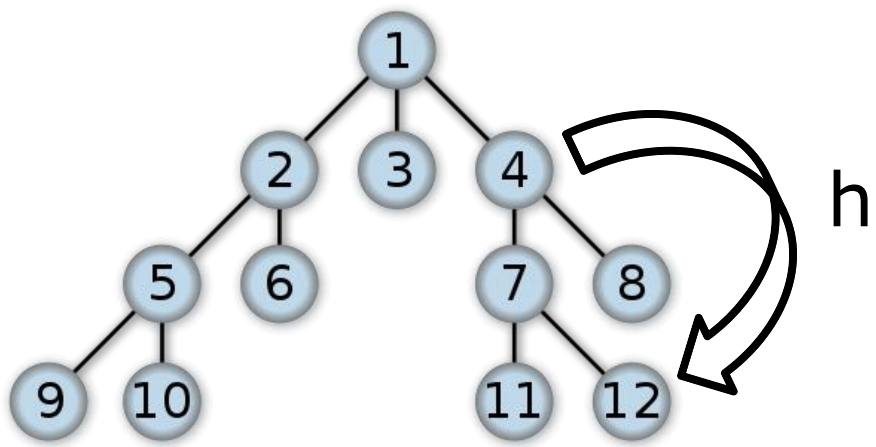
(b: branching factor, C: solution cost, e: minimum edge cost)

Informed Search

- DFS, BFS, UCS do not use any **prior information** about goal
- Informed search techniques exploit additional knowledge about the goal
- A **heuristic** (=cost estimate) is used to guide the search
- Speeds up the search

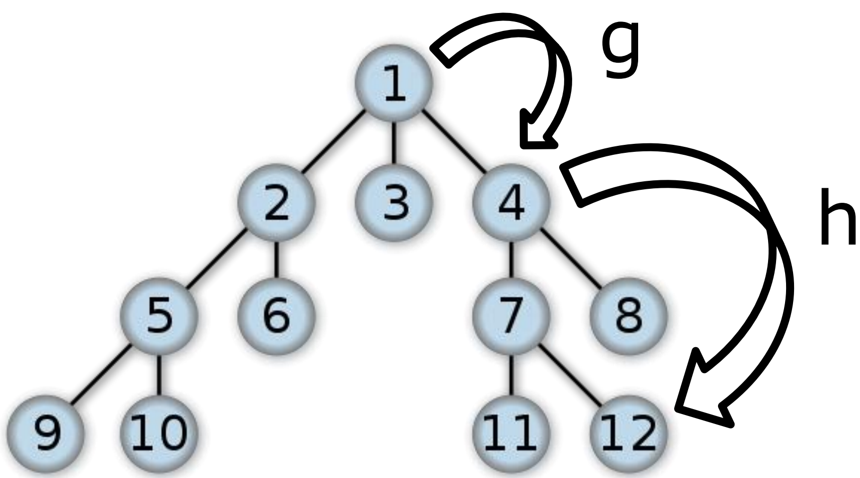
Greedy Search

- Greedily explore the node that **seems** closest to the goal with a **heuristic h**
- Ignores so far accumulated path cost
- At each step expand the node with minimum **estimated** cost to the goal



A* Search

- A* combines UCS and greedy search
- Jointly considers the **accumulated path cost** (g) and a **heuristic** (h)



A*: Minimize Accumulated and Estimated Cost

- *g(n)*: actual accumulated cost from the initial state to *n*.
- *h(n)*: estimated cost from the state *n* to the goal.
- $f(n) = g(n) + h(n)$, the estimated cost of the cheapest solution through *n*.

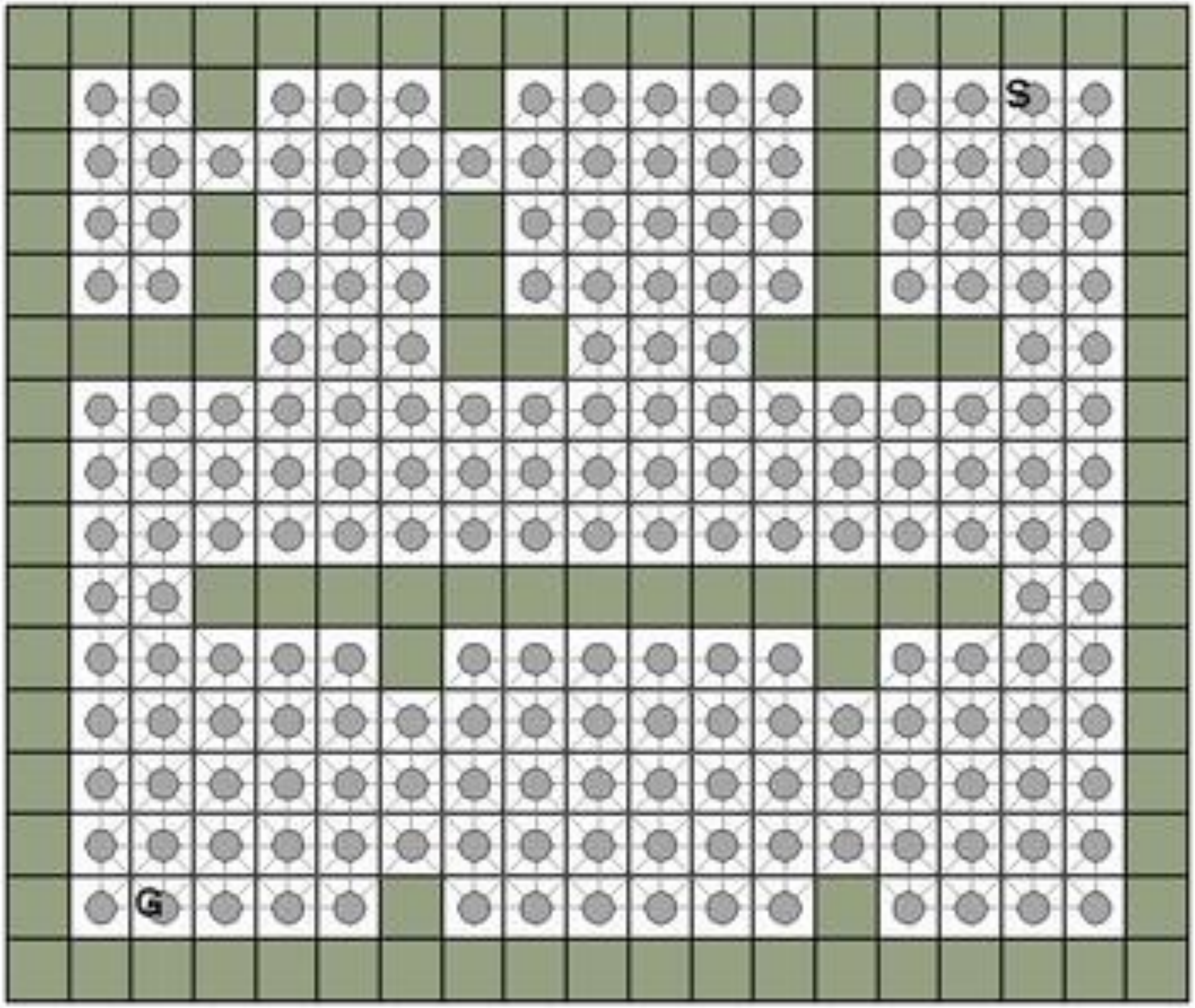
Heuristic for A*

- A* uses $f(n) = g(n) + h(n)$
- Let $h^*(n)$ be the actual cost of the optimal path from n to the next goal.
- h is admissible if for all n *holds*:

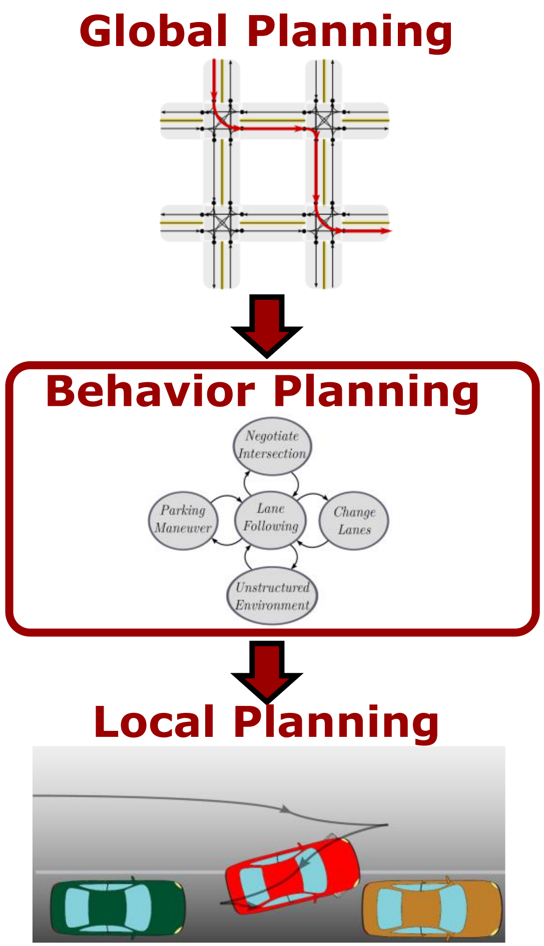
$$h(n) \leq h^*(n)$$

- We require for A* that h is admissible
- E.g., the straight-line distance is admissible in the Euclidean space.

A* Search Example



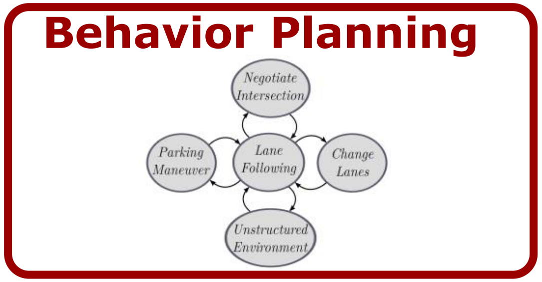
Behavior Planning



[Paden et al.]

Behavior Planning

Sequence of waypoints and
perception of neighborhood



Maneuver specification
(Stop, drive straight, overtake,...)

[Paden et al.]

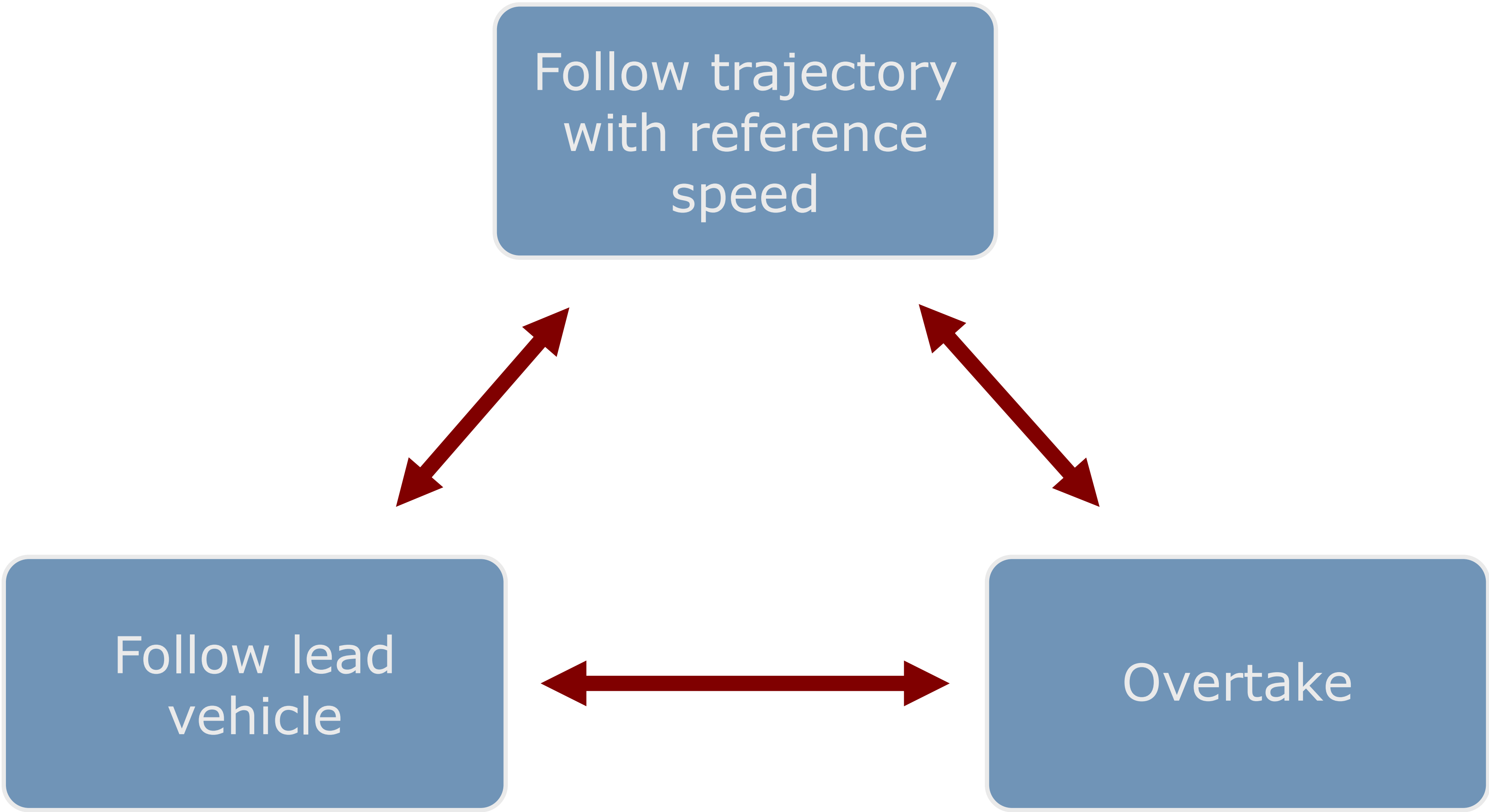
Behavior Planning

- Following a global plan requires different maneuvers/behaviors
  - Interactions with other traffic participants
  - Stop signs, emergency stops, overtaking manuevers
- Cover all possible situations by a state machine

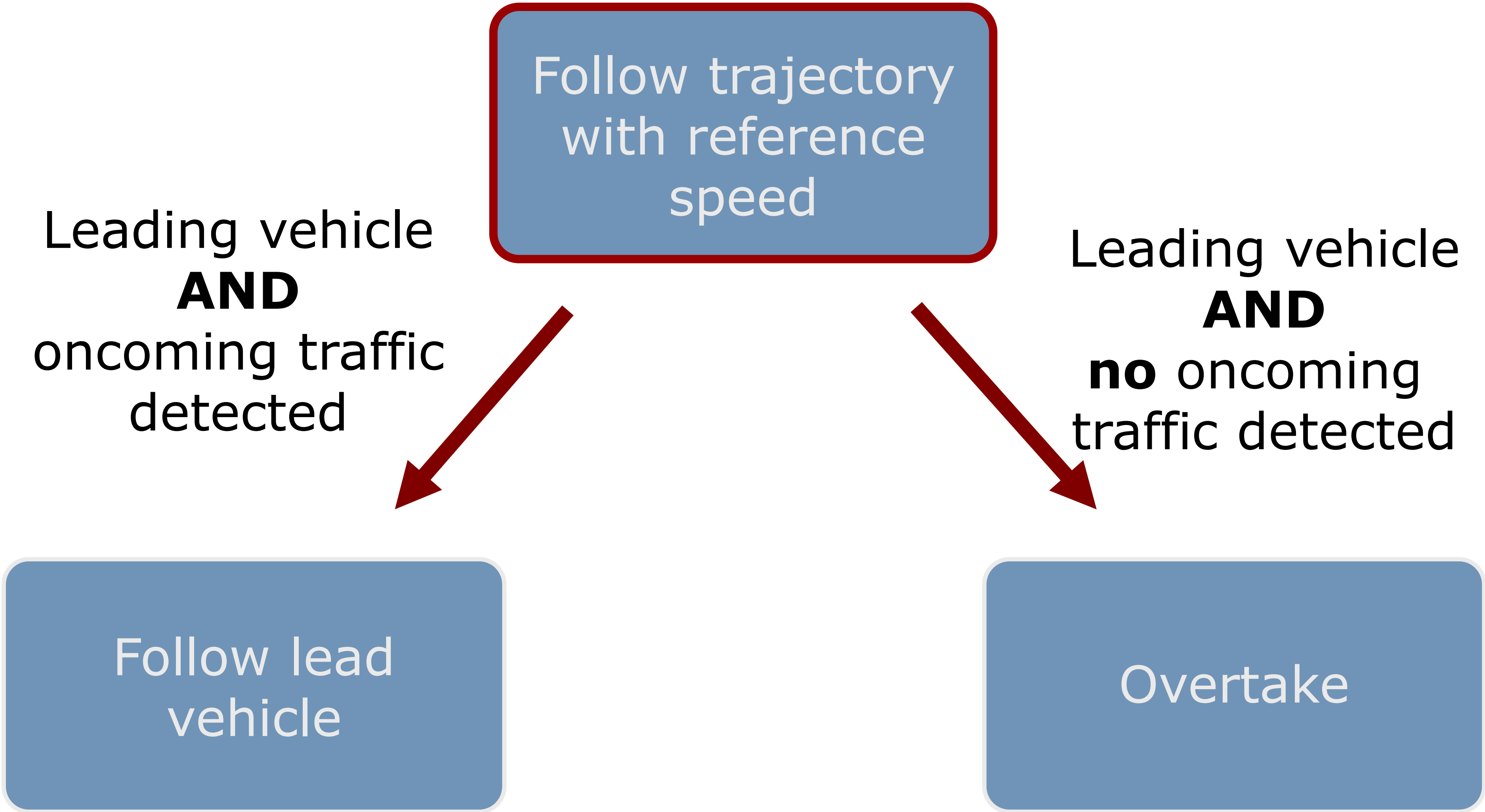
Finite State Machine

- Finite amount of possible states
- Current state defines a maneuver
- Transition between states based on inputs
- Here: Pre-defined rules determine transitions like **If** X **THEN** Y

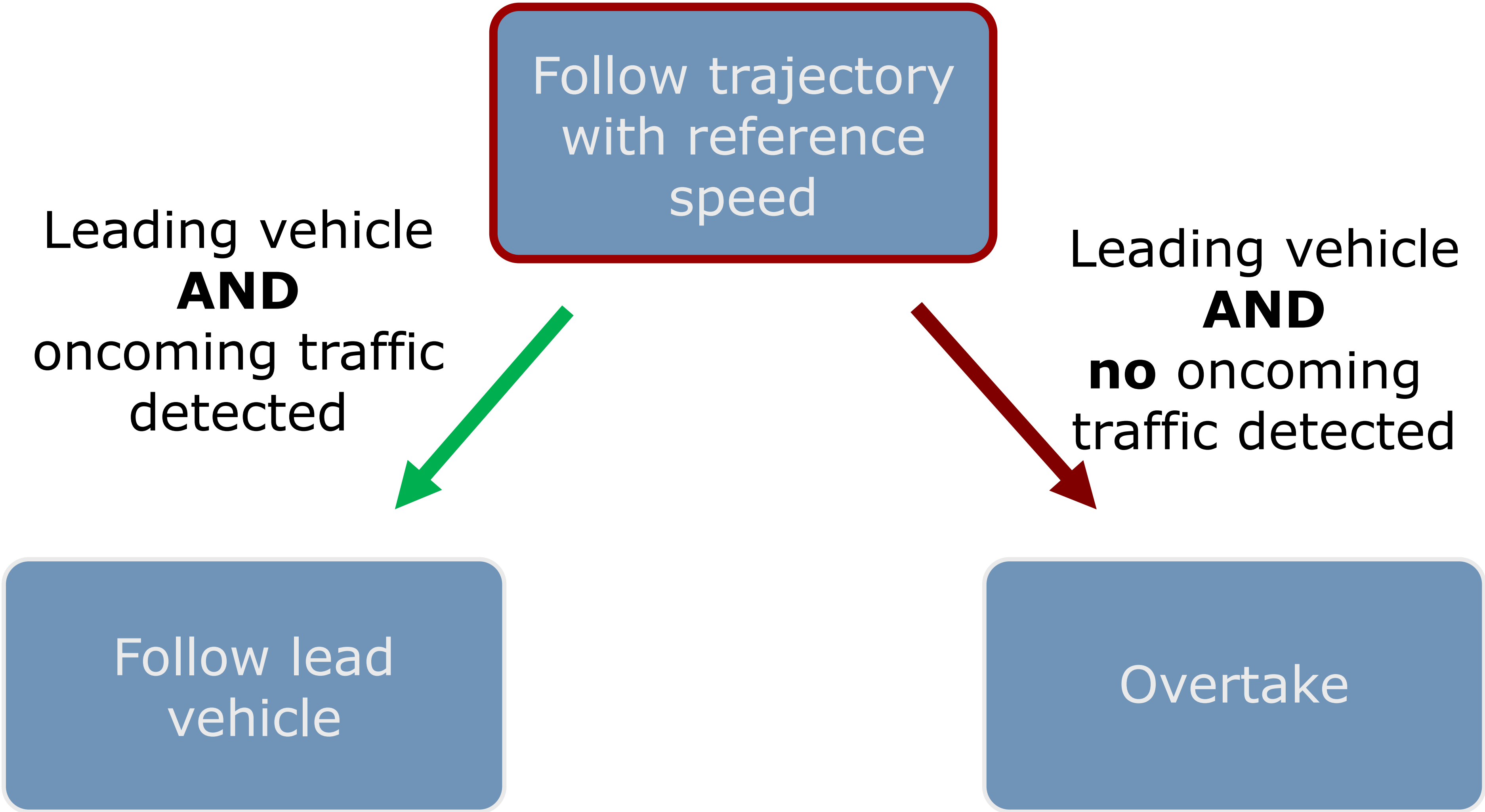
Finite State Machine - Example



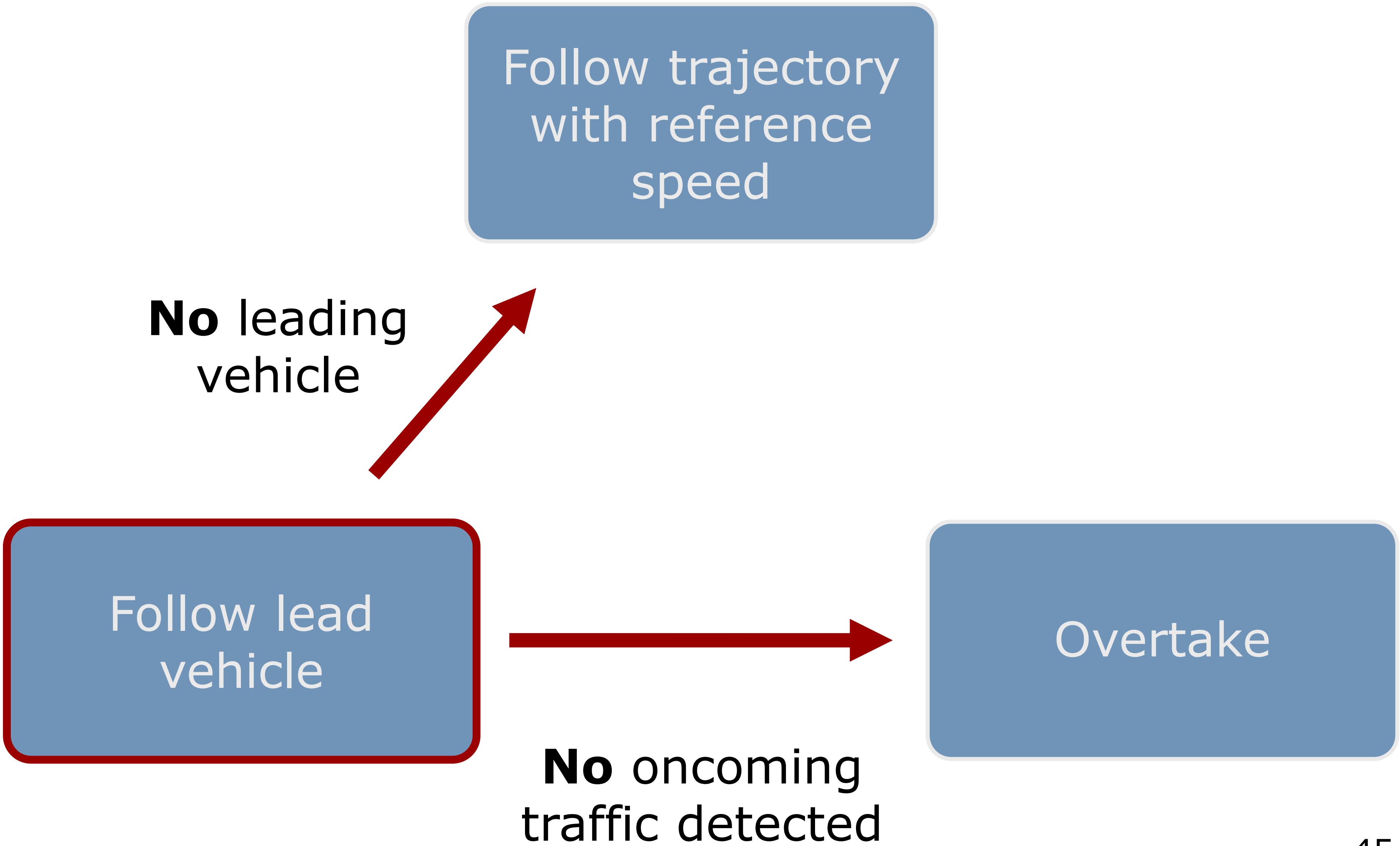
Finite State Machine - Example



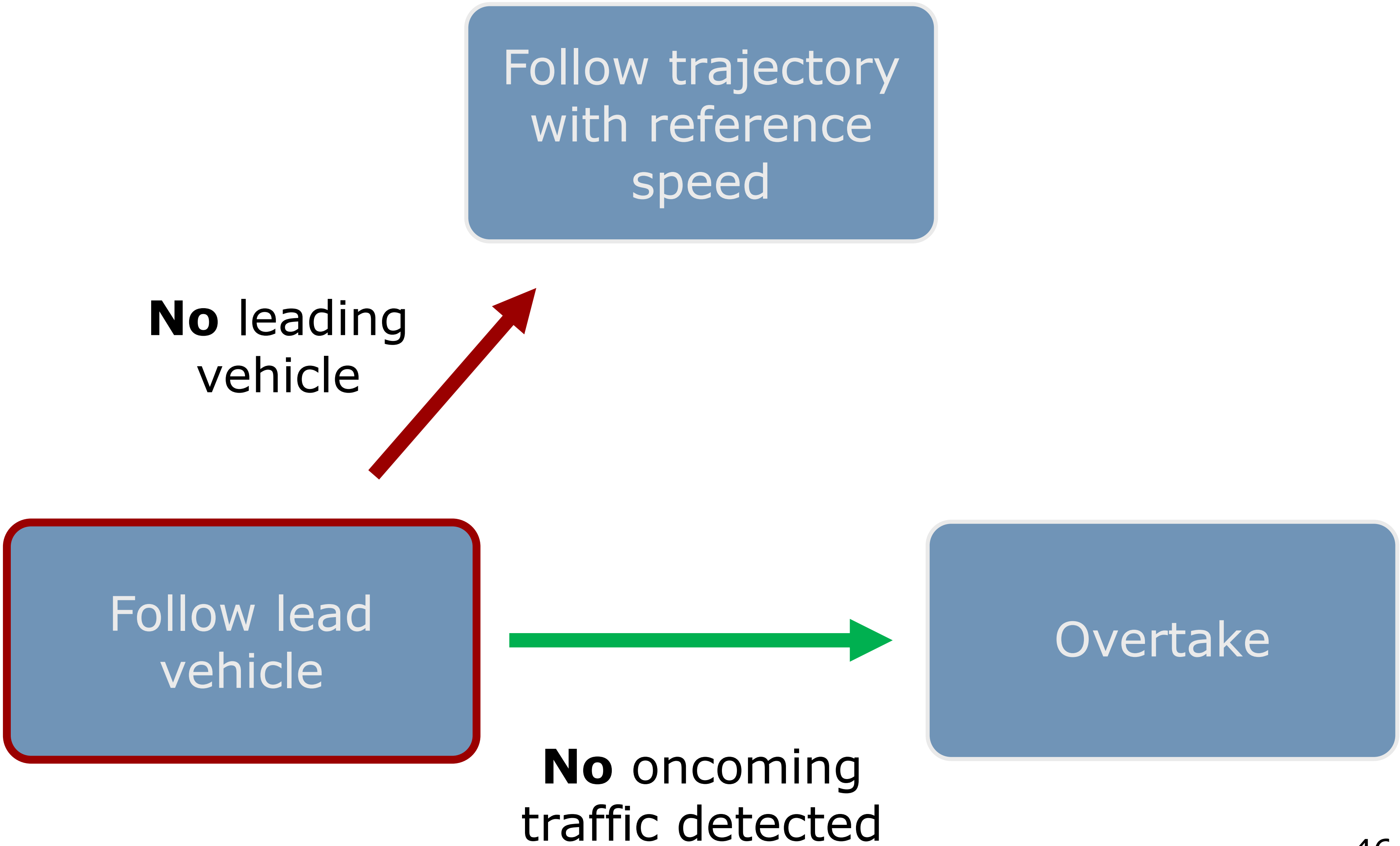
Finite State Machine - Example



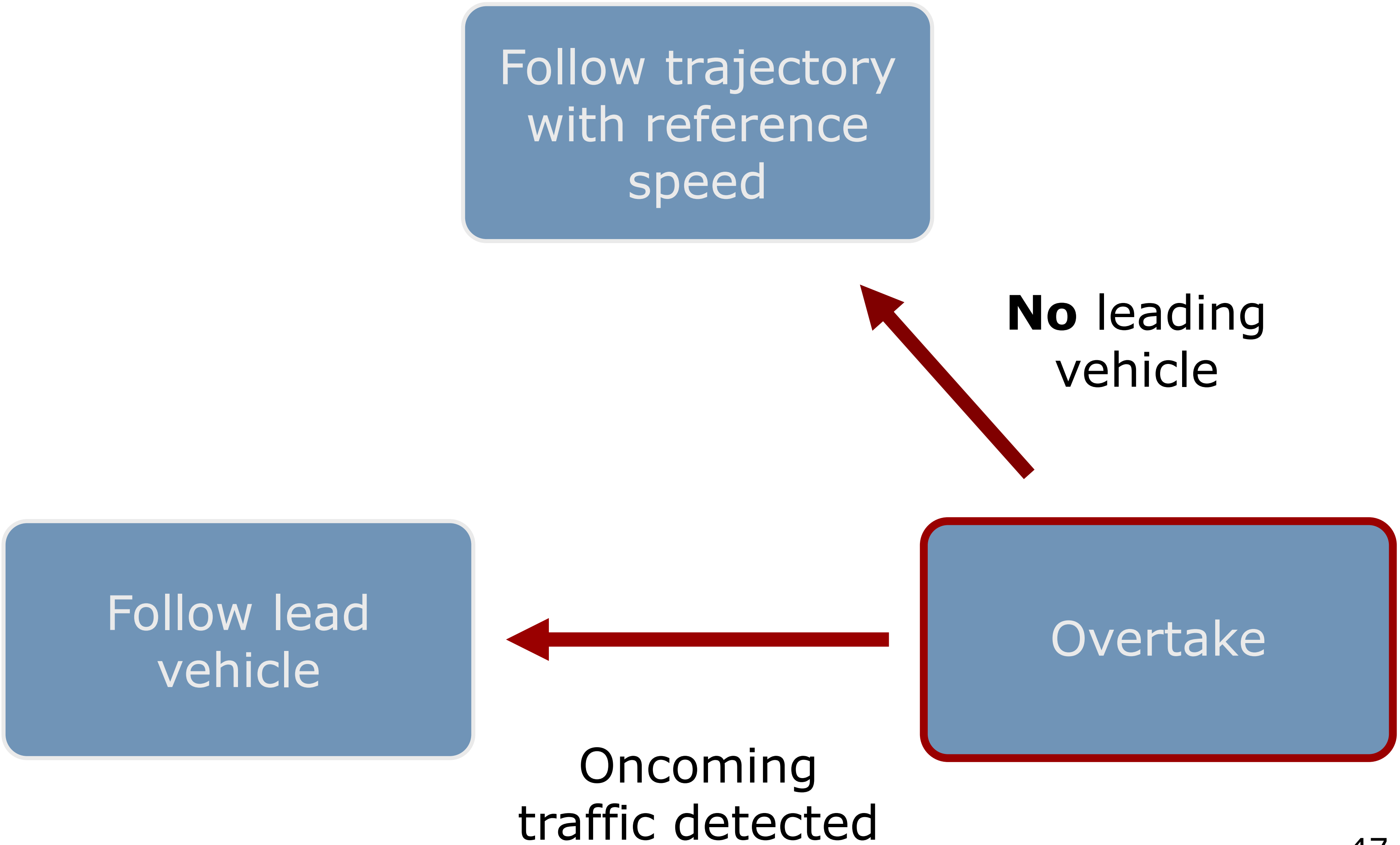
Finite State Machine - Example



Finite State Machine - Example



Finite State Machine - Example



Behavior Estimation

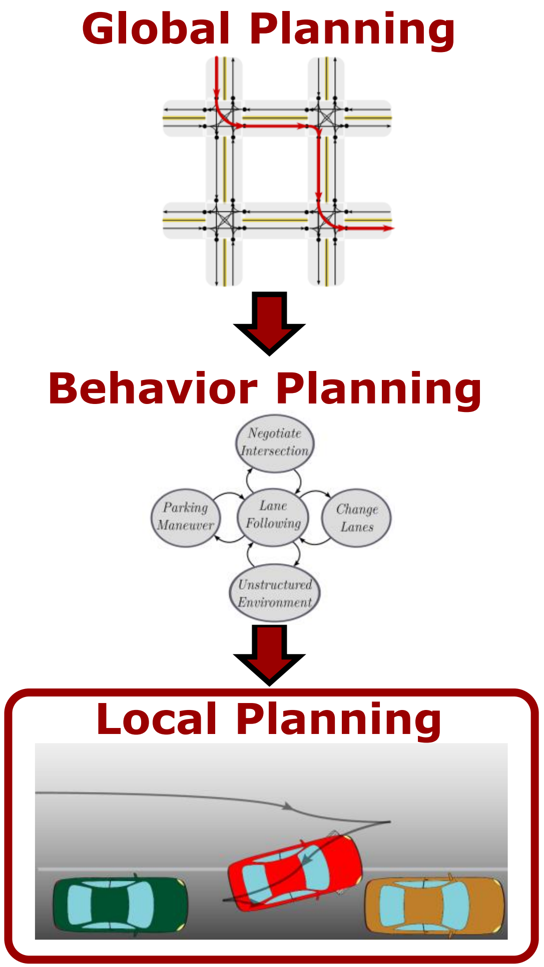
- Transitions between maneuvers depend on traffic participants
- Future behaviors of others are not known
- Behavior estimation can support decision making

[Courtesy of Waymo]

Behavior Estimation

- Knowledge of:
  - Past positions, velocities, accelerations
  - Map information
  - Sensor data (camera, LiDAR, radar)
- Physics-based, maneuver-based and interaction-aware approaches
- Multimodal predictions cover multiple possible outcomes

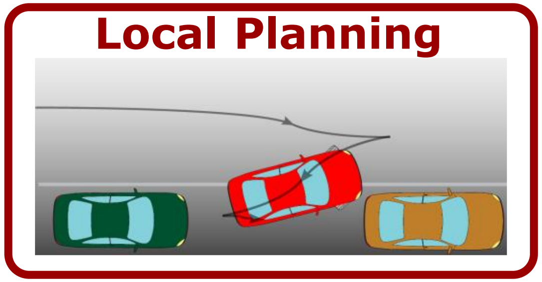
Local Planning



[Paden et al.]

Local Planning

Desired maneuver
and measurement of surroundings



Path with velocity
profile to follow

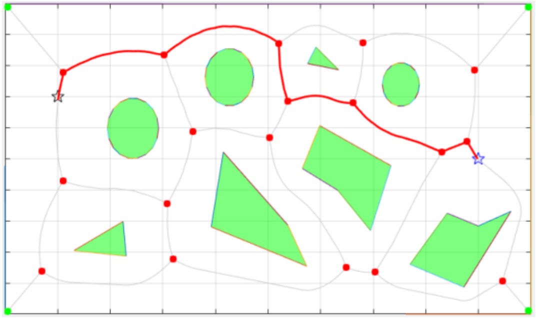
Local Planning

- Returns reference trajectory to follow within a predefined horizon
- Generates a velocity profile
- Trajectory should be feasible and obstacle free

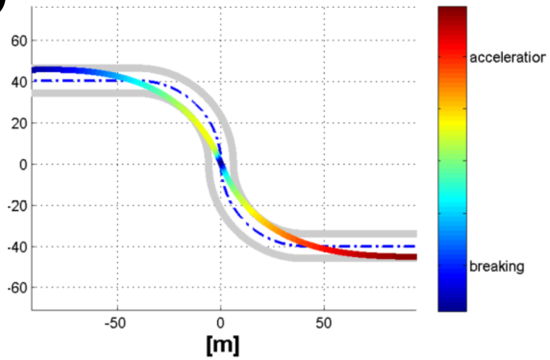
➡ Generate, score and pass trajectories to tracking controller

Path Generation Examples I

- Combinatorial planner
  + Explicitly model obstacles
  - Can become intractable



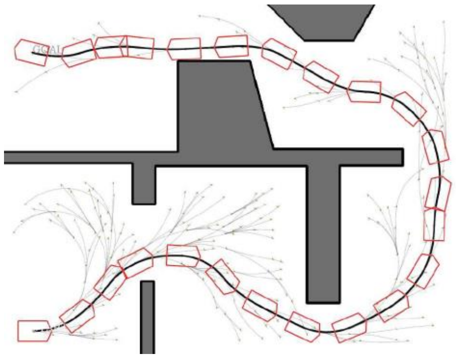
- Variational planner
  + Optimize with respect to obstacles and dynamics
  - Slow and might not converge



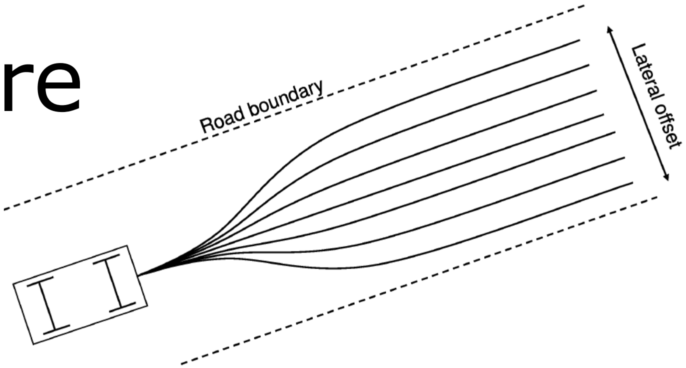
[Magid et al., Rucco et al.]

Path Generation Examples II

- Sampling based planner
  + Fast, easy to implement
  - Often sub-optimal



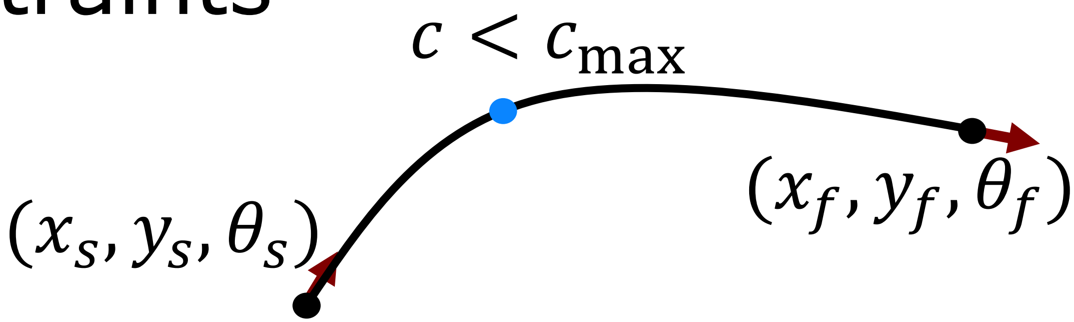
- Lattice based planner
  + Adapt to road structure
  - Limited action space



[Blanco et al., Thrun et al.]

Parametric Curves

- Account for boundary conditions and kinematic constraints



- Example: Cubic spline

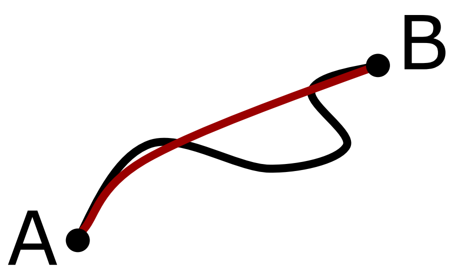
$$\boldsymbol{r}(u) = [x(u), y(u)] \qquad u \in [0,1]$$

$$x(u) = \alpha_3 u^3 + \alpha_2 u^2 + \alpha_1 u + \alpha_0$$

$$y(u) = \beta_3 u^3 + \beta_2 u^2 + \beta_1 u + \beta_0$$

Objective Functions

- Optimize choice of local trajectory
- Combine different objectives like distance to reference, control effort, curvature, collision, ...



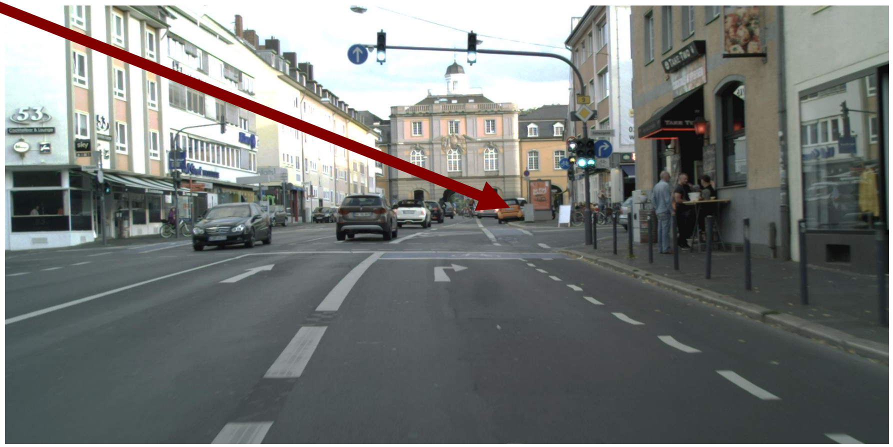
- Goal: Reduce sum of weighted (discounted) objectives

Obstacles

- Static
  - Parked cars
  - Illegal lanes
  - Sidewalk
- Dynamic
  - Other vehicles
  - Pedestrians

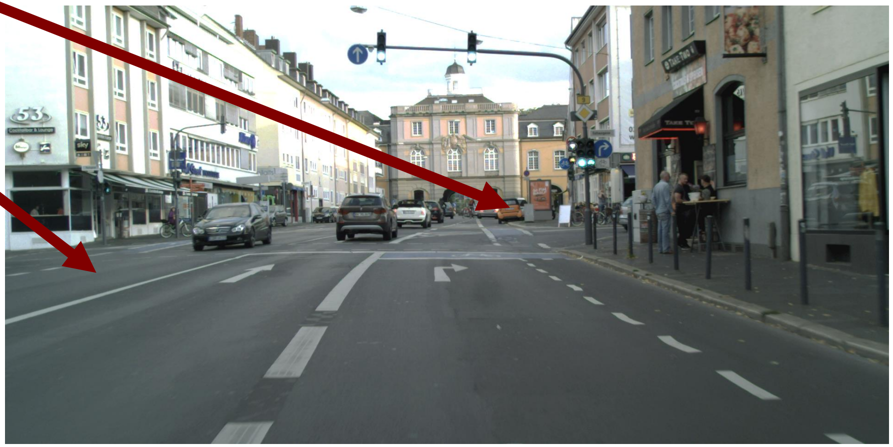
Obstacles

- Static
  - Parked cars
  - Illegal lanes
  - Sidewalk
- Dynamic
  - Other vehicles
  - Pedestrians



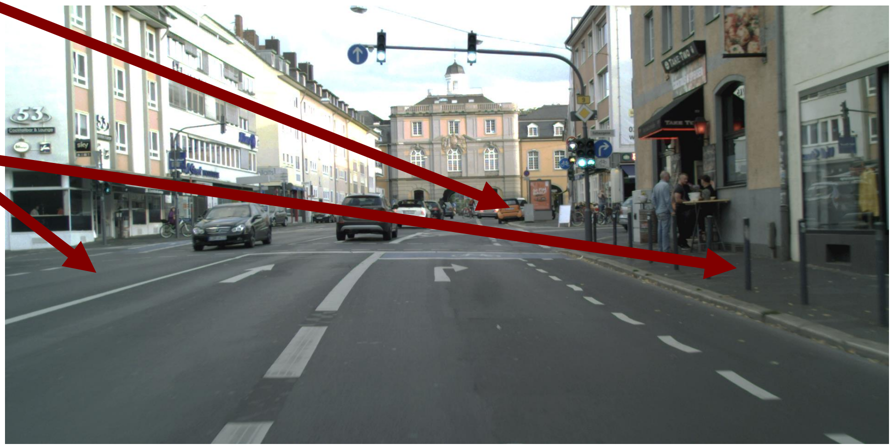
Obstacles

- Static
  - Parked cars
  - Illegal lanes
  - Sidewalk
- Dynamic
  - Other vehicles
  - Pedestrians



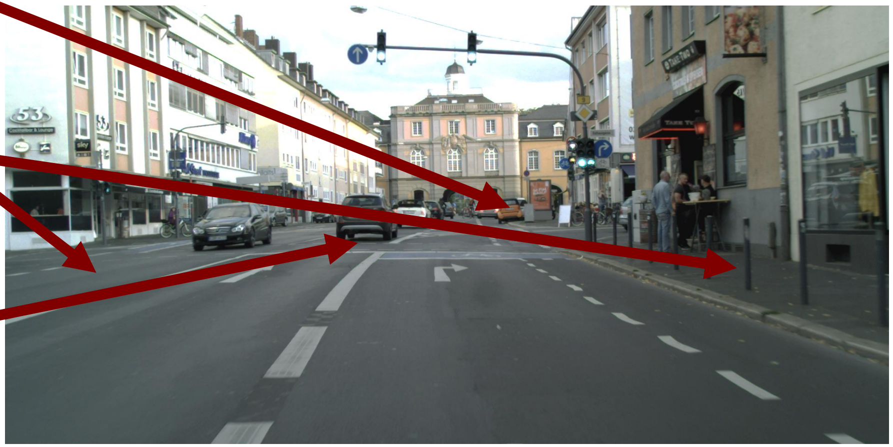
Obstacles

- Static
  - Parked cars
  - Illegal lanes
  - Sidewalk
- Dynamic
  - Other vehicles
  - Pedestrians



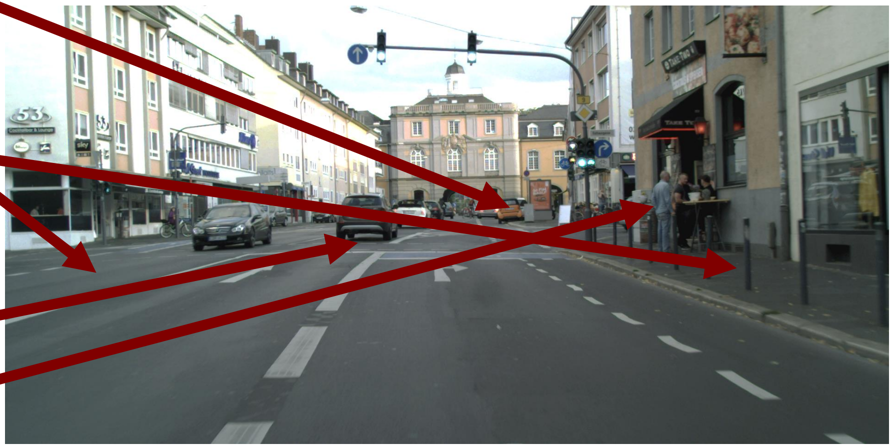
Obstacles

- Static
  - Parked cars
  - Illegal lanes
  - Sidewalk
- Dynamic
  - Other vehicles
  - Pedestrians



Obstacles

- Static
  - Parked cars
  - Illegal lanes
  - Sidewalk
- Dynamic
  - Other vehicles
  - Pedestrians



Collision Avoidance

Generate **collision-free** trajectories

or

Generate arbitrary trajectories and **check for collision**

- Use safety margins
- Collision-free trajectory might not exist for long planning horizons

Collision Avoidance

Generate **collision-free** trajectories

or

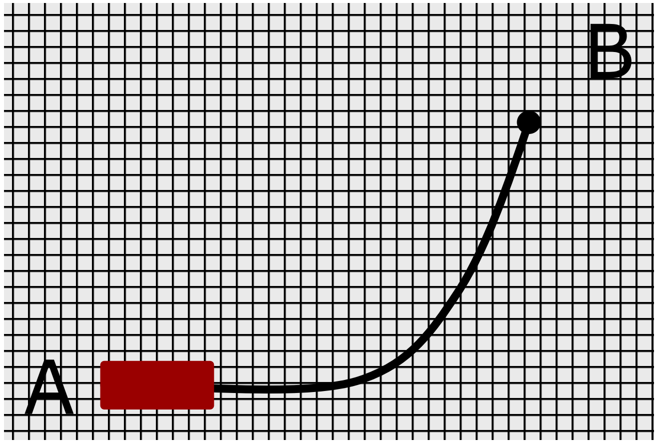
Generate arbitrary trajectories and **check for collision**

- Use safety margins
- Collision-free trajectory might not exist for long planning horizons

Collision Checking

- Time-to-collision (TTC)
- Swath computation for occupancy grid maps
- Circle Collision Checking

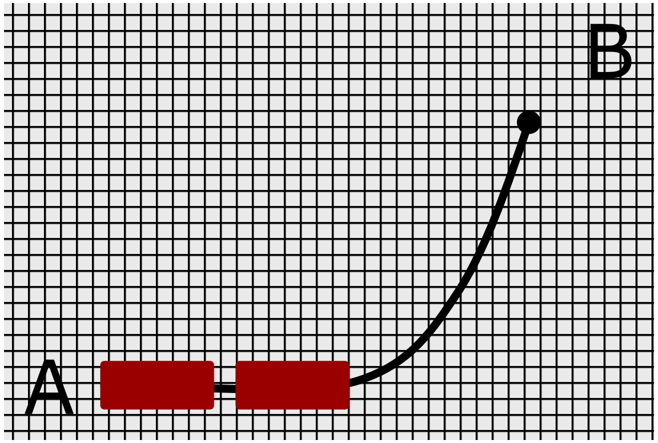
Collision Checking



- Time-to-collision (TTC)
- Swath computation for occupancy grid maps
- Circle Collision Checking

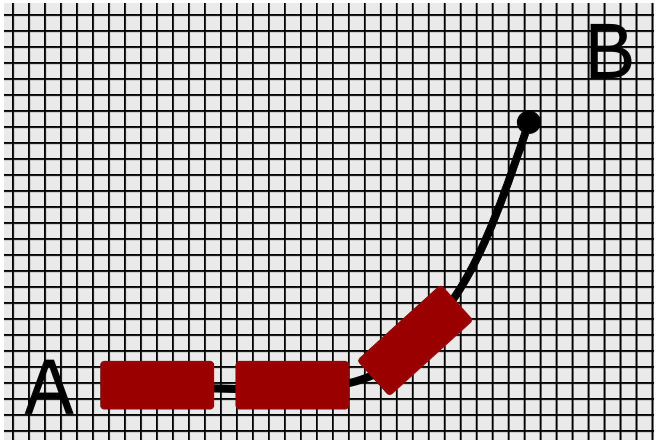
Collision Checking

- Time-to-collision (TTC)
- Swath computation for occupancy grid maps
- Circle Collision Checking



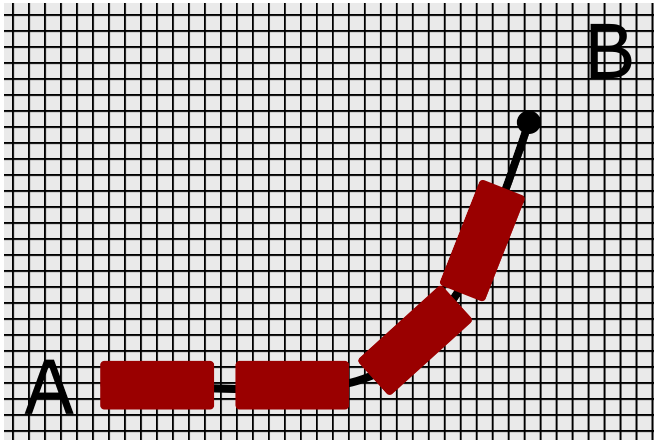
Collision Checking

- Time-to-collision (TTC)
- Swath computation for occupancy grid maps
- Circle Collision Checking

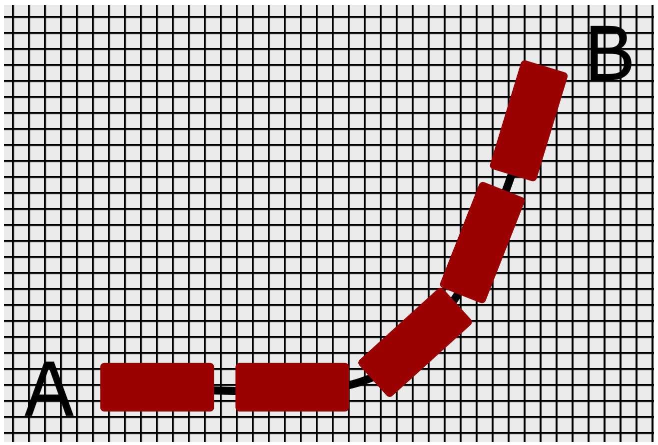


Collision Checking

- Time-to-collision (TTC)
- Swath computation for occupancy grid maps
- Circle Collision Checking



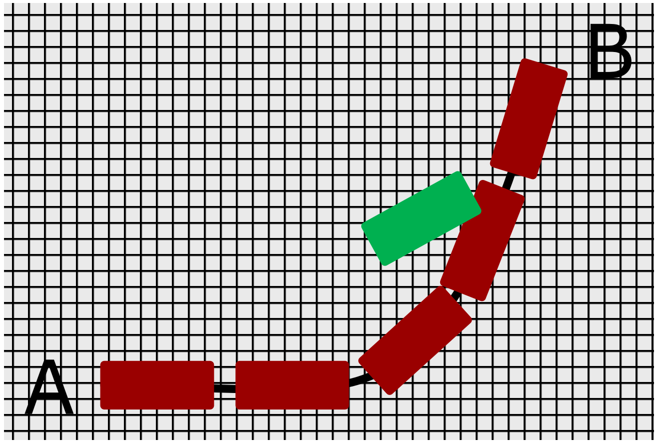
Collision Checking



- Time-to-collision (TTC)
- Swath computation for occupancy grid maps
- Circle Collision Checking

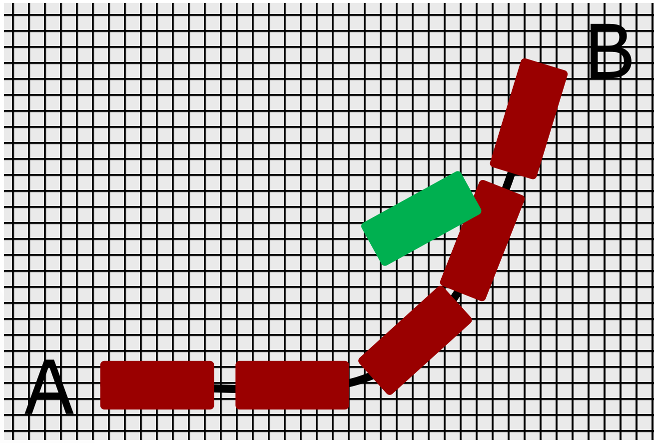
Collision Checking

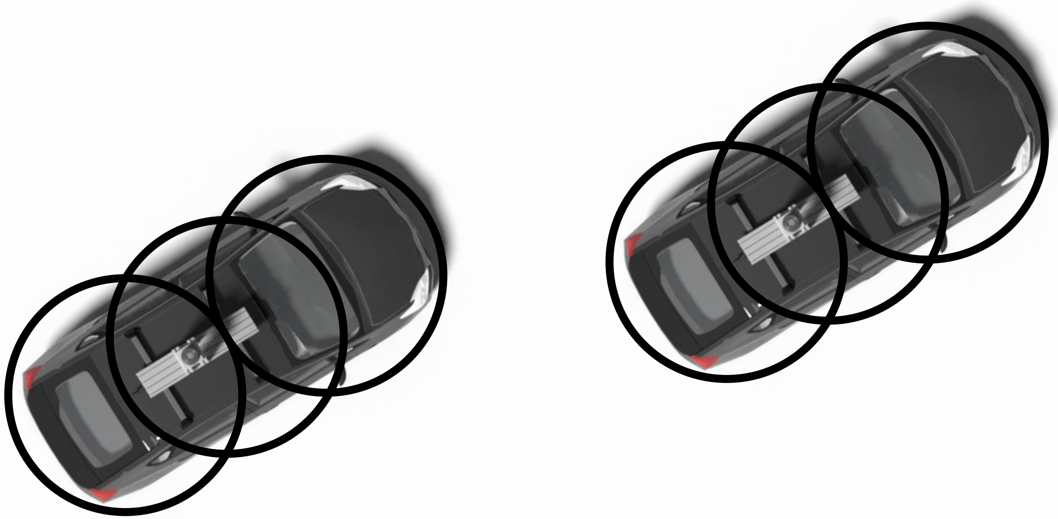
- Time-to-collision (TTC)
- Swath computation for occupancy grid maps
- Circle Collision Checking



Collision Checking

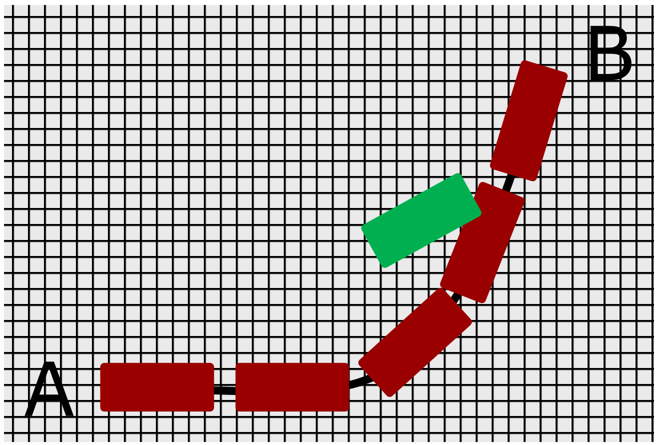
- Time-to-collision (TTC)
- Swath computation for occupancy grid maps
- Circle Collision Checking

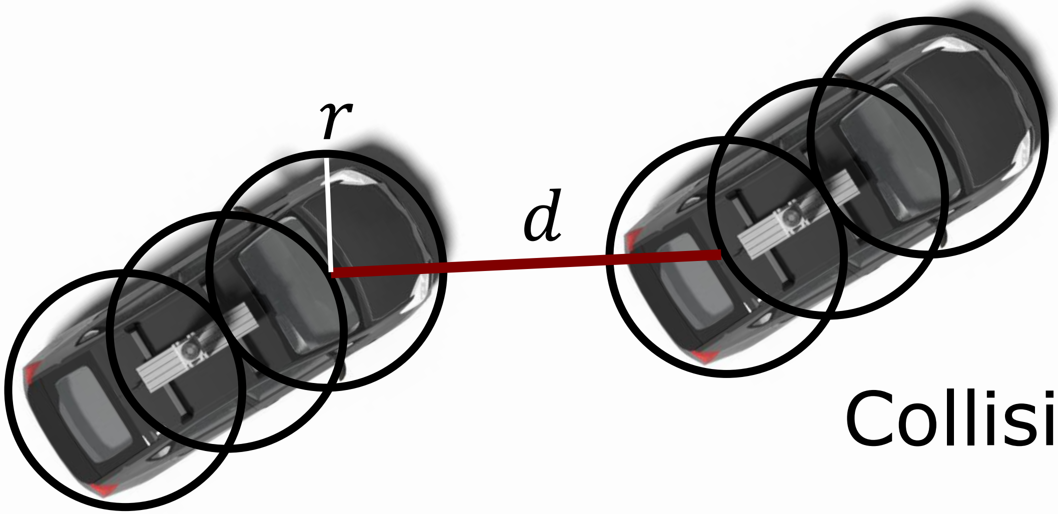




Collision Checking

- Time-to-collision (TTC)
- Swath computation for occupancy grid maps
- Circle Collision Checking





Collision if $d < 2r$

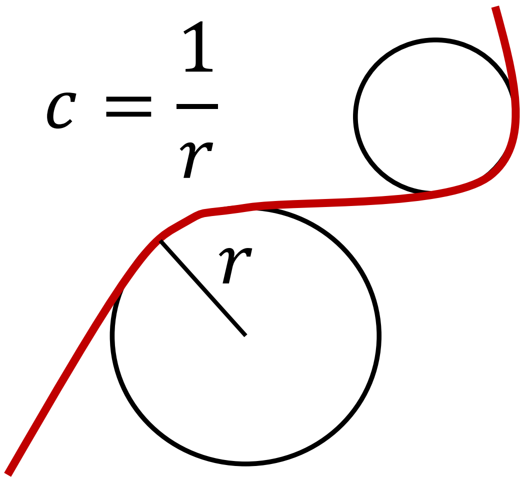
Motion Constraints

- Bicycle model limits curvature c
- Curvature c is inversely proportional to radius r of turning circle
- Lateral acceleration should be limited

$$a_{lat} = \frac{v^2}{r} \leq a_{lat_{max}}$$

$$cv^2 \leq a_{lat_{max}}$$

$$c = \frac{1}{r}$$

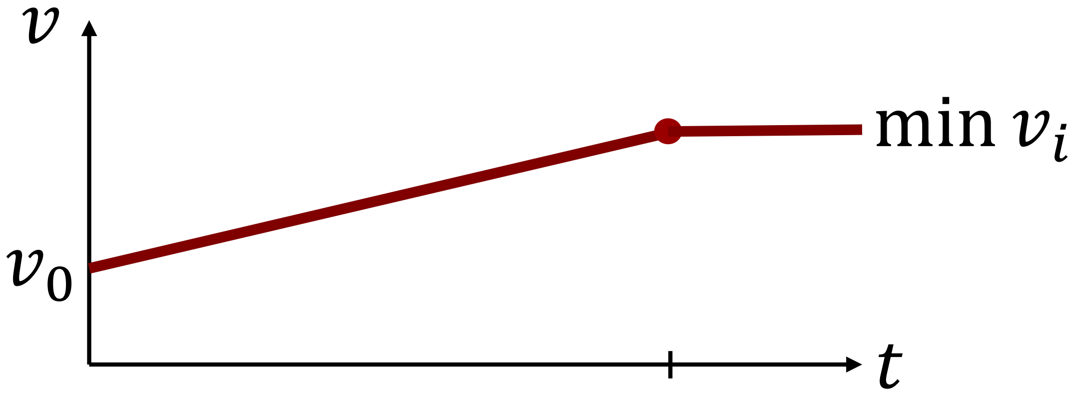


Velocity Profile Generation

- Velocity profile is i.a. limited by

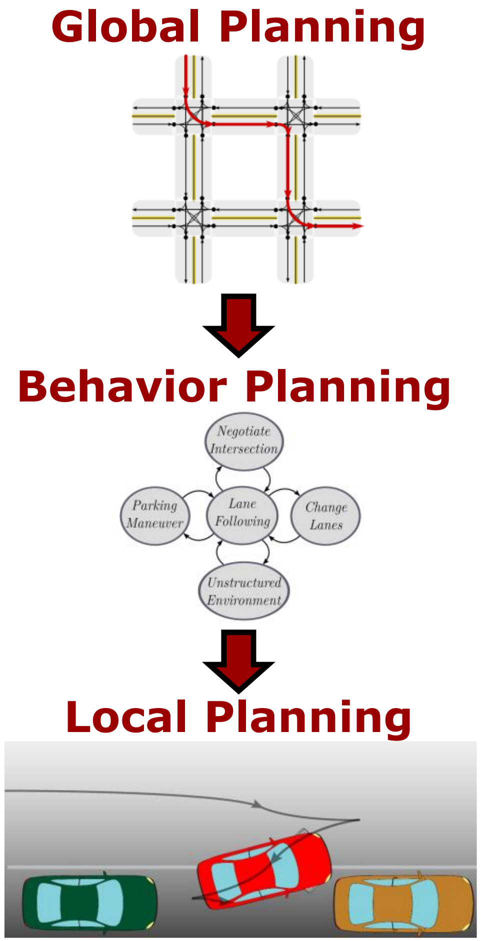
v_{limit}	Speed limit
v_{c}	Imposed constraint of lateral acceleration
v_{beh}	Reference from Behavior Planner

- Easiest implementation: Linear ramp



Summary

- Divide planning problem into subtasks
- Global planning returns optimal overall route
- Behavior planning accounts for traffic rules and participants
- Local planning computes feasible trajectory



[Paden et al.]

Thank you for your attention

References

- B. Paden et al. “A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles.” *IEEE Transactions on Intelligent Vehicles* 1 (2016): 33-55.
- G. Lozenguez et al. “Punctual versus continuous auction coordination for multi-robot and multi-task topological navigation.” *Autonomous Robots* 40 (2016): 599-613.
- F. Poggenhans et al. “Lanelet2: A high-definition map framework for the future of automated driving.” *2018 21st International Conference on Intelligent Transportation Systems (ITSC)* (2018): 1672-1679.
- E. Magid et al. “Voronoi-based trajectory optimization for UGV path planning.” *2017 International Conference on Mechanical, System and Control Engineering (ICMSC)* (2017): 383-387.
- A. Rucco et al. “Computing minimum lap-time trajectories for a single-track car with load transfer.” *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)* (2012): 6321-6326.
- J. Blanco et al. “TP-Space RRT – Kinematic Path Planning of Non-Holonomic Any-Shape Vehicles.” *International Journal of Advanced Robotic Systems* 12 (2015): n. pag.
- S. Thrun et al. “Stanley: The robot that won the DARPA Grand Challenge.” *J. Field Robotics* 23 (2006): 661-692.