Automated Verification of Social Law Robustness in STRIPS

Erez Karpas Alexander Shlefyman Moshe Tennenholtz

Faculty of Industrial Engineering and Management
Technion — Israel Institute of Technology

Workshop on Distributed and Multi-Agent Planning — ICAPS 2016

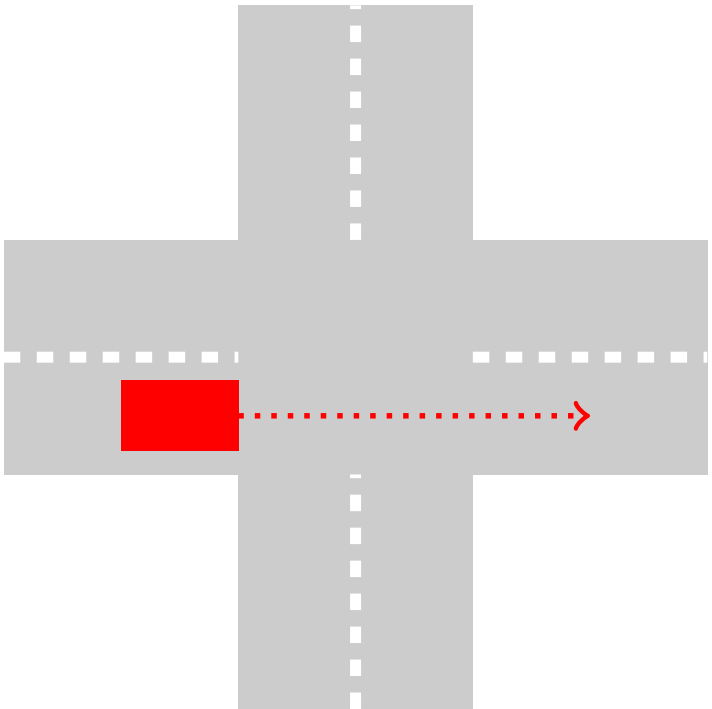
Motivation

- Multi-agent planning is hard
  - . . . centralized or distributed
- We would like each agent to be able to plan on its own
  - . . . without considering what the other agents might do
- This simplifies the planning problem significantly

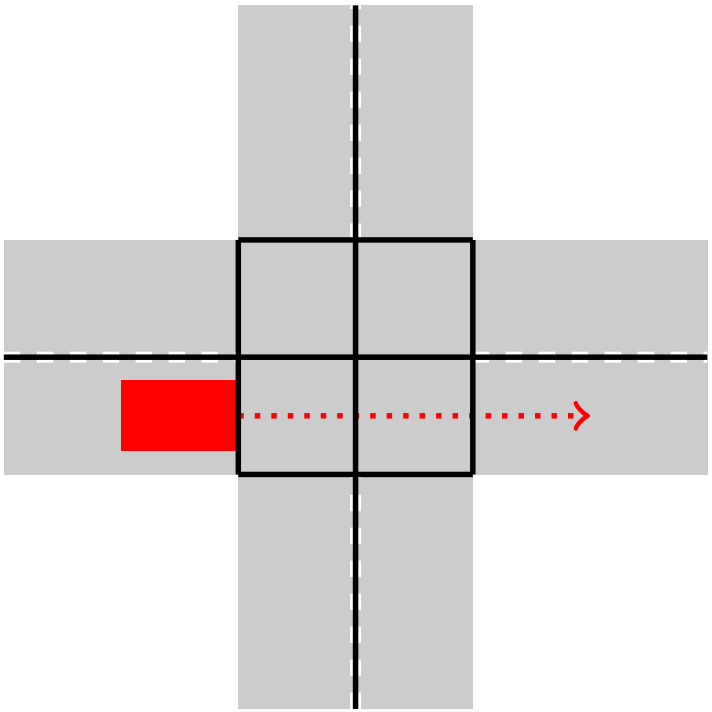
Our Setting: Multi-Agent Systems

- MA-STRIPS, except each agent has individual goal
- Formalized as $\Pi = \langle F, \{A_i\}_{i=1}^{n}, I, \{G_i\}_{i=1}^{n} \rangle$, where
  - F is a set of facts
  - $I \subseteq F$ is the initial state
  - $G_i \subseteq F$ is the goal of agent i
  - A_i is the set of actions of agent i

Example: Intersection

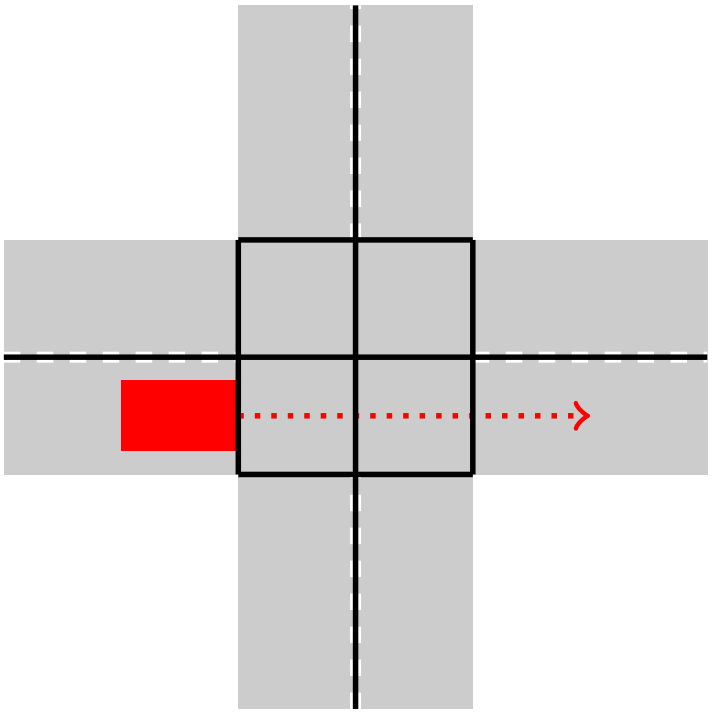


Example: Intersection



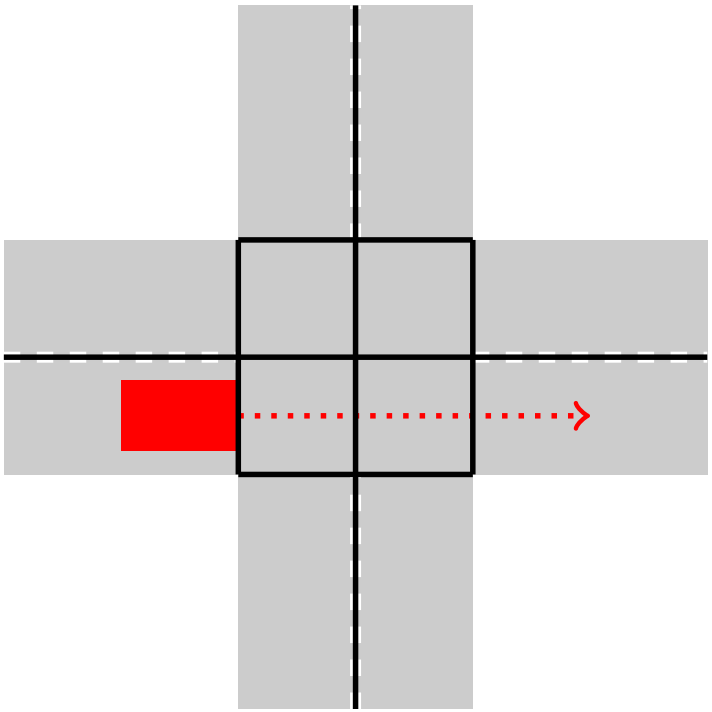
- Facts:
  - AT(car, region)
  - CLEAR(region)

Example: Intersection



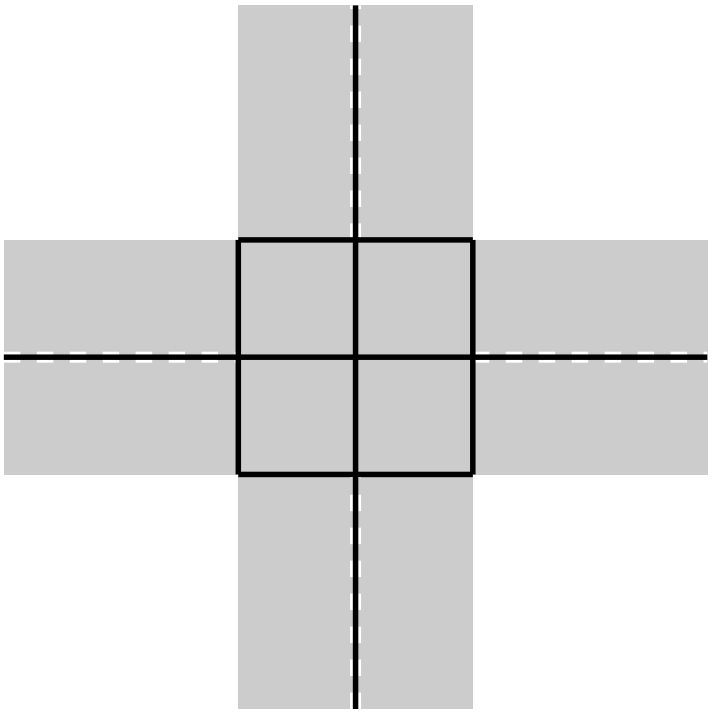
- Facts:
  - AT(car, region)
  - CLEAR(region)
- Actions:
  - DRIVE(car,from,to)
  - ARRIVE(car,region)

Example: Intersection



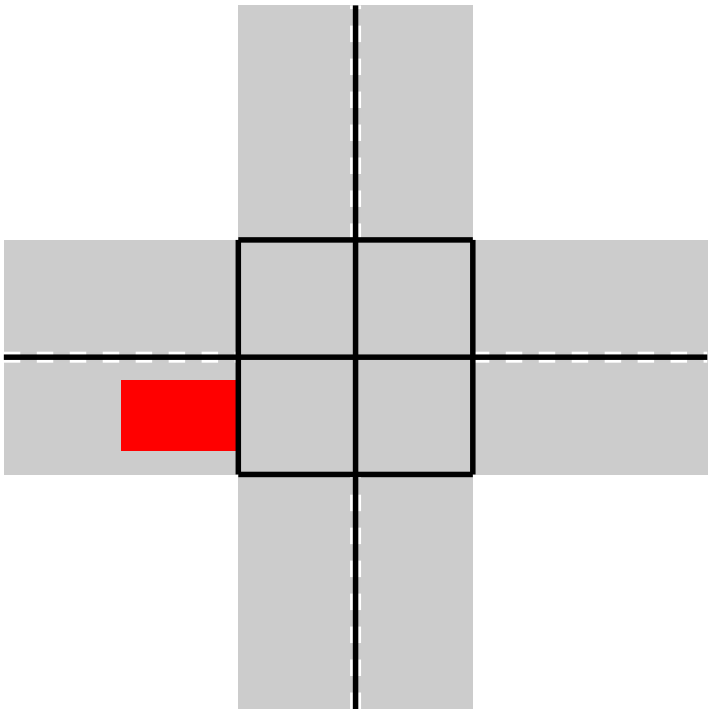
- Facts:
  - AT(car, region)
  - CLEAR(region)
- Actions:
  - DRIVE(car,from,to)
  - ARRIVE(car,region)

Example: Intersection



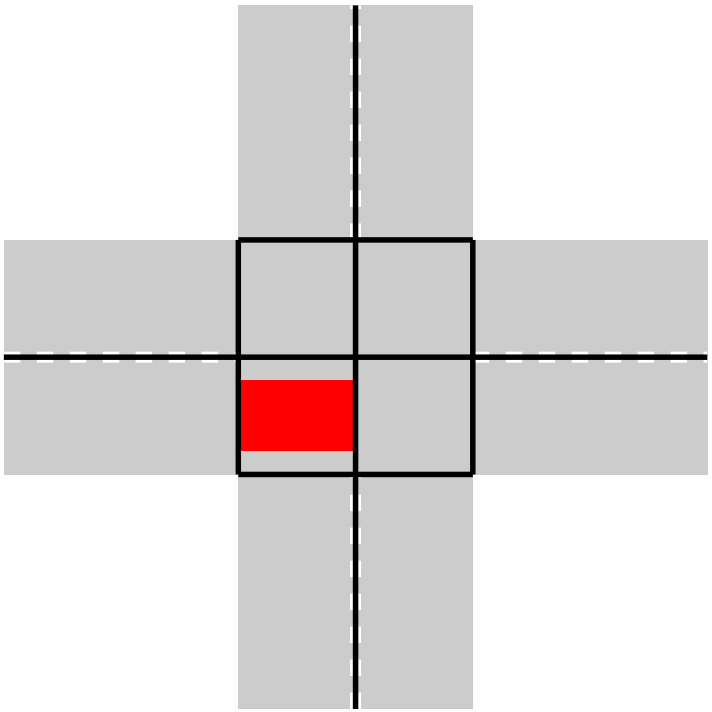
- Facts:
  - AT(car, region)
  - CLEAR(region)
- Actions:
  - DRIVE(car,from,to)
  - ARRIVE(car,region)
- Example plan for red car:
  - ARRIVE(red, Went)
  - DRIVE(red, Went, SW)
  - DRIVE(red, SW, SE)
  - DRIVE(red, SE, Eex)

Example: Intersection



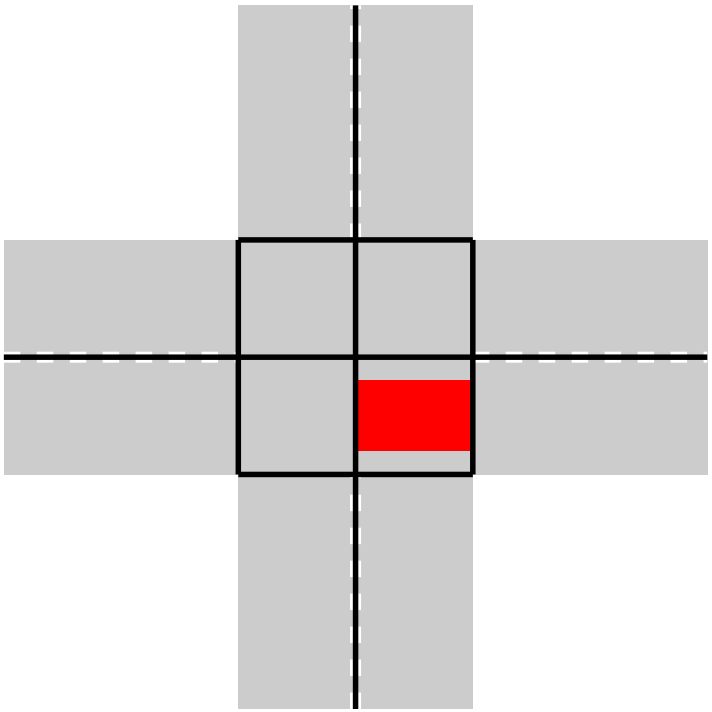
- Facts:
  - AT(car, region)
  - CLEAR(region)
- Actions:
  - DRIVE(car,from,to)
  - ARRIVE(car,region)
- Example plan for red car:
  - ARRIVE(red, Went)
  - DRIVE(red, Went, SW)
  - DRIVE(red, SW, SE)
  - DRIVE(red, SE, Eex)

Example: Intersection



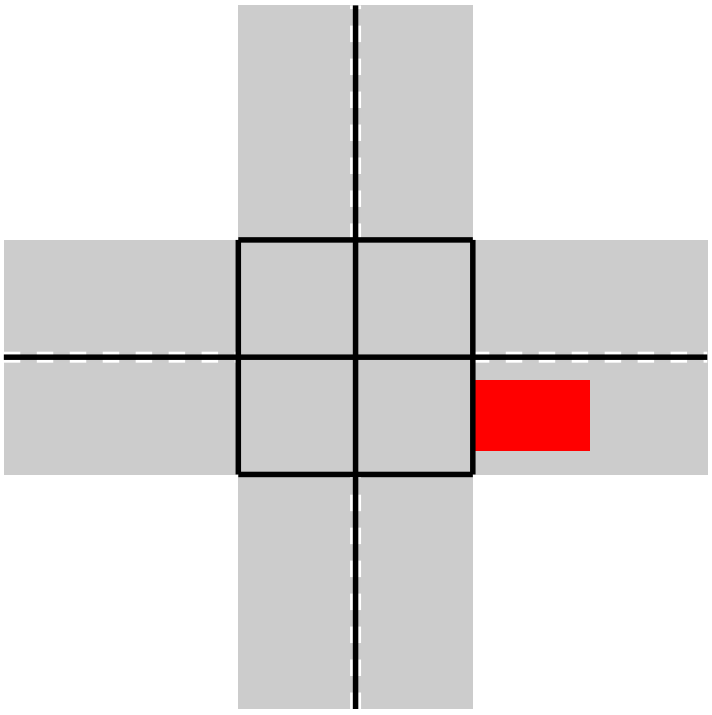
- Facts:
  - AT(car, region)
  - CLEAR(region)
- Actions:
  - DRIVE(car,from,to)
  - ARRIVE(car,region)
- Example plan for red car:
  - ARRIVE(red, Went)
  - DRIVE(red, Went, SW)
  - DRIVE(red, SW, SE)
  - DRIVE(red, SE, Eex)

Example: Intersection



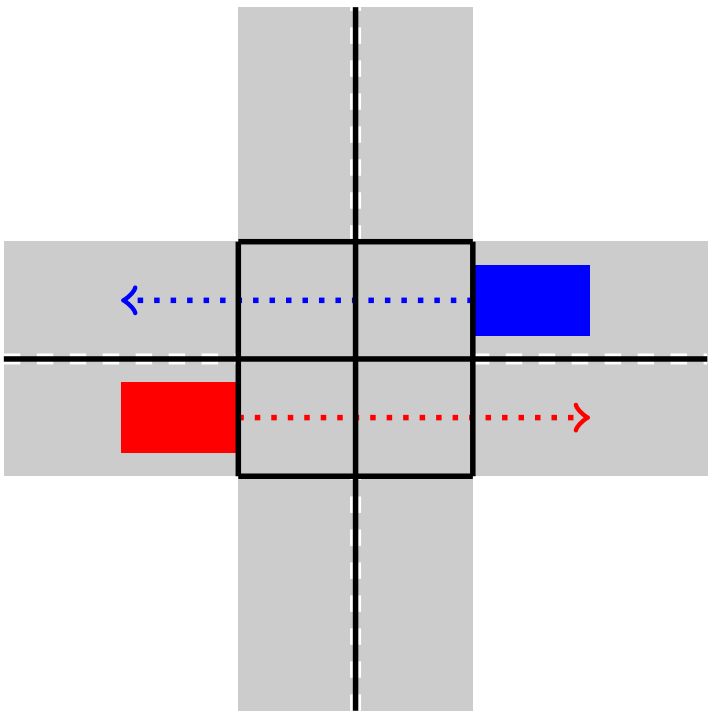
- Facts:
  - AT(car, region)
  - CLEAR(region)
- Actions:
  - DRIVE(car,from,to)
  - ARRIVE(car,region)
- Example plan for red car:
  - ARRIVE(red, Went)
  - DRIVE(red, Went, SW)
  - DRIVE(red, SW, SE)
  - DRIVE(red, SE, Eex)

Example: Intersection



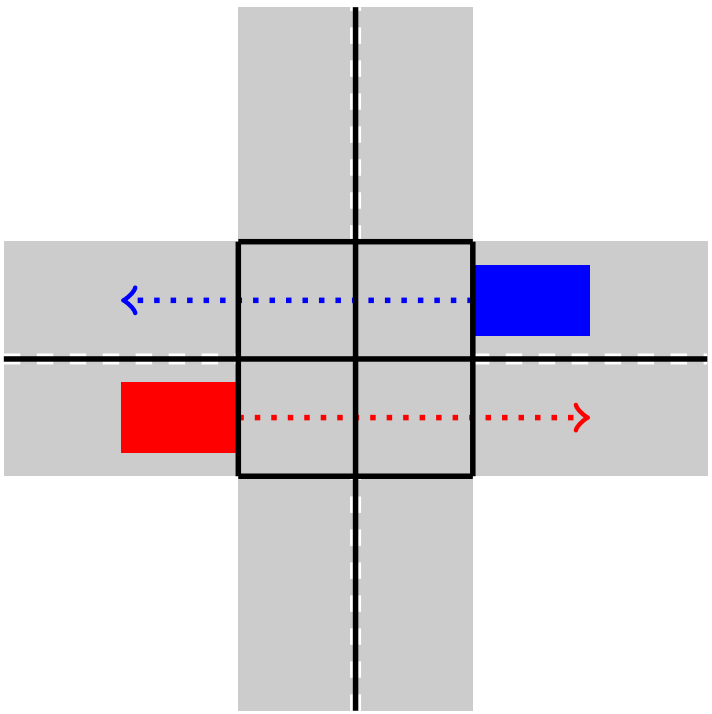
- Facts:
  - AT(car, region)
  - CLEAR(region)
- Actions:
  - DRIVE(car,from,to)
  - ARRIVE(car,region)
- Example plan for red car:
  - ARRIVE(red, Went)
  - DRIVE(red, Went, SW)
  - DRIVE(red, SW, SE)
  - DRIVE(red, SE, Eex)

Example: Intersection with 2 Cars



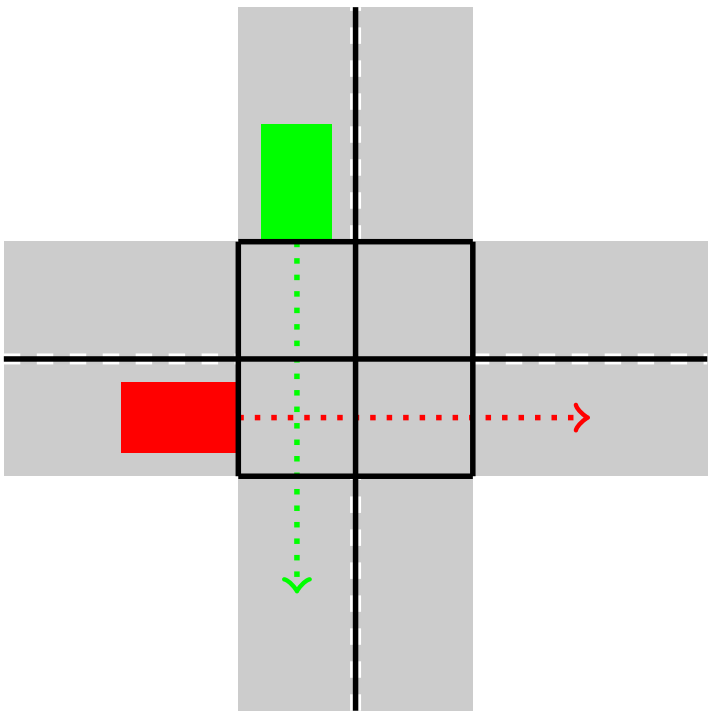
- Plan for red car:
  - ARRIVE(red, Went)
  - DRIVE(red, Went, SW)
  - DRIVE(red, SW, SE)
  - DRIVE(red, SE, Eex)
- Plan for blue car:
  - ARRIVE(blue, Eent)
  - DRIVE(blue, Eent, NE)
  - DRIVE(blue, NE, NW)
  - DRIVE(blue, NW, Wex)
- Can these plans interfere with each other?

Example: Intersection with 2 Cars



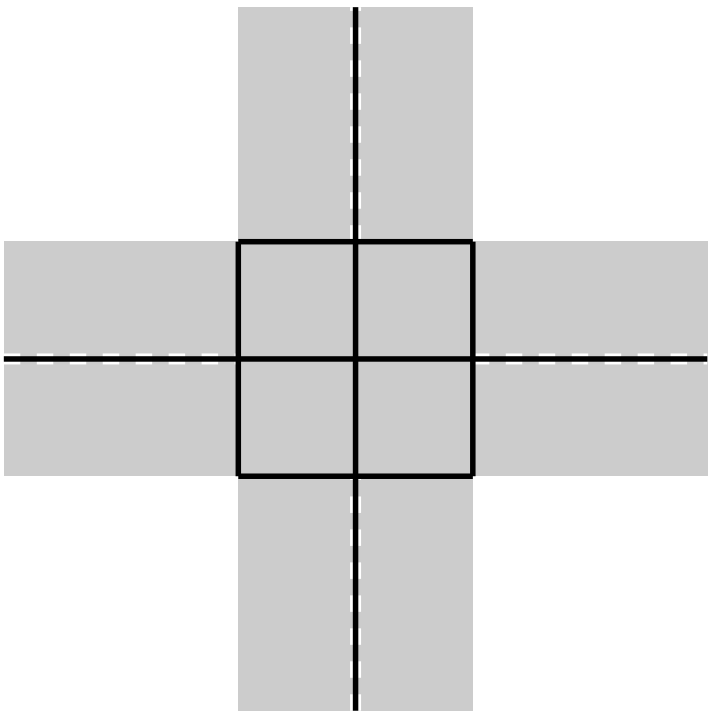
- Plan for red car:
  - ARRIVE(red, Went)
  - DRIVE(red, Went, SW)
  - DRIVE(red, SW, SE)
  - DRIVE(red, SE, Eex)
- Plan for blue car:
  - ARRIVE(blue, Eent)
  - DRIVE(blue, Eent, NE)
  - DRIVE(blue, NE, NW)
  - DRIVE(blue, NW, Wex)
- Can these plans interfere with each other?

Example: Intersection with 2 Crossing Cars



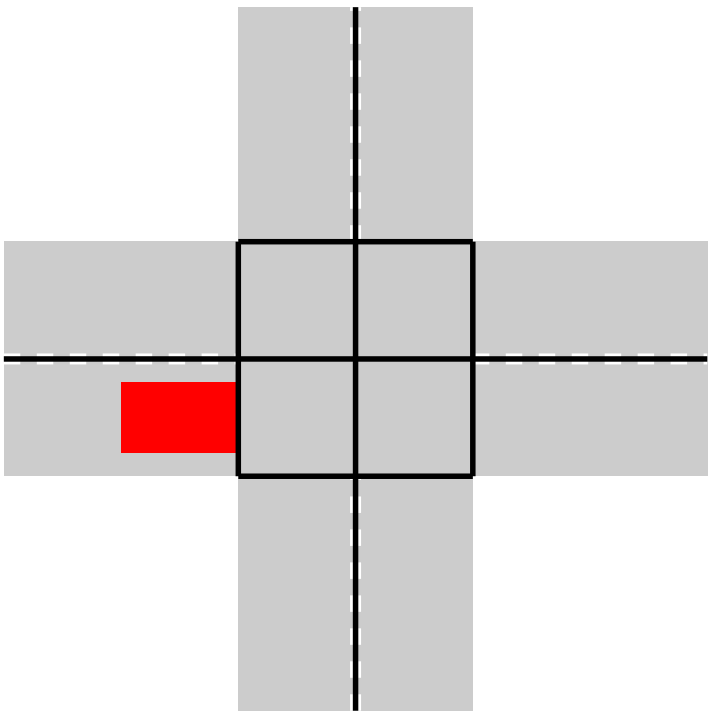
- Plan for red car:
  - ARRIVE(red, Went)
  - DRIVE(red, Went, SW)
  - DRIVE(red, SW, SE)
  - DRIVE(red, SE, Eex)
- Plan for green car:
  - ARRIVE(green, Nent)
  - DRIVE(green, Nent, NW)
  - DRIVE(green, NW, SW)
  - DRIVE(green, SW, Sex)
- Can these plans interfere with each other?

Example: Intersection with 2 Crossing Cars



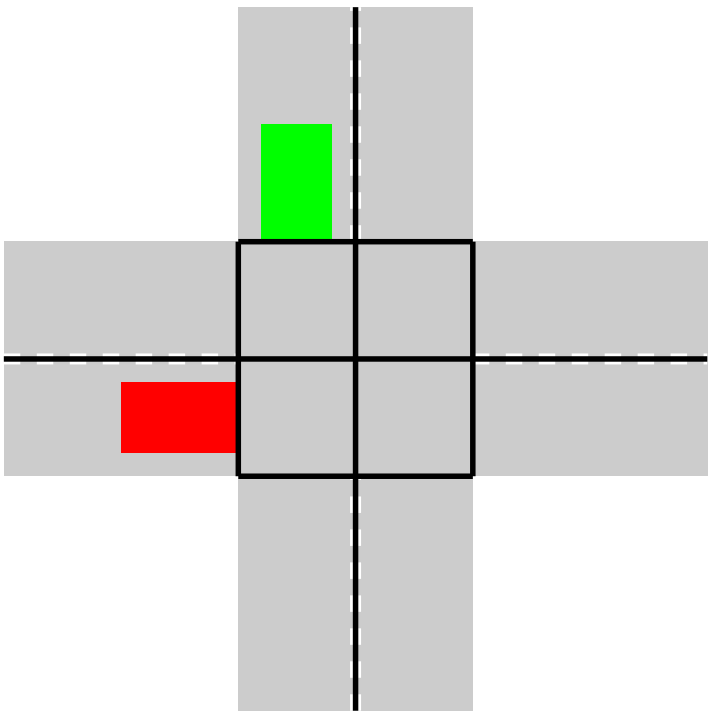
- Plan for red car:
  - ARRIVE(red, Went)
  - DRIVE(red, Went, SW)
  - DRIVE(red, SW, SE)
  - DRIVE(red, SE, Eex)
- Plan for green car:
  - ARRIVE(green, Nent)
  - DRIVE(green, Nent, NW)
  - DRIVE(green, NW, SW)
  - DRIVE(green, SW, Sex)
- Can these plans interfere with each other?

Example: Intersection with 2 Crossing Cars



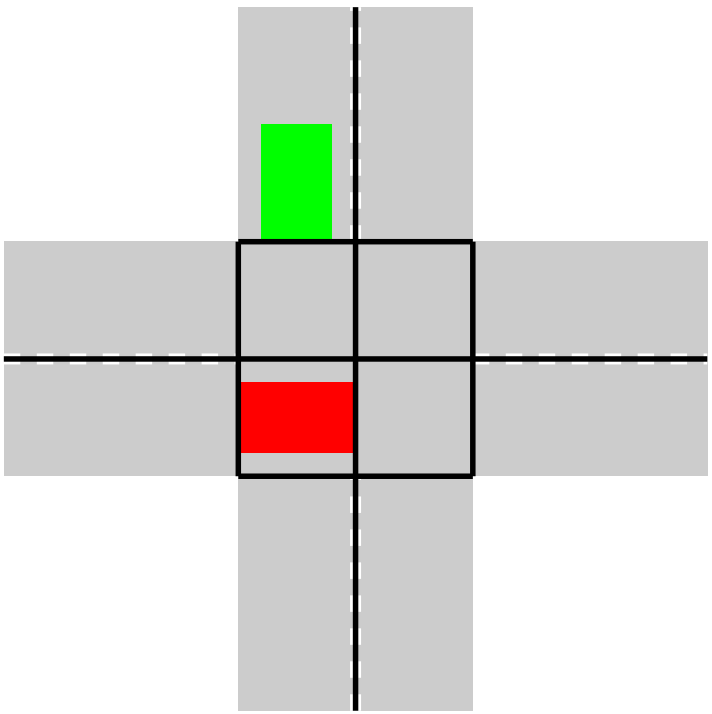
- Plan for red car:
  - ARRIVE(red, Went)
  - DRIVE(red, Went, SW)
  - DRIVE(red, SW, SE)
  - DRIVE(red, SE, Eex)
- Plan for green car:
  - ARRIVE(green, Nent)
  - DRIVE(green, Nent, NW)
  - DRIVE(green, NW, SW)
  - DRIVE(green, SW, Sex)
- Can these plans interfere with each other?

Example: Intersection with 2 Crossing Cars



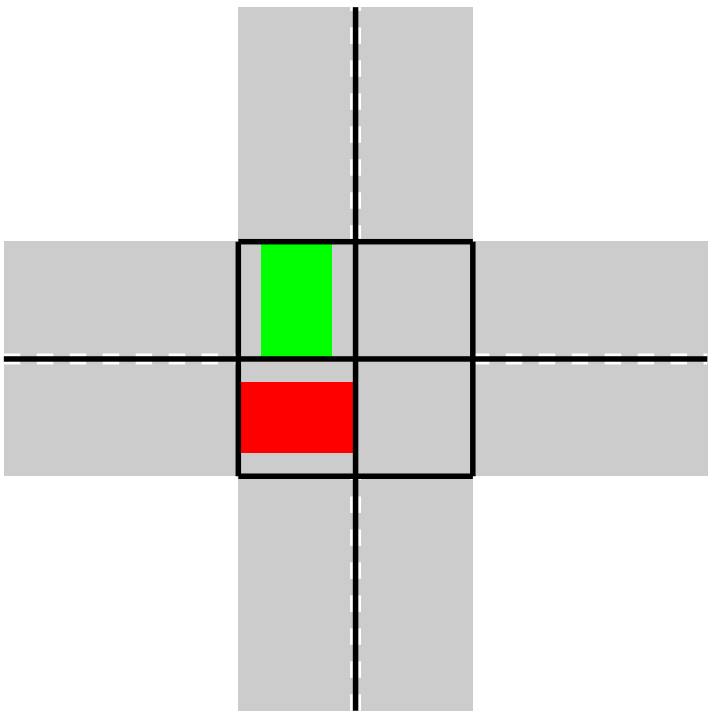
- Plan for red car:
  - ARRIVE(red, Went)
  - DRIVE(red, Went, SW)
  - DRIVE(red, SW, SE)
  - DRIVE(red, SE, Eex)
- Plan for green car:
  - ARRIVE(green, Nent)
  - DRIVE(green, Nent, NW)
  - DRIVE(green, NW, SW)
  - DRIVE(green, SW, Sex)
- Can these plans interfere with each other?

Example: Intersection with 2 Crossing Cars



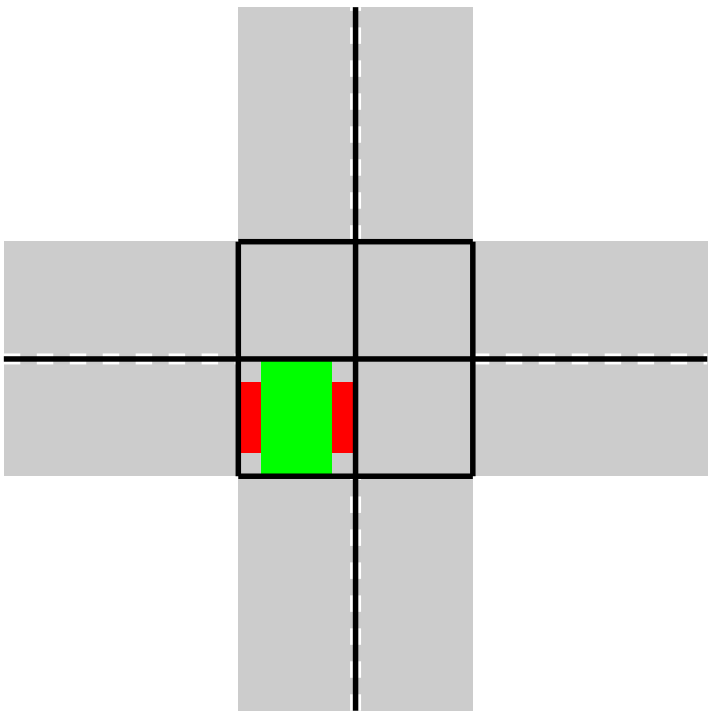
- Plan for red car:
  - ARRIVE(red, Went)
  - DRIVE(red, Went, SW)
  - DRIVE(red, SW, SE)
  - DRIVE(red, SE, Eex)
- Plan for green car:
  - ARRIVE(green, Nent)
  - DRIVE(green, Nent, NW)
  - DRIVE(green, NW, SW)
  - DRIVE(green, SW, Sex)
- Can these plans interfere with each other?

Example: Intersection with 2 Crossing Cars



- Plan for red car:
  - ARRIVE(red, Went)
  - DRIVE(red, Went, SW)
  - DRIVE(red, SW, SE)
  - DRIVE(red, SE, Eex)
- Plan for green car:
  - ARRIVE(green, Nent)
  - DRIVE(green, Nent, NW)
  - DRIVE(green, NW, SW)
  - DRIVE(green, SW, Sex)
- Can these plans interfere with each other?

Example: Intersection with 2 Crossing Cars



- Plan for red car:
  - ARRIVE(red, Went)
  - DRIVE(red, Went, SW)
  - DRIVE(red, SW, SE)
  - DRIVE(red, SE, Eex)
- Plan for green car:
  - ARRIVE(green, Nent)
  - DRIVE(green, Nent, NW)
  - DRIVE(green, NW, SW)
  - DRIVE(green, SW, Sex)
- Can these plans interfere with each other?

Social Laws to the Rescue

- A social law restricts the set of legal actions available to agents
  - Introduced by Tennenholtz, Moses and Shoham more than 25 years ago
  - Motivated by laws in a human society
- What is a good social law?
  - Robustness — social law guarantees agents can achieve their goals
  - Efficiency — plans obeying the social law are as efficient as plans that do not

Encoding Social Laws in MA-STRIPS

- A social law restricts the set of legal actions available to agents
- It might be tempting to say that a social law takes in an MA-STRIPS system, and outputs another MA-STRIPS system where the legal actions have been restricted.
- Such a social law can be described by:
  - The facts it adds or removes,
  - The actions it adds or removes,
  - The preconditions, add effects, or delete effects it adds or removes from each existing action,
  - The facts it adds or removes from the initial state
  - The facts it adds or removes from each agent's goal
- But such social laws will have very limited usefulness

Social Laws with Synchronization

- What is a good social law for the intersection example?
  - Must include some form of yielding
- In our setting, in addition to the above modifications, a social law can specify some action preconditions as waitfor
- Executing actions with waitfor preconditions:
  - When the executive is given an action with waitfor preconditions to execute, it will first monitor these preconditions, and only dispatch the action when they are satisfied
  - In a robotic system, this means we can only wait for what the robot can sense

Social Laws with Synchronization

- What is a good social law for the intersection example?
  - Must include some form of yielding
- In our setting, in addition to the above modifications, a social law can specify some action preconditions as waitfor
- Executing actions with waitfor preconditions:
  - When the executive is given an action with waitfor preconditions to execute, it will first monitor these preconditions, and only dispatch the action when they are satisfied
  - In a robotic system, this means we can only wait for what the robot can sense

Social Laws with Synchronization

- What is a good social law for the intersection example?
  - Must include some form of yielding
- In our setting, in addition to the above modifications, a social law can specify some action preconditions as waitfor
- Executing actions with waitfor preconditions:
  - When the executive is given an action with waitfor preconditions to execute, it will first monitor these preconditions, and only dispatch the action when they are satisfied
  - In a robotic system, this means we can only wait for what the robot can sense

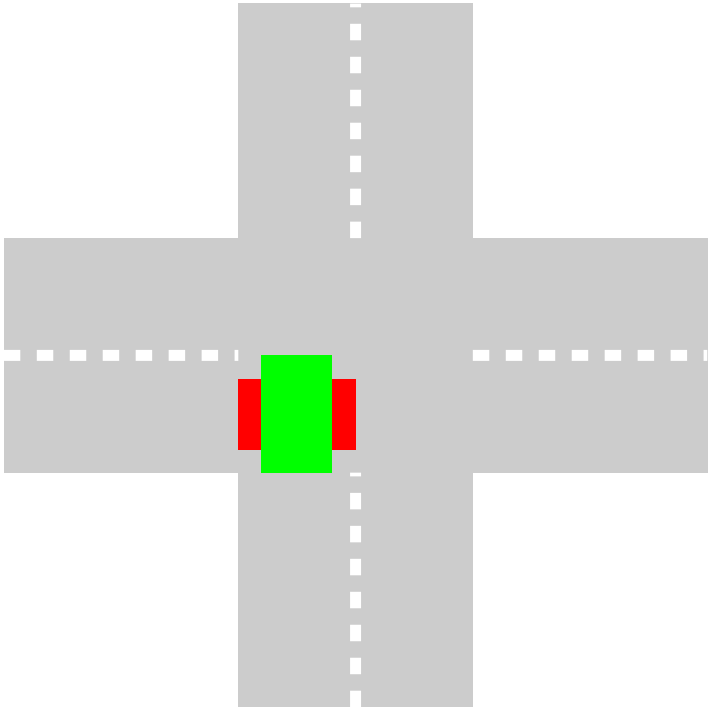
Social Laws with Synchronization

- What is a good social law for the intersection example?
  - Must include some form of yielding
- In our setting, in addition to the above modifications, a social law can specify some action preconditions as waitfor
- Executing actions with waitfor preconditions:
  - When the executive is given an action with waitfor preconditions to execute, it will first monitor these preconditions, and only dispatch the action when they are satisfied
  - In a robotic system, this means we can only wait for what the robot can sense

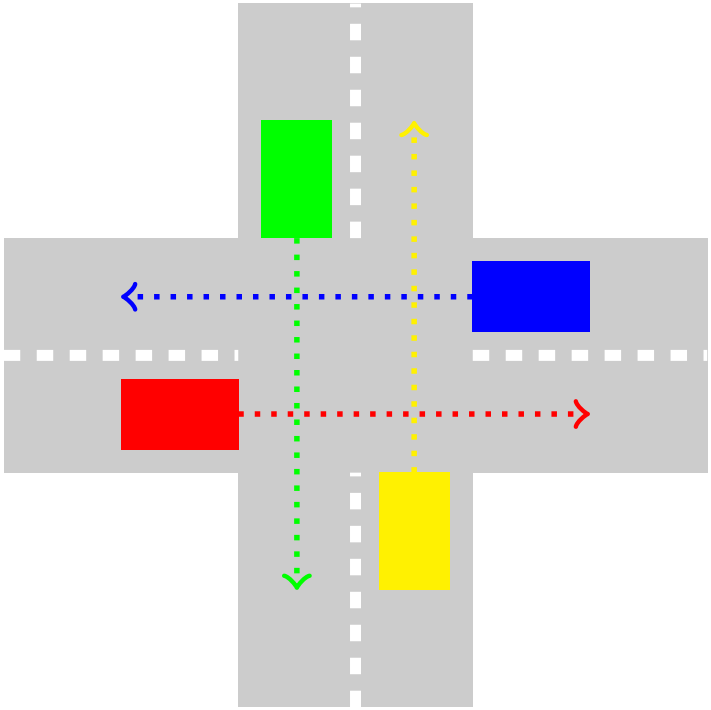
Execution Model

- We are now ready to define our execution model formally:
- Given a multi-agent setting $\Pi = \langle F, \{A_i\}_{i=1}^n, I, \{G_i\}_{i=1}^n \rangle$
- The projection of Π for agent i is the single agent STRIPS planning problem $\Pi_i = \langle F, A_i, I, G_i \rangle$
- Each agent i has some plan π_i which solves Π_i
- Plans are merged by a scheduler:
  - Scheduler chooses which agent acts next
  - The agent which is chosen then executes the next action in its plan
  - Scheduler must choose an agent with a next action whose *waitfor* preconditions are satisfied in the current state
  - No other assumptions about fairness of the scheduler

Two Types of Errors



Collision: agent executes an action whose precondition does not hold



Deadlock: all agents are waiting for some precondition (or finished)

Robustness

- A social law l for multi agent setting $\Pi = \langle F, \{A_i\}_{i=1}^n, I, \{G_i\}_{i=1}^n \rangle$ is robust to:

  rational iff for all agents $i = 1 \ldots n$, for all individual solutions π_i for Π_i, for all possible action sequences π resulting from any arbitrary interleaving of $\{\pi_i\}_{i=1}^n$ which respects *waitfor* preconditions, π achieves $G_1 \cup \ldots \cup G_n$

  adversarial against i iff for all individual solutions π_i for Π_i, for all possible action sequences π resulting from an arbitrary interleaving of π_i which respects *waitfor* preconditions with any valid action sequence of all other agents, π achieves G_i

  adversarial iff it is robust to *adversarial against i* for all $i = 1 \ldots n$

Robustness: Rational vs. Adversarial

- Rational — assumes all agents want to achieve their goal
- Adversarial — assumes nothing about the other agents
- Easy to see:
  - Adversarial robustness $\Rightarrow$ Rational robustness
- Lemma 1:
  - VERIFY-RATIONAL $\geq_p$ VERIFY-ADVERSARIAL
  - Intuition: to verify if agent i is robust to adversarial, treat all other agents as a single "virtual" agent, with an empty goal

Verifying Robustness

- Verify if social law l is rationally robust for multi-agent setting Π by compilation to classical planning:
  - Planner can choose plan for each agent and controls scheduler
    - Scheduler is adversarial
  - Objective: choose plans and find some interleaving of actions (which respects waitfor) which causes an error
  - Caveat: each individual plan must work by itself
    - $n+1$ copies of each fact: local copy for each agent, and global copy
    - Create 3 copies of each action: success, fail, deadlock
    - Each action affects local copy as if it succeeds
    - Success version also succeeds in global copy
    - Fail version achieves *failure* — goal in global copy
    - Deadlock version (waiting for global fact f) must guarantee that f does not hold at the end
  - No solution $\Rightarrow$ rationally robust
  - Exact excruciating details in the paper

Verifying Robustness

- Verify if social law l is rationally robust for multi-agent setting Π by compilation to classical planning:
  - Planner can choose plan for each agent and controls scheduler
    - Scheduler is adversarial
  - Objective: choose plans and find some interleaving of actions (which respects waitfor) which causes an error
  - Caveat: each individual plan must work by itself
    - $n+1$ copies of each fact: local copy for each agent, and global copy
    - Create 3 copies of each action: success, fail, deadlock
    - Each action affects local copy as if it succeeds
    - Success version also succeeds in global copy
    - Fail version achieves *failure* — goal in global copy
    - Deadlock version (waiting for global fact f) must guarantee that f does not hold at the end
  - No solution $\Rightarrow$ rationally robust
  - Exact excruciating details in the paper

Verifying Robustness

- Verify if social law l is rationally robust for multi-agent setting Π by compilation to classical planning:
  - Planner can choose plan for each agent and controls scheduler
    - Scheduler is adversarial
  - Objective: choose plans and find some interleaving of actions (which respects waitfor) which causes an error
  - Caveat: each individual plan must work by itself
    - $n+1$ copies of each fact: local copy for each agent, and global copy
    - Create 3 copies of each action: success, fail, deadlock
    - Each action affects local copy as if it succeeds
    - Success version also succeeds in global copy
    - Fail version achieves *failure* — goal in global copy
    - Deadlock version (waiting for global fact f) must guarantee that f does not hold at the end
  - No solution $\Rightarrow$ rationally robust
  - Exact excruciating details in the paper

Verifying Robustness

- Verify if social law I is rationally robust for multi-agent setting Π by compilation to classical planning:
  - Planner can choose plan for each agent and controls scheduler
    - Scheduler is adversarial
  - Objective: choose plans and find some interleaving of actions (which respects waitfor) which causes an error
  - Caveat: each individual plan must work by itself
    - $n+1$ copies of each fact: local copy for each agent, and global copy
    - Create 3 copies of each action: success, fail, deadlock
    - Each action affects local copy as if it succeeds
    - Success version also succeeds in global copy
    - Fail version achieves *failure* — goal in global copy
    - Deadlock version (waiting for global fact f) must guarantee that f does not hold at the end
  - No solution $\Rightarrow$ rationally robust
  - Exact excruciating details in the paper

Verifying Robustness

- Verify if social law l is rationally robust for multi-agent setting Π by compilation to classical planning:
  - Planner can choose plan for each agent and controls scheduler
    - Scheduler is adversarial
  - Objective: choose plans and find some interleaving of actions (which respects waitfor) which causes an error
  - Caveat: each individual plan must work by itself
    - $n+1$ copies of each fact: local copy for each agent, and global copy
    - Create 3 copies of each action: success, fail, deadlock
    - Each action affects local copy as if it succeeds
    - Success version also succeeds in global copy
    - Fail version achieves *failure* — goal in global copy
    - Deadlock version (waiting for global fact f) must guarantee that f does not hold at the end
  - No solution $\Rightarrow$ rationally robust
  - Exact excruciating details in the paper

Verifying Robustness

- Verify if social law I is rationally robust for multi-agent setting Π by compilation to classical planning:
  - Planner can choose plan for each agent and controls scheduler
    - Scheduler is adversarial
  - Objective: choose plans and find some interleaving of actions (which respects waitfor) which causes an error
  - Caveat: each individual plan must work by itself
    - $n+1$ copies of each fact: local copy for each agent, and global copy
    - Create 3 copies of each action: success, fail, deadlock
    - Each action affects local copy as if it succeeds
    - Success version also succeeds in global copy
    - Fail version achieves *failure* — goal in global copy
    - Deadlock version (waiting for global fact f) must guarantee that f does not hold at the end
  - No solution $\Rightarrow$ rationally robust
  - Exact excruciating details in the paper

Verifying Robustness

- Verify if social law I is rationally robust for multi-agent setting Π by compilation to classical planning:
  - Planner can choose plan for each agent and controls scheduler
    - Scheduler is adversarial
  - Objective: choose plans and find some interleaving of actions (which respects waitfor) which causes an error
  - Caveat: each individual plan must work by itself
    - $n+1$ copies of each fact: local copy for each agent, and global copy
    - Create 3 copies of each action: success, fail, deadlock
    - Each action affects local copy as if it succeeds
    - Success version also succeeds in global copy
    - Fail version achieves *failure* — goal in global copy
    - Deadlock version (waiting for global fact f) must guarantee that f does not hold at the end
  - No solution $\Rightarrow$ rationally robust
  - Exact excruciating details in the paper

Verifying Robustness

- Verify if social law I is rationally robust for multi-agent setting Π by compilation to classical planning:
  - Planner can choose plan for each agent and controls scheduler
    - Scheduler is adversarial
  - Objective: choose plans and find some interleaving of actions (which respects waitfor) which causes an error
  - Caveat: each individual plan must work by itself
    - $n+1$ copies of each fact: local copy for each agent, and global copy
    - Create 3 copies of each action: success, fail, deadlock
    - Each action affects local copy as if it succeeds
    - Success version also succeeds in global copy
    - Fail version achieves *failure* — goal in global copy
    - Deadlock version (waiting for global fact f) must guarantee that f does not hold at the end
  - No solution $\Rightarrow$ rationally robust
  - Exact excruciating details in the paper

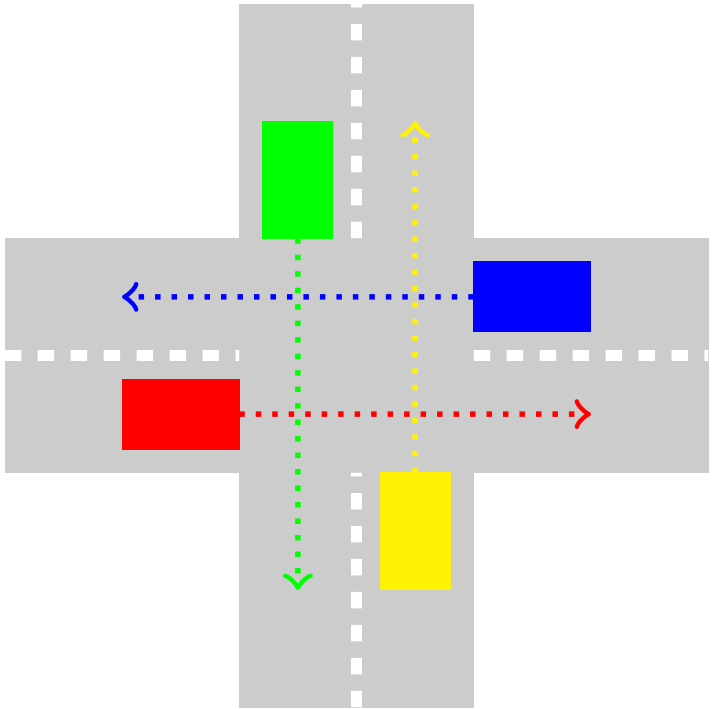
Verifying Robustness

- Verify if social law l is rationally robust for multi-agent setting Π by compilation to classical planning:
  - Planner can choose plan for each agent and controls scheduler
    - Scheduler is adversarial
  - Objective: choose plans and find some interleaving of actions (which respects waitfor) which causes an error
  - Caveat: each individual plan must work by itself
    - $n+1$ copies of each fact: local copy for each agent, and global copy
    - Create 3 copies of each action: success, fail, deadlock
    - Each action affects local copy as if it succeeds
    - Success version also succeeds in global copy
    - Fail version achieves *failure* — goal in global copy
    - Deadlock version (waiting for global fact f) must guarantee that f does not hold at the end
  - No solution $\Rightarrow$ rationally robust
  - Exact excruciating details in the paper

Verifying Robustness

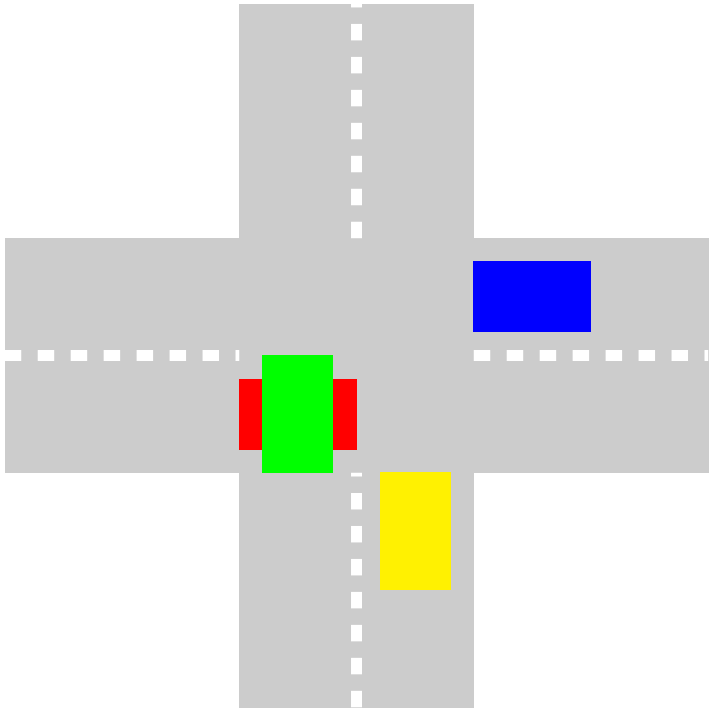
- Verify if social law l is rationally robust for multi-agent setting Π by compilation to classical planning:
  - Planner can choose plan for each agent and controls scheduler
    - Scheduler is adversarial
  - Objective: choose plans and find some interleaving of actions (which respects waitfor) which causes an error
  - Caveat: each individual plan must work by itself
    - $n+1$ copies of each fact: local copy for each agent, and global copy
    - Create 3 copies of each action: success, fail, deadlock
    - Each action affects local copy as if it succeeds
    - Success version also succeeds in global copy
    - Fail version achieves *failure* — goal in global copy
    - Deadlock version (waiting for global fact f) must guarantee that f does not hold at the end
  - No solution $\Rightarrow$ rationally robust
  - Exact excruciating details in the paper

Intersection Example 1



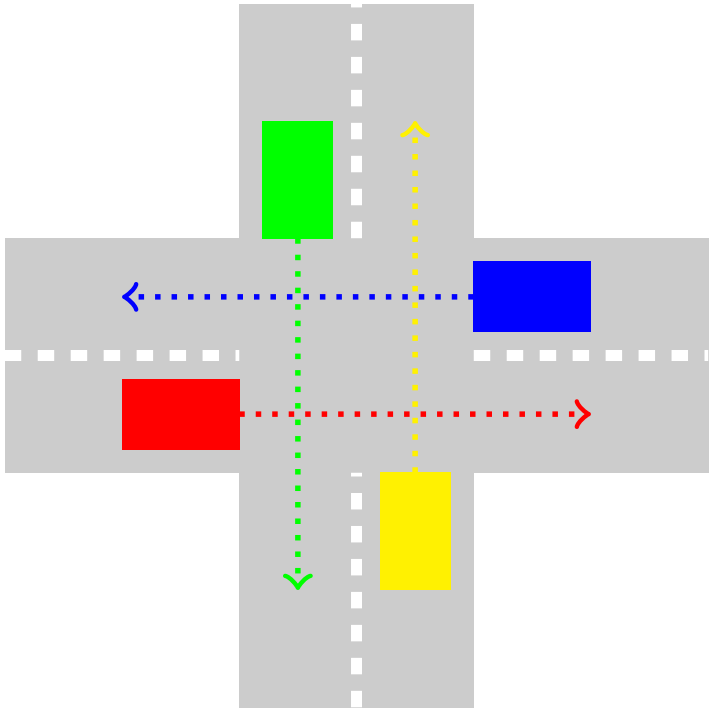
- Social law: empty
- Verification result: collision

Intersection Example 1



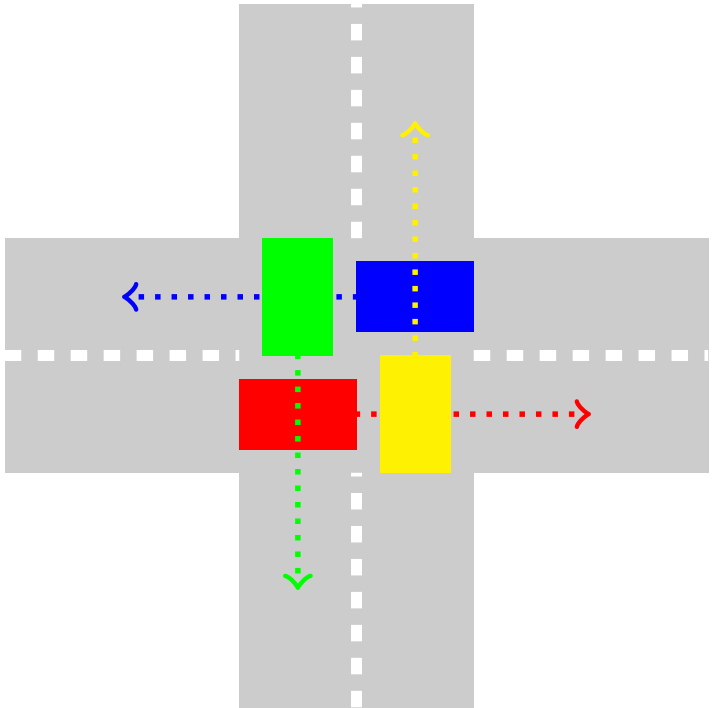
- Social law: empty
- Verification result: collision

Intersection Example 2



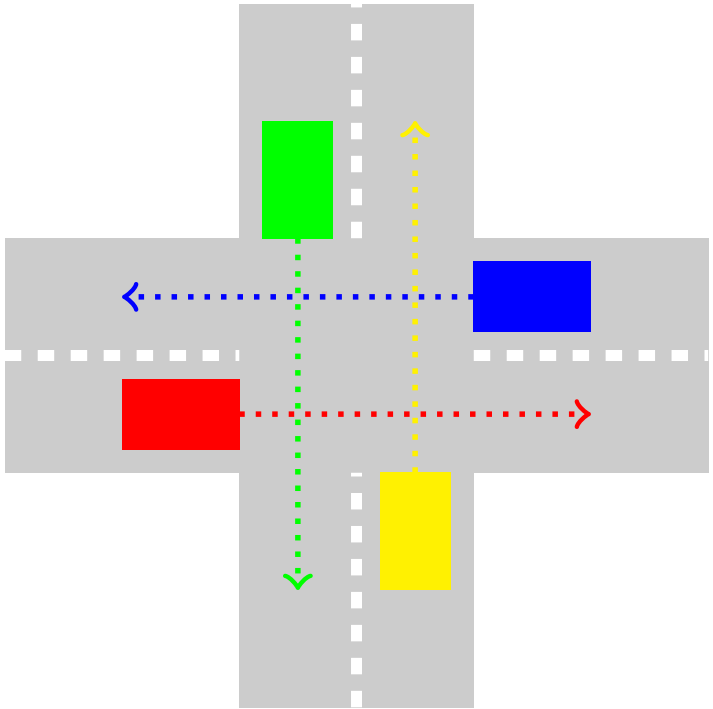
- Social law: do not drive into non-clear location
- Verification result: deadlock

Intersection Example 2



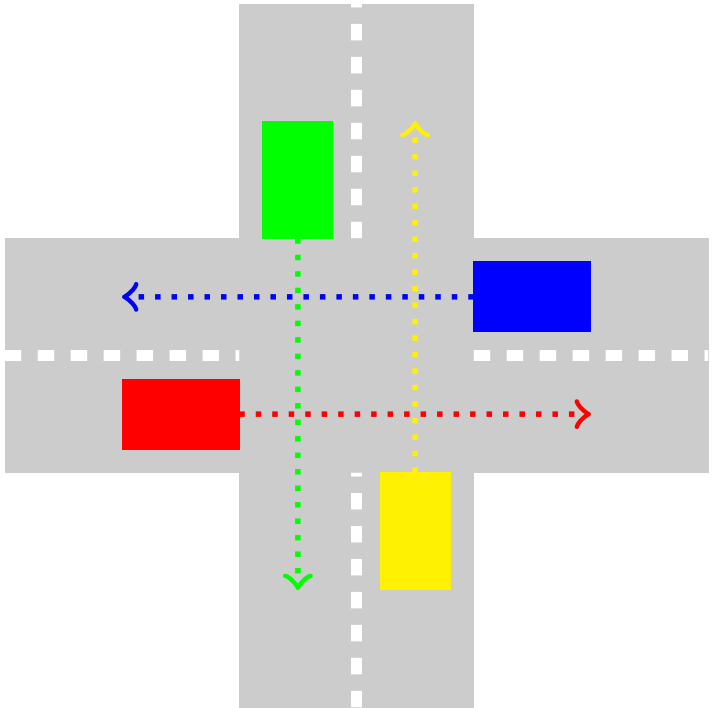
- Social law: do not drive into non-clear location
- Verification result: deadlock

Intersection Example 3



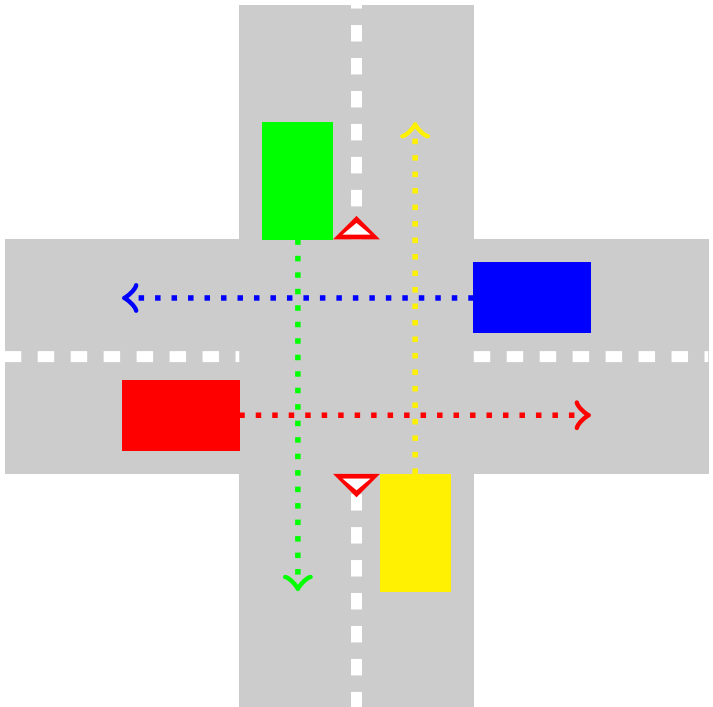
- Social law: yield to car coming from right
- Verification result: deadlock

Intersection Example 3



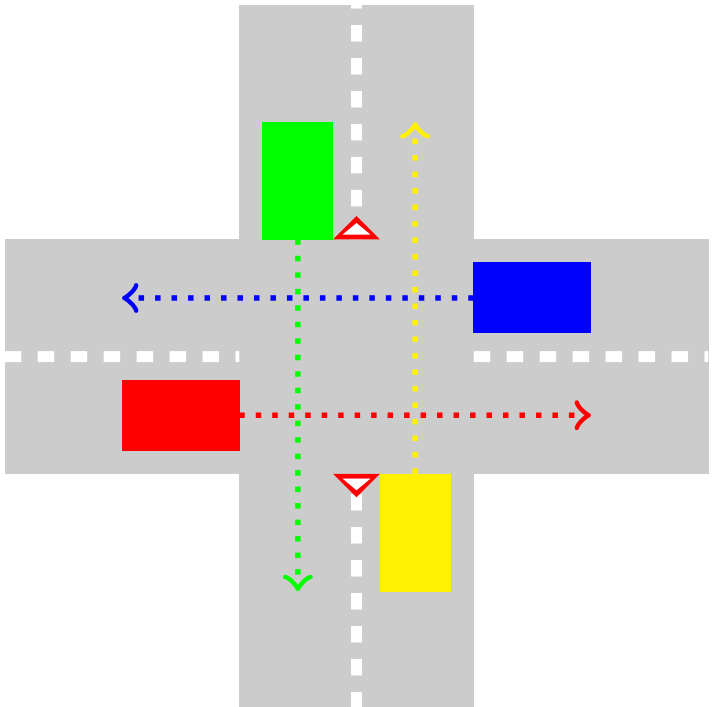
- Social law: yield to car coming from right
- Verification result: deadlock

Intersection Example 4



- Social law: cars coming from north or south yield to cars coming from their right
- Verification result: robust

Intersection Example 4



- Social law: cars coming from north or south yield to cars coming from their right
- Verification result: robust

CoDMAP Benchmarks

- We also used CoDMAP benchmarks
- To convert them from cooperative planning to coordination problems, we need to have a goal for each agent,
- Assign each goal fact to an agent
  - If goal fact mentions an agent *i* as first argument, assign it to *i*
  - Otherwise, assign it to a random agent
- We only kept instances for which all agents could solve their individual problem
- Used FF planner with 5 minute timeout
- No surprise: very few are rationally robust

CoDMAP Benchmarks: Planning Time Results

BLOCKSWORLD	
Instance	Time (s)
9-0	0.1
9-1	0.09
9-2	0.11
10-0	0.1
10-1	0.08
10-2	0.09
11-0	0.17
11-1	0.18
11-2	0.11
12-0	0.18
12-1	0.26
13-0	0.32
13-1	0.48
14-0	0.44
14-1	0.19
15-0	4.17
15-1	1.83
16-1	1.83
16-2	0.43
17-0	8.23

DRIVERLOG	
Instance	Time (s)
pfile1	0
pfile2	0
pfile3	0
pfile4	0
pfile5	0
pfile6	0
pfile7	0
pfile8	0
pfile9	0
pfile10	0
pfile11	0.01
pfile12	0.01
pfile13	0.02
pfile14	0.04
pfile15	0.17
pfile16	1.25
pfile17	28.63
pfile18	23.53
pfile19	88.06
pfile20	111.54

ZENOTRAVEL	
Instance	Time (s)
pfile3	0
pfile4	0
pfile5	0
pfile6	0
pfile7	0
pfile8	0
pfile9	0
pfile10	0.01
pfile12	0
pfile13	0.04
pfile14	0.1
pfile15	0.22
pfile16	0.12
pfile17	0.61
pfile18	1.04
pfile19	5.26
pfile20	2.77
pfile21	3.85
pfile22	5.46
pfile23	15.06

SATELLITES	
Instance	Time (s)
p05	0.11
p07	0.24
p21	243.38
p24	—
p25	—

SOKOBAN	
Instance	Time (s)
p06-1	—

Summary

- Contributions
  - Connected social laws to multi-agent planning
  - Defined notions of robustness
  - Showed how to check robustness by compilation to classical planning, and showed this works quickly on benchmarks
- Future Work
  - Automatically synthesize social laws
  - Apply to robots

Thank You

Questions?