Why is it so hard to make self-driving cars? (Trustworthy Autonomous Systems)

Waymo
March 19, 2021

Joseph Sifakis

Autonomous systems – Main Characteristics

Autonomous systems are essential for reaching the Industrial IoT vision.

- They emerge from the needs to further automate existing organizations by progressive and incremental replacement of human operators by autonomous agents.

- They are very different from game-playing robots or intelligent personal assistants.

- They are often critical and should exhibit "broad intelligence" by handling knowledge in order to
  - Manage dynamically changing sets of possibly conflicting goals – this reflects the trend of transitioning from "narrow" or "weak" AI to "strong" or "general" AI.
  - Cope with uncertainty of complex, unpredictable cyber physical environments.
  - Harmoniously collaborate with human agents e.g. "symbiotic" autonomy.

- Two different technical avenues both falling short of the autonomy challenge:
  - traditional model-based critical systems engineering, successfully applied to aircraft and production systems, proves to be inadequate.
  - industrial end-to-end AI-enabled solutions currently available e.g. NVIDIA's PilotNet, fail to provide the required strong trustworthiness guarantees.

Autonomous systems – For a New Scientific and Engineering Foundation

	Systems Engineering is facing a huge gap, moving				
FROM	Small size	Centralized	Automated	Predictable Envt	Elicitable Specs
TO	Complex	Decentralized	Autonomous	Unpredictable Envt	Non-elicitable Specs

We need a new scientific and engineering foundation that cannot be obtained by simply combining existing results developed for more than two decades and focusing mainly on SW systems e.g. Autonomic computing, Adaptive systems, Autonomous Agent Systems and brings answers to the following problems:

1. Bridging the gap between Automation and Autonomy
    - What are the technical solutions for enhancing a system's autonomy?
    - For each enhancement, what are the implied technical difficulties and risks?

2. Relate system trustworthiness to knowledge truthfulness about the developed system.

3. Move from traditional system design to "hybrid" design seeking trade offs between trustworthiness of model-based and performance of data-based approaches.

4. Develop new theory for the validation of autonomous systems based on simulation and testing - allowing to provide conclusive trustworthiness evidence.

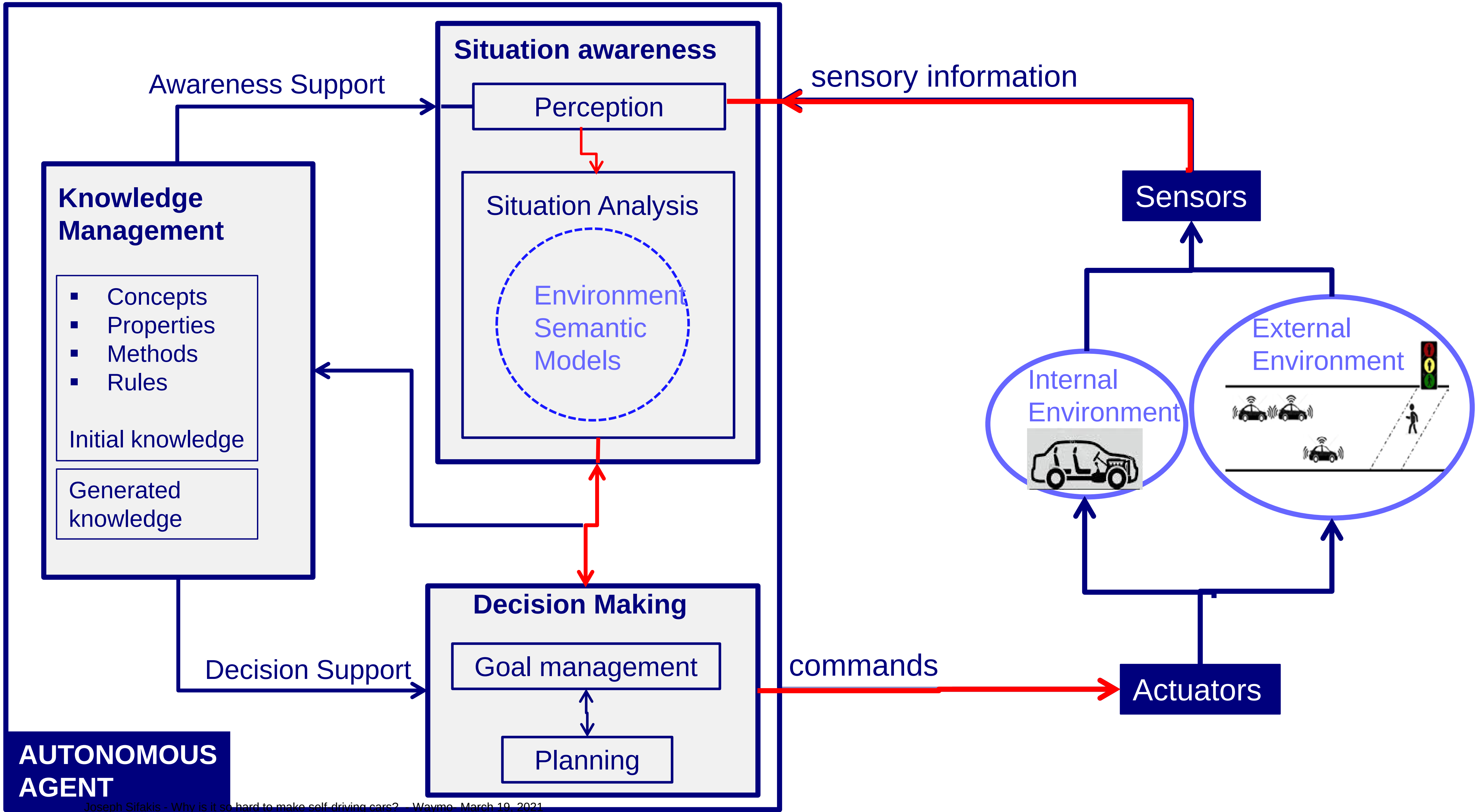
O
V
E
R
V
I
E
W

- Why is it so hard?

- Toward Trustworthy Autonomy
  - Trustworthy Autopilot Design
  - When Self-driving Cars are Safe Enough?

- Discussion

Why is it so hard? – Autonomous Agent Architecture



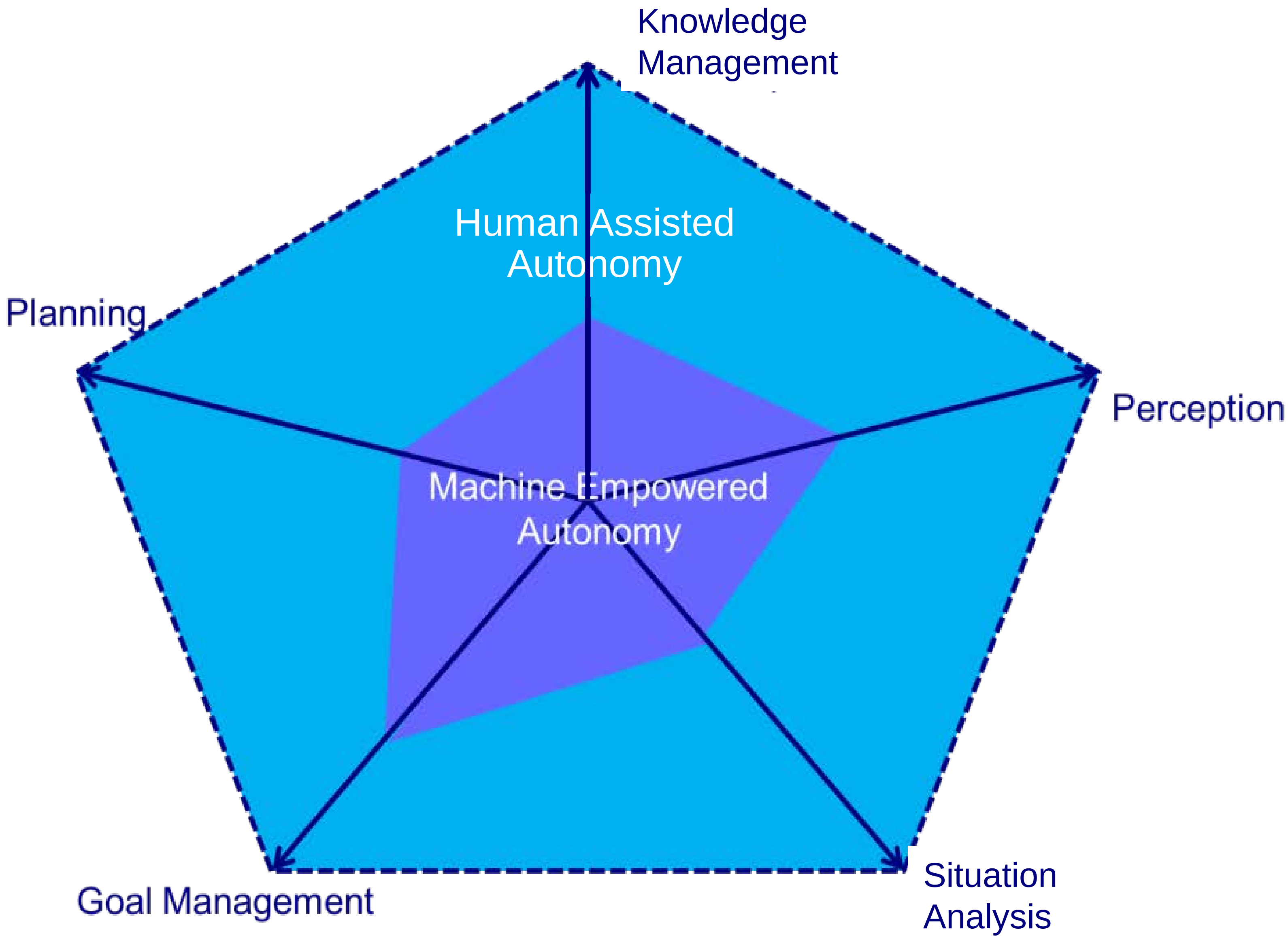
Why is it so hard? – SAE Autonomy Levels

	SAE AUTONOMY LEVELS
Level 0	No automation
Level 1	Driver assistance required
	The driver still needs to maintain full situational awareness and control of the vehicle e.g. cruise control.
Level 2	Partial automation options available
	Autopilot manages both speed and steering under certain conditions, e.g. highway driving.
Level 3	Supervised Autonomy
	The car, rather than the driver, takes over actively monitoring the environment when the system is engaged. However, human drivers must be prepared to respond to a "request to intervene"
Level 4	Geofenced autonomy
	Self driving is supported only in limited areas or under special circumstances, like traffic jams
Level 5	Full autonomy
	No human intervention is required e.g. a robotic taxi

AUTOMATION (ADAS)

AUTONOMY

Why is it so hard? – Symbiotic Autonomy



- Study modes of interaction between machines and humans to accomplish complex tasks/missions
- For each mode and function, determine the division of roles in terms of responsibility/accountability and reactivity/proactivity
- Implement each mode as a protocol amenable to formalization and verification

- Why is it so hard?

- Toward Trustworthy Autonomy
  - Trustworthy Autopilot Design
  - When Self-driving Cars are Safe Enough?

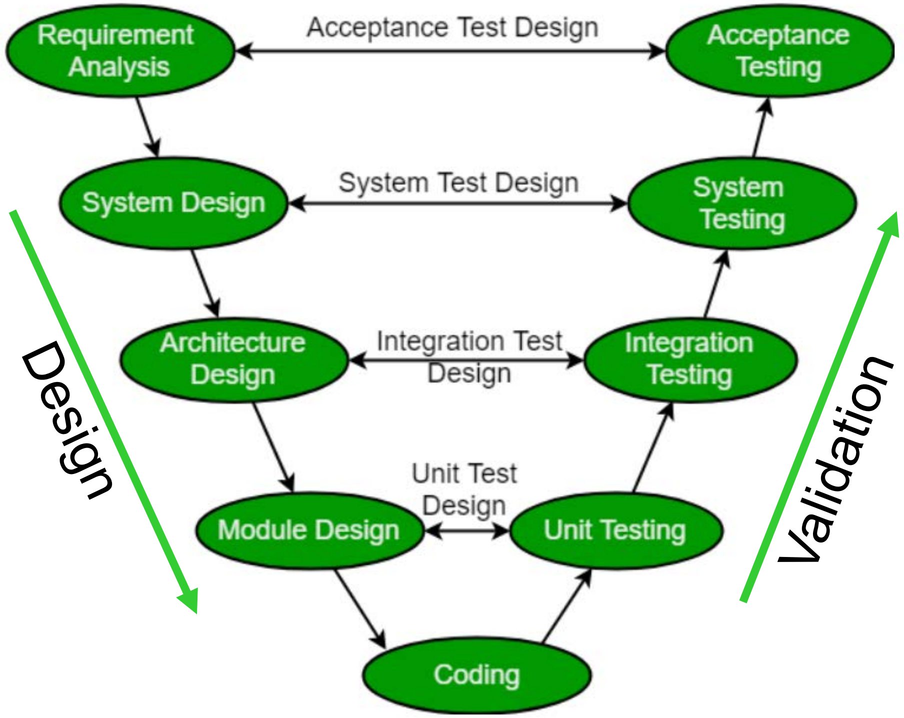
- Discussion

OVERVIEW

Autopilot Design – Critical Systems Engineering Limitations

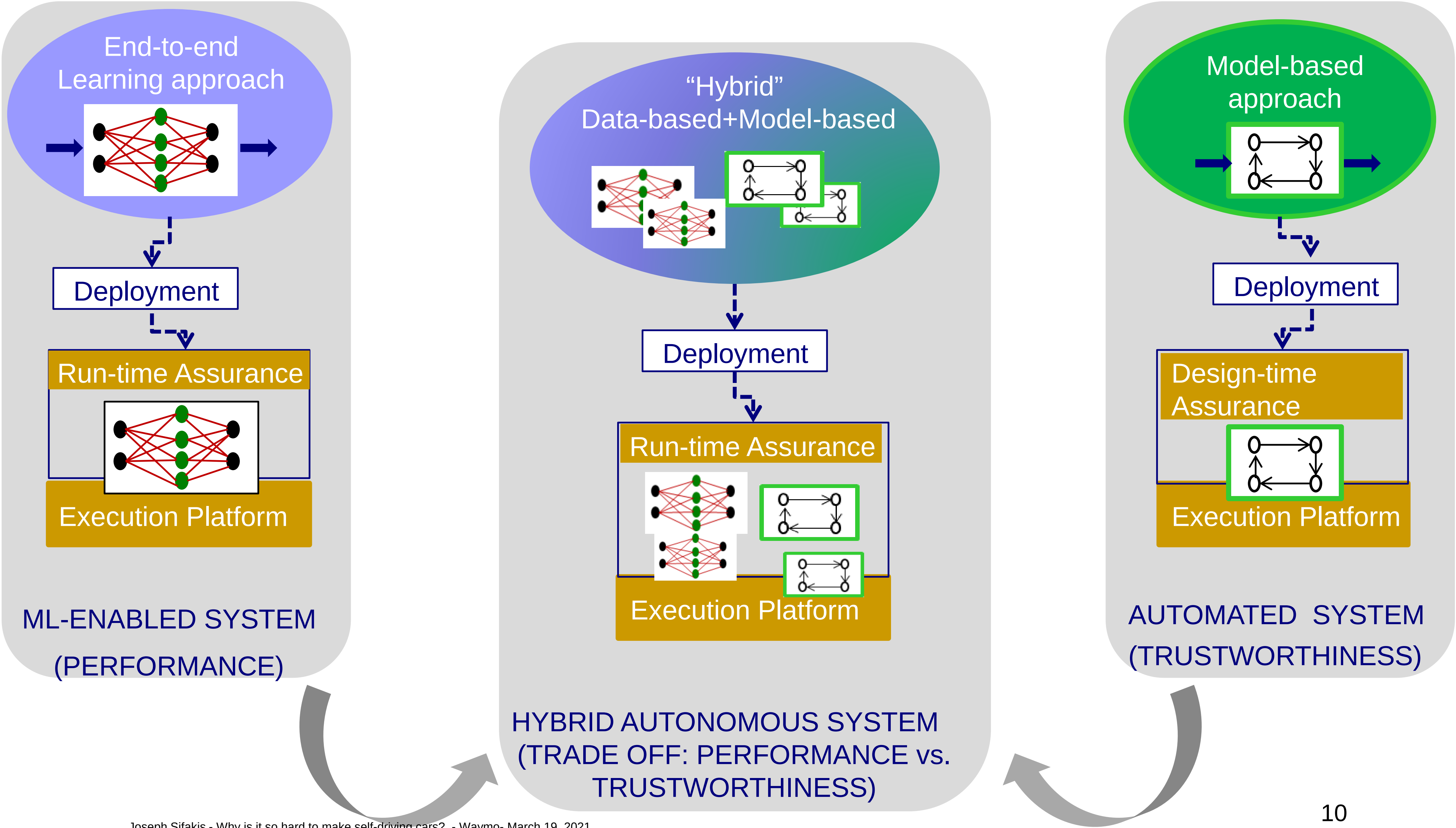
Critical systems design flows follow model-based prescriptive frameworks recommended by standards e.g. ISO26262

- Assume that system development is top-down and validation is bottom-up.
- Assume that all requirements are initially known, can be clearly formulated and understood.
- Consider that global system requirements can be broken down into requirements satisfied by system components.
- Focus on providing model-based conclusive evidence that the system is safe e.g. 10^{-9} failures per hour of flight



- The model-based paradigm is defeated by the overwhelming complexity and diversity of autonomous systems
- This explains the adoption by industry of end-to-end machine-learning-enabled techniques which however preclude conclusive safety guarantees

Autopilot Design –Taking the Best from Each



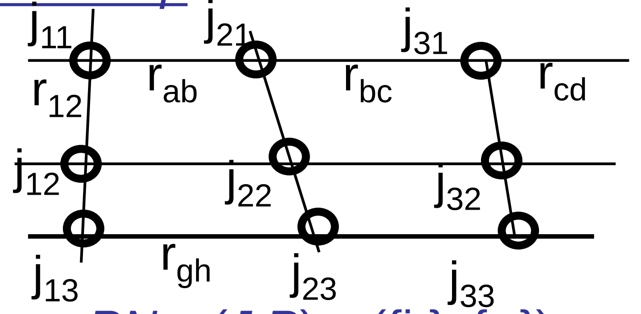
Autopilot Design – Principles

- Hybrid approach:
  - Combine ML (for perception only) with a full-fledged model-based decision process
  - Rely on a functional characterization of autonomy distinguishing between main functions
  - Provide strong trustworthiness guarantees
- Hierarchical decomposition
  - Avoid on line plan generation and consider instead that autonomous behavior can be implemented by integrating a set of (precomputed) maneuver protocols – strong assumption to be validated experimentally !!
  - Each protocol corresponds to a “driving mode” or “driving skills” determined be hierarchical decomposition of the function of driving
  - Protocols are needed to deal with
    1. Different types of junctions such as roundabouts, splits and joins, entrance and exit adaptors, intersections, crossings (toll station pedestrian crossing, railroad crossing);
    2. Different types maneuvers for different contexts e.g. overtaking, platooning, parking.

Autopilot Design – Hierarchical Semantic Model

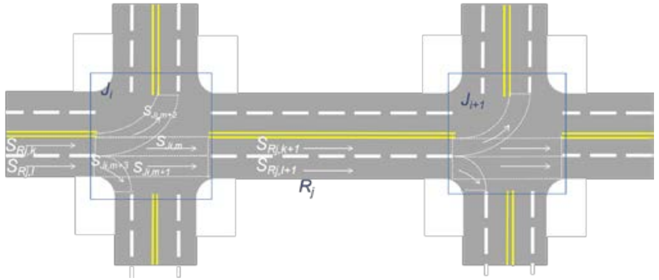
Maps

Road map



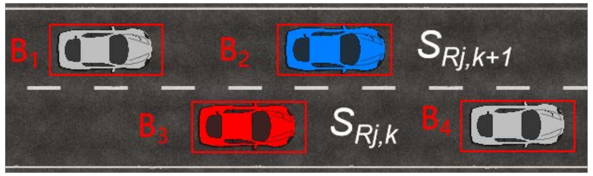
$RN = (J,R) = (\{j_k\}, \{r_s\})$

Lane map



$SN = (S,E)$

Synthesized local map



Hierarchical Autopilot

Mission Planning - Level 4

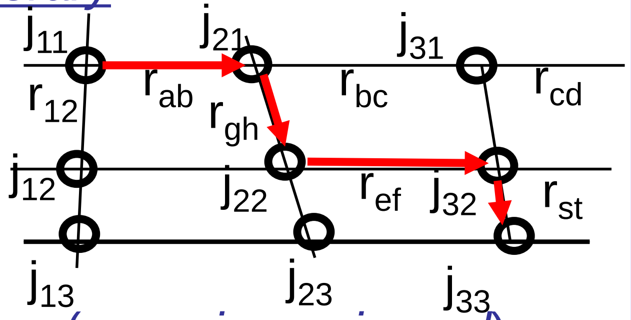
Path Planning - Level 3

Maneuver Planning & protocols - Level 2

Trajectory Planning - Level 1

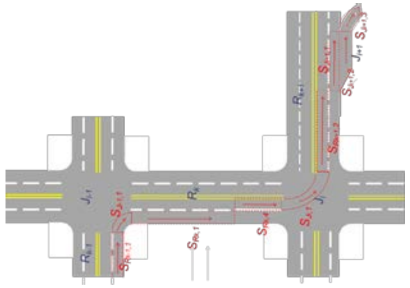
Goals

Itinerary



$(o, r_{ab}, j_{21}, \ldots\ j_k, r_k, d)$

Path

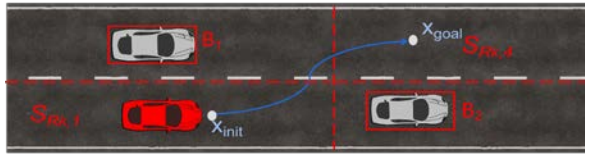


$(s_{Rk\text{-}1,1}, s_{Ji\text{-}1,1}, s_{Rk,1}, s_{Rk,4}, s_{Ji,1}, s_{Rk+1,2})$

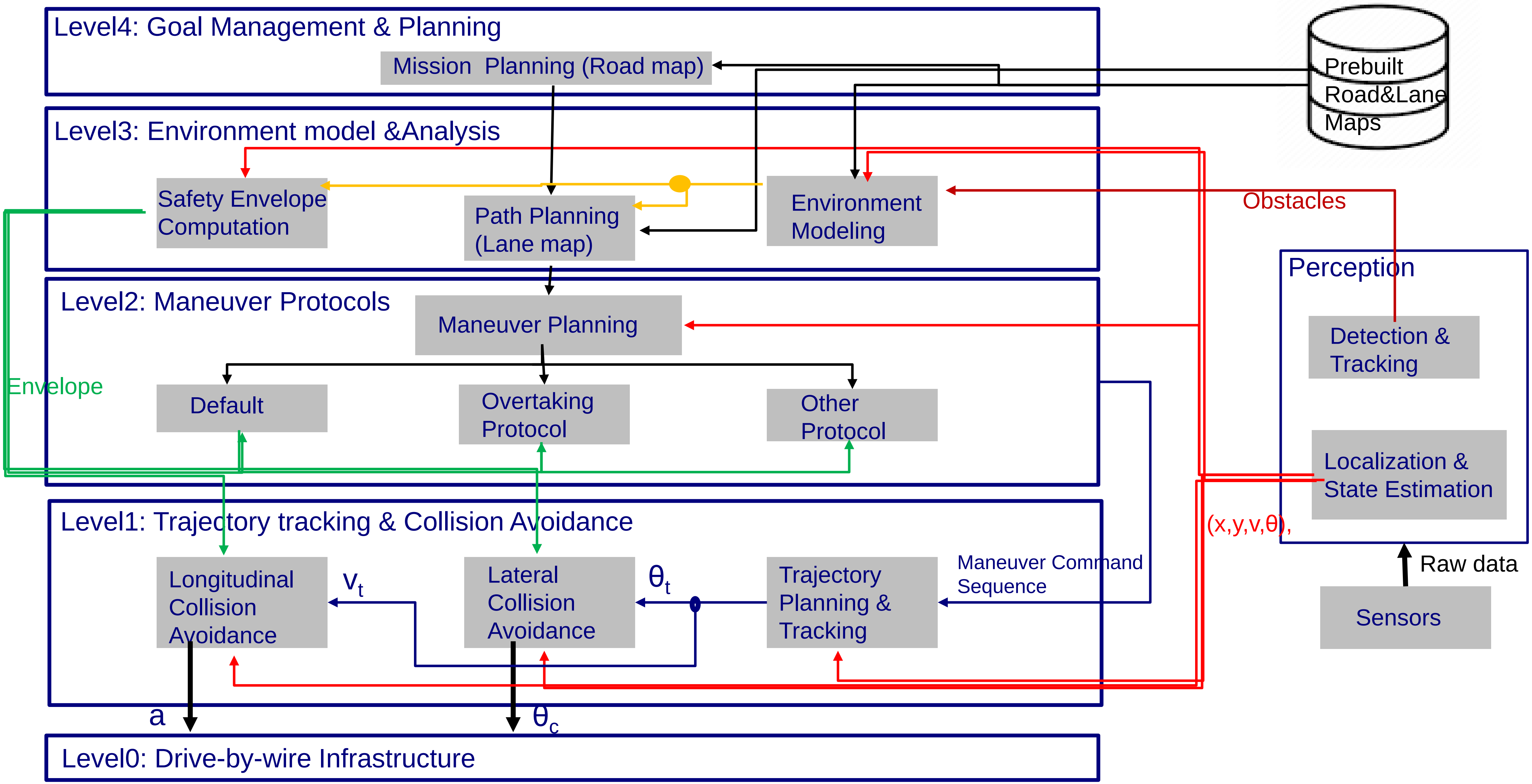
Maneuver sequences

$(lane\_change, s_{Rk,1}, s_{Rk,4})$

Trajectory



Autopilot Design – Hierarchical Architecture



Autopilot Design – Risk Management

RISK ANALYSIS

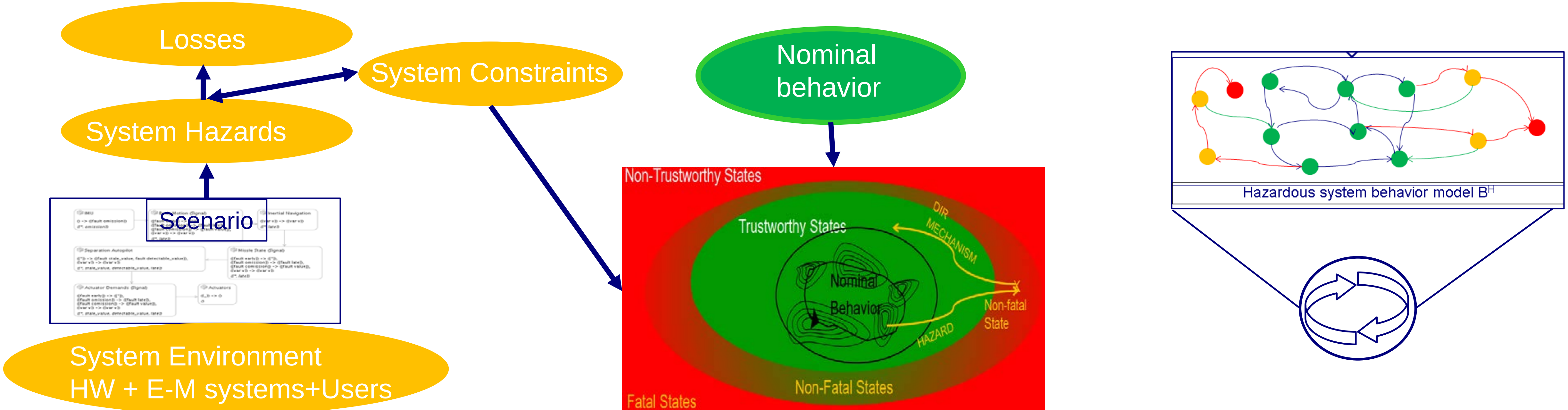
(analyze the causes of losses and estimate their likelihood in terms of probabilistic scenarios)

RISK MITIGATION

(design and implement for each hazard a corresponding diagnostics/detection/mitigation mechanism)

RISK EVALUATION

(estimate the probability that a system meets a set of properties characterizing its correctness)



Autopilot Design – Risk Analysis Levels

We can combine the different techniques depending on the type of system and the desired degree of analysis.

1. Global reasoning at system level e.g. traditional approaches with FTA , failure modes and effects criticality analysis (FMECA), event tree analysis (ETA),
   - determining AND/OR causality relations between hazards e.g. Fault Tree Analysis
   - components are just functions with no behavior – the can be OK or KO with some probability

2. Analysis based on a hierarchical control structure e.g. STPA
   - conceptual model independent from the actual implementation
   - local reasoning at each component level

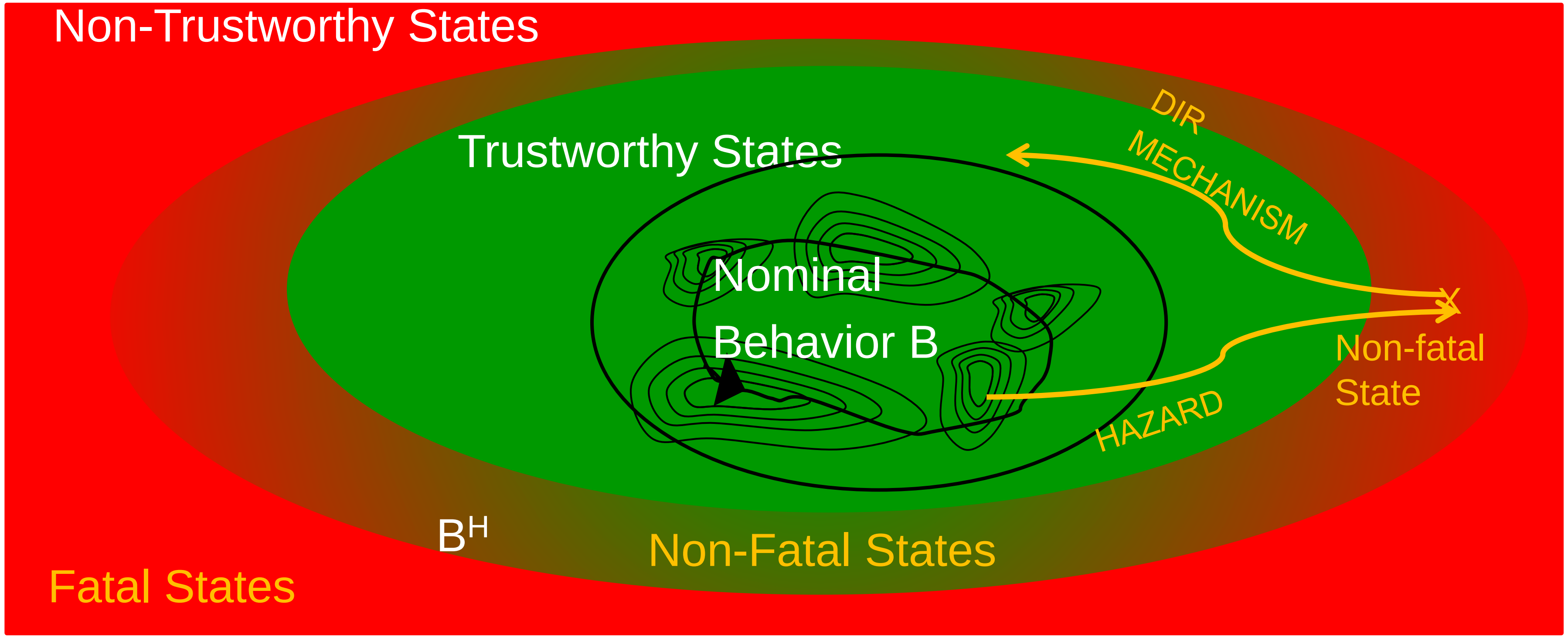
3. Data flow analysis with known I/O behavior of components with respect to predefined global types – global causality analysis e,g. FPTC (Fault Propagation and Transformation Calculus)

4. Behavioral analysis of a system component-based model with nominal and hazardous states e.g. BIP with statistical model checking, AADL, AltaRica,
   - Can be combined with Risk Analysis techniques to take account only manifestations of of low level hazards
   - Can be adapted to cover all aspects of Risk Analysis, FDIR analysis, Reliability Evaluation

Autopilot Design – Risk Mitigation: FDIR

Problem:

1) develop a model B^H connecting nominal behavior B to detectable hazards – not all hazards of Risk Analysis will appear at model level;
2) design and implement failure detection isolation and recovery mechanisms for classes of hazards.



- Static risk mitigation techniques cannot be fully applied to autonomous systems:
  - Overwhelming environment complexity and lack of predictability;
  - Use of "black-box" ML-enabled components.

Autopilot Design – Risk Mitigation: Failure Typology

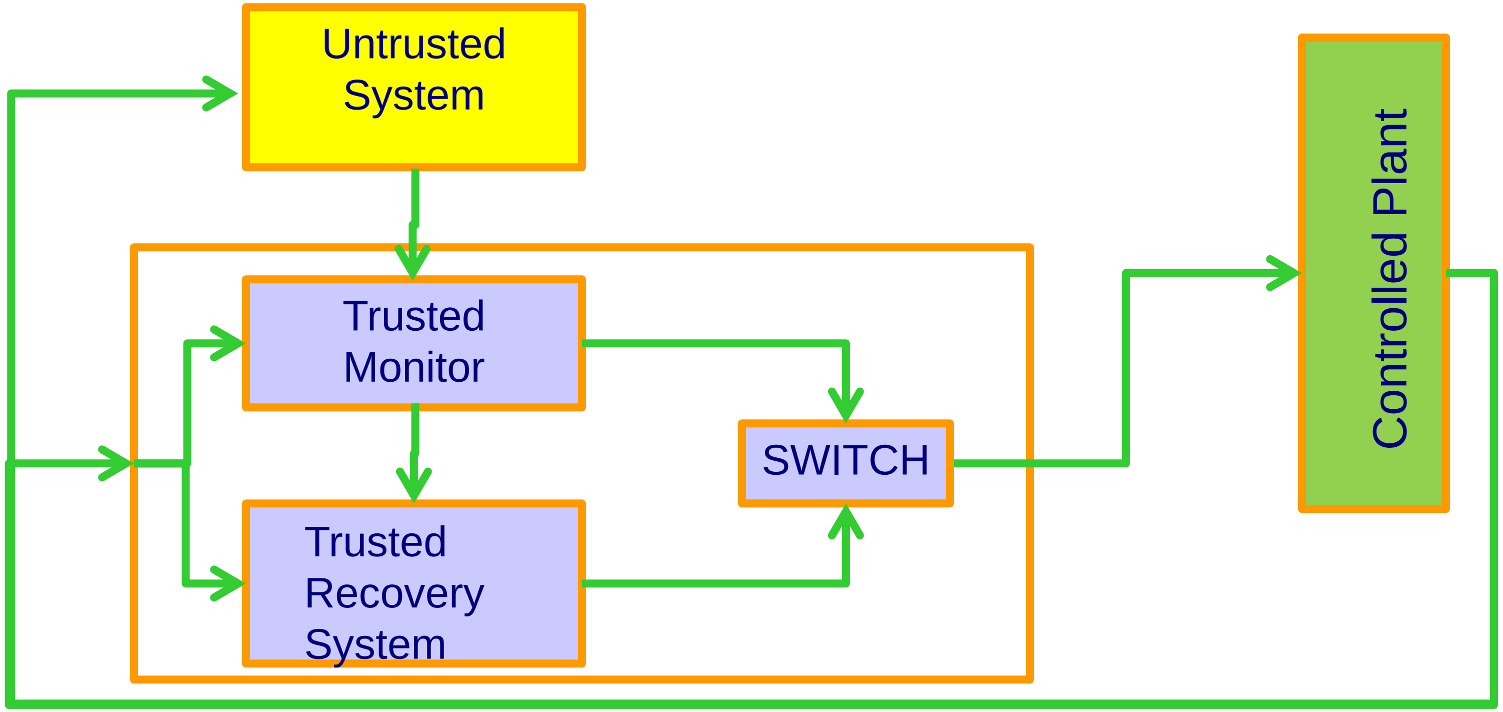
1	Vehicle Failure
2	Control Loss With Prior Vehicle Action
3	Control Loss Without Prior Vehicle Action
4	Running Red Light
5	Running Stop Sign
6	Road Edge Departure With Prior Vehicle Maneuver
7	Road Edge Departure Without Prior Vehicle Maneuver
8	Road Edge Departure While Backing Up
9	Animal Crash With Prior Vehicle Maneuver
10	Animal Crash Without Prior Vehicle Maneuver
11	Pedestrian Crash With Prior Vehicle Maneuver
12	Pedestrian Crash Without Prior Vehicle Maneuver
13	Pedalcyclist Crash With Prior Vehicle Maneuver
14	Pedalcyclist Crash Without Prior Vehicle Maneuver
15	Backing Up Into Another Vehicle
16	Vehicle(s) Turning – Same Direction
17	Vehicle(s) Parking – Same Direction
18	Vehicle(s) Changing Lanes – Same Direction
19	Vehicle(s) Drifting – Same Direction
20	Vehicle(s) Making a Maneuver – Opposite Direction
23	Lead Vehicle Accelerating
24	Lead Vehicle Moving at Lower Constant Speed
25	Lead Vehicle Decelerating
26	Lead Vehicle Stopped
27	Left Turn Across Path From Opposite Directions at Signalized Junctions
28	Vehicle Turning Right at Signalized Junctions
29	Left Turn Across Path From Opposite Directions at Non-Signalized Junctions
30	Straight Crossing Paths at Non-Signalized Junctions
31	Vehicle(s) Turning at Non-Signalized Junctions
32	Evasive Action With Prior Vehicle Maneuver
33	Evasive Action Without Prior Vehicle Maneuver
34	Non-Collision Incident
35	Object Crash With Prior Vehicle Maneuver
36	Object Crash Without Prior Vehicle Maneuver
37	Other

Pre-crash failure typology covering 99.4% of light-vehicle crashes for 5,942,000 cases.
Source: Pre-Crash Scenario Typology for Crash Avoidance Research, DOT HS 810 767, April 2017.

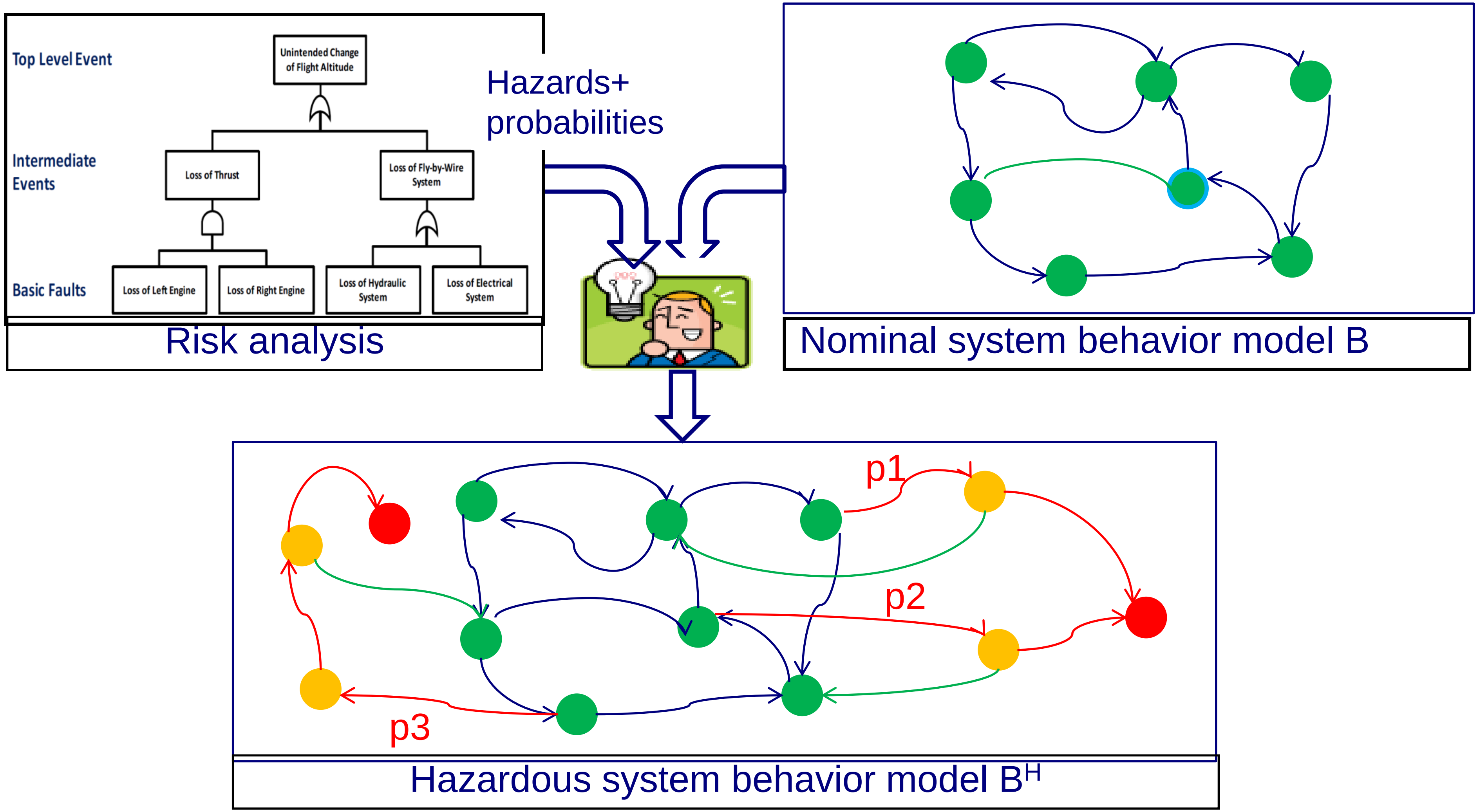
Autopilot Design – Risk Mitigation: Run-time Assurance

- Idea: replace static DIR analysis at design time by run-time monitoring and recovery

- The architecture integrates the Untrusted System and the Run-time Assurance System consisting of a Trusted Monitor and a Trusted Recovery System.
  - The Trusted Monitor detects discrepancies from the nominal behaviour;
  - The Trusted Recovery System is a trusted simplified version of the untrusted system that is able to provide some minimal service when some hazard occurs;
  - The Switch provides the output of the Untrusted System as long as no hazard is detected; otherwise, the Trusted Recovery System takes over.



Autopilot Design – Risk Evaluation: Behavioral Aproach



- Estimating "likelihood" to reach bad states
- Strategies for avoiding bad states
- Design space exploration of parametric models for risk minimization

- Why is it so hard?

- Toward Trustworthy Autonomy
    - Trustworthy Autopilot Design
    - When Self-driving Cars are Safe Enough?

- Discussion

OVERVIEW

When Self-driving Cars are Safe Enough? – The “Miles Argument”

Waymo has now driven 10 billion autonomous miles in simulation

Darrell Etherington @etherington / 11:17 pm CEST • July 10, 2019

Comment

It is possible to compute (*) the number of miles needed to drive without accident

- for given rate of accident type per mile driven
- for given confidence level

Accident type	Miles needed at confidence level	
	95%	50%
With fatality	291 million	67 million
With personal injury	5.4 million	1.2 million
Crash (=police-reported accident)	1.5 million	344.000
Any accident (estimated)	745.000	172.000

(*) https://www.rand.org/content/dam/rand/pubs/research_reports/RR1400/RR1478/RAND_RR1478.pdf, Rand report April 2016

The “miles argument” should be substantiated with “model-based” evidence

When Self-driving Cars are Safe Enough? – Gaps in the State of the Art

- Global system validation is achievable only through simulation and testing, which nonetheless should take into account the following:

  - All the simulated miles are not equally efficacious - how a simulated mile is related to a “real mile” ?
  - Any technically sound safety evaluation should be model-based e.g. relies on criteria defined on an implicit or an explicit system model.
  - We need evidence that simulation covers a good deal of the many and diverse situations e.g. different types of roads, traffic conditions, weather conditions etc.

- We need validation theory based on the semantic simulation model.

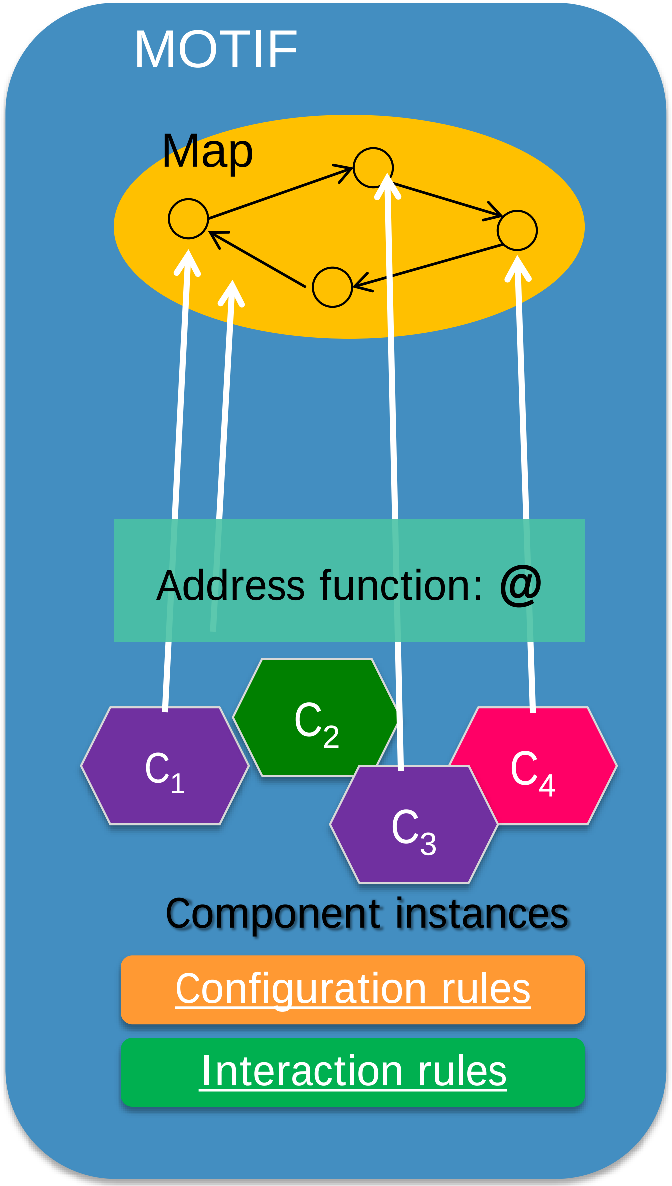
  - Notions of coverage measuring the degree to which relevant system configurations have been explored, as for structural testing of software systems.
  - Scenario description languages to explore/detect corner cases and high risk situations, exactly as for functional testing software systems.
  - Verdicts and diagnostics about the relationship between failures and various risk factors (road structure, congestion level, weather) and violations of traffic regulations.

When Self-driving Cars are Safe Enough? – Simulation Key Issues

- Whatever design approach is taken, simulation is of paramount importance for validation – and raises a large variety of problems from purely technical to theoretical ones.
- Not only the appearance should be realistic but also it should be real: the execution mechanism should rely on a semantic model of the environment consistent with laws of Geometry and Physics.
- Note that realism and consistency with reality are hard to reconcile - simulation environments built on top of game engines lack semantic awareness.

1. Realism: agent behavior and environment look real in a way that is accurate or true to life.
2. Modeling: expressive modeling language e.g. DSL for the component-based description of mobile agents and their dynamic coordination.
3. Semantic awareness: the simulated system dynamics is rooted in transition system semantics.
   - Notion of state allowing controllability and repeatability of experiments.
   - Notion of execution sequence distinguishing between controllable and uncontrollable actions
   - Multiscale multigrain modeling of time scales and of their correlation with space scales
4. Performance: run-time infrastructure federating simulation engines e.g. HLA, FMI

When Self-driving Cars are Safe Enough? – Modeling: DR-BIP



DR-BIP (Dynamic Reconfigurable BIP) is a DSL that supports a modeling methodology relying on the following principles:

- ❑ A system is a set of (architecture) motifs

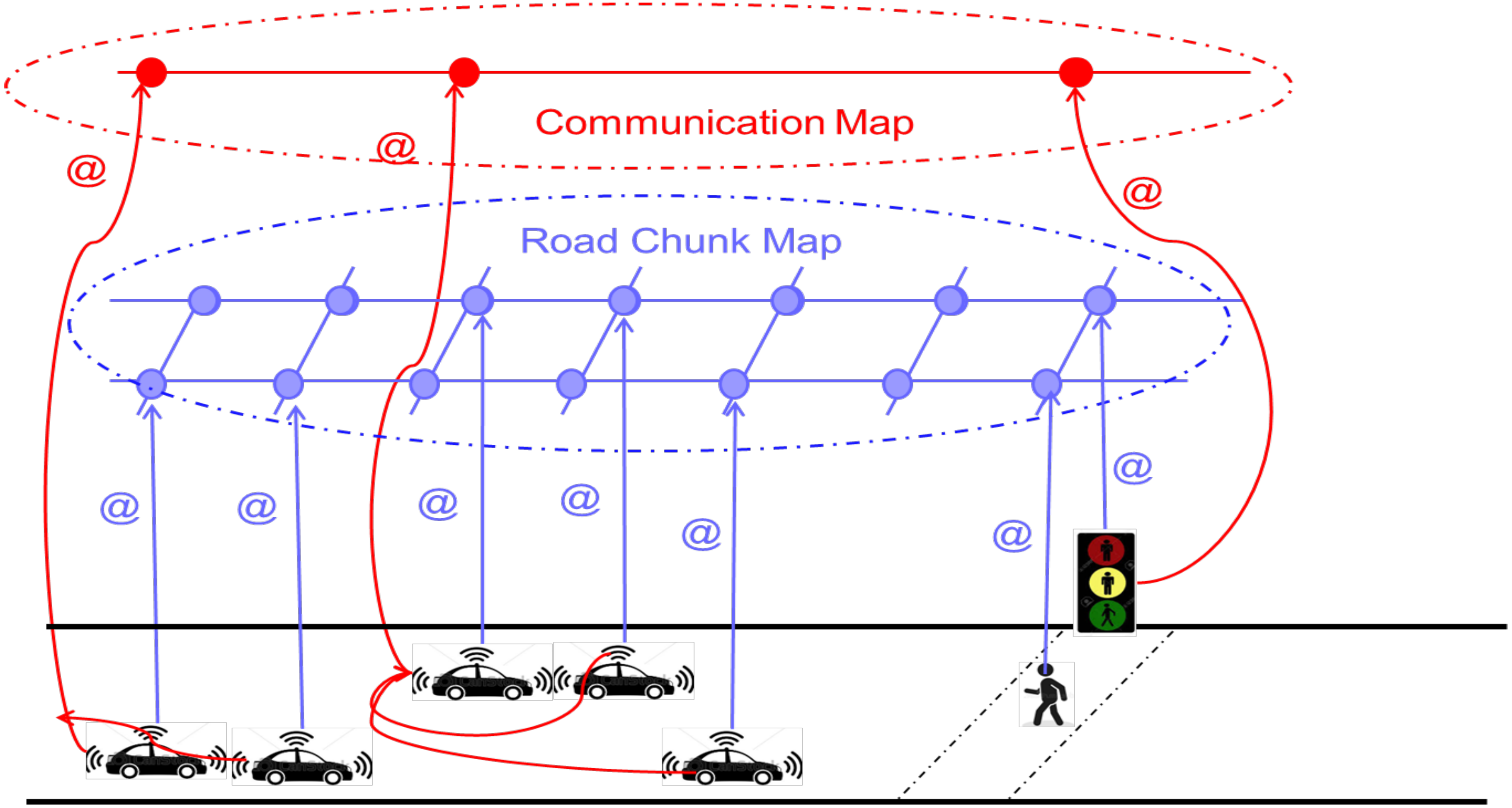
- ❑ A motif is a "world" where components live, a coordination mode consisting of
- ▪ A set of components, instances of types of agents or objects
- ▪ A map that is a graph (N,E) used to describe relations between components e.g. geographical, organizational, etc.
- ▪ An address function @ mapping components into nodes of the map

- ▪ Interaction rules: define interactions (atomic multiparty synchronization) between components

- ▪ Configuration rules:
  - Mobility of components (change of @)
  - Creation/deletion of components
  - Dynamic change of the map

The meaning of systems models is defined using operational semantics

When Self-driving Cars are Safe Enough? – Modeling: DR-BIP



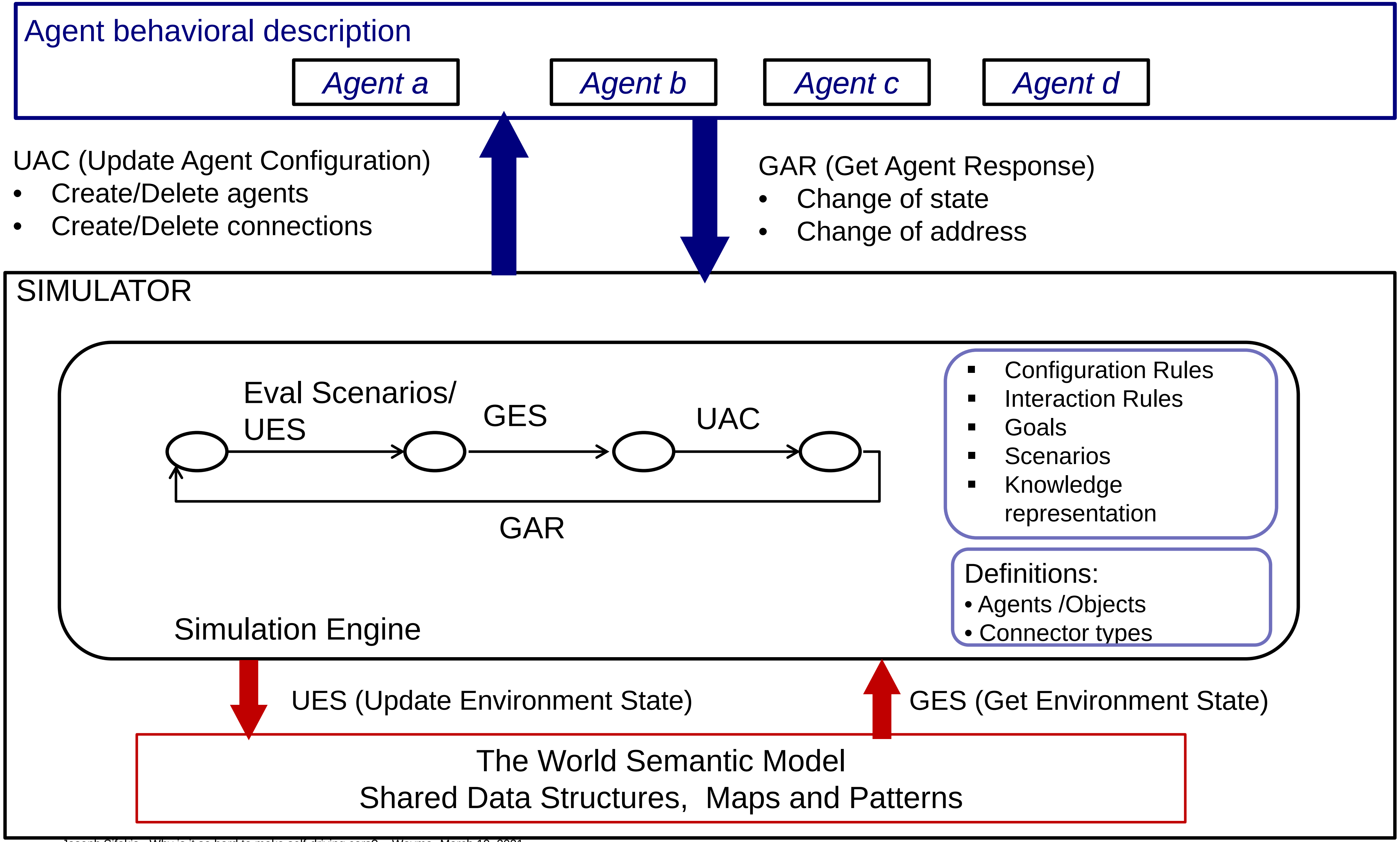
Interaction rule:

for all a,a':vehicle, if [dist(@(a),@(a'))<l] then exchange(a.speed,a'.speed).

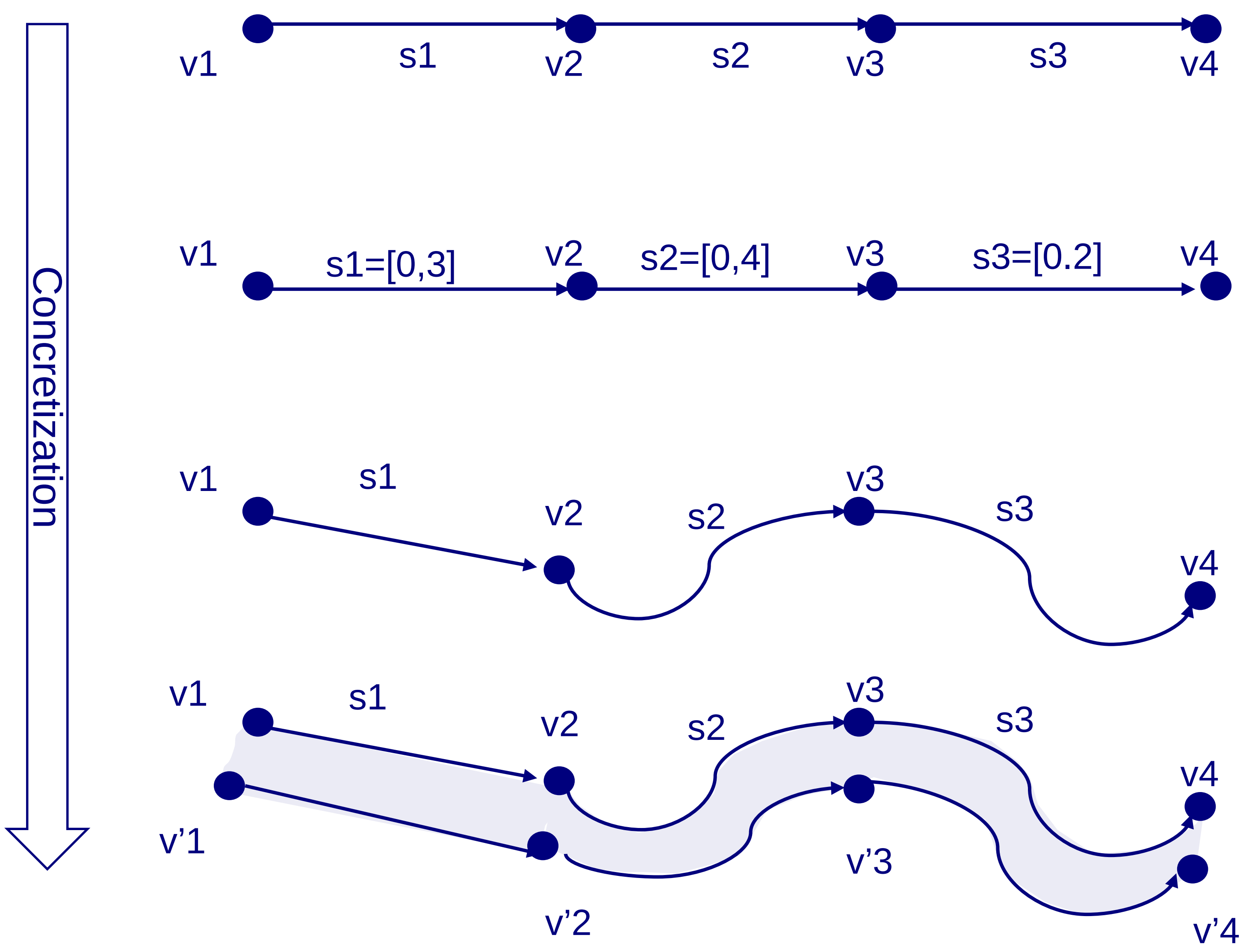
Mobility rule :

for all a:vehicle if @(a)=n and @$^{-1}$(n+1)=empty then @(a):=n+1.

When Self-driving Cars are Safe Enough? – Simulation Environment



When Self-driving Cars are Safe Enough? – Maps



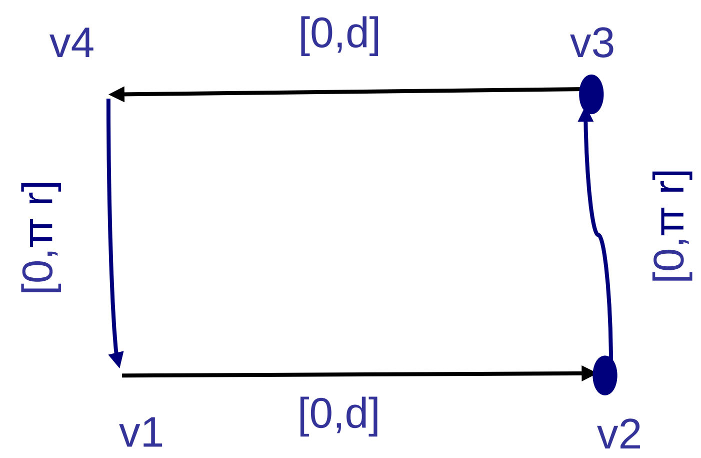
A map is a directed graph:
Vertices v_i : positions
Labels s_i: road segments

Interval map
(each segment is a continuous interval)

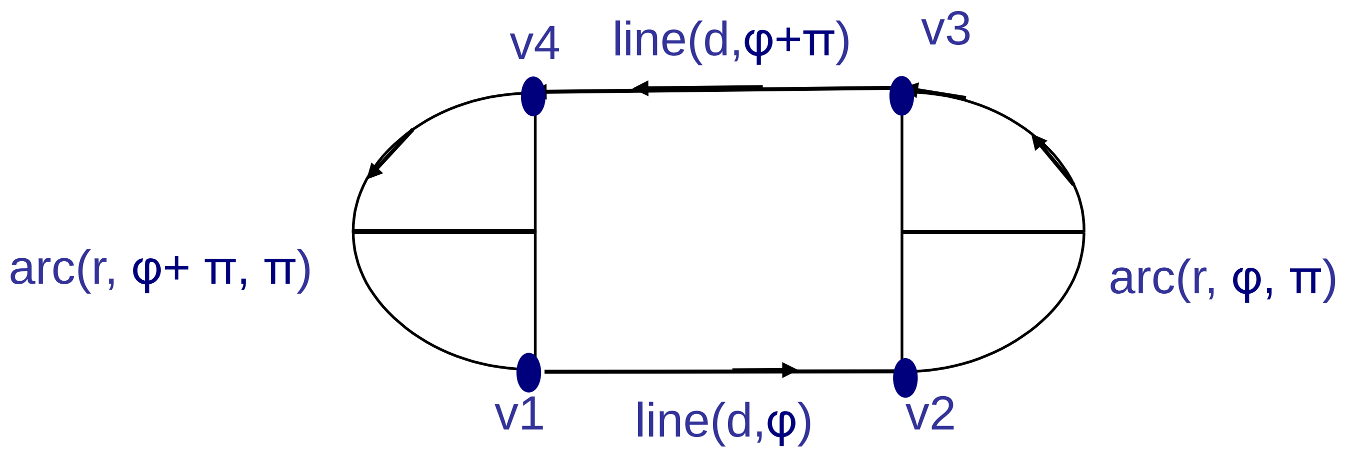
Curve map
(each segment is a curve and for two successive segments s; s' we have s.φ2=s'.φ1

Region map
(each segment is a region with additionall conditions for the composition of regions)

When Self-driving Cars are Safe Enough? – Configuration Logic



Interval Map



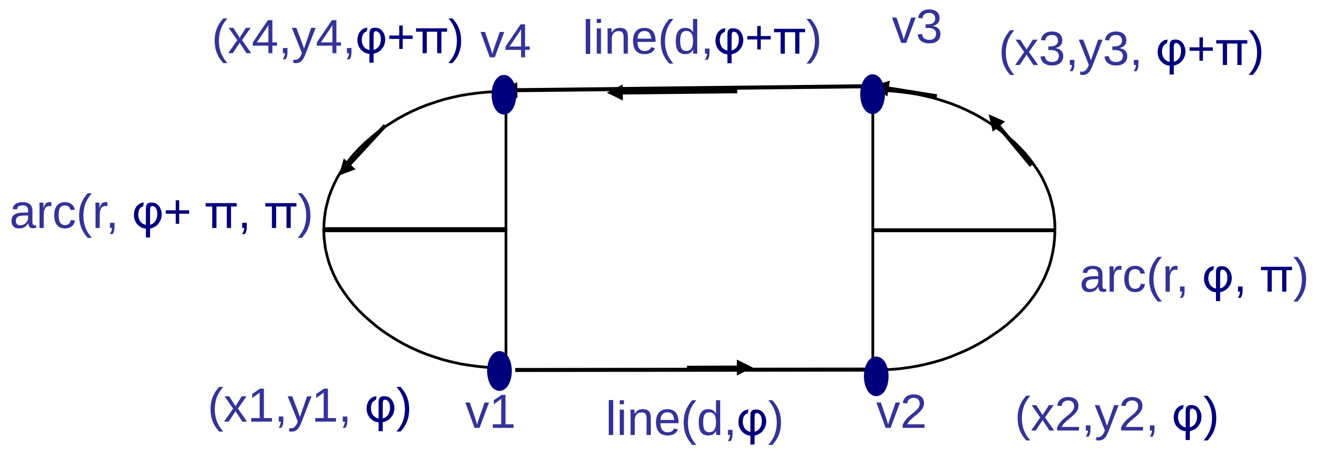
Curve Map

Ring(d,r) = ∃ v1,v2,v3,v4 line(d,φ) (v1,v2)∨ arc (r, φ,π)(v2,v3) ∨ line (d, φ+π)(v3,v4) ∨ arc(r, φ+π,π)(v4,v1)

We have for each segment s a mapping function m

- m[s] (x0,y0, φ0) = (x,y,φ)
- m [line(d)] (x,y,φ)= (x+d cosφ, y+d sinφ, φ) ---
- m [arc(r, θ)](x,y,φ) =
  (x- r sin(φ) + r sin(θ+φ), y+r cos(φ) - r cos (θ+φ), θ+φ)

m[s1.s2]= m[s2](m[s1])



Geometric Interpretation of a Curve Map

When Self-driving Cars are Safe Enough? – Validation: Scenarios (1)

OK/KO

Properties

MONITOR (Runtime Verification of Properties)

∀c1,c2

- @(c1) --d→ @(c2) => d >= 10
- At_Intsec(c1)& In__Intsec(c2) => c1 yielding c2
- @(c1) --d→ @(c2) /\ d>100 =>
  < > (speeding-up(c1) & slowing-down(c2))

Verification Engine

Current Configuration – evaluation of atomic propositions

Meaning of atomic propositions

SYSTEM

c1 c2 c3

ci cj cn

Execution Engine

PROPERTY SPECIFICATION LANGUAGE
(First Order Configuration Logic)

1. Configuration language e.g. connectivity and state predicates with quantification over components, contexts

2. Contexts e.g. Motifs, Maps, Modes

3. Temporal logic relations e.g. usually linear temporal logic or time stamps

OVERVIEW

- Why is it so hard?

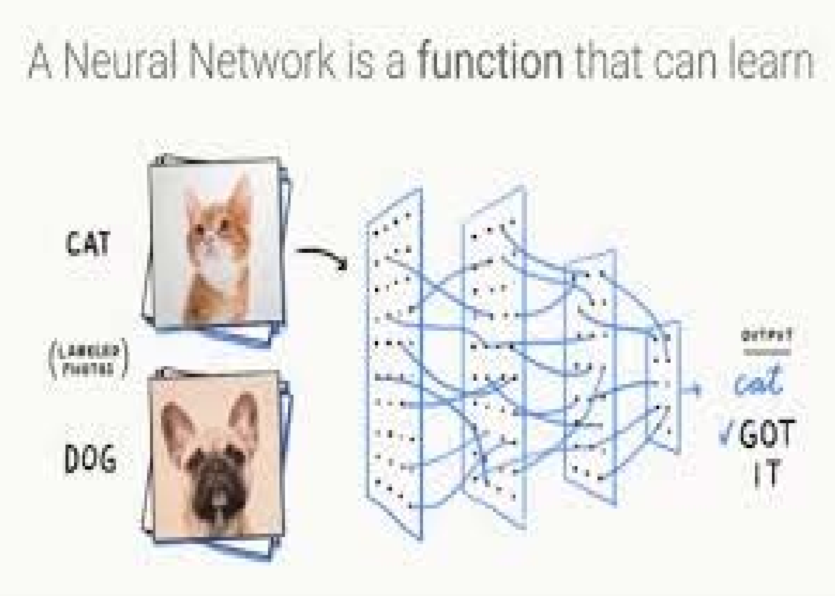
- Toward Trustworthy Autonomy

  - Trustworthy Autopilot Design

  - When Self-driving Cars are Safe Enough?

- Discussion

Discussion – Two Questions about Neural Networks



- What Neural Networks Are (Not) Good For?
  - The computed function for black and white images is $F_{cd}:\{0,1\}^n \rightarrow \{cat, dog\}$
  - How about the function $F_{eo}:\{0,1\}^n \rightarrow \{even, odd\}$ that gives "even" if the number of 1 in the input sequence is even and 0 otherwise?
  - How about the function $F_{corr}:\{0,1\}^n \rightarrow \{bug, nobug\}$ where the input is a program?

Sensitivity (robustness) matters !!

- When Neural Nets are explainable?
  - Given a neural net computing a function F(X) = Y , explainability means that it is possible to extract a function F_{app} e.g. in the form of a <u>program</u> such that F_{app} approximates F modulo some similarity relation ~
    $\forall$ X,X' X'~X implies $F_{app}(X') \sim F(X)$.
    Note that we can check the correctness of F_{app} with respect to properties preserved by ~.
  - How F_{app} can be extracted from F? In principle, by structural analysis, by composing the behaviors of nodes
- Note that
  - $F_{cd}:\{0,1\}^n \rightarrow \{cat, dog\}$ cannot be explainable because "cat" and "dog" is not amenable to formalization
  - $F_{ca}: \{v_a, v_e, d) \rightarrow \{acc, dec\}$, collision avoidance system, is explainable (relation between quantities)

Discussion – Human Situation Awareness Cannot be Matched

To match human-level performance, systems should be able to deal with knowledge of the common sense world.

- Our mind is equipped with a semantic model of the world
  - used to Interpret sensory information and natural language in particular;
  - progressively built and automatically updated though learning and reasoning;
  - integrating in a huge network knowledge acquired along lifespan and involving concepts, cognition rules and patterns.

WIRED BACKCHANNEL BUSINESS CULTURE GEAR IDEAS SCIENCE SECURITY MY ACCOUNT GIVE A GIFT

ANDY GREENBERG SECURITY 10.11.2020 04:00 PM

Split-Second ‘Phantom’ Images Can Fool Tesla’s Autopilot

Researchers found they could stop a Tesla by flashing a few frames of a stop sign for less than half a second on an internet-connected billboard.

Discussion – The Superiority of Human Situation Awareness

To match human-level performance, systems should be able to deal with knowledge of the common sense world.

- Our mind is equipped with a semantic model of the world
  - used to Interpret sensory information and natural language in particular;
  - progressively built and automatically updated though learning and reasoning;
- integrating in a huge network knowledge acquired along lifespan and involving concepts, cognition rulepatterns.

- Human understanding combines: 1) bottom-up reasoning from sensor level to the semantic model of the mind; and 2) top-down from the semantic network to perception.

- It is highly improbable that we could ever build such semantic models given their overwhelming complexity - as evidenced by the very little progress in semantic analysis of natural languages so far.

*“**Intelligence** is what you use when you don't know what to do.” Jean Piaget*

Discussion – Some Conclusions

- The trustworthy autonomous systems challenge <u>is not only about intelligent agents</u>, it involves equally important <u>systems engineering issues</u>.

- Hybrid design could leverage on a solid body of knowledge for safe and efficient decision making and thus enhance confidence.
  - End-to-end monolithic AI-enabled solutions taking the “brute force way” <u>precluding safety guarantees</u> are likely not to be accepted;
  - The challenge is linking symbolic and non-symbolic knowledge e.g. sensory information and models of the environment.
- Global system validation is achievable <u>only</u> through simulation and testing.
  - Realistic and semantically sound modeling becomes of paramount importance for validation
  - Any technically sound safety evaluation <u>should be model-based</u> e.g. relies on criteria defined on an implicit or an explicit system model.

- There is a big gap between automated and autonomous systems – the transition cannot be progressive: ADAS cannot gradually evolve into self-driving systems!!

- Nonetheless, autonomic complexity drastically scales down for enhanced situation awareness (perception) and environment predictability

- To reach the vision we need to develop a new scientific and engineering foundation. And this will take some time.

Thank you