Space and Time for Traffic Manoeuvres

Ernst-Rüdiger Olderog
Christopher Bischopink

Festkolloquium for Jan Peleska
3 March 2023

Carl von Ossietzky Universität Oldenburg

Collaboration with Jan

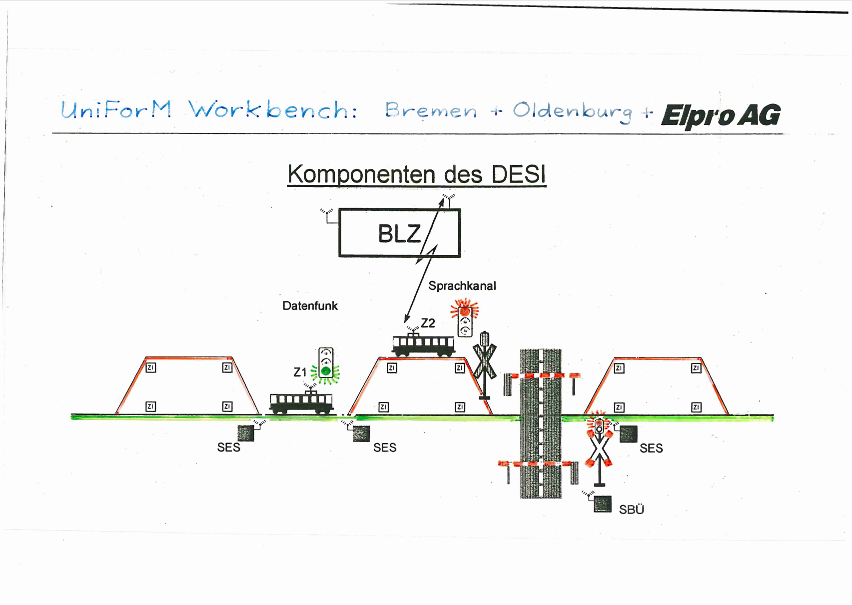
Project UniForM Workbench 1995–98 [KPOB99]:

- Bernd Krieg-Brückner
- Jan Peleska
- Ernst-Rüdiger Olderog
- Alexander Baer, Elpro LET Berlin

Part of project contents:

- Application area: railway control for trams

Single-track Line Segment



Collaboration with Jan

Project UniForM Workbench, 1995–98 [KPOB99]:

- Bernd Krieg-Brückner
- Jan Peleska
- Ernst-Rüdiger Olderog
- Alexander Baer, Elpro LET Berlin

Part of project contents:

- Application area: railway control for trams
- programming language ST dedicated for Programmable Logic Controllers (PLCs).
- Henning Dierks introduced PLC-Automata [Die01]: formal semantics amenable for real-time model checking

From Trains to Cars: The Challenge

Cooperation with Anders P. Ravn , Aalborg University.

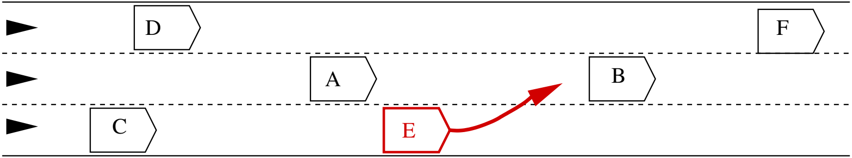
Prove safety (collision freedom) of
traffic manoeuvres on different types of roads.

From Trains to Cars: The Challenge

Cooperation with Anders P. Ravn, Aalborg University.

Prove safety (collision freedom) of
traffic manoeuvres on different types of roads.

motorways: Hilscher, Linker, O. and Ravn [HLOR11]

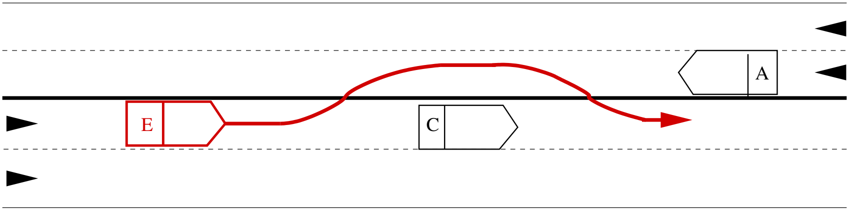


From Trains to Cars: The Challenge

Cooperation with Anders P. Ravn, Aalborg University.

Prove safety (collision freedom) of
traffic manoeuvres on different types of roads.

country roads: Hilscher, Linker and O. [HLO13]



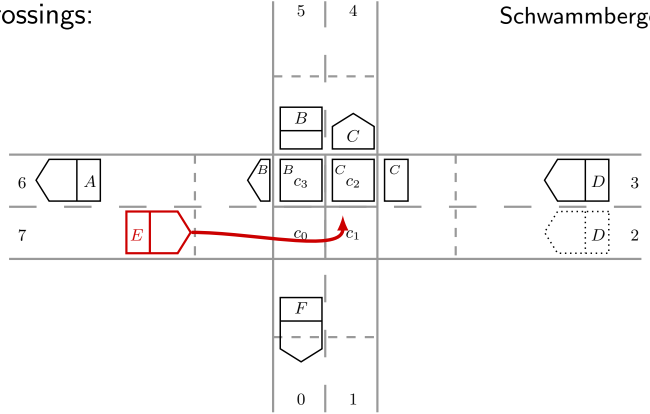
From Trains to Cars: The Challenge

Cooperation with Anders P. Ravn, Aalborg University.

Prove safety (collision freedom) of traffic manoeuvres on different types of roads.

urban crossings:

Schwammberger [Sch18]



Safety is hybrid system verification problem:

$$\text{car dynamics} + \text{car controllers} + \text{assumptions} \models \text{safety}$$

Spatial Properties

Safety is hybrid system verification problem:

$$\text{car dynamics} + \text{car controllers} + \text{assumptions} \models \text{safety}$$

Collision freedom is a spatial property.

Our approach is based on

$$\text{spatial logic} + \text{abstract controllers}$$

hiding car dynamics.

Spatial Properties

Hilscher, Linker, O. and Ravn [HLOR11]

Safety is hybrid system verification problem:

$$\text{car dynamics} + \text{car controllers} + \text{assumptions} \models \text{safety}$$

Collision freedom is a spatial property.

Our approach is based on

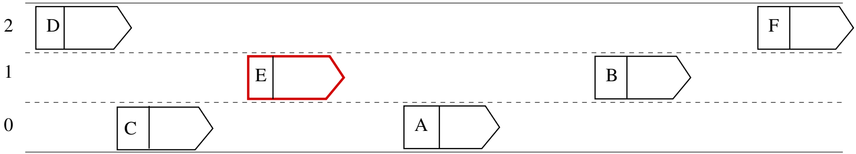
$$\text{spatial logic} + \text{abstract controllers}$$

hiding car dynamics.

Dedicated Multi-lane Spatial Logic inspired by work in ProCoS:

- Interval temporal logic — Moszkowski [Mos85]
- Duration Calculus — Zhou, Hoare and Ravn [ZHR91]

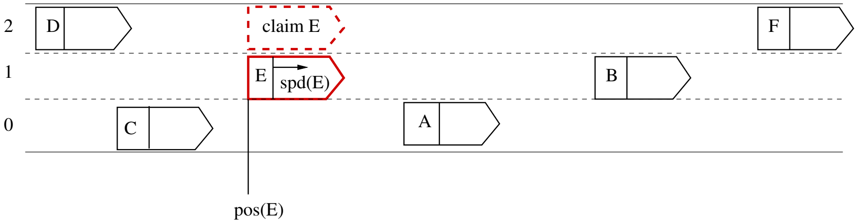
Model



Preliminaries:

- Car identifiers globally unique: $A, B, \ldots$
  Set of all car identifiers: $\mathbb{I}$
- Infinite road ($\mathbb{R}$)
- Lanes: $\mathbb{L} = \{0, \ldots, N\}$

Model



A traffic snapshot is a structure $\mathcal{TS} = (res, clm, pos, spd, acc)$, where

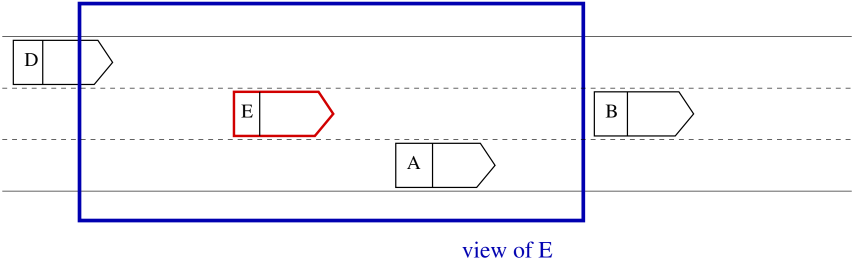
- $res/clm : \mathbb{I} \to \mathcal{P}(\mathbb{L})$: set of lanes each car reserves/claims,
- $pos/spd/acc : \mathbb{I} \to \mathbb{R}$: position/speed/acceleration of each car.

Transitions

$\mathcal{TS} \xrightarrow{\alpha} \mathcal{TS}'$ for an action α of the following type:

$\mathcal{TS} \xrightarrow{t} \mathcal{TS}'$	time passes
$\mathcal{TS} \xrightarrow{\mathsf{acc}(C,a)} \mathcal{TS}'$	accelerate
$\mathcal{TS} \xrightarrow{\mathsf{c}(C,n)} \mathcal{TS}'$	claim
$\mathcal{TS} \xrightarrow{\mathsf{wd\_c}(C)} \mathcal{TS}'$	withdraw claim
$\mathcal{TS} \xrightarrow{\mathsf{r}(C)} \mathcal{TS}'$	reserve
$\mathcal{TS} \xrightarrow{\mathsf{wd\_r}(C,n)} \mathcal{TS}'$	withdraw reservation

Local View



View $V = (L, X, E)$, where

- L subinterval of $\mathbb{L}$,
- X subinterval of $\mathbb{R}$,
- $E \in \mathbb{I}$ identifier of car under consideration.

Multi-lane Spatial Logic with Scopes

Fränzle, Hansen and Ody [FHO15]

MLSLS: Syntax

Car variables: c, d and special variable ego

Formulae φ

$$\varphi ::= \mathit{true} \mid c = d \mid \mathit{free} \mid \mathit{re}(c) \mid \mathit{cl}(c) \mid \ell = k \qquad \text{(atoms)}$$

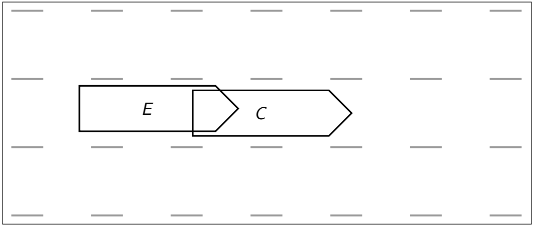
$$\mid \neg\varphi_1 \mid \varphi_1 \wedge \varphi_2 \mid \exists c \colon \varphi_1 \qquad \text{(FOL)}$$

$$\mid \varphi_1 \frown \varphi_2 \mid \begin{array}{c}\varphi_2\\ \varphi_1\end{array} \mid cs : \varphi_1 \qquad \text{(chop and scope)}$$

MLSLS: Semantics

Somewhere: $$\langle \phi \rangle \equiv true \frown \begin{pmatrix} true \\ \phi \\ true \end{pmatrix} \frown true$$

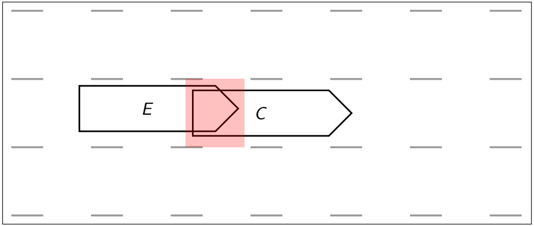
Example : Collision check



MLSLS: Semantics

Somewhere: $$\langle \phi \rangle \equiv \mathit{true} \frown \begin{pmatrix} \mathit{true} \\ \phi \\ \mathit{true} \end{pmatrix} \frown \mathit{true}$$

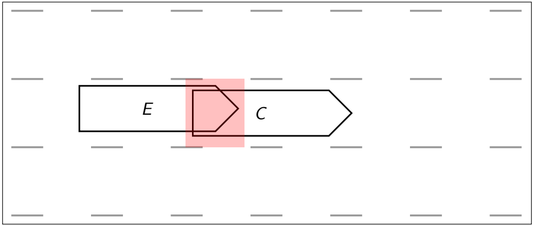
Example : Collision check



MLSLS: Semantics

Somewhere: $$\langle \phi \rangle \equiv true \frown \begin{pmatrix} true \\ \phi \\ true \end{pmatrix} \frown true$$

Example : Collision check

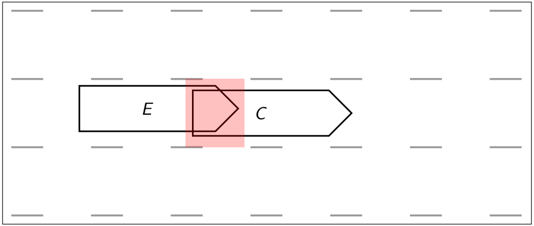


$$\langle re(\mathrm{ego}) \wedge re(c) \rangle$$

MLSLS: Semantics

Somewhere: $$\langle \phi \rangle \equiv \mathit{true} \frown \begin{pmatrix} \mathit{true} \\ \phi \\ \mathit{true} \end{pmatrix} \frown \mathit{true}$$

Example : Collision check



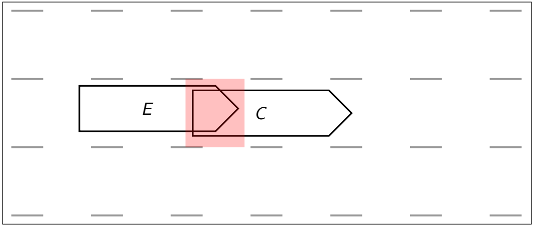
$$\langle re(\mathrm{ego}) \land re(c) \rangle$$

$$cc \equiv \exists c : c \neq \mathrm{ego} \land \langle re(\mathrm{ego}) \land re(c) \rangle$$

MLSLS: Semantics

Somewhere: $$\langle \phi \rangle \equiv \mathit{true} \frown \begin{pmatrix} \mathit{true} \\ \phi \\ \mathit{true} \end{pmatrix} \frown \mathit{true}$$

Example : Collision check

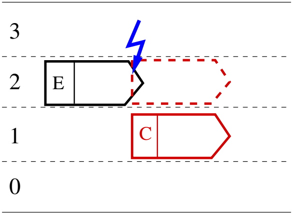


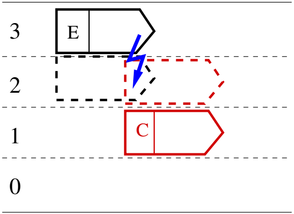
$$cc \equiv \exists c\colon c \neq \mathrm{ego} \wedge \langle re(\mathrm{ego}) \wedge re(c) \rangle$$

Safety from ego's perspective: $\neg cc$

Potential collision

Claim of another car C overlaps with E's own reservation or claim:





$$pc'(\text{ego}) \equiv \exists c : c \neq \text{ego} \land \langle cl(c) \land (re(\text{ego}) \lor cl(\text{ego}))\rangle$$

State Clock Logic

Raskin and Schobbens [RS97]

The syntax of SCL formulae ψ is as follows:

$$\psi ::= p \mid \neg\psi \mid \psi_1 \vee \psi_2 \mid \psi_1 \, \mathcal{U} \, \psi_2 \mid \psi_1 \, \mathcal{S} \, \psi_2 \mid \psi \lhd_{\sim c} \mid \rhd_{\sim c} \psi,$$

where p ranges over propositional symbols and $\sim \in \{<, \leq, =, \geq, >\}$.

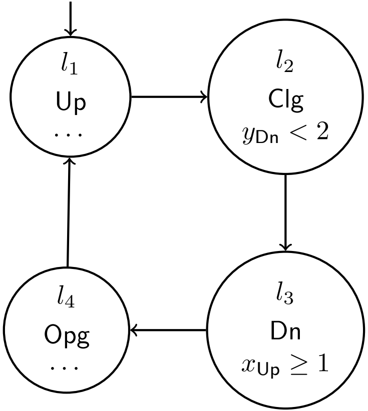
- State history operator $\psi \lhd_{\sim c}$ specifies that the time since ψ held at last must satisfy $\sim c$.
- State prophecy operator $\rhd_{\sim c} \psi$ specifies that the time until ψ holds next must satisfy $\sim c$.

Models of SCL formulae are sets of timed sequences of states .

SC Automata : a variant of Timed Automata with

- a history clock x_p and a prophecy clock y_p

for each proposition p.



Properties:

SC Automata accept the timed sequences of states that satisfy SCL formulae.

E.g. $\mathsf{Clg} \rightarrow \rhd_{<2}\ \mathsf{Dn}$

$\mathsf{Dn} \rightarrow \lhd_{\geq 1}\ \mathsf{Up}$

SC Automata are complementable and language inclusion decidable.

We combine SCL and MLSLS by instantiating the uninterpreted propositions p of SCL with MLSLS formulae .

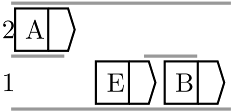
Example :

$$\varphi = \langle pc'(\mathrm{ego})\rangle \rightarrow \triangleright_{<2} \neg\langle pc'(\mathrm{ego})\rangle$$

SC Automata can be used to accept TMLSL formulae.
We employ them as monitors for such formulae.

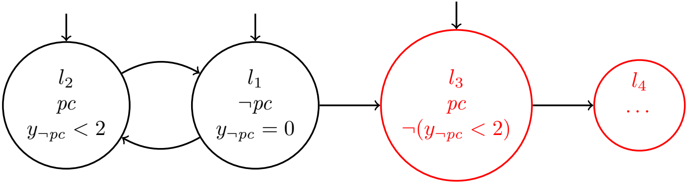
Example: Planned Lane Change

Consider the following traffic snapshot:



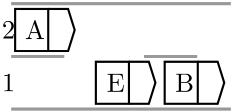
$$\begin{array}{ll} pos(A) = 0 & spd(A) = 13 \\ pos(E) = 10 & spd(E) = 9 \\ pos(B) = 16 & spd(B) = 9 \end{array}$$

- Car A has a monitor for $\varphi = \langle pc'(\text{ego})\rangle \to \rhd_{<2} \neg\langle pc'(\text{ego})\rangle$:



Example: Planned Lane Change

Consider the following traffic snapahot:



$$\begin{array}{ll} pos(A) = 0 & spd(A) = 13 \\ pos(E) = 10 & spd(E) = 9 \\ pos(B) = 16 & spd(B) = 9 \end{array}$$

- Car A has a monitor for $\varphi = \langle pc'(\mathrm{ego})\rangle \to \rhd_{<2} \neg\langle pc'(\mathrm{ego})\rangle$.
- Plan of car E for near future:

$$\omega = \langle(\mathsf{c}(E,2), 3.5), (\mathsf{r}(E), 8)\rangle$$

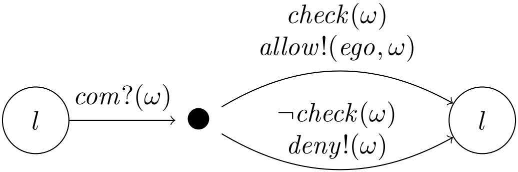
- Car E broadcasts this plan to neighbouring cars.

Supervisor in car A

Car A receives plan of E for near future

$$\omega = \langle (\mathsf{c}(E,2), 3.5), (\mathsf{r}(E), 8) \rangle$$

via $com?\omega$ of the negotiation transitions of its supervisor

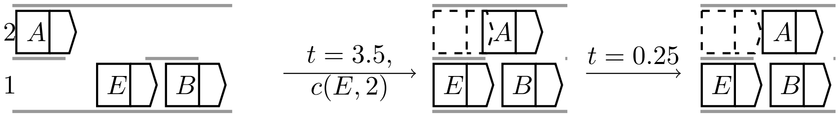


which checks it using its monitor for φ.

Timely resolution

Car A receives plan of E: $\omega = \langle(\mathsf{c}(E,2), 3.5), (\mathsf{r}(E), 8)\rangle$

Supervisor in car A performs *check*(ω) :



$pos(A) = 0 \quad spd(A) = 13$ | $pos(A) = 45.5$ | $pos(A) = 48.75$

$pos(E) = 10 \quad spd(E) = 9$ | $pos(E) = 41.5$ | $pos(E) = 43.75$

$pos(B) = 16 \quad spd(B) = 9$ | $pos(B) = 47.5$ | $pos(B) = 49.75$

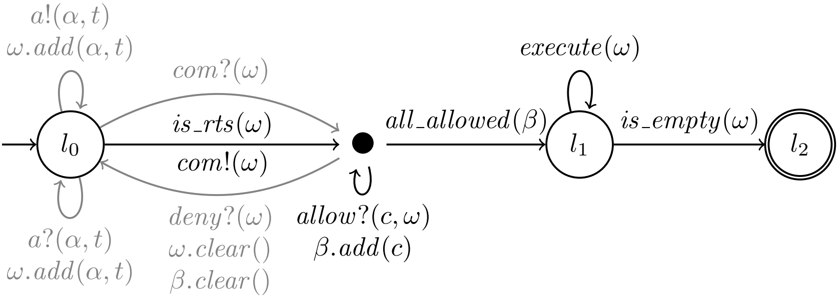
It leads to potential collision $\langle pc'(A)\rangle$ at time 3.5

that is resolved after 0.25 time units.

So supervisor finds satisfaction of φ and broadcasts *allow*$!(A, \omega)$.

Controller $\mathcal{C}$, here in car E

Plan of E: $\boldsymbol{\omega} = \langle (\mathsf{c}(E,2), 3.5), (\mathsf{r}(E), 8) \rangle$



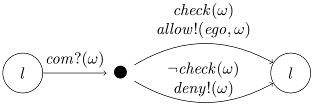
Here car A broadcasts $\mathit{allow}!(A, \boldsymbol{\omega})$.

Thus E can execute its plan.

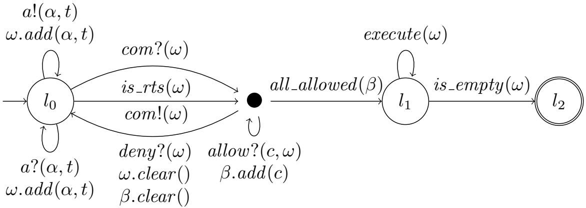
Terminology

- Monitor $\mathcal{A}_\varphi$: SC Automaton for checking a TMLSL formula φ
- Supervisor (Enforcer) $\mathcal{A}'_\varphi$:

  monitor $\mathcal{A}_\varphi$ enriched with negotiating transitions



- Controller $\mathcal{C}$:



Checking a sequence of actions

Consider the monitor $\mathcal{A}_\varphi$ for the TMLSL formula φ.

Given a sequence of actions ω up to a time bound t with $\mathcal{TS}_0 \xrightarrow{\omega} \mathcal{TS}$.

Question: Does $\mathcal{TS}_0, \omega \vDash_t \varphi$ hold ?

Lemma

For $l \in L_0$ with $\mathcal{TS}_0 \vDash \mathcal{L}(l)$ the following holds:

$$\begin{array}{lll} & & \mathit{check}(\mathcal{TS}_0, l, \omega) = \mathit{true} \\ \{\mathit{by\ def.}\} & \mathit{iff} & \exists\, l' \in \mathit{locset}(\mathcal{A}_\varphi, \mathcal{TS}_0, \omega) \wedge l' \notin \mathit{bad} \\ & \mathit{iff} & \mathcal{TS}_0, \omega \vDash_t \varphi. \end{array}$$

Soundness

Theorem

For every finite timed state sequence $m \in L(\mathcal{A}'_\varphi \,||\, \mathcal{C})$ *up to time* t *there is a finite timed word of actions* ω *up to time* t *such that*

- ω *has been successfully checked by the enforcer* $\mathcal{A}'_\varphi$*,*
- *the states in* m *describe the evolution of* $\mathcal{TS}_0$ *along* ω*,*
- ω *and* $\mathcal{TS}_0$ *are a model of* φ *up to time* t*,*

  in symbols: $TS_0, \omega \models_t \varphi$.

Timed sequences of actions and states :

$$\begin{array}{lccccccccc}
\omega: & TS_0 & \xrightarrow{(\alpha_0, t_0)} & TS_1 & \xrightarrow{(\alpha_1, t_1)} & TS_2 & \xrightarrow{(\alpha_2, t_2)} & \cdots & \xrightarrow{(\alpha_{n-1}, t_{n-1})} & TS_n \\
 & \top\!\!\top & & \top\!\!\top & & \top\!\!\top & & & & \uparrow \\
m: & s_0 & & s_1 & & s_2 & & & & t
\end{array}$$

Completeness

Theorem

Every finite timed word of actions ω *with* $\mathcal{TS}_0, \omega \models_t \varphi$ *satisfies*

$$m(\mathcal{TS}_0, \omega) \in L(\mathcal{A}'_{\varphi, L'_0} \,||\, \mathcal{C}) \ ,$$

where $L'_0 = \{\, l \in L_0 \mid \ \mathcal{TS}_0 \models \mathcal{L}(l) \,\}$.

Timed sequences of actions and states :

$$\begin{array}{lcccccccc}
\omega : & TS_0 & \xrightarrow{(\alpha_0, t_0)} & TS_1 & \xrightarrow{(\alpha_1, t_1)} & TS_2 & \xrightarrow{(\alpha_2, t_2)} \cdots & \xrightarrow{(\alpha_{n-1}, t_{n-1})} & TS_n \\
 & \rotatebox{-90}{$\models$} & & \rotatebox{-90}{$\models$} & & \rotatebox{-90}{$\models$} & & & \uparrow \\
m : & s_0 & & s_1 & & s_2 & & & t
\end{array}$$

Conclusion

We presented an approach for autonomous cars on motorways

- to check their manoeuvre plans for the near future before carrying them out,
- based on a combination of a spatial and a timed logic.

Future work:

- Planning beyond a time bound t ?
- How to arrive at the manoeuvre plans ?
- Other types of roads.

Dear Jan,

Congratulations

to your achievements in

applying mathematical rigour to real practical problems!

All the best for your future!

References I

Christopher Bischopink and Ernst-Rüdiger Olderog.
Spatial and timing properties in highway traffic.
In Helmut Seidl, Zhiming Liu, and Corina S. Pasareanu, editors, *Theoretical Aspects of Computing, ICTAC 2022, Proceedings*, volume 13572 of *Lecture Notes in Computer Science*, pages 114–131. Springer, 2022.

Henning Dierks.
PLC-automata: a new class of implementable real-time automata.
Theor. Comput. Sci., 253(1):61–93, 2001.

Yliès Falcone.
You should better enforce than verify.
In Howard Barringer, Ylies Falcone, Bernd Finkbeiner, Klaus Havelund, Insup Lee, Gordon Pace, Grigore Roşu, Oleg Sokolsky, and Nikolai Tillmann, editors, *Runtime Verification*, pages 89–105. Springer, 2010.

Martin Fränzle, Michael R. Hansen, and Heinrich Ody.
No need knowing numerous neighbours - towards a realizable interpretation of MLSL.
In Roland Meyer, André Platzer, and Heike Wehrheim, editors, *Correct System Design*, volume 9360 of *Lecture Notes in Computer Science*, pages 152–171. Springer, 2015.

Mario Gleirscher and Jan Peleska.
Complete test of synthesised safety supervisors for robots and autonomous systems.
In Marie Farrell and Matt Luckcuck, editors, *Proceedings Third Workshop on Formal Methods for Autonomous Systems, FMAS 2021, Virtual*, volume 348 of *EPTCS*, pages 101–109, 2021.

M. Hilscher, S. Linker, and E.-R. Olderog.
Proving safety of traffic manoeuvres on country roads.
In Zhiming Liu, Jim Woodcock, and Huibiao Zhu, editors, *Theories of Programming and Formal Methods*, volume 8051 of *LNCS*, pages 196–212. Springer, 2013.

References II

Martin Hilscher, Sven Linker, Ernst-Rüdiger Olderog, and Anders P. Ravn.
An abstract model for proving safety of multi-lane traffic manoeuvres.
In Shengchao Qin and Zongyan Qiu, editors, *Formal Methods and Software Engineering*, pages 404–419. Springer, 2011.

Bernd Krieg-Brückner, Jan Peleska, Ernst-Rüdiger Olderog, and Alexander Baer.
The UniForM workbench, a universal development environment for formal methods.
In Jeannette M. Wing, Jim Woodcock, and Jim Davies, editors, *FM'99 - Formal Methods, World Congress on Formal Methods in the Development of Computing Systems, Proceedings, Volume II*, volume 1709 of *Lecture Notes in Computer Science*, pages 1186–1205. Springer, 1999.

B. Moszkowski.
A temporal logic for multilevel reasoning about hardware.
Computer, 18(2):10–19, 1985.

Jean-François Raskin and Pierre-Yves Schobbens.
State clock logic: A decidable real-time logic.
In Oded Maler, editor, *Hybrid and Real-Time Systems*, volume 1201 of *Lecture Notes in Computer Science*, pages 33–47. Springer, 1997.

Maike Schwammberger.
An abstract model for proving safety of autonomous urban traffic.
Theor. Comput. Sci., 744:143–169, 2018.

Zhou Chaochen, C. A. R. Hoare, and A. P. Ravn.
A calculus of durations.
Information Processing Letters, 40(5):269–276, 1991.