Verified Traffic Networks: Component-based Verification of Cyber-Physical Flow Systems

Andreas Müller – andreas.mueller@jku.at
Stefan Mitsch - stefan.mitsch@jku.at
André Platzer - aplatzer@cs.cmu.edu

Johannes Kepler University, Linz
Department of Cooperative Information Systems (CIS)
http://cis.jku.at/

Carnegie Mellon University, Pittsburgh
Computer Science Department
http://www.ls.cs.cmu.edu/

Overview

Introduction – Traffic Management

Traffic Management System

- Operate traffic through control actions

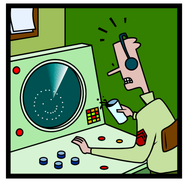
→Safety of critical actions is crucial

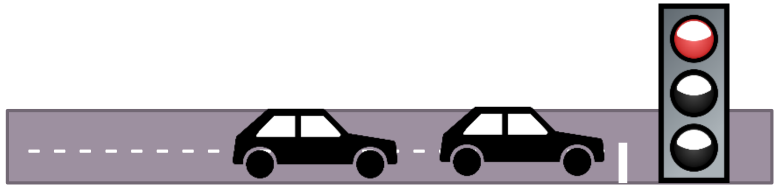
Introduction – Traffic Management

Traffic Management System

- Operate traffic through control actions

→Safety of critical actions is crucial



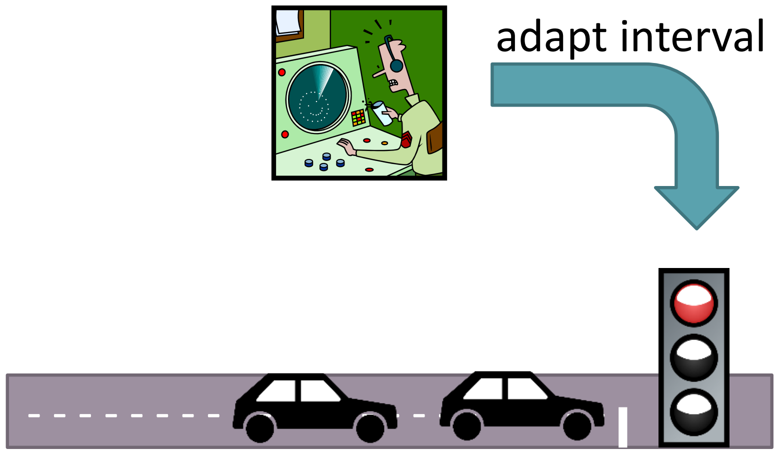


Introduction – Traffic Management

Traffic Management System

- Operate traffic through control actions

→Safety of critical actions is crucial

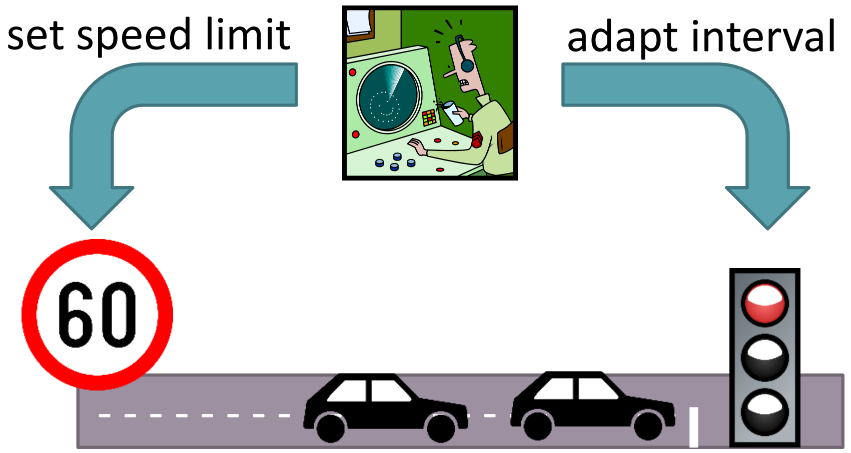


Introduction – Traffic Management

Traffic Management System

- Operate traffic through control actions

→Safety of critical actions is crucial



Introduction – Traffic Management

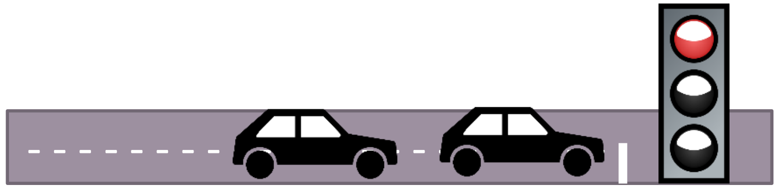
Traffic Management System

- Operate traffic through control actions

→Safety of critical actions is crucial

Safety

- No traffic breakdown=load never exceeds capacity



Introduction – Traffic Management

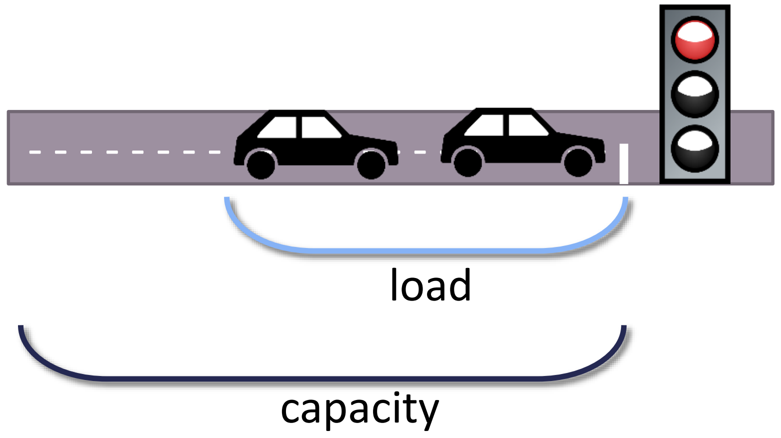
Traffic Management System

- Operate traffic through control actions

→Safety of critical actions is crucial

Safety

- No traffic breakdown=load never exceeds capacity



Introduction – Traffic Management

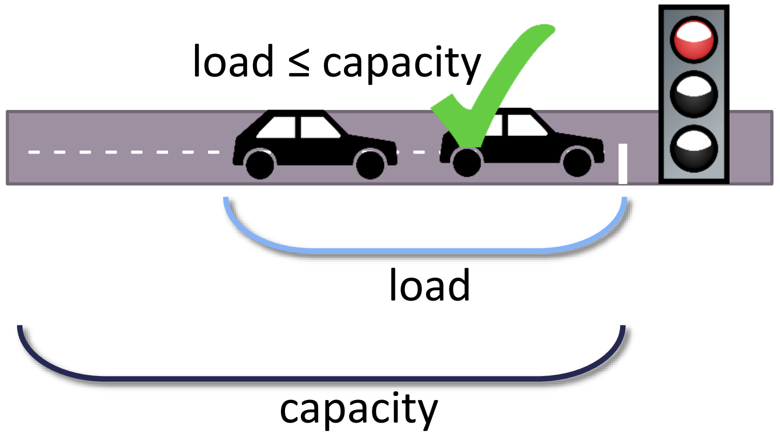
Traffic Management System

- Operate traffic through control actions

→Safety of critical actions is crucial

Safety

- No traffic breakdown=load never exceeds capacity



Introduction – Traffic Management

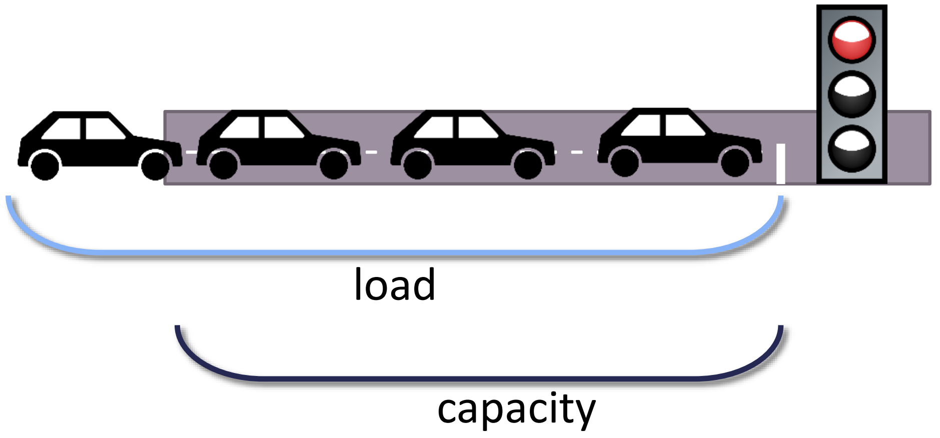
Traffic Management System

- Operate traffic through control actions

→Safety of critical actions is crucial

Safety

- No traffic breakdown=load never exceeds capacity



Introduction – Traffic Management

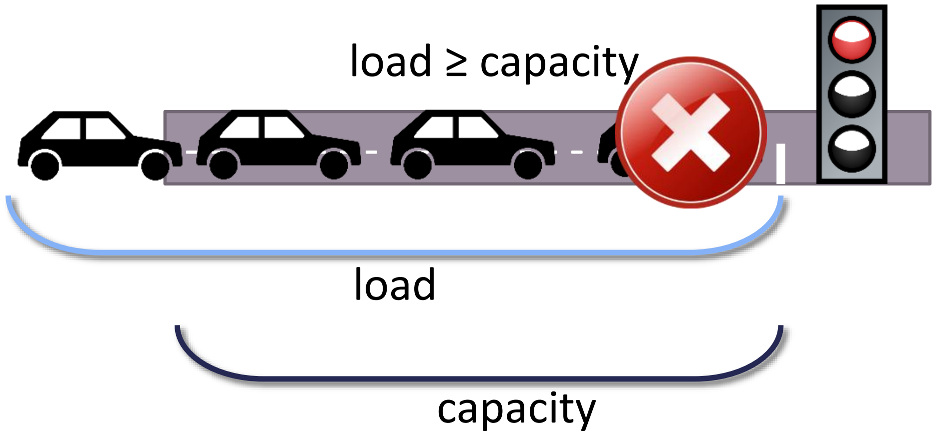
Traffic Management System

- Operate traffic through control actions

→Safety of critical actions is crucial

Safety

- No traffic breakdown=load never exceeds capacity



Introduction – Traffic Management

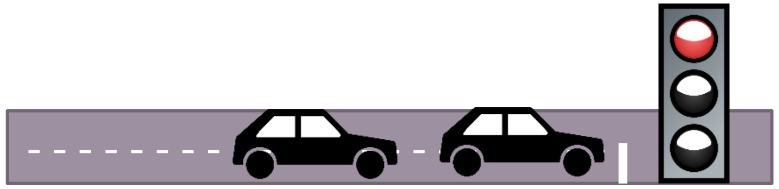
Traffic Management System

- Operate traffic through control actions

→Safety of critical actions is crucial

Safety

- No traffic breakdown=load never exceeds capacity
- Property: Starting in **safe state**, **all runs** stay in **safe state**



Introduction – Traffic Management

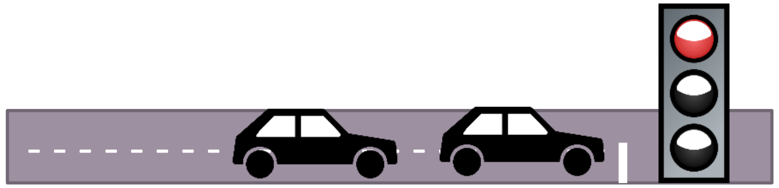
Traffic Management System

- Operate traffic through control actions

→Safety of critical actions is crucial

Safety

- No traffic breakdown=load never exceeds capacity
- Property: Starting in **safe state**, **all runs** stay in **safe state**



Cyber-physical systems (CPS)

- **Cyber** and **physical** capabilities
- Continuous physical-part: **traffic flow**
- Discrete cyber-part: **traffic light switching**

Introduction – Traffic Management

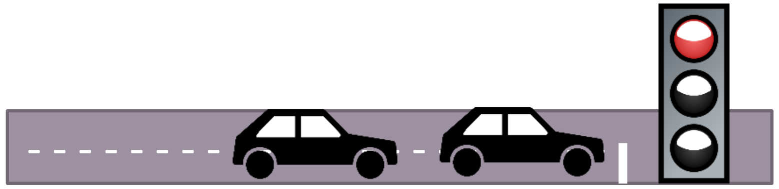
Traffic Management System

- Operate traffic through control actions

→Safety of critical actions is crucial

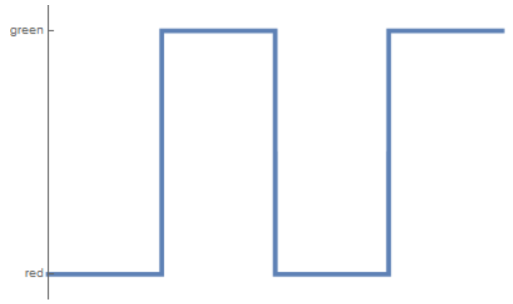
Safety

- No traffic breakdown=load never exceeds capacity
- Property: Starting in **safe state**, **all runs** stay in **safe state**

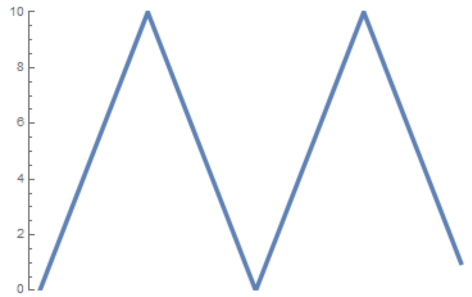


Cyber-physical systems (CPS)

- **Cyber** and **physical** capabilities
- Continuous physical-part: **traffic flow**
- Discrete cyber-part: **traffic light switching**



$tl \coloneqq red/green$



$load' = tl$

Introduction – Traffic Management

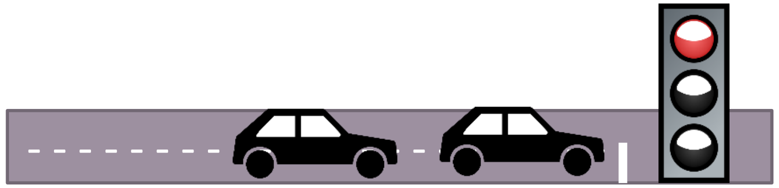
Traffic Management System

- Operate traffic through control actions

→Safety of critical actions is crucial

Safety

- No traffic breakdown=load never exceeds capacity
- Property: Starting in **safe state**, **all runs** stay in **safe state**



Cyber-physical systems (CPS)

- **Cyber** and **physical** capabilities
- Continuous physical-part: **traffic flow**
- Discrete cyber-part: **traffic light switching**

Methods to analyze **models** of CPS

- Simulation and Testing (analyze **some** runs): good for complex phenomena
- Verification (mathematically prove correctness of **all** runs): simplified models

Introduction – Traffic Management

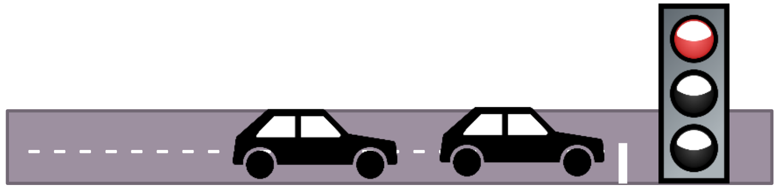
Traffic Management System

- Operate traffic through control actions

→Safety of critical actions is crucial

Safety

- No traffic breakdown=load never exceeds capacity
- Property: Starting in **safe state**, **all runs** stay in **safe state**



Cyber-physical systems (CPS)

- **Cyber** and **physical** capabilities
- Continuous physical-part: **traffic flow**
- Discrete cyber-part: **traffic light switching**

Methods to analyze **models** of CPS

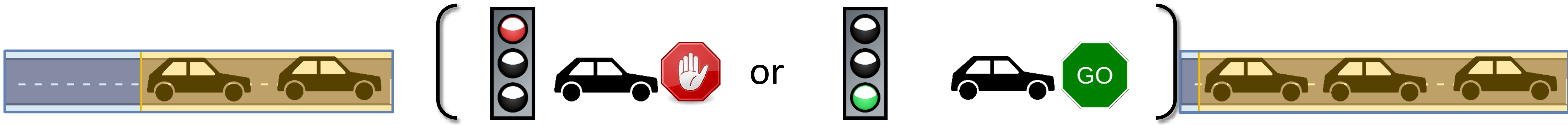
- Simulation and Testing (analyze **some** runs): good for complex phenomena
- **Verification (mathematically prove correctness of all runs): simplified models**

Introduction – Verification

Verification

- Transform property by user-guided application of proof rules
- Starting in **safe state**, **all runs** stay in **safe state**

Example



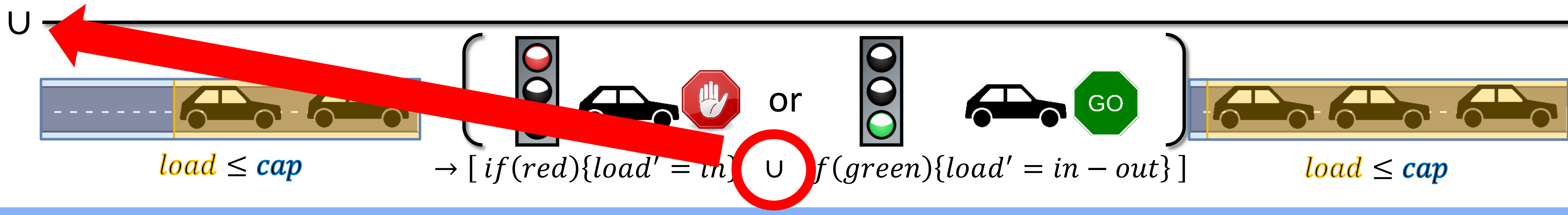
$load \leq cap \quad \rightarrow [\, if(red)\{load' = in\} \;\cup\; if(green)\{load' = in - out\}\,] \quad load \leq cap$

Introduction – Verification

Verification

- Transform property by user-guided application of proof rules
- Starting in **safe state**, **all runs** stay in **safe state**

Example



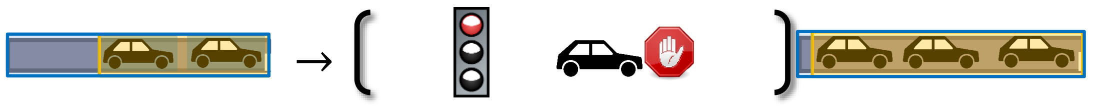
$load \leq cap \rightarrow [\, if(red)\{load' = in\} \cup if(green)\{load' = in - out\}\,]\; load \leq cap$

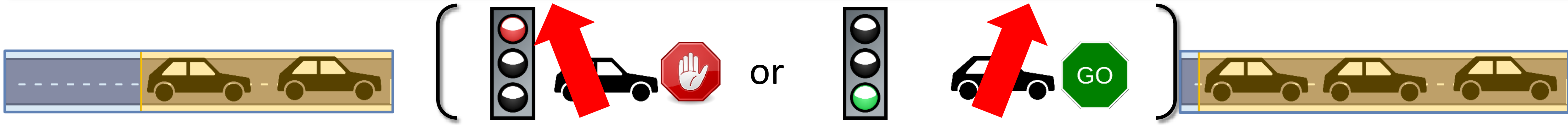
Introduction – Verification

Verification

- Transform property by user-guided application of proof rules
- Starting in **safe state**, **all runs** stay in **safe state**

Example





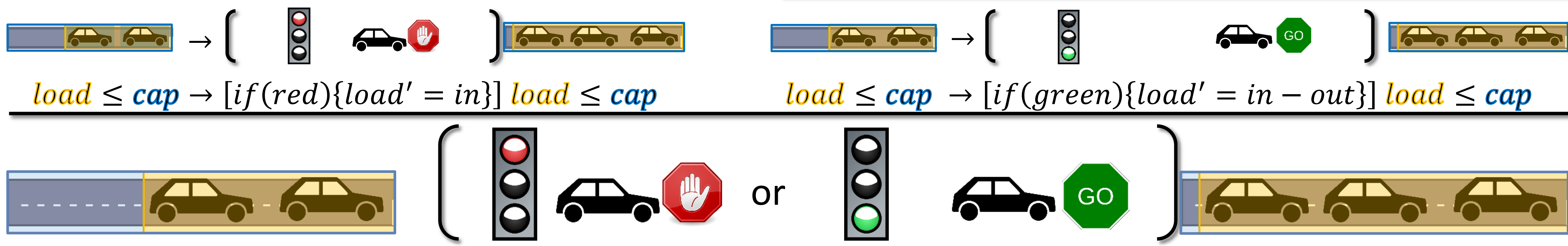
$$\cup\ \frac{load \le cap \to [if(red)\{load' = in\}]\, load \le cap \qquad load \le cap \to [if(green)\{load' = in - out\}]\, load \le cap}{load \le cap \to [\, \underbrace{if(red)\{load' = in\}} \cup \underbrace{if(green)\{load' = in - out\}}\,]\, load \le cap}$$

Introduction – Verification

Verification

- Transform property by user-guided application of proof rules
- Starting in **safe state**, **all runs** stay in **safe state**

Example



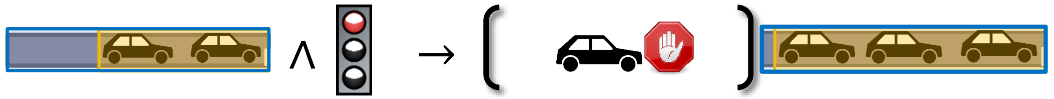
$$\cup\ \frac{load \le cap \rightarrow [if(red)\{load' = in\}]\ load \le cap \qquad \overline{\dots}\ \ load \le cap \rightarrow [if(green)\{load' = in - out\}]\ load \le cap}{load \le cap \rightarrow [\, if(red)\{load' = in\} \cup if(green)\{load' = in - out\}\,]\ load \le cap}$$

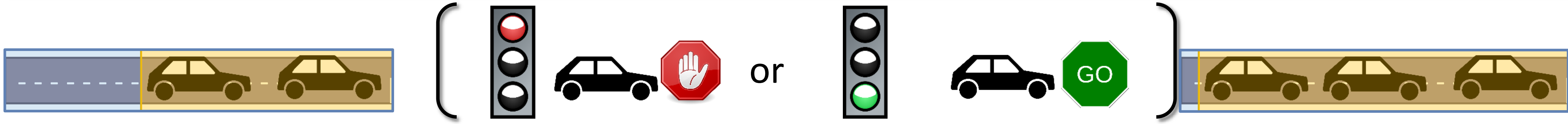
Introduction – Verification

Verification

- Transform property by user-guided application of proof rules
- Starting in **safe state**, **all runs** stay in **safe state**

Example





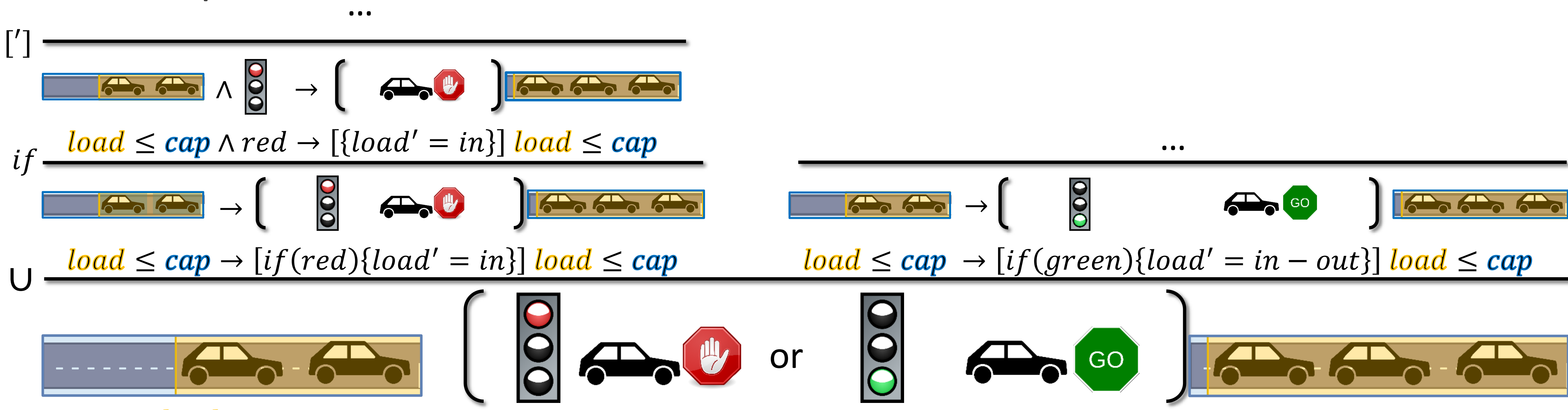
$$\dfrac{\dfrac{load \le cap \land red \to [\{load' = in\}]\; load \le cap}{load \le cap \to [if(red)\{load' = in\}]\; load \le cap}\,if \qquad \dfrac{\ldots}{load \le cap \;\to [if(green)\{load' = in - out\}]\; load \le cap}}{load \le cap \;\to [\; if(red)\{load' = in\} \;\cup\; if(green)\{load' = in - out\}\;]\; load \le cap}\,\cup$$

Introduction – Verification

Verification

- Transform property by user-guided application of proof rules
- Starting in **safe state**, **all runs** stay in **safe state**

Example



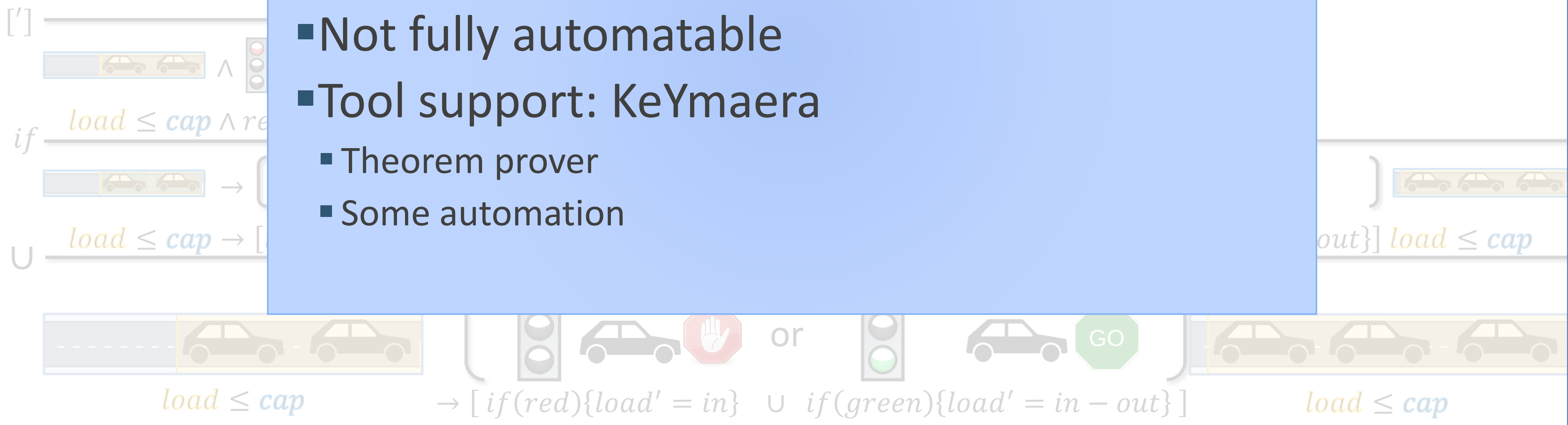
$$\dfrac{\dfrac{\dfrac{\dots}{load \leq cap \land red \rightarrow [\{load' = in\}]\, load \leq cap}\;[']}{load \leq cap \rightarrow [if(red)\{load' = in\}]\, load \leq cap}\;if \qquad \dfrac{\dots}{load \leq cap \rightarrow [if(green)\{load' = in - out\}]\, load \leq cap}}{load \leq cap \rightarrow [\,if(red)\{load' = in\} \cup if(green)\{load' = in - out\}\,]\, load \leq cap}\;\cup$$

Introduction – Verification

Verification

- Transfor
- Starting

Example



$[']$
$\wedge$
$load \leq cap \wedge re$
if
$\rightarrow$
$load \leq cap \rightarrow [$
$out\}]\ load \leq cap$
$\cup$
or
GO
$load \leq cap$
$\rightarrow [\, if(red)\{load' = in\} \;\cup\; if(green)\{load' = in - out\}\,]$
$load \leq cap$

Verification

- One rule application/proof step per statement
- Not fully automatable
- Tool support: KeYmaera
  - Theorem prover
  - Some automation

Challenges

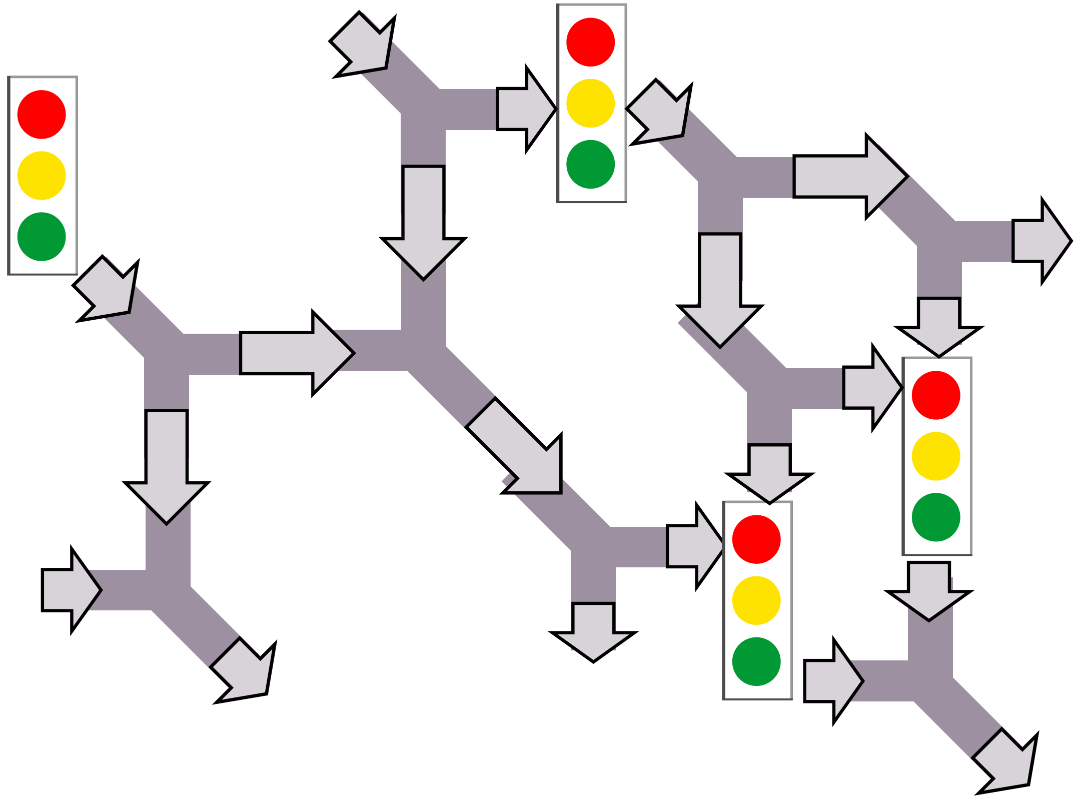
Real systems are large

- Verification for large systems is challenging

Challenges

Real systems are large

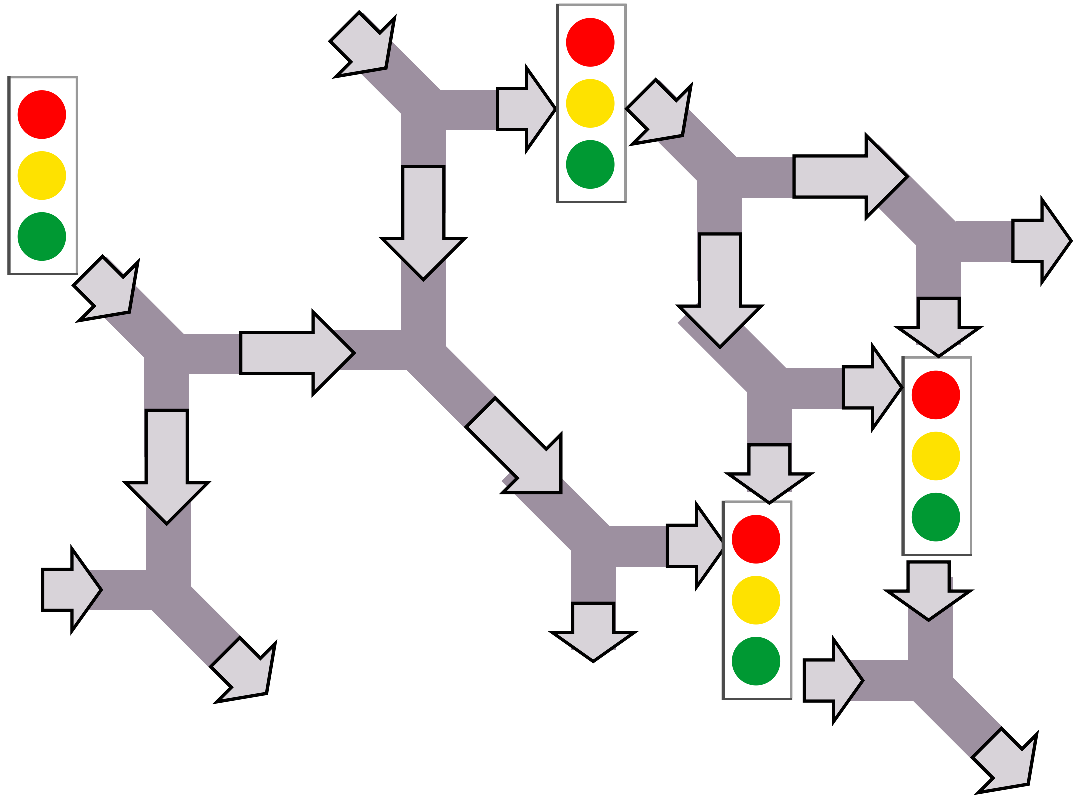
- Verification for large systems is challenging



Challenges

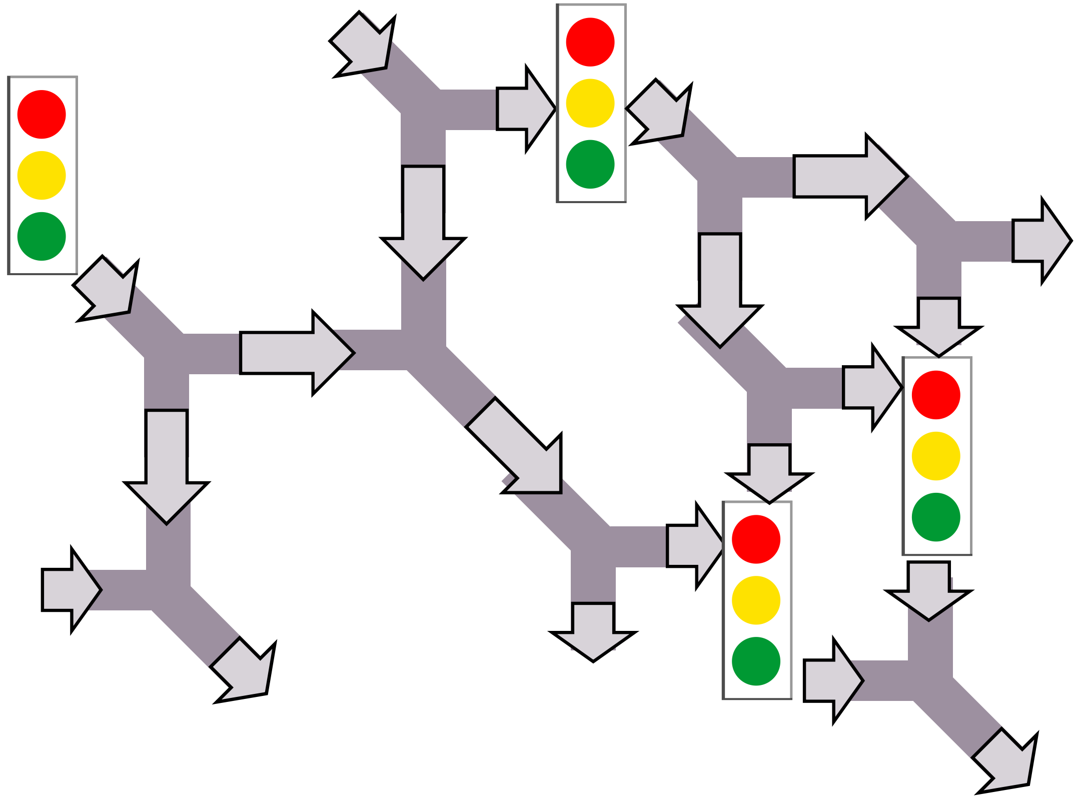
Real systems are large

- Verification for large systems is challenging



Challenges

Real systems are large

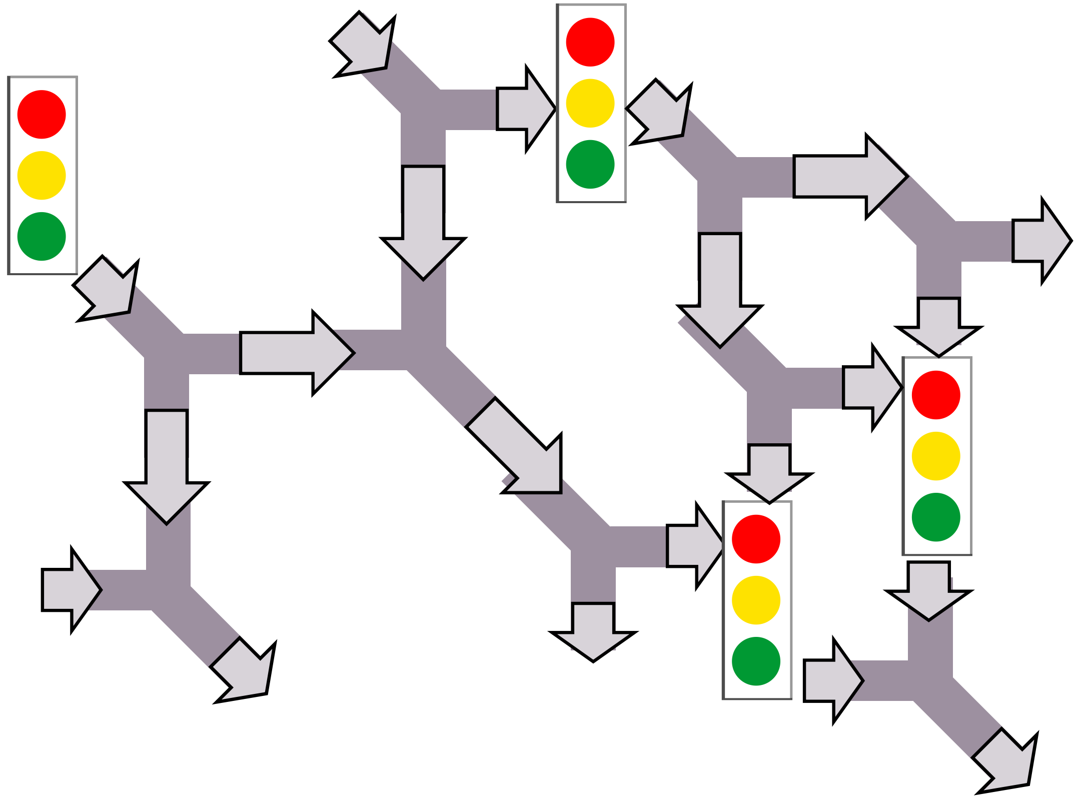


Challenges

Real systems are large

Any change to the model requires full re-verification

- Re-verification only for affected parts

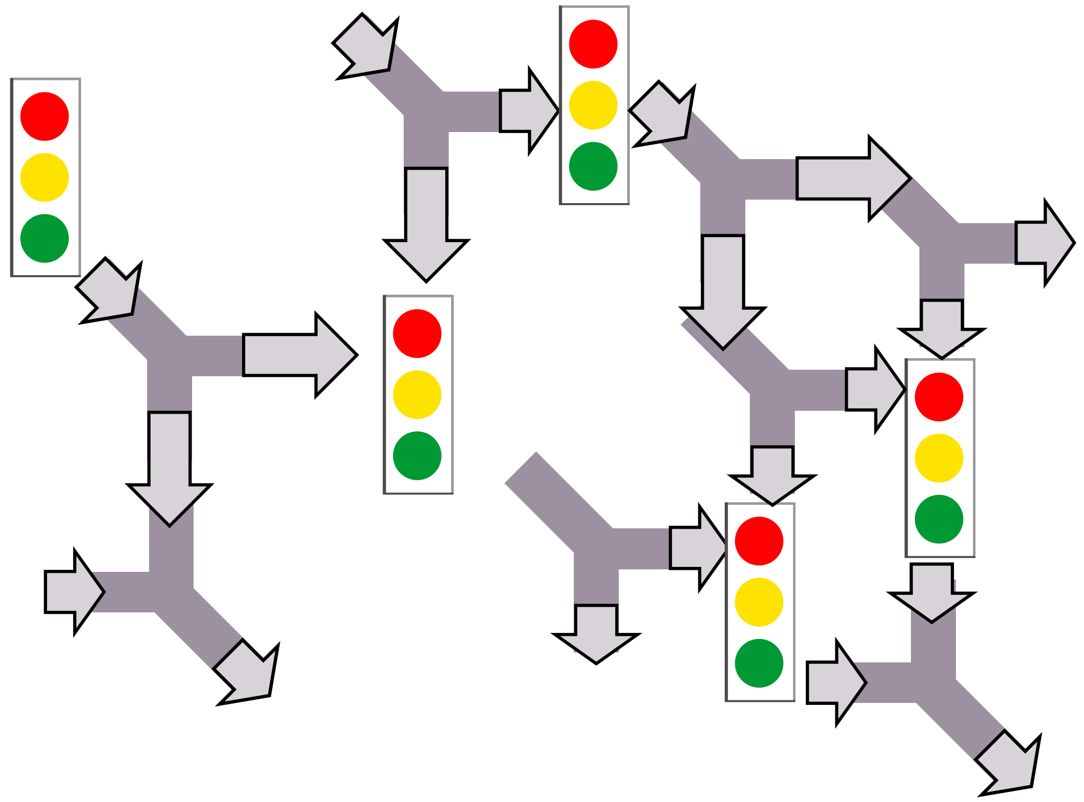


Challenges

Real systems are large

Any change to the model requires full re-verification

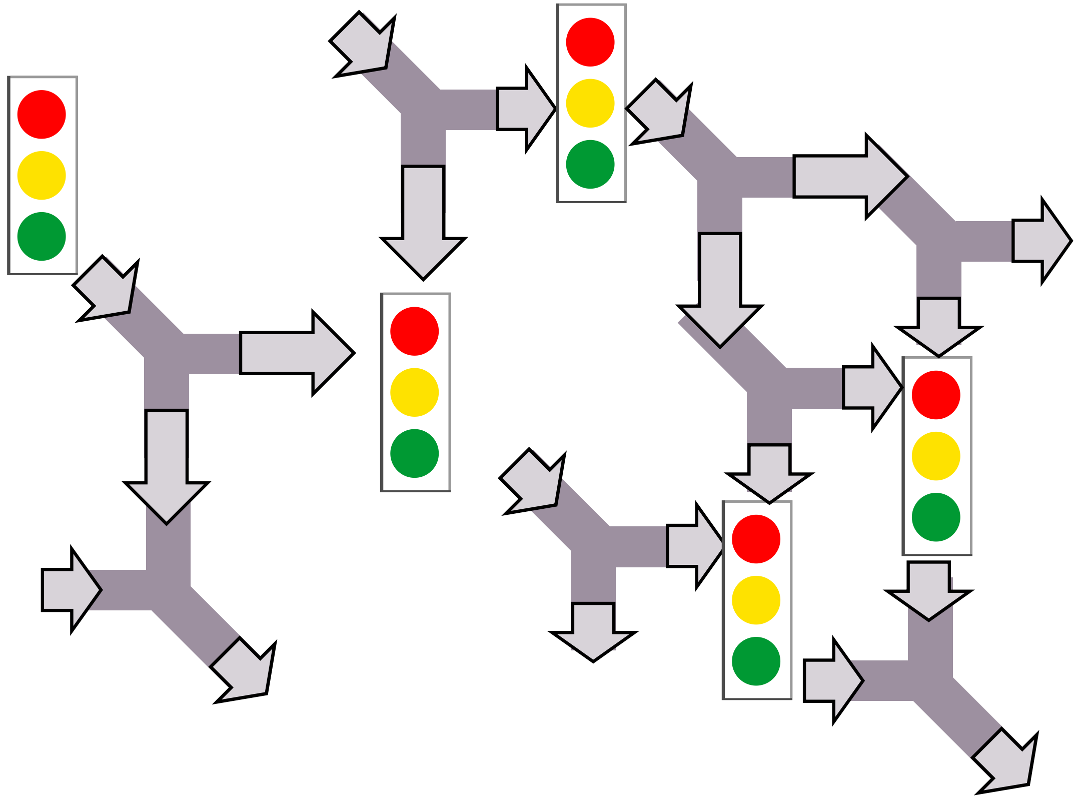
- Re-verification only for affected parts



Challenges

Real systems are large

Any change to the model requires full re-verification



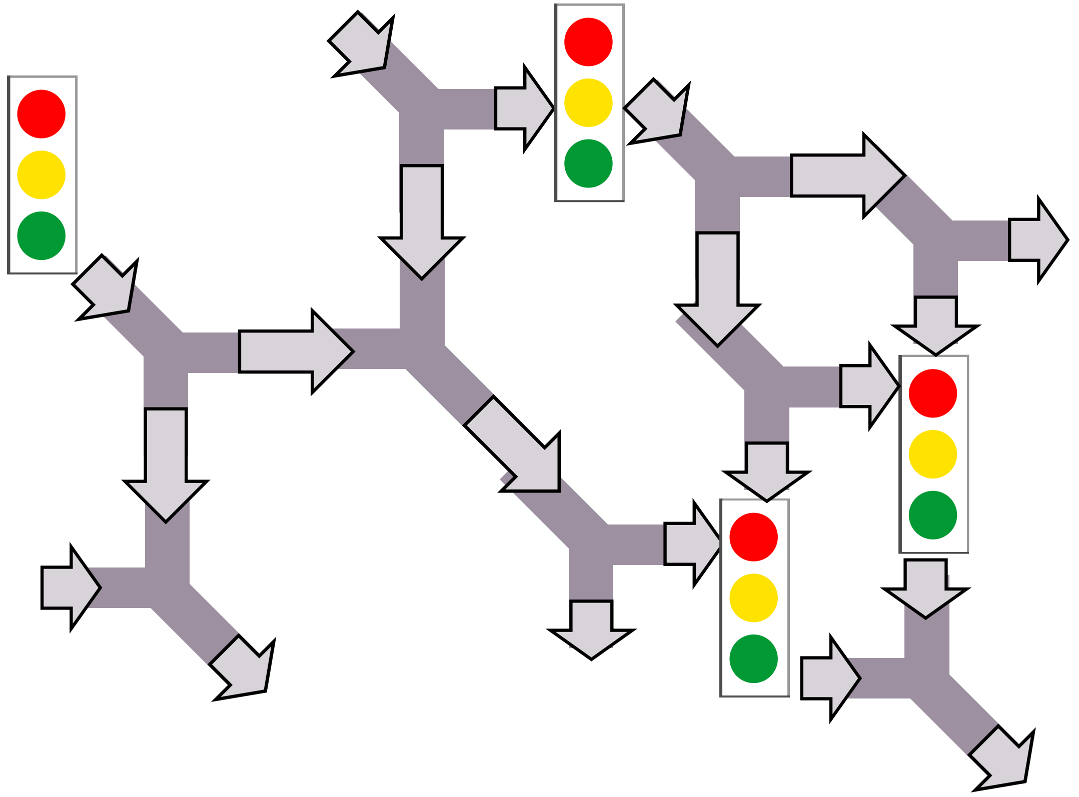
Challenges

Real systems are large

Any change to the model requires full re-verification

Systems often consist of multiple similar patterns

- Redundancy should be utilized in verification



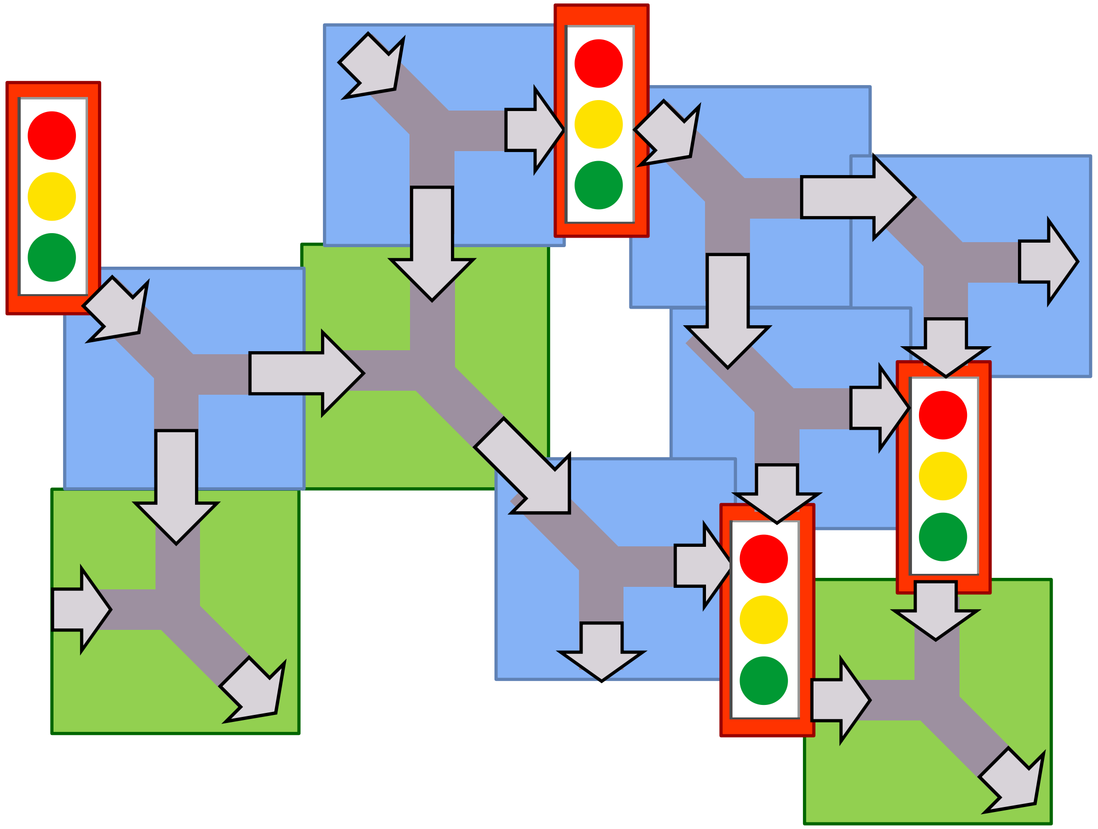
Challenges

Real systems are large

Any change to the model requires full re-verification

Systems often consist of multiple similar patterns

- Redundancy should be utilized in verification



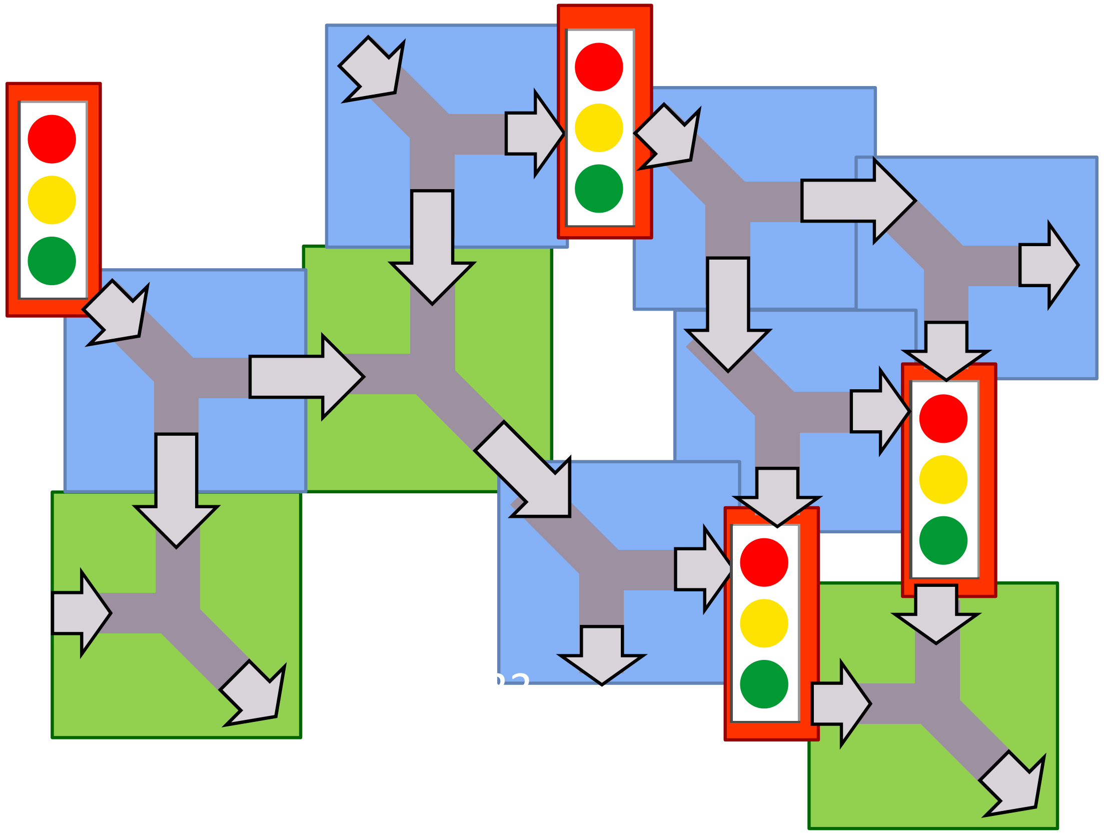
Challenges

Real systems are large

Any change to the model requires full re-verification

Systems often consist of multiple similar patterns

- Redundancy should be utilized in verification

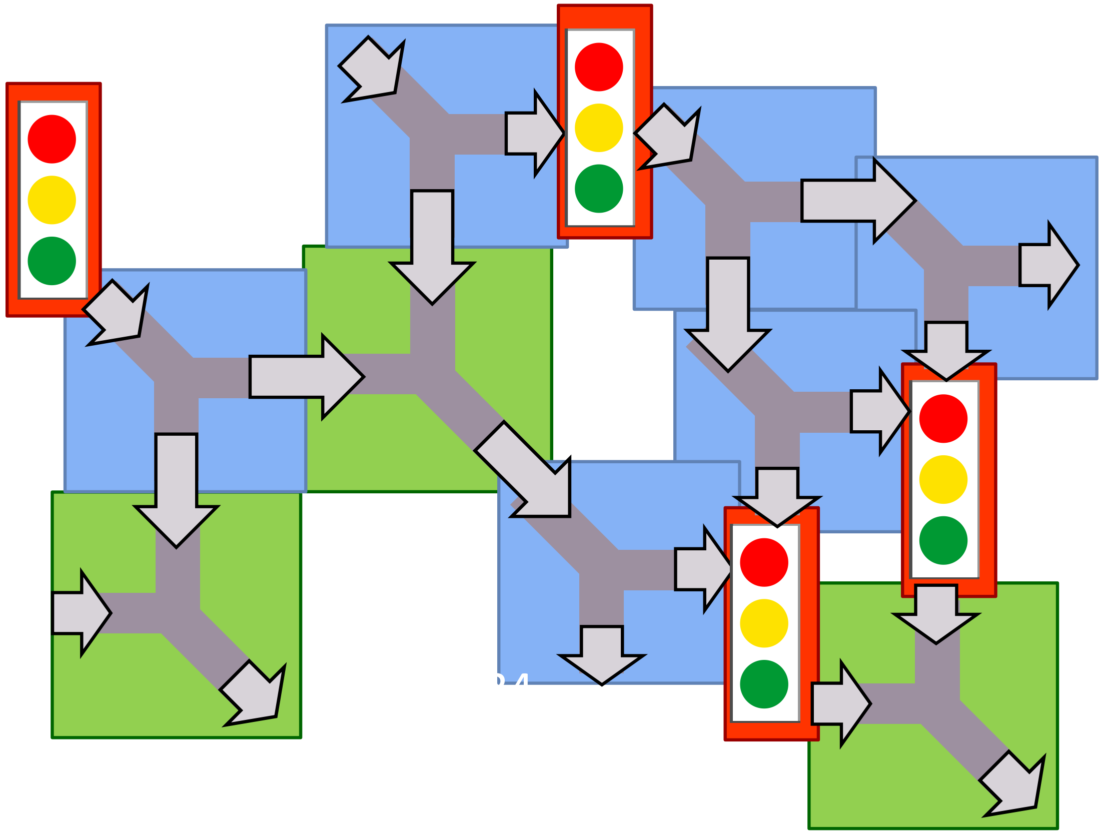


Challenges

Real systems are large

Any change to the model requires full re-verification

Systems often consist of multiple similar patterns



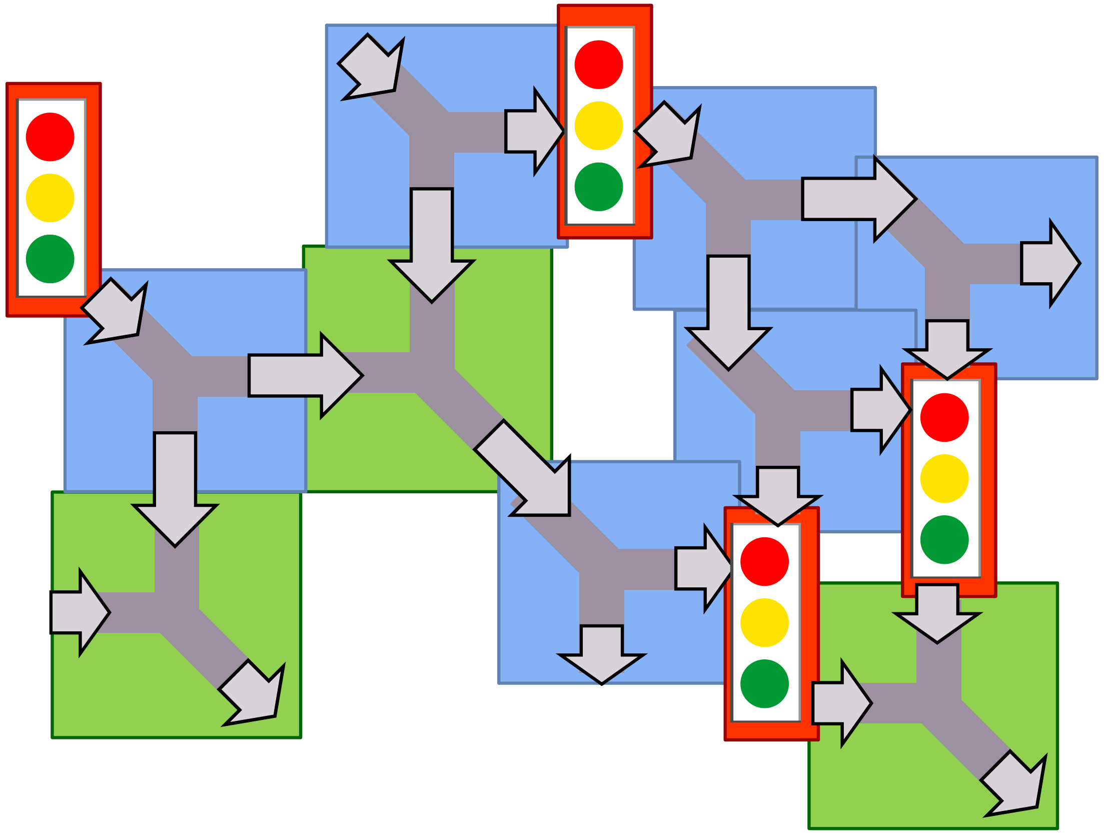
Challenges

Real systems are large

Any change to the model requires full re-verification

Systems often consist of multiple similar patterns

Component-based modeling



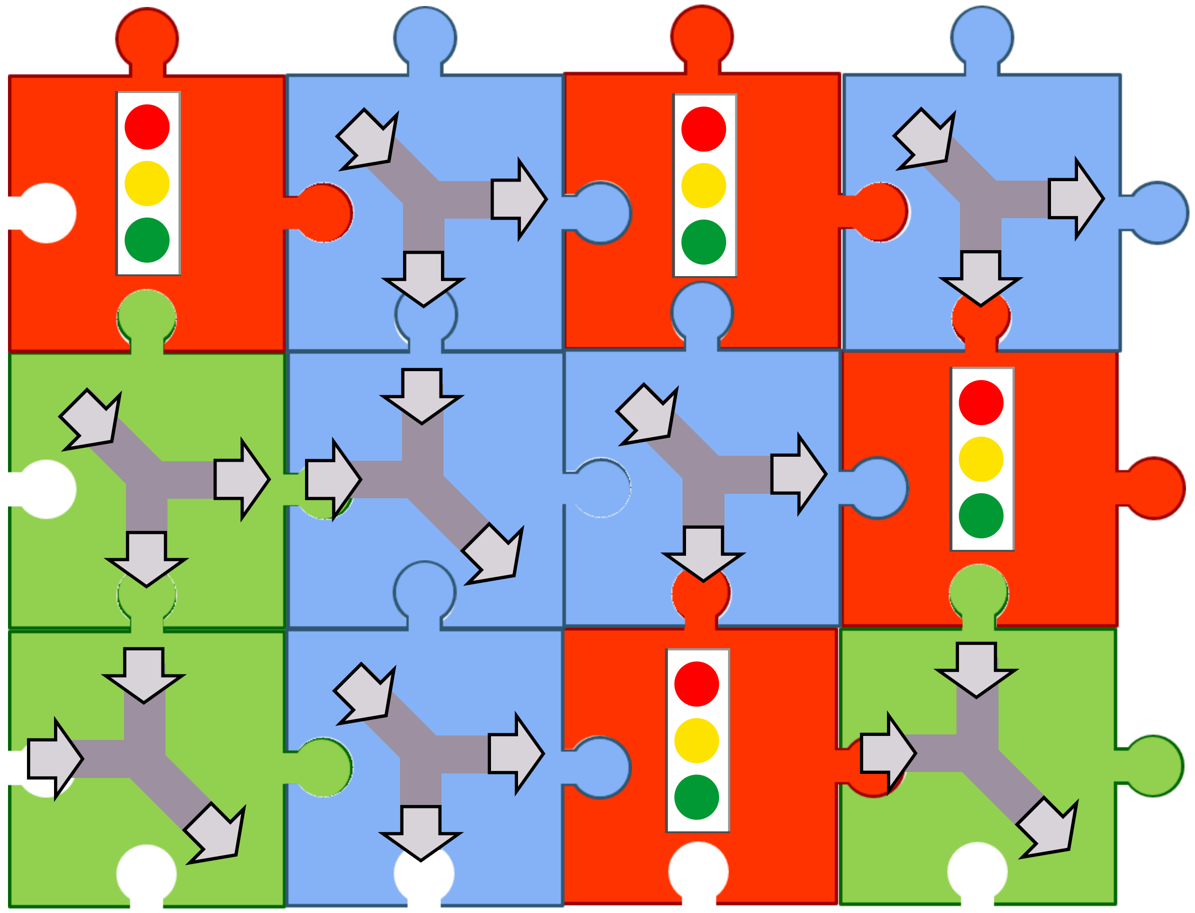
Challenges

Real systems are large

Any change to the model requires full re-verification

Systems often consist of multiple similar patterns

Component-based modeling



Challenges

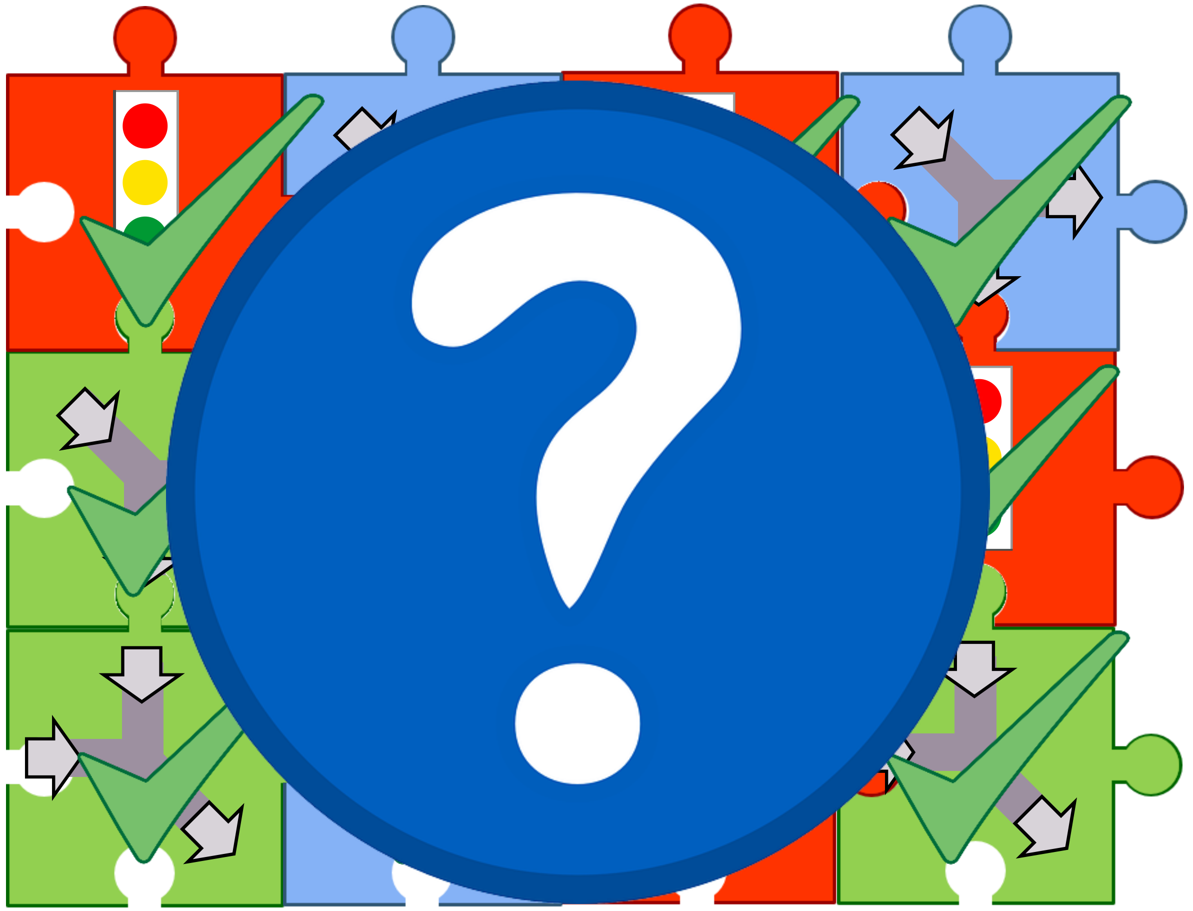
Real systems are large

Any change to the model requires full re-verification

Systems often consist of multiple similar patterns

Component-based modeling

- Verified components do not necessarily entail verified system



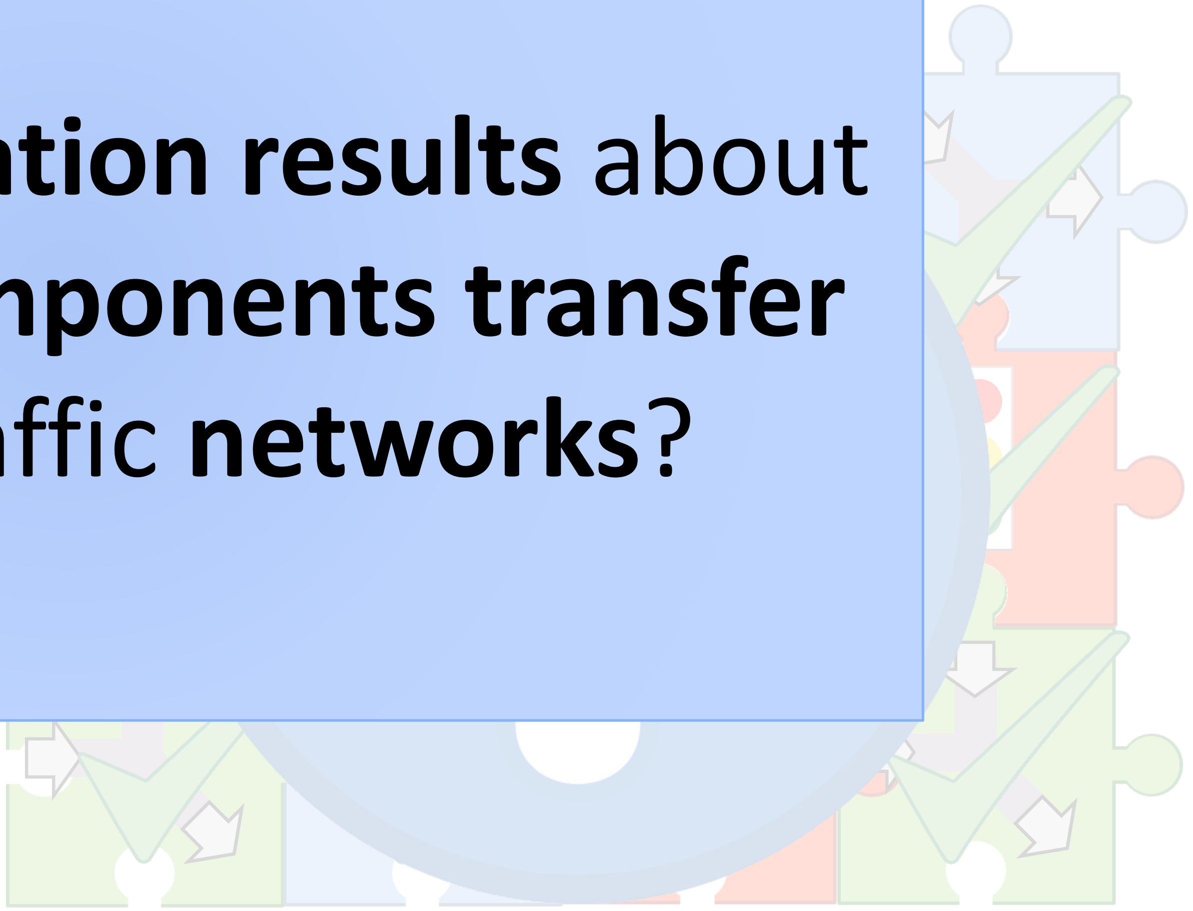
Challenges

Real systems are large

Any change
requires ful

Systems oft
multiple sir

Component-based modeling



How do **verification results** about traffic flow **components transfer** to entire traffic **networks**?

Approach

Component-based Verification

- Verified Components and Verified Composition
- Composition comes down to arithmetic checks

Process

(1) Model component types

(2) Verify safety conditions for each type and their composition

- No traffic breakdown

(3) Compose component instances to form system model

- Check arithmetic constraints

Result

- Fully verified system model

Approach

Component-based Verification

- Verified Components and Verified Composition
- Composition comes down to arithmetic checks

Process

(1) Model component types

(2) Verify safety conditions for each type and their composition

- No traffic breakdown

(3) Compose component instances to form system model

- Check arithmetic constraints

- Once per type
- Verification expert

- Once per network
- Traffic expert

Result

- Fully verified system model

Approach – Components

Generic component

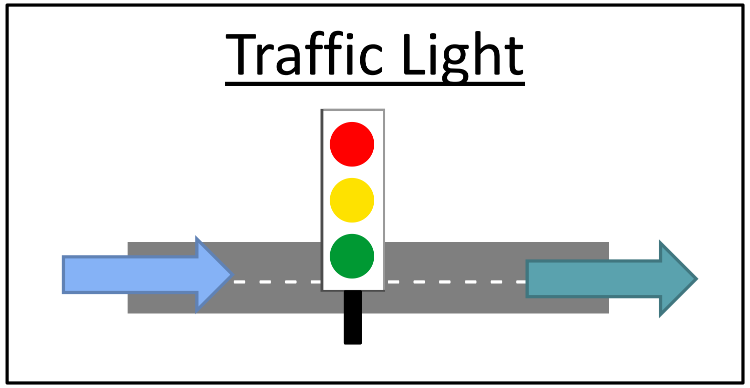
- Inflows
  (load, capacity, actual, max)
- Outflows
  (actual, max)
- Controller

Approach – Components

Generic component

- Inflows
  (load, capacity, actual, max)
- Outflows
  (actual, max)
- Controller

Example:

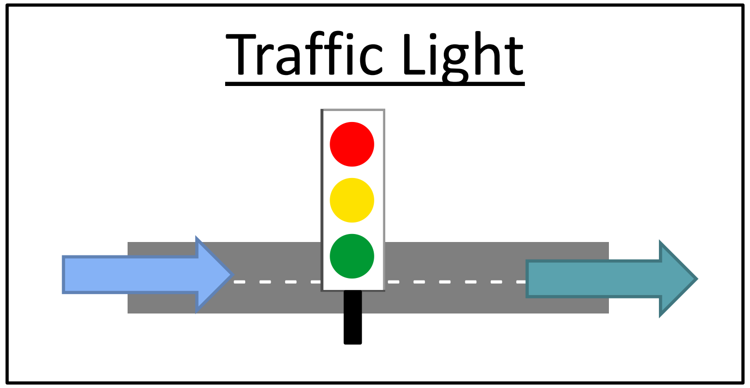


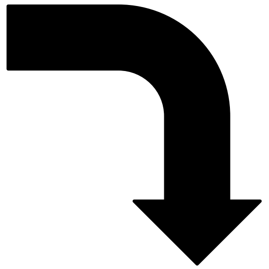
Approach – Components

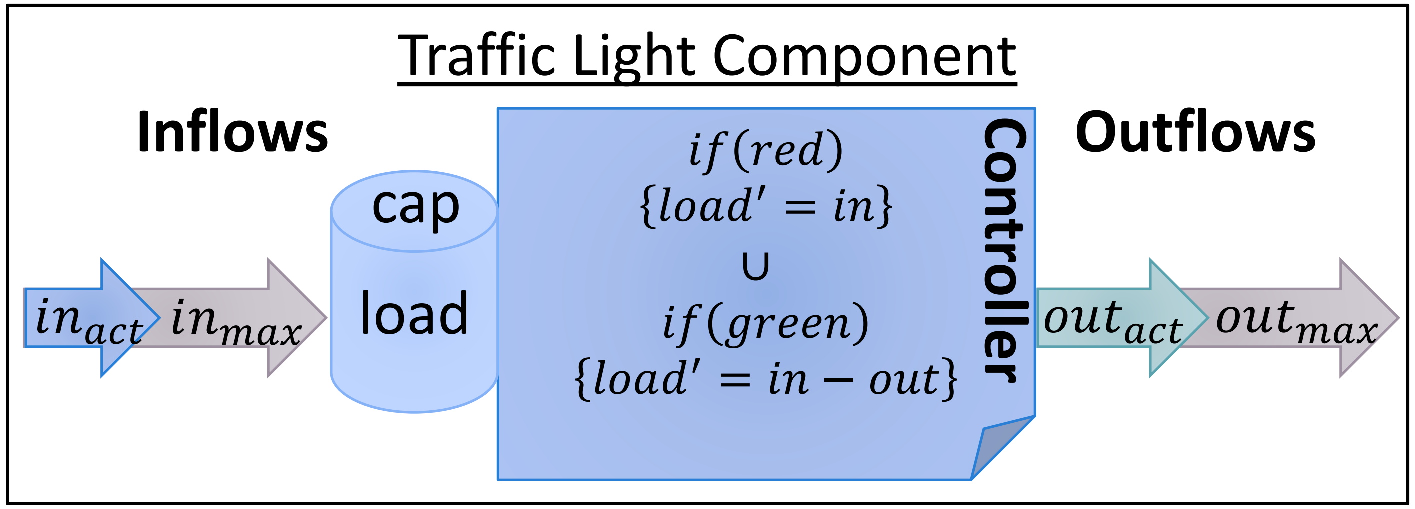
Generic component

- Inflows
  (load, capacity, actual, max)
- Outflows
  (actual, max)
- Controller

Example:







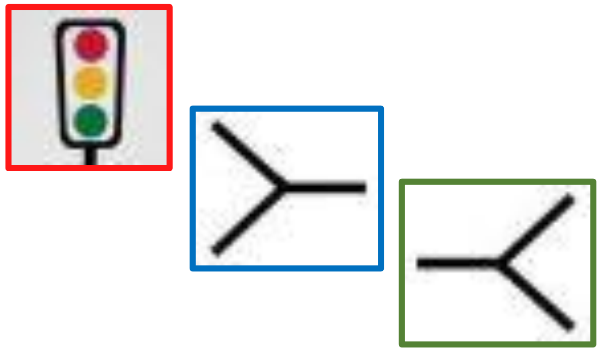
Approach – Components

Generic component

- Inflows
  (load, capacity, actual, max)
- Outflows
  (actual, max)
- Controller

Component types

- Traffic light (one in, one out)
- Flow merge (two in, one out)
- Flow split (one in, two out)



Approach – Safety Properties

Safety Property: No traffic breakdown occurs

- No load ever exceeds its capacity
- Must once be verified for each component type

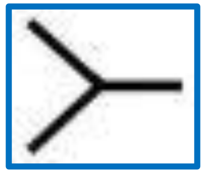
Approach – Safety Properties

Safety Property: No traffic breakdown occurs

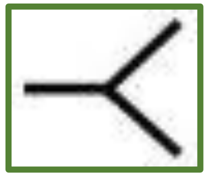
- No load ever exceeds its capacity
- Must once be verified for each component type

Contracts

$$cap \geq \max\left(T_{rg} * i_{max}, T * i_{max} - \max\left(0, o_{max} * \frac{T - T_{rg}}{2}\right)\right) \rightarrow [hp_{tl}]\,(t \leq T \rightarrow load \leq cap)$$



$$cap1 \geq T * i1_{max} \wedge cap2 \geq T * i2_{max} \rightarrow [hp_m]\left(t \leq T \rightarrow (load1 \leq cap1 \wedge load2 \leq cap2)\right)$$



$$cap \geq \max\left(0, T * (i_{max} - \min(o1_{max}, o2_{max}))\right) \rightarrow [hp_s]\,(t \leq T \rightarrow load \leq cap)$$

Approach – Safety Properties

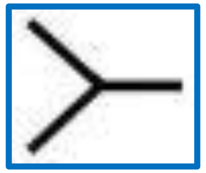
Safety Property: No traffic breakdown occurs

- No load ever exceeds its capacity
- Must once be verified for each component type

Contracts

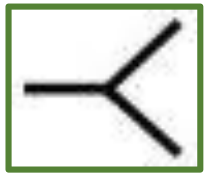
$cap \geq \max\Big(T_{rg} * i_{max}, T * i_{ma}$ … $\max\Big(0, o$ … $* \frac{T - T_{rg}}{2}\Big)\Big) \rightarrow [hp_{tl}]\,(t \leq T \rightarrow load \leq cap)$

Verified in KeYmaera



$cap1 \geq T * i1_{max} \land cap2 \geq T * i2$ … $[hp_m]\,(t \leq$ … $\rightarrow (load1 \leq cap1 \land load2 \leq cap2))$

Verified in KeYmaera



$cap \geq \max(0, T * (i_{max} - \min(o1_{max}, o2_{max}$ … $\rightarrow [hp_s]\,(t \leq T \rightarrow load \leq cap)$

Verified in KeYmaera

Approach – Composition

Compose components

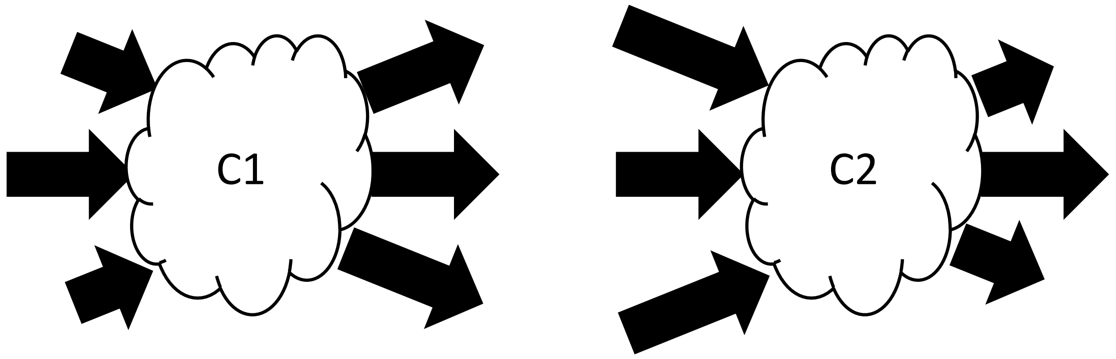
- Connect Outputs to Inputs
- Flow is passed on

- $\leq o_{max}$
- Both components safe

→Composition is again a safe component

Rebuild overall network

- Compose components until desired network is rebuilt
- Check if condition fulfilled



Approach – Composition

Compose components

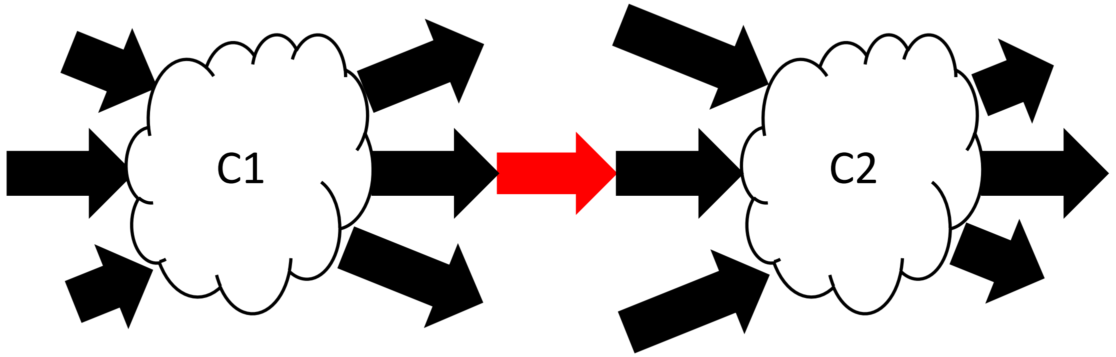
- Connect Outputs to Inputs
- Flow is passed on

- $\leq o_{max}$
- Both components safe

→Composition is again a safe component

Rebuild overall network

- Compose components until desired network is rebuilt
- Check if condition fulfilled



Approach – Composition

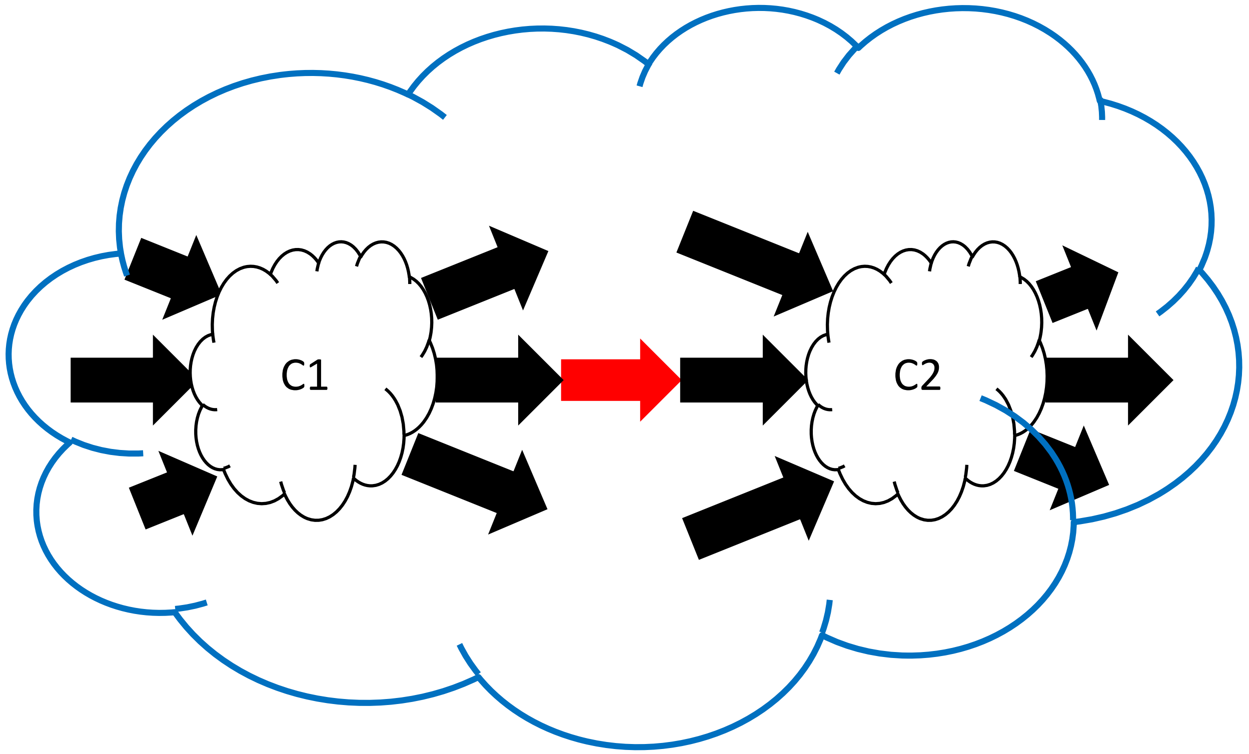
Compose components

- Connect Outputs to Inputs
- Flow is passed on
- $\leq o_{max}$
- Both components safe

→Composition is again a safe component

Rebuild overall network

- Compose components until desired network is rebuilt
- Check if condition fulfilled



Approach – Composition

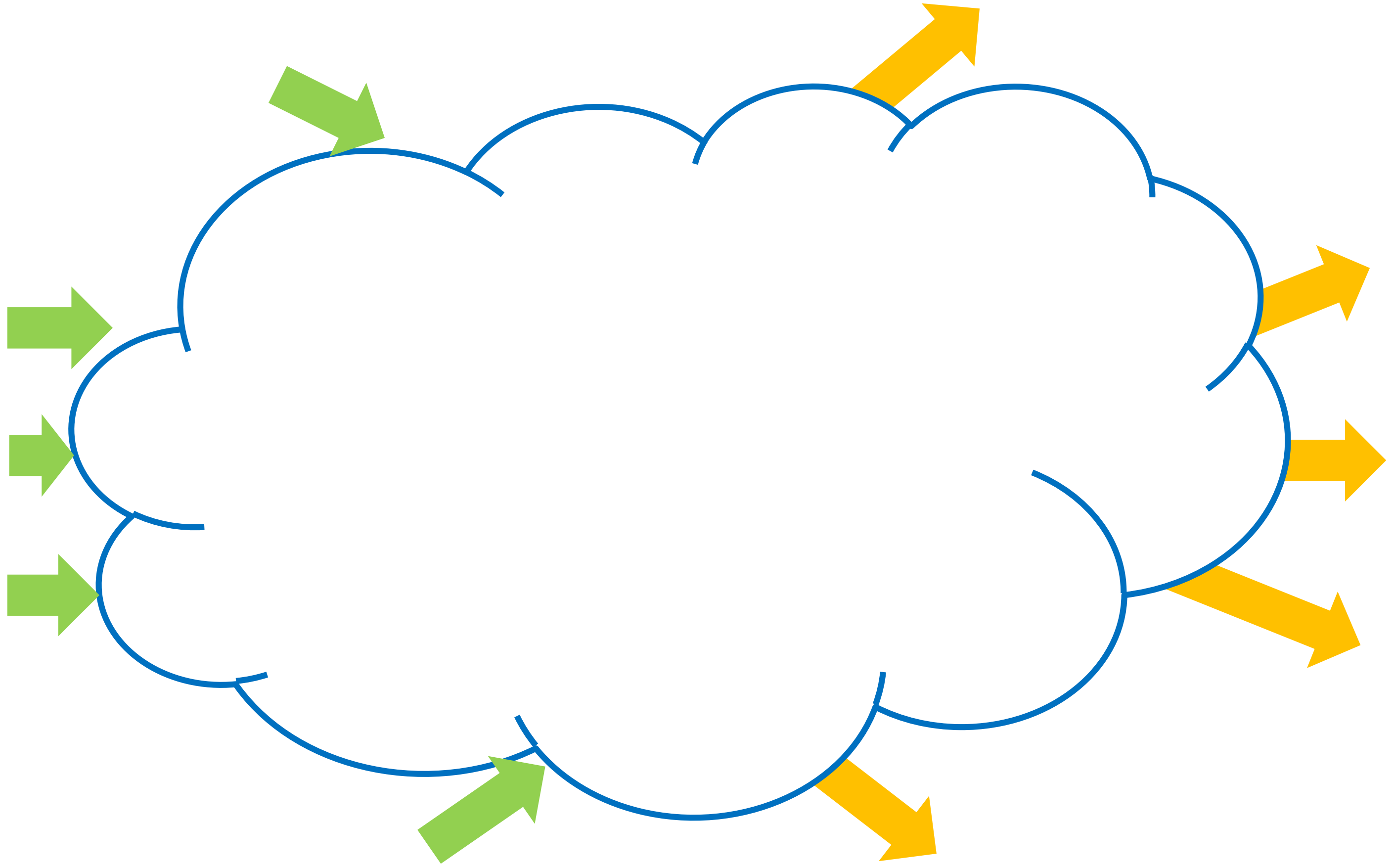
Compose components

- Connect Outputs to Inputs
- Flow is passed on
- $\leq o_{max}$
- Both components safe

→Composition is again a safe component

Rebuild overall network

- Compose components until desired network is rebuilt
- Check if condition fulfilled



Approach – Composition

o max oo o max mmaaxx o max

Compose components

- Connect Outputs to Inputs
- Flow is passed on

Theorem: Preserve Safety

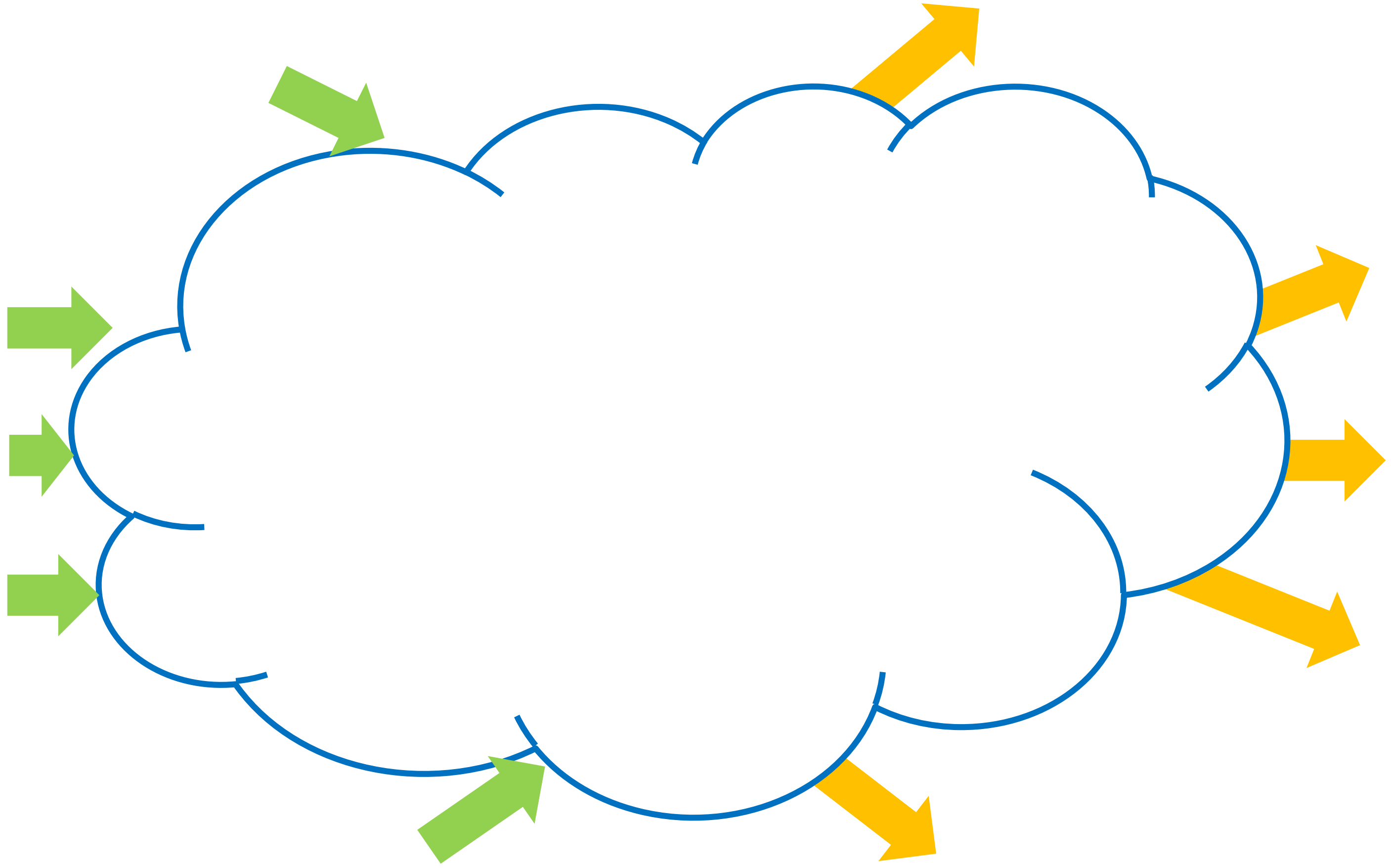
- Both components safe

→Composition is again a safe component

→Composition is again a safe component

Rebuild overall network

- Compose components until desired network is rebuilt
- Check if condition fulfilled



Approach – Composition

o max oo o max mmaaxx o max

Compose components

- Connect Outputs to Inputs
- Flow is passed on

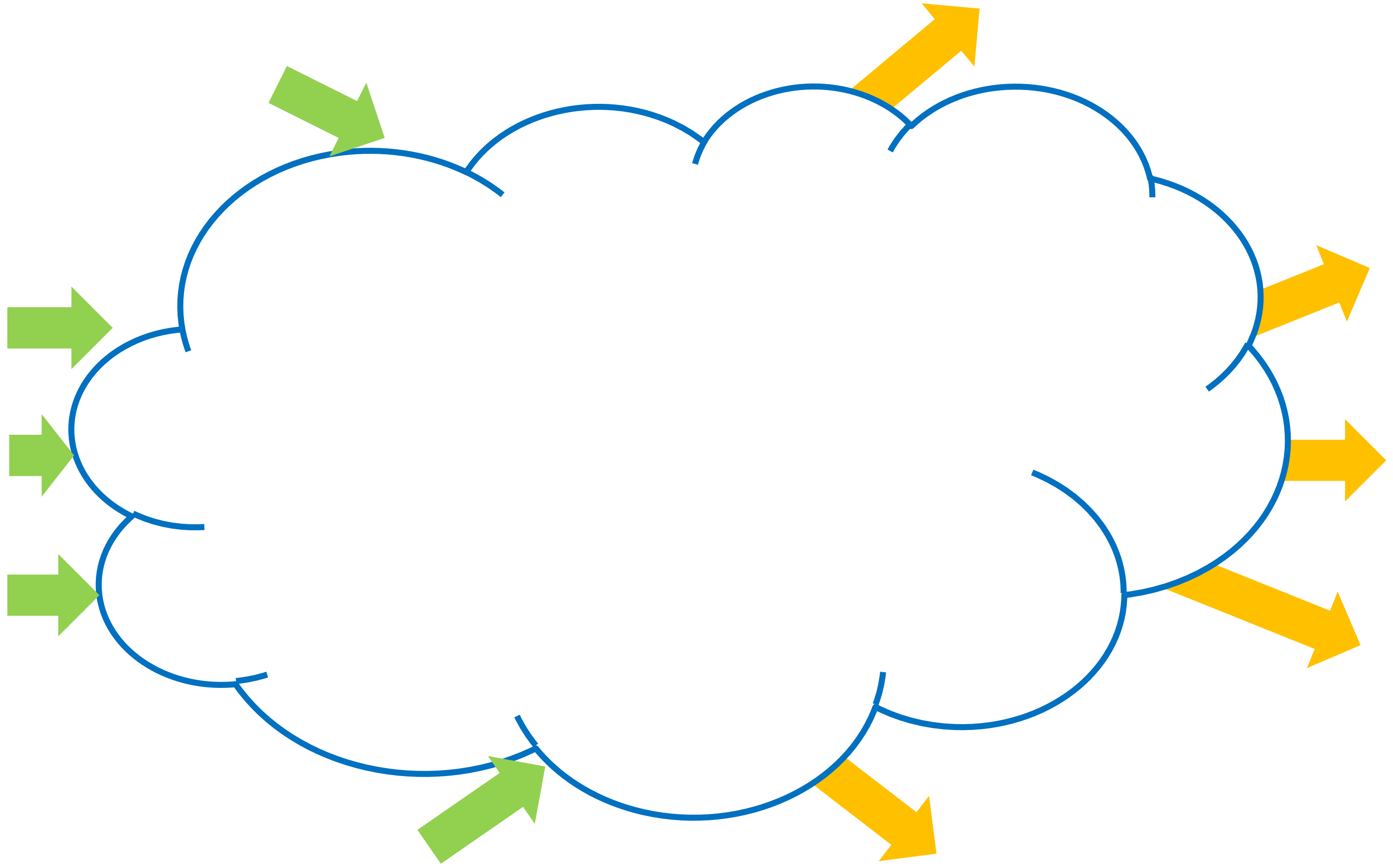
Theorem: Preserve Safety

- Both components safe

→Composition is again a safe component

Rebuild overall network

- Compose components until desired network is rebuilt
- Check if condition fulfilled
- Check if condition fulfilled

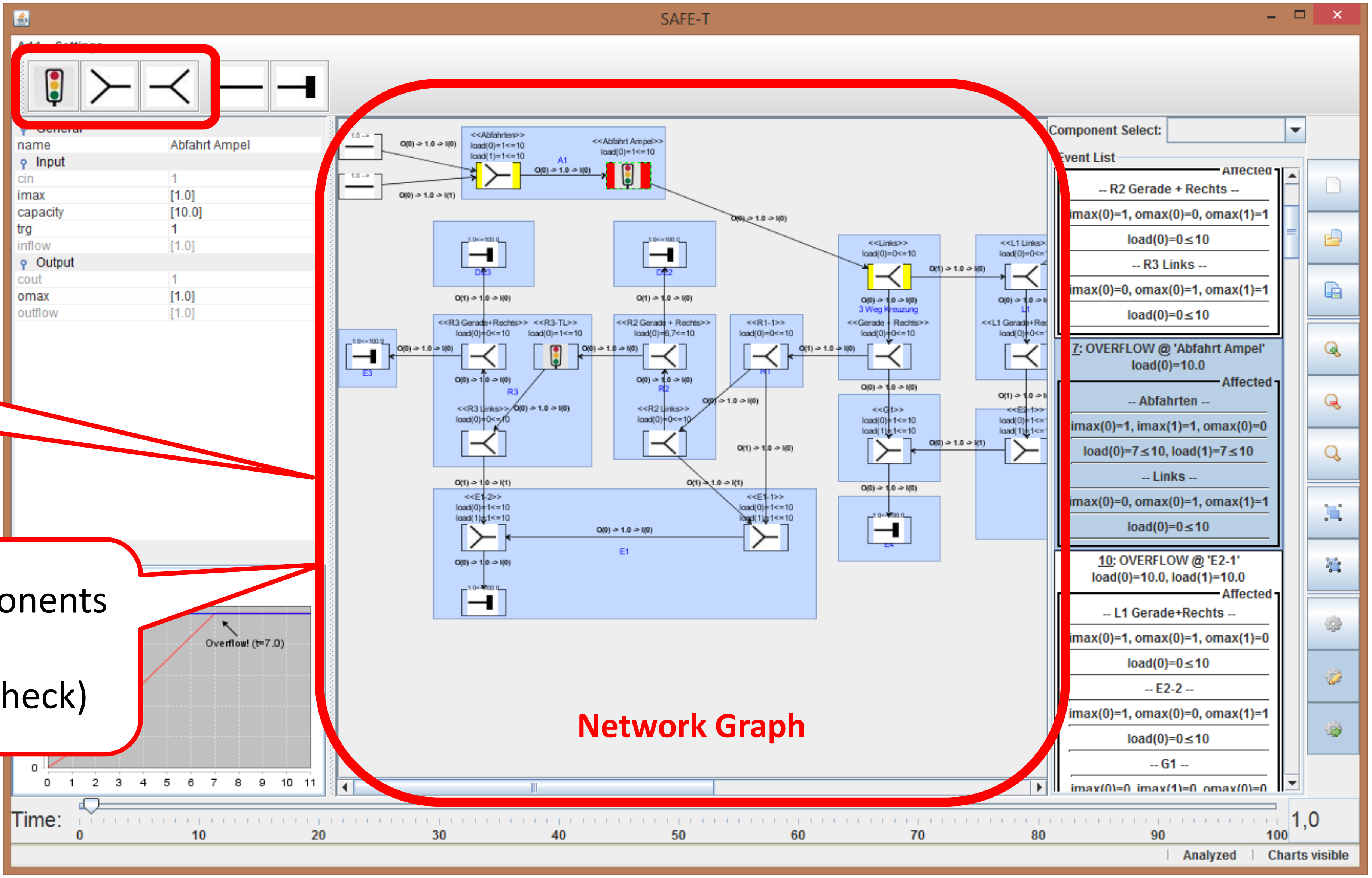


Implementation – SAFE-T

Extensible Component Library

Add components

Connect components (automatic compatibility check)



General

name	Abfahrt Ampel
Input	
cin	1
imax	[1.0]
capacity	[10.0]
trg	1
inflow	[1.0]
Output	
cout	1
omax	[1.0]
outflow	[1.0]

Component Select:

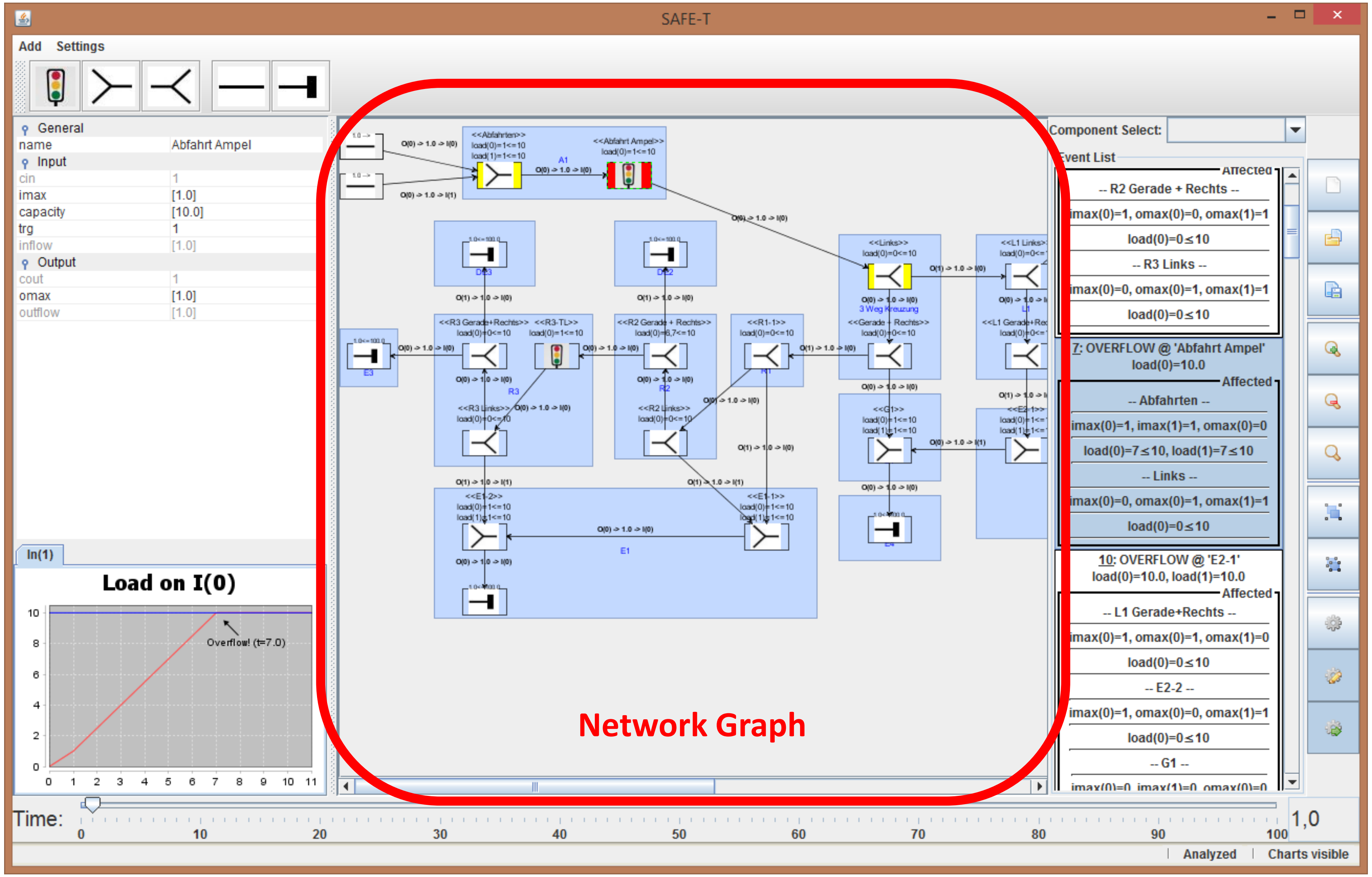
Event List

Affected
-- R2 Gerade + Rechts --
imax(0)=1, omax(0)=0, omax(1)=1
load(0)=0≤10
-- R3 Links --
imax(0)=0, omax(0)=1, omax(1)=1
load(0)=0≤10

7: OVERFLOW @ 'Abfahrt Ampel'
load(0)=10.0
Affected
-- Abfahrten --
imax(0)=1, imax(1)=1, omax(0)=0
load(0)=7≤10, load(1)=7≤10
-- Links --
imax(0)=0, omax(0)=1, omax(1)=1
load(0)=0≤10

10: OVERFLOW @ 'E2-1'
load(0)=10.0, load(1)=10.0
Affected
-- L1 Gerade+Rechts --
imax(0)=1, omax(0)=1, omax(1)=0
load(0)=0≤10
-- E2-2 --
imax(0)=1, omax(0)=0, omax(1)=1
load(0)=0≤10
-- G1 --
imax(0)=0, imax(1)=0, omax(0)=0

Implementation – SAFE-T



General	
name	Abfahrt Ampel
Input	
cin	1
imax	[1.0]
capacity	[10.0]
trg	1
inflow	[1.0]
Output	
cout	1
omax	[1.0]
outflow	[1.0]

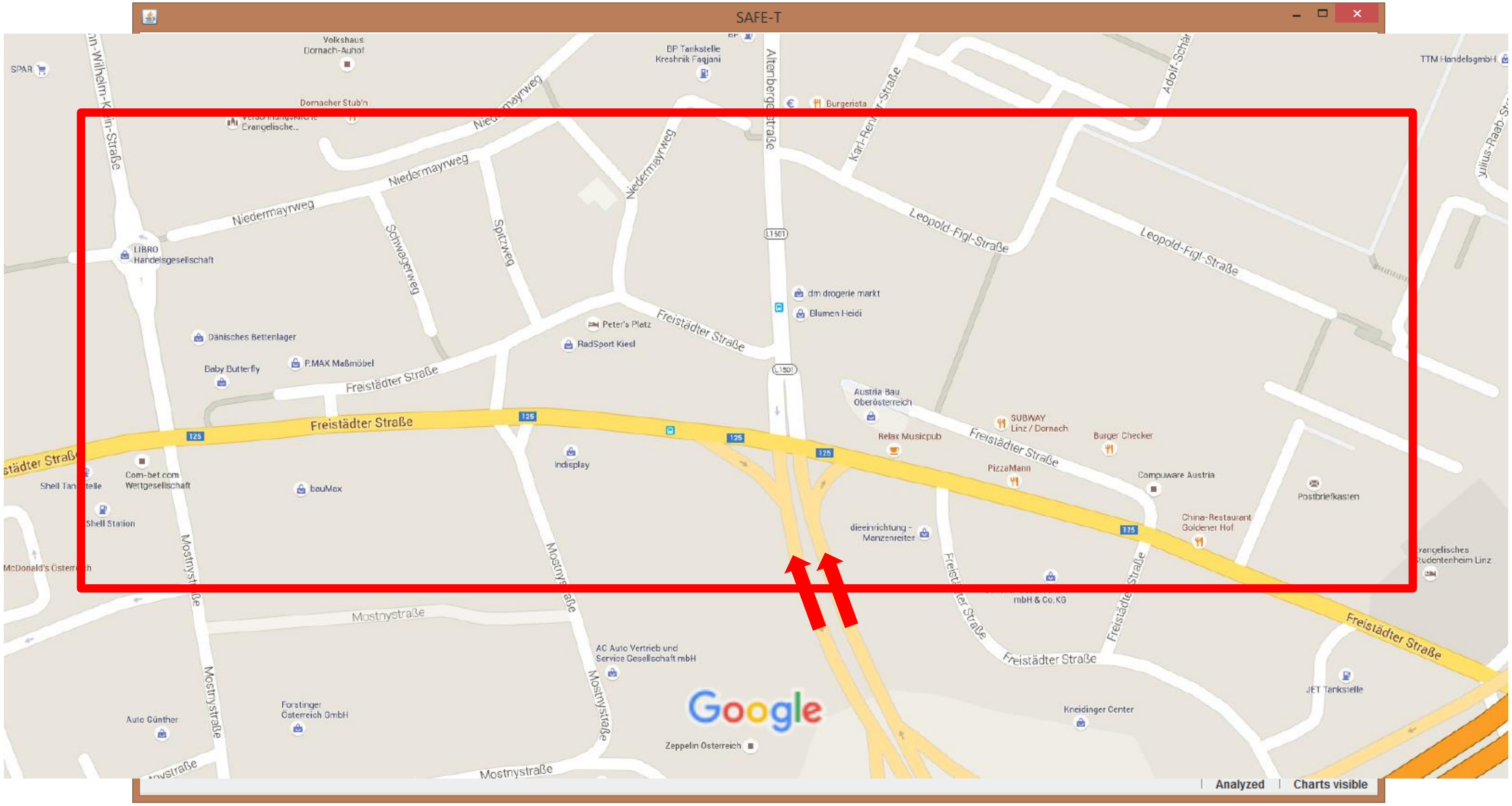
Affected
-- R2 Gerade + Rechts --
imax(0)=1, omax(0)=0, omax(1)=1
load(0)=0≤10
-- R3 Links --
imax(0)=0, omax(0)=1, omax(1)=1
load(0)=0≤10

7: OVERFLOW @ 'Abfahrt Ampel'
load(0)=10.0
Affected
-- Abfahrten --
imax(0)=1, imax(1)=1, omax(0)=0
load(0)=7≤10, load(1)=7≤10
-- Links --
imax(0)=0, omax(0)=1, omax(1)=1
load(0)=0≤10

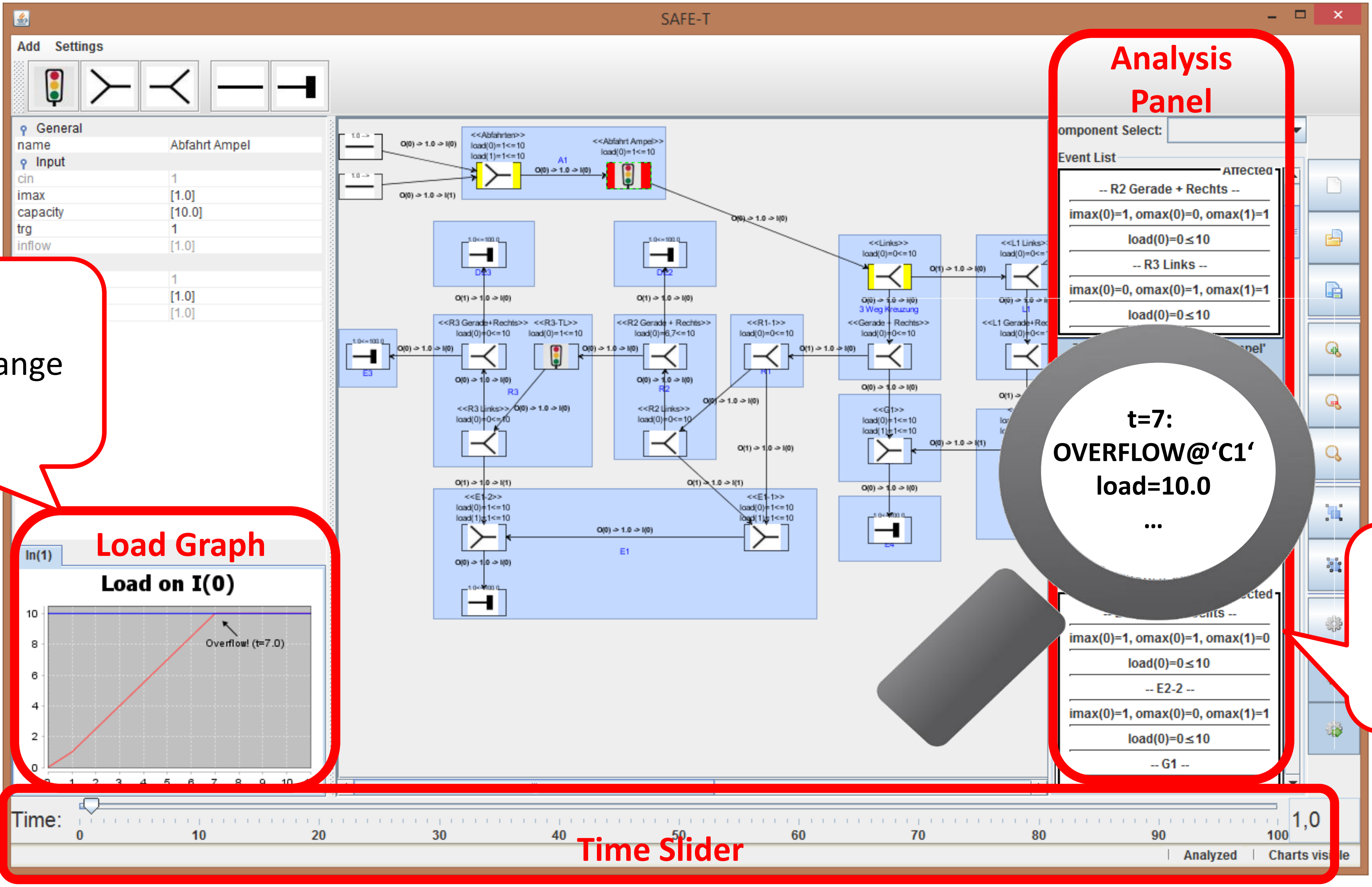
10: OVERFLOW @ 'E2-1'
load(0)=10.0, load(1)=10.0
Affected
-- L1 Gerade+Rechts --
imax(0)=1, omax(0)=1, omax(1)=0
load(0)=0≤10
-- E2-2 --
imax(0)=1, omax(0)=0, omax(1)=1
load(0)=0≤10
-- G1 --
imax(0)=0, imax(1)=0, omax(0)=0

Network Graph

Implementation – SAFE-T



Implementation – SAFE-T



-- R2 Gerade + Rechts --
imax(0)=1, omax(0)=0, omax(1)=1
load(0)=0≤10
-- R3 Links --
imax(0)=0, omax(0)=1, omax(1)=1
load(0)=0≤10

t=7:
OVERFLOW@'C1'
load=10.0
...

imax(0)=1, omax(0)=1, omax(1)=0
load(0)=0≤10
-- E2-2 --
imax(0)=1, omax(0)=0, omax(1)=1
load(0)=0≤10
-- G1 --

Simulate Model: How do loads change over time?

Analyze model: How long is it safe?

Analyze model: Which components overflows first?

Conclusion

Traffic Network

- X Traffic lights
- Y Flow Splits
- Z Flow Merges
- N Connections

		Monolithic	Component-based
Number of Proofs		1 (presumably large)	3 + N Checks (traffic light/split/merge)
Model Size	# Variables	X*6 + Y*6 + Z*7	6/6/7
	LoC	X*60 + Y*50 + Z*50	60/50/50
Connect…	…Components	Reproof of Composite	Arithmetic Check
Change…	…Component or Properties	Reproof Entire Model	Redo Arithmetic Checks
	…Connections	Reproof Entire Model	Redo Arithmetic Checks
Add…	…Component Type	Reproof Entire Model	Reproof Component Model

Conclusion

Example Network

- 5 Traffic lights
- 5 Flow Splits
- 5 Flow Merges
- 10 Connections

		Monolithic	Component-based
Number of Proofs		1	3 + 10 Checks (traffic light/split/merge)
Model Size	# Variables	95	6/6/7
	LoC	800	60/50/50
Connect…	…Components	Reproof of Composite	Arithmetic Check
Change…	…Component or Properties	Reproof Entire Model	Redo Arithmetic Checks
	…Connections	Reproof Entire Model	Redo Arithmetic Checks
Add…	…Component Type	Reproof Entire Model	Reproof Component Model

Conclusion

Example Network

- 5 Traffi
- 5 Flow

			-based
Number of Proofs			ecks lit/merge)
Model Size	# V		
	Lo		50
Connect…	…C		Check
Change…	…C		ic Checks
	…C		ic Checks
Add…	…Component Type	Reproof Entire Model	Reproof Component Model

Advantages

- Small proofs & checks instead of one huge proof
- Increased reusability
- Easy model evolution

Limitation

- Simplified models

Verified Traffic Networks: Component-based Verification of Cyber-Physical Flow Systems

THANKS FOR YOUR ATTENTION!

Related Work

Component-based CPS modeling and verification

- Few handle discrete and continuous CPS aspects
- Formal verification is not considered
- E.g.: Damm et al. [1], Henzinger et al. [2]

Traffic models

- Plethora of models
- Mostly purely continuous
- Verification not considered
- E.g.: Greenshields et al. [3], Lighthill et al. [4]

Intelligent traffic management systems

- Support traffic operators
- Complementary to our approach
- E.g.: Baumgartner et al. [5], Almejalli et al. [6]

[1] Damm, W.; et al. (2010): Towards Component Based Design of Hybrid Systems: Safety and Stability. In: *Time for Verification*. Springer Berlin Heidelberg.
[2] Henzinger, T.; et al. (2001): Assume-Guarantee Reasoning for Hierarchical Hybrid Systems. In: *Hybrid Systems: Computation and Control.* Springer Berlin Heidelberg.
[3] Greenshields, B. D.; et al. (1933): The Photographic Method of Studying Traffic Behavior. In: *Proceedings of the 13th Annual Meeting of the Highway Research Board.*
[4] Lighthill, M. J.;et al. (1955): On Kinematic Waves. II. A Theory of Traffic Flow on Long Crowded Roads. In: *Proceedings of the Royal Society of London*.
[5] Baumgartner, N.; et al. (2014): A Tour of BeAware! – A situation awareness framework for control centers. In: *Information Fusion 20.*
[6] Almejalli, K.; et al. (2007): Intelligent Traffic Control Decision Support System. In: Applications of Evolutionary Computing. Springer Berlin Heidelberg.

Future Work

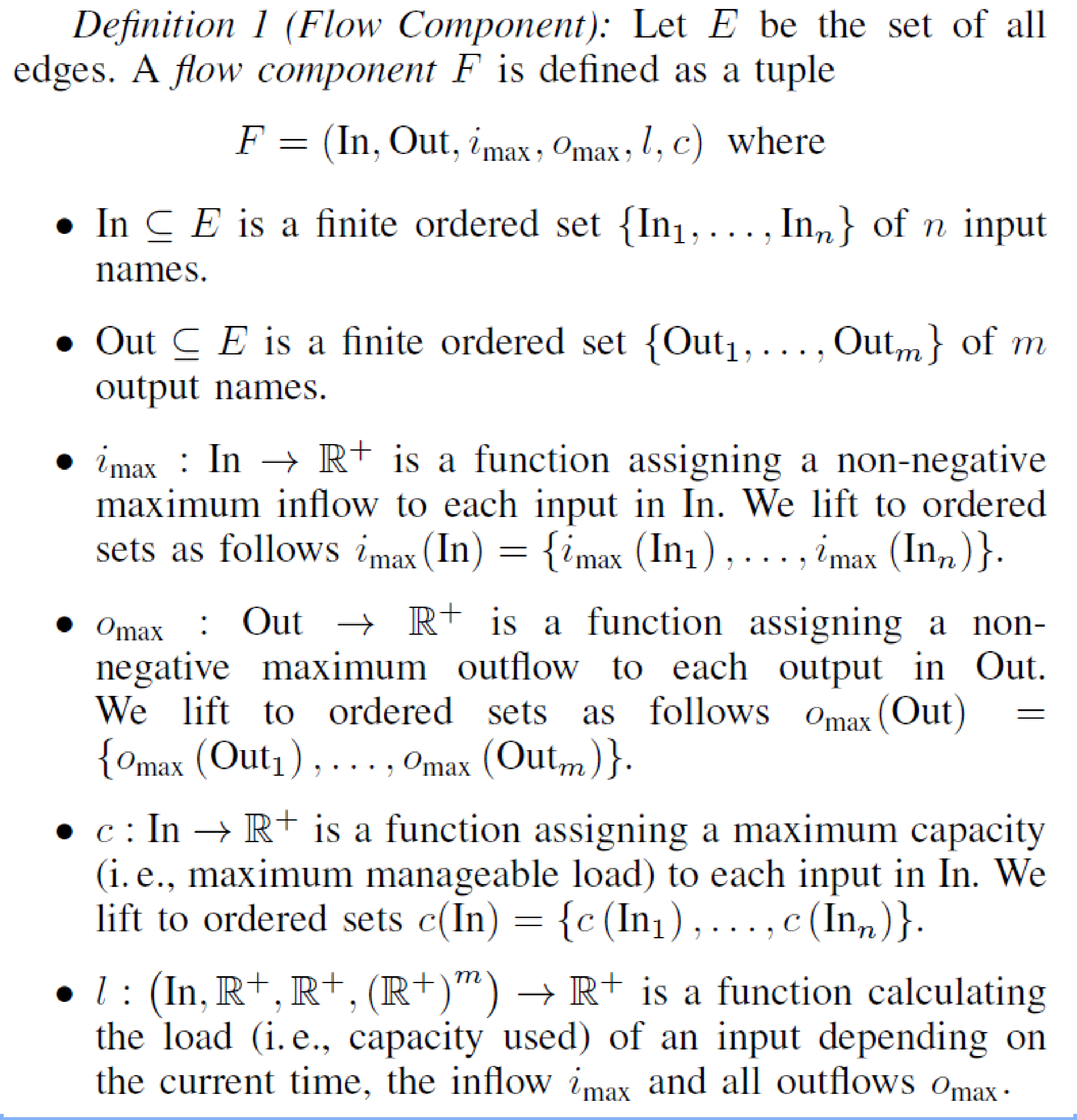
Consider traffic phenomena (e.g., shock-waves)

Introduce further components

Automatically transform networks into components and compositions

Generic Component Definitions

- Currently work-in-progress



Definition 1 (Flow Component): Let E be the set of all edges. A *flow component* F is defined as a tuple

$$F = (\mathrm{In}, \mathrm{Out}, i_{\max}, o_{\max}, l, c) \text{ where}$$

- $\mathrm{In} \subseteq E$ is a finite ordered set $\{\mathrm{In}_1, \ldots, \mathrm{In}_n\}$ of n input names.
- $\mathrm{Out} \subseteq E$ is a finite ordered set $\{\mathrm{Out}_1, \ldots, \mathrm{Out}_m\}$ of m output names.
- $i_{\max} : \mathrm{In} \rightarrow \mathbb{R}^+$ is a function assigning a non-negative maximum inflow to each input in In. We lift to ordered sets as follows $i_{\max}(\mathrm{In}) = \{i_{\max}(\mathrm{In}_1), \ldots, i_{\max}(\mathrm{In}_n)\}$.
- $o_{\max} : \mathrm{Out} \rightarrow \mathbb{R}^+$ is a function assigning a non-negative maximum outflow to each output in Out. We lift to ordered sets as follows $o_{\max}(\mathrm{Out}) = \{o_{\max}(\mathrm{Out}_1), \ldots, o_{\max}(\mathrm{Out}_m)\}$.
- $c : \mathrm{In} \rightarrow \mathbb{R}^+$ is a function assigning a maximum capacity (i. e., maximum manageable load) to each input in In. We lift to ordered sets $c(\mathrm{In}) = \{c(\mathrm{In}_1), \ldots, c(\mathrm{In}_n)\}$.
- $l : (\mathrm{In}, \mathbb{R}^+, \mathbb{R}^+, (\mathbb{R}^+)^m) \rightarrow \mathbb{R}^+$ is a function calculating the load (i. e., capacity used) of an input depending on the current time, the inflow $i_{\max}$ and all outflows $o_{\max}$.

Definition 3 (Sequential Composition): Let

$$F^s = (\mathrm{In}^s, \mathrm{Out}^s, i^s_{\max}, o^s_{\max}, l^s, c^s), \text{ for } s \in \{1,2\}$$

be flow-components, with disjoint inputs and outputs (i. e., $\mathrm{In}^1 \cap \mathrm{In}^2 = \mathrm{Out}^1 \cap \mathrm{Out}^2 = \emptyset$) and $\mathcal{C} : \mathrm{Out}^1 \rightharpoonup \mathrm{In}^2$ be a partial (i. e., not every output must be mapped when connecting two components), injective (i. e., every input is only mapped to one output upon connection of components) function, mapping connected outputs and inputs between the two components. We define $\mathcal{O}$ as the domain of $\mathcal{C}$ (i. e., all values $x \in \mathrm{Out}^1$ such that $\mathcal{C}(x)$ is defined) and $\mathcal{I}$ as the the image of $\mathcal{C}$ (i. e., all values $y \in \mathrm{In}^2$ such that $y = \mathcal{C}(x)$ holds for some $x \in \mathrm{Out}^1$).

We define the sequential composition $F^3 = F^1 \bullet\!\!-\!\!\circ^{\mathcal{C}} F^2$ of flow components F^1 and F^2 by connecting outputs of F^1 to inputs of F^2 according to a function $\mathcal{C}$, with $|\mathcal{O}| > 0$, where

$$F^3 = (\mathrm{In}^3, \mathrm{Out}^3, i^3_{\max}, o^3_{\max}, l^3, c^3) \text{ with}$$

- $\mathrm{In}^3 = (\mathrm{In}^2 \setminus \mathcal{I}) \cup \mathrm{In}^1$
- $\mathrm{Out}^3 = \mathrm{Out}^2 \cup (\mathrm{Out}^1 \setminus \mathcal{O})$
- $n_3 = |\mathrm{In}^3| = |\mathrm{In}^1| + |\mathrm{In}^2| - |\mathcal{C}|$ and
  $m_3 = |\mathrm{Out}^3| = |\mathrm{Out}^1| + |\mathrm{Out}^2| - |\mathcal{C}|$
- $i^3_{\max} : \mathrm{In} \to \mathbb{R}^+$, with
  $\forall \mathrm{In}_k \in \mathrm{In}^1 \;.\; i^3_{\max}(\mathrm{In}_k) = i^1_{\max}(\mathrm{In}_k)$ and
  $\forall \mathrm{In}_l \in \mathrm{In}^2 \cap \mathrm{In}^3 \;.\; i^3_{\max}(\mathrm{In}_l) = i^2_{\max}(\mathrm{In}_l)$
- $o^3_{\max} : \mathrm{Out} \to \mathbb{R}^+$, with
  $\forall \mathrm{Out}_k \in \mathrm{Out}^1 \cap \mathrm{Out}^3 \;.\; o^3_{\max}(\mathrm{Out}_k) = o^1_{\max}(\mathrm{Out}_k)$ and
  $\forall \mathrm{Out}_l \in \mathrm{Out}^2 \;.\; o^3_{\max}(\mathrm{Out}_l) = o^2_{\max}(\mathrm{Out}_l)$,
- $l^3 : (\mathrm{In}, \mathbb{R}^+, \mathbb{R}^+, (\mathbb{R}^+)^{m_3}) \to \mathbb{R}^+$, with
  $$\forall \mathrm{In}_k \in \mathrm{In}^1 \;.\; l^3\left(\mathrm{In}_k, t, i^3_{\max}(\mathrm{In}_k), o^3_{\max}(\mathrm{Out}^1)\right) = l^1\left(\mathrm{In}_k, t, i^3_{\max}(\mathrm{In}_k), o^3_{\max}(\mathrm{Out}^1)\right) \text{ and}$$
  $$\forall \mathrm{In}_l \in \mathrm{In}^2 \cap \mathrm{In}^3 \;.\; l^3\left(\mathrm{In}_l, t, i^2_{\max}(\mathrm{In}_l), o^3_{\max}(\mathrm{Out}^2)\right) = l^2\left(\mathrm{In}_l, t, i^2_{\max}(\mathrm{In}_l), o^3_{\max}(\mathrm{Out}^2)\right)$$
- $c^3 : \mathrm{In} \to \mathbb{R}^+$, with $\forall \mathrm{In}_k \in \mathrm{In}^1 \;.\; c^3(\mathrm{In}_k) = c^1(\mathrm{In}_k)$ and
  $\forall \mathrm{In}_l \in \mathrm{In}^2 \cap \mathrm{In}^3 \;.\; c^3(\mathrm{In}_l) = c^2(\mathrm{In}_l)$.

Model 1 Traffic flow in a traffic light

$$tl \equiv (ctrl_{tl}; plant_{tl})^* \quad (1)$$
$$ctrl_{tl} \equiv \text{if } (t_c = T_c) \text{ then } t_c := 0;\ go := (go-1)^2 \text{ fi} \quad (2)$$
$$i_{act} := *;\ ?(0 \le i_{act} \le i_{max}); \quad (3)$$
$$\text{if } (l > 0) \text{ then } o_{act} := o_{max} \quad (4)$$
$$\text{else } o_{act} := \min(i_{act}, o_{max}) \text{ fi}; \quad (5)$$
$$plant_{tl} \equiv l' = i_{act} - o_{act} \cdot go, t' = 1, t_c' = 1 \quad (6)$$
$$\&\ t_c \le T_c \wedge l \ge 0 \quad (7)$$

Proposition 1 (Traffic Light Load Safety): We want the traffic light to be load-safe in order to avoid an overflow which would result in a traffic breakdown. A flow component with one input and one output is load-safe per Def. 2 if

$$l\left(\text{In}_1, t, i_{max}\left(\text{In}_1\right), \{o_{max}\left(\text{Out}_1\right)\}\right) \le c\left(\text{In}_1\right) \quad .$$

Thus, a traffic light is safe (ψ_{tl}) if it is load-safe for up to a maximum time T.

$$\psi_{tl} \equiv (t \le T \rightarrow l \le c)$$

When started in a safe initial state ϕ_{tl}, the traffic light component *tl* ensures load safety ψ_{tl}

$$\phi_{tl} \rightarrow [tl]\psi_{tl} \quad (8)$$

where

$$\phi_{tl} \equiv t = 0 \wedge 0 \le t_c \le T_c \wedge T_c > 0 \wedge T > 0 \wedge l = 0$$
$$\wedge\, c \ge \max(T_c \cdot i_{max}, T \cdot i_{max} - \max\left(0, o_{max} \cdot \frac{T - T_c}{2}\right))$$
$$\wedge\, 0 \le i_{max} \wedge 0 \le o_{max} \wedge go \cdot (go - 1) = 0 \quad .$$

Model 2 Traffic flow in a traffic flow merge component

```
$tfm \equiv (ctrl_{tfm};\ plant_{tfm})^*$   (9)
$ctrl_{tfm} \equiv road := *;\ ?(0 \le road \le 1);$   (10)
    $i1_{act} := *;\ ?(0 \le i1_{act} \le i1_{max});$   (11)
    $i2_{act} := *;\ ?(0 \le i2_{act} \le i2_{max});$   (12)
    if $(l1 > 0 \lor l2 > 0)$ then $o_{act} := o_{max}$   (13)
    else $o_{act} := \min(i1_{act} + i2_{act}, o_{max})$ fi;   (14)
$plant_{tfm} \equiv l1' = i1_{act} - o_{act} \cdot (1 - road), t' = 1,$   (15)
    $l2' = i2_{act} - o_{act} \cdot road \ \&\ l1 \ge 0 \land l2 \ge 0$   (16)
```

Proposition 2 (Merge Load Safety): We want the traffic flow merge component to be load-safe in order to avoid an overflow which would result in a traffic breakdown. A flow component with two inputs and one output is load safe if

$$l\left(\text{In}_i, t, i_{\max}\left(\text{In}_i\right), \left\{o_{\max}\left(\text{Out}_1\right)\right\}\right) \le c\left(\text{In}_i\right) \ \text{ for } i \in \{1,2\} \ .$$

Thus, a traffic flow merge is safe (ψ_{tfm}) if it is load-safe for up to a maximum time T:

$$\psi_{tfm} \equiv \left(t \le T \to (l1 \le c1 \land l2 \le c2)\right) \ .$$

A traffic flow merge component *tfm* ensures load safety ψ_{tfm}, cf. (17), when started in a safe initial state ϕ_{tfm} (18).

$$\phi_{tfm} \to [tfm]\psi_{tfm} \tag{17}$$

$$\begin{aligned}\phi_{tfm} \equiv\ & t = 0 \land 0 \le i1_{max} \land 0 \le i2_{max} \land 0 \le o_{max} \\ & \land c1 \ge T \cdot i1_{max} \land c2 \ge T \cdot i2_{max} \\ & \land l1 = l2 = 0 \land 0 \le road \le 1\end{aligned} \tag{18}$$

Model 3 Traffic flow in a traffic flow split component

$$
\begin{aligned}
\mathit{tfs} &\equiv (\mathit{ctrl}; \mathit{plant})^* && (19)\\
\mathit{ctrl}_{\mathit{tfs}} &\equiv i_{\text{act}} := *;\ ?(0 \le i_{\text{act}} \le i_{\max}); && (20)\\
&\quad \mathit{road} := *;\ ?(0 \le \mathit{road} \le 1); && (21)\\
&\quad \text{if } (l > 0) \text{ then } o1_{\text{act}} := o1_{\max}; o2_{\text{act}} := o2_{\max} && (22)\\
&\quad \text{else } o1_{\text{act}} := \min(i_{\text{act}}, o1_{\max}); && (23)\\
&\qquad o2_{\text{act}} := \min(i_{\text{act}}, o2_{\max}) \text{ fi}; && (24)\\
\mathit{plant}_{\mathit{tfs}} &\equiv l' = i_{\text{act}} - o1_{\text{act}} \cdot (1 - \mathit{road}) - o2_{\text{act}} \cdot \mathit{road}, && (25)\\
&\quad t' = 1 \ \&\ l \ge 0 && (26)
\end{aligned}
$$

Proposition 3 (Split Load Safety): We want traffic flow split components to be load-safe in order to avoid an overflow which would result in a traffic breakdown. A flow component with one input and two outputs is load-safe per Def. 2 if

$$l\left(\text{In}_1, t, i_{\max}\left(\text{In}_1\right), \{o_{\max}\left(\text{Out}_1\right), o_{\max}\left(\text{Out}_2\right)\}\right) \le c\left(\text{In}_1\right) \quad .$$

Thus, a traffic flow split component is safe ψ_{tfs} if it is load-safe for up to a maximum time T.

$$\psi_{\mathit{tfs}} \equiv (t \le T \to l \le c)$$

When started in a safe initial state ϕ_{tfs}, the traffic flow split component *tfs* ensures load safety ψ_{tfs}

$$\phi_{\mathit{tfs}} \to [\mathit{tfs}]\psi_{\mathit{tfs}} \qquad (27)$$

where

$$
\begin{aligned}
\phi_{\mathit{tfs}} \equiv t = 0 \wedge T > 0 \wedge 0 \le i_{\max} \wedge 0 \le o1_{\max} \wedge 0 \le o2_{\max}\\
\wedge\, c \ge \max\Big(0, T \cdot \left(i_{\max} - \min\left(o1_{\max}, o2_{\max}\right)\right)\Big)\\
\wedge\, l = 0 \wedge 0 \le \mathit{road} \le 1 \quad .
\end{aligned}
$$