Lightweight Formal Methods for the Development of High-Assurance Network Systems

Assaf Kfoury

with contributions from

Azer Bestavros, Adam Bradley, Andrei Lapets, and Michael Ocean

iBench Initiative
http://www.cs.bu.edu/groups/ibench/
snBench
http://csr.bu.edu/snbench/

Computer Science

Formal Methods in Network Studies …

1. **Compositional Analysis/Specification and its Benefits**
   (mostly with A. Bestavros)
2. **An Application of Model Checking: Safe Composition of Arbitrary Network Protocols**
   (mostly with A. Bradley and A. Bestavros)
3. **Resource Allocation in Sensor Networks using a Strongly-Typed Domain-Specific Language**
   (with M. Ocean and A. Bestavros)
4. **The Stable-Paths Problem and the Promise of an Automatic Lightweight Proof-Assistant**
   (with K. Donnelly and A. Lapets)

Formal Methods in Network Studies …

1. **Compositional Analysis/Specification and its Benefits (mostly with A. Bestavros)**
2. An Application of Model Checking: Safe Composition of Arbitrary Network Protocols (mostly with A. Bradley and A. Bestavros)
3. Resource Allocation in Sensor Networks using a Strongly-Typed Domain-Specific Language (with M. Ocean and A. Bestavros)
4. **The Stable-Paths Problem and the Promise of an Automatic Lightweight Proof-Assistant (with K. Donnelly and A. Lapets)**

time permitting

Notes

Many networking problems naturally lend themselves to graph-theoretic formulations that are far more complex than classical problems of graph theory. The modeling graphs are typically large, with hundreds or thousands or more nodes, often placed on a fluctuating grid, e.g., paths between nodes may fail or degrade or reappear, slowly or abruptly, and parameters regulating flow along paths may be conflicting requirements or defined differently at different nodes, locally or globally.

We propose a methodology for the specification and analysis of network systems which attempts to support such uneven features in the large graphs modeling them. The proposed methodology would allow for a **compositional** (as opposed to **whole-system**) analysis which is additionally incremental (distributed in time) and modular (distributed in space). Towards this goal, we leverage concepts and techniques rooted in mathematical logic and the formal methods of computer science. Our methodology calls for the definition of a strongly-typed domain-specific language to specify network configurations and the properties we wish to enforce. We pay special attention to keeping these concepts and techniques **lightweight**. i.e., making the parts available to users "friendly" (relatively simple and easy to use) while the more complicated parts remain hidden "under the roof". We illustrate these ideas with a particular application: the specification and analysis of vehicular traffic networks.

The preceding is only one approach out of many that promote the use of formal methods for a rigorous analysis of properties of network systems. Such approaches have been introduced by many researchers, including ourselves, in recent years. We conclude with a brief overview of other formal methodologies we have developed or refined from others.

Preamble

In a survey article more than a decade ago, E.M. Clarke and J.M. Wing identified 6 "fundamental concepts" that should be investigated further in "future directions" of research in *Formal Methods*. Of these six, 5 relate to our concerns in one way or another, now adapted to networking problems:

1. **composition**
2. **decomposition**
3. **abstraction**
4. **reusable models and theories**
5. **combination of mathematical theories**

"Formal Methods: State of the Art and Future Directions"
by E.M. Clarke and J.M. Wing,
***ACM Computing Surveys*, Vol. 28 No. 4, Dec 1996**

Notes on Clarke-Wing survey paper

Essentially paraphrasing Clarke and Wing:

Composition = compose methods, compose specifications, compose models, compose theories, and compose proofs.
Decomposition = develop efficient methods for decomposing computationally demanding global property into local properties whose verification is computationally simple.
Abstraction = real systems are difficult to specify/verify without abstraction. Identify different kinds of abstractions, tailored for certain kinds of systems or problem domains. Develop ways to justify them formally, possibly using automated help.
Reusable models and theories = rather than define models/theories from scratch each time a new application is tackled, develop formal means to have reusable and parametrized models/theories.
Combination of mathematical theories = many safety-critical systems have both digital and analog components. Hybrid systems require reasoning about both discrete and continuous mathematics.
Data structures and algorithms = develop new ones to handle larger search spaces and larger systems.

Overview

Central Theme: **Compositionality**

Overview

Central Theme: **Compositionality**

- not to overcome a state-explosion phenomenon, as in **Model Checking**,
- but rather to support partial/varying specification and distributivity in time/space, closer to compositionality as attempted in some approaches (mostly embryonic, many false starts) in the static analysis of programs.

Notes on Model Checking

Model Checking is a method for verifying that finite-state systems, such as sequential circuit designs and communication protocols, satisfy their specifications. Specifications are expressed as logic formulas, typically in temporal logic, and the system is modeled as a state-transition graph. Verification means to check whether every state in this graph satisfies the formula expressing the specification. For all its great successes, MC also has limitations:

(1) Expressing the specification as a temporal logic formula is not always straightforward, sometimes impossible. What if, for example, the specification is best expressed in the form of linear, or quadratic, or other algebraic constraints?

(2) Compositionality and other concepts (such as decompositionality, modularization, abstraction) are introduced in MC precisely in order to overcome its central obstacle: the state-explosion problem. For example, compositionality is not envisioned as a way to tackle incremental, time-varying and space-varying, highly modular systems – our concern!

(3) MC's greatest successes are in verifying circuit designs and communication protocols. The verification of other kinds of software and hardware systems is not so easily tackled using MC. Take for example the verification of programs, much less their specification/design, which have turned out to be much harder using the tools of MC.

Overview

Central Theme: **Compositionality**

Guiding Questions:

- what is **compositional analysis/specification**?
- what is it **good** for?
- how to **formulate** and **implement** it?
- any **killer app**?

Overview

Central Theme: **Compositionality**

Guiding Questions:

- what is **compositional analysis/specification**?
- what is it **good** for?
- how to **formulate** and **implement** it?
- any **killer app**?

Thesis: **lightweight formal methods can help**

Why "lightweight"? *The logical formalisms that are required to use, analyze, and specify systems are kept to a bare minimum, while more complicated parts of these formalisms remain hidden "under the hood"*

What is CA/S? Preliminary schematic answer …

modules

A	one-module system
A ⊛ B	two-module system
A ⊛ B ⊛ C	three-module system
A ⊛ 〈 〉 ⊛ C	three-module system, one missing ?
A ⊛ D ⊛ C	three-module system
…	…

What is CA/S? Preliminary schematic answer …

modules	whole-system analysis
A	⟦A⟧
A ⊛ B	⟦A ⊛ B⟧
A ⊛ B ⊛ C	⟦A ⊛ B ⊛ C⟧
A ⊛ 〈 〉 ⊛ C	⟦A ⊛ 〈 〉 ⊛ C⟧ ?
A ⊛ D ⊛ C	⟦A ⊛ D ⊛ C⟧
…	…

What is CA/S? Preliminary schematic answer …

modules	whole-system analysis	compositional analysis
A	$[\![A]\!]$	$= [\![A]\!]$
$A \circledast B$	$[\![A \circledast B]\!]$	$= [\![A]\!] \star [\![B]\!]$
$A \circledast B \circledast C$	$[\![A \circledast B \circledast C]\!]$	$= [\![A]\!] \star [\![B]\!] \star [\![C]\!]$
$A \circledast \langle\,\rangle \circledast C$	$[\![A \circledast \langle\,\rangle \circledast C]\!]$?	$= [\![A]\!] \star [\![\,\langle\,\rangle\,]\!] \star [\![C]\!]$
$A \circledast D \circledast C$	$[\![A \circledast D \circledast C]\!]$	$= [\![A]\!] \star [\![D]\!] \star [\![C]\!]$
…	…	…

what about a system with 1000's or more components?

What is CA/S good for? Preliminary answer …

- It's good to know that a network agent (e.g., router, HTTP proxy, …) doesn't crash, when it's disconnected and idle
- It's better to know it won't crash when connected to (composed with) another agent
- It's even better to know it won't crash when composed with a whole bunch of other agents in some arbitrary configuration!

What kind of systems?

Broadly speaking:

- Artifacts – **software** or **hardware** or **combination**

But also:

- **L**arge – 100's, or 1000's, or ... components
- **I**ncremental – distributed in time
- **M**odular – distributed in space
- space/time-varying, separately designed components, prone to fail abruptly, to decay, to delay, to grow, …

Interlude: Scalability

- We build **scalable** systems –
  systems that are easily expanded/upgraded on demand
- Example: **a scalable computer network** –
  one that maintains its performance as it expands
- Different dimensions of scalability:

  - **load scalability** –
    system easily modified to handle heavier loads
  - **geographic scalability** –
    system easily expanded to a more distributed geographic pattern
  - **administrative scalability** –
    system easily supports an increasing number of organizations
  - **etc.**

Analysis or specification?

Broadly speaking:

Analysis is reactive, **post** system design, for

- Detecting malfunctions
- Certifying absence of malfunctions

Specification is preventive, **pre/during** design, for

- Enforcing desirable guarantees
- Avoiding malfunctions from faulty design

An Application ...

- Traffic Network
  - air traffic
  - railroad traffic
  - internet traffic
  - telecommunication traffic
  - vehicular traffic 
  - ant traffic
  - etc.

- not yet a killer app?

Footnote about Ant Traffic

"army ants" in Central America (*Eciton burchelli*) are the pinnacle of traffic organization in the actual world. No one's time is more valuable than anyone else's, no one is preventing anyone else from passing, and no one is making anyone else wait. In spite of very complex network patterns used by half a million or more workers, they never get into traffic jams.

"Self-organized lane formation and optimized traffic flow in army ants"

by I.D. Couzin and N.R.Franks
***Proceedings of the Royal Society of London, Series B270*, pp 139-146, 2003.**

Vehicular Traffic

- 3 basic traffic parameters along a road segment:
  - velocity v
  - density (or occupancy or concentration) k
  - volume (or traffic flow) q
- Fundamental relation of traffic flow:

$$q = v \cdot k$$

v distance traveled per unit time, e.g. km/hour

k number of vehicles per unit distance, e.g. n/km

q number of vehicles per unit time, e.g. n/hour

Vehicular Traffic

Traffic **volume** restricted to range of safe (or optimal) values:

$$q_{\text{max}} = v_{\text{max}} \cdot k_{\text{max}}$$
$$q_{\text{ave}} = v_{\text{ave}} \cdot k_{\text{ave}}$$
$$q_{\text{min}} = v_{\text{min}} \cdot k_{\text{min}}$$

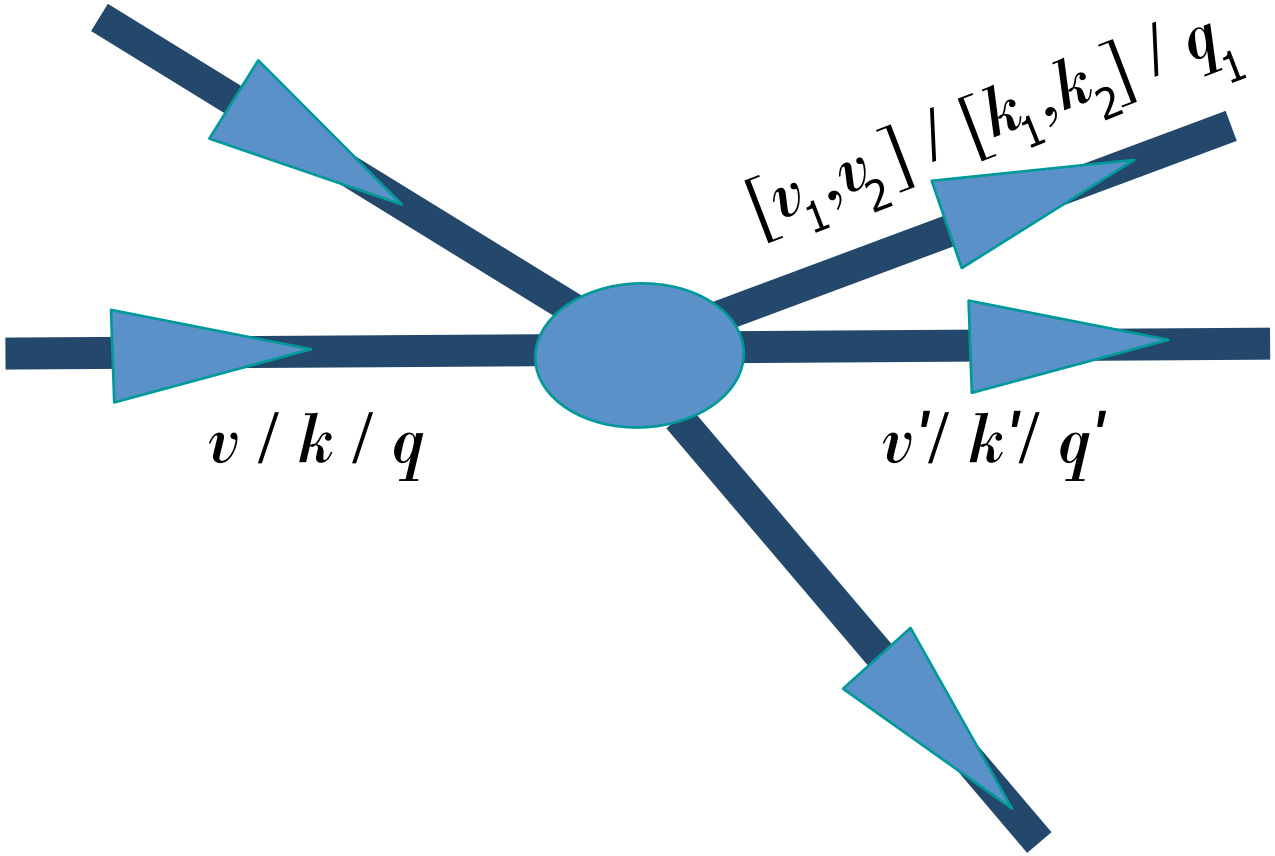
Actual volume of safe/optimal traffic falls in range:

if $q_{\text{min}} < q < q_{\text{max}}$ **then** $q \neq v_{\text{min}} \cdot k_{\text{min}}$

and $q \neq v_{\text{max}} \cdot k_{\text{max}}$

Similar safety/optimality conditions on **velocity** and **density**.

Vehicular Traffic



Every road is specified by a triple $v / k / q$ or $v / [k_1,k_2] / q$ or $[v_1,v_2] / [k_1,k_2] / q$ or ... etc., each entry in triple prescribing a safe **bound** (min or max) or a safe **interval**.

Vehicular Traffic

- Additional Traffic Parameters:
  - size of cluster (of consecutive vehicles) m
  - size of gap (distance between two clusters) n
- Other Relation of Traffic Flow:

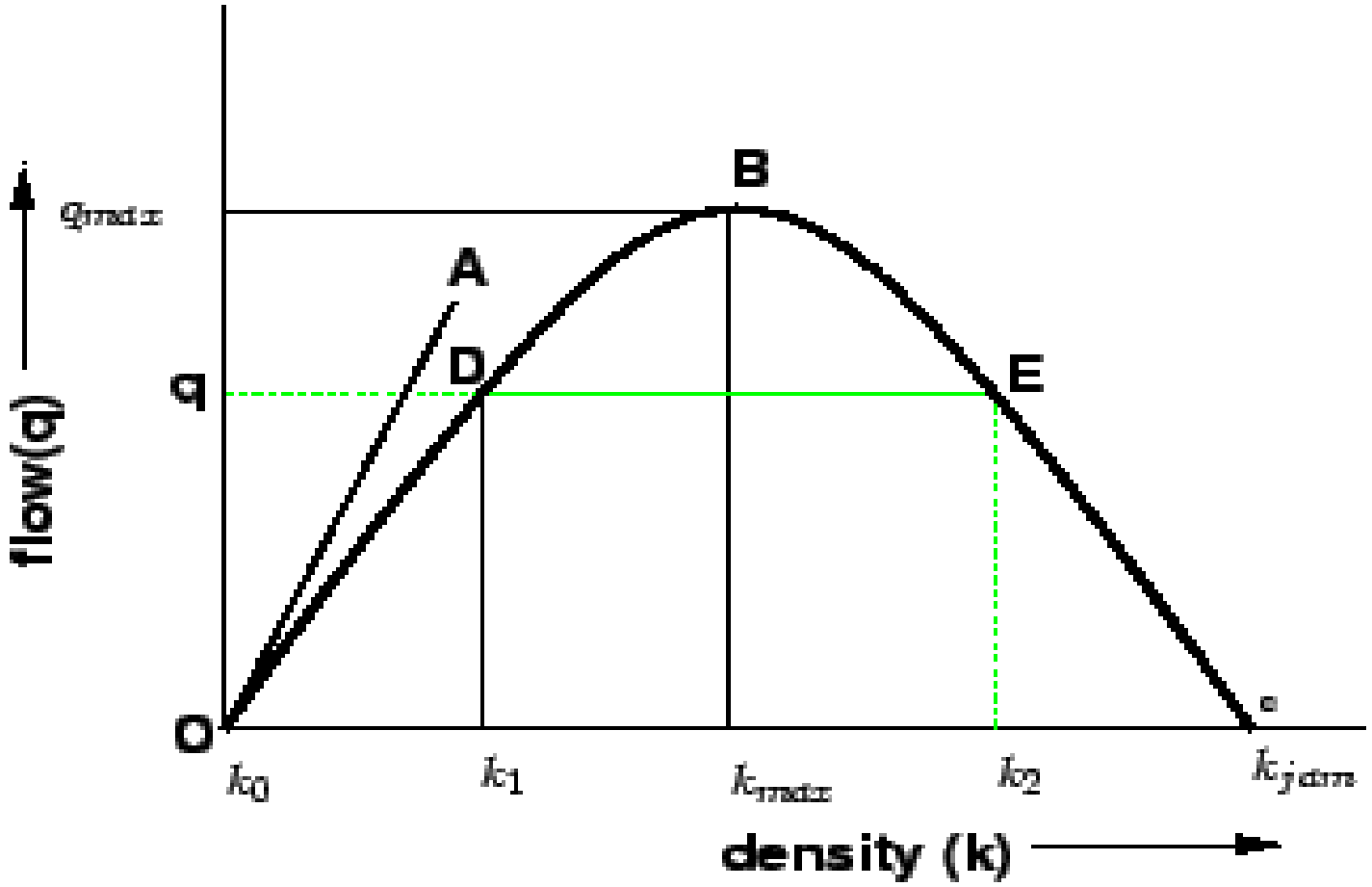
$$k = [(\sum m)/(\sum m)+(\sum n)] \cdot k_{\text{jam}}$$

k_{jam} = maximum number of vehicles that fit "bumper-to-bumper" in one unit distance

In actual practice, k_{jam} is never reached ...

In Actual Practice …

- the flow-density (q,k) curve – parabolic rather than linear in k (when v constant):



Courtesy of Tom Matthew

In Actual Practice …

- the flow-density (q,k) curve
  - the relationship is normally represented by a parabolic curve
  - at jam density, flow will be zero because the vehicles are not moving.
  - there will be some density between zero density and jam density, when the flow is maximum.

For this presentation: Assume the flow-density curve is linear in k rather than parabolic (when v is constant) up to k_{max} .

Notes

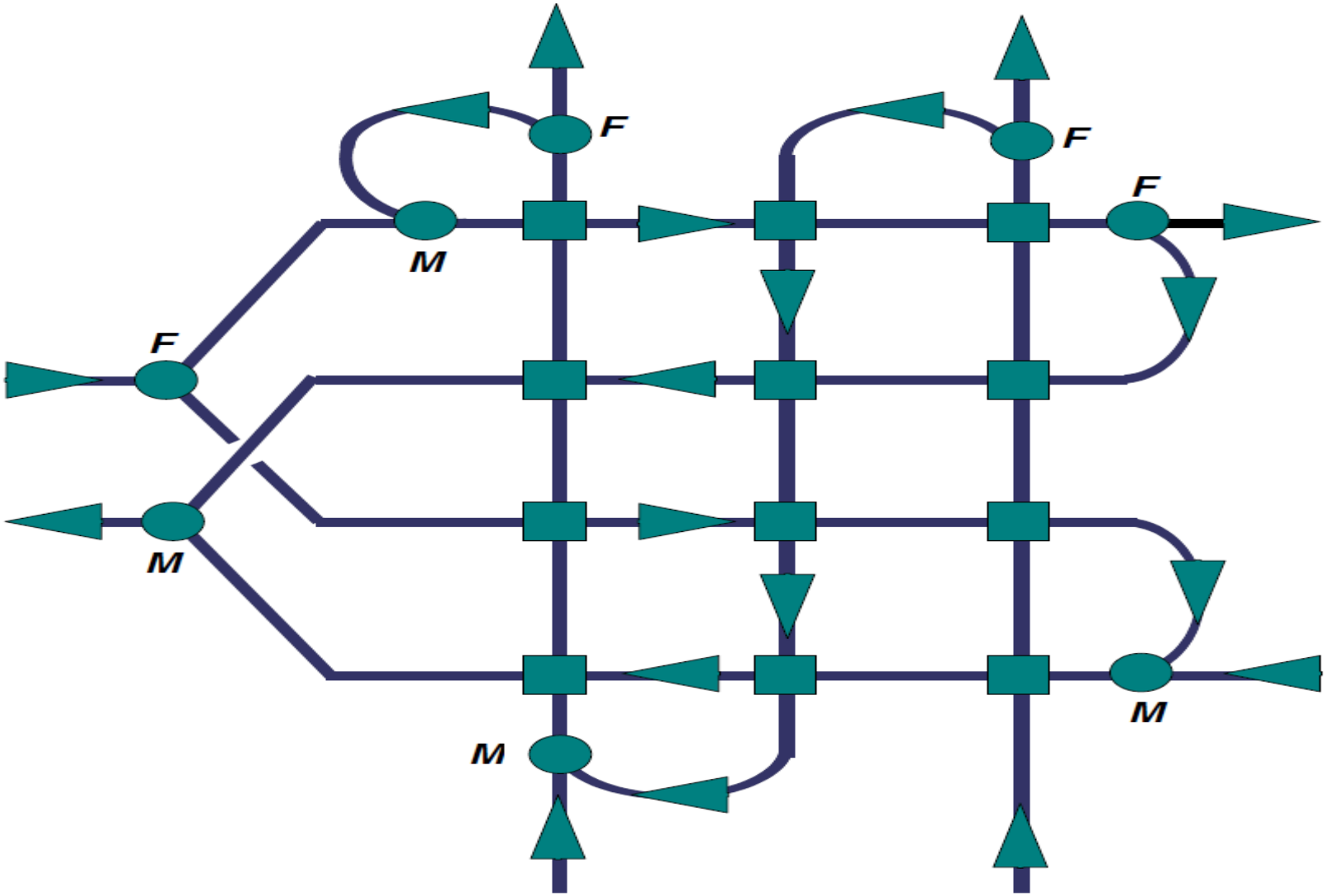
A good deal of work in "traffic engineering", "theory of traffic flow", and related studies, involves traffic modeling and computer simulation. Simulation models are designed to mimic the behavior of traffic networks in the real world, much of it typically based on data collected in observation of actual vehicular traffic. They are used for many goals:

1. Evaluate signal control strategies for an existing geometric pattern of roads.
2. Quantify traffic performance in response to different geometric design, before commitment of resources to construction.
3. Train personnel in charge of traffic management and road maintenance (state police, road repair crews, etc.)
4. Safety analysis. Simulation models are used to "recreate" accident scenarios in the search to build safer vehicles and roadways.
5. etc.

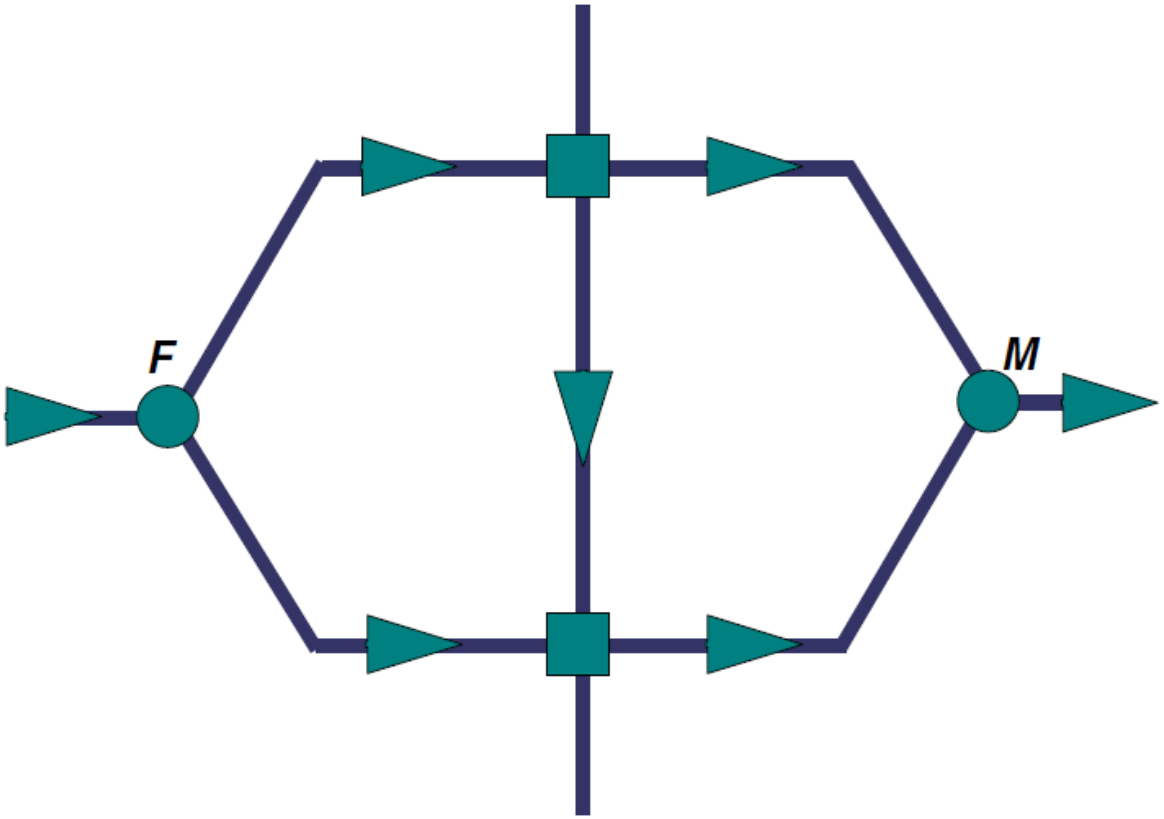
What stands out in all these studies is the extent to which vehicular traffic is considered an objective reality, observed and measured from the outside, much like a fluid flowing into different channels or water into a pattern of streams and rivers. Many of these studies are inspired by fluid dynamics, not by problems in reactive systems or sense-and-respond systems. Some of these studies use fairly involved continuous mathematics, requiring a good deal of mathematical background in differential equations, analysis, ergodic theory, and related areas.

Our standpoint is different. For one thing, the mathematics we use are mostly discrete, with a good deal of formalisms and conventions drawn from mathematical logic ("formal methods" in CS).

A Small Traffic Network ...



A Tiny Traffic Network ...



Different desirable objectives

- **fairness**: no traffic is permanently blocked
- **deadlock-free**: no two competing traffics in which neither will yield for the other
- **flow/mass conservation**: average entering flow is equal to average exiting flow

Something more complicated:

- **gridlock-free**: no grid of competing traffics ultimately resulting in the blocking of all pathways

Something more complicated still:

- **minimize changes in kinetic energy**: because less braking and accelerating means less fuel consumption

$$\textit{kinetic energy} = (1/2) \cdot k \cdot v^2$$

Difficulties of whole-system analysis ...

We illustrate the difficulties with a small example of a traffic network where each node is one of 3 junctions:

- **two-way fork**
- **two-way merge**
- **two-way crossing**

Invariant property to be enforced across network:

- **absence of traffic backups at every junction**

Write constraints for the 3 junction types accordingly, i.e., to ensure there is no traffic backups in the immediate roads entering the junction.

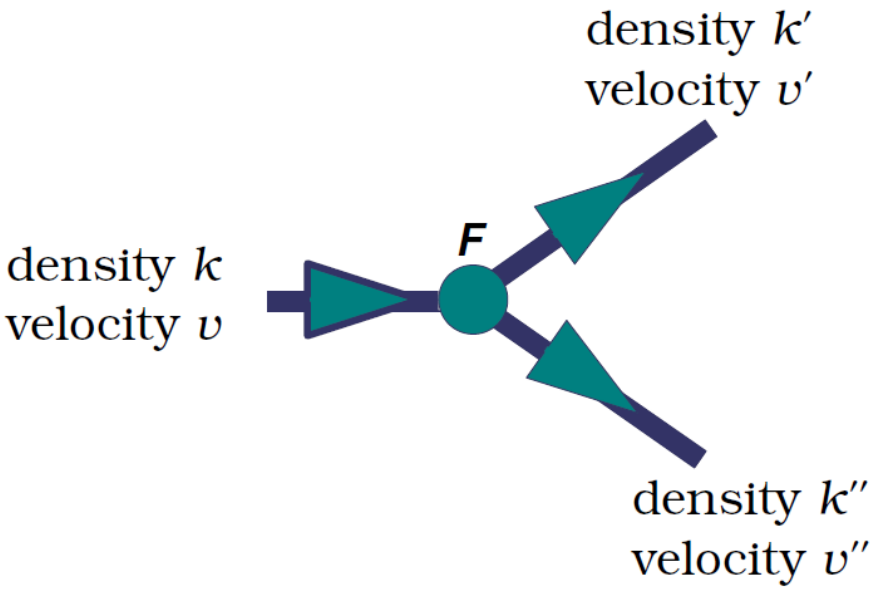
Traffic network nodes

Two-Way Fork

To avoid traffic backups:

1 **linear** constraint

2 **nonlinear** constraints



- $k = k' + k''$
- $v \cdot k \leq v' \cdot k' + v'' \cdot k'$
- **if** $v' \cdot k' + v'' \cdot k'' > 0$ **then** $v \cdot k > 0$

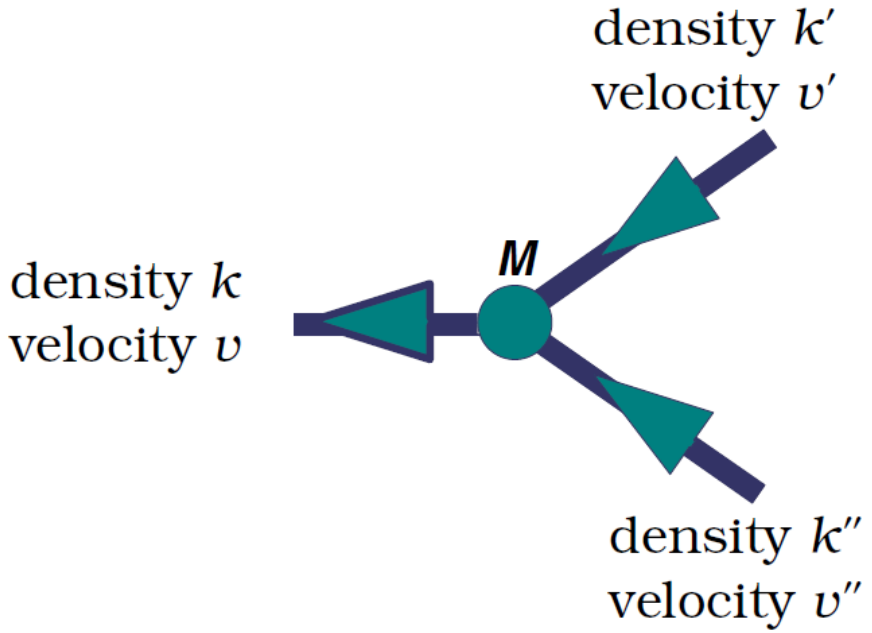
Traffic network nodes

Two-Way Merge

To avoid traffic backups:

1 **linear** constraint

2 **nonlinear** constraints



- $k = k' + k''$
- $v \cdot k \geq v' \cdot k' + v'' \cdot k'$
- **if** $v \cdot k > 0$ **then** $v' \cdot k' + v'' \cdot k'' > 0$

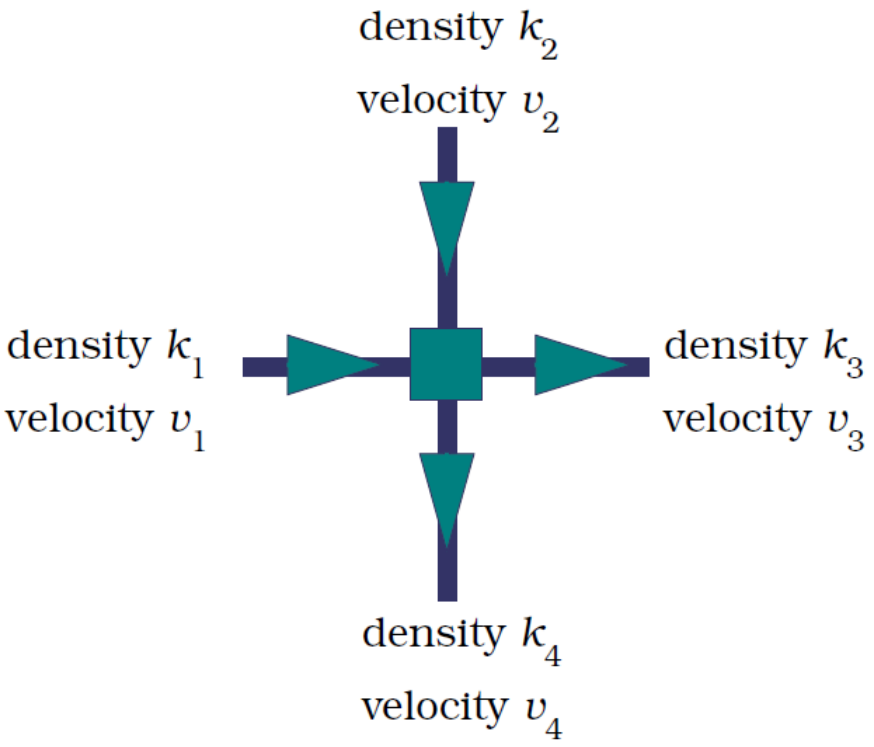
Traffic network nodes

Two-Way Crossing

To avoid traffic backups:

2 **linear** constraints

2 **nonlinear** constraints



- $k_1 + k_2 \leq 100$
- $k_1 + k_2 = k_3 + k_4$
- $v_1 \cdot k_1 + v_2 \cdot k_2 \leq v_3 \cdot k_3 + v_4 \cdot k_4$
- **if** $v_3 \cdot k_3 + v_4 \cdot k_4 > 0$ **then** $v_1 \cdot k_1 + v_2 \cdot k_2 > 0$

(assume $k_{\mathbf{max}} = 100$)

Traffic network nodes

Non-Mixing

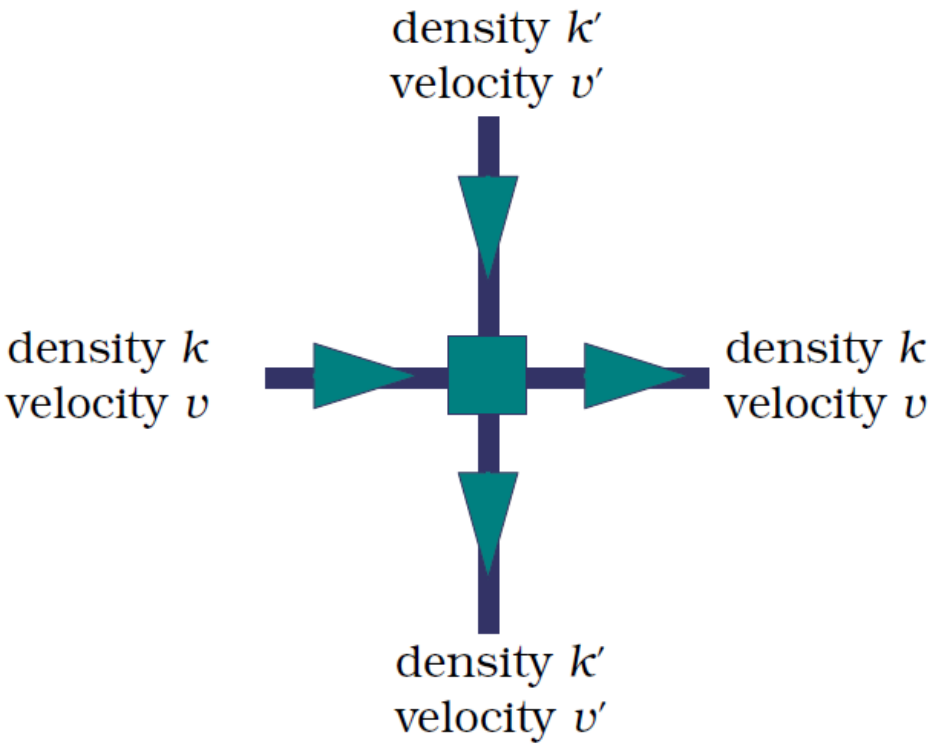
Two-Way Crossing

1 **linear** constraint

to avoid backups

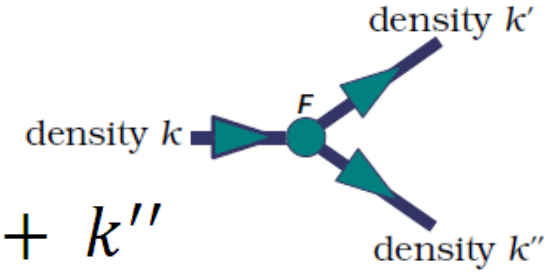
- $k_1 + k_2 \leq 100$

(assume $k_{\mathbf{max}} = 100$)

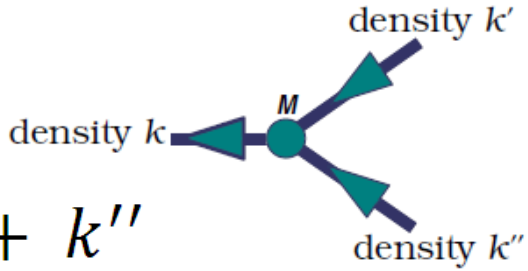


Traffic network nodes

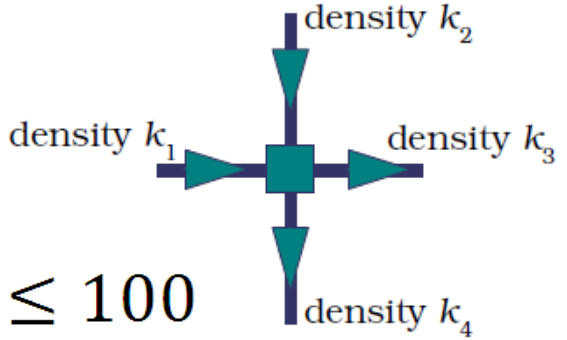
- Assume
  Uniform Velocity
  throughout network
- To avoid backups:
  only **linear** constraints
  - 1 for two-way fork
  - 1 for two-way merge
  - 2 for two-way crossing



- $k = k' + k''$



- $k = k' + k''$



- $k_1 + k_2 \leq 100$
- $k_1 + k_2 = k_3 + k_4$

Example: Tiny Traffic Network …

Constraints

safe local intervals/types:

$$20 \le v_1, v_2, v_3 \le 80$$

$$20 \le k_1, k_2, k_3 \le 90$$

$$30 \le v_4, v_5 \le 60$$

$$40 \le k_4, k_5 \le 70$$

no backups:

$$k_1 = k_2 + k_3$$

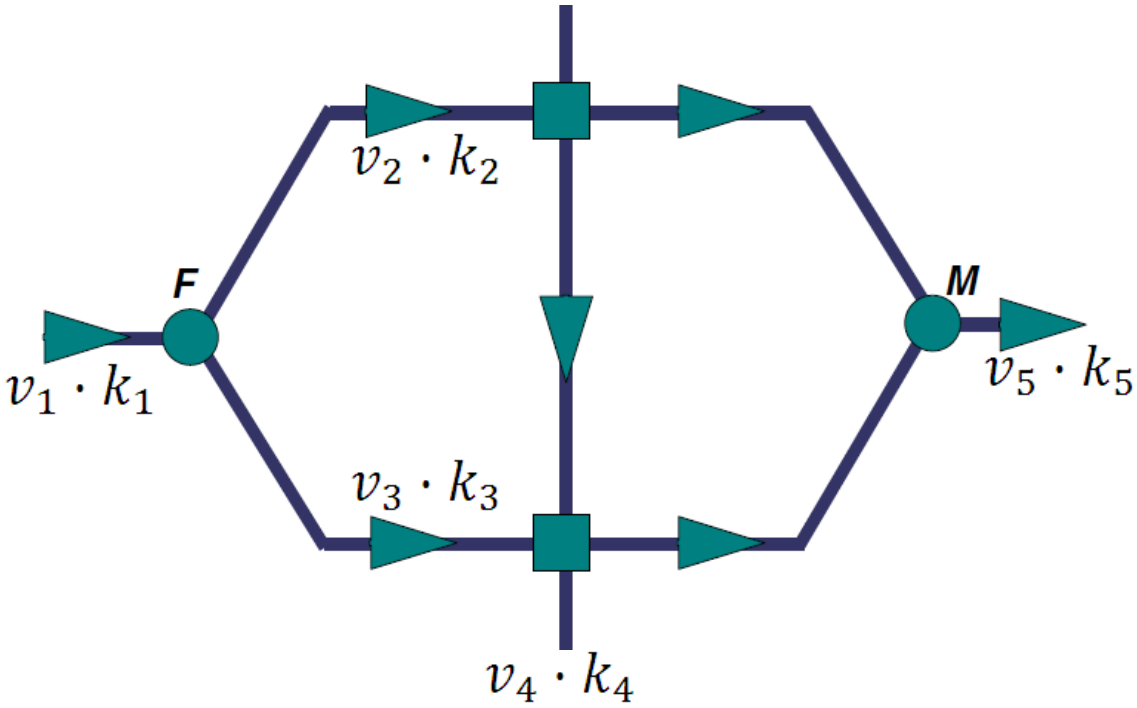
$$k_2 + k_3 = k_5$$

$$k_2 + k_4 \le 100$$

$$k_3 + k_4 \le 100$$

$$v_1 \cdot k_1 \le v_2 \cdot k_2 + v_3 \cdot k_3$$

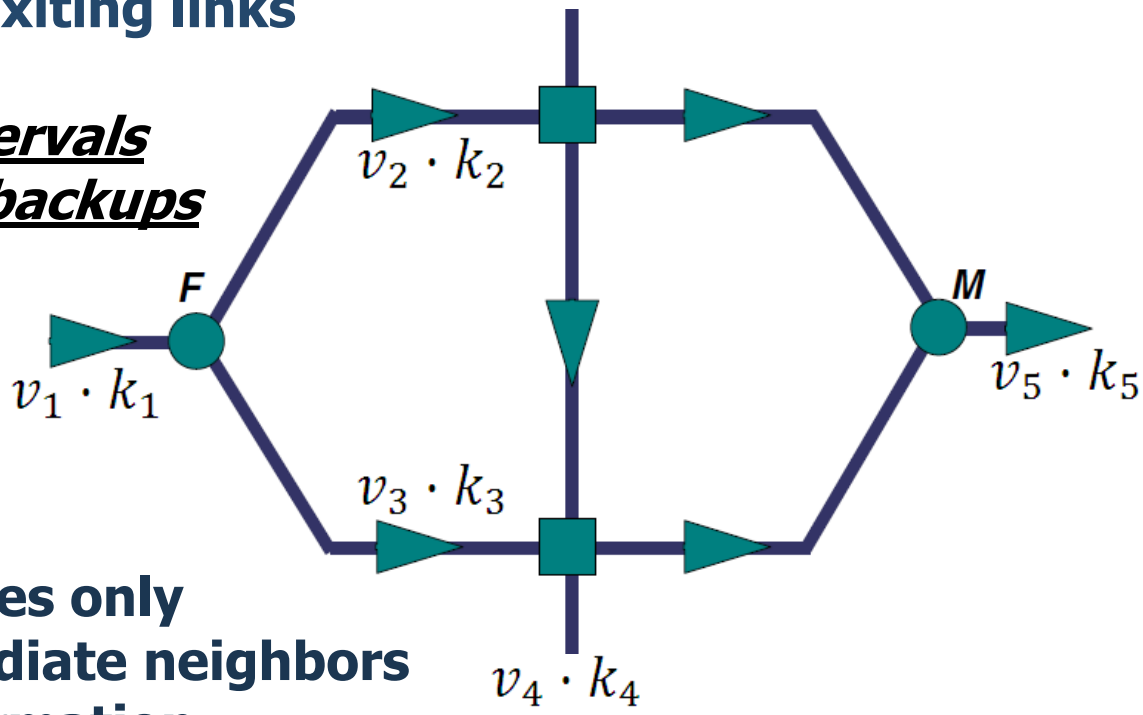
$$v_2 \cdot k_2 + v_3 \cdot k_3 \le v_5 \cdot k_5$$



Example: Tiny Traffic Network …

Goal and Guidelines. Each traffic junction computes least restrictive (*velocity interval*)•(*density interval*) on each of its entering/exiting links without violating:

- the given ***safe local intervals***
- the constraints for ***no-backups***



Possible Limitations.

- Each traffic junction uses only information from immediate neighbors
- Incomplete global information
- Only partial or no global traffic administrator

More Objective Functions. To be added later, different for different small groups of contiguous traffic junctions (or "modules")

(1) At an entering road (input) the "best" type is the least restrictive to incoming traffic flow, i.e., the most desirable velocity intervals and density intervals at entering roads are the widest possible ones.

(2) At an exiting road (output) the "best" type is the most permissive to outgoing flow, i.e., the most desirable velocity intervals and density intervals at entering roads are again the widest possible ones.

So, the strongest possible specification for a traffic network tries to maximize velocity intervals and density intervals at both entering and exiting roads. This is a counterpoint to what we usually set up as the strongest specification for a computer program: the weakest or least restrictive type at the input, and the strongest or most precise type at the output.

We will infer types starting form an exiting link or an entering link. And just as with a program, our game plan is the following:

Starting at an exit link and a type for this link, find the "best" preconditions , i.e. , the least restrictive types at entry links.

Starting at an entry link and a type for this link, find the "best" postconditions, i.e., also the least restrictive types at exit links.

If we had global information about the network, i.e., simultaneous access to (1) all the velocity variables, (2) all the density variables, (3) all the given local safe intervals, and (4) all the constraints for no-backups everywhere, then things would be simpler – just conceptually, because we would still have to cope with complexity issues that would make computing solutions infeasible in practice.

Suppose we give priority to link 5, i.e., we want the least stringent (velocity,density) requirements at link 5 and then compute the least stringent (v,d) requirements for the other links accordingly. The corresponding (velocity,density) variables at link 5 are v_5 and k_5, and we want to maximize the velocity interval and density interval. We therefore have 4 objective functions:

Minimize (k_5^min – 40) with 40 <= k_5^min <= 70
Minimize (70 – k_5^max) with 40 <= k_5^max <= 70
Minimize (v_5^min – 30) with 30 <= v_5^min <= 60
Minimize (60 – v_5^max) with 30 <= v_5^max <= 60

where k_5^min, k_5^max, v_5^min, v_5^max are new variables corresponding to min and max of the density and velocity on link 5, which we can solve for according to established techniques of linear programming, quadratic programming, quadratically-constrained quadratic programming, and other similar optimization methods. But all of these methods quickly run into complexity obstacles.

Tiny Traffic Network …

Inferring Safe Intervals/Types

safe local intervals/types:

$$20 \le v_1, v_2, v_3 \le 80$$

$$20 \le k_1, k_2, k_3 \le 90$$

$$30 \le v_4, v_5 \le 60$$

$$40 \le k_4, k_5 \le 70$$

no backups:

$$k_1 = k_2 + k_3$$

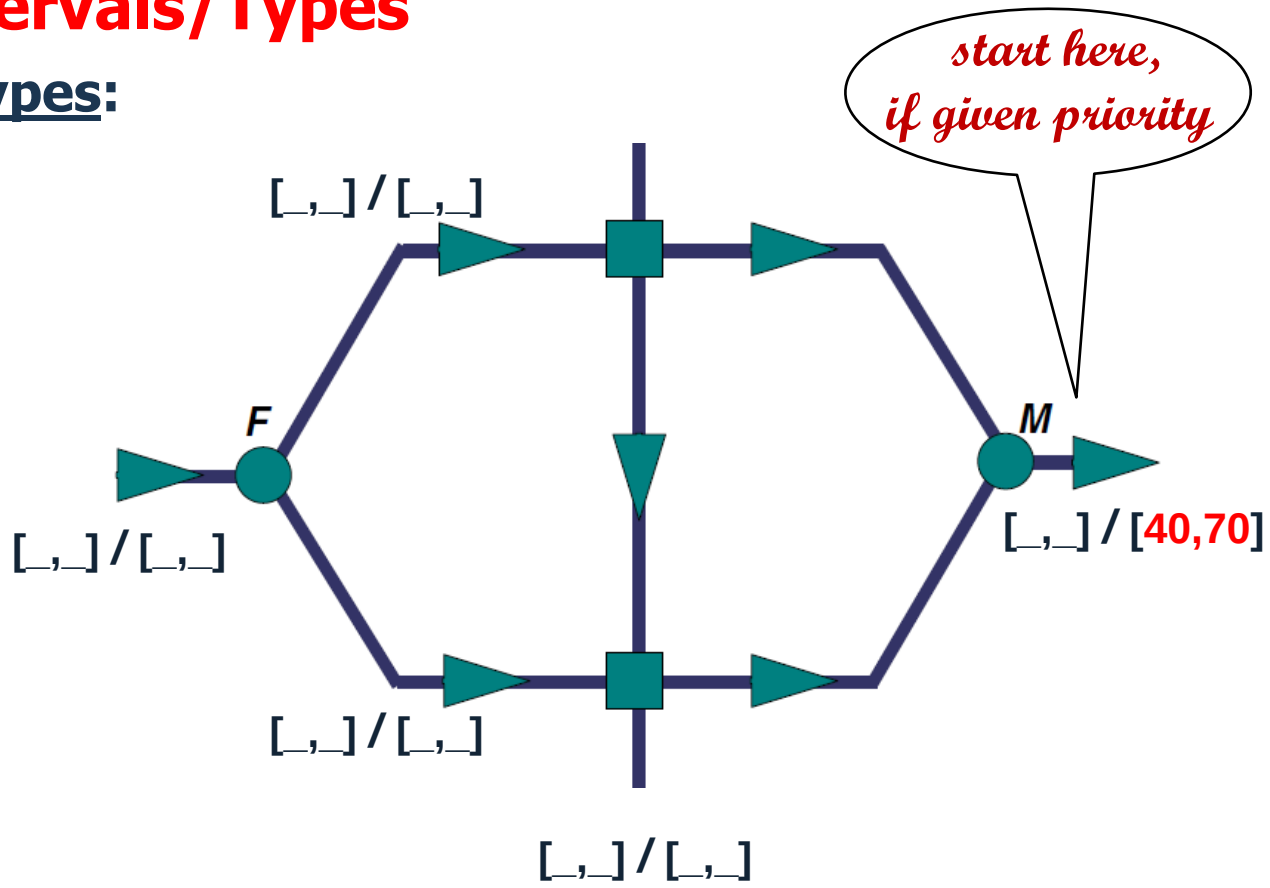
$$k_2 + k_3 = k_5$$

$$k_2 + k_4 \le 100$$

$$k_3 + k_4 \le 100$$

$$v_1 \cdot k_1 \le v_2 \cdot k_2 + v_3 \cdot k_3$$

$$v_2 \cdot k_2 + v_3 \cdot k_3 \le v_5 \cdot k_5$$



Tiny Traffic Network …

Inferring Safe Intervals/Types

safe local intervals/types:

$$20 \le v_1, v_2, v_3 \le 80$$

$$20 \le k_1, k_2, k_3 \le 90$$

$$30 \le v_4, v_5 \le 60$$

$$40 \le k_4, k_5 \le 70$$

no backups:

$$k_1 = k_2 + k_3$$

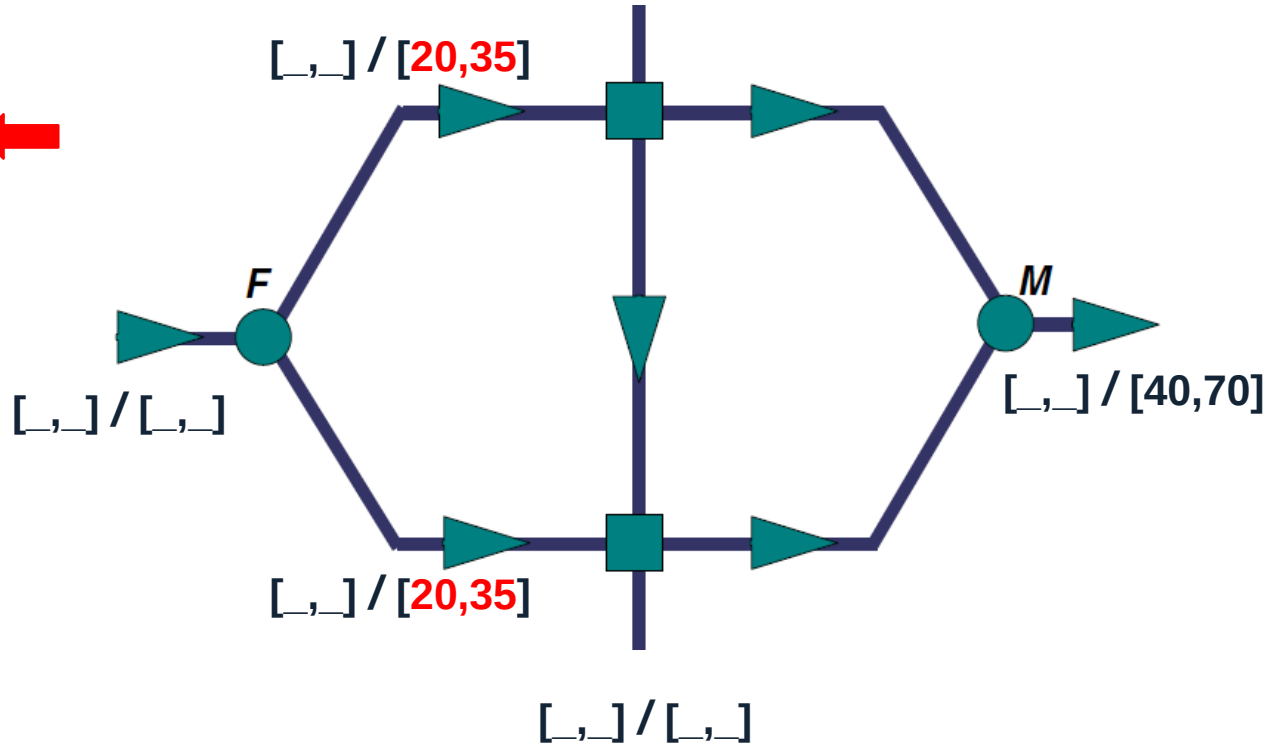
$$k_2 + k_3 = k_5$$

$$k_2 + k_4 \le 100$$

$$k_3 + k_4 \le 100$$

$$v_1 \cdot k_1 \le v_2 \cdot k_2 + v_3 \cdot k_3$$

$$v_2 \cdot k_2 + v_3 \cdot k_3 \le v_5 \cdot k_5$$



Tiny Traffic Network ...

Inferring Safe Intervals/Types

safe local intervals/types:

$$20 \le v_1, v_2, v_3 \le 80$$

$$20 \le k_1, k_2, k_3 \le 90$$

$$30 \le v_4, v_5 \le 60$$

$$40 \le k_4, k_5 \le 70$$

no backups:

$$k_1 = k_2 + k_3$$

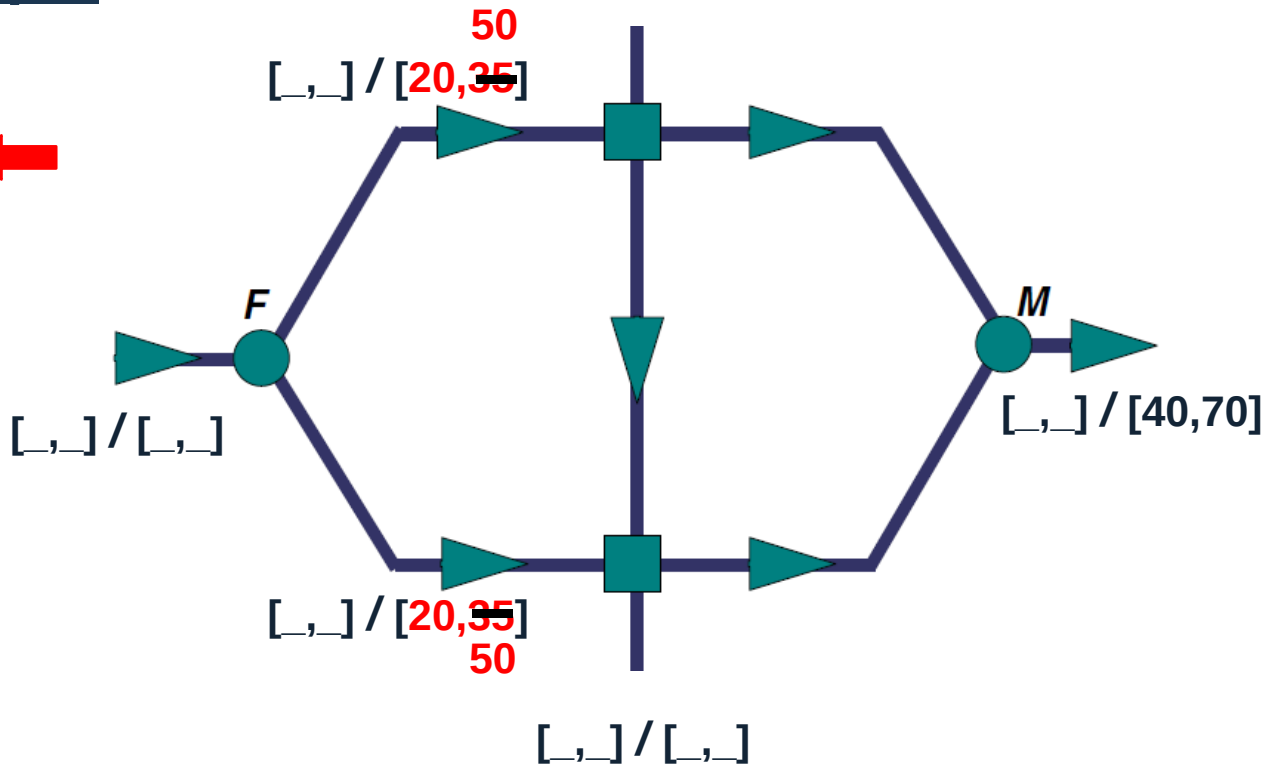
$$k_2 + k_3 = k_5$$

$$k_2 + k_4 \le 100$$

$$k_3 + k_4 \le 100$$

$$v_1 \cdot k_1 \le v_2 \cdot k_2 + v_3 \cdot k_3$$

$$v_2 \cdot k_2 + v_3 \cdot k_3 \le v_5 \cdot k_5$$



recorded constraints:

$$k_2 + k_3 \le 70$$

Tiny Traffic Network …

Inferring Safe Intervals/Types

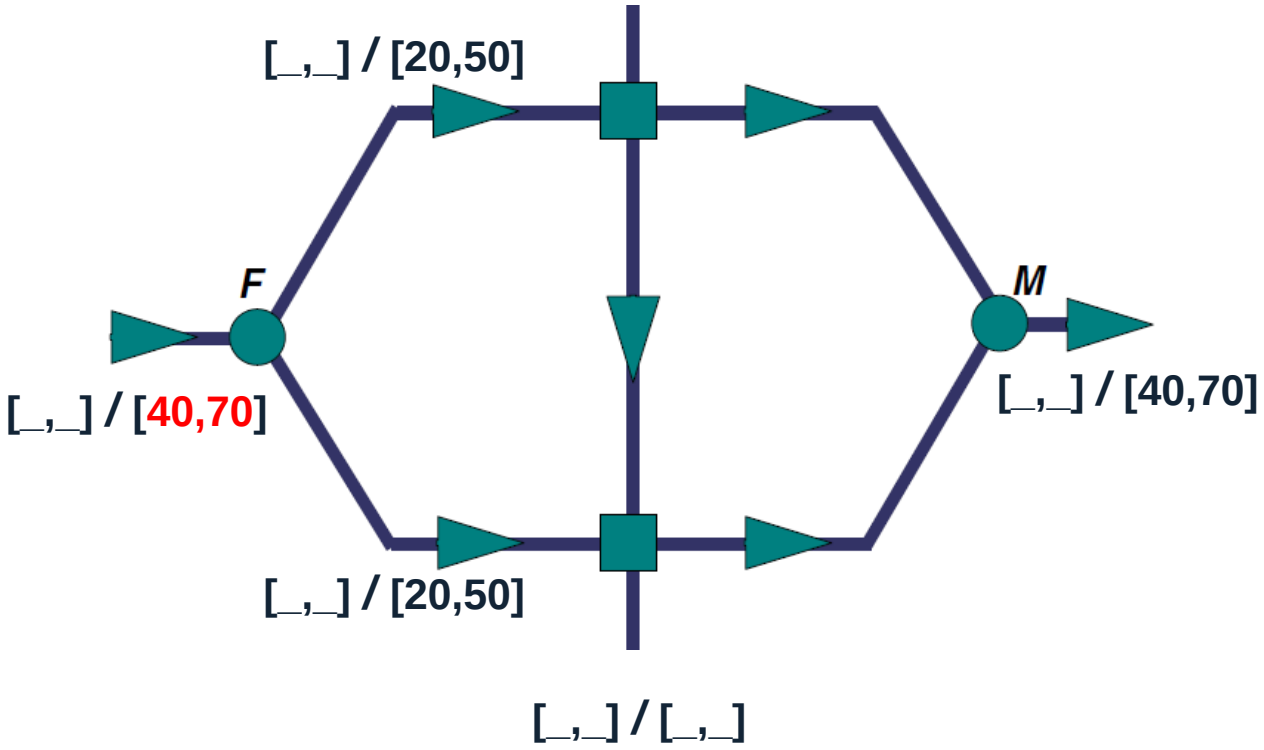
safe local intervals/types:

$$20 \le v_1, v_2, v_3 \le 80$$

$$20 \le k_1, k_2, k_3 \le 90$$

$$30 \le v_4, v_5 \le 60$$

$$40 \le k_4, k_5 \le 70$$



no backups:

$$k_1 = k_2 + k_3$$

$$k_2 + k_3 = k_5$$

$$k_2 + k_4 \le 100$$

$$k_3 + k_4 \le 100$$

$$v_1 \cdot k_1 \le v_2 \cdot k_2 + v_3 \cdot k_3$$

$$v_2 \cdot k_2 + v_3 \cdot k_3 \le v_5 \cdot k_5$$

recorded constraints:

$$k_2 + k_3 \le 70$$

Tiny Traffic Network …

Inferring Safe Intervals/Types

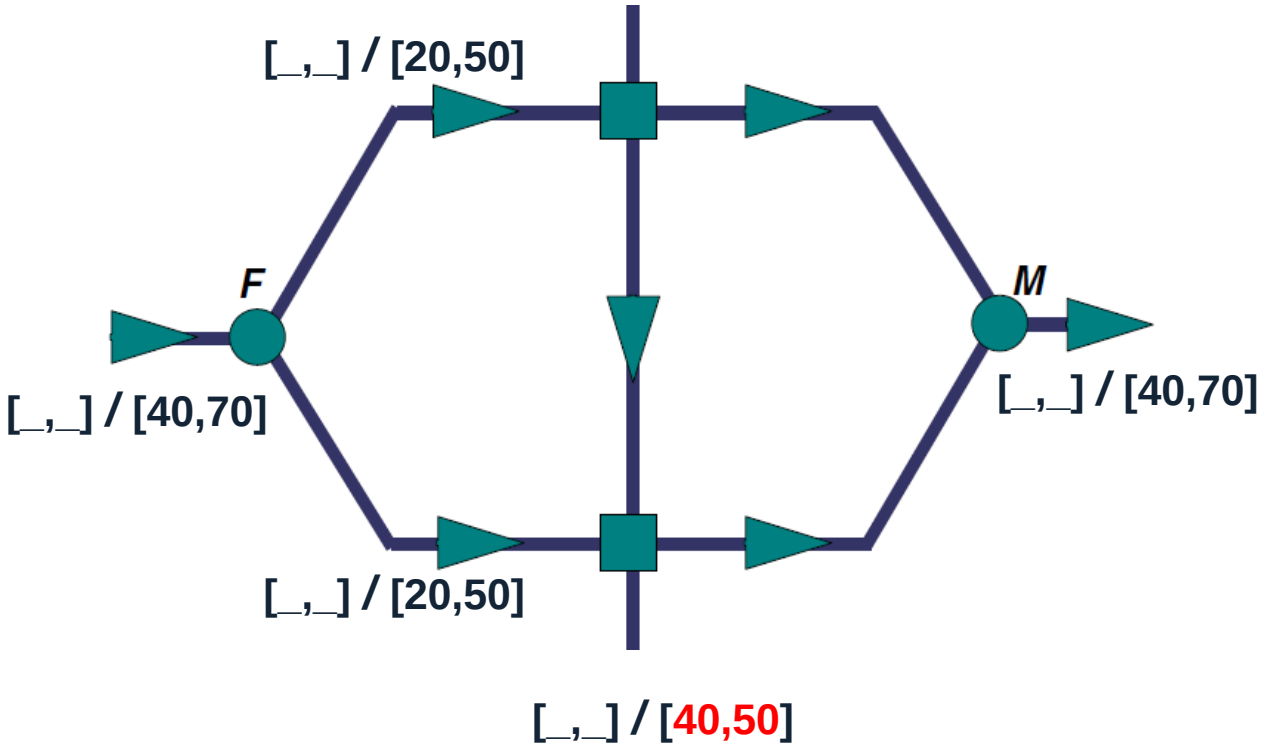
safe local intervals/types:

$$20 \le v_1, v_2, v_3 \le 80$$

$$20 \le k_1, k_2, k_3 \le 90$$

$$30 \le v_4, v_5 \le 60$$

$$40 \le k_4, k_5 \le 70$$



no backups:

$$k_1 = k_2 + k_3$$

$$k_2 + k_3 = k_5$$

$$k_2 + k_4 \le 100$$

$$k_3 + k_4 \le 100$$

$$v_1 \cdot k_1 \le v_2 \cdot k_2 + v_3 \cdot k_3$$

$$v_2 \cdot k_2 + v_3 \cdot k_3 \le v_5 \cdot k_5$$

recorded constraints:

$$k_2 + k_3 \le 70$$

Tiny Traffic Network …

Inferring Safe Intervals/Types

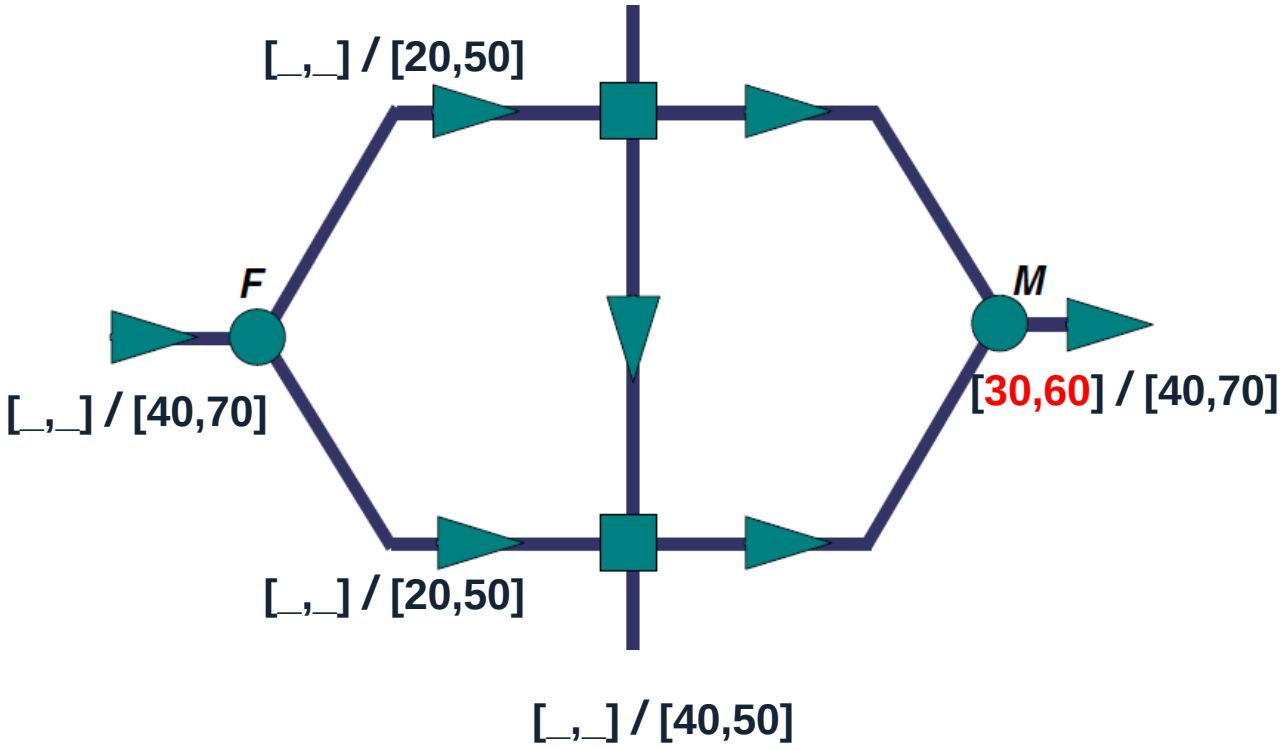
safe local intervals/types:

$$20 \le v_1, v_2, v_3 \le 80$$

$$20 \le k_1, k_2, k_3 \le 90$$

$$30 \le v_4, v_5 \le 60$$

$$40 \le k_4, k_5 \le 70$$



no backups:

$$k_1 = k_2 + k_3$$

$$k_2 + k_3 = k_5$$

$$k_2 + k_4 \le 100$$

$$k_3 + k_4 \le 100$$

$$v_1 \cdot k_1 \le v_2 \cdot k_2 + v_3 \cdot k_3$$

$$v_2 \cdot k_2 + v_3 \cdot k_3 \le v_5 \cdot k_5$$

recorded constraints:

$$k_2 + k_3 \le 70$$

Tiny Traffic Network ...

Inferring Safe Intervals/Types

safe local intervals/types:

$$20 \le v_1, v_2, v_3 \le 80$$

$$20 \le k_1, k_2, k_3 \le 90$$

$$30 \le v_4, v_5 \le 60$$

$$40 \le k_4, k_5 \le 70$$

no backups:

$$k_1 = k_2 + k_3$$

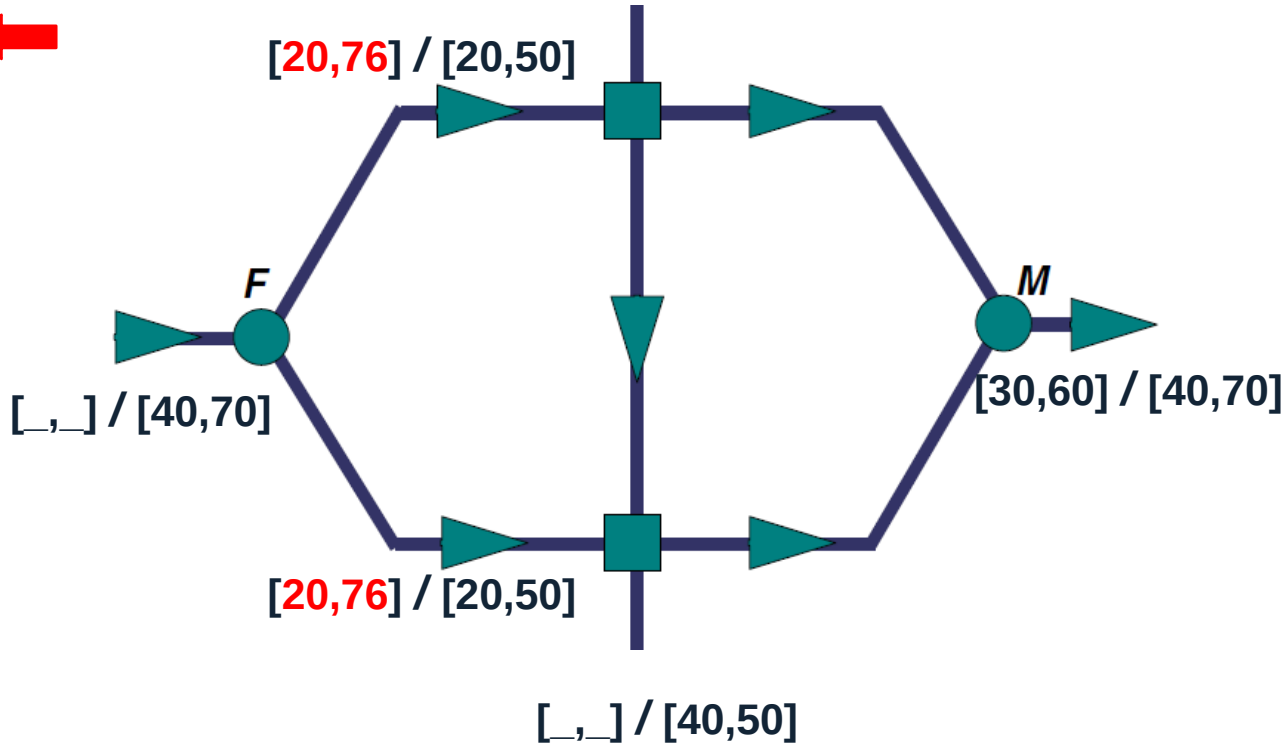
$$k_2 + k_3 = k_5$$

$$k_2 + k_4 \le 100$$

$$k_3 + k_4 \le 100$$

$$v_1 \cdot k_1 \le v_2 \cdot k_2 + v_3 \cdot k_3$$

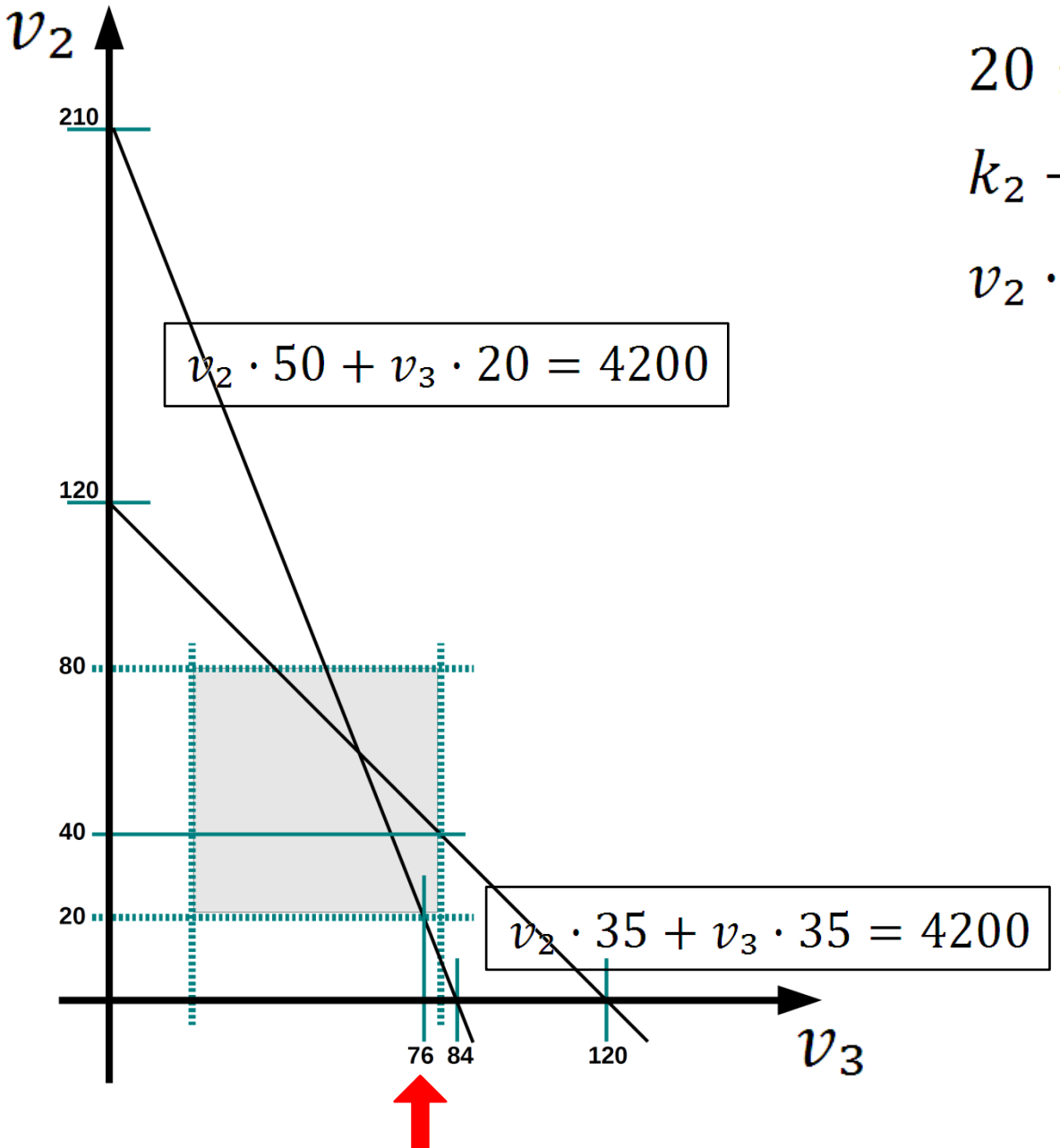
$$v_2 \cdot k_2 + v_3 \cdot k_3 \le v_5 \cdot k_5$$



recorded constraints:

$$k_2 + k_3 \le 70$$

Manipulating constraints …



$$20 \leq k_2, k_3 \leq 50$$

$$k_2 + k_3 \leq 70$$

$$v_2 \cdot k_2 + v_3 \cdot k_3 \leq 4200$$

Tiny Traffic Network …

Inferring Safe Intervals/Types

safe local intervals/types:

$$20 \le v_1, v_2, v_3 \le 80$$

$$20 \le k_1, k_2, k_3 \le 90$$

$$30 \le v_4, v_5 \le 60$$

$$40 \le k_4, k_5 \le 70$$

no backups:

$$k_1 = k_2 + k_3$$

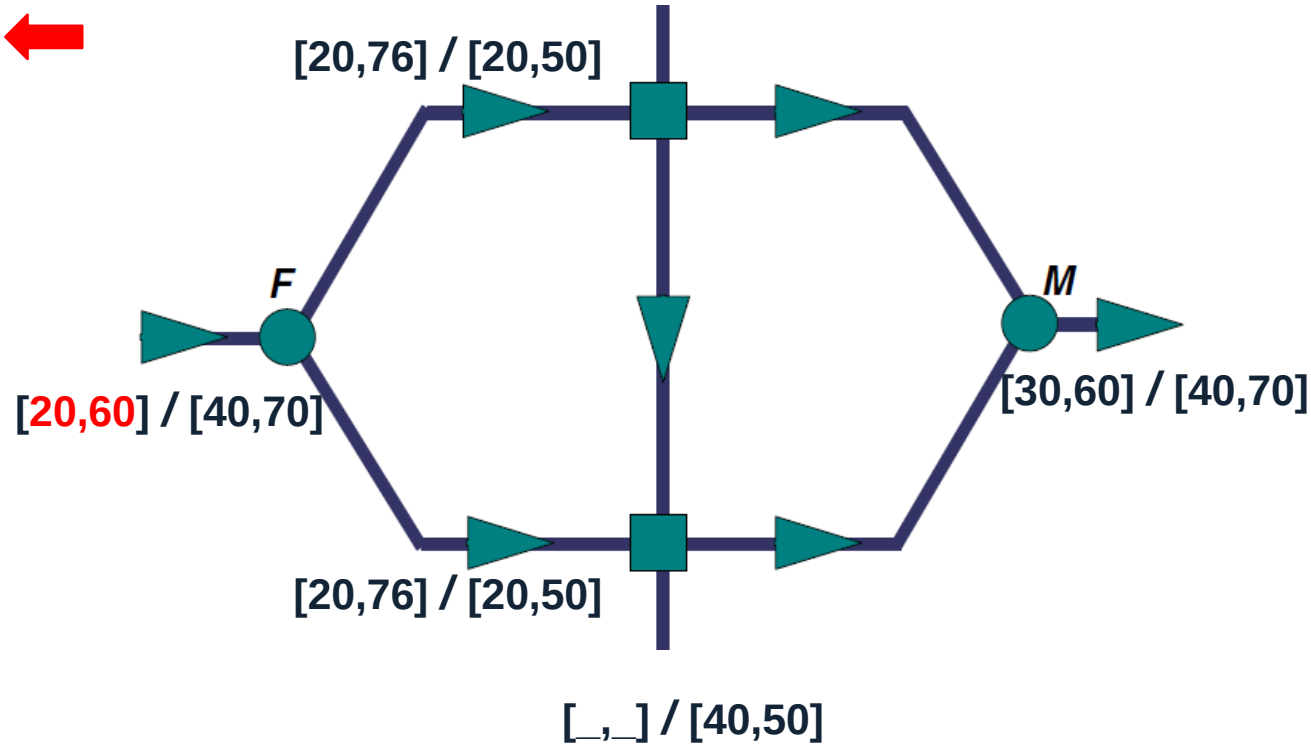
$$k_2 + k_3 = k_5$$

$$k_2 + k_4 \le 100$$

$$k_3 + k_4 \le 100$$

$$v_1 \cdot k_1 \le v_2 \cdot k_2 + v_3 \cdot k_3$$

$$v_2 \cdot k_2 + v_3 \cdot k_3 \le v_5 \cdot k_5$$



recorded constraints:

$$k_2 + k_3 \le 70$$

Tiny Traffic Network …

Inferring Safe Intervals/Types

safe local intervals/types:

$$20 \le v_1, v_2, v_3 \le 80$$

$$20 \le k_1, k_2, k_3 \le 90$$

$$30 \le v_4, v_5 \le 60$$

$$40 \le k_4, k_5 \le 70$$

no backups:

$$k_1 = k_2 + k_3$$

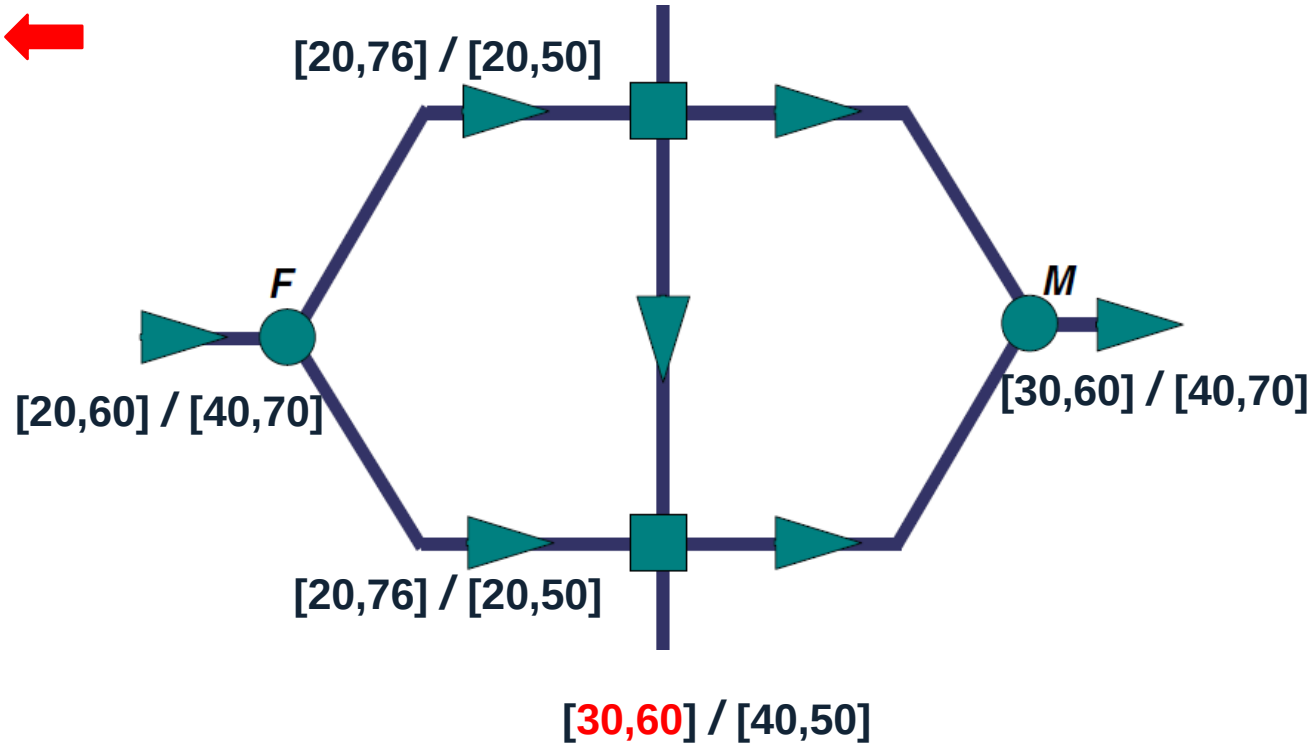
$$k_2 + k_3 = k_5$$

$$k_2 + k_4 \le 100$$

$$k_3 + k_4 \le 100$$

$$v_1 \cdot k_1 \le v_2 \cdot k_2 + v_3 \cdot k_3$$

$$v_2 \cdot k_2 + v_3 \cdot k_3 \le v_5 \cdot k_5$$



recorded constraints:

$$k_2 + k_3 \le 70$$

Tiny Traffic Network, specified for no-backups

❑ **All inferred intervals are subintervals/subtypes of initially given local intervals.**

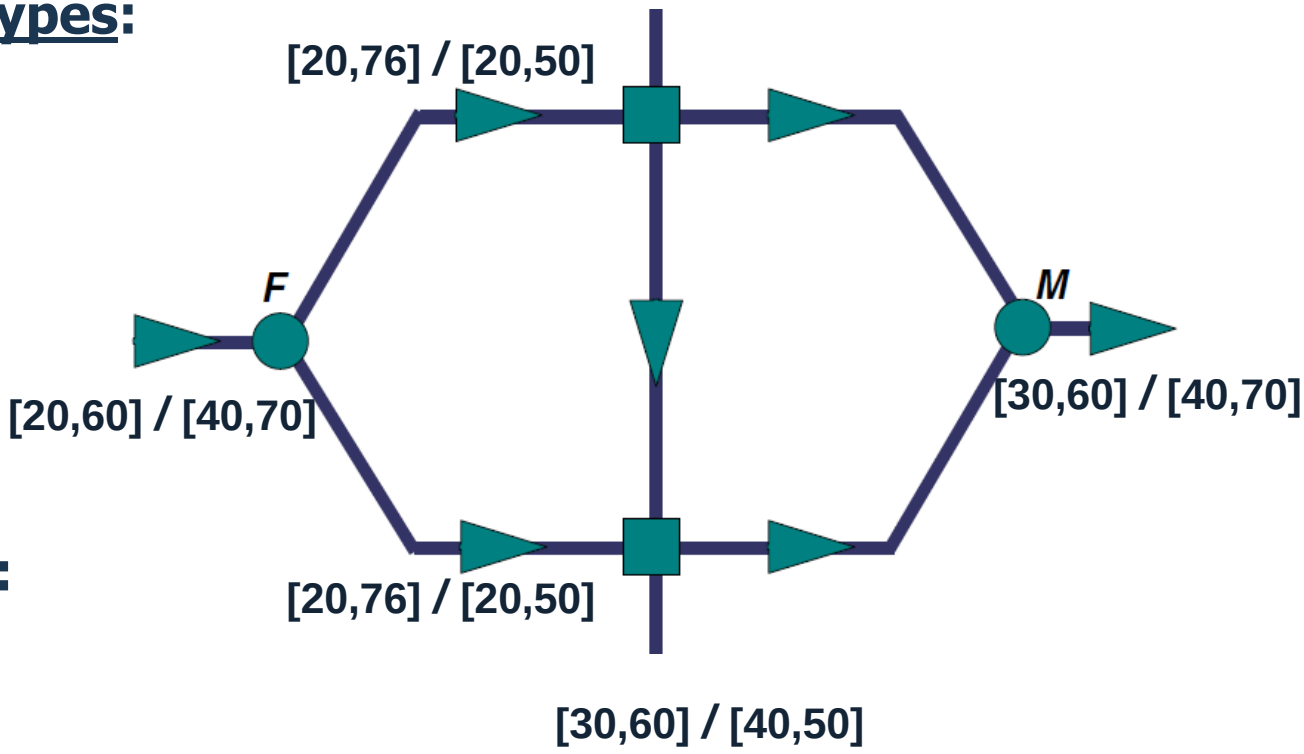
safe local intervals/types:

$$20 \leq v_1, v_2, v_3 \leq 80$$

$$20 \leq k_1, k_2, k_3 \leq 90$$

$$30 \leq v_4, v_5 \leq 60$$

$$40 \leq k_4, k_5 \leq 70$$



recorded constraints:

$$k_2 + k_3 \leq 70$$

❑ **Restricting in-flows $v_1 \cdot k_1$ and $v_4 \cdot k_4$ to inferred types [20,60] / [40,70] and [30,60] / [40,50], respectively, will guarantee no traffic backups.**

Tiny Traffic Network, another specification for no-backups

- **All inferred intervals are subintervals/subtypes of initially given local intervals.**

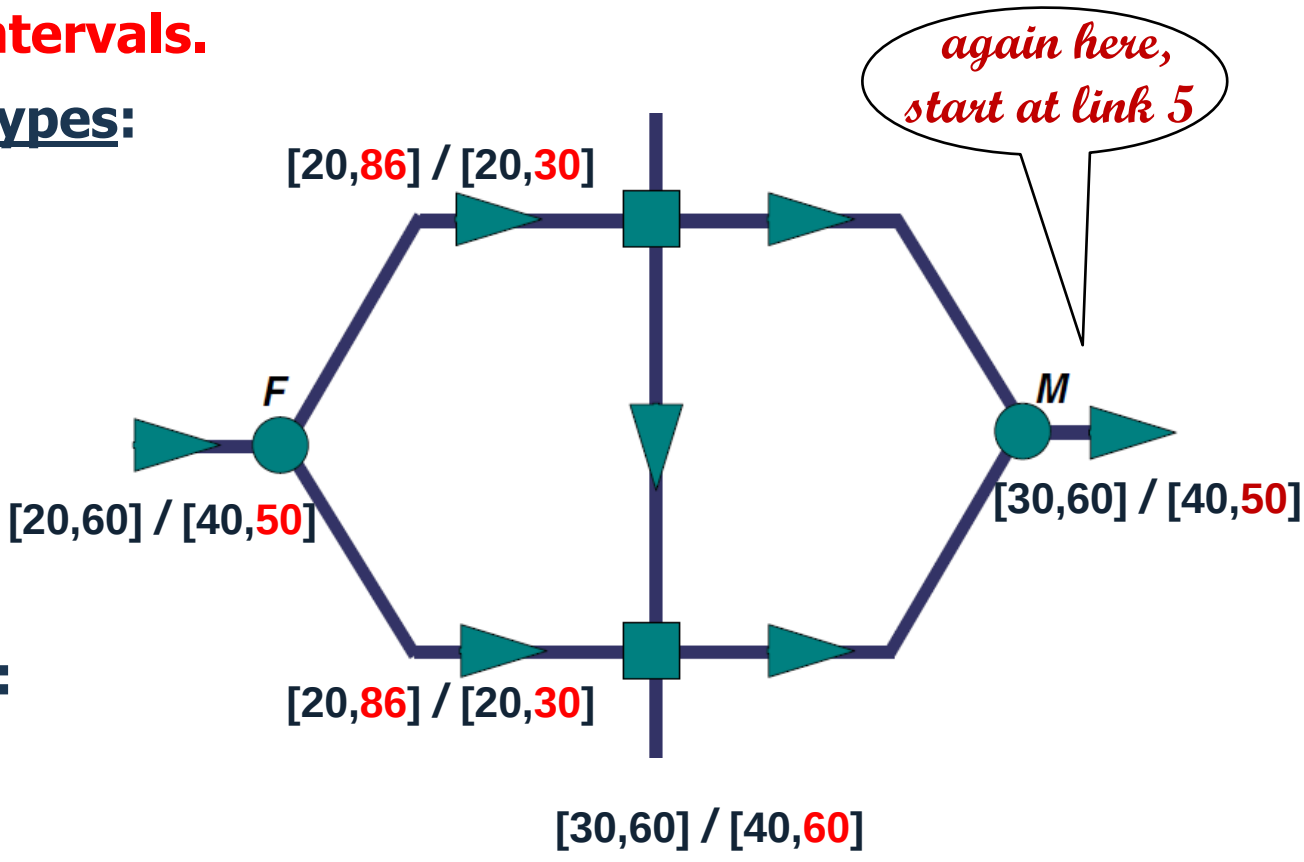
safe local intervals/types:

$$20 \le v_1, v_2, v_3 \le 80$$

$$20 \le k_1, k_2, k_3 \le 90$$

$$30 \le v_4, v_5 \le 60$$

$$40 \le k_4, k_5 \le 70$$



recorded constraints:

$$k_2 + k_3 \le 50$$

- **Restricting in-flows $v_1 \cdot k_1$ and $v_4 \cdot k_4$ to inferred types [20,60] / [40,50] and [30,60] / [40,60], respectively, will guarantee no traffic backups.**

Tiny Traffic Network, as a module based on 1st specification

$\sigma_1 = [20,60]/[40,70]$

$\sigma_2 = [30,60]/[40,50]$

$\tau_1 = [30,60]/[40,50]$

$\tau_2 = [30,60]/[40,70]$

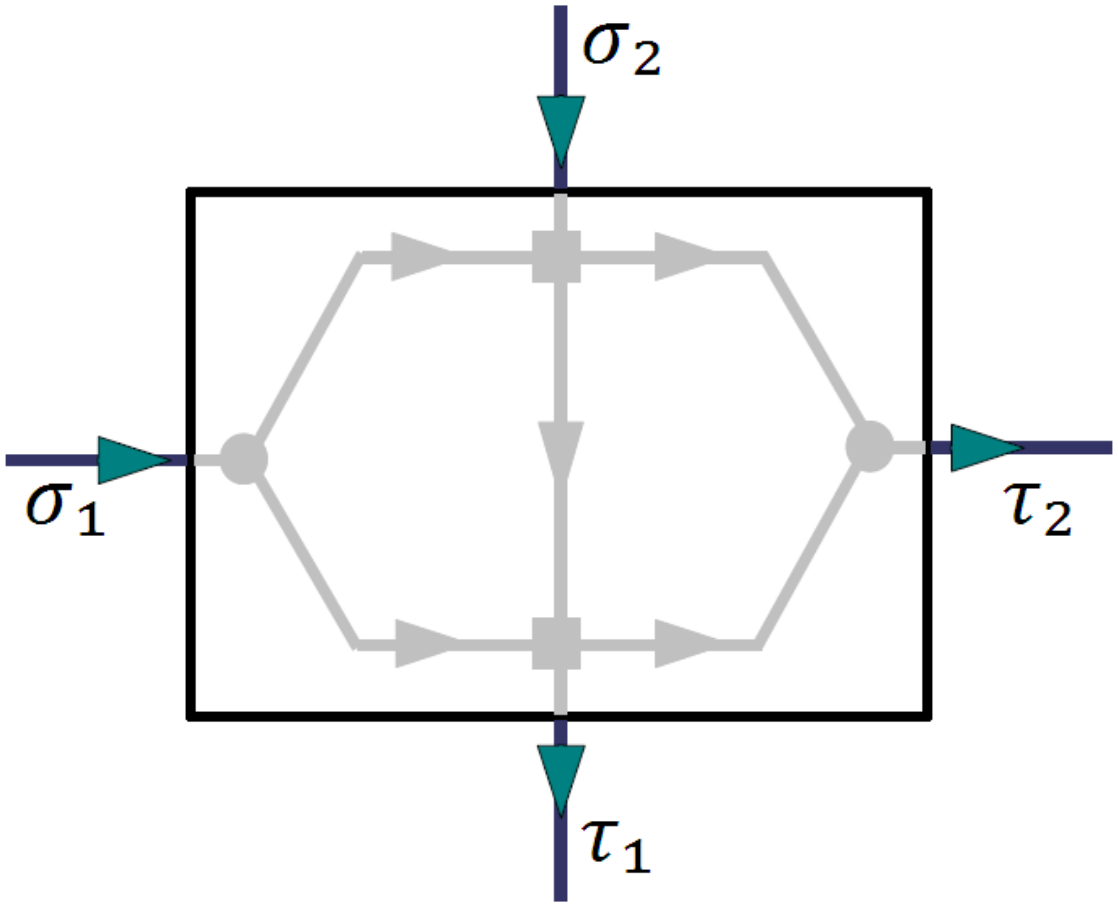
subtyping relationships:

$\tau_1 <: \sigma_1$

$\tau_1 <: \sigma_2$

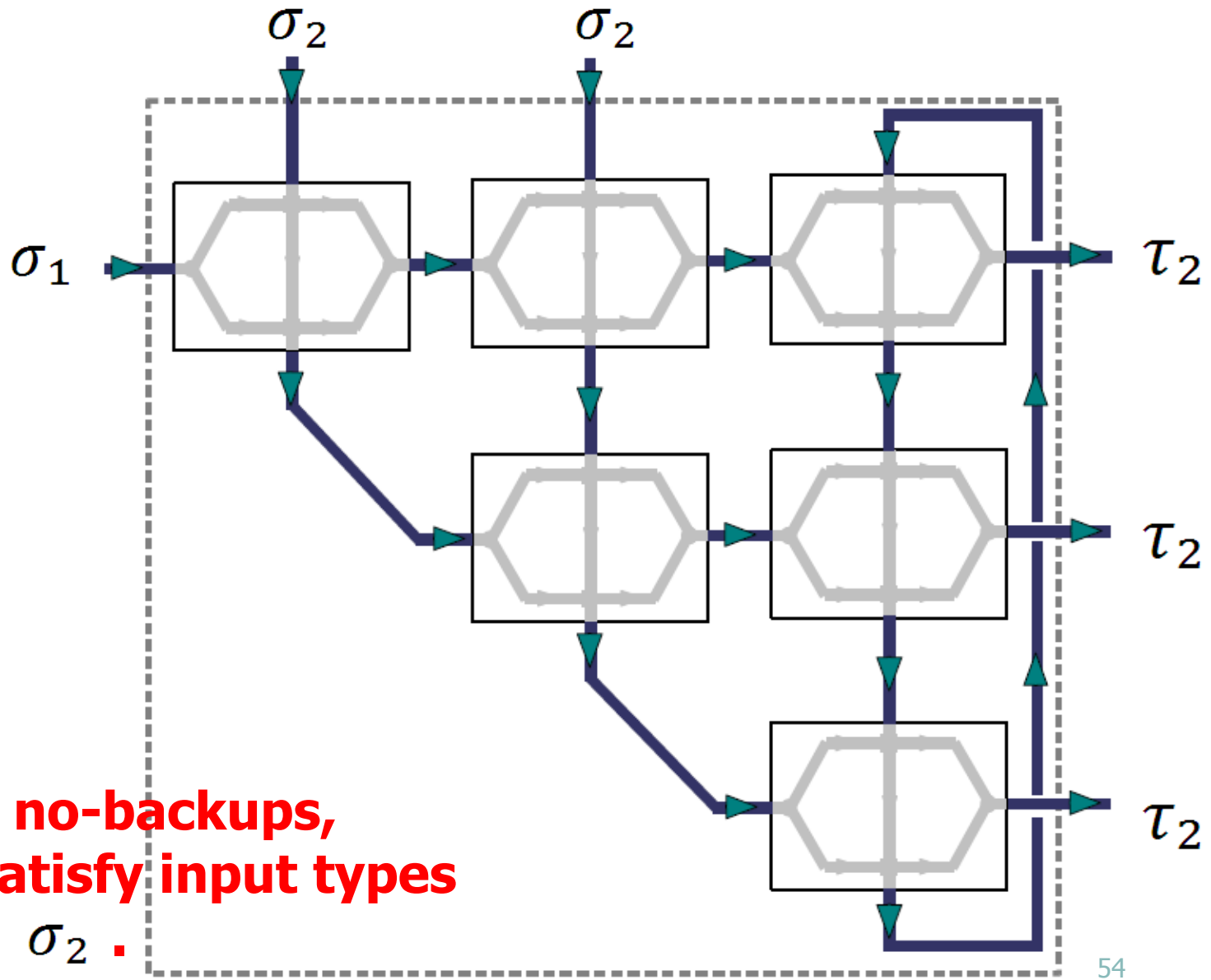
$\tau_2 <: \sigma_1$

$\tau_2 \not<: \sigma_2$



We can ignore the recorded constraint $k_2 + k_3 \leq 70$ because it regulates traffic densities inside the encapsulation/module.

Safe Composition of Several Modules (based on spec 1)



Guaranteed no-backups, if in-flows satisfy input types σ_1, σ_2, and σ_2.

Tiny Traffic Network, as a module based on 2nd specification

$\sigma_1' = [20,60]/[40,50]$

$\sigma_2' = [30,60]/[40,60]$

$\tau_1' = [30,60]/[40,60]$

$\tau_2' = [30,60]/[40,50]$

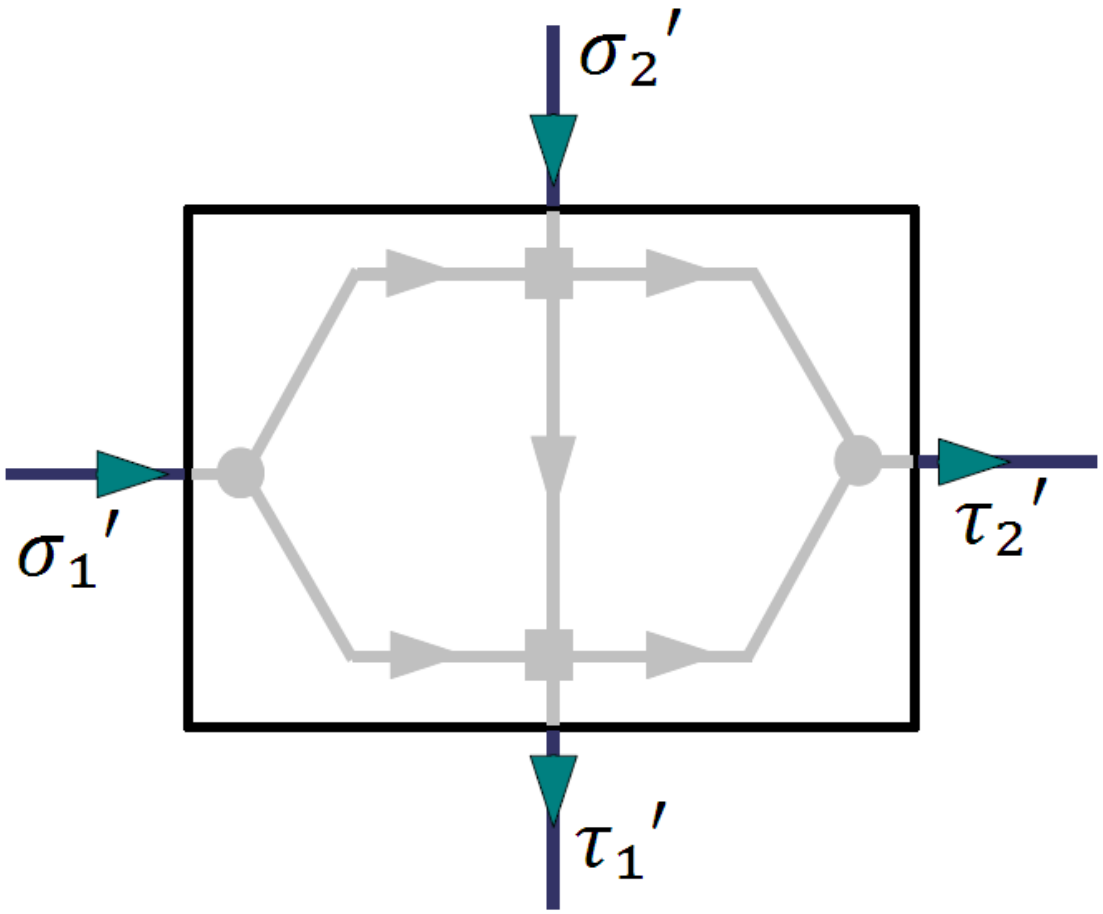
subtyping relationships:

$\tau_1' \not<: \sigma_1'$

$\tau_1' <: \sigma_2'$

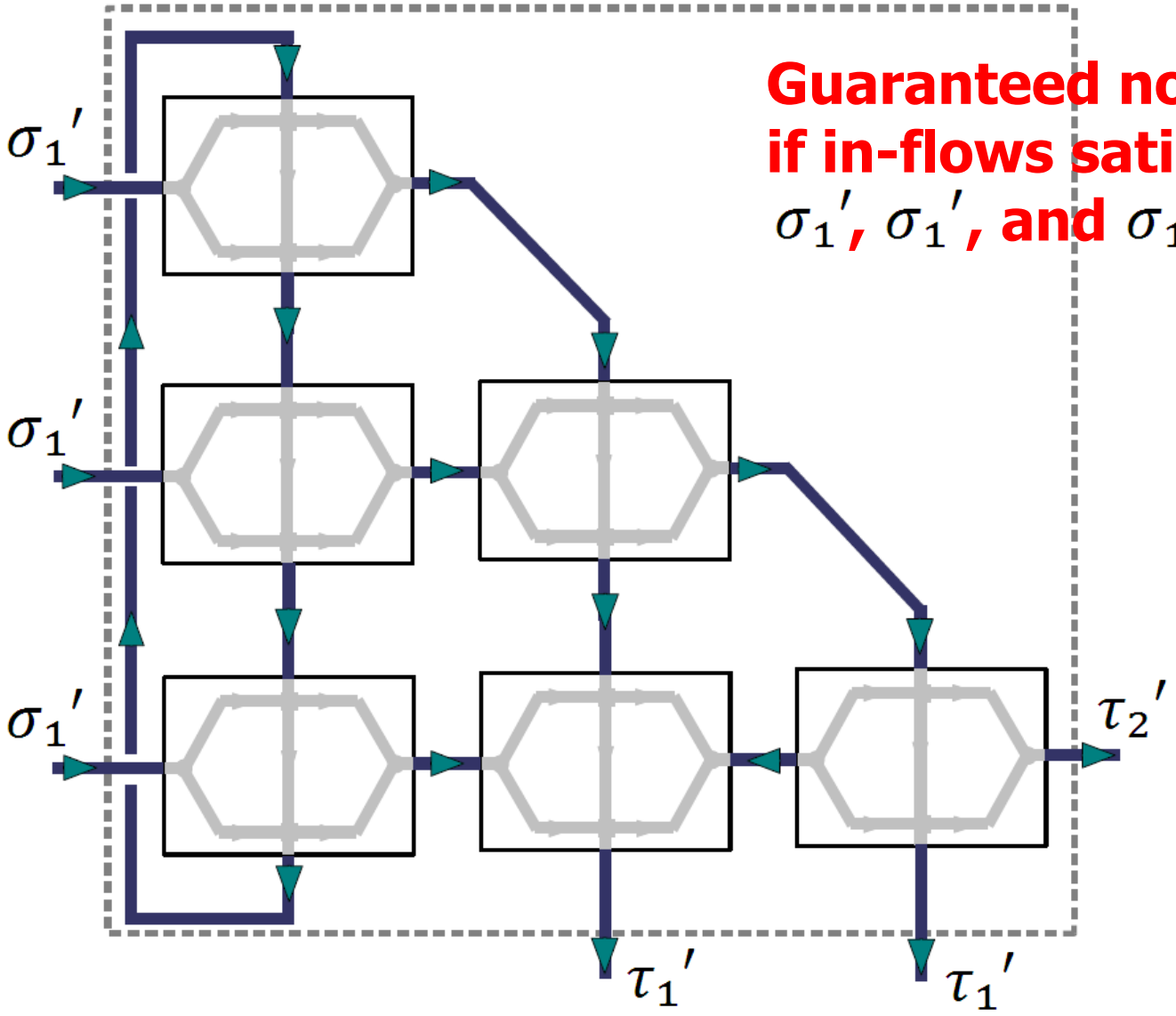
$\tau_2' <: \sigma_1'$

$\tau_2' <: \sigma_2'$



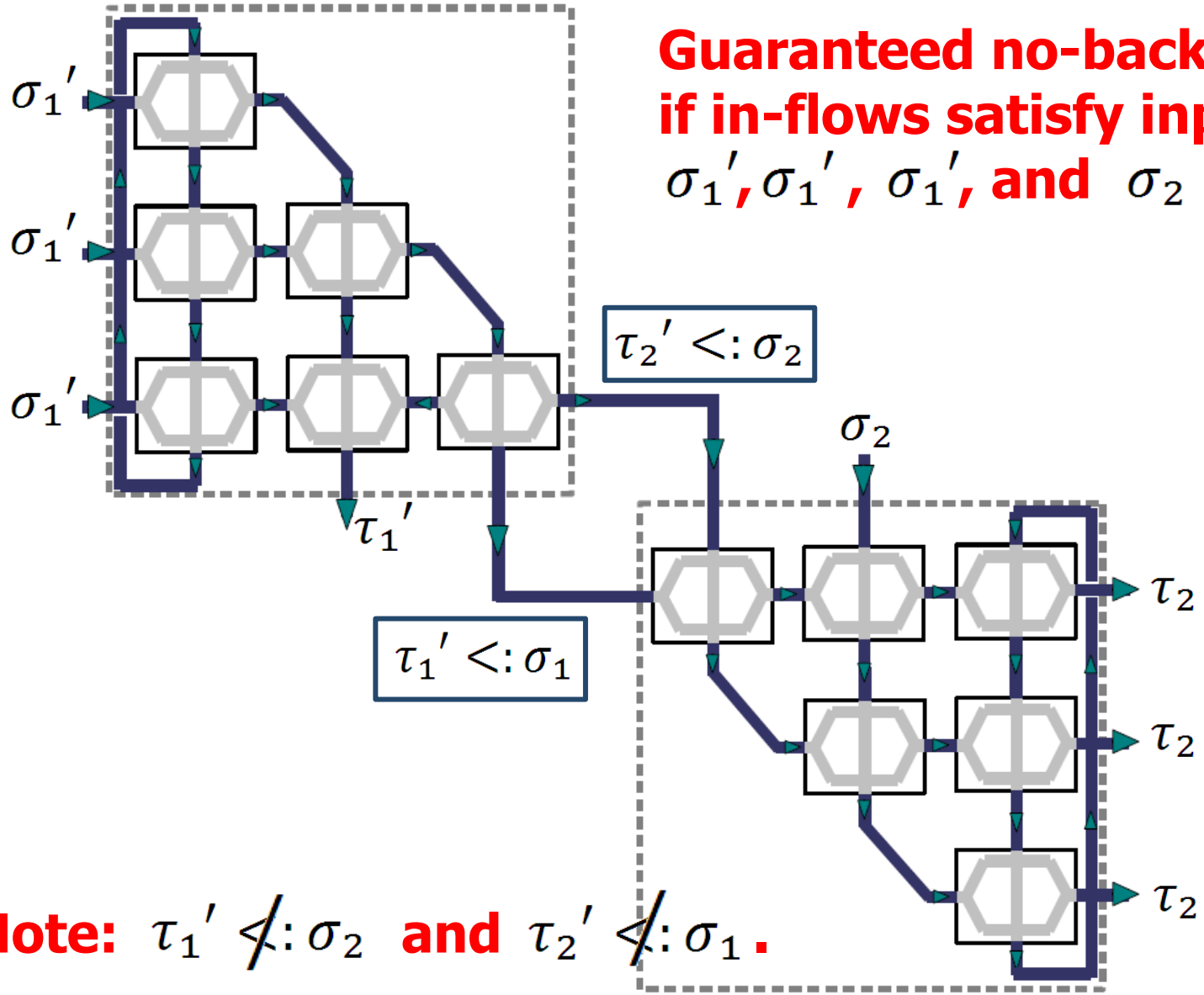
We can ignore the recorded constraint $k_2 + k_3 \leq 50$ because it regulates traffic densities inside the encapsulation/module.

Another Safe Composition of Modules (based on spec 2)



Guaranteed no-backups, if in-flows satisfy input types σ_1', σ_1', and σ_1' .

More Safe Composition of Several Modules ...



Guaranteed no-backups, if in-flows satisfy input types $\sigma_1', \sigma_1', \sigma_1'$, and σ_2.

Note: $\tau_1' \not<: \sigma_2$ **and** $\tau_2' \not<: \sigma_1$.

Tiny Traffic Network + Objective Functions

Constraints

safe local intervals/types:

$$20 \le v_1, v_2, v_3 \le 80$$

$$20 \le k_1, k_2, k_3 \le 90$$

$$30 \le v_4, v_5 \le 60$$

$$40 \le k_4, k_5 \le 70$$

no backups:

$$k_1 = k_2 + k_3$$

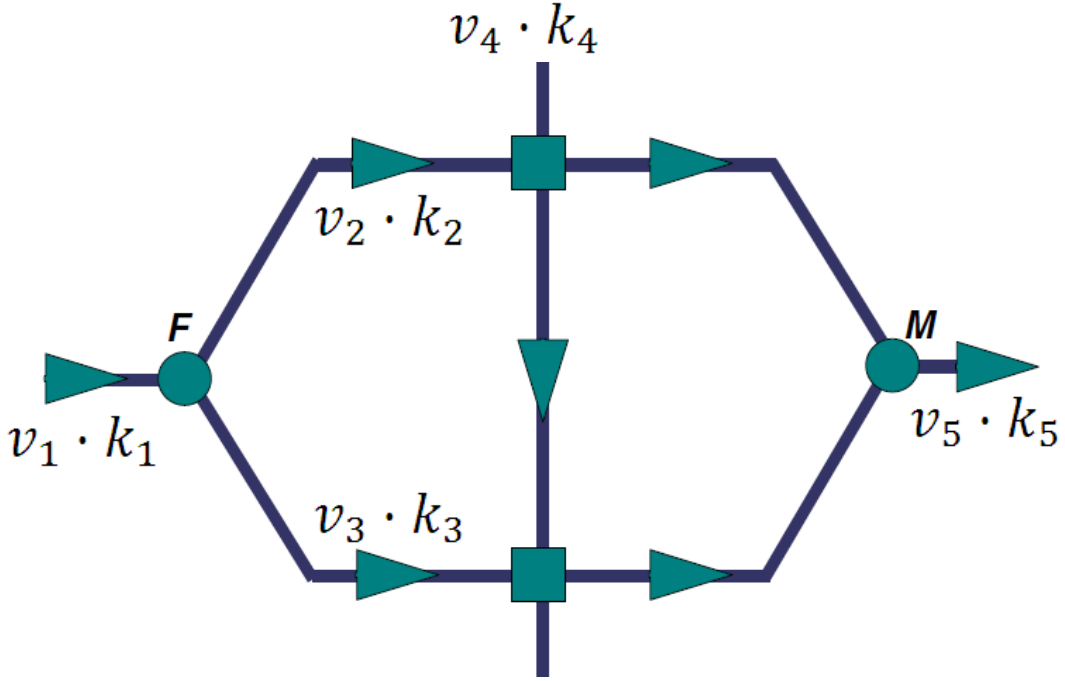
$$k_2 + k_3 = k_5$$

$$k_2 + k_4 \le 100$$

$$k_3 + k_4 \le 100$$

$$v_1 \cdot k_1 \le v_2 \cdot k_2 + v_3 \cdot k_3$$

$$v_2 \cdot k_2 + v_3 \cdot k_3 \le v_5 \cdot k_5$$



new

Several Objectives (among others)

A. **maximize sum of in-flows:** $v_1 \cdot k_1 + v_4 \cdot k_4$

B. **minimize in-flow diff:** $|v_1 \cdot k_1 - v_4 \cdot k_4|$

minimize out-flow diff: $|v_4 \cdot k_4 - v_5 \cdot k_5|$

C. **minimize change in kinetic energies:**

$$(\tfrac{1}{2}) \cdot k_2 \cdot (|{v_1}^2 - {v_2}^2| + |{v_2}^2 - {v_5}^2|) +$$
$$(\tfrac{1}{2}) \cdot k_3 \cdot (|{v_1}^2 - {v_3}^2| + |{v_3}^2 - {v_5}^2|)$$

Tiny Traffic Network + Objective Functions

Two approaches to solving objectives A, B, and C, without violating constraints for no-backups:

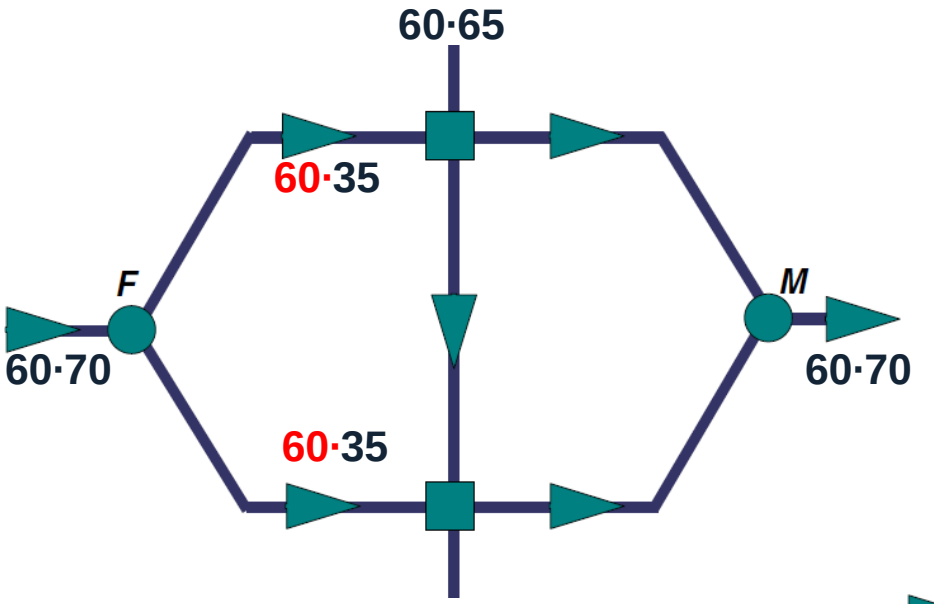
1. **Find solution or solutions, if any, ignoring inferred types for no-backups.**
2. **Find solution or solutions, if any, respecting inferred types for no-backups.**

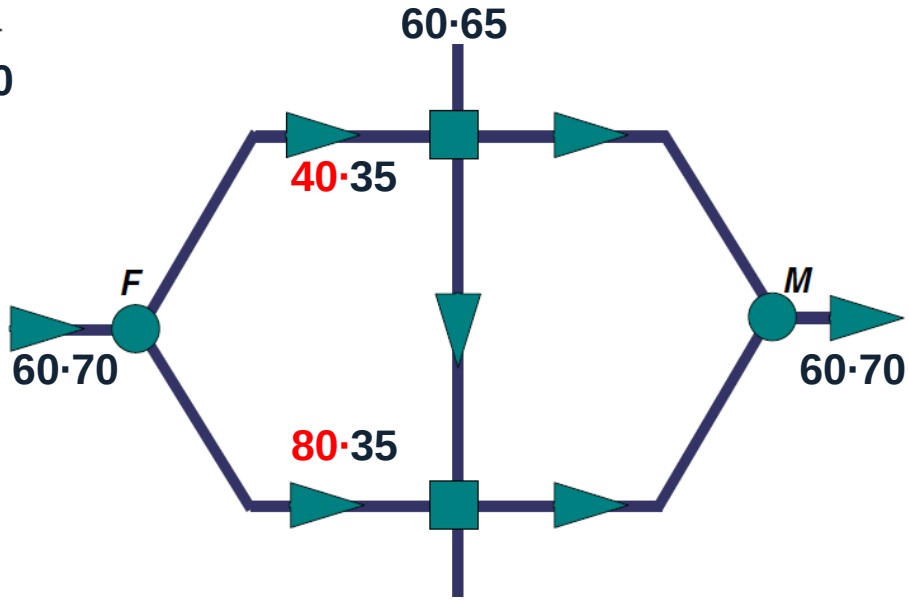
- **Approach 1 will return optimal solutions, at the price of higher complexity, forcing us to revisit all the constraints for no-backups. Or it is impossible because of unavailable global communication.**
- **Approach 2 will return less-than-optimal solutions, but with the benefit of not revisiting the original constraints for no-backups (except for the recorded ones).**

Objectives A, B, and C can be considered jointly (more difficult to solve) or separately (easier to solve).

Tiny Traffic Network + Objective A

Solutions for objective A, ignoring inferred types for no-backups:



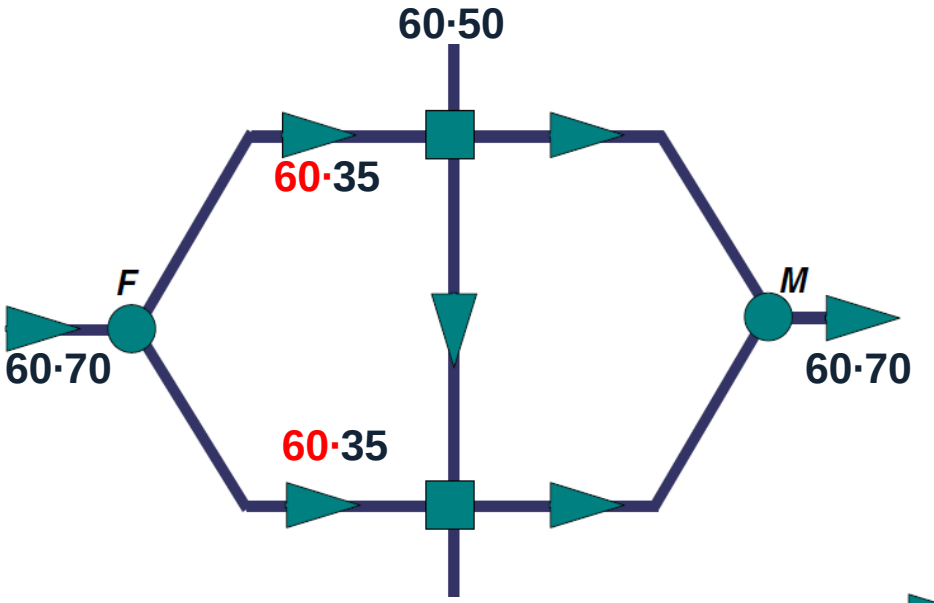


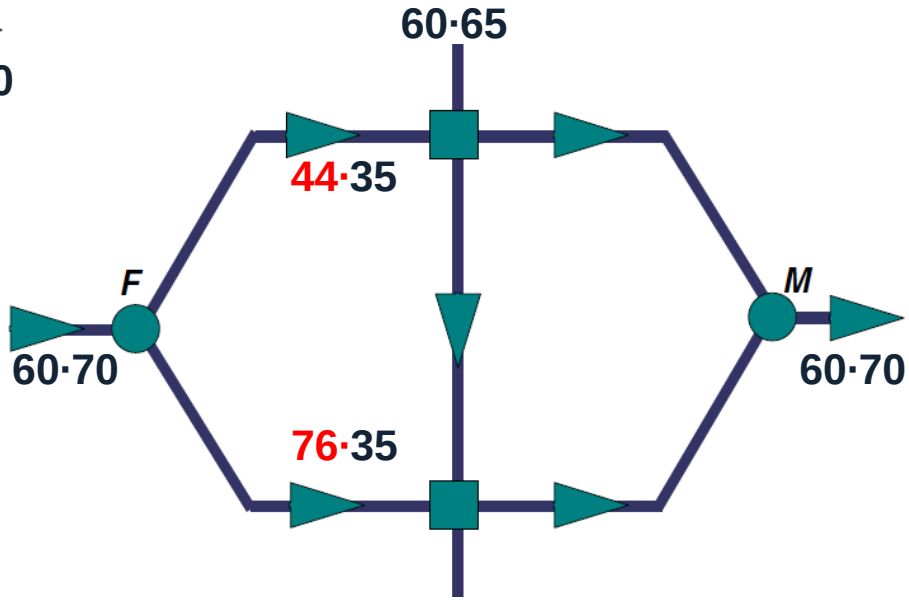
Only difference between solutions is in the internal velocities (in red)

Among all solutions for objective A, top solution is only solution for objective C

Tiny Traffic Network + Objective A

Solutions for objective A, respecting inferred types for no-backups (in the 1-st specification):



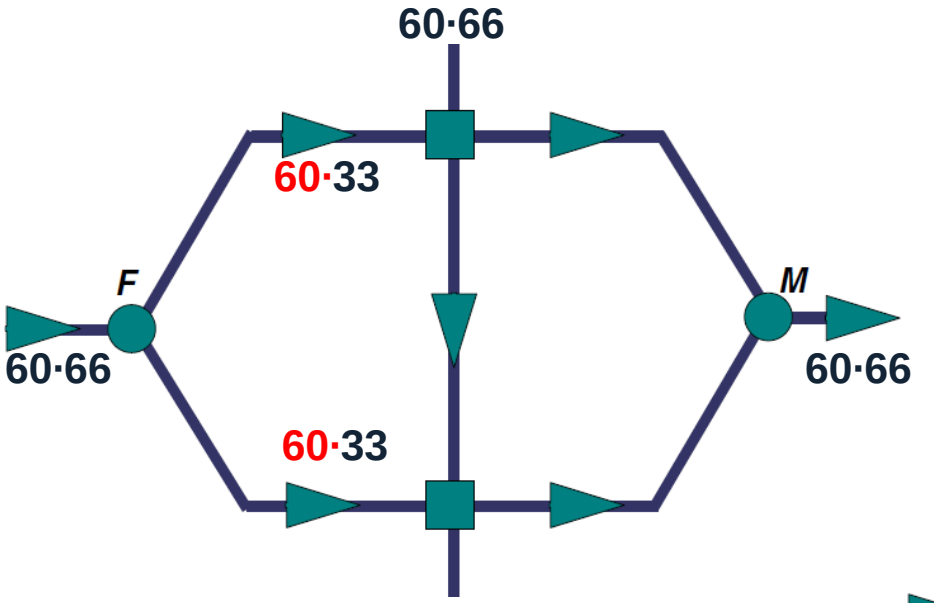


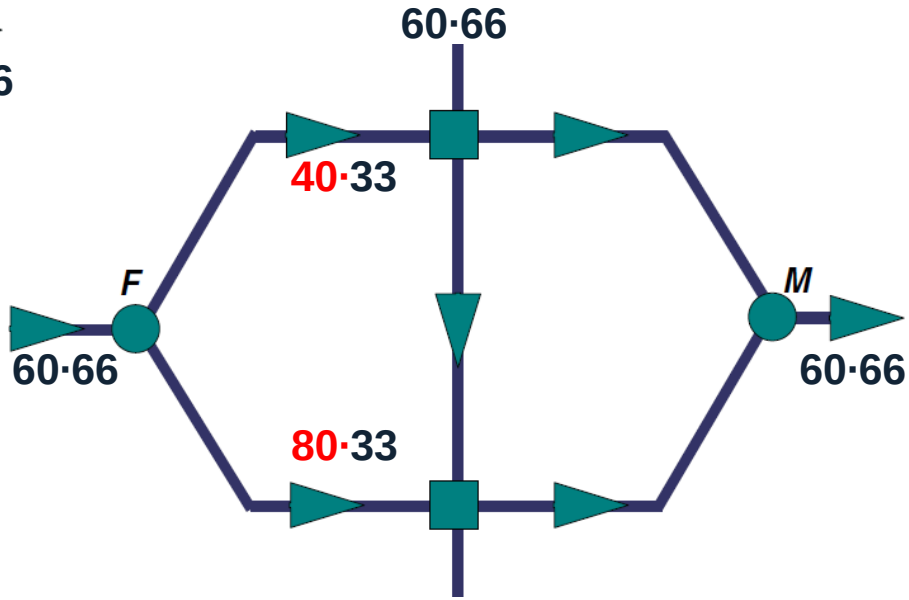
Only difference between solutions is in the internal velocities (in red)

Among all solutions for objective A respecting inferred types for no-backups, top solution is only solution for objective C

Tiny Traffic Network + Objective B

Maximal-flow (integer) solutions for objective B, ignoring inferred types for no-backups:



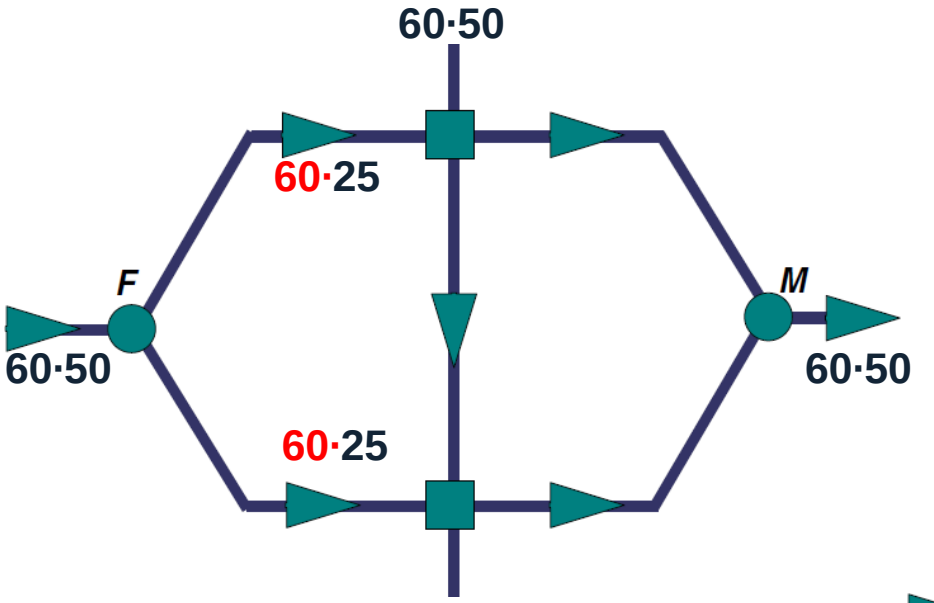


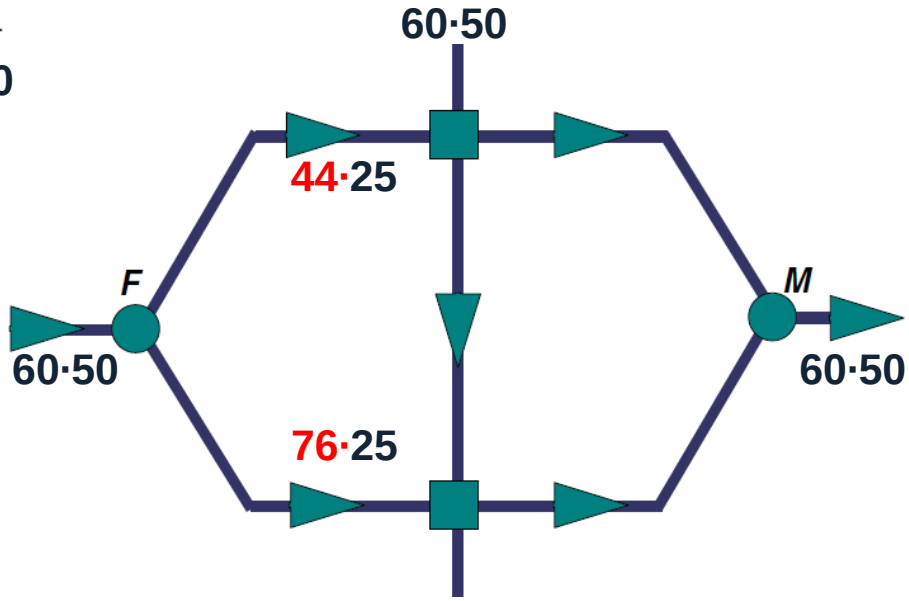
Only difference between solutions is in the internal velocities (in red)

Among all solutions for objective B, top solution is only solution for objective C

Tiny Traffic Network + Objective B

Maximal-flow (integer) solutions for objective B, respecting inferred types for no-backups (in the 1-st specification):



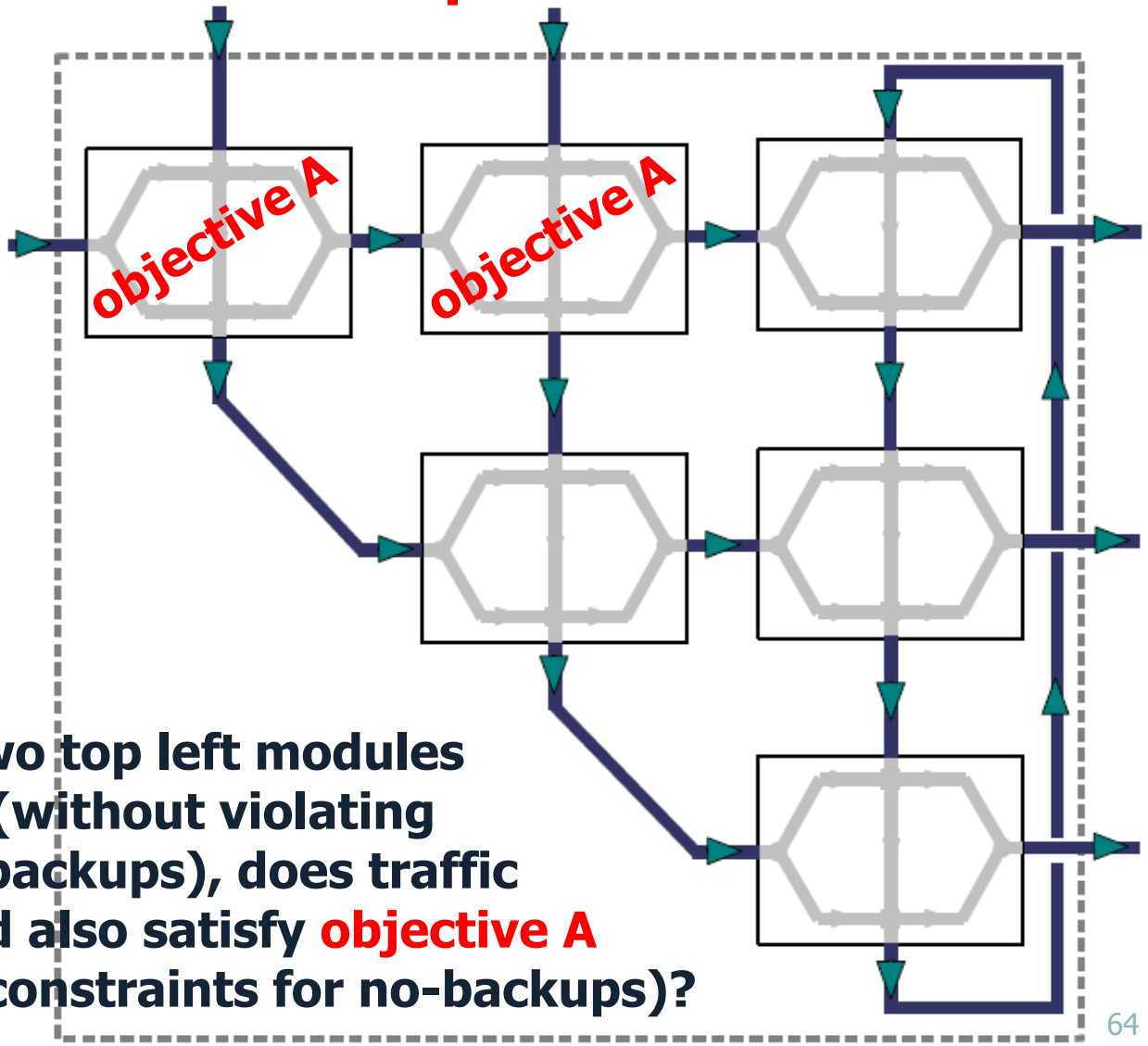


Only difference between solutions is in the internal velocities (in red)

Among all solutions for objective B respecting inferred types for no-backups, top solution is only solution for objective C

Is the Composition Safe?

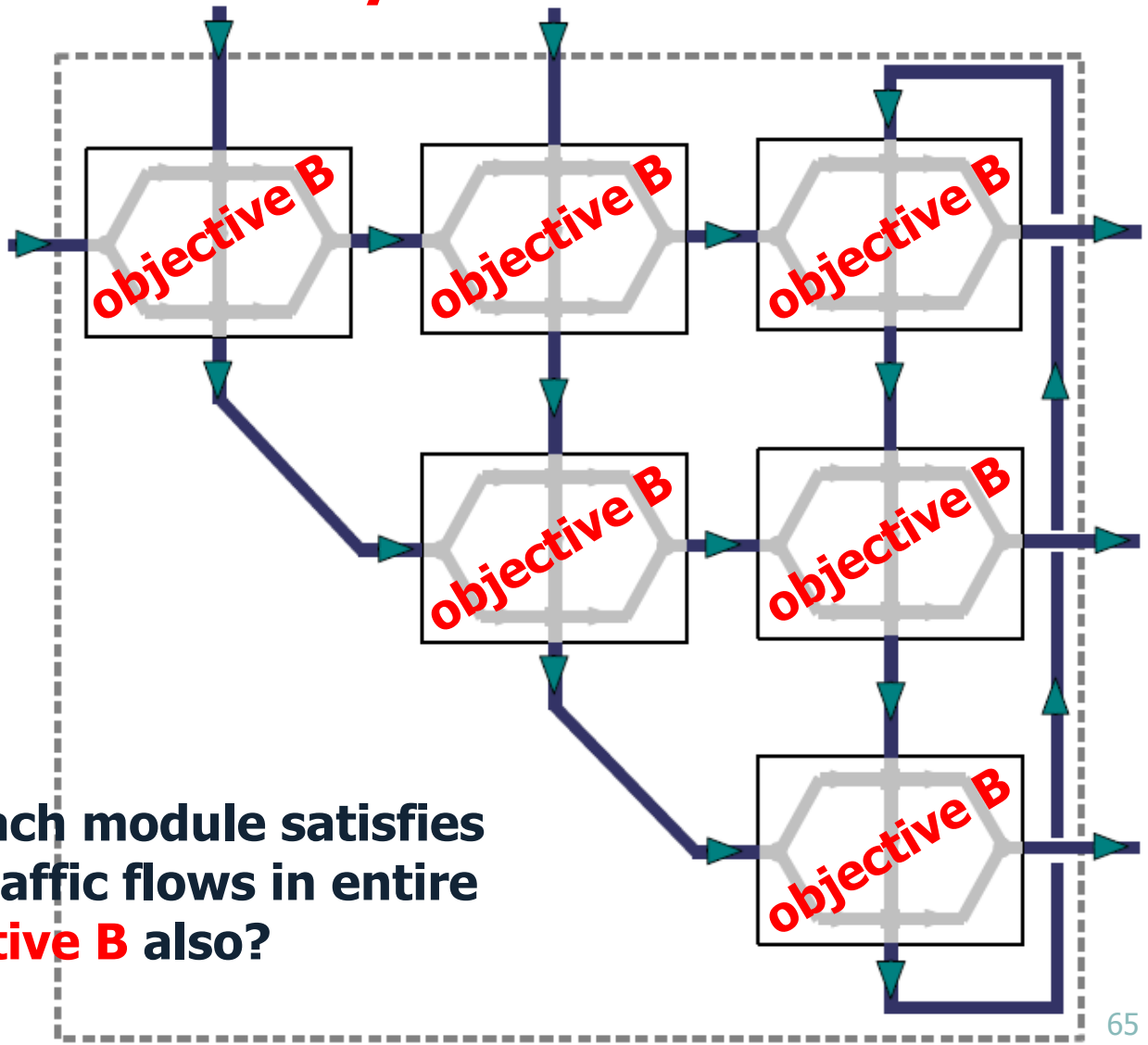
Objective A Satisfied in Two Top Left Modules



If traffic flows in two top left modules satisfy objective A (without violating constraints for no-backups), does traffic flows for entire grid also satisfy objective A (without violating constraints for no-backups)?

Is the Composition Safe?

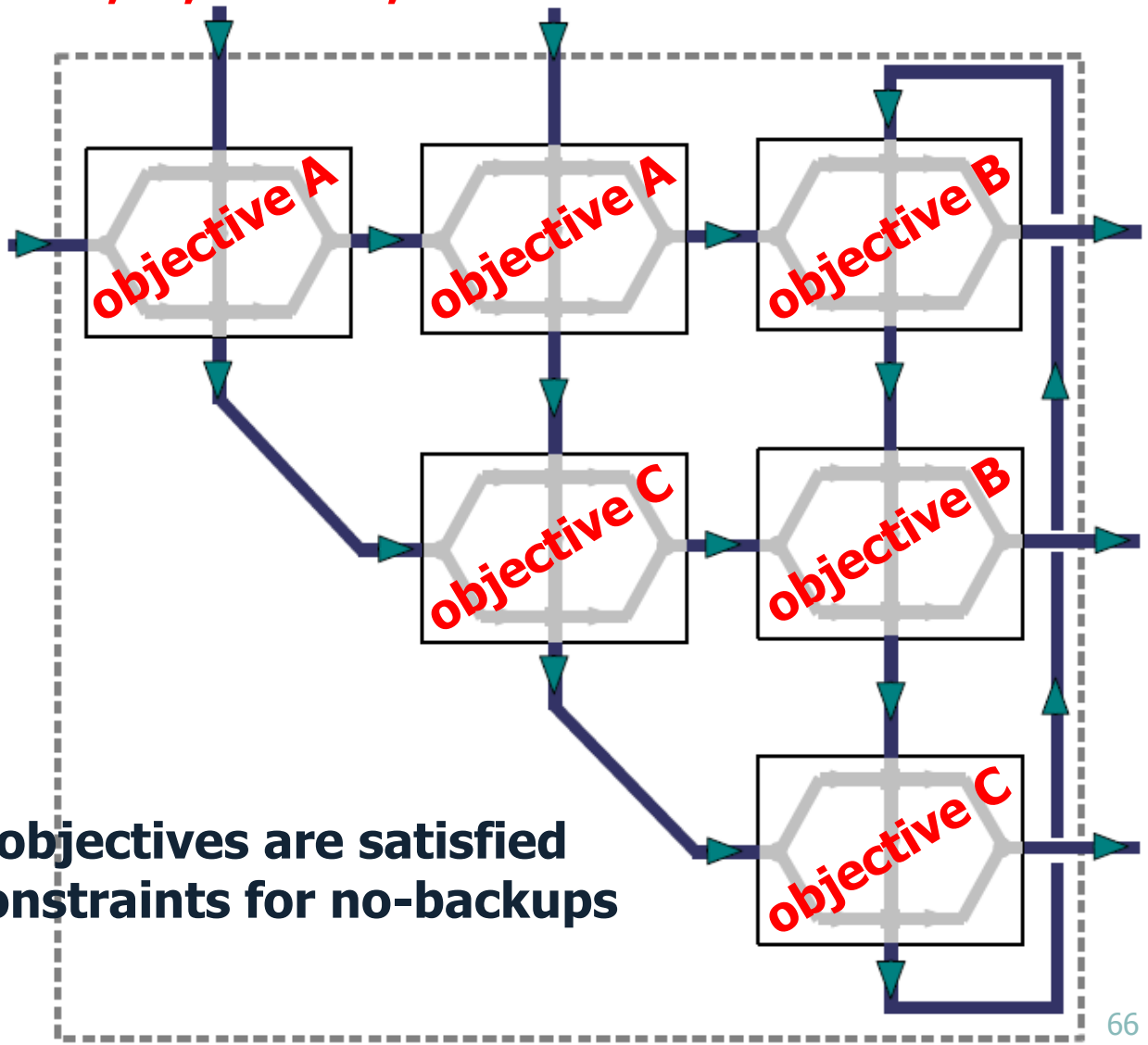
Objective B Satisfied in Every Module



If traffic flows in each module satisfies objective B, does traffic flows in entire grid satisfies objective B also?

Is the Composition Safe?

Mixing Objectives A, B, and C, in Different Modules



Which of the three objectives are satisfied globally? Are the constraints for no-backups satisfied?

Complexity Considerations

Cost of whole-system analysis, ignoring inferred types for no-backups, and finding solutions using:

- linear programming
- integer linear programming
- linearly constrained quadratic programming
- quadratically constrained quadratic programming
- worse

Questions So Far?

Formal Methods in Networking Studies …

1. **Compositional Analysis/Specification and its Benefits (mostly with A. Bestavros)**
2. An Application of Model Checking: Safe Composition of Arbitrary Network Protocols (mostly with A. Bradley and A. Bestavros)
3. Resource Allocation in Sensor Networks using a Strongly-Typed Domain-Specific Language (with M. Ocean and A. Bestavros)
4. **The Stable-Paths Problem and the Promise of an Automatic Lightweight Proof-Assistant (with K. Donnelly and A. Lapets)**

time permitting

Lightweight Formal Methods for the Development of High-Assurance Network Systems

Assaf Kfoury

with contributions from

Azer Bestavros, Adam Bradley, Andrei Lapets, and Michael Ocean

iBench Initiative
http://www.cs.bu.edu/groups/ibench/
snBench
http://csr.bu.edu/snbench/

Computer Science