A process algebra with global variables

(M.S. Bouwman, et al.)

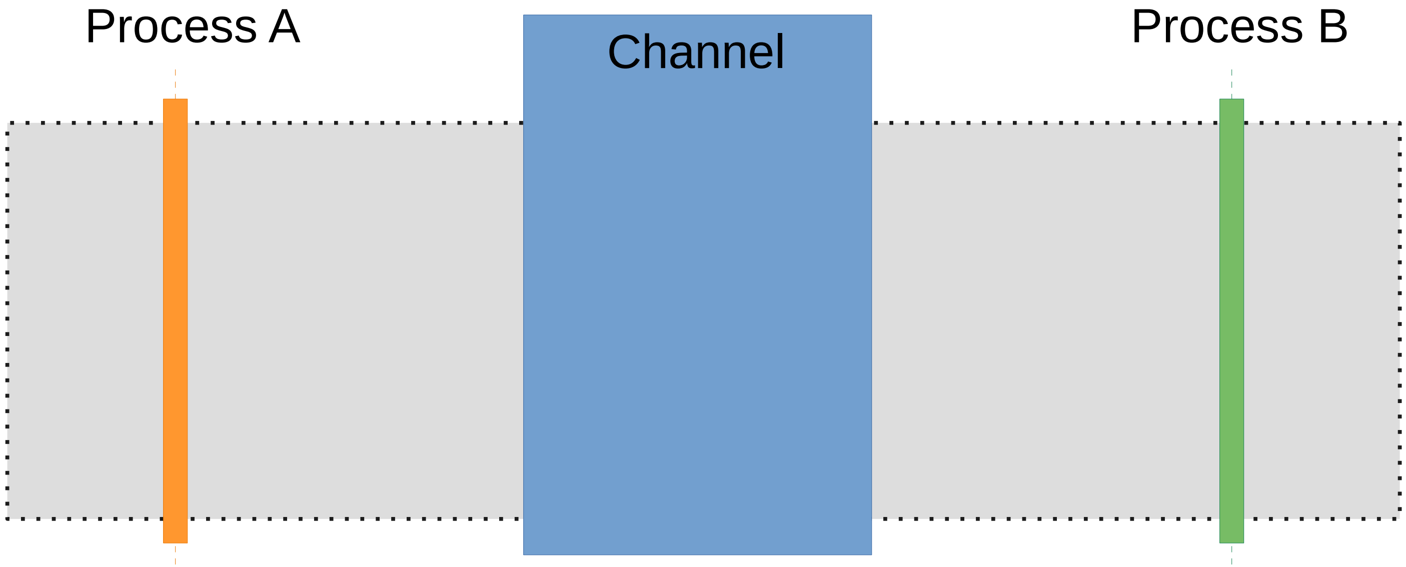
CTCT Seminar: Paper Presentation

Message Passing VS Shared Memory

- Many formalisms model communication via some form of *Message Passing*
- For example, via channels:

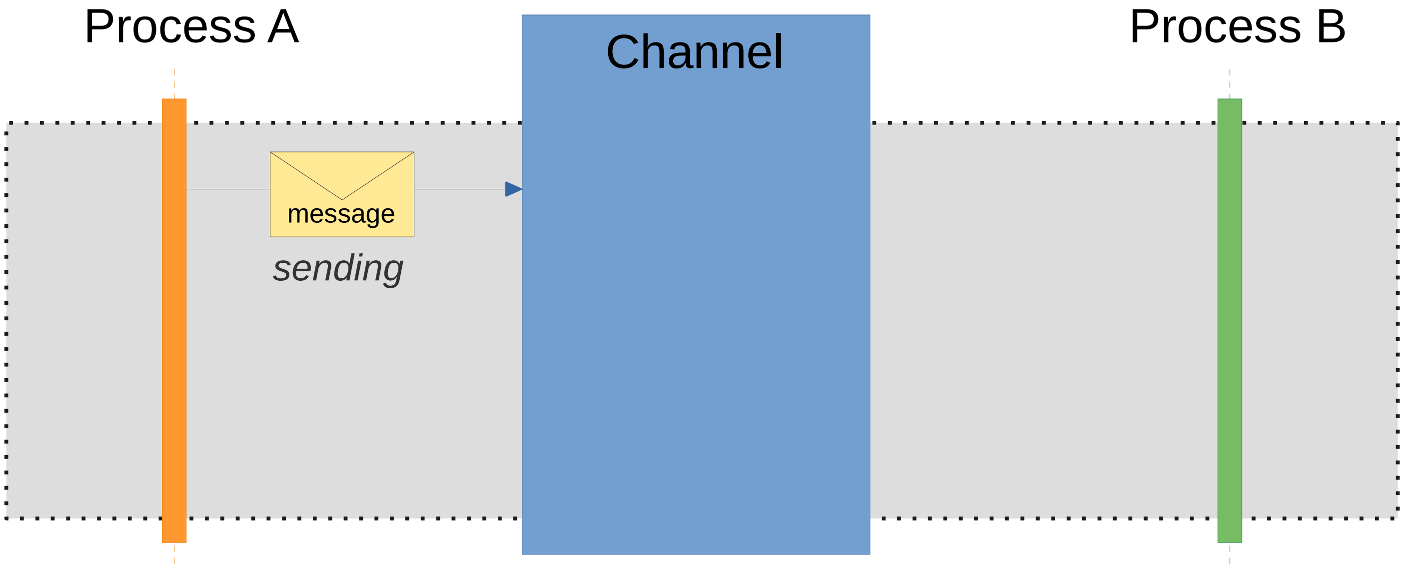
Message Passing VS Shared Memory

- Many formalisms model communication via some form of *Message Passing*
- For example, via channels:



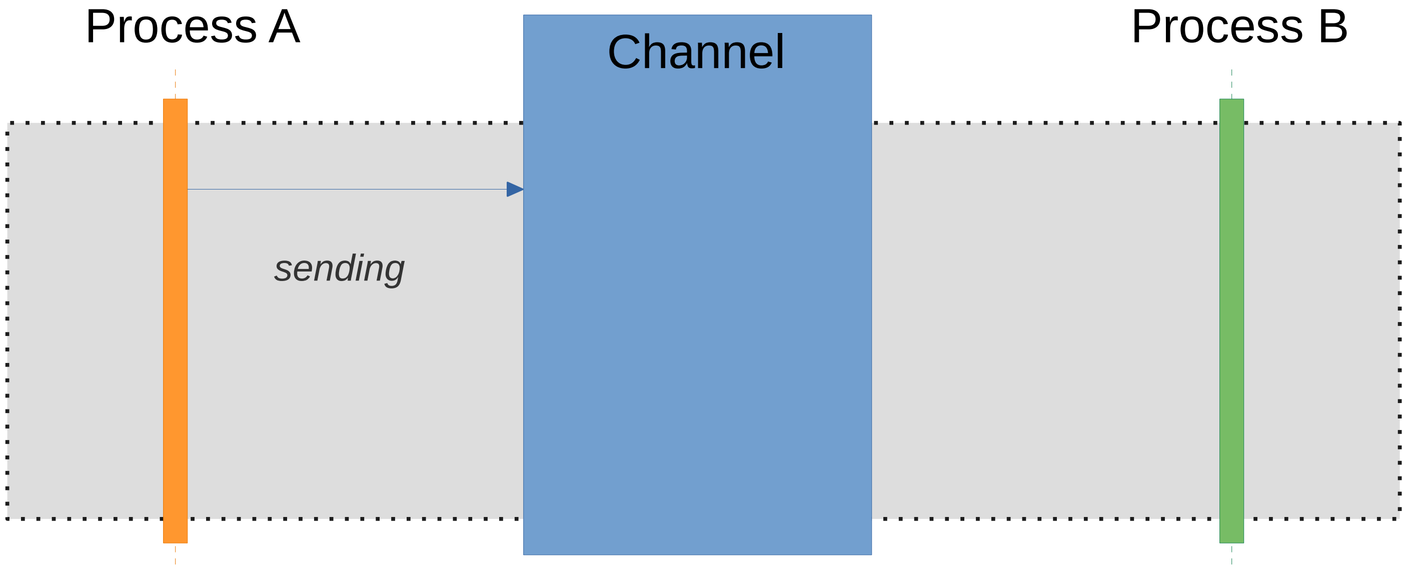
Message Passing VS Shared Memory

- Many formalisms model communication via some form of *Message Passing*
- For example, via channels:



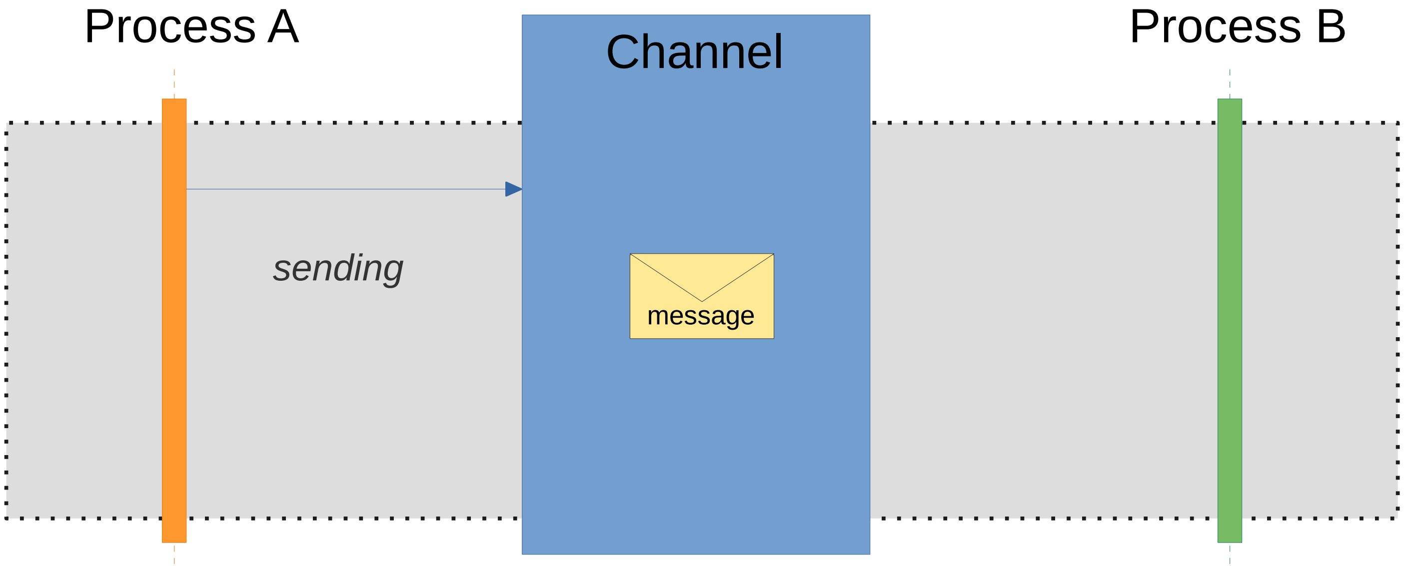
Message Passing VS Shared Memory

- Many formalisms model communication via some form of *Message Passing*
- For example, via channels:



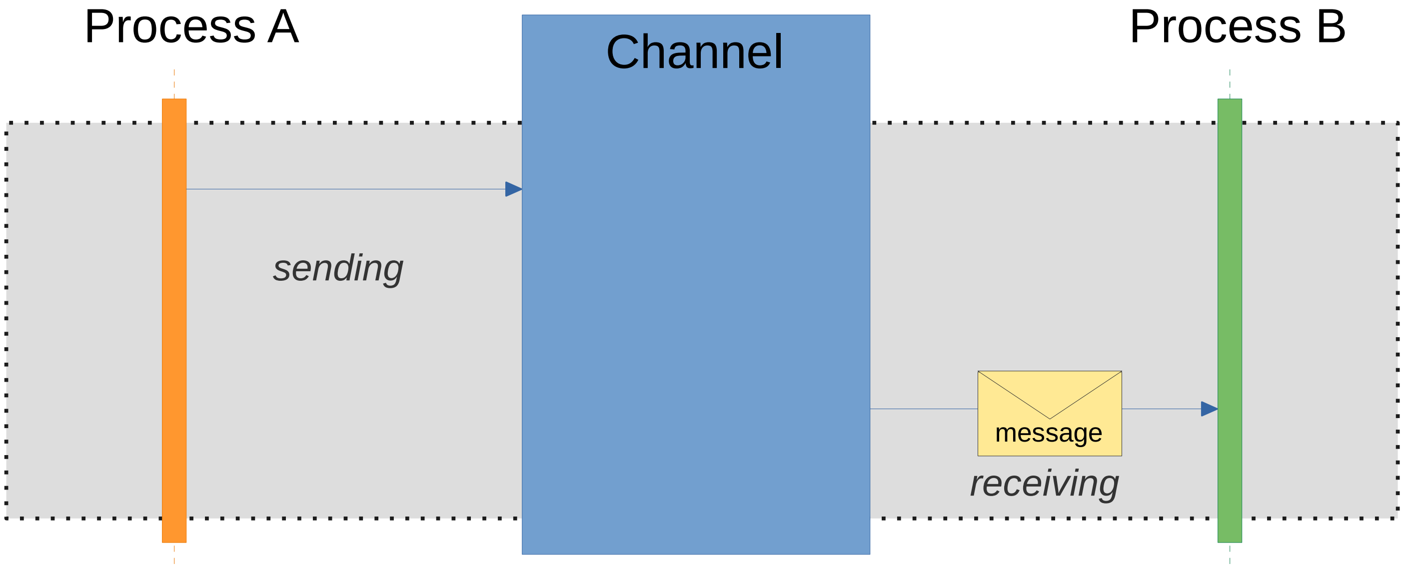
Message Passing VS Shared Memory

- Many formalisms model communication via some form of *Message Passing*
- For example, via channels:



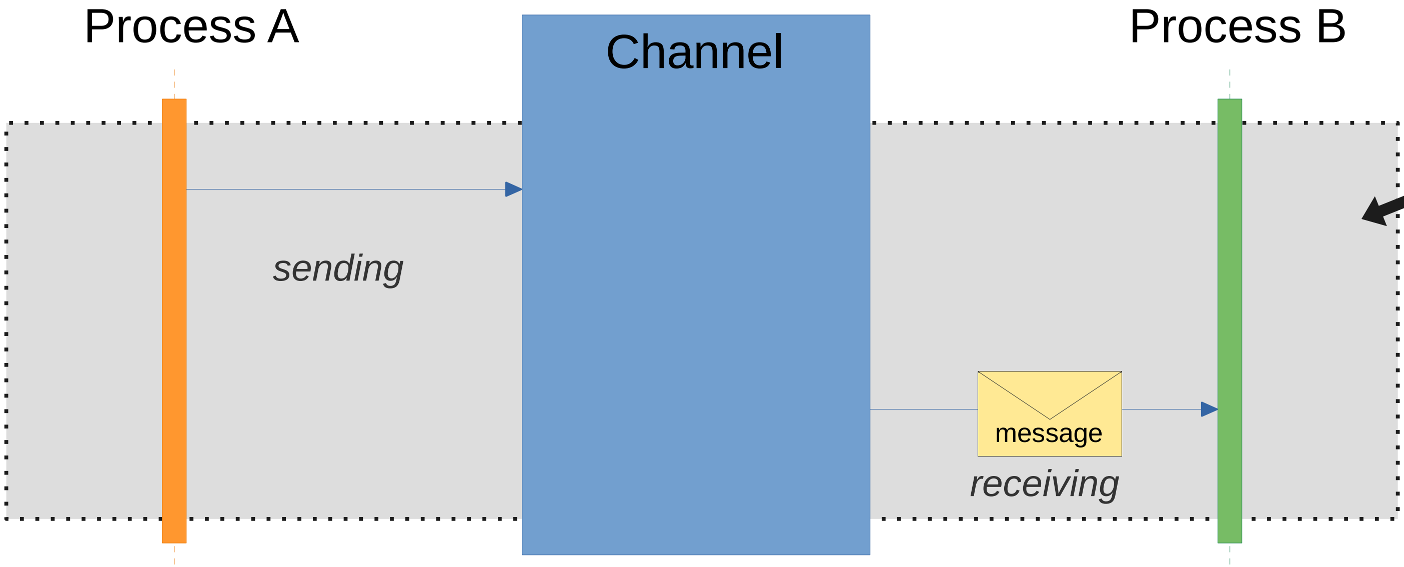
Message Passing VS Shared Memory

- Many formalisms model communication via some form of *Message Passing*
- For example, via channels:



Message Passing VS Shared Memory

- Many formalisms model communication via some form of *Message Passing*
- For example, via channels:



May **synchronize** A and B

or

may be **asynchronous**

Message Passing VS Shared Memory

Message Passing VS Shared Memory

Message passing has its benefits

Message Passing VS Shared Memory

Message passing has its benefits

Message Passing VS Shared Memory

TECHNISCHE UNIVERSITÄT DARMSTADT

Message passing has its benefits

- Fits well if modeled processes are **distributed across network** etc.

Message Passing VS Shared Memory

Message passing has its benefits

- Fits well if modeled processes are **distributed across network** etc.
- Makes receiver of communication **explicit**

Message Passing VS Shared Memory

TECHNISCHE UNIVERSITÄT DARMSTADT

Message passing has its benefits

- Fits well if modeled processes are **distributed across network** etc.
- Makes receiver of communication **explicit**
- **Avoids data races**

Message Passing VS Shared Memory

TECHNISCHE UNIVERSITÄT DARMSTADT

Message passing has its benefits

- Fits well if modeled processes are **distributed across network** etc.
- Makes receiver of communication **explicit**
- **Avoids data races**
- Provides **synchronization** point

Message Passing VS Shared Memory

Message passing has its benefits

- Fits well if modeled processes are **distributed across network** etc.
- Makes receiver of communication **explicit**
- **Avoids data races**
- Provides **synchronization** point
- …

Message Passing VS Shared Memory

Message passing has its benefits

- Fits well if modeled processes are **distributed across network** etc.
- Makes receiver of communication **explicit**
- **Avoids data races**
- Provides **synchronization** point
- …

But sometimes **Shared Memory / Global Variables** are the right model

Message Passing VS Shared Memory

Message passing has its benefits

- Fits well if modeled processes are **distributed across network** etc.
- Makes receiver of communication **explicit**
- **Avoids data races**
- Provides **synchronization** point
- …

But sometimes **Shared Memory / Global Variables** are the right model

Message Passing VS Shared Memory

TECHNISCHE UNIVERSITÄT DARMSTADT

Message passing has its benefits

- Fits well if modeled processes are **distributed across network** etc.
- Makes receiver of communication **explicit**
- **Avoids data races**
- Provides **synchronization** point
- …

But sometimes **Shared Memory / Global Variables** are the right model

- Message passing between threads is **implemented in shared memory** by OS

Message Passing VS Shared Memory

Message passing has its benefits

- Fits well if modeled processes are **distributed across network** etc.
- Makes receiver of communication **explicit**
- **Avoids data races**
- Provides **synchronization** point
- …

But sometimes **Shared Memory / Global Variables** are the right model

- Message passing between threads is **implemented in shared memory** by OS
- Shared memory often **faster** (if not distributed across network)

Message Passing VS Shared Memory

TECHNISCHE UNIVERSITÄT DARMSTADT

Message passing has its benefits

- Fits well if modeled processes are **distributed across network** etc.
- Makes receiver of communication **explicit**
- **Avoids data races**
- Provides **synchronization** point
- …

But sometimes **Shared Memory / Global Variables** are the right model

- Message passing between threads is **implemented in shared memory** by OS
- Shared memory often **faster** (if not distributed across network)
- Some information should be **available at all times**

Message Passing VS Shared Memory

TECHNISCHE UNIVERSITÄT DARMSTADT

Message passing has its benefits

- Fits well if modeled processes are **distributed across network** etc.
- Makes receiver of communication **explicit**
- **Avoids data races**
- Provides **synchronization** point
- …

But sometimes **Shared Memory / Global Variables** are the right model

- Message passing between threads is **implemented in shared memory** by OS
- Shared memory often **faster** (if not distributed across network)
- Some information should be **available at all times**
- …

Message Passing VS Shared Memory

Message passing has its benefits

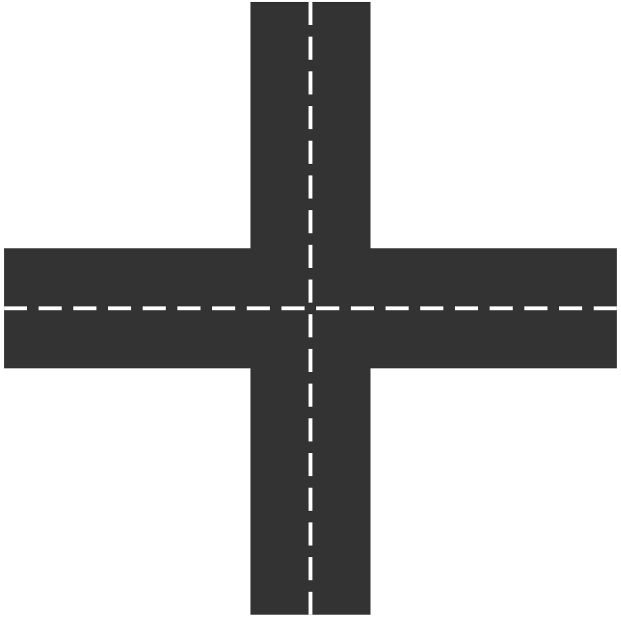
- Fits well if modeled processes are **distributed across network** etc.
- Makes receiver of communication **explicit**
- Provides **synchronization** point
- …

We'll look at an **example** for this!

But sometimes **Shared Memory / Global Variables** are the right model

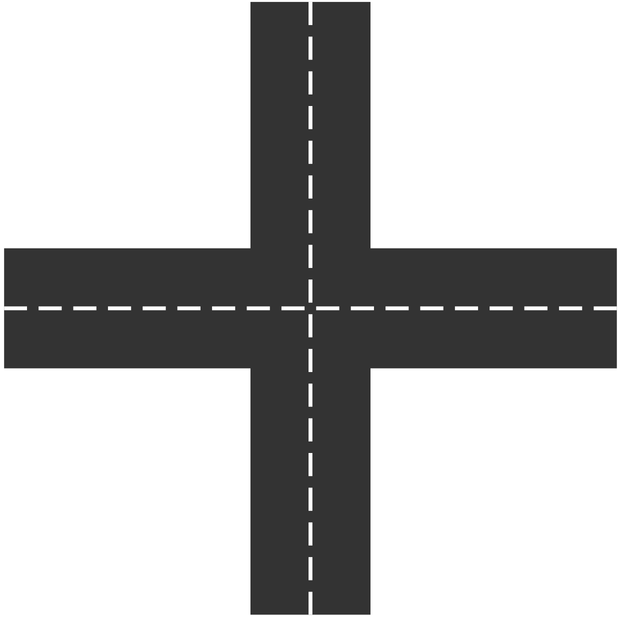
- Message passing between threads is **implemented in shared memory** by OS
- Shared memory often **faster** (if not distributed across network)
- Some information should be **available at all times**
- …

Example: Crossing



Example: Crossing

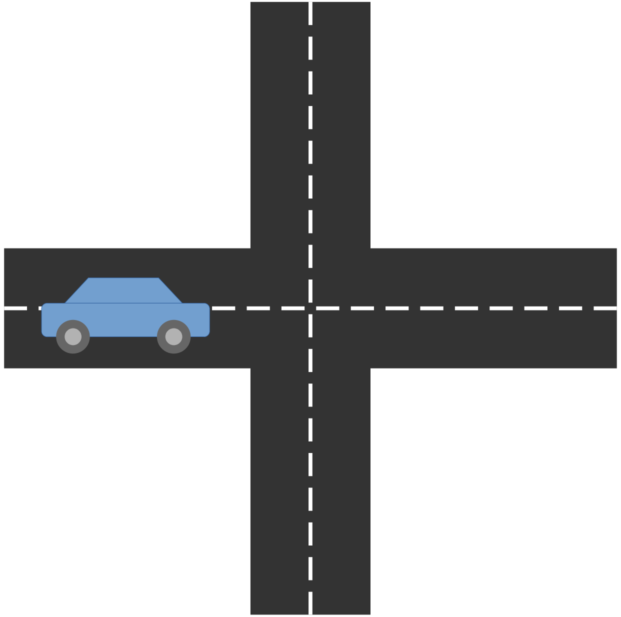
We want to model a crossing…



Example: Crossing

We want to model a crossing…

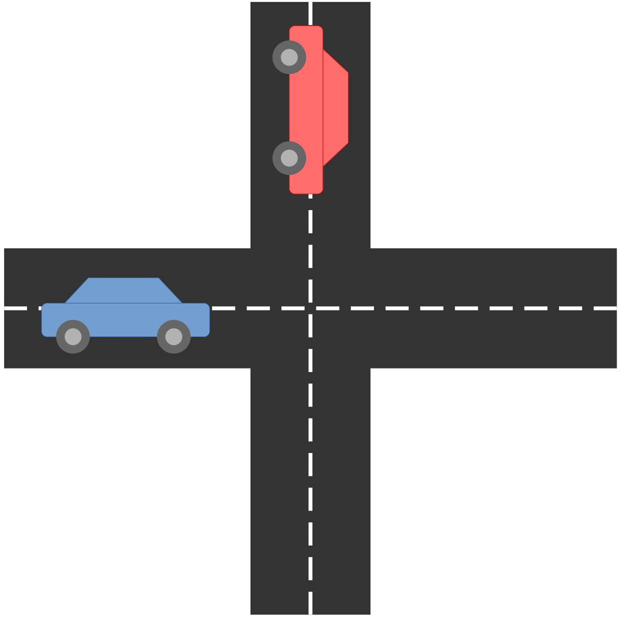
- ...a **car** is a **process**



Example: Crossing

We want to model a crossing…

- ...a **car** is a **process**
- ...and there can be **multiple** of them

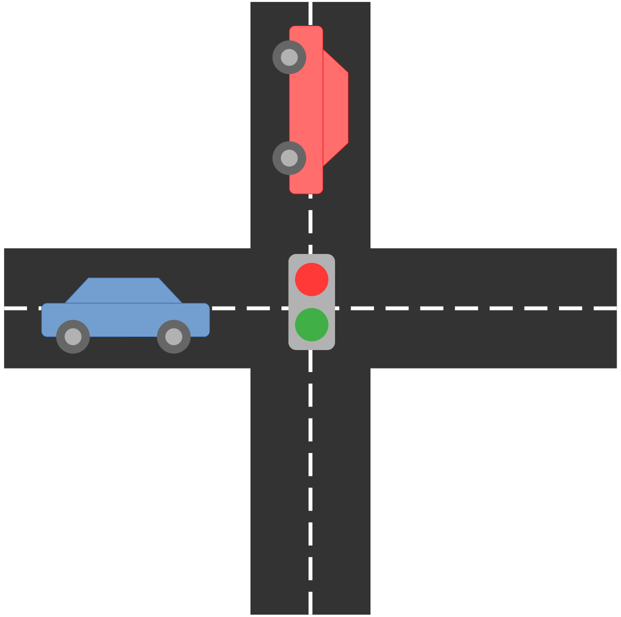


Example: Crossing

We want to model a crossing…

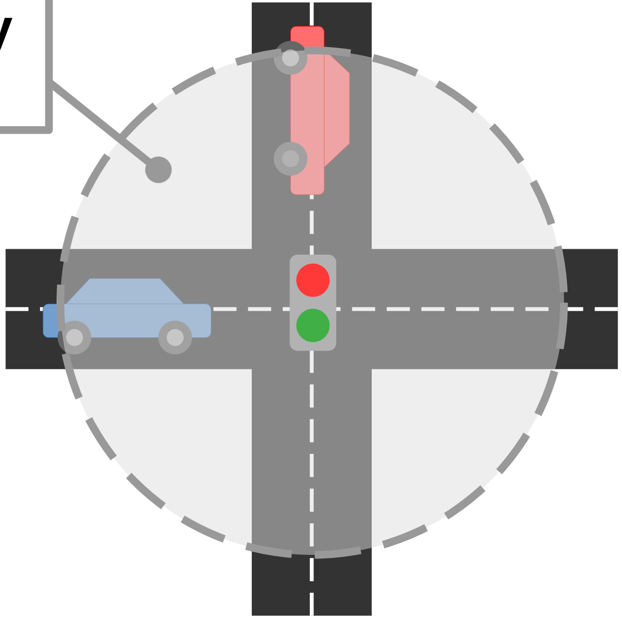
- ...a **car** is a **process**
- ...and there can be **multiple** of them
- ...the **traffic light** is also a **process**

Traffic light **switches anytime** between red/green



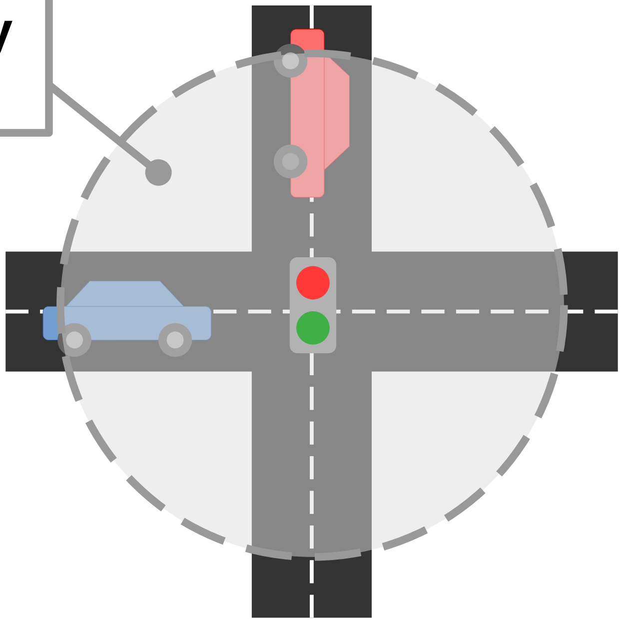
Example: Crossing

Light status is **always globally available** to **all cars**



Example: Crossing

Light status is **always globally available** to **all cars**

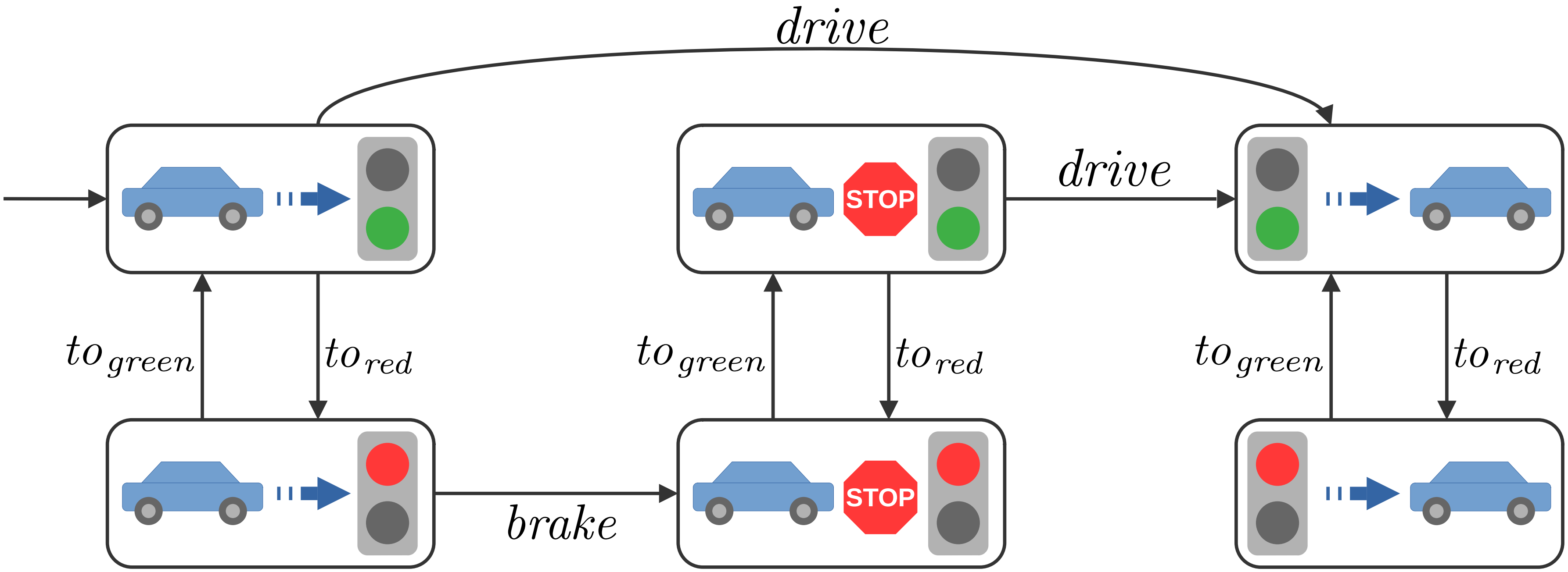


Message Passing is Poor Fit

- Cars don’t synchronously request light status
- Light does not spam status messages to cars

⇒ Model Light as **Global Variable!**

1-Car Transition System



Process Algebra Expressions

Primitives

λ	$v \downarrow d$
abstract actions	variable assignments

Process Algebra Expressions

TECHNISCHE UNIVERSITÄT DARMSTADT

Primitives

λ

abstract actions

$v \downarrow d$

variable assignments

Operators

$\square \parallel \square$

parallel composition

$\square \cdot \square$

sequential composition

$\square + \square$

choice

Process Algebra Expressions

Primitives

λ

abstract actions

$v \downarrow d$

variable assignments

Operators

$\square \parallel \square$

parallel composition

$\square \cdot \square$

sequential composition

$\square + \square$

choice

$(v = d) \rightarrow \square$

conditional

Recursive Definitions / "Calls"

$\langle\mathrm{name}\rangle \stackrel{\mathrm{def}}{=} \square$

$\langle\mathrm{name}\rangle$

Process Algebra Expressions

TECHNISCHE UNIVERSITÄT DARMSTADT

Primitives

λ

abstract actions

$v \downarrow d$

variable assignments

Operators

$\square \parallel \square$

parallel composition

$\square \cdot \square$

sequential composition

$\square + \square$

choice

$(v = d) \rightarrow \square$

conditional

Recursive Definitions / “Calls”

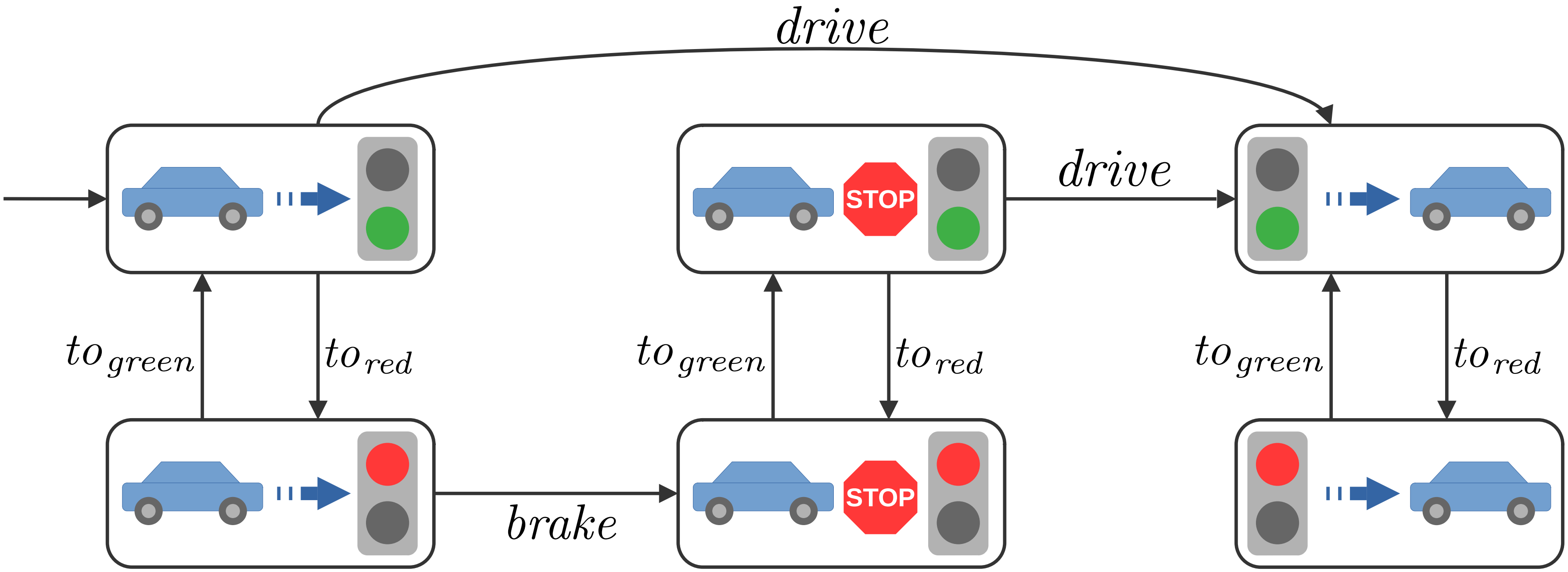
$\langle\text{name}\rangle \stackrel{\text{def}}{=} \square$

$\langle\text{name}\rangle$

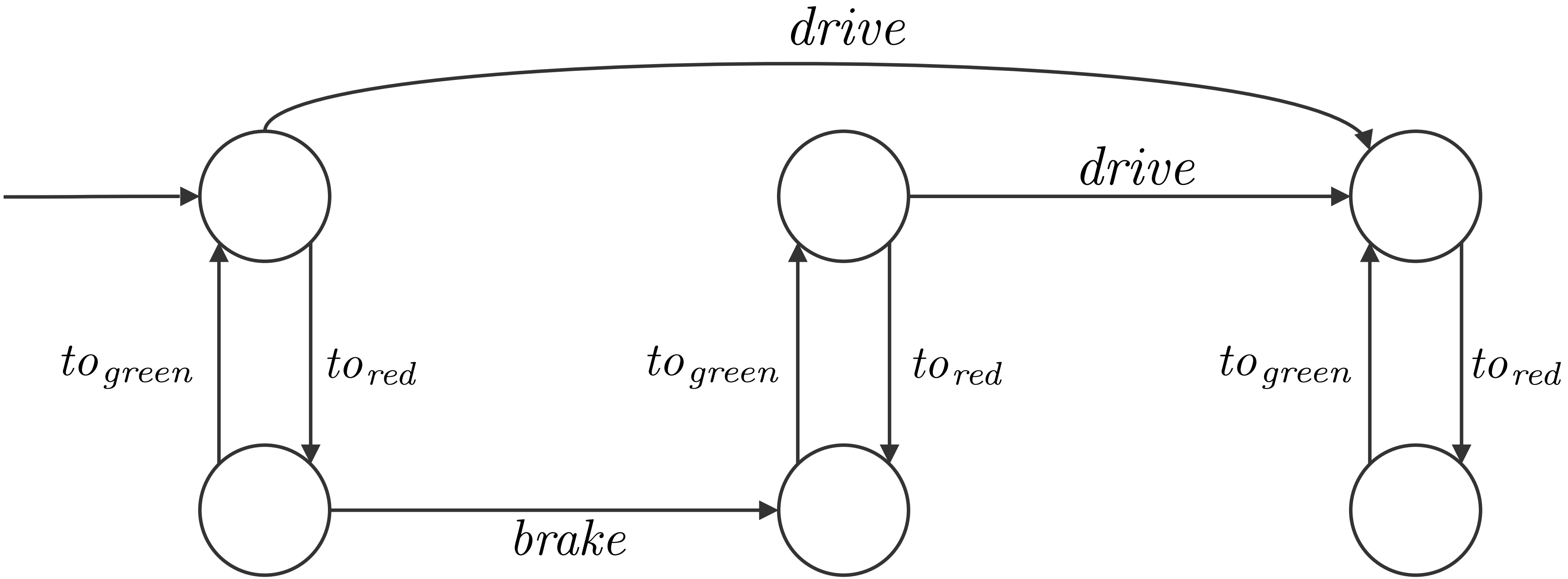
$$P \stackrel{\text{def}}{=} (a \parallel v \downarrow 42) \cdot P$$

$$\text{"Assign 42 to } v \text{ and perform action } a \text{ in parallel, then repeat."}$$

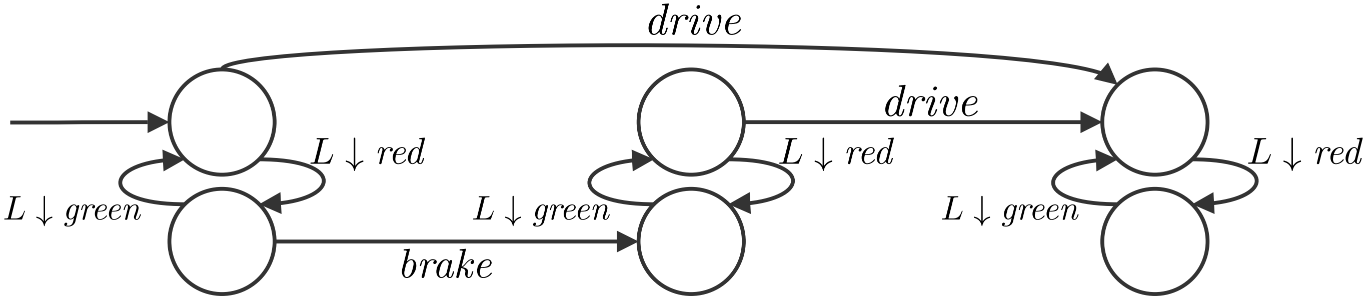
Transition System from Algebra Expression



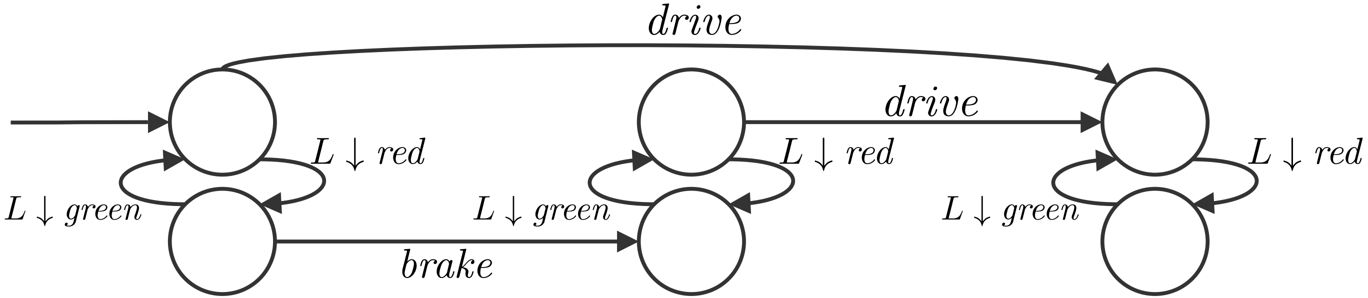
Transition System from Algebra Expression



Transition System from Algebra Expression

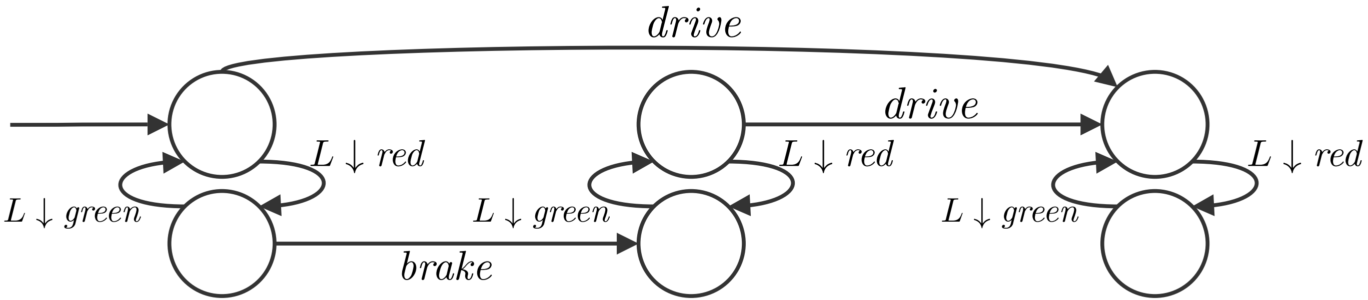


Transition System from Algebra Expression



$$CAR \stackrel{def}{=} ((L = green) \to drive.\delta) + ((L = red) \to brake.((L = green) \to drive.\delta))$$

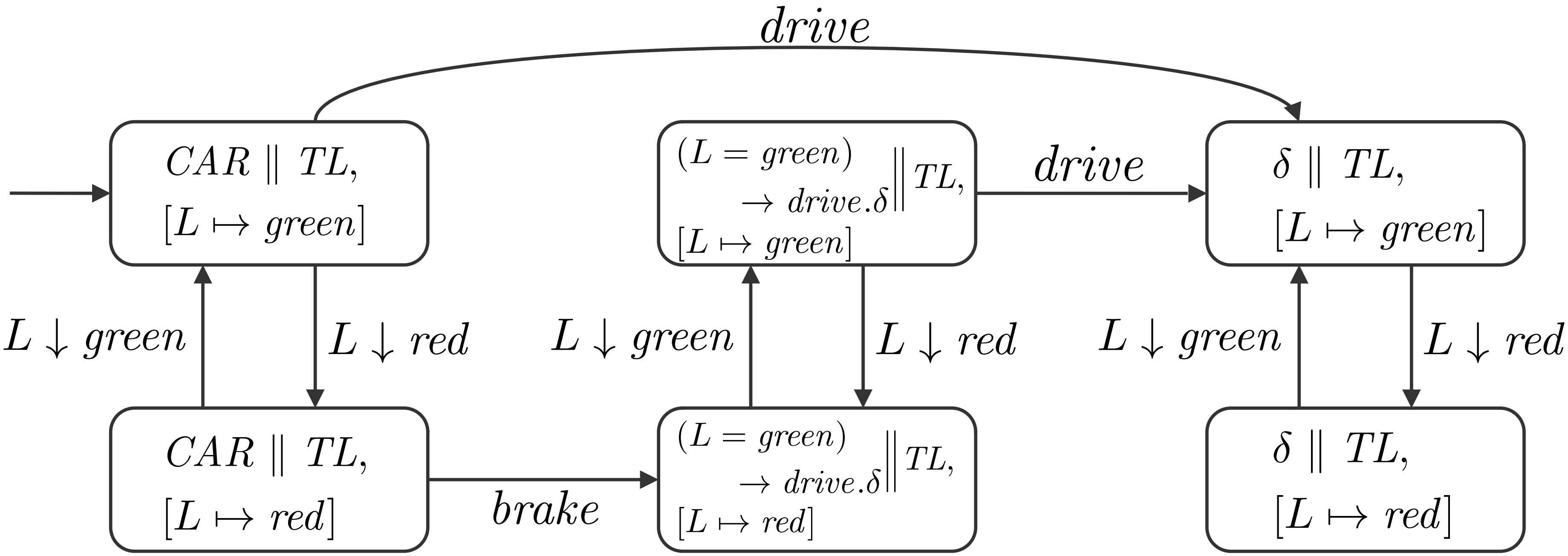
Transition System from Algebra Expression



$$CAR \stackrel{def}{=} ((L = green) \to drive.\delta) + \\ ((L = red) \to brake.((L = green) \to drive.\delta))$$

$$TL \stackrel{def}{=} ((L = green) \to L \downarrow red.TL) + \\ ((L = red) \to L \downarrow green.TL)$$

Transition System from Algebra Expression



TODO / Notes

- **Should I give a formal definition of the process algebra?**

  Continuing with examples and the established intuition seems more accessible to me
  (and also saves time)

Paper: A process algebra with global variables

TECHNISCHE UNIVERSITÄT DARMSTADT

Contributions of the paper

- Introduce global variables to a process algebra — DONE
- Extend Hennessy-Milner logic to reason about new algebra — NEXT
- Notion of bisimilarity
- Translate to existing process algebra

Hennessy-Milner Logic

- Logic to describe properties of transition systems
- e.g. those induced by process algebra expressions

Hennessy-Milner Logic

Property: *“The car will drive, or brake and then drive.”*

$$\langle drive \rangle \top \vee \langle brake \rangle \langle drive \rangle \bot$$

Hennessy-Milner Logic

TECHNISCHE UNIVERSITÄT DARMSTADT

Property: *"The car will drive, or brake and then drive."*

$$\underbrace{\langle drive \rangle \top}_{\text{"there is a transition labeled } drive \text{ after which } \top \text{ holds"}} \vee \langle brake \rangle \langle drive \rangle \bot$$

Hennessy-Milner Logic

TECHNISCHE UNIVERSITÄT DARMSTADT

Property: *“The car will drive, or brake and then drive.”*

$$\langle drive \rangle \top \vee \langle brake \rangle \langle drive \rangle \bot$$

“there is a transition labeled $drive$ after which $\top$ holds”

“there is a transition $brake$ after which there is a transition $drive$ after which $\top$ holds”

Hennessy-Milner Logic

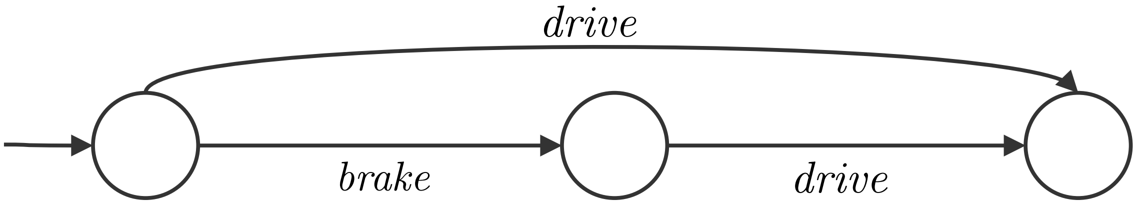
TECHNISCHE UNIVERSITÄT DARMSTADT

Property: *“The car will drive, or brake and then drive.”*

$$\underbrace{\langle drive\rangle\top}\vee\underbrace{\langle brake\rangle\langle drive\rangle\bot}$$

“there is a transition labeled *drive* after which ⊤ holds”

“there is a transition *brake* after which there is a transition *drive* after which ⊤ holds”



Syntax / Intuitive Semantics

$\top, \bot$	True/False Constants
$\wedge, \vee, \neg$	Classical Logic Connectives
$\langle T \rangle \varphi$	There is a $t \in T$ transition after which φ holds
$[T]\varphi$	After all $t \in T$ transitions φ holds
$(v = d)$	Variable Check
$(v \downarrow d)\varphi$	Set: Assume v $=$ d, then φ

NEW!!

Formal Semantics

Let S be the states of a labeled transition system (LTS)

$$[\![\cdot]\!] : HML \to S$$

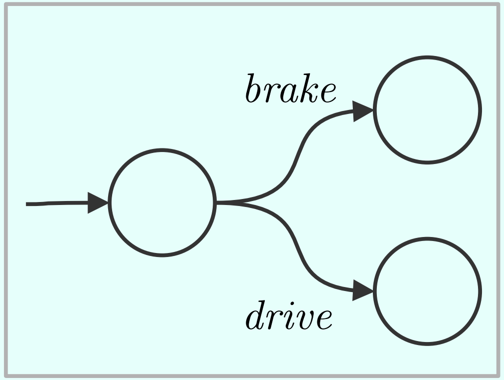
gives for a formula φ the states $[\![\varphi]\!] \subseteq S$ that fulfill it.

Semantics Example

$$\llbracket \langle brake \rangle \top \rrbracket = \{ s \in S \mid \exists s \xrightarrow{brake} s' \wedge s' \in \llbracket \top \rrbracket = S \}$$

Semantics Example

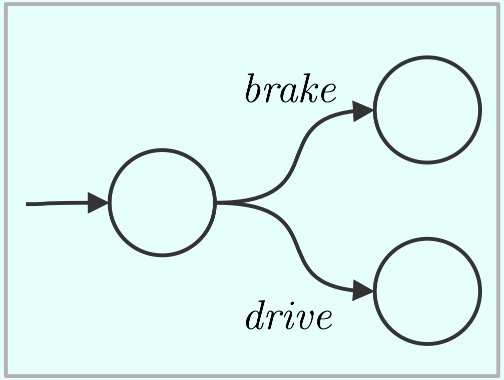
$$[\![\langle brake\rangle\top]\!] = \{s \in S \mid \exists s \xrightarrow{brake} s' \land s' \in [\![\top]\!] = S\}$$



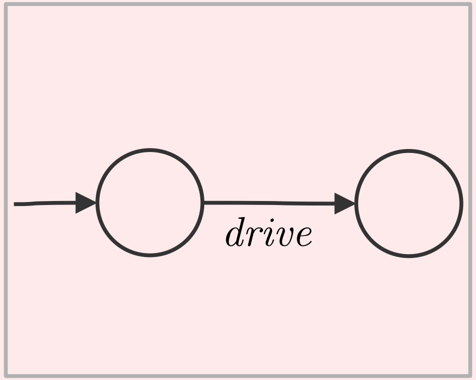
$$\in [\![\langle brake\rangle\top]\!]$$

Semantics Example

$$[\![\langle brake \rangle \top]\!] = \{ s \in S \mid \exists s \xrightarrow{brake} s' \wedge s' \in [\![\top]\!] = S \}$$



$\in [\![\langle brake \rangle \top]\!]$



$\notin [\![\langle brake \rangle \top]\!]$

Semantics Extensions

- Due extensions, semantics **no longer work on all LTS**
- E.g. check formula *(v = d)* must inspect a **variable store**

$\Longrightarrow$ States now must have this form:

$$\langle P, V \rangle$$

Process Algebra Expression

Variable Store / Valuation

$$V(x) = d$$

Check & Set Semantics

$$\llbracket (v = d) \rrbracket = \{ \langle P, V \rangle \in S \mid V(v) = d \}$$

“All states where the valuation for *v* is *d*”

Check & Set Semantics

$$[\![(v = d)]\!] = \{\langle P, V\rangle \in S \mid V(v) = d\}$$

“All states where the valuation for *v* is *d*”

$$[\![(v \downarrow d)\varphi]\!] = \{\langle P, V\rangle \in S \mid \langle P, V[v \mapsto d]\rangle \in [\![\varphi]\!]\}$$

“All states where φ holds if *v* would evaluate to *d*”

TODO / Note

- Maybe a slide should be added which illustrates where a set operator formula is useful?

Paper: A process algebra with global variables

TECHNISCHE UNIVERSITÄT DARMSTADT

Contributions of the paper

- Introduce global variables to a process algebra ✓ DONE
- Extend Hennessy-Milner logic to reason about new algebra ✓ DONE
- Notion of bisimilarity ← NEXT
- Translate to existing process algebra

Bisimilarity

TECHNISCHE UNIVERSITÄT DARMSTADT

- A **bisimulation** is an important relation for any process algebra
- It captures a **form of equivalence** of the transition systems:

*Systems are **bisimilar**,*
*iff they **behave** in **the same** way*

Strong Bisimilarity

- Often, **strong bisimilarity** applies to process algebras
- States *s* and *t* are strongly bisimilar iff
  - *t* can perform all transitions that *s* can perform and vice versa
  - the same applies to the resulting states

$$\text{f. a. } s \xrightarrow{\lambda} s', \text{ exists } t' \text{ s.t.}$$
$$t \xrightarrow{\lambda} t' \text{ and } s' \mathcal{R}\, t'$$
$$\text{and vice-versa}$$

Strong Bisimilarity is No Congruence of New Algebra

- From a congruence relation we expect equivalent systems to combine equivalently:

$$P \underset{strong}{\sim} Q \implies P \parallel R \underset{strong}{\sim} Q \parallel R$$

Strong Bisimilarity is No Congruence of New Algebra

- From a congruence relation we expect equivalent systems to combine equivalently:

$$P \underset{strong}{\sim} Q \implies P \parallel R \underset{strong}{\sim} Q \parallel R$$

- But this **fails** for the PA with Global Variables!

Strong Bisimilarity is No Congruence of New Algebra

- *Why does it fail?*
- Third-party processes may **interfere** with the **memory** of a process!

$$P \overset{def}{=} (L = green) \to drive.\delta \qquad\qquad Q \overset{def}{=} drive.\delta$$

$$P \underset{strong}{\sim} Q$$

...since P has a single *drive* transition and Q has a single *drive* transition

Strong Bisimilarity is No Congruence of New Algebra

TECHNISCHE UNIVERSITÄT DARMSTADT

Let $R \stackrel{def}{=} L \downarrow red.\delta$

race condition with L variable

Strong Bisimilarity is No Congruence of New Algebra

TECHNISCHE UNIVERSITÄT DARMSTADT

Let $R \overset{def}{=} L \downarrow red.\delta$

race condition with L variable

$$Q \parallel R \xrightarrow{L\downarrow red} Q \parallel \delta \xrightarrow{drive} \delta \parallel \delta$$

(valuation omitted)

Strong Bisimilarity is No Congruence of New Algebra

Let $R \stackrel{def}{=} L \downarrow red.\delta$

race condition
with L variable

$$Q \parallel R \xrightarrow{L\downarrow red} Q \parallel \delta \xrightarrow{drive} \delta \parallel \delta$$

(valuation omitted)

$$P \parallel R \xrightarrow{L\downarrow red} P \parallel \delta \not\longrightarrow$$

$L \neq green$

Strong Bisimilarity is No Congruence of New Algebra

TECHNISCHE UNIVERSITÄT DARMSTADT

Let $R \stackrel{def}{=} L \downarrow red.\delta$

race condition with L variable

$$Q \parallel R \xrightarrow{L\downarrow red} Q \parallel \delta \xrightarrow{drive} \delta \parallel \delta$$

(valuation omitted)

$$P \parallel R \xrightarrow{L\downarrow red} P \parallel \delta \not\longrightarrow$$

$L \neq green$

Thus $P \parallel R \underset{strong}{\not\sim} Q \parallel R$!

Stateless Bisimulation

- We want an alternative congruence relation!

  $\longrightarrow$ **Stateless Bisimulation!**

- PA **expressions** are *stateless bisimilar* iff they behave the same **across all valuations**

$$\text{f. a. valuations } V, V', \text{ if } \langle P, V\rangle \xrightarrow{\lambda} \langle P', V'\rangle,$$
$$\text{then exists } Q' \text{ s.t.}$$
$$\langle Q, V\rangle \xrightarrow{\lambda} \langle Q', V'\rangle \text{ and } \langle P', V'\rangle \; \mathcal{R}_{sl} \; \langle Q', V'\rangle$$
$$\text{and vice-versa}$$

Stateless Bisimulation

Bisimilarity no longer determined just by PA expression

- **Initial valuation**
- and **environment**

must be taken into account!

Bisimulation & Hennessy-Milner Logic

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Standard HML** is strong enough to **differentiate non-strongly bisimilar** systems

Bisimulation & Hennessy-Milner Logic

- **Standard HML** is strong enough to **differentiate non-strongly bisimilar** systems
- M.S. Bouwman, et al. show the **extended HML** is strong enough to **differentiate non-stateless bisimilar** expressions:

$$P \underset{sl}{\sim} Q$$

$$\text{iff}$$

$$\forall V, \varphi(\langle P, V \rangle \in [\![\varphi]\!] \iff \langle Q, V \rangle \in [\![\varphi]\!])$$

Proof Details

- Proof omitted for brevity.
- Some important details:
  - Provable only when transition labels yield only **finitely many successors**
    (image-finiteness)
  - The **set operator** ↓ is the **crucial** part of the logic extension:

    Allows to **reason about any possible valuation** by setting any set of values

State-based Bisimulation

- another notion of bisimulation that takes a **fixed initial valuation** into account:

$$\langle P, V_1 \rangle \; \mathcal{R}_{sb} \; \langle Q, V_2 \rangle \; \textit{iff}$$

- $V_1 = V_2$
- if $\langle P, V_1 \rangle \xrightarrow{\lambda} \langle P', V' \rangle$, then $\langle Q, V_2 \rangle \xrightarrow{\lambda} \langle Q', V' \rangle$
- $\langle P', V' \rangle \; \mathcal{R}_{sb} \; \langle Q', V' \rangle$

State-based Bisimulation

- again, non-state-based bisimilar states are **differentiated by the extended logic**
  - though set operator not required here
- like strong bisimulation, this is **not a congruence** relation.

TODO / Notes

- Should I give a concrete use of bisimulation in a concrete analysis of a PA expression?
- I don’t go into detail on the proofs. I figured that would be too technical for a presentation. Is this decision ok?
- Open Question: The logic can distinguish non-stateless-bisimilar, image-finite expressions. Can it actually distinguish non image-finite expressions?

Paper: A process algebra with global variables

Contributions of the paper

- Introduce global variables to a process algebra — DONE
- Extend Hennessy-Milner logic to reason about new algebra — DONE
- Notion of bisimilarity — DONE
- Translate to existing process algebra ← NEXT

Translation to mCRL2

- the PA with global variables can be translated into a PA with just message passing (*mCRL2*)
- Why is this interesting? What does this mean?
  - global variables implementable as **syntactic sugar** into almost any PA
  - all **tooling** of existing PA usable (simulators, verifiers…)

What kind of Translation?

- bidirectional translation
- state-based bisimilarity translated into strong bisimilarity

$$\langle P, V_1 \rangle \underset{sb}{\sim} \langle Q, V_2 \rangle$$

$$\Longleftrightarrow$$

$$\Psi(P, V_1) \underset{strong}{\sim} \Psi(Q, V_2)$$

translation of states

- formulae are preserved

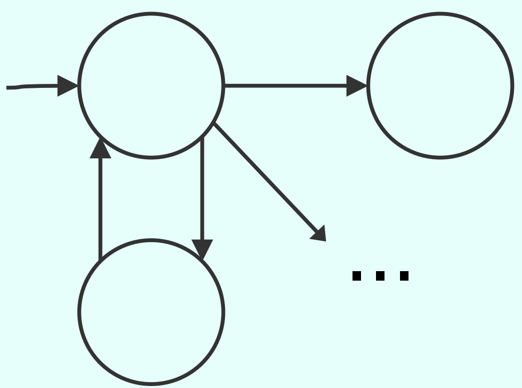
$$\langle P, V \rangle \in [\![\varphi]\!] \iff \Psi(P, V) \in [\![\theta(\varphi)]\!]$$

translation of formulae

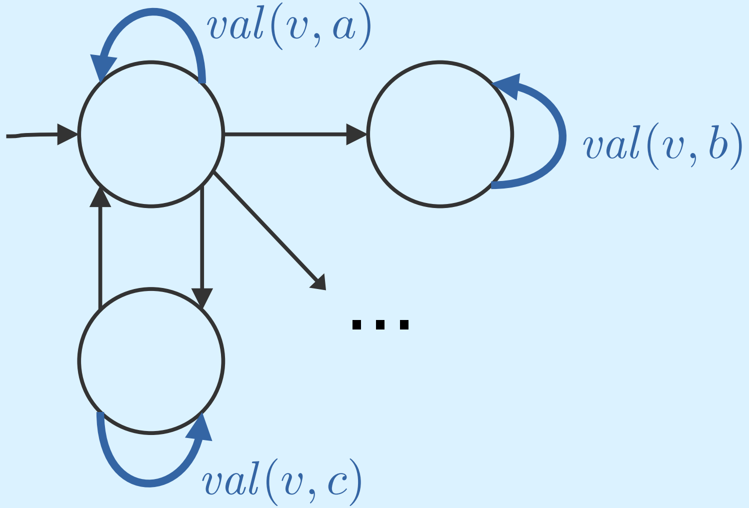
What kind of Translation?

- translated LTS *almost* isomorphic:

Original

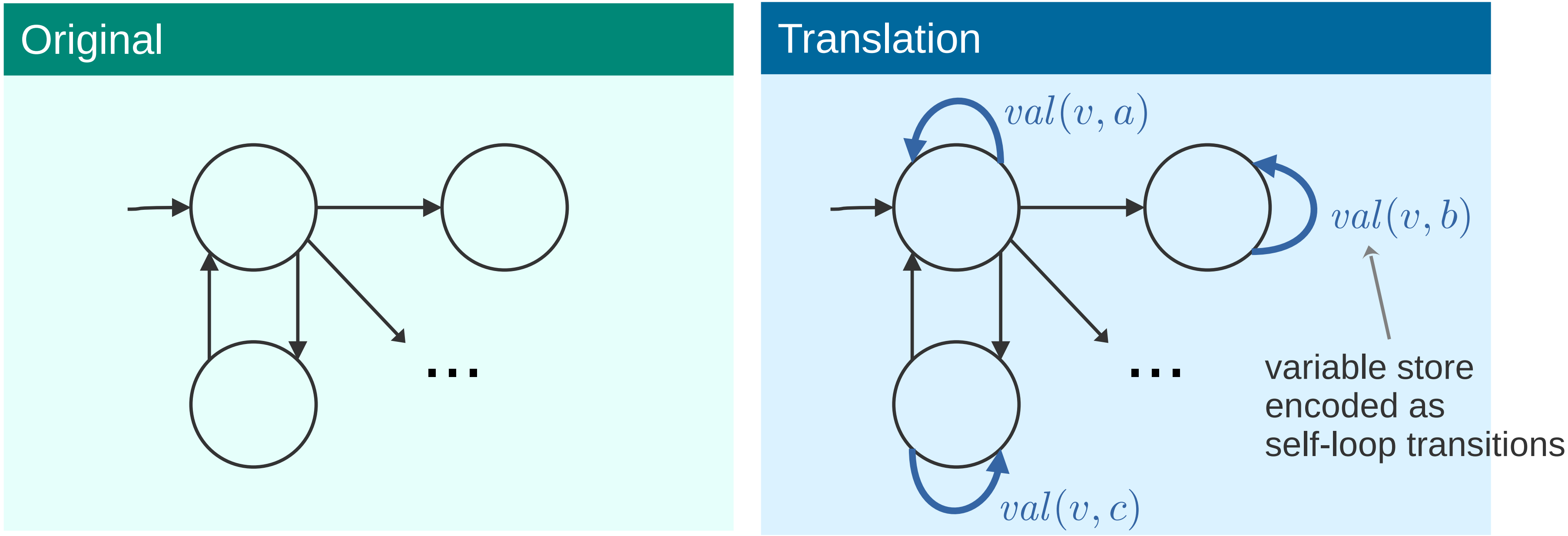


Translation



What kind of Translation?

- translated LTS *almost* isomorphic:



Translation: Conditionals

Simplification!

Conditionals $(v = d) \rightarrow \ldots$

are translated to **lookup messages** $check(v, d)$

to a **central memory management process**

Translation: Conditionals

Simplification!

Conditionals $(v = d) \to \ldots$

are translated to **lookup messages** $check(v, d)$

to a **central memory management process**

Original

Translation: Conditionals

Simplification!

Conditionals $(v = d) \rightarrow \ldots$

are translated to **lookup messages** $check(v, d)$

to a **central memory management process**

Original

$$\rightarrow (L = green) \rightarrow drive.\delta, [L \mapsto green] \xrightarrow{drive} \delta, [L \mapsto green]$$

Translation

$$\rightarrow check(L, green) | drive.\delta \xrightarrow{check(L, green) | drive} \delta$$

Translation: Assignments

Simplification!

Assignments $v \downarrow d$

are now **messages carrying new value** $assign(v, d)$

to the **central memory management process**

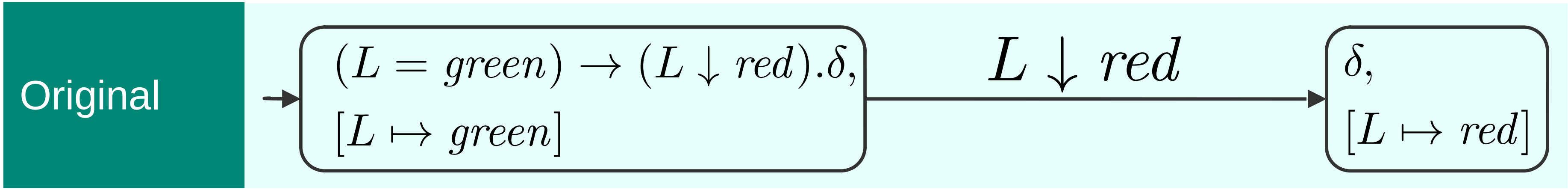
Translation: Assignments

Simplification!

Assignments $v \downarrow d$

are now **messages carrying new value** $assign(v, d)$

to the **central memory management process**



Translation: Assignments

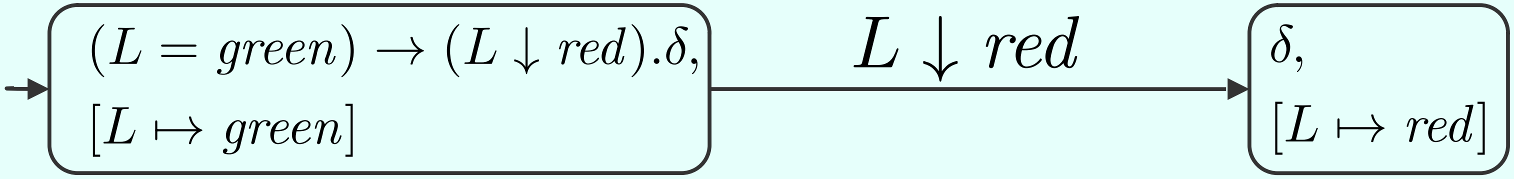
Simplification!

Assignments $v \downarrow d$

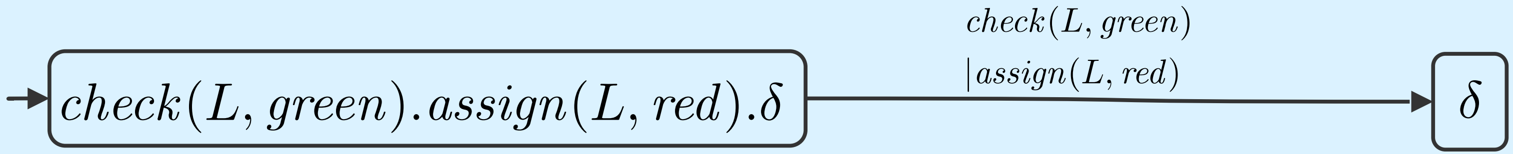
are now **messages carrying new value** $assign(v, d)$

to the **central memory management process**

Original



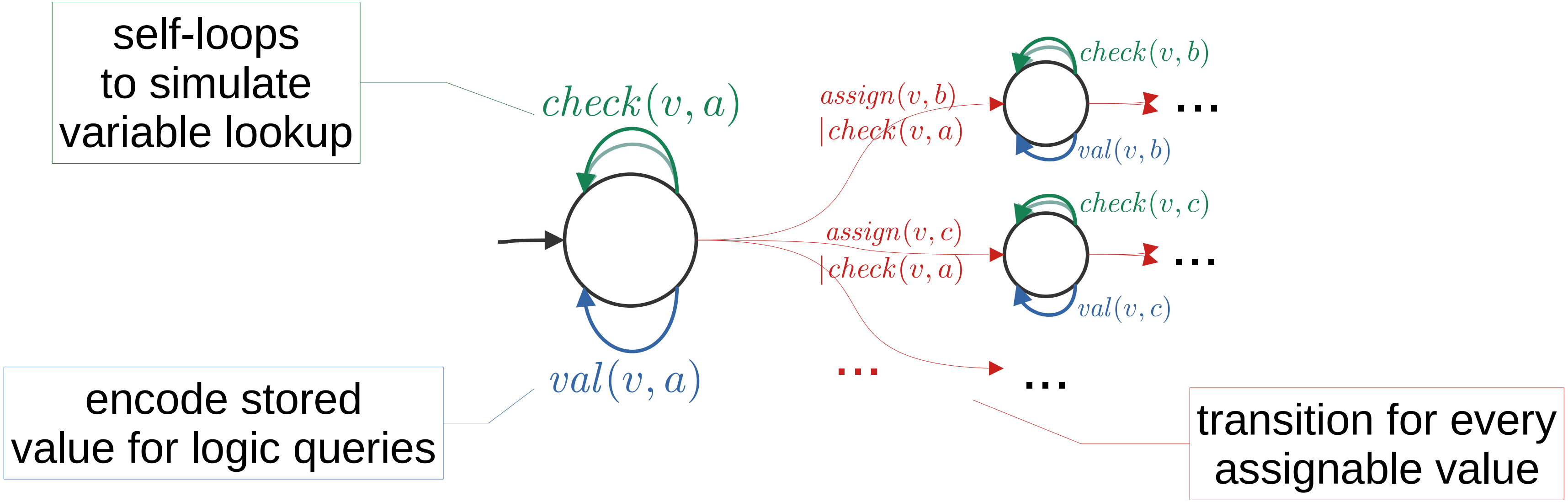
Translation



Translation: Global Variable Service

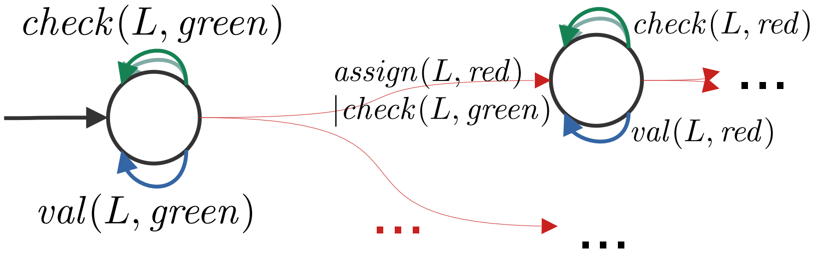
Simplification!

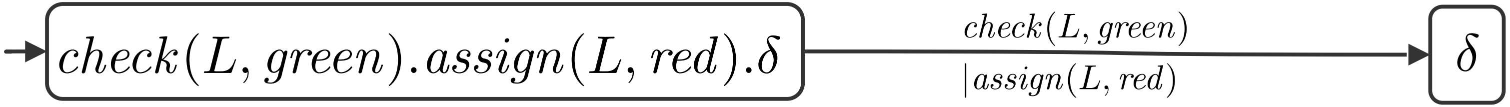
- A process "*Globs*" is added, which **manages global variables**



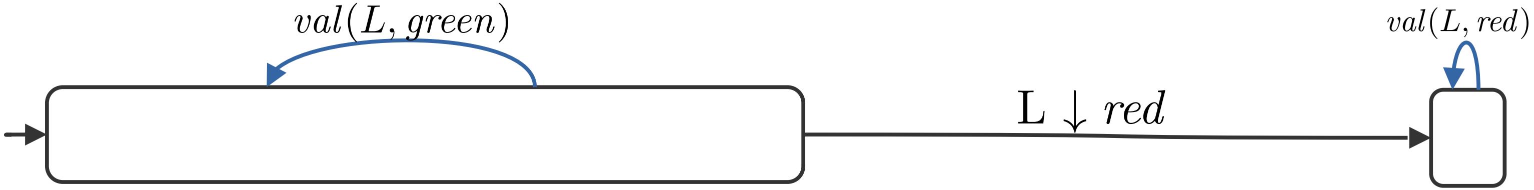
Translation: Combining Parts

Simplification!





Parallel Composition & Synchronization



Translation: Formulae

TECHNISCHE UNIVERSITÄT DARMSTADT

- Set operators will not be translated
- Check operators are translated to a query for the $val(v,d)$ transitions:

$$\theta((v = d)) = \langle(val(v,d))\rangle\top$$

- everything else unchanged

Translation: Correctness

- Proof of mentioned properties omitted for brevity
- **Essentially**, it is shown that the LTS are the same (except for the self-loops)

Paper: A process algebra with global variables

Contributions of the paper

- Introduce global variables to a process algebra — DONE
- Extend Hennessy-Milner logic to reason about new algebra — DONE
- Notion of bisimilarity — DONE
- Translate to existing process algebra — DONE

TODO / Notes

- I extremely simplified the translation
- I do not introduce mCRL2 since I think the audience does not gain anything from being introduced to another algebra
- I explain only using examples
- **I am unsure, if it is alright to simplify such a large part of the paper to this degree…?**

Future Work

- M.S. Bouwman et. al. plan the following continuations of their work:
  - actually **integrating** global variables **into mCRL2**
  - **more complex value checks** than equality
  - making global variables **scoped**

TODO / Notes

- The following “Opinion” section is still very much **WIP**

Opinion

Opinion

+ I like that the authors took the time to discuss most of the important parts of proofs in detail

Opinion

- the translation heavily relies on specific features of mCRL2. (synchronizing multiple messages at once etc.)
- *Open*:
  - Does it translate just as easily to other PAs?
  - Will the LTS still be isomorphic?
- This is not discussed

Opinion

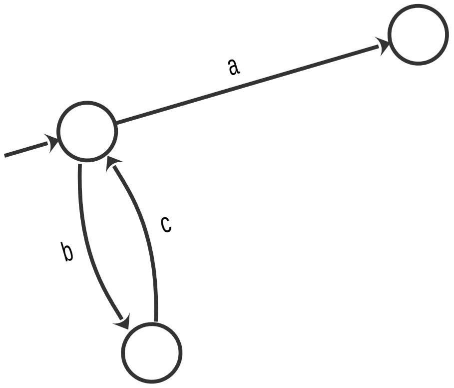
- Real shared memory is accessed truly concurrently (no interleaving)
- This makes data races even harder to debug
- How true concurrency could be modeled is not discussed.

Opinion

TECHNISCHE UNIVERSITÄT DARMSTADT

- Only the presented toy example is introduced in the paper
- No discussion which real systems would actually benefit from being analyzed with this process algebra

Thank you for listening!



$$P \stackrel{\text{def}}{=} (a \parallel b) + (c \parallel d).P$$

Any questions?

$$\varphi \wedge \langle T \rangle \psi$$

Manually ensuring compliance with protocol is…

- susceptible to human error
- Labor-intensive
- Impossible for large systems

Manually ensuring compliance with protocol is…

- susceptible to human error
- Labor-intensive
- Impossible for large systems

Manually ensuring compliance with protocol is…