Abstracting Failures Away From Stateful Dataflow Systems

Aleksey Veresov

Supervised by Philipp Haller and Jonas Spenger

Overview

Introduction → Model → Definitions → Proof → Conclusion

- The first small-step operational semantics of *Asynchronous Barrier Snapshotting*
- A novel and general definition of *failure transparency*
- A proof that ABS provides failure transparency
- A proof technique is devised and presented
- The model, the definitions and the theorems are mechanized in Coq*

Related Work

Scalable and Reliable Data Stream Processing, Paris Carbone, PhD dissertation, 2018
Durable functions: semantics for stateful serverless, Sebastian Burckhardt et al, 2021
Fundamentals of Fault-Tolerant Distributed Computing in Asynchronous Environments, Felix C. Gärtner, 1999

* github.com/aversey/abscoq

Failures in Distributed Systems

Failures are hard to handle in Distributed Systems

- **Google's** apps crashed for around *an hour* in 2020*
- **Facebook** services were down for ~*6 hours* in 2021**
- **Cloudflare** had outage for ~*6 hours* in 2020*** due to a **Byzantine** failure

focus on **crash** failures

Formal Methods are the preferred solution

- **Amazon** applies them to **AWS******
- **Microsoft** uses them on **Azure******
- Other leading companies show interest in them

* nytimes.com/2020/12/14/business/google-down-worldwide.html
** en.wikipedia.org/wiki/2021_Facebook_outage
*** blog.cloudflare.com/a-byzantine-failure-in-the-real-world
**** github.com/ligurio/practical-fm

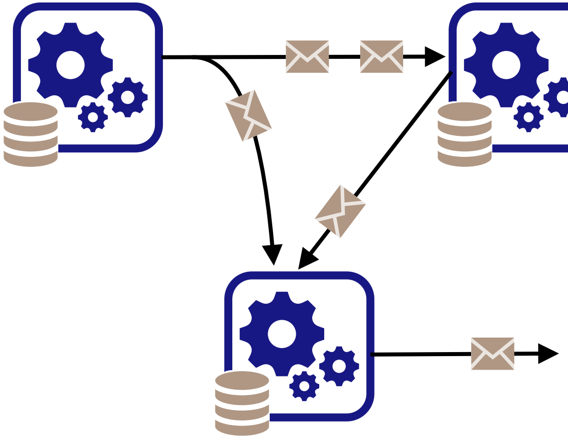
Failures in Stateful Dataflow Systems

Apache Flink

Google Dataflow

Azure Event Hubs

emerging systems
Portals*, **Styx****, etc.



No need to handle failures manually
⇒ widespread use, e.g., in Uber, ByteDance

Flink serves billions of events per second

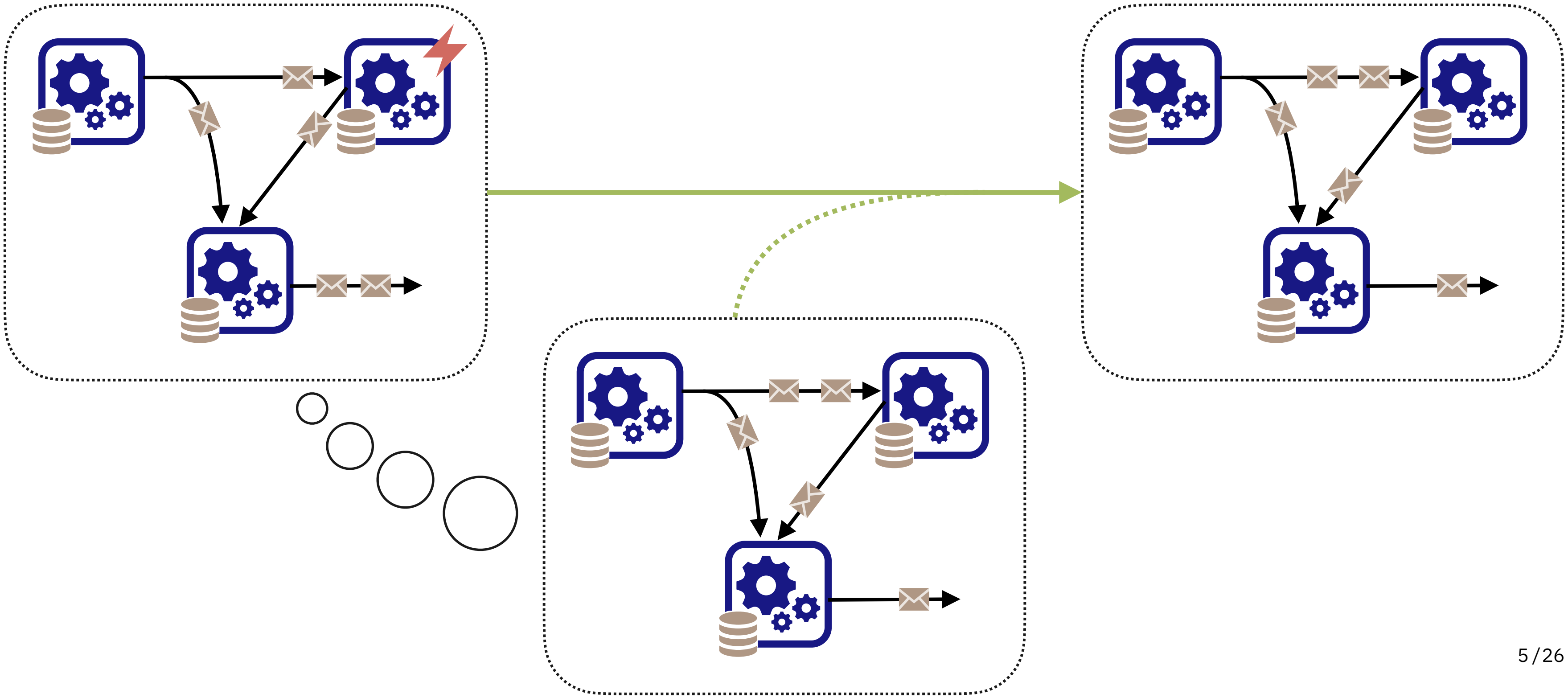
No general formal definition applicable to SOS of

Failure Transparency

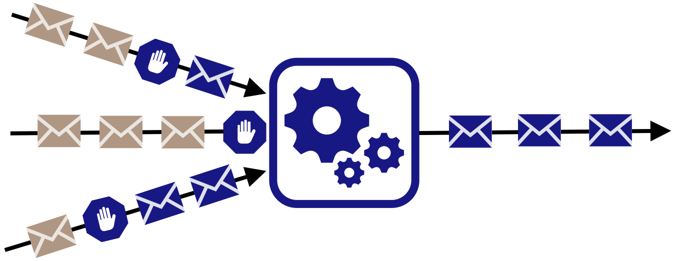
No previous work is applicable to any of the mentioned systems
although the systems are claimed to provide it!

* Spenger et al. *Portals: An Extension of Dataflow Streaming for Stateful Serverless.* Onward! 2022.
** `github.com/delftdata/styx`

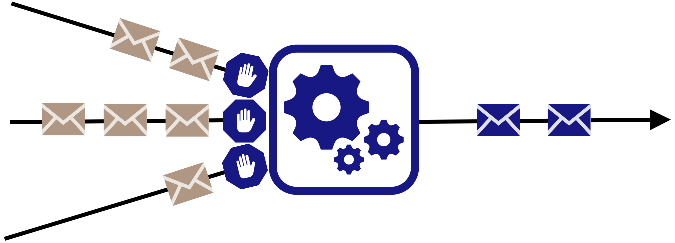
Rollback Recovery



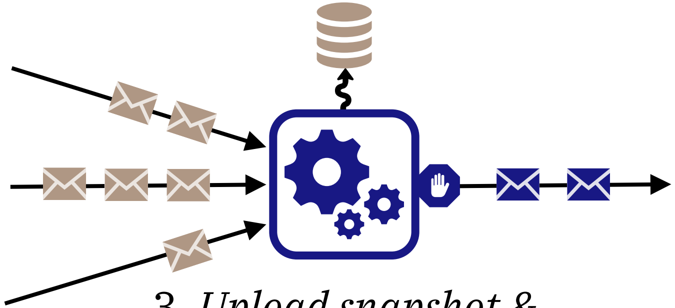
Asynchronous Barrier Snapshotting



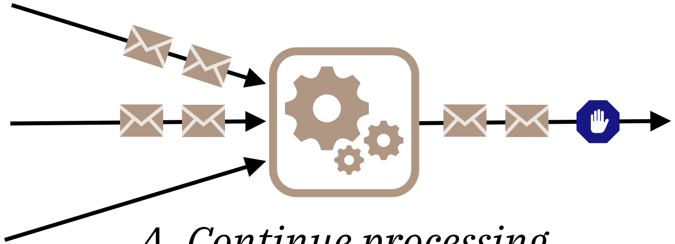
1. *Process up to a barrier*



2. *Barriers are aligned*



3. *Upload snapshot & propagate barrier*



4. *Continue processing*

Small-Step Operational Semantics

$$\textsc{Add}\ \frac{a \in \mathbb{Z} \qquad b \in \mathbb{Z} \qquad c = a + b}{\mathsf{add}(a, b) \to c}$$

$$\textsc{AddL}\ \frac{a \to a'}{\mathsf{add}(a, b) \to \mathsf{add}(a', b)} \qquad\qquad \textsc{AddR}\ \frac{b \to b'}{\mathsf{add}(a, b) \to \mathsf{add}(a, b')}$$

$$\mathsf{add}(\mathsf{add}(1,2), \mathsf{add}(3,4)) \xrightarrow{\textsc{AddL}} \mathsf{add}(3, \mathsf{add}(3,4)) \xrightarrow{\textsc{AddR}} \mathsf{add}(3,7) \xrightarrow{\textsc{Add}} 10$$

$$\mathsf{add}(\mathsf{add}(1,2), \mathsf{add}(3,4)) \xrightarrow{\textsc{AddR}} \mathsf{add}(\mathsf{add}(1,2), 7) \xrightarrow{\textsc{AddL}} \mathsf{add}(3,7) \xrightarrow{\textsc{Add}} 10$$

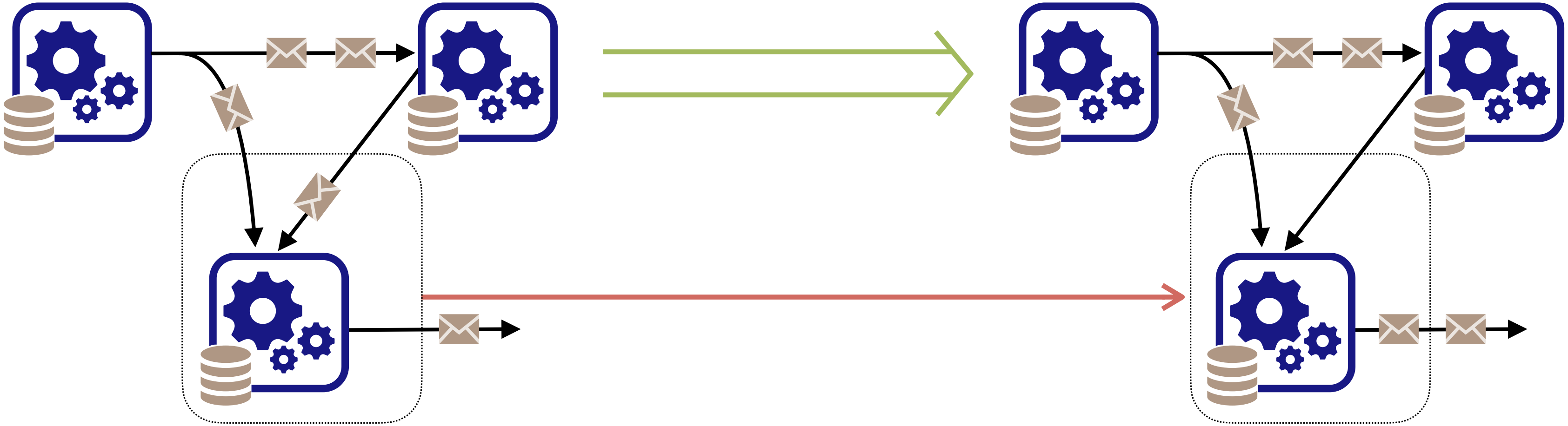
Stateful Dataflow Model

Stateful Dataflow Model

small-step operational semantics of ABS-based systems

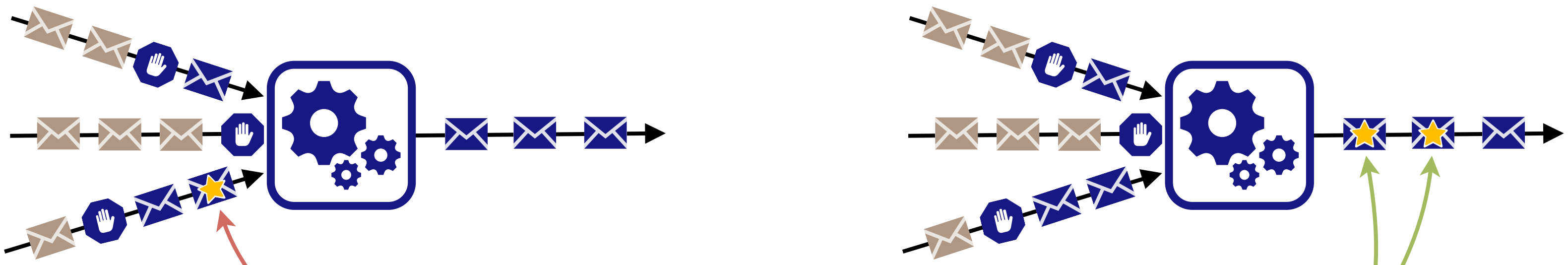
	3 rules describe a failure-free system
+	**2** rules are related to failures
+	**2** rules are auxiliary

Stateful Dataflow Model, S-Step



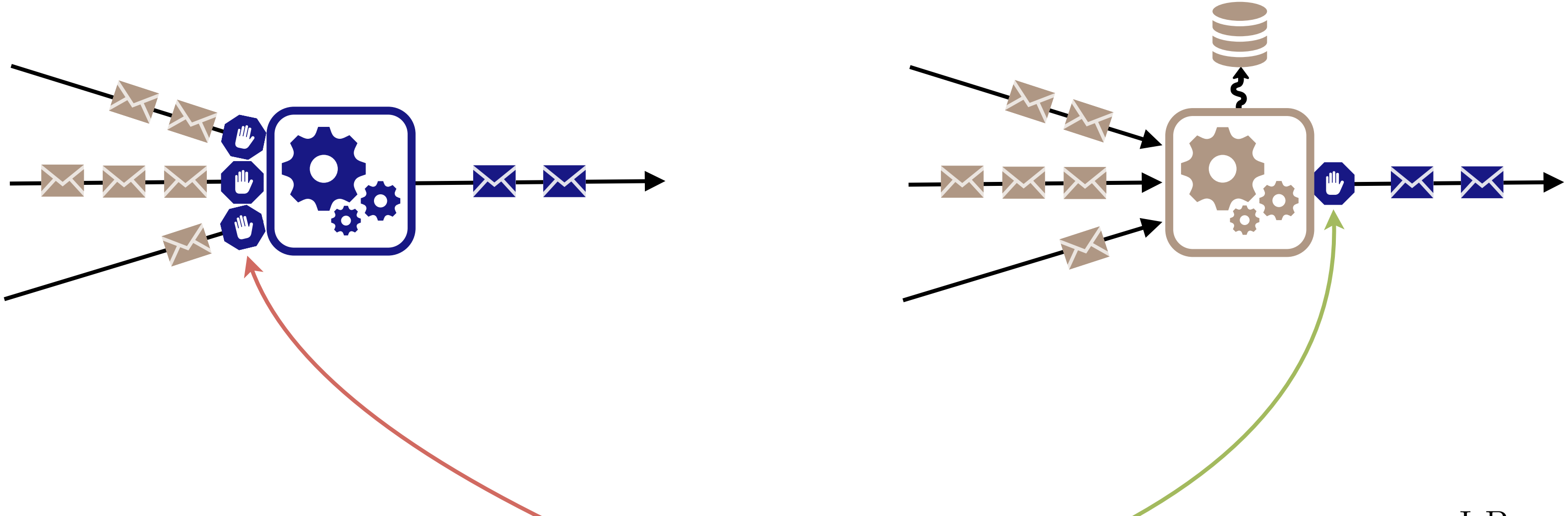
$$\frac{\Pi_p \Vdash \Sigma_p \xrightarrow{X} \Sigma'_p \qquad X(N_p,\, M) = (N'_p,\, M')}{\langle\, \Pi,\, \Sigma,\, N,\, M,\, D \,\rangle \xRightarrow[p]{N_p, X} \langle\, \Pi,\, \Sigma\big[p \mapsto \Sigma'_p\big],\, N\big[p \mapsto N'_p\big],\, M',\, D \,\rangle}\ \textsc{S-Step}$$

Stateful Dataflow Model, I-Event



$$\frac{f(v,w) = v', [W'_i]_i^n}{\mathtt{TK}\langle f, S, o\rangle \Vdash \langle a, \langle e, v\rangle\rangle \xrightarrow{[-S_j\langle e, \mathtt{EV}\langle w\rangle\rangle]:[+o\langle e, \mathtt{EV}\langle W'_i\rangle\rangle]_i^n} \langle a, \langle e, v'\rangle\rangle}\ \textrm{I-E}\textsc{vent}$$

Stateful Dataflow Model, I-Border

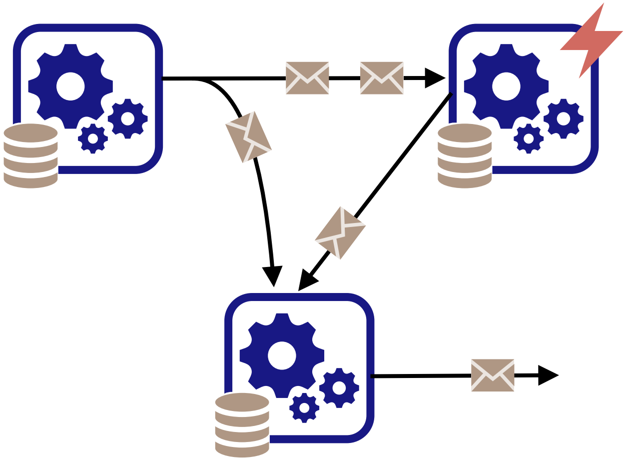


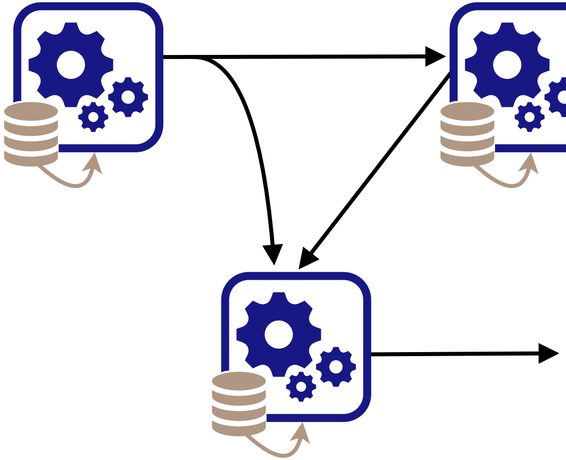
$$\frac{}{\mathtt{TK}\langle f, [S_i]_i^n, o\rangle \Vdash \langle a, \langle e, v\rangle\rangle \xrightarrow{[-S_i\langle e, \mathbf{BD}\rangle]_i^n : [+o\langle e, \mathbf{BD}\rangle]} \langle a[e \mapsto v], \langle e+1, v\rangle\rangle}\ \textsc{I-Border}$$

Stateful Dataflow Model, F-Fail

$$\frac{}{\texttt{TK}\langle\, f, S, o \,\rangle \;\Vdash\; \langle\, a,\, \sigma_{\mathrm{V}} \,\rangle \rightarrow \langle\, a, \mathbf{\texttt{fl}} \,\rangle}\ \textsc{F-Fail}$$

Stateful Dataflow Model, F-Recover

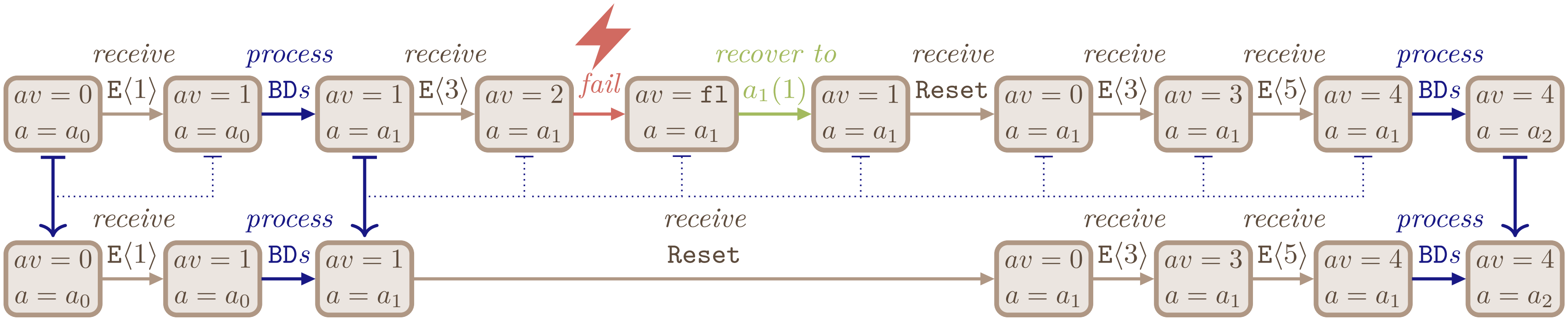




$$\frac{\langle\, a, \texttt{fl} \,\rangle \in \Sigma}{\langle\, \Pi,\ \Sigma,\ N,\ M,\ M_0 \,\rangle \Rightarrow \mathsf{lcs}(\langle\, \Pi,\ \Sigma,\ N,\ M,\ M_0 \,\rangle)}\ \textsc{F-Recover}$$

Failure Transparency Definition

Failure Transparency, on example



Execution in R

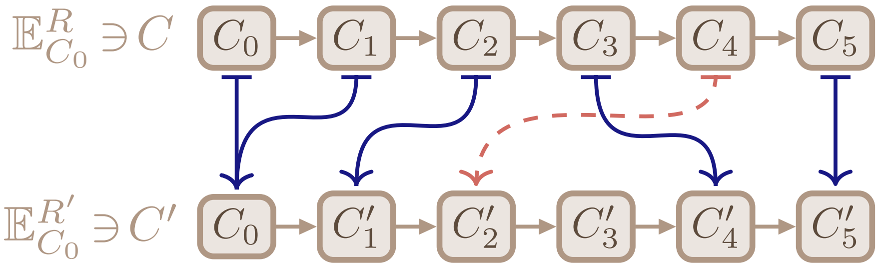
A sequence of configurations $\left[C_i\right]_i^n$ such that $\forall i < n.\ R \vdash C_{i-1} \Rightarrow C_i$.

Observational Explanation $\quad C \;{}^{O}\!\rightleftharpoons^{O'} C'$

Relates two executions C and C’ according to some observability functions O and O’

Intuitively: *all observations reachable in original can be reached in the explanation*

$$\forall m < n.\ \exists m' < n'.\ O(C_m) = O'(C'_{m'})$$



Observational Explainability $R \;{}^{O}\!\overset{T}{\rightleftharpoons}^{O'}\; R'$

Relates two systems R and R' according to observabilities O and O' and translator T

Intuitively: *for each execution of a translation, its source should observationally explain it*

$$\forall\, c' \in \mathrm{dom}(T).\ \forall c.\ c' T c \implies \forall C \in \mathbb{E}_c^R.\ \exists\, C' \in \mathbb{E}_{c'}^{R'}.\ C \;{}^{O}\!\rightleftharpoons^{O'}\; C'$$

Failure Transparency

Observational explainability of a system by its failure-free part

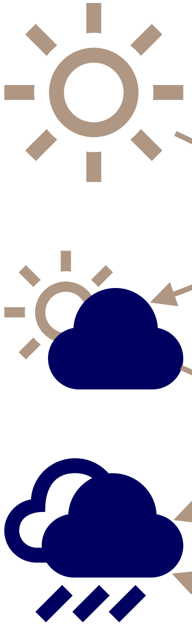
Intuitively: *for each valid (= in K) program,*
no execution with failures (R) produces an observation,
which can not be achieved in its execution without failures (R \ F).

$$R \ {}^{O}\!\xrightleftharpoons{\{(c,c)\ \mid\ c \in K\}}{}^{O} (R \setminus F)$$

It is suitable for a wide range of models in small-step operational semantics!

Composability of Observational Explainability

R'



R

$\hat{R}$

$$R \; {}^{O}\overset{T}{\rightleftharpoons}{}^{O'} \; R'$$

$$\hat{R} \; {}^{\hat{O}}\overset{\hat{T}}{\rightleftharpoons}{}^{\tilde{O}} \; R$$

$$O = O'' \circ \tilde{O}$$

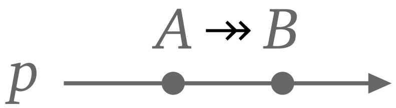
$$\hat{R} \; {}^{O'' \circ \hat{O}}\overset{\hat{T} \circ T}{\rightleftharpoons}{}^{O'} \; R'$$

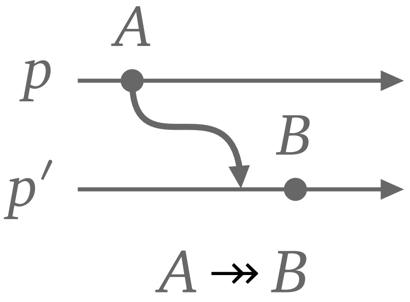
...

down to the real code

Failure Transparency of Stateful Dataflow

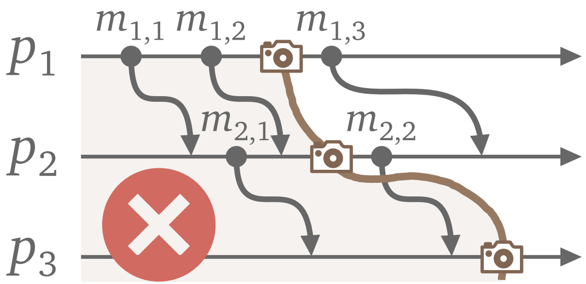
Causality

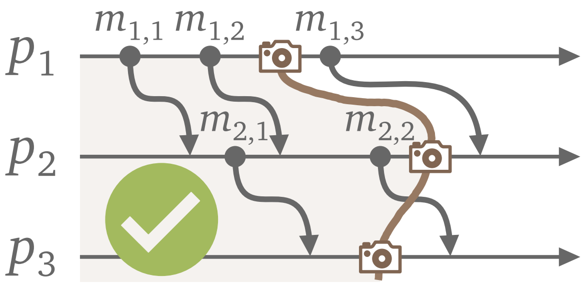


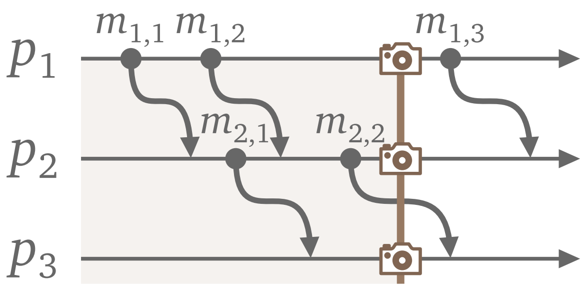


$$\begin{array}{l} A \twoheadrightarrow C \\ C \twoheadrightarrow B \end{array} \implies A \twoheadrightarrow B$$

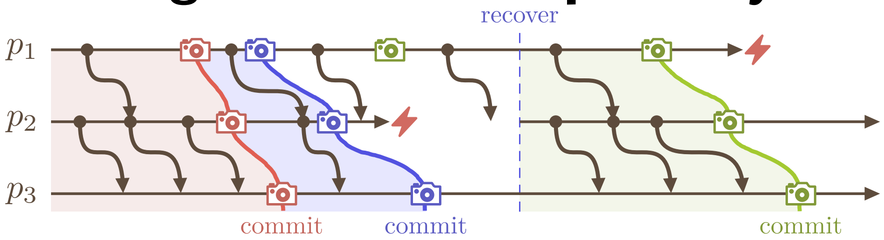
Causal Consistency



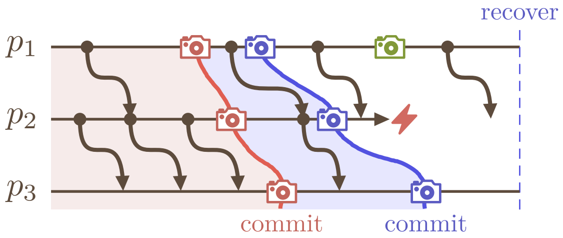




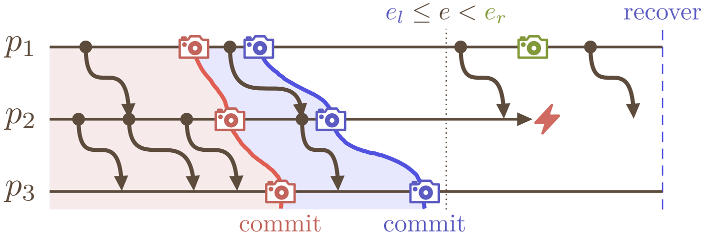
Proving Failure Transparency



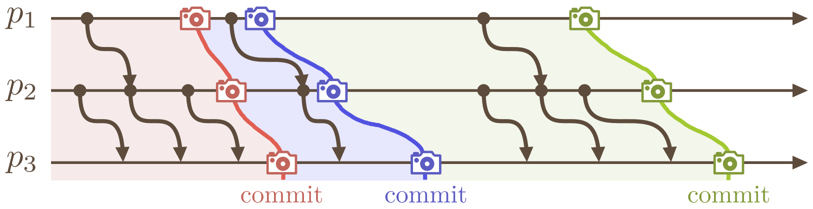
Given: *Original execution*



1. *Separated generation*



2. *Reordered generation*



Construct: *Failure-free observational explanation*

Conclusion

Summary

- Failure Transparency is defined so that it is widely applicable
- The definition enables easier proofs via history manipulation
- The first operational semantics of Asynchronous Barrier Snapshotting is provided
- Asynchronous Barrier Snapshotting provides Failure Transparency

Future Work

- Mechanize the proofs
- Provide lower- and higher-level models of Stateful Dataflow Systems
- Explore end-to-end Failure Transparency of heterogeneous systems