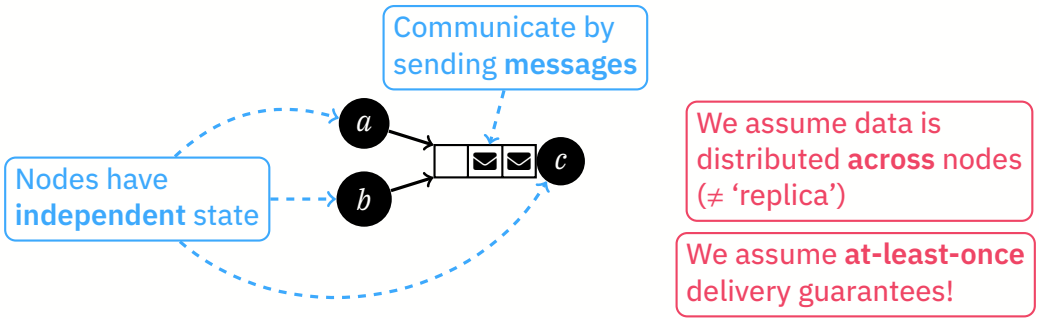
Distributed Consistency Beyond Queries

Tim Baccaert Bas Ketsman

Vrije Universiteit Brussel
Brussels, Belgium

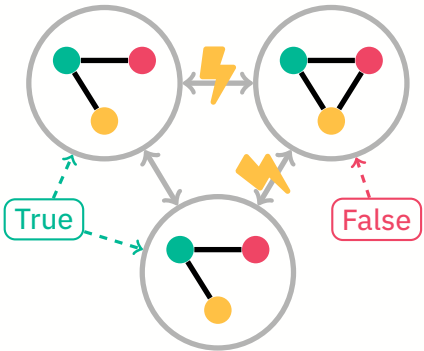
Distributed Setting



Asynchronous Message Passing Systems

Consistency

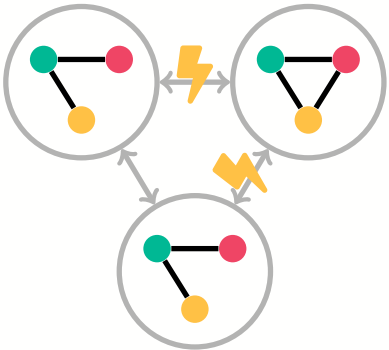
- Are Alice, Bob, and Carol **NOT** mutual friends (*i.e.*, not a triangle)?



Inconsistent!
System is in a ‘split-brain’ state.
If connection restores, then
we have to **retract** our True output!

Consistency

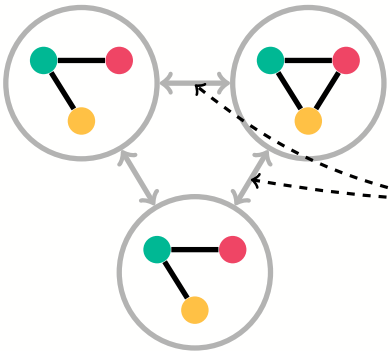
- Are Alice, Bob, and Carol **NOT** mutual friends (*i.e.*, not a triangle)?



Instead: wait for consensus by using a **consensus protocol** (*e.g.*, Paxos, Raft, 2PC, ...)

Consistency

- Are Alice, Bob, and Carol **NOT** mutual friends (*i.e.*, not a triangle)?

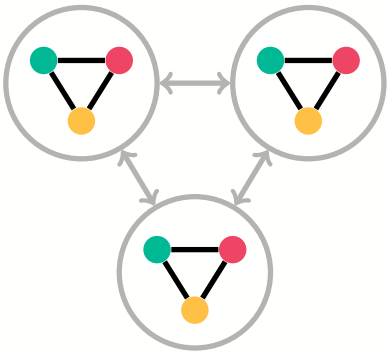


Instead: wait for consensus by using a **consensus protocol** (*e.g.*, Paxos, Raft, 2PC, ...)

Wait until the network partition is gone

Consistency

- Are Alice, Bob, and Carol **NOT** mutual friends (*i.e.*, not a triangle)?



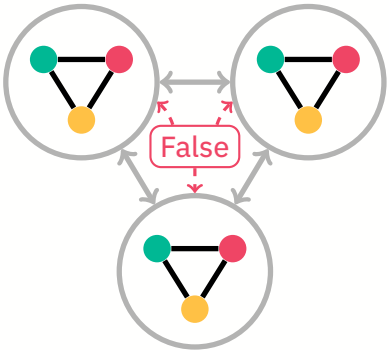
Instead: wait for consensus by using a **consensus protocol** (*e.g.*, Paxos, Raft, 2PC, ...)

Wait until the network partition is gone

Coordinate until state is identical

Consistency

- Are Alice, Bob, and Carol **NOT** mutual friends (*i.e.*, not a triangle)?



Instead: wait for consensus by using a **consensus protocol** (*e.g.*, Paxos, Raft, 2PC, ...)

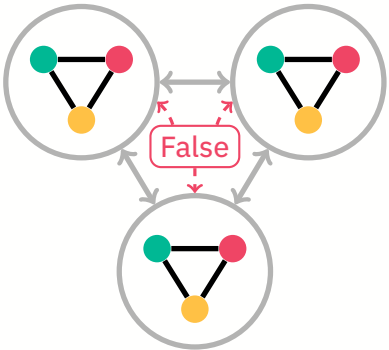
Wait until the network partition is gone

Coordinate until state is identical

Compute the answer

Consistency

- Are Alice, Bob, and Carol **NOT** mutual friends (*i.e.*, not a triangle)?



Instead: wait for consensus by using a **consensus protocol** (*e.g.*, Paxos, Raft, 2PC, ...)

Wait until the network partition is gone

Coordinate until state is identical

Compute the answer

This is slow: can we avoid this?

The **CALM** Theorem

Consistency **A**s **L**ogical **M**onotonicity

A query has a consistent, **coordination-free** distributed implementation if, and only if, it is **monotonic**.

[Ameloot, Neven, Van den Bussche, PODS ’11]

The **CALM** Theorem

Consistency **A**s **L**ogical **M**onotonicity

A query has a consistent, **coordination-free** distributed implementation if, and only if, it is **monotonic**.

[Ameloot, Neven, Van den Bussche, PODS '11]

- Monotonicity: Query Q is **monotone** if $Q(I) \subseteq Q(I \cup J)$ for all instances I, J.
- Example Q: are Alice, Bob, and Carol mutual friends (*i.e.*, a triangle)?

I J $I \cup J$

$Q(I) = \{\}$ (false)
$Q(I \cup J) = \{()\}$ (true)

Staying CALM at PODS

A Brief History

1. CALM Conjecture: coordination-free ⇔ monotone problem

   [Hellerstein, PODS ’10]

2. CALM Theorem: True, for monotone queries

   [Ameloot, Neven, Van den Bussche, PODS ’11]

3. CALM Theorem (ext.): Also true for semi- and disjoint-monotone queries if nodes can verify the **data partitioning strategy** locally.

   [Ameloot, Ketsman, Neven, Zinn, PODS ’14]

Applicability

System Knowledge

- Requirement: nodes in the system under study may access the following
  - **[Ameloot et al. ’11]:** their unique identity (*e.g.*, IP-address, ...)
  - **[Ameloot et al. ’11]:** the unique identities of all nodes in the system
  - **[Ameloot et al. ’14]:** a **local** index (*e.g.*, a hash or range function)

Applicability

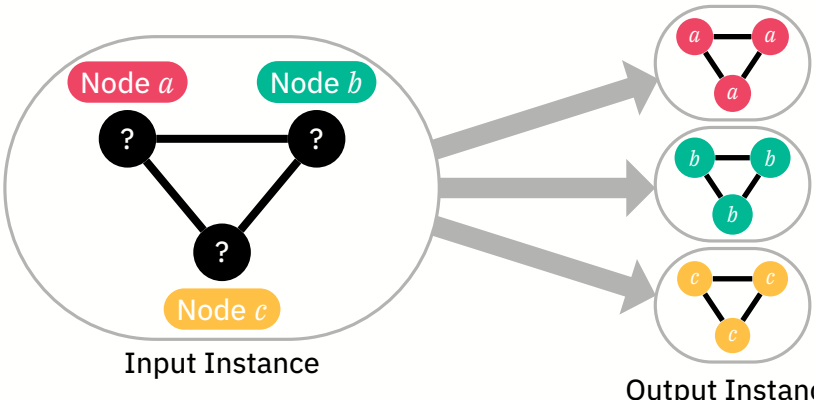
System Knowledge

- Requirement: nodes in the system under study may access the following
  - **[Ameloot et al. ’11]:** their unique identity (*e.g.*, IP-address, ...)
  - **[Ameloot et al. ’11]:** the unique identities of all nodes in the system
  - **[Ameloot et al. ’14]:** a **local** index (*e.g.*, a hash or range function)
- What about other **common properties**?
  - **Leader Node:** a central coordinator node
  - **Order:** an order on active domain elements (*e.g.*, lexical order on bytes)
  - **Distributed Index:** a global index (*e.g.*, what are other nodes responsible for?)
  - ...

Applicability

Program Semantics

- Requirement: problem is expressible as a **deterministic** query
- Some problems are **non-deterministic** or refer to the **location of data**
  - Example: *LEADER-ELECTION* – all nodes agree on **the same** leader



Objectives

1. Model **non-determinism** over instances with facts **tied to specific nodes**
2. **Generalize** the previous system models
3. **Generalize** the **CALM theorem**
4. **Apply** our work to see if $PTIME$ queries over ordered databases are coordination-free [Hellerstein, Alvaro, CACM '20]

Overview

Introduction

Behaviors and Transducers

Computation of Behaviors

Weakly Coordinating Behaviors

Behaviors

Modeling Non-Deterministic Problems over Distributed Instances

Definition

A **distributed behavior** B from a schema σ_{in} to a schema σ_{out} is a (domain-preserving, generic) left-total relation $B \subseteq \mathit{dist}(\sigma_{\text{in}}) \times \mathit{dist}(\sigma_{\text{out}})$

Behaviors

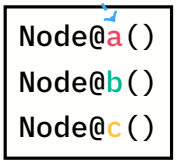
Modeling Non-Deterministic Problems over Distributed Instances

Definition

A **distributed behavior** B from a schema σ_{in} to a schema σ_{out} is a (domain-preserving, generic) left-total relation $B \subseteq dist(\sigma_{\text{in}}) \times dist(\sigma_{\text{out}})$

Example (Leader Election)

Distributed Instance $H \in dist(\sigma_{\text{in}})$: facts annotated with nodes



Input H

Behaviors

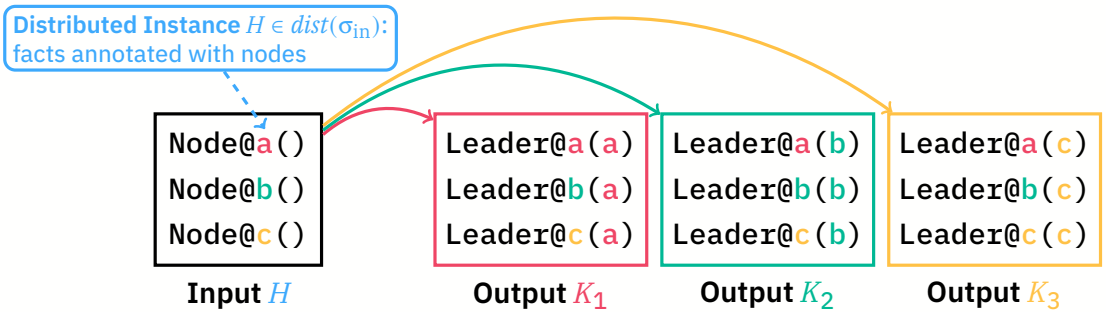
Modeling Non-Deterministic Problems over Distributed Instances

Definition

A **distributed behavior** B from a schema σ_{in} to a schema σ_{out} is a (domain-preserving, generic) left-total relation $B \subseteq \mathit{dist}(\sigma_{\text{in}}) \times \mathit{dist}(\sigma_{\text{out}})$

Example (Leader Election)

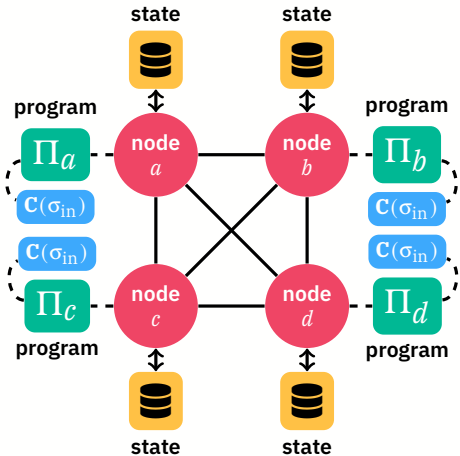
$\{(H, K_1), (H, K_2), (H, K_3)\} \subseteq \mathit{LEADER\text{-}ELECTION}$



Generalized Relational Transducers

Modeling Systems

- Generalizes (Policy-Aware) Relational Transducer Networks [Ameloot et al., '11 & '14]
- each node in $netw(H)$ executes a transducer Π
- Π has access to system relations determined by a constraint $\mathbf{C}$ (notation: $\mathbf{C}$-transducer)
- Semantics in terms of fair runs of transitions between states



Constraints

Modeling System Knowledge

Definition

A **configuration constraint** $\mathbf{C}$ is a **total function** which, for all possible input schemas σ_{in}, maps onto a **behavior** $\mathbf{C}(\sigma_{\text{in}})$ from σ_{in} to σ_{sys}

We can compose constraints (*i.e.*, $\mathbf{C}+\mathbf{C}'$) if the σ_{sys} of their behaviors are disjoint.

Constraints

Modeling System Knowledge

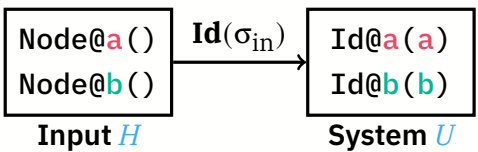
Definition

A **configuration constraint** $\mathbf{C}$ is a **total function** which, for all possible input schemas σ_{in}, maps onto a **behavior** $\mathbf{C}(\sigma_{\text{in}})$ from σ_{in} to σ_{sys}

We can compose constraints (*i.e.*, $\mathbf{C}+\mathbf{C}'$) if the σ_{sys} of their behaviors are disjoint.

Example (**Id** & **All**-constraints)

Id: each node x has access to a fact `Id(`x`)` [Ameloot et al., '11]



Constraints

Modeling System Knowledge

Definition

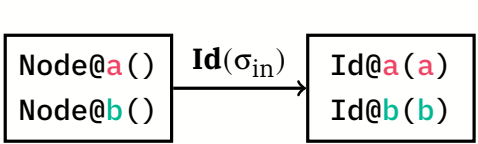
A **configuration constraint** $\mathbf{C}$ is a **total function** which, for all possible input schemas σ_{in}, maps onto a **behavior** $\mathbf{C}(\sigma_{\text{in}})$ from σ_{in} to σ_{sys}

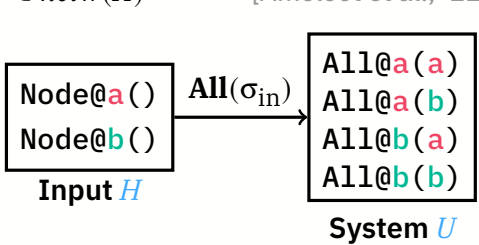
We can compose constraints (*i.e.*, $\mathbf{C}$+$\mathbf{C}'$) if the σ_{sys} of their behaviors are disjoint.

Example (**Id** & **All**-constraints)

Id: each node x has access to a fact `Id(`x`)` [Ameloot et al., '11]

All: each node has facts `All(`y`)` for every $y \in netw(H)$ [Ameloot et al., '11]





Overview

Introduction

Behaviors and Transducers

Computation of Behaviors

Weakly Coordinating Behaviors

Computation

Definition (Behavior Class $\mathcal{F}(\mathbf{C})$)

A $\mathbf{C}$-transducer Π **computes** a behavior B if,

- for all input instances H, and
- for all fair runs $\mathcal{R}$ of Π on H,

we have

union of output facts on all nodes, annotated with their location

$$(H, out(\mathcal{R})) \in B$$

Computation

Definition (Behavior Class $\mathcal{F}(\mathbf{C})$)

A $\mathbf{C}$-transducer Π **computes** a behavior B if,

- for all input instances H, and
- for all fair runs $\mathcal{R}$ of Π on H,

we have

union of output facts on all nodes, annotated with their location

$$(H, out(\mathcal{R})) \in B$$

Definition (Behavior Class $\mathcal{G}(\mathbf{C})$)

B **characterizes** a $\mathbf{C}$-transducer Π if,

- Π computes B, and
- for all pairs $(H, K) \in B$ there exists a run $\mathcal{R}$ of Π on H with $out(\mathcal{R}) = K$

Leader Election

- $\mathbf{Id}$+$\mathbf{All}$-transducers are the model of the first paper [Ameloot et al., '11]
- $\mathcal{F}(\mathbf{Id}+\mathbf{All})$ = all queries (unrelated to coordination-free) [Ameloot et al., '11]
- Is this model strong enough for behaviors? *i.e.*: $\textit{LEADER-ELECTION} \in \mathcal{F}(\mathbf{Id}+\mathbf{All})$?

Leader Election

- $\mathbf{Id}$+$\mathbf{All}$-transducers are the model of the first paper [Ameloot et al., '11]
- $\mathcal{F}(\mathbf{Id}+\mathbf{All})$ = all queries (unrelated to coordination-free) [Ameloot et al., '11]
- Is this model strong enough for behaviors? *i.e.*: $\textit{LEADER-ELECTION} \in \mathcal{F}(\mathbf{Id}+\mathbf{All})$?

Theorem

$\textit{LEADER-ELECTION} \notin \mathcal{F}(\mathbf{Id}+\mathbf{All})$

Leader Election

- $\mathbf{Id}$+$\mathbf{All}$-transducers are the model of the first paper [Ameloot et al., '11]
- $\mathcal{F}(\mathbf{Id}+\mathbf{All})$ = all queries (unrelated to coordination-free) [Ameloot et al., '11]
- Is this model strong enough for behaviors? *i.e.*: $\textit{LEADER-ELECTION} \in \mathcal{F}(\mathbf{Id}+\mathbf{All})$?

Theorem

$\textit{LEADER-ELECTION} \notin \mathcal{F}(\mathbf{Id}+\mathbf{All})$

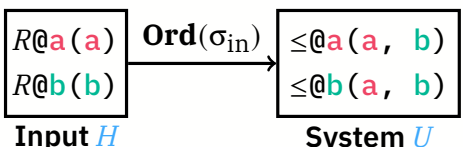
Proof idea.

- Π are defined in terms of **generic**, domain-preserving left-total relations M
  - **genericity:** $(H, K) \in M$ iff $(\beta(H), \beta(K)) \in M$ for all bijective renamings β
- as a result, Π unable to **break the symmetry** present in input instances H
  - Nodes have no way to 'choose' a specific node identifier as a leader

Ord & L

- **Ord**: nodes may access an order relation $\leq$ over active domain
  - Treat $\leq$ as a predicate, modeled by using an access label $acc(\leq) = \text{input}$

[Nash, Ludäscher, PODS '04]

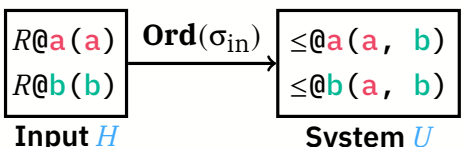


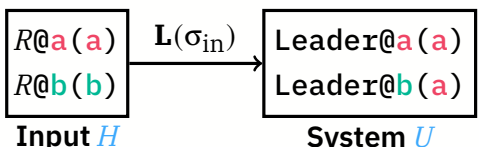
Ord & L

- **Ord**: nodes may access an order relation $\leq$ over active domain
  - Treat $\leq$ as a predicate, modeled by using an access label $acc(\leq) = \text{input}$

[Nash, Ludäscher, PODS '04]

- **L**: each node knows a dedicated leader node `Leader(`y`)`



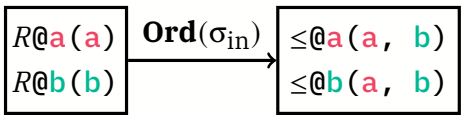


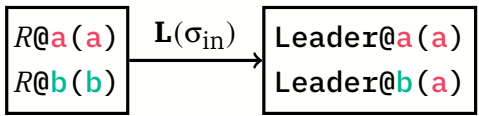
Ord & L

- **Ord**: nodes may access an order relation $\leq$ over active domain
  - Treat $\leq$ as a predicate, modeled by using an access label $acc(\leq) = \text{input}$

[Nash, Ludäscher, PODS '04]

- **L**: each node knows a dedicated leader node `Leader(`y`)`





Proposition

1. *LEADER-ELECTION* $\in \mathcal{F}(\mathbf{Id}+\mathbf{All}+\mathbf{L})$
2. $\mathcal{G}(\mathbf{Id}+\mathbf{All}+\mathbf{Ord}) = \mathcal{G}(\mathbf{Id}+\mathbf{All}+\mathbf{L})$

All Behaviors

- Which class contains all behaviors?

All Behaviors

- Which class contains all behaviors?

Theorem

$\mathcal{G}(\mathbf{Id}+\mathbf{All}+\mathbf{L}) = \textit{All Behaviors}$

All Behaviors

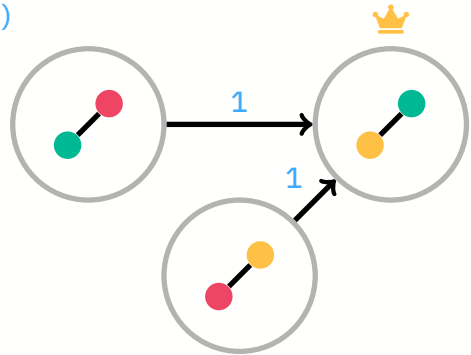
- Which class contains all behaviors?

Theorem

$\mathcal{G}(\mathbf{Id}+\mathbf{All}+\mathbf{L}) = \textit{All Behaviors}$

Proof idea. (All Behaviors $\subseteq \mathcal{G}(\mathbf{Id}+\mathbf{All}+\mathbf{L})$)

1. Each node sends its part of the input to the leader



All Behaviors

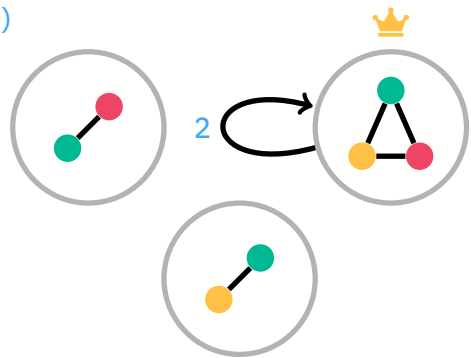
- Which class contains all behaviors?

Theorem

$\mathcal{G}(\mathbf{Id}+\mathbf{All}+\mathbf{L}) = \textit{All Behaviors}$

Proof idea. (All Behaviors $\subseteq \mathcal{G}(\mathbf{Id}+\mathbf{All}+\mathbf{L})$)

1. Each node sends its part of the input to the leader
2. Leader computes behavior B



All Behaviors

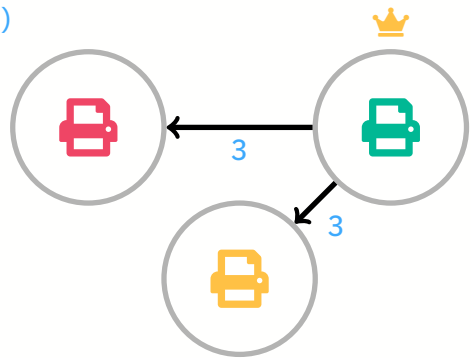
- Which class contains all behaviors?

Theorem

$\mathcal{G}(\mathbf{Id}+\mathbf{All}+\mathbf{L}) = \textit{All Behaviors}$

Proof idea. (All Behaviors $\subseteq \mathcal{G}(\mathbf{Id}+\mathbf{All}+\mathbf{L})$)

1. Each node sends its part of the input to the leader
2. Leader computes behavior B
3. Leader sends the appropriate output back to each node



All Behaviors

- Which class contains all behaviors?

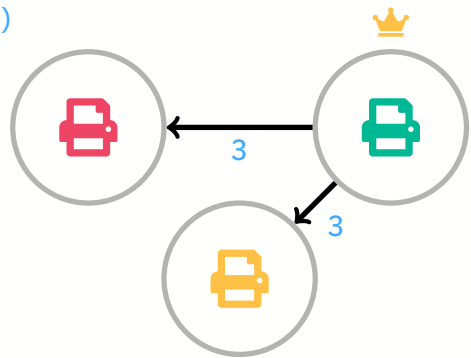
Theorem

$\mathcal{G}(\mathbf{Id}+\mathbf{All}+\mathbf{L}) = \textit{All Behaviors}$

Proof idea. (All Behaviors $\subseteq \mathcal{G}(\mathbf{Id}+\mathbf{All}+\mathbf{L})$)

1. Each node sends its part of the input to the leader
2. Leader computes behavior B
3. Leader sends the appropriate output back to each node

($\mathcal{G}(\mathbf{Id}+\mathbf{All}+\mathbf{L}) \subseteq$ All Behaviors)
Follows from the definition



Overview

Introduction

Behaviors and Transducers

Computation of Behaviors

Weakly Coordinating Behaviors

Coordination-Freedom

Definition

- A behavior B is **coordination-free** if $B \in \mathcal{F}(\mathbf{Id}{+}\mathbf{Ord})$
- B is **weakly coordinating** if $B \in \mathcal{F}(\mathbf{C})$ where $\mathbf{All}(\sigma_{\text{in}}) \notin \mathcal{F}_T(\mathbf{C})$

$\mathcal{F}_T(\mathbf{C})$: class of behaviors computable with 'termination-aware' transducers.

Monotonicity for Behaviors

Definition (Behavior Class $\mathcal{M}_{\text{node}}$)

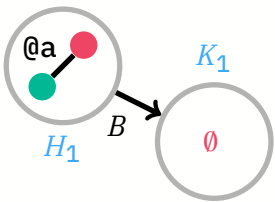
A behavior B is **node monotone** if

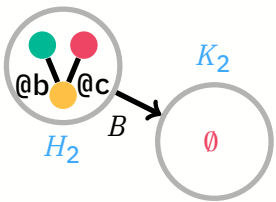
- for all pairs of input instances H_1, H_2 with $netw(H_1) \cap netw(H_2) = \emptyset$, and
- for all $(H_1, K_1), (H_2, K_2) \in B$,

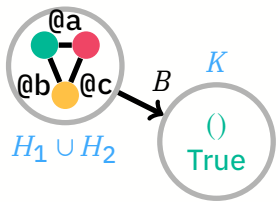
there exists a $(H_1 \cup H_2, K) \in B$ with $K_1 \cup K_2 \subseteq K$

- $\mathcal{M}_{\mathbf{C}}$: class of node monotone behaviors 'modulo' some constraint $\mathbf{C}$

Example (Triangles)







CALM

Theorem

$\mathcal{M}_{\text{node}} = \mathcal{G}(\mathbf{Id}+\mathbf{Ord})$

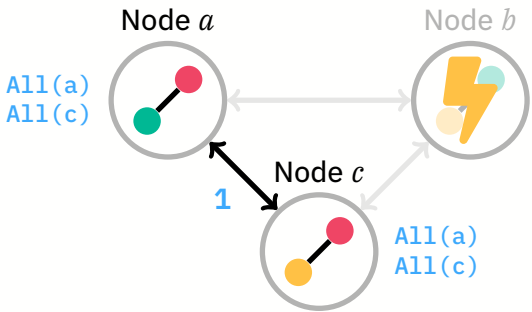
CALM

Theorem

$\mathcal{M}_{\text{node}} = \mathcal{G}(\mathbf{Id}{+}\mathbf{Ord})$

Proof idea. ($\mathcal{M}_{\text{node}} \subseteq \mathcal{G}(\mathbf{Id}{+}\mathbf{Ord})$ only)

1. nodes broadcast their `Id` and agree on a (partial) `All`



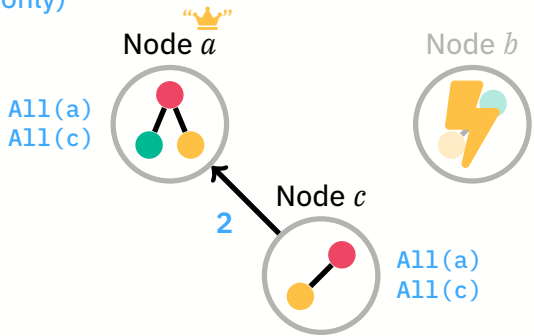
CALM

Theorem

$\mathcal{M}_{\text{node}} = \mathcal{G}(\mathbf{Id}{+}\mathbf{Ord})$

Proof idea. ($\mathcal{M}_{\text{node}} \subseteq \mathcal{G}(\mathbf{Id}{+}\mathbf{Ord})$ only)

1. nodes broadcast their `Id` and agree on a (partial) `All`
2. nodes in the `All` fragment send input to lowest node



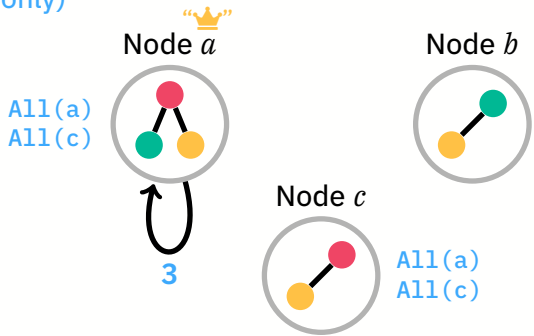
CALM

Theorem

$\mathcal{M}_{\text{node}} = \mathcal{G}(\mathbf{Id}{+}\mathbf{Ord})$

Proof idea. ($\mathcal{M}_{\text{node}} \subseteq \mathcal{G}(\mathbf{Id}{+}\mathbf{Ord})$ only)

1. nodes broadcast their `Id` and agree on a (partial) `All`
2. nodes in the `All` fragment send input to lowest node
3. lowest node computes B



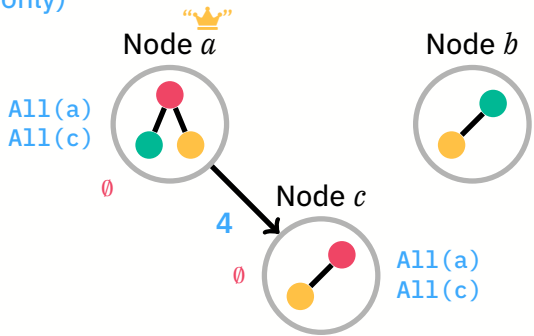
CALM

Theorem

$\mathcal{M}_{\text{node}} = \mathcal{G}(\mathbf{Id}{+}\mathbf{Ord})$

Proof idea. ($\mathcal{M}_{\text{node}} \subseteq \mathcal{G}(\mathbf{Id}{+}\mathbf{Ord})$ only)

1. nodes broadcast their `Id` and agree on a (partial) `All`
2. nodes in the `All` fragment send input to lowest node
3. lowest node computes B
4. redistribute output, then repeat steps 1-4



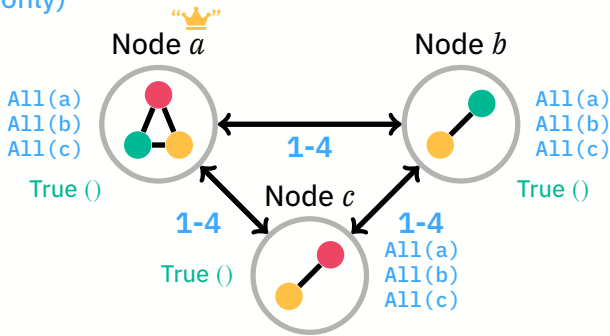
CALM

Theorem

$\mathcal{M}_{\text{node}} = \mathcal{G}(\mathbf{Id}{+}\mathbf{Ord})$

Proof idea. ($\mathcal{M}_{\text{node}} \subseteq \mathcal{G}(\mathbf{Id}{+}\mathbf{Ord})$ only)

1. nodes broadcast their `Id` and agree on a (partial) `All`
2. nodes in the `All` fragment send input to lowest node
3. lowest node computes B
4. redistribute output, then repeat steps 1-4



CALM

Theorem

$\mathcal{M}_{\text{node}} = \mathcal{G}(\mathbf{Id}{+}\mathbf{Ord})$

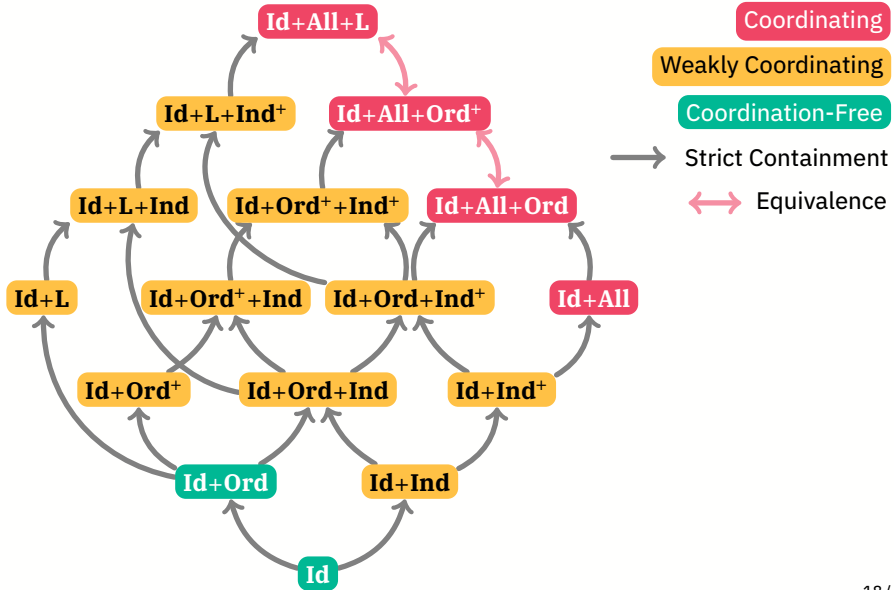
Proof idea. ($\mathcal{M}_{\text{node}} \subseteq \mathcal{G}(\mathbf{Id}{+}\mathbf{Ord})$ only)

1. nodes broadcast their `Id` and agree on a (partial) `All`
2. nodes in the `All` fragment send input to lowest node
3. lowest node computes B
4. redistribute output, then repeat steps 1-4

Theorem

$\mathcal{M}_{\mathbf{C}} = \mathcal{G}(\mathbf{Id}{+}\mathbf{Ord}{+}\mathbf{C})$

Coordination Spectrum



PTIME and CALM

- Are all *PTIME* queries over ordered databases coordination-free?

  [Hellerstein and Alvaro, CACM '20]

- **Ord**$^+$: the **Ord** constraint plus `Min`/`Max` domain element.
- **Ind**: access to a distributed index (which node responsible for which data?)

PTIME and CALM

- Are all *PTIME* queries over ordered databases coordination-free?

[Hellerstein and Alvaro, CACM '20]

- **Ord**$^+$: the **Ord** constraint plus `Min`/`Max` domain element.
- **Ind**: access to a distributed index (which node responsible for which data?)

Theorem

$Q \in PTIME$ exists with $B_Q \notin \mathcal{F}(\mathbf{Id}+\mathbf{Ord}^+ + \mathbf{Ind})$

PTIME and CALM

- Are all *PTIME* queries over ordered databases coordination-free?

[Hellerstein and Alvaro, CACM '20]

- $\mathbf{Ord}^+$: the **Ord** constraint plus `Min`/`Max` domain element.
- **Ind**: access to a distributed index (which node responsible for which data?)

Theorem

$Q \in$ PTIME exists with $B_Q \notin \mathcal{F}(\mathbf{Id}+\mathbf{Ord}^+ + \mathbf{Ind})$

Proof idea.

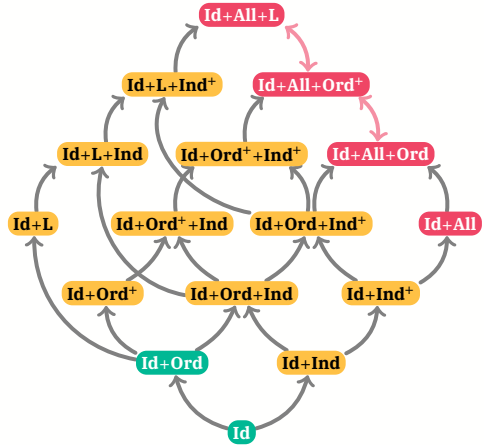
- $Q_{\neg E}$: is there an edge not connected to any other edge in this graph?
- $B_{Q_{\neg E}} \notin \mathcal{M}_{\mathbf{Ord}^+ + \mathbf{Ind}}$

Open Questions

1. Value invention for transducers?
   - Facilitate talking about time, counters, arithmetic, ...
   - Many original CALM-scenarios depend on a notion of time
2. *LEADER-ELECTION* is sometimes computable with **Id** + **All**-transducers if input instance has enough asymmetry built in, precisely when?
3. Investigate connections with related topics:
   - Conflict-Free Replicated Data Types (CRDTs)
   - Strong Eventual Consistency (and other consistency models)
   - Shared Knowledge in Multi-Agent Systems
   - ...
4. Case studies of concrete problems using CALM framework?
   - Transaction Protocols
   - Distributed/Federated Querying
   - Declarative Programming Languages
   - ...

Contributions

- **Behaviors**
  model for non-deterministic problems with data placement
- **Generalized Relational Transducers**
  model any system constraint $\mathbf{C}$
- **CALM for Behaviors**
  $\mathcal{M}_{\text{node}} = \mathcal{G}(\mathbf{Id}{+}\mathbf{Ord})$
- **CALM Generalization**
  $\mathcal{M}_{\mathbf{C}} = \mathcal{G}(\mathbf{Id}{+}\mathbf{Ord}{+}\mathbf{C})$
- **Coordination Spectrum**
  stronger systems = weaker coordination-freedom



Tim Baccaert <tim.baccaert@vub.be>
Bas Ketsman <bas.ketsman@vub.be>

Previous Definitions of Coordination-Freedom

- *Coordination-free*: there exists an **input distribution** so that nodes can compute the output without messaging [Ameloot et al., ’11]
  - **Problem:** assumes data is never tied to specific nodes
- *Coordination-free*: anything that can be computed with a transducer that does not use **All** [Ameloot et al., ’11]
- definitions are equivalent, in the sense that the same queries are coordination-free under both definitions (also used in [Ameloot et al., ’14])

More Detailed Definition of Transducers

Definition

A (generalized, relational) **transducer** Π is a 5-tuple $(\mathbf{C}, M_{in}, M_{del}, M_{snd}, M_{out})$:

- $\mathbf{C}$ is a configuration constraint
- M_{in} is a (generic, domain-preserving) left-total relation mapping input and system facts into this transducer's 'input' buffer
- M_{del} is a (generic, domain-preserving) left-total relation mapping input and system facts into this transducer's 'delete' buffer
- M_{snd} is a (generic, domain-preserving) left-total relation mapping input facts into the global message buffer.
- M_{out} is a (generic, domain-preserving) left-total relation mapping 'input' minus 'delete' state facts into a write-only output buffer

Successor vs Order for $PTIME$

Theorem

$Q \in PTIME$ exists with $B_Q \notin \mathcal{F}(\mathbf{Id}+\mathbf{Ord}^+ + \mathbf{Ind})$

- If we replace $\leq$ by a successor relation, counterexample in our proof breaks. We can enumerate the active domain.
  - $Q_{\neg E}$: is there an edge not connected to any other edge in this graph?
  - $B_{Q_{\neg E}} \in \mathcal{M}_{\mathbf{Succ}^+ + \mathbf{Ind}}$
- then, can be shown that **all queries** become computable
- this includes *being aware that we have computed the All relation*
  - hence, this combination of constraints is not (weakly) coordination-free.