

On the Semantics of Coordination Models for Distributed Systems: the LogOp Case Study



Ronaldo Menezes
Dept. of Computer Sciences
Florida Institute of Technology
rmenezes@cis.fit.edu



Andrea Omicini, Mirko Viroli
DEIS – Cesena
Università di Bologna
{aomicini,mviroli}@deis.unibo.it

Outline of the Talk

- Motivations for a semantic study of LogOp
- LogOp
 - the more intuitive interpretation..
 - a more reasonable one..
- Reasoning about a LogOp infrastructure
 - from the semantic model to the infrastructure
 - duality coordination as a language / as a service
- Lessons Learnt and Final Remarks

General Motivations – Why this Research?

- Need for formal frameworks for the specification of coordination models
- Design of coord. infrastructures as important as the design of coord. models
- The choice of the most suitable formal framework:
 - is relevant for specifying a coordination model
 - avoiding any ambiguity and incorrectness
 - is crucial for specifying an infrastructure
 - checking for the realisability of a model
 - guiding to sound implementations

Specific Motivations – Why LogOp?

- LogOp extends Linda with multi-space operations
 - [Snyder&Menezes@COORDINATION2002]
 - coordination laws dynamically involving a multiplicity of physically distributed tuple spaces
- LogOp raises general problems
 - concerning distributed coordinated systems in general: synchronicity, atomicity, locality, ...
- LogOp lacks a formal semantics
 - is it sensible to just rely on its (prototype) run-time interpreting the informal specification?

LogOp extension to Linda

- Linda in a multi-space setting
 - we stick to the version without matching
 - $out(t,x)$: inserts tuple x into tuplespace t
 - $rd(t,x)$: reads tuple x from tuplespace t
 - $in(t,x)$: removes tuple x from tuplespace t
- LogOp, primitives involving different spaces
 - motivation: need to dynamically define scopes
 - idea: rely on logic operators to combine spaces
- Syntax
 - for any Linda primitive $\pi ::= in|rd|out$
 - $\pi(AND(t_1, \dots, t_n), x)$: over ALL spaces $t_1, \dots, t_n$
 - $\pi(XOR(t_1, \dots, t_n), x)$: over ONE space in $t_1, \dots, t_n$
 - $\pi(OR(t_1, \dots, t_n), x)$: over SOME spaces in $t_1, \dots, t_n$

$$\alpha ::= \pi(\lambda, x), \quad \lambda ::= t \wedge \dots \wedge t \mid t \otimes \dots \otimes t \mid t \vee \dots \vee t$$

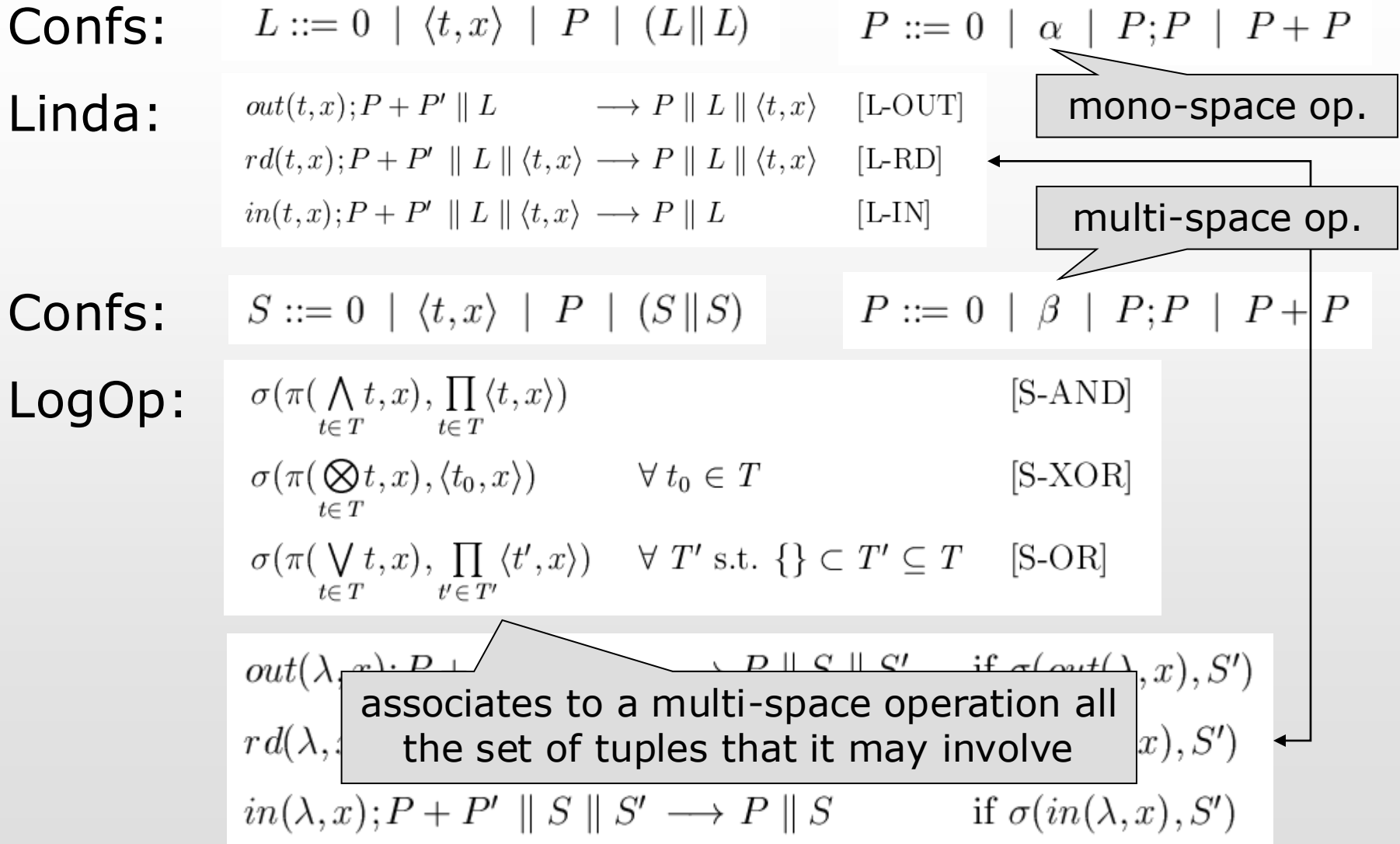
LogOp examples

- Inserting tuples in more spaces, atomically
 - $\text{out}(t_1 \wedge t_2, x)$
 - no need for invoking more primitives!
- Reading the tuple x from some spaces
 - $\text{rd}(t_1 \vee t_2 \vee t_3, x)$
 - “best effort to read as many x as possible”! (?)
 - with matching: yields a list of pairs $[t, x']$
- Removing one tuple x from some spaces
 - $\text{in}(t_1 \otimes t_2 \otimes t_3, x)$
 - only one removal guaranteed!

LogOp: twofold interpretation

- It may admit different interpretations..
 - need for a formal model!
- From the viewpoint of atomicity/synchrony
 - Strong-Synchrony
 - atomicity of the execution of coordination primitives as a global property of the system
 - Weak-Synchrony
 - atomicity of the execution of coordination primitives as a local view of the agent
- .. generality of this twofold interpretation for distributed systems

Strong-Synchrony for LogOp



On Implementation..

- This specification is quite simple, but what about implementation?
- To enforce a notion of global atomicity:
 - transactional mechanisms are required (e.g. locks)
 - they can be quite costly, especially in distributed systems
- An alternative semantics:
 - releases the global atomicity constraints
 - enforces atomicity as perceived by the agent

Weak-Synchrony for LogOp

- Defined as a mapping
 - a LogOp process turned to a Linda cfg of processes

- Examples: $|out(t_1 \wedge t_2, x)|_{\mathcal{O}} = out(t_1, x); out(t_2, x) + out(t_2, x); out(t_1, x)$

$$|in(t_1 \otimes t_2, x)|_{\mathcal{O}} = in(t_1, x) + in(t_2, x)$$

either as for XOR (only one space) or non-deterministically execute on one space and proceed recursively!

- Mapping

$$|\pi(\bigwedge_{t \in T} t, x)|_{\mathcal{O}} \triangleq \sum_{t \in T} \left(\pi(t, x); |\pi(\bigwedge_{t' \in T \setminus \{t\}} t', x)|_{\mathcal{O}} \right) \quad [\text{W-AND}]$$

$$|\pi(\bigotimes_{t \in T} t, x)|_{\mathcal{O}} \triangleq \sum_{t \in T} \pi(t, x) \quad [\text{W-XOR}]$$

$$|\pi(\bigvee_{t \in T} t, x)|_{\mathcal{O}} \triangleq |\pi(\bigotimes_{t \in T} t, x)|_{\mathcal{O}} + \sum_{t \in T} \left(\pi(t, x); |\pi(\bigvee_{t' \in T' \setminus \{t\}} t', x)|_{\mathcal{O}} \right) \quad [\text{W-OR}]$$

Strong vs. Weak

$$in(t_1 \wedge t_2, x); out(t_3, x) \parallel in(t_1 \wedge t_2, x); out(t_3, x) \parallel \langle t_1, x \rangle \parallel \langle t_2, x \rangle$$

strong-
synchrony

weak-
synchrony

$$\begin{aligned} & (in(t_1, x); in(t_2, x) + in(t_2, x); in(t_1, x)); out(t_3, x) \parallel \\ & (in(t_1, x); in(t_2, x) + in(t_2, x); in(t_1, x)); out(t_3, x) \parallel \\ & \langle t_1, x \rangle \parallel \langle t_2, x \rangle \end{aligned}$$

$$\begin{aligned} & in(t_2, x); out(t_3, x) \parallel \\ & (in(t_1, x); in(t_2, x) + in(t_2, x); in(t_1, x)); out(t_3, x) \parallel \\ & \langle t_2, x \rangle \end{aligned}$$

$$out(t_3, x) \parallel in(t_1 \wedge t_2, x); out(t_3, x)$$

$$in(t_1 \wedge t_2, x); out(t_3, x) \parallel \langle t_3, x \rangle$$

$$in(t_2, x); out(t_3, x) \parallel in(t_1, x); out(t_3, x)$$

On Weak-Synchrony

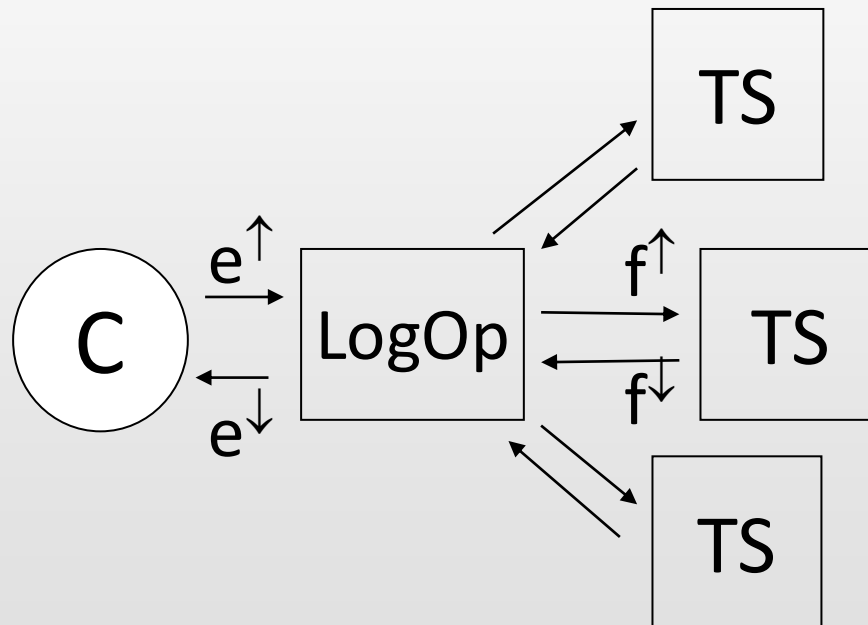
- Basic differences w.r.t. strong-synchrony
 - the execution of a primitive may be influenced by the execution of others
 - the requesting agent remains blocked!
- Pros and Cons of Weak-Synchrony
 - guarantees less properties
 - more easily realisable (involves less overhead)
 - it still has the advantages expected from LogOp
- Is the preferred semantics for an implementation over distributed systems!
 - e.g. where deadlock can be recovered a posteriori

Specifying an Infrastructure

- See [Viroli&Omicini@FOCLASA2002]
- Coordination as a Language
 - endorsed by the application to closed systems
 - coordination as a language to express interaction in concurrent applications
 - good to reason about coordination models
- Coordination as a Service
 - endorsed by the application to open/distributed systems
 - coordination as an interactive service provided by coordination media (first-class entities)
 - good to specify coordination infrastructures

The LogOp Service

- A design for a LogOp infrastructure
 - sticking to existing, legacy Linda tuplespaces
 - one (or more) LogOp coordination medium as frontends



The LogOp Infrastructure

- To be defined:
 - an interactive model for Linda tuplespaces
 - an interactive model for the LogOp medium
 - a composition mechanism
- Composition:
 - according to the framework in [FOCLASA2002]
 - coordination space as a parallel composition of coordination media:
 - one LogOp medium
 - more tuplespaces
 - they interact by asynchronously posting request and reply events
 - using the schema of asynchronous π -calculus

Linda Tuplespaces

- As in [Viroli@SAC2003]

- States:

$$TS ::= 0 \mid f^\uparrow \mid f^\downarrow \mid x \mid (TS \parallel TS)$$

- Behaviour:

$$TS \xrightarrow{f^\uparrow}_{TS} TS \parallel f^\uparrow \quad [\text{TS-REQ}]$$

$$TS \parallel f^\downarrow \xrightarrow{f^\downarrow}_{TS} TS \quad [\text{TS-REP}]$$

$$n \upharpoonright out(x) \parallel TS \xrightarrow{\tau}_{TS} TS \parallel x \quad [\text{TS-OUT}]$$

$$n \upharpoonright rd(x) \parallel TS \parallel x \xrightarrow{\tau}_{TS} TS \parallel x \parallel n \downharpoonright ok \quad [\text{TS-RD}]$$

$$n \upharpoonright in(x) \parallel TS \parallel x \xrightarrow{\tau}_{TS} TS \parallel n \downharpoonright ok \quad [\text{TS-IN}]$$

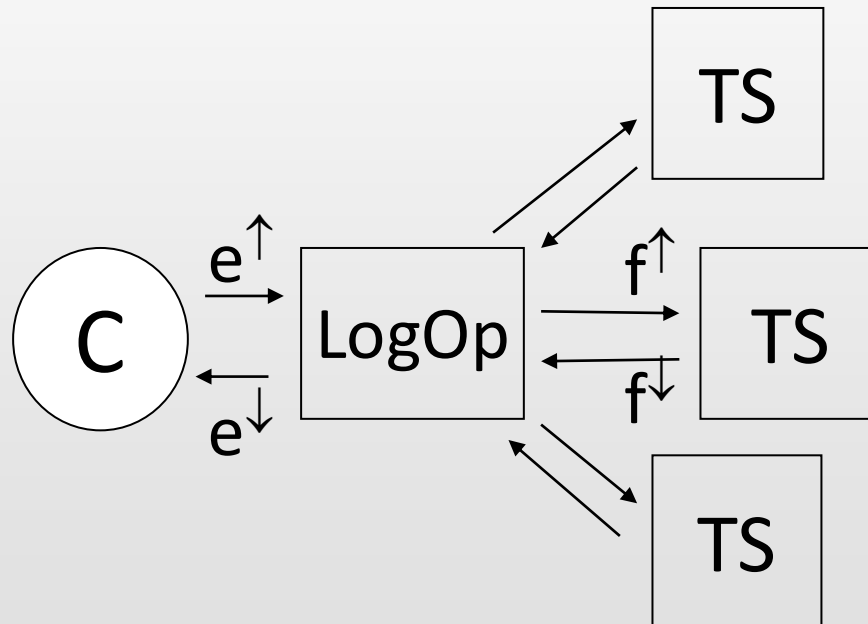
positive reply to “n”

The LogOp Medium

- Conceptual steps:
 1. receives requests for executing LogOp operations (multi-space ops)
 2. executes mono-space ops. over tuplespaces
 3. eventually gathers and manages replies
 4. produces a reply to the agent
- Most notably, for each request two internal handlers are spawned:
 - one sending mono-space requests
 - one managing the mono-space replies

The LogOp Service

- A design for a LogOp infrastructure
 - sticking to existing, legacy Linda tuplespaces
 - one (or more) LogOp coordination media as frontends



Sender Handler

- For primitive OUT, semantics induced by the definition of relation σ

$$\sigma(\pi(\bigwedge_{t \in T} t, x), \prod_{t \in T} \langle t, x \rangle) \quad [\text{S-AND}]$$

$$\sigma(\pi(\bigotimes_{t \in T} t, x), \langle t_0, x \rangle) \quad \forall t_0 \in T \quad [\text{S-XOR}]$$

$$\sigma(\pi(\bigvee_{t \in T} t, x), \prod_{t' \in T'} \langle t', x \rangle) \quad \forall T' \text{ s.t. } \{\} \subset T' \subseteq T \quad [\text{S-OR}]$$

$$h(id \uparrow \text{out}(\bigwedge_{t \in T} t, x), \prod_{t \in T} n \uparrow \text{out}(t, x))$$

$$h(id \uparrow \text{out}(\bigotimes_{t \in T} t, x), n \uparrow \text{out}(t_0, x)) \quad \forall t_0 \in T$$

$$h(id \uparrow \text{out}(\bigvee_{t \in T} t, x), \prod_{t' \in T'} n \uparrow \text{out}(t', x)) \quad \forall T' \text{ s.t. } \{\} \subset T' \subseteq T$$

- For primitives RD and IN (depends on liveness!)
 - either send requests to all tuplespaces
 - or, send requests to some tuplespace (for OR and XOR)

Receiver Handler

- For primitive OUT: just a void process
- For primitive RD:
 1. gets one (XOR), some (OR), or all (AND) replies
 2. produces the reply message for the agent
 3. consumes the possible remaining replies
- For primitive IN:
 - similarly to RD
 - for XOR, each further consumed reply (step 3) is followed by an OUT restoring the removed tuple

Receiver Handler: formulae

- Induced by the mapping for “weak-synchrony”

$$\begin{aligned}
 |\pi(\epsilon, x)|_{\mathcal{O}} &\triangleq 0 \\
 |\pi(\bigwedge_{t \in T} t, x)|_{\mathcal{O}} &\triangleq \sum_{t \in T} \left(\pi(t, x); |\pi(\bigwedge_{t' \in T \setminus \{t\}} t', x)|_{\mathcal{O}} \right) \\
 |\pi(\bigotimes_{t \in T} t, x)|_{\mathcal{O}} &\triangleq \sum_{t \in T} \pi(t, x) \\
 |\pi(\bigvee_{t \in T} t, x)|_{\mathcal{O}} &\triangleq |\pi(\bigotimes_{t \in T} t, x)|_{\mathcal{O}} + \sum_{t \in T} \left(\pi(t, x); |\pi(\bigvee_{t' \in T \setminus \{t\}} t', x)|_{\mathcal{O}} \right)
 \end{aligned}$$

$$recv(id \uparrow \gamma(\epsilon, x), \nu) = !id \downarrow ok$$

$$recv(id \uparrow \gamma(\bigwedge_{t \in T} t, x), \nu) = \sum_{t \in T} ?n_t \downarrow ok; recv(id \uparrow \gamma(\bigwedge_{t' \in T \setminus \{t\}} t', x), \nu)$$

$$recv(id \uparrow \gamma(\bigotimes_{t \in T} t, x), \nu) = \sum_{t \in T} ?\nu(t) \downarrow ok; !id \downarrow ok; finish(id \uparrow \gamma(\bigotimes_{t' \in T \setminus \{t\}} t', x), \nu)$$

$$recv(id \uparrow \gamma(\bigvee_{t \in T} t, x), \nu) = recv(id \uparrow \gamma(\bigotimes_{t \in T} t, x), \nu) + \sum_{t \in T} ?\nu(t) \downarrow ok; recv(id \uparrow \gamma(\bigvee_{t' \in T \setminus \{t\}} t', x), \nu)$$

Remarks

- From a specification to an implementation less straightforward than in Linda [Viroli@SAC2003]
 - because of the distr. & synchronous character of LogOp
 - still, many parts of the weak-synchrony spec. can be reused
- Studying the LogOp service reveals tricks
 - in and rd have similar specification
 - but require quite different implementations
- A model specification can be counter-intuitive
 - strong-synchrony has a simpler semantic model
 - but its implementation would be more cumbersome
 - conversely for weak-synchrony

Lessons Learnt

- Formal models avoid crucial ambiguity
- Choosing the right abstraction level..
 - not only may simplify a specification
 - may reveal fundamental implementation issues
- What is “suitable” for a formal framework in the specification of a coord. infrastructure?
 - be ontologically coherent with the run-time
 - provide conceptual means for compliance

Conclusions

- LogOp is an interesting model raising some general issues
 - this work has paved a way for new development
 - grounded on an infrastructure specification
- Need for suitable formal frameworks for (coordination) infrastructures
 - not every formalism that works is good
- Models and infrastructures should be validated for correctness and compliance
 - and this holds in general for open & distributed systems
 - by extending the work in [SAC2003] there is room for systematically addressing compliance!

On the Semantics of Coordination Models for Distributed Systems: the LogOp Case Study



Ronaldo Menezes
Dept. of Computer Sciences
Florida Institute of Technology
rmenezes@cis.fit.edu



Andrea Omicini, Mirko Viroli
DEIS – Cesena
Università di Bologna
{aomicini,mviroli}@deis.unibo.it