Just Distributability

Bertinoro, June 2023

Rob van Glabbeek, Ursula Goltz
and Jens-Wolfhard Schicke-Uffmann

TECHNISCHE UNIVERSITÄT
CAROLO-WILHELMINA
ZU BRAUNSCHWEIG

Our Question

Which system specifications can be implemented as distributed systems?

Our Question

Which system specifications can be implemented as distributed systems?

Distributed systems:

- Components on different locations
- Signals between components take time to travel.

  So, communication is asynchronous.
-

Specifications allow synchronous communication.

How to simulate synchronous by asynchronous communication?

Our Question

Which system specifications can be implemented as distributed systems?

Distributed systems:

- Components on different locations
- Signals between components take time to travel.

  So, communication is asynchronous.
-

Specifications allow synchronous communication.

How to simulate synchronous by asynchronous communication?

Trivial solution: Locate the entire system in one spot.

Our Question

Which system specifications can be implemented as distributed systems?

Distributed systems:

- Components on different locations
- Signals between components take time to travel.

  So, communication is asynchronous.
- Components only allow sequential behaviour.

Specifications allow synchronous communication.

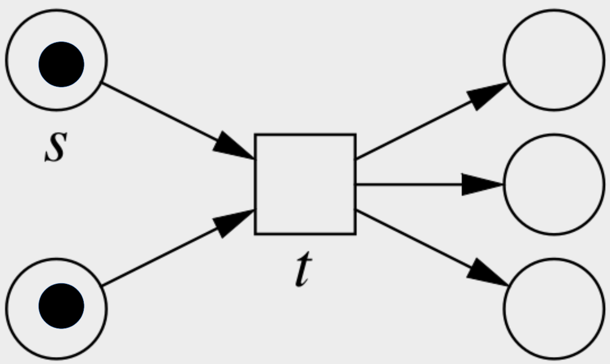
How to simulate synchronous by asynchronous communication?

System model

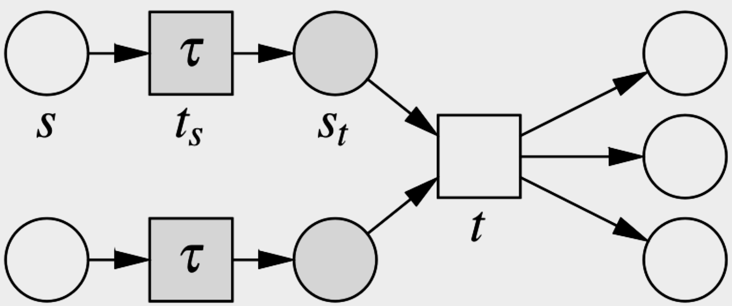
We need to model concurrency.

We pick Petri nets as our system model.

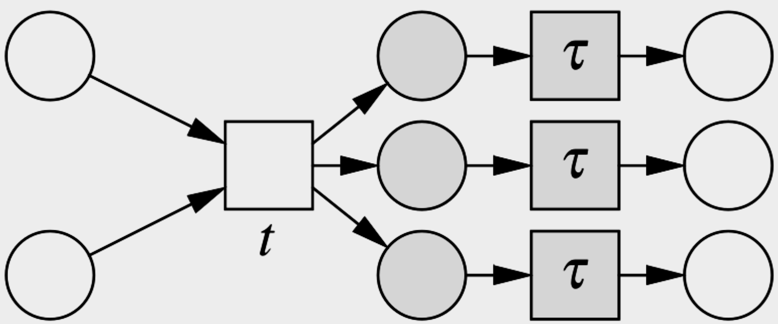
Quick or slow movement of tokens



(a)



(b)



(c)

Outline

- A net is distributed iff it is possible to assign a location to all places and transitions such that a transition is co-located with each of its preplaces and two co-located transitions can never fire in one step.

- A net is distributable iff it may be implemented as a distributed net in the above sense (i.e. consisting of sequential components).

Which nets can be distributed?

Up to which equivalence?

Outline

- A net is distributed iff it is possible to assign a location to all places and transitions such that a transition is co-located with each of its preplaces and two co-located transitions can never fire in one step.

- A net is distributable iff it may be implemented as a distributed net in the above sense (i.e. consisting of sequential components).

Which nets can be distributed?

Up to which equivalence?

[GGS 2008]: There are nets (with Ms) that can not be distributed up to an equivalence that takes concurrency, divergence and branching time into account to a small extent.

Outline

- A net is distributed iff it is possible to assign a location to all places and transitions such that a transition is co-located with each of its preplaces and two co-located transitions can never fire in one step.

- A net is distributable iff it may be implemented as a distributed net in the above sense (i.e. consisting of sequential components).

Which nets can be distributed?

Up to which equivalence?

Main result [Schicke 2008]: An intuitively satisfactory construction that gives a distributed implementation of every net, preserving completed ST-trace equivalence.

But: [Badouel,Caillaud,Darondeau 2002] provides a straightforward, yet unsatisfactory construction with the same properties.

New Requirement

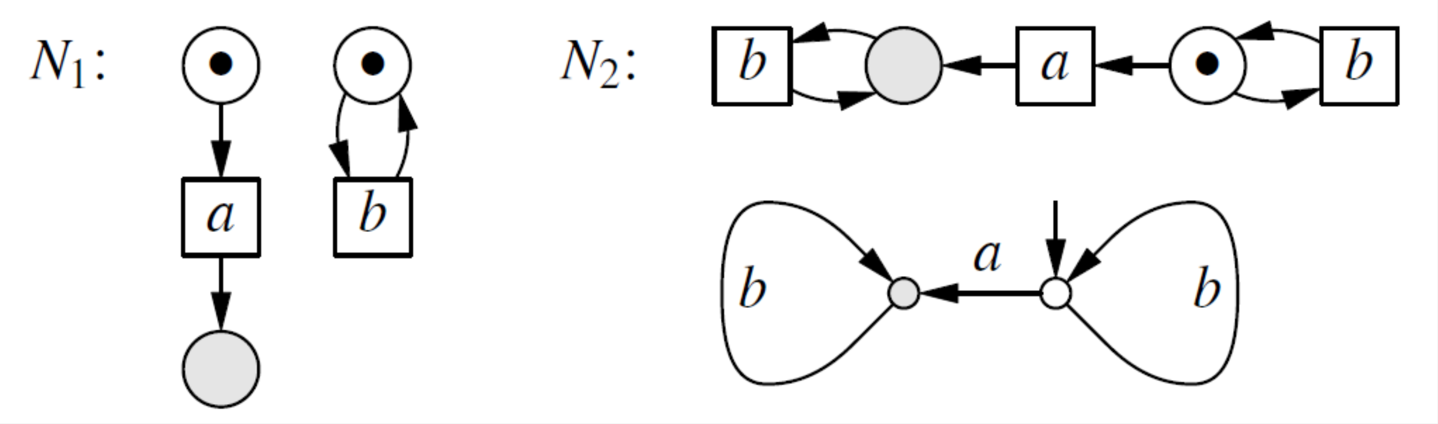
This talk proposes a new requirement on distributed implementations.

If an action is inevitable in the specification,
it must be inevitable in the distributed implementation.

Inevitability

An action is inevitable iff it occurs in each complete run of a net.

A run is complete iff no transition remains continuously enabled.



Open Problem

Does each finite Petri net admit a finite distributed implementation that preserves inevitability as well as interleaving trace equivalence – or even some finer equivalence?

Open Problem

Does each finite Petri net admit a finite distributed implementation that preserves inevitability as well as interleaving trace equivalence – or even some finer equivalence?

So far, we have no idea.

Open Problem

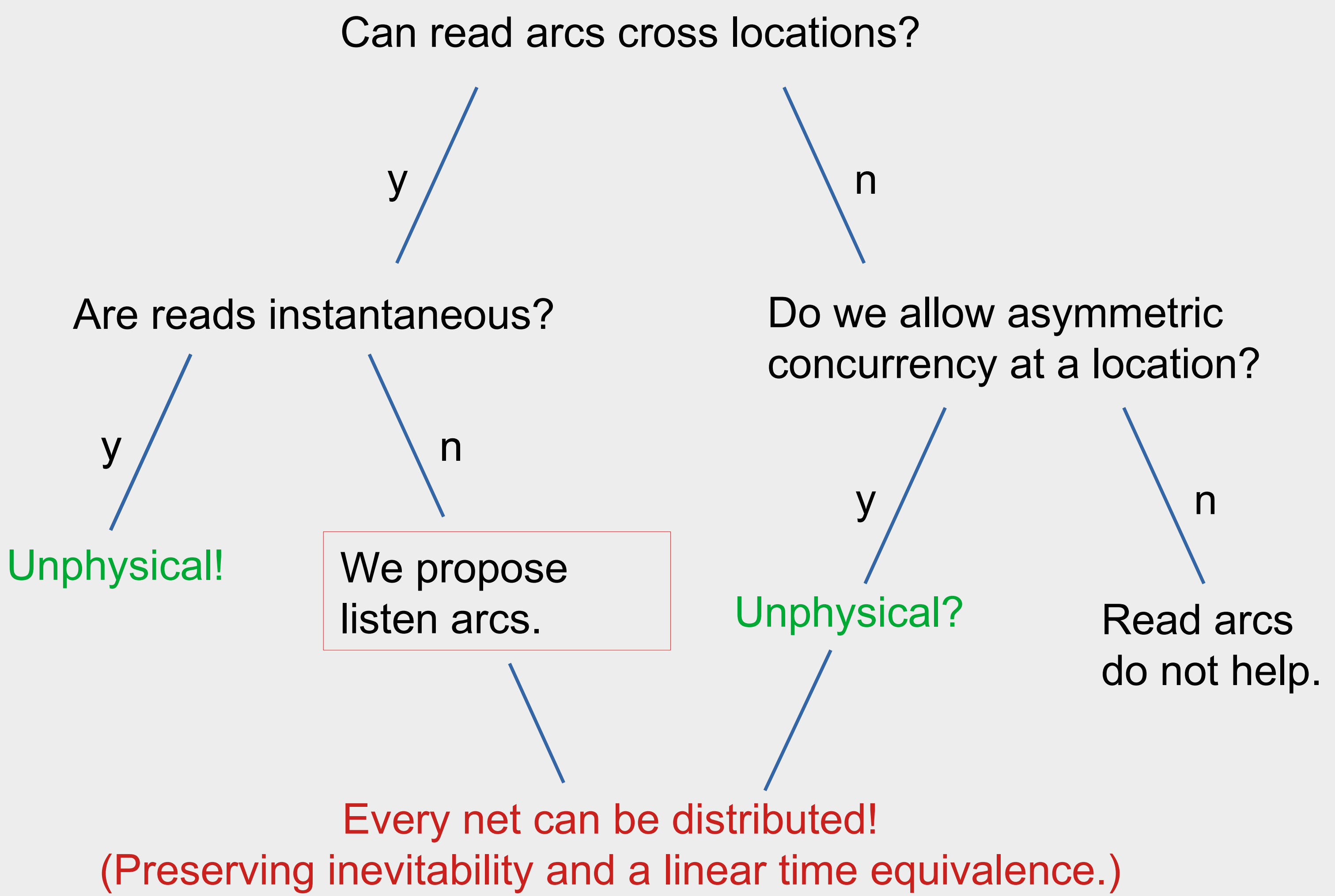
Does each finite Petri net admit a finite distributed implementation that preserves inevitability as well as interleaving trace equivalence – or even some finer equivalence?

So far, we have no idea.

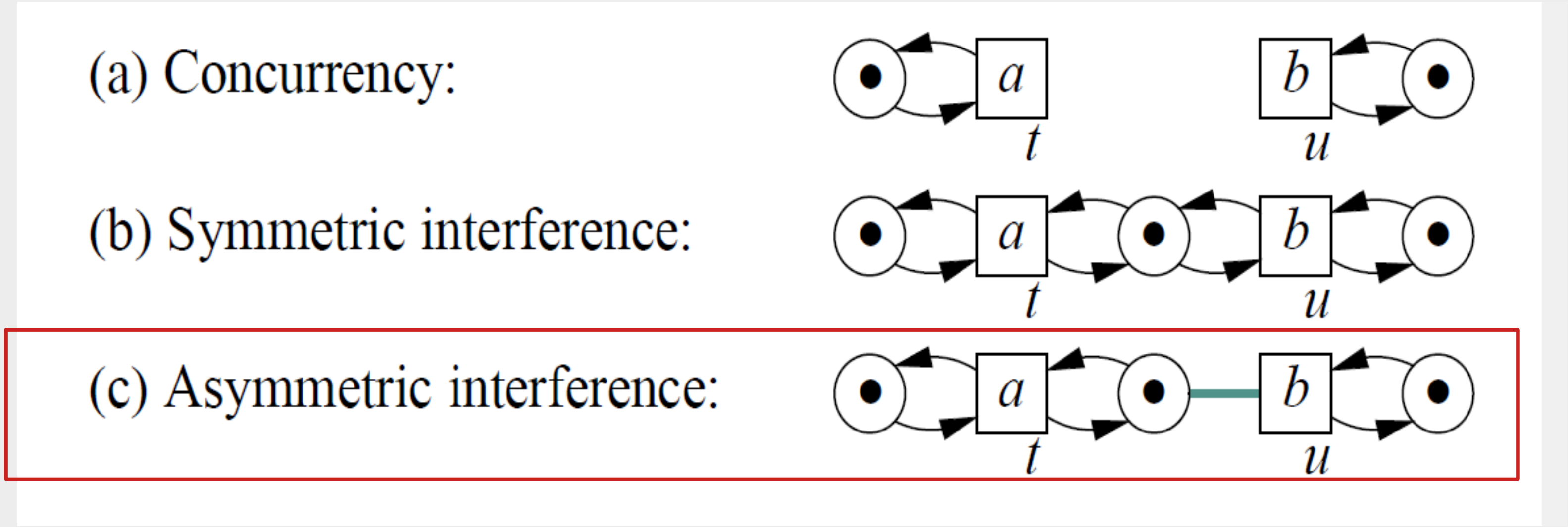
Walter Vogler [2002] has shown: Petri nets enriched with read arcs are more expressive than standard nets.

We now revisit the above question allowing read arcs in distributed implementations.

Read Arcs and Distributability



Asymmetric Concurrency Relation



(a) *a* and *b* inevitable

(b) neither inevitable

(c) *a* inevitable, but *b* not

Listen Arcs

A (labelled) Petri net with listen arcs is a tuple (S, T, F, L, M_0, l) where:

- S and T are disjoint sets (of places and transitions),
- $F \subseteq S \times T \cup T \times S$ (the flow relation),
- $L \subseteq S \times T$ (the listen relation),
- $M_0 \subseteq S \cup L$ (the initial marking), and
- $l: T \to Act$ (the labelling function) for some alphabet Act.

A state of a net is given by a marking $M \subseteq S \cup L$.

The firing rule allows three kinds of state changes:

- A listen arc (s,t) can be activated, $M \to M \cup \{(s,t)\}$, if $s \in M$.
- A listen arc (s,t) can be deactivated, $M \cup \{(s,t)\} \to M$, if $s \notin M$.
- A transition can fire as usual if all its incoming listen arcs are in M.

Complete Runs with Listen Arcs

A run of a net with listen arcs is complete iff

- no transition remains continuously enabled,
- no listen arc (s,t) remains inactive even though s remains marked, and
- no listen arc (s,t) remains active even though s remains unmarked.

Conclusions

- New requirement for distributed implementations of concurrent systems
- Rules out proposed implementations that were intuitively unsatisfactory
- Distributed implementation of general Petri nets, preserving inevitability as well as completed ST-trace equivalence, extending Petri Nets with listen arcs
- Construction also works with read arcs, but not with a strict definition of what is allowed in a distributed implementation

Conclusions

- New requirement for distributed implementations of concurrent systems
- Rules out proposed implementations that were intuitively unsatisfactory
- Distributed implementation of general Petri nets, preserving inevitability as well as completed ST-trace equivalence, extending Petri Nets with listen arcs
- Construction also works with read arcs, but not with a strict definition of what is allowed in a distributed implementation

Further research

- Is such a result possible without read or listen arcs?
- Extend with probabilities
- Proof (im)possibility results for branching time equivalences

References

[Badouel,Caillaud,Darondeau 2002]: É. Badouel, B. Caillaud, P. Darondeau, 2002, *Distributing Finite Automata Through Petri Net Synthesis.* Formal Ascpects of Computing 13(6), pp. 447ff, doi:10.1007/s001650200022

[GGS 2008]: R.J. van Glabbeek, U. Goltz, J.-W. Schicke, 2008, *On Synchronous and Asynchronous Interaction in Distributed Systems*. In E. Ochmański & J. Tyszkiewicz, editors: MFCS 2008, LNCS 5162, Springer, pp. 16ff, doi:10.1007/978-3-540-85238-4_2.

[Schicke 2008]: J.-W. Schicke, 2009, Diplomarbeit, *Synchrony and Asynchrony in Petri nets.* TU Braunschweig

[Vogler 2002]: W. Vogler, *Efficiency of asynchronous systems, read arcs, and the MUTEX-problem*. Theoretical Computer Science 275(1-2), pp. 589ff, doi:10.1016/S0304-3975(01)00300-0

Acknowledgments

to Jens and Christiane Goltz for technical help in preparing slides