Complexity Results for Checking Distributed Implementability

Keijo Heljanko[1] Alin Ştefănescu[2]
speaking

[1]Helsinki University of Technology

[2]University of Stuttgart

7–June–2005

Implementability: The Sequential Case

Specification

Implementability: The Sequential Case

Specification

+

One Agent

Implementability: The Sequential Case


Specification

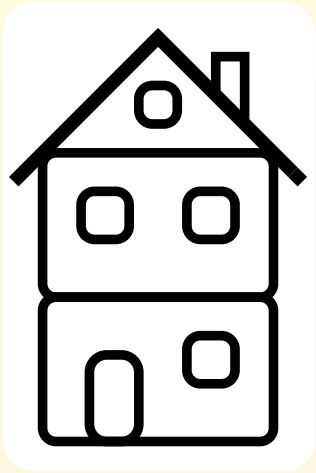
$+$


One Agent

$\overset{?}{\Rightarrow}$

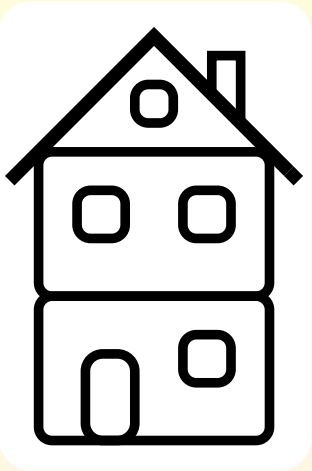

Implementation

Implementability: The Distributed Case



Specification

Implementability: The Distributed Case

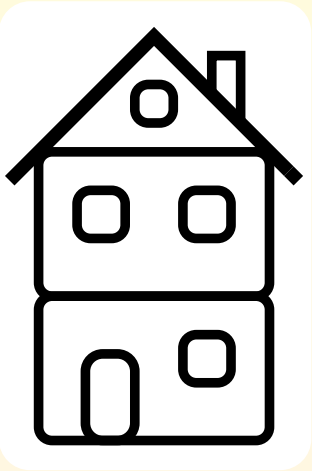


Specification

+

Team of
Communicating Agents

Implementability: The Distributed Case

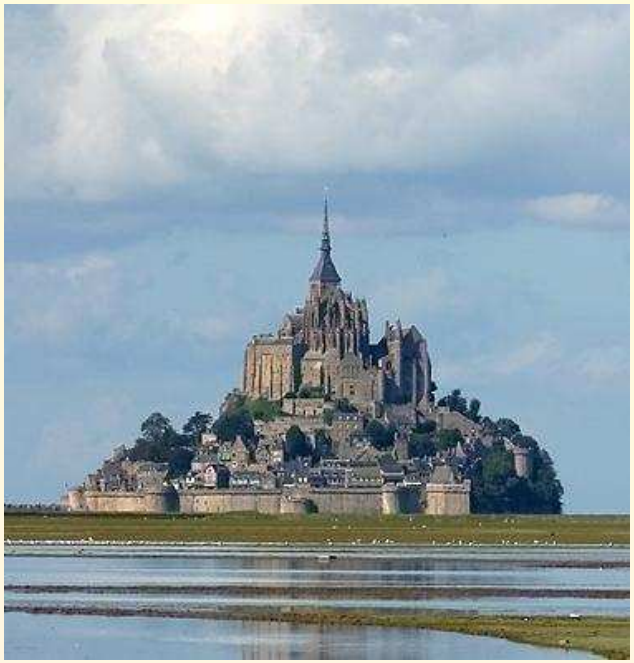


Specification

+

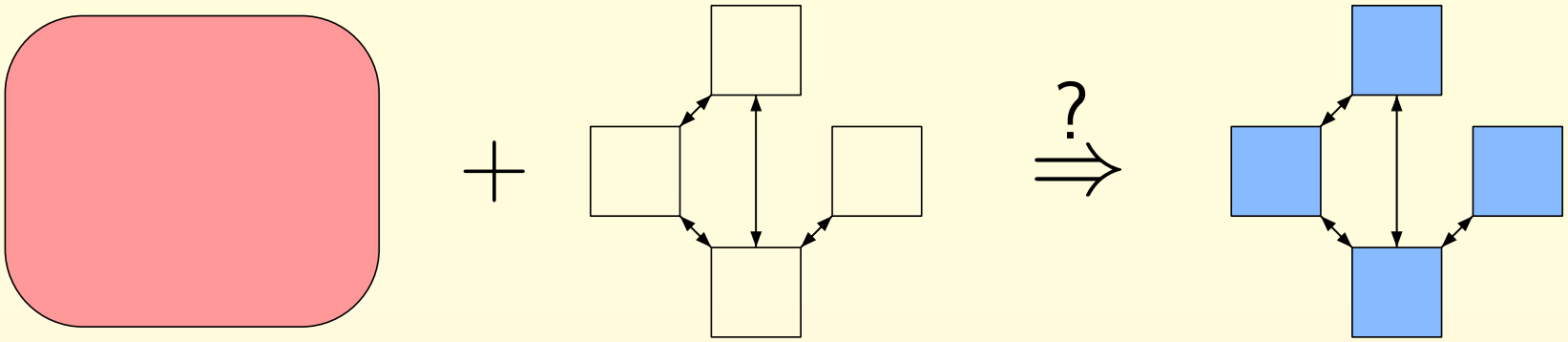
Team of
Communicating Agents

$\stackrel{?}{\Rightarrow}$



Distributed Implementation

The Problem



Transition System | Distribution | Distributed Transition System

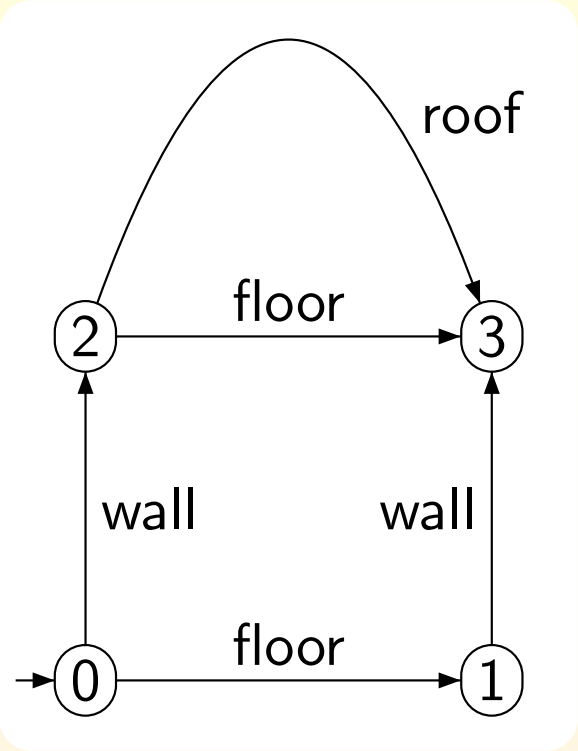
Distributed Implementability Problem

Instance: a labeled transition system TS and a **distribution** Δ of actions over a set of agents

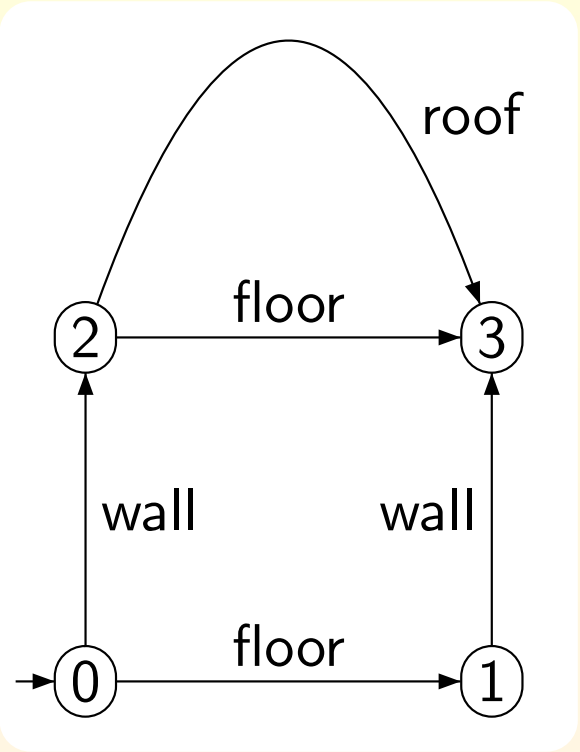
Question: Is there a distributed transition system over Δ whose global state space is equivalent to TS?

equivalent : graph-isomorphic / trace-equivalent / bisimilar

Building a House...



Building a House...



+

roof

floor

Agent 1

roof

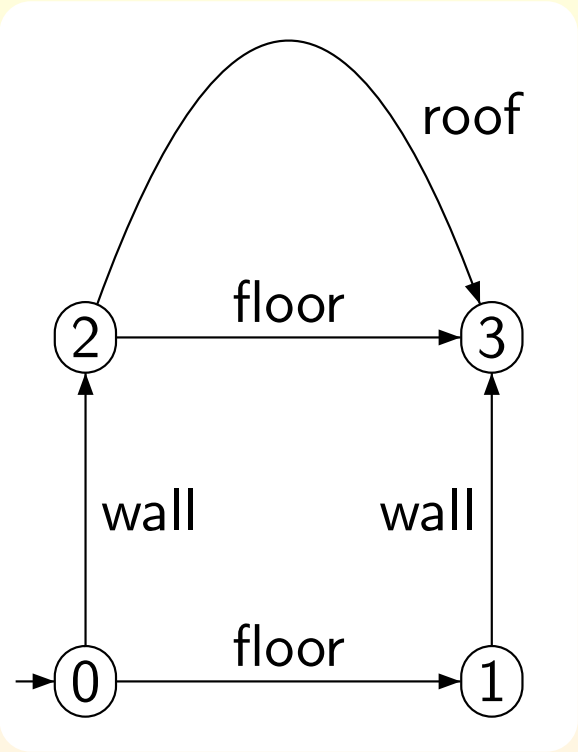
wall

Agent 2

Distribution of {floor,wall,roof} over {1,2}:

- $\Sigma_{local}(1)$={roof,floor}, $\Sigma_{local}(2)$={roof,wall}
- dom(roof)={1,2}, dom(floor)={1}, dom(wall)={2}

Building a House...



+

roof

floor

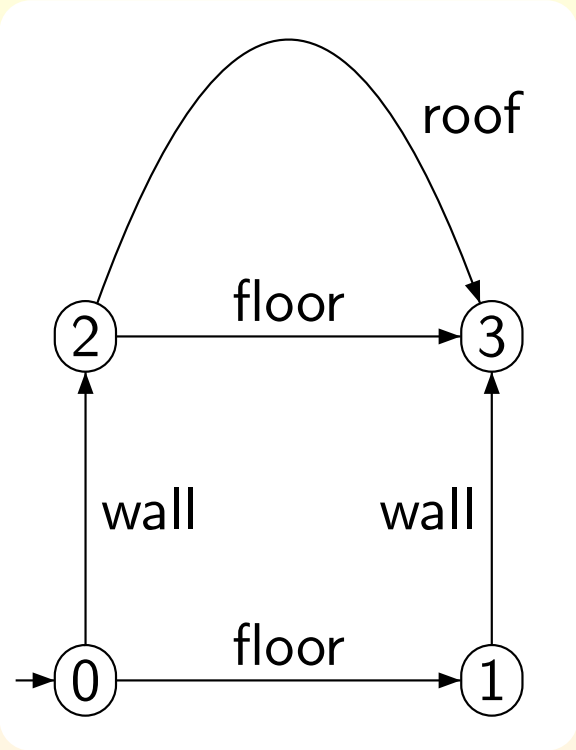
Agent 1

roof

wall

Agent 2

Building a House...



+

roof

floor

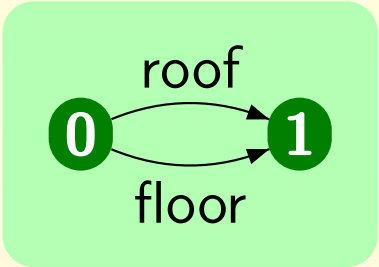
Agent 1

roof

wall

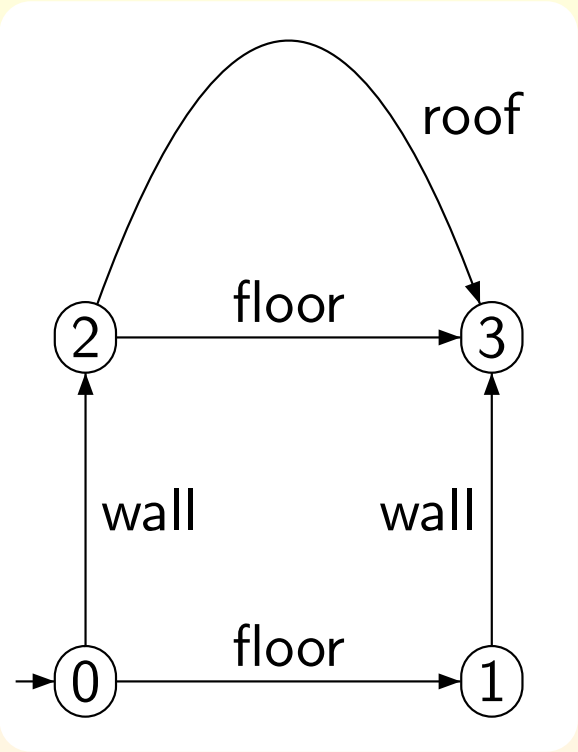
Agent 2

$\Rightarrow$



Agent 1

Building a House...



+

roof

floor

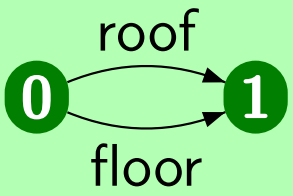
Agent 1

roof

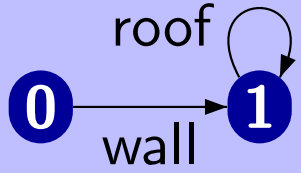
wall

Agent 2

⇒



Agent 1

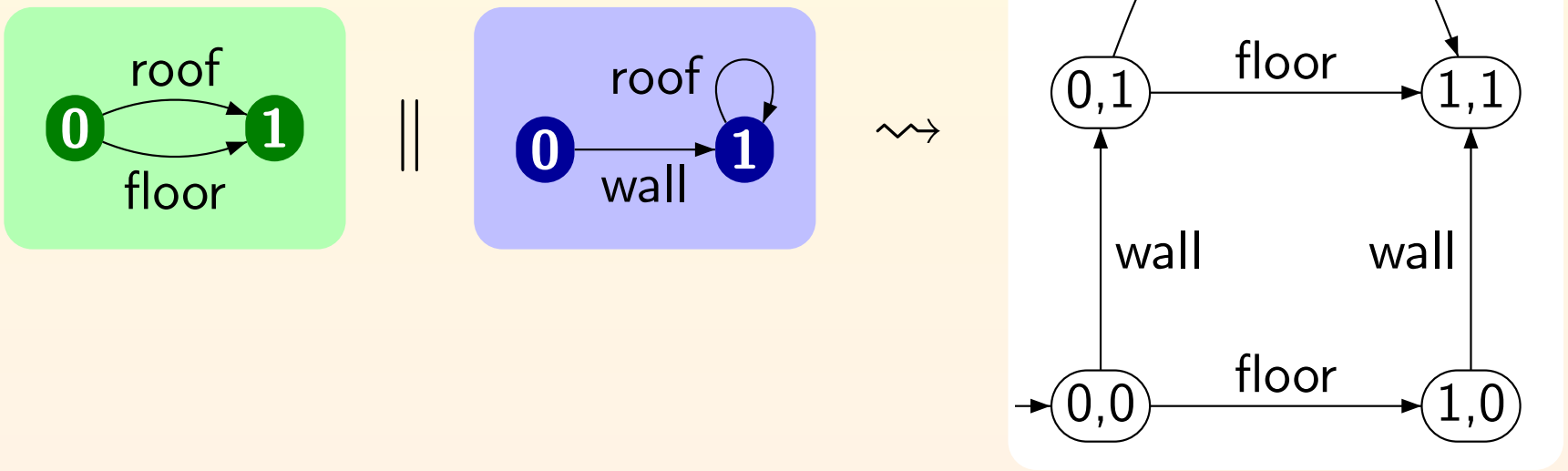


Agent 2

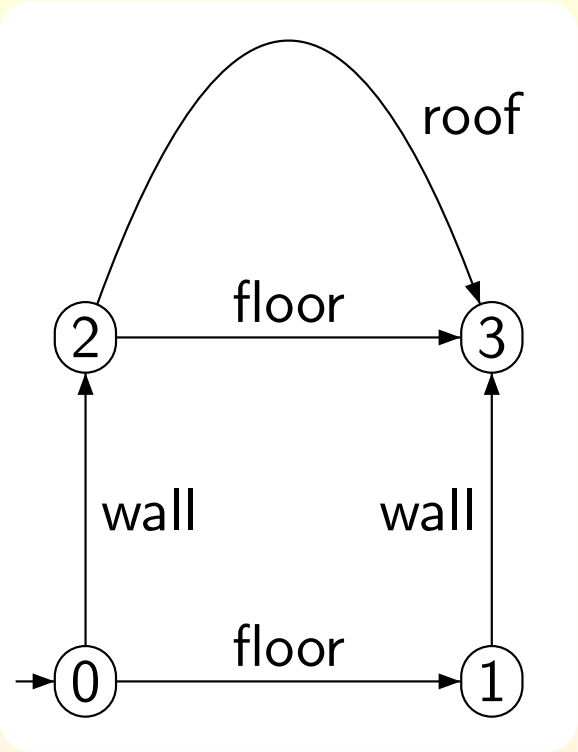
Synchronous Products of Transition Systems

A synchronous product of transition systems consists of a set of local transition systems synchronizing on common actions.

An action is executed if only if all local transition systems from its domain are able to execute that action.



Building a House...



+

roof

floor

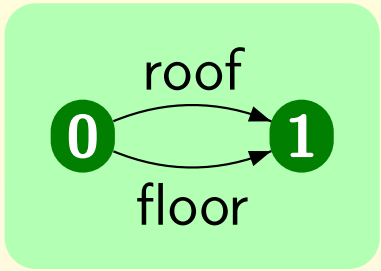
Agent 1

roof

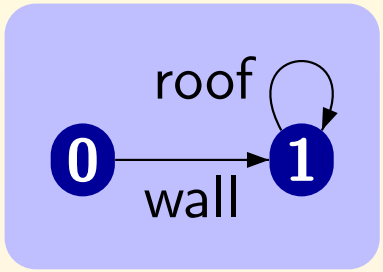
wall

Agent 2

$\Rightarrow$



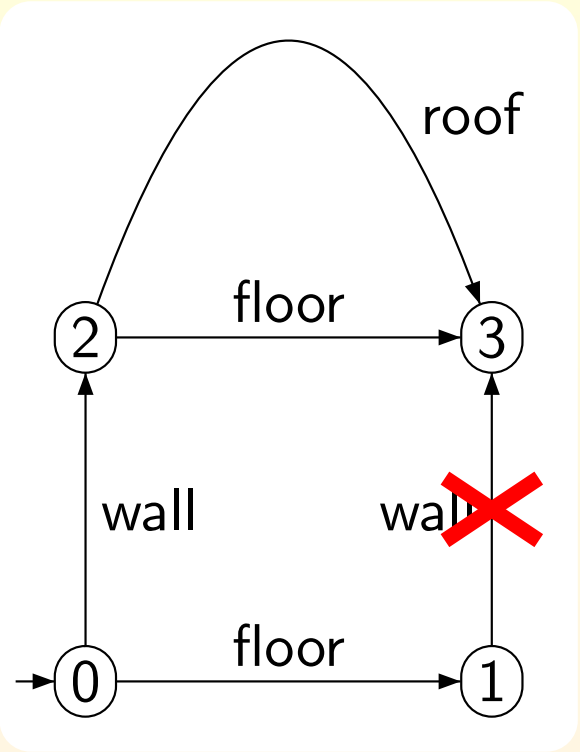
Agent 1



Agent 2

The specification is implementable!

Building a House... Not Always Possible!



+

roof

floor

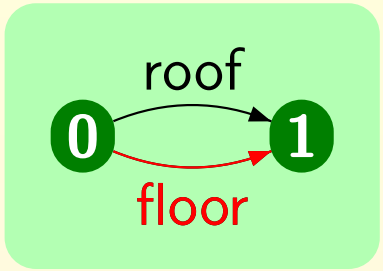
Agent 1

roof

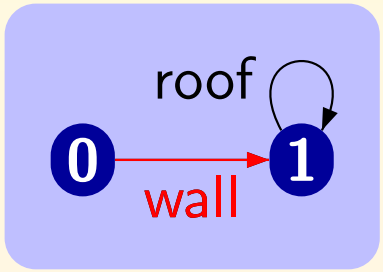
wall

Agent 2

⇒



Agent 1



Agent 2

When the edge (1,wall,3) is deleted,

the specification is no longer implementable!

Asynchronous Automata

Asynchronous automata [Zielonka87] generalize the synchronous products allowing more communication during synchronization.

An action is executed only for chosen tuples of local states of its domain.

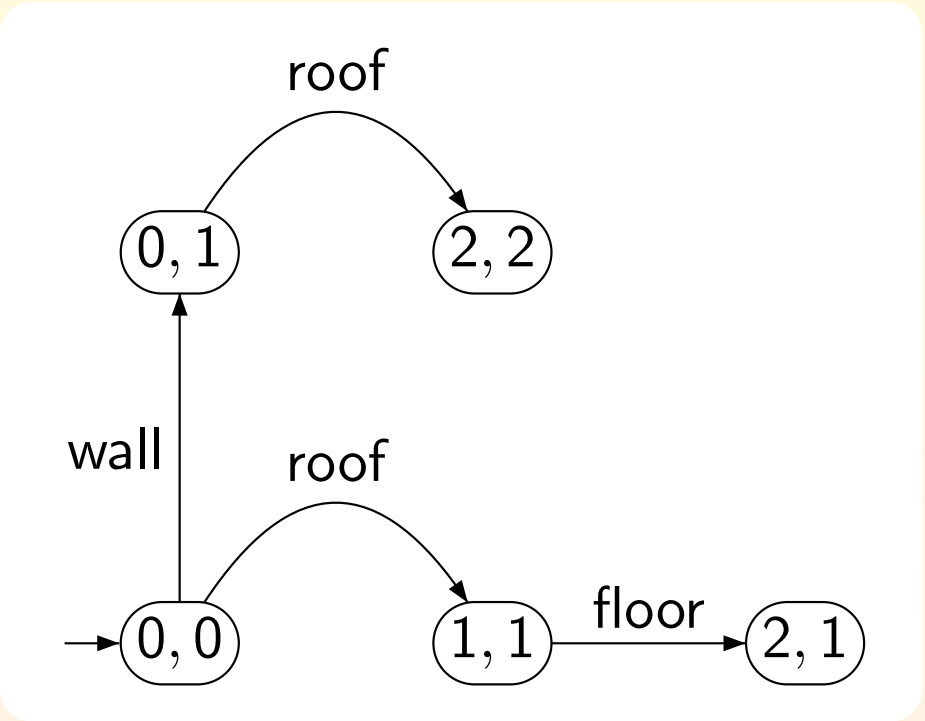
$$1 \rightarrow_{floor} 2$$

$$0 \rightarrow_{wall} 1$$

$$(0,0) \rightarrow_{roof} (1,1)$$

$$(0,1) \rightarrow_{roof} (2,2)$$

$\rightsquigarrow$



Asynchronous Automata

Asynchronous automata [Zielonka87] generalize the synchronous products allowing more communication during synchronization.

An action is executed only for chosen tuples of local states of its domain.

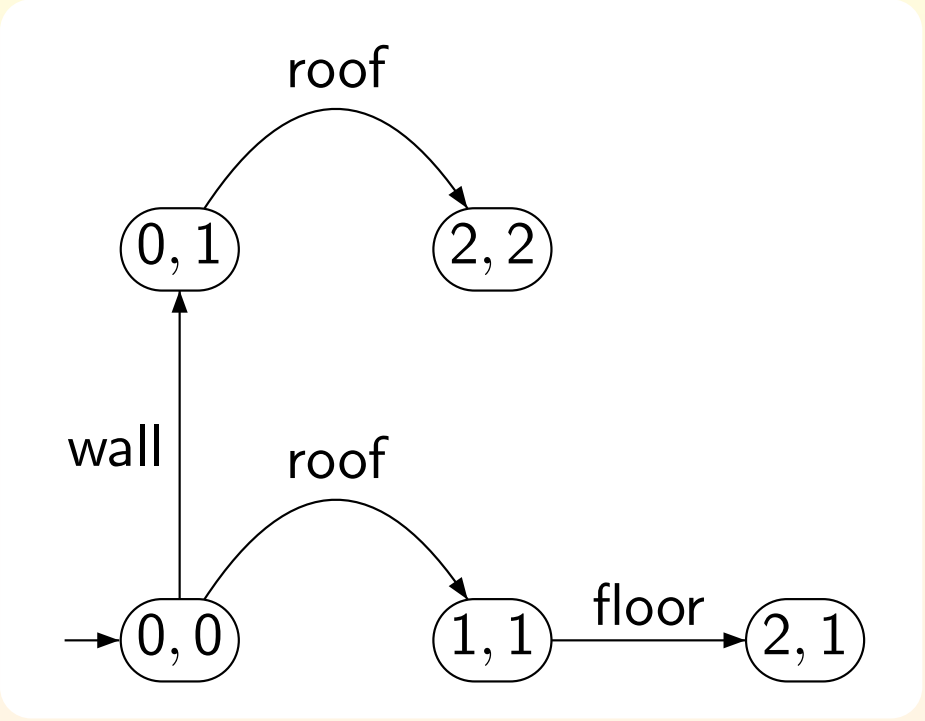
$$1 \rightarrow_{floor} 2$$

$$0 \rightarrow_{wall} 1$$

$$(0,0) \rightarrow_{roof} (1,1)$$

$$(0,1) \rightarrow_{roof} (2,2)$$

$\rightsquigarrow$



Not implementable as a synchronous product! (cf. wall roof floor)

Formally...

A synchronous product of transition systems $\mathcal{SP}$ over a distribution $(\Sigma, Proc, \Delta)$ consists of

- a set of *local* state spaces $(Q_p)_{p \in Proc}$ and
- a set of *local* transitions relations $(\rightarrow_p)_{p \in Proc}$ with $\rightarrow_p \subseteq Q_p \times \Sigma_{local}(p) \times Q_p$.

The global state space of $\mathcal{SP}$ consists of the global states $Q \subseteq \prod_{p \in Proc} Q_p$ *reachable* from a set of initial global states I by

$$(q_p)_{p \in Proc} \xrightarrow{a} (q'_p)_{p \in Proc} \Leftrightarrow \begin{cases} q_p \xrightarrow{a}_p q'_p & \text{for all } p \in dom(a) \\ q_p = q'_p & \text{for all } p \notin dom(a) \end{cases}$$

Formally...

An asynchronous automaton $\mathcal{AA}$ over a distribution $(\Sigma, Proc, \Delta)$ consists of

- a set of *local* state spaces $(Q_p)_{p \in Proc}$ and
- a set of *local* transition relations $(\rightarrow_a)_{a \in \Sigma}$ with $\rightarrow_a \subseteq \prod_{p \in dom(a)} Q_p \times \prod_{p \in dom(a)} Q_p$

The global state space of $\mathcal{AA}$ consists of the global states $Q \subseteq \prod_{p \in Proc} Q_p$ *reachable* from a set of initial global states I by

$$(q_p)_{p \in Proc} \xrightarrow{a} (q'_p)_{p \in Proc} \Leftrightarrow \begin{cases} (q_p)_{p \in dom(a)} \rightarrow_a (q'_p)_{p \in dom(a)} \text{ and} \\ q_p = q'_p \quad \text{for all } p \notin dom(a). \end{cases}$$

Contribution

The distributed implementability problem was studied for a variety of models. E.g. Petri nets, communicating finite state machines, synchronous products, asynchronous automata.

The computational complexity of many variants is known. [BadouelBernardinelloDarondeau95,97] etc.

For synchronous products and asynchronous automata, decision procedures were given, leading easily to upper bounds. [Morin98,99], [CastellaniMukundThiagarajan99], [Mukund02]

Our contribution was to fill some of the missing (lower) bounds.

Implementability modulo Isomorphism

Decision procedures for implementability modulo isomorphism for:

- deterministic specifications [Morin98] and
- nondeterministic specifications [CastellaniMukundThiagarajan99]

(inspired by the theory of regions [EhrenfeuchtRozenberg90])

The problem then solvable in

- deterministic polynomial time for deterministic specifications
- nondeterministic polynomial time for nondeterministic specifications

Lower bound for the nondeterministic case was left open

Implementability modulo Isomorphism

For both synchronous products and asynchronous automata holds:

Theorem

The implementability problem modulo isomorphism for nondeterministic specifications is NP-complete.

Proof idea:

- The problem is *in NP*:
  follows immediately from the characterization from [CMT99]
- The problem is *NP-hard*:
  by reduction from propositional satisfiability SAT
  (the proof is rather technical)

The result holds even for acyclic specifications!

Implementability modulo Trace Equivalence ($\mathcal{SP}$)

To decide whether a specification TS is trace-equivalent to a synchronous product with one initial global state do [CMT99]:

- project TS on the *local alphabets* $\Sigma_{local}(p)$
- synchronize the projections on common actions
- if the synchronization is trace-equivalent to the initial TS, answer 'yes', otherwise 'no'

Theorem [SHRS96]

Checking trace-equivalence of two synchronous products is PSPACE-complete.

Implementability modulo Trace Equivalence ($\mathcal{SP}$)

Theorem

The implementability problem modulo trace equivalence for synchronous products with one initial state is PSPACE-complete.

Proof idea:

- The problem is *in PSPACE*:
  follows immediately from the previous theorem from [SHRS96]
- The problem is *PSPACE-hard*:
  by reduction from nonreachability problem in synchronous products (proved PSPACE-hard in [SHRS96])

The result holds even for deterministic specifications!

For multiple initial states we have only a PSPACE-hardness result

Implementability modulo Trace Equivalence ($\mathcal{AA}$)

Two actions a, b are independent iff $dom(a) \cap dom(b) = \emptyset$

To decide whether a specification TS is trace-equivalent to an asynchronous automaton, check if $Trace(TS)$ is closed under commutation of adjacent independent actions [Zielonka87]

Theorem

The implementability problem modulo trace equivalence for asynchronous automata is PSPACE-complete.
Moreover, it is decidable in deterministic polynomial time for deterministic specifications.

Proof: easy adaptation to the prefix-closure case of techniques from [Muscholl94,PeledWilkeWolper98]

Complexity Bounds Overview

Synchronous products (with one global initial state)

Specification (TS)	Isomorphism	Trace Equivalence
Nondeterministic	NP-complete	PSPACE-complete
Deterministic	P [Mor98]	

Asynchronous automata (with multiple global initial states)

Specification (TS)	Isomorphism	Trace Equivalence
Nondeterministic	NP-complete	PSPACE-complete
Deterministic	P [Mor98]	P

Complexity Bounds Overview (Bisimulation)

Synchronous products (with one global initial state)

Specification (TS)	Isomorphism	Trace Equivalence	Bisim. (determ. impl.)
Nondeterministic	NP-complete	PSPACE-complete	PSPACE-complete
Deterministic	P [Mor98]		

Asynchronous automata (with multiple global initial states)

Specification (TS)	Isomorphism	Trace Equivalence	Bisim. (determ. impl.)
Nondeterministic	NP-complete	PSPACE-complete	P
Deterministic	P [Mor98]	P	

Complexity Bounds Overview (Acyclicity)

Synchronous products (with one global initial state)

Specification (TS)	Isomorphism	Trace Equivalence	Bisim. (determ. impl.)
Nondeterministic Deterministic	NP-complete P [Mor98]	PSPACE-complete	PSPACE-complete
Acyclic & Nondet. Acyclic & Determ.	NP-complete P [Mor98]	coNP-complete	coNP-complete

Asynchronous automata (with multiple global initial states)

Specification (TS)	Isomorphism	Trace Equivalence	Bisim. (determ. impl.)
Nondeterministic Deterministic	NP-complete P [Mor98]	PSPACE-complete P	P
Acyclic & Nondet. Acyclic & Determ.	NP-complete P [Mor98]	coNP-complete P	P

Final Remarks

We studied the complexity of the implementability for synchronous products ($\mathcal{SP}$) and asynchronous automata ($\mathcal{AA}$)

The complexities are similar, with an advantage of $\mathcal{AA}$ over $\mathcal{SP}$ for deterministic specifications.
Moreover, more models will be implementable as $\mathcal{AA}$ than as $\mathcal{SP}$.

However, the size of $\mathcal{SP}$ is linear in the size of the specification, while a state space explosion may occur for $\mathcal{AA}$
(cf. the superexponential bound of Zielonka's construction)

Appendix

Characterization of Async. Automata modulo Isomorphism

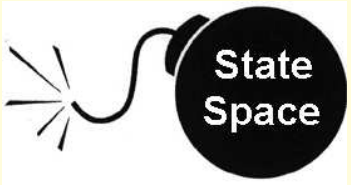
Theorem (Morin99)

Let $(\Sigma, Proc, \Delta)$ be a distribution and $TS = (Q, \Sigma, \rightarrow, I)$ be a transition system. Then, TS is isomorphic to an asynchronous automaton over Δ if and only if for each $p \in Proc$ there exists an equivalence relation $\equiv_p \subseteq Q \times Q$ with:

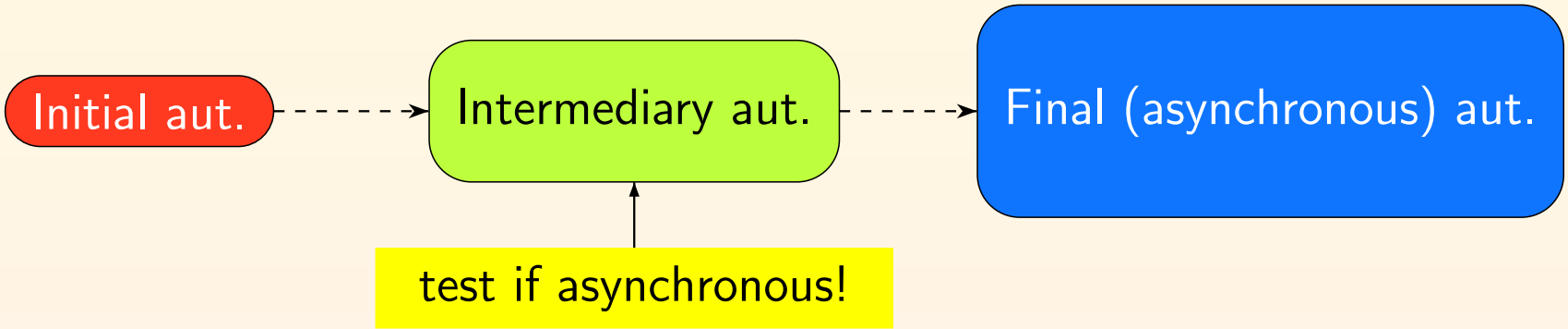
- AA_1: *If $q_1 \xrightarrow{a} q_2$, then $q_1 \equiv_{Proc \setminus dom(a)} q_2$.*
- AA_2: *If $q_1 \equiv_{Proc} q_2$, then $q_1 = q_2$.*
- AA_3: *If $q_1 \xrightarrow{a} q_1'$ and $q_1 \equiv_{dom(a)} q_2$, then there exists q_2' such that $q_2 \xrightarrow{a} q_2'$ and $q_1' \equiv_{dom(a)} q_2'$.*

Heuristic for smaller state space

- Zielonka's procedure outputs very large asynchronous automata



- Usually smaller asynchronous automata accepting the same language exist
- **Heuristic idea** [StefanescuEsparzaMuscholl03] Unfold the initial transition system guided by Zielonka's construction and test if intermediary automata are already asynchronous:



Synthesis Flow – the whole truth