

Distributed Synthesis for Regular and Contextfree Specifications

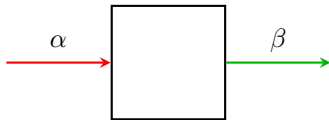
Wladimir Fridman and Bernd Puchala

RWTH Aachen University

August 22nd, 2011

MFCS 2011
Warsaw, Poland

Church's Synthesis Problem

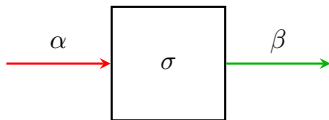


$$\alpha \in (\Sigma_I)^\omega$$

$$\beta \in (\Sigma_O)^\omega$$

$$L \subseteq (\Sigma_I \times \Sigma_O)^\omega$$

Church's Synthesis Problem



$$\alpha \in (\Sigma_I)^\omega$$

$$\beta \in (\Sigma_O)^\omega$$

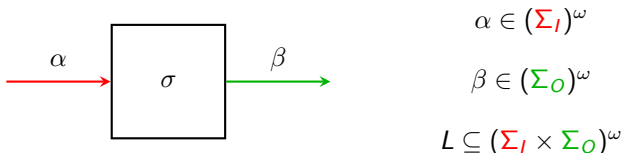
$$L \subseteq (\Sigma_I \times \Sigma_O)^\omega$$

Church's Synthesis Problem

Given: Regular specification (winning condition L)

Question: Is there a finite state implementation (strategy $\sigma : \Sigma_I^* \rightarrow \Sigma_O$) such that any resulting behavior satisfies the specification (σ is winning)

Church's Synthesis Problem



Church's Synthesis Problem

Given: Regular specification (winning condition L)

Question: Is there a finite state implementation (strategy $\sigma : \Sigma_I^* \rightarrow \Sigma_O$) such that any resulting behavior satisfies the specification (σ is winning)

Theorem (Büchi, Landweber 1969)

Given a regular winning condition L , one can decide whether there exists a winning strategy σ . If so, a finite state winning strategy can be constructed effectively.

Extensions

Specifications

Systems

Extensions

Specifications

- Branching Time
- Contextfree
- ...

Systems

Specifications

- Branching Time
- Contextfree
- ...

Systems

- Stochastic
- Distributed (Several Components)
- ...

Extensions

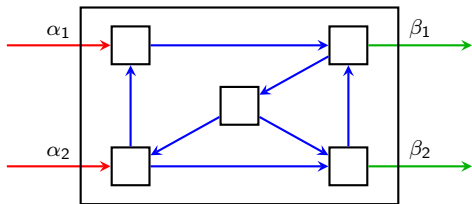
Specifications

- Branching Time
- **Contextfree**
- ...

Systems

- Stochastic
- **Distributed (Several Components)**
- ...

Distributed Systems

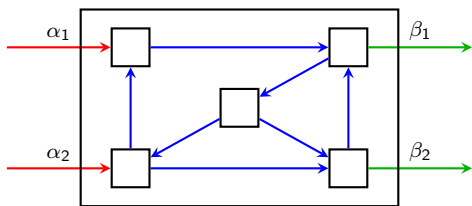


$$\alpha_1 \times \alpha_2 \in (\Sigma_I)^\omega$$

$$\beta_1 \times \beta_2 \in (\Sigma_O)^\omega$$

$$L \subseteq (\Sigma_I \times \Sigma \times \Sigma_O)^\omega$$

Distributed Systems



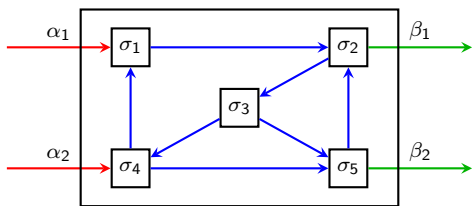
$$\alpha_1 \times \alpha_2 \in (\Sigma_I)^\omega$$

$$\beta_1 \times \beta_2 \in (\Sigma_O)^\omega$$

$$L \subseteq (\Sigma_I \times \Sigma \times \Sigma_O)^\omega$$

We assume that each process has at least one input channel and one output channel

Distributed Systems



$$\alpha_1 \times \alpha_2 \in (\Sigma_I)^\omega$$

$$\beta_1 \times \beta_2 \in (\Sigma_O)^\omega$$

$$L \subseteq (\Sigma_I \times \Sigma \times \Sigma_O)^\omega$$

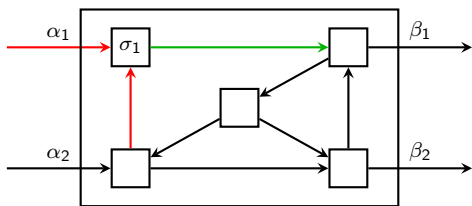
We assume that each process has at least one input channel and one output channel

Distributed Synthesis Problem

Given: Architecture \mathcal{A} , specification L

Question: Is there a distributed winning strategy $(\sigma_1, \dots, \sigma_n)$?

Distributed Systems



$$\alpha_1 \times \alpha_2 \in (\Sigma_I)^\omega$$

$$\beta_1 \times \beta_2 \in (\Sigma_O)^\omega$$

$$L \subseteq (\Sigma_I \times \Sigma \times \Sigma_O)^\omega$$

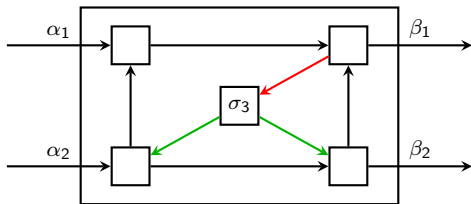
We assume that each process has at least one input channel and one output channel

Distributed Synthesis Problem

Given: Architecture \mathcal{A} , specification L

Question: Is there a distributed winning strategy $(\sigma_1, \dots, \sigma_n)$?

Distributed Systems



$$\alpha_1 \times \alpha_2 \in (\Sigma_I)^\omega$$

$$\beta_1 \times \beta_2 \in (\Sigma_O)^\omega$$

$$L \subseteq (\Sigma_I \times \Sigma \times \Sigma_O)^\omega$$

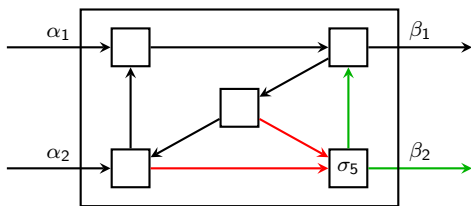
We assume that each process has at least one input channel and one output channel

Distributed Synthesis Problem

Given: Architecture \mathcal{A} , specification L

Question: Is there a distributed winning strategy $(\sigma_1, \dots, \sigma_n)$?

Distributed Systems



$$\alpha_1 \times \alpha_2 \in (\Sigma_I)^\omega$$

$$\beta_1 \times \beta_2 \in (\Sigma_O)^\omega$$

$$L \subseteq (\Sigma_I \times \Sigma \times \Sigma_O)^\omega$$

We assume that each process has at least one input channel and one output channel

Distributed Synthesis Problem

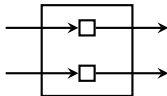
Given: Architecture \mathcal{A} , specification L

Question: Is there a distributed winning strategy $(\sigma_1, \dots, \sigma_n)$?

Distributed Synthesis

Theorem (Pnueli, Rosner 1990)

Distributed synthesis is undecidable for LTL specifications.

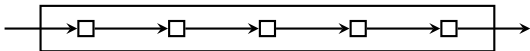
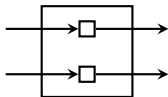


Distributed Synthesis

Theorem (Pnueli, Rosner 1990)

Distributed synthesis is undecidable for LTL specifications.

For pipelines, distributed synthesis is decidable for LTL specifications.

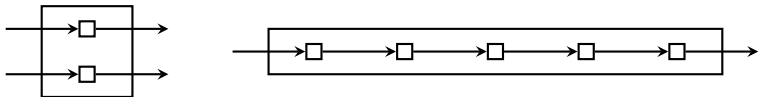


Distributed Synthesis

Theorem (Pnueli, Rosner 1990)

Distributed synthesis is undecidable for LTL specifications.

For pipelines, distributed synthesis is decidable for LTL specifications.



Theorem (Finkbeiner, Schewe 2005)

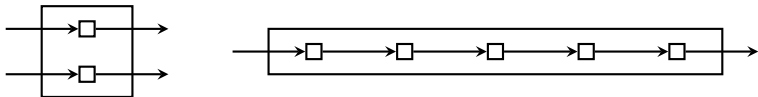
Distributed synthesis is decidable for LTL specifications iff the architecture does not contain an information fork.

Distributed Synthesis

Theorem (Pnueli, Rosner 1990)

Distributed synthesis is undecidable for LTL specifications.

For pipelines, distributed synthesis is decidable for LTL specifications.

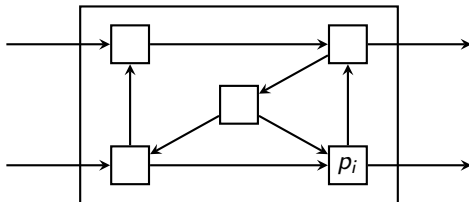


Theorem (Finkbeiner, Schewe 2005)

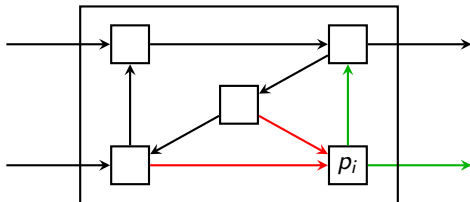
Distributed synthesis is decidable for LTL specifications iff the architecture does not contain an information fork.



Local Specifications

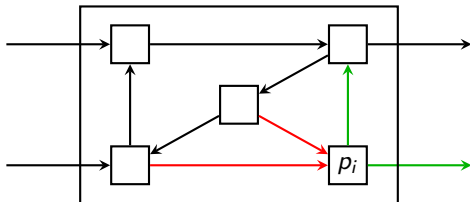


Local Specifications



- Local Specification $L_i \subseteq \Sigma_i^i \times \Sigma_o^i$ for process p_i

Local Specifications



- Local Specification $L_i \subseteq \Sigma_i^i \times \Sigma_o^i$ for process p_i
- Global System Specification $L = \bigcap L_i$

Synthesis for Local Specifications

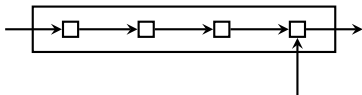
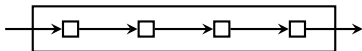
Theorem (Madhusudan, Thiagarajan 2001)

*Restricted to the class of acyclic architectures:
distributed synthesis is decidable for local LTL specifications iff
each connected component is a pipeline or a two-flanked pipeline.*

Synthesis for Local Specifications

Theorem (Madhusudan, Thiagarajan 2001)

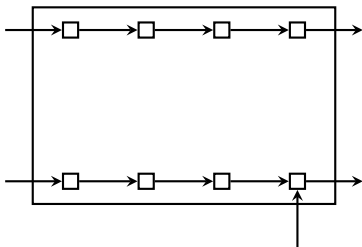
*Restricted to the class of acyclic architectures:
distributed synthesis is decidable for local LTL specifications iff
each connected component is a pipeline or a two-flanked pipeline.*



Synthesis for Local Specifications

Theorem (Madhusudan, Thiagarajan 2001)

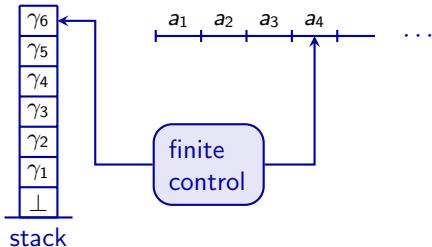
*Restricted to the class of acyclic architectures:
distributed synthesis is decidable for local LTL specifications iff
each connected component is a pipeline or a two-flanked pipeline.*



Contextfree Specifications

Pushdown ω -Automaton $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_{in}, \perp, col)$

- Q finite set of states, q_{in} initial state
- Σ input alphabet
- Γ stack alphabet
- $\delta: Q \times (\Gamma \cup \{\perp\}) \times (\Sigma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q \times \Gamma^*)$ transition function
- $col: Q \rightarrow \mathbb{N}$ coloring function



Theorem (Walukiewicz 1996)

Given a deterministic contextfree specification L , one can decide whether there exists a winning strategy σ . If so, a pushdown winning strategy can be constructed effectively.

Remark

For nondeterministic contextfree specifications, the synthesis problem is undecidable.

Theorem (Walukiewicz 1996)

Given a deterministic contextfree specification L , one can decide whether there exists a winning strategy σ . If so, a pushdown winning strategy can be constructed effectively.

Remark

For nondeterministic contextfree specifications, the synthesis problem is undecidable.

Theorem

The distributed synthesis problem for global deterministic contextfree specifications is undecidable for any architecture with at least two processes and at least one external input channel.

Extensions:

- Deterministic Contextfree Specifications
- Cyclic Architectures

Main Result

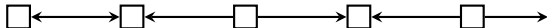
Characterization of Decidable Architectures

Theorem

Any architecture with at least two connected processes with DCF specifications is undecidable.

Theorem

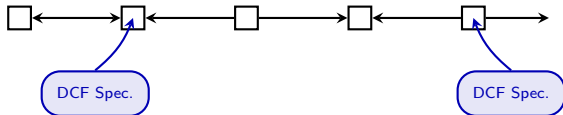
Any architecture with at least two connected processes with DCF specifications is undecidable.



Undecidable Architectures

Theorem

Any architecture with at least two connected processes with DCF specifications is undecidable.



Theorem

Any architecture with two connected processes $p \neq p'$ such that

- p is reachable and has a DCF specification*
- p' is not better informed than p*

is undecidable.

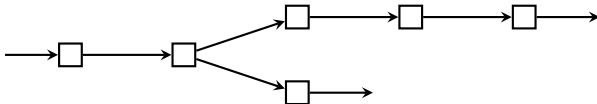
Undecidable Architectures

Theorem

Any architecture with two connected processes $p \neq p'$ such that

- p is reachable and has a DCF specification
- p' is not better informed than p

is undecidable.



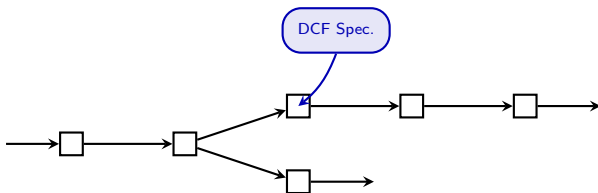
Undecidable Architectures

Theorem

Any architecture with two connected processes $p \neq p'$ such that

- p is reachable and has a DCF specification
- p' is not better informed than p

is undecidable.



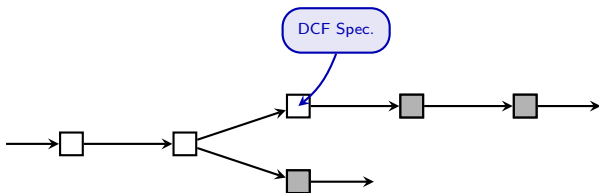
Undecidable Architectures

Theorem

Any architecture with two connected processes $p \neq p'$ such that

- p is reachable and has a DCF specification
- p' is not better informed than p

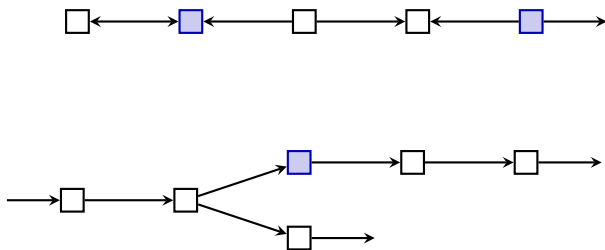
is undecidable.



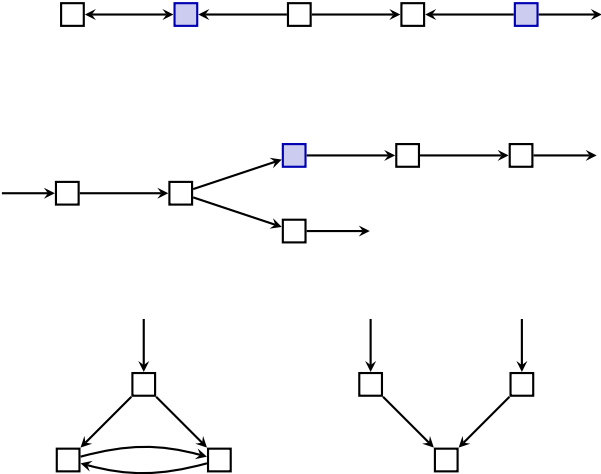
Undecidable Architectures



Undecidable Architectures



Undecidable Architectures



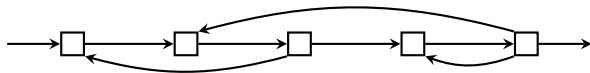
Theorem

Any pipeline with backwards-channels where at most the last process has a DCF specification is decidable.

Special Cases of Decidable Architectures

Theorem

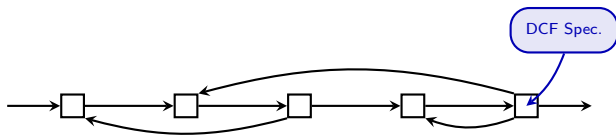
Any pipeline with backwards-channels where at most the last process has a DCF specification is decidable.



Special Cases of Decidable Architectures

Theorem

Any pipeline with backwards-channels where at most the last process has a DCF specification is decidable.



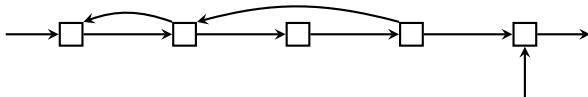
Theorem

Any two-flanked pipeline with backwards-channels is decidable for regular specifications if there is no backward-channel from the last process.

Special Cases of Decidable Architectures

Theorem

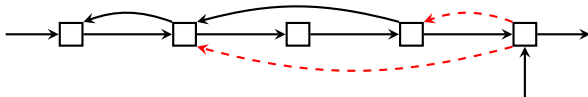
Any two-flanked pipeline with backwards-channels is decidable for regular specifications if there is no backward-channel from the last process.



Special Cases of Decidable Architectures

Theorem

Any two-flanked pipeline with backwards-channels is decidable for regular specifications if there is no backward-channel from the last process.



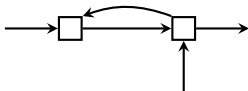
Theorem

Any two-flanked pipeline with backwards-channels consisting of two processes is decidable for regular specifications.

Special Cases of Decidable Architectures

Theorem

Any two-flanked pipeline with backwards-channels consisting of two processes is decidable for regular specifications.



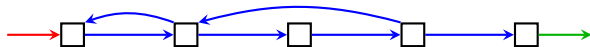
Special Cases of Decidable Architectures

Regular Global Specifications:



Special Cases of Decidable Architectures

Regular Global Specifications:



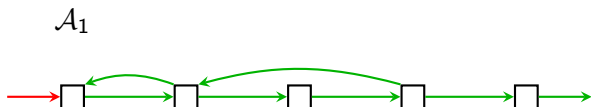
Special Cases of Decidable Architectures

Regular Global Specifications:



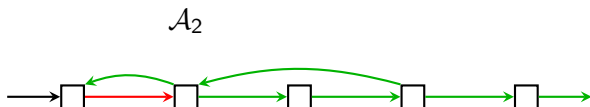
Special Cases of Decidable Architectures

Regular Global Specifications:



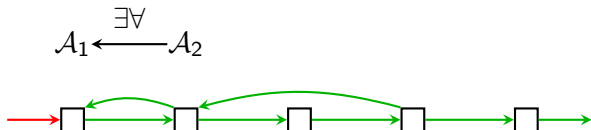
Special Cases of Decidable Architectures

Regular Global Specifications:



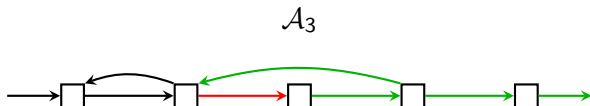
Special Cases of Decidable Architectures

Regular Global Specifications:



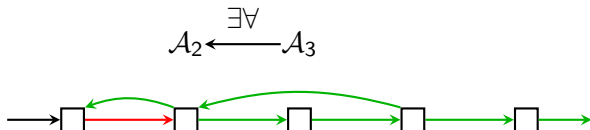
Special Cases of Decidable Architectures

Regular Global Specifications:



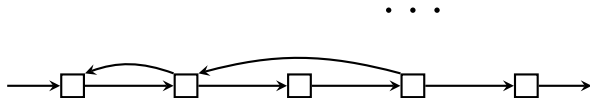
Special Cases of Decidable Architectures

Regular Global Specifications:



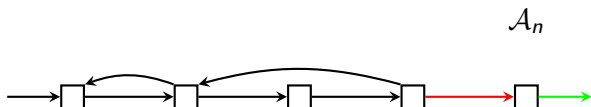
Special Cases of Decidable Architectures

Regular Global Specifications:



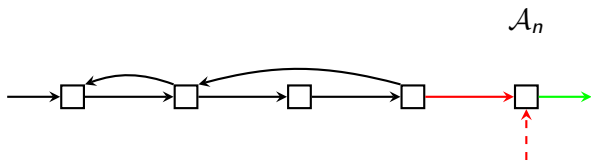
Special Cases of Decidable Architectures

Regular Global Specifications:



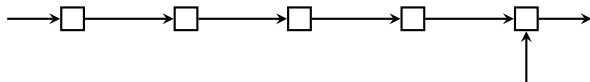
Special Cases of Decidable Architectures

Regular Global Specifications:



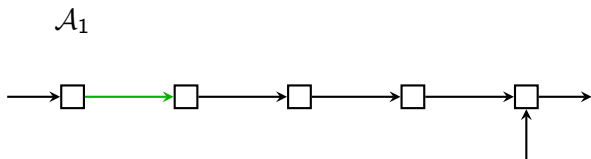
Special Cases of Decidable Architectures

Regular Local Specifications for **Acyclic** Architectures:



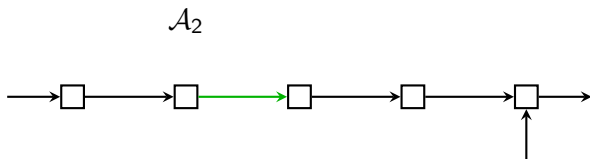
Special Cases of Decidable Architectures

Regular Local Specifications for **Acyclic** Architectures:



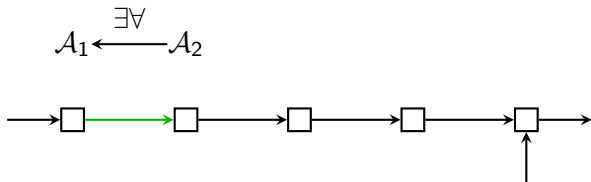
Special Cases of Decidable Architectures

Regular Local Specifications for **Acyclic** Architectures:



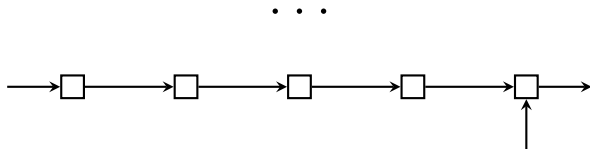
Special Cases of Decidable Architectures

Regular Local Specifications for **Acyclic** Architectures:



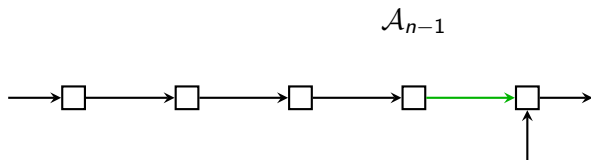
Special Cases of Decidable Architectures

Regular Local Specifications for **Acyclic** Architectures:



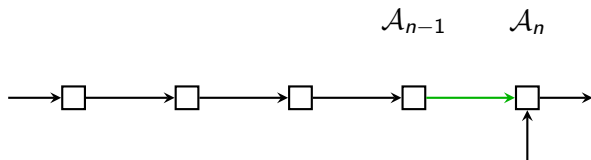
Special Cases of Decidable Architectures

Regular Local Specifications for **Acyclic** Architectures:



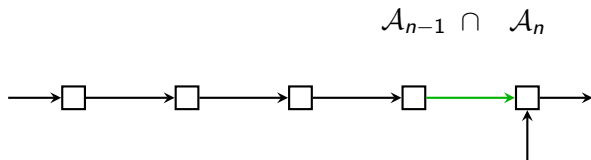
Special Cases of Decidable Architectures

Regular Local Specifications for **Acyclic** Architectures:



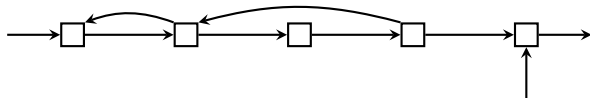
Special Cases of Decidable Architectures

Regular Local Specifications for **Acyclic** Architectures:



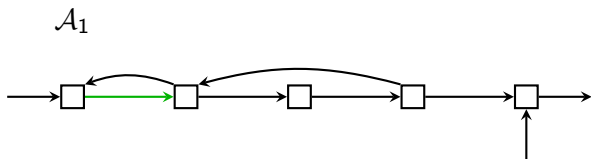
Special Cases of Decidable Architectures

Regular Local Specifications for **Cyclic** Architectures:



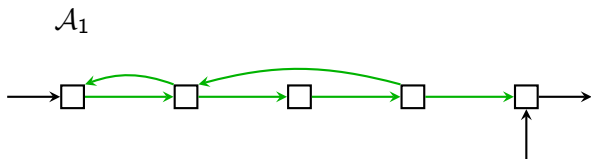
Special Cases of Decidable Architectures

Regular Local Specifications for **Cyclic** Architectures:



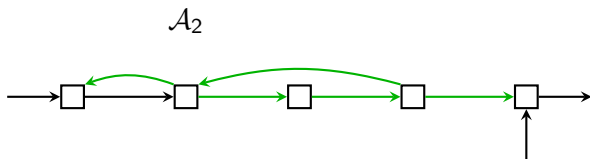
Special Cases of Decidable Architectures

Regular Local Specifications for **Cyclic** Architectures:



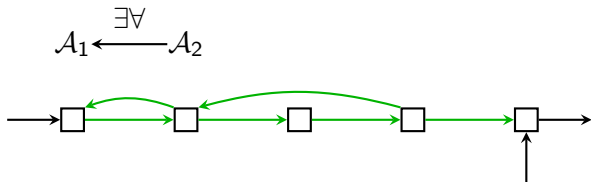
Special Cases of Decidable Architectures

Regular Local Specifications for **Cyclic** Architectures:



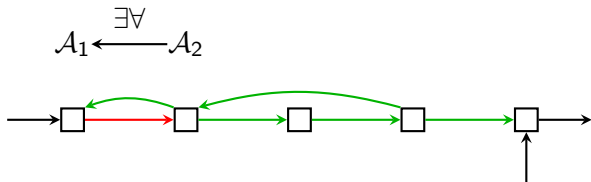
Special Cases of Decidable Architectures

Regular Local Specifications for **Cyclic** Architectures:



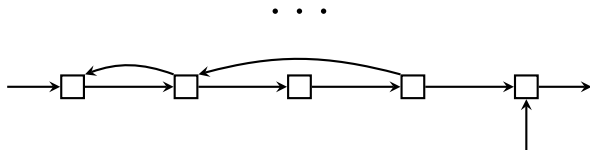
Special Cases of Decidable Architectures

Regular Local Specifications for **Cyclic** Architectures:



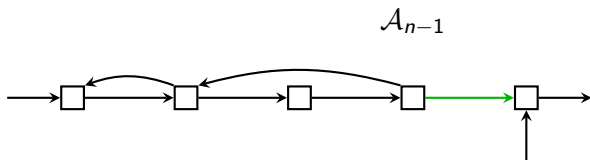
Special Cases of Decidable Architectures

Regular Local Specifications for **Cyclic** Architectures:



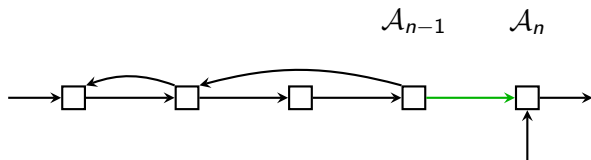
Special Cases of Decidable Architectures

Regular Local Specifications for **Cyclic** Architectures:



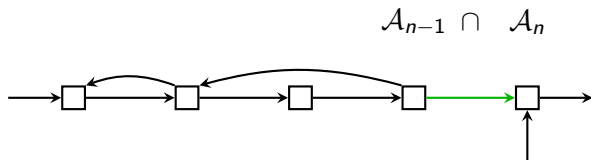
Special Cases of Decidable Architectures

Regular Local Specifications for **Cyclic** Architectures:



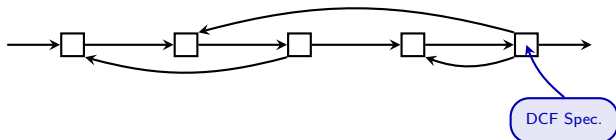
Special Cases of Decidable Architectures

Regular Local Specifications for **Cyclic** Architectures:



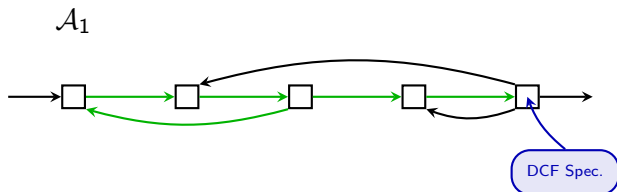
Special Cases of Decidable Architectures

Contextfree **Local** Specifications for **Cyclic** Architectures:



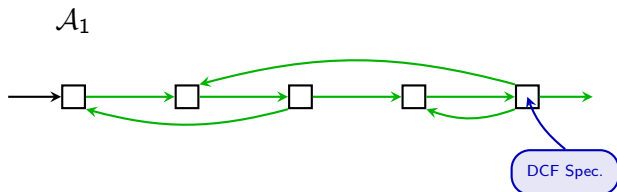
Special Cases of Decidable Architectures

Contextfree **Local** Specifications for **Cyclic** Architectures:



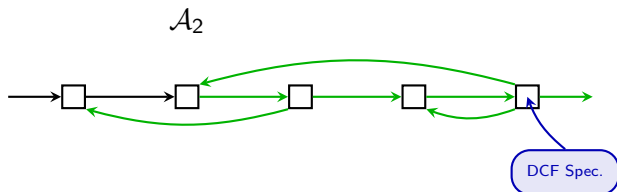
Special Cases of Decidable Architectures

Contextfree **Local** Specifications for **Cyclic** Architectures:



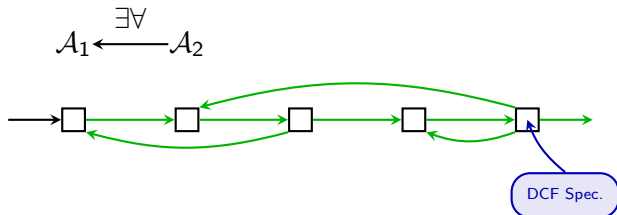
Special Cases of Decidable Architectures

Contextfree **Local** Specifications for **Cyclic** Architectures:



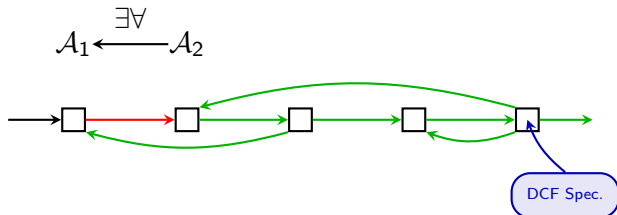
Special Cases of Decidable Architectures

Contextfree **Local** Specifications for **Cyclic** Architectures:



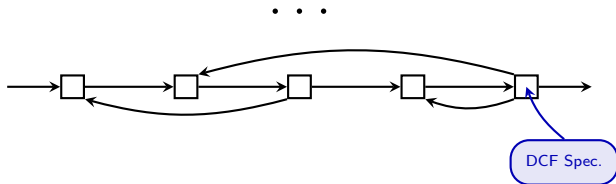
Special Cases of Decidable Architectures

Contextfree **Local** Specifications for **Cyclic** Architectures:



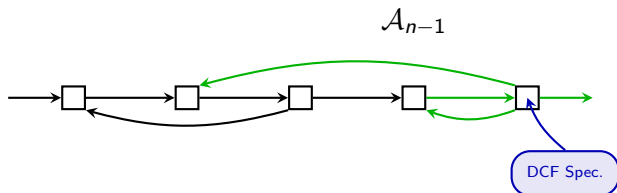
Special Cases of Decidable Architectures

Contextfree **Local** Specifications for **Cyclic** Architectures:



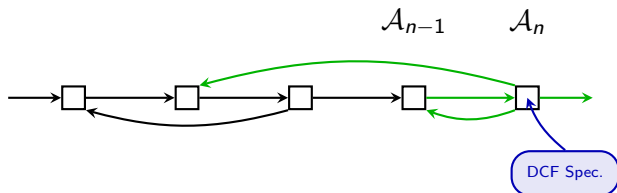
Special Cases of Decidable Architectures

Contextfree **Local** Specifications for **Cyclic** Architectures:



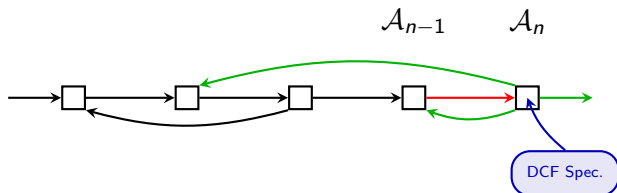
Special Cases of Decidable Architectures

Contextfree **Local** Specifications for **Cyclic** Architectures:



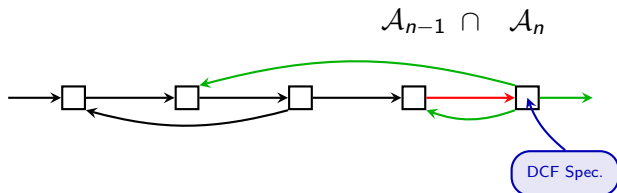
Special Cases of Decidable Architectures

Contextfree **Local** Specifications for **Cyclic** Architectures:

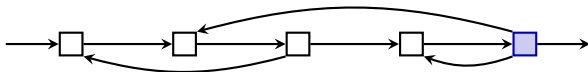


Special Cases of Decidable Architectures

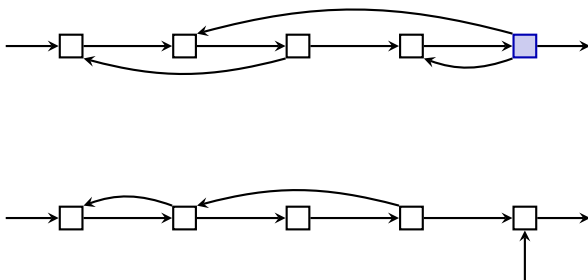
Contextfree **Local** Specifications for **Cyclic** Architectures:



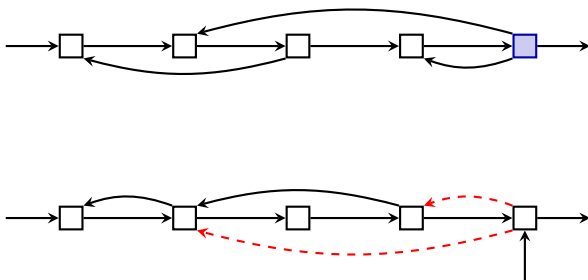
Special Cases of Decidable Architectures



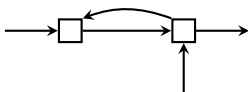
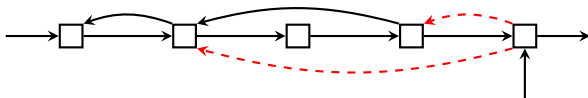
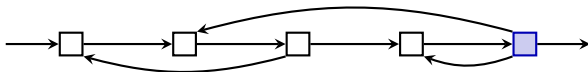
Special Cases of Decidable Architectures



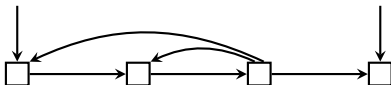
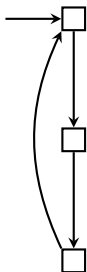
Special Cases of Decidable Architectures



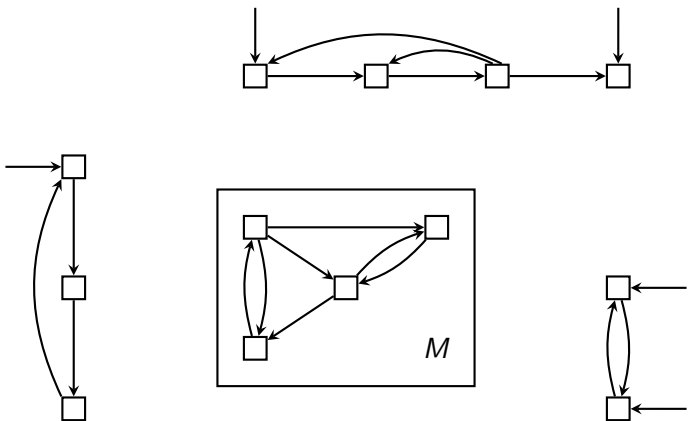
Special Cases of Decidable Architectures



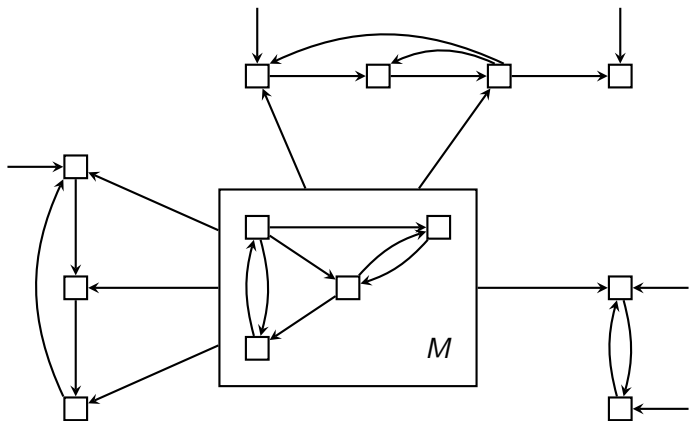
Characterization



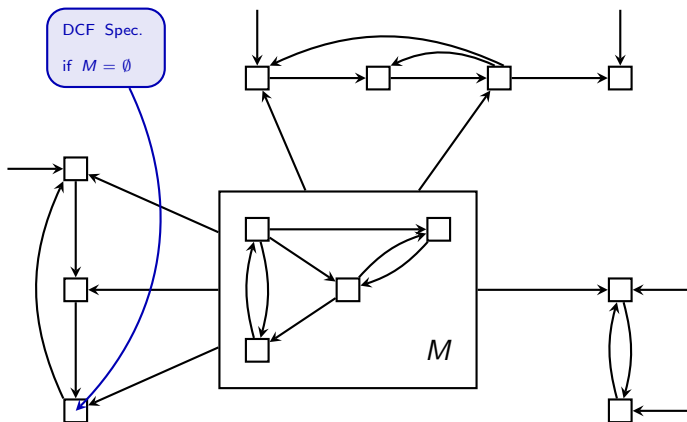
Characterization



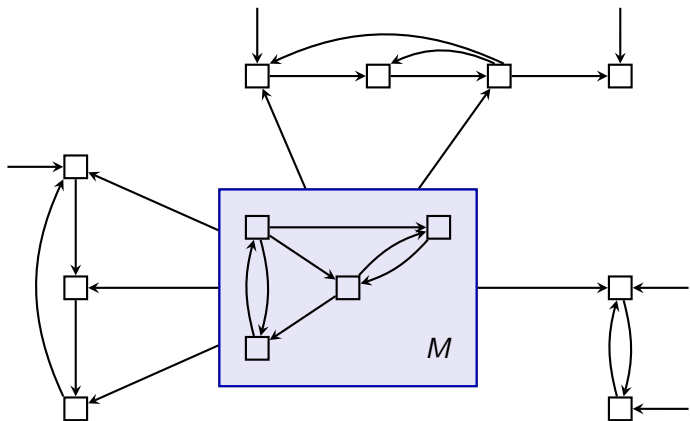
Characterization



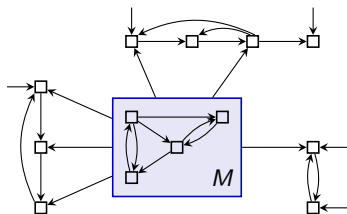
Characterization



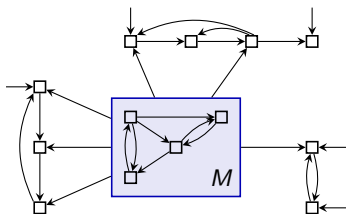
Characterization



Characterization

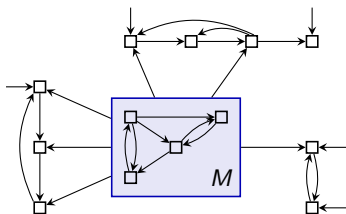


Characterization



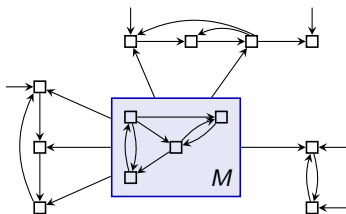
- Construct a deterministic ω -automaton \mathcal{B}_M for M

Characterization



- Construct a deterministic ω -automaton \mathcal{B}_M for M
- for each reachable subarchitecture construct
 - a nondeterministic tree automaton \mathcal{A}
 - a alternating ω -automaton \mathcal{B}
(running over the channels from M and simulating \mathcal{A})

Characterization



- Construct a deterministic ω -automaton \mathcal{B}_M for M
- for each reachable subarchitecture construct
 - a nondeterministic tree automaton \mathcal{A}
 - a alternating ω -automaton \mathcal{B}
(running over the channels from M and simulating \mathcal{A})
- Check whether $\bigcap L(\mathcal{B}) \cap L(\mathcal{B}_M) \neq \emptyset$