$$\delta : Q \times 2^Q \to Q$$

Distributed Automata and Logic

Fabian Reiter

7 February 2019 @ SIF Congress, Bordeaux

IRIF
INSTITUT
DE RECHERCHE
EN INFORMATIQUE
FONDAMENTALE

université
PARIS DIDEROT
PARIS 7

USPC
Université Sorbonne
Paris Cité

Fagin's theorem (1974)

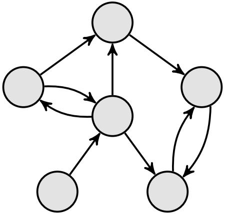
Fagin's theorem (1974)

Fagin's theorem (1974)

∃ SECOND-ORDER LOGIC

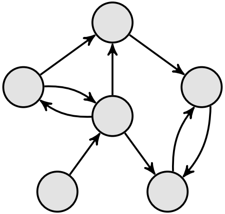
Fagin's theorem (1974)

∃ SECOND-ORDER LOGIC



Fagin's theorem (1974)

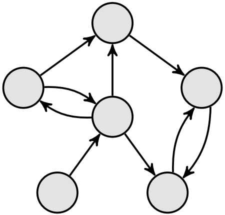
∃ SECOND-ORDER LOGIC



Example: Hamiltonian path

Fagin's theorem (1974)

∃ SECOND-ORDER LOGIC

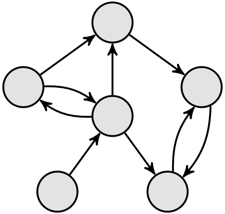


Example: Hamiltonian path

$\exists R \Big(\qquad \Big)$

Fagin's theorem (1974)

∃ SECOND-ORDER LOGIC



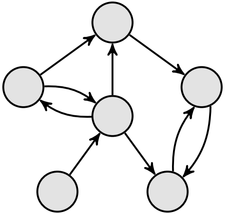
Example: Hamiltonian path

$\exists R \,($ "R is a strict total order" $\wedge$

$)$

Fagin's theorem (1974)

∃ SECOND-ORDER LOGIC

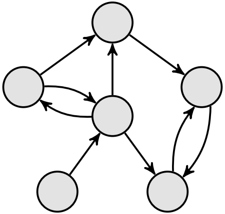


Example: Hamiltonian path

∃R ("R is a strict total order" ∧
"R-successors are adjacent")

Fagin's theorem (1974)

∃ SECOND-ORDER LOGIC



Example: Hamiltonian path

∃R ("R is a strict total order" ∧
"R-successors are adjacent")

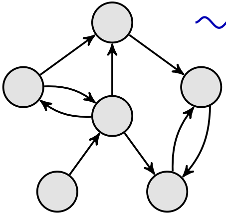
NP TURING MACHINES

Fagin's theorem (1974)

∃ SECOND-ORDER LOGIC

NP TURING MACHINES

encoding

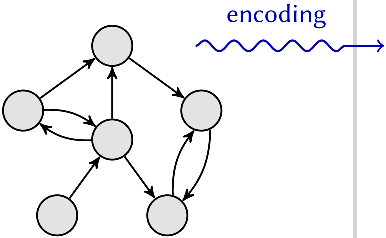


Example: Hamiltonian path

∃R ("R is a strict total order" ∧
"R-successors are adjacent")

Fagin's theorem (1974)

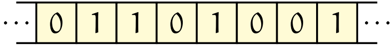
∃ SECOND-ORDER LOGIC



Example: Hamiltonian path

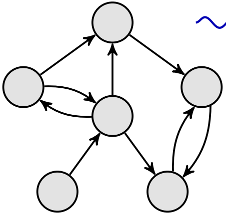
$\exists R\,\big($ "R is a strict total order" $\wedge$
"R-successors are adjacent" $\big)$

NP TURING MACHINES



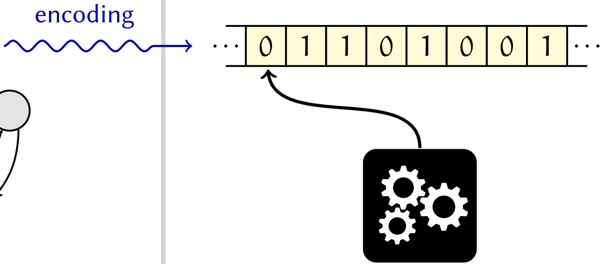
Fagin's theorem (1974)

∃ SECOND-ORDER LOGIC



Example: Hamiltonian path

∃R ("R is a strict total order" ∧
"R-successors are adjacent")

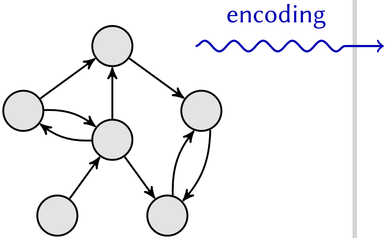


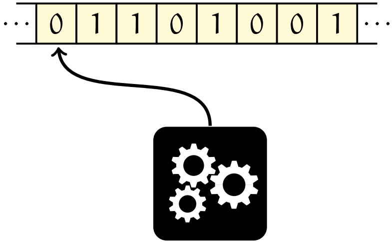
NP TURING MACHINES

Fagin's theorem (1974)

∃ SECOND-ORDER LOGIC

NP TURING MACHINES





Example: Hamiltonian path

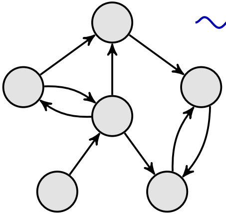
∃R ("R is a strict total order" ∧
"R-successors are adjacent")

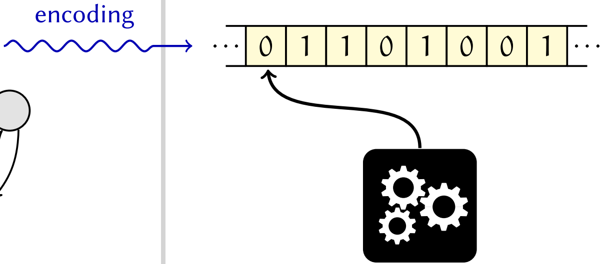
- Nondeterministic moves

Fagin's theorem (1974)

∃ SECOND-ORDER LOGIC

NP TURING MACHINES





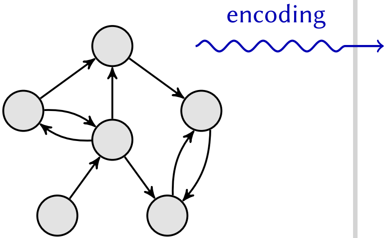
Example: Hamiltonian path

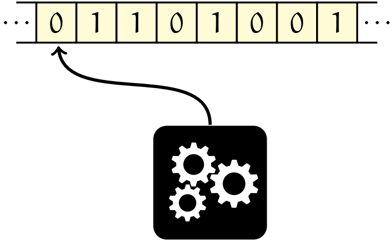
$\exists R\,\big($ "R is a strict total order" $\wedge$
"R-successors are adjacent" $\big)$

- Nondeterministic moves
- Polynomial running time

Fagin's theorem (1974)

∃ SECOND-ORDER LOGIC ⟺ EQUIVALENT ⟺ NP TURING MACHINES



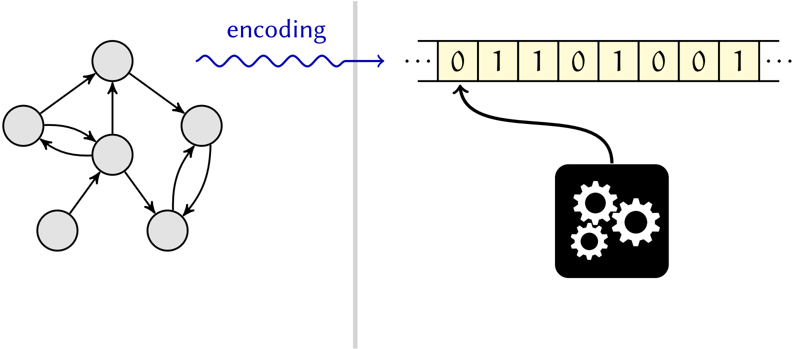


Example: Hamiltonian path

∃R ("R is a strict total order" ∧
"R-successors are adjacent")

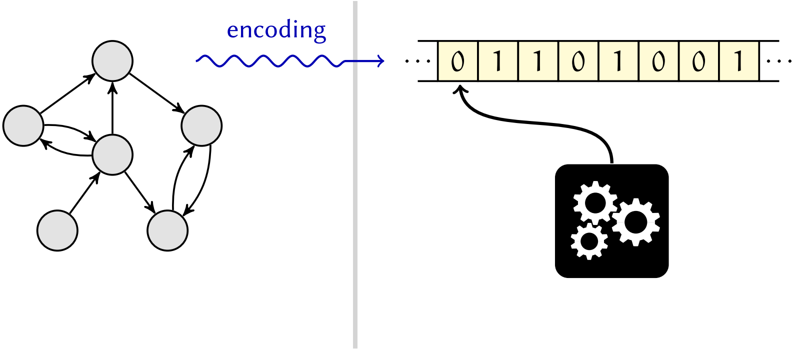
- Nondeterministic moves
- Polynomial running time

Descriptive complexity



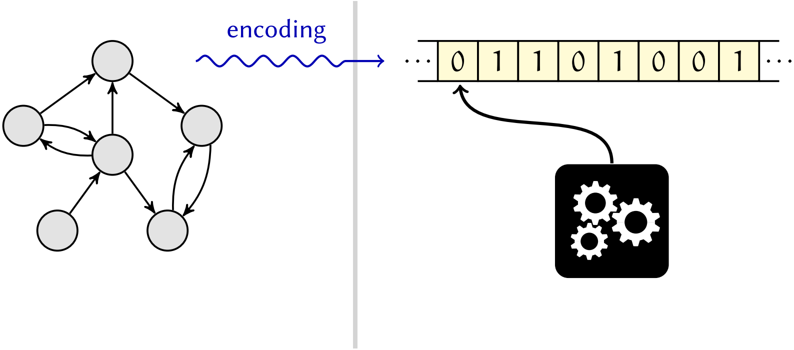
Descriptive complexity

SOME LOGICAL FORMALISM

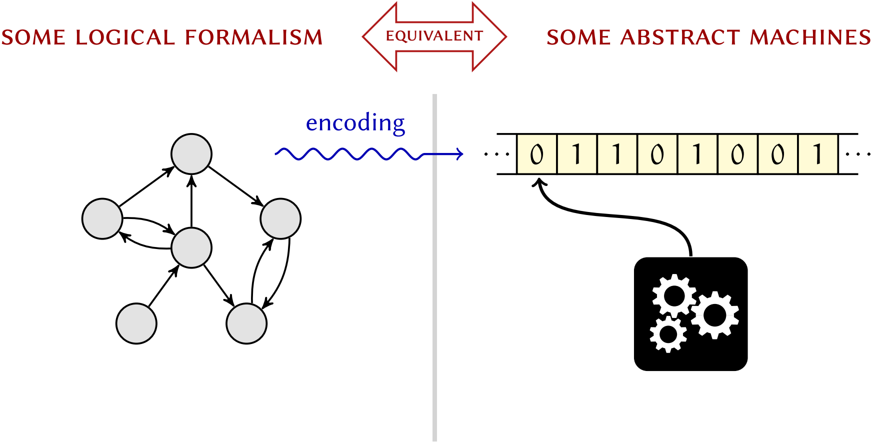


Descriptive complexity

SOME LOGICAL FORMALISM

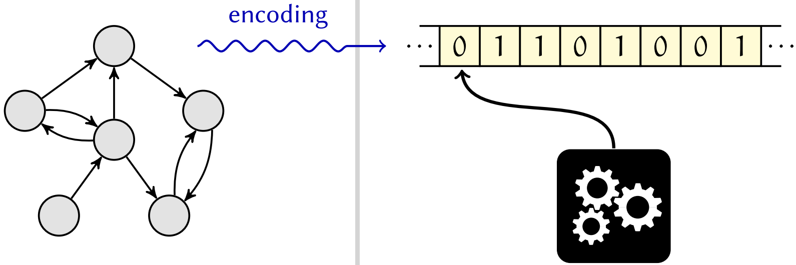


Descriptive complexity



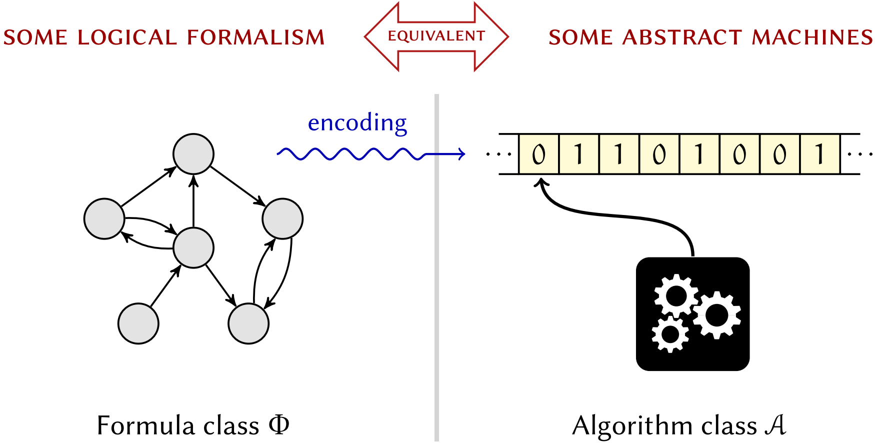
Descriptive complexity

SOME LOGICAL FORMALISM ⟺ EQUIVALENT ⟺ SOME ABSTRACT MACHINES



Formula class Φ

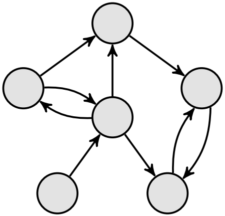
Descriptive complexity



Descriptive distributed complexity

Descriptive distributed complexity

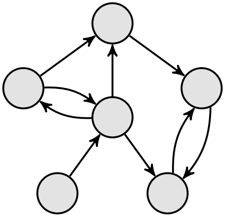
SOME LOGICAL FORMALISM



Formula class Φ

Descriptive distributed complexity

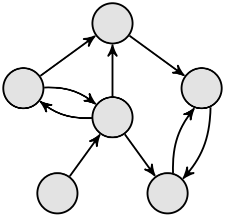
SOME LOGICAL FORMALISM ⟺ EQUIVALENT



Formula class Φ

Descriptive distributed complexity

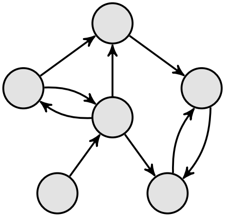
SOME LOGICAL FORMALISM — EQUIVALENT — COMMUNICATING MACHINES



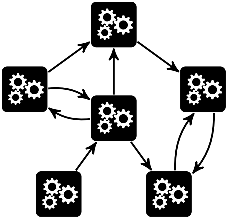
Formula class Φ

Descriptive distributed complexity

SOME LOGICAL FORMALISM — EQUIVALENT — COMMUNICATING MACHINES

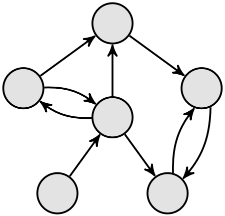


Formula class Φ

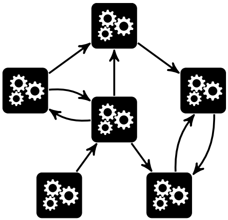


Descriptive distributed complexity

SOME LOGICAL FORMALISM — EQUIVALENT — COMMUNICATING MACHINES

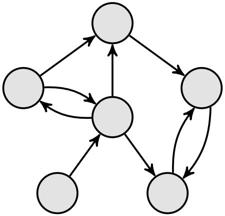


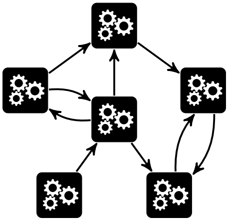
Formula class Φ



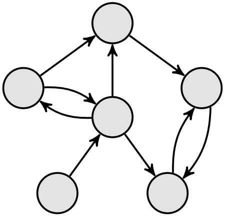
Distributed algorithm class $\mathcal{A}$

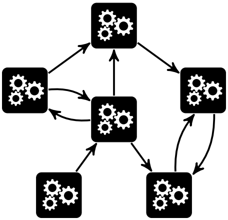
The “Helsinki-Tampere theorem” (2012)





The “Helsinki-Tampere theorem” (2012)

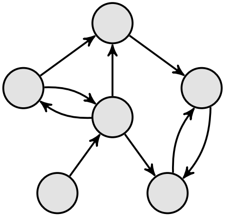


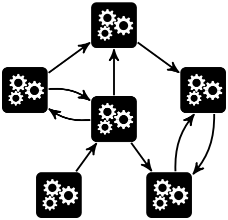


Hella · Järvisalo · Kuusisto · Laurinharju · Lempiäinen · Luosto · Suomela · Virtema

The “Helsinki-Tampere theorem” (2012)

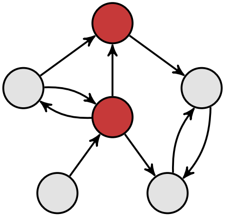
BACKWARD MODAL LOGIC

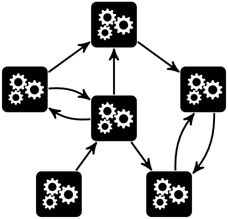




The “Helsinki-Tampere theorem” (2012)

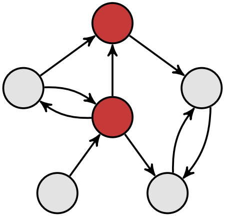
BACKWARD MODAL LOGIC



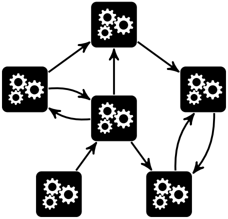


The “Helsinki-Tampere theorem” (2012)

BACKWARD MODAL LOGIC



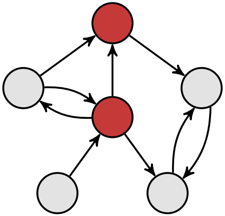
Example: $\bar{\Diamond}(\bar{\Box}\,\text{white} \vee \bar{\Box}\,\text{red})$



Hella · Järvisalo · Kuusisto · Laurinharju · Lempiäinen · Luosto · Suomela · Virtema

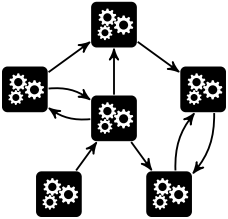
The “Helsinki-Tampere theorem” (2012)

BACKWARD MODAL LOGIC



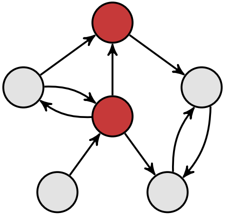
Example: $\bar{\Diamond}(\bar{\Box}\,\text{white} \lor \bar{\Box}\,\text{red})$

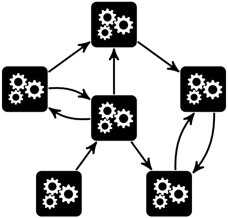
“I have an in-neighbor whose
in-neighbors are all white or all red.”



The “Helsinki-Tampere theorem” (2012)

BACKWARD MODAL LOGIC EQUIVALENT LOCAL DISTRIB. AUTOMATA





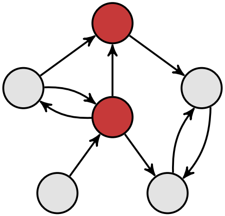
Example: $\overline{\Diamond}(\overline{\Box}\,\text{white} \vee \overline{\Box}\,\text{red})$

“I have an in-neighbor whose
in-neighbors are all white or all red.”

Hella · Järvisalo · Kuusisto · Laurinharju · Lempiäinen · Luosto · Suomela · Virtema

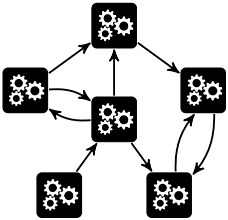
The "Helsinki-Tampere theorem" (2012)

BACKWARD MODAL LOGIC ⟺ EQUIVALENT ⟺ LOCAL DISTRIB. AUTOMATA



Example: $\bar{\Diamond}(\bar{\Box}\,\text{white} \lor \bar{\Box}\,\text{red})$

"I have an in-neighbor whose in-neighbors are all white or all red."



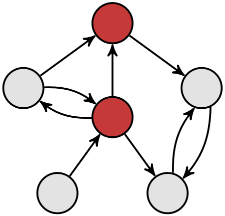
Each node a finite-state machine:

[gear icon] : $Q \times 2^{Q} \to Q$

Hella · Järvisalo · Kuusisto · Laurinharju · Lempiäinen · Luosto · Suomela · Virtema

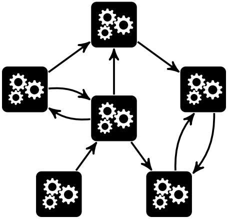
The “Helsinki-Tampere theorem” (2012)

BACKWARD MODAL LOGIC ⟺ EQUIVALENT ⟺ LOCAL DISTRIB. AUTOMATA



Example: $\overline{\Diamond}(\overline{\Box}\,\text{white} \vee \overline{\Box}\,\text{red})$

“I have an in-neighbor whose
in-neighbors are all white or all red.”



Each node a finite-state machine:

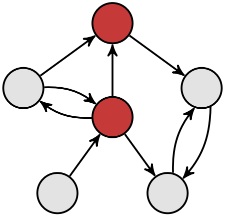
 $: \; Q \times 2^{Q} \to Q$

- Synchronous execution

Hella · Järvisalo · Kuusisto · Laurinharju · Lempiäinen · Luosto · Suomela · Virtema

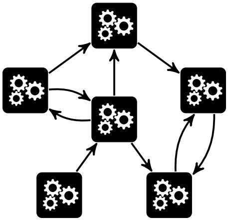
The "Helsinki-Tampere theorem" (2012)

BACKWARD MODAL LOGIC — EQUIVALENT — LOCAL DISTRIB. AUTOMATA



Example: $\overline{\Diamond}\big(\overline{\Box}\,\text{white} \lor \overline{\Box}\,\text{red}\big)$

"I have an in-neighbor whose in-neighbors are all white or all red."



Each node a finite-state machine:

 : $Q \times 2^{Q} \to Q$

- Synchronous execution
- Constant running time

Hella · Järvisalo · Kuusisto · Laurinharju · Lempiäinen · Luosto · Suomela · Virtema

Main contributions

Main contributions

MONADIC SECOND-ORDER LOGIC

Main contributions

MONADIC SECOND-ORDER LOGIC

$$\forall Z \Big(\exists x, y \big(Z(x) \wedge \neg Z(y) \big) \rightarrow \cdots \Big)$$

Main contributions

MONADIC SECOND-ORDER LOGIC | EQUIVALENT | ALTERNATING LOCAL AUTOMATA

$$\forall Z \left(\exists x, y \big(Z(x) \wedge \neg Z(y) \big) \rightarrow \cdots \right)$$

Main contributions

MONADIC SECOND-ORDER LOGIC | EQUIVALENT | ALTERNATING LOCAL AUTOMATA

$$\forall Z \left(\exists x, y \big(Z(x) \wedge \neg Z(y) \big) \rightarrow \cdots \right)$$

$$: Q \times 2^Q \rightarrow 2^Q$$

Main contributions

MONADIC SECOND-ORDER LOGIC ⟺ EQUIVALENT ⟺ ALTERNATING LOCAL AUTOMATA

$\forall Z \Big(\exists x, y \big(Z(x) \wedge \neg Z(y)\big) \rightarrow \cdots\Big)$

$: Q \times 2^Q \rightarrow 2^Q$

\+ Alternation

Main contributions

MONADIC SECOND-ORDER LOGIC ⇔ EQUIVALENT ⇔ **ALTERNATING LOCAL AUTOMATA**

$$\forall Z \left(\exists x, y \big(Z(x) \land \neg Z(y) \big) \to \cdots \right)$$

$: Q \times 2^Q \to 2^Q$

+ Alternation
+ Global acceptance

Main contributions

MONADIC SECOND-ORDER LOGIC ⟺ EQUIVALENT ⟺ ALTERNATING LOCAL AUTOMATA

$$\forall Z \Big(\exists x, y \big(Z(x) \land \neg Z(y) \big) \to \cdots \Big)$$

$: Q \times 2^Q \to 2^Q$

+ Alternation
+ Global acceptance

THE BACKWARD μ-FRAGMENT

Main contributions

MONADIC SECOND-ORDER LOGIC — EQUIVALENT — ALTERNATING LOCAL AUTOMATA

$$\forall Z \left(\exists x, y \big(Z(x) \land \neg Z(y) \big) \to \cdots \right)$$

$: Q \times 2^Q \to 2^Q$

- + Alternation
- + Global acceptance

THE BACKWARD μ-FRAGMENT

$$\mu \begin{pmatrix} X \\ Y \end{pmatrix} . \begin{pmatrix} (R \land Y) \lor \overline{\Diamond} X \\ \overline{\Box} Y \end{pmatrix}$$

Main contributions

MONADIC SECOND-ORDER LOGIC ⟺ EQUIVALENT ⟺ ALTERNATING LOCAL AUTOMATA

$$\forall Z\Big(\exists x, y\big(Z(x) \land \neg Z(y)\big) \rightarrow \cdots\Big)$$

$: Q \times 2^{Q} \rightarrow 2^{Q}$

+ Alternation
+ Global acceptance

THE BACKWARD μ-FRAGMENT ⟺ EQUIVALENT ⟺ ASYNCHRONOUS AUTOMATA

with quasi-acyclic diagrams

$$\mu\begin{pmatrix} X \\ Y \end{pmatrix} . \begin{pmatrix} (R \land Y) \lor \overline{\Diamond} X \\ \overline{\Box} Y \end{pmatrix}$$

Main contributions

MONADIC SECOND-ORDER LOGIC ⇔ EQUIVALENT ⇔ ALTERNATING LOCAL AUTOMATA

$$\forall Z\left(\exists x, y\big(Z(x) \wedge \neg Z(y)\big) \rightarrow \cdots\right)$$

$⚙ : Q \times 2^Q \rightarrow 2^Q$

+ Alternation
+ Global acceptance

THE BACKWARD μ-FRAGMENT ⇔ EQUIVALENT ⇔ ASYNCHRONOUS AUTOMATA *with quasi-acyclic diagrams*

$$\mu\begin{pmatrix} X \\ Y \end{pmatrix} . \begin{pmatrix} (R \wedge Y) \vee \overline{\Diamond} X \\ \overline{\Box} Y \end{pmatrix}$$

$⚙ : Q \times 2^Q \rightarrow Q$

Main contributions

MONADIC SECOND-ORDER LOGIC ⟺ EQUIVALENT ⟺ ALTERNATING LOCAL AUTOMATA

$$\forall Z \Big(\exists x, y \big(Z(x) \wedge \neg Z(y) \big) \rightarrow \cdots \Big)$$

$: Q \times 2^Q \rightarrow 2^Q$

+ Alternation
+ Global acceptance

THE BACKWARD μ-FRAGMENT ⟺ EQUIVALENT ⟺ ASYNCHRONOUS AUTOMATA

with quasi-acyclic diagrams

$$\mu \begin{pmatrix} X \\ Y \end{pmatrix} . \begin{pmatrix} (R \wedge Y) \vee \overline{\Diamond} X \\ \overline{\Box} Y \end{pmatrix}$$

$: Q \times 2^Q \rightarrow Q$

+ Unbounded running time

Main contributions

MONADIC SECOND-ORDER LOGIC ⟺ EQUIVALENT ⟺ ALTERNATING LOCAL AUTOMATA

$$\forall Z \Big(\exists x, y \big(Z(x) \wedge \neg Z(y) \big) \rightarrow \cdots \Big)$$

$: Q \times 2^Q \rightarrow 2^Q$

+ Alternation
+ Global acceptance

THE BACKWARD μ-FRAGMENT ⟺ EQUIVALENT ⟺ ASYNCHRONOUS AUTOMATA *with quasi-acyclic diagrams*

$$\mu \begin{pmatrix} X \\ Y \end{pmatrix} . \begin{pmatrix} (R \wedge Y) \vee \overline{\Diamond} X \\ \overline{\Box} Y \end{pmatrix}$$

$: Q \times 2^Q \rightarrow Q$

+ Unbounded running time
− Asynchronous execution

Perspectives

Perspectives

LOGICAL DESCRIPTIONS:

- A tool to specify and synthesize distributed algorithms?

Perspectives

LOGICAL DESCRIPTIONS:

- A tool to specify and synthesize distributed algorithms?
- The key to a complexity theory for distributed computing?

Perspectives

LOGICAL DESCRIPTIONS:

- A tool to specify and synthesize distributed algorithms?
- The key to a complexity theory for distributed computing?
  - Forces us to formalize our models of computation.

Perspectives

LOGICAL DESCRIPTIONS:

- A tool to specify and synthesize distributed algorithms?
- The key to a complexity theory for distributed computing?
  - Forces us to formalize our models of computation.
  - Can help to identify natural and robust classes of algorithms.

Perspectives

LOGICAL DESCRIPTIONS:

- A tool to specify and synthesize distributed algorithms?
- The key to a complexity theory for distributed computing?
  - Forces us to formalize our models of computation.
  - Can help to identify natural and robust classes of algorithms.
  - Transfers classical complexity theory to the distributed setting.

Perspectives

LOGICAL DESCRIPTIONS:

- A tool to specify and synthesize distributed algorithms?
- The key to a complexity theory for distributed computing?
  - Forces us to formalize our models of computation.
  - Can help to identify natural and robust classes of algorithms.
  - Transfers classical complexity theory to the distributed setting.

Thanks!