Modal Logic and Distributed Message Passing Automata

Antti Kuusisto
Uni of Wrocław

Distributed message passing systems

- Computer networks
- Cellular automata
- Brain
- etc.

Descriptive Complexity of Distributed Computing

L. Hella, M. Järvisalo, A. Kuusisto, J. Laurinharju, T. Lempiäinen, K. Luosto, J. Suomela and J. Virtema. Weak models of distributed computing, with connections to modal logic. PODC 2012.

- Introduces an approach to descriptive complexity of distributed computing.
- Characterizes several complexity classes of distributed computing by related modal logics.
- Separates complexity classes of distributed computing with the help of logical methods.

Descriptive Complexity of Distributed Computing

Hella et al. 2012 only characterizes classes defined by constant time distributed automata.

We obtain the following logical characterizations of classes defined by non-constant-time automata.

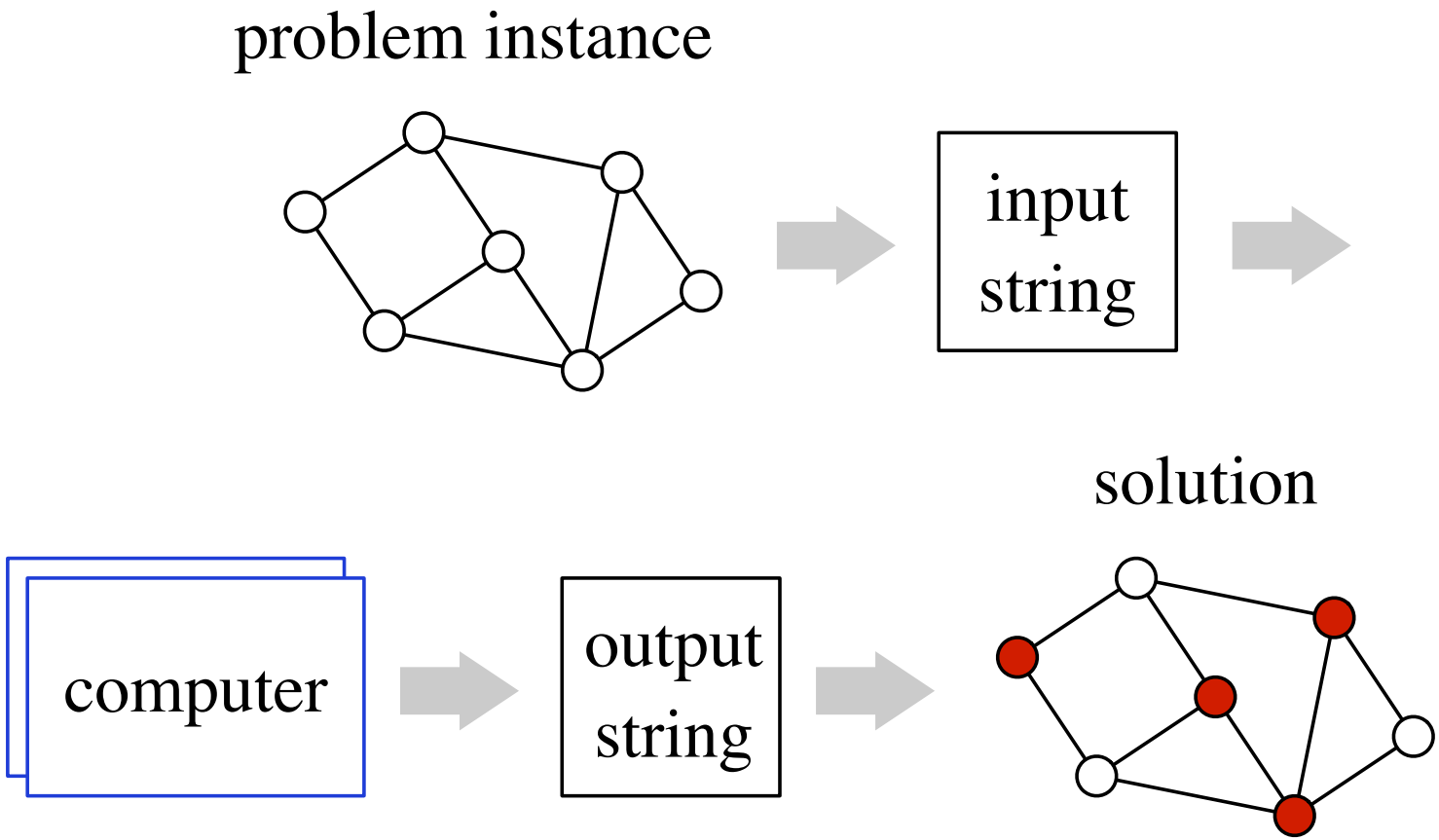
Theorem

Recognizability by finite message passing automata is captured by modal substitution calculus, i.e., $\mathrm{FMPA} = \mathrm{MSC}$.

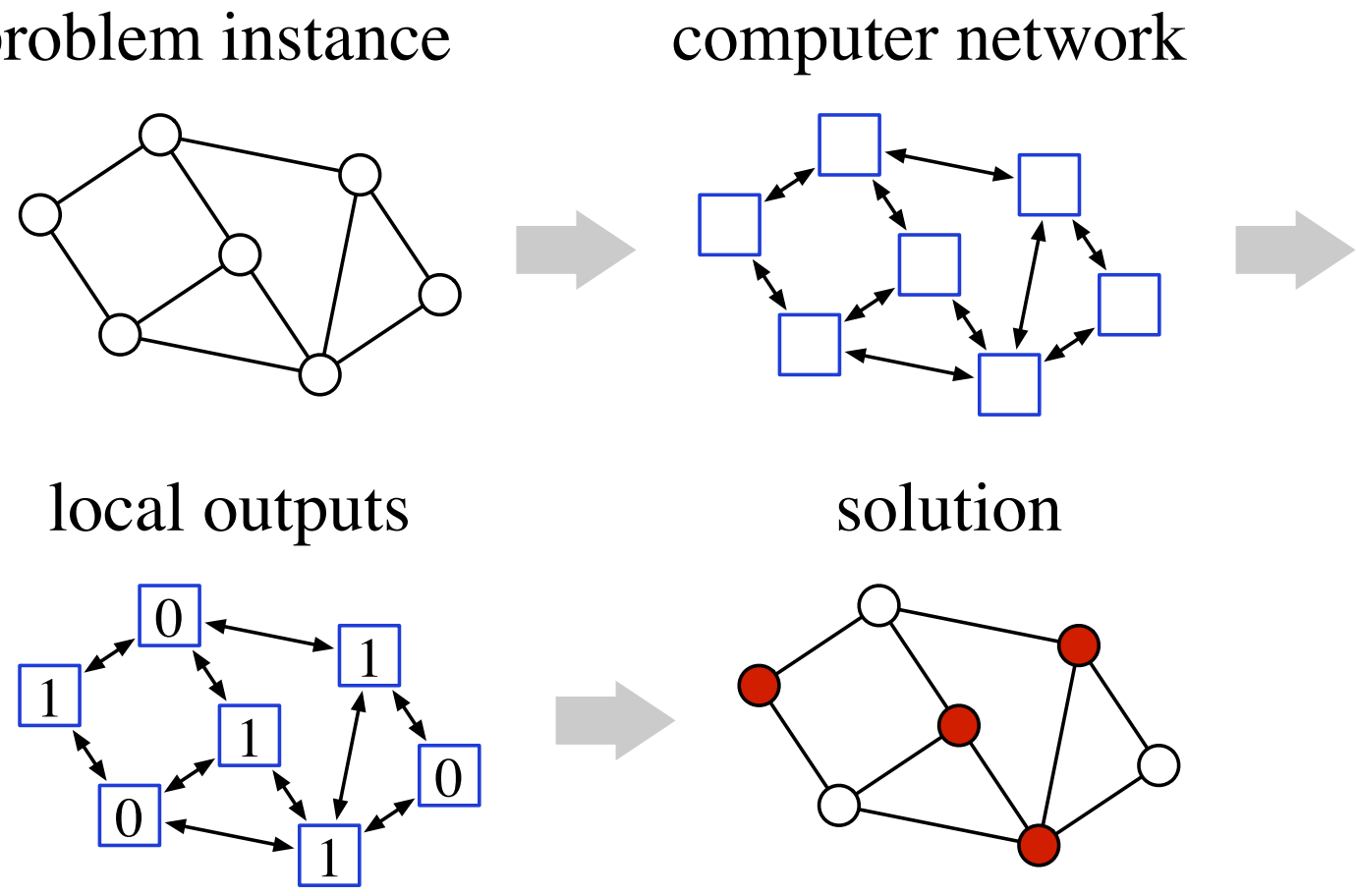
Theorem

Modal theories capture Co-MPA*: a class* $\mathcal{C}$ *of pointed Kripke models is definable by a modal theory iff the complement of* $\mathcal{C}$ *is recognizable by a message passing automaton (*MPA*).*

Classical Computation



Distributed Computation



Deterministic Distributed Algorithms

A distributed system is defined by a directed labelled graph

$$(W, R, p_1, ..., p_k),$$

together with an automaton A.

- Each node $w \in W$ contains a copy (A, w) of the automaton A.
- $R \subseteq W \times W$ is a collection of *communication channels*.
- Predicates $p_i \subseteq W$ encode a *local input* at each node. The i^{th} input bit at node w is 1 iff $w \in p_i$.

Deterministic Distributed Algorithms

Computation proceeds in sychronous steps.

- In *one time step*, each machine (A, w)
  - receives messages from its neighbours and sends messages to its neighbours,
  - updates its state based on the received messages and previous state.

Deterministic Distributed Algorithms

Automaton A is a tuple $(Q, M, \pi, \delta, \mu, F)$.

- Q: states,
- M: messages,
- $\pi : \mathcal{P}(\{p_1, ..., p_k\}) \longrightarrow Q$ gives the initial state of each automaton (A, w),
- $\delta : \mathcal{P}(M) \times Q \longrightarrow Q$ determines the next state of (A, w) based on the previous state and received messages.
- $\mu : Q \longrightarrow M$ constructs a message $m \in M$ that the automaton (A, w) broadcasts to all its neighbours.
- $F \subseteq Q$ is the set of accepting states.

(Hella et al. 2012 studies also other notions of automata.)

Distribured Algorithms

A node w *accepts* if it visits some accepting state $q \in F$ at least once. More formally:

Each communication round $n \in \mathbb{N}$ defines a *global configuration* $f_n : W \longrightarrow Q$.

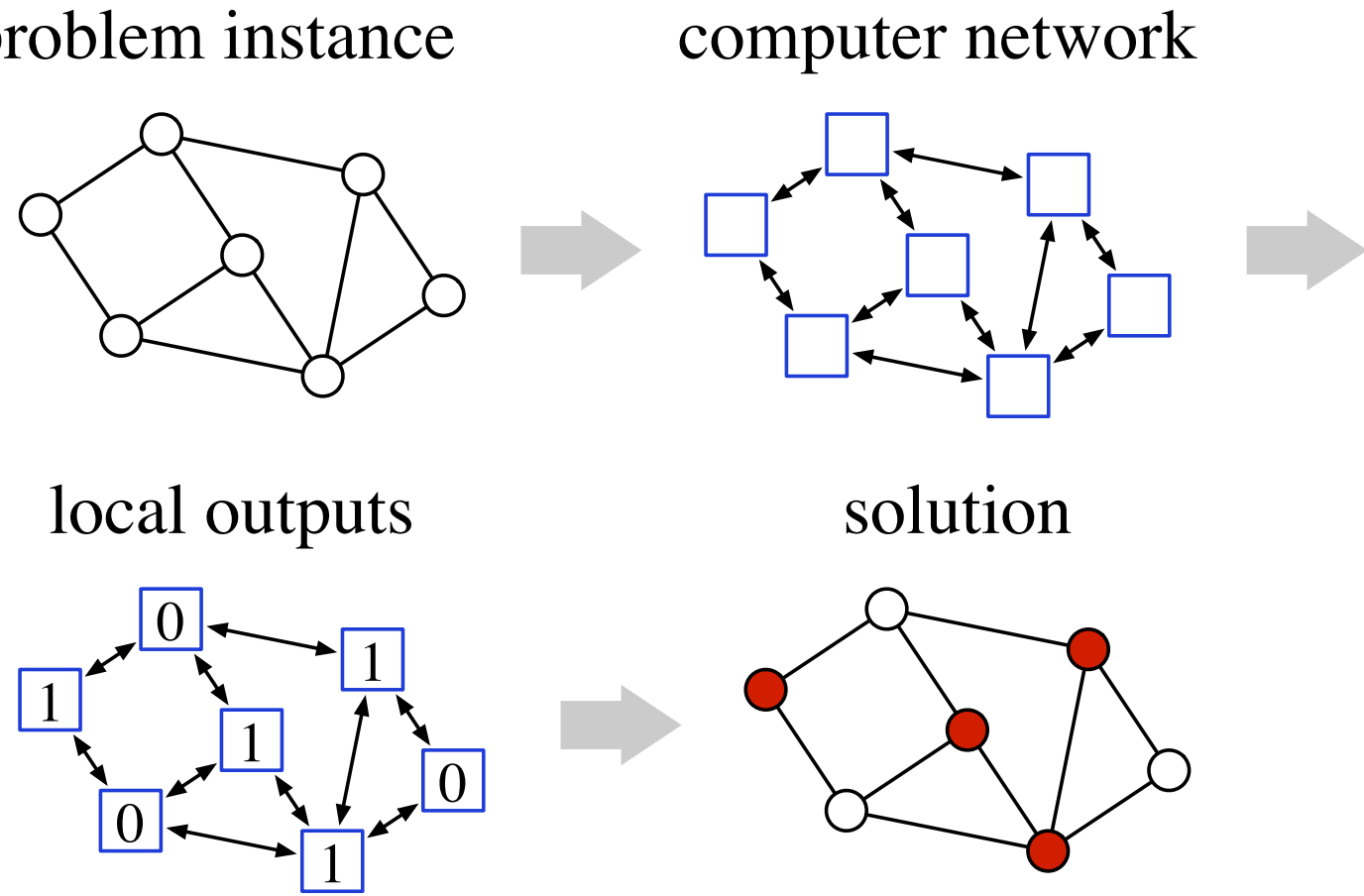
$f_0(w) :=$ initial state at w.

Call $N :=$ the set of messages received by node w in round $n+1$.
Then $f_{n+1}(w) :=$ the new state at $w = \delta(N, f_n(w))$.

The node w *accepts* if $f_k(w) \in F$ for some $k \in \mathbb{N}$.

Distributed Algorithms

Automaton A therefore computes a subset $S \subseteq W$ —the set of accepting nodes—-of the domain W of the distributed network $(W, R, p_1, ..., p_k)$.



Distributed Algorithms

The *decision time* of an automaton A at a node w is the number of communication rounds before the node visits an accepting state *for the first time*.

An automaton A specifies a *constant time algorithm*, if there exists a $k \in \mathbb{N}$ such that the decision time of A at any node of any network is at most k.

Descriptive Complexity of Distributed Computing

Hella et al. 2012 shows (for several types of message passing automata) that

constant time distributed algoritms $=$ formulae of modal logic.

The modal logic used depends on the type of automata studied.
(For example the class $\mathrm{SB}(1)$ is captured by ML; the class $\mathrm{MB}(1)$ is captured by GML, and so on: see the paper for further details.)

$$M, w \models \Diamond\psi \quad \text{iff} \quad \begin{array}{l} w \text{ receives the message "}\psi \text{ is true"} \\ \text{from some } u \text{ such that } (w, u) \in R. \end{array}$$

here $M = (W, R, p_1, ..., p_k)$.

Modal Substitution Calculus

Modal substitution calculus (MSC) consists of *programs* of the following type:

$$\begin{array}{lll} X_1 & :- & p \wedge \neg q \\ X_2 & :- & \Diamond(p \wedge q) \\ X_3 & :- & p \\ X_4 & :- & \Diamond\Box p \end{array} \qquad \begin{array}{lll} X_1 & :- & (X_1 \wedge \neg X_4) \rightarrow \Diamond q \\ X_2 & :- & X_1 \wedge \Diamond X_2 \\ X_3 & :- & p \wedge \Box X_1 \\ X_4 & :- & X_4 \vee \Diamond p \end{array}$$

A program has two two lists of clauses. The clauses on the left are terminal clauses and the ones on the right iteration clauses.

The right hand side of a terminal clause is any formula of modal logic. The right hand side of an iteration clause is a formula of modal logic that can use the *variable symbols* X_i as well as ordinary proposition symbols in $\{ p_1, ..., p_k \}$.

Modal Substitution Calculus

$$\begin{array}{lll} X_1 & :- & \psi_1 \\ . & & \\ . & & \\ . & & \\ X_m & :- & \psi_m \end{array} \qquad \begin{array}{lll} X_1 & :- & \varphi_1 \\ . & & \\ . & & \\ . & & \\ X_m & :- & \varphi_m \end{array}$$

Define $X_i^0 := \psi_i$.

Define X_i^{n+1} to be the modal formula obtained by simultaneously replacing each variable X_j of the schema φ_i by X_j^n.

$M, w \models \mathit{Program}$ iff $M, w \models X_1^n$ for some $n \in \mathbb{N}$.

Here $M = (W, R, p_1, ..., p_k)$ is a Kripke model (or a distributed network).

Descriptive Characterizations

Theorem
MSC *captures recognizability by finite messages passing automata.*

In other words, for each MSC formula, there exists a corresponding FMPA, and vice versa.

Properties of MSC

Theorem

The single variable fragment MSC^1 *of* MSC *is not contained in* MSO.

Properties of MSC

Theorem

The SAT *and* FINSAT *problems of* MSC^1 *are complete for* PSPACE.

Theorem

In the finite, the fragment of μ-calculus that does not use ν (negations on the atomic level) is contained in MSC.

Sketch of proof idea.

It is well known that μ-calculus can be defined in terms of modal equation systems. □

Theorem

The fragment of μ-calculus that does not use μ (negations on the atomic level) is not contained in MSC.

Descriptive Characterizations

Theorem

Modal theories capture Co-MPA, i.e., a class $\mathcal{C}$ of pointed Kripke models is definable by a modal theory iff the complement of $\mathcal{C}$ is recognized by an infinite message passing automaton.

Thx!

Theorem

The single variable fragment MSC^1 *of* MSC *is not contained in* MSO.

Sketch of proof idea:

The program $(X \ :- \ \Box\bot, \quad X \ :- \ \Box X \wedge \Diamond X)$ recognizes the nodes w such that every directed walk from w to a dead-end has exactly the same finite length (and a dead-end is indeed reachable). $\square$

Theorem

The fragment of μ-calculus that does not use μ (negations on the atomic level) is not contained in MSC.

Sketch of proof idea.

MSC cannot define non-reachability of a dead-end: $\nu X.(\Diamond \top \wedge \Box X)$. For an MSC-automaton, nodes of a cycle graph appear similar to internal nodes in sufficiently long line-like graphs. $\square$

Satisfiability is a clopen problem.

The SAT-problem of MSC seems undecidable (Suomela, 2013). Idea: a one-dimensional cellular automaton can simulate the tape of a Turing machine, and modal logic seems expressive enough to deal with unwanted models, such as models with nodes of degree ≥ 3. Also, cycles etc. do not seem to cause any problems here.