Rice-like theorems for automata networks

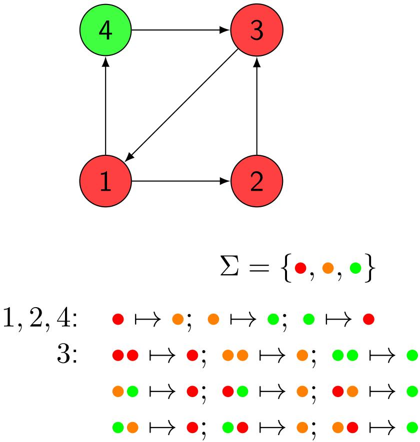
G. Gamard P. Guillon K. Perrot G. Theyssier

WAN 2021, Marseille, France

Automata networks

In this talk:

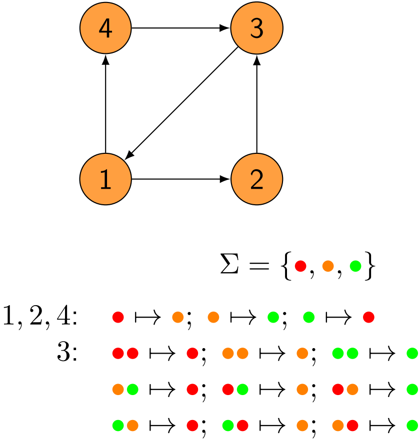
- Finite digraphs of finite automata
- Each node (automaton) has **its own alphabet**, transitions
- A node reads the state of its inbound neighbors to update
- Nodes update in **parallel** (all at the same time)



Automata networks

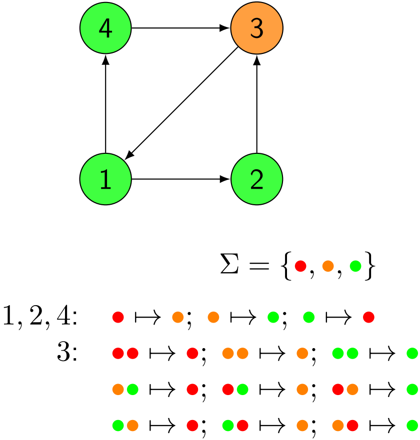
In this talk:

- Finite digraphs of finite automata
- Each node (automaton) has **its own alphabet**, transitions
- A node reads the state of its inbound neighbors to update
- Nodes update in **parallel** (all at the same time)

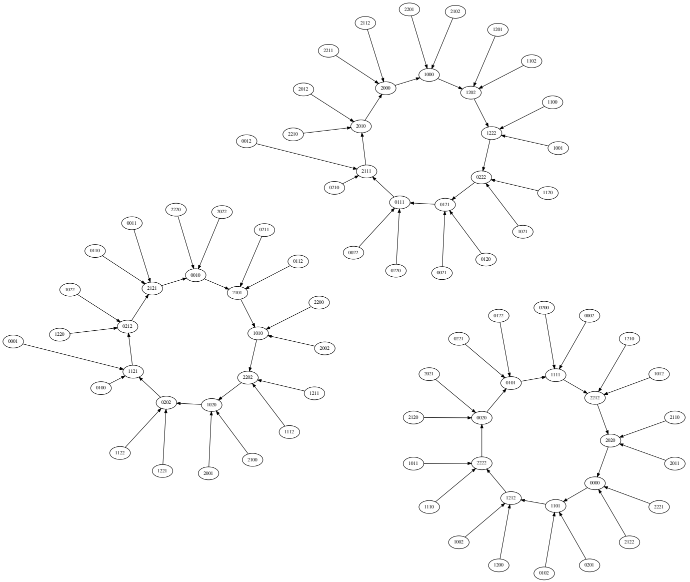


In this talk:

- Finite digraphs of finite automata
- Each node (automaton) has **its own alphabet**, transitions
- A node reads the state of its inbound neighbors to update
- Nodes update in **parallel** (all at the same time)



Transition graphs



Transition graphs

The transition graph of a network is the graph of the "function computed by" this network.

The network is a succint way to describe its transition graph.

The Rice theorem

Theorem (Rice, 1953)

Any nontrivial property of the *function computed by* a Turing Machine is undecidable.

The Rice theorem

Theorem (Rice, 1953)

Any nontrivial property of the *function computed by* a Turing Machine is undecidable.

Our goal: test properties of $\mathcal{G}_F$, given an automata network F.

Metatheorem

Any nontrivial property of the *transition graph of* an Automata Network is hard.

So much fine print...

- Property?
- Nontrivial?
- Hard?

The Rice theorem

Theorem (Rice, 1953)

Any nontrivial property of the *function computed by* a Turing Machine is undecidable.

Our goal: test properties of $\mathcal{G}_F$, given an automata network F.

Metatheorem

Any nontrivial property of the *transition graph of* an Automata Network is hard.

So much fine print...

- Property?
- Nontrivial?
- Hard?

Finite transition graph $\implies$ everything is decidable

"Hard" is something like NP-hard.

Encoding networks

When giving an automata network to an algorithm, how shall we encode it?

Assumptions

- Alphabets are $\{0, \ldots, n-1\}$
- Neighbors of a node are ordered

Encoding

- A *communication* graph (adjacency matrix)
- One Boolean circuit per node
  - States encoded in binary

Nodes are allowed to ignore a neighbor $\neq$ interaction graph.

First-Order properties

Definition (First-order property)

- $\forall x : \phi$ "for all configuration x": variables denote **configurations**
- $T(x, y)$ "x transitions to y in one step"
- $x = y$
- $\phi_1 \wedge \phi_2$
- $\neg\phi$

Examples

- **Fixed point.** $\exists x : T(x, x)$
- **3-cycle.** $\exists x_1, x_2, x_3 : T(x_1, x_2) \wedge T(x_2, x_3) \wedge T(x_3, x_1)$
- **Injectivity.** $\forall x_1, x_2, y : [T(x_1, y) \wedge T(x_2, y)] \implies [x_1 = x_2]$
- **Determinism.** $\forall x, y_1, y_2 : [T(x, y_1) \wedge T(x, y_2)] \implies [y_1 = y_2]$

First-order properties are hard

Definition (ϕ-Dynamics)

ϕ-DYNAMICS

Input: a *deterministic* automata network F

Question: does $\mathcal{G}_F \models \phi$?

Note: ϕ is **not** part of the input!

Theorem (ΓGPT, 2020)

For a fixed ϕ, the problem ϕ-DYNAMICS is either $O(1)$, or NP-hard, or coNP-hard.

A reduction to SAT

Fix ϕ once and for all.

Theorem (ΓGPT, 2020)

The problem ϕ-Dynamics is either $O(1)$, or NP-hard, or coNP-hard.

Let $\sqcup$ denote the disjoint union.

Lemma

ϕ-Dynamics is NP-hard if there are G, J, D with $|J| = |D|$ and:

$$G \sqcup J \sqcup J \sqcup \cdots \sqcup J \sqcup \cdots \sqcup J \not\models \phi,$$

$$G \sqcup J \sqcup J \sqcup \cdots \sqcup D \sqcup \cdots \sqcup J \models \phi.$$

"You can have Jillions of J's, but one D makes a Difference."

A reduction to SAT

Lemma

ϕ-DYNAMICS is NP-hard if there are G, J, D with $|J| = |D|$ and:

$$J \sqcup J \sqcup \cdots \sqcup J \sqcup \cdots \sqcup J \not\models \phi,$$
$$J \sqcup J \sqcup \cdots \sqcup D \sqcup \cdots \sqcup J \models \phi.$$

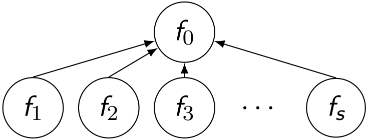
Let S denote an instance of SAT with s variables.

Make a network with $s+1$ automata $f_0, \ldots, f_s$.

Alphabets: f_0 over $\{1, \ldots, |J|\}$ and $f_1, \ldots, f_s$ over $\{0, 1\}$.

Update: f_0 evaluates $S(f_1 \ldots f_s)$;

- if it finds 0, it realizes J,
- if it finds 1, it realizes D.



A reduction to SAT

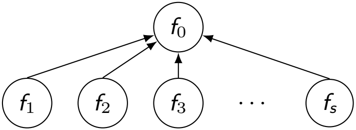
Let S denote an instance of SAT with s variables.

Make a network with $s+1$ automata $f_0, \ldots, f_s$.

Alphabets: f_0 over $\{1, \ldots, |J|\}$ and $f_1, \ldots, f_s$ over $\{0, 1\}$.

Update: f_0 evaluates $S(f_1 \ldots f_s)$;

- if it finds 0, it realizes J,
- if it finds 1, it realizes D.



Claim 1

The dynamics has a copy of D per **positive** assignment for S, and a copy of J per **negative** assignment for S.

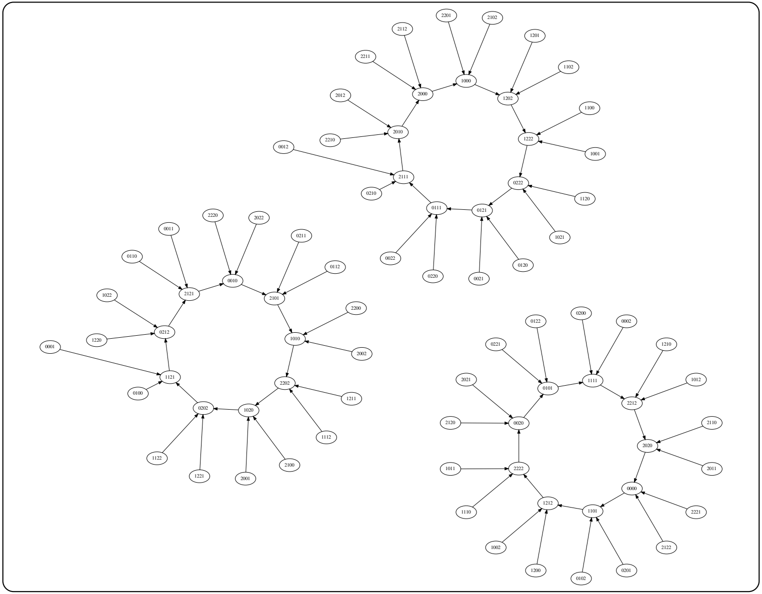
Claim 2

This network is producible in polynomial time.

Finding G, J, D

Our next mission is to find G, J, D as announced.

Determinism



Elementary equivalence

Let m denote the **rank** of ϕ (its number of quantifiers), and G_1, G_2 be graphs.

Definition (Elementary equivalence)

Write $G_1 \equiv_m G_2$ iff G_1, G_2 satisfy the same formulae of rank m.

Lemma (Fraïssé, 1953)

The relation $\equiv_m$ has finitely many classes: $\alpha_1, \ldots, \alpha_{a(m)}$.

Elementary equivalence

Let m denote the **rank** of ϕ (its number of quantifiers), and G_1, G_2 be graphs.

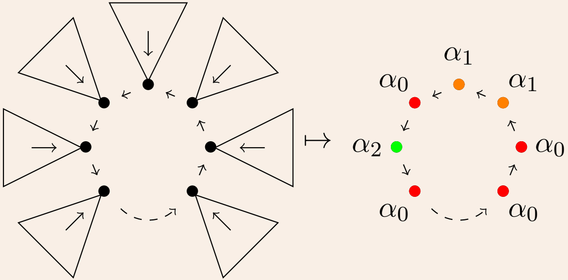
Definition (Elementary equivalence)

Write $G_1 \equiv_m G_2$ iff G_1, G_2 satisfy the same formulae of rank m.

Lemma (Fraïssé, 1953)

The relation $\equiv_m$ has finitely many classes: $\alpha_1, \ldots, \alpha_{a(m)}$.

Definition (Dulc)

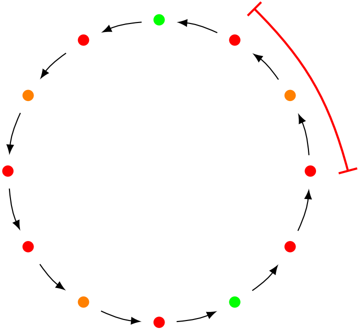


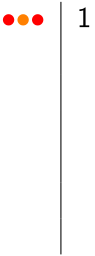
Profiles

Let $e = 2 \cdot 3^m + 1$ and $\infty = m \cdot e$.

Definition (Profile)

The **profile** of a dulc is counting its strings of length m, capped at $m \cdot e$.



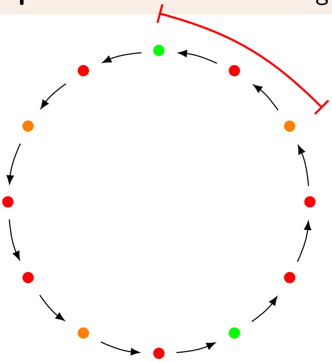


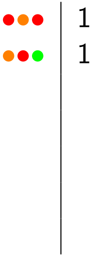
Profiles

Let $e = 2 \cdot 3^m + 1$ and $\infty = m \cdot e$.

Definition (Profile)

The **profile** of a dulc is counting its strings of length m, capped at $m \cdot e$.



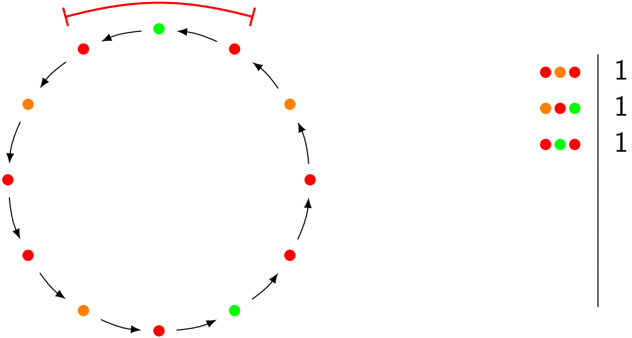


Profiles

Let $e = 2 \cdot 3^m + 1$ and $\infty = m \cdot e$.

Definition (Profile)

The **profile** of a dulc is counting its strings of length m, capped at $m \cdot e$.

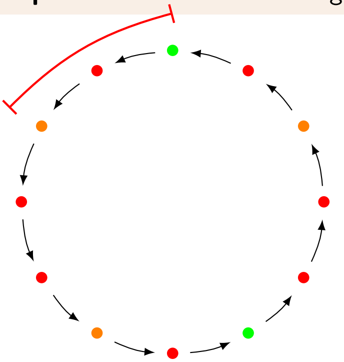


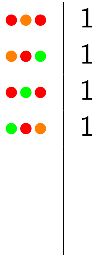
Profiles

Let $e = 2 \cdot 3^m + 1$ and $\infty = m \cdot e$.

Definition (Profile)

The **profile** of a dulc is counting its strings of length m, capped at $m \cdot e$.



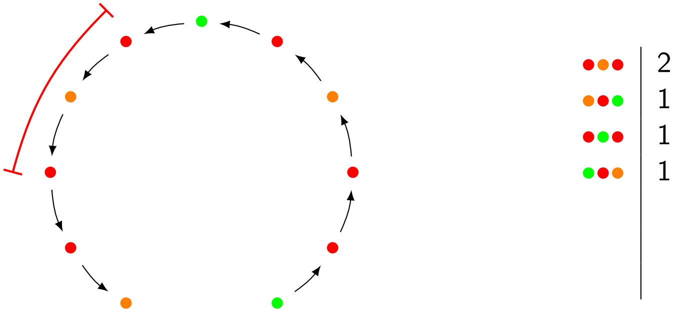


Profiles

Let $e = 2 \cdot 3^m + 1$ and $\infty = m \cdot e$.

Definition (Profile)

The **profile** of a dulc is counting its strings of length m, capped at $m \cdot e$.

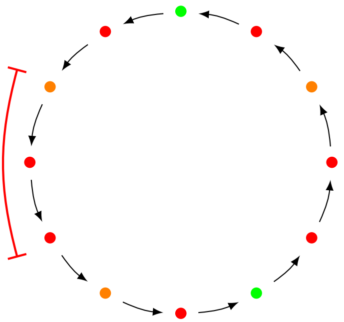


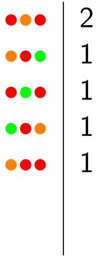
Profiles

Let $e = 2 \cdot 3^m + 1$ and $\infty = m \cdot e$.

Definition (Profile)

The **profile** of a dulc is counting its strings of length m, capped at $m \cdot e$.



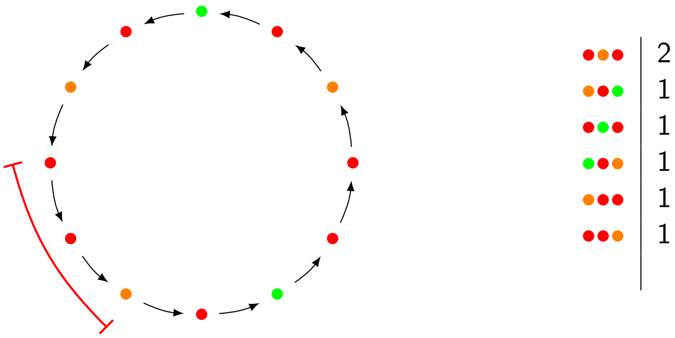


Profiles

Let $e = 2 \cdot 3^m + 1$ and $\infty = m \cdot e$.

Definition (Profile)

The **profile** of a dulc is counting its strings of length m, capped at $m \cdot e$.

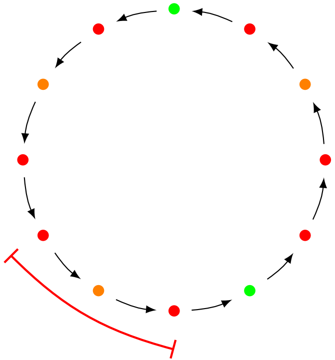


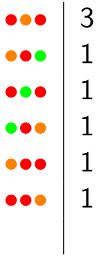
Profiles

Let $e = 2 \cdot 3^m + 1$ and $\infty = m \cdot e$.

Definition (Profile)

The **profile** of a dulc is counting its strings of length m, capped at $m \cdot e$.



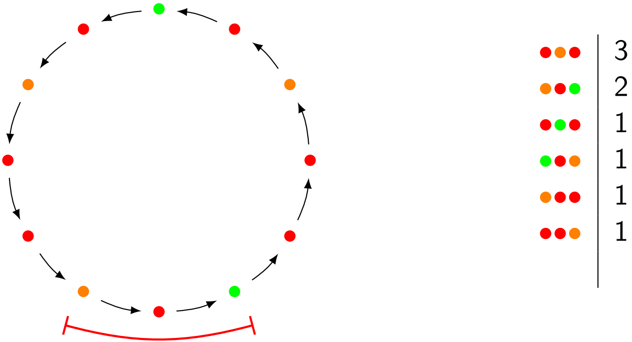


Profiles

Let $e = 2 \cdot 3^m + 1$ and $\infty = m \cdot e$.

Definition (Profile)

The **profile** of a dulc is counting its strings of length m, capped at $m \cdot e$.

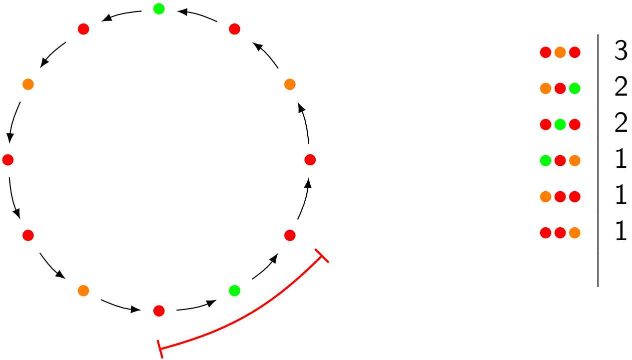


Profiles

Let $e = 2 \cdot 3^m + 1$ and $\infty = m \cdot e$.

Definition (Profile)

The **profile** of a dulc is counting its strings of length m, capped at $m \cdot e$.

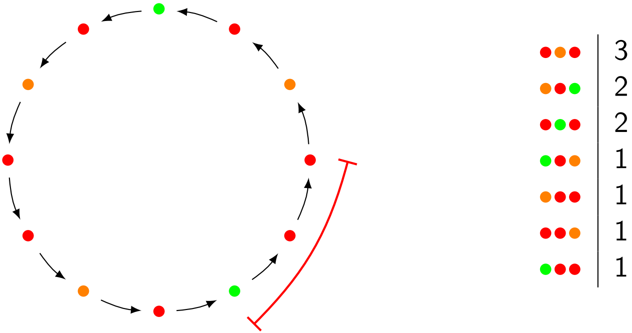


Profiles

Let $e = 2 \cdot 3^m + 1$ and $\infty = m \cdot e$.

Definition (Profile)

The **profile** of a dulc is counting its strings of length m, capped at $m \cdot e$.

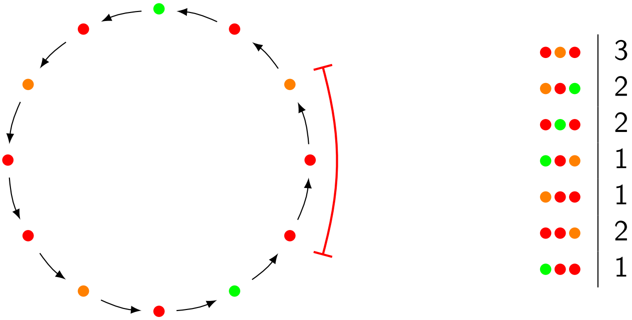


Profiles

Let $e = 2 \cdot 3^m + 1$ and $\infty = m \cdot e$.

Definition (Profile)

The **profile** of a dulc is counting its strings of length m, capped at $m \cdot e$.



Theorem (ΓGPT, 2020)

If $\text{Dulc}(G_1)$ and $\text{Dulc}(G_2)$ have same profile, then $G_1 \equiv_m G_2$.

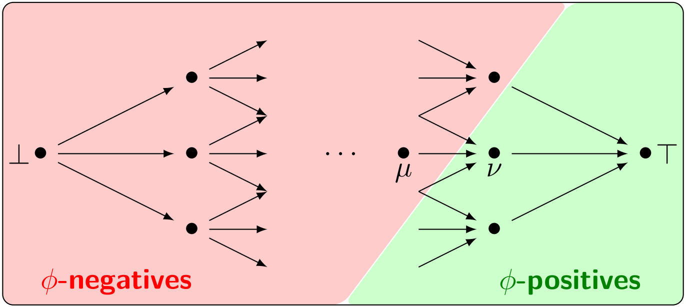
Ordering profiles

Let π_1, π_2 denote profiles.

Write $\pi_1 \leq \pi_2$ iff for all s, we have $\pi_1(s) \leq \pi_2(s)$.

Facts

There are finitely many profiles.
There is a minimal profile ($\bot$) and a maximal profile ($\top$).
Each profile is either **ϕ-positive** or **ϕ-negative**.



Finding G, J, D

Assume ϕ has infinitely many **positive** and **negative** instances.
(Otherwise, ϕ-Dynamics is $O(1)$.)

Assume $\top$ is **ϕ-positive**.
(Otherwise, consider $\neg\phi$ and get coNP-hardness.)

Proof (Existence of J and D)

Infinitely many negative graphs, but finitely many negative profiles.

There's a negative profile π such that $\pi(\widetilde{J}) = \infty$ for some $\widetilde{J}$.

Take a maximal such π. (Any $\pi' > \pi$ is positive.)

- Let G be a graph with profile π.
- Let J be a graph with profile $\widetilde{J}$. (So G and $G \sqcup J$ have same profile.)
- Let D be any graph such that $\pi(D) < \infty$.

First-order properties are hard

Theorem (ΓGPT, 2020)

ϕ-DYNAMICS is either $O(1)$, or NP-hard, or coNP-hard

First-order properties are hard

Theorem (ΓGPT, 2020)

ϕ-Dynamics is either $O(1)$, or NP-hard, or coNP-hard

The same applies to other problems:

ϕ-Bijective-Dynamics

Input: an deterministic automata network F

Promise: F is bijective

Question: does $\mathcal{G}_F \models \phi$?

ϕ-Limit-Dynamics

Input: an deterministic automata network F

Question: does the limit graph of F satisfy ϕ?

Other things are also hard

Theorem (ΓGPT, 2020)

Let ℓ be a level of PH.
There is a formula ϕ_ℓ such that ϕ_ℓ-$\textsc{Dynamics}$ is ℓ-hard.

Other things are also hard

Theorem (ΓGPT, 2020)

Let ℓ be a level of PH.
There is a formula ϕ_ℓ such that ϕ_ℓ-DYNAMICS is ℓ-hard.

Theorem (ΓGPT, 2020)

The following problem is PSPACE-complete:

AN-DYNAMICS

Input: a network F *and a first-order formula* ϕ

Question: does $\mathcal{G}_F \models \phi$?

(This time, ϕ is part of the input!)

Monadic Second-Order properties

Definition (Monadic Second-Order property)

- $\forall x : \phi$
- $\forall \mathbf{X} : \phi$ "for all **set** of configurations $\mathbf{X}$"
- $x \in \mathbf{X}$ "x belongs to $\mathbf{X}$"
- $T(x, y)$, $x = y$, $\phi_1 \wedge \phi_2$, $\neg\phi$

Monadic Second-Order properties

Definition (Monadic Second-Order property)

- $\forall x : \phi$
- $\forall \mathbf{X} : \phi$ "for all **set** of configurations $\mathbf{X}$"
- $x \in \mathbf{X}$ "x belongs to $\mathbf{X}$"
- $T(x, y)$, $x = y$, $\phi_1 \wedge \phi_2$, $\neg\phi$

Examples

- **Chains.** $T^*(x, y) := \exists \mathbf{P} : x, y \in \mathbf{P} \bigwedge \forall z \in \mathbf{P} :$
$$\deg^+_{\mathbf{P}}(x) = 1, \deg^-_{\mathbf{P}}(x) = 0,$$
$$\deg^+_{\mathbf{P}}(y) = 0, \deg^-_{\mathbf{P}}(y) = 1,$$
$$\deg^+_{\mathbf{P}}(z) = 1, \deg^+_{\mathbf{P}}(z) = 1.$$
- **Connexity.** $\forall x, y : T^*(x, y) \wedge T^*(y, x)$.

The "dynamics" problem over MSO

Now fix ψ an **MSO** formula.

Definition (ψ-MSO-Dynamics)

ψ-MSO-DYNAMICS

Input: a **nondeterministic** network F

Question: does $\mathcal{G}_F \models \psi$?

Question

What is the complexity of ψ-MSO-DYNAMICS?

Can we find G, J, D for an arbitrary fixed MSO formula ψ?

Universal D for MSO

Proposition (Bonnet, ΓGPT, 2021)

For all m, there's a graph D_m such that for all ψ of rank m, either:

- for all G, we have $G \sqcup D_m \models \psi$; or
- for all G, we have $G \sqcup D_m \not\models \psi$.

We have a "universal D" that only depends on the rank of ψ!

We might need to consider $\neg\psi$, though.

Universal D for MSO

Proposition (Bonnet, ΓGPT, 2021)

For all m, there's a graph D_m such that for all ψ of rank m, either: for all G, we have $G \sqcup D_m \models \psi$; or for all G, we have $G \sqcup D_m \not\models \psi$.

Lemma 1 (cf. Courcelle's book)

The relation $\equiv_m$ for MSO has finitely many classes $\alpha_1, \ldots, \alpha_{a(m)}$.

Lemma 2

For all G, there's an integer p such that $\bigsqcup^p G \equiv_m \bigsqcup^{p+1} G$.

Proof (Proposition)

Let $A_1, \ldots, A_{a(m)}$ be representatives of $\alpha_1, \ldots, \alpha_{a(m)}$.
Let $\mathbf{p}$ be the max p given by Lemma 2 over $A_1, \ldots, A_{a(m)}$.

$$D_m = \bigsqcup_{i=1}^{a(m)} \bigsqcup^{\mathbf{p}} A_i$$

Tree decompositions

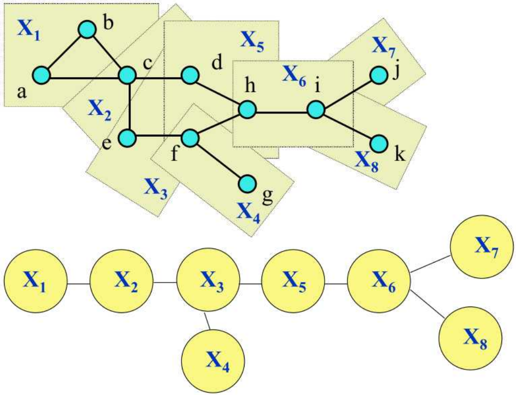


Figure 1: Tree decomposition of width 2 for example graph G. This is the minimum possible treewidth, since G contains a K_3 as a minor.

http://www.mamicode.com/info-detail-2237033.html

G, J for MSO

Let k denote an arbitrary integer.

Proposition (ΓGPT, 2021)

If ψ has an infinity of models and countermodels of treewidth $\leq k$, then there are suitable G and J.

Theorem (Courcelle)

For all ψ, k, there is a tree automaton testing ψ when run on tree decompositions with bags of size k.

Proof (Proposition)

Find the tree automaton for ψ and k and use a pumping lemma.

(We need to replace $\bigsqcup$ with a more complicated "gluing" operator.)

Many MSO formulae are hard

Let k denote an arbitrary integer.

Theorem (ΓGPT, 2021)

If ψ has an infinity of models and countermodels of treewidth $\leq k$, then ψ-$\textsc{Dynamics}$ is either NP- or coNP-hard.

Many MSO formulae are hard

Let k denote an arbitrary integer.

Theorem (ΓGPT, 2021)

If ψ has an infinity of models and countermodels of treewidth $\leq k$, then ψ-$\textsc{Dynamics}$ is either NP- or coNP-hard.

What about ψ not satisfying that condition?

- Those with finitely many (counter)models are $O(1)$.
- The other ones are trick (e.g. $\textsc{clique}$)...
- The pumping techniques do not work anymore for them.

Perspectives

- The last case of the MSO theorem
- The Boolean case (or other fixed alphabet)
- Other update modes
- Other logics (Counting...)
- Your question here

Thank you for your attention!