

REACTIVE SYNTHESIS OVER INFINITE DATA DOMAINS

EMMANUEL FILIOT (ULB)

Based on joint works with

LEO EXIBARD



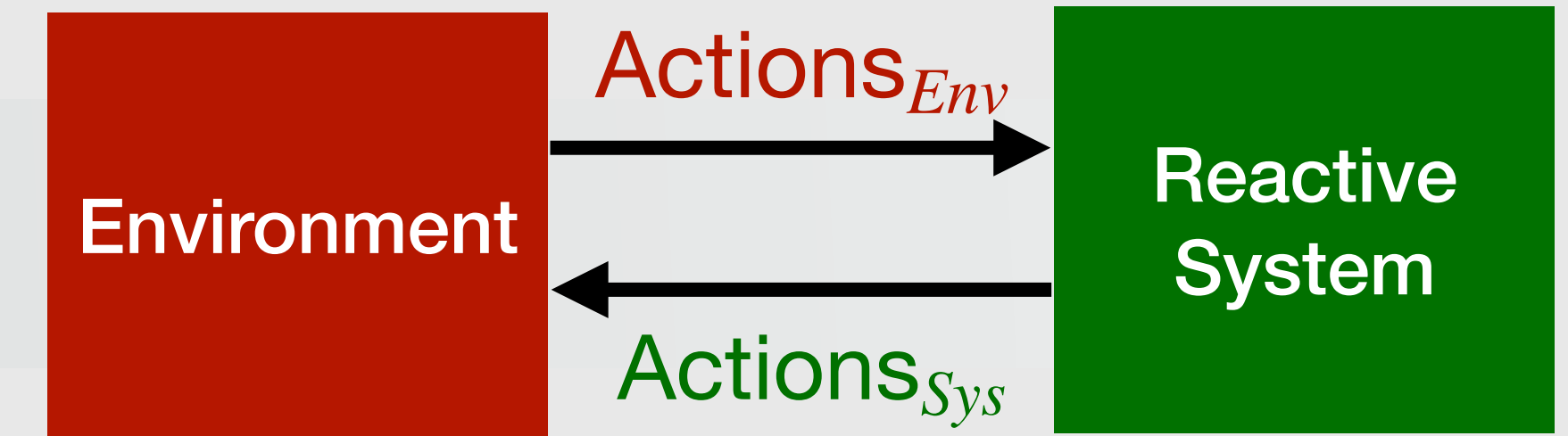
AYRAT KHALIMOV



PIERRE-ALAIN REYNIER

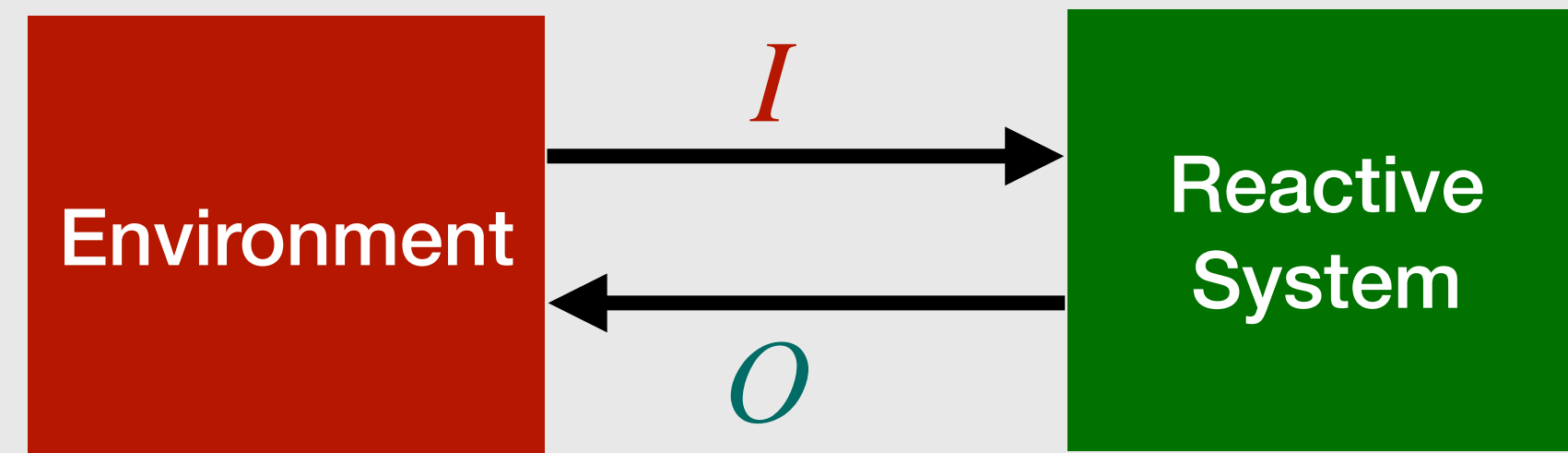


Intro



- **Reactive synthesis (RS):** automatically construct a reactive system from a specification of correct semantical behaviours $S \subseteq (\text{Actions}_{Env} \cdot \text{Actions}_{Sys})^\omega$
- **Formal methods for RS:** logic/automata/games, focus on control, ignore data
- **Objective:** *extend* formal methods for RS with data
- **In this talk:** $S \subseteq [(\text{Actions}_{Env} \times \mathcal{D}) \cdot (\text{Actions}_{Sys} \times \mathcal{D})]^\omega$
- **Questions:**
 - How to model specifications ? How to model reactive systems ?
 - What's decidable ? For which data domains ?

(Data-free) Reactive Synthesis Problem



$$i_1 \cdot o_1 \cdot i_2 \cdot o_2 \dots \in (I \cdot O)^\omega$$

Synthesis Problem

Input: a specification language $S \subseteq (IO)^\omega$

Output: a Mealy machine M such that $L(M) \subseteq S$

Example 1

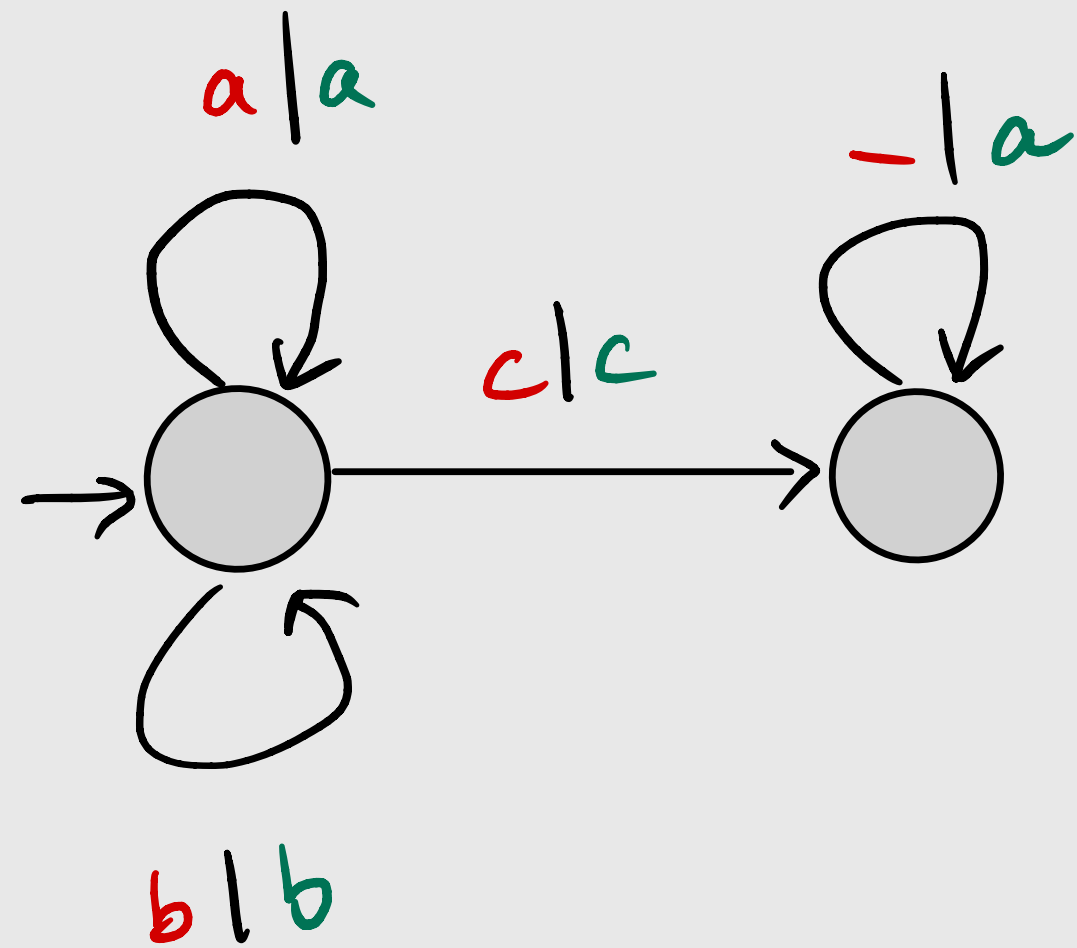
- $I = O = \{a, b, c\}$
- **Spec:** “*relay input up to first c, or forever if no c*”

$$S = (aa + bb)^\omega + (aa + bb)^*cc(IO)^\omega$$

Example 1

- $I = O = \{a, b, c\}$
- **Spec:** “*relay input up to first c, or forever if no c*”

$$S = (aa + bb)^\omega + (aa + bb)^*cc(IO)^\omega$$



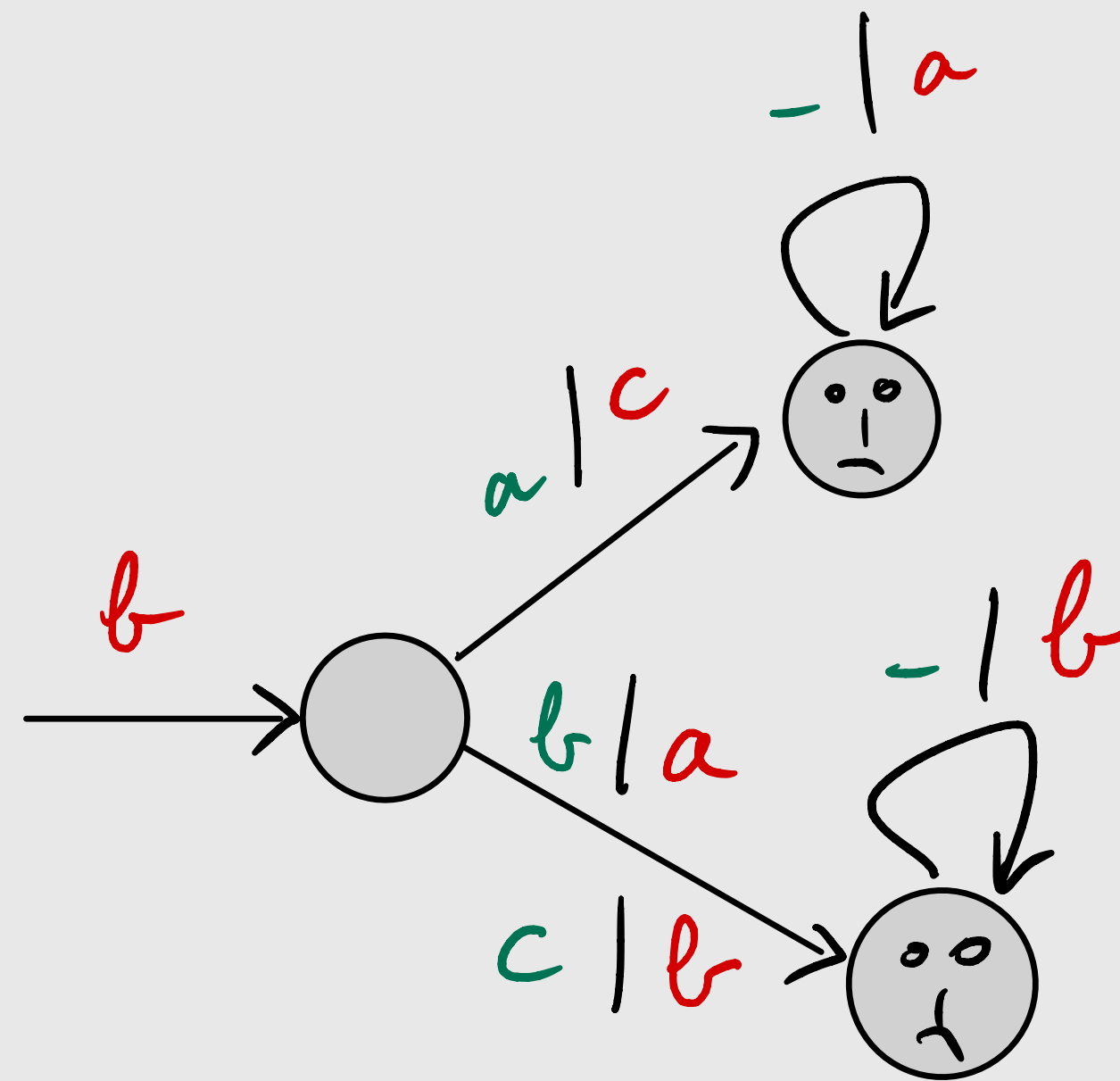
Example 2

- $I = O = \{a, b, c\}$
- **Spec:** $S = (aa + ba)^{\omega} + (aa + bb)^*cc(IO)^{\omega}$
- **Unrealizable !**

Example 2

- $I = O = \{a, b, c\}$
- Spec: $S = (aa + ba)^\omega + (aa + bb)^*cc(IO)^\omega$
- Unrealizable !

Environment can
realize \bar{S} with
a Moore machine

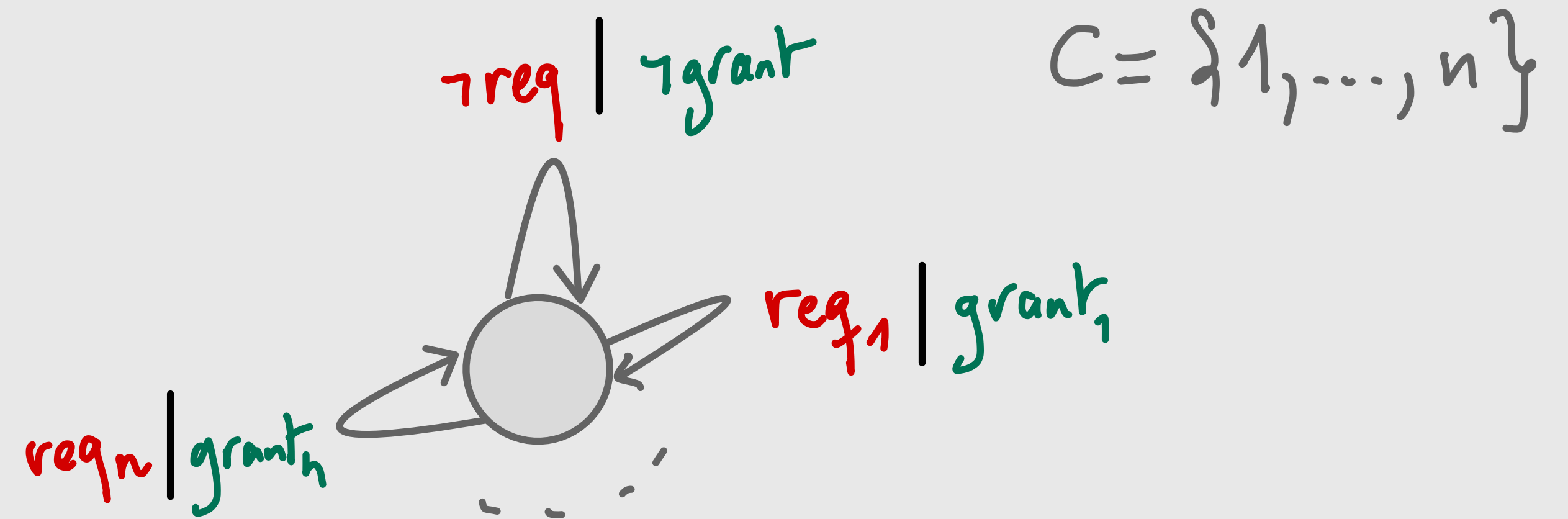


Example 3: request / grant

- $C \subseteq \mathbb{N}$ finite
- $I = \{req_i \mid i \in C\} \cup \{\neg req\}$
- $O = \{grt_i \mid i \in C\} \cup \{\neg grt\}$
- **Spec:** $\bigwedge_{i \in C} G(req_i \rightarrow F(grt_i))$

Example 3: request / grant

- $C \subseteq \mathbb{N}$ finite
- $I = \{req_i \mid i \in C\} \cup \{\neg req\}$
- $O = \{grt_i \mid i \in C\} \cup \{\neg grt\}$
- Spec: $\bigwedge_{i \in C} G(req_i \rightarrow F(grt_i))$

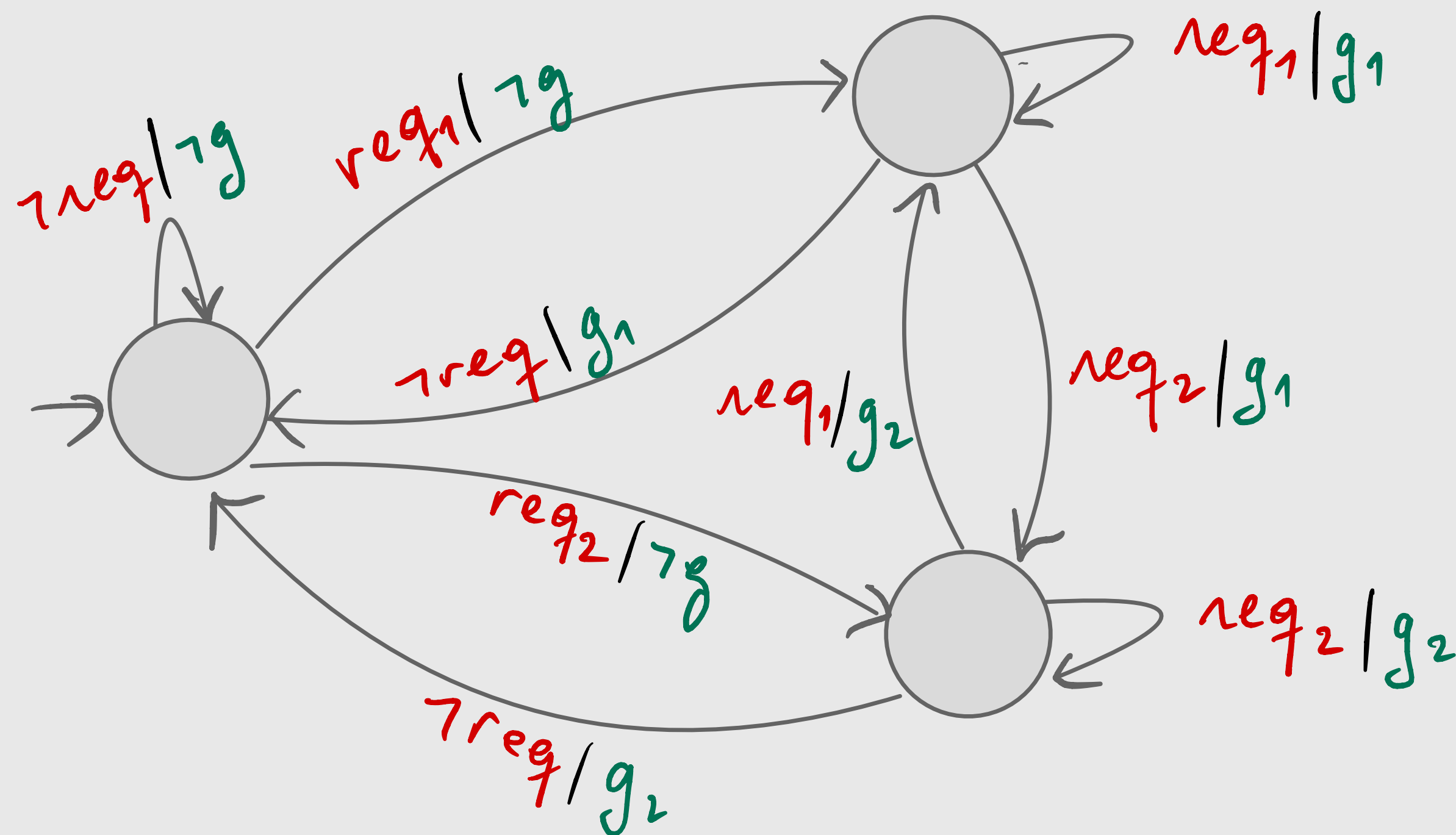


Example 4: request/grant with delay

- Any request must be granted after at least d input steps
- For $d=1$ and $|C|=2$:

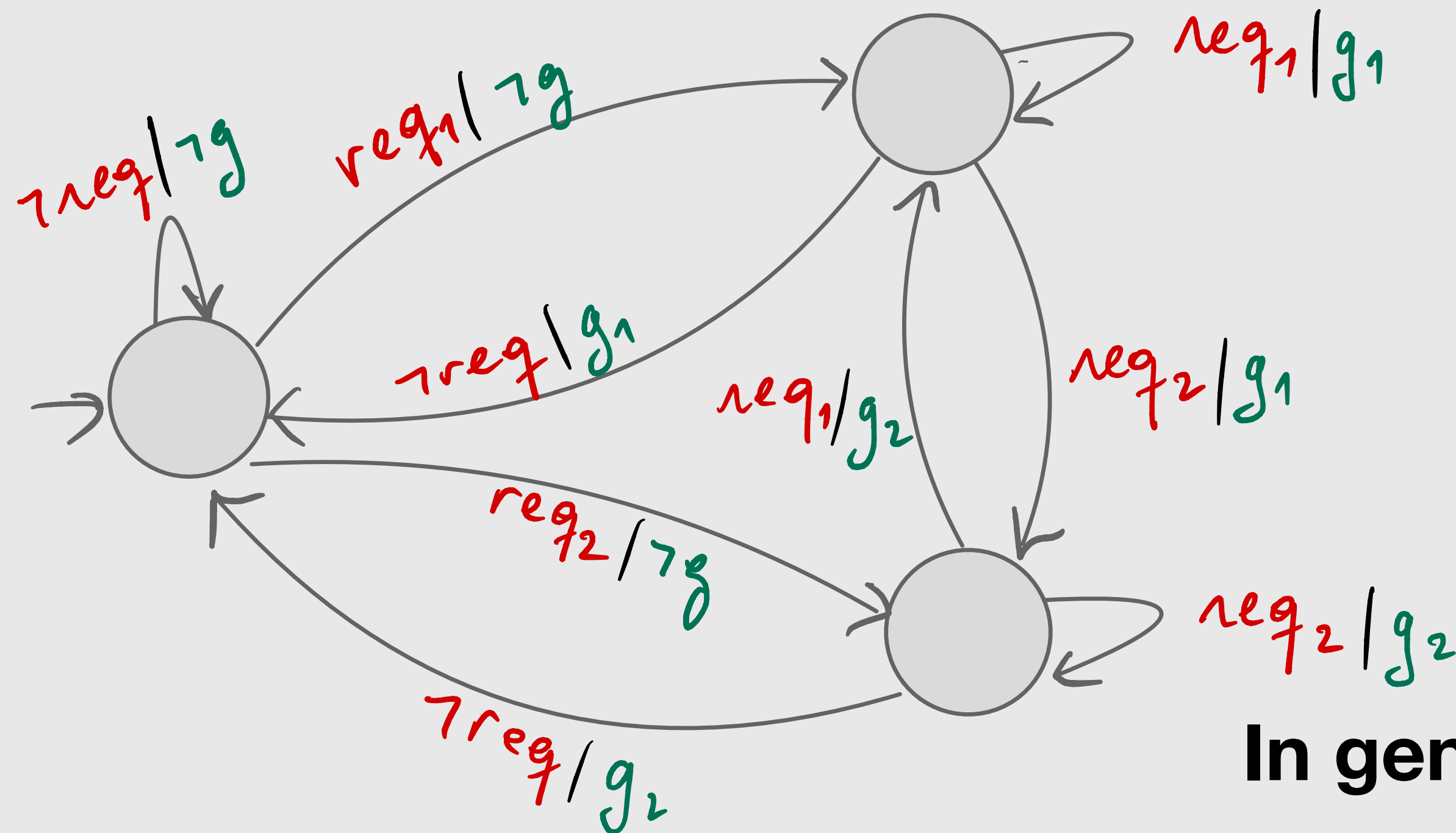
Example 4: request/grant with delay

- Any request must be granted after at least d input steps
- For $d=1$ and $|C|=2$: $C = \{1, 2\}$



Example 4: request/grant with delay

- Any request must be granted after at least d input steps
- For $d=1$ and $|C|=2$:



In general: machine with $O(|C|^d)$ states

Important results in reactive synthesis

- **Classical approach:** logic \rightarrow automata \rightarrow *deterministic* automata \rightarrow games
- **Regular spec:** LTL, MSO, non-det Büchi automata, universal coBüchi automata, det parity automata, good-for-games automata, ...

$2\text{exptime} - c$

$\text{exptime} - c$

Important results in reactive synthesis

- **Classical approach:** logic \rightarrow automata \rightarrow *deterministic* automata \rightarrow games
- **Regular spec:** LTL, MSO, non-det Büchi automata, universal coBüchi automata, det parity automata, good-for-games automata, ...

$2\text{exptime} - c$

$\text{exptime} - c$

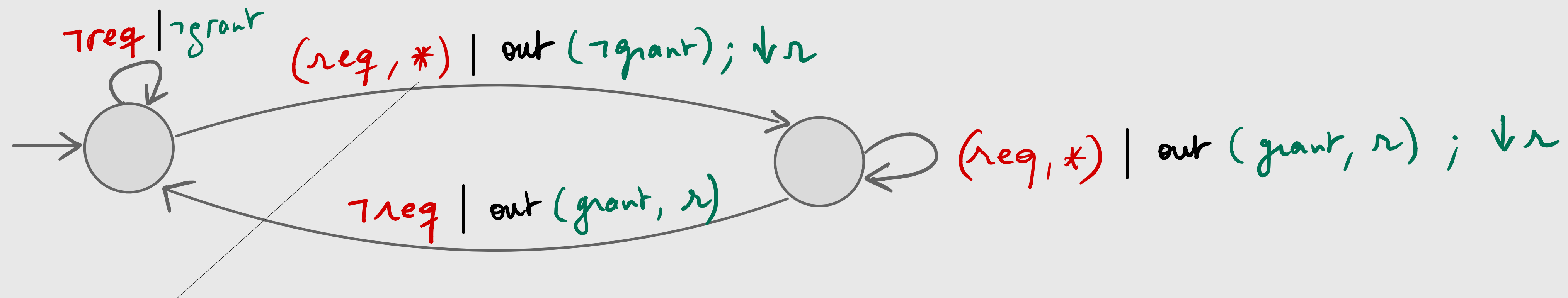
- **Game theory** on graphs
- **Tools:** Strix, LTLSynth, AcaciaBonzai, ...
- Yearly synthesis **competition** since 2014

Delayed request/grant example revisited

- **Spec:** “Any request must be granted after at least d input steps”
- ~~$C \subseteq \mathbb{N}$~~ $C = \mathbb{N}$
- $I = \{(req, i) \mid i \in C\} \cup \{\neg req\}$
- $O = \{(grt, i) \mid i \in C\} \cup \{\neg grt\}$

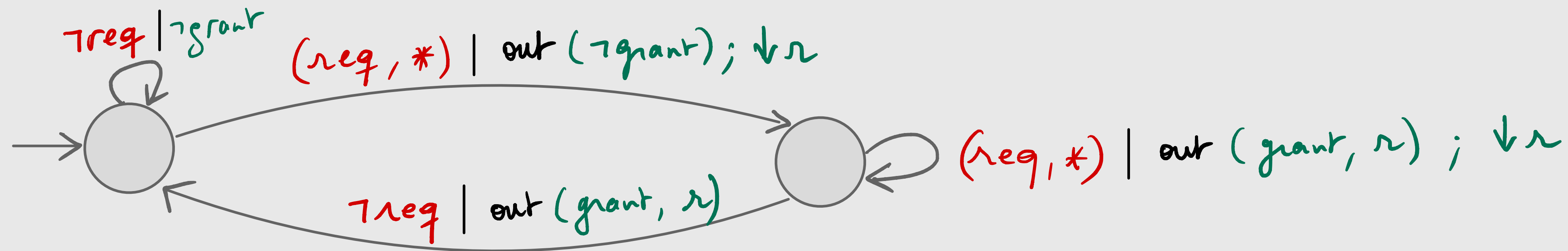
Delayed request/grant example revisited

- **Spec:** “Any request must be granted after at least d input steps”
- $C \subseteq \mathbb{N}$ $C = \mathbb{N}$
- $I = \{(req, i) \mid i \in C\} \cup \{\neg req\}$
- $O = \{(grt, i) \mid i \in C\} \cup \{\neg grt\}$
- Spec realizable by a Mealy machine **with registers**. For $d=1$:



Delayed request/grant example revisited

- **Spec:** “Any request must be granted after at least d input steps”
- $C \subseteq \mathbb{N}$ $C = \mathbb{N}$
- $I = \{(req, i) \mid i \in C\} \cup \{\neg req\}$
- $O = \{(grt, i) \mid i \in C\} \cup \{\neg grt\}$
- Spec realizable by a Mealy machine **with registers**. For $d=1$:



- in general: Mealy machine with $O(d)$ states and $O(d)$ registers
- Priorities between processes: data domain (\mathbb{N}, \leq)

Synthesis Problem over Infinite Data Domains

Definition

Input: a specification language $S \subseteq \mathcal{D}^\omega$ where \mathcal{D} is a data domain

Output: a Mealy machine **with registers** M such that $L(M) \subseteq S$

Specification: FO with \leq_d , constraint LTL, LTL with freeze quantifier, variable automata, det/nondet/**universal register automata**, ...

Register Automata on $(\mathbb{N}, \leq, 0)$

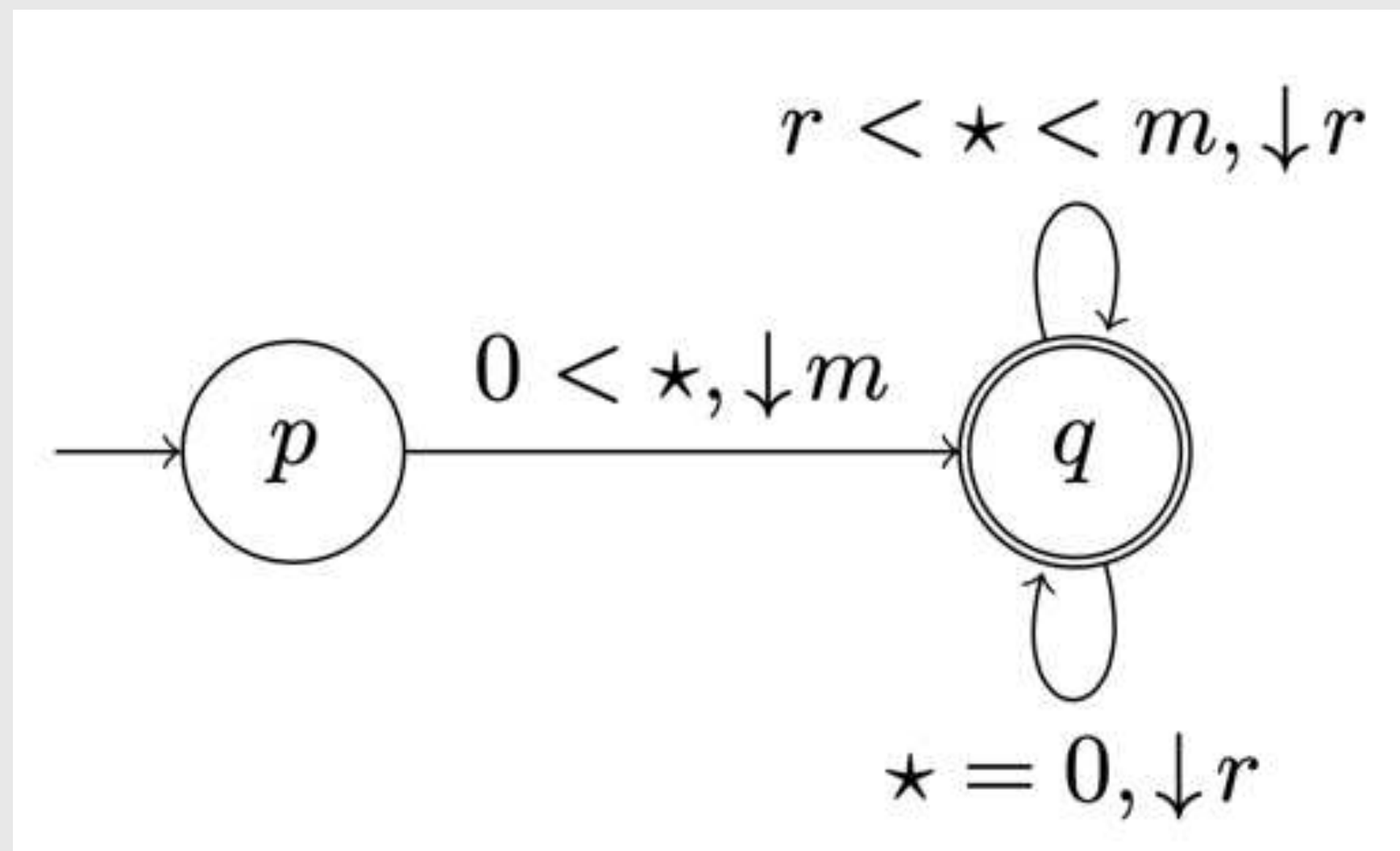
$$A = (Q, q_0, R, \delta, \alpha)$$

states

registers

transitions

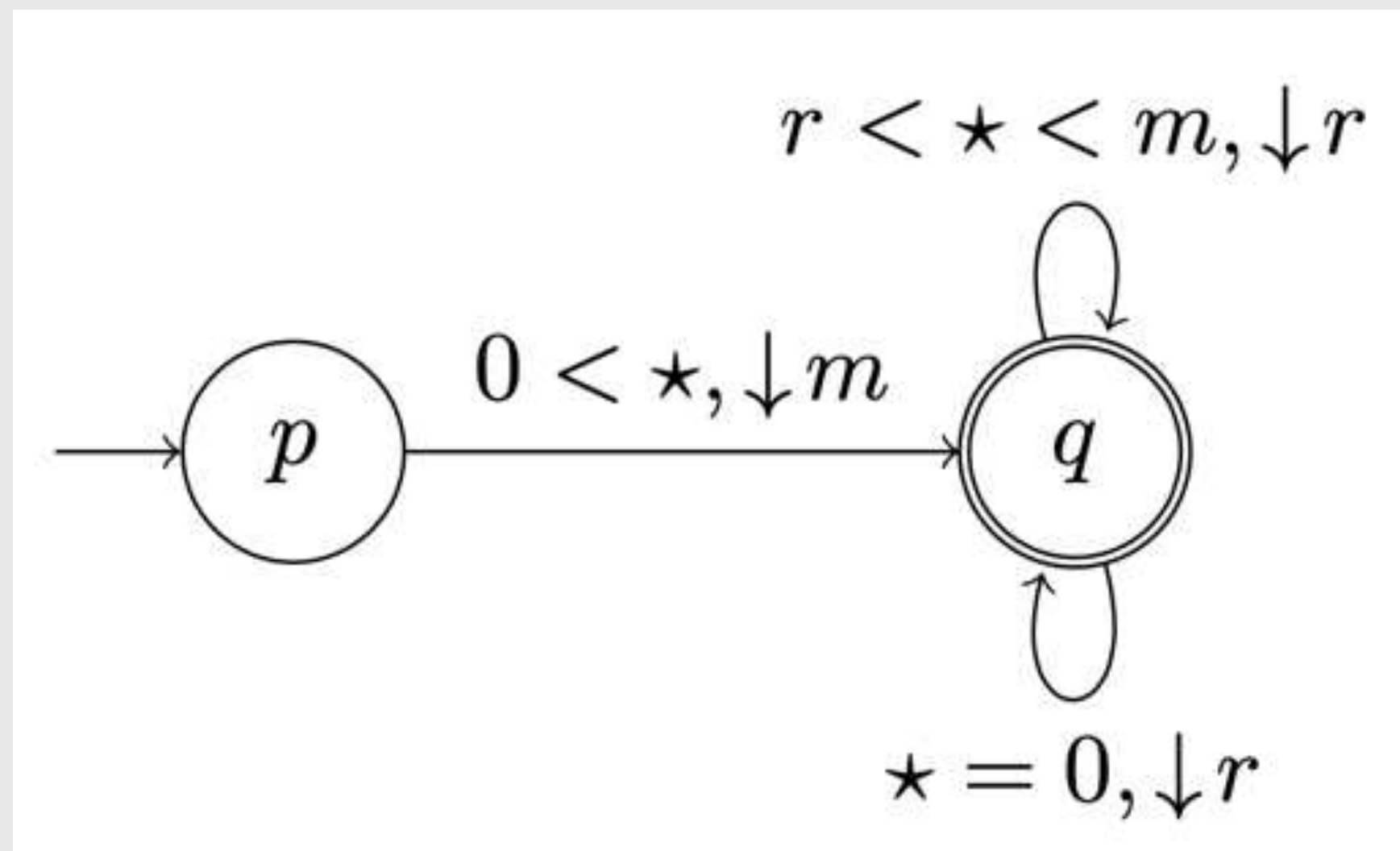
parity acceptance



$$\delta \subseteq Q \times \text{Tests}(R) \times \text{Assignments} \times Q$$

Register Automata on $(\mathbb{N}, \leq, 0)$

$$A = (Q, q_0, R, \delta, \alpha)$$



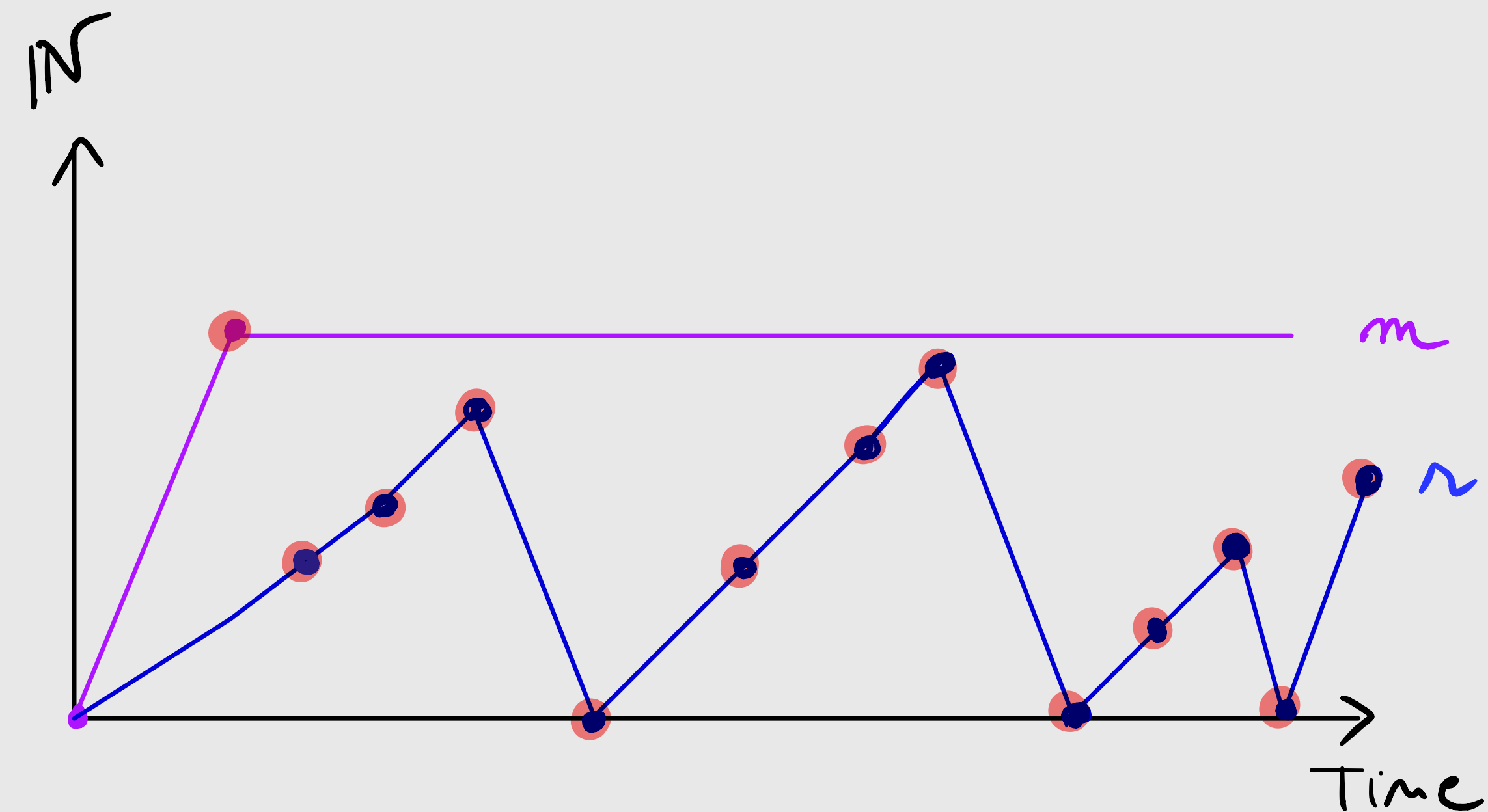
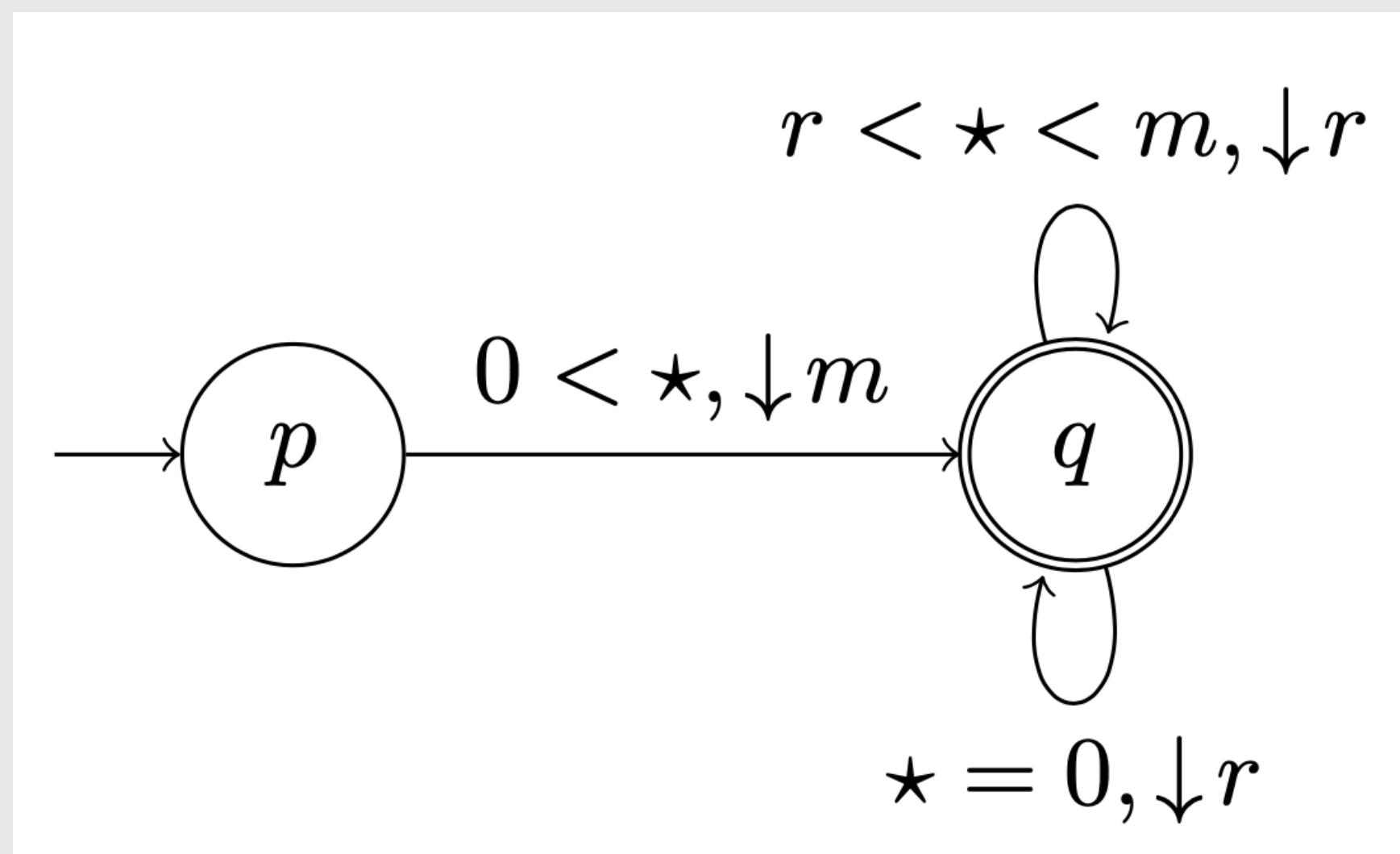
Run: $(p, 0, 0) \xrightarrow{5} (q, 5, 0) \xrightarrow{3} (q, 5, 3) \xrightarrow{4} (q, 5, 4) \xrightarrow{0} (q, 5, 0)$

\uparrow state \uparrow m \uparrow \sim

input data

Register Automata on $(\mathbb{N}, \leq, 0)$

$$A = (Q, q_0, R, \delta, \alpha)$$



Run: $(p, 0, 0) \xrightarrow[5]{\text{input data}} (q, 5, 0) \xrightarrow{3} (q, 5, 3) \xrightarrow{4} (q, 5, 4) \xrightarrow{0} (q, 5, 0)$

Annotations for the first tuple $(p, 0, 0)$:

- p : state
- 0 : m
- 0 : \sim

Results

Synthesis problem

Input: a universal register automaton A over \mathcal{D}
Output: a Mealy machine **with registers** M such that $L(M) \subseteq L(A)$

| | $(\mathbb{D}, =)$ | $(\mathbb{Q}, <)$ | $(\mathbb{N}, <)$ |
|------------------|-------------------|-------------------|-------------------|
| Synthesis | × [3] | × | × |

[3]: L. Exibard: Automatic Synthesis of Systems with Data (PhD thesis), 2021.

Results

Synthesis problem

Input: a universal register automaton A over \mathcal{D}
Output: a Mealy machine **with registers** M such that $L(M) \subseteq L(A)$

Register-Bounded Synthesis problem

Input: a universal register automaton A over \mathcal{D} and $k \in \mathbb{N}$
Output: a Mealy machine **with k registers** M such that $L(M) \subseteq L(A)$

| | $(\mathbb{D}, =)$ | $(\mathbb{Q}, <)$ | $(\mathbb{N}, <)$ |
|-----------|-------------------|-------------------|-------------------|
| Synthesis | ✗ [3] | ✗ | ✗ |

[3]: L. Exibard: Automatic Synthesis of Systems with Data (PhD thesis), 2021.

Results

Synthesis problem

Input: a universal register automaton A over \mathcal{D}
Output: a Mealy machine **with registers** M such that $L(M) \subseteq L(A)$

Register-Bounded Synthesis problem

Input: a universal register automaton A over \mathcal{D} and $k \in \mathbb{N}$
Output: a Mealy machine **with k registers** M such that $L(M) \subseteq L(A)$

| | $(\mathbb{D}, =)$ | $(\mathbb{Q}, <)$ | $(\mathbb{N}, <)$ |
|-----------------------------------|-------------------|-------------------|-------------------|
| Synthesis | ✗ [3] | ✗ | ✗ |
| Register-bounded synthesis | ✓ [1,2] | ✓ [3] | ✓ [4] |

[1]: R.Bloem, B.Maderbacher, A.Khalimov.: Bounded Synthesis of Register Transducers. 2019
[2]: L.Exibard, E.F., P.-A. Reynier: Synthesis of data word transducers. 2019.
[3]: L. Exibard: Automatic Synthesis of Systems with Data (PhD thesis), 2021.
[4]: L. Exibard, E.F., A. Khalimov: Generic solution to register-bounded synthesis. 2022.

Generic Solution to Register-Bounded Synthesis

Thanks to Ayrat for the slides !

Main ideas

- **reduction** to omega-regular synthesis over **finite** alphabets
- **sufficient condition** on the data domain allowing for such a reduction (regapprox domains)
- **prove** that $(\mathbb{Q}, <)$ and $(\mathbb{N}, <)$ are regapprox

ABSTRACTION

$$\delta \subseteq Q \times \underbrace{\text{Tests} \times \text{Assignments} \times Q}_{\substack{\text{finite alphabet} \\ \text{of } \underline{\text{actions}}}}$$

\Rightarrow see register automata as
finite automata over action words

ABSTRACTION

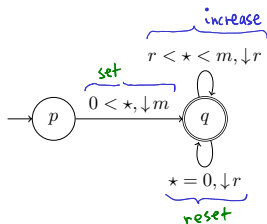
$$\delta \subseteq Q \times \underbrace{\text{Tests} \times \text{Assignments} \times Q}_{\substack{\text{finite alphabet} \\ \text{of } \underline{\text{actions}}}}$$

\Rightarrow see register automata as
finite automata over action words

NOTATION: $L_{\text{syntax}}(A) \stackrel{\text{def}}{=} \text{set of action words accepted by } A$

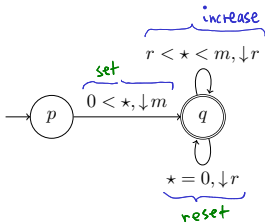
register automaton
↓

ABSTRACTION



set (increase)^ω

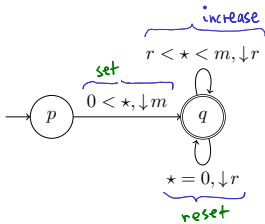
ABSTRACTION



set (increase)^ω

set incr reset incr² reset incr³ reset ...

ABSTRACTION



set (increase)^ω

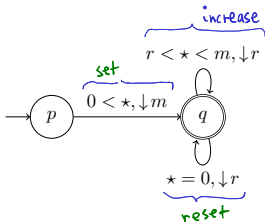
set incr reset incr² reset incr³ reset ...

FEASIBLE ON $w \in 2^{\omega}$

:

"the sequence of actions can be executed on w "

ABSTRACTION



set (increase)^ω

set incr reset incr² reset incr³ reset ...

→ NOT FEASIBLE BY ANY DATA WORD
ON $(\mathbb{N}, <)$!

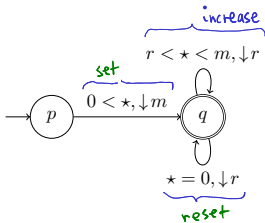
→ BUT FEASIBLE ON $(\mathbb{Q}, <)$

FEASIBLE ON WE 2)^ω

:

"the sequence of
actions can be
executed on w"

ABSTRACTION

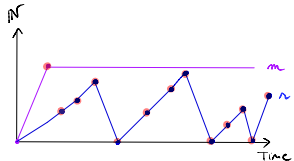


set (increase)^ω

set incr reset incr² reset incr³ reset ...
on $(\mathbb{N}, <)$

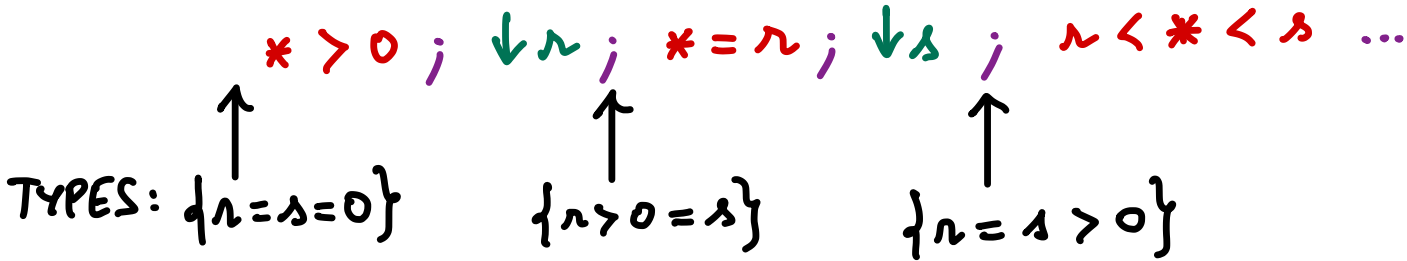
Any feasible action word has the form

set incr^{n₁} reset incr^{n₂} ... where $\exists B$: every $n_i < B$



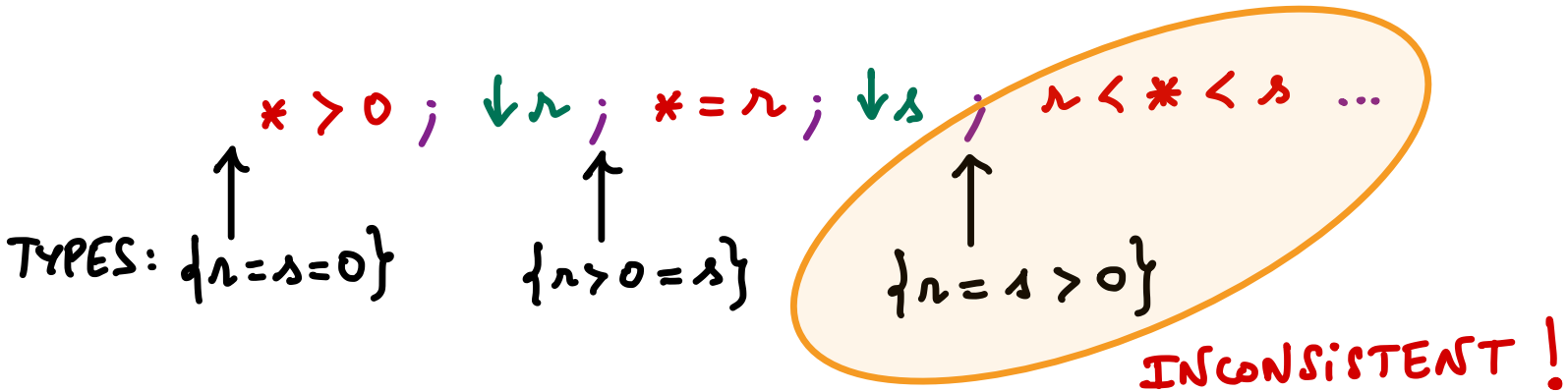
FEASIBILITY IN $(\mathbb{Q}, <)$

- Feasibility = local consistency



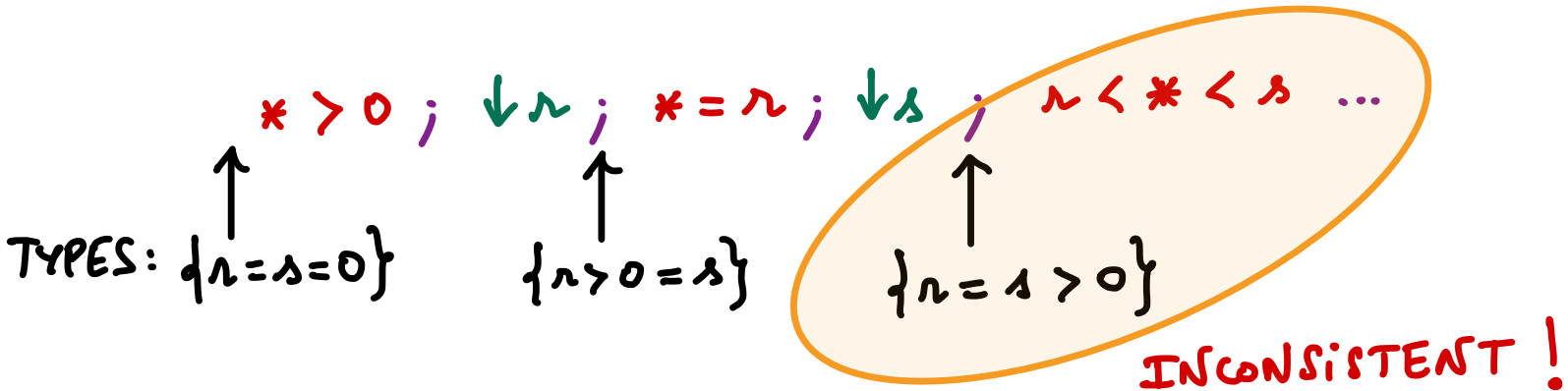
FEASIBILITY IN $(\mathbb{Q}, <)$

- Feasibility = local consistency



FEASIBILITY IN $(\mathbb{Q}, <)$

- Feasibility = local consistency



- **Lemma:** the set of feasible words is omega-regular

FEASIBILITY IN $(\mathbb{N}, <)$

An action word is feasible iff it has:

no inf decreasing chains



no unbounded chains of the form



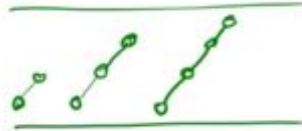
FEASIBILITY IN $(\mathbb{N}, <)$

An action word is feasible iff it has:

no inf decreasing chains



no unbounded chains of the form



Let FEAS be the set of feasible action words over given R .

REDUCTION TO FINITE ALPHABETS



Given S and k , create a *finite-alphabet* specification $W_{S,k}$:

$W_{S,k}$ is realizable by a Mealy machine

\Leftrightarrow

S is realizable by a Mealy machine

REDUCTION TO FINITE ALPHABETS



Given S and k , create a *finite-alphabet* specification $W_{S,k}$:

$W_{S,k}$ is realizable by a Mealy machine

\Leftrightarrow

S is realizable by a Mealy machine

$$\begin{aligned}\overline{W_{S,k}} &= \{\bar{a}_M \mid \exists \bar{a}_S \notin L_{\text{syntax}}(S) \cdot \exists w \in \mathcal{D}^\omega \cdot w \models \bar{a}_M \wedge w \models \bar{a}_S\} \\ &= \{\bar{a}_M \mid \exists \bar{a}_S \notin L_{\text{syntax}}(S) \cdot \bar{a}_M \otimes \bar{a}_S \in \text{FEAS}\}\end{aligned}$$

REDUCTION TO FINITE ALPHABETS



Given S and k , create a *finite-alphabet* specification $W_{S,k}$:

$W_{S,k}$ is realizable by a Mealy machine

\Leftrightarrow

S is realizable by a Mealy machine

$$\begin{aligned}\overline{W_{S,k}} &= \{\bar{a}_M \mid \exists \bar{a}_S \notin L_{\text{syntax}}(S) \cdot \exists w \in \mathcal{D}^\omega \cdot w \models \bar{a}_M \wedge w \models \bar{a}_S\} \\ &= \{\bar{a}_M \mid \exists \bar{a}_S \notin L_{\text{syntax}}(S) \cdot \bar{a}_M \otimes \bar{a}_S \in \text{FEAS}\}\end{aligned}$$

NOT ω -REGULAR IN GENERAL !

GENERIC SOLUTION

Data domain is *regapprox* if for every R there exists eff.constr. ω -regular over-approximation QFEAS

$$\text{QFEAS} \cap \text{lasso} \subseteq \text{FEAS} \subseteq \text{QFEAS}.$$



Theorem:

on regapprox domains, register-bounded synthesis is decidable.

PROOF IDEA

Lemma

S is realizable by a k -reg Mealy machine iff

$W_{S,k}$ is realisable by a Mealy machine iff

$W_{S,k}^{QF}$ is realizable by a Mealy machine



S is defined by a URA

$$W_{S,k} = \{\bar{a}_M \mid \forall \bar{a}_S \cdot \bar{a}_M \otimes \bar{a}_S \in \text{FEAS} \Rightarrow \bar{a}_S \in L_{\text{syntax}}(S)\}$$

$$W_{S,k}^{QF} = \{\bar{a}_M \mid \forall \bar{a}_S \cdot \bar{a}_M \otimes \bar{a}_S \in \text{QFEAS} \Rightarrow \bar{a}_S \in L_{\text{syntax}}(S)\}$$

As $\text{FEAS} \subseteq \text{QFEAS}$, $W_{S,k}^{QF} \subseteq W_{S,k}$

So, $W_{S,k}^{QF}$ is harder to realize.

PROOF IDEA

Lemma

S is realizable by a k-reg Mealy machine iff

$W_{S,k}$ is realisable by a Mealy machine iff

$W_{S,k}^{QF}$ is realizable by a Mealy machine



S is defined by a URA

$$W_{S,k} = \{\bar{a}_M \mid \forall \bar{a}_S \cdot \bar{a}_M \otimes \bar{a}_S \in \text{FEAS} \Rightarrow \bar{a}_S \in L_{\text{syntax}}(S)\}$$

$$W_{S,k}^{QF} = \{\bar{a}_M \mid \forall \bar{a}_S \cdot \bar{a}_M \otimes \bar{a}_S \in \text{QFEAS} \Rightarrow \bar{a}_S \in L_{\text{syntax}}(S)\}$$

Let M s.t. $L(M) \not\subseteq L(W_{S,k}^{QF})$.

$$\exists \bar{a}_M \otimes \bar{a}_S \in \left(\text{QFEAS} \cap L(M) \otimes \overline{L_{\text{syntax}}(S)} \right)$$



PROOF IDEA

Lemma

S is realizable by a k -reg Mealy machine iff

$W_{S,k}$ is realisable by a Mealy machine iff

$W_{S,k}^{QF}$ is realizable by a Mealy machine



S is defined by a URA

$$W_{S,k} = \{\bar{a}_M \mid \forall \bar{a}_S \cdot \bar{a}_M \otimes \bar{a}_S \in \text{FEAS} \Rightarrow \bar{a}_S \in L_{\text{syntax}}(S)\}$$

$$W_{S,k}^{QF} = \{\bar{a}_M \mid \forall \bar{a}_S \cdot \bar{a}_M \otimes \bar{a}_S \in \text{QFEAS} \Rightarrow \bar{a}_S \in L_{\text{syntax}}(S)\}$$

Let M s.t. $L(M) \not\subseteq L(W_{S,k}^{QF})$.

$$\exists \bar{a}_M \otimes \bar{a}_S \in \underbrace{(\text{QFEAS} \cap L(M) \otimes \overline{L_{\text{syntax}}(S)})}_{w\text{-reg}} \cap \text{LASSO}$$



PROOF IDEA

Lemma

S is realizable by a k -reg Mealy machine iff

$W_{S,k}$ is realisable by a Mealy machine iff

$W_{S,k}^{QF}$ is realizable by a Mealy machine



S is defined by a URA

$$W_{S,k} = \{\bar{a}_M \mid \forall \bar{a}_S \cdot \bar{a}_M \otimes \bar{a}_S \in \text{FEAS} \Rightarrow \bar{a}_S \in L_{\text{syntax}}(S)\}$$

$$W_{S,k}^{QF} = \{\bar{a}_M \mid \forall \bar{a}_S \cdot \bar{a}_M \otimes \bar{a}_S \in \text{QFEAS} \Rightarrow \bar{a}_S \in L_{\text{syntax}}(S)\}$$

Let M s.t. $L(M) \not\subseteq L(W_{S,k}^{QF})$.

$$\exists \bar{a}_M \otimes \bar{a}_S \in \underbrace{(\text{QFEAS} \cap L(M) \otimes \overline{L_{\text{syntax}}(S)})}_{w\text{-reg}} \cap \text{LASSO}$$

$$\exists \bar{a}_M \otimes \bar{a}_S \in (\cancel{\text{FEAS}} \cap L(M) \otimes \overline{L_{\text{syntax}}(S)})$$

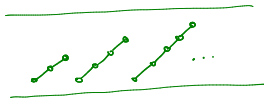
(because $\text{QFEAS} \cap \text{LASSO} = \text{FEAS} \cap \text{LASSO}$). So, $L(M) \not\subseteq W_{S,k}$

DOMAIN $(\mathbb{N}, <)$ is regapprox!

Recall that in $(\mathbb{N}, <)$ action word is feasible iff there are no:

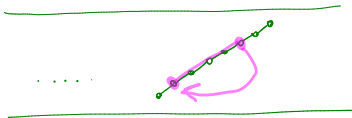


inf decr
chains



unbounded
chains

When considering lasso words:



\Rightarrow enough to check absence
of inf chains



MAIN THEOREM


Reg-bounded synthesis in $(\mathbb{N}, <)$ is solvable in time
$$\exp(\exp(r, k), n, c)$$
for every given universal parity register automaton with r registers, n states, c priorities, and bound k .

A similar complexity holds for domains $(\mathbb{Q}, <)$ and $(\mathbb{D}, =)$.

REDUCTION BETWEEN DOMAINS

If \mathcal{D} reduces to \mathcal{D}' , and \mathcal{D}' is regapprox, then \mathcal{D} is regapprox.

Two definitions of reductions:

- via transducer relations, 
- via first-order formulas.

action word in \mathcal{D} \rightsquigarrow action words in \mathcal{D}'
 $\text{feas} \Rightarrow \exists \text{feas}$

Allows us to state decidability of register-bounded synthesis for $(\mathbb{N}^d, <^d)$ and (Σ^*, \prec) .

 prefix relation

Conclusion

| | $(\mathbb{D}, =)$ | $(\mathbb{Q}, \overset{<}{\neq})$ | $(\mathbb{N}, <)$ |
|----------------------------|-------------------|-----------------------------------|-------------------|
| Synthesis | ✗ [3] | ✗ | ✗ |
| Register-bounded synthesis | ✓ [1,2] | ✓ [3] | ✓ [4] |

→ true for $|R| \geq 2$
 $|R| = 1$??

- **not in this talk:** synthesis is decidable for *deterministic* register-automata
- **future directions / open questions:**
 - other data domains: strings with subword relation, sets of natural numbers with inclusion, ...
 - decidable data-synthesis framework capturing realistic request/grant example
 - parameterised synthesis
 - logical specifications: undec for $\text{FO}_2[<_{\text{pos}}, \text{succ}_{\text{pos}}, =_{\text{data}}]$, what about $\text{FO}_2[<_{\text{pos}}, =_{\text{data}}]$?
 - implementation: refinement techniques