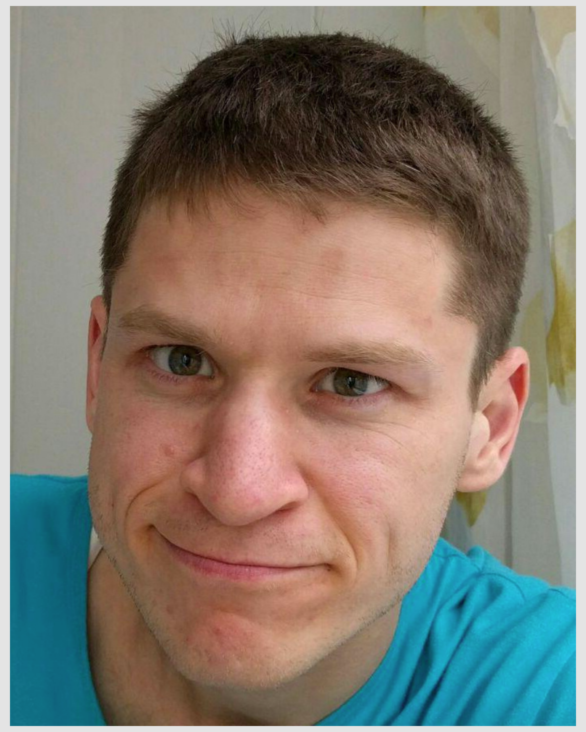
REACTIVE SYNTHESIS OVER INFINITE DATA DOMAINS

EMMANUEL FILIOT (ULB)

Based on joint works with

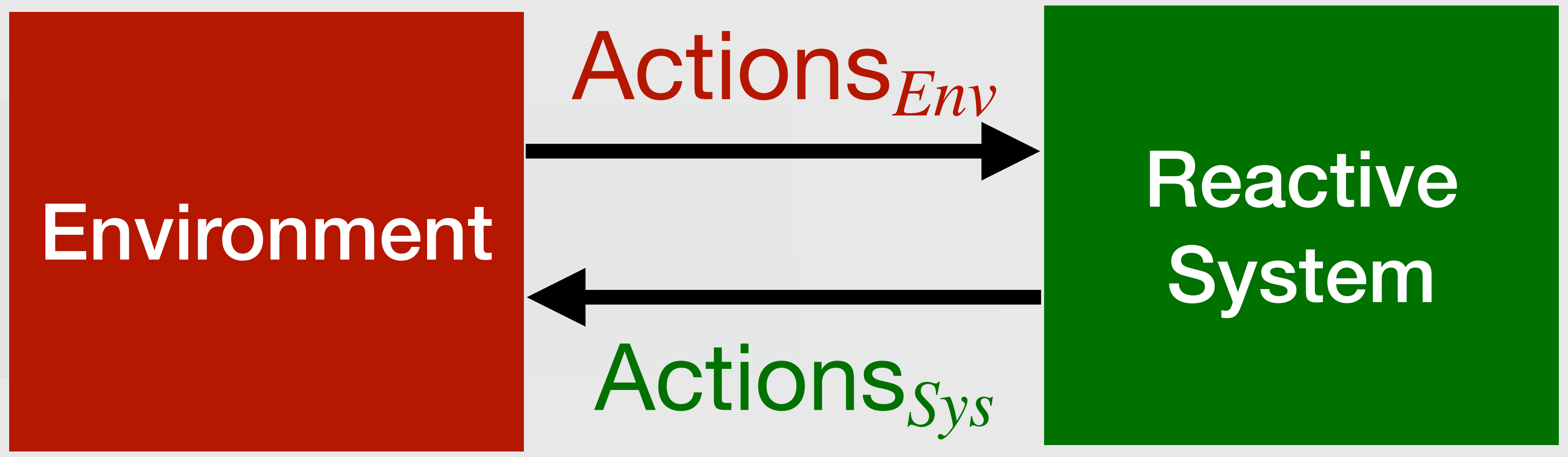
LEO EXIBARD

AYRAT KHALIMOV



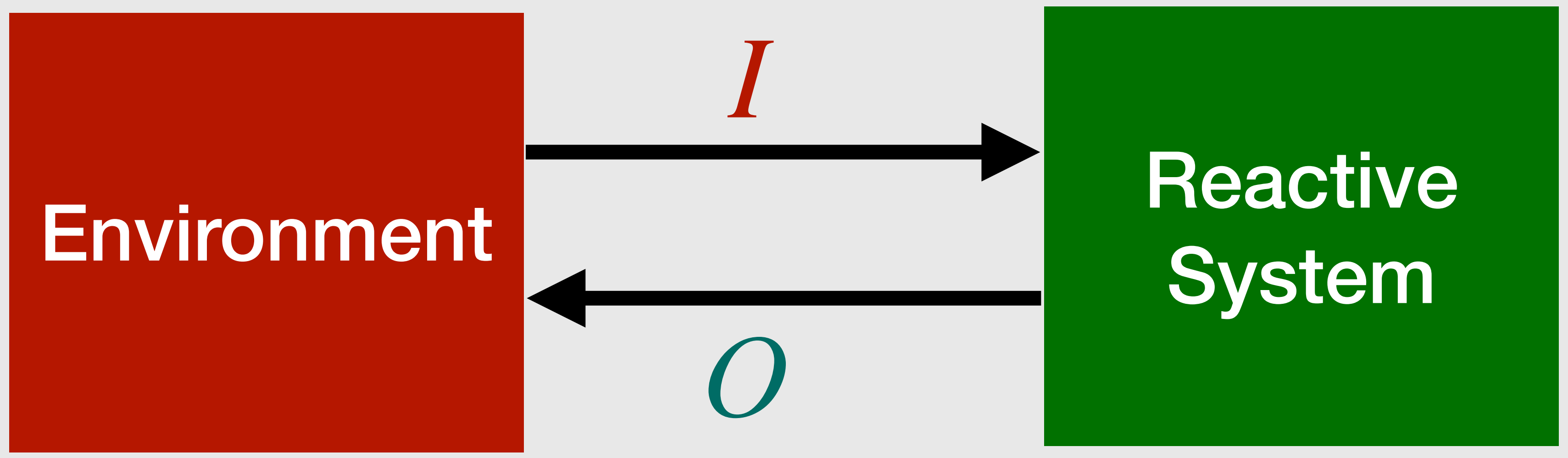
PIERRE-ALAIN REYNIER

Intro



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

(Data-free) Reactive Synthesis Problem



$$i_1 \,.\, o_1 \,.\, i_2 \,.\, o_2 \ldots \in (I \,.\, O)^\omega$$

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"*

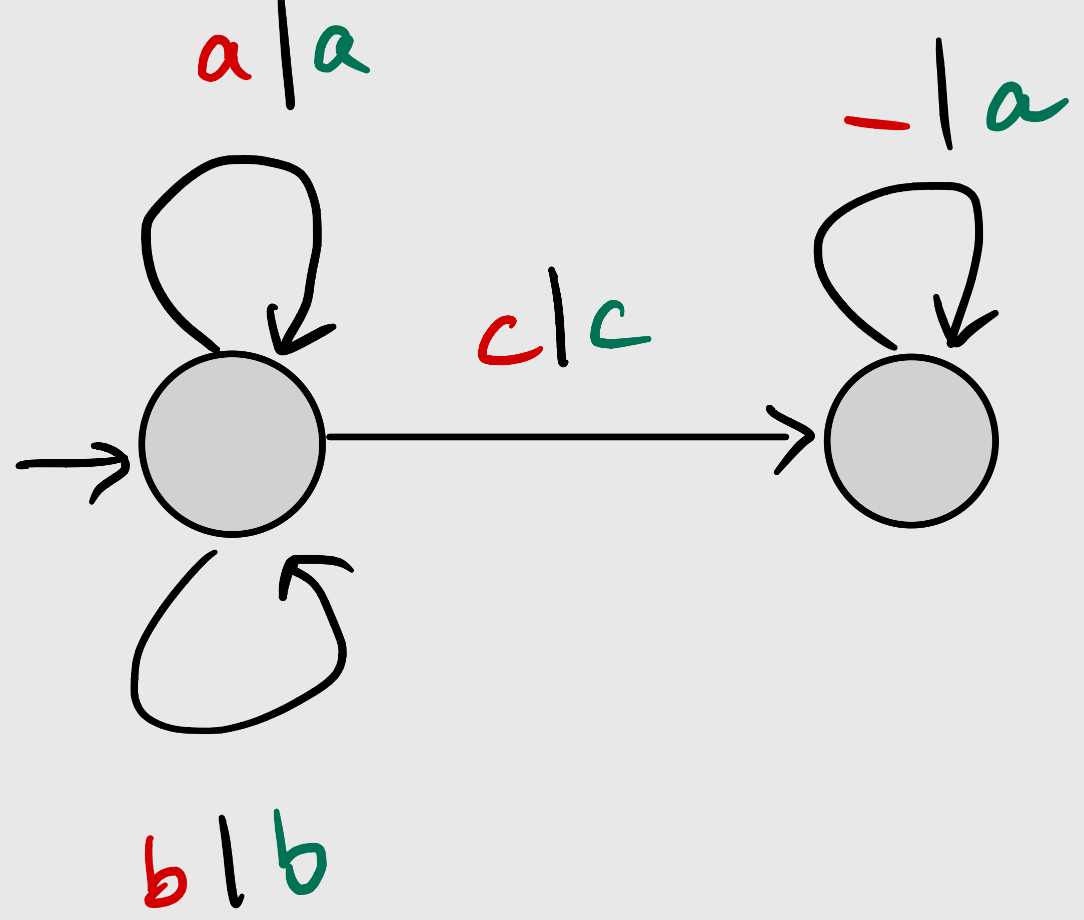
$$S = (aa + bb)^{\omega} + (aa + bb)^{*} cc(IO)^{\omega}$$

Example 1

- $I = O = \{a, b, c\}$

- **Spec:** *"relay input up to first c, or forever if no c"*

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



$f : I^{*} \to O$

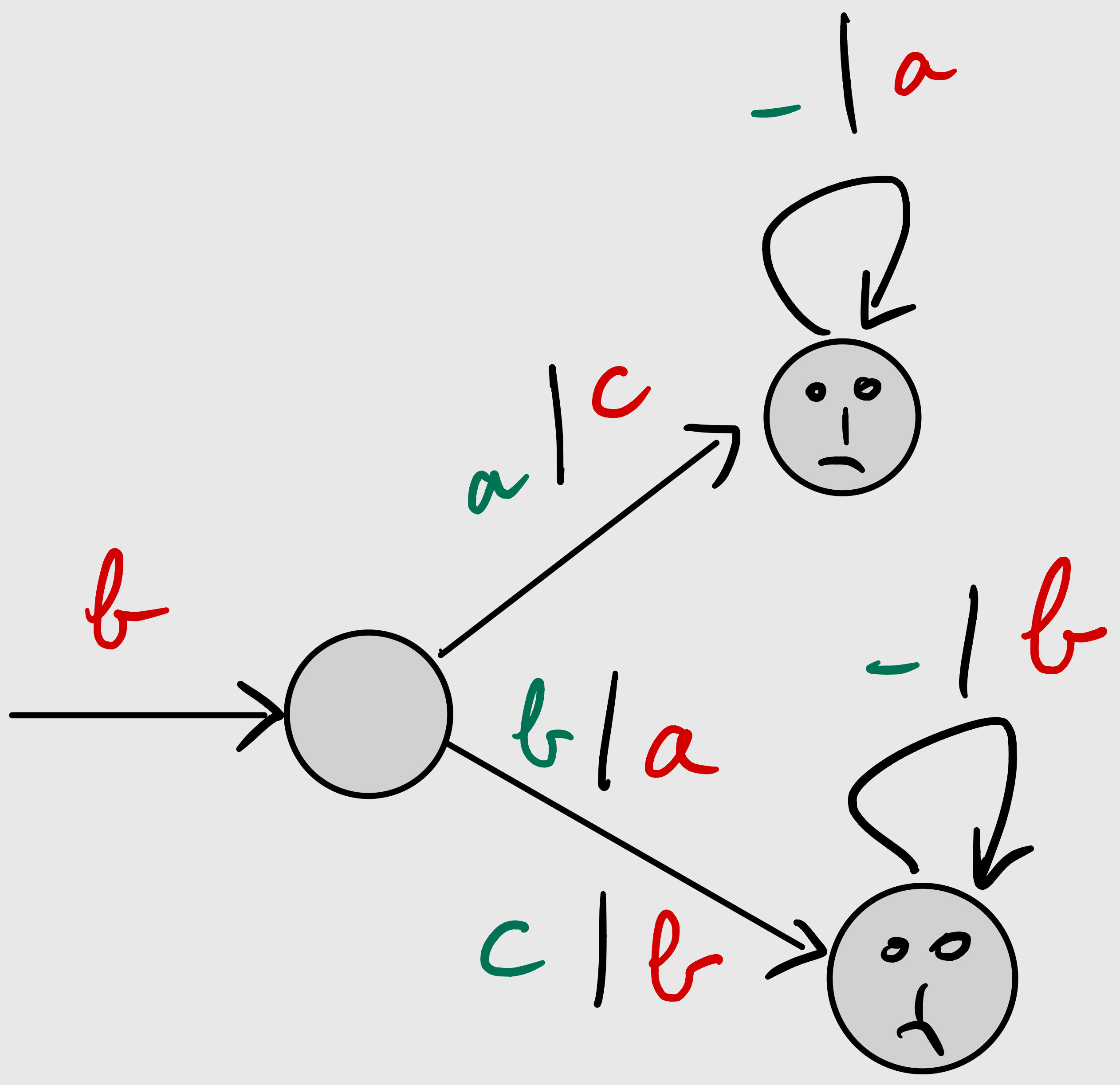
Example 2

- $I = O = \{a, b, c\}$
- **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

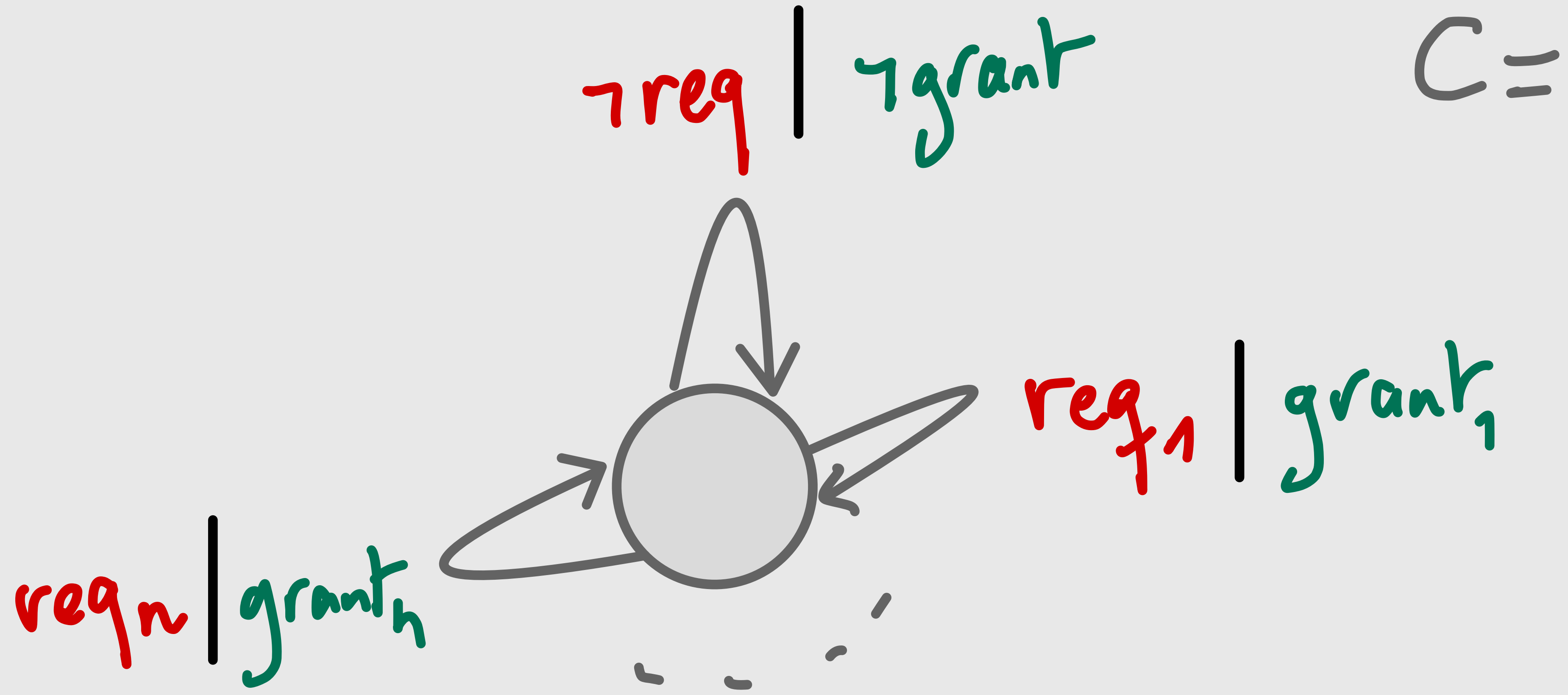


Example 3: request / grant

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

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:** $\bigwedge_{i \in C} G(req_i \rightarrow F(grt_i))$



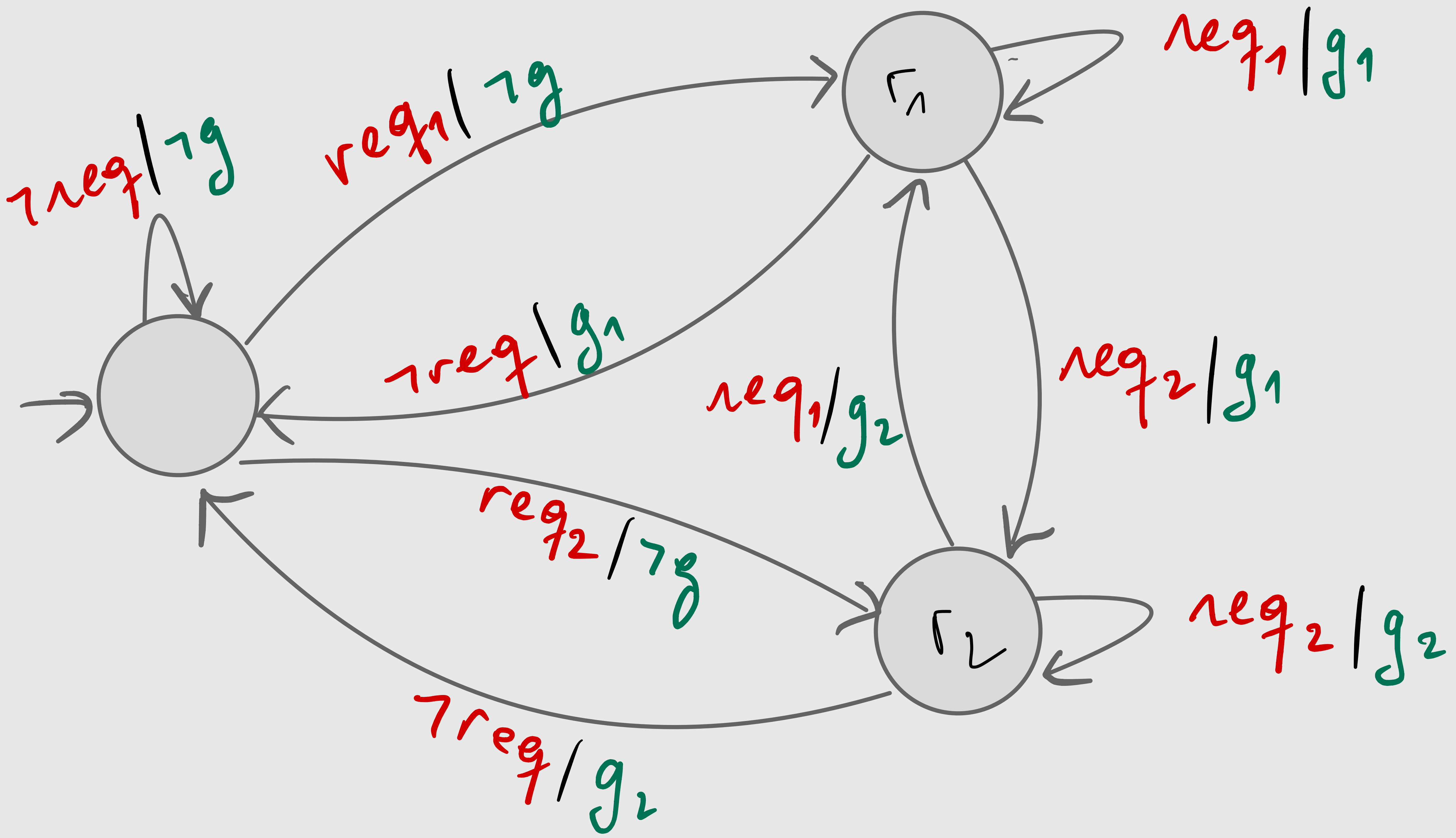
$C = \{1, \dots, n\}$

Example 4: request/grant with delay

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

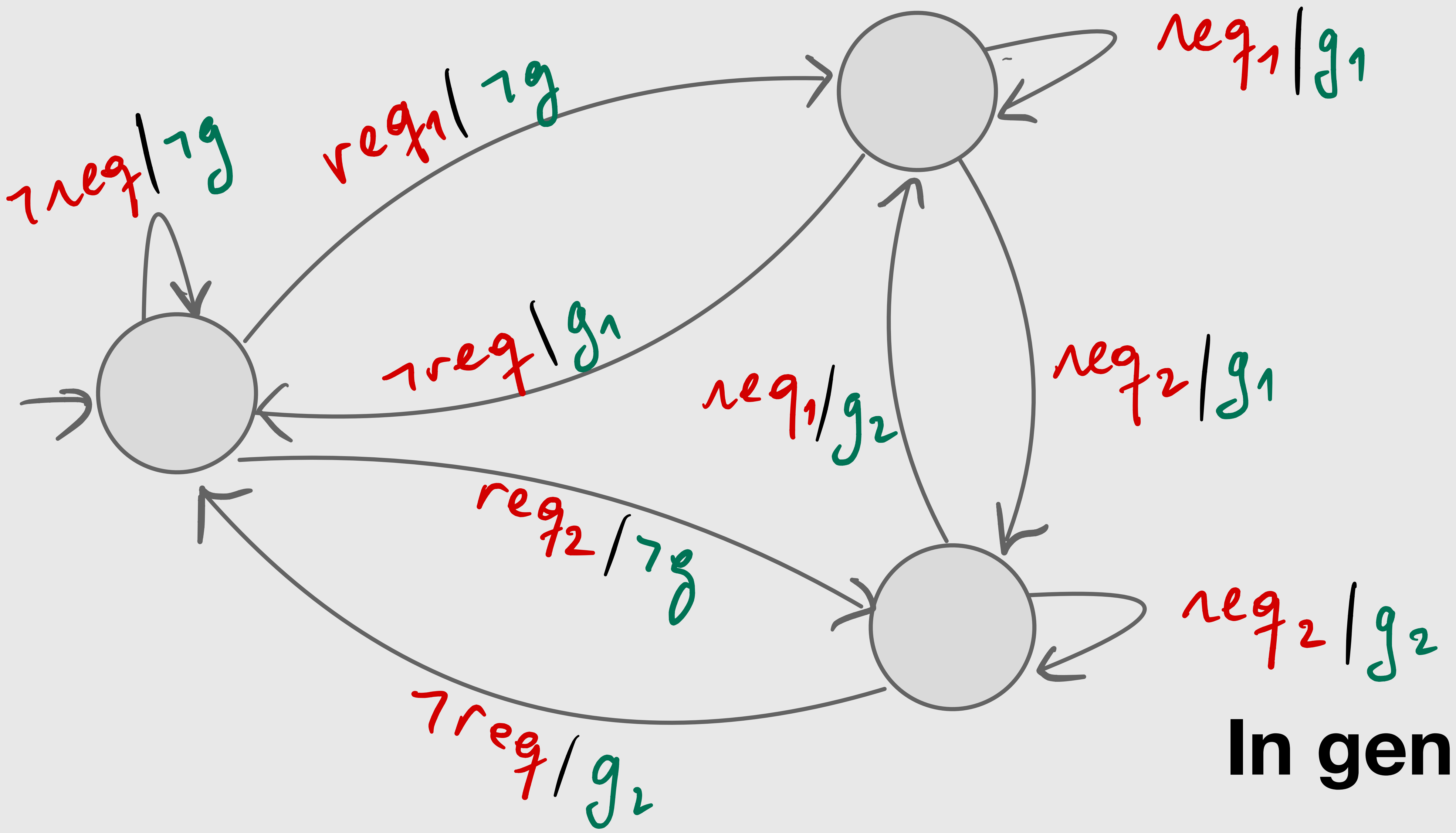
Example 4: request/grant with delay

- 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 → automata → *deterministic* automata → games
- **Regular spec:** LTL, MSO, non-det Büchi automata, universal coBüchi automata, det parity automata, good-for-games automata, …

2exptime-c

exptime-c

Important results in reactive synthesis

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

2exptime-c

exptime-c

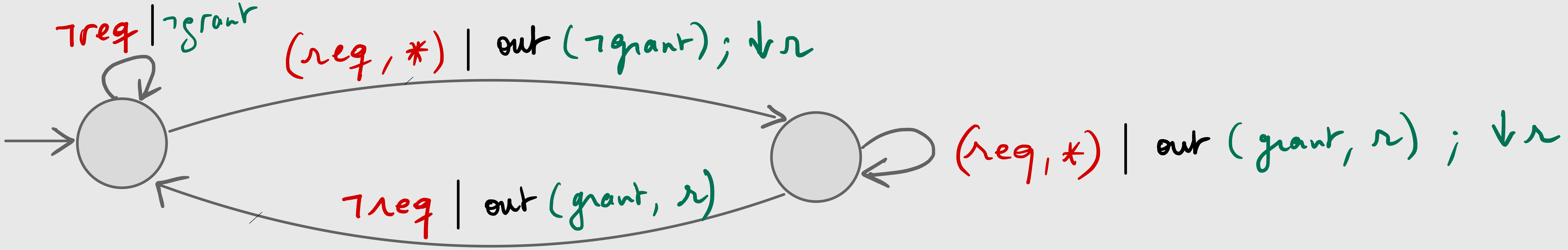
- **Game theory** on graphs
- **Tools:** Strix, LTLSynth, AcaciaBonzai, …
- Yearly synthesis **competition** since 2014

Delayed request/grant example revisited

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

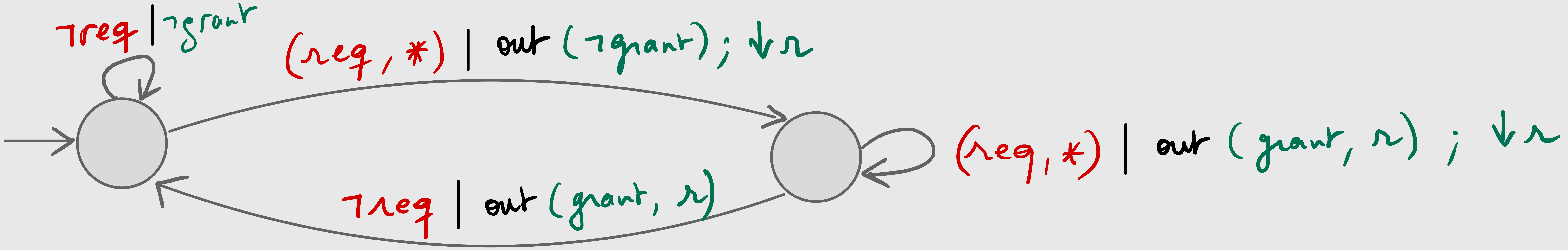
Delayed request/grant example revisited

- **Spec:** *"Any request must be granted after at least* **d** *input steps"*
- $\cancel{C \subseteq \mathbb{N}}$ $C = \mathbb{N}$
- $I = \{(req, i) \mid i \in C\} \cup \{\neg req\}$
- $O = \{(grt, i) \mid i \in C\} \cup \{\neg grt\}$
- 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"*
- $\cancel{C \subseteq \mathbb{N}}$ $C = \mathbb{N}$
- $I = \{(req, i) \mid i \in C\} \cup \{\neg req\}$
- $O = \{(grt, i) \mid i \in C\} \cup \{\neg grt\}$
- 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 $S \subseteq \mathscr{D}^{\omega}$ where $\mathscr{D}$ 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 $(\mathbb{N}, \leq, 0)$

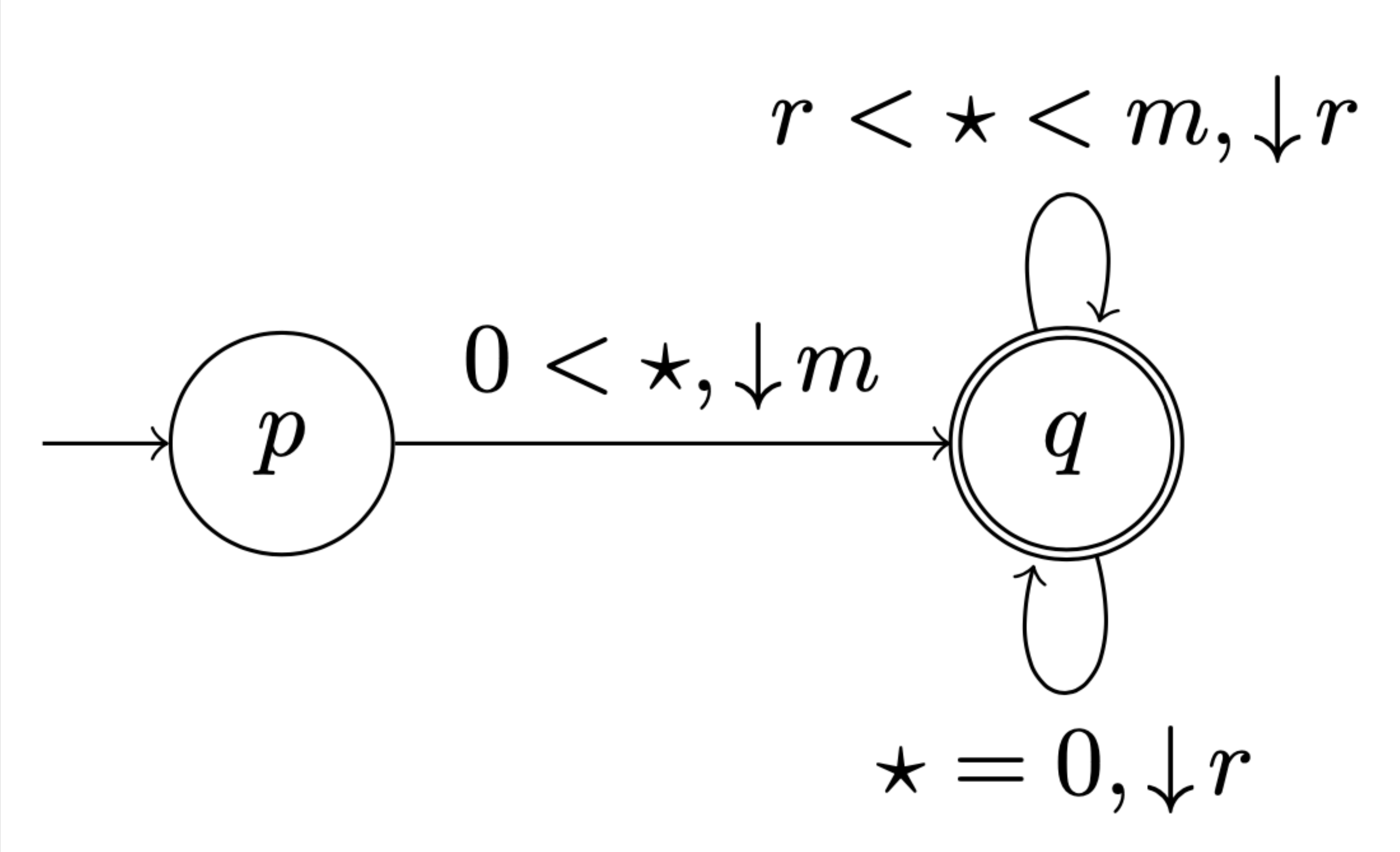
$$A = (Q, q_0, R, \delta, \alpha)$$

states

registers

transitions

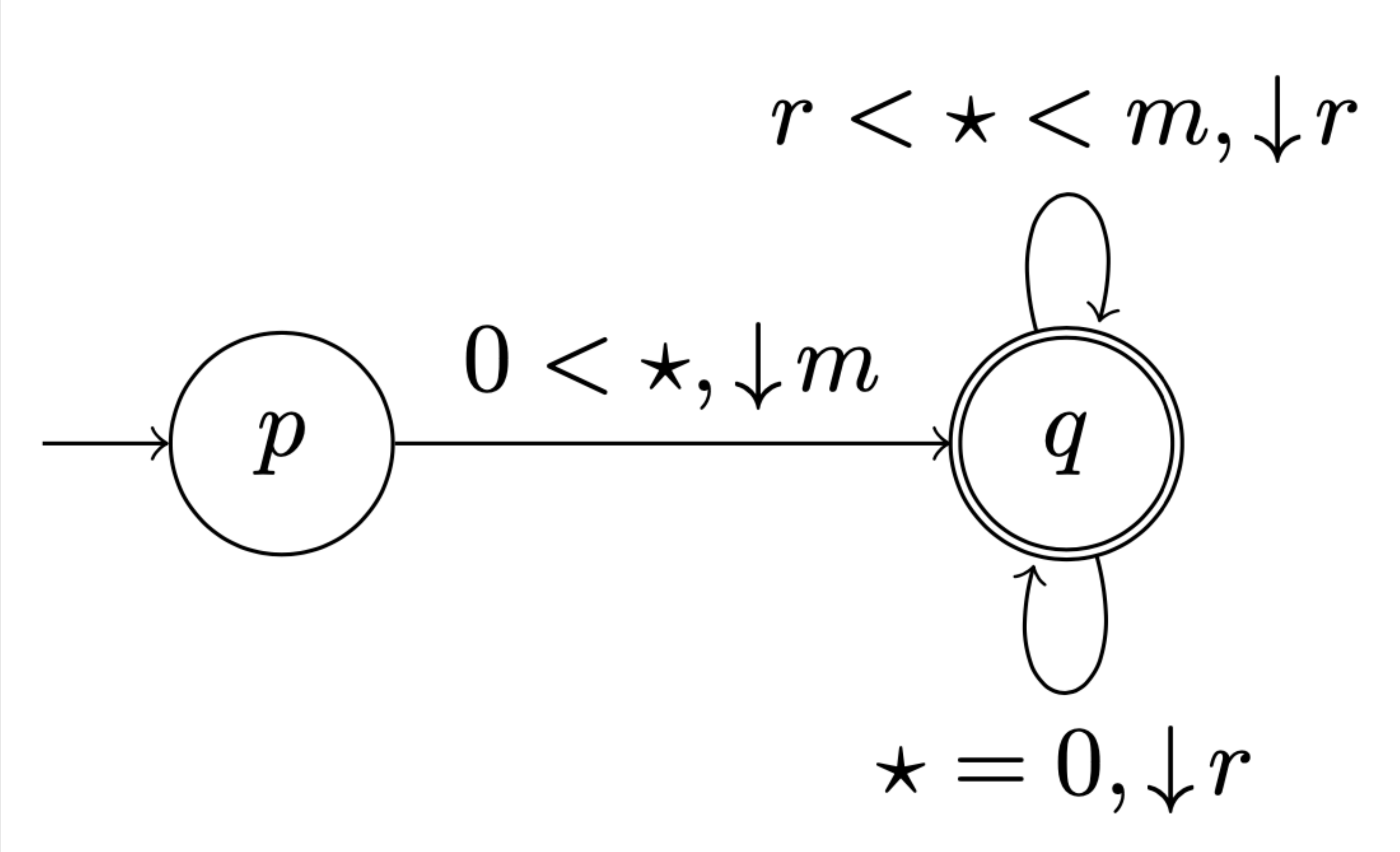
parity acceptance



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

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

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



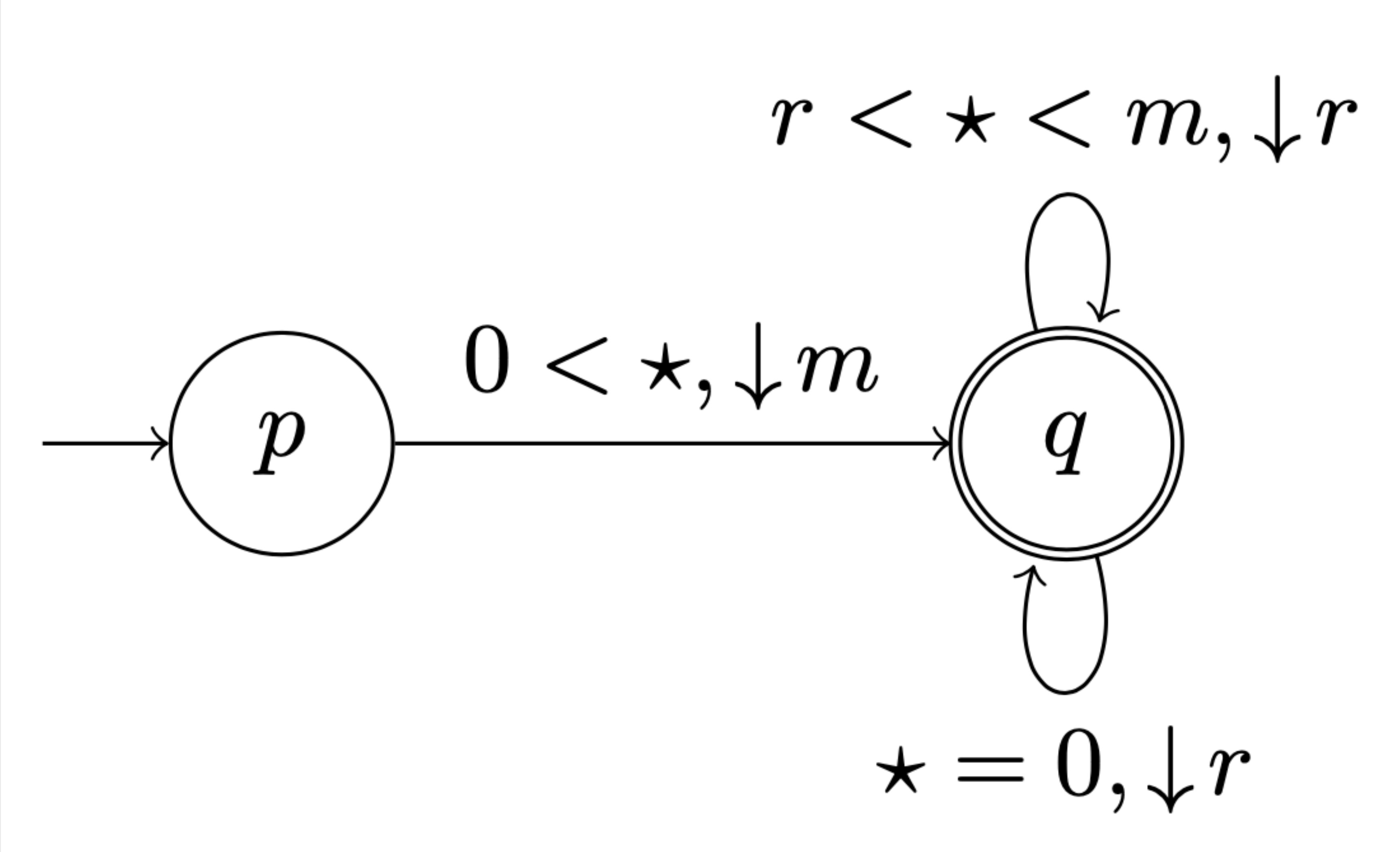
input data

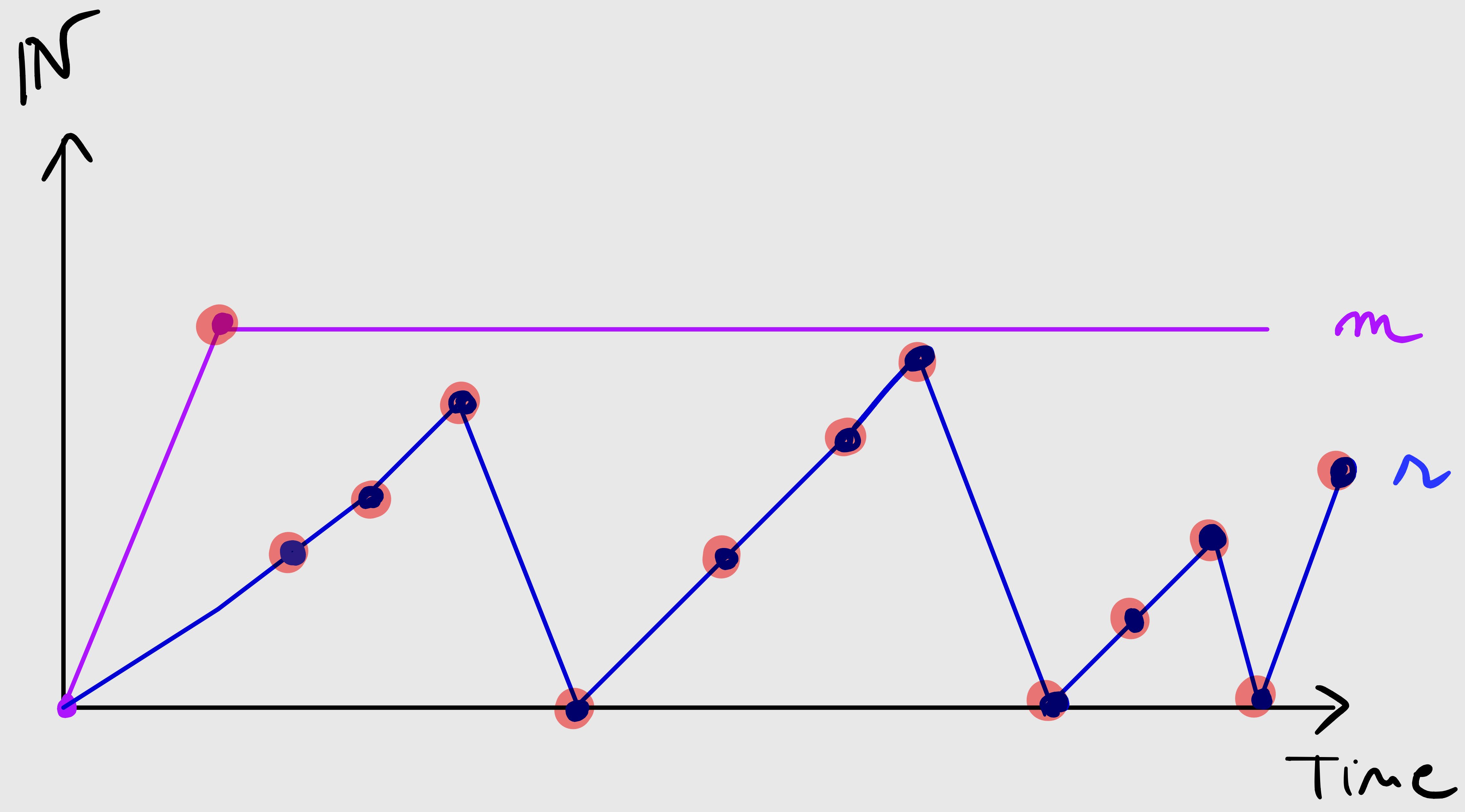
Run: $(p, 0, 0) \xrightarrow{5} (q, 5, 0) \xrightarrow{3} (q, 5, 3) \xrightarrow{4} (q, 5, 4) \xrightarrow{0} (q, 5, 0)$

state m r

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

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





input data

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

state m r

Results

Synthesis problem

Input: a universal register automaton A over $\mathscr{D}$

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 $\mathscr{D}$

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 $\mathscr{D}$ 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

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

$\Longrightarrow$ see register automata as finite automata over <u>action words</u>

Abstraction

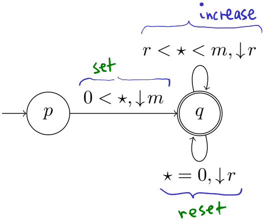
$$\delta \subseteq Q \times \underbrace{\text{Tests} \times \text{Assignments}}_{\text{finite alphabet of } \underline{\text{actions}}} \times Q$$

$\Longrightarrow$ see register automata as finite automata over <u>action words</u>

register automaton

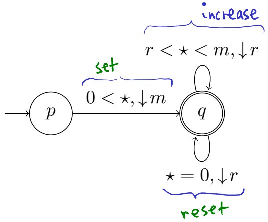
<u>NOTATION</u>: $L_{syntx}(A) =_{def}$ set of action words accepted by A

Abstraction



set $(\text{increase})^{\omega}$

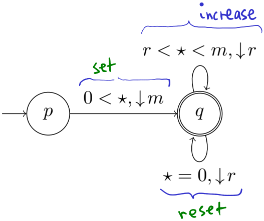
Abstraction



set $(\text{increase})^{\omega}$

set incr reset incr^2 reset incr^3 reset ...

Abstraction

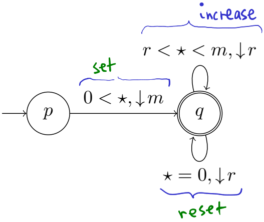


set $(\text{increase})^\omega$

set incr reset incr^2 reset incr^3 reset ...

FEASIBLE on $w \in \mathcal{D}^\omega$: "the sequence of actions can be executed on w"

ABSTRACTION



set (increase)$^\omega$

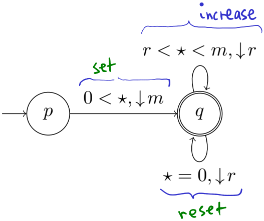
set incr reset incr^2 reset incr^3 reset ...

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

→ BUT FEASIBLE ON $(\mathbb{Q}, <)$

FEASIBLE ON $w \in \mathcal{D}^\omega$: "the sequence of actions can be executed on w"

Abstraction

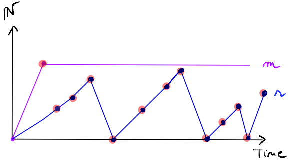


set $(\text{increase})^\omega$

set incr reset incr^2 reset incr^3 reset ...

Any feasible action word on $(\mathbb{N}, <)$ has the form

set incr^{n_1} reset incr^{n_2} ... where $\exists B$: every $n_i < B$



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

- Feasibility = local consistency

$$* > 0 \;;\; \downarrow r \;;\; * = r \;;\; \downarrow s \;;\; r < * < s \;\ldots$$

TYPES: $\{r = s = 0\}$ $\{r > 0 = s\}$ $\{r = s > 0\}$

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

- Feasibility = local consistency

$* > 0 \;;\; \downarrow r \;;\; * = r \;;\; \downarrow s \;;\; r < * < s \;\ldots$

TYPES: $\{r = s = 0\}$ $\qquad$ $\{r > 0 = s\}$ $\qquad$ $\{r = s > 0\}$

INCONSISTENT !

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

- Feasibility = local consistency

$$* > 0 \;;\; \downarrow r \;;\; * = r \;;\; \downarrow s \;;\; r < * < s \;\; \ldots$$

TYPES: $\{r = s = 0\}$ $\{r > 0 = s\}$ $\{r = s > 0\}$

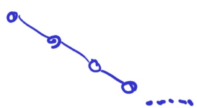
INCONSISTENT !

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

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

An action word is feasible iff it has:

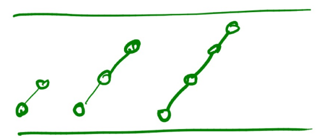
no inf decreasing chains



$$(* < r \; ; \; \downarrow n)^{\omega}$$

$$(* < n \; ; \; \downarrow s \; ; \; x < s \; ; \; \downarrow n)^{\omega}$$

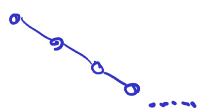
no unbounded chains of the form



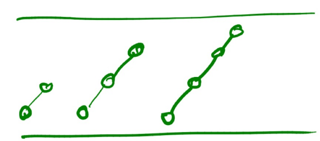
FEASIBILITY IN $(\mathbb{N}, <)$

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 à k registres

REDUCTION TO FINITE ALPHABETS

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

$$W_{S,k} \text{ is realizable by a Mealy machine}$$
$$\Leftrightarrow$$
$$S \text{ is realizable by a Mealy machine}$$ $\bar{a}$ k reg.

$$\overline{W_{S,k}} = \{\bar{a}_M \mid \exists \bar{a}_S \notin L_{syntx}(S) \cdot \exists w \in \mathcal{D}^\omega \cdot w \vDash \bar{a}_M \land w \vDash \bar{a}_S\}$$

$$= \{\bar{a}_M \mid \exists \bar{a}_S \notin L_{syntx}(S) \cdot \bar{a}_M \otimes \bar{a}_S \in \mathsf{FEAS}\}$$

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

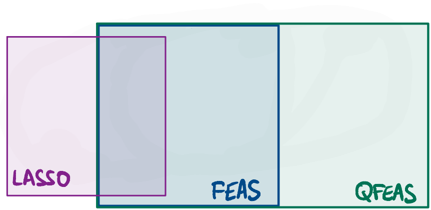
$$\begin{aligned}\overline{W_{S,k}} &= \{\bar{a}_M \mid \exists \bar{a}_S \notin L_{syntx}(S) \cdot \exists w \in \mathscr{D}^\omega \cdot w \models \bar{a}_M \wedge w \models \bar{a}_S\} \\ &= \{\bar{a}_M \mid \exists \bar{a}_S \notin L_{syntx}(S) \cdot \bar{a}_M \otimes \bar{a}_S \in \mathsf{FEAS}\}\end{aligned}$$

NOT ω-REGULAR IN GENERAL !

Generic Solution

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

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



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

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

S is defined by a URA

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

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

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

So, $W^{QF}_{S,k}$ 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

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

S is defined by a URA

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

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

Let M s.t. $L(M) \nsubseteq L(W^{QF}_{S,k})$.

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

PROOF IDEA

Lemma

S is realizable by a k-reg Mealy machine iff

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

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

S is defined by a URA

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

$$W^{QF}_{S,k} = \{\bar{a}_M \mid \forall \bar{a}_S \cdot \bar{a}_M \otimes \bar{a}_S \in \mathsf{QFEAS} \Rightarrow \bar{a}_S \in L_{syntx}(S)\}$$

Let M s.t. $L(M) \nsubseteq L(W^{QF}_{S,k})$.

$$\exists\, \bar{a}_M \otimes \bar{a}_S \in \Big(\underbrace{\mathsf{QFEAS} \cap L(M) \otimes \overline{L_{syntx}(S)}}_{\omega\text{-reg}}\Big) \cap \mathsf{LASSO}$$

PROOF IDEA

Lemma

S is realizable by a k-reg Mealy machine iff

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

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

S is defined by a URA

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

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

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

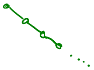
$$\exists\, \bar{a}_M \otimes \bar{a}_S \in \Big(\underbrace{\text{QFEAS} \cap L(M) \otimes \overline{L_{syntx}(S)}}_{\omega\text{-reg}}\Big) \cap \text{LASSO}$$

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

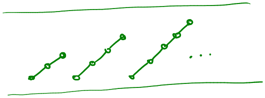
(because QFEAS ∩ LASSO = FEAS ∩ LASSO). So, $L(M) \not\subseteq W_{S,k}$ ∎

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

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

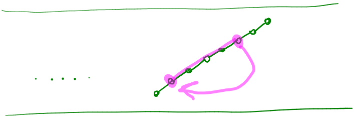


inf decr chains



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 $\mathcal{D}$ <u>reduces</u> to $\mathcal{D}'$, and $\mathcal{D}'$ is regapprox, then D is regapprox.

Two definitions of reductions:
- via transducer relations,
- via first-order formulas.

action word in D ⇝ action words in D'
feas ⟹ ∃ feas

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

prefix relation

Conclusion

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

$\longrightarrow$ 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 $FO_2[<_{pos}, succ_{pos}, =_{data}]$, what about $FO_2[<_{pos}, =_{data}]$?
  - implementation: refinement techniques

DETERMINISTIC REGISTER AUTOMATA

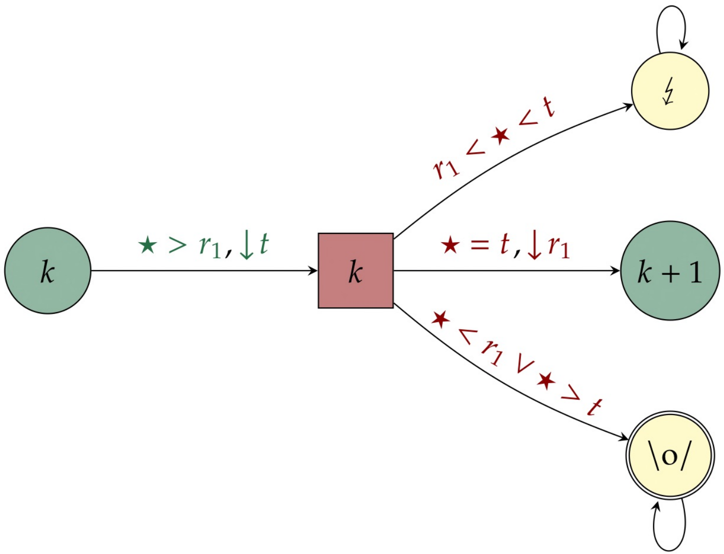
UNCONSTRAINED SYNTHESIS (∃ register Mealy machine $M \models S$?)

$(\mathbb{D}, =)$	$(\mathbb{Q}, <)$	$(\mathbb{N}, <)$
✓	✓	✗ $\longrightarrow$ Simulation of Minsky machines ✓ $\longrightarrow$ data on input only!

CHURCH PROBLEM ($\exists f : \mathbb{D}^* \longrightarrow \mathbb{D}$ such that $f \models S$?)

$(\mathbb{D}, =)$	$(\mathbb{Q}, <)$	$(\mathbb{N}, <)$
✓	✓	✗

register Mealy machine suffice



incrementing r_1 by 1

WHY NOT NRA ?

Fundamental reason: $\exists M \, \forall \tau \, \forall \text{run}$ (URA)

$\exists M \, \forall \tau \, \exists \text{run}$ (NRA)

OTHER INTUITION: $W_{S,k} = \{ \bar{a}_M \mid \forall \bar{a}_S . \bar{a}_S \otimes \bar{a}_M \in FEAS \Rightarrow \bar{a}_S \in L_{Syntx}(S) \}$

→ If Adam can enforce $\bar{a}_M \notin W_{S,k}$, then $\exists \bar{a}_S \notin L_{Syntx}(S) . \exists w \in \mathbb{D}^\omega . w \models \bar{a}_M \wedge w \models \bar{a}_S$

→ From $\bar{a}_S \notin L_{Syntx}(S)$ a rejecting run on w can be constructed

⇒ by enforcing w in the concrete game, Adam wins!

FAILS with NRA

MORE DETAILS ON THE REDUCTION

$$\overline{W_{S,k}} = \{ \bar{a}_M \mid \exists w \in \mathcal{D}^\omega \,.\, w \notin L(S) \wedge w \models \bar{a}_M \}$$

$$= \{ \bar{a}_M \mid \exists w \in \mathcal{D}^\omega \,.\, \underbrace{\exists \text{ non-accept run } \rho \text{ on } w}_{} \wedge w \models \bar{a}_M \}$$

$$= \{ \bar{a}_M \mid \exists w \in \mathcal{D}^\omega \; \boxed{\exists \bar{a}_S \notin L_{syntx}(S) \wedge w \models \bar{a}_S} \wedge w \models \bar{a}_M \}$$

$$= \{ \bar{a}_M \mid \exists \bar{a}_S \notin L_{syntx}(S) \wedge \exists w \in \mathcal{D}^\omega . w \models \bar{a}_S \wedge w \models \bar{a}_M \}$$

$$= \{ \bar{a}_M \mid \exists \bar{a}_S \notin L_{syntx}(S) \wedge \bar{a}_S \otimes \bar{a}_M \in FEAS \}$$