What's Decidable about Discrete Linear Dynamical Systems?

Joël Ouaknine

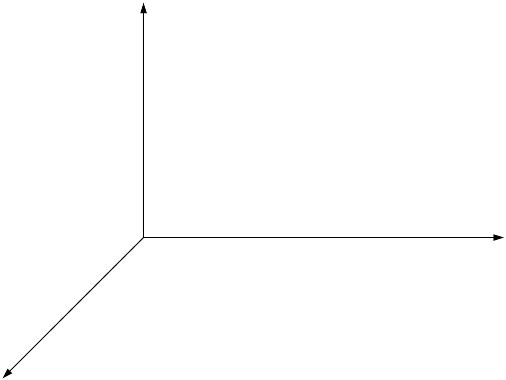
Max Planck Institute for Software Systems

Joint work with: Toghrul Karimov, Shaull Almagor, Ventsi Chonev, Edon Kelmendi, Engel Lefaucheux, Florian Luca, Joris Nieuwveld, David Purser, João Sousa Pinto, Anton Varonka, Markus Whiteland, James Worrell, . . .

Theorietag Automaten und Formale Sprachen
Kaiserslautern, Oktober 2023

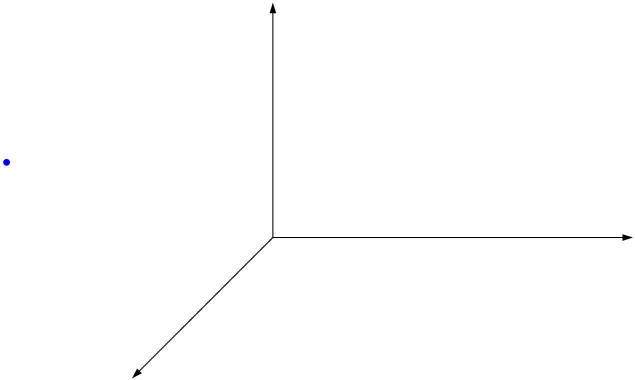
Reachability for discrete linear dynamical systems

Ambient space: $\mathbb{R}^d$ ($\mathbb{R}^3$ in this example)
Starting point: $\mathbf{x} \in \mathbb{Q}^d$
Linear transformation: $\mathbf{M} \in \mathbb{Q}^{d \times d}$



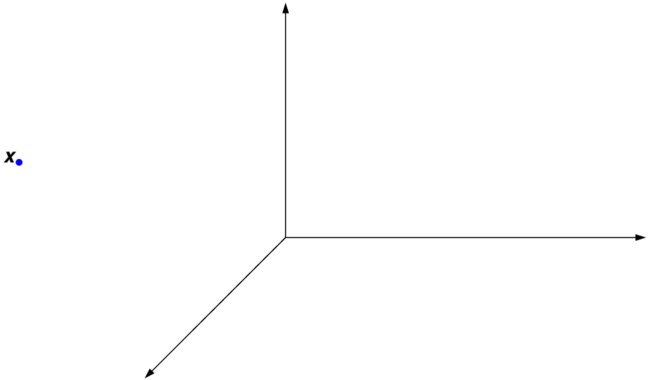
Reachability for discrete linear dynamical systems

Ambient space: $\mathbb{R}^d$ ($\mathbb{R}^3$ in this example)
Starting point: $\mathbf{x} \in \mathbb{Q}^d$
Linear transformation: $\mathbf{M} \in \mathbb{Q}^{d \times d}$



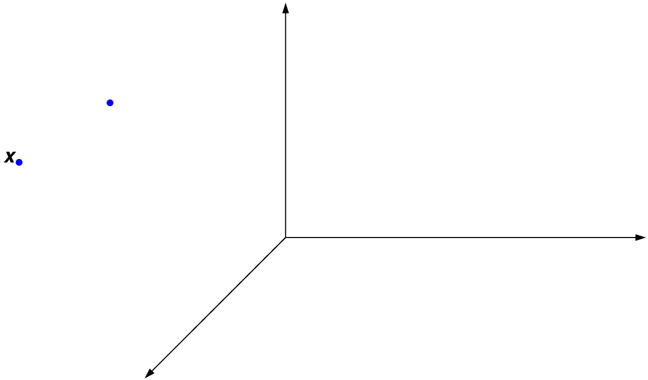
Reachability for discrete linear dynamical systems

Ambient space: $\mathbb{R}^d$ ($\mathbb{R}^3$ in this example)
Starting point: $\mathbf{x} \in \mathbb{Q}^d$
Linear transformation: $\mathbf{M} \in \mathbb{Q}^{d \times d}$



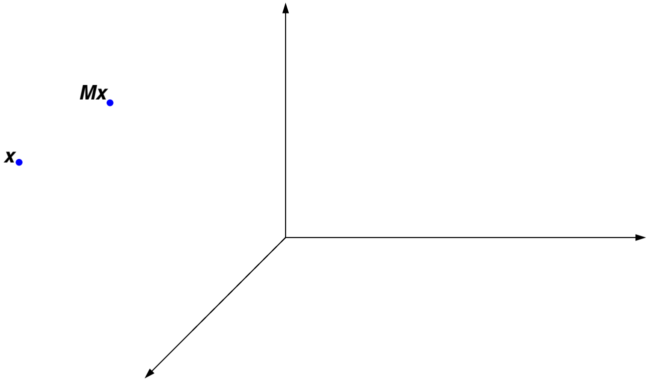
Reachability for discrete linear dynamical systems

Ambient space: $\mathbb{R}^d$ ($\mathbb{R}^3$ in this example)
Starting point: $\mathbf{x} \in \mathbb{Q}^d$
Linear transformation: $\mathbf{M} \in \mathbb{Q}^{d \times d}$



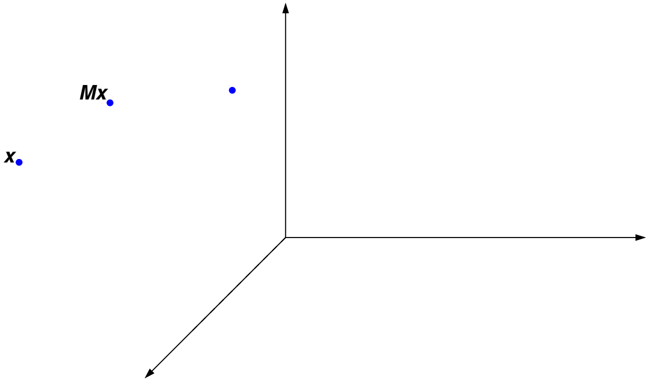
Reachability for discrete linear dynamical systems

Ambient space: $\mathbb{R}^d$ ($\mathbb{R}^3$ in this example)
Starting point: $\mathbf{x} \in \mathbb{Q}^d$
Linear transformation: $\mathbf{M} \in \mathbb{Q}^{d \times d}$



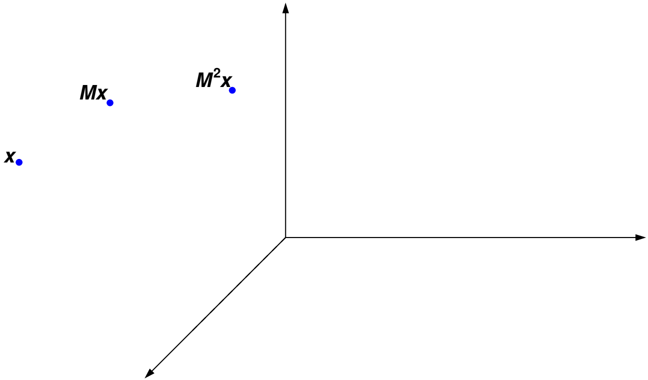
Reachability for discrete linear dynamical systems

Ambient space: $\mathbb{R}^d$ ($\mathbb{R}^3$ in this example)
Starting point: $\mathbf{x} \in \mathbb{Q}^d$
Linear transformation: $\mathbf{M} \in \mathbb{Q}^{d \times d}$



Reachability for discrete linear dynamical systems

Ambient space: $\mathbb{R}^d$ ($\mathbb{R}^3$ in this example)
Starting point: $\mathbf{x} \in \mathbb{Q}^d$
Linear transformation: $\mathbf{M} \in \mathbb{Q}^{d \times d}$

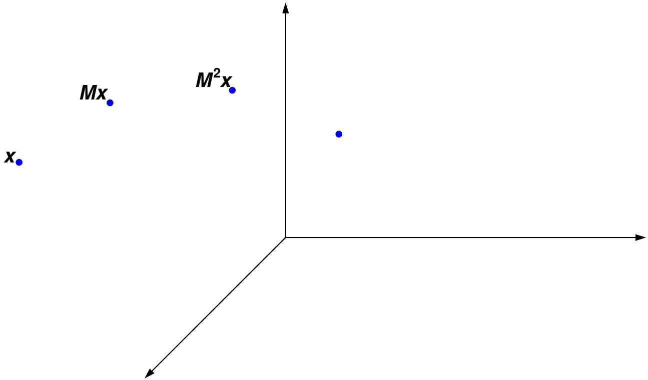


Reachability for discrete linear dynamical systems

Ambient space: $\mathbb{R}^d$ ($\mathbb{R}^3$ in this example)

Starting point: $\mathbf{x} \in \mathbb{Q}^d$

Linear transformation: $\mathbf{M} \in \mathbb{Q}^{d \times d}$

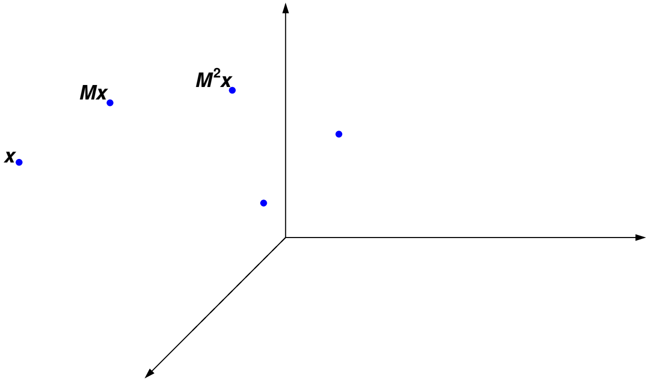


Reachability for discrete linear dynamical systems

Ambient space: $\mathbb{R}^d$ ($\mathbb{R}^3$ in this example)

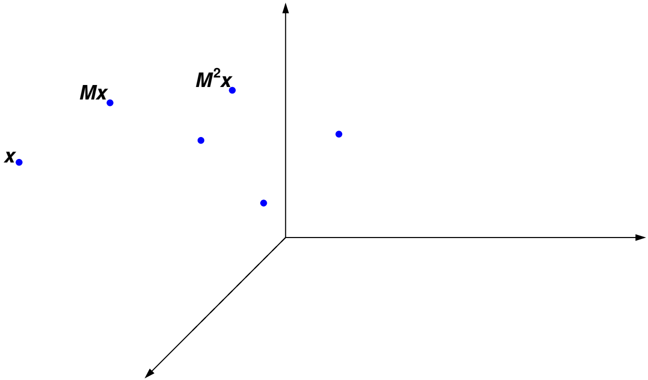
Starting point: $\mathbf{x} \in \mathbb{Q}^d$

Linear transformation: $\mathbf{M} \in \mathbb{Q}^{d \times d}$



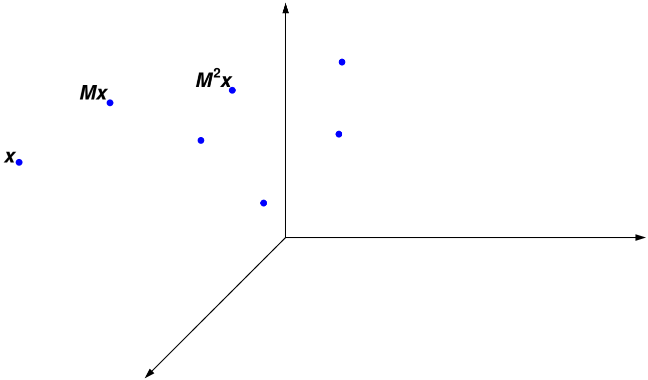
Reachability for discrete linear dynamical systems

Ambient space: $\mathbb{R}^d$ ($\mathbb{R}^3$ in this example)
Starting point: $\mathbf{x} \in \mathbb{Q}^d$
Linear transformation: $\mathbf{M} \in \mathbb{Q}^{d \times d}$



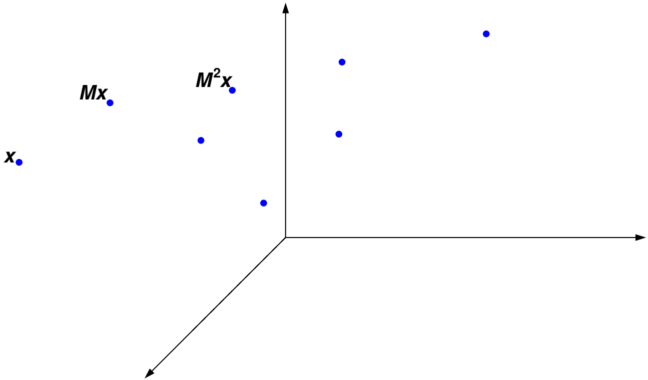
Reachability for discrete linear dynamical systems

Ambient space: $\mathbb{R}^d$ ($\mathbb{R}^3$ in this example)
Starting point: $\mathbf{x} \in \mathbb{Q}^d$
Linear transformation: $\mathbf{M} \in \mathbb{Q}^{d \times d}$



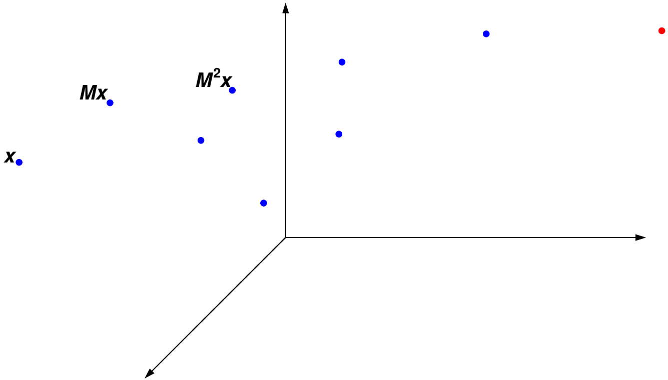
Reachability for discrete linear dynamical systems

Ambient space: $\mathbb{R}^d$ ($\mathbb{R}^3$ in this example)
Starting point: $\mathbf{x} \in \mathbb{Q}^d$
Linear transformation: $\mathbf{M} \in \mathbb{Q}^{d \times d}$



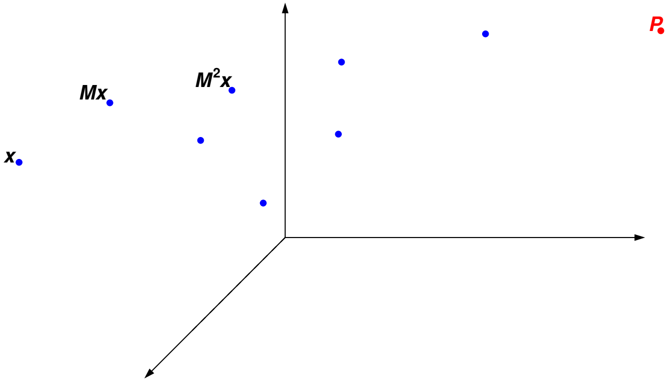
Reachability for discrete linear dynamical systems

Ambient space: $\mathbb{R}^d$ ($\mathbb{R}^3$ in this example)
Starting point: $\mathbf{x} \in \mathbb{Q}^d$
Linear transformation: $\mathbf{M} \in \mathbb{Q}^{d \times d}$



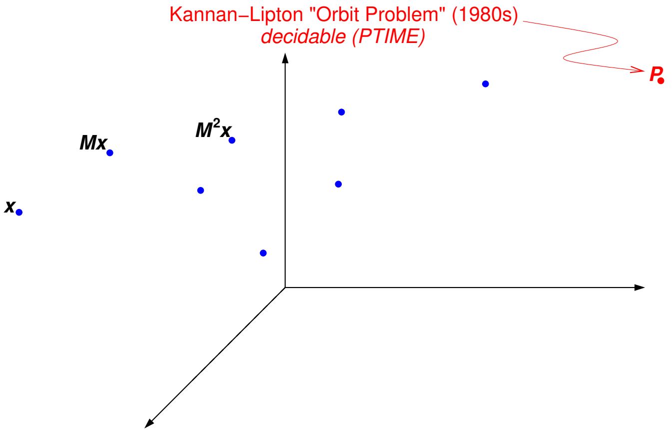
Reachability for discrete linear dynamical systems

Ambient space: $\mathbb{R}^d$ ($\mathbb{R}^3$ in this example)
Starting point: $\mathbf{x} \in \mathbb{Q}^d$
Linear transformation: $\mathbf{M} \in \mathbb{Q}^{d \times d}$



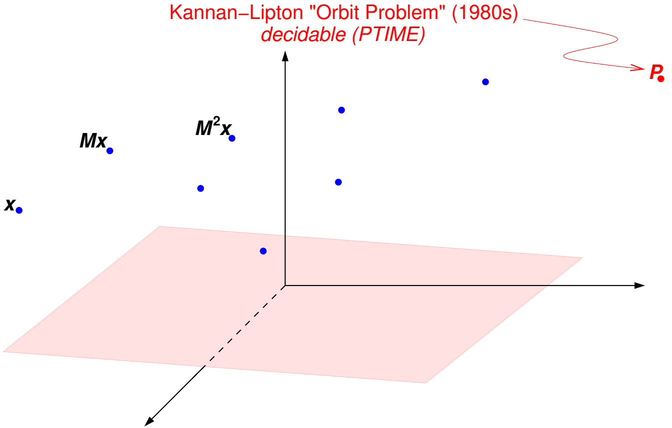
Reachability for discrete linear dynamical systems

Ambient space: $\mathbb{R}^d$ ($\mathbb{R}^3$ in this example)
Starting point: $\mathbf{x} \in \mathbb{Q}^d$
Linear transformation: $\mathbf{M} \in \mathbb{Q}^{d \times d}$



Reachability for discrete linear dynamical systems

Ambient space: $\mathbb{R}^d$ ($\mathbb{R}^3$ in this example)
Starting point: $\mathbf{x} \in \mathbb{Q}^d$
Linear transformation: $\mathbf{M} \in \mathbb{Q}^{d \times d}$

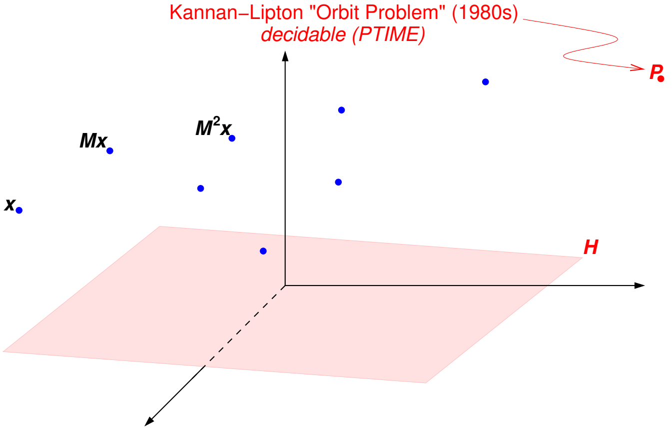


Reachability for discrete linear dynamical systems

Ambient space: $\mathbb{R}^d$ ($\mathbb{R}^3$ in this example)

Starting point: $\mathbf{x} \in \mathbb{Q}^d$

Linear transformation: $\mathbf{M} \in \mathbb{Q}^{d \times d}$

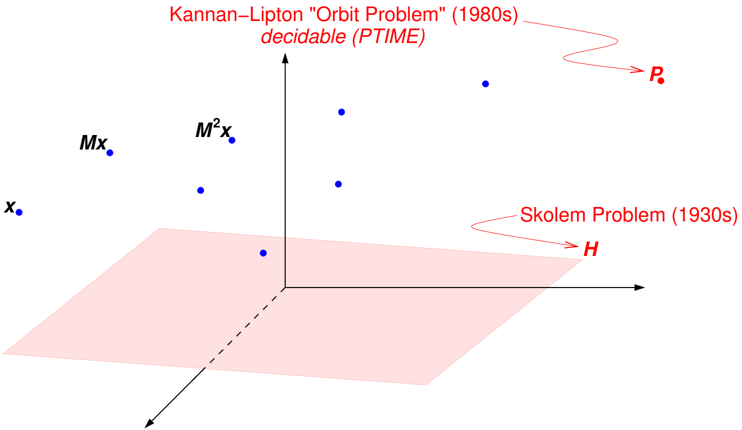


Reachability for discrete linear dynamical systems

Ambient space: $\mathbb{R}^d$ ($\mathbb{R}^3$ in this example)

Starting point: $\mathbf{x} \in \mathbb{Q}^d$

Linear transformation: $\mathbf{M} \in \mathbb{Q}^{d \times d}$

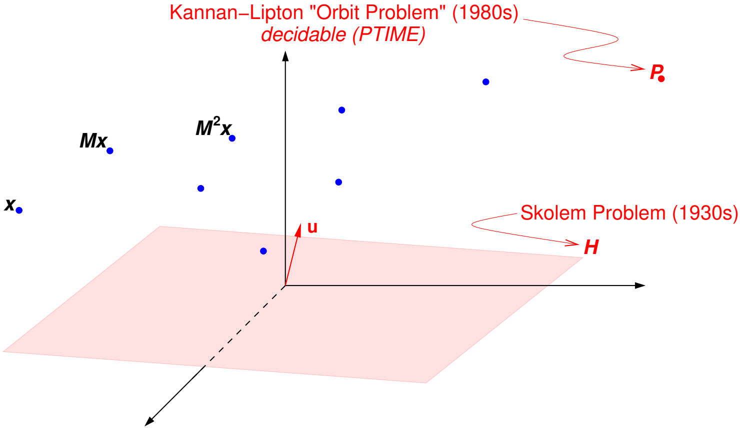


Reachability for discrete linear dynamical systems

Ambient space: $\mathbb{R}^d$ ($\mathbb{R}^3$ in this example)

Starting point: $\mathbf{x} \in \mathbb{Q}^d$

Linear transformation: $\mathbf{M} \in \mathbb{Q}^{d \times d}$

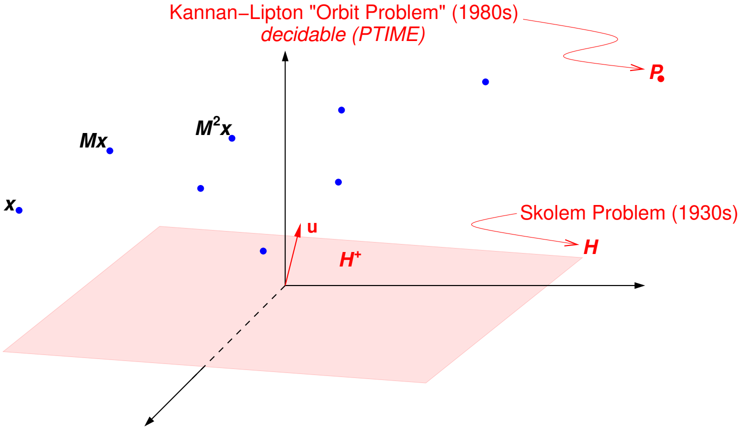


Reachability for discrete linear dynamical systems

Ambient space: $\mathbb{R}^d$ ($\mathbb{R}^3$ in this example)

Starting point: $\mathbf{x} \in \mathbb{Q}^d$

Linear transformation: $\mathbf{M} \in \mathbb{Q}^{d \times d}$

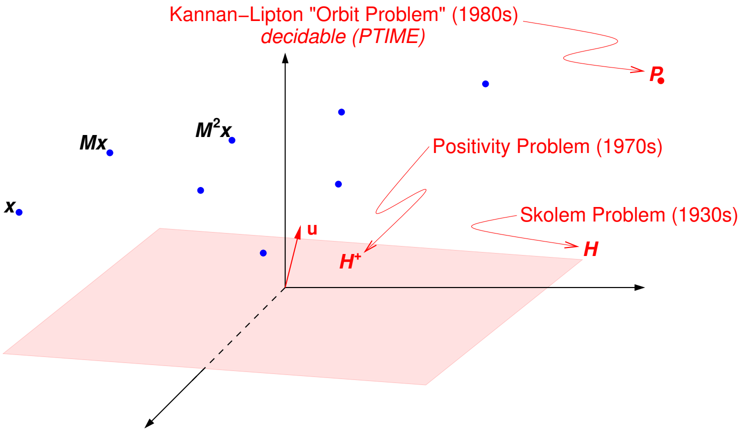


Reachability for discrete linear dynamical systems

Ambient space: $\mathbb{R}^d$ ($\mathbb{R}^3$ in this example)

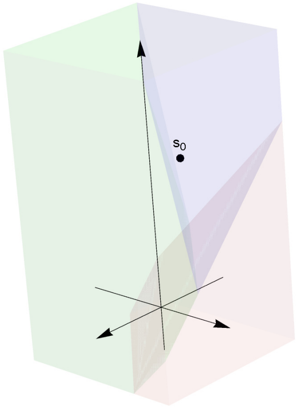
Starting point: $\mathbf{x} \in \mathbb{Q}^d$

Linear transformation: $\mathbf{M} \in \mathbb{Q}^{d \times d}$



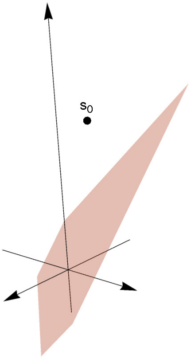
Semialgebraic partitions

Partition $\mathbb{R}^d$ into



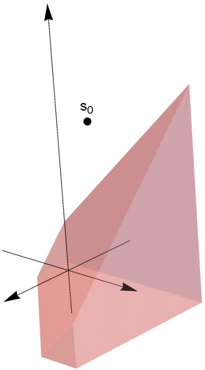
Semialgebraic partitions

Partition $\mathbb{R}^d$ into S_1



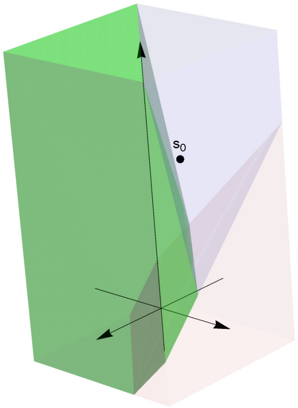
Semialgebraic partitions

Partition $\mathbb{R}^d$ into S_1, S_2



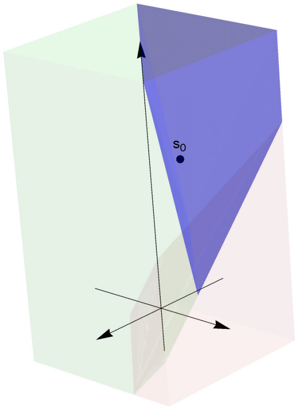
Semialgebraic partitions

Partition $\mathbb{R}^d$ into S_1, S_2, S_3



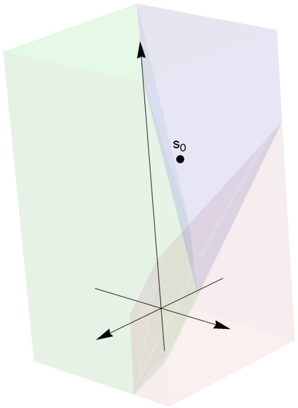
Semialgebraic partitions

Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



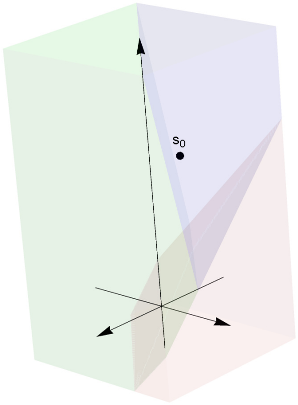
Semialgebraic partitions

Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4

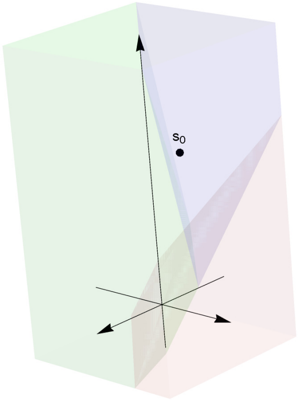


Semialgebraic partitions

Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4

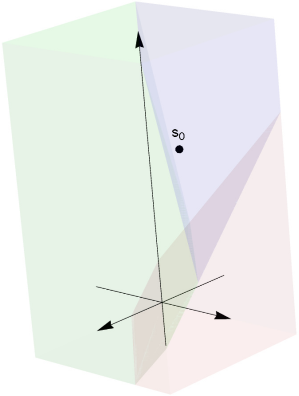


Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



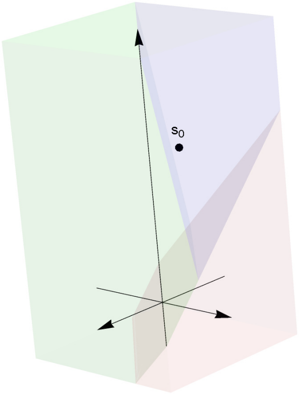
Semialgebraic partitions

Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



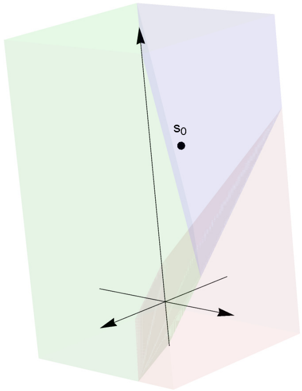
Semialgebraic partitions

Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



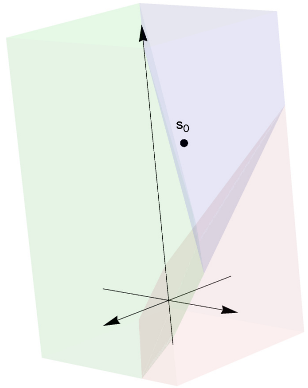
Semialgebraic partitions

Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



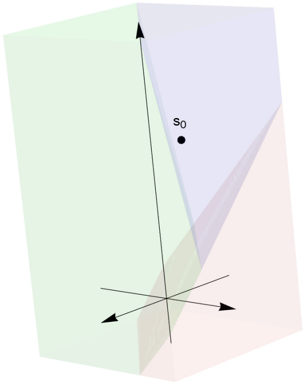
Semialgebraic partitions

Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



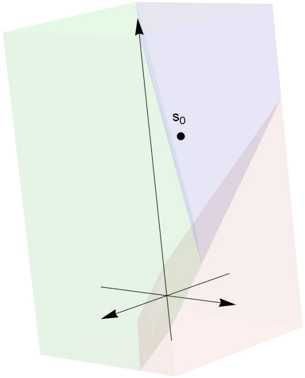
Semialgebraic partitions

Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



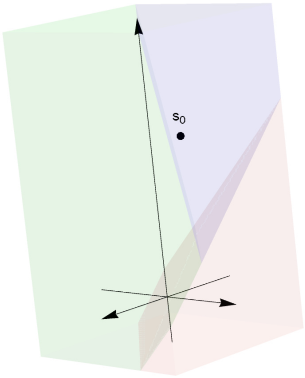
Semialgebraic partitions

Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



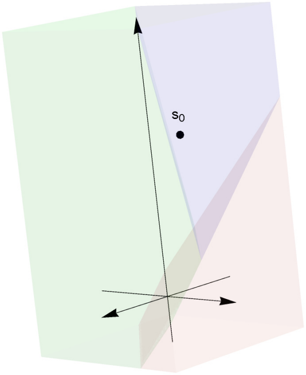
Semialgebraic partitions

Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



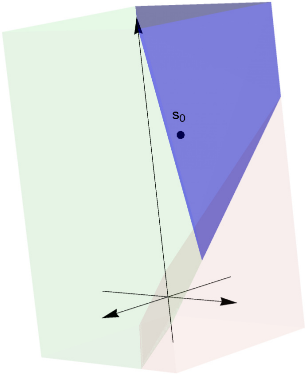
Semialgebraic partitions

Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



Semialgebraic partitions

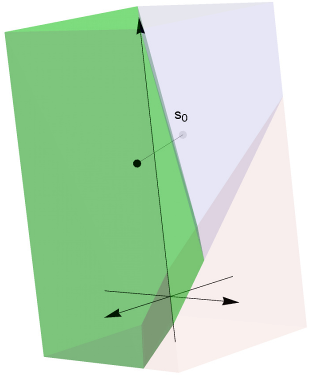
Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



$w=$ 

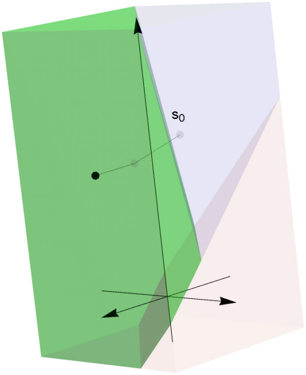
Semialgebraic partitions

Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



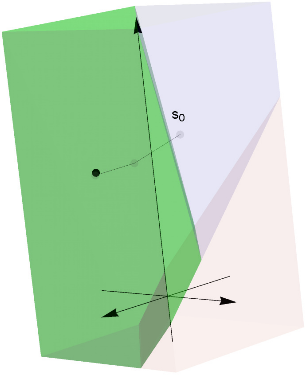
Semialgebraic partitions

Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



Semialgebraic partitions

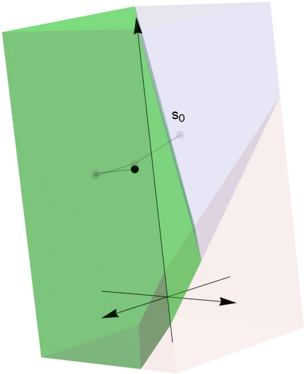
Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



$w=$ 

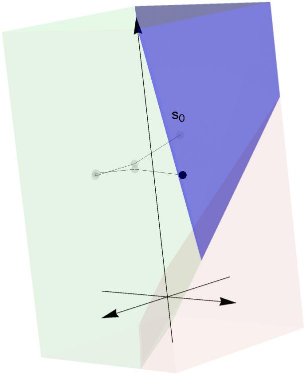
Semialgebraic partitions

Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



Semialgebraic partitions

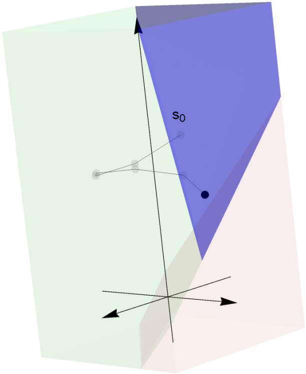
Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



$w=$ 

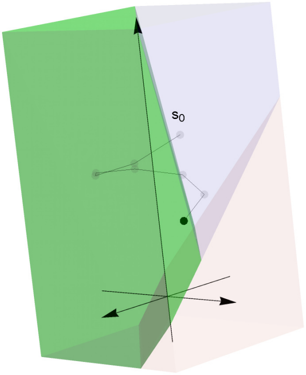
Semialgebraic partitions

Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



Semialgebraic partitions

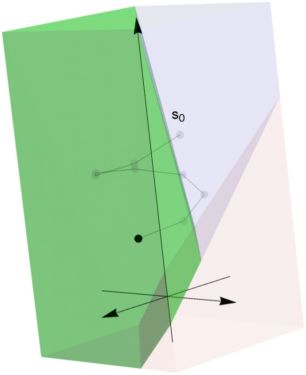
Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



$w=$ 

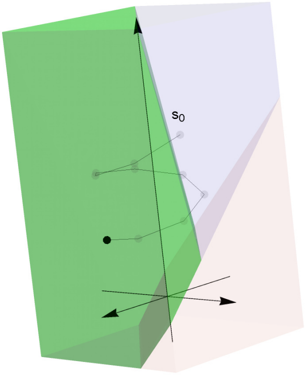
Semialgebraic partitions

Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



Semialgebraic partitions

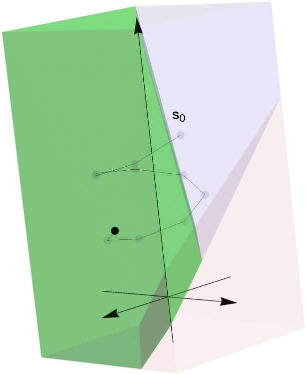
Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



$w=$ 

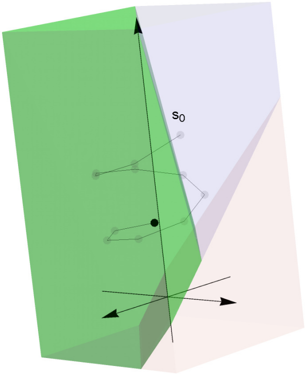
Semialgebraic partitions

Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



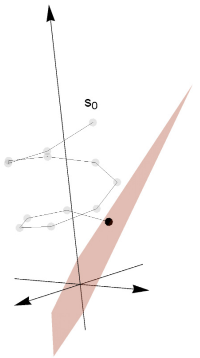
Semialgebraic partitions

Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



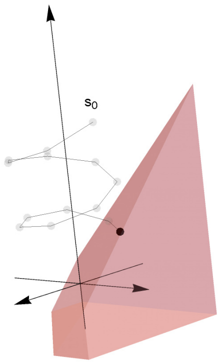
Semialgebraic partitions

Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



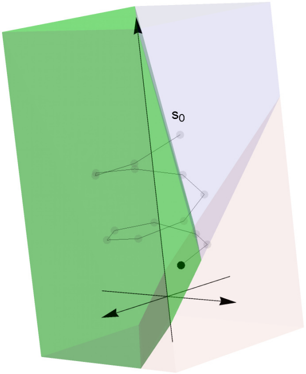
Semialgebraic partitions

Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



Semialgebraic partitions

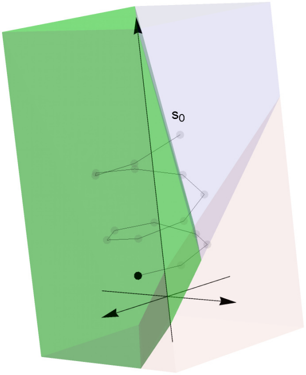
Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



$w =$ 

Semialgebraic partitions

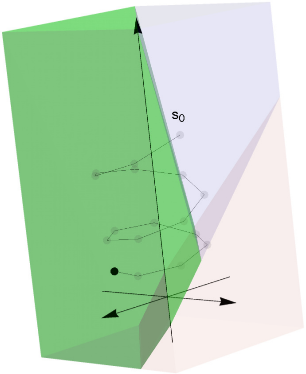
Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



$w=$ 

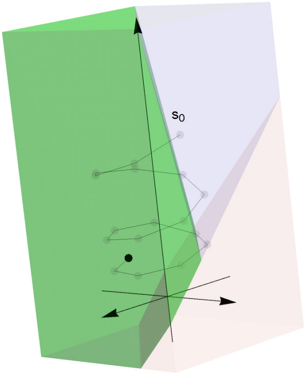
Semialgebraic partitions

Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



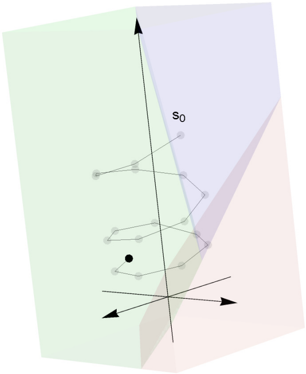
Semialgebraic partitions

Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



Semialgebraic partitions

Partition $\mathbb{R}^d$ into S_1, S_2, S_3, S_4



Model checking discrete linear dynamical systems

$w=$ 

generated by (M, s)

Model checking discrete linear dynamical systems

generated by (M, s)

> The Model-Checking Problem:
> Given $\mathcal{W}$ and a specification φ, decide if $\mathcal{W} \models \varphi$

Model checking discrete linear dynamical systems

$\mathcal{W}=$ 

generated by (M, s)

The Model-Checking Problem:
Given $\mathcal{W}$ and a specification φ, decide if $\mathcal{W} \models \varphi$

- Deciding ω-Regular Properties on Linear Recurrence Sequences
Almagor, Karimov, Kelmendi, O., Worrell, in *POPL* 2021

- What's Decidable about Linear Loops?
Karimov, Lefaucheux, O., Purser, Varonka, Whiteland, Worrell, in *POPL* 2022

- The Power of Positivity
Karimov, Kelmendi, Nieuwveld, O., Worrell, in *LICS* 2023

- What's Decidable about Discrete Linear Dynamical Systems?
Karimov, Kelmendi, O., Worrell, in *Henzinger Festschrift* 2023

Model checking discrete linear dynamical systems

generated by (M, s)

The Model-Checking Problem:
Given $\mathcal{W}$ and a specification φ, decide if $\mathcal{W} \models \varphi$

We consider:

Model checking discrete linear dynamical systems

$\mathcal{W}$= 

generated by (M, s)

> The Model-Checking Problem:
> Given $\mathcal{W}$ and a specification φ, decide if $\mathcal{W} \models \varphi$

We consider:

- Two different kinds of linear dynamical systems (M, s):

Model checking discrete linear dynamical systems

$\mathcal{W}$= 

generated by (M, s)

> The Model-Checking Problem:
> Given $\mathcal{W}$ and a specification φ, decide if $\mathcal{W} \models \varphi$

We consider:

- Two different kinds of linear dynamical systems (M, s):
  - arbitrary linear dynamical systems

Model checking discrete linear dynamical systems

generated by (M, s)

> The Model-Checking Problem:
> Given $\mathcal{W}$ and a specification φ, decide if $\mathcal{W} \vDash \varphi$

We consider:

- Two different kinds of linear dynamical systems (M, s):
  - arbitrary linear dynamical systems
  - *diagonalisable* linear dynamical systems

Model checking discrete linear dynamical systems

$\mathcal{W}$= 

generated by (M, s)

> The Model-Checking Problem:
> Given $\mathcal{W}$ and a specification φ, decide if $\mathcal{W} \models \varphi$

We consider:

- Two different kinds of linear dynamical systems (M, s):
  - arbitrary linear dynamical systems
  - *diagonalisable* linear dynamical systems
- Two different kinds of specification formalisms:

Model checking discrete linear dynamical systems

$\mathcal{W}$= 

generated by (M, s)

> The Model-Checking Problem:
> Given $\mathcal{W}$ and a specification φ, decide if $\mathcal{W} \models \varphi$

We consider:

- Two different kinds of linear dynamical systems (M, s):
  - arbitrary linear dynamical systems
  - *diagonalisable* linear dynamical systems
- Two different kinds of specification formalisms:
  - arbitrary MSO (fancy version of LTL)

Model checking discrete linear dynamical systems

$\mathcal{W}$=

generated by (M, s)

> The Model-Checking Problem:
> Given $\mathcal{W}$ and a specification φ, decide if $\mathcal{W} \models \varphi$

We consider:

- Two different kinds of linear dynamical systems (M, s):
    - arbitrary linear dynamical systems
    - *diagonalisable* linear dynamical systems
- Two different kinds of specification formalisms:
    - arbitrary MSO (fancy version of LTL)
    - *prefix-independent* MSO (denoted piMSO)

Model checking discrete linear dynamical systems

$\mathcal{W}=$ 

generated by (M, s)

The Model-Checking Problem:
Given $\mathcal{W}$ and a specification φ, decide if $\mathcal{W} \models \varphi$

We consider:

- Two different kinds of linear dynamical systems (M, s):
  - arbitrary linear dynamical systems
  - *diagonalisable* linear dynamical systems
- Two different kinds of specification formalisms:
  - arbitrary MSO (fancy version of LTL)
  - *prefix-independent* MSO (denoted piMSO)
- Several different classes of semialgebraic predicates

Model checking discrete linear dynamical systems

$\mathcal{W}=$ 

generated by (M, s)

> The Model-Checking Problem:
> Given $\mathcal{W}$ and a specification φ, decide if $\mathcal{W} \models \varphi$

We consider:

- Two different kinds of linear dynamical systems (M, s):
  - arbitrary linear dynamical systems
  - *diagonalisable* linear dynamical systems
- Two different kinds of specification formalisms:
  - arbitrary MSO (fancy version of LTL)
  - *prefix-independent* MSO (denoted piMSO)
- Several different classes of semialgebraic predicates
- The use (or not) of *Skolem and/or Positivity oracles*

Quiz time! Prefix-independence

Which of the following specs are prefix-independent?

Quiz time! Prefix-independence

Which of the following specs are prefix-independent?

1. SKOLEM: **F** *H*

Quiz time! Prefix-independence

Which of the following specs are prefix-independent?

1. SKOLEM: $\mathbf{F}\ H$
2. POSITIVITY: $\mathbf{G}\ H^{+}$

Quiz time! Prefix-independence

Which of the following specs are prefix-independent?

1. SKOLEM: $\mathbf{F}\ H$
2. POSITIVITY: $\mathbf{G}\ H^+$
3. ULTIMATE POSITIVITY: $\mathbf{F}\ \mathbf{G}\ H^+$

Quiz time! Prefix-independence

Which of the following specs are prefix-independent?

1. SKOLEM: $\mathbf{F}\ H$
2. POSITIVITY: $\mathbf{G}\ H^{+}$
3. ULTIMATE POSITIVITY: $\mathbf{F}\ \mathbf{G}\ H^{+}$
4. INFINITE RECURRENCE: $\mathbf{G}\ \mathbf{F}\ P$

Quiz time! Prefix-independence

Which of the following specs are prefix-independent?

1. SKOLEM: $\mathbf{F}\ H$
2. POSITIVITY: $\mathbf{G}\ H^+$
3. ULTIMATE POSITIVITY: $\mathbf{F}\ \mathbf{G}\ H^+$
4. INFINITE RECURRENCE: $\mathbf{G}\ \mathbf{F}\ P$

Only (3) and (4)

MSO — Monadic Second-Order logic (of order)

MSO — Monadic Second-Order logic (of order)

- The domain is the set of natural numbers $\mathbb{N}$

MSO — Monadic Second-Order logic (of order)

- The domain is the set of natural numbers $\mathbb{N}$
- You are given a finite collection of monadic predicates $(P, Q, R, \ldots)$

MSO — Monadic Second-Order logic (of order)

- The domain is the set of natural numbers $\mathbb{N}$
- You are given a finite collection of monadic predicates $(P, Q, R, \ldots)$
- You can use any integer, and expressions such as "$+2$"

MSO — Monadic Second-Order logic (of order)

- The domain is the set of natural numbers $\mathbb{N}$
- You are given a finite collection of monadic predicates $(P, Q, R, \ldots)$
- You can use any integer, and expressions such as "$+2$"
- You are allowed to use "$=$" and "$<$" between integers

MSO — Monadic Second-Order logic (of order)

- The domain is the set of natural numbers $\mathbb{N}$
- You are given a finite collection of monadic predicates $(P, Q, R, \ldots)$
- You can use any integer, and expressions such as "$+2$"
- You are allowed to use "$=$" and "$<$" between integers
- You can use both first-order variables $(x, y, z, \ldots)$ and second-order monadic variables $(X, Y, Z, \ldots)$

MSO — Monadic Second-Order logic (of order)

- The domain is the set of natural numbers $\mathbb{N}$
- You are given a finite collection of monadic predicates ($P, Q, R, \ldots$)
- You can use any integer, and expressions such as "$+2$"
- You are allowed to use "$=$" and "$<$" between integers
- You can use both first-order variables ($x, y, z, \ldots$) and second-order monadic variables ($X, Y, Z, \ldots$)
  - You can then write "$X(y)$" or "$y \in X$" (same with P etc.)

MSO — Monadic Second-Order logic (of order)

- The domain is the set of natural numbers $\mathbb{N}$
- You are given a finite collection of monadic predicates $(P, Q, R, \ldots)$
- You can use any integer, and expressions such as "$+2$"
- You are allowed to use "$=$" and "$<$" between integers
- You can use both first-order variables $(x, y, z, \ldots)$ and second-order monadic variables $(X, Y, Z, \ldots)$
  - You can then write "$X(y)$" or "$y \in X$" (same with P etc.)
- You are allowed first- and second-order quantifiers, and all Boolean connectives

MSO — Monadic Second-Order logic (of order)

- The domain is the set of natural numbers $\mathbb{N}$
- You are given a finite collection of monadic predicates $(P, Q, R, \ldots)$
- You can use any integer, and expressions such as "$+2$"
- You are allowed to use "$=$" and "$<$" between integers
- You can use both first-order variables $(x, y, z, \ldots)$ and second-order monadic variables $(X, Y, Z, \ldots)$
  - You can then write "$X(y)$" or "$y \in X$" (same with P etc.)
- You are allowed first- and second-order quantifiers, and all Boolean connectives

Example: express *"P eventually contains all even integers"*

MSO — Monadic Second-Order logic (of order)

- The domain is the set of natural numbers $\mathbb{N}$
- You are given a finite collection of monadic predicates $(P, Q, R, \ldots)$
- You can use any integer, and expressions such as "$+2$"
- You are allowed to use "$=$" and "$<$" between integers
- You can use both first-order variables $(x, y, z, \ldots)$ and second-order monadic variables $(X, Y, Z, \ldots)$
  - You can then write "$X(y)$" or "$y \in X$" (same with P etc.)
- You are allowed first- and second-order quantifiers, and all Boolean connectives

Example: express *"P eventually contains all even integers"*

$$\exists X \,.\, 0 \in X \wedge 1 \notin X \wedge \forall x \,.\, (x \in X \leftrightarrow x + 2 \in X) \wedge$$
$$\exists y \,.\, \forall z > y \,.\, z \in X \rightarrow z \in P$$

MSO — Monadic Second-Order logic (of order)

- The domain is the set of natural numbers $\mathbb{N}$
- You are given a finite collection of monadic predicates $(P, Q, R, \ldots)$
- You can use any integer, and expressions such as "$+2$"
- You are allowed to use "$=$" and "$<$" between integers
- You can use both first-order variables $(x, y, z, \ldots)$ and second-order monadic variables $(X, Y, Z, \ldots)$
  - You can then write "$X(y)$" or "$y \in X$" (same with P etc.)
- You are allowed first- and second-order quantifiers, and all Boolean connectives

Example: express *"P eventually contains all even integers"*

$$\exists X \,.\, 0 \in X \wedge 1 \notin X \wedge \forall x \,.\, (x \in X \leftrightarrow x + 2 \in X) \wedge$$
$$\exists y \,.\, \forall z > y \,.\, z \in X \rightarrow z \in P$$

Question: Is this specification prefix-independent?

MSO — Monadic Second-Order logic (of order)

- The domain is the set of natural numbers $\mathbb{N}$
- You are given a finite collection of monadic predicates $(P, Q, R, \ldots)$
- You can use any integer, and expressions such as "$+2$"
- You are allowed to use "$=$" and "$<$" between integers
- You can use both first-order variables $(x, y, z, \ldots)$ and second-order monadic variables $(X, Y, Z, \ldots)$
  - You can then write "$X(y)$" or "$y \in X$" (same with P etc.)
- You are allowed first- and second-order quantifiers, and all Boolean connectives

Example: express *"P eventually contains all even integers"*

$$\exists X \,.\, 0 \in X \wedge 1 \notin X \wedge \forall x \,.\, (x \in X \leftrightarrow x + 2 \in X) \wedge$$
$$\exists y \,.\, \forall z > y \,.\, z \in X \rightarrow z \in P$$

Question: Is this specification prefix-independent? **NO!**

Classes of semialgebraic predicates

We work in ambient space $\mathbb{R}^d$

Classes of semialgebraic predicates

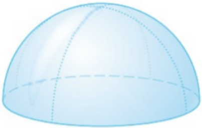
We work in ambient space $\mathbb{R}^d$

Definition ($\mathcal{S}$: the semialgebraic sets)

A set is *semialgebraic* if it can be defined as a Boolean combination of polynomial inequalities.

Classes of semialgebraic predicates

We work in ambient space $\mathbb{R}^d$

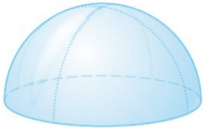


Definition ($\mathcal{S}$: the semialgebraic sets)

A set is *semialgebraic* if it can be defined as a Boolean combination of polynomial inequalities.

Classes of semialgebraic predicates

We work in ambient space $\mathbb{R}^d$



Definition ($\mathcal{S}$: the semialgebraic sets)

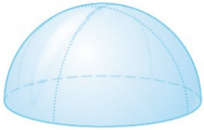
A set is *semialgebraic* if it can be defined as a Boolean combination of polynomial inequalities.

Definition ($\mathcal{C}$: the constructible sets)

A set is *constructible* if it can be defined as a Boolean combination of polynomial equalities. i.e., $\mathcal{T}$ = Boolean closure of *algebraic* sets.

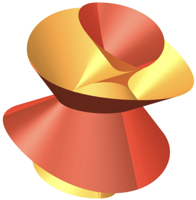
Classes of semialgebraic predicates

We work in ambient space $\mathbb{R}^d$



Definition ($\mathcal{S}$: the semialgebraic sets)

A set is *semialgebraic* if it can be defined as a Boolean combination of polynomial inequalities.

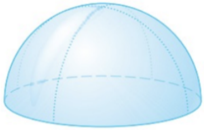


Definition ($\mathcal{C}$: the constructible sets)

A set is *constructible* if it can be defined as a Boolean combination of polynomial equalities. i.e., $\mathcal{T}$ = Boolean closure of *algebraic* sets.

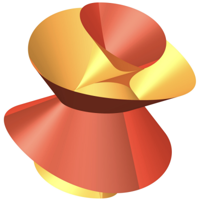
Classes of semialgebraic predicates

We work in ambient space $\mathbb{R}^d$



Definition ($\mathcal{S}$: the semialgebraic sets)

A set is *semialgebraic* if it can be defined as a Boolean combination of polynomial inequalities.



Definition ($\mathcal{C}$: the constructible sets)

A set is *constructible* if it can be defined as a Boolean combination of polynomial equalities. i.e., $\mathcal{T}$ = Boolean closure of *algebraic* sets.

Definition ($\mathcal{T}$: the tame sets)

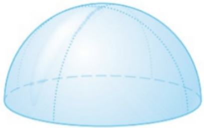
The class $\mathcal{T}$ comprises all semialgebraic sets that are *either* contained in a three-dimensional subspace of $\mathbb{R}^d$, *or* that have intrinsic dimension at most one. $\mathcal{T}$ is defined to be the smallest such class which is closed under all Boolean operations.

Classes of semialgebraic predicates

We work in ambient space $\mathbb{R}^d$

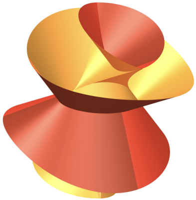
Definition ($\mathcal{S}$: the semialgebraic sets)

A set is *semialgebraic* if it can be defined as a Boolean combination of polynomial inequalities.



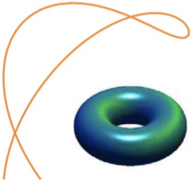
Definition ($\mathcal{C}$: the constructible sets)

A set is *constructible* if it can be defined as a Boolean combination of polynomial equalities. i.e., $\mathcal{T} =$ Boolean closure of *algebraic* sets.

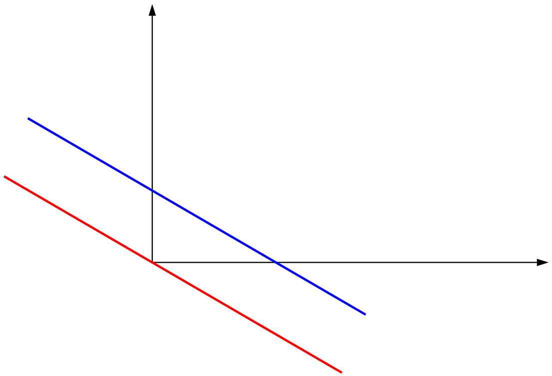


Definition ($\mathcal{T}$: the tame sets)

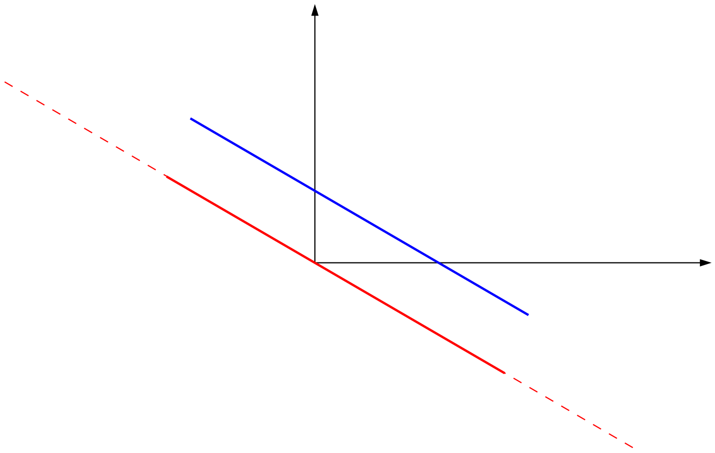
The class $\mathcal{T}$ comprises all semialgebraic sets that are *either* contained in a three-dimensional subspace of $\mathbb{R}^d$, *or* that have intrinsic dimension at most one. $\mathcal{T}$ is defined to be the smallest such class which is closed under all Boolean operations.



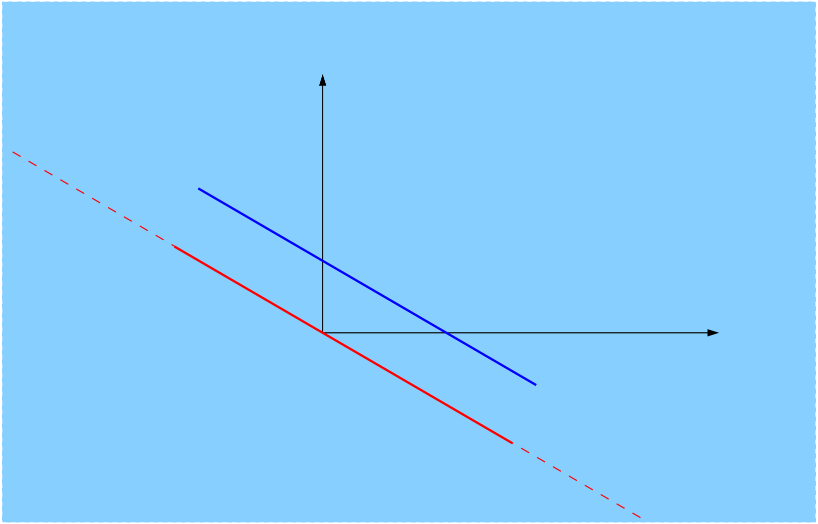
Examples in $\mathbb{R}^2$: two line segments



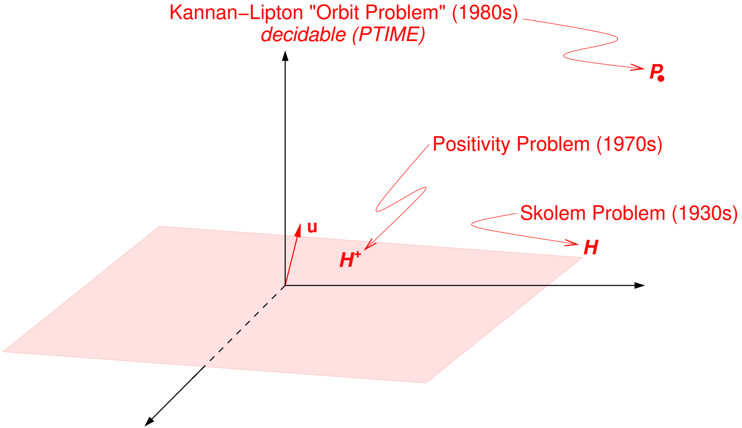
Examples in $\mathbb{R}^2$: two line segments



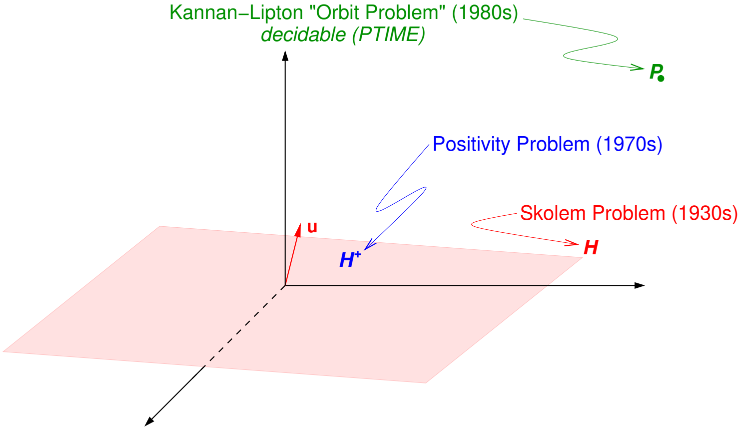
Examples in $\mathbb{R}^2$: two line segments



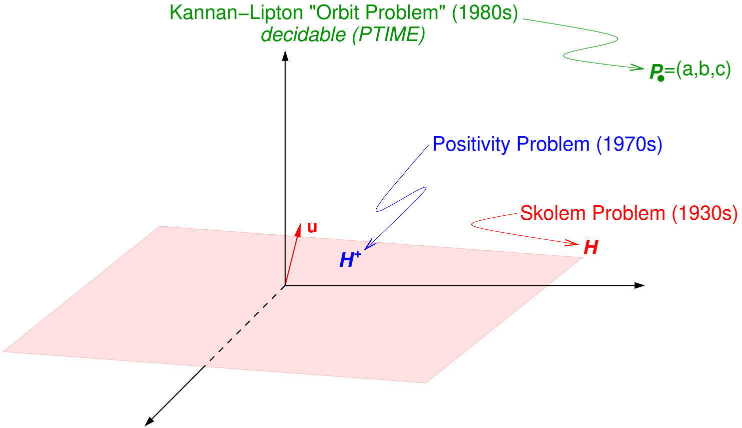
Examples in $\mathbb{R}^3$, $\mathbb{R}^4$, and $\mathbb{R}^5$



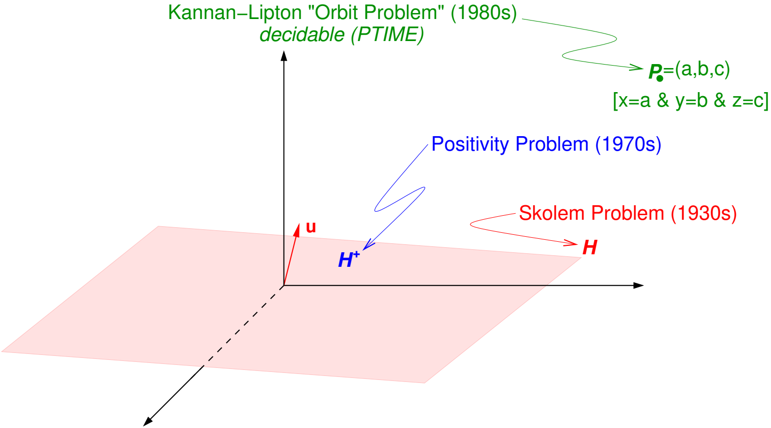
Examples in $\mathbb{R}^3$, $\mathbb{R}^4$, and $\mathbb{R}^5$



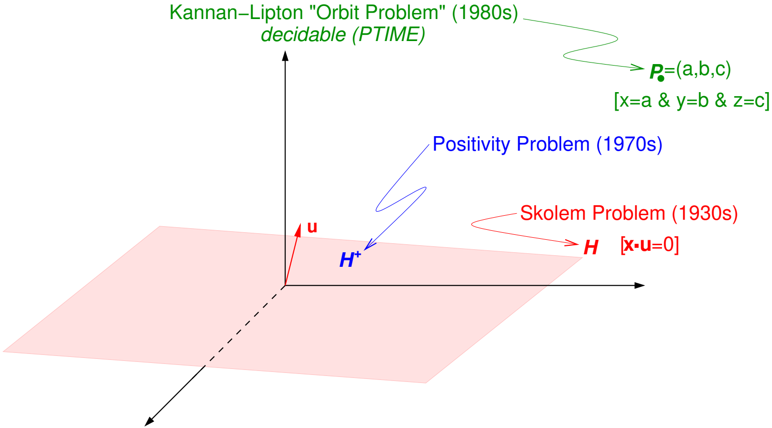
Examples in $\mathbb{R}^3$, $\mathbb{R}^4$, and $\mathbb{R}^5$



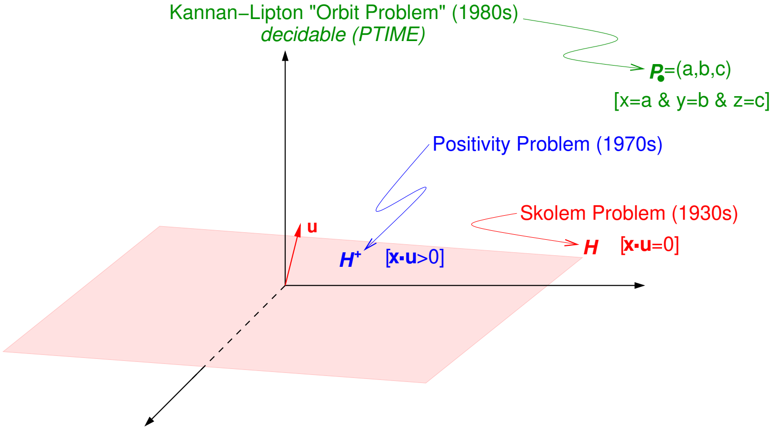
Examples in $\mathbb{R}^3$, $\mathbb{R}^4$, and $\mathbb{R}^5$



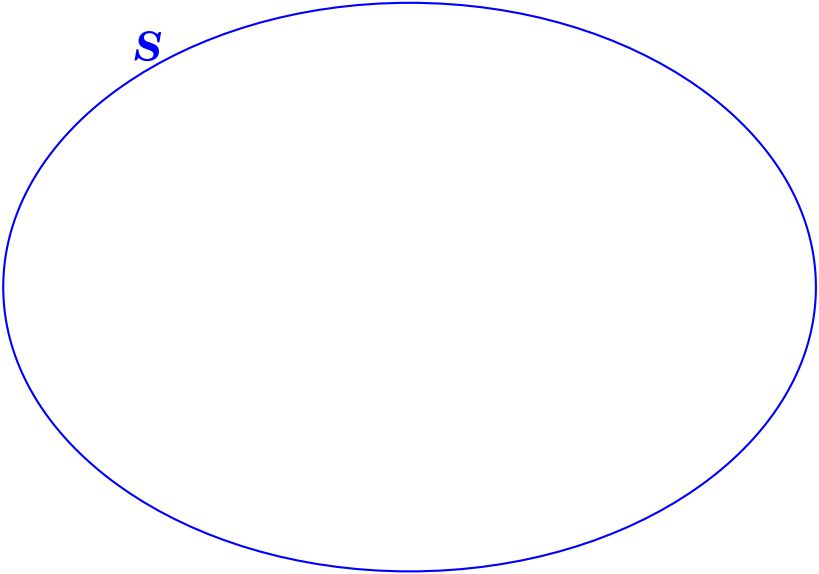
Examples in $\mathbb{R}^3$, $\mathbb{R}^4$, and $\mathbb{R}^5$



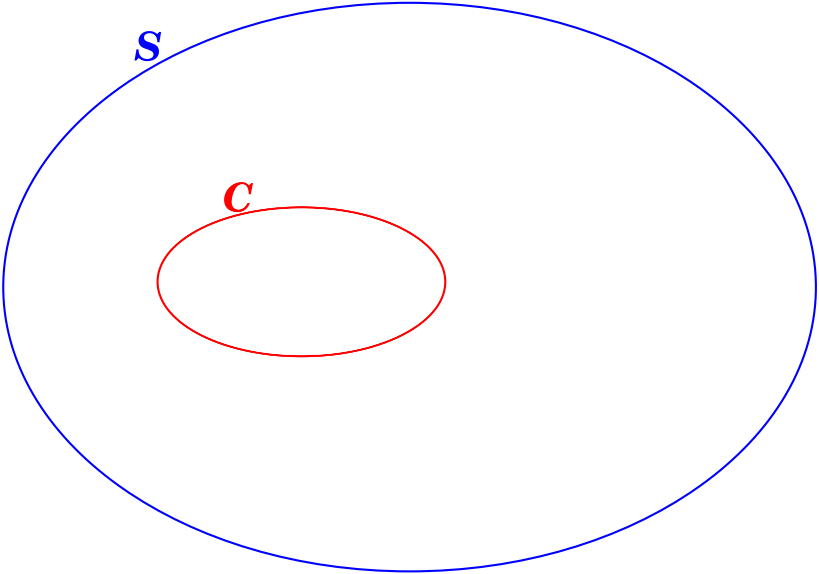
Examples in $\mathbb{R}^3$, $\mathbb{R}^4$, and $\mathbb{R}^5$



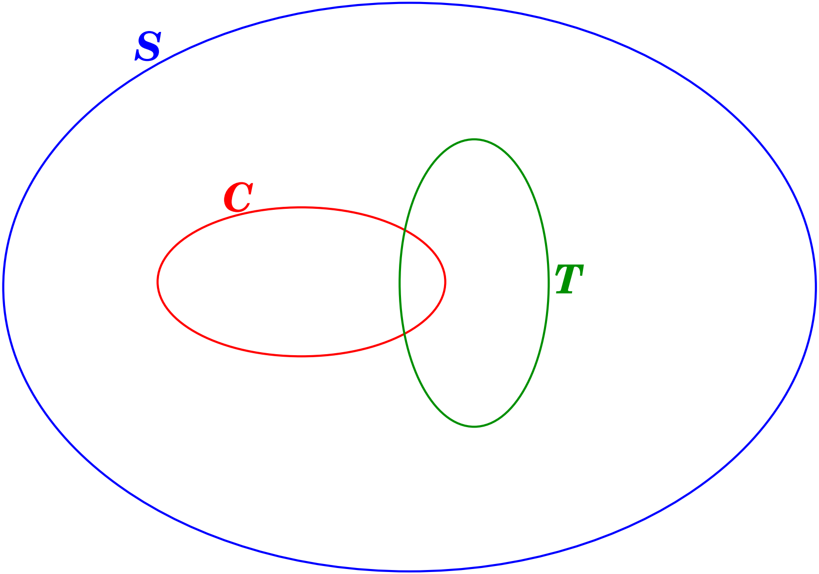
Classes of predicates: summary



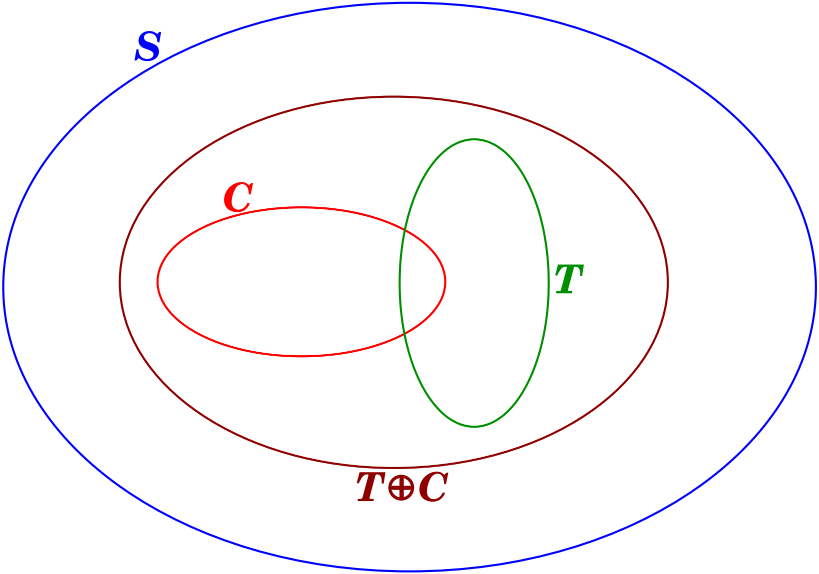
Classes of predicates: summary



Classes of predicates: summary



Classes of predicates: summary



Model checking discrete linear dynamical systems

Model checking discrete linear dynamical systems

	arbitrary LDS	diagonalisable LDS
MSO		
piMSO		

Model checking discrete linear dynamical systems

- unconditional decidability

	arbitrary LDS	diagonalisable LDS
MSO		
piMSO		

Model checking discrete linear dynamical systems

- unconditional decidability

	arbitrary LDS	diagonalisable LDS
MSO	$\mathcal{T}$	$\mathcal{T}$
piMSO		

Model checking discrete linear dynamical systems

- unconditional decidability

	arbitrary LDS	diagonalisable LDS
MSO	$\mathcal{T}$	$\mathcal{T}$
piMSO	$\mathcal{T} \oplus \mathcal{C}$	

Model checking discrete linear dynamical systems

- unconditional decidability

	arbitrary LDS	diagonalisable LDS
MSO	$\mathcal{T}$	$\mathcal{T}$
piMSO	$\mathcal{T} \oplus \mathcal{C}$	$\mathcal{S}$

Model checking discrete linear dynamical systems

- unconditional decidability
- assuming a Skolem oracle

	arbitrary LDS	diagonalisable LDS
MSO	$\mathcal{T}$	$\mathcal{T}$
piMSO	$\mathcal{T} \oplus \mathcal{C}$	$\mathcal{S}$

Model checking discrete linear dynamical systems

- unconditional decidability
- assuming a Skolem oracle

	arbitrary LDS	diagonalisable LDS
MSO	$\mathcal{T}$ / $\mathcal{T} \oplus \mathcal{C}$	$\mathcal{T}$ / $\mathcal{T} \oplus \mathcal{C}$
piMSO	$\mathcal{T} \oplus \mathcal{C}$	$\mathcal{S}$

Model checking discrete linear dynamical systems

- unconditional decidability
- assuming a Skolem oracle

	arbitrary LDS	diagonalisable LDS
MSO	$\mathcal{T}$ / $\mathcal{T} \oplus \mathcal{C}$	$\mathcal{T}$ / $\mathcal{T} \oplus \mathcal{C}$
piMSO	$\mathcal{T} \oplus \mathcal{C}$ / $\mathcal{T} \oplus \mathcal{C}$	$\mathcal{S}$ / $\mathcal{S}$

Model checking discrete linear dynamical systems

- unconditional decidability
- assuming a Skolem oracle
- assuming a Positivity oracle

	arbitrary LDS	diagonalisable LDS
MSO	$\mathcal{T}$ / $\mathcal{T} \oplus \mathcal{C}$	$\mathcal{T}$ / $\mathcal{T} \oplus \mathcal{C}$
piMSO	$\mathcal{T} \oplus \mathcal{C}$ / $\mathcal{T} \oplus \mathcal{C}$	$\mathcal{S}$ / $\mathcal{S}$

Model checking discrete linear dynamical systems

- unconditional decidability
- assuming a Skolem oracle
- assuming a Positivity oracle

	arbitrary LDS	diagonalisable LDS
MSO	$\mathcal{T}$ / $\mathcal{T} \oplus \mathcal{C}$ / $\mathcal{T} \oplus \mathcal{C}$	$\mathcal{T}$ / $\mathcal{T} \oplus \mathcal{C}$
piMSO	$\mathcal{T} \oplus \mathcal{C}$ / $\mathcal{T} \oplus \mathcal{C}$	$\mathcal{S}$ / $\mathcal{S}$

Model checking discrete linear dynamical systems

- unconditional decidability
- assuming a Skolem oracle
- assuming a Positivity oracle

	arbitrary LDS	diagonalisable LDS
MSO	$\mathcal{T}$ / $\mathcal{T} \oplus \mathcal{C}$ / $\mathcal{T} \oplus \mathcal{C}$	$\mathcal{T}$ / $\mathcal{T} \oplus \mathcal{C}$ / $\mathcal{S}$
piMSO	$\mathcal{T} \oplus \mathcal{C}$ / $\mathcal{T} \oplus \mathcal{C}$	$\mathcal{S}$ / $\mathcal{S}$

Model checking discrete linear dynamical systems

- unconditional decidability
- assuming a Skolem oracle
- assuming a Positivity oracle

	arbitrary LDS	diagonalisable LDS
MSO	$\mathcal{T}$ / $\mathcal{T} \oplus \mathcal{C}$ / $\mathcal{T} \oplus \mathcal{C}$	$\mathcal{T}$ / $\mathcal{T} \oplus \mathcal{C}$ / $\mathcal{S}$
piMSO	$\mathcal{T} \oplus \mathcal{C}$ / $\mathcal{T} \oplus \mathcal{C}$ / $\mathcal{T} \oplus \mathcal{C}$	$\mathcal{S}$ / $\mathcal{S}$ / $\mathcal{S}$

Model checking discrete linear dynamical systems

- unconditional decidability
- assuming a Skolem oracle
- assuming a Positivity oracle

	arbitrary LDS	diagonalisable LDS
MSO	$\mathcal{T}$ / $\mathcal{T} \oplus \mathcal{C}$ / $\mathcal{T} \oplus \mathcal{C}$	$\mathcal{T}$ / $\mathcal{T} \oplus \mathcal{C}$ / $\mathcal{S}$
piMSO	$\mathcal{T} \oplus \mathcal{C}$ / $\mathcal{T} \oplus \mathcal{C}$ / $\mathcal{T} \oplus \mathcal{C}$	$\mathcal{S}$ / $\mathcal{S}$ / $\mathcal{S}$

In a precise sense, all these results are *tight:*
improving them runs into Skolem-hardness,
or Positivity-hardness, or Diophantine-hardness

Model checking discrete linear dynamical systems

- unconditional decidability
- assuming a Skolem oracle
- assuming a Positivity oracle

	arbitrary LDS	diagonalisable LDS
MSO	$\mathcal{T}$ / $\mathcal{T} \oplus \mathcal{C}$ / $\mathcal{T} \oplus \mathcal{C}$	$\mathcal{T}$ / $\mathcal{T} \oplus \mathcal{C}$ / $\mathcal{S}$
piMSO	$\mathcal{T} \oplus \mathcal{C}$ / $\mathcal{T} \oplus \mathcal{C}$ / $\mathcal{T} \oplus \mathcal{C}$	$\mathcal{S}$ / $\mathcal{S}$ / $\mathcal{S}$

In a precise sense, all these results are *tight:*
improving them runs into Skolem-hardness,
or Positivity-hardness, or Diophantine-hardness

Moreover, our unconditional decidability algorithm
can produce *correctness certificates!*

Decidability of MSO and extensions

Büchi showed in 1962 that MSO is decidable

Decidability of MSO and extensions

Büchi showed in 1962 that MSO is decidable

Central Question: what kinds of predicates can we add to MSO whilst retaining decidability?

Decidability of MSO and extensions

Büchi showed in 1962 that MSO is decidable

Central Question: what kinds of predicates can we add to MSO whilst retaining decidability?

For example, let $P \subseteq \mathbb{N}$ be the set of prime numbers.
Is MSO(P) decidable??

Decidability of MSO and extensions

Büchi showed in 1962 that MSO is decidable

Central Question: what kinds of predicates can we add to MSO whilst retaining decidability?

For example, let $P \subseteq \mathbb{N}$ be the set of prime numbers.
Is MSO(P) decidable??

This is open! But appears very difficult, e.g.

$$\forall x \,.\, \exists y > x \,.\, P(y) \wedge P(y+2)$$

Decidability of MSO and extensions

Theorem (Semënov, 1984)

If P is **effectively almost-periodic**, *then MSO(P) is decidable.*

Decidability of MSO and extensions

Theorem (Semënov, 1984)

If P is **effectively almost-periodic***, then MSO(P) is decidable.*

A word w is (effectively) almost-periodic if for every finite word u, we can bound the gaps between consecutive occurrences of u in w:

Decidability of MSO and extensions

Theorem (Semënov, 1984)

If P is **effectively almost-periodic**, *then MSO(P) is decidable.*

What about MSO$(P_1, P_2, \ldots, P_k)$?

Decidability of MSO and extensions

Theorem (Semënov, 1984)

If P is **effectively almost-periodic**, *then $MSO(P)$ is decidable.*

What about $\mathrm{MSO}(P_1, P_2, \ldots, P_k)$?

Theorem (Semënov, 1983)

One can define P_1 and P_2 both effectively almost-periodic, such that $MSO(P_1, P_2)$ is undecidable!

Decidability of MSO and extensions

Theorem (Semënov, 1984)

If P is **effectively almost-periodic***, then MSO(P) is decidable.*

What about MSO$(P_1, P_2, \ldots, P_k)$?

Theorem (Semënov, 1983)

One can define P_1 and P_2 both effectively almost-periodic, such that MSO(P_1, P_2) is undecidable!

Much (ongoing) work on this central question! By e.g., Elgot, Rabin, Carton, Thomas, Rabinovich, Fijalkow, Paperman, . . .

Toric words

So how can one ensure that $\mathrm{MSO}(P_1, \ldots, P_k)$ is decidable?

Toric words

So how can one ensure that $\mathrm{MSO}(P_1, \ldots, P_k)$ is decidable?

To answer this question in the context of linear dynamical systems, we developed the theory of **(ultimately) toric words**

Toric words

So how can one ensure that $\mathrm{MSO}(P_1, \ldots, P_k)$ is decidable?

To answer this question in the context of linear dynamical systems, we developed the theory of **(ultimately) toric words**

Theorem

1. *Ultimately toric words are almost-periodic.*
2. *Effectively ultimately toric words are closed under products.*

Toric words

So how can one ensure that $\mathrm{MSO}(P_1, \ldots, P_k)$ is decidable?

To answer this question in the context of linear dynamical systems, we developed the theory of **(ultimately) toric words**

Theorem

1. *Ultimately toric words are almost-periodic.*
2. *Effectively ultimately toric words are closed under products.*

Theorem

Tame predicates give rise to effectively ultimately toric words.

Toric words

So how can one ensure that $\text{MSO}(P_1, \ldots, P_k)$ is decidable?

To answer this question in the context of linear dynamical systems, we developed the theory of **(ultimately) toric words**

Theorem

1. *Ultimately toric words are almost-periodic.*
2. *Effectively ultimately toric words are closed under products.*

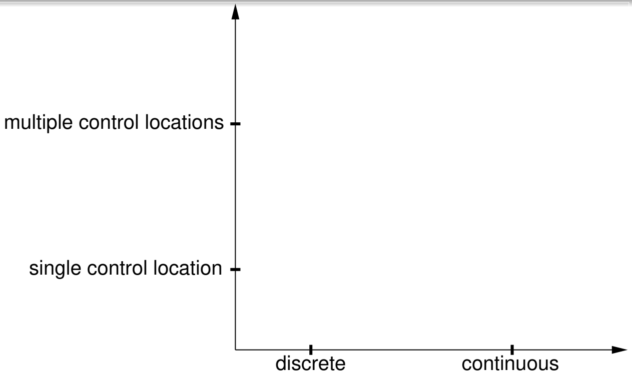
Theorem

Tame predicates give rise to effectively ultimately toric words.

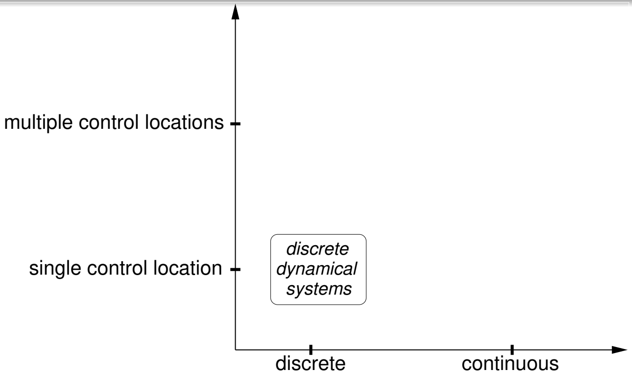
Corollary

Let (M, s) be a linear dynamical system in ambient space $\mathbb{R}^d$, and let $S_1, \ldots, S_k \subseteq \mathbb{R}^d$ be tame semialgebraic predicates.
Let $P_1, \ldots, P_k \subseteq \mathbb{N}$ be the set of visiting times of the orbit of (M, s) in $S_1, \ldots, S_k$ respectively.
Then $MSO(P_1, \ldots, P_k)$ is decidable.

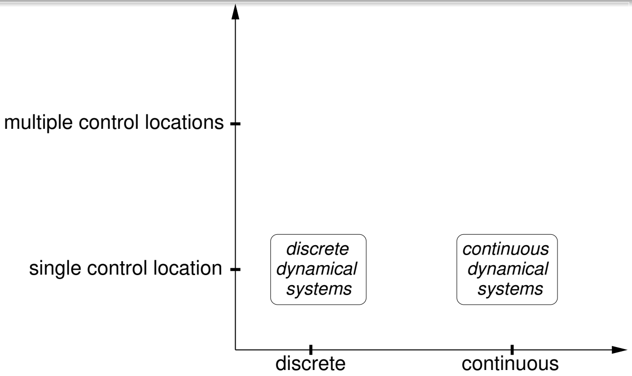
The Algorithmic Theory of Linear Systems



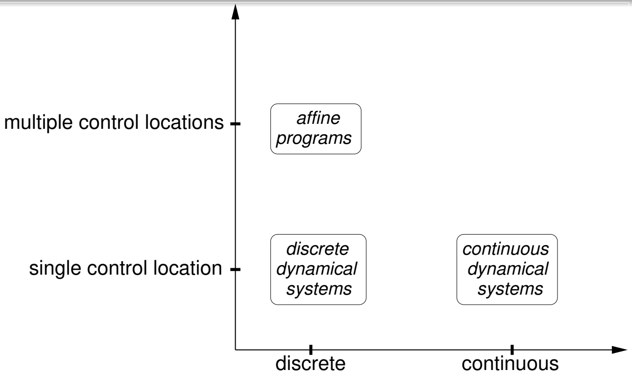
The Algorithmic Theory of Linear Systems



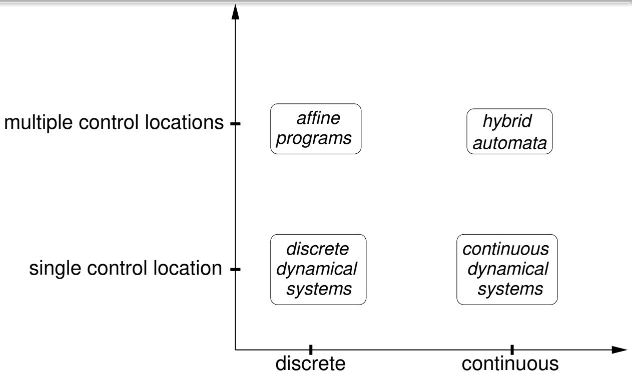
The Algorithmic Theory of Linear Systems



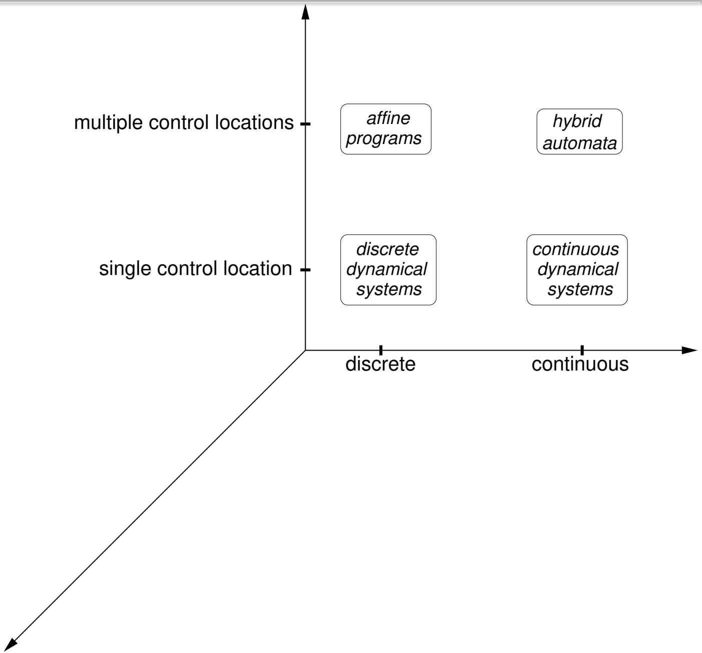
The Algorithmic Theory of Linear Systems



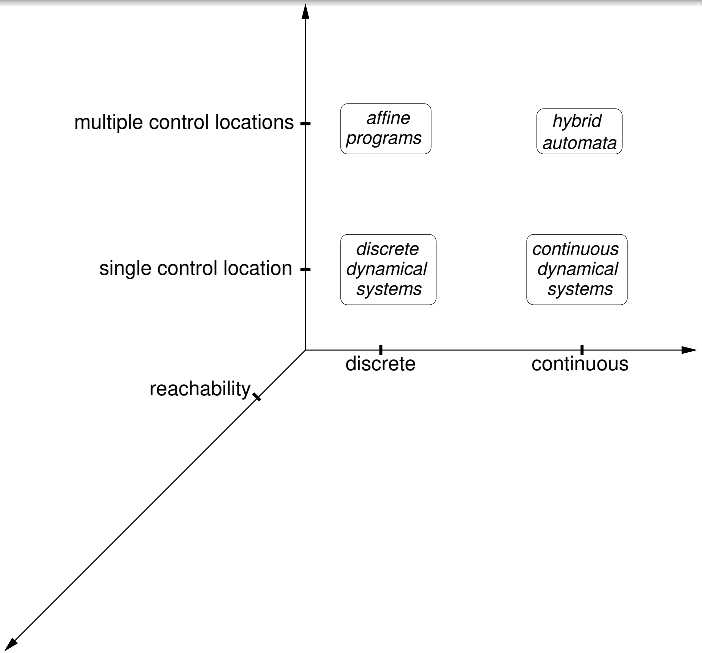
The Algorithmic Theory of Linear Systems



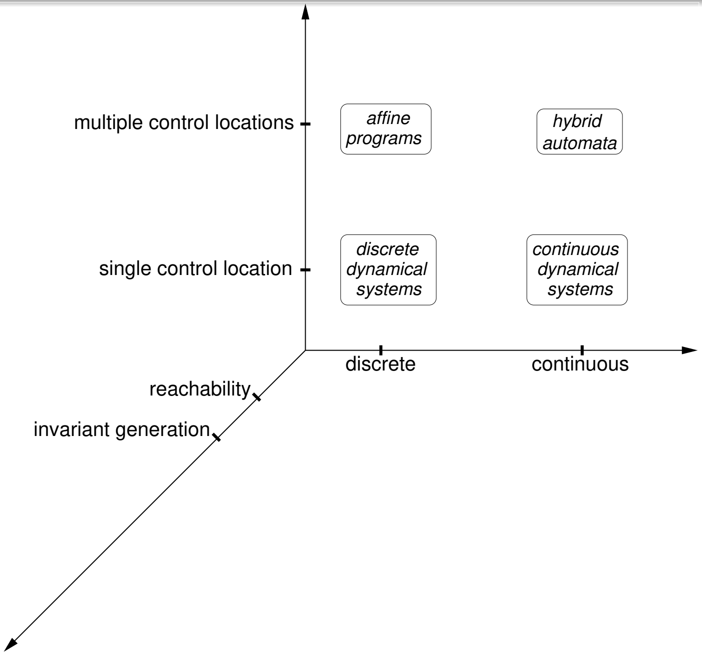
The Algorithmic Theory of Linear Systems



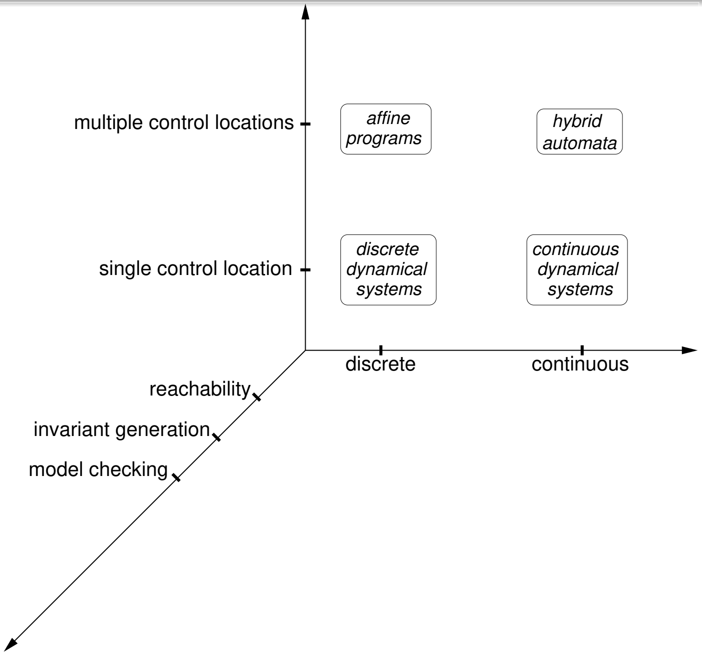
The Algorithmic Theory of Linear Systems



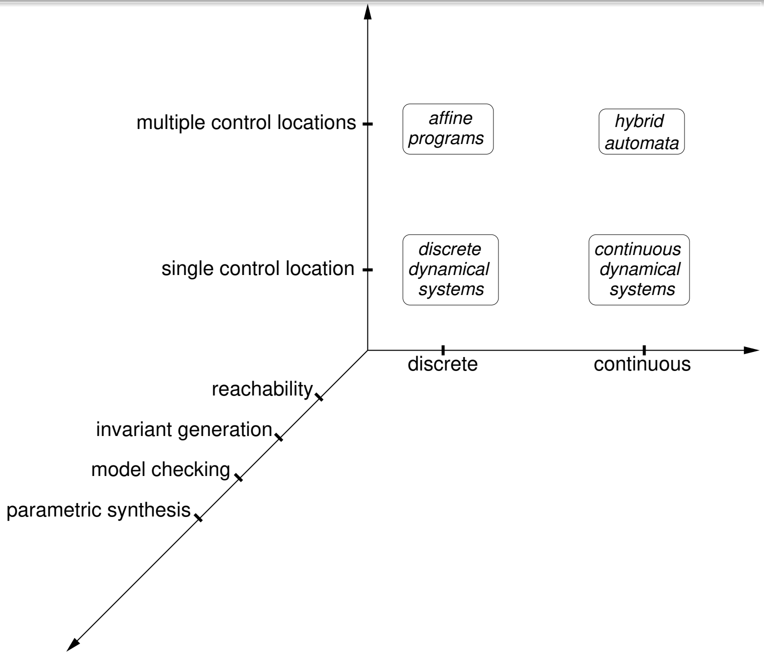
The Algorithmic Theory of Linear Systems



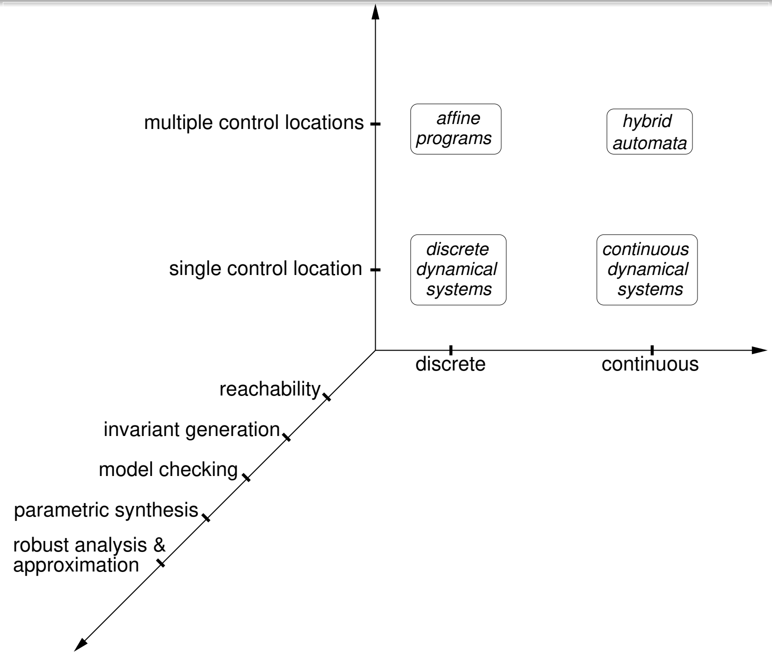
The Algorithmic Theory of Linear Systems



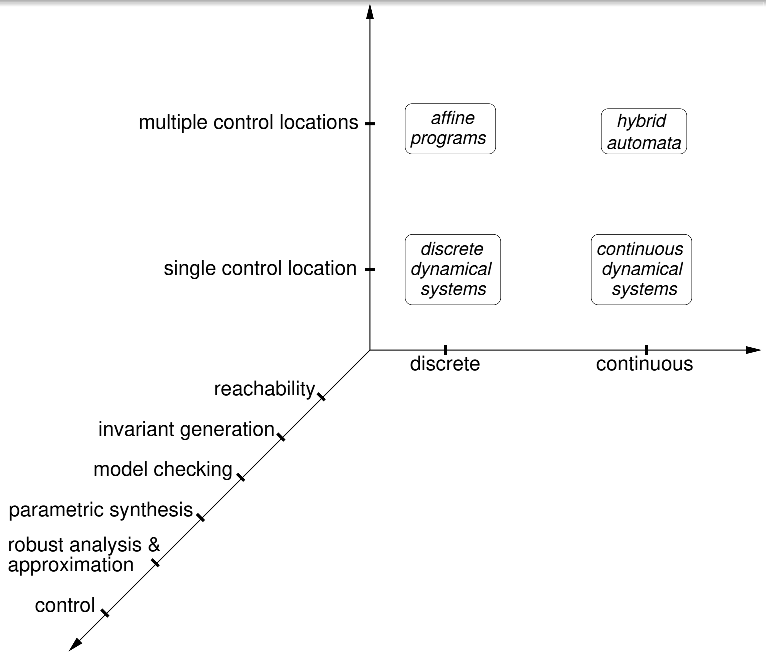
The Algorithmic Theory of Linear Systems



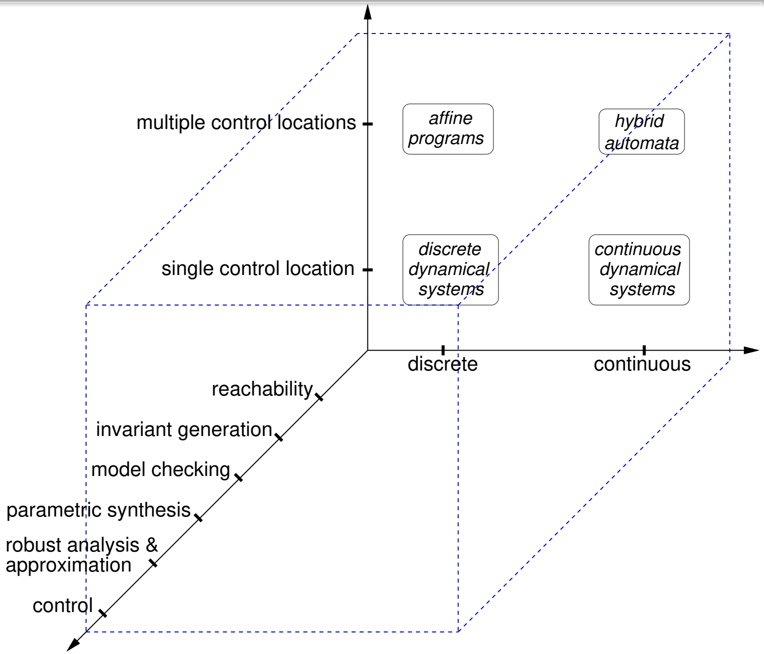
The Algorithmic Theory of Linear Systems



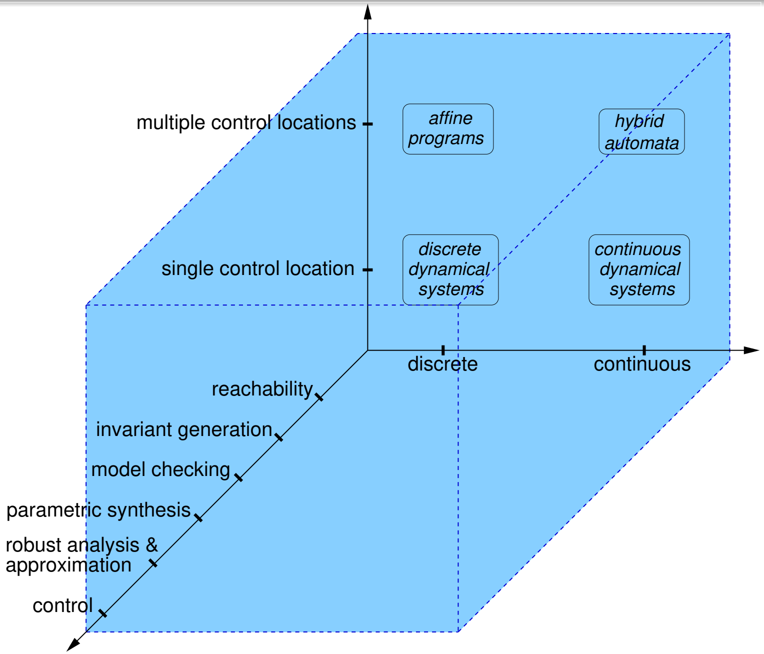
The Algorithmic Theory of Linear Systems



The Algorithmic Theory of Linear Systems



The Algorithmic Theory of Linear Systems



The Algorithmic Theory of Linear Systems