Mining and Proving Conjectures: Discovering Invariants on Integer Sequences

(with an application for short term replanification)

Ekaterina Arafailova, Nicolas Beldiceanu, and Helmut Simonis

14th November 2017

PGMO

The Question Motivating this Work

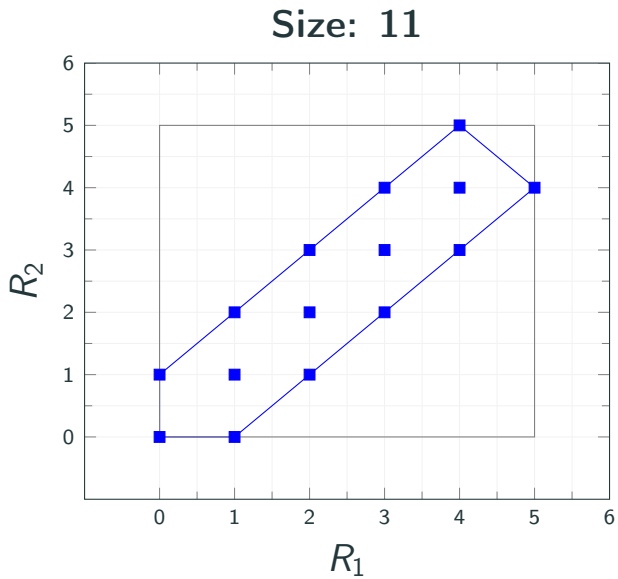
Consider two constraints $\gamma_1(\langle X_1, \ldots, X_n \rangle, R_1)$ and $\gamma_2(\langle X_1, \ldots, X_n \rangle, R_2)$, where each X_i is in $\{1, \ldots, n\}$.

The variables R_1 and R_2 are constrained to be the result of some computations over $\langle X_1, \ldots, X_n \rangle$ depending only on the relations between $X_1, X_2, \ldots, X_k$.

For example, R_1 is the number of peaks in $\langle X_1, \ldots, X_n \rangle$ and R_2 is the number of valleys in $\langle X_1, \ldots, X_n \rangle$.

What is the set of feasible pairs of R_1 and R_2?

Example of Sets of Feasible Pairs of R_1 and R_2: Convex Case

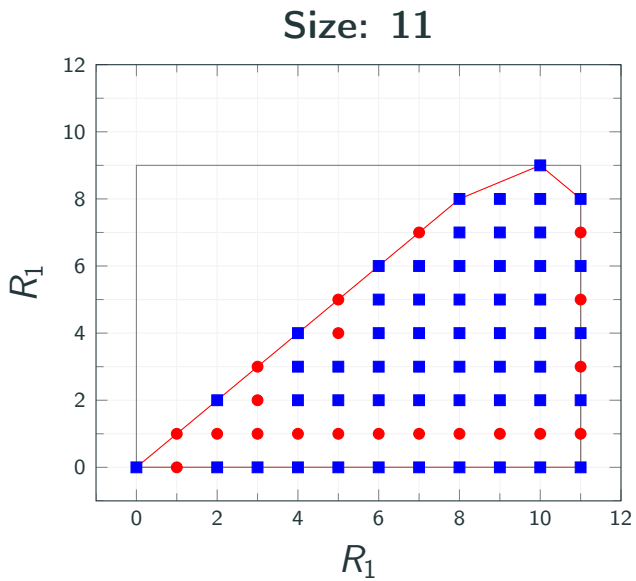


$\gamma_1 = \text{nb\_peak}$

$\gamma_2 = \text{nb\_valley}$

- The set of feasible (blue) points is convex.
- A conjunction of linear inequalities of R_1, R_2 and n.

Example of Sets of Feasible Pairs of R_1 and R_2: Non-Convex Case



$\gamma_1 = \text{sum\_width\_decreasing\_sequence}$

$\gamma_2 = \text{sum\_width\_zigzag}$

- The set of feasible (blue) points is non-convex.
- A conjunction of linear inequalities of is **not enough**.
- A need for **non-linear** characterisation.

Two Emerging Problems for Characterising Infeasible Combinations

1. Generate linear inequalities depending on R_1, R_2 and parameterised by $f(n) \in \{n, n \mod p, \sqrt{n}, \dots\}$, which represent the facets of the convex hull.
2. Generate non-linear invariants eliminating infeasible points on and/or inside of the convex hull.

Two Emerging Problems for Characterising Infeasible Combinations

1. Generate linear inequalities depending on R_1, R_2 and parameterised by $f(n) \in \{n, n \mod p, \sqrt{n}, \dots\}$, which represent the facets of the convex hull.
2. Generate non-linear invariants eliminating infeasible points on and/or inside of the convex hull.

How to solve these two problems in a systematic way for a large family of constraints?

Main Insight...

Two Emerging Problems for Characterising Infeasible Combinations

1. Generate linear inequalities depending on R_1, R_2 and parameterised by $f(n) \in \{n, n \mod p, \sqrt{n}, \dots\}$, which represent the facets of the convex hull.
2. Generate non-linear invariants eliminating infeasible points on and/or inside of the convex hull.

How to solve these two problems in a systematic way for a large family of constraints?

Main Insight...

Use **register automata** and parameterised characterisation.

Take-Away Message

- **Convex Case:** A compositional way of generating cuts from register automata.

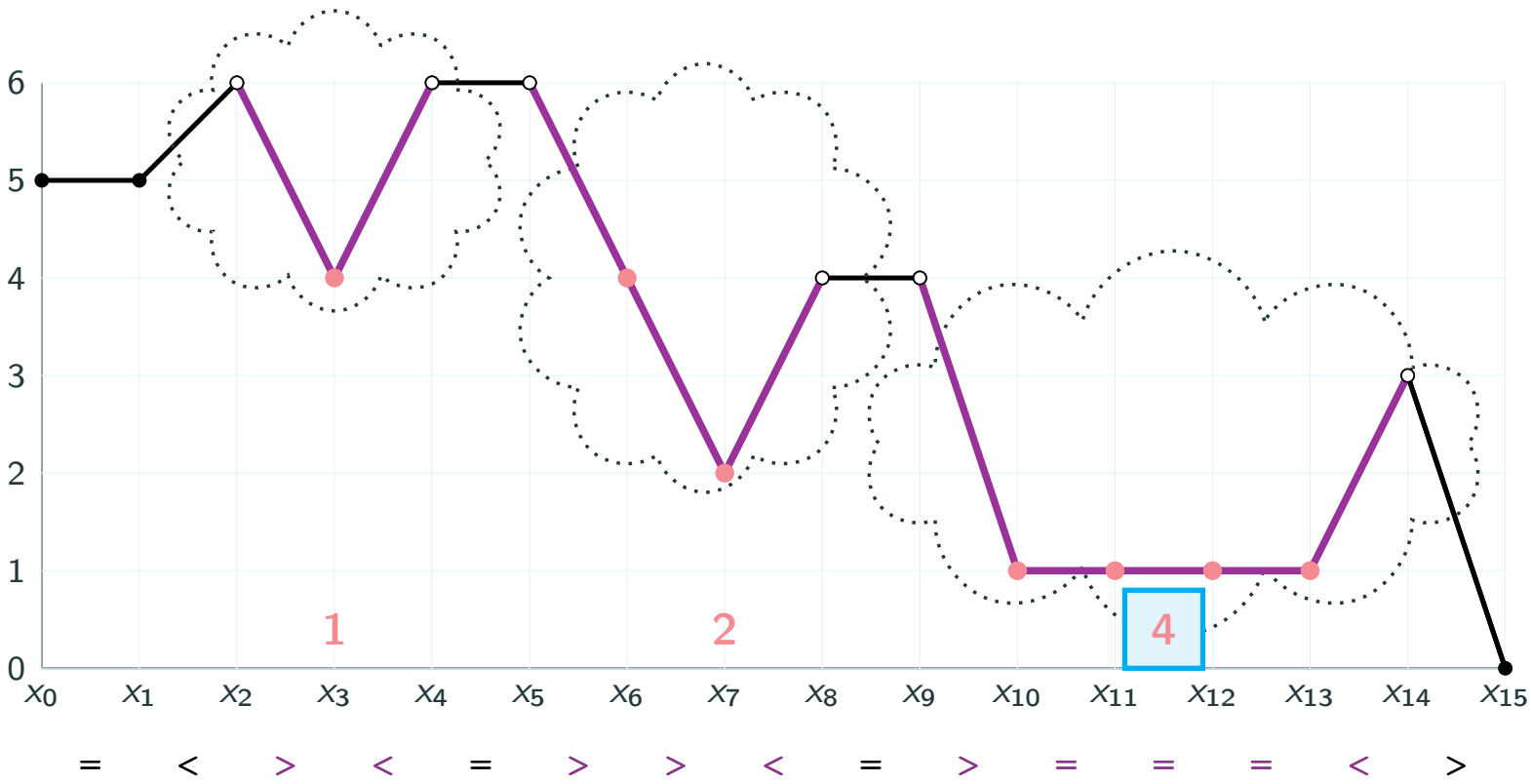
- **Non-Convex Case:** Data Mining + Proof using automata characterising infeasible combinations of points for conjunction of constraints.

Case Study: Time-Series Constraints

- Family of constraints arising from **power systems** modelling.
- Declaratively described by **regular expressions**.
- Baseline implementation as **register automata**.
- Lots of missing propagation, not good enough for use case.
- Work on improving propagators for **all** constraints
  at the same time.

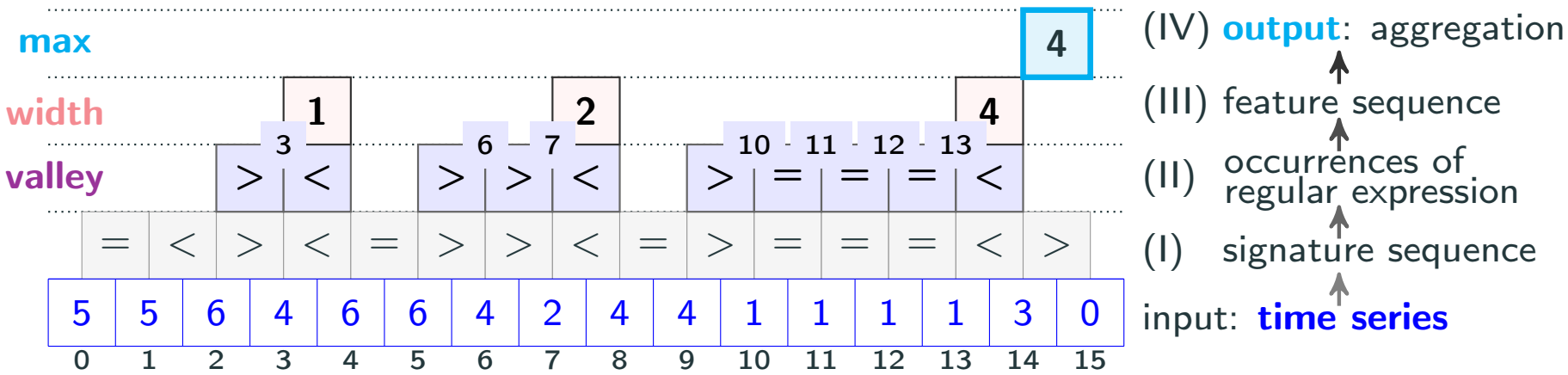
Example of a Time-Series Constraint

Constrain the **maximum** of the **widths** of the **valleys**
in the time series $X = \langle 5, 5, 6, 4, 6, 6, 4, 2, 4, 4, 1, 1, 1, 1, 3, 0 \rangle$.



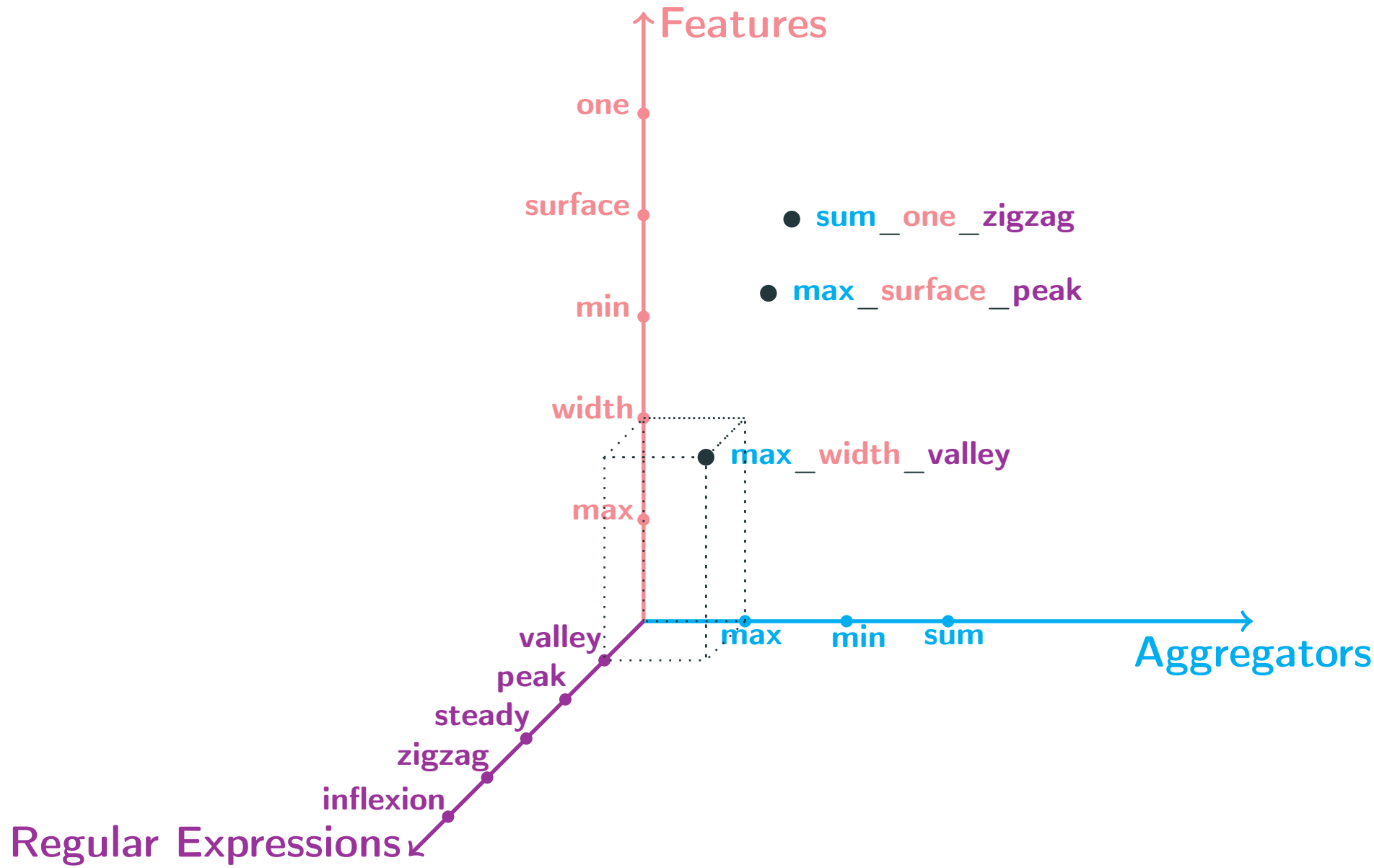
A subsequence $\langle X_i, \ldots, X_j \rangle$ of $\langle X_0, \ldots, X_m \rangle$ is a **valley** if the signature of $\langle X_{i-1}, \ldots, X_{j+1} \rangle$ is a maximal word matching '>(>|=)*(<|=)*<'.

Compositional Time-Series Definition by Multiple Layers of Functions

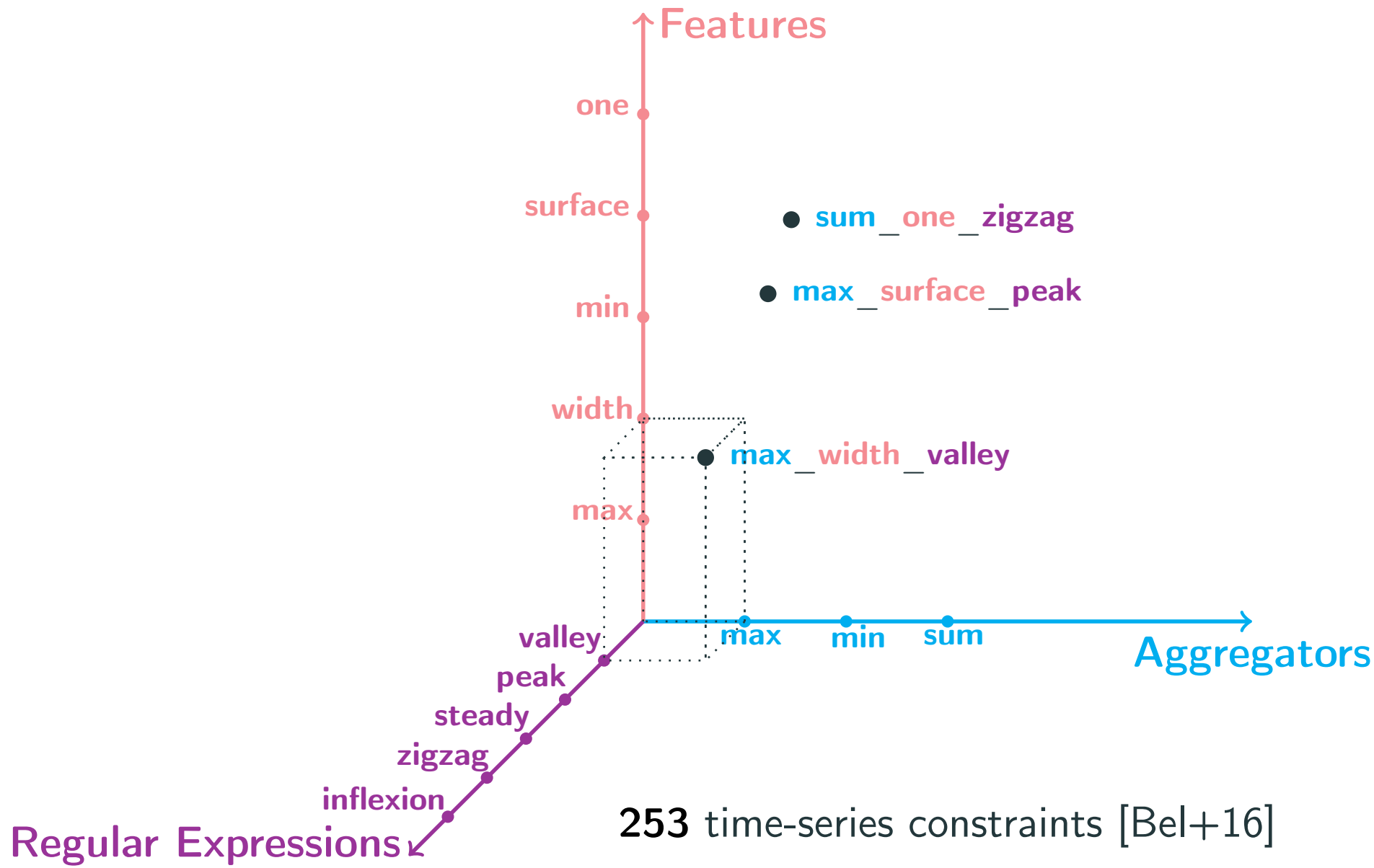


$$\texttt{max\_width\_valley}(\langle 5,5,6,4,6,6,4,2,4,4,1,1,1,1,3,0\rangle, 4)$$

Space of Time-Series Constraints



Space of Time-Series Constraints



253 time-series constraints [Bel+16]

Pratical Motivation of Time-Series Constraints: Unit Commitment Problem for a Power Plant

Let $X = \langle X_1, \ldots, X_n \rangle$ be a production curve of a power plant.

Time-series constraints can be used to implement missing constraints, e.g. **business rules**.

Examples of Business Rules:

- After reaching a production plateau a production unit should stay at the same level for at least k period of times.
  min_width_plateau(X, k).
- No zigzag of production level spanning over more than k periods of times.
  max_width_zigzag(X, k).

Time-Series Constraints Families of This Work

- Only topological constraints,
  i.e. nb_$\sigma(X, R)$ and sum_width_$\sigma(X, R)$.
- R depends only on the relations between X variables,
  but not on their values themselves.
- Representation as register automata
  with linear register updates.
- 35 constraints in the two families.

Generating Linear Inequalities for Two Families of Time-Series Constraints

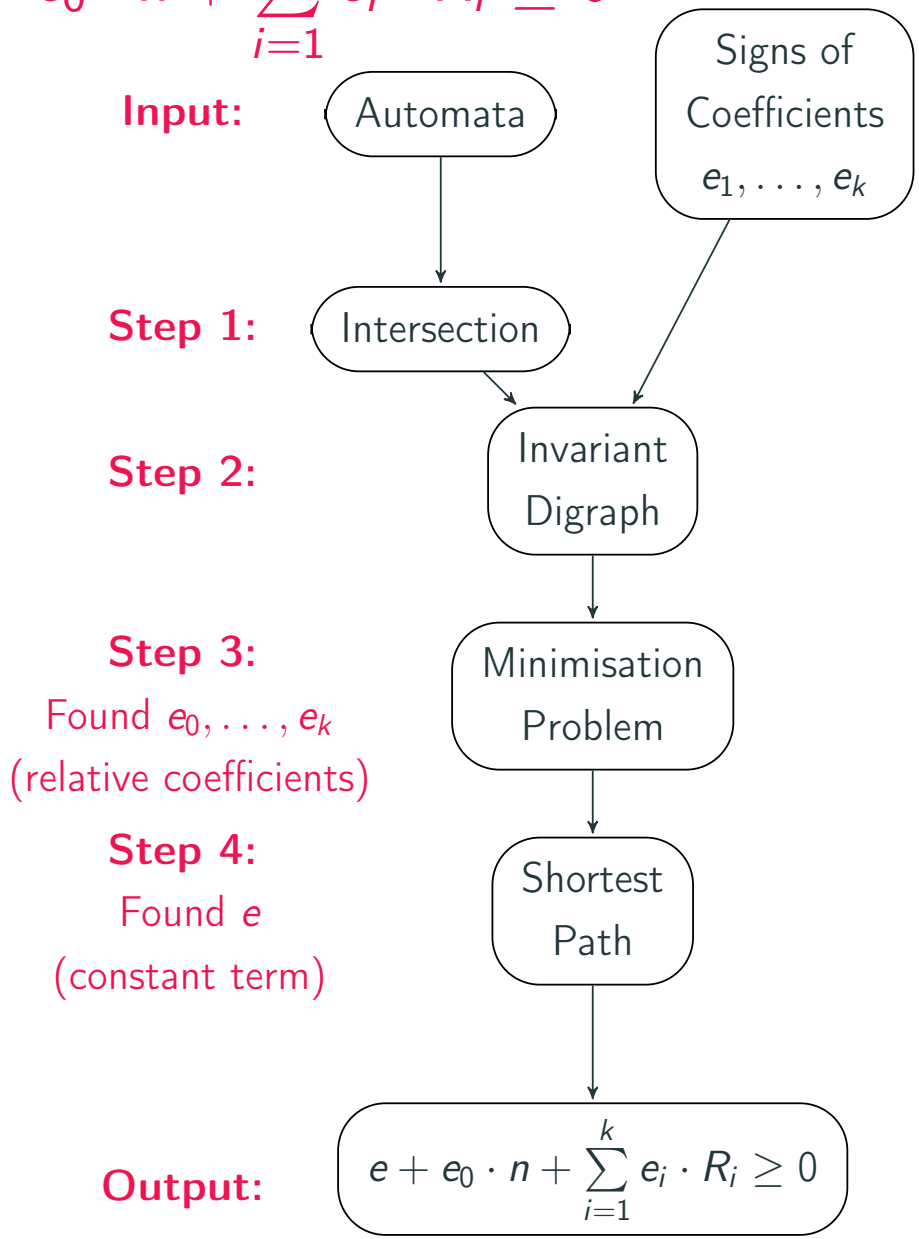
Consider a conjunction of time-series constraints $\gamma_1(X, R_1), \ldots, \gamma_k(X, R_k)$, with X being $\langle X_1, \ldots, X_n \rangle$.

Every γ_i has a representation by a register automaton $\mathcal{M}_i$ satisfying some conditions.

In [ABS17] we proposed a systematic and uniform technique to generate 2^k linear inequalities $e + e_0 \cdot n + \sum_{i=1}^{k} e_i \cdot R_i \geq 0$, with $e, e_0, \ldots, e_k \in \mathbb{Z}$.

These inequalities are true for any sequence recognised by **all** the automata for $\gamma_1, \gamma_2, \ldots, \gamma_k$.

Our Method for Generating Linear Inequalities of the form $e + e_0 \cdot n + \sum_{i=1}^{k} e_i \cdot R_i \geq 0$



Example of Generated Invariants for a Conjunction of Two Constraints

$\text{nb\_peak}(X, R_1) \wedge \text{nb\_valley}(X, R_2)$ with $X = \langle X_1, \ldots, X_n \rangle$

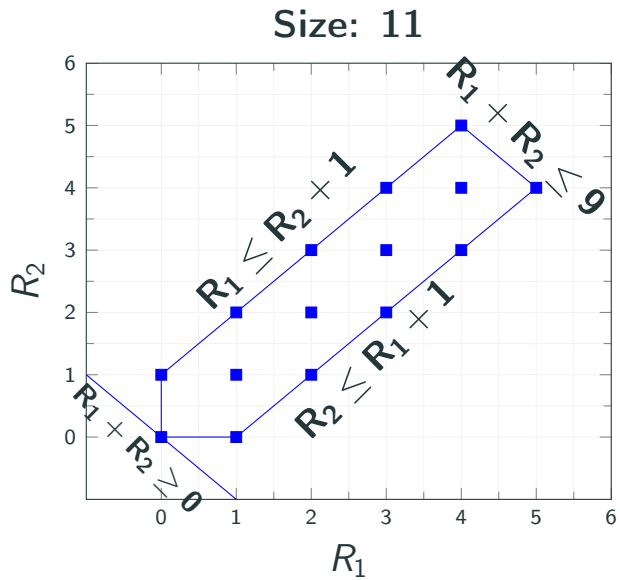
Our method finds the following invariants:

$$R_2 \leq R_1 + 1$$

$$R_1 \leq R_2 + 1$$

$$R_1 + R_2 \leq n - 2$$

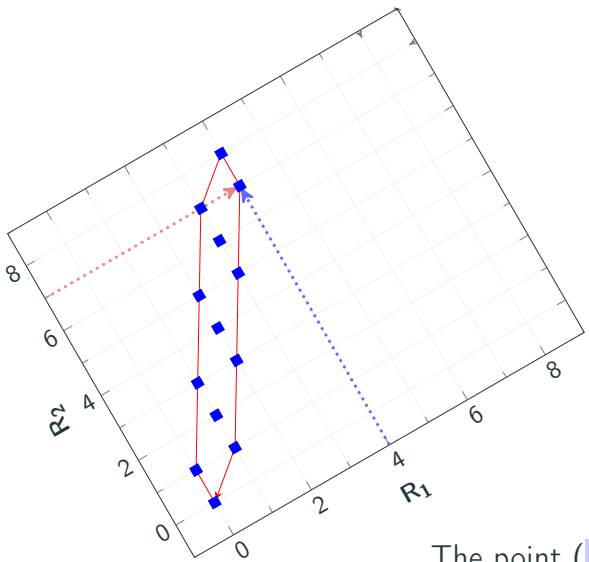
$$R_1 + R_2 \geq 0$$

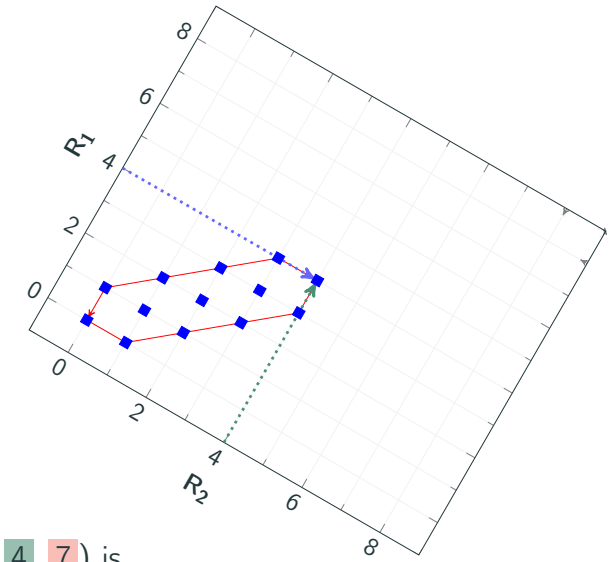


3 out of the 4 found inequalities are facet-defining!

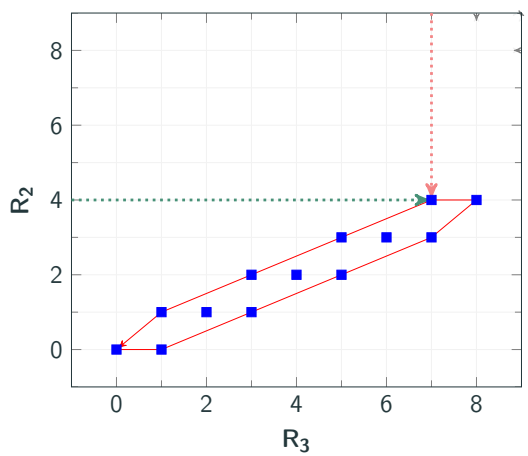
Example of a Generated Invariant for a Conjunction of Three Constraints

$$\text{nb\_peak}(X, R_1) \wedge \text{nb\_valley}(X, R_2) \wedge \text{nb\_inflexion}(X, R_3)$$





The point (4 , 4 , 7) is discarded by $R_1 + R_2 \leq R_3$



Discarded by $R_1 + R_2 \leq R_3$:

- $(4, 4, 7)$
- $(3, 3, 5)$
- $(2, 2, 3)$
- $(1, 1, 1)$

Using Linear Invariants in the Context of Linear Programming

$\text{nb\_peak}(X, P) \wedge \text{nb\_valley}(X, V)$,
where X is a sequence of 100 variables ranging over $[0, 1000]$.

Objective Function: maximise $P - 2 \cdot V$

	LP	LP + cuts
Time	300s	0s
Best objective	1	1
Best bound	49	1
Gap	4800%	0%

LP is a pure linear reformulation [Ara+16]

LP + cuts is a linear reformulation with cuts on every prefix

Generating Non-Linear Invariants that Deal With Missing, Infeasible Cases

Three Phases of our Method:

1. **Generation of Data:** generate all feasible combinations of $R_1, \ldots, R_k$ for a given range of n values.
2. **Mining Phase:** generate hypothesis covering subsets of infeasible points using the generated data.
3. **Proving Phase:** prove the generated hypothesis using register automata.

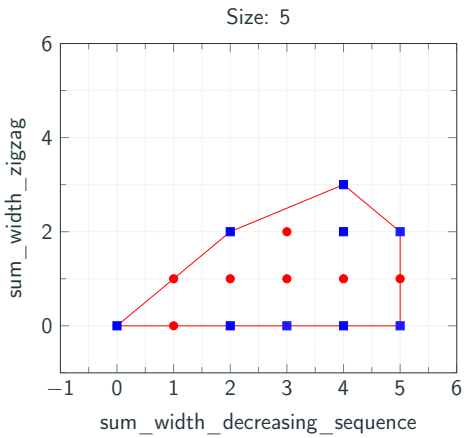
The three phases are **offline** and solver/system **independent**.

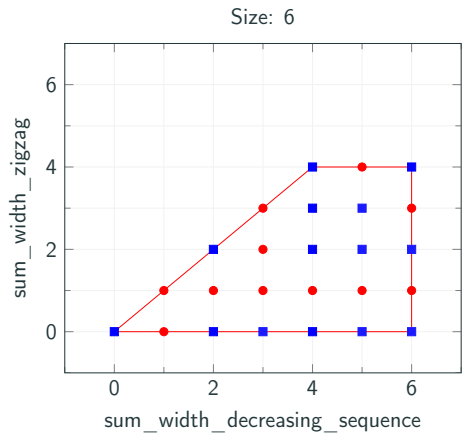
Generation of Data: Our Case

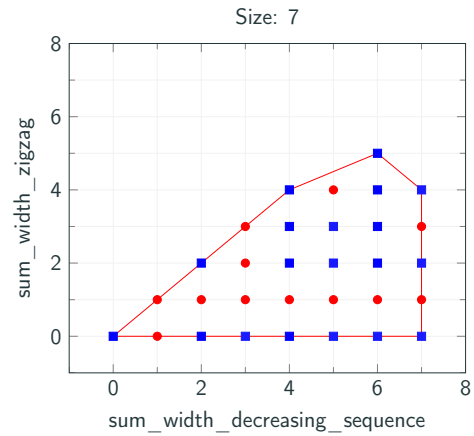
- Pairs of different time-series constraints $\gamma_1(\langle X_1, \ldots, X_n \rangle, R_1)$ and $\gamma_2(\langle X_1, \ldots, X_n \rangle, R_2)$.
- Generate all feasible pairs (R_1, R_2) for $n \in \{1, 2, \ldots, 12\}$.
- Compute the convex hull using Graham's scan.
- Collect all infeasible points inside the convex hull.

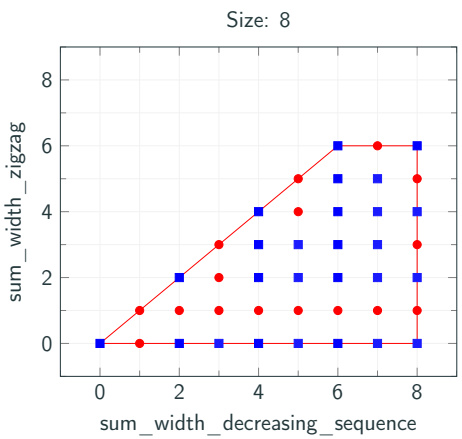
Example of Samples of the Generated Data

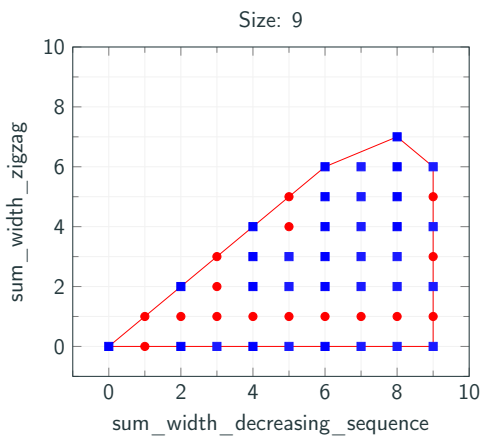
$\gamma_1 = \text{sum\_width\_decreasing\_sequence}$, $\gamma_2 = \text{sum\_width\_zigzag}$

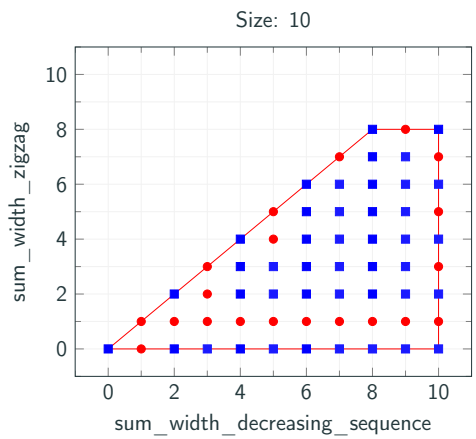








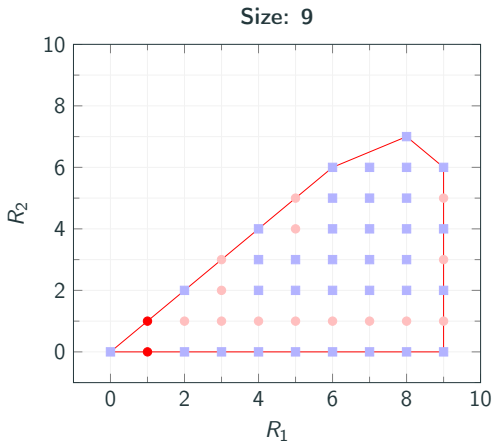




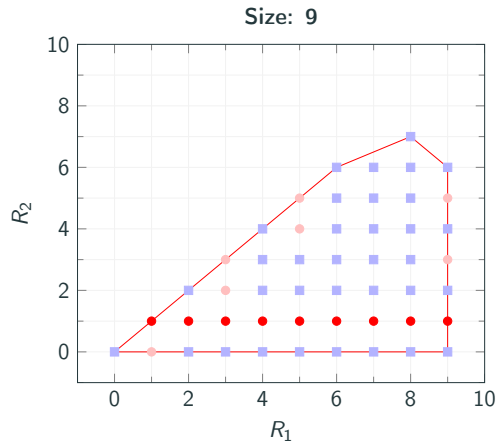
Mining Phase: Generation of Hypothesis

- Consider only samples of sizes from 7 to 12.
- Hypothesis of type $C_1 \wedge C_2 \wedge \cdots \wedge C_p$ to cover infeasible points inside the convex hull.
- Every C_k is a relation from our bias.
- Examples of relations in our bias
  - $R_i = c,\ c \in \mathbb{Z}$
  - $R_i = upper\_bound(R_i, n)$
  - R_i is odd (even)
  - $R_i = R_j$
  - etc.
- Every infeasible point on/inside of the convex hull must be covered by at least one hypothesis.

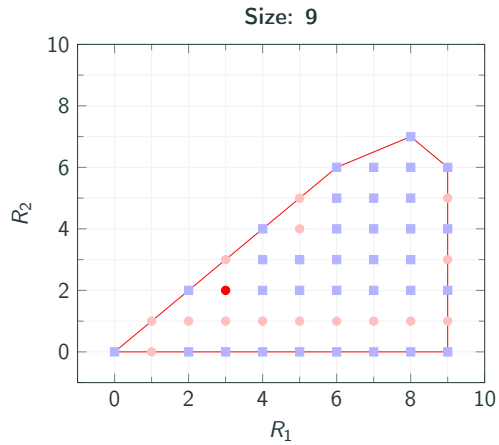
Mining Phase: Example



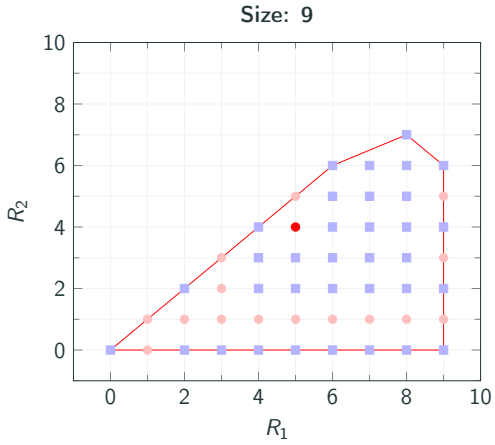
(Group ①) $R_1 = 1$



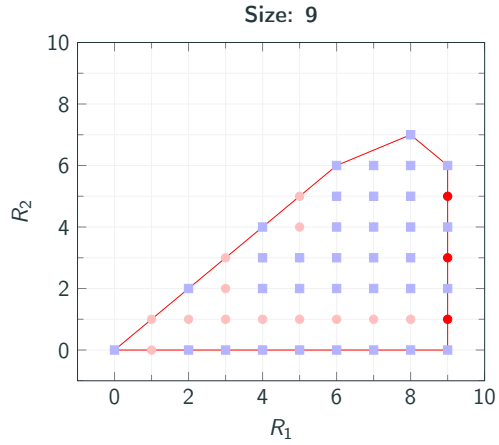
(Group ②) $R_2 = 1$



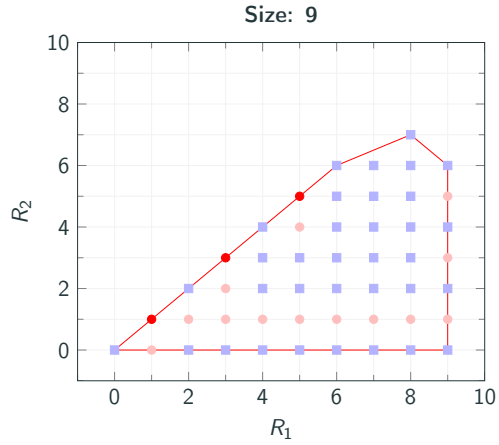
(Group ③) $R_1 = 3 \wedge R_2 = 2$



(Group ④) $R_1 = 5 \wedge R_2 = 4$



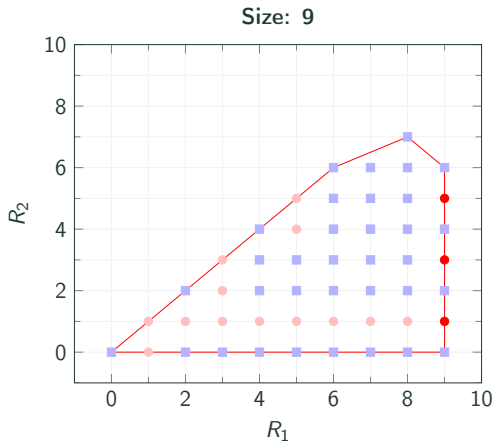
(Group ⑤) $R_1 = up(R_1, n) \wedge R_2 \bmod 2 = 1$



(Group ⑥) $R_1 = R_2 \wedge R_1 \bmod 2 = 1$

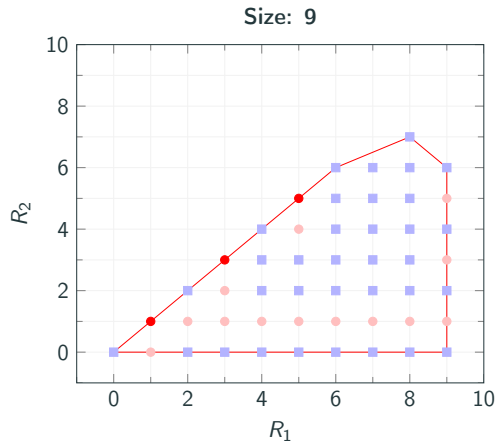
Classification of Groups of Points

1. **Independent Groups**: $H = C_1 \wedge C_2 \wedge \cdots \wedge C_p$, every C_k depends only on one R_i.
2. **Dependent Groups**: $H = C_1 \wedge C2 \wedge \cdots \wedge C_p$, at least one C_k depends on more than one R_i.



(Group ⑤) $R_1 = up(R_1, n) \wedge R_2 \bmod 2 = 1$

Independent Group



(Group ⑥) $R_1 = R_2 \wedge R_1 \bmod 2 = 1$

Dependent Group

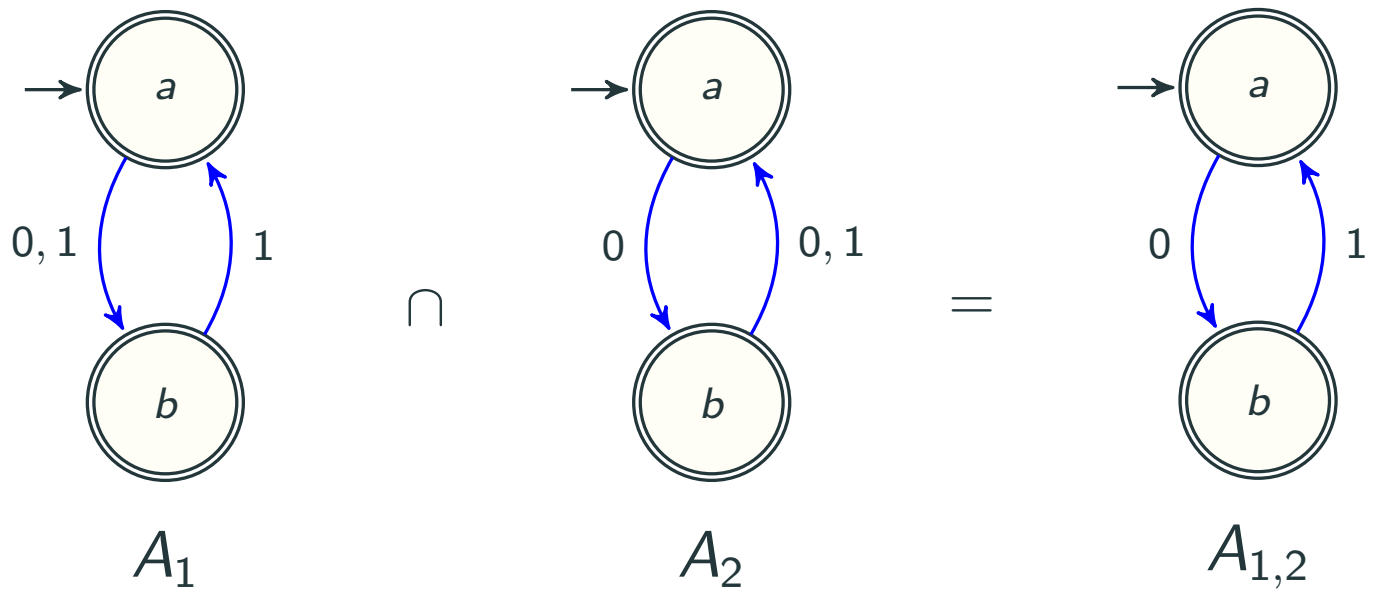
The proof scheme depends on the group type!

Proving Phase: Independent Groups

- For every hypothesis $C_1 \wedge C_2 \wedge \cdots \wedge C_p$,
  generate a **constant size automaton** for each C_i relation.
- Do the **intersection** of the automata for all $C_1, C_2, \ldots C_p$.
- The intersection is an automaton that recognises **all and only**
  sequences satisfying the conjunction $C_1 \wedge C_2 \wedge \cdots \wedge C_p$.
- If the intersection is empty,
  then $C_1 \wedge C_2 \wedge \cdots \wedge C_p$ is not feasible.
- Otherwise, a counter example violating the hypothesis.

Example: Intersection of Two Automata

- Two automata consuming sequences of '0' and '1'.
- A_1: all '0' can appear only in odd positions, e.g. $\langle 0, 1, 0, 1, 1 \rangle$.
- A_2: all '1' can appear only in even positions, e.g. $\langle 0, 1, 0, 1, 0 \rangle$.
- Intersection $A_{1,2}$ of A_1 and A_2 recognises sequences of alternating '0' and '1'.

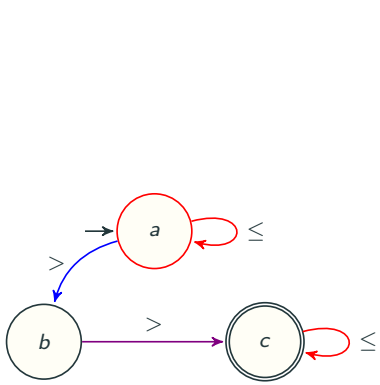


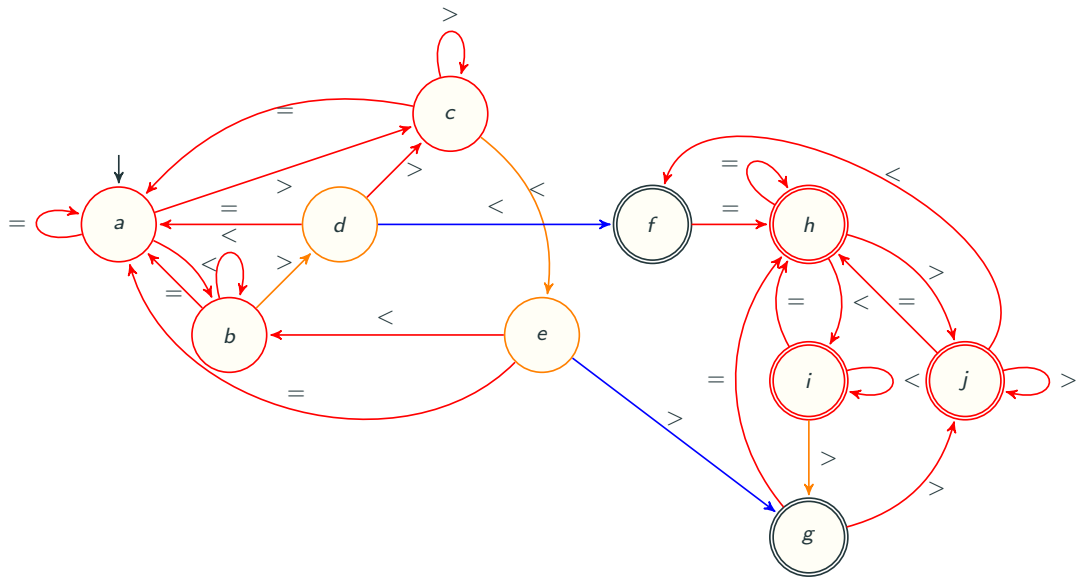
A_1 A_2 $A_{1,2}$

Proving Phase: Independent Group Example

$\text{sum\_width\_decreasing\_sequence}(X, R_1) \wedge \text{sum\_width\_zigzag}(X, R_2)$

An independent group is described by $R_1 = 3 \wedge R_2 = 2$





$R_1 = 3$ $R_2 = 2$

The intersection of two automata is **empty**!

The combination $R_1 = 3$ and $R_2 = 2$ is indeed **infeasible**.

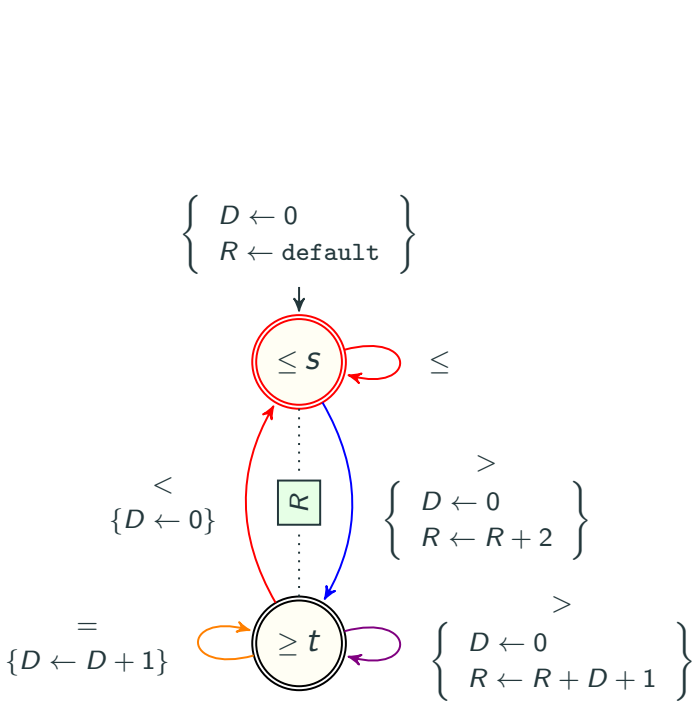
Systematic Generation of Automata for Proving Independent Groups

For two considered families of time-series constraints, we can generate systematically automata for:

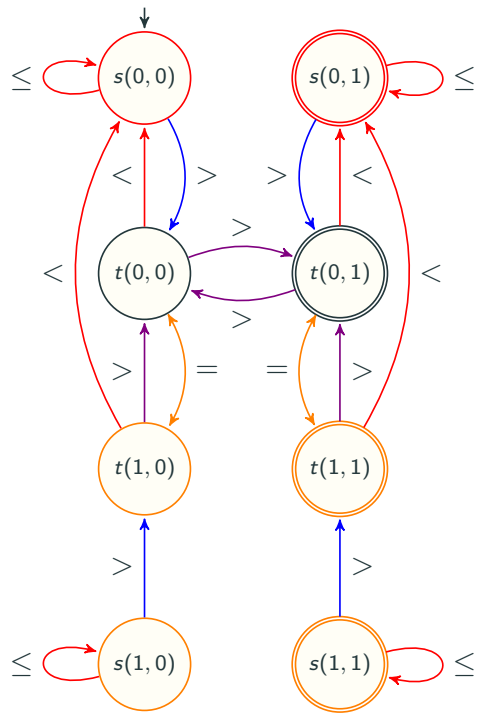
- $R_i = c$, $c \in \mathbb{Z}$,
- $R_i = up(R_i, n) - c$, $c \geq 0 \in \mathbb{Z}$, and γ_i is nb_σ,
- $R_i = up(R_i, n)$, and γ_i is sum_width_σ,
- R_i is odd/even.

Example of Automaton for the 'R is odd' Rule

sum_width_decreasing_sequence(X, R)



(a)

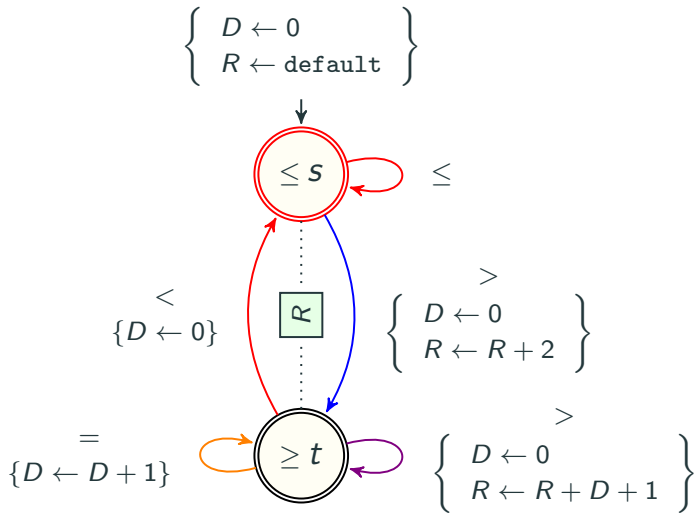


(b)

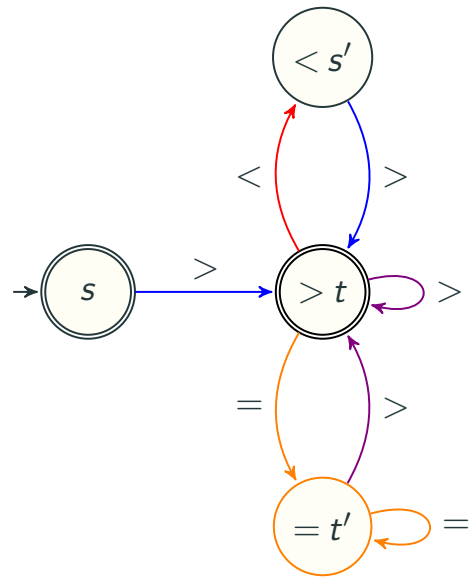
(a) Automaton for the sum_width_decreasing_sequence time-series constraint; (b) Automata for the 'R is *odd*' rule, constructed from (a)

Example of Automaton for the $R = up(R, n)$ Rule

sum_width_decreasing_sequence(X, R)



(a)



(b)

(a) Automaton for the sum_width_decreasing_sequence time-series constraint; (b) Automata for the $R = up(R, n)$ rule, constructed from (a)

Proving Phase: Dependent Groups

- Proof of dependent groups requires case by case consideration.
- The proof consists of verifying a certain property using our cut-generation technique.
- Often, this property is only a sufficient, but not necessary condition, for proving a hypothesis.

Conclusion (Take-Away Message)

- **Convex Case:** A compositional way of generating cuts from register automata. Already evaluated in [ABS17].
- **Non-Convex Case:** Data Mining + Proof using automata characterising infeasible combinations of points for conjunction of constraints. Currently evaluated.
- Our method is offline and solver/system independent.

References I

Ekaterina Arafailova, Nicolas Beldiceanu, and Helmut Simonis. "Generating Linear Invariants for a Conjunction of Automata Constraints". In: *CP 2017, Melbourne, VIC, Australia*. Vol. 10416. LNCS. Springer, 2017, pp. 21–37.

Ekaterina Arafailova et al. "Time-series constraints: Improvements and application in CP and MIP contexts". In: *CP-AI-OR 2016*. Ed. by Claude-Guy Quimper. Vol. 9676. LNCS. Springer, 2016, pp. 18–34.

References II

Nicolas Beldiceanu et al. "Using finite transducers for describing and synthesising structural time-series constraints". In: *Constraints* 21.1 (2016). Journal fast track of CP 2015: summary on p. 723 of LNCS 9255, Springer, 2015, pp. 22–40.