Active learning of Petri Net with sled languages

Work-In-Progress

Manon Le Moigne, **Rémi Parrot** and Olivier Roux

02/11/2026

LS2N
LABORATOIRE
DES SCIENCES
DU NUMÉRIQUE
DE NANTES

Outline

1. Objectives
2. Region theory
3. Semilinear sets learning
4. Conclusion

Objectives

Active learning

Algorithm in the style of Angluin L^* [1]:

- a hidden language L_h
- an oracle answers membership query: "$w \in L_h$?"

[1] D. Angluin, "Learning regular sets from queries and counterexamples," Information and Computation, vol. 75, no. 2, pp. 87–106, 1987.

Our goal

Active learning

Algorithm in the style of Angluin L^* [1]:

- a hidden language L_h
- an oracle answers membership query: "$w \in L_h$?"

Petri net language?

Many possible definitions: ε transition? labelled Petri net? fireable sequences? sequences leading to a accepting marking? sequences covering a marking? [2]

---

[1] D. Angluin, "Learning regular sets from queries and counterexamples," Information and Computation, vol. 75, no. 2, pp. 87–106, 1987.

[2] M. H. T. Hack, "Petri Net Language." Mar. 1976.

Our goal

Active learning

Algorithm in the style of Angluin L^* [1]:

- a hidden language L_h
- an oracle answers membership query: "$w \in L_h$?"

Petri net language?

Many possible definitions: ε transition? labelled Petri net? fireable sequences? sequences leading to a accepting marking? sequences covering a marking? [2]

The class of language we are interested on:

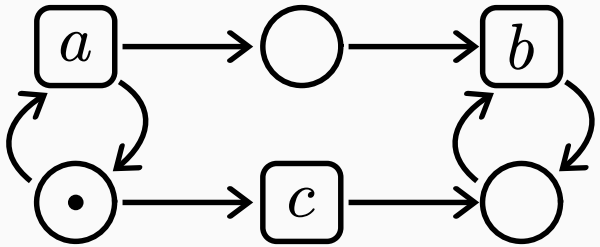
- no ε transition
- label free Petri net: one transition $=$ one letter
- words match fireable sequences of transitions $\rightarrow$ prefix closed

---

[1] D. Angluin, "Learning regular sets from queries and counterexamples," Information and Computation, vol. 75, no. 2, pp. 87–106, 1987.

[2] M. H. T. Hack, "Petri Net Language." Mar. 1976.

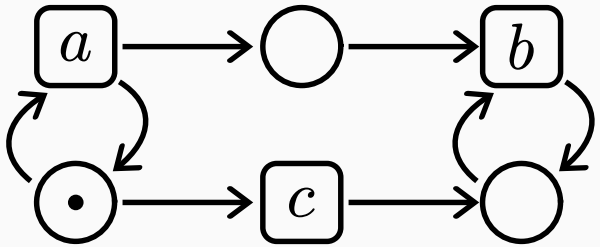
An example



We can fire:

- a as many times as we want
- then c
- then b at most as many times as a

An example



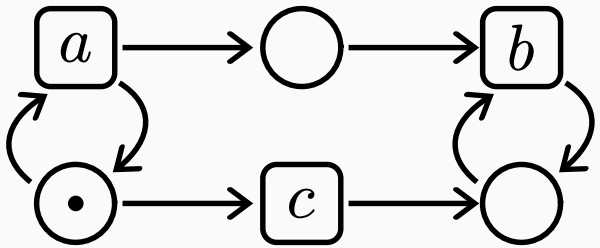
We can fire:

- a as many times as we want
- then c
- then b at most as many times as a

Language

$$L = \{a^n \mid n \in \mathbb{N}\} \cup \{a^n c b^{n-k} \mid n \in \mathbb{N}, k \in [0..n]\}$$

An example



We can fire:

- a as many times as we want
- then c
- then b at most as many times as a

Language

$$L = \{a^n \mid n \in \mathbb{N}\} \cup \{a^n c b^{n-k} \mid n \in \mathbb{N}, k \in [0..n]\}$$

Remark

Definitely not a regular language!

Region theory

Prefix-closed language

A prefix-closed language $L \subseteq E^*$ is identified by an initialized transition system (S, E, δ, s_0) .

Prefix-closed language

A prefix-closed language $L \subseteq E^*$ is identified by an initialized transition system $(S = L, E, \delta, s_0 = \varepsilon)$.

Prefix-closed language

A prefix-closed language $L \subseteq E^*$ is identified by an initialized transition system $(S = L, E, \delta, s_0 = \varepsilon)$.

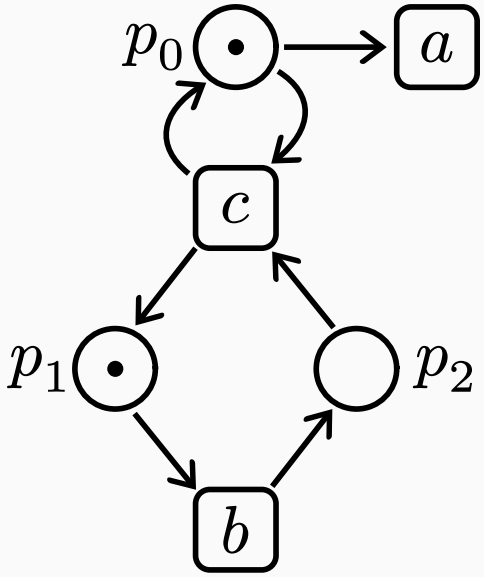
Then, for all $e \in E$, $u \in L$, $\delta(u, e)$ is defined and in S iff $ue \in L$

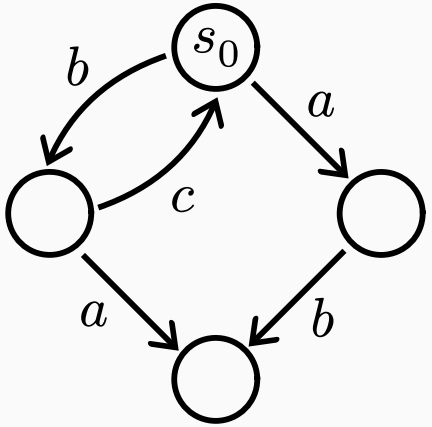
Prefix-closed language

A prefix-closed language $L \subseteq E^*$ is identified by an initialized transition system $(S = L, E, \delta, s_0 = \varepsilon)$.

Then, for all $e \in E$, $u \in L$, $\delta(u, e)$ is defined and in S iff $ue \in L$

Example:





P/T-region of L [3]

A P/T-region is a tuple $r = (r_S, (r_E{}^\circ, {}^\circ r_E))$, where:

- an id $r_S(s)$ is assigned to each state s (in $\mathbb{N}$)
- a pre $r_E{}^\circ(e)$ and post ${}^\circ r_E(e)$ is assigned to each letter e (in $\mathbb{N}$)

[3] E. Badouel, L. Bernardinello, and P. Darondeau, **Petri Net Synthesis**. in Text in Theoretical Computer Science, an EATCS Series. Springer, 2015, p. 339. doi: 10.1007/978-3-662-47967-4.

P/T-region of L [3]

A P/T-region is a tuple $r = (r_S, ({r_E}^\circ, {}^\circ r_E))$, where:

- an id $r_S(s)$ is assigned to each state s (in $\mathbb{N}$)
- a pre ${r_E}^\circ(e)$ and post ${}^\circ r_E(e)$ is assigned to each letter e (in $\mathbb{N}$)

r is a region of L iff:

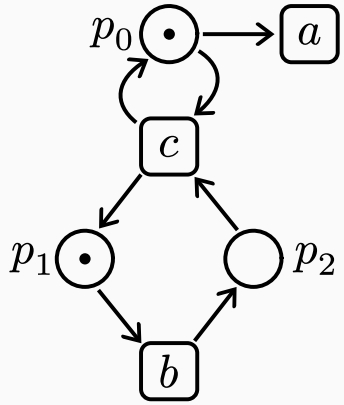
$$ue \in L \implies \begin{cases} r_S(u) \geq {r_E}^\circ(e) \\ r_S(ue) = r_S(u) - {r_E}^\circ(e) + {}^\circ r_E(e) \end{cases}$$

[3] E. Badouel, L. Bernardinello, and P. Darondeau, Petri Net Synthesis. in Text in Theoretical Computer Science, an EATCS Series. Springer, 2015, p. 339. doi: 10.1007/978-3-662-47967-4.

P/T-region of L [3] – Example

r is a region of L iff:

$$ue \in L \Longrightarrow \begin{cases} r_S(u) \geq r_E{}^\circ(e) \\ r_S(ue) = r_S(u) - r_E{}^\circ(e) + {}^\circ r_E(e) \end{cases}$$

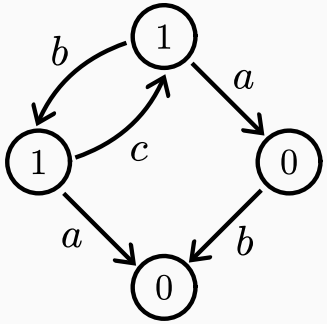


[3] E. Badouel, L. Bernardinello, and P. Darondeau, Petri Net Synthesis. in Text in Theoretical Computer Science, an EATCS Series. Springer, 2015, p. 339. doi: 10.1007/978-3-662-47967-4.

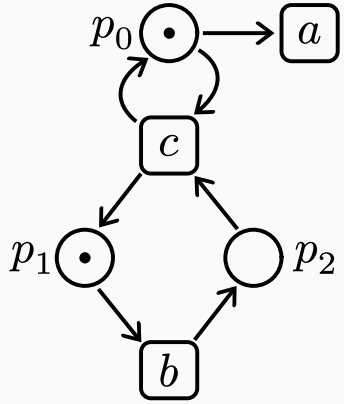
P/T-region of L [3] – Example

r is a region of L iff:

$$ue \in L \Longrightarrow \begin{cases} r_S(u) \geq {r_E}^\circ(e) \\ r_S(ue) = r_S(u) - {r_E}^\circ(e) + {}^\circ r_E(e) \end{cases}$$



$${r_E}^\circ = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, {}^\circ r_E = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

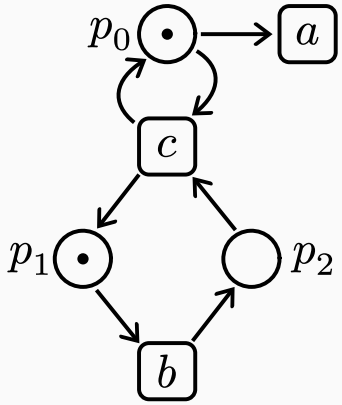


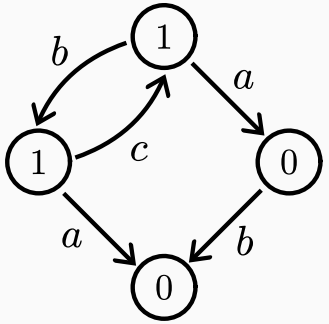
[3] E. Badouel, L. Bernardinello, and P. Darondeau, Petri Net Synthesis. in Text in Theoretical Computer Science, an EATCS Series. Springer, 2015, p. 339. doi: 10.1007/978-3-662-47967-4.

P/T-region of L [3] – Example

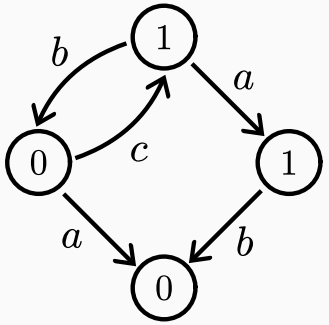
r is a region of L iff:

$$ue \in L \Longrightarrow \begin{cases} r_S(u) \geq {r_E}^\circ(e) \\ r_S(ue) = r_S(u) - {r_E}^\circ(e) + {}^\circ r_E(e) \end{cases}$$





$${r_E}^\circ = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, {}^\circ r_E = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$



$${r_E}^\circ = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, {}^\circ r_E = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

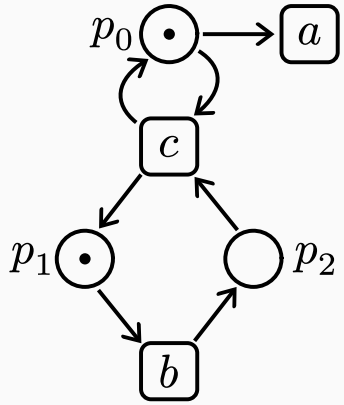
---

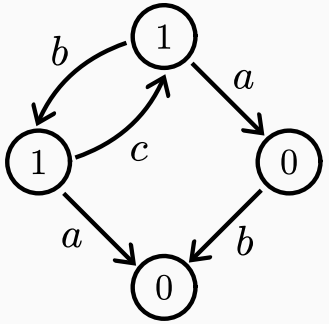
[3] E. Badouel, L. Bernardinello, and P. Darondeau, **Petri Net Synthesis**. in Text in Theoretical Computer Science, an EATCS Series. Springer, 2015, p. 339. doi: 10.1007/978-3-662-47967-4.

P/T-region of L [3] – Example

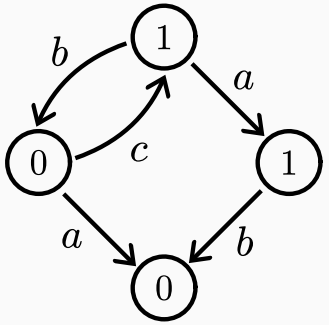
r is a region of L iff:

$$ue \in L \implies \begin{cases} r_S(u) \geq r_E{}^\circ(e) \\ r_S(ue) = r_S(u) - r_E{}^\circ(e) + {}^\circ r_E(e) \end{cases}$$

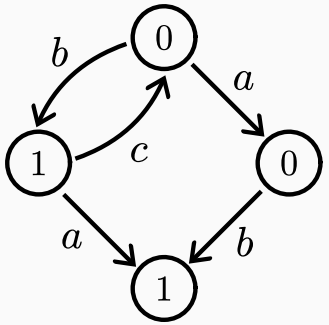




$$r_E{}^\circ = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, {}^\circ r_E = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$



$$r_E{}^\circ = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, {}^\circ r_E = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$



$$r_E{}^\circ = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, {}^\circ r_E = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

[3] E. Badouel, L. Bernardinello, and P. Darondeau, Petri Net Synthesis. in Text in Theoretical Computer Science, an EATCS Series. Springer, 2015, p. 339. doi: 10.1007/978-3-662-47967-4.

Parikh vector

Let $u \in E^*$. $\psi(u)$, the Parikh vector of u, is the vector of **occurrences** of each letter of E in u.
Let $E = \{e_1, ...e_n\}$, then $\psi(u)(i)$ is the number of occurrences of e_i in u.

How to determine the regions?

Parikh vector

Let $u \in E^*$. $\psi(u)$, the Parikh vector of u, is the vector of **occurrences** of each letter of E in u.
Let $E = \{e_1, ...e_n\}$, then $\psi(u)(i)$ is the number of occurrences of e_i in u.

Regions of L

Notice that $r_S(u) = r_S(\varepsilon) + \psi(u).(^\circ r_E - r_E{}^\circ)$

r is a region of L iff:

$ue \in L \Rightarrow r_S(\varepsilon) + \psi(u).(^\circ r_E - r_E{}^\circ) \geq r_E{}^\circ(e)$

How to determine the regions?

Parikh vector

Let $u \in E^*$. $\psi(u)$, the Parikh vector of u, is the vector of **occurrences** of each letter of E in u.
Let $E = \{e_1, ...e_n\}$, then $\psi(u)(i)$ is the number of occurrences of e_i in u.

Regions of L

Notice that $r_S(u) = r_S(\varepsilon) + \psi(u).(^\circ r_E - r_E{}^\circ)$

r is a region of L iff:

$ue \in L \Rightarrow r_S(\varepsilon) + \psi(u).(^\circ r_E - r_E{}^\circ) \geq r_E{}^\circ(e)$

Convex cone

We can rewrite as:
$x_0 + \sum_{i=1}^{n} \psi(u)(i).(x_{n+i} - x_i) - x_j \geq 0$ with $x_i \in \mathbb{N}$

↳ **Convex cone of** $\mathbb{N}^{2n+1}$

How to determine the regions?

Parikh vector

Let $u \in E^*$. $\psi(u)$, the Parikh vector of u, is the vector of **occurrences** of each letter of E in u.
Let $E = \{e_1, ...e_n\}$, then $\psi(u)(i)$ is the number of occurrences of e_i in u.

Regions of L

Notice that $r_S(u) = r_S(\varepsilon) + \psi(u).(^\circ r_E - r_E{}^\circ)$

r is a region of L iff:

$ue \in L \Rightarrow r_S(\varepsilon) + \psi(u).(^\circ r_E - r_E{}^\circ) \geq r_E{}^\circ(e)$

Convex cone

We can rewrite as:
$x_0 + \sum_{i=1}^{n} \psi(u)(i).(x_{n+i} - x_i) - x_j \geq 0$ with $x_i \in \mathbb{N}$

↳ **Convex cone of** $\mathbb{N}^{2n+1}$

Problem

There are possibly infinitely many such inequations → impossible to determine the regions from these

Enablings and Disablings

Let $E = \{e_1, ...e_n\}$ and $L \subseteq E^*$ a prefix-closed language.

-
-

Enablings and Disablings

Let $E = \{e_1, ...e_n\}$ and $L \subseteq E^*$ a prefix-closed language.

- **Enablings** of L are: $L_{e_j} = L|e_j = \{u \mid ue_j \in L\}$
-

Enablings and Disablings

Let $E = \{e_1, ...e_n\}$ and $L \subseteq E^*$ a prefix-closed language.

- **Enablings** of L are: $L_{e_j} = L|e_j = \{u \mid ue_j \in L\}$
- **Disablings** of L are: $L_{\bar{e_j}} = L \setminus L|e_j$

Enablings and Disablings

Let $E = \{e_1, ...e_n\}$ and $L \subseteq E^*$ a prefix-closed language.

- **Enablings** of L are: $L_{e_j} = L|e_j = \{u \mid ue_j \in L\}$
- **Disablings** of L are: $L_{\bar{e_j}} = L \setminus L|e_j$

Semilinear language

Parikh image of a language: $\Psi(L) = \{\psi(u) \mid u \in L\}$. A language L is **semilinear** if its Parikh image is a semilinear set.

Sled languages

Enablings and Disablings

Let $E = \{e_1, ...e_n\}$ and $L \subseteq E^*$ a prefix-closed language.

- **Enablings** of L are: $L_{e_j} = L|e_j = \{u \mid ue_j \in L\}$
- **Disablings** of L are: $L_{\bar{e_j}} = L \setminus L|e_j$

Semilinear language

Parikh image of a language: $\Psi(L) = \{\psi(u) \mid u \in L\}$. A language L is **semilinear** if its Parikh image is a semilinear set.

Sled language

A sled language is a **prefix-closed** language with **semilinear enablings** and **disablings**.

Regions of a sled language

Intuition

1. $\forall e_j$, $\Psi\left(L_{e_j}\right)$ is a semilinear set, i.e. a finite union of linear set: $\psi_0.\{\psi_1, ..., \psi_k\}^*$
2.

Regions of a sled language

Intuition

1. $\forall e_j, \Psi\left(L_{e_j}\right)$ is a semilinear set, i.e. a finite union of linear set: $\psi_0.\{\psi_1, ..., \psi_k\}^*$
2. $ue_j \in L \Leftrightarrow u \in L_{e_j}$

Regions of a sled language

Intuition

1. $\forall e_j, \Psi\left(L_{e_j}\right)$ is a semilinear set, i.e. a finite union of linear set: $\psi_0.\{\psi_1, ..., \psi_k\}^*$
2. $ue_j \in L \Leftrightarrow u \in L_{e_j}$

We can reduce the problem to a finite linear system of the shape:

$$\begin{cases} x_0 + \sum_{i=1}^{n} \psi_0(i).(x_{n+i} - x_i) - x_{n+j} \geq 0 \\ \forall h \in [1..k], \sum_{i=1}^{n} \psi_h(i).(x_{n+i} - x_i) \geq 0 \end{cases}$$

Regions of a sled language

Intuition

1. $\forall e_j, \Psi\left(L_{e_j}\right)$ is a semilinear set, i.e. a finite union of linear set: $\psi_0.\{\psi_1, ..., \psi_k\}^*$
2. $ue_j \in L \Leftrightarrow u \in L_{e_j}$

We can reduce the problem to a finite linear system of the shape:

$$\begin{cases} x_0 + \sum_{i=1}^{n} \psi_0(i).(x_{n+i} - x_i) - x_{n+j} \geq 0 \\ \forall h \in [1..k], \sum_{i=1}^{n} \psi_h(i).(x_{n+i} - x_i) \geq 0 \end{cases}$$

↳ Regions of a sled language form a **polyhedral cone**

Regions of a sled language

Intuition

1. $\forall e_j, \Psi\left(L_{e_j}\right)$ is a semilinear set, i.e. a finite union of linear set: $\psi_0.\{\psi_1, ..., \psi_k\}^*$
2. $ue_j \in L \Leftrightarrow u \in L_{e_j}$

We can reduce the problem to a finite linear system of the shape:

$$\begin{cases} x_0 + \sum_{i=1}^{n} \psi_0(i).(x_{n+i} - x_i) - x_{n+j} \geq 0 \\ \forall h \in [1..k], \sum_{i=1}^{n} \psi_h(i).(x_{n+i} - x_i) \geq 0 \end{cases}$$

↳ Regions of a sled language form a **polyhedral cone**

Result[3]

It is decidable whether a **sled language** can be realized by a **Petri net**.

---

[3] E. Badouel, L. Bernardinello, and P. Darondeau, **Petri Net Synthesis**. in Text in Theoretical Computer Science, an EATCS Series. Springer, 2015, p. 339. doi: 10.1007/978-3-662-47967-4.

Regions of a sled language

Intuition

1. $\forall e_j, \Psi\left(L_{e_j}\right)$ is a semilinear set, i.e. a finite union of linear set: $\psi_0.\{\psi_1, ..., \psi_k\}^*$
2. $ue_j \in L \Leftrightarrow u \in L_{e_j}$

We can reduce the problem to a finite linear system of the shape:

$$\begin{cases} x_0 + \sum_{i=1}^{n} \psi_0(i).(x_{n+i} - x_i) - x_{n+j} \geq 0 \\ \forall h \in [1..k], \sum_{i=1}^{n} \psi_h(i).(x_{n+i} - x_i) \geq 0 \end{cases}$$

↳ Regions of a sled language form a **polyhedral cone**

Result[3]

It is decidable whether a **sled language** can be realized by a **Petri net**.

This procedure produces a Petri net realizing the language (if it exists).

---

[3] E. Badouel, L. Bernardinello, and P. Darondeau, Petri Net Synthesis. in Text in Theoretical Computer Science, an EATCS Series. Springer, 2015, p. 339. doi: 10.1007/978-3-662-47967-4.

Our contribution – active learning of sled languages

Our contribution – active learning of sled languages

First Idea

1. We can **learn the enablings and disablings** with an active learning technique for regular languages (like L^*).
2.

Our contribution – active learning of sled languages

First Idea

1. We can **learn the enablings and disablings** with an active learning technique for regular languages (like L^*).
2. Since regular languages are semilinear, we can decide if the corresponding language can be realized by a Petri net (and construct one).

Our contribution – active learning of sled languages

First Idea

1. We can **learn the enablings and disablings** with an active learning technique for regular languages (like L^*).
2. Since regular languages are semilinear, we can decide if the corresponding language can be realized by a Petri net (and construct one).

Limit

If the enablings and disablings are regular, isn't the language also regular?

Our contribution – active learning of sled languages

First Idea

1. We can **learn the enablings and disablings** with an active learning technique for regular languages (like L^*).
2. Since regular languages are semilinear, we can decide if the corresponding language can be realized by a Petri net (and construct one).

Limit

If the enablings and disablings are regular, isn't the language also regular?

Second idea

1. The procedure only require the **Parikh images of the enablings and disablings**.
2.

Our contribution – active learning of sled languages

First Idea

1. We can **learn the enablings and disablings** with an active learning technique for regular languages (like L^*).
2. Since regular languages are semilinear, we can decide if the corresponding language can be realized by a Petri net (and construct one).

Limit

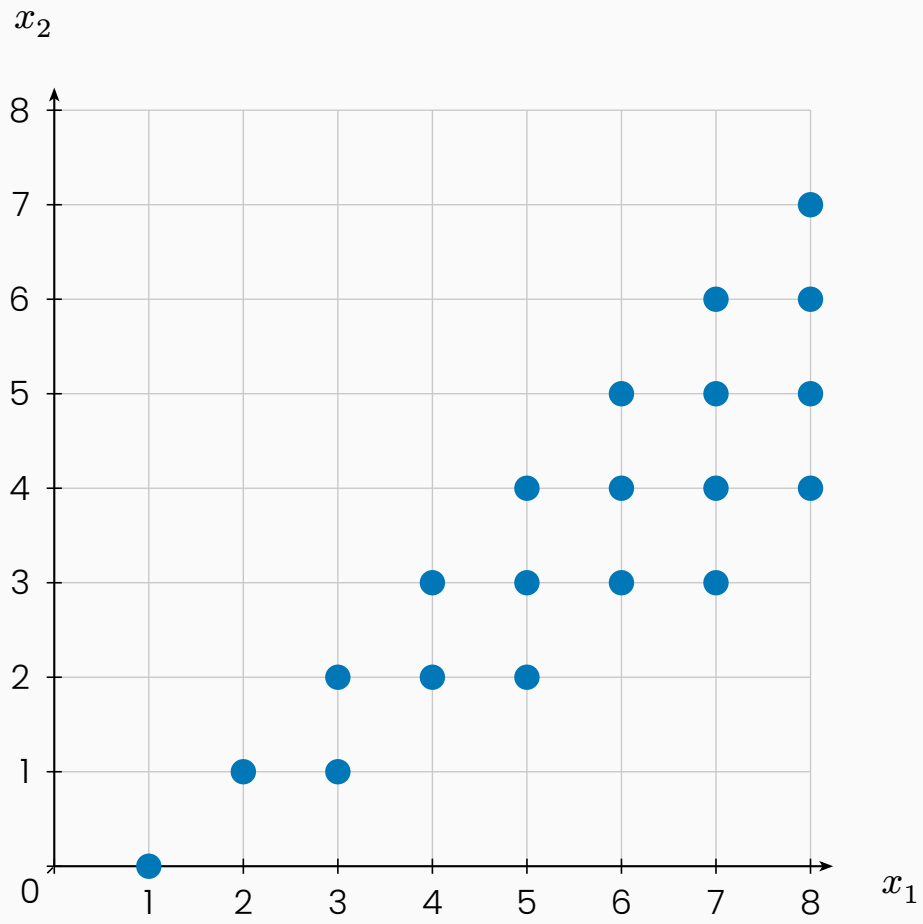
If the enablings and disablings are regular, isn't the language also regular?

Second idea

1. The procedure only require the **Parikh images of the enablings and disablings**.
2. If they are semilinear sets, we need an active learning algorithm for semilinear sets !

Semilinear sets learning

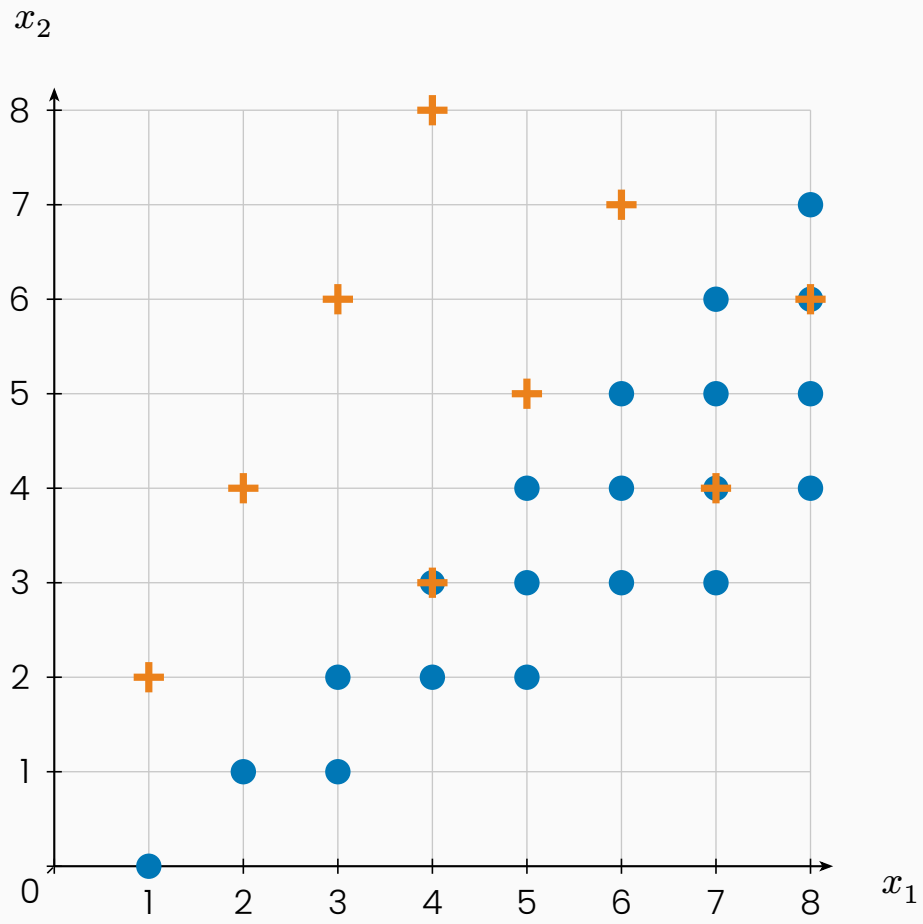
Semilinear sets – An example



A linear set of $\mathbb{N}^2$:

$$\Psi = \begin{pmatrix}1\\0\end{pmatrix}.\left\{\begin{pmatrix}1\\1\end{pmatrix},\begin{pmatrix}2\\1\end{pmatrix}\right\}^*$$

Semilinear sets – An example



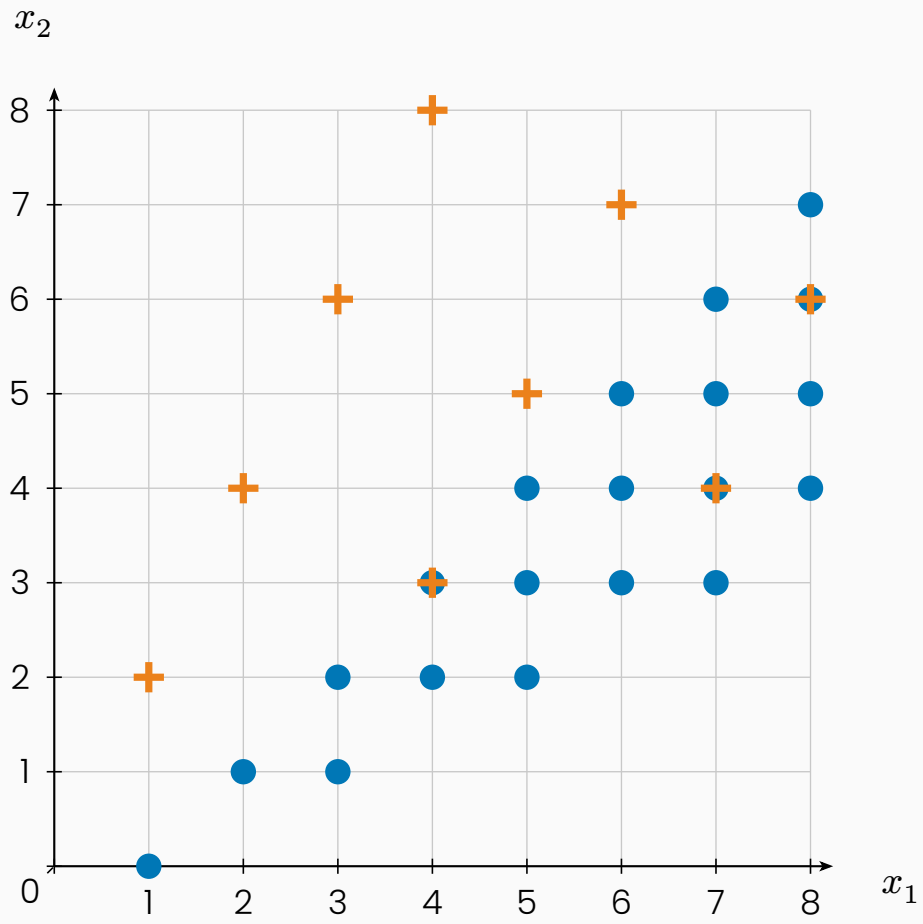
A linear set of $\mathbb{N}^2$:

$$\Psi = \begin{pmatrix}1\\0\end{pmatrix}.\left\{\begin{pmatrix}1\\1\end{pmatrix},\begin{pmatrix}2\\1\end{pmatrix}\right\}^*$$

Another linear set of $\mathbb{N}^2$:

$$\Psi = \begin{pmatrix}1\\2\end{pmatrix}.\left\{\begin{pmatrix}1\\2\end{pmatrix},\begin{pmatrix}3\\1\end{pmatrix}\right\}^*$$

Semilinear sets – An example



A linear set of $\mathbb{N}^2$:

$$\Psi = \binom{1}{0}.\left\{\binom{1}{1}, \binom{2}{1}\right\}^*$$

Another linear set of $\mathbb{N}^2$:

$$\Psi = \binom{1}{2}.\left\{\binom{1}{2}, \binom{3}{1}\right\}^*$$

A semilinear set of $\mathbb{N}^2$:

$$\Psi = \Psi_1 \cup \Psi_2$$

Semilinear sets

A **linear set** Ψ is a set of vectors of natural numbers defined by $\Psi = \left\{ b + \sum_{i=1}^{m} k_i p_i \mid k_1, .., k_m \in \mathbb{N} \right\}$ where b is the **base vector** and $P = \{p_1, .., p_m\}$ the set of **period vectors**. It is noted $\Psi = b.P^*$. A **semilinear set** is a finite union of linear sets.

-
-
-

Semilinear sets

A **linear set** Ψ is a set of vectors of natural numbers defined by $\Psi = \left\{ b + \sum_{i=1}^{m} k_i p_i \mid k_1, .., k_m \in \mathbb{N} \right\}$ where b is the **base vector** and $P = \{p_1, .., p_m\}$ the set of **period vectors**. It is noted $\Psi = b.P^*$. A **semilinear set** is a finite union of linear sets.

Active learning of Semilinear sets

It already exists: a work of Yuji Takada in 1992 [4] !

-
-
-

[4] Y. Takada, "Learning Semilinear Sets from Examples and via Queries," Theor. Comput. Sci., vol. 104, pp. 207–233, 1992.

Semilinear sets

A **linear set** Ψ is a set of vectors of natural numbers defined by $\Psi = \left\{ b + \sum_{i=1}^{m} k_i p_i \mid k_1, .., k_m \in \mathbb{N} \right\}$ where b is the **base vector** and $P = \{p_1, .., p_m\}$ the set of **period vectors**. It is noted $\Psi = b.P^*$. A **semilinear set** is a finite union of linear sets.

Active learning of Semilinear sets

It already exists: a work of Yuji Takada in 1992 [4] !

- An unknown semilinear set Ψ_u
-
-

[4] Y. Takada, "Learning Semilinear Sets from Examples and via Queries," Theor. Comput. Sci., vol. 104, pp. 207–233, 1992.

Semilinear sets

A **linear set** Ψ is a set of vectors of natural numbers defined by $\Psi = \left\{ b + \sum_{i=1}^{m} k_i p_i \mid k_1, .., k_m \in \mathbb{N} \right\}$ where b is the **base vector** and $P = \{p_1, .., p_m\}$ the set of **period vectors**. It is noted $\Psi = b.P^*$. A **semilinear set** is a finite union of linear sets.

Active learning of Semilinear sets

It already exists: a work of Yuji Takada in 1992 [4] !

- An unknown semilinear set Ψ_u
- Goal: Learn a **representation** (bases and periods) of a semilinear set Ψ, such that $\Psi = \Psi_u$
-

[4] Y. Takada, "Learning Semilinear Sets from Examples and via Queries," Theor. Comput. Sci., vol. 104, pp. 207–233, 1992.

Semilinear sets

A **linear set** Ψ is a set of vectors of natural numbers defined by $\Psi = \left\{ b + \sum_{i=1}^{m} k_i p_i \mid k_1, .., k_m \in \mathbb{N} \right\}$ where b is the **base vector** and $P = \{p_1, .., p_m\}$ the set of **period vectors**. It is noted $\Psi = b.P^*$. A **semilinear set** is a finite union of linear sets.

Active learning of Semilinear sets

It already exists: a work of Yuji Takada in 1992 [4] !

- An unknown semilinear set Ψ_u
- Goal: Learn a **representation** (bases and periods) of a semilinear set Ψ, such that $\Psi = \Psi_u$
- Oracle abilities: inclusion queries $\Psi \subseteq \Psi_u$? $\Psi_u \subseteq \Psi$?

---

[4] Y. Takada, "Learning Semilinear Sets from Examples and via Queries," Theor. Comput. Sci., vol. 104, pp. 207–233, 1992.

Takada's Semilinear sets learning algorithm

Procedure FP

- $\mathrm{FP}(\Psi)$ return the minima of $\Psi_u \setminus \Psi$
-
-

[4] Y. Takada, "Learning Semilinear Sets from Examples and via Queries," Theor. Comput. Sci., vol. 104, pp. 207–233, 1992.

Procedure FP

- $\mathrm{FP}(\Psi)$ return the minima of $\Psi_u \setminus \Psi$
- How? It uses a fix-point to find the biggest mask such that $\Psi_u \subseteq (\Psi \cup \mathrm{mask})$
-

[4] Y. Takada, "Learning Semilinear Sets from Examples and via Queries," Theor. Comput. Sci., vol. 104, pp. 207–233, 1992.

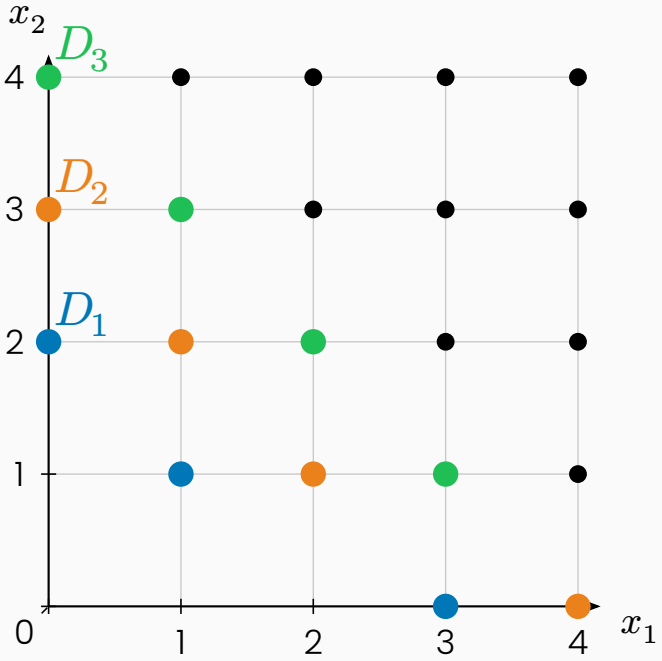
Procedure FP

- $\mathrm{FP}(\Psi)$ return the minima of $\Psi_u \setminus \Psi$
- How? It uses a fix-point to find the biggest mask such that $\Psi_u \subseteq (\Psi \cup \mathrm{mask})$
- What for? The purpose is to find **slices** of Ψ_u

[4] Y. Takada, "Learning Semilinear Sets from Examples and via Queries," Theor. Comput. Sci., vol. 104, pp. 207–233, 1992.

Procedure FP

- $\mathrm{FP}(\Psi)$ return the minima of $\Psi_u \setminus \Psi$
- How? It uses a fix-point to find the biggest mask such that $\Psi_u \subseteq (\Psi \cup \mathrm{mask})$
- What for? The purpose is to find **slices** of Ψ_u



Example: Slices of a semilinear set

[4] Y. Takada, "Learning Semilinear Sets from Examples and via Queries," Theor. Comput. Sci., vol. 104, pp. 207–233, 1992.

Takada's Semilinear sets learning algorithm

Procedure ID

ID() return a representation of Ψ_u

-
-
-
-
-
-
-

$W \leftarrow \varnothing$; $i \leftarrow 0$; $\forall j, D_j \leftarrow \varnothing$
while $\Psi_u \not\subseteq W$
  $D_i \leftarrow \mathrm{FP}\left(\bigcup_{j=1}^{i-1} D_j\right)$; $i \leftarrow i+1$
  for $d \in D_i$ **do**
    for $\Psi_i = b_i.P_i^* \in W$
      if $b_i < d$ **and** $b_i.(P_i \cup \{d - b_i\})^* \subseteq \Psi_u$ **then**
        $W \leftarrow W \cup b_i.(P_i \cup \{d - b_i\})^*$
      end if
    end for
    $W \leftarrow W \cup d.\varnothing^*$
  end for
end while

[4] Y. Takada, "Learning Semilinear Sets from Examples and via Queries," Theor. Comput. Sci., vol. 104, pp. 207–233, 1992.

Procedure ID

ID() return a representation of Ψ_u

- W will contain the result
-
-
-
-
-

$W \leftarrow \varnothing$; $i \leftarrow 0$; $\forall j, D_j \leftarrow \varnothing$
while $\Psi_u \not\subseteq W$
  $D_i \leftarrow \mathrm{FP}\left(\bigcup_{j=1}^{i-1} D_j\right)$; $i \leftarrow i+1$
  for $d \in D_i$ **do**
    for $\Psi_i = b_i.P_i^* \in W$
      if $b_i < d$ **and** $b_i.(P_i \cup \{d - b_i\})^* \subseteq \Psi_u$ **then**
        $W \leftarrow W \cup b_i.(P_i \cup \{d - b_i\})^*$
      end if
    end for
    $W \leftarrow W \cup d.\varnothing^*$
  end for
end while

[4] Y. Takada, "Learning Semilinear Sets from Examples and via Queries," Theor. Comput. Sci., vol. 104, pp. 207–233, 1992.

Procedure ID

ID() return a representation of Ψ_u

- W will contain the result
- by construction $W \subseteq \Psi_u$
-
-
-
-
-

$W \leftarrow \varnothing$; $i \leftarrow 0$; $\forall j, D_j \leftarrow \varnothing$
while $\Psi_u \not\subseteq W$
  $D_i \leftarrow \mathrm{FP}\left(\bigcup_{j=1}^{i-1} D_j\right)$; $i \leftarrow i+1$
  for $d \in D_i$ **do**
    for $\Psi_i = b_i.P_i^* \in W$
      if $b_i < d$ **and** $b_i.(P_i \cup \{d - b_i\})^* \subseteq \Psi_u$ **then**
        $W \leftarrow W \cup b_i.(P_i \cup \{d - b_i\})^*$
      end if
    end for
    $W \leftarrow W \cup d.\varnothing^*$
  end for
end while

[4] Y. Takada, "Learning Semilinear Sets from Examples and via Queries," Theor. Comput. Sci., vol. 104, pp. 207–233, 1992.

Takada's Semilinear sets learning algorithm

Procedure ID

ID() return a representation of Ψ_u

- W will contain the result
- by construction $W \subseteq \Psi_u$
- take next slice D_i
-
-
-
-

$W \leftarrow \varnothing$; $i \leftarrow 0$; $\forall j, D_j \leftarrow \varnothing$
while $\Psi_u \not\subseteq W$
  $D_i \leftarrow \mathrm{FP}\left(\bigcup_{j=1}^{i-1} D_j\right)$; $i \leftarrow i+1$
  for $d \in D_i$ **do**
    for $\Psi_i = b_i.P_i^* \in W$
      if $b_i < d$ **and** $b_i.(P_i \cup \{d - b_i\})^* \subseteq \Psi_u$ **then**
        $W \leftarrow W \cup b_i.(P_i \cup \{d - b_i\})^*$
      end if
    end for
    $W \leftarrow W \cup d.\varnothing^*$
  end for
end while

[4] Y. Takada, "Learning Semilinear Sets from Examples and via Queries," Theor. Comput. Sci., vol. 104, pp. 207–233, 1992.

Takada's Semilinear sets learning algorithm

Procedure ID

ID() return a representation of Ψ_u

- W will contain the result
- by construction $W \subseteq \Psi_u$
- take next slice D_i
- for all the vectors d of the slice
-
-
-

$W \leftarrow \varnothing$; $i \leftarrow 0$; $\forall j, D_j \leftarrow \varnothing$
while $\Psi_u \not\subseteq W$
  $D_i \leftarrow \mathrm{FP}\left(\bigcup_{j=1}^{i-1} D_j\right)$; $i \leftarrow i+1$
  for $d \in D_i$ **do**
    for $\Psi_i = b_i.P_i^* \in W$
      if $b_i < d$ **and** $b_i.(P_i \cup \{d - b_i\})^* \subseteq \Psi_u$ **then**
        $W \leftarrow W \cup b_i.(P_i \cup \{d - b_i\})^*$
      end if
    end for
    $W \leftarrow W \cup d.\varnothing^*$
  end for
end while

[4] Y. Takada, "Learning Semilinear Sets from Examples and via Queries," Theor. Comput. Sci., vol. 104, pp. 207–233, 1992.

Procedure ID

ID() return a representation of Ψ_u

- W will contain the result
- by construction $W \subseteq \Psi_u$
- take next slice D_i
- for all the vectors d of the slice
- add a new linear set with base vector d
-
-

$W \leftarrow \varnothing$; $i \leftarrow 0$; $\forall j, D_j \leftarrow \varnothing$
while $\Psi_u \not\subseteq W$
  $D_i \leftarrow \mathrm{FP}\left(\bigcup_{j=1}^{i-1} D_j\right)$; $i \leftarrow i+1$
  for $d \in D_i$ **do**
    for $\Psi_i = b_i.P_i^* \in W$
      if $b_i < d$ **and** $b_i.(P_i \cup \{d - b_i\})^* \subseteq \Psi_u$ **then**
        $W \leftarrow W \cup b_i.(P_i \cup \{d - b_i\})^*$
      end if
    end for
    $W \leftarrow W \cup d.\varnothing^*$
  end for
end while

[4] Y. Takada, "Learning Semilinear Sets from Examples and via Queries," Theor. Comput. Sci., vol. 104, pp. 207–233, 1992.

Procedure ID

ID() return a representation of Ψ_u

- W will contain the result
- by construction $W \subseteq \Psi_u$
- take next slice D_i
- for all the vectors d of the slice
- add a new linear set with base vector d
- for all the linear sets Ψ_i of W
-

$W \leftarrow \varnothing$; $i \leftarrow 0$; $\forall j, D_j \leftarrow \varnothing$
while $\Psi_u \not\subseteq W$
  $D_i \leftarrow \mathrm{FP}\left(\bigcup_{j=1}^{i-1} D_j\right)$; $i \leftarrow i+1$
  for $d \in D_i$ **do**
    for $\Psi_i = b_i.P_i^* \in W$
      if $b_i < d$ **and** $b_i.(P_i \cup \{d - b_i\})^* \subseteq \Psi_u$ **then**
        $W \leftarrow W \cup b_i.(P_i \cup \{d - b_i\})^*$
      end if
    end for
    $W \leftarrow W \cup d.\varnothing^*$
  end for
end while

[4] Y. Takada, "Learning Semilinear Sets from Examples and via Queries," Theor. Comput. Sci., vol. 104, pp. 207–233, 1992.

Procedure ID

ID() return a representation of Ψ_u

- W will contain the result
- by construction $W \subseteq \Psi_u$
- take next slice D_i
- for all the vectors d of the slice
- add a new linear set with base vector d
- for all the linear sets Ψ_i of W
- if possible add the new period vectors

$W \leftarrow \varnothing$; $i \leftarrow 0$; $\forall j, D_j \leftarrow \varnothing$
while $\Psi_u \not\subseteq W$
  $D_i \leftarrow \mathrm{FP}\left(\bigcup_{j=1}^{i-1} D_j\right)$; $i \leftarrow i+1$
  for $d \in D_i$ **do**
    for $\Psi_i = b_i.P_i^* \in W$
      if $b_i < d$ **and** $b_i.(P_i \cup \{d - b_i\})^* \subseteq \Psi_u$ **then**
        $W \leftarrow W \cup b_i.(P_i \cup \{d - b_i\})^*$
      end if
    end for
    $W \leftarrow W \cup d.\varnothing^*$
  end for
end while

[4] Y. Takada, "Learning Semilinear Sets from Examples and via Queries," Theor. Comput. Sci., vol. 104, pp. 207–233, 1992.

Procedure ID

ID() return a representation of Ψ_u

- W will contain the result
- by construction $W \subseteq \Psi_u$
- take next slice D_i
- for all the vectors d of the slice
- add a new linear set with base vector d
- for all the linear sets Ψ_i of W
- if possible add the new period vectors

$W \leftarrow \varnothing$; $i \leftarrow 0$; $\forall j, D_j \leftarrow \varnothing$
while $\Psi_u \not\subseteq W$
  $D_i \leftarrow \mathrm{FP}\left(\bigcup_{j=1}^{i-1} D_j\right)$; $i \leftarrow i+1$
  for $d \in D_i$ **do**
    for $\Psi_i = b_i.P_i^* \in W$
      if $b_i < d$ **and** $b_i.(P_i \cup \{d - b_i\})^* \subseteq \Psi_u$ **then**
        $W \leftarrow W \cup b_i.(P_i \cup \{d - b_i\})^*$
      end if
    end for
    $W \leftarrow W \cup d.\varnothing^*$
  end for
end while

[4] Y. Takada, "Learning Semilinear Sets from Examples and via Queries," Theor. Comput. Sci., vol. 104, pp. 207–233, 1992.

Conclusion

Conclusion

We plug together two old results:

1. Petri net synthesis from the Parikh images of enablings and disablings
2. Active learning of semilinear sets

We think that it leads to an active learning algorithm of Petri nets (with sled language)

What remains?

Implementation

An algorithm for the inclusion of semilinear sets?

What remains?

Implementation

An algorithm for the inclusion of semilinear sets?

Theory

- How to do **inclusion queries** of semilinear sets (in both ways) in practice?
-

What remains?

Implementation

An algorithm for the inclusion of semilinear sets?

Theory

- How to do **inclusion queries** of semilinear sets (in both ways) in practice?
- Can we have **probabilistic guarantee** is the same fashion as L^*?
  Using only membership queries, can we guarantee that the probability that a word exists in the symmetrical difference of the languages is lower than some ε.

Bibliography

[1] D. Angluin, “Learning regular sets from queries and counterexamples,” **Information and Computation**, vol. 75, no. 2, pp. 87–106, 1987.

[2] M. H. T. Hack, “Petri Net Language.” Mar. 1976.

[3] E. Badouel, L. Bernardinello, and P. Darondeau, **Petri Net Synthesis**. in Text in Theoretical Computer Science, an EATCS Series. Springer, 2015, p. 339. doi: 10.1007/978-3-662-47967-4.

[4] Y. Takada, “Learning Semilinear Sets from Examples and via Queries,” **Theor. Comput. Sci.**, vol. 104, pp. 207–233, 1992.