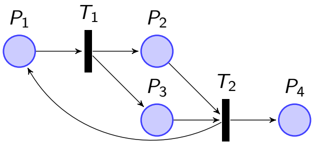
Linear algebra + Petri nets

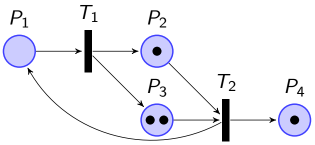
Piotr Hofman
University of Warsaw

Petri Nets.



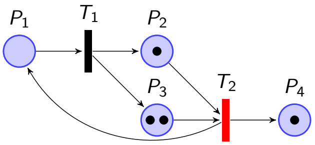
- Places.
- Transitions.

Petri Nets.



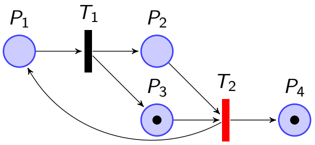
- Places.
- Transitions.
- Tokens, a Marking.

Petri Nets.



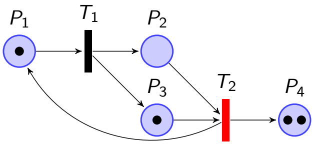
- Places.
- Transitions.
- Tokens, a Marking.
- Firing a transition.

Petri Nets.



- Places.
- Transitions.
- Tokens, a Marking.
- Firing a transition.

Petri Nets.



- Places.
- Transitions.
- Tokens, a Marking.
- Firing a transition.

Questions and tools.

We focus on analysis of systems modelled with Petri nets.

Most important questions:

1. Place coverability,
2. Reachability,
3. Liveness,
4. Death of a transition,
5. Deadlock-freeness.

Most important tools:

1. Coverability: ExpSpace complete,
2. Boundedness: ExpSpace complete,
3. Reachability: at least ExpSpace Hard.

Two solutions:

Do not try to be precise (approximations).

1. Place invariant.
2. State equation.
3. Continuous reachability.
4. Traps and siphons.

Do not try to be general (sub-classes).

1. Free-choice Petri Nets.
2. Conflict free Petri nets.
3. One counter systems.
4. 2-dimensional VASS.
5. Flat systems.

Linear algebra

Integer programming.

Input: An integer matrix M and a vector $\vec{y}$.

Question: If there is a vector $\vec{x} \in \mathbb{N}^d$ such that

$$M \cdot \vec{x} = \vec{y}?$$

Theorem

The integer programming problem is NP-complete.

Linear algebra.

Linear programming.

Input: An integer matrix M and a vector $\vec{y}$.

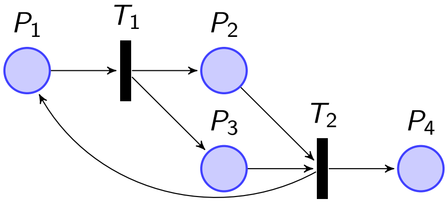
Question: If there is a vector $\vec{x} \in \mathbb{Q}^d_{\geqslant 0}$ such that

$$M \cdot \vec{x} = \vec{y}?$$

Theorem

The linear programming problem is P-complete.

Description of the net, three matrices.



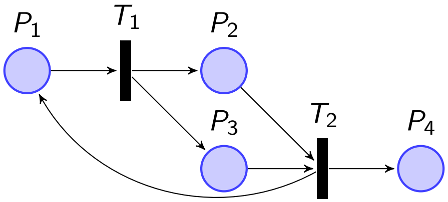
$$Pre(\mathcal{N}) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$Post(\mathcal{N}) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\Delta = Post(\mathcal{N}) - Pre(\mathcal{N})$$

$$\begin{bmatrix} -1 & 1 \\ 1 & -1 \\ 1 & -1 \\ 0 & 1 \end{bmatrix}$$

Description of the net, three matrices.



$$Pre(\mathcal{N}) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$Post(\mathcal{N}) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\vec{0}[i] = 0 \text{ for all } i$$

$$\Delta = Post(\mathcal{N}) - Pre(\mathcal{N})$$

$$\vec{1_p}[i] = \begin{cases} 1 & \text{if } \; p = i \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{bmatrix} -1 & 1 \\ 1 & -1 \\ 1 & -1 \\ 0 & 1 \end{bmatrix}$$

State equation.

Let *Reach*$(\mathcal{N}, \mathfrak{i})$ be a set of configurations reachable from $\mathfrak{i}$ in $\mathcal{N}$.

State equation.

Let $Reach(\mathcal{N}, \mathfrak{i})$ be a set of configurations reachable from $\mathfrak{i}$ in $\mathcal{N}$.

Let $L_{\mathbb{N}}RS(\mathcal{N}, \mathfrak{i}) = \{\vec{y} : \exists_{\vec{x} \in \mathbb{N}^d} \; M \cdot \vec{x} = \vec{y} - \mathfrak{i}\}$.

State equation.

Let $Reach(\mathcal{N}, \mathfrak{i})$ be a set of configurations reachable from $\mathfrak{i}$ in $\mathcal{N}$.

Hard to describe.

Let $L_{\mathbb{N}}RS(\mathcal{N}, \mathfrak{i}) = \{\vec{y} : \exists_{\vec{x} \in \mathbb{N}^d} \ M \cdot \vec{x} = \vec{y} - \mathfrak{i}\}$.

Easier to describe (NP-complete).

State equation.

Let $Reach(\mathcal{N}, \mathfrak{i})$ be a set of configurations reachable from $\mathfrak{i}$ in $\mathcal{N}$.

Hard to describe.

Let $L_{\mathbb{N}}RS(\mathcal{N}, \mathfrak{i}) = \{\vec{y} : \exists_{\vec{x} \in \mathbb{N}^d}\ M \cdot \vec{x} = \vec{y} - \mathfrak{i}\}$.

Easier to describe (NP-complete).

Let $L_{\mathbb{Z}}RS(\mathcal{N}, \mathfrak{i}) = \{\vec{y} : \exists_{\vec{x} \in \mathbb{Z}^d}\ M \cdot \vec{x} = \vec{y} - \mathfrak{i}\}$.

State equation.

Let $Reach(\mathcal{N}, \mathfrak{i})$ be a set of configurations reachable from $\mathfrak{i}$ in $\mathcal{N}$.

Hard to describe.

Let $L_{\mathbb{N}}RS(\mathcal{N}, \mathfrak{i}) = \{\vec{y} : \exists_{\vec{x} \in \mathbb{N}^d}\ M \cdot \vec{x} = \vec{y} - \mathfrak{i}\}$.

Easier to describe (NP-complete).

Let $L_{\mathbb{Z}}RS(\mathcal{N}, \mathfrak{i}) = \{\vec{y} : \exists_{\vec{x} \in \mathbb{Z}^d}\ M \cdot \vec{x} = \vec{y} - \mathfrak{i}\}$.

Easy to describe (PTime).

State equation.

Let $Reach(\mathcal{N}, \mathfrak{i})$ be a set of configurations reachable from $\mathfrak{i}$ in $\mathcal{N}$.

Hard to describe.

Let $L_{\mathbb{N}}RS(\mathcal{N}, \mathfrak{i}) = \{\vec{y} : \exists_{\vec{x} \in \mathbb{N}^d}\ M \cdot \vec{x} = \vec{y} - \mathfrak{i}\}$.

Easier to describe (NP-complete).

Let $L_{\mathbb{Z}}RS(\mathcal{N}, \mathfrak{i}) = \{\vec{y} : \exists_{\vec{x} \in \mathbb{Z}^d}\ M \cdot \vec{x} = \vec{y} - \mathfrak{i}\}$.

Easy to describe (PTime).

Lemma

$Reach(\mathcal{N}, \mathfrak{i}) \subseteq L_{\mathbb{N}}RS(\mathcal{N}, \mathfrak{i}) \subseteq L_{\mathbb{Z}}RS(\mathcal{N}, \mathfrak{i})$.

An application.

Algorithm 1 for reachability.

Start from the initial configuration $\mathfrak{i}$ and exhaustively build a graph of reachable configurations adding nodes one by one.

- if you find $\mathfrak{f}$ then return 1;
- if you can not visit any new configuration then return 0;
- if you run out of memory then return I don't know.

An application.

Algorithm 1 for reachability.

Start from the initial configuration $\mathfrak{i}$ and exhaustively build a graph of reachable configurations adding nodes one by one.

- if you find $\mathfrak{f}$ then return 1;
- if you can not visit any new configuration then return 0;
- if you run out of memory then return I don't know.

Algorithm 2 for reachability.

Start from the initial configuration $\mathfrak{i}$ and exhaustively build a graph of reachable configurations adding nodes one by one; but whenever you want to add a new node $\vec{x}$ to the graph you check if $\mathfrak{f} \in L_{\mathbb{N}}SR(\mathcal{N}, \vec{x})$. You add the node if and only if the answer is yes.

- if you find $\mathfrak{f}$ then return 1;
- if you can not add any new node then return 0;
- if you run out of memory then return "I don't know".

P-flows

$\vec{y}$ is called a P-flow iff $\vec{y} \cdot M = 0$.
If $\vec{y} \geqslant 0$ then we call it P-semiflow.

P-flows

$\vec{y}$ is called a P-flow iff $\vec{y} \cdot M = 0$. If $\vec{y} \geqslant 0$ then we call it P-semiflow.

Lemma

If $\mathfrak{f}$ is reachable from $\mathfrak{i}$ then $\vec{y} \cdot \mathfrak{f} = \vec{y} \cdot \mathfrak{i}$.

P-flows

$\vec{y}$ is called a P-flow iff $\vec{y} \cdot M = 0$. If $\vec{y} \geqslant 0$ then we call it P-semiflow.

Lemma

If $\mathfrak{f}$ is reachable from $\mathfrak{i}$ then $\vec{y} \cdot \mathfrak{f} = \vec{y} \cdot \mathfrak{i}$.

Question

How do we test a boundedness of a place using P-semiflows?

P-flows

$\vec{y}$ is called a P-flow iff $\vec{y} \cdot M = 0$. If $\vec{y} \geqslant 0$ then we call it P-semiflow.

Lemma

If $\mathfrak{f}$ is reachable from $\mathfrak{i}$ then $\vec{y} \cdot \mathfrak{f} = \vec{y} \cdot \mathfrak{i}$.

Question

How do we test a boundedness of a place using P-semiflows?

Lemma

Let $\vec{y}$ be a P-semiflow of the net $\mathcal{N}$, then the number of tokens is bounded for all $1 \leqslant i \leqslant d$ such that $\vec{y}[i] > 0$.

Structural boundedness

A place p in a net $\mathcal{N}$ is structurally bounded if for every initial marking $\mathfrak{i}$ the

$$max\{\vec{1_p}^T \cdot \vec{m} : \vec{m} \in RS(\mathcal{N}, \mathfrak{i})\} \text{ is finite.}$$

Structural boundedness

A place p in a net $\mathcal{N}$ is structurally bounded if for every initial marking $\mathfrak{i}$ the

$$max\{\vec{1_p}^T \cdot \vec{m} : \vec{m} \in RS(\mathcal{N}, \mathfrak{i})\} \text{ is finite.}$$

Theorem

A following conditions are equivalent:

1. a place p in the net $\mathcal{N}$ is structurally bounded,
2. there exists $\vec{y} \geqslant \vec{1_p}$ such that $\vec{y} \cdot \Delta \leqslant \vec{0}$,
3. there does not exist $\vec{x} \geqslant \vec{0}$ such that $\Delta \cdot \vec{x} \geqslant \vec{1_p}$.

Proof

Theorem

A following conditions are equivalent:

1. a place p in the net $\mathcal{N}$ is structurally bounded,
2. there exists $\vec{y} \geqslant \vec{1_p}$ such that $\vec{y} \cdot \Delta \leqslant \vec{0}$,
3. there does not exist $\vec{x} \geqslant \vec{0}$ such that $\Delta \cdot \vec{x} \geqslant \vec{1_p}$.

Proof

Theorem

A following conditions are equivalent:

1. a place p in the net $\mathcal{N}$ is structurally bounded,
2. there exists $\vec{y} \geqslant \vec{1_p}$ such that $\vec{y} \cdot \Delta \leqslant \vec{0}$,
3. there does not exist $\vec{x} \geqslant \vec{0}$ such that $\Delta \cdot \vec{x} \geqslant \vec{1_p}$.

1. $1 \implies 3$ by $\neg 3 \implies \neg 1$

Proof

Theorem

A following conditions are equivalent:

1. a place p in the net $\mathcal{N}$ is structurally bounded,
2. there exists $\vec{y} \geqslant \vec{1_p}$ such that $\vec{y} \cdot \Delta \leqslant \vec{0}$,
3. there does not exist $\vec{x} \geqslant \vec{0}$ such that $\Delta \cdot \vec{x} \geqslant \vec{1_p}$.

1. $1 \implies 3$ by $\neg 3 \implies \neg 1$
2. $3 \implies 2$ by a theorem related to dual programs theorem called alternative theorem.

Theorem

Exactly one of the following systems of equations has a solution:

$$A\vec{x} \geqslant \vec{b}.$$

$$\begin{aligned} \vec{y} &\geqslant \vec{0} \\ \vec{y}^T \cdot A &= \vec{0} \\ \vec{y}^T \cdot \vec{b} &> 0. \end{aligned}$$

Proof

Theorem

A following conditions are equivalent:

1. a place p in the net $\mathcal{N}$ is structurally bounded,
2. there exists $\vec{y} \geqslant \vec{1_p}$ such that $\vec{y} \cdot \Delta \leqslant \vec{0}$,
3. there does not exist $\vec{x} \geqslant \vec{0}$ such that $\Delta \cdot \vec{x} \geqslant \vec{1_p}$.

1. $1 \implies 3$ by $\neg 3 \implies \neg 1$
2. $3 \implies 2$ by a theorem related to dual programs theorem called alternative theorem.

Theorem

Exactly one of the following systems of equations has a solution:

$$A\vec{x} \geqslant \vec{b}.$$

$$\begin{aligned} \vec{y} &\geqslant \vec{0} \\ \vec{y}^T \cdot A &= \vec{0} \\ \vec{y}^T \cdot \vec{b} &> 0. \end{aligned}$$

3. $2 \implies 1$ Direct.

Continuous reachability.

Linear programming + If formula.

Input:	A $r \times c$- integer matrix M and a vector $\vec{y} \in \mathbb{Z}^r$ and a set of predicates of a form $\vec{x}[i] > 0 \implies \vec{x}[j] > 0$.
Question:	If there is a vector $\vec{x} \in \mathbb{Q}^c_{\geqslant 0}$ such that $M \cdot \vec{x} = \vec{y}$ and all predicates are satisfied?

Theorem

The Linear programming + If formula problem is in PTime.

Linear programming + If formula.

Input:	A $r \times c$- integer matrix M and a vector $\vec{y} \in \mathbb{Z}^r$ and a set of predicates of a form $\vec{x}[i] > 0 \implies \vec{x}[j] > 0$.
Question:	If there is a vector $\vec{x} \in \mathbb{Q}^c_{\geqslant 0}$ such that $M \cdot \vec{x} = \vec{y}$ and all predicates are satisfied?

Theorem

The Linear programming + If formula problem is in PTime.

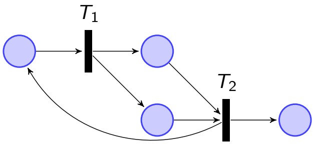
Proof

1. The set of solutions is convex.
2. If for every i there is a solution such that $\vec{x}[i] > 0$ then there is a solution such that $\vec{x}[j] > 0$ for all j.

Linear programming + If formula (the algorithm).

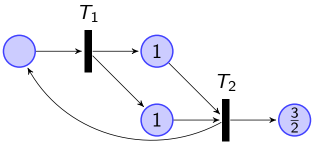
solve(Matrix Δ, Vector $\vec{y}$, set_of_implications $\mathbb{S}$, set_of_zeros $\mathbb{X}$)
{
If there is no solution $\Delta \cdot \vec{x} = \vec{y}$ in $\mathbb{Q}^d_{\geqslant 0}$,
where $x_i = 0$ for all $x_i \in \mathbb{X}$ then return false;
If there is a solution $\Delta \cdot \vec{x} = \vec{y}$ in $\mathbb{Q}^d_{\geqslant 0}$,
where $x_i = 0$ iff $x_i \in \mathbb{X}$ then return true;
Find a new coordinate x_i
which has to be equal 0 in every solution;
Add x_i to $\mathbb{X}$;
Add to $\mathbb{X}$ all x_j that has to be added due to implications;
return solve(M, $\vec{y}$, $\mathbb{S}$, $\mathbb{X}$);
}

Continuous Petri Nets.



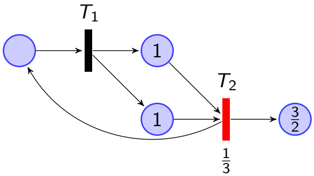
- Marking: $\mathcal{M} : \mathbb{P} \rightarrow \mathbb{Q}$
- Transitions: $\mathbb{T}$
- Firing a transition $\mathfrak{t} \in \mathbb{T}$ with a coefficient $a \in \mathbb{Q}$.

Continuous Petri Nets.



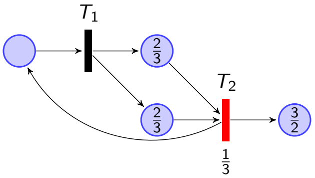
- Marking: $\mathcal{M} : \mathbb{P} \to \mathbb{Q}$
- Transitions: $\mathbb{T}$
- Firing a transition $\mathfrak{t} \in \mathbb{T}$ with a coefficient $a \in \mathbb{Q}$.

Continuous Petri Nets.



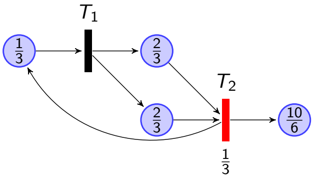
- Marking: $\mathcal{M} : \mathbb{P} \to \mathbb{Q}$
- Transitions: $\mathbb{T}$
- Firing a transition $\mathfrak{t} \in \mathbb{T}$ with a coefficient $a \in \mathbb{Q}$.

Continuous Petri Nets.



- Marking: $\mathcal{M} : \mathbb{P} \to \mathbb{Q}$
- Transitions: $\mathbb{T}$
- Firing a transition $\mathfrak{t} \in \mathbb{T}$ with a coefficient $a \in \mathbb{Q}$.

Continuous Petri Nets.



- Marking: $\mathcal{M} : \mathbb{P} \to \mathbb{Q}$
- Transitions: $\mathbb{T}$
- Firing a transition $\mathfrak{t} \in \mathbb{T}$ with a coefficient $a \in \mathbb{Q}$.

Continuous Petri Nets Reachability.

Input:	Two configurations $\mathfrak{i}$ and $\mathfrak{f}$
Question:	If there is a run form $\mathfrak{i}$ to $\mathfrak{f}$ under continuous semantics.

A simpler variant of the problem.

Suppose, that

$$\forall_i \ (\mathfrak{i}[i] > 0 \text{ and } \mathfrak{f}[i] > 0)\,.$$

$\mathfrak{f}$ is reachable from $\mathfrak{i}$ iff

$$\mathfrak{f} - \mathfrak{i} = \Delta \cdot \vec{x} \text{ where } \vec{x} \in \mathbb{Q}^d_{\geqslant 0}.$$

Continuous Petri Nets Reachability.

Lemma

$\mathfrak{f}$ is reachable from $\mathfrak{i}$ if

1. $$\mathfrak{f} - \mathfrak{i} = \Delta \cdot \vec{x} \text{ where } \vec{x} \in \mathbb{Q}^{d}_{\geqslant 0}$$

2. $$\vec{x}[i] > 0 \text{ and } Pre[j, i] > 0 \implies \mathfrak{i}[j] > 0,$$

3. $$\vec{x}[i] > 0 \text{ and } Post[j, i] > 0 \implies \mathfrak{f}[j] > 0.$$

Continuous Petri Nets Reachability.

Lemma

$\mathfrak{f}$ is reachable from $\mathfrak{i}$ if

1. $\mathfrak{f} - \mathfrak{i} = \Delta \cdot \vec{x}$ where $\vec{x} \in \mathbb{Q}^d_{\geqslant 0}$
2. $\vec{x}[i] > 0$ and $Pre[j, i] > 0 \implies \mathfrak{i}[j] > 0$,
3. $\vec{x}[i] > 0$ and $Post[j, i] > 0 \implies \mathfrak{f}[j] > 0$.

Theorem

$\mathfrak{f}$ is reachable from $\mathfrak{i}$ iff there are two configurations $\mathfrak{i}'$ and $\mathfrak{f}'$ such that

1. there is a run form $\mathfrak{i}$ to $\mathfrak{i}'$ that is using at most d steps.
2. there is a run form $\mathfrak{f}'$ to $\mathfrak{f}$ that is using at most d steps.
3. There is a run form $\mathfrak{i}'$ to $\mathfrak{f}'$ due to Lemma.

Translation to a formula (linear + If).

Lemma

For a given Petri net $\mathcal{N}$ and two configurations $\mathfrak{i}$ and $\mathfrak{f}$ in PTime one can compute a formula (linear programming + if) such that it is satisfiable if and only if $\mathfrak{f}$ is continuously reachable from $\mathfrak{i}$ in the net $\mathcal{N}$.

We use:

Theorem

$\mathfrak{f}$ is reachable from $\mathfrak{i}$ iff there are two configurations $\mathfrak{i}'$ and $\mathfrak{f}'$ such that

1. there is a run form $\mathfrak{i}$ to $\mathfrak{i}'$ that is using at most d steps.
2. there is a run form $\mathfrak{f}'$ to $\mathfrak{f}$ that is using at most d steps.
3. There is a run form $\mathfrak{i}'$ to $\mathfrak{f}'$ due to Lemma.

Q-cover 2015.

IDEA: Take a backward coverability algorithm, and speed it up.

Q-cover 2015.

IDEA: Take a backward coverability algorithm, and speed it up.

What is the main obstacle?

Q-cover 2015.

IDEA: Take a backward coverability algorithm, and speed it up.

CHALLENGE: Size of the representation of the representation of the upward-closed set may get too big.

Q-cover 2015.

IDEA: Take a backward coverability algorithm, and speed it up.

CHALLENGE: Size of the representation of the representation of the upward-closed set may get too big.

How to cut the upward-closed set?

Q-cover 2015.

IDEA: Take a backward coverability algorithm, and speed it up.

CHALLENGE: Size of the representation of the representation of the upward-closed set may get too big.

IDEA: Let $\vec{x} \in M\uparrow$, if there is no $\vec{y} \geqslant \vec{x}$ such that $\vec{y} \in RS(\mathcal{N}, \mathrm{i})$ then we can throw $\vec{x}$ away.

Q-cover 2015.

IDEA: Take a backward coverability algorithm, and speed it up.

CHALLENGE: Size of the representation of the representation of the upward-closed set may get too big.

IDEA: Let $\vec{x} \in M\uparrow$, if there is no $\vec{y} \geqslant \vec{x}$ such that $\vec{y} \in RS(\mathcal{N}, \mathfrak{i})$ then we can throw $\vec{x}$ away.

M. Blondin, A. Finkel, Ch. Haase, S. Haddad, 2015

SOLUTION: Let $\vec{x} \in M\uparrow$, if there is no $\vec{y} \geq \vec{x}$ such that $\vec{y} \in CRS(\mathcal{N}, \mathfrak{i})$ then we can throw $\vec{x}$ away.

Q-cover 2015.

IDEA: Take a backward coverability algorithm, and speed it up.

CHALLENGE: Size of the representation of the representation of the upward-closed set may get too big.

IDEA: Let $\vec{x} \in M\uparrow$, if there is no $\vec{y} \geqslant \vec{x}$ such that $\vec{y} \in RS(\mathcal{N}, \mathrm{i})$ then we can throw $\vec{x}$ away.

M. Blondin, A. Finkel, Ch. Haase, S. Haddad, 2015

SOLUTION: Let $\vec{x} \in M\uparrow$, if there is no $\vec{y} \geq \vec{x}$ such that $\vec{y} \in CRS(\mathcal{N}, \mathrm{i})$ then we can throw $\vec{x}$ away.

Thomas Geffroy, Jérôme Leroux, Grégoire Sutre, 2017

Actually, any over-approximation will work: LRS instead of CRS.

Advertisement.

Internships at the University of Warsaw.

Possibilities:

Prof. Mikołaj Bojanczyk

Logic, Automata, Formal Languages.

Email: bojan@mimuw.edu.pl

Prof. Piotr Sankowski

Algorithms.

Email: sank@mimuw.edu.pl

Prof. Stefan Dziembowski

Cryptography.

Email: S.Dziembowski@crypto.edu.pl