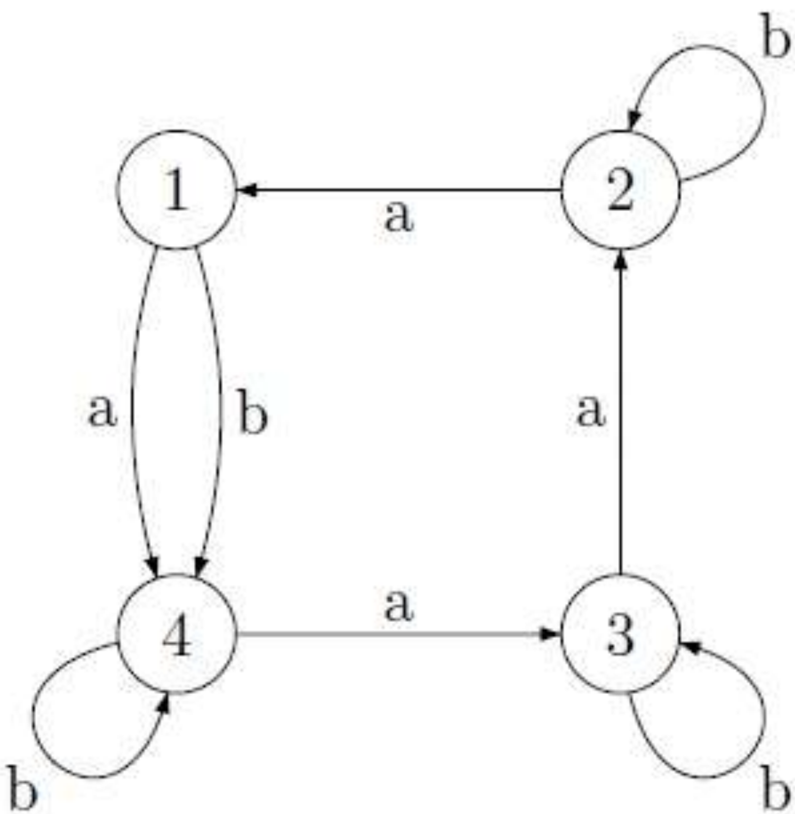
P-NP Threshold for Synchronizing Road Coloring

Adam Roman
Institute of Computer Science
Jagiellonian University
Krakow, POLAND

Faculty of
Mathematics and Computer Science
Jagiellonian University, Krakow, POLAND

Krakow

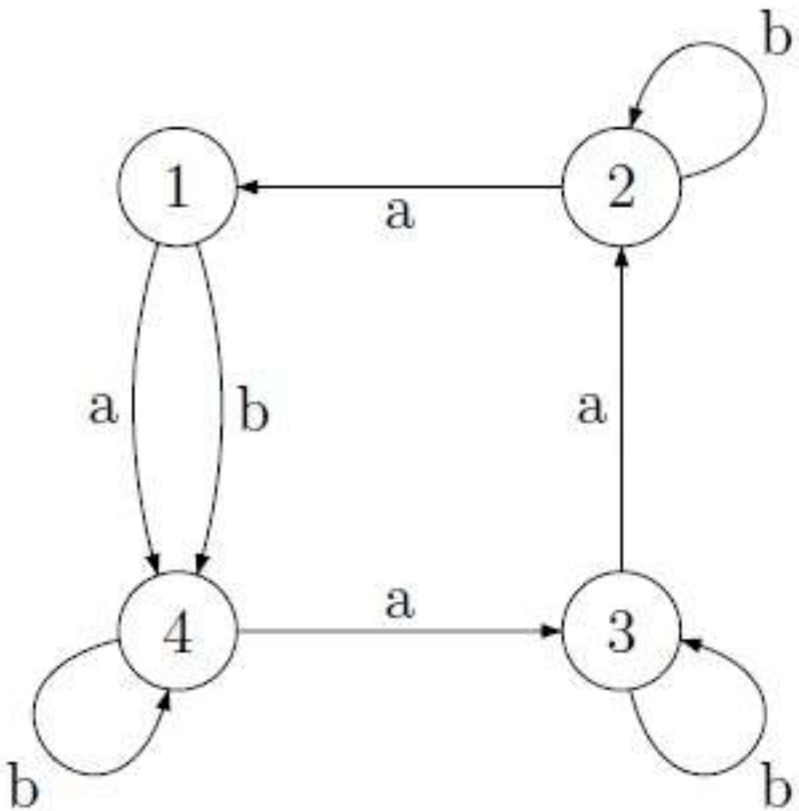
Automaton



- finite set of states Q
- finite alphabet A
- transition function δ

- we consider only deterministic automata
- that is, each state has |A| outgoing edges with different labels

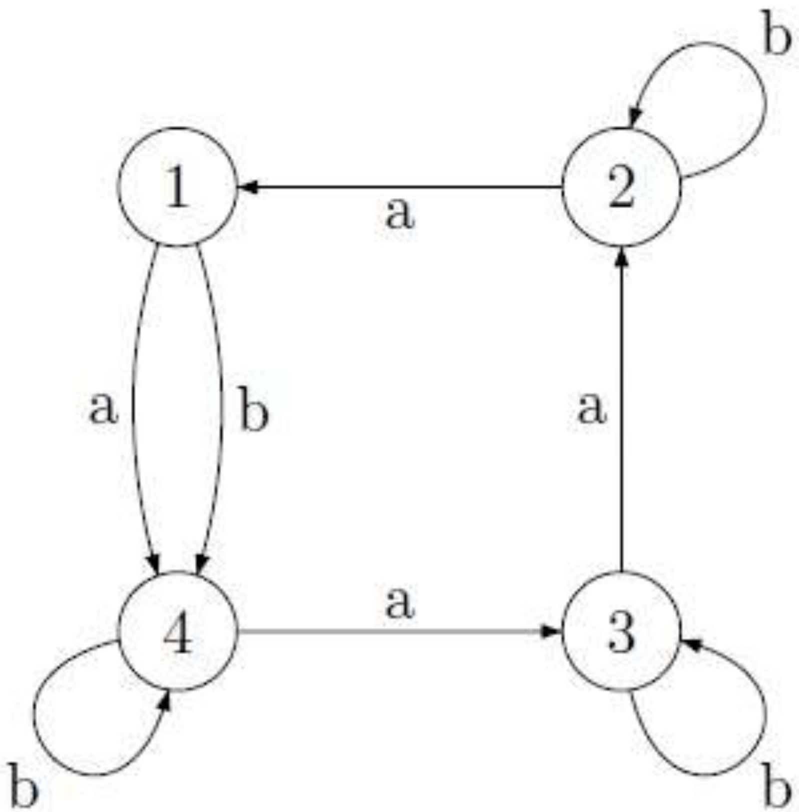
Synchronization



w = baaabaaab

letter	1	2	3	4
b	4	2	3	4
a	3	1	2	3
a	2	4	1	2
a	1	3	4	1
b	4	3	4	4
a	3	2	3	3
a	2	1	2	2
a	1	4	1	1
b	4	4	4	4

Minimal synchronizing word



w = baaabaaab

- $|w|=9$
- for this automaton there is no shorter word that would synchronize all states into one state
- in such case *w* is called the minimal sychronizing word

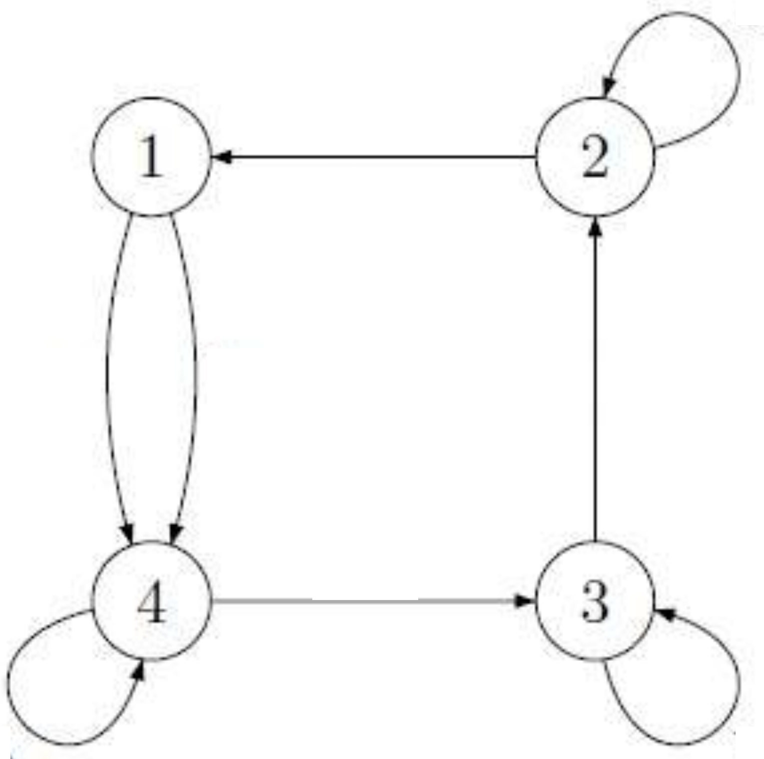
Complexity Problems for Synchronizing Automata

- given an automaton check if it is synchronizable
- given an automaton check if it can be synchronized by word of length C
- given an automaton A and a natural number n check if A has synchronizing word of length $\geq n$
- given an automaton A and a natural number n check if the minimal synchronizing word for A has length n

Complexity Problems for Synchronizing Automata

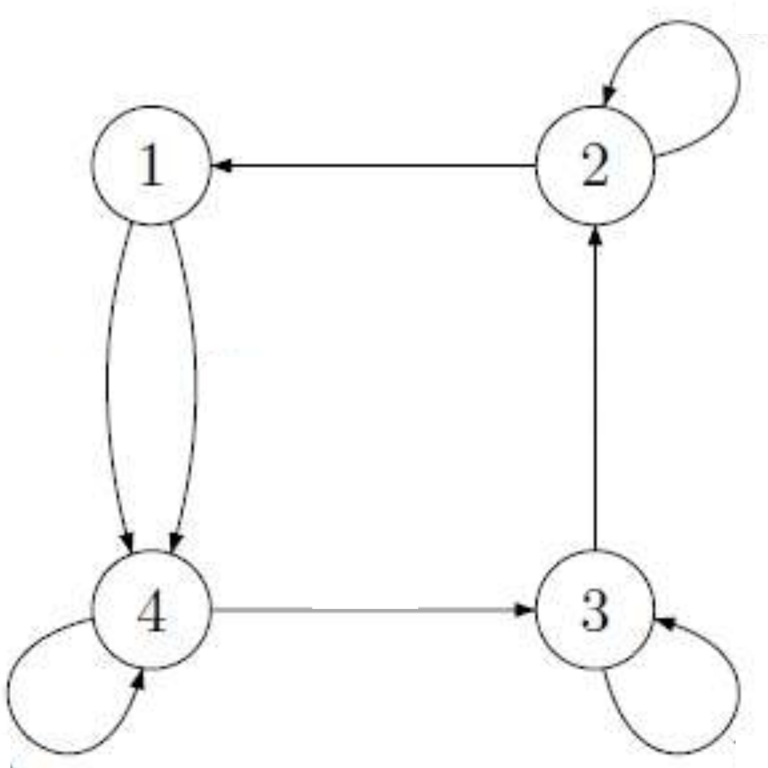
- given an automaton check if it is synchronizable
  P $O(|A| \times |Q|^2)$
- given an automaton check if it can be synchronized by word of length C
  P $O(|A|^C \times |Q|)$
- given an automaton A and a natural number n check if A has synchronizing word of length $\geq n$
  NP-complete
- given an automaton A and a natural number n check if the minimal synchronizing word for A has length n
  DP-complete

Road Coloring Problem (RCP)



- label the directed, constant out-degree graph such that it becomes a synchronizing automaton
- when can we do this? what is 'iff' condition?
- observation: if we can do this, then the g.c.d. of the lengths of all cycles = 1
- but is it also a sufficient condition?

Road Coloring Problem (RCP)



- Trahtman (2010): graph can be labeled in a synchronizable way

  if and only if

  the g.c.d. of the lengths of all cycles = 1
- it is easy to check this condition

New problems appear!

- each problem of the form:
  „given an automaton A check if COND(A) holds"
- can be transformed into:
  „given a constant out-degree graph G, check if there exists a labeling of G, converting G into A, such that COND(A) holds"

New problems appear!

- new problems may seem to be easier than their classical versions, since we can pick a labeling
- … but is it so?

New problems appear!

- new problems may seem to be easier than their classical versions, since we can pick a labeling
- … but is it so?
- we should check it!

CHALLENGE ACCEPTED

Problems and their complexities

Problem	Classical version complexity	'RCP' version complexity
is synchronizable?	P	P
has a synchronizing word of length $\leq C$ ($C=const$)?	P	!!!
has a synchronzing word of length n (n is part of the input)?	NPC	?
has a minimal synchronizing word of length n?	DPC	?

Main result

- The problem: given a constant out-degree (=|A|) graph G check if it can be labeled such that the resulting automaton has a synchronizing word of length $\leq C$
- Family of problems parametrized with 2 numbers: |A| and C
- The complexity depends on |A| and C!

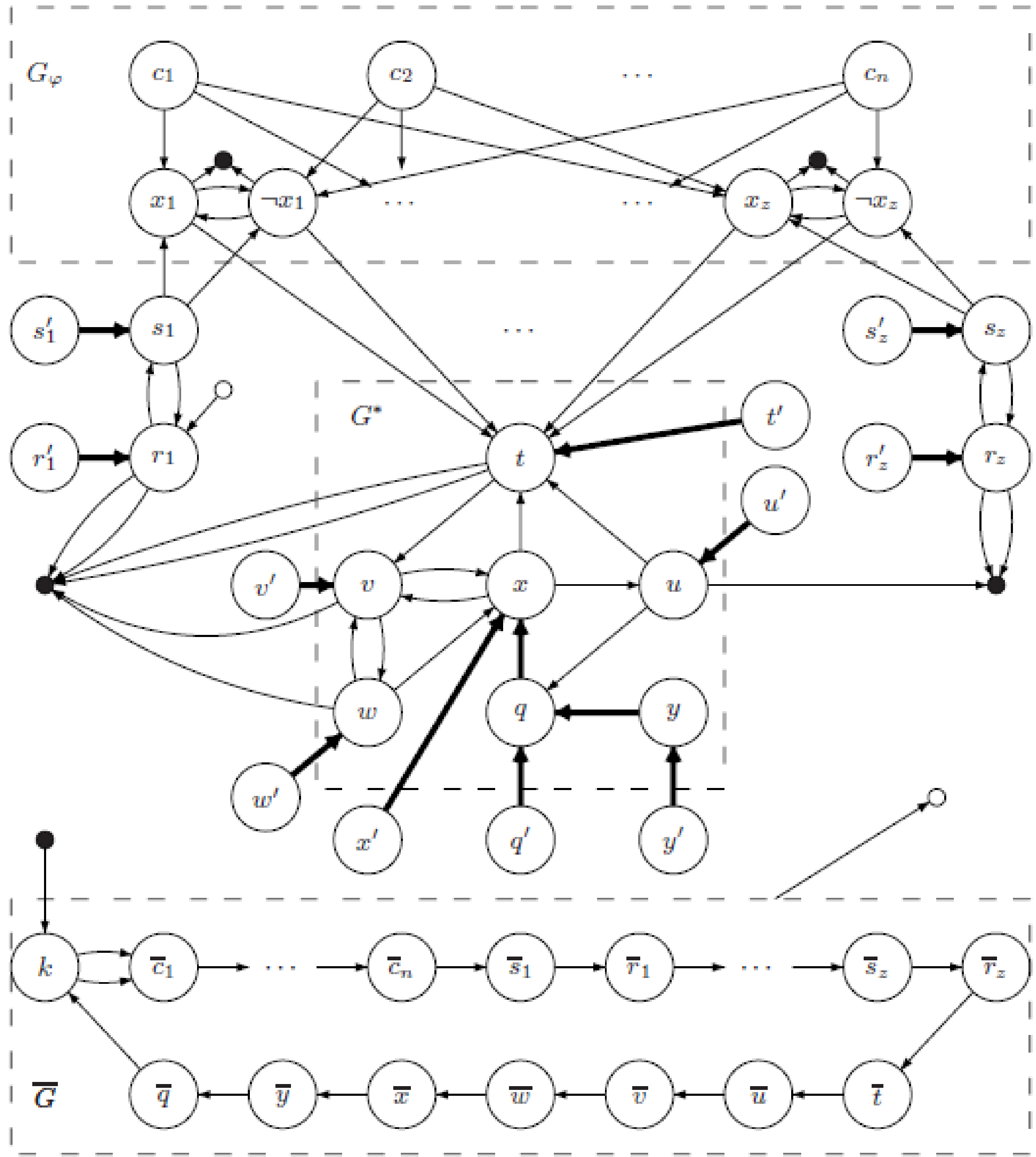
Main result

	C=1	C=2	C=3	C≥4
$\|A\|=1$	P	P	P	P
$\|A\|=2$	P	P	?	?
$\|A\|=3$	P	P	?	NPC
$\|A\|\geq 4$	P	P	?	NPC

|A|≥3

C ≥ 4

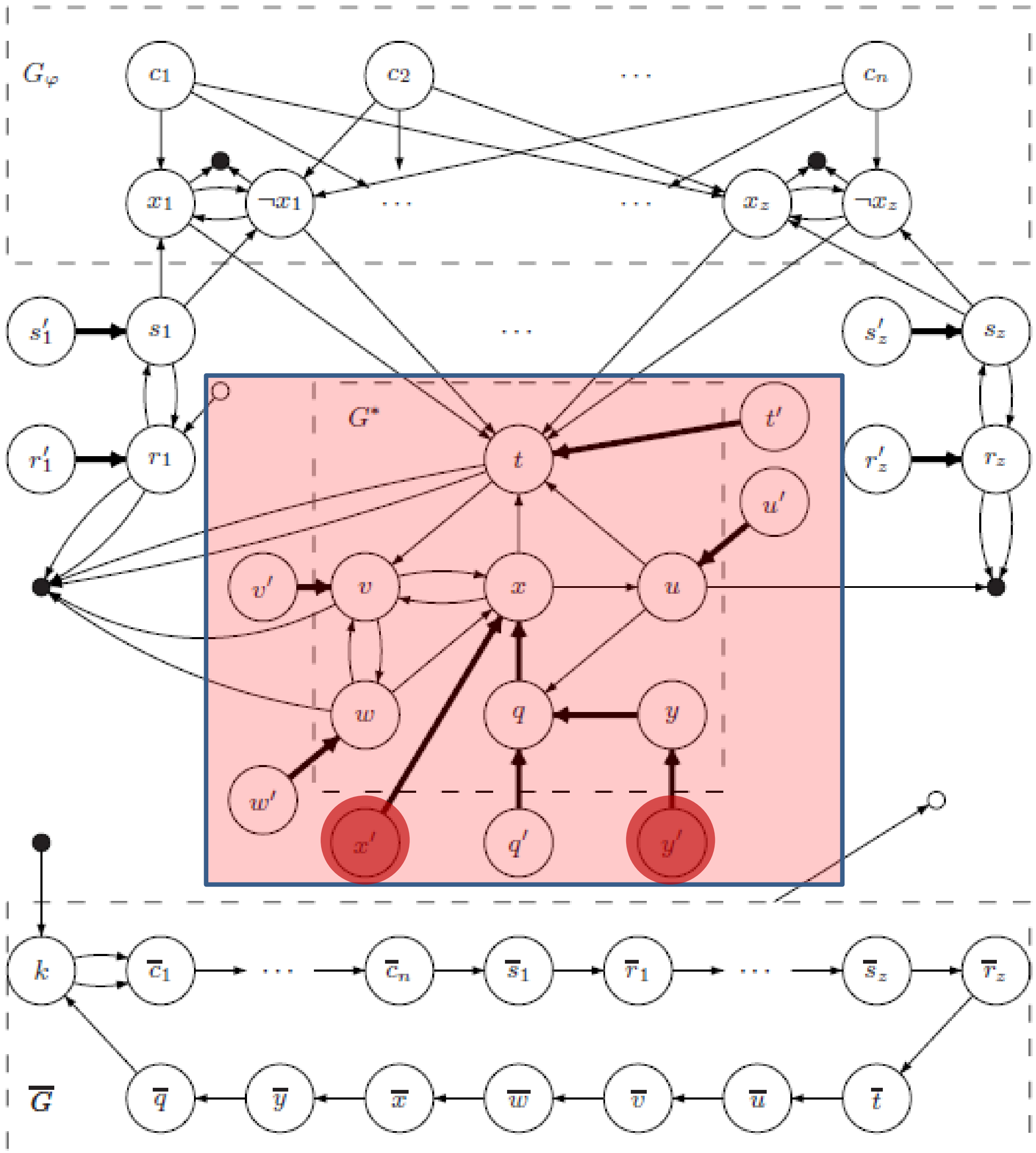
the key idea in the proof



|A|≥3

C ≥ 4

the key idea in the proof



The latest news

- Problem is polynomial for |A|≥3 and C=3 !!!
  - M. Drewienkowski, A. Roman

	C=1	C=2	C=3	C≥4
|A|=1	P	P	P	P
|A|=2	P	P	?	?
|A|=3	P	P	P	NPC
|A|≥4	P	P	P	NPC

The idea of the proof for C=3

- decompose main problem P into 'subproblems' $S_{w1}, \ldots, S_{wn}$ such that
$$P=S_{w1} \cup \ldots \cup S_{wn}$$
- in each subproblem we ask about synchronizing word of a given form *wi*
- it turns out that subproblems themselves and problems expressed as $S_{wp} \setminus S_{wq}$ and $S_{wp} \setminus S_{wq} \setminus S_{wr}$ are easy to solve
- express *P* as sum of differences of some subproblems

Conclusions

- we have almost completely solve the original problem…

Future work

- ALMOST…

- so first, check what with |A|=2
- then, check the complexities for ‘subproblems’ defined by the sets of words

Future work

- for example: what is the complexity of the problem for A={a,b} and some given C, but we allow only words that contain at most 1 letter b
  - this particular problem can be expressed in terms of Horn clauses, so it is in P
  - but what with other sets of words?...

This is the end. Thank you for your attention. Any questions?