Computation theory with atoms

I. Sets with atoms
II. Computation models with atoms

Sławomir Lasota
University of Warsaw

FoPSS School 2019: Nominal Techniques

II. Computation models with atoms

- automata with atoms
- Turing machines with atoms
- other models of computation

computation theory with atoms

orbit-finite automata
[Bojańczyk, Klin, L. 2011, 2014]

orbit-finite pushdown automata
[Clemente, L. 2015, 2019]

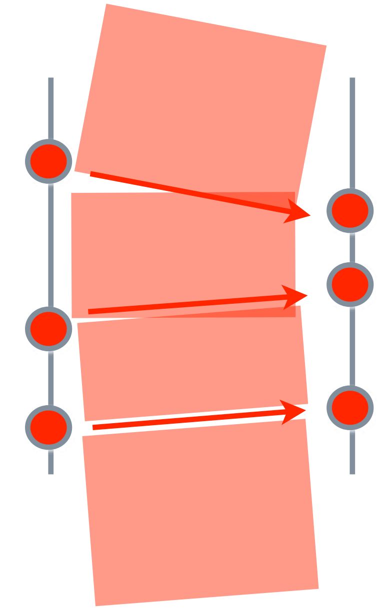
orbit-finite Turing machines
[Bojańczyk, Klin, L., Toruńczyk 2013]
[Klin, L., Ochremiak, Toruńczyk 2014]

tractability in orbit-finite computation
[Bojańczyk, Toruńczyk 2018]

programming languages processing orbit-finite objects
[Bojańczyk, Braud, Klin, L. 2012]
[Klin, Szynwelski 2016]
[Kopczyński, Toruńczyk 2016, 2017]

orbit-finite homomorphism/isomorphism problem
[Klin, Kopczyński, Ochremiak, Toruńczyk 2015]
[Klin, L., Ochremiak, Toruńczyk 2016]
[Keshvardoost, Klin, L., Ochremiak, Toruńczyk 2019]

orbit-finite logics
[Bojańczyk, Place 2012]
[Klin, Łełyk 2017]
[Klin, Eberhart 2019]



In the sequel, atoms are **well-behaved**:

- have finite vocabulary
- are homogeneous
- have bounded substructures
- are effective

hence oligomorphic and
FO = quantifier free logic

orbits of atoms$^{(n)}$ = substructures generated by n atoms

there is a function **b** such that substructures generated by n atoms have size bounded by **b**(n)

finitely generated substructures of atoms are computable

hence quantifier-free logic decidable

although may have arbitrarily high complexity

Automata

any well-behaved atoms

Nondeterministic automata:

- alphabet A
- states Q
- $\delta \subseteq Q \times A \times Q$
- $I, F \subseteq Q$

} orbit-finite sets instead of finite ones = definable sets

Deterministic automata:

- $\delta : Q \times A \to Q$
- initial state $\in Q$

Unambiguous automata, alternating automata: ….

?

Question: Consider an equivariant language accepted by a nondeterministic orbit-finite automaton.
Is this language accepted by an equivariant one?
What about deterministic automata?

Question: Consider an S-supported language accepted by a nondeterministic orbit-finite automaton.
Is this language accepted by an S-supported one?
What about deterministic automata?

?

- alphabet A
- states Q
- $\delta \subseteq Q \times (A \cup \{\varepsilon\}) \times Q$
- $I, F \subseteq Q$

Question: do ε-transition increase the power of nondeterministic automata?

input alphabet: atoms

language: "exactly two different atoms appear"

number of registers may vary from one orbit to another

states: $Q = \text{atoms}^{\leq 2} \cup \{\text{reject}\}$

transitions: $\delta : Q \times A \rightarrow Q$

$\delta((), a) =$	(a)	$a \in$ atoms
$\delta((a), b) =$	(ab)	$a \neq b$
$\delta((a), b) =$	(a)	$a = b$
$\delta((ab), c) =$	reject	$c \neq a, b$

initial state: $()$

accepting states: atoms^2

input alphabet: atoms

language: "exactly two different atoms appear"

registers are not necessarily ordered

states: $Q = \mathcal{P}_{\leq 2}(\text{atoms}) \cup \{\text{reject}\}$

transitions: $\delta : Q \times A \to Q$

$\delta(\varnothing, a) = \{a\}$ $\quad a \in \text{atoms}$

$\delta(\{a\}, b) = \{a, b\}$ $\quad a, b \in \text{atoms}$

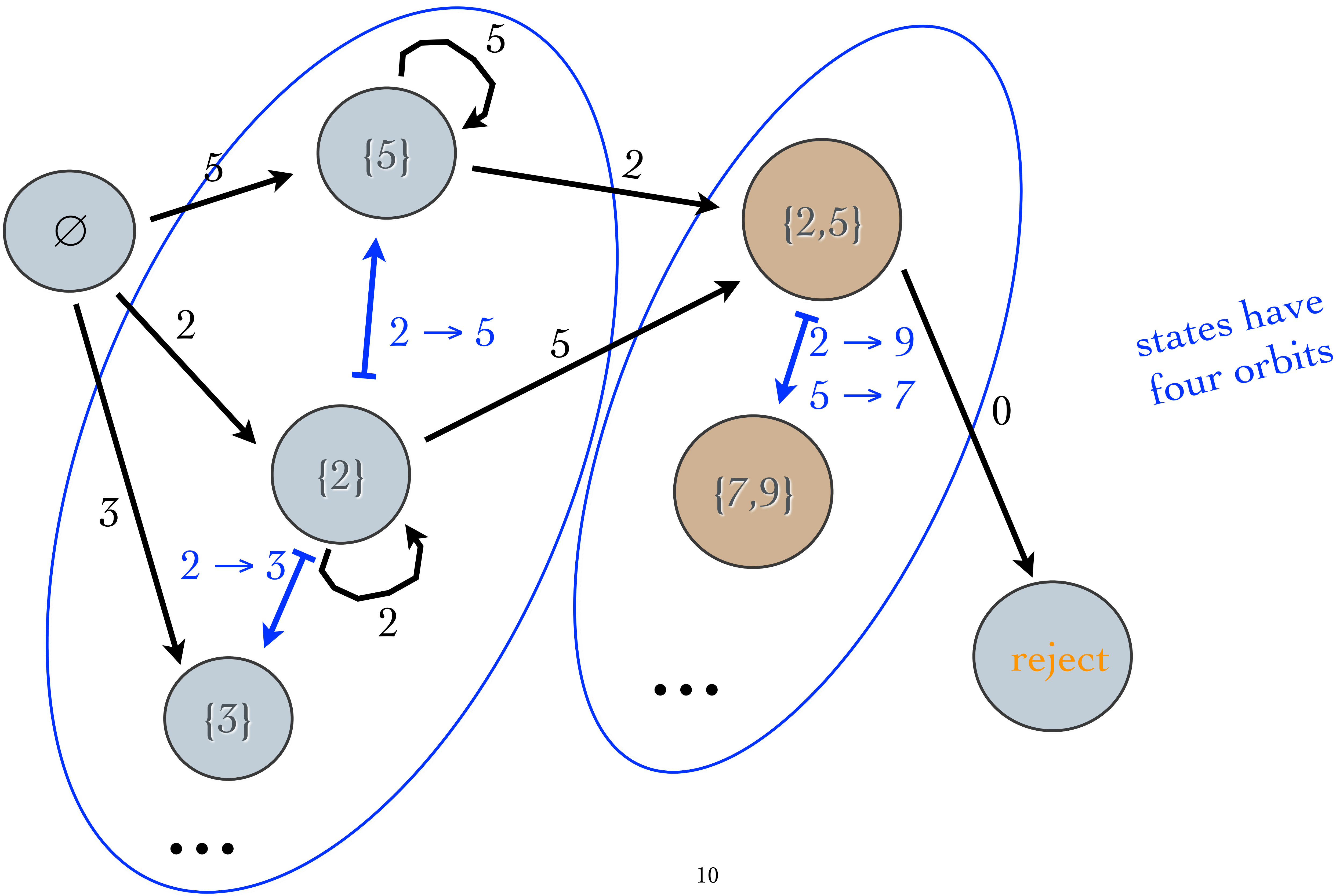
$\delta(\{a, b\}, c) = \text{reject}$ $\quad c \neq a, b$

initial state: $\varnothing$

accepting states: $\mathcal{P}_2(\text{atoms})$

input alphabet: atoms

language: "exactly two different atoms appear"



input alphabet: atoms

language: "last letter appears elsewhere and is different than *7*"

can it be determininized?

finitary nondeterminism

states: $Q = \text{atoms} \cup \{\text{init}, \text{accept}\}$

transitions: $\delta : Q \times A \rightarrow P_{\text{fin}}(Q)$

$\delta(\text{init}, a) = \{\text{init}, a\}$ $\quad a \in \text{atoms}, a \neq 7$

$\delta(a, b) = a$ $\quad a, b \in \text{atoms}, a \neq b$

$\delta(a, b) = \text{accept}$ $\quad a, b \in \text{atoms}, a = b$

initial state: init

accepting states: accept

input alphabet: atoms

language: "last letter **doesn't** appear elsewhere and is different than *7*"

states: Q = atoms $\cup$ {accept}

transitions: $\delta : Q \times A \to Q$

$\delta(a, a)$ = accept $\quad a \in$ atoms

$\delta(a, b)$ = a $\quad a, b \in$ atoms, $a \neq b$

infinitary nondeterminism

initial states: atoms \ {*7*}

accepting states: {accept}

input alphabet: P_2(atoms)

language: "nonempty intersection of all letters, or empty word"

can it be determininized?

states: Q = atoms

transitions: $\delta : Q \times A \to Q$

$\delta(a, \{a,b\}) = a \qquad a, b \in \text{atoms}, a \neq b$

initial states: atoms

accepting states: atoms

input alphabet: P_2(atoms)

language: ”nonempty intersection of all letters, or empty word”

states: $Q = \mathcal{P}_{\leq 2}(\text{atoms}) \cup \{\text{atoms}\}$

transitions: $\delta : Q \times A \to Q$

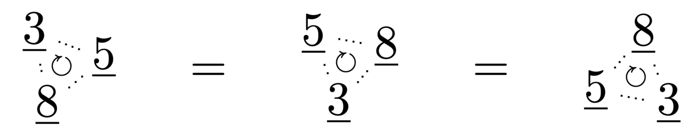
$\delta(x, y) = x \cap y$

initial states: {atoms}

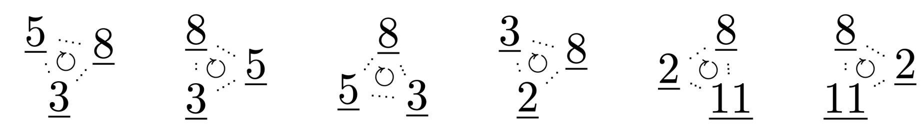
accepting states: all states except $\varnothing$

input alphabet: triples of atoms up to cyclic shift

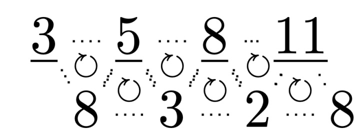
$\{(a,b,c),(b,c,a),(c,a,b)\}$ for a,b,c distinct



language: sequences like



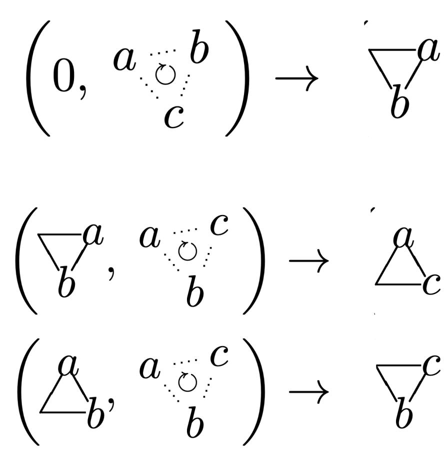
that can be glued into a chain



states: $\{0\} \cup \{\triangle(a,b), \triangledown(a,b) : a,b \text{ distinct}\}$

transitions: $\delta : Q \times A \to P_{fin}(Q)$

isn't it determininistic?



for a,b,c distinct

for a,b,c distinct

initial states: $\{0\}$

accepting states: all states except 0

input alphabet: atoms

language: nonempty monotonic words

states: Q = atoms $\cup$ $\{-\infty\}$

transitions: $\delta : Q \times A \to Q$

$\delta(-\infty, b) = b \quad b \in$ atoms

$\delta(a, b) = b \quad a, b \in$ atoms, $a < b$

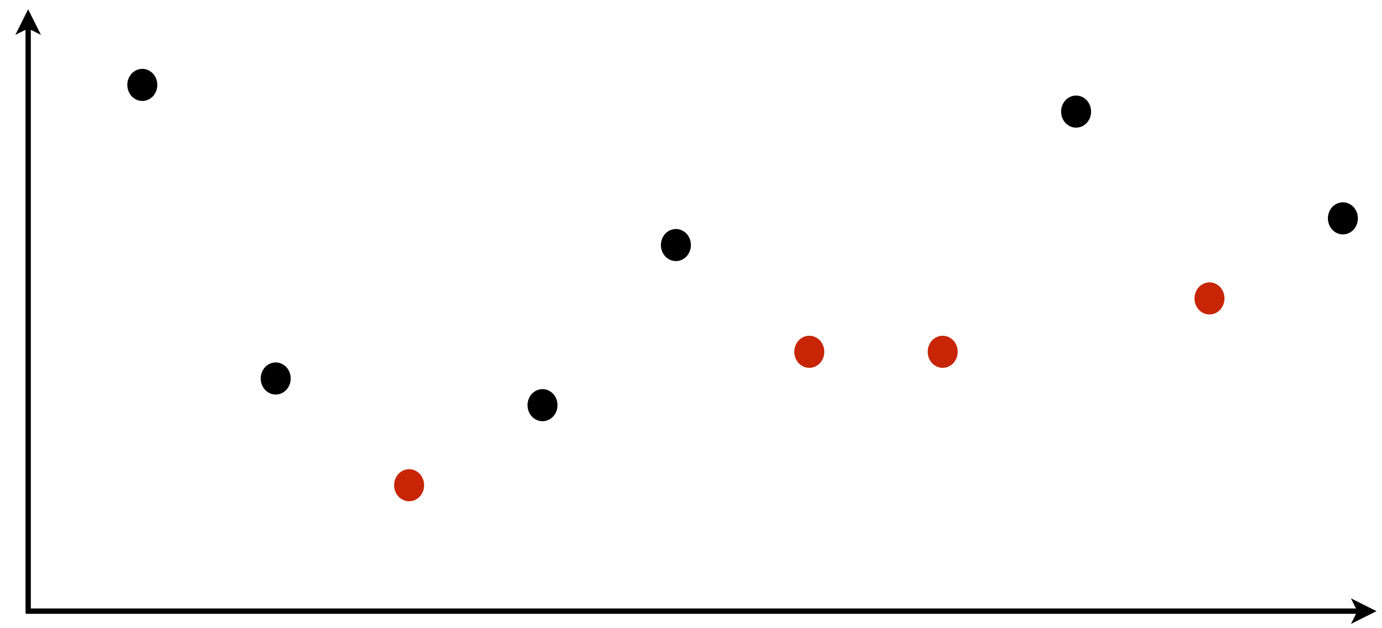
initial state: $-\infty$

accepting states: atoms

input alphabet: atoms

language: ”local minima are monotonic”

?



input alphabet: V

language: **dependent** words = "some subsequence of letters sums up to 0"

states: Q = atoms ∪ {init}

transitions: $\delta : Q \times A \rightarrow P_{fin}(Q)$

can it be determininized?

δ(init, a) = {init, a} a ∈ atoms

δ(a, b) = {a, a+b} a, b ∈ atoms

initial state: init

accepting state: 0

equality atoms $(\mathbb{N}, =)$

Theorem: Every equivariant orbit is isomorphic to atoms$^{(n)}$ modulo G, for some n and G a group of permutations of $\{1\ldots n\}$.

(Non)deterministic orbit-finite automata slightly generalize register automata:

- number of registers (dimension) may vary from one orbit to another
- registers are not necessarily ordered
- alphabet letters may contain more than one atom

ordered for total order atoms $(\mathbb{Q}, <)$

not a design decision but
a property of orbit-finite sets

Expressive power

~~non~~**deterministic**
register automata with
equality tests x = y

=

~~non~~**deterministic**
automata with equality atoms
over alphabet atoms × (a finite set)

- likewise for total order atoms (Q, ≤)

straight set: every orbit isomorphic to atoms$^{(n)}$ for some n

straight automata with equality atoms

Claim: Every (non)deterministic automaton over a straight alphabet A is equivalent to a straight one

Straightization (deterministic case)

Claim: Every (non)deterministic automaton over a straight alphabet A is equivalent to a straight one

straight set: every orbit isomorphic to $\text{atoms}^{(n)}$ for some n

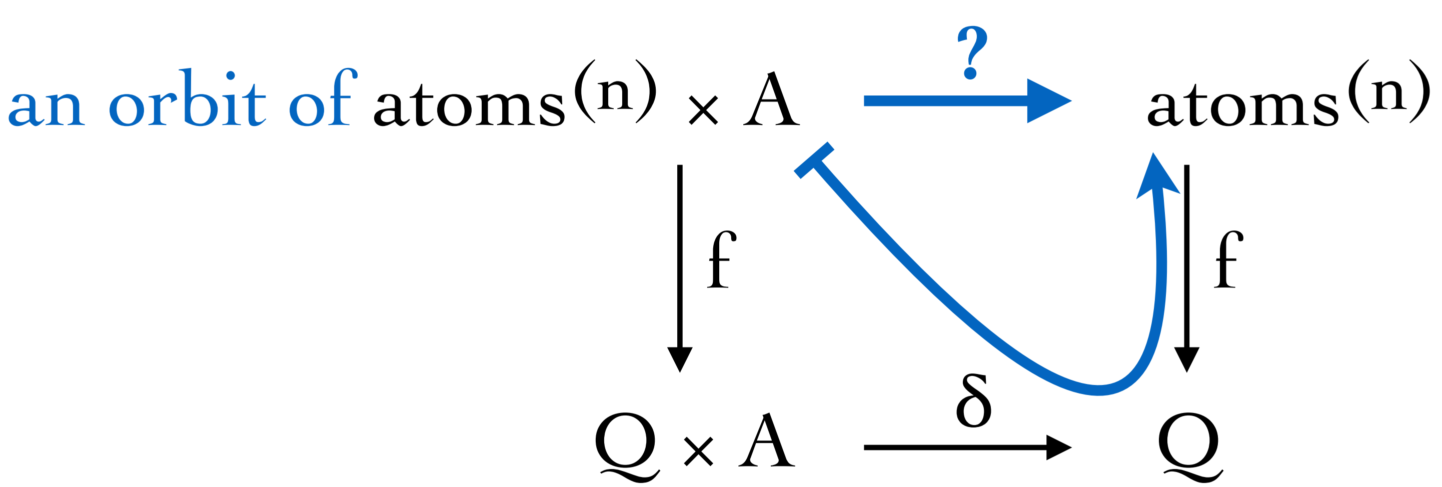
Think of 1-orbit Q

Theorem: Every equivariant orbit is isomorphic to $\text{atoms}^{(n)}/G$, for some n and G a group of permutations of {1…n}.

$f : \text{atoms}^{(n)} \to Q$ support-reflecting

- $\delta \subseteq Q \times A \times Q$ $\qquad f^{-1}(\delta) \subseteq \text{atoms}^{(n)} \times A \times \text{atoms}^{(n)}$

- $\delta : Q \times A \to Q$



Minimization

deterministic
register automata with equality tests $x = y$

=

deterministic
automata with equality atoms over alphabet atoms × (a finite set)

do not minimize

do minimize

any well-behaved atoms

Myhill-Nerode Theorem

Theorem: L is recognized by a deterministic automaton

iff

the set of L-equivalence classes is orbit-finite

The equivalence classes are states of the minimal automaton for L

Two words are L-equivalent
iff
they lead the minimal automaton to the same state

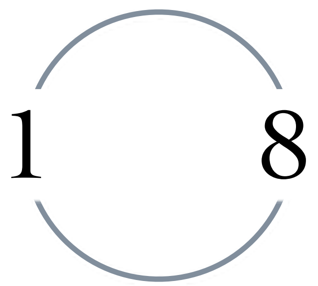
Every equivariant orbit is isomorphic to $\text{atoms}^{(n)}$ modulo G,
for some n and G a group of permutations of {1…n}.

Two words are L-equivalent
iff
they lead the minimal automaton to the same state

input alphabet: atoms

language: "exactly two different atoms appear"

18 and 81 are L-equivalent



after reading first two different data values, the minimal automaton should not remember their order!

this is impossible in register automata!

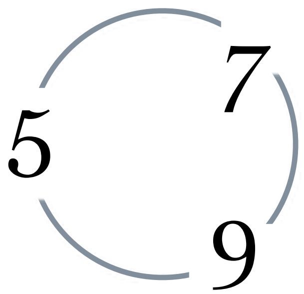
Every equivariant orbit is isomorphic to atoms$^{(n)}$ modulo G, for some n and G a group of permutations of {1...n}.

Two words are L-equivalent
iff
they lead the minimal automaton to the same state

input alphabet: atoms

language: {defdef, defefd, deffde : d, e, f pairwise different}

579, 795 and 957 are L-equivalent

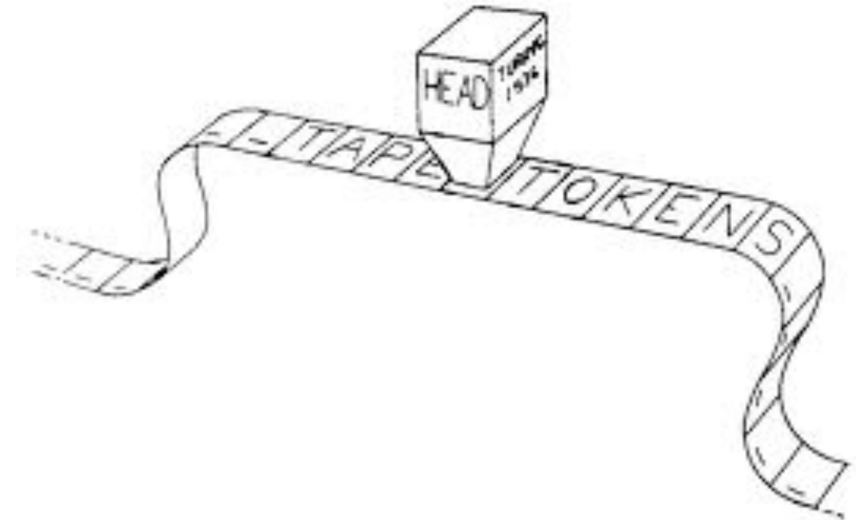


after reading first three letters, the minimal automaton should remember their order up to cyclic shift only!

again, this is impossible in register automata!

- automata with atoms
- **Turing machines with atoms**
- other models of computation

Turing machines



- tape alphabet A
- states Q
- subset $\delta \subseteq Q \times A \times Q \times A \times \{\leftarrow, \rightarrow, \downarrow\}$
- subsets $I, F \subseteq Q$

} orbit-finite sets instead of finite ones

Configurations = $A^* \times Q \times A^*$

Deterministic machines:

- $\delta : Q \times A \to Q \times A \times \{\leftarrow, \rightarrow, \downarrow\}$

input alphabet: atoms

language: "no atom appears twice":

$\{a_1 a_2 \ldots a_n \;:\; a_i \neq a_j \text{ when } i \neq j\}$

tape alphabet: A = atoms ∪ {⊥}

states: Q = atoms ∪ {start, accept, ret}

transitions: $\delta : Q \times A \to Q \times A \times \{\leftarrow, \rightarrow, \downarrow\}$

δ(start, a) =	(a, ⊥, →)	a ∈ atoms
δ(a, b) =	(a, b, →)	a ≠ b, a, b ∈ atoms
δ(a, B) =	(ret, B, ←)	a ∈ atoms
δ(ret, a) =	(ret, a, ←)	a ∈ atoms
δ(ret, ⊥) =	(start, ⊥, →)	
δ(start, B) =	(accept, B, →)	

input alphabet: $P_{\leq 10}$(atoms)

language: "some atom belongs to an odd number of letters”

?

Questions

1. Are TMs with atoms equivalent to classical TMs? **yes**

A - orbit-finite equivariant input alphabet
$L \subseteq A^*$ equivariant

- TM with atoms inputs a word $w \in A^*$
- classical TM inputs **definition** of w

2. Do TMs with atoms determinize? **no!** **$P \neq NP$**

3. Do TMs with atoms determinize when alphabet = atoms? **yes**

4. Has **P vs NP** question the same answer as classically in this case? **$P \neq NP$**

1. Nondeterministic TMs with atoms = classical TMs

$L \subseteq A^*$ equivariant

- TM with atoms inputs a word $w \in A^*$
- classical TM inputs **definition** of w

atoms are **well-behaved**:

- have finite vocabulary
- are homogeneous
- have bounded substructures
- are effective

with atoms $\Longrightarrow$ classical:

- L recognized by a definable TM
- atom-less simulation by manipulating definitions

classical $\Longrightarrow$ with atoms (case A = atoms**):**

- L recognized by a classical TM
- TM with atoms, on input w:
  - computes the quantifier-free formula defining the orbit of w
  - atom-less simulation by manipulating definitions

1. Nondeterministic TMs with atoms = classical TMs

$L \subseteq A^*$ equivariant

- TM with atoms inputs a word $w \in A^*$
- classical TM inputs **definition** of w

atoms are **well-behaved**:

- have finite vocabulary
- are homogeneous
- have bounded substructures
- are effective

Fact: Every equivariant orbit finite set A admits a surjective equivariant function

$$f : \bigcup_{i\in I} \text{atoms}^{(n_i)} \longrightarrow A$$

classical $\Longrightarrow$ **with atoms** (case $A \neq$ atoms):

- L recognized by a classical TM
- $f^{-1}(L)$ too (alphabet = atoms)
- $f^{-1}(L)$ recognized by a TM with atoms M (previous slide)
- TM with atoms, on input w: **guess** $f^{-1}(w)$ and execute M

2. Do TMs with atoms determinize?

In case of equality atoms (N, =) this depends on input alphabet:

- atoms
- ordered pairs of atoms
- unordered pairs of atoms
- unordered pairs of ordered pairs of atoms

} standard

- ordered triples of pairs of atoms modulo even number of flips — non-standard!

In case of total order atoms (Q, <) they do.

alphabet: atoms

- deatomization: replace atoms with binary encodings

a sequence of atoms	2		1		1		9		1
deatomisation	1	#	10	#	10	#	100	#	10

- atom-less simulation of atom-full computation

Fact: TMs over this alphabet do determinize

alphabet: ordered pairs of atoms

(a, b) $\in$ atoms$^{(2)}$

- input word represents a directed graph
- nodes (atoms) can be computed using projections

$$(a, b) \mapsto a \qquad\qquad (a, b) \mapsto b$$

  and stored on the tape

- then any decidable property of directed graphs can be decided deterministically

Fact: TMs over this alphabet do determinize

alphabet: unordered pairs of atoms

equality atoms (N, =)

$$\{a, b\} \in \mathcal{P}_2(\text{atoms})$$

- input word represents an undirected graph
- can nodes (atoms) be computed?

$$\cancel{\{a, b\} \mapsto a}$$

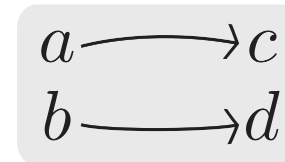
$$(\{a, b\}, \{b, c\}) \mapsto b$$

- then any decidable property of undirected graphs can be decided deterministically

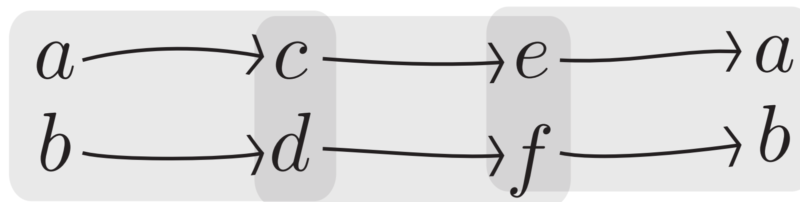
Fact: TMs over this alphabet do determinize

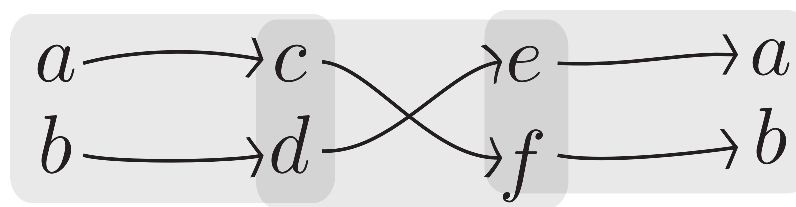
alphabet: unordered pairs of ordered pairs of atoms

$\{(a,c),(b,d)\}$

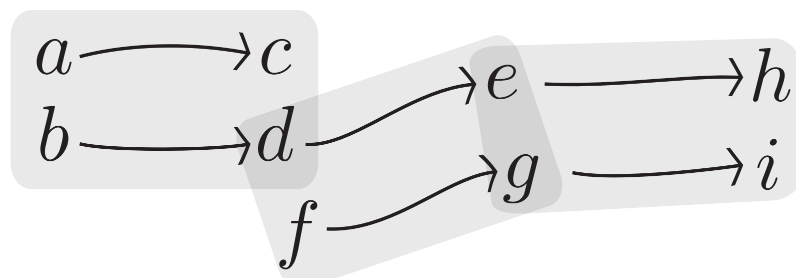


simple strips:



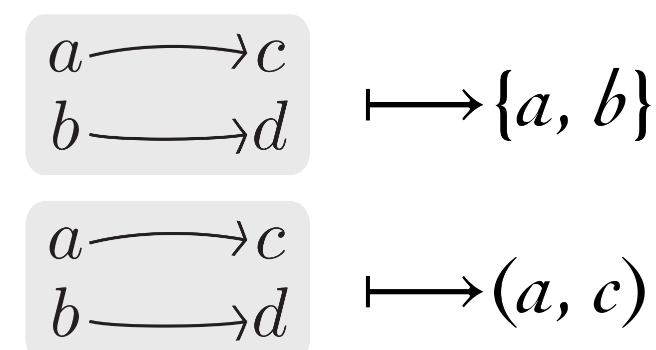


is not a simple strip

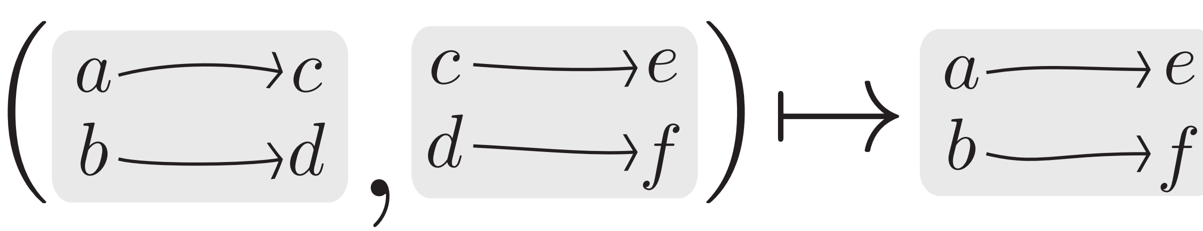


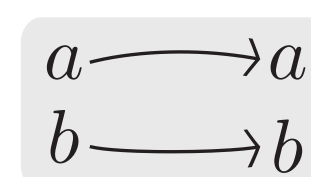
neither

which is legal?



Are simple strips recognized by a deterministic TM?

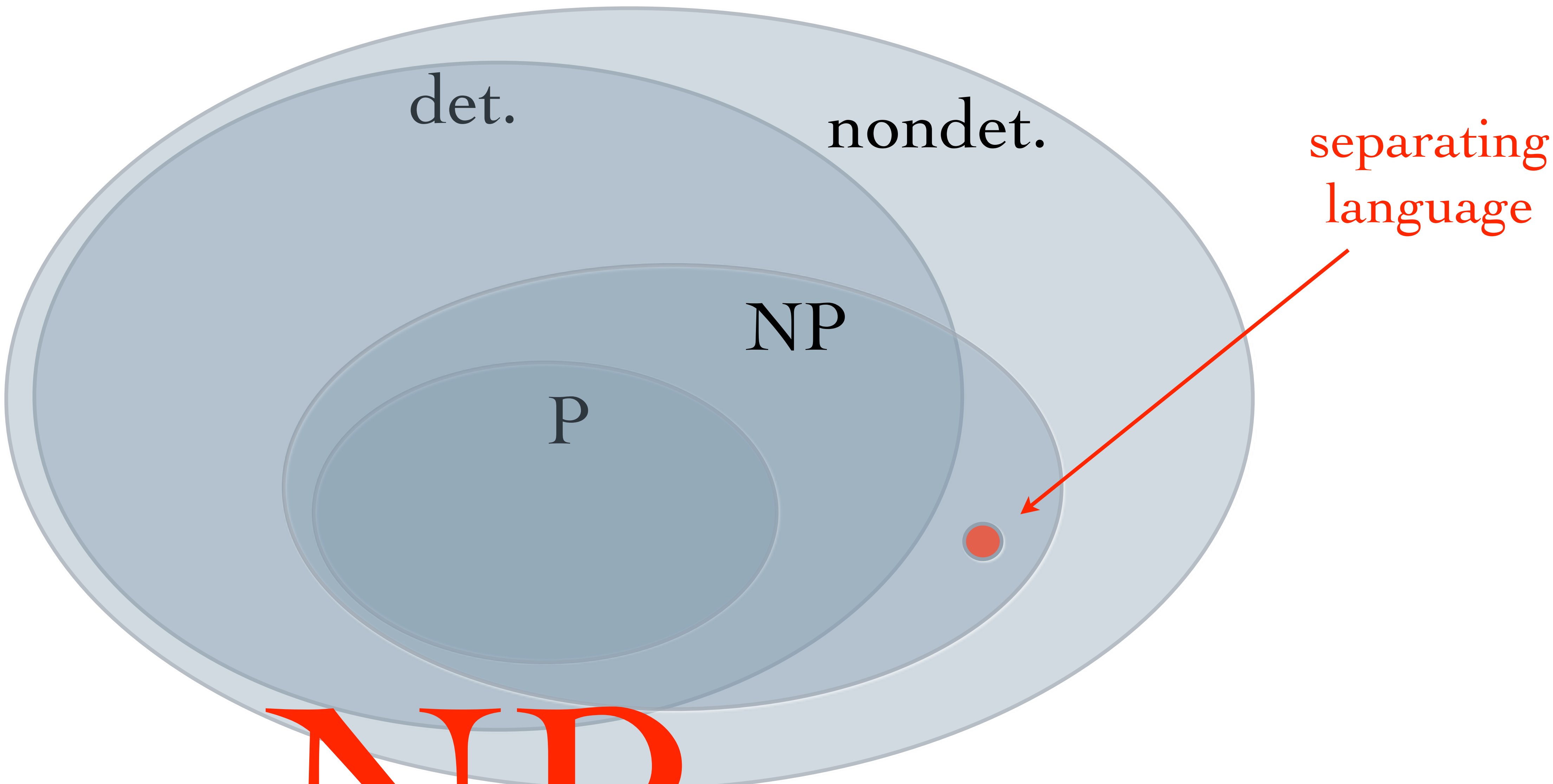




Fact: TMs over this alphabet do determinize

Theorem:

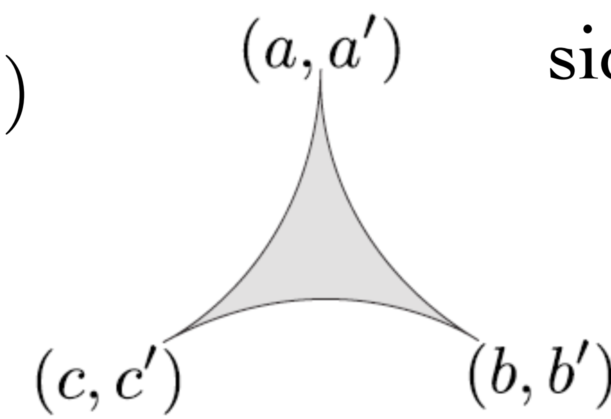
There is an alphabet A, and a language over A that is in NP but is not recognizable by a deterministic TM.



P ≠ NP

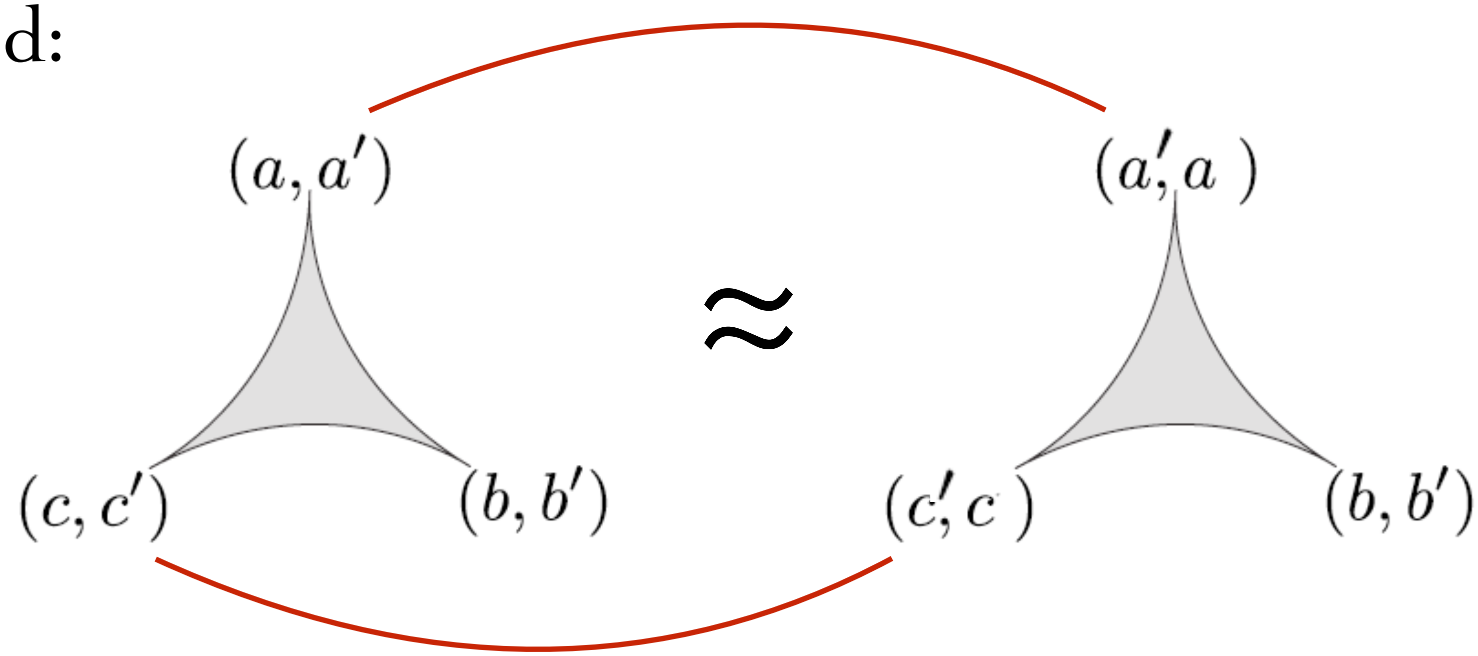
alphabet: ordered triples of ordered pairs of atoms modulo even number of flips

Triangle = $((a, a'), (b, b'), (c, c'))$



side set $\{a, a'\}$

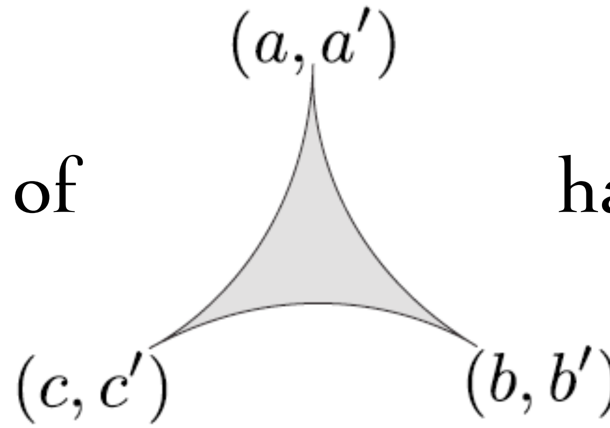
Let triangles with same side sets be equivalent if exactly two pairs are flipped:



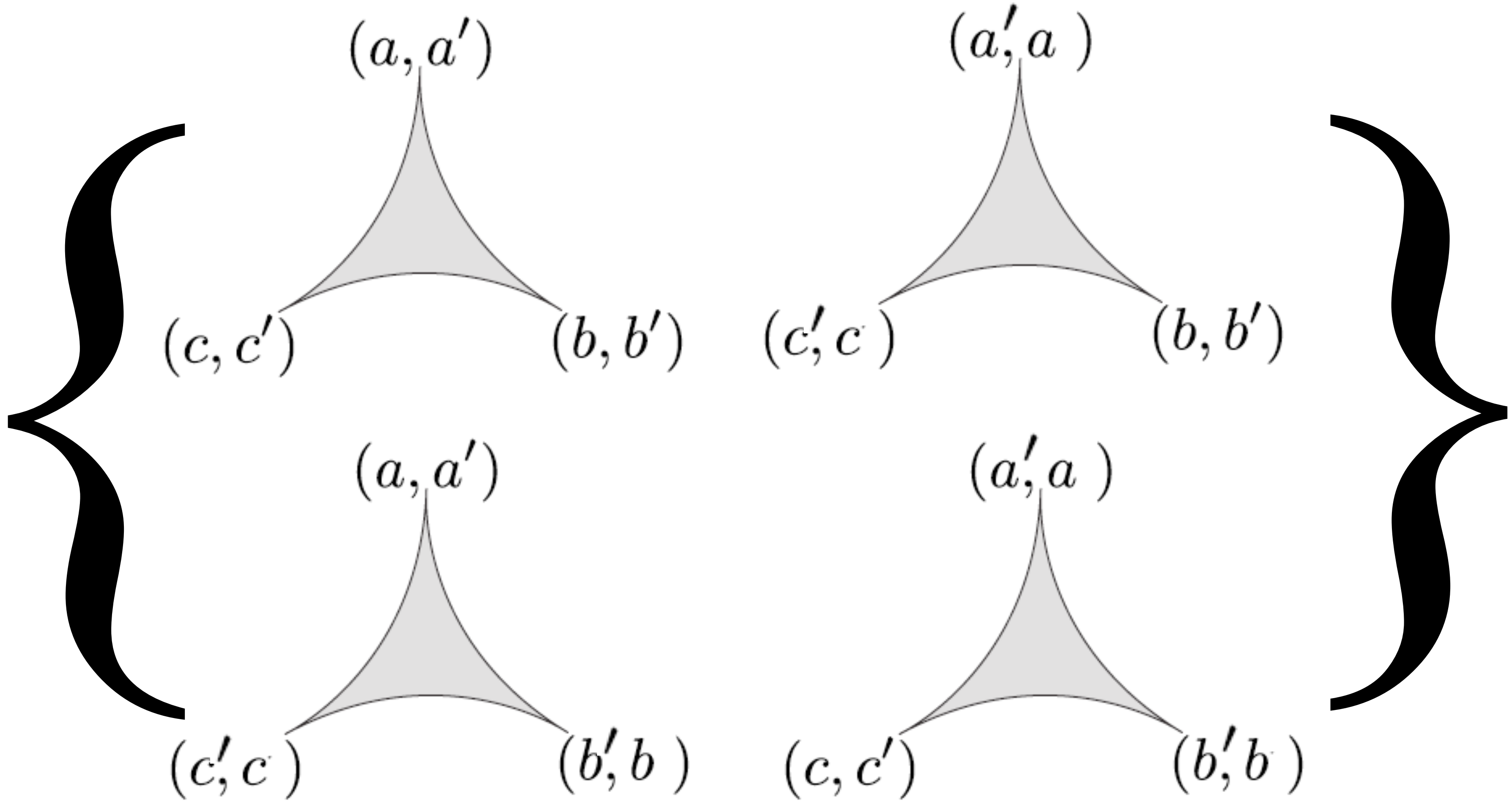
alphabet: equivalence classes of triangles

alphabet: ordered triples of ordered pairs of atoms modulo even number of flips

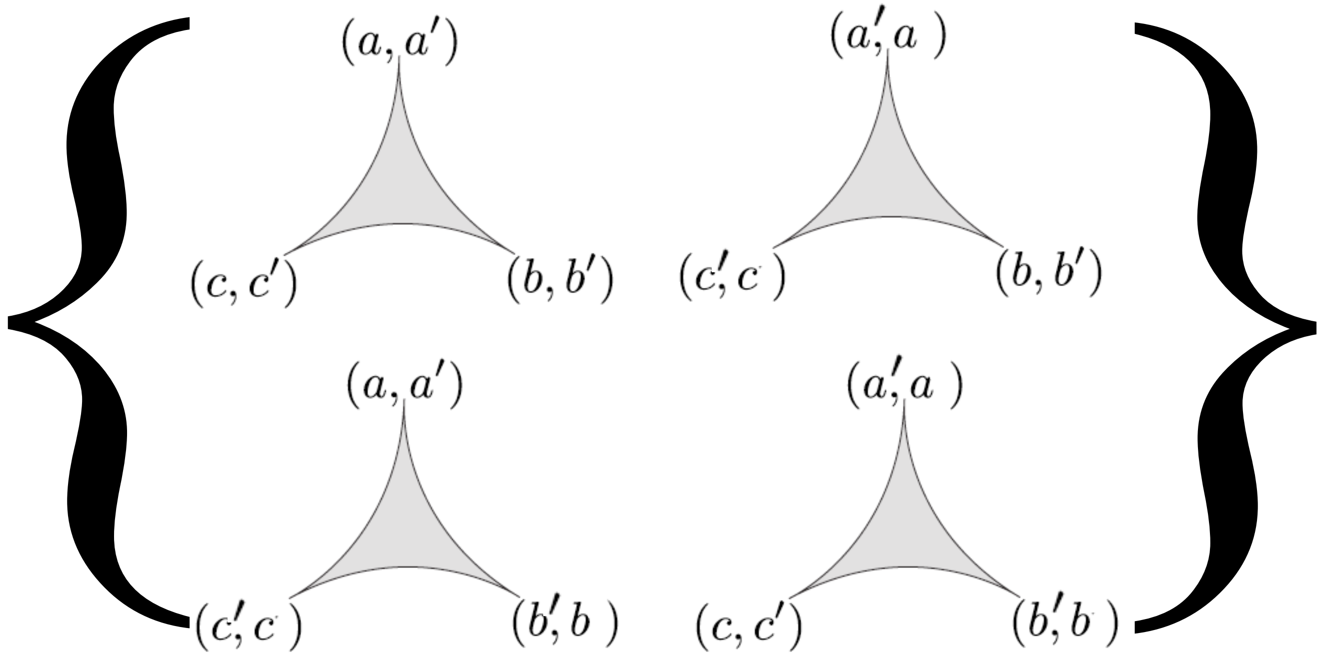
equivalence class of



has four elements:

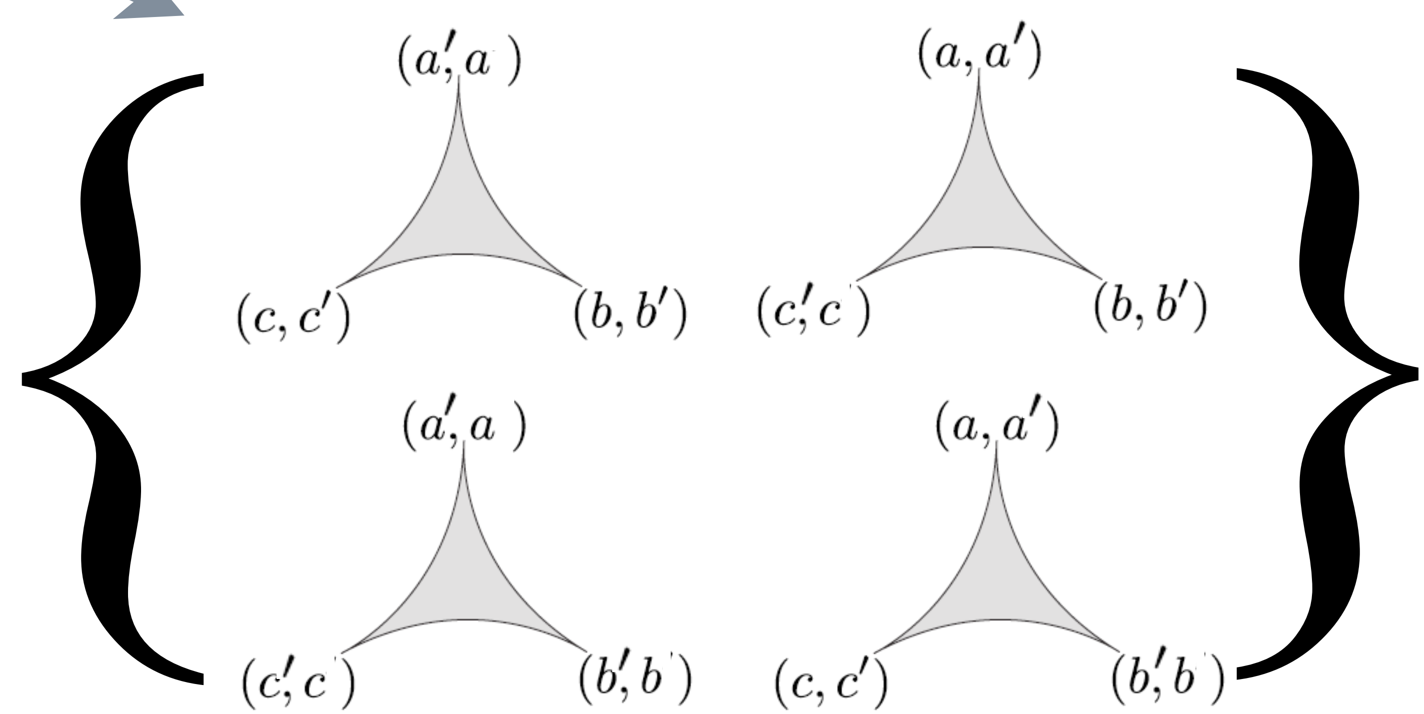


alphabet: ordered triples of
ordered pairs of atoms modulo even number of flips



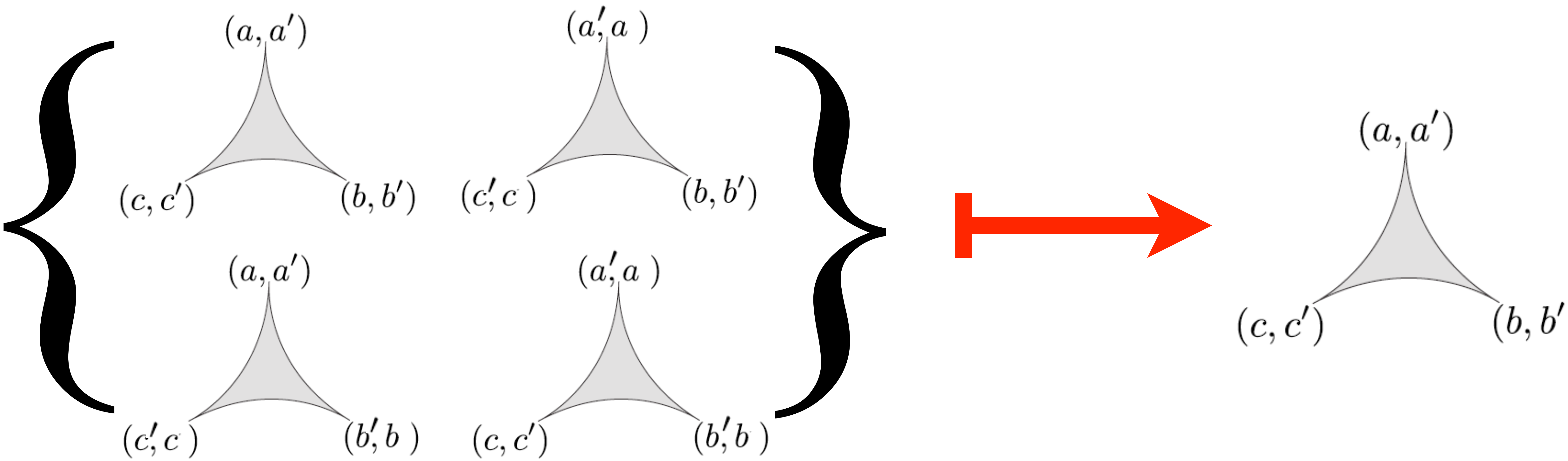
flip one pair

equivariant function



alphabet: ordered triples of
ordered pairs of atoms modulo even number of flips

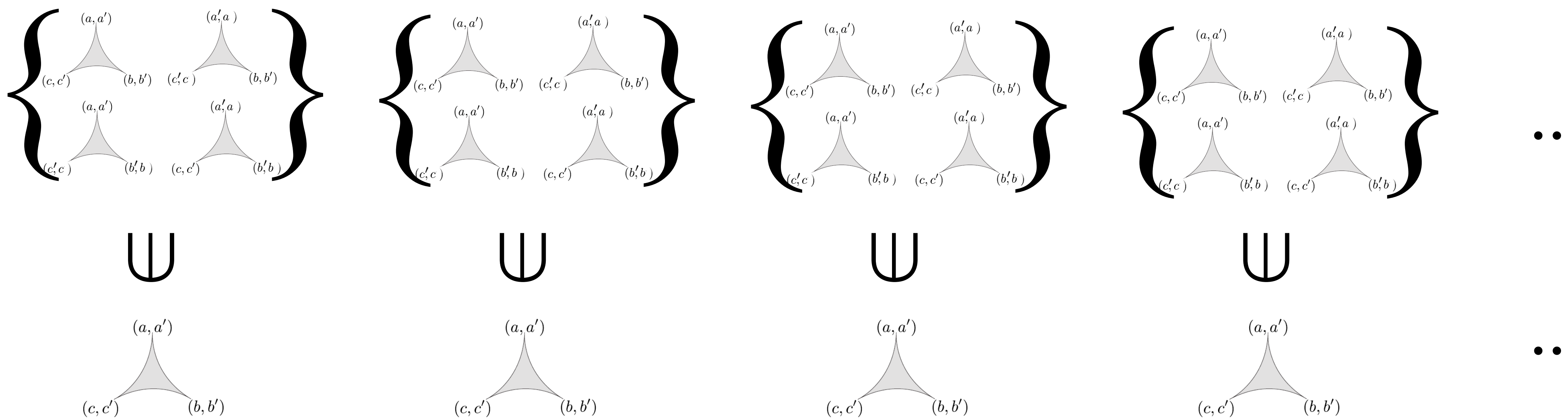
there is no function!



separating language

side sets either equal or disjoint

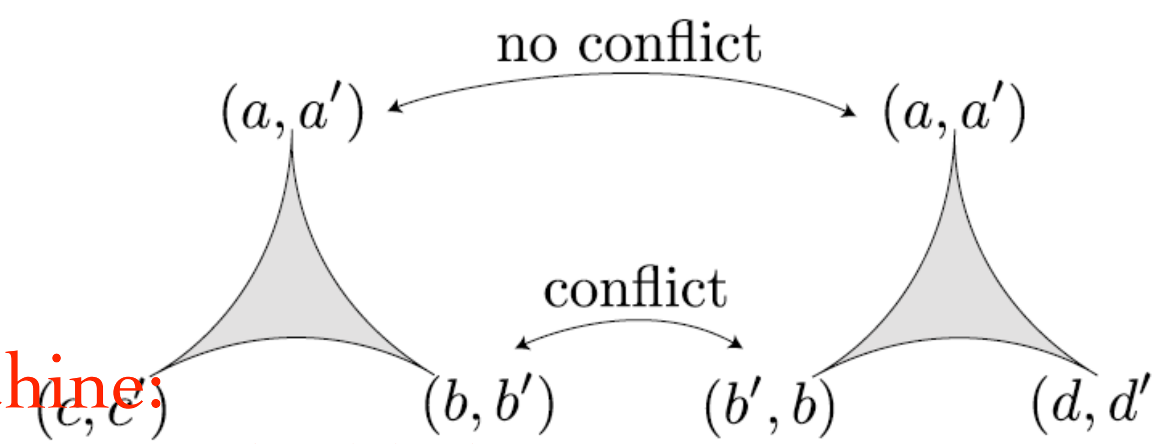
sequence of elements



Language: a word is in the language iff some sequence of elements is conflict-free

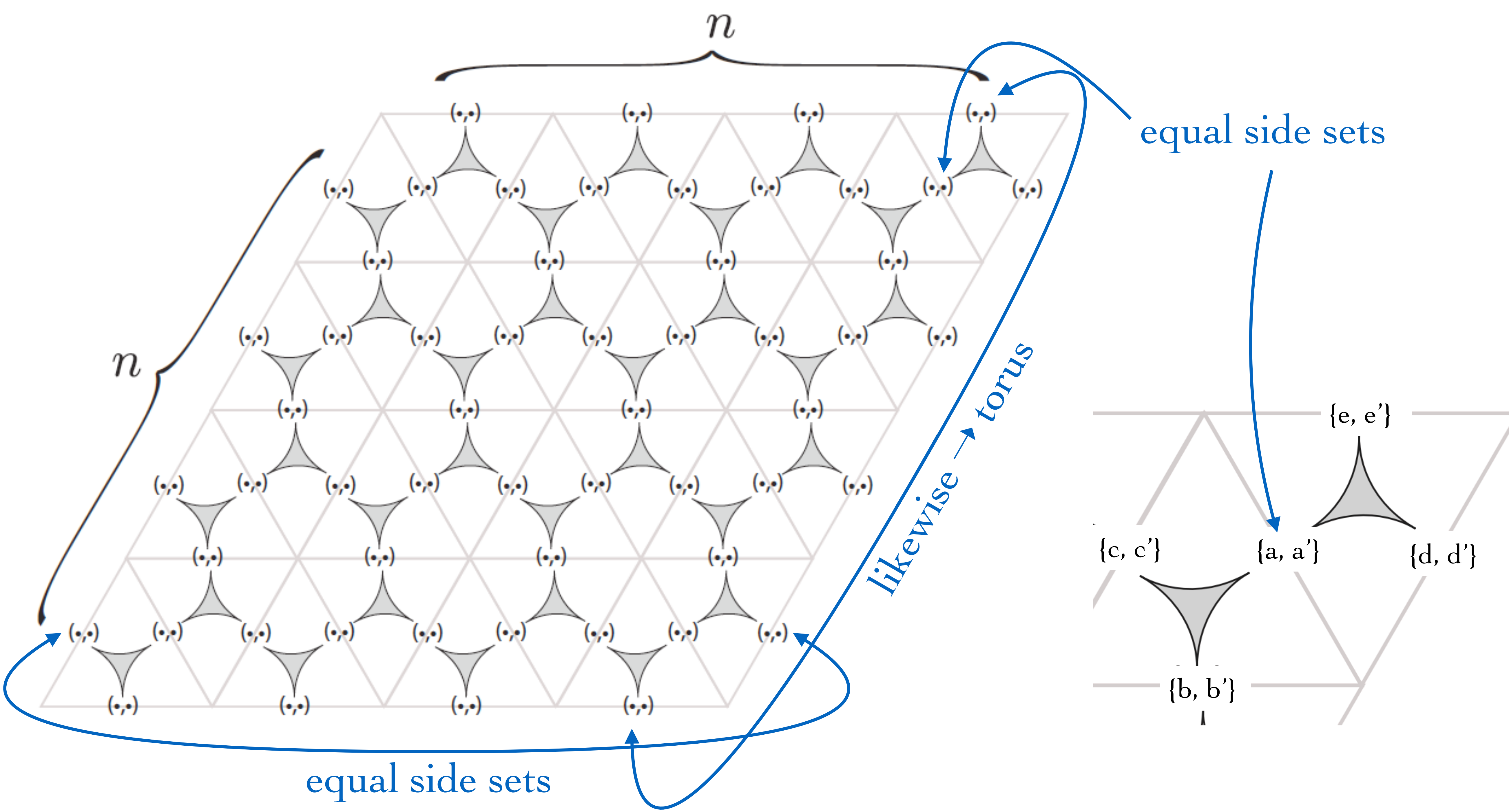
closely related to Cai-Fuerer-Immern

recognized in NP?



not recognized by a deterministic machine: enumeration of sequences of elements is not doable by a deterministic machine

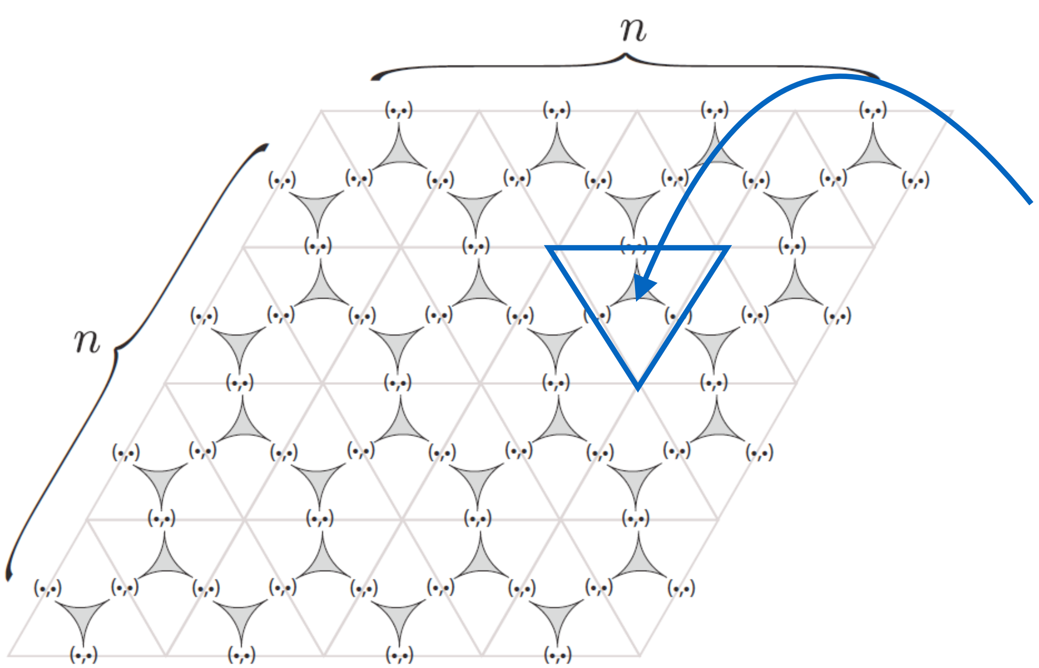
Hard inputs



For sufficiently large n, deterministic machine can not distinguish an input torus from a "flipped" one

but flipping alters membership in the language!

Hard inputs



Flipping one position **in a torus** alters membership in the language

Fix a deterministic machine M

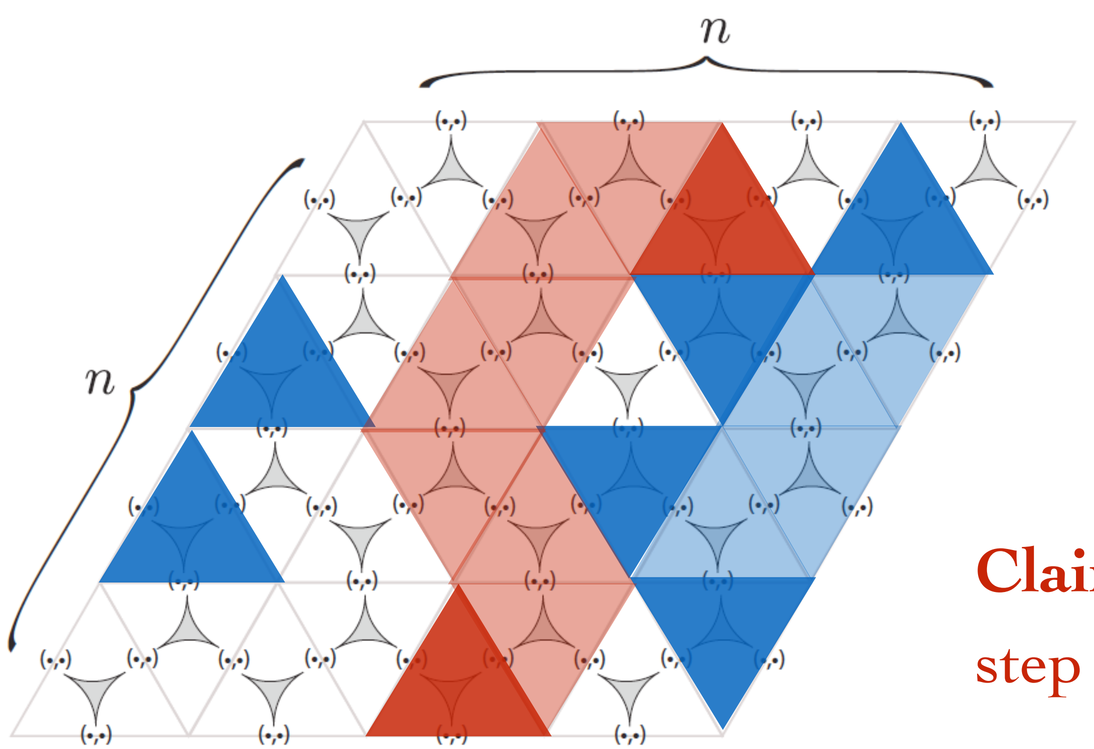
including possibly control state of the machine

Machine M **ignores** a position x after y steps at tape cell z:
content of cell z after y steps would remain the same if the position x was **flipped**

Claim: For n sufficiently large M ignores, after every step at every cell, all positions except for k^2 of them

k := twice the maximal support of a tape cell

Hard inputs



k := twice the maximal support of a tape cell

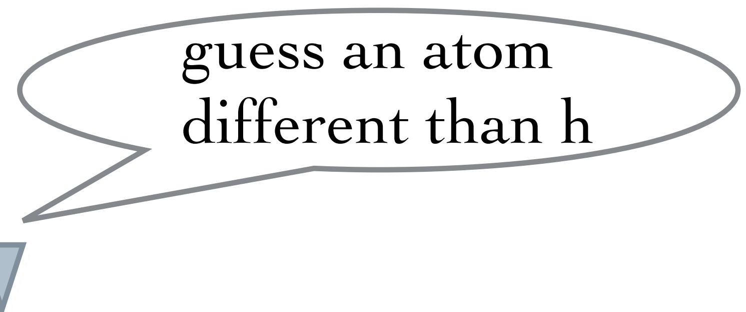
Observation: The greatest connected component C contains all except at most k^2 positions

Claim: For n sufficiently large M ignores, after every step at every cell, all positions except for k^2 of them

Induction on number of steps:

- **Induction base:** initially, M ignores, at every cell, all positions except that one
- **Induction step:**
  - cell content after a step depends on **three** neighbour cell contents before the step
  - hence M ignores, after the step, all except for $3k^2$
  - hence M ignores **some** position in C (for n sufficiently large)
  - hence M ignores **every** position in C (move the flip along the connecting path)

3. TMs with atoms determinize when alphabet = atoms



a d c d f d g y h e u s e d f e r g f f e d s

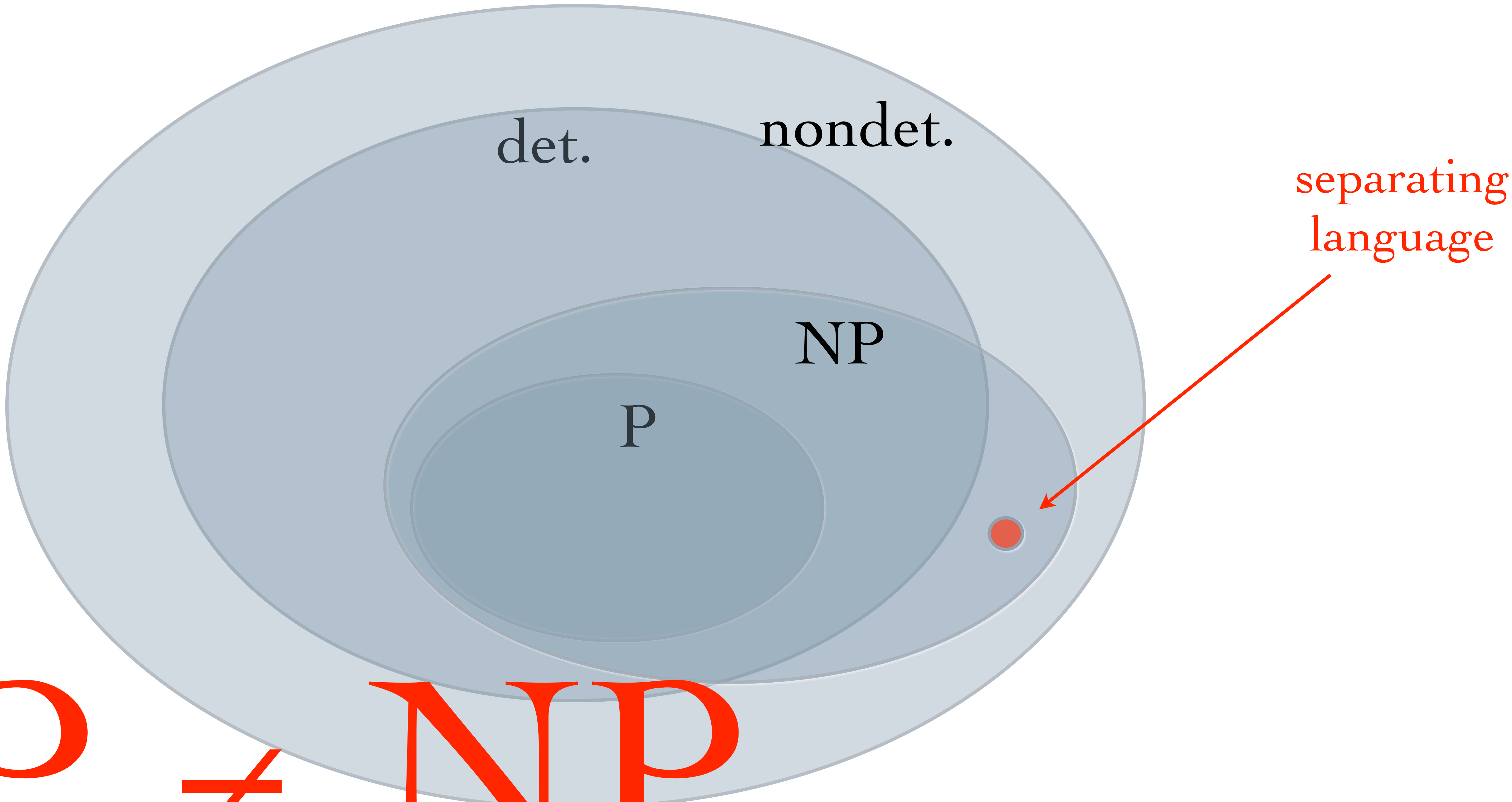
atoms are **well-behaved**:

- have finite vocabulary
- are homogeneous
- have bounded substructures
- are effective

- input word $w \in \text{atoms}^n$
- compute the quantifier-free formula defining the orbit of w
  = the substructure of atoms generated by w
- atom-less simulation by manipulating definitions

4. P ≠ NP when alphabet = atoms

Theorem:
There is a language over the alphabet of atoms that is in NP but not in P.



P ≠ NP

4. P ≠ NP when alphabet = atoms

Claim: $(a_1\ a_2 \ldots a_n), (b_1\ b_2 \ldots b_n) \in \text{atoms}^{(n)}$ are in the same orbit

iff

$$\sum_{i \in I} a_i = 0 \quad \text{iff} \quad \sum_{i \in I} b_i = 0 \text{ for for every } I \subseteq \{1 \ldots n\}$$

4. P ≠ NP when alphabet = atoms

input alphabet: V

language: **dependent** words = ”some subsequence of letters sums up to 0”

Fix a **deterministic** equivariant TM M recognizing the language in polynomial time

W.l.o.g. assume that states Q and tape alphabet T are **straight**:
Every orbit of Q or T is isomorphic to atoms$^{(n)}$ for n ≤ N

Consider the rejecting run on sufficiently long **independent** input word w
We fool M with a **dependent** input w' which M will forcedly reject too

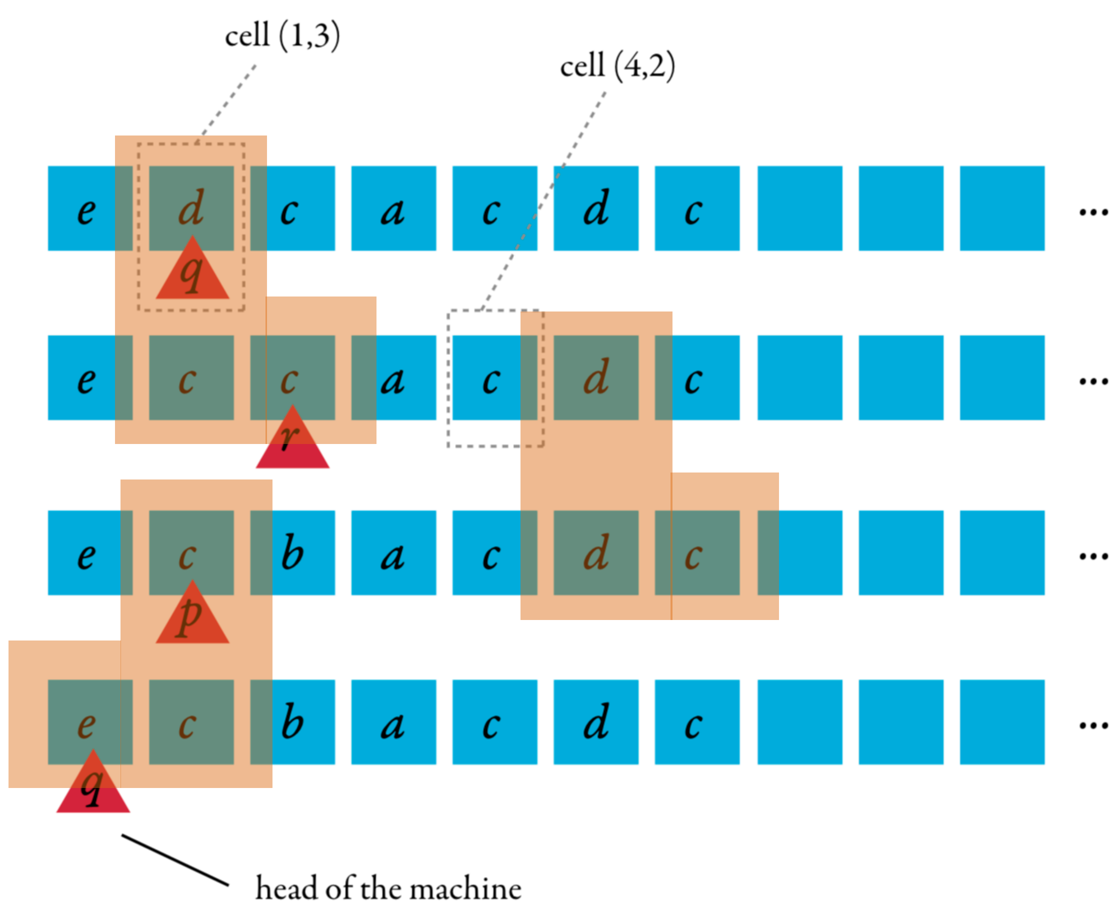
4. P ≠ NP when alphabet = atoms

Every orbit of Q or T is isomorphic to atoms$^{(n)}$ for n ≤ N

Consider the rejecting run on sufficiently long **independent** input word w

We fool M with a **dependent** input w' which M will forcedly reject too

The idea: use locality



4. P ≠ NP when alphabet = atoms

Every orbit of Q or T is isomorphic to atoms$^{(n)}$ for n ≤ N

Consider the rejecting run on sufficiently long **independent** input word w

We fool M with a **dependent** input w' which M will forcedly reject too

All subset of w have pairwise different sums

As the run is of polynomial length (w.r.t. length of w),
there are only polynomially many sums of $3N$ atoms appearing in it

w' := take a subset I of w whose sum is not among them, and
replace some arbitrary element a from I by r := the sum of $I \setminus \{a\}$

$a \longmapsto r$

Claim: $I \setminus \{a\} \cup \{r\}$ is the only subset of w' that sums up to 0

Claim: Every triple of elements of Q ∪ T in run(w) is in the same orbit as the corresponding triple in run(w')

$(a_1\, a_2 \ldots a_n), (b_1\, b_2 \ldots b_n) \in$ atoms$^{(n)}$ are in the same orbit

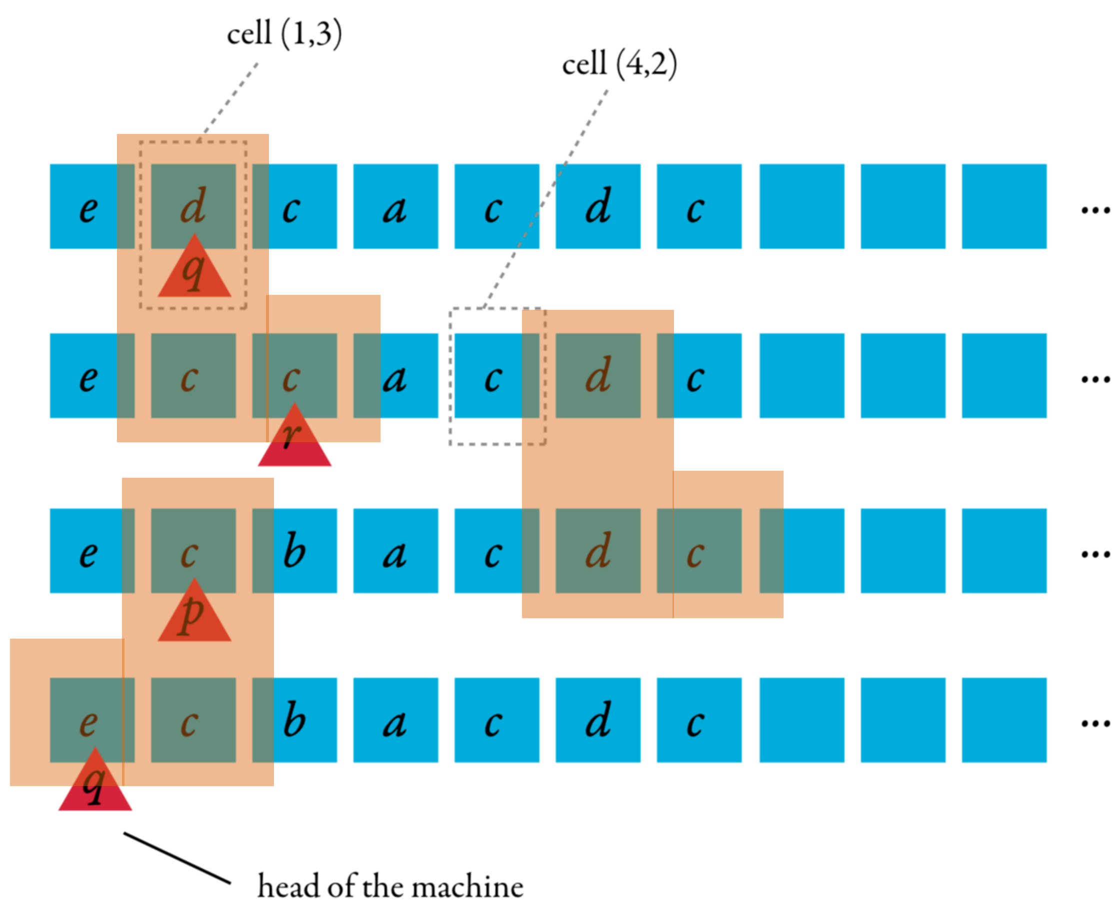
iff

$\sum_{i \in I} a_i = 0$ iff $\sum_{i \in I} b_i = 0$ for for every I ⊆ {1…n}

4. P ≠ NP when alphabet = atoms

Claim: Every triple of elements of Q ∪ T in run(*w*) is in the same orbit as the corresponding triple in run(*w'*)

Claim: run(*w*) is in the same orbit as run(*w'*), hence rejecting too



- automata with atoms
- Turing machines with atoms
- **other models of computation**

Pushdown automata

- alphabet A
- states Q
- stack alphabet S
- $\delta \subseteq Q \times (A \cup \{\varepsilon\}) \times S \times Q \times S^*$
- $I, F \subseteq Q$

orbit-finite sets instead of finite ones

Configurations = $Q \times S^*$

S^*

Deterministic pushdown automata: ...

Theorem: Pushdown automata = prefix-rewriting

Pushdown automata

recognized by a
nondeterministic
orbit-finite automaton

Theorem: Pre*(regular set) is regular for pushdown automata, and may be effectively computed

Corollary: Emptiness of pushdown automata is decidable

Context-free grammars

- nonterminal symbols S
- terminal symbols A
- an initial symbol
- $\delta \subseteq S \times (S \cup A)^*$

} orbit-finite sets instead of finite ones

Theorem: Context-free grammars = pushdown automata

Examples

- a context-free language over 3 atoms
- palindroms

$$S \longrightarrow a\, S\, a \qquad (a \in \text{atoms})$$
$$S \longrightarrow \varepsilon$$

- bracket expressions with brackets $(_a \;\;)_a$ for $a \in$ atoms

any well-behaved atoms

- monotonic bracket expressions ?

total order atoms $(Q, <)$

$$S \longrightarrow (_a \; \underline{a} \;)_a \qquad (a \in \text{atoms})$$
$$\underline{a} \longrightarrow (_b \; \underline{b} \;)_b \qquad (a,b \in \text{atoms},\ a < b)$$
$$\underline{a} \longrightarrow \underline{b}\; \underline{c} \qquad (a,b,c \in \text{atoms},\ a < b,c)$$
$$\underline{a} \longrightarrow \varepsilon \qquad (a \in \text{atoms})$$

Petri nets

- places P
- an initial configuration
- $\delta \subseteq M_{fin}(P) \times M_{fin}(P)$

} orbit-finite sets instead of finite ones

Configurations = finite multisets of places $M_{fin}(P)$

places = atoms × (finite set)

classical sets	sets with equality atoms (N, =)
general Petri nets	elementary nets
data Petri nets	general Petri nets