Computation with atoms

Sławomir Lasota
University of Warsaw

joint work with
Mikołaj Bojańczyk, Bartek Klin, Joanna Ochremiak, Szymon Toruńczyk

LSV, ENS Cachan, 2014.07.08

Outline

- Sets with atoms
- Models of computation in sets with atoms
- Are sets with atoms useful?

Sets with atoms

Sets with atoms

sets with urelements
permutation models
[Fraenkel, Mostowski '30ies]

Sets with atoms

sets with urelements
permutation models
[Fraenkel, Mostowski '30ies]

nominal sets
[Gabbay, Pitts '99]

named sets
[Pistore, Montanari '97]

Sets with atoms

sets with urelements
permutation models
[Fraenkel, Mostowski '30ies]

nominal G-sets
[Bojańczyk, Klin, L. '11]

nominal sets
[Gabbay, Pitts '99]

named sets
[Pistore, Montanari '97]

Sets with atoms

sets with urelements
permutation models
[Fraenkel, Mostowski '30ies]

nominal G-sets
[Bojańczyk, Klin, L. '11]

sets with symmetry

nominal sets
[Gabbay, Pitts '99]

hereditarily finitely-supported sets

named sets
[Pistore, Montanari '97]

Fraenkel-Mostowski sets

sets with atoms

Atoms allow us to model:

Atoms allow us to model:

- data values

Atoms allow us to model:

- data values
- names

Atoms allow us to model:

- data values
- names
- process ids

Atoms allow us to model:

- data values
- names
- process ids
- time-stamps

Atoms allow us to model:

- data values
- names
- process ids
- time-stamps
- numerical values

Atoms allow us to model:

- data values
- names
- process ids
- time-stamps
- numerical values
- ....

Atoms

Atoms

Atoms are a fixed logical structure

Atoms

Atoms are a fixed logical structure

atoms	atom automorphisms

Atoms

Atoms are a fixed logical structure

atoms	atom automorphisms
equality atoms (N, =)	all bijections

Atoms

Atoms are a fixed logical structure

atoms	atom automorphisms
equality atoms (N, =)	all bijections
total order atoms (Q, <)	monotonic bijections

Atoms

Atoms are a fixed logical structure

atoms	atom automorphisms
equality atoms (N, =)	all bijections
total order atoms (Q, <)	monotonic bijections
integer atoms (Z, <)	translations

Atoms

Atoms are a fixed logical structure

atoms	atom automorphisms
equality atoms (N, =)	all bijections
total order atoms (Q, <)	monotonic bijections
integer atoms (Z, <)	translations
(Q, <, +1)	monotonic bijections preserving integer differences

Atoms

Atoms are a fixed logical structure

atoms	atom automorphisms
equality atoms (N, =)	all bijections
total order atoms (Q, <)	monotonic bijections
integer atoms (Z, <)	translations
(Q, <, +1)	monotonic bijections preserving integer differences
∅	trivial

Atoms

Atoms are a fixed logical structure

atoms	atom automorphisms
equality atoms (N, =)	all bijections
total order atoms (Q, <)	monotonic bijections
integer atoms (Z, <)	translations
(Q, <, +1)	monotonic bijections preserving integer differences
∅	trivial
...	...

Atoms

Atoms are a fixed logical structure

atoms	atom automorphisms
equality atoms (N, =)	all bijections
total order atoms (Q, <)	monotonic bijections
integer atoms (Z, <)	translations
(Q, <, +1)	monotonic bijections preserving integer differences
∅	trivial
...	...

Atoms are a parameter in the following

Sets with atoms

Sets with atoms

Classical sets are built using ∅ and { }

e.g. {∅, {∅}, {{∅}}, {∅,{∅}}}

Sets with atoms

Classical sets are built using $\varnothing$ and { }

e.g. $\{\varnothing, \{\varnothing\}, \{\{\varnothing\}\}, \{\varnothing,\{\varnothing\}\}\}$

Sets with atoms are built using $\varnothing$ and { } and atoms

Sets with atoms

Classical sets are built using $\varnothing$ and { }

e.g. $\{\varnothing, \{\varnothing\}, \{\{\varnothing\}\}, \{\varnothing,\{\varnothing\}\}\}$

an atom contains no elements

Sets with atoms are built using $\varnothing$ and { } and atoms

Sets with atoms

Classical sets are built using ∅ and { }

e.g. {∅, {∅}, {{∅}}, {∅,{∅}}}

an atom contains no elements

Sets with atoms are built using ∅ and { } and atoms

Examples: • ∅

Sets with atoms

Classical sets are built using ∅ and { }

e.g. {∅, {∅}, {{∅}}, {∅,{∅}}}

an atom contains no elements

Sets with atoms are built using ∅ and { } and atoms

Examples:
- ∅
- three atoms {a, b, c},

Sets with atoms

Classical sets are built using ∅ and { }

e.g. {∅, {∅}, {{∅}}, {∅,{∅}}}

an atom contains no elements

Sets with atoms are built using ∅ and { } and atoms

Examples:
- ∅
- three atoms {a, b, c},
- a pair (a, b) of atoms, encoded eg. as {a, {a,b}}

Sets with atoms

Classical sets are built using ∅ and { }

e.g. {∅, {∅}, {{∅}}, {∅,{∅}}}

an atom contains no elements

Sets with atoms are built using ∅ and { } and atoms

Examples:

- ∅
- three atoms {a, b, c},
- a pair (a, b) of atoms, encoded eg. as {a, {a,b}}
- atoms∖{a, b, c}

Sets with atoms

Classical sets are built using ∅ and { }

e.g. {∅, {∅}, {{∅}}, {∅,{∅}}}

Sets with atoms are built using ∅ and { } and atoms

an atom contains no elements

Examples:
- ∅
- three atoms {a, b, c},
- a pair (a, b) of atoms, encoded eg. as {a, {a,b}}
- atoms ∖ {a, b, c}
- ordered pairs of atoms

Sets with atoms

Classical sets are built using ∅ and { }

e.g. {∅, {∅}, {{∅}}, {∅,{∅}}}

an atom contains no elements

Sets with atoms are built using ∅ and { } and atoms

Examples:

- ∅
- three atoms {a, b, c},
- a pair (a, b) of atoms, encoded eg. as {a, {a,b}}
- atoms∖{a, b, c}
- ordered pairs of atoms
- finite words over atoms

Sets with atoms

Classical sets are built using ∅ and { }

e.g. {∅, {∅}, {{∅}}, {∅,{∅}}}

an atom contains no elements

Sets with atoms are built using ∅ and { } and atoms

Examples:

- ∅
- three atoms {a, b, c},
- a pair (a, b) of atoms, encoded eg. as {a, {a,b}}
- atoms ∖ {a, b, c}
- ordered pairs of atoms
- finite words over atoms
- finite subsets of atoms

Sets with atoms

Classical sets are built using ∅ and { }

e.g. {∅, {∅}, {{∅}}, {∅,{∅}}}

an atom contains no elements

Sets with atoms are built using ∅ and { } and atoms

Examples:
- ∅
- three atoms {a, b, c},
- a pair (a, b) of atoms, encoded eg. as {a, {a,b}}
- atoms ∖ {a, b, c}
- ordered pairs of atoms
- finite words over atoms
- finite subsets of atoms
- all subsets of atoms

Sets with atoms

Classical sets are built using ∅ and { }

e.g. {∅, {∅}, {{∅}}, {∅,{∅}}}

an atom contains no elements

Sets with atoms are built using ∅ and { } and atoms

Examples:

- ∅
- three atoms {a, b, c},
- a pair (a, b) of atoms, encoded eg. as {a, {a,b}}
- atoms ∖ {a, b, c}
- ordered pairs of atoms
- finite words over atoms
- finite subsets of atoms
- all subsets of atoms
- ....

Sets with atoms

Classical sets are built using ∅ and { }

e.g. {∅, {∅}, {{∅}}, {∅,{∅}}}

an atom contains no elements

Sets with atoms are built using ∅ and { } and atoms

Examples:

- ∅
- three atoms {a, b, c},
- a pair (a, b) of atoms, encoded eg. as {a, {a,b}}
- atoms ∖ {a, b, c}
- ordered pairs of atoms
- finite words over atoms
- finite subsets of atoms
- ~~all subsets of atoms~~
- ....

illegal for (N, =)

Sets with atoms

Classical sets are built using ∅ and { }

e.g. {∅, {∅}, {{∅}}, {∅,{∅}}}

an atom contains no elements

Sets with atoms are built using ∅ and { } and atoms

Examples:
- ∅
- three atoms {a, b, c},
- a pair (a, b) of atoms, encoded eg. as {a, {a,b}}
- atoms ∖ {a, b, c}
- ordered pairs of atoms
- finite words over atoms
- finite subsets of atoms
- ~~all subsets of atoms~~
- .... illegal for (N, =)

legality depends on atom automorphisms

Legal sets with atoms

Legal sets with atoms

Extend atom automorphisms π to all sets element-wise,
e.g. $\pi(\{a, b, c\}) = \{\pi(a), \pi(b), \pi(c)\}$

Legal sets with atoms

Extend atom automorphisms π to all sets element-wise,
e.g. $\pi(\{a, b, c\}) = \{\pi(a), \pi(b), \pi(c)\}$

A set X is supported by a finite set S of atoms, if
every atom automorphism that is identity on S, preserves X:
$(\pi(a) = a$ for all $a \in S)$ implies $\pi(X) = X$

Legal sets with atoms

Extend atom automorphisms π to all sets element-wise,
e.g. $\pi(\{a, b, c\}) = \{\pi(a), \pi(b), \pi(c)\}$

A set X is supported by a finite set S of atoms, if
every atom automorphism that is identity on S, preserves X:
$(\pi(a) = a \text{ for all } a \in S)$ implies $\pi(X) = X$

A set X is legal if it is hereditarily finitely supported:

Legal sets with atoms

Extend atom automorphisms π to all sets element-wise,
e.g. $\pi(\{a, b, c\}) = \{\pi(a), \pi(b), \pi(c)\}$

A set X is supported by a finite set S of atoms, if
every atom automorphism that is identity on S, preserves X:
($\pi(a) = a$ for all $a \in S$) implies $\pi(X) = X$

A set X is legal if it is hereditarily finitely supported:

- X is finitely supported,

Legal sets with atoms

Extend atom automorphisms π to all sets element-wise,
e.g. $\pi(\{a, b, c\}) = \{\pi(a), \pi(b), \pi(c)\}$

A set X is supported by a finite set S of atoms, if
every atom automorphism that is identity on S, preserves X:
$(\pi(a) = a$ for all $a \in S)$ implies $\pi(X) = X$

A set X is legal if it is hereditarily finitely supported:

- X is finitely supported,
- its elements are finitely supported,

Legal sets with atoms

Extend atom automorphisms π to all sets element-wise,
e.g. $\pi(\{a, b, c\}) = \{\pi(a), \pi(b), \pi(c)\}$

A set X is supported by a finite set S of atoms, if
every atom automorphism that is identity on S, preserves X:
$(\pi(a) = a$ for all $a \in S)$ implies $\pi(X) = X$

A set X is legal if it is hereditarily finitely supported:

- X is finitely supported,
- its elements are finitely supported,
- and so on...

Legal sets with atoms

Extend atom automorphisms π to all sets element-wise,
e.g. $\pi(\{a, b, c\}) = \{\pi(a), \pi(b), \pi(c)\}$

A set X is supported by a finite set S of atoms, if
every atom automorphism that is identity on S, preserves X:
$(\pi(a) = a$ for all $a \in S)$ implies $\pi(X) = X$

A set X is legal if it is hereditarily finitely supported:

- X is finitely supported,
- its elements are finitely supported,
- and so on...

Sets supported by ∅ are called equivariant

Examples:

Examples:

- $\varnothing$
- three atoms {a, b, c},
- a pair (a, b) of atoms,

- $\varnothing$
- {a, b, c},
- {a, b},

Examples:

- $\varnothing$
- three atoms $\{a, b, c\}$,
- a pair (a, b) of atoms,
- atoms $\setminus \{a, b, c\}$

- $\varnothing$
- $\{a, b, c\}$,
- $\{a, b\}$,
- $\{a, b, c\}$

Examples:

- $\emptyset$
- three atoms {a, b, c},
- a pair (a, b) of atoms,
- atoms $\setminus$ {a, b, c}

- $\emptyset$
- {a, b, c},
- {a, b},
- {a, b, c}

support = atoms that you use in order to "define" a set

Examples:

- $\emptyset$
- three atoms {a, b, c},
- a pair (a, b) of atoms,
- atoms $\setminus$ {a, b, c}
- ordered pairs of atoms
- finite words over atoms
- finite subsets of atoms
- all subsets of atoms

- $\emptyset$
- {a, b, c},
- {a, b},
- {a, b, c}
- $\emptyset$
- $\emptyset$
- $\emptyset$
- $\emptyset$

support = atoms that you use in order to ”define” a set

Sets with atoms

Sets with atoms

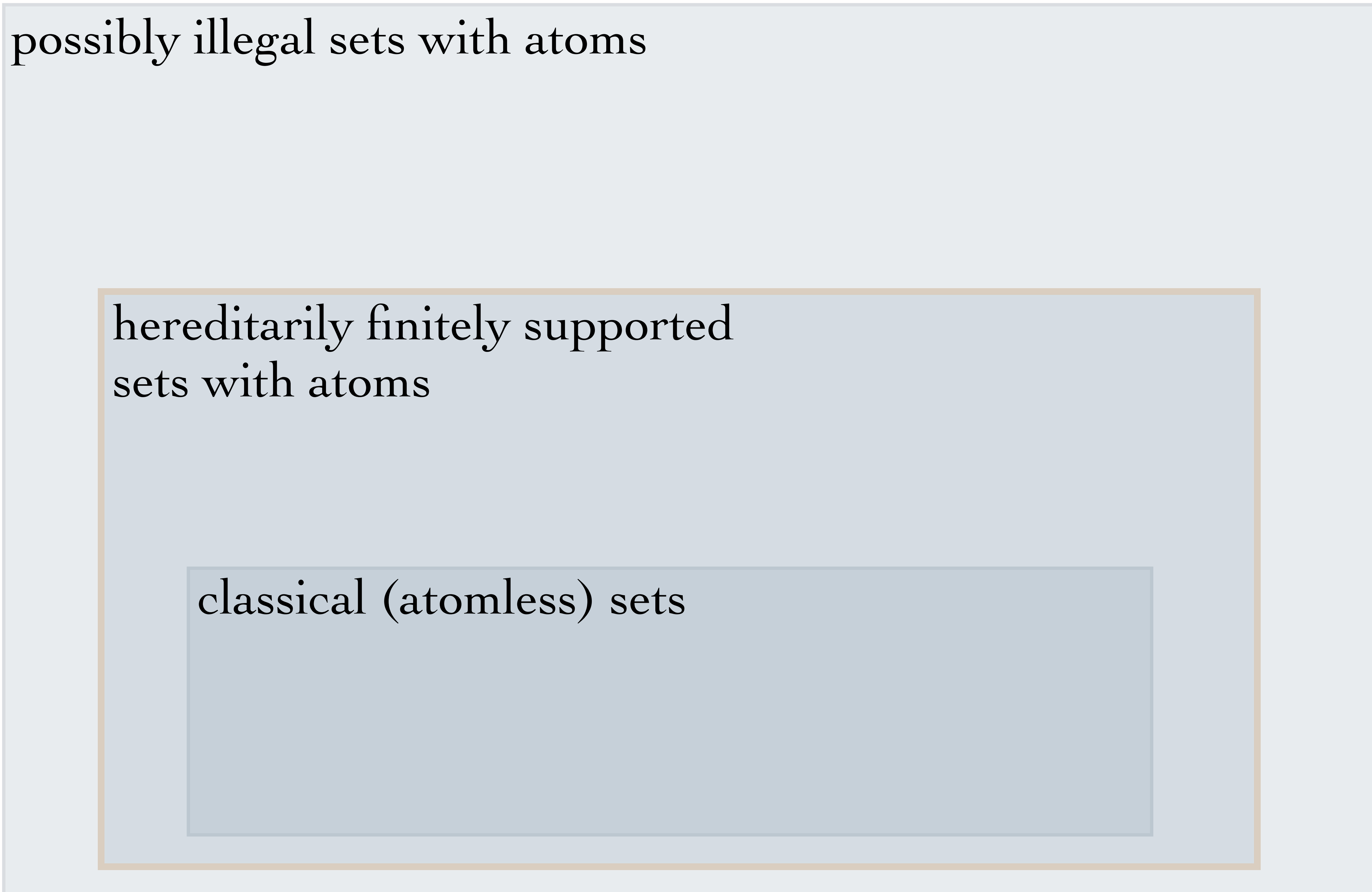
classical (atomless) sets

Sets with atoms

possibly illegal sets with atoms

classical (atomless) sets

Sets with atoms



relax finiteness to...

relax finiteness to...

...finiteness up to atom automorphism

Orbit-finite sets

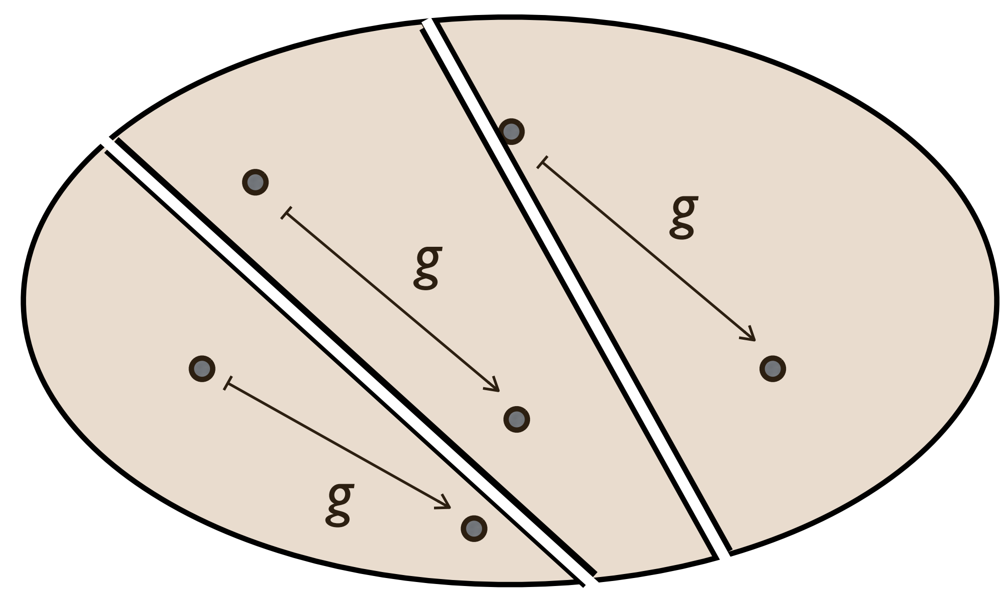
Orbit-finite sets

x, y are in the same orbit if $\pi(x) = y$ for an atom automorphism π

Orbit-finite sets

x, y are in the same orbit if $\pi(x) = y$ for an atom automorphism π

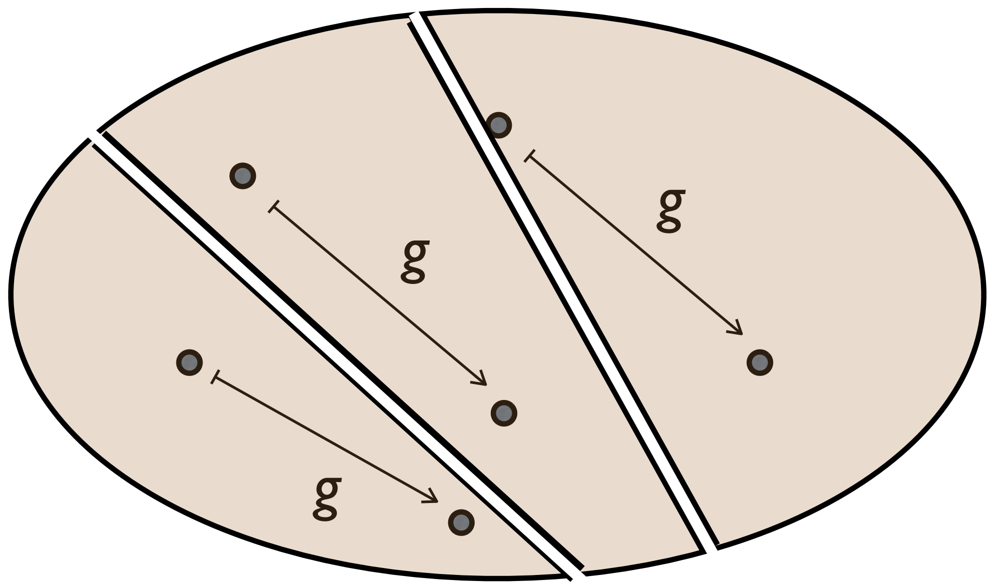
A set is orbit-finite if its partition into orbits is finite



Orbit-finite sets

x, y are in the same orbit if $\pi(x) = y$ for an atom automorphism π

A set is orbit-finite if its partition into orbits is finite



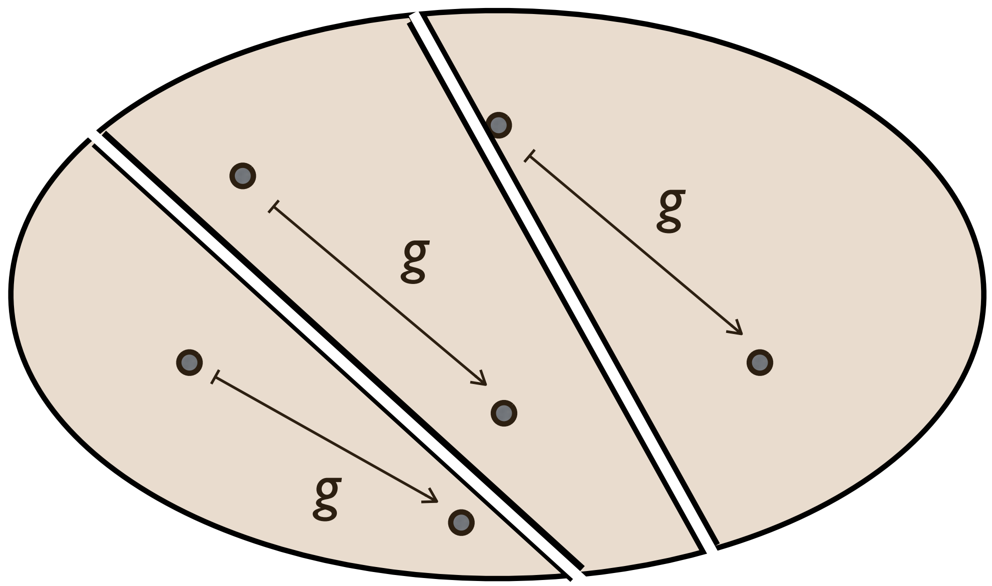
Examples:

- $\emptyset$
- three atoms {a, b, c},
- a pair (a, b) of atoms,
- atoms $\setminus$ {a, b, c}
- ordered pairs of atoms
- finite words over atoms
- finite subsets of atoms
- all subsets of atoms

Orbit-finite sets

x, y are in the same orbit if $\pi(x) = y$ for an atom automorphism π

A set is orbit-finite if its partition into orbits is finite



Examples:

- $\emptyset$
- three atoms {a, b, c},
- a pair (a, b) of atoms,

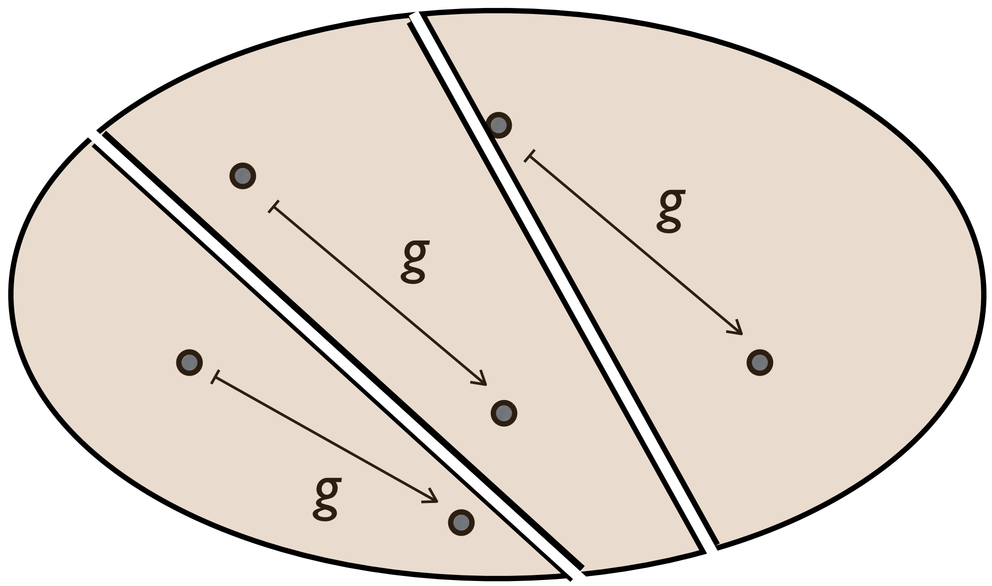
} finite

- atoms \ {a, b, c}
- ordered pairs of atoms
- finite words over atoms
- finite subsets of atoms
- all subsets of atoms

Orbit-finite sets

x, y are in the same orbit if $\pi(x) = y$ for an atom automorphism π

A set is orbit-finite if its partition into orbits is finite



Examples:

- $\emptyset$
- three atoms {a, b, c},
- a pair (a, b) of atoms,

} finite

- atoms $\setminus$ {a, b, c}
- ordered pairs of atoms

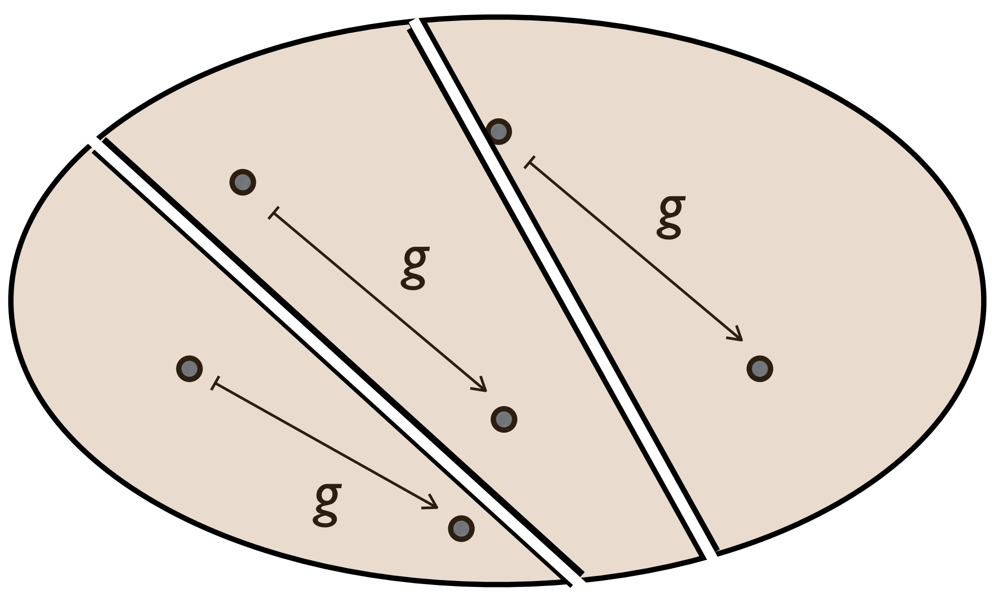
} orbit-finite typically

- finite words over atoms
- finite subsets of atoms
- all subsets of atoms

Orbit-finite sets

x, y are in the same orbit if π(x) = y for an atom automorphism π

A set is orbit-finite if its partition into orbits is finite



Examples:

- ∅
- three atoms {a, b, c},
- a pair (a, b) of atoms,

} finite

- atoms ∖ {a, b, c}
- ordered pairs of atoms

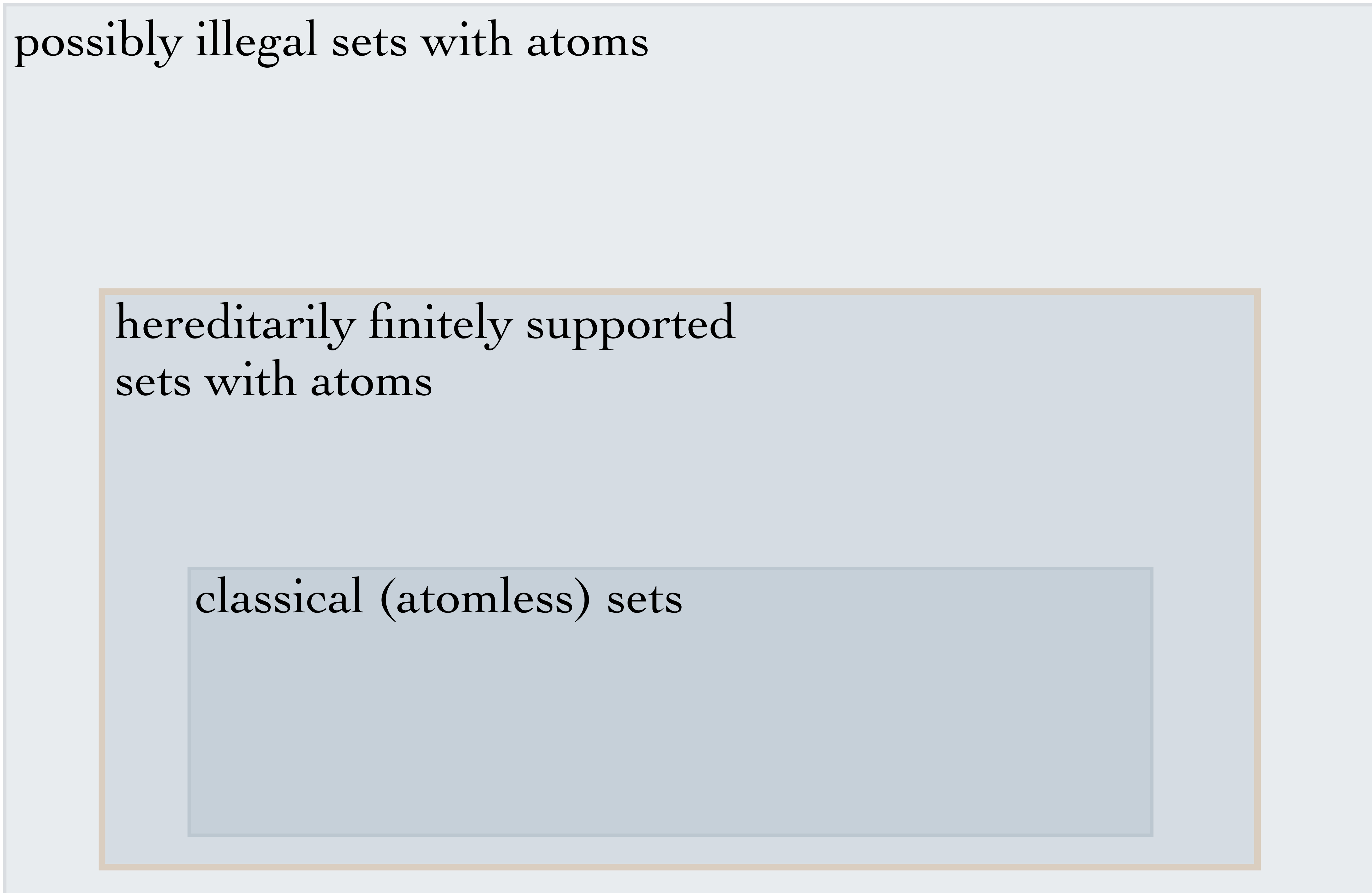
} orbit-finite typically

- finite words over atoms
- finite subsets of atoms

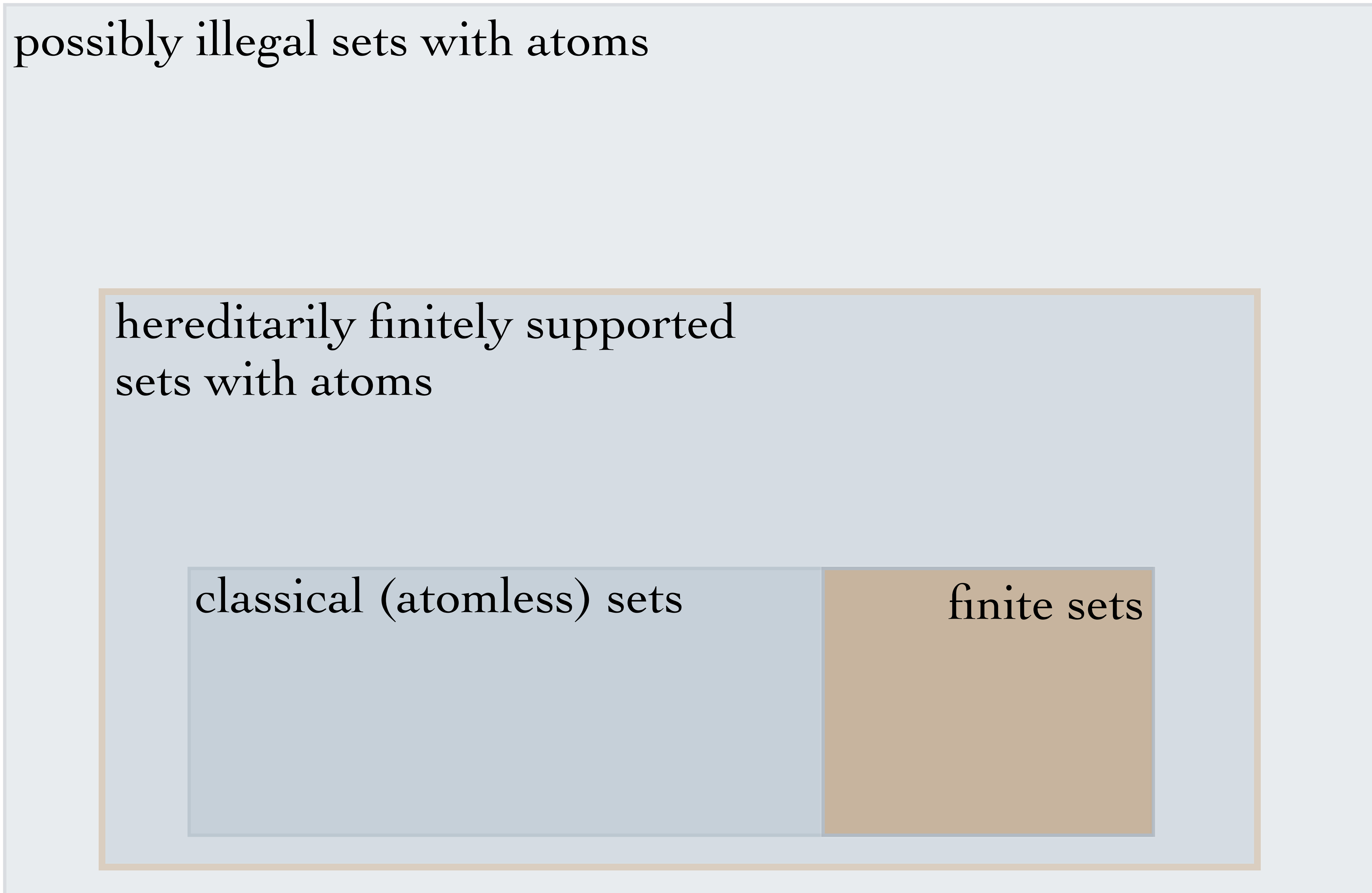
} orbit-infinite

- all subsets of atoms

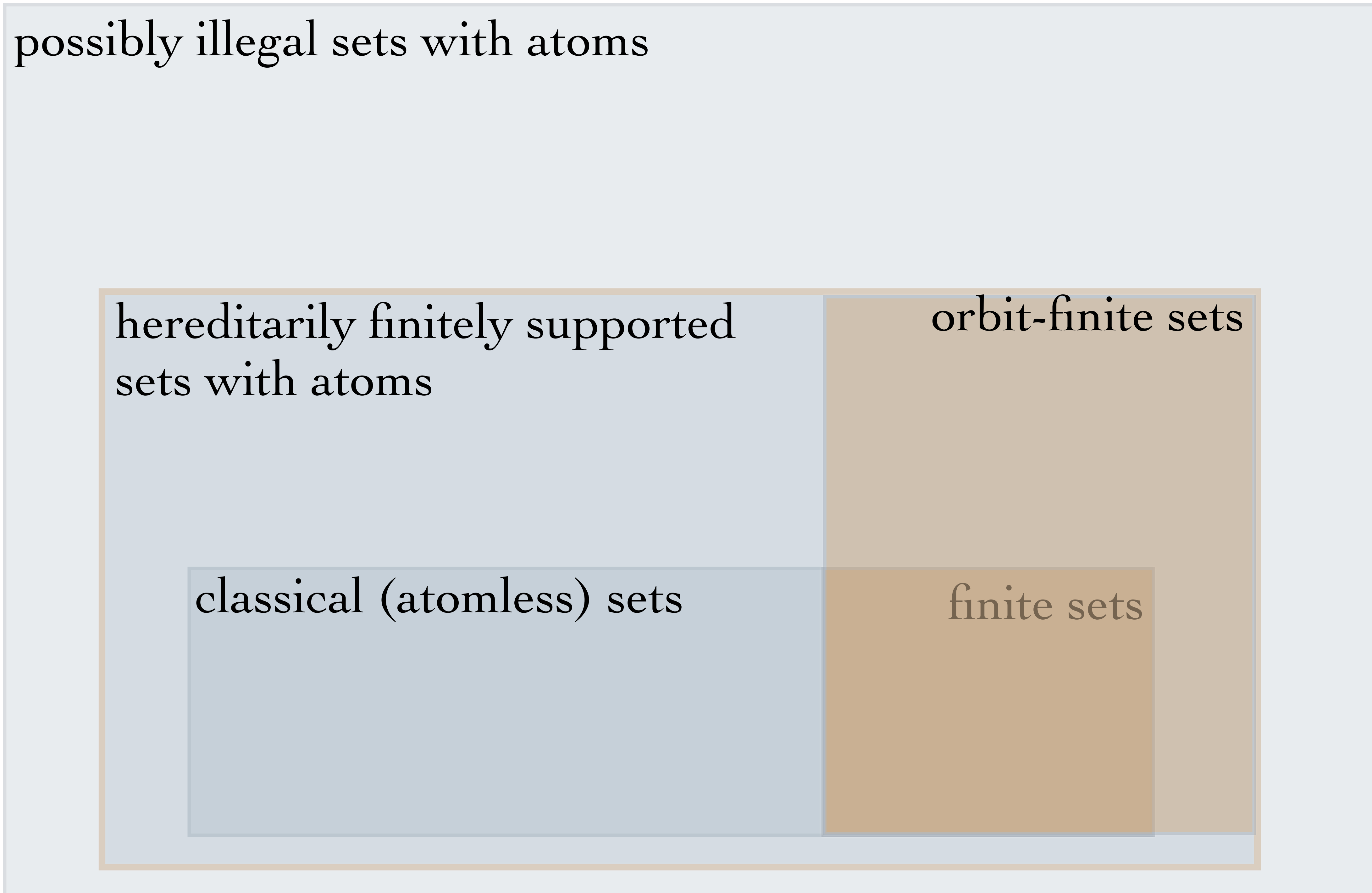
Orbit-finite sets with atoms



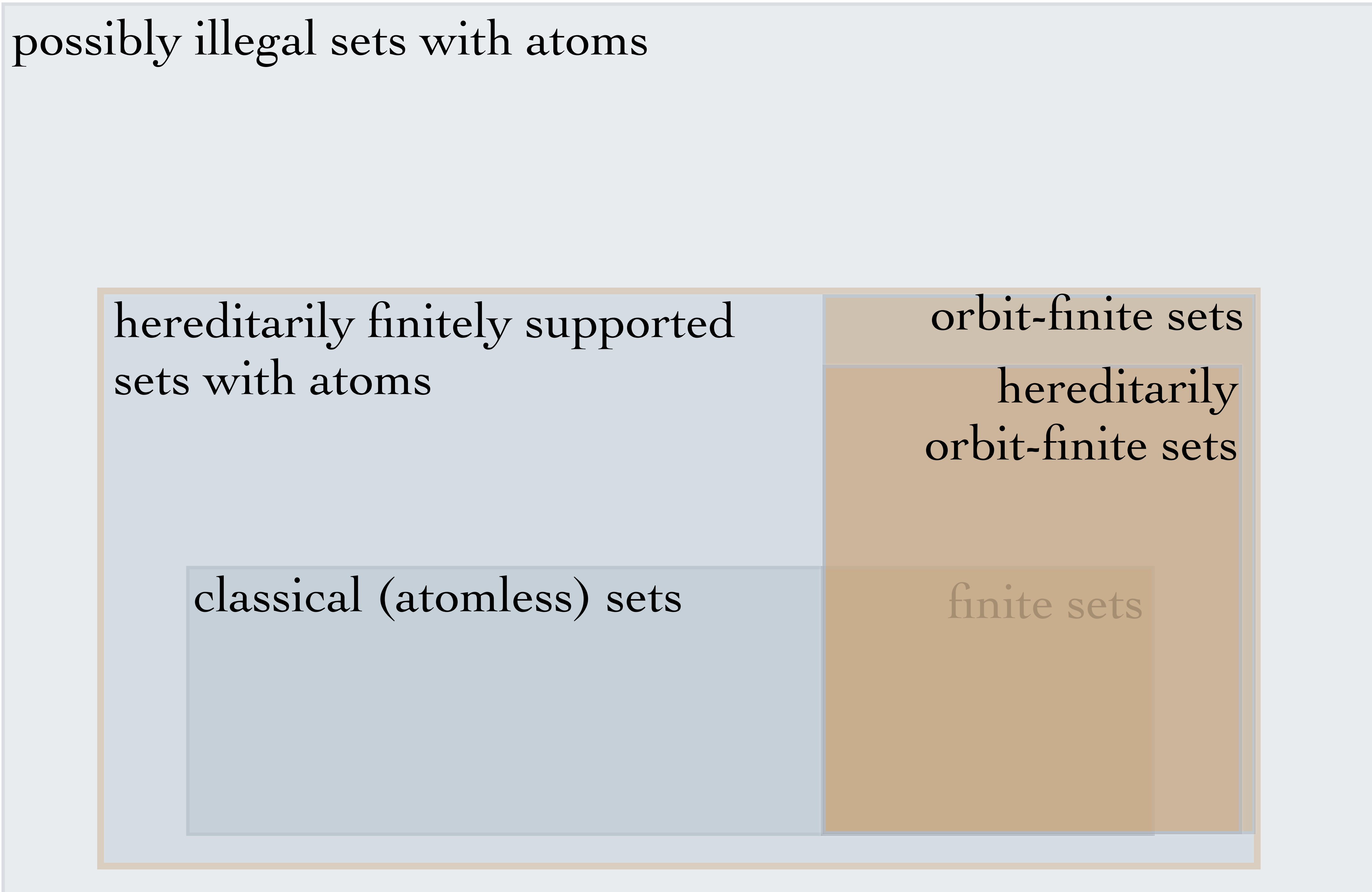
Orbit-finite sets with atoms



Orbit-finite sets with atoms



Orbit-finite sets with atoms



We've relaxed finiteness to orbit-finiteness.

We've relaxed finiteness to orbit-finiteness.

Are orbit-finite sets finitely representable?

Definable sets

Every 1-orbit set is isomorphic to an equivariant quotient of a 1-orbit subset of atoms^n, for some n.

Definable sets

Every 1-orbit set is isomorphic to an equivariant quotient of a 1-orbit subset of atoms^n, for some n.

this defines dimension of a 1-orbit set

Definable sets

Every 1-orbit set is isomorphic to an equivariant quotient of a 1-orbit subset of atoms^n, for some n.

this defines dimension of a 1-orbit set

When atoms are oligomorphic, i.e. atoms^n is orbit-finite for all n,

legal subsets of atoms^n = FO definable subsets of atoms^n

Definable sets

Every 1-orbit set is isomorphic to an equivariant quotient of a 1-orbit subset of atoms^n, for some n.

this defines dimension of a 1-orbit set

When atoms are oligomorphic, i.e. atoms^n is orbit-finite for all n,

legal subsets of atoms^n = FO definable subsets of atoms^n

When atoms are homogeneous and relational,

legal subsets of atoms^n = quantifier-free definable subsets of atoms^n

Definable sets

Examples:

Definable sets

Examples: $x1 = x2 \neq x3$

Definable sets

Examples: $x_1 = x_2 \neq x_3$

$x_1 < x_2 \leq x_3$

Definable sets

Examples:

$x_1 = x_2 \neq x_3$

$x_1 < x_2 \leq x_3$

$x_1 \, E \, x_2 \wedge \neg \, x_2 \, E \, x_3$

} equivariant

Definable sets

Examples:

$x_1 = x_2 \neq x_3$

$x_1 < x_2 \leq x_3$

$x_1 \; E \; x_2 \wedge \neg \; x_2 \; E \; x_3$

} equivariant

$x_1 < x_2 < 7$ supported by $\{7\}$

Homogeneous atoms

Homogeneous atoms

a relational structure $\mathcal{A}$ is homogeneous

if

Homogeneous atoms

a relational structure $\mathcal{A}$ is homogeneous

if

every isomorphism of finite induced substructures of $\mathcal{A}$ extends to an automorphism of the whole structure

Homogeneous atoms

a relational structure $\mathcal{A}$ is homogeneous

if

every isomorphism of finite induced substructures of $\mathcal{A}$ extends to an automorphism of the whole structure

Example: $(\mathbb{Q}, \leq)$

Homogeneous atoms

a relational structure $\mathcal{A}$ is homogeneous

if

every isomorphism of finite induced substructures of $\mathcal{A}$ extends to an automorphism of the whole structure

Example: $(\mathbb{Q}, \leq)$

Homogeneous atoms

a relational structure $\mathcal{A}$ is homogeneous

if

every isomorphism of finite induced substructures of $\mathcal{A}$ extends to an automorphism of the whole structure

Example: $(\mathbb{Q}, \leq)$

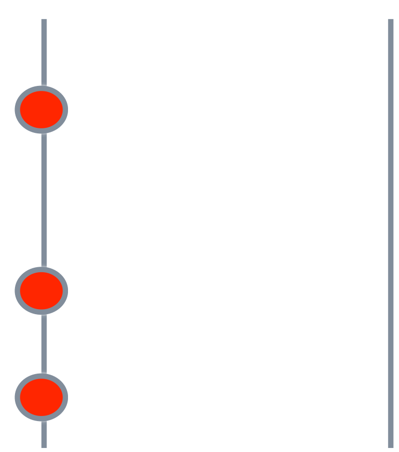
Homogeneous atoms

a relational structure $\mathcal{A}$ is homogeneous

if

every isomorphism of finite induced substructures of $\mathcal{A}$ extends to an automorphism of the whole structure

Example: $(Q, \leq)$



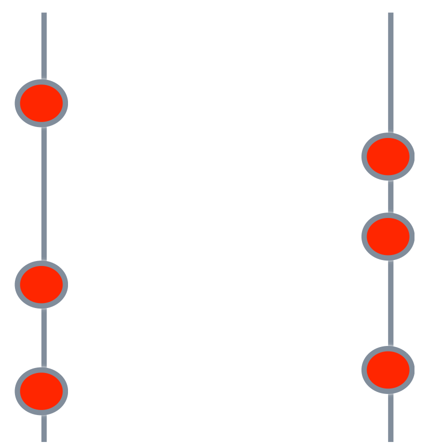
Homogeneous atoms

a relational structure $\mathcal{A}$ is homogeneous

if

every isomorphism of finite induced substructures of $\mathcal{A}$ extends to an automorphism of the whole structure

Example: $(\mathbb{Q}, \leq)$



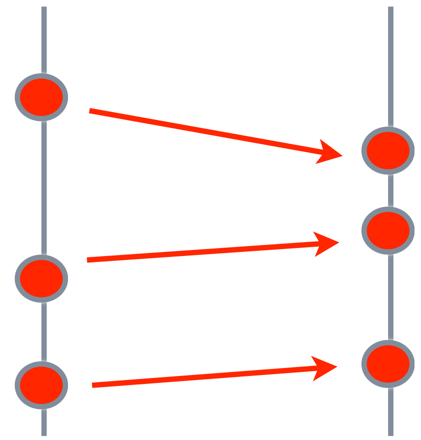
Homogeneous atoms

a relational structure $\mathcal{A}$ is homogeneous

if

every isomorphism of finite induced substructures of $\mathcal{A}$ extends to an automorphism of the whole structure

Example: $(Q, \leq)$



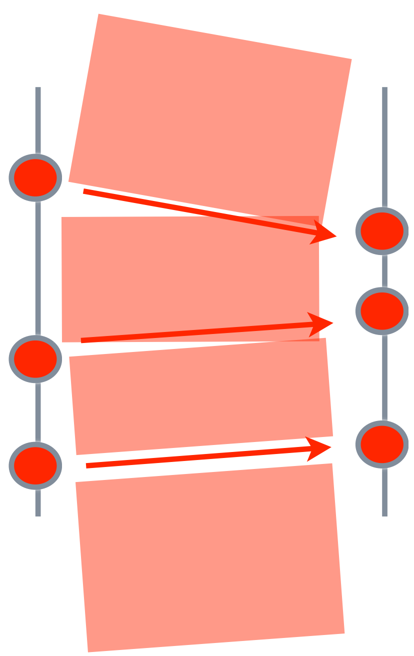
Homogeneous atoms

a relational structure $\mathcal{A}$ is homogeneous

if

every isomorphism of finite induced substructures of $\mathcal{A}$ extends to an automorphism of the whole structure

Example: $(\mathbb{Q}, \leq)$



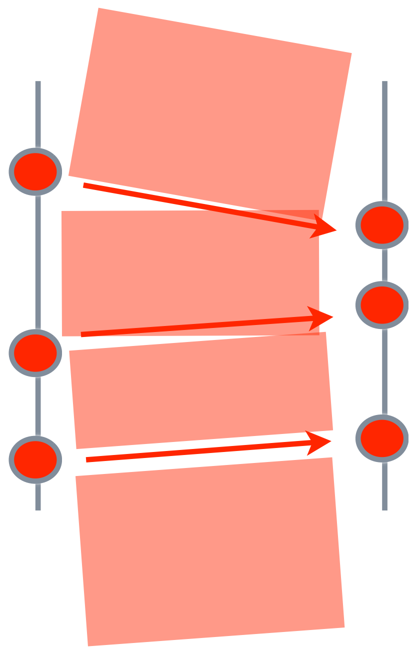
Homogeneous atoms

a relational structure $\mathcal{A}$ is homogeneous

if

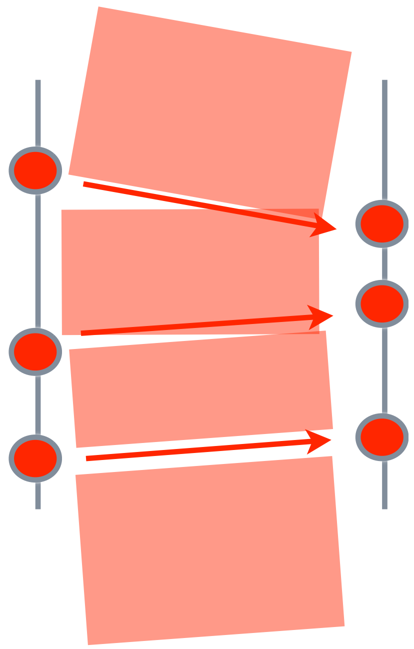
every isomorphism of finite induced substructures of $\mathcal{A}$ extends to an automorphism of the whole structure

Example: $(\mathbb{Q}, \leq)$

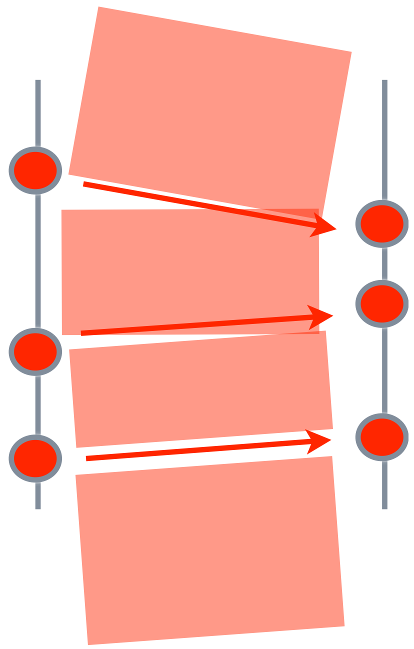


a homogeneous structure is uniquely determined by its finite induced substructures

Homogeneous atoms

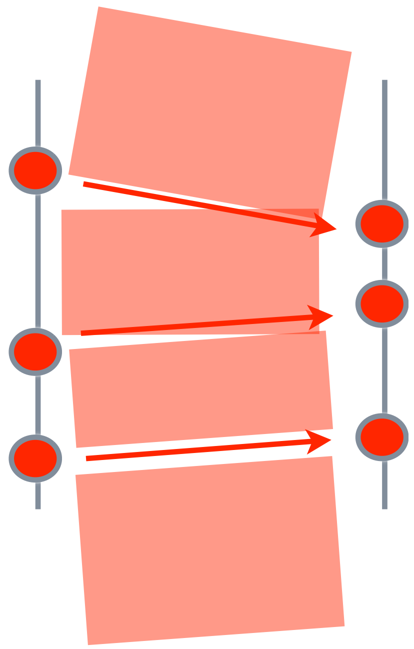


Homogeneous atoms



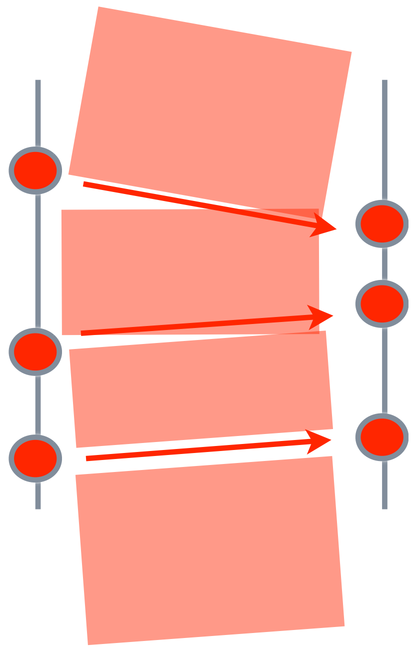
equality atoms (N, =)

Homogeneous atoms



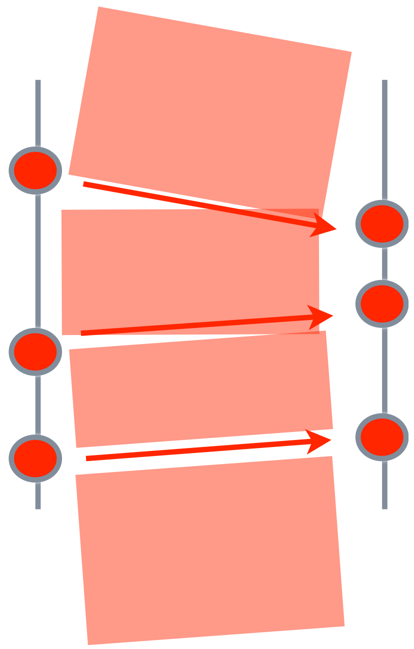
equality atoms (N, =)
total order atoms (Q, <)

Homogeneous atoms



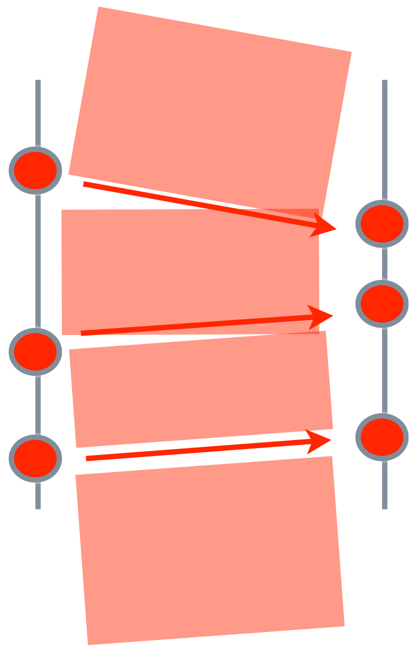
equality atoms (N, =)
total order atoms (Q, <)
integer atoms (Z, <)

Homogeneous atoms



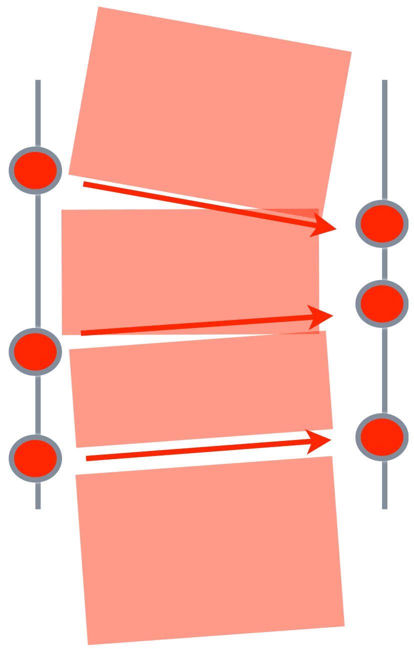
equality atoms (N, =)
total order atoms (Q, <)
~~integer atoms (Z, <)~~

Homogeneous atoms



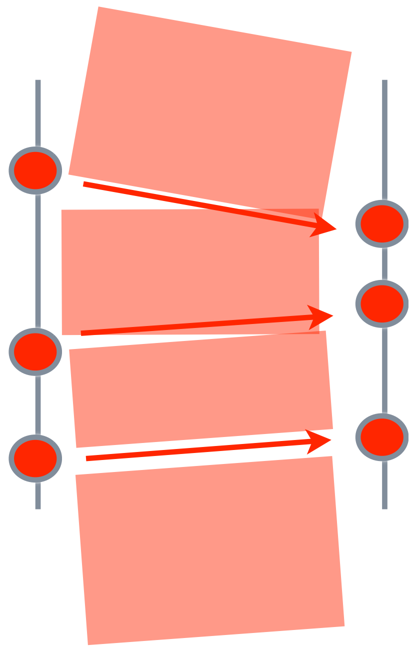
equality atoms (N, =)
total order atoms (Q, <)
~~integer atoms (Z, <)~~
(Q, <, +1)

Homogeneous atoms



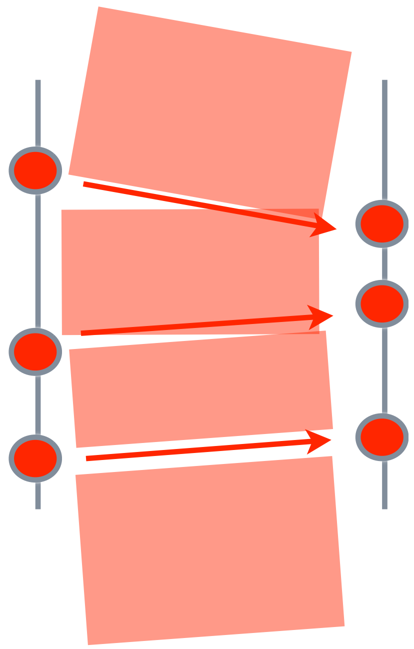
equality atoms (N, =)
total order atoms (Q, <)
~~integer atoms (Z, <)~~
~~(Q, <, +1)~~

Homogeneous atoms



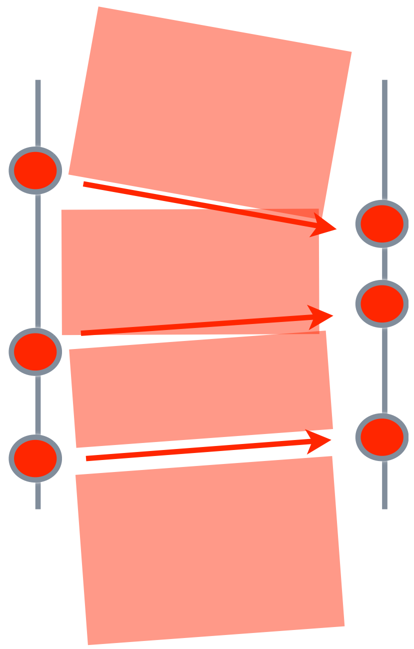
equality atoms (N, =)
total order atoms (Q, <)
~~integer atoms (Z, <)~~
~~(Q, <, +1)~~
∅

Homogeneous atoms



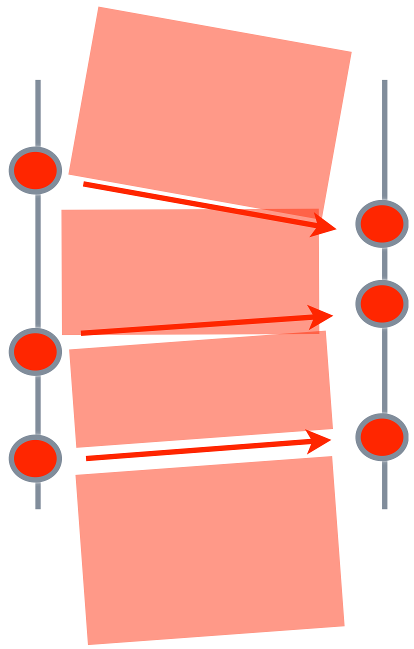
equality atoms (N, =)
total order atoms (Q, <)
~~integer atoms (Z, <)~~
~~(Q, <, +1)~~
∅
universal (random) graph

Homogeneous atoms



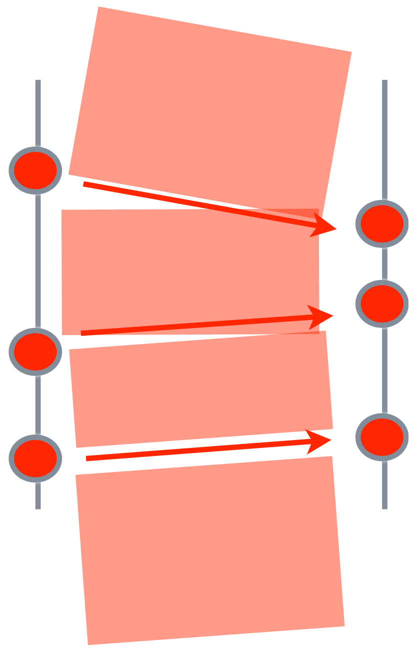
equality atoms (N, =)
total order atoms (Q, <)
~~integer atoms (Z, <)~~
~~(Q, <, +1)~~
∅
universal (random) graph
universal partial order

Homogeneous atoms



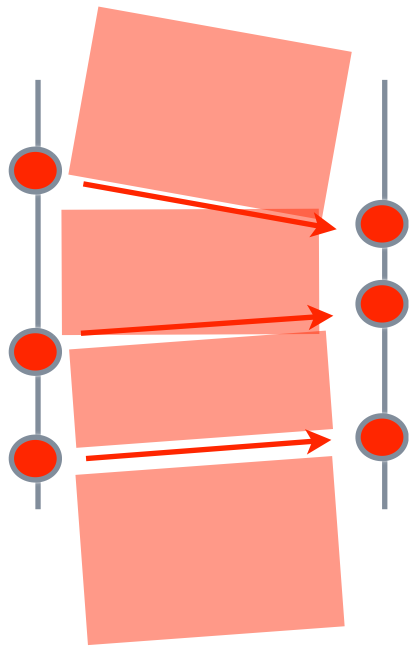
equality atoms $(\mathbb{N}, =)$
total order atoms $(\mathbb{Q}, <)$
~~integer atoms $(\mathbb{Z}, <)$~~
~~$(\mathbb{Q}, <, +1)$~~
$\emptyset$
universal (random) graph
universal partial order
universal equivalence relation

Homogeneous atoms



equality atoms (N, =)
total order atoms (Q, <)
~~integer atoms (Z, <)~~
~~(Q, <, +1)~~
∅
universal (random) graph
universal partial order
universal equivalence relation
universal tournament

Homogeneous atoms



equality atoms (N, =)
total order atoms (Q, <)
~~integer atoms (Z, <)~~
~~(Q, <, +1)~~
∅
universal (random) graph
universal partial order
universal equivalence relation
universal tournament
...

Atoms are assumed in the sequel to be oligomorphic and effective.

Outline

- Sets with atoms

- Models of computation in sets with atoms

- Are sets with atoms useful?

Automata

Automata

- alphabet A

Automata

- alphabet A

atoms $\times$ (a finite set)

Automata

atoms $\times$ (a finite set)

- alphabet A
- states Q

Automata

- alphabet A
- states Q

atoms × (a finite set)

$\text{atoms}^n \times (\text{a finite set})$

Automata

- alphabet A
- states Q
- $\delta \subseteq Q \times A \times Q$

Automata

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

Automata

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

} orbit-finite sets instead of finite ones

Automata

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

} orbit-finite sets instead of finite ones

Deterministic automata:

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

input alphabet: atoms

language: "exactly two different atoms appear"

states:

transitions:

initial state:

accepting states:

input alphabet: atoms

language: "exactly two different atoms appear"

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

transitions:

initial state:

accepting states:

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:

initial state:

accepting states:

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

initial state:

accepting states:

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

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

if in state () atom a is read, goto state (a)

initial state:

accepting states:

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)	$a \neq b$

if in state (a), an atom b ≠ a is read, goto state (ab)

initial state:

accepting states:

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

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

initial state:

accepting states:

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

$\delta((), a) =$	(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:

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

$\delta((), a) =$	(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:

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

$\delta((), a) =$	(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"

states:

transitions:

initial state:

accepting states:

input alphabet: atoms

language: "exactly two different atoms appear"

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

transitions:

initial state:

accepting states:

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:

initial state:

accepting states:

input alphabet: atoms

language: "exactly two different atoms appear"

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

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

initial state:

accepting states:

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

$\delta(\emptyset, a) = \{a\}$ $a \in$ atoms

if in state $\emptyset$ atom a is read, goto state {a}

initial state:

accepting states:

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

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

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

initial state:

accepting states:

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

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

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

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

if in state {a, b}, atom c ≠ a, b is read, reject

initial state:

accepting states:

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

$\delta(\varnothing, a) =$	$\{a\}$	$a \in$ atoms
$\delta(\{a\}, b) =$	$\{a, b\}$	$a, b \in$ atoms
$\delta(\{a, b\}, c) =$	reject	$c \neq a, b$

initial state: $\varnothing$

accepting states:

input alphabet: atoms

language: "exactly two different atoms appear"

registers are not necessarily ordered

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

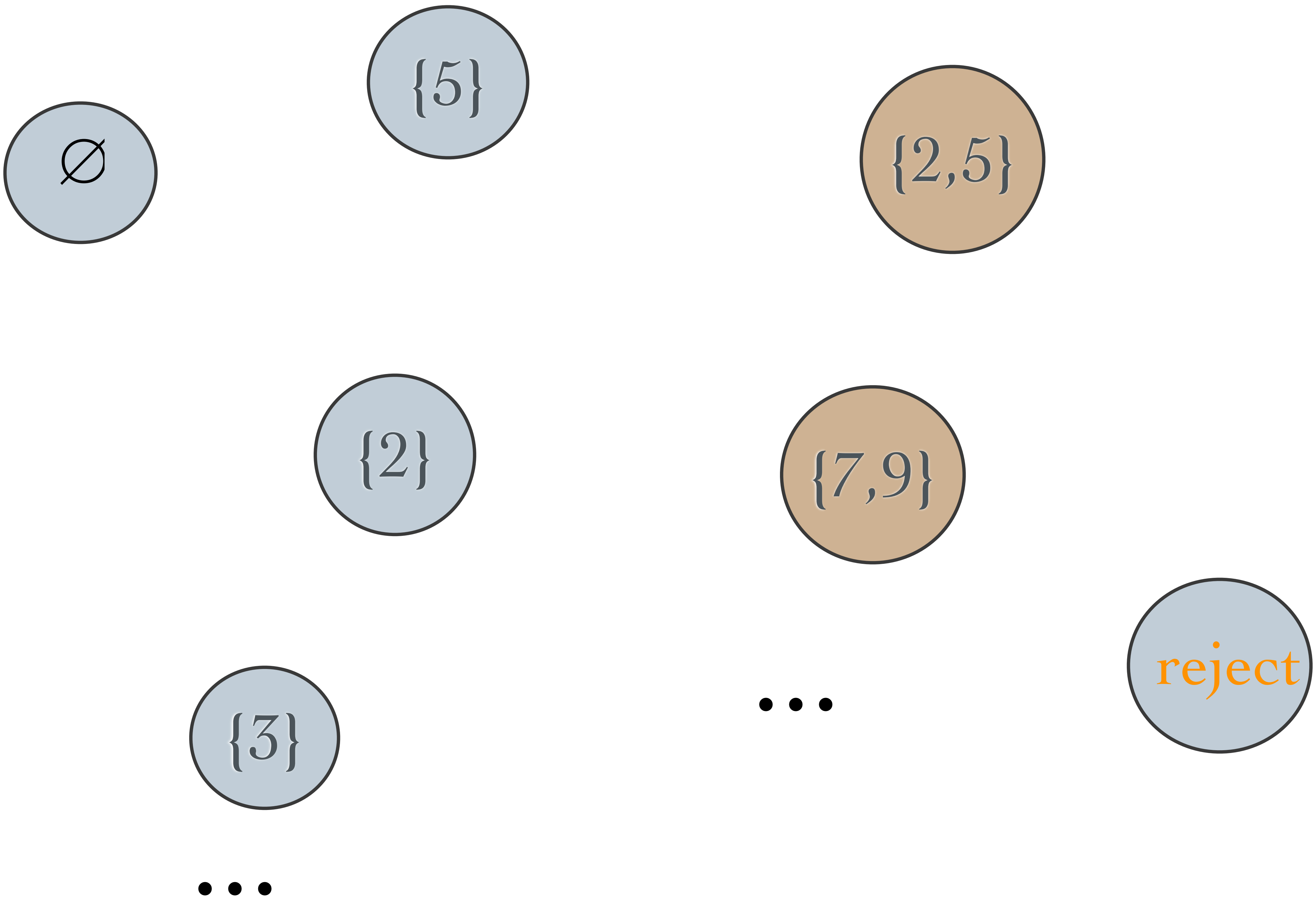
transitions: $\delta : Q \times A \to Q$

$\delta(\emptyset, a) =$	$\{a\}$	$a \in$ atoms
$\delta(\{a\}, b) =$	$\{a, b\}$	$a, b \in$ atoms
$\delta(\{a, b\}, c) =$	reject	$c \neq a, b$

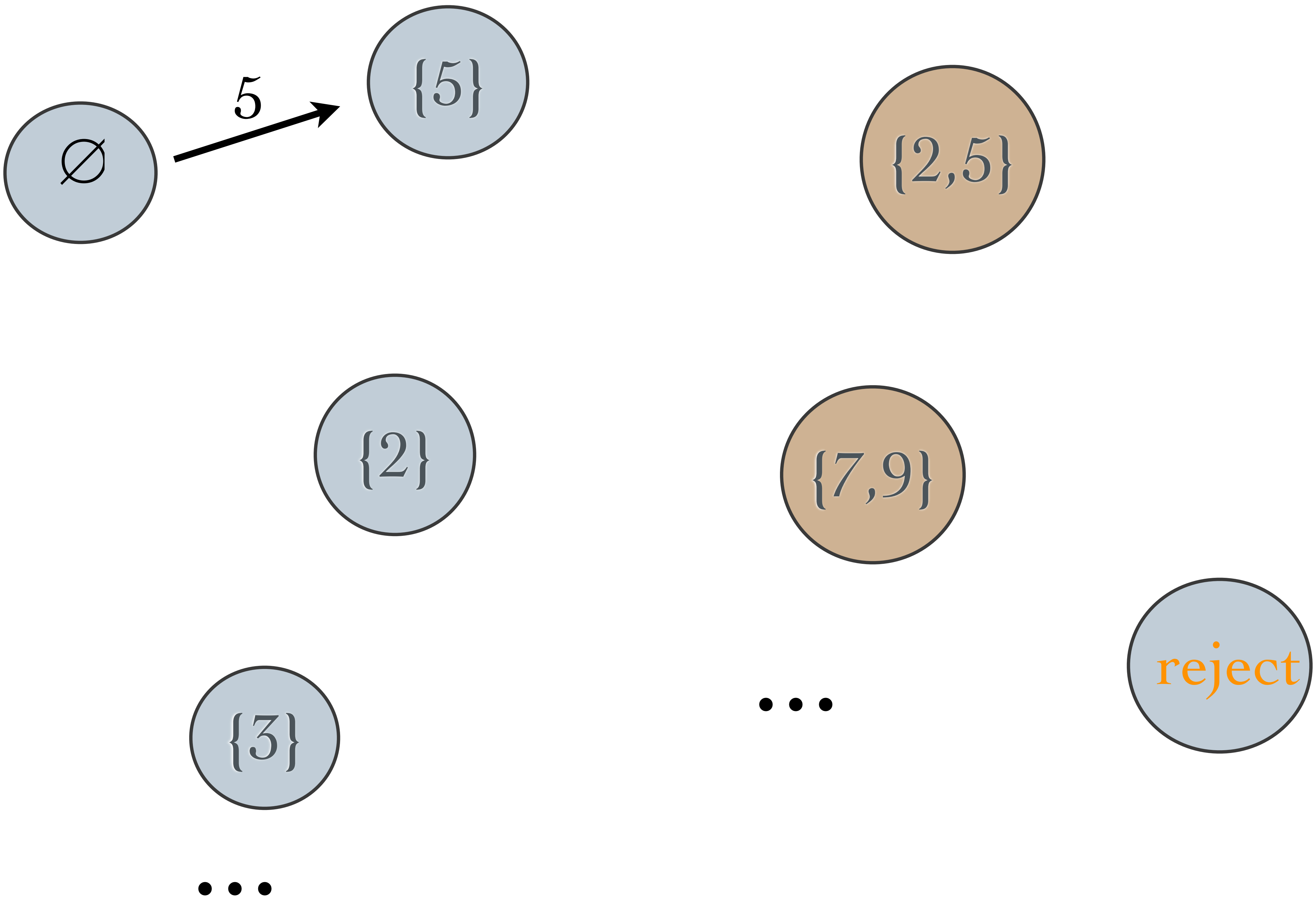
initial state: $\emptyset$

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

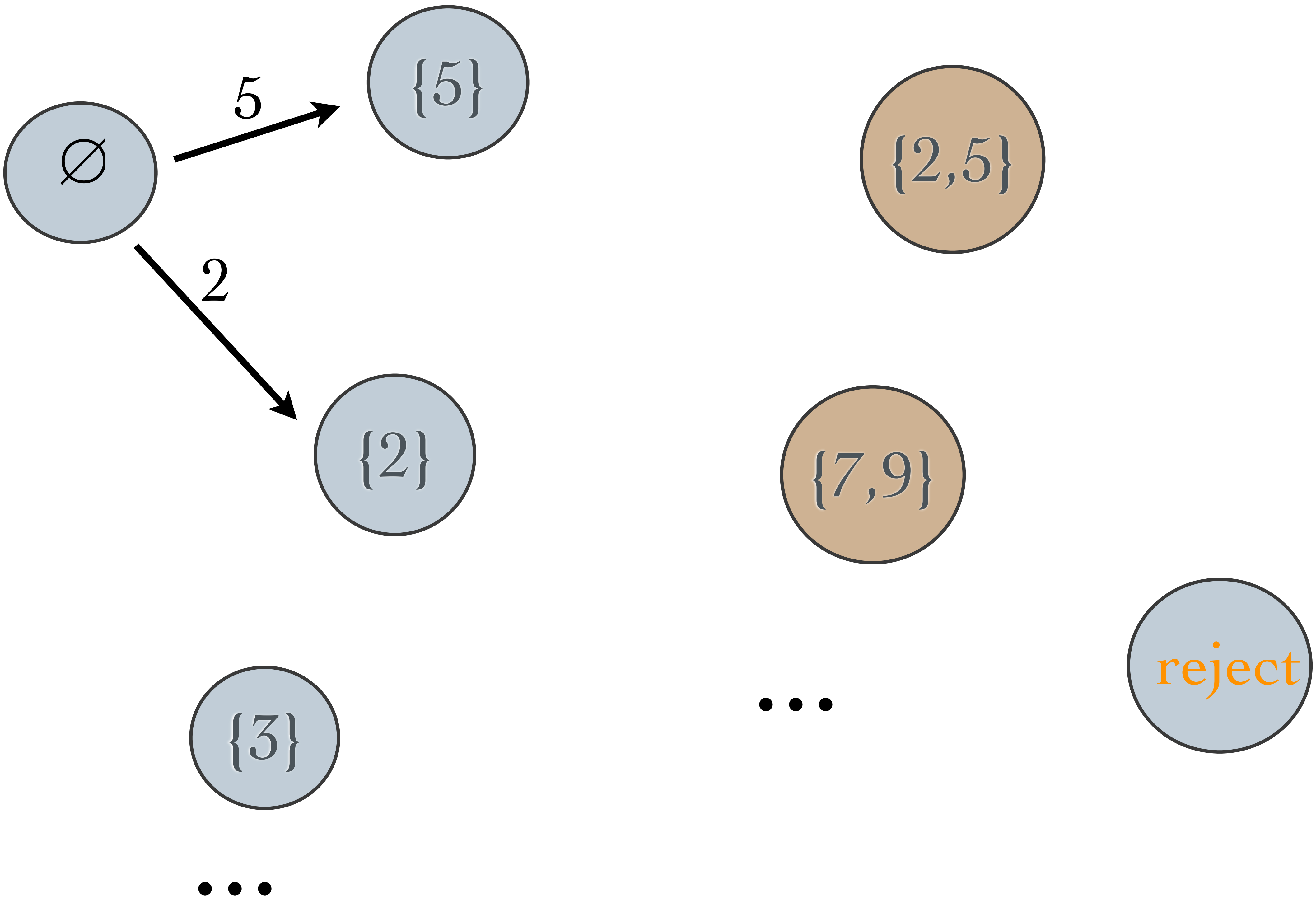
"exactly two different atoms appear"



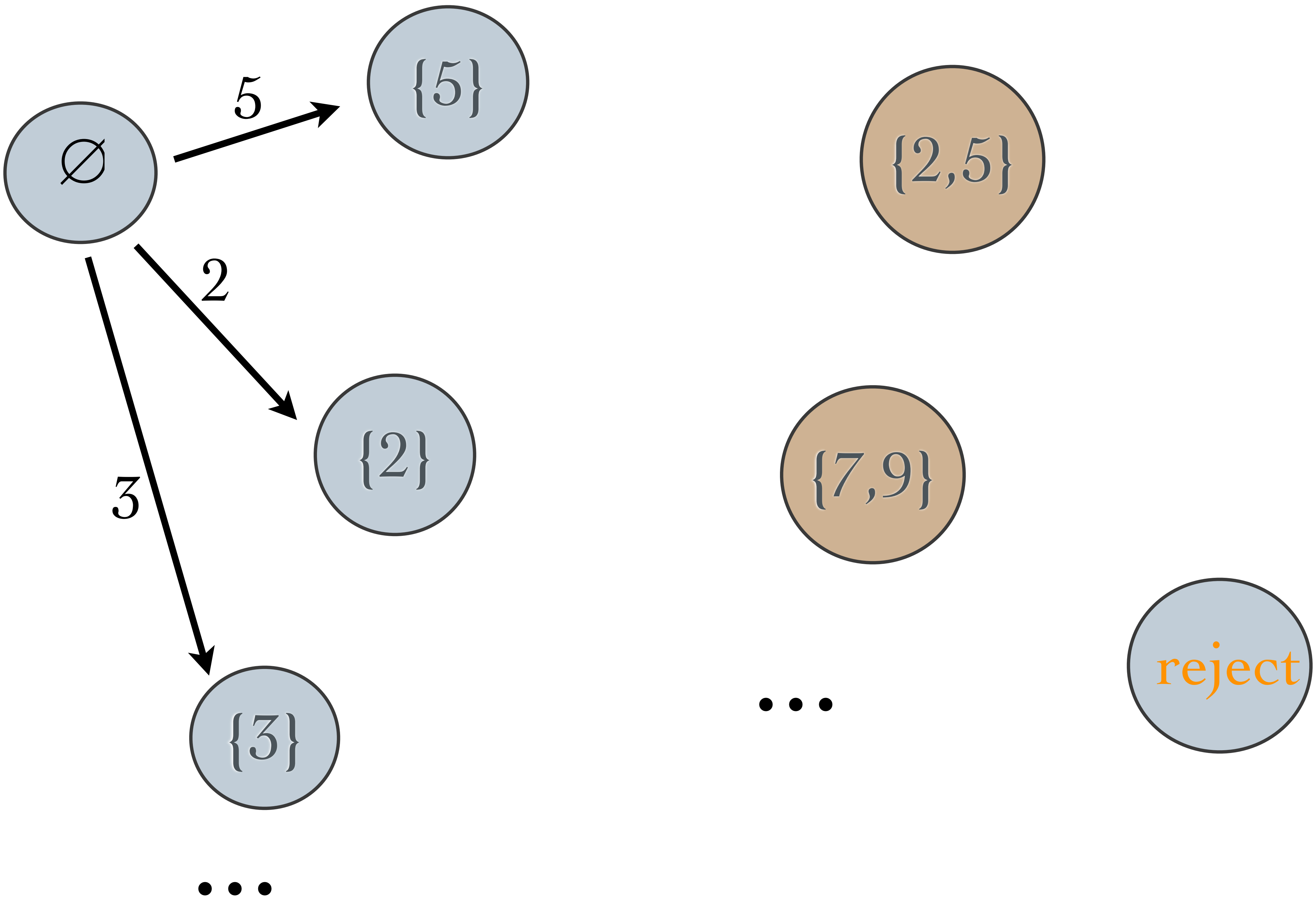
"exactly two different atoms appear"



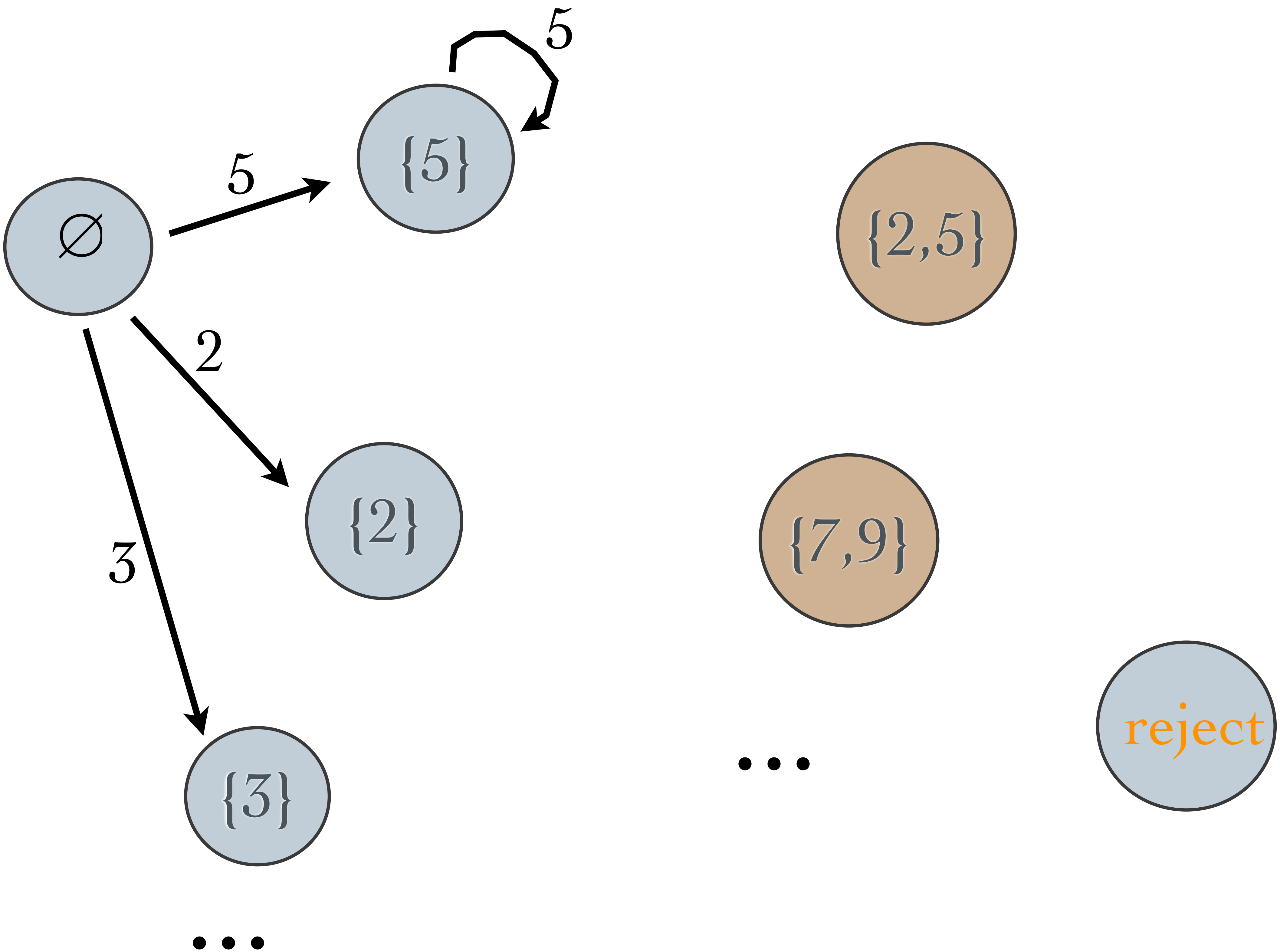
"exactly two different atoms appear"



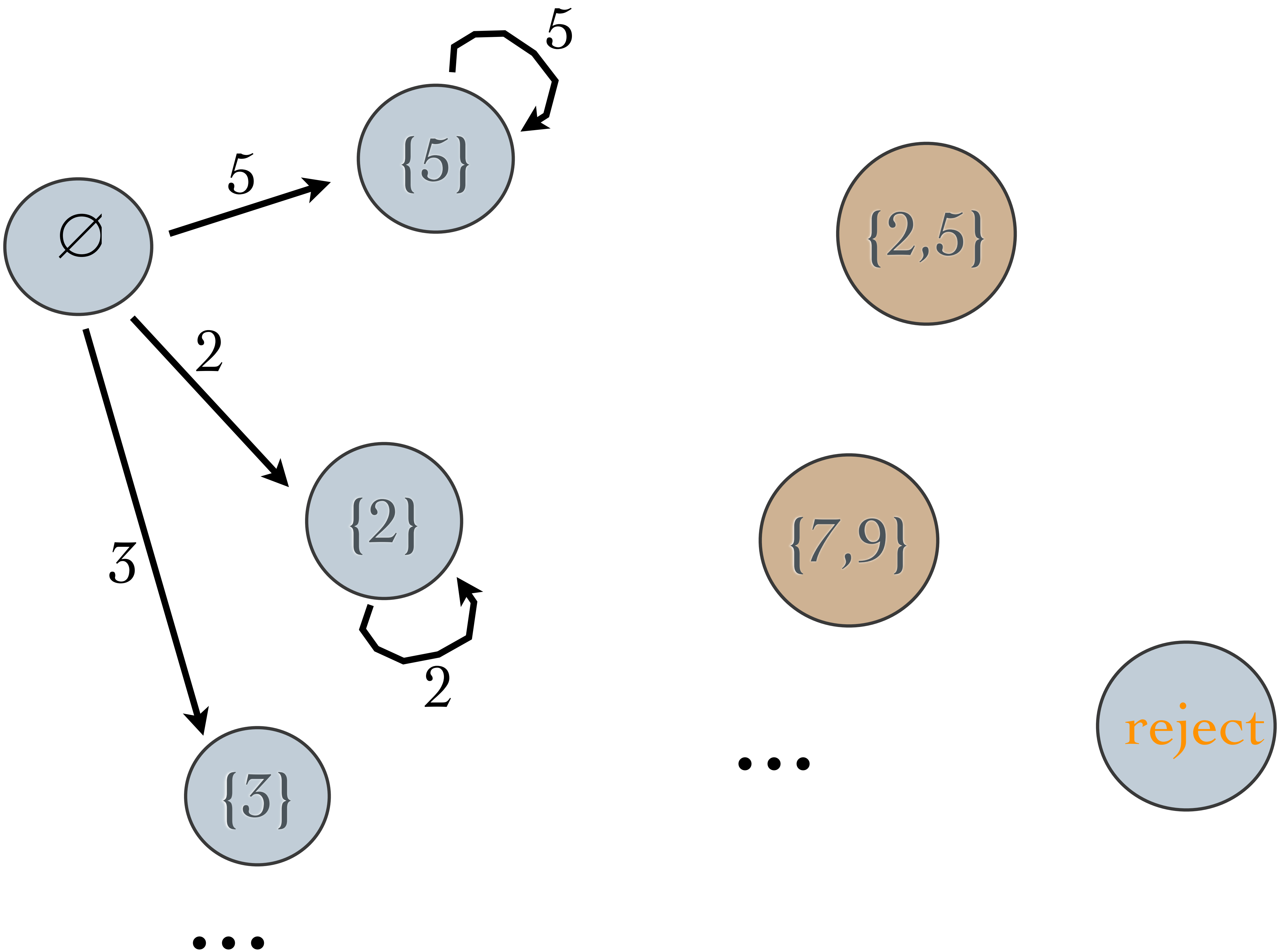
"exactly two different atoms appear"



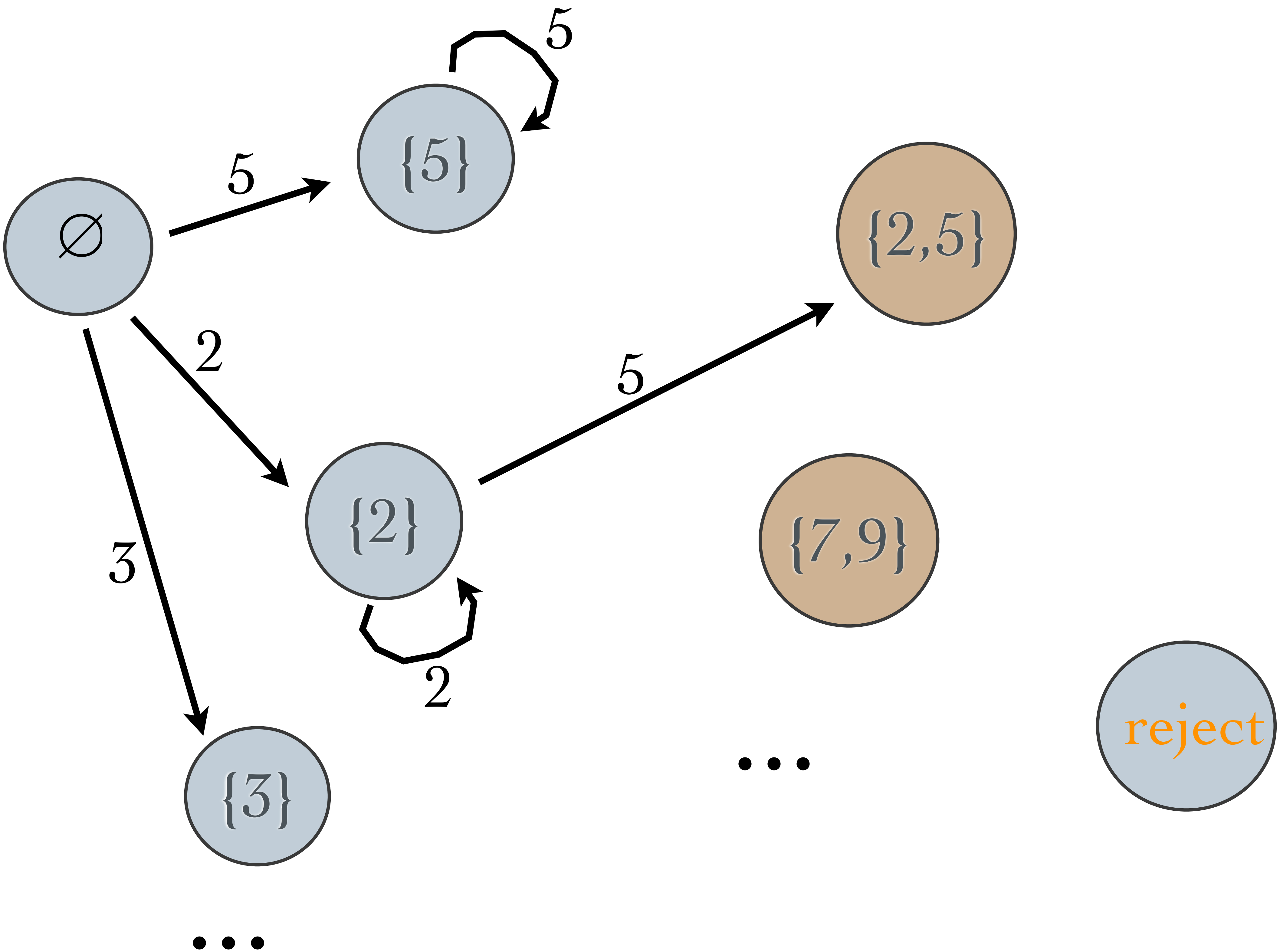
"exactly two different atoms appear"



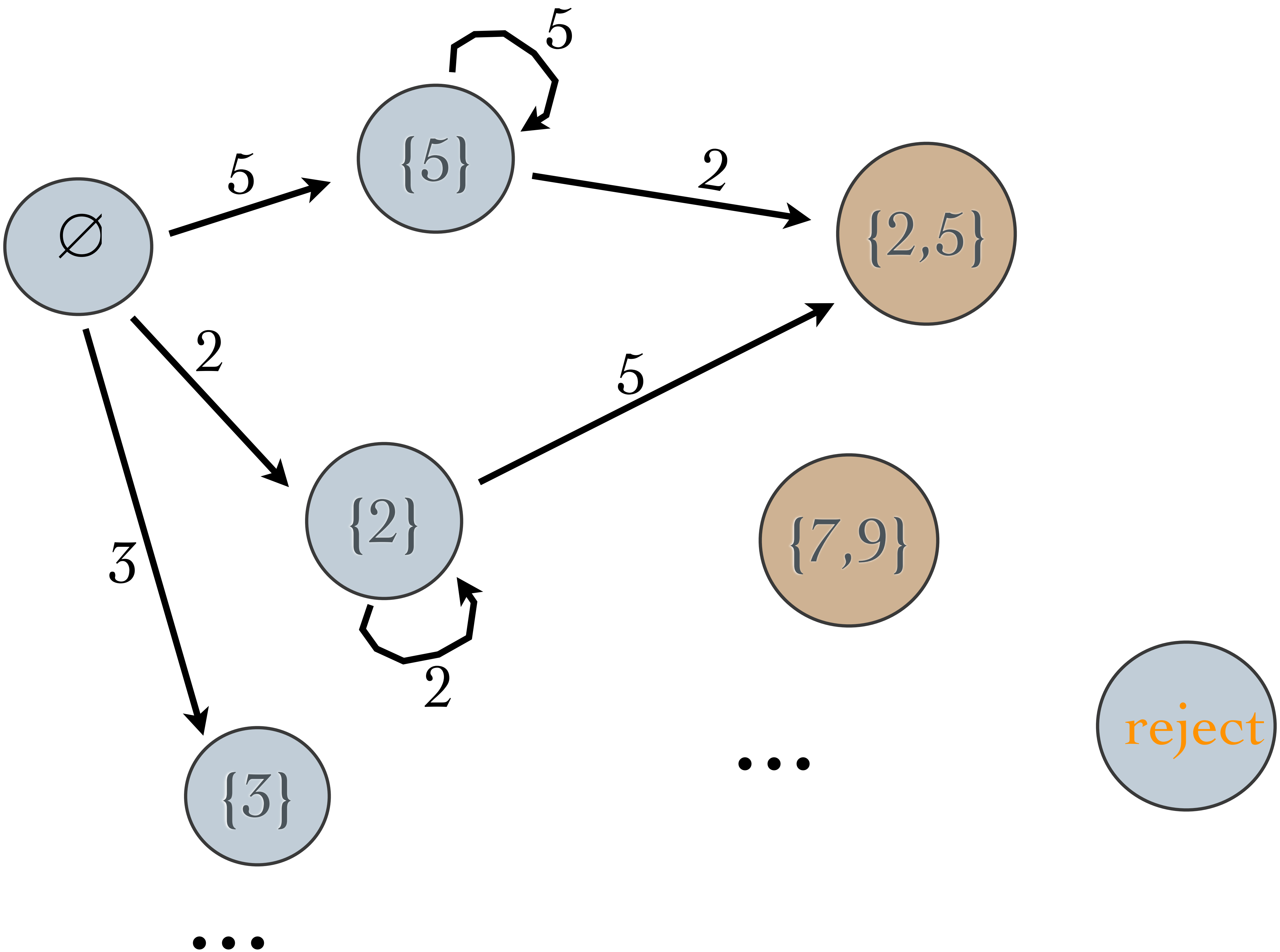
"exactly two different atoms appear"



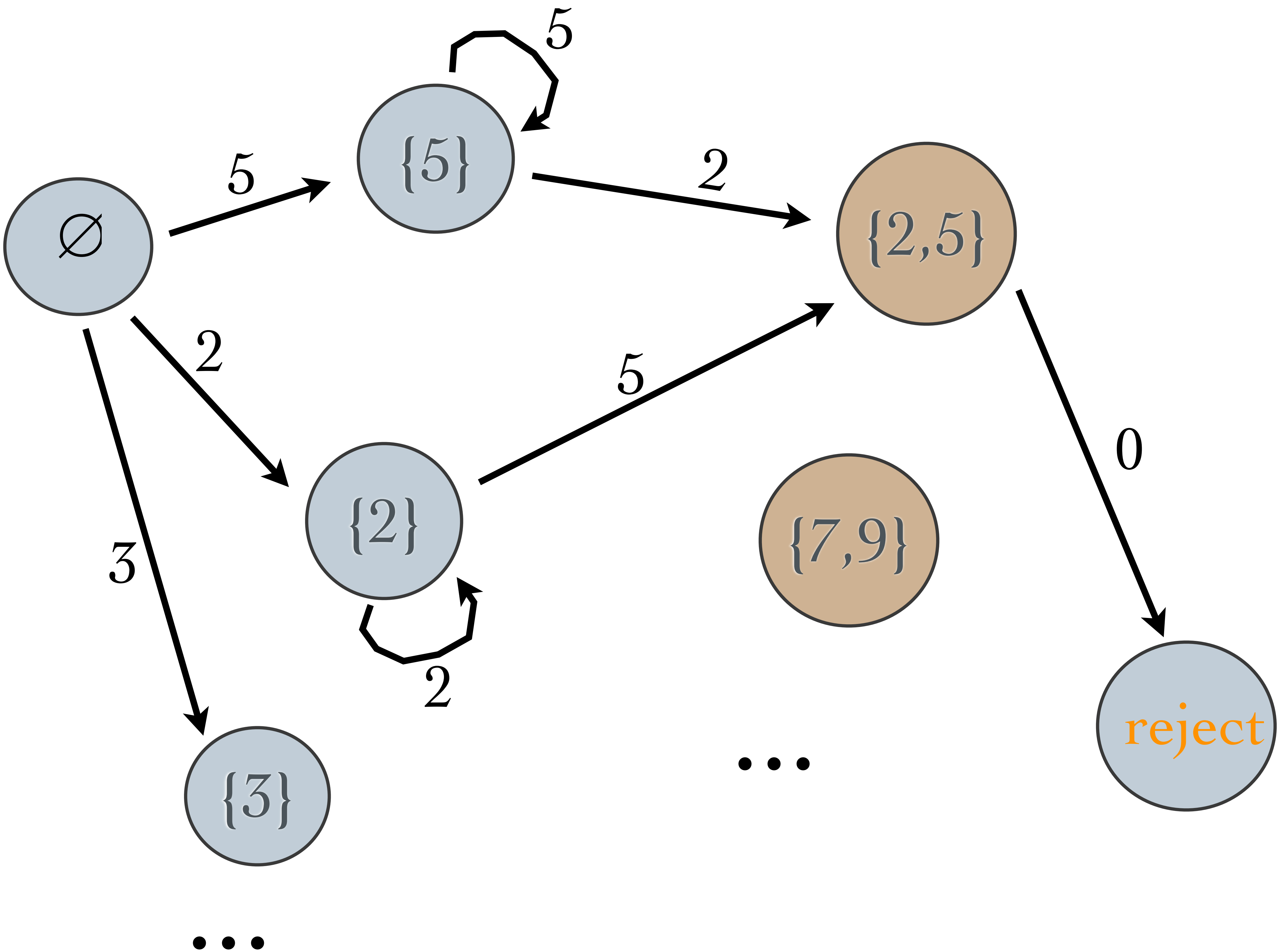
"exactly two different atoms appear"



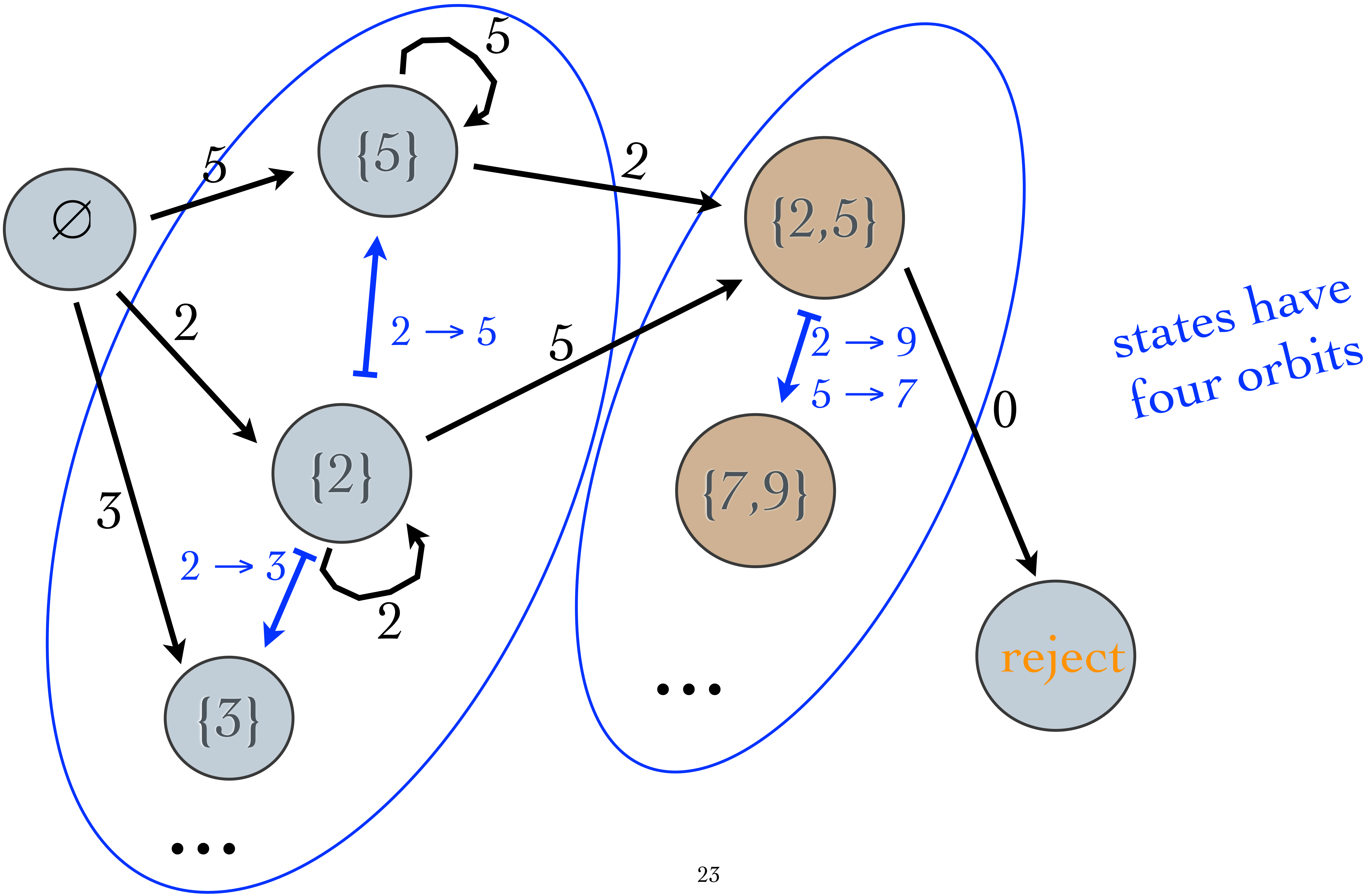
"exactly two different atoms appear"



"exactly two different atoms appear"



"exactly two different atoms appear"



Slight generalization of register automata:

Slight generalization of register automata:

- number of registers may vary from one orbit to another

Slight generalization of register automata:

- number of registers may vary from one orbit to another
- registers are not necessarily ordered

Slight generalization of register automata:

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

Slight generalization of register automata:

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

this is not a design decision,
but a property of orbit-finite sets

Expressive power

register automata
with equality tests
x = y

=

automata with
equality atoms (N, =)
over alphabet
atoms × (a finite set)

Expressive power

register automata
with equality tests
$x = y$

=

automata with
equality atoms ($\mathbb{N}$, =)
over alphabet
atoms × (a finite set)

register automata
with inequality tests
$x \leq y$

=

automata with
total order atoms ($\mathbb{Q}$, ≤)
over alphabet
atoms × (a finite set)

Minimization

register automata
with equality tests
x = y

=

automata with
equality atoms (N, =)
over alphabet
atoms × (a finite set)

Minimization

deterministic
register automata
with equality tests
$x = y$

=

deterministic
automata with
equality atoms ($\mathbb{N}$, =)
over alphabet
atoms × (a finite set)

Minimization

deterministic
register automata
with equality tests
x = y

=

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

do not minimize

Minimization

deterministic
register automata
with equality tests
x = y

=

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

do not minimize

do minimize

Myhill-Nerode Theorem

Myhill-Nerode Theorem

Theorem:

L is recognized by a deterministic automaton

iff

the set of L-equivalence classes is orbit-finite

Myhill-Nerode Theorem

Theorem:

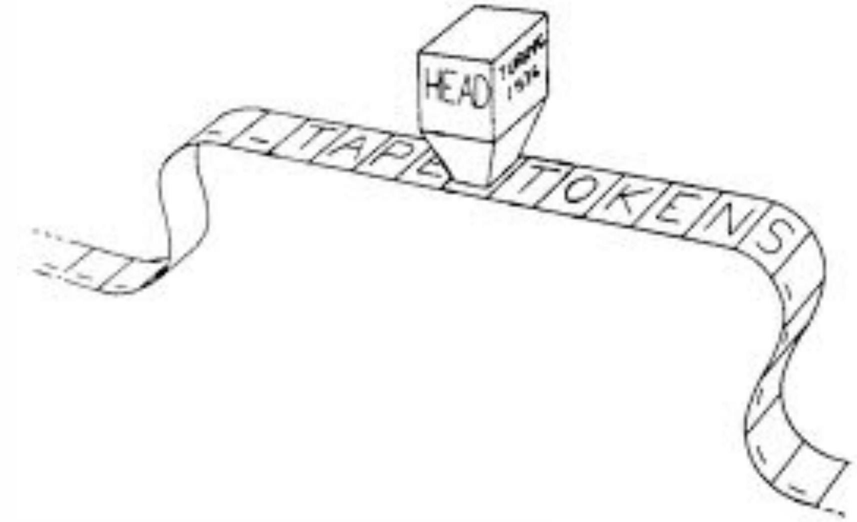
L is recognized by a deterministic automaton

iff

the set of L-equivalence classes is orbit-finite

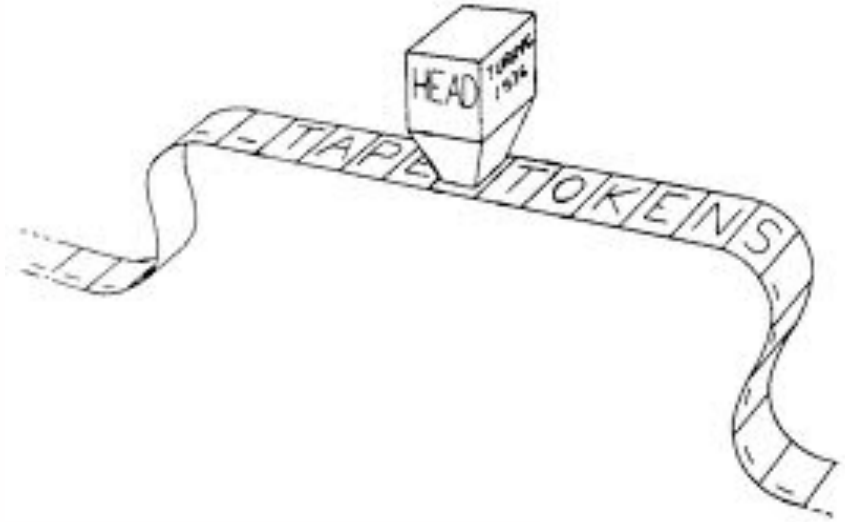
The equivalence classes are states of the minimal automaton for L

Turing machines



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

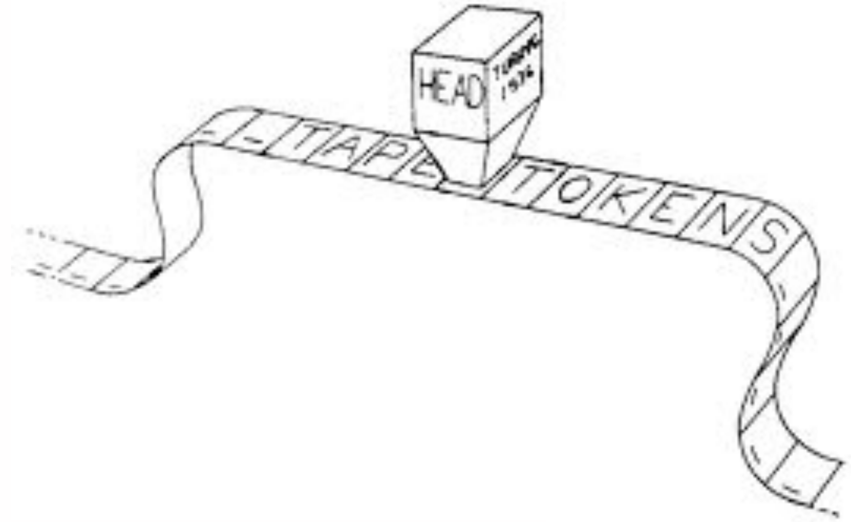
Turing machines



- 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

Turing machines

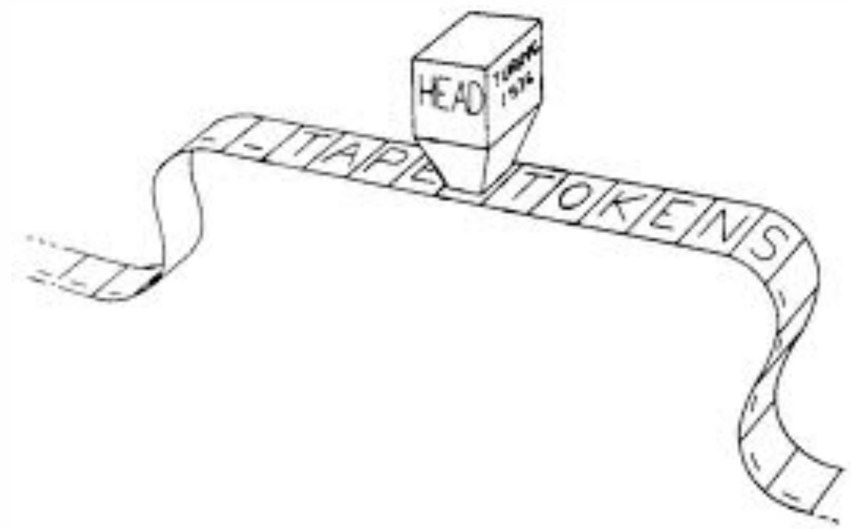


- tape alphabet A
- states Q
- subset $\delta \subseteq Q \times A \times Q \times A \times \{\leftarrow, \rightarrow, \downarrow\}$
- subsets $I, F \subseteq Q$

} orbit-finite sets instead of finite ones

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

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:

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

input alphabet: atoms

language:

tape alphabet:

states:

transitions:

input alphabet: atoms

language: "no atom appears twice":

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

tape alphabet:

states:

transitions:

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 $\cup$ $\{\perp\}$

states:

transitions:

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:

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 = <u>atoms</u> ∪ {start, accept, ret}

transitions: δ : Q × A → Q × A × {←,→,↓}

input alphabet: atoms

language: "no atom appears twice":

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

tape alphabet: $A = \text{atoms} \cup \{\perp\}$

states: $Q = \underline{\text{atoms}} \cup \{\text{start}, \text{accept}, \text{ret}\}$

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

$\delta(\text{start}, a) = (\underline{a}, \perp, \rightarrow)$ $\qquad a \in \text{atoms}$

if in state start atom a is read from tape, goto state <u>a</u>, write ⊥ on tape, and move right

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 = <u>atoms</u> ∪ {start, accept, ret}

transitions: δ : Q × A → Q × A × {←,→,↓}

δ(start, a) = (<u>a</u>, ⊥, →) a ∈ atoms

δ(<u>a</u>, b) = (<u>a</u>, b, →) a ≠ b, a, b ∈ atoms

if in state <u>a</u> atom b ≠ a is read from tape, stay in state <u>a</u>, write b on tape, and move right

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

$\delta(\text{start}, a) = (\underline{a}, \bot, \rightarrow)$ a ∈ atoms

$\delta(\underline{a}, b) = (\underline{a}, b, \rightarrow)$ a ≠ b, a, b ∈ atoms

$\delta(\underline{a}, B) = (\text{ret}, B, \leftarrow)$ a ∈ atoms

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 = <u>atoms</u> ∪ {start, accept, ret}

transitions: δ : Q × A → Q × A × {←,→,↓}

δ(start, a) =	(<u>a</u>, ⊥, →)	a ∈ atoms
δ(<u>a</u>, b) =	(<u>a</u>, b, →)	a ≠ b, a, b ∈ atoms
δ(<u>a</u>, B) =	(ret, B, ←)	a ∈ atoms
δ(ret, a) =	(ret, a, ←)	a ∈ atoms

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 = <u>atoms</u> ∪ {start, accept, ret}

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

δ(start, a) = (<u>a</u>, ⊥, →) a ∈ atoms

δ(<u>a</u>, b) = (<u>a</u>, b, →) a ≠ b, a, b ∈ atoms

δ(<u>a</u>, B) = (ret, B, ←) a ∈ atoms

δ(ret, a) = (ret, a, ←) a ∈ atoms

δ(ret, ⊥) = (start, ⊥, →)

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 = <u>atoms</u> ∪ {start, accept, ret}

transitions: δ : Q × A → Q × A × {←,→,↓}

δ(start, a) = (<u>a</u>, ⊥, →) a ∈ atoms

δ(<u>a</u>, b) = (<u>a</u>, b, →) a ≠ b, a, b ∈ atoms

δ(<u>a</u>, B) = (ret, B, ←) a ∈ atoms

δ(ret, a) = (ret, a, ←) a ∈ atoms

δ(ret, ⊥) = (start, ⊥, →)

δ(start, B) = (accept, B, →)

Pushdown automata

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

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

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 = $Q \times S^*$

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 = $Q \times S^*$

Deterministic pushdown automata: ...

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 = $Q \times S^*$

Deterministic pushdown automata: ...

Theorem:

Pushdown automata = prefix-rewriting

Context-free grammars

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

Context-free grammars

- symbols S
- terminal symbols A ⊆ S
- an initial symbol
- δ ⊆ (S–A)×S*

} orbit-finite sets instead of finite ones

Context-free grammars

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

} orbit-finite sets instead of finite ones

Theorem:

Context-free grammars = pushdown automata

Petri nets

Petri nets

- places P

Petri nets

- places P

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

Petri nets

- places P
- an initial configuration

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

Petri nets

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

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

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 $M_{fin}(P)$

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 $M_{fin}(P)$

classical sets	sets with atoms (N, =)

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 $M_{fin}(P)$

classical sets	sets with atoms (N, =)

places:
atoms × (a finite set)

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 $M_{fin}(P)$

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

places:
atoms × (a finite set)

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 $M_{fin}(P)$

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

places: atoms × (a finite set)

Outline

- Sets with atoms
- Models of computation in sets with atoms
- Are sets with atoms useful?

usefulness of sets with atoms in infinite-state verification

usefulness of sets with atoms in infinite-state verification

- orbit-finite abstractions

usefulness of sets with atoms in infinite-state verification

- orbit-finite abstractions
- clarification and unification of known methods

usefulness of sets with atoms in infinite-state verification

- orbit-finite abstractions
- clarification and unification of known methods
- solver of already solved problems

usefulness of sets with atoms in infinite-state verification

- orbit-finite abstractions
- clarification and unification of known methods
- solver of already solved problems
- solver of previously unsolved problems

usefulness of sets with atoms in infinite-state verification

- orbit-finite abstractions
- clarification and unification of known methods
- solver of already solved problems
- solver of previously unsolved problems
- generator of new interesting problems

usefulness of sets with atoms in infinite-state verification

- orbit-finite abstractions
- clarification and unification of known methods
- solver of already solved problems
- solver of previously unsolved problems
- generator of new interesting problems
- relationships with other fields

orbit-finite abstractions

orbit-finite abstractions

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

orbit-finite abstractions

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

Corollary: Emptiness of pushdown automata is decidable

orbit-finite abstractions

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

Corollary: Emptiness of pushdown automata is decidable

Potential application to orbit-infinite abstractions in analysis of recursive program.

clarification and unification

clarification and unification

Theorem: reachability is decidable for alternating automata with one register/clock

[Ouaknine, Worrel '05]
[L., Walukiewicz '05]
[Demri, Lazic '09]

clarification and unification

Theorem: reachability is decidable for alternating automata with one register/clock

[Ouaknine, Worrel '05]
[L., Walukiewicz '05]
[Demri, Lazic '09]

Idea: concrete configurations $\longmapsto$ abstract configurations (regions)

clarification and unification

Theorem: reachability is decidable for alternating automata with one register/clock

[Ouaknine, Worrel '05]
[L., Walukiewicz '05]
[Demri, Lazic '09]

Idea: concrete configurations $\longmapsto$ abstract configurations (regions)

Theorem: there is one decision procedure for this problem, that terminates when P_{fin}(atoms) is a WQO

[Bojańczyk, Braud, Klin, L. '12]

clarification and unification

Theorem: reachability is decidable for alternating automata with one register/clock

[Ouaknine, Worrel '05]
[L., Walukiewicz '05]
[Demri, Lazic '09]

Idea: concrete configurations $\longmapsto$ abstract configurations (regions)

Theorem: there is one decision procedure for this problem, that terminates when P_{fin}(atoms) is a WQO

[Bojańczyk, Braud, Klin, L. '12]

Idea: regions = orbits

clarification and unification

Theorem: reachability is decidable for alternating automata with one register/clock
[Ouaknine, Worrel '05]
[L., Walukiewicz '05]
[Demri, Lazic '09]

Idea: concrete configurations $\longmapsto$ abstract configurations (regions)

Theorem: there is one decision procedure for this problem, that terminates when P_{fin}(atoms) is a WQO
[Bojańczyk, Braud, Klin, L. '12]

Idea: regions = orbits

solver of already solved problems

solver of already solved problems

- coverabily of timed/data Petri nets
  [Lazic, Newcomb, Ouaknine, Roscoe, Worrell '12]
  [Abdulla, Nylen '01]
- c. s. reachability of timed/data lossy FIFO automata
  [Abdulla, Atig, Cederberg '12]
- emptiness of timed/data pushdown systems
  [Abdulla, Atig, Stenman '12]
  [Dubov, Kaminski '09]
- relating timed and data variants
  [Figueira, Hofman, L.'10]
  [Bonnet, Finkel, Haddad, Rosa-Velardo '10]

solver of previously unsolved problems

solver of previously unsolved problems

- minimization of deterministic register automata
  [Bojańczyk, Klin, L. '11]
- machine-independent characterization of det. timed languages
  [Bojańczyk, L. '12]
- verification of database-driven systems
  [Bojańczyk, Segoufin, Toruńczyk '13]

generator of new problems

generator of new problems

- decidability of reachability for Petri nets
- decidability of equivalence of deterministic pushdown automata
- ...

relationships with other fields

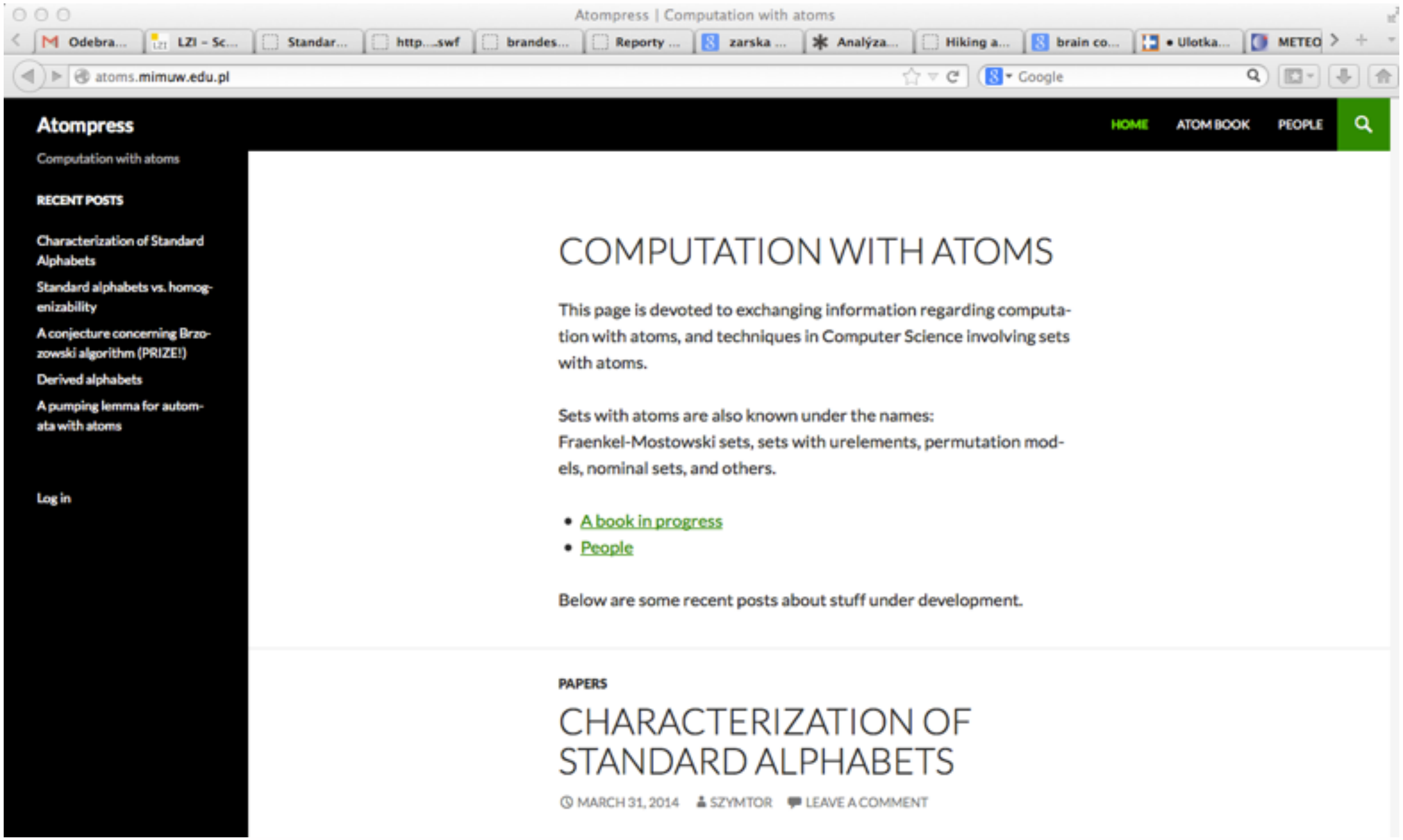
relationships with other fields

- CSP theory
- descriptive complexity

} [Klin, L., Ochremiak, Toruńczyk '14]

- model-theory
  - homogenizability
  - automorphisms with bounded color classes
- finite permutation groups
- ...

visit our blog



thank you!