Checking Whether an Automaton Is Monotonic Is NP-complete

Marek Szykuła

University of Wrocław, Poland

CIAA, 18.08.2015

Monotonic automata

We deal with complete (semi)automata $\mathscr{A} = (Q, \Sigma, \delta)$.

Monotonic automaton

An automaton is monotonic if there exists a linear order $\preceq$ on Q such that:
for every $p, q \in Q$ and $a \in \Sigma$, if $p \preceq q$ then $\delta(p, a) \preceq \delta(q, a)$.

Immediately we have also that $p \preceq q$ implies $\delta(p, w) \preceq \delta(q, w)$, for every word $w \in \Sigma^*$.

Monotonic automata

We deal with complete (semi)automata $\mathscr{A} = (Q, \Sigma, \delta)$.

Monotonic automaton

An automaton is monotonic if there exists a linear order $\preceq$ on Q such that:
for every $p, q \in Q$ and $a \in \Sigma$, if $p \preceq q$ then $\delta(p, a) \preceq \delta(q, a)$.

Immediately we have also that $p \preceq q$ implies $\delta(p, w) \preceq \delta(q, w)$, for every word $w \in \Sigma^*$.

Monotonic automata

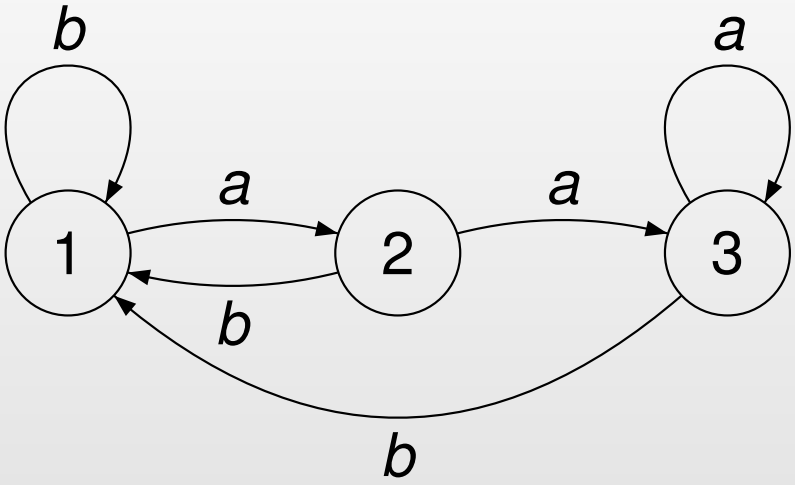
We deal with complete (semi)automata $\mathscr{A} = (Q, \Sigma, \delta)$.

Monotonic automaton

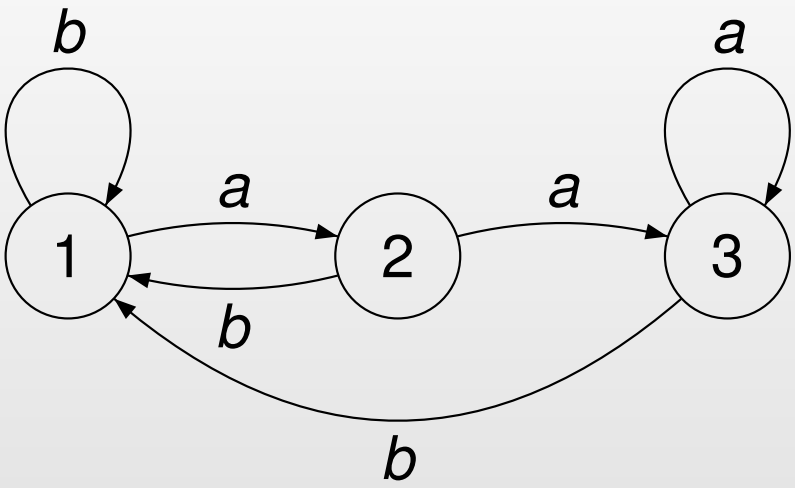
An automaton is monotonic if there exists a linear order $\preceq$ on Q such that:
for every $p, q \in Q$ and $a \in \Sigma$, if $p \preceq q$ then $\delta(p, a) \preceq \delta(q, a)$.

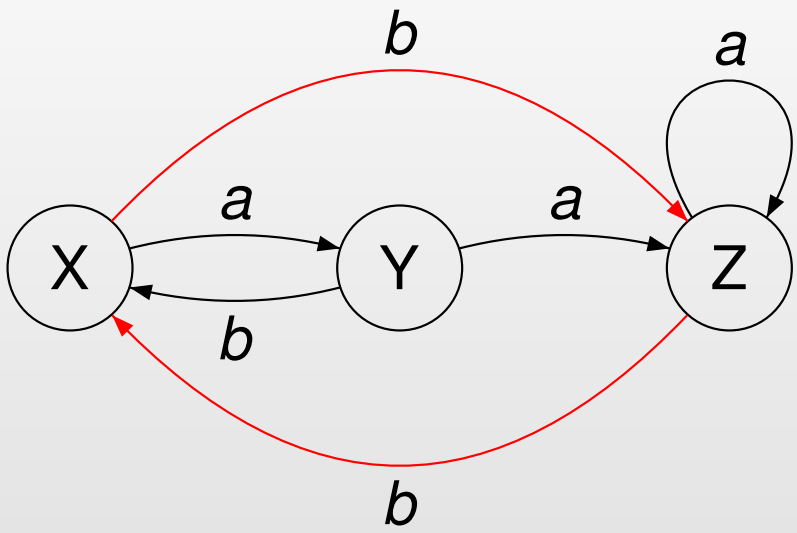
Immediately we have also that $p \preceq q$ implies $\delta(p, w) \preceq \delta(q, w)$, for every word $w \in \Sigma^*$.

Examples of monotonic automata

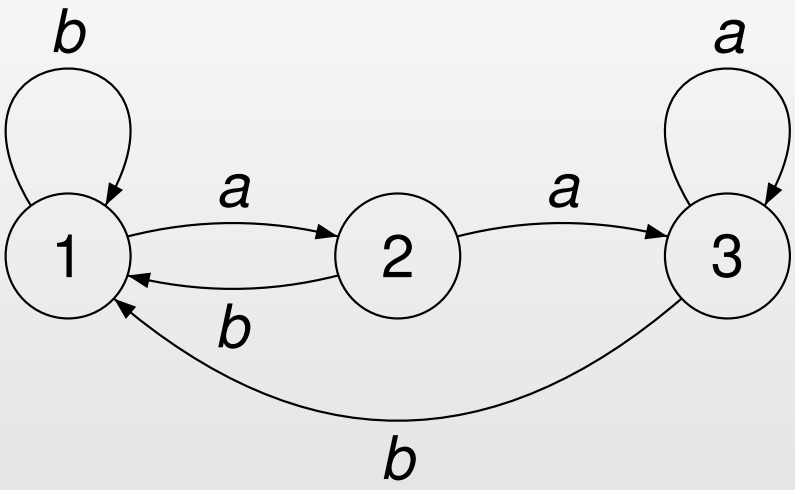


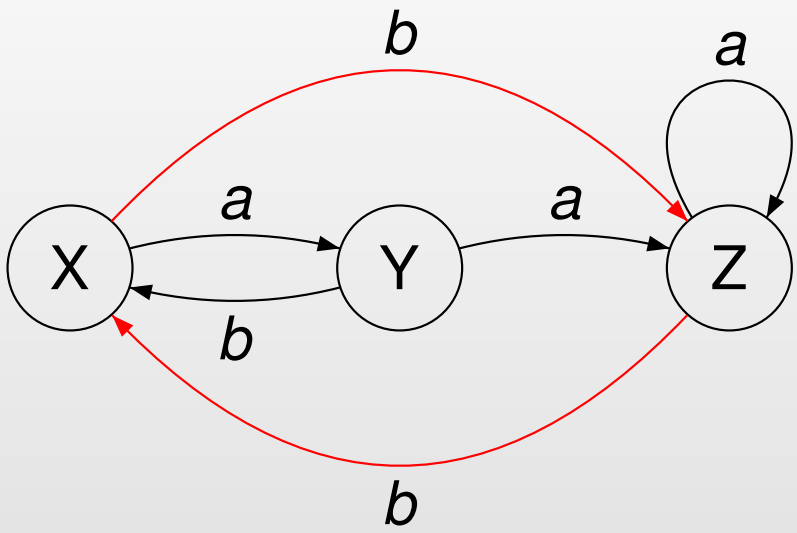
Examples of monotonic automata

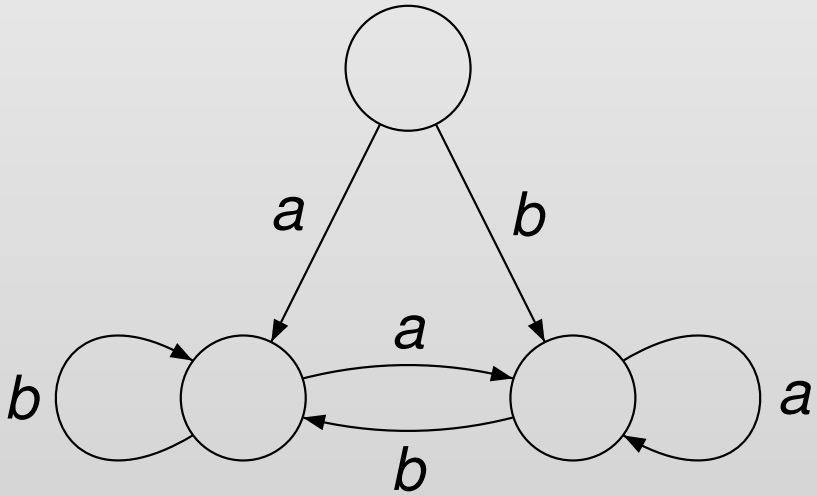




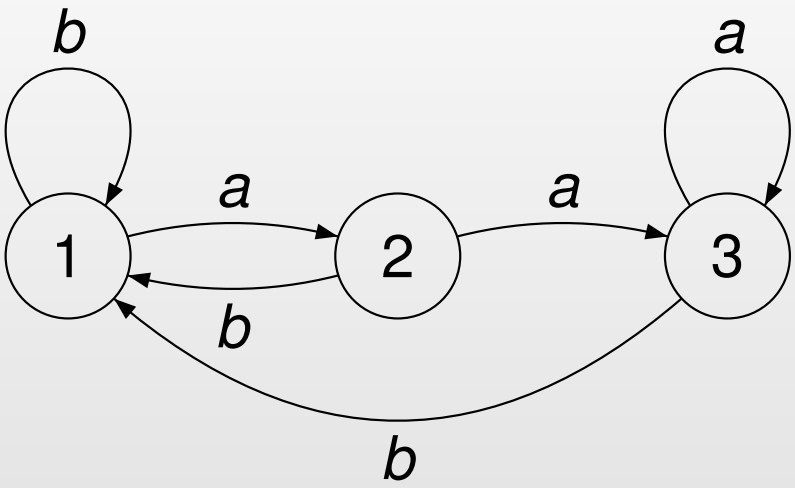
Examples of monotonic automata

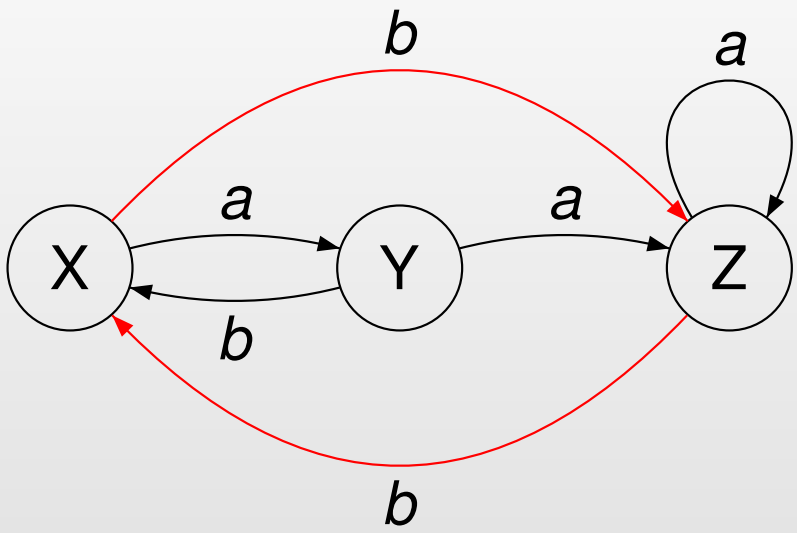


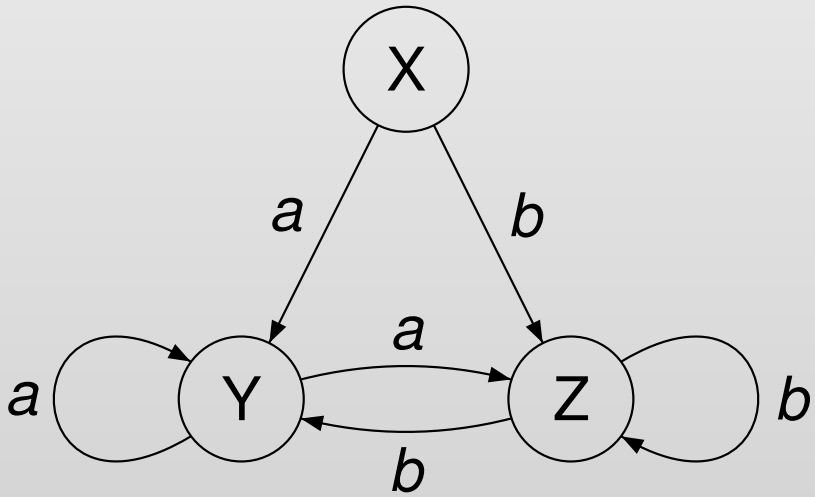




Examples of monotonic automata



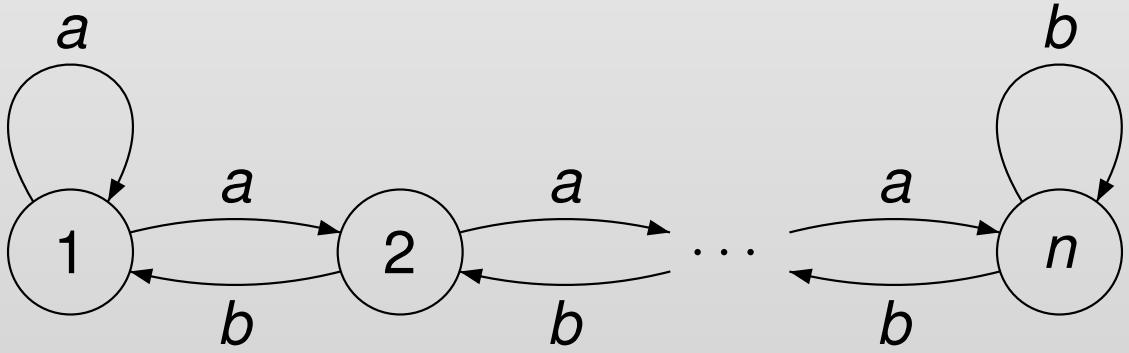




$X \prec Y$ then (by a) $Y \prec Z$, but $X \prec Z$ implies (by b) $Z \prec Y$.

Monotonic automata

- A proper subclass of aperiodic automata (aperiodic – transformations induced by words do not have non-trivial cycles).
- (Gomes, Howie 1992) Maximal transition semigroups are isomorphic to the transition semigroup of:



- This contains all $\binom{2n-1}{n}$ order-preserving transformations.

Oriented automata

- Introduced by Eppstein (1990) under the term “monotonic”.
- Automata with a cyclic order of the states rather than linear.
- This is a wider class containing monotonic automata.

Oriented automata

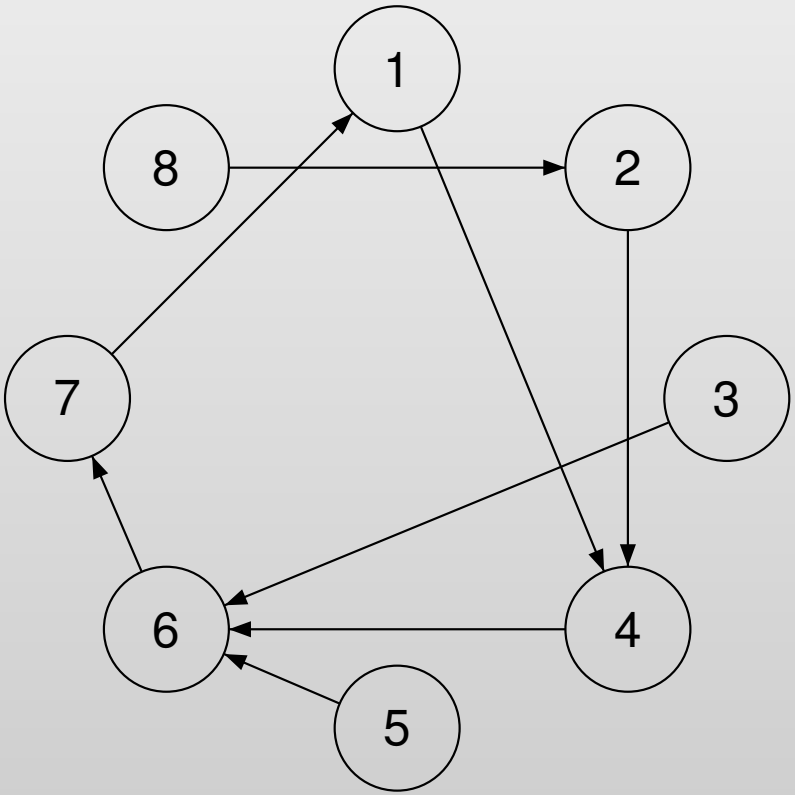
Definition of an oriented automaton

An automaton is oriented if there is a cyclic order of the states that is preserved by the actions of all letters.

Oriented automata

Definition of an oriented automaton

An automaton is oriented if there is a cyclic order of the states that is preserved by the actions of all letters.



$[1, 2, 3, 4, 5, 6, 7, 8]$

↓

$[4, 4, 6, 6, 6, 7, 1, 2]$

↓

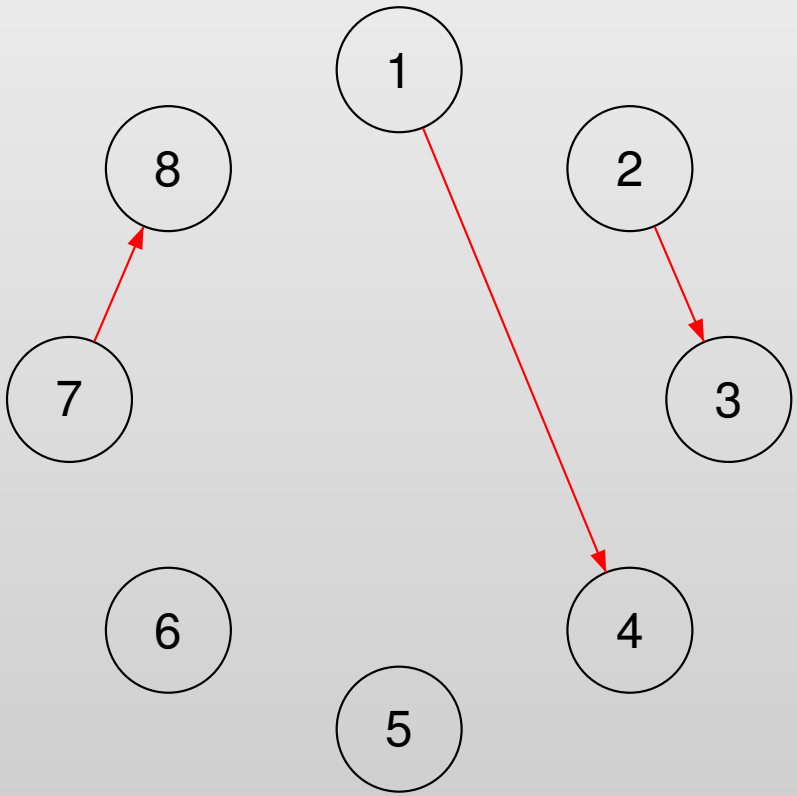
$[4, 6, 7, 1, 2]$

subsequence of the cyclic order

Oriented automata

Definition of an oriented automaton

An automaton is oriented if there is a cyclic order of the states that is preserved by the actions of all letters.



This transformation does not preserve the cyclic order.

Motivation

- Monotonic automata appears in several contexts (e.g. as bulding blocks in largest aperiodic semigroups, infiniteness of the dot-depth hierarchy).
- Recognizing languages recognized by monotonic automata.

Applications in synchronizing automata

- Tight bounds for the length of the shortest reset words:
  - Monotonic: $n-1$ (Ananichev, Volkov 2004).
  - Oriented: $(n-1)^2$ (Eppstein 1990).
- Polynomial algorithms finding a shortest reset word of a given automaton with a preserved order, which is a hard problem in general.

Motivation

- Monotonic automata appears in several contexts (e.g. as bulding blocks in largest aperiodic semigroups, infiniteness of the dot-depth hierarchy).
- Recognizing languages recognized by monotonic automata.

Applications in synchronizing automata

- Tight bounds for the length of the shortest reset words:
  - Monotonic: $n-1$ (Ananichev, Volkov 2004).
  - Oriented: $(n-1)^2$ (Eppstein 1990).
- Polynomial algorithms finding a shortest reset word of a given automaton with a preserved order, which is a hard problem in general.

Motivation

- Monotonic automata appears in several contexts (e.g. as bulding blocks in largest aperiodic semigroups, infiniteness of the dot-depth hierarchy).
- Recognizing languages recognized by monotonic automata.

Applications in synchronizing automata

- Tight bounds for the length of the shortest reset words:
  - Monotonic: $n-1$ (Ananichev, Volkov 2004).
  - Oriented: $(n-1)^2$ (Eppstein 1990).
- Polynomial algorithms finding a shortest reset word of a given automaton with a preserved order, which is a hard problem in general.

Motivation

- Monotonic automata appears in several contexts (e.g. as bulding blocks in largest aperiodic semigroups, infiniteness of the dot-depth hierarchy).
- Recognizing languages recognized by monotonic automata.

Applications in synchronizing automata

- Tight bounds for the length of the shortest reset words:
  - Monotonic: $n-1$ (Ananichev, Volkov 2004).
  - Oriented: $(n-1)^2$ (Eppstein 1990).
- Polynomial algorithms finding a shortest reset word of a given automaton with a preserved order, which is a hard problem in general.

Theorem

The problem of checking whether a given automaton over at least binary alphabet is monotonic is NP-complete.

Proof

1. First we reduce from Monotone Not-All-Equal 3SAT.
   - Like 3SAT, but the formula is satisfied if and only if every clause contains at least one true and one false literal.
   - Literals are positive occurrences of variables (there are no negations).
2. Then we reduce to the problem of recognizing binary monotonic automata.

Theorem

The problem of checking whether a given automaton over at least binary alphabet is monotonic is NP-complete.

Proof

1. First we reduce from Monotone Not-All-Equal 3SAT.
   - Like 3SAT, but the formula is satisfied if and only if every clause contains at least one true and one false literal.
   - Literals are positive occurrences of variables (there are no negations).
2. Then we reduce to the problem of recognizing binary monotonic automata.

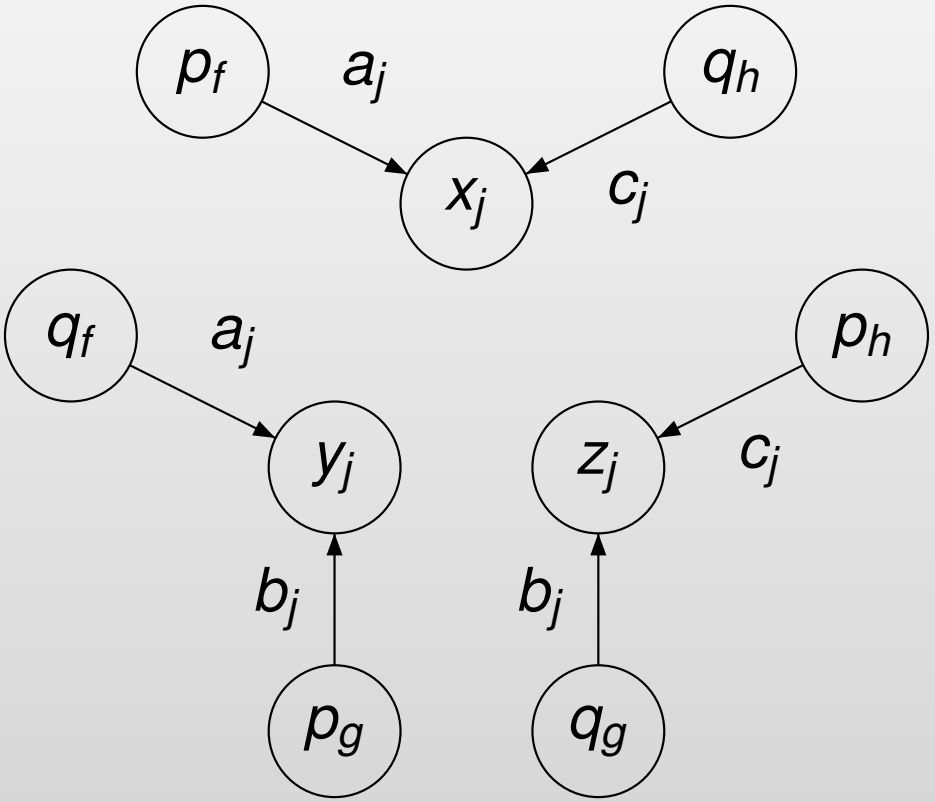
Theorem

The problem of checking whether a given automaton over at least binary alphabet is monotonic is NP-complete.

Proof

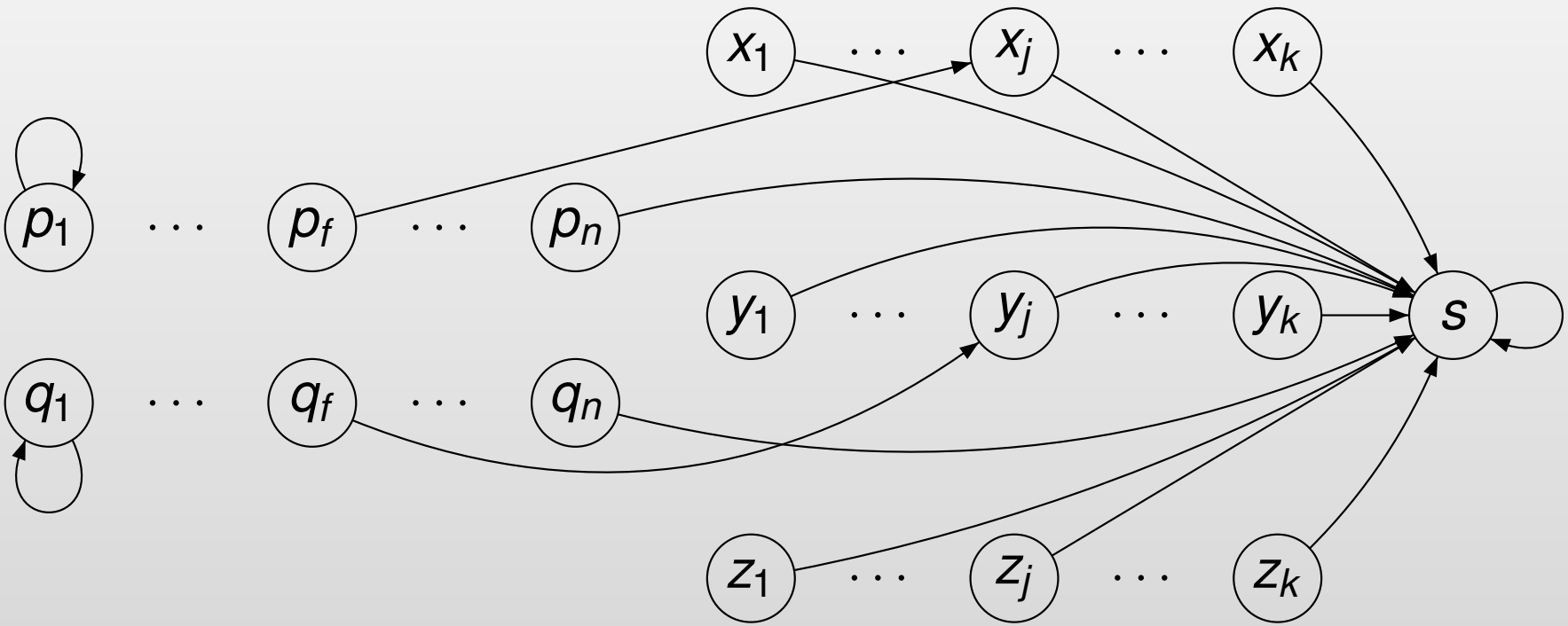
1. First we reduce from Monotone Not-All-Equal 3SAT.
   - Like 3SAT, but the formula is satisfied if and only if every clause contains at least one true and one false literal.
   - Literals are positive occurrences of variables (there are no negations).
2. Then we reduce to the problem of recognizing binary monotonic automata.

The clause gadget



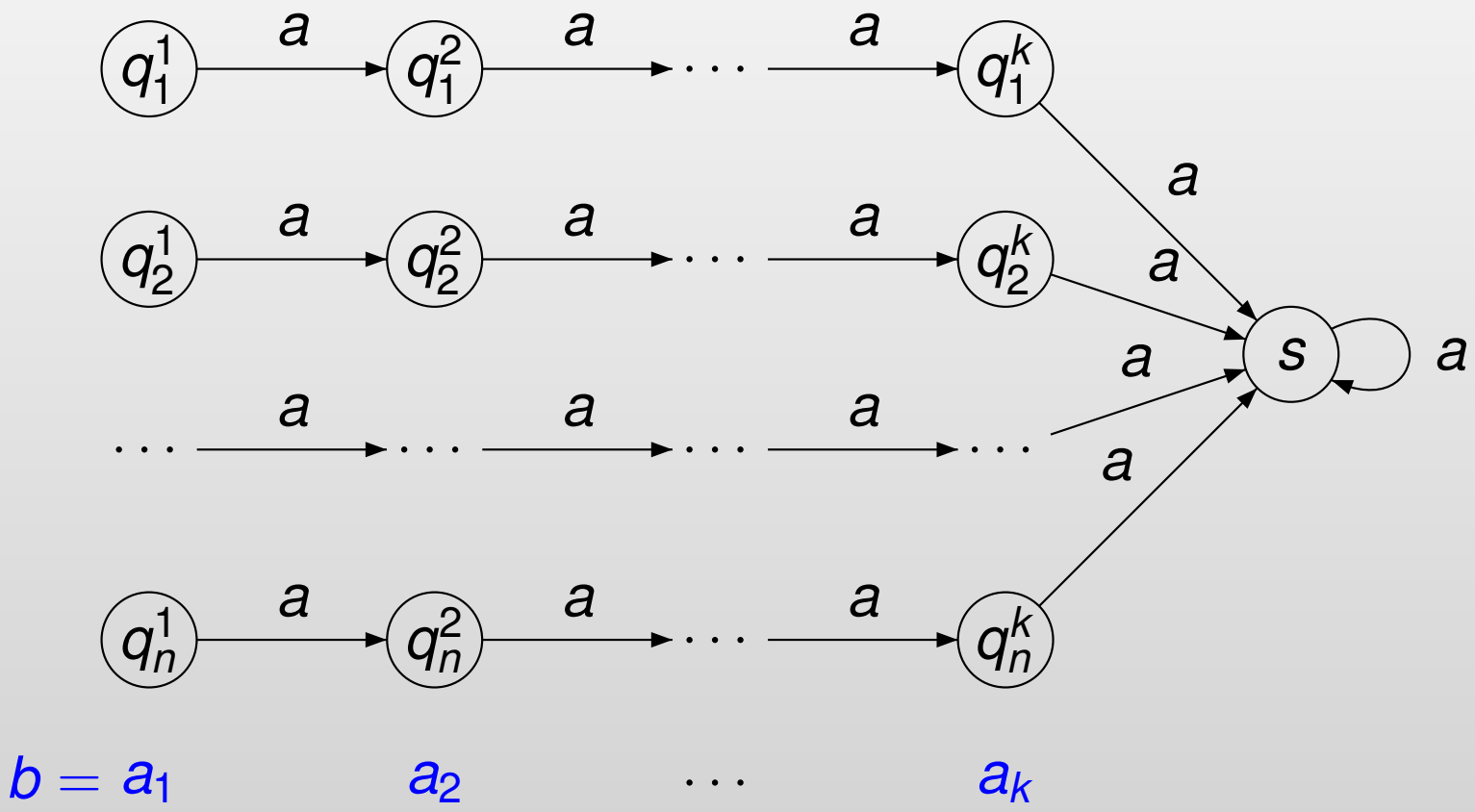
The clause gadget for a j-th clause (v_f, v_g, v_h).

The construction



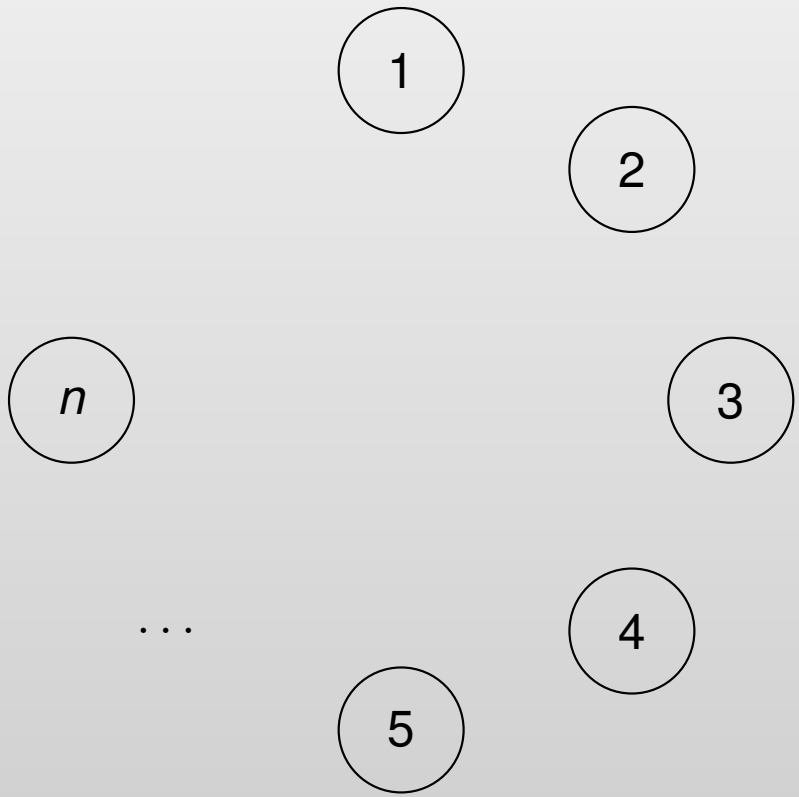
The action of the letter a_j, where v_f is the first variable in C_j.

Restriction to binary alphabets



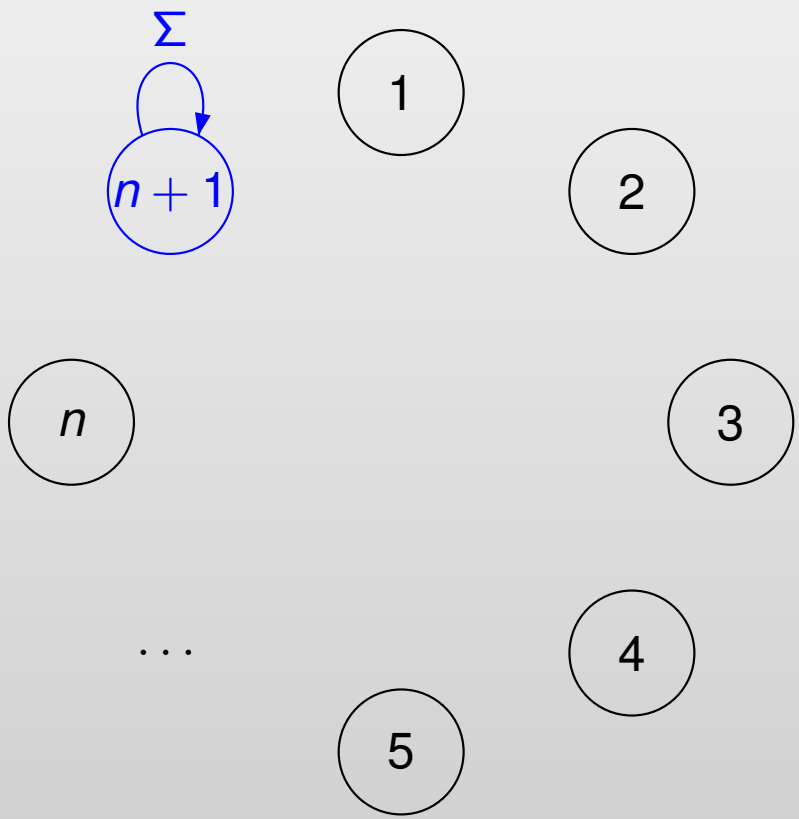
Theorem

The problem of checking whether a given automaton over at least binary alphabet is oriented is NP-complete.



Theorem

The problem of checking whether a given automaton over at least binary alphabet is oriented is NP-complete.

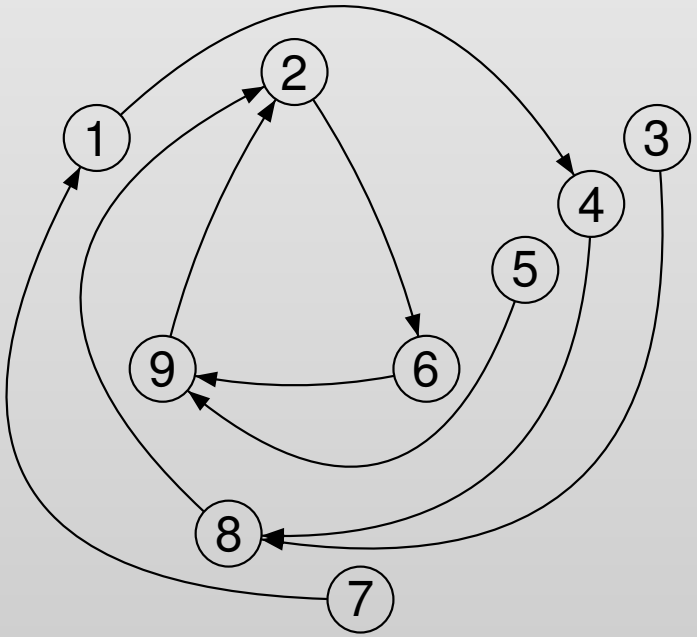


This is oriented if and only
the original automaton is monotonic.

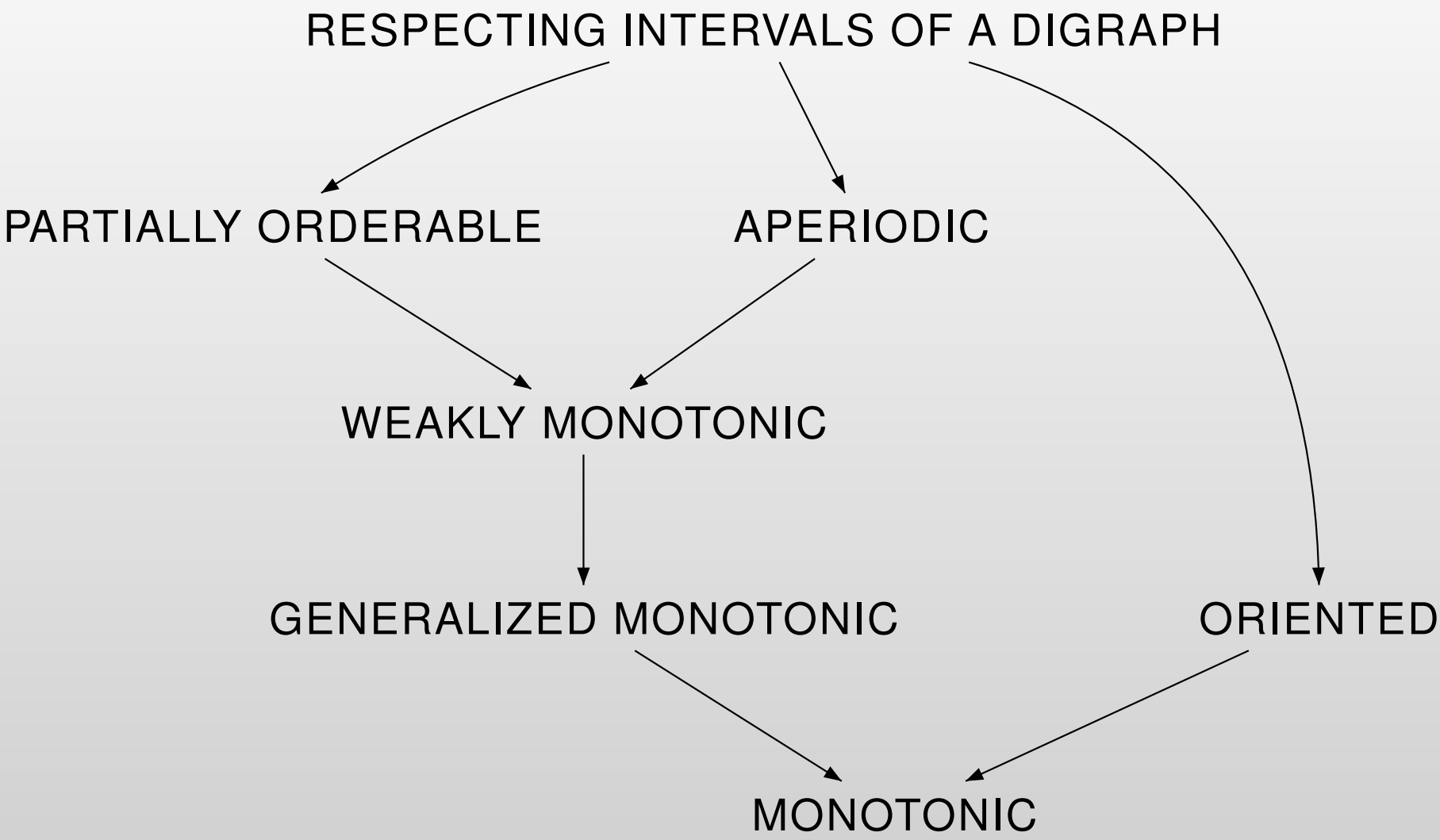
A note about unary alphabets

Unary alphabets

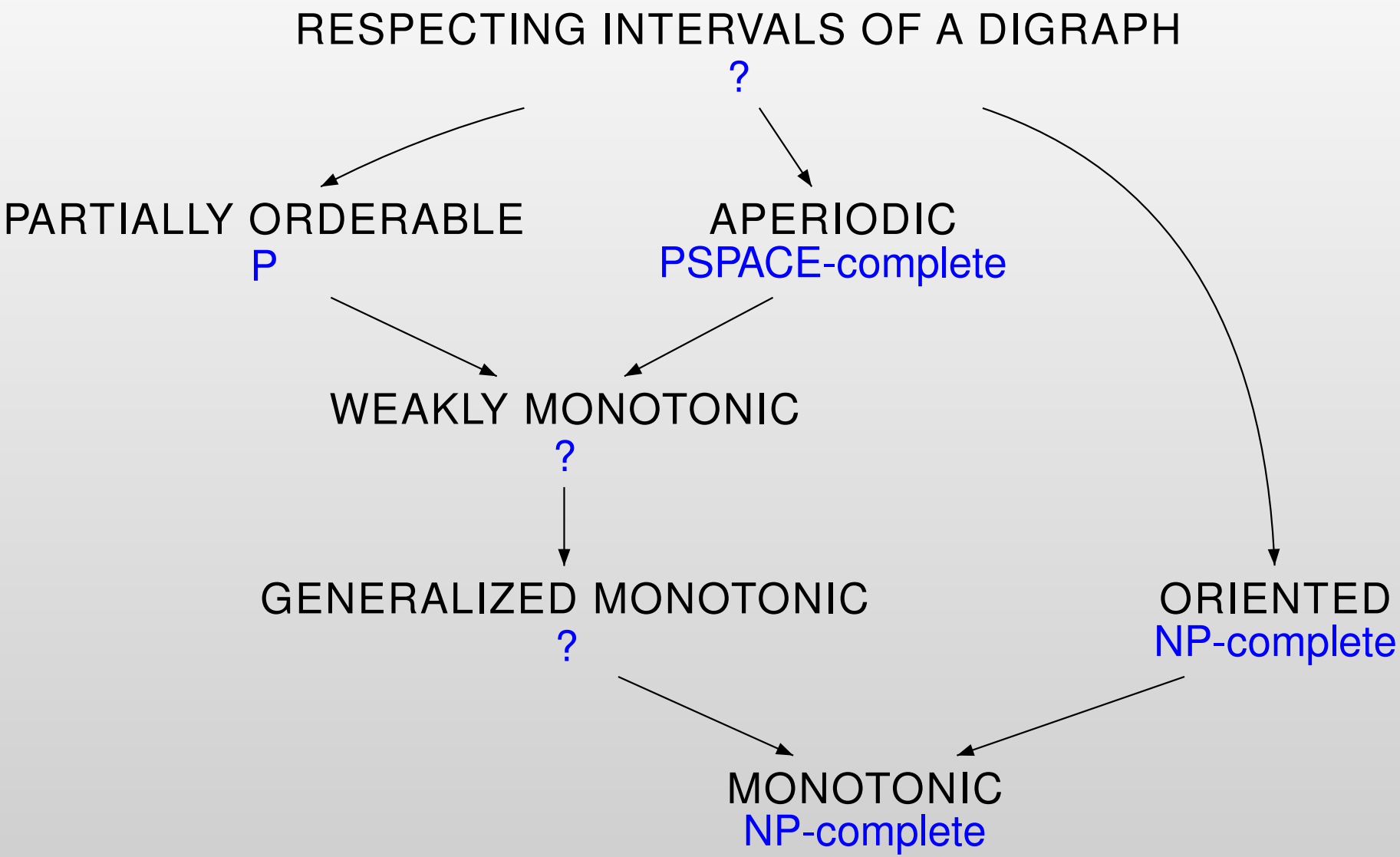
- A unary automaton is monotonic if and only the letter has no non-trivial cycle.
- A unary automaton is oriented if and only if all cycles of the letter have the same length.



Inclusion relations with other subclasses



Inclusion relations with other subclasses



Summary

- From the algorithmic point of view, usefulness of monotonicity and orientability is limited if do not know the preserved order.
- Complexity of recognizing generalized and weakly monotonic?
- Complexity of recognizing automata respecting intervals of a digraph?

Tack så mycket!

Summary

- From the algorithmic point of view, usefulness of monotonicity and orientability is limited if do not know the preserved order.
- Complexity of recognizing generalized and weakly monotonic?
- Complexity of recognizing automata respecting intervals of a digraph?

Tack så mycket!