# Advanced Logic

http://www-sop.inria.fr/members/Martin.Avanzini/teaching/2023/AL/

Martin Avanzini (martin.avanzini@inria.fr)
Etienne Lozes (etienne.lozes@univ-cotedazur.fr)

MASTER INFORMATIQUE
UNIVERSITÉ CÔTE D'AZUR

*2nd Semester M1, 2023*

# Last Lecture

1. the set of WMSO formulas over $\mathcal{V}_1, \mathcal{V}_2$ is given by the following grammar:

$$\phi, \psi ::= \top \mid \bot \mid x < y \mid X(x) \mid \phi \vee \psi \mid \neg \phi \mid \exists x.\phi \mid \exists X.\phi$$

   - first-order variables $\mathcal{V}_1$ range over $\mathbb{N}$ and second-order variables $\mathcal{V}_2$ range over finite sets over $\mathbb{N}$

2. a WMSO formula $\phi$ over second-order variables $\{P_a \mid a \in \Sigma\}$ defines a language

$$L(\phi) \triangleq \{w \in \Sigma^* \mid \underline{w} \vDash \phi\}$$

3. WMSO definable languages are regular, and vice verse

4. Satisfiability and validity decidable in $2^{2^{\cdot^{\cdot^{\cdot^{2^c}}}}}$, the height of this tower essentially depends on quantifiers; this bound cannot be improved
   - in practice, satisfiability/validity often feasible, even for bigger formulas

# Today's Lecture

- Presburger arithmetic
- alternating automata

# Presburger Arithmetic

# Presburger Arithmetic

- Presburger Arithmetic refers to the first-order theory over $(\mathbb{N}, \{0, +, <\})$
- named in honor of Mojżesz Presburger, who introduced it in 1929
- formulas in this logic are derivable from the following grammar:

$$s, t ::= 0 \;\Big|\; x \;\Big|\; s + t$$

$$\phi, \psi ::= \top \;\Big|\; \bot \;\Big|\; s = t \;\Big|\; s < t \;\Big|\; \phi \wedge \psi \;\Big|\; \neg\psi \;\Big|\; \exists x.\phi$$

  where $x$ is a first-order variable
- valuations map first-order variables to $\mathbb{N}$

# Presburger Arithmetic

- Presburger Arithmetic refers to the first-order theory over $(\mathbb{N}, \{0, +, <\})$
- named in honor of Mojżesz Presburger, who introduced it in 1929
- formulas in this logic are derivable from the following grammar:

$$s, t ::= 0 \mid x \mid s + t$$

$$\phi, \psi ::= \top \mid \bot \mid s = t \mid s < t \mid \phi \wedge \psi \mid \neg\psi \mid \exists x.\phi$$

  where $x$ is a first-order variable
- valuations map first-order variables to $\mathbb{N}$

## Applications

Presburger Arithmetic employed — due to the balance between expressiveness and algorithmic properties — e.g. in automated theorem proving and static program analysis

# Examples

$$s, t ::= 0 \mid x \mid s + t$$

$$\phi, \psi ::= \top \mid \bot \mid s = t \mid s < t \mid \phi \wedge \psi \mid \neg\psi \mid \exists x.\phi$$

- ⋆ $m$ is even: ?

# Examples

$$s, t ::= 0 \mid x \mid s + t$$

$$\phi, \psi ::= \top \mid \bot \mid s = t \mid s < t \mid \phi \wedge \psi \mid \neg\psi \mid \exists x.\phi$$

- $m$ is even: $\exists n. m = n + n$, or shorthand $\exists n. m = 2 \cdot n$
  - generally, multiplication by constant $c \in \mathbb{N}$ permissible

# Examples

$$s, t ::= 0 \;\big|\; x \;\big|\; s + t$$

$$\phi, \psi ::= \top \;\big|\; \bot \;\big|\; s = t \;\big|\; s < t \;\big|\; \phi \wedge \psi \;\big|\; \neg\psi \;\big|\; \exists x.\phi$$

- $m$ is even: $\exists n. m = n + n$, or shorthand $\exists n. m = 2 \cdot n$
  - generally, multiplication by constant $c \in \mathbb{N}$ permissible
- $m$ equals $1$: ?

# Examples

$$s, t ::= 0 \mid x \mid s + t$$

$$\phi, \psi ::= \top \mid \bot \mid s = t \mid s < t \mid \phi \wedge \psi \mid \neg \psi \mid \exists x.\phi$$

- $m$ is even: $\exists n. m = n + n$, or shorthand $\exists n. m = 2 \cdot n$
  - generally, multiplication by constant $c \in \mathbb{N}$ permissible
- $m$ equals $1$: $\forall n. n < m \rightarrow n = 0$

# Examples

$$s, t ::= 0 \mid x \mid s + t$$

$$\phi, \psi ::= \top \mid \bot \mid s = t \mid s < t \mid \phi \wedge \psi \mid \neg\psi \mid \exists x.\phi$$

- $m$ is even: $\exists n.m = n + n$, or shorthand $\exists n.m = 2 \cdot n$
  - generally, multiplication by constant $c \in \mathbb{N}$ permissible
- $m$ equals $1$: $\forall n.n < m \to n = 0$
- $m = r \bmod 5$: ?

# Examples

$$s, t ::= 0 \mid x \mid s + t$$

$$\phi, \psi ::= \top \mid \bot \mid s = t \mid s < t \mid \phi \wedge \psi \mid \neg\psi \mid \exists x.\phi$$

- ⋆ $m$ is even: $\exists n. m = n + n$, or shorthand $\exists n. m = 2 \cdot n$
  - generally, multiplication by constant $c \in \mathbb{N}$ permissible
- ⋆ $m$ equals $1$: $\forall n. n < m \rightarrow n = 0$
- ⋆ $m = r \bmod 5$: $\exists n. r < 5 \wedge m = 5 \cdot n + r$

# Examples

$$s, t ::= 0 \mid x \mid s + t$$

$$\phi, \psi ::= \top \mid \bot \mid s = t \mid s < t \mid \phi \wedge \psi \mid \neg\psi \mid \exists x.\phi$$

- $m$ is even: $\exists n. m = n + n$, or shorthand $\exists n. m = 2 \cdot n$
  - generally, multiplication by constant $c \in \mathbb{N}$ permissible
- $m$ equals $1$: $\forall n. n < m \rightarrow n = 0$
- $m = r \bmod 5$: $\exists n. r < 5 \wedge m = 5 \cdot n + r$
- the system of linear equations

$$m + n = 13$$
$$m - n = 1$$

has a solution: ?

## Examples

$$s, t ::= 0 \mid x \mid s + t$$

$$\phi, \psi ::= \top \mid \bot \mid s = t \mid s < t \mid \phi \wedge \psi \mid \neg\psi \mid \exists x.\phi$$

- $m$ is even: $\exists n. m = n + n$, or shorthand $\exists n. m = 2 \cdot n$
  - generally, multiplication by constant $c \in \mathbb{N}$ permissible
- $m$ equals $1$: $\forall n. n < m \to n = 0$
- $m = r \bmod 5$: $\exists n. r < 5 \wedge m = 5 \cdot n + r$
- the system of linear equations

$$m + n = 13$$
$$m - n = 1$$

has a solution: $\exists m. \exists n. m + n = 13 \wedge m = 1 + n$

# A Decision Procedure for Presburger Arithmetic

## General Idea

1. encode natural numbers as binary words (lsb-first order)
   - assignments $\alpha : \mathcal{V} \to \{0, \ldots, 2^m\}$ over $\{x_1, \ldots, x_n\}$ become binary matrices $\underline{\alpha} \in \{0, 1\}^{(m,n)}$

| | $\alpha(x_i)$ | $\underline{\alpha}$ |
|---|---|---|
| $x_1$ | 13 | |
| $x_2$ | 1 | |
| $x_3$ | 3 | |

$$\underline{\alpha} = \begin{pmatrix}1\\1\\1\end{pmatrix}\begin{pmatrix}0\\0\\1\end{pmatrix}\begin{pmatrix}1\\0\\0\end{pmatrix}\begin{pmatrix}1\\0\\0\end{pmatrix}$$

# A Decision Procedure for Presburger Arithmetic

## General Idea

1. encode natural numbers as binary words (lsb-first order)
   - assignments $\alpha : \mathcal{V} \to \{0, \ldots, 2^m\}$ over $\{x_1, \ldots, x_n\}$ become binary matrices $\underline{\alpha} \in \{0,1\}^{(m,n)}$

| | $\alpha(x_i)$ | $\underline{\alpha}$ |
|---|---|---|
| $x_1$ | 13 | $\begin{pmatrix}1\\1\\1\end{pmatrix}\begin{pmatrix}0\\0\\1\end{pmatrix}\begin{pmatrix}1\\0\\0\end{pmatrix}\begin{pmatrix}1\\0\\0\end{pmatrix}$ |
| $x_2$ | 1 | |
| $x_3$ | 3 | |

2. for formula $\phi$, define a DFA $\mathcal{A}_\phi$ recognizing precisely codings $\underline{\alpha}$ of valuations $\alpha$ making $\phi$ become true

# Language of a Formula

let us denote by $\hat{\mathsf{L}}(\phi)$ the language of coded valuations making $\phi$ true:

$$\hat{\mathsf{L}}(\phi) \triangleq \{\underline{\alpha} \mid \alpha \models \phi\}$$

# Language of a Formula

let us denote by $\hat{\mathsf{L}}(\phi)$ the language of coded valuations making $\phi$ true:

$$\hat{\mathsf{L}}(\phi) \triangleq \{\underline{\alpha} \mid \alpha \models \phi\}$$

**Lemma**

*For any formula $\phi$ in Presburger Arithmetic, $\hat{\mathsf{L}}(\phi)$ is regular.*

# Language of a Formula

let us denote by $\hat{\mathsf{L}}(\phi)$ the language of coded valuations making $\phi$ true:

$$\hat{\mathsf{L}}(\phi) \triangleq \{\underline{\alpha} \mid \alpha \models \phi\}$$

Lemma

*For any formula $\phi$ in Presburger Arithmetic, $\hat{\mathsf{L}}(\phi)$ is regular.*

Proof Outline.

By induction on the structure of $\phi$, we construct a DFA $\mathcal{A}_\phi$ recognizing $\hat{\mathsf{L}}(\phi)$.

# Language of a Formula

let us denote by $\hat{\mathsf{L}}(\phi)$ the language of coded valuations making $\phi$ true:

$$\hat{\mathsf{L}}(\phi) \triangleq \{\underline{\alpha} \mid \alpha \models \phi\}$$

**Lemma**

*For any formula $\phi$ in Presburger Arithmetic, $\hat{\mathsf{L}}(\phi)$ is regular.*

**Proof Outline.**

By induction on the structure of $\phi$, we construct a DFA $\mathcal{A}_\phi$ recognizing $\hat{\mathsf{L}}(\phi)$.

- ⋆ $\phi = \top$, $\phi = \bot$: In these cases $\hat{\mathsf{L}}(\phi)$ is easily seen to be regular.
- ⋆ $\phi = (s < t)$ or $\phi = (s = t)$: A corresponding automaton can be constructed (next slide).
- ⋆ $\phi = \neg\phi$ or $\phi = \psi_1 \wedge \psi_2$ From the induction hypothesis, using DFA-complementation and DFA-intersection.
- ⋆ $\phi = \forall x.\psi$: Elimination similar to construction for WMSO formulas.

# Recognizing $s < t$

- an inequality $s < t$ can be represented as $\sum_i a_i \cdot x_i < b$ where $a_i, b \in \mathbb{Z}$

$$2 \cdot x_1 < x_2 + 2 \quad \Longrightarrow \quad 2 \cdot x_1 - 1 \cdot x_2 < 2$$

# Recognizing $s < t$

- an inequality $s < t$ can be represented as $\sum_i a_i \cdot x_i < b$ where $a_i, b \in \mathbb{Z}$

$$2 \cdot x_1 < x_2 + 2 \quad \Longrightarrow \quad 2 \cdot x_1 - 1 \cdot x_2 < 2$$

- the automaton $\mathcal{A}_{s\,<\,t}$ recognizing $s < t$ is defined as follows
  - states $Q$ are inequalities of the form $\langle \sum_i a_i \cdot x_i < d \rangle$

    Intuition: $\mathrm{L}(\langle \sum_i a_i \cdot x_i < d \rangle) = \{\underline{\alpha} \mid \alpha \models \sum_i a_i \cdot x_i < d\}$

# Recognizing $s < t$

- an inequality $s < t$ can be represented as $\sum_i a_i \cdot x_i < b$ where $a_i, b \in \mathbb{Z}$

$$2 \cdot x_1 < x_2 + 2 \quad \Longrightarrow \quad 2 \cdot x_1 - 1 \cdot x_2 < 2$$

- the automaton $\mathcal{A}_{s\,<\,t}$ recognizing $s < t$ is defined as follows
  - states $Q$ are inequalities of the form $\langle \sum_i a_i \cdot x_i < d \rangle$

    Intuition: $\mathrm{L}(\langle \sum_i a_i \cdot x_i < d \rangle) = \{\underline{\alpha} \mid \alpha \models \sum_i a_i \cdot x_i < d\}$
  - the initial state $q_I$ is given by the representation of $s < t$

# Recognizing $s < t$

- an inequality $s < t$ can be represented as $\sum_i a_i \cdot x_i < b$ where $a_i, b \in \mathbb{Z}$

$$2 \cdot x_1 < x_2 + 2 \quad \Longrightarrow \quad 2 \cdot x_1 - 1 \cdot x_2 < 2$$

- the automaton $\mathcal{A}_{s\,<\,t}$ recognizing $s < t$ is defined as follows
  - states $Q$ are inequalities of the form $\langle \sum_i a_i \cdot x_i < d \rangle$
    Intuition: $\mathrm{L}(\langle \sum_i a_i \cdot x_i < d \rangle) = \{\underline{\alpha} \mid \alpha \models \sum_i a_i \cdot x_i < d\}$
  - the initial state $q_I$ is given by the representation of $s < t$
  - the transition function $\delta$ is given by

$$\delta\left(\langle \textstyle\sum_i a_i \cdot x_i < d \rangle, \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}\right) \triangleq \langle \textstyle\sum_i a_i \cdot x_i < \left\lceil \frac{1}{2} \left(d - \sum_i a_i \cdot b_i\right) \right\rceil \rangle$$

since $\sum_i a_i \cdot (b_i + 2 \cdot x_i') < d \iff \sum_i a_i \cdot x_i' < \frac{1}{2} \cdot \left(d - \sum_i a_i \cdot b_i\right)$

# Recognizing $s < t$

- an inequality $s < t$ can be represented as $\sum_i a_i \cdot x_i < b$ where $a_i, b \in \mathbb{Z}$

$$2 \cdot x_1 < x_2 + 2 \quad \Longrightarrow \quad 2 \cdot x_1 - 1 \cdot x_2 < 2$$

- the automaton $\mathcal{A}_{s\,<\,t}$ recognizing $s < t$ is defined as follows
  - states $Q$ are inequalities of the form $\langle \sum_i a_i \cdot x_i < d \rangle$
    Intuition: $\mathrm{L}(\langle \sum_i a_i \cdot x_i < d \rangle) = \{\underline{\alpha} \mid \alpha \models \sum_i a_i \cdot x_i < d\}$
  - the initial state $q_I$ is given by the representation of $s < t$
  - the transition function $\delta$ is given by

$$\delta\left(\langle \textstyle\sum_i a_i \cdot x_i < d \rangle, \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}\right) \triangleq \langle \textstyle\sum_i a_i \cdot x_i < \left\lceil \frac{1}{2} \left(d - \sum_i a_i \cdot b_i\right) \right\rceil \rangle$$

    since $\sum_i a_i \cdot (b_i + 2 \cdot x_i') < d \Leftrightarrow \sum_i a_i \cdot x_i' < \frac{1}{2} \cdot \left(d - \sum_i a_i \cdot b_i\right)$
  - final states are all those states $\sum_i a_i \cdot x_i < d$ with $0 < d$

# Recognizing $s<t$

- an inequality $s<t$ can be represented as $\sum_i a_i \cdot x_i < b$ where $a_i, b \in \mathbb{Z}$

$$2 \cdot x_1 < x_2 + 2 \quad \Longrightarrow \quad 2 \cdot x_1 - 1 \cdot x_2 < 2$$

- the automaton $\mathcal{A}_{s<t}$ recognizing $s<t$ is defined as follows
  - states $Q$ are inequalities of the form $\langle \sum_i a_i \cdot x_i < d \rangle$
    Intuition: $\mathrm{L}(\langle \sum_i a_i \cdot x_i < d \rangle) = \{\underline{\alpha} \mid \alpha \models \sum_i a_i \cdot x_i < d\}$
  - the initial state $q_I$ is given by the representation of $s<t$
  - the transition function $\delta$ is given by

$$\delta\left(\langle \textstyle\sum_i a_i \cdot x_i < d \rangle, \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}\right) \triangleq \langle \textstyle\sum_i a_i \cdot x_i < \left\lceil \frac{1}{2} \left(d - \sum_i a_i \cdot b_i\right) \right\rceil \rangle$$

    since $\sum_i a_i \cdot (b_i + 2 \cdot x_i') < d \Leftrightarrow \sum_i a_i \cdot x_i' < \frac{1}{2} \cdot (d - \sum_i a_i \cdot b_i)$
  - final states are all those states $\sum_i a_i \cdot x_i < d$ with $0 < d$

- finiteness: from initial state $\sum_i a_i \cdot x_i < d$, only $\sum_i a_i + d$ states reachable, hence the overall construction is finite

# Recognizing $s = t$

- an inequality $s = t$ can be represented as $\sum_i a_i \cdot x_i = b$ where $a_i, b \in \mathbb{Z}$

$$2 \cdot x_1 = x_2 + 2 \quad \Longrightarrow \quad 2 \cdot x_1 - 1 \cdot x_2 = 2$$

- the automaton $\mathcal{A}_{s\,=\,t}$ recognizing $s = t$ is defined as follows
  - states $Q$ are inequalities of the form $\langle \sum_i a_i \cdot x_i = d \rangle$
    Intuition: $\mathrm{L}(\langle \sum_i a_i \cdot x_i = d \rangle) = \{\underline{\alpha} \mid \alpha \models \sum_i a_i \cdot x_i = d\}$
  - the initial state $q_I$ is given by the representation of $s = t$
  - the transition function $\delta$ is given by

$$\delta\left(\langle \textstyle\sum_i a_i \cdot x_i = d \rangle, \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}\right) \triangleq \begin{cases} \langle \sum_i a_i \cdot x_i = \frac{1}{2}\left(d - \sum_i a_i \cdot b_i\right) \rangle & \text{if } d - \sum_i a_i \cdot b_i \text{ even,} \\ q_{fail} & \text{otherwise.} \end{cases}$$

    since $\sum_i a_i \cdot (b_i + 2 \cdot x_i') = d \iff \sum_i a_i \cdot x_i' = \frac{1}{2} \cdot \left(d - \sum_i a_i \cdot b_i\right)$
  - final states are all those states $\sum_i a_i \cdot x_i = d$ with $0 = d$
- finiteness: from initial state $\sum_i a_i \cdot x_i = d$, only $\sum_i a_i + d$ states reachable, hence the overall construction is finite

# Decision Problems for Presburger Arithmetic

The Satisfiability Problem

- ⋆ Given: formula $\phi$
- ⋆ Question: is there $\alpha$ s.t $\alpha \models \phi$?

The Validity Problem

- ⋆ Given: formula $\phi$
- ⋆ Question: $\alpha \models \phi$ for all assignments $\alpha$?

# Decision Problems for Presburger Arithmetic

The Satisfiability Problem

- ⋆ Given: formula $\phi$
- ⋆ Question: is there $\alpha$ s.t $\alpha \models \phi$?

The Validity Problem

- ⋆ Given: formula $\phi$
- ⋆ Question: $\alpha \models \phi$ for all assignments $\alpha$?

**Theorem**

*Satisfiability and Validity are decidable for Presburger Arithmetic.*

**Theorem**

*For any formula $\phi$, the constructed DFA recognizing $\hat{\mathsf{L}}(\phi)$ has size $\mathsf{O}(2^{2^n})$.*

# Peano Arithmetic

- Peano's arithmetic is the first-order theory natural integers with vocabulary $\{+, \times, <\}$

# Peano Arithmetic

- Peano's arithmetic is the first-order theory natural integers with vocabulary $\{+, \times, <\}$
- its existential fragment corresponds to the Diophantine equations, i.e., polynomial equations on integers
- Hilbert's 10th problem was to solve Diophantine equations

# Peano Arithmetic

- Peano's arithmetic is the first-order theory natural integers with vocabulary $\{+, \times, <\}$
- its existential fragment corresponds to the Diophantine equations, i.e., polynomial equations on integers
- Hilbert's 10th problem was to solve Diophantine equations
- Youri Matiassevitch, drawing on the work of Julia Robinson, demonstrated that this was an undecidable problem

## Skolem Arithmetic

- Skolem's arithmetic is the first order theory of natural integers with the vocabulary $\{\times, =\}$

# Skolem Arithmetic

- ⋆ Skolem's arithmetic is the first order theory of natural integers with the vocabulary $\{\times, =\}$
- ⋆ Skolem's arithmetic is also decidable
- ⋆ proof goes via reduction to tree automata, closely resembling the proof we have just seen for Presburger's arithmetic

# Alternating Automata

# Angelican vs Demonic Non-Determinism

What is a non-deterministic machine (or automaton)?

- a “machine” which admits several executions on the same input
- put otherwise, during processing, several choices are possible

# Angelican vs Demonic Non-Determinism

What is a non-deterministic machine (or automaton)?

- ⋆ a “machine” which admits several executions on the same input
- ⋆ put otherwise, during processing, several choices are possible
- ⋆ such choices can be resolved in favor (anglican non-determinism) or against (demonic non-determinism) a positive outcome (e.g. acceptance, termination, etc)

# Angelican vs Demonic Non-Determinism

What is a non-deterministic machine (or automaton)?

- ⋆ a "machine" which admits several executions on the same input
- ⋆ put otherwise, during processing, several choices are possible
- ⋆ such choices can be resolved in favor (anglican non-determinism) or against (demonic non-determinism) a positive outcome (e.g. acceptance, termination, etc)
  - Anglican: an angel resolves choices

    $\Longrightarrow$ it is sufficient to have one "good" execution path, to have a positive outcome
  - Demonic: a demon resolves choices

    $\Longrightarrow$ all execution paths must be "good", to have a positive outcome

# Angelican vs Demonic Non-Determinism

What is a non-deterministic machine (or automaton)?

- ⋆ a "machine" which admits several executions on the same input
- ⋆ put otherwise, during processing, several choices are possible
- ⋆ such choices can be resolved in favor (anglican non-determinism) or against (demonic non-determinism) a positive outcome (e.g. acceptance, termination, etc)
  - Anglican: an angel resolves choices

    ⇒ it is sufficient to have one "good" execution path, to have a positive outcome
  - Demonic: a demon resolves choices

    ⇒ all execution paths must be "good", to have a positive outcome

Example

- ⋆ NFAs are based on anglican non-determinism
- ⋆ worst-case complexity analysis assumes demonic non-determinism

# NFAs with Demonic Choice

- ⋆ NFAs incorporate angelic non-determinism because, in order for $w \in \mathsf{L}(\mathcal{A})$, only one accepting run of $w$ has to exists

# NFAs with Demonic Choice

- NFAs incorporate angelic non-determinism because, in order for $w \in \mathsf{L}(\mathcal{A})$, only one accepting run of $w$ has to exists

- demonic non-determinism introduced by re-formulating the acceptance condition

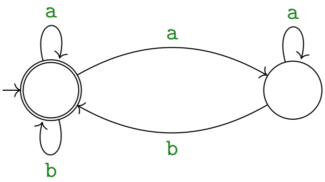
$$\mathsf{L}^{-}(\mathcal{A}) \triangleq \{w \mid \text{all runs on } w \text{ are accepting}\}$$

# NFAs with Demonic Choice

- ⋆ NFAs incorporate angelic non-determinism because, in order for $w \in L(\mathcal{A})$, only one accepting run of $w$ has to exists
- ⋆ demonic non-determinism introduced by re-formulating the acceptance condition

$$L^{-}(\mathcal{A}) \triangleq \{w \mid \text{all runs on } w \text{ are accepting}\}$$

Example
Consider automaton $\mathcal{A}$ over $\Sigma = \{a, b\}$




- ⋆ $L(\mathcal{A}) = \Sigma^*$
- ⋆ $L^{-}(\mathcal{A}) = \epsilon \cup \Sigma^* \cdot b$ (why?)

## Duality of Non-Determinism

- recall that for each NFA $\mathcal{A}$, its dual $\overline{\mathcal{A}}$ is given by complementing final states
- in general, only when $\mathcal{A}$ is deterministic, then $L(\overline{\mathcal{A}}) = \overline{L(\mathcal{A})}$

# Duality of Non-Determinism

- recall that for each NFA $\mathcal{A}$, its dual $\overline{\mathcal{A}}$ is given by complementing final states
- in general, only when $\mathcal{A}$ is deterministic, then $\mathsf{L}(\overline{\mathcal{A}}) = \overline{\mathsf{L}(\mathcal{A})}$

**Proposition**

$$w \in \mathsf{L}(\mathcal{A}) \quad \Longleftrightarrow \quad w \notin \mathsf{L}^{-}(\overline{\mathcal{A}})$$

# Duality of Non-Determinism

- recall that for each NFA $\mathcal{A}$, its dual $\overline{\mathcal{A}}$ is given by complementing final states
- in general, only when $\mathcal{A}$ is deterministic, then $\mathsf{L}(\overline{\mathcal{A}}) = \overline{\mathsf{L}(\mathcal{A})}$

**Proposition**

$$w \in \mathsf{L}(\mathcal{A}) \quad \Longleftrightarrow \quad w \notin \mathsf{L}^{-}(\overline{\mathcal{A}})$$

- regime to resolve non-determinism has no effect on expressiveness of NFAs
- although potentially on the conciseness of the language description through NFAs

# Duality of Non-Determinism

- recall that for each NFA $\mathcal{A}$, its dual $\overline{\mathcal{A}}$ is given by complementing final states
- in general, only when $\mathcal{A}$ is deterministic, then $\mathsf{L}(\overline{\mathcal{A}}) = \overline{\mathsf{L}(\mathcal{A})}$

**Proposition**

$$w \in \mathsf{L}(\mathcal{A}) \quad \Longleftrightarrow \quad w \notin \mathsf{L}^{\overline{\ }}(\overline{\mathcal{A}})$$

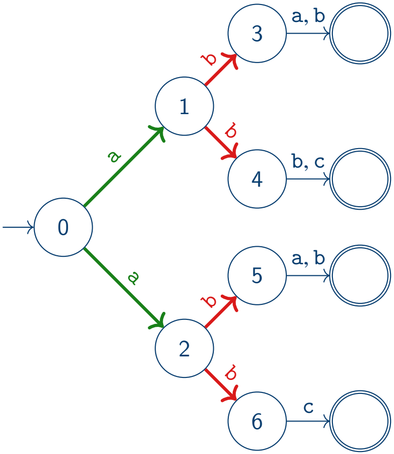
- regime to resolve non-determinism has no effect on expressiveness of NFAs
- although potentially on the conciseness of the language description through NFAs

what happens if we leave regime internal to the automata?

# Alternating Finite Automata

# Alternating Finite Automata

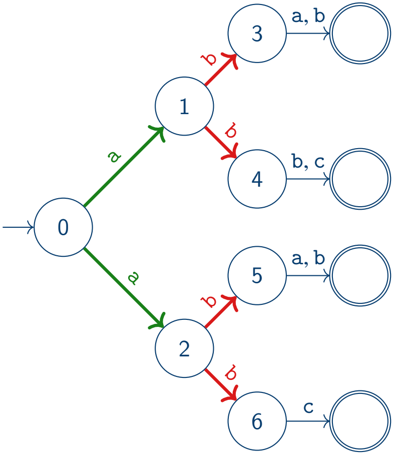
- ⋆ General Idea: mix Anglican an Demonic choice on the level of individual transitions




$$\delta(0, a) = 1 \vee 2$$
$$\delta(1, b) = 3 \wedge 4$$
$$\delta(2, b) = 5 \wedge 6$$
$$\vdots$$

# Alternating Finite Automata

- ⋆ General Idea: mix Anglican an Demonic choice on the level of individual transitions




$$\delta(0, a) = 1 \vee 2$$
$$\delta(1, b) = 3 \wedge 4$$
$$\delta(2, b) = 5 \wedge 6$$
$$\vdots$$

$$L(\mathcal{A}) = a\Big(\overbrace{b\big(\underbrace{a \cup b}_{L(3)}\big) \cap b\big(\underbrace{b \cup c}_{L(4)}\big)}^{L(1)}\Big) \cup a\Big(\underbrace{b\big(\underbrace{a \cup b}_{L(5)}\big) \cap b\,\underbrace{c}_{L(6)}}_{L(2)}\Big)$$

$$= abb \cup \emptyset$$

$$= abb$$

# Alternating Finite Automata, Formally

## Positive Boolean Formulas

- ⋆ let $A = \{a, b, \dots\}$ be a set of atoms
- ⋆ the positive Boolean formulas $\mathbb{B}^+(A)$ over atoms $A$ are given by the following grammar:

$$\phi, \psi ::= a \;\Big|\; \phi \wedge \psi \;\Big|\; \phi \vee \psi$$

  - such formulas are called positive because negation is missing

# Alternating Finite Automata, Formally

## Positive Boolean Formulas

- let $A = \{a, b, \ldots\}$ be a set of atoms
- the positive Boolean formulas $\mathbb{B}^+(A)$ over atoms $A$ are given by the following grammar:

$$\phi, \psi ::= a \;\Big|\; \phi \wedge \psi \;\Big|\; \phi \vee \psi$$

  - such formulas are called positive because negation is missing

- a set $M \subseteq A$ is a model of $\phi$ if $M \models \phi$ where

$$M \models a :\Longleftrightarrow a \in M \qquad M \models \phi \wedge \psi :\Longleftrightarrow M \models \phi \textbf{ and } M \models \psi \qquad M \models \phi \vee \psi :\Longleftrightarrow M \models \phi \textbf{ or } M \models \psi$$

# Alternating Finite Automata, Formally

## Positive Boolean Formulas

- ⋆ let $A = \{a, b, \dots\}$ be a set of atoms
- ⋆ the positive Boolean formulas $\mathbb{B}^+(A)$ over atoms $A$ are given by the following grammar:

$$\phi, \psi ::= a \;\Big|\; \phi \wedge \psi \;\Big|\; \phi \vee \psi$$

  - such formulas are called positive because negation is missing

- ⋆ a set $M \subseteq A$ is a model of $\phi$ if $M \vDash \phi$ where

$$M \vDash a :\Longleftrightarrow a \in M \quad M \vDash \phi \wedge \psi :\Longleftrightarrow M \vDash \phi \textbf{ and } M \vDash \psi \quad M \vDash \phi \vee \psi :\Longleftrightarrow M \vDash \phi \textbf{ or } M \vDash \psi$$

### Example

consider $\phi = a \wedge (b \vee c)$, then

$$\{a, b\} \vDash \phi \qquad \{a, c\} \vDash \phi \qquad \{a\} \nvDash \phi \qquad \{b, c\} \nvDash \phi$$

# Alternating Finite Automata, Formally (II)

an alternating finite automata (AFA) is a tuple $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$ where all components are identical to an NFA except that

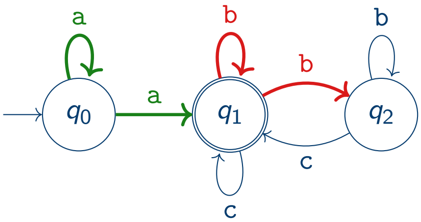
$$\delta : Q \times \Sigma \to \mathbb{B}^+(Q)$$

# Alternating Finite Automata, Formally (II)

an alternating finite automata (AFA) is a tuple $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$ where all components are identical to an NFA except that

$$\delta : Q \times \Sigma \to \mathbb{B}^+(Q)$$

Example




| $\delta$ | a | b | c |
|---|---|---|---|
| $q_0$ | $q_0 \vee q_1$ | $q_\perp$ | $q_\perp$ |
| $q_1$ | $q_\perp$ | $q_1 \wedge q_2$ | $q_1$ |
| $q_2$ | $q_\perp$ | $q_2$ | $q_1$ |
| $q_\perp$ | $q_\perp$ | $q_\perp$ | $q_\perp$ |

# Runs in an AFA

let $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$ be an AFA

- an execution for a word $w = \mathtt{a}_1 \ldots \mathtt{a}_n \in \Sigma^*$ is a tree $T_w$ whose nodes are labeled by states $Q$ s.t.:
  1. the root node of $T_w$ is labeled by the initial state $q_I$
  2. for all nodes $v$ labeled by $q$ on the $i$th layer $(i = 0, \ldots, n-1)$ and successors $v_1, \ldots, v_k$ on layer $i+1$, labeled by $q_1, \ldots, q_k$, respectively:

$$\{q_1, \ldots, q_k\} \models \delta(q, \mathtt{a}_{i+1})$$

# Runs in an AFA

let $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$ be an AFA

- ⋆ an execution for a word $w = \mathtt{a}_1 \ldots \mathtt{a}_n \in \Sigma^*$ is a tree $T_w$ whose nodes are labeled by states $Q$ s.t.:
  1. the root node of $T_w$ is labeled by the initial state $q_I$
  2. for all nodes $v$ labeled by $q$ on the $i$th layer $(i = 0, \ldots, n-1)$ and successors $v_1, \ldots, v_k$ on layer $i+1$, labeled by $q_1, \ldots, q_k$, respectively:

$$\{q_1, \ldots, q_k\} \models \delta(q, \mathtt{a}_{i+1})$$

- ⋆ an execution is accepting if all leafs are labeled by final states

## Runs in an AFA

let $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$ be an AFA

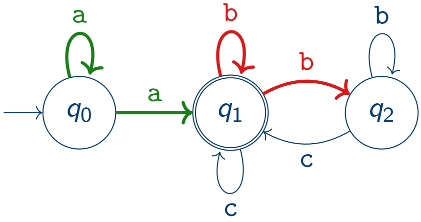
- ★ an execution for a word $w = \mathtt{a}_1 \ldots \mathtt{a}_n \in \Sigma^*$ is a tree $T_w$ whose nodes are labeled by states $Q$ s.t.:
  1. the root node of $T_w$ is labeled by the initial state $q_I$
  2. for all nodes $v$ labeled by $q$ on the $i$th layer $(i = 0, \ldots, n-1)$ and successors $v_1, \ldots, v_k$ on layer $i+1$, labeled by $q_1, \ldots, q_k$, respectively:

$$\{q_1, \ldots, q_k\} \vDash \delta(q, \mathtt{a}_{\mathtt{i+1}})$$

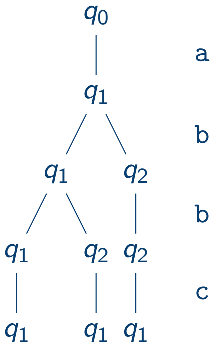
- ★ an execution is accepting if all leafs are labeled by final states
- ★ the language recognized by $\mathcal{A}$ is given by

$$\mathrm{L}(\mathcal{A}) \triangleq \{w \mid \text{there exists an accepting execution } T_w \text{ for } w\}$$

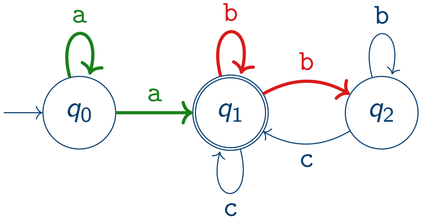
# Example of Accepting Execution for $w = \texttt{abbc}$



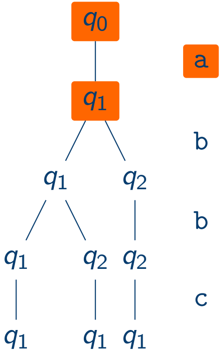

| $\delta$ | $a$ | $b$ | $c$ |
|---|---|---|---|
| $q_0$ | $q_0 \vee q_1$ | $q_\perp$ | $q_\perp$ |
| $q_1$ | $q_\perp$ | $q_1 \wedge q_2$ | $q_1$ |
| $q_2$ | $q_\perp$ | $q_2$ | $q_1$ |
| $q_\perp$ | $q_\perp$ | $q_\perp$ | $q_\perp$ |



# Example of Accepting Execution for $w = \mathtt{abbc}$

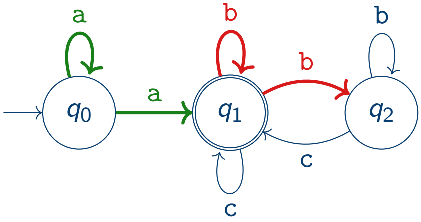



| $\delta$ | $a$ | $b$ | $c$ |
|---|---|---|---|
| $q_0$ | $q_0 \vee q_1$ | $q_\bot$ | $q_\bot$ |
| $q_1$ | $q_\bot$ | $q_1 \wedge q_2$ | $q_1$ |
| $q_2$ | $q_\bot$ | $q_2$ | $q_1$ |
| $q_\bot$ | $q_\bot$ | $q_\bot$ | $q_\bot$ |

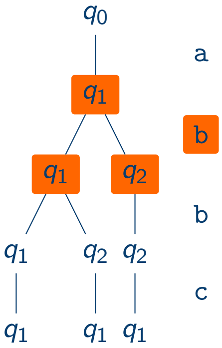



$\{q_1\} \models q_0 \vee q_1$

# Example of Accepting Execution for $w = \texttt{abbc}$

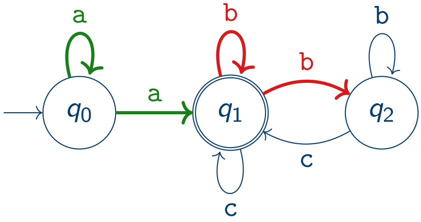



| $\delta$ | $a$ | $b$ | $c$ |
|---|---|---|---|
| $q_0$ | $q_0 \vee q_1$ | $q_\perp$ | $q_\perp$ |
| $q_1$ | $q_\perp$ | $q_1 \wedge q_2$ | $q_1$ |
| $q_2$ | $q_\perp$ | $q_2$ | $q_1$ |
| $q_\perp$ | $q_\perp$ | $q_\perp$ | $q_\perp$ |

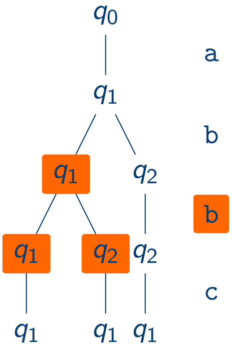



$$\{q_1, q_2\} \models q_1 \wedge q_2$$

# Example of Accepting Execution for $w = \texttt{abbc}$

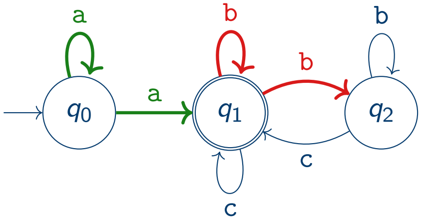



| $\delta$ | $a$ | $b$ | $c$ |
|---|---|---|---|
| $q_0$ | $q_0 \vee q_1$ | $q_\perp$ | $q_\perp$ |
| $q_1$ | $q_\perp$ | $q_1 \wedge q_2$ | $q_1$ |
| $q_2$ | $q_\perp$ | $q_2$ | $q_1$ |
| $q_\perp$ | $q_\perp$ | $q_\perp$ | $q_\perp$ |

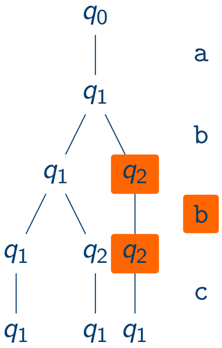



$\{q_1, q_2\} \models q_1 \wedge q_2$

# Example of Accepting Execution for $w = \texttt{abbc}$

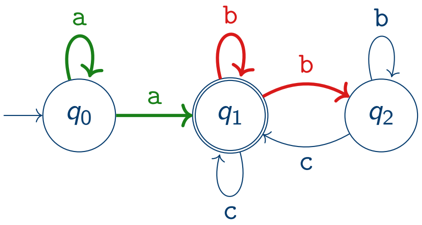



| $\delta$ | $a$ | $b$ | $c$ |
|---|---|---|---|
| $q_0$ | $q_0 \vee q_1$ | $q_\bot$ | $q_\bot$ |
| $q_1$ | $q_\bot$ | $q_1 \wedge q_2$ | $q_1$ |
| $q_2$ | $q_\bot$ | $q_2$ | $q_1$ |
| $q_\bot$ | $q_\bot$ | $q_\bot$ | $q_\bot$ |

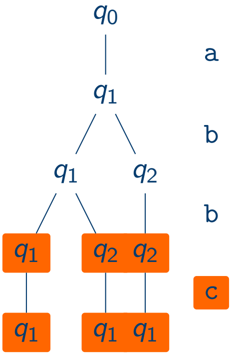



$\{q_2\} \models q_2$

# Example of Accepting Execution for $w = \texttt{abbc}$

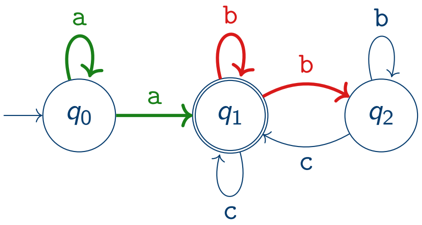



| $\delta$ | $a$ | $b$ | $c$ |
|---|---|---|---|
| $q_0$ | $q_0 \vee q_1$ | $q_\perp$ | $q_\perp$ |
| $q_1$ | $q_\perp$ | $q_1 \wedge q_2$ | $q_1$ |
| $q_2$ | $q_\perp$ | $q_2$ | $q_1$ |
| $q_\perp$ | $q_\perp$ | $q_\perp$ | $q_\perp$ |

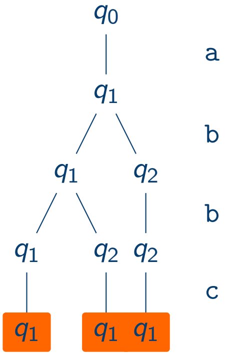



$\{q_1\} \models q_1$

# Example of Accepting Execution for $w = \texttt{abbc}$




| $\delta$ | $a$ | $b$ | $c$ |
|---|---|---|---|
| $q_0$ | $q_0 \vee q_1$ | $q_\perp$ | $q_\perp$ |
| $q_1$ | $q_\perp$ | $q_1 \wedge q_2$ | $q_1$ |
| $q_2$ | $q_\perp$ | $q_2$ | $q_1$ |
| $q_\perp$ | $q_\perp$ | $q_\perp$ | $q_\perp$ |




$\{q_1, q_1, q_1\} \subseteq F$

# Extended Transition Function

the extended transition function

$$\hat{\delta} : \mathbb{B}^+(Q) \times \Sigma^* \to \mathbb{B}^+(Q)$$

is recursively defined by:

$$\hat{\delta}(q, \epsilon) \triangleq q \qquad \hat{\delta}(\phi \vee \psi, w) = \hat{\delta}(\phi, w) \vee \hat{\delta}(\psi, w)$$

$$\hat{\delta}(q, \mathrm{a} \cdot w) \triangleq \hat{\delta}(\delta(q, \mathrm{a}), w) \qquad \hat{\delta}(\phi \wedge \psi, w) = \hat{\delta}(\phi, w) \wedge \hat{\delta}(\psi, w)$$

# Extended Transition Function

the extended transition function

$$\hat{\delta} : \mathbb{B}^+(Q) \times \Sigma^* \to \mathbb{B}^+(Q)$$

is recursively defined by:

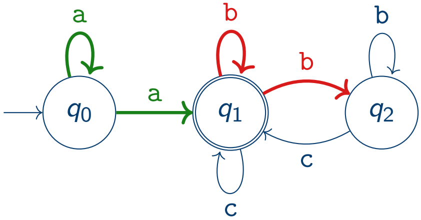
$$\hat{\delta}(q, \epsilon) \triangleq q \qquad \hat{\delta}(\phi \vee \psi, w) = \hat{\delta}(\phi, w) \vee \hat{\delta}(\psi, w)$$

$$\hat{\delta}(q, \mathrm{a} \cdot w) \triangleq \hat{\delta}(\delta(q, \mathrm{a}), w) \qquad \hat{\delta}(\phi \wedge \psi, w) = \hat{\delta}(\phi, w) \wedge \hat{\delta}(\psi, w)$$

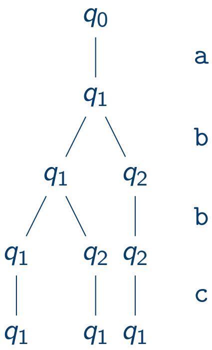
**Lemma**

$$\mathrm{L}(\mathcal{A}) = \{w \mid F \models \hat{\delta}(q_I, w)\}$$

# Example of Accepting Execution for $w = \texttt{abbc}$ (II)




| $\delta$ | $a$ | $b$ | $c$ |
|---|---|---|---|
| $q_0$ | $q_0 \vee q_1$ | $q_\bot$ | $q_\bot$ |
| $q_1$ | $q_\bot$ | $q_1 \wedge q_2$ | $q_1$ |
| $q_2$ | $q_\bot$ | $q_2$ | $q_1$ |
| $q_\bot$ | $q_\bot$ | $q_\bot$ | $q_\bot$ |




$$\begin{aligned}
\hat{\delta}(q_0, \texttt{abbc}) &= \hat{\delta}(q_0 \vee q_1, \texttt{bbc}) \\
&= \hat{\delta}(q_0, \texttt{bbc}) \vee \hat{\delta}(q_1, \texttt{bbc}) \\
&= \hat{\delta}(q_\bot, \texttt{bc}) \vee (\hat{\delta}(q_1, \texttt{bc}) \wedge \hat{\delta}(q_2, \texttt{bc})) \\
&= \hat{\delta}(q_\bot, \texttt{c}) \vee (\hat{\delta}(q_1, \texttt{c}) \wedge \hat{\delta}(q_2, \texttt{c})) \\
&= \hat{\delta}(q_\bot, \epsilon) \vee \hat{\delta}(q_1, \epsilon) \\
&= q_\bot \vee q_1
\end{aligned}$$

$$\{q_1\} \models q_\bot \vee q_1$$

# Comparison to NFAs and DFAs

- AFAs generalise NFAs
  - every DFA is a NFA is an AFA

# Comparison to NFAs and DFAs

- AFAs generalise NFAs
  - every DFA is a NFA is an AFA
- AFAs allow often more succinct encoding / automata constructions

# Comparison to NFAs and DFAs

- AFAs generalise NFAs
  - every DFA is a NFA is an AFA
- AFAs allow often more succinct encoding / automata constructions

Example

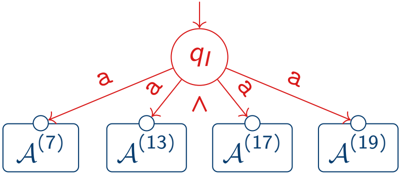
- let $\mathcal{A}^{(m)} = (Q^{(m)}, \{a\}, \delta^{(m)}, q_I^{(m)}, F^{(m)})$ be an NFA with $L(\mathcal{A}^{(m)}) = \{w \mid |w| = 0 \bmod m\}$
  - this NFA has at least $m$ states

# Comparison to NFAs and DFAs

- AFAs generalise NFAs
  - every DFA is a NFA is an AFA
- AFAs allow often more succinct encoding / automata constructions

Example

- let $\mathcal{A}^{(m)} = (Q^{(m)}, \{a\}, \delta^{(m)}, q_I^{(m)}, F^{(m)})$ be an NFA with $L(\mathcal{A}^{(m)}) = \{w \mid |w| = 0 \bmod m\}$
  - this NFA has at least $m$ states
- consider the AFA $\mathcal{A}$ defined from $\mathcal{A}^{(m)}$ for primes $m = 7, 13, 17, 19$ by

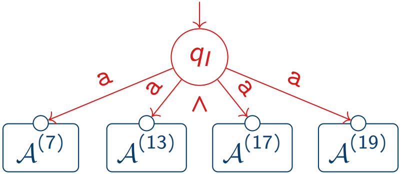


# Comparison to NFAs and DFAs

- AFAs generalise NFAs
  - every DFA is a NFA is an AFA
- AFAs allow often more succinct encoding / automata constructions

Example

- let $\mathcal{A}^{(m)} = (Q^{(m)}, \{a\}, \delta^{(m)}, q_I^{(m)}, F^{(m)})$ be an NFA with $L(\mathcal{A}^{(m)}) = \{w \mid |w| = 0 \bmod m\}$
  - this NFA has at least $m$ states
- consider the AFA $\mathcal{A}$ defined from $\mathcal{A}^{(m)}$ for primes $m = 7, 13, 17, 19$ by



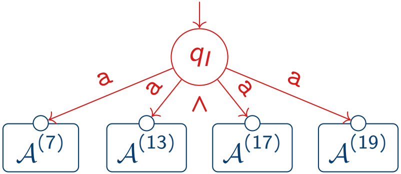

  - $L(\mathcal{A}) = \{w \mid |w| = 1 \bmod 29393\}$ since $29393 = 7 \cdot 13 \cdot 17 \cdot 19$

# Comparison to NFAs and DFAs

- AFAs generalise NFAs
  - every DFA is a NFA is an AFA
- AFAs allow often more succinct encoding / automata constructions

Example

- let $\mathcal{A}^{(m)} = (Q^{(m)}, \{a\}, \delta^{(m)}, q_I^{(m)}, F^{(m)})$ be an NFA with $L(\mathcal{A}^{(m)}) = \{w \mid |w| = 0 \bmod m\}$
  - this NFA has at least $m$ states
- consider the AFA $\mathcal{A}$ defined from $\mathcal{A}^{(m)}$ for primes $m = 7, 13, 17, 19$ by




  - $L(\mathcal{A}) = \{w \mid |w| = 1 \bmod 29393\}$ since $29393 = 7 \cdot 13 \cdot 17 \cdot 19$
  - AFA $\mathcal{A}$ has $57 = 1 + 7 + 13 + 17 + 19$, whereas a corresponding NFA needs 29393 states

# Complementation

- recall: NFA-complementation may blow-up automata sizes by an exponential

**Lemma**

*For every AFA $\mathcal{A}$ there exists an AFA $\overline{\mathcal{A}}$ of equal size such that* $\mathsf{L}(\overline{\mathcal{A}}) = \overline{\mathsf{L}(\mathcal{A})}$

# Complementation

- recall: NFA-complementation may blow-up automata sizes by an exponential

Lemma

*For every AFA $\mathcal{A}$ there exists an AFA $\overline{\mathcal{A}}$ of equal size such that* $\mathsf{L}(\overline{\mathcal{A}}) = \overline{\mathsf{L}(\mathcal{A})}$

Proof Outline.

- let $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$
- define the dual formula $\overline{\phi}$ of $\phi \in \mathbb{B}^+(Q)$ following De Morgans rules

$$\overline{q} \triangleq q \qquad \overline{\phi \vee \psi} \triangleq \overline{\phi} \wedge \overline{\psi} \qquad \overline{\phi \wedge \psi} \triangleq \overline{\phi} \vee \overline{\psi}$$

  - morally, $q \in Q$ re-used for their "negation"; we have (i) $M \models \phi$ iff $Q \setminus M \not\models \overline{\phi}$

# Complementation

- recall: NFA-complementation may blow-up automata sizes by an exponential

Lemma

*For every AFA $\mathcal{A}$ there exists an AFA $\overline{\mathcal{A}}$ of equal size such that* $\mathsf{L}(\overline{\mathcal{A}}) = \overline{\mathsf{L}(\mathcal{A})}$

Proof Outline.

- let $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$
- define the dual formula $\overline{\phi}$ of $\phi \in \mathbb{B}^+(Q)$ following De Morgans rules

$$\overline{q} \triangleq q \qquad \overline{\phi \vee \psi} \triangleq \overline{\phi} \wedge \overline{\psi} \qquad \overline{\phi \wedge \psi} \triangleq \overline{\phi} \vee \overline{\psi}$$

  - morally, $q \in Q$ re-used for their "negation"; we have (i) $M \vDash \phi$ iff $Q \setminus M \nvDash \overline{\phi}$
- we now define $\overline{\mathcal{A}} \triangleq (Q, \Sigma, \overline{\delta}, q_I, Q \setminus F)$ where $\overline{\delta}(q, \mathrm{a}) \triangleq \overline{\delta(q, \mathrm{a})}$ for all $q \in Q$, $\mathrm{a} \in \Sigma$

# Complementation

- recall: NFA-complementation may blow-up automata sizes by an exponential

**Lemma**

*For every AFA $\mathcal{A}$ there exists an AFA $\overline{\mathcal{A}}$ of equal size such that* $\mathrm{L}(\overline{\mathcal{A}}) = \overline{\mathrm{L}(\mathcal{A})}$

**Proof Outline.**

- let $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$
- define the dual formula $\overline{\phi}$ of $\phi \in \mathbb{B}^+(Q)$ following De Morgans rules

$$\overline{q} \triangleq q \qquad \overline{\phi \vee \psi} \triangleq \overline{\phi} \wedge \overline{\psi} \qquad \overline{\phi \wedge \psi} \triangleq \overline{\phi} \vee \overline{\psi}$$

  - morally, $q \in Q$ re-used for their "negation"; we have (i) $M \models \phi$ iff $Q \setminus M \not\models \overline{\phi}$
- we now define $\overline{\mathcal{A}} \triangleq (Q, \Sigma, \overline{\delta}, q_I, Q \setminus F)$ where $\overline{\delta}(q, \mathrm{a}) \triangleq \overline{\delta(q, \mathrm{a})}$ for all $q \in Q$, $\mathrm{a} \in \Sigma$
  - by induction on $|w|$ it can now be shown that (ii) $\hat{\overline{\delta}}(q, w) = \overline{\hat{\delta}(q, w)}$

# Complementation

- recall: NFA-complementation may blow-up automata sizes by an exponential

Lemma

*For every AFA $\mathcal{A}$ there exists an AFA $\overline{\mathcal{A}}$ of equal size such that* $\mathrm{L}(\overline{\mathcal{A}}) = \overline{\mathrm{L}(\mathcal{A})}$

Proof Outline.

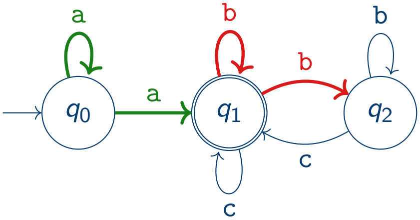
- let $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$
- define the dual formula $\overline{\phi}$ of $\phi \in \mathbb{B}^+(Q)$ following De Morgans rules

$$\overline{q} \triangleq q \qquad \overline{\phi \vee \psi} \triangleq \overline{\phi} \wedge \overline{\psi} \qquad \overline{\phi \wedge \psi} \triangleq \overline{\phi} \vee \overline{\psi}$$

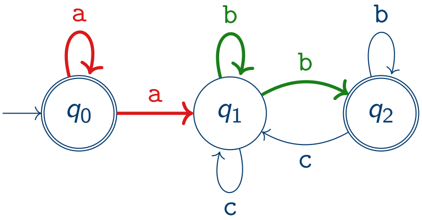
  - morally, $q \in Q$ re-used for their "negation"; we have (i) $M \vDash \phi$ iff $Q \backslash M \nvDash \overline{\phi}$
- we now define $\overline{\mathcal{A}} \triangleq (Q, \Sigma, \overline{\delta}, q_I, Q \backslash F)$ where $\overline{\delta}(q, \mathrm{a}) \triangleq \overline{\delta(q, \mathrm{a})}$ for all $q \in Q$, $\mathrm{a} \in \Sigma$
  - by induction on $|w|$ it can now be shown that (ii) $\hat{\overline{\delta}}(q, w) = \overline{\hat{\delta}(q, w)}$
  - overall, we have

$$w \notin \mathrm{L}(\mathcal{A}) \overset{def.}{\iff} F \nvDash \hat{\delta}(q_I, w) \overset{(i)}{\iff} Q \backslash F \vDash \overline{\hat{\delta}(q_I, w)} \overset{(ii)}{\iff} Q \backslash F \vDash \hat{\overline{\delta}}(q_I, w) \overset{def.}{\iff} w \in \mathrm{L}(\overline{\mathcal{A}})$$

# Example




$\Updownarrow$ complement



# Relationship with Regular Languages

# AFAs Recognize *REG*

Theorem

*For every AFA* $\mathcal{A}$ *there exist a DFA* $\mathcal{B}$ *with* $\mathrm{O}(2^{2^{|\mathcal{A}|}})$ *states such that* $\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{B})$.

# AFAs Recognize *REG*

Theorem

*For every AFA $\mathcal{A}$ there exist a DFA $\mathcal{B}$ with $O(2^{2^{|\mathcal{A}|}})$ states such that $L(\mathcal{A}) = L(\mathcal{B})$.*

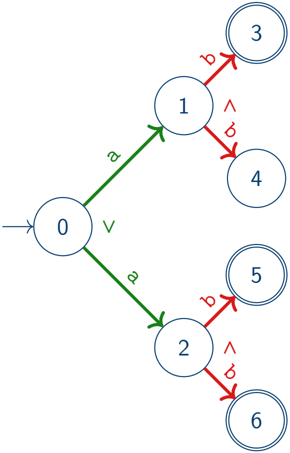
Proof Outline.

let $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$

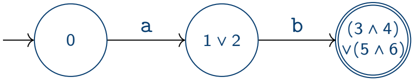
Idea:

- the states of $\mathcal{B}$ are formulas
- $\phi \xrightarrow{a} \psi$ in $\mathcal{B}$ if $\hat{\delta}(\phi, a) = \psi$
  - Example: $\delta(p, a) = q \wedge r$ and $\delta(q, a) = r \implies p \vee q \xrightarrow{a} (q \wedge r) \vee r$
  - a run $q_I \xrightarrow{a_1} \cdots \xrightarrow{a_n} \phi$ thus models $\hat{\delta}(q_I, a_1 \ldots a_n) = \phi$
- the formula $q_I$ is the initial state
- the formulas modeled by $F$ are final

# Example




the initial AFA




the translated DFA

# AFAs Recognize *REG*

Theorem

*For every AFA $\mathcal{A}$ there exist a DFA $\mathcal{B}$ with $O(2^{2^{|\mathcal{A}|}})$ states such that $L(\mathcal{A}) = L(\mathcal{B})$.*

Proof Outline.

let $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$

Idea:

- the states of $\mathcal{B}$ are formulas
- $\phi \xrightarrow{a} \psi$ in $\mathcal{B}$ if $\hat{\delta}(\phi, a) = \psi$
  - Example: $\delta(p, a) = q \wedge r$ and $\delta(q, a) = r \implies p \vee q \xrightarrow{a} (q \wedge r) \vee r$
  - a run $q_I \xrightarrow{a_1} \cdots \xrightarrow{a_n} \phi$ thus models $\hat{\delta}(q_I, a_1 \ldots a_n) = \phi$
- the formula $q_I$ is the initial state
- the formulas modeled by $F$ are final
- to keep the construction finite, we'll identify equivalent formulas

# AFAs Recognize *REG*

Theorem

*For every AFA $\mathcal{A}$ there exist a DFA $\mathcal{B}$ with $O(2^{2^{|\mathcal{A}|}})$ states such that $L(\mathcal{A}) = L(\mathcal{B})$.*

Proof Outline.

let $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$

Formally:

- the equivalence $\sim$ on $\mathbb{B}^+(Q)$ is given by $\phi \sim \psi$ if $\{M \mid M \models \phi\} = \{M \mid M \models \psi\}$
  - $q \sim q \vee q \sim q \wedge q$ but $q \not\sim p \vee q \not\sim p \wedge q$

# AFAs Recognize *REG*

Theorem

*For every AFA $\mathcal{A}$ there exist a DFA $\mathcal{B}$ with $O(2^{2^{|\mathcal{A}|}})$ states such that $L(\mathcal{A}) = L(\mathcal{B})$.*

Proof Outline.

let $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$

Formally:

- the equivalence $\sim$ on $\mathbb{B}^+(Q)$ is given by $\phi \sim \psi$ if $\{M \mid M \models \phi\} = \{M \mid M \models \psi\}$
  - $q \sim q \vee q \sim q \wedge q$ but $q \not\sim p \vee q \not\sim p \wedge q$
- the equivalence class $[\phi]_\sim$ can be simply conceived as the formula $\phi$, with equivalent formulas $\phi \sim \psi$ identified
  - $[q \vee q]_\sim = \{q, q \vee q, q \wedge q, \ldots\}$

# AFAs Recognize *REG*

Theorem

*For every AFA $\mathcal{A}$ there exist a DFA $\mathcal{B}$ with $\mathrm{O}(2^{2^{|\mathcal{A}|}})$ states such that $\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{B})$.*

Proof Outline.

let $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$

Formally:

- the equivalence $\sim$ on $\mathbb{B}^+(Q)$ is given by $\phi \sim \psi$ if $\{M \mid M \models \phi\} = \{M \mid M \models \psi\}$
  - $q \sim q \vee q \sim q \wedge q$ but $q \not\sim p \vee q \not\sim p \wedge q$
- the equivalence class $[\phi]_\sim$ can be simply conceived as the formula $\phi$, with equivalent formulas $\phi \sim \psi$ identified
  - $[q \vee q]_\sim = \{q, q \vee q, q \wedge q, \dots\}$
- the set of all such equivalence classes $\mathbb{B}^+(Q)/\sim$ contains $\mathrm{O}(2^{2^{|Q|}})$ elements

# AFAs Recognize *REG*

Theorem

*For every AFA* $\mathcal{A}$ *there exist a DFA* $\mathcal{B}$ *with* $\mathrm{O}(2^{2^{|\mathcal{A}|}})$ *states such that* $\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{B})$.

Proof Outline.

let $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$

Formally:

- ⋆ the equivalence $\sim$ on $\mathbb{B}^+(Q)$ is given by $\phi \sim \psi$ if $\{M \mid M \vDash \phi\} = \{M \mid M \vDash \psi\}$
  - $q \sim q \vee q \sim q \wedge q$ but $q \not\sim p \vee q \not\sim p \wedge q$
- ⋆ the equivalence class $[\phi]_\sim$ can be simply conceived as the formula $\phi$, with equivalent formulas $\phi \sim \psi$ identified
  - $[q \vee q]_\sim = \{q, q \vee q, q \wedge q, \ldots\}$
- ⋆ the set of all such equivalence classes $\mathbb{B}^+(Q)/\sim$ contains $\mathrm{O}(2^{2^{|Q|}})$ elements
- ⋆ $\mathcal{B} \triangleq (\mathbb{B}^+(Q)/\sim, \Sigma, q_I, \delta_\sim, \{[\phi]_\sim \mid F \vDash \phi\})$ where $\delta_\sim([\phi]_\sim, \mathtt{a}) \triangleq [\hat{\delta}(\phi, \mathtt{a})]_\sim$ recognises $\mathrm{L}(\mathcal{A})$

# From AFAs to NFA

Theorem

*For every AFA* $\mathcal{A}$ *there exist a NFA* $\mathcal{B}$ *with* $O(2^{|\mathcal{A}|})$ *states such that* $L(\mathcal{A}) = L(\mathcal{B})$.

Proof Outline.

- let $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$
- idea: models executions, states of the NFA are the levels of the execution tree
  - the construction is simpler, at the expense of non-determinism

## From AFAs to NFA

**Theorem**

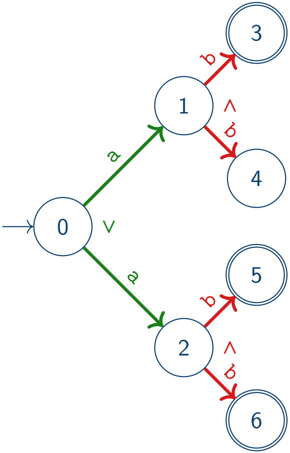
*For every AFA $\mathcal{A}$ there exist a NFA $\mathcal{B}$ with $\mathrm{O}(2^{|\mathcal{A}|})$ states such that $\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{B})$.*

**Proof Outline.**

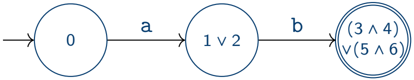
- let $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$
- idea: models executions, states of the NFA are the levels of the execution tree
  - the construction is simpler, at the expense of non-determinism
- the NFA is given by $\mathcal{B} \triangleq (2^Q, \Sigma, \{q_I\}, \delta', \{M \mid M \subseteq F\})$ where

$$N \in \delta'(M, \mathrm{a}) \quad :\Longleftrightarrow \quad N \models \bigwedge_{q \in M} \delta(q, \mathrm{a})$$

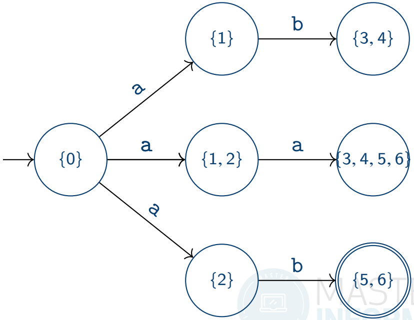
# Example (II)




the initial AFA




the translated DFA




the translated NFA