ω-Automata
ω-Automata

• Automata that accept (or reject) words of infinite length.

• Languages of infinite words appear:
  – in verification, as encodings of non-terminating executions of a program.
  – in arithmetic, as encodings of sets of real numbers.
ω-Languages

- An ω-word is an infinite sequence of letters.
- The set of all ω-words is denoted by $\Sigma^\omega$.
- An ω-language is a subset of $\Sigma^\omega$.
- A language $L_1$ can be concatenated with an ω-language $L_2$ to yield the ω-language $L_1L_2$, but two ω-languages cannot be concatenated.
- The ω-iteration of a language $L \subseteq \Sigma^*$, denoted by $L^\omega$, is an ω-language.
- Observe:
  - $\{ab\}^*$ contains infinitely many words, $\{ab\}^\omega$ contains only one
  - $\emptyset^\omega = \epsilon^\omega = \emptyset$
ω-Regular Expressions

• ω-regular expressions have syntax

\[ s ::= r^ω | rs_1 | s_1 + s_2 \]

where \( r \) is an (ordinary) regular expression.

• The ω-language \( L_ω(s) \) of an ω-regular expression \( s \) is inductively defined by

\[
L_ω(r^ω) = (L(r))^ω \\
L_ω(rs_1) = L(r)L_ω(s_1) \\
L_ω(s_1 + s_2) = L_ω(s_1) ∪ L_ω(s_2)
\]

• An ω-language is ω-regular if it is the language of some ω-regular expression.
The Quest for a Trinity

Regular languages

DFA

NFA

RE
The Quest for a Trinity

ω-RE

ω-Regular languages

N??

D??
The Rules of the Quest

• Automata should still have states, transitions, and initial states, only the acceptance condition can change.
The Rules of the Quest

• Automata should still have states, transitions, and initial states, only the acceptance condition can change.
• For automata on finite words the acceptance condition depends only on the last state of a run (i.e., runs that end in the same state are all accepting or rejecting).
The Rules of the Quest

- Automata should still have states, transitions, and initial states, only the acceptance condition can change.
- For automata on finite words the acceptance condition depends only on the last state of a run (i.e., runs that end in the same state are all accepting or rejecting).
- For automata on infinite words we choose: the acceptance condition depends only on the set of states visited infinitely often by a run (i.e., runs that visit the same states infinitely often are all accepting or rejecting).
Basic notions: Semi-automata

- A **semi-automaton** is a tuple $S = (Q, \Sigma, \delta, Q_0)$ of states, alphabet, transitions, and initial states.
Basic notions: Runs

• A run of a semi-automaton is an infinite sequence of states and transitions starting at an initial state.

\[ \rho_1 = q_0 \rightarrow q_1 \rightarrow q_1 \rightarrow q_1 \rightarrow q_1 \rightarrow q_1 \cdots \]
\[ \rho_2 = q_0 \rightarrow q_0 \rightarrow q_0 \rightarrow q_0 \rightarrow q_0 \rightarrow q_0 \cdots \]
\[ \rho_3 = q_0 \rightarrow q_1 \rightarrow q_0 \rightarrow q_0 \rightarrow q_0 \rightarrow q_1 \cdots \]
Basic notions: Runs

• The set of states visited infinitely often by a run \( \rho \) is denoted \( \text{inf}(\rho) \)

\[
\begin{align*}
\rho_1 &= q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_1 \xrightarrow{a} q_1 \xrightarrow{a} q_1 \cdots & \text{inf}(\rho_1) &= \{q_1\} \\
\rho_2 &= q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0 \cdots & \text{inf}(\rho_2) &= \{q_0\} \\
\rho_3 &= q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \cdots & \text{inf}(\rho_3) &= \{q_0, q_1\}
\end{align*}
\]
Basic notions: Acceptance conditions

- An acceptance condition is a mapping $\alpha: 2^Q \rightarrow \{0,1\}$ that determines for every set $Q' \subseteq Q$ of states whether the runs $\rho$ with $\inf(\rho) = Q'$ are accepting or not.

- $\alpha_1$: $\{q_0\} \mapsto 0$, $\{q_1\} \mapsto 0$, $\{q_0, q_1\} \mapsto 0$
- $\alpha_2$: $\{q_0\} \mapsto 0$, $\{q_1\} \mapsto 1$, $\{q_0, q_1\} \mapsto 1$
Basic notions: $\omega$-Automata

- An $\omega$-automaton is a pair $A = (S, \alpha)$, where $S$ is a semi-automaton and $\alpha$ is an acceptance condition.
Basic notions: $\omega$-Language

- An $\omega$-automaton $A$ accepts an $\omega$-word if it has at least one accepting run on it. The $\omega$-language $L_\omega(A)$ of $A$ is the set of $\omega$-words it accepts.

- $\alpha_1$: $\{q_0\} \mapsto 0, \{q_1\} \mapsto 0, \{q_0, q_1\} \mapsto 0 \quad \emptyset$

- $\alpha_2$: $\{q_0\} \mapsto 0, \{q_1\} \mapsto 1, \{q_0, q_1\} \mapsto 1 \quad ?$
Basic notions: \(\omega\)-Language

- An \(\omega\)-automaton \(A\) accepts an \(\omega\)-word if it has at least one accepting run on it. The \(\omega\)-language \(L_\omega(A)\) of \(A\) is the set of \(\omega\)-words it accepts.

- \(\alpha_1\): \(\{q_0\} \mapsto 0, \{q_1\} \mapsto 0, \{q_0, q_1\} \mapsto 0\) \(\emptyset\)

- \(\alpha_2\): \(\{q_0\} \mapsto 0, \{q_1\} \mapsto 1, \{q_0, q_1\} \mapsto 1\) infinitely many \(a\)
Basic notions: $\omega$-Language

• An $\omega$-automaton $A$ accepts an $\omega$-word if it has at least one accepting run on it. The $\omega$-language $L_\omega(A)$ of $A$ is the set of $\omega$-words it accepts.

• $\alpha_1$: $\{q_0\} \mapsto 0, \{q_1\} \mapsto 0, \{q_0, q_1\} \mapsto 0$  
  $\emptyset$

• $\alpha_2$: $\{q_0\} \mapsto 0, \{q_1\} \mapsto 1, \{q_0, q_1\} \mapsto 1$  
  $(b^*a)^\omega$
Types of $\omega$-automata

- There are many different types of acceptance conditions (Büchi, co-Büchi, Rabin, Streett, parity, Muller, generalized Büchi, Emerson-Lei ...)

  They lead to different types of $\omega$-automata: Büchi automata, co-Büchi automata, etc.

- A type is defined by stating a property that an acceptance condition may or may not satisfy. The type is the subset of all possible acceptance conditions that satisfy the property.

- This set of slides explains why this variety is needed.
Büchi automata

• Invented by J.R. Büchi, swiss logician.
**Büchi automata**

- An acceptance condition $\alpha : 2^Q \to \{0,1\}$ is a **Büchi condition** if there is a set $F \subseteq Q$ of accepting states such that $\alpha(Q') = 1$ iff $Q' \cap F \neq \emptyset$.

  - $\{q_0\} \mapsto 0, \{q_1\} \mapsto 1, \{q_0, q_1\} \mapsto 1$ is Büchi $F = \{q_1\}$
  - $\{q_0\} \mapsto 0, \{q_1\} \mapsto 1, \{q_0, q_1\} \mapsto 0$ is not Büchi

- By definition, a run $\rho$ is accepting iff $\inf(\rho) \cap F \neq \emptyset$ iff (in words) $\rho$ visits $F$ infinitely often.

- A Büchi condition $\alpha$ is completely determined by $F$. We write $A = (S,F) = (Q, \Sigma, \delta, Q_0, F)$. 
Some Büchi automata
From $\omega$-regular expressions to NBAs

NFA for $r$

NBA for $r^\omega$
From $\omega$-regular expressions to NBAs
From $\omega$-regular expressions to NBAs
From NBAs to $\omega$-regular expressions

- **Lemma**: Let $A$ be a NFA, and let $q, q'$ be states of $A$. The language $L_q^{q'}$ of words with runs leading from $q$ to $q'$ and visiting $q'$ exactly once after leaving $q$ is regular.

- Let $r_q^{q'}$ denote a regular expression for $L_q^{q'}$. 
From NBAs to \( \omega \)-regular expressions

• Example:

\[
\begin{align*}
    r_0^1 &= (a + b + c)^*(b + c) \\
    r_0^2 &= (a + b + c)^*b \\
    r_1^1 &= (b + c) \\
    r_2^2 &= b + (a + c)(a + b + c)^*b
\end{align*}
\]
From NBAs to $\omega$-regular expressions

• Given a NBA $A$, we look at it as a NFA, and compute regular expressions $r_q^{q'}$.

• We show:

$$L_\omega(A) = L \left( \sum_{q \in F} r_q^{q_0} (r_q^q \omega) \right)$$

– An $\omega$-word belongs to $L_\omega(A)$ iff it is accepted by a run that starts at $q_0$ and visits some accepting state $q$ infinitely often.
From NBAs to $\omega$-regular expressions

- Example:

$$L_\omega(A) = r_0^1 (r_1^1)^\omega + r_0^2 (r_2^2)^\omega$$

$$r_0^1 = (a + b + c)^*(b + c)$$
$$r_0^2 = (a + b + c)^*b$$
$$r_1^1 = (b + c)$$
$$r_2^2 = b + (a + c)(a + b + c)^*b$$
DBAs are less expressive than NBAs

• **Prop.:** The $\omega$-language $(a + b)^* b^\omega$ of words containing finitely many $a$ is not recognized by any DBA.
DBAs are less expressive than NBAs

• **Prop.**: The $\omega$-language $(a + b)^* b^\omega$ of words containing finitely many $a$ is not recognized by any DBA.

• **Proof**: By contradiction. Assume some DBA recognizes $(a + b)^* b^\omega$. 

  – DBA accepts $a \, b \rightarrow$ DFA accepts $(a + b)(a + b)(a + b) \ldots$
  
  – By determinism and finite number of states, the DBA accepts $(a + b)(a + b)(a + b) \ldots (a + b)(a + b) \ldots$ which does not belong to $(a + b)^* b^\omega$. 
DBAs are less expressive than NBAs

- **Prop.**: The $\omega$-language $(a + b)^*b^\omega$ of words containing finitely many $a$ is not recognized by any DBA.
- **Proof**: By contradiction. Assume some DBA recognizes $(a + b)^*b^\omega$.
  - DBA accepts $b^\omega$ $\rightarrow$ DFA accepts $b^{n_0}$
DBAs are less expressive than NBAs

• **Prop.:** The $\omega$-language $(a + b)^* b^\omega$ of words containing finitely many $a$ is not recognized by any DBA.

• **Proof:** By contradiction. Assume some DBA recognizes $(a + b)^* b^\omega$.

  – DBA accepts $b^\omega$  \rightarrow DFA accepts $b^{n_0}$
  – DBA accepts $b^{n_0}a b^\omega$  \rightarrow DFA accepts $b^{n_0}a b^{n_1}$
DBAs are less expressive than NBAs

- **Prop.:** The $\omega$-language $(a + b)^* b^\omega$ of words containing finitely many $a$ is not recognized by any DBA.
- **Proof:** By contradiction. Assume some DBA recognizes $(a + b)^* b^\omega$.
  
  - DBA accepts $b^\omega$ $\rightarrow$ DFA accepts $b^{n_0}$
  - DBA accepts $b^{n_0} a b^\omega$ $\rightarrow$ DFA accepts $b^{n_0} a b^{n_1}$
  - DBA accepts $b^{n_0} a b^{n_1} a b^\omega$ $\rightarrow$ DFA accepts $b^{n_0} a b^{n_1} a b^{n_2}$ etc.
DBAs are less expressive than NBAs

- **Prop.**: The $\omega$-language $\omega\omega$ of words containing finitely many $a$ is not recognized by any DBA.

- **Proof**: By contradiction. Assume some DBA recognizes $(a + b)^* b^\omega$.
  - DBA accepts $b^\omega$ \implies DFA accepts $b^{n_0}$
  - DBA accepts $b^{n_0} a b^\omega$ \implies DFA accepts $b^{n_0} a b^{n_1}$
  - DBA accepts $b^{n_0} a b^{n_1} a b^\omega$ \implies DFA accepts $b^{n_0} a b^{n_1} a b^{n_2}$ etc.
  - By determinism and finite number of states, the DBA accepts $b^{n_0} a b^{n_1} a b^{n_2} ... a b^{n_i} (a b^{n_{i+1}} ... a b^{n_j})^\omega$ for some $i < j$. This word does not belong to $(a + b)^* b^\omega$. 
Büchi automata do not form a Trinity
Co-Büchi automata

- An accepting condition \( \alpha: 2^Q \rightarrow \{0,1\} \) is a co-Büchi condition if there is a set \( F \) of accepting states such that \( \alpha(Q') = 1 \) iff \( Q' \cap F = \emptyset \).
  
  \[ \begin{align*}
  \{q_0\} &\mapsto 0, \{q_1\} \mapsto 1, \{q_0, q_1\} \mapsto 0 \quad \text{is co-Büchi} \\
  \{q_0\} &\mapsto 0, \{q_1\} \mapsto 1, \{q_0, q_1\} \mapsto 1 \quad \text{is not co-Büchi}
  \end{align*} \]

- Equivalently: \( \rho \) is accepting iff \( \inf(\rho) \cap F = \emptyset \)
  iff (in words) \( \rho \) visits \( F \) finitely often.

- A co-Büchi condition \( \alpha \) is completely determined by \( F \).
  We write \( A = (S,F) = (Q,\Sigma,\delta,Q_0,F) \) (danger!)
Co-Büchi automata

• Let $A$ be a Büchi automaton, let $B$ be the same co-Büchi automaton (with the same set $F$), and let $\rho$ be a run:
  – $\rho$ is accepting in $A$ if it visits $F$ infinitely often
  – $\rho$ is accepting in $B$ if it visits $F$ finitely often

• So an accepting run of $A$ is a rejecting run of $B$ and vice versa.

• Therefore: If $A$ is a DBA recognizing an $\omega$-language $L$, then $B$ is a DCA recognizing $\overline{L}$.

• Not necessarily true for NBA!
Which are the languages?
Determinizing co-Büchi automata

• Given a NCA $A$ we construct a DCA $B$ such that $L(A) = L(B)$.

• We proceed in three steps:
  – We assign to every $\omega$-word $w$ a directed acyclic graph $\text{dag}(w)$ that ``contains´´ all runs of $A$ on $w$.
  – We prove that $w$ is accepted by $A$ iff $\text{dag}(w)$ is infinite but contains only finitely many breakpoints.
  – We construct a DCA $B$ such that $w$ is accepted by $B$ iff $\text{dag}(w)$ is infinite but contains only finitely many breakpoints.
• Running example:
\( \text{dag}(aba^\omega) \)

\[q, 0 \xrightarrow{a} q, 1 \xrightarrow{a} r, 1 \]

\[q, 2 \xrightarrow{a} q, 3 \xrightarrow{a} q, 4 \]

\[\ldots\]

\[\text{dag}((ab)^\omega) \]

\[q, 0 \xrightarrow{a} q, 1 \xrightarrow{a} r, 1 \]

\[q, 2 \xrightarrow{a} q, 3 \xrightarrow{a} r, 3 \]

\[\ldots\]
• $A$ accepts $w$ iff some infinite path of $\text{dag}(w)$ only visits accepting states finitely often.
Levels of a **dag**

Level 0  
Level 1  
Level 2  
Level 3  
Level 4
Breakpoints of a *dag*

- We define inductively the set of levels that are breakpoints:
  - Level 0 is always a breakpoint.
  - If level $l$ is a breakpoint, then the next level $l'$ such that every path from $l$ to $l'$ visits an accepting state at some level between $l+1$ and $l'$ is also a breakpoint.
Only two breakpoints

Infinitely many breakpoints
Lemma: $A$ accepts $w$ iff $\text{dag}(w)$ is infinite and has only finitely many breakpoints.

Proof:

$(\Rightarrow)$ If $A$ accepts $w$, then it has at least one run on $w$, and so $\text{dag}(w)$ is infinite.

Moreover, the run visits accepting states only finitely often, and so after it stops visiting accepting states there are no further breakpoints.
Lemma: $A$ accepts $w$ iff $\text{dag}(w)$ is infinite and has only finitely many breakpoints.

Proof:

$(\Leftarrow)$ Assume $\text{dag}(w)$ is infinite and has only finitely many breakpoints. Let $l$ be the last breakpoint.

Since $\text{dag}(w)$ is infinite, for every $l' > l$ there is a path from $l$ to $l'$ that visits no accepting states.

The subdag containing all these paths is infinite and has finite degree.

By König‘s Lemma the dag contains an infinite path.
If we could tell if a level is a breakpoint by looking at it and to no other level, then we could take the set of all levels/breakpoints as the set of states/accepting states of the DCA.
However, in order to decide if a level is a breakpoint we need information about its `history`.
Solution: add that information to the level.
• States: pairs \([P, O]\) where:
  – \(P\) is the set of states of a level, and
  – \(O \subseteq P\) is the set of states “that owe a visit to the set of accepting states“.

• Formally: \(q \in O\) if \(q\) is the endpoint of a path starting at the last breakpoint that has not yet visited any accepting state.
Constructing the DCA

- **States**: pairs \([P, O]\)
- **Initial state**: pair \([Q_0, \emptyset]\).
- **Transitions**: \(\delta([P, O], a) = [P', O']\) where \(P' = \delta(P, a)\) and \(O'\) is given by:
  - \(O' = \delta(O, a) \setminus F\) if \(O \neq \emptyset\) (automaton updates set of owing states)
  - \(O' = \delta(P, a) \setminus F\) if \(O = \emptyset\) (automaton starts search for next breakpoint)
- **Accepting states**: pairs \([P, \emptyset]\) (no owing states)
\textbf{NCAtoDCA}(A) \\
\textbf{Input:} NCA \( A = (Q, \Sigma, \delta, Q_0, F) \) \\
\textbf{Output:} DCA \( B = (\tilde{Q}, \Sigma, \tilde{\delta}, \tilde{q}_0, \tilde{F}) \) with \( L_\omega(A) = L_\omega(B) \) \\
1 \( \tilde{Q}, \tilde{\delta}, \tilde{F} \leftarrow \emptyset; \tilde{q}_0 \leftarrow [Q_0, \emptyset] \) \\
2 \( W \leftarrow \{ \tilde{q}_0 \} \) \\
3 \textbf{while } W \neq \emptyset \textbf{ do} \\
4 \quad \textbf{pick } [P, O] \textbf{ from } W; \textbf{ add } [P, O] \textbf{ to } \tilde{Q} \\
5 \quad \textbf{if } O = \emptyset \textbf{ then add } [P, O] \textbf{ to } \tilde{F} \\
6 \quad \textbf{for all } a \in \Sigma \textbf{ do} \\
7 \quad \quad P' = \delta(P, a) \\
8 \quad \quad \textbf{if } O \neq \emptyset \textbf{ then } O' \leftarrow \delta(O, a) \setminus F \textbf{ else } O' \leftarrow \delta(P, a) \setminus F \\
9 \quad \quad \textbf{add } ([P, O], a, [P', O']) \textbf{ to } \tilde{\delta} \\
10 \quad \quad \textbf{if } [P', O'] \notin \tilde{Q} \textbf{ then add } [P', Q'] \textbf{ to } W \\
\\n- \textbf{Complexity: at most } 3^n \textbf{ states}
Running example
Co-Büchi Automata do not form a Trinity

**Lemma:** No DCA (and so no NCA) recognizes the $\omega$-language $(b^*a)^\omega$.

**Proof:** Assume the contrary. Then the same automaton seen as a DBA recognizes the complement $\omega$-language $(a + b)^*b^\omega$. Contradiction.

It can be proven that all $\omega$-languages accepted by NCA are $\omega$-regular (exercise!). So NCA are strictly less expressive than NBA.
Co-Büchi Automata do not form a Trinity 

ω-RE

ω-Regular languages

NCA

DCA
Generalizing NBAs

- Recall: No DBA for \((a + b)^* b^\omega\)
- Can be „repaired“ by combining Büchi and co-Büchi conditions:

Runs that visit \(q\) finitely often and moreover visit \(r\) infinitely often recognize \((a + b)^* b^\omega\)
• A Rabin pair is a pair $\langle F, G \rangle$ of sets of states.

• An accepting condition $\alpha: 2^Q \rightarrow \{0,1\}$ is a Rabin condition if there is a set $\mathcal{R}$ of Rabin pairs such that

$$\alpha(Q') = 1 \quad \text{iff} \quad Q' \cap F \neq \emptyset \quad \text{and} \quad Q' \cap G = \emptyset$$

for some pair $\langle F, G \rangle \in \mathcal{R}$.

$\rho$ is accepting

iff $\inf(\rho) \cap F \neq \emptyset$ and $\inf(\rho) \cap G = \emptyset$ for some $\langle F, G \rangle \in \mathcal{R}$

iff (in words) $\rho$ visits $F$ infinitely often and $G$ finitely often

for some $\langle F, G \rangle \in \mathcal{R}$. 
The accepting condition

\[ \{q_0\} \mapsto 1, \{q_1\} \mapsto 1, \{q_0, q_1\} \mapsto 0 \]

is neither Büchi nor co-Büchi, but it is the Rabin condition

\[ \{ \langle q_0, q_1 \rangle, \langle q_1, q_0 \rangle \} \]

(two Rabin pairs)

• Büchi condition \( F \equiv \) Rabin condition \( \{ \langle F, \emptyset \rangle \} \)

• Co-Büchi condition \( G \equiv \) Rabin condition \( \{ \langle Q, G \rangle \} \)

• Theorem (Safra): Any NRA with \( n \) states can be effectively transformed into a DRA with \( n^{O(n)} \) states.
From Rabin to Büchi automata

• Let $A$ be a NRA with condition $\{\langle F_1, G_1 \rangle, \ldots, \langle F_m, G_m \rangle \}$.

• Let $A_1, \ldots, A_m$ be NRAs with the same semi-automaton as $A$ but Rabin conditions $\{\langle F_1, G_1 \rangle \}$, $\ldots$, $\{\langle F_m, G_m \rangle \}$ respectively.

• We have: $L(A) = L(A_1) \cup \ldots \cup L(A_m)$

• We proceed in two steps:

  1. we construct for each NRA $A_i$ an NBA $A'_i$ such that $L(A_i) = L(A'_i)$

  2. we (easily) construct an NBA $A'$ such that $L(A') = L(A'_1) \cup \ldots \cup L(A'_m)$
Transitions leaving $Q \setminus G_i$ are duplicated and redirected to the copy of $Q \setminus G_i$.

NBA with accepting condition

$$(Q \setminus G_i') \cap F_i'$$
Beyond Trinities

- Can we find a class $X$ of $\omega$-automata such that
- $\mathsf{RE}$, $\mathsf{NXA}$, $\mathsf{DXA}$ form a Trinity, and
- Boolean operations for $\mathsf{DXAs}$ can be implemented „as for $\mathsf{DFAs}$“?

1) For every $\mathsf{DXA}$ $A = (S, \alpha)$ there is a $\mathsf{DXA}$ $\overline{A} = (S, \overline{\alpha})$ recognizing $\overline{L_\omega(A)}$

2) For every two $\mathsf{DXAs}$ $A_1 = (S_1, \alpha_1)$ and $A_2 = (S_2, \alpha_2)$ there is a $\mathsf{DXA}$ $A_\cup = ([S_1, S_2], \alpha_\cup)$ recognizing $L_\omega(A_1) \cup L_\omega(A_2)$

3) For every two $\mathsf{DXAs}$ $A_1 = (S_1, \alpha_1)$ and $A_2 = (S_2, \alpha_1)$ there is a $\mathsf{DXA}$ $A_\cap = ([S_1, S_2], \alpha_\cap)$ recognizing $L_\omega(A_1) \cap L_\omega(A_2)$
Beyond Trinities


- Given two DRAs $A_1 = (S_1, \alpha_1)$ and $A_2 = (S_2, \alpha_2)$, the DRA $A_\cup = ([S_1, S_2], \alpha)$ where

  $$\alpha = \left\{ \langle F_1 \times Q_2, G_1 \times Q_2 \rangle : \langle F_1, G_1 \rangle \in \alpha_1 \right\} \cup \left\{ \langle Q_1 \times F_2, Q_1 \times G_2 \rangle : \langle F_2, G_2 \rangle \in \alpha_2 \right\}$$

  recognizes $L_\omega(A_1) \cup L_\omega(A_2)$
Beyond Trinities

• Two further Trinities (see notes):

• A final Trinity:
  – Muller automata: 1): Yes. 2): Yes. 3): Yes.
Muller automata

• Automata with arbitrary acceptance conditions.

• A Muller automaton (NMA) is an automaton $A = (S, \alpha)$ where $\alpha: 2^Q \rightarrow \{0,1\}$ is an arbitrary acceptance condition.

• We represent $\alpha$ by the set $\mathcal{F}$ of all sets of states $Q' \subseteq Q$ such that $\alpha(Q') = 1$.

• A run is accepting if the set of states it visits infinitely often is equal to one of the sets in $\mathcal{F}$.

• Theorem: RE, NMA, and DMA form a Trinity.
Muller automata

• Infinitely many $a$

\{ \{q_a\}, \{q_a, q_b\}, \{q_a, q_c\}, \{q_a, q_b, q_c\} \}
Muller automata

- Infinitely many $a$
  $$\{ \{q_a\},\{q_a,q_b\},\{q_a,q_c\},\{q_a,q_b,q_c\} \}$$
- Infinitely many $a$ or infinitely many $b$
Muller automata

- Infinitely many $a$
  \[ \{ \{q_a\}, \{q_a, q_b\}, \{q_a, q_c\}, \{q_a, q_b, q_c\} \} \]

- Infinitely many $a$ or infinitely many $b$
  \[ \{ \{q_a\}, \{q_b\}, \{q_a, q_b\}, \{q_a, q_c\}, \{q_a, q_c\}, \{q_a, q_b, q_c\} \} \]
Muller automata

- Infinitely many $a$
  \[
  \{ \{q_a\}, \{q_a, q_b\}, \{q_a, q_c\}, \{q_a, q_b, q_c\} \}
  \]

- Infinitely many $a$ or infinitely many $b$
  \[
  \{ \{q_a\}, \{q_b\}, \{q_a, q_b\}, \{q_a, q_c\}, \{q_a, q_c\}, \{q_a, q_b, q_c\} \}
  \]

- Infinitely many $a$ and infinitely many $b$
Muller automata

- Infinitely many $a$
  \[
  \{ \{q_a\}, \{q_a, q_b\}, \{q_a, q_c\}, \{q_a, q_b, q_c\} \}
  \]

- Infinitely many $a$ or infinitely many $b$
  \[
  \{ \{q_a\}, \{q_b\}, \{q_a, q_b\}, \{q_a, q_c\}, \{q_a, q_c\}, \{q_a, q_b, q_c\} \}
  \]

- Infinitely many $a$ and infinitely many $b$
  \[
  \{ \{q_a, q_b\}, \{q_a, q_b, q_c\} \}
  \]
Muller automata

- Infinitely many $a$
  \[
  \{ \{ q_a \}, \{ q_a, q_b \}, \{ q_a, q_c \}, \{ q_a, q_b, q_c \} \}
  \]

- Infinitely many $a$ or infinitely many $b$
  \[
  \{ \{ q_a \}, \{ q_b \}, \{ q_a, q_b \}, \{ q_a, q_c \}, \{ q_a, q_c \}, \{ q_a, q_b, q_c \} \}
  \]

- Infinitely many $a$ and infinitely many $b$
  \[
  \{ \{ q_a, q_b \}, \{ q_a, q_b, q_c \} \}
  \]

- Finitely many $a$ or finitely many $b$
Muller automata

- Infinitely many $a$
  \[ \{ \{q_a\} , \{q_a, q_b\}, \{q_a, q_c\}, \{q_a, q_b, q_c\} \} \]

- Infinitely many $a$ or infinitely many $b$
  \[ \{ \{q_a\} , \{q_b\}, \{q_a, q_b\}, \{q_a, q_c\}, \{q_a, q_c\}, \{q_a, q_b, q_c\} \} \]

- Infinitely many $a$ and infinitely many $b$
  \[ \{ \{q_a, q_b\}, \{q_a, q_b, q_c\} \} \]

- Finitely many $a$ or finitely many $b$
  \[ \{ \{q_a\} , \{q_b\}, \{q_c\}, \{q_a, q_c\}, \{q_b, q_c\} \} \]
Boolean operations on DMAs

Let $A = (S, \mathcal{F})$ be a DMA. The DRA $\bar{A} = (S, \bar{\mathcal{F}})$, where

$$\bar{\mathcal{F}} = \{R \subseteq Q : R \not\in \mathcal{F}\}$$

recognizes $\overline{L_\omega(A)}$. 
Boolean operations on DMAs

Let $A = (S, \mathcal{F})$ be a DMA. The DRA $\tilde{A} = (S, \tilde{\mathcal{F}})$, where

$$\tilde{\mathcal{F}} = \{ R \subseteq Q : R \notin \mathcal{F} \}$$

recognizes $L_\omega(A)$.

**Problem:** $\tilde{\mathcal{F}}$ can be exponentially larger than $\mathcal{F}$.
Boolean operations on DMAs

• Let $A_1 = (S_1, \mathcal{F}_1)$ and $A_2 = (S_2, \mathcal{F}_2)$ be DMAs.

• Given $R \subseteq Q_1 \times Q_2$, let $R|_1$ and $R|_2$ denote the projections of $R$ on $Q_1$ and $Q_2$.

• The DRAs $A_U = ([S_1, S_2], \mathcal{F}_U)$ and $A_n = ([S_1, S_2], \mathcal{F}_n)$, where

  $$\mathcal{F}_U = \{ R \subseteq Q_1 \times Q_2 : R|_1 \in \mathcal{F}_1 \text{ or } R|_2 \in \mathcal{F}_2 \}$$

  $$\mathcal{F}_n = \{ R \subseteq Q_1 \times Q_2 : R|_1 \in \mathcal{F}_1 \text{ and } R|_2 \in \mathcal{F}_2 \}$$

recognize $L_\omega(A_1) \cup L_\omega(A_2)$ and $L_\omega(A_1) \cap L_\omega(A_2)$.

• Same problem as for complementation: $\mathcal{F}_U$ and $\mathcal{F}_n$ can be exponentially larger than $\mathcal{F}$. 
## Summary

<table>
<thead>
<tr>
<th>Automaton Type</th>
<th>Expr</th>
<th>Det</th>
<th>Union</th>
<th>Inters.</th>
<th>Comp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFA/DFA</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>NBA/DBA (Büchi)</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>NCA/DCA (Co-Büchi)</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>NRA/DRA (Rabin)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>NSA/DSA (Streett)</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>NPA/DPA (Parity)</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>NMA/DMA (Muller)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

**Expr**: Is there a conversion from RE to NXA?
**Det**: Is there a conversion from NXA to DXA?
**Union**: Does pairing work for DXA and union?
**Inters**: Does pairing work for DXA and intersection?
**Comp**: Can DXA be complemented without changing the semi-automaton?

_Y_: the underlying conversion or operation has polynomial blow-up