A 5–tuple \((\Sigma, Q, Q_0, \delta, F)\), where

- \(\Sigma\) : Input alphabet
- \(Q\) : Finite set of states
- \(Q_0 \subseteq Q\) : Initial states
- \(\delta \subseteq Q \times \Sigma \times Q\) : State transition relation
- \(F\) : Set of final/accepting states
A run of $A$ on $\alpha \in \Sigma^\omega$ is a sequence $\rho : \mathbb{N} \rightarrow Q$ such that
- $\rho(0) \in Q_0$
- $\rho(i + 1) \in \delta(\rho(i), \alpha(i))$

$\alpha = a b b b b b \cdots$, $\rho_1 = q_1 q_2 q_2 q_2 q_2 q_2 q_2 \cdots$
A run of $A$ on $\alpha \in \Sigma^\omega$ is a sequence $\rho : \mathbb{N} \rightarrow Q$ such that

- $\rho(0) \in Q_0$
- $\rho(i + 1) \in \delta(\rho(i), \alpha(i))$

An automaton may have several runs on $\alpha$.

- $\alpha = abbbbbb \cdots$, $\rho_1 = q_1q_2q_2q_2q_2q_2q_2 \cdots$
- $\rho_2 = q_1q_1q_2q_2q_2q_2q_2 \cdots$
A run of $\mathcal{A}$ on $\alpha \in \Sigma^\omega$ is a sequence $\rho : \mathbb{N} \to Q$ such that
- $\rho(0) \in Q_0$
- $\rho(i + 1) \in \delta(\rho(i), \alpha(i))$

An automaton may have several runs on $\alpha$.

$\rho$ is accepting iff
$\inf(\rho) \cap F \neq \emptyset$

$\alpha$ is accepted by $\mathcal{A}$ ($\alpha \in L(\mathcal{A})$) iff there is an accepting run of $\mathcal{A}$ on $\alpha$.

- $\alpha = a b b b b b \cdots$, $\rho_1 = q_1 q_2 q_2 q_2 q_2 q_2 q_2 \cdots$
- $\rho_2 = q_1 q_1 q_2 q_2 q_2 q_2 q_2 q_2 \cdots$
Ambiguous automata

$A$ is *ambiguous* if there exists $\alpha \in L(A)$ such that there are $\geq 2$ accepting runs of $A$ on $\alpha$. Otherwise, $A$ is *unambiguous*. 

An ambiguous NBW

```
q1
q2
q3
q4
q5
qs
a
b
a, b
```

Final state

$\alpha = ab$

$\omega$
Ambiguous automata

$A$ is *ambiguous* if there exists $\alpha \in L(A)$ such that there are $\geq 2$ accepting runs of $A$ on $\alpha$. Otherwise, $A$ is *unambiguous*.

An ambiguous NBW

- $\alpha = ab^\omega$
- $\rho_1 = q_1q_2^\omega$
- $\rho_2 = q_1q_1q_2^\omega$
**Strongly Unambiguous automata**

- **Final run of** $A$ on $\alpha$: A run $\rho$ starting from any state in $Q$ such that $\text{inf}(\rho) \cap F \neq \emptyset$.
  - A word $\not\in L(A)$ may have 0 or more final runs
  - A word $\in L(A)$ has $\geq 1$ final runs
**Strongly Unambiguous automata**

- **Final run** of $A$ on $\alpha$: A run $\rho$ starting from any state in $Q$ such that $\inf(\rho) \cap F \neq \emptyset$.
  - A word $\notin L(A)$ may have 0 or more final runs
  - A word $\in L(A)$ has $\geq 1$ final runs

- NBW $A$ is **strongly unambiguous** if for every $\alpha \in \Sigma^\omega$, there is exactly one final run.

---

Hrishikesh Karmarkar  
Supratik Chakraborty  
Ranking based Techniques for Disambiguating Büchi Automata
**Strongly Unambiguous automata**

- *Final run of* \(A\) *on* \(\alpha\): A run \(\rho\) starting from *any* state in \(Q\) such that \(\inf(\rho) \cap F \neq \emptyset\).
  - A word \(\notin L(A)\) may have 0 or more final runs.
  - A word \(\in L(A)\) has \(\geq 1\) final runs.

- **NBW** \(A\) is *strongly unambiguous* if for every \(\alpha \in \Sigma^\omega\), there is exactly one final run.

- Not all unambiguous automata are strongly unambiguous.
**Strongly Unambiguous automata**

- **Final run** of $A$ on $\alpha$: A run $\rho$ starting from any state in $Q$ such that $\inf(\rho) \cap F \neq \emptyset$.
  - A word $\not\in L(A)$ may have 0 or more final runs
  - A word $\in L(A)$ has $\geq 1$ final runs

- NBW $A$ is **strongly unambiguous** if for every $\alpha \in \Sigma^\omega$, there is exactly one final run.

- Not all unambiguous automata are strongly unambiguous.

---

**Diagram:**

- Deterministic (hence unambiguous) but not strongly unambiguous
- Strongly unambiguous
Containment relations

UNBW: Unambiguous NBW, SUNBW: Strongly unambiguous NBW, DBW: Deterministic Büchi automata over words

Expressive power-wise

\[ \text{DBW} \subsetneq \text{NBW} \equiv \text{UNBW} \equiv \text{SUNBW} \]

Automata structure-wise

```
NBW
  \text{DBW}
  \text{UNBW}
  \text{SUNBW}
```
What this talk is about

Given an NBW, construct UNBW accepting the same language and using as few states as possible.

Relevant earlier work:
Arnold 1983: UNBW expressively equivalent to NBW
Carton & Michel 2003: Effective construction of SUNBW, size bound $O((12^n)^n)$
Kähler and Wilke 2008: Effective construction of UNBW, size bound $O((3^n)^n)$.
Bousquet and Löding 2010: Equivalence and inclusion problems for SUNBW are poly-time

Our contribution
Effective construction of UNBW, size bound $O((n^2(0.76n))^n)$.

Same as best known bound for NBW complementation!
What this talk is about

Given an NBW, construct UNBW accepting the same language and using as few states as possible.

Relevant earlier work:

- Arnold 1983: UNBW expressively equivalent to NBW
- Carton & Michel 2003: Effective construction of SUNBW, size bound $O((12n)^n)$
- Kähler and Wilke 2008: Effective construction of UNBW, size bound $O((3n)^n)$.
- Bousquet and Löding 2010: Equivalence and inclusion problems for SUNBW are poly-time
What this talk is about

Given an NBW, construct UNBW accepting the same language and using as few states as possible.

Relevant earlier work:

- Arnold 1983: UNBW expressively equivalent to NBW
- Carton & Michel 2003: Effective construction of SUNBW, size bound $O((12n)^n)$
- Kähler and Wilke 2008: Effective construction of UNBW, size bound $O((3n)^n)$.
- Bousquet and Löding 2010: Equivalence and inclusion problems for SUNBW are poly-time

Our contribution

- Effective construction of UNBW, size bound $O(n^2 \cdot (0.76n)^n)$
- Same as best known bound for NBW complementation!
Why care about disambiguation?

- Of course, a theoretically interesting problem
- Can it lead to a better understanding of what kinds of NBW admit easy determinization?
- Practical application? Seek inputs from the audience.
Run DAGs

Run DAG for $a^\omega$

Vertices with $qs$ not shown for clarity

Final state

Hrishikesh Karmarkar  Supratik Chakraborty  Ranking based Techniques for Disambiguating Büchi Automata
Intuitively, assign a metric to each vertex in run DAG such that the metric changes in a desirable way only along “good” runs.
Intuitively, assign a metric to each vertex in run DAG such that the metric changes in a desirable way only along “good” runs.

Intuitively, assign a metric to each vertex in run DAG such that the metric changes in a desirable way only along “good” runs.


Recent spurt of work triggered by similar metrics defined by Kupferman & Vardi (2001 onwards)
Intuitively, assign a metric to each vertex in run DAG such that the metric changes in a desirable way only along “good” runs.

- Recent spurt of work triggered by similar metrics defined by Kupferman & Vardi (2001 onwards)
  - Schewe (2009) used this approach to match upper bound of NBW complementation within $O(n^2)$ of lower bound
Intuitively, assign a metric to each vertex in run DAG such that the metric changes in a desirable way only along “good” runs.


Recent spurt of work triggered by similar metrics defined by Kupferman & Vardi (2001 onwards)

- Schewe (2009) used this approach to match upper bound of NBW complementation within $O(n^2)$ of lower bound
- We use Kupferman-Vardi style rankings
Kupferman-Vardi style ranking

\[ n: \text{Number of states in NBW} \]
\[ V: \text{Set of run DAG vertices} \]
\[ r: V \rightarrow \{1, 2, \ldots, 2n + 1\}: \text{Ranking function} \]
Kupferman-Vardi style ranking

\( n \): Number of states in NBW
\( V \): Set of run DAG vertices
\( r : V \rightarrow \{1, 2, \ldots, 2n + 1\} \): Ranking function

**Constraints on ranks**
- Vertices corresponding to final states must not get odd ranks
Kupferman-Vardi style ranking

\( n \): Number of states in NBW
\( V \): Set of run DAG vertices
\( r : V \to \{1, 2, \ldots, 2n + 1\} \): Ranking function

**Constraints on ranks**
- Vertices corresponding to final states must not get odd ranks
- Ranking cannot increase along any path in run DAG
Kupferman-Vardi style ranking

- $n$: Number of states in NBW
- $V$: Set of run DAG vertices
- $r: V \rightarrow \{1, 2, \ldots 2n + 1\}$: Ranking function

**Constraints on ranks**
- Vertices corresponding to final states must not get odd ranks
- Ranking cannot increase along any path in run DAG

- **Odd ranking**: Every path eventually trapped in an odd rank
- **Even ranking** otherwise
Example of KV-ranking:

Example KV-ranking of run DAG for $a^\omega$
Theorem (Kupferman-Vardi 2001)

An \( \omega \)-word \( \alpha \in \overline{L(A)} \) iff there is an odd ranking of the run DAG of \( A \) on \( \alpha \).

Example ranking for \( ba^\omega \)
Applications of Kupferman-Vardi’s theorem

- Series of followup work on NBW complementation using KV-ranking

Schewe (2009) finally gave a construction yielding a complement NBW of size $O(n^2 \cdot (0.76^n n))$.

Several optimizations possible on basic construction. One such set of optimizations leads to an unambiguous complementation construction, and a disambiguation construction too! Achieves same bound of $O(n^2 \cdot (0.76^n n))$.
Series of followup work on NBW complementation using KV-ranking

Schewe (2009) finally gave a construction yielding a complement NBW of size $O(n^2(0.76n)^n)$
Applications of Kupferman-Vardi’s theorem

- Series of followup work on NBW complementation using KV-ranking
- Schewe (2009) finally gave a construction yielding a complement NBW of size $O(n^2 \cdot (0.76n)^n)$
  - Lower bound $\Omega((0.76n)^n)$. 

Several optimizations possible on basic construction
One such set of optimizations leads to an unambiguous complementation construction, and a disambiguation construction too! Achieves same bound of $O(n^2 \cdot (0.76n)^n)$. 

Hrishikesh Karmarkar Supratik Chakraborty
Series of followup work on NBW complementation using KV-ranking

Schewe (2009) finally gave a construction yielding a complement NBW of size $O(n^2 (0.76n)^n)$
- Lower bound $\Omega((0.76n)^n)$.

Several optimizations possible on basic construction
Applications of Kupferman-Vardi’s theorem

- Series of follow-up work on NBW complementation using KV-ranking
- Schewe (2009) finally gave a construction yielding a complement NBW of size $O(n^2(0.76n)^n)$
  - Lower bound $\Omega((0.76n)^n)$.
- Several optimizations possible on basic construction
- One such set of optimizations leads to an unambiguous complementation construction, and a disambiguation construction too!
Applications of Kupferman-Vardi’s theorem

Series of followup work on NBW complementation using KV-ranking

Schewe (2009) finally gave a construction yielding a complement NBW of size $O(n^2(0.76n)^n)$
  - Lower bound $\Omega((0.76n)^n)$.

Several optimizations possible on basic construction

One such set of optimizations leads to an unambiguous complementation construction, and a disambiguation construction too!
  - Achieves same bound of $O(n^2, (0.76n)^n)$. 
Recall KV-ranking

$n$: Number of states in NBW

$V$: Set of run DAG vertices

$r: V \rightarrow \{1, 2, ..., 2^n + 1\} \cup \{\infty\}$: Ranking function

Constraints on ranks
- Vertices corresponding to final states must not get odd ranks
- Ranking cannot increase along any path in run DAG
- Every path eventually trapped in odd rank or in $\infty$
- We call this a full ranking of the run DAG.
Extending KV-ranks

Recall KV-ranking

$n$: Number of states in NBW
$V$: Set of run DAG vertices
$r : V \rightarrow \{1, 2, \ldots 2n + 1\} \cup \{\infty\} :$ Ranking function

Constraints on ranks

Vertices corresponding to final states must not get odd ranks
Ranking cannot increase along any path in run DAG
Every path eventually trapped in odd rank or in $\infty$
We call this a full ranking of the run DAG.
Recall KV-ranking

\( n \): Number of states in NBW

\( V \): Set of run DAG vertices

\( r : V \rightarrow \{1, 2, \ldots, 2n + 1\} \cup \{\infty\} \): Ranking function

### Constraints on ranks

- Vertices corresponding to final states must not get odd ranks
- Ranking cannot increase along any path in run DAG
Recall KV-ranking

\( n \): Number of states in NBW

\( V \): Set of run DAG vertices

\( r : V \rightarrow \{1, 2, \ldots, 2n + 1\} \cup \{\infty\} \): Ranking function

Constraints on ranks

- Vertices corresponding to final states must not get odd ranks
- Ranking cannot increase along any path in run DAG
- Every path eventually trapped in odd rank or in \( \infty \)
Recall KV-ranking

- \( n \): Number of states in NBW
- \( V \): Set of run DAG vertices
- \( r : V \rightarrow \{1, 2, \ldots 2n + 1\} \cup \{\infty\} \): Ranking function

**Constraints on ranks**

- Vertices corresponding to final states must not get odd ranks
- Ranking cannot increase along any path in run DAG
- Every path eventually trapped in odd rank or in \( \infty \)

We call this a **full ranking** of the run DAG.
Example of full ranking

Example full ranking for $a^\omega$
Given run DAG $G$, full ranking $r^*$ of $G$ is \textbf{minimal} iff for all full rankings $r$ of $G$, $r^*(v) \leq r(v)$ for all vertices $v$ in $G$. 

Non-minimal full ranking 

Minimal full ranking
Theorem

For every run DAG, there exists a unique minimal full ranking. A word $\alpha$ is accepted by $A$ iff the minimal full ranking of the run DAG assigns $\infty$ as the rank of the root vertex.

F-vertex: Vertex in run DAG for which the state is final.

Local properties (successors)
Every vertex that is not an $F$-vertex has a successor with the same rank.
Every even ranked vertex either has a successor with the same rank or one with the next lower odd rank.
Properties of minimal full rankings

Theorem

For every run DAG, there exists a unique minimal full ranking. A word $\alpha$ is accepted by $A$ iff the minimal full ranking of the run DAG assigns $\infty$ as the rank of the root vertex.

$F$-vertex: Vertex in run DAG for which the state is final.
Properties of minimal full rankings

Theorem

For every run DAG, there exists a unique minimal full ranking. A word $\alpha$ is accepted by $A$ iff the minimal full ranking of the run DAG assigns $\infty$ as the rank of the root vertex.

$F$-vertex: Vertex in run DAG for which the state is final.

Local properties (successors)
Properties of minimal full rankings

**Theorem**

*For every run DAG, there exists a unique minimal full ranking. A word \( \alpha \) is accepted by \( A \) iff the minimal full ranking of the run DAG assigns \( \infty \) as the rank of the root vertex.*

**F-vertex**: Vertex in run DAG for which the state is final.

**Local properties (successors)**

- Every vertex that is not a F-vertex has a successor with the same rank
Properties of minimal full rankings

**Theorem**

For every run DAG, there exists a unique minimal full ranking. A word $\alpha$ is accepted by $A$ iff the minimal full ranking of the run DAG assigns $\infty$ as the rank of the root vertex.

$F$-vertex: Vertex in run DAG for which the state is final.

**Local properties (successors)**

- Every vertex that is not a $F$-vertex has a successor with the same rank
- Every even ranked vertex either has a successor with the same rank or one with the next lower odd rank
Properties of minimal full rankings

Global properties (descendants)

- Every even ranked vertex has at least one descendant with the next lower odd rank.
- Every odd ranked (>1) vertex has at least one $F$-vertex descendant with the next lower even rank.
- Every path from every even ranked vertex eventually encounters a vertex with a lower rank.
- Every $\infty$ ranked vertex has at least one $\infty$ ranked $F$-vertex descendant.
- Every $\infty$ ranked vertex has at least one descendant with the largest non-infinity rank in range of the ranking function.
Global properties (descendants)

- Every even ranked vertex has at least one descendant with the next lower odd rank.
Properties of minimal full rankings

**Global properties (descendants)**

- Every even ranked vertex has at least one descendant with the next lower odd rank.
- Every odd ranked (> 1) vertex has at least one $F$-vertex descendant with the next lower even rank.
Properties of minimal full rankings

Global properties (descendants)

- Every even ranked vertex has at least one descendant with the next lower odd rank.
- Every odd ranked (> 1) vertex has at least one $F$-vertex descendant with the next lower even rank.
- Every path from every even ranked vertex eventually encounters a vertex with a lower rank.
Properties of minimal full rankings

Global properties (descendants)

- Every even ranked vertex has at least one descendant with the next lower odd rank.
- Every odd ranked (> 1) vertex has at least one $F$-vertex descendant with the next lower even rank.
- Every path from every even ranked vertex eventually encounters a vertex with a lower rank.
- Every $\infty$ ranked vertex has at least one $\infty$ ranked $F$-vertex descendant.
Properties of minimal full rankings

Global properties (descendants)

- Every even ranked vertex has at least one descendant with the next lower odd rank.
- Every odd ranked (> 1) vertex has at least one $F$-vertex descendant with the next lower even rank.
- Every path from every even ranked vertex eventually encounters a vertex with a lower rank.
- Every $\infty$ ranked vertex has at least one $\infty$ ranked $F$-vertex descendant.
- Every $\infty$ ranked vertex has at least one descendant with the largest non-infinity rank in range of the ranking function.
Intuition of disambiguation construction

Vertices with qs not shown for clarity

Level 3
Level 2
Level 1
Level 0

q1
q2
q3
q4
q5

∞
4
3
2
1

Hrishikesh Karmarkar Supratik Chakraborty

Ranking based Techniques for Disambiguating Büchi Automata
Intuition of disambiguation construction

Vertices with qs not shown for clarity

Level 0

Level 1

Level 2

Level 3
Construct an automaton whose states are full-ranked levels of run DAG.
Construct an automaton whose states are full-ranked levels of run DAG

- Goal: Only minimally full-ranked levels must be accepted
Construct an automaton whose states are full-ranked levels of run DAG
  - Goal: Only minimally full-ranked levels must be accepted
  - Local properties of minimal full-ranking easy to enforce in transition relation

Global properties checked one vertex (and also one rank) at a time
Decompose every global property of an infinite run into properties of finite segments of the run, which can then be concatenated.
Ensure that each finite segment satisfies relevant property checkable over finite steps
Acceptance condition simply ensures that every finite segment of an infinite run satisfies relevant properties and root vertex is ranked $\infty$
Construct an automaton whose states are full-ranked levels of run DAG

- Goal: Only minimally full-ranked levels must be accepted
- Local properties of minimal full-ranking easy to enforce in transition relation
- Enforcing global properties requires maintaining additional book-keeping information

Global properties checked one vertex (and also one rank) at a time
Decompose every global property of an infinite run into properties of finite segments of the run, which can then be concatenated.
Ensure that each finite segment satisfies relevant property checkable over finite steps
Acceptance condition simply ensures that every finite segment of an infinite run satisfies relevant properties and root vertex is ranked $\infty$
Construct an automaton whose states are full-ranked levels of run DAG
  - Goal: Only minimally full-ranked levels must be accepted

Local properties of minimal full-ranking easy to enforce in transition relation

Enforcing global properties requires maintaining additional book-keeping information
  - Global properties checked one vertex (and also one rank) at a time
  - Decompose every global property of an infinite run into properties of finite segments of the run, which can then be concatenated.
  - Ensure that each finite segment satisfies relevant property checkable over finite steps
Disambiguation construction

- Construct an automaton whose states are full-ranked levels of run DAG
  - Goal: Only minimally full-ranked levels must be accepted
- Local properties of minimal full-ranking easy to enforce in transition relation
- Enforcing global properties requires maintaining additional book-keeping information
  - Global properties checked one vertex (and also one rank) at a time
  - Decompose every global property of an infinite run into properties of finite segments of the run, which can then be concatenated.
  - Ensure that each finite segment satisfies relevant property checkable over finite steps
- Acceptance condition simply ensures that every finite segment of an infinite run satisfies relevant properties and root vertex is ranked $\infty$
State representation

State of resulting automaton:

\[(S, O, X, f, i), \text{ where}\]

- \(S\) : subset of states of NBW in current level
- \(f\) : ranking function at current level
- \(i\) : rank of vertices for which (decomposed) global properties are currently being checked
- \(O \subseteq S\) : subset of states with rank \(i\) for which global properties yet to be checked
- \(X \subseteq S\) : subset of states being used to check global property of one state with rank \(i\)
State representation

State of resulting automaton:

\((S, O, X, f, i)\), where

- \(S\) : subset of states of NBW in current level
- \(f\) : ranking function at current level
- \(i\) : rank of vertices for which (decomposed) global properties are currently being checked
- \(O \subseteq S\) : subset of states with rank \(i\) for which global properties yet to be checked
- \(X \subseteq S\) : subset of states being used to check global property of one state with rank \(i\)

Total count of states is \(O(n^2 \cdot (0.76n)^n)\)

- Uses a modification of a counting argument used by Schewe (2009) for NBW complementation
Why is it unambiguous?

Recall minimal full-ranking for every run DAG is unique. Our construction accepts only those runs that enforce both local and global properties of minimal full-ranking. Accepted full-ranking is minimal. Any two accepting runs must differ in the ranking of at least one level. Since minimal ranking is unique, only one accepting run possible.
Why is it unambiguous?

- Recall minimal full-ranking for every run DAG is unique.
Why is it unambiguous?

- Recall minimal full-ranking for every run DAG is unique.
- Our construction accepts only those runs that enforce both local and global properties of minimal full-ranking.
Why is it unambiguous?

- Recall minimal full-ranking for every run DAG is unique.
- Our construction accepts only those runs that enforce both local and global properties of minimal full-ranking
  - Accepted full-ranking is minimal
Recall minimal full-ranking for every run DAG is unique.

Our construction accepts only those runs that enforce both local and global properties of minimal full-ranking
  - Accepted full-ranking is minimal

Any two accepting runs must differ in the ranking of at least one level
Why is it unambiguous?

- Recall minimal full-ranking for every run DAG is unique.
- Our construction accepts only those runs that enforce both local and global properties of minimal full-ranking
  - Accepted full-ranking is minimal
- Any two accepting runs must differ in the ranking of at least one level
- Since minimal ranking is unique, only one accepting run possible
Using a variant of KV-ranking (similar to that used by Carton and Michel), we obtain a UNBW (not SUNBW) with better bound than reported in the literature.

We conjecture that this matches the lower bound for disambiguation.

Shows potential close connection between disambiguation and complementation.