Ranking based Techniques for Disambiguating Büchi Automata

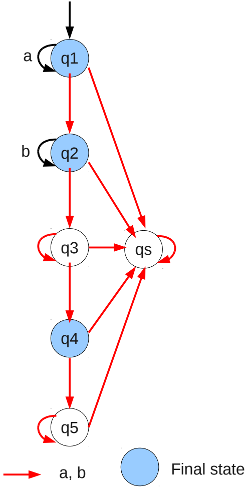
Hrishikesh Karmarkar Supratik Chakraborty

January 29, 2011

Non-deterministic Büchi automata over words (NBW)

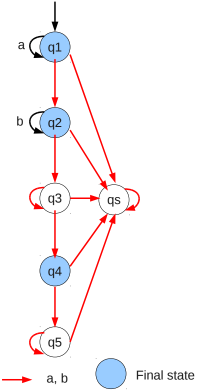
A 5−tuple $(\Sigma, Q, Q_0, \delta, F)$, where

- Σ : 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



Runs and acceptance

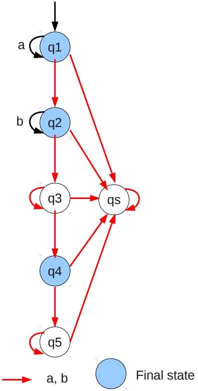
- A *run* of $\mathcal{A}$ on $\alpha \in \Sigma^\omega$ is a sequence $\rho : \mathbb{N} \rightarrow Q$ such that
  - $\rho(0) \in Q_0$
  - $\rho(i+1) \in \delta(\rho(i), \alpha(i))$



- $\alpha = abbbbb \cdots, \ \rho_1 = q_1 q_2 q_2 q_2 q_2 q_2 q_2 \cdots$

Runs and acceptance

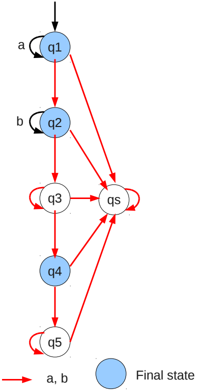
- A *run* of $\mathcal{A}$ on $\alpha \in \Sigma^\omega$ is a sequence $\rho : \mathbb{N} \rightarrow Q$ such that
  - $\rho(0) \in Q_0$
  - $\rho(i+1) \in \delta(\rho(i), \alpha(i))$
- An automaton may have several runs on α.



- $\alpha = abbbbb\cdots$, $\rho_1 = q_1 q_2 q_2 q_2 q_2 q_2 q_2 \cdots$
- $\rho_2 = q_1 q_1 q_2 q_2 q_2 q_2 q_2 \cdots$

Runs and acceptance

- 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 α.
- ρ is accepting iff $\inf(\rho) \cap F \neq \emptyset$
- α is accepted by $\mathcal{A}$ ($\alpha \in L(\mathcal{A})$) iff there is an accepting run of $\mathcal{A}$ on α.



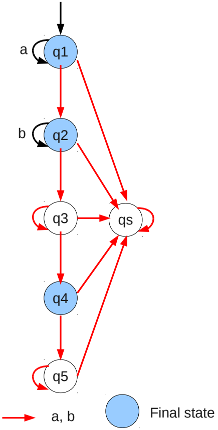
- $\alpha = abbbbb\cdots$, $\rho_1 = q_1 q_2 q_2 q_2 q_2 q_2 q_2 \cdots$
- $\rho_2 = q_1 q_1 q_2 q_2 q_2 q_2 q_2 \cdots$

Ambiguous automata

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

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

An ambiguous NBW



- $\alpha = ab^\omega$
- $\rho_1 = q_1 q_2^\omega$
- $\rho_2 = q_1 q_1 q_2^\omega$

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

Strongly Unambiguous automata

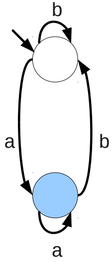
- *Final run* of $\mathcal{A}$ on α: A run ρ starting from *any* state in Q such that $\inf(\rho) \cap F \neq \emptyset$.
    - A word $\notin L(\mathcal{A})$ may have 0 or more final runs
    - A word $\in L(A)$ has ≥ 1 final runs
- NBW $\mathcal{A}$ is *strongly unambiguous* if for every $\alpha \in \Sigma^\omega$, there is exactly one final run.

Strongly Unambiguous automata

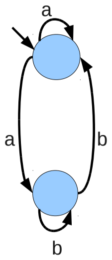
- *Final run* of $\mathcal{A}$ on α: A run ρ starting from *any* state in Q such that $\inf(\rho) \cap F \neq \emptyset$.
  - A word $\notin L(\mathcal{A})$ may have 0 or more final runs
  - A word $\in L(A)$ has ≥ 1 final runs
- NBW $\mathcal{A}$ is *strongly unambiguous* if for every $\alpha \in \Sigma^\omega$, there is exactly one final run.
- Not all unambiguous automata are strongly unambiguous.

Strongly Unambiguous automata

- *Final run* of $\mathcal{A}$ on α: A run ρ starting from *any* state in Q such that $\inf(\rho) \cap F \neq \emptyset$.
    - A word $\notin L(\mathcal{A})$ may have 0 or more final runs
    - A word $\in L(A)$ has ≥ 1 final runs
- NBW $\mathcal{A}$ is *strongly unambiguous* if for every $\alpha \in \Sigma^\omega$, there is exactly one final run.
- Not all unambiguous automata are strongly unambiguous.



Deterministic (hence unambiguous) but not strongly unambiguous



Strongly unambiguous

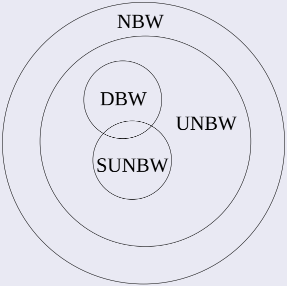
Containment relations

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



What this talk is about

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

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

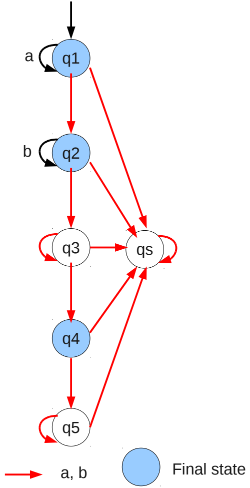
Our contribution

- Effective construction of UNBW, size bound $O(n^2.(0.76n)^n)$
  - Same as best known bound for NBW complementation!

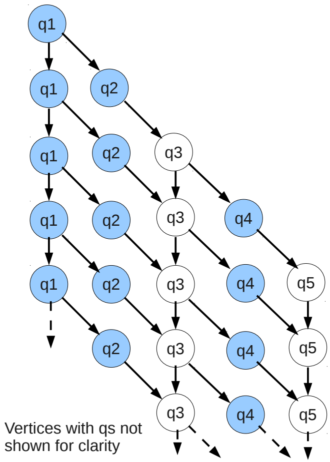
Why care about disambiguation?

- 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^ω



Ranking run DAGs

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

Ranking run DAGs

- Intuitively, assign a metric to each vertex in run DAG such that the metric changes in a desirable way only along "good" runs.
- Early work by Michel (1984?), Klarlund (1991): Ranking functions/progress measures for Büchi complementation

Ranking run DAGs

- Intuitively, assign a metric to each vertex in run DAG such that the metric changes in a desirable way only along "good" runs.
- Early work by Michel (1984?), Klarlund (1991): Ranking functions/progress measures for Büchi complementation
- Recent spurt of work triggered by similar metrics defined by Kupferman & Vardi (2001 onwards)

Ranking run DAGs

- Intuitively, assign a metric to each vertex in run DAG such that the metric changes in a desirable way only along "good" runs.
- Early work by Michel (1984?), Klarlund (1991): Ranking functions/progress measures for Büchi complementation
- 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

Ranking run DAGs

- Intuitively, assign a metric to each vertex in run DAG such that the metric changes in a desirable way only along "good" runs.
- Early work by Michel (1984?), Klarlund (1991): Ranking functions/progress measures for Büchi complementation
- 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: Number of states in NBW
V: Set of run DAG vertices
$r : V \rightarrow \{1, 2, \ldots 2n + 1\}$: 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

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

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

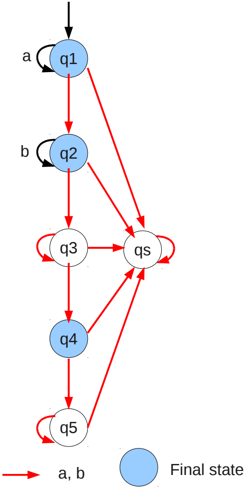
Constraints on ranks

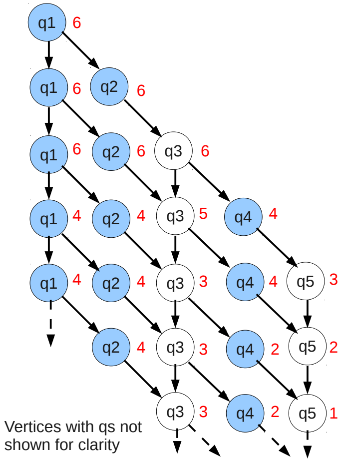
- Vertices corresponding to final states must not get odd ranks
- Ranking cannot increase along any path in run DAG

- *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^ω

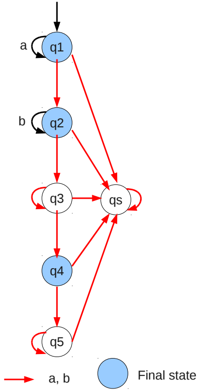




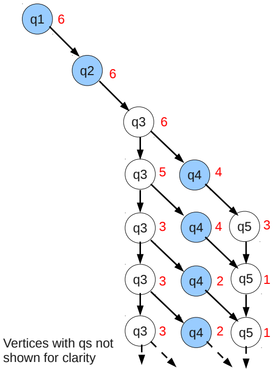
Ranking based complementation

Theorem (Kupferman-Vardi 2001)

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



Example ranking for ba^ω



Applications of Kupferman-Vardi's theorem

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

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 $O(n^2.(0.76n)^n)$

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 $\Omega((0.76n)^n)$.

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

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!

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 $O(n^2, (0.76n)^n)$.

Extending KV-ranks

Recall KV-ranking

Extending KV-ranks

Recall KV-ranking

n: Number of states in NBW

V: Set of run DAG vertices

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

Extending KV-ranks

Recall KV-ranking

n: Number of states in NBW

V: Set of run DAG vertices

$r : V \to \{1, 2, \ldots 2n+1\}$ $\boxed{\cup\{\infty\}}$: Ranking function

Constraints on ranks

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

Extending KV-ranks

Recall KV-ranking

n: Number of states in NBW

V: Set of run DAG vertices

$r : V \to \{1, 2, \ldots 2n + 1\}$ $\boxed{\cup\{\infty\}}$: 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 ∞

Extending KV-ranks

Recall KV-ranking

n: Number of states in NBW

V: Set of run DAG vertices

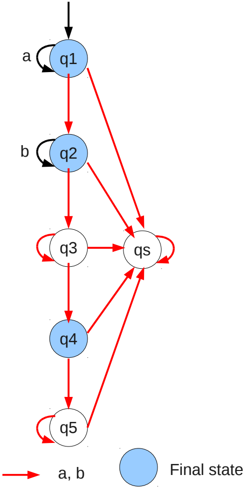
$r : V \rightarrow \{1, 2, \ldots 2n + 1\}$ $\cup\{\infty\}$: Ranking function

Constraints on ranks

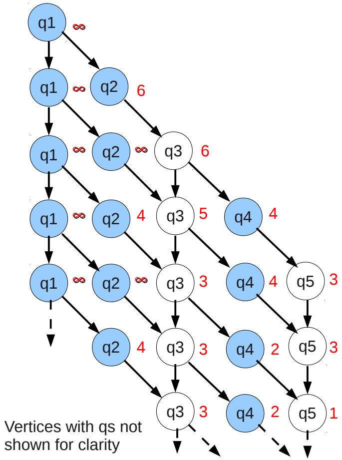
- Vertices corresponding to final states must not get odd ranks
- Ranking cannot increase along any path in run DAG
- Every path eventually trapped in odd rank or in ∞

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

Example of full ranking



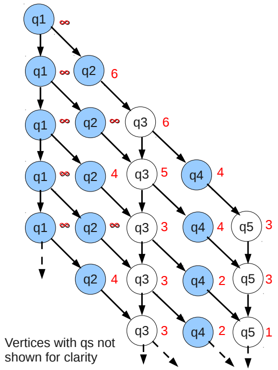
Example full ranking for a^{ω}



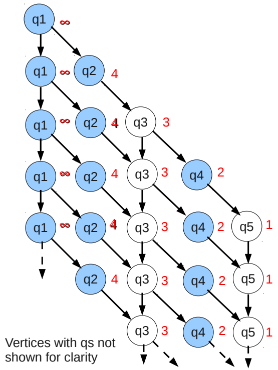
Minimal full rankings

Given run DAG G, full ranking r^* of G is **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



Properties of minimal full rankings

Theorem

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

Properties of minimal full rankings

Theorem

For every run DAG, there exists a unique minimal full ranking. A word α is accepted by $\mathcal{A}$ iff the minimal full ranking of the run DAG assigns ∞ 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 α is accepted by $\mathcal{A}$ iff the minimal full ranking of the run DAG assigns ∞ 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 α is accepted by $\mathcal{A}$ iff the minimal full ranking of the run DAG assigns ∞ 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 α is accepted by $\mathcal{A}$ iff the minimal full ranking of the run DAG assigns ∞ 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)

Properties of minimal full rankings

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)

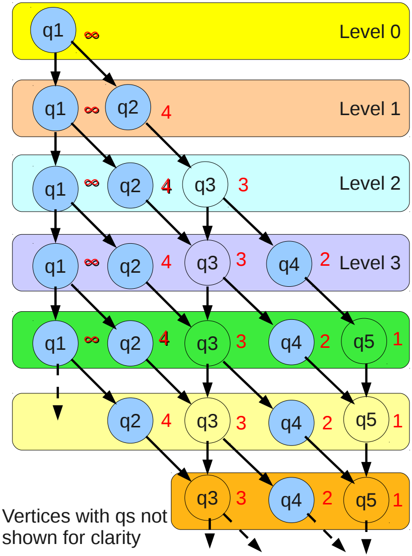
- Every even ranked vertex has at least one descendant with the next lower odd rank
- Every odd ranked (> 1) vertex has at least one F-vertex descendant with the next lower even rank
- Every path from every even ranked vertex eventually encounters a vertex with a lower rank
- Every ∞ ranked vertex has at least one ∞ ranked F-vertex descendant

Properties of minimal full rankings

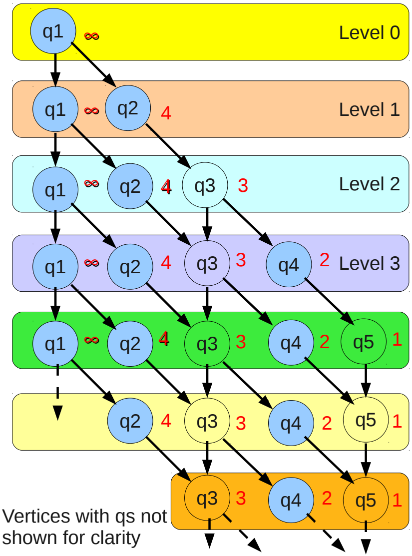
Global properties (descendants)

- 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 ∞ ranked vertex has at least one ∞ ranked F-vertex descendant
- Every ∞ ranked vertex has at least one descendant with the largest non-infinity rank in range of the ranking function.

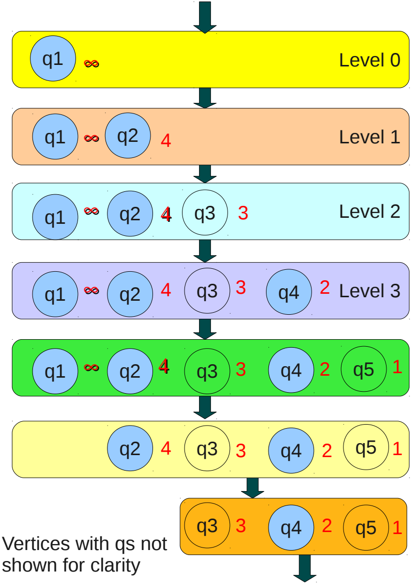
Intuition of disambiguation construction



Intuition of disambiguation construction



Vertices with qs not shown for clarity



Vertices with qs not shown for clarity

Disambiguation construction

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

Disambiguation construction

- Construct an automaton whose states are full-ranked levels of run DAG
  - Goal: Only minimally full-ranked levels must be accepted

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

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

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

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 ∞

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:

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

- 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

Total count of states is $O(n^2.(0.76n)^n)$

- Uses a modification of a counting argument used by Schewe (2009) for NBW complementation

Why is it unambiguous?

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

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

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

Conclusion

- 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