Width of Non-Deterministic Automata

Denis Kuperberg [1], Anirban Majumdar [2]

[1]ENS Lyon, France

[2]Chennai Mathematical Institute, India

March 01, 2018

Introduction

- NFA $\rightarrow$ DFA: Important
  - Complementation of NFA
  - Language inclusion testing
  - Synthesis of reactive controllers
  - Text searching
  - Regular Expression matching

    ...

Introduction

- NFA $\rightarrow$ DFA: Important
  - Complementation of NFA
  - Language inclusion testing
  - Synthesis of reactive controllers
  - Text searching
  - Regular Expression matching

  ...

- NFA $\rightarrow$ DFA: Hard
  - Exponential blow-up unavoidable.
  - Heuristics are developed to speed-up determinization or avoid it.

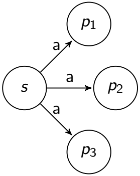
Introduction

- NFA $\rightarrow$ DFA

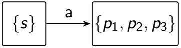
Introduction

- NFA $\rightarrow$ DFA : Subset Construction

- NFA $\rightarrow$ DFA : Subset Construction

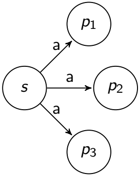


$\rightarrow$

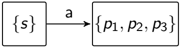


Introduction

- NFA $\rightarrow$ DFA : Subset Construction



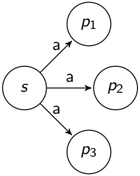
$\rightarrow$



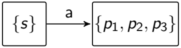
  - Follows n runs simultaneously (n is the number of states)

Introduction

- NFA $\rightarrow$ DFA : Subset Construction

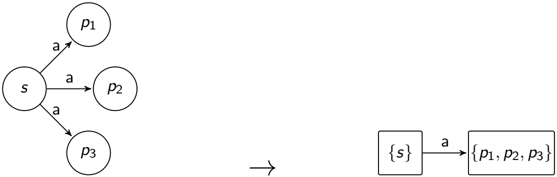


$\rightarrow$



  - Follows n runs simultaneously (n is the number of states)
  - Is 'n' always necessary ?

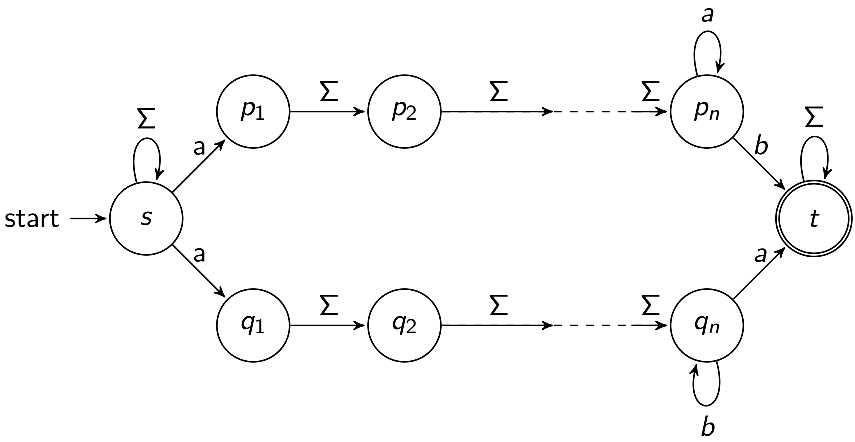
- NFA → DFA : Subset Construction



  - Follows n runs simultaneously (n is the number of states)
  - Is *‘n’* always necessary ?
  - Can we get a DFA keeping track of *‘k’ runs* ? ($k < n$)

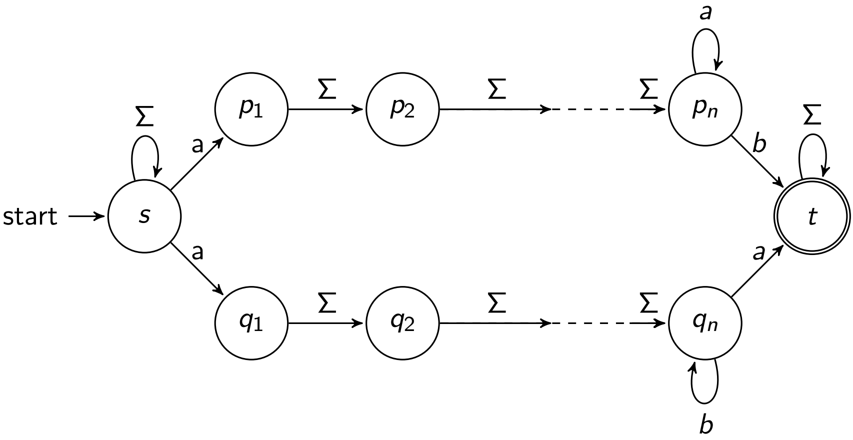
Example...

$\Sigma = \{a, b\}$



Example...

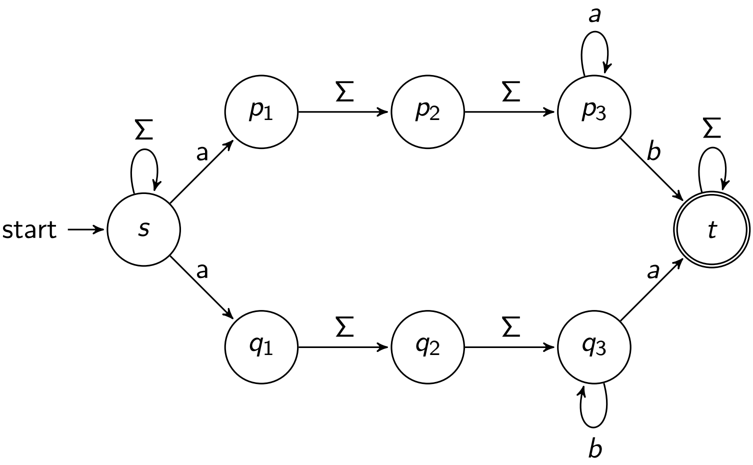
$\Sigma = \{a, b\}$



Language: $\Sigma^* a \Sigma^{\geq n}$

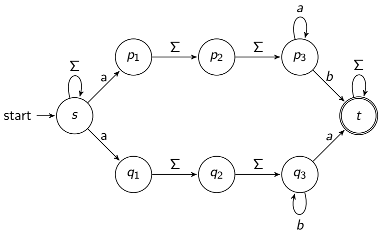
Example...

$\mathbf{n = 3}$:

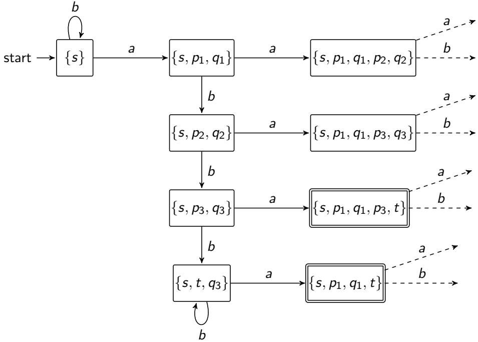


Language: $\Sigma^* a \Sigma^{\geq 3}$

Example...

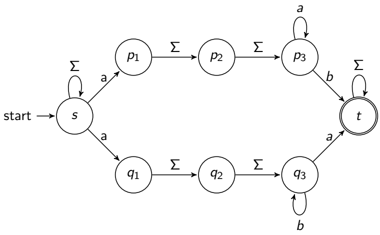


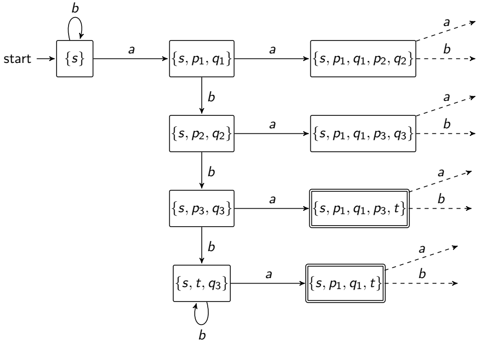
start
s
p_1
p_2
p_3
q_1
q_2
q_3
t
Σ
a
b



start
$\{s\}$
$\{s, p_1, q_1\}$
$\{s, p_1, q_1, p_2, q_2\}$
$\{s, p_2, q_2\}$
$\{s, p_1, q_1, p_3, q_3\}$
$\{s, p_3, q_3\}$
$\{s, p_1, q_1, p_3, t\}$
$\{s, t, q_3\}$
$\{s, p_1, q_1, t\}$
a
b

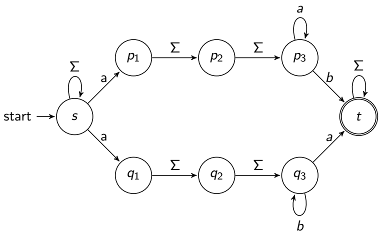
Example...

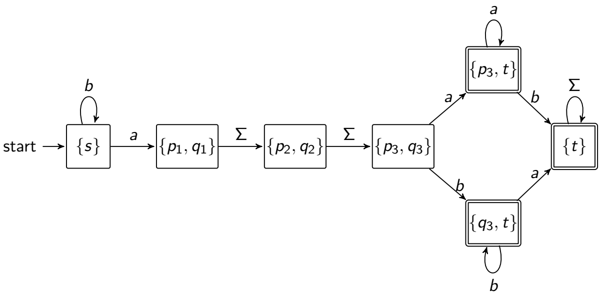




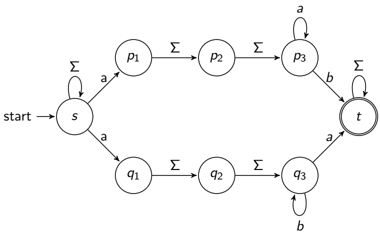
Exponential number of states

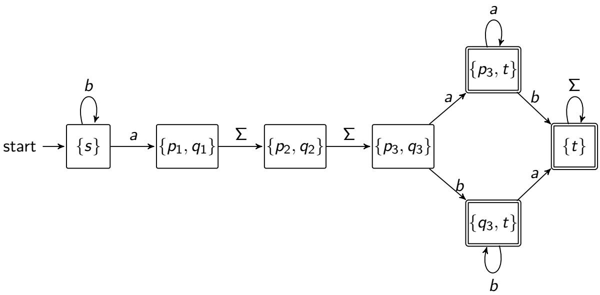
Example...


start
s
a
a
Σ
p1
q1
Σ
Σ
p2
q2
Σ
Σ
p3
q3
a
b
b
a
t
Σ


start
{s}
b
a
{p1, q1}
Σ
{p2, q2}
Σ
{p3, q3}
a
b
{p3, t}
a
{q3, t}
b
b
a
{t}
Σ

Example...

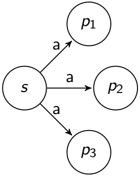




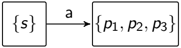
2 runs are sufficient

Introduction

- NFA $\rightarrow$ DFA : Subset Construction



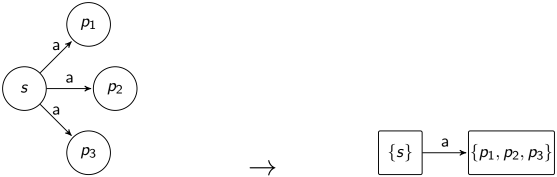
$\rightarrow$



  - Follows n runs simultaneously (n is the number of states)
  - Is 'n' always necessary ?
  - Can we get a DFA keeping track of 'k' *runs* ? ($k < n$)
  - **Save space in determinization.**

Introduction

- NFA $\rightarrow$ DFA : Subset Construction

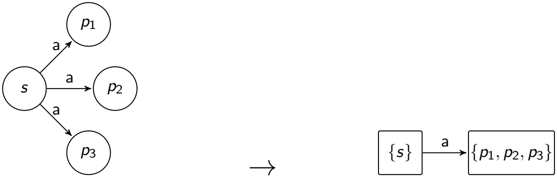


  - Follows n runs simultaneously (n is the number of states)
  - Is *'n'* always necessary ?
  - Can we get a DFA keeping track of *'k' runs* ? ($k < n$)
  - **Save space in determinization.**

- Goal : Find *minimum 'k'* such that k runs are enough.

Introduction

- NFA $\rightarrow$ DFA : Subset Construction



  - Follows *n* runs simultaneously (n is the number of states)
  - Is *'n'* always necessary ?
  - Can we get a DFA keeping track of *'k' runs* ? ($k < n$)
  - **Save space in determinization.**

- Width : *minimum 'k'* such that k runs are enough.

Good-For-Games Automata

Good-For-Games Automata

$\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a NFA
Consider the following two-player game on $\mathcal{A}$.

$\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a NFA

Consider the following two-player game on $\mathcal{A}$.

In each round, Adam chooses a letter and Eve chooses a transition.

$\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a NFA
Consider the following two-player game on $\mathcal{A}$.
In each round, Adam chooses a letter and Eve chooses a transition.

q_0

Good-For-Games Automata

$\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a NFA
Consider the following two-player game on $\mathcal{A}$.
In each round, Adam chooses a letter and Eve chooses a transition.

$$q_0 \xrightarrow{a_0}$$

$\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a NFA
Consider the following two-player game on $\mathcal{A}$.
In each round, Adam chooses a letter and Eve chooses a transition.

$$q_0 \xrightarrow{a_0} q_1$$

Good-For-Games Automata

$\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a NFA
Consider the following two-player game on $\mathcal{A}$.
In each round, Adam chooses a letter and Eve chooses a transition.

$$q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1}$$

Good-For-Games Automata

$\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a NFA
Consider the following two-player game on $\mathcal{A}$.
In each round, Adam chooses a letter and Eve chooses a transition.

$$q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} q_3 \rightarrow \cdots$$

Good-For-Games Automata

$\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a NFA
Consider the following two-player game on $\mathcal{A}$.
In each round, Adam chooses a letter and Eve chooses a transition.

$$q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} q_3 \rightarrow \cdots$$

Eve wins a play if ($w \in \mathcal{L}(\mathcal{A}) \Rightarrow$ the run is accepting).

$\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a NFA
Consider the following two-player game on $\mathcal{A}$.
In each round, Adam chooses a letter and Eve chooses a transition.

$$q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} q_3 \rightarrow \cdots$$

Eve wins a play if ($w \in \mathcal{L}(\mathcal{A}) \Rightarrow$ the run is accepting).
$\mathcal{A}$ is called **GFG** iff Eve has a winning strategy

Good-For-Games Automata

$\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a NFA
Consider the following two-player game on $\mathcal{A}$.
In each round, Adam chooses a letter and Eve chooses a transition.

$$q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} q_3 \rightarrow \cdots$$

Eve wins a play if ($w \in \mathcal{L}(\mathcal{A}) \Rightarrow$ the run is accepting).
$\mathcal{A}$ is called **GFG** iff Eve has a winning strategy : Whatever word Adam chooses, if it is in the language, the strategy gives an accepting run.

Good-For-Games Automata

$\mathcal{A} = (Q, \Sigma, \delta, q_0, \alpha)$ be an automaton with acceptance condition α.
Consider the following two-player game on $\mathcal{A}$.
In each round, Adam chooses a letter and Eve chooses a transition.

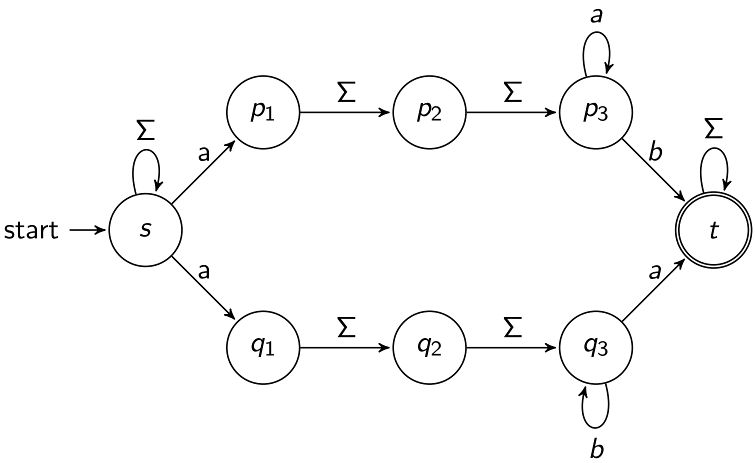
$$q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} q_3 \rightarrow \cdots$$

Eve wins a play if ($w \in \mathcal{L}(\mathcal{A}) \Rightarrow$ the run is accepting).
$\mathcal{A}$ is called **GFG** iff Eve has a winning strategy : Whatever word Adam chooses, if it is in the language, the strategy gives an accepting run.

Example GFG

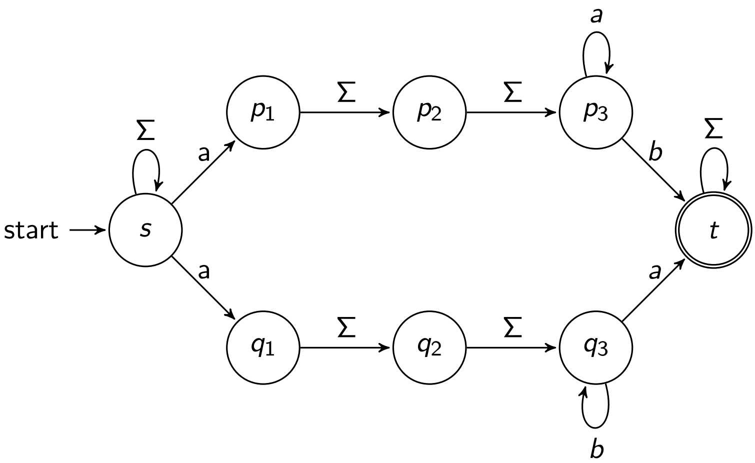
- Every DFA is GFG.
  - Eve's choice is deterministic; if the word chosen by Adam is in the language of the automaton, then the run is accepting.

Example Non-GFG



Language: $\Sigma^* a \Sigma^{\geq 3}$.

Example Non-GFG



Language: $\Sigma^* a \Sigma^{\geq 3}$.

If Eve chooses the upper path, $a^3 a$ is not accepted.

If Eve chooses the lower path, $a^3 b$ is not accepted.

Some Results...

Some Results...

On finite words,

- $\mathcal{A}$ is GFG $\Rightarrow$ $\mathcal{A}$ can be determinized deleting some unnecessary transitions

Some Results...

On finite words,

- $\mathcal{A}$ is GFG $\Rightarrow$ $\mathcal{A}$ can be determinized deleting some unnecessary transitions (DBP).

Some Results...

On finite words,

- $\mathcal{A}$ is GFG $\Rightarrow$ $\mathcal{A}$ can be determinized deleting some unnecessary transitions (DBP).
- Checking GFGness of a NFA is in **P**

Some Results...

On finite words,

- $\mathcal{A}$ is GFG $\Rightarrow$ $\mathcal{A}$ can be determinized deleting some unnecessary transitions (DBP).
- Checking GFGness of a NFA is in **P** and removing useless transitions to obtain a DFA is also in **P**.

Width

$\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$ be a NFA.
We define width($\mathcal{A}$):

Width

$\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$ be a NFA.
We define width($\mathcal{A}$):
We play the GFG game on $\mathcal{A}$, except that now Eve can choose sets of size at most 'k'.

Width

$\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$ be a NFA.
We define width($\mathcal{A}$):
We play the GFG game on $\mathcal{A}$, except that now Eve can choose sets of size at most 'k'.

$$\{q_0\} \xrightarrow{a_0} X_1 \xrightarrow{a_1} X_2 \xrightarrow{a_2} X_3 \rightarrow \cdots$$

$|X_i| \leq k$.

Width

$\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$ be a NFA.
We define width($\mathcal{A}$):
We play the GFG game on $\mathcal{A}$, except that now Eve can choose sets of size at most ‘k’.

$$\{q_0\} \xrightarrow{a_0} X_1 \xrightarrow{a_1} X_2 \xrightarrow{a_2} X_3 \rightarrow \cdots$$

$|X_i| \leq k$.
Eve wins a play if $(a_1 a_2 \ldots a_r \in \mathcal{L}(\mathcal{A}) \Rightarrow X_r \cap F \neq \emptyset)$ for all r.

Width

$\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$ be a NFA.
We define width($\mathcal{A}$):
We play the GFG game on $\mathcal{A}$, except that now Eve can choose sets of size at most 'k'.

$$\{q_0\} \xrightarrow{a_0} X_1 \xrightarrow{a_1} X_2 \xrightarrow{a_2} X_3 \rightarrow \cdots$$

$|X_i| \leq k$.
Eve wins a play if $(a_1 a_2 \ldots a_r \in \mathcal{L}(\mathcal{A}) \Rightarrow X_r \cap F \neq \emptyset)$ for all r.

If Eve has a winning strategy then we say width($\mathcal{A}$) $\leq k$.

Width

$\mathcal{A} = (Q, \Sigma, q_0, \Delta, \alpha)$ be an automaton with acceptance condition α.
We define width($\mathcal{A}$):
We play the GFG game on $\mathcal{A}$, except that now Eve can choose sets of size at most 'k'.

$$\{q_0\} \xrightarrow{a_0} X_1 \xrightarrow{a_1} X_2 \xrightarrow{a_2} X_3 \rightarrow \cdots$$

$|X_i| \leq k$.
Eve wins a play if $(a_1 a_2 \ldots \quad \in \mathcal{L}(\mathcal{A}) \Rightarrow$the sequence $X_0 X_1 X_2 \ldots$ contains an accepting run of $\mathcal{A}$).
If Eve has a winning strategy then we say width($\mathcal{A}$) $\leq k$.

Width

$\mathcal{A} = (Q, \Sigma, q_0, \Delta, \alpha)$ be an automaton with acceptance condition α.
We define width($\mathcal{A}$):
We play the GFG game on $\mathcal{A}$, except that now Eve can choose sets of size at most 'k'.

$$\{q_0\} \xrightarrow{a_0} X_1 \xrightarrow{a_1} X_2 \xrightarrow{a_2} X_3 \rightarrow \cdots$$

$|X_i| \leq k$.
Eve wins a play if $(a_1 a_2 \ldots \quad \in \mathcal{L}(\mathcal{A}) \Rightarrow$the sequence $X_0 X_1 X_2 \ldots$ contains an accepting run of $\mathcal{A}$).
If Eve has a winning strategy then we say width($\mathcal{A}$) $\leq k$.

- An automaton being GFG is equivalent to having width 1.

k-Subset Construction

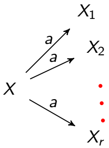
k-Subset Construction

$\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$ be a NFA.
$\mathcal{A}_k$ be the subset construction where size of each set is bounded by 'k'.

k-Subset Construction

$\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$ be a NFA.
$\mathcal{A}_k$ be the subset construction where size of each set is bounded by 'k'.
$\mathcal{A}_k = (Q_k, \Sigma, \{q_0\}, \Delta', F')$, where

- Δ' :

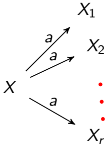


, where $|X_i| \leq k$; and $X_i \subseteq \Delta(X, a)$

k-Subset Construction

$\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$ be a NFA.
$\mathcal{A}_k$ be the subset construction where size of each set is bounded by 'k'.
$\mathcal{A}_k = (Q_k, \Sigma, \{q_0\}, \Delta', F')$, where

- Δ' :



, where $|X_i| \leq k$; and $X_i \subseteq \Delta(X, a)$

- $\mathcal{A}$ has **width** $\leq$ **k** iff $\mathcal{A}_k$ is GFG.

Some Results...

On finite words,

- $\mathcal{A}$ is GFG $\Rightarrow$ $\mathcal{A}$ is DBP.
- Checking GFGness of a NFA is in **P**.
- Checking if width($\mathcal{A}$) $\leq$ k is PSPACE-hard and in EXPTIME.

On finite words,

- $\mathcal{A}$ is GFG $\Rightarrow$ $\mathcal{A}$ is DBP.
- Checking GFGness of a NFA is in **P**.
- Checking if width($\mathcal{A}$) $\leq$ k is PSPACE-hard and in EXPTIME.
  Hardness Proof : Reduction from "Universality problem for NFA".

Determinization

- NFA $\rightarrow$ DFA :

Determinization

- NFA $\rightarrow$ DFA :

  We check if $\mathcal{A}$ is GFG.

Determinization

- NFA $\rightarrow$ DFA :

  We check if $\mathcal{A}$ is GFG. Yes

Determinization

- NFA $\rightarrow$ DFA :

  We check if $\mathcal{A}$ is GFG. $\mathcal{A}$ is DBP, hence determinize $\mathcal{A}$

Determinization

- NFA $\rightarrow$ DFA :

  We check if $\mathcal{A}$ is GFG. No

Determinization

- NFA $\rightarrow$ DFA :

  We check if $\mathcal{A}$ is GFG. No
  We check if $\mathcal{A}_2$ is GFG.

Determinization

- NFA $\rightarrow$ DFA :

  We check if $\mathcal{A}$ is GFG. No
  We check if $\mathcal{A}_2$ is GFG. Yes

Determinization

- NFA $\rightarrow$ DFA :

  We check if $\mathcal{A}$ is GFG. No

  We check if $\mathcal{A}_2$ is GFG. $\mathcal{A}_2$ is DBP, hence determinize $\mathcal{A}_2$

- NFA $\rightarrow$ DFA :

  We check if $\mathcal{A}$ is GFG. No
  We check if $\mathcal{A}_2$ is GFG. No
  $\ldots$

Determinization

- NFA $\rightarrow$ DFA :

  We check if $\mathcal{A}$ is GFG. No
  We check if $\mathcal{A}_2$ is GFG. No
  ...
  In the worst case, we reach $\mathcal{A}_n$ and do the whole subset construction.

Some Results...

On infinite words, if the acceptance condition is Co-Büchi condition,

- GFG automata are not necessarily DBP.
- GFG automata are useful for synthesis and other purposes.
- In some cases, GFG automata are exponentially smaller than any equivalent deterministic automata.

Some Results...

On infinite words, if the acceptance condition is Co-Büchi condition,

- GFG automata are not necessarily DBP.
- GFG automata are useful for synthesis and other purposes.
- In some cases, GFG automata are exponentially smaller than any equivalent deterministic automata.
- Checking DBPness is NP-complete.
- Checking GFGness is in P. **[Kuperberg, Skrzypczak ’15]**
- We can find the width almost in the same way as we did for NFA.

Co-Büchi Case

Determinization : Breakpoint Construction

Co-Büchi Case

Determinization : Breakpoint Construction

$\mathcal{A}_k$: Keep track of sets of size atmost k (**k-Breakpoint Construction**)

Co-Büchi Case

Determinization : Breakpoint Construction

$\mathcal{A}_k$: Keep track of sets of size atmost k (**k-Breakpoint Construction**)

- $\mathcal{A}$ has **width** $\leq$ **k** iff $\mathcal{A}_k$ is GFG.

Calculate Width

We check if $\mathcal{A}$ is GFG.

Calculate Width

We check if $\mathcal{A}$ is GFG. Yes

Calculate Width

We check if $\mathcal{A}$ is GFG. $\mathit{Width} = 1$

Calculate Width

We check if $\mathcal{A}$ is GFG. No

Calculate Width

We check if $\mathcal{A}$ is GFG. No

We check if $\mathcal{A}_2$ is GFG.

Calculate Width

We check if $\mathcal{A}$ is GFG. No

We check if $\mathcal{A}_2$ is GFG. Yes

Calculate Width

We check if $\mathcal{A}$ is GFG. No

We check if $\mathcal{A}_2$ is GFG. $Width = 2$

Calculate Width

We check if $\mathcal{A}$ is GFG. No

We check if $\mathcal{A}_2$ is GFG. No

$\dots$

Calculate Width

We check if $\mathcal{A}$ is GFG. No

We check if $\mathcal{A}_2$ is GFG. No

$\dots$

In the worst case, we reach $\mathcal{A}_n$ and do the whole breakpoint construction. $Width = n$

Büchi Case

For Büchi? : Similar ; k-Safra

Büchi Case

For Büchi? : Similar ; k-Safra :

- Only build Safra trees labelled by at most k states.

Büchi Case

For Büchi? : Similar ; k-Safra :

- Only build Safra trees labelled by at most k states.
- $\mathcal{A}$ has **width** $\leq$ **k** iff $\mathcal{A}_k$ is GFG.

Summary

- Iterative construction to translate **nondeterministic** automata to **GFG**.
- Over finite words, GFG is equivalent to DBP; hence we get DFA.
- For Co-Büchi, testing GFG is easier than checking DBPness.
- Can extend for Büchi acceptance condition also.

Concluding Remarks

- **Future Work :**
  - Complexity of GFGness checking for arbitrary Parity/Rabin conditions (essential bottleneck in our algorithm with Büchi input).
  - Understanding the link between width and structure of the automaton.
  - Implementing this approach and testing it against classical determinization/inclusion software.

  ...

Questions?

Thank You