Multi-Island Finite Automata and Their Even Computation

Martin Tomko

paper co-authored with:
Dušan Kolář, Alexander Meduna

Faculty of Information Technology, BUT

December 5, 2022

Table of contents

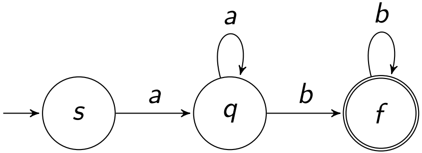
Finite Automata

Finite Automata: Example, Graphical Representation

The GFA

$$M = (\{s, q, f\}, \{a, b\}, \{sa \to q, qa \to q, qb \to f, fb \to b\}, s, f)$$

can be represented as:



The language accepted by this automaton is

$$L(M) = \{a^n b^m \mid n, m \geq 1\}.$$

Finite Automata: Definition

A *generalized finite automaton* (GFA) is a 5-tuple $M = (Q, \Sigma, R, s, f)$, where

- Q – a finite *set of states*,
- Σ – a finite, nonempty *input alphabet*,
- $R \subseteq Q \times \Sigma^* \times Q$ – a finite set of *rules*:
  - $(p, w, q) \in R$ written as $pw \to q$,
- $s \in Q$ – the *initial state*,
- $f \in Q$ – the *final state*.

Finite Automata: Definition

A *generalized finite automaton* (GFA) is a 5-tuple
$M = (Q, \Sigma, R, s, f)$, where

- Q – a finite *set of states*,
- Σ – a finite, nonempty *input alphabet*,
- $R \subseteq Q \times \Sigma^* \times Q$ – a finite set of *rules*:
  - $(p, w, q) \in R$ written as $pw \rightarrow q$,
- $s \in Q$ – the *initial state*,
- $f \in Q$ – the *final state*.

Note these peculiarities:

- The model is non-deterministic;

Finite Automata: Definition

A *generalized finite automaton* (GFA) is a 5-tuple
$M = (Q, \Sigma, R, s, f)$, where

- Q – a finite *set of states*,
- Σ – a finite, nonempty *input alphabet*,
- $R \subseteq Q \times \Sigma^* \times Q$ – a finite set of *rules*:
  - $(p, w, q) \in R$ written as $pw \to q$,
- $s \in Q$ – the *initial state*,
- $f \in Q$ – the *final state*.

Note these peculiarities:

- The model is non-deterministic;
- The production rules allow reading entire strings;

Finite Automata: Definition

A *generalized finite automaton* (GFA) is a 5-tuple $M = (Q, \Sigma, R, s, f)$, where

- Q – a finite *set of states*,
- Σ – a finite, nonempty *input alphabet*,
- $R \subseteq Q \times \Sigma^* \times Q$ – a finite set of *rules*:
  - $(p, w, q) \in R$ written as $pw \to q$,
- $s \in Q$ – the *initial state*,
- $f \in Q$ – the *final state*.

Note these peculiarities:

- The model is non-deterministic;
- The production rules allow reading entire strings;
- There is only a single final state.

Transition Graph of a GFA

An edge-labelled directed graph $G = (V, E, W)$, where:

- $V = Q$,

Transition Graph of a GFA

An edge-labelled directed graph $G = (V, E, W)$, where:

- $V = Q$,
- $E = \{(u, v) \in Q \times Q \mid \exists w \in \Sigma^* : (uw \to v) \in R\}$,

Transition Graph of a GFA

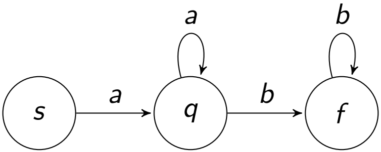
An edge-labelled directed graph $G = (V, E, W)$, where:

- $V = Q$,
- $E = \{(u, v) \in Q \times Q \mid \exists w \in \Sigma^* : (uw \to v) \in R\}$,
- $W : (u, v) \mapsto \{w \in \Sigma^* \mid (uw \to v) \in R\}$.

Transition Graph of a GFA

An edge-labelled directed graph $G = (V, E, W)$, where:

- $V = Q$,
- $E = \{(u, v) \in Q \times Q \mid \exists w \in \Sigma^* : (uw \to v) \in R\}$,
- $W : (u, v) \mapsto \{w \in \Sigma^* \mid (uw \to v) \in R\}$.

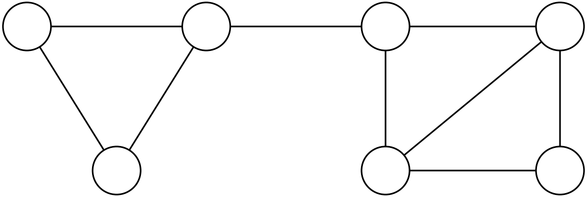


- $V = \{s, q, f\}$,
- $E = \{(s, q), (q, q), (q, f), (f, f)\}$,
  - $W(s, q) = \{a\}$,
  - $W(q, q) = \{a\}$,
  - $W(q, f) = \{b\}$,
  - $W(f, f) = \{b\}$

Bridges and Islands

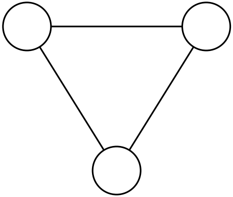
Connected graph

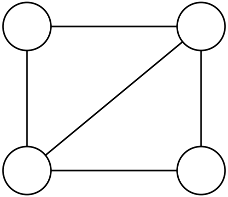
Connected graph: Any two nodes are connected by an undirected path.



Disconnected graph

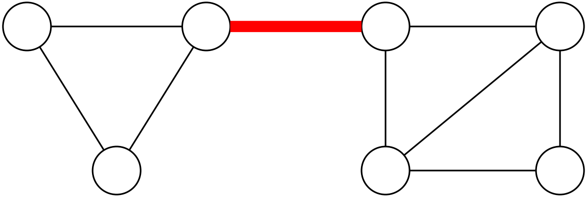
Connected graph: Any two nodes are connected by an undirected path.





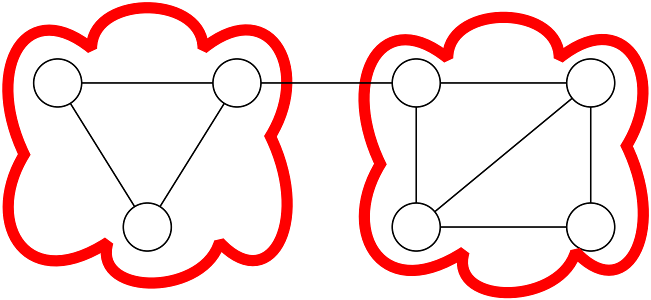
Bridge

Bridge: an edge such that when it is removed, the graph is no longer connected.



Island

A *bridgeless island* = a maximal bridgeless connected component



Every node and edge is either a bridge or contained in exactly one bridgeless island.

Islands in Automata

Islands in Automata: The Structure

- A state is *useful* if it occurs on some path from s to f;
- Otherwise, it is *useless*;

Islands in Automata: The Structure

- A state is *useful* if it occurs on some path from s to f;
- Otherwise, it is *useless*;
- Assuming no useless states, the islands will always be aranged linearly:

$$I_1 \longrightarrow I_2 \longrightarrow \cdots \longrightarrow I_n$$

Islands in Automata: The Structure

- A state is *useful* if it occurs on some path from s to f;
- Otherwise, it is *useless*;
- Assuming no useless states, the islands will always be aranged linearly:

$$I_1 \longrightarrow I_2 \longrightarrow \cdots \longrightarrow I_n$$

- Sketch of Proof:
  1. Think of an "island graph" – the nodes are islands, the edges are bridges;
  2. This graph is necessarily a tree;
  3. There must be exactly one path between I_s and I_f;
  4. All states are useful, so all islands must lie on this path.

Islands in Automata: Number Variability

- For any integers m, n, a GFA with m bridges can be converted into an equivalent GFA with n bridges;

Islands in Automata: Number Variability

- For any integers m, n, a GFA with m bridges can be converted into an equivalent GFA with n bridges;
- Idea of proof:
  - Redundant states and transitions can merge existing islands and create new ones.

Islands in Automata: An abstraction

- a *k-bridge island* in G:
  - a maximal connected subgraph of G containing exactly k bridges
  - the merging of $k+1$ bridgeless islands and their connecting bridges

Islands in Automata: An abstraction

- a *k-bridge island* in G:
  - a maximal connected subgraph of G containing exactly k bridges
  - the merging of $k+1$ bridgeless islands and their connecting bridges
- We can explicitly specify which islands we want:
  a) Explicitly describe which states form which islands,

Islands in Automata: An abstraction

- a *k-bridge island* in G:
  - a maximal connected subgraph of G containing exactly k bridges
  - the merging of $k+1$ bridgeless islands and their connecting bridges
- We can explicitly specify which islands we want:
  a) Explicitly describe which states form which islands,
  b) Select the bridges that will actually divide islands;

Islands in Automata: An abstraction

- a *k-bridge island* in G:
  - a maximal connected subgraph of G containing exactly k bridges
  - the merging of $k + 1$ bridgeless islands and their connecting bridges
- We can explicitly specify which islands we want:
  a) Explicitly describe which states form which islands,
  b) Select the bridges that will actually divide islands;
     - $\binom{b}{n-1}$ ways to select n islands in a GFA with b bridges.

n-Island GFA

- An n-island GFA (n-IGFA) is:
  - A GFA M (with at least $n-1$ bridges),
  - Along with a set Γ of selected bridges;

n-Island GFA

- An n-island GFA (n-IGFA) is:
  - A GFA M (with at least $n-1$ bridges),
  - Along with a set Γ of selected bridges;
- Let $\mathcal{L}(GFA_n)$ denote the class of languages accepted by n-IGFA;
- $\mathcal{L}(\mathrm{GFA}_n) = \mathrm{REG}$ for any $n \geq 1$;

n-Island GFA

- An n-island GFA (n-IGFA) is:
  - A GFA M (with at least $n-1$ bridges),
  - Along with a set Γ of selected bridges;
- Let $\mathcal{L}(GFA_n)$ denote the class of languages accepted by n-IGFA;
- $\mathcal{L}(\text{GFA}_n) = \text{REG}$ for any $n \geq 1$;
- Sketch of proof:
  1. n-IGFA are special cases of GFA;
  2. A GFA along with $\Gamma = \emptyset$ is a 1-IGFA;
  3. An n-IGFA can be transformed into an equivalent m-IGFA.

Even Computations

Even Computations

- An n-IGFA accepts the same language as the underlying GFA...

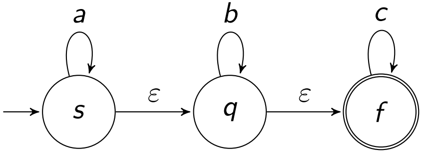
Even Computations

- An n-IGFA accepts the same language as the underlying GFA...
- ...unless we add an additional constraint to their computation:

Even Computations

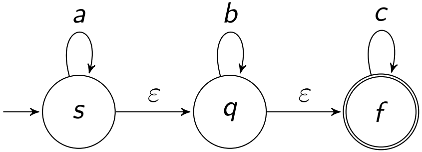
- An n-IGFA accepts the same language as the underlying GFA. . .
- . . . unless we add an additional constraint to their computation:
- A computation of an n-IGFA is *even* if **the same number of steps is taken in each island**.

Even Computations: Example (1/2)



- Note: ε denotes the *empty string*;

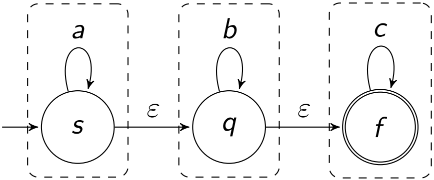
Even Computations: Example (1/2)



- Note: ε denotes the *empty string*;
- The language accepted by this automaton is

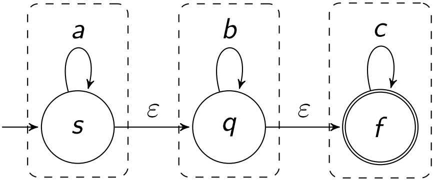
$$L(M) = \{a^i b^j c^k \mid i, j, k \geq 0\}.$$

Even Computations: Example (2/2)



- Note: ε denotes the *empty string*;
- Let us consider islands defined by the bridges $\Gamma = \{(s, q), (q, f)\}$:

Even Computations: Example (2/2)



- Note: ε denotes the *empty string*;
- Let us consider islands defined by the bridges $\Gamma = \{(s, q), (q, f)\}$:
- The language accepted by this automaton *by even computations with regard to* Γ is

$$L_e(M, \Gamma) = \{a^n b^n c^n \mid n \geq 0\};$$

- $L_e(M, \Gamma) \in \text{CS} \setminus \text{CF}$.

Accepting Power

Accepting Power: n-PRLG

- Let $\mathcal{L}_e(\mathrm{GFA}_n)$ denote the class of languages accepted by n-IGFA by even computations;

Accepting Power: n-PRLG

- Let $\mathcal{L}_e(\mathrm{GFA}_n)$ denote the class of languages accepted by n-IGFA by even computations;
- Equivalent power to *n-parallel right linear grammars* (n-PRLG):
  - (N, Σ, P, S);
  - P contains rules of the forms:
    a) $S \rightarrow x$, where $x \in \Sigma^*$,
    b) $S \rightarrow A_1 \cdots A_n$, where $A_i \in N$,
    c) $A \rightarrow xB$, where $A, B \in N \setminus \{S\}, x \in \Sigma^*$,
    d) $A \rightarrow x$, where $A \in N \setminus \{S\}, x \in \Sigma^*$;
  - All nonterminals rewritten at once;

Accepting Power: n-PRLG

- Let $\mathcal{L}_e(\mathrm{GFA}_n)$ denote the class of languages accepted by n-IGFA by even computations;
- Equivalent power to *n-parallel right linear grammars* (n-PRLG):
  - (N, Σ, P, S);
  - P contains rules of the forms:
    a) $S \to x$, where $x \in \Sigma^*$,
    b) $S \to A_1 \cdots A_n$, where $A_i \in N$,
    c) $A \to xB$, where $A, B \in N \setminus \{S\}, x \in \Sigma^*$,
    d) $A \to x$, where $A \in N \setminus \{S\}, x \in \Sigma^*$;
  - All nonterminals rewritten at once;
- We denote the class of languages generated by n-PRLGs by PRL_n.

n-PRLG: Example

- $G = (\{S, A, B\}, \{a, b\}, P, S)$,
- P contains the following rules:
  - $S \rightarrow AB \mid BA$,
  - $A \rightarrow aA \mid \varepsilon$,
  - $B \rightarrow bB \mid \varepsilon$;

n-PRLG: Example

- $G = (\{S, A, B\}, \{a, b\}, P, S)$,
- P contains the following rules:
  - $S \to AB \mid BA$,
  - $A \to aA \mid \varepsilon$,
  - $B \to bB \mid \varepsilon$;
- G is a 2-PRLG;

n-PRLG: Example

- $G = (\{S, A, B\}, \{a, b\}, P, S)$,
- P contains the following rules:
  - $S \to AB \mid BA$,
  - $A \to aA \mid \varepsilon$,
  - $B \to bB \mid \varepsilon$;
- G is a 2-PRLG;
- $L(G) = \{a^n b^n, b^n a^n \mid n \geq 0\}$

Proof: $\mathcal{L}_e(\mathrm{GFA}_n) \subseteq \mathrm{PRL}_n$

- Let $M = (Q, \Sigma, R, s, f)$ be an n-IGFA along with a set Γ of bridges;

Proof: $\mathcal{L}_e(\text{GFA}_n) \subseteq \text{PRL}_n$

- Let $M = (Q, \Sigma, R, s, f)$ be an n-IGFA along with a set Γ of bridges;
- For the j-th island, let:
  - Q_j denote its set of states,
  - s_j its *entry state*, and
  - f_j its *exit state*;

Proof: $\mathcal{L}_e(\text{GFA}_n) \subseteq \text{PRL}_n$

- Let $M = (Q, \Sigma, R, s, f)$ be an n-IGFA along with a set Γ of bridges;
- For the j-th island, let:
  - Q_j denote its set of states,
  - s_j its *entry state*, and
  - f_j its *exit state*;
- Construct the grammar $G = (Q \cup \{S\}, \Sigma, P, S)$ where $S \notin Q \cup \Sigma$ and $P = P_s \cup P_i \cup P_f$ where:

Proof: $\mathcal{L}_e(\mathrm{GFA}_n) \subseteq \mathrm{PRL}_n$

- Let $M = (Q, \Sigma, R, s, f)$ be an n-IGFA along with a set Γ of bridges;
- For the j-th island, let:
  - Q_j denote its set of states,
  - s_j its *entry state*, and
  - f_j its *exit state*;
- Construct the grammar $G = (Q \cup \{S\}, \Sigma, P, S)$ where $S \notin Q \cup \Sigma$ and $P = P_s \cup P_i \cup P_f$ where:
  - $P_s = \{S \to s_1 \cdots s_n\}$,

Proof: $\mathcal{L}_e(\mathrm{GFA}_n) \subseteq \mathrm{PRL}_n$

- Let $M = (Q, \Sigma, R, s, f)$ be an n-IGFA along with a set Γ of bridges;
- For the j-th island, let:
  - Q_j denote its set of states,
  - s_j its *entry state*, and
  - f_j its *exit state*;
- Construct the grammar $G = (Q \cup \{S\}, \Sigma, P, S)$ where $S \notin Q \cup \Sigma$ and $P = P_s \cup P_i \cup P_f$ where:
  - $P_s = \{S \to s_1 \cdots s_n\}$,
  - $P_i = \{p \to xq \mid (px \to q) \in R \text{ and } p, q \in Q_j \text{ for some } 1 \leq j \leq n\}$,

Proof: $\mathcal{L}_e(\mathrm{GFA}_n) \subseteq \mathrm{PRL}_n$

- Let $M = (Q, \Sigma, R, s, f)$ be an n-IGFA along with a set Γ of bridges;
- For the j-th island, let:
  - Q_j denote its set of states,
  - s_j its *entry state*, and
  - f_j its *exit state*;
- Construct the grammar $G = (Q \cup \{S\}, \Sigma, P, S)$ where $S \notin Q \cup \Sigma$ and $P = P_s \cup P_i \cup P_f$ where:
  - $P_s = \{S \rightarrow s_1 \cdots s_n\}$,
  - $P_i = \{p \rightarrow xq \mid (px \rightarrow q) \in R \text{ and } p, q \in Q_j \text{ for some } 1 \leq j \leq n\}$,
  - $P_f = \{f_j \rightarrow x | (f_j x \rightarrow s_{j+1}) \in R \text{ for some } 1 \leq j < n\} \cup \{f \rightarrow \varepsilon\}$;

Proof: $\mathcal{L}_e(\mathrm{GFA}_n) \subseteq \mathrm{PRL}_n$

- Let $M = (Q, \Sigma, R, s, f)$ be an n-IGFA along with a set Γ of bridges;
- For the j-th island, let:
  - Q_j denote its set of states,
  - s_j its *entry state*, and
  - f_j its *exit state*;
- Construct the grammar $G = (Q \cup \{S\}, \Sigma, P, S)$ where $S \notin Q \cup \Sigma$ and $P = P_s \cup P_i \cup P_f$ where:
  - $P_s = \{S \to s_1 \cdots s_n\}$,
  - $P_i = \{p \to xq \mid (px \to q) \in R \text{ and } p, q \in Q_j \text{ for some } 1 \leq j \leq n\}$,
  - $P_f = \{f_j \to x | (f_j x \to s_{j+1}) \in R \text{ for some } 1 \leq j < n\} \cup \{f \to \varepsilon\}$;
- $L(G) = L_e(M, \Gamma)$.

Proof: $\mathrm{PRL}_n \subseteq \mathcal{L}_e(\mathrm{GFA}_n)$ – The Problem

- The converse direction is considerably harder to prove; consider the grammar from before:

Proof: $\mathrm{PRL}_n \subseteq \mathcal{L}_e(\mathrm{GFA}_n)$ – The Problem

- The converse direction is considerably harder to prove; consider the grammar from before:
- $G = (\{S, A, B\}, \{a, b\}, P, S)$,
- P contains the following rules:
  - $S \rightarrow AB \mid BA$
  - $A \rightarrow aA \mid \varepsilon$
  - $B \rightarrow bB \mid \varepsilon$
- $L(G) = \{a^n b^n, b^n a^n \mid n \geq 0\}$

Proof: $\mathrm{PRL}_n \subseteq \mathcal{L}_e(\mathrm{GFA}_n)$ – The Problem

- The converse direction is considerably harder to prove; consider the grammar from before:
- $G = (\{S, A, B\}, \{a, b\}, P, S)$,
- P contains the following rules:
  - $S \to AB \mid BA$
  - $A \to aA \mid \varepsilon$
  - $B \to bB \mid \varepsilon$
- $L(G) = \{a^n b^n, b^n a^n \mid n \geq 0\}$
- We can easily form components to accept substrings of the forms a^n or b^n in each island, and even computations will ensure equal length;

Proof: $\mathrm{PRL}_n \subseteq \mathcal{L}_e(\mathrm{GFA}_n)$ – The Problem

- The converse direction is considerably harder to prove; consider the grammar from before:
- $G = (\{S, A, B\}, \{a, b\}, P, S)$,
- P contains the following rules:
  - $S \rightarrow AB \mid BA$
  - $A \rightarrow aA \mid \varepsilon$
  - $B \rightarrow bB \mid \varepsilon$
- $L(G) = \{a^n b^n, b^n a^n \mid n \geq 0\}$
- We can easily form components to accept substrings of the forms a^n or b^n in each island, and even computations will ensure equal length;
- How do we ensure that the a^n component in the first island will only work with the b^n component in the second island and vice versa?

Proof: $\mathrm{PRL}_n \subseteq \mathcal{L}_e(\mathrm{GFA}_n)$ – The Problem

- The converse direction is considerably harder to prove; consider the grammar from before:
- $G = (\{S, A, B\}, \{a, b\}, P, S)$,
- P contains the following rules:
  - $S \to AB \mid BA$
  - $A \to aA \mid \varepsilon$
  - $B \to bB \mid \varepsilon$
- $L(G) = \{a^n b^n, b^n a^n \mid n \geq 0\}$
- We can easily form components to accept substrings of the forms a^n or b^n in each island, and even computations will ensure equal length;
- How do we ensure that the a^n component in the first island will only work with the b^n component in the second island and vice versa?
- In general, how do we deal with different initial rules of an n-PRLG?

Proof: $\mathrm{PRL}_n \subseteq \mathcal{L}_e(\mathrm{GFA}_n)$ – An Example Solution

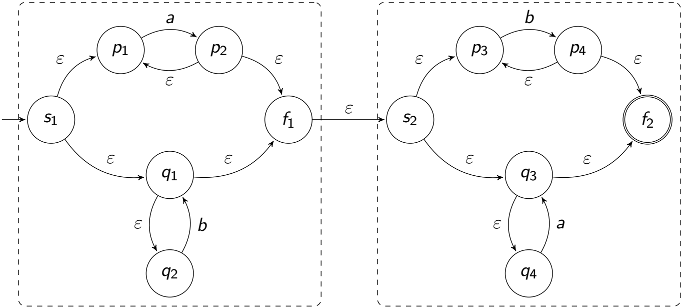
- The trick is to encode the form of the accepted string in the number of steps in each island;

Proof: $\mathrm{PRL}_n \subseteq \mathcal{L}_e(\mathrm{GFA}_n)$ – An Example Solution

- The trick is to encode the form of the accepted string in the number of steps in each island;
- For example, an odd number for the form $a^n b^n$, and an even number for the form $b^n a^n$:

Proof: $\mathrm{PRL}_n \subseteq \mathcal{L}_e(\mathrm{GFA}_n)$ – An Example Solution

- The trick is to encode the form of the accepted string in the number of steps in each island;
- For example, an odd number for the form $a^n b^n$, and an even number for the form $b^n a^n$:



Proof: $\mathrm{PRL}_n \subseteq \mathcal{L}_e(\mathrm{GFA}_n)$ – General Case (Sketch, 1/2)

- Let m be the number of starting production rules of the input grammar of the form $S \to A_1 \cdots A_n$;

Proof: $\mathrm{PRL}_n \subseteq \mathcal{L}_e(\mathrm{GFA}_n)$ – General Case (Sketch, 1/2)

- Let m be the number of starting production rules of the input grammar of the form $S \to A_1 \cdots A_n$;
- Associate each of these starting rules with a remainder modulo $m+1$ with the remainder 0 reserved for starting rules of the form $S \to x$, $x \in \Sigma^*$;

Proof: $\mathrm{PRL}_n \subseteq \mathcal{L}_e(\mathrm{GFA}_n)$ – General Case (Sketch, 1/2)

- Let m be the number of starting production rules of the input grammar of the form $S \to A_1 \cdots A_n$;
- Associate each of these starting rules with a remainder modulo $m+1$ with the remainder 0 reserved for starting rules of the form $S \to x$, $x \in \Sigma^*$;
- Each island i (for $1 \leq i \leq n$) will contain the following kinds of states:

Proof: $\mathrm{PRL}_n \subseteq \mathcal{L}_e(\mathrm{GFA}_n)$ – General Case (Sketch, 1/2)

- Let m be the number of starting production rules of the input grammar of the form $S \to A_1 \cdots A_n$;
- Associate each of these starting rules with a remainder modulo $m + 1$ with the remainder 0 reserved for starting rules of the form $S \to x$, $x \in \Sigma^*$;
- Each island i (for $1 \leq i \leq n$) will contain the following kinds of states:
  - Entry state s_i and exit state f_i,

Proof: $\mathrm{PRL}_n \subseteq \mathcal{L}_e(\mathrm{GFA}_n)$ – General Case (Sketch, 1/2)

- Let m be the number of starting production rules of the input grammar of the form $S \to A_1 \cdots A_n$;
- Associate each of these starting rules with a remainder modulo $m+1$ with the remainder 0 reserved for starting rules of the form $S \to x$, $x \in \Sigma^*$;
- Each island i (for $1 \leq i \leq n$) will contain the following kinds of states:
  - Entry state s_i and exit state f_i,
  - States $\langle i, j \rangle$ where $1 \leq j \leq m$ for the initial generation of remainder j,

Proof: $\mathrm{PRL}_n \subseteq \mathcal{L}_e(\mathrm{GFA}_n)$ – General Case (Sketch, 1/2)

- Let m be the number of starting production rules of the input grammar of the form $S \to A_1 \cdots A_n$;
- Associate each of these starting rules with a remainder modulo $m+1$ with the remainder 0 reserved for starting rules of the form $S \to x$, $x \in \Sigma^*$;
- Each island i (for $1 \leq i \leq n$) will contain the following kinds of states:
  - Entry state s_i and exit state f_i,
  - States $\langle i, j \rangle$ where $1 \leq j \leq m$ for the initial generation of remainder j,
  - States $\langle A, i, j \rangle$ where $1 \leq j \leq m$ and $A \in N \setminus \{S\}$ to represent nonterminal A simulated with predetermined remainder j,

Proof: $\mathrm{PRL}_n \subseteq \mathcal{L}_e(\mathrm{GFA}_n)$ – General Case (Sketch, 1/2)

- Let m be the number of starting production rules of the input grammar of the form $S \to A_1 \cdots A_n$;
- Associate each of these starting rules with a remainder modulo $m+1$ with the remainder 0 reserved for starting rules of the form $S \to x$, $x \in \Sigma^*$;
- Each island i (for $1 \leq i \leq n$) will contain the following kinds of states:
  - Entry state s_i and exit state f_i,
  - States $\langle i, j \rangle$ where $1 \leq j \leq m$ for the initial generation of remainder j,
  - States $\langle A, i, j \rangle$ where $1 \leq j \leq m$ and $A \in N \setminus \{S\}$ to represent nonterminal A simulated with predetermined remainder j,
  - States $\langle i, j, k, B \rangle$ where $1 \leq j, k \leq m$ and $B \in (N \setminus \{S\}) \cup \{\varepsilon\}$, which use k as a counter to drag out the rewriting of A to B to $m+1$ moves.

Proof: $\mathrm{PRL}_n \subseteq \mathcal{L}_e(\mathrm{GFA}_n)$ – General Case (Sketch, 2/2)

- The automaton constructed will contain the following kinds of rules:

Proof: $\mathrm{PRL}_n \subseteq \mathcal{L}_e(\mathrm{GFA}_n)$ – General Case (Sketch, 2/2)

- The automaton constructed will contain the following kinds of rules:
  - Rules to generate a remainder (at the start of each island):
    - $s_i \to \langle i, 1\rangle$, $\langle i, j\rangle \to \langle i, j+1\rangle$;

Proof: $\mathrm{PRL}_n \subseteq \mathcal{L}_e(\mathrm{GFA}_n)$ – General Case (Sketch, 2/2)

- The automaton constructed will contain the following kinds of rules:
  - Rules to generate a remainder (at the start of each island):
    - $s_i \to \langle i, 1\rangle$, $\langle i, j\rangle \to \langle i, j+1\rangle$;
  - Rules to pair a given remainder with the corresponding computation within each island,
    - $\langle 1, m\rangle \to f_1$, $\langle i, m\rangle \to f_i$ for $S \to x \in P$,
    - $s_i \to \langle A_{i1}, i, 1\rangle$, $\langle i, j-1\rangle \to \langle A_{ij}, i, j\rangle$ for $p_j : S \to A_{1j} \cdots A_{nj} \in P$;

Proof: $\mathrm{PRL}_n \subseteq \mathcal{L}_e(\mathrm{GFA}_n)$ – General Case (Sketch, 2/2)

- The automaton constructed will contain the following kinds of rules:
  - Rules to generate a remainder (at the start of each island):
    - $s_i \to \langle i, 1\rangle$, $\langle i, j\rangle \to \langle i, j+1\rangle$;
  - Rules to pair a given remainder with the corresponding computation within each island,
    - $\langle 1, m\rangle \to f_1$, $\langle i, m\rangle \to f_i$ for $S \to x \in P$,
    - $s_i \to \langle A_{i1}, i, 1\rangle$, $\langle i, j-1\rangle \to \langle A_{ij}, i, j\rangle$ for $p_j : S \to A_{1j} \cdots A_{nj} \in P$;
  - Rules to simulate grammar rules of the form $A \to xB$ and $A \to x$, $A, B \in N$, $x \in \Sigma^*$, along with ε-rules ensuring that each rule is simulated in exactly $m+1$ steps,
    - $\langle A, i, j\rangle x \to \langle i, j, 1, B\rangle$ for $A \to xB \in P$,
    - $\langle i, j, k, B\rangle \to \langle i, j, k+1, B\rangle$, $\langle i, j, m, B\rangle \to \langle B, i, j\rangle$, $\langle i, j, m, \varepsilon\rangle \to f_i$;

Proof: $\mathrm{PRL}_n \subseteq \mathcal{L}_e(\mathrm{GFA}_n)$ – General Case (Sketch, 2/2)

- The automaton constructed will contain the following kinds of rules:
  - Rules to generate a remainder (at the start of each island):
    - $s_i \to \langle i, 1\rangle$, $\langle i, j\rangle \to \langle i, j+1\rangle$;
  - Rules to pair a given remainder with the corresponding computation within each island,
    - $\langle 1, m\rangle \to f_1$, $\langle i, m\rangle \to f_i$ for $S \to x \in P$,
    - $s_i \to \langle A_{i1}, i, 1\rangle$, $\langle i, j-1\rangle \to \langle A_{ij}, i, j\rangle$ for $p_j : S \to A_{1j} \cdots A_{nj} \in P$;
  - Rules to simulate grammar rules of the form $A \to xB$ and $A \to x$, $A, B \in N$, $x \in \Sigma^*$, along with ε-rules ensuring that each rule is simulated in exactly $m+1$ steps,
    - $\langle A, i, j\rangle x \to \langle i, j, 1, B\rangle$ for $A \to xB \in P$,
    - $\langle i, j, k, B\rangle \to \langle i, j, k+1, B\rangle$, $\langle i, j, m, B\rangle \to \langle B, i, j\rangle$, $\langle i, j, m, \varepsilon\rangle \to f_i$;
  - Bridge rules:
    - $f_i \to s_{i+1}$.

Corollary: $\mathrm{PRL}_n = \mathcal{L}_e(\mathrm{GFA}_n)$

- $\mathrm{PRL}_n = \mathcal{L}_e(\mathrm{GFA}_n)$
- Proof: See previous slides

Accepting Power

- The following is known about the accepting power of n-PRLGs:

Accepting Power

- The following is known about the accepting power of n-PRLGs:
- $\mathrm{REG} = \mathrm{PRL}_1 \subset \mathrm{PRL}_k \subset \mathrm{PRL}_{k+1} \subset \mathrm{CS}$ for any $k > 1$;
- $\mathrm{PRL}_2 \subset \mathrm{CF}$;
- $\mathrm{PRL}_n \not\subseteq \mathrm{CF}$, $\mathrm{CF} \not\subseteq \mathrm{PRL}_n$, $n \geq 3$;

Accepting Power

- The following is known about the accepting power of *n*-PRLGs:
- $\mathrm{REG} = \mathrm{PRL}_1 \subset \mathrm{PRL}_k \subset \mathrm{PRL}_{k+1} \subset \mathrm{CS}$ for any $k > 1$;
- $\mathrm{PRL}_2 \subset \mathrm{CF}$;
- $\mathrm{PRL}_n \not\subseteq \mathrm{CF}$, $\mathrm{CF} \not\subseteq \mathrm{PRL}_n$, $n \geq 3$;
- Finally, $\mathrm{PRL}_n = \mathcal{L}_e(\mathrm{GFA}_n)$ for all $n \geq 1$.

Accepting Power: Summary

- $\mathcal{L}_e(\mathrm{GFA}_n)$ equivalent to languages generated by n-PRLGs:
  - An infinite hierarchy between REG and CS;
  - For $n \geq 3$ incomparable with CF.
- For compactness, EI_n will denote $\mathcal{L}_e(\mathrm{GFA}_n)$ in the following diagram: