Group Input Machine

Ruslans Tarasovs   Rūsiņš Freivalds

University of Latvia

SOFSEM'2009

Introduce a stronger computation model

- Previous models and history.
- Our approach: Group Input Machine.
- Examples and results.
- Technical details.

Problem: Finite state machines are too primitive

$$(\Sigma, S, s_0, \delta, F)$$

- Actions are deterministic.
- Finite memory (number of states).
- Actions on input are limited.
- Input structure is too simple.

Solutions: Finite state machines are too primitive

Actions are not deterministic

- Non-detereministic automata.
- Probabilistic automata.

Infinite memory

- Pushdown automata.
- Turing machines.

Extended action set

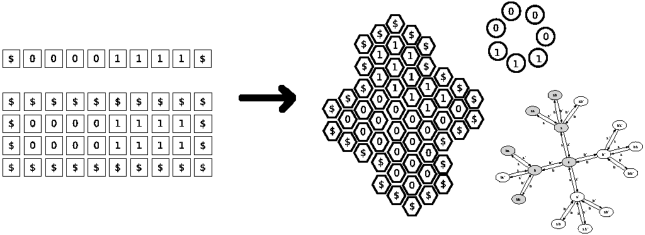
- Two-way automata.

Flexible input structure

- Storage modification machines.

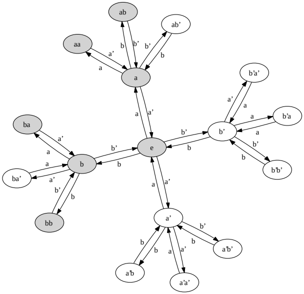
The Idea

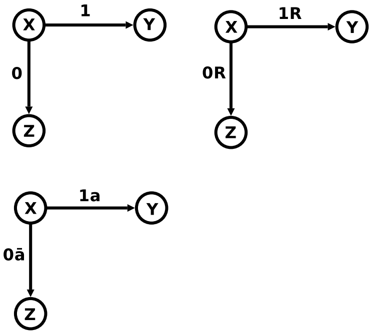
- Replace linear tapes by structured input area.



Group Input Machine

- Group elements = data cells.
- Generating set of the group = movements.
- Special \$ symbol = movement restriction.





Group Input Machine: Examples

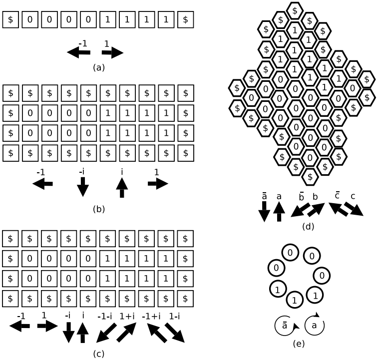
(a) $G = (\mathbf{Z}, +),\ D = \{1\}.$

(b) $G = (\mathbf{C}, +),\ D = \{1, i\}.$

(c) $G = (\mathbf{C}, +),$
$D = \{1, i, 1 + i, 1 - i\}.$

(d) $D = \{a, b, c\}.$

(e) $G =< \{a\} >, a^n = a.$



Group Input Machine: Examples

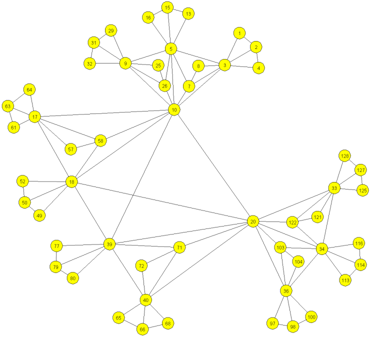


Figure: Braid group

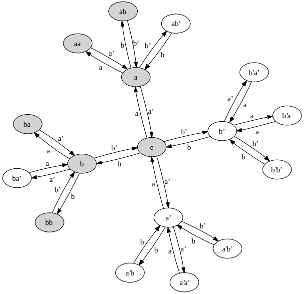


Figure: Free group

Definition

Word

Let G be a group, A is a finite set, and w is a function $G \to A$. Then we say that w is the word over group G and A is the word alphabet.

Automaton

Finite deterministic group automaton (FDGA) is a halting automaton with transition function $f : Q_0 \times A \to Q \times (D \cup D^{-1})$, where $D^{-1} = \{d | d^{-1} \in D\}$.

Definition

Configuration

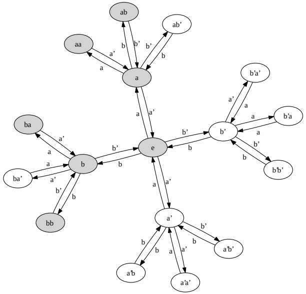
Let w be a word over group $(G, \bullet)$, and M is an arbitrary FDGA. Let define computation of M on w as a tuple $(K, \xrightarrow{t})$, where K is set of configurations and $\xrightarrow{t}$ is a transition relation. Configuration is a tuple (q, c), where $q \in Q$ and $c \in G$. We say that configuration is terminal if $q \in \{q_a, q_d\}$. We define that $(q, c) \xrightarrow{t} (q', c')$ if $c' = c \bullet d$ and $f(q, w(c)) = (q', d)$, where $d \in D \cup D^{-1}$.

Execution

Let say that $k_0, k_1, \ldots, k_n$ is execution of M on w if $\forall i : k_i \xrightarrow{t} k_{i+1}$, $k_0 = (q_0, e)$, where e is a neutral element of G, and k_n is a terminal configuration.

Examples: Free Group

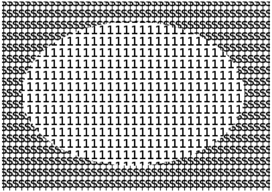
- Let say that G is a free group if there exists such $S \subseteq G$ that every element $x \in G$ could be written as $s_1 s_2 \ldots s_n$ in one and only one way, where $s_i \in S \cup S^{-1}$ and $\forall i : s_i \neq s_{i+1}^{-1}$.

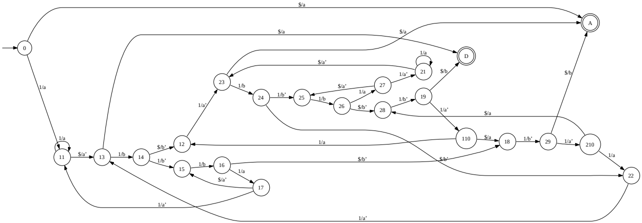


- Simple structure: no additional dependencies.

Free Group: 1-based languages

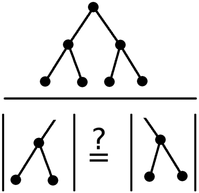
- (L_1) How to find if area size is even?
- Follow wall, remember state.
- Deterministic algorithm.





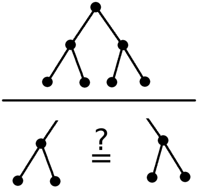
Free Group: 1-based languages

- (L_2) How to find if root tree branches have the same size?
- Deterministic algorithm is impossible.
- Does not have enough memory to remember and compare sizes.
- Probabilistic algorithm exists.
- Use probabilistic nature of automata as additional memory (R.Freivalds, 1981).



Free Group: 1-based languages

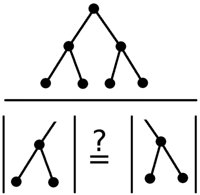
- (L_3) How to find if all root tree branches are exactly the same?
- Both deterministic and probabilistic algorithms are not possible.
- Uses The Markov Chain Tree theorem (F.T.Leighton, R.L.Rivest, 1983).



Free Group: 1-based languages, details

Problem

- (L_2) How to find if root tree branches have the same size?
- Deterministic algorithm is impossible.



Free Group: 1-based languages, details

Proof

- Each subtree - one fixed size square result matrix. Each (row, column) pair coresponds to (input state, output state). For automaton with n states there will be $n \times n$ matrix.
- Such matrix fully define everything particular input subtree does to the state of automaton.
- Get contradiction on large enough branches.

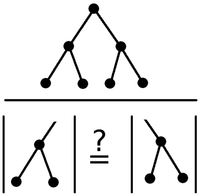
$$w_0(x) = \begin{cases} 1, & \exists i, k \in \mathbf{N}, k \leq n! + 2 : d_i^k = x, \\ \$, & \text{otherwise.} \end{cases}$$

$$w_1(z) = \begin{cases} w_0(z), & z = d_i^k, \text{ where } d_i \neq d, \\ 1, & \exists i, k \in \mathbf{N}, k \leq n! + 2 - (|y| - |x|) : d^k = x, \\ \$, & \text{otherwise.} \end{cases}$$

Free Group: 1-based languages, details

Problem

- (L_2) How to find if root tree branches have the same size?
- Probabilistic algorithm exists.



Free Group: 1-based languages, details

Solution

1. Let define $D = \{d_1, d_2, ..., d_n\}$.
2. For each i from 1 to $n-1$ repeat:
   1. $k_1 := 0; k_2 := 0;$
   2. Repeat t times:
      1. Walk around d_i branch. On each visited element break with probability $1-c$.
      2. Walk around d_{i+1} branch. On each visited element break with probability $1-c$.
      3. If d_i branch was visited fully then $k_1 := k_1 + 1$.
      4. If d_{i+1} branch was visited fully then $k_2 := k_2 + 1$.
      5. If both were not visited fully, repeat iteration.
   3. If $k_1 \neq k_2$ then decline the word.
3. If all iterations finished accept the word.

Conclusion

Summary

- Group Input Machine definition.
- Working examples of the machine.
- Results related to free groups.

Future

- More resuts on other groups, more generic results.
- Relationship between group properties and algorithm properties.

Questions?

Why use groups?

- Groups are simple.

Why use groups?

- Groups are simple.
- Inverse elements.

Why use groups?

- Groups are simple.
- Inverse elements.
- Similar to maze.

Why use groups?

- Groups are simple.
- Inverse elements.
- Similar to maze.