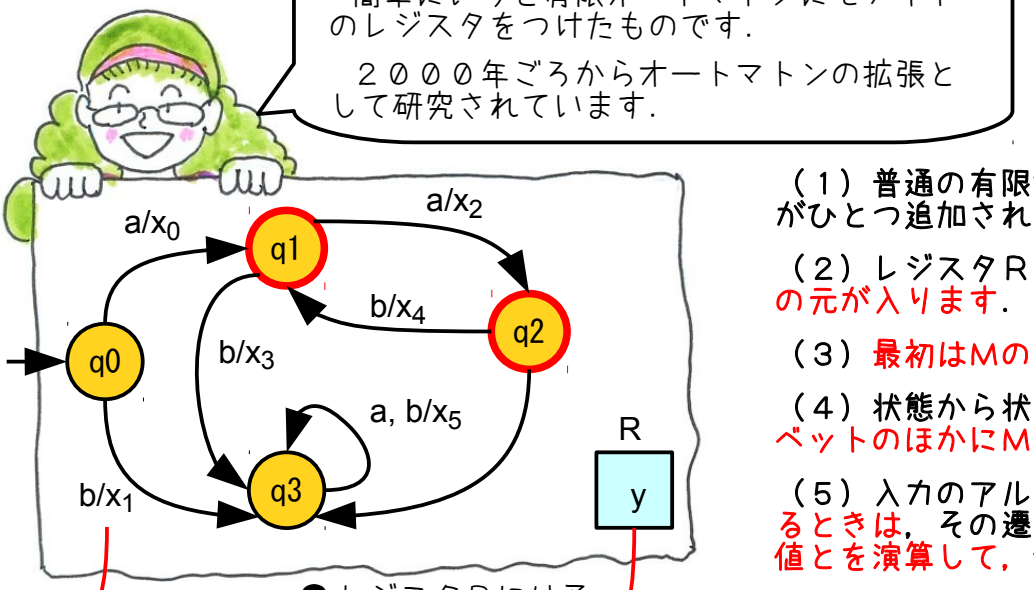
M-Automata (G-Automata)

Akihiko Koga
2016.07.12 初出
07.15 改定



● レジスタＲには予め決められたモノイドＭの元が入ります．

● 遷移は
アルファベット／Ｍの元
という形をしています．
● 「Ｍの元」は遷移するときレジスタＲに掛け込みます．

（１）普通の有限オートマトンにレジスタＲがひとつ追加されています．

（２）レジスタＲには予め決めたモノイドＭの元が入ります．

（３）最初はＭの単位元が入っています．

（４）状態から状態への遷移にはアルファベットのほかにＭの元が指定されています．

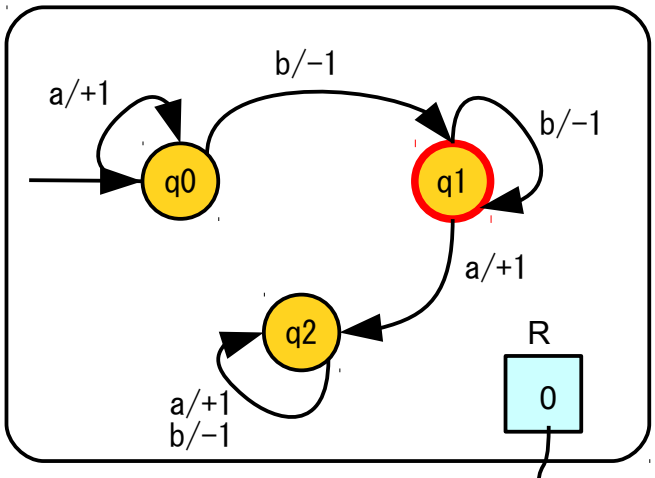
（５）入力のアルファベットに対して遷移するときは，その遷移のＭの元とレジスタＲの値とを演算して，新しいＲの値とします．

（６）予め定めた受理状態（太い赤丸の状態）に来たとき，レジスタＲの値が単位元に戻っていれば，その語を受理します．

（７）［重要］Ｒには演算結果をいれるだけで，Ｒの値を見て遷移することはできません．あくまで，受理状態のときに単位元に戻っているかだけ判断します．

（８）決定論的なオートマトンと非決定論的なオートマトンがあります．非決定論的なものは同じアルファベットに対して複数の遷移があり得ます．アルファベット無しの遷移もあります．この両者で語を受理する能力が違います．

例１



整数Ｚの加法群を取ります．
単位元は０です．

● このようなオートマトンを M-オートマトンといいます．

（特にＭを群Ｇにとったときは，G-オートマトンといいます．）

M-Automaton のいろいろな例

さらに，いくつか具体例を当たっていきましょう．その前に，２つモノイド（一方は群）を定義します．

自由群と polycyclic monoid です．
文章が多くてごめんなさいね．

自由群

記号の集合 $\Sigma_1=\{x, y, \ldots\}$ が与えられたとき，その逆元記号の集合 $\Sigma_2=\{x', y', \ldots\}$ を加えた記号の集合

$$\Sigma=\Sigma_1 \cup \Sigma_2 = \{x, y, \ldots, x', y', \ldots\}$$

の文字の並びの集合 Σ^* に

$$w \cdot w' = w' \cdot w = 1 \text{ for } w \in \Sigma_1$$

の簡約規則を入れたもの．つまり，文字列の中で，$x \cdot x'$ や $x' \cdot x$ の並びは消してよい訳です．

自由群は，Σ_1 の要素の個数 n で構造が決まりますので F の右下に n を書いて表します．例えば，F_2 は２個の記号とそれぞれの逆元からなる自由群です．

polycyclic monoid

polycyclic monoid も自由群と同じように記号の集合 $\Sigma_1=\{x, y, \ldots\}$ が与えられたとき，それを打ち消す記号の集合 $\Sigma_2=\{x', y', \ldots\}$ から作りますが，こちらは右側からだけしか消すことはできません．つまり

$$\Sigma=\Sigma_1 \cup \Sigma_2 = \{x, y, \ldots, x', y', \ldots\}$$

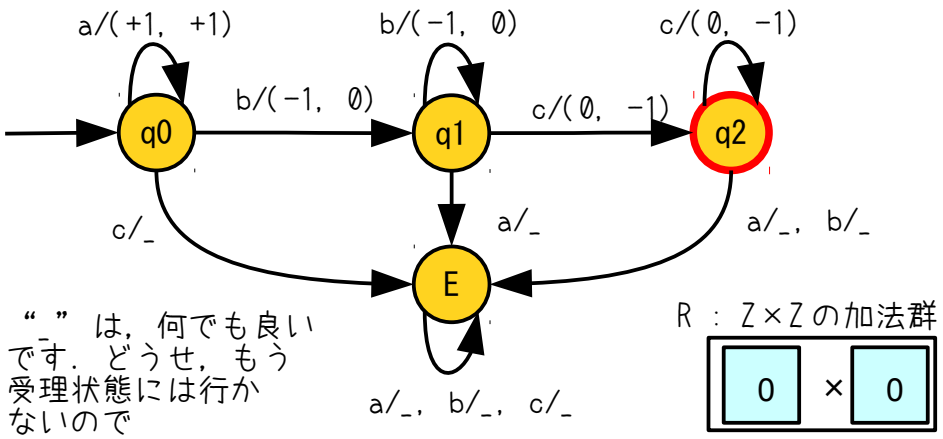
の文字の並びの集合 Σ^* に

$$w \cdot w' = 1 \text{ for } w \in \Sigma_1$$

の簡約規則を入れたものです．

これはスタックをモデル化しているモノイドです．使う記号の個数に従って P_1, P_2, P_3, ... と書きます．２個の記号から作った P_2 を bicyclic monoid と言います．

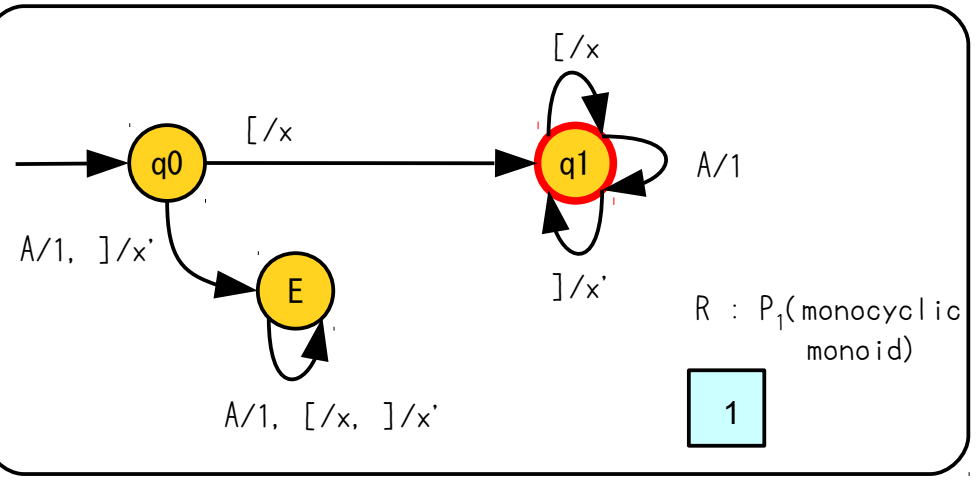
例２ $\{a^n b^n c^n \mid n > 0\}$ を受理



これは文脈依存言語ですね．

(+1, +1) はベクトル (x, y) から (x+1, y+1) を作ることを意味します．あと，加法群ですから単位元は (0, 0) です．

例３ リスト（例えば [A [A] [[A]] A] ）を受理



ここでは，自由群は使いませんでしたね．次ページのまとめの表で使います．
ごめんなさい．

● 実は，このタイプのオートマトンは，昔から繰り返し研究されています．
例えば，１９７５年前後のGreibachの n counter blind automaton はレジスタとして

$$\mathbb{Z} \times \cdots \times \mathbb{Z}$$

を取ったものです（ここでZは整数の加法群です）．

まあ，昔から似た概念はあったわけよ．

● ２０００年以降の動きとして

◆レジスタをモノイドとして一般化した枠組みが確立された

◆群論などへの応用もあり，数学者側からのアプローチが始まった

という動きがあります．

● モノイドMの取り方で色々な言語を受理できるようになります．

モノイド	受理できる言語のクラス	備考
有限モノイド	正規言語 (regular)	―
生成元2個以上の polycyclic monoid	文脈自由言語 (context-free)	決定論的な遷移のものだけでは文脈自由全体を覆えないらしいです．
生成元2個以上の自由群（F_∞, F_n, F_2）		
―	文脈依存言語 (context-sensitive)	丁度，文脈依存言語全体を覆うモノイドはありません（次の行のモノイドは文脈依存言語を含む言語の族を受理）．
$F_2 \times F_2$	再帰的列挙可能（recursively enumerable）	まあ，F_2というスタックもどきでも，2つ使うとチューリングマシンの能力を持ってしまうということでしょうか．

モノイドと受理できる言語のクラスの関係はざっとこんな感じ

●PDF で参照できる文献としては，博士論文の一部の
James Lance Ross: Languages Accepted by Automata with a Counter. 2013, 37p
が，初歩から記述してあって分かりやすかったです．

● ほかに次のようなものがあります．これらもPDFでダウンロード可能です．

J. M. Corson: Extended finite automata and word problems. Internat. J. Algebra Comput., 15(3):455-466, 2005.

M. Kambites: Word problem recognisable by deterministic blind monoid automata. Theoret. Comput. Sci., 362:232-2337, 2006.

Mark Kambites: Formal languages and groups as memory. Communications in algebra, 37(1) (2009) 193-208. arXiv:math/0601061

M. Elder: G-automata, counter languages and the Chomsky hierarchy. In C. Campbell, M. Quick, E. Robertson, and G. Smith, editors, Proceedings of Groups St Andrews 2005,

バイバイ