Complete Pushdown Languages

(Preliminary)
W. L. Ruzzo
circa 5/15/79

The attached notes were never finished, published, refereed, or even very carefully proofread, so please lower your expectations appropriately before reading. Technical notation is probably most similar to

Harrison, Michael A
Introduction to formal language theory / Michael A. Harrison
Reading, Mass. : Addison-Wesley Pub. Co., c1978

The omitted references are (probably) to:

The hardest context-free language
Greibach, S.A. (Univ. California, Los Angeles, CA, USA) **Source:** *SIAM Journal on Computing*, v 2, n 4, Dec. 1973, p 304-10

On the tape complexity of deterministic context-free languages
Sudborough, I.H. (Northwestern Univ., Evanston, IL, USA) **Source:** *Journal of the Association for Computing Machinery*, v 25, n 3, July 1978, p 405-14

R. E. Ladner, R. J. Lipton, L. J. Stockmeyer, Alternating Pushdown Automata. Proceedings of Nineteenth Annual Symposium on Foundations of Computer Science, 1978, 92-106.

I hope you find it useful.

Larry Ruzzo
3/30/2006

5/15/79

COMPLETE PUSHDOWN LANGUAGES

(Preliminary)

W.L. Ruzzo

In this note, we give a simple techinque for constructing languages which are complete for various classes of pushdown automaton languages. In particular, this technique gives a "hardest context-free language" similar to Griebach's [Gr] and a "hardest deterministic cfl" similar to Sudborough's [Su]. Other applications include complete sets for two-way deterministic- and non-deterministic PDA's and log-space auxiliary PDA's. These 3 languages are log-space complete in P by well-known results. A similar construction gives a complete language for alternating - PDA's, which is complete in exponential time [L L S 7 8] .

The basic idea of these constructions is as follows. Let Σ_n = $\{a_1, a_2, .., a_n, \bar{a}_1, ... \bar{a}_n\}$ and let D_n be the Dyck setover Σ_n, i.e., the set generated by the cfg $\{S \rightarrow a_i S \bar{a}_i S | 1 \leq i \leq n\}$. Consider the set of directed graphs with one designated start vertex, one or more designated final vertices, and with all edges labeled by strings in Σ_n^*. The desired complete languages will be the set of (encodings of) such graphs having the property that there is a path from the start vertex to some final vertex such that the concatenation of the edge labels along the path is a word in D_n. Call this set of graphs G_n. The characteristics of the different PDA classes are reflected in different restrictions on the set of graphs considered. For example, 1-way PDA's correspond to G_n^a , the subset of G_n consisting of acyclic graphs with vertices listed in topological order.

For definiteness, we'll discuss the construction for 1-way non-deterministic PDA's (henceforth 1NDPA). It isn't hard to see how a 1NPDA could recognize G_n^a . The PDA guesses which edge to follow at each step, using its stack in the obvious way to process the edge label. The top of the stack is also used temporarily while "following an edge" to record the "name" of the "destination" vertex. Details are left to the reader.

An arbitrary 1NPDA language L may be reduced to G_n^a as follows. Let $M = (Q, \Sigma, \Gamma, \delta, q_o, Z_o, F)$ be a 1NPDA accepting L ; w log M is realtime (Λ-free), and accepts by empty store. For $x = a_1 a_2 \ldots a_\ell \in L$, construct the graph $g_{M,x}$ (see fig. 1) having vertices $Q \times \{1, \ldots, \ell\}$ with edges $(q, i) \xrightarrow{\bar{z}\gamma} (p, i+1)$ iff M has a move from state q with input a_i and top of stack z which pushes γ and enters state q; i.e., $(p, \gamma) \in \delta(q, a_i, z)$. It's easy to show by induction that $(q_o, a_1 \ldots a_i, z_o) \overset{i}{\underset{M}{\vdash}} (p, \Lambda, \gamma)$ iff there is a path in $g_{M,x}$ from $(q_o, 1)$ to $(p, i+1)$ whose label after cancellation is γ.

The set of edges from vertices $\{(q, i) | q \in Q\}$ depends only on a_i, so with a suitable choice of coding, the mapping $x \to g_{M,x}$ can be a homomorphism. The resulting language is very similar to Griebach's [Gr].

Let $G_n^{a,d}$ be the set of acyclic graphs with the property that every edge label leaving a given vertex begins with a "bared" letter $\bar{z}$, and no two begin with the same letter. Then the construction given above can reduce (via homomorphism) every realtime 1DPDA language to $G_n^{a,d}$. Further, we can log-space reduce <u>any</u> 1DPDA language to $G_n^{a,d}$,

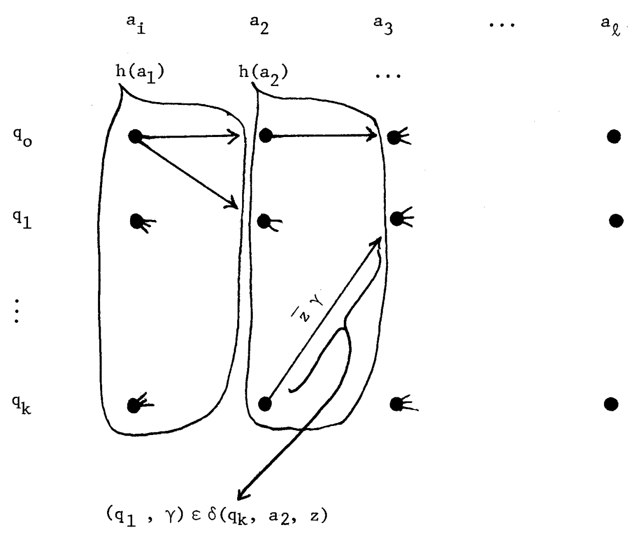
a_i
a_2
a_3
$\cdots$
a_ℓ
$h(a_1)$
$h(a_2)$
$\cdots$
q_o
q_1
$\vdots$
q_k
$z\ \gamma$
$(q_1\ ,\ \gamma)\ \varepsilon\ \delta(q_k,\ a_2,\ z)$

Figure 1

since we know there can't be more than a linear number of consecutive Λ-moves in any accepting computation.

Similarly, we can log-space reduce any polynomial time bounded log-space auxiliary PDA (DPDA) to G_n^a ($G_n^{a,d}$), the major difference being (1) vertices correspond to state and work tape contents and (2) the basic graph is copied a polynomial number of times, with left moves of the input head being represented by edges into the next copy to the right (in order to preserve the topologically sorted requirement.) This gives a simpler proof of Sudborough's chracterization of these languages as $\{L | L \leq_{\log} \text{CFL (DCFL)}\}$.

The results for 2-way PDA's and auxiliary PDA's are similar, but we don't have the "acyclic, topologically sorted" constraint.

We can generalize the above techniques to cover alternating PDA's by treating the graphs as and-or graphs; i.e., the vertices are partitioned into and-nodes and or-nodes. The condition on the set of graphs to be accepted is changed so that if a path reaches an and-node (or-node) then all continuations (one continuation) must lead (recursively) to final vertices with path labels in D_n.

Fixes to "complete PDA" constr

① For $\leq_h$, need to fix Z_0 on stack bottom: ~~[illegible]~~ ~~$(q_0, i) \xrightarrow{\bar{Z_0} \gamma}$~~

~~[illegible]~~ ~~q_0~~

assume q_0 never re-entered, & have q_0 transitions labeled $\xrightarrow{\gamma}$ not $\xrightarrow{\bar{Z_0}\gamma}$

② For $\leq_h$ need to mark "final" vertices: assume accept by empty stack & seperate cols by marker eg $\$$, then define Final ~~vertices~~ to be any after last $\$$.

③ reduce to fixed n in D_n by standard coding of $a_i \to a_1 (a_2)^i$

$\bar{a_i} \to (\bar{a_2})^i \, \bar{a_1}$