A Decision Procedure for the Equivalence of Two dpdas One of Which is Linear

(Extended Abstract)

Yair Itzhaik
Technion, Israel Institute of Technology
and
Amiram Yehudai
Tel-Aviv University

1. Introduction

The equivalence problem for deterministic pushdown automata (dpda) remains open despite extensive research. In recent years, several techniques have been used to obtain decision procedures for the equivalence problem that work for certain deterministic subfamilies. For a survey of these results refer to [HHY, OHI].

In this paper, we show how a technique called parallel stacking, introduced by Valiant [V2], and refined by Beeri [B], may be used to obtain an equivalence test for two dpdas, one of which is 1-turn (sometimes called linear dpda). In [V2, B], the equivalence of two finite-turn dpdas is shown to be decidable. Recently Ukkonen [U] presented a decision procedure, based on alternate stacking, for the equivalence of two proper dpdas, one of which is finite-turn. Proper dpdas are a proper subfamily of the dpdas, although Ukkonen conjectures that every deterministic language may be accepted by some proper dpda. We have recently learnt that Oyamaguchi, Inagaki and Honda have shown that the equivalence problem for two dpdas, one of which is finite-turn, is also decidable. As of this writing, however, we have not seen their paper [OIH].

Our decision procedure will work as follows. Given a dpda M and a 1-turn dpda $\bar{M}$, we first construct a new dpda M'. M' simulates M, but it knows whether or not, on the input being considered, $\bar{M}$ has made its only turn. After $\bar{M}$ made its turn M' continues the simulation without increasing its stack. We will show that M is equivalent to $\bar{M}$ if and only if they are both equivalent to M'. Then we check equivalence of M' versus each of the original machines. It turns out that both these equivalences may be checked using techniques similar to that of [V2], since M' is sufficiently "similar" to both M and $\bar{M}$.

Section 2 lists some of the preliminary definitions needed in

our presentation. In section 3 we discuss the construction of M'. Section 4 reviews parallel stacking [V2, B] and the following two sections describe the two decision procedures for the equivalence of $\bar{M}$ and M' and of M and M' respectively. Finally, in section 7, we conclude with our main result and some comments.

2. Preliminaries

We briefly list the definitions of dpdas and finite-turn dpdas. ε denotes the empty word. A dpda is denoted by a 7-tuple $M = (Q,\Sigma,\Gamma,\delta,s_0,z_0,F)$ where Q, Σ and Γ are respectively finite sets of states, input symbols, and stack symbols, s_0 is the initial state, z_0 in Γ is the initial stack symbol, $F \subseteq Q\times(\Gamma\cup\{\varepsilon\})$ is the set of accepting or final modes and $\delta: Q\times(\Sigma\cup\{\varepsilon\})\times\Gamma \to Q\times\Gamma^*$, the transition function, is a partial function satisfying the determinism condition: if $\delta(s,\varepsilon,A)$ is defined then for all $a \in \Sigma$, $\delta(s,a,A)$ is not defined. If the determinism condition is not required, then M is a pushdown automaton (pda).

If $\delta(s,\pi,A) = (s',y)$, we usually write $(s,A) \overset{\pi}{\to} (s',y)$. This transition rule of M has mode (s,A) and input π. If $\pi = \varepsilon$ the transition rule is called an ε-rule, and (s,A) is an ε-mode. Otherwise (s,A) is a reading-mode. We may assume all modes in F to be reading modes (except those with empty stack). We also assume that if $(s,A) \overset{\varepsilon}{\to} (s',y)$ then $y = \varepsilon$, and if $(s,A) \overset{a}{\to} (s',y)$ then $|y| \leq 2$.

A pair $(s,\omega A)$, s in Q, A in Γ, ω in Γ^* is a configuration with mode (s,A) while (s,ε) is a configuration with mode (s,ε). Configuration $c = (s,\omega)$ has state s and stack ω. The stack height of c is $|c| = |\omega|$.

If $(s,A) \overset{\pi}{\to} (s',y)$ is a transition rule, then we write $(s,\omega A) \overset{\pi}{\to} (s,\omega y)$ for any ω in Γ^* and call it a move of M which reads π, and if $\pi = \varepsilon$ we call it an ε-move. A derivation (or an α-derivation) $c \overset{\alpha}{\to} c$ is a sequence of such moves through successive configurations where α is the concatenation of the input symbols read by the moves. α may be omitted if it is irrelevant.

The language $L(c)$ accepted from a configuration c is $L(c) = \{\alpha\in\Sigma^* \mid c \overset{\alpha}{\to} c',\ \text{mode } (c')\in F\}$, and $L(M) = L((s_0,z_0))$ is the language accepted by M; here configuration $c_s = (s_0,z_0)$ is the initial configuration of M. A configuration is reachable if $c_s \overset{\alpha}{\to} c$ for some α.

Two configurations c and c' are equivalent ($c \equiv c'$), if

$L(c) = L(c')$, and two dpda's M and M' are equivalent ($M \equiv M'$), if $L(M) = L(M')$.

A dpda is called finite-turn if there is a bound on the number of times the direction of the stack movement can change. More precisely, a derivation is an upstroke if no single move in it decreases the height of the stack and a downstroke if no move increases the height of the stack. A dpda is finite-turn if there is an integer k such that every derivation in M from the initial configuration can be segmented into $\leq k$ strokes alternating in direction. We associate with each configuration c an order which is the number of turns made in a derivation $c_s \overset{\alpha}{\to} c$. We will assume a finite-turn dpda to be in a normal form such that the order of a configuration is uniquely determined by its state (cf[V2, B]).

We will be particularly interested with 1-turn dpdas, sometimes referred to as linear dpdas. It should be noted that not all deterministic linear context free languages are accepted by such a device. The following linear context free language is accepted by a dpda, but not by a finite-turn dpda : $L = (a^+b^+)^* L_0 (a^+b^+)^*$ where $L_0 = \{a^n b^n \mid n>0\}$.

3. Construction of M'

Before we present the construction of M', we need the following lemma which is adapted from the work of Valiant on regularity tests [V1]

Lemma 1.

Let M be a dpda. There exists an integer ℓ (depending upon M) such that for each pair of configurations c_0 and c with $c_0 \to c$ and $|c| - |c_0| > \ell$ either

(i) One can eliminate a segment from the top ℓ symbols of c obtaining a configuration c' such that

$$c_0 \to c', \quad |c'| < |c| \quad \text{and} \quad c' \equiv c$$

(ii) there is an infinite collection $c_1, c_2, \ldots$ of pairwise inequivalent configurations with $c_0 \to c_i$.

Proof.

Valiant [V1] proved this result for $c_0 = c_s$. The same proof holds for any c_0, and in particular the bound ℓ is independent of c_0. (For other variations of this lemma cf [HY,IY].) □

Note that if $L(c_0)$ is regular then case (ii) is impossible.

Hence the height of configurations in derivations starting from c_0 may be controlled. This fact plays an important role in the following construction.

Let M be a dpda and $\bar{M}$ a 1-turn dpda. We define a new dpda M' that acts as follows. Given any input string α, M' simulates M while it also remembers the state and stack top of $\bar{M}$. This is done until $\bar{M}$ makes its first erase move (i.e. $\bar{M}$ makes its first and only turn). Suppose M is in configuration $c = (s,w)$ now. Then $L(c)$ must be regular if $\bar{M} \equiv M$, since the equivalent configuration $\bar{c}$ of $\bar{M}$ can only start derivations that never increase the stack height. Then M' enters a new mode of operation. It simulates M, but never increases its stack. Whenever the stack in M grows M' stores the extra symbols in the finite state control. Whenever the number of extra symbol grows beyond ℓ, M' will consult the top ℓ symbols and remove a segment of them so that the new configuration is equivalent to the old one, provided $\bar{M} \equiv M$.

It follows that

Lemma 2.

Let M be a dpda, $\bar{M}$ a 1-turn dpda and let M' be defined from M and $\bar{M}$ as above. $M \equiv \bar{M}$ if and only if $M \equiv M'$ and $\bar{M} \equiv M'$.

4. Parallel stacking

The decision procedures for $\bar{M} \equiv M'$ and $M \equiv M'$ are carried out using the parallel stacking technique of Valiant [V2]. We now review the technique before showing its applicability to our cases. For more details see [V2,B].

Valiant [V2] introduced the parallel stacking technique to simulate two dpdas by a single pda. Using this approach, he was able to prove the decidability of equivalence for two finite turn dpdas. The technique was subsequently improved by Beeri [B], who turned what used to be a partial decision procedure into a decision procedure.

Parallel stacking works as follows. Given dpdas M_1 and M_2 we first form their disjoint union $M = (Q,\Sigma,\Gamma,\delta,s_0,Z_0,F)$. Now we describe a new pda N to simulate M_1 and M_2. In general, each configuration of N simulates two configurations of M. The stack of N has two tracks and there is a state of M associated with each of them. The stack is also divided into segments of bounded size separated by ceilings. Each ceiling occupies both tracks and contains a pair of modes, which are the modes of the two tracks at the time the ceiling was created, and a pair of indicators $(X,Y) \in \{L,R\}^2$.

The indicator X specifies that the left track above the ceiling is to be associated with the X track below (where L stands for left and R for right). Similarly for Y and the right track. The indicators may be used to uniquely reconstruct, from N's configuration, the two configurations of M being simulated. N can manipulate, in one move, the top segment of the stack. This can be done since the segments are bounded in length.

Usually, M conducts a simultaneous simulation of the steps M would have performed in each of the two configurations now represented. The simulation may leave one configuration unchanged if it has a reading mode while the other one is in ε-mode. In this case the simulation consumes no input, and a ε-move is applied to the ε-mode. To insure bounded segment sizes N will occasionally perform the following steps, depending on the contents of the top segment:

(a) If one track of the top segment becomes empty, the ceiling below it is removed and the two topmost segments fused. The indicator pair (X,Y) of that ceiling is consulted to decide how to associate the segments, and the indicator pair of the ceiling below is changed to (X,Y). It is easy to verify that this step does not alter the two configurations of M now being simulated.

(b) If each track in the top segment is of length larger than 1, then a new ceiling is placed just below the top symbol of each track with indicator pair (L,R). (Again, the configurations remain unchanged.)

(c) If in the top segment one track contains exactly one symbol while the other is "long" N will nondeterministically move as follows. A shorter stack segment (whose length is at most a constant ρ) with the same mode and indicator is chosen, and N nondeterministically continues to simulate this new configuration (obtained when only the top segment is replaced as described) against either of the old configurations. Valiant [V2] showed the existence of such a replacement configuration and Beeri [B] showed how the shorter replacement segment is computed from the current modes and the top ceiling. The two old configurations are equivalent if and only if they are equivalent to the replacement configuration. We refer to this nondeterministic step as c-transformation.

N accepts when both segments are in reading modes exactly one of which is accepting.

If this can be made to work, then N will accept the empty set if and only if the two original configurations are equivalent. The

only thing that remains to be shown, is that the lengths of stack segments are bounded.

Valiant shows [V2] that there are two cases where a segment may grow out of bound (also see [B])

(i) A decreasing track is emptied causing two segments (in the other track) to merge.

(ii) One track is increasing (quickly) while the other is decreasing too slowly to allow a c-transformation.

It is shown in [V2,B] that only a bounded amount of growth can occur before the minimal order of the two configurations is increased (both machines make turns, or a similar effect is achieved by a c-transformation). Thus, for two finite-turn configurations, the simulation may be carried out.

We would like to tune this technique to other situations, in which the two dpdas are not known to be finite-turn.

5. Equivalence of $\bar{M}$ and M'

We use parallel stacking with the following modifications. As long as $\bar{M}$ is in an upstroke, we add ceilings whenever necessary. Whenever $\bar{M}$'s stack grows too fast, we carry out the c-transformation as usual. A c-transformation yields one pair of $\bar{M}$ configurations (which causes no problem as $\bar{M}$ is finite turn) or a pair with one configuration of $\bar{M}$ and one of M' (similar to the original). Whenever M''s stack grows faster, we pad the stack in $\bar{M}$ by placing a dummy symbol just below the top symbol of the stack. These dummy symbols will be erased when exposed, leaving the state of $\bar{M}$ unaltered. This does not change the 1-turn property of $\bar{M}$.

When $\bar{M}$ makes a turn, M' also makes its last turn. From now on, both machines are in a downstroke. We can now continue the simulation as usual, making c-transformations when necessary. (Note that these do not change the modes of the machines, which control the direction of the stack).

We have to show that segments cannot grow without bound when the configurations being simulated are - an $\bar{M}$ configuration in an upstroke and an M' configuration. (The other cases involve two finite turn configurations and are thus covered by [V2, B]). In our case, it is easy to show, by induction, that segments in the M' stack being simulated are of size 1 throughout. Therefore, anytime the $\bar{M}$ top segment grows beyond ρ, a c-transformation will be applicable and there-

fore segments of the M stack are bounded.

We have thus proved the following result.

Lemma 3.

Equivalence of $\bar{M}$ and M' is decidable by parallel stacking.

6. Equivalence of M and M'

We use parallel stacking. At first, M' acts just like M so the stacks grow together (and all segments are of size 1). When M' departs from M, it is in its final downstroke. For any M configuration c, let h(c) denote the M' configuration obtained from it by leaving the stack intact and letting the top portion stored in the finite state control be empty. Clearly, if L(c) is regular, then $c \equiv h(c)$ (e.g. M' can successfully simulate c). We use this idea to devise a new transformation that will be used after every single step of simulation when M' is in its final downstroke. If the configurations we simulate are c against c', we nondeterministically continue with c against h(c) and c' against h(c). Since L(c') is indeed regular, this is a valid transformation. Simulation of c' against h(c) causes no problem, since they are both finite turn (we may use c-transformation when needed, etc.). On the other hand, $|c| = |h(c)|$ so that applying this as described, insures that the pairs of M and M' configurations we simulate will never have different height.

We must show that a simulation step, followed by our transformation, cannot lead to a loop due to ε-moves (i.e. generate the same pair of configurations). We only need to insure that c and h(c) are distinct from the pair of configuration that where represented prior to the application of the simulation step that led to c and c'. (The other pair c' and h(c) is of two M' configurations.) But c and h(c) have identical modes. If they were the configurations prior to the simulation step then both would have been changed by it.

This completes the proof of our remaining lemma.

Lemma 3.

Equivalence of M and M' is decidable by parallel stacking.

Note that the simulation of M and M' may be carried out using just one stack, which holds M's stack. Any time M' starts to grow its "buffer", which is kept in the finite state control, the top symbol of the stack is marked. When the buffer is emptied, the simula-

tion continues (using just the stack) provided the top of the stack is marked (and the mark is then removed.) If the stack has some symbols above the mark when the buffer is emptied, then the simulation fails. The simulation also fails if the mark is reached when the buffer is not empty.

7. Conclusions

The main result now follows.

Theorem 1.

It is decidable whether two dpda's, one of which is linear (i.e. 1-turn) are equivalent.

Proof.

Follows from lemmas 2,3 and 4. □

Note that we could get a single simulating machine to work as follows. First, simulate M and $\bar{M}$ as described by the algorithm to decide equivalence of M' and $\bar{M}$. When $\bar{M}$ makes its turn, and the configurations are c and $\bar{c}$, nondeterministically continue to check equivalence of c and $h(c)$ and of $\bar{c}$ and $h(c)$ according to the appropriate algorithm.

Subclass containment Problems (SCP) are well known problems related to the equivalence problem [FG]. The SCP for 1-turn languages is: given a dpda M, does M accept a language that can also be accepted by a 1-turn dpda? From [FG] it follows that if the SCP for 1-turn languages is decidable then so is the equivalence problem for two dpdas one of which is known to accept a 1-turn language. (This is slightly stronger than our result.) We conjecture that the SCP for 1-turn languages is decidable. Note that this problem, as well as the following two variations on it, become polinomially decidable, when M is a real time dpda accepting by empty store. Is the language accepted by M a finite-turn language? [I1,I2], is it linear context free? [G].

Another possible extension of our result is to obtain a decision procedure for the equivalence of two dpdas, one of which is finite-turn. We have learnt that this problem was recently solved by Oyamaguchi, Inagaki and Honda [OIH]. We do not know, however, how their decision procedure works.

References

[B] C. Beeri, An improvement on Valiant's decision procedure for equivalence of deterministic finite-turn pushdown automata, TCS 3(1976) 305-320.

[FG] E.P. Friedman and S.A. Greibach, On equivalence and subclass containment problems for deterministic context-free languages, Info. Proc. Let. 7 (1978) 287 - 290.

[G] S.A. Greibach, Linearity is polynomially decidable for realtime pushdown store automata, 42(1979), 27-37.

[HHY] M.A. Harrison, I.M. Havel and A. Yehudai, On equivalence of grammars through transformation trees, TCS 9(1979), 173-205.

[HY] M.A. Harrison and A. Yehudai, A hierarchy of deterministic languages, JCSS 19(1979), 63-78.

[I1] Y. Itzhaik, Deterministic pushdown automata, Master Thesis, Technion, IIT , June 1978, (In Hebrew.)

[I2] Y. Itzhaik, On containment problems for finite-turn languages, in preparation, 1981.

[IY] Y. Itzhaik and A. Yehudai, Checking deterministic languages for the real time strict property, Submitted for publication (1980).

[OHI] M. Oyamaguchi, N. Honda and Y. Inagaki, The equivalence problem for real-time strict deterministic languages, Info. Contr. 45(1980), 90-115.

[OIH] M. Oyamaguchi, Y. Inagaki and N. Honda, The extended equivalence problems for some subclasses of deterministic pushdown automata, 1980.

[U] E. Ukkonen, The equivalence problem for some non-real-time deterministic pushdown automata, Proceedings of the 12th STOC Conference (1980), 29-38.

[V1] L.G. Valiant, Regularity and related problems for deterministic pushdown automata, JACM 22(1975), 1-10.

[V2] L.G. Valiant, The equivalence problem for deterministic finite-turn pushdown automata, Info. Contr. 25, (1974), 123-133.