Subsequence Automata with Default Transitions

Philip Bille, Inge Li Gørtz, and Frederik Rye Skjoldjensen

Technical University of Denmark
{phbi,inge,fskj}@dtu.dk

Abstract. Let S be a string of length n with characters from an alphabet of size σ. The *subsequence automaton* (often called the *directed acyclic subsequence graph*) is the minimal deterministic finite automaton accepting all subsequences of S. A straightforward construction shows that the size (number of states and transitions) of the subsequence automaton is $O(n\sigma)$ and that this bound is asymptotically optimal.
In this paper, we consider subsequence automata with *default transitions*, that is, special transitions to be taken only if none of the regular transitions match the current character, and which do not consume the current character. We show that with default transitions, much smaller subsequence automata are possible, and provide a full trade-off between the size of the automaton and the *delay*, i.e., the maximum number of default transition followed before consuming a character.
Specifically, given any integer parameter k, $1 < k \leq \sigma$, we present a subsequence automaton with failure transition of size $O(nk \log_k \sigma)$ and delay $O(\log_k \sigma)$. Hence, with $k = 2$ we obtain an automaton of size $O(n \log \sigma)$ and delay $O(\log \sigma)$. On the other extreme, with $k = \sigma$, we obtain an automaton of size $O(n\sigma)$ and delay $O(1)$, thus matching the bound for the standard subsequence automaton construction. The key component of our result is a novel hierarchical automata construction of independent interest.

1 Introduction

Let S be a string of length n with characters from an alphabet of size σ. A *subsequence* of S is any string obtained by deleting zero or more characters from S. The *subsequence automaton* (often called the *directed acyclic subsequence graph*) is the minimal deterministic finite automaton accepting all subsequences of S. Baeza-Yates [1] initiated the study of subsequence automata. He presented a simple construction using $O(n\sigma)$ size (size denotes the total number of states *and* transitions) and showed that this bound is optimal in the sense that there are subsequence automata of size at least $\Omega(n\sigma)$. He also considered variations with encoded input strings and multiple strings. Subsequently, several researchers have further studied subsequence automata (and its variants) [2–9]. See also the surveys by Troníček and others [10, 11]. The general problem of *subsequence indexing*, not limited to automata based solutions, is investigated by Bille et al. [12].

In this paper, we consider subsequence automata in the context of *default transitions*, that is, special transitions to be taken only if none of the regular transitions match the current character, and which do not consume the current character. Each state has at most one default transition and hence the automaton remains deterministic. The key point of default transitions is to reduce the size of standard automata at the cost of introducing a *delay*, i.e., the maximum number of default transition followed before consuming a character. For instance, given a pattern string of length m the classic Knuth-Morris-Pratt (KMP) [13] string matching algorithm may be viewed as an automaton with default transitions (typically referred to as *failure transitions*). This automaton has size $O(m)$, whereas the standard automaton with no default transition would need $\Theta(m\sigma)$ space. The delay of the automaton in the KMP algorithm is either $O(m)$ or $O(\log m)$ depending on the version. Similarly, the Aho-Corasick string matching algorithm for multiple strings may also be viewed as an automaton with default transitions [14]. More recently, default transitions have also been used extensively to significantly reduce sizes of deterministic automata for regular expression [15, 16]. The main idea is to effectively enable states with large overlapping identical sets of outgoing transitions to "share" outgoing transitions using default transitions.

Surprisingly, no non-trivial bounds for subsequence automata with default transitions are known. Naively, we can immediately obtain an $O(n\sigma)$ size solution with $O(1)$ delay by using the standard subsequence automaton (without default transitions). At the other extreme, we can build an automaton with $n+1$ states (each corresponding to a prefix of S) with a standard and a default transition from the state corresponding to the ith prefix to the state corresponding to the $i+1$st prefix (the standard transition is labeled $S[i+1]$). It is straightforward to show that this leads to an $O(n)$ size solution with $O(n)$ delay. Our main result is a substantially improved trade-off between the size and delay of the subsequence automaton:

Theorem 1. *Let S be a string of n characters from an alphabet of size σ. For any integer parameter k, $1 < k \leq \sigma$, we can construct a subsequence automaton with default transitions of size $O(nk \log_k \sigma)$ and delay $O(log_k \sigma)$.*

Hence, with $k = 2$ we obtain an automaton of size $O(n \log \sigma)$ and delay $O(\log \sigma)$. On the other extreme, with $k = \sigma$, we obtain an automaton of size $O(n\sigma)$ and delay $O(1)$, thus matching the bound for the standard subsequence automaton construction.

To obtain our result, we first introduce the *level automaton.* Intuitively, this automaton uses the same states as the standard solution, but hierarchically orders them in a tree-like structure and samples a selection of their original transitions based on their position in the tree, and adds a default transition to the next state on a higher level. We show how to do this efficiently leading to a solution with $O(n \log n)$ size and $O(\log n)$ delay. To achieve our full trade-off from Theorem 1 we show how to augment the construction with additional tricks for small alphabets and generalize the level automaton with parameter k,

$1 < k \leq \sigma$, where large k reduces the height of the tree but increases the number of transitions.

2 Preliminaries

A *deterministic finite automaton* (DFA) is a tuple $A = (Q, \Sigma, \delta, q_0, F)$ where Q is a set of nodes called *states*, δ is a set of labeled directed edges between states called *transitions* where each label is a character from the alphabet Σ, $q_0 \in Q$ is the *initial* state and $F \subseteq Q$ is a set of *accepting* states. No outgoing transitions from the same state have the same label. The *size* of A is the sum of the number of states and transitions.

We can think of A as an *edge-labeled directed graph.* Given a string P and a path p in A we say that p and P match if the concatenation of the labels on the transitions in p is P. We say that A *accepts* a string P if there is a path in A, from q_0 to any state in F, that matches P. Otherwise A *rejects* P.

A *deterministic finite automaton with default transitions* is a deterministic finite automaton AD where each state can have a single unlabeled default transition. Given a string P and a path p in AD we define a match between P and p as before, with the exception that for any default transition d in p the corresponding character in P cannot match any standard transition out of the start state of d. Definition of accepted and rejected strings are as before. The *delay* of AD is the maximum length of any path that only uses default transitions.

A subsequence of S is a string P, obtained by removing zero or more characters from S. A *subsequence automaton* constructed from S, denoted SA, is a deterministic finite automaton that accepts string P iff P is a subsequence of S. The SA is often called the *directed acyclic subsequence graph* or DASG. The SA has $n + 1$ states, all accepting, that we identify with the integers $\{0, 1, \ldots, n\}$. For each state s, $0 \leq s \leq n$, we have the following transitions:

– For each unique character α in $S[s + 1, n]$, there is a transition labeled α to the smallest state $s' > s$ such that $S[s'] = \alpha$.

The SA has size $O(n\sigma)$ since every state can have at most σ transitions. An example of an SA is given in Fig. 1.

A *subsequence automaton with default transitions* constructed from S, denoted SAD, is a deterministic finite automaton with default transitions that accepts string P iff P is a subsequence of S.

The next section explores different configurations of transitions and default transitions in SADs.

3 New Trade-Offs for Subsequence Automata.

We now present a new trade-off for subsequence automata, with default transitions. We will gradually refine our construction until we obtain an automaton that gives the result presented in Theorem 1. In each construction we have $n+1$

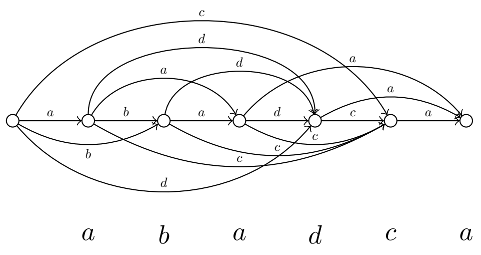

Fig. 1. An example of an SA constructed from the string *abadca*.

states that we identify with the integers $\{0, 1, \ldots, n\}$. Each of these states represents a prefix of the string S and are all accepting states. We first present the *level automaton* that gives the first non-trivial trade-off that exploits default transitions. The general idea is to construct a hierarchy of states, such that every path that only uses default transitions is guaranteed to go through states where the outdegree increases at least exponentially. The level automaton is a SAD of size $O(n \log n)$ and delay $O(\log n)$. By arguing that any path going through a state with outdegree σ will do so by taking a regular transition, we are able to improve both the size and delay of the level automaton. This results in the *alphabet aware level automaton* which is a SAD of size $O(n \log \sigma)$ and delay $O(\log \sigma)$. Finally we present a generalized construction that gives a trade-off between size and delay by letting parameter k, $1 < k \leq \sigma$, be the base of the exponential increase in outdegree on paths with only default transitions. This SAD has size $O(nk \log_k \sigma)$ and delay $O(\log_k \sigma)$. With $k = 2$ we get an automaton of size $O(n \log \sigma)$ and delay $O(\log \sigma)$. In the other extreme, for $k = \sigma$ we get an automaton of size $O(n\sigma)$ and delay $O(1)$.

3.1 Level Automaton

The level automaton is a SAD with $n+1$ states that we identify with the integers $\{0, 1, \ldots, n\}$. All states are accepting. For each state $i > 0$, we associate a level, level(i), given by:

$$\text{level}(i) = \max(\{x \mid i \bmod 2^x = 0\})$$

Hence, level(i) is the exponent of the largest power of two that divides i. We do not associate any level with state 0. For a state s, we define $\overline{s}$ to be the smallest state such that $\overline{s} > s$ and $\text{level}(\overline{s}) \geq \text{level}(s) + 1$.

The configuration of the transitions in the level automaton are as follows: From state 0 we have a default transition to state 1 and a regular transition to state 1 with label $S[1]$. For every other state s, $1 \leq s \leq n$, we have the following transitions.

- A failure transition to state $\overline{s}$. If no such state exist, the state s does not have a failure transition.

- For each unique character α in $S[s+1, \min(\overline{s}, n)]$, there is a transition labeled α to the smallest state $s' > s$ such that $S[s'] = \alpha$.

An example of the level automaton constructed from the string *abacbabcabad* and alphabet $\{a, b, c, d\}$ is given in Fig. 2. The dashed arrows denote default transitions and the vertical position of the states denotes their level.

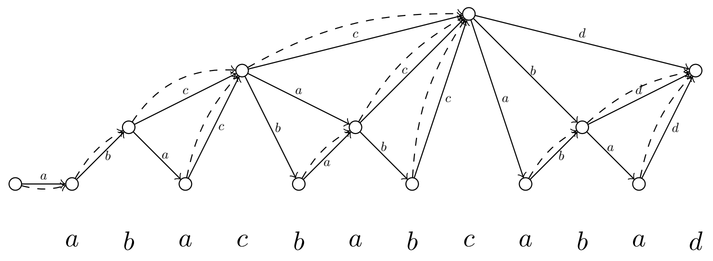

Fig. 2. The level automaton constructed from the string *abacbabcabad*.

We first show that the level automaton is a SAD for S, i.e., the level automaton accepts a string iff. the string is a subsequence of S. To do so suppose that P is a string of length m accepted by the level automaton and let $s_1, s_2, \ldots, s_m$ be the sequence of states visited with regular transitions on the path that accepts P. From the definition of the transition function, we know that if a transition with label α leads to state s', then $S[s'] = \alpha$. This means that $S[s_1]S[s_2] \ldots S[s_m]$ spells out a subsequence of S if the sequence $s_1, s_2, \ldots, s_m$ is strictly monotonically increasing. From the definition of the transitions, a state s only have transitions to states s' if $s' > s$. Hence, the sequence is strictly monotonically increasing.

For the other direction, let P be a subsequence of S. Let $S[1, s_m]$ be the minimal prefix of S that contains P as a subsequence and let $s_1, s_2, \ldots, s_m$ be a strictly monotonically increasing sequence of states, such that $S[s_1]S[s_2] \ldots S[s_m] = P$. Assume for contradiction that P is not accepted by the level automaton. Let P_p be the largest prefix of P that is accepted by the level automaton such that the sequence of states visited with regular transition on the matching path is a prefix of $s_1, s_2, \ldots, s_m$. Let s_i be the last state of that prefix and let $\alpha = S[s_{i+1}]$. Consider the path of states $s_i, d_1, \ldots, d_j$ obtained by continuously following default transitions from s_i. When the default transition of s_i leads to d_1 we know that s_i has a transition with label α, if α occurs in $S[s_i+1, d_1]$. Applying this argument to the rest of the states on the path, we know that one of the states $s_i, d_1, \ldots, d_j$ has a transition labeled α, if α occurs in $S[s_i + 1, \min(\overline{d_j}, n)]$, where $\min(\overline{d_j}, n) = n$ because d_j has no default transition. Since $S[s_{i+1}]S[s_{i+2}] \ldots S[s_m]$ is a subsequence of $S[s_i + 1, n]$ one of the states $s_i, d_1, \ldots, d_j$ has a transition to s_{i+1} with label α. This contradicts how we initial selected P_p.

For all $s > 0$, we want to show the following property of $\overline{s}$ and level(s):

$$\overline{s} - s = 2^{\text{level}(s)} \tag{1}$$

By definition $2^{\text{level}(s)}$ divides s. Hence, the next integer, larger than s, that $2^{\text{level}(s)}$ divides must be $s + 2^{\text{level}(s)}$. But since $2^{\text{level}(s)}$ was the largest power of two that divides s, we know that the decomposition of s into a sum of unique powers of two must contain $2^{\text{level}(s)}$ and also that this power is the smallest. Hence, when we decompose $s + 2^{\text{level}(s)}$ into unique powers of two we know that the smallest power of two in this decomposition is at least $2^{\text{level}(s)+1}$. The definition of $\overline{s}$ implies that $\overline{s} = s + 2^{\text{level}(s)}$ which is exactly what we wanted to show.

Assume that the delay of the level automaton is due to path p. We want to bound the length of p. The state s on p is the only state on p that can have transitions to the states $s+1, s+2, \ldots, \overline{s}$. Since $\overline{s} - s = 2^{\text{level}(s)}$ and level$(\overline{s}) >$ level(s), the length of p is bounded by $O(\log n)$ because a state s at level $O(\log n)$ can have transitions to $\overline{s} - s = 2^{O(\log n)} = O(n)$ states.

At each level l we have $O(n/2^{l+1})$ states, because every 2^lth state is divided by 2^l, but 2^l is only the largest divisor in every second of these cases. Because $\overline{s} - s = 2^{\text{level}(s)}$ each state at level l has at most 2^l outgoing transitions. Therefore, each level contribute with size at most $n/2^{l+1} \cdot 2^l = O(n)$. Since the delay is $O(\log n)$ we have at most $O(\log n)$ levels and the total size therefore becomes $O(n \log n)$.

In summary, we have shown the following result.

Lemma 1. *Let S be a string of n characters. We can construct a subsequence automaton with default transitions of size $O(n \log n)$ and delay $O(\log n)$.*

3.2 Alphabet aware level automaton

We introduce the *Alphabet aware level automaton.* When the level automaton reaches a state s where $\overline{s} - s \geq \sigma$, then s can have up to σ outgoing transitions without violating the space analysis above. The level automaton only has a transition for each unique character in $S[s+1, \min(\overline{s}, n)]$. Hence, for all states s in the alphabet aware level automaton where $\overline{s} - s \geq \sigma$, we let s have a transition for each symbol α in Σ, to the smallest state $s' > s$ such that $S[s'] = \alpha$. No accepting path would ever take a default transition from a state with σ outgoing transitions. Hence, states with σ outgoing transitions do not need default transitions.

We change the level function to reflect this. For each state $1 \leq i \leq n$ we have that:

$$\text{level}(i) = \min(\lceil \log_2 \sigma \rceil, \max(\{x \mid i \bmod 2^x = 0\})) \tag{2}$$

The configuration of the transitions in the alphabet aware level automaton is as follows: From state 0 we have a default transition to state 1 and a regular transition to state 1 with label $S[1]$. For every other state s, $1 \leq s \leq n$, we have the following transitions.

- A failure transition to state $\overline{s}$. If no such state exist, the state s does not have a failure transition.
- If $\overline{s} - s < \sigma$ then for each unique character α in $S[s+1, \min(\overline{s}, n)]$, there is a transition labeled α to the smallest state $s' > s$ such that $S[s'] = \alpha$.
- If $\overline{s} - s \geq \sigma$ then for each unique character α in $S[s+1, n]$, there is a transition labeled α to the smallest state $s' > s$ such that $S[s'] = \alpha$.

An example of the alphabet aware level automaton constructed from the string *abacbabcabad* and alphabet $\{a, b, c, d\}$ is given in Fig. 3. The level automaton in Fig. 2 is constructed from the same string and the same alphabet. For comparison, state 4 in Fig. 3 now has outdegree σ and has transitions to the first succeeding occurrence of any unique character and state 8 has been constrained to level $\lceil \log_2 \sigma \rceil$.

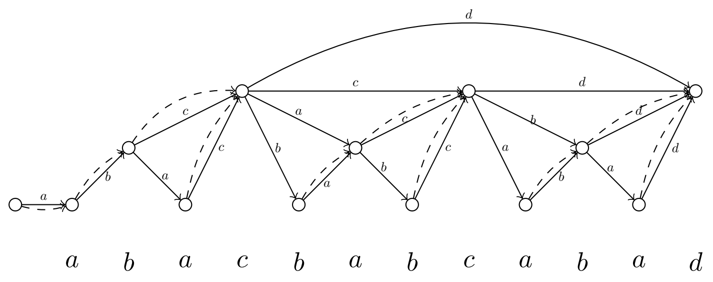

Fig. 3. The alphabet level automaton constructed from the string *abacbabcabad*.

It is easy to show that the alphabet aware automaton is a SAD by slightly modifying the arguments that lead to Lemma 1. Now, it is not necessarily true that $\min(\overline{d_j}, n) = n$ for all paths $s_i, d_1, \ldots, d_j$. So assume that $\min(\overline{d_j}, n) < n$. But because d_j has no default transition, we know instead that d_j has σ outgoing transitions where one of them has label α, if α occurs in $S[d_j + 1, n]$. From this we can conclude that one of the states $s_i, d_1, \ldots, d_j$ has a transition to s_{i+1} with label α.

The delay is now bounded by $O(\log \sigma)$ because no state is assigned a level higher than $\lceil \log_2 \sigma \rceil$. The size is bounded by $O(n \log \sigma)$ because each level has size $O(n)$ and we have at most $O(\log \sigma)$ levels.

In summary, we have shown the following result.

Lemma 2. *Let S be a string of n characters. We can construct a SAD of S with size $O(n \log \sigma)$ and delay $O(\log \sigma)$.*

3.3 Full trade off

We can generalize the construction above by introducing parameter k, $1 < k \leq \sigma$, which is the base of the exponential increase in outdegree of states on every path that only uses default transitions. Now, when we follow a default transition from s to $\overline{s}$, the number of outgoing transitions increase with a factor k instead of a factor 2. This gives a trade-off between size and delay in the SAD determined by k. Increasing k gives a shorter delay of the SAD but increases the size and vice versa.

Each state, except state 0, is still associated with a level, but we need to generalize the level function to account for the parameter k. For every k and $1 \leq i \leq n$ we have that:

$$\text{level}(i,k) = \min(\lceil \log_k \sigma \rceil, \max(\{x \mid i \bmod k^x = 0\})) \tag{3}$$

Now, the level function gives the largest power of k that divides i.

The configuration of the transitions in the generalized alphabet aware level automaton is identical to the configuration of the alphabet aware automaton but is restated here for completeness. From state 0 we have a default transition to state 1 and a regular transition to state 1 with label $S[1]$. For every other state s, $1 \leq s \leq n$, we have the following transitions:

– A failure transition to state $\overline{s}$. If no such state exist, the state s does not have a failure transition.
– If $\overline{s} - s < \sigma$ then for each unique character α in $S[s+1, \min(\overline{s}, n)]$, there is a transition labeled α to the smallest state $s' > s$ such that $S[s'] = \alpha$.
– If $\overline{s} - s \geq \sigma$ then for each unique character α in $S[s+1, n]$, there is a transition labeled α to the smallest state $s' > s$ such that $S[s'] = \alpha$.

We can show that the generalized alphabet aware automaton is a SAD by the same arguments that lead to 2.

With the new definition of the level function we have that

$$\overline{s} - s = k^{\text{level}(s,k)+1} - (s \bmod k^{\text{level}(s,k)+1}) \tag{4}$$

for all $s > 0$.

The modulo term gives the difference between s and the largest $j < s$ such that $k^{\text{level}(s,k)+1}$ divides j. We are interested in the difference between s and the smallest $i > s$ such that $k^{\text{level}(s,k)+1}$ divides i. But since the difference between j and i is $k^{\text{level}(s,k)+1}$, we just subtract the modulo term from $k^{\text{level}(s,k)+1}$ to get the difference between s and i. This expression gives the number of outgoing transitions from state s. The delay is bounded by $O(\log_k \sigma)$ because we increase the number of outgoing transitions with a factor k each time we follow a default transition and no state is assigned a level higher than $\lceil \log \sigma \rceil$.

Each state has at most $\overline{s} - s = k^{\text{level}(s,k)+1} - (s \bmod k^{\text{level}(s,k)+1}) \leq k^{\text{level}(s,k)+1}$ outgoing transitions. At level l we have $O(n(k-1)/(k^{l+1}))$ states each with $O(k^{l+1})$ outgoing transitions such that each level has size $O(nk)$. The size of the automaton becomes $O(nk \log_k \sigma)$ because we have $O(\log_k \sigma)$ levels.

In summary, we have shown Theorem 1.

References

1. Baeza-Yates, R.A.: Searching subsequences. Theoret. Comput. Sci. **78**(2) (1991) 363–376
2. Troníček, Z., Shinohara, A.: The size of subsequence automaton. Theoret. Comput. Sci. **341**(1) (2005) 379–384
3. Crochemore, M., Melichar, B., Troníček, Z.: Directed acyclic subsequence graph: Overview. J. Disc. Algorithms **1**(3-4) (2003) 255–280
4. Crochemore, M., Troncek, Z.: Directed acyclic subsequence graph for multiple texts. Technical Repport, Institut Gaspard-Monge (1999) 99–13
5. Troníček, Z.: Episode matching*. In: Proc. 12th. CPM. (2001) 143–146
6. Hoshino, H., Shinohara, A., Takeda, M., Arikawa, S.: Online construction of subsequence automata for multiple texts. In: Proc. 7th SPIRE. (2000) 146–152
7. Farhana, E., Ferdous, J., Moosa, T., Rahman, M.S.: Finite automata based algorithms for the generalized constrained longest common subsequence problems. In: Proc. 17th SPIRE. (2010) 243–249
8. Bannai, H., Inenaga, S., Shinohara, A., Takeda, M.: Inferring strings from graphs and arrays. In: Proc. 28th MFCS. (2003) 208–217
9. Tronìček, Z.: Operations on DASG. In: Proc. 4th WIA. (1999) 82–91
10. Troníček, Z.: Searching subsequences. Ph. D. Thesis, Department of Computer Science and Engineering, FEE CTU in Prague (2001)
11. Troniĉek, Z.: Common subsequence automaton. In: Proc. 8th CIAA. (2003) 270–275
12. Bille, P., Farach-Colton, M.: Fast and compact regular expression matching. Theoret. Comput. Sci. **409** (2008) 486 – 496
13. Knuth, D.E., James H. Morris, J., Pratt, V.R.: Fast pattern matching in strings. SIAM J. Comput. **6**(2) (1977) 323–350
14. Aho, A.V., Corasick, M.J.: Efficient string matching: an aid to bibliographic search. Commun. ACM **18**(6) (1975) 333–340
15. Kumar, S., Dharmapurikar, S., Yu, F., Crowley, P., Turner, J.: Algorithms to accelerate multiple regular expressions matching for deep packet inspection. In: Proc. 12th SIGCOMM. (2006) 339–350
16. Hayes, C.L., Luo, Y.: Dpico: a high speed deep packet inspection engine using compact finite automata. In: Proc. 3rd ANCS. (2007) 195–203