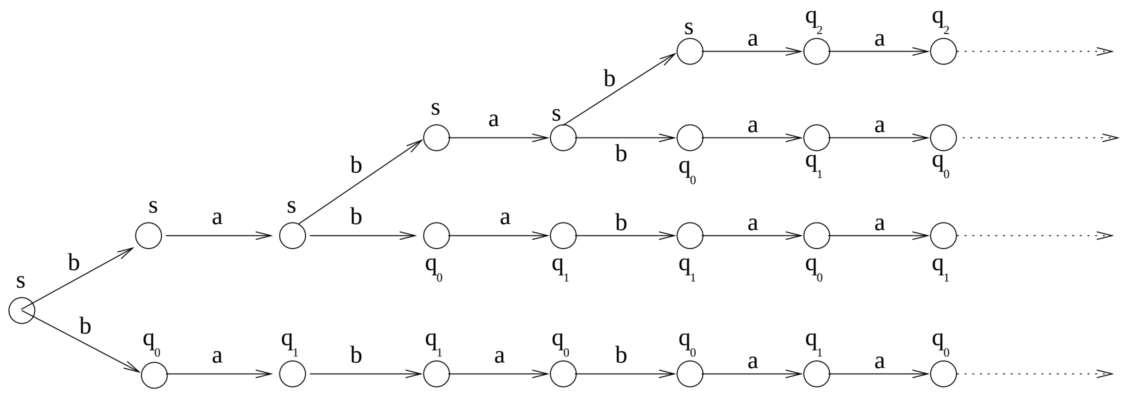
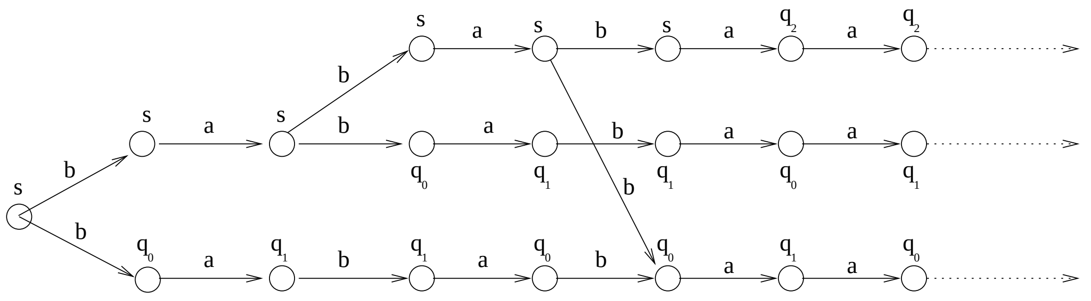
Lecture 10: Complementation via Alternating Automata

The last theorem of Lecture 9 assures us that in any Büchi game one of the two players has a positional winning strategy. Thus, for any alternating Büchi automaton (ABA) A and word w, either the automaton or the pathfinder has a positional winning strategy in the game $\mathcal{G}(A, w)$. Thus, if A accepts w there is a positional run for A on w. Consider the ABA A_1 from the previous lecture. Here is a positional accepting run for this automaton on input $bababaaaa\ldots$.



Observe that positionality only places requirements on occurances of a state at the same level and NOT across levels. The move at the s-labelled state at level 2 is not the same as that at the s-labelled state at level 6. However, both the q_0-labelled states at level 6 have identical subtrees below them. In any level, we can collapse together all the vertices that are labelled by the same state (or equivalently (since the run is positional) whenever the entire subtree rooted at these vertices is isomorphic) to obtain a DAG. Such a DAG has at the most $|Q|$ vertices at each level. Here is the DAG obtained from the above run:



Henceforth, by a "positional run" we refer to this DAG representation of the run. We can also directly define this run-DAG for an automaton $A = (Q, \Sigma, \delta, s, F)$ on an input $a_1a_2\ldots$ as the DAG satisfying:

1. The vertices at level i in are labelled by elements of $Q \times \{i\}$ (and at the most one vertex is labelled by any label.)

2. There is unique vertex at level 1 labelled by $(s, 1)$.

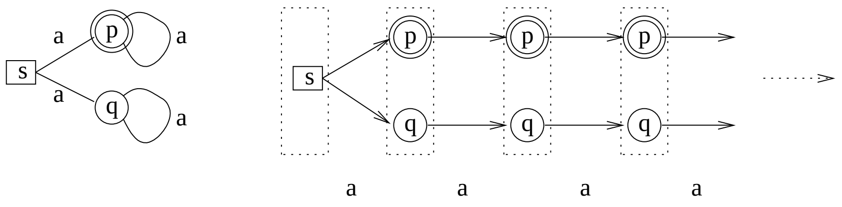
3. If some vertex at level i is labelled by (q, i) and the set of vertices at level $i+1$ to which this vertex is connected by edges is $\{(q_1, i+1), (q_2, i+1) \ldots (q_m, i+1)\}$ then $\{q_1, \ldots q_m\} \models \delta(q, a_i)$.

(We could also insist that any vertex at levels $2, 3, \ldots$ must have an indegree of at least 1, but we don't have to.)

A run is accepting if every vertex in the run has an outdegree of at least 1 and every path through the DAG visits vertices in F infinitely often.

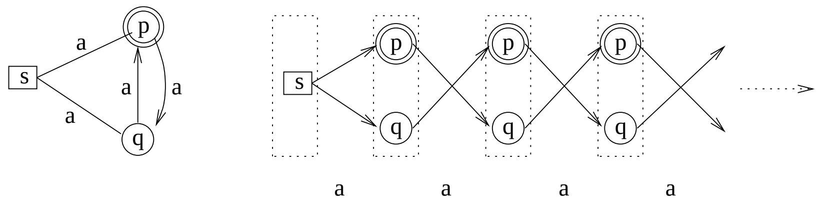
As in the case finite automata, positional runs of an alternating automaton can be simulated using a nondeterministic automaton which simply generates the run-DAG level by level. That is, given an alternating autmaton $A = (Q, \Sigma, \delta, s, F)$ we can construct the nondeterministic automaton $A' = (2^Q, \Sigma, \delta', \{s\}, \emptyset)$ (ignore the accepting set for the moment), with $\delta'(X, a) = \{X' \mid \forall q \in X.\ X' \models \delta(q, a)\}$. Thus X' must have enough states to ensure that $\delta(q, a)$ is satisfied for each $q \in X$. It is quite easy to check that this automaton simulates the positional runs of the alternating automaton level by level.

Can we set the accepting set to some F' so that the resulting NBA accepts the same language as A? Well, some obvious candidates are $F_1' = \{X \mid X \cap F \neq \emptyset\}$ and $F_2' = \{X \mid X \subseteq F\}$. Will either of these work? Here is a counter example to the choice F_1'. The transitions out of s are conjunctive and the right-handside of the figure indicates the unique run of this automaton on $aaa\ldots$.



At each level there is at least one state from F, however the run is not accepting as there is a path that does not visit F infintely often. Thus this word is not accepted by the alternating Büchi automaton. On the other hand, every level of this run intersects the set F. In the above figure, the dotted boxes indicate the states of the NBA simulating the ABA level by level. From the second state onwards every state of this NBA is in F_1 and thus it would erroneously accept this word.

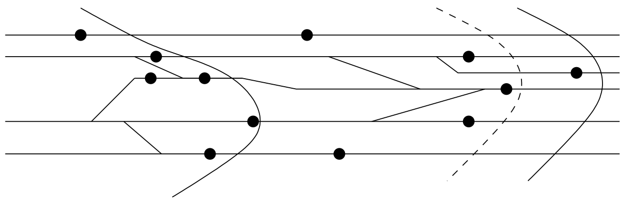
Now, consider the following example:



Here the word is accepted by the alternating automaton as you can see that every path in the unique run on $aaa\ldots$ visits p infinitely often. The states of the NBA are marked by the dotted lines. Notice that none of these states would belong to F_2' since $q \notin \{p\} = F$.

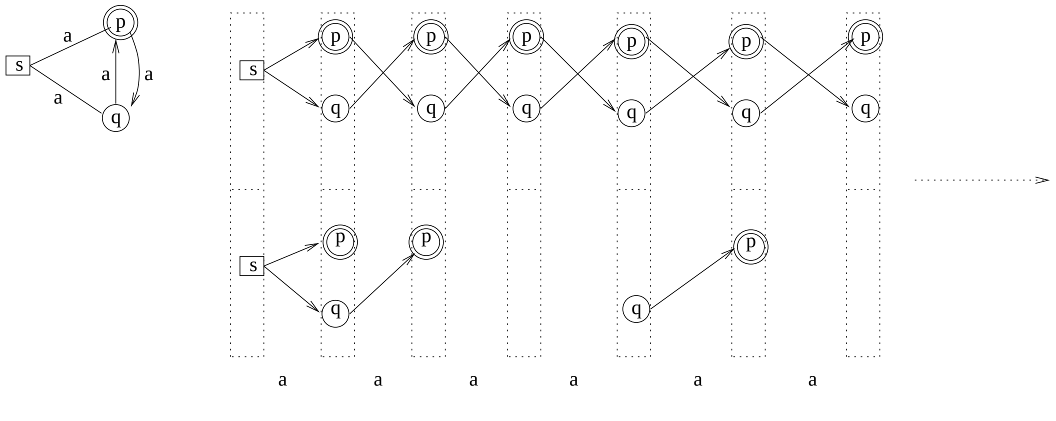
In effect the first choice is an overapproximation and the latter an underapproximation. It turns out that in order for the NBA to simulate the ABA and verify that every path visits F infinitely often we need to maintain more information in the NBA. The idea is to do the following:

1. Simulate the ABA level by level.
2. Verify that we can slice the run into infinitely many segments such that in each segment every path in the segment visits F.



The dark circles denote elements of F. The solid curve on the left identifies a segment (everything to its left) in which every path visits an element of F. The part between the first solid curve and the dotted curve is not good enough. There is one path ("forked" within this segment) that does not visit states in F. The part between the first and second solid curves is a good segment — every path in this segment visits F.

We simulate the run of the ABA level by level as usual. At the "beginning" of each slice, we make a copy of the current set of paths (that are not already in a state in F) in the simulation. In this second copy we extend a path only if it has not visited a final state yet. If at some point the second copy becomes empty (indicating that we have completed one more good slice), we copy the current set of paths from the first copy and repeat this process all over again. If the second copy becomes empty infinitely often then we know that the run can be sliced into infinitely many good slices and thus every path in this run visits F infinitely often. Here is an alternating automaton and a run of its equivalent NBA on the word $aaa\ldots$.



The dotted rectagular boxs contains the states of the NBA. Each box is divided into two by a line in the middle. The part of the state above the middle line simulates the ABA level by level. Below the middle line, a simulation on the current slice is carried out. At the beginning s is copied at the lower level and after 2 moves we find that all paths starting at s have visited a state in F. Thus the lower component of the state reached after reading *aaa* is empty indicating that the first slice has ended. In the next move we copy the non-final states from the upper copy to start the simulation on next slice and so on.

Here is a formal description of the NBA A' simulating the alternating automaton A. $A' = (Q', \Sigma, \delta', (\{s\}, \{s\}), F')$ where

$$\begin{array}{lcl} Q' & = & \{(X,Y) \in 2^Q \mid Y \subseteq X\} \\ F' & = & \{(X, \emptyset) \mid X \in 2^Q\} \\ \delta'((X,Y),a) & = & \{(\bigcup_{q \in X} X_q, \bigcup_{q \in Y} X_q \setminus F) \mid \text{ where } X_q \in 2^Q \text{ such that } X_q \models \delta(q,a)\} \end{array}$$

Note that for states in $X \cap Y$, we make the same move in both copies and clearly this is necessary.

Theorem 1 *(Miyano and Hayashi) For any alternating Büchi automaton A with n states there is a nondeterministic Büchi automaton A' with at the most $2^{O(n)}$ states with $L(A') = L(A)$. Thus the class of languages accepted by alternating Büchi automata is the class of ω-regular languages.*

Proof: It is not difficult to check that the automaton A' defined above satisfies the requirements of this theorem. ■

1 Complementation and co-Büchi Automata

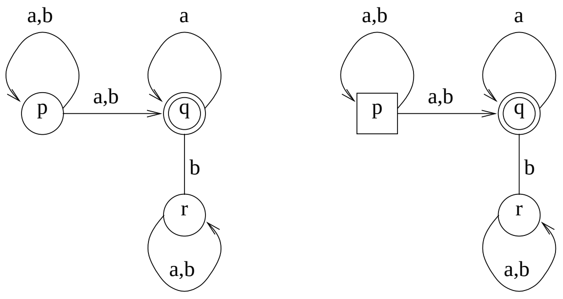
How do we complement alternating Büchi automata? Well, we take the automaton with the "dual" transition relation and "complement" the acceptance condition. Unfortunately, the complement of the Büchi acceptance condition is NOT a Büchi condition. The complement says that "the path does NOT visit F infinitely often" or equivalently it says that " the path visits F finitely often". This is not equivalent to insisting that $Q \setminus F$ is hit infinitely often as a path could visit both Q as well as $Q \setminus F$ infinitely often.

Let us define a *co-Büchi automaton* A to be a tuple $(Q, \Sigma, \delta, s, F)$ where Q, Σ, δ, s and F are as before. The definition of a run is also the same as for Büchi automata. The only difference is in the definition of *accepting runs.* Here, a run ρ is accepting if $\mathit{inf}(\rho) \cap F = \emptyset$. We can also extend this idea to define alternating co-Büchi automata (Aco-BA). We leave the details to the reader.

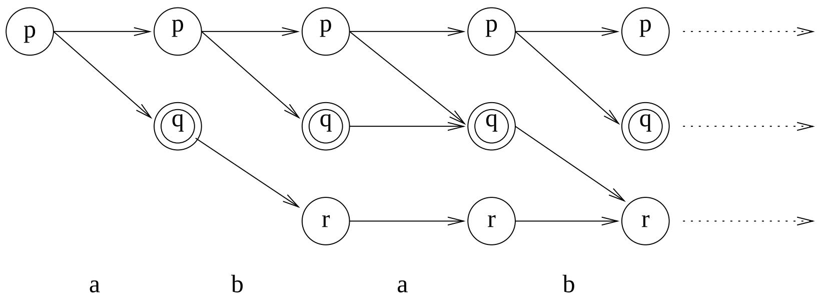
Theorem 2 *Let $A = (Q, \Sigma, \delta, s, F)$ be an alternating Büchi automaton. Then, the alternating co-Büchi automaton $A^d = (Q, \Sigma, \delta^d, s, F)$ accepts the language $\overline{L(A)}$.*

Proof: Let $w \in \Sigma^\omega$. A does not accept w if and only if the automaton does not have a winning strategy in the Büchi game $\mathcal{G}(A, w)$. This happens if and only if the pathfinder has a winning strategy in the game $\mathcal{G}(A, w)$. That is, if and only if, the pathfinder has a strategy that ensures that every play in $\mathcal{G}(A, w)$ visits F only finitely often. But the game $\mathcal{G}(A^d, w)$ is just the game $\mathcal{G}(A, w)$ with the roles of the automaton and the pathfinder interchange. Thus this is equivalent to saying that the automaton has a strategy in the game $\mathcal{G}(A^d, w)$ to ensure that any play consistent with this strategy visits F finitely often. And by the connection between wins and strategies, this happens if and only if the co-Büchi automaton A^d has an accepting run on the word w. ■

Here is a NBA automaton A_f accepting the set of words with finite number of $b's$ and the corresponding Aco-BA A_∞ accepting the set of words with infinitely many bs.



Since the only state exhibiting nondeterminism is p, the correponding dual automaton is one in which this state is a $\forall$ state. Here is an accepting run for this automaton on the word $ababab\ldots$



Notice that every path through this DAG visits q only finitely often (as a matter of fact, no path visits q more than twice).

To obtain a NBA accepting $\overline{L(A)}$ from a NBA A, we need to tranform the Aco-BA A' to an equivalent NBA.

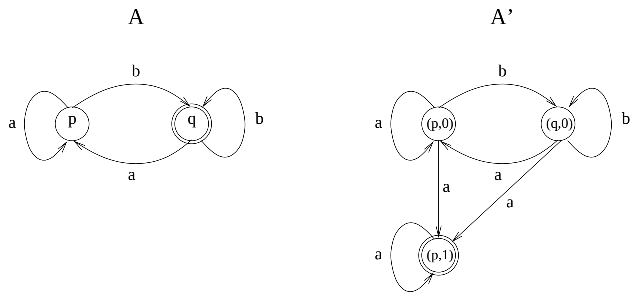
2 Transforming co-Büchi automata into Büchi Automata

How do we transform a Nondeterministic co-Büchi automaton into an equivalent Büchi automaton? For this, let us examine the structure of an accepting run of a Nco-BA . The run looks like $q_1 \xrightarrow{a_1} q_2 \ldots q_{i-1} \xrightarrow{a_{i-1}} q_i \xrightarrow{a_i} \ldots$, where $q_j \notin F$ for all $j \geq i$ for some i. The run decomposes into two parts, an initial finite fragment and an infinite suffix which stays entirely within $Q \setminus F$. Thus, we simply have to nondeterministically guess a position and verify that F is not visited beyond that point.

Can we translate this into a Büchi condition? We need to transform a property of the form $\exists x.\forall y > x.\phi(y)$ to a property of the form $\exists^{\infty} y.\phi(y)$ (i.e. there exists infinitely many ys such that ...).

This is quite easy to arrange. Make two copies of the automaton, in the second copy delete all the states in F. The automaton begins in the first copy and nondeterministically moves to the second copy. This automaton visits states in the second copy infinitely often if and only if beyond a point it stays entirely within the second copy. Thus by treating every state in the second copy as an accepting state (recall that the second copy only has the states $Q \setminus F$) we get an equivalent Büchi automaton.

Here is a co-Büchi automaton A and its equivalent Büchi automaton A':



Moreover, observe that the automaton A' when considered as a co-Büchi automaton with the set of states in the first copy as the accepting set accepts $L(A)$. This is because of the special structure of this automaton. A run either visits the first copy infinitely often and the second copy finitely often or visits the first copy finitely often and second copy infinitely often. So saying "finitely often" about one copy is the same as saying "infinitely often" about the other copy.

Theorem 3 *Let $A = (Q, \Sigma, \delta, s, F)$ be a nondeterministic co-Büchi automaton. Then, the Büchi automaton $A' = (Q', \Sigma, \delta', (s, 0), Q \times \{1\})$ where*

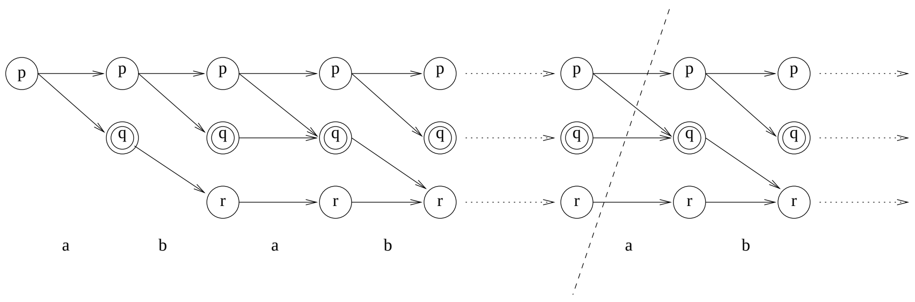
$$\begin{array}{lcl} Q' & = & Q \times \{0\} \ \cup \ (Q \setminus F) \times \{1\} \\ \delta((q,0),a) & = & \delta(q,a) \times \{0,1\} \cap Q' \\ \delta((q,1),a) & = & \delta(q,a) \times \{1\} \cap Q' \end{array}$$

accepts the same language as A. Further the co-Büchi automaton $(Q', \Sigma, \delta, (s, 0), Q \times \{0\})$ also accepts the same language.

Exercise: Write down a formal proof of Theorem **??**.

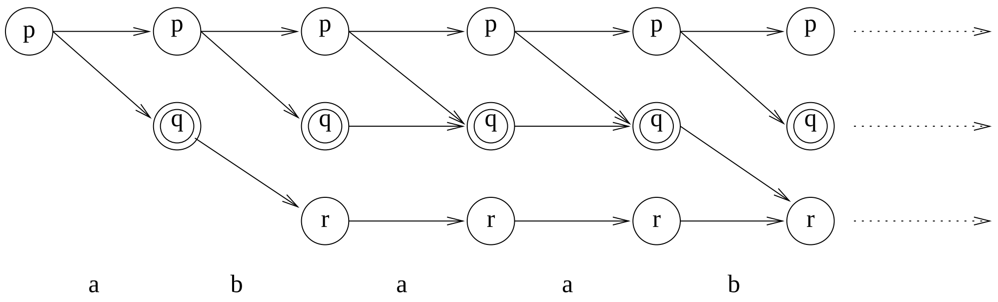
2.1 Alternating co-Büchi automata to Büchi automata

Can we lift the above argument to work for Alternating co-Büchi automata? Let us consider the Aco-BA A_{∞} described in the previous section and its run on $ababab\ldots$

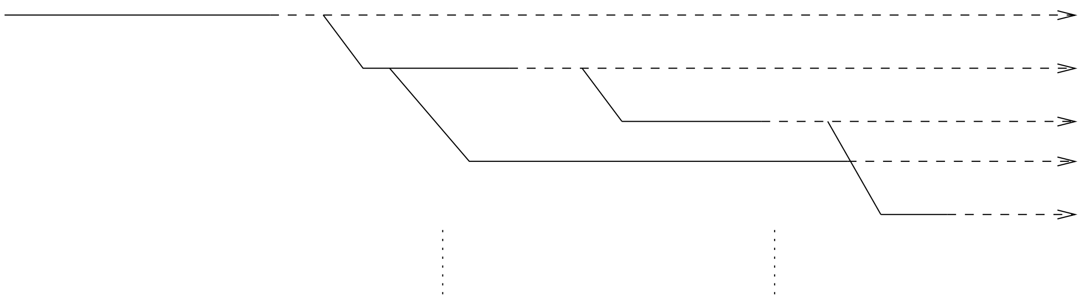


No matter where we slice this DAG the suffix always contains states in F. Thus, a simple two copy idea like the one used for nondeterministic co-Büchi automata will not work here.

Well, in this run no path visits an accepting state more than twice. If we try go generalize this to claim that we may bound the number of visits to accepting states visited along any path in an accepting run of a Aco-BA we shall fail miserably. Consider a run of A_∞ on $abaabaaabaaaab\ldots$ and you will see that for each i there is a path in this DAG that visits the accepting state q at least i times. Yet, there is no path with infinitely many visits to q.

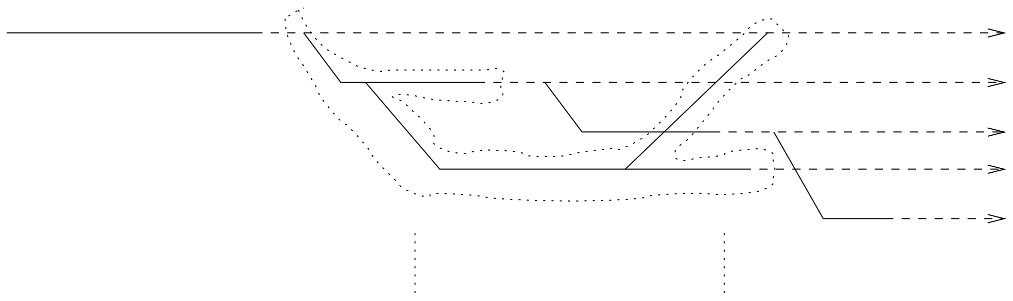


This reason why we cannot slice the run into two with the suffix staying entirely within $Q \setminus F$, is because we are dealing with a DAG and not a path. Even though a path within the DAG may have settled down to staying within $Q \setminus F$, there may be other branches that lead out from this path that visit F. Now, let us consider any such branch. Any extension of such a branch would also settle down to staying within $Q \setminus F$, but once again there may be other branches that lead out that visit final states and so on.



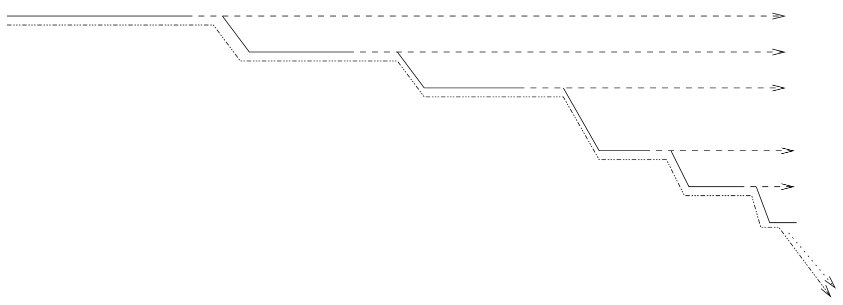
In this figure, the dashed lines correspond to suffices of paths that do not visit states in F. The top line branches after it settles into its $Q \setminus F$ phase to the second line, which in turn branches into the third line after settling into its $Q \setminus F$ phase and so on. We shall call such a branch from a $Q \setminus F$ suffix to a segment that visits F to be a bad branch. Once we enter a bad branch (i.e. a segment that visits F) we stay within the SAME bad branch as long as we do not reach a point from where there is a $Q \setminus F$ path leading out.

To be precise: First let us say that a vertex is *good* if there is an infinite path leading out from the vertex that stays entirely within $Q \setminus F$. Thus any good vertex must be in $Q \setminus F$. We call a vertex *bad* if it is not good. The set of good vertices will divide the set of bad vertices into groups and it is these groups that we refer to as bad branches. Two bad vertices belong to the same bad branch if we can reach one from the other without visiting any good vertices, using the edges of the DAG in either direction.



In the above figure there are 3 bad branches and the largest of the three is indicated by an enclosing dotted line.

Can any path through the run DAG visit infinitely many bad branches? Of course not, for if that were the case there would also be an infinite path that that visits F infinitely often. (Well, this needs proof, but take it on faith for the moment!)



So any path visits only a finite number of bad branches. From the assertion that every path visits F finitely often we have moved to the assertion that any path visits finitely many bad branches.

Suppose that there is a bound K for the longest sequence of such branchings among all accepting runs of A. With this assumption I claim that we can now translate the automaton A into an equivalent Büchi automaton that uses $2K + 1$ copies of the A. I shall sketch the construction in words here and a more formal presentation is made a little later.

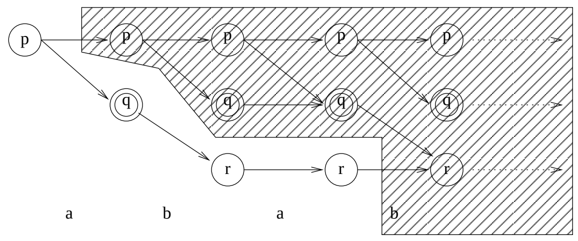
The idea is to duplicate the construction used in the nondeterministic case, but with $2K + 1$ copies. The copies $0, 2, \ldots 2K$ have an entire copy of the automaton A. The copies $1, 3, 5 \ldots 2K + 1$ have only copies of states in $Q \setminus F$. Transitions are set up so that one could either move at the same level or to the next level (or for that matter any lower level.) Note that now we are dealing with alternating automata, and so the effect of a transition is a set of states (that satisfy $\delta(q, a)$). We are free to choose which subset of this set is to stay at the same level and which subsets move to each of the higher numbered levels.

We start at the state s in level 0. We simulate the accepting run of the alternating automaton in the Büchi automaton by moving down to the next odd level from a even level whenever the current path has an extension that never visits F and moving down from an odd level to the next even level on a branch that visits final states. The fact that the number of bad branchings along any path is bounded by K ensures that we have enough levels to go down. Moreover, since every path eventually visits no more bad branches, it settles down in some odd level. Thus, in the simulation of an accepting run of the Alternating co-Büchi automaton by this alternating automaton, every path in the run settles down in some odd numbered level. On the other hand, in the simulation of a nonaccepting run, there must be at least one path that settles down in an even numbered level (since odd numbered levels do not have any F labelled vertices). Thus, we have constructed a Alternating Büchi automaton (with the set of vertices at odd numbered levels as the accepting set) that accepts the same language as the Alternating co-Büchi automaton we started with.

2.2 The Kupferman-Vardi Construction

Let G be an accepting run (DAG) of the alternating co-Büchi automaton A on some work $a_1a_2\ldots a_i\ldots$. We shall associate a number $\mathsf{rank}(v)$ with each vertex v in this DAG. It measures the maximum number of bad branches along any path starting at this vertex. Clearly, if there is a path from v to v' then $\mathsf{rank}(v) \geq \mathsf{rank}(v')$. Since G is a DAG computing such a rank is sort of similar to a topological sorting of this graph starting at the "leaves" and that is what we do. Leaves in our setting will correspond to vertices from where one cannot visit bad branches. These are vertices from where every path stays within $Q - F$.

In any DAG G', for any vertex v we write $\mathcal{I}_{G'}(v)$ to mean the sub-DAG rooted at v.

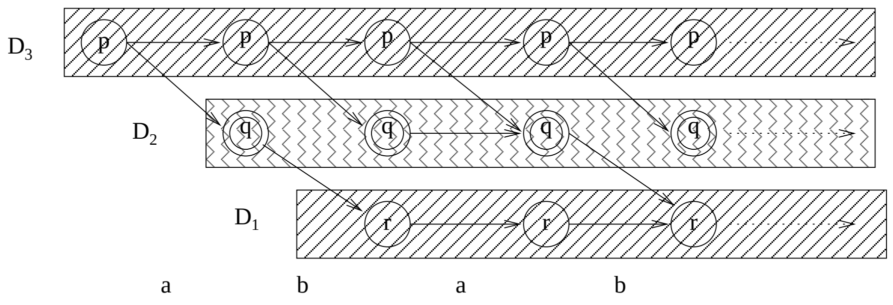


The ideal defined by $(p, 2)$ in the accepting run of A on $abab\ldots$.

We define two sequences of vertices $G_0, G_1, \ldots$ and $D_0, D_1, \ldots$. We shall refer to these sets also as DAGs, to mean the induced sub-DAG on these vertices.

$$\begin{array}{lcl} G_0 & = & G \\ & & \\ D_{2i} & = & \{v \mid \mathcal{I}_{G_{2i}}(v) \text{ is finite}\} \\ D_{2i+1} & = & \{v \mid \mathcal{I}_{G_{2i+1}} \text{ has no vertices from } F\} \\ & & \\ G_{i+1} & = & G_i \setminus D_i \end{array}$$

Thus, at even numbered levels we remove vertices that have only a finite number of reachable vertices. In odd numbered levels we remove a vertex v if the subDAG rooted at v has no vertices in F. Here is the breakup of the accepting run of A on $abab\ldots$.



Note that any vertex in $D_1, D_3, \ldots$ must be a good vertex. Vertices in $D_2, D_3 \ldots$ are bad vertices. D_0 is an anamoly as it might pick up vertices that have only finitely many successors but none of those successors may be in F. We will assume that D_0 is empty (equivalently that $\delta(q,a) \neq \mathsf{false}$ for any q, a, which can always be arranged).

In D_1 we pick out all the vertices in the DAG for which the subtree below contains no vertices in F. That is, the vertices in the DAG for which any path starting at any of these vertices stays within $Q \setminus F$ and thus visits NO bad branches.

In D_2 we pick up all the vertices that form parts of bad branches which lead into D_1. Thus starting from any such vertex we can visit at the most one bad branch before settling into D_1. In D_3 we pick up good vertices such that any bad branch reachable from these vertices lead into D_1. Thus any path starting at these vertices visits at most one bad branch along the way.

In D_4 we pick out bad vertice that form parts of bad branches that lead into D_3, D_2 and D_1. In D_5 we pick up good vertices that lead into $D_4, D_3, \ldots.$ Thus any path leading out from such a vertex may visit at the most two bad branches (one from D_4 and one from D_2) and so on.

Thus, for any vertex in $D_{2i} \cup D_{2i+1}$ we may visit at the most i bad branches along any path. We now show that G_{2n} is empty. Thus every vertex in G belongs to one of $D_1, D_2 \ldots D_{2n-1}$. Thus no path in the run-DAG visits more than n bad branches.

The following facts are quite easy to check:

1. Any path through vertices in D_{2i} is finite. This follows from the definition of D_{2i}.

2. If G_{2i+1} is nonempty then it is infinite. As a matter of fact if $v \in G_{2i+1}$ then there is an infinite path starting at v.

3. If G_{2i} is infinite then G_{2i+1} is nonempty (and hence infinite). This follows from the fact (using König's Lemma) whenever G_{2i} is infinite, it must have an infinite path and all the vertices in this infinite path will appear in G_{2i+1}.

4. For each i if G_{2i+1} is non-empty then D_{2i+1} is nonempty.
   Proof: We know that if G_{2i+1} is nonempty then it is infinite. If D_{2i+1} is empty then

every vertex v in G_{2i+1} has some vertex labelled by F in $\mathcal{I}_{G_{2i+1}}(v)$. It is quite trivial to conclude via König's Lemma that then there must be a path visiting F labelled vertices infinitely often in G_{2i+1}. This contradicts the fact that we started with an accepting run. ■

5. If $v \in D_{2i+1}$ then every node in $\mathcal{I}_{G_{2(i+1)}}(v)$ is in D_{2i+1}.

Let the *width* of a level in a DAG be the number of vertices at that level.

Lemma 4 *For each i, there is an N_i such that for each $j \geq N_i$ the width of level j in G_{2i} is bounded by $n - i$.*

Proof: By induction on i. The base case with $i = 0$ is trivially true. Suppose the result holds for G_{2i}. If this graph is finite then G_{2i+1} and G_{2i+2} are both empty and the result follows. Otherwise, this graph is infinite, which means G_{2i+1} is infinite and thus D_{2i+1} is non-empty. Thus, by 2 and 5 above, there is an N such that at least one vertex from all levels above N belongs to D_{2i+1}. Thus, there is an $N_{i+1} = Max(N, N_i)$ such that the width of all levels above N_{i+1} is at the most $n - i - 1$. ■

Thus, $G_{2n} = \emptyset$. We set $\mathsf{rank}(v) = i$ if $v \in D_i$. Thus if $\mathsf{rank}(v) = m$ then any path starting at v visits at the most $m/2$ bad branches. Moreover, $\mathsf{rank}(v) \leq 2n$ for any $v \in G$.

2.3 Constructing the equivalent alternating Büchi automaton

The automaton A' has $2n + 1$ copies, numbered $0, 1, \ldots 2n$, of the automaton A.

Let $A' = (Q', \Sigma, \delta', (s, 2n), F')$ where

$$\begin{array}{lcl} Q' & = & Q \times \{0, 2, 4, \ldots, 2n\} \cup (Q \setminus F) \times \{1, 3, \ldots, 2n-1\} \\ F' & = & (Q \setminus F) \times \{1, 3, \ldots, 2n-1\} \\ \delta'((q, i), a) & = & \delta(q, a) \otimes \{i, i-1, \ldots, 0\} \end{array}$$

where $\phi \otimes \{i, i-1, \ldots 0\}$ is the formula obtained by replacing each occurance of any state p in ϕ by $\bigvee_{j \geq i}(p, i)$.

Note the special structure of the transitions: They simulate the transitions of A, however one is free to choose to move to a lower numbered copy. Moreover, transitions either connect states in the same copy or go from a higher numbered copy to a lower numbered copy. In particular, there is no way a run can move from a lower numbered copy to a higher numbered copy. Thus, any path must eventually settle down in one of the copies $\{0, 1, \ldots, 2n\}$ (i.e. In any path, there is an infinite suffix that stays within one copy.)

Given an accepting run ρ of the Alternating co-Büchi automaton A on a word w, by computing the rank of each vertex v and labelling it by $(\rho(v), \mathsf{rank}(v))$ we get a run of the Alternating Büchi automaton A' on w. We claim that this run is also accepting. This is because, if some path settles down in some even numbered copy $2i$ that means that there is an infinite path in G of vertices in D_{2i} which is not possible. So it must settle down in some odd numbered level. Thus this run is accepting.

Further, given any run of A' on w, by simply erasing the second component, we get a run of A on w. Moreover, if this run was accepting then every path settles down in some fixed odd numbered level and thus every path stays within $Q \setminus F$ eventually. Thus we get an accepting run of A. Hence $L(A) = L(A')$. Thus we have the following theorem:

Theorem 5 *(Kupferman–Vardi) Let A be a alternating co-Büchi automaton with n states. Then the alternating Büchi automaton constructed above, with $O(n^2)$ states, accepts the same language.*

Since every run of A' eventually settles down in some level or the other it follows that the alternating co-Büchi automaton $A'' = (Q', \Sigma, \delta', (s, 2n), Q' \setminus F')$ accepts the same language as A and A'.

Definition 6 *(Muller-Schupp) An alternating automaton $A = (Q, \Sigma, \delta, s, F)$ is said to be a* weak alternating automaton *if there is a partial order $(P, <)$ and a map $\mathsf{rank} : Q \to P$ such that every state p that appears in $\delta(q, a)$, $\mathsf{rank}(p) \leq \mathsf{rank}(q)$ and $F = \lambda^{-1}(X)$ for some subset X of P.*

The construction described above has thus translated an Alternating co-Büchi automaton into an equivalent weak alternating automaton.

It is quite trivial to observe that the weak alternating Büchi automaton $A = (Q, \Sigma, \delta, s, F)$ accepts the same language as the weak alternating co-Büchi automaton $A = (Q, \Sigma, \delta, s, Q \setminus F)$. Thus, we have identified a subclass of alternating automata for which the translation from the co-Büchi acceptance to Büchi acceptance is trivial, and shown how to translate any alternating co-Büchi automaton into this class.

Putting together the theorems **??** and **??** we see that every alternating co-Büchi automaton can be translated to an equivalent nondeterministic Büchi automaton A'' of size $2^{O(n^2)}$ (its state space is $\{(X, Y) \mid Y \subseteq X \subseteq Q'\}$). Hence, we can complement any Büchi automaton with a state space blowup of at most $2^{O(n^2)}$.

A simple observation will allow us to reduce this complexity to $2^{O(nlogn)}$. The accepting run of A' constructed from an accepting run of A (described above) has a special property: At any level there is at the most one copy of any state. Thus, if X is any state reached in the level by level simulation of this automaton and $(p, i) \in X$ for some $p \in Q$, then $(p, j) \notin X$ for any $j \neq i$. Thus, we can restrict the first component of the states of A'' to be $\{X \subseteq Q' \mid \forall p.\ ((p, j) \in X \wedge (p, k) \in X) \Rightarrow j = k\}$. What is the size of this set? This size of this set is bounded by the product of the number of subsets of Q' ($2^{O(n)}$) and the number of maps that assign ranks (in the range $0, 1, \ldots 2n$) to elements of Q ($O(n!)$). That is, $O(2^{O(n)} n^{O(logn)}) = O(2^{O(nlogn)})$. As we shall see in the next lecture this is optimal.