ACTS Workshop
30th January 2009

Improved state-count for determinization of non-deterministic Buchi automata: A Safra-tree based approach

A conjecture

Hrishikesh Karmarkar and Supratik Chakraborty
IIT Bombay

Motivation

- Complementation of non-deterministic Büchi word automata - a well studied and important problem.
- Complementation can be carried out - directly or via determinization.
- Both complementation and determinization for NBW are non-trivial.
- Complementation upper bound has reduced continuously - current bound stands at $(0.8n)^n$ [Schewe09].
- Det. upper-bound more or less stable.
- Piterman06 reduced it to $2n(n!^2)$ - DPW with $2n$ indices. Schewe09 gives a determinization construction - DRW with $L.o(1.65n^n)$ states and $2^n - 1$ accepting pairs, where L is the size of the input alphabet.
- We have reason to believe that determinization upper bounds can match those of complementation.

Safra trees

- Our ongoing attempt uses Safra trees.
- Consider usual Safra trees.
- Has the following -
  1. Set of nodes with a designated root node.
  2. A node *naming* function that assigns names from a finite vocabulary to every node.
  3. A node *labelling* function that assigns sets of Büchi-states to every node.
  4. A *parenthood* function extendable to an *ancestor* relation.
  5. A *sibling* relation extendable to a *left* relation.
  6. A *coloring* function that colors every node with one of White/Red/Green.

Modified Safra trees (mS-trees).

- mS-trees are Safra trees with additional color in {Yellow, NotYellow} for nodes. Hence, the color for a node is a pair in $\{White, Red, Green\} \times \{Yellow, NotYellow\}$.
  Notation: A color from $\{White, Red, Green\}$ *is called the first-color of a node* (χ_1)*, while a color from* $\{Yellow, NotYellow\}$ *is called the second-color* (χ_2) *of a node.*

- Eventual goal: To reorder nodes such that all nodes that are removed finitely often appear to the left of or above all vertices that are removed infinitely often.

- To attain our goal : When a node is deleted move all nodes to its *right* to the left. This requires us to move nodes as children of nodes that did not spawn them.

- Semantics of color Yellow : If a node v has parent u and is also colored Yellow, then u *has not* spawned v.

- What if a node has only Yellow children and it becomes empty? Safra mechanism to pull back descendants does not work. We can move all Yellow nodes left.
- How far left can we move? The leftmost path perhaps.
- What if a node on the leftmost path has only Yellow children? Where do we move?
- Important Property of mS-trees: *mS-trees admit non-planarity*. This is required to make sure we can always move *left*.
- Non-planarity causes siblings to have no order. We need order. We need to embed back into a plane.
- In Safra trees, Büchi runs can move left or below by merging. The left and descendant relations dictate the merging.
- With reordering (called Horizontal Compression) we have another way of moving runs in addition to merging. How do we

manage merging in Yellow nodes?

- What is really required is a new way of coding nodes.

Codes for nodes in Safra trees

- A code for a node is a sequence $c = (i_0, i_1, i_2, \ldots, i_{k-1}, i_k)$, with each $i_j \in \mathbb{N}$.
- c is the code for node v in Safra tree if v can be reached from the root via a path that visits vertices $v_{i_0}, v_{i_1}, v_{i_2}, \ldots, v_{i_{k-1}}, v_{i_k}$.
- $v_{i_0} = root$ and $v_{i_k} = v$ and v_{i_j} is the ${i_j}^{th}$ child in left-right order of vertex $v_{i_{j-1}}$ for $j = 1, \ldots, k$.
- root has code 0 by default.
- Example: If v has code 012, then v can be reached by visiting the root, then the first child of the root and then the second child of this first child.
- A node that is to the left or below has lower code. Ex: 012 is lower than 013 and 012 is lower than 01. Merging happens by comparing codes and merging into lower code nodes.

Codes in mS-trees.

- We need codes that address problems caused due to Yellow nodes.
- Extend codes of Safra trees.
- Separate tree from codes. Example: A node to the rightmay have a lower code in a non-natural coding scheme.
- We need to find useful codes, not arbitrary ones.
- One such useful code *but that failed* is given.
- A sequence $c = (i_0, i_1, i_2, \ldots, i_{k-1}, i_k)$, with each $i_j \in \mathbb{N} \cup \{\#\}$.
- *Notation: node*(c)=*node with code c.*

  Suppose, *node*(c) is in plane numbered p. Then *node*$(c\#)$ is an immediate right-sibling of *node*(c) in plane numbered $p+1$. Similarly, *node*$(c\#^i)$ for $i > 1$ is the right-sibling of *node*(c) in plane numbered $p+i$.
- Example: Let $c = 12$, then code $c\# = 12\#$.

Then $node(12\#)$ is really a right-sibling of $node(12)$ in the new coding scheme, but $node(12\#)$ is in a higher plane than $node(12)$.

Under the usual coding scheme for Safra trees $node(12)$ and $node(13)$ are immediate siblings on the same plane. In the new coding scheme $node(13)$ is to the right of $node(12)$ and $node(12\#)$.

- Codes are still totally ordered.

Codes in mS-trees.

- Let $c_1 = c_0 n$ be a code.
  1. Right siblings of $node(c_1)$: $node(c_0 m \#^i)$, $m \geq n$, $i \geq 0$, but $m = n \Rightarrow i > 0$.
  2. Left siblings of $node(c_1)$: $node(c_0 m \#^i)$, $m < n$, $i \geq 0$.
  3. Children of $node(c_1)$: $node(c_1 m)$, $m > 0$.
  4. Parent of $node(c_1)$: $node(c_0)$.
- Let $c_1 = c_0 n \#^i$ be a code, with $i, n > 0$.
  1. Right siblings of $node(c_1)$: $node(c_1 \#^j)$, $j > 0$ and $node(c_0 m \#^j)$, $m > n$, $j \geq 0$.
  2. Left-siblings of $node(c_1)$: $node(c_0 n \#^j)$, for $j < i$ and $node(c_0 m \#^j)$, for $m \leq n$, $j \geq 0$ but $m = n \Rightarrow j < i$.
  3. Children of $node(c_1)$: $node(c_1 m)$, $m > 0$.
  4. Parent of $node(c_1)$: $node(c_0)$.

Ordering over codes.

- Let c_1, c_2 be two codes in an mS-tree. Let a_1, a_2 be the first symbols reading from left to right where c_1, c_2 differ.
- If c_1, c_2 are not prefixes of one another then, we say $c_1 \leq c_2$ if
  1. $a_1, a_2 \in N$, $a_1 \leq a_2$ (L_1)
     Ex: $c_1 = 12$, $c_2 = 13$.
  2. $a_1 \in N$, $a_2 = \#$ (L_2)
     Ex: $c_1 = 123$, $c_2 = 12\#$.
- If c_1 is a prefix of c_2, then $c_1 \leq c_2$ if
  1. $c_2 = c_1\#c_3$, c_3 is possibly ϵ. (L_3)
     For this $node(c_2)$ is the descendant of a right-sibling of $node(c_1)$.
- This naturally defines the *LEFT* relation between nodes i.e c_1 is to the left of c_2 if $c_1 \leq c_2$ by the above criteria.

Ordering over codes.

- If c_2 is a prefix of c_1 then, we say $c_1 \leq c_2$ if
  1. $c_1 = c_2 n c_3$, $n \in \mathbb{N}$ and c_3 is possibly ϵ. (D_1)
     For this, $node(c_1)$ is indeed a descendant of $node(c_2)$.
  2. This naturally defines the $ANCESTOR$ relation between nodes i.e c_2 is an ancestor of c_1 if $c_1 \leq c_2$ by the above criteria.

A natural node naming scheme.

Notation: $Name(n)$ is the name assigned to node n, where $1 \leq Name(n) \leq I$, and I is the total number of nodes in the mS-tree.

- For all pairs of nodes n_1, n_2 such that $n_1 \neq n_2$, let $code(n_1) = c_1$ and $code(n_2) = c_2$.
- If $c_1 \leq c_2$ by any one of criteria L_1, L_2, L_3 then $Name(node(c_1)) \leq Name(node(c_2))$.
- If $c_1 \leq c_2$ by criteria D_1, then $Name(node(c_2)) \leq Name(node(c_1))$.
- The solution for the above constraints gives a unique naming scheme for mS-trees.
- The obtained naming scheme is nothing but a preorder naming scheme i.e if node n is the i^{th} node visited in a preorder traversal of the mS-tree then node n gets name i.

The determinization construction

- Given a NBW $B = (\Sigma, Q^b, Q^b_0, \delta^b, T^b)$ with $|Q^b| = n$, we construct a DPW $R = (\Sigma, Q^r, q^r_0, \delta^r, T^r)$ that accepts the same language.
- States of R are mS-trees with the node labels disjoint subsets of Q^b.
- Additionally with every state we store two variables e, f that are used to define the acceptance condition.
- t_0 be the mS-tree consisting of just the root node v_0 with node label Q^b_0 and color (White, NotYellow). The initial state of R is $q^r_0 = t_0$.
- The transition function of DPW R is computed using procedure *NextState*. Given a state $t \in Q^r$ and a letter $a \in \Sigma$, invoking *NextState* with (t, a) gives $t' = \delta^r(t, a)$.

  Notation: $\lambda(v)$ is the (Büchi state) label of node v.

The determinization construction

Notation: A Yellow/NotYellow node is a node with χ_2 Yellow/NotYellow.

Procedure NextState (t, a)

1. For the label of every node in t, compute the next Büchi states via δ^b. Set χ_1 of all nodes to White.
2. For every Büchi state s in the tree nodes, find the leftmost and lowermost node that contains s. For this order all nodes containing s with respect to the $\leq$ ordering on their codes and find the node v with the lowest code. Keep s in v and remove it from labels of all other nodes.
3. For every node v if $\lambda(v) \cap T^b \neq \emptyset$, create a new node v' with $\lambda(v') = \lambda(v) \cap T^b$ and add it as the rightmost child of v on the same plane as v. Hence, if code of v is c, then code of v' is cm for some $m \in \mathbb{N}$. Set the first color of v' to Red.

The determinization construction

4. If $\lambda(v) = \emptyset$ and v has atleast one NotYellow child then
   (a) For every NotYellow descendant u of v reachable via a path consisting of only NotYellow nodes, set $\lambda(v) = \lambda(v) \cup \lambda(u)$ and set $\lambda(u) = \emptyset$.
   (b) Set χ_1 of v to Green.

   Notation: The leftmost path in an mS-tree is the path consisting of only vertices with codes in 01^*.

5. For every node v, in reverse preorder sequence, not on the leftmost path and such that $\lambda(v) = \emptyset$
   (a) Collapse entire subtree of v and add all Büchi states to a new node v'. Color v' with (Red, Yellow).
   (b) Let u be the rightmost and lowermost node (using the $\leq$ ordering on codes) among all nodes to the left of v and let u'

be the parent of u.

(c) If u' is not an ancestor of v, then add v' as the rightmost child of u' *on the same plane as* u'. Hence, if the code of u' is c', then the code of v' is $c'm$ for some $m \in \mathbb{N}$.

(d) If u' is an ancestor of v, then add v' as a child of u' and as the immediate right-sibling of u on the *next plane as* u. Hence, if the code of u is c then the code of v' is $c\#$.

(e) Compress (reorder) mS-tree by the following steps

   i. Let $u_0, u_1, u_2, \ldots, u_k$ be vertices such that $u_0 = v$ and $u_k = root$ and u_{i+1}=parent of u_i for all $i \in \{0, 1, \ldots, k-1\}$.

   ii. Delete vertex v.

   iii. For each $i \in \{1, \ldots, k-1\}$, if u_i has an immediate right sibling r_i then set u_i to be the parent of r_i (from its original parent u_{i+1}). Set χ_2 of r_i to Yellow.

6. Among all nodes on the leftmost path let v be the node with the largest code such that first-color of v is Green. Then -
   (a) Collapse the entire subtree of v into a single node v'.
   (b) Add v' as the rightmost child of the root *on the same plane* as the root.
7. *Set e,f variables* : Let t' be the mS-tree obtained after all the above steps have been completed. We set the e, f variables.
   (a) If the vertex in preorder position i of t' is not named i, then the vertex in preorder position i is called *Red*.
   (b) Set f to the minimum j such that the vertex in preorder position j is called *Green*, else if no such j exists then f is set to $2n + 1$.
   (c) Set e to the minimum j such that the vertex in preorder position j is called *Red*, else if no such j exists then e is set to $2n + 1$.

(d) Set the name of the vertex in preorder position i in t' to i, for $i \in \{1, \ldots, n\}$. Return (t')

The parity acceptance condition is defined by by setting up the following sets
$F_0 = \{q \in Q^r | f = 1 \text{ and } e > 1\}$
$F_{2n-1} = Q^r \setminus \bigcup_{j=0}^{2n-2} F_j$
$F_{2i+1} = \{q \in Q^r | e = i + 2 \text{ and } f \geq e\}$
$F_{2i+2} = \{q \in Q^r | f = i + 2 \text{ and } e > f\}$ where $i = 0, \ldots, n - 2$.

A run of R is accepting if there is an even j such that states from F_j are seen infinitely often, while states from F_i for all $i < j$ are seen finitely often.

Correctness

Theorem 1 *On a word α, if there is an even j such that states from F_j are seen infinitely often, while states from F_i for all $i < j$ are seen finitely often then there is a Büchi accepting run in B.*

1. Statement of theorem implies the existence of a node v that is Green infinitely often and Red finitely often along the run of states (mS-trees) of R on α.

2. Proof of theorem now follows Safra's original proof exactly. Construct finitely branching infinite tree and use König's Lemma.

Correctness

Theorem 2 *One a word α, if there is an accepting Büchi run in B, then there is an even j such that states from F_j are seen infinitely often, while states from F_i for all $i < j$ are seen finitely often.*

Let ρ be the accepting Büchi run. We break the proof into two parts.

(I) In the mS-trees along the run ρ of R, suppose the leftmost path has a node that turns Green infinitely often.

1. Look at the highest such node on the leftmost path, say v.
2. After a sufficiently long finite prefix of ρ, no node above v turns Green.
3. The only way in which a node on the leftmost path can be deleted if its ancestor is Green.
4. Hence, v is Red finitely often and its code and its preorder position eventually get fixed.

(II) In the mS-trees along the run ξ of R, suppose no node on the leftmost path turns Green infinitely often.

1. After a sufficiently long finite prefix of ξ, after which no node on the leftmost path turns Green, let us trace the movement of ρ.
2. The run ρ cannot move to a child of the root after this finite prefix.
3. Hence, ρ can move to a node to the left or below the current node and it can only move up into an ancestor that is colored Green.
4. Since, no node on the leftmost path is Green now, the run ρ always stays to the right of the leftmost path.
5. Look at the codes of all nodes that contain ρ after the finite prefix. Let c be the longest common prefix of these codes.
6. Since, ρ cannot move left and down indefinitely it must move back up infinitely often and it moves up into an ancestor that is

colored Green.

7. But, since c is the longest common prefix of all codes that ρ is in, the run ρ must come back to the node of c infinitely often.

8. Hence the node of c is Green infinitely often.

9. If a node to the left or above the node of c is deleted, it would result in the node of c being moved left and c cannot be the longest common prefix of codes then.

10. Hence, the code and preorder position of the node with code c is fixed and since ρ is always in the subtree of the node with code c, it is Red finitely often.

The existence of a node that is Green infinitely often and Red finitely often and whose preorder position is fixed implies that there is an even j such that states from F_j are seen infinitely often, while states from F_i for all $i < j$ are visited only finitely often.

Size of parity automaton

1. Nodes in every mS-tree are named according to their preorder position.
2. This does away with the need to store node names.
3. We store only the Büchi states for each node of an mS-tree in preorder, suitable separated (n^n with a multiplier of k^n for the separator information, for constant k).
4. Information regarding structure of the tree ($2^{(2n+1)}$ multiplier).
5. Node colors (6^n multiplier).
6. e,f variables ($2n$ multiplier)
7. Number of states of the deterministic parity automaton is upper bounded by (cn^n) for a suitable constant c, with $2n$ parity sets.