ABSTRACT

A deterministic automaton is defined by a tuple $A = \langle S, \Sigma, D, \delta \rangle$ where

- $S$ is a finite set of states,
- $\Sigma$ is a finite alphabet consisting of $p$ input letters,
- $D \subseteq S \times \Sigma$ is called the domain,
- $\delta : D \rightarrow S$ is a transition function.

When $D = S \times \Sigma$, then $A$ is called complete, otherwise $A$ is called partial.

A synchronizing sequence $w$ for an automaton $A$ is a sequence of inputs such that without knowing the current state of $A$, when $w$ is applied to automaton $A$, $A$ will have a final particular active state at the end (Eppstein, 1990).

An automaton $A$ is called strongly connected if every state is reachable from every other state by using a sequence of inputs. Otherwise, $A$ is called non-strongly connected.

When an automaton is non-strongly connected, it can be represented as a group of strongly connected automata, called strongly connected components (SCCs).

Given a non-strongly connected automaton $A$, we suggest a method to build a synchronizing sequence for $A$ by using the synchronizing sequences of the SCCs of $A$.

PROJECT DETAILS

• A complete automaton $A = \langle S, \Sigma, D, \delta \rangle$ can be decomposed into a set of SCCs $\{A_1, A_2, \ldots, A_j\}$, where $A_i = \langle S_i, \Sigma, D_i, \delta_i \rangle$, for $1 \leq i \leq j$, such that for all $1 \leq i < j \leq k$, $S_i \cap S_j = \emptyset$, and $S_1 \cup S_2 \cup \ldots \cup S_j = S$.

• An SCC $A_i = \langle S_i, D_i, \delta_i \rangle$ is called a sink component if $D_i = S_i \Sigma$. In other words, for a sink component all the transitions of the states in $S_i$ in $A$ are preserved in $A_i$.

• Let $A = \langle S, \Sigma, D, \delta \rangle$ be an automaton and $\{A_1, A_2, \ldots, A_j\}$ be the SCCs of $A$. We consider the SCCs of $A$ sorted as $\{A_1, A_2, \ldots, A_j\}$ such that for any $1 \leq i < j \leq k$, there do not exist $A_i \times S_j \in S \Sigma$ where $R(S_i, w) = S_j$.

Theorem: Let $A = \langle S, \Sigma, D, \delta \rangle$ be a complete automaton and $\{A_1, A_2, \ldots, A_j\}$ be the sorted SCCs of $A$, where $A_i = \langle S_i, D_i, \delta_i \rangle$. For $1 \leq i \leq k$, let $A_i$ be the completion of $A_i$, $S' = S_i \cap \delta(S_i, \sigma_{j+1})$ and $\alpha_i$ is a $S_i$-synchronizing sequence for $A_i$. Then $\sigma_{j+1}\alpha_i$ is a synchronizing sequence for $A$.

CONCLUSIONS

• Our method can be used with any synchronizing heuristic to make it work faster on non-strongly connected automata.

• The improvement in the speed increases as the number of SCCs increases.

• In case of Greedy, our method can find shorter synchronizing sequences in shorter time compared to the application of Greedy directly to automata.

• SynchroP finds shorter synchronizing sequences compared to Greedy but it takes more time. With our method, we can use SynchroP to find shorter synchronizing sequences than Greedy in a shorter time than Greedy as the number of SCCs increase.

• Greedy requires $O(n^k)$ and SynchroP requires $O(n^k)$ time, where $n$ is the number of states.

• If there are $k$ strongly connected components with equal sizes, the complexity of Greedy and SynchroP applied with our method becomes $O(kn^k)$ and $O(kn^k)$ respectively.

REFERENCES


