1.2 Pointer Machines

The memory configuration of a ***Pointer Machine (PM)***, called ***pointer graph***, is a finite directed labeled multigraph. One node R is marked as ***root*** and has directed paths to all nodes. Nodes can ***see*** and change the configuration of their out-neighborhood of constant (2 suffices) depth. Edges (***pointers***) are labeled with ***colors*** from a finite alphabet common to all graphs handled by a given program. The pointers coming out of a node must have different colors (which bounds the outdegree). Some colors are designated as ***working*** and not used in inputs/outputs. One of them is called ***active***, as also are pointers carrying it and nodes seeing them. Active pointers must have inverses, form a tree to the root, and can be dropped only in leaves.

All active nodes each step execute an identical ***program***. At its first ***pulling*** stage, node A acquires copies of all pointers of its children using "composite" colors: e.g., for a two-pointer path (A, B, C) colored x, y, the new pointer (A, C) is colored xy, or an existing z-colored pointer (A, C) is recolored $\{z, xy\}$. A also spawns a new node with pointers to and from it. Next, A transforms the colors of its set of pointers, drops the pointers left with composite colors, and vanishes if no pointers are left. Nodes with no path from the root are forever invisible and considered dropped. The computation is initiated by inserting an active loop-edge into the root. When no active pointers remain, the graph, with all working colors dropped, is the output.

Exercise: Design a PM transforming the input graph into the same one with two extra pointers from each node: to its parent in a BFS spanning tree and to the root. Hint: Nodes with no path *to* the root can never be activated. but can be copied with pointers, copies connected to the root, the original input removed.

PM can be ***parallel, PPM*** [Barzdin', Kalnin's 74] or ***sequential, SPM***. SPM differ in that only pointers to the root, their sources, and nodes that have pointers with inverses to these sources can be active.

A ***Kolmogorov*** or ***Kolmogorov-Uspenskii*** Machine (KM) [Kolmogorov, Uspenskii 58], is a special case of Pointer Machine [Schoenhage 80] with the restriction that all pointers have inverses. This implies the bounded in/out-degree of the graph which we further assume to be constant.

Fixed Connection Machine (FCM) is a variant of the PKM with the restriction that pointers once created *cannot* be removed, only re-colored. So when the memory limits are reached, the pointer structure freezes, and the computation can be continued only by changing the colors of the pointers.

PPM is the most powerful model we consider: it can simulate the others in the same space/time. E.g., cellular automata make a simple special case of a PPM which restricts the Pointer Graph to be a grid.

Exercise. Design a machine of each model (TM, CA, KM, PPM) which determines if an input string x has a form ww, $w \in \{a, b\}^*$. Analyze time (depth) and space. KM/PPM takes input x in the form of colors of edges in a chain of nodes, with root linked to both ends. The PPM nodes also have pointers to the root. Below are hints for TM,SPM,CA. The space is $O(\|x\|)$ in all three cases.

Turing and Pointer Machines. TM first finds the middle of ww by capitalizing the letters at both ends one by one. Then it compares letter by letter the two halves, lowering their case. The complexity is: $T(x) = O(\|x\|^2)$. SPM acts similarly, except that the root keeps and updates the pointers to the borders between the upper and lower case substrings. This allows constant time access to these borders. So, $T(x)=O(\|x\|)$.

Cellular Automata. The computation starts with the leftmost cell sending right two signals. Reaching the end the first signal turns back. The second signal propagates three times slower, so they meet in the middle of ww and disappear. While alive, the second signal copies the input field i of each cell into a special field c. The c symbols will try to move right whenever the next cell's c field is blank. So the chain of these symbols alternating with blanks will start moving right from the middle of ww. Upon reaching the end they will push the blanks out and pack themselves back into a copy of the left half of ww shifted right. When a c symbol does not have a blank at the right to move to, it compares itself with the i field of the same cell. If they differ, a signal is generated which halts all activity and rejects x. If all comparisons are successful, the last c generates the accepting signal. The depth is: $T(x) = O(\|x\|)$.