3.3 Reductions; Non-Deterministic and Alternating TM; Time and Space

To ***reduce*** (see definition in sec. 2.2) Halting game to Linear Chess we introduce a few concepts.

A ***non-deterministic*** Turing Machine (NTM) is a TM that sometimes offers a (restricted) transition choice, made by a ***driver***, a function (of the TM configuration) of unrestricted complexity. A deterministic (ordinary) TM M accepts a string x if $M(x)$=yes; an NTM M does if there exists a driver d s.t. $M_d(x)$=yes. NTM represent single player games – puzzles – with a simple transition rule, e.g., Rubik's Cube.
One can compute the winning strategy in exponential time by exhaustive search of all d.

Home Work: Prove all such games have P-time winning strategies, or show some have not.
Will get you grade A for the course, $1,000,000 Award and a senior faculty rank at a school of your choice.

The ***alternating*** TM (ATM) is a variation of the NTM driven by two alternating drivers (players) l, r. A string is accepted if there is l such that for any $r : M_{l,r}(x)$= yes. Our games could be viewed as ATM returning the result of the game in linear space but possibly exponential time, M prompts l and r alternatingly to choose their moves (in several steps for multi-bit moves) and computes the resulting position, until a winner emerges. Accepted strings describe winning (i.e. having a winning strategy) positions.

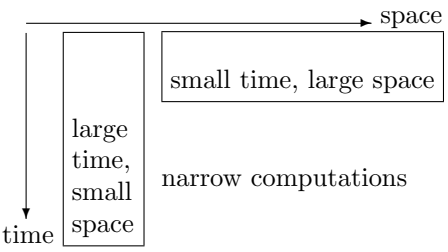
Linear Chess (LC) Simulation of TM-Games. The simulation first represents the Halting Game as an ATM computation simulated by the Ikeno TM (2.1) (using the "A/B" command for players' input). The UTM is viewed as an array of 1-pointer cellular automata: Weak cells as rightward, Shy leftward. Upon termination, the TM head is set to move to the end of the tape, eliminating all loser pieces.

This is viewed as a game of ***1d-Chess (1dC)***, a variant of LC, where the table, not the "Gender Rule", determine the victorious piece, and not only the vanquished piece is replaced, but also the winning piece may be "promoted to" another type of the same side. The types are states of Ikeno TM showing Loyalty (pointer direction) $d \in \{W, S\}$, gender g (= previous d), and 6/6/6/5 ranks (trit $t \in \{0, 1, *\}$ with $'$ bit p).

Exercise: Describe LC simulation of 1dC. **Hint:** Each 1dC transition is simulated in several LC stages. Let L,R be the two pieces active in 1dC. In odd stages L (in even stages R) changes gender while turning pointer twice. The last stage turns pointer only once and possibly changes gender. In the first stage L appends its rank with R's p bit. All other stages replace old 1dC rank with the new one. R appends its old t bit (only if $t \neq *$) to its new rank. Subsequent stages drop both old bits, marking L instead if it is the new 1dC head. Up to 4 more stages are used to exit any mismatch with 1dC new d, g bits.

Space-Time Trade-off. ***Deterministic*** linear space computations are games where any position has at most one (and easily computable) move. We know no general superlinear lower bound or subexponential upper bound for time required to determine their outcome. This is a big open problem.

Recall that on a parallel machine: ***time*** is the number of steps until the last processor halts; ***space*** is the amount of memory used; ***volume*** is the combined number of steps of all processors. "***Small***" will refer to values bounded by a polynomial of the input length; "***large***" to exponential. Let us call computations ***narrow*** if *either* time *or* space are polynomial, and ***compact*** if both (and, thus, volume too) are. **An open question:** Do all exponential volume algorithms (e.g., one solving Linear Chess) allow an equivalent *narrow* computation?



Alternatively, can every *narrow* computation be converted into a compact one? This is equivalent to the existence of a P-time algorithm for solving any ***fast*** game, i.e. a game with a P-time transition rule and a move counter limiting *the number of moves* to a polynomial. The sec. 3.1 algorithm can be implemented in parallel P-time for such games. Converse also holds, similarly to the Halting Game.

[Stockmeyer, Meyer 73] solve compact games in P-space: With $M \subset \{0, 1\}$ run depth-first search on the tree of all games – sequences of moves. On exiting each node it is marked as the active player's win if some move leads to a child so marked; else as his opponent's. Children's marks are then erased. Conversely, compact games can simulate any P-space algorithms. Player A declares the result of the space-k, time-2^k computation. If he lies, player B asks him to declare the memory state in the middle of that time interval, and so by a k-step binary search catches A's lie on a mismatch of states at two adjacent times. This has some flavor of trade-offs such as saving time at the expense of space in dynamic programming.

Thus, fast games (i.e. compact alternating computations) correspond to narrow deterministic computations; general games (i.e. narrow alternating computations) correspond to large deterministic ones.