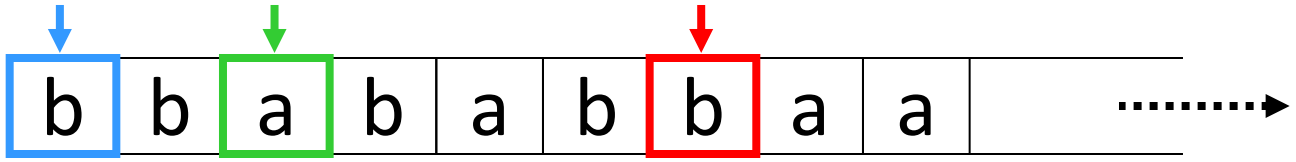
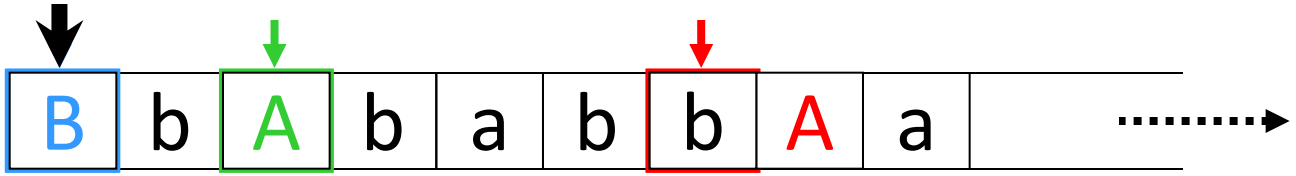
CS3102 Theory of Computation

Turing Machine "Enhancements"

Multiple heads:



Idea: Mark heads locations on tape and simulate

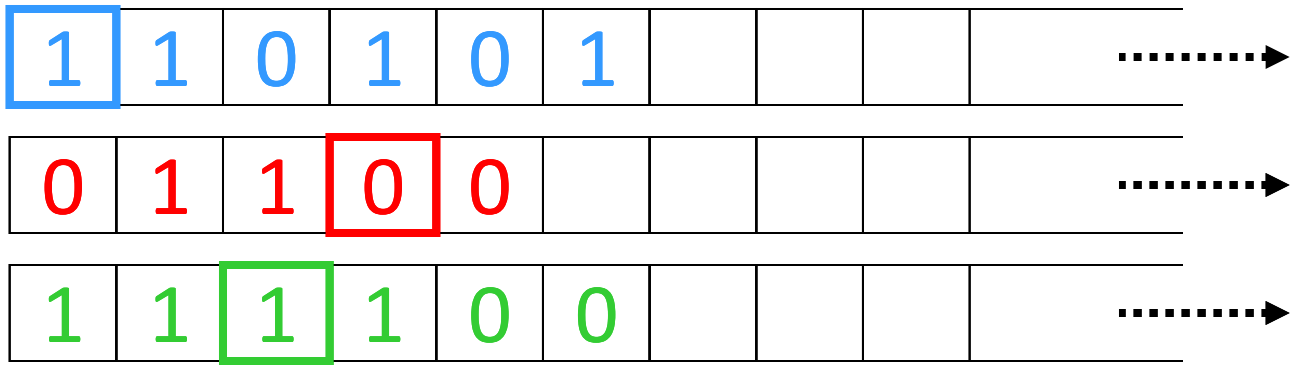


Modified δ' processes each "virtual" head independently:

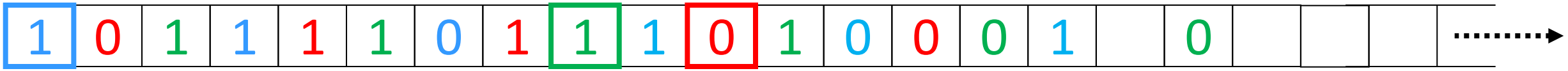
- Each move of δ is simulated by a long scan & update
- δ' updates & marks all "virtual" head positions

Turing Machine "Enhancements"

Multiple tapes:



Idea: Interlace multiple tapes into a single tape

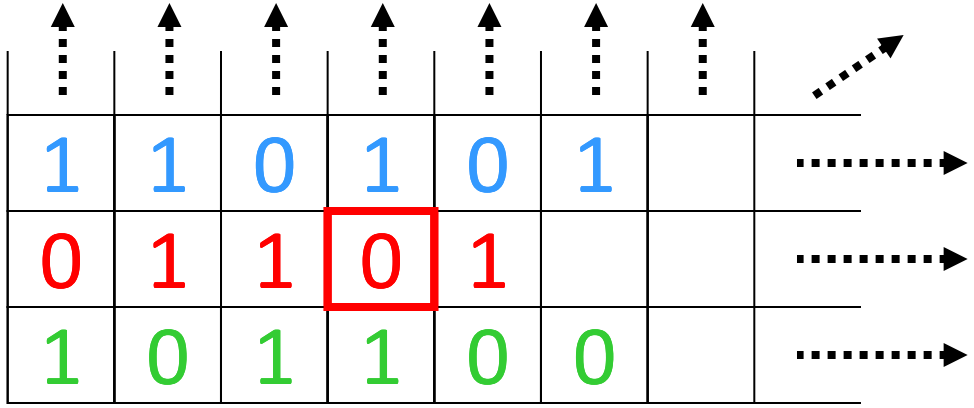


Modified δ' processes each "virtual" tape independently:

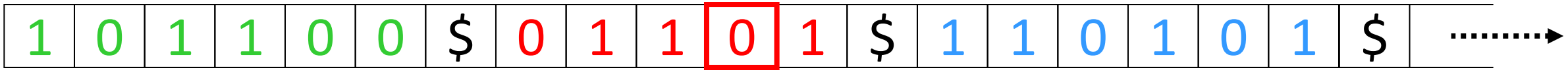
- Each move of δ is simulated by a long scan & update
- δ' updates R/W head positions on all "virtual tapes"

Turing Machine "Enhancements"

Two-dimensional tape:



Idea: Flatten 2-D tape into a 1-D tape

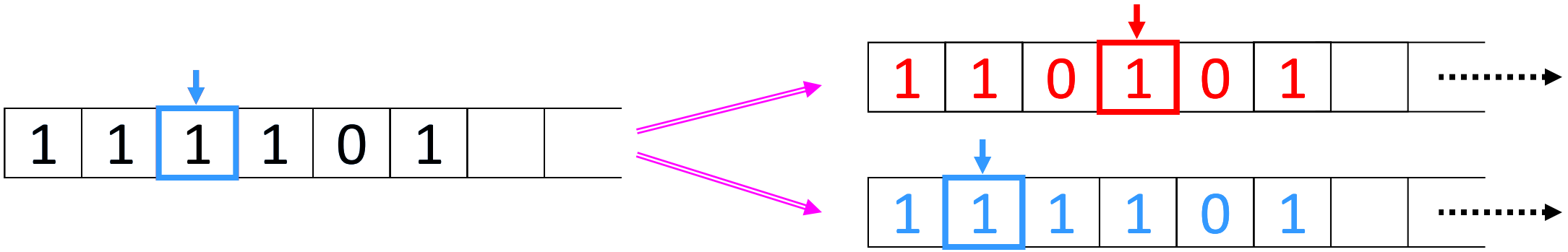


Modified 1-D δ' simulates the original 2-D δ:

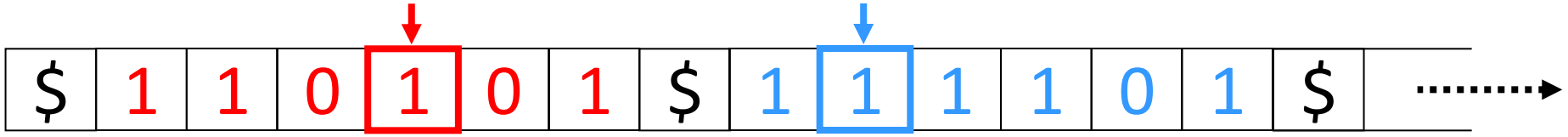
- Left/right δ moves: δ' moves horizontally
- Up/down δ moves: δ' jumps between tape sections

Turing Machine "Enhancements"

Non-determinism:



Idea: Parallel-simulate non-deterministic threads



Modified deterministic δ' simulates the original ND δ:

- Each ND move by δ spawns another independent "thread"
- All current threads are simulated "in parallel"

Enumerators

- An enumerator for language L is a TM which prints all strings in L onto its tape
- Lexicographic enumerator
  - Prints them in lexicographic order
- A language is recognizable if and only if it has an enumerator
- A language is decidable if and only if it has a lexicographic enumerator

Enumerable = Recognizable

- Let's assume that I have an enumerator for the language. How can I build a recognizer for that language?
  - To recognize a language, I need a turing machine that, for input w, it will accept w if it belongs to the language
  - Each time the enumerator prints a string, check if it's w. If so, then accept
- Let's assume that I have a recognizer for the language. How can I build a enumerator for that language?
  - First try recognizing the empty string. If that is accepted, print it
  - Next try the next string. If that is accepted, print it.

Lexicographically Enumerable = Decidable

- If I can enumerate in order, how can I decide a string?
  - For each string that's printed, if it is w then accept
  - If it is lexicographically greater than w (longer than w), then reject
- If I can decide the language, how can I lexicographically enumerate it?
    - First try deciding the empty string. If that is accepted, print it
    - Next try the next string. If that is accepted, print it.

Closure properties of Recognizable

- Closed under:
  - Union
  - Intersection
  - Concatenation
  - Kleene
- Not closed under:
  - Complement

Not closed under Complement

- If L and $\bar{L}$ are both recognizable, then L is decidable.
- To determine if $w \in L$:
  - Run w on recognizer for L for 5 steps
  - Run w on recognizer for $\bar{L}$ for 5 steps
  - Repeat until one of them accepts

Not closed under Complement

- If L and $\bar{L}$ are both recognizable, then L is decidable.
- To determine if $w \in L$:
  - Have the enumerator for L print something. If it's equal to w then accept
  - Have the enumerator for $\bar{L}$ print something. If that's equal to w then reject.
  - Repeat.

Some Non-Recognizable Languages

- $COHALT = \{< M, w > | M \text{ does not halt on } w\}$
- $ALL_{TM} = \{< M > | L(M) = \Sigma^*\}$
  - Reduce $COHALT$ to ALL_{TM}

Some Non-Recognizable Languages

- $COHALT = \{< M, w > | M \text{ does not halt on } w\}$
- $ALL_{TM} = \{< M > | L(M) = \Sigma^*\}$
  - Reduce $COHALT$ to ALL_{TM}

Reduce $COHALT$ to ALL_{TM}

- Use a recognizer for ALL_{TM} to recognize $COHALT$
- Given a $COHALT$ instance (i.e. M, w), build a new machine M' such that $L(M') = \Sigma^*$ if and only if $M(w)$ runs forever

Psuedocode for M'

```
public static boolean mPrime(string x){
        count = 0;
        while(M(w) hasn't halted){
                if(count > |x|){
                        return true;
                }
                run M(w) for 1 step;
                count++;
        }
        return false;
}
```

If $M(w)$ halts, there is a longest string I can accept

The only way to accept all strings is for $M(w)$ to run forever

Reduction

- Given an instance M, w of $COHALT$
- Use M, w to build M'
- As the recognizer for ALL_{TM} if $L(M') = \Sigma^*$
- Its answer tells us if $M(w)$ runs forever