Máquinas de Turing III
(*Turing Machines*)

Alan Reyes-Figueroa

Teoría de la Computación **(Aula 25) 13.noviembre.2023**

Restricciones

Extensiones

Modelos de programación

Resumen

- A primera vista, las máquinas de Turing no parecen muy poderosas.
  - ¿Realmente pueden hacer todo lo que un computador puede hacer?
- Discutimos algunas modificaciones para convencernos del potencial de las MT y que pueden simular a cualquier computador.

Resumen

- Necesitamos estudiar las restricciones en el modelo básico de una máquina de Turing (e.g., cintas infinitas sólo en una dirección).
- Asumiendo una forma restricta hace más simple el hablar de simular máquinas de Turing.
  - Esto es esencial para exhibir un lenguaje que no sea recursivamente enumerable.

Resumen

- También necesitamos estudiar generalizaciones del modelo básico de máquina de Turing.
- Necesitamos argumentar que no existe un modelo más poderoso en el caso de “computar” cosas.
  - Ejemplo: Una máquina de Turing no-determinista con 50 cintas 6-dimensionales no es mejor que el modelo básico.

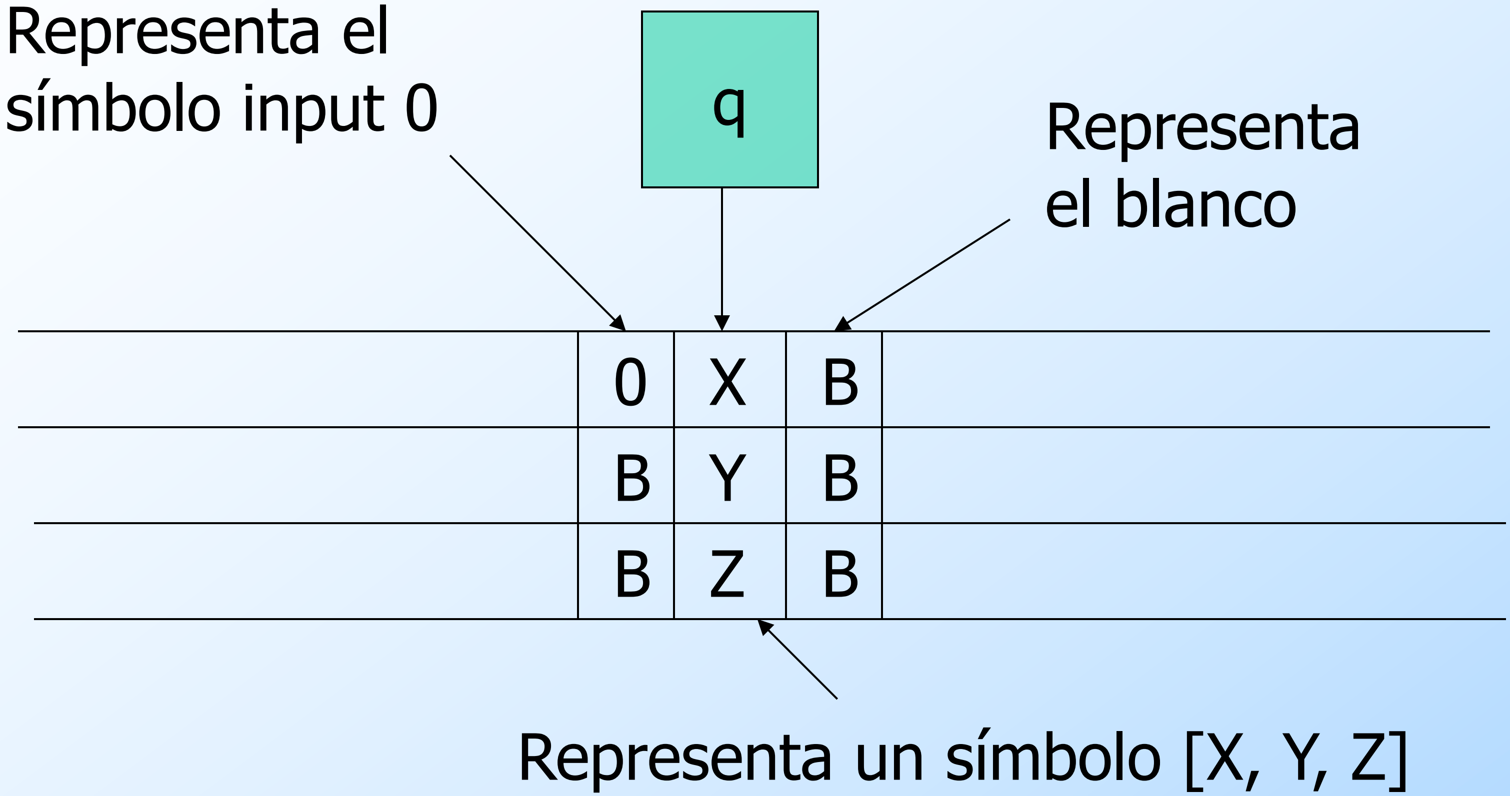
Modelo: Tracks múltiples

- Pensar en símbolos de cinta como vectores de k componentes.

$$X = (X_1, X_2, X_3, \ldots, X_k)$$

- Cada componente X_i se elige dentro de un alfabeto Σ_i.
- Esto hace que el modelo tenga aparentemente k pistas (o *tracks*).
- Hacemos los símbolos input blancos en todas, excepto en una de las pistas.

Tracks múltiples

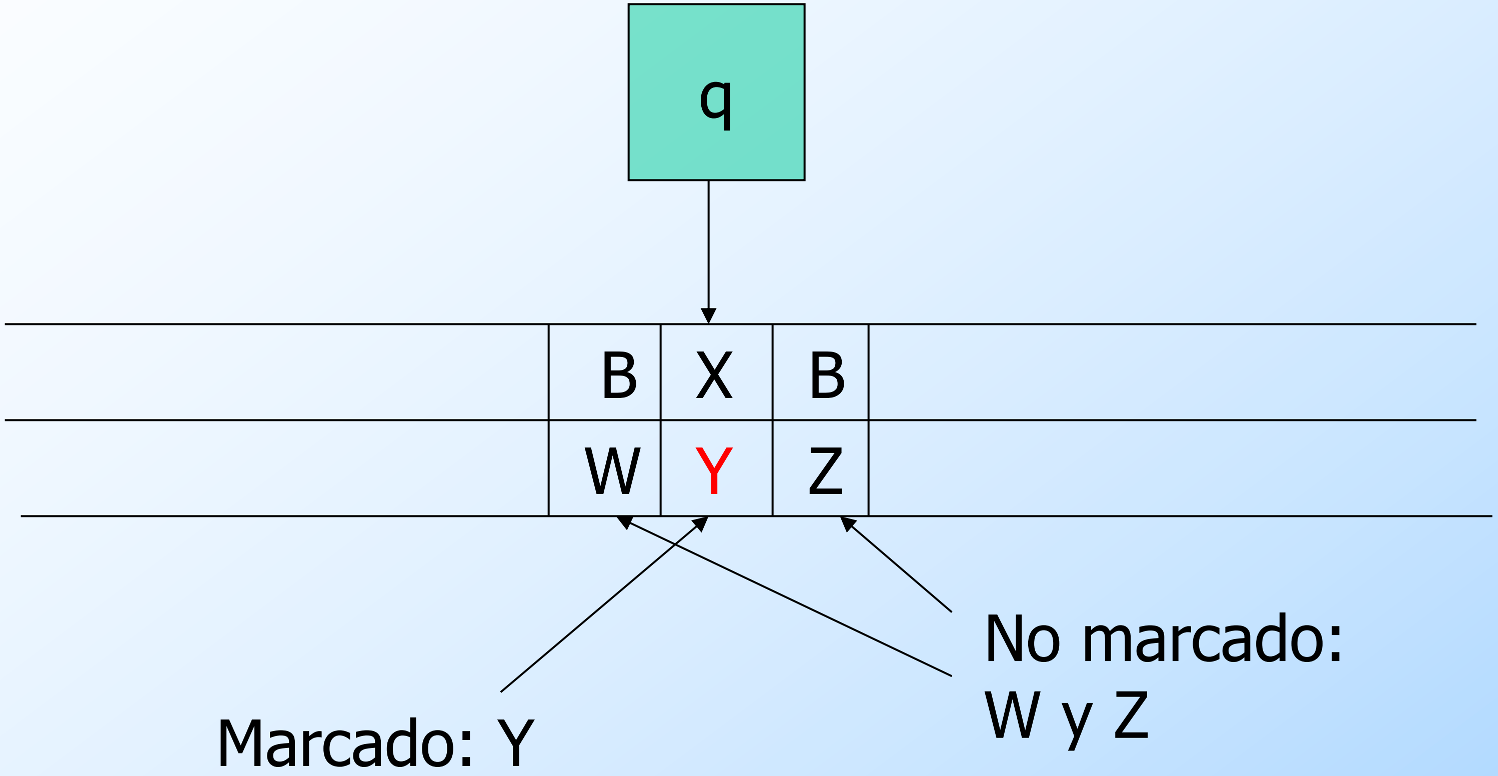


$$\delta: Q \times (\Gamma_1 \times .. \times \Gamma_k) \rightarrow Q \times (\Gamma_1 \times .. \times \Gamma_k) \times D$$

Truco: Marcaje

- Un uso común para las cintas o tracks adicionales es el de *marcar* ciertas posiciones.
- Casi todos las celdas tienen una B (blank) en esta pista, pero varios tienen símbolos especiales (marcas) que permiten a la máquina encontrar lugares particulares en la cinta.

Marcaje



Truco: Almacenamiento en caché

- En inglés (*caching in the state*)
- El estado puede también ser un vector.
- La primera componente es el *estado control* (control state).
- Las otras componentes guardan información de un alfabeto finito.

Ejemplo: usando los trucos

- Construimos una máquina de Turing que copia su entrada input **w** de manera infinita.
- Estados control:
  - q: Marca la posición actual y recuerda los símbolos input vistos.
  - p: Corre a la derecha, recordando el símbolo y buscando una B. Escribe este símbolo.
  - r: Corre a la izquierda, buscando la marca.

Ejemplo

- Los estados tienen la forma [x, Y], donde
  - x es q, p, ó r
  - Y es 0, 1, ó B.
  - Sólo p usa los símbolos 0 y 1.
- Los símbolos de cinta son de la forma [U, V], donde
  - U es X (la "marca") ó B.
  - V es 0, 1 (los símbolos input) ó B.
  - [B, B] es el blanco de la máquina de Turing; [B, 0] y [B, 1] son los inputs.

Función de Transición

- Convención: *a* y *b* cada uno representa "ó 0 ó 1." (exclusivo)
- δ([q,B], [B,a]) = ([p,a], [X,a], R).
  - En el estado q, copiamos el símbolo input bajo el lector (i.e., *a*) en el estado.
  - Marcamos la posición de lectura.
  - Vamos al estado p, y nos movemos a la derecha.

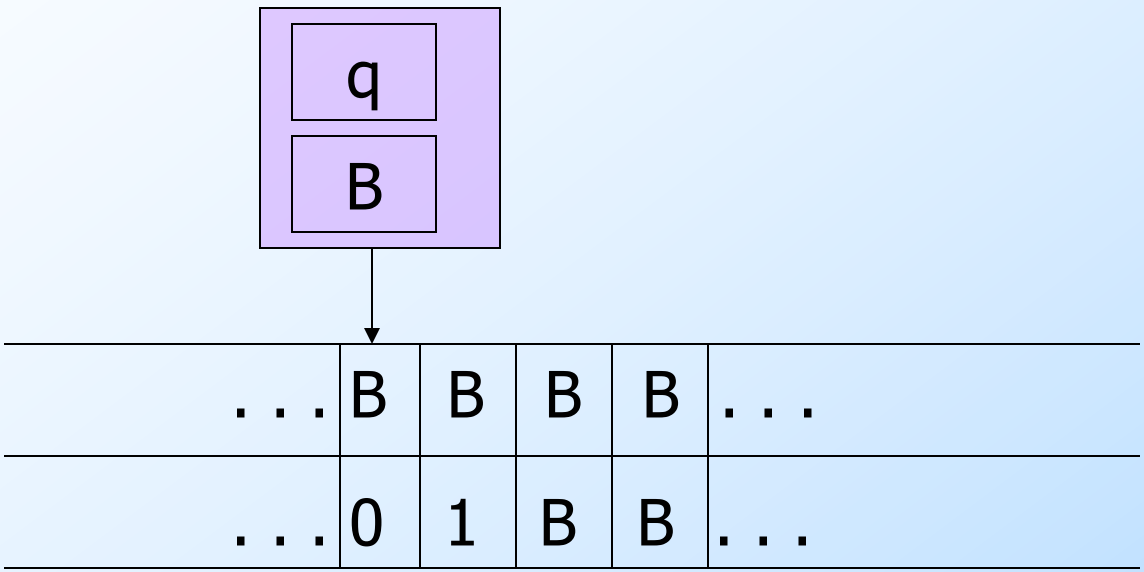
Función de Transición

- $\delta([p,a], [B,b]) = ([p,a], [B,b], R)$.
  - En el estado p, hacia la derecha, buscamos un blanco (no sólo B en la pista de marcas).
- $\delta([p,a], [B,B]) = ([r,B], [B,a], L)$.
  - Cuando encontramos la B, la reemplazamos por el símbolo (*a*) guardado en la “cache”.
  - Vamos al estado r y nos movemos a la izq.

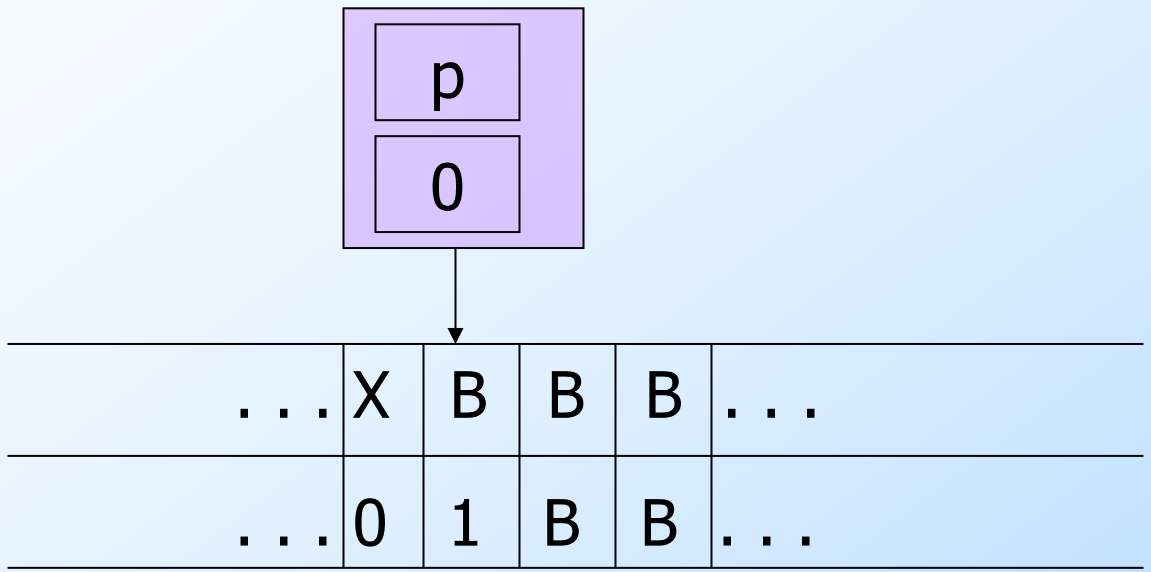
Función de Transición

- $\delta([r,B], [B,a]) = ([r,B], [B,a], L)$.
  - En el estado r, nos movemos a la izquierda, buscando la marca.
- $\delta([r,B], [X,a]) = ([q,B], [B,a], R)$.
  - Cuando hallamos la marca, vamos al estado q y nos movemos a la derecha.
  - Removemos la marca de donde estaba.
  - q pondrá una nueva marca, y el ciclo se repite.

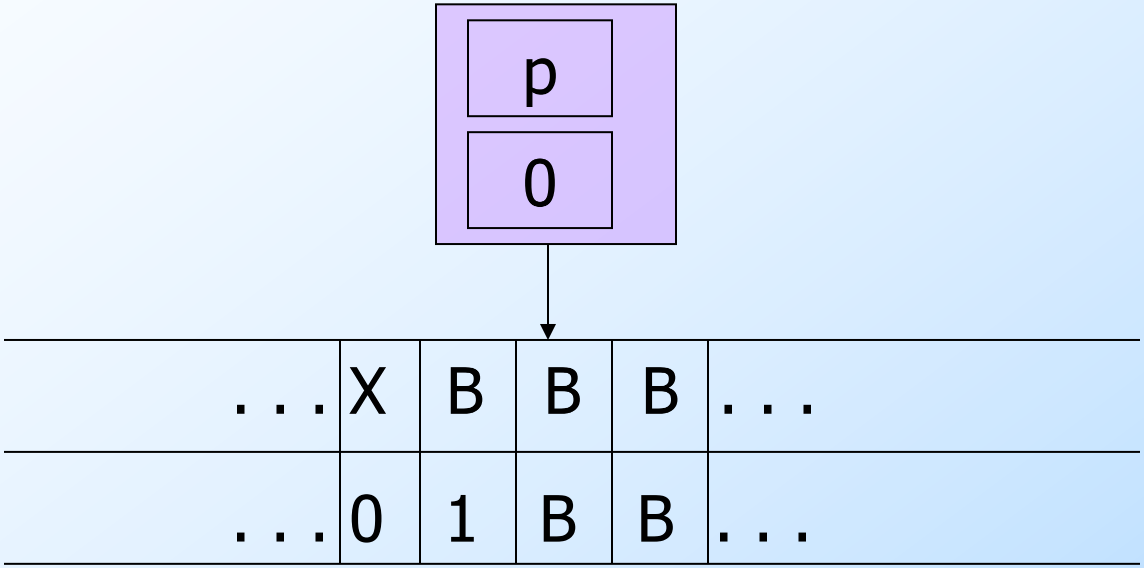
Simulación



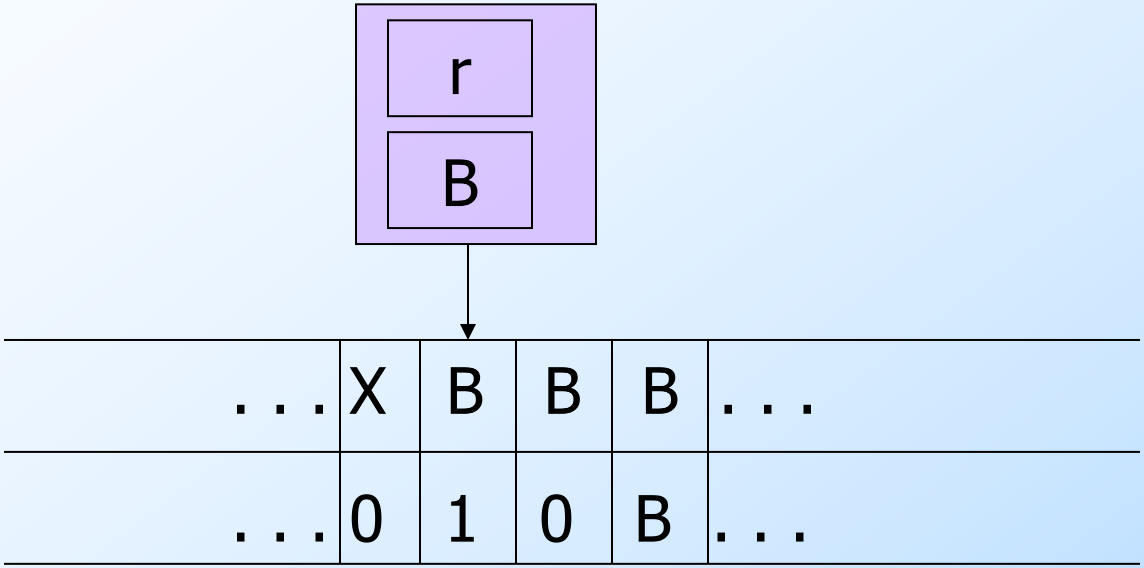
Simulación



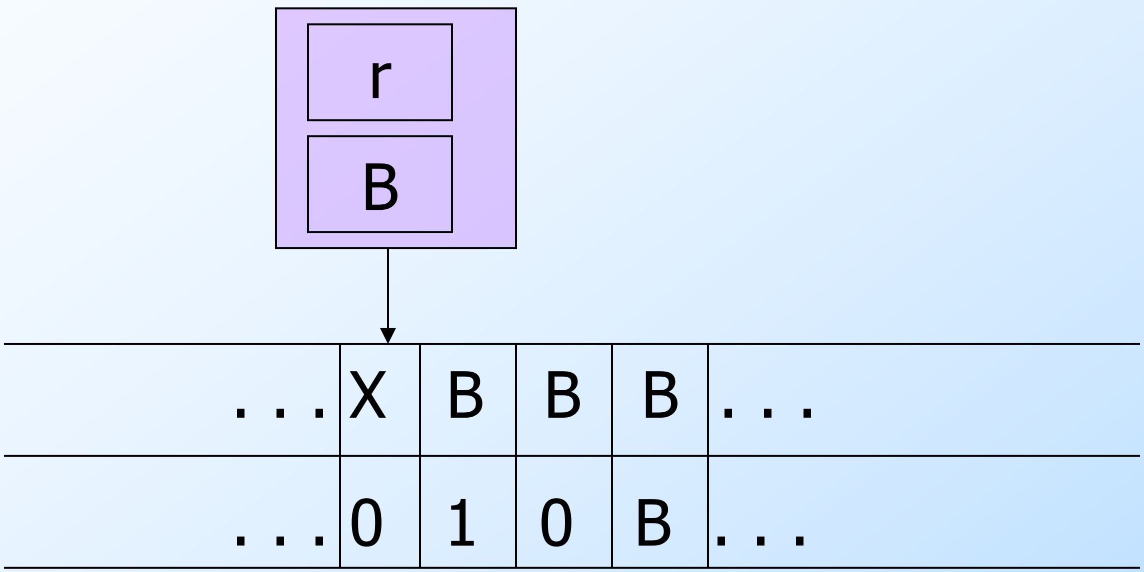
Simulación



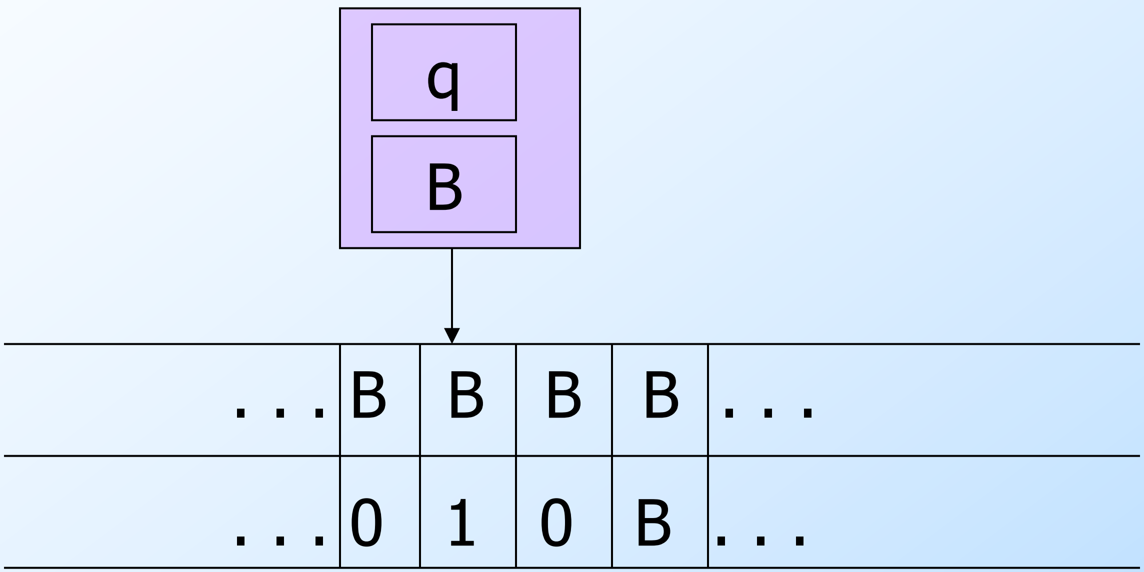
Simulación



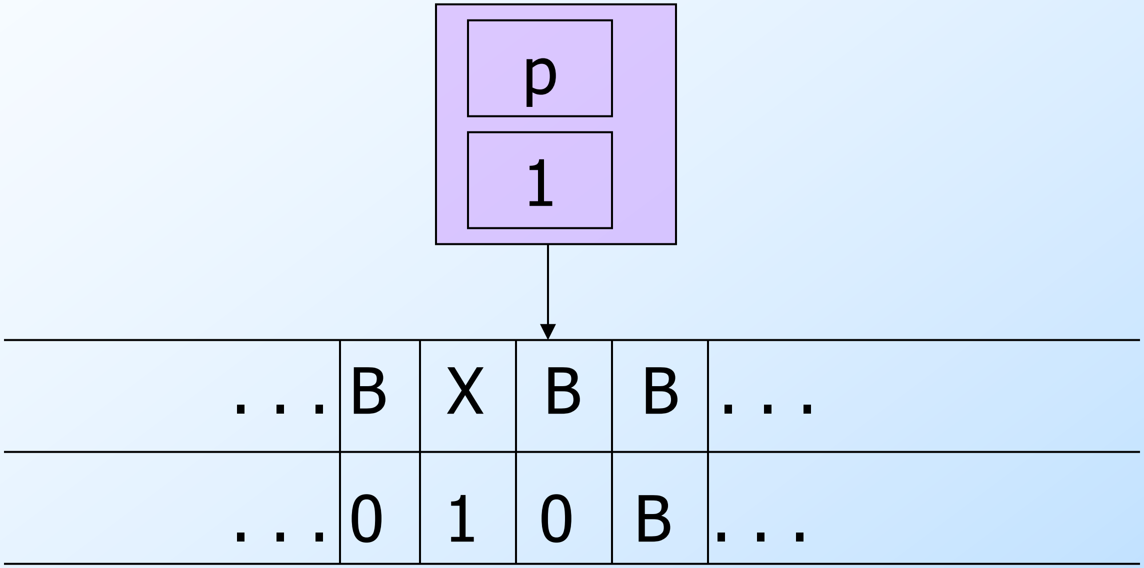
Simulación



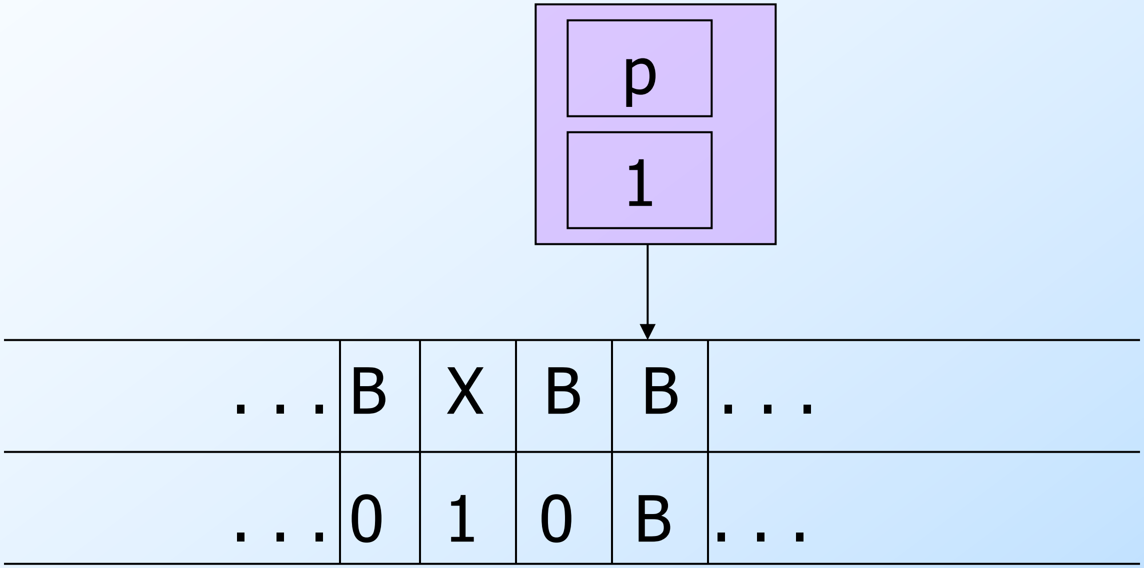
Simulación



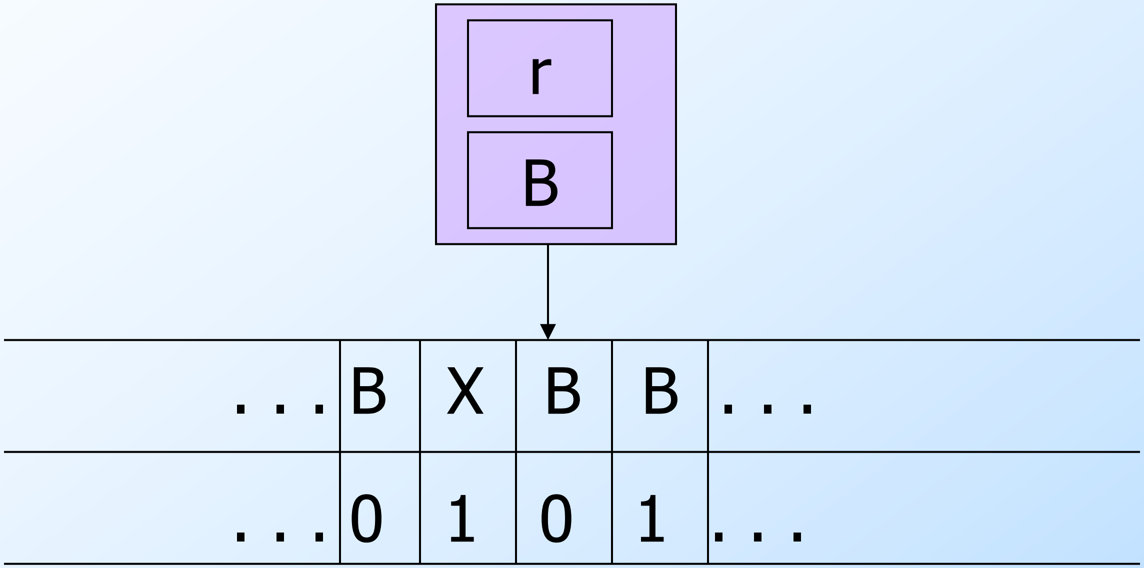
Simulación



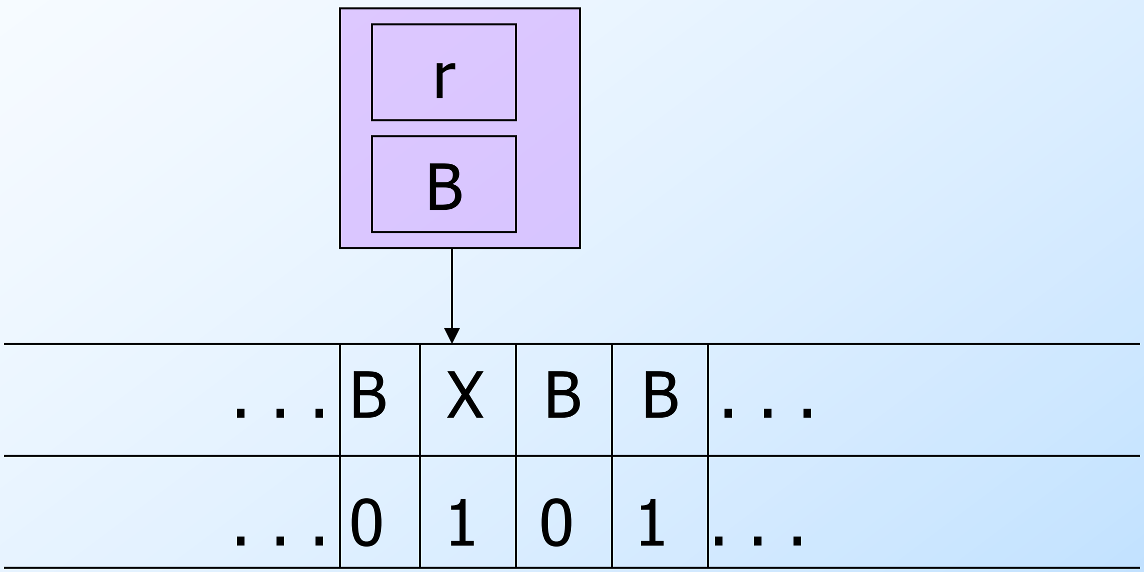
Simulación



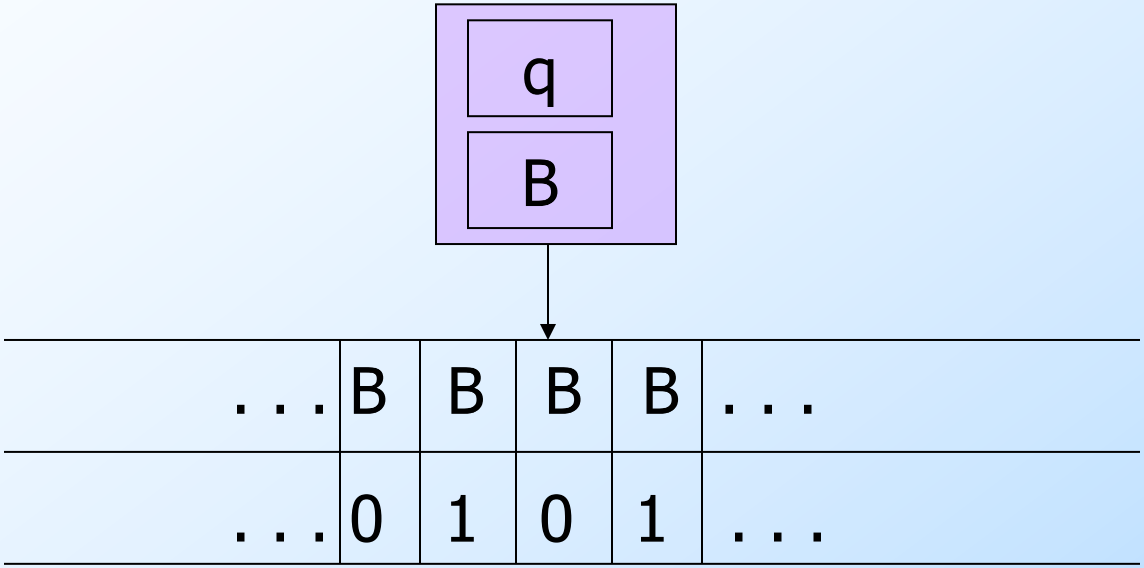
Simulación



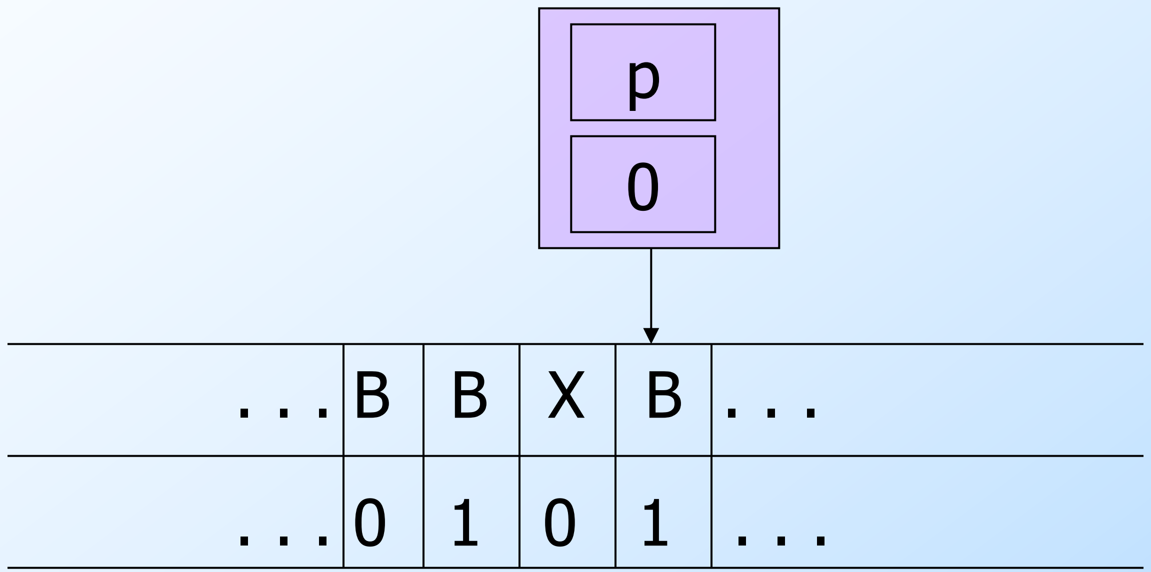
Simulación



Simulación



Simulación

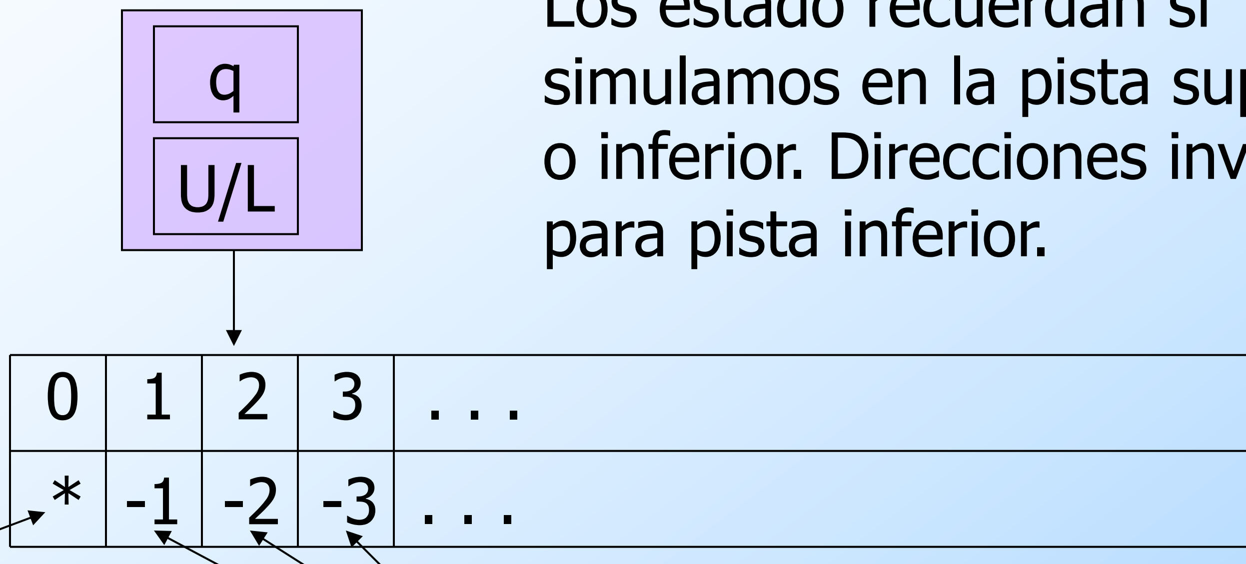


Cintas Semi-infinitas

- Podemos asumir que la máquina de Turing nunca se mueve a la izquierda de la posición inicial del lector.
- Denotamos esta posición por 0; las celdas a la derecha son las 1, 2, … y las celdas a la izquierda son las -1, -2, …
- La nueva máquina de Turing tiene dos pistas:
  - La 1ra guarda las celdas 0, 1, 2, …
  - La 2da guarda el marcador, celdas -1, -2, …

Simulando Cinta Infinita con una Cinta Semi-infinita

Los estado recuerdan si simulamos en la pista superior o inferior. Direcciones inversas para pista inferior.



Escribe * Aquí en la 1a movida.

No se necesita hacer nada, ya que éstos son B al inicio.

Más Restricciones – Read in Text

- Dos stacks pueden simular una cinta.
  - Uno guarda las posiciones a la izquierda del lector; el otro guarda las posiciones a la derecha.
- De hecho, definimos ambos stacks como *counters* = sólo dos símbolos de stack, uno de los cuales sólo aparece en el segundo.

Extensiones

- La idea es construir modelos de máquinas de Turing más generales (o con más capacidad) que la máquina de Turing estándar.

1. Máquinas multi-cinta.
2. Máquinas de Turing no-deterministas.
3. Guardar pares de key-values.

Máquinas multi-cinta

- Permitimos a una máquina de Turing poseer k cintas, ($k \geq 1$).
- Los movimientos de la máquina dependen del estado, y de todos los símbolos bajo los k lectores de cinta.

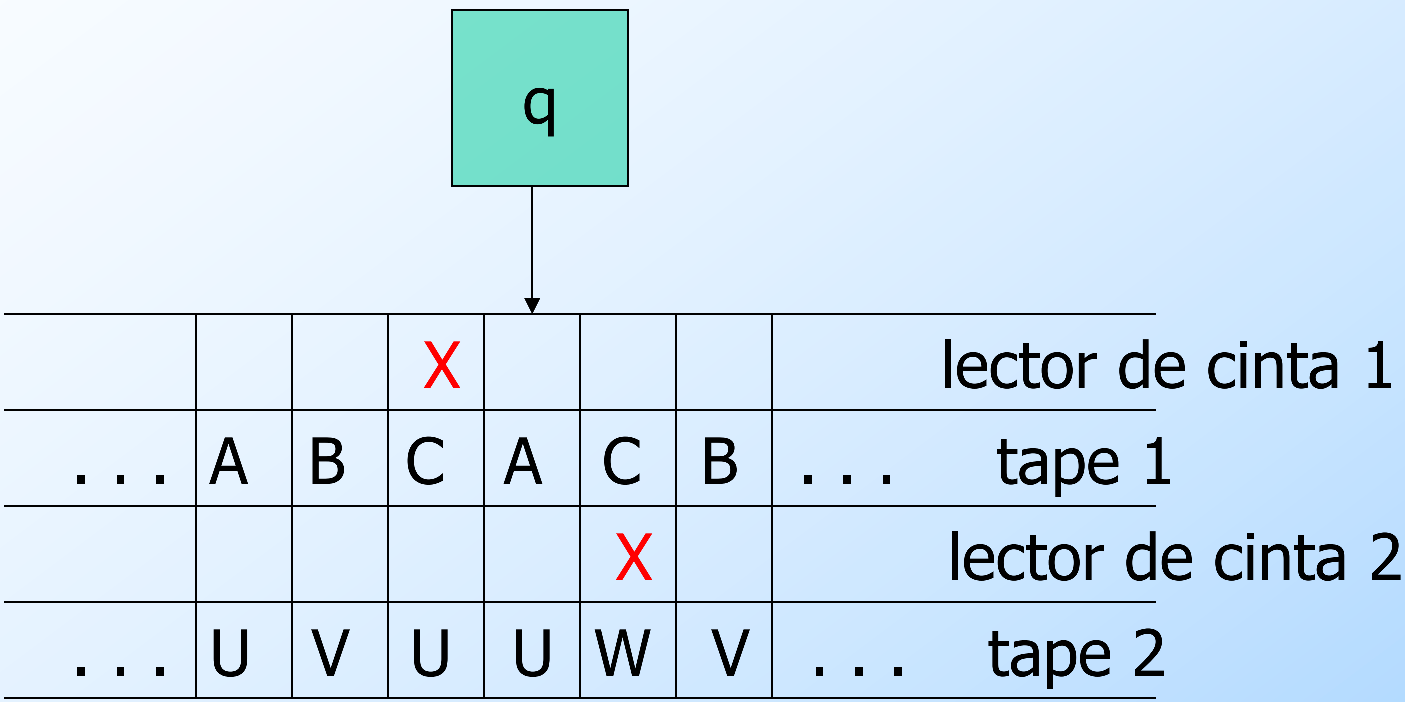
$$\delta: Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times D^k$$

- En una movida, la máquina puede cambiar de estado, escribir en cada una de las cintas, y mover cada lector de manera independiente.

Simulando k cintas con una

- Usamos 2k pistas.
- Cada cinta de la máquina multi-cinta se representa por un track diferente.
- La posición del lector en cada cinta se representa por una marca en un track adicional.

Simulación multi-cinta



MT no-deterministas

- Permiten a la máquina de Turing la opción de moverse o no moverse, en cada paso.
  - Cada elección es una tripla (q,Y,d) (estado, símbolo, dirección), al igual que en el caso determinístico.
- La máquina acepta su input w si alguna secuencia de movimientos conduce a un estado de aceptación.

Simular una MTN por una MTD

- La máquina determinista MTD mantiene en su cinta una cola con los ID's de la MTN.
- Una segunda pista se utiliza para marcar ciertas posiciones:
  1. Una marca para el ID en el inicio de la cola.
  2. Una marca para copiar el ID del inicio y hacer un movimiento.

Cinta de la MTD

Lugar donde se copia el ID_k con un movimiento

Frente de la cola

X Y

ID_0 # ID_1 # … # ID_k # ID_{k+1} … # ID_n # New ID

Final de la cola

Simular la MTD

- La máquina determinista MTD encuentra el ID en en inicio de la cola.
- Busca el estado en ese ID para poder determinar los movimientos permitidos desde ese ID.
- Si hay m movimientos posibles, crea m nuevos ID, uno para cada movimiento, al final de la cola.

Operación de la MTD

- Los m nuevos ID se crean uno a la vez.
- Una vez creados, el marcador de la parte delantera de la cola se mueve un ID hacia la parte trasera de la cola.
- Sin embargo, si una ID creada tiene un estado de aceptación, l MTD acepta y se detiene.

¿Por qué la construcción MTN --> MTD funciona?

- Hay un límite superior, digamos k, en el número de movimientos posibles de la MTN para cualquier combinación de estado/símbolo.
- Por lo tanto, cualquier ID accesible desde el ID inicial mediante n movimientos de la MTN será construida por la MTD después de construir como máximo $(k^{n+1}-k)/(k-1)$ ID's.

Sum of $k+k^2+\ldots+k^n$

¿Por qué?

- Si la MTN acepta, lo hace en alguna secuencia de n opciones de movimiento.
- Por lo tanto, el ID con un estado de aceptación será construido por la MTD en un gran número de sus propios movimientos.
- Si la MTN no acepta, no hay forma de que la MTD acepte.

Ventaja de las Extensiones

- Ahora tenemos una muy buena situación:
- Cuando discutimos la construcción de las MT particulares que toman otras MT como entrada, podemos asumir que la MT de entrada es lo más simple posible.
- Por ejemplo, uno, cinta semi-infinita, determinista.
- Pero la MT simuladora puede tener muchas cintas, ser no determinista, etc.

Computadores

- Recordemos que, dado que una computadora real tiene una memoria finita, en cierto sentido es más "débil" que una Máquina de Turing.
- Imaginemos una computadora que almacena (infinitos) pares del tipo (nombre, valor) = (key, value).
  - Generaliza un espacio de direcciones.

Simulando un Key-Value Store mediante una MT

- La MT utiliza una de varias cintas para contener una secuencia arbitrariamente grande de pares de (nombre, valor) en el formato #nombre*valor #...
- Marcamos, usando una segunda pista, el extremo izquierdo de la secuencia.
- Una segunda cinta puede contener un nombre cuyo valor queremos buscar.

Búsqueda

- Comenzando en el extremo izquierdo del *store*, comparamos el nombre de búsqueda (key) con cada nombre en el store.
- Cuando encontramos una coincidencia, tomamos lo que sigue entre el * y el siguiente # como valor.

Inserción

- Supongamos que queremos insertar un par key-value (k,v) o reemplazar el valor actual asociado con la etiqueta k por v.
- Realizamos una búsqueda del nombre k. Si no lo encuentra, agregamos k*v# al final del store.

Inserción

- Si encontramos #n*v'#, necesitamos reemplazar v' por v.
- Si v es más corto que v’, podemos dejar espacios en blanco para completar el reemplazo.
- Pero si v es más largo que v', necesitaremos hacer espacio.

Inserción

- Usamos una tercera cinta para copiar todo desde la primera cinta a la derecha de v’.
- Marcamos la posición del * a la izquierda de v' antes de hacer esta copia.
- Copiamos de la tercera cinta a la primera, dejando suficiente espacio para v.
- Finalmente, escribimos v donde estaba v’.