3 Formalizaciones - Un Panorama I

José de Jesús Lavalle Martínez

Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación
Computabilidad CCOS 257

Contenido

Motivación

- En la sección anterior, el concepto de calculabilidad efectiva se describió muy informalmente.

Motivación

- En la sección anterior, el concepto de calculabilidad efectiva se describió muy informalmente.
- Ahora queremos hacer aquellas ideas precisas (es decir, hacerlas parte de las matemáticas).

Motivación

- En la sección anterior, el concepto de calculabilidad efectiva se describió muy informalmente.
- Ahora queremos hacer aquellas ideas precisas (es decir, hacerlas parte de las matemáticas).
- En efecto, varios enfoques para hacerlo serán descritos: dispositivos de cálculo idealizados, definiciones generativas (es decir, la clase más pequeña que contiene ciertas funciones iniciales y que es cerrada bajo ciertas construcciones), lenguajes de programación y definibilidad en lenguajes formales.

Motivación

- En la sección anterior, el concepto de calculabilidad efectiva se describió muy informalmente.
- Ahora queremos hacer aquellas ideas precisas (es decir, hacerlas parte de las matemáticas).
- En efecto, varios enfoques para hacerlo serán descritos: dispositivos de cálculo idealizados, definiciones generativas (es decir, la clase más pequeña que contiene ciertas funciones iniciales y que es cerrada bajo ciertas construcciones), lenguajes de programación y definibilidad en lenguajes formales.
- Es un hecho significativo que estos enfoques, a pesar de ser muy distintos, producen conceptos exactamente equivalentes.

Motivación

- En la sección anterior, el concepto de calculabilidad efectiva se describió muy informalmente.
- Ahora queremos hacer aquellas ideas precisas (es decir, hacerlas parte de las matemáticas).
- En efecto, varios enfoques para hacerlo serán descritos: dispositivos de cálculo idealizados, definiciones generativas (es decir, la clase más pequeña que contiene ciertas funciones iniciales y que es cerrada bajo ciertas construcciones), lenguajes de programación y definibilidad en lenguajes formales.
- Es un hecho significativo que estos enfoques, a pesar de ser muy distintos, producen conceptos exactamente equivalentes.
- Esta sección da un panorama general de un número de maneras diferentes (pero equivalentes) de formalizar los conceptos de calculabilidad efectiva.

Motivación

- En la sección anterior, el concepto de calculabilidad efectiva se describió muy informalmente.
- Ahora queremos hacer aquellas ideas precisas (es decir, hacerlas parte de las matemáticas).
- En efecto, varios enfoques para hacerlo serán descritos: dispositivos de cálculo idealizados, definiciones generativas (es decir, la clase más pequeña que contiene ciertas funciones iniciales y que es cerrada bajo ciertas construcciones), lenguajes de programación y definibilidad en lenguajes formales.
- Es un hecho significativo que estos enfoques, a pesar de ser muy distintos, producen conceptos exactamente equivalentes.
- Esta sección da un panorama general de un número de maneras diferentes (pero equivalentes) de formalizar los conceptos de calculabilidad efectiva.
- En capítulos posteriores se desarrollarán con todo detalle algunas de estas maneras.

Máquinas de Turing I

- A principios de 1935, Alan Turing era un estudiante de 22 años de edad graduado en el King's College en Cambridge.

Máquinas de Turing I

- A principios de 1935, Alan Turing era un estudiante de 22 años de edad graduado en el King's College en Cambridge.
- Bajo la asesoría de Max Newman, estuvo trabajando en el problema de formalizar el concepto de calculabilidad efectiva.

Máquinas de Turing I

- A principios de 1935, Alan Turing era un estudiante de 22 años de edad graduado en el King's College en Cambridge.
- Bajo la asesoría de Max Newman, estuvo trabajando en el problema de formalizar el concepto de calculabilidad efectiva.
- En 1936, se enteró del trabajo de Alonzo Church, en Princeton.

Máquinas de Turing I

- A principios de 1935, Alan Turing era un estudiante de 22 años de edad graduado en el King's College en Cambridge.
- Bajo la asesoría de Max Newman, estuvo trabajando en el problema de formalizar el concepto de calculabilidad efectiva.
- En 1936, se enteró del trabajo de Alonzo Church, en Princeton.
- Church también había estado trabajando sobre el mismo problema, y en su artículo de 1936, "Un problema insoluble de teoría elemental de números", presentó la afirmación categórica de que la clase de funciones calculables efectivamente debe ser idéntica a la clase de funciones definibles en el *cálculo lambda*, un lenguaje formal para especificar la construcción de funciones.

Máquinas de Turing I

- A principios de 1935, Alan Turing era un estudiante de 22 años de edad graduado en el King's College en Cambridge.
- Bajo la asesoría de Max Newman, estuvo trabajando en el problema de formalizar el concepto de calculabilidad efectiva.
- En 1936, se enteró del trabajo de Alonzo Church, en Princeton.
- Church también había estado trabajando sobre el mismo problema, y en su artículo de 1936, "Un problema insoluble de teoría elemental de números", presentó la afirmación categórica de que la clase de funciones calculables efectivamente debe ser idéntica a la clase de funciones definibles en el *cálculo lambda*, un lenguaje formal para especificar la construcción de funciones.
- Aún más, Church mostró que exactamente la misma clase de funciones puede ser caracterizada en términos de derivabilidad formal de ecuaciones.

Máquinas de Turing II

- Luego Turing terminó de escribir rápidamente su artículo, en el cual presentó un enfoque muy diferente para caracterizar las funciones calculables efectivamente, pero uno que -como demostró- una vez más producía la misma clase de funciones que Church había propuesto.

Máquinas de Turing II

- Luego Turing terminó de escribir rápidamente su artículo, en el cual presentó un enfoque muy diferente para caracterizar las funciones calculables efectivamente, pero uno que -como demostró- una vez más producía la misma clase de funciones que Church había propuesto.
- Con el estímulo de Newman, Turing fue a Princeton por dos años, donde escribió su tesis doctoral bajo la dirección de Alonzo Church.

Máquinas de Turing II

- Luego Turing terminó de escribir rápidamente su artículo, en el cual presentó un enfoque muy diferente para caracterizar las funciones calculables efectivamente, pero uno que -como demostró- una vez más producía la misma clase de funciones que Church había propuesto.
- Con el estímulo de Newman, Turing fue a Princeton por dos años, donde escribió su tesis doctoral bajo la dirección de Alonzo Church.
- El artículo de Turing sigue siendo una introducción a sus ideas muy legible. ¿Cómo podría un empleado diligente realizar un cálculo siguiendo instrucciones?

Máquinas de Turing II

- Luego Turing terminó de escribir rápidamente su artículo, en el cual presentó un enfoque muy diferente para caracterizar las funciones calculables efectivamente, pero uno que -como demostró- una vez más producía la misma clase de funciones que Church había propuesto.
- Con el estímulo de Newman, Turing fue a Princeton por dos años, donde escribió su tesis doctoral bajo la dirección de Alonzo Church.
- El artículo de Turing sigue siendo una introducción a sus ideas muy legible. ¿Cómo podría un empleado diligente realizar un cálculo siguiendo instrucciones?
- Él o ella podría organizar el trabajo en una libreta. En cualquier momento dado, su atención se centra en una página en particular.

Máquinas de Turing II

- Luego Turing terminó de escribir rápidamente su artículo, en el cual presentó un enfoque muy diferente para caracterizar las funciones calculables efectivamente, pero uno que -como demostró- una vez más producía la misma clase de funciones que Church había propuesto.
- Con el estímulo de Newman, Turing fue a Princeton por dos años, donde escribió su tesis doctoral bajo la dirección de Alonzo Church.
- El artículo de Turing sigue siendo una introducción a sus ideas muy legible. ¿Cómo podría un empleado diligente realizar un cálculo siguiendo instrucciones?
- Él o ella podría organizar el trabajo en una libreta. En cualquier momento dado, su atención se centra en una página en particular.
- Siguiendo sus instrucciones, podría alterar la página, y entonces podría cambiar a otra página.

Máquinas de Turing II

- Luego Turing terminó de escribir rápidamente su artículo, en el cual presentó un enfoque muy diferente para caracterizar las funciones calculables efectivamente, pero uno que -como demostró- una vez más producía la misma clase de funciones que Church había propuesto.
- Con el estímulo de Newman, Turing fue a Princeton por dos años, donde escribió su tesis doctoral bajo la dirección de Alonzo Church.
- El artículo de Turing sigue siendo una introducción a sus ideas muy legible. ¿Cómo podría un empleado diligente realizar un cálculo siguiendo instrucciones?
- Él o ella podría organizar el trabajo en una libreta. En cualquier momento dado, su atención se centra en una página en particular.
- Siguiendo sus instrucciones, podría alterar la página, y entonces podría cambiar a otra página.
- Y como la libreta es suficientemente grande (o el suministro de papel nuevo es suficientemente basto) nunca llega a la última página.

Máquinas de Turing III

- El alfabeto de símbolos disponibles para el empleado debe ser finito; si hubieran infinitos símbolos, entonces habrían dos que serían arbitrariamente similares y se podría confundir.

Máquinas de Turing III

- El alfabeto de símbolos disponibles para el empleado debe ser finito; si hubieran infinitos símbolos, entonces habrían dos que serían arbitrariamente similares y se podría confundir.
- Luego podemos sin perdida de generalidad pensar que podemos escribir sobre una página de la libreta un solo símbolo.

Máquinas de Turing III

- El alfabeto de símbolos disponibles para el empleado debe ser finito; si hubieran infinitos símbolos, entonces habrían dos que serían arbitrariamente similares y se podría confundir.
- Luego podemos sin perdida de generalidad pensar que podemos escribir sobre una página de la libreta un solo símbolo.
- Y podemos imaginar que las páginas de la libreta están puestas una al lado de la otra, formando una cinta de papel, que consiste de cuadrados, cada cuadrado está en blanco o tiene impreso un símbolo.

Máquinas de Turing III

- El alfabeto de símbolos disponibles para el empleado debe ser finito; si hubieran infinitos símbolos, entonces habrían dos que serían arbitrariamente similares y se podría confundir.
- Luego podemos sin perdida de generalidad pensar que podemos escribir sobre una página de la libreta un solo símbolo.
- Y podemos imaginar que las páginas de la libreta están puestas una al lado de la otra, formando una cinta de papel, que consiste de cuadrados, cada cuadrado está en blanco o tiene impreso un símbolo.
- Por uniformidad, podemos pensar que un cuadrado blanco contiene el símbolo "blanco" B.

Máquinas de Turing III

- El alfabeto de símbolos disponibles para el empleado debe ser finito; si hubieran infinitos símbolos, entonces habrían dos que serían arbitrariamente similares y se podría confundir.
- Luego podemos sin perdida de generalidad pensar que podemos escribir sobre una página de la libreta un solo símbolo.
- Y podemos imaginar que las páginas de la libreta están puestas una al lado de la otra, formando una cinta de papel, que consiste de cuadrados, cada cuadrado está en blanco o tiene impreso un símbolo.
- Por uniformidad, podemos pensar que un cuadrado blanco contiene el símbolo "blanco" B.
- En cada etapa de su trabajo, el empleado -o la máquina mecánica- puede alterar el cuadrado que está examinando, puede poner atención al siguiente cuadrado o al previo, y puede ver las instrucciones para saber cual de ellas debe seguir a continuación.

Máquinas de Turing III

- El alfabeto de símbolos disponibles para el empleado debe ser finito; si hubieran infinitos símbolos, entonces habrían dos que serían arbitrariamente similares y se podría confundir.
- Luego podemos sin perdida de generalidad pensar que podemos escribir sobre una página de la libreta un solo símbolo.
- Y podemos imaginar que las páginas de la libreta están puestas una al lado de la otra, formando una cinta de papel, que consiste de cuadrados, cada cuadrado está en blanco o tiene impreso un símbolo.
- Por uniformidad, podemos pensar que un cuadrado blanco contiene el símbolo "blanco" B.
- En cada etapa de su trabajo, el empleado -o la máquina mecánica- puede alterar el cuadrado que está examinando, puede poner atención al siguiente cuadrado o al previo, y puede ver las instrucciones para saber cual de ellas debe seguir a continuación.
- Turing describió la última parte como un "cambio de estado mental".

Máquinas de Turing IV

- Turing escribió, “Ahora podemos construir una máquina para hacer el trabajo”.

Máquinas de Turing IV

- Turing escribió, “Ahora podemos construir una máquina para hacer el trabajo”.
- Tal máquina es, por supuesto, ahora llamada *máquina de Turing*, una frase que usó primero Church en su revisión al artículo de Turing que apareció en *The Journal of Simbolic Logic*.

Máquinas de Turing IV

- Turing escribió, "Ahora podemos construir una máquina para hacer el trabajo".
- Tal máquina es, por supuesto, ahora llamada *máquina de Turing*, una frase que usó primero Church en su revisión al artículo de Turing que apareció en *The Journal of Simbolic Logic*.
- La máquina tiene una cinta potencialmente infinita. Inicialmente el numeral o palabra dada de entrada se escribe en la cinta, la cual es blanca en los restantes cuadrados.

Máquinas de Turing IV

- Turing escribió, "Ahora podemos construir una máquina para hacer el trabajo".
- Tal máquina es, por supuesto, ahora llamada *máquina de Turing*, una frase que usó primero Church en su revisión al artículo de Turing que apareció en *The Journal of Simbolic Logic*.
- La máquina tiene una cinta potencialmente infinita. Inicialmente el numeral o palabra dada de entrada se escribe en la cinta, la cual es blanca en los restantes cuadrados.
- La máquina es capaz de estar en cualquiera de los "estados" finitos (la palabra "mental" es inapropiada para un máquina).

Máquinas de Turing IV

- Turing escribió, "Ahora podemos construir una máquina para hacer el trabajo".
- Tal máquina es, por supuesto, ahora llamada *máquina de Turing*, una frase que usó primero Church en su revisión al artículo de Turing que apareció en *The Journal of Simbolic Logic*.
- La máquina tiene una cinta potencialmente infinita. Inicialmente el numeral o palabra dada de entrada se escribe en la cinta, la cual es blanca en los restantes cuadrados.
- La máquina es capaz de estar en cualquiera de los "estados" finitos (la palabra "mental" es inapropiada para un máquina).
- En cada paso del cálculo, dependiendo del estado en que se encuentre en el momento, la máquina puede cambiar el símbolo en el cuadrado que está examinando en ese momento, y puede dirigir su atención al cuadrado a la izquierda o la derecha, y puede entonces cambiar su estado a otro estado.

Máquinas de Turing IV

- Turing escribió, "Ahora podemos construir una máquina para hacer el trabajo".
- Tal máquina es, por supuesto, ahora llamada *máquina de Turing*, una frase que usó primero Church en su revisión al artículo de Turing que apareció en *The Journal of Simbolic Logic*.
- La máquina tiene una cinta potencialmente infinita. Inicialmente el numeral o palabra dada de entrada se escribe en la cinta, la cual es blanca en los restantes cuadrados.
- La máquina es capaz de estar en cualquiera de los "estados" finitos (la palabra "mental" es inapropiada para un máquina).
- En cada paso del cálculo, dependiendo del estado en que se encuentre en el momento, la máquina puede cambiar el símbolo en el cuadrado que está examinando en ese momento, y puede dirigir su atención al cuadrado a la izquierda o la derecha, y puede entonces cambiar su estado a otro estado.
- La cinta se extiende sin fin en ambas direcciones.

Máquinas de Turing V

- El programa para esta máquina de Turing se puede dar mediante una tabla.

Máquinas de Turing V

- El programa para esta máquina de Turing se puede dar mediante una tabla.
- Donde los posibles estados de la máquina son $q_1, \ldots, q_r$, cada línea de la tabla es una quíntupla $\langle q_i, S_j, S_k, D, q_m \rangle$ la que será interpretada como que siempre que la máquina esté en el estado q_i y el cuadrado examinado contiene el símbolo S_j, entonces el símbolo debe ser cambiado a S_k y la máquina debe cambiar su atención al cuadrado a la izquierda (si $D = L$) o a la derecha (si $D = R$), y debe cambiar su estado a q_m.

Máquinas de Turing V

- El programa para esta máquina de Turing se puede dar mediante una tabla.
- Donde los posibles estados de la máquina son $q_1, \ldots, q_r$, cada línea de la tabla es una quíntupla $\langle q_i, S_j, S_k, D, q_m \rangle$ la que será interpretada como que siempre que la máquina esté en el estado q_i y el cuadrado examinado contiene el símbolo S_j, entonces el símbolo debe ser cambiado a S_k y la máquina debe cambiar su atención al cuadrado a la izquierda (si $D = L$) o a la derecha (si $D = R$), y debe cambiar su estado a q_m.
- Posiblemente el símbolo S_j es el símbolo "blanco" B, lo que significa que el cuadrado examinado es blanco; posiblemente S_k es B, lo que significa que lo que sea que esté en el cuadrado tiene que ser borrado.

Máquinas de Turing VI

- Para que el programa no sea ambiguo, no deben haber dos quintuplas diferentes con los primeros dos componentes iguales.

Máquinas de Turing VI

- Para que el programa no sea ambiguo, no deben haber dos quintuplas diferentes con los primeros dos componentes iguales.
- Al relajar el requerimiento con respecto a la ausencia de ambigüedad, obtenemos el concepto de máquina de Turing *no determinista*, la que será útil posteriormente, en la discusión de computabilidad factible).

Máquinas de Turing VI

- Para que el programa no sea ambiguo, no deben haber dos quintuplas diferentes con los primeros dos componentes iguales.
- Al relajar el requerimiento con respecto a la ausencia de ambigüedad, obtenemos el concepto de máquina de Turing *no determinista*, la que será útil posteriormente, en la discusión de computabilidad factible).
- Uno de los estados, digamos, q_1, es designado como el estado inicial, el estado en el que la máquina empieza a calcular.

Máquinas de Turing VI

- Para que el programa no sea ambiguo, no deben haber dos quintuplas diferentes con los primeros dos componentes iguales.
- Al relajar el requerimiento con respecto a la ausencia de ambigüedad, obtenemos el concepto de máquina de Turing *no determinista*, la que será útil posteriormente, en la discusión de computabilidad factible).
- Uno de los estados, digamos, q_1, es designado como el estado inicial, el estado en el que la máquina empieza a calcular.
- Si empezamos a ejecutar la máquina en este estado, y examinamos el primer cuadrado de su entrada, podría (o tal vez no), después de algún número de pasos, alcanzar un estado y un símbolo para los cuales su tabla carece de una quintupla que tenga ese estado y ese símbolo como sus primeros dos componentes.

Máquinas de Turing VI

- Para que el programa no sea ambiguo, no deben haber dos quintuplas diferentes con los primeros dos componentes iguales.
- Al relajar el requerimiento con respecto a la ausencia de ambigüedad, obtenemos el concepto de máquina de Turing *no determinista*, la que será útil posteriormente, en la discusión de computabilidad factible).
- Uno de los estados, digamos, q_1, es designado como el estado inicial, el estado en el que la máquina empieza a calcular.
- Si empezamos a ejecutar la máquina en este estado, y examinamos el primer cuadrado de su entrada, podría (o tal vez no), después de algún número de pasos, alcanzar un estado y un símbolo para los cuales su tabla carece de una quintupla que tenga ese estado y ese símbolo como sus primeros dos componentes.
- En ese punto, la máquina *para*, y podemos mirar en la cinta (iniciando con el cuadrado que estaba bajo consideración) para ver que numeral o palabra de salida tiene.

Máquinas de Turing VII

- Ahora suponga que Σ es un alfabeto finito (el símbolo B no cuenta como un miembro de Σ).

Máquinas de Turing VII

- Ahora suponga que Σ es un alfabeto finito (el símbolo B no cuenta como un miembro de Σ).
- Sea Σ^* el conjunto de todas las palabras sobre este alfabeto (esto es, Σ^* es el conjunto de todas las cadenas, incluyendo a la cadena vacía, que consisten de miembros de Σ).

Máquinas de Turing VII

- Ahora suponga que Σ es un alfabeto finito (el símbolo B no cuenta como un miembro de Σ).
- Sea Σ^* el conjunto de todas las palabras sobre este alfabeto (esto es, Σ^* es el conjunto de todas las cadenas, incluyendo a la cadena vacía, que consisten de miembros de Σ).
- Suponga que f es una función parcial de aridad-k de Σ^* en Σ^*.

Máquinas de Turing VII

- Ahora suponga que Σ es un alfabeto finito (el símbolo B no cuenta como un miembro de Σ).
- Sea Σ^* el conjunto de todas las palabras sobre este alfabeto (esto es, Σ^* es el conjunto de todas las cadenas, incluyendo a la cadena vacía, que consisten de miembros de Σ).
- Suponga que f es una función parcial de aridad-k de Σ^* en Σ^*.
- Diremos que f es *computable a la Turing* si existe una máquina de Turing $\mathcal{M}$ que, cuando empieza en su estado inicial escaneando el primer símbolo de una tupla$-k$ $\vec{w}$ de palabras (escritas sobre la cinta, con un cuadrado blanco entre las palabras, y con el resto de la cinta en blanco), se comporta como sigue:

Máquinas de Turing VII

- Ahora suponga que Σ es un alfabeto finito (el símbolo B no cuenta como un miembro de Σ).
- Sea Σ^* el conjunto de todas las palabras sobre este alfabeto (esto es, Σ^* es el conjunto de todas las cadenas, incluyendo a la cadena vacía, que consisten de miembros de Σ).
- Suponga que f es una función parcial de aridad-k de Σ^* en Σ^*.
- Diremos que f es *computable a la Turing* si existe una máquina de Turing $\mathcal{M}$ que, cuando empieza en su estado inicial escaneando el primer símbolo de una tupla$-k$ $\vec{w}$ de palabras (escritas sobre la cinta, con un cuadrado blanco entre las palabras, y con el resto de la cinta en blanco), se comporta como sigue:
  - Si $f(\vec{w})\downarrow$ (es decir, si $\vec{w} \in \mathrm{dom} f$) entonces $\mathcal{M}$ eventualmente para, y en ese momento, está escaneando el símbolo más izquierdo de la palabra $f(\vec{w})$ (la cual es seguida por un cuadrado blanco).

Máquinas de Turing VII

- Ahora suponga que Σ es un alfabeto finito (el símbolo B no cuenta como un miembro de Σ).
- Sea Σ^* el conjunto de todas las palabras sobre este alfabeto (esto es, Σ^* es el conjunto de todas las cadenas, incluyendo a la cadena vacía, que consisten de miembros de Σ).
- Suponga que f es una función parcial de aridad-k de Σ^* en Σ^*.
- Diremos que f es *computable a la Turing* si existe una máquina de Turing $\mathcal{M}$ que, cuando empieza en su estado inicial escaneando el primer símbolo de una tupla$-k$ $\vec{w}$ de palabras (escritas sobre la cinta, con un cuadrado blanco entre las palabras, y con el resto de la cinta en blanco), se comporta como sigue:
  - Si $f(\vec{w})\downarrow$ (es decir, si $\vec{w} \in \mathrm{dom} f$) entonces $\mathcal{M}$ eventualmente para, y en ese momento, está escaneando el símbolo más izquierdo de la palabra $f(\vec{w})$ (la cual es seguida por un cuadrado blanco).
  - Si $f(\vec{w})\uparrow$ (es decir, si $\vec{w} \notin \mathrm{dom} f$) entonces $\mathcal{M}$ nunca para.

Ejemplo 1

Tome un alfabeto de dos letras $\Sigma = \{a, b\}$. Sea $\mathcal{M}$ la máquina de Turing dada por el siguiente conjunto con seis quintuplas[a], donde q_1 es designado como el símbolo inicial:

$$\langle q_1, a, a, R, q_1 \rangle$$
$$\langle q_1, b, b, R, q_1 \rangle$$
$$\langle q_1, B, a, L, q_2 \rangle$$
$$\langle q_2, a, a, L, q_2 \rangle$$
$$\langle q_2, b, b, L, q_2 \rangle$$
$$\langle q_2, B, B, R, q_3 \rangle$$

Suponga que empezamos esta máquina en el estado q_1, escaneando la primer letra de una palabra w. La máquina se mueve (en el estado q_1) hacia el final derecho de w, donde añade la letra a. Entonces se mueve (en el estado q_2) de regreso hacia el final izquierdo de la palabra, donde para (en el estado q_3). Así, $\mathcal{M}$ calcula la función total $f(w) = wa$.

---

[a] En el original hay un error, la quinta quintupla aparece como $\langle q_2, b, b, R, q_2 \rangle$. Diga cuál es el comportamiento de esa máquina y qué calcula. N. del T.

Máquinas de Turing VIII

- Necesitamos adoptar convenciones especiales para manejar la palabra vacía λ, la cual ocupa cero cuadrados.

Máquinas de Turing VIII

- Necesitamos adoptar convenciones especiales para manejar la palabra vacía λ, la cual ocupa cero cuadrados.
- Esto se puede hacer de diferentes maneras, la siguiente es la manera escogida, si la máquina para escaneando un cuadrado blanco, entonces la palabra de salida es λ.

Máquinas de Turing VIII

- Necesitamos adoptar convenciones especiales para manejar la palabra vacía λ, la cual ocupa cero cuadrados.
- Esto se puede hacer de diferentes maneras, la siguiente es la manera escogida, si la máquina para escaneando un cuadrado blanco, entonces la palabra de salida es λ.
- Para una función de aridad-uno f, para calcular $f(\lambda)$, simplemente iniciamos con una cinta en blanco.

Máquinas de Turing IX

- Para una función de aridad-dos g, para calcular $g(w, \lambda)$, iniciamos sólo con la palabra w, escaneando el primer símbolo de w.

Máquinas de Turing IX

- Para una función de aridad-dos g, para calcular $g(w, \lambda)$, iniciamos sólo con la palabra w, escaneando el primer símbolo de w.
- Y para calcular $g(\lambda, w)$, también empezamos con sólo la palabra w sobre la cinta, pero escaneando el cuadrado blanco justo a la izquierda de w.

Máquinas de Turing IX

- Para una función de aridad-dos g, para calcular $g(w, \lambda)$, iniciamos sólo con la palabra w, escaneando el primer símbolo de w.
- Y para calcular $g(\lambda, w)$, también empezamos con sólo la palabra w sobre la cinta, pero escaneando el cuadrado blanco justo a la izquierda de w.
- Y en general, para darle a una función de aridad-k la entrada $\vec{w} = \langle u_1, \ldots, u_k \rangle$ consistiendo de k palabras de longitudes $n_1, \ldots, n_k$,iniciamos la máquina escaneando el primer cuadrado de la configuración de entrada de longitud $n_1 + \cdots + n_k + k - 1$

  $(n_1$ símbolos de $u_1)B(n_2$ símbolos de $u_2)B \cdots B(n_k$ símbolos de $u_k)$

  con el resto de la cinta en blanco.

Máquinas de Turing IX

- Para una función de aridad-dos g, para calcular $g(w, \lambda)$, iniciamos sólo con la palabra w, escaneando el primer símbolo de w.
- Y para calcular $g(\lambda, w)$, también empezamos con sólo la palabra w sobre la cinta, pero escaneando el cuadrado blanco justo a la izquierda de w.
- Y en general, para darle a una función de aridad-k la entrada $\vec{w} = \langle u_1, \ldots, u_k \rangle$ consistiendo de k palabras de longitudes $n_1, \ldots, n_k$,iniciamos la máquina escaneando el primer cuadrado de la configuración de entrada de longitud $n_1 + \cdots + n_k + k - 1$

  $$(n_1 \text{ símbolos de } u_1)B(n_2 \text{ símbolos de } u_2)B \cdots B(n_k \text{ símbolos de } u_k)$$

  con el resto de la cinta en blanco.
- Aquí cualquier n_i puede ser cero; en el caso extremo, todos pueden ser cero.

Máquinas de Turing X

- Un inconveniente obvio de estas convenciones es que no existe diferencia entre el par $\langle u, v \rangle$ y el triple $\langle u, v, \lambda \rangle$.

Máquinas de Turing X

- Un inconveniente obvio de estas convenciones es que no existe diferencia entre el par $\langle u, v \rangle$ y el triple $\langle u, v, \lambda \rangle$.
- Otras convenciones evitan este inconveniente, a costa de introducir su propia idiosincracia.

Máquinas de Turing X

- Un inconveniente obvio de estas convenciones es que no existe diferencia entre el par $\langle u, v \rangle$ y el triple $\langle u, v, \lambda \rangle$.
- Otras convenciones evitan este inconveniente, a costa de introducir su propia idiosincracia.
- La definición de computabilidad a la Turing puede ser adaptada fácilmente a funciones parciales de aridad-k sobre $\mathbb{N}$.

Máquinas de Turing X

- Un inconveniente obvio de estas convenciones es que no existe diferencia entre el par $\langle u, v \rangle$ y el triple $\langle u, v, \lambda \rangle$.
- Otras convenciones evitan este inconveniente, a costa de introducir su propia idiosincracia.
- La definición de computabilidad a la Turing puede ser adaptada fácilmente a funciones parciales de aridad-k sobre $\mathbb{N}$.
- La manera más simple de hacerlo es usando numerales en base-1. Tomemos el alfabeto de una letra $\Sigma = \{|\}$ cuya única letra es el símbolo $|$.

Máquinas de Turing X

- Un inconveniente obvio de estas convenciones es que no existe diferencia entre el par $\langle u, v \rangle$ y el triple $\langle u, v, \lambda \rangle$.
- Otras convenciones evitan este inconveniente, a costa de introducir su propia idiosincracia.
- La definición de computabilidad a la Turing puede ser adaptada fácilmente a funciones parciales de aridad-k sobre $\mathbb{N}$.
- La manera más simple de hacerlo es usando numerales en base-1. Tomemos el alfabeto de una letra $\Sigma = \{|\}$ cuya única letra es el símbolo $|$.
- O para ser más convencionales, sea $\Sigma = \{1\}$, usando el símbolo 1 en lugar de $|$.

Máquinas de Turing X

- Un inconveniente obvio de estas convenciones es que no existe diferencia entre el par $\langle u, v \rangle$ y el triple $\langle u, v, \lambda \rangle$.
- Otras convenciones evitan este inconveniente, a costa de introducir su propia idiosincracia.
- La definición de computabilidad a la Turing puede ser adaptada fácilmente a funciones parciales de aridad-k sobre $\mathbb{N}$.
- La manera más simple de hacerlo es usando numerales en base-1. Tomemos el alfabeto de una letra $\Sigma = \{|\}$ cuya única letra es el símbolo $|$.
- O para ser más convencionales, sea $\Sigma = \{1\}$, usando el símbolo 1 en lugar de $|$.
- Entonces la configuración inicial para el triple $\langle 3, 0, 4 \rangle$ es

$$111BB1111$$

Tesis de Church-Turing I

- Entonces la *tesis de Church*, también llamada -particularmente en el contexto de máquinas de Turing- *tesis de Church-Turing*, es la afirmación de que este concepto de computabilidad a la Turing es la formalización correcta del concepto informal de calculabilidad efectiva.

Tesis de Church-Turing I

- Entonces la *tesis de Church*, también llamada -particularmente en el contexto de máquinas de Turing- *tesis de Church-Turing*, es la afirmación de que este concepto de computabilidad a la Turing es la formalización correcta del concepto informal de calculabilidad efectiva.
- Ciertamente la definición refleja la idea de seguir instrucciones predeterminadas, sin limitar la cantidad de tiempo que puede requerir.

Tesis de Church-Turing I

- Entonces la *tesis de Church*, también llamada -particularmente en el contexto de máquinas de Turing- *tesis de Church-Turing*, es la afirmación de que este concepto de computabilidad a la Turing es la formalización correcta del concepto informal de calculabilidad efectiva.
- Ciertamente la definición refleja la idea de seguir instrucciones predeterminadas, sin limitar la cantidad de tiempo que puede requerir.
- El nombre "tesis de Church-Turing" obscurece el hecho de que Church y Turing siguieron caminos muy distintos para llegar a conclusiones equivalentes.

Tesis de Church-Turing I

- Entonces la *tesis de Church*, también llamada -particularmente en el contexto de máquinas de Turing- *tesis de Church-Turing*, es la afirmación de que este concepto de computabilidad a la Turing es la formalización correcta del concepto informal de calculabilidad efectiva.
- Ciertamente la definición refleja la idea de seguir instrucciones predeterminadas, sin limitar la cantidad de tiempo que puede requerir.
- El nombre "tesis de Church-Turing" obscurece el hecho de que Church y Turing siguieron caminos muy distintos para llegar a conclusiones equivalentes.
- La tesis de Church ha alcanzado aceptación universal. Kurt Gödel, escribiendo en 1964 sobre el concepto de "sistema formal" en lógica, implicando la idea de que el conjunto de deducciones correctas debe ser un conjunto decidible, dijo que "debido al trabajo de A. M. Turing, ahora se puede dar una definición precisa e incuestionablemente adecuada del concepto general de sistema formal".

Tesis de Church-Turing II

- Y otros concuerdan.

Tesis de Church-Turing II

- Y otros concuerdan.
- La robustez del concepto de computabilidad a la Turing se evidencia por el hecho de que es insensible a ciertas modificaciones a la definición de una máquina de Turing.

Tesis de Church-Turing II

- Y otros concuerdan.
- La robustez del concepto de computabilidad a la Turing se evidencia por el hecho de que es insensible a ciertas modificaciones a la definición de una máquina de Turing.
- Por ejemplo, podemos imponer limitaciones al tamaño del alfabeto, o podemos insistir en que la máquina nunca se mueva a la izquierda de su punto de inicio.

Tesis de Church-Turing II

- Y otros concuerdan.
- La robustez del concepto de computabilidad a la Turing se evidencia por el hecho de que es insensible a ciertas modificaciones a la definición de una máquina de Turing.
- Por ejemplo, podemos imponer limitaciones al tamaño del alfabeto, o podemos insistir en que la máquina nunca se mueva a la izquierda de su punto de inicio.
- Ninguna de éstas afectará a la clase de funciones parciales computables a la Turing.

Tesis de Church-Turing III

- Turing desarrollo estas ideas antes de la introducción de las computadoras modernas digitales.

Tesis de Church-Turing III

- Turing desarrollo estas ideas antes de la introducción de las computadoras modernas digitales.
- Después de la segunda guerra mundial, Turing jugó un papel activo en el desarrollo de las primeras computadoras, y en el campo emergente de la inteligencia artificial.

Tesis de Church-Turing III

- Turing desarrollo estas ideas antes de la introducción de las computadoras modernas digitales.
- Después de la segunda guerra mundial, Turing jugó un papel activo en el desarrollo de las primeras computadoras, y en el campo emergente de la inteligencia artificial.
- Durante la guerra, trabajó descifrando el código Enigma usado por los Alemanes en el campo de batalla, trabajo militarmente importante, el cual permaneció clasificado hasta después de la muerte de Turing.

Tesis de Church-Turing III

- Turing desarrollo estas ideas antes de la introducción de las computadoras modernas digitales.
- Después de la segunda guerra mundial, Turing jugó un papel activo en el desarrollo de las primeras computadoras, y en el campo emergente de la inteligencia artificial.
- Durante la guerra, trabajó descifrando el código Enigma usado por los Alemanes en el campo de batalla, trabajo militarmente importante, el cual permaneció clasificado hasta después de la muerte de Turing.
- Uno puede especular si Turing habría formulado sus ideas de manera diferente, si su trabajo lo hubiera hecho después de la introducción de las computadoras digitales.

El problema del castor ocupado I

- Suponga que queremos una máquina de Turing, iniciando con su cinta en blanco, que escriba tantos unos como pueda, y luego pare.

El problema del castor ocupado I

- Suponga que queremos una máquina de Turing, iniciando con su cinta en blanco, que escriba tantos unos como pueda, y luego pare.
- Con un número limitado de estados, ¿cuántos 1's podemos obtener?

El problema del castor ocupado I

- Suponga que queremos una máquina de Turing, iniciando con su cinta en blanco, que escriba tantos unos como pueda, y luego pare.
- Con un número limitado de estados, ¿cuántos 1's podemos obtener?
- Para hacer las cosas más precisas, tome máquinas de Turing con el alfabeto $\{1\}$ (así, los únicos símbolos son B y 1).

El problema del castor ocupado I

- Suponga que queremos una máquina de Turing, iniciando con su cinta en blanco, que escriba tantos unos como pueda, y luego pare.
- Con un número limitado de estados, ¿cuántos 1's podemos obtener?
- Para hacer las cosas más precisas, tome máquinas de Turing con el alfabeto $\{1\}$ (así, los únicos símbolos son B y 1).
- Permitiremos que tales máquinas tengan n estados, más un estado de paro (puede ocurrir como el último miembro de una quintupla, pero no como el primer miembro).

El problema del castor ocupado I

- Suponga que queremos una máquina de Turing, iniciando con su cinta en blanco, que escriba tantos unos como pueda, y luego pare.
- Con un número limitado de estados, ¿cuántos 1's podemos obtener?
- Para hacer las cosas más precisas, tome máquinas de Turing con el alfabeto $\{1\}$ (así, los únicos símbolos son B y 1).
- Permitiremos que tales máquinas tengan n estados, más un estado de paro (puede ocurrir como el último miembro de una quintupla, pero no como el primer miembro).
- Para cada n, sólo existen finitas máquinas de Turing esencialmente distintas.

El problema del castor ocupado I

- Suponga que queremos una máquina de Turing, iniciando con su cinta en blanco, que escriba tantos unos como pueda, y luego pare.
- Con un número limitado de estados, ¿cuántos 1's podemos obtener?
- Para hacer las cosas más precisas, tome máquinas de Turing con el alfabeto $\{1\}$ (así, los únicos símbolos son B y 1).
- Permitiremos que tales máquinas tengan n estados, más un estado de paro (puede ocurrir como el último miembro de una quintupla, pero no como el primer miembro).
- Para cada n, sólo existen finitas máquinas de Turing esencialmente distintas.
- Algunas de ellas, iniciando con la cinta en blanco, podrían no parar.

El problema del castor ocupado I

- Suponga que queremos una máquina de Turing, iniciando con su cinta en blanco, que escriba tantos unos como pueda, y luego pare.
- Con un número limitado de estados, ¿cuántos 1's podemos obtener?
- Para hacer las cosas más precisas, tome máquinas de Turing con el alfabeto $\{1\}$ (así, los únicos símbolos son B y 1).
- Permitiremos que tales máquinas tengan n estados, más un estado de paro (puede ocurrir como el último miembro de una quintupla, pero no como el primer miembro).
- Para cada n, sólo existen finitas máquinas de Turing esencialmente distintas.
- Algunas de ellas, iniciando con la cinta en blanco, podrían no parar.
- Por ejemplo, la máquina de un-estado

$$\langle q_1, B, 1, R, q_1 \rangle$$

sigue escribiendo por siempre sin parar.

El problema del castor ocupado I

- Suponga que queremos una máquina de Turing, iniciando con su cinta en blanco, que escriba tantos unos como pueda, y luego pare.
- Con un número limitado de estados, ¿cuántos 1's podemos obtener?
- Para hacer las cosas más precisas, tome máquinas de Turing con el alfabeto $\{1\}$ (así, los únicos símbolos son B y 1).
- Permitiremos que tales máquinas tengan n estados, más un estado de paro (puede ocurrir como el último miembro de una quintupla, pero no como el primer miembro).
- Para cada n, sólo existen finitas máquinas de Turing esencialmente distintas.
- Algunas de ellas, iniciando con la cinta en blanco, podrían no parar.
- Por ejemplo, la máquina de un-estado

$$\langle q_1, B, 1, R, q_1 \rangle$$

  sigue escribiendo por siempre sin parar.
- Pero entre aquellas que paran, buscamos las que escriben muchos 1's.

El problema del castor ocupado II

- Defina $\sigma(n)$ como el número más grande de 1's que pueden escribirse por una máquina de Turing de n estados como se describió aquí antes de que pare.

El problema del castor ocupado II

- Defina $\sigma(n)$ como el número más grande de 1's que pueden escribirse por una máquina de Turing de n estados como se describió aquí antes de que pare.
- Por ejemplo, $\sigma(1) = 1$, ya que la máquina de un-estado

$$\langle q_1, B, 1, R, q_H \rangle$$

(el estado de paro q_H no cuenta) escribe un 1, y ninguna de las otras máquinas de Turing de un-estado lo hacen mejor.

El problema del castor ocupado II

- Defina $\sigma(n)$ como el número más grande de 1's que pueden escribirse por una máquina de Turing de n estados como se describió aquí antes de que pare.
- Por ejemplo, $\sigma(1) = 1$, ya que la máquina de un-estado

$$\langle q_1, B, 1, R, q_H \rangle$$

(el estado de paro q_H no cuenta) escribe un 1, y ninguna de las otras máquinas de Turing de un-estado lo hacen mejor.

- No hay muchas otras máquinas de un-estado, y uno puede examinarlas todas en un plazo de tiempo razonable.

El problema del castor ocupado II

- Defina $\sigma(n)$ como el número más grande de 1's que pueden escribirse por una máquina de Turing de n estados como se describió aquí antes de que pare.
- Por ejemplo, $\sigma(1) = 1$, ya que la máquina de un-estado

$$\langle q_1, B, 1, R, q_H \rangle$$

  (el estado de paro q_H no cuenta) escribe un 1, y ninguna de las otras máquinas de Turing de un-estado lo hacen mejor.
- No hay muchas otras máquinas de un-estado, y uno puede examinarlas todas en un plazo de tiempo razonable.
- Acordemos que $\sigma(0) = 0$. Entonces σ es una función total.

El problema del castor ocupado II

- Defina $\sigma(n)$ como el número más grande de 1's que pueden escribirse por una máquina de Turing de n estados como se describió aquí antes de que pare.
- Por ejemplo, $\sigma(1) = 1$, ya que la máquina de un-estado

$$\langle q_1, B, 1, R, q_H \rangle$$

  (el estado de paro q_H no cuenta) escribe un 1, y ninguna de las otras máquinas de Turing de un-estado lo hacen mejor.
- No hay muchas otras máquinas de un-estado, y uno puede examinarlas todas en un plazo de tiempo razonable.
- Acordemos que $\sigma(0) = 0$. Entonces σ es una función total.
- También es no decreciente ya que no es un impedimento tener un estado extra para trabajar.

El problema del castor ocupado III

- A pesar del hecho de que $\sigma(n)$ es meramente el miembro más grande de cierto conjunto finito, no existe un algoritmo que nos permita, en general, evaluarlo.

El problema del castor ocupado III

- A pesar del hecho de que $\sigma(n)$ es meramente el miembro más grande de cierto conjunto finito, no existe un algoritmo que nos permita, en general, evaluarlo.

El problema del castor ocupado III

- A pesar del hecho de que $\sigma(n)$ es meramente el miembro más grande de cierto conjunto finito, no existe un algoritmo que nos permita, en general, evaluarlo.

Ejemplo 2

Aquí hay un candidato de dos-estados:

$$\langle q_1, B, 1, R, q_2 \rangle$$
$$\langle q_1, 1, 1, L, q_2 \rangle$$
$$\langle q_2, B, 1, L, q_1 \rangle$$
$$\langle q_2, 1, 1, R, q_H \rangle$$

Empezando con una cinta en blanco, esta máquina escribe cuatro 1's consecutivos, y luego para (después de seis pasos), leyendo el tercer 1. Está invitado a verificarlo corriendo la máquina. Concluimos que $\sigma(2) \geq 4$.

Teorema de Rado I

Teorema 1

Teorema de Rado (1962): La función σ no es computable a la Turing. Aún más, para cualquier función f computable a la Turing, tenemos que $f(x) < \sigma(x)$ para todo x suficientemente grande. Esto es, σ domina eventualmente a cualquier función total computable a la Turing.

Teorema de Rado II

- *Prueba:* Asuma que hemos dado alguna función f computable a la Turing.

Teorema de Rado II

- *Prueba:* Asuma que hemos dado alguna función f computable a la Turing.
- Debemos mostrar que σ eventualmente la domina.

Teorema de Rado II

- *Prueba:* Asuma que hemos dado alguna función f computable a la Turing.
- Debemos mostrar que σ eventualmente la domina.
- Defina (por razones que pueden parecer inicialmente misteriosas) la función g:

$$g(x) = \text{máx}(f(2x), f(2x+1)) + 1.$$

Teorema de Rado II

- *Prueba:* Asuma que hemos dado alguna función f computable a la Turing.
- Debemos mostrar que σ eventualmente la domina.
- Defina (por razones que pueden parecer inicialmente misteriosas) la función g:

$$g(x) = \text{máx}(f(2x), f(2x+1)) + 1.$$

- Entonces g es total y uno puede mostrar que es computable a la Turing.

Teorema de Rado II

- *Prueba:* Asuma que hemos dado alguna función f computable a la Turing.
- Debemos mostrar que σ eventualmente la domina.
- Defina (por razones que pueden parecer inicialmente misteriosas) la función g:

$$g(x) = \text{máx}(f(2x), f(2x+1)) + 1.$$

- Entonces g es total y uno puede mostrar que es computable a la Turing.
- Así existe alguna máquina de Turing $\mathcal{M}$ con, digamos, k estados que la computa, usando el alfabeto $\{1\}$ y notación base-1.

Teorema de Rado II

- *Prueba:* Asuma que hemos dado alguna función f computable a la Turing.
- Debemos mostrar que σ eventualmente la domina.
- Defina (por razones que pueden parecer inicialmente misteriosas) la función g:

$$g(x) = \text{máx}(f(2x), f(2x+1)) + 1.$$

- Entonces g es total y uno puede mostrar que es computable a la Turing.
- Así existe alguna máquina de Turing $\mathcal{M}$ con, digamos, k estados que la computa, usando el alfabeto $\{1\}$ y notación base-1.
- Para cada x, sea $\mathcal{N}_x$ la máquina de Turing de $(x+k)$ estados que primero escribe x 1's sobre la cinta, y luego imita a $\mathcal{M}$.

Teorema de Rado II

- *Prueba:* Asuma que hemos dado alguna función f computable a la Turing.
- Debemos mostrar que σ eventualmente la domina.
- Defina (por razones que pueden parecer inicialmente misteriosas) la función g:

$$g(x) = \text{máx}(f(2x), f(2x+1)) + 1.$$

- Entonces g es total y uno puede mostrar que es computable a la Turing.
- Así existe alguna máquina de Turing $\mathcal{M}$ con, digamos, k estados que la computa, usando el alfabeto $\{1\}$ y notación base-1.
- Para cada x, sea $\mathcal{N}_x$ la máquina de Turing de $(x+k)$ estados que primero escribe x 1's sobre la cinta, y luego imita a $\mathcal{M}$.
- Los x estados nos permiten escribir x 1's en la cinta de manera directa, y luego están los k estados de $\mathcal{M}$.

Teorema de Rado III

- Entonces, $\mathcal{N}_x$, cuando inicia con una cinta en blanco, escribe $g(x)$ 1's en la cinta y para.

Teorema de Rado III

- Entonces, $\mathcal{N}_x$, cuando inicia con una cinta en blanco, escribe $g(x)$ 1's en la cinta y para.
- Así, $g(x) \leq \sigma(x + k)$, por la definición de σ.

Teorema de Rado III

- Entonces, $\mathcal{N}_x$, cuando inicia con una cinta en blanco, escribe $g(x)$ 1's en la cinta y para.
- Así, $g(x) \leq \sigma(x+k)$, por la definición de σ.
- Así, tenemos

$$f(2x), f(2x+1) < g(x) \leq \sigma(x+k),$$

y si $x \geq k$, entonces

$$\sigma(x+k) \leq \sigma(2x) \leq \sigma(2x+1).$$

Teorema de Rado III

- Entonces, $\mathcal{N}_x$, cuando inicia con una cinta en blanco, escribe $g(x)$ 1's en la cinta y para.
- Así, $g(x) \leq \sigma(x+k)$, por la definición de σ.
- Así, tenemos

$$f(2x), f(2x+1) < g(x) \leq \sigma(x+k),$$

y si $x \geq k$, entonces

$$\sigma(x+k) \leq \sigma(2x) \leq \sigma(2x+1).$$

- Poniendo las dos líneas juntas, vemos que $f < \sigma$ de $2k$ en adelante. $\dashv$

Teorema de Rado IV

- Así σ crece más, eventualmente, que cualquier función total computable a la Turing.

Teorema de Rado IV

- Así σ crece más, eventualmente, que cualquier función total computable a la Turing.
- ¿Qué tan rápido crece?

Teorema de Rado IV

- Así σ crece más, eventualmente, que cualquier función total computable a la Turing.
- ¿Qué tan rápido crece?
- Entre los números más pequeños, $\sigma(2) = 4$.

Teorema de Rado IV

- Así σ crece más, eventualmente, que cualquier función total computable a la Turing.
- ¿Qué tan rápido crece?
- Entre los números más pequeños, $\sigma(2) = 4$.
- El ejemplo precedente mostró que $\sigma(2) \geq 4$.

Teorema de Rado IV

- Así σ crece más, eventualmente, que cualquier función total computable a la Turing.
- ¿Qué tan rápido crece?
- Entre los números más pequeños, $\sigma(2) = 4$.
- El ejemplo precedente mostró que $\sigma(2) \geq 4$.
- La otra desigualdad no es enteramente trivial ya que existen miles de máquinas de dos-estados.

Teorema de Rado IV

- Así σ crece más, eventualmente, que cualquier función total computable a la Turing.
- ¿Qué tan rápido crece?
- Entre los números más pequeños, $\sigma(2) = 4$.
- El ejemplo precedente mostró que $\sigma(2) \geq 4$.
- La otra desigualdad no es enteramente trivial ya que existen miles de máquinas de dos-estados.
- También se ha mostrado que $\sigma(3) = 6$ y $\sigma(4) = 13$.

Teorema de Rado IV

- Así σ crece más, eventualmente, que cualquier función total computable a la Turing.
- ¿Qué tan rápido crece?
- Entre los números más pequeños, $\sigma(2) = 4$.
- El ejemplo precedente mostró que $\sigma(2) \geq 4$.
- La otra desigualdad no es enteramente trivial ya que existen miles de máquinas de dos-estados.
- También se ha mostrado que $\sigma(3) = 6$ y $\sigma(4) = 13$.
- De aquí en adelante, sólo son conocidas cotas inferiores.

Teorema de Rado IV

- Así σ crece más, eventualmente, que cualquier función total computable a la Turing.
- ¿Qué tan rápido crece?
- Entre los números más pequeños, $\sigma(2) = 4$.
- El ejemplo precedente mostró que $\sigma(2) \geq 4$.
- La otra desigualdad no es enteramente trivial ya que existen miles de máquinas de dos-estados.
- También se ha mostrado que $\sigma(3) = 6$ y $\sigma(4) = 13$.
- De aquí en adelante, sólo son conocidas cotas inferiores.
- En 1984, se encontró que $\sigma(5)$ es al menos 1915.

Teorema de Rado IV

- Así σ crece más, eventualmente, que cualquier función total computable a la Turing.
- ¿Qué tan rápido crece?
- Entre los números más pequeños, $\sigma(2) = 4$.
- El ejemplo precedente mostró que $\sigma(2) \geq 4$.
- La otra desigualdad no es enteramente trivial ya que existen miles de máquinas de dos-estados.
- También se ha mostrado que $\sigma(3) = 6$ y $\sigma(4) = 13$.
- De aquí en adelante, sólo son conocidas cotas inferiores.
- En 1984, se encontró que $\sigma(5)$ es al menos 1915.
- En 1990, se elevó a 4098.

Teorema de Rado IV

- Así σ crece más, eventualmente, que cualquier función total computable a la Turing.
- ¿Qué tan rápido crece?
- Entre los números más pequeños, $\sigma(2) = 4$.
- El ejemplo precedente mostró que $\sigma(2) \geq 4$.
- La otra desigualdad no es enteramente trivial ya que existen miles de máquinas de dos-estados.
- También se ha mostrado que $\sigma(3) = 6$ y $\sigma(4) = 13$.
- De aquí en adelante, sólo son conocidas cotas inferiores.
- En 1984, se encontró que $\sigma(5)$ es al menos 1915.
- En 1990, se elevó a 4098.
- Y $\sigma(6) > 3.1 \times 10^{10566}$.

Teorema de Rado IV

- Así σ crece más, eventualmente, que cualquier función total computable a la Turing.
- ¿Qué tan rápido crece?
- Entre los números más pequeños, $\sigma(2) = 4$.
- El ejemplo precedente mostró que $\sigma(2) \geq 4$.
- La otra desigualdad no es enteramente trivial ya que existen miles de máquinas de dos-estados.
- También se ha mostrado que $\sigma(3) = 6$ y $\sigma(4) = 13$.
- De aquí en adelante, sólo son conocidas cotas inferiores.
- En 1984, se encontró que $\sigma(5)$ es al menos 1915.
- En 1990, se elevó a 4098.
- Y $\sigma(6) > 3.1 \times 10^{10566}$.
- Y $\sigma(7)$ debe ser astronómico.

Teorema de Rado V

- Estas cotas inferiores se establecen usando codificaciones ingeniosamente complejas para crear pequeñas máquinas de Turing que escriben muchos 1's y luego paran.

Teorema de Rado V

- Estas cotas inferiores se establecen usando codificaciones ingeniosamente complejas para crear pequeñas máquinas de Turing que escriben muchos 1's y luego paran.
- Probar además cotas superiores sería difícil.

Teorema de Rado V

- Estas cotas inferiores se establecen usando codificaciones ingeniosamente complejas para crear pequeñas máquinas de Turing que escriben muchos 1's y luego paran.
- Probar además cotas superiores sería difícil.
- En efecto, uno puede mostrar, bajo algunas suposiciones razonables, que cotas superiores para $\sigma(n)$ son probables sólo para finitos n's.

Teorema de Rado VI

- Si pudieramos solucionar el problema de paro, tendríamos el siguiente método para calcular $\sigma(n)$:

Teorema de Rado VI

- Si pudieramos solucionar el problema de paro, tendríamos el siguiente método para calcular $\sigma(n)$:
  1. Lista todas las máquinas de n-estados.

Teorema de Rado VI

- Si pudieramos solucionar el problema de paro, tendríamos el siguiente método para calcular $\sigma(n)$:
  1. Lista todas las máquinas de n-estados.
  2. Descarta aquellas que nunca paran.

Teorema de Rado VI

- Si pudieramos solucionar el problema de paro, tendríamos el siguiente método para calcular $\sigma(n)$:
  1. Lista todas las máquinas de n-estados.
  2. Descarta aquellas que nunca paran.
  3. Ejecuta aquellas que paran.

Teorema de Rado VI

- Si pudieramos solucionar el problema de paro, tendríamos el siguiente método para calcular $\sigma(n)$:
  1. Lista todas las máquinas de n-estados.
  2. Descarta aquellas que nunca paran.
  3. Ejecuta aquellas que paran.
  4. Selecciona la puntuación más alta.

Teorema de Rado VI

- Si pudieramos solucionar el problema de paro, tendríamos el siguiente método para calcular $\sigma(n)$:
  1. Lista todas las máquinas de n-estados.
  2. Descarta aquellas que nunca paran.
  3. Ejecuta aquellas que paran.
  4. Selecciona la puntuación más alta.
- El segundo paso es el que nos da problemas.

Teorema de Rado VI

- Si pudieramos solucionar el problema de paro, tendríamos el siguiente método para calcular $\sigma(n)$:
  1. Lista todas las máquinas de n-estados.
  2. Descarta aquellas que nunca paran.
  3. Ejecuta aquellas que paran.
  4. Selecciona la puntuación más alta.
- El segundo paso es el que nos da problemas.
- Nueva información sobre la función σ de Rado sigue descubriéndose.

Teorema de Rado VI

- Si pudieramos solucionar el problema de paro, tendríamos el siguiente método para calcular $\sigma(n)$:
  1. Lista todas las máquinas de n-estados.
  2. Descarta aquellas que nunca paran.
  3. Ejecuta aquellas que paran.
  4. Selecciona la puntuación más alta.
- El segundo paso es el que nos da problemas.
- Nueva información sobre la función σ de Rado sigue descubriéndose.
- Noticias recientes se pueden obtener de la página Web mantenida por Heiner Marxen, `https://turbotm.de/~heiner/BB/index.html`.