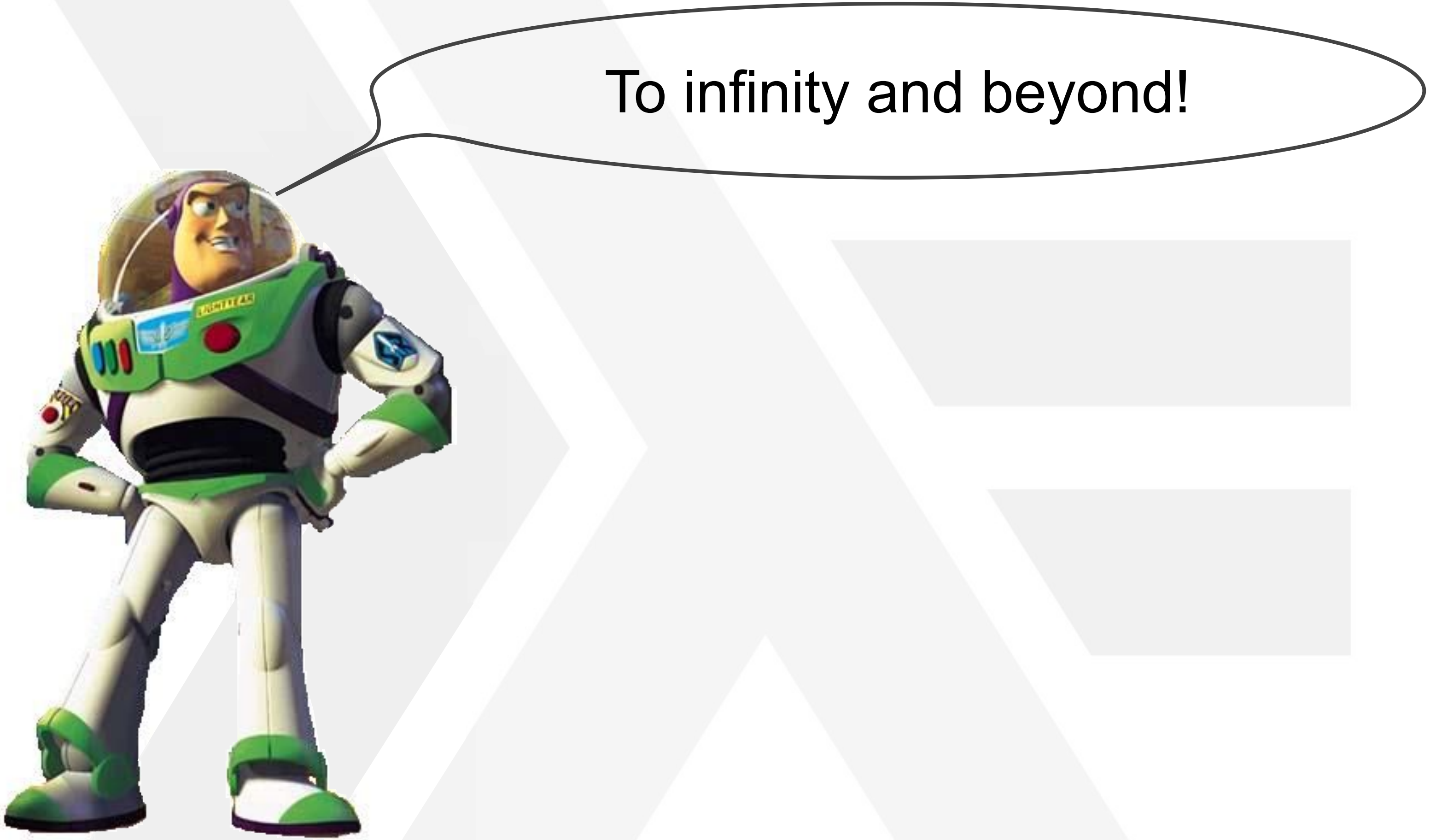
Programación Funcional

Clases teóricas

por Pablo E. “Fidel” Martínez López

7. Inducción y recursión I


To infinity and beyond!

Motivación de inducción y recursión

Motivación de inducción y recursión

- Quedaron planteadas 3 preguntas
  - ¿Cómo definir tipos algebraicos con infinitos elementos?
  - ¿Cómo saber si las ecuaciones son orientadas?
  - ¿Qué falta para poder demostrar en estos casos?

Motivación de inducción y recursión

- Quedaron planteadas 3 preguntas
  - ¿Cómo definir tipos algebraicos con infinitos elementos?
  - ¿Cómo saber si las ecuaciones son orientadas?
  - ¿Qué falta para poder demostrar en estos casos?
- Parece ser necesaria ***inducción*** o ***recursión***
  - ¿Realmente entendemos la inducción matemática?
  - ¿Qué es la inducción? ¿Qué es la recursión?

Inducción estructural

Inducción estructural

- La ***inducción estructural*** es una *técnica* que responde las 3 preguntas planteadas

Inducción estructural

- La ***inducción estructural*** es una *técnica* que responde las 3 preguntas planteadas
  - Permite
    - Definir tipos algebraicos con infinitos elementos

Inducción estructural

- La ***inducción estructural*** es una *técnica* que responde las 3 preguntas planteadas
  - Permite
    - Definir tipos algebraicos con infinitos elementos
    - Definir ecuaciones *orientadas* que usan la función que se está definiendo del lado derecho

Inducción estructural

- La ***inducción estructural*** es una *técnica* que responde las 3 preguntas planteadas
  - Permite
    - Definir tipos algebraicos con infinitos elementos
    - Definir ecuaciones orientadas que usan la función que se está definiendo del lado derecho
    - Probar propiedades sobre elementos de estos conjuntos

Inducción estructural

- La ***inducción estructural*** es una *técnica* que responde las 3 preguntas planteadas
  - Permite
    - Definir tipos algebraicos con infinitos elementos
    - Definir ecuaciones orientadas que usan la función que se está definiendo del lado derecho
    - Probar propiedades sobre elementos de estos conjuntos
- Operacionalmente la llamamos ***recursión estructural***

Inducción estructural

- Ejemplo (dejando Haskell de lado por ahora)
  - Definir el conjunto $\aleph$ de cadenas de Ss terminadas en Z
  - ¿Cómo hacerlo constructivamente?

Inducción estructural

- Ejemplo (dejando Haskell de lado por ahora)
  - Definir el conjunto $\aleph$ de cadenas de Ss terminadas en Z
  - ¿Cómo hacerlo constructivamente?
  - Podrían ponerse condiciones al conjunto buscado…
    - **Condición 1:** Z tiene que estar en $\aleph$ (Tiene 0 Ss)
    - **Condición 2:** si **c** está en $\aleph$, entonces S**c** tiene que estar en $\aleph$

Inducción estructural

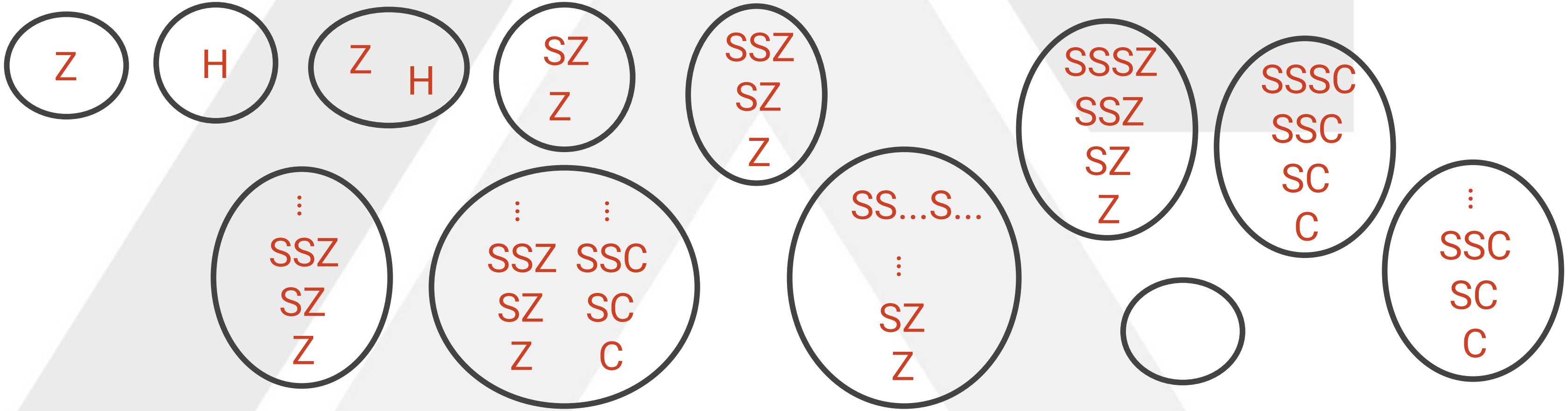
- Ejemplo (dejando Haskell de lado por ahora)
  - Definir el conjunto ℵ de cadenas de Ss terminadas en Z
  - ¿Cómo hacerlo constructivamente?
  - Podrían ponerse condiciones al conjunto buscado…
    - **Condición 1:** Z tiene que estar en ℵ (Tiene 0 Ss)
    - **Condición 2:** si **c** está en ℵ, entonces S**c** tiene que estar en ℵ
  - ¿Cómo saber si un conjunto dado es ℵ?

Inducción estructural

- ¿Cómo saber qué conjuntos satisfacen las condiciones?
  - **Condición 1:** Z tiene que estar en $\aleph$
  - **Condición 2:** si **c** está en $\aleph$, entonces S**c** tiene que estar en $\aleph$
- Analicemos algunos conjuntos posibles

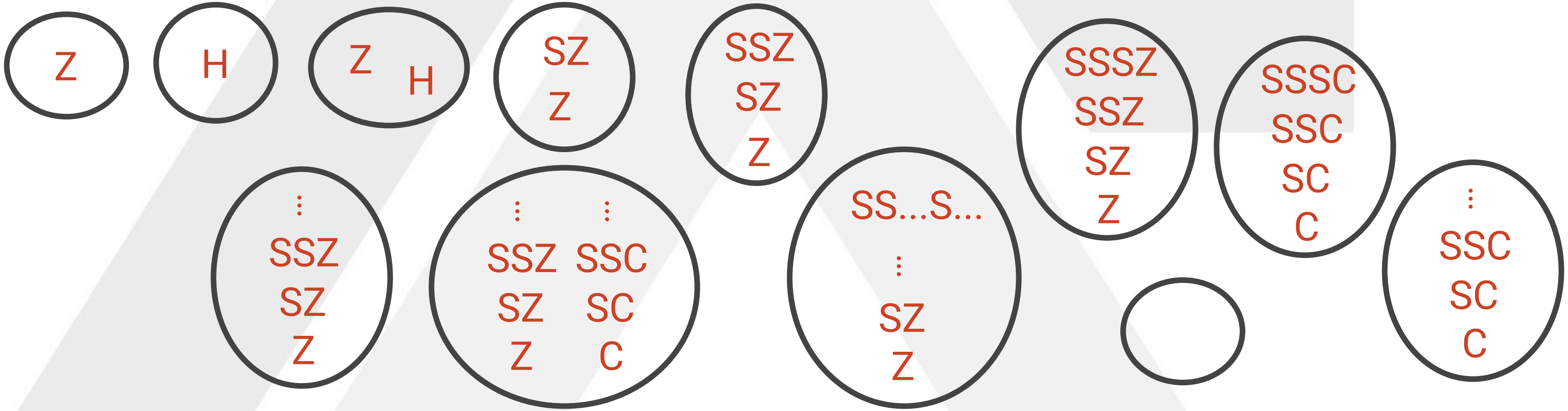
Inducción estructural

- ¿Cómo saber qué conjuntos satisfacen las condiciones?
  - **Condición 1:** Z tiene que estar en $\aleph$
  - **Condición 2:** si **c** está en $\aleph$, entonces S**c** tiene que estar en $\aleph$
- Analicemos algunos conjuntos posibles



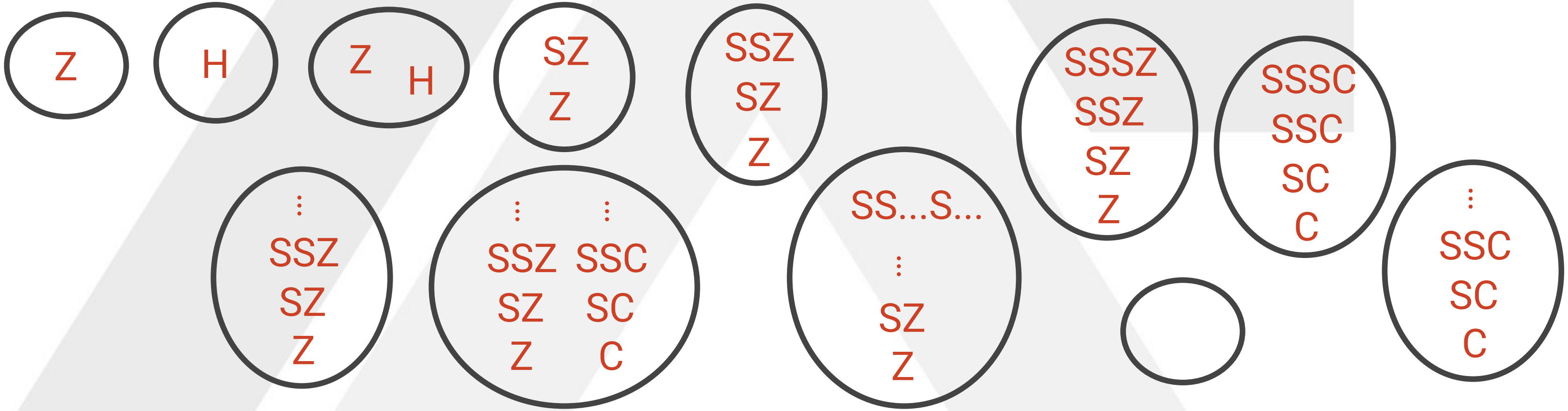
Inducción estructural

- ¿Cómo saber qué conjuntos satisfacen las condiciones?
  - **Condición 1:** Z tiene que estar en $\aleph$
  - **Condición 2:** si **c** está en $\aleph$, entonces S**c** tiene que estar en $\aleph$
- Analicemos algunos conjuntos posibles y vayamos descartando



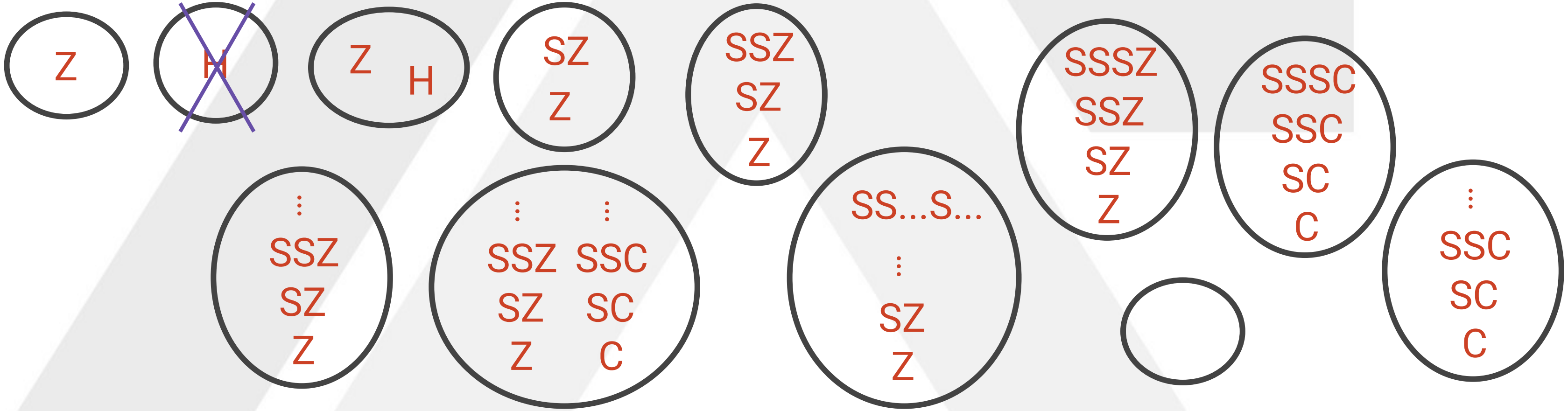
Inducción estructural

- ¿Cómo saber qué conjuntos satisfacen las condiciones?
  - **Condición 1:** Z tiene que estar en ℵ
  - **Condición 2:** si **c** está en ℵ, entonces S**c** tiene que estar en ℵ
- Analicemos algunos conjuntos posibles y vayamos descartando



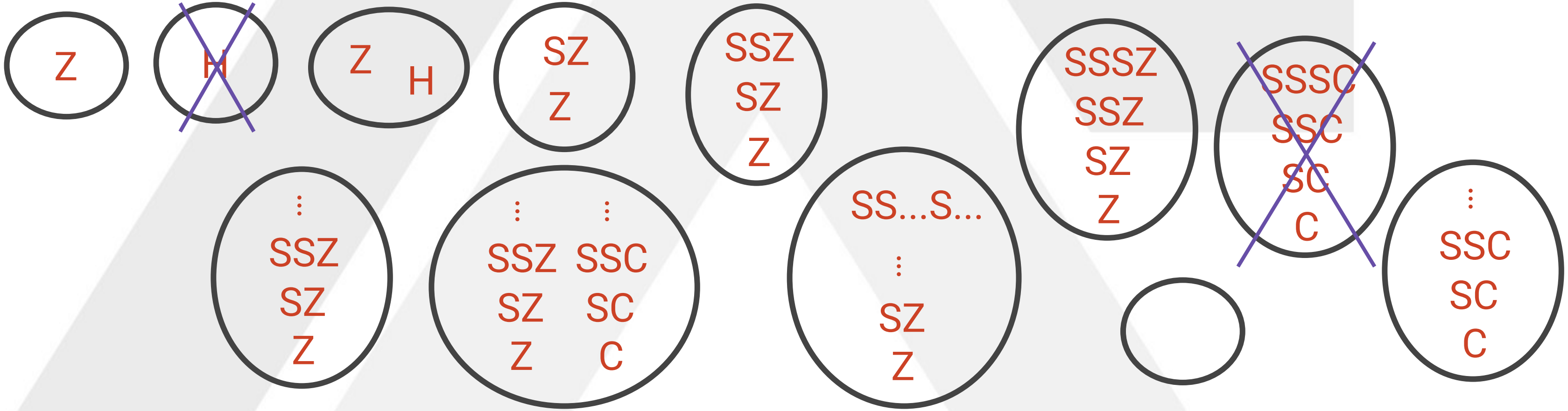
Inducción estructural

- ¿Cómo saber qué conjuntos satisfacen las condiciones?
  - **Condición 1:** Z tiene que estar en $\aleph$
  - **Condición 2:** si **c** está en $\aleph$, entonces S**c** tiene que estar en $\aleph$
- Analicemos algunos conjuntos posibles y vayamos descartando



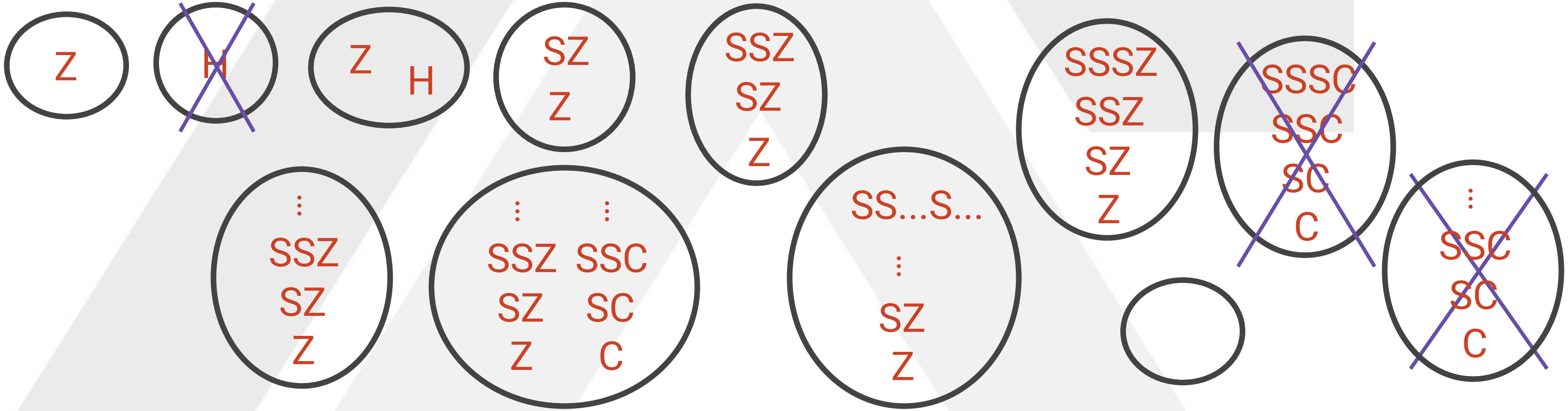
Inducción estructural

- ¿Cómo saber qué conjuntos satisfacen las condiciones?
  - **Condición 1:** Z tiene que estar en $\aleph$
  - **Condición 2:** si **c** está en $\aleph$, entonces S**c** tiene que estar en $\aleph$
- Analicemos algunos conjuntos posibles y vayamos descartando



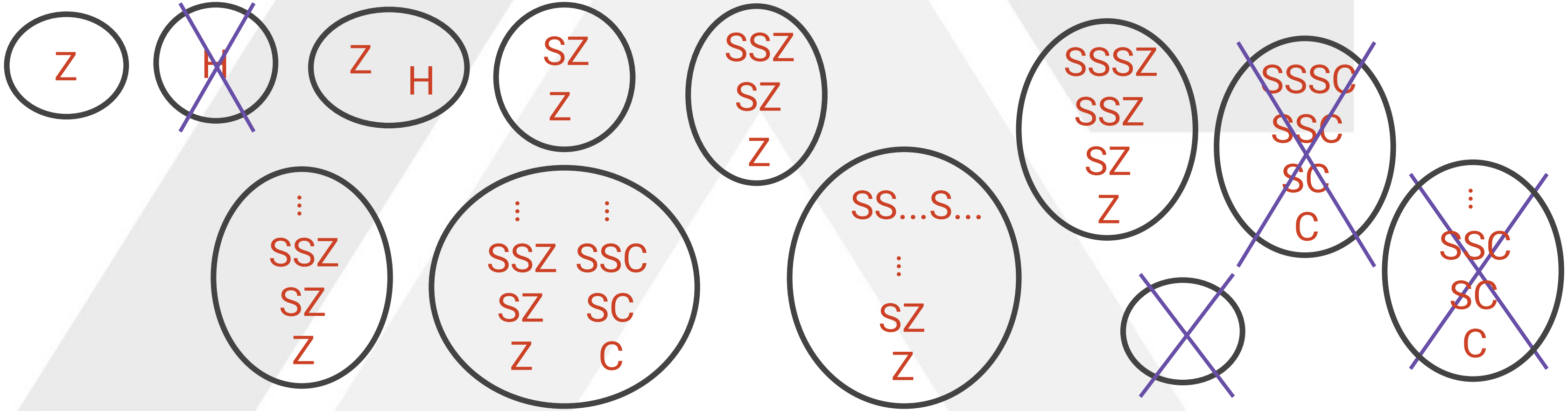
Inducción estructural

- ¿Cómo saber qué conjuntos satisfacen las condiciones?
  - **Condición 1:** Z tiene que estar en $\aleph$
  - **Condición 2:** si **c** está en $\aleph$, entonces S**c** tiene que estar en $\aleph$
- Analicemos algunos conjuntos posibles y vayamos descartando



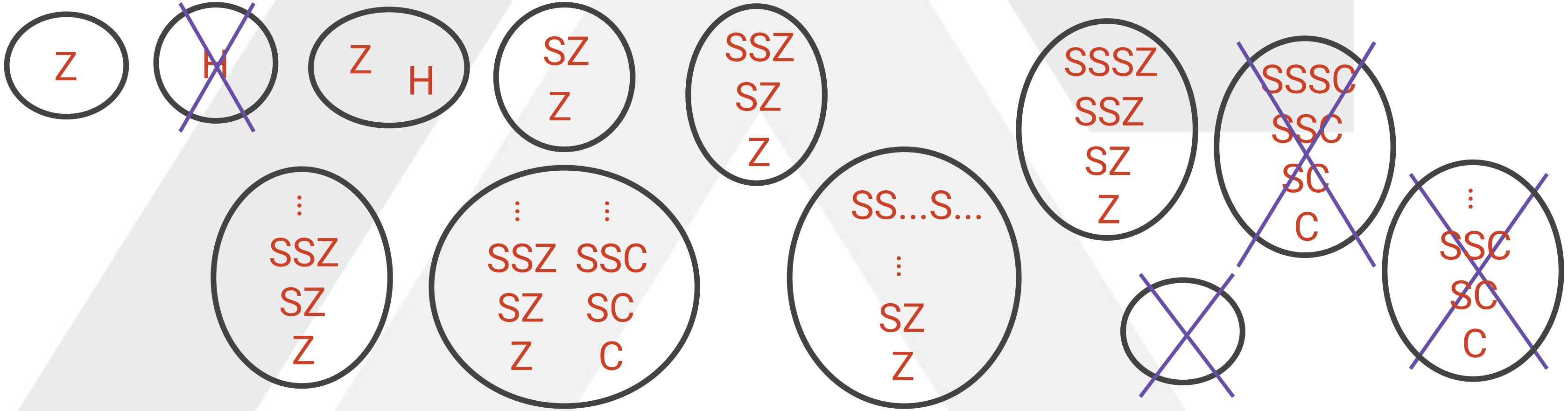
Inducción estructural

- ¿Cómo saber qué conjuntos satisfacen las condiciones?
  - **Condición 1:** Z tiene que estar en $\aleph$
  - **Condición 2:** si **c** está en $\aleph$, entonces S**c** tiene que estar en $\aleph$
- Analicemos algunos conjuntos posibles y vayamos descartando



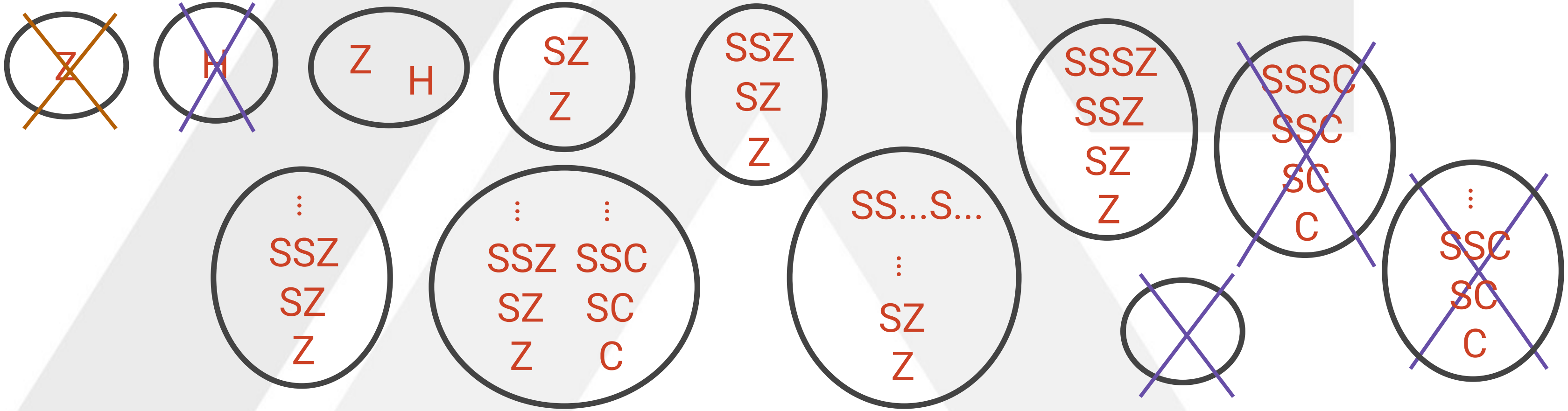
Inducción estructural

- ¿Cómo saber qué conjuntos satisfacen las condiciones?
  - **Condición 1:** Z tiene que estar en $\aleph$
  - **Condición 2:** si **c** está en $\aleph$, entonces S**c** tiene que estar en $\aleph$
- Analicemos algunos conjuntos posibles y vayamos descartando



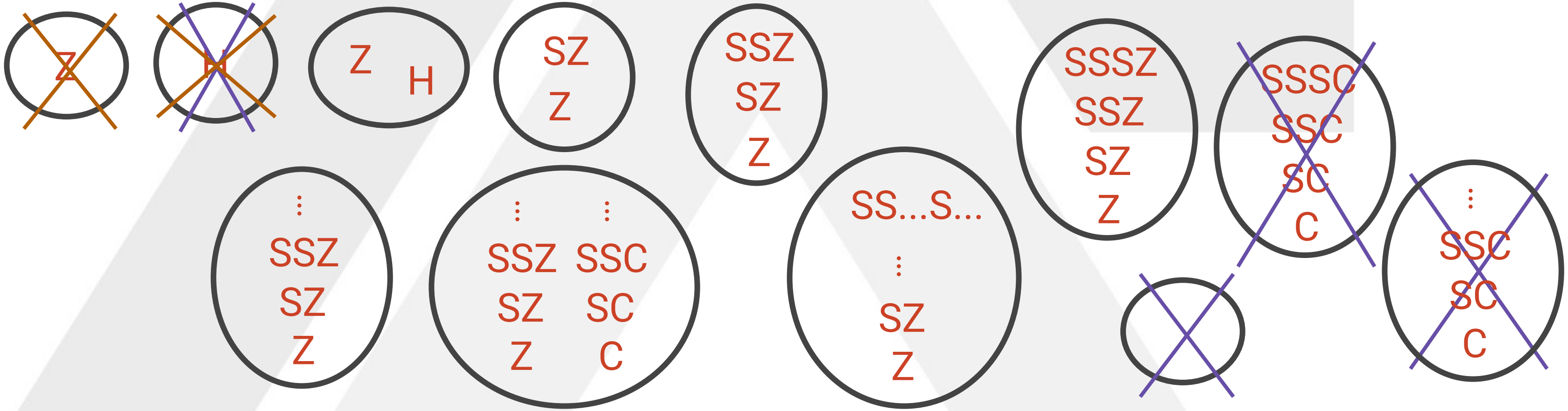
Inducción estructural

- ¿Cómo saber qué conjuntos satisfacen las condiciones?
  - **Condición 1:** Z tiene que estar en ℵ
  - **Condición 2:** si **c** está en ℵ, entonces S**c** tiene que estar en ℵ
- Analicemos algunos conjuntos posibles y vayamos descartando



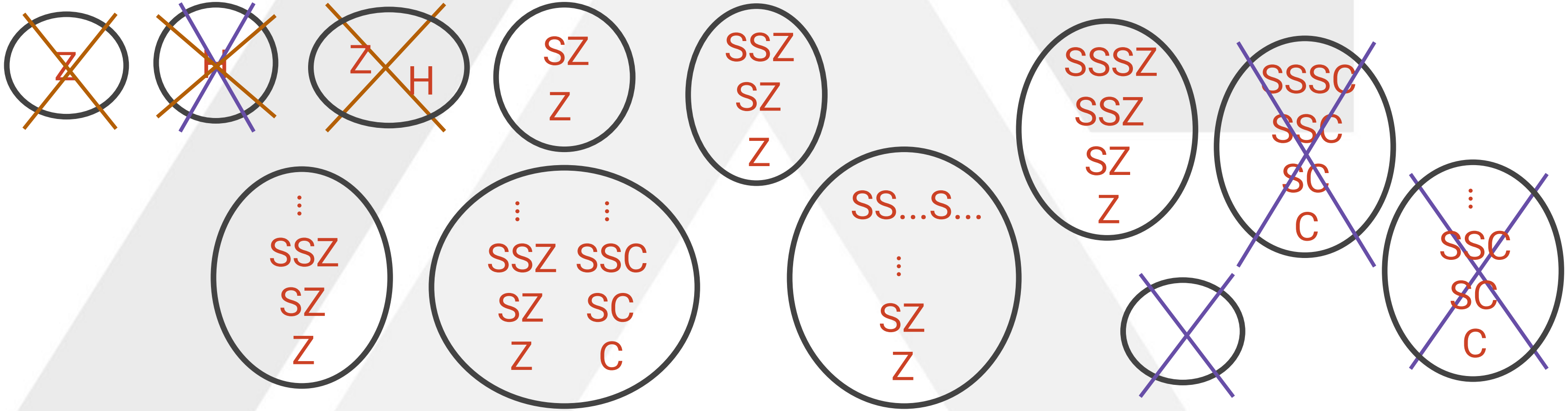
Inducción estructural

- ¿Cómo saber qué conjuntos satisfacen las condiciones?
  - **Condición 1:** Z tiene que estar en ℵ
  - **Condición 2:** si **c** está en ℵ, entonces S**c** tiene que estar en ℵ
- Analicemos algunos conjuntos posibles y vayamos descartando



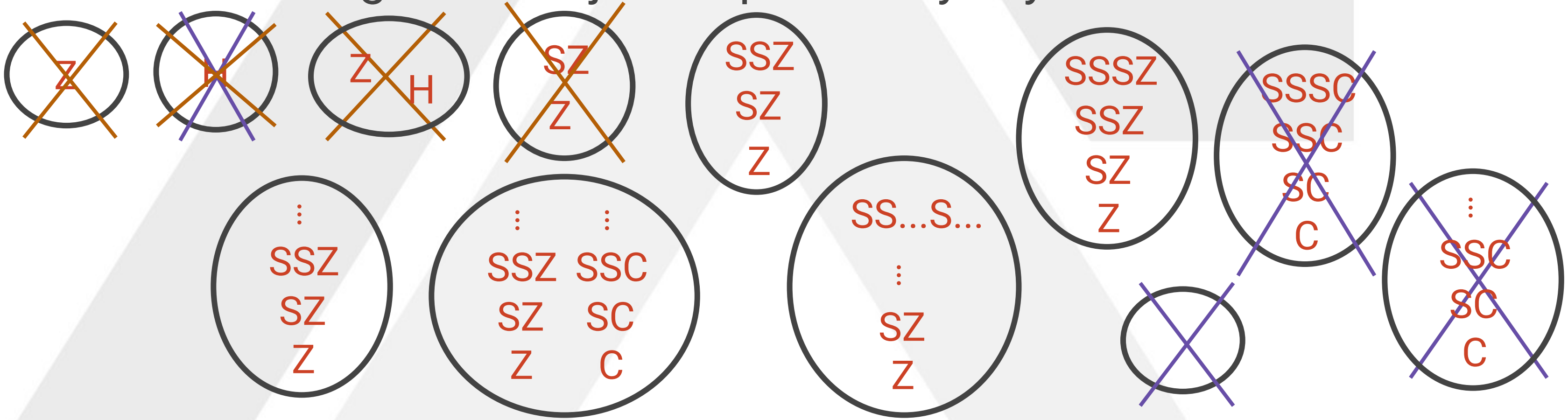
Inducción estructural

- ¿Cómo saber qué conjuntos satisfacen las condiciones?
  - **Condición 1:** Z tiene que estar en $\aleph$
  - **Condición 2:** si **c** está en $\aleph$, entonces S**c** tiene que estar en $\aleph$
- Analicemos algunos conjuntos posibles y vayamos descartando



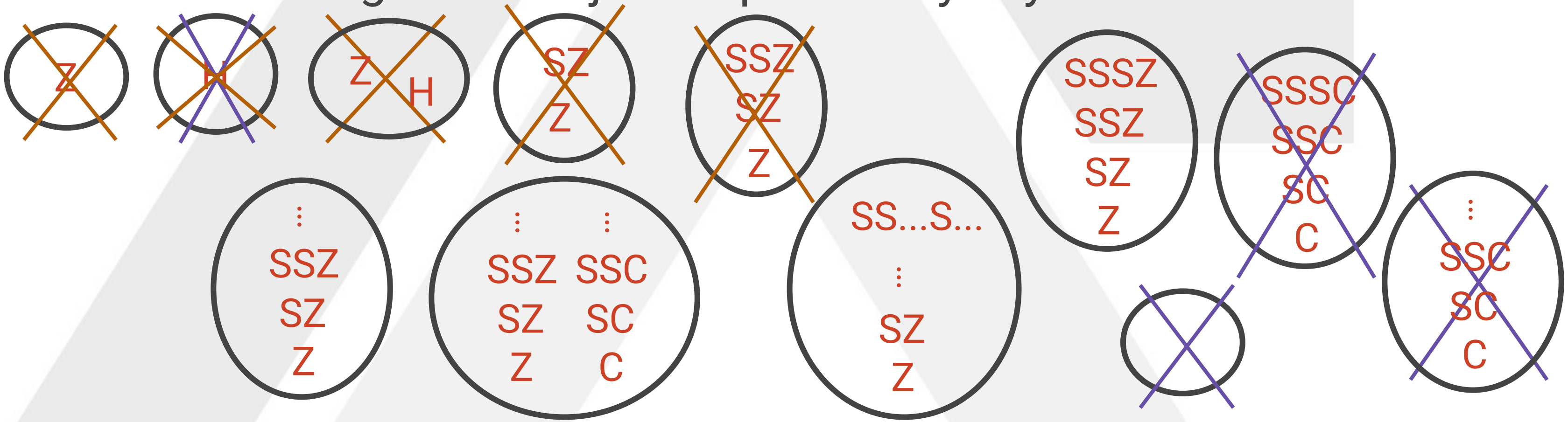
Inducción estructural

- ¿Cómo saber qué conjuntos satisfacen las condiciones?
  - **Condición 1:** Z tiene que estar en $\aleph$
  - **Condición 2:** si **c** está en $\aleph$, entonces S**c** tiene que estar en $\aleph$
- Analicemos algunos conjuntos posibles y vayamos descartando



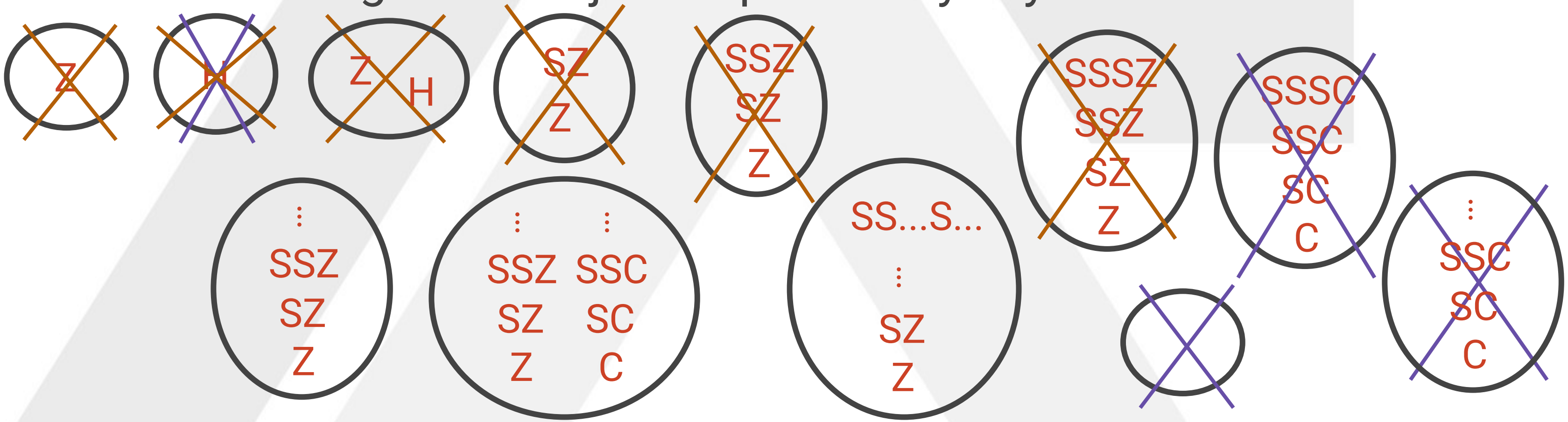
Inducción estructural

- ¿Cómo saber qué conjuntos satisfacen las condiciones?
  - **Condición 1:** Z tiene que estar en ℵ
  - **Condición 2:** si **c** está en ℵ, entonces S**c** tiene que estar en ℵ
- Analicemos algunos conjuntos posibles y vayamos descartando



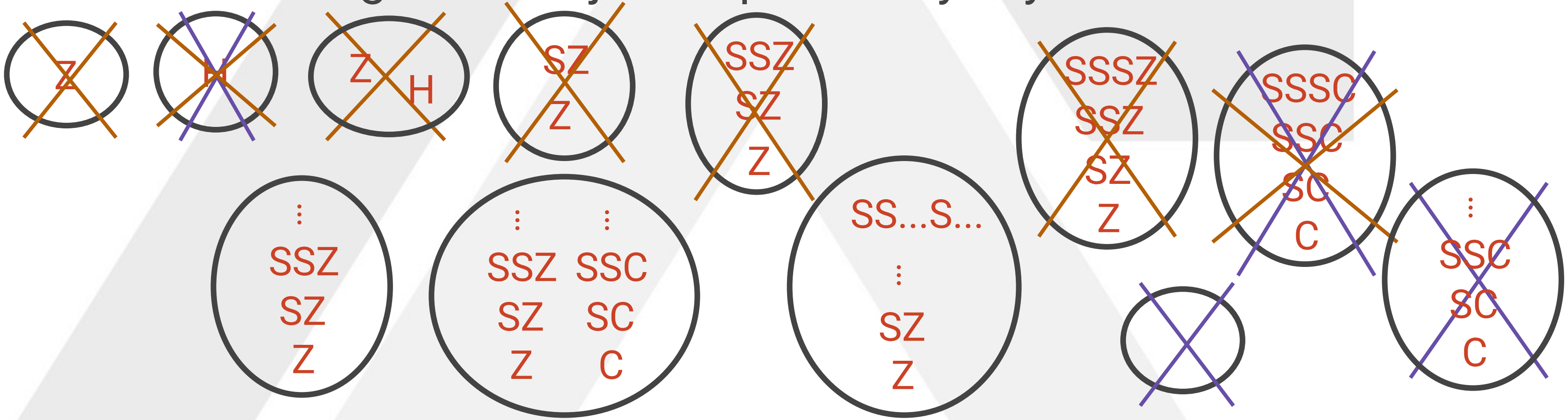
Inducción estructural

- ¿Cómo saber qué conjuntos satisfacen las condiciones?
  - **Condición 1:** Z tiene que estar en $\aleph$
  - **Condición 2:** si **c** está en $\aleph$, entonces S**c** tiene que estar en $\aleph$
- Analicemos algunos conjuntos posibles y vayamos descartando



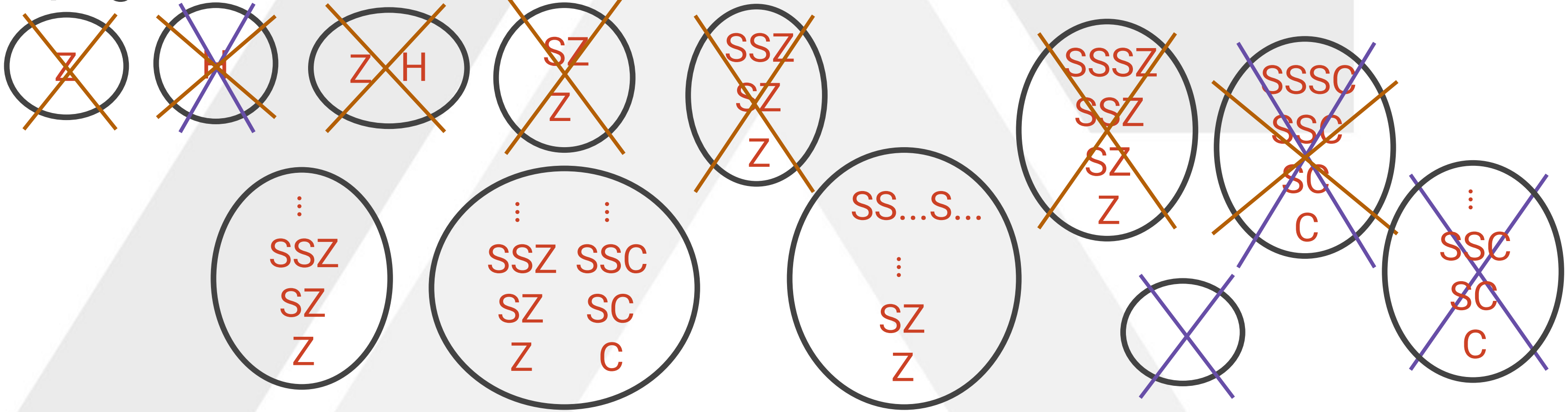
Inducción estructural

- ¿Cómo saber qué conjuntos satisfacen las condiciones?
  - **Condición 1:** Z tiene que estar en $\aleph$
  - **Condición 2:** si **c** está en $\aleph$, entonces S**c** tiene que estar en $\aleph$
- Analicemos algunos conjuntos posibles y vayamos descartando



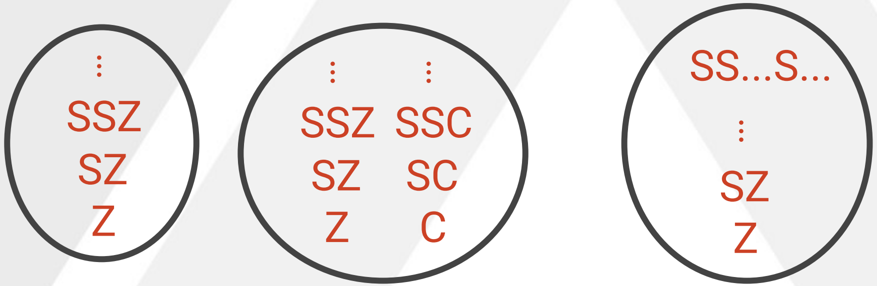
Inducción estructural

- ¿Cómo saber qué conjuntos satisfacen las condiciones?
  - **Condición 1:** Z tiene que estar en ℵ
  - **Condición 2:** si **c** está en ℵ, entonces S**c** tiene que estar en ℵ
- ¡Sigue habiendo muchos válidos!



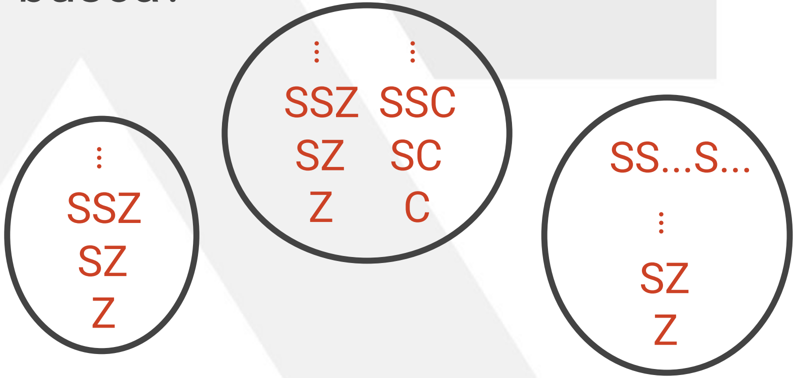
Inducción estructural

- ¿Cómo saber qué conjuntos satisfacen las condiciones?
  - **Condición 1:** Z tiene que estar en $\aleph$
  - **Condición 2:** si **c** está en $\aleph$, entonces S**c** tiene que estar en $\aleph$
- ¡Sigue habiendo muchos válidos!



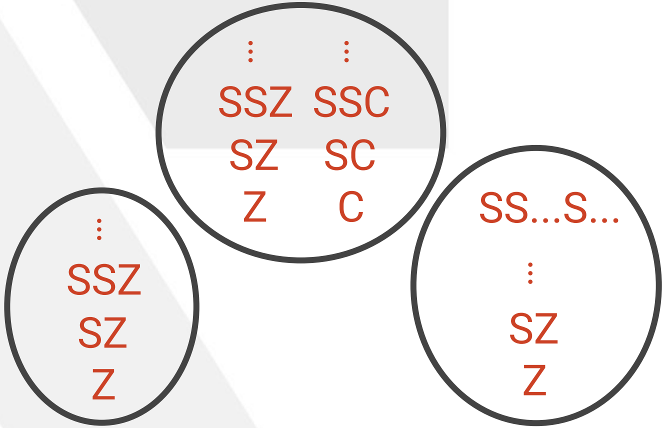
Inducción estructural

- ¿Cómo saber qué conjuntos satisfacen las condiciones?
  - **Condición 1:** Z tiene que estar en $\aleph$
  - **Condición 2:** si **c** está en $\aleph$, entonces S**c** tiene que estar en $\aleph$
- ¡Sigue habiendo muchos válidos!
  - ¿Qué diferencia al que se busca?



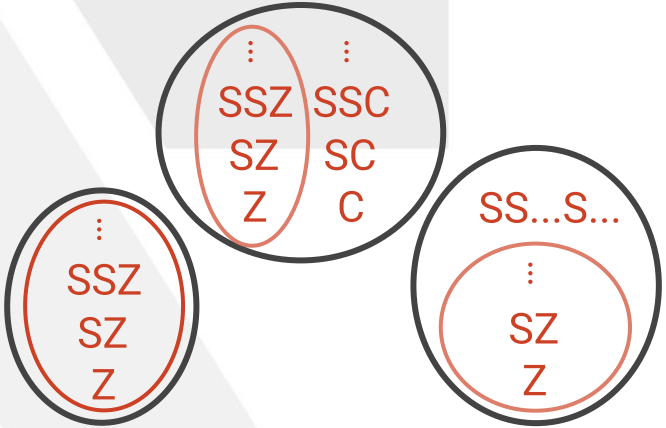
Inducción estructural

- ¿Cómo saber qué conjuntos satisfacen las condiciones?
  - **Condición 1:** Z tiene que estar en $\aleph$
  - **Condición 2:** si **c** está en $\aleph$, entonces S**c** tiene que estar en $\aleph$
- ¡Sigue habiendo muchos válidos!
  - ¿Qué diferencia al que se busca?



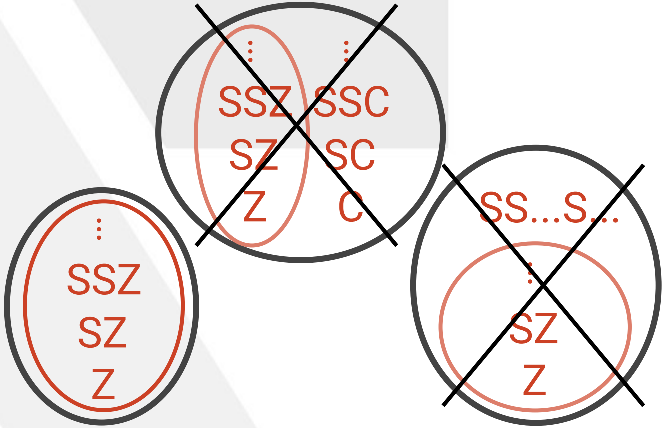
Inducción estructural

- ¿Cómo saber qué conjuntos satisfacen las condiciones?
  - **Condición 1:** Z tiene que estar en $\aleph$
  - **Condición 2:** si **c** está en $\aleph$, entonces S**c** tiene que estar en $\aleph$
- ¡Sigue habiendo muchos válidos!
  - ¿Qué diferencia al que se busca?
    - ¡Está contenido en todos los que cumplen las propiedades!



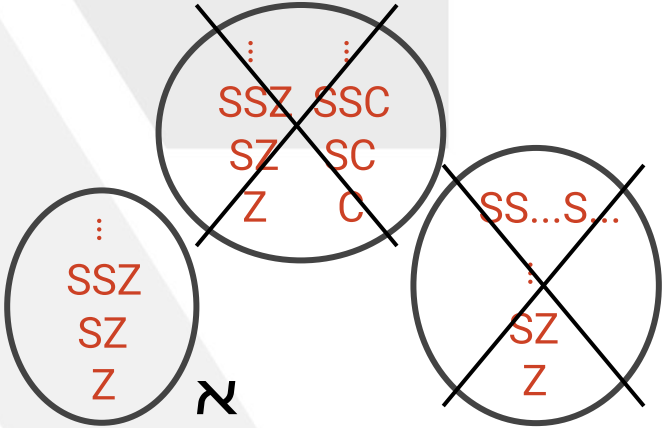
Inducción estructural

- ¿Cómo saber qué conjuntos satisfacen las condiciones?
  - **Condición 1:** Z tiene que estar en $\aleph$
  - **Condición 2:** si **c** está en $\aleph$, entonces S**c** tiene que estar en $\aleph$
- ¡Sigue habiendo muchos válidos!
  - ¿Qué diferencia al que se busca?
    - ¡Está contenido en todos los que cumplen las propiedades!
    - **Condición adicional:** es el *menor* de los que cumplen las anteriores



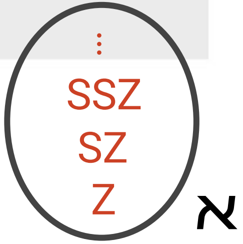
Inducción estructural

- Definición del conjunto ℵ buscado
  - **Condición 1:** Z tiene que estar en ℵ
  - **Condición 2:** si **c** está en ℵ, entonces S**c** tiene que estar en ℵ
  - **Condición adicional:** es el *menor* de los que cumplen



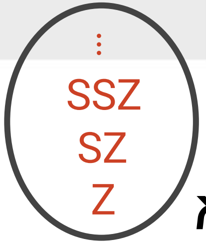
Inducción estructural

- Definición del conjunto $\aleph$ buscado
  - **Condición 1:** Z tiene que estar en $\aleph$
  - **Condición 2:** si **c** está en $\aleph$, entonces S**c** tiene que estar en $\aleph$
  - **Condición adicional:** es el *menor* de los que cumplen



Inducción estructural

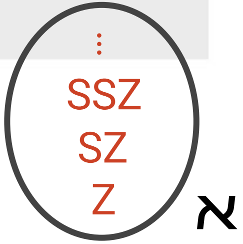
- Definición del conjunto $\aleph$ buscado
  - **Condición 1:** Z tiene que estar en $\aleph$
  - **Condición 2:** si **c** está en $\aleph$, entonces S**c** tiene que estar en $\aleph$
  - **Condición adicional:** es el *menor* de los que cumplen
- ¿Qué características tiene esta definición?



$\aleph$

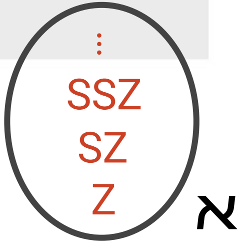
Inducción estructural

- Definición del conjunto $\aleph$ buscado
  - **Condición 1:** Z tiene que estar en $\aleph$
  - **Condición 2:** si **c** está en $\aleph$, entonces S**c** tiene que estar en $\aleph$
  - **Condición adicional:** es el *menor* de los que cumplen
- ¿Qué características tiene esta definición?
  - Una condición es una *afirmación*



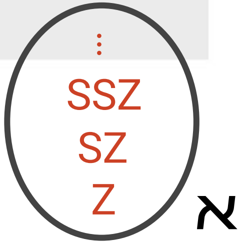
Inducción estructural

- Definición del conjunto $\aleph$ buscado
  - **Condición 1:** Z tiene que estar en $\aleph$
  - **Condición 2:** si **c** está en $\aleph$, entonces S**c** tiene que estar en $\aleph$
  - **Condición adicional:** es el *menor* de los que cumplen
- ¿Qué características tiene esta definición?
  - Una condición es una *afirmación*
  - Una condición es una *implicación*



Inducción estructural

- Definición del conjunto $\aleph$ buscado
  - **Condición 1:** Z tiene que estar en $\aleph$
  - **Condición 2:** si **c** está en $\aleph$, entonces S**c** tiene que estar en $\aleph$
  - **Condición adicional:** es el *menor* de los que cumplen
- ¿Qué características tiene esta definición?
  - Una condición es una *afirmación*
  - Una condición es una *implicación*
  - La condición extra provee *unicidad*



Inducción estructural

- ¿Cualquier conjunto de condiciones sirve?
- Intento de definición “inductiva” del conjunto $\aleph$
  - **Condición 1:** Z tiene que estar en $\aleph$
  - **Condición 2:** si **c** está en $\aleph$, entonces S**c** tiene que estar en $\aleph$
  - **Condición 3:** si **c** está en $\aleph$, entonces SS**c** tiene que estar en $\aleph$

Inducción estructural

- ¿Cualquier conjunto de condiciones sirve?
- Intento de definición "inductiva" del conjunto $\aleph$
  - **Condición 1:** Z tiene que estar en $\aleph$
  - **Condición 2:** si **c** está en $\aleph$, entonces S**c** tiene que estar en $\aleph$
  - **Condición 3:** si **c** está en $\aleph$, entonces SS**c** tiene que estar en $\aleph$
- Observar que la condición 3 no aporta información nueva
  - No es *productiva*
  - Una definición por inducción necesita ***reglas productivas***

Inducción estructural

- La definición de un conjunto por ***inducción estructural*** se compone de condiciones (llamadas **reglas**)

Inducción estructural

- La definición de un conjunto por ***inducción estructural*** se compone de condiciones (llamadas ***reglas***)
  - Una o varias ***reglas base:*** afirmaciones directas

Inducción estructural

- La definición de un conjunto por ***inducción estructural*** se compone de condiciones (llamadas **reglas**)
  - Una o varias ***reglas base*:** afirmaciones directas
  - Una o varias ***reglas inductivas*:** implicaciones (productivas)

Inducción estructural

- La definición de un conjunto por ***inducción estructural*** se compone de condiciones (llamadas **reglas**)
  - Una o varias ***reglas base*:** afirmaciones directas
  - Una o varias ***reglas inductivas*:** implicaciones (productivas)
  - Una única ***condición adicional*:** pedir que sea el *menor*

Inducción estructural

- La definición de un conjunto por ***inducción estructural*** se compone de condiciones (llamadas **reglas**)
  - Una o varias ***reglas base:*** afirmaciones directas
    - e.g. Z tiene que estar en ℵ
  - Una o varias ***reglas inductivas:*** implicaciones (productivas)
  - Una única ***condición adicional:*** pedir que sea el *menor*

Inducción estructural

- La definición de un conjunto por ***inducción estructural*** se compone de condiciones (llamadas **reglas**)
  - Una o varias ***reglas base***: afirmaciones directas
    - e.g. Z tiene que estar en $\aleph$
  - Una o varias ***reglas inductivas***: implicaciones (productivas)
    - e.g. si **c** está en $\aleph$, entonces S**c** tiene que estar en $\aleph$
  - Una única ***condición adicional***: pedir que sea el *menor*

Inducción estructural

- La definición de un conjunto por ***inducción estructural*** se compone de condiciones (llamadas **reglas**)
  - Una o varias ***reglas base***: afirmaciones directas
    - e.g. Z tiene que estar en ℵ
  - Una o varias ***reglas inductivas***: implicaciones (productivas)
    - e.g. si **c** está en ℵ, entonces S**c** tiene que estar en ℵ
  - Una única ***condición adicional***: pedir que sea el *menor*
    - el menor que cumple todas las condiciones anteriores
    - provee *unicidad* al conjunto buscado

Inducción estructural

Definición de un conjunto por ***inducción estructural***
❑ Una o varias ***reglas base*** (afirmaciones directas) ❑ Una o varias ***reglas inductivas*** (implicaciones productivas)
❑ Una única ***condición adicional*** (pedir que sea el *menor*)

Inducción estructural

Definición de un conjunto **S** por ***inducción estructural***

- ***Reglas base:*** $\mathbf{z_1}$ está en **S**

  …

  $\mathbf{z_n}$ está en **S**
- ***Reglas inductivas:*** si $\mathbf{e_{11}}$, … , $\mathbf{e_{i1}}$ están en **S**, entonces $\mathbf{e_1}$ está en **S**

  …

  si $\mathbf{e_{1k}}$, … , $\mathbf{e_{ik}}$ están en **S**, entonces $\mathbf{e_k}$ está en **S**
- El *MENOR* conjunto que cumple todas las reglas (en el sentido de la inclusión)

Inducción estructural

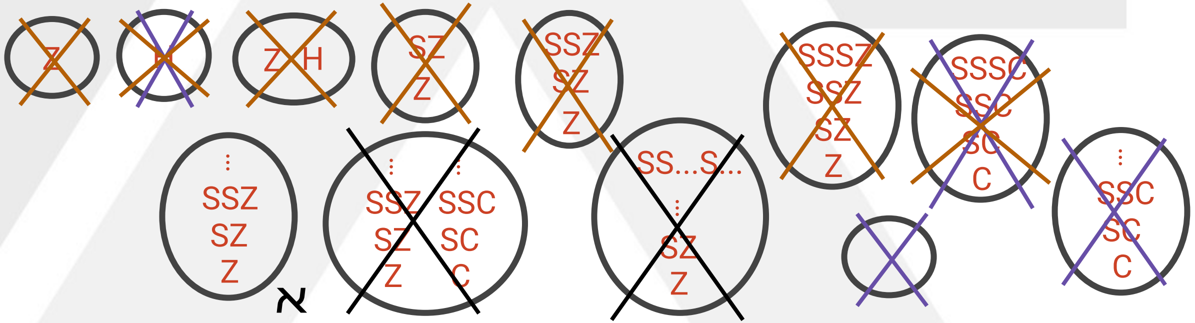
- Conjunto definido por ***inducción estructural***
  - También llamado ***conjunto inductivo***
    - Se define mediante *reglas básicas* y *reglas inductivas*
    - Siempre está la condición de ser el *menor* conjunto que cumple todas las reglas (en el sentido de la inclusión)
  - Las reglas pueden usarse de diversas formas
    - Como condiciones
    - Como formas de agregar elementos
    - Para verificar si un elemento es miembro del conjunto

Inducción estructural

- Definición *inductiva* de $\aleph$
  - **Regla base:** Z está en $\aleph$
  - **Regla inductiva:** si **c** está en $\aleph$, entonces S**c** está en $\aleph$
    - (La regla de ser el menor es implícita en el adjetivo “inductivo”)

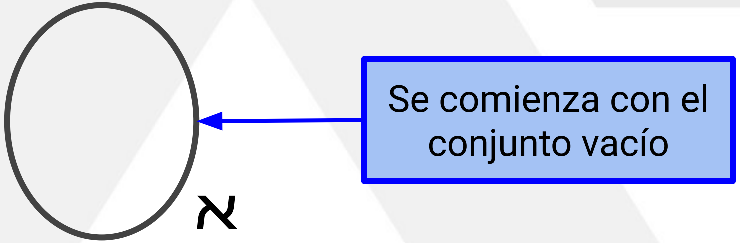
Inducción estructural

- Definición *inductiva* de ℵ
  - **Regla base:** Z está en ℵ
  - **Regla inductiva:** si **c** está en ℵ, entonces S**c** está en ℵ
    - (La regla de ser el menor es implícita en el adjetivo "inductivo")
- Reglas como condiciones (ya visto)



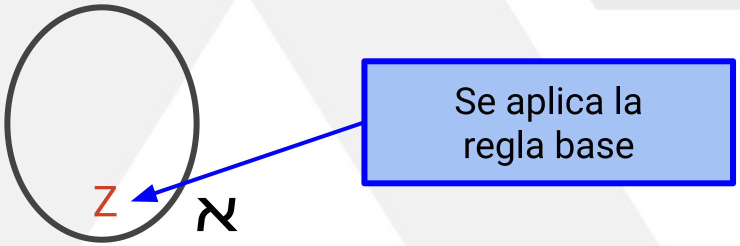
Inducción estructural

- Definición *inductiva* de $\aleph$
  - **Regla base:** Z está en $\aleph$
  - **Regla inductiva:** si **c** está en $\aleph$, entonces S**c** está en $\aleph$
    - (La regla de ser el menor es implícita en el adjetivo “inductivo”)
- Reglas como forma de agregar elementos



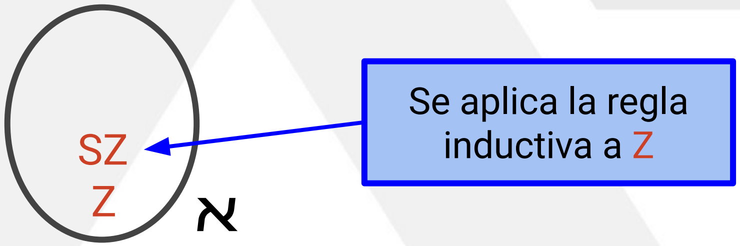
Inducción estructural

- Definición *inductiva* de ℵ
  - **Regla base:** Z está en ℵ
  - **Regla inductiva:** si **c** está en ℵ, entonces S**c** está en ℵ
    - (La regla de ser el menor es implícita en el adjetivo “inductivo”)
- Reglas como forma de agregar elementos



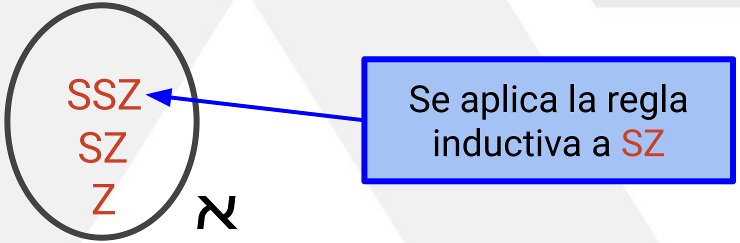
Inducción estructural

- Definición *inductiva* de $\aleph$
  - **Regla base:** Z está en $\aleph$
  - **Regla inductiva:** si **c** está en $\aleph$, entonces S**c** está en $\aleph$
    - (La regla de ser el menor es implícita en el adjetivo “inductivo”)
- Reglas como forma de agregar elementos



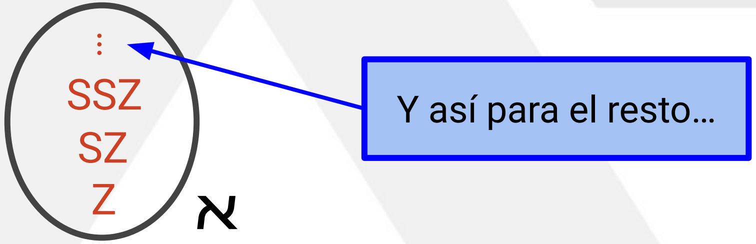
Inducción estructural

- Definición *inductiva* de $\aleph$
  - **Regla base:** Z está en $\aleph$
  - **Regla inductiva:** si **c** está en $\aleph$, entonces S**c** está en $\aleph$
    - (La regla de ser el menor es implícita en el adjetivo “inductivo”)
- Reglas como forma de agregar elementos



Inducción estructural

- Definición *inductiva* de $\aleph$
  - **Regla base:** Z está en $\aleph$
  - **Regla inductiva:** si **c** está en $\aleph$, entonces S**c** está en $\aleph$
    - (La regla de ser el menor es implícita en el adjetivo “inductivo”)
- Reglas como forma de agregar elementos



Inducción estructural

- Definición *inductiva* de ℵ
  - **Regla base:** Z está en ℵ
  - **Regla inductiva:** si **c** está en ℵ, entonces S**c** está en ℵ
    - (La regla de ser el menor es implícita en el adjetivo “inductivo”)
- Reglas para ver si un elemento es parte del conjunto

¿ SSSSZ está en ℵ ?

Inducción estructural

- Definición *inductiva* de $\aleph$
  - **Regla base:** Z está en $\aleph$
  - **Regla inductiva:** si **c** está en $\aleph$, entonces S**c** está en $\aleph$
    - (La regla de ser el menor es implícita en el adjetivo "inductivo")
- Reglas para ver si un elemento es parte del conjunto

¿ S$\boxed{\text{SSSZ}}^{\,\mathbf{c}}$ está en $\aleph$?

¿ SSSZ está en $\aleph$?

Por la regla inductiva, debería también valer que…

Inducción estructural

- Definición *inductiva* de $\aleph$
  - **Regla base:** Z está en $\aleph$
  - **Regla inductiva:** si **c** está en $\aleph$, entonces S**c** está en $\aleph$
    - (La regla de ser el menor es implícita en el adjetivo “inductivo”)
- Reglas para ver si un elemento es parte del conjunto

¿ SSSSZ está en $\aleph$?

¿ S$\boxed{\text{SSZ}}$ está en $\aleph$?

c

¿ SSZ está en $\aleph$?

Por la regla inductiva, debería también valer que…

Inducción estructural

- Definición *inductiva* de $\aleph$
  - **Regla base:** Z está en $\aleph$
  - **Regla inductiva:** si **c** está en $\aleph$, entonces S**c** está en $\aleph$
    - (La regla de ser el menor es implícita en el adjetivo “inductivo”)
- Reglas para ver si un elemento es parte del conjunto

¿ SSSSZ está en $\aleph$?

¿ SSSZ está en $\aleph$?

¿ S$\boxed{\text{SZ}}$ está en $\aleph$? **c**

¿ SZ está en $\aleph$?

Por la regla inductiva, debería también valer que…

Inducción estructural

- Definición *inductiva* de ℵ
  - **Regla base:** Z está en ℵ
  - **Regla inductiva:** si **c** está en ℵ, entonces S**c** está en ℵ
    - (La regla de ser el menor es implícita en el adjetivo “inductivo”)
- Reglas para ver si un elemento es parte del conjunto

¿ SSSSZ está en ℵ ?

¿ SSSZ está en ℵ ?

¿ SSZ está en ℵ ?

¿ S$\underset{}{\boxed{\text{Z}}}^{\mathbf{c}}$ está en ℵ ?

¿ Z está en ℵ ?

Por la regla inductiva, debería también valer que…

Inducción estructural

- Definición *inductiva* de ℵ
  - **Regla base:** Z está en ℵ
  - **Regla inductiva:** si **c** está en ℵ, entonces S**c** está en ℵ
    - (La regla de ser el menor es implícita en el adjetivo “inductivo”)
- Reglas para ver si un elemento es parte del conjunto

¿ SSSSZ está en ℵ ?

¿ SSSZ está en ℵ ?

¿ SSZ está en ℵ ?

¿ SZ está en ℵ ?

¡ Z está en ℵ !

Por la regla base

Inducción estructural

- Definición *inductiva* de ℵ
  - **Regla base:** Z está en ℵ
  - **Regla inductiva:** si **c** está en ℵ, entonces S**c** está en ℵ
    - (La regla de ser el menor es implícita en el adjetivo "inductivo")
- Reglas para ver si un elemento es parte del conjunto

¿ SSSSZ está en ℵ ?

¿ SSSZ está en ℵ ?

¿ SSZ está en ℵ ?

¡ SZ está en ℵ !

¡ Z está en ℵ !

Por la regla inductiva

Inducción estructural

- Definición *inductiva* de $\aleph$
  - **Regla base:** Z está en $\aleph$
  - **Regla inductiva:** si **c** está en $\aleph$, entonces S**c** está en $\aleph$
    - (La regla de ser el menor es implícita en el adjetivo "inductivo")
- Reglas para ver si un elemento es parte del conjunto

¿ SSSSZ está en $\aleph$?

¿ SSSZ está en $\aleph$?

¡ SSZ está en $\aleph$!

¡ SZ está en $\aleph$!

¡ Z está en $\aleph$!

Por la regla inductiva

Inducción estructural

- Definición *inductiva* de ℵ
  - **Regla base:** Z está en ℵ
  - **Regla inductiva:** si **c** está en ℵ, entonces S**c** está en ℵ
    - (La regla de ser el menor es implícita en el adjetivo “inductivo”)
- Reglas para ver si un elemento es parte del conjunto

¿ SSSSZ está en ℵ ?

¡ SSSZ está en ℵ !

¡ SSZ está en ℵ !

¡ SZ está en ℵ !

¡ Z está en ℵ !

Por la regla inductiva

Inducción estructural

- Definición *inductiva* de ℵ
  - **Regla base:** Z está en ℵ
  - **Regla inductiva:** si **c** está en ℵ, entonces S**c** está en ℵ
    - (La regla de ser el menor es implícita en el adjetivo “inductivo”)
- Reglas para ver si un elemento es parte del conjunto

¡ SSSSZ está en ℵ !

¡ SSSZ está en ℵ !

¡ SSZ está en ℵ !

¡ SZ está en ℵ !

¡ Z está en ℵ !

Por la regla inductiva

Inducción estructural

- Definición *inductiva* de $\aleph$
  - **Regla base:** Z está en $\aleph$
  - **Regla inductiva:** si **c** está en $\aleph$, entonces S**c** está en $\aleph$
    - (La regla de ser el menor es implícita en el adjetivo "inductivo")
- Reglas para ver si un elemento es parte del conjunto

¡ SSSSZ está en $\aleph$!

Inducción estructural

- Definición *inductiva* de $\aleph$
  - **Regla base:** Z está en $\aleph$
  - **Regla inductiva:** si **c** está en $\aleph$, entonces S**c** está en $\aleph$
    - (La regla de ser el menor es implícita en el adjetivo "inductivo")
- Reglas para ver si un elemento es parte del conjunto

¡ SSSSZ está en $\aleph$!

¿ SSSSH está en $\aleph$?

Inducción estructural

- Definición *inductiva* de $\aleph$
  - **Regla base:** Z está en $\aleph$
  - **Regla inductiva:** si **c** está en $\aleph$, entonces S**c** está en $\aleph$
    - (La regla de ser el menor es implícita en el adjetivo "inductivo")
- Reglas para ver si un elemento es parte del conjunto

¡ SSSSZ está en $\aleph$!

¿ S$\overset{\text{c}}{\boxed{\text{SSSH}}}$ está en $\aleph$?

¿ SSSH está en $\aleph$?

Por la regla inductiva, debería también valer que…

Inducción estructural

- Definición *inductiva* de $\aleph$
  - **Regla base:** Z está en $\aleph$
  - **Regla inductiva:** si **c** está en $\aleph$, entonces S**c** está en $\aleph$
    - (La regla de ser el menor es implícita en el adjetivo “inductivo”)
- Reglas para ver si un elemento es parte del conjunto

¡ SSSSZ está en $\aleph$!

¿ SSSSH está en $\aleph$?

¿ S$\overset{c}{\boxed{\text{SSH}}}$ está en $\aleph$?

¿ SSH está en $\aleph$?

Por la regla inductiva, debería también valer que…

Inducción estructural

- Definición *inductiva* de $\aleph$
  - **Regla base:** Z está en $\aleph$
  - **Regla inductiva:** si **c** está en $\aleph$, entonces S**c** está en $\aleph$
    - (La regla de ser el menor es implícita en el adjetivo “inductivo”)
- Reglas para ver si un elemento es parte del conjunto

¡ SSSSZ está en $\aleph$!

¿ SSSSH está en $\aleph$?

¿ SSSH está en $\aleph$?

¿ S$\overset{\textbf{c}}{\boxed{\text{SH}}}$ está en $\aleph$?

¿ SH está en $\aleph$?

Por la regla inductiva, debería también valer que…

Inducción estructural

- Definición *inductiva* de $\aleph$
  - **Regla base:** Z está en $\aleph$
  - **Regla inductiva:** si **c** está en $\aleph$, entonces S**c** está en $\aleph$
    - (La regla de ser el menor es implícita en el adjetivo “inductivo”)
- Reglas para ver si un elemento es parte del conjunto

¡ SSSSZ está en $\aleph$!

¿ SSSSH está en $\aleph$?

¿ SSSH está en $\aleph$?

¿ SSH está en $\aleph$?

c

¿ S$\boxed{\text{H}}$ está en $\aleph$?

¿ H está en $\aleph$?

Por la regla inductiva, debería también valer que…

Inducción estructural

- Definición *inductiva* de $\aleph$
  - **Regla base:** Z está en $\aleph$
  - **Regla inductiva:** si **c** está en $\aleph$, entonces S**c** está en $\aleph$
    - (La regla de ser el menor es implícita en el adjetivo “inductivo”)
- Reglas para ver si un elemento es parte del conjunto

¡ SSSSZ está en $\aleph$!

¿ SSSSH está en $\aleph$?

¿ SSSH está en $\aleph$?

¿ SSH está en $\aleph$?

¿ SH está en $\aleph$?

¡ H NO está en $\aleph$!

Ninguna regla aplica

Inducción estructural

- Definición *inductiva* de $\aleph$
  - **Regla base:** Z está en $\aleph$
  - **Regla inductiva:** si **c** está en $\aleph$, entonces S**c** está en $\aleph$
    - (La regla de ser el menor es implícita en el adjetivo "inductivo")
- Reglas para ver si un elemento es parte del conjunto

¡ SSSSZ está en $\aleph$!

¿ SSSSH está en $\aleph$?

¿ SSSH está en $\aleph$?

¿ SSH está en $\aleph$?

¡ SH NO está en $\aleph$!

¡ H NO está en $\aleph$!

No se cumple el antecedente de la regla inductiva

Inducción estructural

- Definición *inductiva* de $\aleph$
  - **Regla base:** Z está en $\aleph$
  - **Regla inductiva:** si **c** está en $\aleph$, entonces S**c** está en $\aleph$
    - (La regla de ser el menor es implícita en el adjetivo "inductivo")
- Reglas para ver si un elemento es parte del conjunto

¡ SSSSZ está en $\aleph$!

¿ SSSSH está en $\aleph$?

¿ SSSH está en $\aleph$?

¡ SSH NO está en $\aleph$!

¡ SH NO está en $\aleph$!

¡ H NO está en $\aleph$!

No se cumple el antecedente de la regla inductiva

Inducción estructural

- Definición *inductiva* de $\aleph$
  - **Regla base:** Z está en $\aleph$
  - **Regla inductiva:** si **c** está en $\aleph$, entonces S**c** está en $\aleph$
    - (La regla de ser el menor es implícita en el adjetivo “inductivo”)
- Reglas para ver si un elemento es parte del conjunto

¡ SSSSZ está en $\aleph$!

¿ SSSSH está en $\aleph$?

¡ SSSH NO está en $\aleph$!

¡ SSH NO está en $\aleph$!

¡ SH NO está en $\aleph$!

¡ H NO está en $\aleph$!

No se cumple el antecedente de la regla inductiva

Inducción estructural

- Definición *inductiva* de ℵ
  - **Regla base:** Z está en ℵ
  - **Regla inductiva:** si **c** está en ℵ, entonces S**c** está en ℵ
    - (La regla de ser el menor es implícita en el adjetivo “inductivo”)
- Reglas para ver si un elemento es parte del conjunto

¡ SSSSZ está en ℵ !

¡ SSSSH NO está en ℵ !

¡ SSSH NO está en ℵ !

¡ SSH NO está en ℵ !

¡ SH NO está en ℵ !

¡ H NO está en ℵ !

No se cumple el antecedente de la regla inductiva

Inducción estructural

- Definición *inductiva* de $\aleph$
  - **Regla base:** Z está en $\aleph$
  - **Regla inductiva:** si **c** está en $\aleph$, entonces S**c** está en $\aleph$
    - (La regla de ser el menor es implícita en el adjetivo "inductivo")
- Reglas para ver si un elemento es parte del conjunto

¡ SSSSZ está en $\aleph$!   ¡ SSSSH NO está en $\aleph$!

Inducción estructural

- Conjunto definido por ***inducción estructural***
- ¿Qué propiedades tiene?
  - ¿Cuántos elementos tiene?
  - ¿Cuántas reglas satisface un elemento exactamente?
  - ¿Se pueden usar las reglas para ordenar los elementos de alguna forma?
  - ¿Puede haber elementos de tamaño infinito?
- Analicemos cada una de ellas por separado

Inducción estructural

- Conjunto definido por ***inducción estructural***
  - ¿Cuántos elementos tiene?

Inducción estructural

- Conjunto definido por ***inducción estructural***
  - ¿Cuántos elementos tiene?
    - No puede estar vacío (por las reglas base)
    - Por cada elemento diferente, cada regla inductiva exige que haya otro elemento… (pues son *productivas*)

Inducción estructural

- Conjunto definido por ***inducción estructural***
  - ¿Cuántos elementos tiene?
    - No puede estar vacío (por las reglas base)
    - Por cada elemento diferente, cada regla inductiva exige que haya otro elemento… (pues son *productivas*)
  - ¡Debe tener infinitos elementos!

Inducción estructural

- Conjunto definido por ***inducción estructural***
  - ¿Cuántas reglas satisface cada elemento del conjunto?

Inducción estructural

- Conjunto definido por ***inducción estructural***
  - ¿Cuántas reglas satisface cada elemento del conjunto?
    - ¿Puede un elemento no satisfacer ninguna?

Inducción estructural

- Conjunto definido por ***inducción estructural***
  - ¿Cuántas reglas satisface cada elemento del conjunto?
    - ¿Puede un elemento no satisfacer ninguna?
    - ¿Puede satisfacer más de una regla?

Inducción estructural

- Conjunto definido por ***inducción estructural***
  - ¿Cuántas reglas satisface cada elemento del conjunto?
    - ¿Puede un elemento no satisfacer ninguna? No, no sería el *menor*
    - ¿Puede satisfacer más de una regla? No, son reglas *productivas*

Inducción estructural

- Conjunto definido por ***inducción estructural***
  - ¿Cuántas reglas satisface cada elemento del conjunto?
    - ¿Puede un elemento no satisfacer ninguna? No, no sería el *menor*
    - ¿Puede satisfacer más de una regla? No, son reglas *productivas*
  - ¡Cada elemento satisface exactamente UNA regla!
    - O bien una regla base (***elemento base***)
    - O bien una regla inductiva (***elemento inductivo***)
      (y en ese caso hay otros elementos que lo "justifican":
      sus ***partes***; e.g. **c** es parte de S**c**, por la regla inductiva)

Inducción estructural

- Conjunto definido por ***inducción estructural***
  - ¿Se pueden usar las reglas para ordenar elementos?

Inducción estructural

- Conjunto definido por ***inducción estructural***
  - ¿Se pueden usar las reglas para ordenar elementos?
    - Los **elementos base** son los *más chicos*

Inducción estructural

- Conjunto definido por ***inducción estructural***
  - ¿Se pueden usar las reglas para ordenar elementos?
    - Los **elementos base** son los *más chicos*
    - Un **elemento inductivo** es *más grande* que sus **partes**

Inducción estructural

- Conjunto definido por ***inducción estructural***
  - ¿Se pueden usar las reglas para ordenar elementos?
    - Los **elementos base** son los *más chicos*
    - Un **elemento inductivo** es *más grande* que sus partes
  - Orden ***"es parte de"***
    - Z es parte de SZ
    - SZ es parte de SSZ
    - SSZ es parte de SSSZ
    - etc.

Inducción estructural

- Conjunto definido por ***inducción estructural***
  - ¿Puede haber elementos de tamaño infinito?

Inducción estructural

- Conjunto definido por ***inducción estructural***
  - ¿Puede haber elementos de tamaño infinito?
    - O sea, con infinitas partes
    - Suponiendo que sí, ¿qué sucede si se remueven?

Inducción estructural

- Conjunto definido por ***inducción estructural***
  - ¿Puede haber elementos de tamaño infinito?
    - O sea, con infinitas partes
    - Suponiendo que sí, ¿qué sucede si se remueven?
      - ¡Siguen valiendo todas las reglas!

Inducción estructural

- Conjunto definido por ***inducción estructural***
  - ¿Puede haber elementos de tamaño infinito?
    - O sea, con infinitas partes
    - Suponiendo que sí, ¿qué sucede si se remueven?
      - ¡Siguen valiendo todas las reglas!
  - ¡No hay elementos de tamaño infinito!
    - O sea, ningún elemento tiene infinitas partes
    - El orden *"es parte de"* es ***bien fundado***

Inducción estructural

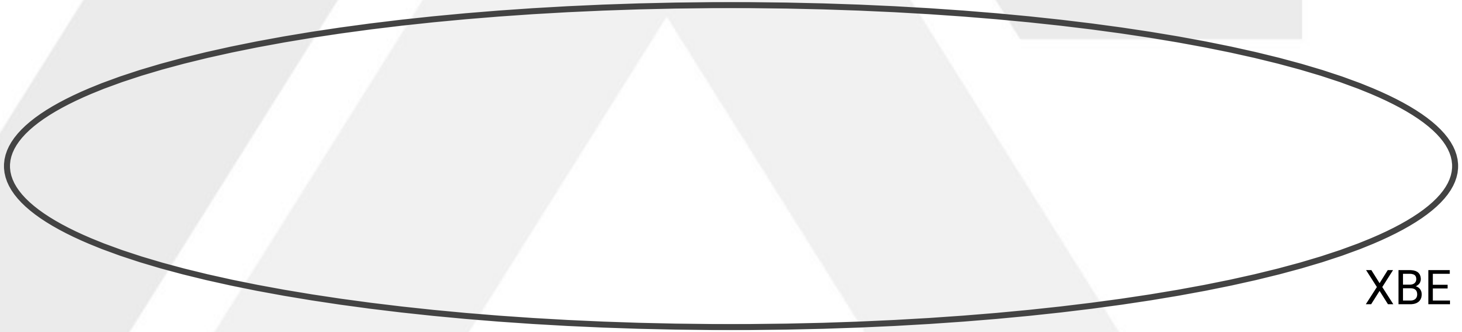
- Conjunto definido por ***inducción estructural***
- ¿Qué propiedades tiene?
  - Tiene infinitos elementos
  - Cada elemento satisface *exactamente* una regla
  - Se puede definir el orden *“es parte de”*
    - Y es **bien fundado** (no hay cadenas descendentes infinitas)
  - No puede haber elementos de tamaño infinito
    - Se puede desarmar un elemento en todas sus partes y subpartes y terminar en algún momento

Inducción estructural

- Definición *inductiva* de XBE
  - **Regla base 1:** △ está en XBE
  - **Regla base 2:** ▽ está en XBE
  - **Regla inductiva 1:** si **e** está en XBE, entonces **#$e//** está en XBE
  - **Regla inductiva 2:** si $\mathbf{e_1}$ está en XBE y $\mathbf{e_2}$ está en XBE, entonces $\mathbf{\#e_1@e_2//}$ está en XBE

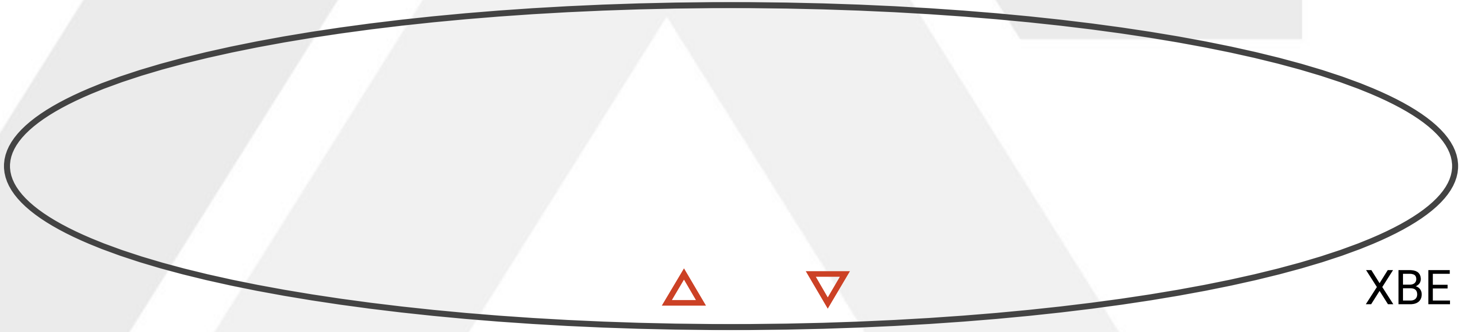
Inducción estructural

- Definición *inductiva* de XBE
  - **Regla base 1:** △ está en XBE
  - **Regla base 2:** ▽ está en XBE
  - **Regla inductiva 1:** si **e** está en XBE, entonces $\#\$e//$ está en XBE
  - **Regla inductiva 2:** si e_1 está en XBE y e_2 está en XBE, entonces $\#e_1@e_2//$ está en XBE



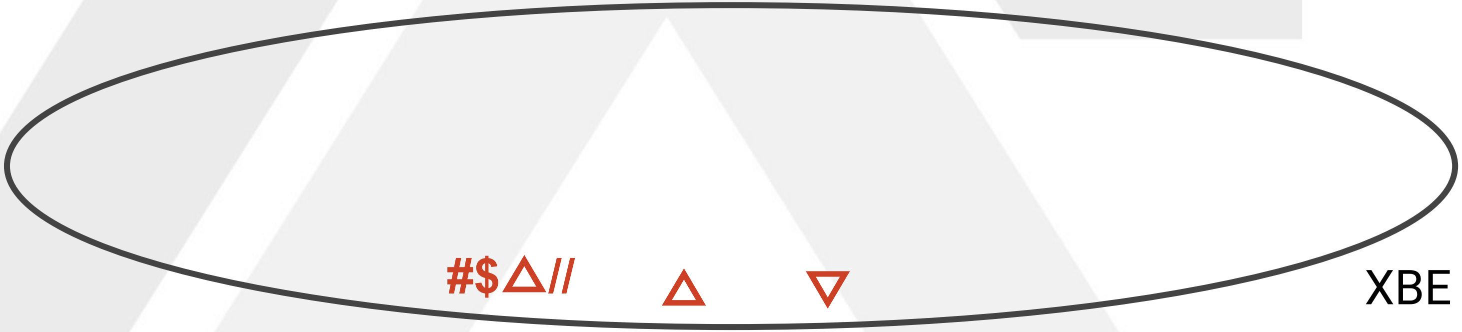
Inducción estructural

- Definición *inductiva* de XBE
  - **Regla base 1:** △ está en XBE
  - **Regla base 2:** ▽ está en XBE
  - **Regla inductiva 1:** si **e** está en XBE, entonces $\#\$e//$ está en XBE
  - **Regla inductiva 2:** si e_1 está en XBE y e_2 está en XBE, entonces $\#e_1@e_2//$ está en XBE



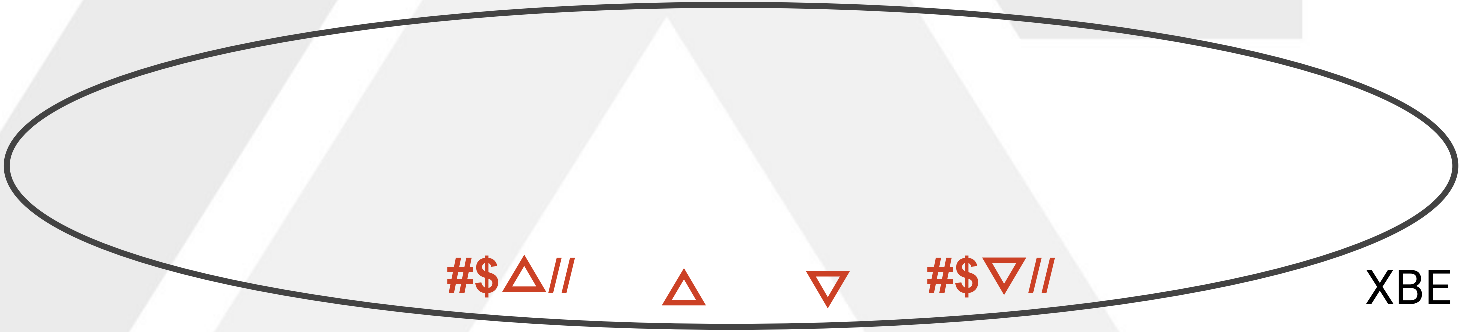
Inducción estructural

- Definición *inductiva* de XBE
  - **Regla base 1:** △ está en XBE
  - **Regla base 2:** ▽ está en XBE
  - **Regla inductiva 1:** si **e** está en XBE, entonces **#$e//** está en XBE
  - **Regla inductiva 2:** si $\mathbf{e_1}$ está en XBE y $\mathbf{e_2}$ está en XBE, entonces **#**$\mathbf{e_1}$**@**$\mathbf{e_2}$**//** está en XBE



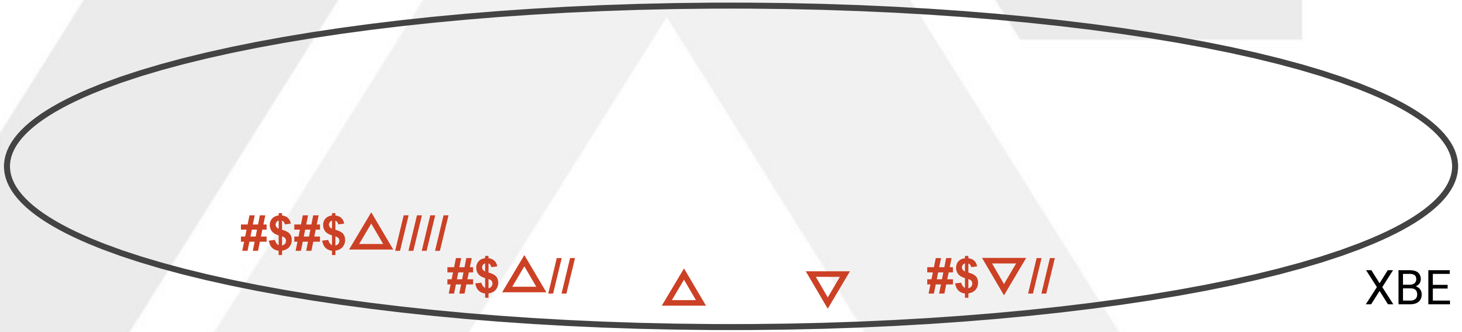
Inducción estructural

- Definición *inductiva* de XBE
  - **Regla base 1:** △ está en XBE
  - **Regla base 2:** ▽ está en XBE
  - **Regla inductiva 1:** si **e** está en XBE, entonces **#$e//** está en XBE
  - **Regla inductiva 2:** si $\mathbf{e_1}$ está en XBE y $\mathbf{e_2}$ está en XBE, entonces $\mathbf{\#e_1@e_2//}$ está en XBE



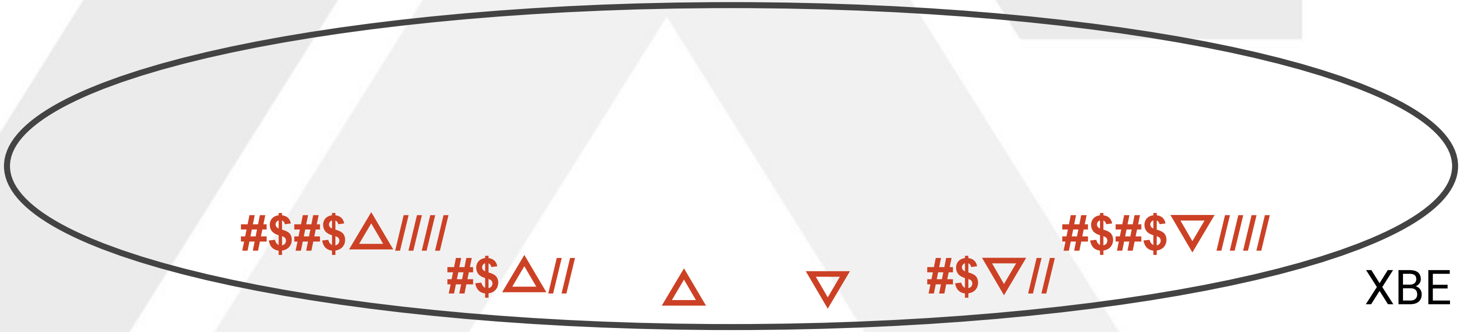
Inducción estructural

- Definición *inductiva* de XBE
  - **Regla base 1:** △ está en XBE
  - **Regla base 2:** ▽ está en XBE
  - **Regla inductiva 1:** si **e** está en XBE, entonces **#$e//** está en XBE
  - **Regla inductiva 2:** si **e_1** está en XBE y **e_2** está en XBE, entonces **#e_1@e_2//** está en XBE



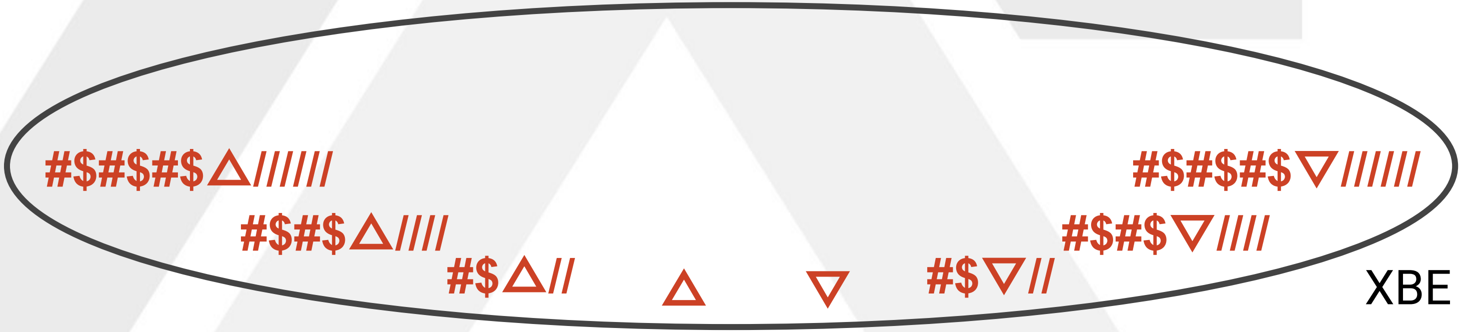
Inducción estructural

- Definición *inductiva* de XBE
  - **Regla base 1:** △ está en XBE
  - **Regla base 2:** ▽ está en XBE
  - **Regla inductiva 1:** si **e** está en XBE, entonces **#$e//** está en XBE
  - **Regla inductiva 2:** si $\mathbf{e_1}$ está en XBE y $\mathbf{e_2}$ está en XBE, entonces $\mathbf{\#e_1@e_2//}$ está en XBE



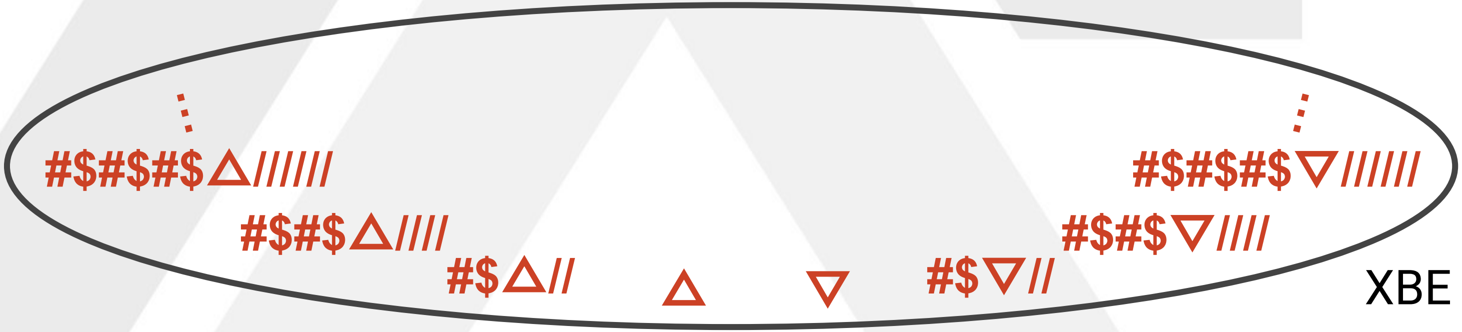
Inducción estructural

- Definición *inductiva* de XBE
  - **Regla base 1:** △ está en XBE
  - **Regla base 2:** ▽ está en XBE
  - **Regla inductiva 1:** si **e** está en XBE, entonces **#$e//** está en XBE
  - **Regla inductiva 2:** si $\mathbf{e}_1$ está en XBE y $\mathbf{e}_2$ está en XBE, entonces **#**$\mathbf{e}_1$**@**$\mathbf{e}_2$**//** está en XBE



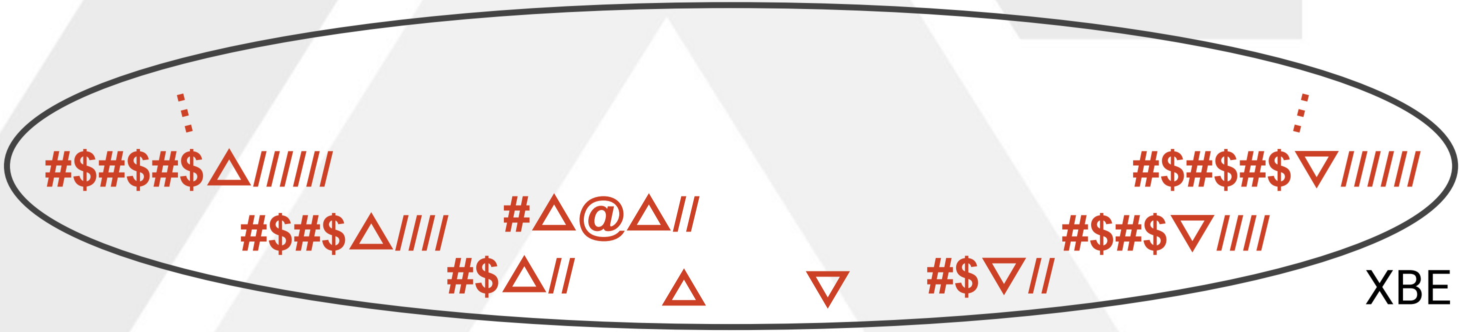
Inducción estructural

- Definición *inductiva* de XBE
  - **Regla base 1:** △ está en XBE
  - **Regla base 2:** ▽ está en XBE
  - **Regla inductiva 1:** si **e** está en XBE, entonces **#$e//** está en XBE
  - **Regla inductiva 2:** si $\mathbf{e}_1$ está en XBE y $\mathbf{e}_2$ está en XBE, entonces **#**$\mathbf{e}_1$**@**$\mathbf{e}_2$**//** está en XBE



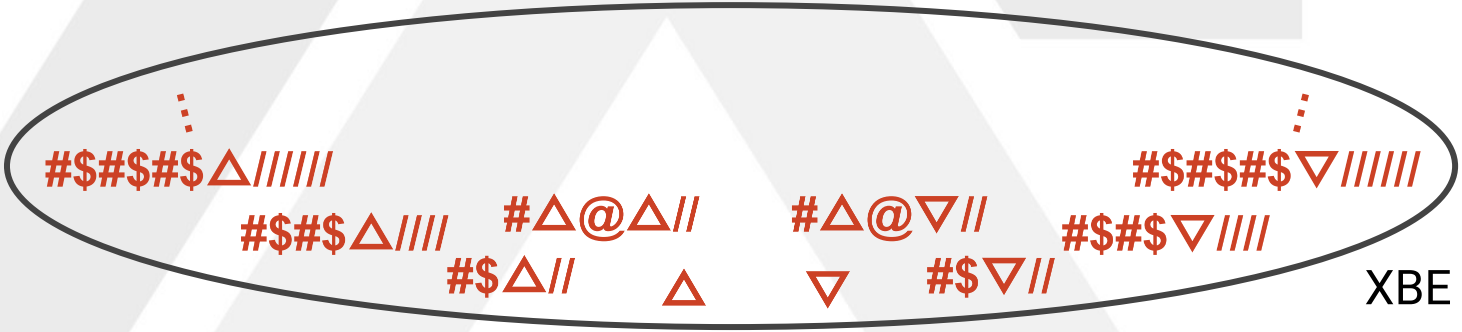
Inducción estructural

- Definición *inductiva* de XBE
  - **Regla base 1:** △ está en XBE
  - **Regla base 2:** ▽ está en XBE
  - **Regla inductiva 1:** si **e** está en XBE, entonces $\#\$e//$ está en XBE
  - **Regla inductiva 2:** si $\mathbf{e}_1$ está en XBE y $\mathbf{e}_2$ está en XBE, entonces $\#\mathbf{e}_1@\mathbf{e}_2//$ está en XBE



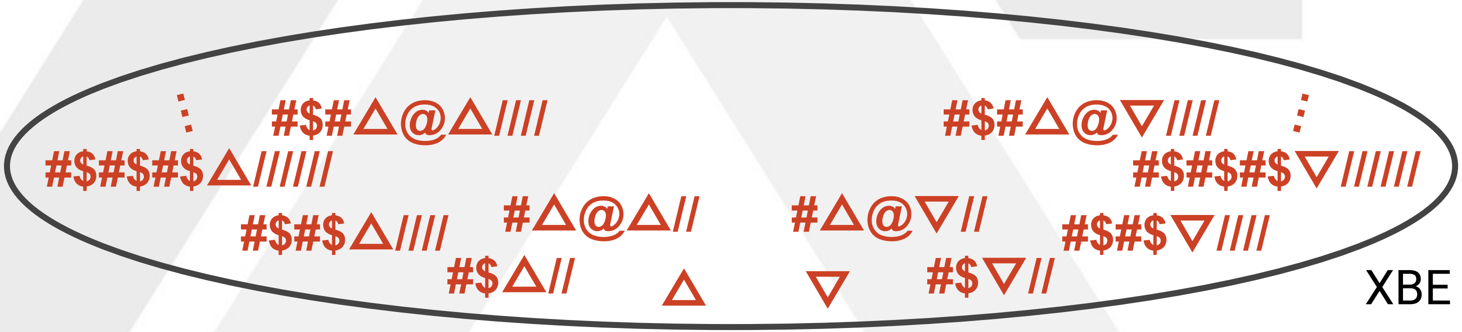
Inducción estructural

- Definición *inductiva* de XBE
  - **Regla base 1:** △ está en XBE
  - **Regla base 2:** ▽ está en XBE
  - **Regla inductiva 1:** si **e** está en XBE, entonces **#$e//** está en XBE
  - **Regla inductiva 2:** si $\mathbf{e_1}$ está en XBE y $\mathbf{e_2}$ está en XBE, entonces $\mathbf{\#e_1@e_2//}$ está en XBE



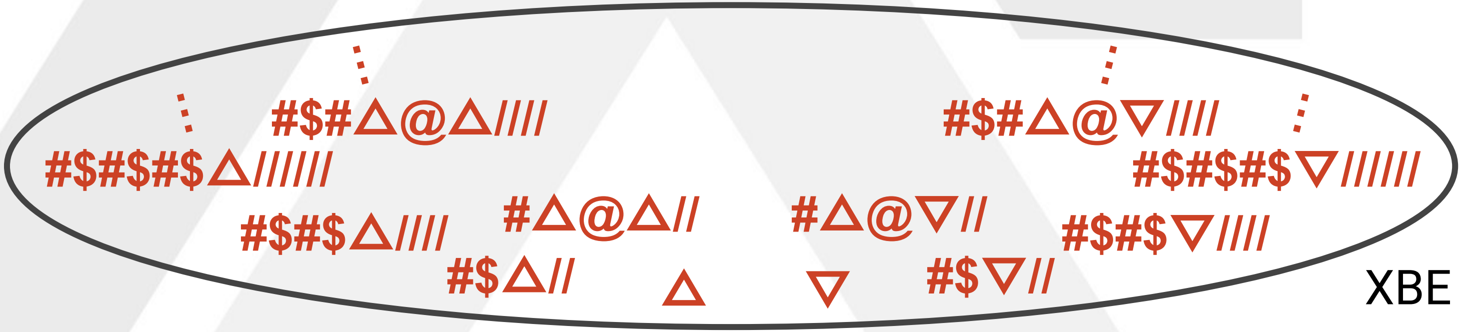
Inducción estructural

- Definición *inductiva* de XBE
  - **Regla base 1:** △ está en XBE
  - **Regla base 2:** ▽ está en XBE
  - **Regla inductiva 1:** si **e** está en XBE, entonces **#$e//** está en XBE
  - **Regla inductiva 2:** si $\mathbf{e_1}$ está en XBE y $\mathbf{e_2}$ está en XBE, entonces $\mathbf{\#e_1@e_2//}$ está en XBE



Inducción estructural

- Definición *inductiva* de XBE
  - **Regla base 1:** △ está en XBE
  - **Regla base 2:** ▽ está en XBE
  - **Regla inductiva 1:** si **e** está en XBE, entonces **#$e//** está en XBE
  - **Regla inductiva 2:** si $\mathbf{e_1}$ está en XBE y $\mathbf{e_2}$ está en XBE, entonces $\mathbf{\#e_1@e_2//}$ está en XBE

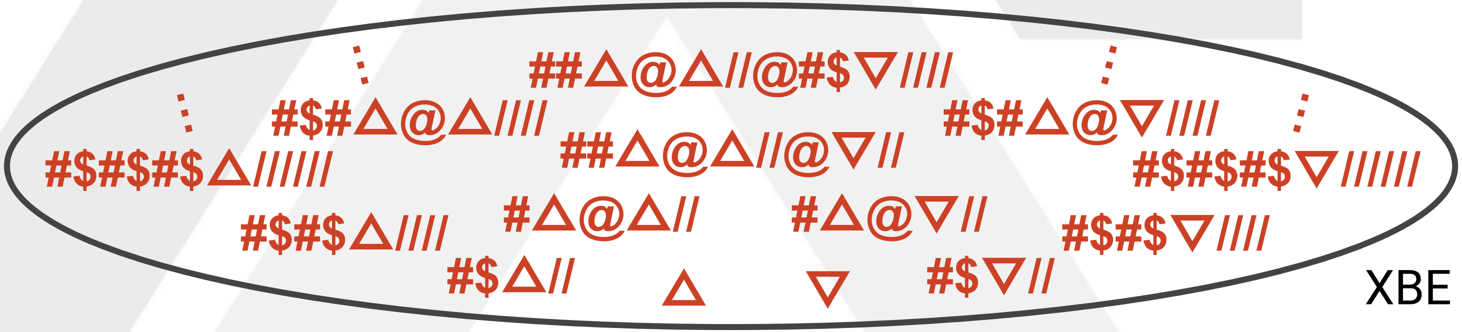


Inducción estructural

- Definición *inductiva* de XBE
  - **Regla base 1:** △ está en XBE
  - **Regla base 2:** ▽ está en XBE
  - **Regla inductiva 1:** si **e** está en XBE, entonces **#$e//** está en XBE
  - **Regla inductiva 2:** si $\mathbf{e_1}$ está en XBE y $\mathbf{e_2}$ está en XBE, entonces $\mathbf{\#e_1@e_2//}$ está en XBE

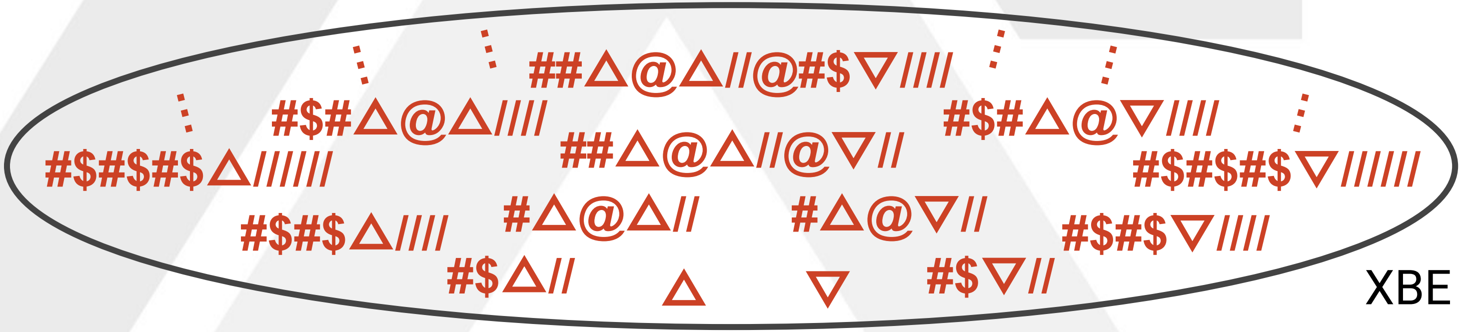
Inducción estructural

- Definición *inductiva* de XBE
  - **Regla base 1:** △ está en XBE
  - **Regla base 2:** ▽ está en XBE
  - **Regla inductiva 1:** si **e** está en XBE, entonces **#$e//** está en XBE
  - **Regla inductiva 2:** si **e_1** está en XBE y **e_2** está en XBE, entonces **#e_1@e_2//** está en XBE



Inducción estructural

- Definición *inductiva* de XBE
  - **Regla base 1:** △ está en XBE
  - **Regla base 2:** ▽ está en XBE
  - **Regla inductiva 1:** si **e** está en XBE, entonces **#$e//** está en XBE
  - **Regla inductiva 2:** si $\mathbf{e_1}$ está en XBE y $\mathbf{e_2}$ está en XBE, entonces **#$\mathbf{e_1}$@$\mathbf{e_2}$//** está en XBE



Inducción estructural

- Definición *inductiva* de XBE
  - **Regla base 1:** △ está en XBE
  - **Regla base 2:** ▽ está en XBE
  - **Regla inductiva 1:** si **e** está en XBE, entonces **#$e//** está en XBE
  - **Regla inductiva 2:** si $\mathbf{e_1}$ está en XBE y $\mathbf{e_2}$ está en XBE,
    entonces **#**$\mathbf{e_1}$**@**$\mathbf{e_2}$**//** está en XBE

¿**###$△//@#$▽////@▽//** está en XBE?

Inducción estructural

- Definición *inductiva* de XBE
  - **Regla base 1:** △ está en XBE
  - **Regla base 2:** ▽ está en XBE
  - **Regla inductiva 1:** si **e** está en XBE, entonces **#$e//** está en XBE
  - **Regla inductiva 2:** si $\mathbf{e_1}$ está en XBE y $\mathbf{e_2}$ está en XBE, entonces **#**$\mathbf{e_1}$**@**$\mathbf{e_2}$**//** está en XBE

¿Qué regla debe aplicarse?

¿**###$△//@#$▽////@▽//** está en XBE?

Inducción estructural

- Definición *inductiva* de XBE
  - **Regla base 1:** △ está en XBE
  - **Regla base 2:** ▽ está en XBE
  - **Regla inductiva 1:** si **e** está en XBE, entonces **#$e//** está en XBE
  - **Regla inductiva 2:** si $\mathbf{e_1}$ está en XBE y $\mathbf{e_2}$ está en XBE, entonces **#e_1@e_2//** está en XBE

¿Qué regla debe aplicarse?
Ri2

e1 e2

¿###$△//@#$▽////@▽// está en XBE?

¿##$△//@#$▽//// está en XBE?

¿▽ está en XBE?

Inducción estructural

- Definición *inductiva* de XBE
  - **Regla base 1:** △ está en XBE
  - **Regla base 2:** ▽ está en XBE
  - **Regla inductiva 1:** si **e** está en XBE, entonces **#$e//** está en XBE
  - **Regla inductiva 2:** si $\mathbf{e_1}$ está en XBE y $\mathbf{e_2}$ está en XBE, entonces **#$\mathbf{e_1}$@$\mathbf{e_2}$//** está en XBE

¿Qué regla debe aplicarse?
Ri2 y **Rb2**

¿###$△//@#$▽////@▽// está en XBE?

¿#$\underbrace{\boxed{\#\$\triangle//}}_{e1}$@$\underbrace{\boxed{\#\$\triangledown//}}_{e2}$// está en XBE?

¡▽ está en XBE!

¿#$△// está en XBE? ¿#$▽// está en XBE?

Inducción estructural

- Definición *inductiva* de XBE
  - **Regla base 1:** △ está en XBE
  - **Regla base 2:** ▽ está en XBE
  - **Regla inductiva 1:** si **e** está en XBE, entonces **#$e//** está en XBE
  - **Regla inductiva 2:** si $\mathbf{e_1}$ está en XBE y $\mathbf{e_2}$ está en XBE, entonces $\mathbf{\#e_1@e_2//}$ está en XBE

¿Qué regla debe aplicarse?
Ri1 (×2)

¿###$△//@#$▽////@▽// está en XBE?

¿##$△//@#$▽//// está en XBE?

¡▽ está en XBE!

¿#$$\boxed{△}^{e}$// está en XBE?

¿#$$\boxed{▽}^{e'}$// está en XBE?

¿△ está en XBE?

¿▽ está en XBE?

Inducción estructural

- Definición *inductiva* de XBE
  - **Regla base 1:** △ está en XBE
  - **Regla base 2:** ▽ está en XBE
  - **Regla inductiva 1:** si **e** está en XBE, entonces **#$e//** está en XBE
  - **Regla inductiva 2:** si $\mathbf{e_1}$ está en XBE y $\mathbf{e_2}$ está en XBE, entonces **#**$\mathbf{e_1}$**@**$\mathbf{e_2}$**//** está en XBE

¿Qué regla debe aplicarse?
Rb1 y **Rb2**

¿**###$△//@#$▽////@▽//** está en XBE?

¿**##$△//@#$▽////** está en XBE?

¡**▽** está en XBE!

¿**#$△//** está en XBE?

¿**#$▽//** está en XBE?

¡**△** está en XBE!

¡**▽** está en XBE!

Inducción estructural

- Definición *inductiva* de XBE
  - **Regla base 1:** △ está en XBE
  - **Regla base 2:** ▽ está en XBE
  - **Regla inductiva 1:** si **e** está en XBE, entonces **#$e//** está en XBE
  - **Regla inductiva 2:** si $\mathbf{e_1}$ está en XBE y $\mathbf{e_2}$ está en XBE, entonces $\mathbf{\#e_1@e_2//}$ está en XBE

¡###$△//@#$▽////@▽// está en XBE!

¡##$△//@#$▽//// está en XBE!

¡▽ está en XBE!

¡#$△// está en XBE!

¡#$▽// está en XBE!

¡△ está en XBE!

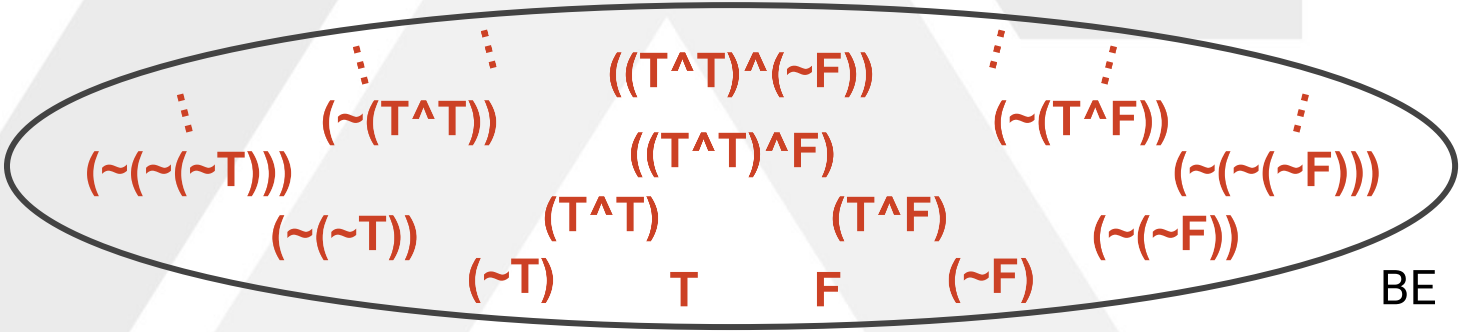
¡▽ está en XBE!

Inducción estructural

- Definición *inductiva* de BE
  - **Regla base 1:** **T** está en BE
  - **Regla base 2:** **F** está en BE
  - **Regla inductiva 1:** si **e** está en BE, entonces **(~e)** está en BE
  - **Regla inductiva 2:** si $\mathbf{e_1}$ está en BE y $\mathbf{e_2}$ está en BE, entonces $\mathbf{(e_1 \wedge e_2)}$ está en BE

Inducción estructural

- Definición *inductiva* de BE
  - **Regla base 1:** **T** está en BE
  - **Regla base 2:** **F** está en BE
  - **Regla inductiva 1:** si **e** está en BE, entonces **(~e)** está en BE
  - **Regla inductiva 2:** si $\mathbf{e_1}$ está en BE y $\mathbf{e_2}$ está en BE, entonces **(**$\mathbf{e_1}$**^**$\mathbf{e_2}$**)** está en BE



Inducción estructural

- Definición *inductiva* de BE
  - **Regla base 1:** **T** está en BE
  - **Regla base 2:** **F** está en BE
  - **Regla inductiva 1:** si **e** está en BE, entonces **(~e)** está en BE
  - **Regla inductiva 2:** si $\mathbf{e_1}$ está en BE y $\mathbf{e_2}$ está en BE, entonces $\mathbf{(e_1\wedge e_2)}$ está en BE

¡**(((~T)^(~F))^F)** está en BE!

¡**((~T)^(~F))** está en BE!

¡**F** está en BE!

¡**(~T)** está en BE!

¡**(~F)** está en BE!

¡**T** está en BE!

¡**F** está en BE!

Recursión estructural

Recursión estructural

- Comparar las definiciones de XBE y de BE
  - ¿Qué características comparten?
    - **Regla base 1:** △ está en XBE
    - **Regla base 2:** ▽ está en XBE
    - **Regla inductiva 1:** si **e** está en XBE, entonces **#$e//** está en XBE
    - **Regla inductiva 2:** si $\mathbf{e_1}$ está en XBE y $\mathbf{e_2}$ está en XBE, entonces **#**$\mathbf{e_1}$**@**$\mathbf{e_2}$**//** está en XBE

    - **Regla base 1:** **T** está en BE
    - **Regla base 2:** **F** está en BE
    - **Regla inductiva 1:** si **e** está en BE, entonces **(~e)** está en BE
    - **Regla inductiva 2:** si $\mathbf{e_1}$ está en BE y $\mathbf{e_2}$ está en BE, entonces **(**$\mathbf{e_1}$**^**$\mathbf{e_2}$**)** está en BE

Recursión estructural

- Comparar las definiciones de XBE y de BE
  - ¿Qué características comparten?
    - Misma cantidad de reglas base y misma cantidad de reglas inductivas con la misma cantidad de antecedentes cada una
  - ¿Qué implica esto?
    - Los conjuntos tienen la misma cantidad de elementos
    - Los órdenes “es parte de” de ambos son similares
  - Los conjuntos tienen la misma ***estructura***
    - Los símbolos particulares NO son importantes

Recursión estructural

- ¿Cómo definir funciones sobre los elementos de un conjunto inductivo **S**?

Recursión estructural

- ¿Cómo definir funciones sobre los elementos de un conjunto inductivo **S**?
  - Aprovechar la *estructura*

Recursión estructural

- ¿Cómo definir funciones sobre los elementos de un conjunto inductivo **S**?
  - Aprovechar la *estructura*

$$
\begin{array}{l}
f :: S \to T \\
f\ z_1 = \ldots \\
\ldots \\
f\ z_n = \ldots \\
f\ e_1 = \ldots f\ e_{11} \ldots f\ e_{i1} \ldots \\
\ldots \\
f\ e_k = \ldots f\ e_{1k} \ldots f\ e_{ik} \ldots
\end{array}
$$

Recursión estructural

Definición de una función por ***recursión estructural***

- Por cada **elemento base**, dar el resultado directamente
- Por cada **elemento inductivo**, dar el resultado usando el valor de transformar las *partes* con la misma *función que se define*

Recursión estructural

Definición de **f :: S -> T** por ***recursión estructural***

```
f :: S -> T
f z1 = ...
...
f zn = ...
f e1 = ... f e11 ... f ei1 ...
...
f ek = ... f e1k ... f eik ...
```

Recursión estructural

- Recordar la definición *inductiva* de XBE
  - **Regla base 1:** △ está en XBE
  - **Regla base 2:** ▽ está en XBE
  - **Regla inductiva 1:** si **e** está en XBE, entonces **#$e//** está en XBE
  - **Regla inductiva 2:** si $\mathbf{e_1}$ está en XBE y $\mathbf{e_2}$ está en XBE, entonces $\mathbf{\#e_1@e_2//}$ está en XBE
- Definir `nhash,nbar :: XBE -> Int`, el número de **#** y el de **/**
  - ¿Por qué se precisa recursión estructural?
    - ¿Cómo saber cuántos **#** hay en **#$e//** sin saber cuántos hay en `e`?

Recursión estructural

- Definir `nhash,nbar :: XBE -> Int`, el número de # y el de /

Recursión estructural

- Definir `nhash,nbar :: XBE -> Int`, el número de # y el de /
  - Por recursión estructural

Se decide usar recursión estructural

Recursión estructural

- Definir **`nhash,nbar :: XBE -> Int`**, el número de # y el de /
  - Por recursión estructural

```
nhash :: XBE -> Int
nhash △          = ...
nhash ▽          = ...
nhash #$e//      = ... nhash e ...
nhash #e1@e2//   = ... nhash e1 ... nhash e2 ...

nbar :: XBE -> Int
nbar △          = ...
nbar ▽          = ...
nbar #$e//      = ... nbar e ...
nbar #e1@e2//   = ... nbar e1 ... nbar e2 ...
```

Se plantea la estructura en base a las reglas

Recursión estructural

- Definir **nhash,nbar :: XBE -> Int**, el número de # y el de /
  - Por recursión estructural

nhash :: XBE -> Int
nhash △ = ...
nhash ▽ = ...
nhash #$e// = 1 + nhash e
nhash #e$_1$@e$_2$// = 1 + nhash e$_1$ + nhash e$_2$

nbar :: XBE -> Int
nbar △ = ...
nbar ▽ = ...
nbar #$e// = 2 + nbar e
nbar #e$_1$@e$_2$// = 2 + nbar e$_1$ + nbar e$_2$

Se definen los casos inductivos

Recursión estructural

- Definir **nhash,nbar :: XBE -> Int**, el número de # y el de /
  - Por recursión estructural

```
nhash :: XBE -> Int
nhash △         = 0
nhash ▽         = 0
nhash #$e//     = 1 + nhash e
nhash #e1@e2//  = 1 + nhash e1 + nhash e2

nbar :: XBE -> Int
nbar △          = 0
nbar ▽          = 0
nbar #$e//      = 2 + nbar e
nbar #e1@e2//   = 2 + nbar e1 + nbar e2
```

Se completa con los casos base

Recursión estructural

- ¿Y por qué funciona la *recursión estructural*?
  - Si usa la misma función, ¿por qué termina (y no da $\perp$)?

Recursión estructural

- ¿Y por qué funciona la *recursión estructural*?
  - Si usa la misma función, ¿por qué termina (y no da $\perp$)?
  - ¡El orden “**es parte de**” es *bien fundado*!
    - O sea, nunca se puede descomponer un elemento en sus partes de forma infinita…
    - De esta forma, la reducción *tiene que terminar*

Recursión estructural

- ¿Y por qué funciona la *recursión estructural*?
  - Si usa la misma función, ¿por qué termina (y no da $\perp$)?
  - ¡El orden “**es parte de**” es *bien fundado*!
    - O sea, nunca se puede descomponer un elemento en sus partes de forma infinita…
    - De esta forma, la reducción *debe terminar*
  - La condición de ser el ***menor*** es esencial para esto

Recursión estructural

- ¿Y por qué funciona la *recursión estructural*?
  - Si usa la misma función, ¿por qué termina (y no da $\perp$)?
  - ¡El orden "**es parte de**" es *bien fundado*!
    - O sea, nunca se puede descomponer un elemento en sus partes de forma infinita…
    - De esta forma, la reducción *debe terminar*
  - La condición de ser el ***menor*** es esencial para esto
  - **Una función recursiva estructural *termina* de reducir para todos los elementos inductivos**

Inducción estructural

- ¿Y por qué funciona la *recursión estructural*?
  - Si usa la misma función, ¿por qué termina (y no da ⊥)?
    - Las reglas dan una forma de desarmar un elemento en sus partes
    - Esa forma es aprovechada por la reducción

`nbar` ###\$△//@#\$▽////@▽// = ??

Inducción estructural

- ¿Y por qué funciona la *recursión estructural*?
  - Si usa la misma función, ¿por qué termina (y no da ⊥)?
    - Las reglas dan una forma de desarmar un elemento en sus partes
    - Esa forma es aprovechada por la reducción

`nbar` ###$△//@#$▽////@▽// = ??

`nbar` ##$△//@#$▽//// = ??   `nbar` ▽ = ??

`nbar` #$△// = ??   `nbar` #$▽// = ??

`nbar` △ = ??   `nbar` ▽ = ??

Inducción estructural

- ¿Y por qué funciona la *recursión estructural*?
  - Si usa la misma función, ¿por qué termina (y no da $\perp$)?
    - Las reglas dan una forma de desarmar un elemento en sus partes
    - Esa forma es aprovechada por la reducción

`nbar` ###$△//@#$▽////@▽// = 2+6+0

`nbar` ##$△//@#$▽//// = 2+2+2 `nbar` ▽ = 0

`nbar` #$△// = 2+0 `nbar` #$▽// = 2+0

`nbar` △ = 0 `nbar` ▽ = 0

Inducción estructural

- ¿Y por qué funciona la *recursión estructural*?
  - Si usa la misma función, ¿por qué termina (y no da ⊥)?
    - Las reglas dan una forma de desarmar un elemento en sus partes
    - Esa forma es aprovechada por la reducción

`nbar` ###$△//@#$▽////@▽// = 8

`nbar` ##$△//@#$▽//// = 6 `nbar` ▽ = 0

`nbar` #$△// = 2 `nbar` #$▽// = 2

`nbar` △ = 0 `nbar` ▽ = 0

Recursión estructural

- Definir `xbe2be :: XBE -> BE` y `be2xbe :: BE -> XBE` que transformen elementos de un conjunto a otro y de vuelta

Recursión estructural

- Definir `xbe2be :: XBE -> BE` y `be2xbe :: BE -> XBE` que transformen elementos de un conjunto a otro y de vuelta
  - Por recursión estructural

Se decide usar recursión estructural

Recursión estructural

- Definir **`xbe2be :: XBE -> BE`** y **`be2xbe :: BE -> XBE`** que transformen elementos de un conjunto a otro y de vuelta
  - Por recursión estructural

```
xbe2be :: XBE -> BE
xbe2be △        = ...
xbe2be ▽        = ...
xbe2be #$e//    = ... xbe2be e ...
xbe2be #e1@e2// = ... xbe2be e1 ... xbe2be e2 ...

be2xbe :: BE -> XBE
be2xbe T       = ...
be2xbe F       = ...
be2xbe (~e)    = ... be2xbe e ...
be2xbe (e1^e2) = ... be2xbe e1 ... be2xbe e2 ...
```

Se plantea la estructura en base a las reglas

Recursión estructural

- Definir **`xbe2be :: XBE -> BE`** y **`be2xbe :: BE -> XBE`** que transformen elementos de un conjunto a otro y de vuelta
  - Por recursión estructural

```
xbe2be :: XBE -> BE
xbe2be △         = ...
xbe2be ▽         = ...
xbe2be #$e//     = (~xbe2be e)
xbe2be #e1@e2//  = (xbe2be e1^xbe2be e2)

be2xbe :: BE -> XBE
be2xbe T       = ...
be2xbe F       = ...
be2xbe (~e)    = #$be2xbe e//
be2xbe (e1^e2) = #be2xbe e1@be2xbe e2//
```

Se definen los casos inductivos

Recursión estructural

- Definir **xbe2be :: XBE -> BE** y **be2xbe :: BE -> XBE** que transformen elementos de un conjunto a otro y de vuelta
  - Por recursión estructural

```
xbe2be :: XBE -> BE
xbe2be △          = T
xbe2be ▽          = F
xbe2be #$e//      = (~xbe2be e)
xbe2be #e1@e2//   = (xbe2be e1^xbe2be e2)

be2xbe :: BE -> XBE
be2xbe T       = △
be2xbe F       = ▽
be2xbe (~e)    = #$be2xbe e//
be2xbe (e1^e2) = #be2xbe e1@be2xbe e2//
```

Se completa con los casos base

Ejemplo de aplicación

Uso de la inducción/recursión estructural

- Definición *inductiva* de $\mathbb{N}$
  - **Regla base:** 0 está en $\mathbb{N}$
  - **Regla inductiva:** si **n** está en $\mathbb{N}$, entonces **n+1** está en $\mathbb{N}$
- Observar que las reglas son productivas
- Se cumplen las propiedades de conjuntos inductivos
  - Tiene infinitos elementos
  - Todos sus elementos son finitos
  - El orden usual de *“menor”* (<) es *“es parte de”*

Uso de la inducción/recursión estructural

- Definición *inductiva* de $\mathbb{N}$
  - **Regla base:** 0 está en $\mathbb{N}$
  - **Regla inductiva:** si **n** está en $\mathbb{N}$, entonces **n+1** está en $\mathbb{N}$
- ¿Cómo sería una definición de función sobre $\mathbb{N}$?

Uso de la inducción/recursión estructural

- Definición *inductiva* de $\mathbb{N}$
  - **Regla base:** 0 está en $\mathbb{N}$
  - **Regla inductiva:** si **n** está en $\mathbb{N}$, entonces **n+1** está en $\mathbb{N}$
- ¿Cómo sería una definición de función sobre $\mathbb{N}$?

```
f :: ℕ -> T
f  0       = ...
f  (n+1)   = ... f  n ...
```

Uso de la inducción/recursión estructural

- Definición *inductiva* de $\mathbb{N}$
  - **Regla base:** 0 está en $\mathbb{N}$
  - **Regla inductiva:** si **n** está en $\mathbb{N}$, entonces **n+1** está en $\mathbb{N}$
- ¿Cómo sería una definición de función sobre $\mathbb{N}$?

```
f :: ℕ -> T
f 0       = ...
f (n+1)   = ... f n ...
```

Usualmente se escribe de esta otra forma

```
f :: ℕ -> T
f 0 = ...
f n = ... f (n-1) ...
```

Uso de la inducción/recursión estructural

- Definición *inductiva* de $\mathbb{N}$
  - **Regla base:** 0 está en $\mathbb{N}$
  - **Regla inductiva:** si **n** está en $\mathbb{N}$, entonces **n+1** está en $\mathbb{N}$
- ```
  f :: ℕ -> T
  f 0 = ...
  f n = ... f (n-1) ...
  ```
- ¡Los números naturales son un conjunto inductivo estructural!
- Funciones como **`factorial`** funcionan por su estructura
  ```

Inducción y recursión en Haskell

Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición inductiva de estructuras?
  - ¿Definición de funciones por recursión estructural?
  - ¿Y la 3era pregunta?
- ¿Qué cosas interesantes se pueden expresar mediante la utilización de estas técnicas?

Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición inductiva de estructuras?

Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición inductiva de estructuras?
    - Usando tipos algebraicos

Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición inductiva de estructuras?
    - Usando tipos algebraicos
  - ¿Cómo expresar casos base?
  - ¿Cómo expresar casos recursivos?

Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición inductiva de estructuras?
    - Usando tipos algebraicos
  - ¿Cómo expresar casos base?
    - Usando constructores con o sin argumentos
  - ¿Cómo expresar casos recursivos?
    - ¡Usando constructores que usen el mismo tipo definido como argumento!

Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición inductiva de estructuras?

Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición inductiva de estructuras?

```haskell
data Mapa = Equis Cofre | Recto Int Mapa | Giro Dir Mapa
data Cofre = Monedas Int | CorazonDe Pirata | Joyas Int | Nada
data Pirata = DavyJones | WillTurner
data Dir = Izq | Der
```

Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición inductiva de estructuras?

```haskell
data Mapa = Equis Cofre | Recto Int Mapa | Giro Dir Mapa
data Cofre = Monedas Int | CorazonDe Pirata | Joyas Int | Nada
data Pirata = DavyJones | WillTurner
data Dir = Izq | Der
```

Caso base

Casos inductivos

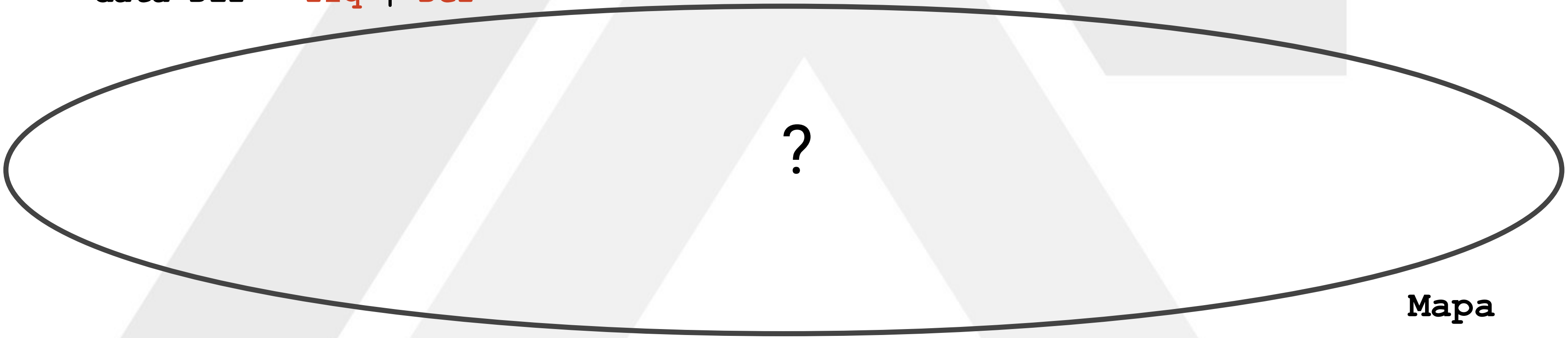
Partes NO inductivas

Partes inductivas

Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición inductiva de estructuras?

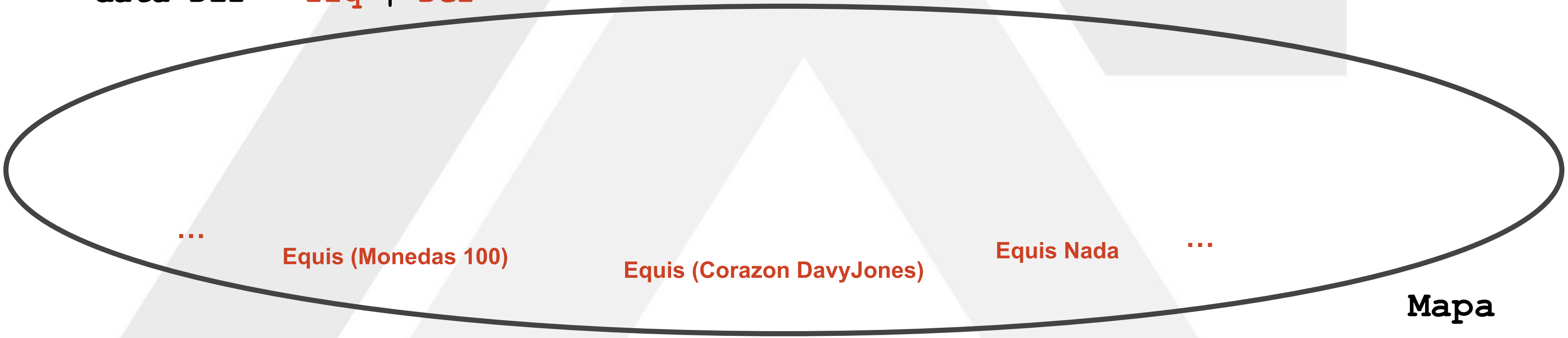
```
data Mapa = Equis Cofre | Recto Int Mapa | Giro Dir Mapa
data Cofre = Monedas Int | CorazonDe Pirata | Joyas Int | Nada
data Pirata = DavyJones | WillTurner
data Dir = Izq | Der
```



Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición inductiva de estructuras?

```
data Mapa = Equis Cofre | Recto Int Mapa | Giro Dir Mapa
data Cofre = Monedas Int | CorazonDe Pirata | Joyas Int | Nada
data Pirata = DavyJones | WillTurner
data Dir = Izq | Der
```



Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición inductiva de estructuras?

```
data Mapa = Equis Cofre | Recto Int Mapa | Giro Dir Mapa
data Cofre = Monedas Int | CorazonDe Pirata | Joyas Int | Nada
data Pirata = DavyJones | WillTurner
data Dir = Izq | Der
```

⋮

Recto 30 (Giro Izq (Recto 10 (Equis (Monedas 250))))

Giro Izq (Recto 10 (Equis (Monedas 250)))

Giro Der (Recto 30 (Equis Nada))

Recto 30 (Equis Nada)

…

Recto 10 (Equis (Monedas 250))

Equis (Monedas 100)

Equis Nada

…

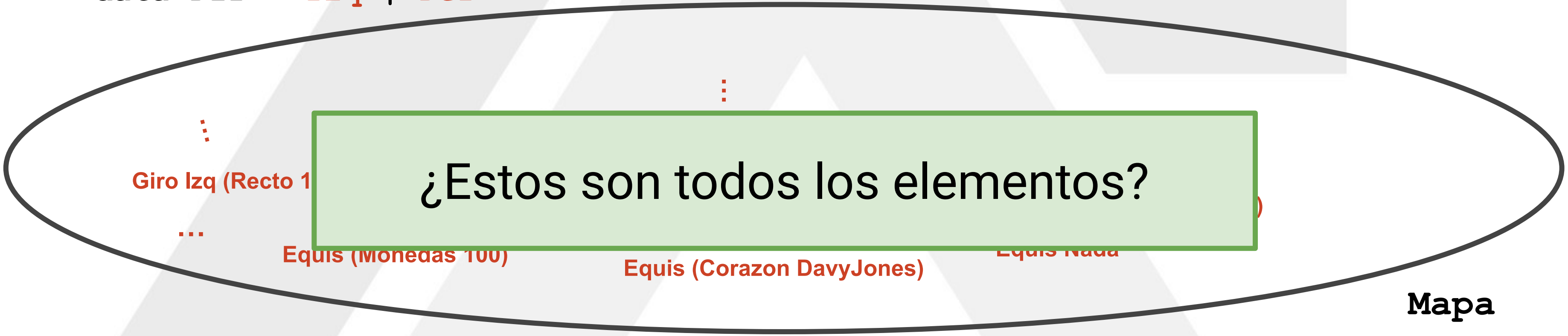
Equis (Corazon DavyJones)

Mapa

Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición inductiva de estructuras?

```
data Mapa = Equis Cofre | Recto Int Mapa | Giro Dir Mapa
data Cofre = Monedas Int | CorazonDe Pirata | Joyas Int | Nada
data Pirata = DavyJones | WillTurner
data Dir = Izq | Der
```



¿Estos son todos los elementos?

Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición inductiva de estructuras?

```
data Mapa = Equis Cofre | Recto Int Mapa | Giro Dir Mapa
data Cofre = Monedas Int | CorazonDe Pirata | Joyas Int | Nada
data Pirata = DavyJones | WillTurner
data Dir = Izq | Der
```

Mapa

Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición inductiva de estructuras?

```
data Mapa = Equis Cofre | Recto Int Mapa | Giro Dir Mapa
data Cofre = Monedas Int | CorazonDe Pirata | Joyas Int | Nada
data Pirata = DavyJones | WillTurner
data Dir = Izq | Der
```

Inducción y recursión en Haskell

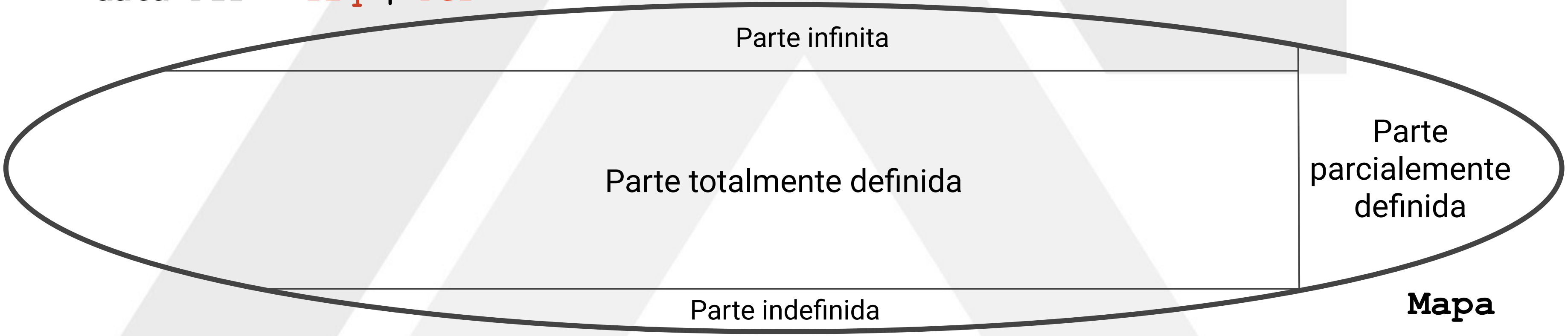
- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición inductiva de estructuras?

```
data Mapa = Equis Cofre | Recto Int Mapa | Giro Dir Mapa
data Cofre = Monedas Int | CorazonDe Pirata | Joyas Int | Nada
data Pirata = DavyJones | WillTurner
data Dir = Izq | Der
```

Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición inductiva de estructuras?

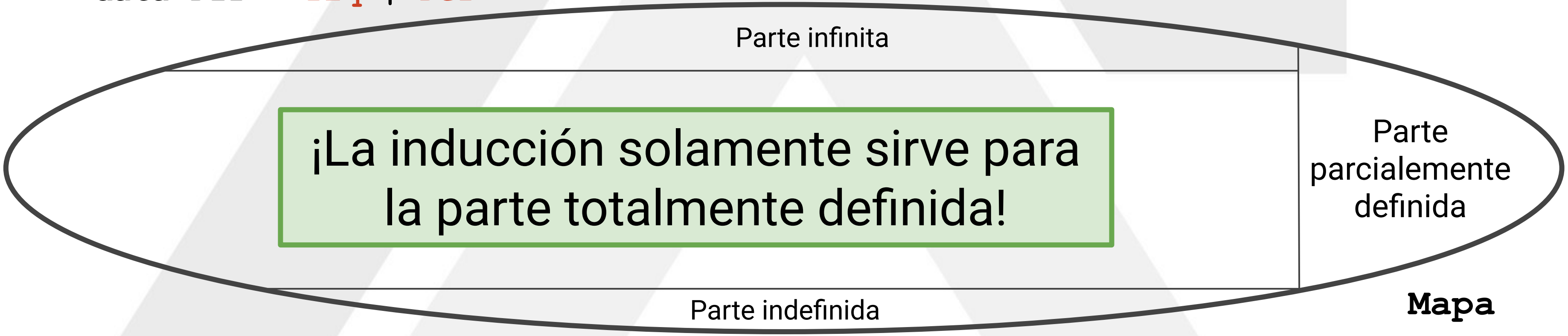
```
data Mapa = Equis Cofre | Recto Int Mapa | Giro Dir Mapa
data Cofre = Monedas Int | CorazonDe Pirata | Joyas Int | Nada
data Pirata = DavyJones | WillTurner
data Dir = Izq | Der
```



Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición inductiva de estructuras?

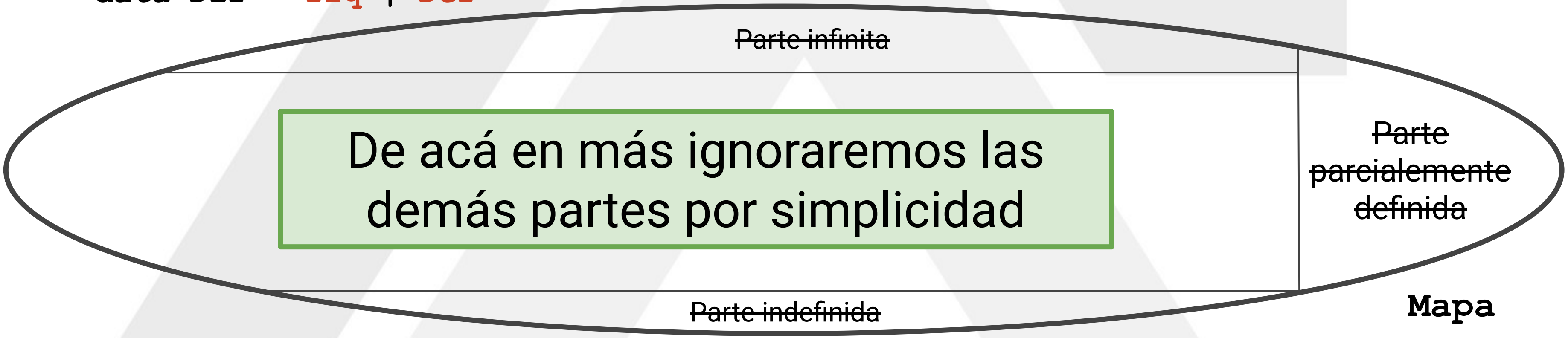
```haskell
data Mapa = Equis Cofre | Recto Int Mapa | Giro Dir Mapa
data Cofre = Monedas Int | CorazonDe Pirata | Joyas Int | Nada
data Pirata = DavyJones | WillTurner
data Dir = Izq | Der
```



Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición inductiva de estructuras?

```
data Mapa = Equis Cofre | Recto Int Mapa | Giro Dir Mapa
data Cofre = Monedas Int | CorazonDe Pirata | Joyas Int | Nada
data Pirata = DavyJones | WillTurner
data Dir = Izq | Der
```



Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición inductiva de estructuras?

```
data Mapa = Equis Cofre | Recto Int Mapa | Giro Dir Mapa
data Cofre = Monedas Int | CorazonDe Pirata | Joyas Int | Nada
data Pirata = DavyJones | WillTurner
data Dir = Izq | Der
```

Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición de funciones por recursión estructural?

Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición de funciones por recursión estructural?

```
data Mapa = Equis Cofre | Recto Int Mapa | Giro Dir Mapa

tesoroEn :: Mapa -> Cofre
tesoroEn ...
```

Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición de funciones por recursión estructural?

```
data Mapa = Equis Cofre | Recto Int Mapa | Giro Dir Mapa

tesoroEn :: Mapa -> Cofre
tesoroEn ...
```

¿Cómo definirla?

Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición de funciones por recursión estructural?

```
data Mapa = Equis Cofre | Recto Int Mapa | Giro Dir Mapa

tesoroEn :: Mapa -> Cofre
tesoroEn ...
```

¡Por recursión en la estructura de **Mapa**!

¿Cómo definirla?

Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición de funciones por recursión estructural?

```
data Mapa = Equis Cofre | Recto Int Mapa | Giro Dir Mapa

tesoroEn :: Mapa -> Cofre
tesoroEn (Equis c)    = … c …
tesoroEn (Recto n m) = … n … tesoroEn m …
tesoroEn (Giro d m)  = … d … tesoroEn m …
```

¡Por recursión en la estructura de **Mapa**!

Primero se plantea el esquema

Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición de funciones por recursión estructural?

```
data Mapa = Equis Cofre | Recto Int Mapa | Giro Dir Mapa

tesoroEn :: Mapa -> Cofre
tesoroEn (Equis c)    = ... c ...
tesoroEn (Recto n m)  = caminarYDar n (tesoroEn m)
tesoroEn (Giro d m)   = girarYDar d (tesoroEn m)
```

¡Se definen los casos inductivos!

Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición de funciones por recursión estructural?

```
data Mapa = Equis Cofre | Recto Int Mapa | Giro Dir Mapa

tesoroEn :: Mapa -> Cofre
tesoroEn (Equis c)    = dar c
tesoroEn (Recto n m)  = caminarYDar n (tesoroEn m)
tesoroEn (Giro d m)   = girarYDar d (tesoroEn m)
```

Se definen los casos base

Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición de funciones por recursión estructural?

```
data Mapa = Equis Cofre | Recto Int Mapa | Giro Dir Mapa

tesoroEn :: Mapa -> Cofre
tesoroEn (Equis c)    = dar c
tesoroEn (Recto n m)  = caminarYDar n (tesoroEn m)
tesoroEn (Giro d m)   = girarDar d (tesoroEn m)

dar c            = c    -- Implementaciones triviales
caminarYDar n c  = c    -- Podrían complicarse con un
girarYDar d c    = c    --  terreno y verificaciones…
```

Se completa con las funciones auxiliares

Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición de funciones por recursión estructural?

```
data Mapa = Equis Cofre | Recto Int Mapa | Giro Dir Mapa

comprimido :: Mapa -> Mapa   -- Comprime tramos rectos consecutivos
comprimido (Equis c)    = ... c ...
comprimido (Recto n m) = ... n ... comprimido m ...
comprimido (Giro d m)  = ... d ... comprimido m ...
```

Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición de funciones por recursión estructural?

```
data Mapa = Equis Cofre | Recto Int Mapa | Giro Dir Mapa

comprimido :: Mapa -> Mapa  -- Comprime tramos rectos consecutivos
comprimido (Equis c)    = … c …
comprimido (Recto n m)  = compRecto n (comprimido m)
comprimido (Giro d m)   = Giro d (comprimido m)
```

Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición de funciones por recursión estructural?

```
data Mapa = Equis Cofre | Recto Int Mapa | Giro Dir Mapa

comprimido :: Mapa -> Mapa  -- Comprime tramos rectos consecutivos
comprimido (Equis c)   = Equis c
comprimido (Recto n m) = compRecto n (comprimido m)
comprimido (Giro d m)  = Giro d (comprimido m)
```

Inducción y recursión en Haskell

- ¿Cómo usar todas estas técnicas en Haskell?
  - ¿Definición de funciones por recursión estructural?

```
data Mapa = Equis Cofre | Recto Int Mapa | Giro Dir Mapa

comprimido :: Mapa -> Mapa  -- Comprime tramos rectos consecutivos
comprimido (Equis c)   = Equis c
comprimido (Recto n m) = compRecto n (comprimido m)
comprimido (Giro d m)  = Giro d (comprimido m)

compRecto n (Recto n' m) = Recto (n+n') m
compRecto n m            = Recto n m
```

Inducción y recursión en Haskell

- ¿Qué cosas interesantes se pueden expresar?
- ¿Y qué pasó con la 3era pregunta?

Producción de video de
Valeria Sawady O'Connor

Inc

Producción de video de
Valeria Sawady O'Connor

Inc

Producción de video de
Valeria Sawady O’Connor

Inducción y recursión en Haskell

- ¿Qué cosas interesantes se pueden expresar?
- ¿Y qué pasó con la 3era pregunta?
  - Quizás pueda conocerse algo de esto en…

La Programación Desconocida

Nah. En la próxima clase. :)

Resumen

Resumen

- ***Definición*** de conjuntos ***por inducción estructural***
  - **Reglas base**, **reglas inductivas**, el *MENOR*
- ***Definición*** de funciones ***por recursión estructural***
  - Una ecuación por cada regla base e inductiva
- Aplicación de las ideas en Haskell
  - Tipos algebraicos recursivos
  - Funciones por recursión estructural sobre ellos