LIMITACIONES DE LA DIAGONALIZACIÓN

- Para demostrar la indecidibilidad del siguiente conjunto:

$$K_0 = \{ x : \varphi_x(0)\downarrow \}$$

¿Es aplicable el método?
NO

- Supongamos demostrado por diagonalización que la siguiente función es incomputable:

$$f(x) = \begin{cases} \text{true} & \text{si } W_x = \{ y : y \bmod 2 = 0 \} \\ \text{false} & \text{c.c.} \end{cases}$$

Para demostrar que la siguiente función tampoco es computable:

$$g(x) = \begin{cases} \text{true} & \text{si } W_x = \{ y : y \bmod 2 \neq 0 \} \\ \text{false} & \text{c.c.} \end{cases}$$

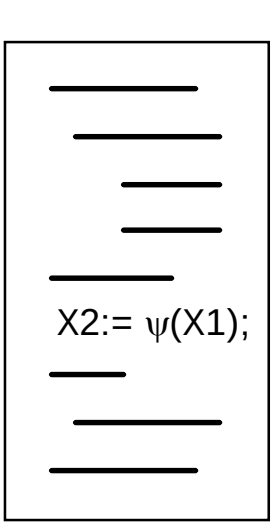
¿Nos ayuda la incomputabilidad de **f** para demostrar la de **g**?
NO

REDUCCIÓN ENTRE PROBLEMAS

- Decimos (coloquialmente) que el problema A se reduce al problema B cuando BASTA CON solucionar B PARA solucionar A.
  - ¿Quiero aprender a multiplicar?. Basta con saber sumar
  - ¿Quiero aprender inglés? Basta con perder el miedo al ridículo
  - ¿Quiero aprobar Esforciología de la Mancomunicación II?. Basta con comprarte el libro del profesor Paco Ruptillo y que te vea mucho con él bajo el brazo
- Cuando A se reduce a B tenemos un algoritmo que a partir de la solución de B nos resuelve A. Pero ello no implica que TENGAMOS dicha solución de B.
  - ¿Quiero salvar la vida de Kennedy?. Basta con viajar al pasado y convencerle para que no vaya a Dallas.
- Tampoco implica que la solución de B SEA IMPRESCINDIBLE para resolver A
  - ¿Quiero que mi compañero de piso deje de fumar?. Basta con contratar un asesino profesional.
- Pero si A se reduce a B y A es IMPOSIBLE de resolver, entonces B también es imposible
  - ¿Quiero que EEUU se retire de Iraq?. Basta con que su gobierno esté interesado en la paz mundial.

LA REDUCCIÓN EN COMPUTABILIDAD

- Decimos que la función χ se reduce a la función ψ cuando un programa-while para computar ψ nos permita construir otro para χ.

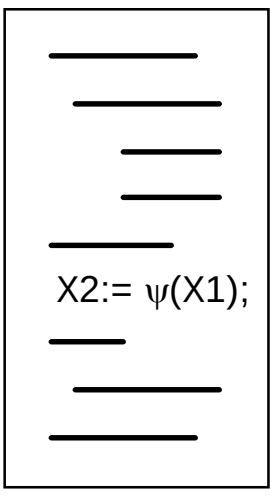


ψ computable

$\chi \cong \varphi_P$

$\Rightarrow \chi$ computable

- Entonces la computabilidad de ψ implica la de χ.



χ incomputable

$\chi \cong \varphi_P$

$\Rightarrow \psi$ incomputable

- Pero, sobre todo, si χ resulta INCOMPUTABLE ψ también lo será.

ESQUEMA DE REDUCCIÓN

- Para demostrar que una función ψ no es computable

1. Suponemos que ψ es computable

2. Elegimos otra función χ que no sea computable.

3. Basándonos en la presunta computabilidad de ψ construimos un programa que compute a χ (absurdo)

4. Concluimos que ψ no era computable

EJEMPLO I: EL PROBLEMA DE LA INICIALIZACIÓN CORRECTA DE VARIABLES

- Queremos ver si es decidible el predicado:

  I(x,y,n) $\Leftrightarrow$ al ejecutar el programa P_x sobre el dato y la variable XN es utilizada en algún momento anterior a su inicialización explícita.

- Reduciremos **H** a **I**

  1. Sean el programa P_x y el dato y. Queremos saber si $P_x(y)\downarrow$
  2. Observando P_x determinamos una variable XN que no aparezca en él.
  3. Añadimos X0:=cdr(XN); al final de P_x obteniendo P_z
  4. Preguntamos si **I(z,y,N)**. Dado que **XN** sólo aparece al final del programa, sólo puede utilizarse incorrectamente si P_z (y por tanto P_x) termina. Por ello **I(z,y,N)** $\Rightarrow P_x(y)\downarrow$, mientras que $\neg$**I(z,y,N)** $\Rightarrow P_x(y)\uparrow$

```
N:=ult_variable(X1)+1;
Z:=haz_composición(X1, haz_asig_cdr(0, N));
X0:=I(Z, X2, N);
```

EJEMPLO II: K_0 NO ES DECIDIBLE

- Reduciremos **H** a $\mathbf{C_{K0}}$

Basta conseguir que P_z sea la expansión del programa

```
X1:=y;
Px
```

Pues tendremos $P_z(0)\downarrow \Leftrightarrow P_x(y)\downarrow$

```
Z:= haz _asig_vacia(1);
AUX := X2^R;
while nonem?(AUX) loop
    if car_a?(AUX) then
        Z := haz_composición(Z,
                haz_asig_cons_a(1, 1));
        end if;
    if car_b?(AUX) then Z := haz_composición(Z,
                haz_asig_cons_b(1, 1));
    end if;
    AUX := cdr(AUX);
end loop;
Z := haz_composición(Z, X1);
X0 := C_K0(Z);
```

FUNCIONES DE REDUCCION

- Problema crucial: construcción de P_z a partir de P_x. Lo sistematizamos mediante una función total de transformación: $z = h(x)$

- Sean $A, B \in \Sigma^*$. Decimos que A es REDUCIBLE a B ($A \leq B$) si existe $h: \Sigma^* \to \Sigma^*$ computable y total que verifica:

$$\forall x \ (x \in A \leftrightarrow h(x) \in B)$$

- Propiedades:

  * $A \leq A$
  * $A \leq B \wedge B \leq C \Rightarrow A \leq C$
  * $A \leq B \Rightarrow \overline{A} \leq \overline{B}$
  * $A \leq B \wedge B \in \Sigma_0 \Rightarrow A \in \Sigma_0$

    (y por tanto, si $A \leq B \wedge A \notin \Sigma_0 \Rightarrow B \notin \Sigma_0$)
  * $A \leq B \wedge B \in \Sigma_1 \Rightarrow A \in \Sigma_1$

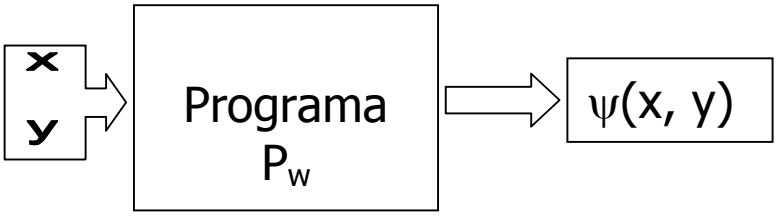
    (y por tanto, si $A \leq B \wedge A \notin \Sigma_1 \Rightarrow B \notin \Sigma_1$)

TEOREMA S_M_N (PARA M = N = 1)

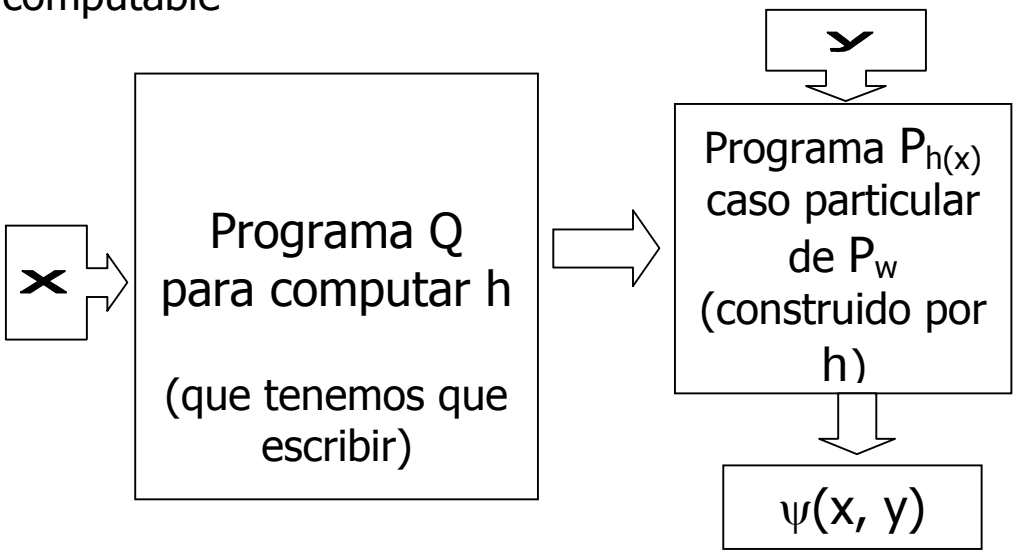
- Para toda función computable ψ^2: $\Sigma^* \times \Sigma^* \longrightarrow \Sigma^*$ existe una función h: $\Sigma^* \rightarrow \Sigma^*$ total y computable tal que

$$\psi(x, y) \cong \varphi_{h(x)}(y)$$

* ψ computable mediante P_w



* h computable



PROGRAMA Q PARA COMPUTAR LA FUNCIÓN TOTAL H DEL TEOREMA S_M_N

```
X0 := haz_asig_vacía(0);
X0 := haz_composición (X0,
                haz_composición (haz_asig_cons_a1(2, 1),
                        haz_asig_cdr(2, 2)));
X0 := haz_composición (X0,
                        haz_asig_vacía(1));
AUX := X1^R;
while nonem?(AUX) loop
      Z := primero(AUX);
      case Z is
            when a1 => X0 := haz_composición (X0,
                                    haz_asig_cons_a1(1,1));
            ...
            when ak => X0 := haz_composición (X0,
                                    haz_asig_cons_ak(1,1));
      end case;
      AUX := cdr(AUX);
end loop;
X0 := haz_composición(X0, w);
```