

Equivalenza operativa 3.3

equivalenza operativa

# Equivalenza tra programmi

La manipolazione simbolica dei programmi è possibile se possiamo garantire che la loro semantica sia preservata

Utile per studiare le trasformazioni dei programmi  
(interpreti, compilatori, ottimizzazione del codice, refactoring...

ma... quando due comandi sono equivalenti?

# Equivalenza operativa

memoria iniziale di default

un primo tentativo

$$\forall x. \sigma_0(x) = 0$$

$$c_1 \sim_o c_2 \stackrel{\Delta}{=} \forall \sigma. ( \langle c_1, \sigma_0 \rangle \longrightarrow \sigma \Leftrightarrow \langle c_2, \sigma_0 \rangle \longrightarrow \sigma )$$

Cosa non va in questa definizione?

$$x := y + 1 \sim_o x := 1$$

$$C[\bullet] \stackrel{\Delta}{=} y := 1; [\bullet]$$

$$C[x := y + 1] \stackrel{\Delta}{=} y := 1; x := y + 1$$

$$C[x := 1] \stackrel{\Delta}{=} y := 1; x := 1$$

$$C[x := y + 1] \not\sim_o C[x := 1]$$

# Operational equivalence

qualsiasi memoria iniziale

$$c_1 \sim_o c_2 \triangleq \forall \sigma, \sigma'. ( \langle c_1, \sigma \rangle \longrightarrow \sigma' \Leftrightarrow \langle c_2, \sigma \rangle \longrightarrow \sigma' )$$

molto meglio:  
e' una relazione di  
equivalenza  
e anche una congruenza  
(lo vedremo dopo)

# Equivalenza concreta

$$c \stackrel{?}{\sim}_0 w$$

$$w \triangleq \text{while } x > 0 \text{ do } x := x - 1$$

$$c \triangleq \text{if } x > 0 \text{ then } x := 0 \text{ else skip}$$

Consideriamo un  $\sigma$  generico, se  $\sigma(x) \leq 0$ :

$$\langle c, \sigma \rangle \longrightarrow \sigma'$$

$$\swarrow \langle x > 0, \sigma \rangle \longrightarrow \mathbf{ff}, \langle \text{skip}, \sigma \rangle \longrightarrow \sigma'$$

$$\swarrow^* \langle \text{skip}, \sigma \rangle \longrightarrow \sigma'$$

$$\swarrow \sigma' = \sigma \quad \square$$

$$\langle c, \sigma \rangle \longrightarrow \sigma$$

$$\langle w, \sigma \rangle \longrightarrow \sigma'$$

$$\swarrow \sigma' = \sigma \langle x > 0, \sigma \rangle \longrightarrow \mathbf{ff}$$

$$\swarrow^* \square$$

$$\langle w, \sigma \rangle \longrightarrow \sigma$$

# Esempio (cont.)

$c \stackrel{?}{\sim}_0 w$

$w \triangleq \text{while } x > 0 \text{ do } x := x - 1$

$c \triangleq \text{if } x > 0 \text{ then } x := 0 \text{ else skip}$

Consideriamo un  $\sigma$  generico, se  $\sigma(x) > 0$ :

$\langle c, \sigma \rangle \longrightarrow \sigma'$

$\swarrow \langle x > 0, \sigma \rangle \longrightarrow \mathbf{tt}, \langle x := 0, \sigma \rangle \longrightarrow \sigma'$

$\swarrow^* \langle x := 0, \sigma \rangle \longrightarrow \sigma'$

$\swarrow^* \sigma' = \sigma[0/x] \quad \square$

$\langle c, \sigma \rangle \longrightarrow \sigma[0/x]$

$\forall n > 0. P(n) \quad P(n) \triangleq \forall \sigma. \sigma(x) = n \Rightarrow \langle w, \sigma \rangle \longrightarrow \sigma[0/x]$

# Esempio (cont.)

$c \stackrel{?}{\sim}_o w$

$w \triangleq \text{while } x > 0 \text{ do } x := x - 1$

$c \triangleq \text{if } x > 0 \text{ then } x := 0 \text{ else skip}$

$\forall n > 0. P(n) \quad P(n) \triangleq \forall \sigma. \sigma(x) = n \Rightarrow \langle w, \sigma \rangle \longrightarrow \sigma[0/x]$

$P(1) \triangleq \forall \sigma. \sigma(x) = 1 \Rightarrow \langle w, \sigma \rangle \longrightarrow \sigma[0/x]$

consideriamo un  $\sigma$  generico e assumiamo  $\sigma(x) = 1$

$\langle w, \sigma \rangle \longrightarrow \sigma'$

$\swarrow \langle x > 0, \sigma \rangle \longrightarrow \mathbf{tt}, \langle x := x - 1, \sigma \rangle \longrightarrow \sigma'', \langle w, \sigma'' \rangle \longrightarrow \sigma'$

$\swarrow^* \langle x := x - 1, \sigma \rangle \longrightarrow \sigma'', \langle w, \sigma'' \rangle \longrightarrow \sigma'$

$\swarrow^*_{\sigma'' = \sigma[0/x]} \langle w, \sigma[0/x] \rangle \longrightarrow \sigma'$

$\swarrow_{\sigma' = \sigma[0/x]} \langle x > 0, \sigma[0/x] \rangle \longrightarrow \mathbf{ff}$

$\swarrow^* \square$

$\langle w, \sigma \rangle \longrightarrow \sigma[0/x]$



# Esempio (cont.)

$c \stackrel{?}{\sim}_o w$

$w \triangleq \mathbf{while} \ x > 0 \ \mathbf{do} \ x := x - 1$

$c \triangleq \mathbf{if} \ x > 0 \ \mathbf{then} \ x := 0 \ \mathbf{else} \ \mathbf{skip}$

$\forall n > 0. P(n) \quad P(n) \triangleq \forall \sigma. \sigma(x) = n \Rightarrow \langle w, \sigma \rangle \longrightarrow \sigma[0/x]$

$\forall n > 0. P(n) \Rightarrow P(n+1)$  prendiamo un generico  $n > 0$

Assumiamo  $P(n) \triangleq \forall \sigma. \sigma(x) = n \Rightarrow \langle w, \sigma \rangle \longrightarrow \sigma[0/x]$

Proviamo  $P(n+1) \triangleq \forall \sigma. \sigma(x) = n+1 \Rightarrow \langle w, \sigma \rangle \longrightarrow \sigma[0/x]$

consideriamo un  $\sigma$  generico e assumiamo  $\sigma(x) = n+1$

$\langle w, \sigma \rangle \longrightarrow \sigma'$

$\swarrow \langle x > 0, \sigma \rangle \longrightarrow \mathbf{tt}, \langle x := x - 1, \sigma \rangle \longrightarrow \sigma'', \langle w, \sigma'' \rangle \longrightarrow \sigma'$

$\swarrow^* \langle x := x - 1, \sigma \rangle \longrightarrow \sigma'', \langle w, \sigma'' \rangle \longrightarrow \sigma'$

$\swarrow^*_{\sigma'' = \sigma[n/x]} \langle w, \sigma[n/x] \rangle \longrightarrow \sigma'$

per ipotesi induttiva  $P(n)$  sappiamo che  $\langle w, \sigma \rangle \longrightarrow \sigma[0/x]$

# Equivalenza parametrica

$$w \stackrel{?}{\sim}_o ww \qquad w \stackrel{\Delta}{=} \text{while } b \text{ do } c$$
$$ww \stackrel{\Delta}{=} \text{while } b \text{ do } (\overbrace{\text{while } b \text{ do } c}^w)$$

Consideriamo un  $\sigma$  generico, se  $\langle b, \sigma \rangle \longrightarrow \mathbf{ff}$ :

$$\langle w, \sigma \rangle \longrightarrow \sigma'$$

$$\swarrow_{\sigma'=\sigma} \langle b, \sigma \rangle \longrightarrow \mathbf{ff}$$

$$\swarrow^* \square$$

$$\langle w, \sigma \rangle \longrightarrow \sigma$$

$$\langle ww, \sigma \rangle \longrightarrow \sigma'$$

$$\swarrow_{\sigma'=\sigma} \langle b, \sigma \rangle \longrightarrow \mathbf{ff}$$

$$\swarrow^* \square$$

$$\langle ww, \sigma \rangle \longrightarrow \sigma$$

# Esempio(cont.)

$$w \stackrel{?}{\sim}_o ww \qquad w \triangleq \text{while } b \text{ do } c$$

$$ww \triangleq \text{while } b \text{ do } (\overbrace{\text{while } b \text{ do } c}^w)$$

Consideriamo un  $\sigma$  generico,  $\langle b, \sigma \rangle \longrightarrow \mathbf{tt}$ : se  $\langle w, \sigma \rangle \longrightarrow \sigma'$ :

$$\begin{array}{l} \langle ww, \sigma \rangle \longrightarrow \sigma'_1 \\ \swarrow \langle b, \sigma \rangle \longrightarrow \mathbf{tt}, \langle w, \sigma \rangle \longrightarrow \sigma'', \langle ww, \sigma'' \rangle \longrightarrow \sigma'_1 \\ \swarrow^* \langle w, \sigma \rangle \longrightarrow \sigma'', \langle ww, \sigma'' \rangle \longrightarrow \sigma'_1 \\ \swarrow^*_{\sigma'' = \sigma'} \langle ww, \sigma' \rangle \longrightarrow \sigma'_1 \\ \swarrow_{\sigma'_1 = \sigma'} \langle b, \sigma' \rangle \longrightarrow \mathbf{ff} \\ \swarrow^* \square \qquad \qquad \qquad \langle ww, \sigma \rangle \longrightarrow \sigma' \end{array}$$

**esercizio per casa**

$$\forall b, c, \sigma, \sigma'. \langle \text{while } b \text{ do } c, \sigma \rangle \longrightarrow \sigma' \Rightarrow \langle b, \sigma' \rangle \longrightarrow \mathbf{ff}$$

# Esempio (con.)

$$w \stackrel{?}{\sim}_o ww \quad w \triangleq \text{while } b \text{ do } c$$
$$ww \triangleq \text{while } b \text{ do } (\overbrace{\text{while } b \text{ do } c}^w)$$

Consideriamo un  $\sigma$  generico, se  $\langle b, \sigma \rangle \longrightarrow \mathbf{tt}$ : se  $\langle w, \sigma \rangle \not\rightarrow$ :

$$\langle ww, \sigma \rangle \longrightarrow \sigma'_1$$

$$\swarrow \langle b, \sigma \rangle \longrightarrow \mathbf{tt}, \langle w, \sigma \rangle \longrightarrow \sigma'', \langle ww, \sigma'' \rangle \longrightarrow \sigma'_1$$


$$\swarrow^* \langle w, \sigma \rangle \longrightarrow \sigma'', \langle ww, \sigma'' \rangle \longrightarrow \sigma'_1$$


divergera', perche' per ipotesi  $\langle w, \sigma \rangle \not\rightarrow$


$$\langle ww, \sigma \rangle \not\rightarrow$$

# Una nota sulla divergenza

due comandi qualsiasi che divergono sono sempre equivalenti

$$w_1 \stackrel{?}{\sim}_o w_2$$


$$w_1 \stackrel{?}{\sim}_o w_3$$


$$w_1 \stackrel{?}{\sim}_o w_4$$


$w_1 \triangleq$  **while true do skip**

$w_2 \triangleq$   **$x := 0$ ; while  $x \geq 0$  do  $x := x + 1$**

$w_3 \triangleq$   **$y := 0$ ; while  $y \leq 0$  do  $y := y - 100$**

$w_4 \triangleq$   **$y := x + 1$ ; while  $x \neq y$  do ( $x := x - 1$ ;  $y := y + 1$ )**

# Un problema difficile

$w_1 \triangleq$  while true do skip

$p \triangleq$   $x := 3;$

$s := 1;$

**while**  $x \neq s$  **do** (

$x := x + 2;$

$s := 1;$

$i := 2;$

**while**  $2 \times i \leq x$  **do** (

**if**  $x \% i = 0$  **then**  $s := s + i$

**else skip;**

$i := i + 1$

)

)

$w_1 \stackrel{?}{\sim}_o p$

# Un problema difficile

$w_1 \triangleq$  while true do skip

$w_1 \stackrel{?}{\sim}_o p$

termina sempre

quando  $x \geq 2$ , alla fine,  $s$  prende la somma di tutti i divisori propri di  $x$

```
 $s := 1;$   
 $i := 2;$   
while  $2 \times i \leq x$  do (  
    if  $x \% i = 0$  then  $s := s + i$   
    else skip;  
     $i := i + 1$   
)
```

# Un problema difficile

$w_1 \triangleq$  while true do skip

$w_1 \stackrel{?}{\sim}_o p$

$$s := \sum_{i \in \text{div}(x)} i$$

where  $\text{div}(x) \triangleq \{1\} \cup \{d \mid 1 < d < x, x \% d = 0\}$

$$i := 1 + (x/2)$$



# Un problema difficile

$w_1 \triangleq$  while true do skip

$p \triangleq$   $x := 3;$  un numero dispari  $w_1 \stackrel{?}{\sim}_o p$

$s := 1;$  somma di divisori propri di  $x$

while  $x \neq s$  do ( esce quando  $x$  e' perfetto

$x := x + 2;$  prossimo numero dispari

$s := \sum_{i \in \text{div}(x)} i$  somma di divisori propri di  $x$

where  $\text{div}(x) \triangleq \{1\} \cup \{d \mid 1 < d < x, x \% d = 0\}$

$i := 1 + (x/2)$

)

un numero e' perfetto

se e' la somma dei suoi divisori propri

$$6 = 1 + 2 + 3$$

# Un problema difficile

$w_1 \triangleq$  while true do skip

$p \triangleq$

termina quando viene trovato il primo **numero dispari perfetto**

$w_1 \stackrel{?}{\sim}_o p$



fino al 2021

non si sa se un numero dispari perfetto esiste

Ochem, Pascal; Rao, Michaël.

Odd perfect numbers are greater than  $10^{1500}$ .

Mathematics of Computation n.81(279): 1869–1877.

(2012) doi:10.1090/S0025-5718-2012-02563-4

un numero e' **perfetto**

se e' la somma dei suoi divisori propri

# Per le menti curiose

Euclide ha provato che  $2^{p-1}(2^p - 1)$  è un numero perfetto pari quando  $2^p - 1$  è primo

Eulero ha provato che ogni numero perfetto pari ha la forma  $2^{p-1}(2^p - 1)$  con  $2^p - 1$  primo

Numero primi della forma  $(2^p - 1)$  sono chiamati numeri primi di Mersenne

Numero pari perfetti e numeri primi di Mersenne sono in corrispondenza in 1 a 1

Perchè  $2^p - 1$  sia primo è necessario che  $p$  primo

Se  $p$  è primo,  $2^p - 1$  non è necessariamente primo

Fino al 2021, sono stati trovati solo 51 numeri primi di Mersenne

Non si sa se ce ne sono un numero infinito

# Per le menti curiose

prime number $p$	Mersenne prime $2^p - 1$	even perfect number $2^{p-1}(2^p - 1)$
2	3	6
3	7	28
5	31	496
7	127	8.128
13	8.191	33.550.336
17	131.071	8.589.869.056
...	...	...
82.589.933	24.862.048 digits!	49.724.095 digits!

# Un problema difficile

$w_1 \triangleq$  while true do skip

$p_1 \triangleq$   $x := 2;$

$s := 1;$

**while**  $x \neq s$  **do** (

$x := x + 2;$

$s := 1;$

$i := 2;$

**while**  $2 \times i \leq x$  **do** (

**if**  $x \% i = 0$  **then**  $s := s + i$

**else skip;**

$i := i + 1$

)

)

$w_1 \stackrel{?}{\sim}_o p_1$



$x := 6 \stackrel{?}{\sim}_o p_1$



$x := 6;$

$s := 6; \stackrel{?}{\sim}_o p_1$

$i := 4;$



# Una variante

$w_1 \triangleq$  while true do skip

$p_2 \triangleq$   $s := 1;$

**while**  $x \neq s$  **do** (

$x := x + 2;$

$s := 1;$

$i := 2;$

**while**  $2 \times i \leq x$  **do** (

**if**  $x \% i = 0$  **then**  $s := s + i$

**else skip;**

$i := i + 1$

)

)

$w_1 \overset{?}{\sim}_o p_2$




$x := 6;$

$s := 6;$   $\overset{?}{\sim}_o p_2$

$i := 4;$



# Un'ultima istanza

$$p_2 \stackrel{?}{\sim}_o p_3$$


$p_2 \triangleq$  `s := 1;`  
`while x ≠ s do (`  
    `x := x + 2;`  
    `s := 1;`  
    `i := 2;`  
    `while 2 × i ≤ x do (`  
        `if x % i = 0 then s := s + i`  
            `else skip;`  
        `i := i + 1`  
    `)`  
`)`

$p_3 \triangleq$  `s := 1;`  
`while x ≠ s do (`  
    `x := x + 1;`  
    `s := 1;`  
    `i := 2;`  
    `while 2 × i ≤ x do (`  
        `if x % i = 0 then s := s + i`  
            `else skip;`  
        `i := i + 1`  
    `)`  
`)`