Frame

funzione totale da

A a $B_\perp$

$$B_\perp = B \cup \{\perp\}$$

$$f : A \to B_\perp$$

$$\omega : A \to B_\perp$$

frame vuoto

$$\omega(x) = \perp$$

Aggiornamento di un frame

$$f : A \rightarrow B_{\perp}$$

$$\underbrace{\left(f\left[{}^{b}/_{a}\right]^{mod} \right)}(p) = \begin{cases} b & \text{se } p = a \\ f(p) & \text{altrimenti} \end{cases}$$

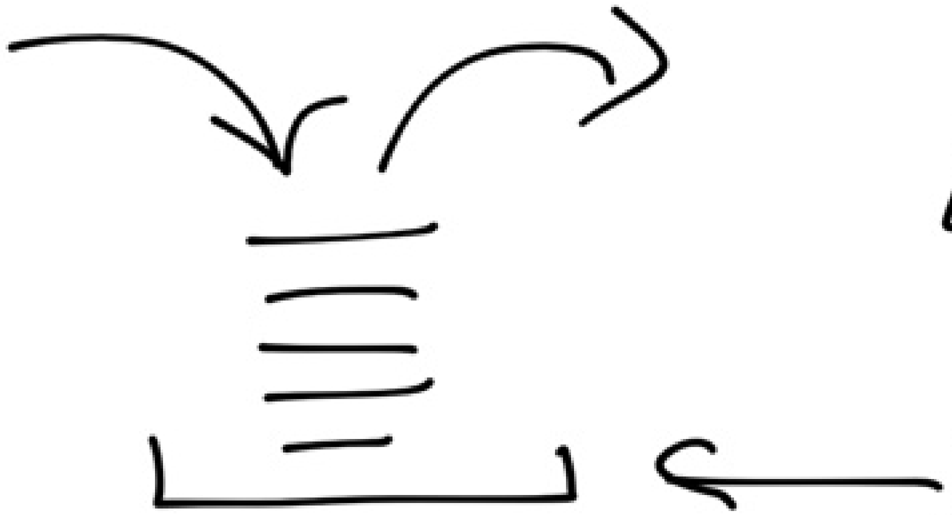
se $f(a) \neq \perp$

$$f : A \to B_\perp$$

$$\underbrace{\left(f\left[{}^{b}\!/_{a}\right]^{add}\right)}(p) = \begin{cases} b & \text{se } p = a \\ \underbrace{f(p)} & \text{altrimenti} \end{cases}$$

$$\neq$$

$$\text{se } \quad f(a) = \perp$$



LIFO

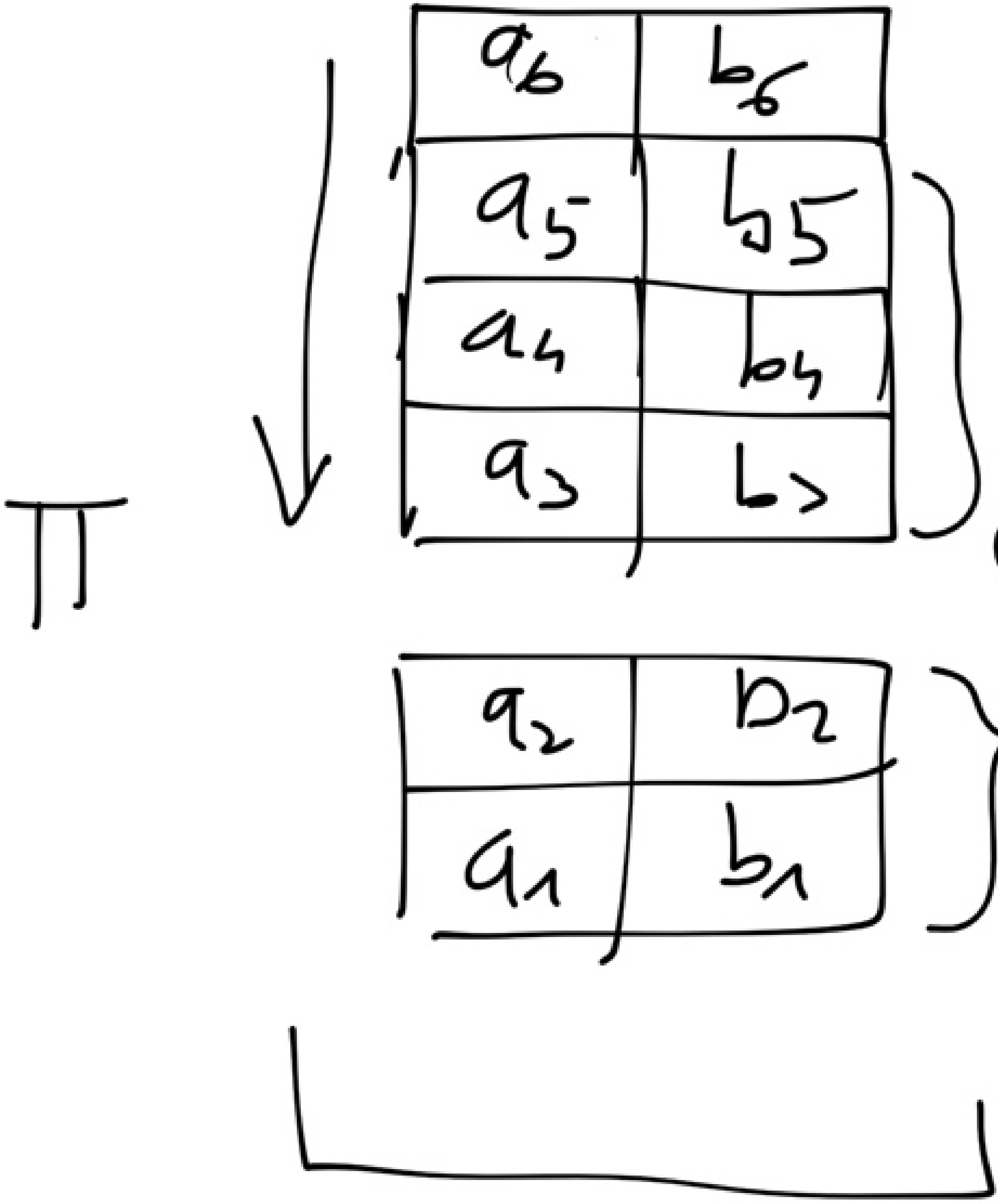
Pile

PILE DI FRAME

$$\Pi = \{\Omega\} \cup \{ f . \pi \mid f : A \to B_\perp \wedge \pi \in \Pi \}$$

insieme di tutte le pile di funzioni da A a $B_\perp$

Ω pile vuote



Π

a_6	b_6
a_5	b_5
a_4	b_4
a_3	b_3

a_2	b_2
a_1	b_1

$a_i \in A$

$b_i \in B$

$f_2 : A \to B_\perp$

$f_1 : A \to B_\perp$

$\Pi(a_4)$ _

$$\Pi(a) = \begin{cases} \bot & \text{se } \Pi = \Omega \\ f(a) & \text{se } \Pi = f\cdot\Pi' \\ & \text{e } f(a) \neq \bot \\ \Pi'(a) & \text{se } \Pi = f\cdot\Pi' \\ & \text{e } f(a) = \bot \end{cases}$$

$a \in A$

$$\Pi\left[{}^{b}/_{a}\right]^{mod} = \begin{cases} f.\Pi'\left[{}^{b}/_{a}\right]^{mod} & \text{se } \Pi = f.\Pi' \text{ e } f(a) = \perp \\ f\left[{}^{b}/_{a}\right]^{mod}.\Pi' & \text{se } \Pi = f.\Pi' \text{ e } f(a) \neq \perp \end{cases}$$

$$\pi\left[{}^{b}\!/\!{}_{a}\right]^{add} = f\left[{}^{b}\!/\!{}_{a}\right]^{add} \cdot \pi'$$

non vuote

(se $\pi = f \cdot \pi'$

e $f(a) = \perp$ ← a non compare in f

$$B_\bot = B \cup \{\bot\}$$

```
type 'a bottom = Bottom
        | Def of 'a ;;
```

```
Def 5 ;;
- : int bottom
    = Def 5
```

```
Bottom ;;
- : 'a bottom = Bottom
```

$f : {'a} \to {'b}\ \text{bottom}$

$\boxed{f[{}^{b}/_{a}]^{add}}$ se $f(a) = \bot$

l'operazione di add ha tre argomenti

$f : {'a} \to {'b}\ \text{bottom}$

$a : {'a}$ $\quad$ $b : {'b}\ \text{bottom}$

il risultato di add ha tipo $\underline{{'a} \to {'b}\ \text{bottom}}$

```
let add f a b = match f a with
    Bottom ->
      let g p = if p = a then b
                else f p
    in g ;;
```

$$\to \text{'a} \to \text{'b bottom}$$

$$\text{add} : \underbrace{(\text{'a} \to \text{'b bottom})}_{f} \to \underbrace{\text{'a}}_{a} \to \underbrace{\text{'b bottom}}_{b} \to \underbrace{\qquad}_{ms}$$

$$\left(f\left[{}^{b}/_{a}\right]^{mod}\right)(p) = \begin{cases} b & \text{se } p = a \\ f(p) & \text{altrimenti} \end{cases} \Bigg|\ \text{se } f(a) \neq \bot$$

```
let update f a b = match f a with
 Def _ -> let g p =
      if p = a then b else f p
  in g ;;
update : ('a -> 'b bottom) -> 'a -> 'b bottom
  -> 'a -> 'b bottom = <fun>
```

'b'	40
'a'	~~30~~ 20
'c'	10

```
let f x = if x = 'a' then Def 30
      else if x = 'b' then Def 40
            else Bottom ;;
f : char -> int bottom = <fun>
```

```
f 'a' ;;
- : int bottom = Def 30

f 'b' ;;
- : int bottom = Def 40

f 'c' ;;
- : int bottom = Bottom
```

```
let g = add f 'c' (Def 10) ;;
g : char -> int bottom = <fun>

g 'a' ;;
- : int bottom = Def 30

g 'c' ;;
- : int bottom = Def 10
```

```
let f1 = update g 'a' (Def 10) ;;
f1 : char -> int bottom = <fun>

f1 'c' ;;
- : int bottom = Def 10

f1 'a' ;;
- : int bottom = Def 10
```

$$\Pi = \{\underset{\substack{\downarrow\\ [\,]}}{\Omega}\} \cup \{\underbrace{f.\pi}_{\substack{\downarrow\\ f::\pi}} \mid f : A \to B_\bot \wedge \pi \in \Pi\}$$

```
let rec search pi a = match pi with
   [] -> Bottom
| f::pi1 when f a <> Bottom -> f a
| f:: pi1 when f a = Bottom ->
                 search pi1 a
```

search : ('a -> 'b bottom) list -> 'a -> 'b bottom

$$\Pi(a) = \begin{cases} \perp & \text{se } \Pi = \Omega \\ f(a) & \text{se } \Pi = f.\Pi' \text{ e } f(a) \neq \perp \\ \Pi'(a) & \text{se } \Pi = f.\Pi' \text{ e } f(a) = \perp \end{cases}$$

$$\Pi\left[{}^{b}/_{a}\right]^{add} = f\left[{}^{b}/_{a}\right]^{add} . \Pi'$$

se $\Pi = f . \Pi'$ e $f(a) = \bot$

```
let add_stack pi a b = match pi with
    f :: pi1 when f a = Bottom
    -> (add f a b) :: pi1

add_stack : ('a -> 'b bottom) list ->
    'a -> 'b bottom -> ('a -> 'b bottom) list
```

$$\Pi\left[{}^{b}/_{a}\right]^{mod} = \begin{cases} f \cdot \Pi'\left[{}^{b}/_{a}\right]^{mod} & \text{se } \Pi = f \cdot \Pi' \\ & \text{e } f(a) = \bot \\ f\left[{}^{b}/_{a}\right]^{mod} \cdot \Pi' & \text{se } \Pi = f \cdot \Pi' \\ & \text{e } f(a) \neq \bot \end{cases}$$

```
let rec update_stack pi a b =
        match pi with

  f::pi1 when f a = Bottom
        -> f::(update_stack pi1 a b)
  f::pi1 when f a <> Bottom
    -> (update f a b)::pi1
```

ambiente

frame di ambiente

$$\varphi : Ide \to Loc_{\perp}$$

nomi delle variabili del programma C

indirizzi di memoria dove trovano il valore delle variabili

insieme delle pile di ambiente

$$\Phi = \{ \varphi \mid \varphi : Ide \to Loc_{\perp} \}$$

$$P = \{ \Omega \} \cup \{ \varphi . p \mid \varphi \in \Phi \wedge p \in P \}$$

memoria

$\nu : Loc \to Val_\perp$ frame di memoria

$N = \{\nu \mid \nu : Loc \to Val_\perp\}$

insieme di tutti i frame di memoria

$M = \{\Omega\} \cup \{\nu . \mu \mid \nu \in N \wedge \mu \in M\}$

insieme di tutte le pile di frame di memoria

STATO

t	l_5
z	l_4
x	l_3
w	l_2

y	l_1
x	l_0

ambiente

l_5	15
l_4	7
l_3	11
l_2	10

l_1	-2
l_0	5

memoria

2013_11_19

Sintassi del linguaggio C

Dichiarazioni (creano lo stato)

Espressioni (calcolano valori)

Comandi (modificano lo stato)

Exp → Num | Exp Op Exp | ....
Ide | ...

Dec → Type Ide = Exp; |
Type Ide ;

Com → Ide = Exp; |
if (Exp) Com else Com |
if (Exp) Com |

while (Exp) Com |
Block

Block → { Declist Comlist }

DecList → ε | Dec Declist

Comlist → ε | Com ComList