Predicati, confronti ed operatori logici

Nome Uso	$True$ se:
TT: identicamente vero $TT : x$	$\forall x$
FF: identicamente vero $FF : x$	$\neg\exists x$
ISSEQ: sequenza? $isseq : s$	$s \in Seq$
ISFUN: funzione? $isfun : f$	$f \in Fun$
ISPAIR: coppia, sequenza di lunghezza due? $ispair : x$	$x =< x_1, x_2 >$
ISSTRING: stringa? $isstring : s$	$s \in string$
ISBOOL: booleano? $isbool : x$	$x \in boolean$
ISNUM: numero? $isnum : x$	$x \in numbers$
ISINT: intero (lunghezza arbitraria)? $isint : x$	$x \in bignums$
ISREAL: reale (virgola mobile)? $isreal : x$	$x \in floatingpoint$
EQ, NEQ: uguaglianza e diseguaglianza. $eq :< x_1, x_2 >$ $neq :< x_1, x_2 >$	 $x_1 = x_2$, identità per funzioni $x_1 \neq x_2$ uguaglianza per il resto

Tipo	Nomi	Uso	Significato
confronto curried	lt,gt,le,ge	$p : n_1 : n_2$	$<, >, \leq, \geq$
confronto su sequenze	less,greater lesseq,greatereq	$p :< n_1, ..., n_n >$	Sequenze in ordine
Confronto con 0	isnumpos,isintpos,... isnumneg,isintneg,...	$p : n$	Numeri > 0 Numeri < 0

Sintassi	Semantica
$and :< x_1, ..., x_n >$	False se $\exists x_i = False \mid <>$
$or :< x_1, ..., x_n >$	False se $\forall x_i = False \mid <>$
$not : x$	True se $x = False \mid <>$

$isseqof : p :< x_1, \ldots, x_n >$	True se $\forall x_i\, p : x_i = True$	$isseqof = AND \sim AA$

Primitive aritmentiche

Tipo	Nome	Versione Raised
Su liste	add,sub,mul,div	+,-,*,/
Binarie	** (elevamento potenza)	
Unarie	sub	

Nome	Valore
intmin ,intmax	intero inferiore e superiore
abs, sign	valore assoluto e segno
exp,ln	e^x e logaritmo
sin,cos, tan,... asin,acos,atan	Funzioni trigonometriche e inverse
sqrt	Radice quadrata

Primitive su sequenze

Nome e descrizione Uso	Risultato
AL: Append Left $al :< x_1, < x_2, \ldots, x_n >>$	$\equiv < x_1, \ldots, x_n >$
AR: Append Right $ar :<< x_1, \ldots, x_{n-1} >, x_n >$	$\equiv < x_1, \ldots, x_n >$
SEL: Seleziona elemento $sel : i :< x_1, \ldots, x_n >$ $si :< x_1, \ldots, x_n >$	$\equiv x_i$ se $i \leq n$
LEN: Lunghezza $len :< x_1, \ldots, x_n >$	$\equiv n$
LIST: Crea Sequenza $list : x$	$\equiv < x >$
FIRST: Primo elemento $first :< x_1, ..., x_n >$	$\equiv x_1$
LAST: Ultimo elemento $last :< x_1, ..., x_n >$	$\equiv x_n$
TAIL: Coda della sequenza $tail :< x_1, ..., x_n >$	$\equiv < x_2, \ldots, x_n >$
TAILR: Coda di destra $tailr :< x_1, ..., x_n >$	$\equiv < x_1, ..., x_{n-1} >$
REVERSE: Rigira sequenza $reverse :< x_1, ..., x_n >$	$\equiv < x_n, \ldots, x_1 >$
CAT: Concatenazione di sequenze $cat :<< x_{1,1}, \ldots, x_{1,n_1} >,$ $\ldots < x_{n,1}, \ldots, x_{n,n_n} >>$	$\equiv < x_{1,1}, \ldots, x_{1,n_1},$ $\ldots, x_{n,1}, \ldots, x_{n,n_n} >$
DISTL: Distribuisci elemento a sinistra $distl :< x, < y_1, \ldots, y_n >>$	$\equiv << x, y_1 >, \ldots, < x, y_n >>$

DISTR: Distribuisci elemento a destra	
$distr :<< x_1, \ldots, x_n >, y >$	$\equiv << x_1, y >, \ldots, < x_n, y >>$

TRANS: Trasponi sequenza di sequenze.	
$trans :<< x_{1,1}, \ldots, x_{1,m} >,$	$\equiv << x_{1,1}, \ldots, x_{n,1} >,$
$\ldots < x_{n,1}, \ldots, x_{n,m} >>$	$\ldots < x_{1,m}, \ldots, x_{n,m} >>$

INTSTO: Intervallo di interi partendo da 1.	
$intsto : i$	$\equiv < 1, ..., i >$

FROMTO: Intervallo interi	
$fromto :< a, b >$	$\equiv < a, ..., b >$se $a < b$
$a..b$	$< b, ..., a >$se $b > a$
	$< a >$se $a = b$

#,##: Ripeti sequenza.	
$\# : n : x$	$\equiv < x_1, ..., x_n >$con $x_1 = ... = x_n = x$
$\#\# : n : x$	$\equiv cat : (\# : n : x)$

Funzionali

Nome e descrizione	
Uso	Risultato:

ID : Funzione Identità	
$ID : x$	$\equiv x$

K : Costante all'applicazione	
$K : x : y$	$\equiv x$

Curry : Funzione 2° ordine -> funzione 2-aria	
$C : f : x : y$	$\equiv f :< x, y >$

APPLY : Applica funzione ad un argomento	
$APPLY :< f, x >$	$\equiv f : x$

COMP : Composizione funzioni	
$COMP :< f_1, \ldots, f_n >: x$	$\equiv f_1 : (f_2 : (\ldots (f_n : x) \ldots)$
$\sim:< f_1, \ldots, f_n >: x$	
$(f_1 \sim f_2 \sim \ldots \sim f_n) : x$	

CONS : Costruisci funzione vettoriale.	
$CONS :< f_1, \ldots, f_n >: x$	$\equiv< f_1 : x, \ldots, f_n : x >$
$[f_1, \ldots, f_n] : x$	

IF : Scelta su predicato:	
$IF :< p, f, g >: x$	$\equiv f : x$ se $p : x = True$
	$\equiv g : x$ se $p : x = False$

AA : Applica ad ogni elemento:	
$AA : f :< x_1, \ldots, x_n >$	$\equiv< f : x_1, \ldots, f : x_n >$

LIFT : Solleva funzione n-aria da valori a funzioni	
$LIFT : f :< g_1, \ldots g_n >$	$\equiv f \sim [g_1, \ldots, g_n]$
$LIFT : f :< g_1, \ldots g_n >: x$	$\equiv f :< g_1 : x, \ldots, g_n : x >$

RAISE : Solleva se applicata a funzioni	
$RAISE : f$	$\equiv IF :< IsSeqOf : IsFun, lift : f, f >$
$RAISE : f :< g_1, \ldots g_n >$	$\equiv \begin{cases} lift : f :< g_1, \ldots g_n >= \\ \quad f \sim [g_1, \ldots, g_n] \text{ se } g_i \in funs \\ f :< g_1, ..., g_n > \text{ altrimenti} \end{cases}$
INSL : Distribuisci funzione a sinistra	
$INSL : f :< x_1, \ldots, x_n >$	$\equiv f :< insl : f :< x_1, \ldots, x_{n-1} >, x_n >$
$INSL : f :< x >$	$\equiv x$
INSR : Distribuisci funzione a destra	
$INSR : f :< x_1, \ldots, x_n >$	$\equiv f :< x_1, insr : f :< x_2, \ldots, x_n >>$
$INSR : f :< x >$	$\equiv x$
TREE : Distribuisci funzione ad albero	
$TREE : f :< x_1, \ldots, x_n >$	$\equiv f :< tree : f :< x_1, \ldots, x_k >,$
	$\quad tree : f :< x_k, \ldots, x_n >>$
	ove $k = intmin : (n/2)$
$TREE : f :< x >$	$\equiv x$
MERGE : Unisci due sequenze in ordine secondo un predicato	
$MERGE : f :< a, b >$	$\equiv x$ se $y =<>$
con $a =< a_1, \ldots >$	$\equiv y$ se $x =<>$
$b =< b_1, \ldots >$	$\equiv al :< a_1, merge : f :< tail : x, y >>$
	se $f :< a_1, b_1 >= TRUE$
	$\equiv al :< b_1, merge : f :< x, tail : y >>$
	se $f :< a_1, b_1 >= FALSE$
CASE : Scelta su predicati	
$CASE :<< p_1, f_1 >,$	$\equiv IF :< p_1, f_1, if :< \ldots$
$\ldots, < p_n, f_n >>$	$\quad if :< p_n, f_n, SIGNAL : \ldots >$

Eccezioni

$$signal : x = exception(x)$$

$$catch :< f, g >: x = \begin{cases} g :< x, v > & \text{se } y = exception(v) \\ y & altrimenti \end{cases} \quad \text{ove } y = f : x$$

Primitive su geometria

Costruttori

- $CUBOID :< l_1, \ldots, l_n >$

```
DEF Cuboid (sides::IsSeqOf:IsRealPol)
```

Un paralelepipedo di dimensione n e dai lati di misura l_i.

- $SIMPLEX : n$

```
DEF SIMPLEX (n::IsInt)
```

  Il simplesso standard di ordine n immerso in $\Re^n$

- $CYLINDER :< r, h, n >$

```
DEF CYLINDER (r,h::IsRealPos;n::IsIntPos)
```

  Un cilindro di raggio r ed altezza h approssimato da un cilindroide con poligono di base (sulle prime due coordinate) di n lati.

- $QUOTE :< x, \ldots, x_n >$

```
DEF QUOTE (x::IsSeqOf:IsReal)
```

  Costruisce il complesso poliedrale di dimensione (1,1) costituito da segmenti di lunghezza x_i se maggiore di zero, intervallati da spazi di lunghezza sempre x_i se minore di zero.

- $MKPOL :<< p_1, \ldots, p_n >, < c_1, \ldots, c_m >, < pol_1, \ldots, pol_l >>$

```
DEF MKPOL (points::IsSeqOf:(IsSeqOf:IsReal);
       cells,pols::IsSeqOf:(IsSeqOf:IsIntPos))
```

  Costruisce il complesso poliedrale composto dai poliedri descritti da pol_k, lista di liste di indici alle celle c_j, a loro volta lista di liste di indici ai punti p_i dello spazio euclideo n dimensionale.

Distruttore.

- $UKPOL : pol$

```
DEF UKPOL (pol::ISpol)
```

  Restituisce la descrizione dei parametri della $MKPOL$ del complesso poliedrale equivalente a *pol* ma costituito da un semplice complesso poliedrale di poliedri. $MKPOL\tilde{}UKPOL$ effettivamente appiattisce un complesso poliedrale qualunque.

Osservatori, Predicati

- $IsPol : pol$

```
DEF IsPol (p::TT)
```

  Pol è un complesso poliedrale.

- $ISPOLDIM :< d, n >: pol$

```
DEF ISPOLDIM (d,n::IsIntPos)
```

Pol è un complesso poliedrale di dimensione intrinseca d immerso (dimensione estrinseca) nello spazio Euclideo n dimensionale.

- DIM : *pol*

  ```
  DEF DIM (pol::IsPol)
  ```

  Restituisce la dimensione intrinseca del complesso poliedrale *pol*.

- RN : *pol*

  ```
  DEF RN (pol::IsPol)
  ```

  Restituisce la dimensione dello spazio d'immersione (dimensione estrinseca) del complesso poliedrale *pol*.

- $MAX \,|\, MIN \,|\, MED$: n : *pol*

  ```
  DEF MAX|MIN|MED (n::IsInt) (pol::IsPol)
  ```

  Rispettivamente Massima, Minima, Media coordinata n del complesso poliedrale *pol*.

- BOX : *pol*

  ```
  DEF BOX (pol::IsPol)
  ```

  Restituisce il 'bounding box' del complesso *pol*.

Costruttori da altri Complessi Poliedrali.

- $STRUCT :< pol_1 \,|\, fun_1 \,,\, ... \,,\, pol_n \,|\, fun_n >$

  ```
  DEF STRUCT (elems::IsSeqOf:(OR~[IsPol,IsFun]))
  ```

  Restituisce il complesso poliedrale costruito dai pol_j a cui sono applicate tutte le funzioni $fun_i \,, i < j$ precedenti. Nella implementazione Scheme non è necessario che le fun_i siano trasformazioni affini.

- $TOP \,|\, BOTTOM \,|\, LEFT \,|\, RIGHT \;:< pol_1, pol_2 >$

  ```
  DEF TOP|BOTTOM|LEFT|RIGHT (pol1,pol2::IsPol)
  ```

  Pone pol_2 rispettivamente sopra, sotto, a sinistra e a destra di pol_1.

Trasformazioni Affini:

- MAT :$<< x_{1,1}, \ldots, x_{1,n} >, \ldots, < x_{n,1}, \ldots, x_{m,n} >>$: pol

  $OMAT$:$<< x_{1,1}, \ldots, x_{1,n} >, \ldots, < x_{n,1}, \ldots, x_{m,n} >>$: pol

  ```
  DEF MAT  (IsSeqOf:(IsSeqOf:IsReal))
  DEF OMAT (IsSeqOf:(IsSeqOf:IsReal))
  ```

  Trasformazione affine e di embedding applicata a *pol* descritta da una matrice, rispettivamente OMAT e MAT se in coordinate omogenee o normali.

- $T : i : x : pol \mid T :< i_1, \ldots, i_n >:< x_1, \ldots, x_n >: pol$

  $S : i : x : pol \mid S :< i_1, \ldots, i_n >:< x_1, \ldots, x_n >: pol$

  ```
  DEF T (i::OR~[IsIntPos,IsSeqOf:IsIntPos])
          (x::OR~[IsReal,IsSeqOf:IsReal])
          (pol::IsPol)
  DEF S (i::OR~[IsIntPos,IsSeqOf:IsIntPos])
          (x::OR~[IsReal,IsSeqOf:IsReal])
          (pol::IsPol)
  ```

  Traslazione e scalamento di *pol* sulla coordinata i (coordinate i_i) della quantità x (delle quantità x_i).

- $R :< i, j >: x : pol$

  $R :<< i, j >_1, \ldots, < i, j >_n >:< x_1, \ldots x_n >: pol$

  ```
  DEF R (plane::OR~[IsPairOf:IsIntPos,
                IsSeqOf:IsPairOf:IsIntPos])
        (x::OR~[IsReal,IsSeqOf:IsReal)
        (pol::IsPol)
  ```

  Rotazione di *pol* della quantità $x_{(i)}$ rispetto al piano $< i, j >_{(i)}$.

- H

  Shearing

Geometria parametrica.

- $Map : f : pol$

  ```
  DEF MAP (f::IsFun) (pol::IsPol)
  ```

  - *pol* è un complesso poliedrale di dimensioni (d_1, n_1).
  - f è una funzione che applicata ad una sequenza di numeri $< x_1, \ldots, x_{n_1} >$ restituisce un'altra sequenza $< y_1, \ldots, y_{n_2} >$, con $n_2 \geq n_1$.

Le sequenze $< x_1, \ldots, x_{n_1} >$ e $< y_1, \ldots, y_{n_2} >$ rappresentano le coordinate degli spazi Euclidei, rispettivamente ε^{n_1} e ε^{n_2}. La f rappresenta quindi un morfismo tra questi due spazi. Il complesso risultante avrà dimensione (d_1, n_2) e sarà il *pol* originario con lo spazio di supporto trasformato. In genere questa funzione è utilizzata per la costruzione di geometrie curve parametriche.

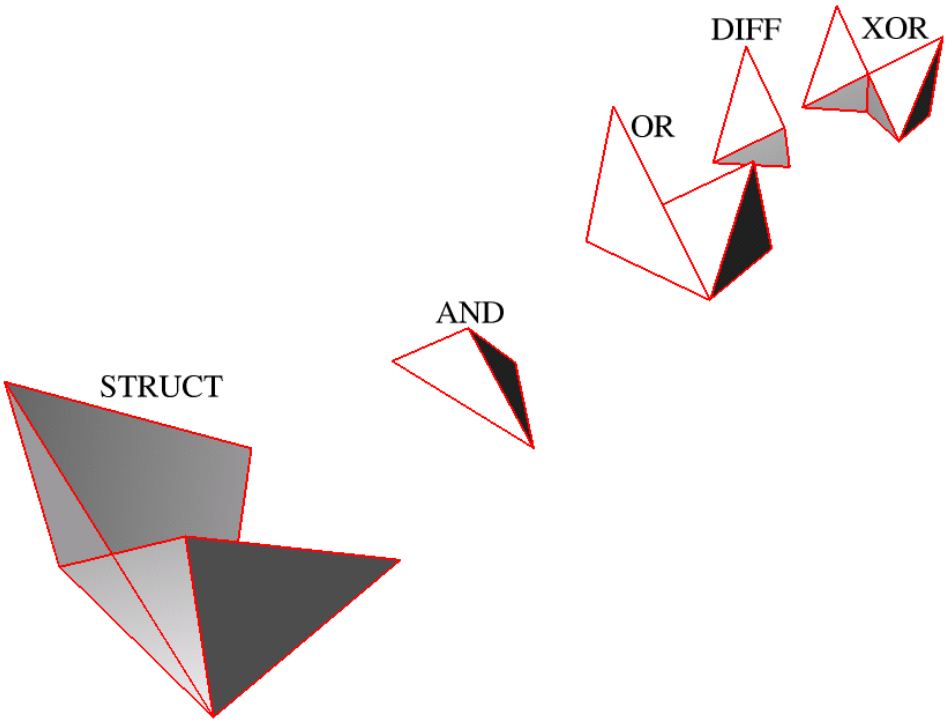
Operazioni Booleane

Si applicano ad una coppia di complessi $bool :< pol_1, pol_2 >$ e ne restituiscono l'operazione booleana (insiemistica) corrispondente.

- AND, INTERSECTION, &
- OR, UNION,|
- DIFF, DIFFERENCE,-
- XOR, ||,SDIFFERENCE

Esempio:

```
def test_bool=
    STRUCT~[STRUCT,K:(T:2:2),&,
                   K:(T:2:2),|,
                   K:(T:2:2),-,
                   K:(T:2:2),||];
test_bool:<(t:1:-1:(simplex:3)),
            (S:1:-1:(simplex:3))>;
```



Altre Operazioni:

Queste operazioni sono specifiche della rappresentazione HPC:

- @: $i : pol$

  $@i : pol$ (forma contratta)

  Operazione di estrazione dello $i - esimo$ scheletro (confine di dimensione i) dal complesso pol. Se le dimensioni di pol sono (d, n) il risultato sarà di dimensioni (i, n).

- $POWER | * :< pol_1, pol_2 >$

  Moltiplicazione di complessi poliedrali pol_1di dimensione n e pol_2di dimensione m. Il risultato sarà un complesso poliedrale di dimensione $n + m$

- &&:$< pol_1, pol_n >$

  Estrusione delle intersezioni.