

A.A. 2021/22

FONDAMENTI DI  
ANALISI E VERIFICA DEL SOFTWARE

PROF. ISABELLA MASTROENI

FABS :)

## NOTA

**Questi appunti/sbobinatura/versione “discorsiva” delle slides sono per mia utilità personale,**  
quindi pur avendole revisionate potrebbero essere ancora presenti typos, commenti/aggiunte personali (che  
anzi, lascio di proposito) e nel caso peggiore qualche inesattezza!

Comunque spero siano utili! 

Appunti e puttanate sono scritte *principalmente* in questo stile.

**Questo file fa parte della mia collezione di sbobinature,  
che è disponibile (e modificabile!) insieme ad altre in questa repo:  
<https://github.com/fabfabretti/sbobinamento-seriale-uniVR>**

## INDICE

NOTA .....	2
Indice.....	3
Interpretazione astratta .....	5
Interpretazione astratta: Costruzione formale dei domini.....	9
Interpretazione astratta: Proprietà delle computazioni astratte.....	16
Interpretazione astratta: Computazioni astratte.....	19
Analisi non distributiva.....	26
Analisi non distributiva: Propagazione delle costanti .....	27
Analisi non distributiva: Analisi degli intervalli.....	32
Analisi non distributiva: Widening (e narrowing).....	38
Altre analisi non distributive .....	41
Riassunto: interpretazione astratta.....	42



# INTERPRETAZIONE ASTRATTA

L16 – Interpretazione astratta 1: l’idea

## Introduzione

### Mini recap

Nelle lezioni precedenti abbiamo visto un approccio all’analisi statica che parte da ciò che vogliamo calcolare, ovvero dall’informazione che vogliamo osservare dall’esecuzione del nostro programma. In particolare, abbiamo visto che per proprietà relativamente semplici (=vicine alla sintassi, che non descrivono informazioni riguardanti il contenuto delle variabili), aka che riguardano la relazione fra elementi del programma che i contenuti.

Per questi tipi di analisi, abbiamo visto che riusciamo a costruire – fornendo direttamente l’equazione che vogliamo risolvere o in modo semantico, aka fornendo una semantica di elaborazione dell’informazione costruita in modo induttivo sulla sintassi – riusciamo a dare la soluzione ricercata.

### Una nuova soluzione...

Il problema si pone quando vogliamo guardare un po’ più a fondo, ovvero **siamo interessati a guardare dentro lo stato della macchina e i veri valori delle variabili**.

A questo punto, anche se riusciamo a fornire la semantica, questa semantica non è distributiva, e fa sì che ci sia necessariamente una perdita di informazione data dal fatto che analizzando localmente l’analisi sui punti di programma **perdiamo informazioni rispetto alla semantica desiderata, ovvero quella che unsice su tutti i cammini (MOP)**.

Questa parte dell’informazione è il prezzo aggiuntivo da pagare per avere un calcolo che si avvicina sempre di più all’informazione concreta.

È chiaro che l’approccio che descriviamo ora avrà bisogno di tutta una infrastruttura extra, dato che in confronto le analisi di data flow erano molto più semplici. Per poter delle garanzie nella costruzione della semantica astratta, ci servirà un’infrastruttura – che chiamiamo **interpretazione astratta** – che mi permette di formalizzare la relazione fra il mondo concreto, di cui vogliamo dire qualcosa, e il mondo astratto, che è il mondo approssimato su cui vogliamo dire qualcosa

### Def. Interpretazione astratta

**Framework formale** (=insieme di contenuti, risultati e concetti) **che permette di descrivere dettagliatamente la relazione tra il mondo concreto di cui vogliamo dire qualcosa e il mondo astratto approssimato su cui possiamo dire qualcosa.**

Questo dando garanzie di poter comunque dare informazioni certe (benché approssimate) su aspetti di interesse del mondo concreto.

L’obiettivo quindi è sempre dimostrare in modo automatico proprietà dei nostri programmi, e l’astrazione è il procedimento che usiamo per arginare il problema della non decidibilità della semantica concreta.

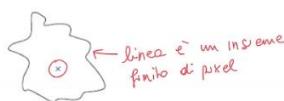
## Intuizione

Lo capiamo usando la grafica :)

### Oggetti

#### Def: Oggetto

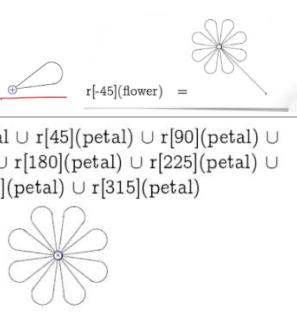
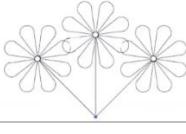
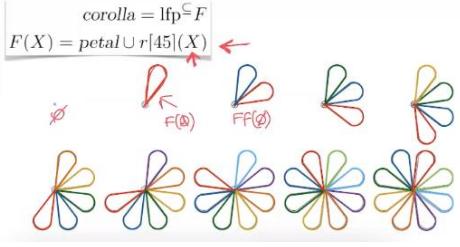
L’oggetto è una coppia: **un’origine** (punto di riferimento x) e un **insieme finito di pixels neri** (su uno sfondo bianco).



Supponiamo quindi di definire nel dominio concreto un **oggetto fiore**, che possiamo ottenere attraverso una serie di operazioni concrete a partire da degli oggetti concreti.

## Operazioni

Introduciamo le operazioni che ci servono per il fiore:

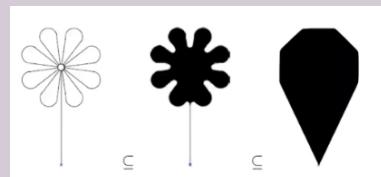
<i>Costanti</i>	Oggetto base che, per combinazione, ci permette di ottenere l'oggetto finale. Ad esempio <i>Petal</i> .	$\text{petal} = \underline{\text{ }}$
<i>Rotazione</i>	$r[\alpha](o)$ di un oggetto $o$ di un certo angolo $\alpha$ attorno alla sua origine	$r[45](\text{petal}) = \underline{\text{ }} \quad r[-45](\text{flower}) = \underline{\text{ }}$
<i>Unione</i>	$o_1 \cup o_2$ degli oggetti $o_1$ e $o_2$ è l'unione dei due oggetti sovrapponendoli sull'origine.	$\text{corolla} = \text{petal} \cup r[45](\text{petal}) \cup r[90](\text{petal}) \cup r[135](\text{petal}) \cup r[180](\text{petal}) \cup r[225](\text{petal}) \cup r[270](\text{petal}) \cup r[315](\text{petal})$
	Posso unirla alla rotazione per ottenere una <i>corolla</i> :	
<i>Stem</i>	$\text{stem}(o)$ è un'operazione che aggiunge il gambo ad un oggetto, attaccandosi all'origine e mettendo l'origine sull'inizio del gambo.	$\text{flower} = \text{stem}(\text{corolla})$
<i>Boquet</i>	È l'insieme delle operazioni $\text{boquet} = r[-45](\text{flower}) \cup \text{flower} \cup r[45](\text{flower})$	
<i>Corolla per punto fisso</i>	La corolla può essere calcolata per punto fisso, ovvero possiamo trovare un operatore monotono che calcolato iterativamente:	$\text{corolla} = \text{lfp}^{\subseteq} F$
	Prende l'oggetto a cui viene applicato, lo ruota di 45 gradi e aggiunge un petalo.	$F(X) = \text{petal} \cup r[45](X)$
	Ci accorgiamo del punto fisso perché il posto dove dovremmo mettere il petalo è già occupato.	

## Approssimazioni

Ora vogliamo approssimare questi oggetti concreti.

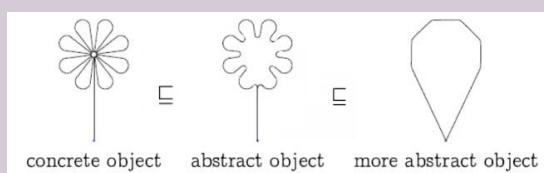
### Upper approximation

L'approssimazione superiore di un oggetto è un oggetto con la stessa origine ma più pixels.



### Def. Oggetto astratto

È una rappresentazione matematica o al computer dell'approssimazione di un oggetto concreto. Per esempio, in questo caso decidiamo di rappresentare l'insieme di pixel con il suo contorno.



Un oggetto è più astratto di un altro se corrisponde a un oggetto concreto più largo.

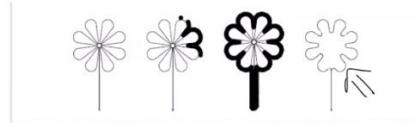
### Def. Dominio astratto

Un dominio astratto è un insieme di oggetti astratti più operazioni astratte che approssimano quelle concrete

### Def. Astrazione

Una funzione di astrazione  $\alpha$  mappa ciascun oggetto concreto ad un'approssimazione, rappresentata da un oggetto astratto  $\alpha(o)$ .

L'idea è di disegnare un contorno con una penna di una certa dimensione; cambiando la dimensione cambia l'astrazione. Consideriamo solo il contorno esterno degli oggetti; più largo è il contorno, più è astratta la visione.



### Concretizzazione

Una funzione di concretizzazione  $\gamma$  mappa oggetti astratti  $\bar{o}$  nell'oggetto concreto  $\gamma(\bar{o})$ , che rappresenta, ovvero che è il suo significato concreto.

Corrisponde ad assegnare il significato concreto dell'oggetto astratto che stiamo osservando. L'idea è di riempire il contorno, in modo da prendere tutte quelle figure che stanno dentro quel contorno.



L'idea è che lì dentro avrei potuto avere anche altre linee concrete anziché i petali; comunque sarebbe stato contenuto in questa concretizzazione, perché hanno la stessa astrazione.

### Def. Comparare astrazioni

Più si perde informazione più si è astratti (diametro della penna più largo),

Le forme diverse di astrazione non sono comparabili (uso penne diverse)

Le funzioni di astrazioni e concretizzazioni sono modellate dal concetto di connessione di Galois.

### Connessione di Galois

La connessione di galois è una coppia di funzioni  $\alpha, \gamma$  tali che:

1. $\alpha$ e $\gamma$ sono monotone, ovvero conservano l'ordine	<p>- <math>\alpha</math> is monotonic.</p> <p>implies</p>
2. Il processo può aggiungere rumore, quindi se prendo un fiore, ne faccio un contorno (astraggo) e lo riempio (concretizzo) ottengo qualcosa che contiene il fiore originale.	<p>- <math>\gamma</math> is monotonic.</p> <p>implies</p> <p>- for all concrete objects <math>x</math>, <math>\gamma \circ \alpha(x) \supseteq x</math>.</p> <p>flower      <math>\alpha(\text{flower})</math>      <math>\gamma(\alpha(\text{flower}))</math></p>

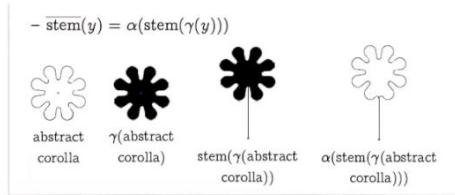
<p>3. Se noi prima concretizziamo un oggetto astratto e poi lo astraiamo c'è un'identità;</p>	<p>- for all abstract objects <math>y</math>, <math>\alpha \circ \gamma(y) \sqsubseteq y</math>.</p>	
<p>4. L'ordinamento astratto è definito come trasferimento dell'ordinamento concreto nel mondo astratto.</p>	<p>- <math>x \sqsubseteq y</math> is defined as <math>\gamma(x) \sqsubseteq \gamma(y)</math>.</p>	

## Trasferimento delle operazioni

Dato il dominio concreto e astratto, e data l'associazione fra i due (approssimazione), possiamo astrarre tutti gli altri oggetti.

### Operazioni semplici

È necessario definire delle operazioni ad hoc che lavorino solo su oggetti astratti preservandone le informazioni. Lo faccio su tutte le operazioni:



### Unione

<p><b>Non basta unire gli oggetti astratti!</b> Ci sarebbe un punto in cui perderemmo del contorno, e otterremmo qualcosa che <b>non ha l'invariante dell'avere come invariante una penna con un certo spessore</b>. Quindi devo <b>concretizzare ogni elemento, unisco gli oggetti concreti e a questo punto ne disegno il contorno</b>.</p>	<p><math>x \sqcup y = \alpha(\gamma(x) \sqcup \gamma(y))</math></p>
---	---

### Abstract fix-point

Esattamente come le operazioni di stem e di unione, anche le operazioni di punto fisso possono essere trasferite dal mondo concreto al mondo astratto.

$$\text{abstract corolla} = \alpha(\text{concrete corolla}) = \alpha(\text{lfp}^{\sqsubseteq} F)$$

where  $F(X) = \text{petal} \cup r[45](X)$

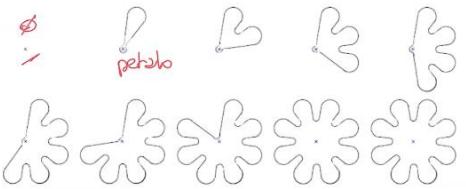
$$\begin{aligned} - \alpha(F(X)) \\ &= \alpha(\text{petal} \cup r[45](X)) \\ &= \alpha(\text{petal}) \sqcup \alpha(r[45](X)) \\ &= \alpha(\text{petal}) \sqcup \bar{r}[45](\alpha(X)) \\ &= \text{abstract petal} \sqcup \bar{r}[45](\alpha(X)) \\ &= \bar{F}(\alpha(X)) \end{aligned}$$

by defining

$$\boxed{\bar{F}(X) = \text{abstract petal} \sqcup \bar{r}[45](X)}$$

and so:

$$-\text{abstract corolla} = \alpha(\text{concrete corolla}) = \alpha(\text{lfp}^{\sqsubseteq} F) = \text{lfp}^{\sqsubseteq} \bar{F}$$



La situazione di aver costruito sul nostro dominio l'operatore di punto fisso che riesce precisamente a raggiungere il punto fisso concreto è la situazione ideale; in generale invece succede che il punto fisso astratto sarà un'approssimazione ulteriore del punto fisso concreto.

Sull'esempio: Questa è sempre una situazione ideale detta completezza del dominio; in generale, potrei aver raggiunto un'approssimazione della corolla astratta.

## INTERPRETAZIONE ASTRATTA: COSTRUZIONE FORMALE DEI DOMINI

### L17 - Interpretazione astratta 2 - Costruzione formale

Vediamo gli **strumenti formali** che usiamo per modellare e descrivere le basi dell'interpretazione astratta.

L'oggetto principale usato per descrivere il legame fra mondo concreto e mondo astratto sono le **connessioni di Galois**. Non è l'unico (ce ne sono 4), ma 3 di questi sono equivalenti.

### Inserzioni di Galois

Galois è il più intuitivo.



#### Def. Connessione di Galois

Una connessione di Galois è una coppia di funzioni  $\alpha$ ,  $\gamma$  definite tra un dominio concreto  $C$  e un dominio astratto  $A$ , dove:

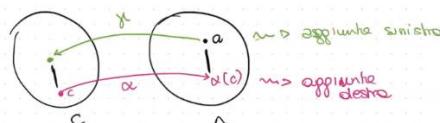
- $GC : \alpha, \gamma$  tra un dominio concreto  $C$  e un dominio astratto  $A$ , dove per semplicità consideriamo  $A, C$  reticolli completi ( $\leq_A, \leq_C$ )
- $\alpha : C \rightarrow A$  funzione di astrazione
- $\gamma : A \rightarrow C$  funzione di concretizzazione.

E vale anche che:

- $\alpha, \gamma$  sono monotone, ovvero  $\forall x, y \in C | x \leq_C y \Rightarrow \alpha(x) \leq_A \alpha(y) \wedge \forall x, y \in A | x \leq_A y \Rightarrow \gamma(x) \leq_C \gamma(y)$
- $\alpha(c) \leq_A a \Leftrightarrow c \leq_C \gamma(a)$   
(questa condizione garantisce l'esistenza della migliore approssimazione possibile per ogni elemento concreto)

Per semplicità, consideriamo una situazione in cui  $A$  e  $C$  sono reticolli completi.

Graficamente:



(ovvero dire che l'astrazione di un elemento è più piccolo di un elemento astratto vuol dire che la concretizzazione di quell'oggetto astratto è più grande dell'oggetto concreto da cui siamo partiti; ovvero  $\alpha$  e  $\gamma$  non alterano la relazione di ordine).

In generale,  $\alpha$  è detta anche aggiunta destra e  $\gamma$  è detta anche aggiunta sinistra.

$$\text{Notazione : } (C, \leq_C) \xrightarrow{\alpha} (A, \leq_A) \quad \xleftarrow{\gamma}$$

Questo:

$$\alpha(c) \leq_A a \Leftrightarrow c \leq_C \gamma(a)$$

È equivalente a dire che

$$c \leq_C \gamma(\alpha(c))$$

(ovvero se astraggo e poi concretizzo, tendenzialmente ho perso informazione e ho aggiunto rumore).

E vale anche che:

$$\alpha \circ \chi(\alpha) \leq_{\Delta} \alpha$$

Ovvero vale l'uguaglianza; questo è meno intuitivo ma mo ci spiega..

Ho più elementi che vanno sullo stesso, quindi poi tornando in dietro "vado verso il basso" (nn o cpt)

### Inserzioni di Galois

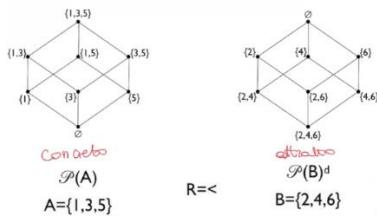
Oltre alle connessioni di Galois, esistono anche le Inserzioni di Galois: la differenza è che **non hanno nel dominio astratto A nessun "elemento inutile"**, cioè ogni elemento astratto ha uno specifico significato concreto da osservare.

In generale possiamo parlare di trasformazioni di Galois perché **ogni connessione di Galois può essere ridotta ad una inserzione di Galois** (eliminando gli elementi astratti inutili, ovvero quelli per cui esiste un elemento più preciso (=più piccolo) che ha lo stesso significato).

### Esempio

Non entriamo nel merito delle definizioni formali perché andremmo ad appesantire inutilmente l'intuizione. **Meno male**. L'idea è quella di vedere cosa vuol dire, partendo da un aconnesione di Galois che non è inserzione (aka ho degli elementi inutili), ridurla a un'inserzione.

Supponiamo di avere i seguenti due domini:



*Notiamo che sono ordinati al contrario (in concreto il più piccolo è bottom, in astratto il più piccolo è tutto)*

Ora costruiamo una **funzione di astrazione e concretizzazione** che permettano di vedere l'insieme a destra come un'astrazione del dominio a sinistra.

Prendiamo come astrazione la seguente:

$$\forall A' \subseteq A. \alpha(A') = \{b \in B \mid A' < b\}$$

Ovvero, intuitivamente, mappa un insieme nell'insieme più grande di elementi in cui tutti gli elementi sono più grandi di quelli da cui stiamo partendo.

	<p>Per esempio, partendo da <math>\{1\}</math>, prendiamo più grande il sottoinsieme di <math>\{2, 4, 6\}</math> che ha solo elementi più grandi... quindi <math>\{2, 4, 6\}</math>.</p>
	<p>Per gli altri elementi, il concetto è sempre lo stesso: partendo da <math>\{3\}</math>, dovrò prendere solo <math>\{4, 6\}</math> perché 2 è più piccolo.</p>
	<p>Tutti gli altri quattro elementi hanno solo 6, perché contengono 5.</p>

Già qui possiamo osservare che, di tutto il dominio astrato, **solo tre elementi** sono immagine di  $\alpha$ ; quindi ci viene il sospetto che siano inutili. Possiamo verificarlo andando a calcolare la  $\gamma$ , ovvero la funzione che preso un elemento astratto ne restituisce il significato concreto.

$$\forall B' \subseteq B. \gamma(B') = \{a \in A \mid a < B'\}$$

In questo caso, è l'insieme massimale di elementi che stanno sotto tutti gli elementi da cui sono partiti.

	<p>Se parto da <math>\{2,4,6\}</math>, l'unico insieme massimale che sta sotto a tutti e quindi va bene è <math>\{1\}</math>; questo è vero per qualunque insieme che contiene il 2. Quindi, secondo questa <math>\gamma</math>, <math>\{2,4,6\}</math>, <math>\{2,6\}</math>, <math>\{2,4\}</math>, <math>\{2\}</math> hanno tutti lo stesso significato (=elemento concreto).</p>
	<p>Similarmente per tutto ciò che contiene 4! Quindi ho un elemento astratto, <math>\{4\}</math>, che ha lo stesso significato concreto di un altro elemento astratto, <math>\{4,6\}</math>.</p>
	<p>Sia 6 che <math>\emptyset</math> hanno lo stesso significato.</p>

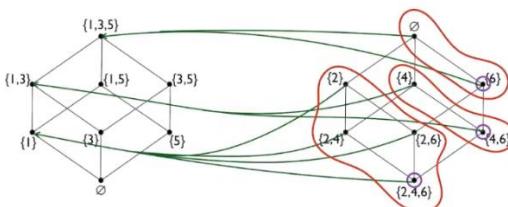
È chiaro che quello che mi interessa è solo **il più preciso di questo insieme equivalente**, perché quando astraiamo vogliamo una astrazione il più precisa possibile!

Osserviamo anche che la funzione, oltre ad essere monotona, **rispetta anche la condizione sull'essere connessione di Galois** (relazioni alfa gamma e gamma alfa).

- Se applico alfa su  $\emptyset$  vado in  $\{2,4,6\}$ ; tornando indietro vado in  $\{1\}$ , che è più in alto. Quindi  $\gamma \circ \alpha(\emptyset) \geq (\emptyset)$
- Se parto da  $\{2,6\}$  e applico  $\alpha \circ \gamma(\{2,6\})$  finisco in  $\{2,4,6\}$ , che è più in basso. Quindi  $\alpha \circ \gamma(\{2,6\}) \geq \{2,6\}$

Quindi, per riuscire ad arrivare alle inserzioni di Galois, vogliamo far sì che  **$\alpha \circ \gamma$  sia identità**. Semplicemente modificando il dominio astratto (e non l'astrazione) riusciamo ad arrivare a questo principio; in altre parole, teniamo invariate le funzioni di trasferimento ma eliminiamo gli elementi inutili.

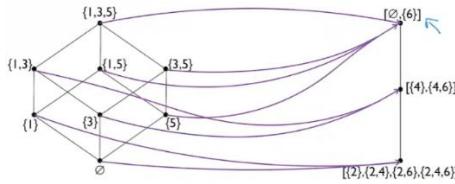
In questo caso osserviamo gli elementi che abbiamo visto avere lo stesso significato:



Quindi, due elementi vengono messi assieme se hanno la stessa funzione di concretizzazione.

$$B' = B'' \text{ iff } \gamma(B') = \gamma(B'') \text{ iff } \forall a \in A. aR B' \Leftrightarrow aR B''$$

Prendendo, anziché i veri elementi astratti, le classi di equivalenza... ottengo che SENZA CAMBIARE  $\alpha$  E  $\gamma$  HO OTTENUTO UN'INSERZIONE DI GALOIS (profe io le voglio strabene ma è la quinta volta che lo ripete)



## Altre formalizzazioni

Ora vediamo come questi due formalismi sono equivalenti ad altri formalismi che anziché guardare il dominio astratto guardano direttamente il significato sul dominio concreto.

Modello attraverso una funzione o un sottodomino l'astrazione del dominio concreto.

### Secondo modello: operatore di chiusura superiore (UCO)

È una tipologia di funzioni. Dato un dominio  $P$  (non specifichiamo concreto o astratto, so that's why  $P$ )

Dati un dominio  $P$  con un suo ordinamento  $\leq_P$ , una funzione  $\rho: P \rightarrow P$  è UCO se:

- $\rho$  è **monotona**, ovvero  $\forall xy \in P . x \leq_P y \Rightarrow \rho(x) \leq_P \rho(y)$
- $\rho$  è **estensiva**, ovvero  $\forall x \in P . x \leq_P \rho(x)$  (ovvero  $\rho$  va sempre verso l'alto, aka qualcosa di meno preciso)
- $\rho$  è **idempotente**, ovvero  $\forall x \in P . \rho(x) = \rho\rho(x)$ , aka può perdere precisione ma la perde tutta in un'unica applicazione; qualunque applicazione successiva non perde ulteriore informazione.

Verificheremo poi che  $\gamma \circ \alpha$  è una UCO nelle inserzioni di Galois; abbiamo già visto che è monotona ed estensiva, e si può dimostrare che è idempotente:

$$\rho\rho(x) = \gamma \circ \alpha \circ \rho \circ \gamma(c) = \gamma \circ \text{identità} \circ \alpha(c) = \gamma \circ \alpha(c)$$

Il passaggio a questo operatore è di permetterci di ignorare la rappresentazione astratta degli elementi concreti che abbiamo scelto di osservare con precisione, e considerare solamente la trasformazione che associa ad ogni elemento concreto la sua migliore approssimazione; ovvero che associa ad ogni elemento concreto la proprietà astratta di interesse che lo caratterizza.

### Moore family

È un particolare tipo di sottodomino.

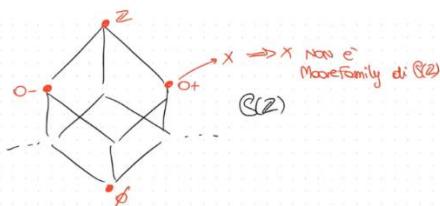
Partiamo da un dominio  $P$  e il suo ordinamento  $\leq_P$  reticolo completo; allora  $X \subseteq P$  è una Moore family di  $P$  (quindi è un concetto relativo al dominio concreto che lo contiene) se  $X = \mathcal{M}(X)$

con

$$\mathcal{M}(X) = \{\Lambda_P S \mid S \subseteq X\}, \text{ ovvero } X \text{ è chiuso per } \Lambda_P \text{ (chiusura di Moore)}$$

In altre parole, sia  $S \subseteq X \Rightarrow \Lambda_P S \in X$ . In altre parole ancora, se abbiamo  $S$  e  $\Lambda_S, \Lambda_S \in X$ . Cioè tutti gli and concreti di  $S$  sono in  $X$ .

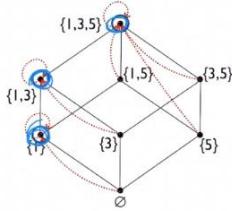
Avevamo inizialmente fatto questo esempio sui segni:



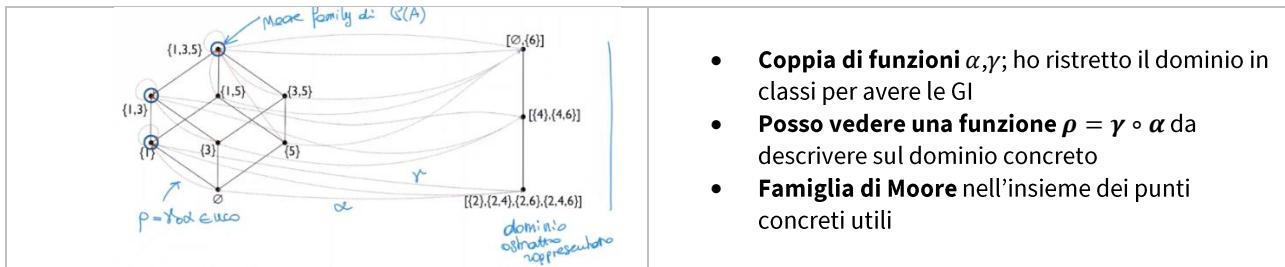
In questo caso, se prendo l'and dei due elementi  $0^+, 0^-$  ottengo come risultato  $0$ , che non appartiene a  $X$ . Aka, esiste  $S = \{0^+, 0^-\}$  tale che  $\Lambda_S = 0 \notin X$

La relazione con il dominio astratto sta nel fatto che l'insieme dei punti fissi di un UCO  $\rho: P \rightarrow P$  formano una Moore family di  $P$ , e ogni Moore family di un dominio ne rappresenta una proprietà.

Tornando all'esempio, notiamo che i punti "minimi" che avevamo trovato nell'esempio sono una Moore family del dominio concreto.



Mettendo tutti insieme, abbiamo tutti i modi fgraficamente:



Questi modi di rappresentare il dominio astratto coesistono e sono perfettamente equivalenti.

Ognuno può essere usato in contesti diversi:

- Galois **parla del dominio astratto**, e **dipende dalla rappresentazione degli elementi**; è utile quando dobbiamo **implementare**, perché il nostro obiettivo è che dobbiamo rappresentare nel modo più semplice le proprietà, e quindi ci fa comodo andare nel dominio astratto, dove le rappresentazioni sono semplificate.
- Al contrario, la Moore Family e la UCO sono **rappresentazioni indipendenti da come rappresento il dominio astratto**, ma sono **meno adatti a contesti applicativi** perché agiscono solo sugli oggetti concreti e l'oggetto concreto ha difficoltà di rappresentazione, ma sono spesso più utili nel contesto di verifica di proprietà del dominio astratto e sui ragionamenti formali sulle proprietà.

## Relazioni fra le rappresentazioni

GI vs famiglie di Moore

$$C \xrightleftharpoons[\alpha]{\gamma} A \quad \text{iff} \quad A \text{ è isomorfo a una famiglia di Moore di } C$$

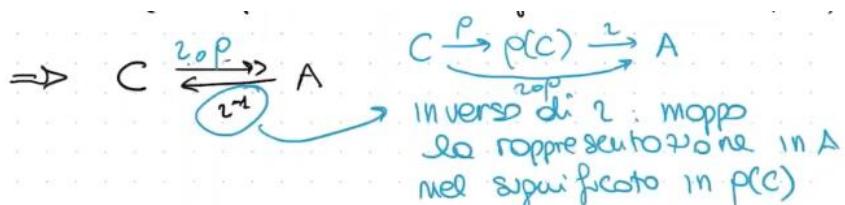
Ovvero  $\exists X$  famiglia di Moore di  $C$  tale che  $\exists \iota : A \rightarrow X$ , ovvero c'è una **rappresentazione biunivoca** tra gli elementi di  $A$  e gli elementi di  $X$ .

È quello che avviene nel nostro esempio: abbiamo tre elementi in catena che, in modo isomorfo, corrispondono alla Moore family.

UCO vs GI

Partendo da  $\rho \in UCO(C)$  possiamo costruire un GI:

$\exists$  isomorfismo tra  $\iota : \rho(C) \rightarrow A$  (con  $\iota$  isomorfismo,  $A$  rappresentazione degli elementi di  $\rho(C)$ )



Si può dimostrare che le due frecce sono un'astrazione, e quindi sono un'inserzione di Galois.

Si possono dimostrare le proprietà.... 53:02 TODO

$$\underbrace{z^{-1} \circ z \circ \rho}_{\text{id}} = \rho \quad \text{nel caso generale}$$

$$\rho \circ \underbrace{z \circ z^{-1}}_{\text{id}} (\alpha) = \alpha$$

## GI vs UCO

Dalle definizioni

$$C \xrightleftharpoons[\gamma]{\alpha} A \quad \forall \alpha \in \text{UCO}$$

## UCO vs Famiglie di Moore

Dalle definizioni

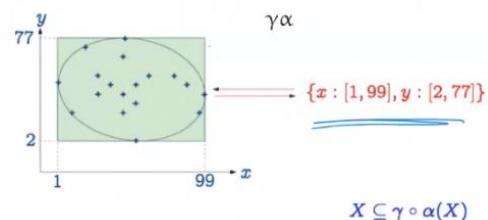
$\rho \in \text{UCO}(C) \Rightarrow \rho(C)$  è famiglia  
insieme di punti fissi di  $C$

## Esempio di dominio astratto: gli intervalli

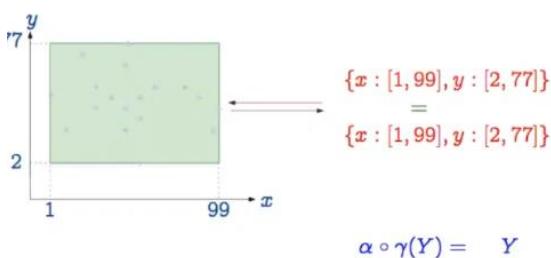
Vogliamo rappresentare valore minimo e massimo per ciascuna delle variabili.

Astrazione $\alpha$	Concretizzazione $\gamma$
$X \subseteq Y \Rightarrow \alpha(X) \subseteq \alpha(Y)$	$X \subseteq Y \Rightarrow \gamma(X) \subseteq \gamma(Y)$
<b>In questo caso l'astrazione prende gli intervalli; rappresento i punti di una variabile tramite due solo punti (i limiti).</b>	<b>L'astrazione è monotona perché se prendo un insieme più piccolo, anche i suoi limiti saranno più piccoli di quelli dell'insieme più grande.</b>

## Composizione



La composizione  $\gamma \circ \alpha$  è estensiva: se astraggo e poi riconcretizzo sicuramente ottengo un insieme (più grande) che contiene anche i punti da cui eravamo partiti



La composizione di  $\alpha \circ \gamma$  è l'identità: se parto da un insieme convesso che ha tutti i punti che stanno in un certo insieme di coordinate, ovviamente quando concretizzo ottengo lo stesso insieme.

## Reticolo delle interpretazioni astratte

Fino ad ora abbiamo detto che lo stesso concetto può essere rappresentato secondo più domini astratti.

Allo stesso modo, sullo stesso insieme di oggetti concreti possiamo osservare proprietà diverse. Quindi, di fatto, abbiamo un insieme di possibili astrazioni di un dominio concreto.

Vediamo le caratteristiche di questo insieme.

Supponiamo di avere

- $C$  dominio concreto e reticolo completo, con  $\langle C, \leq_p, \wedge, \vee, T, \perp \rangle$
- $A_i \in UCO(C)$  insieme dei possibili domini astratti.

Possiamo dimostrare che

$$\langle UCO(C), \sqsubseteq, \sqcap, \sqcup, \exists x.T, \forall x.x \rangle$$

*precisione relativa*      *e' un reticolo completo*

Vediamo tutte le cose.

$\sqsubseteq$  : Precisione relativa

**Confronta il grado di precisione dei domini astratti**, dove un elemento è più preciso se contiene più elementi.

$A_1 \sqsubseteq A_2$  (**è più preciso di**) se corrisponde a un insieme di elementi più grande.

Dal punto di vista delle chiusure,

$$\rho \sqsubseteq \eta \Leftrightarrow \forall y \in C . \rho(y) \leq \eta(y) \Leftrightarrow \rho(C) \supseteq \eta(C)$$

Ovvero se ho più elementi con i quali posso osservare con precisione il dominio concreto.

$\sqcap$  : Lower bound: prodotto ridotto

Con questo operatore relativo posso definire il prodotto ridotto  $\sqcap$ , dove  $\sqcap A_i$  è il più piccolo (astratto) fra tutti i domini che contengono tutti gli  $A_i$ .

Non è banalmente l'unione, perché l'unione di una Moore family può non essere una Moore family. In funzione delle Moore family possiamo anche esprimere come

$$\text{"}\mathcal{M} \sqcup_i A_i\text{"}$$

Non è super preciso perché in realtà potrebbero essere rappresentazioni diverse, ma vbb.

$\sqcup$  : Upper bound

È più facile: di fatto  $\sqcup A_i$  è il più grande (concreto) dominio astratto contenuto in tutti gli  $A_i$ .

Poiché l'intersezione di Moore family è una Moore family possiamo dire che  $\sqcap A_i = \sqcup A_i$

$\lambda x.T$  : Top

E' il dominio più astratto di tutti, quello che non osserva nulla; nessun elemento concreto è osservato con precisione. Tutti gli elementi concreti sono mappati nel top.

$\lambda x.x$  : Identità

Osserva tutto; ogni elemento concreto è mappato in sé stesso; è l'identità.

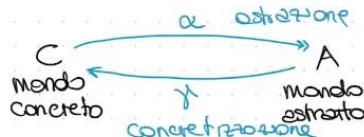
## INTERPRETAZIONE ASTRATTA: PROPRIETÀ DELLE COMPUTAZIONI ASTRATTE

### L18 - Interpretazione astratta 3 - Trasferimento del calcolo

Vediamo come si trasferisce un calcolo da un dominio concreto a uno astratto.

Per ora abbiamo parlato solo della costruzione dei domini astratti: parliamo da un mondo concreto in cui vogliamo dare risposte ma che sono non decidibili, in quanto nella semantica concreta ogni proprietà interessante non è decidibile.

Quindi, proviamo a guardare al programma attraverso il mondo astratto, nel quale possiamo costruire la **semantica e dare risposte in modo decidibile**. E' poco utile avere un mondo astratto arbitrario: vogliamo che sia collegato al mondo concreto attraverso il concetto di inserzione di Galois, ovvero di una coppia di funzioni che mi permettono di mettere in relazione gli elementi del mondo astratto nel mondo concreto.



Ora l'obiettivo è poter spostare anche il calcolo.

Partiamo da una funzione definita sugli oggetti concreti, così come nel fiore abbiamo trasferito unione e altre operazioni.

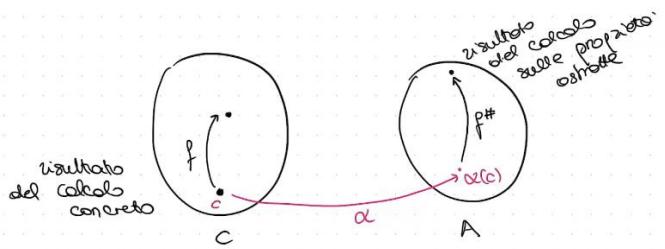
$$f : C \rightarrow C \text{ operazioni su elementi concreti che vogliamo trasferire sui corrispondenti elementi di } A.$$

$$f^{\#} : A \rightarrow A$$

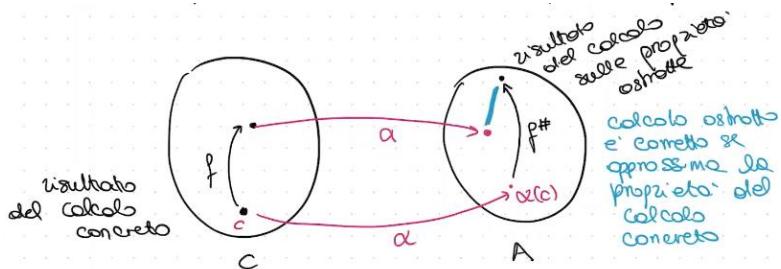
Correttezza (soundness)

**Non possiamo garantire che le due operazioni siano perfettamente collegate.** Definiamo una **relazione di correttezza** della funzione astratta ovvero diciamo che:

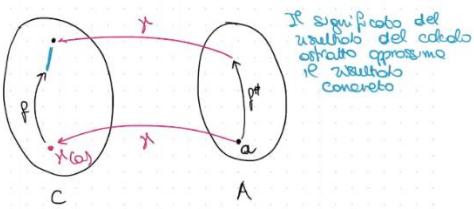
$f^{\#}$  è corretta per  $f$  se  $\alpha f(C) \leq_A f^{\#} \circ \alpha(C)$ .  
Graficamente:



Quello che vogliamo che succeda è che **il calcolo astratto è corretto se approssima la proprietà del calcolo concreto**.



Questa proprietà deriva dal confrontare i due calcoli nel mondo astratto, ma in modo analogo potrei caratterizzare la correttezza confrontare i due risultati sul mondo concreto.



In forma di formula:

$$f\gamma(a) \leq_c \gamma f^\wedge \#(a)$$

$$C \xrightarrow{\alpha} A, f: C \rightarrow C \Rightarrow f^* \text{ odd } \Leftrightarrow f_* \text{ odd } \quad (\text{if } f \text{ is odd})$$

$$\text{con} \quad \boxed{\text{KK}} \quad f \circ f(a) \leq_c f \circ f^*(c) \quad : \text{(*)} \quad \alpha \circ f(c) \leq_A f^* \circ \alpha(c)$$

In particolare possiamo osservare che se partiamo dalla condizione che:

$$\forall x \in C. \alpha f(x) \leq_n f^* \alpha(x)$$

*(ovvero per ogni elemento concreto vale la correttezza)*

Allora possiamo dimostrare che...

$$\alpha^p \gamma(x) \leq_{\sim} f^*(x)$$

$$\forall x \in C. \quad \alpha f(x) \leq_{\alpha} f^* \alpha(x) \iff \underbrace{\alpha f}_{\text{be } \alpha}(x) \leq_{\alpha} f^*(x)$$

Ovvero, una funzione è corretta solo se approssima la BCA, best correct approximation.

Tradotto,  $f^\#$  è sound se e solo se approssima  $f^\# = \alpha f \gamma : A \rightarrow A$

Questa costruzione è ok per approssimare le operazioni totali su  $C$ , ma non è accettabile come trasformazione dell'intera semantica, perché il nostro obiettivo è NON passare da  $f$  (perché  $f$  ha i problemi di decidibilità).

## Correttezza su chiusure

L'uso di questa funzione ci permette di osservare come la relazione di correttezza si può scrivere in termini di chiusure:

$\alpha f(x) \leq_A \alpha f \alpha(x) \rightarrow$  per monotonie di gamma posso applicare gamma da entrambi i lati  $\rightarrow \gamma \alpha f(x) \leq_C \gamma \alpha f \gamma \alpha(x)$

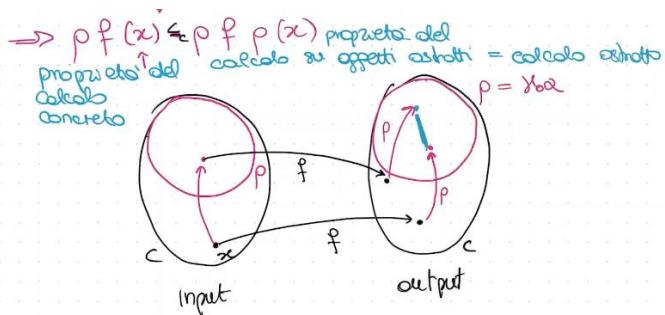
Definiamo  $\gamma\alpha$  come  $\rho \in UCO(C)$ , quindi possiamo riscrivere la relazione di correttezza come

$$\rho f(x) = \rho f\rho(x)$$

Graficamente, guardiamo i due domini C/A come input e output; all'interno abbiamo  $\rho = \gamma \circ \alpha$

Astratto

Applicando  $f$  all'oggetto in input approssimato avendoci applicato  $\rho$ , quello che succede è che ho un'approssimazione (in blu).



(qui confrontiamo il calcolo astratto col calcolo concreto)

Per estensività di  $\rho$ , che è UCO perché siamo in una GI, questa proprietà vale sempre: vale per monotonia di  $\rho f$  e per estensività di  $\rho$ , quindi ho sempre che  $x \leq \rho(x)$

### Concreto

Se anziché dal mondo astratto partiamo dal mondo concreto, allora

$$f^*\alpha(c) \leq_c \gamma \circ f^*(c) \Leftrightarrow f^*\alpha(c) \leq \gamma \circ f^*\gamma(c)$$

↪ bca    essendo  $\alpha$  una GI allora  
ogni elemento astratto è  
l'immagine di un elemento  
concreto  $\Rightarrow c = \alpha(c)$

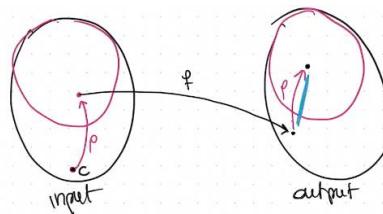
Sostituiamo  $\alpha(c)$ :

$$\Rightarrow f^*\alpha(c) \leq_c \gamma \circ f^*\alpha(c)$$

*per  $\alpha(c)$*

$$\Leftrightarrow f\rho(c) \leq_c \rho f\rho(c)$$

Graficamente...



*(qui confrontiamo la proprietà del calcolo astratto che approssima il calcolo astratto)*

Qui è più semplice: per estensività di  $\rho$  ho che  $\rho(f(c)) \geq f\rho(c)$

Costruire un'analisi statica nel framework dell'interpretazione astratta, ovvero mediante GI-UCO), garantisce al orrettezza del calcolo astratto.

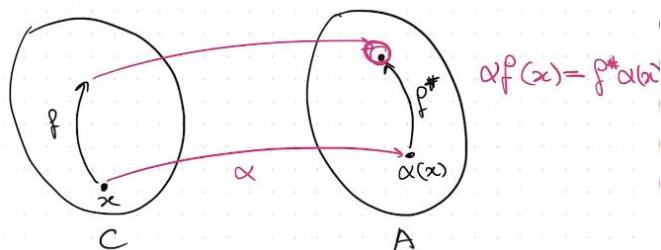
= ANALISI CORRETTA PER COSTRUZIONE!!

### Precisione

Quando possiamo dire che un'analisi è precisa relativamente a un calcolo?

Intuitivamente, l'analisi è precisa quando eseguire il calcolo nel mondo astratto o nel mondo concreto non fa alcuna differenza.

Un'analisi è precisa quando quello che non osserviamo non è rilevante per il calcolo. Sempre graficamente:



## INTERPRETAZIONE ASTRATTA: COMPUTAZIONI ASTRATTE

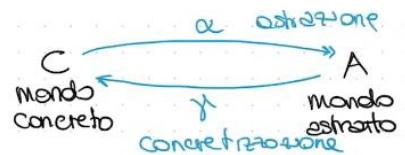
### L16 – Interpretazione astratta – Trasferimento del calcolo

Vediamo come si trasferisce un calcolo da un dominio concreto a uno astratto.

Nella semantica concreta, per Rice, il calcolo non è decidibile; quindi abbiamo scelto di guardare al nostro programma attraverso un mondo astratto, nel quale possiamo costruire la semantica e dare risposte in modo decidibile.

**È poco utile avere un mondo astratto costruito in mondo arbitrario;** vogliamo sia **collegato al mondo concreto** attraverso il concetto di **inserzione di Galois**, ovvero di una coppia di unzioni alpha, gamma di astrazione e concretizzazione che permettono di mettere in relazione gli elementi.

In particolare, **l'astrazione collega un oggetto concreto con la sua migliore approssimazione possibile**, e la concretizzazione associa a ogni elemento astratto il suo significato.



L'obiettivo è calcolare direttamente in astratto, ovvero spostare la computazione.

Partiamo quindi da una funzione definita sugli oggetti concreti. Le operazioni su elementi di C devono passare a essere su elementi di A corrispondenti.

Quindi, vogliamo costruire una funzione

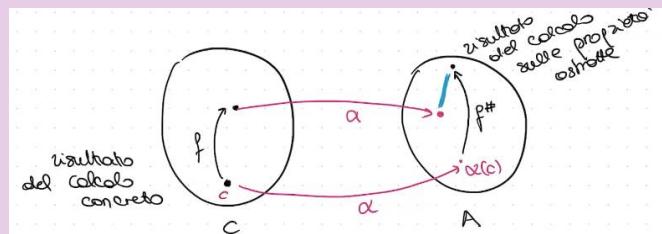
$$f^\# : A \rightarrow A$$

Per garantire che  $f^\#$  sia collegata dobbiamo **costruire una relazione**, ovvero **definiamo la relazione di correttezza** di una funzione astratta

*Def. Fuzione di correttezza di una funzione astratta*

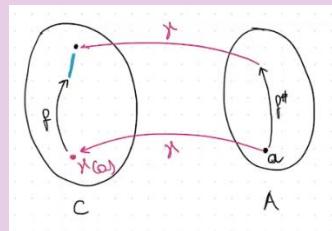
Diciamo che  $f^\#$  è corretta per  $f$  se  $\alpha \circ f(c) \leq_A f^\# \circ \alpha(c)$ , ovvero il calcolo astratto è corretto se approssima la proprietà del calcolo concreto.

Graficamente:



Questa rappresentazione è data confrontando i due calcoli nel mondo astratto. In modo del tutto analogo, possiamo confrontarli nel mondo concreto.

$$f\gamma(a) \leq_c \gamma f^\#(a)$$



Il significato del calcolo astratto approssima il risultato concreto.

Sono due forme perfettamente equivalenti, ovvero soddisfa la prima se e solo se soddisfa la seconda: date  $\alpha, \gamma$  GI e  $f : c \rightarrow c, \Rightarrow f^\#$  soddisfa la prima condizione sse soddisfa la seconda.

Questo vale per le connessioni di galois, non solo le inserzioni.

Se poi partiamo anche dalla condizione di correttezza, ovvero  $\forall x \in C. \alpha f(x) \leq_A f^\# \alpha(x)$ , allora possiamo dimostrare che è vero se e solo se  $\alpha f\gamma(x) \leq_A f^\#(x)$

Cioè una funzione è corretta se e solo se è più approssimata della BCA, best correct approximation  $\alpha f\gamma : A \rightarrow A$

Questa costruzione come BCA può andar bene per i singoli operatori,  
ma non per l'intera semantica perché l'obiettivo NON passare da  $f$ !

### Soundness su chiusure

Nel caso di un'inserzione di galois, possiamo scrivere la relazione di correttezza in termini di chiusure.

$$\alpha f(x) \leq_A \alpha f\gamma$$

Perché qualunque altra funzione approssimata è meno precisa della BCA, ovvero di  $\alpha f\gamma$ . Inoltre, ogni funzione che è più grande della BCA è sicuramente corretta, quindi è sufficiente ragionare sulla relazione di correttezza per la BCA. Quindi sostituiamo  $f^\#$  con la sua BCA.

A questo punto, per monotonia di  $\gamma$ , possiamo applicare  $\gamma$  da entrambe le parti

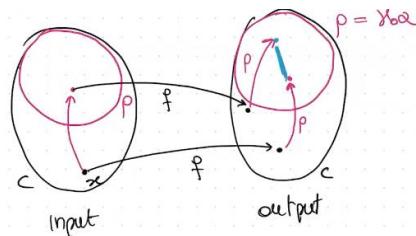
$$\gamma \alpha f(x) \leq_C \gamma \alpha f \gamma \alpha(x)$$

Possiamo ridefinire  $\gamma\alpha$  come  $\rho \in UCO$ , perché abbiamo già detto che  $\gamma\alpha$  è una UCO del dominio concreto. Di conseguenza possiamo riscrivere la relazione di correttezza come

$\rho f(x)$	$\leq_C$	$\rho f \rho(x)$
Proprietà del calcolo concreto		Proprietà del calcolo su oggetti astratti = calcolo astratto

Questa comunque vale sempre per monotonia di  $\rho$ , a patto che stiamo ragionando con una UCO; e questo è sempre vero se stiamo usando una GI.

Graficamente dobbiamo guardare un po' la prospettiva; anziché concreto e astratto guardiamo i due domini come input e output, passiamo fra i due domini con la funzione  $f$  e guardiamo i punti come i punti fissi della funzione di chiusura  $\rho = \gamma\alpha$ .



Quello che facciamo è partire da un punto concreto  $x$ , applicare al risultato  $\rho$  che sicuramente andrà in uno dei punti concerti di punto fisso. Questo è  $\rho(f(x))$ .

Dall'altra parte invece appliciamo  $\rho(x)$ , finendo quindi dentro  $\rho$ , poi appliciamo  $f$  e infine appliciamo  $\rho$ . Quello che avviene è che il fatto di aver applicato  $f$  su oggetti approssimati può comportare un'approssimazione.

Inoltre, se prendiamo

$$f\gamma(a) \leq_C \gamma \circ f^\#(a)$$

*(relazione di correttezza)*

Possiamo anche qui sostituire  $f^\#(a)$  con la BCA per vederlo in termini di chiusure:

$$f\gamma(a) \leq \gamma \alpha f\gamma(a)$$

Essendo  $\alpha\gamma$  una GI, allora ogni elemento astratto è l'astrazione di un elemento concreto; quindi sostituiamo  $a = \alpha(c)$

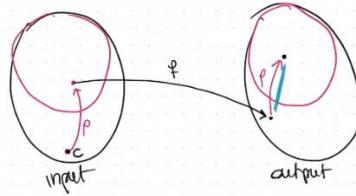
$$f\gamma\alpha(c) \leq_C \gamma \alpha f \gamma \alpha(c)$$

E nuovamente abbiamo  $\gamma\alpha$  è una  $\rho \in UCO(C)$ , quindi possiamo riscrivere esattamente

$$f\rho(c) \leq_c \rho f\rho(c)$$

$$\text{Per estensività di } \rho, \rho(f(\rho(c))) \geq f\rho(c)$$

La relazione è leggermente diversa, però ha esattamente lo stesso significato. Vediamo graficamente:



Qui confrontiamo la proprietà del calcolo astratto che approssima il calcolo astratto con il calcolo astratto.

$f\rho(c) \leq_c \rho f\rho(c)$  è sempre vera, perché è estensiva, quindi quella che davvero deve caratterizzare il calcolo è  $\rho f(x) \leq_c \rho f\rho(x)$ .

Costruire una analisi statica nel framework dell'interpretazione astratta, ovvero mediante GI-UCO, garantisce la correttezza per costruzione. :)

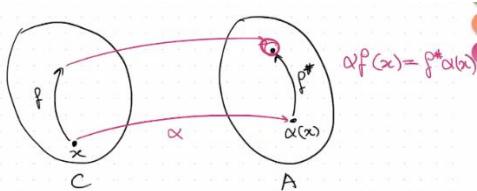
### Precisione

Quando è che diciamo che un'analisi è precisa?

Ovviamente, se andiamo a guardare la correttezza, l'analisi è precisa quando eseguire il calcolo nel mondo astratto o nel mondo concreto non fa alcuna differenza. Ovvero, eseguire il calcolo nel monod astratto è qualcosa che lavora sulle proprietà e non su valori concreti, però quello che succede è che questi due elementi vanno a coincidere; il calcolo astratto restituisce esattamente la proprietà del mondo concreto.

In altre parole, quell'informazione che abbiamo buttato via andando nel mondo astratto non era informazione utile per il calcolo che ci interessa effettuare.

Un'analisi è precisa quando quello che non osserviamo non è rilevante ai fini del calcolo.



Se sostituiamo  $f^{\#}$  con la BCA, come prima, otteniamo

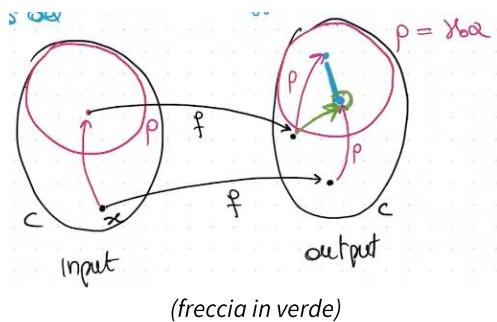
$$\alpha f(x) = \alpha f \gamma \alpha(x)$$

Possiamo applicare da entrambe e parti

$$\gamma \alpha f(x) = \gamma \alpha f \gamma \alpha(x)$$

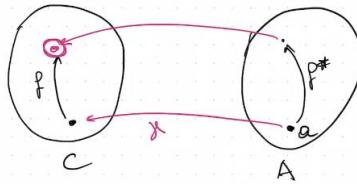
Ovvero, sostituendo  $\rho$

$$\rho f(x) = \rho f \rho(x)$$



Cioè la proprietà  $\rho$  del risultato ottenuto calcolando sugli elementi astratti è lo stesso del risultato ottenuto calcolando sugli elementi concreti.

Detto diversamente, l'astrazione dell'input non causa perdita di informazione nel calcolo.



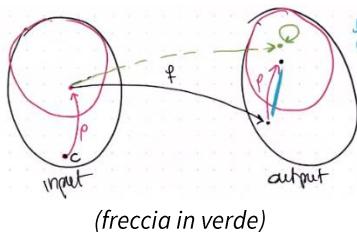
Il significato del calcolo astratto coincide col calcolo concreto.

Rifacendo le stesse operazioni che abbiamo fatto per la correttezza, otteniamo che

$$f^\# = \alpha f \gamma$$

$$f\rho(x) = \rho f \rho(x)$$

Osservare la proprietà del calcolo astratto non genera perdita di informazione.



## Esempi

### Backward non completo

Siamo sugli interi, e l'astratto sono i segni.

$$C = \mathcal{B}(\mathbb{Z}) \quad A = \text{Segni} \quad (\text{Pos}, \text{Neg}, \perp, \top)$$

Prendiamo come esempio di funzione la somma.

$$\underbrace{\text{Pos}}_{\text{f}} \text{ } \underbrace{\text{Pos}}_{\text{f}} \text{ } \underbrace{\text{Pos}}_{\text{p}} \Rightarrow \text{Pos} +^* \text{Neg} = \text{Non so}$$

abbiamo perso informazione

Se invece facciamo il calcolo completo abbiamo una risposta:

$$\underbrace{\text{Pos}}_{\text{f}} \Rightarrow \text{Segni}(5 + (-7)) = \text{Neg}$$

non BW completo

Il problema è nell'input: il fatto di aver calcolato la somma sui sogni ha fatto sì che non ci fosse abbastanza informazione; l'astrazione dell'input è quella che mi ha fatto perdere completamente l'informazione di cui la somma aveva bisogno per dare una risposta precisa; non c'è modo di cambiare l'osservazione i output perché la risposta diventi più precisa.

$$\underbrace{\text{Pos}}_{\text{f}} \text{ } \underbrace{\text{Pos}}_{\text{f}} \Rightarrow \text{Pos} +^* \text{Neg} = \text{Non so}$$

FW completo

(ho perso talmente tanta informazione che comunque non ho speranza di avere un significato; la perdita di precisione è dovuta a una precisione dell'input, e per questo si parla di backwards incompletezza).

### Forward incompletezza

$$C = \mathcal{B}(\mathbb{Z}) \quad A = \text{Costanti} \quad (\text{osserviamo se il valore} \quad \text{è una costante m})$$

$f = \square$   $e_1 \square e_2$  prende non det. un valore tra  
4 quello di  $e_1$  e quello di  $e_2$

$$p f p : \underline{p} (\underline{\text{const}_1 \square \text{const}_2}) = \text{non so}$$

$\downarrow$   
 $\text{const}_1, \text{const}_2$

Qui il problema è che  $\{\text{const}_1, \text{const}_2\}$  non fa parte delle parti di  $A$ , quindi non so.

$$p p : (\underline{\text{const}_1 \square \text{const}_2}) = \underline{\{\text{const}_1, \text{const}_2\}} \in \mathcal{B}(\mathbb{Z})$$

no FW completo

Per essere preciso mi basterebbe poter aggiungere alla  $p$  gli insiemi di cardinalità 2. A questo punto non sarebbe più l'insieme delle costanti ma qualcos'altro, ma questa osservazione aggiuntiva mi permetterebbe di mantenere la precisione. Quindi è il risultato che ci fa perdere precisione, e non il fatto che abbiamo approssimato l'input.

$$p f \quad p (\underline{m_1 \square m_2}) = \underline{\text{non so}} \Rightarrow \text{BN completo}$$

$\downarrow m_1, m_2$

Esempio 2

$$C = \mathcal{B}(\mathbb{Z} \cup \mathbb{R} \cup \mathbb{S}\mathbf{tr}) \quad A = \text{Typ} = \{\text{Int}, \text{Float}, \text{Str}\}$$

$f = +$  (sommiamo un numero a una stringa, la stringa viene convertita in numero)

$$\text{Es. } 2 + \text{ciao} \rightsquigarrow 2 + 0 = 2$$

$$p f p \rightsquigarrow \text{Int} +^* \text{Str} = \text{Non so}$$

Se sommiamo un intero a una stringa, il problema è che la stringa potrebbe avere un prefisso intero e non avere un prefisso oppure avere un prefisso reale e quindi float. Questo cambia il risultato, perché se il prefisso è float la somma con intero comunque mi dà un int, mentre se il prefisso è float la somma con intero mi dà un float. Quindi in realtà non posso sapere quale sarà il tipo risultato

$$p f p \rightsquigarrow p(\text{Int} +^* \text{Str}) = p(\underline{\{\text{Int}, \text{Float}\}}) = \text{non so}$$

$$p f \rightsquigarrow \begin{cases} \text{Int} \\ \text{Float} \end{cases} \neq$$

$$p f p \rightsquigarrow p(\text{Int} +^* \text{Str}) = p(\underline{\{\text{Int}, \text{Float}\}}) = \text{non so}$$

$$p f \rightsquigarrow \begin{cases} \text{Int} \\ \text{Float} \end{cases} \neq \text{No BN completo}$$

$$p p \rightsquigarrow \{\text{Int}, \text{Float}\} \neq \text{No FW completo}$$

In questo caso:

- **Backwards completezza:** Il problema è nell'input, perché non ho saputo distinguere fra le stringhe che mi danno somma di un intero a intero e somma di un intero e float a float. Quindi, per la BW completezza potremmo pensare di raffinare la  $\rho$  per essere maggiormente precisi.

$$\text{Typ}' = \{\text{Int}, \text{Float}, \text{StrInt}, \text{StrFloat}\}$$

In questo caso,

$$\rho f p \rightsquigarrow \rho ( \text{Int} + \begin{array}{l} \xrightarrow{\text{StrInt}} \text{Int} \\ \xrightarrow{\text{StrFloat}} \text{Float} \end{array} ) = \text{Int} \sqcup \text{Float}$$

$\rho f \quad \text{MD} \quad \begin{array}{c} \text{Int} \\ \text{Float} \end{array} \quad \begin{array}{c} \xrightarrow{\quad} \\ \quad \end{array} \quad \text{Int} \sqcup \text{Float}$

- **Forward completezza:** vorremmo un  $\rho$  preciso sull'output, quindi potremmo aggiungere un tipo numerico generico

$$\text{Typ}' = \{\text{Num}, \text{Int}, \text{Float}, \text{Str}\}$$

$$\rho f p = \rho \{\text{Int}, \text{Float}\} = \text{Num}$$

$$f\rho = \{\text{Int}, \text{Float}\} = \text{Num}$$

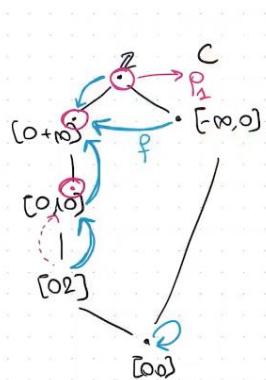
È da osservare che nel contesto dell'analisi statica il tipo di precisione ch'esi utilizza, perché ha il significato di andare a confrontare le proprietà astratte del calcolo concreto e astratto, è la backwards completezza.

Completezza

La completezza è una proprietà del dominio astratto rispetto ad un'operazione  $f$ . Un dominio è corretto o meno per una certa operazione.

Esempio

La funzione operatore approssima il quadrato



Notiamo che  $\rho$  non è BW completo per  $f$ :

$$\begin{aligned} \rho f p ([0, 2]) &= \rho f [0, 10] \\ &= \rho [0, 10] = [0, 10] \\ \rho f [0, 2] &= \rho [0, 10] = [0, 10] \end{aligned}$$

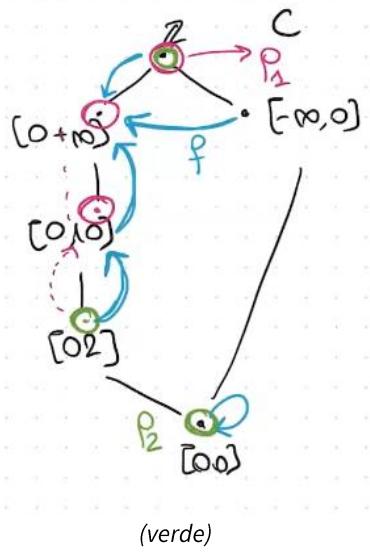
Invece è forward completo; lo si può dimostrare formalmente, ma vediamo solo un esempio

$$f_{\rho_1}[\rho_2] = f[\rho_1, \rho_2] = [\rho_1 + \rho_2]$$

Intuitivamente:

- la FWCompletezza deriva dal fatto che ogni elemento nella chiusura  $\rho_1$  ha come immagine di  $f$  un elemento che è dentro la chiusura  $\rho_1$
- La BWCompletezza deriva dal fatto che ci sono punti non dentro  $\rho_1$  che vengono mappati in elementi di  $\rho_1$

Esempio 2



- È BWC
- Non è FWC

$$\begin{aligned} P_2 f P_2 [\rho_2] &= P_2 f [\rho_2] = P_2 [\rho_1] = \mathbb{Z} \\ P_2 f [\rho_2] &= P_2 [\rho_1, \rho_2] = \mathbb{Z} \quad f P_2 = [\rho_1, \rho_2] \neq \mathbb{Z} \end{aligned}$$

Trasferire la semantica

$$F : C \rightarrow C \quad \text{reticolo, concreto}$$

$$\alpha : C \rightarrow A \quad (\text{GI tra } C \text{ e } A)$$

L'ultimo passaggio che ci serve è quello del **trasferimento della semantica**; il passaggio con la BCA va bene per le operazioni, ma non va bene per calcoli che possono divergere.

Vogliamo calcolare il least fixed point senza calcolare  $lfp(F)$ , che può divergere dato che siamo sul concreto.

Tipicamente quello che si fa è **trasferire il punto fisso**, ovvero si cerca un operatore  $\bar{F} : A \rightarrow A$  t.c.  $lfp \bar{F} = \alpha lfp F$

La soluzione ideale sarebbe prendere  $\bar{F} = \alpha F \gamma$ , ma purtroppo questo NON implica quest'uguaglianza. Esistono delle condizioni specifiche aggiuntive per garantire questo trasferimento in modo esatto, e sono abbastanza forti ma non ci addentriamo.

Dobbiamo accettare quindi di approssimare: una situazione in cui

$$\exists \bar{F} t. c. \alpha lfp F \not\subseteq lfp \bar{F}$$

(ovvero esiste un operatore che non riusciremo a costruire che approssima meglio)

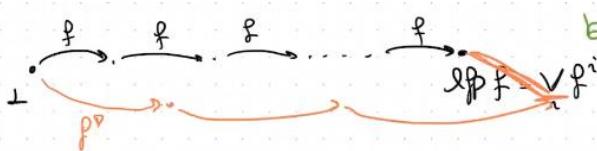
Le **condizioni che garantiscono il trasferimento** approssimato sono più deboli, e quindi più facili da garantire e da ottenere. Anche il punto fisso astratto comunque può avere problemi di terminazione. In particolare, **questo termina se:**

- $A$  è finito
- $A$  è ACC, ovvero non ha catene ascendenti infinite

Quindi, se  $A$  è **infinito non ACC** non abbiamo garanzie che il *lfp* astratto termini, e quindi sia decidibile. **Questo ad esempio succede negli intervalli**, dove ho catene ascendenti infinite e dominio infinito; questo significa che il dominio astratto così non è sufficiente.

Per garantire terminazione abbiamo bisogno di:

- Un **operatore astratto**  $\bar{F}$  su  $A$
  - Un **calcolo approssimato del lfp**, basato su un operatore detto di **widening** invece che sul *least upper bound*, definito sul dominio.
- È un acceleratore:** se partiamo da bottom, arriva a un punto fisso approssimato che può essere un punto fisso più grande o un post-punto fisso



## ANALISI NON DISTRIBUTIVE

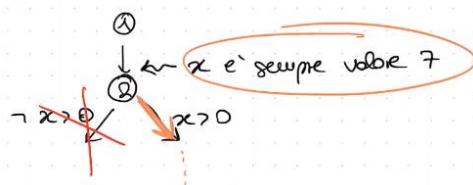
### 19 – Interpretazione astratta – Propagazione delle costanti

Le analisi non distributive sono analisi che riguardano il cosa viene calcolato, e sono quindi considerate le più difficili da calcolare. Entrano nel merito dei valori che attribuiamo alle variabili, e cercano proprietà di questi valori. La prima che introduciamo è la propagazione delle costanti.

## ANALISI NON DISTRIBUTIVE: PROPAGAZIONE DELLE COSTANTI

<<< Dato un punto P, determina se una variabile nel punto di programma P è sempre un valore costante.

È un'analisi che permette di fare la cosiddetta valutazione parziale, che a sua volta è collegata al concetto di specializzazione discusso a fondamenti; cerchiamo di capire se una variabile ha sempre lo stesso valore e in questo caso possiamo usare quel valore per valutare parzialmente il nostro programma e vedere se alcuni dei calcoli possono essere elaborati in funzione di quel valore costante che ha il programma.



Ci permette anche di capire che valori raggiungono un certo punto di programma, e come side effect di capire se un punto di programma è raggiungibile.

Esempio

Supponiamo di avere un programma che formalizza una serie di variabili

	a	b	c	d	e
1	1	1	2	3	3
2	1	1	2	3	3
3	1	2	3	4	-1
4	1	2	3	4	-1
5	1	2	3	3	0
6	1	2	3	3	0
7	1	2	3	3	0
8	1	2	3	3	0
9	1	2	3	3	0
10	1	2	3	3	0

47:58

È facile notare che eseguendo nel concreto il programma il valore di c è sempre 3, ma nello schemino non lo vedo perché vedo sommati dei valori di cui non so il valore.

1. Determiniamo il dominio delle informazioni astratte

...ovvero delle proprietà che voglio osservare con precisione.

Per esempio, decidiamo che il dominio astratto sono le singole variabili; quando un valore raccoglie più di un valore al suo interno non siamo più in grado di osservarlo.

$$C = \wp(\mathbb{Z}) \quad A = \{m \mid m \in \mathbb{Z}\} \cup \{\top, \perp\}$$

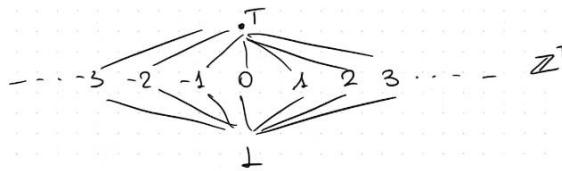
Non sappiamo se è reale (che dipende da più valori) o se è un errore di approssimazione. In generale, comunque vuol dire che non possiamo dire nulla sull'essere costante di quella variabile e quindi non c'è più nulla che possiamo osservare con precisione.

Riusciamo a costruire una GI fra C e A usando le seguenti funzioni:

$$\alpha(x) = \begin{cases} \perp & x = \emptyset \\ m & x = \{m\}, m \in \mathbb{Z} \\ \top & \text{altrimenti} \end{cases} \quad \beta(a) = \begin{cases} \emptyset & a = \perp \\ \{m\} & a = m \\ \mathbb{Z} & \text{altrimenti} \end{cases}$$

Non so

E l'ordine è



$$x, y \in \mathbb{Z}^T \\ x \sqsubseteq y \quad \text{sse} \quad x = y \quad \vee \quad y = T \quad (\vee \quad x = \perp)$$

Questa è un'**astrazione dei valori**, ma la **semantica astratta calcola su stati**!

Quindi non è ancora questa l'astrazione che ci interessa, ci interessano gli stati astratti.

## 2. Traduzione in stati astratti

Stato :  $\sigma : \text{Var} \rightarrow \mathbb{Z}$

Stato collecting :  $S : \text{Var} \rightarrow \mathcal{B}(\mathbb{Z})$

Stato astratto :  $D : \text{Var} \rightarrow \mathbb{Z}^T \Rightarrow \mathbb{D} = (\text{Var} \rightarrow \mathbb{Z}^T) \sqcup \{\perp\}$   
(costanti)

Stati  
astratti  
nelle  
costanti

Il bottom rappresenta i punti non raggiungibili. Sarà minore di ogni memoria del nostro dominio.

$D_\perp \stackrel{\text{def}}{=} \forall x. x \mapsto \perp$   
memoria con nessuna  
variabile inizializzata

## 3. Definizione degli operatori

Ordimento :  $D_1 \sqsubseteq D_2 \quad \text{sse} \quad \forall x \in \text{Var}. D_1(x) \sqsubseteq_{\text{cost}} D_2(x)$

Lub :  $\bigcup_i D_i(x) = \begin{cases} \mathbb{Z} & \text{se } \forall i. D_i(x) = \mathbb{Z} \in \mathbb{Z} \\ T & \text{altrimenti} \end{cases}$

$\mathbb{D}, \sqsubseteq, \sqcup$  è reticolo completo  $\rightarrow$  dominio astratto  
per le costanti

Dobbiamo definire una semantica astratta dei nostri programmi nel dominio delle costanti:

$\forall D \in \mathbb{D} . \llbracket \cdot \rrbracket^D \text{ mi dice se ogni variabile}$   
 $\text{è costante (che valore) o meno.}$

Vogliamo questa info per ogni punto di programma, quindi cerchiamo la soluzione MOP: definiamo

$$D^*[v] = \bigcup \llbracket \pi \rrbracket^D \perp \mid \pi : \text{start} \rightarrow v \}$$

Ovvero **prendiamo tutti i cammini che arrivano al nodo**, ne facciamo la **semantica** e ne **combiniamo** il risultato.

Questa analisi è forward e possible.

Dobbiamo quindi definire esattamente la semantica approssimata astratta

Dobbiamo quindi definire  $\llbracket \cdot \rrbracket^*$ ,  $(u, lub, v)$  ordini  
 $\llbracket \cdot \rrbracket^{lub} \perp$  che  
raggiunge u

## Definizione della semantica approssimata astratta

### 1. Trasferimento delle operazioni

$\square$  è una operazione su  $\mathbb{Z}$   
dobbiamo definire  $\square^*$  su  $\mathbb{Z}^T$

$$a, b \in \mathbb{Z}^T \quad a \square^* b = \begin{cases} T & \text{se } a=T \vee b=T \\ a \square b & \text{altrimenti} \end{cases}$$

$\Rightarrow$  bca di  $\square$

### 2. Definiamo la semantica delle espressioni $[\![e]\!]^* D ([\![e]\!]: D \rightarrow \mathbb{Z}^\perp)$

Per la data flow analisi non la abbiamo mai definita, perché non ci è mai davvero interessato quale fosse il valore restituito dall'espressione; ci interessava solo la struttura sintattica.

Ovvio che invece in un'analisi non distributiva **la valutazione delle espressioni prende importanza, perché restituisce un valore nel dominio astratto che decidiamo di osservare**.

Definiamo induttivamente la semantica:

$$\left\{ \begin{array}{l} c \in \mathbb{Z} . [\![c]\!]^* D = c \quad [\![x]\!]^* D = D(x) \in \mathbb{Z}^T \\ [\![\square e]\!]^* D = \square^* [\![e]\!]^* D \quad \text{semantico astratto} \\ [\![e_1 \square e_2]\!]^* D = [\![e_1]\!]^* D \square^* [\![e_2]\!]^* D \quad \text{delle espressioni} \end{array} \right.$$

Esempi di valutazione dell'espressione astratta

$$\begin{aligned} D &= [x \mapsto 2, y \mapsto T] \\ [\![x+7]\!]^* D &= [\![x]\!]^* D +^* [\![7]\!]^* D \\ &= D(x) +^* 7 = 2 +^* 7 = 9 \\ [\![x+y]\!]^* D &= [\![x]\!]^* D +^* [\![y]\!]^* D = D(x) +^* D(y) \\ &= 2 +^* T = T \end{aligned}$$

$$\begin{aligned} [\![j]\!]^* D &= D \\ [\![\text{NonZero}(e)]\!]^* D &= \begin{cases} \perp & O = [\![e]\!]^* D \\ D & \text{altrimenti } (\exists m \neq 0. [\![e]\!]^* D = m \vee [\![e]\!]^* D = T) \end{cases} \\ [\![\text{Zero}(e)]\!]^* D &= \begin{cases} \perp & O \neq [\![e]\!]^* D \\ D & \text{altrimenti: } ([\![e]\!]^* D = 0 \vee [\![e]\!]^* D = T) \end{cases} \end{aligned}$$

$[\![x \leftarrow e]\!]^* D = D[x \mapsto [\![e]\!]^* D]$

$[\![\text{input}(x)]\!]^* D = D[x \mapsto T] \quad \text{sappiamo che è costante} \quad \text{ma non sappiamo due valore}$

Quindi la semantica di un cammino sarà

$$\pi = k_0 \dots k_m \quad [\![\pi]\!]^* = [\![k_m]\!]^* \circ \dots \circ [\![k_0]\!]^* D_\perp$$

$\swarrow [\![lab_m]\!]^*$

Purtroppo questa semantica non è distributiva, quindi **se anche calcoliamo la soluzione MFP questa non è assolutamente uguale alla MOP, anzi in generale la contiene strettamente, ma è l'unica soluzione che possiamo costruire**.

**Comunque, la semantica è monotona e il dominio è ACC, quindi la MFP esiste ed è calcolabile come minima soluzione del sistema di disequazioni.**

Come la costruisce? Fornendo la soluzione del sistema di disequazioni come sempre fatto per la data flow analysis:

$$\begin{cases} D[\text{entuf}] \supseteq D_1 \\ D[N] \supseteq [\text{lab}]^* D[u] \end{cases} \quad (u, \text{lab}, N)$$

## Esempio di non distributività

**Non dimostriamo** che non è distributiva, però ne diamo un esempio.

$$\begin{aligned} & [x \leftarrow x+y]^* D_1 \sqcup [x \leftarrow x+y]^* D_2 \\ & D_1[x \mapsto 5] \qquad \qquad D_2[x \mapsto 5] \\ & \underbrace{[x \mapsto 5, y \mapsto 3]}_{\text{LUB}} \sqcup \underbrace{[x \mapsto 5, y \mapsto 2]}_{\text{LUB}} = [x \mapsto 5, y \mapsto T] \end{aligned}$$

Se invece calcoliamo la semantica combinando le memorie localmente, allora vediamo che il LUB fra D1 e D2 sia x che y hanno valori diversi:

$$\begin{aligned} & [[x \leftarrow x+y]]^*(D_1 \sqcup D_2) = [[x \leftarrow x+y]]^* [x \mapsto T, y \mapsto T] \\ & = [x \mapsto T, y \mapsto T] \end{aligned}$$

C'è una differenza fra le due memorie, quindi la semantica non è distributiva. Non possiamo calcolare precisamente la semantica lungo i cammini come semantica delle combinazioni locali della memoria raggiunta.

Di conseguenza, quando facciamo nel secondo modo abbiamo un risultato più astratto.

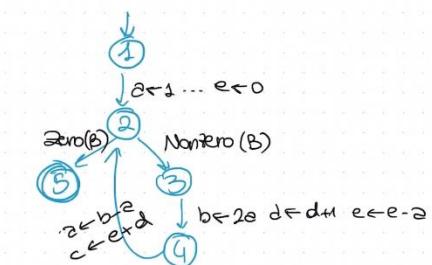
## Esempio di analisi

Facciamo l'esempio introdotto prima.

```

1  a:=1; b:=2; c:=3; d:=3; e:=0
2  while B {
3      b:= 2*a; d:= d+1; e:= e-a;
4      a:= b-e; c:= e+d;
5 }
```

### 1. Control flow graph



## 2. Sistema di disequazioni

$$D(1) = D_L$$

$$D(2) = [e \mapsto 1, b \mapsto 2, c \mapsto 3, d \mapsto 3, e \mapsto 0] \sqcup$$

$$\underline{D(4)} [a \mapsto D(4)(a) -^* D(4)(b), c \mapsto D(4)(c) +^* D(4)(d)]$$

$$D(3) = D(2)$$

$$D(4) = D(3) [b \mapsto 2 *^* D(3)(a), d \mapsto D(3)(d) + 1, e \mapsto D(3)(e) - \frac{D(3)(a)}{D(3)(a)}]$$

5

$$D(5) = D(2)$$



## 3. Cerchiamo la soluzione come MFP.

	0	1	2
1	D <sub>L</sub>	D <sub>L</sub>	
2	↓	a $\mapsto$ 1 b $\mapsto$ 2 c $\mapsto$ 3 d $\mapsto$ 3 e $\mapsto$ 0	
3	↓	"	
4	↓	a $\mapsto$ 1 b $\mapsto$ 2 c $\mapsto$ 3 d $\mapsto$ 4 e $\mapsto$ -1	
5	↓	"	

	0	1	2
1	D <sub>L</sub>	D <sub>L</sub>	D <sub>L</sub>
2	↓	a $\mapsto$ 1 b $\mapsto$ 2 c $\mapsto$ 3 d $\mapsto$ 3 e $\mapsto$ 0	a $\mapsto$ 1 b $\mapsto$ 2 c $\mapsto$ 3 d $\mapsto$ 4 e $\mapsto$ -1
3	↓	"	"
4	↓	a $\mapsto$ 1 b $\mapsto$ 2 c $\mapsto$ 3 d $\mapsto$ 4 e $\mapsto$ -1	a $\mapsto$ 1 b $\mapsto$ 2 c $\mapsto$ 3 d $\mapsto$ T e $\mapsto$ T
5	↓	"	"

→

	0	1	2	3
1	D <sub>L</sub>	D <sub>L</sub>	D <sub>L</sub>	D <sub>L</sub>
2	↓	a $\mapsto$ 1 b $\mapsto$ 2 c $\mapsto$ 3 d $\mapsto$ 3 e $\mapsto$ 0	a $\mapsto$ 1 b $\mapsto$ 2 c $\mapsto$ 3 d $\mapsto$ T e $\mapsto$ T	(a $\mapsto$ 1, b $\mapsto$ 2, c $\mapsto$ 3, d $\mapsto$ T, e $\mapsto$ T)
3	↓	"	"	"
4	↓	a $\mapsto$ 1 b $\mapsto$ 2 c $\mapsto$ 3 d $\mapsto$ 4 e $\mapsto$ -1	(a $\mapsto$ 1, b $\mapsto$ 2, c $\mapsto$ 3 d $\mapsto$ T, e $\mapsto$ T)	(a $\mapsto$ 1, b $\mapsto$ 2, c $\mapsto$ 3, d $\mapsto$ T, e $\mapsto$ T)
5	↓	"	"	"

Nota

Possiamo migliorare l'analisi considerando che ci sono delle guardie la cui informazione calcolata è rappresentabile come informazione.

If  $x=7$  then  $\{ \text{supposo } x=7 \}$   $\xrightarrow{x=7}$   $\{ \text{informazione rappresentabile in } Z \}$

Possiamo descrivere questa semantica migliorata come:

$$[\text{Nonzero}(a=e)]^* D = \begin{cases} \perp & [x=e]^* D = 0 \\ D_L & \text{altr.m.} \end{cases}$$

$$\hookrightarrow D_L = D[x \mapsto D(x) \cap [e]^* D]$$

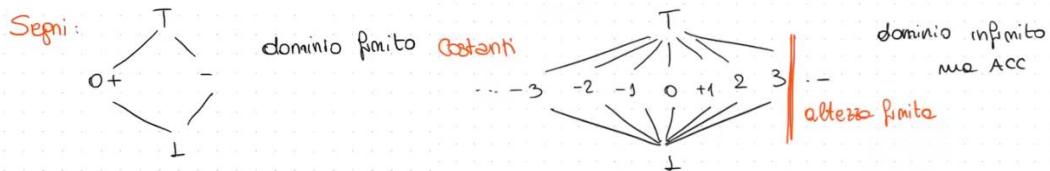
$$= [e]^* D$$

## ANALISI NON DISTRIBUTIVE: ANALISI DEGLI INTERVALLI

L20 - Interpretazione astratta 5 - Analisi degli intervalli

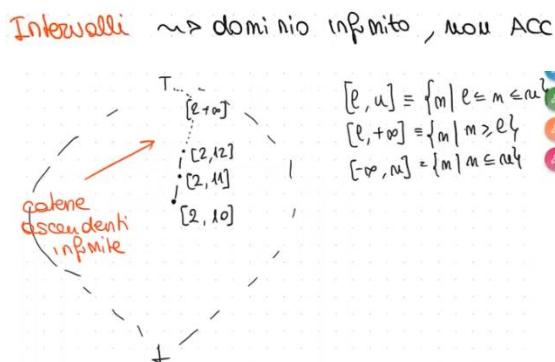
### Introduzione

La scorsa volta abbiamo capito come funziona l'analisi definita attraverso un dominio astratto, e in particolare un dominio astratto che abbia definita una relazione con gli oggetti concreti e che mi permette quindi di calcolare la semantica del programma su oggetti che sono dati. Abbiamo fatto questa analisi sulle costanti, quindi il nostro obiettivo era verificare se c'erano dei valori che restavano costanti.



Purtroppo questi sono casi ideali e anche purtroppo rari; in generale, quando ci interessa una proprietà che abbia almeno un minimo di informazione rilevante per analisi più sofisticata **perdiamo sia la finitezza, sia l'essere ACC**.

Vediamo quindi in questo ambito gli intervalli:



Abbiamo visto che il calcolo del punto fisso lavora incrementando i valori, ovvero raccogliendo i possibili valori che ogni variabile può avere durante l'esecuzione; **lavorare in un dominio non ACC significa lavorare in un dominio che può divergere, perché posso continuare ad aggiungere elementi senza mai trovare un punto fisso.**

L'aspetto interessante dell'analisi degli intervalli è che è un caso semplice che contiene tutta la difficoltà dell'analisi statica.

### Analisi degli intervalli

L'analisi degli intervalli ha come obiettivo l'**identificazione del range di valori che una variabile può acquisire durante la sua esecuzione. Solitamente ci interessano le variabili in un ciclo**, perché sono le uniche dove ci può essere un po' di variabilità.

*Dove approssimiamo?*

Gli intervalli sono **insiemi convessi**, quindi **approssimano "riempendo i buchi"**.

Se abbiamo quindi una variabile x che può assumere i valori

$$x \rightsquigarrow \{1 \ 5 \ 20 \ 35\}$$

L'intervallo "riempie i buchi", e prende tutti i valori in mezzo:

$$\{1 \ldots 5 \ldots 20 \ldots 35\} \rightarrow [1 \ldots \ldots 35]$$

Come al solito, dobbiamo seguire dei passi.

- Costruzione/definizione del dominio degli elementi che vogliamo osservare con precisione.

$$\mathbb{I} = \{ [l, u] \mid l \in \mathbb{Z} \cup \{-\infty\}, u \in \mathbb{Z} \cup \{+\infty\}, l \leq u \}$$

(aggiungiamo +/- infinito per poter fare gli intervalli aperti)

Ora dobbiamo assicurarchi che esista un dominio astratto, ovvero che esista un'inserzione di Galois tra questo dominio e l'insieme del concreto.

$$\mathbb{I} = \{ [l, u] \mid l \in \mathbb{Z} \cup \{-\infty\}, u \in \mathbb{Z} \cup \{+\infty\}, l \leq u \}$$

<-GI-> Concreto : Insiemi di valori su  $\mathbb{Z}$  per le variabili  $\mathcal{G}(\mathbb{Z})$

Quindi: esiste la GI tra  $\mathcal{G}(\mathbb{Z})$  e  $\mathbb{I}$ ?

Dobbiamo fornire la  $\alpha$  e la  $\gamma$ .

-  $\alpha$

$$\alpha(X) = [\min_{\infty}(X), \max_{\infty}(X)] \in \mathbb{I}$$

ovvero, prendo il minimo e il massimo del mio insieme di elementi e ci faccio un intervallo. Si può dimostrare che è monotona ma vbb.

Nota: dobbiamo ridefinire  $\min$  e  $\max$  perché nella loro versione standard non danno risposta in caso di  $\infty$ !

$$\min_{\infty}(X) = \begin{cases} \text{mex}_{-\infty} & \text{se } \forall m \in X, m \leq m \\ & \text{altrimenti} \end{cases} \quad (\text{minimo di } X)$$

$$\max_{\infty}(X) = \begin{cases} \text{mex}_{+\infty} & \text{se } \forall m \in X, m \geq m \\ & \text{altrimenti} \end{cases}$$

-  $\gamma$

$$\gamma([l, u]) = \{n \in \mathbb{Z} \mid l \leq n \leq u\}$$

Ovvero, quando torno nel concreto metto tutti i numeri che stanno dentro il mio intervallo astratto. Evidentemente ne avrò  $\geq$  quanti ne avevo in partenza.

Nota: Mettiamo  $\leq_{\infty}$  e non  $\leq$  giusto per specificare che  $\forall m. -\infty \leq_{\infty} m \wedge m \leq_{\infty} +\infty$

Dimostriamo l'inserzione di Galois

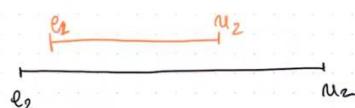
A questo punto dobbiamo dimostrare che queste sono una inserzione di Galois, ovvero che dato un qualunque insieme esiste sempre la migliore approssimazione possibile, ovvero esiste sempre il più piccolo intervallo che contiene l'inserzione da cui partiamo.

$$\mathcal{G}(\mathbb{Z}) \xrightarrow{\alpha} \mathbb{I}$$

Operazioni del reticolo

- $[l_1 u_1] \sqsubseteq [l_2 u_2]$  – è contenuto

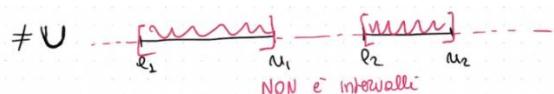
Quando è che due intervalli sono in relazione fra di loro?  
Per essere contenuto, deve avvenire che  $l_1 > l_2 \wedge u_1 < u_2$ .



- $[l_1 u_1] \sqcup [l_2 u_2]$  – least upperbound: è il più piccolo intervallo che contiene entrambi.

Sicuramente non può essere l'unione, perché l'unione crea buchi: Quindi, ci basta fare l'intervalllo che contiene l'unione per "riempire i buchi". Faccio l'intervalllo del lowe bound più piccolo e dell'upper bound più grande.

$$[l_1 u_1] \sqcup [l_2 u_2] = [\min(l_1, l_2), \max(u_1, u_2)]$$



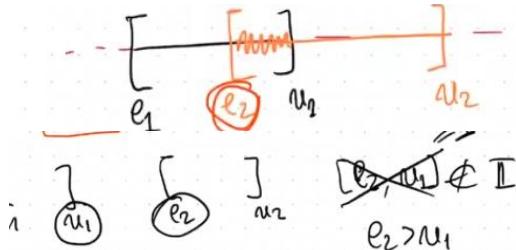
- $[l_1 u_1] \cap [l_2 u_2]$  – greatest lowerbound

E' più semplice, perché l'intersezione di intervalli è un intervallo 😊

Prendo il più grande dei lowerbound e il più grande degli upperbound.

$$[l_1 u_1] \cup [l_2 u_2] = [\max(l_1, l_2), \min(u_1, u_2)]$$

Se ho insiemi disgiunti, semplicemente è ⊥.



## 2. Trasferiamo le operazioni sul dominio astratto $\mathcal{D}$

Ne vediamo alcune.

- Somma:**  $[l_1 u_1] +^{\#} [l_2 u_2] = [l_1 + l_2, u_1 + u_2]$

Ricordandoci di gestire l'infinito, quindi  $-\infty + n = -\infty$  e  $+\infty + n = +\infty$

- Negazione:**  $-\# [l_1 u_1] = [-u_1 - l_1]$

- Moltiplicazione:** è la BCA della moltiplicazione fra interi

- Confronto:**  $[l_1 u_1] ==^{\#} \dots$

Il punto qui è che voglio rispondere a questo test in modo da poter approssimare la risposta concreta "n==m"; se dico vero devo essere certo che quel test è vero anche nel concreto

- [1 1] if  $l_1 = u_1 = j_2 = u_2$**

Posso dire con certezza che vero sugli intervalli implica vero sui valori solo quando ho un solo elemento...

Stabliamo che restituire [1 1] corrisponde a restituire True per evitare di dover fare casini fra intervalli e booleani.

- [0 0] if  $u_1 < l_2 \vee u_2 < l_1$**

Ovvero, se sono disgiunti so per certo che non c'è modo che i due valori siano uguali.

- [0 1] in tutti gli altri casi**

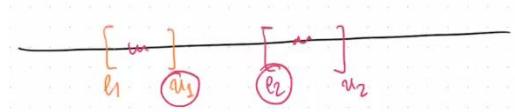
Cioè, non posso essere certo se sia vero o falso.

- Ordinamento**  $[l_1 u_1] \leq^{\#} [l_2 u_2]$

- [1 1] if  $u_1 \leq l_2$**

- [0 0] if  $u_2 \leq l_1$**

- [0 1] in tutti gli altri casi**



## 4. Passiamo dai valori agli stati

In realtà, il nostro dominio concreto vero non sono solo i valori ma sono degli stati del tipo

$$\begin{aligned} \text{Stati} : \text{Var} &\rightarrow \mathbb{S}(\mathbb{D}) \\ &\uparrow \\ \text{Stato} &= (\text{Var} \rightarrow \mathbb{I}) \cup \{\perp\} \\ &\text{concreto} \end{aligned}$$

Usiamo bottom per denotare uno stato di errore o di non eseguibilità. Chiamiamo  $D$  gli stati.

$\alpha$

Estendiamo l'alpha già vista agli stati.

$$\alpha(D) \Rightarrow \forall x \in \text{Var} \quad x \mapsto \underbrace{\alpha(D(x))}_{\substack{\text{insieme di valori} \\ \text{associati} \\ \text{a } x \text{ da } D}}$$

Esempio su un singolo stato:

$$D \left\{ \begin{array}{l} x \mapsto x \\ y \mapsto y \\ z \mapsto z \end{array} \right. \xrightarrow{\alpha(D)} \left\{ \begin{array}{l} x \mapsto \alpha(x) \\ y \mapsto \alpha(y) \\ z \mapsto \alpha(z) \end{array} \right. \quad \text{insieme di valori associati a } x \text{ da } D$$

Esempio su un insieme di stati con una semantica collective:

$$\left\{ \begin{array}{l} x \mapsto 2 \\ y \mapsto 3 \\ z \mapsto 4 \end{array} \right|, \quad \left\{ \begin{array}{l} x \mapsto 3 \\ y \mapsto 4 \\ z \mapsto 5 \end{array} \right| \subseteq \text{Stati}$$

\\$

$$\begin{array}{l} x \mapsto \{2, 3\} \\ y \mapsto \{3, 4\} \\ z \mapsto \{4, 5\} \end{array}$$

**Abbiamo perso l'informazione di quali numeri sono in quale insieme** (dato che abbiamo messo tutto assieme)

L'ordinamento rimane sempre uguale: quando confronto gli stati astratti confronto i valori di ogni variabile:

$$D_1 \sqsubseteq D_2 \text{ sse } \forall x \in Var, D_1(x) \sqsubseteq_{intervalli} D_2(x)$$

#### 4. Soluzione cercata

Come in tutti i casi, la soluzione ideale cercata è la soluzione MOP: in questo caso la definiamo come

*Soluzione cercata*

MOP:  $\mathcal{Y}^*[\alpha] = \bigcup \{ [\pi]^* D_1 \mid \pi: \text{entry} \rightarrow \alpha \}$

*stato astratto iniziale con variabili non inizializzate*

Per capire se la MOP può effettivamente essere definita, dunque, vado innanzitutto a definire la mia semantica.

*Abstract edge effect: semantica astratta sul CFG*

La definiamo induttivamente sulle tichette che possiamo avere sugli archi del CFG.

**Dovremmo definire anche la semantica delle espressioni, che però non ci dà perché è analoga a quella della propagazione delle copie.**

$$\begin{aligned} D \in D & \quad [j]^* D = D \\ \uparrow \text{Stati} & \\ \text{astratti} & \quad [x \leftarrow e]^* D = D[x \mapsto [e]^* D] \end{aligned}$$

*semantica espressioni analoghe alle definizioni x proposizione costante*

$$\begin{aligned} [ \text{Non zero}(e) ]^* D &= \begin{cases} \perp & [0, 0] = [e]^* D \\ D & \text{altrimenti } [-2, 2] \end{cases} \\ [ \text{Zero}(e) ]^* D &= \begin{cases} \perp & [0, 0] \notin [e]^* D \leftarrow [2, 10] \\ D & [0, 0] \subseteq [e]^* D \leftarrow [-2, 2] \end{cases} \end{aligned}$$

$$[\text{input}(x)]^* D = D[x \mapsto T] \quad \text{perdiamo informazioni sul valore di } x$$

- **Nonzero** (ramo da prendere se l'espressione è vera): vado in  $\perp$  solo se ho la certezza di non percorrere quel ramo, ovvero se l'espressione è sicuramente falsa, e quindi se l'intervallo è esattamente  $[0, 0]$
- **Zero** (ramo da prendere se l'espressione è falsa): l'espressione sicuramente non è falsa quando, in tutto l'intervallo dell'espressione valutata, NON HO MAI  $[0, 0]$ .
- Quando abbiamo un input potremmo ottenere qualunque valore.

Purtroppo, questa semantica non è distributiva. Quindi, non possiamo calcolare la MOP come soluzione MFP (=del sistema di disequazione).

*Sistema di disequazioni per II*

Sappiamo che è **forward**, che è **possible** e abbiamo uno **stato iniziale**; quindi ci sono tutti gli ingredienti per definire il sistema di disequazioni.

E' di tipo **possible**, quindi sovrastimiamo; partiamo dall'informazione iniziale  $D_\perp$ , cioè associamo ad ogni variabile il valore minimo.

Per ogni arco costruiamo

$$\begin{cases} \mathcal{J}[\text{entry}] \supseteq D_\perp \\ \mathcal{J}[N] \supseteq [\text{lab}]^* \mathcal{J}[n] \end{cases} \quad \forall K = (\underline{u}, \text{lab}, \underline{v})$$

Come dicevamo, purtroppo non abbiamo la distributività (che ci permetterebbe di dire che la soluzione è la MOP) né la finitezza, che ci permetterebbe di garantire che questo sistema di disequazioni abbia soluzione in tempo finito.

La **monotonia**, che invece abbiamo, mi garantisce che questo punto esiste, ma l'esistenza potrebbe anche esistere all'infinito 😞 La soluzione sarà, quinidi, di non cercare la soluzione migliore possibile ma cercare la soluzione raggiungibile in tempo finito.

### Migliorare la semantica

Come per le costanti, anche gli intervalli sono un dominio di tipo numerico, quindi alcuni tipi di test trovano una rappresentazione dentro il dominio degli intervalli; quinidi possiamo rendere la seguente informazione  $D$  più precisa:

$$\begin{aligned} [\text{Non zero}(e)]^* D &= \begin{cases} \perp & [0,0] = [e]^* D \\ D & \text{altrimenti } \leftarrow [-2,2] \end{cases} \\ [\text{zero}(e)]^* D &= \begin{cases} \perp & [0,0] \notin [e]^* D \leftarrow [2,10] \\ D & [0,0] \subseteq [e]^* D \leftarrow [-2,2] \end{cases} \end{aligned}$$

Anziché lasciare  $D$  esattamente quello che è all'ingresso, se l'espressione me lo permette, posso integrare l'informazione sulla falsità o meno della guardia dentro lo stato che raggiungo.

Quindi, arriviamo a una cosa di questo tipo: uniamo  $D(x)$  precedente e lo mettiamo in or con la nuova informazione ottenuta...

$$D_1 = \begin{cases} D[x \mapsto D(x) \cap [e_1]^* D] & \text{se } e = (x = e_1) \\ D[x \mapsto D(x) \cap [-\infty, -u_1]] & \text{se } e = x < e_1 \\ D[x \mapsto D(x) \cap [e_1, \infty]] & \text{se } e = x > e_1 \end{cases}$$

$[e_1]^* D = [e, u]$

E' chiaro che questo vale solo quando  $e = x \{=, <, >\} e_1$

Si può fare analogamente per il caso 0.

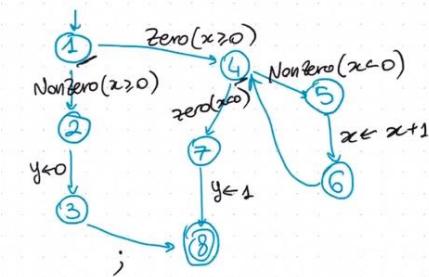
### Esempio di risoluzione

---

```

if (x >= 0) then 3 y <- 0
else 4 while x < 0 do x <- x+1
      y <- x
  
```

## 1. Costruiamo il CFG



## 2. Costruiamo il sistema

$\mathcal{J}(1) = \overline{\mathcal{J}_1} \rightarrow$  Non avere informazioni ostacolate sulle variabili equivalenti a dire non so ( $T$ ) per ogni variabile

$$\mathcal{J}(2) = \mathcal{J}(1) \cap \mathcal{J}_2 [x \mapsto [0, +\infty]] = \mathcal{J}(1) \cap [0, +\infty] \text{ per } x$$

$$\mathcal{J}(3) = \mathcal{J}(2) [y \mapsto [0, 0]]$$

$$\mathcal{J}(4) = (\mathcal{J}(1) \cap [-\infty, 1] \text{ per } x) \sqcup \mathcal{J}(6)$$

$$\mathcal{J}(5) = \mathcal{J}(4) \cap [-\infty, -1] \text{ per } x$$

$$\mathcal{J}(6) = \mathcal{J}(5) [x \mapsto \mathcal{J}(5)(x) +^* [1, 1]]$$

$$\mathcal{J}(7) = \mathcal{J}(4) \cap [0, +\infty] \text{ per } x$$

$$\mathcal{J}(8) = \mathcal{J}(7) [y \mapsto \mathcal{J}(7)(x)] \sqcup \mathcal{J}(3)$$

## 3. Facciamo il calcolo.

Come partenza mettiamo tutti bottom, perché stiamo andando per possible – e quindi ad incrementare.

	0	1
1	$y_L$	$y_L$
2	$\perp$	$[0, +\infty]x$ $T_y$
3	$\perp$	$[0, +\infty]x$ $[0, 0]y$
4	$\perp$	$[-\infty, -1]x$ $T_y$
5	$\perp$	$[-\infty, -1]x$ $T_y$
6	$\perp$	$[-\infty, 0]x$ $T_y$
7	$\perp$	$\perp$
8	$\perp$	$\perp$

$$[-\infty, -1] +^* [1, 1] = [-\infty, 0]$$

$\perp$	$[0, +\infty]x$ $T_y$	$[0, +\infty]x$ $T_y$
$\perp$	$[0, +\infty]x$ $[0, 0]y$	$[0, +\infty]x$ $[0, 0]y$
$\perp$	$[-\infty, -1]x$ $T_y$	$[-\infty, -1]x$ $T_y$
$\perp$	$[-\infty, -1]x$ $T_y$	$[-\infty, -1]x$ $T_y$
$\perp$	$[-\infty, 0]x$ $T_y$	$[-\infty, 0]x$ $T_y$
$\perp$	$[-\infty, 0]x$ $T_y$	$[-\infty, 0]x$ $T_y$
$\perp$	$[0, 0]x$ $T_y$	$[0, 0]x$ $T_y$
$\perp$	$[0, 0]y$	$[0, 0]y$

$$[-\infty, 0]x = \boxed{[-\infty, 0]x}$$

$$[0, +\infty]x = \boxed{[0, +\infty]x}$$

Punto fisso raggiunto.

Il problema è che basta un test che richieda 100 o 1000 iterazione ch e anche gli intervalli ci danno risposte in tempi non accettabili.

Da questa operazione quindi nasce il **widening**.

## ANALISI NON DISTRIBUTIVE: WIDENING (E NARROWING)

### Il widening

Serve ad **accelerare le computazioni di punto fisso** garantendo la convergenza. Questo avviene attraverso la definizione di un operatore che permette di eseguire dei salti per poter raggiungere un punto fisso approssimato.

Il **widening** si rappresenta come triangolino girato:

$$\begin{array}{c} P \text{ dominio } P, E \\ \nabla : P \times P \longrightarrow P \end{array}$$

L'idea è che **per ogni insieme di elementi appartenenti a P, x deve essere rispettato un insieme di vincoli:**

1.

$$\forall x, y \in P. \quad x \sqsubseteq (x \nabla y) \wedge y \sqsubseteq x \nabla y$$

$\nwarrow$   
 $x \nabla y$

2. Data una catena  $x^0 \sqsubseteq x^1 \sqsubseteq x^2 \dots \sqsubseteq x^n$

a. Costruiamo una catena

$$\begin{array}{c} x^0 \sqsubseteq x^1 \sqsubseteq x^2 \dots \sqsubseteq x^n \sqsubseteq \\ \swarrow \quad \searrow \quad \swarrow \quad \searrow \quad \swarrow \quad \searrow \\ y^0 = x^0, \quad y^1 = y^0 \nabla x^1 \quad \dots \quad y^{n+1} = y^n \nabla x^{n+1} \end{array}$$

$y^i$  è una catena non strettamente crescente

dove **combiniamo sempre l'elemento precedente della nuova catena col nuovo elemento della catena originale.**

Prima o poi converge.

b. Così non sembra significativa, ma possiamo osservare che le nostre disequazioni sono del tipo

$$\begin{array}{ll} x_i \sqsupseteq f_i(x_1 \dots x_n) & f(i) \sqsupseteq \sqcup_{j=1}^n f_j(\dots) \\ \Downarrow & \\ x_i = x_i \sqcup f_i(x_1 \dots x_n) & \end{array}$$

= un oggetto è più grande di un altro se e solo se è uguale al least upperbound di se stesso con quell'elemento.

c. A questo punto **sostituiamo il widening al least upperbound**, e otteniamo:

$$\begin{array}{c} x_i = x_i \sqcup f_i(x_1 \dots x_n) \\ \Downarrow \\ \boxed{x_i = x_i \nabla f_i(x_1 \dots x_n)} \end{array}$$

**Questo sistema di disequazioni garantisce la costruzione di una catena che ha sicuramente punto fisso in un numero finito di passi.**

Quindi, l'idea di fondo è di dare **un operatore che sostituiremo al least upper bound**. Il lub è il migliore operatore possibile, quindi permette che la catena possa essere potenzialmente infinita e quindi divergere; invece **il widening tra tutti gli upperbound va a scegliere specificamente una sottocatena che a un certo punto converge**.

Ovviamente **approssima**: di conseguenza, di solito, non viene considerato da solo ma in coppia con un'operazione di narrowing, che presa la soluzione molto approssimata del widening cerca di capire se il calcolo aveva delle informazioni che abbiamo buttato via mentre potevamo sfruttarle. Non la vediamo formalmente, ma direttamente intuitivamente sugli intervalli.

### Widening degli intervalli

Non è commutativo, quindi lo devo dare in entrambi i versi: a sinistra abbiamo il vecchio valore, e a destra abbiamo il nuovo valore. Questa informazione è usata nel widening.

$$\underline{\perp \triangleright D} = \underline{D \triangleright \perp} = D$$

Nel caso generale voglio calcolare

$$D_1 \triangleright D_2 (x) = D_1(x) \triangleright D_2(x)$$

↗ nel dominio  
II

Quindi, nello specifico,

$$[l_1, u_1] \triangleright [l_2, u_2] = [l, u]$$

Vogliamo essere conservativi, quindi mantenere sempre la cosa con più valori:

$$[l_1, u_1] \triangleright [l_2, u_2] = [l, u]$$

$$l = \begin{cases} l_1 & l_1 \leq l_2 \\ -\infty & l_2 < l_1 \end{cases}$$

The diagram shows a horizontal spring divided into two segments by a vertical dashed line. The left segment is labeled  $l_1$  and the right segment is labeled  $l_2$ .

Nel caso in cui  $l_2$  sia più piccolo (e quindi dovrei effettivamente allargare il mio intervallo), il widening dà per scontato che potrei andare avanti all'infinito; quindi dà direttamente  $-\infty$ .

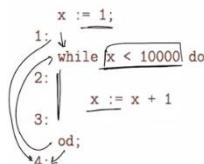
$$u = \begin{cases} u_1 & u_1 \geq u_2 \\ +\infty & u_2 > u_1 \end{cases}$$

The diagram shows a horizontal line with two points marked:  $u_1$  and  $u_2$ . An arrow points from  $u_1$  towards  $u_2$ .

Analogamente per il limite superiore: se ho un trend di crescita vado direttamente a  $+\infty$

Sto cercando di catturare la situazione in cui, linearmente, l'upperbound cresce linearmente e il lowebound decresce linearmente.

### Esempio di widening



Questo programma termina, ma ha una marea di iterazioni. Quello che avviene e che, avendo scritto le equazioni, ottengo

$$\begin{cases} X_1 = [1, 1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1, 1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

Iniziando il calcolo ottengo

$$\begin{cases} X_1 = \emptyset \\ X_2 = \emptyset \\ X_3 = \emptyset \\ X_4 = \emptyset \end{cases} \rightarrow \text{itero} \rightarrow \begin{cases} X_1 = [1, 1] \\ X_2 = [1, 1] \\ X_3 = [2, 2] \\ X_4 = \emptyset \end{cases}$$

$(x_4 \text{ rimane vuoto finché non riesco a includere } 10'000!) \rightarrow \dots$

Senza widening **dovremmo arrivare a 10'000 per avere una differenza in  $x_4$** . Allora si introduce il widening: si può mettere anche solo in punti precisi, per esempio nei while, per evitare di perdere precisione in quei punti dove non è necessario. Si può anche stabilire una soglia, per cui se un'iterazione sta andando avanti da troppo tempo si passa al widening per accelerare.

Qui, il punto critico è il punto di ingresso nel while  $x_2$ . Immaginiamo di essere andati avanti qualche altra iterazione:

$$\left\{ \begin{array}{l} X_1 = [1, 1] \\ X_2 = [1, 5] \cup [1, 6] \\ X_3 = [2, 6] \\ X_4 = \emptyset \end{array} \right.$$

Il widening mi porta a dire che  $x_2 = [1, 5] \cup [1, 6]$  va a  $[1, +\infty]$ : il lowebound viene lasciato stare, dato che è stabile, e l'upperbound viene allargato.

Andando avanti con il widening applicato in questo punto otteniamo

$$\left\{ \begin{array}{l} X_1 = [1, 1] \\ X_2 = [1, +\infty] \\ X_3 = [2, +\infty] \\ X_4 = \emptyset \end{array} \right. \rightarrow \left\{ \begin{array}{l} X_1 = [1, 1] \\ X_2 = [1, 9999] \\ X_3 = [2, +\infty] \\ X_4 = \emptyset \end{array} \right. \rightarrow \left\{ \begin{array}{l} X_1 = [1, 1] \\ X_2 = [1, 9999] \\ X_3 = [2, +10000] \\ X_4 = \emptyset \end{array} \right. \rightarrow \left\{ \begin{array}{l} X_1 = [1, 1] \\ X_2 = [1, 9999] \\ X_3 = [2, +10000] \\ X_4 = [+10000, +10000] \end{array} \right.$$

In questo caso, avendo una sola variabile e avendo la guardia che descrive un'informazione che ha a che fare con gli intervalli, ottengo che non ho perso informazione 😊 **Così ho fatto 2 passi anziché 10'000**.

I problemi si pongono quando abbiamo più variabili, cioè nella gran parte dei casi! Pensando alla modifica

```

x := 1;
1:
  while x < 10000 do
2:
  x := x + 1
3:   y := y + 1
  od;
4:

```

Succederebbe che il widening si applicherebbe anche a  $y$  ... ma non ho nessuna guardia che mi fa raffinare di nuovo l'informazione!

$$\left\{ \begin{array}{l} X_1 = [1, 1] \\ X_2 = [1, 9999] \leftarrow [1, +\infty] \\ X_3 = [2, +10000] \leftarrow \\ X_4 = [+10000, +10000] \end{array} \right.$$

## Il narrowing

Il narrowing, che non vediamo, è il meccanismo che poi mi permetterebbe di risistemare la  $y$ . Praticamente, nell'ordine del calcolo, guarderebbe se la  $y$  aveva un valore definito prima del widening e in caso affermativo capisce se è il caso di riportare quel valore definito al posto di  $\infty$ .

Con queste informazioni possiamo determinare, ad esempio, se c'è overflow o meno per  $x$ :

```

x := 1;
1: {x = 1} ←
  while x < 10000 do
2: {x ∈ [1, 9999]} ←
  x := x + 1 ← no overflow
3: {x ∈ [2, +10000]} ←
  od;
4: {x = 10000} ←

```

## ALTRÉ ANALISI NON DISTRIBUTIVE

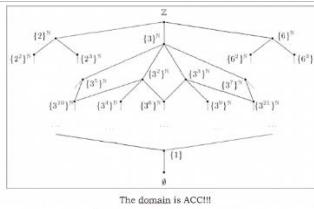
- **Congruenze**

Mi permettono di determinare delle regolarità nel resto della divisione; sono dei punti (buchi), che hanno una certa regolarità – come, appunto, essere modulo di qualcosa o  $3 + 2N$ .

Il dominio di tutti questi oggetti è un dominio astratto che posso formalizzare con connessioni di Galois e sticazzi e mazzi.

- **Potenze naturali**

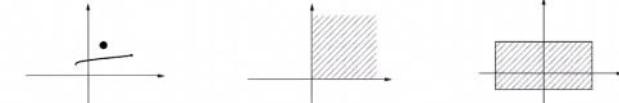
Sono addirittura ACC, perché gli insiemi più grandi hanno esponenti più piccoli or smth.



- **Domini non relazionali**

ovvero che parlano di un punto in maniera indipendente da tutti gli altri e parlano di una singola variabile

- Costanti
- Segni
- Intervalli
- Congruenze semplici
- Analisi delle potenze



Constant Propagation

$$X_i = c_i$$

[Kildall73]

Signs

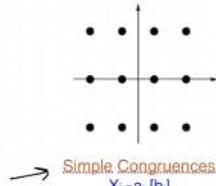
$$X_i \geq 0 \wedge X_i \leq 0$$

[CC76]

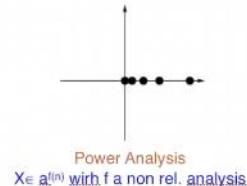
Intervals

$$X_i \in [a_i; b_i]$$

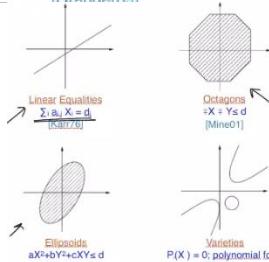
[CC76]



Simple Congruences  
 $X_i \equiv a_i \pmod{b_i}$   
[Granacher99]



Power Analysis  
 $X \in a^{f(n)}$  with f a non rel. analysis  
[Mastroeni01]



Ellipsoids  
 $aX_1^2 + bY_2^2 + cXY_1 \leq d$   
[Feret 2004]



Varieties  
 $P(X) = 0$ ; polynomial form  
[SAS04]

- **Domini relazionali**

Mettono in relazione le diverse variabili

- Equazioni lineari
- Ottagoni
- Ellisse
- Poliedro
- Etc.



Linear Equalities  
 $\sum a_{ij} X_j = d$   
[Karp70]



Octagons  
 $|X_j| \leq d$   
[Mineo01]

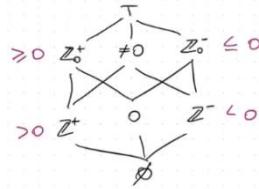


Polyhedra  
 $\sum a_{ij} X_j \leq d$   
[CH78]

## RIASSUNTO: INTERPRETAZIONE ASTRATTA

### 1. Costruzione dell'analisi

*Individuare gli elementi da osservare.  
ad esempio, nei segni, possiamo prendere →*



*Descrivere le operazioni astratte*

In questo caso la semantica delle espressioni.  
Per esempio, per la moltiplicazione sarebbe

*	T	Z_0^-	Z_0^+	Z^-	Z^+	0
T	T	T	T	T	T	T
Z_0^-	T	Z_0^+	Z_0^-	Z_0^+	Z_0^-	0
Z_0^+	T	Z_0^-	Z_0^+	Z_0^-	Z_0^+	0
Z^-	T	Z_0^+	Z_0^-	Z_0^+	Z_0^-	0
Z^+	T	Z_0^-	Z_0^+	Z_0^-	Z_0^+	0
0	T	0	0	0	0	0

$$[\![e_1 \square e_2]\!]^D = [\![e_1]\!]^D \square^* [\![e_2]\!]^D$$

*operazione  $\square$   
spaziosa sul dominio  
astratto  $S$*

*Descrivere la semantica dei comandi (abstract edge effect)*

$D \sqcap [\![e]\!]^D$  sarebbe il lowerbound fra le due cose; è abbastanza universale come cosa, perché in questo modo se  $[\![e]\!]^D$  produce una descrizione utile prendo quello, mentre se invece mi dà top posso prendere quello che avevo prima. (or smth)

$$[\![;]\!]^D = D \quad [\![x \leftarrow e]\!]^D = D[x \mapsto [\![e]\!]^D]$$

$$[\![\text{input}(x)]\!]^D = D[x \mapsto T]$$

$$[\![\text{zero}(e)]\!]^D = \begin{cases} \perp & \alpha(o) \notin [\![e]\!]^D \\ D \sqcap [\![\neg e]\!]^D & \text{altrimenti} \end{cases}$$

$$[\![\text{Non zero}(e)]\!]^D = \begin{cases} \perp & \alpha(o) = [\![e]\!]^D \xrightarrow{\text{approssimazione in } S \text{ di } 0} (\text{se } \alpha(o) = 0) \\ D \sqcap [\![e]\!]^D & \text{altrimenti} \end{cases}$$

*questo  
possibile  
essere se 0  
è esattamente  
rappresentato  
nel dominio  
astratto*

### 2. Calcolo dell'analisi

*Definisco il sistema di equazioni*  
(promemoria: noi mettiamo l'uguale anziché il  $\equiv$  perché di fatto noi prendiamo sempre il least upper bound del valore precedente rispetto a quello nuovo.

*Svolgo le equazioni*

$$\left\{ \begin{array}{l} S[\text{entry}] \supseteq D_T \xrightarrow{\text{informazione iniziale}} \text{equivale a dire} \\ \qquad \qquad \qquad x \mapsto T \text{ che non abbiamo info sulle variabili} \\ S[n] \supseteq [\![\text{lab}]\!]^* S[\text{a}] \quad \forall R = (n, \text{lab}, n) \\ \qquad \qquad \qquad \uparrow \end{array} \right.$$