Alma Mater Studiorum · Università di Bologna

Scuola di Scienze

Corso di Laurea in Informatica

Sulla Riscrittura dei Termini in Presenza di Effetti Probabilistici

Relatore:
Chiar.mo Prof.
Ugo dal Lago

Presentata da:
Stefano Volpe

I sessione
Anno Accademico 2022/23

A Chia, per avermi tenuto a galla

A Lalla, per questi quindici anni

Take care of the sense and the sounds will take care of themselves.

Tu bada al senso, che i suoni sapranno badare a se stessi.

La Duchessa, da Le avventure di Alice nel paese delle meraviglie, *di Lewis Carroll*

Sommario

Nel contesto intersezionale a cavallo fra riscrittura dei termini e teoria delle probabilità, vengono presentati gli ordini probabilisticamente monotoni, una tecnica per costurire sistemi astratti di riduzione probabilistici (come definiti da Avanzini et al. [2]) a partire da sistemi astratti di riduzione. Vengono definite due varianti di questo metodo. Per entrambe, viene dimostrato che gli ordini probabilisticamente monotoni così generati sono quasi certamente terminanti forti (SAST, *Strong Almost-Sure Terminating*), e viene mostrato un esempio pratico di costruzione di tali ordini. Viene infine provato che esistono ordini di percorso lessicografici (LPO, *Lexicographic Path Orders*) che non soddisfano i prerequisiti necessari per indurre ordini probabilisticamente monotoni di alcun tipo.

Indice

Indice dei programmi

1 Prerequisiti

Questo capitolo introduttivo pone essenziali fondamenta per la trattazione che gli seguirà. Una volta chiarito il contesto disciplinare in cui la riscrittura probabilistica dei termini si colloca, vengono rievocati in modo particolarmente rigoroso concetti preliminari di algebra universale e teoria delle probabilità.

1.1 Introduzione informale

1.1.1 La riscrittura dei termini

La riscrittura dei termini, fondata sulla logica equazionale, affianca alle identità di quest'ultima delle regole di riscrittura direzionate. Non si limita quindi a esprimere il fatto che un termine t sia equivalente a un termine u, ma può anche rappresentare la nozione che t sia "riscrivibile" in un passo di computazione come u. A cavallo fra l'informatica teorica, l'algebra universale, la dimostrazione automatica di teoremi e la programmazione funzionale, la riscrittura dei termini è interessata ai sistemi di calcolo convergenti (anche se potenzialmente non-deterministici) e alle tecniche per individuarli.

Anche se loro predecessori erano già noti agli inizi del XX secolo, storicamente, i sistemi di riscrittura dei termini "propriamente detti" vedono la luce con i lavori di Evans [5] (1951) prima, e di Knuth e Bendix [6] (1970) poi. Inizialmente, lo scopo inteso era quello di produrre sistemi di riscrittura di termini cosiddetti "canonici", con lo scopo di provare uguaglianze in determinate teorie equazionali. Verso la fine degli anni 70, l'emergere dei sistemi equazionali come strumenti per specificare tipi di dato astratto avrebbe ravvivato l'entusiasmo per la disciplina.

In questa esposizione rimarremo, per quanto possibile, fedeli al modello di riscrittura dei termini delineato da Baader e Nipkow [3]. Per applicarlo a un sistema di calcolo è sufficiente riconoscere in esso:

- una segnatura, e cioè una funzione che attribuisca un'arietà a ogni simbolo di funzione;
- un sistema di riscrittura probabilistico, e cioè un insieme di regole di riscrittura (opportunamente generalizzate tramite uso di variabili) che esprimano i possibili passi di computazione.

Nella nostra esposizione (dedicata invece al caso probabilistico), avremo cura di formalizzare i vincoli che questi oggetti devono rispettare.

Esempio 1.1.1 (Somme fra numeri naturali)**.** *Se nel nostro modello vogliamo considerare solo numeri naturali e somme fra essi, una possibile segnatura* Σ *potrebbe essere:*

$$\Sigma(0) = 0$$

$$\Sigma(S) = 1$$

$$\Sigma(+) = 2$$

Qui, S *è la funzione successore. Si noti come in algebra universale i simboli di costante siano semplicemente visti come simboli di funzione 0-ari.*

Esempio 1.1.2 (Somme fra numeri naturali (continua))**.** *Si considerino i termini costruiti in base alla segnatura dell'esempio di cui sopra. Usando* X *e* Y *come variabili che spaziano sull'insieme di tali termini, possiamo specificare le due regole che permettono di valutarli come espressioni aritmetiche:*

$$X + 0 \to X$$

$$X + S(Y) \to S(X + Y)$$

Queste regole sono orientate: affinché la nostra computazione progredisca, esse indicano che devono essere applicate da sinistra verso destra.

1.1.2 Linguaggi funzionali probabilistici

I modelli di calcolo probabilistici non sono solo una bizzarria teorica, ma presentano utili applicazioni pratiche, che spaziano dagli algoritmi casuali alla crittografia. Nel caso di un generico linguaggio funzionale universale, un modo semplice di realizzare un'estensione probabilistica anch'essa universale è aggiungere un operatore di campionamento su una distribuzione di Bernoulli equa (e cioè quella del lancio di una moneta non truccata).

Esempio 1.1.3 (*Quicksort* casuale)**.** *Il* Quicksort *propriamente detto ha, nel caso peggiore, costo temporale asintotico* $\Theta(n^2)$*. La sua variante casuale ha costo temporale asintotico atteso* $\Theta(n\ log(n))$*. La nostra esposizione sarà costellata di diversi esempi in Standard ML [7], tutti focalizzati sulla semplicità anziché su prestazioni ottimali.*

Programma 1.1: *Quicksort* casuale (`rq-sort.sml`)

```
1  (* Generatore pseudocasuale *)
2  val rand = Random.rand (0, 0)
3
4  (* Estrae l'indice di un elemento a caso della lista *)
5  fun randomIndex l = Random.randRange (0, length l) rand
6
7  (* Separa l'elemento alla data posizione dal resto della lista *)
8  fun pick l index =
9    (List.nth (l, index), List.take (l, index) @ List.drop (l, index + 1))
10
11 (* Separa un elemento casuale dal resto della lista *)
12 fun randomPick l = pick l (randomIndex l)
13
14 (* Partiziona la lista usando un pivot casuale *)
15 fun randomPartition l =
16   let
17     val (pivot, rest) = randomPick l
18     val (left, right) = List.partition (fn x => x < pivot) rest
19   in
20     (left, pivot, right)
21   end
22
23 (* Ordina le sottoliste sinistra e destra, poi vi frappone il pivot *)
24 fun sortPartitions (left, pivot, right) = rqSort left @ [pivot] @ rqSort right
25
26 (* Quicksort casuale (implementazione non in loco) *)
27 and rqSort [] = []
28   | rqSort l = sortPartitions (randomPartition l);
```

Ai fini del nostro studio, Standard ML non fungerà solo da linguaggio-oggetto in cui esprimere programmi probabilistici di cui provare proprietà utili: lo useremo anche come meta-linguaggio per formalizzare gli stessi concetti esposti, sulla traccia di Baader e Nipkow [3], da cui riprenderemo anche alcuni frammenti di implementazione. Tutti i materiali sono disponibili presso:

`https://codeberg.org/Foxy/probabilistic-lexicographic-path-orders`

1.1.3 La riscrittura probabilistica dei termini

Il modello di riscrittura probabilistica dei termini su cui questo studio si basa è sostanzialmente analogo a quello presentato da Avanzini et al. [2], a sua volta ispirato a Bournez e Garnier [4], in contrasto a quello di Agha et al. [1]. Lo svolgersi della riscrittura è concepito come un processo decisionale di Markov. Se, da un lato, a ogni passo si ha una nondeterministica selezione della seguente applicazione corretta di regola, dall'altro lo stesso fatto che ciascuna regola possa avere più esiti possibili contribuisce con un proprio grado di incertezza.

Esempio 1.1.4 (Passeggiata aleatoria)**.** *Si consideri la seguente segnatura per termini che rappresentano numeri naturali:*

$$\Sigma(0) = 0$$

$$\Sigma(S) = 1$$

A partire da essa, usando X come variabile per un qualsiasi termine basato su di essa, possiamo specificare una regola di riscrittura che trasforma un numero nel suo successore con $\frac{1}{2}$ di probabilità, e nel suo precedente con il restante $\frac{1}{2}$ di probabilità:

$$S(X) \to \{\!\{\frac{1}{2} : X; \frac{1}{2} : S(S(X))\}\!\}$$

Questa regola di riscrittura, anche se presa singolarmente, forma già un corretto sistema probabilistico di riscrittura dei termini.

1.2 Concetti preliminari

1.2.1 Algebra universale

Termini

Come da consueto, i termini con cui lavoreremo saranno costruiti usando simboli di funzione e variabili. Per rappresentare l'arietà di ciascun simbolo di funzione, introduciamo il concetto di segnatura.

Definizione 1.2.1 (Segnatura)**.** Sia Φ un insieme di simboli di funzione. Una segnatura per Φ è una qualsiasi funzione

$$\Sigma_\Phi \quad : \quad \Phi \to \mathbb{N}$$

Intuitivamente, una segnatura associa a ciascun simbolo di funzione la sua arietà. Si noti che nella nostra trattazione l'insieme dei numeri naturali $\mathbb{N}$ include sempre 0. Possiamo inoltre partizionare i simboli di funzione in base alla loro arietà.

Definizione 1.2.2 (Insieme degli elementi n-ari)**.** Siano Σ_Φ una segnatura per Φ e $n \in \mathbb{N}$. L'insieme $\Sigma_\Phi^{(n)}$ di tutti gli elementi n-ari di Φ è definito come segue:

$$\Sigma_\Phi^{(n)} := \{f \in \Phi \quad | \quad \Sigma_\Phi(f) = n\}$$

I simboli di costante sono chiaramente già contemplati nella definizione precedente:

Definizione 1.2.3 (Simbolo di costante)**.** Sia Σ_Φ una segnatura per Φ. Un simbolo di funzione $e \in \Phi$ si dice essere un simbolo di costante rispetto a Σ_Φ se e solo se

$$e \in \Sigma_\Phi^{(0)}$$

Possiamo ora procedere a combinare i simboli di funzione con le variabili per formare termini. Per evitare ambiguità, imponiamo che l'insieme dei simboli di funzione e quello delle variabili siano distinti.

Definizione 1.2.4 (Insieme dei termini)**.** Siano Σ_Φ una segnatura e X un insieme di variabili tali che $\Phi \cap X = \varnothing$. Definiamo per induzione $T(\Sigma_\Phi, X)$, l'insieme dei Σ_Φ-termini su X:

1. $X \subseteq T(\Sigma_\Phi, X)$;
2. $\forall n \in \mathbb{N} \quad \forall f \in \Sigma_\Phi^{(n)} \quad \forall t_1, \ldots, t_n \in T(\Sigma_\Phi, X) \quad f(t_1, \ldots, t_n) \in T(\Sigma_\Phi, X)$;
3. nient'altro è un Σ_Φ-termine su X.

Con la clausola base, abbiamo reso ogni variabile un termine; con quella induttiva, abbiamo reso ogni applicazione di un simbolo di funzione a dei termini (dove la arietà del simbolo è rispettata) essa stessa un termine. Ora che abbiamo specificato il rapporto di disgiunzione fra l'insieme dei simboli di funzione e quello delle variabili, non abbiamo più interesse a esplicitare pedantemente il primo ogni volta. Da qui in poi, useremo Σ per riferirci a una segnatura per un qualche insieme di simboli di funzione.

Baader e Nipkow [3], per esempio, partono da questa definizione per costruire il loro tipo `term` in Standard ML:

Programma 1.2: termini di un generico TRS (`traat/trs.ML`, 14-16)

```
1 type vname = string * int;
2
3 datatype term = V of vname | T of string * term list;
```

La definzione per induzione appena data può essere vista come una grammatica disambigua per i termini che ci permette di costruire da essi degli alberi di sintassi astratta. Inoltre, dato un termine, possiamo identificare univocamente una qualsiasi posizione all'interno del suo albero di sintassi astratta specificando l'unico cammino discendente che ci permette di raggiungerla a partire dalla radice.

Nella definizione che segue, usiamo $\mathbb{N}^*$ per indicare l'insieme di tutte le possibili stringhe costruite su $\mathbb{N}$, ed ϵ per indicare la stringa vuota.

Definizione 1.2.5 (Insieme delle posizioni)**.** Siano $T(\Sigma, X)$ l'insieme dei Σ-termini su X e $s \in T(\Sigma, X)$. Definiamo per induzione $\mathcal{P}os(s) \subseteq \mathbb{N}^*$, l'insieme delle posizioni del termine s:

$$\mathcal{P}os(s) = \begin{cases} \{\epsilon\} & \text{se } s \in X \\ \{\epsilon\} \cup \bigcup_{i=1}^{n} \{ip \mid p \in \mathcal{P}os(s_i)\} & \text{se } s = f(s_1, \ldots, s_n) \end{cases}$$

Nella definizione di cui sopra, ciascun cammino è codificato come una stringa: se n è l'i-esimo passo di una posizione, allora il l'i-esimo passo del cammino ad essa associato ha come destinazione il figlio di indice n del nodo corrente. L'insieme delle posizioni ci dà inoltre un'idea della "misura" di un dato termine.

Definizione 1.2.6 (Dimensione di un termine)**.** Siano $T(\Sigma, X)$ l'insieme dei Σ-termini su X e $s \in T(\Sigma, X)$. La dimensione del termine s è:

$$|s| \quad := \quad |\mathcal{P}os(s)|$$

Avendo le posizioni, possiamo ora estrarre da un termine i suoi sottotermini.

Definizione 1.2.7 (Sottotermine)**.** Siano $T(\Sigma, X)$ l'insieme dei Σ-termini su X, $s \in T(\Sigma, X)$ e $p \in \mathcal{P}os(s)$. Per induzione su p definiamo $s|_p$, il sottotermine di s in posizione p:

$$s|_p = \begin{cases} s & \text{se } p = \epsilon \\ s_i|_q & \text{se } p = iq \text{ (e quindi } s = f(s_1, \ldots, s_n)) \end{cases}$$

Sostituzioni e contesti

Un termine può essere modificato rimpiazzando uno dei suoi sottotermini con un altro.

Definizione 1.2.8 (Termine ottenuto per rimpiazzamento)**.** Siano $T(\Sigma, X)$ l'insieme dei Σ-termini su X, $s, t \in T(\Sigma, X)$ e $p \in \mathcal{P}os(s)$. Per induzione su p definiamo $s[t]_p$, il termine ottenuto da s rimpiazzando il suo sottotermine in posizione p con il termine t:

$$s[t]_p = \begin{cases} t & \text{se } p = \epsilon \\ f(s_1, \ldots, s_i[t]|_q, \ldots, s_n) & \text{se } p = iq \text{ (e quindi } s = f(s_1, \ldots, s_n)) \end{cases}$$

Molto spesso, saremo interessati a rimpiazzare le variabili presenti in un dato termine, e quindi a sapere quali di esse effettivamente occorrano in esso.

Definizione 1.2.9 (Insieme delle variabili occorrenti in un termine)**.** Siano $T(\Sigma, X)$ l'insieme dei Σ-termini su X e $s \in T(\Sigma, X)$. L'insieme delle variabili occorrenti in s è:

$$\mathcal{V}ar(s) := \{x \in X \mid \exists p \in \mathcal{P}os(s) \quad s|_p = x\}$$

Sempre con l'intento di vincolarci al rimpiazzamento di termini che siano anche variabili, sarà importante sapere se un sottotermine in una posizione data sia effettivamente una variabile o meno.

Definizione 1.2.10 (Posizione di variabile)**.** Siano $T(\Sigma, X)$ l'insieme dei Σ-termini su X, $s \in T(\Sigma, X)$ e $p \in \mathcal{P}os(s)$. p è una posizione di variabile se e solo se

$$s|_p \in X$$

Le sostituzioni permettono di rimpiazzare uniformemente un qualsiasi numero finito di variabili all'interno di un termine. Nel seguito, costruiremo i nostri termini su insiemi di variabili sì infiniti, ma numerabili, e cioè di cardinalità $\aleph_0$.

Definizione 1.2.11 (Sostituzione di variabili)**.** Siano V un insieme di variabili tale che $|V| = \aleph_0$ e $T(\Sigma, V)$ l'insieme dei Σ-termini su V. Una $T(\Sigma, V)$-sostituzione di variabili σ è una qualsiasi funzione

$$\sigma : V \to T(\Sigma, V) \quad , \quad |\{x \in V \mid x \neq \sigma(x)\}| < \aleph_0$$

Le variabili che una sostituzione non mappa in sé stesse (e sulle quali quindi la sostituzione di variabili effettuamente agisce) costituiscono il dominio di detta sostituzione.

Definizione 1.2.12 (Dominio di una sostituzione di variabili)**.** Sia σ una $T(\Sigma, V)$-sostituzione di variabili. Il dominio di σ, $\mathcal{D}om(\sigma)$, è

$$\mathcal{D}om(\sigma) := \{x \in V \mid x \neq \sigma(x)\}$$

Il concetto di sostituzione di variabili è facilmente estendibile ai termini.

Definizione 1.2.13 (Sostituzione per termini)**.** Sia σ una $T(\Sigma, X)$-sostituzione di variabili. Per induzione su $t \in T(\Sigma, X)$ definiamo $\hat{\sigma}$, la $T(\Sigma, X)$-sostituzione per termini basata su σ:

$$\hat{\sigma}(s) = \begin{cases} \sigma(s) & \text{se } s \in V \\ f(\sigma(s_1), \ldots, \sigma(s_n)) & \text{se } s = f(s_1, \ldots, s_n) \end{cases}$$

Quando parliamo di "dominio di una sostituzione di termini", ovviamente facciamo riferimento al dominio della sostituzione di variabili sottostante.

Definizione 1.2.14 (Dominio di una sostituzione di termini)**.** Siano σ una $T(\Sigma, X)$-sostituzione di variabili e $\hat{\sigma}$ la $T(\Sigma, X)$-sostituzione di termini basata su σ. Il dominio di $\hat{\sigma}$, $\mathcal{D}om(\hat{\sigma})$, è:

$$\mathcal{D}om(\hat{\sigma}) := \mathcal{D}om(\sigma)$$

Spesso, tornerà utile considerare l'insieme di tutte le possibili (Σ, V)-sostituzioni di termini.

Definizione 1.2.15 (Insieme delle sostituzioni di termini)**.** Siano V un insieme di variabili tale che $|V| = \aleph_0$ e $T(\Sigma, V)$ l'insieme dei Σ-termini su V. L'insieme delle (Σ, V)-sostituzioni di termini, $\mathcal{S}ub(T(\Sigma, V))$, è:

$$\mathcal{S}ub(T(\Sigma, V)) := \{\hat{\sigma} : T(\Sigma, V) \to T(\Sigma, V) | \hat{\sigma} \text{ è una } (\Sigma, V)\text{-sostituzione di termini}\}$$

Quando sarà chiaro dal contesto, potremmo limitarci a riferirci alla sostituzioni di termini come "sostituzioni", e evitare di denominarle facendo uso di accenti circonflessi.

L'ultimo concetto di algebra universale di cui necessitiamo è quello di contesto.

Definizione 1.2.16 (Contesto)**.** Sia V un insieme di variabili tale che $|V| = \aleph_0$ e $\Box \notin V$. Un contesto C è un qualsiasi Σ-termine su $V \cup \{\Box\}$ tale che

$$|\{p \in \mathcal{P}os(C) \mid C|_p = \Box\}| = 1$$

Definizione 1.2.17 (Popolazione di un contesto)**.** Sia C un contesto per Σ-termini su V in cui $\Box$ occorre in posizione p e sia s un Σ-termine su V. La popolazione del contesto C con s, $C[s]$, è definita come

$$C[s] := C[s]_p$$

Un contesto funge insomma da termine contenente un singolo "buco" $\Box$, pronto a essere riempito tramite sostituzione. Avremo modo di fare uso dell'insieme di tutti i contesti per Σ-termini su V.

Definizione 1.2.18 (Insieme dei contesti)**.** Siano V un insieme di variabili tale che $|V| = \aleph_0$ e $T(\Sigma, V)$ l'insieme dei Σ-termini su V, con $\Box \notin V$. L'insieme dei contesti per Σ-termini su V, $\mathcal{C}on(T(\Sigma, V))$, è:

$$\mathcal{C}on(T(\Sigma, V)) := \{C \in T(\Sigma, V \cup \{\Box\}) \mid C \text{ è un contesto}\}$$

1.2.2 Probabilità

I prerequisiti di probabilità di cui necessitiamo poggiano sul concetto di multinsieme finito.

Definizione 1.2.19 (Multinsieme finito)**.** Sia A un insieme finito. Un multinsieme finito su A è una qualsiasi funzione

$$M : A \to \mathbb{N}$$

D'ora in poi, quando parleremo di multinsiemi, ci concederemo la libertà di sottointenderne la finitezza. Intuitivamente, un multinsieme mappa ogni suo elemento nel numero di "copie" di tale elemento contenute nel multinsieme stesso. Possiamo unire una quantità finita di multinsiemi sullo stesso insieme di partenza semplicemente sommando, per ciascun elemento, i rispettivi numeri di copie.

Definizione 1.2.20 (Unione di un numero finito di multinsiemi finiti)**.** Siano A un insieme finito e I un insieme finito tale che

$$\forall i \in I \quad M_i \text{ è un multinsieme finito su } A$$

In questo caso:

$$\forall a \in A \quad (\biguplus_{i \in I} M_i)(a) := \sum_{i \in I} M_i(a)$$

Anche quando sommiamo tutti gli elementi di un multinsieme dobbiamo tener conto dei numeri delle copie di ciascuno di essi.

Definizione 1.2.21 (Somma di un multinsieme finito rispetto a una funzione)**.** Sia M un multinsieme finito su A e sia $f : A \to \mathbb{R}$. La somma di M rispetto a f è

$$\sum_{a \in M} f(a) := \sum_{a \in A} M(a) f(a)$$

Un multinsieme di facile definizione è quello vuoto, che non contiene alcuna copia di alcun elemento.

Definizione 1.2.22 (Multinsieme vuoto)**.** Sia A un insieme finito. Il multinsieme vuoto su A, $\varnothing_A$, è definito come

$$\forall a \in A \quad \varnothing_A(a) := 0$$

Possiamo costruire multinsiemi anche in altri modi, per esempio partendo da un insieme.

Definizione 1.2.23 (Multinsieme finito da un insieme finito di indici)**.** Siano A e I insiemi finiti tali che

$$\forall i \in I \quad a_i \in A$$

In questo caso, $M = \{\!\{a_i \mid i \in I\}\!\}$, è il multinsieme finito su A tale che:

$$\forall a \in A \quad M(a) := |\{i \in I \mid a_i = a\}|$$

In Standard ML, un modo pigro di rappresentare un multinsieme è tramite una lista che ne contiene tutte le ipotetiche copie nel giusto numero e senza un ordine prestabilito:

Programma 1.3: multinsiemi come liste (`multidistributions.sml`, 1-2)

```
1 (* Assumendo di usare solo multinsiemi finiti da insiemi finiti di indici *)
2 type 'a multiset = 'a list;
```

Nel caso finito dei numeri naturali da 1 a n inclusi, possiamo anche orientarci verso una scrittura più sintetica.

Definizione 1.2.24 (Multinsieme finito da un insieme finito di indici)**.** Siano A un insieme finito e $n \in \mathbb{N}$ tali che

$$\forall i \in \{1, \ldots, n\} \quad a_i \in A$$

In questo caso:

$$\{\!\{a_1, ..., a_n\}\!\} := \{\!\{a_i \mid i \in \{1, \ldots, n\}\}\!\}$$

Così rappresentati, i multinsiemi si prestano molto più facilmente all'operazione di unione vista sopra.

Teorema 1.2.1 (Formula dell'unione di un numero finito di multinsiemi finiti)**.** *Siano I e A due insieme finiti tali che*

$$\forall i \in I \quad \{\!\{a_{i,j} \mid j \in J_i\}\!\} \in \mathbb{N}^A$$

Allora,

$$\biguplus_{i \in I} \{\!\{a_{i,j} \mid j \in J_i\}\!\} = \{\!\{a_{i,j} \mid i \in I \wedge j \in J_i\}\!\}$$

Dimostrazione. Ovvio per Definizione 1.2.20 e Definizione 1.2.23. □

Ci apprestiamo ora a introdurre il concetto di sottomultidistribuzione, che per brevità abbrevieremo come "multidistribuzione". Un esempio di zucchero sintattico che ci concederemo sarà quello di usare $p : a$ per indicare coppie ordinate (p, a), con la connotazione intesa di usare p come una probabilità e a come un esito.

Definizione 1.2.25 ((Sotto)multidistribuzione)**.** Una (sotto)multidistribuzione μ su un insieme A è un multinsieme su $[0, 1] \times A$ tale che:

$$\sum_{p:a \in \mu} p \leq 1$$

Programma 1.4: multidistribuzioni (`multidistributions.sml`, 4-5)

```
1 (* Assumendo reali in [0, 1] e di somma <= 1 *)
2 type 'a multidistribution = (real * 'a) multiset;
```

Volendo assegnare un nome alla quantità vincolata dalla precedente definizione, potremmo chiamarla "probabilità totale" della multidistribuzione in questione.

Definizione 1.2.26 (Probabilità totale di una multidistribuzione)**.** La probabilità totale di una multidistribuzione μ è definita come

$$|\mu| := \sum_{p:a\in\mu} p$$

Il calcolo della probabilità totale di una data multidistribuzione in Standard ML è molto semplice.

Programma 1.5: probabilità totale (`multidistributions.sml`, 7-11)

```
1 (* real list -> real *)
2 fun sum l = foldr (fn (x, y) => x + y) 0 l
3
4 (* 'a multidistribution -> real *)
5 fun totalProbability mu = sum (map (fn (p, _) => p) mu)
```

Definiamo l'insieme di tutte le multidistribuzioni su un dato insieme.

Definizione 1.2.27 (Insieme delle multidistribuzioni)**.** L'insieme delle multidistribuzioni su un dato insieme A, $\mathcal{M}_{\leq 1}(A)$, è

$$\mathcal{M}_{\leq 1}(A) := \{M : [0,1] \times A \to \mathbb{N} \mid M \text{ è una multidistribuzione su } A\}$$

Se una funzione è applicabile agli elementi di un dato insieme, potremmo volerla applicare simultaneamente a un intera multidistribuzione su tale insieme.

Definizione 1.2.28 (Sollevamento di una funzione alle multidistribuzioni)**.** Sia $f : A \to B$. Definiamo il sollevamento di f alle multidistribuzioni come l'omonima funzione $f : \mathcal{M}_{\leq 1}(A) \to \mathcal{M}_{\leq 1}(B)$ tale che, se I è un insieme tale che $|I| \leq \aleph_0$ e

$$\forall i \in I \quad p_i : a_i \in [0,1] \times A$$

allora:

$$f(\{\{p_i : a_i \mid i \in I\}\}) := \{\{p_i : f(a_i) \mid i \in I\}\}$$

Le multidistribuzioni la cui probabilità totale raggiunge il limite superiore di cui sopra sono dette "proprie".

Definizione 1.2.29 (Multidistribuzione propria)**.** Una multidistribuzione μ si dice "propria" se e solo se

$$|\mu| = 1$$

Come sopra, definiamo ora l'insieme di tutte le multidistribuzioni proprie su un dato insieme.

Definizione 1.2.30 (Insieme delle multidistribuzioni proprie)**.** L'insieme delle multidistribuzioni proprie su un dato insieme A, $\mathcal{M}(A)$, è

$$\mathcal{M}(A) := \{M : [0,1] \times A \to \mathbb{N} \mid M \text{ è una multidistribuzione propria su } A\}$$

Una multidistribuzione è facilmente moltiplicabile per uno scalare: è sufficiente moltiplicare per lo scalare in questione tutte le probabilità coinvolte.

Definizione 1.2.31 (Moltiplicazione scalare di una multidistribuzione)**.** Sia $\{\!\{q_i : a_i \mid i \in I\}\!\}$ una multidistrubuzione su A e sia $p \in \mathbb{R}$. La moltiplicazione scalare di $\{\!\{q_i : a_i \mid i \in I\}\!\}$ per p è definita come:

$$p \cdot \{\!\{q_i : a_i \mid i \in I\}\!\} := \{\!\{p \cdot q_i : a_i \mid i \in I\}\!\}$$

Prevedere la probabilità totale di una combinazione lineare di multidistribuzioni è semplice.

Teorema 1.2.2 (Probabilità totale di una combinazione lineare di multidistribuzioni)**.** *Siano I e A insiemi finiti tali che*

$$\forall i \in I \quad (p_i \in \mathbb{R} \wedge \mu_i \in \mathcal{M}_{\leq 1}(A))$$

Allora, vale la seguente uguaglianza:

$$|\biguplus_{i \in I} p_i \cdot \mu_i| = \sum_{i \in I} p_i \cdot |\mu_i|$$

Dimostrazione. Per ogni $i \in I$, sia $\mu_i = \{\!\{q_{i,j} : a_{i,j} \mid i \in I\}\!\}$. Usando il Teorema 1.2.1, si ha:

$$|\biguplus_{i\in I} p_i \cdot \mu_i| = |\biguplus_{i\in I} p_i \cdot \{\!\{q_{i,j} : a_{i,j} \mid i \in I\}\!\}| = |\biguplus_{i\in I} \{\!\{p_i \cdot q_{i,j} : a_{i,j} \mid i \in I\}\!\}| =$$

$$= |\{\!\{p_i \cdot q_{i,j} : a_{i,j} \mid i \in I \wedge j \in J_i\}\!\}| = \sum_{i\in I}\sum_{j\in J_i} p_i \cdot q_j = \sum_{i\in I} p_i \cdot \sum_{j\in J_i} q_j =$$

$$= \sum_{i\in I} p_i \cdot |\mu_i|$$

□

Come intuibile, possiamo dimostrare che l'insieme delle multidistribuzioni su un dato insieme sia chiuso rispetto alle combinazioni sottoconvesse.

Corollario 1.2.1 (Chiusura delle multidistribuzioni rispetto alle combinazioni sottoconvesse)**.** *Siano I e A insiemi finiti tali che*

$$(\forall i \in I \quad (p_i \in \mathbb{R} \wedge \mu_i \in \mathcal{M}_{\leq 1}(A))) \wedge \sum_{i\in I} p_i \leq 1$$

Allora,

$$\biguplus_{i\in I} p_i \cdot \mu_i \in \mathcal{M}_{\leq 1}(A)$$

Dimostrazione. Per la Definizione 1.2.25, mi riduco a dimostrare

$$|\biguplus_{i\in I} p_i \cdot \mu_i| \leq 1$$

che, per il Teorema 1.2.2, è equivalente a

$$\sum_{i\in I} p_i \cdot |\mu_i| \leq 1$$

Siccome sempre per la Definizione 1.2.25 si dà il caso che

$$\forall i \in I |\mu_i| \leq 1$$

il membro di sinistra è maggiorato da $\sum_{i \in I} p_i$, che è minore o uguale a 1 per ipotesi. Anche la disuguaglianza originale è quindi dimostrata. □

Nel caso particolare delle multidistribuzioni su numeri reali, possiamo parlare di valore atteso.

Definizione 1.2.32 (Valore atteso di una multidistribuzione su numeri reali)**.** Sia $\mu \in \mathcal{M}_{\leq 1}(\mathbb{R})$. Il valore atteso della multidistribuzione sui numeri reali μ, $\mathbb{E}(\mu)$, è definito come:

$$\mathbb{E}(\mu) := \sum_{p:a \in \mu} p \cdot a$$

2 La riscrittura probabilistica dei termini

In questo capitolo, presentiamo la riscrittura probabilistica dei termini come un processo decisionale di Markov, sulle orme di Bournez e Garnier [4]. Nessun risultato originale viene presentato: ci si limita a un'esposizione rigorosa che sia anche funzionale al resto dello studio. Per aiutare la lettrice e il lettore, il caso probabilistico sarà sempre preceduto da quello non-probabilistico, anche qualora quest'ultimo non sia necessario ai nostri scopi.

2.1 Sistemi di riduzione astratti probabilistici

Presupposto per parlare di riscrittura dei termini è aver chiaro il concetto di sistemi di riduzione astratti, che prescinde dallo studio dei termini lavorando su oggetti arbitrari.

Definizione 2.1.1 (Sistema di riduzione astratto)**.** Sia A un insieme. $\to$ è un sistema di riduzione astratto su A se e solo se

$$\to \subseteq A^2$$

Quando passiamo allo scenario probabilistico, usiamo le multidistribuzioni. Facciamo inoltre uso dell'abbreviazione PARS (*Probabilistic Abstract Reduction System*).

Definizione 2.1.2 (Sistema di riduzione astratto probabilistico)**.** Sia A un insieme. $\rightarrow$ è un sistema di riduzione astratto probabilistica su A se e solo se

$$\rightarrow \subseteq A \times \mathcal{M}(A)$$

I dati che non possono essere ridotti prendono il nome di "forme normali". La loro definizione resta sostanzialmente inalterata nel caso probabilistico.

Definizione 2.1.3 (Forma normale di un sistema di riduzione astratto)**.** Siano $\rightarrow$ un sistema di riduzione astratto su A e $a \in A$. Si dice che a è una forma normale in $\rightarrow$ quando

$$\nexists b \in A \quad a \rightarrow b$$

Definizione 2.1.4 (Forma normale di un sistema di riduzione astratto probabilistico)**.** Siano $\rightarrow$ un sistema di riduzione astratto probabilistico su A e $a \in A$. Si dice che a è una forma normale in $\rightarrow$ quando

$$\nexists \mu \in \mathcal{M}(A) \quad a \rightarrow \mu$$

Le forme normali rappresentano il risultato finale di un processo di riduzione, e il loro insieme è quindi un oggetto di rilievo.

Definizione 2.1.5 (Insieme delle forme normali di un sistema di riduzione astratto)**.** Sia $\rightarrow$ un sistema di riduzione astratto su A. L'insieme delle forme normali di $\rightarrow$, $NF_{\rightarrow}$, è definito come

$$NF_{\rightarrow} := \{a \in A \mid \nexists b \in A \quad a \rightarrow b\}$$

Definizione 2.1.6 (Insieme delle forme normali di un sistema di riduzione astratto probabilistico)**.** Sia $\rightarrow$ un sistema di riduzione astratto probabilistico su A. L'insieme delle forme normali di $\rightarrow$, $NF_{\rightarrow}$, è definito come

$$NF_{\rightarrow} := \{a \in A \mid \nexists \mu \in \mathcal{M}(A) \quad a \rightarrow \mu\}$$

A partire da un PARS su A, possiamo costruire un ARS che espliciti il comportamento del primo oggetto su un'arbitraria multidistribuzione su A. In

virtù del Corollario 1.2.1, la seguente definizione riguarda davvero un ARS su $\mathcal{M}_{\leq 1}(A)$.

Definizione 2.1.7 (Relazione di riduzione probabilistica)**.** Sia $\to$ un PARS su A. La relazione di riduzione probabilistica di $\to$, $\xrightarrow{\mathcal{M}}$, è il più piccolo ARS su $\mathcal{M}_{\leq 1}(A)$ che rispetta le seguenti regole:

1. $\forall a \in NF_{\to} \quad \{\!\{1 : a\}\!\} \xrightarrow{\mathcal{M}} \varnothing$;
2. $\forall (a, \mu) \in \to \quad \{\!\{1 : a\}\!\} \xrightarrow{\mathcal{M}} \mu$;
3. se I è un insieme finito tale che $\sum_{i \in I} p_i \leq 1 \wedge \forall i \in I (p_i \geq 0 \wedge \mu_i \xrightarrow{\mathcal{M}} \nu_i)$, allora $\biguplus_{i \in I} p_i \cdot \mu_i \xrightarrow{\mathcal{M}} \biguplus_{i \in I} p_i \cdot \nu_i$.

Le relazioni di riduzione probabilistiche ci permettono di modellare l'evolversi di un processo di riduzione, che identifichiamo con una sequenza di riduzione probabilistica.

Definizione 2.1.8 (Sequenza di riduzione)**.** Siano $\to$ un PARS su A e $a \in A$. Una sequenza $(a_i)_{i \in \{1,\ldots,n\}}$ con $n \in \mathbb{N}_{>0}$ è una sequenza di riduzione in $\to$ da a se e solo se:

$$a_1 = a \wedge a_n \in NF_{\to} \wedge \forall i \in \{2, \ldots, n\} \quad a_i \to a_{i+1}$$

Definizione 2.1.9 (Sequenza di riduzione probabilistica da una multidistribuzione)**.** Siano $\to$ un PARS su A e $\mu \in \mathcal{M}_{\leq 1}(A)$. Una successione $(\mu_i)_{n \in \mathbb{N}}$ è una sequenza di riduzione probabilistica in $\to$ da μ se e solo se:

$$\mu_0 = \mu \wedge \forall n \in \mathbb{N} \quad \mu_n \xrightarrow{\mathcal{M}} \mu_{n+1}$$

Ci tornerà spesso utile fare riferimento all'insieme di tutte le possibili sequenze di riduzioni probabilistiche da una data multidistribuzione.

Definizione 2.1.10 (Insieme delle sequenze di riduzione)**.** Siano $\to$ un ARS su A e $a \in A$. L'insieme delle sequenze di riduzione in $\to$ da a, $red_{\to}(a)$, è definito come

$$red_{\to}(a) := \{S \mid S \text{ è una sequenza di riduzione in } \to \text{ da } a\}$$

Definizione 2.1.11 (Insieme delle sequenze di riduzione probabilistica da una multidistribuzione)**.** Siano $\to$ un PARS su A e $\mu \in \mathcal{M}_{\leq 1}(A)$. L'insieme delle sequenze di riduzione probabilistica (s.r.p.) in $\to$ da μ, $red_{\to}(\mu)$, è definito come

$$red_{\to}(\mu) := \{S : \mathbb{N} \to \mathcal{M}_{\leq 1}(A) \mid S \text{ è una s.r.p. in } \to \text{ da } \mu\}$$

Per estensione, possiamo parlare di sequenze di riduzione probabilistica anche in assenza di multidistribuzioni.

Definizione 2.1.12 (Sequenza di riduzione probabilistica da un valore)**.** Siano $\to$ un PARS su A e $a \in A$. Una successione $(\mu_i)_{n\in\mathbb{N}}$ è una sequenza di riduzione probabilistica in $\to$ da a se e solo se è una sequenza di riduzione probabilistica in $\to$ da $\{\!\{1 : a\}\!\}$.

Definizione 2.1.13 (Insieme delle sequenze di riduzione probabilistica da un valore)**.** Siano $\to$ un PARS su A e $a \in A$. L'insieme delle sequenze di riduzione probabilistica in $\to$ da a, $red_{\to}(a)$, è definito come

$$red_{\to}(a) := red_{\to}(\{\!\{1 : a\}\!\})$$

A breve, vedremo nelle sequenze di riduzione il soggetto adatto per il nostro discorso riguardante la terminazione.

2.2 Sistemi probabilistici di riscrittura dei termini

La riscrittura dei termini introduce un proprio lessico specifico, legato al campo semantico della (ri)scrittura stessa.

Definizione 2.2.1 (Regola di riscrittura)**.** Siano V un insieme di variabili tale

che $|V| = \aleph_0$ e $T(\Sigma, V)$ l'insieme dei Σ-termini su V. Definiamo $l \to r$, una generica Σ-regola di riscrittura su V come

$$l \to r := (l, r) \quad , \quad \begin{cases} (l, r) \in T^2(\Sigma, V) \\ l \notin V \\ \mathcal{V}ar(l) \supseteq \mathcal{V}ar(r) \end{cases}$$

Manteniamo il secondo e il terzo vincolo, che chiaramente permettono di evitare casi patologici o di ovvia non-terminazione, più per compatibilità con lavori precedenti che per una reale necessità.

Nella loro versione probabilistica, le regole di riscrittura sostituiscono il loro membro destro con una multidistribuzione. Avremo quindi prima bisogno di esplicitare la definizione di insieme delle variabili occorrenti in una multidistribuzione su un insieme di termini.

Definizione 2.2.2 (Insieme delle variabili occorrenti in una multidistribuzione su un insieme di termini)**.** Siano V un insieme di variabili tale che $|V| = \aleph_0$, $T(\Sigma, V)$ l'insieme dei Σ-termini su V e μ tale che $\mu \in \mathcal{M}_{\leq 1}(T(\Sigma, V))$. L'insieme delle variabili occorrenti in μ è:

$$\mathcal{V}ar(\mu) := \bigcup_{s \in T(\Sigma,V), \mu(s) > 0} \mathcal{V}ar(s)$$

Definizione 2.2.3 (Regola di riscrittura probabilistica)**.** Siano V un insieme di variabili tale che $|V| = \aleph_0$ e $T(\Sigma, V)$ l'insieme dei Σ-termini su V. Definiamo $l \to \mu$, una generica Σ-regola di riscrittura probabilistica su V come

$$l \to \mu := (s, \mu) \quad , \quad \begin{cases} (l, \mu) \in T(\Sigma, V) \times \mathcal{M}(T(\Sigma, V)) \\ l \notin V \\ \mathcal{V}ar(l) \supseteq \mathcal{V}ar(\mu) \end{cases}$$

Siamo ora in grado di passare a un concetto centrale: quello di sistema (probabilistico o meno) di riscrittura dei termini.

Definizione 2.2.4 (Sistema di riscrittura di termini)**.** Siano V un insieme di variabili tale che $|V| = \aleph_0$ e $T(\Sigma, V)$ l'insieme dei Σ-termini su V. Un insieme R si dice essere un sistema di riscrittura di Σ-termini su V se e solo se:

$$\forall x \in R, \quad x \text{ è una } \Sigma\text{-regola di riscrittura su } V$$

L'estensione probabilistica è banale. Per brevità, talvolta lo abbrevieremo come PTRS (*Probabilistic Term Rewriting System*).

Definizione 2.2.5 (Sistema di riscrittura di termini probabilistico)**.** Siano V un insieme di variabili tale che $|V| = \aleph_0$ e $T(\Sigma, V)$ l'insieme dei Σ-termini su V. Un insieme R si dice essere un sistema di riscrittura probabilistico di Σ-termini su V se e solo se:

$$\forall x \in R, \quad x \text{ è una } \Sigma\text{-regola di riscrittura probabilistica su } V$$

Per esprimere il modo in cui le regole di un sistema di riscrittura dei termini possano essere applicate per ridurre un termine in un altro, abbiamo bisogno di un'appropriata relazione di riscrittura.

Definizione 2.2.6 (Relazione di riscrittura)**.** Siano V un insieme di variabili tale che $|V| = \aleph_0$ e R un sistema di riscrittura di Σ-termini su V. La relazione di riscrittura di R, $\to_R$, è il seguente ARS su $T(\Sigma, V)$:

$$\to_R := \{(C[\hat{\sigma}(l)], C[\hat{\sigma}(r)]) \mid C \in \mathcal{C}on(T(\Sigma, V)), \sigma \in \mathcal{S}ub(T(\Sigma, T)), l \to r \in R\}$$

Prima di passare alla versione probabilistica, è essenziale definire come le multidistribuzioni interagiscano con sostituzioni e contesti.

Definizione 2.2.7 (Sostituzione per multidistribuzioni)**.** Siano $\hat{\sigma}$ una $T(\Sigma, X)$-sostituzione di variabili e μ una multidistribuzione su $T(\Sigma, X)$. La $T(\Sigma, X)$-sostituzione per multidistribuzioni basata su $\hat{\sigma}$, $\hat{\hat{\sigma}}$, è definita come:

$$\hat{\hat{\sigma}} := \{\!\{p : \hat{\sigma}(a) \mid p : a \in \mu\}\!\}$$

Definizione 2.2.8 (Popolazione di un contesto per multidistribuzioni)**.** Sia C un contesto per Σ-termini su V in cui $\Box$ occorre in posizione p e sia μ una multidistribuzione su $T(\Sigma, V)$. La popolazione del contesto C con μ, $C[\mu]$, è definita come

$$C[s] := \{\!\{p : C[a] \mid p : a \in \mu\}\!\}$$

Disponiamo ora di tutti gli strumenti necessari.

Definizione 2.2.9 (Relazione di riscrittura probabilistica)**.** Siano V un insieme di variabili tale che $|V| = \aleph_0$ e R un PTRS per Σ-termini su V. La relazione di riscrittura probabilistica di R, $\to_R$, è il seguente PARS su $T(\Sigma, V)$:

$$\to_R := \{(C[\hat{\sigma}(l)], C[\hat{\hat{\sigma}}(\mu)]) \mid C \in \mathcal{C}on(T(\Sigma, V)), \sigma \in \mathcal{S}ub(T(\Sigma, T)), l \to \mu \in R\}$$

Esplicitata la definizione di relazione di riscrittura probabilistica, abbiamo ora modo di fare riferimento al PARS "sotteso" a un generico PTRS.

2.3 La terminazione probabilistica

In un contesto probabilistico, la comune nozione di terminazione cede il passo a un carosello di opzioni via via più stringenti. Definiremo queste proprietà inizialmente su generici PARS. Poi, le istanzieremo nel caso dei PTRS, che diremo godere di esse quando lo stesso vale per le rispettive relazioni di riscrittura probabilistica. Prima di fare questo, però, ci concediamo un momento per rievocare la definizione non probabilistica di sistema di riscrittura dei termini terminante.

Definizione 2.3.1 (Sistema di riduzione astratto terminante)**.** Sia $\to$ un ARS su A. Si dice che $\to$ è terminante quando

$$\nexists \{a_i\}_{i \in \mathbb{N}} \in A^{\mathbb{N}}, \forall i \in \mathbb{N} \quad a_i \to a_{i+1}$$

Definizione 2.3.2 (Sistema di riscrittura dei termini terminante)**.** Sia R un TRS. R è terminante quando il suo ordine di riscrittura lo è.

2.3.1 Terminazione quasi certa

Concluso il caso non probabilistico, partiamo con la condizione più debole, quella di terminazione quasi certa (AST, *Almost Sure Termination*), che vuole simulare l'imposizione che le sequenze infinite di riduzione si verifichino con probabilità tendente a zero.

Definizione 2.3.3 (Sistema di riduzione astratto probabilistico quasi certamente terminante)**.** Sia $\to$ un sistema di riduzione astratto probabilistico su un insieme A. $\to$ si dice essere quasi certamente terminante se e solo se

$$\forall \mu \in \mathcal{M}_{\leq 1}(A), (\mu_n)_{n\in\mathbb{N}} \in red_{\to}(\mu) \quad \lim_{n\to\infty} |\mu_n| = 0$$

Definizione 2.3.4 (Sistema di riscrittura dei termini probabilistico quasi terminante)**.** Sia R un PTRS. Si dice che R è quasi certamente terminante se e solo se la sua relazione di riscrittura probabilistica è quasi certamente terminante.

2.3.2 Terminazione quasi certa positiva

L'idea di terminazione può anche essere modellata vincolando la lunghezza attesa delle derivazioni.

Definizione 2.3.5 (Lunghezza di derivazione di una sequenza di riduzione)**.** Siano $\to$ un ARS su A, $a \in A$ e $(a_i)_{i\in\{1,\dots,n\}} \in red_{\to}(a)$. La lunghezza della sequenza di riduzione probabilistica $(a_i)_{i\in\{1,\dots,n\}}$ rispetto all'ARS $\to$, $edl_{\to}((a_i)_{i\in\{1,\dots,n\}})$, è definita come

$$dl_{\to}((a_i)_{i\in\{1,\dots,n\}}) := n$$

Definizione 2.3.6 (Lunghezza di derivazione attesa di una sequenza di riduzione probabilistica)**.** Siano $\to$ un PARS su A, $\mu \in \mathcal{M}_{\leq 1}(A)$ e $\vec{\mu} \in red_{\to}(\mu)$. La lunghezza attesa della sequenza di riduzione probabilistica $\vec{\mu}$ rispetto al PARS $\to$, $edl_{\to}(\vec{\mu})$, è definita come

$$edl_{\to}(\vec{\mu}) := \sum_{n\in\mathbb{N}} |\mu_n|$$

In particolare, il vincolo che ci interessa in un'ottica di terminazione è quello di finitezza, che vorremmo imporre su tutte le possibili sequenze di riduzione. Questa accezione di terminazione probabilistica è detta terminazione quasi certa positiva (PAST, *Positive Almost Sure Termination*).

Definizione 2.3.7 (Sistema di riduzione astratto probabilistico quasi certamente terminante positivamente)**.** Sia $\to$ un sistema di riduzione astratto probabilistico su un insieme A. $\to$ si dice essere quasi certamente terminante positivamente se e solo se

$$\forall \mu \in \mathcal{M}_{\leq 1}(A), \vec{\mu} \in red_{\to}(\mu) \quad edl_{\to}(\vec{\mu}) \text{ è finito}$$

Definizione 2.3.8 (Sistema di riscrittura dei termini probabilistico quasi certamente terminante positivamente)**.** Sia R un PTRS. Si dice che R è quasi certamente terminante positivamente se e solo se la sua relazione di riscrittura probabilistica è quasi certamente terminante positivamente.

Come già anticipato, PAST è una proprietà più forte di AST.

Teorema 2.3.1 (PAST implica AST)**.** *Sia R un PTRS. Se R è PAST, allora R è AST.*

Dimostrazione. Se R è PAST, allora lo è anche la sua relazione di riscrittura probabilistica, che chiamiamo $\to$. Di conseguenza, una sua generica sequenza di riduzione avrà come serie delle probabilità totali dei suoi termini un valore finito. Quindi, la successione dei termini in questione converge a 0. Perciò $\to$ è AST, e pertanto anche R deve esserlo. □

2.3.3 Terminazione quasi certa forte

Un sistema di riscrittura dei termini PAST ci dà garanzia che tutte le sequenze di riduzione della sua relazione di riscrittura abbiano una lunghezza attesa sì finita, ma potenzialmente arbitrariamente grande. Ci interessa quindi definire un limite superiore per le lunghezze di derivazione (che chiameremo "altezza di derivazione") a partire da una data multidistribuzione iniziale.

Definizione 2.3.9 (Altezza di derivazione)**.** Siano $\to$ un ARS su A e $a \in A$. L'altezza di derivazione attesa di a rispetto all'ARS $\to$, $dh_{\to}(a)$, è definita come

$$dh_{\to}(a) := \sup\{dl_{\to}(\vec{a}) \mid \vec{a} \in red_{\to}(a)\}$$

Definizione 2.3.10 (Altezza di derivazione attesa di una multidistribuzione)**.** Siano $\to$ un PARS su A e $\mu \in \mathcal{M}_{\leq 1}(A)$. L'altezza di derivazione attesa di μ rispetto al PARS $\to$, $edh_{\to}(\mu)$, è definita come

$$edh_{\to}(\mu) := \sup\{edl_{\to}(\vec{\mu}) \mid \vec{\mu} \in red_{\to}(\mu)\}$$

Non pretendiamo quindi che esista un unico vincolo superiore alle lunghezze di derivazione attese comune a tutte le multidistribuzioni iniziali, ma solo che, per ogni distribuzione degenere che assegna a un valore iniziale probabilità 1, disponiamo di un vincolo locale di questo tipo. Necessitiamo quindi di estendere la nostra definizione di altezza di derivazione attesa anche agli elementi dell'insieme iniziale.

Definizione 2.3.11 (Altezza di derivazione attesa di un valore)**.** Siano $\to$ un PARS su A e $a \in A$. L'altezza di derivazione attesa di a rispetto al PARS $\to$, $edh_{\to}(\mu)$, è definita come

$$edh_{\to}(a) := edh_{\to}(\{\!\{1 : a\}\!\})$$

Alla proprietà in questione daremo il nome di "terminazione quasi certa forte" (SAST, *Strong Almost Sure Termination*).

Definizione 2.3.12 (Sistema di riduzione astratto probabilistico quasi certamente terminante forte)**.** Sia $\to$ un sistema di riduzione astratto probabilistico su un insieme A. $\to$ si dice essere quasi certamente terminante forte se e solo se

$$\forall a \in A \quad edh_{\to}(a) \text{ è finito}$$

Definizione 2.3.13 (Sistema di riscrittura dei termini probabilistico quasi certamente terminante forte)**.** Sia R un PTRS. Si dice che R è quasi certamente

terminante forte se e solo se la sua relazione di riscrittura probabilistica è quasi certamente terminante forte.

In questo caso, SAST è più stringente di PAST. Per provarlo, necessitiamo di introdurre qualche lemma.

Lemma 2.3.1 (Scomposizione di una sequenza di riduzione)**.** *Siano* $\to$ *un PARS su* A *e* $\vec{\mu} \in \mathcal{M}^{\mathbb{N}}_{\leq 1}(A)$*. Allora:*

$$\vec{\mu} \in red_{\to}(\biguplus_{i \in I} p_i \cdot \nu_i)$$

se e solo se:

$$\exists f : I \to \mathcal{M}^{\mathbb{N}}_{\leq 1}(A), (\vec{\mu} = \biguplus_{i \in I} p_i \cdot f(i) \wedge \forall i \in I, f(i) \in red_{\to}(\nu_i))$$

Dimostrazione. Da sinistra a destra: costruiamo una procedura che, preso un valore per $i \in I$ costruisce $\vec{\rho_i} = f(i)$ per induzione su $n \in \mathbb{N}$, indice del termine corrente nella successione di multidistribuzioni.

- caso base: costruiamo i termini di indice 0 in modo tale che $\forall i \in I, \rho_{i,0} = \nu_i$;
- caso induttivo: assumiamo, per ipotesi induttiva, che $\mu_n = \biguplus_{i \in I} p_i \cdot \rho_{i,n}$. Costruiamo i termini di indice $n+1$ in modo tale che $\forall i \in I, \rho_{i,n} \xrightarrow{\mathcal{M}} \rho_{i,n+1}$. Quindi, per la Definizione 2.1.7, segue $\biguplus_{i \in I} p_i \cdot \rho_{i,n} \xrightarrow{\mathcal{M}} \biguplus_{i \in I} p_i \cdot \rho_{i,n+1}$, e cioè $\mu_n \xrightarrow{\mathcal{M}} \mu_{n+1}$.

Da destra a sinistra: ovvio per la scelta delle regole nella Definizione 2.1.7. □

Lemma 2.3.2 (Formula dell'altezza di derivazione attesa)**.** *Siano* $\to$ *un PARS su* A *e* $\biguplus_{i \in I} p_i \cdot \mu_i) \in \mathcal{M}_{\leq 1}(A)$*. Allora:*

$$edh_{\to}(\biguplus_{i \in I} p_i \cdot \mu_i) = \sum_{i \in I} p_i \cdot edh_{\to}(\mu_i)$$

Dimostrazione.

$$edh_{\to}(\biguplus_{i \in I} p_i \cdot \mu_i) = \sup\{edl_{\to}(\vec{\nu}) \mid \vec{\nu} \in red_{\to}(\biguplus_{i \in I} p_i \cdot \mu_i)\} =$$

$$= \sup\{\sum_{n\in\mathbb{N}} |\nu_n| \mid \vec{\nu} \in red_{\to}(\biguplus_{i\in I} p_i \cdot \mu_i)\} =$$

(Per il Lemma 2.3.1:)

$$= \sup\{\sum_{n\in\mathbb{N}} |\biguplus_{i\in I} p_i \cdot \nu_{n,i}| \mid \forall i \in I (\nu_{n,i})_{n\in\mathbb{N}} \in red_{\to}(\mu_i)\} =$$

(Per il Corollario 1.2.1:)

$$= \sup\{\sum_{i\in I} p_i \cdot \sum_{n\in\mathbb{N}} |\nu_{n,i}| \mid \forall i \in I (\nu_{n,i})_{n\in\mathbb{N}} \in red_{\to}(\mu_i)\} =$$

$$= \sum_{i\in I} p_i \cdot \sup\{\sum_{n\in\mathbb{N}} |\nu_n| \mid (\nu_n)_{n\in\mathbb{N}} \in red_{\to}(\mu_i)\} =$$

$$= \sum_{i\in I} p_i \cdot edh_{\to}(\mu_i)$$

□

Teorema 2.3.2 (SAST implica PAST)**.** *Sia R un PTRS. Se R è SAST, allora R è PAST.*

Dimostrazione. Se R è SAST, allora lo è anche la sua relazione di riscrittura probabilistica, che chiamiamo $\to$. Sia $\to$ un PARS su A. Allora, per ogni $a \in A$, $edh_{\to}(a)$ (e cioè $edh_{\to}(\{\!\{1 : a\}\!\})$) è finito. Quindi, per il Lemma 2.3.2, anche l'altezza di derivazione attesa di una generica multidistribuzione finita su A è finita. Di conseguenza, lo è anche la lunghezza di derivazione attesa, da essa superiormente limitata. Perciò $\to$ è PAST, e pertanto anche R deve esserlo. □

Per provare questo teorema, abbiamo dovuto introdurre il concetto di altezza di derivazione attesa. Una sua particolerità è che essa coincide con il proprio valore atteso.

Teorema 2.3.3 (Altezza di derivazione attesa e suo valore atteso)**.** *Siano $\to$ un PARS su A e $\mu \in \mathcal{M}_{\leq 1}(A)$. Allora:*

$$edh_{\to}(\mu) = \mathbb{E}(edh_{\to}(\mu))$$

Dimostrazione. Segue dal Lemma 2.3.2 considerando

$$edh_{\to}(\mu) = edh_{\to}(\biguplus_{p:a\in\mu} p \cdot \{\!\{1 : a\}\!\})$$

e osservando che quindi

$$\sum_{p:a\in\mu} p \cdot edh_{\to}(\{\!\{1 : a\}\!\}) = \mathbb{E}(edh_{\to}(\mu))$$

□

2.4 Funzioni di rango probabilistiche

Già a partire dal caso non probabilistico, lo strumento storicamente più diffuso per provare la terminazione è la riduzione tramite codifica a sistemi già riconosciuti come terminanti, come $>$ su $\mathbb{N}$.

Definizione 2.4.1 (Codifica fra sistemi di riduzione astratti)**.** Siano $\to$ un ARS su A, $\sqsupset$ un ARS su B e $f : A \to B$. Si dice che f è una codifica di $\to$ in $\sqsupset$ se e solo se:

$$\forall (a_1, a_2) \in \to, f(a_1) \sqsupset f(a_2)$$

Dati due sistemi di riduzione astratti, potrebbero esistere più funzioni distinte che codifichino correttamente il primo nel secondo: usiamo il termine "codificabile" quando ne esiste almeno una.

Definizione 2.4.2 (Sistema di riduzione astratto codificabile)**.** Siano $\to$ e $\sqsupset$ due ARS. Si dice che $\to$ è codificabile in $\sqsupset$ quando esiste almeno una codifica di $\to$ in $\sqsupset$.

L'essere codificabile in almeno un ARS terminante è in effetti condizione sia sufficiente che necessaria per la terminazione.

Teorema 2.4.1 (Correttezza e completezza del metodo di codifica fra ARS per provare la terminazione)**.** *Sia $\to$ un ARS. $\to$ è terminante se e solo se è codificabile in almeno un ARS terminante.*

Dimostrazione. Da sinistra a destra: ovvio, siccome la funzione identità codifica $\to$ in sé stesso. Da destra a sinistra: assumiamo che esista una catena discendente infinita e ne deriviamo l'assurdo. Se infatti essa esistesse, potremmo mapparla tramite la codifica nell'ARS ipotizzato terminante e otterremo una catena discendente infinita in quest'ultimo, e questo è in contraddizione con il suo essere terminante. □

La definizione di codifica non cambia di molto quando si vuole passare allo scenario probabilistico, se ricordiamo che possiamo sollevare funzioni rispetto alle multidistribuzioni grazie alla Definizione 1.2.28.

Definizione 2.4.3 (Codifica fra sistemi di riduzione astratti probabilistici)**.** Siano $\to$ un PARS su A, $\sqsupset$ un PARS su B e $f : A \to B$. Si dice che f è una codifica di $\to$ in $\sqsupset$ se e solo se:

$$\forall (a, \mu) \in \to, f(a) \sqsupset f(\mu)$$

Prima di passare a dimostrare correttezza e completezza anche di quest'ultimo metodo di dimostrazione della terminazione, necessitiamo di qualche risultato intermedio.

Lemma 2.4.1. *Siano $\to$ e $\sqsupset$ due PARS e $f : A \to B$ una codifica di $\to$ in $\sqsupset$. Allora:*

$$\forall (\mu, \nu) \in \xrightarrow{\mathcal{M}}, \exists \xi \in \mathcal{M}_{\leq 1}(B), f(\mu) \sqsupset^{\mathcal{M}} \nu \uplus \xi$$

Dimostrazione. Per induzione sulla struttura della derivazione di $\mu \xrightarrow{\mathcal{M}} \nu$:

- primo caso base: $\{\!\{1 : a\}\!\} \xrightarrow{\mathcal{M}} \varnothing$, con $a \in NF_{\to}$.

  Dobbiamo dimostrare $\exists \xi \in \mathcal{M}_{\leq 1}(B), \{\!\{1 : f(a)\}\!\} \sqsupset^{\mathcal{M}} \xi$, che è ovvio per la scelta delle nostre regole di derivazione;

- secondo caso base: $\{\!\{1 : a\}\!\} \xrightarrow{\mathcal{M}} \mu$, con $(a, \mu) \in \to$.

  Dobbiamo dimostrare $\exists \xi \in \mathcal{M}_{\leq 1}(B), \{\!\{1 : f(a)\}\!\} \sqsupset^{\mathcal{M}} f(\mu) \uplus \xi$. Se scegliamo $\varnothing$ come ξ, questo è ovvio per la definizione di codifica;

- caso induttivo: $\biguplus_{i\in I} p_i{\cdot}\mu_i \xrightarrow{\mathcal{M}} \biguplus_{i\in I} p_i{\cdot}\nu_i$, con I insieme finito, $\sum_{i\in I} p_i \leq 1$ e $\forall i \in I(p_i \geq 0 \wedge \mu_i \xrightarrow{\mathcal{M}} \nu_i)$.

  Dobbiamo dimostrare $\exists \xi \in \mathcal{M}_{\leq 1}(B), \biguplus_{i\in I} p_i{\cdot}f(\mu_i) \sqsupset^{\mathcal{M}} (\biguplus_{i\in I} p_i{\cdot}f(\nu)) \uplus \xi$, che segue dall'applicazione dell'ipotesi induttiva su $\mu_i \xrightarrow{\mathcal{M}} \nu_i$ per ogni $i \in I$. In questo caso scegliamo come valore di ξ l'unione di tutti i valori che questa variabile assume nelle varie applicazioni dell'ipotesi induttiva.

□

Il prossimo lemma ci permette di "trasportare" sequenze di riduzione da un PARS a un altro usando opportunamente una codifica data.

Lemma 2.4.2. *Siano $\to$ e $\sqsupset$ due PARS e $f : A \to B$ una codifica di $\to$ in $\sqsupset$. Allora:*

$$\forall \mu \in \mathcal{M}_{\leq 1}(A) \quad \forall (\mu_n)_{n\in\mathbb{N}} \in red_{\to}(\mu) \quad \exists (\nu_n)_{n\in\mathbb{N}} \in red_{\sqsupset}(f(\mu)),$$

$$(edl((\mu_n)_{n\in\mathbb{N}}) \leq edl((\nu_n)_{n\in\mathbb{N}}) \wedge \forall n \in \mathbb{N} \quad f(\mu_n) \subseteq \nu_n)$$

Dimostrazione. Costruiamo $(\nu_n)_{n\in\mathbb{N}}$ induttivamente sulla struttura di $(\mu_n)_{n\in\mathbb{N}}$:

- passo base: $\nu_0 = f(\mu_0)$;
- passo induttivo: per ipotesi induttiva, possiamo assumere $f(\mu_n) \subseteq \nu_n$, e cioè
$$\exists \xi \in \mathcal{M}_{\leq 1}(B), \nu_n = f(\mu_n) \uplus \xi$$
Inoltre, per il Lemma 2.4.1:
$$\exists \rho \in \mathcal{M}_{\leq 1}(B), f(\mu_n) \sqsupset f(\mu_{n+1}) \uplus \rho$$
Sia $\xi' \in \mathcal{M}_{\leq 1}(M)$ scelto a piacere in modo che $\xi \sqsupset \xi'$. Costruiamo l'elemento corrente della successione come $\nu_{n+1} = f(\mu_{n+1}) \uplus \rho \uplus \xi'$. Segue che $\nu_n \sqsupset \nu_{n+1}$.

Osserviamo infine che, siccome per costruzione vale

$$\forall n \in \mathbb{N}, |\mu_n| \leq |\nu_n|$$

allora:

$$edl_{\to}((\mu_n)_{n\in\mathbb{N}}) \leq edl_{\sqsupset}((\nu_n)_{n\in\mathbb{N}})$$

e questo conclude la prova. □

Questo era l'ultimo lemma necessario alla dimostrazione del teorema sopra citato, che esplicita il valore delle codifiche come strumento per dimostrare la terminazione di PARS.

Teorema 2.4.2 (Correttezza e completezza delle codifiche per provare SAST)**.** *Un PARS $\to$ su A è SAST se e solo se è codificabile in un qualche PARS SAST $\sqsupset$. Inoltre, in questo caso, vale:*

$$\forall a \in A, edh_{\to}(a) \leq edh_{\sqsupset}(f(a))$$

Dimostrazione. Da sinistra a destra: ovvio, siccome la funzione identità codifica $\to$ in sé stesso. Da destra a sinistra: assumiamo di avere una codifica di $\to$ in un qualche PARS SAST $\sqsupset$ e dimostriamo che $\to$ sia SAST considerando un generico $a \in A$. Per il Lemma 2.4.2, possiamo asserire che

$$\sup\{edl_{\to}(\vec{\mu}) \mid \vec{\mu} \in red_{\to}(a)\} \leq \sup\{edl_{\sqsupset}(\vec{\nu}) \mid \vec{\nu} \in red_{\sqsupset}(f(a))\}$$

e quindi $edh_{\to}(a) \leq edh_{\sqsupset}(f(a))$. Perciò se $\sqsupset$ è SAST, lo stesso deve valere per $\to$. □

Di norma, si tenta di codificare verso semplici (P)ARS sull'insieme dei numeri reali non negativi.

Definizione 2.4.4 ($\geq \epsilon+$)**.** Sia $\epsilon \in \mathbb{R}^+$. Definiamo $[\geq \epsilon+]$, l'ARS su $\mathbb{R}_{\geq 0}$:

$$[\geq \epsilon+] := \{(a, b) \in \mathbb{R}^2_{\geq 0} \mid a \geq \epsilon + b\}$$

Per lo scenario probabilistico, lavoriamo invece con i valori attesi.

Definizione 2.4.5 ($\geq \epsilon + \mathbb{E}$)**.** Sia $\epsilon \in \mathbb{R}^+$. Definiamo $[\geq \epsilon + \mathbb{E}]$, il PARS su $\mathbb{R}_{\geq 0}$:

$$[\geq \epsilon + \mathbb{E}] := \{(a, \mu) \in \mathbb{R}_{\geq 0} \times \mathcal{M}(\mathbb{R}_{\geq 0}) \mid a \geq \epsilon + \mathbb{E}(\mu)\}$$

Ecco quindi il nostro strumento principe per provare SAST: le funzioni di rango probabilistiche, anche note come "funzioni di Lyapunov".

Definizione 2.4.6 (Funzione di rango probabilistica)**.** Sia $\to$ un PARS. Si dice che f è una funzione di rango probabilistica per $\to$ quando f è una codifica di $\to$ in $[\geq \epsilon + \mathbb{E}]$.

L'idea di una funzione di rango è quella di fungere da "funzione misura" dei passi di computazione attesi rimanenti prima della terminazione. Rimangono da provare correttezza e completezza di questa strategia.

2.4.1 Correttezza

Cominciamo osservando quali garanzie ci offra un singolo passo di riduzione fondato su $[\geq \epsilon + \mathbb{E}]$.

Lemma 2.4.3. *Siano $\mu, \nu \in \mathcal{M}_{\leq 1}(\mathbb{R}_{\geq 0})$. Se $\mu[\geq \epsilon + \mathbb{E}]^{\mathcal{M}}\nu$, allora $\mathbb{E}(\mu) \geq \epsilon \cdot |\nu| + \mathbb{E}(\nu)$.*

Dimostrazione. Per induzione sulla struttura della derivazione di $\mu[\geq \epsilon+\mathbb{E}]^{\mathcal{M}}\nu$:

- primo caso base: $\{\!\{1 : a\}\!\}[\geq \epsilon + \mathbb{E}]^{\mathcal{M}}\varnothing$, con $a < \epsilon$. Dobbiamo dimostrare $a \geq 0$, che è ovvio siccome $a \in \mathbb{R}_{\geq 0}$ per ipotesi;
- secondo caso base: $\{\!\{1 : a\}\!\}[\geq \epsilon+\mathbb{E}]^{\mathcal{M}}\nu$, con $a \geq \epsilon+\mathbb{E}(\nu)$ e $\nu \in \mathcal{M}(\mathbb{R}_{\geq 0})$. In questo caso, quindi, dobbiamo dimostrare proprio $a \geq \epsilon + \mathbb{E}(\nu)$, che è ovvio per ipotesi;
- caso induttivo: $\biguplus_{i\in I} p_i \cdot \mu_i[\geq \epsilon + \mathbb{E}]^{\mathcal{M}} \biguplus_{i\in I} p_i \cdot \nu_i$, con $\forall i \in I, \mu_i[\geq \epsilon + \mathbb{E}]^{\mathcal{M}}\nu_i$. Per ipotesi induttiva, possiamo assumere $\forall i \in I, \mathbb{E}(\mu_i) \geq \epsilon \cdot |\nu_i| + \mathbb{E}(\nu_i)$. Moltiplicando ogni disuguaglianza per p_i e sommandole in seguito membro a membro, otteniamo $\sum_{i\in I} p_i \cdot \mathbb{E}(\mu_i) \geq \sum_{i\in I} p_i \cdot (\epsilon \cdot |\nu_i| + \mathbb{E}(\nu_i))$, e quindi $\mathbb{E}(\mu) \geq \epsilon \cdot |\nu| + \mathbb{E}(\nu)$.

□

Lemma 2.4.4. *Sia A un insieme. Allora:*

$$\forall \mu_0 \in \mathcal{M}_{\leq 1}(A), \vec{\mu} \in red_{[\geq \epsilon+\mathbb{E}]}(\mu_0), \mathbb{E}(\mu_0) \geq \epsilon \cdot edl(\vec{\mu})$$

Dimostrazione. Partiamo dimostrando $\forall n, m \in \mathbb{N}(n \geq m \rightarrow \mathbb{E}(\mu_m)\epsilon \cdot \sum_{i=m+1}^{n} |\mu_i|)$. Procediamo per induzione su $n - m$:

- caso base: ci riduciamo a dimostrare $\mathbb{E}(\mu_m) \geq 0$, che è ovvio;
- caso induttivo: per il Lemma 2.4.3 e l'ipotesi induttiva, abbiamo

$$\mathbb{E}(\mu_m) \geq \epsilon \cdot |\mu_{m+1}| + \mathbb{E}(\mu_{m+1}) \geq \epsilon \cdot |\mu_{m+1}| + \epsilon \cdot \sum_{i=m+2}^{n} |\mu_i| = \epsilon \cdot \sum_{i=m+1}^{n} |\mu_i|$$

Nel caso particolare con $m = 0$, si ha

$$\forall n \in \mathbb{N}, \mathbb{E}(\mu_m) \geq \epsilon \cdot \sum_{i=1}^{n} |\mu_i|$$

che, essendo una proprietà definitivamente applicabile per n arbitrariamente grandi, è valida anche per il limite della successione in questione. □

Teorema 2.4.3 (Correttezza delle funzioni di rango probabilistiche per SAST)**.** *Se $\epsilon \in \mathbb{R}^+$, allora*

$$\forall a \in \mathbb{R}_{\geq 0}, edh_{[\geq \epsilon + \mathbb{E}]}(a) \leq \frac{a}{\epsilon}$$

(e quindi $[\geq \epsilon + \mathbb{E}]$ è SAST).

Dimostrazione. Per il Lemma 2.4.4, vale:

$$\forall \vec{\mu} \in red_{[\geq \epsilon + \mathbb{E}]}(a), edl_{[\geq \epsilon + \mathbb{E}]}(\vec{\mu}) \leq \frac{a}{\epsilon}$$

e quindi

$$edh_{[\geq \epsilon + \mathbb{E}]}(a) \leq \frac{a}{\epsilon}$$

□

2.4.2 Completezza

La dimostrazione di completezza è invece più semplice, e non richiede lemmi preliminari.

Teorema 2.4.4 (Completezza delle funzioni di rango probabilistiche per SAST)**.** *Se $\to$ è un PARS SAST, esiste almeno una funzione di rango probabilistica per $\to$.*

Dimostrazione. Sia $\to$ un PARS SAST su un generico insieme A. Siccome $\to$ è SAST, $edh_\to : A \to \mathbb{R}_{\geq 0}$. Se $\to = \varnothing$, il teorema è banalmente soddisfatto dalla codifica vuota. In caso contrario, deve esistere almeno una coppia $a \to \mu$, con $|\mu| = 1$. Osserviamo che:

$$edh_\to(a) = \sup\{edl_\to(\vec{\mu}) \mid \vec{\mu} \in red_\to(a)\} \geq \sup\{|\mu| + edl_\to(\vec{\mu}) \mid \vec{\mu} \in red_\to(\mu)\} =$$

$$= |\mu| + \sup\{edl_\to(\vec{\mu}) \mid \vec{\mu} \in red_\to(\mu)\} = |\mu| + edh_\to(\mu) =$$

(Per il Teorema 2.3.3:)

$$= |\mu| + \mathbb{E}(edh_\to(\mu)) =$$

Questo ci porta a concludere:

$$edh_\to(a)[\geq 1 + \mathbb{E}]edh_\to(\mu)$$

Il nostro esempio di funzione di rango probabilistica per $\to$ è quindi $edh_\to$. □

3 Ordini probabilisticamente monotoni

Al termine dello scorso capitolo, abbiamo presentato una tecnica corretta e completa per provare che un PTRS sia SAST: esibire una funzione di rango probabilistica per esso. In questo ultimo capitolo facciamo leva su questo risultato per fornire una nuova tecnica di costruzione di PARS SAST a partire da ARS che rispettano determinate condizioni.

3.1 Motivazione

Dato un generico PARS che vogliamo dimostrare essere SAST, la nostra prova costruttiva del Teorema 2.4.4 ci suggerirebbe di fare affidamento sull'altezza attesa della derivazione. Dimostrare che essa sia davvero una funzione di Lyapunov, però, non è sempre impresa facile nel caso generale.

Questo problema ha una controparte non-probabilistica, da cui prenderemo ispirazione: è la ricerca di ordini di semplificazione, e cioè ordini stretti di riscrittura che soddisfano la seguente proprietà del sottotermine.

Definizione 3.1.1 (Proprietà del sottotermine)**.** Sia $S \subseteq T^2(\Sigma, V)$. Si dice che S gode della proprietà del sottotermine quando vale che

$$\forall t \in T(\Sigma, V), p \in \mathcal{P}os(t) \setminus \epsilon \quad tSt|_p$$

In assenza di effetti probabilistici, proviamo che un TRS sia terminante mostrando che la sua relazione di riscrittura sia un ordine di semplificazione

secondo la definizione di cui sotto; in presenza di effetti probabilistici, invece, proviamo che un PTRS sia SAST semplicemente costruendo una sua codifica in $[\geq \epsilon + \mathbb{E}]$.

Definizione 3.1.2 (Ordine di semplificazione)**.** Sia $S \subseteq T^2(\Sigma, V)$. Si dice che S è un ordine di semplificazione su $T(\Sigma, V)$ quando S è un ordine stretto su $T(\Sigma, V)$ che gode della proprietà del sottotermine.

Entrambi gli approcci forniscono insomma la garanzia di terminazione nei loro rispettivi ambiti. Si potrebbe quindi pensare, sulla base di questa analogia, di espandere le definizioni di specifici ordini di semplificazione al contesto probabilistico, e adattare di conseguenza le tencniche per la loro costruzione. Questa strategia ha già portato i suoi frutti: Avanzini et al. [2] hanno effettuato questa operazione per il metodo di interpretazione, mostrandone i risultati per gli ordini di semplificazione polinomiali e matriciali.

In questo capitolo seguiremo invece una direttrice più ingenua: la costruzione di tecniche che, applicate a generiche famiglie di ordini di semplificazione noti nel rispetto di certe precondizioni, portino a famiglie di PTRS SAST.

3.2 Definizione

3.2.1 Ordini probabilisticamente monotoni

L'idea degli ordini probabilisticamente monotoni è quella di partire da un ordine non probabilistico $\succ$ e limitare inferiormente con un valore strettamente positivo la probabilità che ad ogni passo di computazione un termine venga riscritto in uno con esso messo in relazione da $\succ$. Ecco gli accorgimenti che ci guidano verso la loro definizione:

- imporremo che le multidisitribuzioni proprie di arrivo siano costruite a partire da insiemi finiti di indici. Questa non è un'assunzione forte: essendo i PTRS che vogliamo catturare tipicamente finiti, ed essendoci pertanto un numero finito di applicazioni di regola a partire da un dato termine, la seguente definizione fungerà comunque da ottimo prototipo di relazione di riscrittura probabilistica alla base di un possibile PTRS;

- in vista di una futura funzione di rango probabilistico, e cioè codifica verso $[\geq \epsilon + \mathbb{E}]$, vogliamo vincolare inferiormente la decrescita della lunghezza di derivazione attesa lungo un passo di riduzione. Per fare ciò, dovremo imporre valori minimi anche ad almeno una probabilità;
- la nostra definizione di (sotto)multidistribuzione tollera anche probabilità nulle. È desiderabile che la nostra costruzione si comporti indifferentemente fra due multidistribuzioni che differiscono solo per la presenza di esiti caratterizzati da tali probabilità degeneri.

Segue un primo tentativo di definizione.

Definizione 3.2.1 (Ordine probabilisticamente monotono)**.** Siano $\succ$ un sistema di riduzione astratto su A tale che

$$\forall a \in A \quad dh_{\succ}(a) \in \mathbb{N}$$

e $\epsilon \in \mathbb{R}_{>0}$. L'ordine probabilisticamente monotono (o PMO, *probabilistically monotonic order*) indotto da $\succ$, $>_{p\succ}$, è il sistema di riduzione astratto probabilistico su A che ha per elementi tutte e sole le coppie della forma

$$(a, \{\!\{p_i : b_i \mid i \in I\}\!\})$$

(e cioè i cui secondi elementi sono sempre multinsiemi finiti da insiemi finiti di indici) che rispettano la seguente condizione:

$$\forall i \in I \quad (p_i = 0 \vee a \succeq b_i)$$

$$\wedge$$

$$\exists J \subseteq I \quad (\sum_{j \in J} p_j \geq \epsilon \quad \wedge \quad \forall j \in J \quad a \succ b_j)$$

Il primo congiunto del precedente predicato esprime una proprietà che deve essere condivisa da ogni elemento della multidistribuzione di arrivo: il rispettare in modo non stretto l'ordine di partenza. Ovviamente, gli esiti che si verificano con probabilità nulla sono esentati da questo vincolo. Il secondo

congiunto, invece, asserisce l'esistenza di un sottogruppo di esiti che si faccia carico di "diminuire" il valore atteso durante un passo di riduzione, esso potrebbbe essere composto perfino da un unico elemento o dall'intero insieme I, ma di certo non può essere l'insieme vuoto, siccome abbiamo scelto $\epsilon \in \mathbb{R}_{>0}$.

La seguente semplice implementazione in Standard ML fa leva sulla nozione che nessuna probabilità assuma mai valori negativi. Se infatti esiste almeno un $J \subseteq I$ che rispetti la condizione di cui sopra, allora essa sarà di certo rispettata anche da $\{j \in I \mid a \succ b_j\}$ (`maxJ` nel codice).

Programma 3.1: ordine probabilisticamente monotono (`mpo.sml`, 5-16)

```
1  (* ('a * 'a) -> order *)
2  fun geq ord (x, y) = ord (x, y) = GR orelse x = y
3
4  (* term multidistribution -> term -> (term * term -> ord)
5     -> term multidistribution *)
6  fun maxJ mu a ord = (filter (fn (_, bi) => ord (a, bi) = GR) mu)
7
8  (* (term * term -> order) -> term * term multidistribution -> real -> ord *)
9  fun mpo ord (a, mu) epsilon =
10   (forall (fn (pi, bi) => pi <= 0 orelse geq ord (a, bi)) mu)
11   andalso
12   (totalProbability (maxJ mu a ord) >= epsilon)
```

Prima di proseguire, ci concediamo il tempo di analizzare due esempi pratici di (non-)applicazione di questo concetto.

Esempio 3.2.1 (Sistema di riscrittura dei termini per *silly*)**.** *Si consideri la procedura* ***`silly`*** *del seguente programma in Standard ML:*

Programma 3.2: procedura `silly` (`geo.sml`, 1-12)

```
1  (* Generatore pseudocasuale *)
2  val rand = Random.rand (0, 0);
3
4  (* unit -> bool *)
5  fun coinToss _ = Random.randRange (0, 1) rand = 1;
6
7  (* int -> int *)
8  fun silly 0 = 0
9    | silly (n + 1) =
10       if coinToss ()
11         then silly n
12         else n + 1;
```

Questo programma può essere descritto scegliendo $\Phi = \{0, S, silly\}$ *(zero, funzione successore, procedura SML). Ovviamente* $\Sigma(0) = 0$, $\Sigma(S) = 1$ *e* $\Sigma(silly) = 1$. *Il nostro PTRS* R *avrà due sole regole, definite con l'ausilio della variabile* n*:*

$$silly(0) \quad R \quad \{\!\{1 : 0\}\!\}$$

$$silly(S(n)) \quad R \quad \left\{\!\left\{ \frac{1}{2} : silly(n), \quad \frac{1}{2} : S(n) \right\}\!\right\}$$

Costruiremo un ordine probabilisticamente monotono indotto dall'ARS $\succ$ *proposto a breve, che a sua volta fa uso della seguente definizione di "altezza di un termine"* $t \in T(\Sigma, V)$, $h(t)$*:*

$$h(t) = \begin{cases} 0 & t \in V \vee t \in \Phi \wedge \Sigma(t) = 0 \\ 1 + \max_{i \in \{1,\dots,n\}} h(s_i) & t = s(s_i, \dots, s_n) \end{cases}$$

L'ARS su $T(\Sigma, V)$ *denominato* $\succ$ *citato sopra è definito univocamente dalla seguente condizione:*

$$s \succ t \leftrightarrow h(s) > h(t)$$

Quindi $>_{p\succ}$ *è l'ordine probabilisticamente monotono indotto da* $\succ$*, e la relazione di riscrittura di* R *è in effetti un sottoinsieme di* $>_{p\succ}$*. Di certo l'altezza di derivazione è, in questo caso, finita per ogni termine.*

Il prossimo scenario è invece meno fortunato.

Esempio 3.2.2 (Sistema di riscrittura dei termini per `geo`)**.** *Si consideri la seguente procedura* `geo`*, ispirata a una successione geometrica:*

Programma 3.3: procedura `geo` (`geo.sml`, 14-17)

```
1 (* int -> int *)
2 fun geo n = if coinToss ()
3   then geo (n + 1)
4   else n;
```

Aggiungendo ai valori assegnati a Φ *e* Σ *nell'esempio precedente geo, con* $\Sigma(geo) = 1$*, costruiamo stavolta il nostro PTRS* R *come un singoletto contenente solo la seguente regola, che fa sempre uso della variabile* n*:*

$$geo(n) \quad R \quad \left\{\!\!\left\{ \frac{1}{2} : geo(S(n)), \quad \frac{1}{2} : n \right\}\!\!\right\}$$

Affinché possiamo anche in questo caso concepire la relazione di riscrittura di R come sottoinsieme di un PMO $>_{p\succ}$ indotto da un qualche ordine $\succ$, deve valere $geo(n) \succeq geo(S(n))$. Siccome $geo(n) \neq geo(S(n))$, in particolare, allora $geo(n) \succ geo(S(n))$. Questo garantirebbe l'esistenza di una catena discendente infinita $geo(n) \succ geo(S(n)) \succ geo(S(S(n))) \succ \ldots$. Tale stato di cose, ovviamente, è incompatibile con il vincolo di finitezza delle altezze di derivazione dei termini.

La scomoda situazione appena descritta ci spinge a una riformulazione della nostra definizione di ordine probabilisticamente monotono, che è attualmente inadeguata per la gestione di queste casistiche.

3.2.2 Ordini probabilisticamente monotoni via codifiche

Lo studio dell'Esempio 3.2.2 ci ha costretto a una scelta forzata: abbiamo dovuto imporre $geo(n) \succ geo(S(n))$ solo perché $geo(n) \neq geo(S(n))$. Tuttavia, guardando il nostro PTRS R, si potrebbe sostenere che $geo(n)$ e $geo(S(n))$ abbiano la stessa altezza di derivazione attesa e che quindi, se anche non sintatticamente uguali, dovrebbero essere considerati "equivalenti" dalla nostra definizione. La prima clausola della nostra nuova formulazione, quindi, si basa su una funzione di codifica scalare dei termini per astrarre dall'identità sintattica limitandosi a confrontare solo i valori delle codifiche dei due termini in questione. Ovviamente, il ruolo inteso della codifica è quello di rappresentare l'altezza di derivazione dell'ARS di partenza (o quantomeno una grandezza strettamente monotona rispetto ad essa).

Definizione 3.2.2 (Ordine probabilisticamente monotono via codifica)**.** Siano $\succ$ un sistema di riduzione astratto su A, $\epsilon, \delta \in \mathbb{R}_{>0}$ e $f : A \to \mathbb{R}_{>0}$ una codifica di $\succ$ in $[\geq \delta+]$. L'ordine probabilisticamente monotono indotto da $\succ$ via

codifica f, $>_{p\succ,f}$, è il sistema di riduzione astratto probabilistico su A che ha per elementi tutte e sole le coppie della forma

$$(a, \{\!\{p_i : b_i \mid i \in I\}\!\})$$

(e cioè i cui secondi elementi sono sempre multinsiemi finiti da insiemi finiti di indici) che rispettano la seguente condizione:

$$\forall i \in I \quad (p_i = 0 \vee f(a) \succeq f(b_i))$$

$$\wedge$$

$$\exists J \subseteq I \quad (\sum_{j \in J} p_j \geq \epsilon \quad \wedge \quad \forall j \in J \quad a \succ b_j)$$

Si noti che, ora che viene fatto usa della funzione di codifica, non è più necessario imporre la finitezza dell'altezza di derivazione per ogni termine. La nostra formalizzazione in Standard ML invece rimane pressoché invariata, con l'aggiunta di un parametro funzionale che rappresenta f:

Programma 3.4: ordine probabilisticamente monotono (`mpo.sml`, 18-23)

```
1 (* (term * term -> order) -> term * term multidistribution -> real ->
2    (term -> real) -> ord *)
3 fun mpo ord (a, mu) epsilon f =
4   (forall (fn (pi, bi) => pi <= 0 orelse f a >= f bi) mu)
5   andalso
6   (totalProbability (maxJ mu a ord) >= epsilon)
```

Ora possiamo con fiducia costruire un ARS ad hoc per `geo` che induca il nostro PMO, tenendo però presente che questa volta necessiteremo anche di una codifica.

Esempio 3.2.3 (Sistema di riscrittura dei termini probabilistico quasi certamente terminante forte per `geo`)**.** *Siano Φ un insieme, $\Phi' \subseteq \Phi$, Σ una segnatura su Φ, V un insieme di variabili e $s \in T(\Sigma, V)$. Il numero di occorrenze di simboli in Φ' all'interno di s, $|s|_{\Phi'}$ è definito come:*

$$|s|_{\Phi'} := \begin{cases} 0 & s \in V \\ \sum_{i=1}^{n} |s_i|_{\Phi'} & s = f(s_1, \ldots, s_n) \wedge f \notin \Phi' \\ 1 + \sum_{i=1}^{n} |s_i|_{\Phi'} & s = f(s_1, \ldots, s_n) \wedge f \in \Phi' \end{cases}$$

Gli ordini booleani che andiamo a definire si basano sul semplice confronto di questa quantità, che deve strettamente decrescere a ogni passo di riduzione.

Siano Φ un insieme, $\Phi' \subseteq \Phi$, Σ una segnatura su Φ, V un insieme di variabili e $s \in T(\Sigma, V)$. L'ordine booleano indotto da Φ', $>_{\Phi'}$, è l'unico ARS su $T(\Sigma, V)$ che soddisfa la seguente condizione:

$$\forall s, t \in T(\Sigma, V) \quad (s >_{\Phi'} t \leftrightarrow |s|_{\Phi'} > |t|_{\Phi'})$$

Partendo dall'Esempio 3.2.2, possiamo correttamente costruire il PMO indotto da $>_{\{geo\}}$ via codifica $|\cdot|_{\{geo\}}$.

3.3 Terminazione quasi certa forte

Dimostrare la terminazione quasi certa forte degli ordini probabilisticamente monotoni (via codifica o meno) è, ovviamente, funzionale a provare che anche i PTRS che hanno un PMO come relazione di riscrittura probabilistica siano SAST (per la Definizione 2.3.13). Perfino nel caso in cui la relazione di riscrittura probabilistica in questione sia anche solo un sottoinsieme di un PMO si ha la sicurezza che il PTRS in questione sia SAST: la funzione identità, infatti, è di certo una codifica corretta dalla relazione di riscrittura verso il PMO. Procediamo quindi a provare che ogni PMO sia SAST. Per farlo, naturalmente, useremo lo strumento delle funzioni di rango probabilistiche, già introdotte nel precedente capitolo. Per aiutarci a traslare i risultati di terminazione in presenza di effetti probabilistici, faremo uso della seguente definizione ausiliaria, sulla base della quale costruiremo poi l'unico lemma necessario alla dimostrazione.

Definizione 3.3.1 (Controparte probabilistica di un sistema di riduzione astratto)**.** Sia $\to$ un sistema di riduzione astratto su A. Definiamo la sua

controparte probabilistica, $\rightarrowtail$, come il seguente sistema di riduzione astratto probabilistico su A:

$$\rightarrowtail := \{(a, \mu) \in A \times \mathcal{M}(A) \mid \exists b \in A \quad (\mu = \{\!\{1 : b\}\!\} \wedge a \to b)\}$$

Il seguente lemma garantisce che le controparti probabilistiche degli ordini di percorso lessicografici siano SAST se e solo se tali ordini abbiamo solo altezze di derivazione finite. Ai fini della nostra esposizione, tali controparti fungono quindi da "nucleo" di partenza per provare SAST su tutti gli ordini probabilisticamente monotoni indotti da ARS con altezze di derivazione esclusivamente finite.

Lemma 3.3.1 (Funzioni di Lyapunov delle controparti probabilistiche dei sistemi di riduzione astratti)**.** *Sia $\to$ un sistema di riduzione astratto su A e $\epsilon \in \mathbb{R}_{>0}$. Una generica funzione $f : A \to \mathbb{R}_{>0}$ è una codifica di $\to$ in $[\geq \epsilon+]$ se e solo se f è una codifica della controparte probabilistica di $\to$, $\rightarrowtail$, in $[\geq \epsilon+\mathbb{E}]$.*

Dimostrazione. Siccome $f(a_i) = f(\{\!\{1 : a_i\}\!\})$ e $f(\varnothing) = 0$ le due nozioni di monotonia diventano equivalenti. □

In realtà, la controparte probabilistica di un ARS è essa stessa un caso degenere di ordine di ordine probabilisticamente monotono: basta fissare $\epsilon = 1$ e, in tutte le coppie della relazione ottenuta, $J = I$. Partire dal loro studio è quindi il passo più naturale verso il caso generale.

Partiamo dagli PMO via codifica: ci concentreremo in seguito su quelli semplici.

Teorema 3.3.1 (Terminazione quasi certa forte degli ordini probabilisticamente monotoni via codifica)**.** *Sia $\succ$ un sistema di riduzione astratto su A e $\delta \in \mathbb{R}_{>0}$. L'ordine probabilisticamente monotono indotto da $\succ$ via codifica $f : A \to \mathbb{R}_{>0}$ di $\succ$ in $[\geq \delta+]$, $>_{p\succ,f}$, è quasi certamente terminante forte.*

Dimostrazione. Si consideri $\sqsupset$, la controparte probabilistica di $\succ$. Per il precedente lemma, f codifica $\sqsupset$ in $[\geq \delta + \mathbb{E}]$. Sia inoltre $\epsilon \in \mathbb{R}_{>0}$ la costante usata per definire $>_{p\succ,f}$. Per chiusura di $\mathbb{R}_{>0}$ rispetto alla moltiplicazione, $\epsilon\delta \in \mathbb{R}_{>0}$.

Quindi $[\geq \epsilon\delta + \mathbb{E}]$ rispetta il vincolo di positività stretta della relativa definizione. Consideriamo ora un generico elemento di $>_{p\succ,f}$, che indicheremo con $(a, \{\!\{p_i : b_i \mid i \in I\}\!\})$ (per definizione di $>_{p\succ,f}$, la multidistribuzione ha necessariamente questa forma, e non abbiamo quindi perdita di generalità):

1. sia $i \in I$. Il primo congiunto della condizione espressa da $>_{p\succ,f}$ richiede che $p_i = 0 \vee f(a) \geq f(b_i)$. In entrambi i casi, osserviamo che quindi vale $p_i f(a) \geq p_i f(b_i)$ (siccome $p_i \geq 0$).

2. sia $J \subseteq I$ un insieme che soddisfa il secondo congiunto della condizione espressa da $>_{p\succ,f}$. Quindi $\sum_{j\in J} p_j \geq \epsilon \wedge \forall j \in J \quad a \succ b_j$. Fissiamo un generico indice $j \in J$. Da $a \succ b_j$ segue $a \sqsupset \{\!\{1 : b_j\}\!\}$. Sapendo che f è una codifica di $\sqsupset$ in $[\geq \delta + \mathbb{E}]$, otteniamo $f(a) \geq \delta + \mathbb{E}(f(\{\!\{1 : b_j\}\!\}))$ e perciò $f(a) \geq \delta + f(b_j)$. Poiché $p_j > 0$, vale che $p_j f(a) \geq p_j\delta + p_j f(b_j)$. Per ogni possibile $j \in J$, sommiamo membro a membro le istanze di quest'ultima disuguaglianza e otteniamo $\sum_{j\in J}(p_j f(a)) \geq \sum_{j\in J}(p_j\delta) + \sum_{j\in J}(p_j f(b_j))$. Per il primo congiunto della condizione imposta da $>_{p\succ,f}$, $\sum_{j\in J} p_j \geq \epsilon$ e, siccome $\delta \in \mathbb{R}_{>0}$, allora $(\sum_{j\in J} p_j)\delta \geq \epsilon\delta$. Quindi $\sum_{j\in J}(p_j f(a)) \geq \epsilon\delta + \sum_{j\in J}(p_j f(b_j))$.

Sommando membro a membro la disuguaglianza (2) con tutte le istanze della (1) per ogni $i \in I \setminus J$, si ottiene:

$$\sum_{j\in J}(p_j f(a)) + \sum_{i\in I\setminus J}(p_i f(a)) \geq \epsilon\delta + \sum_{j\in J}(p_j f(b_j)) + \sum_{i\in I\setminus J}(p_i f(b_i))$$

$$\sum_{i\in I}(p_i f(a)) \geq \epsilon\delta + \sum_{i\in I}(p_i f(b_i))$$

$$f(a)\sum_{i\in I} p_i \geq \epsilon\delta + \sum_{i\in I}(p_i f(b_i))$$

Ricordiamo che $>_{p\succ,f}$ è un PARS, e come tale i secondi membri dei suoi elementi sono sempre multidistribuzioni proprie ($\sum_{i\in I} p_i = 1$):

$$f(a) \geq \epsilon\delta + \sum_{i\in I}(p_i f(b_i)) = \epsilon\delta + \mathbb{E}(f(\{\!\{p_i : b_i \mid i \in I\}\!\}))$$

Quindi f è una codifica di $>_{p\succ,f}$ in $[\geq \epsilon\delta+\mathbb{E}]$, e cioè una funzione di Lyapunov. Perciò $>_{p\succ,f}$ è quasi certamente terminante forte. □

Ottenuto questo importante risultato, la terminazione quasi certa forte degli ordini probabilistici monotoni senza codifica segue come corollario.

Corollario 3.3.1 (Terminazione quasi certa forte degli ordini probabilisticamente monotoni senza codifica)**.** *Sia $\succ$ un sistema di riduzione astratto su A tale che*

$$\forall a \in A \quad dh_{\succ}(a) \in \mathbb{N}$$

. L'ordine probabilisticamente monotono indotto da $\succ$ in $[\geq \delta+]$, $>_{p\succ}$, è quasi certamente terminante forte.

Dimostrazione. Siccome $\forall a \in A \quad dh_{\succ}(a) \in \mathbb{N}$ e inoltre $\forall a, b \in A \quad (a \succeq b \rightarrow dh_{\succ}(a) \geq dh_{\succ}(b))$, $dh_{\succ}$ è di certo una codifica di A in $[\geq 1+]$ e inoltre, scelta per $>_{p\succ,dh_{\succ}}$ la stessa costante $\epsilon \in \mathbb{R}_{>0}$ usata per definire $>_{p\succ}$, vale che:

$$\forall a, b \in A \quad (a >_{p\succ} b \rightarrow a >_{p\succ,dh_{\succ}} b)$$

Dal Teorema 3.3.1, poi, segue che $>_{p\succ,dh_{\succ}}$ è SAST. Quindi, anche $>_{p\succ}$, che ne è un sottoinsieme, lo è. □

3.4 Esempio di non-applicazione: gli ordini di percorso lessicografici

L'unico vero limite della Definizione 3.2.2 è la necessarietà di una codifica in $[\geq \delta+]$. In altre parole, è richiesto partire sempre da ordini le cui controparti probabilistiche siano SAST. In questo paragrafo, esibiamo una famiglia di ordini di semplificazione estremamente popolari (gli ordini di percorso lessicografici) e mostriamo che, non essendo SAST, essi non inducono ordini probabilisticamente monotoni (con o senza codifica).

Gli ordini di percorso lessicografici appartengono alla famiglia degli ordini di percorso ricorsivi, che confrontano dapprima i simboli alla radice dei due termini in questione e poi, ricorsivamente, i loro sottotermini immediati. In

particolare, gli ordini di percorso lessicografici, come suggerisce il nome, considerano i sottotermini immediati come tuple ordinate i cui i-esimi elementi diventano rilevanti solo a parità di tutti quelli che li precedono. Se invece avessimo scelto di prescindere dall'ordine dei sottotermini, avremmo potuto effettuare uno studio degli ordini di percorso con multinsiemi.

Definizione 3.4.1 (Ordine di percorso lessicografico)**.** Siano Φ un insieme finito di simboli di funzione, Σ una segnatura per Φ, V un insieme di variabili e $>$ un ordine stretto su Σ. L'ordine di percorso lessicografico (o LPO, *lexicographic path order*) su $T(\Sigma, V)$ indotto da $>$, $>_{lpo}$, è la relazione su $T(\Sigma, V)$ che rispetta la seguente condizione ricorsiva lessicografica su (s, t):

$\forall s, t \in T(\Sigma, V), s >_{lpo} t \iff$

(LPO1) $t \in \mathcal{V}ar(s) \quad \wedge \quad s \neq t \quad \vee$

(LPO2) $s = f(s_1, \ldots, s_m) \quad \wedge \quad t = g(t_1, \ldots, g_n) \quad \wedge$

(LPO2a) $(\exists i \in \{1, \ldots m\} \quad s_i \geq_{lpo} t \quad \vee$

(LPO2b) $f > g \quad \wedge \quad \forall j \in \{1, \ldots, n\} \quad s >_{lpo} t_j \quad \vee$

(LPO2c) $f = g \quad \wedge \quad \forall j \in \{1, \ldots, n\} \quad s >_{lpo} t_j \quad \wedge$
$\exists i \in \{1, \ldots, m\} \quad (s_i >_{lpo} t_i \quad \wedge \quad \forall k \in \{1, \ldots, i-1\} \quad s_k = t_k))$

Si osserva che:

- $\geq_{lpo}$ è il quasi-ordine che consiste nella chiusura riflessiva di $>_{lpo}$: non potrebbe ovviamente essere l'ordine di percorso lessicografico indotto da $\geq$, che non è stretto;
- la ricorsione di cui sopra è ben definita, siccome usata solo su coppie di termini lessicograficamente più "piccole" di (s, t).

Per formalizzare questo concetto, Baader e Nipkow [3] partono dal presentare un'implementazione per un generico ordine lessicografico.

Programma 3.5: ordine lessicografico (`traat/orders.ML`, 14-21)

```
1 datatype order = GR | EQ | NGE;
2
3 (* lex: ('a * 'b -> order) -> 'a list * 'b list -> order *)
4 fun lex ord ([],[]) = EQ
```

```
5   | lex ord (x::xs,y::ys) = case ord(x,y) of
6         GR => GR
7       | EQ => lex ord (xs,ys)
8       | NGE => NGE;
```

Risulta necessaria anche la seguente definizione di `forall`.

Programma 3.6: quantificatore universale per liste (`traat/library.ML`, 22-24)

```
1 (* forall: ('a -> bool) -> 'a list -> bool *)
2 fun forall p []      = true
3   | forall p (x::xs) = p(x) andalso forall p xs;
```

L'ordine risultante è ovviamente definito sulle istanze di `term`, il tipo precedentemente definito:

Programma 3.7: ordine di percorso lessicografico (`traat/termorders.ML`, 17-31)

```
1
2 (* (string * string -> order) -> term * term -> order *)
3 fun lpo ord (s,t) = case (s,t) of
4       (s, V x) => if s = t then EQ
5                   else if occurs x s then GR (*LPO1*) else NGE
6     | (V _, T _) => NGE
7     | (T(f,ss), T(g,ts)) => (*LPO2*)
8         if forall (fn si => lpo ord (si,t) = NGE) ss
9         then case ord(f,g) of
10                GR => if forall (fn ti => lpo ord (s,ti) = GR) ts
11                      then GR (*LPO2b*) else NGE
12              | EQ => if forall (fn ti => lpo ord (s,ti) = GR) ts
13                      then lex (lpo ord) (ss,ts) (*LPO2c*)
14                      else NGE
15              | NGE => NGE
```

Come già anticipato, gli ordini di percorso lessicografici sono inadatti alla costruzione di un qualsiasi tipo di ordini probabilisticamente monotoni.

Teorema 3.4.1 (Esistenza di ordini di percorso lessicografici che non inducono ordini probabilisticamente monotoni via codifica)**.** *Esiste almeno un ordine di percorso lessico che non induce nessun ordine probabilisticamente monotono via codifica.*

Dimostrazione. Consideriamo, sempre nel contesto dell'Esempio 3.2.3, le seguenti procedure:

Programma 3.8: procedure `double`, `exp` e `wait` (`expwait.sml`, 3-13)

```
1 (* int -> int *)
2 fun double 0 = 0
3   | double (n + 1) = 2 + double n
4
5 (* int -> int *)
6 fun exp 0 = 1
7   | exp (n + 1) = double exp n
8
9 (* int -> int *)
10 fun wait 0 = 0
11   | wait (n + 1) = wait n
```

Possiamo formalizzarle aggiungendo *double*, *exp* e *wait* a Φ, con $\Sigma(double) = \Sigma(exp) = \Sigma(wait) = 1$. Aggiungiamo ora opportune nuove regole per descrivere il comportamento di questi nuovi simboli di funzione. Come di consueto, facciamo affidamento alla variabile ausiliaria n. Il simbolo di funzione *double* rappresenta il doppio del numero naturale passatogli come argomento:

$$double(0) \quad R \quad \{\!\{1 : 0\}\!\}$$

$$double(S(n)) \quad R \quad \{\!\{1 : S(S(double(n)))\}\!\}$$

Il simbolo di funzione *exp* rappresenta la potenza di due che ha per esponente il numero naturale passatogli come argomento:

$$exp(0) \quad R \quad \{\!\{1 : S(0)\}\!\}$$

$$exp(S(n)) \quad R \quad \{\!\{1 : double(exp(n))\}\!\}$$

Infine, il simbolo di funzione *wait* non rappresenta un valore particolare (la scelta di 0 nel programma Standard ML è solo un dettaglio implementativo). Ciò che è interessante è che $wait(n)$ venga portato in forma normale dopo esattamente $n + 1$ passi di riduzione:

$$wait(0) \quad R \quad \{\!\{1 : 0\}\!\}$$

$$wait(S(n)) \quad R \quad \{\!\{1 : wait(n)\}\!\}$$

Consideriamo ora un qualche ordine $>$ su Φ tale che $exp > double > S$. Chiamiamo l'ordine lessicografico da esso indotto $>_{lpo}$. Assumiamo che esso, a sua

volta, induca un ordine probabilisticamente monotono $>_{plpo}$ via codifica e ci riproponiamo di ridurci all'assurdo. Usando la Definizione 3.4.1 e la Definizione 3.2.2, è facile verificare che tutte le regole fino a qui presentate (incluse quelle che descrivono *geo*) appartengono a $>_{plpo}$. Quindi esistono sequenze di riduzione probabilistica ammesse da $>_{plpo}$ che rispettano la seguente forma:

$$\mu_n = \begin{cases} \{\!\{1 : wait(exp(geo(0)))\}\!\} & n = 0 \\ \{\!\{2^{-n} : wait(exp(geo(n))), 2^{-n} : wait(exp(n-1)), \dots\}\!\} & n > 0 \end{cases}$$

Per ottenerle, infatti, è sufficiente considerare a ogni passo di riduzione probabilistica l'applicazione dell'unica regola già citata specifica per *geo*. Possiamo stimare per difetto la lunghezza di derivazione attesa di un generico elemento μ di questa famiglia di sequenze: per ogni termine della sequenza con indice $n > 0$ possiamo contare sull'esistenza di un elemento $(2^{-n} : wait(exp(n-1)))$, che causerà a cascata almeno altri 2^{n-1} elementi caratterizzati dalla stessa probabilità 2^{-n}. Quindi:

$$edl_{>_{plpo}}(\mu) = \sum_{n\in\mathbb{N}} |\mu_i| \geq \sum_{n\in\mathbb{N}_{>0}} (2^{-n}2^{n-1}) = \sum_{n\in\mathbb{N}_{>0}} \frac{1}{2} = +\infty$$

Il PTRS in questione, quindi, non è PAST e, a maggior ragione, non è perciò di certo neanche SAST. Essendo esso un ordine probabilisticamente monotono, questo è in contraddizione con il Teorema 3.3.1. Assurdo. □

Ovviamente, questo risultato si applica anche in assenza di codifiche.

Corollario 3.4.1 (Esistenza di ordini di percorso lessicografici che non inducono ordini probabilisticamente monotoni senza codifica)**.** *Esiste almeno un ordine di percorso lessico che non induce nessun ordine probabilisticamente monotono senza codifica.*

Dimostrazione. Per il precedente teorema, esiste un ordine di percorso lessicografico che non induce ordini probabilisticamente monotoni via codifica. Quindi il corollario è banalmente dimostrato, perché se esso inducesse un tale ordine probabilisticamente monotono senza codifica, potremmo costruirne uno

via codifica in modo analogo a quando fatto nella prova del Corollario 3.3.1, contraddicendoci. □

3.5 Conclusioni

Nel corso del nostro lavoro, abbiamo ripercorso le nozioni preliminari per trattare i sistemi di riscrittura dei termini probabilistici. Abbiamo definito gli ordini probabilisticamente monotoni, uno strumento per costruire sistemi di riduzione astratti probabilistici a partire da sistemi di riduzione astratti tradizionali. Per entramble le varianti (fra esse non equivalenti) proposte, abbiamo provato che tutti gli ordini probabilisticamente monotoni sono quasi certamente terminanti forti, con tanto di esempi applicativi. Abbiamo anche ottenuto costruttivamente un risultato negativo: non tutti gli ordini di percorso lessicografici sono adatti a indurre ordini probabilisticamente monotoni di un qualche tipo.

Bibliografia

1. G. Agha, J. Meseguer, e K. Sen. «PMaude: Rewrite-based Specification Language for Probabilistic Object Systems». *Electronic Notes in Theoretical Computer Science* 153:2, 2006. Proceedings of the Third Workshop on Quantitative Aspects of Programming Languages (QAPL 2005), pp. 213–239. ISSN: 1571-0661. DOI: https://doi.org/10.1016/j.entcs.2005.10.040. URL: https://www.sciencedirect.com/science/article/pii/S1571066106002672.

2. M. Avanzini, U. Dal Lago, e A. Yamada. «On probabilistic term rewriting». *Science of Computer Programming* 185, 2020, p. 102338. ISSN: 0167-6423. DOI: https://doi.org/10.1016/j.scico.2019.102338. URL: https://www.sciencedirect.com/science/article/pii/S0167642319301339.

3. F. Baader e T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998. DOI: 10.1017/CBO9781139172752.

4. O. Bournez e F. Garnier. «Proving Positive Almost-Sure Termination». In: *Term Rewriting and Applications*. A cura di J. Giesl. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 323–337. ISBN: 978-3-540-32033-3.

5. T. Evans. «On multiplicative systems defined by generators and relations: II. Monogenic loops». *Mathematical Proceedings of the Cambridge Philosophical Society* 49:4, 1953, pp. 579589. DOI: 10.1017/S0305004100028772.

6. D. E. Knuth e P. B. Bendix. «Simple Word Problems in Universal Algebras». In: *Automation of Reasoning: 2: Classical Papers on Computational Logic 1967–1970*. A cura di J. H. Siekmann e G. Wrightson. Springer Berlin Heidelberg, Berlin, Heidelberg, 1983, pp. 342–376. ISBN: 978-3-642-81955-

1. DOI: 10.1007/978-3-642-81955-1_23. URL: https://doi.org/10.1007/978-3-642-81955-1_23.

7. D. MacQueen, R. Harper, J. Reppy, e L. Bergstrom. *Standard ML family github project*. 2014. URL: https://smlfamily.github.io/.

Ringraziamenti

Il lavoro che avete appena letto corona simbolicamente il mio percorso triennale presso l'Università di Bologna. Sono quindi in obbligo di ricordare chiunque mi abbia accompagnato in questi tre anni.

Il Professor Ugo Dal Lago, mio relatore, è stato con me estremamente comprensivo e di aiuto in questi mesi di lavoro, sia per la stesura dell'elaborato che nel mio percorso di orientamento in uscita.

Ogni collega ha concorso, in propria misura, alla mia crescita. Ricordo in particolare Matti, mio modello di studente lavoratore, Luca, che mi ha mostrato come la formazione fra pari sia importante quanto quella impartita dal corpo docente, e Andreea, che mi ha insegnato che l'università si costruisce sulle persone, e non sui programmi. Chiunque ha contribuito e sta tuttora contribuendo al progetto di *CSUnibo*, ai miei occhi, concretizza gli ideali di un'istruzione superiore umanamente fondata sulla cooperazione e di una società equa basata sull'accesso aperto all'informazione.

Insieme al Prof. Dal Lago, tengo a citare anche i Proff. Sacerdoti Coen e Davoli: hanno confermato in me il desiderio di diventare un professore comprensivo ancor prima di un ricercatore capace.

Lasciando il mio appartamento, saluto con rammarico Dan, che mi ha sopportato sia in aula che in camera doppia, Fede e Ale.

I gruppi della *Federazione Universitaria Cattolica Italiana* di Bologna e di tutta Italia mi hanno accompagnato nel mio cammino di ricerca. Le magiche volontarie del circolo ludico LGBT+ de *La Gilda* del *Cassero* mi hanno guidato nella più bella esperienza di volontariato. Le mie colleghe e i miei colleghi in *devDept* hanno avuto estrema pazienza con me in ufficio. La famiglia della *Calisthenics NoGravity* mi ha offerto una valvola di sfogo.

Le due ragazze a cui questa tesi è dedicata non hanno bisogno dei miei encomi: a entrambe auguro tutto il bene che mi hanno saputo offrire.

Sono estremamente in debito con ciascuna di queste persone; a tutte loro, il mio più umile grazie. Ciò che costruirò da qui in avanti sarà realtà solo grazie a loro.