Alma Mater Studiorum · Università di Bologna

facoltà di scienze matematiche, fisiche e naturali

Corso di Laurea Magistrale in Informatica

INDAGINI IN DEEP INFERENCE

Tesi di Laurea in Tipi e Linguaggi di Programmazione

Relatore:
Chiar.mo Prof.
Simone Martini

Presentata da:
Andrea Simonetto

II Sessione
Anno Accademico 2009/2010

Alla venerata memoria
di mio nonno, Gino Simonetto.

Wir müssen wissen. Wir werden wissen.
(David Hilbert)

Introduzione

La matematica ama parlare di sé stessa. Dalla teoria dei numeri – ritenuta da Gauss la "regina della matematica" – all'analisi – l'hilbertiana "sinfonia coerente dell'universo" – la matematica pura è abituata a porsi ad oggetto delle proprie speculazioni. Anche la *logica matematica* possiede spiccate tendenze narcisistiche[1]; nondimeno, vista la sua propensione ad affrontare questioni di fondamento, è certamente una delle branche che ama maggiormente parlare di matematica. Tra i vari contributi è doveroso citare, come esempi eccellenti, l'*aritmetica di Peano* e la *teoria assiomatica degli insiemi* di Zermelo–Fraenkel.

Uno dei concetti cardine di tutta la matematica è quello di *dimostrazione*. Per studiare questi oggetti, i logici hanno sviluppato un'infrastruttura nota come *teoria della dimostrazione*. In questa tesi faremo una panoramica sulla teoria della dimostrazione, concentrandoci su uno degli sviluppi più recenti, una tecnica nota come *deep inference*.

La deep inference è una nuova *metodologia* in proof theory utile a progettare *famiglie di sistemi formali* (o *formalismi*) con buone proprietà, quali:

- l'*efficienza* nella rappresentazione delle dimostrazioni: alcuni sistemi rendono disponibili dimostrazioni più brevi di quanto possano fare altri (questi aspetti sono studiati in *complessità delle dimostrazioni* o *proof complexity*);

[1] Secondo Locke, la logica è "l'anatomia del pensiero". Occorre tuttavia precisare che usando il termine "logica", Locke intendeva quella parte di filosofia che studia il ragionamento e l'argomentazione; la logica matematica cominciò a fiorire solo un secolo e mezzo dopo la sua morte.

- *analiticità*: alcuni sistemi vengono naturalmente con algoritmi di *ricerca delle dimostrazioni* (o di *proof search*) *efficienti*, altri no, altri ancora solo con alcuni accorgimenti. L'analiticità è la proprietà chiave per ottenere algoritmi di proof search efficienti;

- l'abilità di esprimere dimostrazioni che sono matematicamente naturali, cioè senza artefatti sintattici "amministrativi" (si parla in questi casi di *burocrazia delle dimostrazioni*). Uno dei problemi di ricerca principali in proof theory è trovare una buona corrispondenza tra le dimostrazioni e il loro significato. In particolare, il problema dell'*identità delle dimostrazioni* è prominente, e consiste nel trovare nozioni di equivalenza tra dimostrazioni non banali, supportate da semantiche appropriate alle dimostrazioni e ai sistemi formali.

I *formalismi* controllano, in larga parte, la progettazione delle regole d'inferenza. Per esempio, la deduzione naturale prescrive che, per ogni connettivo, siano date due regole: una d'introduzione e una di eliminazione. In tutti i formalismi tradizionali (ma anche in quelli più moderni derivati dai primi), viene adottato un meccanismo noto come *shallow inference* o *inferenza di superficie*, nel quale le regole di inferenza operano sui connettivi più prossimi alla radice delle formule – quando vengono viste come alberi, cioè quando ci si concentra sulla loro *struttura sintattica*.

L'inferenza di superficie è una metodologia molto naturale, poiché permette di procedere per induzione strutturale diretta sulle formule. Tuttavia non è ottimale riguardo alcune proprietà dei sistemi formali, quali quelle sopra menzionate. In particolare:

- sembra sia incapace di fornire formalismi analitici che siano efficienti riguardo la complessità delle dimostrazioni;

- le dimostrazioni tendono ad avere molta burocrazia, cioè rappresentazioni sintatticamente complesse degli argomenti matematici.

Inoltre la shallow inference fatica a relazionarsi con le logiche modali. Teorie logiche modali possono essere definite nei sistemi di Frege-Hilbert, ma

ottenere analiticità per essi è una sfida molto ardua, in alcuni casi ancora irrisolta. In più è altrettanto difficile (se non addirittura impossibile) esprimere sistemi formali per logiche non-commutative.

Nel seguito mostreremo alcuni tra i maggiori risultati di teoria della dimostrazione ottenuti mediante un formalismo deep inference, noto come *calcolo delle strutture*. Il calcolo delle strutture è un contributo importante nello sviluppo della metodologia deep inference, per la sua semplicità e la sua somiglianza coi formalismi tradizionali. Grazie al calcolo delle strutture, sono stati ottenuti i seguenti risultati:

- la logica classica, intuizionista, lineare a alcune logiche modali possono essere espresse in sistemi che godono di analiticità;

- è possibile esprimere logiche lineari munite di *operazioni non-commutative* in sistemi analitici, e si dimostra che queste logiche non possono essere formalizzate analiticamente nel calcolo dei sequenti; inoltre questi sistemi logici sono in forte corrispondenza con le algebre di processo;

- sono state sviluppate tecniche nuove e generali di normalizzazione, e sono state scoperte *nozioni del tutto nuove* di normalizzazione, in aggiunta alla tradizionale *cut elimination*;

- la maggior parte dei sistemi sviluppati sono costituiti interamente da regole d'inferenza *locali*; una regola d'inferenza locale ha *complessità computazionale costante*. La località è una proprietà difficile da conseguire, e non è ottenibile nel calcolo dei sequenti per la logica classica;

- i sistemi ottenuti sono estremamente modulari; questo significa una *forte indipendenza tra le regole d'inferenza*;

- moti sistemi sono stati implementati, grazie a tecniche che producono regole d'inferenza atte a migliorare l'efficienza senza sacrificare le proprietà teoriche;

- tutti i sistemi ottenuti sono semplici, nel senso che le regole d'inferenza sono *contenute e intelligibili*.

Il calcolo delle strutture è una generalizzazione di molti formalismi shallow inference, in particolare del calcolo dei sequenti. Questo significa che ogni dimostrazione data in questi formalismi shallow inference, può essere "mimata" nel calcolo delle strutture, preservandone la complessità e senza perdita di proprietà strutturali.

Indice

Elenco delle figure

Capitolo 1

Formalismi e metodologie

La teoria della dimostrazione nacque sul finire del XIX secolo, ad opera di David Hilbert e dei suoi collaboratori. Fu un periodo in cui la matematica visse una crisi senza precedenti, intaccata in prossimità dei suoi fondamenti logici dal manifestarsi di una serie di paradossi, proprio mentre la scuola intuizionista di Brouwer ne metteva in dubbio alcuni princìpi filosofici basilari, fatti che sommati minavano alla base la maggior parte della matematica esistente.

Tuttavia, forti evidenze empiriche suffragavano la matematica conosciuta, e molti matematici rifiutarono di abbandonarla o rifondarla: tra loro, Hilbert avanzò un programma di salvataggio completo. Egli propose di formulare la matematica classica come teoria formale assiomatica, e in seguito di provarne la *consistenza* (ossia la non contraddittorietà).

Prima della proposta di Hilbert, la consistenza di teorie assiomatiche veniva provata esibendo un "modello": data una teoria assiomatica, un *modello* è un sistema di oggetti, presi da qualche altra teoria, tali da soddisfare gli assiomi, cioè, ad ogni oggetto o nozione primitiva della teoria assiomatica, viene fatto corrispondere un oggetto o una nozione dell'altra teoria, in modo tale che gli assiomi corrispondano a teoremi nell'altra teoria. Se l'altra teoria è consistente, anche quella assiomatica deve esserlo. Un esempio famoso è dato dalla dimostrazione di Beltrami (1868) della consistenza della geome-

tria iperbolica: egli provò che le rette nel piano non-euclideo della geometria iperbolica, potevano essere rappresentate dalle geodesiche su una superficie di curvatura costante e negativa nello spazio euclideo. Da questo concluse che il piano iperbolico dev'essere consistente, a patto che la geometria euclidea lo sia.

È chiaro che il metodo del modello è relativo. La teoria assiomatica è consistente solo se il suo modello lo è. Ma per provare l'assoluta consistenza della matematica classica, il metodo dei modelli non offriva speranze: nessuna teoria matematica era accettabile come modello, poiché da ognuna di esse sarebbe fatalmente riemerso il problema di partenza, cioè dimostrarne la consistenza.

Hilbert propose di affrontare il problema in maniera diretta: per provare la consistenza di una teoria, si deve dimostrare al suo interno una proposizione sulla teoria stessa, cioè un teorema su tutte le possibili dimostrazioni della teoria. La branca di matematica che si occupa di questi aspetti di formalizzazione e riflessione, venne battezzata da Hilbert "metamatematica" o "teoria della dimostrazione".

Purtroppo il sogno di Hilbert s'infranse nel 1931 con il Secondo Teorema d'Incompletezza di Gödel [1931], che enuncia l'impossibilità per sistemi abbastanza espressivi da formalizzare l'aritmetica di dimostrare la propria consistenza: purtroppo la quasi totalità della matematica da salvare, passava per l'aritmetica. Tuttavia la teoria della dimostrazione sopravvisse a questo scossone, diventando importante in vari ambiti, tra i quali l'informatica.

In informatica, i dimostratori automatici di teoremi richiedono uno studio della struttura combinatoriale delle dimostrazioni, mentre nella programmazione logica la deduzione è usata come fondamento della computazione. Inoltre esistono forti connessioni tra sistemi logici e linguaggi di programmazione funzionali, e tecniche di proof theory sono state utilizzate per porre dei vincoli di complessità computazionale ad alcuni di questi linguaggi (ad esempio in Girard [1995a]).

Uno dei princìpi fondamentali in proof theory è che la formalizzazione

di una teoria richiede una totale astrazione dal significato, cioè un *sistema formale* dovrebbe essere una mera manipolazione simbolica spogliata di ogni interpretazione semantica. Dato un sistema formale, distinguiamo il livello rigoroso del sistema stesso (o *livello oggetto*), dal livello in cui esso viene studiato (il *meta-livello*) espresso nel linguaggio della matematica intuitiva e informale.

Inoltre, per essere convincenti, gli strumenti usati nelle meta-teorie dovrebbero essere ristretti a tecniche – chiamate *finitarie* dai formalisti, o, in un accezione più moderna, *combinatorie* – che impiegano solo oggetti intuitivi e processi effettivi (in accordo con la scuola intuizionista). Nessuna classe infinita di oggetti deve poter essere trattata come un "tutto"; le prove di esistenza dovrebbero sempre esibire, almeno implicitamente, un testimone.

La proof theory è dunque una collezione di meta-teorie finitarie, espresse nel linguaggio ordinario e con l'ausilio di simboli matematici – come variabili di meta-livello (o *meta-variabili*) introdotte ove necessario – tali da caratterizzare le proprietà dei vari sistemi formali. In questo capitolo introdurremo le nozioni di base della teoria della dimostrazione, partendo da insiemi finiti e generando quelli infiniti con procedure effettive, calcolabili.

Quella di cui abbiamo discusso finora, va oggi sotto il nome di teoria della dimostrazione *strutturale*, cioè un'analisi combinatoriale della struttura delle dimostrazioni formali; gli strumenti centrali sono il *Teorema di eliminazione del taglio* e quello di *normalizzazione*.

Il percorso che seguiremo in questo capitolo è liberamente ispirato a Kleene [1952], e si articola in quattro sezioni, le prime tre piuttosto standard, mentre la quarta aggiunta appositamente per trattare il tema della tesi, ossia la deep inference. Nell'ordine:

1. si definirà uno strumento linguistico formale, in grado di produrre dei *linguaggi oggetto* che costituiranno gli elementi base della logica da indagare;

2. sarà introdotto un livello linguistico formale superiore (o *meta-livello*),

tale da permetterci di ragionare sui vari linguaggi oggetto, e saranno date le definizioni e gli strumenti di indagine basilari;

3. verranno formalizzati i concetti di *deduzione* e di *dimostrazione*, a partire da un generico *linguaggio oggetto*, usando gli strumenti del *metalivello*, e saranno presi in esame i *formalismi* (i.e. le famiglie di sistemi) esprimibili con tali strumenti;

4. si definirranno e si metteranno a confronto le due *metodologie di inferenza*: di superficie e di profondità (shallow *versus* deep inference).

Affronteremo il tutto sempre tenedo presente il vincolo di effettiva costruibilità delle procedure e la caratterizzazione combinatoria delle tecniche e degli strumenti via via introdotti, in pieno stile formalista.

1.1 Linguaggi formali

In questa sezione svilupperemo le basi di linguaggi formali che utilizzeremo da qui in avanti. Alcuni concetti saranno forniti in maniera intuitiva, altri in modo più preciso: per approfondimenti su linguaggi formali e grammatiche, si rimanda ad Aho and Ullman [1972]; Aho et al. [2006].

Definizione 1.1.1 (Alfabeti, stringhe, linguaggi, sottolinguaggi)**.** *Un* alfabeto Σ *è un insieme finito non vuoto di* simboli. *Una* stringa su un alfabeto[1] Σ *è una sequenza finita di simboli scelti da* Σ. *La* stringa vuota ϵ *è la stringa composta da zero simboli; essa è una stringa su qualunque alfabeto.*

Siano $x = a_1 \cdots a_n$ *e* $y = b_1 \cdots b_m$ *due stringhe su un alfabeto* Σ*: la loro* concatenazione *(si denota giustapponendo* x *a* y*) è la stringa* $xy = a_1 \cdots a_n b_1 \cdots b_m$. *In particolare, per ogni stringa* w*, si ha* $\epsilon w = w\epsilon = w$.

[1] Per convenzione useremo le lettere minuscole prese dall'inizio dell'alfabeto inglese per denotare i simboli, e le lettere minuscole alla fine dell'alfabeto, di solito w, x, y, z, per denotare le stringhe.

Dato un alfabeto Σ*, definiamo:*

$$\begin{array}{lcll} \Sigma^0 & = & \{\epsilon\} & \textit{(Stringa di lunghezza 0)} \\ \Sigma^{n+1} & = & \{aw \mid a \in \Sigma, w \in \Sigma^n\} & \textit{(Stringhe di lunghezza } n+1\textit{)} \\ \Sigma^* & = & \bigcup_{n\in\mathbb{N}} \Sigma^n & \textit{(Stringhe su } \Sigma\textit{)} \end{array}$$

Un linguaggio $\mathscr{L}$ su un alfabeto Σ *è un'insieme di stringhe su* Σ *(cioè* $\mathscr{L} \subseteq \Sigma^*$*). Infine, dato un linguaggio* $\mathscr{L}$*, si definisce* sottolinguaggio di $\mathscr{L}$ *qualunque insieme di stringhe* $\mathscr{L}' \subseteq \mathscr{L}$*.*

Definizione 1.1.2 (Grammatiche, linguaggio generato)**.** *Una grammatica è una quadrupla* $G = (\Sigma, \mathcal{C}, S, \mathcal{P})$*, dove:*

- Σ *è un alfabeto di* simboli grammaticali*;*
- $\mathcal{C} \subseteq \Sigma$ *è un insieme di simboli, detti* categorie sintattiche *(o* simboli non terminali*, in contrapposizione con gli altri simboli grammaticali* $\Sigma \setminus \mathcal{C}$*, chiamati* simboli terminali*);*
- $S \in \mathcal{C}$ *è una particolare categoria sintattica, chiamata* simbolo iniziale*, che rappresenta il linguaggio da definire;*
- $\mathcal{P} \subseteq \Sigma^* \times \Sigma^*$ *è un insieme di coppie di stringhe* $(\alpha, \beta) \in \Sigma^* \times \Sigma^*$*, chiamate* produzioni grammaticali*.* α *è chiamata* testa della produzione*, mentre* β *è il* corpo della produzione*.*

Data una grammatica $(\Sigma, \mathcal{C}, S, \mathcal{P})$*, la* riscrittura ad un passo *(*$\rightsquigarrow$*) è un'applicazione di una delle produzioni in* $\mathcal{P}$*. Formalmente: siano* $\alpha, \beta, x, y \in \Sigma^*$*, allora:*

$$x\alpha y \rightsquigarrow x\beta y \quad \textit{sse} \quad \textit{esiste } (\alpha, \beta) \in \mathcal{P}$$

La riscrittura multipasso *(*$\rightsquigarrow^*$*) è la chiusura riflessiva e transitiva di quella ad un passo:*

$$w_1 \rightsquigarrow^* w_n \quad \textit{sse} \quad w_1 \rightsquigarrow \cdots \rightsquigarrow w_n \quad \textit{per qualche } n \geq 0$$

In particolare $w \rightsquigarrow^* w$ *per ogni* $w \in \Sigma^*$*.*

Il linguaggio generato da una grammatica $G = (\Sigma, \mathcal{C}, S, \mathcal{P})$ *(denotato con* $\mathscr{L}_G$*) è l'insieme delle stringhe di simboli terminali ottenibili tramite riscrittura multipasso a partire dal simbolo iniziale. In simboli:*

$$\mathscr{L}_G = \{w \in (\Sigma \smallsetminus \mathcal{C})^* \mid S \rightsquigarrow^* w\}$$

Definizione 1.1.3 (Grammatiche context-free, BNF)**.** *Sia* $G = (\Sigma, \mathcal{C}, S, \mathcal{P})$ *una grammatica. Una* regola di produzione *è* context-free *se è della forma* (P, β) *con* $P \in \mathcal{C}$ *e* $\beta \in \Sigma^*$*. Una* grammatica *si dice* context-free *se ogni sua regola di produzione è context-free.*

Un modo compatto ed elegante per scrivere le regole di produzione grammaticale, è quello di usare la forma di Backus-Naur *o BNF (Backus et al. [1960]). Sia* $G = (\Sigma, \mathcal{C}, S, \mathcal{P})$ *una grammatica e sia:*

$$\mathcal{P} = \{(\alpha_1, \beta_{1,1}), \ldots, (\alpha_1, \beta_{1,n_1}), \ldots, (\alpha_k, \beta_{k,1}), \ldots, (\alpha_k, \beta_{k,n_k}), \ldots\}$$

il suo insieme di produzioni. Allora $\mathcal{P}$ *in BNF si rappresenta come segue:*

$$\begin{array}{lll} \alpha_1 & ::= & \beta_{1,1} \mid \beta_{1,2} \mid \cdots \mid \beta_{1,n_1} \\ \cdots & \cdots & \cdots \\ \alpha_k & ::= & \beta_{k,1} \mid \cdots \mid \beta_{k,n_k} \\ \cdots & \cdots & \cdots \end{array}$$

L'ultimo (ma non ultimo) strumento sintattico che consideriamo, serve per fare emergere le *profonde simmetrie* che soggiacciono ai sistemi logici formali, ed è uno degli strumenti più usati in proof theory: il sequente. I sequenti sono una notazione sintattica, finalizzata ad inserire le formule logiche in ambienti adatti al ragionamento logico-deduttivo. Più precisamente:

Definizione 1.1.4 (Sequente)**.** *Dato un linguaggio* $\mathscr{L}$*, un* sequente *è un espressione del tipo:*

$$\Gamma \vdash \Delta$$

dove Γ, Δ *sono* liste finite *(eventualmente vuote) di stringhe di* $\mathscr{L}$ *– con le usuali operazioni definite sulle liste:* Γ, P *è l'aggiunta di una stringa* P *in*

coda ad una lista Γ*, mentre* Γ, Γ' *è la concatenazione delle liste* Γ *e* Γ'*; non ci sono simboli per la lista vuota.*

Il simbolo $\vdash$ *è noto come* turnstile *o* tornello *e fu originariamente introdotto in Frege [1879].*

L'idea intuitiva è che il sequente afferma (ipotizza) la deducibilità di almeno una formula logica in Δ a partire dalle premesse in Γ. Se $\Gamma = P_1, \ldots, P_n$ e $\Delta = Q_1, \ldots, Q_m$, il sequente $\Gamma \vdash \Delta$ è da intendersi come:

$$\textbf{Se} \quad P_1 \ e \cdots e \ P_n \quad \textbf{allora} \quad Q_1 \ \textit{oppure} \cdots \textit{oppure} \ Q_m$$

dove i significati di *"Se ... allora ..."*, *"e"* ed *"oppure"* devono essere resi espliciti in maniera formale.

Il sequente avente una lista vuota alla *destra* del tornello ($\Gamma \vdash$), afferma l'*inconsistenza delle premesse*, quello avente la lista vuota alla *sinistra* del tornello ($\vdash \Delta$), afferma che Δ è un *teorema*, ossia che è vero a prescindere da ogni premessa. Il *sequente vuoto* (cioè avente liste vuote alla *destra* ed alla *sinistra* del tornello) *afferma il falso* (se in un sistema logico-formale si è in grado di *dimostrare il falso*, allora esso è *inconsistente*).

1.2 Meta-livello

In questa sezione introdurremo i principali strumenti meta-linguistici: in genere nei testi di logica questo aspetto è lasciato perlopiù ad un livello intuitivo. Ho cercato, con questa presentazione originale, di renderli più formali perché, sebbene spesso sottovalutati, ritengo che offrano alcuni interessanti spunti di riflessione.

Osserviamo come, data una grammatica:

$$G = (\Sigma, \mathcal{C} = \{S_1, \ldots, S_n\}, S_i, \mathcal{P})$$

al variare di $i \in \{1, \ldots, n\}$ si producano linguaggi $\mathscr{L}_i$ diversi, seppur correlati tra loro, in funzione di quale categoria sintattica scegliamo come simbolo iniziale.

Definizione 1.2.1 (Meta-variabili, meta-linguaggi, (sotto)formule)**.** *Per ogni categoria sintattica S_i, definiamo un insieme finito $\mathscr{M}_i$ di* meta-variabili *di categoria S_i, che sono dei "segnaposti" per una qualche stringa di $\mathscr{L}_i$. Grazie alle meta-variabili, possiamo imporre dei vincoli sulla forma delle stringhe di $\mathscr{L}_G$.*

Il meta-livello linguistico *$\mathscr{L}'_G$ è quello in cui, giunti ad un certo passo di riscrittura, sostituiamo ad ogni occorrenza di simboli non terminali, una meta-variabile di categoria corrispondente. Il* meta-alfabeto *è composto dai terminali e dalle meta-variabili:*

$$\Sigma' = \Sigma \setminus \mathcal{C} \cup \bigcup_{1 \leq i \leq n} \mathscr{M}_i$$

mentre il meta-linguaggio *di G è definito da:*

$$\mathscr{L}'_G = \bigcup \{\varphi(w) \mid S \rightsquigarrow^* w\}$$

dove $\varphi(w)$ sta per "qualunque sostituzione di metavariabili al posto dei simboli non terminali in w". In simboli, se a denota un terminale:

$$\begin{aligned} \varphi &: \Sigma^* \to \mathscr{P}(\Sigma'^*) \\ \varphi(\epsilon) &= \{\epsilon\} \\ \varphi(aw) &= \{ay \mid y \in \varphi(w)\} \\ \varphi(S_i w) &= \{\sigma y \mid \sigma \in \mathscr{M}_i, y \in \varphi(w)\} \end{aligned}$$

Usiamo l'appellativo formula *per riferirci alle stringhe del meta-linguaggio $\mathscr{L}'_G$. Data una formula w appartenente a $\mathscr{L}'_G$, per* sottoformula *s'intende una qualunque porzione di w, che sia a sua volta compresa in $\mathscr{L}'_G$.*

È immediato dimostrare che, data una grammatica G, il meta-linguaggio è più ricco del linguaggio, cioè che, per ogni G:

$$\mathscr{L}_G \subset \mathscr{L}'_G$$

Infatti, al meta-linguaggio appartengono banalmente tutte le stringhe di $\mathscr{L}_G$ (se spingiamo la riscrittura fino a produrre stringhe di terminali, la funzione

φ non fa niente), mentre in $\mathscr{L}_G$ non ci sono le meta-variabili e quindi è strettamente incluso.

Le meta-variabili si *istanziano* a stringhe del linguaggio oggetto tramite *unificazione*, grazie alla quale è anche possibile eseguire delle *istanze parziali* tra meta-variabili e altre formule del meta-linguaggio.

Osserviamo che i linguaggi sono insiemi infiniti, così come il loro meta-livello genera insiemi infiniti. Tuttavia la base da cui sono generati questi insiemi (la grammatica) è finita e la procedura di generazione è concreta. Inoltre per grammatiche context-free verificare se una stringa appartiene o meno al linguaggio generato è un problema decidibile (in tempo polinomiale), e l'operazione di unificazione è anch'essa effettiva. Insomma, tutti gli strumenti dati fin qui sono *finitari*, in pieno stile formalista.

Il meta-linguaggio ci permette di ragionare induttivamente (ricorsivamente) sulla struttura delle stringhe di un linguaggio. Normalmente l'induzione è concentrata sulla parte più esterna delle formule, cioè su quella di superficie: ma la metodologia deep inference aggiunge qualcosa in più.

Definizione 1.2.2 (Contesti, saturazione, ordine)**.** *Data una grammatica* $G = (\Sigma, \mathcal{C}, S, \mathcal{P})$*, il* linguaggio dei contesti su G*, denotato con* Ξ_G*, è definito come il linguaggio generato dalla grammatica aumentata* $(\Sigma \cup \{\bullet\}, \mathcal{C}, S, \mathcal{P} \cup \{(S, \bullet)\})$*, dove* $\bullet \notin \Sigma$ *è un nuovo simbolo terminale chiamato* contesto vuoto*. Un* contesto (generico) *è una stringa di* $\mathbb{C} \in \Xi_G$ *(si indica spesso con* $\mathbb{C}\{\bullet\}$ *per enfatizzare il fatto che è un contesto).*

Intuitivamente un contesto è una stringa di $\mathscr{L}_G$ *con alcuni "buchi" (denotati da* $\bullet$*) che possono a loro volta essere riempiti con stringhe di* $\mathscr{L}_G$*. L'operazione di* saturazione di un contesto $\mathbb{C}$ con una stringa $w \in \mathscr{L}_G$ *si indica con* $\mathbb{C}\{w\}$*, e consiste nella sostituzione testuale di* w *al posto di tutte le occorrenze di* $\bullet$ *dentro* $\mathbb{C}$*; l'*ordine di un contesto *(in simboli* $\|\mathbb{C}\|$*) è il numero di occorrenze di* $\bullet$ *al suo interno. Ambedue si definiscono formalmente per induzione sulla struttura di* $\mathbb{C}$ *come mostrato in Figura 1.1.*

Infine, sia $n \geq 0$ *un numero naturale: il* linguaggio dei contesti di ordine

Saturazione contesti			**Ordine contesti**		
$\epsilon\{w\}$	$=$	ϵ	$\|\epsilon\|$	$=$	0
$(\bullet y)\{w\}$	$=$	$w(y\{w\})$	$\|\bullet\, y\|$	$=$	$1 + \|y\|$
$(ay)\{w\}$	$=$	$a(y\{w\})$	$\|ay\|$	$=$	$\|y\|$

Figura 1.1: Definizioni induttive di *saturazione* e *ordine* di un contesto

n su G *(i.e.* Ξ^n_G*) è così definito:*

$$\Xi^n_G = \{\mathbb{C} \in \Xi_G \mid \|\mathbb{C}\| = n\}$$

Data una grammatica $G = (\Sigma, \mathcal{C}, S, \mathcal{P})$, osserviamo che si ha $\Xi^0_G = \mathscr{L}_G$, poiché per produrre le stringhe di Ξ^0_G non si usa mai la regola di produzione aggiuntiva $(S, \bullet)$, ma solo quelle in $\mathcal{P}$, esattamente come accade per $\mathscr{L}_G$. Usando un argomento, analogo è possibile dimostrare che:

$$\mathscr{L}_G = \{\mathbb{C}\{w\} \mid \mathbb{C} \in \Xi^1_G, w \in \mathscr{L}_G\}$$

Infatti i contesti di Ξ^1_G sono generati usando una (e una sola) volta la regola di produzione aggiuntiva $(S, \bullet)$; in G, dove questa regola non è presente, tutto quello che è possibile fare è riscrivere S con una delle altre produzioni di $\mathcal{P}$ per S, che equivale a sostituire *quella* occorrenza di S con una delle stringhe del linguaggio generato da G, cioè proprio $\mathscr{L}_G$.

Per $n > 1$ non è possibile ottenere risultati analoghi su Ξ^n_G, poiché se da un lato è possibile riscrivere occorrenze diverse di S in modi diversi, dall'altro la saturazione di un contesto ammette un solo parametro in $\mathscr{L}_G$ (che viene replicato sempre uguale n volte). Inoltre $n = 0$ è un caso triviale, perché la saturazione dei contesti in Ξ^0_G non produce effetti (non si fanno sostituzioni). L'unico caso degno di nota è $n = 1$: esso rappresenta il punto di contatto tra il concetto di saturazione di un contesto – i.e. "sostituzione di *una* variabile (fresca)" – e quello più generale di riscrittura: saturazione di un contesto e riscrittura multipasso coincidono per $n \leq 1$, cioè quando il processo di riscrittura è sostituito da quello di saturazione *al più in un singolo punto.*

I contesti di ordine 1 su una grammatica sono uno strumento molto potente che ci consente di focalizzare l'attenzione su una porzione specifica di una stringa del linguaggio che dipende dalla sua struttura sottostante, astraendoci dal resto. Per la loro rilevanza, d'ora in poi quando parleremo di *contesti* intenderemo sempre quelli di ordine 1.

Anche i contesti sono strumenti di meta-livello, perché trascendono il linguaggio oggetto, per permetterci di ragionare su esso. Per di più, un contesto può essere saturato con una qualunque formula di meta-livello: considerare arbitrari contesti ci permette di ragionare su classi di formule molto estese, ossia su formule *immerse* in contesti arbitrari, cioè ad *arbitrari livelli di profondità*, concetto cardine di tutta la deep inference. Usando i contesti e il meta-linguaggio, possiamo ragionare per induzione strutturale sulla stringhe del linguaggio *a qualsiasi livello di profondità*. Inoltre, anche i contesti si possono *unificare* con una procedura effettiva.

Al meta-livello ragioniamo su sequenti definiti sul meta-linguaggio. Le definizioni e le tecniche viste in precedenza per singole formule, si estendono in maniera naturale alle liste di formule e ai sequenti: in particolare, è possibile l'*unificazione di sequenti* e considerare sequenti composti da formule *immerse in contesti arbitrari* (con procedure effettive).

1.3 Sistemi formali e formalismi

Le dimostrazioni sono l'oggetto di studio della proof theory; in questa sezione esplicitiamo la nozione di dimostrazione. Il termine "dimostrare" deriva dal latino *demonstrare*, composto dalla radice *de-* (di valore intensivo) e da *monstrare* ("mostrare", "far vedere"), da cui il significato di *rendere manifesto con fatti e con segni certi*. In matematica una dimostrazione è un *processo di deduzione* che, partendo da *premesse* assunte come valide (ipotesi) o da proposizioni dimostrate in virtù di tali premesse, determina la necessaria validità di una nuova *proposizione* in funzione della (sola) *coerenza formale* del ragionamento. Le proposizioni saranno dunque stringhe appartenenti ad

un linguaggio formale; il processo di deduzione sarà scandito dalla corretta applicazione di alcune regole di base in qualche modo riconosciute come elementari e la coerenza formale dovrà essere opportunamente formalizzata e fungerà da argomento a sostegno della bontà delle regole scelte.

Definizione 1.3.1 (Regole, derivazioni, dimostrazioni)**.** *Un* sistema formale *è una coppia* $(\mathscr{L}, \mathscr{S})$ *composta da un* linguaggio $\mathscr{L}$ *(generato da qualche grammatica* G*, tipicamente – ma non necessariamente – context-free) e da un insieme di regole d'inferenza (o* sistema di deduzione*)* $\mathscr{S}$*. Date le formule* $P_1, \ldots, P_n, Q \in \mathscr{L}'$*, una* regola di inferenza (ρ) *è un'espressione della forma:*

$$\frac{P_1 \qquad \cdots \qquad P_n}{Q}\,(\rho)$$

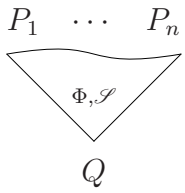
dove $P_1, \ldots, P_n$ *sono chiamate* premesse della regola (ρ) *mentre* Q *ne è la* conclusione*. Una regola di inferenza senza premesse (i.e. avente* $n = 0$*) è chiamata* assioma*, mentre, per* $n > 0$*, è detta* regola d'inferenza propria*. In genere le premesse e la conclusione di* (ρ) *sono formule (o sequenti) aventi una certa forma di superficie – ed eventualmente, nell'approccio deep inference, immerse in arbitrari contesti. A questo modo di procedere, cioè di specificare la* forma *delle (eventuali) ipotesi e della conclusione delle regole d'inferenza, ci si riferisce spesso in letteratura col termine* schema *(p.e. dicendo "schema d'assioma").*

Un passo d'inferenza *o* applicazione *o* istanza *di una regola d'inferenza* (ρ) *è un'espressione della forma:*

$$\frac{P'_1 \qquad \cdots \qquad P'_n}{Q'}\,(\rho)$$

dove $P'_1, \ldots, P'_n, Q' \in \mathscr{L}'$ *sono formule ottenute rispettivamente per unificazione (anche parziale, al meta-livello) con* $P_1, \ldots, P_n, Q \in \mathscr{L}'$*. Le stringhe* $P'_1, \ldots, P'_n$ *sono chiamate* premesse dell'applicazione di (ρ) *mentre* Q' *ne è la* conclusione*. Indicheremo anche il nome della regola accanto alla barra orizzontale di derivazione, quando questo sarà d'aiuto alla comprensione e non sarà fonte d'ambiguità.*

Una derivazione Φ *da una lista di premesse* $P_1, \ldots, P_n$ *ad una conclusione* Q *è un albero di istanze di regole in* $\mathscr{S}$*, avente* Q *come radice e* $P_1, \ldots, P_n$ *come foglie, e indicato con:*



Nel seguito ometteremo Φ *e/o* $\mathscr{S}$ *quando questo non comporterà ambiguità.*

Infine, una dimostrazione *è una derivazione avente per come premesse* $P_1, \ldots, P_n$ *solo istanze di assiomi. La indicheremo con:*

$$\begin{array}{c} \Phi \Vert\!\Vert \mathscr{S} \\ Q \end{array}$$

omettendo Φ *e/o* $\mathscr{S}$ *quando e se necessario.*

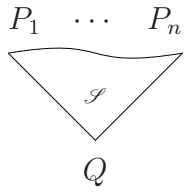
La *derivabilità* in un sistema formale ("da un insieme di formule Γ è possibile derivare la formula P", o anche "P è derivabile da Γ") è un concetto sintattico, così come lo è la *dimostrabilità* – in contrapposizione al concetto di *verità* e a quello di *modello*, che sono invece di natura semantica. Finora non abbiamo mai parlato di verità: in questa sede non ci occuperemo degli aspetti semantici legati ai sistemi formali, che non sono oggetto di studio di proof theory, rimandando per questi ad Abramsky et al. [1992]; Barwise [1977]; Chang et al. [1973].

Il concetto di derivabilità si estende anche alle regole dei sistemi formali: un regola è derivabile quando è ottenibile tramite altre regole. Ma una regola può anche essere *ammissibile* (o *eliminabile*): questo accade quando la sua presenza all'interno del sistema non altera l'insieme di formule dimostrabili, ossia eliminando la regola dal sistema, si riescono a dimostrare *le stesse cose*. Questo vale anche per le regole derivabili, ma mentre in quel caso era sufficiente sostituire la regola con la sua derivazione, per regole ammissibili occorre ristrutturare l'albero di prova.

Definizione 1.3.2 (Regole derivabili e ammissibili)**.** *Una regola* (ρ) *è* derivabile *per un sistema* $\mathscr{S}$ *se, per ogni istanza di* (ρ)*:*

$$\frac{P_1 \quad \cdots \quad P_n}{Q}\,(\rho)$$

esiste una derivazione:



Una regola (ρ) *è* ammissibile *(o* eliminabile*) per un sistema* $\mathscr{S}$ *se, per ogni dimostrazione* $\begin{matrix}\Vdots\\ Q\end{matrix}^{\mathscr{S}\cup\{(\rho)\}}$ *esiste una dimostrazione* $\begin{matrix}\Vdots\\ Q\end{matrix}^{\mathscr{S}}$*.*

In teoria della dimostrazione ci concentriamo (al meta-livello) sulle proprietà dei sistemi formali, cioè le proprietà di cui godono le dimostrazioni espresse in qualche sistema formale. Ma *quale* sistema formale? Seguendo Troelstra and Schwichtenberg [1996], possiamo raggrupparli in alcune grandi famiglie chiamate *formalismi*:

- *Sistemi assiomatici* o *sistemi alla Frege-Hilbert* (Hilbert and Ackermann [1928]; Frege [1879]): in questo approccio accettiamo un numero molto ristretto di regole d'inferenza proprie (p.e. nella logica proposizionale solo una, il *modus ponens*) mentre il resto del sistema deduttivo sarà composto da assiomi; le derivazioni in questo formalismo furono originariamente concepite per rispecchiare le dimostrazioni espresse in linguaggio naturale, obiettivo ambizioso e scarsamente raggiunto da questo approccio in cui le dimostrazioni tendono invece ad essere molto dettagliate e "pedanti". È l'approccio più datato e grazie ad esso è stato possibile formalizzare con successo sistemi deduttivi rilevanti quali: la logica classica, quella intuizionista ed alcune logiche modali;

- *Deduzione naturale*: introdotta nel celebre Gentzen [1935] (assieme, come vedremo, al *calcolo dei sequenti*) questa famiglia di sistemi è pensata

per mimare il ragionamento logico-deduttivo umano (da qui l'aggettivo "naturale"). Non ci sono assiomi e le formule possono essere composte e decomposte (usando il gergo tecnico, rispettivamente *introdotte* ed *eliminate*).

Un'operazione comune nella pratica matematica è quella di *ragionare per assunzioni*: la deduzione naturale mette a disposizione un artificio per compiere questa operazione, e i sistemi espressi in deduzione naturale godono di buone proprietà, relativamente semplici da dimostrare (vedi il classico Prawitz [1965]);

- *Calcolo dei sequenti*: dovuto a Gentzen, questo è lo strumento preferito in teoria della dimostrazione per le ottime proprietà di cui gode. Fa tipicamente uso di pochi assiomi (p.e. nella logica proposizionale solo uno, l'assioma *identità*) e molte regole d'inferenza proprie, che permettono di comporre nuove formule a partire dalle premesse (usando la terminologia della deduzione naturale, sono presenti le regole di *introduzione* ma non quelle di *eliminazione*).

Volendo aderire ad una visione "proof theoretical", ci concentreremo in seguito sul calcolo dei sequenti. È tuttavia doveroso osservare che questi formalismi consentono di definire sistemi formali aventi il medesimo potere espressivo: in altre parole, nessuno prevale a priori sugli altri, dipende dal *setting* in cui ci poniamo.

Inoltre questi sono solo i formalismi "classici"; esistono altre famiglie di sistemi formali, che possono essere usate per mettere in evidenza altri aspetti importanti del processo deduttivo e delle dimostrazioni. Tra questi è doveroso citare le *Proof Nets*, introdotte in Girard [1987] allo scopo di far emergere alcune simmetrie dei sistemi formali che erano "oscurate" dal calcolo dei sequenti.

Le tre famiglie sopra descritte adottano tutte shallow inference come specifica delle regole d'inferenza (per quanto questa sia una scelta del tutto

arbitraria). Ma l'approccio deep inference apre la via ad (almeno) un quarto formalismo:

- *Calcolo delle strutture*: introdotto in Guglielmi [2002], i sistemi deduttivi in calcolo delle strutture constano di un piccolo numero di assiomi e di regole d'inferenza proprie, e godono di una notevole quantità di proprietà, sostanzialmente estendendo quelle studiate per il calcolo dei sequenti. Le nuove prospettive aperte nell'ambito del calcolo delle strutture, ne fanno uno strumento di grande interesse e in continuo sviluppo da parte della comunità scientifica.

1.4 Metodologie: shallow *versus* deep inference

Le metodologie guidano la progettazione dei sistemi deduttivi e il processo di inferenza: le due metodologie conosciute allo stato dell'arte sono *shallow* e *deep inference* e le esamineremo a turno.

Definizione 1.4.1 (Shallow inference)**.** *Per* shallow inference *o* inferenza di superficie*, intendiamo la* metodologia *d'inferenza che interpreta l'insieme delle regole d'inferenza come* schemi *che disciplinano il comportamento della deduzione* in funzione del connettivo principale *delle formule.*

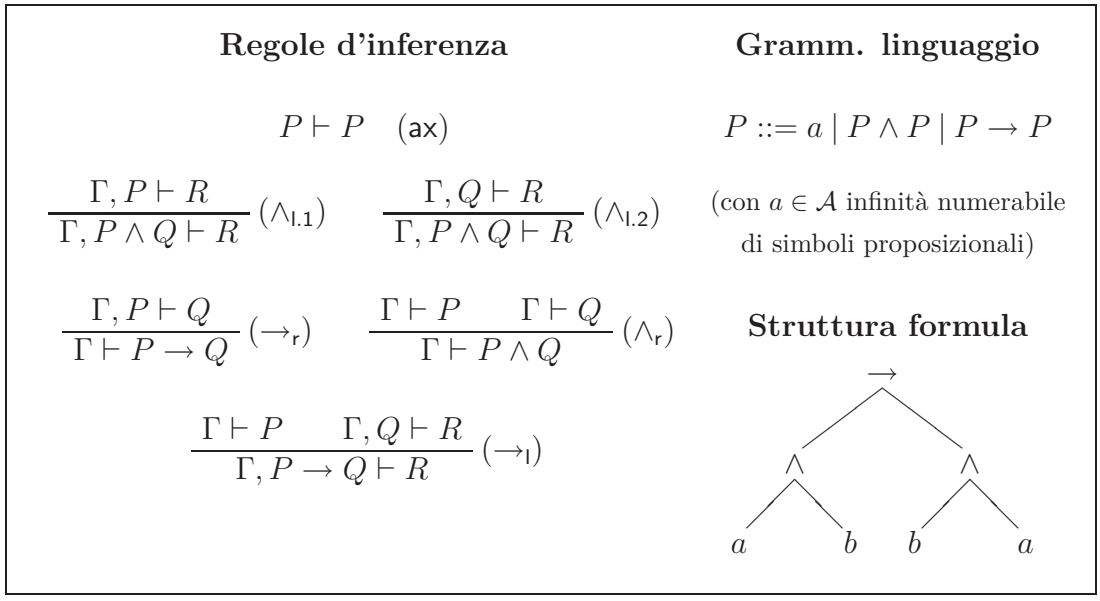


Figura 1.2: Sistema formale in shallow inference ed esempio di formula

Essendo l'approccio più datato, è anche il più usato in letteratura, come in *deduzione naturale* (i sistemi NK ed NJ usano shallow inference) e nel *calcolo dei sequenti* (sistemi LK, LJ). Ad esempio: per derivare $\vdash (a \wedge b) \to (b \wedge a)$ con le regole d'inferenza in Figura 1.2, consideriamo la struttura della formula $(a \wedge b) \to (b \wedge a)$ ed osserviamo che il connettivo principale è $\to$. A questo punto l'unica regola applicabile (i.e. istanziabile, ricordiamo che le

regole d'inferenza sono *schemi*) in shallow inference è $(\rightarrow_{\mathsf{r}})$. In questo modo otteniamo:

$$\frac{a \wedge b \vdash b \wedge a}{\vdash (a \wedge b) \rightarrow (b \wedge a)}$$

Procedendo in maniera analoga osserviamo che ci sono tre regole applicabili per derivare $a \wedge b \vdash b \wedge a$, cioè:

- $(\wedge_{\mathsf{l}.1})$ produce la derivazione:

$$\frac{\dfrac{a \vdash b \wedge a}{a \wedge b \vdash b \wedge a}}{\vdash (a \wedge b) \rightarrow (b \wedge a)}$$

  da cui è applicabile solo $(\wedge_{\mathsf{r}})$ che produce una derivazione bloccata (cioè un albero le cui foglie non sono istanze di assiomi, né sono derivabili dalle regole del sistema);

- $(\wedge_{\mathsf{l}.2})$ analogo al precedente;

- $(\wedge_{\mathsf{r}})$ produce la derivazione:

$$\frac{\dfrac{\begin{matrix}(1)\\ a \wedge b \vdash b\end{matrix} \qquad \begin{matrix}(2)\\ a \wedge b \vdash a\end{matrix}}{a \wedge b \vdash b \wedge a}}{\vdash (a \wedge b) \rightarrow (b \wedge a)}$$

  in cui è possibile applicare $(\wedge_{\mathsf{l}.1})$ o $(\wedge_{\mathsf{l}.2})$ sia alla formula (1) che alla (2). L'unica combinazione che porta ad una conclusione – cioè in cui ogni foglia è un'istanza di (ax) – è un'applicazione di $(\wedge_{\mathsf{l}.2})$ a (1) e di $(\wedge_{\mathsf{l}.1})$ a (2), ottenendo così:

$$\frac{\dfrac{\dfrac{b \vdash b}{a \wedge b \vdash b} \qquad \dfrac{a \vdash a}{a \wedge b \vdash a}}{a \wedge b \vdash b \wedge a}}{\vdash (a \wedge b) \rightarrow (b \wedge a)}$$

La procedura descritta nell'esempio è nota come *proof search* ed è automatizzabile (p.e. si può basare sulla risoluzione come avviene in PROLOG) per sistemi *in cui la regola di taglio è ammissibile*.

Definizione 1.4.2 (Deep inference)**.** *Per* deep inference *o* inferenza di profondità *intendiamo la metodologia in cui le regole d'inferenza si possono applicare ad arbitrari contesti, e quindi ad arbitrari livelli di profondità, in contrapposizione a quanto avviene nell'inferenza di superficie o* shallow inference*. Il ruolo dei contesti è quello di permettere l'accesso alla struttura delle formule senza dover usare alberi di derivazione (cioè senza decomposizione strutturale delle formule). Per questa ragione le regole d'inferenza in deep inference hanno al più una premessa: pertanto le derivazioni prendono la forma di liste. Per enfatizzare il fatto che le derivazioni sono* alberi degeneri *(i.e. ogni nodo ha al più un figlio), usiamo la notazione:*

$$\begin{array}{c} P \\ \Phi \Big\| \mathscr{S} \\ Q \end{array}$$

per indicare una derivazione Φ *che usa le regole in* $\mathscr{S}$*, e avente premessa* P *e conclusione* Q*.*

Dimostriamo l'analogo della formula di prima, usando il sistema deep inference in Figura 1.3. Invece dell'implicazione, qui abbiamo solo la negazione sugli atomi, quindi la formula di prima diventa: $(\overline{a} \vee \overline{b}) \vee (b \wedge a)$.

In questo caso abbiamo raggruppato le regole d'inferenza in "regole logiche" (simili a quelle viste prima) e "regole strutturali". Quest'ultime servono a formalizzare il fatto che i connettivi di congiunzione e di disgiunzione godono della proprietà commutativa – rispettivamente regole $(\wedge_{\mathsf{com}})$ e $(\vee_{\mathsf{com}})$ – e di quella associativa – regole $(\wedge_{\mathsf{as}})$ e $(\vee_{\mathsf{as}})$ – e che inoltre l'atomo t (risp. f) è elemento neutro per il connettivo di congiunzione (risp. disgiunzione). A parte per la proprietà commutativa, che è intrinsecamente simmetrica, per le altre bisognerebbe specificare anche le regole opposte; ad esempio, per $(\wedge_{\mathsf{as}})$ servirebbe una regola:

$$\frac{\mathbb{C}\{P \wedge (Q \wedge R)\}}{\mathbb{C}\{(P \wedge Q) \wedge R\}} \, (\wedge_{\mathsf{as}}^{-1})$$

Per questa ragione, in deep inference, si è soliti *sostituire le regole strutturali con una relazione d'equivalenza* tra formule.

Regole logiche

$$\mathsf{t} \quad (\mathsf{ax})$$

$$\frac{\mathbb{C}\{\mathsf{t}\}}{\mathbb{C}\{a \vee \overline{a}\}}\,(\mathsf{id}) \qquad \frac{\mathbb{C}\{P \wedge (Q \vee R)\}}{\mathbb{C}\{(P \wedge Q) \vee R\}}\,(\mathsf{s})$$

Grammatica linguaggio

$$P ::= \mathsf{t} \,|\, \mathsf{f} \,|\, a \,|\, \overline{a} \,|\, P \vee P \,|\, P \wedge P$$

(con $a \in \mathcal{A}$ infinità numerabile di simboli proposizionali)

Regole strutturali

$$\frac{\mathbb{C}\{P\}}{\mathbb{C}\{P \wedge \mathsf{t}\}}\,(\wedge_{\mathsf{t}}) \qquad \frac{\mathbb{C}\{P\}}{\mathbb{C}\{P \vee \mathsf{f}\}}\,(\vee_{\mathsf{f}})$$

$$\frac{\mathbb{C}\{P \wedge Q\}}{\mathbb{C}\{Q \wedge P\}}\,(\wedge_{\mathsf{com}}) \qquad \frac{\mathbb{C}\{(P \wedge Q) \wedge R\}}{\mathbb{C}\{P \wedge (Q \wedge R)\}}\,(\wedge_{\mathsf{as}})$$

$$\frac{\mathbb{C}\{P \vee Q\}}{\mathbb{C}\{Q \vee P\}}\,(\vee_{\mathsf{com}}) \qquad \frac{\mathbb{C}\{(P \vee Q) \vee R\}}{\mathbb{C}\{P \vee (Q \vee R)\}}\,(\vee_{\mathsf{as}})$$

Dimostrazione d'esempio

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathsf{t}}{\mathsf{t} \wedge \mathsf{t}}\,(\wedge_{\mathsf{t}})}{(a \vee \overline{a}) \wedge \mathsf{t}}\,(\mathsf{id})}{(a \vee \overline{a}) \wedge (b \vee \overline{b})}\,(\mathsf{id})}{((a \vee \overline{a}) \wedge b) \vee \overline{b}}\,(\mathsf{s})}{(b \wedge (a \vee \overline{a})) \vee \overline{b}}\,(\wedge_{\mathsf{com}})}{((b \wedge a) \vee \overline{a}) \vee \overline{b}}\,(\mathsf{s})}{(b \wedge a) \vee (\overline{a} \vee \overline{b})}\,(\vee_{\mathsf{as}})}{(\overline{a} \vee \overline{b}) \vee (b \wedge a)}\,(\vee_{\mathsf{com}})$$

Figura 1.3: Sistema formale in deep inference ed esempio di dimostrazione

Come possiamo vedere, la dimostrazione della medesima formula di prima è completamente mutata: innanzitutto osserviamo che le regole induttive, nel calcolo delle strutture, hanno sempre e solo una premessa ed una conclusione. Questo fa sì che le dimostrazioni si sviluppino solo in altezza, collassando il lavoro strutturale svolto dagli alberi, all'interno dei contesti. Inoltre il sistema formale è meno rigido di quello visto nell'esempio precedente, nel senso che la metodologia deep inference permette un'applicazione più capillare delle regole d'inferenza, e quindi in generale un maggior grado di libertà e di non-determinismo.

Capitolo 2

Logica classica proposizionale

In questo capitolo presenteremo una serie di definizioni e risultati tradizionali in proof theory di logica classica proposizionale, e studieremo le proprietà formali del suo corrispettivo in deep inference, chiamato Sistema SKS.

Il Sistema che andremo a studiare (chiamato LKp) è il frammento proposizionale del Sistema LK di Gentzen [1935], il cui nome è l'acronimo di *Logik Klassische* ossia "logica classica" (mentre la "p" sta appunto per proposizionale). Il linguaggio $\mathscr{L}_{\mathsf{LKp}}$ è quello dei sequenti in Definizione 1.1.4, aventi per formule quelle generate dalla grammatica:

$$G_{\mathsf{LKp}} = (\{\neg, \wedge, \vee, \rightarrow, [a{-}z], S\}, \{S\}, S, \mathcal{P})$$

dove $[a{-}z]$ è una notazione abbreviata per i caratteri dell'alfabeto inglese, e le produzioni in $\mathcal{P}$, sono:

$$S ::= [a{-}z]+ \mid \neg S \mid S \wedge S \mid S \vee S \mid S \rightarrow S$$

dove $[a{-}z]+$ appartiene alla *notazione EBNF* (*BNF estesa*) e significa semplicemente "qualunque stringa non vuota di caratteri alfabetici". Solitamente s'introduce un'ulteriore generalizzazione, considerando la produzione (S, a) dove a è una meta-variabile appartenente ad un insieme $\mathcal{A}$ composto da un'infinità numerabile di stringhe (alfabetiche, alfa-numeriche, indicizzate

con apici, pedici, ...), chiamate genericamente "simboli proposizionali". Osserviamo che le stringhe di $\mathcal{A}$ sarebbero facilmente ottenibili da una grammatica context-free, questa semplificazione serve solo ad alleggerire la notazione, mentre preserva intatto il carattere finitario del linguaggio oggetto.

Il sistema deduttivo $\mathscr{S}_{\mathsf{LKp}}$ è dato usando forma di superficie dei sequenti. Le regole d'inferenza si possono dividere in quattro gruppi: assiomi, taglio, regole strutturali e regole logiche.

Le regole strutturali sono di fondamentale importanza, perché permettono di manipolare l'ordine ed il numero delle formule del sequente. Sono tre:

1. L'*ordine* delle premesse (e delle conclusioni) *non è rilevante*. Da qui otteniamo le regole di *permutazione*:

$$\frac{\Gamma, P, Q, \Gamma' \vdash \Delta}{\Gamma, Q, P, \Gamma' \vdash \Delta}\,(\mathsf{perm_l}) \qquad \frac{\Gamma \vdash \Delta, P, Q, \Delta'}{\Gamma \vdash \Delta, Q, P, \Delta'}\,(\mathsf{perm_r})$$

2. Assumere due volte la stessa premessa (o la stessa conclusione) è equivalente ad assumerla una volta sola. Questa osservazione ci conduce alle regole di *contrazione*:

$$\frac{\Gamma, P, P \vdash \Delta}{\Gamma, P \vdash \Delta}\,(\mathsf{cont_l}) \qquad \frac{\Gamma \vdash Q, Q, \Delta}{\Gamma \vdash Q, \Delta}\,(\mathsf{cont_r})$$

3. È sempre possibile sia aggiungere nuove ipotesi (rafforzare l'antecedente), sia aggiungere nuove conclusioni (indebolire il conseguente). In generale il sequente ne risulterà indebolito (in un caso serve un'ipotesi in più affinché funzioni, nell'altro a parità di ipotesi dimostra una cosa più vaga, con più possibili conseguenze). Questa è pertanto chiamata regola di *indebolimento*:

$$\frac{\Gamma \vdash \Delta}{\Gamma, P \vdash \Delta}\,(\mathsf{w_l}) \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash Q, \Delta}\,(\mathsf{w_r})$$

Per individuare gli assiomi, ci poniamo la seguente domanda: quando si può sostenere che un sequente $\Gamma \vdash \Delta$ è *evidentemente* vero? Chiaramente quando $\Gamma \cap \Delta \neq \varnothing$, cioè quando almeno una delle premesse in Γ compare tra

le conclusioni in Δ. Questo sarà l'unico assioma del nostro Sistema, non ci sono altri criteri evidenti per passare dalle premesse alle conclusioni senza fare inferenza. In virtù delle regole strutturali, sappiamo che l'ordine non conta: pertanto dimostrare che esiste un $P \in \Gamma$ tale che $P \in \Delta$, si può scrivere: $P, \Gamma \vdash P, \Delta$. Inoltre possiamo sempre applicare la regola d'indebolimento a sinistra e a destra, per ottenere alfine:

$$P \vdash P \quad (\mathsf{ax})$$

Quella di taglio è un'altra regola fondamentale, che ci si aspetta che sia soddisfatta da ogni sistema deduttivo. Il suo scopo è garantire la *componibilità* delle dimostrazioni; questa proprietà è ampiamente sfruttata nella pratica matematica: per provare un teorema complesso, si può cominciare dimostrando dei lemmi più semplici, che possono essere poi composti per ottenere il risultato cercato. La formulazione della *regola di taglio* è pertanto la seguente:

$$\frac{\Gamma \vdash P, \Delta \qquad \Gamma', P \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\,(\mathsf{cut})$$

Infine le regole logiche sono quelle che specificano il comportamento dei connettivi logici. Come abbiamo già avuto modo di menzionare, nel calcolo dei sequenti è solo possibile *introdurre* nuovi connettivi ma mai di *eliminarli*. Questo fatto è alla base di una proprietà molto importante, detta *della sottoformula* (vedi Definizione 2.0.3). Le regole d'introduzione dei connettivi saranno classificate in *destre* (indicate con una "r" a pedice) o *sinistre* (indicate con "l") a seconda che permettano d'introdurre il connettivo a destra oppure a sinistra del tornello. Vediamole rapidamente, ci sono quattro connettivi nel nostro linguaggio:

1. **Congiunzione**: introdurre una congiunzione a sinistra significa rafforzare la premessa aggiungendo un'ipotesi. Come abbiamo detto in precedenza, il significato intuitivo del sequente va specificato formalmente, e queste regole chiarificano come le formule a sinistra del turnstile siano da considerarsi in congiunzione tra loro:

$$\frac{\Gamma, P \vdash \Delta}{\Gamma, P \wedge Q \vdash \Delta}\,(\wedge_{\mathsf{l}.1}) \qquad \frac{\Gamma, Q \vdash \Delta}{\Gamma, P \wedge Q \vdash \Delta}\,(\wedge_{\mathsf{l}.2})$$

A destra invece, cioè per concludere che una congiunzione $P \wedge Q$ vale, sotto un certo insieme di ipotesi, dobbiamo aver dimostrato separatamente i due rami P e Q a partire dalle stesse assunzioni, cioè:

$$\frac{\Gamma \vdash P, \Delta \qquad \Gamma \vdash Q, \Delta}{\Gamma \vdash P \wedge Q, \Delta}\,(\wedge_{\mathsf{r}})$$

2. **Disgiunzione**: il ragionamento e le regole seguono in maniera perfettamente simmetrica quanto visto per la congiunzione. Non c'è da sorprendersi, poiché la disgiunzione è il connettivo duale alla congiunzione, e poiché il sequente è fatto in modo da rispettare naturalmente tale simmetria. A destra del tornello, le formule sono da considerarsi in disgiunzione tra loro, e quindi abbiamo:

$$\frac{\Gamma \vdash P, \Delta}{\Gamma \vdash P \vee Q, \Delta}\,(\vee_{\mathsf{r}.1}) \qquad \frac{\Gamma \vdash Q, \Delta}{\Gamma \vdash P \vee Q, \Delta}\,(\vee_{\mathsf{r}.2})$$

mentre a sinistra, se da un set comune di ipotesi Γ unito ad un'ipotesi P riusciamo a concludere che valgono certe conclusioni Δ, e indipendentemente, dallo stesso set di premesse Γ unito stavolta ad una formula Q, siamo in grado di concludere le medesime conclusioni Δ, allora da Γ e $P \vee Q$ possiamo concludere che vale Δ, cioè:

$$\frac{\Gamma, P \vdash \Delta \qquad \Gamma, Q \vdash \Delta}{\Gamma, P \vee Q \vdash \Delta}\,(\vee_{\mathsf{l}})$$

3. **Implicazione**: se la virgola a sinistra e a destra del tornello si comportano rispettivamente come una congiunzione e come una disgiunzione, il tornello stesso è l'implicazione. Questo fatto è reso evidente dalla regola d'introduzione destra della freccia. Infatti, la regola è:

$$\frac{\Gamma, P \vdash Q, \Delta}{\Gamma \vdash P \to Q, \Delta}\,(\to_{\mathsf{r}})$$

cioè afferma che se otteniamo una certa conclusione Q supponendo Γ e P, con solo Γ è possibile concludere che "se vale P allora Q", cioè proprio $P \to Q$. Le altre conclusioni in Δ non giocano alcun ruolo intuitivo per questa regola, se non di preservare una certa omogeneità nella forma del sequente. Abbiamo visto come una formula P sia in grado di passare da sinistra a destra del tornello, tramutandosi in un'implicazione. Anche il passaggio inverso è possibile:

$$\frac{\Gamma \vdash P, \Delta \qquad \Gamma, Q \vdash \Delta}{\Gamma, P \to Q \vdash \Delta}\,(\to_{\mathsf{l}})$$

Qui si è dimostrato che da Γ si deriva P e, indipendentemente, che da Γ unito all'ipotesi aggiuntiva Q si conclude Δ. Allora da Γ e supponendo che P implichi Q è possibile concludere Δ.

4. **Negazione**: in virtù di quanto visto finora, il comportamento della negazione dovrebbe risultare piuttosto semplice. Infatti, se consideriamo una singola formula, passare da una parte all'altra del tornello significa introdurre una negazione (la negazione di una formula è equivalente ad un'implicazione in cui dalla validità della formula si conclude il falso). Formalmente:

$$\frac{\Gamma \vdash P, \Delta}{\Gamma, \neg P \vdash \Delta}\,(\neg_{\mathsf{l}}) \qquad \frac{\Gamma, P \vdash \Delta}{\Gamma \vdash \neg P, \Delta}\,(\neg_{\mathsf{r}})$$

Facciamo un esempio di *regola derivabile* nel Sistema LKp: scriviamo la regola *destra di congiunzione* e la regola di *destra di congiunzione generalizzata*:

$$\frac{\Gamma \vdash P, \Delta \qquad \Gamma \vdash Q, \Delta}{\Gamma \vdash P \wedge Q, \Delta}\,(\wedge_{\mathsf{r}}) \qquad \frac{\Gamma \vdash P, \Delta \qquad \Gamma' \vdash Q, \Delta'}{\Gamma, \Gamma' \vdash P \wedge Q, \Delta, \Delta'}\,(\wedge_{\mathsf{r}}^{\mathsf{gen}})$$

- $(\wedge_{\mathsf{r}})$ è banalmente derivabile da $(\wedge_{\mathsf{r}}^{\mathsf{gen}})$, infatti basta porre $\Gamma' = \Gamma$ e $\Delta' = \Delta$ per ottenere:

$$\dfrac{\dfrac{\Gamma \vdash P, \Delta \qquad \Gamma \vdash Q, \Delta}{\Gamma, \Gamma \vdash P \wedge Q, \Delta, \Delta}\,(\wedge_{\mathsf{r}}^{\mathsf{gen}})}{\Gamma \vdash P \wedge Q, \Delta}$$

dove la doppia barra orizzontale indica un certo numero applicazioni di regole strutturali, in questo caso *permutazione* e *contrazione*. D'ora in avanti useremo sempre questa convenzione.

- $(\wedge_{\mathsf{r}}^{\mathsf{gen}})$ è derivabile da $(\wedge_{\mathsf{r}})$:

$$\dfrac{\dfrac{\Gamma \vdash P, \Delta}{\Gamma, \Gamma' \vdash P, \Delta, \Delta'} \quad \dfrac{\Gamma' \vdash Q, \Delta'}{\Gamma, \Gamma' \vdash Q, \Delta, \Delta'}}{\Gamma, \Gamma' \vdash P \wedge Q, \Delta, \Delta'}\,(\wedge_{\mathsf{r}})$$

Ragionamenti analoghi valgono per la regole sinistre di disgiunzione e implicazione (generalizzate). Per quanto riguarda le regole *ammissibili*, avremo modo nel seguito di dimostrare *l'ammissibilità della regola di taglio* in LKp.

Definizione 2.0.3 (Proprietà della sottoformula)**.** *Si dice che* una regola d'inferenza (ρ) gode della proprietà della sottoformula *sse per ogni sua istanza:*

$$\dfrac{P_1 \qquad \cdots \qquad P_n}{Q}$$

si ha che $P_1, \ldots, P_n$ sono sottoformule di Q. Questa definizione si estende naturalmente alle regole del calcolo dei sequenti, imponendo che tutte le formule nelle premesse (sia a destra che a sinistra del turnstile) siano sottoformule di quelle presente nel sequente conclusione.

Inoltre si dice che un sistema formale gode della proprietà della sottoformula *quando tutte le sue regole d'inferenza ne godono.*

La proprietà della sottoformula è molto rilevante, perché conferisce ai sistemi una natura "costruttiva", il che ha molte importanti ripercussioni sulla meccanizzazione del processo inferenziale e sulla proof search. Un risultato classico è il seguente:

Teorema 2.0.4 (Consistenza)**.** *Sia dato un sistema formale, espresso mediante il calcolo dei sequenti, non triviale (che non contiene il sequente vuoto tra gli assiomi). Allora, se gode della proprietà della sottoformula, esso è consistente (cioè non permette di derivare il sequente vuoto).*

Dimostrazione. La dimostrazione è immediata, poiché, se il sequente vuoto non è fra gli assiomi del sistema, dev'essere derivato con una regola d'inferenza propria (ρ). Ma per la proprietà della sottoformula, la regola d'inferenza (ρ) può avere per premesse solo sottoformule di quelle nel sequente vuoto, cioè non può avere premesse, ma (ρ) è propria per ipotesi. Pertanto il sequente vuoto non è derivabile ed il sistema è consistente. □

Da una rapida ispezione alle regole del Sistema LKp, ci accorgiamo che la proprietà della sottoformula vale per tutte le regole d'inferenza tranne che per la regola di taglio. Infatti (cut) introduce un'arbitraria formula P tra le premesse. Onde preservare la proprietà della sottoformula, seguiamo i passi di Gentzen, dimostrando uno dei teoremi centrali della proof theory, noto come "Gentzen Hauptsatz", che ci garantisce che la regola di taglio è ammissibile all'interno del Sistema.

2.1 Eliminazione del taglio

Ci accingiamo a dimostrare una proprietà essenziale per la logica classica (e non solo), chiamata Hauptsatz, o teorema di eliminazione del taglio. L'Hauptsatz presumibilmente traccia il confine tra la logica e la nozione più ampia di sistema formale. Per sottolinearne l'importanza, Girard usa il motto:

> *"A sequent calculus without cut elimination is like a car without engine"* – Girard [1995b]

Definizione 2.1.1 (Grado, altezza derivazioni)**.** *Il* grado di una formula $\delta(P)$ *è definito per induzione strutturale come segue:*

- $\delta(a) = 1$ *per a simbolo proposizionale*
- $\delta(P \wedge Q) = \delta(P \vee Q) = \delta(P \to Q) = 1 + \max\{\delta(P), \delta(Q)\}$
- $\delta(\neg P) = 1 + \delta(P)$

Il grado di un'applicazione della regola di taglio *è definito come il grado della formula che elimina.*

Il grado $\delta(\Phi)$ di una derivazione *è il massimo tra i gradi delle regole di taglio che vi compaiono. In particolare* $\delta(\Phi) = 0$ *se* Φ *non fa uso della regola di taglio.*

*Infine, l'*altezza $h(\Phi)$ di una derivazione *è quella associata all'albero* Φ*: se la regola conclusiva di* Φ *ha come premesse le derivazioni* $\Phi_1, \ldots, \Phi_n$*, allora* $h(\Phi) = 1 + \max\{h(\Phi_1), \ldots, h(\Phi_n)\}$ *(mentre se* $n = 0$*, cioè se* Φ *è istanza di un assioma, allora* $h(\Phi) = 0$*).*

Lemma 2.1.2. *Sia* Φ *una derivazione della forma seguente:*

$$\dfrac{\dfrac{P_1 \quad \cdots \quad P_n}{\Gamma \vdash P, \Delta}\,(\rho_{\mathsf{l}}) \qquad \dfrac{P_{n+1} \quad \cdots \quad P_{n+m}}{\Gamma', P \vdash \Delta'}\,(\rho_{\mathsf{r}})}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\,(\mathsf{cut})$$

in cui (ρ_{l}) *(la premessa di sinistra del cut) è una regola logica "destra", mentre* (ρ_{r}) *(premessa destra del cut) è una regola logica "sinistra", tali da introdurre entrambe la formula* P*. Allora esiste una derivazione:*

$$\begin{array}{c} P'_1 \quad \cdots \quad P'_k \\ \hline \\ \Psi \\ \\ \hline \Gamma, \Gamma' \vdash \Delta, \Delta' \end{array}$$

con $\{P'_1, \ldots, P'_k\} \subseteq \{P_1, \ldots, P_{n+m}\}$ *e tale che* $\delta(\Psi) < \delta(\Phi)$*.*

Dimostrazione. Procediamo per casi sulla premessa sinistra dell'applicazione di (cut). Il fatto di concentrarci sulla premessa sinistra è del tutto irrilevante, poiché grazie alla simmetria delle regole logiche del Sistema LKp, se nella premessa sinistra s'introduce un certo connettivo (con una regola logica "destra"), questo dovrà essere introdotto anche nella premessa di destra (con una regola logica simmetrica "sinistra").

1. $(\wedge_{\mathsf{r}})$ e $(\wedge_{\mathsf{l}.1})$: qui abbiamo $P = Q \wedge R$.

$$\dfrac{\dfrac{\Gamma \vdash Q, \Delta \qquad \Gamma \vdash R, \Delta}{\Gamma \vdash Q \wedge R, \Delta}\,(\wedge_{\mathsf{r}}) \qquad \dfrac{\Gamma', Q \vdash \Delta'}{\Gamma', Q \wedge R \vdash \Delta'}\,(\wedge_{\mathsf{l}.1})}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\,(\mathsf{cut})$$

La derivazione Φ sopra si trasforma in Ψ come segue:

$$\dfrac{\Gamma \vdash Q, \Delta \qquad \Gamma', Q \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\,(\mathsf{cut})$$

Osserviamo che $\delta(\Psi) = \delta(\Phi) - \delta(R)$, cioè il grado di Ψ è diminuito di un fattore $\delta(R) > 0$.

2. $(\wedge_{\mathsf{r}})$ e $(\wedge_{\mathsf{l}.2})$: in maniera simmetrica, qui abbiamo:

$$\dfrac{\dfrac{\Gamma \vdash Q, \Delta \qquad \Gamma \vdash R, \Delta}{\Gamma \vdash Q \wedge R, \Delta}\,(\wedge_{\mathsf{r}}) \qquad \dfrac{\Gamma', R \vdash \Delta'}{\Gamma', Q \wedge R \vdash \Delta'}\,(\wedge_{\mathsf{l}.2})}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\,(\mathsf{cut})$$

che si trasforma nuovamente in Ψ di grado inferiore:

$$\dfrac{\Gamma \vdash R, \Delta \qquad \Gamma', R \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\,(\mathsf{cut})$$

3. $(\vee_{\mathsf{r}.1})$ e $(\vee_{\mathsf{l}})$: qui abbiamo $P = Q \vee R$. Questo è il duale del caso 1:

$$\dfrac{\dfrac{\Gamma \vdash Q, \Delta}{\Gamma \vdash Q \vee R, \Delta}\,(\vee_{\mathsf{r}.1}) \qquad \dfrac{\Gamma', Q \vdash \Delta' \qquad \Gamma', R \vdash \Delta'}{\Gamma', Q \vee R \vdash \Delta'}\,(\vee_{\mathsf{l}})}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\,(\mathsf{cut})$$

che si trasforma in Ψ come segue:

$$\dfrac{\Gamma \vdash Q, \Delta \qquad \Gamma', Q \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\,(\mathsf{cut})$$

col grado di Ψ diminuito di un fattore $\delta(R)$.

4. $(\vee_{\mathsf{r}.2})$ e $(\vee_{\mathsf{l}})$: $P = Q \vee R$, caso simmetrico al precendente e duale a 2, produciamo una derivazione Ψ avente grado pari a $\delta(\Phi) - \delta(Q)$, a partire da Φ:

$$\dfrac{\dfrac{\Gamma \vdash R, \Delta}{\Gamma \vdash Q \vee R, \Delta}\,(\vee_{\mathsf{r}.2}) \qquad \dfrac{\Gamma', Q \vdash \Delta' \qquad \Gamma', R \vdash \Delta'}{\Gamma', Q \vee R \vdash \Delta'}\,(\vee_{\mathsf{l}})}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\,(\mathsf{cut})$$

nel modo seguente:

$$\dfrac{\Gamma \vdash R, \Delta \qquad \Gamma', R \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\,(\mathsf{cut})$$

5. $(\neg_{\mathsf{r}})$ e $(\neg_{\mathsf{l}})$: qui abbiamo $P = \neg Q$. La derivazione Φ pertanto è:

$$\dfrac{\dfrac{\Gamma, Q \vdash \Delta}{\Gamma \vdash \neg Q, \Delta}\,(\neg_{\mathsf{r}}) \qquad \dfrac{\Gamma' \vdash Q, \Delta'}{\Gamma', \neg Q \vdash \Delta'}\,(\neg_{\mathsf{l}})}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\,(\mathsf{cut})$$

Costruiamo Ψ scambiando le premesse di Φ e applicando direttamente il taglio, per ottenere una derivazione di grado $\delta(\Phi) - 1$, come segue:

$$\dfrac{\dfrac{\Gamma' \vdash Q, \Delta' \qquad \Gamma, Q \vdash \Delta}{\Gamma', \Gamma \vdash \Delta', \Delta}\,(\mathsf{cut})}{\Gamma, \Gamma' \vdash \Delta, \Delta'}$$

6. $(\rightarrow_{\mathsf{r}})$ e $(\rightarrow_{\mathsf{l}})$: $P = Q \rightarrow R$. Allora Φ:

$$\dfrac{\dfrac{\Gamma, Q \vdash R, \Delta}{\Gamma \vdash Q \rightarrow R, \Delta}\,(\rightarrow_{\mathsf{r}}) \qquad \dfrac{\Gamma' \vdash Q, \Delta' \qquad \Gamma', R \vdash \Delta'}{\Gamma', Q \rightarrow R \vdash \Delta'}\,(\rightarrow_{\mathsf{l}})}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\,(\mathsf{cut})$$

si trasforma in Ψ come segue:

$$\dfrac{\dfrac{\dfrac{\Gamma' \vdash Q, \Delta' \qquad \Gamma, Q \vdash R, \Delta}{\Gamma', \Gamma \vdash \Delta', R, \Delta}\,(\mathsf{cut})}{\Gamma, \Gamma' \vdash R, \Delta, \Delta'} \qquad \dfrac{\Gamma', R \vdash \Delta'}{\Gamma, \Gamma', R \vdash \Delta, \Delta'}}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\,(\mathsf{cut})$$

osserviamo che in quest'ultimo caso il problema è stato risolto usando *due* tagli, entrambi di grado inferiore.

□

Definizione 2.1.3 (Rimozione)**.** *Sia P una formula e Γ una lista di formule: allora $\Gamma \setminus P$ denota Γ in cui* tutte le occorrenze *della formula P sono state* rimosse.

Il seguente lemma dice che una (eventuale) applicazione della regola di taglio finale può essere eliminata. La sua complessa formulazione tiene conto delle regole strutturali che possono interferire col taglio.

Lemma 2.1.4. *Sia P una formula di grado d, e siano Φ, Φ' rispettivamente le dimostrazioni di $\Gamma \vdash \Delta$ e di $\Gamma' \vdash \Delta'$ ambedue di grado minore di d. Allora è possibile costruire una dimostrazione Ψ di $\Gamma, \Gamma' \smallsetminus P \vdash \Delta \smallsetminus P, \Delta'$ di grado minore di d.*

Dimostrazione. Ψ è costruito per induzione su $h(\Phi) + h(\Phi')$, ma sfortunatamente non in maniera simmetrica rispetto Φ e Φ': ad un certo punto la preferenza sarà data a Φ od a Φ', e Ψ sarà irreversibilmente affetta da questa scelta.

Siano Φ e Φ' rispettivamente:

$$\dfrac{\overset{\Phi_1}{\Vdash} \quad\quad \overset{\Phi_n}{\Vdash}}{\Gamma_1 \vdash \Delta_1 \quad \cdots \quad \Gamma_n \vdash \Delta_n}\,(\rho) \qquad \dfrac{\overset{\Phi'_1}{\Vdash} \quad\quad \overset{\Phi'_m}{\Vdash}}{\Gamma'_1 \vdash \Delta'_1 \quad \cdots \quad \Gamma'_m \vdash \Delta'_m}\,(\rho')$$
$$\Gamma \vdash \Delta \qquad\qquad\qquad \Gamma' \vdash \Delta'$$

e siano $i \in \{1, \ldots, n\}$ e $j \in \{1, \ldots, m\}$. Ci sono vari casi da considerare:

1. Φ è un assioma. Ci sono due sottocasi:

   (a) Φ prova $P \vdash P$. Allora la dimostrazione Ψ di $P, \Gamma' \smallsetminus P \vdash \Delta'$ è ottenuta da Φ' mediante l'applicazione di regole strutturali.

   (b) Φ prova $Q \vdash Q$, con $Q \neq P$. Anche in questo caso applichiamo regole strutturali a Φ' per ottenere $Q, \Gamma' \smallsetminus Q \vdash Q, \Delta'$.

2. Φ' è un assioma. Questo caso è del tutto analogo al precedente; è interessante notare che se Φ e Φ' sono entrambi assiomi, abbiamo arbitrariamente privilegiato Φ (e questo potrebbe avere delle ripercussioni sulla complessità di Ψ).

3. (ρ) è una regola strutturale. L'ipotesi induttiva per Φ_1 e Φ' ci danno una dimostrazione Ψ_1 per $\Gamma_1, \Gamma' \smallsetminus P \vdash \Delta_1 \smallsetminus P, \Delta'$. Allora Ψ è ottenuto da Ψ_1 mediante regole strutturali. Questo è possibile perché, qualunque sia la regola strutturale (ρ), questa gode della proprietà della sottoformula, e quindi Γ_1 è composto esclusivamente di sottoformule di Γ, così come $\Delta_1 \smallsetminus P$ è composto solo di sottoformule di $\Delta \smallsetminus P$. Quindi per ottenere il

sequente conclusivo di Ψ, non dovrà essere tolta alcuna formula presente nella conclusione di Ψ_1, ma al massimo lo si dovrà *indebolire*.

4. (ρ') è una regola strutturale: analogo al precedente.

5. (ρ) è una regola logica, tranne una regola logica destra che introduce P. L'ipotesi induttiva per Φ_i e Φ' ci da n dimostrazioni Ψ_i di $\Gamma_i, \Gamma' \setminus P \vdash \Delta_i \setminus P, \Delta'$. Poiché la regola (ρ) non introduce nuove occorrenze di P a destra del tornello, questa è applicabile alle Ψ_i per ottenere Ψ: $\Gamma, \Gamma' \setminus P \vdash \Delta \setminus P, \Delta'$.

6. (ρ') è una regola logica: analogo al precedente.

7. Sia (ρ) che (ρ') sono regole logiche: (ρ) è una regola logica destra che introduce P, mentre (ρ') è una regola logica sinistra che introduce P. Questo è l'ultimo caso rimanente, nonché l'unico interessante, ed è simmetrico. Per ipotesi induttiva, applicata a:

   (a) Φ_i e Φ', otteniamo le dimostrazioni Ψ_i di $\Gamma_i, \Gamma' \setminus P \vdash \Delta_i \setminus P, \Delta'$; ora, applicando (ρ) alle Ψ_i, e usando delle regole strutturali, otteniamo la dimostrazione Υ di $\Gamma, \Gamma' \setminus P \vdash P, \Delta \setminus P, \Delta'$;

   (b) Φ e Φ'_j, otteniamo le dimostrazioni Ψ'_j di $\Gamma, \Gamma'_j \setminus P \vdash \Delta \setminus P, \Delta'_j$; ora, applicando (ρ') alle Ψ'_j, e con l'ausilio di regole strutturali, otteniamo la dimostrazione Υ' di $\Gamma, \Gamma' \setminus P, P \vdash \Delta \setminus P, \Delta'$.

   Ora abbiamo due dimostrazioni, Υ e Υ', che si concludono come richiesto, se non per un'occorrenza di troppo della formula P. Applicando la regola di taglio ad Υ e Υ', otteniamo una dimostrazione Υ'' di:

$$\dfrac{\begin{array}{cc} \Upsilon\Big\| & \Upsilon'\Big\| \\ \Gamma, \Gamma' \setminus P \vdash P, \Delta \setminus P, \Delta' & \Gamma, \Gamma' \setminus P, P \vdash \Delta \setminus P, \Delta' \end{array}}{\Gamma, \Gamma' \setminus P, \Gamma, \Gamma' \setminus P \vdash \Delta \setminus P, \Delta', \Delta \setminus P, \Delta'} \,(\mathsf{cut})$$

   che con semplici manipolazioni strutturali è riducibile a:

$$\Gamma, \Gamma' \setminus P \vdash \Delta \setminus P, \Delta'$$

Tuttavia il grado del taglio usato in Υ'' è troppo elevato (è proprio di grado d). Ma questo è precisamente il caso in cui si applica il Lemma 2.1.2, grazie al quale il taglio in Υ'' può essere rimpiazzato con una derivazione di grado minore di d, e avente la stessa conclusione, dalla quale, mediante regole strutturali, possiamo ottenere Ψ.

□

Il prossimo lemma, che ci condurrà al risultato finale, afferma che è sempre possibile trasformare una dimostrazione in modo tale da diminuirne il grado. Formalmente:

Lemma 2.1.5. *Sia Φ una dimostrazione di grado $d > 0$ per un certo sequente. Allora è possibile costruire una dimostrazione Ψ per il medesimo sequente, avente grado inferiore.*

Dimostrazione. Per induzione sull'altezza $h(\Phi)$ della dimostrazione iniziale. Sia (ρ) l'ultima regola applicata in Φ e siano Φ_i le premesse di (ρ). Abbiamo due casi:

1. (ρ) non è un taglio di grado d. Per ipotesi induttiva, abbiamo Ψ_i di grado minore di d, a cui possiamo applicare (ρ) per ottenere Ψ;

2. (ρ) è un taglio di grado d:

$$\dfrac{\begin{array}{cc}\Phi_1\Big\| & \Phi_2\Big\| \\ \Gamma \vdash P, \Delta & \Gamma', P \vdash \Delta'\end{array}}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\,(\mathsf{cut})$$

   Osserviamo che poiché il grado di questo (cut) è d, abbiamo $\delta(P) = d$. Per ipotesi induttiva:

$$\begin{array}{cc}\Psi_1\Big\| & \Psi_2\Big\| \\ \Gamma \vdash P, \Delta & \Gamma', P \vdash \Delta'\end{array}$$

   hanno grado minore di d, e possiamo applicarvi il Lemma 2.1.4 per produrre $\Gamma, \Gamma' \smallsetminus P \vdash \Delta \smallsetminus P, \Delta'$ di grado inferiore a d; con alcune applicazioni di regole strutturali, otteniamo infine Ψ.

□

Teorema 2.1.6 (Gentzen Hauptsatz)**.** *La regola di taglio è ammissibile nel Sistema* LKp*.*

Dimostrazione. È sufficiente iterare l'applicazione del lemma precedente per trasformare una dimostrazione di grado strettamente positivo, in una di grado nullo, e quindi esente da applicazioni della regola di taglio. □

Il processo di eliminazione dei tagli fa esplodere l'altezza delle dimostrazioni. Infatti il Lemma 2.1.4 fa crescere l'altezza della prova in modo lineare nel caso peggiore (di un fattore $\kappa = 4$, senza considerare le applicazioni delle regole strutturali). Il Lemma 2.1.5 comporta una crescita esponenziale nel caso pessimo, cioè ridurre il grado di 1 può accrescere l'albero di prova da h a κ^h, poiché usando il Lemma 2.1.4 moltiplichiamo per κ ad ogni unità di altezza.

Quindi, mettendo tutto assieme, applicare l'Hauptsatz comporta una crescita iperesponenziale. Partendo da una dimostrazione di grado d e altezza h se ne ottiene una avente altezza $\mathcal{H}(d, h)$, dove:

$$\begin{aligned} \mathcal{H}(0, h) &= h \\ \mathcal{H}(d+1, h) &= \kappa^{\mathcal{H}(d,h)} \end{aligned}$$

L'Hauptsatz – in varie forme, come la normalizzazione nel λ-calcolo – è utilizzabile come fondamento teorico per la computazione. Per esempio, consideriamo un editor di testo: può essere visto come un insieme di lemmi generici (corrispondenti alle varie procedure di formattazione, impaginazione, ...) che possono essere applicati a input concreti, come una pagina scritta da qualche utente. Il numero di input possibili è chiaramente infinito e infatti i lemmi sono fatti per trattare infiniti casi; ma quando eseguiamo il programma su un certo input – ad esempio per produrre in output una visualizzazione del testo – i riferimenti a queste infinità scompaiono. Concretamente, questa eliminazione dell'infinito è effettuata sostituendo sistematicamente le variabili (gli input dei lemmi) con il testo inserito dall'utente, in altre parole, eseguendo il programma.

Questo è esattamente quello che fa l'algoritmo di eliminazione del taglio. Ecco perché la struttura della procedure di cut elimination è così importante (osservazione fatta nel Lemma 2.1.4). La strategia adottata nel ristrutturare la dimostrazione, effettuando le sostituzioni, produce delle scelte che sono, in generale, irreversibili. Questo può essere un problema, e si può risolvere ad esempio usando, al posto del calcolo dei sequenti, la deduzione naturale, che gode della proprietà di *confluenza* (o *proprietà di Church-Rosser*), la quale garantisce che *le scelte fatte sono sempre reversibili*. Purtroppo la deduzione naturale soffre di altri problemi, e specialmente non gode della proprietà della sottoformula, e non si relaziona bene con la simmetria classica (ha molte premesse ma una sola conclusione). L'approccio deep inference può offrire diversi vantaggi nei confronti di ambedue questi formalismi.

2.2 Deep inference e simmetria

L'eliminazione del taglio è un'idea centrale della proof theory. Se spostiamo tutto alla destra del turnstile e applichiamo qualche regola strutturale, la regola di taglio diventa:

$$\frac{\vdash P, \Delta \quad \vdash \neg P, \Delta'}{\vdash \Delta, \Delta'}\ (\mathsf{cut}^1)$$

Quando letta dal basso verso l'alto, la regola di taglio introduce una formula arbitraria P, insieme alla sua negazione $\neg P$. Osserviamo ora la regola d'identità, manipolata nuovamente per portare tutto a destra del tornello:

$$\vdash P, \neg P \qquad (\mathsf{id}^1)$$

Ci accorgiamo che, quando letta dall'alto al basso, anch'essa introduce una formula arbitraria assieme alla sua negazione. È chiaro che le due regole sono intimamente correlate. Tuttavia, la loro dualità è oscurata dal fatto che le simmetrie verticali sono nascoste nel calcolo dei sequenti: le derivazioni sono alberi, e gli alberi sono verticalmente asimmetrici.

Per rivelare la dualità tra le due regole, occorre ripristinare questa simmetria verticale. La forma ad albero delle derivazioni nel calcolo dei sequenti è dovuta alla presenza di regole d'inferenza con due premesse. Per esempio la regola destra di congiunzione, nella versione ad un lato diventa:

$$\frac{\vdash P, \Delta \quad \vdash Q, \Delta'}{\vdash P \wedge Q, \Delta, \Delta'} \, (\wedge_{\mathsf{r}}^{1})$$

in cui è presente un'asimmetria: due premesse ma solo una conclusione. O per meglio dire: un connettivo nella conclusione, ma nessuno tra le premesse.

Questa asimmetria può essere riparata. Sappiamo che la virgola a destra del turnstile corrisponde alla disgiunzione, e che i diversi rami dell'albero di derivazione corrispondono a congiunzioni; pertanto la regola $(\wedge_{\mathsf{r}}^{1})$ può essere riscritta come:

$$\frac{\vdash (P \vee \Delta) \wedge (Q \vee \Delta')}{\vdash (P \wedge Q) \vee \Delta \vee \Delta'} \, (\wedge_{\mathsf{r}}^{1.1})$$

In tal modo andiamo ad indentificare una parte del livello oggetto (i connettivi tra le formule) con il meta-livello (i rami dell'albero di derivazione). Così facendo rendiamo il sistema "incompleto", poiché uno degli scopi degli alberi di derivazione è quello di permettere alle regole d'inferenza di essere applicate in profondità, seguendo la struttura sintattica delle formule. Consideriamo la derivazione:

$$\dfrac{\dfrac{\cdots}{\vdash \mathbb{C}\{P \wedge Q\}} \qquad \dfrac{\vdash P, \Delta \quad \vdash Q, \Delta'}{\vdash P \wedge Q, \Delta, \Delta'} \, (\wedge_{r}^{1})}{}$$

in cui la conclusione contiene la sottoformula $P \wedge Q$. Leggendola dal basso all'alto, il motivo per cui la regola $(\wedge_{\mathsf{r}}^{1})$ può essere applicata, è che le applicazioni nella derivazione sottostante decompongono il contesto $\mathbb{C}\{\bullet\}$ e ne distribuiscono il contenuto tra le foglie dell'albero di derivazione.

Se vogliamo eliminare la forma ad albero delle derivazioni per ottenere un sistema completamente simmetrico, dobbiamo in qualche modo riconferire alle derivazioni l'abilità di accedere alle sottoformule: questo può essere fatto direttamente, usando la metodologia deep inference. In questo modo, l'assioma d'identità e la regola di taglio diventano:

$$\frac{\mathsf{t}}{P \vee \neg P}\,(\mathsf{id}) \qquad\qquad \frac{P \wedge \neg P}{\mathsf{f}}\,(\mathsf{cut})$$

da cui è evidente il carattere duale delle due: una può essere ottenuta dall'altra scambiando e negando la premessa e la conclusione. A questa nozione di dualità ci si riferisce con l'aggettivo *contrappositiva*.

Avremo modo di osservare una profonda simmetria, tutte le regole d'inferenza si raggrupperanno in coppie duali, come identità e taglio. Questa dualità si estenderà naturalmente alle derivazioni: per ottenere la duale di una derivazione, basterà negare ogni formula e "girare la derivazione sottosopra", cioè leggerla dal basso verso l'alto.

2.2.1 Sistema SKS generalizzato

Presentiamo un sistema formale per la logica classica proposizionale, che da una parte segue la tradizione del calcolo dei sequenti, in particolare possiede una regola di taglio e la sua ammissibilità è dimostrata, mentre dall'altra, in contrasto col calcolo dei sequenti, ha regole che si applicano a profondità arbitraria nelle formule e le derivazioni sono alberi degeneri (i.e. liste, le regole hanno al più una premessa). In questo Sistema potremo osservare una simmetria verticale nelle regole che mancava nel calcolo dei sequenti.

Definizione 2.2.1 (Linguaggio SKSg, equivalenza)**.** *Sia $\mathcal{P}$ un insieme infinito enumerabile di* simboli proposizionali*. L'insieme degli* atomi *$\mathcal{A}$ è così definito:*

$$\mathcal{A} = \{p, \overline{p} \mid p \in \mathcal{P}\}$$

dove $\bar{\cdot}$ è una funzione di negazione primitiva sui simboli proposizionali*. La*

negazione si estende facilmente a tutti gli atomi definendo $\overline{\overline{p}} = p$ *per ogni simbolo proposizionale negato* $\overline{p}$*.*

Siano $\mathsf{t}, \mathsf{f} \notin \mathcal{A}$ *simboli costanti o* unità *(che denotano rispettivamente* il vero *ed* il falso*) e sia* $a \in \mathcal{A}$*. Il* linguaggio di SKSg *è definito dalle seguenti regole BNF di produzione:*

$$\begin{array}{lcll} T & ::= & \mathsf{t} \mid \mathsf{f} \mid a & \textit{(termini)} \\ P & ::= & T \mid (P, P) \mid [P, P] & \textit{(formule)} \end{array}$$

dove (P_1, P_2) *e* $[P_1, P_2]$ *denotano rispettivamente la* congiunzione *e la* disgiunzione *delle formule* P_1 *e* P_2*. Come prima, dato un contesto* $\mathbb{C}\{\bullet\}$ *e una formula* P*, indichiamo con* $\mathbb{C}\{P\}$ *la formula ottenuta saturando il contesto* $\mathbb{C}$ *con la formula* P*. Ad esempio, sia* $\mathbb{C}\{\bullet\} = [a, (\bullet, c)]$*: allora* $\mathbb{C}\{\overline{b}\} = [a, (\overline{b}, c)]$ *mentre* $\mathbb{C}\{(b_1, b_2)\} = [a, ((b_1, b_2), c)]$*; in quest'ultimo caso possiamo adottare la convenzione di* omettere le parentesi graffe attorno ai termini composti *e pertanto di scrivere semplicemente* $\mathbb{C}(b_1, b_2)$*.*

Consideriamo due formule equivalenti quando appartengono alla relazione indotta dalle equazioni in Figura 2.1.

Infine, una formula è in forma normale negata *quando la negazione occorre solo sui simboli proposizionali.*

In virtù della proprietà associativa, adottiamo la seguente convenzione:

$$\begin{array}{rcl} [P_1, P_2, \ldots, P_n] & = & [P_1, [P_2, [\ldots, P_n] \ldots]] \\ (P_1, P_2, \ldots, P_n) & = & (P_1, (P_2, (\ldots, P_n) \ldots)) \end{array}$$

cioè scriviamo rispettivamente liste di disgiunzioni e congiunzioni senza curarci di come le sottoformule siano associate tra loro, assumendo quando non specificato che associno a destra.

Osserviamo inoltre che l'equivalenza ci permette di spingere la negazione all'interno delle formule fino a livello degli atomi, scambiando ogni volta congiunzione e disgiunzione in stile De Morgan.

Definizione 2.2.2 (Dualità, simmetria)**.** *Il* duale di una regola *d'inferenza si ottiene scambiando la premessa con la conclusione e negando ambedue. Ad*

Associatività

$$[[P,Q],R] = [P,[Q,R]]$$
$$((P,Q),R) = (P,(Q,R))$$

Commutatività

$$[P,Q] = [Q,P]$$
$$(P,Q) = (Q,P)$$

Unità

$$[\mathsf{t},\mathsf{t}] = \mathsf{t} \qquad [\mathsf{f},P] = P$$
$$(\mathsf{f},\mathsf{f}) = \mathsf{f} \qquad (\mathsf{t},P) = P$$

Negazione

$$\begin{aligned} \overline{\mathsf{f}} &= \mathsf{t} \\ \overline{\mathsf{t}} &= \mathsf{f} \\ \overline{[P,Q]} &= (\overline{P},\overline{Q}) \\ \overline{(P,Q)} &= [\overline{P},\overline{Q}] \\ \overline{\overline{P}} &= P \end{aligned}$$

Chiusura contestuale

$$\frac{P = Q}{\mathbb{C}\{P\} = \mathbb{C}\{Q\}}$$

Equivalenza

$$P = P \qquad \frac{P = Q}{Q = P} \qquad \frac{P = Q \qquad Q = R}{P = R}$$

Figura 2.1: Equivalenza tra formule di SKSg

esempio:

$$\frac{\mathsf{t}}{[P,\overline{P}]}\,(\mathsf{i}{\downarrow}) \qquad \frac{(P,\overline{P})}{\mathsf{f}}\,(\mathsf{i}{\uparrow})$$

Un sistema deduttivo *è* simmetrico *se per ogni regola d'inferenza esso contiene anche la duale.*

Il Sistema deduttivo SKSg è riportato in Figura 2.2. Il suo nome è un acronimo, in cui la prima "S" indica che è simmetrico, la "K" sta per "Klassisch" (come nel Sistema LK) e la "S" finale dice che il Sistema è espresso nel calcolo delle strutture (il termine "struttura" è usato per indicare una lista di formule in congiunzione o in disgiunzione). La "g" minuscola indica che il Sistema è *generalizzato*, che significa che le regole non sono ristrette alla forma atomica.

È possibile dimostrare (Brünnler [2004]) che questo sistema formale cattura tutte le dimostrazioni esprimibili in LKp, passando per un sistema in-

$$\mathsf{t}\quad(\mathsf{ax}) \qquad \frac{\mathbb{C}\{\mathsf{t}\}}{\mathbb{C}[P,\overline{P}]}\,(\mathsf{i}{\downarrow}) \qquad \frac{\mathbb{C}\{\mathsf{f}\}}{\mathbb{C}\{P\}}\,(\mathsf{w}{\downarrow}) \qquad \frac{\mathbb{C}[P,P]}{\mathbb{C}\{P\}}\,(\mathsf{c}{\downarrow}) \qquad \frac{\mathbb{C}(P,[Q,R])}{\mathbb{C}[(P,Q),R]}\,(\mathsf{s})$$

$$\text{(nessuna regola per } \mathsf{f}) \qquad \frac{\mathbb{C}(P,\overline{P})}{\mathbb{C}\{\mathsf{f}\}}\,(\mathsf{i}{\uparrow}) \qquad \frac{\mathbb{C}\{P\}}{\mathbb{C}\{\mathsf{t}\}}\,(\mathsf{w}{\uparrow}) \qquad \frac{\mathbb{C}\{P\}}{\mathbb{C}(P,P)}\,(\mathsf{c}{\uparrow})$$

Figura 2.2: Sistema deduttivo SKSg

termedio chiamato calcolo dei sequenti "ad un lato" o calcolo dei sequenti di Gentzen-Schütte (Schütte [1950]; Troelstra and Schwichtenberg [1996]).

Le regole (s), (w↓) e (c↓) sono chiamate rispettivamente *scambio*, *indebolimento* e *contrazione*. Le duali portano lo stesso nome, con l'aggiunta del prefisso "co-", ad esempio (w↑) è chiamata *co-indebolimento*. La regola di scambio è duale a sé stessa, o *auto-duale*.

Mentre è immediato osservare la corrispondenza tra (w↓) e (c↓) in SKS e le regole di indebolimento e contrazione nel calcolo dei sequenti, le loro duali non hanno corrispettivi in LKp. Il loro ruolo è quello di assicurare la simmetria del Sistema; se non siamo interessati alla simmetria, si può dimostrare che queste regole (e anche il taglio, cioè tutte le regole aventi la freccia rivolta verso l'alto) sono ammissibili. Infatti la nozione di dimostrazione è inerentemente asimmetrica: il duale di una dimostrazione *non è* una dimostrazione, bensì è una derivazione che si conclude con l'unità f, ossia una *refutazione*.

Il primo meta-teorema che andremo a dimostrare, ci dà una caratterizzazione del Sistema SKSg, mettendo in relazione il concetto di derivazione con quello di dimostrazione.

Teorema 2.2.3 (Deduzione)**.**

Esiste una derivazione $\begin{array}{c} P \\ \Psi \Vert\, \mathsf{SKSg} \\ Q \end{array}$ *se e solo se esiste una dimostrazione* $\begin{array}{c} \Phi \Vert\, \mathsf{SKSg} \\ [\overline{P}, Q] \end{array}$.

Dimostrazione. La dimostrazione Φ può essere ottenuta, data una derivazio-

ne Ψ, come segue:

$$\dfrac{\dfrac{\dfrac{\mathsf{t}}{[\overline{P}, P]}\,(\mathsf{i}{\downarrow})}{[\overline{P},\Psi]}\,\mathsf{SKSg}}{[\overline{P}, Q]}$$

Osserviamo che grazie alla metodologia deep inference, è stato possibile racchiudere l'intera derivazione Ψ all'interno del contesto $\mathbb{C}\{\bullet\} = [\overline{P}, \bullet]$.

La derivazione Ψ si ottiene da Φ come segue:

$$\dfrac{\dfrac{\dfrac{P = (P, \mathsf{t})}{(P,\Phi)}\,\mathsf{SKSg}}{\dfrac{(P, [\overline{P}, Q])}{[(P, \overline{P}), Q]}\,(\mathsf{s})}}{[\mathsf{f}, Q] = Q}\,(\mathsf{i}{\uparrow})$$

Anche in questo caso la trasformazione ha avuto successo perché è stato possibile usare la dimostrazione Φ nel contesto $\mathbb{C}'\{\bullet\} = (P, \bullet)$. □

2.2.2 Località: il Sistema SKS

Le regole d'inferenza che duplicano una quantità illimitata di informazione sono problematiche dal punto di vista della complessità e dell'implementazione, ad esempio, della proof search. Nel calcolo dei sequenti, la regola di contrazione:

$$\dfrac{\Gamma, P, P \vdash \Delta}{\Gamma, P \vdash \Delta}\,(\mathsf{cont_l})$$

quando letta dall'alto al basso duplica una formula P di dimensione arbitraria. Qualunque sia il meccanismo effettivo che compie questa duplicazione, esso necessita di una visione *globale* delle copie di P presenti: se ad esempio pensiamo di implementare la contrazione su un sistema distribuito, in cui ogni processore ha una quantità limitata di memoria locale, la formula P

potrebbe essere replicata in processori diversi. In questo caso nessun processore avrebbe una visone globale delle copie di P, e bisognerebbe usare un meccanismo *ad hoc* per gestire questa situazione. Chiamiamo *locali* le regole d'inferenza che non necessitano di una visione globale su formule di dimensione arbitraria, e *non-locali* le altre.

Mentre è possibile utilizzare tecniche per risolvere questa situazione nelle implementazioni, una questione interessante è trovare un approccio teorico che sia in grado di eliminare le regole non-locali. Questo è possibile, riducendo le regole non-locali alla loro forma atomica. Ad esempio, l'identità:

$$\frac{}{\mathbb{C}[P, \overline{P}]}\,(\mathsf{i}\downarrow) \qquad \text{è sostituita dalla regola} \qquad \frac{}{\mathbb{C}[a, \overline{a}]}\,(\mathsf{ai}\downarrow)$$

dove a è un simbolo proposizionale.

Operazioni analoghe possono essere fatte anche nel calcolo dei sequenti; l'unica regola problematica è, appunto, la contrazione. Essa non può semplicemente essere ristretta alla forma atomica nel Sistema SKSg. Il problema si risolve inserendo nel Sistema una nuova regola, introdotta in Brünnler and Tiu [2001] e chiamata *mediale*:

$$\frac{\mathbb{C}[(P, Q), (R, S)]}{\mathbb{C}([P, R], [Q, S])}\,(\mathsf{m})$$

Questa regola non ha analoghi nel calcolo dei sequenti, ma è chiaramente corretta, poiché è derivabile da $\{(\mathsf{c}\downarrow), (\mathsf{w}\downarrow)\}$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[(P, Q), (R, S)]}{[(P, Q), (R, [Q, S])]}\,(\mathsf{w}\downarrow)}{[(P, Q), ([P, R], [Q, S])]}\,(\mathsf{w}\downarrow)}{[(P, [Q, S]), ([P, R], [Q, S])]}\,(\mathsf{w}\downarrow)}{\mathbb{C}[([P, R], [Q, S]), ([P, R], [Q, S])]}\,(\mathsf{c}\downarrow)}{\mathbb{C}([P, R], [Q, S])}$$

Il prossimo teorema ci garantisce la derivabilità del Sistema locale KS in Figura 2.3:

$$\mathsf{t}\quad(\mathsf{ax}) \qquad \frac{\mathbb{C}\{\mathsf{t}\}}{\mathbb{C}[a,\overline{a}]}\,(\mathsf{ai}{\downarrow}) \qquad \frac{\mathbb{C}\{\mathsf{f}\}}{\mathbb{C}\{a\}}\,(\mathsf{aw}{\downarrow}) \qquad \frac{\mathbb{C}[a,a]}{\mathbb{C}\{a\}}\,(\mathsf{ac}{\downarrow})$$

$$\text{(nessuna regola per } \mathsf{f}) \qquad \frac{\mathbb{C}(a,\overline{a})}{\mathbb{C}\{\mathsf{f}\}}\,(\mathsf{ai}{\uparrow}) \qquad \frac{\mathbb{C}\{a\}}{\mathbb{C}\{\mathsf{t}\}}\,(\mathsf{aw}{\uparrow}) \qquad \frac{\mathbb{C}\{a\}}{\mathbb{C}(a,a)}\,(\mathsf{ac}{\uparrow})$$

$$\frac{\mathbb{C}(P,[Q,R])}{\mathbb{C}[(P,Q),R]}\,(\mathsf{s}) \qquad\qquad \frac{\mathbb{C}[(P,Q),(R,S)]}{\mathbb{C}([P,R],[Q,S])}\,(\mathsf{m})$$

Figura 2.3: Regole del Sistema *locale* SKS

Teorema 2.2.4. *Le regole* $(\mathsf{i}{\downarrow})$, $(\mathsf{w}{\downarrow})$ *e* $(\mathsf{c}{\downarrow})$ *sono derivabili, rispettivamente da* $\{(\mathsf{ai}{\downarrow}),(\mathsf{s})\}$, $\{(\mathsf{aw}{\downarrow}),(\mathsf{s})\}$, $\{(\mathsf{ac}{\downarrow}),(\mathsf{m})\}$. *Dualmente, le regole* $(\mathsf{i}{\uparrow})$, $(\mathsf{w}{\uparrow})$ *e* $(\mathsf{c}{\uparrow})$ *sono risp. derivabili da* $\{(\mathsf{ai}{\uparrow}),(\mathsf{s})\}$, $\{(\mathsf{aw}{\uparrow}),(\mathsf{s})\}$, $\{(\mathsf{ac}{\uparrow}),(\mathsf{m})\}$.

Dimostrazione. Data un'istanza di una delle seguenti regole:

$$\frac{\mathbb{C}\{\mathsf{t}\}}{\mathbb{C}[P,\overline{P}]}\,(\mathsf{i}{\downarrow}) \quad , \quad \frac{\mathbb{C}\{\mathsf{f}\}}{\mathbb{C}\{P\}}\,(\mathsf{w}{\downarrow}) \quad , \quad \frac{\mathbb{C}[P,P]}{\mathbb{C}\{P\}}\,(\mathsf{c}{\downarrow})$$

costruiamo una nuova derivazione per induzione strutturale su P:

- P è un atomo. Allora l'istanza di una regola generale è anche un'istanza della corrispettiva in forma atomica.

- $P = \mathsf{t}$ o $P = \mathsf{f}$. Allora l'istanza di una regola generale è un'istanza della relazione d'equivalenza, con l'eccezione dell'indebolimento quando $P = \mathsf{f}$. Allora la regola d'indebolimento generale è sostituita da:

$$\frac{\mathbb{C}\{\mathsf{f}\} = \mathbb{C}(\mathsf{f},[\mathsf{t},\mathsf{t}])}{\mathbb{C}[(\mathsf{f},\mathsf{t}),\mathsf{t}] = \mathbb{C}\{\mathsf{t}\}}\,(\mathsf{s})$$

- $P = [Q, R]$. Per ipotesi induttiva, usando rispettivamente le sole regole $\{(\mathsf{ai}{\downarrow}), (\mathsf{s})\}$, $\{(\mathsf{aw}{\downarrow}), (\mathsf{s})\}$ e $\{(\mathsf{ac}{\downarrow}), (\mathsf{m})\}$, abbiamo:

$$\begin{array}{cc} \mathbb{C}\{\mathsf{t}\} & \mathbb{C}\{\mathsf{t}\} \\ {\scriptstyle \Phi_Q^{(\mathsf{i}\downarrow)}}\big\| & {\scriptstyle \Phi_R^{(\mathsf{i}\downarrow)}}\big\| \\ \mathbb{C}[Q, \overline{Q}] & \mathbb{C}[R, \overline{R}] \end{array} \quad , \quad \begin{array}{c} \mathbb{C}\{\mathsf{f}\} \\ {\scriptstyle \Phi_Q^{(\mathsf{w}\downarrow)}}\big\| \\ \mathbb{C}\{Q\} \end{array} \quad \begin{array}{c} \mathbb{C}\{\mathsf{f}\} \\ {\scriptstyle \Phi_R^{(\mathsf{w}\downarrow)}}\big\| \\ \mathbb{C}\{R\} \end{array} \quad , \quad \begin{array}{c} \mathbb{C}[Q, Q] \\ {\scriptstyle \Phi_Q^{(\mathsf{c}\downarrow)}}\big\| \\ \mathbb{C}\{Q\} \end{array} \quad \begin{array}{c} \mathbb{C}[R, R] \\ {\scriptstyle \Phi_R^{(\mathsf{c}\downarrow)}}\big\| \\ \mathbb{C}\{R\} \end{array}$$

  da cui è possibile derivare:

$$\begin{array}{c} \mathbb{C}\{\mathsf{t}\} \\ {\scriptstyle \Phi_R^{(\mathsf{i}\downarrow)}}\big\| \\ \mathbb{C}[\overline{R}, R] \\ {\scriptstyle \Phi_Q^{(\mathsf{i}\downarrow)}}\big\| \\ \dfrac{\mathbb{C}([\overline{Q}, Q], [\overline{R}, R])}{\mathbb{C}[(\overline{R}, [\overline{Q}, Q]), R]}\,(\mathsf{s}) \\ \dfrac{}{\mathbb{C}[(\overline{Q}, \overline{R}), [Q, R]]}\,(\mathsf{s}) \end{array} \quad , \quad \begin{array}{c} \mathbb{C}\{\mathsf{f}\} = \mathbb{C}[\mathsf{f}, \mathsf{f}] \\ {\scriptstyle \Phi_Q^{(\mathsf{w}\downarrow)}}\big\| \\ \mathbb{C}[Q, \mathsf{f}] \\ {\scriptstyle \Phi_R^{(\mathsf{w}\downarrow)}}\big\| \\ \mathbb{C}[Q, R] \end{array} \quad , \quad \begin{array}{c} \mathbb{C}[Q, Q, R, R] \\ {\scriptstyle \Phi_Q^{(\mathsf{c}\downarrow)}}\big\| \\ \mathbb{C}[Q, R, R] \\ {\scriptstyle \Phi_R^{(\mathsf{c}\downarrow)}}\big\| \\ \mathbb{C}[Q, R] \end{array}$$

- $P = (Q, R)$. L'ipotesi induttiva è identica a quella del caso precedente, da cui è possibile derivare:

$$\begin{array}{c} \mathbb{C}\{\mathsf{t}\} \\ {\scriptstyle \Phi_R^{(\mathsf{i}\downarrow)}}\big\| \\ \mathbb{C}[R, \overline{R}] \\ {\scriptstyle \Phi_Q^{(\mathsf{i}\downarrow)}}\big\| \\ \dfrac{\mathbb{C}([Q, \overline{Q}], [R, \overline{R}])}{\mathbb{C}[(R, [Q, \overline{Q}]), \overline{R}]}\,(\mathsf{s}) \\ \dfrac{}{\mathbb{C}[(Q, R), [\overline{Q}, \overline{R}]]}\,(\mathsf{s}) \end{array} \quad , \quad \begin{array}{c} \mathbb{C}\{\mathsf{f}\} = \mathbb{C}(\mathsf{f}, \mathsf{f}) \\ {\scriptstyle \Phi_Q^{(\mathsf{w}\downarrow)}}\big\| \\ \mathbb{C}(Q, \mathsf{f}) \\ {\scriptstyle \Phi_R^{(\mathsf{w}\downarrow)}}\big\| \\ \mathbb{C}(Q, R) \end{array} \quad , \quad \begin{array}{c} \dfrac{\mathbb{C}[(Q, R), (Q, R)]}{\mathbb{C}([Q, Q], [R, R])}\,(\mathsf{m}) \\ {\scriptstyle \Phi_Q^{(\mathsf{c}\downarrow)}}\big\| \\ \mathbb{C}(Q, [R, R]) \\ {\scriptstyle \Phi_R^{(\mathsf{c}\downarrow)}}\big\| \\ \mathbb{C}(Q, R) \end{array}$$

I casi duali si dimostrano allo stesso modo, "girando sottosopra" le dimostrazioni e negando tutte le formule. □

Questo è un risultato molto significativo e difficilmente ottenibile usando il calcolo dei sequenti. Inoltre la dimostrazione è costruttiva e modulare, caratteristiche che ci permetteranno in seguito di utilizzare regole generalizzate con la consapevolezza di poterle sempre sostituire con una procedura effettiva con le loro versioni atomiche.

2.2.3 Rompere la simmetria: il Sistema KS

Dimostriamo che nel Sistema SKS le regole con la freccia rivolta in alto ($\rho\uparrow$) sono *ammissibili* (e quindi in particolare anche la regola di taglio lo è). Il Sistema risultante dall'eliminazione delle regole ($\rho\uparrow$) è chiamato Sistema KS, ed è riportato in Figura 2.4.

In questa sezione seguiamo la dimostrazione di Brünnler [2004], a cui ho apportato alcune modifiche personali di carattere tecnico.

Lemma 2.2.5. *Ogni regola di SKS è derivabile usando solo la sua duale,* identità, taglio *e* switch.

Dimostrazione. Le regole (s) e (m) sono auto-duali, e pertanto banalmente derivabili. Un'istanza di una regola $\dfrac{\mathbb{C}\{P\}}{\mathbb{C}\{Q\}}\,(\rho\uparrow)$ può essere sostituita da:

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathbb{C}\{P\}}{\mathbb{C}(P,[\overline{Q},Q])}\,(\mathsf{i}\downarrow)}{\mathbb{C}[(P,\overline{Q}),Q]}\,(\mathsf{s})}{\mathbb{C}[(P,\overline{P}),Q]}\,(\rho\downarrow)}{\mathbb{C}\{Q\}}\,(\mathsf{i}\uparrow)$$

e lo stesso vale per le regole ($\rho\downarrow$). □

Prima di proseguire con la cut elimination, occorre stabilire una semplice proposizione, valida per la maggior parte dei sistemi espressi col calcolo delle strutture.

Proposizione 2.2.6. *Per ogni struttura P, Q e contesto $\mathbb{C}$, esiste una deri-*
vazione $\begin{array}{c}\mathbb{C}[P,Q]\\ \Vert\{(\mathsf{s})\}\\ [\mathbb{C}\{P\},Q]\end{array}$.

Dimostrazione. Per induzione sulla dimensione del contesto $\mathbb{C}$.

1. Il caso base è $\mathbb{C}\{\bullet\} = \bullet$, da cui si deriva che esiste una derivazione (vuota) per $[P, Q]$.

2. $\mathbb{C}\{\bullet\} = [R, \mathbb{C}'\{\bullet\}]$. Allora, per ipotesi induttiva, esiste una derivazione:

$$\begin{array}{c} \mathbb{C}'[P, Q] \\ \Pi \Big\| \{(\mathsf{s})\} \\ [\mathbb{C}'\{P\}, Q] \end{array}$$

che può essere usata per costruire:

$$\begin{array}{c} [R, \mathbb{C}'[P, Q]] \\ [R,\Pi] \Big\| \{(\mathsf{s})\} \\ [R, \mathbb{C}'\{P\}, Q] \end{array}$$

e $[R, \mathbb{C}'\{P\}, Q]$ è proprio uguale a $[\mathbb{C}\{P\}, Q]$.

3. $\mathbb{C}\{\bullet\} = (R, \mathbb{C}'\{\bullet\})$. Qui l'ipotesi induttiva ci dà:

$$\begin{array}{c} \mathbb{C}'[P, Q] \\ \Pi \Big\| \{(\mathsf{s})\} \\ [\mathbb{C}'\{P\}, Q] \end{array}$$

che può essere usata per costruire:

$$\dfrac{\begin{array}{c} (R, \mathbb{C}'[P, Q]) \\ (R,\Pi) \Big\| \{(\mathsf{s})\} \\ (R, [\mathbb{C}'\{P\}, Q]) \end{array}}{[(R, \mathbb{C}'\{P\}), Q] = [\mathbb{C}\{P\}, Q]}\ (\mathsf{s})$$

□

Definizione 2.2.7 (Taglio atomico di superficie)**.** *Un'istanza della regola di taglio atomica* (ai↑) *è chiamata* shallow *(o* taglio atomico di superficie*) quando è della forma:*

$$\dfrac{[S, (a, \overline{a})]}{S}\ (\mathsf{ai}{\uparrow})$$

Lemma 2.2.8. *La regola di taglio atomica* (ai↑) *è derivabile usando* taglio atomico di superficie *e* switch.

Dimostrazione. Ogni formula $\mathbb{C}(a, \overline{a})$ è equivalente a $\mathbb{C}[\mathsf{f}, (a, \overline{a})]$. Per la Proposizione 2.2.6, esiste una derivazione:

$$\begin{array}{c} \mathbb{C}[\mathsf{f}, (a, \overline{a})] \\ \Big\| \{(\mathsf{s})\} \\ [\mathbb{C}\{\mathsf{f}\}, (a, \overline{a})] \end{array}$$

Pertanto basta porre $S = \mathbb{C}\{\mathsf{f}\}$ per effettuare la trasformazione:

$$\frac{\mathbb{C}(a, \overline{a})}{\mathbb{C}\{\mathsf{f}\}}\,(\mathsf{ai}{\uparrow}) \qquad \rightsquigarrow \qquad \begin{array}{c} \mathbb{C}(a, \overline{a}) \\ \Big\| \{(\mathsf{s})\} \\ \dfrac{[S, (a, \overline{a})]}{S}\,(\mathsf{ai}{\uparrow}) \end{array}$$

□

Lemma 2.2.9. *Ogni dimostrazione* $\begin{array}{c} \Big\Vert \mathsf{KS} \\ \mathbb{C}\{a\} \end{array}$ *può essere trasformata in* $\begin{array}{c} \Big\Vert \mathsf{KS} \\ \mathbb{C}\{\mathsf{t}\} \end{array}$.

Dimostrazione. Risalendo la dimostrazione, sostituiamo nelle regole l'occorrenza di a e le sue copie prodotte per contrazione, con l'unità t. Le istanze delle regole (s) e (m) rimangono intatte, le istanze di (ac↓) si riducono ad applicazioni della relazione d'equivalenza $=$. Le altre applicazioni vengono sostituite dalle seguenti derivazioni:

$$\frac{\mathbb{C}\{\mathsf{f}\}}{\mathbb{C}\{a\}}\,(\mathsf{aw}{\downarrow}) \qquad \rightsquigarrow \qquad \frac{\mathbb{C}\{\mathsf{f}\} = \mathbb{C}(\mathsf{f}, [\mathsf{t}, \mathsf{t}])}{\mathbb{C}[(\mathsf{f}, \mathsf{t}), \mathsf{t}] = \mathbb{C}\{\mathsf{t}\}}\,(\mathsf{s})$$

$$\frac{\mathbb{C}\{\mathsf{t}\}}{\mathbb{C}[a, \overline{a}]}\,(\mathsf{ai}{\downarrow}) \qquad \rightsquigarrow \qquad \frac{\mathbb{C}\{\mathsf{t}\} = \mathbb{C}[\mathsf{t}, \mathsf{f}]}{\mathbb{C}[\mathsf{t}, \overline{a}]}\,(\mathsf{aw}{\downarrow})$$

□

thbs

$$\mathsf{t}\quad(\mathsf{ax}) \qquad \frac{\mathbb{C}\{\mathsf{t}\}}{\mathbb{C}[a,\overline{a}]}\,(\mathsf{ai}{\downarrow}) \qquad \frac{\mathbb{C}\{\mathsf{f}\}}{\mathbb{C}\{a\}}\,(\mathsf{aw}{\downarrow}) \qquad \frac{\mathbb{C}[a,a]}{\mathbb{C}\{a\}}\,(\mathsf{ac}{\downarrow})$$

$$\frac{\mathbb{C}(P,[Q,R])}{\mathbb{C}[(P,Q),R]}\,(\mathsf{s}) \qquad \frac{\mathbb{C}[(P,Q),(R,S)]}{\mathbb{C}([P,R],[Q,S])}\,(\mathsf{m})$$

Figura 2.4: Regole del Sistema KS

Teorema 2.2.10. *Ogni dimostrazione* $\vdash_{\mathsf{SKS}} P$ *può essere trasformata in una dimostrazione* $\vdash_{\mathsf{KS}} P$.

Dimostrazione. Grazie al Lemma 2.2.5, sappiamo che l'unica regola da eliminare è il taglio $(\mathsf{ai}{\uparrow})$. Grazie al Lemma 2.2.8 possiamo sostituire tutti i tagli con tagli di superficie. Partendo dall'alto, selezioniamo la prima istanza della regola di taglio:

$$\dfrac{\dfrac{\Pi \vdash_{\mathsf{KS}}}{[R,(a,\overline{a})]}\,(\mathsf{ai}{\uparrow})}{R}$$
$$\Phi \vdash_{\mathsf{KS}\cup\{(\mathsf{ai}\uparrow)\}} P$$

Applicando due volte il Lemma 2.2.9 a Π, otteniamo:

$$\Pi_1 \vdash_{\mathsf{KS}} [R,a] \quad , \quad \Pi_2 \vdash_{\mathsf{KS}} [R,\overline{a}]$$

Partendo dalla conclusione e risalendo la dimostrazione Π_1, sostituiamo l'occorrenza di a e le sue copie prodotte per contrazione, con la formula R. Le istanze delle regole (m) e (s) rimangono intatte, mentre le istanze di $(\mathsf{ac}{\downarrow})$

e $(\mathsf{aw}\downarrow)$ vengono sostituite dalle loro versioni generalizzate:

$$\frac{\mathbb{C}[a,a]}{\mathbb{C}\{a\}}\,(\mathsf{ac}\downarrow) \qquad \rightsquigarrow \qquad \frac{\mathbb{C}[R,R]}{\mathbb{C}\{R\}}\,(\mathsf{c}\downarrow)$$

$$\frac{\mathbb{C}\{\mathsf{f}\}}{\mathbb{C}\{a\}}\,(\mathsf{aw}\downarrow) \qquad \rightsquigarrow \qquad \frac{\mathbb{C}\{\mathsf{f}\}}{\mathbb{C}\{R\}}\,(\mathsf{w}\downarrow)$$

Le istanze di $(\mathsf{ai}\downarrow)$ sono sostituite da $\mathbb{C}\{\Pi_2\}$:

$$\frac{\mathbb{C}\{\mathsf{t}\}}{\mathbb{C}[a,\overline{a}]}\,(\mathsf{ai}\downarrow) \qquad \rightsquigarrow \qquad \begin{array}{c}\mathbb{C}\{\mathsf{t}\}\\ \mathbb{C}\{\Pi_2\}\Big\|\mathsf{KS}\\ \mathbb{C}[R,\overline{a}]\end{array}$$

Il risultato di questa sostitituzione di Π_2 dentro Π_1 è una dimostrazione Π_3, grazie alla quale possiamo costruire:

$$\begin{array}{c}\Pi_3\Big\|\mathsf{KS}\\ \dfrac{[R,R]}{R}\,(\mathsf{c}\downarrow)\\ \Phi\Big\|\mathsf{KS}\cup\{(\mathsf{ai}\uparrow)\}\\ P\end{array}$$

Ora basta procedere induttivamente verso il basso per rimuovere le rimanenti istanze di $(\mathsf{ai}\uparrow)$. Alla fine di questo procedimento, le regole generalizzate possono essere rimosse usando la procedura descritta nella dimostrazione del Teorema 2.2.4. □

Capitolo 3

Logica lineare

La logica lineare è un'estensione della logica classica ideata da Jean-Yves Girard verso la fine degli anni '80 (Girard [1987]; Girard et al. [1989]; Girard [1995b]). La caratteristica peculiare della logica lineare è che tratta l'implicazione come *fenomeno causale* anziché (com'è pratica comune in matematica) come *concetto stabile*:

se A e $A \Rightarrow B$ allora B, *ma* A *è ancora valida.*

Un'implicazione causale non può essere reiterata, poiché le condizioni iniziali sono modificate dopo il suo utilizzo; questo processo di modifica delle premesse (condizioni) è noto in fisica come *reazione*[1]. Per esempio, se A è "spendere una moneta nel distributore automatico di bevande (o DAB)" e B è "prendere un caffè", la moneta viene persa nel processo, che quindi non si può ripetere una seconda volta. Esistono tuttavia casi, sia in matematica che nella vita reale, in cui le reazioni non esistono o sono trascurabili: ad esempio un lemma che resta sempre vero, o un tecnico che possiede la chiave del DAB e può recuperare ogni volta la sua moneta. Questi sono i casi che Girard chiama *situazioni*, cioè condizioni durature e immutevoli (o verità stabili), e sono comunque gestibili in logica lineare tramite speciali connettivi (gli *esponenziali*, "!" e "?"). Gli esponenziali esprimono la reiterabilità

[1] Quello di reazione è un concetto base anche della teoria dei modelli concorrenti, vedi ad esempio Milner et al. [1992]; Sangiorgi and Walker [2001].

di un'azione, ossia l'assenza di reazioni; tipicamente $!A$ significa "spendere quante monete si vogliono". Usiamo il simbolo $\multimap$ per denotare l'implicazione causale (o *implicazione lineare*); vale la seguente equazione:

$$A \Rightarrow B \quad = \quad (!A) \multimap B$$

cioè B è causato da un certo numero d'iterazioni di A.

Una *azione di tipo A* consisterà nel tirare fuori una certa moneta dalla tasca di qualcuno (ci potrebbero essere diverse azioni di questo tipo, poiché potremmo disporre di diverse monete). Analogamente saranno disponibili un certo numero di caffè nel distributore automatico, perciò ci saranno diverse *azioni di tipo B*.

La logica lineare apre nuovi interessanti scenari sulla visione dei connettivi classici: ad esempio esistono *due* congiunzioni ($\otimes$ o "per", inteso in senso di moltiplicazione, ed & o "con") corrispondenti a due usi radicalmente differenti della parola "e". Ambedue le congiunzioni esprimono la disponibilità di due azioni; ma nel caso di $\otimes$, saranno fatte tutt'e due, mentre nel caso di &, solo una delle due sarà eseguita (ma noi potremo decidere quale). Ad esempio, siano A, B, C:

A : spendere una moneta nel DAB
B : prendere un caffè
C : prendere un tè

Data un'azione di tipo $A \multimap B$ e una di tipo $A \multimap C$, non sarà possibile formare un'azione di tipo $A \multimap B \otimes C$, poiché per una moneta non si potrà mai avere ciò che ne costa due (sarà invece possibile formare un'azione di tipo $A \otimes A \multimap B \otimes C$, cioè avere due bevande in cambio di due monete). Comunque potremmo sempre produrre un'azione di tipo $A \multimap B \& C$ come sovrapposizione delle due. Per eseguire quest'ultima azione dovremmo prima scegliere tra le possibili azioni che vogliamo produrre e in seguito effettuare quella scelta. Questo è analogo a quanto accade col costrutto if ... then ... else ... ben noto in informatica: infatti, sia la parte then ... che quella else ... sono disponibili, ma solo una di esse verrà eseguita. Per quanto "&" abbia delle

ovvie caratteristiche disgiuntive, sarebbe tecnicamente errato vederlo come disgiunzione: infatti in logica lineare sia $A \,\&\, B \multimap A$, sia $A \,\&\, B \multimap B$ sono dimostrabili.

In logica lineare, in maniera del tutto speculare, abbiamo due disgiunzioni, che sono $\oplus$ o "più", e $\parr$ o "par" (mnemonico per *parallelo*). $\oplus$ è il duale di "&" ed esprime la presenza di due opzioni: in questo caso però, non sarà possibile scegliere quale delle due eseguire. La differenza tra & e $\oplus$ è la stessa che c'è in informatica tra nondeterminismo esterno ed interno. Infine $\parr$ è il duale di $\otimes$.

Il più importante connettivo lineare è la *negazione lineare* $\overline{}$ o "nil". Poiché l'implicazione lineare si può sempre riscrivere come $\overline{A} \parr B$, "nil" è l'unica operazione negativa della logica lineare. La negazione lineare si comporta come la trasposizione in algebra lineare, esprime cioè *dualità*, ovverosia un cambio di prospettiva:

$$\textit{azione di tipo } A = \textit{reazione di tipo } \overline{A}$$

La proprietà principale di "nil" è che, come accade in logica classica, $\overline{\overline{A}}$ può essere identificato con A stesso. A differenza della logica classica però, la logica lineare gode di una *semplice interpretazione costruttiva*. Il carattere involutivo di "nil" assicura il comportamento *alla De Morgan* per tutti i connettivi ed i quantificatori, ad esempio:

$$\exists x.A \quad = \quad \overline{(\forall x.\overline{A})}$$

che può sembrare insolito ad un primo sguardo, specialmente se consideriamo che l'esistenziale in logica lineare è un operatore *effettivo*: tipicamente si dimostra $\exists x.A$ dimostrando $A[t/x]$ per un certo termine t. Questo comportamento di "nil" deriva dal fatto che $\overline{A}$ nega (cioè *reagisce con*) una singola azione di tipo A, mentre la negazione classica nega solo alcune (non specificate) iterazioni di A, che tipicamente porta ad una disgiunzione di lunghezza non specificata. La negazione lineare è da un lato più primitiva, e dall'altro più forte (e anche più difficile da trattare) di quella classica.

Grazie alla presenza degli esponenziali, la logica lineare è espressiva quanto quella classica o quella intuizionista. Di fatto è più espressiva. Qui bisogna essere cauti: è lo stesso problema della logica intuizionista, che è anch'essa "più espressiva" di quella classica. Tecnicamente il potere espressivo è equivalente: ma i connettivi della logica lineare possono esprimere in maniera primitiva cose che in logica classica possono essere espresse solo tramite complesse traduzioni *ad hoc*. L'introduzione di nuovi connettivi è quindi la chiave di volta verso formalizzazioni più semplici ed efficaci; la restrizione a vari frammenti apre le frontiere a linguaggi con specifico potere espressivo, ad esempio con una complessità computazionale nota (Girard [1998]; Lafont [2002]; Dal Lago and Baillot [2006]).

Un notevole problema aperto è quello di trovare una versione convincente di logica lineare non-commutativa. Anche se molti convengono sul fatto che la non-commutatività ha ragione d'esser considerata a questo livello (esistono svariati esempi di problemi intrinsecamente non-commutativi, si pensi all'operatore di prefisso del π-calcolo), semantiche non trivali di non-commutatività non sono note. Unite all'introduzione di una semantica naturale, le metodologie per raggiungere un sistema non-commutativo potrebbero comportare un effettivo guadagno di potere espressivo, in relazione al caso commutativo.

3.1 Calcolo dei sequenti lineari

Definiamo la sintassi della logica lineare classica (o CLL, acronimo di Classical Linear Logic):

Definizione 3.1.1 (Linguaggio CLL)**.** *Sia $\mathcal{P}$ un insieme infinito enumerabile di* simboli proposizionali. *L'insieme degli* atomi $\mathcal{A}$ *è così definito:*

$$\mathcal{A} = \{p, \overline{p} \mid p \in \mathcal{P}\}$$

dove $\overline{\cdot}$ è una funzione di negazione primitiva sui simboli proposizionali. *La*

$$
\begin{array}{rclcrcl}
\overline{1} & = & \bot & & \overline{\bot} & = & 1 \\
\overline{\top} & = & 0 & & \overline{0} & = & \top \\
\overline{P \otimes Q} & = & \overline{P} \parr \overline{Q} & & \overline{P \parr Q} & = & \overline{P} \otimes \overline{Q} \\
\overline{P \,\&\, Q} & = & \overline{P} \oplus \overline{Q} & & \overline{P \oplus Q} & = & \overline{P} \,\&\, \overline{Q} \\
\overline{!P} & = & ?\overline{P} & & \overline{?P} & = & !\overline{P}
\end{array}
$$

$$P \multimap Q = \overline{P} \parr Q$$

Figura 3.1: Definizione di negazione e implicazione lineari

negazione si estende facilmente a tutti gli atomi definendo $\overline{\overline{p}} = p$ *per ogni simbolo proposizionale negato* $\overline{p}$*.*

Siano $1, \bot, 0, \top \notin \mathcal{A}$ *simboli costanti o* unità*, e sia* $a \in \mathcal{A}$*. Il* linguaggio CLL delle formule lineari classiche *è così definito:*

$$
\begin{array}{rcll}
T & ::= & 1 \mid \bot \mid 0 \mid \top \mid a & \textit{(termini)} \\
P & ::= & T \mid P \oplus P \mid P \,\&\, P \mid P \otimes P \mid P \parr P \mid !P \mid ?P & \textit{(formule)}
\end{array}
$$

I connettivi $\otimes$, $\parr$, $\multimap$, insieme agli elementi neutri 1 (relativamente a $\otimes$) e $\bot$ (relativamente a $\parr$) sono chiamati *moltiplicativi*; i connettivi $\&$, $\oplus$, insieme agli elementi neutri $\top$ (relativamente a $\&$) e 0 (relativamente a $\oplus$) sono chiamati *additivi*; i connettivi ! e ? sono chiamati *esponenziali*. Questa notazione è stata scelta perché facile da memorizzare: infatti essa suggerisce che $\otimes$ sia moltiplicativo e congiuntivo, con elemento neutro 1, mentre $\oplus$ è additivo e disgiuntivo, con elemento neutro 0; inoltre, anche la distributività di $\otimes$ su $\oplus$ è suggerita dalla notazione.

La negazione si estende alle formule, come mostrato in Figura 3.1; inoltre l'implicazione lineare è definita con l'ausilio di negazione e connettivo "par".

Procediamo mostrando in Figura 3.2 un sistema deduttivo per la logica lineare reminiscente il calcolo dei sequenti di Gentzen [1935]. Come visto in precedenza, un sequente è un'espressione $\Gamma \vdash \Delta$, in cui $\Gamma = P_1, \ldots, P_n$ e $\Delta = Q_1, \ldots, Q_m$ sono sequenze finite di formule. Il significato inteso di

$\Gamma \vdash \Delta$ è:

$$P_1 \text{ e } \ldots \text{ e } P_n \quad \text{implica} \quad Q_1 \text{ oppure } \ldots \text{ oppure } Q_m$$

dove il senso di "e", "implica" e "oppure" devono essere specificati formalmente. I sequenti lineari sono ad un lato, cioè della forma $\vdash \Gamma$; sequenti nella forma generale $\Gamma \vdash \Delta$ si possono "mimare" usando $\vdash \overline{\Gamma}, \Delta$.

Nel calcolo dei sequenti lineari abbiamo rimosso le regole strutturali di indebolimento (è sempre possibile aggiungere una formula nella premessa o nella conclusione del sequente) e contrazione (la molteplicità di una formula non conta) in virtù delle critiche mosse dalla scuola lineare. La possibilità di utilizzare queste operazioni è tuttavia ripristinata grazie all'introduzione degli operatori ! e ?.

Identità e taglio restano invariate rispetto al Sistema LKp, così come la regola di permutazione.

La situazione è diversa per quanto riguarda la congiunzione: come avveniva in precedenza, per dimostrare una congiunzione tra P e Q bisogna aver dimostrato separatamente sia P che Q, ma in assenza della regola d'indebolimento, possiamo distinguere il caso in cui le dimostrazioni di P e Q siano fatte nello stesso ambiente (Γ nella regola (&)), o in ambienti diversi (Γ e Δ in ($\otimes$)).

Un ragionamento analogo vale per la disgiunzione: in logica lineare possiamo infatti distinguere il caso in cui, nel dimostrare la disgiunzione di P e Q, disponiamo solo di P (regola ($\oplus_{\mathsf{l}}$)), solo di Q (regola ($\oplus_{\mathsf{r}}$)), e quello in cui abbiamo ambedue (regola ($\parr$)).

È possibile introdurre nuove formule, indebolendo il sequente, a patto che queste siano "marcate" con l'operatore !; per questa classe di formule (chiamate formule "perché non" o "why not") la molteplicità non è rilevante: inoltre ogni formula può essere trasformata in una *why not* grazie alla regola di *derelizione* (drlc). La regola di *promozione* (!) permette "aumentare" la molteplicità di una formula di una quantità arbitraria.

Il Sistema così ottenuto gode di buone proprietà, oltre ad avere una gra-

Identità / taglio

$$\vdash P, \overline{P} \quad (\mathsf{id}) \qquad \frac{\vdash \Gamma, P \qquad \vdash \overline{P}, \Delta}{\vdash \Gamma, \Delta}\,(\mathsf{cut})$$

Regole strutturali

$$\frac{\vdash \Gamma, P, Q, \Delta}{\vdash \Gamma, Q, P, \Delta}\,(\mathsf{perm})$$

Regole logiche

$$\vdash 1 \quad (\mathsf{one}) \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \bot}\,(\mathsf{false})$$

$$\frac{\vdash \Gamma, P \qquad \vdash \Delta, Q}{\vdash \Gamma, \Delta, P \otimes Q}\,(\otimes) \qquad \frac{\vdash \Gamma, P, Q}{\vdash \Gamma, P \parr Q}\,(\parr)$$

$$\vdash \Gamma, \top \quad (\mathsf{true}) \qquad \textit{(nessuna regola per } \mathsf{0}\textit{)}$$

$$\frac{\vdash \Gamma, P \qquad \vdash \Gamma, Q}{\vdash \Gamma, P \,\&\, Q}\,(\&) \qquad \frac{\vdash \Gamma, P}{\vdash \Gamma, P \oplus Q}\,(\oplus_{\mathsf{l}}) \quad \frac{\vdash \Gamma, Q}{\vdash \Gamma, P \oplus Q}\,(\oplus_{\mathsf{r}})$$

$$\frac{\vdash ?\Gamma, P}{\vdash ?\Gamma, !P}\,(!) \qquad \frac{\vdash \Gamma}{\vdash \Gamma, ?P}\,(\mathsf{weak})$$

$$\frac{\vdash \Gamma, P}{\vdash \Gamma, ?P}\,(\mathsf{drlc}) \qquad \frac{\vdash \Gamma, ?P, ?P}{\vdash \Gamma, ?P}\,(\mathsf{cntr})$$

Figura 3.2: Sistema deduttivo per CLL

nularità più fine rispetto alla logica classica. Per CLL è possibile dimostrare la *cut elimination*:

Teorema 3.1.2 (Hauptsatz lineare)**.** *La regola di taglio lineare è eliminabile da CLL.*

Dimostrazione. La dimostrazione segue un argomento del tutto analogo a

quello visto per la logica classica nel Teorema 2.1.6, con alcune semplificazioni dovute al fatto di non dover trattare le usuali regole strutturali. □

Nuovamente, la dimostrazione risultante dalla procedura di cut elimination non è univocamente determinata, a causa della *permutazione delle regole*. Ad esempio, nella derivazione:

$$\dfrac{\dfrac{\vdash \Gamma, P}{\vdash \Gamma', P}\,(\rho) \qquad \dfrac{\vdash \overline{P}, \Delta}{\vdash \overline{P}, \Delta'}\,(\sigma)}{\vdash \Gamma', \Delta'}\,(\mathsf{cut})$$

non c'è nessun modo ovvio di eliminare l'applicazione di (cut), poiché le regole (ρ) e (σ) non agiscono su P e $\overline{P}$. Quindi l'idea è di "spingere il cut verso l'alto":

$$\dfrac{\vdash \Gamma, P \qquad \vdash \overline{P}, \Delta}{\dfrac{\dfrac{\vdash \Gamma, \Delta}{\vdash \Gamma', \Delta}\,(\rho)}{\vdash \Gamma', \Delta'}\,(\sigma)}\,(\mathsf{cut})$$

ma così facendo abbiamo arbitrariamente privilegiato la regola (ρ) rispetto alla (σ), mentre l'altra scelta:

$$\dfrac{\vdash \Gamma, P \qquad \vdash \overline{P}, \Delta}{\dfrac{\dfrac{\vdash \Gamma, \Delta}{\vdash \Gamma, \Delta'}\,(\sigma)}{\vdash \Gamma', \Delta'}\,(\rho)}\,(\mathsf{cut})$$

sarebbe stata altrettanto legittima. La scelta compiuta in questo passo della cut elimination è in generale irreversibile: a meno che (ρ) o (σ) non siano successivamente eliminate, non sarà più possibili scambiarle. Per eliminare questa fonte di non-determinismo, fu introdotto in Girard [1987] un nuovo formalismo, basato sulla teoria dei grafi, e chiamato *Proof Nets*.

Il Sistema CLL non è l'unico rappresentante della classe delle logiche lineari. Vista l'ampia gamma di regole che possiede, questo Sistema può essere suddiviso in moduli con interessanti proprietà computazionali: il punto è proprio che la logica lineare è in grado di trattare naturalmente con le risorse (rappresentate dalla *molteplicità* delle formule), e per questo ci si riferisce ad

Grammatica di MLL+mix

$$P ::= a \mid \overline{P} \mid P \otimes P \mid P \parr P$$

(con $a \in \mathcal{A}$ infinità numerabile di simboli proposizionali)

Sistema deduttivo

$$\vdash P, \overline{P}\ (\mathsf{id}) \qquad \frac{\vdash \Gamma, P \qquad \vdash Q, \Delta}{\vdash \Gamma, \Delta, P \otimes Q}\ (\otimes) \qquad \frac{\vdash \Gamma, P, Q}{\vdash \Gamma, P \parr Q}\ (\parr) \qquad \frac{\vdash \Gamma \qquad \vdash \Delta}{\vdash \Gamma, \Delta}\ (\mathsf{mix})$$

Figura 3.3: Sistema MLL+mix

essa con l'appellativo *resource-conscious*; in informatica avere coscienza delle risorse significa saper distinguere varie classi di complessità.

Tra i vari sottosistemi, quelli che maggiormente divergono dalla logica classica (e intuizionista), sono chiamati LLL (Light Linear Logic) ed ELL (Elementary Linear Logic) – Girard [1995a]; Danos and Joinet [2001]. Essi seguono dalla scoperta che, in assenza degli esponenziali, la procedura di eliminazione dei tagli può essere eseguita in tempo lineare.

Per i nostri scopi, ci occuperemo esclusivamente del frammento moltiplicativo: questo è il più semplice ed il più piccolo frammento di logica lineare (fu anche il primo che venne trasposto nelle Proof Nets, per via della sua semplicità). Nella fattispecie tratteremo d'ora in avanti il Sistema in Figura 3.3, chiamato MLL+mix, cioè *Multiplicative Linear Logic* con l'aggiunta della regola mix che "fonde" i sequenti provenienti da due diversi sottoalberi di derivazione. La negazione è definita dalle leggi di De Morgan:

$$\begin{aligned} \overline{P \otimes Q} &= \overline{P} \parr \overline{Q} \\ \overline{P \parr Q} &= \overline{P} \otimes \overline{Q} \end{aligned}$$

Avremo modo di osservare una natuale corrispondenza di questo Sistema ed il suo corrispettivo in deep inference: il Sistema LBV di Guglielmi [2002].

3.2 Sistema LBV

È il più semplice Sistema deep inference concepibile: un calcolo proposizionale composto da *due operatori duali* (del tutto simili a congiunzione e disgiunzione classici), una *negazione auto-duale* alla De Morgan e una *unità logica*. Come nel caso classico, le formule sono considerate uguali modulo una relazione di equivalenza. Le regole sono l'assioma (ax) per l'unità e la regola di scambio (s); inoltre la regola d'identità (chiamata anche *regola d'interazione*) e quella di taglio (o *regola di co-interazione*), nella versione generalizzata:

$$\frac{\mathbb{C}\{\circ\}}{\mathbb{C}[P,\overline{P}]}\ (\mathsf{i}\downarrow) \qquad\qquad \frac{\mathbb{C}(P,\overline{P})}{\mathbb{C}\{\circ\}}\ (\mathsf{i}\uparrow)$$

che tuttavia, come prima, possono essere ridotte alla loro forma atomica, dando origine al Sistema di Figura 3.4.

Teorema 3.2.1 (Località di LBV+cut)**.** *La regola* (i↓) *è derivabile da* $\{(\mathsf{ai}\downarrow),(\mathsf{s})\}$. *Dualmente, la regola* (i↑) *è derivabile da* $\{(\mathsf{ai}\uparrow),(\mathsf{s})\}$.

Dimostrazione. Data l'istanza $\dfrac{\mathbb{C}\{\circ\}}{\mathbb{C}[P,\overline{P}]}\ (\mathsf{i}\downarrow)$ procediamo per induzione strutturale su P. Il caso duale (i↑) si dimostra allo stesso modo.

Casi base

1. $P = \circ$. Ovvio, poiché $\mathbb{C}[P,\overline{P}] = \mathbb{C}\{\circ\}$.
2. P è un atomo: Allora (i↓) è un'istanza di (ai↓).

Casi induttivi

3. $P = [R,S]$. Per ipotesi induttiva, abbiamo due derivazioni Φ_R e Φ_S:

$$\begin{array}{cc} \mathbb{C}\{\circ\} & \mathbb{D}\{\circ\} \\ \Phi_R \Big\| \{(\mathsf{ai}\downarrow),(\mathsf{s})\} & \Phi_S \Big\| \{(\mathsf{ai}\downarrow),(\mathsf{s})\} \\ \mathbb{C}[R,\overline{R}] & \mathbb{D}[S,\overline{S}] \end{array}$$

$$P ::= \circ \mid a \mid \overline{a} \mid [P, P] \mid (P, P) \qquad \text{(con } a \in \mathcal{A} \text{ infinità numerabile di simboli proposizionali)}$$

$$\circ \quad (\mathsf{ax}) \qquad \frac{\mathbb{C}\{\circ\}}{\mathbb{C}[a, \overline{a}]}\,(\mathsf{ai}{\downarrow}) \qquad \frac{\mathbb{C}(a, \overline{a})}{\mathbb{C}\{\circ\}}\,(\mathsf{ai}{\uparrow}) \qquad \frac{\mathbb{C}(P, [Q, R])}{\mathbb{C}[(P, Q), R]}\,(\mathsf{s})$$

Figura 3.4: Sistema LBV+cut

da cui è possibile ottenere:

$$\begin{array}{c} \mathbb{C}\{\circ\} \\ \Vert_{\Phi_R}\,\{(\mathsf{ai}{\downarrow}),(\mathsf{s})\} \\ \mathbb{C}[R, \overline{R}] = \mathbb{C}([R, \overline{R}], \circ) \\ \Vert_{\Phi_S}\,\{(\mathsf{ai}{\downarrow}),(\mathsf{s})\} \\ \dfrac{\mathbb{C}([R, \overline{R}], [S, \overline{S}])}{\mathbb{C}[R, (\overline{R}, [S, \overline{S}])]}\,(\mathsf{s}) \\ \dfrac{\mathbb{C}[R, (\overline{R}, [S, \overline{S}])]}{\mathbb{C}[R, S, (\overline{R}, \overline{S})]}\,(\mathsf{s}) \end{array}$$

4. Infine, il caso $P = (R, S)$ è analogo al precedente.

□

Esiste una corrispondenza 1:1 tra Sistema MLL+mix e Sistema LBV.

Definizione 3.2.2 (Trasformazioni LBV↔MLL)**.**

$$\begin{array}{rcl|rcl} \underline{\cdot} : \mathscr{L}_{\mathsf{MLL}} \to \mathscr{L}_{\mathsf{LBV}} & & & \underline{\underline{\cdot}} : \mathscr{L}_{\mathsf{LBV}} \to \mathscr{L}_{\mathsf{MLL}} & & \\ \underline{a} & = & a & \underline{\underline{a}} & = & a \\ \underline{P \parr Q} & = & [\underline{P}, \underline{Q}] & \underline{\underline{[P, Q]}} & = & \underline{\underline{P}} \parr \underline{\underline{Q}} \\ \underline{P \otimes Q} & = & (\underline{P}, \underline{Q}) & \underline{\underline{(P, Q)}} & = & \underline{\underline{P}} \otimes \underline{\underline{Q}} \end{array}$$

Inoltre la definizione di $\underline{\cdot}$ *si estende facilmente ai sequenti:*

$$\underline{\vdash P_1, \ldots, P_n} = [\underline{P_1}, \ldots, \underline{P_n}]$$

per $n > 0$*; per* $n = 0$ *si pone* $\underline{\vdash} = \circ$*.*

Teorema 3.2.3 (Equivalenza di LBV e MLL+mix)**.**

i) Se il sequente $\vdash P$ *è dimostrabile in MLL+mix, allora la struttura* $\underline{P}$ *è dimostrabile in LBV.*

ii) Se la struttura P *(in forma normale, con* $P \neq \circ$*) è dimostrabile in LBV, allora il sequente* $\vdash \underline{\underline{P}}$ *è dimostrabile in MLL+mix.*

Questo teorema (dimostrato per la prima volta in Guglielmi [2002]) stabilisce una correlazione tra calcolo dei sequenti e calcolo delle strutture; come già accennato, è possibile conseguire un risultato analogo per il Sistema SKS, ma, ad esempio, la proprietà di località non vale per la logica classica proposizionale nel calcolo dei sequenti.

3.2.1 Eliminazione del taglio

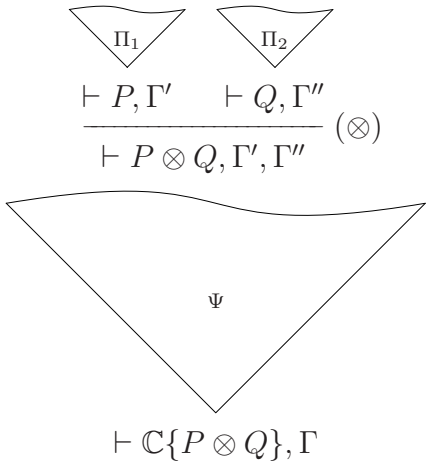
L'argomento calssico per dimostrare l'eliminazione del taglio nel calcolo dei sequenti, risiede nel fatto che, quando le formule principali del taglio sono introdotte in entrambi i rami, esse determinano che regole saranno applicate immediatamente sopra a quella di taglio. Questo è conseguenza del fatto che le formule hanno un connettivo principale, e le regole logiche si basano solo su quello, e su nessun'altra proprietà delle formule.

Questo fatto non vale nel calcolo delle strutture. Per dimostrare la cut elimination nel Sistema LBV, occorre appoggiarsi ad un'altra proprietà, scoperta in Guglielmi [2002], e chiamata *scissione* o *splitting*. Essa è una generalizzazione della tecnica vista nella dimostrazione di eliminazione del taglio per il sistema SKS. Si consideri la dimostrazione del sequente:

$$\vdash \mathbb{C}\{P \otimes Q\}, \Gamma$$

dove $\mathbb{C}\{P \otimes Q\}$ è una formula contenente la sottoformula $P \otimes Q$. Sappiamo per certo che nella dimostrazione ci deve essere un'istanza della regola $(\otimes)$ che

scinde P da Q assieme ai rispettivi contesti. Siamo nella seguente situazione:



corrispondente a

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\Pi_2\Vdash\\ [Q,\Gamma'']\end{matrix}\quad \begin{matrix}\Pi_1\Vdash\\ ([P,\Gamma'],[Q,\Gamma''])\end{matrix}}{[([P,\Gamma'],Q),\Gamma'']}\,(\mathsf{s})}{[(P,Q),\Gamma',\Gamma'']}\,(\mathsf{s})}{\begin{matrix}\Psi\Vdash\\ [\mathbb{C}(P,Q),\Gamma]\end{matrix}}$$

La derivazione Ψ implementa lo splitting, che è ottenuto in due passi:

1. riduzione del contesto: se $\mathbb{C}\{P\}$ è dimostrabile, allora $\mathbb{C}$ può essere ridotto, risalendo nella dimostrazione, ad un contesto $[\bullet, S]$, per un S opportuno, tale che $[P, S]$ è dimostrabile. Nell'esempio sopra, $[\mathbb{C}\{\bullet\}, \Gamma]$ è ridotto a $[\bullet, S]$ per un certo S;

2. scissione di superficie: se $[(R, T), P]$ è dimostrabile, allora P può essere ridotto, risalendo nella dimostrazione, ad una struttura $[P_1, P_2]$ tali che $[R, P_1]$ e $[T, P_2]$ sono dimostrabili. Nell'esempio, S è scisso in $[\Gamma', \Gamma'']$.

Grazie al Teorema di splitting, abbiamo la capacità di scindere un copar in due dimostrazioni, una per ogni rispettiva sottoformula: l'importanza di tale capacità ai fini della cut elimination, diventa chiara se consideriamo la regola di taglio nel Sistema LBV:

$$\dfrac{\mathbb{C}(a, \overline{a})}{\mathbb{C}\{\circ\}}\,(\mathsf{ai}{\uparrow})$$

Il contesto $\mathbb{C}$ viene scisso in due componenti S_1 e S_2 tali che esistono le dimostrazioni $\begin{matrix}\Pi_1\Vdash\\ [a, S_1]\end{matrix}$ e $\begin{matrix}\Pi_2\Vdash\\ [\overline{a}, S_2]\end{matrix}$. Ora possiamo sfruttare il fatto che gli atomi

a e $\overline{a}$ possono essere introdotti, rispettivamente nelle conclusioni di Π_1 e Π_2, solo mediante applicazioni della regola (ai↓) (fatto non vero, ad esempio, nel Sistema KS). A questo punto siamo in grado di isolare il segmento di dimostrazione che introduce gli atomi che verranno in seguito rimossi dal taglio, e possiamo pertanto trasformare questa sezione per bloccare il flusso degli atomi diretti al taglio sul nascere.

Teorema 3.2.4 (Shallow splitting)**.** *Se* $[(R,T),P]$ *è dimostrabile in* LBV*, allora esistono* P_1 *e* P_1 *tali che:*

$$\begin{array}{c}[P_1,P_2]\\ \Vert_{\mathsf{LBV}}\\ P\end{array}$$

e $[R,P_1]$ *e* $[T,P_2]$ *siano entrambi dimostrabili in* LBV*.*

Dimostrazione. Consideriamo l'ordinamento lessicografico sui naturali:

$$(m',n') \prec (m,n) \quad \text{sse} \quad m' < m \text{ oppure } (m' = m \text{ e } n' < n)$$

Vogliamo procedere per induzione completa su due quantità: l'altezza della dimostrazione di $[(R,T),P]$ e la *lunghezza delle formule*, definita induttivamente da:

$$\begin{array}{lcllcl} |\circ| & = & 0 & |[P,Q]| & = & |P|+|Q| \\ |a| & = & 1 & |(P,Q)| & = & |P|+|Q| \\ |\overline{P}| & = & |P| & & & \end{array}$$

Dato il meta-enunciato:

$$\begin{array}{rcl} C(m,n) & = & \forall R,T,P. \\ & & \quad \forall (m',n') \preceq (m,n). \\ & & \quad \left(m' = \left| [(R,T),P] \right| \quad \wedge \quad \text{esiste } \begin{array}{c}\Vdots\\ [(R,T),P]\end{array} \text{ con altezza } n' \right) \\ & & \quad \Rightarrow \quad \exists P_1,P_2. \left(\begin{array}{ccccc} [P_1,P_2] & & \Vdots & & \Vdots \\ \Vert & \wedge & [R,P_1] & \wedge & [T,P_2] \\ P & & & & \end{array} \right) \end{array}$$

il teorema è equivalente a $\forall m, n.C(m, n)$. Per ipotesi induttiva possiamo suppore di avere una dimostrazione di $C(m', n')$ per ogni $(m', n') \prec (m, n)$.

La lunghezza di $[(R, T), P]$ è m e l'altezza della sua dimostrazione è n. Consideriamo l'istanza dell'ultima regola di questa dimostrazione:

$$\dfrac{\begin{array}{c}\Vdots\\ Q\end{array}}{[(R, T), P]}\,(\rho)$$

Procediamo per casi su (ρ) (assumiamo sempre $P \neq \circ$ e $R \neq T \neq \circ$, perché in questi casi il teorema vale banalmente):

1. $(\rho) = (\mathsf{ai}{\downarrow})$. Questa regola si può applicare in tre diversi modi:

   1.1. all'interno di R, cioè:

   $$\dfrac{\begin{array}{c}\Vdots\\ [(R', T), P]\end{array}}{[(R, T), P]}\,(\mathsf{ai}{\downarrow})$$

   Per ipotesi induttiva esistono P_1, P_2 tali che:

   $$\begin{array}{c}[P_1, P_2]\\ \Vdots\\ P\end{array} \quad , \quad \begin{array}{c}\Vdots\\ [R', P_1]\end{array} \quad , \quad \begin{array}{c}\Vdots\\ [T, P_2]\end{array}$$

   È sufficiente applicare $(\mathsf{ai}{\downarrow})$ in coda alla dimostrazione di $[R', P_1]$ per ottenere:

   $$\dfrac{\begin{array}{c}\Vdots\\ [R', P_1]\end{array}}{[R, P_1]}\,(\mathsf{ai}{\downarrow})$$

   1.2. all'interno di T. Analogo al caso precedente.

   1.3. all'interno di P, cioè:

   $$\dfrac{\begin{array}{c}\Vdots\\ [(R, T), P']\end{array}}{[(R, T), P]}\,(\mathsf{ai}{\downarrow})$$

Anche in questo caso procediamo per induzione diretta, a meno di un'applicazione di $(\mathsf{ai}{\downarrow})$ in:

$$\begin{array}{c}[P_1, P_2] \\ \Vert \text{ (per I.H.)} \\ P' \\ \hline P \end{array} (\mathsf{ai}{\downarrow})$$

2. $(\rho) = (\mathsf{s})$. Se la regola si applica all'interno di R, T o P, procediamo in maniera del tutto analoga a quanto fatto nel caso $(\rho) = (\mathsf{ai}{\downarrow})$. Esistono altre due possibilità:

   2.1. $R = (R', R'')$, $T = (T', T'')$, $P = [P', P'']$ e:

$$\frac{\begin{array}{c}\Vert \\ [([(R', T'), P'], R'', T''), P'']\end{array}}{[(R', R'', T', T''), P', P'']} (\mathsf{s})$$

   Possiamo applicare l'ipotesi induttiva per ottenere:

$$\begin{array}{c}[P_1', P_2'] \\ \Vert \\ P''\end{array} \quad , \quad \begin{array}{c}\Pi_1 \Vert \\ [(R', T'), P', P_1']\end{array} \quad , \quad \begin{array}{c}\Pi_2 \Vert \\ [(R'', T''), P_2']\end{array}$$

   Ora possiamo applicare nuovamente l'ipotesi induttiva su Π_1 e su Π_2: infatti, anche se non conosciamo l'altezza di queste due dimostrazioni, sappiamo che la loro dimensione è inferiore a:

$$\left| [(R', R'', T', T''), P', P''] \right|$$

   perché per ipotesi, l'istanza di (s) non è triviale. Pertanto abbiamo

per ipotesi induttiva:

$$\begin{array}{c}[P_1'', P_2''] \\ \Vert \\ [P', P_1']\end{array} \quad , \quad \begin{array}{c}\Vert \\ [R', P_1'']\end{array} \quad , \quad \begin{array}{c}\Vert \\ [T', P_2'']\end{array}$$

$$\begin{array}{c}[P_1''', P_2'''] \\ \Vert \\ P_2'\end{array} \quad , \quad \begin{array}{c}\Vert \\ [R'', P_1''']\end{array} \quad , \quad \begin{array}{c}\Vert \\ [T'', P_2''']\end{array}$$

Ora, ponendo $P_1 = [P_1'', P_1''']$ e $P_2 = [P_2'', P_2''']$ otteniamo:

$$\begin{array}{c}[P_1'', P_2'', P_1''', P_2'''] \\ \Vert \\ [P_1'', P_2'', P_2'] \\ \Vert \\ [P', P_1', P_2'] \\ \Vert \\ [P', P'']\end{array} \quad , \quad \dfrac{\begin{array}{c}\Vert \\ [R'', P_1'''] \\ \Vert \\ [([R', P_1''], R''), P_1''']\end{array}}{[(R', R''), P_1'', P_1''']}\,(\mathsf{s}) \quad , \quad \dfrac{\begin{array}{c}\Vert \\ [T'', P_2'''] \\ \Vert \\ [([T', P_2''], T''), P_2''']\end{array}}{[(T', T''), P_2'', P_2''']}\,(\mathsf{s})$$

2.2. $P = [(P', P''), U', U'']$ e:

$$\dfrac{\begin{array}{c}\Vert \\ [([(R, T), P', U'], P''), U'']\end{array}}{[(R, T), (P', P''), U', U'']}\,(\mathsf{s})$$

Per ipotesi induttiva abbiamo:

$$\begin{array}{c}[U_1, U_2] \\ \Vert \\ U''\end{array} \quad , \quad \begin{array}{c}\Vert_{\Pi} \\ [(R, T), P', U', U_1]\end{array} \quad , \quad \begin{array}{c}\Vert \\ [P'', U_2]\end{array}$$

È di nuovo è possibile applicare l'ipotesi induttiva su Π poiché:

$$\Big|\, [(R, T), P', U', U_1] \,\Big| < [([(R, T), P', U'], P''), U'']$$

per ottenere:

$$\begin{array}{c}[P_1, P_2] \\ \Vert \\ [P', U', U_1]\end{array} \quad , \quad \begin{array}{c}\Vert \\ [R, P_1]\end{array} \quad , \quad \begin{array}{c}\Vert \\ [T, P_2]\end{array}$$

Ora possiamo costruire:

$$\dfrac{\begin{array}{c}[P_1, P_2]\\ \Vdash\\ [P', U', U_1]\\ \Vdash\\ [(P', [P'', U_2]), U', U_1]\end{array}}{\begin{array}{c}[(P', P''), U', U_1, U_2]\\ \Vdash\\ [(P', P''), U', U'']\end{array}}\,(\mathsf{s})$$

□

Teorema 3.2.5 (Riduzione del contesto)**.** *Per ogni struttura P ed ogni contesto $\mathbb{C}$ tale che $\mathbb{C}\{P\}$ è dimostrabile in* LBV*, esiste una struttura C tale che, per ogni struttura X esistono le derivazioni:*

$$\begin{array}{c}[C, X]\\ \Vdash_{\mathsf{LBV}}\\ \mathbb{C}\{X\}\end{array} \qquad e \qquad \begin{array}{c}\\ {}_{\mathsf{LBV}}\Vdash\\ [C, P]\end{array}$$

Dimostrazione. Per induzione sulla dimensione di $\mathbb{C}\{\bullet\}$. Il caso base è triviale, $C = \circ$. I casi induttivi sono:

1. $\mathbb{C}\{\bullet\} = (\mathbb{C}'\{\bullet\}, Q)$, per qualche $Q \neq \circ$. Se $\mathbb{C}\{P\}$ è dimostrabile, allora devono esistere le dimostrazioni di $\mathbb{C}'\{P\}$ e di Q. Applicando l'ipotesi induttiva su $\mathbb{C}'\{P\}$, otteniamo C tale che, per ogni X:

$$\begin{array}{c}[C, X]\\ \Vdash\\ \mathbb{C}'\{X\}\\ \Vdash\\ (\mathbb{C}'\{X\}, Q)\end{array}$$

   e tale che $[C, P]$ è dimostrabile in LBV. Lo stesso argomento si applica quando $\mathbb{C}\{\bullet\} = (Q, \mathbb{C}'\{\bullet\})$ con $Q \neq \circ$.

2. $\mathbb{C}\{\bullet\} = [\mathbb{C}'\{\bullet\}, Q]$, per qualche $Q \neq \circ$. Assumiamo che $\mathbb{C}'\{\bullet\}$ non sia un par: questa ipotesi non è limitativa, perché è sempre possibile far "rientrare" il parallelo in Q, lasciando $\mathbb{C}'$ come copar. Se alla fine di questo processo otteniamo $\mathbb{C}'\{\bullet\} = \bullet$, il teorema è banalmente provato. Quindi $\mathbb{C}'\{\bullet\} = (\mathbb{C}''\{\bullet\}, Q')$ con $Q' \neq \circ$. Per il Teorema 3.2.4, esistono:

$$\begin{array}{c}[Q_1, Q_2] \\ \Vert_{\mathsf{LBV}} \\ Q\end{array} \quad , \quad \begin{array}{c}\amalg\Vert_{\mathsf{LBV}} \\ [\mathbb{C}''\{P\}, Q_1]\end{array} \quad , \quad \begin{array}{c}\Vert_{\mathsf{LBV}} \\ [Q', Q_2]\end{array}$$

Ora, applicando l'ipotesi induttiva su Π, otteniamo:

$$\begin{array}{c}[C, X] \\ \Vert \\ [\mathbb{C}''\{X\}, Q_1] \\ \Vert \\ \dfrac{[([Q', Q_2], \mathbb{C}''\{X\}), Q_1]}{[(\mathbb{C}''\{X\}, Q'), Q_1, Q_2]}\,(\mathsf{s}) \\ \Vert \\ [(\mathbb{C}''\{X\}, Q'), Q] = (\mathbb{C}'\{X\}, Q) = \mathbb{C}\{X\}\end{array} \qquad \text{e} \qquad \begin{array}{c}\Vert \\ [C, P]\end{array}$$

Analogamente si dimostra $\mathbb{C}'\{\bullet\} = (Q', \mathbb{C}''\{\bullet\})$ e si usa lo stesso argomento per $\mathbb{C}\{\bullet\} = [Q, \mathbb{C}'\{\bullet\}]$.

□

Corollario 3.2.6 (Splitting)**.** *Per ogni struttura P e Q e contesto $\mathbb{C}$, se $\mathbb{C}(P, Q)$ è dimostrabile in* LBV*, allora esistono due strutture S_1 e S_2 tali che, per ogni struttura X, esistono le derivazioni:*

$$\begin{array}{c}[X, S_1, S_2] \\ \Vert_{\mathsf{LBV}} \\ \mathbb{C}\{X\}\end{array} \quad , \quad \begin{array}{c}\Vert_{\mathsf{LBV}} \\ [P, S_1]\end{array} \quad , \quad \begin{array}{c}\Vert_{\mathsf{LBV}} \\ [Q, S_2]\end{array}$$

Dimostrazione. Prima si applica il Teorema 3.2.4, poi il Teorema 3.2.5. □

Infine, prima di passare alla cut elimination, occorre enunciare un ultimo semplice risultato.

Proposizione 3.2.7. *Per ogni struttura P, Q e contesto $\mathbb{C}$, esiste una derivazione:*

$$\begin{array}{c} \mathbb{C}[P,Q] \\ \Big\| \{(\mathsf{s})\} \\ [\mathbb{C}\{P\}, Q] \end{array}$$

Dimostrazione. Per induzione sulla dimensione del contesto $\mathbb{C}$. Questa dimostrazione è uguale a quella della Proposizione 2.2.6, che stabilisce la stessa proposizione per il Sistema KS. □

Teorema 3.2.8 (Cut elimination)**.** *La regola* (ai↑) *è ammissibile in LBV.*

Dimostrazione. Consideriamo la dimostrazione:

$$\dfrac{\begin{array}{c} \Vdots \\ \mathbb{C}(a, \overline{a}) \end{array}}{\mathbb{C}\{\circ\}} (\mathsf{ai}{\uparrow})$$

Per il Corollario 3.2.6, esistono S_1 e S_2 tali che esistono le derivazioni:

$$\begin{array}{c} [S_1, S_2] \\ \Big\| \\ \mathbb{C}\{\circ\} \end{array} \quad , \quad \begin{array}{c} \Pi_1 \Vdots \\ [a, S_1] \end{array} \quad , \quad \begin{array}{c} \Pi_2 \Vdots \\ [\overline{a}, S_2] \end{array}$$

Vogliamo individuare, nella dimostrazione Π_1, il punto in cui l'atomo a viene introdotto. Certamente deve esistere un contesto $\mathbb{C}'$ tale che $S_1 = \mathbb{C}'\{\overline{a}\}$. Inoltre, deve esistere un contesto $\mathbb{C}''$ tale che:

$$\dfrac{\begin{array}{c} \Pi_1'' \Vdots \\ \mathbb{C}''\{\circ\} \end{array}}{\mathbb{C}''[a, \overline{a}]} (\mathsf{ai}{\downarrow})$$
$$\begin{array}{c} \Pi_1' \Big\| \\ [a, \mathbb{C}'\{\overline{a}\}] \end{array}$$

sia la dimostrazione Π_1 in cui abbiamo individuato l'applicazione della regola $(\mathsf{ai}{\downarrow})$. Ora, sostituendo in Π_1' le occorrenze di a e $\overline{a}$ con $\circ$, otteniamo una dimostrazione Ψ_1', grazie alla quale è possibile dimostrare:

$$\begin{array}{c} \Pi_1'' \Big\Vert \\ \mathbb{C}''\{\circ\} \\ \Psi_1' \Big\Vert \\ \mathbb{C}'\{\circ\} \end{array}$$

Analogamente possiamo trasformare la dimostrazione di Π_2 in una dimostrazione di $\mathbb{D}'\{\circ\}$ dove $S_2 = \mathbb{D}'\{a\}$. Ora possiamo concludere, esibendo la seguente dimostrazione:

$$\begin{array}{c} \Vert \\ \mathbb{C}'\{\circ\} \\ \Vert \\ \dfrac{\mathbb{C}'\{\mathbb{D}'\{\circ\}\}}{\mathbb{C}'\{\mathbb{D}'[a,\overline{a}]\}}\,(\mathsf{ai}{\downarrow}) \\ \Phi \Big\Vert \\ [\mathbb{C}'\{\overline{a}\}, \mathbb{D}'\{a\}] \\ \Vert \\ \mathbb{C}\{\circ\} \end{array}$$

in cui Φ è ottenuta applicando due volte la Proposizione 3.2.7.

Possiamo ripetere induttivamente l'argomento per ogni dimostrazione di $\mathsf{LBV} \cup \{(\mathsf{ai}{\uparrow})\}$, partendo dall'alto, ed eliminare una per una tutte le istanze di $(\mathsf{ai}{\uparrow})$. □

Conclusioni

La deep inference offre una prospettiva nuova e moderna in teoria della dimostrazione. Grazie a questa metodologia, il lavoro strutturale svolto dagli alberi in shallow inference, viene collassato nell'uso dei contesti, che sono un concetto fondamentale in questo approccio. A questo proposito, è interessante osservare come questo metta in relazione diretta il modo di operare tipico in proof theory (con alberi di derivazione) con il mondo dei sistemi di riscrittura. Infatti le derivazione nel calcolo delle strutture possono essere linearizzate, leggendole dal basso in alto (verso della proof search), e le regole d'inferenza si possono vedere come regole di riscrittura; quali corrispondenze si possono trovare in questo senso? A quali sistemi di riscrittura corrispondono i sistemi in calcolo delle strutture, e di quali proprietà godono?

Inoltre, osserviamo come, nelle procedure di cut elimination per il calcolo delle strutture, l'attenzione sia posta sugli atomi da eliminare, a conseguenza del fatto che questi sistemi godono di località. Una sotto-procedura invariante nella cut elimination è la discesa nella dimostrazione alla ricerca del taglio, per poi risalire seguendo il flusso degli atomi coinvolti. Da questa osservazione nasce un nuovo filone di ricerca in deep inference che tratta i cosiddetti "flussi atomici" o "atomic flows"; per una introduzione all'argomento, vedere Gundersen [2009].

Infine, esistono molti problemi rilevanti riguardanti la complessità delle dimostrazioni, alcuni dei quali ancora aperti, altri già risolti, ad esempio in Jeřábek [2009]; Bruscoli and Guglielmi [2009]; Bruscoli et al. [2009]. Il calcolo delle strutture è un formalismo molto espressivo, ma difficile da trat-

tare a causa del forte non-determinismo che comporta la fine grana (i.e. la vasta applicabilità) delle sue regole; Kahramanoğulları [2006] ha ideato una tecnica capace di ridurre questo non-determinismo.

Tutte le fonti e le informazioni riguardanti le ricerche in deep inference, sono reperibili online sulla pagina di Alessio Guglielmi, uno dei maggiori promotori di questo approccio, all'indirizzo:

http://alessio.guglielmi.name/res/cos/

Bibliografia

Abramsky, S., Gabbay, D. M., and Maibaum, T., editors 1992. *Handbook of Logic in Computer Science*. Oxford University Press, Oxford.

Aho, A. V., Lam, M. S., Sethi, R., and Ullman, J. D. 2006. *Compilers: Principles, Techniques, and Tools (2nd Edition)*. Addison Wesley, 2 edition.

Aho, A. V. and Ullman, J. D. 1972. *The theory of parsing, translation, and compiling*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Backus, J. W., Bauer, F. L., Green, J., Katz, C., McCarthy, J., Perlis, A. J., Rutishauser, H., Samelson, K., Vauquois, B., Wegstein, J. H., van Wijngaarden, A., and Woodger, M. 1960. Report on the algorithmic language algol 60. *Commun. ACM*, 3(5):299–314.

Barwise, J. 1977. *Handbook of Mathematical Logic*. North-Holland, Amsterdam.

Beaney, M., editor 1997. *The Frege Reader*. Blackwell, London.

Brünnler, K. 2004. *Deep Inference and Symmetry in Classical Proofs*. Logos Verlag, Berlin. http://www.iam.unibe.ch/~kai/Papers/phd.pdf.

Brünnler, K. and Tiu, A. F. 2001. A local system for classical logic. In *Lecture Notes in Artificial Intelligence*, pages 347–361. Springer-Verlag.

Bruscoli, P. and Guglielmi, A. 2009. On the proof complexity of deep inference. *ACM Transactions on Computational Logic*, 10(2):1–34. Article 14. http://cs.bath.ac.uk/ag/p/PrComplDI.pdf.

Bruscoli, P., Guglielmi, A., Gundersen, T., and Parigot, M. 2009. Quasipolynomial normalisation in deep inference via atomic flows and threshold formulae. http://cs.bath.ac.uk/ag/p/QuasiPolNormDI.pdf.

Chang, Chen, C., and Keisler, H. J. 1973. *Model theory*. North-Holland Pub. Co.; American Elsevier, Amsterdam, New York.

Dal Lago, U. and Baillot, P. 2006. Light affine logic, uniform encodings and polynomial time. *Mathematical Structures in Computer Science*, 16(4):713–733.

Danos, V. and Joinet, J.-b. 2001. Linear logic & elementary time. *Information and Computation*, 183.

Frege, G. 1879. *Begriffsschrift: eine der arithmetische nachgebildete Formelsprache des reinen Denkens*. L. Nebert, Halle a/S. Translated, as *Begriffsschrift*: A Formula Language for Pure Thought Modelled on that of Arithmetic, by Michael Beaney, in Beaney 1997.

Gentzen, G. 1935. Untersuchungen über das logische schließen ii. *Mathematische Zeitschrift*, 39.

Girard, J.-Y. 1987. Linear logic. *Theoretical Computer Science*, 50:1–102.

Girard, J.-Y. 1995a. Light linear logic.

Girard, J.-Y. 1995b. Linear logic: its syntax and semantics. In *Advances in Linear Logic*, pages 1–42. Cambridge University Press.

Girard, J.-Y. 1998. Light linear logic. *Inf. Comput.*, 143(2):175–204.

Girard, J.-Y., Lafont, Y., and Taylor, P. 1989. *Proofs and Types*. Cambridge University Press.

Gödel, K. 1931. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik*, 38(1):173–198.

Guglielmi, A. 2002. A system of interaction and structure. Technical report.

Gundersen, T. 2009. *A General View of Normalisation Through Atomic Flows*. PhD thesis, University of Bath. http://tel.archives-ouvertes.fr/docs/00/50/92/41/PDF/thesis.pdf.

Hilbert, D. and Ackermann, W. F. 1928. Grundzüge der theoretischen logik.

Jeřábek, E. 2009. Proof complexity of the cut-free calculus of structures. *Journal of Logic and Computation*, 19(2):323–339. http://www.math.cas.cz/~jerabek/papers/cos.pdf.

Kahramanoğulları, O. 2006. Reducing nondeterminism in the calculus of structures. In Hermann, M. and Voronkov, A., editors, *LPAR 2006*, volume 4246 of *Lecture Notes in Computer Science*, pages 272–286. Springer-Verlag. http://dx.doi.org/10.1007/11916277˙19.

Kleene, S. C. 1952. *Introduction to metamathematics*. North-Holland, Amsterdam.

Lafont, Y. 2002. Soft linear logic and polynomial time. *Theoretical computer science*, 318.

Milner, R., Parrow, J., and Walker, D. 1992. A calculus of mobile processes. *Information and Computation*, 100(1):1–77.

Prawitz, D. 1965. *Natural Deduction: a proof-theoretical study*. Dover Publications.

Sangiorgi, D. and Walker, D. 2001. *π-calculus: A Theory of Mobile Processes*. Cambridge University Press, New York, NY, USA.

Schütte, K. 1950. Schlussweisen-Kalküle der Prädikatenlogik. *Mathematische Annalen*, 122:47–65.

Troelstra, A. S. and Schwichtenberg, H. 1996. *Basic Proof Theory*. Cambridge University Press, New York, NY, USA.

Ringraziamenti

Ringrazio anzitutto i mei genitori Carmen e Roberto, senza i quali tutto questo non sarebbe stato possibile. Con loro, ringrazio tutta la mia famiglia per l'amore che mi hanno dato dacché sono al mondo.

Ringrazio i miei amici per le ore passate a discutere insieme, per aver ascoltato pazientemente i miei vaneggianti sproloqui, ma soprattutto per avermi dato la certezza di aver sempre qualcuno su cui contare.

Infine ringrazio i miei professori, per avermi ascoltato e per la pazienza che hanno avuto nel sopportare questo tremendo rompiscatole. Senza i vostri insegnamenti, ma non solo, senza il vostro esempio, non sarei quello che sono.

Grazie di cuore a tutti quanti, grazie a chi ha sempre creduto in me, grazie a chi non ci ha creduto mai, grazie agli amici ma anche ai nemici, grazie al contributo di tutti perché mi è stato indispensabile per raggiungere, oggi, questo risultato.