Logica Matematica

Lecture Notes

Corso del Prof. Stefano Aguzzoli

Carlo Castoldi e Edoardo Marangoni

University of Milan
Department of Computer Science
17 giugno 2021

Indice

Disclaimer

Questo documento contiene degli appunti presi durante il corso tenutosi nell'A.A. 2020/2021. Benché i materiali si attengono a ciò che è stato spiegato nel corso, queste note **non sono in alcun modo approvate dal docente**, con tutte le conseguenze implicate.

CAPITOLO 1

Introduzione

La Logica è una materia che ha una tradizione millenaria e trae le sue origini in ambito filosofico: la definizione che vedremo noi è infatti presa da quell'ambito, e noi la declineremo in una forma moderna. La Logica è lo studio dei meccanismi del ragionamento razionale. In altre parole, si intende lo studio della capacità di trarre conseguenze (corrette) da date assunzioni o premesse. Le assunzioni sono delle informazioni che affermano che il mondo sta in un certo modo e sono gestibili formalizzandole in un linguaggio formale. Queste informazioni codificano uno o più mondi possibili: vogliamo trarne delle conclusioni a partire di esse in un modo razionale.

1.1 Motivazioni

La Logica studia come si ragiona in maniera corretta e, per studiare come si ragiona, si può utilizzare come prima schematizzazione il partire da delle assunzioni vere e da quelle discendere a delle conclusioni.

Un esempio: ogni uomo è mortale. Socrate è un uomo. Dunque, Socrate è mortale. Questo è, in linguaggio naturale, un esempio di quanto detto prima: a partire da due informazioni date per vere (ogni uomo è mortale e Socrate è un uomo) si traggono delle conseguenze. Quanto fatto prima è un *sillogismo*, un punto antichissimo nella storia della Logica.

Altro esempio: ogni gatto ha sette zampe. Pluto è un gatto. Dunque, Pluto ha sette zampe. Anche questa è una deduzione esatta, nonostante per l'esperienza comune la prima assunzione è falsa; tuttavia, la Logica si occupa di *ogni* universo e pertanto il ragionamento è valido. Ogni Blabla è glug. Sbappo è Blabla. Dunque, Sbappo è glug. Questa forma di ragionamento è altrettanto corretta. Per non farsi distrarre dalle stranezze irrilevanti, si utilizza un linguaggio centrale per il discorso della Logica. Si formalizza quindi questo ragionamento: in primo luogo si astrae, fornendo un modello matematico per ragionare.

$$(\forall x P(x) \to Q(x) \land P(s)) \to Q(s)$$

Ogni fiore è profumato. La Rosa è profumata. Dunque, la Rosa è un fiore. Per mostrare che questo ragionamento è falso, si può anche utilizzare l'intuizione: se Rosa è mia nonna, benché il senso metaforico sia valido, Rosa è un po' vecchia e pertanto il ragionamento non è valido. In che modo è cambiato il ragionamento?

$$(\forall x P(x) \to Q(x) \land Q(s)) \nrightarrow P(s)$$

In questo caso, si sta cercando di verificare la premessa data la conclusione, al contrario di quanto accadeva per il sillogismo aristotelico; questo modo di ragionare non può funzionare.

Matematica Vi sono almeno due sensi per cui la Logica è matematica. Il primo è quello che abbiamo introdotto immediatamente al discorso iniziale: la matematica è utilizzata per la necessità di *astrarre* solamente le informazioni rilevanti scartando il resto in un contesto con molte informazioni che non ci interessano, come accade per il linguaggio naturale. La trasformazione delle assunzioni e delle conclusioni da una forma in linguaggio naturale alla forma astratta permette di arrivare a delle **forme**. I concetti che andremo a formalizzare avranno una sintassi dettata da un **linguaggio formale** e un significato semantico **algebrico-insiemistico**, ottenendo un **formalismo**, un modo preciso, rigoroso e privo di ambiguità per esprimere ciò che si vuole esprimere in Logica.

In seconda battuta, la Logica si usa per studiare le strutture matematiche, ossia si usa *per fare* matematica. Aprendo un testo qualsiasi di Logica matematica si vedrà come gli esempi più interessanti siano basati sulla matematica: gruppi, campi e teoremi vari.

Una terza parola chiave, che conclude la parte motivazionale, è **Informatica**, intesa come *Computer Science*, inteso come capire il processo dei sistemi computazionali. È stata infatti una grande rivincita della Logica durante il secolo scorso, che ha visto nascere i fondamenti della computazione partendo da strumenti logici. Se esiste, la differenza tra *Logica* e *Informatica* sono i focus diversi: la prima è più vicina ad un approccio dichiarativo, concentrandosi su ciò che si può concludere da determinate premesse, mentre la seconda è più vicina ad approcci procedurali o imperativi.

Esercizio Se piove prendo l'ombrello. È lo stesso caso di dire che:

1. Se non piove non prendo l'ombrello
2. Se non prendo l'ombrello non piove
3. Se prendo l'ombrello allora piove
4. O non piove o prendo l'ombrello
5. Piove solo se prendo l'ombrello
6. Se prendo l'ombrello piove
7. Piove se e solo se prendo l'ombrello
8. Nessuna delle precedenti

Benché non si sappia cosa voglia dire "lo stesso caso", tentiamo di dare le soluzioni a questo problema. Nel linguaggio naturale non si può fare a meno di sentire una dinamica: si vede che piove e allora si prende l'ombrello e si esce. La logica proposizionale non vede questa dinamicità: per farlo si devono elaborare formule che esplicitano la dinamicità.

Una definizione più precisa di cosa voglia dire che due "frasi" siano "uguali": esse sono "equivalenti" quando sono vere nelle medesime circostanze. Questa è nuovamente una definizione che pecca di precisione in quanto non espressa matematicamente. Cosa vuol dire "medesime" e "circostanze"? Un'interpretazione intuitiva che mette in luce la "circostanza" della frase "Se piove prendo l'ombrello" è che vi è una dipendenza tra il fatto che *piove* e il fatto di *prendere l'ombrello*.

In tutte le circostanze possibili, vi sono delle situazioni in cui è vero che piove e delle situazioni in cui non è vero e analogamente accade per il fatto di prendere l'ombrello. Di tutte le possibili circostanze ci interessano solo quattro: quando non piove e quando non si prende l'ombrello, quando piove e si prende l'ombrello, quando piove e non si prende l'ombrello e, infine, quando non piove e si prende l'ombrello.

La frase si può dunque rappresentare nella forma

$$P \to Q$$

che rappresenta il *se...allora*. L'implicazione è infatti la più difficile da accettare a livello intuitivo. Senz'altro ci sono delle situazioni in cui non abbiamo dubbi: per esempio, quando sia P e Q sono vere, cioè, sapendo che piove e che si prende l'ombrello la relazione causale sembra sussistere e quindi $P \to Q$ è verificata; quando P è vero e Q è falso, risulta infine che $P \to Q$ è falsa. Ora, potrebbe succedere che la risposta non sia esattamente quella che ci aspettiamo: cominciamo assumendo che non piova ma si prenda l'ombrello. Risulta che $P \to Q$ è vera, anche se l'antecedente è falso: già questa cosa può suonare strano, in quanto non suona giusto che il fatto che non piova implichi il fatto che si prenda l'ombrello. La questione riguarda sostanzialmente il linguaggio naturale di per sé e torneremo su questo discorso in futuro: con lo stesso approccio, si arriva a dire che se P e Q sono false allora $P \to Q$ è vera.

Allora, per concludere il nostro esempio: si può cominciare dicendo che l'ottava frase è falsa, ossia vi sono, tra le prime sette frasi, alcune frasi equivalenti. La prima è sbagliata, in quanto vi è la possibilità di prendere l'ombrello anche se non piove. Il rapporto di causalità si rivede anche nella terza frase, che è falsa. La quarta frase necessità l'analisi della disgiunzione, l'OR, che in questa situazione va interpretato come un OR inclusivo, ossia un $\vee$. Analizzando la frase "O non piove o prendo l'ombrello" ci si accorge come si possa tradurre in $\neg P \vee Q$, che ha la stessa "immagine di verità" dell'implicazione, pertanto anche la quarta è uguale. Questo è importante in quanto è una **realizzazione materiale** dell'implicazione! La quinta frase si può nuovamente interpretare come $P \to Q$ ed è pertanto vera. La sesta frase inverte il rapporto, facendo in modo che $Q \to P$, che è falso; analogamente accade per la settima.

Un'ulteriore interpretazione dell'implicazione è: Ogni qualvolta P è vera, anche Q è vera.

1.2 Programma

Il corso tratterà inizialmente la Logica Proposizionale, mentre la seconda parte tratterà la Logica Predicativa o del Prim'ordine. Della Logica Proposizionale si definirà la sintassi, quindi Alfabeto, Connettivi e Formule per poi parlare di valutazione, tabelle di verità e princìpi come verofunzionalità e bivalenza. Si discuteranno le tautologie, le contraddizioni, le formule soddisfacibili e la nozione centrale di Conseguenza logica. Seguentemente si tratteranno decidibilità, correttezza e completezza, ma in realtà il primo Teorema che tratteremo sarà quello di Compattezza. Dopodiché si potranno affrontare i metodi formali di deduzione per la Logica Proposizionale. Accenneremo ai Seguenti e ai Tableau, oltre che ai calcoli alla Hilbert e i metodi assiomatici. Ci concentreremo sulla metodologia più adatta alla deduzione automatica, ossia i metodi refutazionali basati sul principio di risoluzione.

La seconda parte del corso tratterà la Logica dei Predicati, che per noi sarà un sinonimo di Logica del Prim'ordine. Dal punto di vista sintattico, si affronteranno più approfonditamente alfabeto, quantificatori, simboli di predicato e i simboli di funzione. Seguirà la semantica: descriveremo la semantica di Tarski, con la nozione fondamentale di L-Struttura e modelli; il concetto di Conseguenza logica,completezza e correttezza nell'ambito della Logica del Prim'ordine. Termineremo con i metodi di deduzione e le forme normali. Assieme alla Teoria di Herbrand, quest'ultime ci permetteranno di arrivare alle tecniche di deduzione automatica.

Parte I

Logica Proposizionale

CAPITOLO 2

Introduzione alla Logica Proposizionale

Si comincia ora l'introduzione alla Logica Proposizionale, partendo tuttavia da un concetto espresso senza formalizzarlo immediatamente:

Definizione (Enunciato)**.** Con il termine **enunciato** si intende una frase o un'espressione per la quale sia sensato chiedersi se sia vera o se sia falsa in ogni data circostanza, ossia ha un **valore di verità** relativo ad una certa circostanza.

"Piove" è un enunciato, così come "prendo l'ombrello" e "se piove prendo l'ombrello". Sapremmo già dire che quest'ultimo ha qualcosa di diverso dai primi: quest'ultimo infatti è un **enunciato composto**, mentre i primi sono **enunciati atomici**. Ci sono frasi che non sono enunciati e possiamo anche limitarci all'italiano per trovarne alcuni: "Paolo corre?" e "Piove?" non sono enunciati. Oltre al linguaggio naturale vi sono anche altre frasi che non sono enunciati, per esempio "2".

2.1 Senso, denotazione e connotazione di un enunciato

Il senso filosofico dei concetti di **denotazione** e **connotazione** verrà tralasciato e, per questo, verrà visto quanto basta per distinguerli.

Si considerino le seguenti espressioni del linguaggio dell'aritmetica:

- 4
- 2^2
- il predecessore di 5
- $3+1$

Nessuno di questi è un enunciato, ma non è necessario che lo siano. Sappiamo dire cosa significhino, in quanto matematicamente sono sempre modi per esprimere il *numero naturale* "quattro".
Questo esempio inquadra a livello intuitivo cosa sia la **denotazione** (il numero naturale quattro) e la **connotazione** (quattro diversi modi per ottenere quattro).

Anche a questo livello stiamo dicendo una cosa interessante, in quanto questo implica che dobbiamo essere molto precisi riguardo cosa dovrebbe essere la *denotazione* di qualcosa. Un'espressione ha, quindi, una denotazione che è qualcosa di diverso dalla sua connotazione.

In un modo astratto, una espressione E è un **nome** (e.g., 4, 2^2, "il predecessore di 5" o $3+1$) di qualcosa, il quale si riferisce in modo univoco a qualche **entità**. L'entità alla quale si riferisce l'espressione è essa stessa la denotazione. La connotazione, in questo senso, è

quanto l'espressione effettivamente esprime, ossia tutto il resto dell'informazione contenuta nell'espressione stessa.

La logica studia la denotazione degli enunciati (chiamati anche *sentences*) e non le connotazioni, in quanto esse sono troppo difficili da gestire a livello iniziale. Le denotazioni godono infatti dell'importante proprietà dell'**invarianza per sostituzione**, ossia se ad un'espressione si cambiano delle parti sostituendole con parti denotazionalmente uguali, la denotazione globale non cambia, mentre la connotazione può potenzialmente cambiare totalmente, come dimostrano le quattro frasi iniziali.

2.1.1 Denotazione di un Enunciato

Si può definire ora, più formalmente, cosa sia la **denotazione** di un enunciato. Si prenda, per esempio, l'enunciato

$$4 = \text{pred}(5)$$

e si applichi una sostituzione con espressioni denotazionalmente equivalenti:

$$4 = 4$$

Il principio d'invarianza dice che questi due enunciati sono **denotazionalmente** equivalenti in quanto entrambe sono enunciati "veri". Questo benché l'ultimo enunciato contenga meno informazioni rispetto all'altro.
Questo ci dice che la denotazione di un enunciato è il suo *valore di verità*, e perciò gli enunciati non sono altro che una forma connotazionale di una costante: `vero` o `falso`.

L'oggetto della Logica Proposizionale sono le **proposizioni**: il contenuto denotazionale degli enunciati, che può essere `vero` o `falso`.

2.2 Enunciati e Connettivi

Alcuni enunciati semplici come "Piove" o "Paolo corre" sono definiti **atomici** in quanto la loro denotazione è solamente un valore di verità. Altri enunciati, definiti **composti**, sono enunciati che si possono "smontare", come "Piove e c'è vento", che è chiaramente composto dagli enunciati atomici "Piove" e "c'è vento". Il connettivo "e" è ciò che li unisce, come potrebbe accadere anche per "o", "non" e "se...allora".

Definizione (Enunciato semplice)**.** Un enunciato semplice (o atomico) rappresenta un fatto *denotato* da un valore di verità, detto **valore semantico**. È rappresentato da un *simbolo* chiamato **lettera proposizionale** scelto dall'insieme infinito $\mathcal{L}$: il **linguaggio proposizionale**.

Per esempio: $p, q, r, p_1, \cdots \in \mathcal{L}$

Definizione (Enunciato composto)**.** Un enunciato composto è una rappresentazione di un fatto composto da enunciati semplici tramite i seguenti **connettivi**: $\wedge, \vee, \neg$ e $\rightarrow$.
Il valore denotazionale di ogni enunciato composto dipende dai valori denotazionali degli enunciati atomici e dal valore semantico dei connettivi che lo compongono.

Per esempio: $p \wedge q$, $p \vee q$, $\neg p$

CAPITOLO 3

Sintassi della Logica Proposizionale

Dopo aver introdotto la Logica Proposizionale, si può ora formalizzare la struttura sintattica degli enunciati: l'insieme $\mathcal{F}_\mathcal{L}$ degli enunciati costruibili sul linguaggio $\mathcal{L}$ rispettando la *sintassi degli enunciati* definita come segue:

Definizione. (Sintassi degli Enunciati) Vi sono diversi modi per definire la sintassi degli enunciati:

1. $\mathcal{F}_\mathcal{L}$ è il più piccolo insieme tale che
   - per ogni $p \in \mathcal{L}$ si ha $p \in \mathcal{F}_\mathcal{L}$
   - se $A, B \in \mathcal{F}_\mathcal{L}$ allora anche $(A \wedge B) \in \mathcal{F}_\mathcal{L}$, $(A \vee B) \in \mathcal{F}_\mathcal{L}$, $(A \to B) \in \mathcal{F}_\mathcal{L}$ e $(\neg A) \in \mathcal{F}_\mathcal{L}$, dove A e B possono essere a loro volta enunciati complessi.
2. $\mathcal{F}_\mathcal{L}$ è l'intersezione di tutti gli insiemi $X \subseteq (\mathcal{L} \cup \{\wedge, \vee, \to, \neg\})^*$[1] tali che
   - $L \subseteq X$
   - se $A, B \in X$ allora $(A \wedge B)$, $(A \vee B)$, $(\neg A)$ e $(A \to B)$ sono contenuti in X.
3. (induttiva) $\mathcal{F}_\mathcal{L}$ è l'insieme che rispetta le condizioni seguenti:
   - $\mathcal{L} \subseteq \mathcal{F}_\mathcal{L}$: se $p \in \mathcal{L}$, allora $p \in \mathcal{F}_\mathcal{L}$
   - Se $A, B \in \mathcal{F}_\mathcal{L}$, allora $(A \wedge B)$, $(A \vee B)$, $(\neg A)$ e $(A \to B)$ sono contenuti in $\mathcal{F}_\mathcal{L}$
   - Nient'altro appartiene a $\mathcal{F}_\mathcal{L}$.

Esercizio Scrivere qualcosa che non sia un enunciato utilizzando solamente la sintassi della logica proposizionale. Un esempio è $pq\neg$. Un altro è $\to q$.

3.1 $\mathcal{L}$-Costruzioni

Per stabilire se una stringa $w \in (\{\wedge, \vee, \neg, \to\} \cup L)^*$ è anche $w \in \mathcal{F}_\mathcal{L}$ si deve trovare una **$\mathcal{L}$-costruzione**—o *certificato*—adatta: una sequenza finita di formule $w_1, w_2, \cdots, w_n$ tale che $w_n = w$ e ogni w_i:

- o è una lettera proposizionale $w_i \in \mathcal{L}$

[1] L'utilizzo dell'operatore * è paragonabile all'operatore Stella di Kleene nella Teoria dei Linguaggi e denota l'insieme di tutte le stringhe di lunghezza finita componibili utilizzando le lettere dell'alfabeto indicato, in questo caso i connettivi e le lettere proposizionali.

- o esiste $j < i$ tale che $w_i = \neg w_j$
- o esistono $k, j < i$ tali per cui $w_i = w_k \wedge w_j$ o $w_i = w_k \vee w_j$ o $w_i = w_k \rightarrow w_j$.

Per esempio, sia $w = (p \wedge q) \rightarrow r$. Per dimostrare che $w \in \mathcal{F}_{\mathcal{L}}$ si possono introdurre come $\mathcal{L}$-costruzione:

$$w_1 = p, w_2 = q, w_3 = (p \wedge q), w_4 = r, w_5 = (p \wedge q) \rightarrow r$$

questa $\mathcal{L}$-costruzione *certifica* che $w \in \mathcal{F}_{\mathcal{L}}$. In termini computazionali, si può controllare velocemente che quanto fatto sopra sia una costruzione corretta e che pertanto $w \in \mathcal{F}_{\mathcal{L}}$. Si noti, inoltre, che questa $\mathcal{L}$-costruzione non è unica.

Proprietà (Proprietà di unica leggibilità dei Certificati)**.** Se $w \in \mathcal{F}_{\mathcal{L}}$, allora vale uno e uno solo dei seguenti casi: o $w \in \mathcal{L}$, o esiste v tale che $w = \neg v$, o esistono v_1, v_2 tali che $w = (v_1 \wedge v_2)$, $(v_1 \vee v_2)$ o $(v_1 \rightarrow v_2)$, dove i v_i sono determinati univocamente.

Questa proprietà garantisce l'esistenza di una sorta di operazione inversa della costruzione di certificati: mentre quest'ultimo "monta" una formula, questa proprietà garantisce il fatto che sia possibile "smontarla" in un unico modo!

Questa proprietà non è condivisa tra tutti i linguaggi formali: per esempio nelle grammatiche una stringa $w = ciao$ può essere composta da $v_1 = \varepsilon$ e $v_2 = ciao$ eccetera. Spesso, il concetto di leggibilità può essere espresso anche tramite il concetto di **albero di parsing** di una formula, che è unico e mostra come essa sia costruita.

3.2 Osservazioni e convenzioni riguardo alla sintassi

Esistono ulteriori connettivi (o connettivi derivati) oltre a quelli utilizzati fino ad ora, ossia $\wedge$, $\vee$, $\neg$ e $\rightarrow$: uno di questi è $\bot$, che è un connettivo di arità zero che denota il falso; un altro è $\top$, che è un connettivo zerario che denota il sempre vero e chiaramente $\bot = \neg\top$. Altro connettivo è $\iff$, definito come $(A \rightarrow B) \wedge (B \rightarrow A)$.

Una seconda osservazione riguarda l'uso delle parentesi: per come abbiamo definito le formule, l'oggetto $p \wedge q \notin \mathcal{F}_{\mathcal{L}}$ in quando mancano le parentesi. In formule complesse, le parentesi possono aggiungere complicatezza e portare l'errore: useremo il buonsenso per "dimenticarci" delle parentesi laddove non ci sia pericolo di confusione. Tuttavia non bisogna farsi trasportare troppo, in quanto esistono delle parentesi necessarie, come per esempio $(p \wedge q) \vee r$ oppure $p \wedge (q \vee r)$.

CAPITOLO 4

Semantica della Logica Proposizionale Classica

4.1 Fondamenti della Semantica

Per dare una definizione di semantica si utilizzano due princìpi guida: il **principio di bivalenza** e il **principio di verofunzionalità**.

Principio (Bivalenza)**.** Un enunciato, in ogni circostanza, è o vero o falso.

Il principio di bivalenza ha come conseguenza il principio del terzo escluso.

Principio (Verofunzionalità, composizionalità o estensibilità)**.** Il valore di verità di un enunciato composto dipende solo dal valore di verità degli enunciati che lo compongono e dal significato del connettivo che li unisce.

Dato il principio di verosimiglianza, per dare la semantica ad un connettivo come $\wedge$ si deve dire qual è, sotto ogni circostanza, il valore di verità di $A \wedge B$, il quale può essere `vero` o `falso` e può dipendere solo dal valore di verità di A e B e dal significato fissato per $\wedge$.
È per questo necessario definire informalmente una *circostanza* come un assegnamento che definisce lo stato vero o falso di una lettera proposizionale (o di una formula), definita come una funzione v:

$$v : \mathcal{L} \to \{0,1\}$$

Quindi $v(A \wedge B) \in \{0,1\}$ e, grazie al principio di Verofunzionalità, dipende solo da $v(A)$, $v(B)$ e dal significato fissato per $\wedge$, definito come

$$I_\wedge : \{0,1\}^2 \to \{0,1\}$$

E si ha, quindi

$$v(A \wedge B) = I_\wedge(v(A), v(B))$$

Importanza della Verofunzionalità Non è una proprietà così scontata come sembrerebbe: si supponga per un attimo che invece di Logica si stia studiando Probabilità, tentando di formalizzarla come stiamo formalizzando ora la Logica Proposizionale.
Si consideri una circostanza in cui due eventi hanno probabilità $p(A) = p(B) = \frac{1}{2}$. Sappiamo che la probabilità $p(A \to A) = 1$ rappresenta l'evento certo. Se valesse la proprietà verofunzionale si potrebbe dire che visto che $p(\frac{1}{2} \to \frac{1}{2}) = 1$, allora vale anche $p(A \to B) = 1$. Assurdo!
Per concretezza, si immagini $A =$ "domani piove" e $B =$ "a Pasqua nevica".

In conclusione, la Verofunzionalità è una caratteristica stringente della Logica Proposizionale da non dare affatto per scontata.

4.1.1 Definizione di Semantica

Siamo finalmente pronti per dare una definizione formale della **semantica** della Logica Proposizionale:

Definizione (Semantica della Logica Proposizionale)**.** Un assegnamento (o valutazione) è un'arbitraria funzione

$$v : \mathcal{L} \to \{0,1\}$$

che formalizza la nozione intuitiva di circostanza (o *mondo possibile*).

Definizione (Semantica dei connettivi)**.** La **semantica dei connettivi** è espressa tramite delle funzioni:

$$I_\wedge : \{0,1\}^2 \to \{0,1\}$$

che identificano una *tabella di verità*.

I valori di verità dei connettivi possono anche essere espressi in forma algebrica, per esempio:

$$\widetilde{v}(A \to B) = min\{1 - \widetilde{v}(A), \widetilde{v}(B)\}$$

Definizione (Semantica degli Enunciati)**.** La **semantica degli enunciati** è l'**estensione canonica**

$$\widetilde{v} : \mathcal{F}_\mathcal{L} \to \{0,1\}$$

ossia l'estensione di v a tutte le formule, definita in questo modo:

- $\widetilde{v}(p) = v(p)$ se $p \in \mathcal{L}$
- $\widetilde{v}(A \wedge B) = I_\wedge(\widetilde{v}(A), \widetilde{v}(B))$ se $p \in \mathcal{F}_\mathcal{L}$
- $\widetilde{v}(A \vee B) = I_\vee(\widetilde{v}(A), \widetilde{v}(B))$se $p \in \mathcal{F}_\mathcal{L}$
- $\widetilde{v}(A \to B) = I_\to(\widetilde{v}(A), \widetilde{v}(B))$ se $p \in \mathcal{F}_\mathcal{L}$
- $\widetilde{v}(\neg A) = I_\neg(\widetilde{v}(A))$ se $p \in \mathcal{F}_\mathcal{L}$

Con un poco importante abuso notazionale, si tenderà ad evitare di esplicitare $\widetilde{v}$ in favore della notazione v.

4.2 Nozioni Semantiche Fondamentali

Definizione (Tautologia)**.** Una formula $F \in \mathcal{F}_\mathcal{L}$ è una *tautologia* se e solo se

$$\forall \widetilde{v} : \mathcal{F}_\mathcal{L} \to \{0,1\} \quad \widetilde{v}(F) = 1$$

Definizione (Formula Soddisfacibile)**.** Una formula $F \in \mathcal{F}_\mathcal{L}$ è *soddisfacibile* se e solo se

$$\exists \widetilde{v} : \mathcal{F}_\mathcal{L} \to \{0,1\} : \widetilde{v}(F) = 1$$

Definizione (Contraddizione)**.** Una formula $F \in \mathcal{F}_\mathcal{L}$ è una *contraddizione* (o insoddisfacibile, refutabile) se e solo se

$$\forall \widetilde{v} : \mathcal{F}_\mathcal{L} \to \{0,1\} \quad \widetilde{v}(F) = 0$$

C'è un fatto molto semplice che lega tra di loro questi concetti:

Teorema. *$F \in \mathcal{F}_\mathcal{L}$ è una tautologia se e solo se $\neg F$ è insoddisfacibile.*

Dimostrazione.

$$\begin{aligned} & F \text{ tauto} \\ \iff & \widetilde{v}(F) = 1 \quad \forall \widetilde{v} : \mathcal{F}_\mathcal{L} \to \{0,1\} && \text{per def. di tauto} \\ \iff & \widetilde{v}(\neg F) = 0 \quad \forall \widetilde{v} : \mathcal{F}_\mathcal{L} \to \{0,1\} && \text{per def. } I_\neg \\ \iff & \neg F \text{ insodd.} && \text{per def. di contraddizione} \end{aligned}$$

□

4.2.1 Principio d'Induzione su $\mathcal{F}_\mathcal{L}$

Benché non sia una sfaccettatura prettamente semantica, in quanto verrà (anzi, è già stato) utilizzato abbondantemente, è importante formalizzare il **principio d'induzione**. Sia P una proprietà delle formule; si ha che P vale per ogni formula $F \in \mathcal{F}_\mathcal{L}$ se e solo se:

- **base** ($F = p$): P vale per ogni $p \in \mathcal{L}$
- **passo**: se P vale per $A, B \in \mathcal{F}_\mathcal{L}$, allora vale anche per $\neg A$, $A \to B$, $A \vee B$ e $A \wedge B$.

Una proprietà si può dimostrare induttivamente anche grazie alla definizione induttiva di $\mathcal{F}_\mathcal{L}$. Sia:

$$I = \{F \in \mathcal{F}_\mathcal{L} : \text{ la proprietà } P \text{ vale per } F\}$$

Si vuole mostrare $I = \mathcal{F}_\mathcal{L}$, ossia che I è esattamente l'insieme di tutte le formule. Questo si fa mostrando che $I \subseteq \mathcal{F}_\mathcal{L}$ (ovvio per def. di I) e che $\mathcal{F}_\mathcal{L} \subseteq I$, come segue.

Dimostrazione. I soddisfa il primo e il secondo punto della definizione induttiva di $\mathcal{F}_\mathcal{L}$ perché contiene tutte le lettere proposizionali (base) e, induttivamente le formule composte da sotto-formule $A, B \in I$. Allora I conterrà anche $(\neg A)$, $(A \to B)$, $(A \wedge B)$ e $(A \vee B)$.
Grazie al terzo punto della definizione, invece, si può affermare che $I = \mathcal{F}_\mathcal{L}$ in quanto non sono contenute altre formule in I (per definizione). □

Come esempio, si dimostra induttivamente il seguente lemma:

Lemma (Verità di una Formula)**.** *Sia $F \in \mathcal{F}_\mathcal{L}$ e siano $v, v' : \mathcal{L} \to \{0,1\}$.*
Se $v(p) = v'(p) \forall p \in \mathcal{L} \in F$[1], allora $\widetilde{v}(F) = \widetilde{v}'(F)$.

Dimostrazione. Per induzione su $\mathcal{F}_\mathcal{L}$.

- **base** ($F = p \in \mathcal{L}$):

$$\begin{aligned} & v(p) = v'(p) \\ \iff & \widetilde{v}(p) = \widetilde{v}'(p) && \text{per def. } \widetilde{v} \end{aligned}$$

- **passo**:

[1] Per ogni letterale nell'insieme dei letterali nella formula.

– $F = \neg A$:

$$\begin{aligned} \widetilde{v}(F) &= 1 - \widetilde{v}(A) && \text{per def. } \widetilde{v} \text{ e } I_\neg \\ &= 1 - \widetilde{v}'(A) && \text{per I.H.} \\ &= \widetilde{v}'(F) && \text{per def. } \widetilde{v} \text{ e } I_\neg \end{aligned}$$

– Se $F = (A \wedge B)$:

$$\begin{aligned} \widetilde{v}(F) &= \min\{\widetilde{v}(A), \widetilde{v}(B)\} && \text{per def. } \widetilde{v} \text{ e } I_\wedge \\ &= \min\{\widetilde{v}'(A), \widetilde{v}'(B)\} && \text{per I.H.} \\ &= \widetilde{v}'(F) && \text{per def. } \widetilde{v} \text{ e } I_\wedge \end{aligned}$$

– uguale per gli altri connettivi.

□

Il lemma appena provato ci garantisce che se vogliamo calcolare $\widetilde{v}(F)$ ci basta calcolare la tabella di verità di F. A questo punto si può calcolare, per ogni assegnamento, se una formula è vera, soddisfacibile, tautologica o insoddisfacibile.

Esercizio Date $A, B \in \mathcal{F}_\mathcal{L}$ si esaminino le seguenti formule. La formula $A \wedge \neg A$ è insoddisfacibile, come si può osservare dalla sua tabella di verità nella Tabella 4.1.

A	$A \wedge \neg A$
0	0
1	0

Tabella 4.1

La formula $A \to \neg A$ è soddisfacibile ma non tautologica, come si può osservare dalla sua tabella di verità nella Tabella 4.2.

A	$A \to \neg A$
0	1
1	0

Tabella 4.2

Si noti come è stato detto che A, B siano state definite come appartenenti all'insieme delle formule e non alle lettere (L), "imbrogliando", un poco, rispetto al lemma precedente. Tuttavia, A e B possono essere considerate come *metavariabili* a prescindere dalla loro complessità. Non è invece possibile fare il contrario, ossia considerare lettere come delle formule: per esempio, non è possibile dire che una lettera sia una tautologia.

Esercizio Verificare che le seguenti siano Tautologie per ogni $A, B \in \mathcal{F}_\mathcal{L}$:

1. $A \to (B \to A)$ (Weakening, Prefixing)
2. $(\neg A \to A) \to A$ (Consequentia Mirabilis)

3. $(A \to B) \vee (B \to A)$

4. $(A \to B) \to (\neg B \to \neg A)$ (Contronominale)

5. $(A \wedge B) \to A$

6. $A \vee \neg A$ (Tertium non datur)

7. $\bot \to A$ (Ex Falsum Quodlibet Sequitur)

Definizione (Tautologia)**.** Se la formula F è una tautologia, si indicherà

$$\models F$$

4.3 Semantica degli Insiemi di Formule

Definizione (Teoria)**.** Sia $\Gamma \subseteq \mathcal{F}_{\mathcal{L}}$ un insieme di formule. Γ è detto **teoria** ed è un modello per un mondo in cui tutte le formule che gli appartengono sono vere.

Si dice che Γ è soddisfacibile ($v \models \Gamma$) se e solo se

$$\exists v : \mathcal{L} \to \{0,1\} \quad \forall \gamma \in \Gamma : v(\gamma) = 1$$

o, in notazione alternativa, $v \models \gamma$.
Al contrario, la teoria Γ è insoddisfacibile ($v \not\models \Gamma$) se e solo se

$$\forall v : \mathcal{L} \to \{0,1\} \ \ \exists \gamma \in \Gamma : v \not\models \gamma$$

Definizione (Conseguenza Logica)**.** Sia $\Gamma \subseteq \mathcal{F}_{\mathcal{L}}$ e $A \in \mathcal{F}_{\mathcal{L}}$. La formula A è una **conseguenza logica** di Γ se e solo se

$$\forall v : v \models \Gamma, \ v \models A$$

Si denoterà questo fatto con la notazione $\Gamma \models A$.

Esercizio Esprimere il concetto di formula tautologica o tautologia ($\models A$) attraverso il concetto di conseguenza logica.
Quando si ammette una teoria, si restringe il campo dei possibili assegnamenti. Una formula è tautologica quanto è vera in ogni circostanza ed è verificata da ogni assegnamento. Sia Γ definito in questo modo. $\Gamma \subseteq \mathcal{F}_{\mathcal{L}}$ un insieme composto solo da tautologie. Allora $\Gamma \models A$. Dato che è sempre meglio avere come teoria la più semplice possibile, si può definire $\Gamma = \emptyset$, concludendo $\emptyset \models A$.
Ogni tanto si può utilizzare la notazione $\Gamma \cup \{A\} \models B$, che a volte viene semplificato in $\Gamma, A \models B$.

4.3.1 Proprietà semantiche fondamentali delle Teorie

Lemma 1 (Conseguenza Logica di una Formula da una Teoria insoddisfacibile)**.** *Sia $\Gamma \subseteq \mathcal{F}_{\mathcal{L}}$ una teoria e $A \in \mathcal{F}_{\mathcal{L}}$. Si ha che $\Gamma \models A$ se e solo se $\Gamma \cup \{\neg A\}$ è insoddisfacibile.*

Dimostrazione.

$$
\begin{array}{lll}
& \Gamma \models A & \\
\iff & \forall v \quad (\forall \gamma \in \Gamma,\ v \models \gamma) \text{ implica } v \models A & \text{per def. } \models \\
\iff & \forall v \quad \neg(\forall \gamma \in \Gamma,\ v \models \gamma) \vee v \models A & \text{implicazione materiale} \\
\iff & \forall v \quad \neg(\forall \gamma \in \Gamma,\ v(\gamma) = 1) \vee v(A) = 1 & \\
\iff & \forall v \quad (\exists \gamma \in \Gamma : v(\gamma) = 0) \vee v(\neg A) = 0 & \\
\iff & \forall v \quad \exists B \in \Gamma \cup \{\neg A\} \text{ t.c. } v(B) = 0 & \\
\iff & \Gamma \cup \{\neg A\} \text{ è insodd.} & \text{per def. insodd.}
\end{array}
$$

□

Lemma 2 (Lemma semantico o di deduzione)**.** *Siano P, Q due formule. Allora si dice che* $P \models Q$ *se e solo se* $\models P \to Q$.

Dimostrazione. Assumiamo che $P \models Q$.
Per definizione, $\forall v : \mathcal{L} \to \{0,1\} : v(P) = 1$ implica $v(Q) = 1$. È quindi chiaro che ogni volta che $v(P) = 1$, si ha $v(Q) = 1$; è perciò impossibile avere P vero e Q falso. Ne segue che $\models P \to Q$ è una tautologia, per definizione di $I_{\to}$. □

Il teorema generalizza il lemma alla seguente situazione.

Teorema (Thm. semantico o di deduzione)**.** *Sia* Γ *una teoria e P, Q formule. Allora* $\Gamma, P \models Q$ *se e solo se* $\Gamma \models P \to Q$.

Dimostrazione.

$$
\begin{array}{lll}
& \Gamma, P \models Q & \\
\iff & \forall v \quad ((\forall \gamma \in \Gamma, v(\gamma) = 1) \wedge v(P) = 1) \to v(Q) = 1 & \text{per def. } \models \\
\iff & \forall v \quad (\forall \gamma \in \Gamma, v(\gamma) = 1) \to v(P \to Q) = 1 & A \wedge B \to C \equiv A \to (B \to C) \\
\iff & \Gamma \models P \to Q & \text{per def. di } \models
\end{array}
$$

□

4.3.2 Teorema di Compattezza

Il teorema di compattezza è un teorema fondamentale della logica e varrà anche per la logica del primo ordine. Lo proviamo ora per la logica proposizionale. È in qualche modo, in forma astratta, un teorema di completezza.

Prima di mostrare l'enunciato, si introduce il concetto. È stata data la nozione di teoria, $\Gamma \subseteq \mathcal{F}_{\mathcal{L}}$: ci si chiede se serve, nella logica proposizionale, considerare teorie infinite (composte da un numero infinito di formule). A priori, sembrerebbe proprio di sì - d'altronde $\mathcal{F}_{\mathcal{L}}$ è di cardinalità numerabile - e si vedrà affrontando la Logica dei Predicati che una singola formula al primo ordine contiene informazioni di un numero infinito di formule proposizionali; questo basta per giustificare il caso in cui Γ sia una teoria infinita. Il punto è che non sembra possibile gestire la teoria infinita: qui torna utile il teorema di compattezza.

Il teorema di compattezza permette di ridurre l'analisi della soddisfacibilità di una teoria eventualmente infinita all'esame della soddisfacibilità dei suoi sottoinsiemi finiti. Tuttavia, è ovvio che il numero di sottoinsiemi finiti sia infinito.

Definizione. Γ *è finitamente soddisfacibile* (fin. sodd.) se $\forall \Gamma' \subseteq_\omega \Gamma$ (leggasi "Γ' sottoinsieme finito di Γ") si ha che Γ' è soddisfacibile.

Teorema 1 (di Compattezza)**.** $\Gamma \subseteq \mathcal{F}_\mathcal{L}$ *è soddisfacibile* $\iff$ Γ' *è soddisfacibile* $\forall \Gamma' \subseteq_\omega \Gamma$.

Γ è soddisfacibile $\implies$ Γ' è soddisfacibile $\forall \Gamma' \subseteq \Gamma$ è ovvio per definizione di soddisfacibilità di una teoria.

Dimostrazione ($\Longleftarrow$)

Il succo della dimostrazione sarà la relazione tra finitamente soddisfacibile e soddisfacibile: vogliamo provare Γ fin. sodd. $\implies$ Γ soddisfacibile.

Fissato una successione senza ripetizioni di tutte le formule in $F_i \in \mathcal{F}_\mathcal{L}$:

$$F_1, F_2, \cdots, F_k, \cdots$$

si costruisce una successione infinita di insiemi di formule:

$$D_0, D_1, \cdots, D_k, \cdots$$

definita induttivamente sull'indice i di D_i:

- **base**: $D_0 = \Gamma$
- **passo**: $D_{n+1} = \begin{cases} D_n \cup \{F_{n+1}\} & \text{se } D_n \cup \{F_{n+1}\} \text{ è finitamente soddisfacibile} \\ D_n \cup \{\neg F_{n+1}\} & \text{altrimenti} \end{cases}$

Bisogna sottolineare che la definizione di D_n non garantisce, a priori, che ogni D_n sia finitamente soddisfacibile, anche se si dimostrerà che è in effetti così. In altre parole, definire $D_n = D_{n-1} \cup \{\neg F_{n+1}\}$ se l'alternativa non è finitamente soddisfacibile, non ci assicura a priori che D_n sia fin. sodd.; per arrivare a tale conclusione è necessaria una dimostrazione. Si definisce infine

$$D = \bigcup_{i \in \mathbb{N}} D_i$$

Prima di passare alle effettive dimostrazioni, si noti come per dimostrare una proprietà di Γ stiamo cercando di dimostrare qualcosa di relativo a un insieme infinitamente più grande, D; benché questa possa sembrare un'idea balzana, il Teorema di Compattezza ci dimostrerà che questo è il procedimento giusto per ottime ragioni.

Fatto 1: *D_n è finitamente soddisfacibile $\forall n$.*

Dimostrazione. Per induzione su n:

- **base**: $D_0 = \Gamma$ è finitamente soddisfacibile per ipotesi.
- **passo** D_{n+1}: per assurdo, assumiamo D_{n+1} non finitamente soddisfacibile.

$$\begin{aligned} &\iff \exists D' \subseteq_\omega D_n \cup \{F_{n+1}\} \text{ tale che } D' \text{ è insoddisfacibile} && (1) \\ &\quad \text{e } \exists D'' \subseteq_\omega D_n \cup \{\neg F_{n+1}\} \text{ tale che } D'' \text{ è insoddisfacibile} && (2) \end{aligned}$$

Senza perdita di generalità, si può assumere che:

$$\begin{aligned}
& F_{n+1} \in D' \text{ e } \neg F_{n+1} \in D'' && \text{perchè } D_n \text{ è sodd. per ipotesi induttiva} \\
\iff & D' = E' \cup \{F_{n+1}\} \text{ e } D'' = E'' \cup \{\neg F_{n+1}\} && \text{con } E', E'' \subseteq_\omega D_n : F_{n+1} \notin E', \neg F_{n+1} \notin E'' \\
\iff & D' \subseteq E' \cup E'' \cup \{F_{n+1}\} \text{ e} && \text{insod. per } (1) \\
& D'' \subseteq E' \cup E'' \cup \{\neg F_{n+1}\} && \text{insod. per } (2)
\end{aligned}$$

Ma $E' \cup E'' \subseteq_\omega D_n$ e per ipotesi induttiva D_n è finitamente soddisfacibile, ed è quindi $E' \cup E''$ soddisfacibile ed esiste un assegnamento tale che $v \models E' \cup E''$. Sappiamo che $v(F_{n+1})$ è uguale a 0 o a 1 e pertanto non è possibile che entrambi $\{F_{n+1}\}$ e $\{\neg F_{n+1}\}$ siano falsi. Abbiamo raggiunto la contraddizione che conclude la prova per assurdo mostrando D_{n+1} finitamente soddisfacibile. Questo chiude a sua volta la prova per induzione, mostrando che ogni D_n per $n \in N$ è finitamente soddisfacibile.

□

Fatto 2: $D = \bigcup_{i \in \omega} D_i$ *è a sua volta finitamente soddisfacibile.*

Questo non è necessariamente ovvio a partire dal fatto che $D = \bigcup_{i \in \omega} D_i$ con D_i fin. sodd.

Dimostrazione. Siano:

- $D' \subseteq_\omega D$, un sottoinsieme finito qualunque, con $D' = \{F_{i_1}, F_{i_2}, \cdots, F_{i_u}\}$
- $k = \max_{j=1,\ldots,u} i_j$, l'indice massimo

Allora:

$$\begin{aligned}
& D' \subseteq_\omega D_k \\
\implies & D' \text{ è finitamente soddisfacibile} && \text{per Fatto 1} \\
\implies & D \text{ è finitamente soddisfacibile} && \text{quando } D' = D
\end{aligned}$$

□

Fatto 3: *Per ogni formula o enunciato F_t esattamente **una** tra F_t e $\neg F_t$ appartiene a D.*

Anche questo non è necessariemente garantito, in quanto F_t e $\neg F_t$ non hanno lo stesso indice, ossia $\neg F_t$ non ha indice t.

Dimostrazione. Per costruzione della sequenza di D_i:

$$\begin{aligned}
& F_t \in D_t \vee \neg F_t \in D_t \\
\implies & F_t \in D \vee \neg F_t \in D && \text{dato che ogni } D_t \subseteq D
\end{aligned}$$

Si può escludere che ci siano entrambe: $\{F_t, \neg F_t\} \nsubseteq D$. Infatti, per assurdo:

$$\begin{aligned}
& \{F_t, \neg F_t\} \subseteq D \\
\implies & D \text{ è fin. sodd.} && \text{per Fatto 2} \\
\implies & \{F_t, \neg F_t\} \text{ è sodd.} && \text{per def. fin. sodd.} \\
\implies & \bot && \text{per la semantica della negazione } I_\neg
\end{aligned}$$

□

Fatto 4: *Sia* $v_D : \mathcal{L} \to \{0,1\}$ *t.c.* $\forall p \in \mathcal{L},\ v_D(p) = 1 \iff p \in D$. *Allora:*

$$v_D \models D$$

Ovvero $\forall F \in D,\ \widetilde{v}_D(F) = 1$

Si noti che v_D è *ben definito* in quanto necessariamente $p \in D$ o $p \notin D$.

Dimostrazione. Per induzione strutturale proviamo la coimplicazione:

$$\forall F \in \mathcal{F}_{\mathcal{L}},\ \widetilde{v}_D(F) = 1 \iff F \in D$$

Questo non è uguale alla definizione precedente, in quanto in principio è stato definito per le lettere proposizionali e non per la funzione estesa ($\widetilde{v}_D$). Ci basterebbe provare che $F \in D$ implica $v_D(F) = 1$ ma per convenienza proviamo anche che $v_D(F) = 1$ implica $F \in D$.

- **base**: $F = p$, con $p \in \mathcal{L}$

$$v_D(F) = 1 \iff v_D(p) \iff p \in D \iff F \in D \text{ per definizione di } v_D$$

- **passo**:
  - se $F = \neg G$:

$$\begin{aligned} & v_D(F) = 1 \\ \iff & v_D(\neg G) = 1 \\ \iff & v_D(G) = 0 && \text{per def. di } I_\neg \\ \iff & G \notin D && \text{per ipotesi induttiva} \\ \iff & \neg G \in D && \text{per Fatto 3} \\ \iff & F \in D \end{aligned}$$

  - se $F = (G \wedge H)$:

$$\begin{aligned} & v_D(F) = 1 \\ \iff & v_D(G \wedge H) = 1 \\ \iff & v_D(G) = 1 \wedge v_D(H) = 1 && \text{per def. di } I_\wedge \\ \iff & G \in D \wedge H \in D && \text{per ipotesi induttiva} \\ \iff & (G \wedge H) \in D \end{aligned}$$

L'ultima implicazione è dimostrabile per assurdo:

$$\begin{aligned} & (G \wedge H) \notin D && \text{deduzione assurda } (\Rightarrow) \\ \Longrightarrow & \neg(G \wedge H) \in D && \text{per Fatto 3} \\ \Longrightarrow & \{G, H, \neg(G \wedge H)\} \subseteq_\omega D \text{ fin. sodd.} && \text{per Fatto 2, ma } \bot \text{ per } I_\neg, I_\wedge \end{aligned}$$

$$\begin{aligned} & G \notin D \vee H \notin D && \text{deduzione assurda } (\Leftarrow) \\ \Longrightarrow & \text{se } G \notin D : \ \neg G \in D && \text{per Fatto 3} \\ & \{G \wedge H, \neg G\} \subseteq_\omega D \text{ fin. sodd.} && \text{per Fatto 2, ma } \bot \text{ per } I_\neg, I_\wedge \\ & \text{se } H \notin D : \ \neg H \in D && \text{per Fatto 3} \\ & \{G \wedge H, \neg H\} \subseteq_\omega D \text{ fin. sodd.} && \text{per Fatto 2, ma } \bot \text{ per } I_\neg, I_\wedge \end{aligned}$$

– se $F = (G \vee H)$:

$$\begin{aligned} & v_D(F) = 1 \\ \iff & v_D(G \vee H) = 1 \\ \iff & v_D(G) = 1 \vee v_D(H) = 1 && \text{per def. di } I_\vee \\ \iff & G \in D \vee H \in D && \text{per ipotesi induttiva} \\ \iff & (G \vee H) \in D \end{aligned}$$

L'ultima implicazione è dimostrabile per assurdo:

$$\begin{aligned} & (G \vee H) \notin D && \text{deduzione assurda}(\Rightarrow) \\ \implies & \neg(G \vee H) \in D && \text{per Fatto 3} \\ \implies & \text{se } G \in D : \ \{G, \neg(G \vee H)\} \subseteq_\omega D \text{ fin. sodd.} && \text{per Fatto 2, ma } \perp \\ & \text{se } H \in D : \ \{H, \neg(G \vee H)\} \subseteq_\omega D \text{ fin. sodd.} && \text{per Fatto 2, ma } \perp \end{aligned}$$

$$\begin{aligned} & \neg(G \in D \vee H \in D) && \text{deduzione assurda } (\Leftarrow) \\ \implies & G \notin D \wedge H \notin D && \text{per De Morgan} \\ \implies & \neg G \in D \wedge \neg H \in D && \text{per Fatto 3} \\ \implies & \{\neg G, \neg H, \{G \vee H\}\} \subseteq_\omega D \text{ fin. sodd.} && \text{per Fatto 2, ma } \perp \end{aligned}$$

– se $F = (G \to H)$: *uguale, usa il fatto che è vero se G è falso o H è vero*

□

Ora che abbiamo dimostrato che, se Γ è fin. sodd., esiste un assegnamento $v_D : v_D \models D$, abbiamo dimostrato anche che $v_D \models \Gamma$ (dato che $\Gamma \subseteq D$).
Γ è dunque soddisfacibile.

Contronominale

Noi useremo per lo più la versione *contronominale* del Teorema di Compattezza:
Γ insoddisfacibile $\implies \exists \Gamma' \subseteq_\omega \Gamma : \Gamma'$ insoddisfacibile.

4.3.3 Osservazioni sul Teorema di Completezza

La dimostrazione del Teorema di Compattezza lavora ampliando al massimo la teoria Γ che si vuole analizzare:

$$\Gamma = D_0 \subseteq D_1 \subseteq D_2 \cdots \subseteq D_k \cdots \subseteq D$$

Definizione (Insieme massimamente soddisfacibile)**.** Un sottoinsieme di formule $E \subseteq \mathcal{F}_\mathcal{L}$ è **massimamente soddisfacibile** sse E è sodd. e $\forall F \in \mathcal{F}_\mathcal{L} : F \notin E$ si ha che $E \cup \{F\}$ è insodd.

Osservazione 1. D è un **ampliamento massimale** di Γ, in quanto se $\overline{D} = D \cup \{F\}$ con $F \notin D$ allora $\overline{D}$ è insoddisfacibile per Fatto 3 ($\neg F \in D$).

D è solo *uno* degli ampliamenti massimali possibili, ed è quello che è stato costruito a partire da Γ e da una regola di costruzione di D_{n+1} non univoca che dipende dall'elenco delle formule (i.e., può succedere che, dato un letterale p, sia $D_n \cup \{p\}$ sia $D_n \cup \{\neg p\}$ siano sodd.).

Osservazione 2. A livello delle informazioni contenute, un insieme massimale soddisfacibile D si comporta come un assegnamento (e viceversa).

Per ogni assegnamento $v : \mathcal{L} \to \{0,1\}$ si può creare $D_v = \{F \in \mathcal{F}_\mathcal{L} : v(F) = 1\}$ massimamente soddisfacibile.

Utilizzo Il Teorema di Compattezza verrà usato per decidere quando $\Gamma \stackrel{?}{\models} A$ (A è conseguenza logica di Γ), problema fondamentale della logica proposizionale e del primo ordine.
Si usa nella sua forma contronominale assieme al Lemma 1:

$$\Gamma \models A \iff \Gamma \cup \{\neg A\} \text{ insodd.} \iff \exists \Gamma' \subseteq_\omega \Gamma : \Gamma' \text{ insoddisfacibile}$$

La deduzione automatica sono infatti quei metodi refutazionali che mirano a dimostrare l'insoddisfacibilità di una teoria o di una singola formula.

Si supponga che Γ sia una teoria finita ($\Gamma = \{B_1, \cdots, B_n\}$) e che si voglia sapere se $\Gamma \stackrel{?}{\models} A$.

$$\begin{aligned} & \Gamma \models A \\ \iff & B_1, \cdots, B_n \models A \\ \iff & \{B_1, \cdots, B_n, \neg A\} \text{ insodd.} && \text{(come teoria)} \\ \iff & B_1 \wedge \cdots B_n \wedge \neg A \text{ insodd.} && \text{(come formula)} \end{aligned}$$

Decidere se la formula finale è sodd. o insodd. è NP-completo e si può parzialmente risolvere con i SAT-solver.

Se, invece, Γ è infinito non si può più risolvere la soddisfacibilità come formula. È qua che si utilizza il Teorema di Compattezza: partendo da un $\Gamma_0 = \emptyset$ si cerca il $\Gamma_i \subseteq_\omega \Gamma : \Gamma_i$ è insodd. Se si trova allora anche Γ è insodd.
Tuttavia non c'è un criterio *generale* per sapere se Γ è sodd. in tempo finito.

4.4 Equivalenza Semantica

I seguenti enunciati:

$$A \vee B \qquad\qquad B \vee A$$

sono due formule *sintatticamente* differenti perché sono stringhe di simboli scritte diversamente. Il loro **significato**, tuttavia, è uguale e, pertanto, per ogni $v : \mathcal{L} \to \{0,1\}$ si ha che:

$$v(A \vee B) \equiv v(B \vee A)$$

Quella appena definita è una **relazione**.

Definizione (Relazione n-aria)**.** $R \subseteq S^n$ è una relazione n-aria su un insieme S.

È un sottoinsieme dell'insieme di tutte le n-ple di elementi di S, in altre parole $R \subseteq S^n$.
Per esempio, una relazione binaria è una relazione R su S tale che $R \subseteq S^2 = S \times S$.

Definizione (Relazione di equivalenza)**.** R è una **relazione d'equivalenza** sse:

- R è una relazione binaria: $R \subseteq S^2$
- R è riflessiva: $\forall s \in S, (s,s) \in R$
- R è simmetrica: $\forall s_1, s_2 \in S, (s_1, s_2) \in R \to (s_2, s_1) \in R$
- R è transitiva: $\forall s_1, s_2, s_3 \in S, (s_1, s_2) \in R \wedge (s_2, s_3) \in R \to (s_1, s_3) \in R$

4.4.1 Partizioni e classi di equivalenza

Sia R una relazione di equivalenza su S.

Definizione (Classe di equivalenza)**.** La *classe di equivalenza* di $s \in S$ rispetto a R è:

$$[s]_R = \{t \in S : (s,t) \in R\}$$

Osservazione. $(s,t) \in R \iff [s]_R = [t]_R$

Definizione (Partizione)**.** $P(S) = \{B_1, \cdots, B_k, \cdots\}$ con i *blocchi* $B_i \subseteq S$, tali che:

- $B_i \cap B_j = \emptyset$ per ogni $i, j \in I$, dove I è l'insieme di indici di B
- $\cup_{i \in I} B_i = S$

I due concetti, partizioni e relazioni di equivalenza, sono legati tra di loro:

- ogni relazione di equivalenza R su S determina una partizione P_R di S composta dalle sue classi di equivalenza.
$$P_R = \{[s]_R : s \in S\}$$
- ogni partizione $P(S)$ composta da blocchi B_i determina una relazione d'equivalenza R_P su S:
$$\forall (s,t) \in S^2,\ (s,t) \in R_P \iff \exists! i \in I : s,t \in B_i$$
che è chiaramente riflessiva, simmetrica e transitiva.

4.4.2 Equivalenza Logica

Definizione (Equivalenza Logica)**.** Relazione $\equiv \subseteq {\mathcal{F}_\mathcal{L}}^2$ tale che:

$$\forall (A,B) \in {\mathcal{F}_\mathcal{L}}^2,\ A \equiv B \iff \forall v : \mathcal{L} \to \{0,1\},\ \widetilde{v}(A) = v(B)$$

Osservazione. $\equiv$ è una relazione di *equivalenza*

Dimostrazione. Siano $A, B, C \in \mathcal{F}_\mathcal{L}$:

- $\equiv$ è riflessiva: $A \equiv A$, infatti $\forall v : \mathcal{L} \to \{0,1\},\ \widetilde{v}(A) = \widetilde{v}(A)$
- $\equiv$ è simmetrica: $A \equiv B \to B \equiv A$, infatti $\forall v : \mathcal{L} \to \{0,1\},\ \widetilde{v}(A) = \widetilde{v}(B) \to \forall v,\ \widetilde{v}(B) = \widetilde{v}(A)$.
- $\equiv$ è transitiva: $A \equiv B \wedge B \equiv C \to A \equiv C$, infatti $\forall v, \widetilde{v}(A) = \widetilde{v}(B) \wedge \forall v, \widetilde{v}(B) = \widetilde{v}(C) \to \forall v, \widetilde{v}(A) = \widetilde{v}(C)$

□

Sembrerebbe che $\equiv$ catturi il significato (i.e. v) delle formule, prescindendo dalla loro sintassi!

Essendo una relazione di equivalenza, $\equiv$ può essere vista come partizione $P_\equiv$ di formule $\mathcal{F}_\mathcal{L}$ (anche scritta $\mathcal{F}_\mathcal{L}/_\equiv$) e perciò:

$$\mathcal{F}_\mathcal{L}/_\equiv = \{[A]_\equiv : A \in \mathcal{F}_\mathcal{L}\} \qquad \text{per 4.4.1}$$

ovvero è l'insieme delle classi di equivalenza delle formule rispetto a $\equiv$.
Prossimamente doteremo $\mathcal{F}_\mathcal{L}/_\equiv$ di operazioni tali da renderla una *struttura algebrica*.

4.4.3 Sostituzione di lettera proposizionale con Formula

Definizione (Sostituzione di lettera proposizionale con formula)**.** Siano $A, B \in \mathcal{F}_{\mathcal{L}}$ delle formule e sia $p \in \mathcal{L}$ una lettera proposizionale. Si definisce la formula $A[B/p]$ ottenuta dalla formula A sostituendo induttivamente ogni occorrenza della lettera proposizionale p con la formula B:

- **base**: $A = q$, con $q \in \mathcal{L}$

$$q[B/p] := \begin{cases} B & \text{se } p = q \\ q & \text{se } p \neq q \end{cases}$$

- **passo**:

  - caso $A = \neg A_1$:
    $(\neg A_1)[B/p] := \neg(A_1[B/p])$
  - caso $A = (A_1 \wedge A_2)$:
    $(A_1 \wedge A_2)[B/p] := (A_1[B/p]) \wedge (A_2[B/p])$
  - caso $A = (A_1 \vee A_2)$:
    $(A_1 \vee A_2)[B/p] := (A_1[B/p]) \vee (A_2[B/p])$
  - caso $A = (A_1 \to A_2)$:
    $(A_1 \to A_2)[B/p] := (A_1[B/p] \to A_2[B/p])$

Per esempio, $(p \wedge \neg q)[(t \to q)/p] = (t \to q) \wedge \neg q$.

Lemma 3 (di Sostituzione)**.** *Siano* $\mathfrak{v} : \mathcal{L} \to \{0,1\}$ *e* $S, T \in \mathcal{F}_{\mathcal{L}}$.
$\mathfrak{v}(S) = \mathfrak{v}(T) \implies \forall A \in \mathcal{F}_{\mathcal{L}}$ *e* $\forall p \in \mathcal{L},\ \mathfrak{v}(A[S/p]) = \mathfrak{v}(A[T/p])$

ossia, se si sostituisce nella stessa formula una lettera proposizionale con due enunciati che hanno lo stesso valore di verità, il valore delle due sostituzioni è uguale.

Dimostrazione. Per induzione strutturale su A:

- **base**: se $A = q$, con $q \in \mathcal{L}$:

$$\begin{cases} \text{se } q = p & \widetilde{\mathfrak{v}}(q[S/p]) = \widetilde{\mathfrak{v}}(S) = \widetilde{\mathfrak{v}}(T) = \widetilde{\mathfrak{v}}(q[T/p]) \\ \text{se } q \neq p & \widetilde{\mathfrak{v}}(q[S/p]) = \widetilde{\mathfrak{v}}(q) = \widetilde{\mathfrak{v}}(q[T/p]) \end{cases}$$

- **passo**:

  - caso $A = \neg B$:

$$\begin{aligned} \widetilde{\mathfrak{v}}(B[S/p]) &= \widetilde{\mathfrak{v}}(B[T/p]) && \text{per ipotesi induttiva} \\ \neg\widetilde{\mathfrak{v}}(B[S/p]) &= \neg\widetilde{\mathfrak{v}}(B[T/p]) && \text{per def. } I_{\neg} \end{aligned}$$

– caso $A = (B \wedge C)$:

$$\begin{aligned}
&\widetilde{v}((B \wedge C)[S/p]) \\
&= \widetilde{v}(B[S/p] \wedge C[S/p]) && \text{per def. di sostituzione 4.4.3} \\
&= \min\{\widetilde{v}(B[S/p]),\ \widetilde{v}(C[S/p])\} && \text{per } I_\wedge \\
&\widetilde{v}((B \wedge C)[T/p]) \\
&= \widetilde{v}(B[T/p] \wedge C[T/p]) && \text{per def. di sostituzione 4.4.3} \\
&= \min\{\widetilde{v}(B[T/p]),\ \widetilde{v}(C[T/p])\} && \text{per } I_\wedge \\
&\left.\begin{aligned}\widetilde{v}(B[S/p]) &= \widetilde{v}(B[T/p]) \\ \widetilde{v}(C[S/p]) &= \widetilde{v}(C[T/p])\end{aligned}\right\} && \text{per ipotesi induttiva} \\
&\Longrightarrow \min\{\widetilde{v}(B[S/p]),\ \widetilde{v}(C[S/p])\} \\
&= \min\{\widetilde{v}(B[T/p]),\ \widetilde{v}(C[T/p])\}
\end{aligned}$$

– Analogamente gli altri operatori binari

□

Teorema (di Sostituzione)**.** *Siano* $S, T \in \mathcal{F}_\mathcal{L}$*,*
$S \equiv T \implies \forall A \in \mathcal{F}_\mathcal{L}$ *e* $p \in \mathcal{L}$, $A[S/p] \equiv A[T/p]$

A differenza del Lemma 3, il Teorema di Sostituzione utilizza l'equivalenza al posto dell'assegnamento.

Dimostrazione.

$$\begin{aligned}
&S \equiv T \\
&\Longrightarrow \forall v : \mathcal{L} \to \{0,1\},\ v(S) = v(T) && \text{per def. } \equiv\ (4.4.2) \\
&\Longrightarrow \forall v : \mathcal{L} \to \{0,1\},\ v(A[S/p]) = v(A[T/p]) && \text{per Lemma di Sostituzione } (3) \\
&\Longrightarrow A[S/p] \equiv A[T/p] && \text{per def. } \equiv
\end{aligned}$$

□

4.4.4 Connettivi aggiuntivi

Doppia implicazione

$$F \leftrightarrow G \equiv (F \to G) \wedge (G \to F)$$

Due formule sono logicamente equivalenti sse la loro coimplicazione è vera sotto ogni assegnamento:

$$\models (F \leftrightarrow G) \iff F \equiv G$$

Costanti logiche Le costanti logiche (o *connettivi zerari*) possono essere derivate da altre formule:

$$\begin{aligned}
\bot &:= p \wedge \neg p && \text{(o qualsiasi altra contraddizione)} \\
\top &:= p \to p && \text{(o qualsiasi altra tautologia)}
\end{aligned}$$

o primitive:

$$\bot := I_\bot : \{0,1\}^0 \to \{0,1\}, \text{ la funzione vuota che mappa in } 0$$
$$\top := I_\top : \{0,1\}^0 \to \{0,1\}, \text{ la funzione vuota che mappa in } 1$$

Il che vuol dire che $\forall \mathfrak{v} : \mathcal{L} \to \{0,1\}$:

$$\widetilde{\mathfrak{v}}(\bot) = 0 \qquad \text{e} \qquad \widetilde{\mathfrak{v}}(\top) = 1$$

4.4.5 Equivalenze logiche notevoli

Date $F, G, H \in \mathcal{F}_\mathcal{L}$ le seguenti sono equivalenze logiche:

- Idempotenza:

$$F \vee F \equiv F \qquad F \wedge F \equiv F$$

- Commutatività:

$$F \vee G \equiv G \vee F \qquad F \wedge G \equiv F \wedge F$$

- Associatività:

$$F \vee (G \vee H) \equiv (F \vee G) \vee H \qquad F \wedge (G \wedge H) \equiv (F \wedge G) \wedge H$$

- Assorbimento:

$$F \vee (F \wedge G) \equiv F \qquad F \wedge (F \vee G) \equiv F$$

- Distributività:

$$F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H) \qquad F \wedge (G \vee H) \equiv (F \wedge G) \vee (F \wedge H)$$

- Doppia negazione:

$$\neg\neg F \equiv F$$

- Leggi di De Morgan:

$$\neg(F \vee G) \equiv \neg F \wedge \neg G \qquad \neg(F \wedge G) \equiv \neg F \vee \neg G$$

- Interdefinibilità:

$$F \to G \equiv (\neg F) \vee G \qquad F \to G \equiv \neg(F \wedge \neg G)$$
$$F \vee G \equiv (\neg F) \to G \qquad F \wedge G \equiv \neg(F \to G)$$
$$F \vee G \equiv \neg(\neg F \wedge \neg G) \qquad F \wedge G \equiv \neg(\neg F \vee \neg G)$$

- Costanti nelle implicazioni:

$$\neg A \equiv A \to \bot \qquad A \equiv \top \to A$$

- Complemento:

$$A \vee \neg A \equiv \top \qquad\qquad A \wedge \neg A \equiv \bot$$

- Elementi neutri:

$$A \vee \bot \equiv A \qquad\qquad A \wedge \top \equiv A$$

- Unità:

$$A \vee \top \equiv \top \qquad\qquad A \wedge \bot \equiv \bot$$

- De Morgan su n formule:

$$\neg(F_1 \vee F_2 \vee \cdots F_n) \equiv \neg F_1 \wedge \neg F_2 \wedge \cdots \neg F_n$$
$$\neg(F_1 \wedge F_2 \wedge \cdots F_n) \equiv \neg F_1 \vee \neg F_2 \vee \cdots \neg F_n$$

- Distributibilità generalizzata:

$$(F_1 \vee \cdots \vee F_u) \wedge (G_1 \vee \cdots \vee G_n) \equiv \bigvee_{i=1}^{u} \bigvee_{j=1}^{v} (F_i \wedge G_j)$$
$$(F_1 \wedge \cdots \wedge F_u) \vee (G_1 \wedge \cdots \wedge G_n) \equiv \bigwedge_{i=1}^{u} \bigwedge_{j=1}^{v} (F_i \wedge G_j)$$

Esercizio Provare l'idempotenza tramite l'assorbimento.

Dimostrazione. Sia $p \in \mathcal{L}$ con $p \notin F$:

$$(F \vee p)[F/p] = F \vee F \qquad (F \vee p)[F \wedge (F \vee F)/p] = F \vee (F \wedge (F \vee F)) \tag{4.1}$$

Ma:

$$\begin{aligned}
& F \equiv F \wedge (F \vee F) && \text{per assorbimento 4.4.5} \\
\Longrightarrow\ & (F \vee p)[F/p] \equiv (F \vee p)[F \wedge (F \vee F)/p] && \text{per Thm di Sostituzione} \\
\Longrightarrow\ & F \vee F \equiv F \vee (F \wedge (F \vee F)) && \text{per 4.1} \\
\Longrightarrow\ & F \vee F \equiv F && \text{per assorbimento 4.4.5}
\end{aligned}$$

□

Tutte le leggi che abbiamo indicato discendono dalle quattro leggi di **commutatività**, **associatività**, **assorbimento**, **distrubtività** e **complemento**, pertanto sarebbe stato abbastanza postulare queste quattro leggi e le rimanenti sarebbero "autonomamente" verificate.

4.4.6 Osservazioni su $\equiv$

Per inquadrare meglio questo fatto, facciamo delle osservazioni finali su $\equiv$.

Osservazione. Se A e B sono due enunciati e $A \equiv B$ e $\models A$, allora $\models B$.

Una conseguenza di ciò è che tutte le tautologie e, analogamente le contraddizioni, sono tra di loro logicamente equivalenti. C'è, quindi, una classe di equivalenza delle tautologie e una delle contraddizioni.

Osservazione. Se $A \equiv B$ e A è (in)sodd., allora B è (in)sodd.

Osservazione. Se $A \models B$ e $B \models A$ allora $A \equiv B$.

Questo si può vedere grazie al Lemma 2 di Deduzione: $\models A \to B$ e $\models B \to A$ e, grazie alla definizione di coimplicazione (4.4.4), si ha $A \equiv B$.

L'osservazione più importante è la seguente:

Osservazione. L'equivalenza logica è più di una relazione di equivalenza rispetto ad un'equivalenza semplice: infatti, $\equiv$ è una **congruenza** rispetto ai connettivi usati come *operazioni* (e.g. $\wedge : \mathcal{F}_{\mathcal{L}} \times \mathcal{F}_{\mathcal{L}} \to \mathcal{F}_{\mathcal{L}}$).

Per esempio, per la congruenza rispetto a $\wedge$, date quattro formule $A, B, C, D \in \mathcal{F}_{\mathcal{L}}$ tali che $A \equiv B$ e $C \equiv D$, si ha che $A \wedge C \equiv B \wedge D$. Analogamente accade per ogni altro connettivo.

Questa proprietà non è ovvia! Vi sono, infatti, situazioni in cui questo non accade anche per una relazione d'equivalenza, la quale non è automaticamente anche una congruenza. Per esempio, si prenda come relazione d'equivalenza quella tale per cui ogni numero dispari appartiene allo stesso blocco e ogni numero pari appartiene al proprio singleton e si consideri come operazione la somma dei numeri naturali. In altre parole, si ha che, fissata la relazione R, $[3]_R = \{1,3,5,7,9,\cdots\}$, $[2]_R = \{2\}$, $[4]_R = \{4\}$ eccetera. Questa è una relazione d'equivalenza poiché è una partizione di $\mathbb{N}$, ma non è una congruenza rispetto alla somma dei naturali: sommando, per esempio, $1+1$ il risultato ricade in $[2]$, mentre sommando $3+1$ il risultato ricade in $[4]$, anche se $1 \in [3]$ appartengono alla stessa classe d'equivalenza.

4.5 Algebrizzabilità

Il fatto che l'equivalenza logica sia una congruenza, è la chiave per definire l'**algebrizzabilità** della Logica Booleana, la quale sarà utilizzata per definire le *Forme Normali* in seguito. Prima di introdurre formalmente questo concetto, è necessario introdurre un concetto molto importante, ossia le **relazioni d'ordine**.

4.5.1 Relazioni d'ordine (parziale)

Definizione (Relazione d'ordine)**.** Una **relazione d'ordine** R è una relazione binaria $R \subseteq S^2$ che soddisfa le tre proprietà di:

- Riflessività: $\forall s \in S,\ (s,s) \in R$
- Antisimmetria: $\forall s,t \in S,\ (s,t) \in R$ e $(t,s) \in R \implies s = t$
- Transitività: $s,t,r \in S,\ (s,t) \in R$ e $(t,r) \in R \implies (s,r) \in R$

Spesso, per indicare questa relazione si utilizza il simbolo $\leq$, a prescindere dal fatto che si indichi effettivamente il senso di ordinamento che si intende solitamente. Le relazioni d'ordine possono essere totali o parziali: nel primo caso, dati qualunque due elementi dell'insieme di appartenenza sussiste necessariamente una relazione tra i due. Nel secondo caso, invece, questo non è necessariamente vero.

Un esempio di relazione d'ordine *totale* è $(\mathbb{N}, \leq)$, ossia $a \leq b$ se esiste $c \in \mathbb{N}$ tale che $a + c = b$; questo ordine è totale poiché dati due numeri naturali qualunque si può sempre definire un ordine tra di essi.

Un esempio di relazione *parziale* è $(\mathbb{N}, |)$ ossia si indica che sussiste la relazione tra due numeri a, b dicendo $a \leq b$ se $a|b$, ossia a divide b ed esiste c tale che $a \cdot c = b$; questa relazione è decisamente diversa.

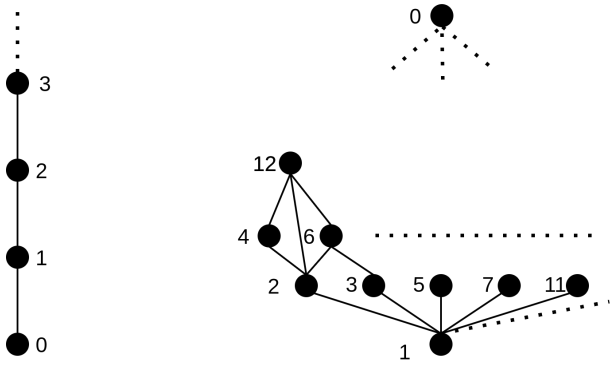

Figura 4.1: Due relazioni d'ordine.

Nella Figura 4.1 si può vedere una rappresentazione (tramite reticoli, definiti a seguire) delle due relazioni d'ordine. La prima, quella totale, è rappresentata a destra: si può notare come sia "lineare" a confronto della seconda, parziale, che è rappresentata a sinistra: quest'ultima infatti si dirama: alla base ha 1, il numero che divide tutti gli altri; al primo "strato" ha tutti i numeri primi, il secondo i multipli dei multipli dei numeri primi (che saranno comunque collegati ai numeri primi, in quanto saranno divisibili anche per essi) e, commettendo un abuso che solitamente viene concesso, in cima vi è il numero 0 che è divisibile da tutti gli altri, benché non sia divisibile per sé stesso.

Reticoli

Un insieme P con una relazione d'ordine (parziale) $\leq$, notato $(P, \leq)$ è detto insieme parzialmente ordinato o, in inglese, partially ordered set (*poset*). L'insieme delle classi di equivalenza delle formule è un insieme parzialmente ordinato con delle peculiarità.

Tra tutti i *poset*, ci interessano quelli con alcune particolari proprietà strutturali, chiamati **reticoli**.

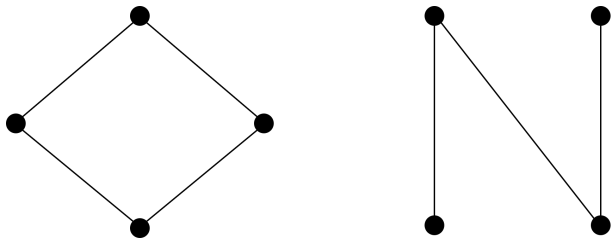

Figura 4.2: Due reticoli.

La Figura 4.2 riporta due immagini. Non entrambi sono reticoli: quello più a destra, infatti, non lo è. Cosa li distingue? La proprietà strutturale (più importante per i nostri scopi) che li divide, è che per quello più a sinistra, il reticolo, data ogni qualsiasi coppia di elementi possibile, si può sempre trovare quale dei due sia il minimo elemento che è più grande della coppia e il massimo elemento che è più piccolo della coppia. Questo non avviene nell'altro caso, infatti i due elementi "in alto", non hanno un minimo elemento più grande di loro, mentre i due elementi "in basso" non hanno un massimo elemento più piccolo di loro. In altre parole, non hanno **estremi inferiori** ed **estremi superiori**, che sono invece la proprietà fondamentale dei reticoli.

Si definiscono quindi questi due concetti.

Definizione (Estremo inferiore)**.** Dati due elementi x, y appartenenti ad un insieme parzialmente ordinato $(P, \leq)$, si definisce

$$\inf(\{x,y\}) = \max\{z \in (P,\leq) : z \leq x \text{ e } z \leq y\}$$

e analogamente si definisce

Definizione (Estremo superiore)**.** Dati due elementi x, y appartenti ad un insieme parzialmente ordinato $(P, \leq)$, si definisce

$$\sup(\{x,y\}) = \min\{z \in (P,\leq) : x \leq z \text{ e } y \leq z\}$$

Si possono generalizzare inf() e sup() da un sottoinsieme $S \subseteq P$, con P poset $(P, \leq)$:

$$\inf(S) = \max\{z \in (P,\leq) : z \leq s \ \forall s \in S\} \qquad \text{max dei minoranti di } S$$
$$\sup(S) = \min\{z \in (P,\leq) : s \leq z \ \forall s \in S\} \qquad \text{min dei maggioranti di } S$$

A questo punto si può definire il minimo di un sottoinsieme (e analogamente il massimo), se esiste, come:

$$\min(S) := z \in S : z = \inf(S) \qquad \max(S) := z \in S : z = \sup(S)$$

Si può ora definire formalmente un reticolo:

Definizione (Reticolo come poset)**.** Un reticolo è un insieme parzialmente ordinato $(R, \leq)$: $\forall x, y \in R$ esistono $\inf(\{x,y\})$ e $\sup(\{x,y\})$.

Dopo aver definito i reticoli, si definisce un'ulteriore struttura, necessaria per concludere l'argomento dell'Algebrizzazione della Logica Booleana.

4.5.2 Struttura Algebrica

Definizione. Dato un insieme S e le operazioni $*_1, *_2, \cdots, *_n$ delle operazioni su S, allora $(S, *_1, *_2, \cdots, *_n)$ è una **struttura algebrica**.

Definizione. Visto come una struttura algebrica, un reticolo è $(R, \sqcup, \sqcap)$, con $\sqcup, \sqcap$ operazioni $\sqcup : R^2 \to R$ e $\sqcap : R^2 \to R$, tale che le due operazioni siano commutative, associative e valga l'assorbimento.

Lemma. *Ogni reticolo visto come insieme parzialmente ordinato è anche una struttura algebrica*

$$(R,\leq) \iff (R,\sqcup,\sqcap)$$

Dimostrazione. ($\Longrightarrow$) con $R \equiv R$, $\forall a,b \in R$, siano:

$$a \sqcap b := \inf\{a,b\} \qquad\qquad a \sqcup b := \sup\{a,b\}$$

visto che si può verificare che per inf() e sup() valgono le proprietà di commutatività, assorbimento e associatività.

($\Longleftarrow$) con $R \equiv R$ definiamo $\forall a,b \in R$:

$$a \leq b \text{ sse } a = a \sqcap b \qquad\qquad b \leq a \text{ sse } a = a \sqcup b$$

$(R,\leq)$ è quindi un poset, infatti si può verificare che $\leq$ è riflessiva, antisimmetrica e transitiva (4.5.1).
$(R,\leq)$ è anche un reticolo perché $\forall a,b \in R$ esiste inf e sup: sia $c \leq a$ e $c \leq b$

$$\begin{aligned}
& c \leq a \text{ e } c \leq b \\
\iff & c = c \sqcap a \text{ e } c = c \sqcap b && \text{per def. } \leq \\
\iff & c = (c \sqcap b) \sqcap a \\
\iff & c = c \sqcap (b \sqcap a) && \text{per associatività} \\
\iff & c \leq b \sqcap a && \text{per def. } \leq \\
\iff & c \leq \inf\{b,a\} && \text{per def. inf()}
\end{aligned}$$

Per il sup si ragiona in modo analogo. □

4.5.3 Definizione di Algebra Booleana

A questo punto, siamo finalmente pronti per definire cosa sia l'Algebra Booleana.

Definizione (Algebra Booleana)**.** L'Algebra Booleana è un sistema $(S,\sqcap,\sqcup,0,1,()^c)$ di operazioni tali che $(S,\sqcup,\sqcap)$ sia un reticolo limitato, complementato, vivo:

- **limitato**: $\forall a \in S$:

$$a \sqcap 0 = a \qquad (a \geq 0)$$
$$a \sqcap 1 = a \qquad (a \leq 0)$$

- **complementato**: $\forall a \in S$

$$a \sqcap a^c = 0 \qquad\qquad a \sqcup a^c = 1$$

- **distributivo**: $\forall a,b,c \in S$:

$$a \sqcap (b \sqcup c) = (a \sqcap b) \sqcup (a \sqcap c) \qquad\qquad a \sqcup (b \sqcap c) = (a \sqcup b) \sqcap (a \sqcup c)$$

Un esempio è il seguente: dato un insieme A si consideri $(P(A),\cup,\cap,\emptyset,A,()^c)$ e si ha $(P(A),\subseteq)$ tale che $A \subseteq B \iff A = A \cap B$.

Esercizio Si dimostri che le seguenti sistemi siano un'algebra booleana:

- $(F(A), \cup, \cap, \emptyset, A, ()^c)$
  con A un insieme infinito, e $F(A)$ l'insieme dei suoi sottoinsiemi finiti o complementi di insiemi finiti
- $(\{0,1\}, \min, \max, 0, 1, 1 - \_)$

Algebra di Lindenbaum

L'algebra delle formule, chiamata anche **Algebra di Lindenbaum delle Formule**, è definita come

$$(\mathcal{F}_\mathcal{L}/_\equiv, \wedge, \vee, \bot, \top, \neg)$$

Dato che $\equiv$ è una congruenza, si possono definire le seguenti operazioni con $A, B \in \mathcal{F}_\mathcal{L}$:

$$\begin{aligned} [A]_\equiv \wedge [B]_\equiv &= [A \wedge B]_\equiv \\ [A]_\equiv \vee [B]_\equiv &= [A \vee B]_\equiv \\ [A]_\equiv \to [B]_\equiv &= [A \to B]_\equiv \\ \neg [A]_\equiv &= [\neg A]_\equiv \\ [\top]_\equiv &= \text{ tautologie} \\ [\bot]_\equiv &= \text{ contraddizioni} \end{aligned}$$

La cardinalità dell'insieme delle classi di equivalenza scrivibili con n variabili è:

$$|\mathcal{F}_\mathcal{L}{}^{(n)}/\equiv| = 2^{2^n} = |\{F : \{0,1\}^n \to \{0,1\}\}| = |\mathcal{P}(\{1, \cdots, 2^n\})|$$

4.5.4 Funzioni Termine

Al fine di arrivare ad un discorso completo sulle Forme Normali, si continua a ragionare sulle classi d'equivalenza delle formule e la loro struttura, ora da un punto di vista che sembra essere diverso ma che in conclusione si riunirà con quanto detto precedentemente.

Si rifletta sul significato di

$$\models F \iff \forall v : \mathcal{L} \to \{0,1\},\ v(F) = 1 \tag{4.2}$$

in termini "computazionali" significa *tenere fermo* l'argomento (F) e far cambiare le funzioni. Sarebbe comodo scambiare i ruoli: la formula si comporta come una funzione che prende come argomenti gli assegnamenti.

Siano $A \in \mathcal{F}_\mathcal{L}$ una formula e $n \in \mathbb{N}$. Le lettere proposizionali che occorrono in A si indicano con

$$Var(A) \subseteq \mathcal{L} \subseteq \{p_1, p_2, \cdots, p_n\}$$

Sia ora $\hat{A}$ una **funzione termine** $\hat{A} : \{0,1\}^n \to \{0,1\}$ tale che:

$$\forall v : \mathcal{L} \to \{0,1\},\ \hat{A}(v(p_1), v(p_2), \cdots, v(p_n)) = \widetilde{v}(A)$$

Definizione (Funziona Termine)**.** Definizione di $\hat{A} : \{0,1\}^n \to \{0,1\}$ induttiva su A:

- **base**: Se $A = p_i$, con $p_i \in \{p_1, p_2, \cdots, p_n\}$ un letterale:

$$\hat{A} := \hat{p}_i : \{0,1\}^n \to \{0,1\}$$

con $\hat{p}_i(t_1, \cdots, t_n) = t_i$, l'$i$-esima proiezione. Quindi $\forall \nu : \mathcal{L} \to \{0,1\}$ si verifica che:

$$\hat{A} = \hat{p}_i(\nu(p_1), \cdots, \nu(p_n)) = \nu(p_i) = \widetilde{\nu}(p_i) = \widetilde{\nu}(A)$$

- **passo**: $\forall (t_1, \cdots, t_n) \in \{0,1\}^n$
  - Se $A = \neg B$: $\hat{A}(t_1, \cdots, t_n) = 1 - \hat{B}(t_1, \cdots, t_n)$
  - Se $A = B \wedge C$: $\hat{A}(t_1, \cdots, t_n) = \min\{\hat{B}(t_1, \cdots, t_n),\ \hat{C}(t_1, \cdots, t_n)\}$
  - Se $A = B \vee C$: $\hat{A}(t_1, \cdots, t_n) = \max\{\hat{B}(t_1, \cdots, t_n),\ \hat{C}(t_1, \cdots, t_n)\}$
  - Se $A = B \to C$: $\hat{A}(t_1, \cdots, t_n) = \max\{1 - \hat{B}(t_1, \cdots, t_n),\ \hat{C}(t_1, \cdots, t_n)\}$

Ora si possono vedere come *funzioni termine* la riflessione iniziale (4.2):

$$\models F \iff \hat{F}(t_1, \cdots, t_n) = 1,\ \forall (t_1, \cdots, t_n)$$

e l'equivalenza logica:

$$[F]_{\equiv} = [G]_{\equiv} \iff F \equiv G \iff \hat{F}(t_1, \cdots, t_n) = \hat{G}(t_1, \cdots, t_n),\ \forall t_i \in \{0,1\}^n$$

Allo stesso modo si può dare una definizione diversa di Algebra Booleana, *isomorfa* all'Algebra di Lindenbaum.

Definizione (Algebra Booleana su Funzioni Termine sui primi n termini)**.**

$$\hat{\mathcal{F}}_{\mathcal{L}}^{(n)} = (\{\hat{A} : \{0,1\}^n \to \{0,1\}, A \in \mathcal{F}_{\mathcal{L}}\}, \wedge, \vee, \neg, \bot, \top)$$

dove i significati degli operatori sono quelli espressi nella definizione di funzioni termine.

4.5.5 Teorema di Completezza Funzionale

Concentriamoci sull'algebra $\hat{\mathcal{F}}_{\mathcal{L}}^{(n)}$ appena espressa e una $Funz^{(n)}$ definita come:

$$Funz^{(n)} = (\{f : \{0,1\}^{(n)} \to \{0,1\}\}, \wedge, \vee, \neg, \bot, \top)$$

Che rapporto sussiste tra le due Algebre? Per ogni $n \in \mathbb{N}$, la prima ha come universo l'insieme di tutte le Funzioni Termine per ogni formula $A \in \mathcal{F}_{\mathcal{L}}$, mentre la seconda ha come universo tutte le funzioni $f : \{0,1\}^n \to \{0,1\}$.
Sicuramente, data la restrizione, si può affermare che:

$$\hat{\mathcal{F}}_{\mathcal{L}}^{(n)} \subseteq Funz^{(n)}$$

Si dimostra, ora, che vale anche:

$$\hat{\mathcal{F}}_{\mathcal{L}}^{(n)} \supseteq Funz^{(n)}$$

Questo fatto prende il nome di **Teorema di Completezza Funzionale**. A parole, ciò che afferma è che ogni funzione $f : \{0,1\}^n \to \{0,1\}$ è esprimibile come una formula $F \in \mathcal{F}_{\mathcal{L}}$.
Chiaramente, se dimostrato, vuol dire che le due algebre sono in realtà *isomorfe*.

Teorema (di Completezza Funzionale)**.**

$$\hat{F}_L^{(n)} \supseteq Funz^{(n)},\ \forall n \in \mathbb{N}$$

La dimostrazione consiste nel fornire per ogni $n \in \mathbb{N}$ e ogni funzione $f : \{0,1\}^n \to \{0,1\}$ una formula *costruibile* $F \in \mathcal{F}_{\mathcal{L}}{}^{(n)} : f = \hat{F}$.

Dimostrazione. Sia $f : \{0,1\}^n \to \{0,1\}$ una funzione, la cui cardinalità del dominio è 2^n. Per ogni combinazione degli n argomenti $(a_{i_1}, a_{i_2}, \cdots, a_{i_n})$ esiste un valore $b_i \in \{0,1\}$ associato:

$$f := \begin{cases} (a_{1_1}, a_{1_2}, \cdots, a_{1_j}, \cdots, a_{1_n}) & \longrightarrow b_1 \\ (a_{2_1}, a_{2_2}, \cdots, a_{2_j}, \cdots, a_{2_n}) & \longrightarrow b_2 \\ \quad\vdots & \quad\vdots \\ (a_{i_1}, a_{i_2}, \cdots, a_{i_j}, \cdots, a_{i_n}) & \longrightarrow b_i \\ \quad\vdots & \quad\vdots \\ (a_{(2^n)_1}, a_{(2^n)_2}, \cdots, a_{(2^n)_j}, \cdots, a_{(2^n)_n}) & \longrightarrow b_{(2^n)} \end{cases} \tag{4.3}$$

Costruiamo una formula $F \in \mathcal{F}_{\mathcal{L}}$ in modo tale che $\hat{F} = f$.

- se $f(t_1, \cdots, t_n) = 0$ per ogni $(t_1, \cdots, t_n)$: $F = \bot$
- altrimenti sia $I \subseteq \{1, 2, \cdots, 2^n\}$ tale che:

  $$i \in I \iff b_i = 1$$

  (intuitivamente, I contiene i numeri delle "righe" della tabella di verità della funzione f in cui l'immagine è il numero 1).
  Per ogni $i \in I$ si considera la i-esima tupla $(a_{i_1}, a_{i_2}, \cdots, a_{i_n})$ di argomenti di f.
  Per ogni $j \in \{1, \cdots, n\}$ si definisce:

  $$A_{ij} = \begin{cases} p_j & \text{se } a_{ij} = 1 \\ \neg p_j & \text{se } a_{ij} = 0 \end{cases} \qquad \text{con } A_{ij} \in \mathcal{F}_{\mathcal{L}}{}^{(n)}$$

  Si pone $F = \bigvee_{i \in I} \bigwedge_{j \in n} A_{ij}$

  Si mostra che $\hat{F} = f$, ovvero che per ogni riga $i \in I$:

  $$[\bigwedge_{j=1}^{n} \hat{A}_{ij}(t_1, \cdots, t_n)] = 1 \iff (t_1, \cdots, t_n) = (a_{i_1}, \cdots, a_{i_n})$$

  Infatti

  – se $(t_1, \cdots, t_n) = (a_{i_1}, \cdots, a_{i_n})$, $\forall j \leq n$:

  $$\hat{A}_{ij}(t_1, \cdots, t_n) = t_j = \begin{cases} a_{i_j} & \text{se } a_{i_j} = 1 \\ 1 - a_{i_j} & \text{se } a_{i_j} = 0 \end{cases} = 1$$

  – se $(t_1, \cdots, t_n) \neq (a_{i_1}, \cdots, a_{i_n})$, $\exists t_j \neq a_{i_j}$:

  $$\hat{A}_{ij}(t_1, \cdots, t_n) = t_j = \begin{cases} 1 - a_{i_j} & \text{se } a_{i_j} = 1 \\ a_{i_j} & \text{se } a_{i_j} = 0 \end{cases} = 0$$

Quindi $(A_{i_1} \wedge A_{i_2} \wedge \cdots \wedge A_{i_n})$ può essere resa vera solo da un assegnamento. In definitiva,

$$\hat{F} = \bigvee_{i \in I} \bigwedge_{j}^{n} \hat{A}_{i_j}(t_1, \cdots, t_n) = 1 \iff (t_1, \cdots, t_n) \text{ è la riga } i\text{-esima, e } i \in I$$

Quindi $\hat{F} = f$.

□

Considerazioni sul Teorema di Completezza Funzionale

Per come è stato provato il Teorema di Completezza Funzionale, la funzione termine si può anche esprimere come:

$$G = \bigwedge_{i \in J} \bigvee_{j=1}^{n} B_{ij}$$

con

$$J \subseteq \{0, \cdots, 2^n - 1\} \qquad i \in J \iff b_j = 0 \qquad B_{ij} = \begin{cases} \neg p_j & a_{ij} = 1 \\ p_j & a_{ij} = 0 \end{cases}$$

Si ha, quindi, che $\hat{F} = f = \hat{G}$[2], il che è una dimostrazione che una qualsiasi formula può essere messa in una **forma normale** perché $\hat{\mathcal{F}}_{\mathcal{L}}^{(n)} = Funz^{(n)}$.

Sebbene il Teorema possa sembrare banale, in quanto la sua dimostrazione è semplice, il suo contenuto non è nullo né scontato. Un esempio di mancanza funzionale è già presente nella Logica Proposizionale stessa, nonostante la dimostrazione precedente, rimuovendo il vincolo su n e definendo

$$Funz^{\omega} := \{f : \{0,1\}^{\omega} \to \{0,1\}\}$$

con $\omega = |\mathbb{N}|$, in altre parole l'insieme di tutte le successioni infinite di valori $\{0,1\}$. Se valesse il Teorema di Completezza Funzionale anche in questo caso, si dovrebbe poter affermare che tale insieme è incluso nelle classi di equivalenza scritte nelle classi di equivalenza delle formule scritte con ω variabili:

$$Funz^{\omega} \subseteq \hat{\mathcal{F}}_{\mathcal{L}}^{\omega} = {\mathcal{F}_{\mathcal{L}}}^{\omega}/_{\equiv}$$

Ma in realtà

$$|Funz^{\omega}| = |\mathbb{R}| \geq |\mathbb{N}| = |{\mathcal{F}_{\mathcal{L}}}^{\omega}/_{\equiv}|$$

e pertanto non è valido.

Forme Normali Canoniche Le forme normali che abbiamo trovato non sono desiderabili per gli scopi futuri, in quanto sono molto prolisse: ognuno degli A_{ij} menziona tutte le lettere proposizionali $p_1, \cdots, p_n$ (analogamente per le forme congiuntive) ed è infatti chiamata Forma Normale Disgiuntiva (or di and) **Canonica** o **Completa**.
Si noti la prolissità nel costruire una $F \in \mathcal{F}_{\mathcal{L}}$ t.c. $\hat{F} = f : \{0,1\}^3 \to \{0,1\}$ con:

$$f(t_1, t_2, t_3) = t_1 \quad \forall (t_1, t_2, t_3)$$

È chiaro che la formula più "corta" è $F = t_1$, ma con la forma canonica si rappresenta come:

$$F = (p_1 \wedge \neg p_2 \wedge \neg p_3) \vee (p_1 \wedge \neg p_2 \wedge p_3) \vee (p_1 \wedge p_2 \wedge \neg p_3) \vee (p_1 \wedge p_2 \wedge p_3)$$

[2]Ciò che è stato fatto è esattamente uguale alla costruzione dei minterm e maxterm per una data tavola di verità.

Insieme funzionalmente completo di connettivi C'è un'altra nozione, legata alla Completezza Funzionale, che è molto interessante, ossia la nozione di *insieme funzionalmente completo di connettivi*.

Definizione (Insieme funzionalmente completo di connettivi)**.** Un insieme

$$C \subseteq \{\wedge, \vee, \rightarrow, \neg, \leftrightarrow, \bot, \top\}$$

è definito *funzionalmente completo* se per ogni $F \in \mathcal{F}_{\mathcal{L}}$ esiste una $G \in \mathcal{F}_{\mathcal{L}}$ scritta usando solo i connettivi in C tale che $F \equiv G$ e $\hat{F} = \hat{G}$, ossia $[F]_{\equiv} = [G]_{\equiv}$.

Un insieme di connettivi funzionalmente completo è dato dal Teorema di Completezza Funzionale stesso, ossia $C = \{\wedge, \vee, \neg\}$. Si può mostrare facilmente che anche $\{\neg, \wedge\}$ e $\{\neg, \vee\}$ sono completi (in quanto, grazie alle leggi di De Morgan, $(A \vee B) = \neg(\neg A \wedge \neg B)$). Questi non sono gli unici insiemi funzionalmente completi: $\{\neg, \rightarrow\}$ (in quanto $(A \wedge B) \equiv \neg(A \rightarrow \neg B)$ e $(A \vee B) \equiv ((\neg A) \rightarrow B)$ e $\{\rightarrow, \bot\}$. In particolare, un insieme composto da un solo connettivo, il `nand`, è anch'esso funzionalmente completo, in quanto ogni altro connettivo è ricostruibile utilizzando solo

$$A \mathbin{\overline{\wedge}} B = \neg(A \wedge B)$$

infatti $\neg A \equiv A \mathbin{\overline{\wedge}} A$ e, ovviamente, $A \wedge B \equiv \neg(A \mathbin{\overline{\wedge}} B)$ per definizione.

4.6 Forme Normali

Dopo averle introdotte col Teorema di Completezza Funzionale, possiamo passare ad un esame più approfondito delle Forme Normali. Le Forme Normali Congiuntive e Disgiuntive Canoniche sono esageratamente prolisse. Sarebbe utile conservare l'idea delle Forme Normali con formule più succinte.

4.6.1 Forma Normale Negata

Una terza forma normale, chiamata **Negation Normal Form** (NNF) che, per definizione, è l'insieme delle formule che rispettano le seguenti proprietà:

- non contiene $\rightarrow$
- $\neg$ occorre solo applicato a lettere proposizionali

quindi, per esempio

$$p_1 \wedge \neg(p_2 \vee p_3) \notin NNF$$
$$p_1 \wedge (\neg p_2 \vee \neg p_3) \in NNF$$

Definizione (Sottoformula)**.** G è sottoformula di F sse G occorre in ogni $\mathcal{L}$-costruzione di F e si indica con $G \preccurlyeq F$

Lemma. *Ogni formula $F \in \mathcal{F}_{\mathcal{L}}$ è equivalente a una $F^N \in NNF$.*

Dimostrazione. Per provare il lemma, descriviamo come trasformare F in F^N in un numero finito di passi, dove ogni passo produce una formula equivalente alla precedente. Per ottenere la sequenza si applica ad ogni passo una delle seguenti trasformazioni a $G \preccurlyeq F$:

1. $C \to D \rightsquigarrow (\neg C) \vee D$
2. $\neg\neg C \rightsquigarrow C$
3. $\neg(C \vee D) \rightsquigarrow \neg C \wedge \neg D$
4. $\neg(C \wedge D) \rightsquigarrow \neg C \vee \neg D$

Un'osservazione preliminare su questo algoritmo è che si può mostrare che applicando le quattro regole in un ordine qualsiasi si ottiene sempre $F^N \in NNF$. Si mostrerà che partendo da (1) e applicando le altre trasformazioni si possa arrivare a F^N: sia data $F \in \mathcal{F}_\mathcal{L}$. In un numero finito di applicazioni di (1) si ottiene una F' in IFNF (Implication-Free Normal Form), equivalente a F.

Si introduce una misura di *distanza*: si definisce $i(E)$ come il numero di connettivi $\to$ nella formula E. Si ha, quindi, che $E \in IFNF \iff i(E) = 0$. Inoltre, se $E \notin IFNF$, allora E' ottenuta applicando (1) su E deve avere $i(E) > i(E')$.
È chiaro che $F \equiv F'$ poiché l'applicazione di (1) è semplicemente l'applicazione di una equivalenza eseguita grazie al Teorema di Sostituzione. Quindi, in un certo numero di passi si ha

$$F = F_0 \equiv \cdots \equiv F'(\in IFNF) \equiv F_i \equiv \cdots \equiv F_n(\in NNF)$$

e si nota che le applicazioni di (2), (3) e (4) trattano tutte equivalenze logiche. Ciò che bisogna dimostrare è che la sequenza di trasformazioni termini, ossia l'algoritmo non è infinito ($n \in \mathbb{N}$).

Per dimostrare che l'algoritmo termina, siano ora $A, E \in \mathcal{F}_\mathcal{L}$ e si definisce $\ell(A) =$ "n° di occorrenze di connettivi in A". Definiamo quindi $m(E)$ come la somma del numero di connettivi presenti nelle sottoformule $\neg A$ di E:

$$m(E) = \sum_{\neg A \preccurlyeq E} \ell(A)$$

Si noti che

$$m(E) = 0 \iff E \in NNF$$

Infatti:

- se $E \in NNF$:

$$\begin{aligned} & \forall \neg A \preccurlyeq E,\ A \in \mathcal{L} && \text{per def. } NNF \\ \iff & \forall \neg A \preccurlyeq E,\ \ell(A) = 0 \\ \iff & m(E) = 0 \end{aligned}$$

- se $m(E) = 0$:

$$\sum_{\neg A \preccurlyeq E} \ell(A) = 0 \iff \forall \neg A \preccurlyeq E,\ \ell(A) = 0 \iff \forall \neg A \preccurlyeq E,\ A \in \mathcal{L}$$

In un passaggio eseguito con una delle $(2), (3), (4)$ generando

$$E \rightsquigarrow E'$$

si ha che $m(E) > m(E')$. Se si riesce a dimostrare quest'ultima affermazione, allora si dimostra automaticamente che il procedimento termina in un numero finito di passi.

(2) $\neg\neg C \rightsquigarrow C$: In questo caso

$$\begin{cases} m(\neg\neg C) & = \sum_{\neg A \preccurlyeq \neg\neg C} \ell(A) = \ell(\neg C) + \sum_{\neg A \preccurlyeq \neg C} \ell(A) = \ell(C) + 1 + \ell(C) + \sum_{\neg A \preccurlyeq C} \ell(A) = \\ & = 1 + 2 \cdot \ell(C) + \sum_{\neg A \preccurlyeq C} \ell(A) \\ m(C) & = \sum_{\neg A \preccurlyeq C} \ell(A) \end{cases}$$

$$\implies m(\neg\neg C) > m(C)$$

(3) $\neg(C \vee D) \rightsquigarrow \neg C \wedge \neg D$: In questo caso

$$\begin{cases} m(\neg(C \vee D)) = \sum_{\neg A \preccurlyeq \neg(C \vee D)} \ell(A) & = \ell(C \vee D) + \sum_{\neg A \preccurlyeq C} \ell(A) + \sum_{\neg A \preccurlyeq D} \ell(A) \\ & = \ell(C) + \ell(D) + \sum_{\neg A \preccurlyeq C} \ell(A) + \sum_{\neg A \preccurlyeq D} \ell(A) + 1 \\ m(\neg C \wedge \neg D) = \sum_{\neg A \preccurlyeq \neg C \wedge \neg D} \ell(A) & = \ell(C) + \ell(D) + \sum_{\neg A \preccurlyeq C} \ell(A) + \sum_{\neg A \preccurlyeq D} \ell(A) \end{cases}$$

$$\implies m(\neg(C \vee D)) > m(\neg C \wedge \neg D)$$

Analogamente per (4).

Data una formula $F \in \mathcal{F}_{\mathcal{L}}$, si può costruire una successione finita

$$F = F_0 \equiv \cdots \equiv F'(\in IFNF) \equiv F_i \equiv \cdots \equiv F_n(\in NNF)$$

e ovviamente $F \equiv F_n$, per un qualche $n \in \mathbb{N}$. □

4.6.2 Forma Normale Congiuntiva e Disgiuntiva

Le CNF o FNC, Forme Normali Congiuntive, sono un sovrainsieme delle CNF *complete* o canoniche.

Definizione (Letterale)**.** Un *letterale* è una formula nella forma p o $\neg p$ per qualche lettera proposizionale p (i.e., o è una lettera proposizionale o è la sua negata). La definizione di **letterale opposto** a uno dato è, dato A letterale, l'opposto di A:

$$\bar{A} = \begin{cases} \neg p & \text{se } A = p \\ p & \text{se } A = \neg p \end{cases}$$

Definizione (Clausola)**.** Una *clausola* è una disgiunzione di un numero finito k di letterali:

$$\ell_1 \vee \ell_2 \cdots \vee \ell_k$$

dove ogni ℓ_i è un letterale.

Definizione (CNF o FNC)**.** Una **forma normale congiuntiva** è una congiunzione di un numero finito h di clausole:

$$C_1 \wedge C_2 \wedge \cdots \wedge C_h$$

dove ogni C_j è una clausola.

Definizione (Clausola vuota)**.** Una **clausola vuota** □ è definita come la disgiunzione di zero letterali.

Questo concetto è importante e non va confuso col concetto di **CNF vuota** $\emptyset$, definita come la congiunzione di zero clausole. Prendendo una sequenza di letterali e formule

$$\begin{aligned} F_0 &= \square \\ F_1 &= \ell_1 \\ F_2 &= \ell_1 \vee \ell_2 \\ F_3 &= \ell_1 \vee \ell_2 \vee \ell_3 \\ &\vdots \\ F_i &= \ell_1 \vee \ell_2 \vee \ell_3 \vee \cdots \vee \ell_i \end{aligned}$$

possiamo notare che ogni assegnamento che soddisfa una formula F_i, soddisfa anche quella successiva F_{i+1}. Per questo $F_i \rightarrow F_{i+1}$ è una tautologia.
Seguendo questa definizione la clausola vuota è insoddisfacibile e pertanto è $\square \equiv \bot$.

Per l'insieme di CNF vuote, si ha

$$\begin{aligned} &\emptyset \\ &C_1 \\ &C_1 \wedge C_2 \\ &C_1 \wedge C_2 \wedge C_3 \\ &\vdots \\ &C_1 \wedge C_2 \wedge C_3 \wedge C_4 \end{aligned}$$

qui, al contrario, ogni assegnamento che soddisfa una clausola C_i, soddisfa anche la precedente C_{i-1}.
Seguendo questa definizione la CNF vuota è soddisfacibile ed è, in realtà, soddisfatta da ogni assegnamento: $\emptyset \equiv \top$.

Definizione (DNF o FND)**.** Una *forma normale disgiuntiva* è una disgiunzione di un numero finito n di formule:

$$D_1 \vee D_2 \vee \cdots \vee D_n$$

dove ogni D_i è una congiunzione chiamata **co-clausola** di un numero finito v di letterali:

$$\ell_1 \wedge \ell_2 \wedge \cdots \wedge \ell_v$$

DNF si tratta, quindi, di una struttura duale rispetto alle CNF. Si noti la terminologia speciale per le CNF: questa terminologia è riservata alle CNF in quanto la dualità tra CNF e DNF verrà spezzata.

Lemma. *Ogni formula $F \in \mathcal{F}_{\mathcal{L}}$ è logicamente equivalente a una formula $F^c \in CNF$ e a una formula $F^d \in DNF$.*

Questo è già stato dimostrato grazie al teorema di completezza funzionale, che tuttavia utilizzava le CNF e DNF complete.

esistenza DNF/CNF. Per induzione strutturale su F:

- **base**: se $F = p$ con $p \in \mathcal{L}$ allora $p = p^c = p^d$.
- **passo**:

– se $F = \neg G$, per ipotesi induttiva si ha $G^c \equiv G$ e $G^d \equiv G$. Allora, se

$$G = C_1 \wedge C_2 \wedge \cdots \wedge C_h \qquad \text{dove ogni } C_i = \ell_{i_1} \vee \cdots \vee \ell_{i_{k_i}}$$

si ha che:

$$\begin{aligned} \neg G \equiv \neg(G^c) &\equiv \neg(C_1 \wedge \cdots \wedge C_h) \\ &\equiv \neg C_1 \vee \neg C_2 \vee \cdots \vee \neg C_h := (\neg G)^d \qquad \text{per DeMorgan} \end{aligned}$$

$$\begin{aligned} \text{dove ogni } \neg C_i &\equiv \neg(\ell_1 \wedge \ell_2 \wedge \cdots \wedge \ell_v) \\ &\equiv \neg\ell_{i_1} \wedge \cdots \wedge \neg\ell_{i_{k_i}} \qquad \text{per DeMorgan} \\ &\equiv \bar{\ell}_{i_1} \wedge \cdots \wedge \bar{\ell}_{i_{k_i}} \qquad \neg\ell_{i_j} \text{ è } \equiv \text{ a un letterale opposto} \end{aligned}$$

$$\implies (\neg G)^d \in DNF$$

e similmente:

$$\begin{aligned} \neg G \equiv \neg(G^d) &\equiv \neg(D_1 \vee \cdots \vee D_n) \\ &\equiv \neg D_1 \wedge \neg D_2 \wedge \cdots \wedge \neg D_n := (\neg G)^c \qquad \text{per DeMorgan} \end{aligned}$$

$$\begin{aligned} \text{con } \neg D_i &\equiv \neg(\ell_{i_1} \wedge \cdots \wedge \ell_{i_{k_i}}) \\ &\equiv \neg\ell_{i_1} \wedge \cdots \wedge \neg\ell_{i_{v_i}} \qquad \text{per DeMorgan} \\ &\equiv \bar{\ell}_{i_1} \wedge \cdots \bar{\neg}\ell_{i_{v_i}} \qquad \neg\ell_{i_j} \text{ è } \equiv \text{ a un letterale opposto} \end{aligned}$$

$$\implies (\neg G)^c \in CNF$$

Quindi esistono sia $F^d := (\neg G)^d \equiv \neg(G^c)$ che $F^c := (\neg G)^c \equiv \neg(G^d)$.

– se $F = (G \wedge H)$, per ipotesi induttiva si ha:

$$\begin{aligned} G^c &\equiv G & G^d &\equiv G \\ H^c &\equiv H & H^d &\equiv H \end{aligned}$$

Si definisce, allora

* $F \equiv F^c$
  con $F^c := (G^c \wedge H^c)$ e, banalmente $(G^c \wedge H^c) \in CNF$
* F^d:

$$\begin{aligned} F &\equiv (G^d \wedge H^d) \\ &\equiv (G_1 \vee \cdots \vee G_u) \wedge (H_1 \vee \cdots \vee H_v) \\ &\equiv \bigvee_{i,j} (G_i \wedge H_j) \qquad \text{per Distributività Generalizzata } (4.4.5) \end{aligned}$$

Ma visto che C_i e H_j sono *co-clausole*, anche $(G_i \wedge H_j)$ lo è.

$$\implies F^d := \bigvee_{i=1}^{u} \bigvee_{j=1}^{v} (G_i \wedge H_i) \in DNF$$

– se $F = (G \vee H)$, per ipotesi induttiva si ha:

$$G^c \equiv G \qquad G^d \equiv G$$
$$H^c \equiv H \qquad H^d \equiv H$$

Si definisce, allora

* $F \equiv F^d$
  con $F^d := (G^d \vee H^d)$ e, banalmente $(G^d \vee H^d) \in DNF$
* F^c:

$$\begin{aligned} F &\equiv (G^c \vee H^c) \\ &\equiv (G_1 \wedge \cdots \wedge G_u) \vee (H_1 \wedge \cdots \wedge H_v) \\ &\equiv \bigwedge_{i,j} (G_i \vee H_j) \qquad \text{per Distributività Generalizzata } (4.4.5) \end{aligned}$$

Ma visto che C_i e H_j sono *clausole*, anche $(G_i \vee H_j)$ lo è.

$$\Longrightarrow F^c := \bigwedge_{i=1}^{u} \bigwedge_{j=1}^{v} (G_i \vee H_i) \in CNF$$

generalizzata

– se $F = (G \to H)$, ci si comporta analogamente contando che $F \equiv (\neg G \vee H)$

□

Si nota, tuttavia, che muoversi tra DNF e CNF con la distributività generalizzata causa un notevole allungamento delle formule. Per esempio:

$$\bigvee_{i=1}^{n} (p_i \wedge q_i) \equiv \bigwedge_{i=1}^{2^n} (\ell_{i_1} \vee \cdots \vee \ell_{i_n})$$

quindi c'è una dilatazione esponenziale. Questo è un bel problema.

Ci sono algoritmi più intelligenti per generare da DNF delle formule CNF più corte? Purtroppo no.

CAPITOLO 5

Complessità Computazionale e Deduzione Automatica

In questo momento, per decidere se una formula $F \in \mathcal{F}_{\mathcal{L}}$ che menziona n lettere proposizionali diverse sia soddisfacibile o meno, l'algoritmo che conosciamo computa la sua tabella di verità, analizzando quindi 2^n possibilità, ossia un numero esponenziale. qui Cominciamo definendo cosa sia un problema di decisione:

Definizione (Problema di Decisione)**.** Dato Σ^* un insieme di tutte le *parole* finite costruite coi simboli di Σ—un *alfabeto* finito—e un *Linguaggio* $L \subseteq \Sigma^*$, non necessariamente finito, il problema di decisione consiste nel verificare se una stringa finita w appartenga o meno al Linguaggio L

Ad esempio, fissato

$$\Sigma = \{\wedge, \vee, \neg, \rightarrow, (,), p, |\}$$

si possono definire alcuni problemi, per esempio

$$\mathcal{F}_{\mathcal{L}} \subseteq \Sigma^*$$

dove il problema risiede nel capire se una certa formula di lunghezza finita risiede in $\mathcal{F}_{\mathcal{L}}$ oppure no. Vi sono dei problemi principalmente di natura sintattica, come appunto $\mathcal{F}_{\mathcal{L}}$ ma anche NNF, CNF e DNF. Vi sono problemi in cui la natura semantica è decisiva, per esempio SAT, dove

$$w \in \Sigma^* : w \in SAT \iff w \in \mathcal{F}_{\mathcal{L}} \wedge w \text{ è soddisfacibile}$$

oppure $TAUTO$, dove una stringa finita appartiene a $TAUTO$ sse è una formula ed è una tautologia, ossia

$$w \in \Sigma^* : w \in TAUTO \iff w \in \mathcal{F}_{\mathcal{L}} \wedge \models w$$

e analogamente $UNSAT$, il problema di decidere se una certa formula è insoddisfacibile.

5.1 Complessità Computazionale

5.1.1 Efficienza

Alcuni dei problemi elencati precedentemente possono essere risolti (decisi) in maniera efficiente, ossia data una *parola* w decidere se appartiene ad un certo linguaggio: esempi di soluzioni efficienti sono quelle che riguardano tutti i problemi sintattici, risolvibili in tempo al massimo quadratico e in realta anche lineare con alcune tecniche di parsing.

I problemi che riguardano la semantica, invece, sono tipicamente più complessi da risolvere e infatti, per quanto conosciamo fino ad ora, si possono risolvere solo (nel caso peggiore) con un analisi dell'intera tabella di verità della formula, quindi con un numero esponenziale di calcoli. È importante rimarcare nuovamente che benché sussista questa difficoltà, verificare che un singolo assegnamento verifichi la formula è invece un calcolo semplice che richiede un tempo decisamente più contenuto rispetto al problema di decidibilità.

Quindi, vi sono dei problemi decidibili efficientemente, dei problemi decidibili molto difficilmente e verificabili facilmente e dei problemi decidibili molto difficilmente e verificabili difficilmente; da qui parte una gerarchia di classi di complessità infinita, ma a noi interesseranno solo le prime due.

Per essere più precisi, si fissa un modello astratto di computazione, che fornisca la nozione di *passo elementare*, in modo da poter ragionare precisamente sull'efficienza dei problemi. Qualunque modello di computazione che costituisca un modello realistico di calcolatore, con associata una nozione di passo elementare per costruire algoritmi, classifica i problemi nello stesso modo.

Osservazione (Tesi di Church-Turing)**.** Ogni modello ragionevole di computazione è equivalente.

Uno dei modelli di computazione è quello delle Macchine di Turing (MdT): un'idea della Macchina di Turing è immaginarla come una macchina che lavora su un nastro infinito in lettura e scrittura, mantenendo uno stato e operando su una singola porzione di nastro in lettura utilizzando un programma per muoversi tra gli stati e scrivere sul nastro.

Si può passare ora alla definizione formale delle classi dei problemi:

Definizione (Classe $\mathbb{P}$)**.** La classe dei problemi "efficientemente decidibili" è definita, con una certa ideologia sottesa, come tutti quei problemi che possono essere risolti in tempo polinomiale, ossia esiste una certa Macchina di Turing T e un polinomio $p : \mathbb{N} \to \mathbb{N}$ per i quali per ogni $w \in \Sigma^*$ la computazione della MdT sull'input w, denotato $T(w)$ termina entro $p(||w||)$ passi e per ogni $w \in L$ si ha che $T(w)$ accetta il problema e per ogni $w \notin L$ si ha che $T(w)$ non accetta, ossia risolve il problema.

Definizione (Classe $\mathbb{NP}$)**.** La classe dei problemi "verificabili efficientemente" è definita come tutti quei problemi tali per cui esiste una Macchina di Turing deterministica e due polinomi $p, q : \mathbb{N} \to \mathbb{N}$ tali che per ogni $w \in \Sigma^*$ e ogni $z \in \Gamma^*$, ossia un **certificato**, che si può immaginare prodotto da un oracolo, si ha che $T(w,z)$ termina entro $p(||w||)$ passi e si ha che per ogni $w \in L$ esiste $z \in \Gamma^*$ tale che $||z|| \leq q(||w||)$ (ossia il certificato è sufficientemente corto) e $T(w,z)$ accetta e per ogni $w \notin L$ si ha che per ogni possibile $z \in \Gamma^*$ sufficientemente corto $T(w,z)$ rifiuta, ossia "valida" il certificato.

Definizione (Riducibilità)**.** Un problema $L_1 \subseteq \Sigma^*$ è **riducibile in tempo polinomiale** a un altro problema $L_2 \subseteq \Gamma^*$ se e solo se esistono una Macchina di Turing T_{L_1,L_2} e un polinomio $p : \mathbb{N} \to \mathbb{N}$ tale che per ogni $w \in L_1$ $T(w)$ trasfroma w in $w' \in \Gamma^*$ in un numero di passi minore o uguale a $p(||w||)$.

Per indicare la relazione di riducibilità tra due problemi si indica la notazione $L_1 \preceq_p L_2$ per indicare che il primo problema è riducibile polinomialmente al secondo; la nozione di riducibilità è utile poiché rende possibile risolvere istanze del problema L_1 "riscrivendole" come se fossero istanze di L_2 (chiaramente questa utilità si verifica quando è più facile risolvere L_2 di L_1). Un esempio di riducibilità polinomiale è

$$TAUTO \preceq_p UNSAT$$

poiché la trasformazione $w \to \neg w$ è semplice e si sa che w è tautologica se e solo se $\neg w$ è insoddisfacibile.

Definizione (Problemi $\mathbb{NP}$-completi)**.** Un problema appartenente alla classe $\mathbb{NP}_c$ è un problema $L \subseteq \Sigma^*$ se e solo se

- $L \in \mathbb{NP}$
- ogni $L' \in \mathbb{NP}$ è tale che $L' \preceq_p L$

La seconda proprietà si chiama $\mathbb{NP}$-hardness.

Come corollario della definizione dei problemi $\mathbb{NP}_c$, si ha che risolvendo polinomialmente un problema di tale classe si dimostra che

$$\mathbb{P} = \mathbb{NP}$$

Teorema (di Cook-Levin)**.** *$CNFSAT \in \mathbb{NP}$-completo. ($\implies$ $SAT \in \mathbb{NP}$-completo)*

$CNFSAT \preceq_p SAT$ è vero perché una formula in CNF è comunque ancora una formula e pertanto la trasformazione—i.e., la funzione identità—è ovviamente polinomiale.
Si mostra ora che $SAT \preceq_p CNFSAT$, conservando non l'equivalenza logica ma la *relazione di equisoddisfacibilità*. Questo ci permetterà anche di trasformare polinomialmente le *DNF* in *CNF* equisoddisfacibili. Infattibile se si cercasse di mantenere l'$\equiv$.

5.1.2 Equisoddisfacibilità

Definizione (Equisoddisfacibilità)**.** Siano $A, B \in \mathcal{F}_{\mathcal{L}}$. A e B sono equisoddisfacibili (*equisodd*) sse:

$$A \; sodd. \iff B \; sodd.$$

ossia se A e B sono entrambe soddisfsacibili o entrambe insoddisfacibili.

Osservazioni sull'equisoddisfacibilità

Osservazione 1. L'equivalenza implica l'equisoddisfacibilità, ma non è vero il contrario.

1. Date le due formule $\neg(A \wedge B)$ e $\neg A \vee \neg B$, è noto che sono equivalenti grazie alle leggi di De Morgan e sono, di conseguenza, anche equisoddisfacibili.
2. $\neg(A \wedge B)$ e $(\neg A \wedge \neg B)$ non sono equivalenti, infatti l'assegnamento $A = 1$ e $B = 1$ soddisfa solo una delle due, tuttavia sono equisoddisfacibili.
3. $A \wedge \neg A$ e $\neg A \wedge \neg B$ non sono né equivalenti né equisoddisfacibili.

Osservazione 2. *equisodd.* è un tipo di relazione d'equivalenza:

- A *equisodd.* A
- A *equisodd.* B, quindi B *equisodd.* A
- A *equisodd.* B e B *equisodd.* C, quindi A *equisodd.* C

Osservazione 3. *equisodd*. non è una congruenza rispetto ai connettivi, pensati come operazioni.
Per esempio, rispetto a $\neg$:
se A *equisodd*.B con $\models A$ e $\not\models B$ (i.e. B non è tautologica), allora vuol dire che $\neg A$ **non** è *equisodd*. con $\neg B$—perché $A \in \bot$ ma B potrebbe essere soddisfacibile.

Osservazione 4. *equisodd*. è più grezza di $\equiv$, infatti le classi di equivalenza delle formule su n variabili $\mathcal{F}_{\mathcal{L}}^{(n)} / \equiv$, sono 2^{2^n}.
L'equisoddisfacibilità, invece, ha unicamente due classi: l'insieme delle formule insoddisfacibili ($[\bot]$) e tutte le rimanenti, ossia tutte quelle soddisfacibili da almeno un assegnamento.

Quest'ultima osservazione era deducibile anche dalla Osservazione 1.

5.1.3 Riduzione $SAT \preceq CNFSAT$

Sia $A \in \mathcal{F}_{\mathcal{L}}$ tale che A sia in Negation Normal Form, ossia $A \in NNF$.
Se $A \notin CNF$, allora contiene almeno una violazione, ovvero una sottoformula del tipo $C \vee (D_1 \wedge D_2)$ oppure $(D_1 \wedge D_2) \vee C$ (c'è un $\wedge$ interno ad un $\vee$).

Senza perdita di generalità trattiamo, d'ora in poi, solo il primo dei due casi. Sia $A \in NNF$, ogni sua violazione $B = C \vee (D_1 \wedge D_2)$ si sostituisce con una nuova lettera proposizionale $a \in L$ che ancora non è presente in A. Si crea quindi B' tale che:

$$B' := B'' \wedge (\neg a \vee D_1) \wedge (\neg a \vee D_2) \qquad \text{con } B'' := B[a/D_1 \wedge D_2]$$

Si osservi che $(\neg a \vee D_1) \wedge (\neg a \vee D_2) \equiv a \rightarrow (D_1 \wedge D_2)$.

Lemma. *B' e B sono equisoddisfacibili, ma in genere non sono equivalenti.*
E B' ha $2n+1$ clausole, di cui 1 di n letterali e $2n$ di 2 letterali .

Dimostrazione $B \in SAT \implies B' \in SAT$. Per ipotesi, dato che B è soddisfacibile, sia $v : \mathcal{L} \rightarrow \{0,1\}$ tale che $v(B) = 1$, ossia $v \models B$. Si definisce

$$v_a : \mathcal{L} \rightarrow \{0,1\} = \begin{cases} v_a(p) = v(p) & \forall p \in L, p \neq a \\ v_a(a) = v(D_1 \wedge D_2) \end{cases}$$

L'assegnamento v_a è ben definito, nel senso che non da alla stessa lettera proposizionale due assegnamenti diversi. Si nota che $v_a \models a \rightarrow (D_1 \wedge D_2)$, dato che per definizione $v_a(a) = v(D_1 \wedge D_2)$ e per interpretazione dell'implicazione $0 \rightarrow 0 = 1$ e $1 \rightarrow 1 = 1$; dunque:

$$v_a(B') = v_a(B'' \wedge a \rightarrow (D_1 \wedge D_2)) = v_a(B'') \wedge 1 = v_a(B'') \tag{5.1}$$

B e B' si possono riscrivere come:

$$\begin{aligned} B &= B''[D_1 \wedge D_2/a] && \text{dato che } a \text{ non appare in } B \\ B'' &= B''[a/a] \end{aligned}$$

e grazie al Lemma 3 di Sostituzione si può affermare che $v(B) = v(B'')$ in quanto $v(a) = v(D_1 \wedge D_2)$.

Quindi:

$$\begin{aligned} 1 &= v(B) && \text{ipotesi} \\ &= v_a(B) && a \text{ non occore in } B \\ &= v_a(B'') && \text{per il Lemma di Sost.} \\ &= v_a(B') && \text{per 5.1} \end{aligned}$$

ossia l'assegnamento che soddisfa B soddisfa anche B', ossia sono equisoddisfacibili. □

Dimostrazione $B' \in SAT \implies B \in SAT$. Se B' è soddisfacibile da un assegnamento della forma v_a—ossia tale che $v_a(a) = v(D_1 \wedge D_2)$—allora si può tranquillamente ribaltare la catena di uguaglianze precedente:

$$\begin{aligned} v_a \models B' \text{ sse } 1 &= v_a(B') \\ &= v_a(B'') && \text{per 5.1} \\ &= v_a(B) && \text{per il Lemma di Sost.} \\ \text{sse } v_a &\models B \end{aligned}$$

Si supponga, ora, che B' sia soddisfatto solo da assegnamenti w che non sono della forma v_a, ossia $w(a) \neq w(D_1 \wedge D_2)$. Vi sono allora due casi:

- $w(a) = 0 \neq w(D_1 \wedge D_2) = 1$
- $w(a) = 1 \neq w(D_1 \wedge D_2) = 0$

Tuttavia quest'ultimo caso è impossibile, poiché $w(a \to D_1 \wedge D_2) = 1 \to 0 = 0 = w(B')$ e quindi w non soddisferebbe B'.

Quindi rimane il primo caso e bisogna mostrare che effettivamente soddisfa B' e non porta ad un assurdo

Osservazione. Sia E un'espressione formata solo da $0, 1, \wedge, \vee$—interpretabili come min e max. Il valore di E, quindi, è o 0 o 1.
Sia E' ottenuta da E rimpiazzando nessuna o più occorrenze del simbolo 0 con 1.
Allora $E \leq E'$.

Questo perché $0, 1, \min, \max$ sono tutte funzioni non decrescenti, e E ed E' sono composizioni di funzioni non decrescenti, pertanto sono non decrescenti neanche loro.
(un'altra prova è per induzione strutturale su E).

Ora, tornando al problema principale, si ha che:

$$w(B') = 1 \qquad w(a) = 0 \qquad w(D_1 \wedge D_2) = 1$$

Se $Var(B) = \{p_1, \cdots p_n\}$, allora $w(B)$ e $w(B'')$ si possono esprimere come funzioni termine di B'':

$$\begin{aligned} w(B'') &= \hat{B}''(w(p_1), \cdots, w(p_n), w(a)) \\ w(B) &= \hat{B}''(w(p_1), \cdots, w(p_n), w(D_1 \wedge D_2)) && \text{per def. } B'' \end{aligned}$$

Dato che la formula iniziale A era in NNF, anche B'', B' e B lo sono, e $w(B'')$ e $w(B)$ sono quindi considerabili come espressioni costruite su $0, 1, \wedge, \vee$, per l'osservazione precedente.
Si può concludere, quindi, che

$$w(B'') = \hat{B}''(w(p_1), \cdots, w(p_n), 0) \leq \hat{B}''(w(p_1), \cdots, w(p_n), 1) = w(B)$$

ma $w(B'') = 1$, quindi $w(B) = 1$ □

Osservazione (Nota finale)**.** Si sarebbe potuto definire B', alternativamente, come segue:

$$\begin{aligned} B' &:= B[a/D_1 \wedge D_2] \wedge (a \leftrightarrow (D_1 \wedge D_2))^c \\ &= B[a/D_1 \wedge D_2] \wedge (\neg a \vee D_1) \wedge (\neg a \vee D_2) \wedge ((D_1 \wedge D_2) \rightarrow a)^c \\ &= B[a/D_1 \wedge D_2] \wedge (\neg a \vee D_1) \wedge (\neg a \vee D_2) \wedge (a \vee \neg D_1 \vee \neg D_2) \end{aligned}$$

Nella prova di $B' \in SAT \implies B \in SAT$ si sarebbe potuto mostrare che B' è soddisfatto solo da assegnamenti nella forma di v_a tali che $v_a(a) = v(D_1 \wedge D_2)$.

Dilatazione della riduzione *NNF-CNF* Nel passaggio da B a B' si osserva una dilatazione nella lunghezza della formula, ossia B' è *più lungo* di B; data la formula generale

$$B' := B[a/D_1 \wedge D_2] \wedge (\neg a \vee D_1) \wedge (\neg a \vee D_2)$$

La parte che subisce la sostituzione può essersi accorciata, a cui però si aggiungono una decina di simboli. Quindi $||B'|| \leq ||B|| + k$, con k una costante dipendente dal modo in cui si conta la lunghezza (si contano le parentesi come simboli eccetera).
Per ogni "violazione" B nella formula originale A, la formula risultante equisoddisfacibile senza violazioni è più lunga di $k \cdot v$ caratteri, con $v =$ #violazioni.
Inoltre, visto che in A non può essere più lunga di v, la formula risultante A' avrà lunghezza $||A'|| \leq ||A|| + k \cdot v \leq (k+1) \cdot ||A||$, una dilatazione *lineare*.

La tecnica usata per ridurre $SAT \preceq_p CNFSAT$ che consiste nel rimpiazzare B con B' equisoddisfacibile è ispirata alla riduzione

$$SAT \preceq_p 3CNFSAT$$

dovuta a Karp. Il passaggio $B \implies B'$ è un esempio del cosiddetto "Tseytin's Trick", che si usa per ridurre $F \in \mathcal{F}_{\mathcal{L}}$ a una in $3CNFSAT$ ad essa equisoddisfacibile.

Definizione (Problemi $3CNFSAT$)**.** $3CNFSAT \subseteq CNFSAT \subseteq \mathcal{F}_{\mathcal{L}}$ è costituito da tutte e sole le CNF dove ogni clausola contiene esattamente 3 letterali.
$3CNFSAT \in \mathbb{NP}$-completo.

Osservazione (Problemi $2CNFSAT$)**.** $2CNFSAT \in \mathbb{P}$.

Algorithm 1 Algoritmo di riduzione a CNF equisoddisfacibili

Require: $F \in \mathcal{F}_{\mathcal{L}}$
Ensure: $A \in CNF : F \; equisodd. \; A$
  $A : A \in NNF \wedge A \equiv F$
  while $A \notin CNF$ **do**
    $B \leftarrow$ violazione in A
    $B' \leftarrow B[a/D_1 \wedge D_2] \wedge (\neg a \vee D_1) \wedge (\neg a \vee D_2)$ con $a \notin Var(B)$
    $A \leftarrow A[B/B']$
  end while
  return A

Esempi

1 Sia $A := p \vee (q \wedge r)$ in NNF. Si trasforma ora in una formula equisoddisfacibile in CNF:

$$A' := (p \vee a) \wedge (\neg a \vee q) \wedge (\neg a \vee r)$$

Queste due formule non sono equivalenti, infatti vi sono assegnamenti che portano a risultati diversi; tuttavia sono equisoddisfacibili.

2 Sia $A := (p_1 \wedge p_2) \vee (p_2 \wedge q_2) \vee (p_3 \wedge q_3)$. Applicando iteramente la sostituzione, si ha:

$$(a_1 \vee (p_2 \wedge q_2) \vee (p_3 \wedge q_3)) \wedge (\neg a_1 \vee p_1) \wedge (\neg a_1 \vee q_1)$$
$$(a_1 \vee a_2 \vee (p_3 \wedge q_3)) \wedge (\neg a_1 \vee p_1) \wedge (\neg a_1 \vee q_1) \wedge (\neg a_2 \vee p_2) \wedge (\neg a_2 \vee p_2)$$
$$(a_1 \vee a_2 \vee a_3) \wedge (\neg a_1 \vee p_1) \wedge (\neg a_1 \vee q_1) \wedge (\neg a_2 \vee p_2) \wedge (\neg a_2 \vee q_2) \wedge (\neg a_3 \vee p_3) \wedge (\neg a_3 \vee q_3)$$

Data una formula con n letterali si generano $2 \cdot n + 1$ clausole, di cui una con n letterali e $2n$ con due letterali, laddove utilizzando la distributività se ne generavano 2^n ognuna con n letterali.

5.2 Deduzione Automatica

Ricordiamo che noi vorremmo studiare $\Gamma \overset{?}{\models} A$, che si può ricondurre ad un problema di soddisfacibilità o al suo complementare di insoddisfacibilità. Tuttavia:

$$\mathbb{NP}\text{-complete} \ni SAT \preceq_p CNFSAT$$
$$\text{co}\mathbb{NP}\text{-complete} \ni TAUTO \preceq_p SAT^c \preceq_p CNFUNSAT$$

Il motivo per cui non studiamo invece $DNFSAT$ e $DNFUNSAT$—che sono in $\mathbb{P}$—è che $SAT \not\preceq_p DNFSAT$. Non c'è un algoritmo per "tradurre" una formula arbitraria equisoddisfacibile in DNF in modo che sia sufficientemente corta: i metodi conosciuti allungano esponenzialmente la formula.

Definizione (Variazione notazionale)**.** Date $F_i \in \mathcal{F}_\mathcal{L}$, le formule

$$F_1 \wedge F_2 \wedge \cdots \wedge F_k \qquad\qquad F_1 \vee F_2 \vee \cdots \vee F_k$$

si possono scrivere senza parentesi grazie all'associatività, si possono scambiare le formule interne nelle formule esterne ($F_1 \wedge F_2 \equiv F_2 \wedge F_1$) grazie alla commutatività e, infine, si possono espandere le singole formule ($F_1 \wedge F_1 \equiv F_1$) grazie all'idempotenza.

Per questo motivo d'ora in poi per le CNF tralasceremo gli operatori $\wedge$ e $\vee$ in favore di una notazione insiemistica. Per esempio:

$$(p \vee q \vee \neg r) \wedge (q \vee r \vee a) \wedge p$$
$$\Downarrow$$
$$\{\{p, q, \neg r\}, \{q, r, a\}, \{p\}\}$$

Una teoria costituita da CNF è indicata come l'insieme di tutte le clausole appartenenti a qualche CNF della teoria (di fatto è una CNF più grande). Inoltre:

$\emptyset = CNF$ vuota
$\square =$ clausola vuota
$\{\cdots, \square, \cdots\} = CNF$ contenente la clausola vuota.

Il problema di definire se A è conseguenza logica di una teoria Γ

$$\Gamma \stackrel{?}{\models} A$$

si può ridurre a calcolare la insoddisfacibilità di $\Gamma \cup \{\neg A\}$, che è uguale a calcolare l'insoddisfacibilità della sua forma in CNF $S := \{\Gamma^c \cup \{\neg A\}^c\}$ dove $\Gamma^c := \{\gamma^c | \gamma \in \Gamma\}$ e $\{\neg A\}^c \in CNF$.

Il problema $\Gamma \models A$ è stato ridotto polinomialmente al problema $S \in CNFSAT$, dove S è un insieme di clausole considerate come insiemi di letterali, eventualmente infinito.

5.2.1 Metodi refutazionali

S è soddisfacibile se esiste $\nu : \mathcal{L} \to \{0,1\}$ tale che per ogni clausola $C \in S, \nu \models C$, in altre parole esiste un letterale $\ell \in C$ tale che $\nu \models \ell$.

Se l'obiettivo è dimostrare che S è insoddisfacibile, una strategia può essere ampliare S in un nuovo insieme $S \subseteq S'$ in modo tale che S' è logicamente equivalente o almeno equisoddisfacibile a S. Ad esempio, se ampliando iterativamente S in S', S'' fino a $S^{(u)}$ e alla fine la clausola vuota $\square$ appartiene a $S^{(u)}$ allora quest'ultimo è insoddisfacibile, e quindi anche S lo è. Questa idea può essere sfruttata disegnando **metodi refutazionali**, ossia metodi che hanno l'obiettivo di provare l'insoddisfacibilità di un insieme di clausole, in questo frangente, ma più in generale anche di una formula o una teoria. Questi metodi sono basati sulla regola di inferenza chiamata **principio di risoluzione**, il quale è utile per un tipo di calcolo particolarmente adatto a essere automatizzato, ossia SAT solver e Theorem Prover.

Definizione (Principio di Risoluzione)**.** Date due clausole C_1 e C_2 si dice che D è la **risolvente** di C_1 e C_2 sul pivot ℓ se e solo se

- $\ell \in C_1$
- $\bar{\ell} \in C_2$
- $D := (C_1 \setminus \{\ell\}) \cup (C_2 \setminus \{\bar{\ell}\})$

e si scriverà $D = \mathbb{R}(C_1, C_2; \ell, \bar{\ell})$.

Esercizio

Mostrare che $(C_1 \setminus \{\ell\}) \cup (C_2 \setminus \{\bar{\ell}\})$ può essere diverso da $(C_1 \cup C_2) \setminus \{\ell, \bar{\ell}\}$.

Sia $C_1 = \{a, \neg a\}$ e $C_2 = \{a, \neg a, b\}$. Allora

$$(C_1 \setminus \{a\}) \cup (C_2 \setminus \{\neg a\}) = \{a, \neg a, b\}$$
$$(C_1 \cup C_2) \setminus \{a, \neg a\} = \{b\}$$

Esempio

Sia $C_1 := \{x, y, \neg t\}$ e $C_2 := \{u, \neg y, t\}$; si ha

$$D_1 = \mathbb{R}(C_1, C_2; y, \neg y) = \{x, \neg t, u\}$$
$$D_2 = \mathbb{R}(C_1, C_2; \neg t, t) = \{x, y, u, \neg y\}$$

Lemma 4 (di correttezza della risoluzione)**.** *Sia* $D = \mathbb{R}(C_1, C_2; \ell, \bar{\ell})$*, allora*

$$\{C_1, C_2\} \models D$$

Dimostrazione. Dimostriamo che $v \models C_1$ e $v \models C_2 \implies \models D$.
Sia $v : \mathcal{L} \to \{0,1\}$ tale che:

$$\begin{aligned}
& v \models C_1 \text{ e } v \models C_2 \\
\implies & \exists m \in C_1, n \in C_2 \text{ t.c. } v \models m \text{ e } v \models n \\
& \text{Per assurdo assumiamo } m = \ell \text{ e } n = \bar{\ell} \\
\implies & v(\ell) = 1 \text{ e } v(\bar{\ell}) = 1 \\
\implies & \bot \\
\implies & m \neq \ell \text{ o } n \neq \bar{\ell} \\
\implies & m \in D \text{ o } n \in D \\
\implies & v(D) = 1
\end{aligned}$$

□

Corollario. *Se* $\mathbb{R}(C_1, C_2; \ell, \bar{\ell}) = \square$ *allora* $\{C_1, C_2\}$ *è insoddisfacibile, in quanto* $\{C_1, C_2\} \models \square$.

Definizione (*Calcolo refutazionale basato su risoluzione*)**.** Sia S un insieme di clausole e sia $S := S_0, S_1, \cdots, S_k$ una successione di insiemi tale che per ogni $i = 0, \cdots, k-1$ si ha che S_{i+1} si ottiene unendo a S_i una o più risolventi di clausole in S_i.
Se $\square \in S_k$, allora S è insoddisfacibile.

L'obiettivo è mostrare che i calcoli refutazionali basati su applicazioni ripetute della risoluzione sono corretti e completi (refutazionalmente). In genere, per un calcolo logico si studiano infatti le due seguenti proprietà:

Definizione (Correttezza)**.** Un calcolo si definisce **corretto** se i certificati che produce testimoniano il vero, ossia sono corretti, in altre parole non produce certificati fasulli.

Per esempio, nel frangente attuale $S_0, S_1, \cdots, S_k \ni \square$ è un certificato corretto dell'insoddisfacibilità di $S = S_0$.
Questo genere di calcolo è corretto, e la dimostrazione scende direttamente dal Lemma 4, come vedremo tra poco.

Definizione (Completezza)**.** Un calcolo è completo se non omette alcun certificato.

Nella situazione attuale vuol dire che se S è insoddisfacibile, allora esiste $S_0, S_1, \cdots, S_k$ tale che si produce $\square \in S_k$.

Prima di mostrare che il calcolo refutazionale è completo, è necessario prepararsi la strada, poiché non sarà banale.

Teorema (Teorema di Completezza del principio di risoluzione (o di Robinson))**.** *Un insieme S di clausole è insoddisfacibile* $\iff \square \in \mathcal{R}^*(S)$*, dove* $\mathcal{R}^*(S)$ *è definito come segue:*

$$\begin{aligned}
\mathcal{R}^0(S) &:= S \\
\mathcal{R}^1(S) &:= S \cup \{D : D = \mathbb{R}(C_1, C_2; \ell, \bar{\ell})\} \qquad \textit{per qualche } C_1, C_2 \in S \textit{ e } \ell \in C_1 \textit{ e } \bar{\ell} \in C_2 \\
\mathcal{R}^2(S) &:= \mathcal{R}(\mathcal{R}(S)) \\
&\cdots \\
\mathcal{R}^{t+1}(S) &:= \mathcal{R}(\mathcal{R}^t(S)) \\
&\cdots \\
\mathcal{R}^*(S) &:= \bigcup_{i \in \omega} \mathcal{R}^i(S)
\end{aligned}$$

Il calcolo $\mathcal{R}^*$ è un metodo *a forza bruta*, applica il Principio di Risoluzione calcolando ciecamente tutte le risoluzioni possibili. Per questo non c'è da sperare che faccia molto meglio delle tavole di verità.

Dimostrazione. Correttezza del calcolo $\mathcal{R}$*:* $\square \in \mathcal{R}^*(S) \implies S$ *insoddisfacibile*

$$\begin{aligned}
& \square \in \mathcal{R}^*(S) \\
\iff & \exists i \in \omega : \square \in \mathcal{R}^i(S) && \text{per def. } \mathcal{R}^*(S) \\
\iff & \mathcal{R}^i(S) \textit{ insodd}. && \text{per def. di clausola vuota} \\
\iff & \mathcal{R}^i(S) \equiv \mathcal{R}^{i-1}(S) \equiv \cdots \equiv \mathcal{R}^0(S) = S && \text{per Lemma 4} \\
\iff & S \textit{ insodd}.
\end{aligned}$$

□

Dimostrazione. Completezza Refutazionale del calcolo $\mathcal{R}$*: S insodd.* $\implies \square \in \mathcal{R}^*(S)$
Dato che S è insoddisfacibile, per il Teorema 1 di Compattezza esiste un $S_{fin} \subseteq_\omega S$ finito e S_{fin} è insoddisfacibile; bisogna capire come sia fatto S_{fin}: l'insieme finito S_{fin} contiene un numero finito di lettere proposizionali $Var(S_{fin}) \subseteq \{p_1, p_2, \cdots, p_n\}$ per qualche $n \in \omega$, $p_i \in \mathcal{L}$.
La notazione $C^n_{\mathcal{L}}$ indica l'insieme di tutte le clausole scrivibili sulle prime n lettere proposizionali.

Osservazione. $S_{fin} \subseteq C^n_{\mathcal{L}}$ e $C^0_{\mathcal{L}} = \{\square\}$

A fortiori: dato che $S_{fin} \subseteq C^n_{\mathcal{L}}$, si ha $S_{fin} \subseteq (C^n_{\mathcal{L}} \cap S) \subseteq (C^n_{\mathcal{L}} \cap \mathcal{R}^*(S))$ e pertanto $C^n_{\mathcal{L}} \cap \mathcal{R}^*(S)$ è insodd., dato che S_{fin} è insodd.
Dimostreremo, quindi, che $C^k_{\mathcal{L}} \cap \mathcal{R}^*(S)$ è insodd. per ogni $k = n, \cdots, 1, 0$. Compreso, per $k = 0$:

$$C^0_{\mathcal{L}} \cap \mathcal{R}^*(S) \text{ insodd.}$$

Tuttavia dobbiamo capire com'è fatto $C^0_{\mathcal{L}} \cap \mathcal{R}^*(S)$:

$$C^0_{\mathcal{L}} \cap \mathcal{R}^*(S) = \begin{cases} \emptyset & \rightsquigarrow C^0_{\mathcal{L}} \cap \mathcal{R}^*(S) \text{ sodd.} \rightsquigarrow \perp \\ \{\square\} & \rightsquigarrow \text{ unico caso possibile} \end{cases}$$

Ma quindi, per def. di $\cap$, $\square \in \mathcal{R}^*(S)$!

Per dimostrare che per ogni $k = n, \cdots, 1, 0$

$$C^k_{\mathcal{L}} \cap \mathcal{R}^*(S) \text{ è insoddisfacibile}$$

si usa l'induzione decrescente su k:

- **base** ($k = n$): $C^n_{\mathcal{L}} \cap \mathcal{R}^*(S)$ insodd. (già dimostrato)
- **passo** ($k-1$): si assume l'asserto vero per $n, n-1, \cdots, k$ e si dimostra per $k-1$, ossia

$$C^n_{\mathcal{L}} \cap \mathcal{R}^*(S), C^{n-1}_{\mathcal{L}} \cap \mathcal{R}^*(S), \cdots, C^k_{\mathcal{L}} \cap \mathcal{R}^*(S) \text{ insodd.}$$

Per assurdo, si assuma $C^{k-1}_{\mathcal{L}} \cap \mathcal{R}^*(S)$ sodd., dunque esiste $v \models C$ per ogni $C \in C^k_{\mathcal{L}} \cap \mathcal{R}^*(S)$; si definiscono ora

$$v^+ : \mathcal{L} \to \{0,1\},\ v^+(p_k) = 1$$
$$v^- : \mathcal{L} \to \{0,1\},\ v^-(p_k) = 0$$

mentre per ogni altra $p_i \in C^k_{\mathcal{L}}$, $v(p_i) = v^+(p_i) = v^-(p_i)$; si noti che $p_k \notin C^{k-1}_{\mathcal{L}}$. Per ipotesi induttiva esistono $C_1, C_2 \in C^k_{\mathcal{L}} \cap \mathcal{R}^*(S)$ insoddisfacibili, tali che:

$$v^+(C_1) = 0 \qquad\qquad v^-(C_2) = 0$$

La lettera proposizionale p_k occorre in C_1, altrimenti vorrebbe dire che $C_1 \in C^{k-1}_{\mathcal{L}} \cap \mathcal{R}^*(S)$ e, visto che $C^{k-1}_{\mathcal{L}} \cap \mathcal{R}^*(S)$ è sodd. per ipotesi assurda, vorrebbe dire che $v^+(C_1) = v(C_1) = 1$. Assurdo.
Deve però apparire come letterale $\neg p_k$, così che $v^+(C_1) = 0$ come da ipotesi.
Analogamente, p_k occorre in C_2 e più precisamente come letterale positivo p_k e la prova è identica, *mutantis mutandis*.

Dunque, esiste $D = \mathbb{R}(C_1, C_2; \neg p_k, p_k) = (C_1 \setminus \{\neg p_k\}) \cup (C_2 \setminus \{p_k\})$, con:

$$\begin{aligned} & p_k, \neg p_k \notin D \\ \iff & D \in C^{k-1}_{\mathcal{L}} \cap \mathcal{R}^*(S) && \text{perché non compare } p_k \\ \iff & v(D) = 1 && \text{per ipotesi assurda} \\ \iff & v \models D \end{aligned}$$

.

Vi sono due casi:

1. v soddisfa qualche letterale in $C_1 \setminus \{\neg p_k\}$. Ma allora $v(C_1) = 1$ e $v^+(C_1) = 1$, assurdo.
2. v soddisfa qualche letterale in $C_2 \setminus \{p_k\}$, e allora $v^-(C_2) = 1$, assurdo.

Non essendoci altri casi, si è raggiunta la contraddizione che conclude la dimostrazione per assurdo, dunque $C^k_{\mathcal{L}} \cap \mathcal{R}^*(S)$ è insoddisfacibile per ogni k, $k = 0$ incluso.

Dunque $C^0_{\mathcal{L}} \cap \mathcal{R}^*(S)$ è insoddisfacibile, pertanto $C^0 \cap \mathcal{R}^*(S) = \{\square\}$ e $\square \in \mathcal{R}^*(S)$. $\square$

Osservazione. Se S è un insieme finito di clausole, la costruzione di $\mathcal{R}^*(S)$ costituisce una *procedura di decisione*, cioè sia che S sia insodd. sia che S sia sodd. termina in tempo finito, dando la risposta corretta.

Dimostrazione. Infatti, essendo S finito, vengono usate solo un numero finito $n = |Var(S)|$ di lettere proposizionali diverse, dunque:

- letterali scrivibili su $\{p_1, \cdots, p_n\}$: $2 \cdot n$

- letterali diversi che occorrono: $\leq 2 \cdot n$
- clasuole scrivibili su $2 \cdot n$ letterali: $\leq 2^{2 \cdot n}$

In S, quindi, occorrono al più $2^{2 \cdot n}$ clausole.
Si nota, ora, che la risoluzione non introduce mai nuovi letterali, dunque la sequenza $\mathcal{R}^0(S) \subseteq R^1(S) \subseteq \cdots \mathcal{R}^k(S) \subseteq \cdots$ è tale che che ogni $\mathcal{R}^i(S)$ è un sottoinsieme delle $2^{2 \cdot n}$ clausole scrivibili su $\{p_1, \cdots, p_n\}$.
Dunque esiste un $t \in \omega$ tale che $\mathcal{R}^t(S) = \mathcal{R}^{t+1}(S)$, poiché prima o poi tutte le clausole saranno contenute; pertanto

$$\mathcal{R}^t(S) = \mathcal{R}^{t+1}(S) = \cdots = \mathcal{R}^*(S)$$

Se si trova $\square \in \mathcal{R}^i(S)$, si può concludere che S sia insoddisfacibile.
Se, al contrario, $\mathcal{R}^t(S) = \mathcal{R}^{t+1}(S)$ e $\square \notin \mathcal{R}^t(S)$, si può concludere che S sia soddisfacibile perché non ci sono due clausole con due letterali opposti. $\square$

Osservazione. Se, invece, S è un infinito di clausole, dato un qualsiasi $S_{fin} \subseteq_\omega S$ esiste $t \in \omega$ tale che $\mathcal{R}^t(S_{fin}) = \mathcal{R}^{t+1}(S_{fin}) = \mathcal{R}^*(S_{fin})$, tuttavia questo non fornisce una procedura di decisione per S, ma esclusivamente di *semidicesione*. Questo perché in genere non si sa "scegliere" S_{fin} e il meglio che si può fare è, presa una successione infinita di sottoinsiemi infiniti

$$S_1 \subseteq S_2 \subseteq \cdots \subseteq S_k \subseteq \cdots \text{ t.c. } \bigcup_i S_i = S$$

e calcolare $\mathcal{R}^*(S_i)$ per ogni i: se $\square \in S_i$ allora S insoddisfacibile, altrimenti si aumenta i e si procede al passo successivo.

La Completezza Refutazionale non è la Completezza *tout-court*. È qualcosa che è più debole, e in realtà proprio per questo è una proprietà desiderabile dal punto di vista computazionale.

Definizione (Deduzione per Risoluzione)**.** Una deduzione per risoluzione di una clausola C da un insieme di clausole S (indicata con $S \vdash_R C$) è una sequenza finita di clausole $C_1, C_2, \cdots, C_n$ tale che:

- $C_n = C$
- $\forall C_i,\ C_i \in S$ oppure $C_i = \mathbb{R}(C_j, C_k, \ell, \bar{\ell})$, con $j, k < i$

In particolare, una deduzione per risoluzione della clausola vuota ($S \vdash_R \square$) è detta **refutazione** di S.

Teorema 2 (di Completezza Refutazionale)**.** *Un insieme di clausole S è insodd. sse* $S \vdash_R \square$.

Al momento, la refutazione la sappiamo costruire solo tramite il metodo $\mathcal{R}^*(S)$.

Esempio

$$S = \{\{a, b, \neg c\}, \{a, b, c\}, \{a, \neg b\}\} \overset{?}{\models} \{a\}$$

Tuttavia non sappiamo risolvere questo problema direttamente, quindi si trasforma il problema in un problema di insoddisfacibilità e si crea una refutazione:

$$S' := \{\{a, b, \neg c\}, \{a, b, c\}, \{a, \neg b\}, \{\neg a\}\} \text{ è insodd.?}$$

Non è soddisfacibile, poiché

$$\dfrac{\dfrac{\dfrac{\{a,b,\neg c\}\in S' \qquad \{a,b,c\}\in S}{\{a,b\}}\,(\mathbb{R}\text{ con }\ell=c)) \qquad \{a,\neg b\}\in S'}{\{a\}}\,(\mathbb{R}\text{ con }\ell=b)) \qquad \{\neg a\}\in S'}{\{\square\}}\,(\mathbb{R}\text{ con }\ell=a))$$

Si noti che, se non si considera l'ultima risoluzione, si è riusciti a dedurre a da dalla teoria iniziale, come richiesto ($S \vdash_R \{a\}$)!

Tuttavia non si riesce a farlo in maniera generale, infatti:

$$\begin{aligned} & \Gamma \models F & \\ \iff & \Gamma, \neg F \text{ insodd.} & \text{per Lemma 1} \\ \iff & \Gamma^c, (\neg F)^c \vdash_R \square & \text{per Teorema 2} \\ \impliedby & \Gamma^c \vdash_R F^c & \text{non è un } \iff !! \end{aligned}$$

E proprio questa è la debolezza della Completezza Refutazionale rispetto alla Completezza *tout court* di un calcolo C (per cui invece varrebbe $\Gamma \models F \iff \Gamma \vdash_C F$).
Si può dedurre un calcolo refutazionale da una deduzione per risoluzione, ma non il contrario!

Si consideri, per esempio questa semplice refutazione:

$$\{\{b\},\{\neg b\},\{\neg a\}\} \vdash_R \square$$

ma visto che la Risoluzione non introduce mai nuovi letterali, non si potrà mai a provare

$$\{\{b\},\{\neg b\}\} \vdash_R \{a\}$$

5.2.2 Sistemi Assiomatici (Calcoli alla Hilbert)

I Sistemi Assiomatici sono un tipo di calcolo tradizionale completo *tout court*. Hanno una formalizzazione tra le più semplici, ma non sono in particolar modo adatti alla ricerca *human-oriented* e nemmeno alla ricerca di prove tramite algoritmi.

Definizione (Calcoli alla Hilbert)**.** Dato un insieme di assiomi (che sono tautologie della Logica), come per esempio il seguente, che è corretto e completo per la Logica Proposizionale classica

$$\begin{cases} A \to (B \to A) \\ (A \to (B \to C)) \to ((A \to B) \to (A \to C)) \\ (\neg B \to \neg A) \to (A \to B) \end{cases}$$

con $A, B, C \in \mathcal{F}_{\mathcal{L}}$ e avendo regole di inferenza come il *modus ponens*

$$\dfrac{A \qquad A \to B}{B}$$

una prova di A da una teoria Γ nel Calcolo alla Hilbert ($\Gamma \vdash_H A$) è una successione finita di formule:

$$A_1, A_2, \cdots, A_u$$

tale che:

1. $A_u = A$
2. ogni A_i per $i = 1, \cdots, u$ è tale che:

(a) $A_i \in \Gamma$

(b) A_i è un'istanza di assioma

(c) esistono $j,k < i$ tali che $A_j = A_k \to A_i$, quindi

$$\frac{A_k \qquad A_k \to A_i}{A_i}$$

Teorema (Completezza Forte del Calcolo H)**.** $\Gamma \models A \iff \Gamma \vdash_H A$*, anche per teorie* Γ *infinite.*

Il Calcolo H non è particolarmente adatto alla deduzione automatica, come sottolineato precedentemente. I vari tipi di calcolo corrispondono ad esigenze diverse: quando la necessità è la deduzione automatica, vi sono ottime ragioni per preferire il Calcolo Refutazionale. Diamo ora qualche evidenza del perché non sia così saggio utilizzare il Calcolo H. Quest'ultimo è semplice dal punto di vista concettuale, ma tirar fuori prove è più complicato.

Differenze tra i due Calcoli

Differenze nel verificare se $\forall A \in \mathcal{F}_\mathcal{L},\ \neg\neg A \models A$ è vera.

Calcolo R Bisogna inizialmente trasformare il $\forall A \in \mathcal{F}_\mathcal{L}$ in lettere proposizionali $\in \mathcal{L}$:

$$\neg\neg p \models p$$

Si studia l'insoddisfacibilità di

$$\{\neg\neg p, \neg p\} \models \bot$$

che si traduce in forma normale e si cerca di refutarlo:

$$\{\{p\},\{\neg p\}\} \vdash_R \square$$

Si risolve con risoluzioni *automatiche*: $\mathbb{R}(\{p\},\{\neg p\}; p, \neg p) = \square$, pertanto la conseguenza logica è vera.

Calcolo H Nel Calcolo di Hilbert si può tranquillamente ragionare sulle metavariabili, tuttavia bisogna "inventarsi" una dimostrazione (MP = Modus Ponens):

$$\dfrac{\dfrac{\dfrac{\dfrac{(1)}{\neg\neg A \to (\neg\neg\neg\neg A \to \neg\neg A)}\ \text{MP} \qquad \dfrac{}{\neg\neg A}\in\Gamma}{\neg\neg\neg\neg A \to \neg\neg A} \qquad \dfrac{}{(\neg\neg\neg\neg A \to \neg\neg A) \to (\neg A \to \neg\neg\neg A)}\ (3)}{\neg A \to \neg\neg\neg A}\ \text{MP} \qquad \dfrac{}{(\neg A \to \neg\neg\neg A) \to (\neg\neg A \to A)}\ (3)}{\dfrac{\neg\neg A \to A}{A}\ \text{MP} \qquad \dfrac{}{\neg\neg A}\in\Gamma}\ \text{MP}$$

Il calcolo H non presenta un calcolo meccanico e facilmente automatizzabile. Non è facile scegliere quale assioma applicare.

Esercizio Dimostrare che $\models A \to A$

Si prenda una formula B soddisfacibile, già provata, ossia $\models_H B$.

$$\dfrac{\dfrac{B \qquad B \to (A \to B)}{A \to B} \qquad \dfrac{A \to (B \to A) \qquad (A \to (B \to A)) \to (A \to A)}{(A \to B) \to (A \to A)}}{A \to A}$$

5.2.3 Procedura refutazionale di Davis-Putnam

Si supponga che sia stato definito un insieme di clausole

$$S = \{\{p,q\},\{p,\neg q\},\{\neg p,q\},\{\neg p,\neg q\}\}$$

e ci si chiede se sia refutabile. Un umano può vedere a primo d'occhio che la soluzione più facile è:

$$\dfrac{\dfrac{p,q \qquad p,\neg q}{p} \qquad \dfrac{\neg p,q \qquad \neg p,\neg q}{\neg p}}{\{\square\}}$$

Utilizzando, invece, il calcolo refutazionale $\mathcal{R}^*(S)$ estensivamente necessiterebbero 35 passaggi visto che bisogna provare tutti i risolventi possibili (e.g., anche su $C_1 = \{p,q\}$ e $C_2 = \{\neg p,q\}$, e così via).

Vorremmo designare una tecnica che permetta di mantenere la completezza refutazionale selezionando, in qualche modo, i risolventi da calcolare.

Definizione (Sussunzione)**.** Una clausola C_1 **sussume** C_2 sse $C_1 \subset C_2$, ossia è un insieme proprio di C_2.
In termini logici: $\models C_1 \to C_2$, e quindi $\{C_1, C_2\} \equiv \{C_1\}$.

Definizione (Regola di Sussunzione)**.** Sia S un insieme di clausole e sia S' ottenuto da S eliminando tutte le clausole sussunte. Allora si può concludere che $S \equiv S'$.

Osservazione. S' non contiene sussunte, ossia non è ulteriormente riducibile per sussunzione.

Definizione (Clausola Banale)**.** Una clausola C è detta **banale** (trivial) se contiene sia ℓ che il sua opposto $\bar{\ell}$ per qualche letterale ℓ. Ne segue che $C \equiv \top$ o $\models C$.

Definizione (Regola di rimozione banali)**.** Se S' è ottenuta rimuovendo da S tutte le banali, $S \equiv S'$.

Osservazione. Sia $D = \mathbb{R}(C_1, C_2; \ell, \bar{\ell})$ la risoluzione di clausole C_1 e C_2 non banali, allora $\ell \notin D$ e $\bar{\ell} \notin D$, cioè $(C_1 \setminus \{\ell\}) \cup (C_2 \setminus \{\bar{\ell}\}) = (C_1 \cup C_2) \setminus \{\ell, \bar{\ell}\}$

Definizione (Passo semplice di DPP)**.** Dato un input S finito, già ripulito di banali e sussunte, un passo di *Davis-Putnam Procedure* si articola nei seguenti "micropassi":

1. Scegliere una lettera proposizionale $p \in Var(S)$. p sarà detto il *pivot* del passo.

2. S' è l'insieme delle p-esonerate, ossia l'insieme delle clausole in S in cui non occorre p come lettera proposizionale (né come p, né come $\neg p$).

3. S'' è l'insieme delle p-risolventi ed è l'insieme delle clausole ottenute per risoluzione sul pivot p in $S \setminus S'$.

4. S''' è l'insieme $S' \cup S''$ ripulito. È l'output del passo di DPP.

Esempi di DPP

1 Sia $S_0 = \{\{a, b, \neg c\}, \{a, \neg b, d\}, \{a, \neg b, \neg c, \neg d\}, \{\neg a, d\}, \{\neg a, \neg c, \neg d\}, \{c\}\}$.

- S_0 è ripulito.
  - pivot: c
  - c-esonerate: $\{a, \neg b, d\}$ e $\{\neg a, d\}$
  - c-risolventi: $\{a, b\}, \{a, \neg b, \neg d\}, \{\neg a, \neg d\}$
  - output ripulito: $\{\{a, \neg b, d\}, \{\neg a, d\}, \{a, b\}, \{a, \neg b, \neg d\}, \{\neg a, \neg d\}\}$
- – pivot: a
  - a-esonerate: $\emptyset$
  - a-risolventi: $\{\neg b, d\}, \{\neg b, d, \neg d\}, \{b, d\}, \{\neg b, d, \neg b\}, \{b, \neg d\}, \{\neg b, \neg d\}$
  - output ripulito: $\{\{\neg b, d\}, \{b, d\}, \{b, \neg d\}, \{\neg b, \neg d\}\}$
- – pivot: b
  - b-esonerate: $\emptyset$
  - b-risolventi: $\{d\}, \{d, \neg d\}, \{\neg d, d\}\{\neg d\}$
  - output ripulito: $\{\{d\}, \{\neg d\}\}$
- – pivot: d
  - d-esonerate: $\emptyset$
  - d-risolventi: $\square$
  - output ripulito: $\{\square\}$

Dato che è stata ottenuta la clausola vuota, si dichiara la clausola iniziale S_0 insoddisfacibile.

2 Sia $S_0 = \{\{a, \neg b, c\}, \{b, \neg c\}, \{a, c, \neg d\}, \{\neg b, \neg d\}, \{a, b, d\}, \{\neg a, b, d\}, \{b, \neg c, d\}, \{c, \neg c\}\}$.

- S_0 ripulito: $\{\{a, \neg b, c\}, \{b, \neg c\}, \{a, c, \neg d\}, \{\neg b, \neg d\}, \{a, b, d\}, \{\neg a, b, d\}\}$
  - pivot: b
  - b-esonerate: $\{a, c, \neg d\}$
  - b-risolventi: $\{a, c, \neg c\}, \{a, c, d\}, \{a, \neg a, c, d\}, \{\neg c, \neg d\}, \{a, \neg d, d\}, \{\neg a, \neg d, d\}$
  - output ripulito: $S_1 := \{\{a, c, \neg d\}\}, \{a, c, d\}, \{\neg c, \neg d\}$
- – pivot: c
  - c-esonerate: $\emptyset$
  - c-risolventi: $\{a, d, \neg d\}, \{a, \neg d\}$
  - output ripulito: $S_2 := \{\{a, \neg d\}\}$
- – pivot: a
  - a-esonerate: $\emptyset$
  - a-risolventi: $\emptyset$

– output ripulito: $S_3 := \emptyset$

Mostreremo che questo vuol dire che S_0 è soddisfacibile.
La lettera proposizionale d non è mai stata scelta come lettera pivot, quindi è una *fuoriuscita*.

Delineamo, prima di affrontare la dimostrazione, la strategia che permette di costruire un assegnamento che soddisfa S_0 dopo un'esecuzione di DPP giunta a buon fine.
Sia $v : \mathcal{L} \to \{0,1\}$ tale che $v \models S_0$ definita come:

$$v := \begin{cases} d: & v(d) = 0 \quad \text{alle fuoriuscite basta un valore fissato arbitrariamente} \\ a: & v(a) = 1 \quad \text{soddisfa } S_2 \\ c: & v(c) = 0 \quad \text{soddisfa } S_1 \\ b: & v(b) = 1 \quad \text{soddisfa } S_0 \end{cases}$$

Con $v(c)$ si era liberi perché tutte le clausole erano già soddisfatte dai precedenti assegnamenti.

Teorema di completezza (e correttezza) refutazionale di DPP

Teorema 3. *Un insieme **finito** S di clausole tale che $|Var(S)| = n$ è insoddisfacibile sse entro al più n passi si ottiene la clausola vuota $\square$.*
S è soddisfacibile sse entro al più n passi si ottiene l'insieme vuoto di clausole $\emptyset$.
Non vi sono altri modi di terminare.

Dimostrazione di Terminazione. Sia $DPP(S) := S_0, S_1, \cdots, S_k$ una sequenza di clausole, dove S_i è ottenuto da S_{i-1} attraverso un passo di DPP sul pivot $q_i \in Var(S)$.
Si osserva che S_i non conterrà più alcuna occorrenza di q_i e, quindi, dopo al più $k \leq n$ passi si ottiene $Var(S_k) = \emptyset$, dunque o $S_k = \{\square\}$ oppure $S_k = \emptyset$. $\square$

Dimostrazione di Correttezza e Completezza Refutazionale di DPP.

$$\text{Correttezza di DPP: } \begin{cases} S_k = \{\square\} \implies S_0 \text{ insoddisfacibile} \\ S_0 \text{ soddisfacibile} \implies S_k = \emptyset \qquad \text{(contronominale)} \end{cases}$$

Dimostrata dal Lemma 4 di Correttezza della Risoluzione [visto che $S_k \equiv S_0$ (?!)].

$$\text{Completezza di DPP: } \begin{cases} S_0 \text{ insoddisfacibile} \implies S_k = \{\square\} \\ S_k = \emptyset \implies S_0 \text{ soddisfacibile} \qquad \text{(contronominale)} \end{cases}$$

Dimostriamo la contronominale, chiamata anche *Lemma di Model Building* poiché la dimostrazione si occupa di mostrare come costruire l'assegnamento che soddisfa S.

Per induzione decrescente su k:

- **base** ($j = k$): $S_k = \emptyset$ è soddisfacibile da qualsiasi assegnamento.
- **passo induttivo** (j): Dato per ipotesi induttiva $v : \mathcal{L} \to \{0,1\}$ tale che $v \models S_{i+1}$, si definiscono le due varianti

  $$v^+ : \mathcal{L} \to \{0,1\} \quad v^+(p) = 1$$
  $$v^- : \mathcal{L} \to \{0,1\} \quad v^-(p) = 0$$

  dove p è il pivot di $S_i \rightsquigarrow S_{i+1}$, mentre $v(q) = v^+(q) = v^-(q)$ per tutte le lettere proposizionali $q \neq p$.

  Si mostrerà che una delle due varianti soddisfa S_i.
  Per assurdo, si assume $v^+ \not\models S_i$ e $v^- \not\models S_i$.
  Allora esistono $C_1, C_2 \in S_i$ tali che $v^+(C_1) = 0$ e $v^-(C_2) = 0$.

- $p \notin C_1 e \neg p_1 \notin C_1$: allora C_1 sarebbe p-esonerata, dunque $C_1 \in S_{i+1}$. Per ipotesi induttiva $\nu \models S_{i+1}$, quindi anche $\nu \models C_1$.
  $\Rightarrow \nu^+ \models C_1$, assurdo.
- $\neg p \in C_1$: altrimenti, se $p \in C_1$, $\nu^+(C_1) = 1$. Assurdo

Analogamente per $p \in C_2$.
Ne segue che $D = \mathbb{R}(C_1, C_2; \neg p, p)$ e che $D \in p$-risolventi. Quindi $D \in S_{i+1}$, il che vuol dire che $\nu \models D$. Tuttavia:

$$
\begin{aligned}
& \nu \models D \\
\Longrightarrow\ & \nu^+ \models D \text{ oppure } \nu^- \models D && \nu^{\pm} \text{ aggiunge una lettera proposizionale a } \nu \\
\Longrightarrow\ & \nu^+ \models C_1 \text{ oppure } \nu^- \models C_2 && \text{per def. } \nu^{\pm} \\
\Longrightarrow\ & \nu^+ \models S_i \text{ oppure } \nu^- \models S_i \\
\Longrightarrow\ & \bot && \text{per ipotesi assurda}
\end{aligned}
$$

Dunque S_i è soddisfacibile e, pertanto, anche S_0.

□

Osservazione. Sia S un insieme finito di clausole tale che $Var(S) = \{p_1, \cdots, p_n\}$. Allora $DPP(S)$ termina in $k \leq n$ passi. Se $k \leq n \leq ||S||$, si potrebbe concludere che si ha una procedura che funziona in tempo polinomiale e decide se S è soddisfacibile o meno, e quindi $P = NP$.

Se si potesse garantire che $||S_i||$ sia sempre polinomiale rispetto a $||S||$ allora davvero $k \leq n$ passi di DPP porterebbero a un tempo di esecuzione complessivo che è polinomiale rispetto a $||S||$.
E dunque $SAT \in P$, e $P = NP$. Purtroppo questo non si può garantire.

Un risultato, dovuto ad Haken, afferma che ogni prova per refutazione del *Principio della Piccionaia* su n "piccioni" richiede tempo almeno $T(n) = \Omega(2^n)$.

Definizione (Principio della Piccionaia)**.** Il Principio della Piccionaia, notazionalmente espresso su un numero naturale n come

$$PHP(n)$$

afferma che $n+1$ piccioni non possono occupare n posti nella piccionaia in modo che ogni posto abbia al più un piccione.

Come formalizzare in clausole il $PHP(n)$? Definendo P i "piccioni" e H i "posti", si può definire

$$\bigwedge_{i \in P} \bigvee_{j \in H} p_{ij} \quad \text{(ogni piccione ha trovato posto)}$$

Questo, però va unito al fatto che ogni piccione ha al più un posto:

$$\bigwedge_{i \neq j\ i,j \in P} \bigwedge_{k \in H} (\neg p_{ik} \vee \neg p_{jk}) \quad \text{nessuna coppia di piccioni sta nel posto } k$$

Quindi, la formalizzazione finale sarà

$$PHP(n) := \bigwedge_{i \in P} \bigvee_{j \in H} p_{ij} \ \wedge \bigwedge_{i \neq j\ i,j \in P} \bigwedge_{k \in H} (\neg p_{ik} \vee \neg p_{jk})$$

Mostrare che il Principio della Piccionaia è vero consiste nel mostrare che $PHP(n)$ è insoddisfacibile, ossia che $PHP(n) \in CNFUNSAT\ \forall n \in \mathbb{N}$. Haken ha dimostrato che la risoluzione mediante refutazione di questo problema impiega, per ogni n, tempo esponenziale.

Complessità di DPP

Dato che i passi di DPP sono "pochi"—$k \leq |Var(S)|$—anche se nel caso peggiore DPP richiede tempo esponenziale, nella pratica è comunque un algoritmo molto più efficiente di $\mathcal{R}^*$ e delle tavole di verità.
Alcuni frammenti, ossia sottolinguaggi di CNFSAT, sono noti avere tempo di decisione con DPP che è polinomiale, come KROMSAT e HORNSAT.
Una CNF è HORN sse ha solo clausole di Horn, ossia della forma:

$$\square \qquad \overset{\text{(unità)}}{\{p\}} \qquad \overset{\text{(solo negati)}}{\{\neg p_1, \neg p_2, \cdots\}} \qquad \overset{\text{(un solo positivo)}}{\{\neg p_1, \neg p_2, \cdots, q\}}$$

Una CNF è KROM sse ha solo clausole di Krom, ossia se ha al più due letterali ($KROMSAT = 2CNFSAT$).

Davis Putnam Logemann Loveland Procedure

La DPLL è un metodo di "reingegnerizzazione" della DPP. Su un insieme finito S di clausole, DPLL cerca di costruire un assegnamento che soddisfi S.

1. L'idea è di costruire un assegnamento parziale che viene esteso ad ogni passo di una nuova lettera proposizionale p, chiamata pivot.

2. Se si riesce ad assegnare tutte le lettere in $Var(S)$ si è ottenuto un assegnamento che mostra S soddisfacibile.
   Se non si giunge a tal punto, il teorema di completezza e correttezza di DPLL (che non vediamo) dimostra che S è insoddisfacibile.

3. Ad ogni passo si propaga tutta l'informazione che si guadagna estendendo l'assegnamento al pivot.
   Sostanzialmente, quello che era il passo finale della DPP si utilizza come azione fondamentale nella DPLL.

4. Se non ci sono informazioni sfruttabili per la scelta del pivot—guidata da regole che vedremo—allora si "spezza" la procedura in due parti: una in cui si assegna al pivot il valore 1 e una a cui si assegna al pivot il valore 0.

5. La prova risulta dunque in una struttura ad *albero* i cui rami si visitano in "backtracking". L'implementazione del backtracking è soggetto di ampie ricerche (e.g., backtracking non cronologico)

6. Se in un ramo si raggiunge $\emptyset$ si è costruito un assegnamento che soddisfa l'insieme iniziale e se su *tutti* i rami si raggiunge $\square$, allora S è insoddisfacibile.

Le regole utilizzate dalla DPPL sono quelle della DPP adattate a questo contesto.
Il punto di partenza è un *assegnamento vuoto*.

Definizione (Assegnamento Parziale e Vuoto)**.** Un assegnamento parziale è una funzione parziale $v : \mathcal{L} \to \{0, 1, ?\}$.
Un *assegnamento vuoto* è un assegnamento parziale tale che $v(p_i) =? \forall p_i \in \{p_1, \cdots, p_n\}$.
Un *assegnamento completo* o totale sulle prime n lettere proposizionali è una mappa $v : \{p_1, \cdots, p_n\} \to \{0, 1, ?\}$ tale che $v(p_i) \neq? \forall p_i \in \{p_1, \cdots, p_n\}$.

Regole di DPLL

- *Regola iniziale*: $\emptyset \vdash S$ è la radice dell'albero di prova.
- *Sussunzione*: se $v(p_i) = 1$ allora si possono cancellare a S tutte le clausole che contengono il letterale p_i

$$\frac{v, v(p_i) = 1 \vdash S \cup \{\{p_i\} \cup C\}}{v, v(p_i) = 1 \vdash S}$$

  analogamente, se $v(p_i) = 0$, allora si possono cancellare da S tutte le clausole che contengono il letterale $\neg p_i$.
- *Risoluzione unitaria*: si fanno Risolventi in cui una delle due clausole ha un solo letterale. Viene quindi codificata come assegnamento, e alla clausola rimanente è inutile avere il letterale opposto.
  Se $v(p_i) = 0$ si cancella da ogni clausola di S il letterale p_i

$$\frac{\{\neg p_i\} \qquad \{p_i\} \cup C}{C}\ \mathbb{R}$$

  Ossia, in termini di "nodi" DPLL

$$\frac{v, v(p_i) = 0 \vdash S \cup \{\{p_i\} \cup C\}}{v, v(p_i) = 0 \vdash S \cup \{C\}}$$

  mentre se $v(p_i) = 1$, allora si elimina da ogni clausola di S il letterale $\neg p_i$.
- *Asserzione*: se S contiene la clausola $\{p_i\}$, allora si estende v ponendo $v(p_i) = 1$, cancellando $\{p_i\}$ da S

$$\frac{v \vdash S \cup \{\{p_i\}\}}{v, v(p_i) = 1 \vdash S}$$

  mentre si fa l'opposto se S contiene $\{\neg p_i\}$.
- *Letterale puro*: Se il letterale p_i occorre in S e $\neg p_i$ non occorre, allora si estende v ponendo $v(p_i) = 1$,

$$\frac{v \models S}{v, v(p_i) = 1 \vdash S}\ \text{se } p_i \in C \in S, \neg p_i \notin C\ \forall C \in S$$

  mentre si fa l'opposto se $\neg p_i$ occorre in S e p_i non occorre in S, ponendo $v(p_i) = 0$.
- *Spezzamento*: in ogni momento, si può biforcare la prova in due sottoprove, dando origine ad una struttura ad albero, in cui in una si pone per il pivot scelto $v(p) = 1$ e nell'altra si pone $v(p) = 0$.

$$\frac{v \vdash S}{v, v(p_i) = 0 \vdash S \quad || \quad v, v(p_i) = 1 \vdash S}$$

- *Terminazione*: se in un ramo si ottiene $v \vdash \emptyset$ allora si prova che v è completo su $Var(S)$ e $v \models S$. Al contrario, se su tutti i rami si ottiene $v \vdash \square$, allora S è insoddisfacibile.
- Ogni ramo ha come foglia $v \vdash \emptyset$ oppure $v \vdash \square$.

Parte II

Logica dei Predicati

CAPITOLO 6

Introduzione e Sintassi

Si può immaginare la Logica del Primo Ordine come "costruita" sulla base della logica proposizionale. Il sillogismo aristotelico

$$\frac{\text{Ogni uomo è mortale} \qquad \text{Socrate è un uomo}}{\text{Quindi Socrate è mortale}}$$

espresso come possibile nella Logica proposizionale diventa

$$\frac{p \qquad q}{r}$$

Ci si chiede se sia vero, quindi:

$$p, q \overset{?}{\models} r$$

Con DPLL ci si chiede se $\{p, q, \neg r\}$ sia insoddisfacibile. Il primo passo di DPLL afferma

$$\dfrac{\dfrac{\dfrac{\emptyset \vdash \{\{p\},\{q\},\{\neg r\}\}}{\nu(p) = 1 \vdash \{\{q\},\{\neg r\}\}}}{\nu(p) = 1, \nu(q) = 1 \vdash \{\{\neg r\}\}}}{\nu(r) = 0, \nu(p) = 1, \nu(q) = 1 \vdash \emptyset}$$

e pertanto

$$p, q \models r \text{ è falso}$$

che va ovviamente contro la nostra intuizione. La modellizzazione della frase in italiano non è più adeguata a descrivere il mondo su cui si sta lavorando! Per gestire argomentazioni razionali come il sillogismo aristotelico è necessario dotarsi di un linguaggio più *espressivo* rispetto alla Logica Proposizionale, arricchendone la Sintassi con nuovi operatori, variabili, costanti e la Semantica assegnando un modo di interpretare i nuovi "ingredienti" del linguaggio in modo da poter rappresentare situazioni più raffinate che nella Logica Proposizionale.

Si arriverà a scrivere

$$\forall x(U(x) \to M(x)), U(s) \models M(s)$$

per indicare il sillogismo aristotelico. Oltre ad andare a vedere come mettere in piedi, di primo acchito a livello sintattico, tutta questa struttura, si studierà anche il modo per *risolvere* delle asserzioni, riutilizzando le tecnologie introdotte per la logica proposizionale. In particolare, per fare un esempio introduttivo, si tornerà a chiedersi se il sillogismo aristotelico sia valido ponendosi il quesito

$$\{\{\neg U(x), M(x)\}, \{U(s)\}, \{\neg M(s)\}\} \text{ è soddisfacibile?}$$

Si consideri nuovamente

Ogni uomo è mortale

I modi di designare direttamente dell'*Universo* sono un ingrediente importante della Logica dei Predicati, fornendo per esempio il modo di affermare che Socrate sia mortale, riferendosi ad un preciso elemento. Si necessita un modo per designare un elemento non preciso, in maniera indiretta:

Il padre di ogni uomo è un uomo

Si può concludere, quindi, che il padre di Socrate sia un mortale, nonostante sia una designazione indiretta di un individuo dell'Universo; si può inoltre tradurre quanto detto come

$$\frac{\forall x(U(x) \to M(x)) \qquad \forall x(U(x) \to U(p(x)) \qquad U(s)}{M(p(s))}$$

Per decidere questa *deduzione*, ci sarà qualcosa come

$$\{\{\neg U(x), M(x)\}, \{\neg U(x), U(p(x))\}, \{U(s)\}, \{\neg M(p(s))\}\}$$

Quindi, scopriremo che non basterà sostituire al posto di x la "lettera" s, ma sarà anche necessario considerare $p(s)$. A questo punto, sarà immediato arrivare alla conclusione che la situazione sia in realtà un po' più complicata: come si considera $p(s)$, si dovrebbe considerare anche $p(p(s))$, $p(p(p(s)))$...

Nel nostro caso, si trova una possibile refutazione istanziando la x su $p(s)$ come segue:

$$\dfrac{\dfrac{\dfrac{\neg U(p(s)), M(p(s)) \qquad \neg M(p(s))}{\neg U(p(s))} \qquad \dfrac{\neg U(s), U(p(s))}{\neg U(s)}}{\square} \qquad U(s)}$$

Vedremo, quindi, perché e quando si possono utilizzare queste istanzianzioni.

6.1 Sintassi della Logica del Primo Ordine

Partiamo, dunque, fissando i dettagli sintattici, cioè il Linguaggio della Logica del Primo Ordine nella sua natura grammaticale.

Nella Logica Proposizionale bastava fissare $\mathcal{L}$ come insieme infinito di lettere proposizionali per costruire direttamente ogni formula ammissibile in $\mathcal{F}_{\mathcal{L}}$.

Nella Logica del Primo Ordine, invece, vi sono diversi linguaggi detti **linguaggi elementari**. Questi sono un insieme degli elementi per costruire il *vero* linguaggio in senso formale, ovvero le formule della Logica dei Predicati.

Definizione (Linguaggio Elementare)**.** I linguaggi elementari sono definiti come $\mathcal{L} = (\mathcal{P}, \mathcal{F}, \alpha, \beta)$, dove:

- $\mathcal{P}$ è un insieme non vuoto di simboli, detti **predicati** (o simboli di predicato)
- $\mathcal{F}$ è l'insieme di simboli detti **di funzione**. È disgiunto da $\mathcal{P}$; $\mathcal{P} \cap F = \emptyset$
- α è una funzione $\mathcal{P} \to \mathbb{N}$ che assegna l'arietà a ogni $p \in \mathcal{P}$
- β è una funzione $\mathcal{F} \to \mathbb{N}$ che assegna l'arietà a ogni $f \in \mathcal{F}$

Ogni elemento del linguaggio varia in base al linguaggio stesso, tuttavia vi sono "ingredienti" intrinsecamente comuni a tutti i linguaggi elementari:

- $\mathcal{V}$ (o *Var*): insieme infinito di simboli detti **variabili individuali**, chiamate così perché la loro interpretazione sarà quella di un *individuo generico* dell'universo del discorso
- *Connettivi*: $\wedge, \vee, \neg, \rightarrow, \bot, \top, \leftrightarrow, \cdots$
- *Quantificatori*: $\forall, \exists$.

Il linguaggio $\mathcal{F}_\mathcal{L}$ delle formule sul linguaggio elementare $\mathcal{L}$ sarà definito a partire dai simboli in $\mathcal{L}$ e dai possibili connettivi.

Osservazione. Esiste un predicato speciale nella logica del primo ordine che a livello sintattico è uguale agli altri, ma la sua interpretazione è fissata. Se $\mathcal{P}$ contiene il simbolo $'='$, la sua arietà sarà fissata $\alpha('=') = 2$ e $\mathcal{L} = (\mathcal{P}, \mathcal{F}, \alpha, \beta)$ è detto *linguaggio con identità*.

Definizione ($\mathcal{L}$-termini)**.** Sia $\mathcal{L} = (\mathcal{P}, \mathcal{F}, \alpha, \beta)$.
Allora l'insieme $T_\mathcal{L}$ degli $\mathcal{L}$-termini è definito come segue:

- **base**:
  - ogni $x \in \mathcal{V}_\mathcal{L}$ è un termine.
  - ogni $c \in \mathcal{F}$ tale che $\beta(c) = 0$, c è un termine detto **costante**
- **passo**: siano $f \in \mathcal{F}$ tale che $\beta(f) = n$ e $t_1, t_2, \cdots, t_n \in T_\mathcal{L}$ termini già costruiti, allora $f(t_1, t_2, \cdots, t_n)$ è un $\mathcal{L}$-termine.
- nient'altro è un $\mathcal{L}$-termine.

I termini sono dei nomi che designano un oggetto nell'universo del discorso.

Definizione (Formule)**.** Il ruolo delle *formule di* $\mathcal{L}$ è predicare sopra dei determinati $\mathcal{L}$-termini, affermando se certe relazioni tra di loro sono vere o false.

L'insieme $\mathcal{F}_\mathcal{L}$ delle $\mathcal{L}$-formule è definito induttivamente come segue:

- **base**: siano $p \in \mathcal{P}$ tale che $\alpha(p) = n$ e $t_1, \cdots, t_n \in T_\mathcal{L}$ degli $\mathcal{L}$-termini, allora $P(t_1, \cdots, t_n)$ è una $\mathcal{L}$-formula detta **formula atomica**.
  Le formule atomiche sono il corrispettivo nella Logica del Primo Ordine delle lettere proposizionali.
- **passo**: se $A, B \in \mathcal{F}_\mathcal{L}$, allora sono $\mathcal{L}$-formule anche $(A \wedge B)$, $(A \vee B)$, $(\neg A)$, $(A \rightarrow B)$
- se $A \in \mathcal{F}_\mathcal{L}$ e $x \in \mathcal{V}_\mathcal{L}$, allora sono $\mathcal{L}$-formule anche $(\forall xA)$ e $(\exists xA)$.
- nient'altro è una $\mathcal{L}$-formula.

Nota: se in A non occorre x, $\forall xA$ e $\exists xA$ sono comunque formule; così come anche $\forall x(\forall xA)$ e $\exists x(\forall xA)$.

Esercizio Dare una nozione di *certificato* per $\mathcal{L}$-termini e per $\mathcal{L}$-formule analoga alla $\mathcal{L}$-costruzione in logica proposizionale.

6.1.1 Terminologia

Introduciamo alcune nozioni terminologiche per potersi riferire alla sintassi della Logica dei Predicati. Si userà liberamente la scrittura semplificata di formule omettendo coppie di parentesi quando questo non causa ambiguità, ossia invece di $(\forall xA)$ si scriverà $\forall xA$.

Definizione (Termine Ground)**.** Un termine è detto **chiuso** o **ground** se è costruito senza utilizzare *variabili*.

Definizione (Variabili vincolate e libere)**.** Una occorrenza di una variabile x in una formula $A \in \mathcal{F}_{\mathcal{L}}$ è detta **vincolata** se occorre all'interno di una sottoformula di A (ossia una formula che appare in ogni certificato della $\mathcal{L}$-formula A) del tipo $\forall xB$ o $\exists xB$, altrimenti è detta **libera**.

Per esempio, nella formula

$$A = (\forall xR(x,y)) \vee P(x)$$

x è sia vincolata che libera, mentre y è libera. Nella formula

$$A' = \forall x(\forall yR(x,y)) \vee P(x)$$

Sia x che y occorrono sempre vincolate.

Questo ci permette di introdurre il concetto fondamentale sul quale lavoreremo: infatti, non ragioneremo più su *formule* quando definiremo la semantica della Logica dei Predicati, ma si ragionerà su un particoalre tipo di formula:

Definizione ($\mathcal{L}$-enunciato o formula chiusa)**.** Un $\mathcal{L}$-**enunciato** ($\mathcal{L}$-sentence) o *formula chiusa* è una $\mathcal{L}$-formula senza variabili libere.

Definizione (Sostituzione)**.** Con la notazione $A[t/x]$ con $A \in \mathcal{F}_{\mathcal{L}}$, $x \in \mathcal{V}_{\mathcal{L}}$ e t un $\mathcal{L}$-termine, si intende la formula ottenuta rimpiazzando simultaneamente tutte e sole le occorrenze **libere** di x con t.

CAPITOLO 7

Semantica della Logica del Primo Ordine

7.1 Semantica di Tarski

La semantica esprime *come e quando* un $\mathcal{L}$-enunciato è vero (o falso) in una data circostanza. È necessario fissare e formalizzare la nozione di *circostanza* o "mondo possibile". Una volta fatto ciò, sarà necessario formalizzare l'*interpretazione* degli enunciati (e quindi dei termini) in ogni circostanza formalizzata.

Nella Logica Proposizionale la circostanza è un *assegnamento*.
Il logico polacco Alfred Tarski ha proposto di formalizzare la *verità* in una **circostanza** nella Logica del Primo Ordine come corrispondenza con lo *stato di fatto*.

Per la **semantica tarskiana**, nel linguaggio naturale l'enunciato "La neve è bianca" è un enunciato vero se e solo se la neve è bianca. Pertanto, ci si può immaginare di matematizzare questo concetto tramite la teoria degli insiemi, affermando che esista un certo insieme chiamato *Universo del discorso*, composto da vari individui, tra i quali vi è un certo elemento che rappresenta la neve e, rispecchiando la nostra esperienza, si vuole che effettivamente sia vero che la neve sia bianca. Pertanto, "si forza" l'appartenenza della neve all'insieme degli oggetti del discorso che sono bianchi.

In termini matematici, e per il concetto stesso della semantica tarskiana, è necessario considerare Universi in cui vi è la possibilità che la neve non sia bianca, ossia l'enunciato è falso.

A priori, le opzioni *esistono* tutte (la neve è bianca, la neve non è bianca), ma si può utilizzare la formalizzazione degli enunciati anche per modellizzare l'universo, ossia asserendo che il fatto che la neve sia bianca sia vera, e questo sappiamo già farlo con l'uso delle *teorie*.

Per esempio, rielaboriamo il solito sillogismo aristotelico: L'insieme di enunciati

$$\Gamma := \{\{\forall x(U(x) \to M(x))\}, \{U(s)\}\}$$

benché non sia in FNC, è un *teorema*, mentre

$$A := M(s)$$

è la deduzione. Per ottenere un'infrastruttura sulla quale arrivare a compiere il calcolo deduttivo

$$\Gamma, \neg A \text{ è soddisfacibile?}$$

è necessario "concentrarsi" sugli Universi in cui Γ è vero. Si possono considerare diversi **Universi del discorso**, composti da vari *individui*: un enunciato caratterizza alcuni universi in cui un enunciato è vero a scapito di altri.

Si consideri, ora, un universo composto da altri universi: chiamando ciò che abbiamo considerato fino ad ora un *universo* una $\mathcal{L}$-struttura, si costruisce l'insieme di tutte le $\mathcal{L}$-strutture; data una teoria Γ si potrà identificare, nell'insieme di tutte le $\mathcal{L}$-strutture, tutte quelle che modellano Γ, ossia tutti quelli in cui le assunzioni Γ sono valide.

Definiamo formalmente le $\mathcal{L}$-strutture:

Definizione ($\mathcal{L}$-struttura)**.** Sia dato un linguaggio elementare $\mathcal{L} = (\mathcal{P}, \mathcal{F}, \alpha, \beta)$.
Una $\mathcal{L}$-struttura è una coppia $\mathcal{A} = (U, I)$, dove U è un insieme *non vuoto* detto "universo del discorso" (o dominio), mentre I è la funzione d'*interpretazione* che interpreta i simboli linguistici dentro all'universo del discorso.
Più specificatamente:

- per ogni simbolo di predicato $P \in \mathcal{P}$ con arietà $\alpha(P) = n$, $I(P) \subseteq U^n$ (e.g, $M(s)$)
- per ogni simbolo di funzione $f \in \mathcal{F}$ con arietà $\beta(f) = n$, $I(f) : U^n \to U$ (e.g. $p(s)$)

La definizione di $\mathcal{L}$-struttura data formulizza a livello del Primo Ordine la nozione intuitiva di "circostanza" o "mondo possibile". A livello proposizionale, essa è formalizzata dalla nozione di assegnamento.

Osservazione (Costanti)**.** Se $c \in \mathcal{F}$ e $\beta(c) = 0$

$$I(c) : U^0 \to U$$

e $U^0 = \{f : \emptyset \to U\}$, la cui cardinalità è

$$|U^0| = |\{f : \emptyset \to U\}| = |\{\emptyset\}| = 1$$

Pertanto

$$I(c) : \{*\} \to U$$

(la notazione $\{*\}$ rappresenta l'insieme che contiene l'insieme vuoto), che equivale a "scegliere" un elemento di U. Identifichiamo una funzione

$$g : \{*\} \to U$$

con l'elemento scelto da $g(*)$. In altre parole, le costanti sono funzioni zerarie.

Osservazione (Predicati zerari)**.** Se $p \in \mathcal{P}, \alpha(P) = 0$ è un simbolo di predicato zerario, si ha

$$I(P) \subseteq U^0 \quad I(P) \subseteq \{*\}$$

I cui sottoinsiemi sono sé stesso $\{*\}$ (quindi 1) e l'insieme vuoto $\emptyset$ (quindi 0).
L'interpretazione di un predicato zerario svolge, quindi, la funzione di lettera proposizionale. Un modo per giustificare il significato dato a $\{*\}$ e $\emptyset$ è interpretando le proposizioni tramite la loro funzione caratteristica $\chi_P : U^n \to \{0, 1\}$, definita come

$$\bar{a} \in U^n \in I(P) \iff \chi_P(\bar{a}) = 1$$

Osservazione (Interpretazione fissa del predicato di Uguaglianza)**.** Se $= \in \mathcal{P}$ con arietà $\alpha(=) = 2$, allora $\mathcal{L}$ è un *linguaggio con identità* (o uguaglianza). È un predicato importante perché l'interpretazione di "=" è fissata nella $\mathcal{L}$-struttura associata $\mathcal{A} = (U, I)$:

$$I(=) \subseteq U^2 \quad \text{e} \quad I(=) := \{(a, a) : a \in U\}$$

In altre parole, $x = y$ è vero se $(I(x), I(y)) \in I(=)$. Un elemento è in relazione d'identità solo con sé stesso.

L'eccezione sull'interpretazione del predicato di uguaglianza, fissato per ogni linguaggio elementare che lo contiene, è giustificato dal fatto che con la potenza espressiva della Logica dei Predicati, si riesce a dire che una relazione binaria gode della proprietà riflessiva, simmetrica e transitiva.

Ci si può anche spingere a dire che per ogni altro simbolo, sia di predicato che di funzione, quella relazione si comporta in maniera *congruenziale*. Anche questo non è sufficiente per inchiodare l'identità, in quanto una congruenza su un insieme che non è l'identità rispetta gli stessi assiomi.

In altre parole, o l'identià si codifica a forza come si sta facendo ora nella Logica del Primo Ordine, oppure se ne ha una versione molto indebolita, in quanto non si ha modo di distinguere con formule della Logica dei Predicati l'identità da una congruenza sullo stesso insieme.

7.1.1 Semantica degli enunciati in ogni $\mathcal{L}$-struttura

Abbiamo dato la nozione di $\mathcal{L}$-struttura ma non si è ancora detto come si interpretano gli enunciati, formalmente, cioè quando un enunciato è vero o meno in tale $\mathcal{L}$-struttura. Intuitivamente, quello che si vuole fare è quanto anticipato, ossia scegliere le $\mathcal{L}$-strutture che verificano una certa teoria. Tuttavia, questo deve essere eseguito seguendo una definizione rigorosa e sistematicamente.

Si procederà per strati, ossia definendo prima come interpretare i termini e successivamente formule ed enunciati, induttivamente. Si deve catturare l'idea intuitiva: "la neve è bianca" verrà formalizzato in qualcosa come $B(n)$, dove B è un predicato unario e $n \in A$ è una costante (quindi esiste una funzione zeraria il cui risultato è n). Si vorrà formalizzare il tutto in modo tale da avere una qualche $\mathcal{L}$-struttura tale che $I(n)$ sia contenuta in $I(B)$; quindi si avrà che $B(n)$ è vera in $\mathcal{A}$ se e solo se $I(n) \in I(B)$:

$$\mathcal{A} \in B(n) \iff I(n) \in I(B)$$

Il passo più ostico nella definizione di interpretazione di termini e formule (ed enunciati) sarà nell'interpretazione dei quantificatori; per esempio, si supponga che si vuole stabilire se

$$\mathcal{A} \models \exists x A$$

che significa: se esiste un $a \in U$ tale che $\mathcal{A} \models A[a/x]$ e analogamente per l'altro quantificatore

$$\mathcal{A} \models \forall x A$$

che è vero se e solo se per ogni $a \in U$ vale $\mathcal{A} \models A[a/x]$.

Purtroppo a non è un elemento sintattico (i.e., un termine), ma è un elemento *semantico*, e pertanto $A[a/x]$ non è definita. Non c'è nessuna ragione per cui ogni elemento di una $\mathcal{L}$-struttura debba avere, a priori, un nome: piuttosto, è vero il contrario. Per esempio, se si vuole parlare dei reali $\mathbb{R}$, si vorrebbe farlo con un linguaggio elementare $\mathcal{L}$ che contiene solo finitamente molti simboli (e.g., $0-9$ se si vuole lavorare in base 10).
Noi ovvieremo a questo problema dando un nome agli elementi semantici dell'universo derivato dalla semantica stessa.

Definizione (Espansione di un $\mathcal{L}$-struttura)**.** Dati un linguaggio $\mathcal{L} = (\mathcal{P}, \mathcal{F}, \alpha, \beta)$ e una $\mathcal{L}$-struttura $\mathcal{A}(U, I)$, definiamo **espansione** di $\mathcal{L}$ in un nuovo linguaggio $\mathcal{L}_{\mathcal{A}}$ come segue:

- per ogni $a \in U$ l'insieme delle costanti di $\mathcal{L}$ viene arricchito con un nuovo simbolo $\bar{a}$ (i.e., nome di a)

- $\bar{a}$ viene interpretato in a

Formalmente:

$$\mathcal{L}_{\mathcal{A}} := (\mathcal{P}, \mathcal{F}^{\mathcal{A}}, \alpha, \beta^{\mathcal{A}}) \qquad I(\bar{a}) := a \text{ per ogni } a \in U$$

con:

$$\mathcal{F}^{\mathcal{A}} := \mathcal{F} \cup \{\bar{a} : a \in U\} \qquad \begin{cases} \beta^{\mathcal{A}}(f) = \beta(f) & \text{per ogni } f \in \mathcal{F} \\ \beta^{\mathcal{A}}(\bar{a}) = 0 & \text{per ogni } a \in U \end{cases}$$

Definizione (Interpretazione degli $\mathcal{L}_{\mathcal{A}}$-Termini ground)**.** Dati $\mathcal{L} = (\mathcal{P}, \mathcal{F}, \alpha, \beta)$ e $\mathcal{A} = (U, I)$ e la $\mathcal{L}_{\mathcal{A}}$-struttura espansa, si definisce induttivamente:

- **base** ($c \in \mathcal{F}$, con $\beta(c) = 0$): se c è una "vecchia" costante già in $\mathcal{L}$, ha interpretazione $I(c)$ già definita da $\mathcal{A}$ per definizione di $\mathcal{L}$-struttura
- **base** ($\bar{a} \in \mathcal{F}^{\mathcal{A}}$, con $\beta(a) = 0$): se $\bar{a}$ è un nome "nuovo" dell'elemento $a \in U$, ha interpretazione $I(\bar{a}) = a$.
- **passo**: sia $f \in \mathcal{F}$ un simbolo di funzione con arietà $\beta(f) = n$ e siano $t_1, t_2, \cdots, t_n$ $\mathcal{L}_{\mathcal{A}}$-Termini ground che hanno già un'interpretazione per ipotesi induttiva.
  Allora $I(f(t_1, \cdots, t_n)) := I(f)(I(t_1), \cdots, I(t_n))$.

Si noti che $I(f)(I(t_1), \cdots, I(t_n))$ è valida (i.e. $\in U$) perché $I(f) : U^n \to U$ per definizione.

Definizione (Interpretazione degli Enunciati)**.** Sia data $\mathcal{A} = (U, I)$ una $\mathcal{L}$-struttura; allora, si definisce induttivamente la nozione $\mathcal{A} \models A$ (i.e. A è vera in $\mathcal{A}$) per ogni $\mathcal{L}_{\mathcal{A}}$-*enunciato* A:

- **base**: sia $A = P(t_1, \cdots, t_n)$ per $P \in \mathcal{P}$ con $\alpha(P) = n$, e $t_1, \cdots, t_n$ $\mathcal{L}_{\mathcal{A}}$-Termini ground. Allora

$$\mathcal{A} \models P(t_1, \cdots, t_n) \iff \underbrace{\overbrace{I(t_1)}^{\in U}, \cdots, \overbrace{I(t_n)}^{\in U}}_{\in U^n} \in \underbrace{I(P)}_{\subseteq U^n}$$

- **passo**:
  - $\mathcal{A} \models A_1 \wedge A_2 \iff \mathcal{A} \models A_1$ e $\mathcal{A} \models A_2$
  - $\mathcal{A} \models A_1 \vee A_2 \iff \mathcal{A} \models A_1$ o $\mathcal{A} \models A_2$
  - $\mathcal{A} \models \neg A_1 \iff \mathcal{A} \not\models A_1$
  - $\mathcal{A} \models A_1 \to A_2 \iff \mathcal{A} \not\models A_1$ o $\mathcal{A} \models A_2$
  - $\mathcal{A} \models \forall x A \iff \mathcal{A} \models A[\bar{a}/x]$ per ogni $a \in U$
  - $\mathcal{A} \models \exists x A \iff \mathcal{A} \models A[\bar{a}/x]$ per almeno un $a \in U$

Si noti che $\bar{a}$ è un termine, mentre x è una variabile individuale, quindi A diventa un $\mathcal{L}_{\mathcal{A}}$-enunciato a tutti gli effetti.

Abbiamo quindi definito induttivamente $\mathcal{A} \models A$ per gli $\mathcal{L}_{\mathcal{A}}$-enunciati A. A questo punto ci si può "dimenticare" degli $\mathcal{L}_{\mathcal{A}}$-enunciati che non sono $\mathcal{L}$-enunciati (i.e., le formule in cui vi è almeno una variabile libera).

Definizione (Chiusura universale di una formula con variabili libere)**.** Se A è una formula ben formata, ossia $A \in \mathcal{F}_{\mathcal{L}}$ ed eventualmente, non un enunciato, quindi $FV(A) \neq \emptyset$, dove FV è definito come l'insieme di variabili libere che appaiono in A, ci si può chiedere se

$$\mathcal{A} \models A$$

Definito, quindi,

$$FV(A) = \{x_1, \cdots, x_n\}$$

si definisce la chiusura universale di A

$$\forall[A] := \forall x_1 \forall x_2 \cdots \forall x_n A$$

e si postula che

$$\mathcal{A} \models A \iff \mathcal{A} \models \forall[A]$$

ossia $\mathcal{A}$ modella una formula ben formata A se e solo se $\mathcal{A}$ rende vera la sua chiusura universale, che è un $\mathcal{L}$-enunciato.

Osservazione. Se $\mathcal{L}$ è un linguaggio con identità (o con uguaglianza), ossia $\mathcal{L} = (\mathcal{P}, \mathcal{F}, \alpha, \beta)$ e $= \in \mathcal{P}$, $\alpha(=) = 2$, allora, ricordando che in ogni $\mathcal{L}$-struttura $\mathcal{A} = (U, I)$, si deve avere per definizione $I(=) = \{(a,a) : a \in U\}$.
Pertanto $\mathcal{A} \models t_1 = t_2$ se e solo se $I(t_1) = I(t_2)$, con t_1 e t_2 $\mathcal{L}$-Termini ground (che, più correttamente, andrebbe scritto in forma postfissa "$= (t_1, t_2)$").

7.1.2 Definizione alternativa di $\mathcal{A} \models A$

In questa definizione alternativa, le variabili sono oggetti di interesse; tuttavia le due definizioni—quella basata sull'ampliamento $\mathcal{L} \to \mathcal{L}_{\mathcal{A}}$ e quella che andiamo a dare—sono equivalenti.

Definizione (Ambiente)**.** Un **ambiente di interpretazione** in una $\mathcal{L}$-struttura $\mathcal{A} = (U, I)$ è una funzione

$$v : Var \to U$$

ossia ad ogni variabile viene assegnato un elemento di U: $v(x) \in U$.

Definizione (Variante di ambiente)**.** La **variante di ambiente** servirà a dare la semantica alle espressioni quantificate ed è indicato $v[a/x](y)$ e si definisce

$$v[a/x](y) := \begin{cases} v(y) & \text{se } y \neq x \\ a & \text{se } y = x \end{cases}$$

Definizione (Interpretazione degli $\mathcal{L}$-Termini)**.** L'interpretazione degli $\mathcal{L}$-Termini (anche aperti!)

- *variabili*: per ogni $x \in Var$ e $v : Var \to U$,

$$I_v(x) := v(x)$$

- *costanti*: per ogni $c \in \mathcal{F}$ con $\beta(c) = 0$,

$$I_v(c) := I(c)$$

- *termini composti*: per ogni $f \in \mathcal{F}$ con $\beta(f) > 0$

$$I_v(f(t_1, \cdots, t_n)) := (I(f))(I_v(t_1), \cdots, I_v(t_n))$$

Definizione (Interpretazione delle formule)**.** Si indica con $\mathcal{A} \models_v A$ se $\mathcal{A}$ rende vera A nell'ambiente $v : Var \to U$, e si definisce

- *atomica*:

$$\mathcal{A} \models_v P(t_1, \cdots, t_2) \iff (I_v(t_1), \cdots, I_v(t_n)) \in I(P)$$

- *proposizioni*:
  - $\mathcal{A} \models_v A_1 \wedge A_2 \iff \mathcal{A} \models A_1$ e $\mathcal{A} \models A_2$
  - $\mathcal{A} \models_v A_1 \vee A_2 \iff \mathcal{A} \models A_1$ o $\mathcal{A} \models A_2$
  - $\mathcal{A} \models_v \neg A_1 \iff \mathcal{A} \not\models A_1$
  - $\mathcal{A} \models_v A_1 \to A_2 \iff \mathcal{A} \not\models A_1$ o $\mathcal{A} \models A_2$
  - $\mathcal{A} \models_v \forall x A \iff \mathcal{A} \models_v A[a/x]$ per ogni $a \in U$
  - $\mathcal{A} \models_v \exists x A \iff \mathcal{A} \models_v A[a/x]$ per almeno una $a \in U$

Concludiamo dicendo che $\mathcal{A} \models A$ sse *per ogni* $v : Var \to A$ si ha $\mathcal{A} \models_v A$.

7.1.3 Intuizione: differenze delle due semantiche

Si esprime, ora, un'idea intuitiva di come avvenga il processo di interpretazione di un $\mathcal{L}$-enunciato. Nella prima definizione si ha il linguaggio elementare $\mathcal{L}$ e delle $\mathcal{L}$-strutture che vengono associate.

$$\mathcal{L} \rightsquigarrow \begin{cases} \mathcal{A} \rightsquigarrow \mathcal{L}_{\mathcal{A}} \\ \mathcal{B} \rightsquigarrow \mathcal{L}_{\mathcal{B}} \\ \mathcal{C} \rightsquigarrow \mathcal{L}_{\mathcal{C}} \\ \vdots \end{cases}$$

Per ogni $\mathcal{L}$-struttura si crea l'espansione associata.

Nella seconda definizione si ha nuovamente il linguaggio $\mathcal{L}$ e delle $\mathcal{L}$-strutture, tuttavia esse non si espandono ma vi sono un numero di ambienti da considerare:

$$\mathcal{L} \rightsquigarrow \begin{cases} \mathcal{A} \begin{cases} \mathcal{A}, v_1 \\ \mathcal{A}, v_2 \\ \mathcal{A}, v_3 \\ \cdots \end{cases} \\ \mathcal{B} \begin{cases} \mathcal{B}, v_1 \\ \mathcal{B}, v_2 \\ \mathcal{B}, v_3 \\ \cdots \end{cases} \\ \vdots \end{cases}$$

7.2 Terminologia e Nozioni

Definizione ($\mathcal{L}$-teoria)**.** Una $\mathcal{L}$-teoria è un insieme di $\mathcal{L}$-enunciati.

Se Γ è una $\mathcal{L}$-teoria e $\mathcal{A}$ è una $\mathcal{L}$-struttura, si dice che $\mathcal{A}$ è modello di Γ ($\mathcal{A} \models \Gamma$) sse $\mathcal{A} \models \gamma$ per ogni $\gamma \in \Gamma$.

La $\mathcal{L}$-teoria Γ è soddisfacibile sse esiste almeno un $\mathcal{A}$ tale che $\mathcal{A} \models \Gamma$, altrimenti Γ è insoddisfacibile.

Sia Γ una $\mathcal{L}$-teoria e A un $\mathcal{L}$-enunciato. Allora A è conseguenza logica di Γ e si scrive $\Gamma \models A$ sse A è vera in ogni modello di Γ, ossia per ogni $\mathcal{L}$-struttura $\mathcal{A}$, $\mathcal{A} \models \Gamma \implies \mathcal{A} \models A$.

Definizione (Verità Logica)**.** Un $\mathcal{L}$-enunciato A è detto **verità logica** sse per ogni $\mathcal{L}$-struttura $\mathcal{A}$, $\mathcal{A} \models A$.

Il concetto di Verità Logica è analogo al concetto di tautologia nella Logica Proposizionale. Un esempio di verità logica è $\forall x(x = x)$.
Lo scopo della nostra indagine sarà, d'ora in poi, cercare di stabilire se

$$\Gamma \overset{?}{\models} A$$

dati Γ una $\mathcal{L}$-teoria e A un $\mathcal{L}$-enunciato. Se Γ è finita, ossia $\Gamma = \{\gamma_1, \cdots, \gamma_n\}$, varrà nuovamente il fatto

$$\Gamma \models A \iff \gamma_1 \wedge \cdots \wedge \gamma_n \wedge \{\neg A\} \text{ insodd.}$$

ossia ci si riduce all'analisi di una singola formula, mentre invece se Γ è infinito, possiamo solo chiederci se

$$\Gamma \models A \iff \Gamma \cup \{\neg A\} \text{ insodd.}$$

Anche nel caso in cui Γ è finito, ci saranno casi in cui il processo di deduzione sarà solo *semidecidibile*, quando nella Logica Proposizionale il problema di soddisfacibilità almeno era decidibile.

7.2.1 Esempi di $\mathcal{L}$-teorie

Congruenze su un $\mathcal{L}$-Linguaggio $\mathcal{L}$

Data una relazione R, si vorrebbe modellarla come congruenza.

$$\Gamma := \begin{cases} \forall x \ R(x,x) & \text{riflessiva} \\ \forall x \forall y \ R(x,y) \to R(y,x) & \text{simmetrica} \\ \forall x \forall y \forall z \ R(x,y) \wedge R(y,z) \to R(x,z) & \text{transitiva} \\ \forall x_1 \cdots \forall x_n \forall y_1 \cdots \forall y_n & \\ \quad (R(x_1,y_1) \wedge \cdots \wedge R(x_n,y_n)) \to R(f(x_1,\cdots,x_n), f(y_1,\cdots y_n)) & \\ \forall x_1 \cdots \forall x_n \forall y_1 \cdots \forall y_n & \\ \quad (R(x_1,y_1) \wedge \cdots \wedge R(x_n,y_n)) \to (P(x_1,\cdots x_n) \to P(y_1,\cdots,y_n)) & \end{cases}$$

I primi tre $\mathcal{L}$-enunciati—o *assiomi* di Γ—assiomatizzano una relazione d'equivalenza, ossia

$$\text{se } \mathcal{A} \models \Gamma \text{ allora } I(R) \text{ è una relazione d'equivalenza}$$

Il quarto $\mathcal{L}$-enunciato assiomatizza la congruenza R rispetto ad una generica $f \in \mathcal{F}$ con $\beta(f) = n$, mentre il quinto fa la stessa cosa rispetto un generico predicato $P \in \mathcal{P}$.
Si noti che la relazione di uguaglianza ($=$) è una relazione di congruenza, ma non è per forza vero il contrario! Inoltre, per un limite espressivo della Logica del Primo ordine, non si riesce a distinguere queste due relazioni. Per questo noi abbiamo fissato una specifica semantica all'$=$.

Relazione d'ordine (parziale)

$$\Gamma_{PO} := \begin{cases} \forall x \ \ R(x,x) & \text{riflessiva} \\ \forall x,y \ \ (R(x,y) \wedge R(y,x)) \rightarrow = (x,y) & \text{antisimmetrica} \\ \forall x \forall y \forall z \ \ (R(x,y) \wedge R(y,z)) \rightarrow R(x,z) & \text{transitiva} \end{cases}$$

Si noti l'utilizzo del predicato d'uguaglianza ($=$), pertanto il Linguaggio è dotato di tale predicato. Un esempio è l'insieme dei naturali con la relazione di minore-o-uguale che modella Γ:

$$(\mathbb{N}, \leq) \models \Gamma_{PO}$$

(la notazione utilizzata significa che $I(R) = \leq$) così come i naturali con l'ordine di divisibilità:

$$(\mathbb{N}, |) \models \Gamma_{PO}$$

Relazione d'ordine (totale) Si può costruire una $\mathcal{L}$-teoria Γ_{TO} che, "allargando" Γ_{PO} con un solo $\mathcal{L}$-enunciato, modella solo uno dei due esempi:

$$\Gamma_{TO} := \Gamma_{PO} \cup \{\forall x \forall y \ \ (R(x,y) \vee R(y,x))\}$$

Effettivamente, $(\mathbb{N}, \leq) \models \Gamma_{PO}$, mentre $(\mathbb{N}, |) \not\models \Gamma_{PO}$ perché vi sono elementi m, n tali che né $m|n$ né $n|m$.

Aggiungendo nuovi assiomi si possono caratterizzare altri tipi di ordine: ad esempio, si può caratterizzare un ordine che abbia un elemento minimo:

$$\Gamma_{min} := \Gamma_{PO} \cup \{\exists m \forall x \ \ (x \neq m \wedge \neg R(x,m))\}$$

Teoria dei gruppi (1)

Un gruppo è una struttura algebrica. Un gruppo è un modello dell'insieme di assiomi Γ_G sul linguaggio $\mathcal{L} = (\mathcal{P}, \mathcal{F}, \alpha, \beta)$.

$$\Gamma_{G_1} := \begin{cases} \forall x \forall y \forall z \ \ ((x * y) * z) = (x * (y * z)) & \text{associatività} \\ \forall x \ \ (x * e) = x \wedge (e * x) = x & \text{elemento neutro} \\ \forall x \ \ (x * x^{-1}) = e \wedge (x^{-1} * x = e) & \text{invertibilità} \end{cases}$$

con $\mathcal{P} = \{=\}$, $\mathcal{F} = \{*, e, \_^{-1}\}$ e relative arietà α e β.
$(G, \cdot, ^{-1}, 0) \models \Gamma_{G_1}$ se e solo se G è un gruppo; in altri termini un gruppo è una struttura con un'operazione associativa in cui ogni elemento è invertibile.
Per esempio, $(\mathbb{Z}, +, -_1, 0) \models \Gamma_{G_1}$ e $(\mathbb{Q} \setminus \{0\}, \cdot, ^{-1}, 1) \models \Gamma_{G_1}$.

Teoria dei gruppi (2) Una seconda formulazione è la seguente

$$\Gamma_{G_2} := \begin{cases} \forall x \forall y \forall z \ \ ((x*y)*z) = (x*(y*z)) & \text{associatività} \\ \forall x \ \ (x*e) = x \wedge (e*x) = x & \text{elemento neutro} \\ \forall x \exists y \ \ (x*y) = e \wedge (y*x) = e & \text{esistenza inverso} \end{cases}$$

Questa $\mathcal{L}$-teoria rimuove la specifica dell'inverso. Se si ottiene un gruppo che modella l'altra formulazione, esso andrà bene anche per questo, ma non è detto il viceversa. Ora, infatti, non è più necessario specificare l'operazione inverso, e si può quindi affermare $(\mathbb{Z},+,0) \models \Gamma_{G_2}$, $(Q,+,0) \models \Gamma_{G_2}$, eccetera. Come anticipato, questo vale anche per i gruppi esplicitati prima: $(\mathbb{Q} \setminus 0, \cdot, ^{-1}, 1) \models \Gamma_{G_2}$ e così via, anche se l'inverso non viene utilizzato come simbolo esplicito. Potremmo quindi utilizzare, al posto del simbolo dell'inverso, qualsiasi cosa e rimarrà comunque un modello di Γ_{G_2}:

$$(\mathbb{Z},+,+1,0) \models \Gamma_{G_2}$$

Ma ovviamente non è vero il contrario, ossia

$$(\mathbb{Z},+,+1,0) \not\models \Gamma_{G_1}$$

in questo senso, abbastanza sottile, le due formulazioni non sono equivalenti.

Possiamo quindi dire che la Teoria dei Gruppi (1) è la Teoria dei Gruppi (2) *skolemizzata* perché priva dell'esistenziale).

Tipi di dato: Stack

Definiamo il linguaggio $\mathcal{L} = (\mathcal{P}, \mathcal{F}, \alpha, \beta)$ con

$$\mathcal{P} = \{Stack, Elem, =\} \qquad \mathcal{F} = \{push, pop, top, nil\}$$

$$\Gamma_{Stack} := \begin{cases} \forall x \ \ (Stack(x) \vee Elem(x)) \\ \forall x \ \ (\neg Stack(x) \vee \neg Elem(x)) \\ \forall x \forall y \ \ ((Stack(x) \wedge Elem(y)) \rightarrow top(push(x,y)) = y) \\ \forall x \forall y \ \ ((Stack(x) \wedge Elem(y)) \rightarrow pop(push(x,y)) = x) \\ \forall x \ \ (Stack(x) \rightarrow push(pop(x), top(x)) = x) \end{cases}$$

Aritmetica di Peano

Sia $\mathcal{L}_{PA} = (\mathcal{P}, \mathcal{F}, \alpha, \beta)$, con $\mathcal{P} = \{=\}$, $\alpha(=) = 2$, $\mathcal{F} = \{+, \times, s, 0\}$ $\beta(+) = \beta(\times) = 2$, $\beta(s) = 1$, $\beta(0) = 0$. Inoltre sia P una formula con una sola variabile libera ($FV(P) = \{v\}$).
L'aritmetica di Peano afferma:

$$\Gamma_{PA} := \begin{cases} \forall x \ \ \neg(0 = s(x)) \\ \forall x, \forall y \ \ (s(x) = s(y)) \rightarrow (x = y) \\ \forall x \ \ (x + 0 = x) \\ \forall x \forall y \ \ (x + s(y)) = s(x + y) \\ \forall x \ \ (x \times 0 = 0) \\ \forall x \forall y \ \ (x \times s(y)) = (x + (x \times y)) \\ (P[0/v] \wedge \forall x \ \ (P[x/v] \rightarrow P[s(x)/v])) \rightarrow \forall x \ \ P[x/v] \end{cases}$$

L'ultimo postulato è una descrizione di infiniti molti assiomi (uno per ogni P) che codificano il principio di induzione—principio alla base dei numeri naturali.

CAPITOLO 8

Risoluzione Automatica

Lo scopo del nostro corso è analizzare come sia possibile capire se una certa teoria $\Gamma \stackrel{?}{\models} A$ in modo automatico. Se $\Gamma = \{\gamma_1, \cdots, \gamma_n\}$, $\Gamma \models A$ se e solo se $\gamma_1 \wedge \gamma_2 \wedge \cdots \wedge \gamma_n \wedge \neg A$ è insoddisfacibile. Quindi, informalmente, il nostro scopo è, dato A un $\mathcal{L}$-enunciato, stabilire se A è insoddisfacibile.

8.1 Forme Normali

Cercheremo di seguire il più possibile ciò che è stato fatto anche per la Logica Proposizionale, ossia riportare il tutto in CNF per poi applicare qualcosa di simile al calcolo refutazionale - tuttavia, nella Logica dei Predicati, questo passaggio è leggermente più complicato.

Prima di esaminare questa procedura, diamo delle definizioni utili.

Definizione (Equivalenza Logica)**.** Siano A e B $\mathcal{L}$-enunciati. A è logicamente equivalente a B sse A e B hanno gli stessi modelli per ogni $\mathcal{L}$-struttura, cioè:

$$A \equiv B \iff \forall \mathcal{A} \ \ (\mathcal{A} \models A \iff \mathcal{A} \models B)$$

Equivalentemente si può definire tramite la seguente verità logica:

$$\models (A \to B) \wedge (B \to A)$$

Definizione (Equisoddisfacibilità)**.** Siano A e B due $\mathcal{L}$-enunciati. A *equisodd*. B sse A ha un modello sse B ha un modello; anche se sono modelli diversi.

Definizione (Forma Normale Prenessa (PNF))**.** Un $\mathcal{L}$-enunciato è in PNF sse è nella forma

$$\overbrace{Q_1 x_1, \ldots, Q_n x_n}^{\text{prefisso}} \overbrace{M}^{\text{matrice}}$$

dove $Q_i \in \{\forall, \exists\}$ e M una F.b.F. priva di quantificatori.

Ogni $\mathcal{L}$-enunciato è logicamente equivalente ad un $\mathcal{L}$-enunciato in PNF, ossia

$$A \equiv A^P, \text{ con } A^P \in PNF$$

A^P si costruisce efficientemente da A utilizzando la ripetuta riscrittura tramite **equivalenze logiche notevoli**, ossia partendo da un certo A_0, si arriva in un numero limitato di passi ad A_N che è in PNF:

$$A := A_0 \rightsquigarrow A_1 \rightsquigarrow A_2 \ldots \rightsquigarrow A_n = A^P \in PNF$$

8.1.1 Equivalenze Notevoli

Sia P una F.b.F. e sia z una variabile che **non occorre** (né libera né vincolata) in P.

Rinomina

$$\begin{cases} \exists x P \equiv \exists z P[z/x] \\ \forall x P \equiv \forall z P[z/x] \end{cases}$$

De Morgan

$$\begin{cases} \neg\forall x P \equiv \exists x \neg P \\ \neg\exists x P \equiv \forall x \neg P \\ \forall x P \equiv \neg\exists x \neg P \\ \exists x P \equiv \neg\forall x \neg P \end{cases}$$

Dimostrazione. di (1)

$$\begin{aligned} & \mathcal{A} \models \neg\forall x P \\ \iff & \mathcal{A} \not\models \forall x P \\ \iff & \text{non è vero che } \forall a \in A \;\; \mathcal{A} \models P[\bar{a}/x] \\ \iff & \text{non per tutti gli } a \in A \;\; \mathcal{A} \models P[\bar{a}/x] \\ \iff & \text{ per almeno un } a \in A \;\; \mathcal{A} \not\models P[\bar{a}/x] \\ \iff & \text{ esiste almeno un } a \in A \;\; \mathcal{A} \models \neg P[\bar{a}/x] \\ \iff & \mathcal{A} \models \exists x \neg P \end{aligned}$$

□

Quantificazione Ridondante

$$\begin{cases} \forall x \forall y \;\; P \equiv \forall y \forall x \;\; P & \\ \exists x \exists y \;\; P \equiv \exists y \exists x \;\; P & \\ \forall x \;\; P \equiv P & \text{se } x \notin FV(P) \\ \exists x \;\; P \equiv P & \text{se } x \notin FV(P) \end{cases}$$

È importante notare che invece non vale, in genere, $\forall x \exists y \;\; P \equiv \exists y \forall x \;\; P$.

Quantificatore rispetto a $\wedge$ e $\vee$

$$\begin{cases} \forall x \;\; (P_1 \wedge P_2) \equiv \forall x \;\; P_1 \wedge \forall x \;\; P_2 & \\ \exists x \;\; (P_1 \vee P_2) \equiv \exists x \;\; P_1 \vee \exists x \;\; P_2 & \\ \forall x \;\; (P_1 \vee P_2) \equiv (\forall x \;\; P_1) \vee P_2 & \text{se } x \notin FV(P_2) \\ \exists x \;\; (P_1 \wedge P_2) \equiv (\exists x \;\; P_1) \wedge P_2 & \text{se } x \notin FV(P_2) \end{cases}$$

Si noti che, per esempio, non valgono:

$$\forall x \;\; (P_1 \vee P_2) \neq \forall x \;\; P_1 \vee \forall x \;\; P_2 \qquad\qquad \exists x \;\; (P_1 \wedge P_2) \neq \exists x \;\; P_1 \wedge \exists x \;\; P_2$$

Siano, infatti, P l'insieme dei numeri pari, mentre D l'insieme dei numeri dispari. Allora:

$$\mathbb{N} \models \forall x(P(x) \vee D(x)) \qquad \mathbb{N} \not\models \forall x P(x) \vee \forall x D(x)$$

Analogamente:

$$\mathbb{N} \not\models \exists x(P(x) \wedge D(x)) \qquad \mathbb{N} \models \exists x P(x) \wedge \exists x D(x)$$

Trasformazione in FNP

Il modo corretto di mettere una formula precedente in Forma Normale Prenessa è usando la rinomina:

$$\exists x P(x) \wedge \exists x D(x) \equiv \exists x \exists y(P(x) \wedge D(y))$$

Dimostrazione.

$$\begin{aligned} \exists x P(x) \wedge \exists x D(x) &\equiv \exists x(P(x) \wedge \exists x D(x)) && \text{perché } x \notin FV(\exists x D(x)) \\ &\equiv \exists x(P(x) \wedge \exists y D(y)) && \text{Rinomina (8.1.1)} \\ &\equiv \exists x \exists y(P(x) \wedge D(y)) && \text{perché } y \notin FV(\exists x D(x)) \end{aligned}$$

□

Altre Equivalenze Notevoli

Dati due quantificatori $Q_1, Q_2 \in \{\forall, \exists\}$:

$$\begin{cases} Q_1 x P_1 \vee Q_2 x P_2 \equiv Q_1 x Q_2 z(P_1 \vee P_2[z/x]) \\ Q_1 x P_1 \wedge Q_2 x P_2 \equiv Q_1 x Q_2 z(P_1 \wedge P_2[z/x]) \end{cases}$$

con $z \notin FV(P_1) \cup FV(P_2)$, o in generale una nuova variabile.
Inoltre, se $x \notin FV(Q)$ (o con rinomina):

$$\begin{cases} \forall x P \to Q \equiv \exists x(P \to Q) \\ \exists x P \to Q \equiv \forall x(P \to Q) \\ Q \to \exists x P \equiv \forall x(Q \to P) \\ Q \to \forall x P \equiv \exists x(Q \to P) \end{cases}$$

Dato un $\mathcal{L}$-enunciato A, c'è una sequenza

$$A := A_0 \rightsquigarrow A_1 \rightsquigarrow A_2 \ldots \rightsquigarrow A_n = A^P \in PNF$$

e ogni passaggio $A_i \rightsquigarrow A_{i+1}$ è ottenuto applicando una delle equivalenze notevoli elencate, dunque $A \equiv A^P$.

Osservazione. Il numero di equivalenze notevoli da utilizzare per la trasformazione è

$$n \leq p(||A||)$$

per un qualche polinomio p.

8.1.2 Forma Normale di Skolem

Si vuole arrivare nuovamente al calcolo refutazionale basato su *risoluzione*, ma per poter dimostrare che una certa formula è insoddisfacibile non basta che sia in PNF, ma deve essere anche in Forma Normale di Skolem.

Definizione (Forma Normale di Skolem)**.** Un $\mathcal{L}$-enunciato $A \in PNF$ si dice in Forma Normale di Skolem sse non contiene occorrenze del quantificatore esitenziale.

$$\text{sia } A := Q_1 x_1 Q_2 x_2 \cdots Q_n x_n M$$
$$\text{allora } A \in SNF \iff Q_i = \forall i \in n$$

Vi sono altre traduzioni di calcolo, che invece mirano alla prova diretta (come i calcoli alla Hilbert per la Logica Proposizionale) in cui si vuole eliminare il quantificatore universale.

Skolemizzazione

Vi sarà dunque un procedimento, chiamato **skolemizzazione**, che porta un generico $\mathcal{L}$-enunciato A in Forma Normale Prenessa ad un enunciato **equisoddisfacibile** in Forma Normale di Skolem.

Esempio di Skolemizzazione Riprendiamo le due formulazioni date della teoria dei gruppi:

$$\Gamma_{G_1} := \begin{cases} \forall x \forall y \forall z \ ((x*y)*z) = (x*(y*z)) & \text{associatività} \\ \forall x (x*e) = x \wedge (e*x) = x & \text{elemento neutro} \\ \forall x (x*x^{-1}) = e \wedge (x^{-1}*x = e) & \text{invertibilità} \end{cases}$$

$$\Gamma_{G_2} := \begin{cases} \forall x \forall y \forall z \ ((x*y)*z) = (x*(y*z)) & \text{associatività} \\ \forall x (x*e) = x \wedge (e*x) = x & \text{elemento neutro} \\ \forall x \exists y (x*y) = e \wedge (y*x) = e & \text{esistenza inverso} \end{cases}$$

La seconda formulazione non è equivalente alla prima, tuttavia sono perlomeno equisoddisfacibili. Si può notare che nella seconda formulazione vi è un assioma con un quantificatore esistenziale. Si può pensare che la prima formulazione sia la Forma Skolemizzata della seconda formulazione.

Nella seconda formulazione, il terzo assioma è descritto come

$$\forall x \exists y (x*y) = e \wedge (y*x) = e$$

Questo sarà vero in una struttura (che sarà un gruppo) $\mathcal{A} = (U, I)$ se e solo se per ogni $a \in U$ esiste $b \in U$ tale che $\bar{a} * \bar{b} = e$ e $\bar{b} * \bar{a} = e$. L'idea sottesa per portare questo assioma nella sua Forma Skolemizzata è che si può associare all'esistenza di b una certa funzione di a, in questo caso *l'inverso* di a. In altre parole, ad ogni a si associa un elemento $f(a)$ tale che l'assioma sia verificato. Nell'esempio concreto, si arriva dunque a definire una nuova $\mathcal{L}$-struttura $\mathcal{A}'$ in modo tale che

$$\mathcal{A}' \models \forall x (x*x^{-1} = e \wedge x^{-1}*x = e)$$

La funzione inverso è esattamente il "gioco" sotteso alla Skolemizzazione: ora, l'assioma legge che $\mathcal{A}'$ è un modello per l'assioma se e solo se per ogni $a \in U$ si ha che $\bar{a} * \bar{a}^{-1} = e$ e $\bar{a}^{-1} * \bar{a} = e$, postulando in qualche senso che la funzione $f(a) = b$ esiste ed è chiamata inverso.

È chiaro che questo processo non può preservare l'equivalenza logica per due motivi: il primo, superficiale, è che i due $\mathcal{L}$-enunciati sono scritti in due linguaggi diversi. Il secondo, più profondo, è che quando si interpreta il nuovo simbolo f, c'è *un* modo per interpretarlo bene, ma esistono anche tutte le altre interpretazioni che possono far fallire la formula; l'unica cosa che si sa è che c'è *almeno una* interpretazione che possa far funzionare le cose.

Processo di Skolemizzazione Questo discorso è assolutamente generale ed è il cuore della Skolemizzazione:

$$\mathcal{A} \models A = \forall x_1 \cdots \forall x_n \exists y B$$

Senza perdità di generalità, si assume che in A non occorrano quantificazioni ridondanti, cioè in B non occorrono cose come $\forall x$, $\exists x$, in altre parole ogni occorrenza di $x_1, \cdots, x_n, y \in B$ è libera. In secondo luogo, in B possono occorrere altri quantificatori, pertanto $\exists y$ è solo l'occorrenza "più esterna".

Applicando la definizione di semantica di Tarski si legge che quanto è scritto è vero se e solo se per ogni $(a_1, \cdots, a_n) \in U^n$ esiste un $b \in U$ tale che

$$\mathcal{A} \models A = B[\bar{a}_1/x_1] \cdots [\bar{a}_n/x_n][\bar{b}/y]$$

il concetto di esistenza (di b in questo caso) viene riletto come un modo per associare un immagine ad un argomento di una qualche funzione: esisterà quindi

$$f(a_1, \cdots, a_n) = b$$

Ciò che la Skolemizzazione "fa" è introdurre un simbolo di funzione n-ario $\bar{f}$ nel linguaggio interpretandolo come la funzione f; dunque, la $\mathcal{L}$-struttura iniziale viene modificata in una $\mathcal{L}$-struttura estesa che deve interpretare anche $\bar{f}$:

$$\mathcal{A}' \models A^S = \forall x_1 \cdots \forall x_n B[f(x_1, \cdots, x_n)/y]$$

creando quindi A^S, la Forma Skolemizzata di A.
Allo stesso modo si passa da un linguaggio $\mathcal{L}$ ad uno $\mathcal{L}^S$:

$$\mathcal{L} \rightsquigarrow \mathcal{L}^S = (\mathcal{P}, \mathcal{F} \cup \{f\}, \alpha, \beta^S) \qquad \text{con} \begin{cases} \beta^S(f) = n \\ \beta^S(x) = \beta(x) \quad \text{altrimenti} \end{cases}$$

E da una $\mathcal{L}$-struttura ad una $\mathcal{L}^S$-struttura:

$$\mathcal{A} = (U, I) \rightsquigarrow (\mathcal{A}, h) = (U, I_h) \qquad \text{con} \begin{cases} I_h(f) = h \\ I_h(x) = I(x) \quad \text{altrimenti} \end{cases}$$

Dove h è una funzione n-aria $h : U^n \to U$

Equisoddisfacibilità Definiamo A e la sua forma Skolemizzata A^S

$$A := \exists x P(x) \qquad A^S := P[c/x] = P(c)$$

Sia $\mathcal{A} = (U, I)$ la $\mathcal{L}$-struttura con:

$$U = \{e_1, e_2\} \qquad I(P) = \{e_2\}$$

Allora $(\mathcal{A}, e_2) \models A$ e $(\mathcal{A}, e_1) \models A$, infatti:

$$\begin{aligned} (\mathcal{A}, e_2) \models A &\iff (\mathcal{A}, e_2) \models \exists x P(x) \\ &\iff \text{esiste } a \in U \text{ tale che } \mathcal{A} \models P(x)[\bar{a}/x] \\ &\iff \text{esiste } a \in U \text{ tale che } \mathcal{A} \models P(a) \\ &\iff \text{esiste: } a = e_2 \quad \mathcal{A} \models P(e_2) && \text{perché } I_{e_2}(\bar{e}_2) \in I_{e_2}(P) \\ &\iff \top && \text{perchè } e_2 \in \{e_2\} \end{aligned}$$

Parimenti, si ha anche che

$$\begin{aligned}
(\mathcal{A}, e_1) \models A &\iff (\mathcal{A}, e_1) \models \exists x P(x) \\
&\iff \text{esiste } a \in U \text{ tale che } \mathcal{A} \models P(x)[\bar{a}/x] \\
&\iff \text{esiste } a \in U \text{ tale che } \mathcal{A} \models P(a) \\
&\iff \text{esiste: } a = e_2 \quad \mathcal{A} \models P(e_2) && \text{perché } I_{e_1}(\bar{e}_2) \in I_{e_1}(P) \\
&\iff \top && \text{perchè } e_2 \in \{e_2\}
\end{aligned}$$

L'interpretazione della funzione di Skolem (in questo caso la costante c) è irrilevante per la valutazione di A!

Al contrario, se si interpreta erroneamente c, A^S non è soddisfatta:

$$\begin{aligned}
(\mathcal{A}, e_1) \not\models A^S &\iff (\mathcal{A}, e_1) \not\models P[c/x] = P(c) \\
&\iff I_{e_1}(c) \notin I_{e_1}(P) \\
&\iff e_1 \notin \{e_2\}
\end{aligned}$$

Ossia, abbiamo dimostrato che:

$$A \not\equiv A^S \qquad \text{e} \qquad \not\models A \to A^S$$

e non possono, pertanto, essere logicamente equivalenti. Tuttavia sono equisoddisfacibili:

$$\models A^S \to A$$

Dimostrazione di $\models A^S \to A$. Siano A una formula in PNF e A^S la sua Skolemizzata:

$$A := \forall x_1 \cdots \forall x_n \exists y B \qquad\qquad A^S := \forall x_1 \cdots \forall x_n B[f(x_1, \cdots, x_n)/y]$$

E sia $(\mathcal{A}, h)$ la $\mathcal{L}^S$-struttura che soddisfa A^S, con $\mathcal{A} = (U, I)$. Allora:

$$\begin{aligned}
& (\mathcal{A}, h) \models A^S \\
\iff & (\mathcal{A}, h) \models B[f(x_1, \cdots, x_n)/y][\bar{a}_1/x_1] \cdots [\bar{a}_n/x_n] && \text{per ogni } (a_1, \cdots, a_n) \in U^n \\
\iff & (\mathcal{A}, h) \models B[\bar{a}_1/x_1] \cdots [\bar{a}_n/x_n][f(\bar{a}_1, \cdots, \bar{a}_n)/y] && \text{per ogni } (a_1 \cdots, a_n) \in U^n \\
& \text{visto che in } B \text{ non vi sono occorrenze libere di } x_1, \cdots, x_n \\
\iff & (\mathcal{A}, h) \models B[\bar{a}_1/x_1] \cdots [\bar{a}_n/x_n][\bar{b}/y] && \text{per ogni } (a_1, \cdots, a_n) \in U^n \\
& \text{posto } b := h(a_1, \cdots, a_n), \text{ ricordandosi che } I_h(f) = h \\
\iff & \text{esiste un } b \in U \text{ t.c. } (\mathcal{A}, h) \models B[\bar{a}_1/x_1] \cdots [\bar{a}_n/x_n][\bar{b}/y] && \text{per ogni } (a_1, \cdots, a_n) \in U^n \\
\iff & (\mathcal{A}, h) \models \forall x_1 \cdots \forall x_n \exists y B \\
\iff & (\mathcal{A}, h) \models A \\
\implies & \models A^S \to A
\end{aligned}$$

□

Questo non basta a dimostrare che siano equisoddisfacibili. Mostreremo ora che A e A^S sono equisoddisfacibili: si deve mostrare che se $(\mathcal{A}, h) \models A$ allora esiste $g : U^n \to U$ tale che $(\mathcal{A}, g) \models A^S$.

Dimostrazione.

$$\begin{aligned}
&(\mathcal{A},h) \models A \\
\iff &\text{esiste } b \in U \text{ t.c. } (\mathcal{A},h) \models B[\bar{a}_1/x_1]\cdots[\bar{a}_n/x_n][\bar{b}/y] && \text{per ogni } (a_1,\cdots,a_n) \in U^n \\
\iff &\text{esiste } b \in U \text{ t.c. } \mathcal{A} \models B[\bar{a}_1/x_1]\cdots[\bar{a}_n/x_n][\bar{b}/y] && \text{per ogni } (a_1,\cdots,a_n) \in U^n \\
&\text{perché non vi sono occorrenze del simbolo } h
\end{aligned}$$

Si definisce ora la funzione $g : U^n \to U$:

$$g(a_1,\cdots,a_n) := b \quad \text{per ogni } n\text{-pla } (a_1,\cdots,a_n) \in U^n$$

Ci si chiede ora quando vale

$$\begin{aligned}
&(\mathcal{A},g) \models A^S \\
\iff &(\mathcal{A},g) \models B[f(x_1,\cdots,x_n)/y][\bar{a}_1/x_1]\cdots[\bar{a}_n/x_n] && \text{per ogni } (a_1,\cdots,a_n) \in U^n \\
\iff &(\mathcal{A},g) \models B[\bar{a}_1/x_1]\cdots[\bar{a}_n/x_n][f(\bar{a}_1,\cdots,\bar{a}_n)/y] && \text{per ogni } (a_1,\cdots,a_n) \in U^n \\
\iff &(\mathcal{A},g) \models B[\bar{a}_1/x_1]\cdots[\bar{a}_n/x_n][\bar{b}/y] && \text{per ogni } (a_1,\cdots,a_n) \in U^n \\
\iff &(\mathcal{A},g) \models A && \text{per passaggio precedente}
\end{aligned}$$

pertanto

$$\text{se } (\mathcal{A},h) \models A \implies \text{esiste } g : U^n \to U \text{ t.c. } (\mathcal{A},g) \models A^S$$

Con h e g interpretazioni di f della prima—non usata—e della seconda $\mathcal{L}$-Struttura, rispettivamente. □

8.1.3 Forma Normale Congiuntiva

Sia A un $\mathcal{L}$-enunciato. Siamo passati da A ad $A^P \in PNF$ mantenendo l'equivalenza logica per poi passare a $A^S \in SNF$ mantenendo, in genere, solo l'equisoddisfacibilità. Ciò che manca per ricalcare il lavoro fatto riguardo la Logica Proposizionale, è portare il tutto in CNF:

$$A \xrightarrow{\equiv} A^P \xrightarrow{\text{equisod.}} (A^P)^S \rightsquigarrow ((A^P)^S)^C$$

Quindi, A^{P^S} è nella forma $\forall x_1 \cdots \forall x_n M$, dove in M non occorrono quantificatori. Inoltre, A^{P^S} è un $\mathcal{L}$-enunciato, pertanto $FV(M) \subseteq \{x_1,\cdots,x_n\}$.
Consideriamo ora ogni occorrenza di formula atomica $P(t_1,\cdots,t_n)$ in M come una lettera proposizionale; M può dunque essere pensata come una formula della Logica Proposizionale e il processo di trasformazione verso la CNF si ottiene applicando uno degli algoritmi visti, preservando equivalenza logica—con la distributività—o solo l'equisoddisfacibilità. Si ottiene, quindi:

$$M \xrightarrow[\text{equisod.}]{\equiv} M^C$$

Concludendo, quindi che il processo di "traduzione" è

$$A \xrightarrow{\equiv} A^P \xrightarrow{\text{equisod.}} (A^P)^S \xrightarrow[\text{equisod.}]{\equiv} ((A^P)^S)^C$$

8.2 Risoluzione Automatica

8.2.1 Preprocessamento

Siamo partiti dal chiederci se

$$\Gamma \overset{?}{\models} A \underset{(0)}{\longrightarrow} \Gamma, \neg A \text{ insod.?} \underset{(1)}{\xrightarrow{\text{se } \Gamma \text{ finito}}} \gamma_1 \wedge \cdots \wedge \gamma_n \wedge \neg A \text{ insod.?} \underset{(2)}{\longrightarrow} (\gamma_1 \wedge \cdots \wedge \gamma_n \wedge \neg A)^{PS^C} \text{ insod.?}$$

manca un'ultima trasformazione.
Infatti, sia $F \in PNF \cap SNF \cap CNF$ un enunciato in forma prenessa, Skolemizzata e congiuntiva:

$F = \forall x_1 \cdots \forall x_n C_1 \wedge C_2 \wedge \cdots \wedge C_u$
dove ogni clausola $C_i = (\ell_1 \vee \ell_2 \vee \cdots \vee \ell_v)$
dove ogni letterale $\ell_{i_j} = \begin{cases} P(t_1, \cdots, t_w) \\ \neg P(t_1, \cdots, t_w) \end{cases}$ per un qualche $P(t_1, \cdots, t_w)$ formula atomica

Operiamo, quindi, i seguenti passaggi di notazione:

- Omettiamo i quantificatori universali. Possiamo farlo per due ragioni:

  1. Scriverli esplicitamente non aggiunge nessuna informazione: sappiamo che tutte le variabili che occorrono sono quantificate universalmente.
  2. $\mathcal{A} \models F$ se e solo se $\mathcal{A} \models \forall[F]$, con F F.b.F.

- Notazione a sequenti.
  Si riscrive la formula $F = C_1 \wedge \cdots \wedge C_2$ come insieme delle sue clausole:

  $$S_F = \{C_1, \cdots, C_n\}$$

  Questa riscrittura ha vari vantaggi:

  – Ogni formula γ_i in $\Gamma \cup \{\neg A\}$ si può mettere in PNF, SNF e CNF indipendentemente, evitando di creare una formula gigante al passaggio (1) descritto sopra
  – Se Γ è infinito S_Γ sarà un insieme infinito di clausole finite

  Inoltre, si riscrive anche ogni clausola $C_i = \ell_{i_1} \vee \cdots \vee \ell_{i_v}$:

  $$C_i = \underbrace{\{\ell_{i_j} \text{ che occorrono negati, senza } \neg\}}_{=N_i} \cup \underbrace{\{\ell_{i_j} \text{ che occorrono positivi}\}}_{=P_i}$$

  $$C_i = N_i \implies P_i$$

  Quest'ultima è una notazione *a sequenti* della clausola C_i.

Esempio

Ci chiediamo se sia vero che

$$\forall x \exists y (R(x,y) \to Q(f(x))) \overset{?}{\models} \forall x \exists y (R(x,y) \to Q(f(y)))$$

Il primo passo è riportare il tutto in una domanda di insoddisfacibilità:

$$\forall x \exists y(R(x,y) \to Q(f(x))), \neg(\forall x \exists y(R(x,y) \to Q(f(y)))) \text{ insod.?}$$

Si procede, ora, mettendo in PNF, SNF e CNF le due formule. La prima,

$$\forall x \exists y(R(x,y) \to Q(f(x)))$$

è già in PNF. Si procede sostituendo a y un simbolo di funzione, creando la funzione di Skolem utilizzando un nuovo simbolo:

$$\forall x(R(x,s(x)) \to Q(f(x)))$$

e infine in CNF (o a sequenti):

$$\forall x(\neg R(x,s(x)) \vee Q(f(x)))$$

concludendo con

$$C_1 := \{\neg R(x,s(x)), Q(f(x))\}$$

e scritto in forma a sequenti

$$S_1 := \overbrace{R(x,s(x))}^{N_1} \implies \overbrace{Q(f(x))}^{P_1}$$

La seconda formula

$$\neg(\forall x \exists y(R(x,y) \to Q(f(y)))) = \exists x \forall y \neg(R(x,y) \to Q(f(y)))$$

è già in PNF, e per portarla in SNF si sostituisce a x un simbolo di funzione (costante), utilizzando un nuovo simbolo:

$$\forall y \neg(R(c,y) \to Q(f(y)))$$

e si porta in CNF:

$$\neg(R(c,y) \to Q(f(y))) \equiv \neg(\neg R(c,y) \vee Q(f(y))) \equiv R(c,y) \wedge \neg Q(f(y))$$

ottenendo due clausole:

$$C_2 = \{R(c,y)\}$$
$$C_3 = \{\neg Q(f(y))\}$$

e due sequenti:

$$S_2 : \implies \overbrace{R(c,y)}^{P_2}$$
$$S_3 : \overbrace{Q(f(y))}^{N_3} \implies$$

E si riscrive, infine,

$$\{R(x,s(x)) \implies Q(f(x)), \implies R(c,y), Q(f(y)) \implies\} \text{ insod.?} \tag{8.1}$$

Vedremo ora come rispondere a questa domanda.
Spoiler: è un problema semidecidibile, anche quando c'è un numero finito di clausole in notazione a sequenti. Questo perché potrebbero comunque codificare infinite molte informazioni!

8.2.2 Teoria di Herbrand

Sia S un insieme di clausole nella Logica dei Predicati; quindi, si immagina S ottenuto mettendo in PNF, SNF, CNF uno statement del tipo $\Gamma \models A$, di cui vogliamo stabilire la soddisfacibilità. Questo sarà eseguito riducendo S insoddisfacibile a un problema di insoddisfacibilità nella Logica Proposizionale.

Il primo passo è la seguente definizione:

Definizione (Astrazione Proposizionale)**.** L'**astrazione proposizionale** è una funzione che associa iniettivamente ad ogni *formula atomica* ground (quindi nella forma $P(t_1, \cdots, t_n)$ dove ogni t_i è un termine ground, ossia non appaiono variabili) una lettera proposizionale, denotata con $p_{P(t_1,\cdots,t_n)}$.

Per esempio, se una clausola C espressa in sequenti è

$$C : A_1, \cdots, A_u \implies B_1, \cdots, B_u$$

la sua astrazione proposizionale è la clausola ottenuta astraendo ogni lettera:

$$C : p_{A_1}, \cdots, p_{A_u} \implies p_{B_1}, \cdots, p_{B_u}$$

La **Teoria di Herbrand** ha suo culmine in quello che è ormai noto come **Teorema di Herbrand**, anche se nella letteratura ci sono delle perplessità riguardo il suo status di *teorema*:

Teorema (di Herbrand)**.** *Un insieme di clausole S della Logica dei Predicati è insoddisfacibile se e solo se esiste un insieme finito di istanze ground di clausole di S la cui astrazione proposizionale è (proposizionalmente) insoddisfacibile.*

Per approfondire la Teoria di Herbrand, introduciamo delle nozioni.

Definizione (Istanziazione Ground)**.** Sia S un insieme di clausole del Primo Ordine, e sia G un insieme di termini ground. Allora, con la notazione S/G indichiamo l'istanziazione ground di S su G, vale a dire l'insieme di tutte le clausole (che risulteranno ground) ottenute da ciascuna clausola $C \in S$ sostituendo in G ogni sua variabile x con qualche $t \in G$, in modo simultaneo, in tutti i modi possibili.

Esempio Sia

$$\begin{aligned} S &:= \{\{ \implies P(x), D(x)\}, \{P(x), D(x) \implies\}\} \\ &= \{\{P(x), D(x)\}, \{\neg P(x), \neg D(x)\}\} \\ &= (P(x) \vee D(x)) \wedge (\neg P(x) \vee \neg D(x)) \end{aligned}$$

Sia G ad esempio

$$G = \{a, b, c\}$$

Allora S/G è

$$\begin{aligned} S/G :=& \{\{ \implies P(a), D(a)\}, \{ \implies P(b), D(b)\}, \{ \implies P(c), D(c)\}, \\ & \{P(a), D(a) \implies\}, \{P(b), D(b) \implies\}, \{P(c), D(c) \implies\}\} \end{aligned}$$

Sia ora

$$G' = \{a, f(b, c)\}$$

allora S/G' è

$$S/G := \{\{ \implies P(a), D(a)\}, \{ \implies P(f(b,c)), D(f(b,c))\},$$
$$\{P(a), D(a) \implies\}, \{P(f(b,c)), D(f(b,c)) \implies\}\}$$

Siamo interessati ad istanziare S su un particolare insieme G, ossia quello che ci permette migliormente di studiare il nostro problema: questo insieme è chiamato *Universo di Herbrand*:

Definizione (Universo di Herbrand)**.** L'Universo di Herbrand H_S di un insieme di clausole S del Primo Ordine è definito come segue.
Sia $\mathcal{F}_S^0$ l'insieme di tutti i simboli di funzione—anche le costanti—che occorrono in S.

- Se $\mathcal{F}_S^0$ non contiene costanti, allora si pone $\mathcal{F}_S = \mathcal{F}_S^0 \cup \{z\}$, dove z è una costante nuova.
- Se $\mathcal{F}_S^0$ contiene qualche costante, allora $\mathcal{F}_S := \mathcal{F}_S^0$.

H_S è per definizione l'insieme di tutti i termini costruibili utilizzando i simboli di $\mathcal{F}_S$, che sono necessariamente ground.

Esempi

1 Sia $S := \{\{U(x) \implies M(x)\}, \{ \implies U(s)\}, \{M(s) \implies\}\}$, allora:

$$\mathcal{F}_S^0 = \{s\} \qquad \mathcal{F}_S = \{s\} \qquad H_S = \{s\}$$

2 Sia $S := \{\{ \implies P(x), D(x)\}, \{P(x), D(x) \implies\}\}$, allora:

$$\mathcal{F}_S^0 = \emptyset \qquad \mathcal{F}_S = \{z\} \qquad H_S = \{z\}$$

3 Sia $S := \{\{ \implies P(f(x)), D(x)\}\}$, allora:

$$\mathcal{F}_S^0 = \{f(\cdot)\} \qquad \mathcal{F}_S = \{f(\cdot), z\} \qquad H_S = \{z, f(z), f(f(z)), \cdots\}$$

Argomenteremo che S è insoddisfacibile se e solo se $(S/H_S)^*$, ossia l'astrazione proposizionale dell'istanziazione di S sul suo Universo di Herbrand H_S è insoddisfacibile; quest'ultimo insieme è spesso infinito, in quanto—oltre ai simboli di funzione "nativi" del linguaggio su cui è costruito S—è costruito a seguito della Skolemizzazione dell'insieme S.

I due teoremi "di Herbrand" enunciati seguono entrambi piuttosto facilmente dalla semantica di Herbrand, che è il vero cuore della questione.

8.2.3 Semantica di Herbrand

La semantica di Herbrand afferma che è necessario considerare solo le $\mathcal{L}$-strutture di Herbrand.

Definizione ($\mathcal{L}$-struttura di Herbrand)**.** Una $\mathcal{L}$-struttura di Herbrand è un tipo particolare di $\mathcal{L}$-struttura generale—o $\mathcal{L}$-struttura *di Tarski*.
Ricordiamo i vicoli che una $\mathcal{L}$-struttura di Tarski $\mathcal{A} = (U, I)$ deve rispettare:

- U scelto arbitrariamente t.c. $U \neq \emptyset$
- $I(c) \in U$, scelto arbitrariamente

- $I(f)$ (se $\beta(f) = n$) è una funzione $U^n \to U$ scelto arbitrariamente
- $I(P) \subseteq U^n$ (con $\alpha(P) = n$), scelto arbitrariamente

Una $\mathcal{L}$-struttura di Herbrand $\mathcal{H} = (H_S, H)$ deve, inoltre, rispettare:

- H_S è l'Universo di Herbrand, fissato.
  Per costruzione ha almeno un termine ground: $H_S \neq \emptyset$
- $H(c) \in H_S$, t.c. $H(c) = c$
- $H(f)$ (se $\beta(f) = n$) è una funzione $H_S^n \to H_S$ t.c.

$$(H(f))(t_1, \overbrace{\cdots}^{\in H_S}, t_n) := f(t_1, \cdots, t_n)$$

- $H(P) \subseteq H_S^n$ (con $\alpha(P) = n$), scelto arbitrariamente.

Si noti che in una $\mathcal{L}$-struttura generica si ha un universo d'interesse, formato da oggetti fortemente semantici (e.g., "numeri", "orbite di pianeti", "oggetti geometrici",...).
H_S, invece, è un universo di termini, ed è quindi puramente sintattico. Con la semantica di Herbrand, sintassi e semantica diventano una cosa sola! Infatti c ed f, oggetti sintattici, vengono interpretati in loro stessi, ma nella loro forma semantica.

Teorema 4 (Teorema Fondamentale della Teoria di Herbrand)**.** *Sia S un insieme di clausole del Primo Ordine, allora:*

$$\textit{esiste } \mathcal{A} = (U, I) \textit{ t.c. } \mathcal{A} \models S \iff \textit{esiste } \mathcal{H} = (H_s, H) \textit{ t.c. } \mathcal{H} \models S$$

S è soddisfacibile, ossia per S esiste un modello (di Tarski) $\mathcal{A} = (U, I)$ tale che $\mathcal{A} \models S$ se e solo se S ha un modello di Herbrand, ossia se esiste una $\mathcal{L}$-struttura di Herbrand $\mathcal{H} = (H_s, H)$ tale che $\mathcal{H} \models S$.

Ora, ci chiediamo quale sia la relazione tra $(S/H_S)^*$ e $\mathcal{H}_S \models S$.

Lemma 5.

$$\mathcal{H}_S \models S \iff v_{\mathcal{H}_S} \models (S/H_S)^*$$

Dimostrazione. ($\Longrightarrow$)
Sia $v_{\mathcal{H}_S}$: Lettere Proposizionali $\to \{0,1\}$, un assegnamento tale che:

$$v_{\mathcal{H}_S}(p_{P(t_1,\cdots,t_n)}) = \begin{cases} 1 \iff \mathcal{H}_S \models P(t_1, \cdots, t_n) \\ 0 \iff \mathcal{H}_S \not\models P(t_1, \cdots, t_n) \end{cases}$$

($\Longleftarrow$)
Viceversa, data v : Lettere Proposizionali $\to \{0,1\}$, si definisce:

$$\mathcal{H}_v = (H_S, H_v) = \begin{cases} (t_1, \cdots, t_n) \in H_v(P) \iff v(p_{P(t_1,\cdots,t_n)}) = 1 \\ (t_1, \cdots, t_n) \notin H_v(P) \iff v(p_{P(t_1,\cdots,t_n)}) = 0 \end{cases}$$

□

Teorema Riassuntivo (Herbrand, compattezza, completezza, calcolo Refutazionale R^*, DPP e DPLL)

Dato un insieme S di clausole della Logica del Primo Ordine, è equivalente affermare:

1. S è insoddisfacibile
2. S non ha modelli di Herbrand
   [*per Thm 4 Fondamentale della Teoria di Herbrand*]
3. $(S/H_S)^*$ è proposizionalmente insoddisfacibile
   [*per Lemma* 5]
4. Esiste $S' \subseteq_\omega (S/H_S)^*$ tale che S' è proposizionalmente insoddisfacibile
   [*per Thm 1 di Compattezza Proposizionale*]
5. Esiste $H' \subseteq_\omega H_S$ tale che $(S/H')^*$ è finito e proposizionalmente insoddisfacibile
   [*per compattezza e teoria degli insiemi*]
6. Esiste $H' \subseteq_\omega H_S$ e $h \geq 0$ tale che $\square \in \mathbb{R}^h((S/H)^*)$
   [*per Thm 2 di Completezza Refutazionale*]
7. Esiste $H' \subseteq_\omega H_S$ e una refutazione $DPP \vdash_{\mathbb{R}} (S/H')^*$ o anche $DPLL \vdash_{\mathbb{R}} (S/H')^*$
   [*per Thm di Completezza Refutazionale di DPP (3) e DLPP (5.2.3)*]

La catena $1 \cdots 7$ fornisce un algoritmo per *semidecidere* l'insoddisfacibilità di S. Questo perché non sappiamo come prendere $H' \subseteq_\omega H_S$, e bisogna tentare con S' via via più grandi. Ci si ferma solo quando se ne trova un $(S/H')^*$ refutabile.

8.2.4 Metodi Refutazionali

Ritorniamo, ora, a rispondere alla domanda 8.1, ovvero:

$$S = \{R(x, s(x)) \implies Q(f(x)), \implies R(c,y), Q(f(y)) \implies\} \text{ insod.?}$$

Si procede alla risoluzione del problema dell'insoddisfacibilità, seguendo i passi definiti precedentemente:

$$\mathcal{F}_S = \{c, s(\cdot), f(\cdot,\cdot)\} \quad H_S = \{c, s(c), f(c), f(s(c)), s(f(c)), s(s(c)), \cdots \overbrace{f(\cdots s(\cdots(c)\cdots))}^{\text{numero finito di } s \text{ e } f}, \cdots\}$$

Pertanto, è impossibile costruire $(S/H_S)^*$ direttamente, in quanto è anch'esso sarebbe infinito. Si passa, quindi, a considerare sottoinsiemi finti di H_S, per esempio:

(1) $H = \{c\}$:

$$(S/H)^* = \{p_{R(c,s(c))} \implies p_{Q(f(c))}, \implies p_{R(c,c)}, p_{Q(f(c))} \implies\}$$

Possiamo provare ad effettuare la risoluzione

$$\mathbb{R} \text{ con } \ell = p_{Q(f(c))} \; \frac{p_{R(c,s(c))} \implies p_{Q(f(c))} \qquad p_{Q(f(c))} \implies}{p_{R(c,s(c))} \implies}$$

e rimaniamo con

$$(S/H)^* = \{p_{R(c,s(c))} \implies, \implies p_{R(c,c)}\}$$

coi quali non si può fare niente, in quanto le astrazioni proposizionali sono diverse.
NOTA: d'ora in poi evitiamo di applicare le astrazioni proposizionali e considereremo i predicati come dei termini ground

(2) $H' = \{c, s(c)\}$:

$$\begin{aligned}(S/H')^* = \{&R(c,s(c)) \implies Q(f(c)),\ \implies R(c,c), Q(f(c)) \implies,\\ &R(s(c),s(s(c))) \implies Q(f(s(c))),\ \implies R(c,s(c)), Q(f(s(c))) \implies\}\end{aligned}$$

nuovamente:

$$\mathbb{R}\ \text{con}\ \ell = Q(f(c))\ \dfrac{\mathbb{R}\ \dfrac{R(c,s(c)) \implies Q(f(c)) \qquad Q(f(c)) \implies}{R(c,s(c)) \implies} \qquad \implies R(c,s(c))}{\implies (\equiv \square)}$$

pertanto $(S/H')^*$ è insoddisfacibile.
Quindi S/H_S è insoddisfacibile.
S è insoddisfacibile ed il problema iniziale (8.1) era vero.

Tuttavia, già da questo esempio si può intuire che c'è qualche difetto: si vorrebbe considerare sottoinsiemi finiti $H \subseteq H_S$ utili, dai quali si generano solo le clausole che potrebbero essere utili, senza generare clausole inutili.

Esempio: "paradosso" dell'uomo col cappello Si asserisce che la seguente è una verità logica:

$$\models \exists x(P(x) \to \forall y P(y)) \qquad \begin{array}{l}\text{c'è un } x \text{ che se si mette il cappello,}\\ \text{allora tutti gli altri } y \text{ lo mettono}\end{array}$$

Ma è davvero così? Se lo fosse, allora:

$$\neg(\exists x(P(x) \to \forall y P(y))) \text{ è insodd.}$$

$$\begin{aligned}&\forall x \neg(P(x) \to \forall y P(y))\\ \equiv\ &\forall x \neg(\neg P(x) \vee \forall y P(y))\\ \equiv\ &\forall x(P(x) \wedge \neg(\forall y P(y)))\\ \equiv\ &\forall x \exists y(P(x) \wedge \neg P(y))\\ \textit{equisodd}.\ &\forall x(P(x) \wedge \neg P(f(x))) \qquad \text{per Skolemizzazione}\end{aligned}$$

Che si traduce in:

$$S = \{\implies P(x), P(f(x)) \implies\} \text{ è insodd.}$$

con:

$$\mathcal{F}_S = \{f(\cdot), \dot{z}\} \qquad H_S = \{\dot{z}, f(\dot{z}), f(f(\dot{z})), \cdots f^{(k)}(\cdots), \cdots\}$$

Si cerca, quindi, un $H \subseteq_\omega H_S$ che dimostri l'isoddisfacibilità:

(1) $H = \{\dot{z}\}$:

$$S = \{\implies P(\dot{z}), P(f(\dot{z})) \implies\}$$

ma non si può fare nulla.

(2) $H = \{\mathfrak{z}, f(\mathfrak{z})\}$:

$$S = \{ \Longrightarrow P(\mathfrak{z}), P(f(\mathfrak{z})) \Longrightarrow \} \cup \{ \Longrightarrow P(f(\mathfrak{z})), P(f(f((\mathfrak{z})))) \Longrightarrow \}$$

A questo punto si può risolvere:

$$\frac{P(f(\mathfrak{z})) \Longrightarrow \qquad \Longrightarrow P(f(\mathfrak{z}))}{\Longrightarrow}$$

Pertanto S è insoddisfacibile, e di contro è vero che

$$\models \exists x(P(x) \to \forall y P(y))$$

E ci sono due $\mathcal{L}$-strutture che la rendono vera:

- dove esiste un $x \notin P$ (e perciò si può implicare tutto dal falso)
- dove ogni $x \in U$ è anche $x \in P$ (e perciò ogni elemento "ha il cappello")

Difetti

Sia

$$S = \{ \Longrightarrow P(x,y), Q(x) \}, \{ P(x, f(x)) \Longrightarrow \}$$

si costruiscono

$$\mathcal{F}_S = \{f(\cdot), \mathfrak{z}\} \qquad\qquad H_S = \{\mathfrak{z}, f(\mathfrak{z}), f(f(\mathfrak{z})), \cdots\}$$

Una possibile risoluzione si trova ponendo, ad esempio, $x = \mathfrak{z}$, $y = f(\mathfrak{z})$

$$\frac{\Longrightarrow P(\mathfrak{z}, f(\mathfrak{z})), Q(\mathfrak{z}) \qquad P(\mathfrak{z}, f(\mathfrak{z})) \Longrightarrow}{\Longrightarrow Q(\mathfrak{z})}$$

Allo stesso modo se ne può ha un'altra con $x = f(\mathfrak{z})$, $y = f(f(\mathfrak{z}))$:

$$\frac{\Longrightarrow P(f(\mathfrak{z}), f(f(\mathfrak{z}))), Q(f(\mathfrak{z})) \qquad P(f(\mathfrak{z}), f(f(\mathfrak{z})) \Longrightarrow}{\Longrightarrow Q(f(\mathfrak{z}))}$$

e, continuando così, si possono ottenere infinite altre risoluzioni.

Per il nostro apparato concettuale, tutte queste risoluzioni sono semplici istanziazioni di variabili in termini ground, tuttavia noi umani vediamo qualcosa in più: vi è infatti un *pattern* che ci piacerebbe catturare:

$$y \mapsto f(x)$$

questa è una sostituzione ma non è una istanziazione, in quanto y non "diventa" un termine ground, ma possiede una variabile. Tuttavia, se si riesce a disegnare un quadro in cui questo è lecito, si può risolvere direttamente

$$\frac{\Longrightarrow P(x, f(x)), Q(x) \qquad P(x, f(x)) \Longrightarrow \qquad : y \mapsto f(x)}{\Longrightarrow Q(x)}$$

E da $\Longrightarrow Q(x)$ si può ritrovare, con le istanziazioni delle variabili in termini ground, tutta la famiglia di risoluzioni.

8.2.5 Teoria delle Sostituzioni e Unificazione

Quali clausole del Primo Ordine generano per istanziazione la clausola vuota ($\Longrightarrow$)?
Solo la clausola vuota stessa. L'approccio proposto poco fa, allora, è promettente, poiché:

- Ci sono delle sostituzioni *ottime* che sono "facili" da calcolare e trovare
- ("Lifting") Facendo prima le risoluzioni a livello non ground si riesce a posticipare la fase d'istanziazione (ground) o, addirittura, non avere più bisogno di farla perchè si è trovata la clausola vuota in fase di *lifting*

Abbiamo già discusso di sostituzioni riguardo la Logica Proposizionale, in cui si sostituivano delle lettere proposizionali con una Formula. Nella Logica dei Predicati, abbiamo discusso di sostituzione di variabili individuali con termini. Anche ora discuteremo di qualcosa di simile, ma ora la notazione cambia in quanto sarà proprio la sostituzione il punto focale del discorso.

Definizione (Sostituzione)**.** Una **sostituzione** σ è una mappa

$$\sigma : Var \to \tau_{\mathcal{L}}$$

con $\tau_{\mathcal{L}}$ che indica l'insieme dei termini costruibili sul linguaggio $\mathcal{L}$, anche non ground; σ ha una proprietà importante che rende la computazione delle sostituzioni molto semplice: l'insieme delle variabili x tali che sono "mosse" dalla sostituzione, ossia $x \neq \sigma(x)$, è finito. L'insieme delle variabili "mosse" viene chiamato dominio di σ.

Se il dominio di σ è

$$dom(\sigma) = \{x_1, \cdots, x_k\}$$

allora si può visualizzare σ nel seguente modo:

$$\sigma : x_1 \mapsto t_1, \cdots, x_k \mapsto t_k$$

Questa notazione sostituisce la notazione $([t_i/x_i])$ utilizzata fino ad ora.

Definizione (Estensione Canonica)**.** Una sostituzione σ si estende canonicamente ad una mappa

$$\hat{\sigma} : \tau_{\mathcal{L}} \mapsto \tau_{\mathcal{L}}$$

con la solita definizione di estensione canonica definita induttivamente sugli $\mathcal{L}$-termini:

- **base**:
$$\hat{\sigma}(x) := x \qquad \text{per ogni } x \in Var$$
- **passo**:
$$\hat{\sigma}(f(t_1, \cdots, t_n)) := f(\hat{\sigma}(t_1), \cdots, \hat{\sigma}(t_n))$$

Si può, inoltre, estendere σ non solo ai termini ma a tutte le espressioni che ci servono:

- sia $P(t_1, \cdots, t_n)$ un'atomica del nostro linguaggio. Allora si definisce
$$\hat{\sigma}(P(t_1, \cdots, t_n)) := P(\hat{\sigma}(t_1), \cdots, \hat{\sigma}(t_n))$$
- sia E una espressione d'interesse generica. Allora
$$\hat{\sigma}(E)$$
è l'espressione sostituendo tramite σ simultaneamente tutte le occorrenze dei termini x_i che appaiono in E con $\sigma(x_i)$

Scriveremo spesso $E\sigma$ per indicare $\hat{\sigma}(E)$.

Proprietà delle Sostituzioni

- **Composizione** Se $\gamma, \tau : Var \to \tau_{\mathcal{L}}$ la sostituzione composta $\sigma\tau$ è data da

  $$\sigma\tau(x) = \hat{\tau}(\sigma(x))$$

  Ovviamente $\hat{\sigma}\tau$ applicata su una certa espressione E, denotato quindi $E\sigma\tau$ si ottiene applicando $\hat{\tau}(\sigma(x_i))$ ad ogni variabile occorrente in E.

- **Sostituzione Identica** la sostituzione *id* è definita

  $$id(x_i) = x_i \quad \text{per ogni } x_i \in Var$$

  e ha dominio vuoto ($dom(id) = \emptyset$); gode della proprietà di composizione seguente:

  $$\sigma id = id\sigma = \sigma \text{ per ogni } \sigma : Var \to \tau_{\mathcal{L}}$$

  quindi svolge il ruolo di elemento neutro.

- **Rinomine** una rinomina è una sostituzione $\rho : Var \to \tau_{\mathcal{L}}$ che è una permutazione del suo dominio, ossia

  $$\rho(x_i) = x_j \qquad \text{con } \rho \text{ biettiva sul suo dominio}$$

  quindi ρ scambia un set di variabili.
  Inoltre ogni rinomina è invertibile, cioè esiste una sostituzione ρ' tale che $\rho'(\rho(x_i)) = x_i$, in altre parole $\rho\rho' = \rho'\rho = id$.

- **Sostituzione Generale** σ è detta *più generale* di τ se esiste una sostituzione δ tale che $\sigma\delta = \tau$.
  Per esempio, se

  $$\sigma : x \mapsto f(z),\ y \mapsto z$$
  $$\tau : x \mapsto f(z),\ y \mapsto c,\ z \mapsto c$$

  allora σ è più generale di τ ($\sigma \geq \tau$) perché $\tau = \sigma\delta$ con $\delta : z \mapsto c$:

  $$\sigma\delta : x \underbrace{\overset{\sigma}{\mapsto} f(z) \overset{\delta}{\mapsto}}_{\tau} f(c)$$
  $$\sigma\delta : y \underbrace{\overset{\sigma}{\mapsto} z \overset{\delta}{\mapsto}}_{\tau} c$$
  $$\sigma\delta : z \underbrace{\overset{\sigma}{\mapsto} z \overset{\delta}{\mapsto}}_{\tau} c$$

- **Preordine** La relazione "essere più generale" ($\sigma \geq \tau$) è riflessiva e transitiva, ma in generale non è antisimmetrica. Per questo non può essere una relazione d'ordine, ma di *preordine*.
  Non è antisimmetrica perché è possibile che $\sigma \neq \tau$ anche se $\sigma \geq \tau$ e al contempo $\tau \geq \sigma$, perché hanno sostituzioni distinte. Queste sostituzioni se sono distinte, tuttavia, sono solo rinomine ρ_1, ρ_2 tali che

  $$\sigma\rho_1 = \tau \quad \tau\rho_2 = \sigma$$

- **Equivalenza Sostituzionale** Se $\sigma \geq \tau$ e $\tau \geq \sigma$, dichiariamo che $\sigma \equiv \tau$.

- **Ordine Parziale** L'insieme delle sostituzioni con la relazione "essere più generale" quozientato rispetto a $\equiv$ è un *poset*, ossia

$$\{[\sigma]_{\equiv} : \sigma : Var \to \tau_{\mathcal{L}}\}$$

  è un insieme con ordine parziale rispetto a $\geq$.

L'ultimo concetto che ci serve per arrivare alla definizione della Risoluzione Sollevata o Liftata è il seguente:

Definizione (Unificazione)**.** Un problema di Unificazione è una lista finita di coppie di termini

$$t_1 \stackrel{?}{=} u_1, \cdots, t_n \stackrel{?}{=} u_n$$

La soluzione ad un problema di unificazione è una sostituzione che verifica tutte le uguaglianze nella lista di coppie di termini, ossia una sostituzione

$$\sigma : Var \to \tau_{\mathcal{L}}$$

tale che

$$\hat{\sigma}(t_1) = \hat{\sigma}(u_1), \cdots, \hat{\sigma}(t_n) = \hat{\sigma}(u_n)$$

Definizione (Unificatore)**.** Una soluzione $\mu : Var \to \tau_{\mathcal{L}}$ è detta **unificatore** di massima generalità (*most general unifier*) se e solo se μ è più generale di ogni altra soluzione al problema di unificazione.

Teorema (di Unificazione)**.** *Ogni problema di unificazione U o ammette una soluzione μ che è un MGU—i.e., Most General Unifier—oppure non ammette alcuna soluzione.*

Si noti che se $mgu(U)$ esiste non è necessariamente unico, in quanto è un unica classe di equivalenza $mgu(U) = [\mu]_{\equiv}$.

Esistono algoritmi che, su input un problema di unificazione U, trovano velocemente $\mu = mgu(U)$ se esiste, e si fermano se $mgu(U)$ non esiste.

Ci interesserà risolvere problemi del tipo

$$P(t_1, \cdots, t_m) \stackrel{?}{=} Q(s_1, \cdots, s_n)$$

per P simbolo di predicato m-ario e Q simbolo di predicato n-ario. In altre parole ci interessa unificare due atomiche. Si noti che il problema appena esposto è unificabile se $P = Q$ e dunque $m = n$. Il problema si riduce a

$$P(t_1, \cdots, t_n) \stackrel{?}{=} P(s_1, \cdots, s_n)$$

che a sua volta si riduce a

$$U : t_1 \stackrel{?}{=} s_1, \cdots, t_n \stackrel{?}{=} s_n$$

8.2.6 Calcolo $\mathbb{R}$ della risoluzione liftata

Si supponga di avere le seguenti regole scritte in forma a sequenti:

$$\{\{\Gamma \implies \Delta, A\}, \{B, \Gamma' \implies \Delta'\}\}$$

dove le variabili occorrenti in $\Gamma \implies \Delta, A$ sono disgiunte da quelle in $B, \Gamma' \implies \Delta'$ (il che si può sempre rispettare usando le rinomine).

A e B sono proposizioni atomiche, mentre $\Gamma, \Delta, \Gamma', \Delta'$ sono insiemi di proposizioni atomiche.

Se si riesce a calcolare un $\mu = mgu(A, B)$, allora si può risolvere la **risoluzione liftata** su $\mathbb{R}(\mu(A), \mu(B))$:

$$\frac{\Gamma \implies \Delta, A \qquad B, \Gamma' \implies \Delta'}{\mu(\Gamma), \mu(\Gamma'), \implies \mu(\Delta), \mu(\Delta')}$$

Questa regola incarna il calcolo $\mathbb{R}$. Da sola non basta a garantire la completezza refutazionale, ma è necessario aggiungere almeno una delle seguenti due regole di fattorizzazione:

- **Fattorizzazione Destra**: se si riesce a calcolare un $\mu = mgu(A, B)$, allora si può unificare A e B tenendo un'unica copia dei due:

$$\frac{\Gamma \implies \Delta, A, B}{\Gamma\mu \implies \Delta\mu, A\mu} \; \mu = mgu(A, B)$$

- **Fattorizzazione Sinistra**: uguale, ma lavorando sulla parte a sinistra di $\implies$

$$\frac{\Gamma, A, B \implies \Delta}{\Gamma\mu, A\mu \implies \Delta\mu} \; \mu = mgu(A, B)$$

Una refutazione di un insieme di clausole S nel calcolo $\mathbb{R}$ è una successione di clausole $C_1, C_2, \cdots, C_u$ tale che:

1. $C_u = \implies$ (clausola vuota)
2. per ogni C_i:
   - o C_i è un membro di S, eventualmente rinominato
   - o C_i è ottenuto da eventuali rinomine di C_j, C_k con $(j, k < i)$ per risoluzione liftata o fattorizzazione (destra o sinistra)

Il calcolo $\mathbb{R}$ è *refutazionalmente completo* per ogni insieme di clausole S, quindi anche per tutta la logica del Primo Ordine: se si ha un problema del tipo

$$\Gamma \overset{?}{\models} A \rightsquigarrow \overset{\text{insieme di clausole del Primo Ordine}}{S \text{ insod.?}} \rightsquigarrow S \overset{?}{\vdash}_{\mathbb{R}} \implies$$

Questo ci da, quindi:

- *completezza*: Se $S \vdash_{\mathbb{R}} \implies$, allora S è insodd. e $\Gamma \models A$
- *correttezza*: se $\Gamma \models A$, allora $S \vdash_{\mathbb{R}} \implies$;
  ovvero S è insodd. e pertanto deve essere refutabile col calcolo $\mathbb{R}$

Considerazioni Finali

La completezza refutazionale di $\mathbb{R}$ mostra che la logica del Primo Ordine è *almeno* semidecidibile. "Almeno" in quanto il Teorema di Church afferma che non si può fare di meglio, ossia la Logica del Primo Ordine è semidecidibile ma non decidibile.

Siano $\Gamma \models A$ e S l'insieme, anche infinito, derivato di clausole; con H_S l'universo di Herbrand relativo. Sia

$$S_1 \subseteq_\omega S_2 \subseteq_\omega \cdots \subseteq_\omega S$$

un insieme di sottoinsiemi finiti di S tali che $\bigcup_{i\in\omega} S_i = S$ e per ogni $i \in \omega$ sia H^i l'universo di Herbrand di S_i. Chiaramente, si ha anche che

$$H^1 \subseteq H^2 \subseteq \cdots \subseteq H_s$$

e $\bigcup_{i\in\omega} H^i = H_s$. Per ogni i si definisce

$$H^i_1 \subseteq_\omega H^i_2 \subseteq_\omega \cdots \subseteq_\omega H^i$$

Per ogni $k \in \omega$ e per ogni i, j tale che $i + j = k$ si applica DPP o DPLL a S_i/H^i_j. Se si trova la clausola vuota, allora S è insoddisfacibile. Se non si trova, allora si passa a $k = k + 1$. In questo modo si riescono a perlustrare tutti i sottoinsiemi finiti dello spazio di ricerca, e se S è insoddisfacibile prima o poi si trova.

La morale sottesa a questa considerazione è che se S è insoddisfacibile prima o poi si mostra che lo è. Se S è soddisfacibile non ci sono criteri generali per fermare l'algoritmo.

Definizione (Completezza Formale)**.** Una teoria Γ si dice Formalmente Completa se e solo se per ogni enunciato A è vero o $\Gamma \vdash A$ oppure $\Gamma \vdash \neg A$. Per Completezza e Correttezza Semantica, questo equivale a dire che $\Gamma \models A$ oppure $\Gamma \models \neg A$.

Teorema (Teorema di Incompletezza di Gödel)**.** *Se T è una teoria assiomatizzabile, consistente e che esprime l'Aritmetica di Peano, allora esiste ϕ tale che*

$$T \nvdash \phi \qquad T \nvdash \neg\phi$$

ma $\mathcal{N} \models \phi$ con $\mathcal{N} = (\mathbb{N}, +, \times, 0, s)$. Quindi è formalmente incompleta.

Definizione (Aritmetica di Presburger)**.** Sostanzialmente, è l'Aritmetica di Peano (*PrA*) senza moltiplicazione.
PrA è formalmente completa—e quindi anche decidibile—in tempo doppiamente esponenziale.

Definizione (Aritmetica di Robinson)**.** Sostanzialmente, è l'Aritmetica di Peano senza schema di induzione, più un assioma che sostituisce alcune particolarità dello schema di induzione:

$$\forall x(\neg(x = 0) \to \exists y(s(y) = x))$$

Si può dimostrare che, pur senza induzione, è anch'essa formalmente incompleta, così come tutte le sue estensioni consistenti e assiomatizzabili. Perciò anche PA stessa.
Infatti RA—come PA—è semidecidibile ma non decidibile, tuttavia RA è finitamente assiomatizzabile: ha solo 7 assiomi.

Teorema (Teorema di Church)**.** *La Logica del Primo Ordine è indecidibile*

Dimostrazione. Con $RA = \{A_1, A_2, \cdots, A_7\}$ sia C un enunciato nell'Aritmetica di Robinson stessa

$$C = A_1 \wedge \cdots \wedge A_7$$

e A un enunciato arbitrariamente scelto.
Si suppone, per assurdo, che la Logica del Primo Ordine sia decidibile. Allora:

- $C \to A$ è un enunciato del Primo Ordine,
- dunque è decidibile se $\models C \to A$ oppure $\not\models C \to A$.
  - se fosse $\models C \to A$, allora avremmo

    $$\frac{C \qquad C \to A}{A}\ \text{Modus Ponens}$$

    e che quindi $RA \models A$
  - se, invece, fosse $\not\models C \to A$, allora per il teorema di deduzione $C \not\models A$, dunque $RA \not\models A$

Dunque, abbiamo deciso se $RA \models A$ oppure $RA \not\models A$, cioè: RA è decidibile.
Assurdo. Pertanto, la Logica del Primo Ordine non è decidibile. □

Mentre l'insieme degli enunciati insoddisfacibili è enumerabile, l'insieme degli enunciati soddisfacibili non è enumerabile, poiché se così fosse ci sarebbe un modo per rendere decidibile una qualunque aritmetica, che è assurdo.