Logica 2022/2023

Blascovich Alessio
alessio.blascovich@studenti.unitn.it

Contents

Part I

Modellazione

Chapter I.1

Rappresentazione

I.1.1 Rappresentazione mentale

- **Mondo:** Il mondo è ciò che assumiamo esista.
- **Rappresentazione mentale:** Una rappresentazione mentale è una parte del mondo che descrive il mondo stesso, quindi esiste corrispondza tra cosa esiste nel mondo e una rappresentazione mentale.
  La rappresentazione mentale permette di agire nel mondo e di interagire con altri umani.
- **Rappresentazione mentale analogica:** Una rappresentazione analogica è una semplice rappresentazione che si basa su ciò che percepiamo con i nostri sensi.
- **Rappresentazione linguistica mentale:** La rappresentazione che descrive il contesto di una rappresentazione analogica.
  Viene usata per:
  - **Descrivere:** cosa è successo nella rappresentazione mentale analogica.
  - **Comunicare:** con altri umani approposito della rappresentazione e quindi del mondo.
  - **Imparare:** da cosa viene descritto.
  - **Motiva:** cioè cerca di distinguere quello che non sa da quello che già conosce.

I.1.2 Rappresentazione

- **Rapresentazione:** Ha due proprietà principali:
  - Più umani la percepiscono (come la rappresentazione mentale).
  - E' una parte dello stesso mondo che descrive.
- **Rappresentazione analogica:** Fa una mappatura uno a uno del mondo, considerando anche il contesto in cui ci si trova.
- **Linguistic representation:** Non la ho capita, ste cazzate filosofiche.

I.1.3 Modellazione

I.1.3.1 Modellazione

- **Modellazione:** La modellazione è l'attività che porta alla realizzazione di una rappresentazione attraverso una serie di rappresentazioni mentali intermedie.
- **Teoria:** Identifica la rappresentazione linguistica prodotta da attività di modellazione.
- **Modello:** E' una rappresentazione analogica data da un modello.
  Possiamo anche dire che sia il modello inteso dalla teoria e che T sia la teoria del modello M.
- **Modello mondiale:** Un modello mondiale M_W è una cappia data da teoria T e modello M_T legati dalla formula: $M_W = \langle T, M_T \rangle$.

I.1.3.2 Osservazioni

Nella maggior parte dei modelli mondiali, usati in applicationi CS/AI, è definita solo la teoria.
Il modello inteso è implicito visto che la rappresentazione mentale è simile tra le perone.
Questo modello viene seguito quando il costo degli errori è accettabile.

Alcune volte il modello semantico è sviluppato in seguito e non copre totalmente la semantica.

Alcune volte la mancanza di un modello esplicito porta alla genesi di un dialetto.
Nel caso particolare del IA questa mancanza fa compiere alla macchina azioni imprevedibili perchè impedisce alla macchina di capire quando sbaglia.

Chapter I.2

Modellazione mentale

I.2.1 Conoscenza semantica

I.2.1.1 Teoria e modelli

- **Denotazione e semantiche:** Diremo che la teoria T denota il suo modello inteso M e scriveremo $T = D(M)$.. In modo alternativo possiamo dire che il modello M è la semantica intesa da T e scriveremo $M = S(T)$.

I.2.1.2 Frasi e fatti

- **Fatto:** Un modello $M = \{f\}$ è un insieme di fatti f, dove i fatti sono una rappresentazione analogica una parte della parte di mondo descritta da M.

- **Frase:** Una teoria $T = \{s\}$ è un insieme di frasi s, dove una frase è una rappresentazione linguistica di un insieme di fatti f.

- **Denotazione e semantiche:** Diremo che una frase s denota un fatto f e scriveremo $s = D(f)$.
  Alternarivamente, un fatto f è la semantica intesa da s e scriveremo $f = S(s)$.

I.2.1.3 Osservazioni

Assumiamo sempre che una frase $s \in T$ descriva uno o più fatti $f \in M$.
La nozione della descrizione linguistica e, in particolare, quella della toeria e della frase, possiamo sempre assumere si riferisca (la nozione) a una descrizione mentale possibilmente resa oggettiva tramite un modello che la descrive.

Non esistono rappresentazioni linguistiche senza referenze al mondo.

Una frase $s = D(f)$ può denotare più fatti la D è una relazione e non necessariamente una funzione.
In questi casi diremo che s è ambigua o polisemica (polisemica=esprime più significati).

Un fatto $f = S(s)$ può essere la semantica di più frasi s, allora anche S è una relazione e non per forza una funzione.
Per una fatto f ci sono infiti modi di denotarlo e per questo le frasi che denotano lo stesso fatto vengono dette sinonimi.

Se D e S sono entrambe funzioni allora sono una l'inversa dell'altra.

I.2.1.4 Toerie e modelli

- **Modello minore:** Siano due modelli $M = \{f\}$ e $\widehat{M} = \{f\}$ tali che $\widehat{M} \subseteq M$.
  Diremmo che $\widehat{M}$ è minore rispetto a M e che un fatto f tale che $f \in M$ e $f \notin \widehat{M}$ è detto al di fuori di $\widehat{M}$.

- **Teorie e modelli:** Sia $M = \{f\}$ un insieme diu fatti e $T = \{s\}$ un insieme di sequenze.
  Sia M_T un insieme minore di M, allora T è una teoria del modello M_T se e solo se $\forall\ s \in T$ abbiamo $s = D(f)$ per qualsiasi $f \in M_T$.
  Possiamo anche dire che M_T è un modello di T.

I.2.1.5 Osservazioni

Esistono modelli e teorie che sono dei singoletti, ma i due eventi sono indipendenti.

Modelli di fatti diversi possono rappresentare lo stesso mondo a diversi livelli di astrazione, analogamente vale anche per le teorie di frasi corrispondenti.

Più il modello è astratto meno dettagli contiene rispetto a un modello meno astratto.
Un modello meno astratto può comunque rappresentare una gran parte del mondo.

Un modello piccolo rappresenta una piccola parte del mondo, analogamente fanno le teorie.

I.2.1.6 Correttezza e completezza

- **Correttezza:** Sia $M_T \subseteq M$, allora una teoria T del modello M_T è corretta rispetto a M_T se e solo se:
  $\forall\ s \in T\ \exists f \in M_T \mid f = S(s)$, viene detta incorretta altrimenti.
- **Completezza:** Sia $M_T \subseteq M$, allora una teoria T di un modello M_T vinene detta dette corretta rispetto a M_T se e solo se $\forall\ f \in M_T\ \exists s \in T \mid s = D(f)$, viene detta icompleta altrimenti.
- **Correttezza e completezza:** Sia $M_T \subseteq M$, allora un teoria T di un modello M_T viene detta corretta e completa se rispetta entrambe le due condizioni.

I.2.1.7 Osservazioni

La maggior parte delle volte una toeria risulta incompleta perchè le persone descrivono solo parte di ciò che percepiscono
.

La principale motivazione per l'incorreteza delle teorie è la mancanza di motivazioni.

In applicazione dove il costo per errore è elevato bisogna far rispettare sia correttezza sia completezza. Alcune volte la completezza non è raggiungibile, perciò bisogna preferira la correttezza alla completezza. Solitamente il metro per preferire completezza/correttezza è dato da motivazioni pratiche come studi probabilistici.

I.2.2 Semantica linguistica

I.2.2.1 Linguaggio e dominio

- **Dominio:** Un dominio $D = \{M\}$ è un insieme di modelli M.
- **Linguaggio:** Un linguaggio $L = \{T\}$ è un insieme di teorie T.
- **Denotazione e semantiche:** Diciamo che un linguaggio $L = \{T\}$ denota un dominio $D = \{M\}$ se descrive tutti i suoi modelli e scriveremo $L = Den(D)$.
  Possiamo anche dire che un dominio D è la semantica intesa da L scrivendo $D = S(L)$.

I.2.2.2 Osservazioni

Il dominio è definito come uno spazione o qualsiasi cosa noi possiamo immaginare, cosa non possibile con i modelli in quanto devono essere una rappresentazione della realtà.

Una situazione può essere modellata da diversi domini.
Uno stesso dominio può essere descritto da linguaggi differenti che si concentrano su aspetti diversi.
Queste due proprietà vengono dette eterogeneità semantica.

Un dominio è definito come $D = \{M\}$, ma ricordando la definizione di insieme di modelli come $M = \{f\}$ con $f \in M$ e $M \in D$.
Risulta che $M \subseteq D$, possiamo quindi dire che il dominio è l'insieme potenza (si, in italiano powerset è insieme potenza) dell'insieme dei fatti $\{f\}$.

Un linguaggio definito $L = \{T\}$ è un insieme di modelli T, un linguaggio può anche essere modllato come l'insieme $L = \{s\}$ di tutte le frasi $s \in L$ appartenenti alle teorie $T \in L$.
Concludiamo che $T \subseteq L$, ne risulta che un linguaggio è l'insieme potenza degll'insieme $\{s\}$.

I.2.3 Modellazione mentale

Possiamo pensare al modello mentale come formato da 4 componeneti:

1. **Linguaggio:** lo spazio di tutte le possibili teorie
2. **Dominio:** lo spazio di tutti i possibili casi
3. **Modello:** un insieme di fatti
4. **Teoria:** un insieme di frasi che descrivono i fatti nel modello.

I.2.3.1 Gap semantico

Il mondo causa la generazione del modello mentale con i suoi 4 componenti.
Il modelo mentale rappresenta sia il mondo analogico che quello linguistico, ma si discosterà sempre dal mondo per via del gap semantico.
Questo gap è dato dall'imperfezione umana, dai limiti dei nostri sensi e della nostra lingua.

Chapter I.3

Modello mentale formale

I.3.1 Linguaggio informale

- **Termine:** è un elemento del linguaggio, il termine denota un insieme di entità del mondo.
- **Frase:** è un elemento del linguaggio, la frase denota un insieme di fatti.
- **Sintassi:** la sintassi di un linguaggio L, definito come $L = \{s\}$ un insieme di frasi s, è l'insieme di regole formali che ci permettono di definire tutte le frasi $s \in L$ partendo da primitive chiamate alfabeto.
- **Primita, atomica:** Un termine o frase è primitiva se appartiene all'alfabeto, si dice complesso in tutti gli altri.
  Una frase è atomica se è il caso base delle regole di formazione delle frasi.
  Frasi primitive sono atomiche ma non vale il contrario.
- **Sintassi 2:** la sintassi può anche essere definita come segue
  - Un insieme di termini primitivi chiamati termini alfabetici.
  - Un insieme di regola per la costruzione di termini.
  - Un insieme di regole per la formazione termine a frase.
  - Un insieme di frasi primitive chiamate frasi alfabetiche.
  - Un insieme di regola per la formazione di frasi.
  - Un insieme di regole per la formazione da frase a termine.

I.3.1.1 Algoritmo

Segue il processo Il linguaggio naturale ha una struttura molto più complessa per definire il linguaggio:

1. Dai termini primitivi definire di cosa si sta parlando.
2. Con i termini complessi si va a definire come definire le entità selezionate.
3. Con le frasi atomiche si definiscono le proprietà delle entità da testare.
4. Dalle frasi primitive si vano a definire le verità basiche.
5. Con le frasi complesse si definisce come comporre le descrizioni complesse.
6. Con i termini atomici costruire altri termini atomici dipendeti.

I.3.1.2 Osservazioni

- La sintassi permette di definire frasi dichiarative per composizione.
- Il linguaggio naturale ha una struttura molto più complessa ma può essere ridotta ad una sintassi formale.

I.3.1.3 Ambiguità di rappresentazione

- Ci sono infiniti modelli che rappresentano la stessa situazione nel mondo reale.
- Ci sono infinite teorie per la stessa rappresentazione analogica.
- Tutti i modelli e le teorie sono apprissimati.
- Teorie e modelli diversi seppur rappresentando la stessa cosa possono essere mutualmente esclusivi.
- E' importante non assumere che la nostra rappresentazione mentale sia la stessa degli altri.

I.3.2 Linguaggi semi-formali

I.3.2.1 Sintassi formale

- Una sintassi si dice **formale** se:
  - L'alfabeto è riconoscibile.
  - L'insieme di costruttori è finito.
  - Esiste un algoritmo per verificare la correttezza di un frase.
- **Linguaggio informale:** una sintassi che non è formale è detta informale e il linguaggio che genera è detto **linguaggio informale**.
- **Formula ben formulata (wff):** una frase generata da una sintassi formale è detta formula oppure formula ben formata.
  Una wff w_1 è definita sottoformula di una wff w_2 se w_1 è stata usata per costruire w_2.

I.3.2.2 Esempio

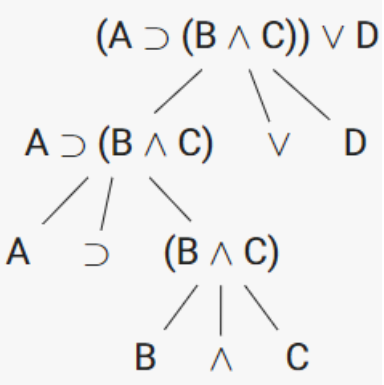
Definiamo un alfabeto riconosciuto:

- Frasi primitive = A, B, C, Q, R, ...
- Connettivi = $\wedge$, $\vee$, $\supset$
- Punteggiatura = (,)

Definiamo un insieme finito di costruttori:

- Ogni frase primitiva è una wff.
- Se A e B sono delle wff in L vale che:
  - A $\wedge$ B è una wff in L.
  - A $\vee$ B è una wff in L.
  - A $\supset$ B è una wff in L.
- Se A è una wff in L allora lo è anche (A).

Ora posso prendere in esempio la stringa (A $\supset$ (B $\wedge$ C)) $\vee$ D e costruire l'albero per rappresentarla.



I.3.2.3 Linguaggi formali

- Un linguaggio si dice **formale** se rispetta le seguenti proprietà:
  - Le formule ed i termini di L sono definiti da formule sintattiche.
  - Il dominio D denotato da L è formalmente definito.
    Chiamiamo D il dominio di interpretazione di L.
  - La denotazione Den di L, indicata con $L = Den(D)$ è una funzione:

    $$I : L \to D$$

    I è chiamata funzione di interpretazione di L.
- Un linguaggio non formale definito da sintassi formale è chiamato **linguaggio semi-formale**.

I.3.3 Linguaggi formali

Riprendendo la definizione precedente.

- Un linguaggio si dice **formale** se rispetta le seguenti proprietà:
  - Le formule ed i termini di L sono definiti da formule sintattiche.
  - Il dominio D denotato da L è formalmente definito.
    Chiamiamo D il dominio di interpretazione di L.
  - La denotazione Den di L, indicata con $L = Den(D)$ è una funzione:

    $$I : L \to D$$

    I è chiamata funzione di interpretazione di L.

I.3.3.1 Linguaggi formali con frasi e termini atomici

- L'interpretazione di un termine è un elemento del dominio.
- E' possibile avere sinonimi ma non polisemia.
- La funzione simbolica è usata per generare termini complessi.
  Viene usata una funzione n-aria che denota lo spazio di tutti i possibili termini che possono essere costruiti.
- Le entità generate da queste funzioni sono tuple (n+1)-arie.

I.3.3.2 Esempi

1

Sia L un linguaggio formale, potrebbe essere eroneamente definito come segue:

- **Frasi dell'alfabeto** = {A,B}
  - A = "Fausto ha meno di 25 anni"
  - B = "Fausto è un professore di IA"
- **Regole di fomrazione delle frasi** = A, B sono le sole formule.
- **D** = { f_1 = "il fatto cheFausto ha 60 anni", f_2 = "il fatto cheFausto è un professore di IA" }
- **Interpretazione** $I : L \to D$
  - $I(A)$ =??? I non è una funzione interpretativa di L, devo aggiungere fatti o abbandonare A.
  - $I(B) = f_2$

2

Il linguaggio formale L che prima era sbagliato potrebbe essere ridefinito come:

- **Frasi dell'alfabeto** = {A,B}
  - A = "Fausto ha meno di 25 anni"
  - B = "Fausto è un professore di IA"
- **Regole di fomrazione delle frasi** = A, B sono le sole formule.
- **D** = { f_1 = "il fatto cheFausto ha 60 anni", f_2 = "il fatto cheFausto è un professore di IA", f_3 = "il fatto che fausto abbia meno di 25 anni" }
- **Interpretazione** $I : L \to D$
  - $I(A) = f_3$
  - $I(B) = f_2$

I.3.3.3 Proposizioni

- **Proposizione(secondo Aristotele):** una proposizione è una frase che afferma o nega un predicato.
- **Proposizione:** è una formula che può essere o vera o falsa.

I.3.3.4 Osservazioni

Consideriamo le tre seguenti frasi.

1. A = "Fausto ha meno di 25 anni"
2. B = "Fausto è un professore di IA"
3. C = "Fausto ha 60 anni"

Possiamo interpretare queste frasi come dei fatti detti $I(A) = f_1$, $I(B) = f_2$, $I(C) = f_3$.
Possiamo anche interpretare il loro valore di verità $I\prime(A) = F$, $I\prime(B) = T$, $I\prime(C) = T$.
Abbiamo cambiato il dominio di interpretazione da D a $D\prime$, quindi dobbiamo anche cambiiare la funzione:

$$I : L \to D \Rightarrow I\prime : L \to D^L$$

Dove $D = \{f\}$ e D^L è un insieme di due distinti valori di proposizioni che stanno per vero, falso.
Riferendoci all'esempio precedente i due domini di interpretazione D e D^L intendono due insiemi molto diveri.

1. D è l'insieme dei fatti che descrivono il mondo.

$$D = \{I(A) = f_1, I(B) = f_2, I(C) = f_3\}$$

2. D^L è l'insieme di giudizi che dicono quali sono i casi nel nostro mondo.

$$D^L = \{I\prime(A) = F, I\prime(A) = F, I\prime(B) = F, I\prime(B) = F, I\prime(C) = F, I\prime(C) = F\}$$

Il passo tra D e D^L risolve il problema dei domini, possiamo affermare che un fatto certo non può essere casuale.

I.3.3.5 Linguaggio formale con proposizioni

I.3.3.6 Esempio

- **Alfabeto** = {A, B}
- **Dominio** D = {0, 1}
- **Interpretazione** $I : L \to D$
- $\mathbf{T}_1$ = {B}, $\mathbf{T}_2$ = {A, B}
- $\mathbf{M}_1$ = {I(B)}, $\mathbf{M}_2$ = {I(A), I(B)}

I.3.3.7 Osservazioni

Consideriamo l'esempio precedente:

- $L = \{A, B\}$
- $T_1 = \{B\}$ $T_2 = \{A, B\}$
- $D = \{1, 0\}$
- I1(L) = {I1(B)=1, I1(A)=0} = {I1(B)}: I1 è un modello M_1 di T_1 ma non di T_2.
- I2(L) = {I2(B)=1, I2(A)=1} = {I2(B), I2(A)}: I2 è un modello M_2 di T_1 e di T_2.
- I3(L) = {I3(B)=0, I3(A)=1} = {I3(A)}: I3 non è un modello M di T_1 e T_2.
- I4(L) = {I4(B)=0, I4(A)=0} = $\emptyset$: I4 non è un modello M di T_1 né di T_2.

Una teoria viene considerata incompleta/parziale se descrive il vero valore **di un sotto insieme** di formule atomiche del linguaggio.
Un modello è un interpretazione che soddisfa tutte le formule di una teoria.
Una teoria può avere più modelli.

I.3.3.8 La negazione

Sia L un linguaggio formale, $I : L \to D$ la sua funzione di interpretazione con $D = \{0, 1\}$ e sia $\neg$ un simbolo primitivo dell'alfabeto allora:

- Se I(A)=1 allora I($\neg$A)=0
- Se I(A)=0 allora I($\neg$A)=1

Dove $\neg$A si legge come "non A".

I.3.3.9 Osservazione

Una teoria T che contiene sia la formula A che $\neg$A non ha modello e viene detta **contradditoria**.

I.3.3.10 Relazione di implicazione

Sia L un linguaggio formale, $I : L \to D$ la sua funzione di interpretazione con $D = \{0, 1\}$, sia $T \subseteq L$ una teoria formale, sia $M \subseteq D$ un modello per T allora $\models_L$ è la funzione di implicazione che associa cosa è vero in M con le wff in T.

$$\models_L \subseteq M \times T$$

Possiamo anche scrivere:

$$M \models_L T$$

Dicendo che M implica T.

I.3.3.11 Negazione con implicazione

Sia L un linguaggio formale, la funzione di interpretazione $I : L \to D$ ...
Allora per ogni formula atomica $A \in L$:

- I $\models$ A se I(A)=True
- I $\models \neg$A se non è vero che I $\models$ A (I $\not\models$ A)

I.3.3.12 Osservazioni

- Una teoria T che contiene sia la formula A che $\neg$A non ha modello e viene detta **contradditoria**.
- Per ogni formula A la formula A$\wedge\neg$A viene detta **contraddizione**.
- Per ogni formula A la formula A$\vee\neg$A viene detta **tautologia**.

I.3.4 Modello mentale formale

I.3.4.1 Nozioni

Un modello formale è un modello formale in cui:

- La relaziona tra formule atomiche ed il dominio di interpretazione sono formalizzate da una funzione di interpretazione.
- Le relazione tra modello e teoria è formalizzata dalla relazione di implicazione.

La costruzione di un modello mentale formalizzato segue i seguenti step:

1. Il linguaggio è dato.
2. Dominio e funzione di interpretazione sono dati.
3. La funzione di implicazione è data.
4. Un modello è costruito assumendo che un certo insieme di fatti sia vero.
5. Ci sono **solo** 2 usi per un modello formale:
   (a) **Modellazione (apprendimento):** il modello formale viene creato/esteso.
   (b) **Ragionamento:** il modello formale viene usato per risolvere delle query.

Un sistema logic-based che ragiona su conoscenze e dati ha due componenti principali:

1. Una conoscenza di base (**KB**) tale che KB$\subseteq$L che è una teoria del mondo.
2. Un sistema di ragionamento che risponde alle query grazie al contenuto del KB.

Chapter I.4

Teoria degli insiemi

I.4.1 Insieme

Insieme: una collezione di elementi la cui descrivono deve essere non ambigua ed univoca

I.4.1.1 Concetti di base

- **Insieme vuoto:** è l'insieme che non contiene elementi e si indica $\emptyset$.
- **Appartenenza:** a$\in$A indica che l'elemento a appartiene all'insieme A.
- **Non appartenenza:** è il contrario dell'appartenenza e si indica a$\notin$A.
- **Uguaglianza:** A = B se e solo A e B contengono gli stessi elementi.
- **Non uguaglianza:** se A e B non sono ugualu si indica con A $\neq$ B.
- **Sottoinsieme:** A$\subseteq$B indica che tutti gli elementi di A appartenegono anche a B.
- **Sottoinsieme proprio:** se e solo se A$\subseteq$B e A$\neq$B allora si dice che A$\subset$B.

I.4.1.2 Power set

Definizione: il power set di un insieme è l'insieme contenente tutti i sottoinsiemi di A.
Se l'insieme ha n elementi il suo power set ha 2^n elementi.

I.4.1.3 Operazioni sugli insiemi

Unione

Dati due insiemi A e B la loro unione A$\cup$B è defina come l'insieme di tutti gli elementi appartenenti sia ad A si che B.

Intersezione

Dati due insiemi A e B definiamo la loro intersezione A$\cap$B come l'insieme che contiene gli elementi che appartengono contemporanemente ad A e a B.

Differenza

Dati due insiemi A e B la loro differenza A$-$B è l'insieme A a cui sono stati tolti tutti gli elementi che aveva in comune con B.

Complemento

Dati due insiemi A e B tali che A$\subseteq$B defininiremo il complemenetare di A in B come l'insieme di tutti gli elementi che appartengono a B ma non ad A, lo indicheremo con $\bar{\text{A}}$ oppure $\text{C}_\text{B}\text{A}$.

I.4.1.4 Proprietà delle operazioni

Con lo stesso insieme:

- A∩A=A
- A∪A=A

Commutative:

- A∩B=B∩A
- A∪B=B∪A

Con l'insieme vuoto:

- A∩∅=∅
- A∪∅=A

Associative:

- (A∩B)∩C=A∩(B∩C)
- (A∪B)∪C=A∪(B∪C)

Dsitributive:

- A∩(B∪C)=(A∩B)∪(A∩C)
- A∪(B∩C)=(A∪B)∩(A∪C)

Leggi di De Morgan:

- $\overline{A \cup B}=\bar{A} \cap \bar{B}$
- $\overline{A \cap B}=\bar{A} \cup \bar{B}$

I.4.1.5 Prodotto cartesiano

Dati deu insiemi A e B, definiamo il prodotto cartesiano di A e B come l'insieme delle tuple ordinate (a,b) tali che a∈A e b∈B.
Formalmente si può esprimere come:

$$A \times B = \{(a,b) : a \in A \text{ and } b \in B\}$$

Oservazioni

- A×B≠B×A
- Il prodotto cartesiano può essere applicato ad n insiemi di insiemi A_1, A_2, ..., A_n.
  $A_1 \times A_2 \times \cdots \times A_n$ sarà l'insieme ordinato di n-tuple (x_1, ..., x_n) dove $x_i \in A_i$ per ogni $i = 1 \dots n$

I.4.2 Relazioni

Definizione: una relazione R dall'insieme A ad un insieme B è un sottoinsieme del prodotto cartesiano di A e B: R⊆A×B.
Se (x,y)∈R scriveremo xRy e diremmo "x è R-relazionato a y".
Una relazione binaria su un insieme A è un sotto insieme R⊆A×A.

Data una relazio e R da A a B:

- Il dominio di R è l'insieme Dom(R)={a∈A | esiste un b∈B che aRb}
- Il codominio di R è l'insieme Cod(R)={b∈B | esiste un a∈A che aRb}

I.4.2.1 Relazione inversa

Sia R una relazione da A a B, la relazione inversa di R è la relazione $R^{-1} \subseteq B \times A$ definita come:

$$R^{-1}=\{(b,a) \mid (a,b)\in R\}$$

I.4.2.2 Proprietà delle relazioni

Sia R una relazione binaria A, allora R può avere le seguenti proprietà:

- **Riflessiva:** se e solo se aRa per ogni a∈A.
- **Simmetrica:** se e solo se aRb implica che bRa per ogni a,b∈A.
- **Transitiva:** se e solo se aRb e bRc implica che aRc per ogni a,b,c∈A.
- **Anti-simmetrica:** se e solo se aRb e bRa implica che a=b per ogni a,b∈A.

I.4.2.3 Relazione di equivalenza

Sia R una relazione binaria su un insieme A.
R è una relazione di equivalenza se e solo se soddisfa le seguenti proprietà:

- Riflessiva
- Simmetrica
- Transitiva

Le relazioni di equivalenza sono spesso indicate con $\sim$ oppure con $\equiv$.

I.4.2.4 Partizione di insiemi

Sia A un insieme , una partizione di A è una famiglia F di sottoinsiemi non vuoti tali che:

- L'unione di tutti i sottoinsiemi sia A.
- I sottoinsiemi sono disgiunti a coppie.

Ogni elemento di A apprtiene esattamente ad un insieme di F.

I.4.2.5 Classe di equivalenza

Sia A un insieme e $\equiv$ una relazione di equivalenza su A, dato x∈A definiamo classe di equivalenza X l'insieme di elementi $x\prime \in A$ tale che $x\prime \equiv x$, formalmente:

$$X=\{x\prime | x\prime \equiv x\}$$

E' possibile prendere ogni elemento di x per ottenere una classe di equivalenza X, la classe di equivalenza è denotata anche da [x].

I.4.2.6 Insieme quoziente

L'insieme quozienete di A è l'insieme delle classi di equivalenza definite da $\equiv$ su A, denotato da $A/\equiv$

Teorema

Data una relazione di equivalenza $\equiv$ su A, la classe di equivalenza definita da $\equiv$ su Aè una partizione di A.

Similmente, data una partizione di A, la relazione R definita come $xRx\prime$ se e solo se x e $x\prime$ appartengono allo stesso sottoinsieme, e una relazione di equivalenza.

I.4.2.7 Relazione di ordinamento

Sia A un insieme e R una relazione binaria su A.
R è un ordinamento parziale, denotato come $\leq$, se:

- **Riflessiva:** $a \leq a$
- **Anti-simmetrica:** $a \leq b$ e $b \leq a$ allora $a=b$.
- **Transitiva:** $a \leq b$ e $b \leq c$ allora $a \leq c$.

Se la relazione vale per tutti gli $a,b \in A$ allora si dice ordinamento totale.
Una relazione è in ordine stretto, denotata con $<$, se:

- **Transitiva:** $a<b$ e $b<c$ allora $a<c$.
- Per ogni $a,b \in A$ o $a<b$ o $b<a$ oppure $a=b$.

I.4.3 Funzioni

Dati due insiemi A e B, una funzione f da A a B è una relazione che associa ad ogni elemento di a in A esattamente un elemento un elemento b in B, denotata con:

$$f: A \to B$$

Il dominio di f è tutto l'insieme A.
L'immagine di ogni elemento a in A è l'elemento di b in B tale che $b=f(a)$.
Il codominio di f è un sottoinsiemi di B, definito come segue:

$$Im_f=\{b \in B \mid \exists\ a \in A \text{ tale che } b=f(a)\}$$

I.4.3.1 Classi di funzioni

1. **Suriettive:** una funzione $f: A \to B$ è suriettiva se ogni elemento in b è l'immagine di qualche elemento in A.
2. **Iniettiva:** una funzione $f: A \to B$ è iniettiva se ogni elemento di A ha un immagine diversa in B.
3. **Biettiva:** una funzione $f: A \to B$ è biettiva se è sia iniettiva che suriettiva.
4. **Inversa:** se $f: A \to B$ è biettiva possiamo defnire una funzione inversa:

$$f^{-1}: B \to A$$

I.4.3.2 Funzioni composte

Siano $f: A \to B$ e $g: B \to C$ funzioni, allora la composizione di f e g creerà la funzione $g \circ f: A \to C$.

- $(g \circ f)(a) = g(f(a))$

Chapter I.5

Ragionamento formale

I.5.1 Logiche

- **Logica:** una logca L è una tripla $\mathcal{L}$=$\langle L, I, \models \rangle$ dove L è un linguaggio formale, I una funzione di interpretazione I: L $\rightarrow$ D e $\models$ una relazione di implicazione.
- **Calcolo logico:** un calcolo logico $\mathcal{C}_{\mathcal{L}}$ è una tupla $\mathcal{C}_{\mathcal{L}} = \langle \mathcal{L}, \mathcal{P} \rangle$ dove $\mathcal{L}$ è una logica e $\mathcal{P}$ un insieme di domande.
- **Ragionamento logico:** dato un calcolo logico $\mathcal{C}_{\mathcal{L}}$, con ragionamento logico indichiamo il processo con il quale si risolve un problema applicando un algoritmo non per forza terminale.

I.5.1.1 Logica proposizionale

Le caratteristiche di questa logica sono:

- Un linguaggio proposizionale contiene solo proposizioni primitive.
- Le formule sono interpretate attraverso un dominio di giudizi.
- Le formule complesse sono formate usando un numero arbitrario di connettivi proposizionali.
- I connettivi proposizionali possono essere.
  - $\neg$ letto come "not" per la negazione.
  - $\wedge$ letto come "and" per le congiunzioni.
  - $\vee$ letto come "or" per le disgiunzioni.
  - $\Longrightarrow$ letto come "implies" per le implicazioni.
  - $\Longleftrightarrow$ letto come "if and only if" per le equivalenze.
  - $\uparrow$ letto come "nand" per le congiunzioni negate.
  - $\downarrow$ letto come "nor" per le disngiunzioni negate.

La logica proposizionale è utile per problemi che possono essere formalizzati per essere indipendenti da strutture interne.

I.5.1.2 Logica del primo ordine

Le caratteristiche di questa logica sono:

- Termini e formule sono complesse.
- Molto spesso le formule primitive non sono parte del linguaggio.
- Termini e formule atomiche sono interpretate su un dominio di entità e fatti.
  Le formule complesse sono interpretate attraverso un dominio di giudizi.
- Le formule complesse sono formate usando connettivi proposizionali e un numero arbitrario di quanificatori.
- I quantificatori sono:

  - $\forall$ letto come "for all" per quantificare tutti i termini di un insieme.
  - $\exists$ letto come "there exists" per dire se esiste almeno un elemento in un insieme.

La logica del primo ordine è utile ogni volta la struttura interna dei termini e la loro composizione per formare la verità o la falsità di una formula atomica è importante, quindi quando è importante essere molto descrittivi.

I.5.1.3 Logica descrittiva

Le caratteristiche di questa logica sono:

- Le logiche proposizionali + una logica del primo ordine permettono di usare solo formule atomiche e non primitive.
- Solo predicati unari e binari sono permessi.
  - Predicati unari detti classi.
  - Predicati binari detti ruoli.
- Termini e formule sono interpretati su un dominio di entità e fatti.
- Le forule complesse sono formate usando connettivi proposizionali e due operatori modali.
- Gli operatori modali sono:
  - $\exists R$ letto come "there exists an element of ..." per quantificare l'esistenza su un codominio di ruoli.
  - $\forall R$ letto come "for all elements of" per la quantificazione universale su un codominio di ruoli.

I modelli di logica descrittiva permettono di rappresentare e ragionare sui diagrammi ER, diagrammi UML e knowledge graph.

I.5.2 Problemi di ragionamento

Il ragionamento è usato per ultimare alcuni compiti come:

- Verifica del modello.
- Soddisfacibilità.
- Validità.
- Insoddisfacibilità.
- Conseguenza logica.
- Equivalenza logica.

I.5.2.1 Tabelle di verità

Sono il modo per dimostrare la verità dei fatta oppure di una proposizione che descrive quei fatti.
Permettono di descriveve ogni possibile modello M di un certo dominio D, questo significa costruire ogni possibile combinazione di fatti.
Infatti esistono 2^n possibili modelli con n il numero di fatti nel dominio.
Due frasi che rappresentano lo stesso fatto avranno lo stesso valore e possono essere rappresentate da una sola proposizione.

Esempi

A	B	A∧B
1	1	1
1	0	0
0	1	0
0	0	0

A	¬A
0	1
1	0

A	B	A↑B
1	1	0
1	0	1
0	1	1
0	0	1

I.5.2.2 Verifica del modello

Data un teoria T e un modello M, la verifica consiste nel verificare che qualsiasi sia M possa essere un modello valido per T, equivale a verificare che M⊨T.

I.5.2.3 Soddisfacibilità

Una teoria T viene detta soddisfacibile se esiste un modello rappresentato da T, invece se viene dato T bisogna verificare se esiste o meno un modello M che coincide con un modello descritto da T.

Esempio

Dobbiamo compilare la tabella di verità di A∧B per vedere se la formula è soddisfacibile o meno.

A	B	A∧B
1	1	1
1	0	0
0	1	0
0	0	0

La formula è soddisfatta visto che c'è almeno un modello che implica la formula, ovvero che è vero.

I.5.2.4 Validità

Una teoria T è detta valida se ogni modello possibile è rappresentato da T.
Una teoria T è detta valida se ogni modello M del nostro dominio soddisfa T, e.g.:

$$\forall M,\ M \models T$$

Esempio

Computiamo ora la tabella di verità di ¬(A∧¬A) per verificare che la formula sia valida o meno.

A	¬A	¬(A∧¬A)
1	0	1
0	1	1

I.5.2.5 Insoddisfacibilità

Una teoria T è detta insoddisfabile se non c'è un modello rappresentato da T.
Una teoria T è detta insoddisfabile se:
Non esiste un modello M tale che

$$M \models T$$

Ogni modello M è tale che:

$$M \not\models T$$

Esempio

Computiamo ora la tabella di verità di A∧¬A per verificare che la formula sia soddisfabile o meno.

A	¬A	A∧¬A
1	0	0
0	1	0

La formula risulta insoddisfabile visto che non ci sono modelli che la implicano.

I.5.2.6 Conseguenza logica

Una teoria T_2 viene detta conseguenza logica di un'altra teoria T_1 se ogni modello rappresentato da T_1 viene rappresentato anche da T_2.

$$\forall M:\ M \models T_1,\ M \models T_2$$

Esempio

Computiamo ora la tabella di verità di A e ¬(¬A∧B∧¬A) per vedere se quest'ultima è una conseguenza di A:

A	B	¬(¬A∧B∧¬A)
1	1	1
1	0	1
0	1	0
0	0	1

Come si può vedere la formula è una conseguenza logica di A visto che ogni volta che A è 1 anche la formula lo è.

I.5.2.7 Equivalenza logica

Una teoria T_1 viene detta equivalenza logica di un'altra teoria T_2 rappresentano lo stesso modello.

$$\forall M: M \models T_1 \text{ se e solo se } M \models T_2$$

Esempio

Computiamo ora la tabella di verità di A e ¬(¬A∧¬(B∧¬B)) per vedere se le due formule sono logicamente equivalenti.

A	B	¬(¬A∧¬(B∧¬B))
1	1	1
1	0	1
0	1	0
0	0	0

Le formule ¬(¬A∧¬(B∧¬B)) e A sono equivalenze logiche visto che sono vere e false negli stessi momenti.

I.5.3 Scegliere una logica

Per affrontare un problema dobbiamo seguire i seguenti passi:

1. Formalizzare le domande e le risposte di un problema.
2. Sviluppare l'approccio che ci sembra più logico.
3. Scegliere il tipo di logica più adatto.
4. Scrivere la teoria T che modella il problema.
5. Usare la logica per risolvere il problema.

Sulla scelta della logica bisogna prestare molta attenzione perchè ogni logica è caratterizzata da:

- **Diversa espressività** Quali problemi di decione posso esprimere.
- **Efficenza computazionale** Quanto è oneroso risolvere un problema decisionale.

I.5.3.1 Decidibilità

Una logica è decidibile se esiste un metodo efficace per verificare che una formula appartenga sia inclusa in una teoria.

- Il metodo effettivo è un algoritmo che dato un problema decisionale ritorna vero o falso.
- Tutte le logiche del corso sono decidibili, tranne la logica del primo ordine.

I.5.3.2 Complessità

Data una logica decidibile la complessità quantifica la difficoltà a computare il ragionamento in una data logica. I linguaggi logici sono classificati secondo vari gradi di complessità:

- P
- NP
- PSpace

  ⋮

I.5.3.3 Espressività

Linguaggio	Frasi NL	Formule
Logica proposizionale	Fausto likes skiing I like skiing	Fausto-likes-skiing I-like-skiing
Logica del primo ordine	Every person likes skiing I like skiing Fausto likes skiing	∀person.like-skiing(person) like-skiing(I) like-skiing(Fausto)
Logica descrittiva	Every perons likes cars	person $\sqsupseteq \exists$ likes.Car

Chapter I.6

Usare modelli formali

I.6.1 Livello di formalizzazione

I.6.1.1 Linguaggio specifico

Esistono diversi tipi di linguaggi specifici che dipendono dal linguaggio che usano.

- **Modelli informali** usano linguaggio naturale.
- **Modelli semi-formali** usano lingiiagi strutturati con sintassi semi-formali e delle semantiche informali.
- **Modelli logici** usano linguaggi formali.

I.6.1.2 Perchè i linguaggi informali?

Usato per	Vantaggi	Svantaggi
Specifiche informali	Economico da usare Utile per interagire con utenti	La semantica è informale Impossibile da automatizzare

I.6.1.3 Perchè i diagrammi?

Usato per	Vantaggi	Svantaggi
Specifiche semi-formali	Economico da usare Utile per interagire con utenti	La semantica è informale Impossibile da automatizzare

I.6.1.4 Perchè le logiche?

Usato per	Vantaggi	Svantaggi
Specifiche formali Automazioni	Molto capibile per le sintassi e semantiche formali Molto efficente da automatizzare	Difficile da usare con gli utenti

I.6.2 Usare le logiche

Esempi di problemi:

- Usare una teoria per concordare.
- Usare una teoria per garantire l'interoperabilità.
- Usare il ragionamento $\models$ per garantire che il programma faccia ciò per cui è pensato.
- Usare il ragionamento per implementare IA.

Part II

Logica Proposizionale

Chapter II.1

La logica

II.1.1 Intuizioni iniziali

Le caratteristiche della logica proposizionale sono:

- Un linguaggio proposizionale contiene solo proposizioni primitive.
- Le formule sono interpretate attraverso un dominio di giudizi.
- Le formule complesse sono formate usando un numero arbitrario di connettivi proposizionali.
- I connettivi proposizionali possono essere.
  - $\neg$ letto come "not" per la negazione.
  - $\wedge$ letto come "and" per le congiunzioni.
  - $\vee$ letto come "or" per le disgiunzioni.
  - $\Longrightarrow$ letto come "implies" per le implicazioni.
  - $\Longleftrightarrow$ letto come "if and only if" per le equivalenze.
  - $\uparrow$ letto come "nand" per le congiunzioni negate.
  - $\downarrow$ letto come "nor" per le disngiunzioni negate.

La logica proposizionale è utile per problemi che possono essere formalizzati per essere indipendenti da strutture interne. Sulla logica proposizionale possiamo fare le seguenti osservazioni:

- Una proposizione è una frase che descrive il mondo.
- Una proposizione può essere o vera o falsa.
- "not P" è vera se P è falsa e viceversa.
- "P and Q" è vera se e solo se P e Q sono entrambe vere.
- "P or Q" per essere vera è sufficiente che solo una delle due sia vera.
- "P implies Q" indica che Q è vera quando lo è P, ma non dice nulla su quando P è falsa.
- "P if and only if Q" indica che P e Q devo essere vere/false allo stesso momento.
- "P xor Q" è vera solo se una delle due è vera.

II.1.2 Sintassi

II.1.2.1 Alfabeto proposizionale

L'alfabeto è composto da:

- Simboli logici: $\neg, \wedge, \vee, \supset, \equiv$.
- Simboli non logici: costanti proposizionali e insiemi **PROP** che contengono simboli P chiamati variabili proposizionali che possono contenere una costante proposizionale come un valore.
- Simboli separatori: "(" e ")".

II.1.2.2 Formule ben formate(wff)

Una formula wff viene definita come segue:

- Ogni P∈**PROP** è una formula atomica.
- Ogni formula atomica è wff.
- Se A e B sono formule, allora ¬A, A∧B, A∨B, A⊃B e A≡B sono formule.

Per leggere una formula è importante sapere che i vari operatori hanno delle diverse priorità, come in matematica le parentesi fungono da modificatore delle priorità.

Simbolo	**Priorità**
¬	1
∧	2
∨	3
⊃	4
≡	5

II.1.2.3 Sottoformule

Una sottoformula di una formula, rappresentata come un albero, indica l'insieme di tutti i suoi sottoalberi.
Viene formalmente definita come:

- A è una sottoformula di se stessa.
- A e B sono sottoformule di A∧B, A∨B, A⊃B e A≡B.
- A è una sottoformula di ¬A.
- Se A è una sottoformula di B e B è una sottoformula di C, allora A è una sottoformula di C.
- A è una sottoformula propria di B se A è sottoformula di B e A≠B.

Esempi

1. Se piove mentre splende il sole apparirà l'arcobaleno.
   p=piove; q=splende il sole; r=arcobaleno;
   (p∧q)⊃r
2. Claudio viene se Elsa viene.
   p=Claudio viene; q=Elsa viene;
   q⊃p
3. Claudio viene se Elsa viene e viceversa.
   p=Claudio viene; q=Elsa viene;
   q≡p
4. Giacomo viene se e solo se Pietro resta a casa.
   p=Giacomo viene; q=Pietro sta a casa;
   p≡q
5. Noi andiamo se non piove.
   p=Noi andiamo; q=piove;
   p≡ ¬q
6. Claudio ed Elsa sono o fratello e sorella o nipoti.
   p=Claudio ed Elsa sono fratello e sorella; q= Claudio ed Elsa sono nipoti;
   p∨q
7. Se perdo se non posso fare una mossa, allora ho perso.
   p=ho perso; q=non posso fare una mossa;
   (q⊃p)⊃p

II.1.3 Funzione di interpretazione

Il dominio di interpretazione della logica proposizionale è D={True, False}.
L'interpretazione proposizionale è una funzione:

$$\text{I: } \mathbf{PROP} \rightarrow \text{D}$$

Se $|\mathbf{PROP}|$ è la cardinalità di **PROP** allora esistono $2^{|\mathbf{PROP}|}$ interpretazioni differenti corrispondenti a tutti i sottoinsiemi di **PROP**.

II.1.4 Implicazione

Diciamo che una funzione di interpretazione implica una formula A se:

- I$\models$A, se I(A)=True con A$\in$**PROP**.
- I$\models \neg$A, se non è vero che I$\models$A.
- I$\models$A$\wedge$B se I$\models$A e I$\models$B.
- I$\models$A$\vee$B se I$\models$A o I$\models$B.
- I$\models$A$\supset$B se I$\models$A allora I$\models$B.
- I$\models$A$\equiv$B se I$\models$A se e solo se I$\models$B.

$\neg$True	False
$\neg$False	True
True $\wedge$ True	True
True $\wedge$ False	False
False $\wedge$ True	False
False $\wedge$ False	False
True $\vee$ True	True
True $\vee$ False	True
False $\vee$ True	True
False $\vee$ False	False

True $\supset$ True	True
True $\supset$ False	False
False $\supset$ True	True
False $\supset$ False	True
True $\equiv$ True	True
True $\equiv$ False	False
False $\equiv$ True	False
False $\equiv$ False	True

II.1.5 Errori comuni

- Noi esprimiamo le congiunzioni con molte parole oltre a "e", degli esempio posso essere "ma", "quindi ", "per tanto", ...
- Alcune vole "e" non unisce due proposizioni intere ma solo due sostantivi.
- Alcune volte "and" unisce due aggettivi.
- Il modo per esprimere una disgiunzione esclusiva è (p$\vee$q)$\wedge\neg$(p$\vee$q), mentre il modo per indicare che hanno valori di verità diversi è negare la loro uguaglianza $\neg$(p$\equiv$q).
- **Anche se:** la frase "p anche se q" può essere tradotta in p$\wedge$(q$\vee\neg$q).

Chapter II.2

Calcolo

II.2.1 Tabelle di verità

Sono il mezzo attraverso il quale generiamo tutte le possibili interpretazioni generate considerando tutte le proposioni atomiche per rispondere ai quesiti di:

- Verifica del modello.
- Soddisfabilità.
- Validità.
- Insoddisfabilità.
- Conseguenza logica.
- Equivalenza logica.

Per costruire una tabella di verità, con n proposioni aomiche, è possibile seguire l'algoritmo:

1. Esiste una riga per ogni interpretazione, quindi ne avrò 2^n.
2. Le prime n colonne comprendono tutte le interpretazioni, mentre l'ultima i valori di verità della formula totale.
   Le colonne nel mezzo contengono i valori di verità delle sottoformule della formula finale presa in considerazione.
3. L'asseganemnto dei valori alle proposizioni atomiche inizia da sinistra verso destra, nella prima colonnna alterno T e F con periodo 1, nella seconda con periodo 2 e nella n con periodo n.

II.2.2 Verifica del modello

Sia / un interpretazione applicata ad un linguaggio proposizionale L.
Possiamo verificare che una formula A∈L è soddisfabile da / applicando in modo ricorsivo l'algoritmo **MCHECK(I,A)**.

II.2.2.1 Algoritmo

Caso base

A=p

```
MCHECK(I⊨p)
if I(p)==True then
    return YES
else
    return NO
```

Caso ricorsivo

A=B∧C

```
MCHECK(I⊨B∧C)
if MCHECK(I⊨B) then
    return MCHECK(I⊨C)
else
    return NO
```

A=B∨C

```
MCHECK(I⊨B∨C)
if MCHECK(I⊨B) then
    return YES
else
    return MCHECK(I⊨C)
```

A=B⊃C

```
MCHECK(I⊨B⊃C)
if MCHECK(I⊨B) then
    return MCHECK(I⊨C)
else
    return YES
```

A=B≡C

```
MCHECK(I⊨B≡C)
if MCHECK(I⊨B) then
    return MCHECK(I⊨C)
else
    return ¬MCHECK(I⊨C)
```

II.2.3 Soddisfacibilità

Possiamo controllare che qualsiasi formula A∈L è soddisfabile applicando l'algoritmo **SAT(A)** che è definito come segue:

- **Input:** inseriamo la formula A della quale vogliamo sapere se esiste un interpretazione che la soddisfa.
- **Output:** ci viene restituita l'interpretazione se la formula è soddisfabile in caso contrario viene ritornato "no".
- Possiamo verificare A sia soddisfabile applicando in moso ricorsivo **MCHECK(I,A)** come segue:
  - Estrarre tutte le proposizioni atomiche di A.
  - Generare tutte le interpretazioni I utili.
  - Applicare **MCHECK(I,A)** finchè non si verifica una delle due condizioni.

  1. Se **MCHECK(I,A)** ritorna "YES" allora viene ritornato I.
  2. Se non ci sono più I da analizare viene ritornato "NO".

- **SAT(A)** compie una cosidetta lazy evaluation infatti salta le interpretazioni quando irrilevanti così evita di valutare 2^n casi.

Esempio

Controllo se (P∧Q)∨(R⊃S) è soddisfabile.
Devo calcolare tutte le possibile I:

- {P, Q, R, S}
- {P, Q, R}

- {P, Q}
  ⋮

Per ogni I rimpiazzare i valori di verità corrispondenti nelle primitive.
Con I={P} abbiamo che:

$$(\text{True} \wedge \text{False}) \vee (\text{False} \supset \text{False})$$
$$\text{False} \vee \text{True}$$
$$\text{True}$$

Quindi la formula è soddisfabile.

II.2.4 Validità

Possiamo controllare che qualsiasi formula A∈L sia valida applicando l' algoritmo **VALID(A)** come segue.

- **Input:** inseriamo la formula A della quale vogliamo sapere se è soddisfabile per ogni interpretazione.
- **Output:** viene ritornato "YES" se è una formula valida, "NO" altrimenti.
- Possiamo ora verificare che A sia valida applicando **MCHECK(I,A)** come segue:
  - Estrarre tutte le proposizioni atomiche di A.
  - Generare tutte le interpretazioni I utili.
  - Applicare **MCHECK(I,A)** finchè non si verifica una delle due condizioni.

  1. Se un **MCHECK(I,A)** ritorna "NO" allora tutta la funzione ritorna "NO".
  2. Se non ci sono più formule da analizzare ritorna "YES".
- **VALID(A)** compie una lazy evaluation così appena una parte ritorna "NO" si ferma.

Esempio

Controllo se (P∧Q)∨(R⊃S) è valida.
Devo calcolare tutte le possibile I:

- {P, Q, R, S}
- {P, Q, R}
- {P, Q}
  ⋮

Per ogni I rimpiazzare i valori di verità corrispondenti nelle primitive.
Con I={P, R} abbiamo che:

$$(\text{True} \wedge \text{False}) \vee (\text{True} \supset \text{False})$$
$$\text{False} \vee \text{False}$$
$$\text{False}$$

Essendo un interpretazione falsa la formula non è valida.

II.2.5 Insoddisfabilità

Possiamo dire per una qualsiasi formula A∈L se è insoddisfabile applicando l'algoritmo **UNSAT(A)** come segue:

- **Input:** inseriamo la formula A della quale vogliamo sapere se è insoddisfabile.
- **Output:** ritorna "YES" se nessuna I soddisfa A, ritorna "NO" altrimenti.
- Possiamo ora verificare che A sia insoddisfabile applicando **MCHECK(I,A)** come segue:
  - Estrarre tutte le proposizioni atomiche di A.

- Generare tutte le interpretazioni I utili.
- Applicare **MCHECK(I,A)** finchè non si verifica una delle due condizioni.

1. Se un **MCHECK(I,A)** ritorna "YES" allora si ritorna "NO".
2. Se non ci sono più formule da analizzare ritorna "YES".

Esempio

Controllo se (P∧Q)∨(R⊃S) è invalida.
Calcolo tutti i possibili I.
Prendo I={P} abbiamo che:

(True∧False)∨(False⊃False)
False∨True
True

In questo caso **UNSAT(A)** ritornerà "NO" perchè la formula è soddisfabile.

II.2.6 Correlazioni tra validità soddisfacibilità e insoddisfabilità

Queste 3 proprietà sono legate tra di loro.
Proposizione 1: se una formula è valida allora è soddisfabile e anche non insoddisfabile.

valida⊃soddisfabile⊃ ¬insoddisfabile

Proposizione 2: se una formula è insoddisfabile allora è non soddisfabile e non valida.

insoddisfabile⊃ ¬soddisfabile⊃ ¬valida

Proposzione 3: data una formula A allora vale che:

A	¬A
valida	insoddisfabile
soddisfabile	¬valida
¬valida	soddisfabile
insoddisfabile	valida

Proposizione 4: Per un qualsiasi insieme finito Γ di formule($A_1, \dots A_n$ con $n \geq 1$) possiamo dire che Γ è:

- Valido se e solo se $A_1 \wedge \cdots \wedge A_n$ è valida.
- Soddisfabile se e solo se $A_1 \vee \cdots \vee A_n$ è soddisfabile.
- Insoddisfabile se e solo se $A_1 \vee \cdots \vee A_n$ è insoddisfabile

II.2.7 Conseguenza logica

Possiamo capire se una qualsiasi $T_1 \models T_2$ è valida applicando l'algoritmo **LOGCONS(T_1,T_2)** come segue:

- **Input:** T_1 è la teoria di partenza che assumo esssere vera mentre T_2 è quella che deve seguire o meno T_1.
- **Output:** ritorna "YES" se T_2 segue da T_1, "NO" altrimenti.
- Possiamo ora verificare che sia una conseguenza logica come segue:
  - Genero tutte le interpretazioni utili I.
  - Applico sistematicamente **MCHECK(I,T_1)** con i seguenti risultati:

1. Se **MCHECK(I,T_1)** ritorna "YES" applico **MCHECK(I,T_2)** che se ritoena "NO" faccio ritornare "NO" da tutto.
2. Se non ci sono più I da analizzare ritorno "YES".

- **LOGCONS(T_1,T_2)** compie una lazy evaluation quando una chiamata ritorna "NO".

II.2.8 Equivalenza logica

Possiamo capire se una qualsiasi T_1 e T_2 sono logicamente equivalenti con l'algoritmo **LOGEQ(T_1,T_2)** come segue:

- **Input:** T_1 è la teoria di partenza che assumo esssere vera mentre T_2 è quella che deve essere equivalente o meno T_1.
- **output:** ritorna "YES se sono equivalenza logica, "NO" altrimenti.
- Possiamo verificare che due teorie sono un'equivalenza logica applicando l'algoritmo:
  - Genero le interpretazioni che potrebbero essermi utili.
  - Applico **MCHECK(I,T_1)** e **MCHECK(I,T_2)**.
  - Se i risultati sono diversi ritorno "NO", ritorno "YES" altrimenti.

Chapter II.3

La procedura di decisione DPLL

II.3.1 Nozioni di base

SAT o UNSAT sono delle proprietà chiave che indicano se una teoria può essere applicata in pratica.
PL SAT è un prblema NP-completo quindi tutti i problemi NP-completi posso essere codificati in SAT.
Teorema di deduzione: data Γ, se $\phi \models \psi$ allora $\Gamma \models \phi \supset \psi$ con Γ possibilmente vuota.

Questo ci permette di ridurre il problema dalla verifica della conseguenza logica ad PL SAT.
PL SAT può essere ridotto con l'uso di CNF PL SAT(Conjunctive Normal Form), a questo punto CNF SAT può essere risolto in modo molto efficente usando l'euristica.

II.3.2 CNF(Conjunctive Normal Form)

Definizioni:

- **Letterale:** è o una variabile proposizionale oppure il negato di una variabile proposizionale, due esempi sono le formule p e $\neg p$.
- **Clausole:** è una disgiunzione($\vee$) di letterali.
- **CNF(Conjunctive Normal Form):** una formula è in CNF se è una congiunzione di varie clausole per esempio:

  $$(p \vee \neg q \vee r) \wedge (q \vee r) \wedge (\neg p \vee \neg q) \wedge r$$

  Una CNF ha sempre la seguente formula:

  $$(L_{(1,1)} \vee \cdots \vee L_{(1,n_1)}) \wedge \cdots \wedge (L_{(m,1)} \vee \cdots \vee L_{(m,n_m)})$$

  Scritto anche come:

  $$\bigwedge_{i=1}^{m}(\bigvee_{j=1}^{n_j} L_{i,j})$$

  dove $L_{i,j}$ è il j-esimo letterale della i-esima formula.

Proprietà delle clausole:

- L'ordine dei letterali in una clausola non importa $\phi \vee \psi \equiv \psi \vee \phi$, infatti $(p \vee q \vee r \vee \neg r) \equiv (\neg r \vee q \vee p \vee r)$.
- Alcuni letterali possono essere uniti, se una clausola c'è più di un'occorrenza dil un letterale posso spostare i due uguali vicini per unirli $\phi \vee \phi \equiv \phi$, infatti $(p \vee q \vee r \vee q \vee \neg r) \equiv (p \vee q \vee r \vee \neg r)$.
- Le clausole sono insiemi di letterali, possiamo costruire un insieme togliendo le disgiunzioni e ignorano le ripetizioni, infatti $(p \vee q \vee r \vee \neg r)$ è rappresentato da $\{p, q, r, \neg r\}$.

Proprietà delle formule CNF:

- L'ordine delle clausole non importa se una formula CNF ϕ è ottenuta riordinando i termini di una formula CNF $\phi\prime$ allora $\phi \equiv \phi\prime$.

- Le clausole uguali posso essere unite in una unica clausola.
- Una formula in CNF può essere vista come un insieme clausole.
- **Esistenza:** ogni formula può essere riscritta in CNF.
- **Equivalenza:** $\models$ CNF(ϕ)$\equiv \phi$.
- **Funzione CNF:** data una formula PL ϕ la funzione CNF(...) che trasforma ϕ nella sua forma CNF è definita ricorsivamente:

$$\begin{aligned}
CNF(p) &= \text{p se p} \in \textbf{PROP} \\
CNF(\neg p) &= \neg\text{p se p} \in \textbf{PROP} \\
CNF(\phi \supset \psi) &= CNF(\neg\phi) \otimes CNF(\psi) \\
CNF(\phi \wedge \psi) &= CNF(\phi) \wedge CNF(\psi) \\
CNF(\phi \vee \psi) &= CNF(\phi) \otimes CNF(\psi) \\
CNF(\phi \equiv \psi) &= CNF(\phi \supset \psi) \wedge CNF(\psi \supset \phi) \\
CNF(\neg\neg\phi) &= CNF(\phi) \\
CNF(\neg(\phi \supset \psi)) &= CNF(\phi) \wedge CNF(\neg\psi) \\
CNF(\neg(\phi \wedge \psi)) &= CNF(\neg\phi) \otimes CNF(\neg\psi) \\
CNF(\neg(\phi \vee \psi)) &= CNF(\neg\phi) \wedge CNF(\neg\psi) \\
CNF(\neg(\phi \equiv \psi)) &= CNF(\phi \wedge \neg\psi) \otimes CNF(\psi \wedge \neg\phi)
\end{aligned}$$

  Dove $(C_1 \wedge \cdots \wedge C_n) \otimes (D_1 \wedge \cdots \wedge D_m)$ è definito come:

$$(C_1 \vee D_1) \wedge \cdots \wedge (C_1 \vee D_m) \wedge \cdots \wedge (C_n \vee D_1) \wedge \cdots \wedge (C_n \vee D_m)$$

Esempio di CNF

Proviamo ora a trasformare in CNF la seguente formula:

$$\begin{gathered}
p1 \equiv (p2 \equiv (p3 \equiv (p4 \equiv (p5 \equiv p6)))) \\
CNF(p1 \supset (p2 \equiv (p3 \equiv (p4 \equiv (p5 \equiv p6))))) \wedge CNF((p2 \equiv (p3 \equiv (p4(p5 \equiv p6)))) \supset p1) \\
\vdots
\end{gathered}$$

la lunghezza della formula diventa di lunghezza esponenziale se si continua, nel caso peggiore la formula CNF(ϕ) è esponenzialmente lunga rispetto a ϕ però calcolare la validità/insoddisfabilità di una formula CNF richiede un tempo lineare.

II.3.3 Soddisfabilità di una formula in CNF

Sia $CNF(\phi) = C_0, \ldots, C_n$ dove $C_0, \ldots, C_n$ sono le clausola in CNF(ϕ), allora vale che:

- $I \models \phi$ se e solo se $I \models C_i$ per ogni i=0, ..., n.
- $I \models C_i$ se e solo se per un letterale $k \in C_i$, $I \models k$.

Per vedere se un modello I soddisfa N non abbiamo bisogno di conoscere tutti i valori dei letterali che appaiono in N.
Per esempio se $I(P)$=True e $I(q)$=false possiamo dire che $I \models \{\{p, q, \neg r\}, \{\neg q, s\}\}$.
Possiamo usare una funzione parziale per assegnare ad alcune variabili dell' alfabeto il valore di verità, grazie a questa valutazione parziale possimo dire che i letterali o le clausole sono True, False o Undefined.

- **True:** una clausola è True se secondo I almeno uno dei suoi ltterali è True.
- **False:** una clausola è falsa se tutti i suoi letterali sono falsi.
- **Undefined:** quando il valore di verità dei suoi letterali è irrilevante per l'interpretazione corrente.

Semplificazione di formule attraverso letterali positivi: per una formula in CNF ϕ ed il termine p allora $\phi|_p$ indica la formula ottenuta da ϕ con:

- Rimpiazzando tutte le occorrenze di p con il valore $\top$.
- Semplificando il risultato rimuovendo:
  - Le clausole contenenti il termine disgiuntivo $\top$.
  - I letterali $\neg\top = \bot$ nelle formule rimanenti.

Semplificazione di formule attraverso letterali negativi: per una formula in CNF ϕ ed il termine p allora $\phi|_{\neg p}$ indica la formula ottenuta da ϕ con:

- Rimpiazzando tutte le occorrenze di $\neg p$ con il valore $\bot$.
- Semplificando il risultato rimuovendo:
  - Le clausole contenenti i termini di disgiunzione $\neg\top = \bot$.
  - I letterali $\top$ nelle clausole rimanenti.

Soddisfabilità di una CNF: sia una $CNF(\phi) = C_0 \dots C_n$ dove i termini sono le clausole delle formula, iteriamo il processo di valutazione letterale, alla fine possiamo finire con due alternative:

1. $\{\}$, cioè un insieme vuoto di clausole quindi ϕ è soddisfabile.
2. $\{\dots\{\}\dots\}$ se sono riuscito a semplificare solo parte delle clausole vuol dire che ϕ è insoddisfabile.

Esempio

$$(p \vee q) \wedge (p \vee \neg p) \wedge (\neg q \vee q) \wedge (\neg q \vee \neg p) \wedge (\neg q \vee \neg p) \wedge (\neg q \vee \neg p) \wedge (\neg q \vee q) \wedge (p \vee \neg p) \wedge (p \vee q)$$

Prima cosa faccio il merge delle clausole uguali:

$$(p \vee q) \wedge (p \vee \neg p) \wedge (\neg q \vee q) \wedge (\neg q \vee \neg p)$$

Ora divido in insiemi:

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

Posso ora applicare una semplificazione per termini positivi in p quindi $\phi|_p$

$$\{\{\top, q\}, \{\top, \bot\}, \{\neg q, q\}, \{\neg q, \bot\}\}$$

Rimango quindi con:

$$\{\{q\}, \{\neg q, q\}, \{\neg q\}\}$$

Semplifico ora in $\neg q$ quindi $\phi|_{\neg q}$

$$\{\{\bot\}, \{\top, \bot\}, \{\top\}\}$$

Semplificando rimango con $\{\}$ quindi la formula è soddisfabile.

II.3.4 La procedura di decisione

II.3.4.1 Semplificazione con unità di propagazione

Clausola unita: se una formula in CNF ϕ contiene una clausola $C = \{I\}$ che consiste un singolo letterale I nel suo insieme.

Esempio

Prendiamo la formula in CNF $\phi = \{\{p\}, \{\neg p\}, \{\neg q, r\}\}$ è soddisfabile solo se se esiste un'interpretazione I tale che $I \models \phi$.

$$\begin{gathered}\{\{p\}, \{\neg p\}, \{\neg q, r\}\}|_p \\ \{\{\top\}, \{\bot\}, \{\neg q, r\}\} \\ \{\{\}, \{\neg q, r\}\} \\ \{\dots, \{\}, \dots\}\end{gathered}$$

Quindi la formula non è soddisfabile.

Osservazioni

- Ci sono casi in cui l'unità di propagazione non genera una delle condizioni di terminazione, quindi dobbiamo asseganre noi "a occhio" i valori di verità ai letterali.
- Ogni letterale genererà un ramo per ogni valore di verità, da qui nasce la quantità esponenziale di funzioni da analizzare.

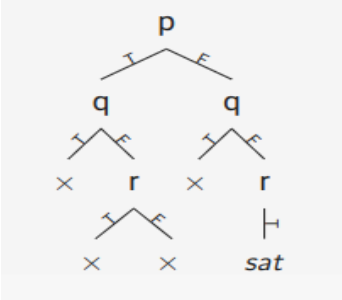
Algoritmo DPLL

```
DPLL(φ, I)
if φ contains the empty clause {} then
    return False;
end
if φ = {} then
    exit with I;
end
select a literal I ∈ C ∈ φ
DPLL(φ|k , I ∪ (I(k) = true)) or DPLL(φ|¬k , I ∪ (I(k) = true))
```

Per arrivare ad una soglia di efficenza c'è bisogno di fare delle scelte euristiche che se si rivelano sbagliate vanno corrette con backtrack.

Esempio

Considerando la formula $(p \vee \neg q) \wedge (p \vee r) \wedge (\neg p)$ dobbiamo allora creare l'albero:



II.3.4.2 Algoritmo DPLL con unità di propagazione

```
Input: un insieme di clausole φ.
Output: Una tabella di verita' che indica quando φ e' soddisfabile.

function DPLL(φ)
    while esiste una clausola {l} in φ do
        φ ← unit-propagate(l, φ);
    while esiste un letterale L che appare in φ do
        φ ← pure-literal-assign(l, φ);
    if φ is empty then return true;
    if φ contains an empty clause then return false;
    l ← select-literal(φ);
    DPLL(φ ∧ {l}) or DPLL(φ ∧ {¬ (l)});
```

Chapter II.4

Decisione procedura basata su Tableau

II.4.1 Nozioni di base

Dobbiamo capire perchè i Tableau possono risolvere il problema di SAT.
Principio di confutazione: ci permette di trasformare un problema di conseguenza logica in uno di insoddisfabilità.

$$\Gamma \models \phi \text{ se e solo se } \Gamma \cup \{\neg\phi\}$$

Come con DPLL vengono dati input all'algoritmo solo le formule rilevanti per la valutazione.
Anche se con i Tableau non serve che la formula sia in CNF e il passo di inferenza è diverso da DPLL anche se creano un algoritmo per la nozione di implicazione.
I Tableau sono, infatti, una rappresentazione diretta della definizione di implicazione questo lo rende concettualmente più semplice da capire ma anche più difficile da scalare su formule complesse.

II.4.2 Sistema Tableau

Tableau: il metodo basato su Tableau è composto da 3 elementi:

1. Un insieme di premesse Γ e di conclusioni ϕ.
2. Un compito, provare che $\Gamma \models \phi$.
3. Le procedure che dimostrano che $\Gamma \cup \{\neg\phi\}$ non è soddisfabile, oppure che $\Gamma \cup \{\phi\}$ è valido quindi soddisfabile

Tableau proposizionali: è un albero con radice dove:

- Ogni nodo indica una proposizione.
- La radice è la formula che dobbiamo provare insoddisfabile.
- I fogli di un nodo n sono generati con apposite regole di espansione ad n o ad uno dei suoi antenati.

II.4.2.1 Regole α

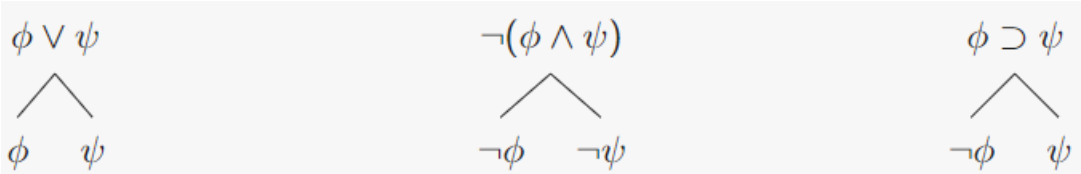
$\phi \wedge \psi$	$\neg(\phi \vee \psi)$	$\neg(\phi \supset \psi)$
$\downarrow$	$\downarrow$	$\downarrow$
ϕ	$\neg\phi$	ϕ
ψ	$\neg\psi$	$\neg\psi$

Le regole α sono congiuntive quindi creano un solo figlio.

Esempio

$$\neg(p \supset (q \vee \neg(p \wedge r)))$$
$$\downarrow$$
$$p$$
$$\neg(q \vee \neg(p \wedge r))$$
$$\downarrow$$
$$\neg q$$
$$p \wedge r$$
$$\downarrow$$
$$p$$
$$r$$

II.4.2.2 Regole β



Le regole β sono disgiuntive quindi creano due rami distinti che sono entrambi da completare, e quindi bene lasciare queste regole per ultime e usarle solo se necessario.

II.4.2.3 Controllare la soddisfacibilità

Se vogliamo controllare che Γ sia soddisfabile metto nella radice del Tableau Γ e applico le regole, se almeno un ramo non chiude vuol dire che che è soddisfabile e dal quel ramo possiamo prendere i valori da dare ai termini atomici.

II.4.2.4 Derivazione

Siano ϕ una formula proposizionale e Γ un insieme di formule proposizionali, scriveremo $\Gamma \vdash \phi$ se esiste un Tableau chiuso per $\Gamma \cup \{\neg\phi\}$.

II.4.2.5 Utilità

- Una formula è insoddisfabile se e solo se oddisfabile, per testare che ϕ sia valida basta testare che il Tableau chiuda tutti i rami di $\neg\phi$.
- Per verificare che ϕ è una conseguenza logica di Γ quindi $\Gamma \models \phi$basta verificare con un Tableau che $\Gamma \wedge \neg phi$ che un ramo chiuda.
- Per verificare l'equivalenza logica basta ricondursi al caso di conseguenza logica.

Per riassumere:

Formula	Tableau	Significato
p	Chiude	p è insoddisfabile, $\neg p$ è valida
p	Aperto	p è soddisfabile
$\neg p$	Chiude	$\neg p$ è insoddisfabile, p è valida
$\neg p$	Aperto	$\neg p$ è soddisfabile

II.4.2.6 Teoremi finali

- **Terminazione:** per ogni Tableau proposizionale dopo un numero finito di espansioni nessuna regola è più applicabile.
- **Equità:** definiamo così un Tableau proposizionale quando ogni non-letterale di un ramo viene analizzato in quel ramo.

- **Solidità:** Se $\Gamma \vdash \phi$ allora $\Gamma \models \phi$.
- **Completezza:** Se $\Gamma \models \phi$ allora $\Gamma \vdash \phi$.

Part III

Logica del Primo Ordine

Chapter III.1

Logica

III.1.1 Intuizione

Fino ad ora abbiamo incontrato solo la logica proposizionale che purtroppo pecca per quanto riguarda il potere espressivo. La difficoltà più grande è che possiamo espremire solo insiemi finiti.

Caratteristiche della logica del primo ordine

- Termini complessi e formule complesse.
- Le formule primitive molte volte non sono parte del linguaggio.
- Termini e formule atomiche sono interpretate su un dominio di fatti e entità.
  Le formule complesse sono interpretate attraverso un dominio di giudizi.
- Le formule complesse sono formate un qualsiasi numero di connettivi proposizionali più un numero di quantificatori.
- I quantificatori sono: $\forall$ e $\exists$.

Esempi di espressività

Come posso esprimere le seguenti frasi?

- Mary is a person.
- John is a person.
- Mary is mortal.

Visto che sono proposizioni atomiche non primitive:

- Person(Mary)
- Person(John)
- Mortal(Mary)

La logica del primo ordine risulta più utile nei seguenti casi:

- All people are mortal.
- There is (at least) a person who is a spy.

Che posso esprimere come:

- $\forall$ people.Mortal(people)
- $\exists$ person.Spy(person)

Con la logica del primo ordine posso anche annidare formule.

- The father of Luca is Italian.

Diventa:

- Italian(fatherOf(Luca))

Infine posso usare i concatenatori visti nella logica proposizionale.

- Every Natural number is either even or odd.

La posso vedere come:

- $\forall$ number.(odd(number) $\vee$ even(number))

III.1.2 Sintassi

La sintassi di un linguaggio L è definita come segue:

- Un insieme di termini primitivi detti termini alfabetici.
- Un insieme di regole per la formazione di formule che permettono di comporre termini atomici.
- Un insieme di regole da termine a frase.
- Un insieme di frasi primitive chiamato alfabeto delle frasi.
- Un insieme di regole per la formazione di frasi.
- Un insieme di regole da frasi a termini.

III.1.2.1 Definizioni

- **Alfabeto:** L'alfabeto della logica del primo ordine è formato da due insimi di simboli simboli logici e simboli non logici.
- **Simboli logici:** I simboli logici sono i seguenti:
  - In modo opzionale le costanti logiche $\bot$ e $\top$.
  - In modo opzionale il simbolo di ugualgianza $=$.
  - Connettivi della logica proposizionale $\wedge$, $\vee$, $\supset$, $\neg$, $\equiv$.
  - Quantificatori $\forall$ e $\exists$.
  - Un insieme infinito di simboli per le variabili $x_1, x_2, \ldots$
- **Simboli non logici:** I simboli non logici sono i seguenti:
  - Un insieme di costanti $c_1, c_2, \ldots$
  - Un insieme di funzioni con la loro arietà $f_1, f_2, \ldots$
  - Un insieme di predicati $P_1, P_2, \ldots$
- **Termini:**
  - Ogni costante c_i e variabile x_i è un termine.
  - Se $t_1 \ldots t_n$ sono termini e f_i una funzione che ha nella firma n elementi, allora $f(t_1 \ldots t_n)$ è un termine.
- **Well-formed formulas**
  - Se $t_1 \ldots t_n$ sono termini e P è un simbolo di relazione con arietà uguale a n, allora $P(t_1 \ldots t_n)$ è una formula atomica.
  - Se A e B sono formule allora $\top$, $\bot$, A$\wedge$B, A$\vee$B, A$\supset$B, $\neg$A e A$\equiv$B sono formule.
  - Se A è uina formula e x una variabile allora $\forall x$.A e $\exists x$.A sono fomrule.

III.1.3 Funzione di interpretazione

Interpretazio2ne FOL: Una interpretazione del primo ordine per il linguaggio:

$$L = (c_1, c_2, \dots, f_1, f_2, \dots, P_1, P_2, \dots)$$

è la tupla

$$\langle \Delta, I \rangle$$

Dove:

- Δ è un insieme non vuoto chiamato **dominio di interpretazione**.
- I è una funzione chiamata **funzione di interpretazione** definita come:
  - $I(c_i) \in \Delta$ (elemento del dominio)
  - $I(f_i) : \Delta^n \to \Delta$ (n-aria funzione sul dominio)
  - $I(P_i) \subseteq \Delta^n$ (n-aria relazione sul dominio)

dove:

- n è l'arietà di f_i e P_i
- $\Delta^n = \Delta \times \cdots \times \Delta$

III.1.3.1 Interpretazione di ground formulas (formule senza variabili)

L'interpretazione di un termine è un' entità nel dominio, possiamo avere sinonimità ma non polisemia.
I simboli di funzione sono usati per generare la descrizione di termini complessi, i simboli di una funzione n-aria denotano lo spazio di tutti i possibili termini.
Le entità generate da una funzione n-aria sono rappresentati come una (n+1)-aria tupla.
Predicati n-ari sono usati per generare formule atomiche, i fatti generati da un predicato n-ario sono rappresentati da un tupla n-aria.

III.1.3.2 Interpretazione di una formula con variabili

Una variabile presente in una formula atomica non può essere interpretata come elemento del dominio, nello stesso momento ad una formula atomica deve essere assegnata un'interpretazione.
Possiamo interpretare una variabile come qualcosa la cui interpretazione ci è ancora ignota ma, che in seguito potrà avere un valore compreso nel dominio.
N.B. nessun valore del dominio può essere scartato a priori.

Assegnamento delle variabili

Sia L un linguaggio del primo ordine e Δ il suo dominio di interpretaione.
Sia A$\in L$ una formual del primo ordine e Var(A)=$\{x_1, \dots, x_n\}$ l'insieme delle variabili occorenti in A.
Un assegnamento a è una funzione dall'insieme delle variabili al dominio di interpretazione Δ:

$$a : \text{Var(A)} \to \Delta$$

Scriveremo $a[x/d]$ per inidcare l'assegnamento a su tutte le variabili tranne x che sono associate a d con $d \in \Delta$.

III.1.4 Conseguenza logica

Siano L un linguaggio formale, $T \subseteq L$ una teoria formale, $I : L \to D$ una funzioane di interpretazione per L, $M \subseteq D$ un modello per T.
Allora $\models_L$ associa cosa c'è di vero in M con le wffs in T:

$$\models_L \subseteq M \times T$$

Possiamo anche scrivere:

$$M \models_L T$$

III.1.4.1 Occorenze libere delle variabili

Occorrenza libera: una variabile occurre libermanete se vale una delle seguenti affermaioni:

- Un'occorrenza di x in un termine t_k è libera in $P(t_1, \ldots, t_k, \ldots, t_n)$.
- Ogni occorenza libera di x in una formula ϕ o ψ rimane liber anche in $\phi \wedge \psi$, $\phi \wedge \psi$, $\phi \supset \psi$, $\phi \equiv \psi$ e $\neg\phi$.
- un'occorrenza libera di x in una formula ϕ rimane libera in $\forall y.\phi$ e $\exists y.\phi$ se y è diversa da x.

Ground formula: una formula si dice grounded se non presenta variabili.
Closed formula: una formula si dice chiusa se non contiene occorrenze libere di una variabile.

III.1.4.2 Soddisfacibilità con riferimento all'assegnamento

Un'interpretazione I soddisfa una formula ϕ considerando un assegnamento a secondo le seguenti formule:

$$\begin{array}{ccc}
I \models t_1 = t_2[a] & \text{iff} & I(t_1)[a] = I(t_2)[a] \\
I \models P(t_1, \ldots, t_n)[a] & \text{iff} & < I(t_1)[a], \ldots, I(t_n)[a] > \in I(P) \\
I \models (\phi \wedge \psi)[a] & \text{iff} & I \models \phi[a] \text{ and } I \models \psi[a] \\
I \models (\phi \vee \psi)[a] & \text{iff} & I \models \phi[a] \text{ or } I \models \psi[a] \\
I \models (\phi \supset \psi)[a] & \text{iff} & I \not\models \phi[a] \text{ or } I \models \psi[a] \\
I \models \neg\phi[a] & \text{iff} & I \not\models \psi[a] \\
I \models (\phi \equiv \psi)[a] & \text{iff} & I \models \phi[a] \text{ iff } I \models \psi[a] \\
I \models \exists x\phi[a] & \text{iff} & \text{there is a } d \in \Delta \text{ such that } I \models \phi[a[x/d]] \\
I \models \forall x\phi[a] & \text{iff} & \text{for all } d \in \Delta, I \models \phi[a[x/d]]
\end{array}$$

III.1.4.3 Soddisfacibilità

Modello rispetto all'assegnamento: un interpretazione I è un modello di ϕ secondo l'assegnamento a se:

$$I \models \phi[a]$$

Modello, soddisfacibilità: un interpretazione I è un modello per ϕ se esiste un assengamento a tale che I sia un modello per ϕ secondo a, una fomrula ϕ è soddisfacibile se esiste una qualsiasi I e una a tale che $I \models \phi[a]$.

Implicazione, verità, soddisfacibilità: le seguenti affermazioni sono equivalenti all'enuinciato $I \models A$:

- La funzione di interpretazione (e anche modello) I implica la formula A.
- La formula A è vera nella funzione di interpretazione (e anch modello) I.
- La formula A è soddisfatta dall'interpretaione (e modello) I.

Chapter III.2

Modeling

III.2.1 Teoremi principali

III.2.1.1 Quantificatori e connettivi proposizionali

- $\forall x.(\phi(x) \wedge \psi(x)) \equiv \forall x.\phi(x) \wedge \forall x.\psi(x)$ è valida
- $\exists x.(\phi(x) \vee \psi(x)) \equiv \exists x.\phi(x) \vee \exists x.\psi(x)$ è valida
- $\forall x.(\phi(x) \vee \psi(x)) \equiv \forall x.\phi(x) \vee \forall x.\psi(x)$ non è valida
- $\exists x.(\phi(x) \wedge \psi(x)) \equiv \exists x.\phi(x) \wedge \exists x.\psi(x)$ non è valida
- $\forall x.\phi(x) \equiv \neg\exists x.\neg\phi(x)$ è valida
- $\forall x.\exists x.\phi(x) \equiv \exists x.\phi(x)$ è valida
- $\exists x.\forall x.\phi(x) \equiv \forall x.\phi(x)$ è valida
- $\forall x.\phi(x) \equiv \exists x.\phi(x)$ non è valida
- $\forall x\exists y.\phi(x,y) \equiv \exists y\forall x.\phi(x,y)$ non è valida

III.2.1.2 Termini liberi per una variabile in una formula

Sia x una variabile occorente in un termine t, allora t è libero rispetto alla variabile x in una formula ϕ se e solo se tutte le occorrenze di x non occorrono in ϕ all'interno dello scope del quantificatore, quindi se tutte le occorenze di x in ϕ sono libere.

III.2.2 Errori nella formalizzazione

- $\supset$ **and** $\forall$**:** lo scope di un quantificatore universale può essere indebolito facendolo diventare la conseguenza di un implicazione dove la premessa (dell' implicazione) riduce lo scope.
  Esempio: la formula:

  $$\forall x(\text{WorksAt}(\text{UniTn}, x) \supset \text{Smart}(x))$$

  Vuol dire:

  "Everybody working at UniTn is smart"

  Se avessi usato una congiunzione avrei rafforzato la formula anzichè ristretto lo scope:

  $$\forall x(\text{WorksAt}(\text{UniTn}, x) \wedge \text{Smart}(x))$$

  Vuol dire:

  "Everybody works at UniTn and everyone is smart"

- $\wedge$ **and** $\exists$**:** lo scope di un quantificatore di esistenza può essere indebolito aggiungendo una congiunza, la quale aggiungerà un nuovo voncolo da rispettare.
  Esempio: la formula:

  $$\exists x(\text{WorksAt}(\text{UniTn}, x) \wedge \text{Smart}(x))$$

  Vuol dire:

  "There is a person working at UniTn and she is smart"

  Se avessi usato una implicazione avrei rafforzato la formula perchè partendo con una premessa vera l'interpretazione ha più modi di essere vera:

  $$\exists x(\text{WorksAt}(\text{UniTn}, x) \supset \text{Smart}(x))$$

  Vuol dire:

  "There is a person so that if (s)he works at UniTn then (s)he is smart"

- **Esiste almeno 1:** l'esistenza non pone alcun limite inferiore se devo prendere domini diversi.
  Esempio: Proviamo a scrivere la formula "There are at least two students attending the Logic class":

  $$\exists x_1 \exists x_2(\text{attend}(x_1, \text{Logic}) \wedge \text{attend}(x_2, \text{Logic}))$$

  Ma la formula così scritta non garantisce che x_1 e x_2 siano distinti, per cui dobbiamo specificarlo.

  $$\exists x_1 \exists x_2(\text{attend}(x_1, \text{Logic}) \wedge \text{attend}(x_2, \text{Logic}) \wedge x_1 \neq x_2)$$

- **Esietono almeno** n**:** in generale per garantrire che n elementi siano diversi usando l'esistenza dobbiamo avere una formula della seguente forma (ϕ sarà la formula da soddisfare):

  $$\exists x_1 \dots x_n(\bigwedge_{i=1}^{n} \phi(x_i) \wedge \bigwedge_{i \neq j=1}^{n} x_j \neq n_i)$$

- **Esistono al più** n**:** il quantificatore universale, invece, non impone un limite superiore.
  Per poter garantire un numero massimale forziamo che differenti quantificatori universali indichino lo stesso elemento (ϕ è la funzione da soddisfare).

  $$\forall x_1 \dots x_{n+1}(\bigwedge_{i=1}^{n+1} \phi(x_i) \supset \bigvee_{i \neq j=1}^{n+1} x_j = n_i)$$

Chapter III.3

Calculus

III.3.1 Nozioni basiche

Soddisfacibilità con riferimento ad un assegnamento: un'interpretazione I soddisfa una formula ϕ con un assegnamento a secondo le seguenti formule:

$$\begin{array}{rcc}
I \models t_1 = t_2[a] & \text{iff} & I(t_1)[a] = I(t_2)[a] \\
I \models P(t_1, \ldots, t_n)[a] & \text{iff} & < I(t_1)[a], \ldots, I(t_n)[a] > \in I(P) \\
I \models (\phi \wedge \psi)[a] & \text{iff} & I \models \phi[a] and I \models \psi[a] \\
I \models (\phi \vee \psi)[a] & \text{iff} & I \models \phi[a] or I \models \psi[a] \\
I \models (\phi \supset \psi)[a] & \text{iff} & I \not\models \phi[a] or I \models \psi[a] \\
I \models \neg\phi[a] & \text{iff} & I \not\models \phi[a] \\
I \models (\phi \equiv \psi)[a] & \text{iff} & I \models \phi[a] iff I \models \psi[a] \\
I \models \exists x\phi[a] & \text{iff} & \text{there is a } d \in \Delta \text{ such that } I \models \phi[a[x/d]] \\
I \models \forall x\phi[a] & \text{iff} & \text{for all } d \in \Delta, I \models \phi[a[x/d]]
\end{array}$$

Modello rispetto ad un assegnamento: un'interpretazione I è un modello di ϕ secondo un'assegnamento a se:

$$I \models \phi[a]$$

Insoddisfacibile: una formula è insoddisfacibile se non è soddisfacibile.

Validità: una formula ϕ è valida se per ogni interpretazione I e ogni assegnamento a vale che:

$$I \models \phi[a]$$

Nella logica del primo ordine valgono le stesse proprietà della logica proposizionale, quindi:

$$\text{Valida} \supset \text{Soddisfacibile} \supset \neg \text{ Insoddisfacibile}$$

$$\text{Insoddisfacibile} \supset \neg \text{ Soddisfacibile} \supset \neg \text{ Valida}$$

se A è	allora ¬ A è
Valida	Insoddisfacibile
Soddisfacibile	¬ Valida
¬ Valida	Soddisfacibile
Inddisfacibile	Valida

Conseguenza logica: una formula ϕ è una conseguenza logica di un insieme di formule Γ se per ogni interpretazione I e ogni assegnamento a

$$I \models \Gamma[a] \text{ implica } I \models \phi[a]$$

dove $I \models \Gamma[a]$ significa che I soddisfa tutte le formule di Γ sotto a.

Equivalenza logica: due formule ϕ e ψ sono logicamente equivalente se e solo se $\phi \models \psi$ e $\psi \models \phi$, scriveremo:

$$\models \phi \equiv \psi$$

Teorema di deduzione: $\Gamma, \phi \models \psi$ se e solo se $\Gamma \models \phi \supset \psi$.

Principio di refutazione: $\Gamma \models \phi$ se e solo se $\Gamma \cup \{\neg\phi\}$ è insoddisfacibile.

Formule chiuse: nelle formule chiuse validità, soddisfacibilità, insoddisfacibile, conseguenza logica e equivalenza logica non dipendono dall'assegnamento, infatti nella notazione scriveremo solo ϕ anzichè $\phi[a]$.

Invarianza dell'assegnazione: siano $[a^1]$ e $[a^2]$ due assegnamenti allora $I \models [a^1]$ se e solo se $I \models [a^2]$ quando $[a^1]$ e $[a^2]$ coincidono in una variabile libera in ϕ.

Chapter III.4

Inferenza della DPLL

III.4.1 Dominio finito

Estensione finita del predicato: questa assunzione enuncia che un predicato P è vero solo per un sottoinsieme finito del proprio dominio originale.
L'assunzione può essere formalizzata nella formula:

$$\forall x(P(x) \equiv (x = c_1 \vee \cdots \vee x = c_n))$$

Assunzione del nome univoco: un linguaggio deve contenere uno e un solo nome per ogni elemento del dominio, assumendo che un linguaggio contenga le costanti $c_1, \ldots, c_n$ abbiamo:

$$\phi_\Delta = \{c_1, \ldots, c_n\}$$
$$\{c_1, \ldots, c_n\} = \Big(\bigwedge_{i \neq j = 1}^{n} c_i \neq c_j \wedge \forall x \Big(\bigvee_{i=1}^{n} c_i = x \Big) \Big)$$

Espansione dei quantificatori: secondo l'ipotesi di un dominio finito con dei nomi univoci la logica del primo ordine può essere proposizionalizzata secondo le seguenti equivalenze:

- $\phi_\Delta = \{c_1, \ldots, c_n\} \models \forall x.\phi(x) \equiv \phi(c_1) \wedge \cdots \wedge \phi(c_n)$
- $\phi_\Delta = \{c_1, \ldots, c_n\} \models \exists x.\phi(x) \equiv \phi(c_1) \vee \cdots \vee \phi(c_n)$

Esempio: prendiamo la formula in linguaggio naturale "Everybody living in Toulouse speak french or spanish", tradotta in FOL risulta in:

$$\forall x.(\text{Lives}(x, \text{Toulouse}) \supset (\text{Speaks}(x, \text{French}) \vee \text{Speaks}(x, \text{Spanish})))$$

Assumiamo che l'insieme delle persone sia:

$$\Delta_1 = \{\text{Robert, Luis, Naji}\}$$

Allora possiamo rendere ground la precende formula, e riscriverla come segue:

$$\begin{gathered}(\text{Lives(Robert,Toulouse)} \supset \text{Speaks(Robert,French)} \vee \text{Speaks(Robert,Spanish)}) \wedge \\ (\text{Lives(Luis,Toulouse)} \supset \text{Speaks(Luis,French)} \vee \text{Speaks(Luis,Spanish)}) \wedge \\ (\text{Lives(Naji,Toulouse)} \supset \text{Speaks(Naji,French)} \vee \text{Speaks(Naji,Spanish)})\end{gathered}$$

Esempio: prendiamo la formula in linguaggio naturale "If someone is noisy, everyone is annoyed", tradotta in FOL risulta:

$$\exists x.(\text{Noisy}(x) \supset \forall y.(\text{Annoyed}(y)))$$

Assumiamo che l'insieme delle persone sia:

$$\Delta_1 = \{\text{Marco, Francesco, Pierre}\}$$

Allora possiamo rendere ground la precende formula, e riscriverla come segue (**N.B.** non viene detto che $x \neq y$):

$$\begin{gathered}(\text{Noisy(Marco)} \supset \text{Annoyed(Marco)} \wedge \text{Annoyed(Francesco)} \wedge \text{Annoyed(Pierre)}) \vee \\ (\text{Noisy(Francesco)} \supset \text{Annoyed(Marco)} \wedge \text{Annoyed(Francesco)} \wedge \text{Annoyed(Pierre)}) \vee \\ (\text{Noisy(Pierre)} \supset \text{Annoyed(Marco)} \wedge \text{Annoyed(Francesco)} \wedge \text{Annoyed(Pierre)})\end{gathered}$$

III.4.2 PNF - Prenex Normal Form

Prenex Normal Form: una formula in logica del primo ordine si dice essere in PNF se ha la seguente forma:

$$Q_1x_1Q_2x_2\dots Q_nx_n.P$$

dove:

- Q_i è un quantificatore di esistenza o universale.
- La o del punto precedente è esclusiva.
- P è un quantificatore libero.

$Q_1x_1Q_2x_2\dots Q_nx_n$ è detto prefisso mentre P è la matrice.

Ogni formula di logica del primo ordine può essere scritta in PNF.

Passi della PNF:

- Muovere le negazionbi all'interno finchè non ci saranno più quantificatori nello scope di una negazione.
- Rinominare le variabili finchè ogni variabile che segue un quantificatore avrà un nome univoco.
- Muovere all'esterno i quantificatori leggendo la formula da sinistra a destra.

Formula		**versione PNF**	**Assunzioni**
$\neg\forall x.\phi x$	$\equiv$	$\exists x.(\neg\phi(x))$	
$\neg\exists x.\phi x$	$\equiv$	$\forall x.(\neg\phi(x))$	
$\forall x.(\phi(x)) \wedge \psi$	$\equiv$	$\forall x.(\phi(x) \wedge \psi)$	$\forall x.\top$
$\exists x.(\phi(x)) \wedge \psi$	$\equiv$	$\exists x.(\phi(x) \wedge \psi)$	$\exists x.\top$
$\forall x.(\phi(x)) \vee \psi$	$\equiv$	$\forall x.(\phi(x) \vee \psi)$	$\forall x.\top$
$\exists x.(\phi(x)) \vee \psi$	$\equiv$	$\exists x.(\phi(x) \vee \psi)$	$\exists x.\top$
$\forall x.(\phi(x)) \supset \psi$	$\equiv$	$\exists x.(\phi(x) \supset \psi)$	$\exists x.\top$
$\exists x.(\phi(x)) \supset \psi$	$\equiv$	$\forall x.(\phi(x) \supset \psi)$	$\forall x.\top$
$\psi \supset \forall x.(\phi(x))$	$\equiv$	$\forall x.(\psi \supset \phi(x))$	$\forall x.\top$
$\psi \supset \exists x.(\phi(x))$	$\equiv$	$\exists x.(\psi \supset \phi(x))$	$\exists x.\top$

Proposizione: ogni formula in PNF con la forma $\exists x_1\exists x_2\dots\forall x_{n-1}\forall x_n.P$ tale che:

- La sua matrice non contenga alcuna funzione e ...
- I suoi prefissi sono composti da una serie (anche vuota) di quantificatori di esistenza e una serie (anche vuota) di quantificatori universali.

è deducibile quindi valida/soddisfacibile/insoddisfacibile.

III.4.3 DPLL in FOL

Adesso abbiamo una formula in FOL che abbiamo trasformato in PNF e a cui abbiamo "ristretto" il dominio per farla diventare una formula in PL.
Essendo una formula in PL possimao applicare le formule per scriverla in CNF e poter applicare DPLL.

$$\begin{aligned}
CNF(p) &= \text{p se p} \in \textbf{PROP} \\
CNF(\neg p) &= \neg\text{p se p} \in \textbf{PROP} \\
CNF(\phi \supset \psi) &= CNF(\neg\phi) \otimes CNF(\psi) \\
CNF(\phi \wedge \psi) &= CNF(\phi) \wedge CNF(\psi) \\
CNF(\phi \vee \psi) &= CNF(\phi) \otimes CNF(\psi) \\
CNF(\phi \equiv \psi) &= CNF(\phi \supset \psi) \wedge CNF(\psi \supset \phi) \\
CNF(\neg\neg\phi) &= CNF(\phi) \\
CNF(\neg(\phi \supset \psi)) &= CNF(\phi) \wedge CNF(\neg\psi) \\
CNF(\neg(\phi \wedge \psi)) &= CNF(\neg\phi) \otimes CNF(\neg\psi) \\
CNF(\neg(\phi \vee \psi)) &= CNF(\neg\phi) \wedge CNF(\neg\psi) \\
CNF(\neg(\phi \equiv \psi)) &= CNF(\phi \wedge \neg\psi) \otimes CNF(\psi \wedge \neg\phi)
\end{aligned}$$

Dove $(C_1 \wedge \cdots \wedge C_n) \otimes (D_1 \wedge \cdots \wedge D_m)$ è definito come:

$$(C_1 \vee D_1) \wedge \cdots \wedge (C_1 \vee D_m) \wedge \cdots \wedge (C_n \vee D_1) \wedge \cdots \wedge (C_n \vee D_m)$$

Esempio

Abbuiamo la frase:

A mechanic likes herself

Tradotta in FOL risulta essere:

$$\exists x.(\text{Mechanic}(x) \wedge \text{Likes}(x, x))$$

Poniamo come dominio sottoinsieme del dominio Δ ={Amanda, Miriam}, quindi la formula ground risulta essere:

$$\psi = p \wedge q \vee p_1 \wedge q_1$$

Adesso che abbiamo una formula in PL possiamo riscriverla in CNF per applicare DPLL e vedere se è soddisfacibile.

$$(p \wedge q) \vee (p_1 \wedge q_1)$$
$$(p \wedge q) \otimes (p_1 \wedge q_1)$$
$$(p \vee p_1) \wedge (p \vee q_1) \wedge (q \vee p_1) \wedge (q \vee q_1)$$
$$(\top \vee p_1) \wedge (\top \vee q_1) \wedge (q \vee p_1) \wedge (q \vee q_1)|_p$$
$$(q \vee p_1) \wedge (q \vee q_1)$$
$$(\top \vee p_1) \wedge (\top \vee q_1)|_q$$

Chapter III.5

Inferenza dei Tableaux

III.5.1 Tableaux del primo ordine

Tableaux del primo ordine: è un albero radicato, ogni suo nodo contiene una frase in FOL.
Il figlio di un nodo n è generato applicando una serie di regole di espansione a n o a uno degli antenati di n.

Il metodo del tablueaux è composto da 3 elementi:

- Un dato insieme di premesse Γ e conclusioni ϕ.
- **Un compito:** provare che $\Gamma \models \phi$.
- **Procedura:** mostrare che $\Gamma \cup \{\neg\phi\}$ è non soddisfacibile che è equivalente a dimostrare che $\Gamma \cup \{\phi\}$ è soddisfacibile.

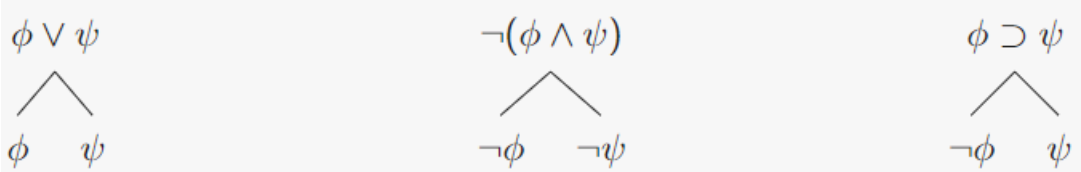
III.5.1.1 Regole α

Sono le stesse della logica proposizionale.

$$\begin{array}{c} \phi \wedge \psi \\ \downarrow \\ \phi \\ \psi \end{array} \qquad \begin{array}{c} \neg(\phi \vee \psi) \\ \downarrow \\ \neg\phi \\ \neg\psi \end{array} \qquad \begin{array}{c} \neg(\phi \supset \psi) \\ \downarrow \\ \phi \\ \neg\psi \end{array}$$

III.5.1.2 Regole β

Anche le regole β sono uguali a quelle delle logica proposizioanale.



III.5.1.3 Regole γ

$$\begin{array}{c} \forall x.\phi(x) \\ \downarrow \\ \phi(t) \end{array} \qquad \begin{array}{c} \neg\exists x.\phi(x) \\ \downarrow \\ \neg\phi(t) \end{array}$$

Dove t è un nuovo termine libero rispetto a x in ϕ.

III.5.1.4 Regole σ

$$\begin{array}{c} \neg\forall x.\phi(x) \\ \downarrow \\ \neg\phi(c) \end{array} \qquad \begin{array}{c} \exists x.\phi(x) \\ \downarrow \\ \phi(c) \end{array}$$

Dove c è una nuova costante non precedentemente apparsa nel tablueaux.

III.5.1.5 Sostituzione $\phi[x/t]$

In questa notazione x è una variabile libera mentre t è un termine, con la notazione $\phi[x/t]$ intendiamo sostituira tutte le occorrenze di x con t in ϕ.

Esempi:

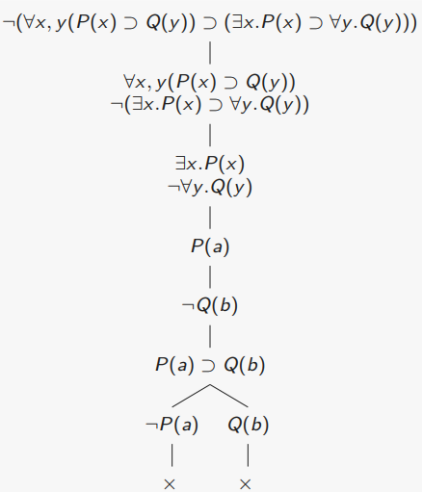
$$P(x, y, f(x))[x/a] = P(a, y, f(a))$$
$$\forall x.P(x, y)[x/b] = \text{Non permesso perchè x non è libera}$$
$$\exists x.P(x, x) \wedge Q(x)[x/c] = \exists x.P(x, x) \wedge Q(c)$$
$$P(x, g(y))[y/f(x)] = P(x, g(f(x)))$$
$$\forall x.P(x, y)[y/f(x)] = \text{Non permesso perchè f(x) non è libera per y nello scope di y}$$

III.5.1.6 Esempio di Tableaux

Verifichiamo ora la validità/soddisfacibilità della seguente formula:

$$\phi = \forall x, y(P(x) \supset Q(y)) \supset (\exists x.P(x) \supset \forall y.Q(y))$$

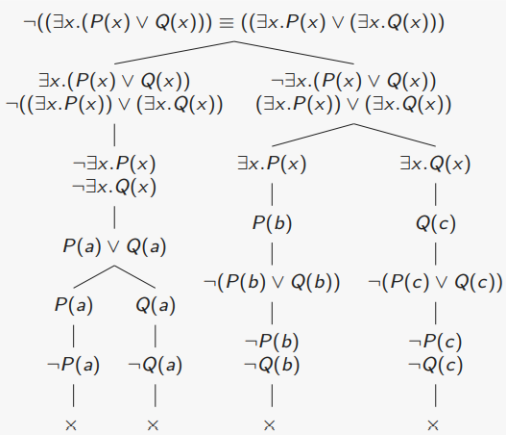
Verifichiamo la validità, quindi facciamo il tablueaux per $\neg\phi$.



Verifichiamo ora la validità/soddisfacibilità della seguente formula:

$$(\exists x.(P(x) \vee Q(x))) \equiv ((\exists x.P(x)) \vee (\exists x.Q(x)))$$

Verifichiamo la validità, quindi facciamo il tablueaux per $\neg\phi$.



III.5.2 Gestione dei Tableaux

In questa sezione ci sono tutte le regole di come bisogna espandere un Tableaux.

Espansione dell'esietnza: se possibile è bene iniziare applicando tutte le regole σ per ottenere tutti i termini con cui lavoreremo.

Espansione universale: dobbiamo applicare le regole γ dopo le regole σ, quando applicate devono matchare i termini già presenti nel Tableaux in particolare matchare le costanti introdotte con le regole σ.

Il termine inserito deve essere inserito perchè si verifichi sempre una chiusura del ramo.

Esempio

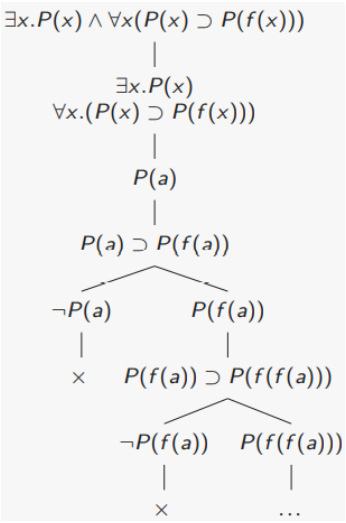
Verifichiamo se $\neg(\forall x.P(x) \land \exists x.\neg P(f(x)))$ è valida/soddisfacibile/insoddisfacibile.

$$\begin{array}{c} \forall x.P(x) \land \exists x.\neg P(f(x)) \\ \downarrow \\ \forall x.P(x) \\ \exists x.\neg P(f(x)) \\ \downarrow \\ \neg P(f(c)) \\ \downarrow \\ P(f(c)) \\ \downarrow \\ \times \end{array}$$

III.5.2.1 Non-terminazione

A differenza della logica proposizionale in FOL i modelli possono essere infiniti, esistono infatti formule soddisfatte solo da modelli infiniti.
Se cerchiamo di costruire un Tableaux per queste formule in cerca di un modello finiremo in un loop di espansioni.

Esempio



A differenza della logica proposizionale in FOL la costruzione dei Tableaux non assicura di avere una fine.

- Se la formule ϕ che indica la radice è insoddisfacibile allora il Tableaux ha una fine e può sempre chiudere.

- Se la formule ϕ che indica la radice è soddisfacibile la costruzione del Tableaux può finire a rimanere con i rami aperti oppure la costruzione può non terminare mai.

Ma come facciamo a sapere che siamo in un loop infinito?
Se non abbiamo ancora chiuso un Tableaux è perchè o può essere espanso in modo infinito oppure perchè non abbiamo trovato ancora un modo corretto di costruirlo.
Non possiamo determinarlo.

III.5.2.2 Contromodello

Ramo aperto saturato: un ramo aperto si chiama saturato se ogni non letterale è stato analizzato almeno una volta e ogni formula γ è stata istanziata con ogni simbolo, che possiamo costruire con i simboli di funzione, su quel ramo.

Prova fallimentare: un Tableaux con un ramo aperto e saturo non potrà mai essere chiuso, quindi possiamo fermarci subito e dichiarare quella prova fallimentare.

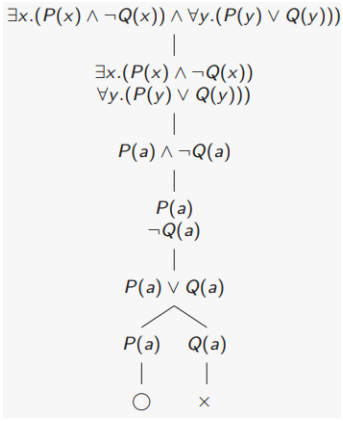
Un modello M, che è un interpretazione, indica come interpretare costanti (elementi del dominio), funzioni e simboli predicativi.

- Dominio: insieme di tutti i termini che possiamo costruire con i simboli di funzioni che appaiono sul ramo.
  E' possibile introdurre una costante fittizia per il valore di un termine.

- Simboli di funzione: interpretati come scritti o usando una costante fittizia.

- Simboli relazionali: interpretati in termini delle loro occorrenze nel ramo.

Esempio



Consideriamo il Tableaux sopra, dalla formula nel ramo aperto è possibile costrire un modello per la formula alla radice. Per la costruzione del modello prendo:

- Δ ={a} la costante apparsa nella formula.
- $I(P)$={a} perchè P(a) appare nel ramo aperto.
- $I(Q) = \{\}$ perchè la formula $\neg Q(a)$ appare nel ramo aperto.

Part IV

Logica Descrittiva

Chapter IV.1

Logica

IV.1.1 Intuizioni

La FOL con la sua espressività permette di usare:

- costanti
- variabili libere
- simboli di funzioni
- ...

Ma nella maggior parte delle applicazioni dell'informatica questa espressività non è richiesta.
La grande espressività della FOL è motivita dal suo uso, ma si porta dietro dei costi di analisi molto elevati.
Il linguaggio della FOL non ha struttura, risulta quindi appiattito, questo nonostante elementi diversi formalizzino intuizioni diverse.
La FOL (e la PL) nella loro semantica hanno lo scopo di dare giudizi riguardanti dei fatti e quindi anche selle conseguenze sui fatti risultati veri.
Per lo stesso motivo FOL e PL non forniscono alcuno strumento per ragionare sui fatti perchè codificati in formule atomiche, il predicato $P(t_1, \ldots, t_n)$ viene usato solo per determinare quando lo stato è vero o falso.

$$\begin{array}{rcl} I \models t_1 = t_2[a] & \text{iff} & I(t_1)[a] = I(t_2)[a] \\ I \models P(t_1, \ldots, t_n)[a] & \text{iff} & < I(t_1)[a], \ldots, I(t_n)[a] > \in I(P) > \end{array}$$

IV.1.1.1 Da FOL a logica descrittiva (DL)

Espressività: l'espressività è molto limitata perchè non abbiamo variabili libere, simboli di funzione e abbiamo solo predicati binari come simboli primitivi.

Decidibilità: dobbiamo ridurre i domini in domini finiti e usare l'assunzione dei nomi univoci, lo standard sono le pratiche dei domini relazionali.

Mancanza di struttura: per questa mancanza della FOL vengono esplicitate varie distinzioni:

- Come sono costrite le classi.
- Classi e ruoli (predicati unari vs binari).
- Schemi contro schemi popolati (TBox e ABox).
- Ragionamento schematico (knowledge-level) contro ragionamento ground (data-level).

Scopo: lo scopo della DL non è ragionare su cosa è vero ma ragionare sui fatti e sulle loro componenti.

IV.1.2 Due logiche

- **Conoscenze:**
  - Linguaggio naturale: informazioni e principi.
  - Computer Science: il nome delle classi, il nome delle relazioni.
  - Logica: predicati simbolici, qunatificatori universali e di esistenza.
- **Dati:**
  - Linguaggio naturale: informazioni fattuali o calcolo.
  - Computer Science: valori dei dati, enità e proprietà degli oggetti.
  - Logic: costanti, termini ground e formule ground.

L'usuale linguaggio è diviso in due insiemi ma collegati tra di loro:

- **TBox - la logica della conoscenza:** il linguaggio e la logica usati per specificare e ragionare sugli schemi usati per modellare i dati.
  Le TBox mappano direttamente in modelli Extended Entity-relationship (EER), relazioni dei DB e Knowledge Graphs (KG).
- **ABox - la logica dei dati:** il linguaggio e la logica usati per memorizzare e ragionare sui dati, si mappa sui dati contenuti nei DB e nei KG.

T significa **Terminologia** mentre **A** vuol dire **Asserzione**, sono collegati perchè le asserzioni fatte nelle ABox sfrutta la terminologia definita dalle TBox.
In questo corso vedremmo la logica descrittiva $\mathcal{ALC}$, ovvero la DL la cui parte proposizionele corrisponde perfettamente alla PL.

IV.1.3 Sintassi TBox

Alfabeto: un alfabeto è composto dai seguenti simboli:

- Simboli non logici: nomi di concetti (classi) quindi come persone, animali, cose ...
- Simboli logici:
  - $\sqcap$ congiunzione, $\sqcup$ disgiunzione e $\neg$ negazione di concetto.
  - $\top$ dominio di interpretazione e $\bot$ insieme vuoto.

Conectto: è un insieme di entità, la relazione formalizzata nei DB.
Di seguito parleremo di concetto (interpretazione) intendendo il nome del concetto (linguaggio).

Ruolo: un lafabeto contiene i seguenti simboli:

- Simboli non logici: nomi di ruoli (relazioni) $R_1, R_2, \ldots, R_n$.
- Simboli logici: $\forall$ e $\exists$.

Esempio di ruoli

Consideriamo i seguenti concetti e nomi:

- Nomi concetti:Vehicle, Boat, Bicycle, Car, Device, Wheel, Engine, Axle, Rotation, Water, Human, Driver, Adult, Child.
- Nomi ruoli: hasPart, poweredBy, capableOf, travelsOn, controls.

Formalizziamo una serie di frasi in linguaggio naturale.

1. Those vehicles that have wheels and are powered by an engine.
2. Those vehicles that have wheels and are powered by a human.

3. Those vehicles that travel on water.
4. Those objects which have no wheels.
5. Those objects which do not travel on water.
6. Those devices that have an axle and are capable of rotation.
7. Those humans who control a vehicle.
8. The drivers of cars.

Le trasposizioni in DL sono:

1. Vehicle⊓∃hasPart.Wheel⊓∃poweredBy.Engine
2. Vehicle⊓∃hasPart.Wheel⊓∃poweredBy.Human
3. Vehivle⊓∃travelsOn.Water
4. ∀hasPart.¬Wheel
5. ∀ travelsOn.¬Water
6. Device⊓∃hasPart.Axle⊓∃capableOf.Rotation
7. Human⊓∃controls.Vehicle
8. Driver⊓∃controls.Car

Descrizione concettuale: dato un insieme di nomi di concetti **C** e un insieme di nomi di ruoli **R**, l'insieme delle descrizioni dei concetti su **C** e **R** è definito come:

- Ogni nome di un concetto è una descrizione di concetto.
- $\top$ e $\bot$ sono descrizioni di concetti.
- Se C e D sono due concetti e r il nome di un ruolo, allora le seguenti formule sono descrizioni di concetti:
  - C $\sqcap$ D
  - C $\sqcup$ D
  - $\neg$ C
  - $\exists$ r.C
  - $\forall$ r.C

La descrizione concettuale non esprime le relazioni in un DB ma le assunzioni che si fanno per fare il design di un DB.

Dominio di interpretazione: un dominio di interpretazione in DL è un insieme i cui elementi sono chiamati individui o oggetti, nel seguente modo:

- Per ogni nome di concetto gli elementi appartenenti alla sua estensione sono parte del dominio.
- Per ogni nome di ruolo gli elementi (coppie ordinate di elementi) appartenenti alla sua estensione sono parte del dominio.

Interpretazione TBox: dato un insieme di nomi di concetti **C** e un insieme di nomi di ruoli **R**; dato un dominio Δ', un interpretazione $I = (\Delta', \cdot')$ è composta dal dominio con una funzione di mappatura che:

- Associa ogni nome di concetto C $\in$ **C** ad un insieme C$' \subseteq \Delta'$.
- Associa ogni nome di ruolo $r \in$**R** ad un insieme $r' \subseteq \Delta' \times \Delta'$.
- Associa descrizioni di concetti composti nel seguente modo:

- $\top' = \Delta'$
- $\bot' = \emptyset$
- $(C \sqcap D)' = C' \cap D'$
- $(C \sqcup D)' = C' \cup D'$
- $\neg C' = \Delta' \backslash C'$
- $(\exists r.C) = \{d \in \Delta' | \text{esiste una } e \in \Delta' \text{con } (d, e) \in r' \text{ e } e \in C'\}$
- $(\forall r.C) = \{d \in \Delta' | \text{per tutte le } e \in \Delta' \text{se } (d, e) \in r' \text{ allora } e \in C'\}$

IV.1.4 Teorie TBox

Concetto di inclusione: un concetto di inclusione è un'espressione della seguente forma:

$$C \sqsubseteq D$$

dove C e D sono descrizioni di concetti. $C \sqsubseteq D$ va letto come "C è sussunta da D".

Esempio

Consideriamo i seguenti concetti e nomi:

- Nomi concetti:Vehicle, Boat, Bicycle, Car, Device, Wheel, Engine, Axle, Rotation, Water, Human, Driver, Adult, Child.
- Nomi ruoli: hasPart, poweredBy, capableOf, travelsOn, controls.

Formalizziamo una serie di frasi in linguaggio naturale.

1. Boats have no wheels.
2. Cars and bicycles do not travel on water.
3. Drivers of cars are adults.
4. Humans are not vehicles.
5. Wheels or engines are not humans.
6. Humans are either adults or children.
7. Adults are not children.

Le trasposizioni in DL sono:

1. Boat$\sqsubseteq \forall$hasPart.$\neg$Wheel
2. Car$\sqcup$Bicycle$\sqsubseteq \forall$travelsOn.$\neg$Water
3. Driver$\sqcap\exists$controls.Car$\sqsubseteq$Adult
4. Human$\sqsubseteq \neg$Vehicle
5. Wheel$\sqcup$Engine$\sqsubseteq \neg$Human
6. Human$\sqsubseteq$Adult$\sqcup$Child
7. Adult$\sqsubseteq \neg$Child

Soddisfabilità dell'inclusione dei concetti: sia **C** un insieme di nomi di concetti e **R** un insieme di nomi di ruoli, allor adato un dominio Δ' e un'interpretazione $I = (\Delta', \cdot')$ abbiamo che:

$$\text{se } C' \subseteq D' \text{ allora } I \text{ è un modello di } C \sqsubseteq D$$

dove C e D sono descrizionali concettuali.

TBox: una TBox è un insieme finito di inclusioni concettuali, come un insieme finito di espressioni $C \sqsubseteq D$, dove C e D sono nomi di concetti (che possibilmente coinvolgono concetti composti).

Esempio

Consideriamo la descrizione concettuale "Dogs are Computer Science professors", che in DL diventa Dog $\sqsubseteq$ CS_Professor. Possiamo dimostrare l'insoddisfacibilità di questa frase con una TBox:

$$\mathcal{T} = \begin{cases} CS\_Professor \sqsubseteq Professor \\ Professor \sqsubseteq Person \\ Person \sqsubseteq \neg Dog \end{cases}$$

Modello per TBox: sia **C** un insieme di nomi di concetti e **R** un insieme di nomi di ruoli, dato Δ' e la sua interpretazione $I = (\Delta', \cdot')$, C e D descrizioni concettuali allora se $C' \subseteq D'$ per ogni $C \sqsubseteq D \in \mathcal{T}$ vuol dire che I è un moodello per $\mathcal{T}$.

Definizione di concetto: una definizione di concetto è un'espressione della seguente forma:

$$C \equiv D \text{ se e solo se } C \sqsubseteq D \text{ e } D \sqsubseteq C.$$

dove C e D sono descrizioni concettuali e $C \equiv D$ va letto come "C è equivalente a D".

Esempio

Consideriamo i seguenti concetti e nomi:

- Nomi concetti:Vehicle, Boat, Bicycle, Car, Device, Wheel, Engine, Axle, Rotation, Water, Human, Driver, Adult, Child.
- Nomi ruoli: hasPart, poweredBy, capableOf, travelsOn, controls.

Formalizziamo una serie di frasi in linguaggio naturale.

1. Cars are exactly those vehicles that have wheels and are powered by an engine
2. Bicycles are exactly those vehicles that have wheels and are powered by a human
3. Boats are exactly those vehicles that travel on water
4. Wheels are exactly those devices that have an axle and are capable of rotation
5. Drivers are exactly those humans who control a vehicle

Le trasposizioni in DL sono:

1. Car≡Vehicle⊓∃hasPart.Wheel⊓∃poweredBy.Engine
2. Bicyle≡ Vehicle⊓∃hasPart.Wheel⊓∃poweredBy.Human
3. Boat≡ Vehicle⊓∃travelsOn.Water
4. Wheel≡Device⊓∃hasPart.Axle⊓∃capableOf.Rotation
5. Driver≡Human⊓∃controls.Vehicle

TBox definita: una TBox è definita se soddiafa le seguenti proprietà:

- Ogni concetto può apparire al più una volta nel lato sinistro di una definizione.
- Sono presenti solo equivalenze concettuali ($C \equiv D$).
- E' aciclica quindi non esistono definizioni del tipo $C \equiv C$.

Esempio

$$\mathcal{T} = \begin{cases} Person \equiv \exists hasname.String \sqcap \forall HasJob.Organization \\ Woman \equiv Person \sqcap Female \\ Man \equiv Person \sqcap \neg Woman \\ Mother \equiv Woman \sqcap \exists hasChild.Person \\ Father \equiv Man \sqcap \exists hasChild.Person \\ Parent \equiv Father \sqcup Mother \end{cases}$$

Unfolding di concetti: un concetto è unfolded se tutte le sue componenti hanno una definizioni.

TBox unfolding: una TBox $\mathcal{T}$ può essere unfolded in una TBox $\mathcal{T}'$ se tutti i suoi concetti sono unfolded.

IV.1.5 Sintassi ABox

Asserzione: un asserzione è un espressione del tipo:

- C(a) asserzione di concetto
- r(a,b) asserzione di ruolo

dove a e b sono individui, C è un concetto (possibilmente composto) e r è il nome di un ruolo.

Interpertazione ABox: sia **C** un insieme di nomi di concetti e **R** un unsieme di nomi di ruoli allora l'interpretazione di un'asserzione è definita come:

- $(C(a))' = a' \in C'$
- $(r(a,b))' = (a', b') \in r'$

dove C è un concetto (possibilmente composto) e r il nome di un ruolo.

IV.1.6 Teorie ABox

ABox: un ABox è un insieme finito di asserzioni di concetti o asserzioni di ruoli, come un insieme finito di espressioni del tipo :

- C(a) asserzione di concetto
- r(a,b) asserione di ruolo

dove a,b sono nomi individuali, C è un concetto composto e r è il nome di un ruolo.

Modello di un ABox: sia **C** un insieme di nomi di concetti e **R** un insieme di nomi di ruoli.
Un modelloo di un ABox è un'interpretazione che soddisfa ogni asserzione nell'ABox.
L'interpetazione deve:

- I soddisfa un'asserzione di concetto $C(a)$ se $a' \in C'$.
- I soddisfa un'asserzione di ruolo $r(a,b)$ se $(a', b') \in r'$.

Espansione di concetti: sia $\mathcal{T}$ una TBox definita e $\mathcal{A}$ un ABox.
Sia C un concetto definito in $\mathcal{T}$ come $C \equiv C'$ e occorrente in $\mathcal{A}$, sia poi a un individuo occorrente in $\mathcal{A}$ allora $C'(a)$, l'espansione di $C(a)$ rispetto a $\mathcal{T}$, è ottenibile sostituendo C con C'.

Espansione di ABox rispetto ad una TBox: applicando l'espansione a tutti i concetti nella ABox e definiti nella TBox la nuova Abox $\mathcal{A}'$ risultante è chiamata espansione di $\mathcal{A}$ rispetto a $\mathcal{T}$.

Esempio

Date le Tbox e ABox:

$$\mathcal{T} = \begin{cases} Mother \equiv Female \sqcap \exists hasChild.Person \\ Female \equiv Person \sqcap \neg Male \end{cases}$$

$$\mathcal{A} = \Big\{ Mother(Anna)$$

L'estensione di $\mathcal{A}$ rispetto a $\mathcal{T}$ é:

$$\mathcal{T}' = \begin{cases} Mother(Anna) \\ Female(Anna) \\ \exists hasChild.Person(Anna) \\ Person(Anna) \\ \neg Male(Anna) \end{cases}$$

Conoscenze di base: una conoscenza di base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ è una tupla composta da una TBox $\mathcal{T}$ ed una ABox $\mathcal{A}$.

Modello di una base di conoscenza: un interpretazione I è un modello per una conoscenza di base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ se è un modello sia per $\mathcal{T}$ e per $\mathcal{A}$.

Chapter IV.2

Modeling

IV.2.1 Principali risultati teorici

DL è una specie di zucchero sintattico per un sottoinsieme di FOL, questo sottoinsieme è ristretto a termini binari, unari e nessun simbolo di funzione.

Da DL a FOL: sia $T_{DF(X)}$ una funzione di traduzione da DL a FOL, abbiamo quindi le seguenti formule:

- $T_{DF(X)}(C) = C(x)$
- $T_{DF(X)}(C \sqcap D) = T_{DF(X)}(C) \wedge T_{DF(X)}(D)$
- $T_{DF(X)}(C \sqcup D) = T_{DF(X)}(C) \vee T_{DF(X)}(D)$
- $T_{DF(X)}(\exists r.C) = \exists y.r(x,y) \wedge TDF(y)(C)$
- $T_{DF(X)}(\forall r.C) = \forall y.r(x,y) \implies TDF(y)(C)$

Possiamo avere le seguenti traduzioni:

- $T_{DF}(C \sqsubseteq D) = \forall x.(T_{DF}(x)(C) \implies T_{DF(x)}(D))$, per tutte le inclusioni concettuali in $\mathcal{T}$
- $T_{DF}(C(a)) = T_{DF(x)}(C[a/x])$, con $C(a) \in \mathcal{A}$
- $TDF(r(a,b)) = r(a,b)$, con $r(a,b) \in \mathcal{A}$

Sia ora $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ una knowledge base in DL con $T_{DF}(\mathcal{T})$ e $T_{DF}(\mathcal{A})$ rispettivamente gli insiemi di tutte le traduzioni delle inclusioni logiche $\mathcal{T}$ e asserzioni $\mathcal{A}$, abbiamo allora i seguenti teoremi:

- $\mathcal{K}$ è soddisfacibile se e solo se $T_{DF}(\mathcal{T}) \wedge T_{DF}(\mathcal{A})$ è soddisfacibile.
- $C \sqsupseteq_{\mathcal{T}} D$ se e solo se è valida la seguente formula:

$$T_{DF(\mathcal{T})} \implies \forall x.(T_{DF(x)}(C) \implies T_{DF(x)}(D))$$

- a è un'istanza di C rispetto a $\mathcal{T}$ se e solo se la seguente formula è valida:

$$(T_{DF}(\mathcal{T}) \wedge T_{DF}(C)) \implies T_{DF(x)}(C[a/x])$$

Ragionamento in DL: per poter applicare degli algoritmi di ragionamento su delle teorie (TBox e ABox) in DL dobbiamo prima tradurle in FOL e successivamente usare i Tableaux per verificare la validità ecc . . . della formula in FOL che abbiamo.

Esempio

Traduzione TBox dal linguaggio naturale prima in DL poi in FOL

1. A teacher is a person who teaches a course
2. A student is a person who attends a course
3. Students do not teach

Traduzioni in DL

1. $Teacher \equiv Person \sqcap \exists teaches.Course$
2. $Student \equiv Person \sqcap \exists attends.Course$
3. $\exists teaches.\top \sqsubseteq \neg Student$

Traduzioni in FOL

1. $\forall x.(Teacher(x) \equiv Person(x) \wedge \exists y.(teaches(x,y) \wedge Course(y)))$
2. $\forall x.(Student(x) \equiv Person(x) \wedge \exists y.(attends(x,y) \wedge Course(y)))$
3. $\forall x.(\exists y.teaches(x,y) \supset \neg Student(x))$

Semantica del mondo chiuso: è una metodologia di interpretazione tipica dei DB, se un elemento non è presente nel DB si assume sia falso.
Permette quindi un solo modello quello in cui tutto ciò conosciuto è vero e tutto quello non conosciuto è falso.

Semantica del mondo aperto: interpretazione tipica dei KG e delle Abox in cui non possiamo fare assunzioni su ciò che non conosciamo.
Permette quindi molteplici modelli in cui ciò che non conosciamo può essere sia vero che falso.

Esempio

Prendiamo in esame la seguente base di conoscenza $\mathcal{K} = (\mathcal{T}, \mathcal{A})$:

$$\mathcal{T} = \left\{ Mother \equiv Female \sqcap \exists hasChild.Person \right.$$

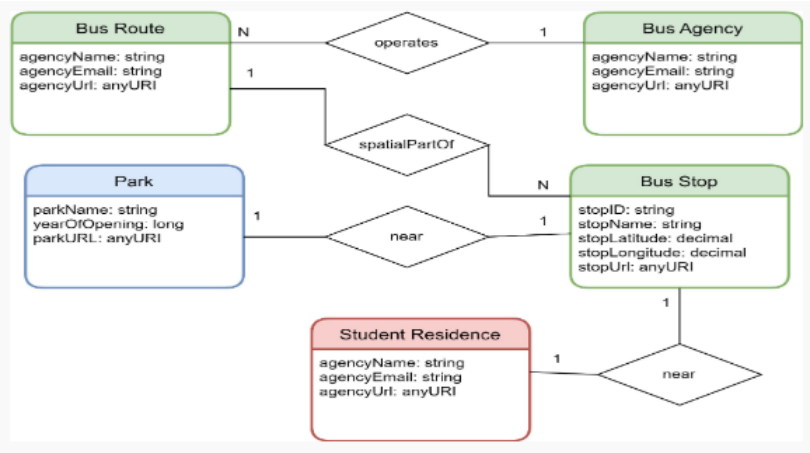
$$\mathcal{A} = \begin{cases} Mother(Anna) \\ hasChild(Anna, Bob) \end{cases}$$

- Per la semantica del mondo chiuso Anna ha solo Bob come figlio.
- Per la semantica del mondo aperto Anna ha almeno Bob come figlio, potrebbe averne più come anche solo Bob.

IV.2.2 Da semi-formale a formale

IV.2.2.1 Modelo ER

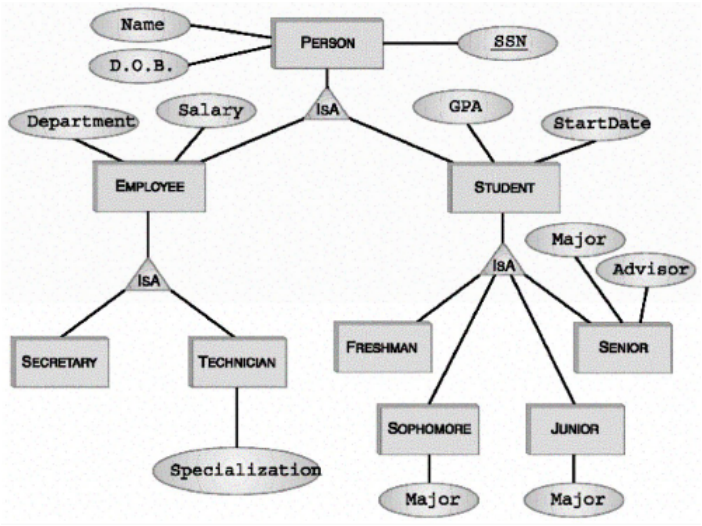
Un ER descrive cose correlate tra di loro nel nostro dominio di conoscenza attraveso classi/entità e relazioni che collegano tra di loro le classi/entità.
L'ER quindi non è altro che un modello astratto che definisce dati o informazioni sulla struttura dei dati.



IV.2.2.2 Modello EER

EER sta per Extended ER questo perchè partiamo da tutti i concetti definiti nei diagrammi ER, ai quali però aggiungiamo i concetti di sotto/super classe con la relazione "**is-a**".
Una super classe è un'entità/classe che può essere divisa in più sotto classi e una sotto classe eredita le proprietà della sua super classe.
Si possono adottare due metodi per creare super/sotto classi.

1. **Generalizzazione:** avendo un gruppo di sotto classi si cercano gli attributi che hanno in comune e con quelli si crea una super classe.
2. **Specializzazione:** con una super classe classe si cerca di identificare un sottoinsieme di entità che hanno alcune caratteristiche diverse.



Vediamo ora cosa vuol dire ontologia, questo termine ha significati differenti in base all'ambito in cui ci si trova.

Ontologia nella filosofia: la disciplina che si occupa di dare un significato alla natura e alla struttura della realtà.

Ontologia in informatica: è un tipo speciale di oggetto contenente informazioni oppure di artifatto prodotto dalla computazione.

- Modella formalmente la struttura di un sistema come le entità rilevanti e le relazioni che vengono scoperte.

Chapter IV.3

Ragionamento

IV.3.1 Problemi di ragionamento con le TBox

I principali problemi quando si opera con le TBox sono 4:

1. **Soddisfacibilità:** un concetto C è soddisfacibile rispetto a $\mathcal{T}$ se esiste un modello I di $\mathcal{T}$ tale che C_I sia non vuoto, in questo caso possiamo anche dire che I è un modello per C.

2. **Sussunzione:** un concetto C è sussunto da un concetto D rispetto a $\mathcal{T}$ se $C' \subseteq D'$ per ogni modello I di $\mathcal{T}$, in questo caso possiamo scrivere $C \sqsubseteq_{\mathcal{T}} D$ oppure $T \models C \sqsubseteq D$.

3. **Equivalenza:** due concetti C e D sono equivalenti rispetto a $\mathcal{T}$ se $C' = D'$ per ogni modello I di $\mathcal{T}$, in questo caso possiamo scrivere $C \equiv_{\mathcal{T}} D$ oppure $\mathcal{T} \models C \equiv D$.

4. **Disgiunzione:** due concetti C e D sono disgiunti rispetto a $\mathcal{T}$ se $C' \cap D' = \emptyset$ per ogni modello I di $\mathcal{T}$, in questo caso scriveremo $C \bot D$.

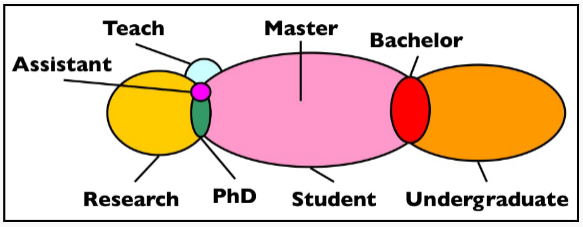
E' possibile notare che se due concetti non si sussumono a vicenda e non sono disgiunti vuol dire che, a livello insiemistico, sono parzialmente sovvrapposti.

Consistenza di concetti: un concetto C è consistente con riferimento a $\mathcal{T}$ se esiste qualche interpretazione che soddisfa l'assioma $\mathcal{T} \cup C$ tale che C sia un insieme non vuoto in quell'interpretazione, un concetto non vuoto viene detto soddisfacibile, insiddisfacibile altrimenti.

Esempio

Consideriamo la TBox:

$$\mathcal{T} = \begin{cases} Undergraduate \sqsubseteq \neg Teach \\ Bachelor \equiv Student \sqcap Undergraduate \\ Master \equiv Student \sqcap \neg Undergraduate \\ PhD \equiv Master \sqcap Research \\ Assistant \equiv PhD \sqcap Teach \end{cases}$$



Proviamo a vedere se il seguente fatto è consistente/soddisfacibile:

$$\mathcal{T} \models Bachelor \sqcap PhD$$

Per provare che è falsa iniziamo a sostituire Bachelor e Phd con le definizioni che ho nella TBox fino a che non trovo un insieme vuoto.

$$\begin{aligned}
&Bachelor \sqcap PhD \\
&\equiv (Student \sqcap Undergraduate) \sqcap (Master \sqcap Research) \\
&\equiv (Student \sqcap Undergraduate) \sqcap ((Student \sqcap \neg Undergraduate) \sqcap Research) \\
&\equiv Student \sqcap \mathbf{Undergraduate} \sqcap \neg\mathbf{Undergraduate} \sqcap Research \\
&\equiv Student \sqcap \bot \sqcap Research
\end{aligned}$$

Sussunzione di concetti: un concetto C è sussunto da un concetto D se in ogni modello di $\mathcal{T}$ l'insieme denotato da C è sottoinsieme dell'insieme denotato da D.

Equivalenza di concetti: è spesso definita usando la sussunzione dei concetti:

$$\mathcal{T} \models C \equiv D \text{ se e solo se } \mathcal{T} \models C \sqsubseteq D \text{ e } \mathcal{T} \models D \sqsubseteq D$$

Esempio

Prendiamo la TBox dell'esempio precedente e proviamo a verificare se $PhD \sqsubseteq Student$ è soddisfacibile.
Formaliziamo il problema in:

$$\mathcal{T} \models PhD \sqsubseteq Student$$

Posso quindi ora espandere PhD e verificare che sia sempre sottoinsieme di Student.

$$\begin{aligned}
&PhD \\
&\equiv Master \sqcap Research \\
&\equiv (Student \sqcap \neg Undergraduate) \sqcap Research \\
&\sqsubseteq Student
\end{aligned}$$

Verifichiamo ora se $Student \equiv Bachelor \sqcap Master$ è consistente con $\mathcal{T}$, formaliziamolo in:

$$\mathcal{T} \models Student \equiv Bachelor \sqcup Master$$

Ora espandiamo il concetto $Bachelor \sqcup Master$ e verifivhiamo che ritorni $Student$.

$$\begin{aligned}
&Bachelor \sqcup Master \\
&\equiv (Student \sqcap Undergraduate) \sqcup (Student \sqcap \neg Undergraduate) \\
&\equiv Student \sqcup (Undergraduate \sqcap \neg Undergraduate) \\
&\equiv Student \sqcup \top \\
&\equiv Student
\end{aligned}$$

Disgiunzioni di concetti: viene usata per verificare se due concetti hanno elementi in comune, formalmente:

$$\mathcal{T} \models C \bot D$$

Però possiamo tenere a mente la definizione di disgiunzione.

$$C \bot D \equiv C \sqcap D \sqsubseteq \bot$$

Esempio

Consideriamo la stassa TBox di prima e verifichiamo se $Undergraduate \sqcap Assistant \sqsubseteq \bot$ è consistente con $\mathcal{T}$, formalizziamo il problema in:

$$\mathcal{T} \models Undergraduate \sqcap Assistant \sqsubseteq \bot$$

Ora espandiamo la prima parte e verifichiamo che se è effettivamente $\sqsubseteq \bot$.

$$\begin{aligned} & Undergraduate \sqcap Assistant \\ & \sqsubseteq \neg Teach \sqcap Assistant \\ & \equiv \neg Teach \sqcap (PhD \sqcap Teach) \\ & \equiv \bot \sqcap PhD \\ & \equiv \bot \end{aligned}$$

Riduzione dei problemi di ragionamento: per due concetti C e D i problemi di sussunzione, equivalenza e disgiunzione possono essere visti come problemi di soddisfacibilità.

- **Sussunzione:** C è sussunto da D se e solo se $C \sqcap \neg D$ è insoddisfacibile.
- **Equivalenza:** C e D sono equivalenti se e solo se $(C \sqcap \neg D)$ e $(D \sqcap \neg C)$ sono entrambe insoddisfacibili.
- **Disgiunzione:** C e D sono disgiunti se e solo se $C \sqcap D$ è insoddisfacibile.

IV.3.2 Problemi di ragionamento con le ABox

I principali problemi di ragionamento con un ABox $\mathcal{A}$ possono essere 4.

1. **Consistenza:** un Abox $\mathcal{A}$ è consistente rispetto ad una TBox $\mathcal{T}$ se se esiste un'interpretazione I che è modello sia di $\mathcal{A}$ si di $\mathcal{T}$.
2. **Controllo di istanza:** controllare ogniqualvolta un associazione $C(a)$ è conseguita da una ABox.
3. **Recupero d'istanze:** dato un concetto C, recuperare tutte le istanze che soddisfano C.
4. **Realizzazione di concetto:** dato un insieme di concetti e un individuo a trovare l'insieme di concetti più specifici C tale che $\mathcal{A} \models C(a)$.

Controllo di consistenza: una ABox $\mathcal{A}$ è consistente rispetto ad una TBox $\mathcal{T}$ se esiste un'interpretazione che è modello di entrambe.

Esempio

Data la Tbox:

$$\mathcal{T} = \begin{cases} Parent \equiv Mother \sqcup Father \\ Father \equiv Male \sqcap hasChild \\ Mother \equiv Female \sqcap hasChild \\ Male \equiv Person \sqcap \neg Female \end{cases}$$

E la Abox:

$$\mathcal{A} = \begin{cases} Mother(Mary) \\ Father(Mary) \end{cases}$$

Possiamo dire dall'espansione di $\mathcal{T}$ che $Mary$ non può essere sia padre che madre, quindi $\mathcal{A}$ è inconsistente.

Controllo di istanza: una procedura per verificare se l'individuo a appartiene a un certo concetto C o ad un certo ruolo r, in formula $\mathcal{A} \models C(a)$ se ogni interpretazione che soddisfa $\mathcal{A}$ soddisfa anche $C(s)$.
Possiamo vedere questo compito di ragionamento come un problema di verifica di consistenza:

$$\mathcal{A} \models C(a) \equiv \mathcal{A} \cup \{\neg C(a)\} \text{ è inconsistente}$$

Se dimostriamo che $\mathcal{A} \cup \{\neg C(a)\}$ è inconsistente allora l'individuo a non appartiene a C.

Esempio

Prendiamo le seguenti Tbox e Abox:

$$\mathcal{T} = \begin{cases} Undergraduate \sqsubseteq \neg Teach \\ Bachelor \equiv Student \sqcap Undergraduate \\ Master \equiv Student \sqcap \neg Undergraduate \\ PhD \equiv Master \sqcap Research \\ Assistant \equiv PhD \sqcap Teach \end{cases}$$

$$\mathcal{A} = \begin{cases} Master(Chen) \\ PhD(Enzo) \\ Assistant(Rui) \end{cases}$$

Ora dobbiamo verificare se $PhD(Rui)$ è soddisfacibile, formalizziamo il problema in:

$$\mathcal{A} \models PhD(Rui)$$

Possiamo ora espandere la TBox con i dati della ABox:

$$Assistant(Rui) \equiv PhD(Rui) \sqcap Teach(Rui)$$

Abbiamo dimostrato che $\mathcal{A} \cup \{\neg PhD(Rui)$è insoddisfacibile$\}$ quindi $PhD(Rui)$ è un'istanza corretta.

Recupero delle istanze: una procedura per recuperare tutte le istanze di uno specifico concetto C.

Realizzazione dei concetti: è la procedura per trovare il concetto C più specifico che soddisfa un'istanza.

Esempio

Facciamo riferimento alla TBox e alla ABox precedenti.

Troviamo tutte le istanze di $\neg Undergraduate\{Chen, Enzo, Rui\}$

$$Master(Chen) \equiv Student(Chen) \sqcap \neg \textbf{Undergraduate(Chen)}$$

$$Phd(Enzo) \equiv Master(Enzo) \sqcap Research(Enzo) \equiv Student(Enzo) \sqcap \neg \textbf{Undergraduate(Enzo)} \sqcap Reserach(Enzo)$$

$$Assistant(Rui) \equiv PhD(Rui) \sqcap Teach(Rui) \equiv Master(Rui) \sqcap Research(Rui) \sqcap Teach(Rui) \equiv Student(Rui) \sqcap \neg \textbf{Undergraduate(Rui)} \sqcap \vdots$$

Adesso data l'istanza Rui e i concetti $\{Student, PhD, Assistant\}$ trova il concetto più specifico possibile che soddisfa Rui, quindi tale che $\mathcal{A} \models C(Rui)$.

Esaminando la TBox vediamo che per essere $Assistant$ dobbiamo essere PhD e per essere PhD dobbiamo essere $Master$ e quindi $Student$, troviamo quindi la relazione:

$$Assistant \sqsubseteq PhD \sqsubseteq Student$$

Quindi il concetto più ristretto a cui appartiene Rui è $Assistant$.

Riduzione del ragionamento: vediamo ora come alcuni problemi di ragionamento possono essere visti come altri tipi di problemi di ragionamento:

- Il recupero di istanze e la realizzazione di concetti possono essere implementate con il controllo di istanze.
- Il controllo di istanze può essere ridotto ad un problema di consistenza.