# Appunti di Logica

*Sul corso di Logica Matematica @ DISI,*
*Università degli Studi di Trento*

A.A. 2018/2019

`github.com/gik98/unitn18-logics`

— 1 —

# PL - Introduzione

La logica delle proposizioni (Propositional Logic -**PL**) è una logica che permette di rappresentare fatti (affermazioni), che possono essere vere o false.

PL si compone di **simboli logici** e **variabili proposizionali**. Una proposizione (formula) è composta quindi di variabili proposizionali uniti da simboli logici. La formula può essere *vera* o *falsa* a seconda dell'assegnazione delle singole variabili.

> **Definizione 1 (Linguaggio della PL)**
> ***Logical symbols:*** *(1)* $\neg$ *(2)* $\wedge$ *(3)* $\vee$ *(4)* $\supset$ *(5)* $\equiv$
> ***PL formulas and sub-formulas***
> - *every logical variable* $P \in \mathrm{P}$ *is an atomic formula*
> - *every atomic formula is a formula*
> - *if* $A$ *and* $B$ *are formulas, then* $\neg A, A \wedge B, A \vee B, A \supset B, A \equiv B$ *are formulas*

Una **funzione di interpretazione** $I : P \rightarrow \{\top, \bot\}$ assegna un valore vero o falso a ciascuna variabile $P \in \mathrm{P}$.

Una funzione di interpretazione è detta **modello** di una funzione $\varphi$ se le sue assegnazioni rendono il valore della funzione vero. In simboli: $I \models \varphi$.

## SAT, UNSAT, VAL

- Una formula A è soddisfacibile (**SAT**) se $\exists I$ funzione di interpretazione t.c. $I \models$ A.
- Una formula A è insoddisfacibile (**UNSAT**) se $\nexists I$ funzione di interpretazione t.c. $I \models$ A.
- Una formula A è valida (**VALID**) se $\forall I, I \models$ A

**Osservazione:**
Se A è **VALID**, $\neg$A è **UNSAT**.
Se A è **SAT**, $\neg$A non è valida.
Se A non è valida, $\neg$A è **SAT**.
Se A è **UNSAT**, $\neg$A è **VALID**.

### Conseguenza e equivalenza logica

- Una formula A è una **conseguenza logica** di un insieme di formule $\Gamma$, in simboli $\Gamma \models$ A sse per ogni funzione di interpretazione $I$ che soddisfa tutte le formule di $\Gamma$, $I$ soddisfa A.
- Due formule A, B sono **equivalenti**, in simboli A $\equiv$ B sse per ogni funzione di interpretazione $I$, $I(\mathrm{A}) = I(\mathrm{B})$.

### Procedure di decisione

**Model checking ($I$, $\varphi$):** $I \overset{?}{\models} \varphi$ ($I$ soddisfa $\varphi$?)

**Satisfiability ($\varphi$):** $\overset{?}{\exists} I | I \models \varphi$ (Esiste un modello che soddisfi $\varphi$?)

**Validity ($\varphi$):** $\overset{?}{\models} \varphi$ ($\varphi$ è soddisfatta da qualsiasi modello?)

**Logical consequence ($\Gamma$, $\varphi$):** $\Gamma \overset{?}{\models} \varphi$ (Ogni modello che soddisfa $\Gamma$ soddisfa anche $\varphi$?)

### Proprietà della conseguenza logica

Siano $\Gamma$ e $\Sigma$ due insiemi di formule proposizionali; $A$, $B$ due formule proposizionali, allora valgono le seguenti proprietà:

**Reflexivity:** $\{A\} \models A$

**Monotonicity:** Se $\Gamma \models A$ allora $\Gamma \cup \Sigma \models A$

**Cut:** Se $\Gamma \models A$ and $\Sigma \cup \{A\} \models B$ allora $\Gamma \cup \Sigma \models B$

**Compactness:** Se $\Gamma \models A$, allora esiste un sottoinsieme finito $\Gamma_0 \subseteq \Gamma$ tale che $\Gamma_0 \models A$

**Deduction theorem** Se $\Gamma, A \models B$ allora $\Gamma \models A \rightarrow B$

**Refutation principle:** $\Gamma \models A$ se e solo se $\Gamma \cup \{\neg A\}$ è insoddisfacibile

### Formalizzazione del linguaggio naturale

- $A$: "It is the case that A"
- $\neg A$: "It is not the case that A"
- $A \wedge B$: "A and B", "A but B", "Although A, B", "Both A and B"
- $A \vee B$: "A or B", "Either A or B"
- $A \rightarrow B$: "If A, then B", "B if A", "B only if A"
- $\neg(A \vee B)$: "Neither A nor B"
- $\neg(A \wedge B)$: "It is not the case that both A and B"

### Assiomatizzazione di Hilbert della logica proposizionale

Il sistema deduttivo di Hilbert propone quattro assiomi fondamentali per descrivere la logica proposizionale. I primi tre caratterizzano le solite proprietà degli operatori proposizionali, la quarta (**Modus ponens**) è una regola per la deduzione logica.

1. $\varphi \supset (\psi \supset \varphi)$
2. $(\varphi \supset (\psi \supset \vartheta)) \supset ((\varphi \supset \psi) \supset (\varphi \supset \vartheta)$
3. $(\neg\psi \supset \neg\varphi) \supset ((\neg\psi \supset \varphi) \supset \psi)$
4. Sia $\varphi$ vero, allora $(\varphi \supset \psi) \supset \psi$

Gli assiomi sono espressi utilizzando solo gli operatori $\supset$ e $\neg$. Si osservi che è possibile definire $\wedge$ e $\vee$ in termini di questi, come segue:

- $\alpha \supset \beta \supset \alpha \wedge \beta$
- $(\alpha \supset \gamma) \supset (\beta \supset \gamma) \supset \alpha \vee \beta$

— 2 —

# PL - CNF & DPLL

**Definizione 2 (Conjunctive Normal Form)**
- *A **literal** is a propositional variable A or the negation of a propositional variable* $\neg A$
- *A **clause** is a disjunction of literals* $\bigvee_{j=1}^{m} A_j$
- *A **formula** is in **CNF** if it is a conjunction of clauses* $\bigwedge_{i=1}^{n} (\bigvee_{j=1}^{m} I_{ij})$

**Proposizioni:**

1. Ogni PL formula può essere ridotta in CNF
2. $\models \text{CNF}(\phi) \equiv \phi$ (Ogni Pl formula è equivalente alla sua riduzione in CNF)

### Notazione insiemistica

Una formula in CNF può essere rappresentata come un insieme di *clauses*, o un insieme di insiemi di *literals*. Le operazioni sono implicite. La generica formula CNF $\bigwedge_{i=1}^{n} (\bigvee_{j=1}^{m} I_{ij})$ può essere rappresentata così: $\{\{I_{1,1}, ..., I_{1,m_1}\}, ..., \{I_{n,1}, ..., I_{n,m_n}\}\}$.

## Riduzione in CNF

- $\text{CNF}(p) = p$ if $p$ is a literal
- $\text{CNF}(\neg p)$ if $\neg p$ is a literal
- $\text{CNF}(\neg\neg p) = \text{CNF}(p)$

- $\text{CNF}(\phi \to \psi) = \text{CNF}(\neg\phi) \oplus \text{CNF}(\psi)$
- $\text{CNF}(\phi \wedge \psi) = \text{CNF}(\phi) \wedge \text{CNF}(\psi)$
- $\text{CNF}(\phi \vee \psi) = \text{CNF}(\phi) \oplus \text{CNF}(\psi)$
- $\text{CNF}(\phi \equiv \psi) = \text{CNF}(\phi \to \psi) \wedge \text{CNF}(\psi \to \phi)$

- $\text{CNF}(\neg(\phi \to \psi)) = \text{CNF}(\phi) \wedge \text{CNF}(\neg\psi)$

- $\mathrm{CNF}(\neg(\phi \wedge \psi)) = \mathrm{CNF}(\neg\phi) \otimes \mathrm{CNF}(\neg\psi)$
- $\mathrm{CNF}(\neg(\phi \vee \psi)) = \mathrm{CNF}(\neg\phi) \wedge \mathrm{CNF}(\neg\psi)$
- $\mathrm{CNF}(\neg(\phi \equiv \psi)) = \mathrm{CNF}(\phi \wedge \neg\psi) \otimes \mathrm{CNF}(\psi \wedge \neg\phi)$

Dove $\otimes$ è definito come segue:

$$(C_1, ..., C_n) \otimes (D_1, ..., D_n) := (C_1 \vee D_1) \wedge ... \wedge (C_1 \vee D_n) \wedge ... \wedge (C_n \vee D_1) \wedge ... \wedge (C_n \vee D_n)$$

## DPLL

**DPLL** è un algoritmo per calcolare la soddisfacibilità di una PL formula ridotta in **CNF**.

**Definizione 3 (Partial evaluation)**
***Osservazioni:***
*Sia $F := C_0, ..., C_n = CNF(\varphi)$, $I$ funzione di interpretazione. Vale quanto segue:*

1. *$I \models \varphi \iff I \models C_i \forall i = 0, ..., n$ ($\varphi$ è soddisfatta se tutte le sue clauses sono soddisfatte)*
2. *$I \models C_i \iff \exists l \in C_i : I \models l$ (Una clause è soddisfatta se almeno uno dei literals che la compongono è soddisfatto)*

***Proposizione:*** *per verificare se $I$ soddisfa $F$ non è necessario conoscere le assegnazioni di ogni literal che compare in $F$.*

**Definizione 4 (Unit propagation)**
*Sia $\varphi$ una PL formula, $I$ funzione di interpretazione; sia $u$ una unit clause $\{u\} \in \varphi$ (clause composta di un solo literal). Vale: $I \models \varphi \rightarrow I : u \mapsto \top$. (segue dalla proprietà (2) sopra esposta: una unit clause non può essere soddisfatta se la sua unica componente non è valutata vera).*

L'algoritmo **DPLL** calcola un possibile modello che soddisfa la PL formula $\varphi$, se esiste. La costruzione del modello $I$ avviene partendo da un insieme vuoto di assegnazioni, via via aumentato.

Ad ogni passo dell'algoritmo le *clauses* di $\varphi$ possono essere in uno dei seguenti stati rispertto al modello parziale $I'$, che va via via costruendosi:

1. una *clause* $c \in \varphi$ è **vera** se $\exists l \in c | I : l \mapsto \top$ (se il modello parziale assegna il valore di verità ad uno dei *literals* che la compongono)
2. una *clause* $c \in \varphi$ è **falsa** se $\forall l \in c | I : l \mapsto \bot$
3. una *clause* $c \in \varphi$ è **indecidibile** se non è né vera, né falsa

Ad ogni passo l'algoritmo effettua un'operazione di assegnazione ad un *literal* di una *clause* ancora in uno stato indecidibile. Data la formula $\varphi$ e un literal $p$, indichiamo con $\varphi_{|p}$ la formula ottenuta sostituendo ad ogni occorrenza di $p$ il valore di verità $\top$, analogamente $\varphi_{|\neg p}$ la formula ottenuta sostiutendo $\bot$ a $p$. Dalle osservazioni sulla *partial evaluation* segue:

- tutte le *clauses* contenenti almeno un *literal* valutato $\top$ possono ora essere rimosse
- tutte le occorrenze di *literal* valutati $\bot$ possono essere rimosse

L'algoritmo termina appena $\varphi$ contiene una *clause* alla quale sono stati rimossi tutti i *literals* (detta **empty clause**), in tal caso la formula è **insoddisfacibile**; oppure quando $\varphi$ non contiene più *clauses*, in tal caso la formula è **soddisfacibile** e $I' = I$.

**Data:** $\varphi$ PL formula ridotta in CNF, $I'$ modello parziale
**Result:** **SAT** se soddisfacibile, **UNSAT** altrimenti

**DPLL**($\varphi$, $I'$):
UnitPropagation($\varphi$, $I'$)
**if** $\{\} \in \varphi$ **then**
  **return** *UNSAT*
**if** $\varphi = \{\}$ **then**
  **return** *(SAT, $I'$)*
$l \leftarrow C \in \varphi$
DPLL($\varphi_{|l}$, $I' \cup \{I'(l) = \top\}$)
DPLL($\varphi_{|l}$, $I' \cup \{I'(l) = \bot\}$)

— 3 —

# PL - Tableaux

## Regole di riduzione

$\alpha$ rules

$$\frac{\phi \land \psi}{\begin{array}{c}\phi\\ \psi\end{array}} \qquad \frac{\neg(\phi \lor \psi)}{\begin{array}{c}\neg\phi\\ \neg\psi\end{array}} \qquad \frac{\neg(\phi \supset \psi)}{\begin{array}{c}\phi\\ \neg\psi\end{array}}$$

$\neg\neg$ elimination

$$\frac{\neg\neg\phi}{\phi}$$

$\beta$ rules

$$\frac{\phi \lor \psi}{\begin{array}{c|c}\phi & \psi\end{array}} \qquad \frac{\neg(\phi \land \psi)}{\begin{array}{c|c}\neg\phi & \neg\psi\end{array}} \qquad \frac{\phi \supset \psi}{\begin{array}{c|c}\neg\phi & \psi\end{array}}$$

Branch closure

$$\frac{\begin{array}{c}\phi\\ \neg\phi\end{array}}{\text{X}}$$

L'equivalenza può essere riscritta come doppia implicazione.

$$\phi \equiv \psi \iff (\phi \supset \psi) \land (\psi \supset \phi)$$

---

**Osservazione:** le $\alpha$- e $\beta$ rules del tableaux sono analoghe a quelle di riduzione in *CNF*:

- una $\alpha$ rule è equivalente a and logico $\land$ delle formule da ridurre;
- una $\beta$ rule è equivalente a or logico (nella forma $\otimes$) fra tutte le formule da ridurre, prese a due a due.

## Metodo del tableaux

Il **tableaux** è un metodo per provare se un insieme di formule dato è **insoddisfacibile**. Di conseguenza, è possibile dimostrare anche la **validità** dell'insieme di formule (dimostrando l'insoddisfacibilità della negazione dell'insieme di formule).

Il **tableaux** costruisce un albero binario, la cui radice è la congiunzione dell'insieme di formule di cui si vuole verificare l'insoddisfacibilità. Nuove foglie sono aggiunte applicando $\alpha$ rules (*deterministic rules*) o $\beta$ rules (*branch splitting*)a una qualsiasi formula che appare in un nodo *ancestor*.

Un ramo dell'albero è **chiuso** se il cammino fra la foglia e la radice contiene formule contradditorie (es. $p$, $\neg p$). Se tutti i rami possono essere chiusi, allora la formula di partenza è insoddisfacibile.

**Osservazione:** è conveniente applicare $\alpha$ rules anziché $\beta$ rules, laddove possibile, in modo da non aumentare il numero di rami dell'albero.

### Interpretazione dal tableaux

Si può dimostrare che un tableaux in PL termina sempre (dopo un numero finito di passi tutti i rami sono chiusi oppure tutte le formule che compaiono nel tableaux sono state valutate). Pertanto, se una formula genera un tableaux che non si chiude, la formula è **soddisfacibile**.

I modelli che rendono il tableaux soddisfacibile possono essere ricavati dai rami rimasti aperti. Per ogni ramo e per ogni variabile proposizionale $p$, vale $I(p) = \top$ se nel cammino dalla foglia alla radice compare $p$; $I(p) = \bot$ se nel cammino dalla foglia alla radice compare $\neg p$. Se né $p$ né $\neg p$ compaiono, $I(p)$ può essere definito arbitrariamente (entrambe le definizioni renderanno la formula soddisfacibile).

— 4 —

# FOL - Introduzione

La logica del primo ordine (First-Order Logic - **FOL**) è un'estensione della propositional logic.

Mentre la PL prevede solamente valori di verità o falsità, FOL prevede *variabili* che rappresentano oggetti del mondo da descrivere, inoltre, questi oggetti possono essere quantificati (si possono descrivere *tutti* gli oggetti o *alcuni* oggetti, senza nominare ciascuno esplicitamente, come sarebbe necessario in PL).

**Definizione 5 (Sintassi della FOL)**
***Logical symbols:*** *Comprende gli stessi simboli della PL, e in più:*

1. *quantificatori* ($\forall$, $\exists$)
2. *variabili* $x_1, x_2, ...$
3. *simbolo di uguaglianza (opzionale)* $=$

***Non-logical symbols:***

- *costanti* $c_1, c_2, ...$
- *funzioni* $f_1, f_2, ...$ *alle quali è associata una arità*
- *relazioni* $P_1, P_2, ...$ *alle quali è associata una arità*

## Terms e formule

Un **term** è l'elemento sintattico che rappresenta un oggetto del mondo.

Ci sono tre possibili tipologie di **term**:

1. **costanti**: descrivono sempre uno specifico oggetto (p. es. "Mario", "Giappone");
2. **variabili**: possono descrivere un qualsiasi oggetto, oppure essere associate ad un quantificatore;
3. **funzioni**: un simbolo applicato a zero, uno o più *terms* - il numero di *terms* è definito dalla *arità* del simbolo funzionale (oss: una funzione con *arità* 0 è equivalente ad una costante).

Un **predicate** o **relation** costituisce una "frase" in FOL. Un simbolo relazionale è applicato a zero, uno o più *terms* - il numero di *terms* è definito dalla *arità* del simbolo relazionale. I **predicate** rappresentano delle relazioni fra oggetti del mondo.

Il simbolo di uguaglianza = può essere visto come una relazione con arità uguale a 2.

Una **formula** è definita come segue.

1. Siano $t_1, ..., t_n$ *terms*, $P$ relazione di arità $n$, allora $P(t_1, ..., t_n)$ è una formula. Vale anche: $t_1, t_2$ *terms*, $t_1 = t_2$ è una formula.
2. Siano $A, B$ formule, allora $A \wedge B$, $A \vee B$, $A \supset B$, $A \equiv B$, $\neg A$ sono formule.
3. Sia $A$ una formula, $x$ una variabile, allora sono formule $\forall x.A$ e $\exists x.A$.

### Interpretazione in FOL

In PL, un'interpretazione consiste nell'associazione di variabili proposizionali al valore vero o falso. In FOL l'interpretazione è definita in modo più complesso; è composta da:

1. un dominio di interpretazione $\Delta$. Questo contiene tutti gli oggetti che vogliamo descrivere
2. una funzione di interpretazione $I$ che mappa i simboli non logici in elementi del dominio:
   - $I(c_i) \in \Delta$ (mappa le costanti in elementi del dominio)
   - $I(P_i) \subset \Delta^n$ (mappa le relazioni di arità n in n-tuple)
   - $I(f_i) : \Delta^n \to \Delta$ (mappa le funzioni di arità n in elementi del dominio)

**Osservazione** (Relazioni e funzioni): esattamente come definite in algebra, le funzioni possono essere viste come una specializzazione delle relazioni. In altre parole, ogni funzione di arità $n$ può essere equivalentemente rappresentata come una relazione di arità $n + 1$. Esempio: $Mary$ è madre di $Joe$, $Jill$ e $Bill$. È possibile definire:

- $motherOf$ come una relazione di $\Delta^2$:

$$motherOf := \{\langle Joe, Mary\rangle, \langle Jill, Mary\rangle, \langle Bill, Mary\rangle\}$$

- $motherOf$ come una funzione $\Delta \to \Delta$:

$$motherOf(Joe) = Mary, \quad motherOf(Jill) = Mary, \quad motherOf(Bill) = Mary$$

- $brotherOf$ come una relazione di $\Delta^2$:

$$brotherOf := \{\langle Joe, Jill\rangle, \langle Jill, Joe\rangle, \langle Joe, Bill\rangle, \langle Bill, Joe\rangle, \langle Bill, Jill\rangle, \langle Jill, Bill\rangle\}$$

- ...ma non è possibile definire brotherOf come una funzione $\Delta \rightarrow \Delta$: $brotherOf(Jill) =?$

Funzioni e predicati, inoltre, sono elementi sintattici diversi: una funzione è semplicemente un *term*, un predicato è a sé già una funzione. Ad esempio, per formalizzare la frase "la madre di Joe è simpatica", hanno lo stesso significato:
- $Nice(motherOf(Jill))$ - $motherOf$ come funzione
- $\exists x.MotherOf(Joe, x) \wedge Nice(x)$ - $MotherOf$ come predicato

### Assignments

Formalmente, si definisce un'**assignment** $a[x/d]$ come una funzione ($a$) che mappa una variabile ($x$) in un elemento del dominio di interpretazione $d \in \Delta$. La funzione è interpretata come segue:
- se è applicata ad una costante, non ha alcun effetto:
$$I(c)[a[x/d]] = c$$
- se è applicata ad una variabile, avviene la sostituzione
$$I(x)[a[x/d]] = d$$
- se è applicata ad una funzione, l'assignment viene applicato ricorsivamente ai parametri della funzione
$$I(f(t_1, ..., t_n))[a[x/d]] = I(f)(I(t_1)[a[x/d]], ..., I(t_n)[a[x/d]])$$

### Free variables

Una **occorrenza libera** (*free occourence*) di una variabile $x$ in una formula $\varphi$ è un'occorrenza di $x$ che non è legata ad un quantificatore $(\forall, \exists)$.

Una **variabile** $x$ è **libera** in $\varphi$ se esiste almeno una occorrenza di $x$ in $\varphi$ che è libera.

Una **formula** $\varphi$ è **ground** se non contiene nessuna variabile.

Una formula $\varphi$ è **closed** se non contiene alcuna variabile libera.
**Esempio:** $\varphi := P(x) \supset \forall x.Q(x)$; $x$ è una variabile libera, infatti la prima occorrenza di $x$ è libera (la seconda non lo è).

Un *term* $t$ è libero per una certa variabile $x$ in una formula $\varphi$ se tutte le occorrenze di $x$ in $\varphi$ non sono nello scope di un quantificatore di una variabile che ha occorrenze in $t$. In altre parole: $t$ è libero per $x$ in $\varphi$ se si può sostituire $t$ al posto di $x$ senza che la formula cambi significato.
**Esempio:** $\varphi := \exists x.brotherOf(x, y)$; $t := z$; $u := x$. Il term $t$ è libero per $y$: $\varphi' := \exists x.brotherOf(x, z)$ ha lo stesso significato di $\varphi$. Il term $u$ non è però libero per $y$: $\varphi'' := \exists x.brotherOf(x, x)$ ha un significato diverso da $\varphi$.

### Procedure di decisione in FOL

Innanzitutto è necessario definire quando una interpretazione è modello di una formula in FOL. Si osservi che, in presenza di variabili libere, il significato della variabile non è definito finché non viene effettuato un *assignment*. Scelte diverse degli *assignment* possono determinare se un'interpretazione è o non è modello di una formula $\varphi$.

Un'interpretazione $I$ soddisfa (è un **modello** per) una formula $\varphi$ rispetto ad un *assignment* $a[x/d]$ secondo le seguenti regole:

1. ("Caso base") Se la formula è una relazione $P$ di arità $n$, allora

$$I \models P(t_1, ..., t_n)[a] \iff \langle I(t_1)[a], ..., I(t_n)[a]\rangle \in I(P)$$

   Ovvero: affiché $\varphi$ sia soddisfatta deve esistere nell'interpretazione della relazione $P$ una relazione che contenga i *terms* $t_1, ..., t_n$ a cui è stata applicata l'associazione $a$.
   Lo stesso è valido per la relazione di uguaglianza:

$$I \models (t_1 = t_2)[a] \iff I(t_1)[a] = I(t_2)[a]$$

2. (Formule composte) Siano $\varphi$, $\psi$ formule, allora (al solito):
   - $I \models (\neg\varphi)[a] \iff I \not\models \varphi[a]$
   - $I \models (\varphi \wedge \psi)[a] \iff (I \models \varphi[a]) \wedge (I \models \psi[a])$
   - $I \models (\varphi \vee \psi)[a] \iff (I \models \varphi[a]) \vee (I \models \psi[a])$
   - $I \models (\varphi \rightarrow \psi)[a] \iff (I \not\models \varphi[a]) \vee (I \models \psi[a])$
   - $I \models (\varphi \equiv \psi)[a] \iff (I \models \varphi[a]) \equiv (I \models \psi[a])$

3. (Quantificatori) Sia $\varphi$ una formula, allora:
   - $I \models (\exists q.\varphi)[a] \iff \exists z \in \Delta | I \models (\varphi[q/z])[a]$
   - $I \models (\forall q.\varphi)[a] \iff \forall z \in \Delta | I \models (\varphi[q/z])[a]$

Una formula $\varphi$ è **soddisfacibile** se esiste una *interpretazione* $I$ e un *assignment* $a$ tali per cui $I \models \varphi[a]$.

Una formula $\varphi$ è **insoddisfacibile** se non è soddisfacibile.

Una formula $\varphi$ è **valida** se per ogni *interpretazione* $I$ e per ogni *assignment* $a$ vale $I \models \varphi[a]$.

**Osservazione**: Se una formula è **chiusa**, la sua validità o (in)soddisfacibilità non dipende dall'assegnazione $a$ (l'assegnazione non ha alcun effetto sulla formula perché la formula non ha variabili libere sulle quali effettuare l'assegnazione).

### Analogia con i DB

Un database relazionale può essere rappresentato in First-Order Logic; l'equivalenza è così descritta:

- le **relazioni** in FOL corrispondono alle **tabelle** del database;

- il dominio di interpretazione $\Delta$ in FOL corrisponde all'insieme di tutti i valori che compaiono nei dati contenuti nelle tabelle;
- la funzione di interpretazione $I$ in FOL corrisponde alle tuple contenute nelle tabelle;
- formule FOL corrispono a query del DB; le interpretazioni che rendono la formula vera corrispondono alla risposta alla query.

**Esempio:** Si consideri il database composto dalla tabella

$$Employee(ename, dep, manager, age)$$

- Elencare tutti i dipartimenti nei quali *Joe* lavora per *Jill*:

  $$\exists y.Employee(Joe, x, Jill, y)$$

  *Spiegazione*: Employee è una relazione FOL; Joe, Jill sono costanti del dominio di interpretazione; $x$ è variabile libera, $y$ è bound dal quantificatore esistenziale. Sono modello della query tutte le interpretazioni che assegnano $ename = Joe$, un qualunque valore di dep e age, $manager = Jill$. Sono resituite le assegnazioni delle variabili libere (dunque il quantificatore esistenziale può essere usato come operatore di proiezione; le variabili che sono legate al quantificatore non sono restituite).
- Elencare i nomi e i dipartimenti di tutti i lavoratori che hanno età uguale a 40 anni.

  $$\exists m, a.Employee(x, y, m, a) \land a = 40$$

- Elencare i dati di tutti gli impiegati che sono manager di loro stessi (wat?):

  $$Employee(x, y, x, z)$$

## Unique Name Assumption

La **UNA** (Unique Name Assumption) è un'assunzione che prevede che ogni elemento del dominio sia rappresentato da una e una sola costante. Tale assunzione si esprime in FOL con la seguente formula: siano $\{c_1, ..., c_n\} \subseteq \Delta$ tutte e sole le costanti del dominio, allora

$$\varphi_{UNA} \coloneqq (\bigwedge_{i=1}^{n} \bigwedge_{j=1, j\neq i}^{n} c_i \neq c_j) \land (\forall x \bigvee_{i=1}^{n} c_i = x)$$

## Grounding

Se $\Delta$ è finito e vale la UNA, formule FOL possono essere proposizionalizzate (**grounded**, nel senso di rese *ground*) riformulando i quantificatori come segue:

- $\forall x.\varphi(x) \equiv \bigwedge_{1}^{n} \varphi(c_i)$

- $\exists x.\varphi(x) \equiv \bigvee_{1}^{n} \varphi(c_i)$

In questo senso, dunque, su un dominio $\Delta$ finito è possibile applicare l'algoritmo **DPLL** su una formula FOL. Una volta resa *ground* e applicata la **UNA**, infatti, la formula FOL è anche una formula PL (contiene solo costanti e operatori proposizionali).

— 5 —

# FOL - Tableaux

Il tableaux in FOL funziona in modo simile a quello della PL, introduce due nuove regole di riduzione legate ai quantificatori e, a differenza della PL, può non terminare.

### Regole di riduzione

$\alpha$ rules

$$\frac{\phi \land \psi}{\begin{array}{c}\phi \\ \psi\end{array}} \qquad \frac{\neg(\phi \lor \psi)}{\begin{array}{c}\neg\phi \\ \neg\psi\end{array}} \qquad \frac{\neg(\phi \supset \psi)}{\begin{array}{c}\phi \\ \neg\psi\end{array}}$$

$\neg\neg$ elimination

$$\frac{\neg\neg\phi}{\phi}$$

$\beta$ rules

$$\frac{\phi \lor \psi}{\phi \mid \psi} \qquad \frac{\neg(\phi \land \psi)}{\neg\phi \mid \neg\psi} \qquad \frac{\phi \supset \psi}{\neg\phi \mid \psi}$$

Branch closure

$$\frac{\begin{array}{c}\phi \\ \neg\phi\end{array}}{\mathrm{X}}$$

$\gamma$ rules

$$\frac{\forall x.\varphi(x)}{\varphi(t)} \qquad \frac{\neg\exists x.\varphi(x)}{\neg\varphi(t)}$$

$\delta$ rules

$$\frac{\neg\forall x.\varphi(x)}{\neg\varphi(c)} \qquad \frac{\exists x.\varphi(x)}{\varphi(c)}$$

1. Nelle $\gamma$ rules, $t$ rappresenta una qualunque costante, anche già presente nel tableaux.
2. Nelle $\delta$ rules, $c$ rappresenta una *nuova costante*, non presente nel tableaux.

Perché funziona? Spiegazione:

1. $\forall x.\varphi(x)$ significa che $\varphi$ deve essere soddisfatta per ogni elemento del dominio di interpretazione $\Delta$. Una *costante* $t$ è un elemento del dominio $\Delta$. Pertanto, $\forall x.\varphi(x) \supset \varphi[x/t]$.
   *È lecito applicare più volte una $\gamma$ rule*, utilizzando costanti diverse: l'applicazione rimane valida infatti per ogni costante del dominio. *Può*

*essere necessario applicare più volte una $\gamma$ rule*, ad esempio, per cercare due branch closure su due rami diversi, con due variabili diverse in conflitto (e che condividono come ancestor un quantificatore universale).

2. $\exists x.\varphi(x)$ è equivalente a $\exists c \in \Delta | \varphi(c)$, dunque $\exists x.\varphi(x)$ è soddisfacibile sse $\varphi(c)$ è soddisfacibile.
   È importante che ogni $\delta$ rule introduca una nuova costante: riutilizzare la stessa nello sviluppo di esistenziali diversi modificherebbe la formula di partenza; imporrebbe infatti che *lo stesso* elemento del dominio $\Delta$ verifichi *ogni* esistenziale!

### Infinite tableaux

Se il *tableaux* è applicato ad una formula FOL $\varphi$, e questa formula è **insoddisfacibile**, la costruzione del tableaux **termina** in un numero finito di passi.

Se, però, il *tableaux* è applicato ad una formula **soddisfacibile**, la sua costruzione **può terminare** in un numero finito di passi e lasciare qualche ramo aperto, **oppure non terminare**.

Un *tableaux* nel quale compare un *ramo aperto saturato* sicuramente non può terminare. Nulla può dirsi in altri casi.

**Osservazione:** in quali condizioni un *tableaux* in FOL può non essere decidibile? Nel caso in cui la formula contenga un quantificatore universale e il dominio di interpretazione sia infinito. La $\gamma$ rule può essere applicata una volta per ogni costante del dominio (quindi un numero infinito di volte, dato il dominio infinito). Questa condizione è necessaria affinché si produca un tableaux infinito, ma non sufficiente: potrebbe comunque essere possibile applicare la branch closure ad ogni ramo aperto.

### Open saturated branch

Un *ramo aperto saturato* (open saturated branch) è un ramo nel quale ogni elemento che non sia un *literal* è stato analizzato almeno una volta e, inoltre, in cui ogni $\gamma$ rule è stata applicata con ogni possibile *term* presente sul ramo.

La costruzione di un *ramo aperto saturato* sicuramente non può terminare. È possibile tentare la costruzione di un modello su un ramo saturato analizzando le formule presenti sul ramo.

— 6 —

# ML - Introduzione

La logica modale è un'estensione della logica proposizionale. Una *modalità* è un'operatore che qualifica ("esprime il modo in cui") una proposizione è valida. Numerosi tipi di modalità diverse possono essere rappresentate in ML; storicamente, il primo tipo di modalità ad essere stato impiegato è la modalità *aletica*, comprendente gli operatori "è necessario che", "è possibile che". Un altro tipo di modalità è la *deontica* ("è obbligatorio", "è permesso", "è vietato").

**Definizione 6 (Linguaggio della ML)**
*Nel contesto della logica modale aletica, sono definiti i due operatori modali che possono essere applicati a formule:*

1. $\Box\varphi$: *"è obbligatorio che $\varphi$"*
2. $\Diamond\varphi$ *"è possibile che $\varphi$"*

*Sia $P$ un insieme di proposizioni, allora:*

1. *ogni $p \in P$ è una formula (atomica);*
2. *se $A, B$ sono formule, allora sono formule $A \wedge B, A \vee B, A \subset B, A \equiv B, \neg A$*
3. *se $A$ è una formula, allora sono formule $\Box A, \Diamond A$*

*L'operatore di possibilità può essere definito in termini dell'operatore di necessità:*

$$\Diamond\varphi := \neg\Box\neg\varphi$$

## Semantica della ML (Relational Structure)

Un **frame** (anche chiamato: Kripke frame, Basic frame o Modal frame) è una coppia $F = \langle W, R \rangle$, in cui $W$ è un insieme e $R$ è una relazione binaria sull'insieme $W$.

- $w \in W$ è un *punto*, o *mondo*, del frame;

- $R \subset W \times W$ definisce le *relazioni di accessibilità* fra i mondi: $\langle w_1, w_2 \rangle \in R$ significa che il mondo $w_2$ è accessibile da $w_1$ (e non necessariamente viceversa) nel frame $F$.

Una **funzione di interpretazione** $I$ in ML è una funzione che assegna valori di verità o falsità *relativi ad un certo mondo* alle formule:

$$I : P \to 2^W$$

(il sottoinsieme dei mondi nei quali la formula $P$ è assegnata al valore di verità); alternativamente:

$$I : (P, W) \to \{\top, \bot\}$$

(l'assegnazione alla formula $P$ nel mondo $W$).

Un **modello** $M = \langle F, I \rangle$ è una coppia frame, interpretazione.

### Soddisfacibilità delle formule in ML

Un modello $M$ soddisfa una formula $\varphi$ in un mondo $w$ se $w \in I(\varphi)$. Con l'osservazione che la soddisfacibilità di una formula è relativa al modello **e** al mondo, valgono le solite definizioni:

- $(M, w) \models \varphi \iff w \in I(\varphi)$
- $(M, w) \models \neg\varphi \iff w \notin I(\varphi)$
- $(M, w) \models \varphi \wedge \psi \iff w \in I(\varphi) \wedge w \in I(\psi)$
- $(M, w) \models \varphi \vee \psi \iff w \in I(\varphi) \vee w \in I(\psi)$
- $(M, w) \models \varphi \subset \psi \iff w \in I(\varphi) \subset w \in I(\psi)$
- $(M, w) \models \varphi \equiv \psi \iff w \in I(\varphi) \equiv w \in I(\psi)$

Per gli operatori di necessità e possibilità valgono le seguenti definizioni:

- $(M, w) \models \Box\varphi \iff \forall w' | wRw', (M, w') \models \varphi)$
- $(M, w) \models \Diamond\varphi \iff \exists w' | wRw', (M, w') \models \varphi)$

A parole: $M$ è un modello per $\Box\varphi$ nel mondo $w$ sse per ogni mondo $w'$ raggiungibile da $w$ ($w$ in $R$-relazione con $w'$), $M$ soddisfa $\varphi$ nel mondo $w'$.

**Osservazione**: Si supponga $w \in W, Y = \{w' \in W | wRw'\} = \emptyset$ (dal mondo $w$ non è possibile raggiungere nessun altro mondo). Allora per una qualsiasi formula $\varphi$ vale:

1. $\Box\varphi$ nel mondo $w$ è necessariamente vero;
2. $\Diamond\varphi$ nel mondo $w$ è necessariamente falso.

**Global satisfiability:** $\varphi$ è globalmente soddisfacibile nel modello $M$ sse $\forall w \in W, (M, w) \models \varphi$ ($M$ soddisfa $\varphi$ in ogni mondo).

Una formula $\varphi$ è **valida in un mondo** $w$ di un frame $F$ ($F, w \models \varphi$) sse $\forall I, M = \langle F, I \rangle$ vale $w \in I(\varphi)$ (qualsiasi assegnazione della funzione di interpretazione rende $\varphi$ vera nel mondo $w$).

Una formula $\varphi$ è **valida** ($F \models \varphi$) sse è valida per tutti i mondi del frame.

### Rappresentazione grafica

Si consideri il frame $F = \langle W, R \rangle$, dove:

$$W = \{A, B, C, D\}$$

$$R = \{\langle A, B\rangle, \langle C, B\rangle, \langle A, C\rangle, \langle A, D\rangle, \langle D, B\rangle\}$$

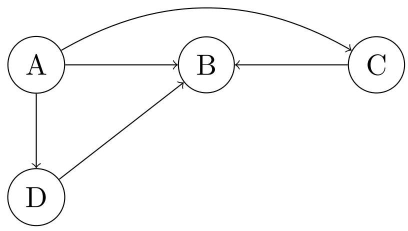


Figure 6.1: Rappresentazione grafica del frame $F$

### Esprimere proprietà sulla struttura del frame

Si consideri il frame $F$ come rappresentato dal grafo di 6.1. Attraverso gli operatori della logica modale è possibile esprimere delle proprietà sulla struttura del grafo (quindi sulla struttura del frame).

Dato il frame $F$, diciamo che il mondo $w$ è successore di $v$ se $\{v, w\} \in R$. La stessa proprietà sulla rappresentazione del frame in forma di grafo diretto può essere espressa come: dato il grafo $G = (V, E)$ il vertice $w \in V$ è successore di $v \in V$ se esiste $e = (v, w) \in E$.

Seguono diversi esempi di proprietà che si possono esprimere su un frame $F = \{W, R\}$, $w \in W$:

1. $\Diamond p$ è vero in $w \iff \exists v$ successore di $w | p$ è vero
2. $\Diamond\Diamond\top$ è vero in $w \iff \exists y$ successore di $v$ successore di $w$
3. $\underbrace{\Diamond \ldots \Diamond}_{\text{n}} p$ è vero in $w \iff$ esiste un cammino di lunghezza $n$ che parte da $w$ per raggiungere il mondo $v | p$ è vero in $v$
4. $\underbrace{\Diamond \ldots \Diamond}_{\text{n}} \top$ è vero in $w \iff$ esiste un cammino di lunghezza $n$ che parte da $w$
5. $\Box\bot$ è vero in $w \iff w$ non ha successori
6. $\Box p$ è vero in $w \iff \forall v$ successori di $w | p$ è vero in $v$
7. $\underbrace{\Box \ldots \Box}_{\text{n}} p$ è vero in $w \iff$ tutti i cammini che partono da $w$ di lunghezza $n$ arrivano in un qualche mondo $v | p$ è vero in ogni $v$

8. $\underbrace{\Box \ldots \Box}_{\text{n}} \bot$ è vero in $w$ $\iff$ tutti i cammini che partono da $w$ sono lunghi al più $n-1$.

### Logiche multi-modali

Le definizioni date per una logica modale con una certa modalità possono essere estese a logiche modali che prevedono di usare più modalità contemporaneamente.

In generale, una logica multi-modale con un numero di modalità $n$ prevederà $n$ operatori $\Box$ ($\Box_1, \ldots, \Box_n$) e $n$ operatori $\Diamond$ ($\Diamond_1 \ldots \Diamond_n$).

Il frame di Kripke per una logica multi-modale ($F = \langle W, R_1, \ldots, R_n \rangle$) prevede un insieme di mondi $W$ e $n$ insiemi di relazioni $R_1, \ldots, R_n$; gli operatori della modalità $i$ sono valutati secondo la relazione $R_i$.

# — 7 —
# ML - Sistemi di logica modale

Diversi sistemi di logica modale possono essere costruiti a seconda di quali assunzioni sono valide per la relazione di accessibilità $R$. Il significato di porre determinate assunzioni su $R$ dipende dalla modalità che è necessario rappresentare.

Si supponga, ad esempio, una logica temporale, in cui $uRv$ ha il significato di "$u$ avviene prima di $v$". È sensato richiedere che una relazione su questo sistema sia **transitiva** ($uRv, vRw \to uRw$), al tempo stesso, una relazione su questo sistema difficilmente deve essere **simmetrica** ($uRv \to vRu$).

## Assiomi della logica modale normale

In aggiunta agli assiomi di Hilbert sulla logica proposizionale, la logica modale introduce l'assioma di distribuzione (**K**) e la regola di necessità (*necessitation rule* - **Nec**).

**Nec**: sia $\varphi$ una tautologia, allora $\Box\varphi$ è vero.

**K**: $\Box(\varphi \supset \psi) \supset (\Box\varphi \supset \Box\psi)$

**Esempio** (Nec): Sia $\varphi := (p \vee \neg p)$; $\varphi$ è una tautologia, e per **Nec** è vero anche $\Box\varphi, \Box\Box\varphi, \ldots$

**Osservazione** (Nec): la *necessitation rule* non può essere scritta come $\varphi \supset \Box\varphi$; questa affermazione non è necessariamente vera! $\varphi$ deve essere una formula valida per gli assiomi.

Per tutte le logiche modali normali (per tutti i frame di Kripke) valgono gli assiomi di Hilbert della logica proposizionale, **K** e **Nec**.

## Altri possibili assiomi della logica modale

### Assioma T (riflessività):

Se un frame $F = \langle W, R \rangle$ è riflessivo, allora vale

$$\Box\varphi \supset \varphi$$

Analogamente, se $F$ è riflessivo, vale $\forall w \in W | wRw$.

**Dimostrazione(correttezza)**

*Dimostrare la correttezza(soundness) dell'assioma significa dimostrare che, dato un frame riflessivo, allora l'assioma T, così come è espresso con gli operatori della logica modale è necessariamente valido. In altre parole, dato un frame riflessivo, se vale la premessa $\Box\varphi$, allora deve valere la conclusione $\varphi$*

Sia $F = \langle W, R\rangle$ un frame di Kripke riflessivo, sia $w \in W$ un qualsiasi mondo del frame.
Sia $M = \langle F, I\rangle$ un modello per il frame di Kripke di cui sopra, sia $(M, w) \models \Box\varphi$.
Si osserva che $wRw$ (il frame è riflessivo). Allora se $(M, w) \models \Box\varphi$, è necessariamente vero $(M, w) \models \varphi$.

**Dimostrazione(completezza)**

*Dimostrare la completezza(completeness) dell'assioma significa dimostrare che, dato un frame non riflessivo, allora l'assioma T, così come è espresso con gli operatori della logica modale non è necessariamente valido. In altre parole, dato un frame non riflessivo in cui vale la premessa, la conclusione può non valere.*

Sia $F = \langle W, R\rangle$ un frame di Kripke non riflessivo. Dunque, esiste $w \in W$ tale per cui $w \not R w$.
Sia $M = \langle F, I\rangle$ un modello per il frame di Kripke di cui sopra, sia $I(\varphi) = W \setminus \{w\}$ ($\varphi$ è vero ovunque nel modello tranne che nel mondo $w$).
Siccome $w$ non può accedere a sé stesso (l'unico mondo in cui $I(\varphi) = \perp$), vale $(M, w) \models \Box\varphi$, e vale anche $(M, w) \not\models \varphi$. Dunque, $F \not\models \Box\varphi \supset \varphi$.

**Assioma B (simmetricità):**

Se un frame $F = \langle W, R\rangle$ è simmetrico, allora vale

$$\varphi \supset \Box\Diamond\varphi$$

Analogamente, se $F$ è simmetrico, vale la seguente proprietà sulla relazione $R$: $\forall v, w \in W | vRw \supset wRv$

**Dimostrazione (correttezza):**

Sia $F = \langle W, R\rangle$ un frame di Kripke simmetrico, sia $w \in W$ un qualsiasi mondo del frame.
Sia $M = \langle F, I\rangle$ un modello per il frame di Kripke di cui sopra, sia vera la premessa $(M, w) \models \varphi$. Per ogni mondo $w'|wRw'$, vale anche $w'Rw$ (il frame è simmetrico), dunque $\forall w'|wRw', (M, w') \models \Diamond\varphi$. Allora, $(M, w) \models \Box\Diamond\varphi$.
Dunque, $(M, w) \models \varphi \supset \Box\Diamond\varphi$.

**Dimostrazione (completezza):**

Sia $F = \langle W, R\rangle$ un frame di Kripke non simmetrico, dunque esistono $w, w' \in W$ tale per cui $wRw'$ ma $w' \not R w$.
Sia $M = \langle F, I\rangle$ un modello per il frame di Kripke di cui sopra, sia $I(\varphi) = \{w\}$ ($\varphi$ è vero solo in $w$). Dunque, è vera la premessa $(M, w) \models \varphi$.
Vale $(M, w') \not\models \Diamond\varphi$, dato che $w' \not R w$; pertanto vale $(M, w) \not\models \Box\Diamond\varphi$. Dunque, $(M, w) \not\models \varphi \supset \Box\Diamond\varphi$.

**Assioma D (serialità):**

Se un frame $F$ è seriale, allora vale

$$\Box\varphi \supset \Diamond\varphi$$

Analogamente, se $F$ è seriale, vale $\forall w \in W \exists u \in W | wRu$

**Dimostrazione (correttezza):**

Sia $F = \langle W, R\rangle$ un frame di Kripke seriale, sia $w \in W$ un qualsiasi mondo del frame. Osservo che, dato $F$ seriale, allora $\exists w' \in W | wRw'$
Sia $M = \langle F, I\rangle$ un modello per il frame di Kripke di cui sopra, sia la premessa $(M, w) \models \Box\varphi$ vera. Dato che $wRw'$, è necessario che $(M, w') \models \varphi$.
Dunque, $(M, w) \models \Diamond\varphi$. Allora $F \models \Box\varphi \supset \Diamond\varphi$

**Dimostrazione (completezza):**

Sia $F = \langle W, R\rangle$ un frame di Kripke non seriale, dunque esiste $w \in W$ tale per cui $\nexists w' \in W | wRw'$ (w non è connesso a nessun altro mondo).
Per qualsiasi modello vale $w \models \Box\varphi$ e $w \not\models \Diamond\varphi$.
Dunque, $F \not\models \Box\varphi \supset \Diamond\varphi$.

**Assioma 4 (transitività):**

Se un frame $F$ è transitivo, allora vale

$$\Box\varphi \supset \Box\Box\varphi$$

Analogamente, se $F$ è transitivo, vale $\forall u, v, w \in W | (uRv \wedge vRw) \supset uRw$

**Dimostrazione(correttezza):**

Sia $F = \langle W, R\rangle$ un frame di Kripke transitivo, siano $w, w' \in W$ due mondi del frame tali per cui $wRw'$.
Sia $M = \langle F, I\rangle$ un modello per il frame di Kripke di cui sopra, sia la premessa $(M, w) \models \Box\varphi$ vera. Dato $wRw'$ allora $(M, w) \models \varphi$.
Vale: $(M, w') \models \Box\varphi$ (1) è vero sse per ogni mondo $w''$ tale che $w'Rw''$ vale

$(M, w'') \models \varphi$ (2). Ma (2) è vero: dato che $F$ è transitivo, osservo che $wRw''$; vale $(M, w) \models$, dunque $(M, w'') \models \varphi$. Allora $(M, w') \models \Box\varphi$ e $(M, w) \models \Box\Box\varphi$. Dunque, $F \models \Box\varphi \supset \Box\Box\varphi$.

**Dimostrazione(completezza):**

Sia $F = \langle W, R \rangle$ un frame di Kripke non transitivo, dunque esistono $w, w', w'' \in W$ tali per cui $wRw', w'Rw'', w\not Rw''$.
Sia $M = \langle F, I \rangle$ un modello per il frame di Kripke di cui sopra, sia $I(\varphi) = W \setminus \{w''\}$ ($\varphi$ è sempre vero, tranne in $w''$).
Vale la premessa, dunque $(M, w) \models \Box\varphi$, che implica $(M, w') \models \varphi$.
Vale $(M, w'') \not\models \varphi$ per come è definita l'interpretazione, quindi $(M, w') \not\models \Box\varphi$, e allora $(M, w) \not\models \Box\Box\varphi$.
Dunque, $F \not\models \Box\varphi \supset \Box\Box\varphi$.

**Assioma 5 (euclideicità):**

Se un frame $F$ è euclideo, allora vale

$$\Diamond\varphi \supset \Box\Diamond\varphi$$

Analogamente, se $F$ è euclideo, vale $\forall u, v, w \in W | (uRv \wedge uRw) \supset vRw$

**Dimostrazione(correttezza):**

Sia $F = \langle W, R \rangle$ un frame di Kripke euclideo, sia $w \in W$.
Sia $M = \langle F, I \rangle$ un modello per il frame di Kripke di cui sopra, supponiamo valga la premessa $(M, w) \models \Diamond\varphi$, allora esiste un mondo $w' \in W$ tale che $wRw'$ e $(M, w') \models \varphi$.
Ora, per qualsiasi altro mondo $w''$ tale che $wRw''$, per euclideicità vale anche $w''Rw'$, allora $(M, w'') \models \Diamond\varphi$. Quindi, se in qualunque mondo $w''$ vale $(M, w'') \models \Diamond\varphi$, allora vale $(M, w) \models \Box\Diamond\varphi$.
Dunque, $F \models \Diamond\varphi \supset \Box\Diamond\varphi$.

**Dimostrazione(completezza):**

Sia $F = \langle W, R \rangle$ un frame di Kripke non euclideo, dunque esistono $w, w', w'' \in W$ tali per cui $wRw', wRw'', w'\not Rw''$.
Sia $M = \langle F, I \rangle$ un modello per il frame di Kripke di cui sopra, sia $I(\varphi) = \{w''\}$ ($\varphi$ è vero solo in $w''$).
Vale $(M, w'') \models \varphi$ e $(M, w) \models \Diamond\varphi$; vale anche però $(M, w') \not\models \Diamond\varphi$, perché $w'\not Rw''$ e non ci sono altri mondi in cui $\varphi$ è vero. Pertanto, $(M, w) \not\models \Box\Diamond\varphi$.
Dunque, $F \not\models \Diamond\varphi \supset \Box\Diamond\varphi$.

### Categorie di sistemi logici modali

I sistemi logici sono categorizzati a seconda di quali assiomi valgono:

| Classe | Assiomi validi | Descrizione |
|---|---|---|
| **K** | | La classe di tutti i frame di Kripke |
| **K4** | 4 | La classe di tutti i frame di Kripke transitivi |
| **KT** | T | La classe di tutti i frame di Kripke riflessivi |
| **KB** | B | La classe di tutti i frame di Kripke simmetrici |
| **KD** | D | La classe di tutti i frame di Kripke seriali |
| **S4** | 4, T | La classe di tutti i frame di Kripke riflessivi e transitivi |
| **S5** | 5, T | La classe di tutti i frame di Kripke con $R$ relazione di equivalenza |
| **S5** | 4, T, B | Caratterizzazione equivalente di S5[1] |

[1] Per esercizio, si può dimostrare con il tableaux FOL: $\models \{4, B\} \equiv \{5\}$

— 8 —

# ML - Tableaux

*Il funzionamento del tableaux in ML presenta delle forti analogie con il tableaux della FOL. Sostanzialmente, l'operatore $\Diamond$ di ML può essere mappato sull'operatore $\exists$ di FOL; l'operatore $\Box$ di ML può essere mappato sull'operatore $\forall$ di FOL. Accortezze devono essere prestate sulle relazioni del frame.*

Al solito, il **tableaux** è un algoritmo per calcolare la soddisfacibilità - insoddisfacibilità - validità di una formula. In logica modale, un tableaux è un albero in cui in ogni nodo è presente un'affermazione relativa alla formula in un certo mondo ($w \models \varphi$, $w \not\models \varphi$), oppure è sfruttata la $R$ relazione per accedere ad un mondo adiacente ($wRw'$), passo necessario per proseguire la costruzione nel caso di formule del tipo $\Box\psi, \Diamond\psi$.

Un ramo è **chiuso** se contiene la contraddizione$w \models \varphi$ e $w \not\models \varphi$ (analogamente: $w \models \varphi, w \models \neg\varphi$), altrimenti è detto **aperto**. Se tutti i rami sono chiusi, il tableaux è detto chiuso.

La logica modale è **decidibile** (dunque la costruzione del tableaux termina in un numero finito di iterazioni).

## Regole di espansione per operatori proposizionali

Sono del tutto analoghe alle classiche $\alpha$ e $\beta$ rules (*alpha equivalence* e *branch splitting*).

$\alpha$ rules

$$\frac{w \models \phi \wedge \psi}{\begin{array}{c} w \models \phi \\ w \models \psi \end{array}} \qquad \frac{w \not\models \phi \vee \psi}{\begin{array}{c} w \not\models \phi \\ w \not\models \psi \end{array}} \qquad \frac{w \not\models \phi \supset \psi}{\begin{array}{c} w \models \phi \\ w \not\models \psi \end{array}}$$

$\neg\neg$ elimination

$$\frac{w \not\models \neg\phi}{w \models \phi}$$

$\beta$ rules

$$\frac{w \models \phi \vee \psi}{w \models \phi \,\big|\, w \models \psi} \qquad \frac{w \not\models \phi \wedge \psi}{w \not\models \phi \,\big|\, w \not\models \psi} \qquad \frac{w \models \phi \supset \psi}{w \not\models \phi \,\big|\, w \models \psi}$$

Branch closure

$$\frac{\begin{array}{c} w \models \phi \\ w \not\models \phi \end{array}}{\text{X}}$$

**Regole di espansione per operatori modali**

$\gamma$ rules

$$\frac{w \models \Box\varphi}{\begin{array}{c} wRw' \\ w' \models \varphi \end{array}} \qquad \frac{w \not\models \Diamond\varphi}{\begin{array}{c} wRw' \\ w' \not\models \varphi \end{array}}$$

$\delta$ rules

$$\frac{w \not\models \Box\varphi}{\begin{array}{c} wRw' \\ w' \not\models \varphi \end{array}} \qquad \frac{w \models \Diamond\varphi}{\begin{array}{c} wRw' \\ w' \models \varphi \end{array}}$$

1. Nelle $\gamma$ rules, $w'$ rappresenta un qualsiasi mondo, anche già presente nel tableaux.
2. Nelle $\delta$ rules, $w'$ rappresenta un nuovo mondo non presente nel tableaux.

— 9 —

# DL - Introduzione

La logica descrittiva è una famiglia di formalismi che permettono una rappresentazione della conoscenza in un dominio - mondo (*Knowledge Base*). DL prevede prima la definizione della **terminologia** rilevanti del dominio (classi di oggetti e gerarchia fra queste); questi concetti sono poi impiegati per specificare proprietà sugli **oggetti** che compongono il dominio (la "descrizione" del "mondo").

Obiettivo della DL è rispondere a task di ragionamento sulla struttura della terminologia (p.es.: il termine - la classe $Canary$ è una specializzazione di $Bird$?) oppure sugli oggetti che compongono il mondo (p.es.: l'oggetto $foo$ gode della proprietà - è membro della classe $Canary$?).

## Knowledge Base

In DL, la Knowledge Base è composta dall'insieme della *terminologia* e della *descrizione* del mondo. Queste due componenti sono chiamate **TBox** e **ABox**.

1. Il **TBox** è l'insieme della *terminologia*, il "vocabolario" della KB. Il vocabolario consiste di *atomic concepts*, che denotano classi di oggetti, e *roles*, che denotano relazioni fra *concepts*. Ulteriori *concepts* derivati possono essere composti combinando *atomic roles*; equivalentemente per le *roles*.
2. La **ABox** introduce *oggetti* del mondo, assegnando loro un *nome* e effettuando *asserzioni* su essi; le asserzioni sono espresse nella terminologia della TBox.

## Attributive Concept Language with Complements

$\mathcal{ALC}$ (Attributive Concept Language with Complements) è un linguaggio di description logics. La descrizione del mondo in $\mathcal{ALC}$ comprende *concept atomici* e *roles atomici* (proprietà che descrivono i concepts); descrizioni complesse possono essere ottenute componento descrizioni atomiche secondo le regole sintatiche.

**Definizione 7 (Linguaggio $\mathcal{ALC}$)**
*I concepts sono composti secondo le seguenti regole sintattiche.*
***Concept atomici:***

1. *A un concept atomico*
   *Esempio: $Person$*
2. *$\top$ il concept universale*
3. *$\bot$ l'opposto del concept universale*
4. *$\neg A$ la negazione di un concept atomico*

***Proposizioni (concept complessi, well formed formulas):***

1. *$F \coloneqq C \sqcap D$ l'intersezione fra due concept*
   *Esempio: $Woman \coloneqq Person \sqcap Female$*
2. *$F \coloneqq C \sqcup D$ l'unione di due concept*
   *Esempio: $Parent \coloneqq Father \sqcup Mother$*
3. *$F \coloneqq \forall R.C$ la restrizione universale di un concept ad un role*
   *Esempio: $F \coloneqq Person \sqcap \forall hasChild.Female$ (F descrive le persone che hanno tutte le figlie femmina -* ***comprese quelle che non hanno figli in generale****)*
   *Esempio: $G \coloneqq Person \sqcap \forall hasChild.\bot$ (G descrive tutte le persone che non hanno figli)*
4. *$F \coloneqq \exists R.C$ la restrizione esistenziale di un concept rispetto ad un role*
   *Esempio: $Parent \coloneqq Person \sqcap \exists hasChild.\top$ (Parent sono le persone che hanno (almeno) un figlio)*
   *Esempio: $F \coloneqq Person \sqcap hasChild.Female$ (F descrive le persone che hanno (almeno) una figlia femmina)*

Un altro linguaggio di description logic è $\mathcal{CL}$; può essere visto come una restrizione di $\mathcal{ALC}$ ai soli operatori della logica proposizionale (non comprende $\forall$ e $\exists$).

## Interpretazione in ALC

Sia $\mathscr{A}$ l'insieme di tutti gli *atomic concepts*, sia $\mathscr{R}$ l'insieme di tutti gli *atomic roles*, sia $\mathscr{N}$ l'insieme dei nomi degli oggetti[1]; l'interpretazione è una coppia $\langle \Delta_I, I \rangle$ composta di:

1. un dominio di interpretazione $\Delta_I$;
2. una funzione di interpretazione $I$:
   a) $I : I(a \in \mathscr{A}) \to A_I \subseteq \Delta_I$ (mappa *atomic concepts* in un sottoinsieme di elementi del dominio),

[1] I nomi sono discussi in seguito, nella descrizione della *ABox*

b) $I : I(r \in \mathscr{R}) \rightarrow R_I \subseteq \Delta_I \times \Delta_I$ (mappa *atomic roles* in un sottoinsieme di coppie di elementi del dominio - una relazione binaria sul dominio),
c) $I : I(n \in \mathscr{N}) \rightarrow d \in \Delta_I$ (mappa *nomi* di oggetti in elementi del dominio)[2].

L'interpretazione si estende anche a concetti composti secondo le seguenti regole:

1. $I(\top) = \Delta_I$ (l'interpretazione del concept universale è l'intero dominio di interpretazione)
2. $I(\bot) = \emptyset$ (analogamente a sopra)
3. $I(\neg A) = \Delta_I \setminus I(A)$ (l'interpretazione di $\neg A$ è complementare a quella di $A$)
4. $I(C \sqcap D) = I(C) \cap I(D)$
5. $I(C \sqcup D) = I(C) \cup I(D)$
6. $I(\forall R.C) = \{a \in \Delta_I | \forall b.\langle a, b\rangle \in I(R) \rightarrow b \in I(C)\}$ (gli elementi del dominio di interpretazione tali che, qualsiasi altro elemento con cui essi sono in $R$-relazione, questo è in $I(C)$)
7. $I(\exists R.C) = \{a \in \Delta_I | \exists b.\langle a, b\rangle \in I(R) \rightarrow b \in I(C)\}$ (gli elementi del dominio di interpretazione tali che esiste almeno un elemento con cui essi sono in $R$-relazione ed è in $I(C)$)

## Asserzioni terminologiche e TBox

Due *concept* $C, D$ sono detti **equivalenti**, e si scrive $C \equiv D$, sse per ogni interpretazione $I$ vale $I(C) = I(D)$

**Esempio**[3]: $\forall hasChild.Female \sqcap \forall hasChild.Student \equiv \forall hasChild.(Female \sqcap Student)$

Un *concept* $C$ è **incluso**[4] da un *concept* $D$ ($C$ è più specifico di $D$), e si scrive $C \sqsubseteq D$, sse per ogni interpretazione $I$ vale $I(C) \subseteq I(D)$.

**Esempio:** $Female \sqsubseteq Person; \forall hasChild.Female \sqsubseteq \forall hasChild.Person$

**Osservazione:** Se $C \sqsubseteq D$ e $D \sqsubseteq C$, allora $C \equiv D$.

Una **definizione** è un'*equivalenza* il cui membro sinistro è un *atomic concept*.

**Esempio:** $Woman \equiv Person \sqcap Female$

Una **specializzazione** è un'*inclusione* il cui membro sinistro è un *atomic concept*.

[2]Ibid.

[3]La dimostrazione è immediata e segue dalle regole sintattiche di $\mathcal{ALC}$ descritte prima

[4]Un sinonimo per "è incluso" è "è sussunto da" (is subsumed by)

**Esempio:** $Student \sqsubseteq Person$
**Esempio:** $Father \sqsubseteq \forall hasChild.Person$

La **terminologia** o **TBox** è un insieme di asserzioni terminologiche, cioé *definizioni* e *specializzazioni.* Queste asserzioni pongono dei vincoli (constraints) su tutti i possibili modelli del mondo.

### Task di ragionamento su TBox

Un *concept* $P$ **è soddisfacibile** rispetto ad una *TBox* $T$ se esiste un'interpretazione $I$ tale che $\forall \vartheta \in T | I \models \vartheta$ (I deve soddisfare i constraints posti dal *TBox*) e $I \models P$, cioé $I(P) \neq \emptyset$.
Se $P$ è soddisfacibile, allora $I$ è detta *modello* di $P$.

Dati due *concept* $P, Q$ e una *TBox* $T$, si possono descrivere i seguenti task di ragionamento:

1. Soddisfacibilità rispetto a $T$: $T \models P$?
2. Sussunzione: $T \models P \sqsubseteq Q$, oppure $T \models Q \sqsubseteq P$?
3. Equivalenza: $T \models P \sqsubseteq Q$ e $T \models Q \sqsubseteq P$?
4. Disgiuntività: $T \models P \sqcap Q \sqsubseteq \bot$?

**Osservazione**: sussunzione, equivalenza e disgiuntività di due *concept* dipendono esclusivamente dalla *TBox* (non dipendono dall'interpretazione), mentre ha senso parlare di un'interpretazione che soddisfa un *concept.*

### Sintassi e semantica di ABox

La *ABox* contiene la descrizione del mondo, si introducono infatti **nomi** per gli oggetti del mondo e si asserisce l'appartenenza di *nomi* a *concept* e a *roles*

Una **concept assertion** è del tipo : *Student*(*paul*), da leggersi "*paul* appartiene a *Student*", in cui *Student* è un *concept* e *paul* un *nome.*
Una **role assertion** è del tipo: $fatherOf(john, julia)$, in cui $fatherOf$ è una relazione binaria (*role*); *john*, *julia* dei *nomi.*

**Unique Name Assumption**: Si assume che *nomi* diversi denotino oggetti diversi.

**TBox Set Constructor**: È lecito definire un *concept*, detto **nominal**, nella *TBox* elencando i *nomi* che lo compongono. Esempio: $Student \equiv \{John, Mary, Paul\}$

### Task di ragionamento su ABox

Data una *ABox* $A$ si possono descrivere i seguenti task di ragionamento rispetto a una *TBox* $T$:

1. Consistenza o soddisfacibilità: $A$ è consistente rispetto a $T$ se esiste un'interpretazione $I$ che è modello sia per $T$ sia per $A$;
2. Instance checking: verificare se un nome $a$ è contenuto dal (è instanza del) *concept* $C$ ($A \models C(a)$ sse ogni interpretazione che soddisfa $A$ soddisfa anche $C(a)$);
3. Instance retrieval: dato un *concept* $C$, ricavare tutte le sue istanze (tutti i *nomi* contenuti in $C$);
4. Concept realization: dato un insieme di *concepts* $\mathscr{C}$ e un *nome* $a$, ricavare il *concept* $C$ più specifico (rispetto all'ordinamento per inclusione $\sqsubseteq$) per cui $a$ è istanza di $C$ ($A \models C(a)$).

**Ragionamento per espansione**: è una tecnica per svolgere i task di ragionamento da compiere su una *ABox* $A$ rispetto ad una *TBox* $T$ aciclica (che non contiene definizioni ciclice). L'**espansione** di $A$ rispetto a $T$ è una *ABox* $A'$ che contiene tutte le affermazioni di $A$ e, in più, ciascuna affermazione $C(a)$ di $A$ è ripetuta su tutti i *concept* $D$ tali per cui $C \sqsubseteq D$.

**Esempio**: se la *TBox* è definita $T = \{C \equiv D \sqcap E\}$ e la *ABox* è definita $A = \{C(a)\}$, l'espansione di $A$ rispetto a $T$ è la *ABox* $A' = \{C(a); D(a); E(a)\}$.

La *ABox* $A$ è **consistente**[5] sse la sua espansione $A'$ è non contiene contraddizioni.

La *ABox* $A$ è **consistente** rispetto ad una *TBox* $T$ sse lo è la sua espansione $A'$, nel modo definito prima (esiste $I$ modello sia per $A$ sia per $T$).

$A \models C(a)$ (**instance checking**) sse $A \cup \{\neg C(a)\}$ non è consistente.

**Instance retrieval** e **Concept realization** possono essere ricavati direttamente scorrendo tutte le affermazioni della *ABox* espansa.

I sistemi di ragionamento della DL possono assumere un **mondo aperto** (**O**pen **W**orld **A**ssumption) o un **mondo chiuso** (**C**losed **W**orld **A**ssumption):

1. assumere un mondo chiuso significa che ciò che non è esplicitamente asserito è falso;
2. assumere un mondo aperto significa che ciò che non è esplicitamente asserito è sconosciuto (né vero, né falso).

**Esempio:** sia $T = \{C \equiv A \sqcap B\}; A = \{A(p)$, è vero $C(q)$? Assumendo OWA: nulla può dirsi; assumendo CWA: no.

[5] Rispetto a **nessuna** *TBox*, o rispetto ad una *TBox* vuota

# — 10 —

# DL - Databases

Una Knowledge Base del tipo di un database relazionale può essere rappresentata attraverso la description logic. Dalla struttura (modello entità - relazioni) si estraggono le asserzioni terminologiche (TBox), la ABox contiene le realizzazioni delle tuple e delle relazioni fra esse.

Tutti i dati che il database può contenere rispettando la struttura sono i possibili modelli; i dati che effettivamente il database contiene sono il modello sul quale vengono svolti i task di ragionamento.

### Schema di esempio

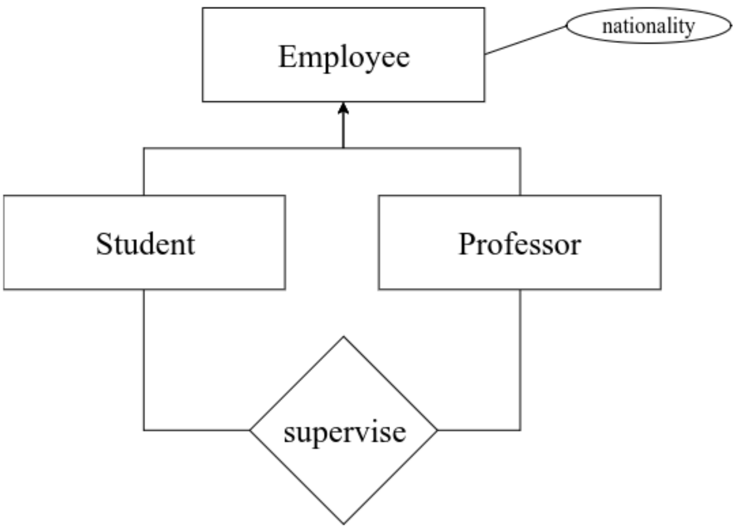


Figure 10.1: Schema E/R

Una possibile TBox per lo schema è la seguente: $\{Professor \sqsubseteq Employee, Student \sqsubseteq Employee, Professor \equiv \forall supervises.Student\}$

Il database contiene le seguenti tuple:
**Student:** $(Joe, Italian)$, $(Jill, American)$, $(James, Canadian)$
**Professor:** $(Foo, Mexican)$

Sono verificate le seguenti relazioni: **supervise**: $\langle Foo, Joe\rangle, \langle Foo, Jill\rangle$

Una ABox che contiene i dati sopra indicati ed è consistente con la TBox è la seguente: $\big\{Student(Joe), Student(Jill), Student(James), Professor(Foo),$ $Nationality(Joe, Mexican), Nationality(Jill, American), Nationality(James, Canadian),$ $Nationality(Foo, Mexican), Supervise(Foo, Joe), Supervise(Foo, Jill)\big\}$

### Queries

Queries di **selezione** sul database si possono in generale tradurre come *reasoning task* sulla ABox del tipo **instance retrieval**.

**Esempio:** *(si consideri $T, A$ la TBox e la ABox dell'esempio precedente)*
NL Query: Who are the mexican employees?
DL Query: $T, A \models Employee \sqcap \exists Nationality.Mexican$[1]
Answer: $\{Joe, Foo\}$

Le queries che richiedono una risposta affermativa o negativa possono essere tradotte come *instance checking*: **Esempio:** NL Query: Is Joe mexican? DL Query: $T, A \models Nationality(Joe, Mexican)$ Answer: YES (la relazione è contenuta direttamente nella ABox).

**Esempio:** NL Query: Is Joe american?
DL Query: $T, A \models Nationality(Joe, American)$ Answer: NO se si assume un mondo chiuso (CWA), nulla può dirsi se si assume un mondo aperto (OWA).

[1]Questa formalizzazione restituirebbe anche employee con più nazionalità, fra cui quella messicana. Alternativamente, $T, A \models Employee \sqcap \forall Nationality.Mexican$ restituirebbe gli employee con nazionalità messicana, *o nessuna nazionalità*.