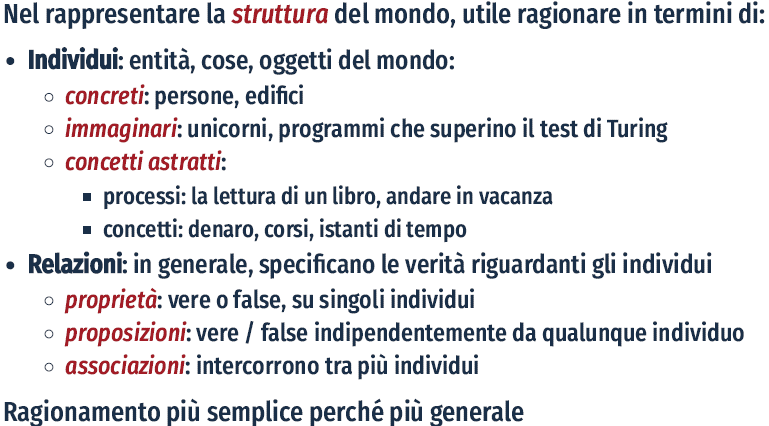
RAPPRESENTAZIONE E RAGIONAMENTO RELAZIONALE

[Cap 12 del libro] La logica del primo ordine serve per separare gli individui dalle relazioni. La rappresentazione passa da quella a stati visti come atomi inscindibili di un problema nei problemi di ricerca ad una rappresentazione fattorizzata tramite variabili nei CSP e queste variabili possono essere sia booleane che non booleane e adesso dopo aver visto le clausole proposizionali affrontiamo la logica del primo ordine che ci dà la possibilità di separare gli individui come entità indipendentemente dal fatto che tali entità siano concrete o astratte dalle relazioni che sussistono su questi individui e che ne danno una rappresentazione delle caratteristiche che essi possiedono.



Questo tipo di ragionamento ci permetterà di essere il più generali possibili e le rappresentazione risulteranno più compatte.

La forza espressiva delle relazioni ci permetterà di gestire:

* Proprietà 🡪 degli individui
* Proposizioni 🡪 indipendenti dagli individui
* Associazioni 🡪 ovvero relazioni che intercorrono tra più individui

Nella rappresentazione proposizionale avevamo degli atomi definiti letterali i quali sono inscindibili e a secondo del nome che utilizziamo per il letterale ci aiutiamo con un nome mnemonico per associare il letterale ad un determinato individuo, tuttavia questo processo non permette di generalizzare una determinata caratteristica che vogliamo sia associata a più individui differenti tra di loro e inoltre cosa più difficile da controllare non si ha la garanzia che due letterali pur avendo un nome mnemonico che sembra fare riferimento ad uno stesso individuo in realtà lo facci ovvero non vi è sicurezza come dimostrato nell’esempio:

Text

Description automatically generated

Per evitare queste problematiche di ambiguità andiamo a separare le relazioni dagli individui e quindi dai precedenti letterali visti tiriamo fuori gli individui che saranno i deviatori(del circuito) 0vvero s1 s2 s3 e teniamo separati da questi le loro proprietà che sono proprietà comuni che possono condividere ovvero la proprietà (o caratteristica di essere su ovvero up oppure di essere funzionanti con ok). E quindi avremo che le proprietà saranno viste in questo caso come relazioni unarie che agiscono su di un individuo come nell’esempio:

Text

Description automatically generated

Le relazioni possono avere qualsiasi grado di arietà.

Questa rappresentazione porta i seguenti vantaggi:

* Risulta + naturale
* È possibile agire anche su domini sconosciuti, senza avere conoscenza del numero degli individui né di quelle che siano le loro caratteristiche
* Ragionamento + generale
* Esistenza incerta degli individui
* Ragionamento su un numero infinito di individui e proprietà

Per effettuare queste modifiche apporteremo un upgrade nella sintassi e nella semantica visto fino ad ora nei costrutti proposizionali e in particolare prima andremo a definire la teoria dei modelli della logica del primo ordine e in seguito la teoria della dimostrazione della logica del primo ordine che è quella che ci interessa dal punto di vista dell’ingegneria della conoscenza.

Il progettista anche in questo caso deve caratterizzare il mondo dandogli un interpretazione intesa che sarà quella che andrà a codificare nella KB.

Diagram

Description automatically generated

A destra abbiamo il designer della KB che nella sua mente associa ad ogni nome un individuo e a dei nomi delle proprietà le quali saranno rappresentate nella KB e nel calcolatore come mostrato a sinistra.

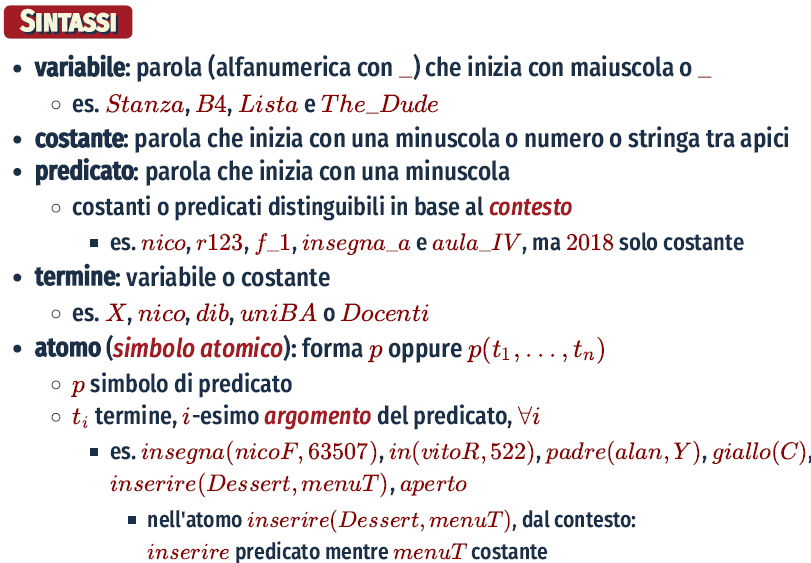
Quanto precedentemente mostrato è un processo che prende il nome di CONCETTUALIZZAZIONE, il quale permette di associare simboli pensati per determinate rappresentazioni agli individui o alle relazioni che essi denotato.

Text

Description automatically generated

DATALOG: LINGUAGGIO RELAZIONALE A REGOLE

Un linguaggio non tale da coprire l’intera logica dei predicati (la logica dei predicati è detta logica del primo ordine). Esso estende il linguaggio delle clausole definite proposizionali con sintassi del calcolo dei predicati e convenzioni del PROLOG(è un linguaggio logico Turing completo).



Una prima distinzione tra la logica proposizionale vista in precedenza e la logica dei predicati di datalog sta nelle Variabili le quali mentre nella logica proposizionale esse risultavano assumere l’essenza di atomo il quale può essere vero o falso adesso una variabile è mappata su di un individuo del dominio e prende un individuo.

Ricordiamo che le variabili in questo contesto ci permettono di dire e di ragionare su fatti in generale poiché esse saranno sostituite da costanti per diventare ground per tali ragioni sono la modalità con la quale generalizziamo il ragionamento sugli individui rappresentati.

Un atomo come visto nello specchietto della sintassi può essere un predicato che non ha argomenti ovvero con arietà pari a 0 in questo caso il predicato sta a significare le variabili proposizionali che potrà essere vero o falso indipendentemente dagli individui.

Oppure nella forma più utilizzata un atomo sarà un predicato con una arietà n > 0.

Si potrà lavorare ancora con le clausole definite proposizionali, ma la differenza sta nel fatto che qui gli atomi sono atomi della logica del primo ordine(ovvero logica dei predicati) e per tale ragione saranno nelle forme precedentemente descritte.

Text

Description automatically generated

A parte la complessità dell’atomo il resto permane indifferente nella sintassi delle clausole e delle regole.

Per introdurre la semantica delle clausole viste con la sintassi introduciamo il termine di ESPRESSIONE che denota in generale un termine o un atomo o una clausola definita o una query. L’espressione GROUND sta a rappresentare il livello base di una proposizione ove avremo un atomo nel quale non sono presenti variabili né consegue che ha solo costanti e non dipende da un dominio di valori che la variabile può catturare.

Text

Description automatically generated

Ora vediamo il concetto di interpretazione che qui nel Datalog cambia poiché prima con le clausole definite un Interpretazione consisteva nel fornire dei grandi di verità agli atomi, qui in quanto gli atomi sono componenti sintattiche complesse l’interpretazione cambia ed è data da una tripla così specificata:

Text

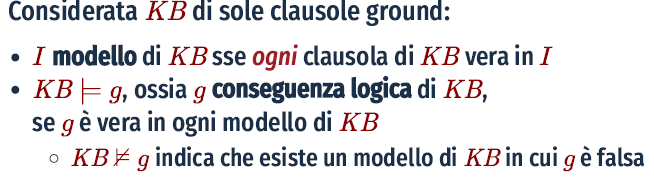
Description automatically generated with medium confidence

* E  sono due funzioni a tutti gli effetti.
* Associa ad un nome di costante C un individuo del dominio, attraverso la funzione phi associamo agli individui un nome.
* L’ Associazione pi greco associa ad ogni simbolo di predicato ovvero ad un nome scelto per un predicato n-ario una funzione booleana(assegna ad ogni simbolo un intera funzione booleana) il cui dominio è Dn ovvero l’insieme delle tuple di n elementi formate da individui del dominio D (quindi una tupla formata da n individui) e a questa tupla associa un valore booleano true o false.

SEMANTICA DEGLI ATOMI GROUND

Text

Description automatically generated with low confidence

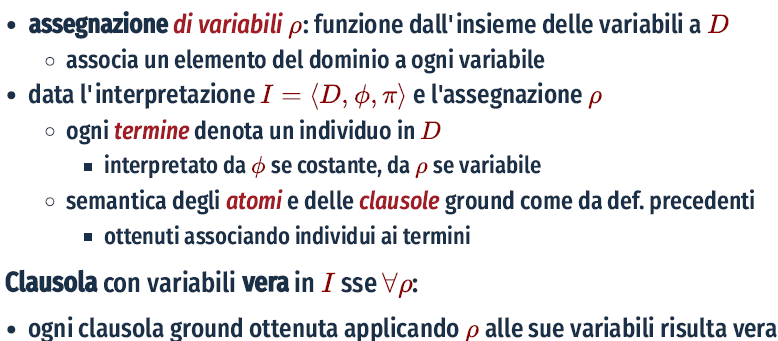


Una clausola ground risulta nell’interpretazione intesa:

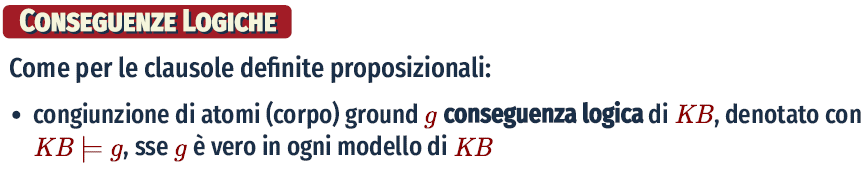
* FALSA se la testa è falsa e il corpo è vero oppure se il corpo è vuoto(poiché corrisponde a vero)
* VERA in tutti gli altri casi (come mostrato dalla tabella di verità dell’implicazione)

INTERPRETAZIONE DELLE VARIABILI

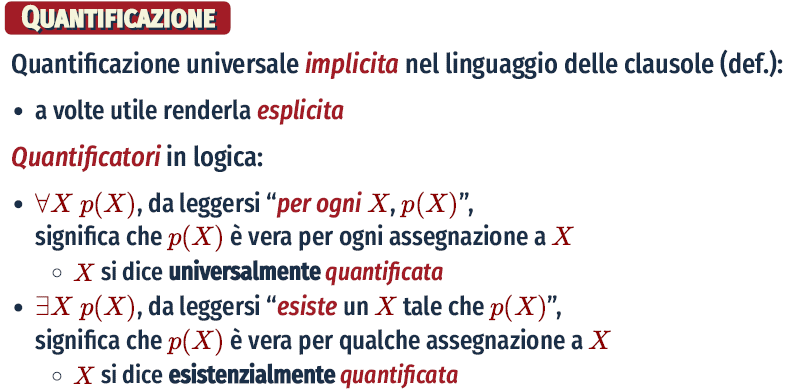
Le variabili vanno interpretate come UNIVERSALMENTE QUATIFICATE, per cui se una variabile compare in un atomo affinché questo sia vero deve essere vero per tutte le variabili ovvero per tutte le sostituzioni che saranno fatte alla variabile in questione con gli individui del dominio. Per introdurre il concetto dell’interpretazione delle variabili necessitiamo introdurre una nuova funzione di assegnazione, la quale andrà ad assegnare alle variabili un elemento del dominio ovvero un individuo.



Da qui si evince la generalità dei predicati del primo ordine in quanto nel caso di una clausola con variabili per essere vera deve essere per ogni elemento del dominio che viene ad essere sostituito alla variabile.



Qui in datalog ovvero con la logica dei predicati il passo significativo che permette di aumentare la potenza del ragionamento sono di fatto le variabili, poiché in caso di atomi ground che contengano solo costanti il ragionamento per provare l’atomo non si discosta da quello visto per il calcolo proposizionale, mentre con l’uso delle variabili adesso la dimostrazione deve essere generalizzata per tutti gli individui che possono prendere i posto della variabile tramite costanti.



Porre attenzione al fatto che presa una regola per le variabili che abbiamo nella testa vale un quantificatore universale mentre in quelle del corpo vale un quantificatore esistenziale e questo è dovuto alle equivalenze logiche come mostrato:

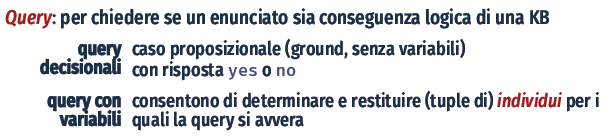
Graphical user interface, text

Description automatically generated

SOSTITUZIONI E DIMOSTRAZIONI (teoria delle dimostrazioni

per le macchine proof theory)

Per quanto riguarda la dimostrazione di clausole ground e quindi che hanno costanti gli algoritmi visti per la proof theory delle clausole definite proposizionali può essere riutilizzato anche in questo caso. Le cose si complicano quando la macchina dovrà dimostrare clausole contenenti variabili. Questo perché per ogni variabile libera la clausola deve essere dimostrata per ogni istanza assunta dalla variabile.



Di fatto andando a sottoporre una query ground senza variabili staremo ragionando come con la logica proposizionale e quindi staremo chiedendo se un atomo è vero oppure non è possibile dimostralo come vero, usando invece una query contenente variabili andiamo a sfruttare la potenza del ragionamento generale con il linguaggio dei predicati del primo ordine e per tale ragione una query con variabili ci permetterà di ritrovare tutte le tuple di individui che rendono vero l’atomo ovvero la query. Quindi una tale query è simile ad una query Sql poiché siamo intenzionati a sapere per quali valori si avvera la query dove i valori sono gli individui. Unna query con sole variabili permette di ritrovare tutte le istanziazioni ground che risultano essere conseguenza logica della base di conoscenza.

**UN ISTANZA DI UNA CLAUSOLA** è una clausola ove alla variabile viene ad essere sostituito un termine(che sarà un elemento del dominio degli individui).

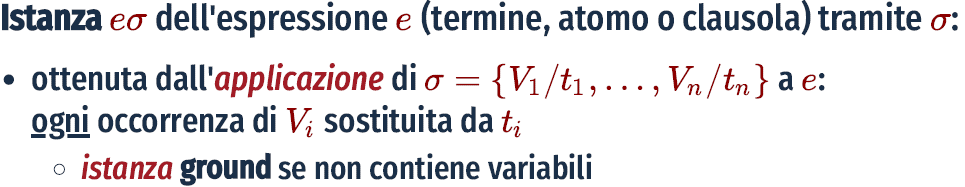
Un istanza viene ad essere definita per sostituzione, ove la **sostituzione** può essere vista come una funzione che alle variabili va ad associare un termine, il quale andrà a sostituire la variabile nella clausola.

**Graphical user interface, text

Description automatically generated with medium confidence**

La sostituzione viene detta in forma normale quando una variabile X è sostituita con un termine ed essa non ricompare a causa di un'altra variabile Y che per il processo di sostituzione viene sostituita a sua volta da un termine che può essere o una variabile o una costante venga ad essere proprio sostituita con X.

Adottiamo una scrittura in notazione post-fissa per indicare il processo di istanzazzione di un’espressione indicata con e (la quale potrà essere un termine, atomo o clausola) al quale viene applicato sigma σ (ovvero il processo di sostituzione da effettuare per ottenere un istanza).



Vediamo alcuni esempi di questo processo che metteranno in evidenza come un atomo complesso ovvero un atomo nella sintassi Datalog per cui della forma p(X) ove X variabile viene ad essere sostituito tramite il processo di sostituzione sigma descritto con le sole parentesi graffe del tipo {X/b, Y/S} :

Text, letter

Description automatically generated

UNIFICAZIONE

Con l’unificazione quello che abbiamo in mente è capire se due espressioni possano essere parificate ovvero se due espressioni che a primo sguardo sembrano essere differenti a livello sintattico al livello semantico vanno ad individuare e mappare lo stesso individuo. E per tale ragioni se si possono parificare ovvero unificare le espressioni sono equivalenti. L’unificazione rende le due espressioni se unificabili uguali dal punto di vista sintattico dopo l’applicazione dell’unificatore che altro non è che una specifica sostituzione che rende le espressioni uguali a livello sintattico.

Adesso ci rimane una complicazione da risolvere la quale riguarda l’algoritmo top-down, in particolare nelle clausole definite proposizionali andavamo a selezionare un atomo nella clausola di risposta che fosse la testa di una certa clausola all’interno della base di conoscenza e con queste scelte si provava a derivare la query data fino ad ottenere yes seguito da un corpo vuoto. Il problema emerge proprio qui ovvero nel caso delle clausole definite, l’atomo è rappresentato da un letterale ed è definito come un individui unico, cosa che qui con l’introduzione delle variabili non è più vera e si complica con il processo di sostituzione. Per risolvere questo problema nasce il processo di unificazione.

Il processo di UNIFICAZIONE è tale da fare in modo che a due espressioni diverse sulle quale agisce lo stesso processo di sostituzione portino ad istanze equivalenti ovvero identiche come mostrato:

A picture containing text

Description automatically generated

Ciò significa che più espressioni potrebbero dare vita alla medesima istanza.

Text

Description automatically generated

Ci sono degli unificatori che risultano essere più generali nelle sostituzioni che applicano.

Text

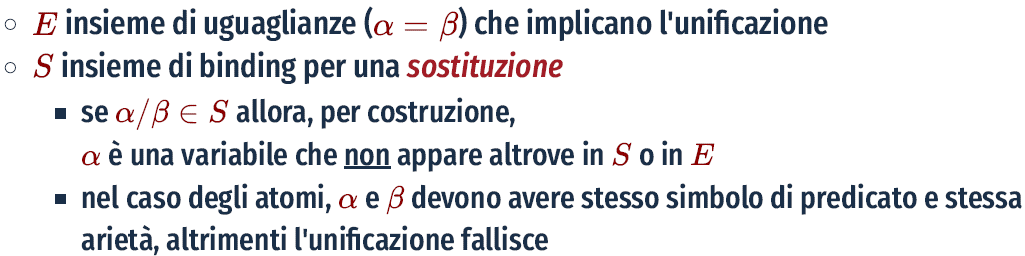
Description automatically generated with medium confidence

Vediamo un esempio di renaming grazie ad un MGU:

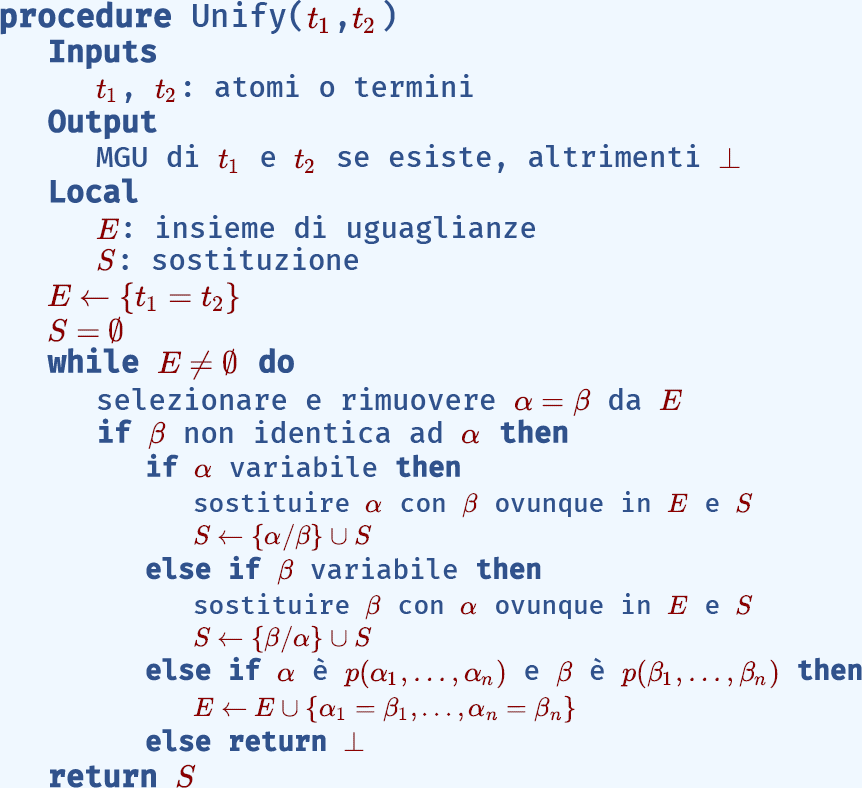
Letter

Description automatically generated with low confidence

Per trovare l’MGU tra 2 termini o due atomi è possibile utilizzare un algoritmo che in caso di insuccesso riporterà il simbolo bottom  , altrimenti ci fornirà MGU. L’algoritmo lavora su di un insieme S che è l’insieme contenente i binding e in partiolare



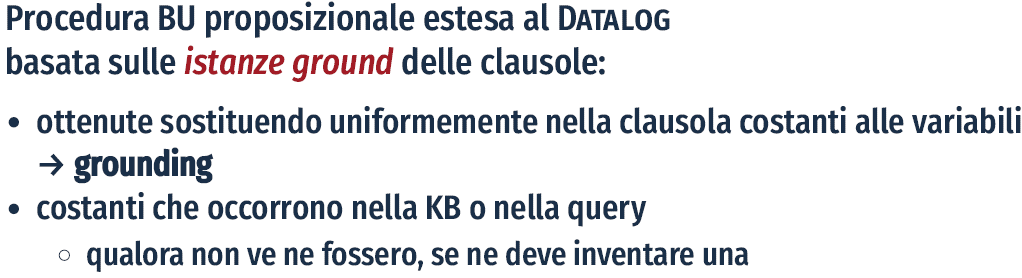
ALGORITMO UNIFY che effettua l’unificazione di 2 termini:



L’obiettivo target che viene a essere impostato nell’algoritmo è E 🡨{t1 = t2} ovvero l’uguaglianza tra i 2 termini(ove i termini rappresentano atomi del linguaggio Datalog per cui un predicato con una certa arietà) e all’inizio l’insieme dei binding S è vuoto.

Una volta entrato nel ciclo se beta e alfa risultano diversi e sono entrambi variabili si esce immediatamente dal ciclo e si ritorna il bottom poiché non è possibile unificare le due espressioni. Mentre in caso di diversità vengono ad essere applicate le sostituzioni come indicate nei rami else e l’insieme dei binding aumenterà l’algoritmo terminerà con successo ovvero dicendoci che le due espressioni possono essere unificate se e solo se l’insieme E risulterà vuoto alla fine della procedura dell’algoritmo se invece sarà diverso da vuoto allora il processo non sarà riuscito ad unificare le due espressioni le quali risulteranno non unificabili.

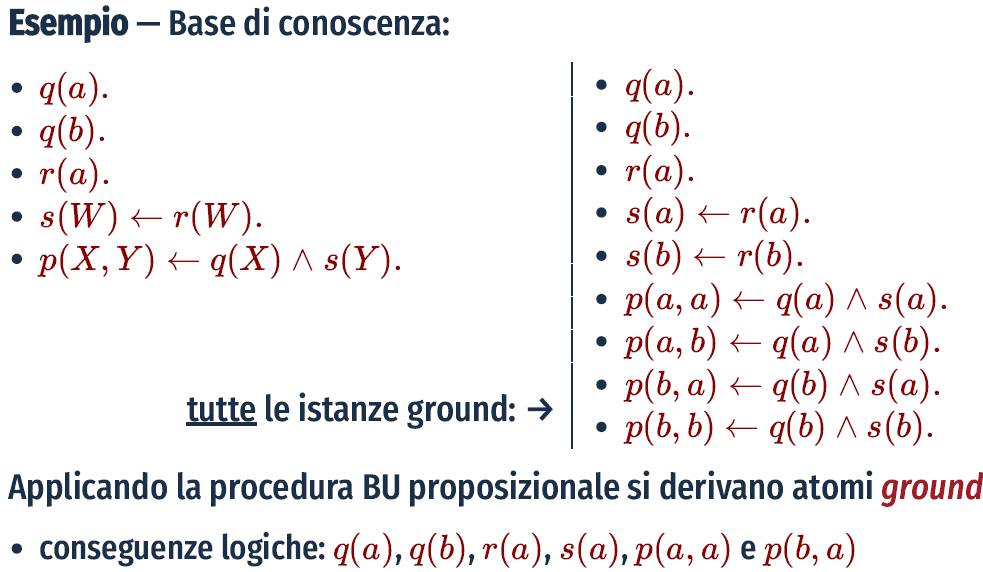
Dopo aver affrontato queste problematiche nel linguaggio dei predicati del primo ordine vediamo gli algoritmi di proof theory ovvero:



Il primo passo affrontato dalla procedura di BOTTOM UP è quello di andare a groundizzare tutte le clausole presenti nella KB andando così a definire tutte le istanziazioni possibili e una volta ottenute le istanze ground procederà a derivare le conseguenze logiche nella modalità classica della BU vista per il linguaggio proposizionale.

La groundizzazione porta ad avere un KB groundizzata ovvero clausole istanziate in tutte le manieri possibili con le costanti che si conoscono ovvero quelle presenti nella KB, a questo punto applicando la versione di BU delle clausole proposizionali si potranno derivare tutte le possibili conseguenze logiche. Nel caso particolare in cui la KB di base ovvero prima di applicare la groundizzazione non contiene alcuna costante, ma solo variabili, allora in tal caso il sistema inventerà una costante dummy che servirà per ragionare in generale sugli individui con lo stesso meccanismo di prima, questa volta la groundizzazione vedrà la sostituzione delle variabili con l’unica costante dummy utilizzata dal sistema.

Esempio dell’applicazione dell’algoritmo BU:



Ci potrebbero essere delle particolari KB nelle quali le verità e quindi i fatti o regole sono dei predicati con variabili, ma non abbiamo nessuna regola semplice del tipo testa e copro vuoto ove il predicato preveda una costante come viene mostrato dal seguente esempio:

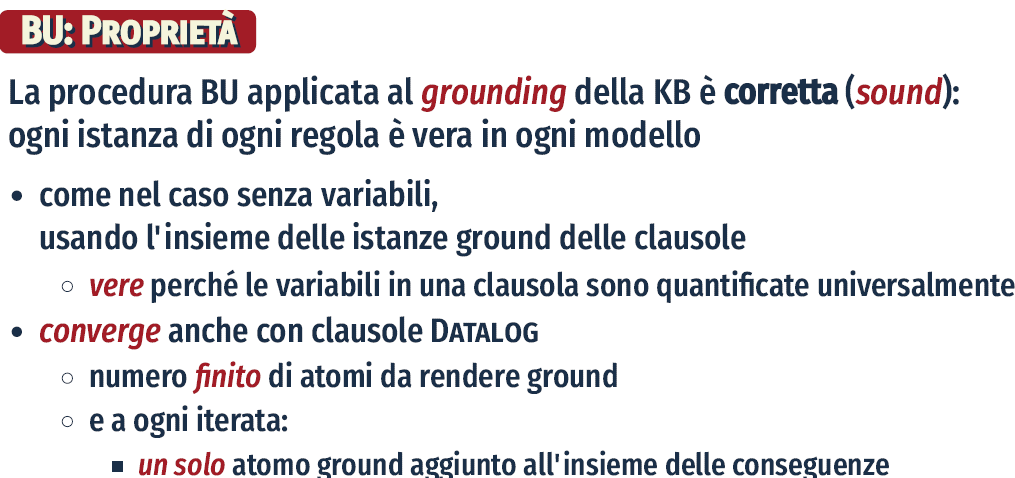
Text

Description automatically generated

In questo caso l’algoritmo per poter funzionare correttamente utilizza una costante c fittizia che utilizzerà per applicare il processo di dimostrazione BU.

Graphical user interface, text, application

Description automatically generated





Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

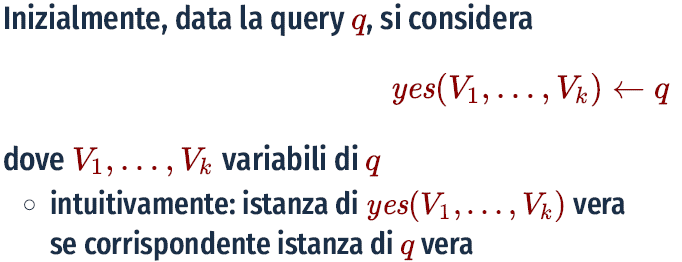
Una volta che la KB è stata groundizzata si può dimostrare che la BU agente su tale KB groundizzata risulta una procedura sound, adesso per verificare che sia anche una procedura corretta il che ci serve per far scattare l’equivalenza tra quello dimostrato dal reasoner che valga come conseguenza logica nel dominio e per farlo si utilizza una speciale interpretazione che prende il nome di interpretazione di Hebrand ove le costanti vengono assunte come i soli individui presenti nel reale dominio, ovvero si assume che tali costanti siano gli individui del dominio.

Nel caso della procedura TD andiamo a lavorare come visto per le clausole definite quindi procedendo dalla risposta con testa yes fino a derivare una regola con testa yes e corpo vuoto, la differenza qui sta nel fatto di gestire gli atomi che al loro interno possono avere dei termini e per gestirli applichiamo l’unificazione e poi dobbiamo gestire la sostituzione dei tk nella clausola generalizzata:

Text

Description automatically generated with medium confidence

La sostituzione dei tk presenti nel predicato yes che si dovrà dimostrare saranno sostituiti dai termini presenti nel corpo che risultano essere l’effettiva query sottoposta al sistema in partenza.



A picture containing company name

Description automatically generated

Text

Description automatically generated with low confidence

Questo passo del TD è importantissimo poiché abbiamo la selezioni di un atomo del corpo della clausola di risposta ove ancora una volta abbiamo una scelta del tipo don’t care e una volta scelto lo sostituiremo con un atomo presente tra le regole della KB la cui testa è unificabile con l’atomo selezionato una volta trovato avremo che la successiva clausola risolvente sarà uguale a quella iniziale con la sostituzione dell’atomo scelto con il corpo della clausola che è possibile unificare e poiché l’unificazione sarà data da una sostituzione facciamo agire sulla nuova risposta “risolvente” la sostituzione che avverava l’unificazione.

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

A picture containing text

Description automatically generated

Il DATALOG è meno complesso del Prolog e la differenza sta nel uso dei simboli di funzione che nel datalog non sono utilizzati mentre nel prolog si, inoltre la differenza sostanziale a livello del ragionamento e che con il datalog un problema affrontato è sempre decidibile, mentre andando ad aggiungerli i simboli di funzione e ottenendo il prolog innanzitutto otteniamo un programma logico turing completo e in seconda istanza il ragionamento si sposta da decidibile a semi—decidibile. Quindi con i simboli di funzione otteniamo un maggiore potere espressivo a livello semantico nel ragionamento, ma lo paghiamo con la semi-decidibilità dei problemi che affrontiamo ove se non prendiamo le dovute precauzioni le procedure di dimostrazione potrebbero entrare in loop infiniti senza fornirci la risposta.

Una Base di Conoscenza che ammetta simboli di funzioni è un vero e proprio programma logico ovvero un linguaggio di programmazione turing completo.

SIMBOLI DI FUNZIONI

Con la logica dei predicati del primo ordine abbiamo aperto una nuova rappresentazione del problema e un nuovo modo di ragionare su di essi, tuttavia, con le rappresentazioni viste con la logica siamo rimasti fino ad ora nell’ambito di problemi decidibili, adesso affronteremo problemi che sforano nell’ambito della non decidibilità e semi-decidibilità in tal caso gli algoritmi in funzione sulle macchine divergono nel processo senza fornire soluzione.

I simboli di funzioni servono a lavorare su un numero potenzialmente infinito di individui.

Le funzioni serviranno a mappare gli individui e in particolare a gestire un numero potenzialmente infinito di individui e quindi un dominio di individui infinito. Le funzioni sono simili ai predicati della logica del primo ordine visti per il Datalog, ma a differenza dei predicati le funzioni hanno un ordine inferiore poiché esse permettono solo di mappare gli individui.

La differenza tra la logica del 1°ordine e quella del 2°ordine o di ordini superiori è che queste ultime ammettono la quantificazione dei predicati definiti su relazioni del primo ordine, ovvero mentre nel 1°ordine i quantificatori sono usati esclusivamente sulle variabili libere qui nel 2°ordine i quantificatori possono essere associati ai predicati definendo semantiche più complesse, questo permette una maggiore espressività tuttavia queste logiche del 2° ordine o superiori non sono utilizzate perché in questo ambito i problemi sono **INDECIDIBILI** e non possono essere messi in correlazione con una macchina di Turing, mentre con i predicati del primo ordine e quindi con la sintassi e semantica del Datalog + i simboli di funzione siamo in ambito di SEMI-DECIDIBILITÀ ove i problemi sono ancora trattabili.

Datalog linguaggio finito che permette di lavorare su problemi decidibili.

Datalog + simboli di funzioni linguaggio non finito poiché permette la ricorsione infinita delle funzioni risulta essere semi-decidibili e questo linguaggio è quello dei predicati del primo ordine.

Datalog + simboli di funzione + ulteriori assunzioni = prolog.

La forza espressiva della logica del secondo ordine sta nel poter sostituire i predicati con variabili andando quindi ad effettuare unn ragionamento quantitativo e generale sui predicati e quindi sulle relazioni del mondo che stiamo rappresentando, questa forza espressiva ci permette di definire e ragionare sulla relazione di transitività la quale è una proprietà dei predicati. Nella logica del secondo ordine si sfocia nell’indecidibilità.

UGUAGLIANZA

Si potrebbero avere ora con le funzioni, costanti e termini diversi che tuttavia identificano e si riferiscono ad uno stesso individuo del dominio. Ora riuscire a istruire una macchina affinché capisca questo concetto di uguaglianza non è affatto banale e per farlo ci serve un predicato particolare che chiameremo predicato speciale di uguaglianza il quale ha il simbolo di = ma sarà un simbolo di predicato.

Text

Description automatically generated with medium confidence

Bisogna far capire al computer che due individui potrebbero essere simili o identici, ma non uguali ove uguaglianza sta ad indicare che sono lo stesso individuo(oggetto).

Per poter usare gli atomi con il predicato di uguaglianza e di seguito poter derivare uguaglianza ovvero clausole con atomi di uguaglianza in testa necessitiamo affrontare questa necessità che ricordiamo ci permetterà di spiegare al calcolatore il significato di simili/identico e quello di è proprio lo stesso uguale, usiamo due strategie:

1. ASSIOMATIZZAZIONE 🡪 ovvero andremo a trattare l’uguaglianza come qualunque altro predicato
2. Procedure speciali d’inferenza per l’uguaglianza.

Con l’assiomatizzazione dell’uguaglianza andiamo ad usare clausole del primo ordine per descrivere l’uguaglianza e per fare questo dobbiamo codificare sotto forma di regole nella KB le 3 proprietà di cui gode l’uguaglianza ovvero:

1. Riflessività
2. Simmetria
3. Transitività

Esempio delle 3 proprietà assiomatizzate nella KB:

Shape

Description automatically generated with low confidence

Queste 3 proprietà esplicitate sotto forma di clausole nella KB permettono alla macchina di lavorare con il predicato di uguaglianza come se fosse un qualunque altro predicato poiché definito da delle regole, tuttavia questo non risulta sufficiente a risolvere il nostro problema di partenza poiché con le funzioni n-arie che vanno ad agire su dei termini o su altre funzioni che agiscono su altri termini, ovvero abbiamo un concetto di ricorsione delle funzioni a più livelli ove abbiamo funzioni che si applicano a funzioni e così via sino ad avere un termini sul quale si applica la catena di funzioni, c’è bisogno di spiegare il concetto di uguaglianza tra un termine e la funzione applicata al termine e quindi tra funzioni applicate a termini che restituiranno lo stesso individuo. Per farlo useremo degli schemi di assiomi che saranno associati a delle precise funzioni con una certa n-arietà ciò significa che avremo tanti schemi di funzione quante sono le funzioni con le quali stiamo lavorando(uno schema per ogni funzione) e all’interno dello schema definiremo cosa significherà il simbolo di uguale il quale potrebbe variare a seconda del predicato.

Text

Description automatically generated

Nel primo esempio abbiamo un atomo che è formato dal predicato = ovvero di uguaglianza che agisce sui due termini di sx e dx che sono funzioni n-arie, quindi tutto quello che vediamo prima del simbolo di implicazione costituisce un atomo la cui verità è verificata quando i congiunti del corpo ovvero X1=Y1 sino a Xn=Yn (gli argomenti sono uguali a coppie ordinate) sono tutti verificati ovvero risultano veri tramite le 3 proprietà dell’uguaglianza che abbiamo codificato sotto forma di clausole o regole della KB, questo ancora non basta poiché dobbiamo spiegare cosa significa l’uguaglianza quando i termini sono argomenti di predicati come viene mostrato del secondo esempio ove per farlo abbiamo una testa costituita da un atomo che risulta un predicato n-ario e come corpo dell’implicazione abbiamo un congiunto con lo stesso predicato, ma applicato a diversi termini e le uguaglianze tra i termini che compaiono come argomenti nel predicato di testa e quelli che compaiono come argomenti nel predicato del corpo rispettandone l’ordinamento(ovvero uguaglianza tra coppie ordinate).

Questi visti e dettagliati sono tutti gli assiomi da aggiungere ad una KB affinché sia possibile per l’agente intelligente riconoscere e capire il concetto di uguaglianza. Questa tecnica ovvero l’assiomatizzazione dell’uguaglianza risulta una tecnica pulita e solida.

Esempi di schemi su funzioni e predicati per assiomatizzare l’uguaglianza:

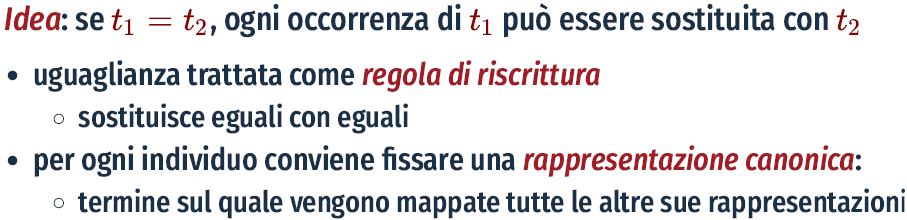
Text, letter

Description automatically generated

Tal punto di vista teorico l’assiomatizzazione è corretta tuttavia dal punto di vista pratico ovvero per le macchina risulta meno efficiente e la terminazione non risulta garantita.

L’altra alternativa è quella che prevede l’uso di procedure speciali di ragionamento. In questo caso l’idea è che, se nel ragionamento si incontra una uguaglianza allora il normale processo di ragionamento sulla KB si arresta è si intraprende una procedura a parte per trattare l’uguaglianza.

Uno dei meccanismi implementati in queste procedure speciali va a vedere se i due termini coinvolti nell’uguaglianza sono sostituibili con un potenziale individuo di modo che se c’è l’esistenza di tale individuo allora la procedura si occuperà di riscrivere quei termini con un termine che vada a eguagliare i precedenti indicando lo specifico individuo. Questa procedura prende il nome di PARAMODULAZIONE:



La procedura di paramodulazione va ad abbattere tutti i termini che rappresentano uno stesso individuo tenendo traccia di tali termini e mappandoli ad un unico termine che designerà l’individuo comune al quale si riferiscono.

UNIQUE NAMES ASSUMPTION UNA

Assunzione che viene fatta spesso nel mondo dei database(basi di dati), viene anche fatta nel Datalog ove quando aggiungiamo le funzioni otteniamo il Prolog (ovvero un vero e proprio linguaggio di programmazione).

Text

Description automatically generated

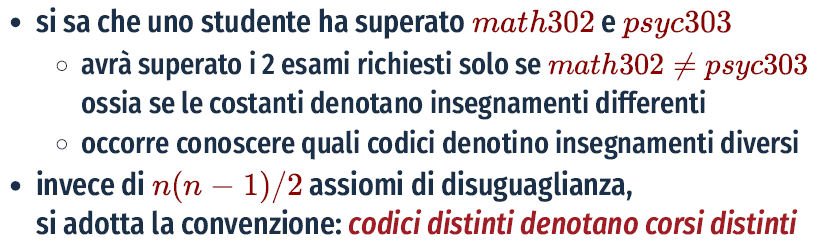




Text

Description automatically generated

Vediamo un esempio di un DataBase di n studenti in cui ogniuno ha due insegnamenti a scelta nel piano di studi:

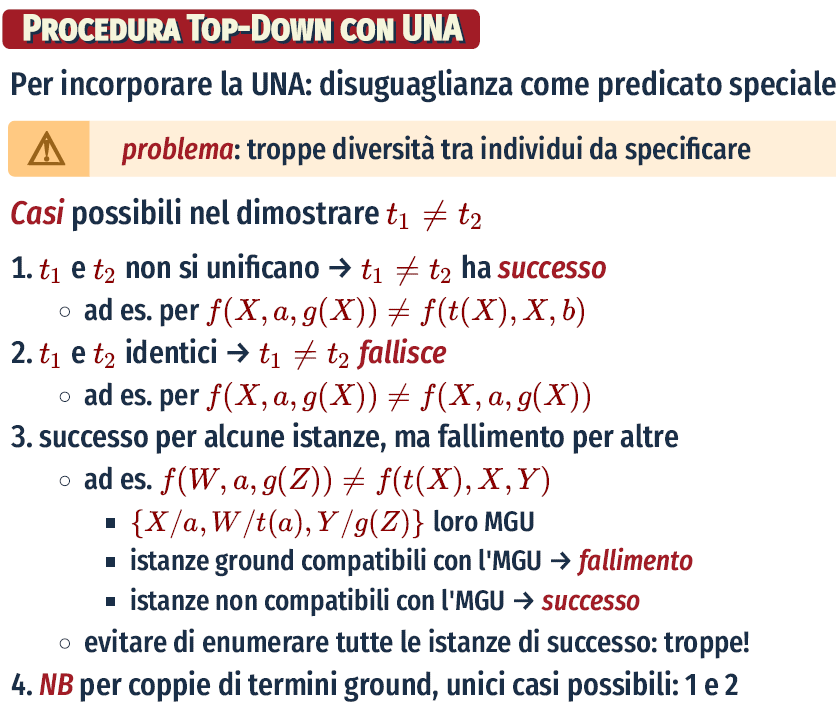


Per fare in modo di ottenere la Unique Names Assumption assiomatizziamo tramite un processo simile a quello visto per l’uguaglianza il concento di disuguaglianza come segue:

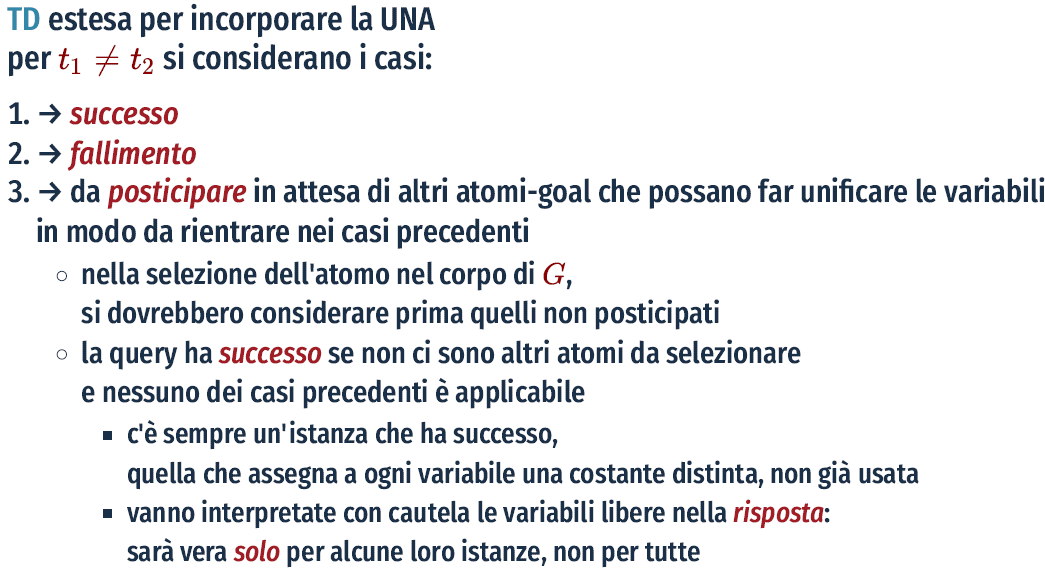
Text, letter

Description automatically generated

Nel caso ora dopo aver capito a cosa serva la UNA utilizzassimo la procedura TD con UNA la quale procedura già con le funzioni abbiamo visto soffre di parecchie problematiche, avremo anche in questo caso per alcune dimostrazione problemi dati dalle troppe diversità tra gli individui che possono portare l’algoritmo a fallire, rendendolo non-deterministico.



Si cerca di risolvere il problema modificando l’algoritmo TD con UNA e posticipando atomi parecchio complessi di modo che in seguito sia più semplice provarli.



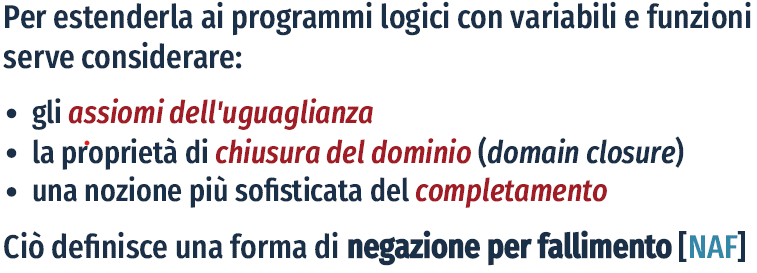
Il differimento dei goal rende la dimostrazione più semplice da dimostrare.

ASSUNZIONE DI CONOSCENZA COMPLETA

(detta anche mondo chiuso)

Per codificare l’assunzione di conoscenza per le basi di conoscenza relazionale (ovvero basate su predicati del primo ordine) ricorriamo ad una forma di negazione per fallimento NAF. Questa ci permetterà di dire che se un enunciato non è conseguenza logica della KB allora è falso.

Per estendere l’assunzione di conoscenza completa ai programmi logici serve:



Quando adottiamo la UNA sotto l’ipotesi di assunzione di completa conoscenza ovvero di mondo chiuso oltre all’assiomatizzazione dell’uguaglianza e della disuguaglianza ci servono le clausole in FORMA NORMALE DI CLARK:

Diagram

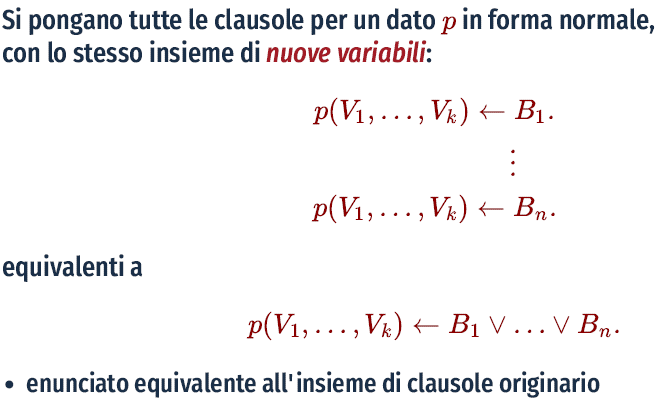
Description automatically generated with low confidence

La formulazione di Clark prevede, la riscrittura di una clausola ove la testa è della forma predicato k-ario con k termini e il corpo un qualsiasi corpo in una nuova scrittura ove i termini del predicato vengono ad essere sostituiti con k variabili e nel corpo vengono posti in maniera esplicita dei quantificatori esistenziali per ogni uno delle vecchie variabili del predicato ed infine si mette il congiunto di uguaglianza tra le k nuove variabili utilizzate e i k termini sostituiti in congiunzione con B il corpo della clausola originale.

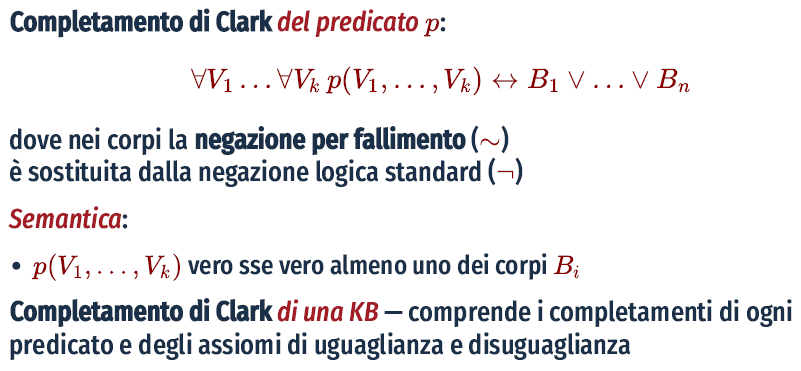
Text

Description automatically generated

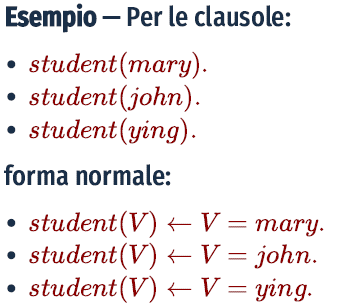
Una volta effettuata questa riscrittura in forma normale di Clark per ogni regola presente nella KB che definisce lo stesso simbolo di predicato p, queste riscritture saranno tutte equivalenti ad una nuova clausola che prevede come testa il predicato riscritto con le k nuove variabili e a destra nel corpo la disgiunzione di tutti i corpi che definiscono quel predicato come segue nell’esempio:

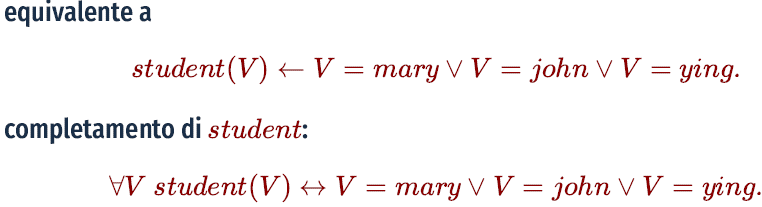


Con il completamento di Clark stiamo dicendo che non esistono altre possibilità di dimostrare il predicato riscritto se non con la disgiunzione dei corpi trovati e questo permette di avere conoscenza completa sul mondo.



Vediamo un esempio di riscrittura per portare in forma normale di Clark delle clausole:





La procedura TD con la NAF diventa una procedura incompleta e incorretta (ovvero no sound no complete).

Text

Description automatically generated

Nel caso la procedura fallisce con la NAF è possibile usare procedure più avanzate che si basano sull’analisi dei casi in cui il goal è fallito.

