**RAPPRESENTAZIONE E RAGIONAMENTO PROPOSIZIONALE**

Andremo a lavorare con KB ovvero basi di conoscenza che saranno la modalità con cui l’ingegnere va a rappresentare il mondo per un sistema basato su conoscenza. Quindi andrà a rappresentare gli stati del mondo tramite delle proposizioni, le quali diventano vere per un determinato stato del mondo mentre per altre interpretazioni sono false. Una volta che il sistema (o l’agente) avrà una rappresentazione del mondo sarà capace di rispondere a domande poste tramite un meccanismo di inferenza logica.

Le osservazione sullo stato corrente del mondo più la background knowledge fornita dalla KB consentiranno al sistema di rispondere alle nostre domande poste.

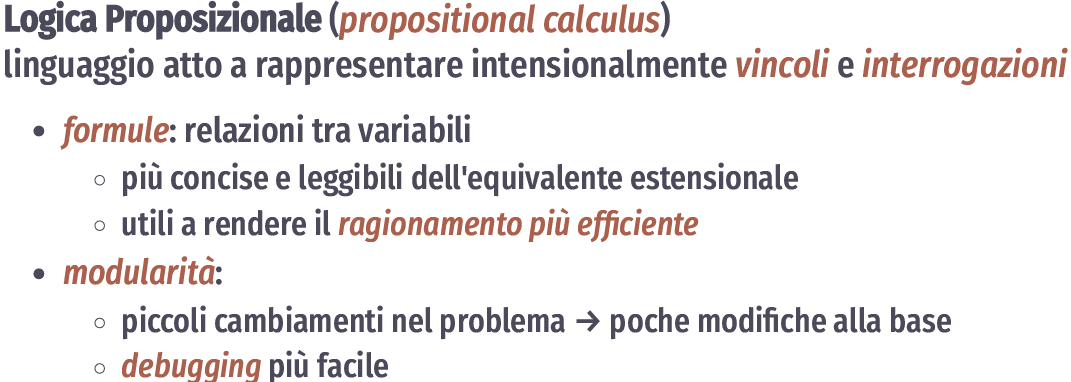
FORMALISMO DELLE PROPOSIZIONI

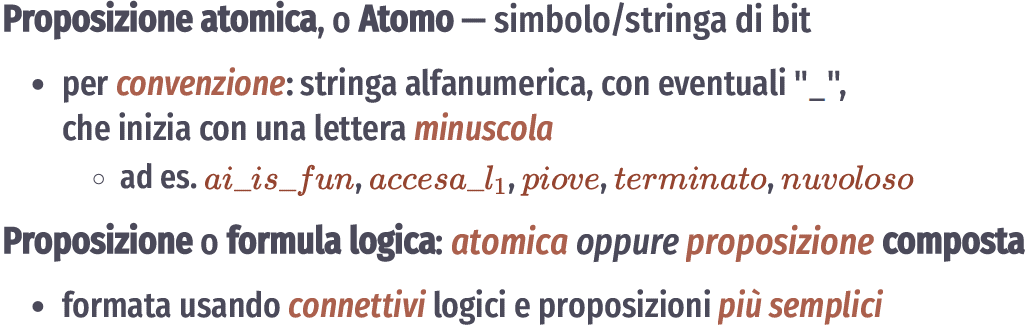
Concetto di proposizione: esso sarà inteso come un enunciato che riguarda il mondo. Quindi una proposizione risulta essere una assegnazione, la quale avrà un grado di verità e su queste proposizioni saranno definiti dei vincoli(similmente a quanto visto con i CSP).

Le proposizioni saranno espresse o in forma estensionale o in forma intensionale(attraverso formule). Esse sono frasi o, meglio, enunciati (queste possono essere viste come una forma di vincolo in maniera intensionale).

La logica proposizionale è il linguaggio più semplice che ci permetterà di rappresentare intensionalmente sia i vincoli che le interrogazioni da passare al sistema per avere una risposta.

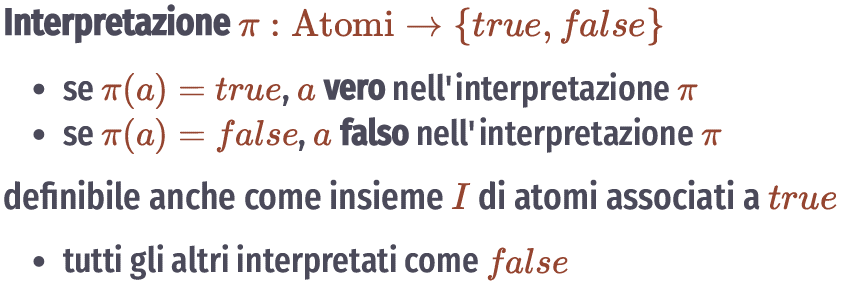
Questa rappresentazione facilità la rappresentazione di una base di conoscenza dal punto ingegneristico, inoltre, il ragionamento dell’agente tramite la logica proposizionale sarà molto vicino al ragionamento umano e quindi sarà più semplice da capire.





Usiamo le lettere minuscole per le proposizioni poiché per le variabili invece usiamo le maiuscole.

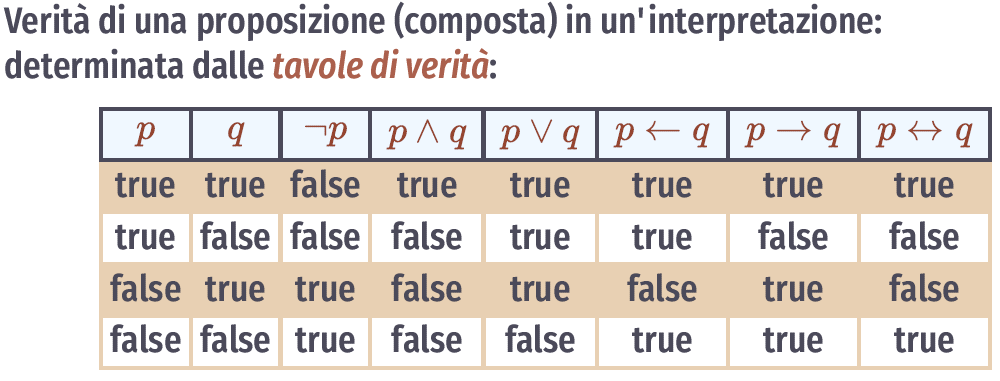
Bisogna attribuire una semantica agli atomi e alle proposizioni che abbiamo formato nella base di conoscenza. L’attribuzione di una specifica semantica ad un atomo ci permette di rappresentare in un determinato modo lo stato del mondo e quindi di modellarlo a seconda delle nostre esigenze. Il processo di interpretazione degli atomi è estremamente importante ed è portato a termine da un esperto del dominio.



Quindi le due possibilità per associare una determinata interpretazione agli atomi sono:

1. La funzione pi-greco che ad un atomo associa il valore di verità
2. Usare un insieme I (sta per interpretazione) di atomi, i quali sono positivi. I restanti atomi al di fuori di tale insieme sono considerati come negativi.

Per attribuire significato agli atomi e alle proposizioni e in seguito codificare le informazioni, ovvero la conoscenza che si vuole rappresentare si utilizzano le tavole di verità che saranno utilizzate anche nel processo di debug.

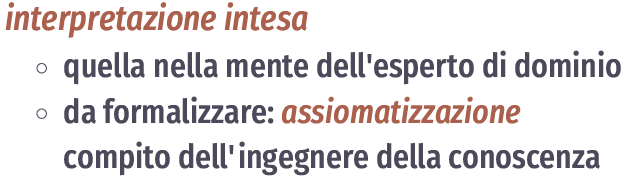


Con la lettera maiuscola I indichiamo l’interpretazione delle proposizioni della base di conoscenza.

Ex falso quodlibet [ o meglio: ex falso sequitur quodlibet]

BASE DI CONOSCENZA KB: è un insieme di proposizioni complicate a piacere che prendono il nome di assiomi, le chiamiamo assiomi perché queste proposizioni le diamo per vere e quindi non le dobbiamo dimostrare(o meglio è l0automa che non dovrà dimostrarle nel processo del suo ragionamento sulla conoscenza della KB), esse rappresentano la nostra background knowledge data per vera. Attraverso questi assiomi gli ingegneri della conoscenza codificano le regole di un mondo che si vuole rappresentare e quello che accade in esso dando delle interpretazioni. Gli assiomi della KB servono affinché un automa (o sistema intelligente) basandosi su di esse fornisca altri fatti che sono veri nel mondo e che quindi aggiunga nuova conoscenza a quella di partenza.

**Il processo con cui l’ingegnere costruisce una base di conoscenza prede il nome di assiomatizzazione.**



Prima di insegnare ad una macchina come ragionare su della conoscenza per fornire risposte serve prima teorizzare il modello sul quale l’automa andrà a ragionare ed è per questo che iniziamo con l’affrontare la teoria dei modelli.

Un modello di KB è un interpretazione nella quale si danno per veri gli assiomi presenti nella KB, il modello servirà, per la base della teoria dei modelli e delle conseguenze logiche, a costruire o, meglio, ad inferire una conseguenza.

I modelli che rendono vera una base di conoscenza potrebbero essere più di uno, questo significa che è possibile avere più interpretazioni degli atomi della KB che risultano date per vere e che verificano come corretta la KB. Tra tutti questi modelli possibili l’ingegnere se lavora bene assiomatizzerà proprio uno di questi modelli affinché sia poi possibile dimostrare che una specifica proposizione che rappresenta uno degli obiettivi del sistema sia conseguenza logica della KB.

Graphical user interface, text, application, email

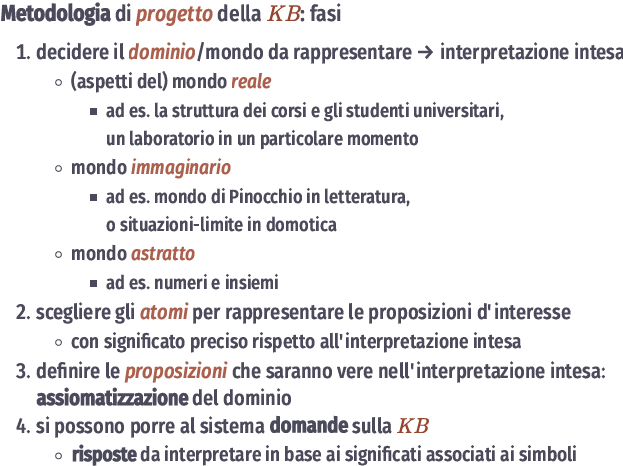
Description automatically generated

Quando g non è conseguenza logica della base di conoscenza allora ciò sta a significare che esiste almeno un modello in cui g non è vera.

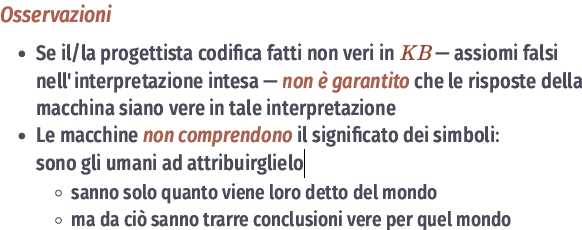
È importante ricordare che, se g non è conseguenza logica della KB ciò non significa che potremmo dedurne che not(g) sia conseguenza logica della KB. Il non essere conseguenza logica della KB per una proposizione vuol dire che di tale proposizione non possiamo stabilirne il grado di verità e quindi non possiamo né dire che sia vera né che sia falsa.

Un ingegnere deve progettare la KB in modo tale che sia gli assiomi della KB siano sempre veri nell’interpretazione intesa. Il simbolo |= non comporta né la computazione da parte del sistema né la dimostrazione per inferenza con cui andrà a ragionare l’agente. Il simbolo, quindi, specifica solo la verità di quello che consegue da certe assunzioni.

Nella costruzione di una KB è di vitale importanza la documentazione del progetto che registra e tiene traccia delle scelte fatte e del perché sono state adottate tali scelte, nonché tiene traccia del significato attribuito all’interpretazione intesa.

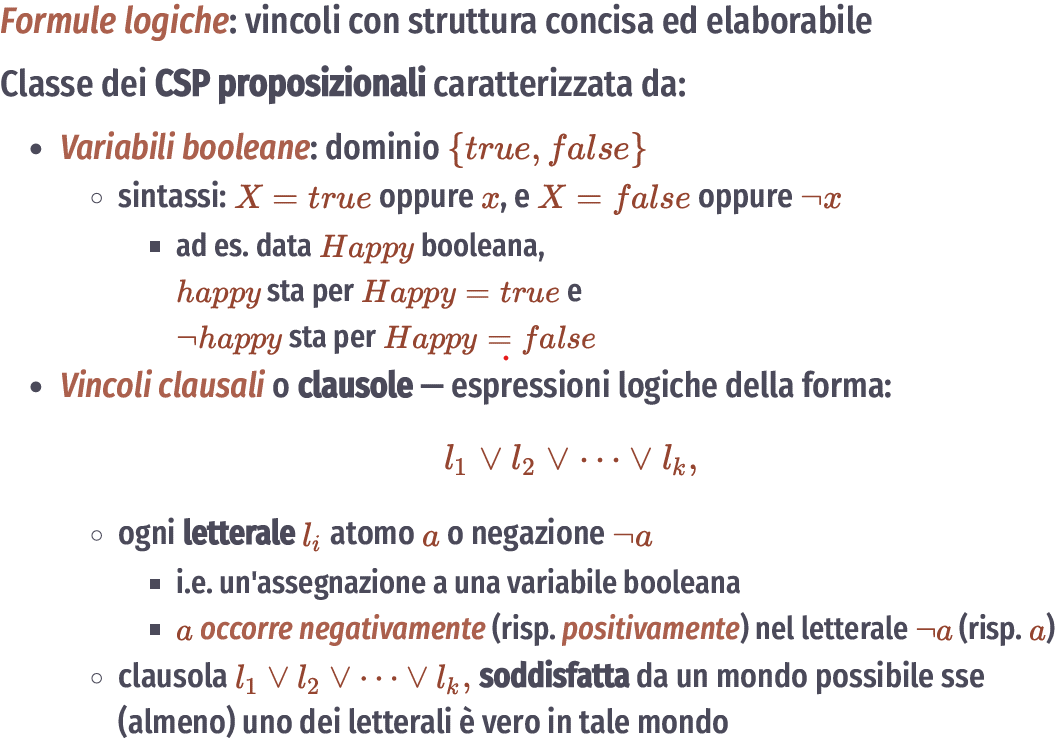


Quando si crea un vocabolario che specifica il significato dei simboli che usiamo questo vocabolario prende il nome particolare di ONTOLOGIA.



VINCOLI PROPOSIZIONALI

Una volta impostati i vincoli potremmo pensare di lavorare su di una Base di Conoscenza con degli algoritmi che risolvono i CSP proprio perché abbiamo a che fare con vincoli anche se sotto una nuova forma ovvero quella proposizionale, i vincoli saranno posti in forma booleana tramite delle clausole. L’implicazione equivale ad una disgiunzione e sarà questa equivalenza che sarà sfruttata dai vincoli che risulteranno dei letterali in forma disgiunta (legati dal simbolo della disgiunzione).

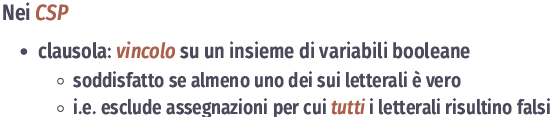


Un vincolo visto come clausola disgiuntiva è soddisfatto nel momento in cui la proposizione clausolare risulta vera.

Grazie a questo approccio proposizionale possiamo pensare ad un CSP sotto forma di vincoli e trasformarlo in forma clausale

CLAUSOLE DEFINITE PROPOSIZIONALI

Sono delle clausole particolari ove il primo letterale è positivo e tutti gli altri sono negativi.



Il seguente esempio mostra come sia possibile rappresentare delle variabili che si usavamo nei CSP come fattorizzazione del problema, in particolare variabili booleane e trasformarle in vincoli clausali:

Text

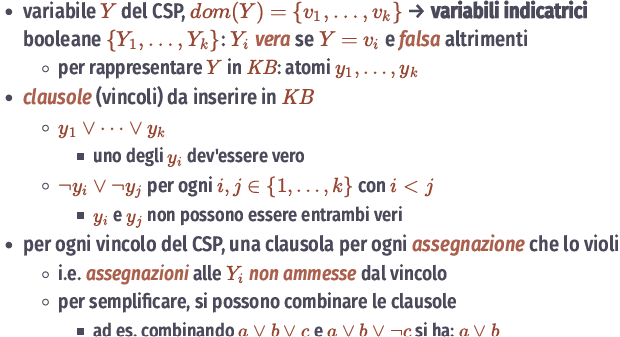
Description automatically generated with medium confidence

Si nota che le Variabili sono quelle scritte con l’iniziale maiuscola.

I CSP finiti possono sempre essere trasformati in un problema di soddisfacibilità di una KB, questa trasformazione ci permetterà di applicare degli algoritmi che ragionano su una KB per andare a risolvere quello che è un CSP trasformato.

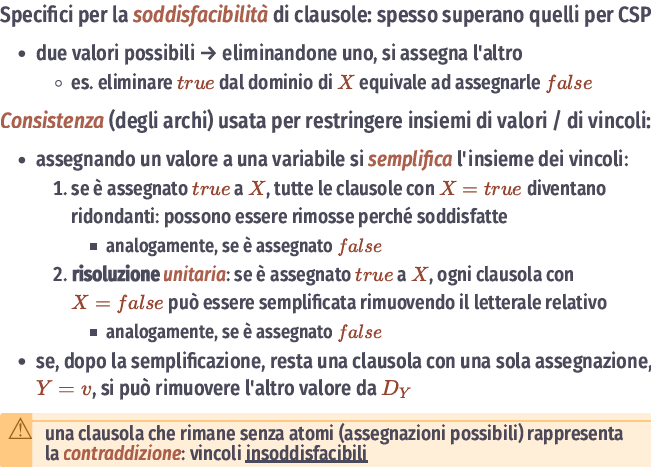
CONVERSIONE CSP 🡪 LOGICA PROPOSIZIONALE

Come possiamo vedere dal seguente schema una variabile Y che abbia un dominio dom(Y) il quale ha cardinalità k viene ad essere tradotta con k variabili indicatrici ogniuna delle quali mapperà un valore del dominio e se presente sarà a True altrimenti sarà a false. Infine, le variabili indicatrici così definite saranno mappate come atomi della KB.



[Non si fanno per l’esame]

ALGORITMI BASATI SU CONSISTENZA PER CLAUSOLE: sono particolari algoritmi che riescono a risolvere un CSP posto in logica proposizionale: [non sarà chiesto all’esame]



Algoritmo DPLL Davis-Putnam-Logemann-Loveland (da usare nel progetto se possibile): ideato nel 1962 è l’algoritmo[risolutore di problema di soddisfacibilità] più famoso per la risoluzione dei CSP in forma di proposizioni logiche esso si basa sulla riduzione dei domini tramite tecnica del PRUNING e riduzione dei vincoli, separazione dei domini ove possibile e assegnazione di letterali puri che altro non sono che letterali che assumono sempre un valore di verità che non cambia. Su questa tipologia di problemi è possibile utilizzare anche algoritmi di ricerca locale in quali convergono molto rapidamente dati i domini booleani delle variabili.

CLAUSOLE DEFINITE PROPOSIZIONALI

È un sotto linguaggio della logica proposizionale che permette di ottenere una rappresentazione della conoscenza senza ambiguità o incertezze. In particolare, la semantica della logica proposizionale non cambia, ma le proposizioni ammesse avranno un forma specifica.

La sintassi delle clausole definite proposizionali è quella che segue:

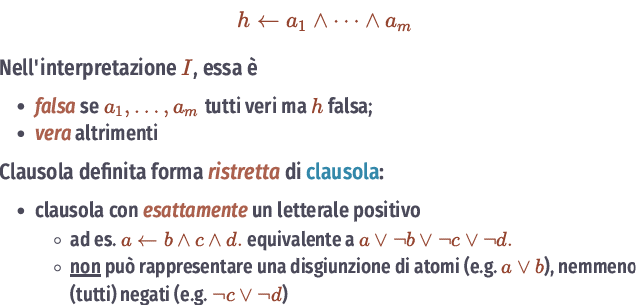
Text

Description automatically generated with medium confidence

Il corpo della clausola definita proposizionale altro non è che una disgiunzione di atomi negati infatti vale la seguente equivalenza logica:

h🡨a1 an ⬄ h ­­­­­­­­­¬ a1 ………­­­­­­­­­¬ an

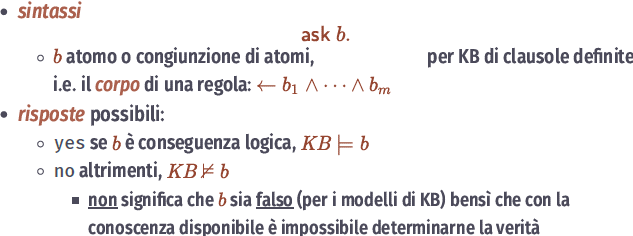
SEMANTICA DELLE CLAUSOLE DEFINITE PROPOSIZIONALI:



Una volta che una KB è stata definita seguendo la sintassi vista delle clausole definite proposizionale essa è pronta per essere fornita ad un agente intelligente affinché unendo le osservazioni sullo stato del mondo alla background knowledge possa rispondere a delle domande (o query) poste dall’utente per sapere il grado di verità di una parte del mondo.

Ricordiamo che tutto quello che nella base di conoscenza è rappresentato come un fatto e non come regola e quindi come conoscenza derivata a partire dai fatti che si sanno, prendono il nome di BACK GROOUND KNOWLEDGE ovvero la conoscenza che si ha a priori sul mondo che si rappresenta e che solitamente è quella rappresentata dall’esperto del dominio.

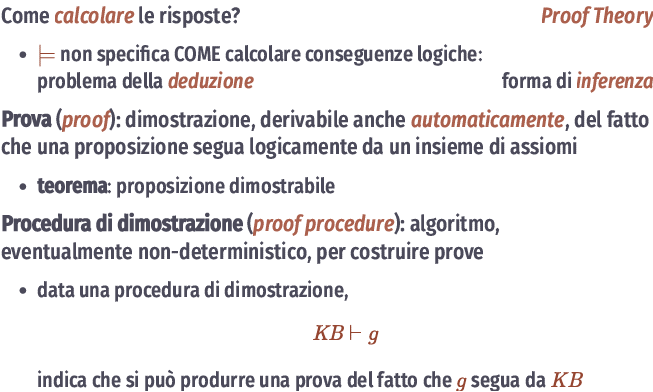
Una query è una domanda tesa a sapere se una data proposizione segua come conseguenza logica da una data base di conoscenza:



La query è : ASK seguito dal letterale atomo di cui vogliamo sapere se esso è conseguenza logica, oppure seguito da una congiunzione di atomi.

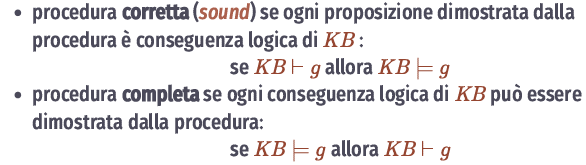
Fino ad ora abbiamo parlato di teoria dei modelli, cioè di come da una base di conoscenza supposta vera sia possibile dire che una proposizione ne sia conseguenza logica, ora invece ci occuperemo di come dire alle macchine di effettuare il processo di proof theory ovvero la teoria con cui la macchina calcola le risposte date in base alla conoscenza di background.

Nella teoria dei modelli la relazione |= ci dice che una proposizione è conseguenza logica di una data KB, ma non specifica il processo con il quale andare a calcolare tale consegue e quindi non spiega il processo inferenziale, per ottenere il processo di prova della conseguenza logica ci occuperemo della PROOF THEORY e della relazione di inferenza che l’agente farà con il simbolo |-- .



Sarà la macchina che andrà ad effettuare la procedura di dimostrazione per noi. Per fare questo c’è bisogno che la procedura di dimostrazione goda di 2 proprietà che permettono di mettere in relazione la teoria dei modelli con la teoria delle dimostrazioni, queste proprietà sono:

1. CORRETTEZZA (SOUND) 2. COMPLETEZZA



Le proprietà descritte sono una l’inverso dell’altra.

PROCEDURA BOTTOM-UP 🡪 BU

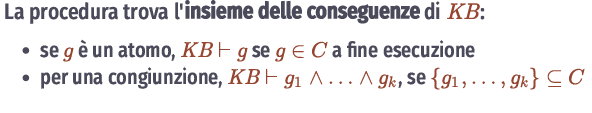
L’obiettivo di tale procedura algoritmica è quello di derivare tutte le conseguenze logiche che sono possibili da derivare da una KB e per farlo segue i seguenti passi:

Graphical user interface, text, application, email

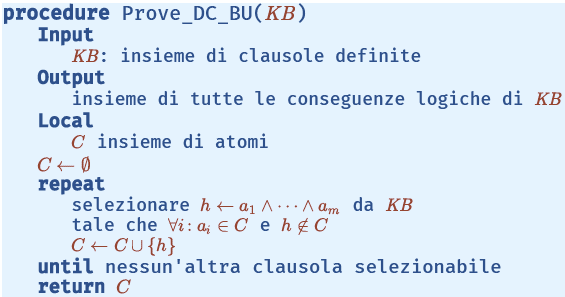
Description automatically generated

La regola di inferenza del modus pones ci dice che, se sappiamo che p 🡪 q è vera e sappiamo anche che p è vera allora possiamo dedurre (o inferire) che q è vera poiché il vero implica solamente il vero.

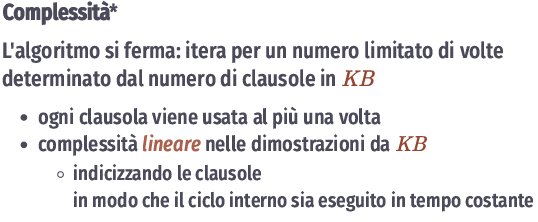
Presentiamo l’algoritmo di BU per la procedura di dimostrazione:



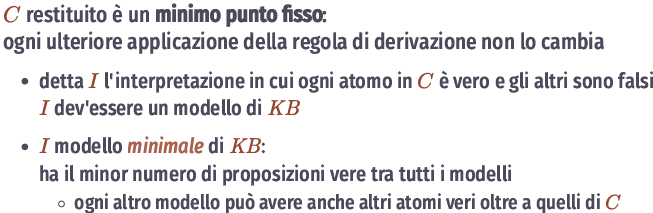
Il loop[repeat until] sarà ripetuto fino a quando si sarà trovato un punto fisso ovvero non è possibile più provare nessun’altra regola nuova, ogni qualvolta viene selezionata una clausola definita proposizionale ed il corpo è stato provato allora possiamo aggiungere la testa della clausola all’insieme C degli atomi provati e che quindi sono dimostrati come conseguenze logiche della KB.



Al termine dell’algoritmo per poter rispondere yes o no (ovvero per dire se è conseguenza logica o meno) allora basterà andare a controllare se l’atomo chiesto sia presente nell’insieme C ovvero degli atomi dimostrati.

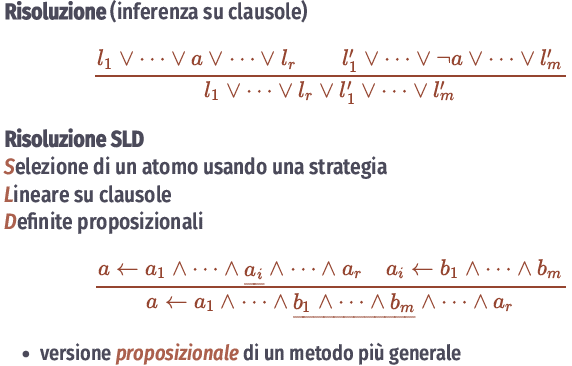


Per le proprietà dell’algoritmo e della sua complessità possiamo avere un punto fisso.



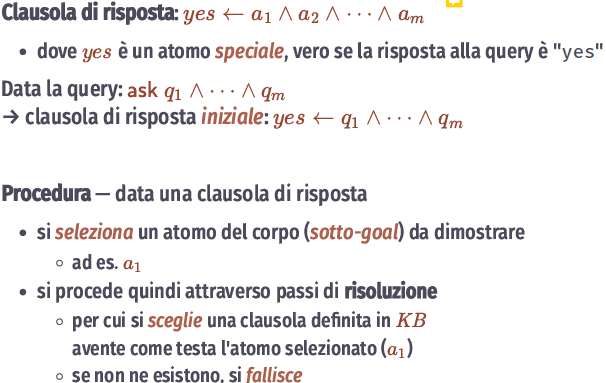
PROCEDURA TOP-DOWN 🡪 TD

In questo caso partiamo da quello che vogliamo dimostrare e procediamo a ritroso per vedere se quello che abbiamo nella KB ci porta a dimostrare il nostro goal. Questo algoritmo prende il nome di ricerca backward (o concatenazione all’indietro) e il processo di inferenza basa sul processo di RISOLUZIONE:



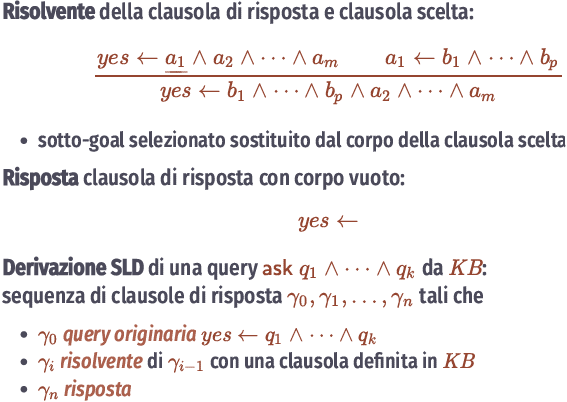
Il processo di RISOLUZIONE si adotta su qualsiasi tipo di clausola che risulti essere una disgiunzione di letterali. In particolare, si prendono in esame 2 disgiunzioni di letterali nelle quali in una vi è un letterale a in forma positiva e nell’altra un letterale not a ovvero in forma negativa, allora se entrambe le clausole risultano vere esse non dipendono dal valore di verità dell’atomo a in questione e quindi nel processo di risoluzione quell’atomo può essere non considerato.

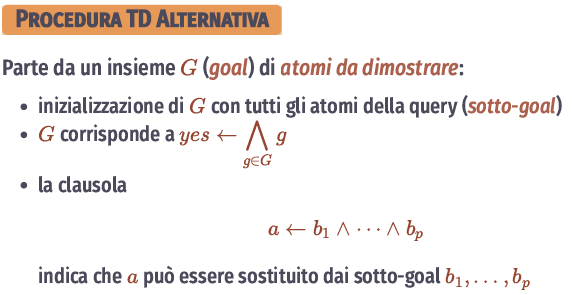
Con l’algoritmo di TD si parte dalla clausola speciale di risposta che è data dalla query come corpo e yes che risulta essere la testa che bisognerà dimostrare.

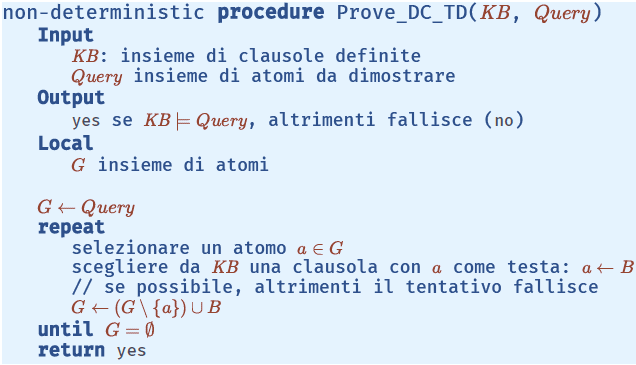


Un sotto goal è un pezzo della clausola disgiuntiva che vogliamo dimostrare, questo viene selezionato per poi essere dimostrato.

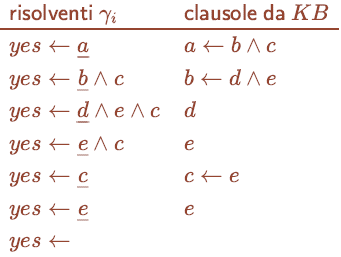
Nella procedura c’è del non determinismo, infatti per prima cosa dovrò selezionare un atomo (un sotto goal) e nessuno mi dice quale dovrò selezionare di volta in volta, dopo averlo selezionato ho una scelta da fare su quale delle clausole presenti nella KB userò per dimostrare il sotto goal anche qui ho del determinismo ma con una differenza sostanziale che va ad impattare enormemente sull’algoritmo, infatti mentre la prima scelta (la selezione del sotto goal) è un non determinismo DON’T CARE questo perché prima o poi li dovrò dimostrare tutti per poter rispondere alla query sottoposta, la seconda scelta influenza la riuscita o meno della dimostrazione e in tal caso servirebbe un euristica per selezionare dalla KB la clausola che mi farebbe fallire il processo di dimostrazione il più velocemente possibile. La fine del processo risolutivo è dato da una clausola yes in testa e un corpo vuoto.



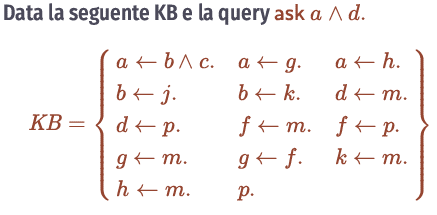


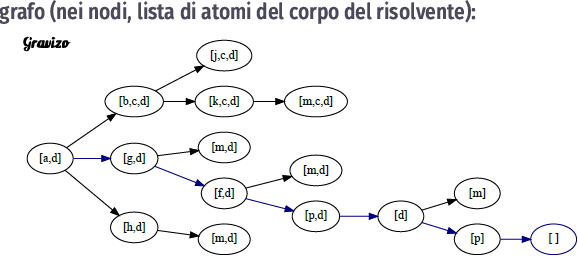


Esempio del processo di risoluzione:



Il non determinismo dell’algoritmo TD ha necessità di essere gestito affinché il fallimento sia naturale e non falsato dalla scelta di un sotto goal per il quale nella base KB non ci fosse una clausola che permettesse di dimostrarlo in quanto necessiterebbe di avere una clausola con il sotto goal selezionato come testa. L’algoritmo TD può essere visto come un grafo in particolare come un albero di ricerca come è possibile vedere nel seguente esempio:

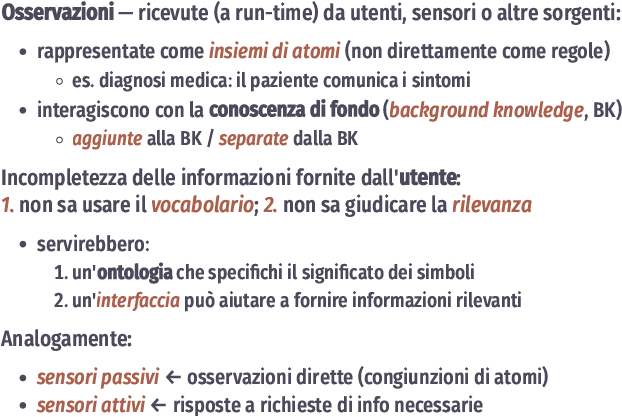




Il BU va avanti di forza bruta, andando a verificare e quindi dimostrare anche atomi che non sono inerenti alla query fatta, ovvero che non servirebbe dimostrare per riuscire a rispondere alla query, dall’altro canto è completo e permette di essere certi che in caso di fallimento il fallimento sia dovuto al fatto che la query non è conseguenza logica della KB altrimenti ci fornirà il punto fisso.

La TD(cerca di effettuare un DFS ovvero una ricerca in profondità) invece si concentra sui letterali che si necessitano per provare la query, quindi, cerca di lavorare con solo quelli strettamente necessari al processo, tuttavia in questo processo potrebbe provare lo stesso atomo più volte, infine il reale problema da affrontare è quello dei cicli infiniti. La strategia di selezione risulta quella che porta per prima al fallimento per risparmiare tempo computazionale.

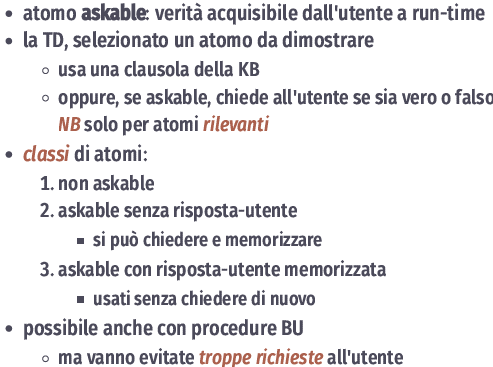
QUESTIONI DI RAPPRESENTAZIONE DELLA CONOSCENZA



Le osservazioni saranno gestite dal sistema come atomi da aggiungere alla conoscenza della KB.

L’ acquisizione di ulteriore informazione sul mondo esterno può avvenire tramite un interfaccia che permetta un interazione tra il sistema agente intelligente e un utente che fornisca queste ulteriori informazioni sullo stato del mondo.

L’acquisizione è un meccanismo ask-the-user nella procedura di ragionamento ove:



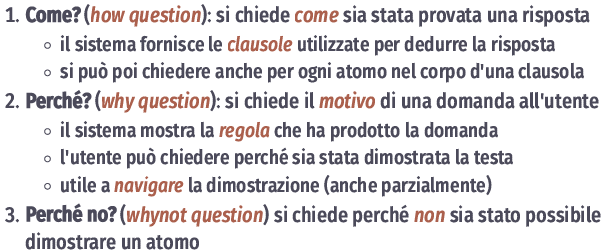
Un atomo ASKABLE è un atomo il cui grado di verità può essere chiesto all’utente, questo serve perché nel momento in cui l’agente effettua un procedura di prova di una query sottoposta potrebbe necessitare di un tale atomo del quale non conoscendone il valore di verità non può usare nel processo dimostrativo quindi per poter proseguire chiede all’utente di dirgli il grado di verità affinché continui la dimostrazione.

Lo scambio di informazione visto fino ad ora vede l’utente in una posizione non del tutto attiva nel senso che aspetta che sia il sistema a chiedergli di fornire informazione aggiuntiva, in una visione più complessa questo scambio di informazione può avere una SIMMETRIA di ruoli tra utente e sistema nel senso che entrambi possono porre domande e fornire informazioni utili all’altro per soddisfare dei processi logico deduttivi. Ad esempio, l’utente può chiedere al sistema come abbia fatto a dimostrare un determinato atomo[questo permette un processo di spiegazione e debugging della KB]. Questo permette di instaurare un protocollo del tipo domanda-risposta:

Text

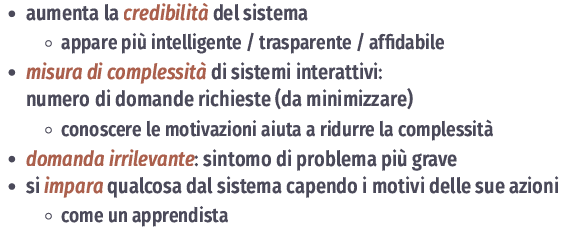
Description automatically generated

Le domande che l’utente può fornire al sistema agente possono essere di vario tipo e sono finalizzate a capire più a fondo il processo deduttivo usato dalla macchina al fine di avere chiara la dimostrazione che la macchina ha utilizzato per rispondere ad una query. I tipi di domanda che è possibile porre sono 3:



Con una domanda di tipo HOW il sistema andrà a mostrare quale clausola presente nella KB ha usato per dimostrare che una precedente query sottoposta fosse conseguenza logica della KB. Una singola domanda HOW permette di vedere la più recente clausola utilizzata nella dimostrazione, questa significa che continuando a chiedere HOW di volta in volta al sistema riuscirò a visualizzare a ritroso il processo di inferenza logica usato dal sistema per la dimostrazione.

Con una domanda WHY (perché) è possibile lavorare sui seguenti aspetti:



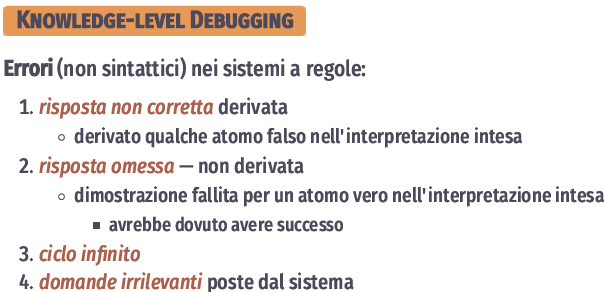
Questo tipo di domanda è usata dall’utente in risposta ad una domanda posta dal sistema per sapere il grado di verità di un atomo askable, quindi una volta che il sistema richiede tale informazione e noi gliela forniamo possiamo chiedere al sistema WHY ovvero perché ha posto tale domanda (a cosa gli serve) e il sistema risponderà fornendo la regola per la quale serve sapere il grado di verità del atomo askable appena chiesto. Nel caso in cui l’utente continuasse a porre domande WHY il sistema fornisce un percorso dei sotto goal dimostrati (o traccia) a ritroso fino alla query originaria che gli è stata posta.

HOW e WHY consentono di navigare l’albero di dimostrazione del sistema il PROOF TREE dove abbiamo:

Text

Description automatically generated with medium confidence

Queste domande possono essere poste al sistema in particolare per effettuare il debugging della KB.



Una risposta errata è considerata una risposta data dalla dimostrazione che risulta falsa nell’interpretazione intesa. In tal caso c’è un bug nella KB che necessita di un fix. Difronte ad un falso positivo si possono avere 2 casi possibili:

* 1 o + atomi presenti nel corpo della clausola utilizzata per dimostrare il falso positivo sono falsi nell’interpretazione intesa e questo porta ad un problema da risolvere in modo ricorsivo.
* Tutti gli atomi presenti nel corpo della clausola utilizzata sono veri nell’interpretazione intesa e allora si conclude che è la clausola ad essere errata ovvero non dovrebbe essere presente nella KB.

Un altro problema con relazione di bug presente nella KB può essere dovuta ad una risposta mancante ovvero una risposta che dovrebbe essere derivata correttamente come conseguenza della KB non viene derivata come tale.

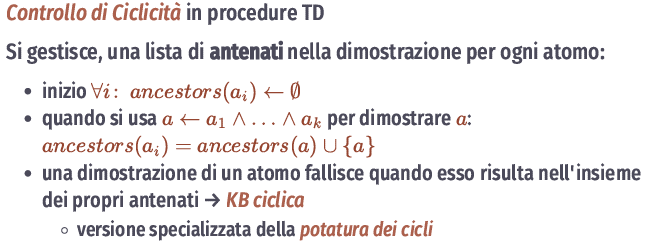


Il fallimento della dimostrazione nella proof theory è indicato tramite il simbolo  per cui  sta a significare che il sistema non è riuscito a dimostrare g dalla KB.

In questo caso significa che nella KB manca qualche clausola che serve al sistema per dimostrare l’atomo sul quale ha fallito.

Nel caso di fallimento è possibile porre al sistema il 3° tipo di domanda ovvero la WHY NOT domanda che serve a chiedere spiegazione da parte del sistema del perché non sia riuscito a dimostrare un atomo.

Un altro indicatore di un problema nella KB sono i cicli infiniti questi sono dovuti a KB cicliche, una KB è definita ciclica quando le clausole in essa contenute sono tali per cui si instaura un ciclo di inferenza infinito nel processo di risoluzione. [i cicli danno problemi al TD, infatti il BU non è affetto da tale problema, è possibile tuttavia utilizzare alcuni accorgimenti a livello del algoritmo TD per evitare i cicli, con adeguate procedure di controllo dei cicli].

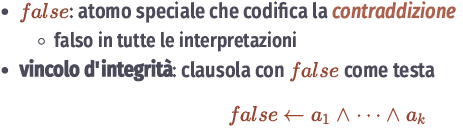


Il controllo dei cicli sta nel mettere in un insieme in memoria tutti gli atomi che vengono utilizzati per la dimostrazione di una testa di modo tale che quando nel proseguo della dimostrazione trovo nuovamente un atomo già utilizzato allora avrò un ciclo e questa strada non dovrà essere considerata nella dimostrazione ovvero l’algoritmo torna indietro per tentare una strada differente per la dimostrazione, questa tecnica risulta una specializzazione della potatura dei cicli vista con gli algoritmi di ricerca.

DIMOSTRAZIONE PER CONTRADDIZIONE

Il punto debole delle KB viste fino ad ora con i processi di risoluzione BU e TD è che riusciamo a dimostrare soltanto alcuni tipi di proposizioni ovvero o le proposizioni atomiche quindi gli atomi oppure le congiunzioni di atomi. Quindi abbiamo la necessità di aumentare le tipologie di proposizioni che possono essere chieste e risolte dal sistema e per farlo abbiamo la necessità di aumentare la complessità del linguaggio aggiungendo un tipo particolare di clausole che ci permetteranno di esplicitare quelle che sono delle contraddizioni presenti nella KB, ovvero ci permette di specificare dei casi impossibili e questo si traduce nella possibilità di implementare un ragionamento per contraddizione.

Il linguaggio delle clausole definite può essere facilmente esteso considerando un unico atomo aggiuntivo alla sintassi ovvero l’atomo FALSE:



False è sempre falso anche nell’interpretazione intesa. Oltre ad aggiungere tale atomo alla sintassi aggiungiamo delle nuove clausole dette CLAUSOLE DI HORNE oltre ad esprimere una proposizione definita con la testa positiva potrà anche avere la testa negativa questa sintassi permette di esprimere i VINCOLI DI INTEGRITA’.

Con i vincoli di integrità che risultano essere clausole con testa l’atomo false quello che facciamo è andare a codificare il logica la contraddizione del ragionamento.

In questo caso abbiamo un linguaggio più espressivo e potente di quelle delle clausole definite e prende il nome di linguaggio di HORNE o linguaggio delle clausole di horne.

Grazie a queste nuove clausole posso sfruttare un nuovo meccanismo inferenziale per la dimostrazione, che permette di derivare negazioni di atomi tramite il processo del MODUS TOLLENS:

Diagram

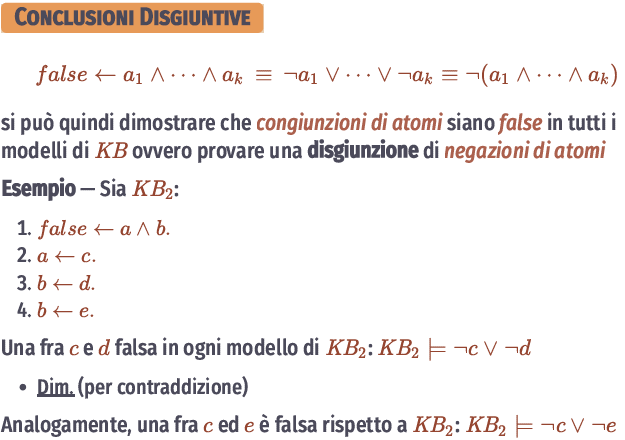
Description automatically generated with medium confidence

Quindi se so che p 🡪 q è vero e ho dimostrato precedentemente che not q è vero allora posso dedurre da ciò che not p è vero sfruttando l’implicazione di p verso q. Questo processo serve per poter dimostrare un atomo negativo, o meglio per stabilire che un letterale negativo sia vero nell’interpretazione intesa e che quindi sia una conseguenza logica della KB.

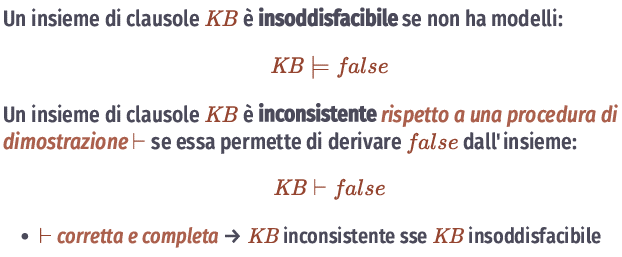
Text

Description automatically generated

Lavorando con la negazione e con le leggi di De Morgan possiamo ottenere delle conclusioni disgiuntive:

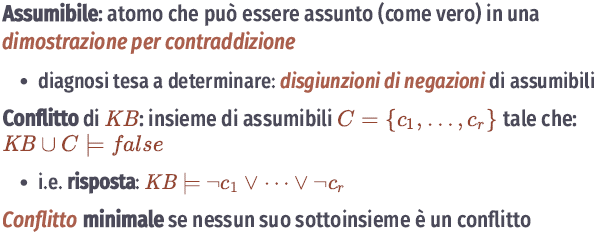


Quindi adesso è possibile dimostrare la negazione di una congiunzione di atomi che si traduce nel provare una disgiunzione di atomi negati.

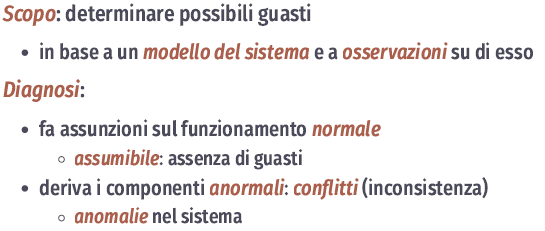


Una differenza notevole con le clausole definite e che queste sono sempre soddisfacibili, per cui con le clausole di Horne guadagniamo una maggiore capacità espressiva e risolutiva pagandola con la possibilità di avere una KB insoddisfacibile. Per capire se una KB è insoddisfacibile potrei chiedere al sistema se riesce a derivare false come conseguenza logica [tramite ask false].

Nel momento in cui mi trovo in situazioni di contraddizione per debuggare la KB farò alcune assunzioni e in particolare potrò assumere un determinato atomo como vero(indipendentemente dal fatto che lo sia o meno) e vedrò quali conseguenze ci saranno:



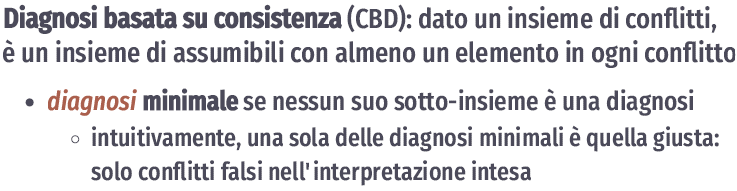
DIAGNOSI BASATA SULLA CONSISTENZA



Text

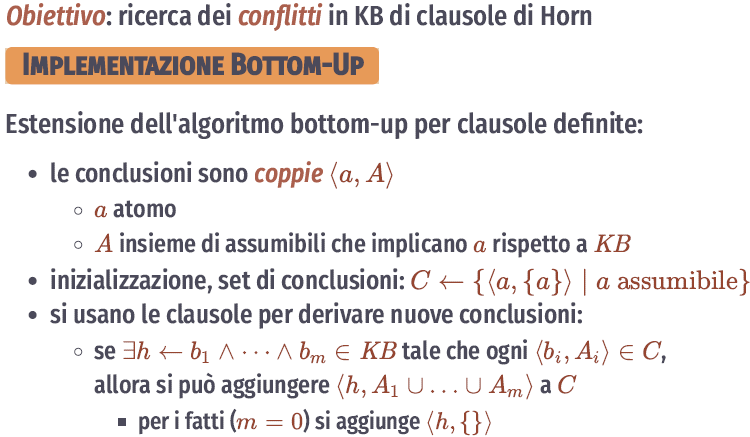
Description automatically generated

CBD = Consistency Based Diagnosys



Per effettuare il processo di prova o dimostrazione delle query con la nuova sintassi data dalle clausole di Horne e dal atomo “false” usiamo gli algoritmi BU e TD con alcune modifiche mirate a lavorare efficientemente con la nuova sintassi.

Si porta avanti un insieme di assumibili utilizzati nella dimostrazione, questi vengono raccolti perché nel caso di una dimostrazione di un atomo falso saranno questi atomi ad essere rivisti per capire se vi sono problemi nella KB.

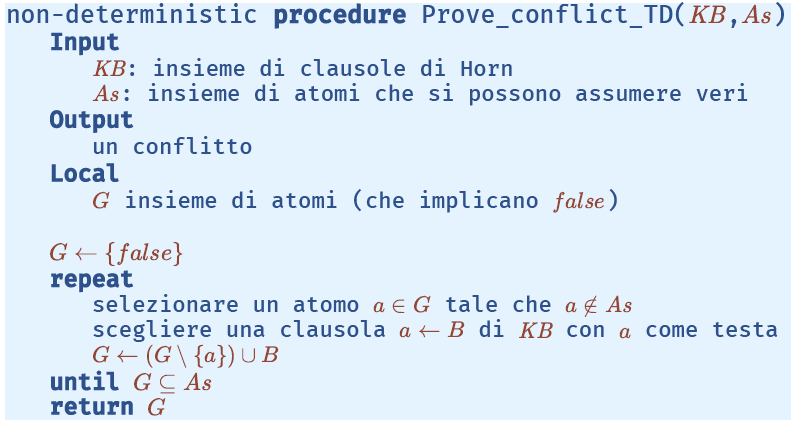


Come è possibile notare dall’algoritmo nel BU modificato per lavorare con le clausole di Horne porto avanti delle coppie di conflitti queste coppie di conflitti stanno a dimostrare che riesco a provare la verità di un atomo ovvero dimostrarla a patto che ci siamo degli assumibili che appunto sono stati assunti per veri e di cui tengo traccia in caso di fallimenti nella dimostrazione.

Text, timeline

Description automatically generated

Nella procedura TD il ragionamento è simile al algoritmo che lavorava su clausole definite proposizionali, ma qui ci sono 2 differenze ovvero la query di partenza da provare anziché avere una testa posta a YES avrà una testa posta a FALSE ed infine gli atomi che verranno usati come assumibili non devono essere dimostrati nella procedura, ma semplicemente raccolti in modo da tenerli sotto controllo in caso di problemi.



ASSUNZIONE DI CONOSCENZA COMPLETA

La prima utilità vista grazie alla modifica del linguaggio e all’introduzione delle clausole di Horne è quella di avere una maggiore possibilità sulle conclusioni che posso dimostrare in particolare la possibilità di provare la negazione di congiunti di atomi( ovvero disgiunti di atomi negati) e di sfruttare la dimostrazione per contraddizione.

Adesso studiamo ulteriori forme di inferenza che possono modificare il comportamento delle procedure di dimostrazione e renderle simili al processo usato dai database relazionali per rispondere alle query. Per fare questo si necessita un assunzione forte che è quella di CONOSCENZA COMPLETA ovvero si conosce tutto riguardo al mondo. Questo significa che nel caso qualcosa non si sappia o, meglio, non si riesca a inferire dalla KB allora si dà per falsa.

In logica normalmente non si assume che una KB definisca tutta la conoscenza possibile su un dato dominio ed è per questo che si dice che il sistema lavora in OWA = OPEN-WORLD ASSUMPTION, diversamente nei data base si usa l’assunzione di una conoscenza che sia esaustiva per poter inferire qualsiasi informazione chiesta sul mondo da parte del sistema e in questo caso si dice che ragioniamo in CWA = CLOSED-WOLRD ASSUMPTION :

Text

Description automatically generated

Per ottenere una base di conoscenza chiusa e necessario accorpare le clausole che provano tutte lo stesso atomo di testa e creare una nuova proposizione che avrà l’antecedente(corpo) composto da un disgiunto di corpi ove questi corpi messi in disgiunzione sono i corpi delle clausole del atomo di testa comune che verificano. Questa nuova preposizione che si forma del tipo:

a 🡨 corpo1 OR corpo2 OR corpo3 ……OR corpon

non può essere considerata come clausola poiché la clausola ha il corpo formato da un insieme di disgiunti che sono atomi ovvero letterali e non formule composte da atomi ed è per questo che si identifica come una proposizione. Questa nuova proposizione aggiunta alla KB ci permette di dire che, se l’atomo di testa a risulta dimostrato e quindi vero lo sarà perché uno degli n corpi e per tale ragione possiamo dire che sarà necessariamente vera anche l’implicazione in senso inverso ovvero quella che vede a come antecedente e il disgiunto di corpi come conseguente:

a 🡪 corpo1 OR corpo2 OR corpo3 …… OR corpon

Text

Description automatically generated with medium confidence

E questa implicazione inversa è vera se e solo se le n clausole che dimostrano a sono le uniche e sole capaci di farlo e questo equivale a fare un assunzione completa ovvero di mondo chiuso.

Questo ragionamento porta a modificare le clausole che avevamo della KB in forma di Horne e metterle in forma proposizionale con il COMPLEMENTO DI CLARK ovvero tutte le precedenti clausole viste per poter asserire l’implicazione inversa vengono riassunte in una doppia implicazione tra la testa e il disgiunto degli n corpi.

Text

Description automatically generated

Grazie a queste modifiche e alla sintassi data dal completamento di Clark ottengo un processo di dimostrazione per fallimento indicato con la tilde ~ che permette di dimostrare la falsità di un letterale ovvero:

A picture containing text

Description automatically generated

Con il complemento di Clark è possibile dimostrare le negazioni ovvero gli atomi negati.

Adesso quindi in una KB potremmo trovare delle regole di questo tipo:

A picture containing text

Description automatically generated

Anche in questo caso potremmo aver dei problemi di cicli infiniti, quindi, bisogna controllare che KB non sia ciclica.

Normalmente in logica con l’aumentare della conoscenza che è presente nella KB aumenta anche la capacità di quello che è possibile dimostrare tramite processi dimostrativi, mentre in alcuni casi come quello dell’uso delle preposizioni con assunzione di conoscenza completa e negazione per fallimento porta invece che aumentare la conoscenza a restringerla poiché ci sono alcune negazioni che portano ad invalidare alcune regole e sfruttando questo meccanismo si ha che sfruttando più conoscenza su più cose queste possono far scattare meno conseguenze poiché vi è una restrizione nelle regole, per tali ragioni si dice che il ragionamento in questo caso non è monotono poiché al crescere della conoscenza non aumentano le conseguenze che posso derivare.

Graphical user interface, text, application

Description automatically generated

Questo tipo di ragionamento può essere visto come è vero un fatto se è vero una premessa e non si sono verificate eccezioni ad esempio:

Il ragionamento si dice per default ovvero posso dire la mia macchina è qui fuori, ma io non so se c’è qualcuno che l’abbia rubata, oppure un altro evento che abbia cambiato quello che do per vero come un alluvione che si sia portata la macchina. Quindi il ragionamento che codifico è quello che normalmente la macchina è qui fuori a meno ché non sia successo qualcosa di anomalo che cambi questo fatto.

Da questo esempio capiamo il perché del nome “ragionamento per default” poiché quello che rappresentiamo è la normalità nel momento in cui non si presentano casi eccezionali o eventi che vadano a cambiare la normalità assunta come standard o default

Text

Description automatically generated with low confidence

La procedura di dimostrazione considerando la NAF fa cambiare leggermente il funzionamento del BU il quale adesso avrà un insieme C delle conseguenze logiche che è possibile derivare dalla KB, nel quale andrà ad aggiungere ogni qualvolta un letterale fallisce la sua negazione ~letterale che sta a significare che è possibile dimostrare ¬letterale (fallito):

Text

Description automatically generated

Con questo algoritmo trovo sia le conseguenze positive che negative.

Text

Description automatically generated

Con la procedura TD e la NAF abbiamo sempre a che fare con il non determinismo che accompagna l’algoritmo TD, in questo caso il funzionamento si basa sulla selezione di un atomo  dal quale parte la dimostrazione per il sotto goal a ove se questa fallisce allora  ha successo e quindi è dimostrabile ¬a altrimenti l’algoritmo fallisce e deve tentare altre strade, in tal caso il fallimento fallisce per divergenza e non è possibile concludere nulla

Text

Description automatically generated

Timeline

Description automatically generated

ABDUZIONE

Ultima forma di inferenza che vedremo nel corso, utilizzata essenzialmente per fare diagnosi di consistenza della KB. La forma di ragionamento di ABDUZIONE è stato inventata dal filosofo Pierce e prevede la formulazione di ipotesi/assunzioni che servono a poter spiegare delle osservazioni fatte. Queste assunzioni che vengono fatte tuttavia non sono aggiunte alla base di conoscenza poiché esse servono a mettere in evidenza delle falle nel ragionamento del sistema.

L’abduzione si differenzia dai meccanismi già visti di ragionamento logico, infatti la deduzione usa la conoscenza presente nella KB e serve a derivare tutte le conseguenze logiche possibili, mentre l’abduzione parte dalle conseguenze logiche e cerca di capire se è possibile ritrovare delle spiegazione che assieme alla conoscenza presente nella KB porti a verificare o meno le conseguenze logiche che abbiamo e da cui partiamo.

È diversa anche dall’induzione ove comporta l’inferenza di relazione generali da casi particolari.

Graphical user interface, text, application, email

Description automatically generated

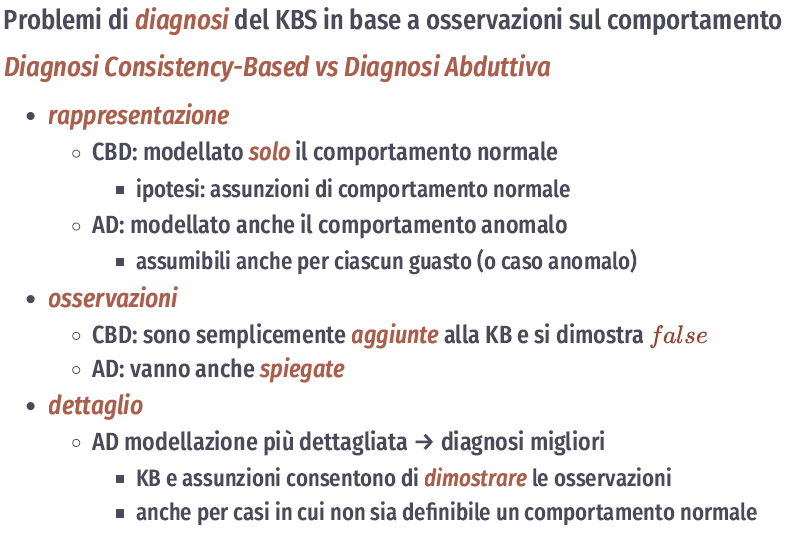
Nelle spiegazioni si parla di scenari costituiti da coppie formate dalla KB e dall’insieme A degli assumibili di modo tale che l’ipotesi fatta con l’abduzione risulti essere un sotto insieme di A tale che unendo l’ipotesi alla KB questo insieme risulti essere soddisfacibile.

Una spiegazione quindi altro non è che uno scenario che spiega un certo goal ovvero una determinata proposizione.

Text

Description automatically generated

Con l’abduzione è possibile andare a modificare il processo di debug della KB visto con la CBD ovvero la consistency based diagnosys, ora usiamo la cosiddetta diagnosi abduttiva AD:



MODELLI CAUSALI [SOLO LETTURA]