Metodi Matematici per l'Informatica - Dispensa 18
(a.a. 23/24, I canale)

Docente: Lorenzo Carlucci (lorenzo.carlucci@uniroma1.it)

1 Equivalenza logica, verità notevoli

Definizione 1 (Equivalenza Logica)**.** *Due formule A, B si dicono logicamente equivalenti se per ogni assegnamento α, $\alpha(A) = \alpha(B)$. In questo caso scriviamo $A \equiv B$.*

Osservazione 1. Si osserva facilmente che $A \equiv B$ se e solo se $\models (A \leftrightarrow B)$

Osservazione 2. La relazione di equivalenza logica è invece una relazione di equivalenza sull'insieme delle proposizioni.

1. $A \equiv A$
2. Se $A \equiv B$ e $B \equiv C$ allora $A \equiv C$
3. Se $A \equiv B$ allora $B \equiv A$.

Possiamo isolare le seguenti **verità notevoli**. Si verificano tutte facilmente usando le tavole di verità o la definizione della semantica per i connettivi booleani.

1. Associatività
   (a) $(A \vee (B \vee C)) \equiv A \vee (B \vee C)$
   (b) $(A \wedge (B \wedge C)) \equiv A \wedge (B \wedge C)$
2. Commutatività
   (a) $(A \vee B) \equiv (B \vee A)$
   (b) $(A \wedge B) \equiv (B \wedge A)$
3. Distributività
   (a) $(A \vee (B \wedge C)) \equiv (A \vee B) \wedge (A \vee C)$
   (b) $(A \wedge (B \vee C)) \equiv (A \wedge B) \vee (A \wedge C)$
4. Leggi di De Morgan
   (a) $\neg(A \vee B) \equiv (\neg A \wedge \neg B)$
   (b) $\neg(A \wedge B) \equiv (\neg A \vee \neg B)$
5. Doppia Negazione $\neg\neg A \equiv A$
6. Idempotenza
   (a) $(A \vee A) \equiv A$

(b) $(A \wedge A) \equiv A$

Il seguente gruppo di leggi logiche notevoli illustra la possibilità di definire alcuni connettivi in funzione di altri.

1. $(A \leftrightarrow B) \equiv ((A \rightarrow B) \wedge (B \rightarrow A))$
2. $(A \rightarrow B) \equiv (\neg A \vee B)$
3. $(A \vee B) \equiv (\neg A \rightarrow B)$
4. $(A \vee B) \equiv \neg(\neg A \wedge \neg B)$
5. $(A \wedge B) \equiv \neg(\neg A \vee \neg B)$

Se conveniamo di usare — all'interno di proposizioni — la costante 1 al posto di una qualunque tautologia e la costante 0 al posto di una qualunque formula insoddisfacibile, allora possiamo formulare le seguenti leggi algebriche aggiuntive.

1. Assorbimento
   (a) $(A \vee 0) \equiv A$
   (b) $(A \wedge 1) \equiv A$
2. Contraddizione, Terzo Escluso
   (a) $(A \vee \neg A) \equiv 1$
   (b) $(A \wedge \neg A) \equiv 0$

Possiamo dimostrare l'equivalenza logica di due formule usando le verità notevoli qui sopra sostituendo, all'interno di una formula, una sua sottoformula con una formula logicamente equivalente.

Esempio Dimostriamo che $\models (A \rightarrow (B \rightarrow C) \leftrightarrow ((A \wedge B) \rightarrow C)$.

$$\begin{aligned} A \rightarrow (B \rightarrow C) &\equiv \neg A \vee (B \rightarrow C) \\ &\equiv \neg A \vee (\neg B \vee C) \\ &\equiv (\neg A \vee \neg B) \vee C \\ &\equiv \neg(A \wedge B) \vee C \\ &\equiv (A \wedge B) \rightarrow C \end{aligned}$$

Esempio Dimostriamo che $\models (A \rightarrow B) \leftrightarrow (\neg B \rightarrow \neg A)$.

$$\begin{aligned} \neg B \rightarrow \neg A &\equiv \neg\neg B \vee \neg A \\ &\equiv B \vee \neg A \\ &\equiv \neg A \vee B \\ &\equiv A \rightarrow B \end{aligned}$$

2 Forma Normale Congiuntiva

Chiamiamo "letterale" una variabile proposizionale o una negazione di una variabile proposizionale.

Diciamo che A è in **Forma Normale Congiuntiva** (CNF) se A è una congiunzione di disgiunzioni di letterali, ossia è della forma seguente, dove gli $L_{i,j}$ sono letterali.

$$(L_{1,1} \vee L_{1,2} \vee \cdots \vee L_{1,m_1}) \wedge (L_{2,1} \vee L_{2,2} \vee \cdots \vee L_{2,m_2}) \wedge \cdots \wedge (L_{n,1} \vee L_{n,2} \vee \cdots \vee L_{n,m_n})$$

Usiamo $\bigwedge_{i \leq n} A_i$, dove le A_i sono formule, come abbreviazione di

$$A_1 \wedge A_2 \wedge \cdots \wedge A_n.$$

e analogamente $\bigvee_{i \leq n} A_i$, dove le A_i sono formule, come abbreviazione di

$$A_1 \vee A_2 \vee \cdots \vee A_n.$$

Con questa notazione, A è una CNF se è della forma

$$\bigwedge_{i \leq n} \bigvee_{j \leq m_i} L_{i,j},$$

dove gli $L_{i,j}$ sono letterali.

Vale il seguente Teorema di Forma Normale Congiuntiva.

Teorema 1 (Forma Normale Congiuntiva)**.** *Per ogni A esiste A^{CNF} tale che A^{CNF} è una CNF e*

$$A \equiv A^{\mathsf{CNF}}.$$

Usando le leggi logiche e la sostituzione di formule equivalenti per formule equivalenti all'interno di una data formula è possibile trasformare una formula A arbitraria in una formula CNF:

1. Sostituisco $B \to C$ con $\neg B \vee C$
2. Sostituisco $B \leftrightarrow C$ con $(\neg B \vee C) \wedge (\neg C \vee B)$
3. Sostituisco $\neg(B \wedge C)$ con $\neg B \vee \neg C$
4. Sostituisco $\neg(B \vee C)$ con $\neg B \wedge \neg C$
5. Sostituisco $\neg\neg B$ con B
6. Applico la legge di distributività dell'OR sull'AND, ossia sostituisco $B \vee (C \wedge D)$ con $(B \vee C) \wedge (B \vee D)$.

Esempio 1.

$$(A \to B) \to (\neg B \to \neg A)$$
$$\neg(A \to B) \vee (\neg B \to \neg A)$$
$$\neg(\neg A \vee B) \vee (\neg\neg B \vee \neg A)$$
$$(A \wedge \neg B) \vee (B \vee \neg A)$$
$$(A \vee B \vee \neg A) \wedge (\neg B \vee B \vee \neg A)$$

In questo modo abbiamo ottenuto una CNF equivalente alla proposizione iniziale. Si verifica facilmente che la CNF ottenuta è una tautologia, perché ciascuno dei suoi congiunti è una tautologia. Possiamo quindi concludere che la formula iniziale è una tautologia.

Alternativamente, possiamo continuare la valutazione dalla penultima riga come segue.

$$(A \vee B \vee \neg A) \wedge (\neg B \vee B \vee \neg A)$$
$$(B \vee 1) \wedge (\neg A \vee 1)$$

così abbiamo dimostrato che la proposizione iniziale è una tautologia (i.e. una verità logica).

Un altro modo per ottenere una CNF equivalente a una data proposizione è il seguente. Sia A una proposizione nelle variabili $p_1, \ldots, p_n$. Consideriamo la sua tavola di verità. Consideriamo le righe in cui A prende valore 0. A ogni riga di questo tipo associamo la clausola ottenuta prendendo la disgiunzione dei letterali ℓ_i per $1 \leq i \leq n$ dove ℓ_i è p_i se p_i ha valore 0 in quella riga ed è $\neg p_i$ se p_i ha valore 1 in quella riga. La congiunzione di tutte queste clausole è equivalente alla formula A di partenza.

Osservazione 3. Una CNF è una tautologia se e soltanto se tutti i suoi congiunti sono tautologie.

Potremmo allora pensare di affrontare il problema di decidere se $A \in \mathsf{TAUT}$ (o equivalentemente $\neg A \in \mathsf{UNSAT}$) scrivendola in CNF e decidendo se i congiunti sono in TAUT. Purtroppo questo metodo non dà luogo a un algoritmo efficiente: la trasformazione di una formula arbitraria in una CNF equivalente può dar luogo a una esplosione esponenziale della dimensione della formula.

Rappresentazione insiemistica di CNF

Fissiamo una rappresentazione agile e compatta per formule in CNF. Una formula F in CNF ha questa forma:

$$C_1 \wedge C_2 \wedge \cdots \wedge C_n$$

dove ogni C_i (detta **clausola**) è una disgiunzione di letterali, ossia C_i è di forma

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

per qualche k, per qualche letterale $\ell_1, \ldots, \ell_k$.

Rappresentiamo una clausola $C = \ell_1 \vee \cdots \vee \ell_k$ come l'insieme dei suoi letterali

$$\{\ell_1, \ldots, \ell_k\}$$

e rappresentiamo una formula $F = C_1 \wedge \cdots \wedge C_n$ come l'insieme delle sue clausole

$$\{C_1, \ldots, C_n\}$$

.

Si noti che se C è una clausola disgiuntiva un assegnamento soddisfa C se e solo se esiste un letterale in C soddisfatto dall'assegnamento.

Usiamo il simbolo $\square$ per indicare la clausola vuota, ossia senza letterali.

Usiamo il simbolo $\emptyset$ per indicare la formula vuota, ossia senza clausole.

Osserviamo che, secondo le definizioni:

1. $\square$ è in UNSAT: infatti non è vero che esiste un assegnamento v tale che esiste un letterale $\ell \in \square$ tale che $v(\ell) = 1$.
2. $\emptyset$ è in SAT: Infatti esiste un assegnamento v tale che per ogni clausola C, se $C \in \emptyset$ allora $v(C) = 1$ (l'implicazione è vera a vuoto).

Si osserva facilmente che se F è una CNF e $\square \in F$ allora anche $F \in \mathsf{UNSAT}$, perché un assegnamento che soddisfa F dovrebbe soddisfare tutte le clausole in F ma $\square$ è insoddisfacibile.

Dato che una clausola è in TAUT se e solo contiene un letterale e la sua negazione, si osserva facilmente che se $C \in F$ e $C \in \mathsf{TAUT}$ allora: $F \in \mathsf{SAT}$ sse $F - \{C\} \in \mathsf{SAT}$. Per questo possiamo assumere da ora in poi, senza perdita di generalità, che le nostre formule non contengano clausole tautologiche. Alternativamente possiamo immaginare di partire da una formula contenente anche clausole tautologiche e di cancellare tutte queste clausole. La formula ottenuta è SAT se e solo se lo era la formula di partenza.

3 Risoluzione

La Risoluzione è un metodo alternativo per risolvere problemi di soddisfacibilità/insoddisfacibilità senza usare le tavole di verità. In moltissimi casi questo metodo risulta molto più efficiente delle tavole di verità.

3.1 Regola di Risoluzione

Se ℓ è un letterale definiamo $\bar{\ell}$ come segue: $\bar{\ell} = \neg p$ se $\ell = p$ (è una variabile proposizonale); $\bar{\ell} = p$ se $\ell = \neg p$ (negazione di variabile proposizionale).

Siano C_1 e C_2 due clausole, ℓ un letterale, tali che $\ell \in C_1$ e $\bar{\ell} \in C_2$. Definiamo

$$\mathsf{Res}_\ell(C_1, C_2) = (C_1 - \{\ell\}) \cup (C_2 - \{\bar{\ell}\}).$$

$\mathsf{Res}_\ell(C_1, C_2)$ è una clausola e viene detto il *risolvente* (o *resolvent*) di C_1 e C_2 rispetto a ℓ. La regola di Risoluzione è la regola che ci permette di passare dalle clausole C_1 e C_2 al loro risolvente $(C_1 - \{\ell\}) \cup (C_2 - \{\bar{\ell}\})$.

L'interesse della nozione è che la formula a destra segue logicamente dalle due clausole a sinistra, ossia vale il seguente Teorema.

Teorema 1 (Lemma di Risoluzione)**.** *Sia ℓ un letterale tale che $\ell \in C_1$ e $\bar{\ell} \in C_2$. Allora*

$$C_1, C_2 \models \mathsf{Res}_\ell(C_1, C_2).$$

Dimostrazione. Supponiamo che esista un assegnamento α che non soddisfa il risolvente ma soddisfa C_1 e C_2. Se $\alpha(C_1) = 1$, dato che C_1 è una disgiunzione di letterali, deve esistere un letterale Q in C_1 tale che $\alpha(Q) = 1$. Ragioniamo per casi.

Caso 1: Q è diverso da ℓ. Allora Q compare anche in $C_1 \setminus \{\ell\}$ e dunque anche nel risolvente $(C_1 \setminus \{\ell\}) \cup (C_2 \setminus \{\bar{\ell}\})$. Dunque α soddisfa il risolvente, contro l'ipotesi.

Caso 2: Q è proprio ℓ. Dato che $\alpha(Q) = 1$ abbiamo $\alpha(\bar{\ell}) = 0$. Ma $\alpha(C_2) = 1$. Dunque deve esistere un altro letterale, diciamo P, in C_2 tale che $\alpha(P) = 1$. Questo letterale è anche in $C_2 \setminus \{\bar{\ell}\}$ e dunque anche nel risolvente. Dunque α soddisfa anche il risolvente, contro l'ipotesi.

Concludiamo che un tale α non può esistere. Dunque abbiamo dimostrato che se un assegnamento α non soddisfa il risolvente, allora α non soddisfa $C_1 \wedge C_2$. In altre parole (per contrapposta): ogni assegnamento α che soddisfa sia C_1 che C_2 soddisfa anche il risolvente.

□

3.2 Correttezza della Risoluzione

Dal Lemma precedente segue che, se partendo da una formula $F = \{C_1, \ldots, C_n\}$ in CNF, siamo in grado di ottenere la clausola vuota $\square$ applicando la regola di Risoluzione, allora la formula di partenza è insoddisfacibile!

Teorema 2 (Correttezza)**.** *Sia $F = \{C_1, \ldots, C_n\}$ una formula in CNF. Se applicando la regola di Risoluzione un numero finito di volte alle clausole in F e ai risultanti risolventi ottengo la clausola vuota $\square$, allora F è insoddisfacibile.*

Dimostrazione. Ogni applicazione della regola di Risoluzione preserva la conseguenza logica. □

Diamo alcuni esempi.

Esempio 1. Consideriamo la formula F seguente

$$\{\{\neg q, p\}, \{r, p\}, \{\neg p, \neg q\}, \{\neg p, s\}, \{q, \neg r\}, \{q, \neg r\}, \{q, \neg s\}\}.$$

Applicando Risoluzione a $\{\neg q, p\}$ e $\{\neg p, \neg q\}$ (sul letterale p) ottengo

$$\{\neg q\}$$

Applicando risoluzione a $\{\neg q\}$ e $\{q, \neg r\}$ ottengo

$$\{\neg r\}$$

Applicando risoluzione a $\{\neg r\}$ e $\{r, p\}$ ottengo

$$\{p\}$$

Applicando risoluzione a $\{p\}$ e $\{\neg p, s\}$ ottengo

$$\{s\}$$

Applicando risoluzione a $\{s\}$ e $\{q, \neg s\}$ ottengo

$$\{q\}$$

Infine, applicando risoluzione a $\{q\}$ e $\{\neg q\}$ ottengo

$$\square$$

Concludiamo così che $F \in \mathsf{UNSAT}$.

Esempio 2. Vogliamo valutare la correttezza dell'argomento seguente:

$$(p \to q), (q \to r) \models (p \to r).$$

Sappiamo che basta valutare se $(p \to q) \land (q \to r) \land \neg(p \to r)$ è insoddisfacibile. Traduciamo la formule in CNF:

1. $p \to q \equiv \neg p \lor q$
2. $q \to r \equiv \neg q \lor r$
3. $\neg(p \to r) \equiv \neg(\neg p \lor r) \equiv \neg\neg p \land \neg r \equiv p \land \neg r$

La congiunzione che dobbiamo verificare insoddisfacibile è dunque la seguente CNF:

$$(\neg p \lor q) \land (\neg q \lor r) \land p \land \neg r.$$

La sua forma insiemistica è la seguente:

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

Applicando Risoluzione a $\{p\}$ e $\{\neg p, q\}$ ottengo

$$\{q\}$$

Applicando Risoluzione a $\{\neg q, r\}$ e $\{q\}$ ottengo

$$\{r\}$$

Applicando Risoluzione a $\{r\}$ e $\{\neg r\}$ ottengo

$$\square$$

Concludo così che la CNF è insoddisfacibile e dunque l'argomento iniziale è corretto.

Esempio 3. Consideriamo la seguente formula F in CNF:

$$F = \{\{\neg p, \neg q, r\}, \{\neg p, \neg q, s\}, \{\neg p_1, \neg q_1, r_1\}, \{\neg r_1, \neg s, s_1\}, \{p\}, \{q\}, \{q_1\}, \{p_1\}, \{\neg s_1\}\}.$$

Possiamo applicare la risoluzione di clausole partendo dall'insieme di clausole F, scegliendo ogni volta una coppia di clausole a cui è possibile applicare la risoluzione. Otteniamo una sequenza finita di clausole come segue:

$$\{\neg p, \neg q, s\}, \{\neg r_1, \neg s, s_1\}, \{\neg p, \neg q, \neg r_1, s_1\}, \{p\}, \{\neg q, \neg r_1, s_1\}, \{q\}, \{\neg r_1, s_1\}, \{\neg s_\}, \{\neg r_1\},$$
$$\{\neg p_1, \neg q_1, r_1\}, \{\neg p_1, \neg q_1\}, \{q_1\}, \{\neg p_1\}, \{p_1\}, \square.$$

Dal fatto che l'ultima clausola è la clausola vuota, che è insoddisfacibile, e dalla proprietà sopra dimostrata che il risolvente è conseguenza logica delle due clausole premesse, possiamo concludere che F è insoddisfacibile.

3.3 Completezza della Risoluzione (Extra)

Da quanto già dimostrato: **se** esiste una successione ordinata di clausole ottenute partendo dalle clausole in F e applicando la risoluzione di clausole, e che termina con la clausola vuota **allora** la formula F è insoddisfacibile. Ossia: se applicando iterativamente la risoluzione alle clausole di una formula raggiungo la clausola vuota, allora la formula di partenza è insoddisfacibile. Si dice che il metodo di Risoluzione è **corretto**.

Consideriamo la domanda opposta: se F è in UNSAT, è vero che posso certificarlo applicando iterativamente la regola di risoluzione di clausole partendo dalle clausole di F e raggiungendo la clausola vuota? Questa proprietà, se vale, è detta **completezza** (del metodo di Risoluzione): il metodo è capace di certificare tutti i casi di $F \in \mathsf{UNSAT}$.

È opportuno introdurre un concetto formale di *derivazione in Risoluzione* di una clausola C da una formula F, che renda rigorosa l'idea di "applicare iterativamente la regola di Risoluzione a partire da un insieme di clausole".

Definizione 2 (Derivazione/Refutazione in Risoluzione)**.** *Sia $F = \{C_1, \ldots, C_n\}$ una formula (insieme finito di clausole). Una sequenza ordinata di clausole*

$$D_1, D_2, \ldots, D_{k-1}, D_k$$

è una **derivazione** *in Risoluzione della clausola D_k se, per ogni $i \in [1, k]$, o $D_i \in F$ oppure esistono $j, h < i$ tali che $D_i = \mathsf{Res}_\ell(D_j, D_h)$, per qualche letterale ℓ. In altre parole, ogni clausola D_i ha un certificato per appartendere alla sequenza/derivazione: o è una clausola di F oppure deriva da due clausole precedenti per applicazione di risoluzione.*

Se esiste una derivazione della clausola C dalla formula $F = \{C_1, \ldots, C_n\}$ scriviamo $C_1, \ldots, C_n \vdash_{\mathsf{RES}} C$, o $F \vdash_{\mathsf{RES}} C$.

Se l'ultima formula in una derivazione è $\square$ la derivazione è detta una **refutazione** *di F ($F \vdash_{\mathsf{RES}} \square$ in simboli).*

Esempio 4. Gli esempi visti sopra si traducono facilmente nel formato di una derivazione/refutazione. Consideriamo i passi di Risoluzione fatti per dimostrare che $(p \to q), (q \to r) \models (p \to r)$. Abbiamo come prima cosa ottenuto la formula F in CNF:

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

Possiamo ordinarli come segue in modo da rispettare la definizione formale di refutazione:

1. $\{\neg p, q\}$ (clausola in F)
2. $\{\neg q, r\}$ (clausola in F)
3. $\{p\}$ (clausola in F)

4. $\{\neg r\}$ (clausola in F)
5. $\{q\}$ (risoluzione da 1 e 3)
6. $\{r\}$ (risoluzione da 2 e 4)
7. $\square$ (risoluzione da 4 e 6)

La sequenza ordinata di clausole $\{\neg p, q\}, \{\neg q, r\}, \{p\}, \{\neg r\}, \{q\}, \{r\}, \square$ è una refutazione della formula F.

In termini di refutazioni il Teorema di Correttezza si può riformulate come segue:

Teorema 3 (Correttezza)**.** *Se* $F \vdash_{\mathsf{RES}} \square$ *allora* $F \in \mathsf{UNSAT}$.

Il seguente teorema dimostra che il metodo di refutazione in Risoluzione è completo relativamente all'insieme UNSAT: ossia, se una formula è veramente in UNSAT, allora è possibile costruire una sua refutazione in Risoluzione.

Teorema 4 (Completezza)**.** *Se* $F \in \mathsf{UNSAT}$ *allora esiste una refutazione in Risoluzione di* F*, i.e.* $F \vdash_{\mathsf{RES}} \square$.

Dimostrazione. Procediamo per Induzione sul numero di variabili nella formula F.

Base: F ha 0 variabili. Allora F è la formula vuota, ossia $\{\square\}$ (ha solo la clausola vuota come elemento), oppure è del tutto vuota, ossia $F = \varnothing$. Nel primo caso abbiamo che $\square$ è una derivazione di $\square$ da F. Nel secondo caso F è soddisfacibile.

Passo Induttivo: sia F una CNF insoddisfacibile e sia p una variabile proposizionale che compare in F. Dividiamo le clausole di F in due parti:

$$F_p = \{C \in F \,:\, p \text{ compare in } C\}.$$

$$F_{\bar{p}} = \{C \in F \,:\, \bar{p} \text{ compare in } C\}.$$

Definiamo la formula F^- come la formula ottenuta da F togliendo tutte le clausole di F_p e di $F_{\bar{p}}$, ossia:

$$F^- = F \setminus (F_p \cup F_{\bar{p}}).$$

Applichiamo Risoluzione tra tutte le clausole di F_p e di $F_{\bar{p}}$, ossia: se $D \cup \{p\}$ è in F_p e $E \cup \{\bar{p}\}$ è in $F_{\bar{p}}$ otteniamo il risolvente $D \cup E$. Sia R la formula ottenuta unendo a F^- tutti i risolventi ottenuti in questo modo, ossia

$$R = F^- \cup \{D \cup E \,:\, D \cup \{p\} \in F_p \text{ e } E \cup \{\bar{p}\} \in F_{\bar{p}}\}.$$

La formula R ha meno variabili della formula F dunque possiamo applicare l'Ipotesi induttiva: se R è insoddisfacibile allora esiste una refutazione in Risoluzione di R. Dimostriamo che R è insoddisfacibile.

Supponiamo per assurdo che α soddisfi R. Consideriamo α_1 definito come α eccetto che per $\alpha_1(p) = 1$; e α_0 definito come α eccetto che per $\alpha_0(p) = 0$.

Per ipotesi la formula iniziale F è insoddisfacibile dunque α_1 non soddisfa F. Deve esistere una clausola $E \cup \{\bar{p}\}$ in F tale che $\alpha_1(E \cup \{\bar{p}\}) = 0$. Dunque α non soddisfa E.

Analogamente, dato che F è insoddisfacibile per ipotesi, α_0 non soddisfa F. Deve esistere una clausola $D \cup \{p\}$ in F tale che $\alpha_0(D \cup \{p\}) = 0$. Dunque α non soddisfa D.

D'altra parte, poiché α soddisfa R per ipotesi, abbiamo che $\alpha(D \cup E) = 1$. Dunque o α_1 soddisfa F oppure α_0 soddisfa F.

Possiamo ora concludere la dimostrazione osservando che se R ha una refutazione per Risoluzione allora anche F ne ha una: infatti R è stata ottenuta da F applicando un numero finito di volte la regola di Risoluzione! □

Complessivamente i risultati di sopra (correttezza e completezza) dimostrano che l'esistenza di una refutazione in Risoluzione di una formula è **equivalente** al fatto che la formula è insoddisfacibile. In altre parole: le refutazioni in Risoluzione sono un metodo affidabile e infallibile per dimostrare se una formula è UNSAT o SAT.

3.4 Algoritmo di decisione per $F \in$ UNSAT

I risultati visti finora danno luogo a un **algoritmo** per testare se $F \in$ UNSAT, per F una formula in CNF.

L'input dell'algoritmo è una formula F in CNF, ossia $F = \{C_1, \ldots, C_n\}$ dove le C_i sono clausole. L'algoritmo procede come segue, usando F come variabile.

> **while** esistono clausole C_i, C_j in F tali che il loro risolvente R non è già in F
> **do** $F = F \cup \{R\}$.

L'algoritmo termina in numero finito di passi perché esiste solo un numero finito di clausole su un dato insieme di variabili proposizionali.

L'algoritmo è corretto, ossia vale il seguente Teorema.

Teorema 5. *F è* UNSAT *se e soltanto se* □ *è nella formula F al termine dell'algoritmo.*

Dimostrazione. Sappiamo che F è UNSAT se e solo se esiste una refutazione in Risoluzione di F, ossia $F \vdash_{\mathsf{RES}} \square$. L'algoritmo proposto enumera tutte le possibili derivazioni in Risoluzione da F. □

L'algoritmo descritto qui sopra, dovuto orginariamente a Davis e Putnam, dà una procedura meccanica di decisione per l'insoddisfacibilità di una formula proposizionale (in CNF).

L'algoritmo qui sopra, e in generale il metodo di Risoluzione è alla base dei cosidetti SAT-solvers, programmi che risolvono problemi di soddisfacibilità/insoddisfacibilità applicati a problemi industriali, di planning, di Intelligenza Artificiale etc. Per quanto questi algoritmi siano efficienti su molte classi di formule proposizionali, non lo sono in assoluto nel caso pessimo: in particolare è possibile individuare famiglie di formule per cui si può dimostrare che la Risoluzione ha un costo esponenziale.

Una di queste famiglie è basata sul Principio dei Cassetti o Principio della Piccionaia (Pigeon Hole Principle. abbreviato PHP). Il PHP dice che se metto $n+1$ piccioni in n piccionaie almeno una piccionaia conterrà almeno due piccioni. Si può formulare agevolmente la negazione di questo principio come una formula in CNF nel linguaggio con variabili proposizionali $p_{i,j}$ con $1 \leq i \leq n+1$ e $1 \leq j \leq n$ con significato intuitivo di: "il piccione i va nella piccionaia j". L'insieme di clausole è il seguente:

1. Ogni piccione va in almeno una piccionaia: per ogni $1 \leq i \leq n+1$ ho la seguente proposizione:
$$p_{i,1} \vee p_{i,2} \vee \cdots \vee p_{i,n}.$$
2. Nessuna piccionaia contiene due piccioni: per ogni $i \neq i'$, $1 \leq i, i' \leq n+1$, per ogni $1 \leq j \leq n$ ho la proposizione:
$$\neg p_{i,j} \vee \neg p_{i',j}.$$

L'insieme delle clausole definite qui sopra è la formuila F che esprime la negazione del PHP. Dato che F è insoddisfacibile (altrimenti esisterebbe un modo di assegnare $n+1$ piccioni a n piccionaie senza conflitti), sappiamo che $F \vdash_{\mathsf{RES}} \square$. Si può però dimostrare che non esistono refutazioni di F in Risoluzione di dimensione meno che esponenziale in n (il risultato è dovuto ad Haken).