

Model-checking simbolico

Approcci al model-checking (MC)

- **Explicit-State MC:** algoritmo esplora gli stati individualmente
 - ❑ automa di Büchi costruito on-the-fly durante esplorazione
 - ❑ performance ottenuta con una serie di ottimizzazioni (partial order reduction, state compression, etc.)
- **Symbolic MC:** algoritmo esplora insiemi di stati rappresentati simbolicamente (formula logica)
 - ❑ L'insieme di successori di un insieme di stati è calcolato dalla rappresentazione dell'insieme e la rappresentazione delle transizioni
- **Bounded MC:** riduzione a soddisfacibilità di formule Booleane (SAT)
 - ❑ Limite sul numero di passi codificati

Algoritmo simbolico per raggiungibilità

- $A = (\Sigma, Q, Q_{in}, \delta)$ macchina a stati finiti
- T : insieme target
- algoritmo manipola insiemi di stati (forward reachability):
 - Inizializzazione: $Q_0 = Q_{in}$
 - Passo: $Q_{i+1} = Q_i \cup \{ q' \mid \exists q \in Q_i, (q, a, q') \in \delta \}$
 - Terminazione:
 - Se $Q_i \cap T \neq \emptyset$, termina con risposta positiva
 - Se $Q_{i+1} == Q_i$ termina con risposta negativa
- si può ottenere un altro algoritmo (backward reachability) partendo dal target T e visitando gli stati a ritroso fino a raggiungere stato iniziale oppure completata esplorazione stati raggiungibili

Insiemi di stati come formule booleane

- per implementare efficientemente algoritmi simbolici occorre una struttura dati efficiente per
 - rappresentare in maniera compatta insiemi di stati
 - facilitare le tipiche operazioni su insiemi (unione, intersezione, test uguaglianza, test vuoto, etc.)
- Insiemi di stati possono essere rappresentati come formule booleane
 - stati sono rappresentati da vettori binari (corrispondenti a valutazioni delle variabili)
 - uno stato appartiene ad un insieme rappresentato da formula φ se corrisponde a valutazione soddisfacente φ

Visita simbolica con formule

- $R_0(X) = Q_{in}(X)$
- $R_{i+1}(X) = R_i(X) \vee \exists Z. (R_i(Z) \wedge T(Z,X))$

dove:

- X, Z sono vettori binari,
- R_j con $j \geq 0$, e Q_{in} sono predicati sui vettori (insiemi) e
- T è una relazione binaria su vettori (relazione di transizione)

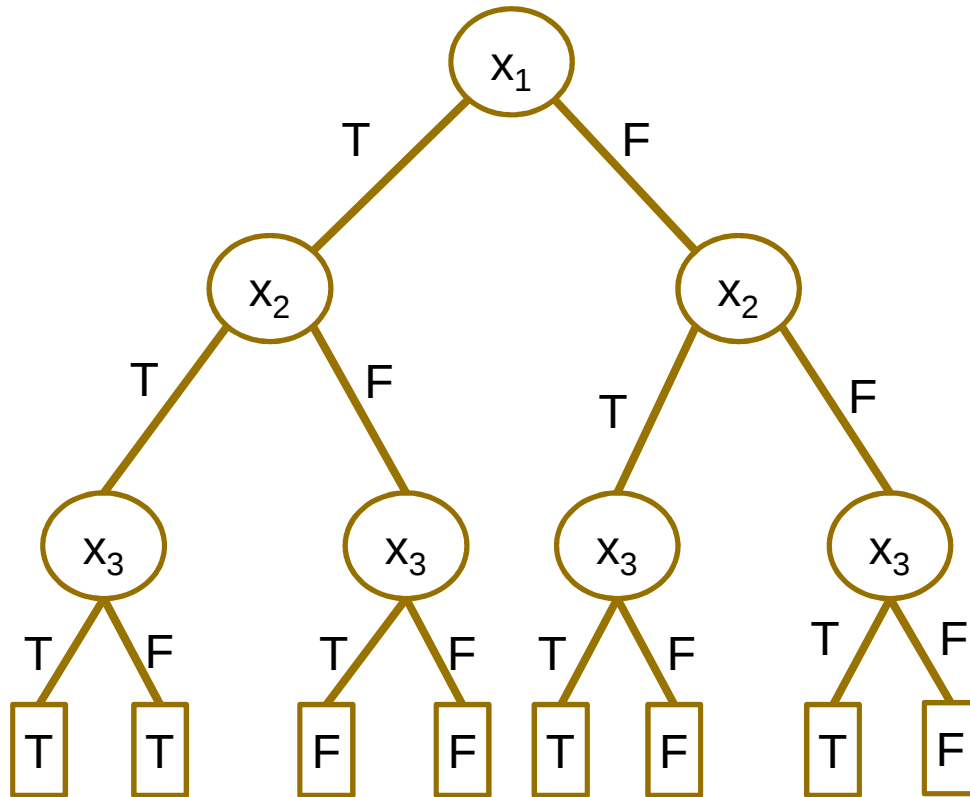
Rappresentazione compatta di formule

- formula $\varphi = (x_1 \wedge x_2) \vee (\neg x_1 \wedge x_3)$
- una formula può essere vista come una funzione booleana e quindi definita attraverso una tavola di verità:

x_1	x_2	x_3	φ
F	F	F	F
F	F	T	T
F	T	F	F
F	T	T	T
T	F	F	F
T	F	T	F
T	T	F	T
T	T	T	T

- ridondante:
 - formula vera se entrambi x_1 e x_2 veri oppure x_1 falso e x_3 vero
- utilizziamo un albero di decisione binario

Albero di decisione binario



- corrisponde alla tavola di verità
- ogni livello (eccetto foglie) corrisponde ad una variabile
- archi uscenti etichettate con valore variabile
- valutazioni corrispondono a cammini dalla radice alle foglie
- valore foglie = valore formula

Diagramma di decisione binario

- possiamo compattare ridondanze
- ad es., non servono tutte le foglie
 - basta solo un rappresentante per T e uno per F

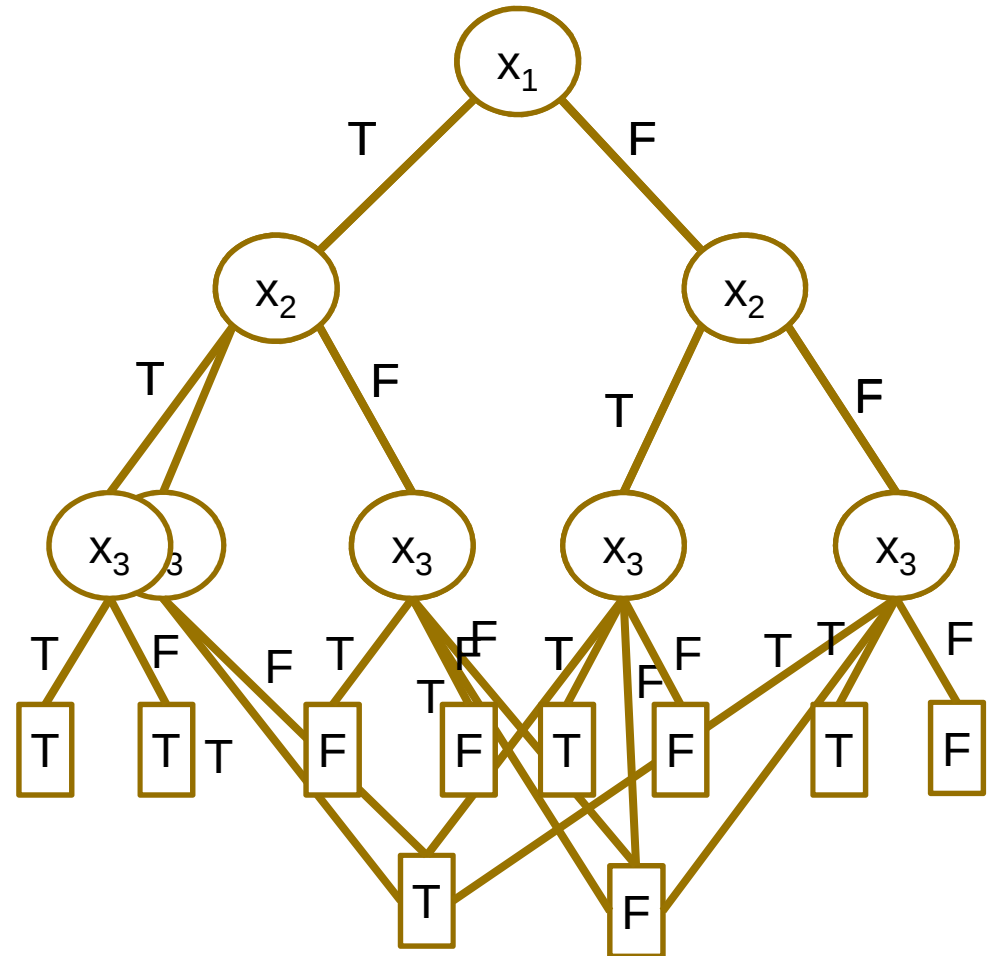


Diagramma di decisione binario

- nodo x_3 più a sinistra collegato a foglia T indipendentemente dal valore assegnato
- stesso per secondo nodo stesso livello
- possiamo rimuoverli e collegare archi da x_2 direttamente alle foglie

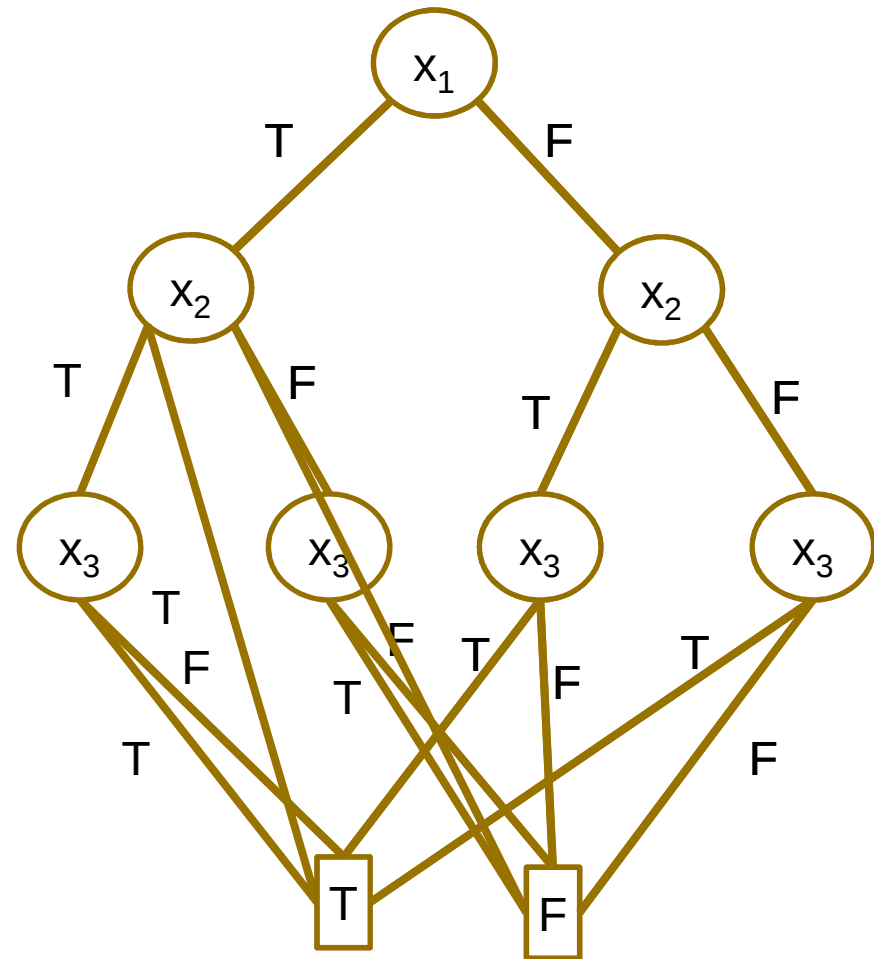


Diagramma di decisione binario

- i sottografi radicati in x_3 coincidono
- possiamo accorparli in un unico sottografo

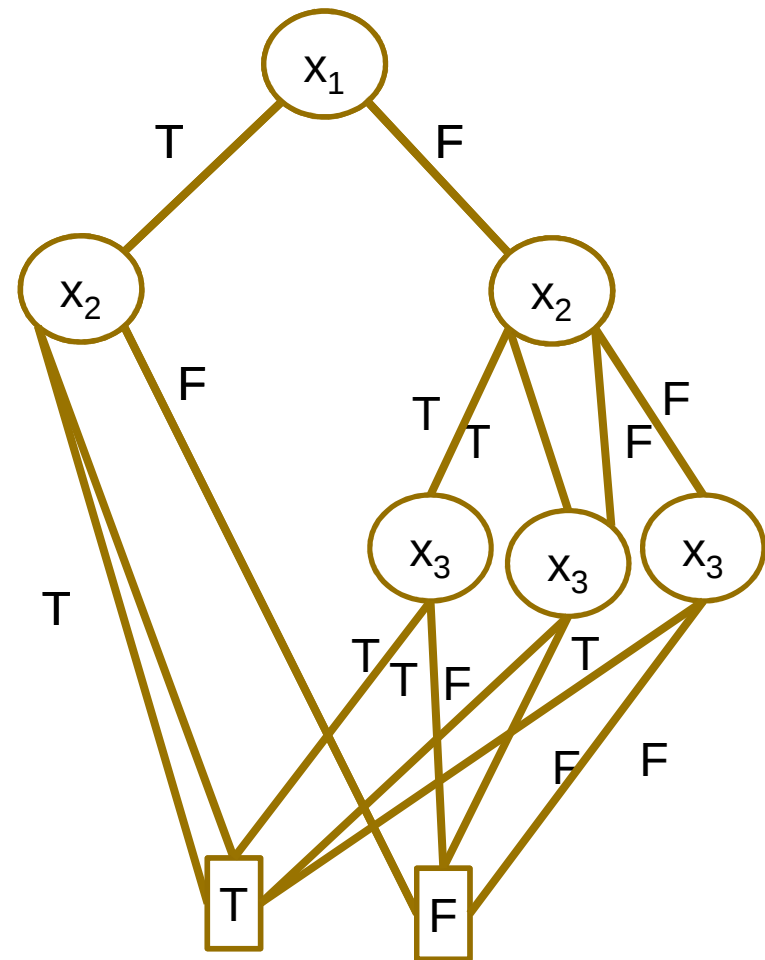


Diagramma di decisione binario

- archi uscenti da nodo x_2 più a destra entrano nello stesso nodo
- possiamo rimuovere questo nodo
- non sono possibili altre semplificazioni
- diagramma finale ha solo 5 nodi contro i 15 di quello iniziale

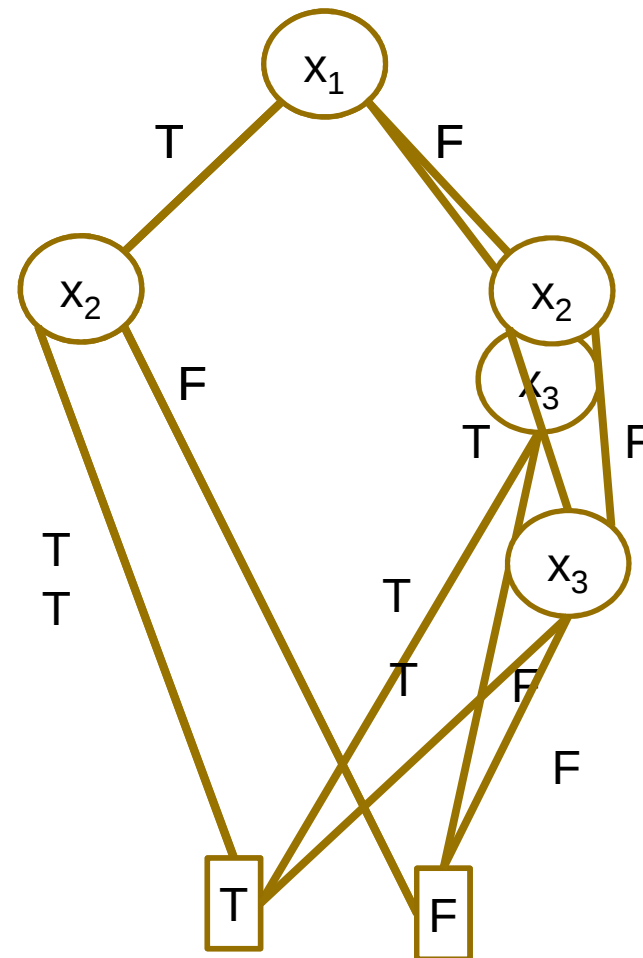


Diagramma di decisione binario (BDD)

- BDD: grafo aciclico direzionato (DAG) t.c.
 - nodi pozzo (**foglie**) sono etichettati con T (true) e F (false)
 - gli altri nodi (**nodi interni**) con variabili booleane
 - esiste un unico nodo sorgente (**radice**)
 - ogni nodo interno ha due archi uscenti:
 - uno etichettato con T (**T-arco**) e l'altro con F (**F-arco**)
 - su ogni cammino dalla radice alle foglie, le variabili sono incontrate sempre nello stesso ordine (ordinamento delle variabili fissato per ogni BDD)
- Nota: cammini di un BDD corrispondono a funzioni di assegnamento delle variabili

Notazione

- Sia $V = \langle v_1, v_2, \dots, v_n \rangle$ una sequenza ordinata di variabili
- Una tupla (v_n, C, D) denota un BDD B su $\langle v_1, v_2, \dots, v_n \rangle$ t.c. :
 - la radice di B è etichettata con v_n
 - C è il BDD su $\langle v_1, v_2, \dots, v_{n-1} \rangle$ la cui radice è collegata alla radice di B con un F-arco
 - D è il BDD su $\langle v_1, v_2, \dots, v_{n-1} \rangle$ la cui radice è collegata alla radice di B con un T-arco

Nota che C e D non sono necessariamente disgiunti (in un BDD minimale hanno sicuramente in comune almeno le foglie)

- $\dim(B)=n$
- f_B denota una formula corrispondente a B

Definizione induttiva dei BDD

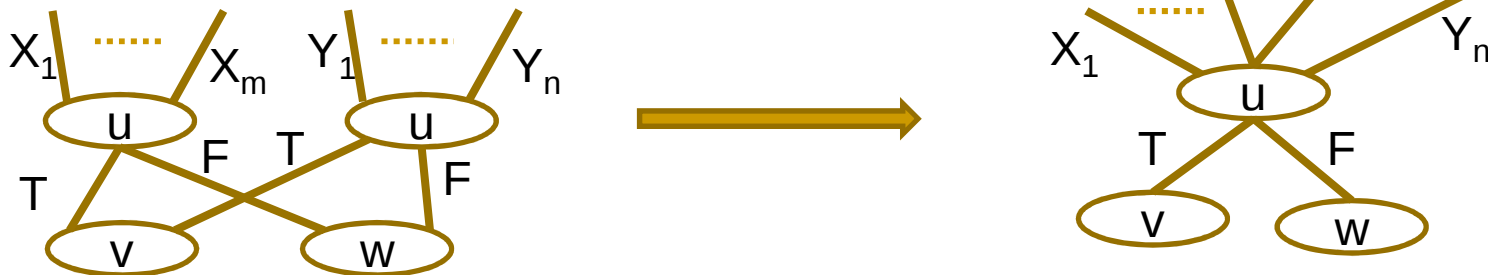
- Sia $V = \langle v_1, v_2, \dots, v_n \rangle$ una sequenza ordinata di variabili
- Un BDD è :
 - un nodo singolo F (rappresenta la formula **false**)
 - un nodo singolo T (rappresenta la formula **true**)
 - se B and C sono due BDD sulla sequenza $\langle v_1, v_2, \dots, v_{i-1} \rangle$ allora il DAG $D=(v_i, B, C)$ è un BDD su $\langle v_1, v_2, \dots, v_i \rangle$ e rappresenta la formula:

$$(\neg v_i \wedge f_B) \vee (v_i \wedge f_C)$$

BDD minimali: riduzione e canonicità

- Fissato l'ordine delle variabili, per ogni funzione booleana esiste un unico BDD corrispondente e si può ottenere applicando le seguenti regole:
 - rimpiazzare foglie con unico rappresentante per T e per F
 - iterativamente a partire dalle foglie (fino a saturazione):

- accorpamento sottografi identici



- eliminazione nodi interni superflui



AND di BDD

- Dati due BDD B and C , il BDD $D=B \otimes C$ per $f_B \wedge f_C$ si ottiene ricorsivamente:
 - se $\dim(B) = \dim(C)=0$, allora $\dim(D)=0$ e D è il singolo nodo T se e solo se C e D sono entrambi T
 - se $\dim(B) = \dim(C)>0$, $B = (v_i, l_B, r_B)$ e $C = (v_i, l_C, r_C)$ allora $D=(v_i, l_B \otimes l_C, r_B \otimes r_C)$
 - se $\dim(B) > \dim(C)$ e $B = (v_i, l_B, r_B)$ allora
$$D=(v_i, l_B \otimes C, r_B \otimes C)$$

Nota che $B \otimes C$ è in genere non minimale anche se B e C sono minimali

OR di BDD

- Dati due BDD B and C , il BDD $D=B\oplus C$ per $f_B \vee f_C$ si ottiene ricorsivamente:
 - se $\dim(B) = \dim(C)=0$, allora $\dim(D)=0$ e D è il singolo nodo F se e solo se C e D sono entrambi F
 - se $\dim(B) = \dim(C)>0$, $B = (v_i, l_B, r_B)$ e $C = (v_i, l_C, r_C)$ allora $D=(v_i, l_B \oplus l_C, r_B \oplus r_C)$
 - se $\dim(B) > \dim(C)$ e $B = (v_i, l_B, r_B)$ allora
$$D=(v_i, l_B \oplus C, r_B \oplus C)$$

Nota che $B \oplus C$ è in genere non minimale anche se B e C sono minimali

NOT e Complessità AND e OR

- Per calcolare, il BDD per la negazione di una formula basta scambiare le etichette delle foglie (T diventa F e F diventa T)
 - richiede tempo costante
- Utilizzando programmazione dinamica, AND e OR di BDD possono essere fatti risolvendo $O(|B|.|C|)$ sottoproblemi
 - per ogni nodo u in B e v in C , (u,v) è risolto una volta
 - si mantengono i risultati per ogni coppia (r,s)
- Quindi, l'algoritmo richiede tempo $O(|B|.|C|)$ e il BDD ottenuto ha taglia $O(|B|.|C|)$.

Restrizione di una variabile a un valore

Per una formula f con n variabili,

$f \downarrow_{(x=b)}$ è la formula con $n-1$ variabili ottenuta da f assegnando x con b

Dato un BDD per f , calcola il BDD per $f \downarrow_{(x=b)}$ come segue:

- visita BDD per f
- redireziona ogni arco che punta a un nodo v etichettato x al nodo u tale che (v,u) è un b -arco



Quantificazione

Si consideri una formula f .

- Quantificazione esistenziale:

$$\exists v.f = f \downarrow_{(v=F)} \vee f \downarrow_{(v=T)}$$

dato un BDD per f , possiamo calcolare un BDD for $\exists v.f$ in tempo $O(n^2)$

- Quantificazione universale (duale):

$$\forall v.f = f \downarrow_{(v=F)} \wedge f \downarrow_{(v=T)}$$

dato un BDD per f , possiamo calcolare un BDD for $\forall v.f$ in tempo $O(n^2)$

AndExists

- nell'algoritmo simbolico che risolve la reachability occorre calcolare

$$\exists Z. (R_i (Z) \wedge T(Z,X))$$

- possiamo combinare gli algoritmo per il calcolo dell'AND e della quantificazione esistenziale così che i BDD da memorizzare hanno solo $|X|$ variabili (invece di $2|X|$ variabili)
 - T è la relazione di transizione data, è costante
 - si devono calcolare BDD che rappresentano insiemi di stati raggiungibili

Model-checking con BDD

```
Reachability( X, Q_in(X), T(X,X'), F(X)) {  
    // X - vars; Q_in, T(X,X') e F sono BDD  
  
    R:=0; R' = Q_in;  
    do {  
        R = R';  
        R' = R  $\vee$   $\exists Z. ( R(Z) \wedge T(Z,X) )$  ;  
    } while (R $\neq$ R' and R $\wedge$ F =  $\emptyset$ );  
  
    if (R $\wedge$ F =  $\emptyset$ )  
        return "unreachable"  
    else  
        return "reachable";  
}
```

Model-checking con BDD

```
Safety( X, Q_in(X), T(X,X'), F(X)) {  
    // X - vars; Q_in, T(X,X') e F sono BDD  
  
    R:=0; R' = Q_in;  
    do {  
        R = R';  
        R' = R  $\vee$   $\exists Z. ( R(Z) \wedge T(Z,X) )$  ;  
    } while (R $\neq$ R' and R $\wedge$ F =  $\emptyset$ );  
  
    if (R $\wedge$ F =  $\emptyset$ )  
        return "safe"  
    else  
        return "unsafe";  
}
```

Modifichiamo reachability

```
ReachableSet( X, Q_in(X), T(X,X') ){  
    // X - vars; Q_in e T(X,X') sono BDD  
    R:=0;  
  
    R' =  $\exists Z. (Q\_in(Z) \wedge T(Z,X) )$  ;  
    // stati iniziali compaiono in R solo se possono essere  
    // raggiunti nuovamente  
  
    do {  
        R = R' ;  
        R' = R  $\vee \exists Z. ( R(Z) \wedge T(Z,X) )$  ;  
    } while (R $\neq$ R') ;  
  
    return R;  
}
```

Model-checking con BDD

```
Buchi( X, Gin(X), T(X,X'), F(X)) {  
    // X - vars; Gin , T(X,X') e F sono BDD  
    F' = F  $\wedge$  ReachableSet(X, Qin(X), T(X,X'));  
    do {  
        F = F';  
        F' = F  $\wedge$  ReachableSet(X, F(X), T(X,X'));  
    } while (F  $\neq$  F');  
  
    if (F =  $\emptyset$ )  
        return "unsatisfied"  
    else  
        return "satisfied";  
}
```

Implementazione di BDD

Alcuni BDD packages disponibili:

- CuDD --- Fabio Somenzi, Colorado Univ.
- VIS --- Colorado, Berkeley

Numerose euristiche implementate in pratica:

- Forward/Backward
 - Ordinamento delle variabili
 - Supporto diretto per domini finiti (MDD)
 - Partizionamento delle transizioni/ della rete
 - Scelta frontiere
-

Ordinamento delle variabili

- aspetto cruciale per la prestazione di algoritmi che usano BDD
- influenza sensibilmente la taglia dei BDD
 - scegliendo il giusto ordinamento, taglia BDD può essere esponenzialmente più succinto
 - Es. $(x_1 \vee y_1) \wedge \dots \wedge (x_n \vee y_n)$
BDD esponenziale in n se ordinamento $\langle x_1, \dots, x_n, y_1, \dots, y_n \rangle$
lineare se ordinamento $\langle x_1, y_1, \dots, x_n, y_n \rangle$
 - non sempre è possibile avere BDD di taglia trattabile (esistono formule per le quali la taglia del BDD è esponenziale nel numero di variabili indipendentemente dall'ordinamento)

Osservazioni

- Il model-checking simbolico con **BDD** ha avuto molto successo nella verifica dell'**hardware**
 - **BDD** non sono ampiamente usati nella verifica del **software**
 - difficili da combinare con partial-order reduction (insieme all'astrazione principale artefice dei successi in software verification)
 - difficile modellare l'allocazione dinamica della memoria
 - in software verification approccio simbolico fa uso di esecuzioni simboliche e vincoli (constraints)
 - Model checker simbolico: nuSMV
-