BASE DI CONOSCENZA → INSIEME DI FORMULE

↳ L'AGENTE INTERAGISCE CON LA KB MEDIANTE UN INTERFACCIA FUNZIONALE TELL - ASK

- TELL → PER AGGIUNGERE NUOVI FATTI ALLA KB
- ASK → PER INTERROGARE LA KB

Function Agente-KB (*percezione*) **returns** *un'azione*
persistent: *KB*, una base di conoscenza
t, un contatore, inizialmente a 0, che indica il tempo
TELL(*KB*, Costruisci_Formula_Percezione(*percezione*, *t*)) → COMUNICA I FATTI (PERCEZIONE RICEVUTA)
azione ← ASK(*KB*, Costruisci_Query_Azione(*t*)) → CHIEDE AZIONE DA ESEGUIRE
TELL(*KB*, Costruisci_Formula_Azione(*azione*, *t*))
t ← *t* + 1
return *azione*

APPROCCIO DICHIARATIVO → INIZIA CON UNA BASE DI CONOSCENZA VUOTA, IL PROGETTISTA ATTRAVERSO LA TELL INSERISCE FORMULE UNA X UNA FINCHE L'AGENTE NON SA COME OPERARE

APPROCCIO PROCEDURALE → CODIFICA IN UN PROGRAMMA I COMPORTAMENTI DESIDERATI SOTTO FORMA DI CODICE

ESEMPIO

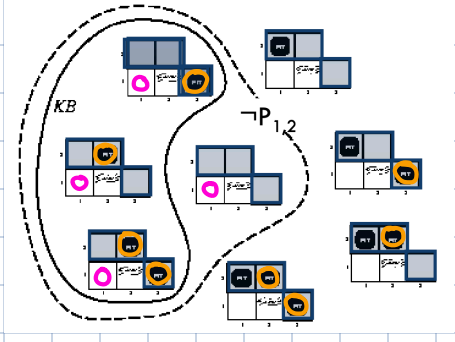
REGOLE WUMPUS :
- STENCH NELLE CASELLE VICINE AL WUMPUS
- BREZZA NELLE CASELLE VICINE AD UNA BUCA/POZZO

$KB \models \alpha$ → KB IMPLICA LOGICAMENTE α

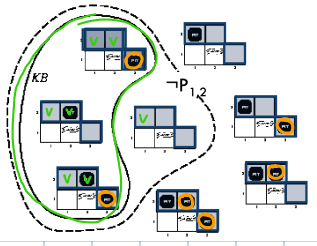
$KB = \{ B_{2,1} , \neg B_{1,1} + \text{REGOLE DEL WUMPUS} \}$ ($B_{2,1}$: BREZZA)

$KB \models \neg P_{1,1}$

$KB \not\models \neg P_{2,2}$



$KB = \{ B_{2,1} , \neg B_{1,1} \}$



MODEL CHECKING

SIA KB LA BASE DI CONOSCENZA, $KB \models \alpha$?

↳ SI ENUMERANO TUTTE LE POSSIBILI INTERPRETAZIONI DI KB, SE k SONO I SIMBOLI USATI CI SARANNO 2^k POSSIBILI MODELLI.

↳ PER CIASCUNA INTERPRETAZIONE (MODELLO):

- SE NON SODDISFA KB → VIENE IGNORATA
- SE SODDISFA KB SI CONTROLLA CHE SODDISFA ANCHE α.

function TT-ENTAILS?(KB, α) **returns** *true* or *false*
  inputs: KB, the knowledge base, a sentence in propositional logic
    α, the query, a sentence in propositional logic

  symbols ← a list of the proposition symbols in KB and α
  return TT-CHECK-ALL(KB, α, *symbols*, { })

---

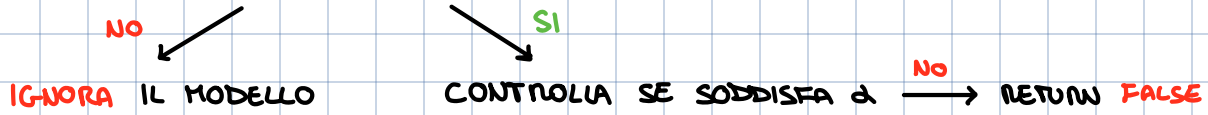
function TT-CHECK-ALL(KB, α, *symbols*, *model*) **returns** *true* or *false*
  if EMPTY?(*symbols*) **then**
    if PL-TRUE?(KB, *model*) **then return** PL-TRUE?(α, *model*)
    else return *true* // *when KB is false, always return true*
  else do
    $P \leftarrow$ FIRST(*symbols*)
    rest ← REST(*symbols*)
    return (TT-CHECK-ALL(KB, α, *rest*, *model* $\cup \{P = true\}$)
      and
      TT-CHECK-ALL(KB, α, *rest*, *model* $\cup \{P = false\}$))

INPUT:

- BASE DI CONOSCENZA KB
- FORMULA α
- ELENCO SIMBOLI PROPOSIZIONALI ANCORA DA ASSEGNARE
- INTERPRETAZ. PARZIALE DEI SIMBOLI

PROCEDURA RICORSIVA:

- QUANDO HA FINITO DI ASSEGNARE I SIMBOLI:
  - CONTROLLA SE L'INTERPRETAZIONE SODDISFA KB



- SE CI SONO SIMBOLI DA ASSEGNARE
  - GENERA 2 RAMI, UNO VERO & UNO FALSO
  - ESPLORA ENTRAMBI I RAMI

- TT-CHECK-ALL((¬A∨B)∧(A∨C), (B∨C), [A, B, C], [ ])

A=t
- TT-CHECK-ALL((¬A∨B)∧(A∨C), (B∨C), [B, C], [A=t])

  B=t
  - TT-CHECK-ALL((¬A∨B)∧(A∨C), (B∨C), [C], [A=t; B=t])
    - TT-CHECK-ALL((¬A∨B)∧(A∨C), (B∨C), [ ], [A=t; B=t; C=t]) OK
    - TT-CHECK-ALL((¬A∨B)∧(A∨C), (B∨C), [ ], [A=t; B=t; C=f]) OK

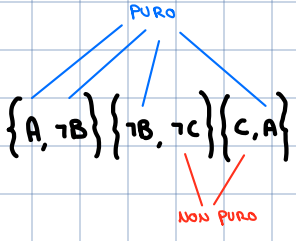
  B=f
  - TT-CHECK-ALL((¬A∨B)∧(A∨C), (B∨C), [C], [A=t; B=f])
    - TT-CHECK-ALL((¬A∨B)∧(A∨C), (B∨C), [ ], [A=t; B=f; C=t]) OK %not a model for KB
    - TT-CHECK-ALL((¬A∨B)∧(A∨C), (B∨C), [ ], [A=t; B=f; C=f] OK %not a model for KB

A=f
- TT-CHECK-ALL((¬A∨B)∧(A∨C), (B∨C), [B, C], [A=f])
  - TT-CHECK-ALL((¬A∨B)∧(A∨C), (B∨C), [C], [A=f; B=t])
    - TT-CHECK-ALL((¬A∨B)∧(A∨C), (B∨C), [ ], [A=f; B=t; C=t]) OK
    - TT-CHECK-ALL((¬A∨B)∧(A∨C), (B∨C), [ ], [A=f; B=t; C=f]) OK %not a model for KB
  - TT-CHECK-ALL((¬A∨B)∧(A∨C), (B∨C), [C], [A=f; B=f])
    - TT-CHECK-ALL((¬A∨B)∧(A∨C), (B∨C), [ ], [A=f; B=f; C=t]) OK
    - TT-CHECK-ALL((¬A∨B)∧(A∨C), (B∨C), [ ], [A=f; B=f; C=f]) OK %not a model for KB

DPLL

SI UTILIZZA PER DETERMINARE SE UNA FORMULA IN CNF É SODDISFACIBILE, CI SONO 3 MIGLIORAMENTI RISPETTO A TT-ENTAILS:

- TERMINAZIONE ANTICIPATA
  - ↳ SE ALMENO UNA CLAUSOLA É FALSA, L'ALGORITMO TERMINA.
- SIMBOLI PURI
  - ↳ UN SIMBOLO CON STESSO SEGNO IN TUTTE LE CLAUSOLE



- CLAUSOLE UNITARIE
  - ↳ QUANDO LA CLAUSOLA DIPENDE DA UN SOLO LETTERALE

{B, ¬C} è unitaria quando C = *True*

↳ C QUINDI É FALSA, ORA LA CLAUSOLA DIPENDE SOLO DA B DEVE ESSERE TRUE.

WALK SAT

UTILIZZATO PER RISOLVERE PROBLEMI DI SODDISFACIBILITÀ BOOLEANA. ESPLORA LO SPAZIO DELLE SOLUZIONI PASSEGGIANDO NELLO SPAZIO DELLE POSSIBILI ASSEGNAZIONI.

INPUT → FORMULA IN CNF $(v\ v) \wedge (v\ v) \ldots$

1. INIZIALIZZAZIONE
   - ↳ ASSEGNA UN VALORE DI VERITÀ T o F A CIASCUNA VARIABILE. (SOLUZIONE CORRENTE)
2. CICLO PRINCIPALE, RIPETI FINCHÉ NON SODDISFA TUTTE LE CLAUSOLE O RAGGIUNGE UN LIMITE DI ITERAZIONI.
   - Sceglie a caso una clausola non ancora soddisfatta
   - Sceglie un simbolo da modificare (*flip*) scegliendo con probabilità *p* (di solito 0,5) tra una delle due:
     - Sceglie un simbolo a caso (passo casuale)
     - Sceglie quello che rende più clausole soddisfatte (passo di ottimizzazione, simile a *min-conflicts*)

SE MAX-FLIPS = ∞ E L'INSIEME DELLE CLAUSOLE É SODDISFACIBILE PRIMA O POI TERMINA, SE INSODDISFACIBILE NON TERMINA.

NON PUÓ ESSERE USATO PER VERIFICARE L'INSODDISFACIBILITÀ.

PSEUDOCODICE

```
function WalkSAT(clauses, p, max-flips) returns a satisfying model or failure
    inputs: clauses, a set of clauses in propositional logic
            p, the probability of choosing to do a "random walk" move
            max-flips, number of flips allowed before giving up

    model ← a random assignment of true/false to the symbols in clauses
    for i = 1 to max-flips do
        if model satisfies clauses then return model
        clause ← a randomly selected clause from clauses that is false in model
        with probability p flip the value in model of a randomly selected symbol
                from clause
        else flip whichever symbol in clause maximizes the number of satisfied clauses
    return failure
```

CONSEGUENZA LOGICA $\models$

UNA FORMULA A É CONSIDERATA CONSEGUENZA LOGICA DI UN INSIEME DI FORMULE KB SE E SOLO SE IN OGNI MODELLO IN CUI KB É VERA ANCHE A É VERA. SI INDICA CON:

$KB \models A$

POSSIAMO USARE IL MODEL-CHECKING PER VERIFICARE UNA CONSEGUENZA LOGICA.

DEDUZIONE LOGICA $\vdash$

SI BASA SULL' APPLICAZIONE DI REGOLE DI INFERENZA PER DERIVARE UNA FORMULA A PARTENDO DA UN INSIEME DI FORMULE KB. SI USANO PROCEDURE DI INFERENZA COME IL MODUS PONENS E L'ELIMINAZIONE DELL' AND.

DEDUZIONE → SEQUENZA DI PASSAGGI IN CUI OGNI PASSO É GIUSTIFICATO DA UNA REGOLA D'INFERENZA, LA CONCLUSIONE FINALE É LA FORMULA CHE SI VUOLE DIMOSTRARE. SI INDICA CON:

$KB \vdash A$