The problem

- **Problem:** Given a set of action operators $OP$, (a representation of) an initial state $I$ and goal state $G$, find a sequence of operator applications $o_1, \ldots, o_n$, leading from the initial state to the goal state.

- **Idea:** Encode it into a model checking problem.
Example

\textbf{Init:} \quad \text{On}(A, B), \text{On}(B, C), \text{On}(C, T), \text{Clear}(A)

\textbf{Goal:} \quad \text{On}(C, B), \text{On}(B, A), \text{On}(A, T)

\textbf{Move}(b, s, d)

\textbf{Precond:} \quad \text{Block}(b) \land \text{Clear}(b) \land \text{On}(b, s) \land \text{Clear}(d) \lor \text{Table}(d) \land b \neq s \land b \neq d \land s \neq d

\textbf{Effect:} \quad \text{Clear}(s) \land \neg \text{On}(b, s) \land \text{On}(b, d) \land \neg \text{Clear}(d)
Encoding in SMV

- **Initial states:**

  \[\text{On}(A, B) \land \text{On}(B, C) \land \text{On}(C, T) \land \text{Clear}(A).\]

- **Goal states:**

  \[\text{On}(C, B) \land \text{On}(B, A) \land \text{On}(A, T).\]

- **Action preconditions and effects:**

  \[
  \text{Move}(A, B, C) \rightarrow \\
  \text{Clear}(A) \land \text{On}(A, B) \land \text{Clear}(C) \land \\
  \text{Clear}(B') \land \neg \text{On}(A', B') \land \\
  \text{On}(A', C') \land \neg \text{Clear}(C').
  \]
Planning strategy

- **Specification:** The goal is not reachable.
- **Plan:** If the property is false, NuSMV produces a counterexample. The counterexample is a plan to reach the goal.
The Tower of Hanoi - Variables

MODULE main
-- Hanoi problem with three poles (left, middle, right)
-- and four ordered disks d1, d2, d3, d4,
-- disk d1 is the biggest one
VAR
d1 : {left,middle,right};
d2 : {left,middle,right};
d3 : {left,middle,right};
d4 : {left,middle,right};
move : 1..4; -- possible moves
DEFINE
  move_d1 := move=1;
  move_d2 := move=2;
  move_d3 := move=3;
  move_d4 := move=4;
--- di is clear iff di!=dj for every j>i

**DEFINE**

clear_d1 :=
  d1!=d2 &
  d1!=d3 &
  d1!=d4;
clear_d2 :=
  d2!=d3 &
  d2!=d4;
clear_d3 :=
  d3!=d4;
clear_d4 := 1;
The Tower of Hanoi - Initial states

-- initially all items are on the left pole

INIT

d1 = left &
d2 = left &
d3 = left &
d4 = left;

TRANS

move_d1 ->

-- only d1 changes
  next(d1) != d1 &
  next(d2) = d2 &
  next(d3) = d3 &
  next(d4) = d4 &

-- no other disks on d1
  clear_d1 &

-- no smaller disks on the next pole
  next(d1) != d2 &
  next(d1) != d3 &
  next(d1) != d4

...
-- spec to find a solution to the problem

SPEC

! EF (d1=right & d2=right & d3=right & d4=right)

> NuSMV hanoi4.smv

*** This is NuSMV 2.3.0 (compiled on Mon Oct 24 13:36:47 UTC 2005)
*** For more information on NuSMV see <http://nusmv.irst.itc.it>
*** or email to <nusmv-users@irst.itc.it>.
*** Please report bugs to <nusmv@irst.itc.it>.

-- specification !EF ((d1 = right & d2 = right) & d3 = right) & d4 = right) is false
-- as demonstrated by the following execution sequence

Trace Description: CTL Counterexample
Trace Type: Counterexample

-> State: 1.1 <-
  d1 = left
d2 = left
d3 = left
d4 = left
move = 4
clear_d4 = 1
clear_d3 = 0
clear_d2 = 0
clear_d1 = 0
move_d4 = 1
move_d3 = 0
move_d2 = 0
move_d1 = 0
...

Alberto Griggio (DISI)
The tic-tac-toe puzzle is modeled with an array of size nine.

```
 1 | 2 | 3
____|___|____
 4 | 5 | 6
____|___|____
 7 | 8 | 9
|   |
```
VAR
    B : array 1..9 of {0,1,2};
-- initially, all squares are empty
INIT
    B[1] = 0 &
    B[2] = 0 &
    B[3] = 0 &
    B[4] = 0 &
    B[5] = 0 &
    B[6] = 0 &
    B[7] = 0 &
    B[8] = 0 &
    B[9] = 0;
-- let us assume that player 1 is the first player
-- players move alternatively

VAR
    player : 1..2;

ASSIGN
    init(player) := 1;
    next(player) := case
        player = 1 : 2;
        player = 2 : 1;
    esac;
Tic-Tac-Toe - The moves

-- move=0 means no move
-- move=i with i>0 means the current player fills B[i]
VAR move : 0..9;
INIT move=0
TRANS
  next(move=0) ->
    next(B[1])=B[1] &
    next(B[3])=B[3] &
    next(B[4])=B[4] &
    next(B[5])=B[5] &
    next(B[6])=B[6] &
    next(B[7])=B[7] &
    next(B[8])=B[8] &
    next(B[9])=B[9]

...
Tic-Tac-Toe - The end of the game

-- "win1" means player 1 wins
-- "win2" means player 2 wins
-- "draw" means nobody wins

DEFINE


win2 := ...
Tic-Tac-Toe - The end of the game

draw := !win1 & !win2 &

TRANS
  (win1 | win2 | draw) <-> next(move) = 0
-- SPECIFICATIONS

-- PLAYER 2

-- player 2 does not have a "winning" strategy
-- SPECIFICATIONS

-- PLAYER 2

-- player 2 does not have a "winning" strategy

SPEC
  ! (AX (EX (AX (EX (AX (EX (AX (EX (AX win2))))))))))
-- SPECIFICATIONS

-- PLAYER 2

-- player 2 does not have a "winning" strategy

SPEC
  ! (AX (EX (AX (EX (AX (EX (AX (AX win2))))))))

-- player 2 has a "non-losing" strategy
-- SPECIFICATIONS

-- PLAYER 2

-- player 2 does not have a "winning" strategy

SPEC
  ! (AX (EX (AX (EX (AX (EX (AX (EX (AX win2))))))))))]]]]]

-- player 2 has a "non-losing" strategy

SPEC
  AX (EX (AX (EX (AX (EX (AX (EX (AX !win1))))))))))]

...
Tic-Tac-Toe - Let’s play

Suppose player one fills 5:

\[
\rightarrow \text{State: } 2.3 \leftarrow B[9] = 2 \text{ player } = 1 \text{ move } = 9 \ldots
\]

Player two may fill 9.
-- player 2 has also a "non-winning" strategy
-- player 2 does not have a "losing" strategy
-- player 2 does not have a "drawing" strategy
-- player 2 has a "non-drawing" strategy
-- player 1 does not have a "winning" strategy
-- player 1 has a "non-losing" strategy
-- player 1 has also a "non-winning" strategy
-- player 1 does not have a "losing" strategy
-- player 1 does not have a "drawing" strategy
-- player 1 has a "non-drawing" strategy