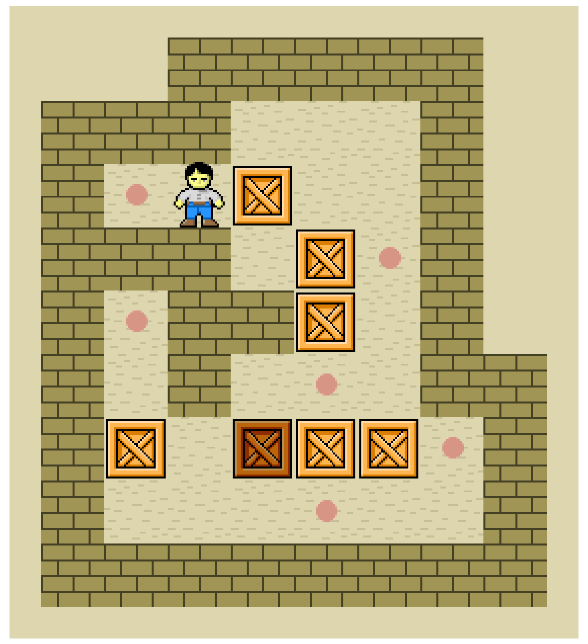
# Sequential System Solving with Modern SAT Solver

## The Sokoban Puzzle – A P-space complete/NP-hard Problem

Given an **initial configuration**, the player is only allowed to **push** the boxes until all targets are covered.

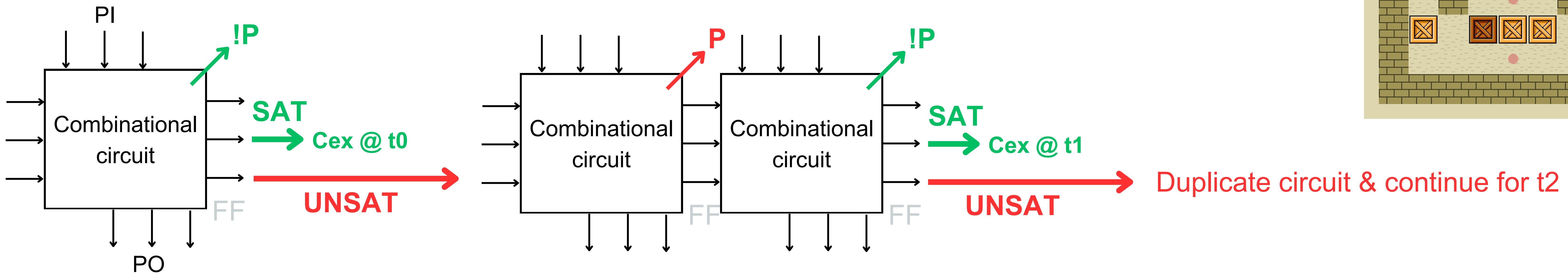
Movement is limited to **horizontal** and **vertical**, one cell at a time.



## Encode as a Bounded Model Checking problem

(1) unroll the **transition relation** of the system **T** times

(2) check existence of valid solution with **increasing bound** T



Liveness property **== solved state property**

A combinational circuit **== transition relation formally encoded as CNF constraints:**

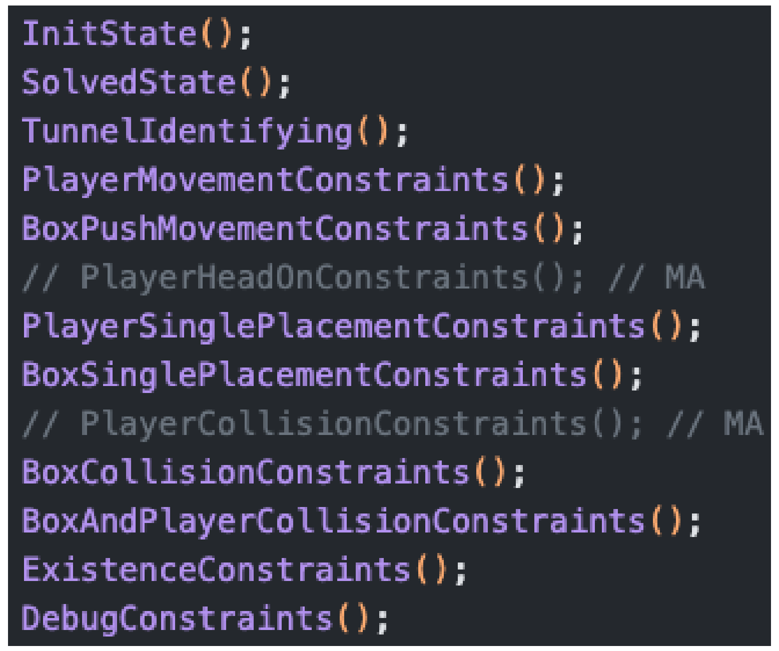
$$U = I(0) \land \bigwedge_{k=0}^{T-1} TR(k, k+1) \land G(T)$$

I(0): initial state constraint

TR(k, k+1): transition relations

G(T): goal state at timeframe T

# Sequential System Solving with Modern SAT Solver



```
InitState();
SolvedState();
TunnelIdentifying();
PlayerMovementConstraints();
BoxPushMovementConstraints();
// PlayerHeadOnConstraints(); // MA
PlayerSinglePlacementConstraints();
BoxSinglePlacementConstraints();
// PlayerCollisionConstraints(); // MA
BoxCollisionConstraints();
BoxAndPlayerCollisionConstraints();
ExistenceConstraints();
DebugConstraints();
```

## Encoding methodology

**P(row, col, $p_i$, t):** encodes whether player $p_i$ is on position (row, col) at time step t

**B(row, col, $b_i$, t):** encodes whether box $b_i$ is on position (row, col) at time step t

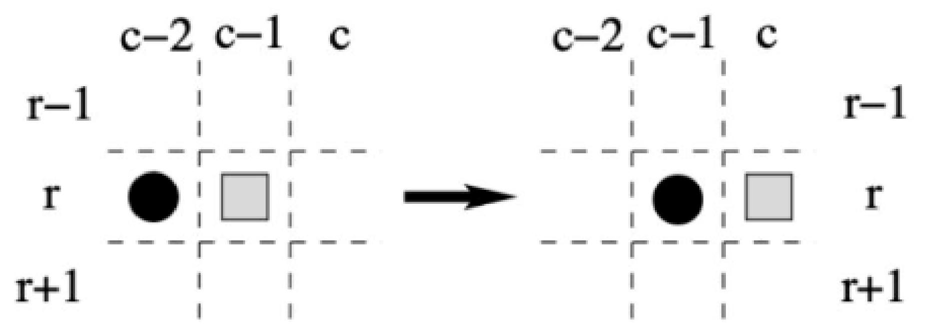
**Number of variables**: # of **walkable grids** x (# of **players** + # of **boxes**) x (**timesteps**+1)

## Constraints (10 necessary constraints in total)

box push constraint, player movement constraint, collision (overlap) constraint, head-on constraint.......

- Encoded as Conjunctive Normal Form (CNF), mostly from **implications**

## Example:
## encoding box push constraint



$$
\begin{aligned}
&B_{r,c,i,t+1}\\
&\Rightarrow [B_{r,c,i,t} \vee\\
&(B_{r,c\text{-}1,i,t} \wedge P_{r,c\text{-}2,pi1,t} \wedge P_{r,c\text{-}1,pi1,t+1}) \vee\\
&(B_{r,c+1,i,t} \wedge P_{r,c+2,pi1,t} \wedge P_{r,c+1,pi1,t+1}) \vee\\
&(B_{r+1,c,i,t} \wedge P_{r+2,c,pi1,t} \wedge P_{r+1,c,pi1,t+1}) \vee\\
&(B_{r\text{-}1,c,i,t} \wedge P_{r\text{-}2,c,pi1,t} \wedge P_{r\text{-}1,c,pi1,t+1}) \vee\\
&(B_{r,c\text{-}1,i,t} \wedge P_{r,c\text{-}2,pi2,t} \wedge P_{r,c\text{-}1,pi2,t+1}) \vee\\
&\ldots]
\end{aligned}
$$

converted to **solver acceptable cnf form** by cartesian product

$$\longrightarrow$$

$$a + bdc + edf + \ldots$$

(i) $\neg(\neg(a + bdc + edf))$

(ii) $\neg(\neg a\,(\neg b+\neg d+\neg c)\ (\neg e+\neg d+\neg f))$

(iii) $\neg(\neg a\neg b\neg e+\neg a\neg b\neg d+\neg a\neg b\neg f+\neg a\neg d\neg e+\ldots)$

(iv) $(a+b+c)(a+b+d)(a+b+f)(a+d+e)(a+d+d)(a+d+f)(a+c+e)(a+c+d)(a+c+f)$

# Sequential System Solving with Modern SAT Solver

## Data Structure

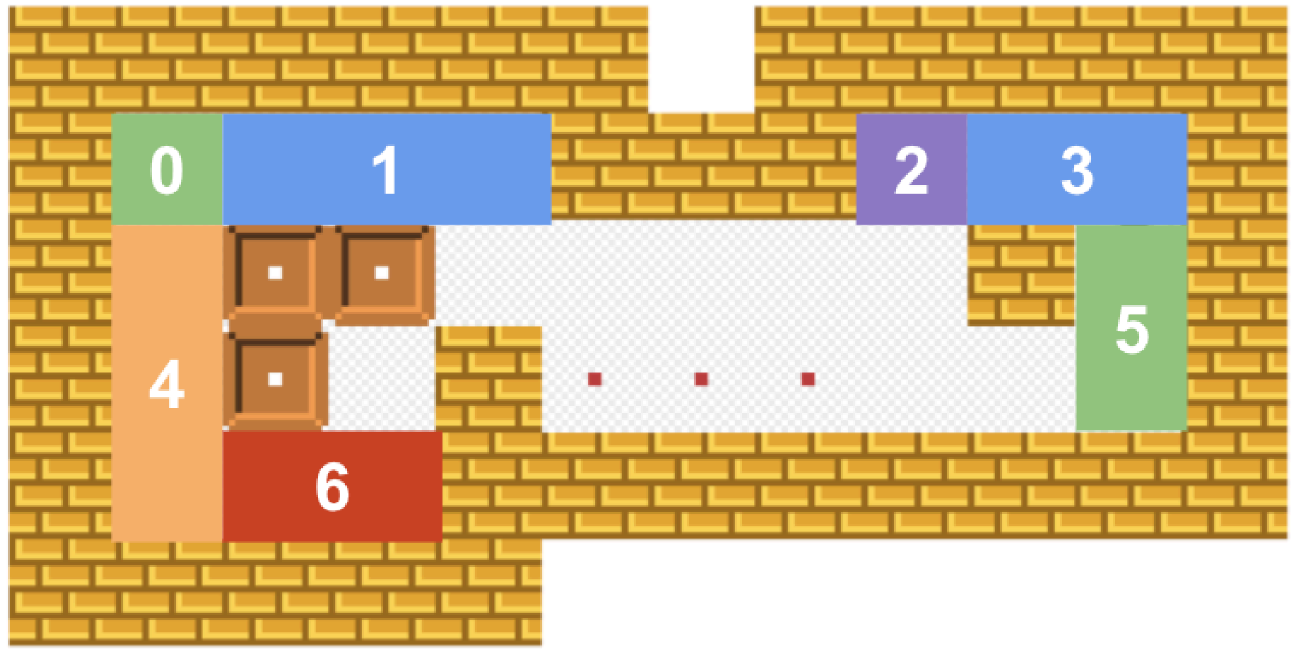
**class Lit**

- stores: x, y, literal index, time, identity
- operator overloading "~" for literal negations:
  - **negates** existing literal's index attribute
  - convenient for using **abc built-in** Var2Lit
  - absolute value as variable index, its **sign** determines pos./neg. literal

**class SokobanSolver**

- AddPlayerLiteral() & AddBoxLiteral():
  - instantiate literals
- constraints():
  - generate constraints
- playerLitManager: map literal key to existing player literal
- boxLitManager: map literal key to existing box literal objects

**Preprocessing:** find **dead end groups** to **remove more box state variables**



```
for walkable cell (r,c) {
    CandidateGroup = bfs(r,c)
    if (CandidateGroup excludes target)
        markDeadGroup(CandidateGroup)
}
```

bfs simulates possible push w/o considering feasibility

→ **simple deadends**

**Other heuristics Tunnel macro**: once player enters an entry-free tunnel, move through it **by implication**

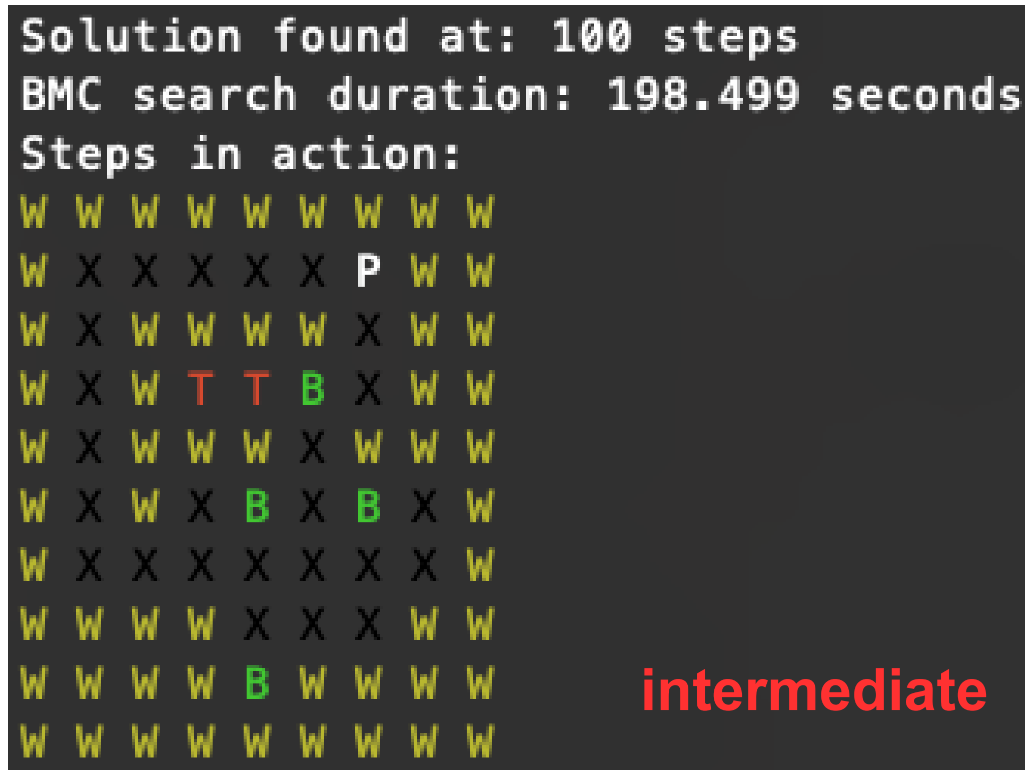
## Sequential System Solving with Modern SAT Solver

### Pseudo code

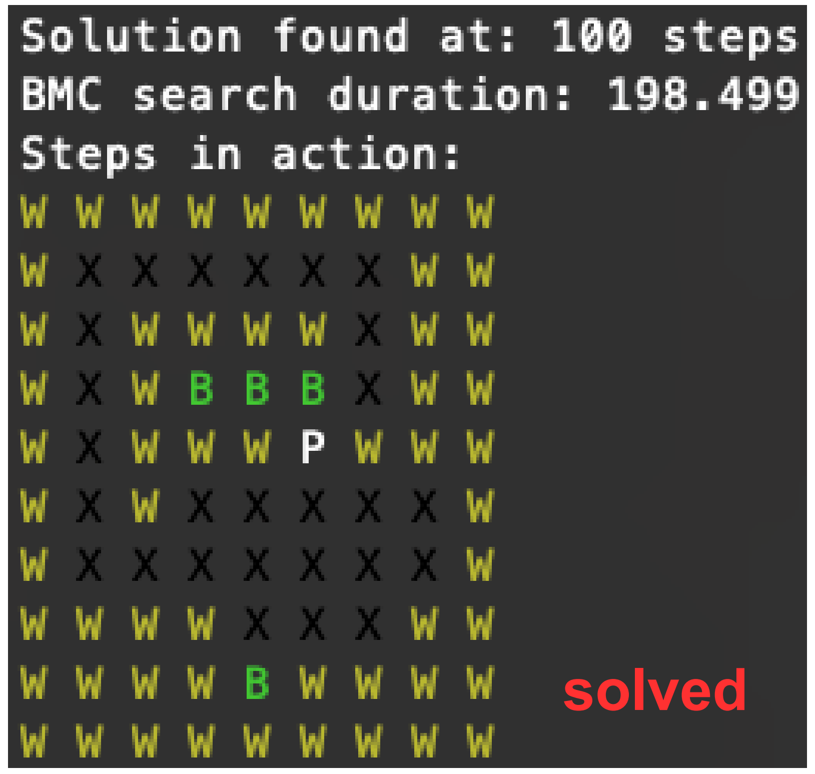
```
step = 1
while true:
        instantiate Solver instance
        Berkeley ABC sat solver pSat
        Solver set step limit to step
        Solver load map map
        Solver preprocess
        Solver create the cnf constraints
        Solver write to pSat
        step++
        if (pSat solved SAT)
                print "BMC solution found!"
                access true literals
                return 0; //exit main function
        delete pSat
```

### Conclusion

- in comparison to IDA*, MCTS, this work **guarantees optimal solution** (fewest moves)
- **binary search** for speed up (improve lower bounds)
- also allows **multiple agents** on map & solve it **optimally**
- testbench:
  - famous **microban** set (first 100 cases)
  - **solved 88/100** ( 1hour time limit )
- microban_61: 100 steps



```
Solution found at: 100 steps
BMC search duration: 198.499 seconds
Steps in action:
W W W W W W W W W
W X X X X X P W W
W X W W W W X W W
W X W T T B X W W
W X W W W X W W W
W X W X B X B X W
W X X X X X X X W
W W W W X X X W W
W W W W B W W W W
W W W W W W W W W
```



```
Solution found at: 100 steps
BMC search duration: 198.499
Steps in action:
W W W W W W W W W
W X X X X X X W W
W X W W W W X W W
W X W B B B X W W
W X W W W P W W W
W X W X X X X X W
W X X X X X X X W
W W W W X X X W W
W W W W B W W W W
W W W W W W W W W
```

# Sequential System Solving with Modern SAT Solver

## Experiment result

| testcase | time (sec) | bmc_S | IDA*_S | reduction |
|---|---|---|---|---|
| microban_1 | 0.652 | 33 | 33 | 0% |
| microban_2 | 0.075 | 16 | 16 | 0% |
| microban_3 | 3.41 | 41 | 41 | 0% |
| microban_4 | 0.296 | 23 | 29 | -21% |
| microban_5 | 3.191 | 25 | 27 | -7% |
| microban_6 | 460.968 | 107 | 115 | -7% |
| microban_7 | 21.419 | 26 | 38 | -32% |
| microban_8 | 330.479 | 97 | 99 | -2% |
| microban_9 | 0.287 | 30 | 30 | 0% |
| microban_10 | 844.679 | 87 | 121 | -28% |
| microban_11 | 19.823 | 78 | 78 | 0% |
| microban_12 | 1.159 | 49 | 49 | 0% |
| microban_13 | 25.426 | 52 | 59 | -12% |
| microban_14 | 1.376 | 51 | 51 | 0% |
| microban_15 | 0.922 | 37 | 43 | -14% |
| **microban_16** | **2085.174** | **100** | **-1** | **-1** |
| microban_17 | 0.443 | 25 | 31 | -19% |
| microban_18 | 13.875 | 71 | 95 | -25% |
| microban_19 | 5.235 | 41 | 45 | -9% |
| microban_20 | 2.007 | 50 | 64 | -22% |
| microban_21 | 0.06 | 17 | 19 | -11% |
| microban_22 | 1.403 | 47 | 49 | -4% |
| microban_23 | 0.005 | 56 | 58 | -3% |
| microban_24 | 1.472 | 35 | 35 | 0% |
| microban_25 | 3.305 | 29 | 33 | -12% |
| microban_26 | 10.404 | 41 | 42 | -2% |
| microban_27 | 12.543 | 50 | 50 | 0% |
| microban_28 | 0.934 | 33 | 33 | 0% |
| microban_29 | 125.383 | 104 | 140 | -26% |
| microban_30 | 0.35 | 21 | 21 | 0% |
| microban_31 | 0.425 | 17 | 17 | 0% |
| microban_32 | 19.849 | 35 | 36 | -3% |
| microban_33 | 28.609 | 41 | 53 | -23% |
| microban_34 | 9.159 | 30 | 45 | -33% |
| microban_37 | 28.421 | 71 | 71 | 0% |
| microban_38 | 1.802 | 37 | 37 | 0% |
| microban_39 | 179.582 | 85 | 93 | -9% |
| microban_40 | 0.921 | 20 | 25 | -20% |
| microban_41 | 13.344 | 50 | 61 | -18% |
| microban_42 | 85.641 | 47 | 61 | -23% |
| microban_43 | 31.697 | 61 | 66 | -8% |
| microban_44 | 0.002 | 1 | 1 | 0% |
| microban_45 | 6.256 | 45 | 47 | -4% |
| microban_46 | 1.732 | 41 | 47 | -13% |
| microban_47 | 79.992 | 83 | 101 | -18% |
| microban_48 | 22.454 | 64 | 67 | -4% |
| **microban_49** | **97.071** | **82** | **-1** | **-1** |
| microban_50 | 82.297 | 76 | 88 | -14% |