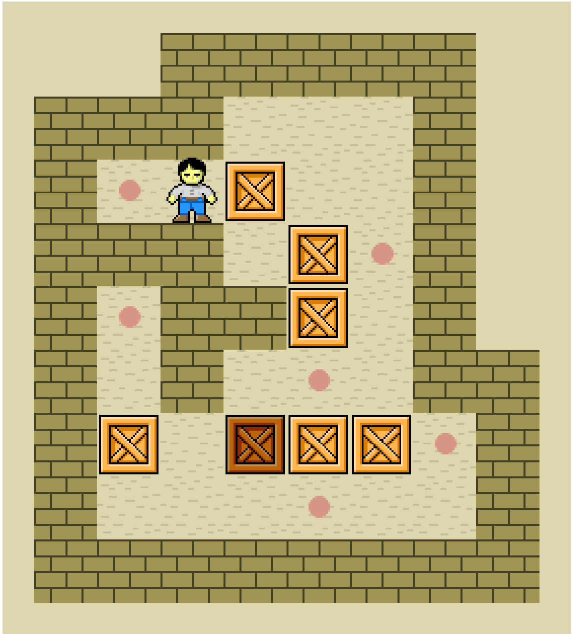
Sequential System Solving with Modern SAT Solver

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.

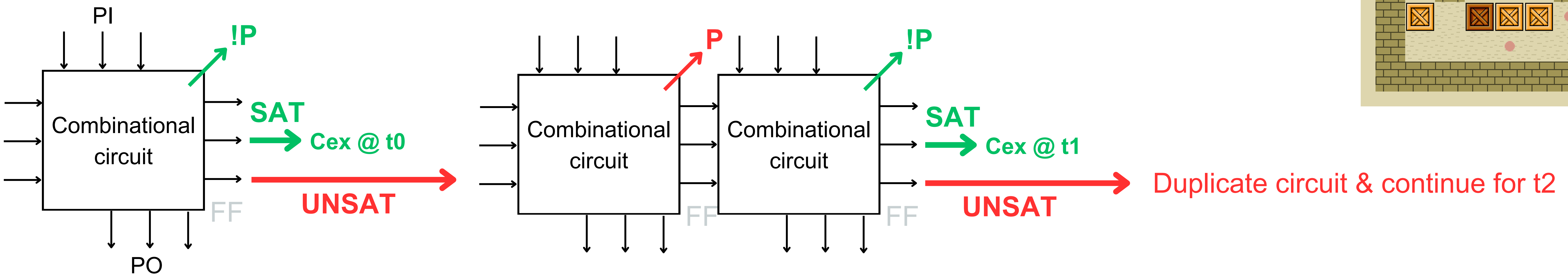
Movement is limited to **horizontal** and **vertical**, one cell at a time.



Encode as a Bounded Model Checking problem

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

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



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

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

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

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)

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

Constraints (10 constraints in total)

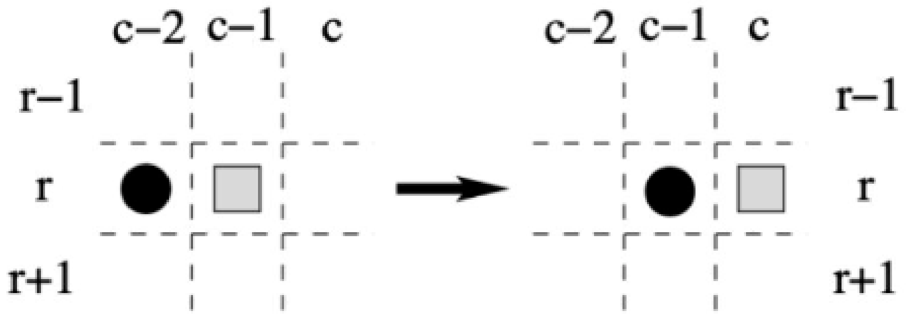
box push constraint, player movement constraint, collision (overlap) constraint, head-on constraint......

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

Sequential System Solving with Modern SAT Solver

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

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

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

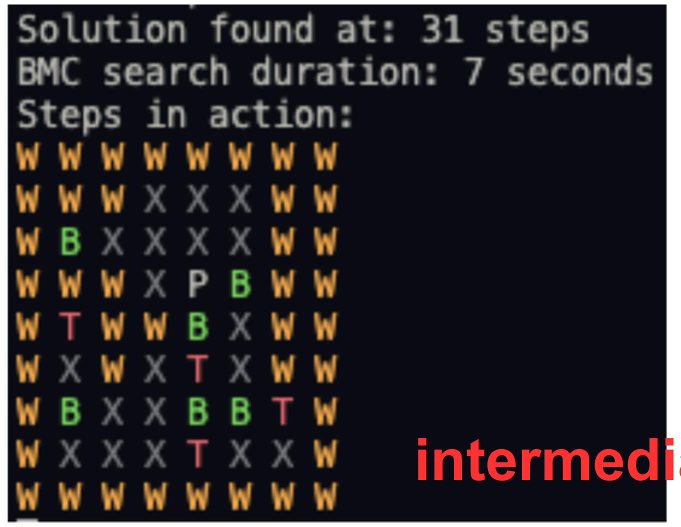
(ii) $\neg(\neg a\,(\neg b+\neg d+\neg c)\;(\neg e+\neg d+\neg f))$

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

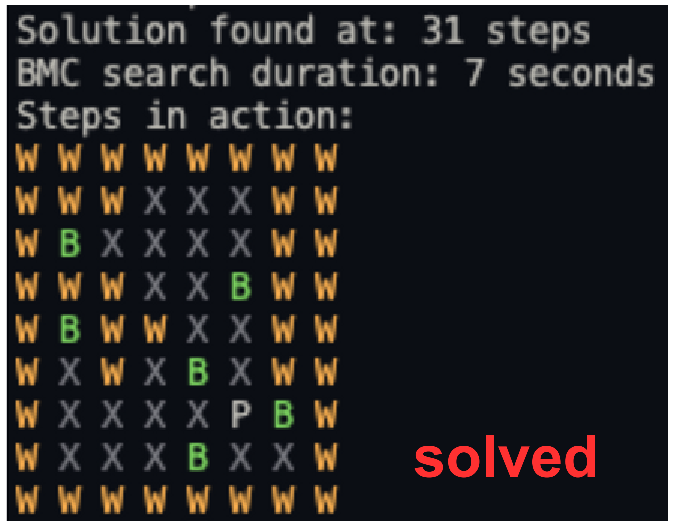
(iv) $(a+b+c)(a+b+d)(a+b+f)(a+d+e)(a+d+d)(a+d+f)(a+c+e)(a+c+d)(a+c+f)$

Conclusion

- in comparison to IDA*, MCTS solvers, this work **guarantees optimal solution** (fewest moves)
- **binary search** for further speed up (improve lower bounds)
- this work also allows **multiple agents** on map and solve it **optimally**
- testbench:
  - famous **microban** set (92 cases) + self designed pattern cases (corner based, tunnel based, open space,...)
  - **solved 79/110**



intermediate



solved