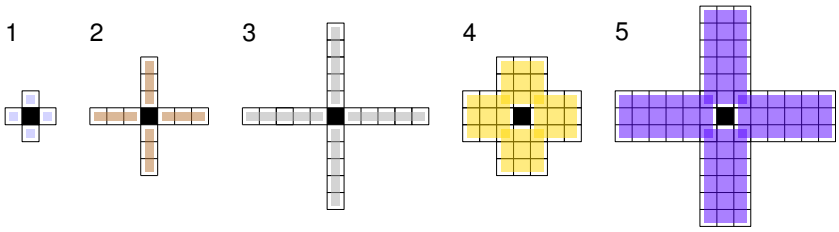
Practical SAT Solving (ST 2025) | Assignment 5

Markus Iser, Dominik Schreiber, Niccolò Rigi-Luperti

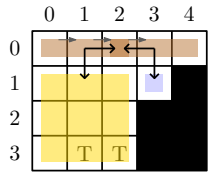
Algorithm Engineering & Scalable Automated Reasoning (KIT) 2024-07-01 – 2024-07-15

1 Competition: SDVSTPP (10(+10) Points)

As an extension of the slightly unrealistic SDVSTP (Assignment 2), we introduce the much more precise *Stardew Valley Soil Tilling Planning Problem* (SDVSTPP). The difference between SDVSTP and SDVSTPP is that the latter now features discrete *time steps* as well as a *player*. At any given point in time, the player is positioned at exactly one of the grid cells. At each time step, the player performs one out of two possible kinds of actions: (a) *walk* to one of the up to four adjacent tiles or (b) perform some tilling action at one of the up to four adjacent tiles. The tilling action's orientation is decided by the player position – the below figure shows all potentially valid operations for each of the five patterns, where the black square is the player's location.



Your task is to write an *optimal SAT-based SDVSTPP planner*. The planner application takes an `.sdvstp` file (just like the SDVSTP solver in Assignment 2) and assumes that the player is initially located at the *top left position*, i.e., at the first cell character read from the `.sdvstp` file. The planner should output the shortest possible plan in terms of time steps (i.e., player actions) that results in a grid state where tiles are tilled exactly according to the input specification (as in SDVSTP). As in the below example, the plan should be output as a single line beginning with "`s `", where `U/D/L/R` corresponds to up/down/left/right movement and where `u/d/l/r` followed by `1/2/3/4/5` corresponds to tilling action 1–5 in up/down/left/right direction, respectively.




```
s R d4 r2 R R l2 d1
```

You receive 10 points for an optimal planner solving easy inputs and ≤10 bonus points based on performance.

Code skeleton: `https://github.com/satlecture/kit2025/blob/main/code/src/sdvstp/sdvstpp.cc`

Some benchmark instances: `https://github.com/satlecture/kit2025/tree/main/exercises/sdvstp`

2 Correctness of RAT (4 Points)

Let F be a CNF formula and c be a RAT clause for F w.r.t. some literal $x \in c$.
Show that $F \cup \{c\}$ is satisfiability-equivalent to F, i.e., $F \cup \{c\}$ is satisfiable if and only if F is satisfiable.

3 LRUP Algorithms (3 + 3 + 3 Points)

(a) Extend the conflict analysis of CDCL to efficiently produce LRUP dependency information ("hints").

(b) Given a CNF formula F and an LRUP proof of unsatisfiability P for F, devise an algorithm using P that finds an *unsatisfiable core* of F, i.e., a subset of clauses in F that is unsatisfiable. The algorithm can perform a single, linear pass over P (in whichever direction) and cannot keep the entire proof in memory.

(c) Consider a CNF formula F and an LRUP proof of unsatisfiability P produced by a parallel clause-sharing solver as in the work by Michaelson et al. (2023, 2025). Devise an algorithm that takes F and P and outputs a proof P' for F with (typically) fewer clause additions.
Hint: Parallel CDCL solver threads may produce similar sets of clauses (especially at the beginning).