Recreational Formal Methods: Designing Vacuum Cleaning Trajectories

Frits Vaandrager
Institute for Computing and Information Sciences
Radboud University Nijmegen
F.Vaandrager@cs.ru.nl

Freek Verbeek
Department of Computer Science
Open University of The Netherlands
Freek.Verbeek@ou.nl

Abstract

We study an example due to Wooldridge of a small robotic agent that will vacuum clean a room. The room is an $n \times n$ grid and at any point the robot can move forward one step or turn right 90 degrees. The problem is to find a deterministic strategy for the robot in which (1) its next action only depends on its current square and orientation (one of north, west, south, east), and (2) all squares are visited infinitely often. We use a model checker and a SAT solver to find such strategies, and a proof assistant to exhibit certain symmetries in the problem.

1 Introduction

In his textbook on multiagent systems, Wooldridge [6] describes an example of a small robotic agent that will clean up a room. Figure 1 illustrates the vacuum world in which this robot operates. It is assumed that the room is a 3×3 grid, and that the robot always starts in square $(0, 0)$ facing north. The agent can suck up dirt, move forward to the next square, or turn right 90°. The goal is to traverse the room continuously searching for dirt and removing dirt. Wooldridge asks for the construction of a deterministic, memoryless strategy which, given the current square and orientation (one of north, west, south, east), and given whether the robot observes dirt, specifies the next action of the agent (one of suck, forward,

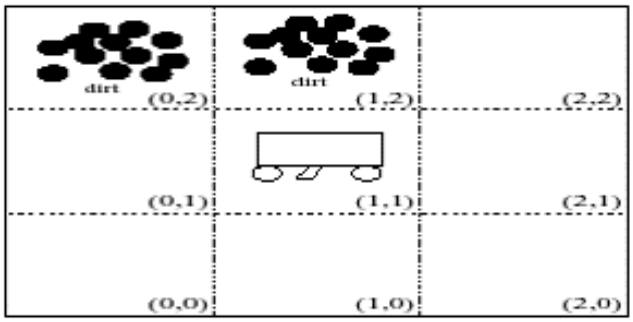

Figure 1: Vacuum world

turn). Assuming that all actions of the robot have their intended effect, this strategy should ensure that the robot will visit all squares infinitely often. Wooldridge gives a partial specification of such a strategy using a number of rules. The first rule states that if the agent is at location (x, y) and it perceives dirt, then the prescribed action is to suck up dirt.

$$In(x, y) \wedge Dirt(x, y) \quad \longrightarrow \quad Do(suck)$$

This rule takes priority over all other possible behaviors of the agent. Next four rules are listed which state that the robot will move from $(0, 0)$ to $(0, 1)$ to $(0, 2)$ and then to $(1, 2)$:

$$\begin{aligned} In(0,0) \wedge Facing(north) \wedge \neg Dirt(0,0) &\longrightarrow Do(forward) \\ In(0,1) \wedge Facing(north) \wedge \neg Dirt(0,1) &\longrightarrow Do(forward) \\ In(0,2) \wedge Facing(north) \wedge \neg Dirt(0,2) &\longrightarrow Do(turn) \\ In(0,2) \wedge Facing(east) &\longrightarrow Do(forward) \end{aligned}$$

According to Wooldridge, "similar rules can easily be generated that will get the agent to $(2, 2)$, and once at $(2, 2)$ back to $(0, 0)$." The first author, however, while diligently preparing a lecture on robotics for a freshman class, failed to find these rules. The problem is how to return to $(0, 0)$ after $(2, 2)$ has been reached. While on the way back, the robot may not revisit any square and orientation where it has been before: in such a case, since the robot is memoryless, it will continue forever on a loop that does not contain square $(0, 0)$. It appears that, after the robot has followed the initial rules specified by Wooldridge, it has painted itself in a corner and can never return to $(0, 0)$. It is not even obvious that there exists a deterministic, memoryless strategy for the robot that visits all squares infinitely often.

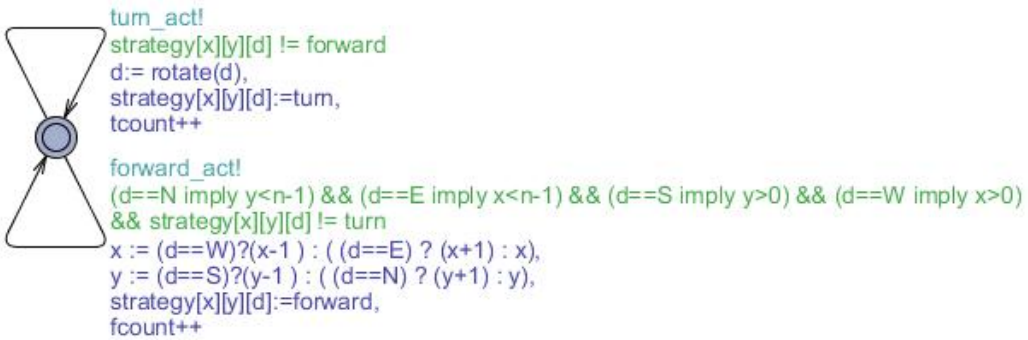

Figure 2: Uppaal model

This note describes how we tackled this problem using a model checker, a SAT solver, and even a proof assistant. The models and logical theories that we describe are available at the URL `http://www.mbsd.cs.ru.nl/publications/papers/fvaan/vacuumworld/`.

2 Model Checking

The problem of finding strategies for the vacuum cleaning robot can easily be encoded in a model checker. We constructed a model using the Uppaal tool [2]. Figure 2 displays the main template of our model. The model is parametrized by a constant `n`, which specifies the size of the grid. We use variables `x` and `y`, which range over type $\texttt{pos} = \{\texttt{0}, , \ldots, \texttt{n} - 1\}$, to store the current position of the robot, and a variable `d`, which ranges over type $\texttt{dir} = \{\texttt{N}, \texttt{W}, \texttt{S}, \texttt{E}\}$, to store the current orientation. Initially, `x` and `y` equal `0`, and `d` equals `N`. There are two transitions in the model, `turn_act!` and `forward_act!`. In the turn transition, the orientation `d` is updated using the function rotate, given by rotate(`N`) = `E`, rotate(`E`) = `S`, rotate(`S`) = `W` and rotate(`W`) = `N`. A forward transition is only enabled when there is a square in front of the robot, to prevent that the robot will hit the wall. In the model we abstract away from the dirt sucking as this is irrelevant for our problem.

An auxiliary array variable `strategy` records, for each position (`i`, `j`) and orientation `k`, the current strategy value, which is either `undefined`, `forward` or `turn`. Initially, `strategy[i][j][k]` is undefined for all `i`, `j` and `k`. Once `strategy[i][j][k]` is set to either `turn` or `forward`, it can never be changed again. We also use auxiliary variables `tcount` and `fcount` to count the total number of turns and forward moves, respectively.

Using the Uppaal verifier, we established that if the robot follows the rules specified by Wooldridge, it indeed paints itself in a corner. In fact, since the following Uppaal query does not hold for our model, there does not even exist a

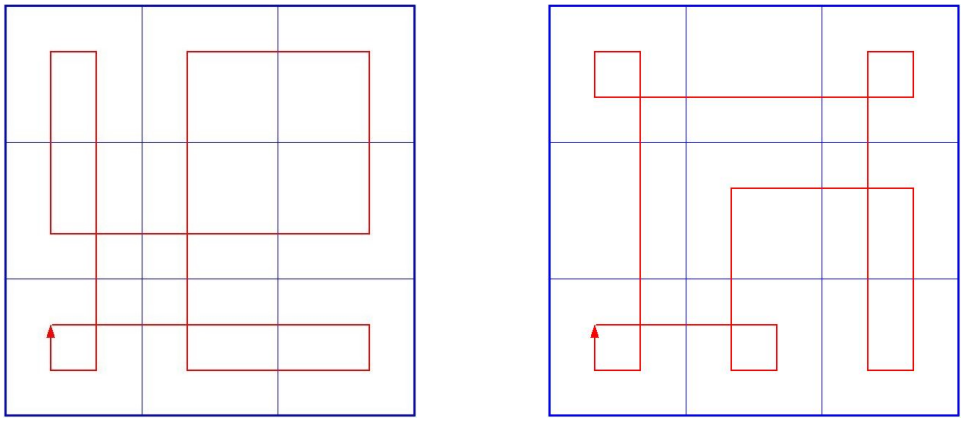

Figure 3: Two strategies with 12 (left) and 16 turns (right)

strategy that follows the rules for $(0, 0)$ and $(0, 1)$:

```
E<> (x==0 && y==0 && d==N &&
     forall (i:pos) forall (j:pos) visited(i,j) &&
     strategy[0][0][N]==forward && strategy[0][1][N]==forward)
```

Here `E <>` is Uppaal notation for the temporal operator $\exists\diamond$ and means "there exists a run leading to a state satisfying". Predicate `visited(i, j)` evaluates to true if the robot has visited square (`i`, `j`), that is, `strategy[i][j][k]` is defined for some orientation `k`. By omitting the last two conjuncts in the above query, we can instruct the Uppaal verifier to search for strategies that visit all squares infinitely often. Figure 3 shows two strategies found by Uppaal. The strategy on the right was (independently) also discovered by Bart van Thiel, one of the students from the robotics class. The two strategies of Figure 3 differ since the left one makes 12 turns whereas the right one makes 16 turns. Clearly, the number of turns in any strategy must be a multiple of 4. Using Uppaal we found that in fact all strategies contain either 12, 16 or 20 turns. Figure 4 shows two strategies, found by Uppaal, which both make 20 turns. These strategies differ since the left one contains 12 forward moves whereas the right one has 14 forward moves. It is easy to see that the number of forward moves in any strategy must be an even number. Using Uppaal we found that all strategies contain either 10, 12 or 14 forward moves.

In theory it is easy to enumerate all strategies using Uppaal: one repeatedly asks Uppaal whether there exists a strategy that is different from all strategies found thus far. In practice, however, this is quite involved, requiring either manual entry of all strategies as part of queries, or a nontrivial script which transforms Uppaal traces into queries. In order to obtain a complete overview of all possible strategies we therefore found it convenient to use a different tool.

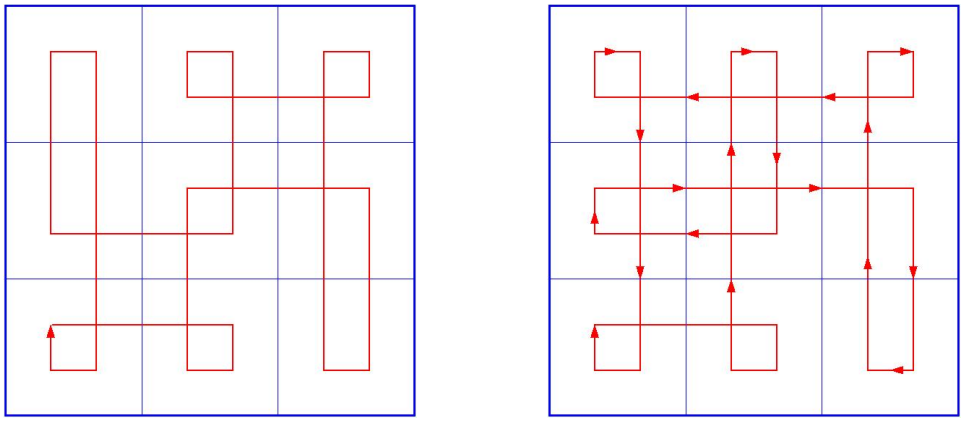

Figure 4: Two strategies with 20 turns and 12 (left) resp. 14 (right) forward moves

3 Constraint Solving

To enumerate vacuum cleaning strategies, we reformulate the problem: we need to find a path of length p that traverses a set of states Q and has to satisfy certain constraints. We then solve this problem for all p such that $l \leq p \leq h$ with l and h some conservatively estimated lower and higher bounds. The constraints can be formulated as a propositional satisfiability (SAT) problem. We therefore use zChaff [4], an automated solver for SAT problems.

For our SAT formulation of finding a vacuum cleaning strategy, we formalize the vacuum world as a labeled transition system.

Definition 3.1. *A* labeled transition system (LTS) *is a triple* $\mathcal{L} = (Q, A, \rightarrow)$*, where* Q *is a set of* states*,* A *is a set of* actions*, and* $\rightarrow \subseteq Q \times A \times Q$ *is a set of* transitions*. We write* $q \xrightarrow{a} q'$ *if* $(q, a, q') \in \rightarrow$*, and* $q \rightarrow q'$ *if there exists an* $a \in A$ *s.t.* $q \xrightarrow{a} q'$*.*

Fix a grid size n. Then our vacuum cleaning world is described by the LTS $\mathcal{V} = (\{0, 1, \ldots, n-1\} \times \{0, 1, \ldots, n-1\} \times \{\mathtt{N}, \mathtt{W}, \mathtt{S}, \mathtt{E}\}, \{\mathit{forward}, \mathit{turn}\}, \rightarrow)$, where relation $\rightarrow$ contains the following transitions, for $x, y \in [0, n-1]$ and $d \in \{\mathtt{N}, \mathtt{W}, \mathtt{S}, \mathtt{E}\}$,

$$
\begin{aligned}
(x, y, d) &\xrightarrow{\mathit{turn}} (x, y, \mathsf{rotate}(d)) \\
(x, y, \mathtt{N}) &\xrightarrow{\mathit{forward}} (x, y+1, \mathtt{N}) \quad \text{if} \quad y < n-1 \\
(x, y, \mathtt{E}) &\xrightarrow{\mathit{forward}} (x+1, y, \mathtt{E}) \quad \text{if} \quad x < n-1 \\
(x, y, \mathtt{S}) &\xrightarrow{\mathit{forward}} (x, y-1, \mathtt{S}) \quad \text{if} \quad y > 0 \\
(x, y, \mathtt{W}) &\xrightarrow{\mathit{forward}} (x-1, y, \mathtt{W}) \quad \text{if} \quad x > 0
\end{aligned}
$$

The SAT formulation of our problem introduces a Boolean variable for each pair (q, t), with q a state and t a natural number such that $0 \leq t < p$. We denote

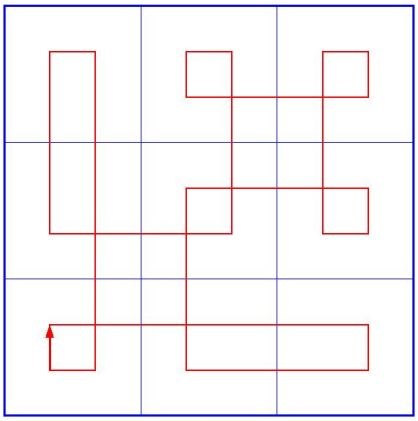

Figure 5: Another strategy with 20 turns and 12 forward moves found by zChaff

the Boolean variable corresponding to (q, t) with $\langle q, t\rangle$. The intended semantics is that Boolean variable $\langle q, t\rangle$ is true iff the vacuum cleaner is in state q at time slot t. All constraints are expressed in terms of these Boolean variables:
The strategy begins in $(0, 0, \mathtt{N})$:

$$\langle (0, 0, \mathtt{N}), 0\rangle$$

The strategy ends in $(0, 0, \mathtt{N})$:

$$\langle (0, 0, \mathtt{N}), p - 1\rangle$$

The strategy is connected:

$$\forall q, t < p - 1 \cdot \langle q, t\rangle \implies \bigvee_{\exists a \cdot q \xrightarrow{a} q'} \langle q', t + 1\rangle$$

The strategy is covering:

$$\forall x, y \cdot \exists d, t \cdot \langle (x, y, d), t\rangle$$

The strategy contains no duplicates other than the starting position:

$$\forall q, t_0, t_1 > t_0 \cdot \langle q, t_0\rangle \wedge q \neq (0, 0, \mathtt{N}) \implies \neg\langle q, t_1\rangle$$

Within one second, zChaff finds 28 solutions, ranging from the smallest (12 turns, 12 forward moves) to the most complex (20 turns, 14 forward moves). However, many solution are symmetric. There are four variants of the left strategy in Figure 3, which can be obtained by rotating the entire strategy 90, 180 and 270 degrees. Rotation and reflection variants of the right strategy of Figure 3 occur

eight times, variants of the left strategy of Figure 4 four times, and variants the right strategy of Figure 4 eight times. The only really new strategy found with zChaff, which has four incarnations, is displayed in Figure 5.

In order to obtain insight in these symmetries, we found it convenient to use a different tool.

4 Symmetries

In this section, we take a closer look at the symmetries that are present in the vacuum cleaning world. The proofs of all the theorems and lemmas in this section have been checked using the proof assistant Isabelle [5]. First we give a slightly more abstract characterization of the strategies Wooldridge asks for.

Definition 4.1. *Let $\mathcal{L} = (Q, A, \rightarrow)$ be an LTS. A* cycle *of $\mathcal{L}$ is a sequence $\sigma = q_1, \ldots, q_k$ of states such that, for all $i < k$, $q_i \rightarrow q_{i+1}$ and $q_k \rightarrow q_1$. A cycle is* minimal *if all states occurring in it are pairwise different.*

Definition 4.2. *Let Q be a a set of states, let σ be a sequence of states from Q, and let $\equiv$ be an equivalence relation on Q. We say that σ* covers *$\equiv$ if each equivalence class C of $\equiv$ contains a state that occurs in σ.*

Let $\approx$ be the equivalence relation that deems two states of the vacuum world LTS $\mathcal{V}$ equivalent if they belong to the same square on the grid:

$$(x, y, d) \approx (x', y', d') \quad \Leftrightarrow \quad x = x' \wedge y = y'.$$

Then the strategies Wooldridge asks for correspond to minimal cycles of the LTS $\mathcal{V}$ that cover $\approx$. Observe that the requirement that a strategy starts with $(0, 0, \mathtt{N})$ is not essential since any minimal cycle of $\mathcal{V}$ that covers $\approx$ contains $(0, 0, \mathtt{N})$, and any state on a cycle can be turned into the initial state by shifting states.

An automorphism is an isomorphism from an object to itself. It preserves the structure and captures a symmetry present in an object.

Definition 4.3. *An* automorphism *for an LTS $\mathcal{L} = (Q, A, \rightarrow)$ is a bijection $f : Q \rightarrow Q$ such that, for all $q, q' \in Q$ and for all $a \in A$, $q \xrightarrow{a} q'$ iff $f(q) \xrightarrow{a} f(q')$.*

The next theorem states that the function R that takes the whole vacuum world LTS and rotates it 90° to the right is an automorphism.

Theorem 4.4. *Let R be the function on states of $\mathcal{V}$ given by*

$$R(x, y, d) \quad = \quad (y, n - 1 - x, \mathsf{rotate}(d)).$$

Then R is an automorphism for vacuum world $\mathcal{V}$.

Theorem 4.5. *Let f be an automorphism for an LTS $\mathcal{L}$ and let σ be a cycle of $\mathcal{L}$. Then $f(\sigma)$ is a cycle of $\mathcal{L}$. Moreover, if σ is minimal then $f(\sigma)$ is also minimal.*

Lemma 4.6. *Let f be a bijection on a set of states Q, let $\equiv$ be an equivalence relation on Q such that $\forall q, q' \in Q : q \equiv q'$ implies $f(q) \equiv f(q')$ ($\equiv$ is a congruence for f), and let σ be a sequence of states in Q that covers $\equiv$. Then $f(\sigma)$ covers $\equiv$.*

Suppose σ is a minimal cycle of $\mathcal{V}$ that covers $\approx$. By Theorems 4.4 and 4.5, $R(\sigma)$ is a minimal cycle of $\mathcal{V}$. Since bijection R trivially is a congruence for $\approx$, it follows by Lemma 4.6 that $R(\sigma)$ covers $\approx$. Thus automorphism R maps strategies to strategies.

With the rotation automorphism R we capture most but not all the symmetries in our vacuum world. Besides rotation of a strategy, we also must consider the reflection of a strategy in the axis $x = 1/2n$. The mirror image of a strategy in which the robot only takes right turns, is a strategy in which the robot only takes left turns. However, in order to obtain a strategy with right turns again we can reverse the direction in which the edges are traversed. Mathematically, we need a notion of "autocontramorphism" to capture these symmetries.

Definition 4.7. *An* autocontramorphism *for an LTS $\mathcal{L} = (Q, A, \rightarrow)$ is a bijection $f : Q \rightarrow Q$ such that, for all $q, q' \in Q$ and for all $a \in A$, $q \xrightarrow{a} q'$ iff $f(q') \xrightarrow{a} f(q)$.*

Theorem 4.8. *Let F be the function on states of $\mathcal{V}$ given by*

$$F(x, y, d) \quad = \quad (n - 1 - x, y, \mathsf{flip}(d)),$$

where function flip *is defined by* $\mathsf{flip}(\mathtt{N}) = \mathtt{S}$, $\mathsf{flip}(\mathtt{E}) = \mathtt{E}$, $\mathsf{flip}(\mathtt{S}) = \mathtt{N}$, *and* $\mathsf{flip}(\mathtt{W}) = \mathtt{W}$, *Then F is an autocontramorphism for vacuum world $\mathcal{V}$.*

Theorem 4.9. *Let f be an autocontramorphism for an LTS $\mathcal{L}$ and let σ be a cycle of $\mathcal{L}$. Then $f(\sigma)$ is a cycle of $\mathcal{L}$. Moreover, if σ is minimal then $f(\sigma)$ is also minimal.*

Suppose σ is a minimal cycle of $\mathcal{V}$ that covers $\approx$. By Theorems 4.8 and 4.9, $F(\sigma)$ is a minimal cycle of $\mathcal{V}$. Since bijection F trivially is a congruence for $\approx$, it follows by Lemma 4.6 that $F(\sigma)$ covers $\approx$. Thus autocontramorphism F maps strategies to strategies.

Modulo the symmetries induced by rotation automorphism R and reflection autocontramorphism F, the 28 vacuum cleaning strategies found by zChaff reduce to the 5 strategies displayed in Figures 3, 4 and 5.

5 Snake Tilings

Now that we have a full understanding and classification of the vacuum cleaning strategies for a 3×3 grid, the natural question arises whether we can also solve this problem for arbitrary $m \times n$ grids, for $m, n \geq 1$. Model checking and SAT solving can only compute strategies in case m and n are small: Uppaal runs out of memory for a 5×5 grid, and the largest instance that we could solve using zChaff was a 7×7 grid.

We can at least prove the existence of vacuum cleaning strategies for arbitrary $m \times n$ grids using "tiles" with incoming and outgoing arrows. Figure 6 illustrates a tiling scheme that we may use to obtain a strategy for an arbitrary $m \times n$ grid, for m, n even and at least 4. The idea is that we can duplicate tile **B** $\frac{m-4}{2}$ times to obtain a bottom row of length m. The $\mathbf{B}^*\mathbf{C}$ pattern can then be copied to the two rows above. Next the row of tile **D** can be copied $\frac{n-4}{2}$ times leading to a tiling of the $m \times n$ grid. Using similar tiling patterns one can prove the existence of vacuum cleaning strategies for arbitrary $m \times n$ grids.

The tilings of Figure 6 are closely related to the work of Kari [3] on infinite snake tiling problems, except that the trajectories ("snakes") of Kari may also turn left and do not have to return to their starting state. Similar tilings were also studied by Adleman et al. [1] in their work on self-assembly.

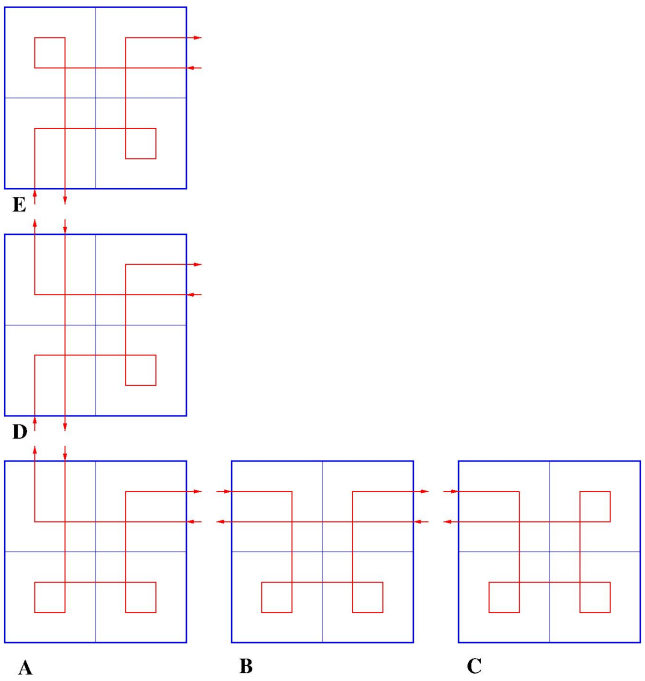

Figure 6: A tiling for $m \times n$ grids, with $m, n \geq 4$ even

6 Conclusion

The vacuum world example of Wooldridge [6] serves as a nice illustration of how the combined use of various tools and techniques from theoretical computer science may help to solve a problem.

We do not expect that this note will revolutionize the vacuum cleaning industry. After all, why would one impose the restriction that a vacuum cleaning robot may only turn right when electric motors just as easily run forward as backward? Why would one restrict to memoryless strategies when memory is so cheap and just adding a single bit to the domain of strategies makes it trivial to design schedules that visit each square infinitely often? Our strategies are also based on the unrealistic assumption that the floor is empty and without obstacles like tables and chairs that must be avoided.

The trajectories of Figures 3, 4 and 5 have some aesthetic quality and may serve as a basis for design of tilings, e.g. a long snake that bites itself in the tail.

The moral of our story is that authors of wonderful textbooks should be careful with the use of phrases like "similar rules can easily be generated". The risk is that colleagues will publish a note in the Bulletin of the EATCS pointing out that in fact the generation of such rules is impossible or at least tricky.

References

[1] Leonard M. Adleman, Jarkko Kari, Lila Kari, and Dustin Reishus. On the decidability of self-assembly of infinite ribbons. In *FOCS*, pages 530–537. IEEE Computer Society, 2002.

[2] G. Behrmann, A. David, K. G. Larsen, J. Håkansson, P. Pettersson, W. Yi, and M. Hendriks. Uppaal 4.0. In *QEST 2006,* 11-14 September 2006, Riverside, CA, USA, pages 125–126. IEEE Computer Society, 2006.

[3] Jarkko Kari. Infinite snake tiling problems. In *DLT 2002, Kyoto, Japan, September 18-21, 2002, Revised Papers*, LNCS 2450, pages 67–77. Springer, 2002.

[4] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In DAC '01, pages 530–535, New York, NY, USA, 2001. ACM.

[5] Lawrence C. Paulson. Isabelle: The next 700 theorem provers. *CoRR*, cs.LO/9301106, 1993.

[6] M. Wooldridge. *An Introduction to MultiAgent Systems, 2nd edition*. John Wiley & Sons Ltd, 2009.