Exercise 2.1: reconstructing of a counter-example

In the class, we considered the following (almost correct) algorithm (Iteration 3 in the slides):

```
operation write(v): % Invoked only to change the value %
i  change REG;
ii if WR = RR then change WR; % Strive to establish WR ≠ RR %
operation read():
1 if WR = RR then return (val);
2 aux := REG; % Conservative value %
3' change RR; % Strive to establish WR = RR %
4 val := REG;
5 if WR = RR then return (val);
7 return (aux)
```

Construct a run of this algorithm in which new-old inversion is observed (following the hints in the slides).

Exercise 2.2: Proof of Tromp’s construction

Prove that the algorithm below indeed implements an atomic bit.

```
operation write(v): % Invoked only to change the value %
i  change REG;
ii if WR = RR then change WR; % Strive to establish WR ≠ RR %
operation read():
1 if WR = RR then return (val);
2 aux := REG; % Conservative value %
3 if WR ≠ RR then change RR;
4 val := REG;
5 if WR = RR then return (val);
6 val := REG;
7 return (aux)
```

Hint: prove that for the reading function \( \pi \) defined in the slides satisfies:

\( A0 : \forall r: \neg(r \to_H \pi(r)). \) (No read returns a value not yet written.)

\( A1 : \forall r, w \in H: (w \to_H r) \Rightarrow (\pi(r) = w) \lor (w \to_H \pi(r)). \) (No read obtains a overwritten value.)

\( A2 : \forall r1, r2: (r1 \to_H r2) \Rightarrow (\pi(r1) = \pi(r2)) \lor (\pi(r1) \to_H \pi(r2)). \) (No new/old inversion.)