hhu.

Modeling and Proving the Parallel Climbers Puzzle in Event-B

Kevin Schwarz

13th Rodin User and Developer Workshop

May 18th, 2026

Preface

- Project work as part of my master's degree
- At the chair of Software Engineering and Programming Languages at Heinrich-Heine-University Düsseldorf[1], headed by Prof. Dr. Michael Leuschel

[1] https://www.cs.hhu.de/lehrstuehle-und-arbeitsgruppen/softwaretechnik-und-programmiersprachen

Specifications of the Puzzle [2]

- A 2-dimensional mountain range with peaks and valleys
- Two climbers (left and right), who start at height 0 at opposite sites of the mountain range
- Goal is to meet at the summit
- But they have to remain at the same height at all time

- Is this possible for any given mountain range?

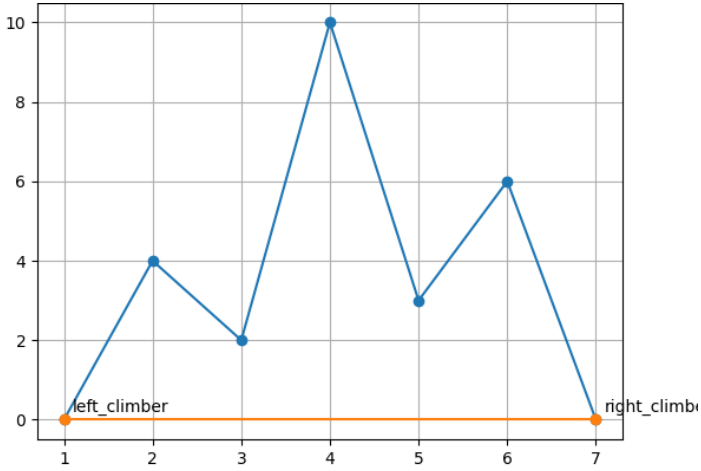
[2] Tucker, A. (1995). The Parallel Climbers Puzzle: A Case Study in the Power of Graph Models. Math Horizons, 3(2), 22–24. https://doi.org/10.1080/10724117.1995.11974954

Further constraints

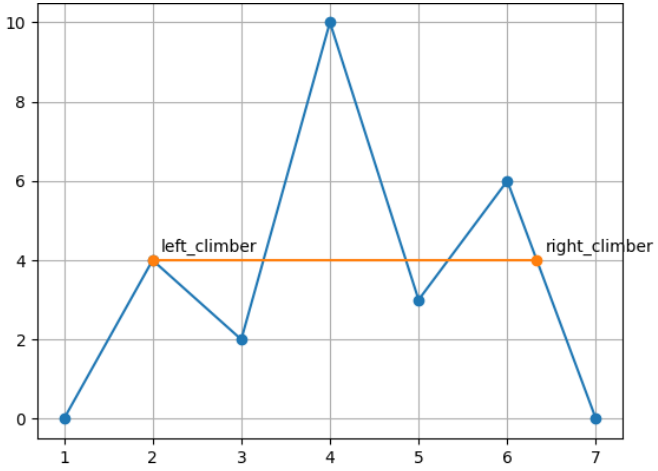
- Mountain range must be strictly greater than 0 except the two starting points of the left and right climber
- The meeting point (summit) must be strictly greater than all other points of the mountain range

- Without loss of generality the mountain range can be considered as a sequence of its extrema

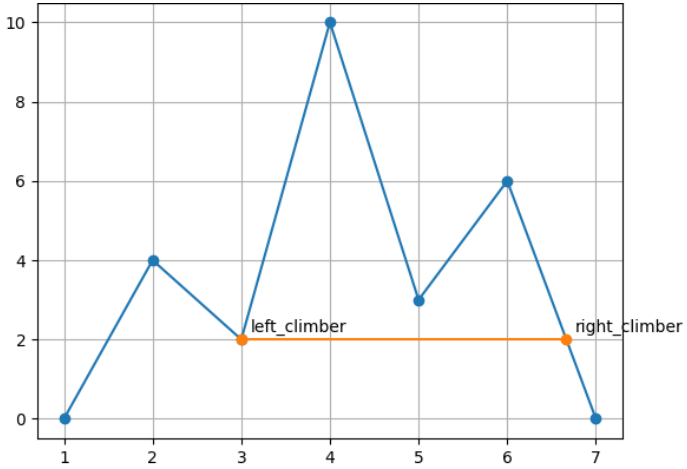
Example - Starting state



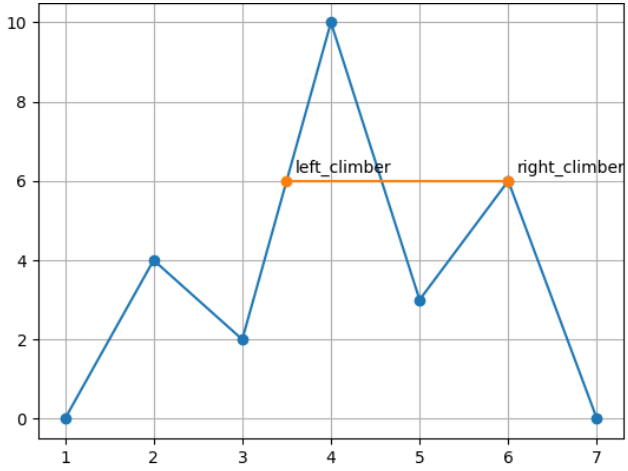
Example - Step 1



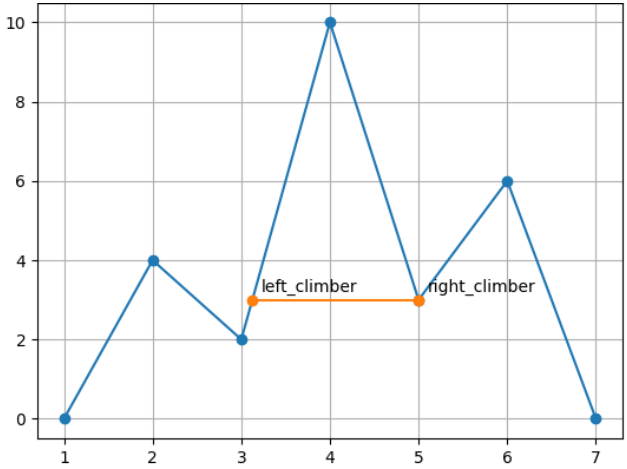
Example - Step 2



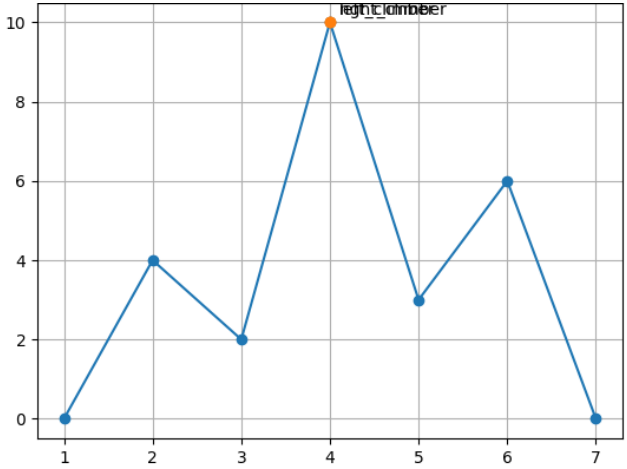
Example - Step 3



Example - Step 4



Example - Step 5



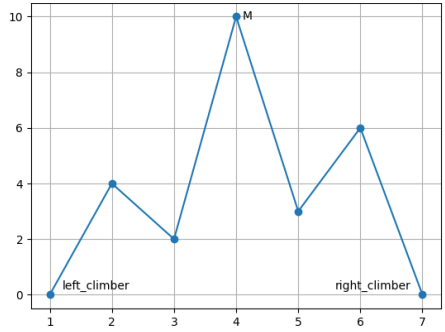
Modeling in Event-B

Heinrich Heine Universität Düsseldorf

```
n1: n ∈ ℕ1 not theorem ›
v1: V ∈ 1‥n ⟶ ℕ not theorem ›
v2: V(1) = 0 ∧ V(n) = 0 not theorem ›
v3: ∀i·i ∈ dom(V) ∧ i ≠ 1 ∧ i ≠ n ⇒ V(i) > 0 not theorem ›
v4: ∀i·i ∈ dom(V) ∧ i + 1 ∈ dom(V) ⇒ V(i) ≠ V(i + 1) not theorem ›
m1: m ∈ dom(V) not theorem ›
m2: ∀i·i ∈ dom(V) ∧ i ≠ m ⇒ V(m) > V(i) not theorem ›


inv1:   left_climber↦right_climber ∈ Valid_Positions not theorem ›


act1:   left_climber ≔ 1↦1 ›
act2:   right_climber ≔ n↦n ›
```



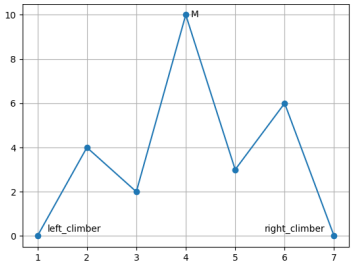
$$V = \{(1 \mapsto 0), (2 \mapsto 4), (3 \mapsto 2), (4 \mapsto 10), (5 \mapsto 3), (6 \mapsto 6), (7 \mapsto 0)\}$$

Modeling in Event-B

Heinrich Heine Universität Düsseldorf

```
pos1:  Positions ∈ {i↦j | i ∈ dom(V) ∧ j ∈ dom(V) ∧ (j = i ∨ j = i + 1)} ⟶ ℙ(N) not theorem ›
pos2:  ∀i·i ∈ dom(Positions)
         ⇒   (V(prj1(i)) < V(prj2(i)) ⇒ Positions(i) = {j | V(prj1(i)) < j ∧ j < V(prj2(i))})
           ∧ (V(prj1(i)) > V(prj2(i)) ⇒ Positions(i) = {j | V(prj2(i)) < j ∧ j < V(prj1(i))})
           ∧ (V(prj1(i)) = V(prj2(i)) ⇒ Positions(i) = {V(prj1(i))}) not theorem ›
```

$$
\begin{aligned}
\mathit{Positions} = \{&(1 \mapsto 1) \mapsto \{0\},\\
&(1 \mapsto 2) \mapsto \{1, 2, 3\},\\
&(2 \mapsto 2) \mapsto \{4\},\\
&(2 \mapsto 3) \mapsto \{3\},\\
&(3 \mapsto 3) \mapsto \{2\}, \dots\}
\end{aligned}
$$



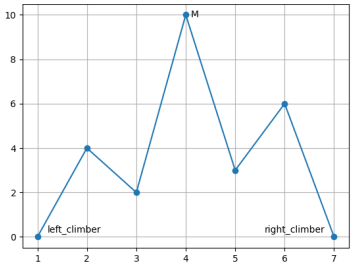
$$V = \{(1 \mapsto 0), (2 \mapsto 4), (3 \mapsto 2), (4 \mapsto 10), (5 \mapsto 3), (6 \mapsto 6), (7 \mapsto 0)\}$$

Modeling in Event-B

Heinrich Heine Universität Düsseldorf

```
valid1: Valid_Positions ∈ dom(Positions) ↔ dom(Positions) not theorem ›
valid2: Valid_Positions = {l↦r |
          l ∈ dom(Positions) ∧ r ∈ dom(Positions)
        ∧ prj2(l) ≤ m ∧ m ≤ prj1(r)
        ∧   (prj1(l) = prj2(l) ∧ prj1(r) = prj2(r)
              ⇒ V(prj1(l)) = V(prj1(r)))
            ∧ (prj1(l) = prj2(l) ∧ prj1(r) ≠ prj2(r)
              ⇒ V(prj1(l)) ∈ Positions(r))
            ∧ (prj1(l) ≠ prj2(l) ∧ prj1(r) = prj2(r)
              ⇒ V(prj1(r)) ∈ Positions(l))} not theorem ›
```

$$\begin{aligned} \textit{Valid\_Positions} = \{&(1 \mapsto 1) \mapsto (7 \mapsto 7),\\ &(1 \mapsto 2) \mapsto (6 \mapsto 7),\\ &(2 \mapsto 2) \mapsto (4 \mapsto 5),\\ &(2 \mapsto 2) \mapsto (6 \mapsto 7), \dots\} \end{aligned}$$



$$V = \{(1 \mapsto 0), (2 \mapsto 4), (3 \mapsto 2), (4 \mapsto 10), (5 \mapsto 3), (6 \mapsto 6), (7 \mapsto 0)\}$$

Modeling in Event-B - Moving the climbers

- move event: $left\_climber \mapsto right\_climber :\in Valid\_Positions$
- So far no restrictions
- Climbers can teleport or stay still

$\Rightarrow$ Restricting the movement, so that there is progress and they cannot teleport
$\Rightarrow$ Introduction of a variant

```
vrn1:    prj1(right_climber) − prj1(left_climber) + prj2(right_climber) − prj2(left_climber) ›
```

$\Rightarrow$ Teleportation and standstill excluded

Modeling in Event-B - Moving the climbers

Heinrich Heine Universität Düsseldorf

```
move:   not extended convergent ›
ANY
◦   next     ›
WHERE
◦   grd1:   next ∈ Valid_Positions not theorem ›
◦   grd2:   prj1(left_climber) = prj2(left_climber) ∧ prj1(right_climber) = prj2(right_climber)
              ⇒   next = prj1(left_climber) + 1 ↦ prj2(left_climber) + 1 ↦ (prj1(right_climber) − 1 ↦ prj2(right_climber)    )
                ∨ next = prj1(left_climber) + 1 ↦ prj2(left_climber) + 1 ↦ (prj1(right_climber) − 1 ↦ prj2(right_climber) − 1)
                ∨ next = prj1(left_climber)     ↦ prj2(left_climber) + 1 ↦ (prj1(right_climber) − 1 ↦ prj2(right_climber) − 1)
                ∨ next = prj1(left_climber) − 1 ↦ prj2(left_climber)     ↦ (prj1(right_climber) − 1 ↦ prj2(right_climber) − 1)
                ∨ next = prj1(left_climber) + 1 ↦ prj2(left_climber) + 1 ↦ (prj1(right_climber)     ↦ prj2(right_climber) + 1)
             not theorem ›
◦   grd3:   prj1(left_climber) = prj2(left_climber) ∧ prj1(right_climber) ≠ prj2(right_climber)
              ⇒   next = prj1(left_climber) + 1 ↦ prj2(left_climber) + 1 ↦ (prj1(right_climber)     ↦ prj2(right_climber)    )
                ∨ next = prj1(left_climber) + 1 ↦ prj2(left_climber) + 1 ↦ (prj1(right_climber)     ↦ prj2(right_climber) − 1)
                ∨ next = prj1(left_climber)     ↦ prj2(left_climber) + 1 ↦ (prj1(right_climber)     ↦ prj2(right_climber) − 1)
                ∨ next = prj1(left_climber) + 1 ↦ prj2(left_climber) + 1 ↦ (prj1(right_climber) + 1 ↦ prj2(right_climber)    )
             not theorem ›
◦   grd4:   prj1(left_climber) ≠ prj2(left_climber) ∧ prj1(right_climber) = prj2(right_climber)
              ⇒   next = prj1(left_climber)     ↦ prj2(left_climber)     ↦ (prj1(right_climber) − 1 ↦ prj2(right_climber) − 1)
                ∨ next = prj1(left_climber) + 1 ↦ prj2(left_climber)     ↦ (prj1(right_climber) − 1 ↦ prj2(right_climber) − 1)
                ∨ next = prj1(left_climber) + 1 ↦ prj2(left_climber)     ↦ (prj1(right_climber) − 1 ↦ prj2(right_climber)    )
                ∨ next = prj1(left_climber)     ↦ prj2(left_climber) − 1 ↦ (prj1(right_climber) − 1 ↦ prj2(right_climber) − 1)
             not theorem ›
THEN
◦   act1:   left_climber ≔ prj1(next) ›
◦   act2:   right_climber ≔ prj2(next) ›
END
```

Proving in Rodin

Heinrich Heine Universität Düsseldorf

- The case

$$prj1(left\_climber) \neq prj2(left\_climber) \wedge prj1(right\_climber) \neq prj2(right\_climber)$$

can be ignored, because it is never reached

```
inv2:   prj1(left climber) = prj2(left climber) ∨ prj1(right climber) = prj2(right climber) not theorem ›
```

Proving in Rodin

- Variant shows that every movement results in progress to the summit
- It has to be shown that as long as the variant is unequal to 0 the move event is executable

Proving in Rodin

Heinrich Heine Universität Düsseldorf

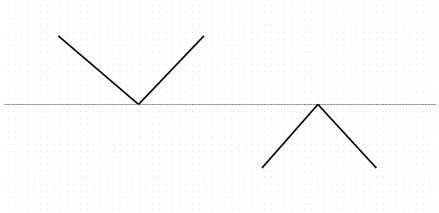
```
inv10:  left_climber ≠ right_climber
          ⇒   (prj1(left_climber) = prj2(left_climber) ∧ prj1(right_climber) = prj2(right_climber)
                 ⇒   prj1(left_climber) + 1 ↦ prj2(left_climber) + 1 ↦ (prj1(right_climber) − 1 ↦ prj2(right_climber)    ) ∈ Valid_Positions
                   ∨ prj1(left_climber) + 1 ↦ prj2(left_climber) + 1 ↦ (prj1(right_climber) − 1 ↦ prj2(right_climber) − 1) ∈ Valid_Positions
                   ∨ prj1(left_climber)     ↦ prj2(left_climber) + 1 ↦ (prj1(right_climber) − 1 ↦ prj2(right_climber) − 1) ∈ Valid_Positions
                   ∨ prj1(left_climber) − 1 ↦ prj2(left_climber)     ↦ (prj1(right_climber) − 1 ↦ prj2(right_climber) − 1) ∈ Valid_Positions
                   ∨ prj1(left_climber) + 1 ↦ prj2(left_climber) + 1 ↦ (prj1(right_climber)     ↦ prj2(right_climber) + 1) ∈ Valid_Positions
              ) not theorem ›
inv11:  left_climber ≠ right_climber
          ⇒   (prj1(left_climber) = prj2(left_climber) ∧ prj1(right_climber) ≠ prj2(right_climber)
                 ⇒   prj1(left_climber) + 1 ↦ prj2(left_climber) + 1 ↦ (prj1(right_climber)     ↦ prj2(right_climber)    ) ∈ Valid_Positions
                   ∨ prj1(left_climber) + 1 ↦ prj2(left_climber) + 1 ↦ (prj1(right_climber)     ↦ prj2(right_climber) − 1) ∈ Valid_Positions
                   ∨ prj1(left_climber)     ↦ prj2(left_climber) + 1 ↦ (prj1(right_climber)     ↦ prj2(right_climber) − 1) ∈ Valid_Positions
                   ∨ prj1(left_climber) + 1 ↦ prj2(left_climber) + 1 ↦ (prj1(right_climber) + 1 ↦ prj2(right_climber)    ) ∈ Valid_Positions
              ) not theorem ›
inv12:  left_climber ≠ right_climber
          ⇒   (prj1(left_climber) ≠ prj2(left_climber) ∧ prj1(right_climber) = prj2(right_climber)
                 ⇒   prj1(left_climber)     ↦ prj2(left_climber)     ↦ (prj1(right_climber) − 1 ↦ prj2(right_climber) − 1) ∈ Valid_Positions
                   ∨ prj1(left_climber) + 1 ↦ prj2(left_climber)     ↦ (prj1(right_climber) − 1 ↦ prj2(right_climber) − 1) ∈ Valid_Positions
                   ∨ prj1(left_climber) + 1 ↦ prj2(left_climber)     ↦ (prj1(right_climber)     ↦ prj2(right_climber)    ) ∈ Valid_Positions
                   ∨ prj1(left_climber)     ↦ prj2(left_climber) − 1 ↦ (prj1(right_climber) − 1 ↦ prj2(right_climber) − 1) ∈ Valid_Positions
              ) not theorem ›
```

Proving in Rodin

- The combination of variant and that one guard is always true builds the proof

Difficulties of the proof

- Many case distinctions (between 20-30 per invariant)
- Proofing via the interactive prover in Rodin takes up a lot of time (approx. 1h for each invariant)
- Planing such big proofs took even longer
- Still there are cases that have not been proven
- But those unproven cases are not reachable



Conclusion & Future Work

Heinrich Heine
Universität Düsseldorf

- Big proofs should be planned in advance
- ”Simple” problem can lead to complex proofs
- Relevancy of tools in the interactive prover (such as ProB, SMT Solver, etc.)
- Making auxiliary theorems in between is beneficial

Contact

Heinrich Heine
Universität Düsseldorf

If you have further questions you can reach me via kevin.schwarz@hhu.de