Example program proof — sequential rule

In this example, we prove the partial correctness of a program fragment that exchanges the values of two (integer) variables.

```
    {A=1 ∧ B=2}
     A:= A+B;
     B:= A–B;
     A:= A–B
  {A=2 ∧ B=1}
```

Since sequential execution occurs twice, we will need to apply the rule twice. Each of these steps requires the determination of a common pre/post-condition.

```
1.  |— {A=1 ∧ B=2}
          A:= A+B
       {A=3 ∧ B=2}
```

by the Axiom of Assignment and logical equivalence on the pre-condition.

```
2.  |— {A=3 ∧ B=2}
          B:= A–B
       {A=3 ∧ B=1}
```

by the Axiom of Assignment and logical equivalence on the pre-condition.

```
3.  |— {A=1 ∧ B=2}
          A:= A+B;
          B:=A–B
       {A=3 ∧ B=1}
```

by the Sequential rule using steps 1 and 2.

```
4. |— {A=3 ∧ B=1}
          A:= A–B
       {A=2 ∧ B=1}
```

by the Axiom of Assignment and logical equivalence on the pre-condition.

```
 5.  |— {A=1 ∧ B=2}
          A:= A+B;
          B:=A–B;
          A:= A–B
       {A=3 ∧ B=1}
```

by the Sequential rule using steps 3 and 4.

To develop this proof, you need to discern an appropriate intermediate assertion that prevails between each pair of statements. With assignment, it is natural to work backward, to determine these. Then work your way along using the sequential execution rule. Determining the intermediate assertions is frequently an iterative process since to go from one step to the next using the Axiom of Assignment, you may need to strengthen the pre-condition or weaken the post-condition, and these adjustments often propagate to the adjoining steps.