# Generating fixes for failed proofs: experiments and results

No Author Given

No Institute Given

The experiment ran on a Windows 11 machine with a 2.1 GHz Intel 12-Core and 32 GB of memory. AutoProof or Proof2Fix was the only computationally-intensive process running during the experiments. Version numbers for the underlying techology are: EiffelStudio 22.05; Boogie 2.11.1.0; Z3 4.8.14.

## 1 Examples with numeric computations

### 1.1 ACCOUNT

ACCOUNT, whose implementation is shown below, is a class that describes the behaviors of bank accounts; it includes a set of features representing basic operations on bank account: `deposit` (line 51), `withdraw` (line 64), and `transfer` (line 77). Fig.1 shows the verification result of this version of ACCOUNT, which suggests a complete functional correctness. Different faults are injected into the correct version, which results in 7 faulty variants of the ACCOUNT class, which will be discussed as follows.

```
1  class
2      ACCOUNT
3  create
4      make
5  feature {NONE} -- Initialization
6      make
7          -- Initialize empty account.
8          note
9              status: creator
10         do
11             balance := 0
12             credit_limit := 0
13         ensure
14             balance_set: balance = 0
15             credit_limit_set: credit_limit = 0
16         end
17 feature -- Access
18     balance: INTEGER
19         -- Balance of this account.
20     credit_limit: INTEGER
21         -- Credit limit of this account.
```

```
    available_amount: INTEGER
        -- Amount available on this account.
        note
            status: functional
        do
            Result := balance - credit_limit
        end
feature -- Basic operations
    set_credit_limit (limit: INTEGER)
        -- Set ‘credit_limit’ to ‘limit’.
        require
            limit_not_positive: limit ≤ 0
            limit_valid: limit ≤ balance
        do
            credit_limit := limit
        ensure
            modify_field (["credit_limit", "closed"], Current)
            credit_limit_set: credit_limit = limit
        end
    deposit (amount: INTEGER)
        -- Deposit ‘amount’ in this account.
        require
            amount ≥ 0
        do
            balance := balance + amount
        ensure
            modify_field (["balance", "closed"], Current)
            balance_increased: balance ≥ old balance
            balance_set: balance = old balance + amount
        end
    withdraw (amount: INTEGER)
        -- Withdraw ‘amount’ from this account.
        require
            amount_not_negative: amount ≥ 0
            amount_available: amount ≤ available_amount
        do
            balance := balance - amount
        ensure
            modify_field (["balance", "closed"], Current)
            balance_set: balance = old balance - amount
            balance_decrease: balance ≤ old balance
        end
    transfer (amount: INTEGER; other: ACCOUNT_1)
        -- Transfer ‘amount’ from this account to ‘other’.
        note
```

```
67              explicit: wrapping
68          require
69              amount_not_negative: amount ≥ 0
70              amount_available: amount ≤ available_amount
71              other ≠ Current
72          do
73              withdraw (amount)
74              other.deposit (amount)
75          ensure
76              modify_field (["balance", "closed"], [Current, other])
77              withdrawal_made: balance = old balance − amount
78              deposit_made: other.balance = old other.balance + amount
79          end
80  invariant
81      credit_limit_not_positive: credit_limit ≤ 0
82      balance_non_negative: balance − credit_limit ≥ 0
83  end
```

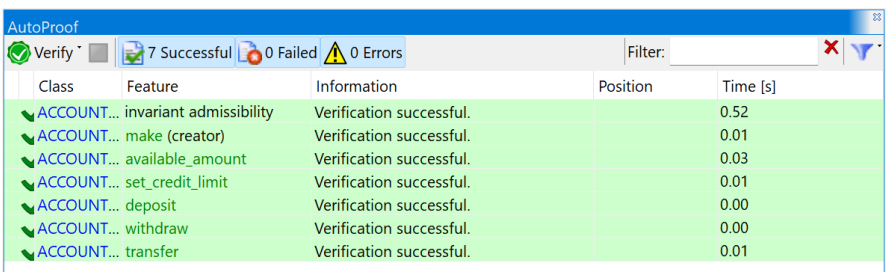



**Fig. 1.** Verification result of `ACCOUNT` in AutoProof: all routines are verified successfully (highlighted with green), which indicates that implementations of those routines are correct with respect to their specifications.

### Variant 1 of `ACCOUNT`

- Fault injection: at line 73, change the postcondition *balance_set* from "`balance = old balance − amount`" into "`balance = old balance + amount`".
- Resulting failure: as shown in Fig. 2(a), the fault results in a violation of postcondition *balance_set* of the `withdraw` procedure.
- Cause of the failure: the implementation of `withdraw` (which *deduces* `balance` by `amount`) and specification (which requires the *increment* of `balance` by `amount`) is inconsistent.
- Proof time: 0.247 sec
- Proof2Fix: No valid fixes are found.

AutoProof

Verify | 6 Successful | 1 Failed | 0 Errors | Filter:

| Class | Feature | Information | Position | Time [s] |
|---|---|---|---|---|
| ACCOUNT_1 | invariant a... | Verification successful. | | 0.40 |
| ACCOUNT_1 | make (cre... | Verification successful. | | 0.01 |
| ACCOUNT_1 | available_... | Verification successful. | | 0.03 |
| ACCOUNT_1 | set_credit_... | Verification successful. | | 0.01 |
| ACCOUNT_1 | deposit | Verification successful. | | 0.00 |
| ACCOUNT_1 | withdraw | Postcondition balance_set may be violated. | 77 | 0.03 |
| ACCOUNT_1 | transfer | Verification successful. | | 0.00 |

(a)

### Variant 2 of `ACCOUNT`

- Fault injection: at line 68, remove the precondition *amount_available* of `withdraw`.
- Resulting failure: as shown in Fig. 2(b), the class invariant *balance_non_negative* (line 96), which states that the balance (represented by `balance` − `amount`) should not be negative, is violated. (Note that a class invariant which is supposed to hold at the entry and exit of every routine.)
- Cause of the failure: the precondition of `withdraw` is too weak; there should be a precondition to constrain the amount permitted in a withdrawal operation.
- Proof time: 0.275 sec
- Proof2Fix: 4 valid fixes out of 211 candidate fixes
- Fixing time: 1.58 minutes

AutoProof

Verify | 6 Successful | 1 Failed | 0 Errors | Filter:

| Class | Feature | Information | Position | Ti... |
|---|---|---|---|---|
| ACCOU... | invariant a... | Verification successful. | | 0.18 |
| ACCOU... | make (cre... | Verification successful. | | 0.01 |
| ACCOU... | available_... | Verification successful. | | 0.03 |
| ACCOU... | set_credit_... | Verification successful. | | 0.02 |
| ACCOU... | deposit | Verification successful. | | 0.00 |
| ACCOU... | withdraw | Invariant balance_non_negative might not hold on call to {ANY}.wrap. | 79 | 0.02 |
| ACCOU... | transfer | Verification successful. | | 0.00 |

(b)

```
withdraw_ID_188 (amount: INTEGER)
  -- Withdraw ‘amount’ from this account.
require
  amount_not_negative: amount ≥ 0
    -- amount_available: amount  ≤ available_amount
        not ((amount) > (0)) -- **Fix
do
  balance := balance − amount
ensure
  modify_field (["balance", "closed"], Current)
  balance_set: balance = old balance − amount
  balance_decrease: balance ≤ old balance
end
```

**Fig. 2.** Fix from Proof2Fix

```
withdraw_ID_202 (amount: INTEGER)
  -- Withdraw ‘amount’ from this account.
require
  amount_not_negative: amount ≥ 0
    -- amount_available: amount  ≤ available_amount
        not (balance <amount) -- **Fix
do
  balance := balance − amount
ensure
  modify_field (["balance", "closed"], Current)
  balance_set: balance = old balance − amount
  balance_decrease: balance ≤ old balance
end
```

**Fig. 3.** Test case from failed proof of *withdrawal_made*

**Fig. 4.** Fix from Proof2Fix

```
withdraw_ID_206 (amount: INTEGER)
  -- Withdraw ‘amount’ from this account.
require
  amount_not_negative: amount ≥ 0
    -- amount_available: amount  ≤ available_amount
        not (credit_limit <amount) -- **Fix
do
  balance := balance − amount
ensure
  modify_field (["balance", "closed"], Current)
  balance_set: balance = old balance − amount
  balance_decrease: balance ≤ old balance
end
```

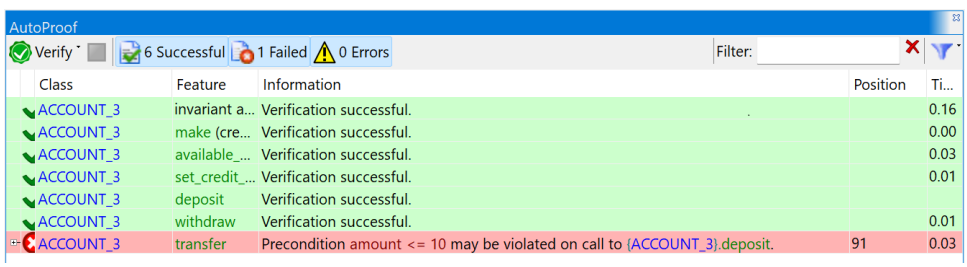
**Fig. 5.** Fix from Proof2Fix

```
withdraw_ID_210 (amount: INTEGER)
  -- Withdraw ‘amount’ from this account.
require
  amount_not_negative: amount ≥ 0
    -- amount_available: amount  ≤ available_amount
        not (amount > available_amount) -- **Fix
do
  balance := balance − amount
ensure
  modify_field (["balance", "closed"], Current)
  balance_set: balance = old balance − amount
  balance_decrease: balance ≤ old balance
end
```

**Fig. 6.** Fix from Proof2Fix

**Variant 3 of** `ACCOUNT`

- Fault injection: after line 54, add a precondition `amount` $\leq$ `10` for `deposit` to strengthen the precondition.
- Resulting failure: as shown in Fig. 7(a), the injected fault results in a failure of `transfer` — it does not satisfy the new precondition `amount` $\leq$ `10` when calling `deposit`.
- Cause of the failure: the inconsistency of specification between a supplier routine `deposit` and its client routine `transfer`: when the precondition of a routine is changed, its client routine should be changed accordingly. In this example, the upper limit of `transfer` should be consistent with the upper limit of `deposit`. In other words, the amount of money in a transfer operation should not exceed the maximum amount that is permitted in a deposit operation.
- Proof time: 0.248 sec
- Proof2Fix: 6 valid fixes out of candidate 111 fixes
- Fixing time: 1.38 minutes




(a)

```
transfer_ID_85 (amount: INTEGER; other: ACCOUNT_5)
    -- Transfer ‘amount’ from this account to ‘other’.
  note
    explicit: wrapping
  require
    other ≠ Void
    amount_not_negative: amount ≥ 0and 0≥ credit_limit and 0≥ other.credit_limit
    amount_available: amount ≤ available_amount
    balance_non_negative: balance ≥ credit_limit and other.balance ≥ other.
        credit_limit
    not ((amount) > (0)) -- **Fix
  do
    withdraw (amount)
    other.deposit (amount)
  ensure
    modify_field (["balance", "closed"], [Current, other])
    withdrawal_made: balance = old balance − amount
    desposit_made: other.balance = old other.balance + amount
  end
```

**Fig. 7.** Fix from Proof2Fix

```
transfer_ID_93 (amount: INTEGER; other: ACCOUNT_5)
    -- Transfer ‘amount’ from this account to ‘other’.
  note
    explicit: wrapping
  require
    other ≠ Void
    amount_not_negative: amount ≥ 0and 0≥ credit_limit and 0≥ other.credit_limit
    amount_available: amount ≤ available_amount
    balance_non_negative: balance ≥ credit_limit and other.balance ≥ other.
        credit_limit
    not (balance > credit_limit) -- **Fix
  do
    withdraw (amount)
    other.deposit (amount)
  ensure
    modify_field (["balance", "closed"], [Current, other])
    withdrawal_made: balance = old balance − amount
    desposit_made: other.balance = old other.balance + amount
  end
```

**Fig. 8.** Fix from Proof2Fix

```
transfer_ID_101 (amount: INTEGER; other: ACCOUNT_5)
    -- Transfer ‘amount’ from this account to ‘other’.
  note
    explicit: wrapping
  require
    other ≠ Void
    amount_not_negative: amount ≥ 0and 0≥ credit_limit and 0≥ other.credit_limit
    amount_available: amount ≤ available_amount
    balance_non_negative: balance ≥ credit_limit and other.balance ≥ other.
        credit_limit
    not (credit_limit <amount) -- **Fix
  do
    withdraw (amount)
    other.deposit (amount)
  ensure
    modify_field (["balance", "closed"], [Current, other])
    withdrawal_made: balance = old balance − amount
    desposit_made: other.balance = old other.balance + amount
  end
```

**Fig. 9.** Fix from Proof2Fix

```
transfer_ID_103 (amount: INTEGER; other: ACCOUNT_5)
    -- Transfer ‘amount’ from this account to ‘other’.
  note
    explicit: wrapping
  require
    other ≠ Void
    amount_not_negative: amount ≥ 0and 0≥ credit_limit and 0≥ other.credit_limit
    amount_available: amount ≤ available_amount
    balance_non_negative: balance ≥ credit_limit and other.balance ≥ other.
        credit_limit
    not (credit_limit <available_amount) -- **Fix
  do
    withdraw (amount)
    other.deposit (amount)
  ensure
    modify_field (["balance", "closed"], [Current, other])
    withdrawal_made: balance = old balance − amount
    desposit_made: other.balance = old other.balance + amount
  end
```

**Fig. 10.** Fix from Proof2Fix

```
transfer_ID_109 (amount: INTEGER; other: ACCOUNT_5)
    -- Transfer ‘amount’ from this account to ‘other’.
  note
    explicit: wrapping
  require
    other ≠ Void
    amount_not_negative: amount ≥ 0and 0≥ credit_limit and 0≥ other.credit_limit
    amount_available: amount ≤ available_amount
    balance_non_negative: balance ≥ credit_limit and other.balance ≥ other.
        credit_limit
    not (amount > deposit_limit) -- **Fix**
  do
    withdraw (amount)
    other.deposit (amount)
  ensure
    modify_field (["balance", "closed"], [Current, other])
    withdrawal_made: balance = old balance − amount
    desposit_made: other.balance = old other.balance + amount
  end
```

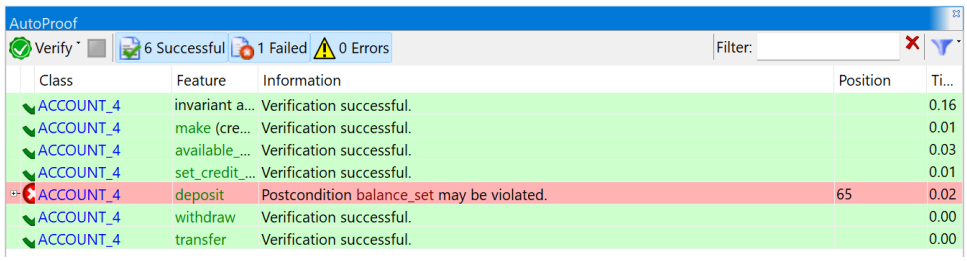
**Fig. 11.** Fix from Proof2Fix

```
transfer_ID_111 (amount: INTEGER; other: ACCOUNT_5)
    -- Transfer ‘amount’ from this account to ‘other’.
  note
    explicit: wrapping
  require
    other ≠ Void
    amount_not_negative: amount ≥ 0and 0≥ credit_limit and 0≥ other.credit_limit
    amount_available: amount ≤ available_amount
    balance_non_negative: balance ≥ credit_limit and other.balance ≥ other.
        credit_limit
    not (available_amount > deposit_limit) -- **Fix
  do
    withdraw (amount)
    other.deposit (amount)
  ensure
    modify_field (["balance", "closed"], [Current, other])
    withdrawal_made: balance = old balance − amount
    desposit_made: other.balance = old other.balance + amount
  end
```

**Fig. 12.** Fix from Proof2Fix

### Variant 4 of `ACCOUNT`

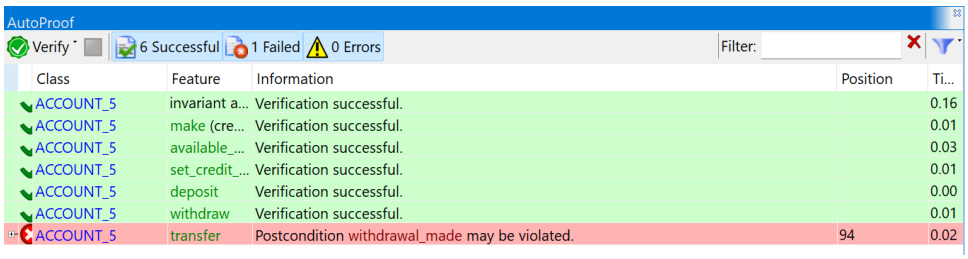
- Fault injection: at line 56, change the body of `deposit` from "`balance` := `balance` + `amount`" into "`balance` := `balance` − `amount`".
- Resulting failure: as shown in Fig. 13(a), the postcondition *balance_set* is violated.
- Cause of the failure: this failure is similar to the failure in Variant 1, which results from the inconsistency between the implementation of `deposit` and its postcondition.
- Proof time: 0.241 sec
- Possible fixes: No valid fixes




(a)

### Variant 5 of ACCOUNT

- Fault injection: at line 87, remove the precondition `other≠Current` of `transfer`.
- Resulting failure: as shown in Fig. 13(b), the fault injection leads to the violation of postcondition *withdrawal_made* when verifying `transfer`.
- Cause of the failure: the precondition of `transfer` is too weak; it should exclude the case where an account transfers money to itself.
- Proof time: 0.243 sec
- Fixing time:
- Proof2Fix: 5 valid fixes out of 98 candidate fixes




| Class | Feature | Information | Position | Ti... |
|---|---|---|---|---|
| ACCOUNT_5 | invariant a... | Verification successful. | | 0.16 |
| ACCOUNT_5 | make (cre... | Verification successful. | | 0.01 |
| ACCOUNT_5 | available_... | Verification successful. | | 0.03 |
| ACCOUNT_5 | set_credit_... | Verification successful. | | 0.01 |
| ACCOUNT_5 | deposit | Verification successful. | | 0.00 |
| ACCOUNT_5 | withdraw | Verification successful. | | 0.01 |
| ACCOUNT_5 | transfer | Postcondition withdrawal_made may be violated. | 94 | 0.02 |

(b)

```
1   transfer_ID_71 (amount: INTEGER; other: ACCOUNT_5)
2       -- Transfer 'amount' from this account to 'other'.
3     note
4       explicit: wrapping
5     require
6       other ≠ Void
7       amount_not_negative: amount ≥ 0and 0≥ credit_limit and 0≥ other.credit_limit
8       amount_available: amount ≤ available_amount
9       balance_non_negative: balance ≥ credit_limit and other.balance ≥ other.
            credit_limit
10      not (other = Current) -- **Fix
11    do
12      withdraw (amount)
13      other.deposit (amount)
14    ensure
15      modify_field ("balance", "closed"], [Current, other])
16      withdrawal_made: balance = old balance − amount
17      desposit_made: other.balance = old other.balance + amount
18    end
```

**Fig. 13.** Fix from Proof2Fix

```
transfer_ID_75 (amount: INTEGER; other: ACCOUNT_5)
    -- Transfer ‘amount’ from this account to ‘other’.
  note
    explicit: wrapping
  require
    other ≠ Void
    amount_not_negative: amount ≥ 0and 0≥ credit_limit and 0≥ other.credit_limit
    amount_available: amount ≤ available_amount
    balance_non_negative: balance ≥ credit_limit and other.balance ≥ other.
        credit_limit
         not ((other = Void) or else (other = Current))
  do
    withdraw (amount)
    other.deposit (amount)
  ensure
    modify_field (["balance", "closed"], [Current, other])
    withdrawal_made: balance = old balance − amount
    desposit_made: other.balance = old other.balance + amount
  end
```

**Fig. 14.** Fix from Proof2Fix

```
transfer_ID_85 (amount: INTEGER; other: ACCOUNT_5)
    -- Transfer ‘amount’ from this account to ‘other’.
  note
    explicit: wrapping
  require
    other ≠ Void
    amount_not_negative: amount ≥ 0and 0≥ credit_limit and 0≥ other.credit_limit
    amount_available: amount ≤ available_amount
    balance_non_negative: balance ≥ credit_limit and other.balance ≥ other.
        credit_limit
         not ((amount) > (0))
  do
    withdraw (amount)
    other.deposit (amount)
  ensure
    modify_field (["balance", "closed"], [Current, other])
    withdrawal_made: balance = old balance − amount
    desposit_made: other.balance = old other.balance + amount
  end
```

**Fig. 15.** Fix from Proof2Fix

```
transfer_ID_93 (amount: INTEGER; other: ACCOUNT_5)
      -- Transfer ‘amount’ from this account to ‘other’.
    note
      explicit: wrapping
    require
      other ≠ Void
      amount_not_negative: amount ≥ 0and 0≥ credit_limit and 0≥ other.credit_limit
      amount_available: amount ≤ available_amount
      balance_non_negative: balance ≥ credit_limit and other.balance ≥ other.
           credit_limit
            not (balance > credit_limit)
    do
      withdraw (amount)
      other.deposit (amount)
    ensure
      modify_field (["balance", "closed"], [Current, other])
      withdrawal_made: balance = old balance − amount
      desposit_made: other.balance = old other.balance + amount
    end
```

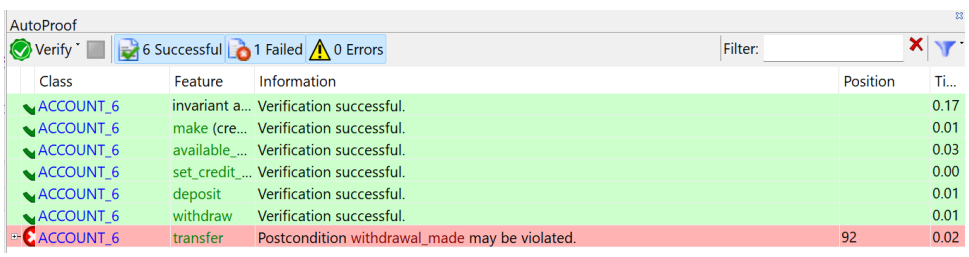
**Fig. 16.** Fix from Proof2Fix

```
transfer_ID_97 (amount: INTEGER; other: ACCOUNT_5)
      -- Transfer ‘amount’ from this account to ‘other’.
    note
      explicit: wrapping
    require
      other ≠ Void
      amount_not_negative: amount ≥ 0and 0≥ credit_limit and 0≥ other.credit_limit
      amount_available: amount ≤ available_amount
      balance_non_negative: balance ≥ credit_limit and other.balance ≥ other.
           credit_limit
            not (credit_limit <amount)
    do

      withdraw (amount)
      other.deposit (amount)
    ensure
      modify_field (["balance", "closed"], [Current, other])
      withdrawal_made: balance = old balance − amount
      desposit_made: other.balance = old other.balance + amount
    end
```

**Fig. 17.** Fix from Proof2Fix

### Variant 6 of ACCOUNT

- Fault injection: at line 73, remove the postcondition *balance_set* of `withdraw`.
- Resulting failure: as shown in Fig. 18(a), the injected fault results in the violation of postcondition *withdrawal_made* when verifying `transfer`.
- Cause of the failure: the postcondition of `withdraw` is incomplete — not strong enough to represent the functionality of `withdraw`; as a result, when reasoning its client routine `transfer`, the prover is not able to establish the postcondition related to the functionality of `withdraw`.
- Proof time: 0.253 sec
- Possible fixes: No valid fixes

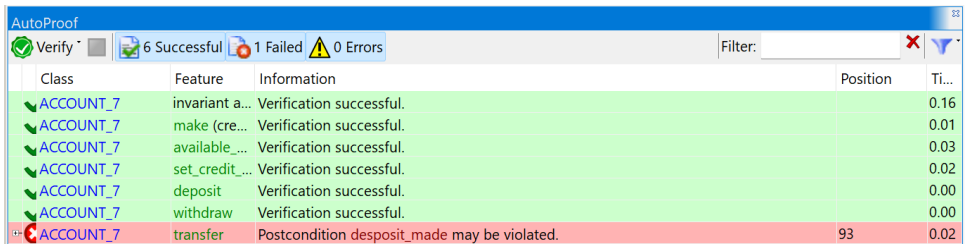



| Class | Feature | Information | Position | Ti... |
|---|---|---|---|---|
| ACCOUNT_6 | invariant a... | Verification successful. | | 0.17 |
| ACCOUNT_6 | make (cre... | Verification successful. | | 0.01 |
| ACCOUNT_6 | available_... | Verification successful. | | 0.03 |
| ACCOUNT_6 | set_credit_... | Verification successful. | | 0.00 |
| ACCOUNT_6 | deposit | Verification successful. | | 0.01 |
| ACCOUNT_6 | withdraw | Verification successful. | | 0.01 |
| ACCOUNT_6 | transfer | Postcondition withdrawal_made may be violated. | 92 | 0.02 |

(a)

**Variant 7 of ACCOUNT**

- Fault injection: at line 60, remove the postcondition *balance_set* of `deposit`.
- Resulting failure: the injected fault, as shown in Fig. 18(b), results in the violation of postcondition *deposit_made* when verifying `transfer`.
- Cause of the failure: similar to the previous failure (in Variant 6), this failure of `transfer` is due to the weakness of the postcondition of its supplier class `deposit` – the postcondition is not strong enough to represent the functionality of `deposit`.
- Proof time: 0.244 sec
- Resulting test case: Fig. **??** shows the test from Proof2Test, which calls `transfer` with input — `Current.balance` = 0, `Current.credit_limit` = 0, `amount` = 0, `other.balance` = 0, and `other.credit_limit` = −7720.
- Possible fixes: No valid fixes




(b)

## 1.2 CLOCK

CLOCK class implements a digital clock counting seconds, minutes, and hours. The version of CLOCK displaying below is verified successfully. The experiment includes 8 different variants of CLOCK class, with different faults injected based on the verified version.

```
class
    CLOCK
create
    make
feature {NONE} -- Initialization
    make
        note
            status: creator
        do
            hours := 0
            minutes := 0
```

```
            seconds := 0
        ensure
            modify_model ([“hours”, “minutes”, “seconds”], Current)
            initialized: hours = 0 and minutes = 0 and seconds = 0
        end
feature -- Access
    hours: INTEGER
            -- Hours of clock.
    minutes: INTEGER
            -- Minutes of clock.
    seconds: INTEGER
            -- Seconds of clock.
feature -- Element change
    set_hours (a_value: INTEGER)
            -- Set ‘hours’ to ‘a_value’.
        require
            valid_hours: 0 ≤ a_value and a_value <24
        do
            hours := a_value
        ensure
            hours_set: hours = a_value
            modify_model (“hours”, Current)
        end
    set_minutes (a_value: INTEGER)
            -- Set ‘minutes’ to ‘a_value’.
        require
            valid_minutes: 0 ≤ a_value and a_value <60
        do
            minutes := a_value
        ensure
            minutes_set: minutes = a_value
            modify_model (“minutes”, Current)
        end
    set_seconds (a_value: INTEGER)
            -- Set ‘seconds’ to ‘a_value’.
        require
            valid_seconds: 0 ≤ a_value and a_value <60
        do
            seconds := a_value
        ensure
            seconds_set: seconds = a_value
            modify_model (“seconds”, Current)
        end
feature -- Basic operations
    increase_hours
```

```
            -- Increase ‘hours’ by one.
        note
            explicit: wrapping
        do
            if hours = 23 then
                set_hours (0)
            else
                set_hours (hours + 1)
            end
        ensure
            hours_increased: hours = (old hours + 1) \\ 24
            modify_model (“hours”, Current)
        end
    increase_minutes
            -- Increase ‘minutes’ by one.
        note
            explicit: wrapping
        do
            if minutes <59 then
                set_minutes (minutes + 1)
            else
                set_minutes (0)
                increase_hours
            end
        ensure
            hours_increased: old minutes = 59 implies
                                                hours = (old hours + 1) \\ 24
            hours_unchanged: old minutes <59 implies
                                                hours = old hours
            minutes_increased: minutes = (old minutes + 1) \\ 60
            modify_model (["minutes", "hours"], Current)
        end
    increase_seconds
            -- Increase ‘seconds’ by one.
        note
            explicit: wrapping
        do
            if seconds ≥ 59 then
                set_seconds (0)
                increase_minutes
            else
                set_seconds (seconds + 1)
            end
        ensure
            hours_increased: old seconds = 59 and old minutes = 59 implies
```

```
                                        hours = (old hours + 1) \\ 24
            hours_unchanged: old seconds < 59 or old minutes <59 implies
                                        hours = old hours
            minutes_increased: old seconds = 59 implies
                                        minutes = (old minutes + 1) \\ 60
            minutes_unchanged: old seconds < 59 implies
                                        minutes = old minutes
            seonds_inreased: seconds = (old seconds + 1) \\ 60
            modify_model ([“seconds”, “minutes”, “hours”], Current)
        end
invariant
    hours_valid: 0 ≤ hours and hours ≤ 23
    minutes_valid: 0 ≤ minutes and minutes ≤ 59
    seconds_valid: 0 ≤ seconds and seconds ≤ 59
end
```

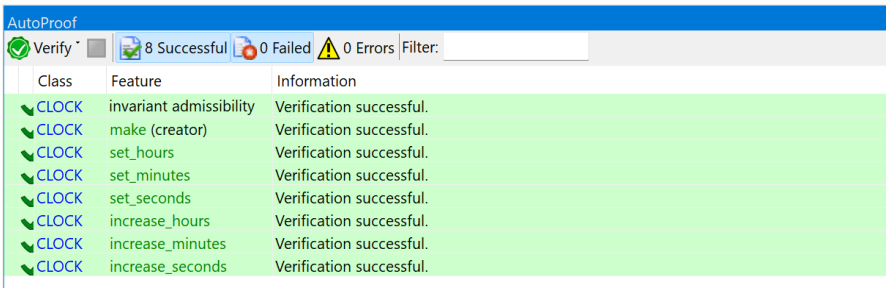



**Fig. 18.** Proof result of CLOCK in AutoProof

### Variant 1 of CLOCK

- Fault injection: at line 73, in the `increase_hours` procedure, change the condition of the then branch from “`hours = 23`” into “`hours = 24`”.
- Resulting failure: as shown in Fig. 19(a), the injected fault results in the violation of precondition `valid_hours` when calling `set_hours` from `increase_hours`.
- Cause of the failure: incorrect implementation of the routine body.
- Proof time: 0.253 sec
- Proof2Fix: 5 valid fixes out of 356 candidate fixes
- Fixing time: 2.72 minutes

AutoProof

Verify | 7 Successful | 1 Failed | 0 Errors | Filter:

| Class | Feature | Information |
|---|---|---|
| CLOCK_1 | invariant admissibility | Verification successful. |
| CLOCK_1 | make (creator) | Verification successful. |
| CLOCK_1 | set_hours | Verification successful. |
| CLOCK_1 | set_minutes | Verification successful. |
| CLOCK_1 | set_seconds | Verification successful. |
| CLOCK_1 | increase_hours | Precondition valid_hours may be violated on call to {CLOCK_1}.set_hours. |
| CLOCK_1 | increase_minutes | Verification successful. |
| CLOCK_1 | increase_seconds | Verification successful. |

(a)

```
increase_hours_ID_185
    -- Increase 'hours' by one.
  note
    explicit: wrapping
  do
    if hours = 23then -- **Fix
      hours := 0-- **Fix
    else
      if hours = 24then
        set_hours (0)
      else
        set_hours (hours + 1)
      end
    end
  ensure
    hours_increased: hours = (old hours + 1) \\ 24
    modify_model ("hours", Current)
  end
```

**Fig. 19.** Fix from Proof2Fix

```
increase_hours_ID_44
    -- Increase 'hours' by one.
  note
    explicit: wrapping
  do
    if (hours) > (0) then -- **Fix
      increase_hours -- **Fix
    else
      if hours = 24then
        set_hours (0)
      else
        set_hours (hours + 1)
      end
    end
  ensure
    hours_increased: hours = (old hours + 1) \\ 24
    modify_model ("hours", Current)
  end
```

**Fig. 20.** Fix from Proof2Fix

```
increase_hours_ID_78
    -- Increase 'hours' by one.
  note
    explicit: wrapping
  do
    if (hours) ≥ (0) then -- **Fix
      increase_hours -- **Fix
    else
      if hours = 24then
        set_hours (0)
      else
        set_hours (hours + 1)
      end
    end
  ensure
    hours_increased: hours = (old hours + 1) \\ 24
    modify_model ("hours", Current)
  end
```

**Fig. 21.** Fix from Proof2Fix

```
increase_hours_ID_180
    -- Increase 'hours' by one.
  note
    explicit: wrapping
  do
    if hours = 23then -- **Fix
      increase_hours -- **Fix
    else
      if hours = 24then
        set_hours (0)
      else
        set_hours (hours + 1)
      end
    end
  ensure
    hours_increased: hours = (old hours + 1) \\ 24
    modify_model ("hours", Current)
  end
```

**Fig. 22.** Fix from Proof2Fix

```
increase_hours_ID_342
    -- Increase 'hours' by one.
  note
    explicit: wrapping
      require -- **Fix
          not ((hours) > (0)) -- **Fix
  do
    if hours = 24then
      set_hours (0)
    else
      set_hours (hours + 1)
    end
  ensure
    hours_increased: hours = (old hours + 1) \\ 24
    modify_model ("hours", Current)
  end
```

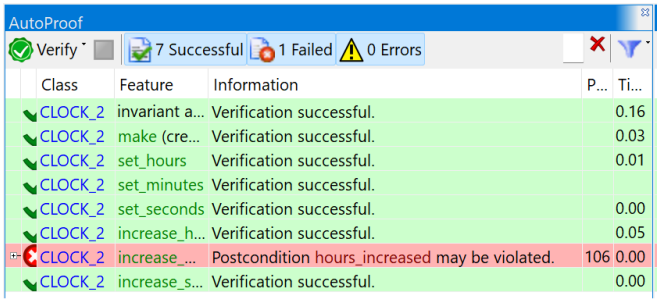
**Fig. 23.** Fix from Proof2Fix

```
increase_hours_ID_350
    -- Increase 'hours' by one.
  note
    explicit: wrapping
      require
          not (hours = 23) -- **Fix
  do
    if hours = 24then
      set_hours (0)
    else
      set_hours (hours + 1)
    end
  ensure
    hours_increased: hours = (old hours + 1) \\ 24
    modify_model ("hours", Current)
  end
```

**Fig. 24.** Fix from Proof2Fix

### Variant 2 of CLOCK

- Fault injection: at line 92, remove the call `increase_hours` in the body of `increase_minutes`.
- Resulting failure: as shown in Fig. 25(a), the postcondition `hours_increased` is not satisfied in the proof of `increase_minutes`.
- Cause of the failure: incorrect implementation of the routine body.
- Proof time: 0.251 sec
- Proof2Fix: 11 valid fixes out of 301 candidate fixes
- fixing time: 2.43 minutes




(a)

```
increase_minutes_ID_50
    -- Increase ‘minutes’ by one.
  note
    explicit: wrapping
  do
          if (minutes) > (0) then -- **Fix
              increase_minutes -- **Fix
          else
         if minutes <59then
           set_minutes (minutes + 1)
         else
          set_minutes (0)
         end
          end
  ensure
    hours_increased: old minutes = 59implies hours = (old hours + 1) \\ 24
    hours_unchanged: old minutes <59implies hours = old hours
    minutes_increased: minutes = (old minutes + 1) \\ 60
    modify_model (["minutes", "hours"], Current)
  end
```

**Fig. 25.** Fix from Proof2Fix

```
increase_minutes_ID_84
    -- Increase ‘minutes’ by one.
  note
    explicit: wrapping
  do
          if (minutes) ≥ (0) then -- **Fix
              increase_minutes -- **Fix
          else
         if minutes <59then
           set_minutes (minutes + 1)
         else
          set_minutes (0)
         end
          end
  ensure
    hours_increased: old minutes = 59implies hours = (old hours + 1) \\ 24
    hours_unchanged: old minutes <59implies hours = old hours
    minutes_increased: minutes = (old minutes + 1) \\ 60
    modify_model (["minutes", "hours"], Current)
  end
```

**Fig. 26.** Fix from Proof2Fix

```
increase_minutes_ID_177
    -- Increase ‘minutes’ by one.
  note
    explicit: wrapping
  do
          if minutes = 59then -- **Fix**
              increase_hours -- **Fix**
          end
    if minutes < 59then
         set_minutes (minutes + 1)
    else
        set_minutes (0)
    end
  ensure
    hours_increased: old minutes = 59implies hours = (old hours + 1) \\ 24
    hours_unchanged: old minutes < 59implies hours = old hours
    minutes_increased: minutes = (old minutes + 1) \\ 60
    modify_model (["minutes", "hours"], Current)
  end
```

**Fig. 27.** Fix from Proof2Fix

```
increase_minutes_ID_186
    -- Increase ‘minutes’ by one.
  note
    explicit: wrapping
  do
          if minutes = 59then -- **Fix
              increase_minutes -- **Fix
          end
    if minutes < 59then
         set_minutes (minutes + 1)
    else
        set_minutes (0)
    end
  ensure
    hours_increased: old minutes = 59implies hours = (old hours + 1) \\ 24
    hours_unchanged: old minutes < 59implies hours = old hours
    minutes_increased: minutes = (old minutes + 1) \\ 60
    modify_model (["minutes", "hours"], Current)
  end
```

**Fig. 28.** Fix from Proof2Fix

```
increase_minutes_ID_230
    -- Increase ‘minutes’ by one.
  note
    explicit: wrapping
  do
          if hours <minutes then -- **Fix
              increase_minutes -- **Fix
          end
    if minutes <59then
         set_minutes (minutes + 1)
    else
        set_minutes (0)
    end
  ensure
    hours_increased: old minutes =59implies hours =(old hours + 1) \\ 24
    hours_unchanged: old minutes <59implies hours =old hours
    minutes_increased: minutes =(old minutes + 1) \\ 60
    modify_model (["minutes", "hours"], Current)
  end
```

**Fig. 29.** Fix from Proof2Fix

```
increase_minutes_ID_287
    -- Increase ‘minutes’ by one.
  note
    explicit: wrapping
  do
    if minutes <59then
         set_minutes (minutes + 1)
    else
        set_minutes (0)
    end
  ensure
    hours_increased: not ((minutes) =(0)) implies old minutes =59implies hours =(
         old hours + 1) \\ 24-- **Fix
    hours_unchanged: old minutes <59implies hours =old hours
    minutes_increased: minutes =(old minutes + 1) \\ 60
    modify_model (["minutes", "hours"], Current)
  end
```

**Fig. 30.** Fix from Proof2Fix

```
increase_minutes_ID_288
    -- Increase ‘minutes’ by one.
  note
    explicit: wrapping
      require
          not ((minutes) > (0)) -- **Fix
  do
    if minutes <59then
         set_minutes (minutes + 1)
    else
        set_minutes (0)
    end
  ensure
    hours_increased: old minutes = 59implies hours = (old hours + 1) \\ 24
    hours_unchanged: old minutes <59implies hours = old hours
    minutes_increased: minutes = (old minutes + 1) \\ 60
    modify_model (["minutes", "hours"], Current)
  end
```

**Fig. 31.** Fix from Proof2Fix

```
increase_minutes_ID_291
    -- Increase ‘minutes’ by one.
  note
    explicit: wrapping
  do
    if minutes <59then
         set_minutes (minutes + 1)
    else
        set_minutes (0)
    end
  ensure
    hours_increased: not ((minutes) ≥ (0))implies old minutes = 59implies hours = (
         old hours + 1) \\ 24-- **Fix
    hours_unchanged: old minutes <59implies hours = old hours
    minutes_increased: minutes = (old minutes + 1) \\ 60
    modify_model (["minutes", "hours"], Current)
  end
```

**Fig. 32.** Fix from Proof2Fix

```
increase_minutes_ID_295
    -- Increase ‘minutes’ by one.
  note
    explicit: wrapping
  do
    if minutes <59then
         set_minutes (minutes + 1)
    else
        set_minutes (0)
    end
  ensure
    hours_increased: not ((minutes) ≤ (0)) implies old minutes = 59implies hours =
         (old hours + 1) \\ 24-- **Fix
    hours_unchanged: old minutes <59implies hours = old hours
    minutes_increased: minutes = (old minutes + 1) \\ 60
    modify_model (["minutes", "hours"], Current)
  end
```

**Fig. 33.** Fix from Proof2Fix

```
increase_minutes_ID_296
    -- Increase ‘minutes’ by one.
  note
    explicit: wrapping
      require
          not (minutes = 59) -- **Fix
  do
    if minutes <59then
         set_minutes (minutes + 1)
    else
        set_minutes (0)
    end
  ensure
    hours_increased: old minutes = 59implies hours = (old hours + 1) \\ 24
    hours_unchanged: old minutes <59implies hours = old hours
    minutes_increased: minutes = (old minutes + 1) \\ 60
    modify_model (["minutes", "hours"], Current)
  end
```

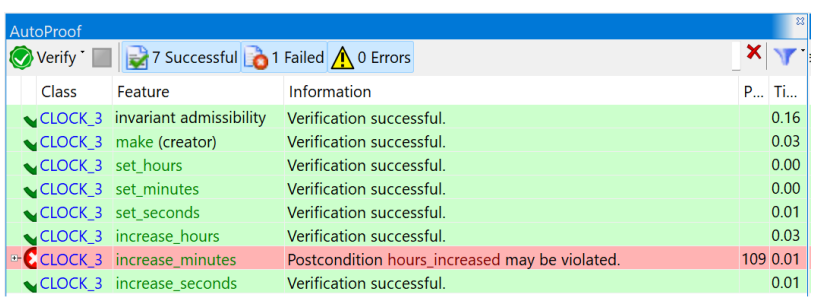
**Fig. 34.** Fix from Proof2Fix

```
increase_minutes_ID_298
    -- Increase 'minutes' by one.
  note
    explicit: wrapping
      require
          not (hours <minutes) -- **Fix
  do
    if minutes <59then
         set_minutes (minutes + 1)
    else
        set_minutes (0)
    end
  ensure
    hours_increased: old minutes =59implies hours =(old hours + 1) \\ 24
    hours_unchanged: old minutes <59implies hours =old hours
    minutes_increased: minutes =(old minutes + 1) \\ 60
    modify_model (["minutes", "hours"], Current)
  end
```

**Fig. 35.** Fix from Proof2Fix

### Variant 3 of `CLOCK`

- Fault injection: at line 79, remove the postcondition of `hours_increased` in the `increased_hours` procedure.
- Resulting failure: as shown in Fig. 36(a), the injected fault leads to violation of postcondition `hours_increased` when verifying `increase_minutes`.
- Cause of the failure: the postcondition of `increase_hours` is too weak to express the full functionality of the routine.
- Proof time: 0.263 sec
- Proof2Fix: No valid fixes




| Class | Feature | Information | P... | Ti... |
|---|---|---|---|---|
| CLOCK_3 | invariant admissibility | Verification successful. | | 0.16 |
| CLOCK_3 | make (creator) | Verification successful. | | 0.03 |
| CLOCK_3 | set_hours | Verification successful. | | 0.00 |
| CLOCK_3 | set_minutes | Verification successful. | | 0.00 |
| CLOCK_3 | set_seconds | Verification successful. | | 0.01 |
| CLOCK_3 | increase_hours | Verification successful. | | 0.03 |
| CLOCK_3 | increase_minutes | Postcondition hours_increased may be violated. | 109 | 0.01 |
| CLOCK_3 | increase_seconds | Verification successful. | | 0.01 |

(a)

### Variant 4 of `CLOCK`

- Fault injection: at line 51, remove the postcondition of `minutes_set` in the `set_minutes` procedure.
- Resulting failure: as shown in Fig. 36(b), the postcondition `minutes_increased` is not satisfied when verifying the `increase_minutes` procedure.
- Cause of the failure: the postcondition of supplier routine `set_minutes` is too weak to represent its full functionality.
- Proof time: 0.259 sec
- Proof2Fix: No valid fixes

AutoProof

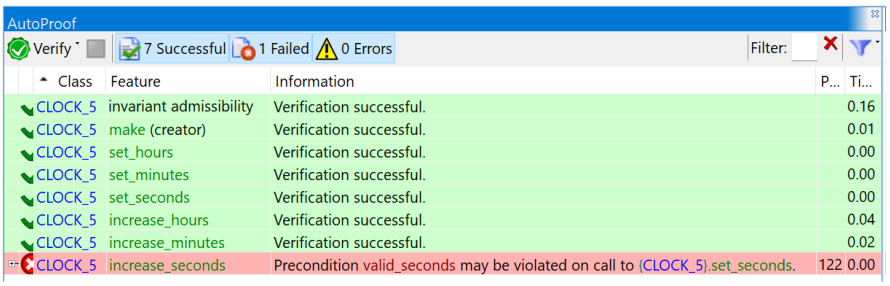
Verify | 7 Successful | 1 Failed | 0 Errors

| Class | Feature | Information | P... | Ti... |
|---|---|---|---|---|
| CLOCK_4 | invariant admissibility | Verification successful. | | 0.16 |
| CLOCK_4 | make (creator) | Verification successful. | | 0.03 |
| CLOCK_4 | set_hours | Verification successful. | | 0.05 |
| CLOCK_4 | set_minutes | Verification successful. | | 0.00 |
| CLOCK_4 | set_seconds | Verification successful. | | 0.00 |
| CLOCK_4 | increase_hours | Verification successful. | | 0.00 |
| CLOCK_4 | increase_minutes | Postcondition minutes_increased may be violated. | 110 | 0.00 |
| CLOCK_4 | increase_seconds | Verification successful. | | 0.01 |

(b)

**Variant 5 of CLOCK**

- Fault injection: at line 108, change the condition of the then branch from "`seconds`≥`59`" into "`seconds`>`59`".
- Resulting failure: as shown in Fig. 36(c), the injected fault leads to the violation of the precondition `valid_seconds` of `set_seconds` when it is called from the procedure `increase_seconds`.
- Cause of the failure: incorrect implementation of the routine body of `increase_seconds`.
- Proof time: 0.241 sec
- Comment: this test is useful (the values in the test input are meaningful) as its execution shows a specific path that leads to the same contract violation as in the proof.
- Proof2Fix: 8 valid fixes out of 356 candidate fixes.
- fixing time: 3.1 minutes




(c)

```
increase_seconds_ID_174
    -- Increase ‘seconds’ by one.
  note
    explicit: wrapping
  do
          if seconds = 59then -- **Fix**
              seconds := 0-- **Fix**
          end
    if seconds > 59then
      set_seconds (0)
      increase_minutes
    else
      set_seconds (seconds + 1)
    end
  ensure
    hours_increased: old seconds = 59and old minutes = 59implies hours = (old hours
        + 1) \\ 24
    hours_unchanged: old seconds <59or old minutes <59implies hours = old hours
    minutes_increased: old seconds = 59implies minutes = (old minutes + 1) \\ 60
    minutes_unchanged: old seconds <59implies minutes = old minutes
    seonds_inreased: seconds = (old seconds + 1) \\ 60
    modify_model (["seconds", "minutes", "hours"], Current)
  end
```

**Fig. 36.** Test case from failed proof of *withdrawal_made*

```
1    increase_seconds_ID_51
2        -- Increase ‘seconds’ by one.
3      note
4        explicit: wrapping
5      do
6              if (seconds) > (0) then
7              increase_seconds
8              else
9        if seconds > 59then
10         set_seconds (0)
11         increase_minutes
12       else
13         set_seconds (seconds + 1)
14       end
15 end
16     ensure
17       hours_increased: old seconds = 59and old minutes = 59implies hours = (old hours
             + 1) \\ 24
18       hours_unchanged: old seconds <59or old minutes <59implies hours = old hours
19       minutes_increased: old seconds = 59implies minutes = (old minutes + 1) \\ 60
20       minutes_unchanged: old seconds <59implies minutes = old minutes
21       seonds_inreased: seconds = (old seconds + 1) \\ 60
22       modify_model (["seconds", "minutes", "hours"], Current)
23     end
```

**Fig. 37.** Test case from failed proof of *withdrawal_made*

```
  increase_seconds_ID_85
      -- Increase ‘seconds’ by one.
    note
      explicit: wrapping
    do
            if (seconds) ≥ (0) then
increase_seconds
else
      if seconds > 59then
        set_seconds (0)
        increase_minutes
      else
        set_seconds (seconds + 1)
      end
end
    ensure
      hours_increased: old seconds = 59and old minutes = 59implies hours = (old hours
            + 1) \\ 24
      hours_unchanged: old seconds <59or old minutes <59implies hours = old hours
      minutes_increased: old seconds = 59implies minutes = (old minutes + 1) \\ 60
      minutes_unchanged: old seconds <59implies minutes = old minutes
      seonds_inreased: seconds = (old seconds + 1) \\ 60
      modify_model (["seconds", "minutes", "hours"], Current)
    end
```

**Fig. 38.** Test case from failed proof of *withdrawal_made*

```
  increase_seconds_ID_187
      -- Increase ‘seconds’ by one.
    note
      explicit: wrapping
    do
            if seconds = 59then
increase_seconds
else
      if seconds > 59then
        set_seconds (0)
        increase_minutes
      else
        set_seconds (seconds + 1)
      end
end
    ensure
      hours_increased: old seconds = 59and old minutes = 59implies hours = (old hours
            + 1) \\ 24
      hours_unchanged: old seconds <59or old minutes <59implies hours = old hours
      minutes_increased: old seconds = 59implies minutes = (old minutes + 1) \\ 60
      minutes_unchanged: old seconds <59implies minutes = old minutes
      seonds_inreased: seconds = (old seconds + 1) \\ 60
      modify_model (["seconds", "minutes", "hours"], Current)
    end
```

**Fig. 39.** Test case from failed proof of *withdrawal_made*

```
  increase_seconds_ID_285
      -- Increase ‘seconds’ by one.
    note
      explicit: wrapping
    do
            if hours <seconds then
increase_seconds
else
      if seconds > 59then
        set_seconds (0)
        increase_minutes
      else
        set_seconds (seconds + 1)
      end
end
    ensure
      hours_increased: old seconds = 59and old minutes = 59implies hours = (old hours
            + 1) \\ 24
      hours_unchanged: old seconds <59or old minutes <59implies hours = old hours
      minutes_increased: old seconds = 59implies minutes = (old minutes + 1) \\ 60
      minutes_unchanged: old seconds <59implies minutes = old minutes
      seonds_inreased: seconds = (old seconds + 1) \\ 60
      modify_model (["seconds", "minutes", "hours"], Current)
    end
```

**Fig. 40.** Test case from failed proof of *withdrawal_made*

```
1    increase_seconds_ID_342
2        -- Increase ‘seconds’ by one.
3      note
4        explicit: wrapping
5  require
6  not ((seconds) > (0))
7      do
8        if seconds > 59then
9          set_seconds (0)
10         increase_minutes
11       else
12         set_seconds (seconds + 1)
13       end
14     ensure
15       hours_increased: old seconds = 59and old minutes = 59implies hours = (old hours
             + 1) \\ 24
16       hours_unchanged: old seconds <59or old minutes <59implies hours = old hours
17       minutes_increased: old seconds = 59implies minutes = (old minutes + 1) \\ 60
18       minutes_unchanged: old seconds <59implies minutes = old minutes
19       seonds_inreased: seconds = (old seconds + 1) \\ 60
20       modify_model (["seconds", "minutes", "hours"], Current)
21     end
```

**Fig. 41.** Test case from failed proof of *withdrawal_made*

### Variant 6 of CLOCK

- Fault injection: at line 95, remove the postcondition `hours_increased` of `increase_minutes` procedure.
- Resulting failure: as shown in Fig. 44(a), the injected fault results in the violation of the postcondition `hours_increased` of `increase_seconds`.
- Cause of the failure: the postcondition of the routine `increase_minutes` is too weak to represent its full functionality.
- Proof time: 0.243 sec
- Possible fixes: No valid fixes

```
1    increase_seconds_ID_350
2        -- Increase ‘seconds’ by one.
3      note
4        explicit: wrapping
5  require
6  not (seconds = 59)
7      do
8        if seconds > 59then
9          set_seconds (0)
10         increase_minutes
11       else
12         set_seconds (seconds + 1)
13       end
14     ensure
15       hours_increased: old seconds = 59and old minutes = 59implies hours = (old hours
             + 1) \\ 24
16       hours_unchanged: old seconds <59or old minutes <59implies hours = old hours
17       minutes_increased: old seconds = 59implies minutes = (old minutes + 1) \\ 60
18       minutes_unchanged: old seconds <59implies minutes = old minutes
19       seonds_inreased: seconds = (old seconds + 1) \\ 60
20       modify_model (["seconds", "minutes", "hours"], Current)
21     end
```

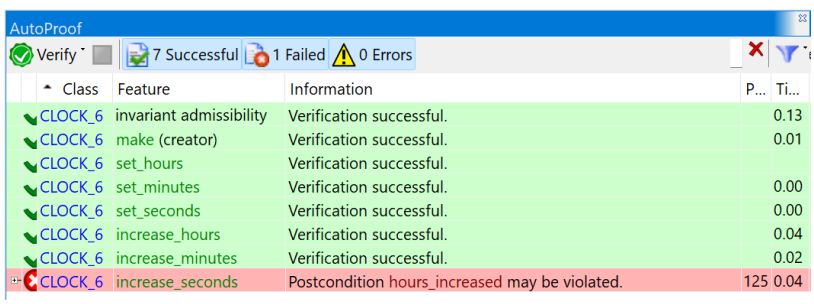
**Fig. 42.** Test case from failed proof of *withdrawal_made*

### Variant 7 of CLOCK

- Fault injection: at line 99, remove the postcondition of `minutes_increased` of `increase_minutes` procedure.
- Resulting failure: as shown in Fig. 44(b), the postcondition `hours_increased` is violated.
- Cause of the failure: the postcondition of the routine `increase_minutes` is too weak to represent its full functionality.
- Proof time: 0.248 sec
- Possible fixes: No valid fixes

```
  increase_seconds_ID_354
      -- Increase 'seconds' by one.
    note
      explicit: wrapping
require
not (hours <seconds)
    do
      if seconds > 59then
        set_seconds (0)
        increase_minutes
      else
        set_seconds (seconds + 1)
      end
    ensure
      hours_increased: old seconds = 59and old minutes = 59implies hours = (old hours
          + 1) \\ 24
      hours_unchanged: old seconds <59or old minutes <59implies hours = old hours
      minutes_increased: old seconds = 59implies minutes = (old minutes + 1) \\ 60
      minutes_unchanged: old seconds <59implies minutes = old minutes
      seonds_inreased: seconds = (old seconds + 1) \\ 60
      modify_model (["seconds", "minutes", "hours"], Current)
    end
```

**Fig. 43.** Test case from failed proof of *withdrawal_made*

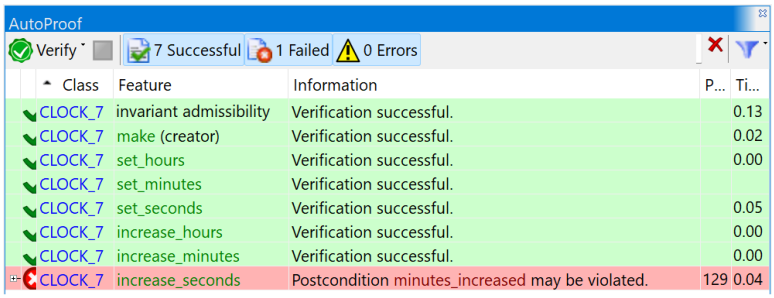


AutoProof

Verify | 7 Successful | 1 Failed | 0 Errors

| Class | Feature | Information | P... | Ti... |
|---|---|---|---|---|
| CLOCK_6 | invariant admissibility | Verification successful. | | 0.13 |
| CLOCK_6 | make (creator) | Verification successful. | | 0.01 |
| CLOCK_6 | set_hours | Verification successful. | | |
| CLOCK_6 | set_minutes | Verification successful. | | 0.00 |
| CLOCK_6 | set_seconds | Verification successful. | | 0.00 |
| CLOCK_6 | increase_hours | Verification successful. | | 0.04 |
| CLOCK_6 | increase_minutes | Verification successful. | | 0.02 |
| CLOCK_6 | increase_seconds | Postcondition hours_increased may be violated. | 125 | 0.04 |

(a)



AutoProof

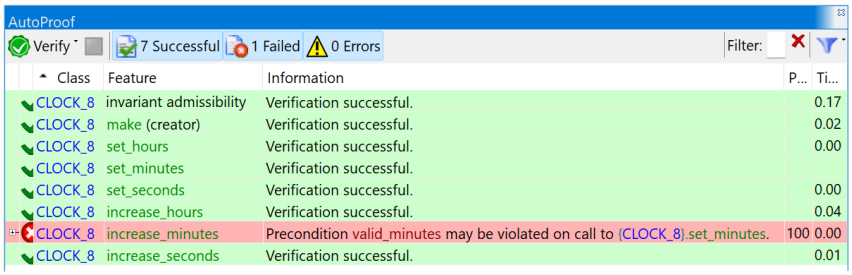
Verify | 7 Successful | 1 Failed | 0 Errors

| Class | Feature | Information | P... | Ti... |
|---|---|---|---|---|
| CLOCK_7 | invariant admissibility | Verification successful. | | 0.13 |
| CLOCK_7 | make (creator) | Verification successful. | | 0.02 |
| CLOCK_7 | set_hours | Verification successful. | | 0.00 |
| CLOCK_7 | set_minutes | Verification successful. | | |
| CLOCK_7 | set_seconds | Verification successful. | | 0.05 |
| CLOCK_7 | increase_hours | Verification successful. | | 0.00 |
| CLOCK_7 | increase_minutes | Verification successful. | | 0.00 |
| CLOCK_7 | increase_seconds | Postcondition minutes_increased may be violated. | 129 | 0.04 |

(b)

### Variant 8 of CLOCK

- Fault injection: at line 88, change the condition of the then branch from "minutes<59" into "minutes≤59" in the increase_minutes procedure.
- Resulting failure: as shown in Fig. 44(c), the injected fault leads to the violation of precondition valid_minutes of the routine set_minutes when calling it from increase_minutes.
- Cause of the failure: incorrect implementation of routine body of increase_minutes.
- Proof time: 0.245 sec
- Possible fixes: 8 valid fixes out 356 candidate fixes
- fixing time: 3.18 minutes




| Class | Feature | Information | P... | Ti... |
|---|---|---|---|---|
| CLOCK_8 | invariant admissibility | Verification successful. | | 0.17 |
| CLOCK_8 | make (creator) | Verification successful. | | 0.02 |
| CLOCK_8 | set_hours | Verification successful. | | 0.00 |
| CLOCK_8 | set_minutes | Verification successful. | | |
| CLOCK_8 | set_seconds | Verification successful. | | 0.00 |
| CLOCK_8 | increase_hours | Verification successful. | | 0.04 |
| CLOCK_8 | increase_minutes | Precondition valid_minutes may be violated on call to {CLOCK_8}.set_minutes. | 100 | 0.00 |
| CLOCK_8 | increase_seconds | Verification successful. | | 0.01 |

(c)

```
1  increase_minutes_ID_50
2       -- Increase 'minutes' by one.
3     note
4       explicit: wrapping
5     do
6
7       if (minutes) > (0) then
8            increase_minutes
9            else
10               if minutes ≤ 59then
11        set_minutes (minutes + 1)
12          -- set_minutes (0)
13          -- increase_hours
14       else
15        set_minutes (0)
16        increase_hours
17       end
18           end
19    ensure
20        hours_increased: old minutes = 59implies hours = (old hours + 1) \\ 24
21      hours_unchanged: old minutes <59implies hours = old hours
22        minutes_increased: minutes = (old minutes + 1) \\ 60
```

```
      modify_model (["minutes", "hours"], Current)
    end


increase_minutes_ID_84
      -- Increase 'minutes' by one.
    note
      explicit: wrapping
    do

      if (minutes) ≥ (0) then
increase_minutes
else
if minutes ≤ 59then
        set_minutes (minutes + 1)
          -- set_minutes (0)
          -- increase_hours
      else
        set_minutes (0)
        increase_hours
      end
end
    ensure
        hours_increased: old minutes = 59implies hours = (old hours + 1) \\ 24
      hours_unchanged: old minutes < 59implies hours = old hours
        minutes_increased: minutes = (old minutes + 1) \\ 60
      modify_model (["minutes", "hours"], Current)
    end

increase_minutes_ID_186
      -- Increase 'minutes' by one.
    note
      explicit: wrapping
    do

      if minutes = 59then
increase_minutes
else
if minutes ≤ 59then
        set_minutes (minutes + 1)
          -- set_minutes (0)
          -- increase_hours
      else
        set_minutes (0)
        increase_hours
      end
end
    ensure
        hours_increased: old minutes = 59implies hours = (old hours + 1) \\ 24
      hours_unchanged: old minutes < 59implies hours = old hours
```

```
        minutes_increased: minutes = (old minutes + 1) \\ 60
      modify_model ("minutes", "hours"), Current)
    end

increase_minutes_ID_230
      -- Increase 'minutes' by one.
    note
      explicit: wrapping
    do

      if hours <minutes then
increase_minutes
else
if minutes ≤ 59then
        set_minutes (minutes + 1)
          -- set_minutes (0)
          -- increase_hours
      else
        set_minutes (0)
        increase_hours
      end
end
    ensure
        hours_increased: old minutes = 59implies hours = (old hours + 1) \\ 24
      hours_unchanged: old minutes <59implies hours = old hours
        minutes_increased: minutes = (old minutes + 1) \\ 60
      modify_model ("minutes", "hours"), Current)
    end

increase_minutes_ID_342
      -- Increase 'minutes' by one.
    note
      explicit: wrapping
require
not ((minutes) > (0))
do

      if minutes ≤ 59then
        set_minutes (minutes + 1)
          -- set_minutes (0)
          -- increase_hours
      else
        set_minutes (0)
        increase_hours
      end
    ensure
        hours_increased: old minutes = 59implies hours = (old hours + 1) \\ 24
      hours_unchanged: old minutes <59implies hours = old hours
        minutes_increased: minutes = (old minutes + 1) \\ 60
      modify_model ("minutes", "hours"), Current)
```

```
    end


increase_minutes_ID_350
      -- Increase ‘minutes’ by one.
    note
      explicit: wrapping
require
not (minutes = 59)
do

      if minutes ≤ 59then
        set_minutes (minutes + 1)
          -- set_minutes (0)
          -- increase_hours
      else
        set_minutes (0)
        increase_hours
      end
    ensure
        hours_increased: old minutes = 59implies hours = (old hours + 1) \\ 24
      hours_unchanged: old minutes <59implies hours = old hours
        minutes_increased: minutes = (old minutes + 1) \\ 60
      modify_model (["minutes", "hours"], Current)
    end


increase_minutes_ID_352
      -- Increase ‘minutes’ by one.
    note
      explicit: wrapping
        require
            not (hours <minutes)
        do

      if minutes ≤ 59then
        set_minutes (minutes + 1)
          -- set_minutes (0)
          -- increase_hours
      else
        set_minutes (0)
        increase_hours
      end
    ensure
        hours_increased: old minutes = 59implies hours = (old hours + 1) \\ 24
      hours_unchanged: old minutes <59implies hours = old hours
        minutes_increased: minutes = (old minutes + 1) \\ 60
      modify_model (["minutes", "hours"], Current)
    end
```

### 1.3 ARITHMETIC

```
note
  description: "Implementation of arithmetic operations based on increment."
class
  ARITHMETIC_1
create
  make
feature -- Initialization
  make
    do
    end
feature -- Addition
  add (a, b: INTEGER): INTEGER
      -- Add two numbers by repeated increment.
      -- Iterative version.
    require
      a_in_range: a ≥ −100 and a ≤ 100
      b_in_range: b ≥ −100 and b ≤ 100
    local
      i: INTEGER
    do
      if b ≥ 0then
        from
          Result := a
          i := b
        invariant
          Result = a + (b − i)
          0 ≤ i and i ≤ b
        until
          i = 0
        loop
          Result := Result + 1
          i := i − 1
        variant
          i
        end
      else
        from
          Result := a
          i := b
        invariant
          Result = a + (b − i)
          b ≤ i and i ≤ 0
        until
          i = 0
        loop
          Result := Result − 1
          i := i + 1
        variant
```

```
          - i
        end
      end
    ensure
      result_correct: Result = a + b
    end
  add_recursive (a, b: INTEGER): INTEGER
      -- Add two numbers by repeated increment.
      -- Recursive version.
    require
      decreases (if b <0then - b else b end)
      a_in_range: a ≥ -100 and a ≤ 100
      b_in_range: b ≥ -100 and b ≤ 100
    do
      if b = 0then
        Result := a
      elseif b > 0then
        Result := add_recursive (a, b - 1) + 1
      else
        Result := add_recursive (a, b + 1) - 1
      end
    ensure
      result_correct: Result = a + b
    end
feature -- Multiplication
  multiply (a, b: INTEGER): INTEGER
      -- Multiply two numbers by repeated addition.
      -- Iterative version.
    require
      b_not_negative: b ≥ 0
      a_in_range: a ≥ -10 and a ≤ 10
      b_in_range: b ≥ -10 and b ≤ 10
    local
      i: INTEGER
    do
      if a = 0or b = 0then
        Result := 0
      else
        from
          Result := a
          i := b
        invariant
          Result = a * (b - i + 1)
          1 ≤ i and i ≤ b
        until
          i = 1
        loop
          Result := add (Result, a)
          i := i - 1
        variant
```

```
        i
      end
    end
  ensure
    result_correct: Result = a * b
  end
 multiply_recursive (a, b: INTEGER): INTEGER
    -- Multiply two numbers by repeated addition.
    -- Recursive version.
  require
    b_not_negative: b ≥ 0
    a_in_range: a ≥ −10 and a ≤ 10
    b_in_range: b ≥ −10 and b ≤ 10
  do
    -- if a = 0or b = 0then
    -- Result := 0
    --else
      if b = 1then
        Result := a
      else
        Result := add_recursive (a, multiply (a, b − 1))
      end
    --end
  ensure
    result_correct: Result = a * b
  end
feature -- Division
-- divide (n, m: INTEGER): TUPLE [quotient, remainder: INTEGER]
--  -- Integer division of ‘n’ divided by ‘m’.
--  -- Iterative version.
-- require
--  n_not_negative: n ≥ 0
--  m_positive: m > 0
--  n_in_range: n ≤ 100
--  m_in_range: m ≤ 100
-- local
--  q, r: INTEGER
-- do
--  from
--   r := n
--   q := 0
--  invariant
--   0 ≤ r
--   n = m * q + r
--  until
--   r < m
--  loop
--   r := add (r, - m)
--   q := q + 1
--  variant
```

```
--   r
--   end
--   Result := [q, r]
--  ensure
--   n = m * Result.quotient + Result.remainder
--  end
  divide_recursive (n, m: INTEGER): TUPLE [quotient, remainder: INTEGER]
      -- Integer division of ‘n’ divided by ‘m’.
      -- Recursive version.
    require
      n_not_negative: n ≥ 0
      m_positive: m > 0
      n_in_range: n ≤ 100
      m_in_range: m ≤ 100
    local
      q, r: INTEGER
      res: TUPLE [quotient, remainder: INTEGER]
    do
      if n < m then
        Result := [0, n]
      else
        res := divide_recursive (add_recursive (n, − m), m)
        Result := [res.quotient + 1, res.remainder]
      end
    ensure
      n = m * Result.quotient + Result.remainder
    end
end
```

### ARITHMETIC_1

- 5 fixes out of 22 candidate fixes
- fixing time: 1.1 minutes

```
multiply_recursive_ID_1 (a, b: INTEGER): INTEGER
      -- Multiply two numbers by repeated addition.
      -- Recursive version.
    require
      b_not_negative: b ≥ 0
      a_in_range: a ≥ −10 and a ≤ 10
      b_in_range: b ≥ −10 and b ≤ 10
    do

      -- if a = 0or b = 0then
      -- Result := 0
      --else
        if not((b) = (0)) then
if b = 1then
          Result := a
```

```
        else
          Result := add_recursive (a, multiply (a, b - 1))
        end
end
      --end
    ensure
      result_correct: Result = a * b
    end

multiply_recursive_ID_9 (a, b: INTEGER): INTEGER
      -- Multiply two numbers by repeated addition.
      -- Recursive version.
    require
      b_not_negative: b >= 0
      a_in_range: a >= -10 and a <= 10
      b_in_range: b >= -10 and b <= 10
    do

      -- if a = 0or b = 0then
      -- Result := 0
      --else
        if not((b) <= (0)) then
if b = 1then
          Result := a
        else
          Result := add_recursive (a, multiply (a, b - 1))
        end
end
      --end
    ensure
      result_correct: Result = a * b
    end


multiply_recursive_ID_11 (a, b: INTEGER): INTEGER
      -- Multiply two numbers by repeated addition.
      -- Recursive version.
    require
      b_not_negative: b >= 0
      a_in_range: a >= -10 and a <= 10
      b_in_range: b >= -10 and b <= 10
    do

      -- if a = 0or b = 0then
      -- Result := 0
      --else
        if not(b = 0) then
if b = 1then
          Result := a
        else
```

```
          Result := add_recursive (a, multiply (a, b - 1))
        end
end
      --end
    ensure
      result_correct: Result = a * b
    end


multiply_recursive_ID_12 (a, b: INTEGER): INTEGER
      -- Multiply two numbers by repeated addition.
      -- Recursive version.
    require
      b_not_negative: b >= 0
      a_in_range: a >= -10 and a <= 10
      b_in_range: b >= -10 and b <= 10
not ((b) = (0))
    do

      -- if a = 0or b = 0then
      -- Result := 0
      --else
        if b = 1then
          Result := a
        else
          Result := add_recursive (a, multiply (a, b - 1))
        end
      --end
    ensure
      result_correct: Result = a * b
    end


multiply_recursive_ID_20 (a, b: INTEGER): INTEGER
      -- Multiply two numbers by repeated addition.
      -- Recursive version.
    require
      b_not_negative: b >= 0
      a_in_range: a >= -10 and a <= 10
      b_in_range: b >= -10 and b <= 10
not ((b) <= (0))
    do

      -- if a = 0or b = 0then
      -- Result := 0
      --else
        if b = 1then
          Result := a
        else
          Result := add_recursive (a, multiply (a, b - 1))
```

```
        end
      --end
    ensure
      result_correct: Result = a * b
    end


multiply_recursive_ID_22 (a, b: INTEGER): INTEGER
      -- Multiply two numbers by repeated addition.
      -- Recursive version.
    require
      b_not_negative: b ≥ 0
      a_in_range: a ≥ −10 and a ≤ 10
      b_in_range: b ≥ −10 and b ≤ 10
not (b = 0)
    do

      -- if a = 0or b = 0then
      -- Result := 0
      --else
        if b = 1then
          Result := a
        else
          Result := add_recursive (a, multiply (a, b − 1))
        end
      --end
    ensure
      result_correct: Result = a * b
    end
```

**ARITHMETIC_2** No valid fixes

**ARITHMETIC_3** Compilation error: ce extraction does not support the tuple type

**ARITHMETIC_4**

- 5 valid fixes out of 30 candidate fixes
- fixing time: 1.02 minutes

```
add_recursive_ID_0 (a, b: INTEGER): INTEGER
      -- Add two numbers by repeated increment.
      -- Recursive version.
    require
      decreases (if b <0then − b else b end)
      a_in_range: a ≥ −100 and a ≤ 100
      b_in_range: b ≥ −100 and b ≤ 100
```

```
    do

      -- if b = 0 then
      -- Result := a
      if b > 0 then
        Result := add_recursive (a, b - 1) + 1
      else
        Result := add_recursive (a, b + 1) - 1
      end
    ensure
      result_correct: Result = a + b
    end

add_recursive_ID_16 (a, b: INTEGER): INTEGER
      -- Add two numbers by repeated increment.
      -- Recursive version.
    require
      decreases (if b < 0 then - b else b end)
      a_in_range: a ≥ -100 and a ≤ 100
      b_in_range: b ≥ -100 and b ≤ 100
not ((a) = (0))
    do

      -- if b = 0 then
      -- Result := a
      if b > 0 then
        Result := add_recursive (a, b - 1) + 1
      else
        Result := add_recursive (a, b + 1) - 1
      end
    ensure
      result_correct: Result = a + b
    end

add_recursive_ID_18 (a, b: INTEGER): INTEGER
      -- Add two numbers by repeated increment.
      -- Recursive version.
    require
      decreases (if b < 0 then - b else b end)
      a_in_range: a ≥ -100 and a ≤ 100
      b_in_range: b ≥ -100 and b ≤ 100
not ((b) = (0))
    do

      -- if b = 0 then
      -- Result := a
      if b > 0 then
        Result := add_recursive (a, b - 1) + 1
      else
        Result := add_recursive (a, b + 1) - 1
```

```
      end
    ensure
      result_correct: Result = a + b
    end

add_recursive_ID_20 (a, b: INTEGER): INTEGER
      -- Add two numbers by repeated increment.
      -- Recursive version.
    require
      decreases (if b <0then − b else b end)
      a_in_range: a ≥ −100 and a ≤ 100
      b_in_range: b ≥ −100 and b ≤ 100
not ((b) > (0))
    do

      -- if b = 0then
      -- Result := a
      if b > 0then
        Result := add_recursive (a, b − 1) + 1
      else
        Result := add_recursive (a, b + 1) − 1
      end
    ensure
      result_correct: Result = a + b
    end

add_recursive_ID_22 (a, b: INTEGER): INTEGER
      -- Add two numbers by repeated increment.
      -- Recursive version.
    require
      decreases (if b <0then − b else b end)
      a_in_range: a ≥ −100 and a ≤ 100
      b_in_range: b ≥ −100 and b ≤ 100
not ((b) ≥ (0))
    do

      -- if b = 0then
      -- Result := a
      if b > 0then
        Result := add_recursive (a, b − 1) + 1
      else
        Result := add_recursive (a, b + 1) − 1
      end
    ensure
      result_correct: Result = a + b
    end

add_recursive_ID_24 (a, b: INTEGER): INTEGER
      -- Add two numbers by repeated increment.
      -- Recursive version.
```

```
    require
      decreases (if b <0then − b else b end)
      a_in_range: a ≥ −100 and a ≤ 100
      b_in_range: b ≥ −100 and b ≤ 100
not ((b) <(0))
    do

      -- if b = 0then
      -- Result := a
      if b > 0then
        Result := add_recursive (a, b − 1) + 1
      else
        Result := add_recursive (a, b + 1) − 1
      end
    ensure
      result_correct: Result = a + b
    end

add_recursive_ID_26 (a, b: INTEGER): INTEGER
      -- Add two numbers by repeated increment.
      -- Recursive version.
    require
      decreases (if b <0then − b else b end)
      a_in_range: a ≥ −100 and a ≤ 100
      b_in_range: b ≥ −100 and b ≤ 100
not ((b) ≤ (0))
    do

      -- if b = 0then
      -- Result := a
      if b > 0then
        Result := add_recursive (a, b − 1) + 1
      else
        Result := add_recursive (a, b + 1) − 1
      end
    ensure
      result_correct: Result = a + b
    end

add_recursive_ID_28 (a, b: INTEGER): INTEGER
      -- Add two numbers by repeated increment.
      -- Recursive version.
    require
      decreases (if b <0then − b else b end)
      a_in_range: a ≥ −100 and a ≤ 100
      b_in_range: b ≥ −100 and b ≤ 100
not (b = 0)
    do

      -- if b = 0then
```

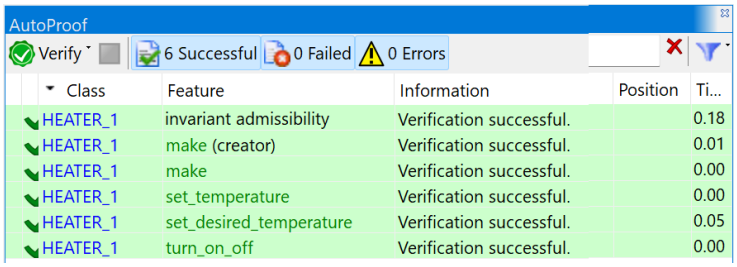
```
      -- Result := a
      if b > 0then
        Result := add_recursive (a, b - 1) + 1
      else
        Result := add_recursive (a, b + 1) - 1
      end
    ensure
      result_correct: Result = a + b
    end

add_recursive_ID_30 (a, b: INTEGER): INTEGER
      -- Add two numbers by repeated increment.
      -- Recursive version.
    require
      decreases (if b <0then - b else b end)
      a_in_range: a ≥ -100 and a ≤ 100
      b_in_range: b ≥ -100 and b ≤ 100
not (a ≠ b)
    do

      -- if b = 0then
      -- Result := a
      if b > 0then
        Result := add_recursive (a, b - 1) + 1
      else
        Result := add_recursive (a, b + 1) - 1
      end
    ensure
      result_correct: Result = a + b
    end
```

### 1.4 HEATER

HEATER class, as shown below, implements a heater, which is automatically turned on/off based on the relation between the current temperature and the desired temperature. Fig. 44 displays the verification result of HEATER, which indicates that the implementation is correct with respect to its specification. In the experiment, 4 variants of faulty HEATER class are derived from this correct version, which will be discussed below.

```
class
    HEATER
create
    make
feature
    make
            -- By default, desired temperature is 20degree, deviation is 2 and
                heater is off
        do
            desired_temp := 20
            is_on := False
        ensure
            default_condition: desired_temp = 20 and is_on = False
        end
feature
    temperature: INTEGER
            -- Current temperature
    desired_temp: INTEGER
            -- Temperature defined by the user
    is_on: BOOLEAN
            -- Is heater turned on?
    Deviation: INTEGER = 2
            -- Deviation for turning on/off the heater
feature
    set_temperature (a_value: INTEGER)
            -- Set the ‘temperature’ to ‘a_value’
        do
            temperature := a_value
        end
    set_desired_temperature (value: INTEGER)
            -- Set the ‘desired_temp’ to ‘value’
        require
            valid_value: value ≥ 10 and value ≤ 100
        do
            desired_temp := value
        ensure
            temperature_set: desired_temp = value
        end
```

```
    turn_on_off
            -- Turn on or turn off the heater automatically based on the current temperature
        require
            desired_temp_valid: desired_temp ≥ 10 and desired_temp ≤ 100
        do
            if is_on then
                if temperature > desired_temp + deviation then
                    is_on := False
                end
            else
                if temperature < desired_temp − deviation then
                    is_on := True
                end
            end
        ensure
            heater_is_turned_off: old (is_on and temperature > desired_temp + deviation) implies (not is_on)
            heater_remains_on: old (is_on and temperature ≤ desired_temp + deviation) implies is_on
            heater_is_turned_on: old (not is_on and temperature < desired_temp − deviation) implies is_on
            heater_remains_off: old (not is_on and temperature ≥ desired_temp − deviation) implies (not is_on)
        end
invariant
    desired_temp_in_bound: desired_temp ≥ 10 and desired_temp ≤ 100
end
```



AutoProof

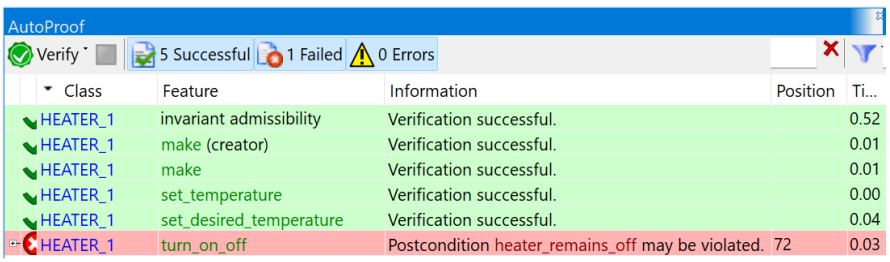
Verify | 6 Successful | 0 Failed | 0 Errors

| Class | Feature | Information | Position | Ti... |
|---|---|---|---|---|
| HEATER_1 | invariant admissibility | Verification successful. | | 0.18 |
| HEATER_1 | make (creator) | Verification successful. | | 0.01 |
| HEATER_1 | make | Verification successful. | | 0.00 |
| HEATER_1 | set_temperature | Verification successful. | | 0.00 |
| HEATER_1 | set_desired_temperature | Verification successful. | | 0.05 |
| HEATER_1 | turn_on_off | Verification successful. | | 0.00 |

**Fig. 44.** Proof result of HEATER in AutoProof

### Variant 1 of HEATER

- Fault injection: at line 59, change the condition of the then branch from "`temperature`< `desired_temp`−`deviation`" into "`temperature`≤ `desired_temp`−`deviation`".
- Resulting failure: as shown in Fig. 45(a), the injected faults results in the violation of the postcondition `heater_remains_off` of the procedure `turn_on_off`.
- Cause of the failure: incorrect implementation of the routine body of `turn_on_off`.
- Proof time: 0.615 sec
- Comment: the test is useful as its execution shows a specific path along which the same contract violation in the proof would be raised.
- Possible fixes: 24 valid fixes out 437 candidate fixes.
- fixing time: 5.28 minutes




| Class | Feature | Information | Position | Ti... |
|---|---|---|---|---|
| HEATER_1 | invariant admissibility | Verification successful. | | 0.52 |
| HEATER_1 | make (creator) | Verification successful. | | 0.01 |
| HEATER_1 | make | Verification successful. | | 0.01 |
| HEATER_1 | set_temperature | Verification successful. | | 0.00 |
| HEATER_1 | set_desired_temperature | Verification successful. | | 0.04 |
| HEATER_1 | turn_on_off | Postcondition heater_remains_off may be violated. | 72 | 0.03 |

(a)

```
1
2  turn_on_off_ID_367
3       -- Turn on or turn off the heater automatically based on the current
            temperature
4     require
5       desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
6     do
7
8       if is_on then
9         if temperature > desired_temp + deviation then
10          is_on := False
11        end
12      else
13        if temperature ≤ desired_temp − deviation then
14          is_on := True
15        end
16      end
17    ensure
18      heater_is_turned_off: old (is_on and temperature > desired_temp +
            deviation) implies (not is_on)
19      heater_remains_on: old (is_on and temperature ≤ desired_temp +
            deviation) implies is_on
20      heater_is_turned_on: old (not is_on and temperature <desired_temp −
            deviation) implies is_on
```

```
heater_remains_off:not (is_on)implies ( old (not is_on and temperature ≥ desired_temp − deviation) implies (not is_on))
    end

turn_on_off_ID_368
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
not (not (is_on))
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature ≤ desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp + deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp + deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp − deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp − deviation) implies (not is_on)
    end

turn_on_off_ID_372
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
not ((temperature) > (0))
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature ≤ desired_temp − deviation then
          is_on := True
        end
      end
    ensure
```

```
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp −
          deviation) implies (not is_on)
    end

turn_on_off_ID_373
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature ≤ desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
heater_remains_off:not ((temperature) > (0))implies ( old (not is_on and
     temperature ≥ desired_temp − deviation) implies (not is_on))
    end

turn_on_off_ID_374
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
not ((temperature) ≥ (0))
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
```

```
        if temperature ≤desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥desired_temp −
          deviation) implies (not is_on)
    end

turn_on_off_ID_375
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥10and desired_temp ≤100
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature ≤desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
heater_remains_off:not ((temperature) ≥(0))implies ( old (not is_on and
     temperature ≥desired_temp − deviation) implies (not is_on))
    end

turn_on_off_ID_383
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥10and desired_temp ≤100
    do

      if is_on then
```

```
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature ≤ desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
heater_remains_off:not ((desired_temp) > (0))implies ( old (not is_on and
    temperature ≥ desired_temp − deviation) implies (not is_on))
    end

turn_on_off_ID_385
      -- Turn on or turn off the heater automatically based on the current
          temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature ≤ desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
heater_remains_off:not ((desired_temp) ≥ (0))implies ( old (not is_on and
    temperature ≥ desired_temp − deviation) implies (not is_on))
    end

turn_on_off_ID_393
      -- Turn on or turn off the heater automatically based on the current
          temperature
    require
```

```
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature ≤ desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
heater_remains_off:not ((deviation) > (0))implies ( old (not is_on and
    temperature ≥ desired_temp − deviation) implies (not is_on))
    end

turn_on_off_ID_395
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature ≤ desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
heater_remains_off:not ((deviation) ≥ (0))implies ( old (not is_on and
    temperature ≥ desired_temp − deviation) implies (not is_on))
    end

```

```
turn_on_off_ID_406
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
not ((temperature) <(desired_temp))
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature ≤ desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp + deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp + deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp − deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp − deviation) implies (not is_on)
    end

turn_on_off_ID_407
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature ≤ desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp + deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp + deviation) implies is_on
```

```
      heater_is_turned_on: old (not is_on and temperature <desired_temp − deviation) implies is_on
heater_remains_off:not ((temperature) <(desired_temp))implies ( old (not is_on and temperature ≥ desired_temp − deviation) implies (not is_on))
    end

turn_on_off_ID_408
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
not ((temperature) ≤ (desired_temp))
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature ≤ desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp + deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp + deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp − deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp − deviation) implies (not is_on)
    end

turn_on_off_ID_409
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature ≤ desired_temp − deviation then
          is_on := True
        end
      end
```

```
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
heater_remains_off:not ((temperature) ≤ (desired_temp))implies ( old (not
     is_on and temperature ≥ desired_temp − deviation) implies (not is_on))
    end

turn_on_off_ID_412
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
not ((temperature) > (deviation))
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature ≤ desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp −
          deviation) implies (not is_on)
    end

turn_on_off_ID_413
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
```

```
      else
        if temperature ≤desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
heater_remains_off:not ((temperature) > (deviation))implies ( old (not is_on
     and temperature ≥desired_temp − deviation) implies (not is_on))
    end

turn_on_off_ID_414
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥10and desired_temp ≤100
not ((temperature) ≥(deviation))
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature ≤desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥desired_temp −
          deviation) implies (not is_on)
    end

turn_on_off_ID_415
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥10and desired_temp ≤100
    do
```

```
      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature ≤ desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp + deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp + deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp − deviation) implies is_on
heater_remains_off:not ((temperature) ≥ (deviation))implies ( old (not is_on and temperature ≥ desired_temp − deviation) implies (not is_on))
    end


turn_on_off_ID_423
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature ≤ desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp + deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp + deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp − deviation) implies is_on
heater_remains_off:not ((desired_temp) > (deviation))implies ( old (not is_on and temperature ≥ desired_temp − deviation) implies (not is_on))
    end
```

```
turn_on_off_ID_425
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature ≤ desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp + deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp + deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp − deviation) implies is_on
heater_remains_off:not ((desired_temp) ≥ (deviation))implies ( old (not is_on and temperature ≥ desired_temp − deviation) implies (not is_on))
    end


turn_on_off_ID_431
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature ≤ desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp + deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp + deviation) implies is_on
```

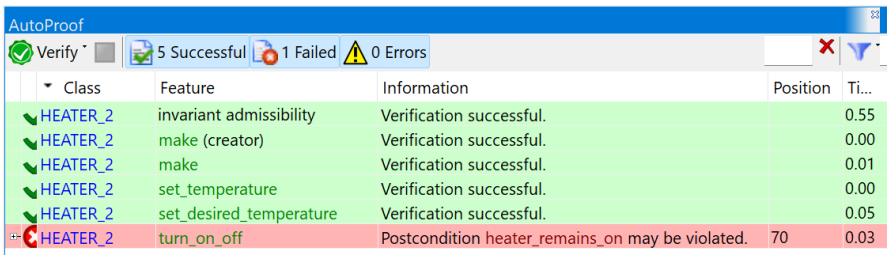
```
      heater_is_turned_on: old (not is_on and temperature <desired_temp − deviation) implies is_on
heater_remains_off:not (deviation = 2)implies ( old (not is_on and temperature ≥ desired_temp − deviation) implies (not is_on))
    end


turn_on_off_ID_432
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
not (temperature − desired_temp + 2= 0)
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature ≤ desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp + deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp + deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp − deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp − deviation) implies (not is_on)
    end


turn_on_off_ID_433
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature ≤ desired_temp − deviation then
          is_on := True
```

```
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
heater_remains_off:not (temperature − desired_temp + 2= 0)implies ( old (not
     is_on and temperature ≥ desired_temp − deviation) implies (not is_on))
    end


turn_on_off_ID_434
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
not (temperature > deviation)
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature ≤ desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_435
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

```

```
      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature ≤ desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
heater_remains_off:not (temperature > deviation)implies ( old (not is_on and
     temperature ≥ desired_temp − deviation) implies (not is_on))
    end


turn_on_off_ID_437
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature ≤ desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
heater_remains_off:not (desired_temp > deviation)implies ( old (not is_on and
     temperature ≥ desired_temp − deviation) implies (not is_on))
    end
```

**Variant 2 of HEATER**

- Fault injection: at line 55, in the body of `turn_on_off`, change the condition of the then branch from "`temperature` > `desired_temp+deviation`" into "`temperature` ≥ `desired_temp+deviation`".
- Resulting failure: as shown in Fig. 45(b), the injected fault results in the violation of postcondition `heater_remains_on` of `turn_on_off` procedure.
- Cause of the failure: incorrect implementation of the routine body of `turn_on_off`.
- Proof time: 0.642 sec
- Comment: similar to Variant 1 of HEATER, the test is useful as its execution shows a specific path along which the same contract violation in the proof would be raised.
- Possible fixes: 25 valid fixes out of 437
- fixing time: 4.05 minutes




| Class | Feature | Information | Position | Ti... |
|---|---|---|---|---|
| HEATER_2 | invariant admissibility | Verification successful. | | 0.55 |
| HEATER_2 | make (creator) | Verification successful. | | 0.00 |
| HEATER_2 | make | Verification successful. | | 0.01 |
| HEATER_2 | set_temperature | Verification successful. | | 0.00 |
| HEATER_2 | set_desired_temperature | Verification successful. | | 0.05 |
| HEATER_2 | turn_on_off | Postcondition heater_remains_on may be violated. | 70 | 0.03 |

(b)

```
1  turn_on_off_ID_366
2       -- Turn on or turn off the heater automatically based on the current
            temperature
3     require
4       desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
5  not (is_on)
6     do
7
8       if is_on then
9         if temperature ≥ desired_temp + deviation then
10          is_on := False
11        end
12      else
13        if temperature <desired_temp − deviation then
14          is_on := True
15        end
16      end
17    ensure
18      heater_is_turned_off: old (is_on and temperature > desired_temp +
            deviation) implies (not is_on)
```

```
      heater_remains_on: old (is_on and temperature ≤desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥desired_temp −
          deviation) implies (not is_on)
    end

turn_on_off_ID_369
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥10and desired_temp ≤100
    do

      if is_on then
        if temperature ≥desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
heater_remains_on:not (not (is_on))implies ( old (is_on and temperature ≤
     desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_372
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥10and desired_temp ≤100
not ((temperature) > (0))
    do

      if is_on then
        if temperature ≥desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
```

```
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥desired_temp −
          deviation) implies (not is_on)
    end

turn_on_off_ID_373
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥10and desired_temp ≤100
    do

      if is_on then
        if temperature ≥desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
heater_remains_on:not ((temperature) > (0))implies ( old (is_on and
     temperature ≤desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_374
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥10and desired_temp ≤100
not ((temperature) ≥(0))
    do

```

```
      if is_on then
        if temperature ≥desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_375
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥10and desired_temp ≤100
    do

      if is_on then
        if temperature ≥desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
heater_remains_on:not ((temperature) ≥(0))implies ( old (is_on and
     temperature ≤desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_383
```

```
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature ≥ desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp + deviation) implies (not is_on)
heater_remains_on:not ((desired_temp) > (0))implies ( old (is_on and temperature ≤ desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp − deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp − deviation) implies (not is_on)
    end


turn_on_off_ID_385
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature ≥ desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp + deviation) implies (not is_on)
heater_remains_on:not ((desired_temp) ≥ (0))implies ( old (is_on and temperature ≤ desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp − deviation) implies is_on
```

```
      heater_remains_off: old (not is_on and temperature ≥ desired_temp − deviation) implies (not is_on)
    end


turn_on_off_ID_393
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature ≥ desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp + deviation) implies (not is_on)
heater_remains_on:not ((deviation) > (0))implies ( old (is_on and temperature ≤ desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp − deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp − deviation) implies (not is_on)
    end


turn_on_off_ID_395
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature ≥ desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
```

```
      heater_is_turned_off: old (is_on and temperature > desired_temp + deviation) implies (not is_on)
heater_remains_on:not ((deviation) ≥ (0))implies ( old (is_on and temperature ≤ desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp − deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp − deviation) implies (not is_on)
    end


turn_on_off_ID_402
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
not ((temperature) > (desired_temp))
    do

      if is_on then
        if temperature ≥ desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp + deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp + deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp − deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp − deviation) implies (not is_on)
    end


turn_on_off_ID_403
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature ≥ desired_temp + deviation then
          is_on := False
```

```
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
heater_remains_on:not ((temperature) > (desired_temp))implies ( old (is_on
     and temperature ≤ desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_404
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
not ((temperature) ≥ (desired_temp))
    do

      if is_on then
        if temperature ≥ desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_405
      -- Turn on or turn off the heater automatically based on the current
           temperature
```

```
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature ≥ desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
heater_remains_on:not ((temperature) ≥ (desired_temp))implies ( old (is_on and
     temperature ≤ desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_412
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
not ((temperature) > (deviation))
    do

      if is_on then
        if temperature ≥ desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
```

```
      heater_remains_off: old (not is_on and temperature ≥ desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_413
      -- Turn on or turn off the heater automatically based on the current
          temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature ≥ desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
heater_remains_on:not ((temperature) > (deviation))implies ( old (is_on and
    temperature ≤ desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_414
      -- Turn on or turn off the heater automatically based on the current
          temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
not ((temperature) ≥ (deviation))
    do

      if is_on then
        if temperature ≥ desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
```

```
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_415
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥10and desired_temp ≤100
    do

      if is_on then
        if temperature ≥desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
heater_remains_on:not ((temperature) ≥(deviation))implies ( old (is_on and
     temperature ≤desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_423
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥10and desired_temp ≤100
    do

      if is_on then
        if temperature ≥desired_temp + deviation then
          is_on := False
```

```
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
heater_remains_on:not ((desired_temp) > (deviation))implies ( old (is_on and
     temperature ≤ desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_425
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature ≥ desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
heater_remains_on:not ((desired_temp) ≥ (deviation))implies ( old (is_on and
     temperature ≤ desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_431
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
```

```
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature ≥ desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
heater_remains_on:not (deviation = 2)implies ( old (is_on and temperature ≤
     desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_432
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
not (temperature − desired_temp − 2= 0)
    do

      if is_on then
        if temperature ≥ desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp −
          deviation) implies (not is_on)
```

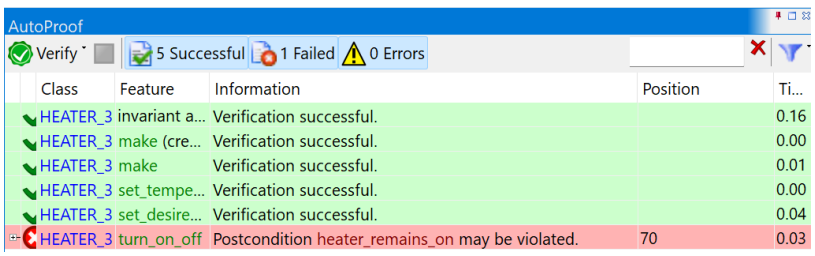
```
    end


turn_on_off_ID_433
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature ≥ desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp + deviation) implies (not is_on)
heater_remains_on:not (temperature − desired_temp − 2= 0)implies ( old ( is_on and temperature ≤ desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp − deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp − deviation) implies (not is_on)
    end


turn_on_off_ID_434
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
not (temperature > deviation)
    do

      if is_on then
        if temperature ≥ desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
```

```
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_435
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature ≥ desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
heater_remains_on:not (temperature > deviation)implies ( old (is_on and
     temperature ≤ desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_437
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature ≥ desired_temp + deviation then
          is_on := False
        end
```

```
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
heater_remains_on:not (desired_temp > deviation)implies ( old (is_on and
    temperature ≤ desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp −
          deviation) implies (not is_on)
    end
```

**Variant 3 of HEATER**

- Fault injection: at line 55, in the body of `turn_on_off`, change the condition of the then branch from "`temperature> desired_temp+deviation`" into "`temperature> desired_temp−deviation`".
- Resulting failure: as shown in Fig. 45(c), the injected fault results in the violation of postcondition `heater_remains_on` of the procedure `turn_on_off`.
- Cause of the failure: incorrect implementation of the routine body of `turn_on_off`.
- Proof time: 0.250 sec
- Comment: similar to Variant 1 and 2 of HEATER, the test is useful as its execution shows a specific path along which the same contract violation in the proof would be raised.
- Possible fixes: 19 valid fixes out of 425 candidate fixes.
- fixing time: 4.17 minutes




(c)

```
1  turn_on_off_ID_366
2       -- Turn on or turn off the heater automatically based on the current
            temperature
3     require
4       desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
5  not (is_on)
6     do
7
8       if is_on then
9         if temperature > desired_temp − deviation then
10          is_on := False
11        end
12      else
13        if temperature <desired_temp − deviation then
14          is_on := True
15        end
16      end
17    ensure
18      heater_is_turned_off: old (is_on and temperature > desired_temp +
            deviation) implies (not is_on)
```

```
      heater_remains_on: old (is_on and temperature ≤desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥desired_temp −
          deviation) implies (not is_on)
    end

turn_on_off_ID_369
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥10and desired_temp ≤100
    do

      if is_on then
        if temperature > desired_temp − deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
heater_remains_on:not (not (is_on))implies ( old (is_on and temperature ≤
     desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_372
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥10and desired_temp ≤100
not ((temperature) > (0))
    do

      if is_on then
        if temperature > desired_temp − deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
```

```
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_373
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥10and desired_temp ≤100
    do

      if is_on then
        if temperature > desired_temp − deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
heater_remains_on:not ((temperature) > (0))implies ( old (is_on and
     temperature ≤desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_374
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥10and desired_temp ≤100
not ((temperature) ≥(0))
    do
```

```
      if is_on then
        if temperature > desired_temp - deviation then
          is_on := False
        end
      else
        if temperature <desired_temp - deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp -
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥desired_temp -
          deviation) implies (not is_on)
    end


turn_on_off_ID_375
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥10and desired_temp ≤100
    do

      if is_on then
        if temperature > desired_temp - deviation then
          is_on := False
        end
      else
        if temperature <desired_temp - deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
heater_remains_on:not ((temperature) ≥(0))implies ( old (is_on and
     temperature ≤desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp -
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥desired_temp -
          deviation) implies (not is_on)
    end
```

```
turn_on_off_ID_383
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp − deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp + deviation) implies (not is_on)
heater_remains_on:not ((desired_temp) > (0))implies ( old (is_on and temperature ≤ desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp − deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp − deviation) implies (not is_on)
    end


turn_on_off_ID_385
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp − deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp + deviation) implies (not is_on)
heater_remains_on:not ((desired_temp) ≥ (0))implies ( old (is_on and temperature ≤ desired_temp + deviation) implies is_on)
```

```
      heater_is_turned_on: old (not is_on and temperature <desired_temp − deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp − deviation) implies (not is_on)
    end


turn_on_off_ID_393
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp − deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp + deviation) implies (not is_on)
heater_remains_on:not ((deviation) > (0))implies ( old (is_on and temperature ≤ desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp − deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp − deviation) implies (not is_on)
    end


turn_on_off_ID_395
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp − deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
```

```
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
heater_remains_on:not ((deviation) ≥ (0))implies ( old (is_on and temperature
     ≤ desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_402
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
not ((temperature) > (deviation))
    do

      if is_on then
        if temperature > desired_temp − deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_403
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
```

```
        if temperature > desired_temp − deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
heater_remains_on:not ((temperature) > (deviation))implies ( old (is_on and
     temperature ≤desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_404
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥10and desired_temp ≤100
not ((temperature) ≥(deviation))
    do

      if is_on then
        if temperature > desired_temp − deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_405
```

```
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp − deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp + deviation) implies (not is_on)
heater_remains_on:not ((temperature) ≥ (deviation))implies ( old (is_on and temperature ≤ desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp − deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp − deviation) implies (not is_on)
    end


turn_on_off_ID_413
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp − deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp + deviation) implies (not is_on)
heater_remains_on:not ((desired_temp) > (deviation))implies ( old (is_on and temperature ≤ desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp − deviation) implies is_on
```

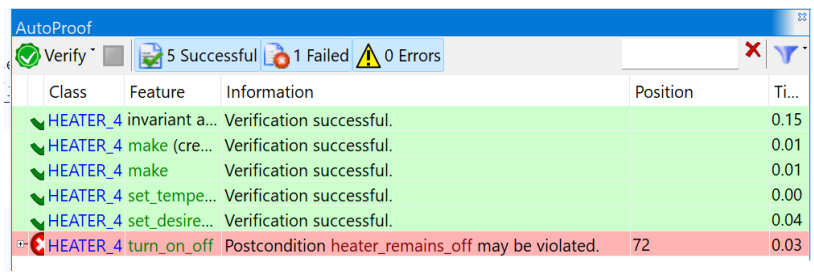
```
      heater_remains_off: old (not is_on and temperature ≥ desired_temp − deviation) implies (not is_on)
    end


turn_on_off_ID_415
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp − deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp + deviation) implies (not is_on)
heater_remains_on:not ((desired_temp) ≥ (deviation))implies ( old (is_on and temperature ≤ desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp − deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp − deviation) implies (not is_on)
    end


turn_on_off_ID_421
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp − deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
```

```
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
heater_remains_on:not (deviation = 2)implies ( old (is_on and temperature ≤
     desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_422
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
not (temperature > deviation)
    do

      if is_on then
        if temperature > desired_temp − deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_423
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp − deviation then
          is_on := False
```

```
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
heater_remains_on:not (temperature > deviation)implies ( old (is_on and
    temperature ≤ desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp −
          deviation) implies (not is_on)
    end


turn_on_off_ID_425
      -- Turn on or turn off the heater automatically based on the current
          temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp − deviation then
          is_on := False
        end
      else
        if temperature <desired_temp − deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
heater_remains_on:not (desired_temp > deviation)implies ( old (is_on and
    temperature ≤ desired_temp + deviation) implies is_on)
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp −
          deviation) implies (not is_on)
    end
```

**Variant 4 of** `HEATER`

- Fault injection: at line 59, change the condition of the then branch from "`temperature< desired_temp−deviation`" into "`temperature< desired_temp+deviation`".
- Resulting failure: as shown in Fig. 45(d), the injected fault leads to the violation of the postcondition `heater_remains_off` in the procedure `turn_on_off`.
- Cause of the failure: incorrect implementation of the routine body of `turn_on_off`.
- Proof time: 0.246 sec
- Comment: similar to the previous variants, the test is useful as its execution shows a specific path along which the same contract violation in the proof would be raised.
- Possible fixes: 17 valid fixes out of 429 candidate fixes
- fixing time: 4.65 minutes




(d)

```
1
2  turn_on_off_ID_367
3       -- Turn on or turn off the heater automatically based on the current
            temperature
4     require
5       desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
6     do
7
8       if is_on then
9         if temperature > desired_temp + deviation then
10          is_on := False
11        end
12      else
13        if temperature <desired_temp + deviation then
14          is_on := True
15        end
16      end
17    ensure
18      heater_is_turned_off: old (is_on and temperature > desired_temp +
            deviation) implies (not is_on)
```

```
      heater_remains_on: old (is_on and temperature ≤desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
heater_remains_off:not (is_on)implies ( old (not is_on and temperature ≥
    desired_temp − deviation) implies (not is_on))
    end

turn_on_off_ID_368
      -- Turn on or turn off the heater automatically based on the current
          temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
not (not (is_on))
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp + deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥desired_temp −
          deviation) implies (not is_on)
    end

turn_on_off_ID_372
      -- Turn on or turn off the heater automatically based on the current
          temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
not ((temperature) > (0))
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp + deviation then
```

```
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥desired_temp −
          deviation) implies (not is_on)
    end

turn_on_off_ID_373
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥10and desired_temp ≤100
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp + deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
heater_remains_off:not ((temperature) > (0))implies ( old (not is_on and
     temperature ≥desired_temp − deviation) implies (not is_on))
    end

turn_on_off_ID_374
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥10and desired_temp ≤100
not ((temperature) ≥(0))
    do

      if is_on then
```

```
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp + deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp −
          deviation) implies (not is_on)
    end

turn_on_off_ID_375
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp + deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
heater_remains_off:not ((temperature) ≥ (0))implies ( old (not is_on and
     temperature ≥ desired_temp − deviation) implies (not is_on))
    end

turn_on_off_ID_383
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
```

```
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp + deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
heater_remains_off:not ((desired_temp) > (0))implies ( old (not is_on and
    temperature ≥ desired_temp − deviation) implies (not is_on))
    end

turn_on_off_ID_385
      -- Turn on or turn off the heater automatically based on the current
          temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp + deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
heater_remains_off:not ((desired_temp) ≥ (0))implies ( old (not is_on and
    temperature ≥ desired_temp − deviation) implies (not is_on))
    end

```

```
turn_on_off_ID_393
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp + deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp + deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp + deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp − deviation) implies is_on
heater_remains_off:not ((deviation) > (0))implies ( old (not is_on and temperature ≥ desired_temp − deviation) implies (not is_on))
    end

turn_on_off_ID_395
      -- Turn on or turn off the heater automatically based on the current temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp + deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp + deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp + deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp − deviation) implies is_on
```

```
heater_remains_off:not ((deviation) ≥ (0))implies ( old (not is_on and
    temperature ≥ desired_temp − deviation) implies (not is_on))
    end

turn_on_off_ID_404
      -- Turn on or turn off the heater automatically based on the current
          temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
not ((temperature) > (deviation))
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp + deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp −
          deviation) implies (not is_on)
    end

turn_on_off_ID_405
      -- Turn on or turn off the heater automatically based on the current
          temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp + deviation then
          is_on := True
        end
      end
    ensure
```

```
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
heater_remains_off:not ((temperature) > (deviation))implies ( old (not is_on
    and temperature ≥ desired_temp − deviation) implies (not is_on))
    end

turn_on_off_ID_406
      -- Turn on or turn off the heater automatically based on the current
          temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
not ((temperature) ≥ (deviation))
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp + deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp −
          deviation) implies (not is_on)
    end

turn_on_off_ID_407
      -- Turn on or turn off the heater automatically based on the current
          temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
```

```
        if temperature <desired_temp + deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
heater_remains_off:not ((temperature) ≥ (deviation))implies ( old (not is_on
     and temperature ≥ desired_temp − deviation) implies (not is_on))
    end

turn_on_off_ID_415
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp + deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
heater_remains_off:not ((desired_temp) > (deviation))implies ( old (not is_on
     and temperature ≥ desired_temp − deviation) implies (not is_on))
    end

turn_on_off_ID_417
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
```

```
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp + deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
heater_remains_off:not ((desired_temp) ≥ (deviation))implies ( old (not is_on
     and temperature ≥ desired_temp − deviation) implies (not is_on))
    end

turn_on_off_ID_423
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp + deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
heater_remains_off:not (deviation = 2)implies ( old (not is_on and
     temperature ≥ desired_temp − deviation) implies (not is_on))
    end

turn_on_off_ID_426
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
```

```
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
not (temperature > deviation)
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp + deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
      heater_remains_off: old (not is_on and temperature ≥ desired_temp −
          deviation) implies (not is_on)
    end

turn_on_off_ID_427
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp + deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
heater_remains_off:not (temperature > deviation)implies ( old (not is_on and
     temperature ≥ desired_temp − deviation) implies (not is_on))
    end
```

```
turn_on_off_ID_429
      -- Turn on or turn off the heater automatically based on the current
           temperature
    require
      desired_temp_valid: desired_temp ≥ 10and desired_temp ≤ 100
    do

      if is_on then
        if temperature > desired_temp + deviation then
          is_on := False
        end
      else
        if temperature <desired_temp + deviation then
          is_on := True
        end
      end
    ensure
      heater_is_turned_off: old (is_on and temperature > desired_temp +
          deviation) implies (not is_on)
      heater_remains_on: old (is_on and temperature ≤ desired_temp +
          deviation) implies is_on
      heater_is_turned_on: old (not is_on and temperature <desired_temp −
          deviation) implies is_on
heater_remains_off:not (desired_temp > deviation)implies ( old (not is_on and
     temperature ≥ desired_temp − deviation) implies (not is_on))
    end
```

### 1.5 LAMP

The LAMP class, as presented below, implements a lamp that has a switch and a dimmer. Its light intensity has three levels: low, medium and high. When the lamp is turned on, its light intensity will be the same as its intensity before it was last turned off. Fig.45 shows the verification result: this version of LAMP is correctly verified. Based on the verified version, 4 faulty variants of LAMP class are created, which are discussed below.

```
class
    LAMP
feature
    light_intensity: INTEGER
        -- Light intensity of the lamp
    is_on: BOOLEAN
        -- Is the lamp on?
  previous_light_intensity: INTEGER
        -- Light intensity of the lamp before it was last turned off
    High_intensity: INTEGER = 100
        -- High light intensity
    Medium_intensity: INTEGER = 75
        -- Medium light intensity
    Low_intensity: INTEGER = 25
        -- Low light intensity
    Zero_intensity: INTEGER = 0
        -- Zero light intensity
feature
    turn_on_off
        -- Turn on the lamp if it is off; turn off the lamp if it is on
        do
            if not is_on then
                is_on := True
                if previous_light_intensity > 0then
                    light_intensity := previous_light_intensity
                else
                    light_intensity := Low_intensity
                end
            else
                is_on := False
                previous_light_intensity := light_intensity
                light_intensity := Zero_intensity
            end
        ensure
            turn_on_1: old (not is_on and previous_light_intensity > 0) implies (
                is_on and light_intensity = old previous_light_intensity)
            turn_on_2: old (not is_on and previous_light_intensity = 0) implies (
                is_on and light_intensity = Low_intensity)
```

```
            turn_off: old is_on implies (not is_on and previous_light_intensity = old light_intensity and light_intensity = Zero_intensity)
    end
    adjust_light
        -- Adjust the light intensity
        require
            lamp_is_on: is_on = True
        do
            if light_intensity = Low_intensity then
                light_intensity := Medium_intensity
            elseif light_intensity = Medium_intensity then
                light_intensity := High_intensity
            elseif light_intensity = High_intensity then
                light_intensity := Low_intensity
            end
        ensure
            from_low_to_medium: old light_intensity = Low_intensity implies light_intensity = Medium_intensity
            from_medium_to_high: old light_intensity = Medium_intensity implies light_intensity = High_intensity
            from_high_to_low: old light_intensity = High_intensity implies light_intensity = Low_intensity
        end
invariant
    value_of_light_intensity: light_intensity = Zero_intensity or light_intensity = Low_intensity or light_intensity = Medium_intensity or light_intensity = High_intensity
    value_of_previous_intensity: previous_light_intensity = Zero_intensity or previous_light_intensity = Low_intensity or previous_light_intensity = Medium_intensity or previous_light_intensity = High_intensity
    light_intensity_when_off: is_on = (light_intensity ≠ Zero_intensity)
end
```

AutoProof

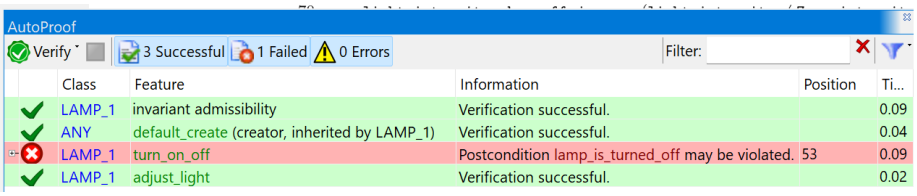
Verify | 4 Successful | 0 Failed | 0 Errors

| | Class | Feature | Information | Position | Ti... |
|---|---|---|---|---|---|
| ✓ | LAMP_1 | invariant admissibility | Verification successful. | | 0.14 |
| ✓ | ANY | default_create (creator, inherited by LAMP_1) | Verification successful. | | 0.05 |
| ✓ | LAMP_1 | turn_on_off | Verification successful. | | 0.04 |
| ✓ | LAMP_1 | adjust_light | Verification successful. | | 0.02 |

**Fig. 45.** Proof result of LAMP in AutoProof

**Variant 1 of LAMP**

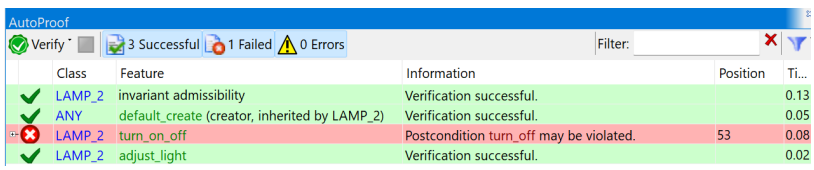
- Fault injection: in the body of `turn_on_off`, switch the order of line 40 and line 41.
- Resulting failure: as shown in Fig. 46(a), the injected fault results in the violation of the postcondition `turn_off` of the procedure `turn_on_off`.
- Cause of the failure: incorrect implementation of the routine body of `turn_on_off`; the value of `light_intensity` should be stored into `previous_light_intensity` before being assigned to a new value.
- Proof time: 0.250 sec
- Comment: the test is useful as its execution shows a specific trace illustrating how the program goes to the same contract violation as in the proof.
- Possible fixes: No valid fixes




(a)

**Variant 2 of** LAMP

- Fault injection: at line 40, remove the assignment `previous_light_intensity := light_intensity`.
- Resulting failure: as shown in Fig. 46(b), the injected fault leads to the violation of postcondition `turn_off` in the procedure `turn_on_off`.
- Cause of the failure: incorrect implementation of the body of `turn_on_off`; the postcondition `turn_off` requires that, the `previous_light_intensity` should store, when the light is turned off, the value of `light_intensity`; this is missing in the implementation.
- Proof time: 0.278 sec
- Comment: the test is useful as its execution shows a specific trace illustrating how the program goes to the same contract violation as in the proof.
- Possible fixes: 10 valid fixes out of 534 candidate fixes.
- fixing time: 6.83 minutes




(b)

```
1
2  turn_on_off_ID_362
3       -- Turn on the lamp if it is off; turn off the lamp if it is on
4  require
5  not (is_on)
6  do
7
8       if not is_on then
9         is_on := True
10        if previous_light_intensity > 0then
11          light_intensity := previous_light_intensity
12        else
13          light_intensity := Low_intensity
14        end
15      else
16        is_on := False
17          -- previous_light_intensity := light_intensity
18        light_intensity := Zero_intensity
19      end
20    ensure
21      turn_on_1: old (not is_on and previous_light_intensity > 0) implies (
            is_on and light_intensity = old previous_light_intensity)
```

```
      turn_on_2: old (not is_on and previous_light_intensity = 0) implies (is_on and light_intensity = Low_intensity)
      turn_off: old is_on implies (not is_on and previous_light_intensity = old light_intensity and light_intensity = Zero_intensity)
    end

turn_on_off_ID_368
      -- Turn on the lamp if it is off; turn off the lamp if it is on
require
not ((is_on) or else (not (not is_on)))
do

      if not is_on then
        is_on := True
        if previous_light_intensity > 0then
          light_intensity := previous_light_intensity
        else
          light_intensity := Low_intensity
        end
      else
        is_on := False
          -- previous_light_intensity := light_intensity
        light_intensity := Zero_intensity
      end
    ensure
      turn_on_1: old (not is_on and previous_light_intensity > 0) implies (is_on and light_intensity = old previous_light_intensity)
      turn_on_2: old (not is_on and previous_light_intensity = 0) implies (is_on and light_intensity = Low_intensity)
      turn_off: old is_on implies (not is_on and previous_light_intensity = old light_intensity and light_intensity = Zero_intensity)
    end

turn_on_off_ID_370
      -- Turn on the lamp if it is off; turn off the lamp if it is on
require
not ((is_on) or else (previous_light_intensity > 0))
do

      if not is_on then
        is_on := True
        if previous_light_intensity > 0then
          light_intensity := previous_light_intensity
        else
          light_intensity := Low_intensity
        end
      else
        is_on := False
          -- previous_light_intensity := light_intensity
        light_intensity := Zero_intensity
```

```
      end
    ensure
      turn_on_1: old (not is_on and previous_light_intensity > 0) implies (
          is_on and light_intensity = old previous_light_intensity)
      turn_on_2: old (not is_on and previous_light_intensity = 0) implies (
          is_on and light_intensity = Low_intensity)
      turn_off: old is_on implies (not is_on and previous_light_intensity =
          old light_intensity and light_intensity = Zero_intensity)
    end

turn_on_off_ID_372
      -- Turn on the lamp if it is off; turn off the lamp if it is on
require
not ((is_on) or else (not (previous_light_intensity > 0)))
do

      if not is_on then
        is_on := True
        if previous_light_intensity > 0then
          light_intensity := previous_light_intensity
        else
          light_intensity := Low_intensity
        end
      else
        is_on := False
          -- previous_light_intensity := light_intensity
        light_intensity := Zero_intensity
      end
    ensure
      turn_on_1: old (not is_on and previous_light_intensity > 0) implies (
          is_on and light_intensity = old previous_light_intensity)
      turn_on_2: old (not is_on and previous_light_intensity = 0) implies (
          is_on and light_intensity = Low_intensity)
      turn_off: old is_on implies (not is_on and previous_light_intensity =
          old light_intensity and light_intensity = Zero_intensity)
    end

turn_on_off_ID_380
      -- Turn on the lamp if it is off; turn off the lamp if it is on
require
not ((light_intensity) > (0))
do

      if not is_on then
        is_on := True
        if previous_light_intensity > 0then
          light_intensity := previous_light_intensity
        else
          light_intensity := Low_intensity
        end
```

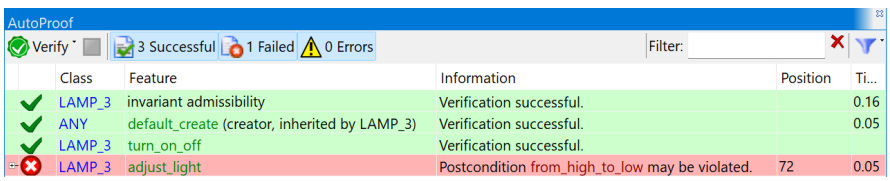
```
      else
        is_on := False
          -- previous_light_intensity := light_intensity
        light_intensity := Zero_intensity
      end
    ensure
      turn_on_1: old (not is_on and previous_light_intensity > 0) implies (
          is_on and light_intensity = old previous_light_intensity)
      turn_on_2: old (not is_on and previous_light_intensity = 0) implies (
          is_on and light_intensity = Low_intensity)
      turn_off: old is_on implies (not is_on and previous_light_intensity =
          old light_intensity and light_intensity = Zero_intensity)
    end

turn_on_off_ID_438
      -- Turn on the lamp if it is off; turn off the lamp if it is on
require
not ((light_intensity) ≥ (low_intensity))
do

      if not is_on then
        is_on := True
        if previous_light_intensity > 0then
          light_intensity := previous_light_intensity
        else
          light_intensity := Low_intensity
        end
      else
        is_on := False
          -- previous_light_intensity := light_intensity
        light_intensity := Zero_intensity
      end
    ensure
      turn_on_1: old (not is_on and previous_light_intensity > 0) implies (
          is_on and light_intensity = old previous_light_intensity)
      turn_on_2: old (not is_on and previous_light_intensity = 0) implies (
          is_on and light_intensity = Low_intensity)
      turn_off: old is_on implies (not is_on and previous_light_intensity =
          old light_intensity and light_intensity = Zero_intensity)
    end

turn_on_off_ID_444
      -- Turn on the lamp if it is off; turn off the lamp if it is on
require
not ((light_intensity) > (zero_intensity))
do

      if not is_on then
        is_on := True
        if previous_light_intensity > 0then
```

```
155          light_intensity := previous_light_intensity
156        else
157          light_intensity := Low_intensity
158        end
159      else
160        is_on := False
161          -- previous_light_intensity := light_intensity
162        light_intensity := Zero_intensity
163      end
164    ensure
165      turn_on_1: old (not is_on and previous_light_intensity > 0) implies (
             is_on and light_intensity = old previous_light_intensity)
166      turn_on_2: old (not is_on and previous_light_intensity = 0) implies (
             is_on and light_intensity = Low_intensity)
167      turn_off: old is_on implies (not is_on and previous_light_intensity =
             old light_intensity and light_intensity = Zero_intensity)
168    end
169
170  turn_on_off_ID_524
171      -- Turn on the lamp if it is off; turn off the lamp if it is on
172  require
173  not (light_intensity ≠ previous_light_intensity)
174  do
175
176      if not is_on then
177        is_on := True
178        if previous_light_intensity > 0then
179          light_intensity := previous_light_intensity
180        else
181          light_intensity := Low_intensity
182        end
183      else
184        is_on := False
185          -- previous_light_intensity := light_intensity
186        light_intensity := Zero_intensity
187      end
188    ensure
189      turn_on_1: old (not is_on and previous_light_intensity > 0) implies (
             is_on and light_intensity = old previous_light_intensity)
190      turn_on_2: old (not is_on and previous_light_intensity = 0) implies (
             is_on and light_intensity = Low_intensity)
191      turn_off: old is_on implies (not is_on and previous_light_intensity =
             old light_intensity and light_intensity = Zero_intensity)
192    end
193
194  turn_on_off_ID_528
195      -- Turn on the lamp if it is off; turn off the lamp if it is on
196  require
197  not (light_intensity ≥ low_intensity)
198  do
```

```
      if not is_on then
        is_on := True
        if previous_light_intensity > 0then
          light_intensity := previous_light_intensity
        else
          light_intensity := Low_intensity
        end
      else
        is_on := False
          -- previous_light_intensity := light_intensity
        light_intensity := Zero_intensity
      end
    ensure
      turn_on_1: old (not is_on and previous_light_intensity > 0) implies (
          is_on and light_intensity = old previous_light_intensity)
      turn_on_2: old (not is_on and previous_light_intensity = 0) implies (
          is_on and light_intensity = Low_intensity)
      turn_off: old is_on implies (not is_on and previous_light_intensity =
          old light_intensity and light_intensity = Zero_intensity)
    end

turn_on_off_ID_530
      -- Turn on the lamp if it is off; turn off the lamp if it is on
require
not (light_intensity > zero_intensity)
do

      if not is_on then
        is_on := True
        if previous_light_intensity > 0then
          light_intensity := previous_light_intensity
        else
          light_intensity := Low_intensity
        end
      else
        is_on := False
          -- previous_light_intensity := light_intensity
        light_intensity := Zero_intensity
      end
    ensure
      turn_on_1: old (not is_on and previous_light_intensity > 0) implies (
          is_on and light_intensity = old previous_light_intensity)
      turn_on_2: old (not is_on and previous_light_intensity = 0) implies (
          is_on and light_intensity = Low_intensity)
      turn_off: old is_on implies (not is_on and previous_light_intensity =
          old light_intensity and light_intensity = Zero_intensity)
    end
```

### Variant 3 of LAMP

- Fault injection: at line 59, in the body of `adjust_light`, change the right-hand side of the assignment from "`Low_intensity`" to "`Medium_intensity`".
- Resulting failure: as shown in Fig. 46(c), the injected fault causes the violation of postcondition `from_high_to_low` of the `adjust_light` routine.
- Cause of the failure: incorrect implementation of the routine body of `adjust_light`.
- Proof time: 0.265 sec
- Comment: the test is useful as its execution shows a specific trace illustrating how the program goes to the same contract violation as in the proof.
- Possible fixes: 4 valid fixes out of 422 candidate fixes.
- fixing time: 3.3 minutes




(c)

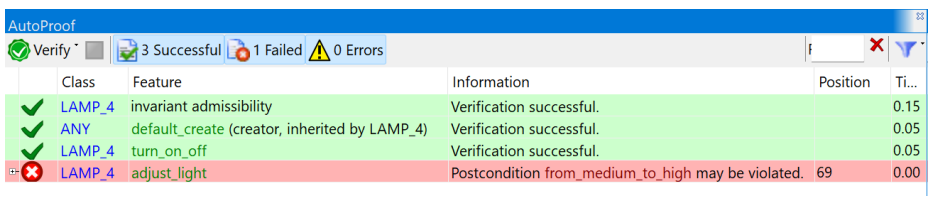
```
1
2  adjust_light_ID_354
3       -- Adjust the light intensity
4     require
5       lamp_is_on: is_on = True
6  not (light_intensity = high_intensity)
7     do
8
9         if light_intensity = Low_intensity then
10          light_intensity := Medium_intensity
11        elseif light_intensity = Medium_intensity then
12          light_intensity := High_intensity
13        elseif light_intensity = High_intensity then
14          light_intensity := Medium_intensity
15        end
16    ensure
17      from_low_to_medium: old light_intensity = Low_intensity implies
            light_intensity = Medium_intensity
18      from_medium_to_high: old light_intensity = Medium_intensity implies
            light_intensity = High_intensity
19      from_high_to_low: old light_intensity = High_intensity implies
            light_intensity = Low_intensity
20    end
21
22 adjust_light_ID_394
```

```
      -- Adjust the light intensity
    require
      lamp_is_on: is_on = True
not ((light_intensity) = (high_intensity))
    do

        if light_intensity = Low_intensity then
          light_intensity := Medium_intensity
        elseif light_intensity = Medium_intensity then
          light_intensity := High_intensity
        elseif light_intensity = High_intensity then
          light_intensity := Medium_intensity
        end
    ensure
      from_low_to_medium: old light_intensity = Low_intensity implies
          light_intensity = Medium_intensity
      from_medium_to_high: old light_intensity = Medium_intensity implies
          light_intensity = High_intensity
      from_high_to_low: old light_intensity = High_intensity implies
          light_intensity = Low_intensity
    end

adjust_light_ID_398
      -- Adjust the light intensity
    require
      lamp_is_on: is_on = True
not ((light_intensity) ≥ (high_intensity))
    do

        if light_intensity = Low_intensity then
          light_intensity := Medium_intensity
        elseif light_intensity = Medium_intensity then
          light_intensity := High_intensity
        elseif light_intensity = High_intensity then
          light_intensity := Medium_intensity
        end
    ensure
      from_low_to_medium: old light_intensity = Low_intensity implies
          light_intensity = Medium_intensity
      from_medium_to_high: old light_intensity = Medium_intensity implies
          light_intensity = High_intensity
      from_high_to_low: old light_intensity = High_intensity implies
          light_intensity = Low_intensity
    end

adjust_light_ID_406
      -- Adjust the light intensity
    require
      lamp_is_on: is_on = True
not ((light_intensity) > (low_intensity))
```

```
    do

        if light_intensity = Low_intensity then
          light_intensity := Medium_intensity
        elseif light_intensity = Medium_intensity then
          light_intensity := High_intensity
        elseif light_intensity = High_intensity then
          light_intensity := Medium_intensity
        end
    ensure
      from_low_to_medium: old light_intensity = Low_intensity implies
          light_intensity = Medium_intensity
      from_medium_to_high: old light_intensity = Medium_intensity implies
          light_intensity = High_intensity
      from_high_to_low: old light_intensity = High_intensity implies
          light_intensity = Low_intensity
    end

adjust_light_ID_414
      -- Adjust the light intensity
    require
      lamp_is_on: is_on = True
not (light_intensity = 100)
    do

        if light_intensity = Low_intensity then
          light_intensity := Medium_intensity
        elseif light_intensity = Medium_intensity then
          light_intensity := High_intensity
        elseif light_intensity = High_intensity then
          light_intensity := Medium_intensity
        end
    ensure
      from_low_to_medium: old light_intensity = Low_intensity implies
          light_intensity = Medium_intensity
      from_medium_to_high: old light_intensity = Medium_intensity implies
          light_intensity = High_intensity
      from_high_to_low: old light_intensity = High_intensity implies
          light_intensity = Low_intensity
    end
```

### Variant 4 of LAMP

- Fault injection: at line 57, change the right-hand side of the assignment from "`High_intensity`" to "`Medium_intensity`".
- Resulting failure: as shown in Fig. 46(d), the postcondition `from_high_to_low` is violated.
- Cause of the failure: incorrect implementation.
- Proof time: 0.270 sec
- Possible fixes: 4 valid fixes out of 406 candidate fixes.
- fixing time: 3.62 minutes




(d)

```
1
2  adjust_light_ID_332
3       -- Adjust the light intensity
4     require
5       lamp_is_on: is_on = True
6  not (light_intensity = medium_intensity)
7     do
8
9       if light_intensity = Low_intensity then
10        light_intensity := Medium_intensity
11      elseif light_intensity = Medium_intensity then
12        light_intensity := Medium_intensity
13      elseif light_intensity = High_intensity then
14        light_intensity := Low_intensity
15      end
16    ensure
17      from_low_to_medium: old light_intensity = Low_intensity implies
            light_intensity = Medium_intensity
18      from_medium_to_high: old light_intensity = Medium_intensity implies
            light_intensity = High_intensity
19      from_high_to_low: old light_intensity = High_intensity implies
            light_intensity = Low_intensity
20    end
21
22 adjust_light_ID_378
23      -- Adjust the light intensity
```

```
    require
      lamp_is_on: is_on = True
not ((light_intensity) <(high_intensity))
    do

      if light_intensity = Low_intensity then
        light_intensity := Medium_intensity
      elseif light_intensity = Medium_intensity then
        light_intensity := Medium_intensity
      elseif light_intensity = High_intensity then
        light_intensity := Low_intensity
      end
    ensure
      from_low_to_medium: old light_intensity = Low_intensity implies
          light_intensity = Medium_intensity
      from_medium_to_high: old light_intensity = Medium_intensity implies
          light_intensity = High_intensity
      from_high_to_low: old light_intensity = High_intensity implies
          light_intensity = Low_intensity
    end

adjust_light_ID_384
      -- Adjust the light intensity
    require
      lamp_is_on: is_on = True
not ((light_intensity) > (low_intensity))
    do

      if light_intensity = Low_intensity then
        light_intensity := Medium_intensity
      elseif light_intensity = Medium_intensity then
        light_intensity := Medium_intensity
      elseif light_intensity = High_intensity then
        light_intensity := Low_intensity
      end
    ensure
      from_low_to_medium: old light_intensity = Low_intensity implies
          light_intensity = Medium_intensity
      from_medium_to_high: old light_intensity = Medium_intensity implies
          light_intensity = High_intensity
      from_high_to_low: old light_intensity = High_intensity implies
          light_intensity = Low_intensity
    end

adjust_light_ID_392
      -- Adjust the light intensity
    require
      lamp_is_on: is_on = True
not (light_intensity = 75)
    do
```

```
    if light_intensity = Low_intensity then
      light_intensity := Medium_intensity
    elseif light_intensity = Medium_intensity then
      light_intensity := Medium_intensity
    elseif light_intensity = High_intensity then
      light_intensity := Low_intensity
    end
  ensure
    from_low_to_medium: old light_intensity = Low_intensity implies
        light_intensity = Medium_intensity
    from_medium_to_high: old light_intensity = Medium_intensity implies
        light_intensity = High_intensity
    from_high_to_low: old light_intensity = High_intensity implies
        light_intensity = Low_intensity
  end
```

# 2 Examples with loops

## 2.1 BINARY_SEARCH

The BINARY_SEARCH class, as shown below, implements the binary search algorithm, which aims to search a value in a sorted integer array by repeatedly dividing the search interval in half. Fig.46 shows the verification result of the class: the implementation is correct with respect to the specification. Based on the correct version, 6 variants of the class are derived and further discussed below.

```
class
    BINARY_SEARCH
feature -- Binary search
    binary_search(a: V_ARRAY [INTEGER]; value: INTEGER): INTEGER
            -- Index of ‘value’ in ‘a’ using binary search. Return 0if not found.
            -- https://en.wikipedia.org/wiki/Binary_search_algorithm#Iterative
        note
            status: impure
        require
            no_overflow: a.count <{INTEGER}.max_value
            a_sorted: across 1|.. | a.count as i all
                        across 1|.. | a.count as j all
                            i ≤ j implies a.sequence[i] ≤ a.sequence[j] end end
            a_size_limit: a.count > 0 and a.count ≤ 10
            a_valid_bound: a.lower < a.upper and a.lower = 1
        local
            low, up, middle: INTEGER
        do
            from
                low := a.lower
                up := a.upper + 1
                Result := a.lower − 1
            invariant
                low_and_up_range: a.lower ≤ low and low ≤ up and up ≤ a.upper + 1
                valid_bound: a.lower < a.upper
                result_range: Result = a.lower − 1 or a.lower ≤ Result and Result ≤ a.upper
                not_in_lower_part: across 1|.. | (low − a.lower) as i all a.sequence[i] <value end
                not_in_upper_part: across (up − a.lower + 1) |.. | a.sequence.count as i all value < a.sequence[i] end
                found: (Result ≥ a.lower and Result ≤ a.upper) implies (a.sequence[Result − a.lower + 1] = value)
            until
```

```
31                  low ≥ up or Result ≥ a.lower
32              loop
33                  middle := low + ((up − low) // 2)
34                  if a[middle] < value then
35                      low := middle + 1
36                  elseif a[middle] > value then
37                      up := middle
38                  else
39                      Result := middle
40                  end
41              variant
42                  (a.upper − Result) + (up − low)
43              end
44          ensure
45              present: a.sequence.has (value) = (Result ≥ a.lower and Result ≤ a.
                    upper)
46              not_present: not a.sequence.has (value) = (Result = a.lower − 1)
47              found_if_present: (Result ≥ a.lower and Result ≤ a.upper) implies (a.
                    sequence[Result − a.lower + 1] = value)
48          end
49  end
```

AutoProof

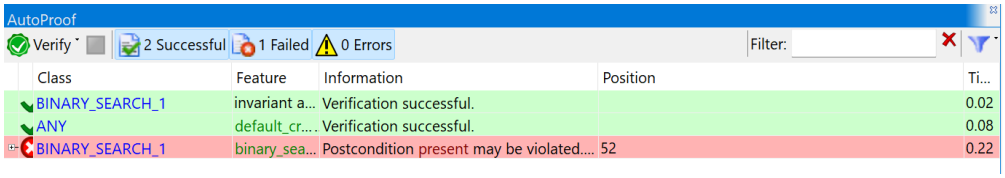
Verify | 3 Successful | 0 Failed | 0 Errors | Filter:

| Class | Feature | Information | Position | Ti... |
|---|---|---|---|---|
| BINARY_SEARCH | invariant admissibility | Verification successful. | | 0.25 |
| ANY | default_create (creator, inherited by BINARY_SEARCH) | Verification successful. | | 0.08 |
| BINARY_SEARCH | binary_search | Verification successful. | | 0.03 |

**Fig. 46.** Proof result of BINARY_SEARCH in AutoProof

**Variant 1 of** `BINARY_SEARCH`

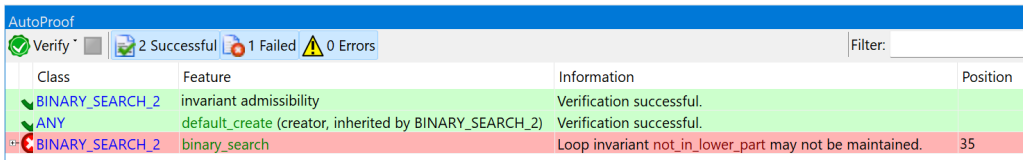
- Fault injection: at line 30, remove the loop invariant `found`.
- Resulting failure: as shown in Fig. 47(a), the injected fault leads to the violation of the postcondition `present`.
- Cause of the failure: weakness/incompleteness of loop invariant.
- Proof time: 0.327 sec
- Comment: when trying to verify postcondition `present`, the prover uses the loop invariant, instead of the loop body, to represent the behaviors of the loop; if the loop invariant is not strong enough to express the functionality of the loop, as in this example, the prover is not able to establish the relevant postcondition; in this case, the counterexample (from which the test is extracted from) is not a real "counterexample" — it does not reveal the fault in the implementation and thus running the resulting test will not raise any exception; the passing test, however, indicates the weakness of the loop invariant.
- Possible fixes: No valid fixes




(a)

**Variant 2 of** `BINARY_SEARCH`

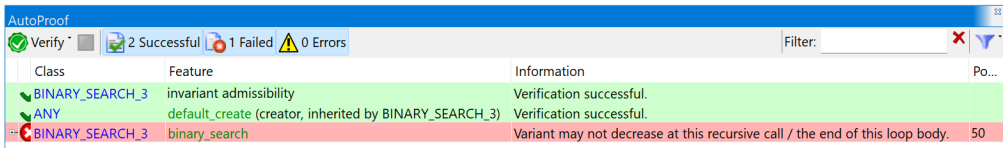
- Fault injection: at line 35, change the condition of the then branch from "`a [middle]<value`" into "`a [middle]≤value`".
- Resulting failure: as shown in Fig. 47(b), the injected fault results in the violation of loop invariant `not_in_lower_part`.
- Cause of the failure: incorrect implementation of loop body.
- Proof time: 0.395 sec
- Comment: the test is useful as it reveals the fault in the program: the first element of the array has the same value as `value` (the value to search); when running the test, it is supposed that the loop ends at the first iteration — `value` is found at position 1 of the array; but due to the incorrect implementation of the routine, the iteration does not stop (the exit condition remains true after the first iteration) and continues for the second iteration, at which the loop invariant `not_in_lower_part` is evaluated to false (`a.sequence`[1] = `value`) and thus causes the exception.
- Possible fixes: No valid fixes




(b)

### Variant 3 of BINARY_SEARCH

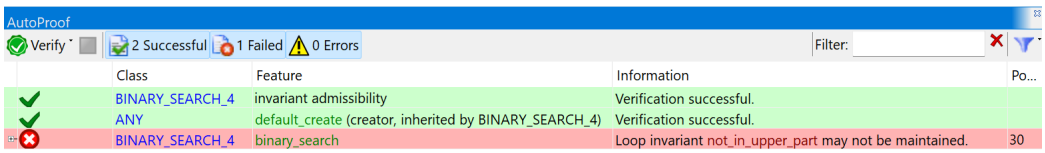
- Fault injection: at line 36, change the assignment from "`low` := `middle` + `1`" into "`low` := `middle`".
- Resulting failure: as shown in Fig. 47(c), the injected fault results in the violation that the variant of the loop does not decrease.
- Cause of the failure: incorrect implementation of loop body.
- Proof time: 0.325 sec
- Comment: the test is useful as it reveals a bug in the program: initially, `low` = `1` and `upper` = `2`; at the first iteration, the program assigns `middle` with `1` (at line 34); the condition of the then branch at line 35 is evaluated true,
- Possible fixes: No valid fixes




(c)

**Variant 4 of** `BINARY_SEARCH`

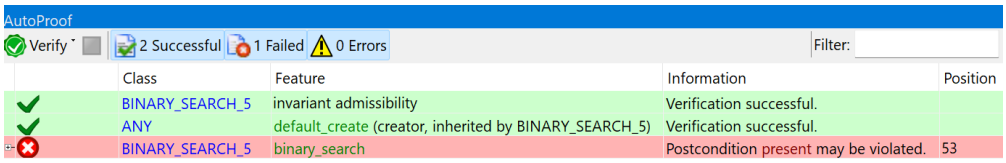
- Fault injection: at line 37, change the condition of the `elseif` branch from "`a[middle]>value`" into "`a[middle]≥value`".
- Resulting failure: as shown in Fig. 47(d), the loop invariant of `not_in_upper_part` is violated.
- Cause of the failure: incorrect implementation of loop body.
- Proof time: 0.288 sec
- Comment: this variant is similar to Variant 3; the test is useful as it shows a concrete trace that leads to the violation of the same contract in the failed proof.
- Possible fixes: No valid fixes




(d)

### Variant 5 of BINARY_SEARCH

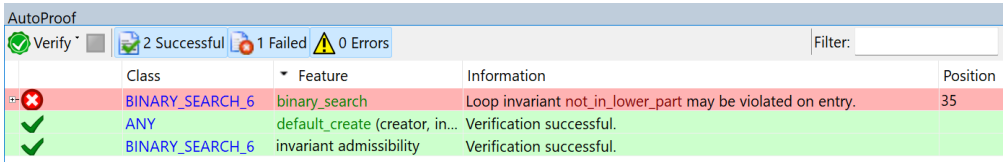
- Fault injection: at line 28, remove the loop invariant `not_in_lower_part`.
- Resulting failure: as shown in Fig. 47(e), the removal of the loop invariant causes the violation of postcondition of `present`.
- Cause of the failure: weakness/incompleteness of loop invariant.
- Proof time: 0.550 sec
- Comment: this variant is similar to Variant 1; the passing test indicates that the proof failure is caused by the weakness of the auxiliary specification (loop invariant), not the implementation.
- Possible fixes: No valid fixes




(e)

**Variant 6 of** `BINARY_SEARCH`

- Fault injection: at line 21, change the loop initialization "`low := a.lower`" into "`low := a.lower + 1`".
- Resulting failure: as shown in Fig. 47(f), the injected fault leads to the violation of the loop invariant `not_in_lower_part` at the entry of the loop (after loop initialization).
- Cause of the failure: incorrect implementation of loop initialization.
- Proof time: 0.356 sec
- Comment: the test is useful as its execution demonstrates a specific case where the program goes to a failure state, violating the same contract as in the proof failure; the values in the test input, however, is not that meaningful to this failure, as running the program with any valid input would cause the same contract violation.
- Possible fixes: No valid fixes




(f)

## 2.2 LINEAR_SEARCH

The LINEAR_SEARCH class, which is displayed below, implements a function that returns the index of a given integer '`value`' in an integer array '`a`' using linear search starting from beginning of the array; if the '`value`' is not found in '`a`', the function returns the value "`a.count + 1`" (`a.count` represents the number of elements in `a`). Fig.47 shows the verification result of LINEAR_SEARCH, which indicates the complete correctness of its functionality. 4 variants of LINEAR_SEARCH are produced based on the correct version and are discussed below.

```
class
    LINEAR_SEARCH
feature -- Basic operations
    linear_search (a: SIMPLE_ARRAY [INTEGER]; value: INTEGER): INTEGER
        require
            array_not_empty: a.count > 0
        do
            from
                Result := 1
            invariant
                result_in_bound: 1 ≤ Result and Result ≤ a.count + 1
                not_present_so_far: across 1|.. | (Result − 1) as i all a.sequence [i]
                    ≠ value end
            until
                Result = a.count + 1 or else a [Result] = value
            loop
                Result := Result + 1
            variant
                a.count − Result + 1
            end
        ensure
            result_in_bound: 1 ≤ Result and Result ≤ a.count + 1
            present: a.sequence.has (value) = (Result ≤ a.count)
            found_if_present: (Result ≤ a.count) implies a.sequence [Result] =
                value
            first_from_front: across 1|.. | (Result − 1) as i all a.sequence [i] ≠
                value end
        end
end
```

AutoProof

Verify | 3 Successful | 0 Failed | 0 Errors

| Class | Feature | Information | Position | Ti... |
|---|---|---|---|---|
| LINEAR_SEARCH_2 | invariant ad... | Verification successful. | | 0.56 |
| ANY | default_creat... | Verification successful. | | 0.02 |
| LINEAR_SEARCH_2 | linear_search | Verification successful. | | 0.07 |

**Fig. 47.** Proof result of `LINEAR_SEARCH` in AutoProof

### Variant 1 of `LINEAR_SEARCH`

- Fault injection: at line 10, change the loop initialization from "`Result` := 1" into "`Result` := 0".
- Resulting failure: as shown in Fig. 48(a), the injected fault leads to the violation of the loop invariant `result_in_bound` at the entry of the loop (after loop initialization).
- Cause of the failure: incorrect implementation of loop initialization.
- Proof time: 0.709 sec
- Comment: similar to the Variant 6 of `BINARY_SEARCH`, the test is useful as its execution demonstrates a specific case where the program goes to a failure state, violating the same contract as in the proof failure; the values in the test input, however, is not that meaningful to this failure, as running the program with any valid input would cause the same contract violation.
- Possible fixes: No valid fixes

AutoProof

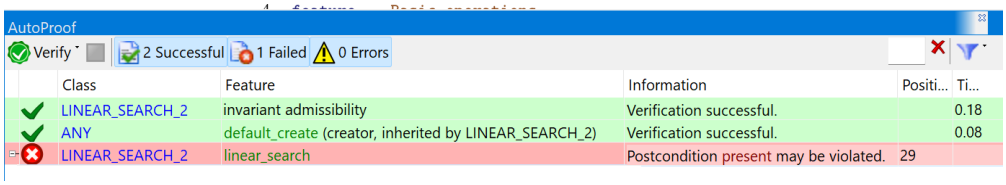
Verify | 2 Successful | 1 Failed | 0 Errors | Filter:

| Class | Feature | Information | P... | Ti... |
|---|---|---|---|---|
| LINEAR_SEARCH_1 | invariant admissibility | Verification successful. | | 0.61 |
| ANY | default_create (creator, inherited by LINEAR_SEARCH_1) | Verification successful. | | 0.08 |
| LINEAR_SEARCH_1 | linear_search | Loop invariant result_in_bound may be violated on entry. | 17 | 0.01 |

(a)

**Variant 2 of** `LINEAR_SEARCH`

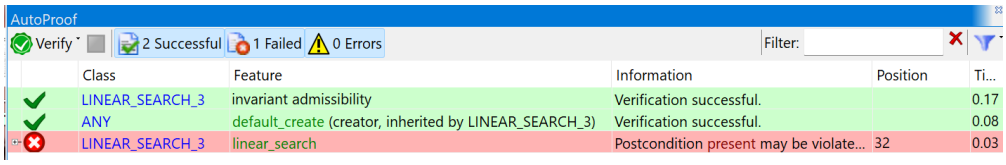
- Fault injection: at line 12, change the left part of the exit condition from "`Result` = `a.count` + `1`" into "`Result` = `a.count`".
- Resulting failure: as shown in Fig. 48(b), the injected fault results in the violation of the postcondition `present`.
- Cause of the failure: incorrect exit condition (the condition for a loop to terminate).
- Proof time: 0.283 sec
- Comment: during the execution of the test, the program terminates after 1 iteration with `Result` = `2`; this leads to the violation of the equality in the postcondition `present` — the left-hand part `a.sequence.has` (`value`) is false, as `value` does not match to any element of the input array `a`, while the right-hand part `Result`≤`a.count` is true (`Result` = `2` and `a.count` = `2`).
- Possible fixes: No valid fixes




(b)

**Variant 3 of** `LINEAR_SEARCH`

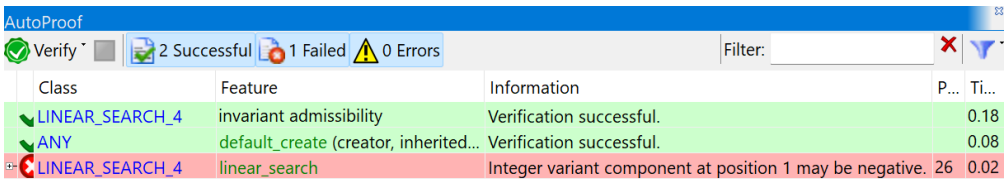
- Fault injection: at line 13, remove the loop invariant `not_present_so_far`.
- Resulting failure: as shown in Fig. 48(c), the injected fault causes the violation of the postcondition `present`.
- Cause of the failure: weakness/incompleteness of loop invariant.
- Proof time: 0.280 sec
- Comment: this variant is similar to Variant 5 of `BINARY_SEARCH`; the passing test indicates that the proof failure is caused by the weakness of the auxiliary specification (loop invariant), not the implementation.
- Possible fixes: No valid fixes




(c)

**Variant 4 of** `LINEAR_SEARCH`

- Fault injection: change the loop variant at line 19 from "`a.count` − `Result` + `1`" into "`a.count` − `Result` − `1`".
- Resulting failure: as shown in Fig. 48(d), the injected faults leads to the violation that "the integer variant component at iteration 1 may be negative".
- Cause of the failure: incorrect loop variant.
- Proof time: 0.279 sec
- Comment: the test is useful as it is able to show how the value of variant varies at each iteration; the values in the test input, however, is not that meaningful, as any other valid test input will have the same effect.
- Possible fixes: No valid fixes




(d)

### 2.3 MAX_IN_ARRAY

The MAX_IN_ARRAY class, as presented below, implements an algorithm that computes the maximum element of an integer array a. Fig.48 shows the verification result of the class, which suggests a complete functional correctness. 6 variants of the class are generated by injecting different faults in the correct version, which will be discussed below.

```
class
    MAX_IN_ARRAY
feature -- Basic operations
    max_in_array (a: SIMPLE_ARRAY [INTEGER]): INTEGER
            -- Find the maximum element of ‘a’.
        require
            array_not_empty: a.count > 0
        local
            i: INTEGER
        do
            Result := a [1]
            from
                i := 2
            invariant
                i_in_bounds: 2 ≤ i and i ≤ a.count + 1
                max_so_far: across 1|.. | (i − 1) as c all a.sequence [c] ≤ Result end
                result_in_array: across 1|.. | (i − 1) as c some a.sequence [c] = Result
                    end
            until
                i = a.count + 1
            loop
                if a [i] > Result then
                    Result := a [i]
                end
                i := i + 1
                variant
                    a.count − i
            end
        ensure
            is_maximum: across 1|.. | a.count as c all a.sequence [c] ≤ Result end
            result_in_array: across 1|.. | a.count as c some a.sequence [c] = Result end
        end
end
```

AutoProof

Verify | 3 Successful | 0 Failed | 0 Errors | Filter:

| | Class | Feature | Information | Position |
|---|---|---|---|---|
| ✓ | MAX_IN... | invariant admissibility | Verification successful. | |
| ✓ | ANY | default_create (creator, inherited by MAX_IN_ARRAY) | Verification successful. | |
| ✓ | MAX_IN... | max_in_array | Verification successful. | |

**Fig. 48.** Proof result of `MAX_IN_ARRAY` in AutoProof

### Variant 1 of `MAX_IN_ARRAY`

- Fault injection: at line 22, change the condition of the then branch into "`a [i] <Result`".
- Resulting failure: as shown in Fig. 49(a), the loop invariant `max_so_far` is violated during the iteration of the loop.
- Cause of the failure: incorrect implementation of the loop body.
- Proof time: 0.288 sec
- Comment: the test is useful as it is able to show a concrete trace that leads to the contract violation; the values in the test input, however, is not that meaningful, as any other valid test input will have the same effect.
- Possible fixes: No valid fixes

AutoProof

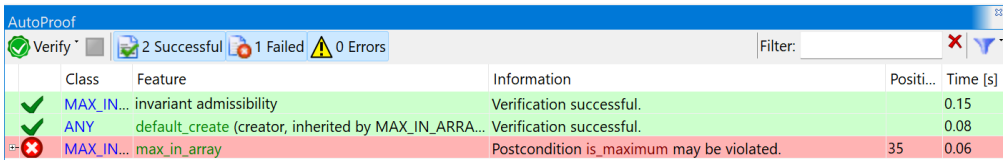
Verify | 2 Successful | 1 Failed | 0 Errors | Filter:

| | Class | Feature | Information | Positi... | Time |
|---|---|---|---|---|---|
| ✓ | MAX_IN... | invariant admissibility | Verification successful. | | 0.14 |
| ✓ | ANY | default_create (creator, inherited by MAX_IN_ARRA... | Verification successful. | | 0.08 |
| ✗ | MAX_IN... | max_in_array | Loop invariant max_so_far may not be maintained. | 22 | 0.06 |

(a)

**Variant 2 of `MAX_IN_ARRAY`**

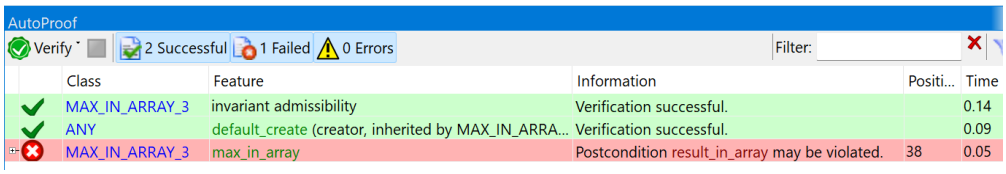
- Fault injection: at line 17, remove the loop invariant `max_so_far`.
- Resulting failure: as shown in Fig. 49(b), the removal of the loop invariant results in the violation of postcondition *is_maximum*.
- Cause of the failure: weakness/incompleteness of loop invariant.
- Proof time: 0.284 sec
- Comment: this variant is similar to Variant 5 of `BINARY_SEARCH`; the passing test indicates that the proof failure is caused by the weakness of the auxiliary specification (loop invariant), not the implementation.
- Possible fixes: No valid fixes




(b)

**Variant 3 of `MAX_IN_ARRAY`**

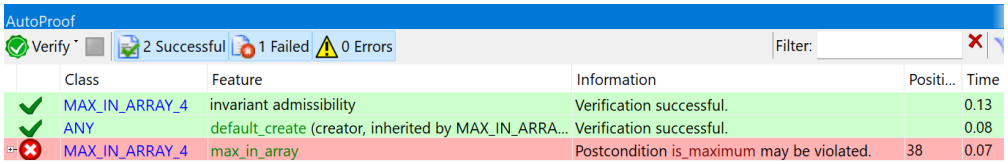
- Fault injection: at line 18, remove the loop invariant *result_in_array*.
- Resulting failure: as shown in Fig. 49(c), the removal of the loop invariant leads to the violation of the postcondition *result_in_array*.
- Cause of the failure: weakness/incompleteness of loop invariant.
- Proof time: 0.278 sec
- Comment: similar to Variant 2, the passing test indicates that the proof failure is caused by the weakness of the loop invariant.
- Possible fixes: No valid fixes




(c)

**Variant 4 of** `MAX_IN_ARRAY`

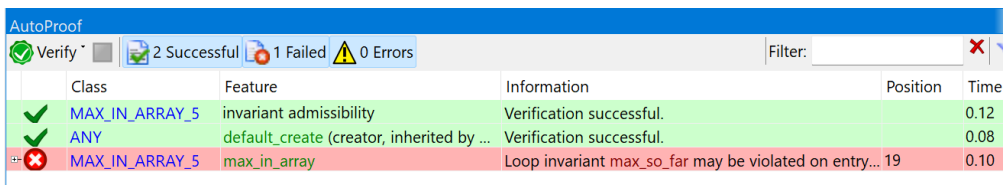
- Fault injection: at line 20, change the exit condition of the loop from "`i` = `a.count` + `1`" into "`i` ≥ `a.count`".
- Resulting failure: as shown in Fig. 49(d), the injected faults causes the violation of the postcondition *is_maximum*.
- Cause of the failure: incorrect exit condition of the loop.
- Proof time: 0.286 sec
- Comment: this test is useful as it shows a specific scenario from which the program will go to a failing state, violating the same failed contract in the proof; during the execution of the test, after going through 2 iterations, the program terminates with `Result` = `0`; this reveals the program fault: the program terminates too early to reach the actual maximum value of `a` (the third element of `a`).
- Possible fixes: No valid fixes




(d)

**Variant 5 of** `MAX_IN_ARRAY`

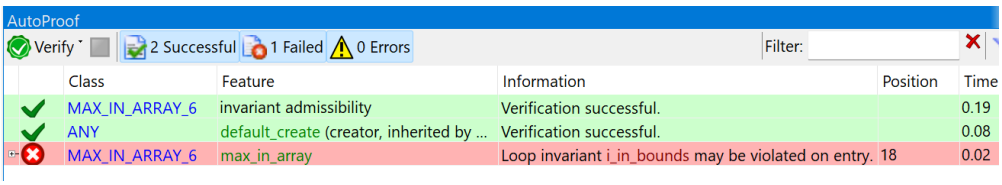
- Fault injection: at line 12, change the statement from "`Result` := `a` [1]" into "`Result` := 0".
- Resulting failure: as shown in Fig. 49(e), the injected fault causes the violation of the loop invariant `max_so_far` on the entry.
- Cause of the failure: incorrect implementation in the code snippet before the loop.
- Proof time: 0.290 sec
- Comment: the test is useful as it is able to show a concrete trace that leads to the contract violation; the values in the test input, however, is not that meaningful, as any other valid test input will have the same effect.
- Possible fixes: No valid fixes




(e)

**Variant 6 of** `MAX_IN_ARRAY`

- Fault injection: at line 14, change the code from "`i := 2`" into "`i := 1`".
- Resulting failure: as shown in Fig. 49(f), the injected fault causes the violation of the loop invariant `i_in_bounds` at the entry of the loop (after loop initialization).
- Cause of the failure: incorrect implementation of the loop initialization.
- Proof time: 0.282 sec
- Comment: similar to Variant 5, the test is useful as it is able to show a concrete trace that leads to the contract violation; the values in the test input, however, is not that meaningful, as any other valid test input will have the same effect.
- Possible fixes: No valid fixes




(f)

## 2.4 SQUARE_ROOT

The SQUARE_ROOT class, as shown below, calculates two approximate square roots x and y of a given positive integer n: the value of n falls between $x^2$ and $y^2$; if n is perfect square, then y = x and $x^2$=n, otherwise y = x + 1. Fig. 49 presents the verification result of the class, which indicates a full functional correctness. By injecting different faults in the correct version, 4 variants of SQUARE_ROOT are derived and discussed below.

```
class
    SQUARE_ROOT
feature
    square_root (n: INTEGER): TUPLE [x: INTEGER; y: INTEGER]
            -- 'x' and 'y' are two approximate square roots of 'n'
        require
            valid_n: n ≥ 0
        local
            x1, x2, mid: INTEGER
        do
            from
                x1 := 0
                x2 := n
            invariant
                valid_result: (x1 = x2 and x1 * x1 = n) or (x1 < x2 and x1 * x1 < n
                    and x2 * x2 ≥ n)
            until
                x2 − x1 ≤ 1 or x1 = x2
            loop
                mid := (x1 + x2) // 2-- integer division
                if mid * mid = n then
                    x1 := mid
                    x2 := mid
                else
                    if mid * mid < n then
                        x1 := mid
                    else
                        x2 := mid
                    end
                end
            variant
                x2 − x1
            end
            Result := [x1, x2]
        ensure
            valid_result: (Result.x = Result.y and Result.x * Result.x = n)
                or (Result.x + 1= Result.y and Result.x * Result.x < n and Result.y
                    * Result.y ≥ n)
```

```
        end
end
```

AutoProof

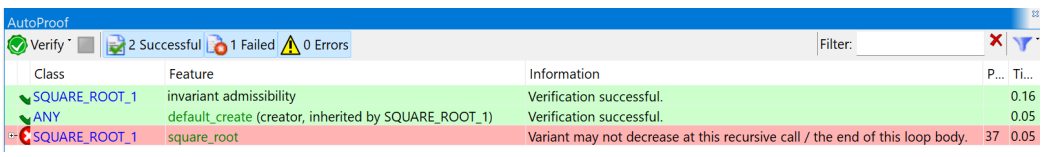
Verify | 3 Successful | 0 Failed | 0 Errors

| Class | Feature | Information | P... | Ti... |
|---|---|---|---|---|
| SQUARE_ROOT | invariant admissibility | Verification successful. | | 0.19 |
| ANY | default_create (creator, inherited by SQUARE_ROOT) | Verification successful. | | 0.04 |
| SQUARE_ROOT | square_root | Verification successful. | | 0.02 |

**Fig. 49.** Proof result of SQUARE_ROOT in AutoProof

**Variant 1 of SQUARE_ROOT**

- Fault injection: at line 17, change the left part of the exit condition from "x2− x1≤1" into "x2− x1<1".
- Resulting failure: as shown in Fig. 50(a), the injected fault leads to the failure that the loop variant is not decreased during the loop iteration.
- Cause of the failure: incorrect exit condition of the loop.
- Proof time: 0.258 sec
- Comment: the test is useful as it is able to show how the value of variant varies at each iteration.
- Possible fixes: 1 valid fixes out of 9 candidate fixes.
- fixing time: 1.58 minutes



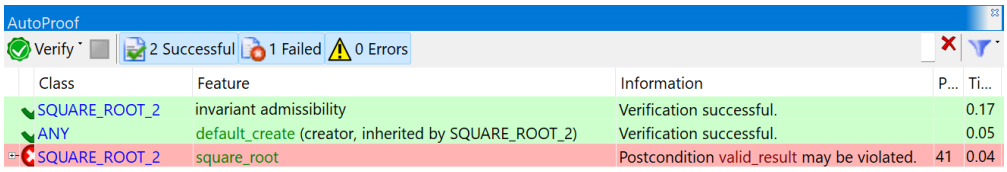

(a)

```
1
2  square_root_ID_3 (n: INTEGER): TUPLE [x: INTEGER; y: INTEGER]
3       -- 'x' and 'y' are two approximate square roots of 'n'
4     require
5       valid_n: n ≥ 0
6  not ((n) > (0))
7     local
8       x1, x2, mid: INTEGER
9     do
10
11      from
12        x1 := 0
13        x2 := n
14      invariant
15        valid_result: (x1 = x2 and x1 * x1 = n) or (x1 < x2 and x1 * x1 < n and x2
            * x2 ≥ n)
16      until
17        x2 − x1 < 1or x1 = x2
18      loop
19        mid := (x1 + x2) // 2-- integer division
20        if mid * mid = n then
21          x1 := mid
22          x2 := mid
23        else
24          if mid * mid < n then
```

```
            x1 := mid
          else
            x2 := mid
          end
        end
      variant
        x2 − x1
      end
      Result := [x1, x2]
    ensure
      valid_result: (Result.x = Result.y and Result.x * Result.x = n)
        or (Result.x + 1 = Result.y and Result.x * Result.x < n and Result.y *
            Result.y ≥ n)
    end
```

### Variant 2 of `SQUARE_ROOT`

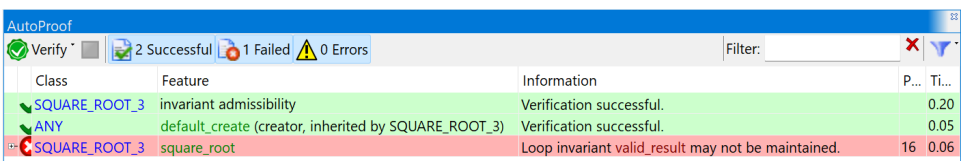
- Fault injection: at line 15, remove the loop invariant `valid_result`.
- Resulting failure: as shown in Fig. 50(b), the injected fault leads to the violation of postcondition `valid_result`.
- Cause of the failure: weakness/incompleteness of loop invariant.
- Proof time: 0.257 sec
- Comment: similar to Variant 2 of `MAX_IN_ARRAY`, the passing test indicates that the proof failure is caused by the weakness of the loop invariant.
- Possible fixes: No valid fixes




(b)

**Variant 3 of SQUARE_ROOT**

- Fault injection: change the condition of the then branch at line 20 from "`mid * mid = n`" into "`mid * mid ≠ n`".
- Resulting failure: as shown in Fig. 50(c), the injected fault results in the violation of the loop invariant `result_so_far`.
- Cause of the failure: incorrect implementation of the loop body.
- Proof time: 0.308 sec
- Comment: the test is useful as it is able to show a concrete trace that leads to the same contract violation as in the proof; during the execution of the test, initially, `x1= 0` and `x2= 3`; at the first iteration, the program assigns `mid` with 1 (line 19); the condition of the then branch is true (`mid * mid ≠ n`) and the program assigns both `x1` and `x2` with the value of `mid` (line 21 and 22); at the beginning of the second iteration, the loop invariant `valid_result` is evaluated as false (`x1= x2` is true but `x1*x1=n` is false).
- Possible fixes: 1 valid fixes out of 9 candidate fixes.
- fixing time: 1.68 minutes



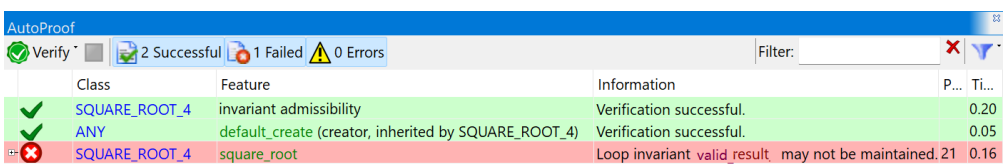

(c)

```
1
2  square_root_ID_3 (n: INTEGER): TUPLE [x: INTEGER; y: INTEGER]
3       -- 'x' and 'y' are two approximate square roots of 'n'
4     require
5       valid_n: n ≥ 0
6  not ((n) > (0))
7     local
8       x1, x2, mid: INTEGER
9     do
10
11      from
12        x1 := 0
13        x2 := n
14      invariant
15        valid_result: (x1 = x2 and x1 * x1 = n) or (x1 < x2 and x1 * x1 < n and x2
            * x2 ≥ n)
16      until
17        x2 − x1 ≤ 1or x1 = x2
18      loop
19        mid := (x1 + x2) // 2-- integer division
```

```
      if mid * mid ≠ n then
        x1 := mid
        x2 := mid
      else
        if mid * mid <n then
          x1 := mid
        else
          x2 := mid
        end
      end
    variant
      x2 − x1
    end
    Result := [x1, x2]
  ensure
    valid_result: (Result.x = Result.y and Result.x * Result.x = n)
      or (Result.x + 1 = Result.y and Result.x * Result.x <n and Result.y *
          Result.y ≥ n)
  end
```

**Variant 4 of** `SQUARE_ROOT`

- Fault injection: at line 24, change the condition of the then branch from "`mid * mid <n`" into "`mid * mid > n`".
- Resulting failure: as shown in Fig. 50(d), the loop invariant `valid_result` is not satisfied during the loop iteration.
- Cause of the failure: incorrect implementation of the loop body.
- Proof time: 0.400 sec
- Resulting test case: Fig. **??** shows the test case from Proof2Test, which calls `square_root` with input argument `n = 11`.
- Comment: the test is useful as it is able to show a concrete trace that leads to the same contract violation as in the proof; during the execution of the test, initially, `x1= 0` and `x2= 11`; at the first iteration, the program assigns `mid` with 5 (line 19); the condition of the `else` branch is true (`mid * mid ≠ n`); the condition of the `then` branch at line 24 is true (`mid * mid > n`), thus the program assigns `x1` with the value of `mid` (line 25); at the beginning of the second iteration, the loop invariant `valid_result` is evaluated as false (in the right-hand side of the `or` clause, `x1<x2` is true but `x1*x1<n` is false), causing an exception of the invariant violation.
- Possible fixes: 1 valid fixes out of 9 candidate fixes
- fixing time: 1.43 minutes




(d)

```
1
2  square_root_ID_3 (n: INTEGER): TUPLE [x: INTEGER; y: INTEGER]
3       -- 'x' and 'y' are two approximate square roots of 'n'
4     require
5       valid_n: n ≥ 0
6  not ((n) > (0))
7     local
8       x1, x2, mid: INTEGER
9     do
10
11      from
12        x1 := 0
13        x2 := n
14      invariant
```

```
15        valid_result: (x1 = x2 and x1 * x1 = n) or (x1 < x2 and x1 * x1 < n and x2
              * x2 ≥ n)
16      until
17        x2 − x1 ≤ 1or x1 = x2
18      loop
19        mid := (x1 + x2) // 2-- integer division
20        if mid * mid = n then
21          x1 := mid
22          x2 := mid
23        else
24          if mid * mid > n then
25            x1 := mid
26          else
27            x2 := mid
28          end
29        end
30      variant
31        x2 − x1
32      end
33      Result := [x1, x2]
34    ensure
35      valid_result: (Result.x = Result.y and Result.x * Result.x = n)
36        or (Result.x + 1= Result.y and Result.x * Result.x < n and Result.y *
              Result.y ≥ n)
37    end
```

## 3 Examples of EiffelBase2

This section describes the experiment of porting existing bugs in the "base" library to the fully-verified library "base2". First, for classes present in "base2" we found the corresponding classes in "base" (Table 1). For each "base" class we ran a 1-hour-long session of AutoTest (running on an Intel i7-7700HQ 3.60GHz processor). This produced 285 failing tests and 79 unresolved tests. We manually analyzed each of 285 failing tests and managed to extract and port 12 faults from "base" to "base2". The section will continue with the detailed descriptions of each fault.

### 3.1 `ARRAY.index_set` for empty arrays

Fig. 50 lists the code of a faulty function in the `ARRAY` class. The `ARRAY` class allows setting arbitrary boundaries (`lower` and `upper`) and empty arrays are represented simply by setting `lower = upper + 1`. All of the following (`lower`, `upper`) pairs are valid and represent an empty array: $(1, 0)$, $(10, 9)$, $(-2, -3)$. On the other hand, the creation procedure `INTEGER_INTERVAL.make` sets specific values $(1, 0)$ for the empty interval case (Fig. 51). This implementation is inconsistent with the `same_bounds` postcondition of `ARRAY.index_set`. Calling `ARRAY.index_set` for empty arrays not having `lower = 1` will result in the violation of `same_bounds`.

**Table 1.** Corresponding classes of "base" library in "base2"

| base | base2 |
|---|---|
| ARRAY | V_ARRAY |
| INDEXABLE_ITERATION_CURSOR | V_ARRAY_ITERATOR |
| ARRAYED_LIST | V_ARRAYED_LIST |
| ARRAYED_LIST_CURSOR | V_ARRAYED_LIST_CURSOR |
| ARRAY2 | V_ARRAY2 |
| CELL | V_CELL |
| LINKABLE | V_LINKABLE |
| LINKED_LIST | V_LINKED_LIST |
| LINKED_LIST_ITERATION_CURSOR | V_LINKED_LIST_ITERATOR |
| LINKED_STACK | V_LINKED_STACK |
| LINKED_QUEUE | V_LINKED_QUEUE |
| HASH_TABLE | V_HASH_TABLE |
| HASH_TABLE_ITERATION_CURSOR | V_HASH_TABLE_ITERATOR |
| RANDOM | V_RANDOM |

```
index_set: INTEGER_INTERVAL
    -- Range of acceptable indexes
  do
    create Result.make (lower, upper)
  ensure then
    same_count: Result.count = count
    same_bounds: (Result.lower = lower) and (Result.upper = upper)
  end
```

**Fig. 50.** Function ARRAY.index_set from "base"

```
make (min_index, max_index: INTEGER)
    -- Set up interval to have bounds ‘min_index’ and
    -- ‘max_index’ (empty if ‘min_index’ > ‘max_index’)
  do
    lower_defined := True
    upper_defined := True
    if min_index ≤ max_index then
      lower_internal := min_index
      upper_internal := max_index
    else
      lower_internal := 1
      upper_internal := 0
    end
  ensure
    lower_defined: lower_defined
    upper_defined: upper_defined
    set_if_non_empty: (min_index ≤ max_index) implies
        ((lower = min_index) and (upper = max_index))
    empty_if_not_in_order: (min_index > max_index) implies is_empty
  end
```

**Fig. 51.** Creation procedure `INTEGER_INTERVAL.make` from “base”

The function was ported to “base2” literally by copying `index_set` into the body of `V_ARRAY`. To fix the bug and verify the function we relaxed the postcondition to specify the specific bounds only in the non-empty cases. To fully verify the fix we added missing contracts to the `V_ARRAY.count` function and implemented a verified but simplified version of the `INTEGER_INTERVAL` class (called `V_INTEGER_INTERVAL`). Other fixes can include the strengthening of the contracts to highlight that in

```
index_set: V_INTEGER_INTERVAL
  do
    create Result.make (lower, upper)
  ensure
    not_void: Result ≠ Void
    same_count_if_not_empty: Result.count = count
    same_bounds_if_not_empty: not is_empty
              implies ((Result.lower = lower) and (Result.upper = upper))
  end
```

**Fig. 52.** Fixed `index_set` injected into “base2”

case of an empty array the integer interval will have `lower = 1` and `upper = 0`.

This bug could be a result of a simple programmer mistake. Different classes have similar concepts (`lower` and `upper`) which are implemented slightly differently.

- fixing time: 2.42 minutes
- valid fixes out of 267 candidate fixes.

```

bugged_ID_258: INTEGER_INTERVAL_1
      -- Range of acceptable indexes
    note
      original_name: index_set
    require
        -- not is_empty
      -- not (lower > upper)
      lower ≥ −10 and lower ≤ 10
      upper ≥ −10 and upper ≤ 10
not ((lower) > (upper))
    do

      create Result.make (lower, upper)
    ensure -- from READABLE_INDEXABLE
      not_void: Result ≠ Void
      same_count: Result.count = count -- or Result = 0
      same_bounds: ((Result.lower = lower) and (Result.upper = upper)) -- bug is here
        -- Empty ‘INTEGER_INTERVAL‘ always has ‘lower = 1‘ and ‘upper = 0‘
        -- while this is not necessary so for empty ‘ARRAY‘
        -- Examples:
        -- lower | upper
        -- ================
        -- -100 | -101
    end

bugged_ID_259: INTEGER_INTERVAL_1
      -- Range of acceptable indexes
    note
      original_name: index_set
    require
        -- not is_empty
      -- not (lower > upper)
      lower ≥ −10 and lower ≤ 10
      upper ≥ −10 and upper ≤ 10
    do

      create Result.make (lower, upper)
    ensure -- from READABLE_INDEXABLE
      not_void: Result ≠ Void
      same_count: Result.count = count -- or Result = 0
same_bounds:not ((lower) > (upper))implies ( ((Result.lower = lower) and (Result.upper = upper))) -- bug is here
```

```
        -- Empty ‘INTEGER_INTERVAL‘ always has ‘lower = 1‘ and ‘upper = 0‘
        -- while this is not necessary so for empty ‘ARRAY‘
        -- Examples:
        -- lower | upper
        -- ================
        -- -100 | -101
    end

bugged_ID_260: INTEGER_INTERVAL_1
      -- Range of acceptable indexes
    note
      original_name: index_set
    require
        -- not is_empty
      -- not (lower > upper)
      lower ≥ −10 and lower ≤ 10
      upper ≥ −10 and upper ≤ 10
not ((lower) ≥ (upper))
    do

      create Result.make (lower, upper)
    ensure -- from READABLE_INDEXABLE
      not_void: Result ≠ Void
      same_count: Result.count = count -- or Result = 0
      same_bounds: ((Result.lower = lower) and (Result.upper = upper)) -- bug
          is here
        -- Empty ‘INTEGER_INTERVAL‘ always has ‘lower = 1‘ and ‘upper = 0‘
        -- while this is not necessary so for empty ‘ARRAY‘
        -- Examples:
        -- lower | upper
        -- ================
        -- -100 | -101
    end

bugged_ID_261: INTEGER_INTERVAL_1
      -- Range of acceptable indexes
    note
      original_name: index_set
    require
        -- not is_empty
      -- not (lower > upper)
      lower ≥ −10 and lower ≤ 10
      upper ≥ −10 and upper ≤ 10
    do

      create Result.make (lower, upper)
    ensure -- from READABLE_INDEXABLE
      not_void: Result ≠ Void
      same_count: Result.count = count -- or Result = 0
```

```
same_bounds:not ((lower) ≥ (upper))implies ( ((Result.lower = lower) and (Result.upper = upper))) -- bug is here
        -- Empty ‘INTEGER_INTERVAL‘ always has ‘lower = 1‘ and ‘upper = 0‘
        -- while this is not necessary so for empty ‘ARRAY‘
        -- Examples:
        -- lower | upper
        -- ================
        -- -100 | -101
    end

bugged_ID_266: INTEGER_INTERVAL_1
      -- Range of acceptable indexes
    note
      original_name: index_set
    require
        -- not is_empty
      -- not (lower > upper)
      lower ≥ −10 and lower ≤ 10
      upper ≥ −10 and upper ≤ 10
not (lower > upper)
    do

      create Result.make (lower, upper)
    ensure -- from READABLE_INDEXABLE
      not_void: Result ≠ Void
      same_count: Result.count = count -- or Result = 0
      same_bounds: ((Result.lower = lower) and (Result.upper = upper)) -- bug is here
        -- Empty ‘INTEGER_INTERVAL‘ always has ‘lower = 1‘ and ‘upper = 0‘
        -- while this is not necessary so for empty ‘ARRAY‘
        -- Examples:
        -- lower | upper
        -- ================
        -- -100 | -101
    end

bugged_ID_267: INTEGER_INTERVAL_1
      -- Range of acceptable indexes
    note
      original_name: index_set
    require
        -- not is_empty
      -- not (lower > upper)
      lower ≥ −10 and lower ≤ 10
      upper ≥ −10 and upper ≤ 10
    do

      create Result.make (lower, upper)
    ensure -- from READABLE_INDEXABLE
      not_void: Result ≠ Void
```

```
      same_count: Result.count = count -- or Result = 0
same_bounds:not (lower > upper)implies ( ((Result.lower = lower) and (Result.upper = upper))) -- bug is here
        -- Empty ‘INTEGER_INTERVAL‘ always has ‘lower = 1‘ and ‘upper = 0‘
        -- while this is not necessary so for empty ‘ARRAY‘
        -- Examples:
        -- lower | upper
        -- ================
        -- -100 | -101
    end
--
    ------------------------------------------------------------------------------
```

### 3.2 ARRAYED_LIST.merge_right with shared area_v2 wipes out both lists

Fig. 53 presents a manually composed test which highlights the failure found by AutoTest in ARRAYED_LIST.merge_right. The execution ends with the violation of

```
  test
    local
      special: SPECIAL [INTEGER]
      array: ARRAY [INTEGER]
      a, b: ARRAYED_LIST [INTEGER]
    do
      create special.make_filled (0, 10)
      special.keep_head (5)
        -- special now has: count = 5, capacity = 10
      create array.make_from_special (special)
      create a.make_from_array (array)
      create b.make_from_array (array)
      a.merge_right (b)
    end
```

**Fig. 53.** Failing test for ARRAYED_LIST.merge_right in “base2”

the new_count postcondition in the merge_right procedure (Fig. 54). In this procedure, when initially area_v2= other.area_v2 (both are references to the same object) and area's capacity is enough to include both lists (l_new_count $\leq$ area_v2.capacity), then the command other.wipe_out empties the same shared area, so the resulting count is just 0. The provided example test (Fig. 53) satisfies these conditions. This procedure can be ported to a simplified version of V_ARRAYED_LIST called SIMPLE_ARRAYED_LIST (Fig. 55). Running verification on this procedure reports that postcondition new_count may be violated. To explore the code further, notice that removing the other.wipe_out line allows the postcondition to verify

```
merge_right (other: ARRAYED_LIST [G])
    -- Merge 'other' into current structure after cursor.
  require -- from DYNAMIC_CHAIN
    extendible: extendible
    not_after: not after
    other_exists: other ≠ Void
    not_current: other ≠ Current
  local
    l_new_count, l_old_count: INTEGER_32
  do
    if not other.is_empty then
      l_old_count := count
      l_new_count := l_old_count + other.count
      if l_new_count > area_v2.capacity then
        area_v2 := area_v2.aliased_resized_area (l_new_count)
      end
      area_v2.insert_data (other.area_v2, 0, index, other.count)
      other.wipe_out
    end
  ensure -- from DYNAMIC_CHAIN
    new_count: count = old count + old other.count
    same_index: index = old index
    other_is_empty: other.is_empty
  end
```

**Fig. 54.** Flat view of ARRAYED_LIST.merge_right from "base"

```
bugged (other: like Current; index: INTEGER)
    note
      explicit: wrapping
    require
      different_other: Current ≠ other
      both_wrapped: area.is_wrapped and other.area.is_wrapped
      not_after: 0 ≤ index and index ≤ area.count
    local
      l_new_count: INTEGER
      count, other_count: INTEGER
    do
      count := area.count
      other_count := other.area.count
      l_new_count := count + other_count
      unwrap
      if l_new_count > count then
        area := area.aliased_resized_area_with_default2 ((\{G\}).default, l_new_count)
      end
      area.move_data (index, index + other_count, count − index)
      area.copy_data (other.area, 0, index, other_count)
      sequence := area.sequence
      wrap
      other.wipe_out
    ensure
      modify (area)
      modify (Current)
      modify (other)
      modify (other.area)
      new_count: area.count = old area.count + old other.area.count
      other_is_empty: other.area.count = 0
    end
```

**Fig. 55.** Procedure `SIMPLE_ARRAYED_LIST.merge_right` using classes from "base2"

but breaks the `other_is_empty` postcondition. There can be several fixes. One straightforward fix is to specify explicitly that the `area`s should be different (Fig. 56). Adding the additional `different_areas` precondition allows the fea-

```
fixed_1 (other: like Current; index: INTEGER)
    note
      explicit: wrapping
    require
      different_other: Current ≠ other
      both_wrapped: area.is_wrapped and other.area.is_wrapped
      not_after: 0 ≤ index and index ≤ area.count
      different_areas: area ≠ other.area
    -- Same lines omitted
    end
```

**Fig. 56.** The first fix for `SIMPLE_ARRAYED_LIST.merge_right`

ture to fully verify. Another fix is to properly use the ownership mechanism of AutoProof. For simplicity, now we will specify ownership in the precondition (Fig. 57). Notice that the old precondition `both_wrapped` is now not present.

```
  fixed_2 (other: like Current; index: INTEGER)
    note
      explicit: wrapping
    require
      different_other: Current ≠ other
      not_after: 0 ≤ index and index ≤ area.count
      current_owns: owns ~ create {MML_SET[ANY]}.singleton (area)
      other_owns: other.owns ~ create {MML_SET[ANY]}.singleton (other.area)
    -- Same lines omitted
    end
```

**Fig. 57.** The second fix for `SIMPLE_ARRAYED_LIST.merge_right`

Since lists (`Current` and `other`) own their `area`s, each of `area`s have an owner. Object cannot be both `wrapped` and owned by a `closed` object. Since objects can have only one `owner`, it can be inferred that `area`s are different, so the two new preconditions `current_owns` and `other_owns` are imply `area` ≠ `other.area` which allows the procedure to fully verify. The `owns` set is constant in this class, so it is more convenient to specify ownership in the class invariant (Fig. 58). In the third fix we move the ownership preconditions to the class invariant `owns_def`. Note that `sequence_equal` invariant is not specific to the fix and was present in all other examples. The bug probably was introduced simple because the pro-

```
  fixed_3 (other: like Current; index: INTEGER)
    note
      explicit: wrapping
    require
      different_other: Current ≠ other
      not_after: 0 ≤ index and index ≤ area.count
    -- Same lines omitted
    end
invariant
  owns_def: owns = create {MML_SET[ANY]}.singleton (area)
  sequence_equal: sequence ~ area.sequence
```

**Fig. 58.** The third fix for `SIMPLE_ARRAYED_LIST.merge_right`

grammer simply overlooked this rare case. It is easy to overlook the possible shared usage of `areas` in "base" classes since not all features clearly specify if their contents are copied into new instances or if they are fully reused (as in the `ARRAY.make_from_special`). However, AutoProof's ownership mechanism forces programmers to make this explicit in contracts.

- fixing time: 9.38 minutes
- fixes

```

bugged_ID_87 (other: like Current; index: INTEGER)
    note
      explicit: wrapping
    require
      different_other: Current ≠ other
      area.is_wrapped and other.area.is_wrapped
      not_after: 0 ≤ index and index ≤ area.count
    local
      l_new_count: INTEGER
      count, other_count: INTEGER
    do

      count := area.count
      other_count := other.area.count
      l_new_count := count + other_count
      unwrap
      if l_new_count > count then
        area := area.aliased_resized_area_with_default2 (({INTEGER}).default,
             l_new_count)
      end
      area.move_data (index, index + other_count, count − index)
      area.copy_data (other.area, 0, index, other_count)
      sequence := area.sequence
      wrap
```

```
      other.wipe_out
    ensure
      modify (area)
      modify (Current)
      modify (other)
      modify (other.area)
new_count:not (not (other = Void))implies ( area.count = old area.count + old
     other.area.count)
      other_is_empty: other.area.count = 0
    end

bugged_ID_91 (other: like Current; index: INTEGER)
    note
      explicit: wrapping
    require
      different_other: Current ≠ other
      area.is_wrapped and other.area.is_wrapped
      not_after: 0 ≤ index and index ≤ area.count
    local
      l_new_count: INTEGER
      count, other_count: INTEGER
    do

      count := area.count
      other_count := other.area.count
      l_new_count := count + other_count
      unwrap
      if l_new_count > count then
        area := area.aliased_resized_area_with_default2 (({INTEGER}).default,
              l_new_count)
      end
      area.move_data (index, index + other_count, count − index)
      area.copy_data (other.area, 0, index, other_count)
      sequence := area.sequence
      wrap
      other.wipe_out
    ensure
      modify (area)
      modify (Current)
      modify (other)
      modify (other.area)
new_count:not (not (other = Current))implies ( area.count = old area.count +
     old other.area.count)
      other_is_empty: other.area.count = 0
    end

bugged_ID_95 (other: like Current; index: INTEGER)
    note
      explicit: wrapping
    require
```

```
      different_other: Current ≠ other
      area.is_wrapped and other.area.is_wrapped
      not_after: 0 ≤ index and index ≤ area.count
    local
      l_new_count: INTEGER
      count, other_count: INTEGER
    do

      count := area.count
      other_count := other.area.count
      l_new_count := count + other_count
      unwrap
      if l_new_count > count then
        area := area.aliased_resized_area_with_default2 (({INTEGER}).default,
             l_new_count)
      end
      area.move_data (index, index + other_count, count − index)
      area.copy_data (other.area, 0, index, other_count)
      sequence := area.sequence
      wrap
      other.wipe_out
    ensure
      modify (area)
      modify (Current)
      modify (other)
      modify (other.area)
new_count:not ((other = Void) or else (not (other = Current)))implies ( area.
     count = old area.count + old other.area.count)
      other_is_empty: other.area.count = 0
    end

bugged_ID_97 (other: like Current; index: INTEGER)
    note
      explicit: wrapping
    require
      different_other: Current ≠ other
      area.is_wrapped and other.area.is_wrapped
      not_after: 0 ≤ index and index ≤ area.count
    local
      l_new_count: INTEGER
      count, other_count: INTEGER
    do

      count := area.count
      other_count := other.area.count
      l_new_count := count + other_count
      unwrap
      if l_new_count > count then
        area := area.aliased_resized_area_with_default2 (({INTEGER}).default,
             l_new_count)
```

```
      end
      area.move_data (index, index + other_count, count − index)
      area.copy_data (other.area, 0, index, other_count)
      sequence := area.sequence
      wrap
      other.wipe_out
    ensure
      modify (area)
      modify (Current)
      modify (other)
      modify (other.area)
new_count:not ((not (other = Void)) or else (other = Current))implies ( area.
     count = old area.count + old other.area.count)
      other_is_empty: other.area.count = 0
    end

bugged_ID_99 (other: like Current; index: INTEGER)
    note
      explicit: wrapping
    require
      different_other: Current ≠ other
      area.is_wrapped and other.area.is_wrapped
      not_after: 0 ≤ index and index ≤ area.count
    local
      l_new_count: INTEGER
      count, other_count: INTEGER
    do

      count := area.count
      other_count := other.area.count
      l_new_count := count + other_count
      unwrap
      if l_new_count > count then
        area := area.aliased_resized_area_with_default2 (({INTEGER}).default,
              l_new_count)
      end
      area.move_data (index, index + other_count, count − index)
      area.copy_data (other.area, 0, index, other_count)
      sequence := area.sequence
      wrap
      other.wipe_out
    ensure
      modify (area)
      modify (Current)
      modify (other)
      modify (other.area)
new_count:not ((not (other = Void)) or else (not (other = Current)))implies (
     area.count = old area.count + old other.area.count)
      other_is_empty: other.area.count = 0
    end
```

```
bugged_ID_105 (other: like Current; index: INTEGER)
    note
      explicit: wrapping
    require
      different_other: Current ≠ other
      area.is_wrapped and other.area.is_wrapped
      not_after: 0 ≤ index and index ≤ area.count
    local
      l_new_count: INTEGER
      count, other_count: INTEGER
    do

      count := area.count
      other_count := other.area.count
      l_new_count := count + other_count
      unwrap
      if l_new_count > count then
        area := area.aliased_resized_area_with_default2 (({INTEGER}).default,
             l_new_count)
      end
      area.move_data (index, index + other_count, count − index)
      area.copy_data (other.area, 0, index, other_count)
      sequence := area.sequence
      wrap
      other.wipe_out
    ensure
      modify (area)
      modify (Current)
      modify (other)
      modify (other.area)
new_count:not ((index) ≥ (0))implies ( area.count = old area.count + old other.
     area.count)
      other_is_empty: other.area.count = 0
    end

bugged_ID_115 (other: like Current; index: INTEGER)
    note
      explicit: wrapping
    require
      different_other: Current ≠ other
      area.is_wrapped and other.area.is_wrapped
      not_after: 0 ≤ index and index ≤ area.count
    local
      l_new_count: INTEGER
      count, other_count: INTEGER
    do

      count := area.count
      other_count := other.area.count
```

```
      l_new_count := count + other_count
      unwrap
      if l_new_count > count then
        area := area.aliased_resized_area_with_default2 (({INTEGER}).default,
             l_new_count)
      end
      area.move_data (index, index + other_count, count - index)
      area.copy_data (other.area, 0, index, other_count)
      sequence := area.sequence
      wrap
      other.wipe_out
    ensure
      modify (area)
      modify (Current)
      modify (other)
      modify (other.area)
new_count:not ((((((((((Current))).area))).sequence))).count = 0)implies ( area.
     count = old area.count + old other.area.count)
      other_is_empty: other.area.count = 0
    end

bugged_ID_117 (other: like Current; index: INTEGER)
    note
      explicit: wrapping
    require
      different_other: Current ≠ other
      area.is_wrapped and other.area.is_wrapped
      not_after: 0 ≤ index and index ≤ area.count
    local
      l_new_count: INTEGER
      count, other_count: INTEGER
    do

      count := area.count
      other_count := other.area.count
      l_new_count := count + other_count
      unwrap
      if l_new_count > count then
        area := area.aliased_resized_area_with_default2 (({INTEGER}).default,
             l_new_count)
      end
      area.move_data (index, index + other_count, count - index)
      area.copy_data (other.area, 0, index, other_count)
      sequence := area.sequence
      wrap
      other.wipe_out
    ensure
      modify (area)
      modify (Current)
      modify (other)
```

```
      modify (other.area)
new_count:not (((((((other))).sequence))).count = 0)implies ( area.count = old area.count + old other.area.count)
      other_is_empty: other.area.count = 0
    end

bugged_ID_121 (other: like Current; index: INTEGER)
    note
      explicit: wrapping
    require
      different_other: Current ≠ other
      area.is_wrapped and other.area.is_wrapped
      not_after: 0 ≤ index and index ≤ area.count
    local
      l_new_count: INTEGER
      count, other_count: INTEGER
    do

      count := area.count
      other_count := other.area.count
      l_new_count := count + other_count
      unwrap
      if l_new_count > count then
        area := area.aliased_resized_area_with_default2 (({INTEGER}).default, l_new_count)
      end
      area.move_data (index, index + other_count, count − index)
      area.copy_data (other.area, 0, index, other_count)
      sequence := area.sequence
      wrap
      other.wipe_out
    ensure
      modify (area)
      modify (Current)
      modify (other)
      modify (other.area)
new_count:not ((((((((((other))).area))).sequence))).count = 0)implies ( area.count = old area.count + old other.area.count)
      other_is_empty: other.area.count = 0
    end
```

### 3.3 ARRAY.force below empty array

Fig. 59 presents a failing test for procedure ARRAY.force. Executing this test results in ARRAY.force's inserted postcondition violation (Fig. 60). The bug is located inside the if empty_area then block. When forcing into an array with empty area, area needs to be grown and the new value needs to be written to the area at the specified index. The code grows the area and saves the new

```
array_force_below_test
  local
    array: ARRAY [INTEGER]
  do
    create array.make_filled (0, 10, 9)
    array.force (1, 5)
  end
```

**Fig. 59.** Failing test for `ARRAY.force`

value at the end of the area (`area.extend(v)`) but this is not always correct. If `i` <`lower`, the new value should be recorded not as the last element of `area` but as the first element of `area`. To simplify the discussion we extracted the relevant code into a special case procedure for "base2" (Fig. 61). Notice the added `is_empty` precondition which signifies the special case which is covered in this version. Verification fails for the same postcondition `inserted` which was failing in the test. The added broader postcondition `sequence_effect` is also failing to verify. To fix the bug we replace the `if not l_increased_by_one then` instruction to handle the problematic case of forcing the new value below the bounds of the array (Fig. 62). The fixed version fully verifies. The programmer might have simply overlooked the rare case of indices below `lower`.

### 3.4 `ARRAY.remove_head` postcondition when removing all items

Fig. 63 shows the procedure `ARRAY.remove_head`. Calling this procedure with `n` > `count` leads to the violation of the `same_upper` postcondition. The postcondition states that `upper` should not change, however the `then` part of the `if n > count` construction changes `upper` (`upper` := `lower` − `1`). Fig. 64 presents `remove_head` ported to "base2". The instruction `upper` := `lower` − `1` had to be changed into `lower` := `1`; `upper` := `0` to satisfy `V_ARRAY`'s invariant. Verification fails because the postcondition `same_upper` can be violated. The version from "base" can be fixed by replacing `upper` := `lower` − `1` with `lower` := `upper` + `1` which will satisfy the postcondition. However, this fix is not applicable in the case of "base2" because the class invariant requires `lower` = `1` when the array is empty. The second fix is to improve the postcondition (Fig. 65). The replaced precondition `same_upper_if_not_empty` specifies that `upper` is not changed only if the resulting array is not empty. The feature verifies fully with this fix.

### 3.5 `INDEXABLE_ITERATION_CURSOR.forth` does not always increment `cursor_index`

Fig. 66 shows two connected features `forth` and `cursor_index`. AutoFix has found several counterexamples in which the postcondition `cursor_index_advanced` did not hold (Table 2). Indeed, with these values the postcondition does not hold and `cursor_index` does not advance. The fix (Fig. 67) is based on clearly specifying

```
force (v: like item; i: INTEGER_32)
    -- Assign item ‘v’ to ‘i’-th entry.
    -- Resize the array if ‘i’ falls out of currently defined bounds; preserve existing items.
    -- In void-safe mode, if ({G}).has_default does not hold, then you can only insert between
    -- ‘lower - 1’ or ‘upper + 1’ position in the ARRAY.
  require
    has_default_if_too_low: (i <lower − 1and lower ≠ {like lower}.min_value) implies ({G}).has_default
    has_default_if_too_high: (i > upper + 1and upper ≠ {like upper}.max_value) implies ({G}).has_default
  local
    old_size, new_size: INTEGER_32
    new_lower, new_upper: INTEGER_32
    l_count, l_offset: INTEGER_32
    l_increased_by_one: BOOLEAN
  do
    new_lower := lower.min (i)
    new_upper := upper.max (i)
    new_size := new_upper − new_lower + 1
    l_increased_by_one := (i = upper + 1) or (i = lower − 1)
    if empty_area then
      make_empty_area (new_size.max (additional_space))
      if not l_increased_by_one then
        area.fill_with (({G}).default, 0, new_size − 2)
      end
      area.extend (v)
    else
      ... -- irrelevant lines omitted
    end
    lower := new_lower
    upper := new_upper
  ensure
    inserted: item (i) = v
    higher_count: count ≥ old count
    lower_set: lower = (old lower).min (i)
    upper_set: upper = (old upper).max (i)
  end
```

**Fig. 60.** Procedure ARRAY.force from “base”

```
force (v: like item; i: INTEGER)
  require
    is_empty
  local
    old_size, new_size: INTEGER_32
    new_lower, new_upper: INTEGER_32
    l_count, l_offset: INTEGER_32
    l_increased_by_one: BOOLEAN
  do
    new_lower := lower.min (i)
    new_upper := upper.max (i)
    new_size := new_upper - new_lower + 1
    l_increased_by_one := (i = upper + 1) or (i = lower - 1)
    area := area.aliased_resized_area (new_size)
    if not l_increased_by_one then
      area.fill_with ((\{G\}).default, 0, new_size - 2)
    end
    area.extend (v)
    lower := new_lower
    upper := new_upper
  ensure
    inserted: item (i) = v
    higher_count: count ≥ old count
    lower_set: lower = (old lower).min (i)
    upper_set: upper = (old upper).max (i)
    modify_model (["sequence", "lower_"], Current)
  end
```

**Fig. 61.** Special case ARRAY.force ported to "base2"

**Table 2.** Counterexamples for INDEXABLE_ITERATION_CURSOR.forth

| is_reversed | target_index | step | first_index |
|---|---|---|---|
| True | 1 | 1 | 0 |
| True | 1 | 2 | 0 |
| True | 1 | 7 | 0 |
| True | 3 | 3 | 0 |
| True | 1 | 100 | 0 |

```
fixed (v: like item; i: INTEGER)
  require -- Same contracts omitted
  local
    old_size, new_size: INTEGER_32
    new_lower, new_upper: INTEGER_32
    l_count, l_offset: INTEGER_32
    l_increased_by_one: BOOLEAN
  do
    new_lower := lower.min (i)
    new_upper := upper.max (i)
    new_size := new_upper - new_lower + 1
    l_increased_by_one := (i = upper + 1) or (i = lower - 1)
    area := area.aliased_resized_area (new_size)
    if not l_increased_by_one then
      if i < lower then
        area.extend (v)
        area.fill_with ((\{G\}).default, 1, new_size - 1)
      else
        area.fill_with ((\{G\}).default, 0, new_size - 2)
        area.extend (v)
      end
    else
      area.extend (v)
    end
    lower := new_lower
    upper := new_upper
  ensure -- Same contracts omitted
  end
```

**Fig. 62.** Special case ARRAY.force fixed in "base2"

```
remove_head (n: INTEGER_32)
    -- Remove first ‘n’ items;
    -- if ‘n’ > count, remove all.
  require
    n_non_negative: n ≥ 0
  do
    if n > count then
      upper := lower − 1
      area := area.aliased_resized_area (0)
    else
      keep_tail (count − n)
    end
  ensure
    new_count: count = (old count − n).max (0)
    same_upper: upper = old upper
  end
```

**Fig. 63.** Original ARRAY.remove_head from “base”

```
bugged (n: INTEGER_32)
  note
    explicit: wrapping
  require
    n_non_negative: n ≥ 0
  do
    if n > count then
      unwrap
      lower := 1
      upper := 0
      area := area.aliased_resized_area (0)
      wrap
    else
      keep_tail (count − n)
    end
  ensure
    new_count: count = (old count − n).max (0)
    same_upper: upper = old upper
  end
```

**Fig. 64.** Buggy ARRAY.remove_head ported to “base2”

```
fixed (n: INTEGER_32)
  -- Same lines omitted
  ensure
    new_count: count = (old count - n).max (0)
    same_upper_if_not_empty: not is_empty implies upper = old upper
  end
```

**Fig. 65.** Fixed ARRAY.remove_head ported to "base2"

```
forth
    -- Move to next position.
  require -- from ITERATION_CURSOR
    valid_position: not after
  do
    if is_reversed then
      target_index := target_index - step
    else
      target_index := target_index + step
    end
  ensure then
    cursor_index_advanced: cursor_index = old cursor_index + 1
  end
cursor_index: INTEGER_32
    -- Index position of cursor in the iteration.
  require
    is_valid: is_valid
  do
    Result := ((target_index - first_index).abs + step - 1) // step + 1
  ensure
    positive_index: Result ≥ 0
  end
```

**Fig. 66.** Original forth and cursor_index features from "base" INDEXABLE_ITERATION_CURSOR

the relationship between `is_reversed` and `first_index` with `last_index` in the class invariant. With such invariant in place the other features can be correctly implemented for each case. The fix includes strengthening the class invariant and updating two features. After implementing the fixes both features verify.

### 3.6 indexable_iteration_cursor_bug_1.log

– fixing time: 2.43 minutes

```

buggy_ID_264
      -- Move to next position.
    require -- from ITERATION_CURSOR
      valid_position: not (lower ≤ target_index and target_index ≤ upper) and lower ≤ upper
        -- not before
      target_index ≥ 0 and target_index ≤ 10
      lower ≥ 0 and lower ≤ 10
      upper ≥ 0 and upper ≤ 10
      first_index ≥ 0 and first_index ≤ 10
not ((is_reversed) or else (before))
    do

      if is_reversed then
        target_index := target_index − step
      else
        target_index := target_index + step
      end
    ensure
        -- cursor_index_advanced: cursor_index_buggy = old cursor_index_buggy + 1
      cursor_index_advanced: (cursor_index_fixed = old cursor_index_fixed + 1)
    end

buggy_ID_265
      -- Move to next position.
    require -- from ITERATION_CURSOR
      valid_position: not (lower ≤ target_index and target_index ≤ upper) and lower ≤ upper
        -- not before
      target_index ≥ 0 and target_index ≤ 10
      lower ≥ 0 and lower ≤ 10
      upper ≥ 0 and upper ≤ 10
      first_index ≥ 0 and first_index ≤ 10
    do

      if is_reversed then
        target_index := target_index − step
```

```
  fixed
    note
      explicit: wrapping
    require
      valid_position: not after
    do
      if before then
        unwrap; target_index := first_index; wrap
      else
        if is_reversed then
          unwrap; target_index := target_index − step; wrap
        else
          unwrap; target_index := target_index + step; wrap
        end
      end
    ensure
      modify_model ("target_index", Current)
      cursor_index_advanced: cursor_index_fixed = old cursor_index_fixed + 1
    end
  cursor_index_fixed: INTEGER_32
    note
      status: pure
    do
      if is_reversed then
        if target_index ≤ first_index then
          Result := (first_index − target_index + step − 1) // step + 1
        end
      else
        if target_index ≥ first_index then
          Result := (target_index − first_index + step − 1) // step + 1
        end
      end
    ensure
      positive_index: Result ≥ 0
      before_is_zero: before implies Result = 0
      reversed_in: is_reversed and target_index ≤ first_index implies Result = (
          first_index − target_index + step − 1) // step + 1
      not_reversed_in: not is_reversed and target_index ≥ first_index implies Result
          = (target_index − first_index + step − 1) // step + 1
    end
invariant
  first_greater_when_reversed: is_reversed implies first_index ≥ last_index
  last_greater_when_not_reversed: not is_reversed implies last_index ≥ first_index
```

**Fig. 67.** Fixed `forth` and `cursor_index` features using "base2" classes

```
      else
        target_index := target_index + step
      end
    ensure
        -- cursor_index_advanced: cursor_index_buggy = old cursor_index_buggy + 1
cursor_index_advanced:not ((is_reversed) or else (before))implies ( ( cursor_index_fixed = old cursor_index_fixed + 1))
    end

buggy_ID_268
      -- Move to next position.
    require -- from ITERATION_CURSOR
      valid_position: not (lower ≤ target_index and target_index ≤ upper) and lower ≤ upper
        -- not before
      target_index ≥ 0and target_index ≤ 10
      lower ≥ 0and lower ≤ 10
      upper ≥ 0and upper ≤ 10
      first_index ≥ 0and first_index ≤ 10
not ((not (is_reversed)) or else (before))
    do

      if is_reversed then
        target_index := target_index − step
      else
        target_index := target_index + step
      end
    ensure
        -- cursor_index_advanced: cursor_index_buggy = old cursor_index_buggy + 1
      cursor_index_advanced: (cursor_index_fixed = old cursor_index_fixed + 1)
    end

buggy_ID_269
      -- Move to next position.
    require -- from ITERATION_CURSOR
      valid_position: not (lower ≤ target_index and target_index ≤ upper) and lower ≤ upper
        -- not before
      target_index ≥ 0and target_index ≤ 10
      lower ≥ 0and lower ≤ 10
      upper ≥ 0and upper ≤ 10
      first_index ≥ 0and first_index ≤ 10
    do

      if is_reversed then
        target_index := target_index − step
      else
```

```
        target_index := target_index + step
      end
    ensure
        -- cursor_index_advanced: cursor_index_buggy = old cursor_index_buggy + 1
cursor_index_advanced:not ((not (is_reversed)) or else (before))implies ( (cursor_index_fixed = old cursor_index_fixed + 1))
    end

buggy_ID_273
      -- Move to next position.
    require -- from ITERATION_CURSOR
      valid_position: not (lower ≤ target_index and target_index ≤ upper) and lower ≤ upper
        -- not before
      target_index ≥ 0and target_index ≤ 10
      lower ≥ 0and lower ≤ 10
      upper ≥ 0and upper ≤ 10
      first_index ≥ 0and first_index ≤ 10
    do

      if is_reversed then
        target_index := target_index − step
      else
        target_index := target_index + step
      end
    ensure
        -- cursor_index_advanced: cursor_index_buggy = old cursor_index_buggy + 1
cursor_index_advanced:not (step = 1)implies ( (cursor_index_fixed = old cursor_index_fixed + 1))
    end

buggy_ID_278
      -- Move to next position.
    require -- from ITERATION_CURSOR
      valid_position: not (lower ≤ target_index and target_index ≤ upper) and lower ≤ upper
        -- not before
      target_index ≥ 0and target_index ≤ 10
      lower ≥ 0and lower ≤ 10
      upper ≥ 0and upper ≤ 10
      first_index ≥ 0and first_index ≤ 10
not (target_index ≠ first_index)
    do

      if is_reversed then
        target_index := target_index − step
      else
        target_index := target_index + step
```

```
      end
    ensure
      -- cursor_index_advanced: cursor_index_buggy = old cursor_index_buggy + 1
      cursor_index_advanced: (cursor_index_fixed = old cursor_index_fixed + 1)
    end

buggy_ID_279
      -- Move to next position.
    require -- from ITERATION_CURSOR
      valid_position: not (lower ≤ target_index and target_index ≤ upper) and lower ≤ upper
      -- not before
      target_index ≥ 0and target_index ≤ 10
      lower ≥ 0and lower ≤ 10
      upper ≥ 0and upper ≤ 10
      first_index ≥ 0and first_index ≤ 10
    do

      if is_reversed then
        target_index := target_index − step
      else
        target_index := target_index + step
      end
    ensure
      -- cursor_index_advanced: cursor_index_buggy = old cursor_index_buggy + 1
cursor_index_advanced:not (target_index ≠ first_index)implies ( ( cursor_index_fixed = old cursor_index_fixed + 1))
    end
```

281

```
INDEXABLE_ITERATION_CURSOR_BUG_1.buggy:::1
lower == upper
(is_reversed) or else (before) == true
(is_reversed) or else (not (before)) == true
(not (is_reversed)) or else (before) == true
(not (is_reversed)) or else (not (before)) == true
step == 1
lower > target_index
lower ≥ step
target_index != first_index
first_index ≥ step
```

### 3.7 linked_list_make_not_before_1.log

– fixing time: 3.13 minutes

```

buggy_ID_593
    do

      before := True
    ensure
      is_before: before
before_constraint:not (before)implies ( before implies (active =
     first_element))
    end

buggy_ID_598
require
not ((not (before)) or else (not (active = first_element)))
do

      before := True
    ensure
      is_before: before
      before_constraint: before implies (active = first_element)
    end

buggy_ID_599
    do

      before := True
    ensure
      is_before: before
before_constraint:not ((not (before)) or else (not (active = first_element)))
     implies ( before implies (active = first_element))
    end

buggy_ID_602
require
not ((not (before)) or else (not (before implies (active = first_element))))
do

      before := True
    ensure
      is_before: before
      before_constraint: before implies (active = first_element)
    end

buggy_ID_603
    do

      before := True
    ensure
      is_before: before
```

```
before_constraint:not ((not (before)) or else (not (before implies (active = first_element))))implies ( before implies (active = first_element))
    end

buggy_ID_629
    do

      before := True
    ensure
      is_before: before
before_constraint:not ((Current) ≠ Void)implies ( before implies (active = first_element))
    end

buggy_ID_630
require
not (active ≠ first_element)
do

      before := True
    ensure
      is_before: before
      before_constraint: before implies (active = first_element)
    end

buggy_ID_631
    do

      before := True
    ensure
      is_before: before
before_constraint:not (active ≠ first_element)implies ( before implies ( active = first_element))
    end
```

631

```

buggy_ID_32
    do

      if after then
after := not (after)
end
before := True
    ensure
      is_before: before
      not_both: not (after and before)
    end

buggy_ID_185
```

```
    do

      if (after) or else (after and before) then
after := not (after)
end
before := True
    ensure
      is_before: before
      not_both: not (after and before)
    end

buggy_ID_221
    do

      if (after) or else (not (not (after and before))) then
after := not (after)
end
before := True
    ensure
      is_before: before
      not_both: not (after and before)
    end

buggy_ID_253
    do

      before := True
    ensure
      is_before: before
not_both:not (before)implies ( not (after and before))
    end

buggy_ID_256
require
not (after)
do

      before := True
    ensure
      is_before: before
      not_both: not (after and before)
    end

buggy_ID_257
    do

      before := True
    ensure
      is_before: before
not_both:not (after)implies ( not (after and before))
```

```
    end

buggy_ID_260
require
not ((before) or else (after))
do

      before := True
    ensure
      is_before: before
      not_both: not (after and before)
    end

buggy_ID_261
    do

      before := True
    ensure
      is_before: before
not_both:not ((before) or else (after))implies ( not (after and before))
    end

buggy_ID_263
    do

      before := True
    ensure
      is_before: before
not_both:not ((before) or else (not (after)))implies ( not (after and before))
    end

buggy_ID_264
require
not ((not (before)) or else (after))
do

      before := True
    ensure
      is_before: before
      not_both: not (after and before)
    end

buggy_ID_265
    do

      before := True
    ensure
      is_before: before
not_both:not ((not (before)) or else (after))implies ( not (after and before))
    end
```

```
buggy_ID_268
require
not ((not (before)) or else (after and before))
do

      before := True
    ensure
      is_before: before
      not_both: not (after and before)
    end

buggy_ID_269
    do

      before := True
    ensure
      is_before: before
not_both:not ((not (before)) or else (after and before))implies ( not (after and
      before))
    end

buggy_ID_272
require
not ((not (before)) or else (not (not (after and before))))
do

      before := True
    ensure
      is_before: before
      not_both: not (after and before)
    end

buggy_ID_273
    do

      before := True
    ensure
      is_before: before
not_both:not ((not (before)) or else (not (not (after and before))))implies (
     not (after and before))
    end

buggy_ID_274
require
not ((after) or else (after and before))
do

      before := True
    ensure
```

```
      is_before: before
      not_both: not (after and before)
    end

buggy_ID_275
    do

      before := True
    ensure
      is_before: before
not_both:not ((after) or else (after and before))implies ( not (after and before))
    end

buggy_ID_277
    do

      before := True
    ensure
      is_before: before
not_both:not ((after) or else (not (after and before)))implies ( not (after and before))
    end

buggy_ID_278
require
not ((after) or else (not (not (after and before))))
do

      before := True
    ensure
      is_before: before
      not_both: not (after and before)
    end

buggy_ID_279
    do

      before := True
    ensure
      is_before: before
not_both:not ((after) or else (not (not (after and before))))implies ( not (after and before))
    end

buggy_ID_283
    do

      before := True
    ensure
```

```
        is_before: before
not_both:not ((Current) ≠ Void)implies ( not (after and before))
    end
```

| Class | Size | Routine | Violated assertion | Failure type | Fixing time | Proof time (s) | Candidate Fixes | Valid Fixes | Proper Fixes |
|---|---|---|---|---|---|---|---|---|---|
| ACCOUNT 7/3 | 97 | withdraw | balance_set | postcondition violation | – | 0.247 | – | 0 | – |
| | | | balance_non_negative | class invariant violation | 1.58 | 0.275 | 211 | 4 | 1 |
| | | deposit | balance_set | postcondition violation | 1.38 | 0.241 | 111 | 6 | 1 |
| | | transfer | amount $\leq$ 10 | precondition violation | – | 0.248 | – | 0 | – |
| | | | withdrawal_made | postcondition violation | 2.72 | 0.243 | 98 | 5 | 2 |
| | | | withdrawal_made | postcondition violation | – | 0.253 | – | 0 | – |
| | | | deposit_made | postcondition violation | – | 0.244 | – | 0 | – |
| CLOCK 8/3 | 131 | increase_hours | valid_hours | precondition violation | 2.72 | 0.253 | 356 | 5 | 1 |
| | | increase_minutes | hours_increased | postcondition violation | 2.43 | 0.251 | 301 | 11 | 1 |
| | | | hours_increased | postcondition violation | – | 0.263 | – | 0 | – |
| | | | minutes_increased | postcondition violation | – | 0.259 | – | 0 | – |
| | | increase_seconds | valid_seconds | precondition violation | 3.1 | 0.241 | 356 | 8 | 1 |
| | | | hours_increased | postcondition violation | – | 0.243 | – | 0 | – |
| | | | minutes_increased | postcondition violation | – | 0.248 | – | 0 | – |
| | | | valid_minutes | precondition violation | 3.18 | 0.245 | 356 | 8 | 1 |
| HEATER 4/4 | 73 | turn_on_off | heater_remains_off | postcondition violation | 5.28 | 0.615 | 437 | 24 | 0 |
| | | | heater_remains_on | postcondition violation | 4.05 | 0.642 | 437 | 25 | 0 |
| 0 | | | heater_remains_on | postcondition violation | 4.17 | 0.250 | 425 | 19 | 0 |
| | | | heater_remains_off | postcondition violation | 4.65 | 0.246 | 429 | 17 | 0 |
| LAMP 4/3 | 71 | turn_on_off | turn_off | postcondition violation | – | 0.250 | – | 0 | – |
| | | | turn_off | postcondition violation | 6.83 | 0.278 | 534 | 10 | 0 |
| | | adjust_light | from_high_to_low | postcondition violation | 3.3 | 0.265 | 422 | 4 | 0 |
| | | | from_medium_to_high | postcondition violation | 3.62 | 0.270 | 406 | 4 | 0 |
| ARITHMETIC 3/2 | 176 | multiply_recursive | result_correct | postcondition violation | 1.1 | 0.301 | 22 | 6 | 3 |
| | | | is_maximum | postcondition violation | – | 0.321 | – | 0 | – |
| | | | add_recursive | postcondition violation | 1.02 | 0.307 | 30 | 9 | 0 |
| BINARY_SEARCH 6/0 | 50 | binary_search | present | postcondition violation | – | 0.327 | – | 0 | – |
| | | | not_in_lower_part | invariant not maintained | – | 0.39 | – | 0 | – |
| | | | not_in_lower_part | invariant not maintained | – | 0.395 | – | 0 | – |
| | | | – | variant not decreased | – | 0.325 | – | 0 | – |
| | | | not_in_upper_part | invariant not maintained | – | 0.288 | – | 0 | – |
| | | | present | postcondition violation | – | 0.550 | – | 0 | – |
| | | | not_in_lower_part | invariant violated on entry | – | 0.356 | – | 0 | – |
| MAX_IN_ARRAY 6/0 | 33 | max_in_array | max_so_far | invariant not maintained | – | 0.288 | – | 0 | – |
| | | | is_maximum | postcondition violation | – | 0.284 | – | 0 | – |
| | | | result_in_array | postcondition violation | – | 0.278 | – | 0 | – |
| | | | is_maximum | postcondition violation | – | 0.286 | – | 0 | – |
| | | | max_so_far | invariant violated on entry | – | 0.290 | – | 0 | – |
| | | | i_in_bounds | invariant violated on entry | – | 0.282 | – | 0 | – |
| SQUARE_ROOT 4/3 | 58 | square_root | – | variant not decreased | 1.58 | 0.258 | 9 | 1 | 0 |
| | | | valid_result | postcondition violation | | 0.257 | – | 0 | – |
| | | | valid_result | invariant not maintained | 1.68 | 0.308 | 9 | 1 | 0 |
| | | | result_so_far | invariant not maintained | 1.43 | 0.400 | 9 | 1 | 0 |
| Total | 556 | 14 | 49 | | 44069 | 15.968 | 13.985 | 37/12 | |

| Class | Size | Routine | Violated assertion | Failure type | Fixing time | Proof time (s) | Candidate Fixes | Valid Fixes | Proper Fixes |
|---|---|---|---|---|---|---|---|---|---|
| V_ARRAY 1/1 | 1756 | index_set | same_bounds | postcondition violation | 2.42 | 0.253 | 267 | 6 | 4 |
| V_ARRAYED_LIST 1/1 | 1090 | merge_right | new_count | postcondition violation | 9.38 | 0.346 | 121 | 9 | 0 |
| V_INDEXABLE_ITERATION_CURSOR 1/1 | 1175 | forth | index_index_adjusted | postcondition violation | 2.43 | 0.255 | 281 | 7 | 0 |
| V_LINKED_LIST 2/2 | 2145 | make | before_constraint | class invariant violation | 3.13 | 0.246 | 631 | 8 | 0 |
| | | | not_both | class invariant violation | 1.53 | 0.310 | 283 | 21 | 2 |
| Total | 556 | 14 | 49 | | 44069 | 15.968 | 13.985 | 37/12 | |