Evolving Verified Cloud Authorization

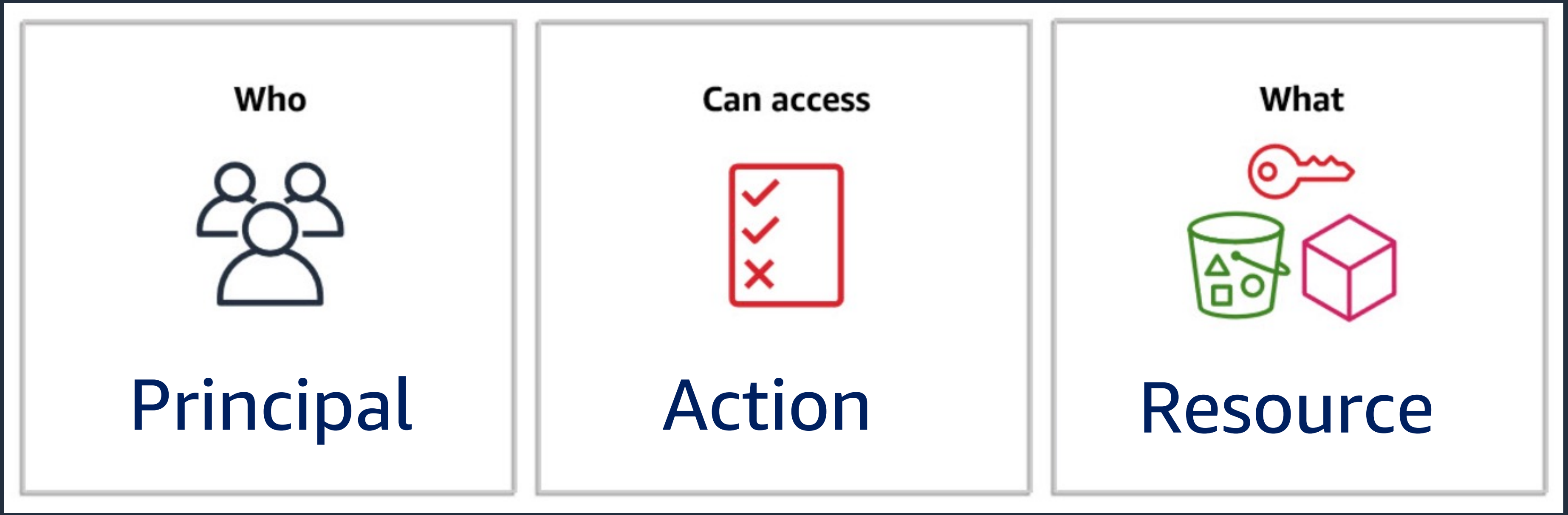
Sean McLaughlin

AWS Identity
seanmcl@amazon.com

© 2022, Amazon Web Services, Inc. or its affiliates.

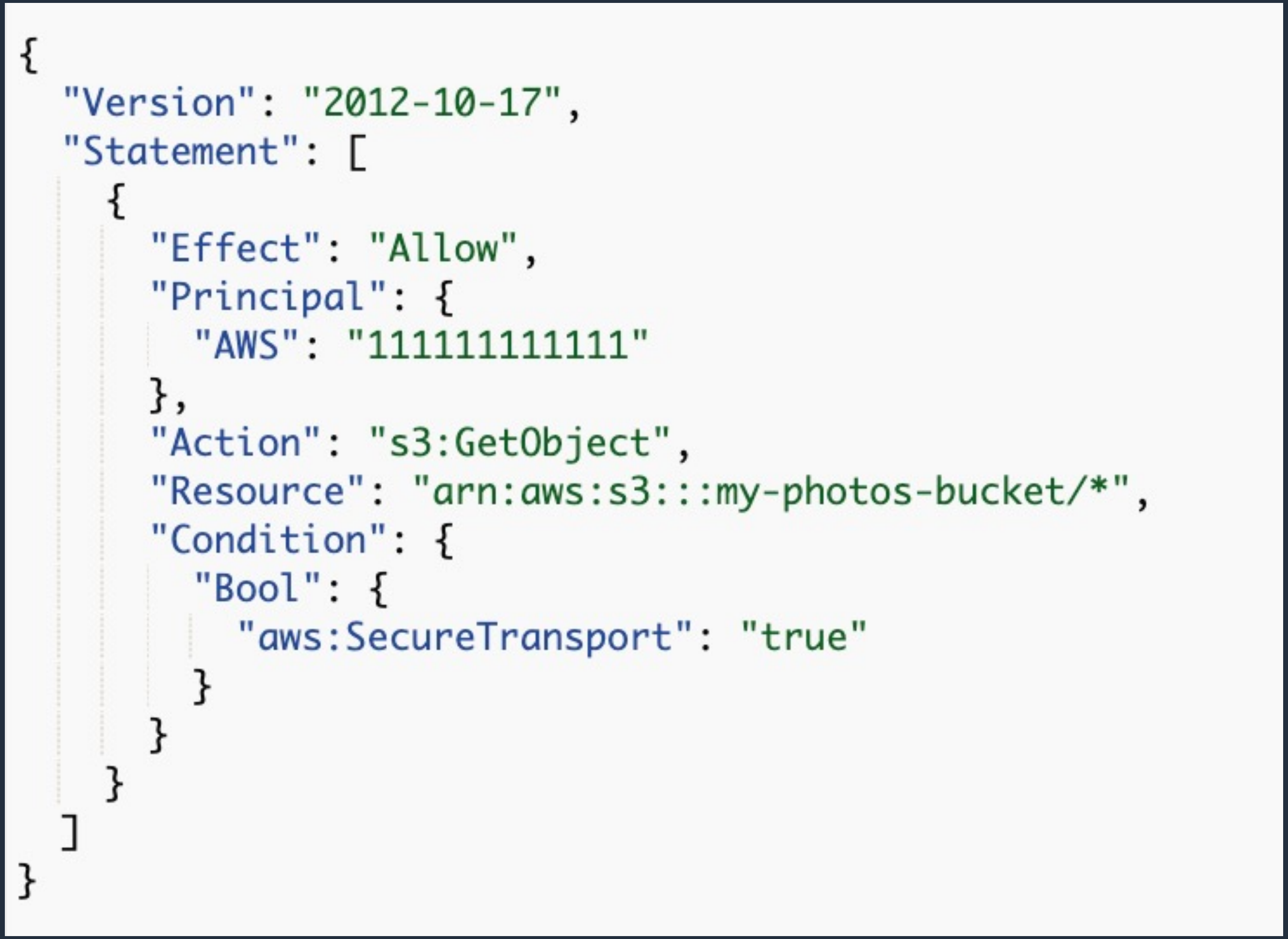
Formal verification enables faster evolution of critical systems

© 2022, Amazon Web Services, Inc. or its affiliates.



© 2022, Amazon Web Services, Inc. or its affiliates.

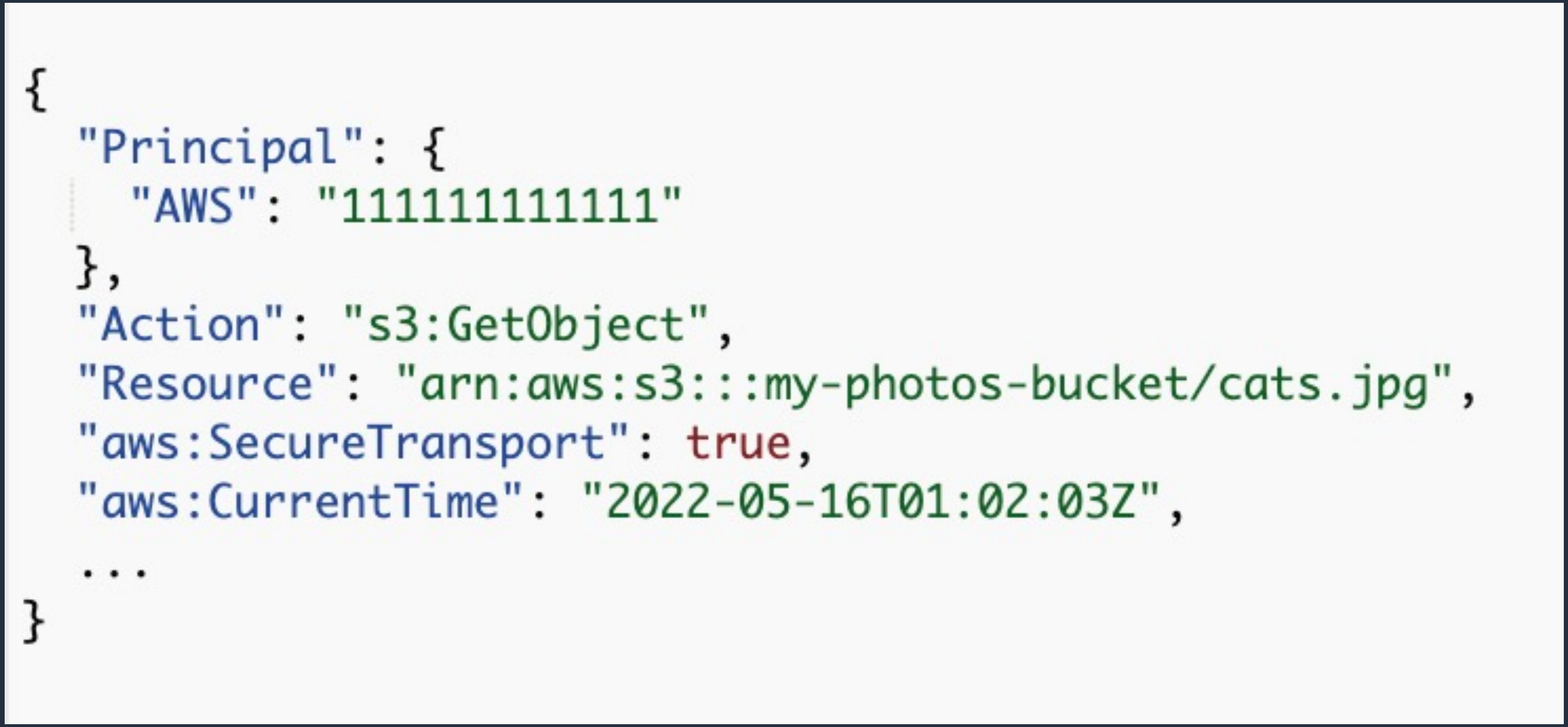
Policies



```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111111111111"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::my-photos-bucket/*",
      "Condition": {
        "Bool": {
          "aws:SecureTransport": "true"
        }
      }
    }
  ]
}
```

© 2022, Amazon Web Services, Inc. or its affiliates.

Requests

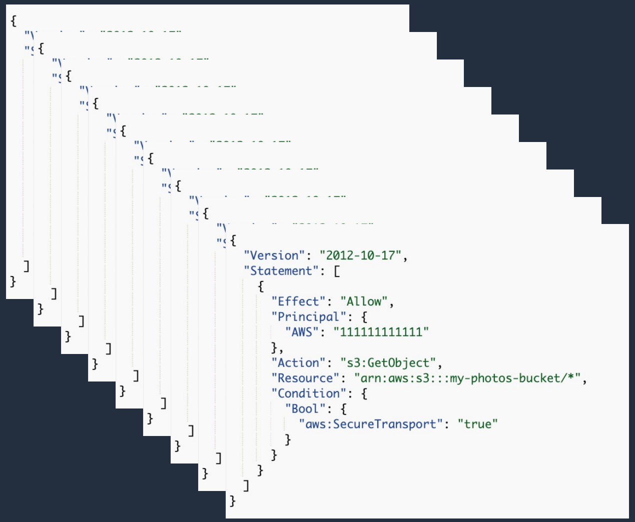


```
{
  "Principal": {
    "AWS": "111111111111"
  },
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:::my-photos-bucket/cats.jpg",
  "aws:SecureTransport": true,
  "aws:CurrentTime": "2022-05-16T01:02:03Z",
  ...
}
```

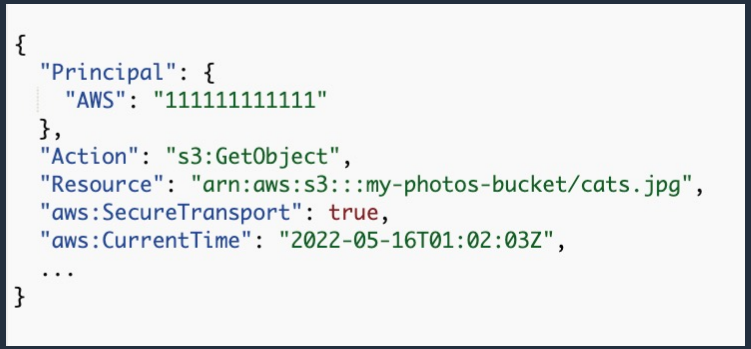
© 2022, Amazon Web Services, Inc. or its affiliates.

The authorization problem

Policies + Request = Allowed



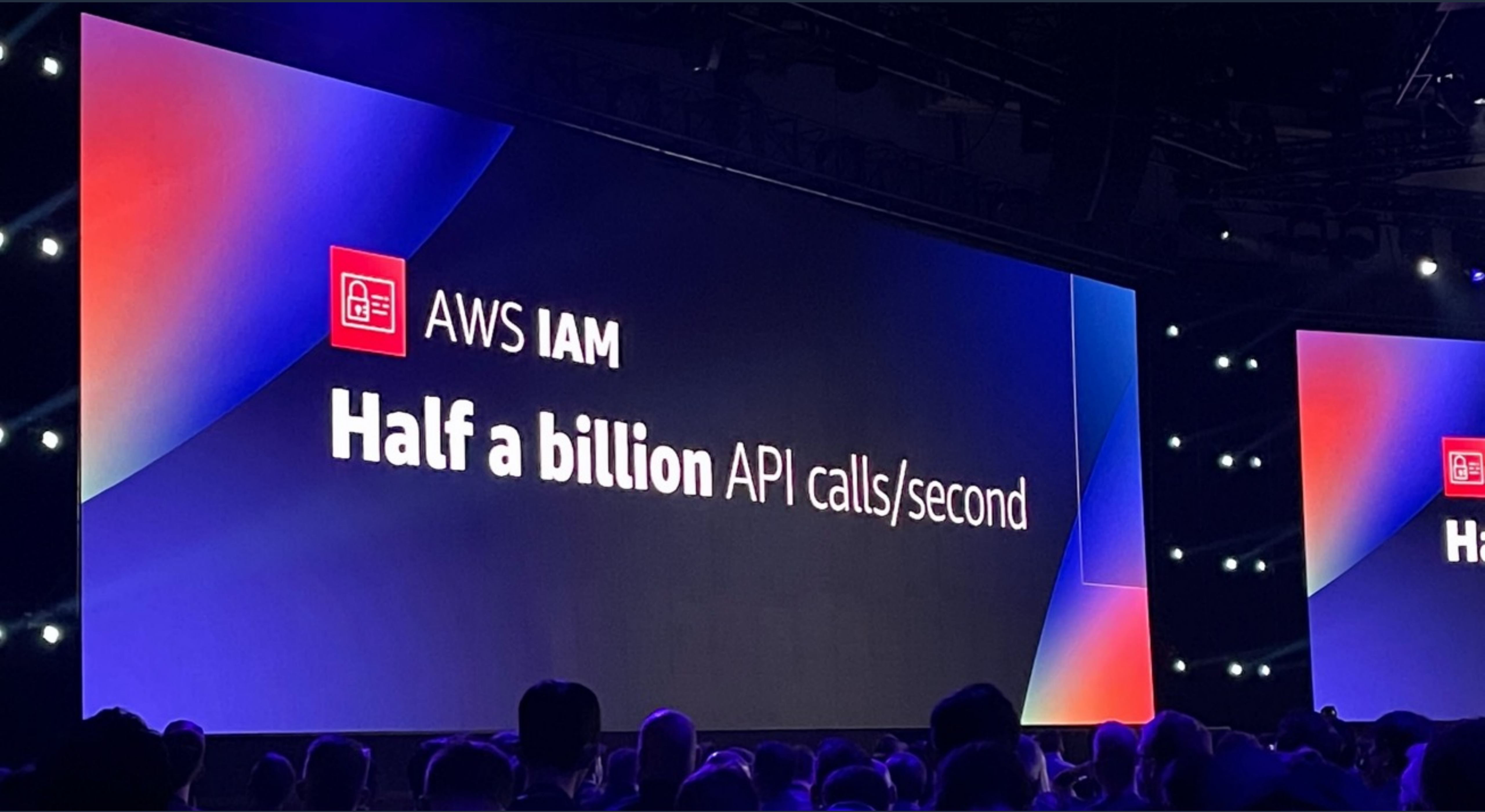
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111111111111"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::my-photos-bucket/*",
      "Condition": {
        "Bool": {
          "aws:SecureTransport": "true"
        }
      }
    }
  ]
}
```



```
{
  "Principal": {
    "AWS": "111111111111"
  },
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:::my-photos-bucket/cats.jpg",
  "aws:SecureTransport": true,
  "aws:CurrentTime": "2022-05-16T01:02:03Z",
  ...
}
```

Yes/No

© 2022, Amazon Web Services, Inc. or its affiliates.



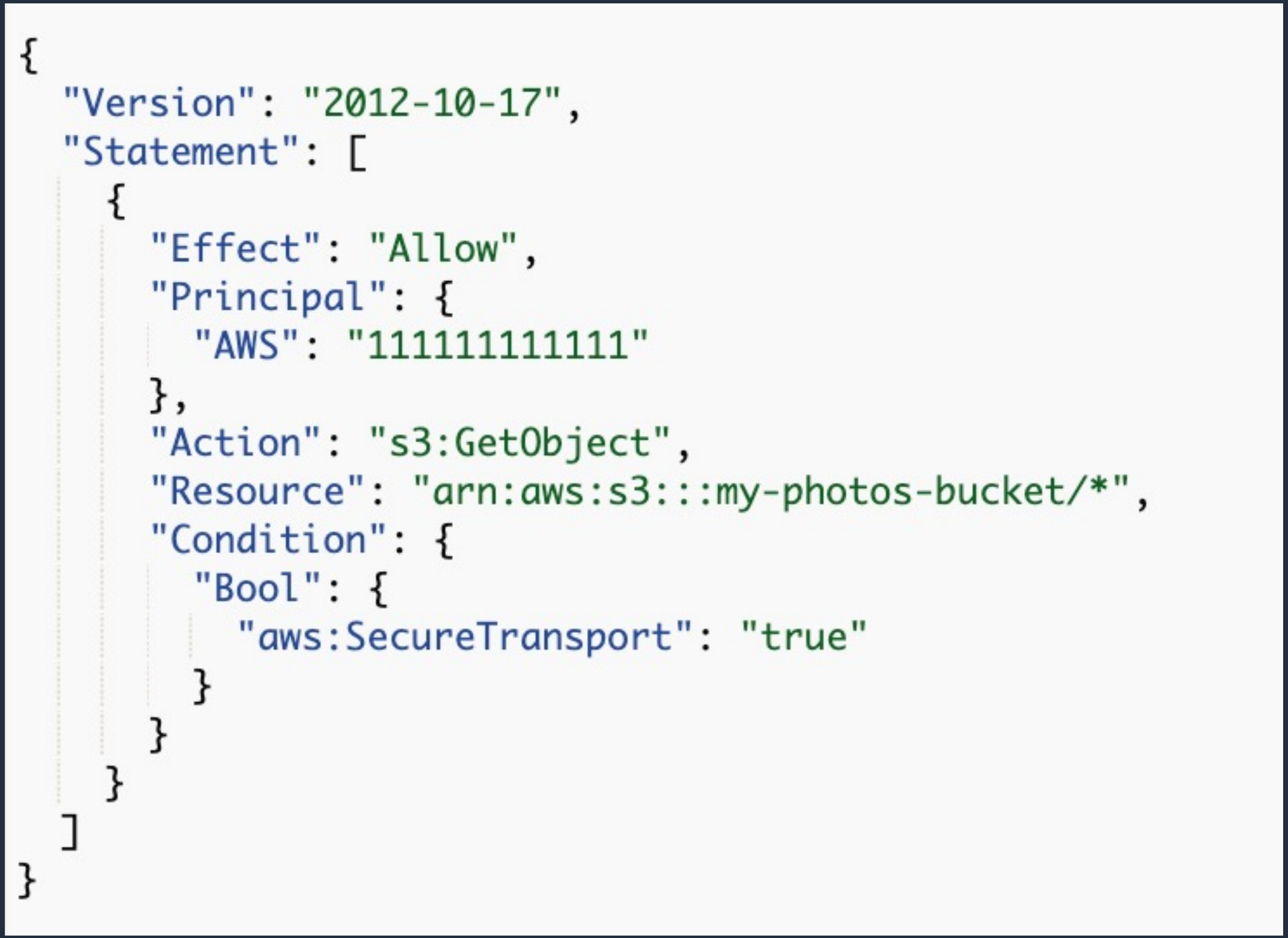
AWS **IAM**

Half a billion API calls/second

Making changes

© 2022, Amazon Web Services, Inc. or its affiliates.

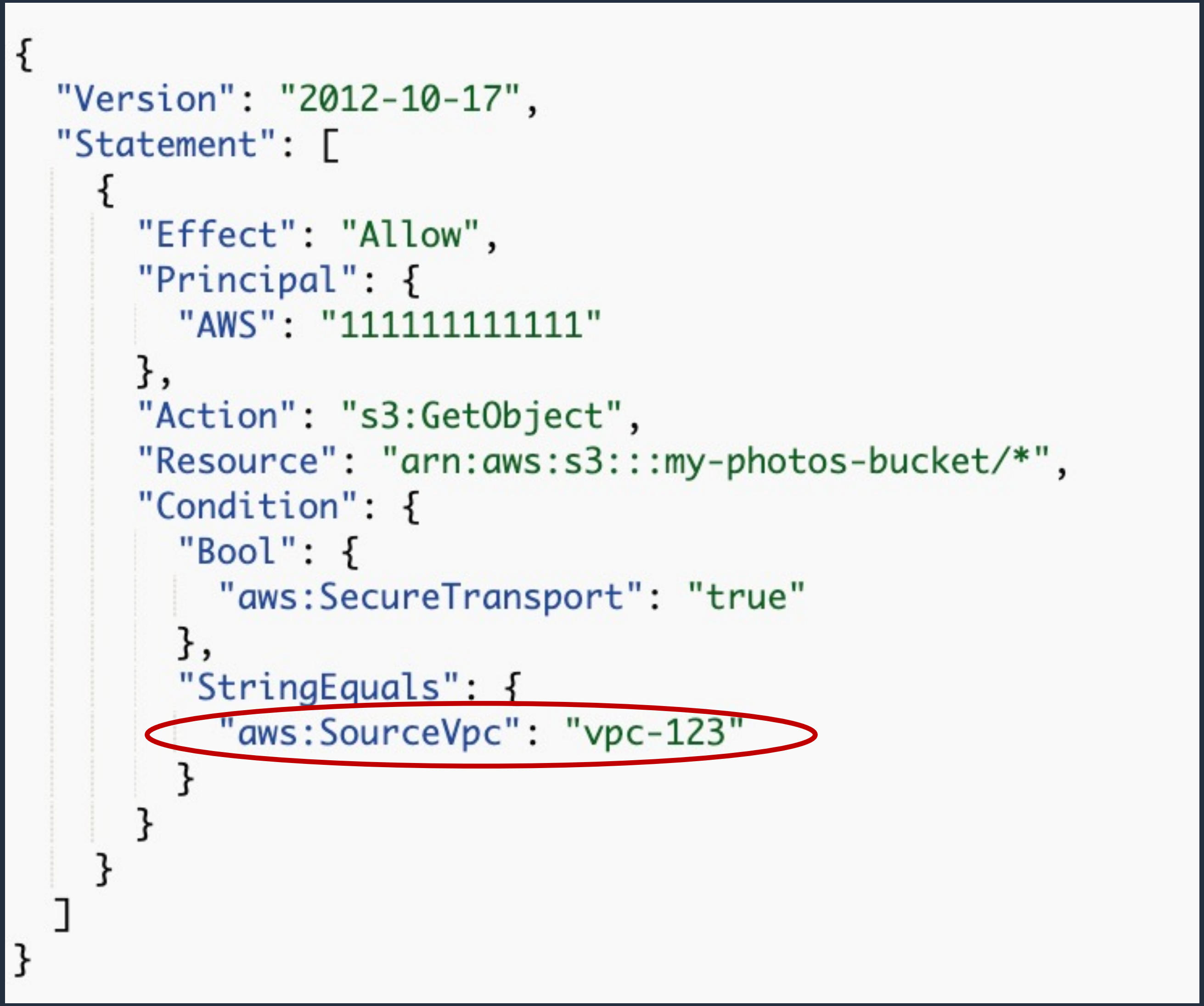
Making changes



```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111111111111"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::my-photos-bucket/*",
      "Condition": {
        "Bool": {
          "aws:SecureTransport": "true"
        }
      }
    }
  ]
}
```

© 2022, Amazon Web Services, Inc. or its affiliates.

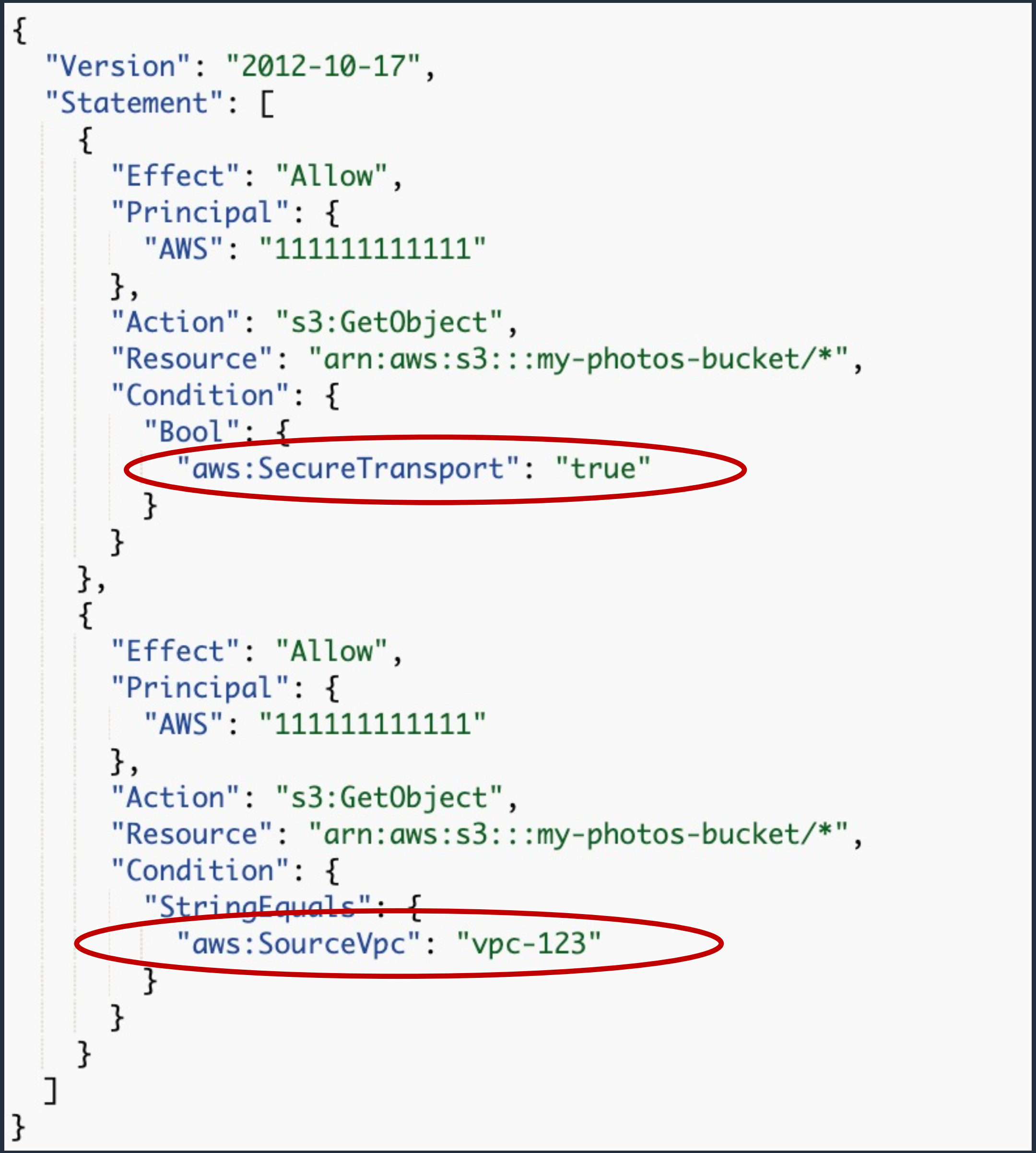
Making changes



```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111111111111"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::my-photos-bucket/*",
      "Condition": {
        "Bool": {
          "aws:SecureTransport": "true"
        },
        "StringEquals": {
          "aws:SourceVpc": "vpc-123"
        }
      }
    }
  ]
}
```

© 2022, Amazon Web Services, Inc. or its affiliates.

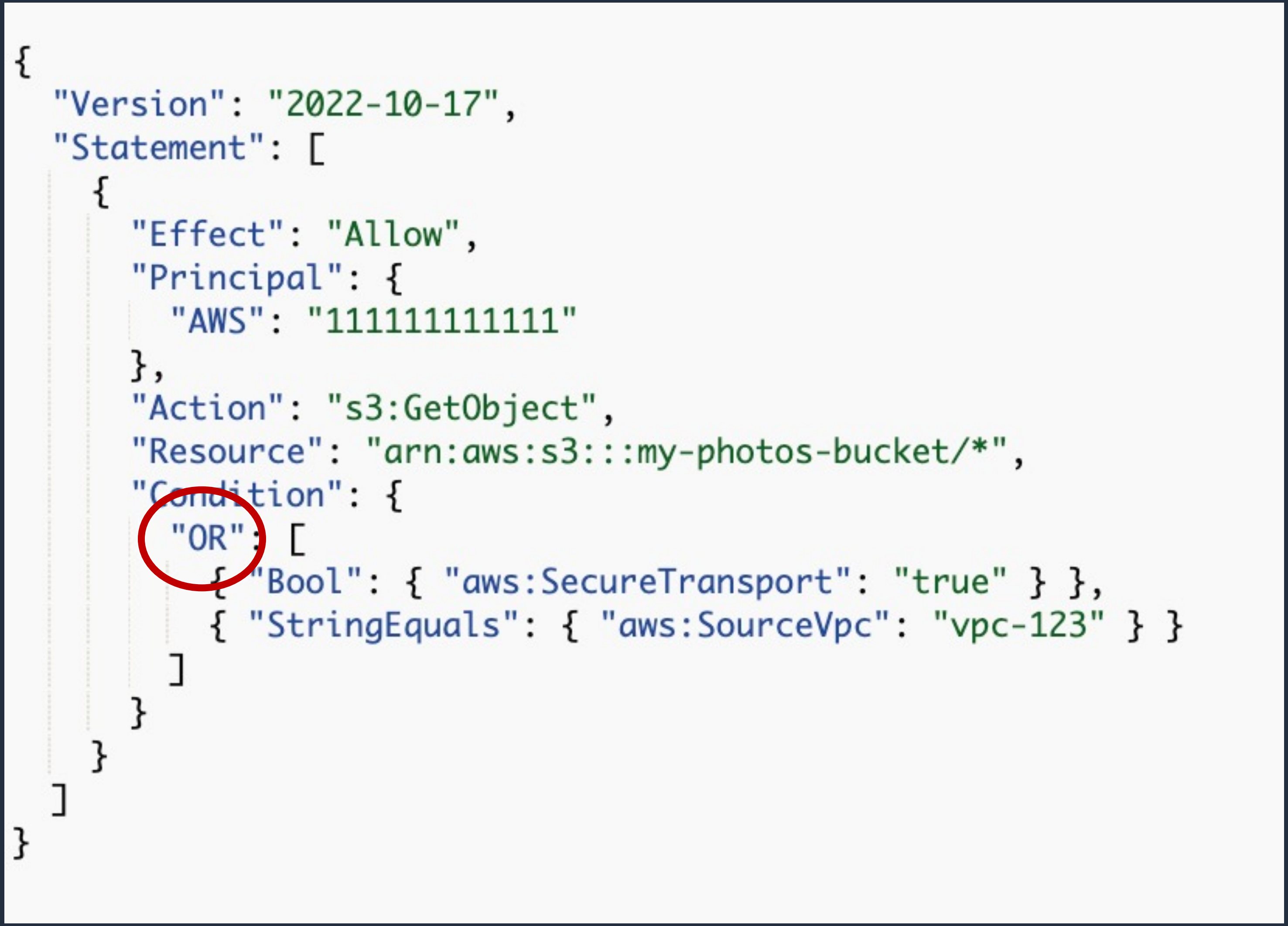
Making changes



```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111111111111"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::my-photos-bucket/*",
      "Condition": {
        "Bool": {
          "aws:SecureTransport": "true"
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111111111111"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::my-photos-bucket/*",
      "Condition": {
        "StringEquals": {
          "aws:SourceVpc": "vpc-123"
        }
      }
    }
  ]
}
```

© 2022, Amazon Web Services, Inc. or its affiliates.

Making changes



```
{
  "Version": "2022-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111111111111"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::my-photos-bucket/*",
      "Condition": {
        "OR": [
          { "Bool": { "aws:SecureTransport": "true" } },
          { "StringEquals": { "aws:SourceVpc": "vpc-123" } }
        ]
      }
    }
  ]
}
```

© 2022, Amazon Web Services, Inc. or its affiliates.

Adding “OR”

Pros

- More direct, succinct policies

Cons

- Availability risk
- Security risk

© 2022, Amazon Web Services, Inc. or its affiliates.

Testing

What we can do

- Fuzzing
- Historical data
- Side-by-side execution

We can’t test exhaustively

- Language is complex (111 pages of guide)
- Single request can have dozens of policies and dozens of variables

© 2022, Amazon Web Services, Inc. or its affiliates.

Formal verification enables faster evolution of critical systems

© 2022, Amazon Web Services, Inc. or its affiliates.

Formal Verification Method

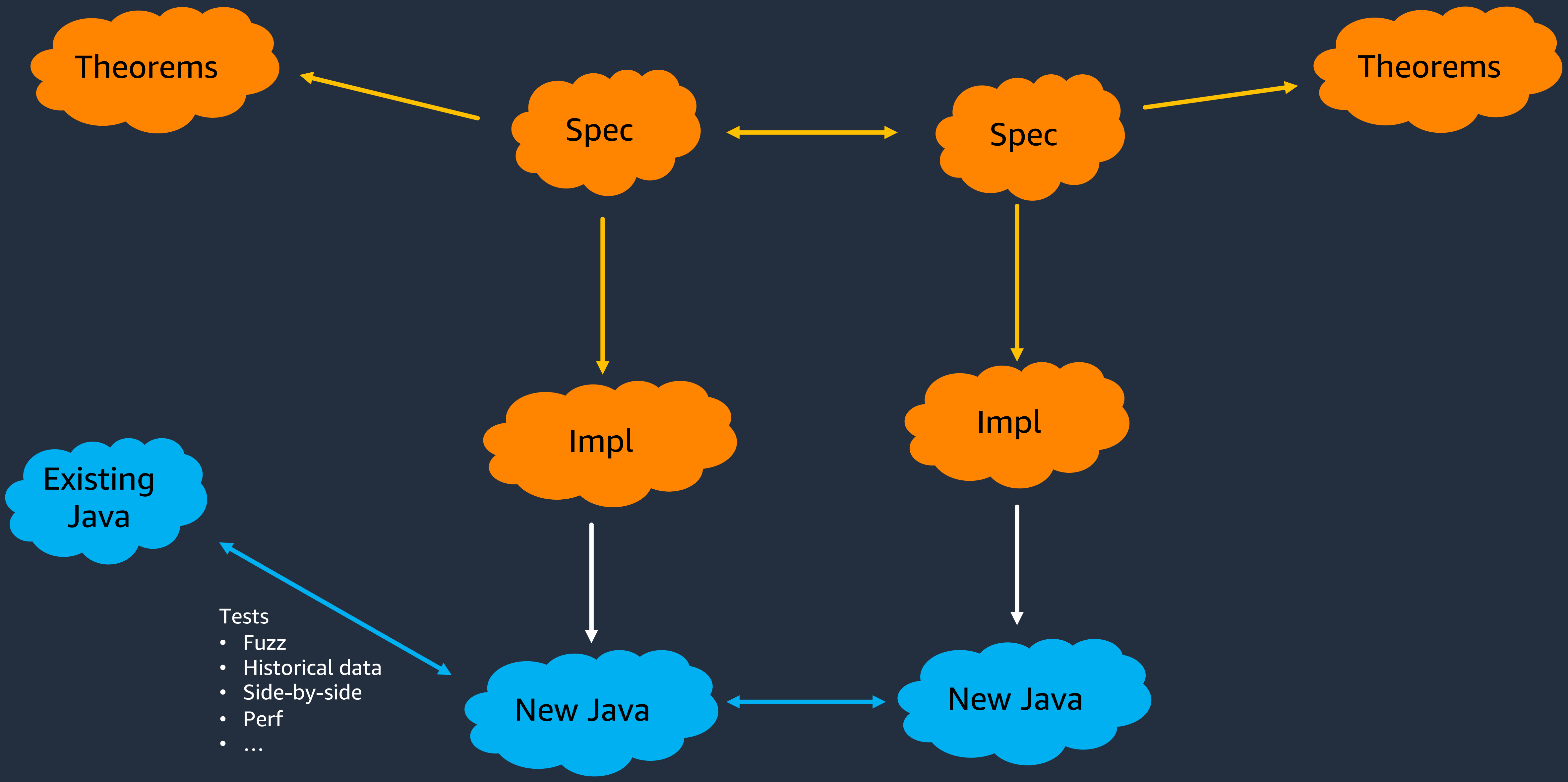
© 2022, Amazon Web Services, Inc. or its affiliates.

Verified Implementation

Idiomatic Compiler

Proof Evolution

© 2022, Amazon Web Services, Inc. or its affiliates.



Tests
- Fuzz
- Historical data
- Side-by-side
- Perf
- …

© 2022, Amazon Web Services, Inc. or its affiliates.

Running Example

```
StringEqualsIgnoreDashes(“ABC”, “ABC”)

StringEqualsIgnoreDashes(“A-B-C”, “ABC”)

StringEqualsIgnoreDashes(“ABC”, “A-B-C”)

StringEqualsIgnoreDashes(“-----”, “”)
```

© 2022, Amazon Web Services, Inc. or its affiliates.

Verified Implementation

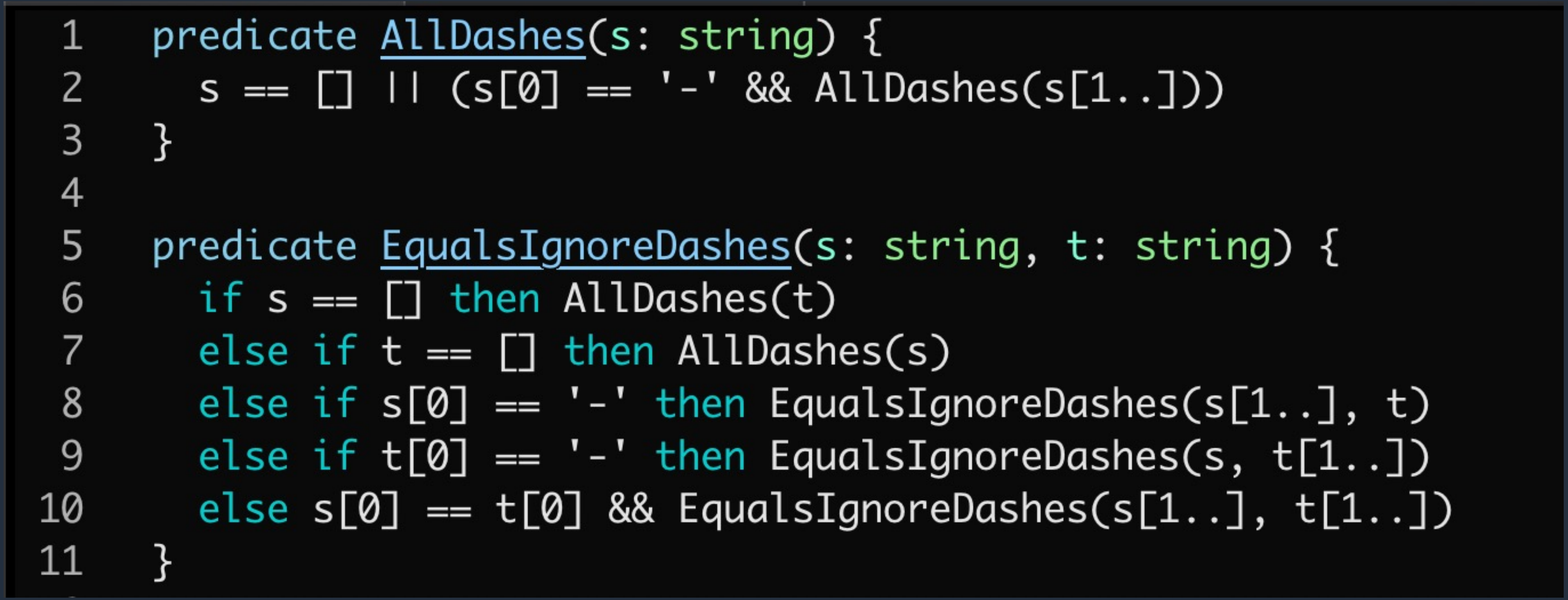
1. Specification
2. Theorems
3. Implementation

Idiomatic Compiler

Proof Evolution

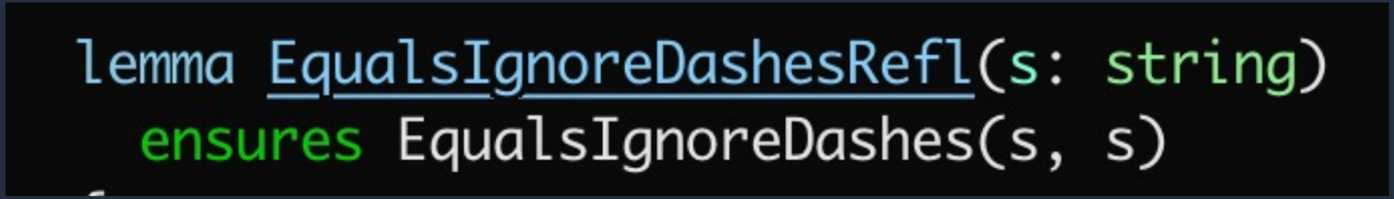
© 2022, Amazon Web Services, Inc. or its affiliates.

1. Specification

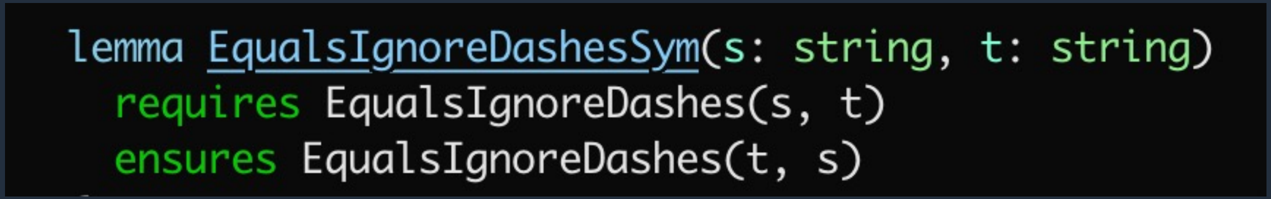

```
predicate AllDashes(s: string) {
  s == [] || (s[0] == '-' && AllDashes(s[1..]))
}

predicate EqualsIgnoreDashes(s: string, t: string) {
  if s == [] then AllDashes(t)
  else if t == [] then AllDashes(s)
  else if s[0] == '-' then EqualsIgnoreDashes(s[1..], t)
  else if t[0] == '-' then EqualsIgnoreDashes(s, t[1..])
  else s[0] == t[0] && EqualsIgnoreDashes(s[1..], t[1..])
}
```

© 2022, Amazon Web Services, Inc. or its affiliates.

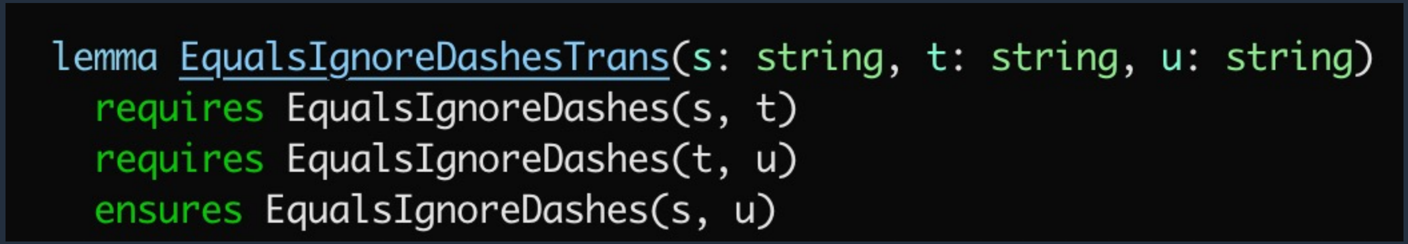
2. Theorems



```
lemma EqualsIgnoreDashesRefl(s: string)
  ensures EqualsIgnoreDashes(s, s)
```



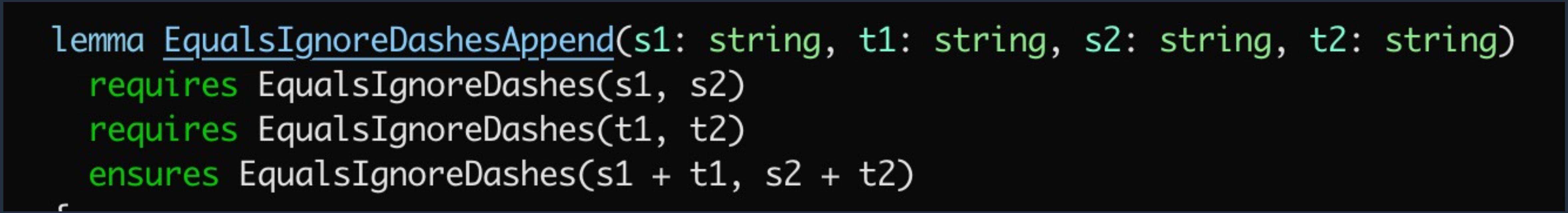
```
lemma EqualsIgnoreDashesSym(s: string, t: string)
  requires EqualsIgnoreDashes(s, t)
  ensures EqualsIgnoreDashes(t, s)
```



```
lemma EqualsIgnoreDashesTrans(s: string, t: string, u: string)
  requires EqualsIgnoreDashes(s, t)
  requires EqualsIgnoreDashes(t, u)
  ensures EqualsIgnoreDashes(s, u)
```

© 2022, Amazon Web Services, Inc. or its affiliates.

2. Theorems



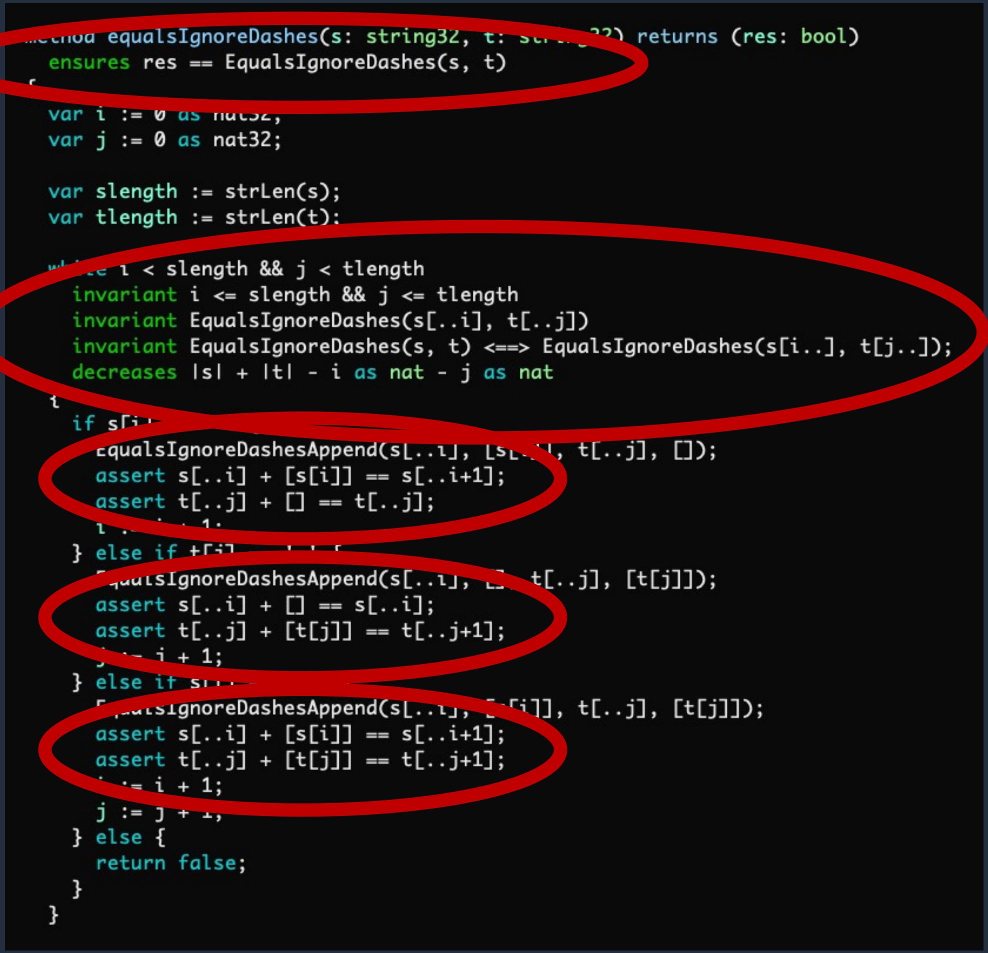
```
lemma EqualsIgnoreDashesAppend(s1: string, t1: string, s2: string, t2: string)
  requires EqualsIgnoreDashes(s1, s2)
  requires EqualsIgnoreDashes(t1, t2)
  ensures EqualsIgnoreDashes(s1 + t1, s2 + t2)
{
```

© 2022, Amazon Web Services, Inc. or its affiliates.

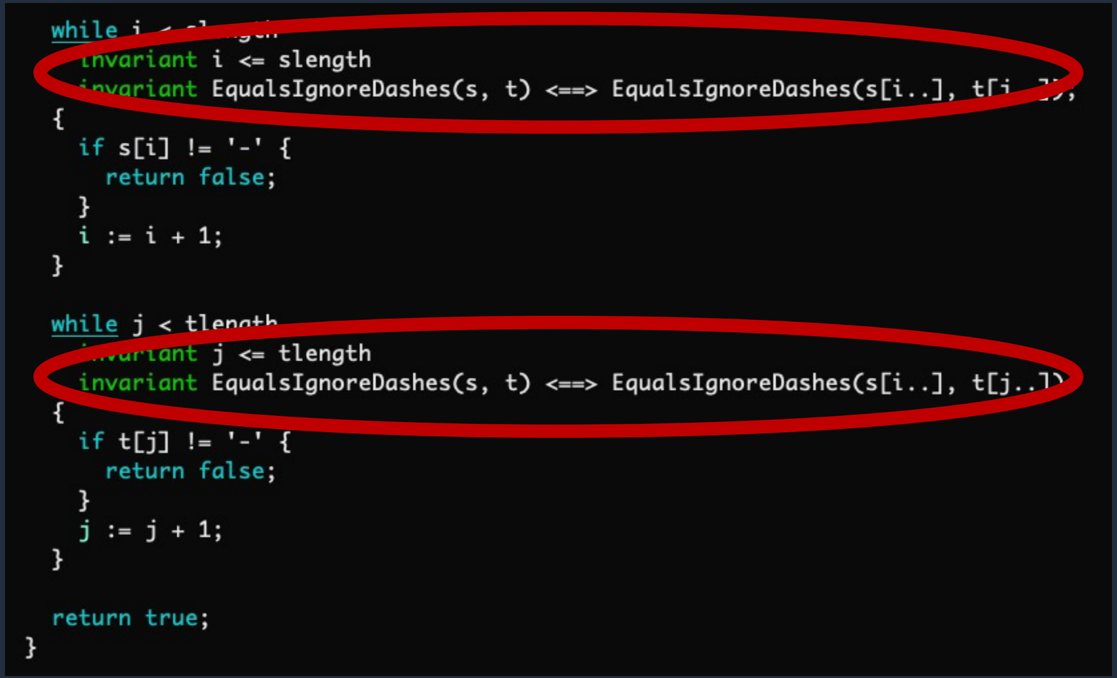
2. Theorems

```
lemma EqualsIgnoreDashesAppend(s1: string, t1: string, s2: string, t2: string)
  requires EqualsIgnoreDashes(s1, s2)
  requires EqualsIgnoreDashes(t1, t2)
  ensures EqualsIgnoreDashes(s1 + t1, s2 + t2)
{
  if s1 == [] {
    assert s1 + t1 == t1;
    assert AllDashes(s2);
    if s2 == [] {
      assert s2 + t2 == t2;
    } else {
      EqualsIgnoreDashesAppend(s1, t1, s2[1..], t2);
      EqualsIgnoreDashesRightAdd(s1 + t1, s2[1..] + t2);
      assert s2 + t2 == ['-'] + (s2[1..] + t2);
    }
  } else {
    if s1[0] == '-' {
      assert EqualsIgnoreDashes(s1[1..] + t1, s2 + t2);
      assert s1 + t1 == ['-'] + (s1[1..] + t1);
    } else {
      if s2 == [] {
      } else {
        if s2[0] == '-' {
          assert s2 + t2 == ['-'] + (s2[1..] + t2);
        } else {
          assert s1 + t1 == [s1[0]] + (s1[1..] + t1);
          assert s2 + t2 == [s2[0]] + (s2[1..] + t2);
        }
      }
    }
  }
}
```

3. Implementation



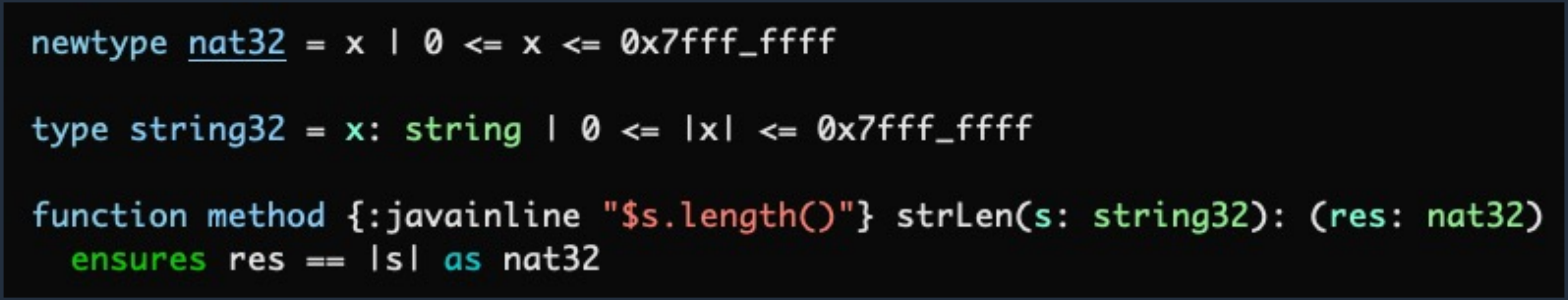
```
method equalsIgnoreDashes(s: string32, t: string32) returns (res: bool)
  ensures res == EqualsIgnoreDashes(s, t)
{
  var i := 0 as nat32;
  var j := 0 as nat32;

  var slength := strLen(s);
  var tlength := strLen(t);

  while i < slength && j < tlength
    invariant i <= slength && j <= tlength
    invariant EqualsIgnoreDashes(s[..i], t[..j])
    invariant EqualsIgnoreDashes(s, t) <==> EqualsIgnoreDashes(s[i..], t[j..]);
    decreases |s| + |t| - i as nat - j as nat
  {
    if s[i [illegible]
      EqualsIgnoreDashesAppend(s[..i], [s[[illegible], t[..j], []);
      assert s[..i] + [s[i]] == s[..i+1];
      assert t[..j] + [] == t[..j];
      i := i + 1;
    } else if t[j] [illegible] {
      EqualsIgnoreDashesAppend(s[..i], [], t[..j], [t[j]]);
      assert s[..i] + [] == s[..i];
      assert t[..j] + [t[j]] == t[..j+1];
      j := j + 1;
    } else if s[i [illegible]
      EqualsIgnoreDashesAppend(s[..i], [s[i]], t[..j], [t[j]]);
      assert s[..i] + [s[i]] == s[..i+1];
      assert t[..j] + [t[j]] == t[..j+1];
      i := i + 1;
      j := j + 1;
    } else {
      return false;
    }
  }
```



```
  while i < slength
    invariant i <= slength
    invariant EqualsIgnoreDashes(s, t) <==> EqualsIgnoreDashes(s[i..], t[j..]);
  {
    if s[i] != '-' {
      return false;
    }
    i := i + 1;
  }

  while j < tlength
    invariant j <= tlength
    invariant EqualsIgnoreDashes(s, t) <==> EqualsIgnoreDashes(s[i..], t[j..]);
  {
    if t[j] != '-' {
      return false;
    }
    j := j + 1;
  }

  return true;
}
```

© 2022, Amazon Web Services, Inc. or its affiliates.

3. Implementation



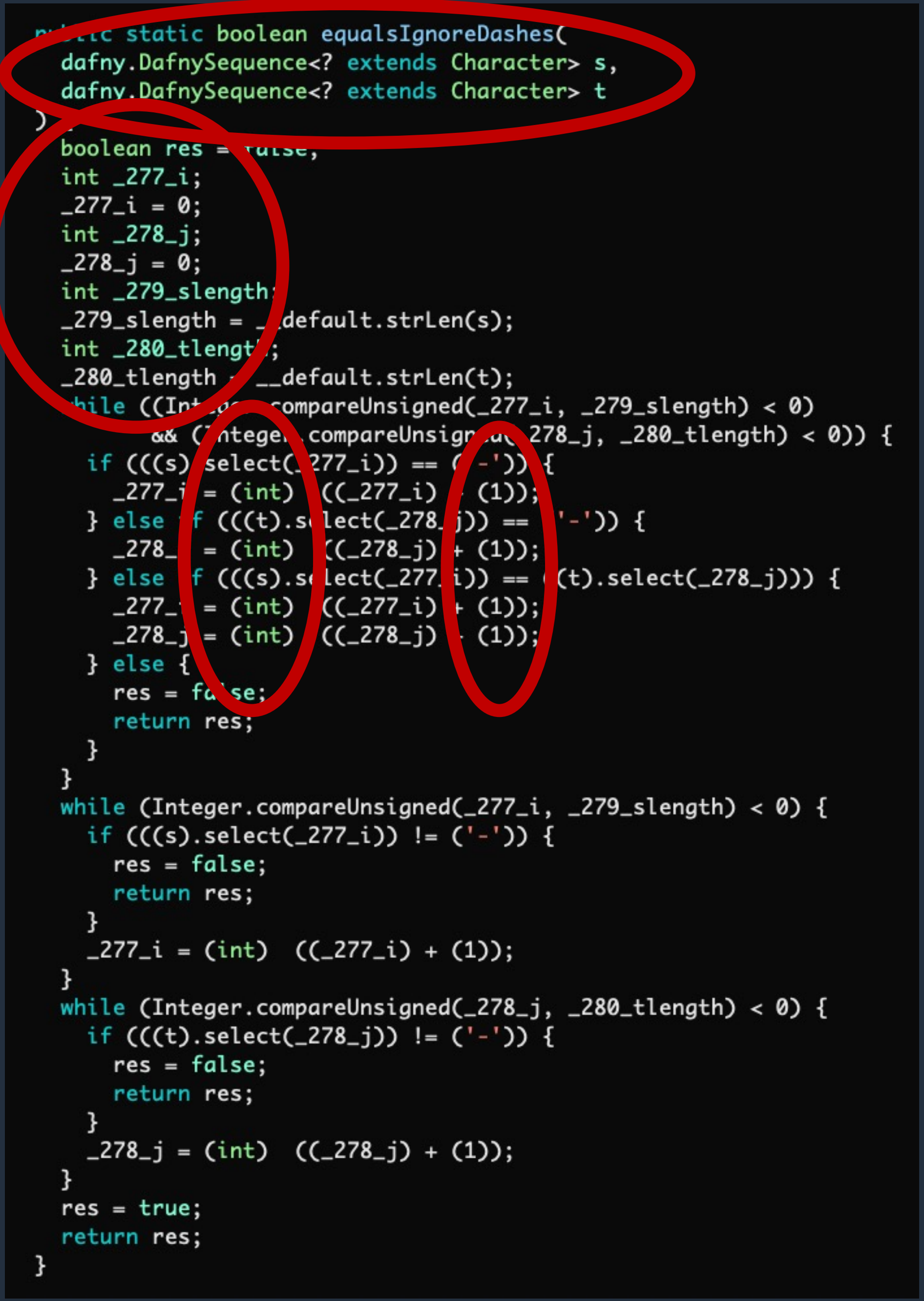
```
newtype nat32 = x | 0 <= x <= 0x7fff_ffff

type string32 = x: string | 0 <= |x| <= 0x7fff_ffff

function method {:javainline "$s.length()"} strLen(s: string32): (res: nat32)
  ensures res == |s| as nat32
```

Verified Implementation

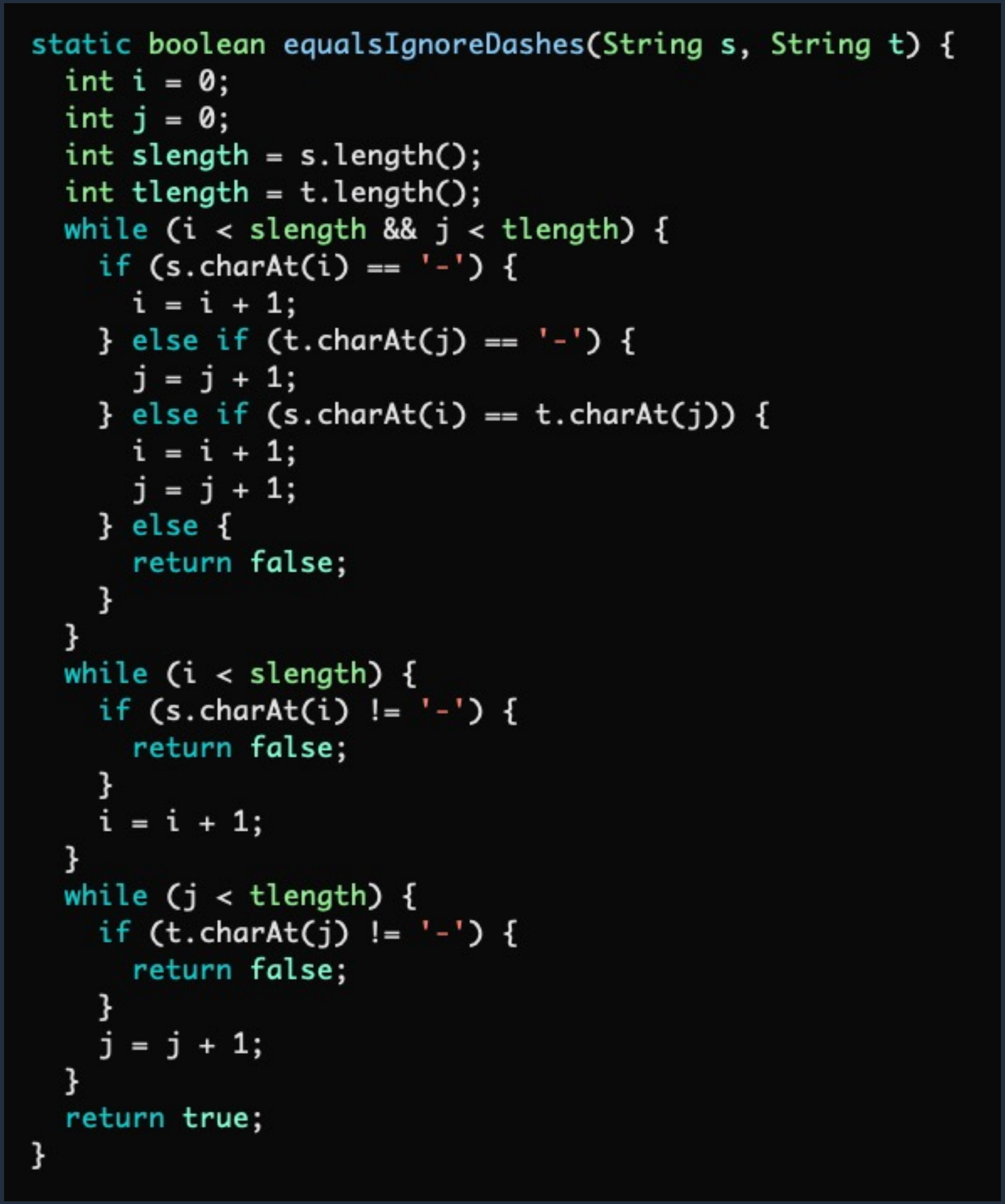
Idiomatic Compiler

Proof Evolution

© 2022, Amazon Web Services, Inc. or its affiliates.



```
public static boolean equalsIgnoreDashes(
  dafny.DafnySequence<? extends Character> s,
  dafny.DafnySequence<? extends Character> t
) {
  boolean res = false;
  int _277_i;
  _277_i = 0;
  int _278_j;
  _278_j = 0;
  int _279_slength;
  _279_slength = __default.strLen(s);
  int _280_tlength;
  _280_tlength = __default.strLen(t);
  while ((Integer.compareUnsigned(_277_i, _279_slength) < 0)
         && (Integer.compareUnsigned(_278_j, _280_tlength) < 0)) {
    if (((s).select(_277_i)) == ('-')) {
      _277_i = (int) ((_277_i) + (1));
    } else if (((t).select(_278_j)) == ('-')) {
      _278_j = (int) ((_278_j) + (1));
    } else if (((s).select(_277_i)) == ((t).select(_278_j))) {
      _277_i = (int) ((_277_i) + (1));
      _278_j = (int) ((_278_j) + (1));
    } else {
      res = false;
      return res;
    }
  }
  while (Integer.compareUnsigned(_277_i, _279_slength) < 0) {
    if (((s).select(_277_i)) != ('-')) {
      res = false;
      return res;
    }
    _277_i = (int)  ((_277_i) + (1));
  }
  while (Integer.compareUnsigned(_278_j, _280_tlength) < 0) {
    if (((t).select(_278_j)) != ('-')) {
      res = false;
      return res;
    }
    _278_j = (int)  ((_278_j) + (1));
  }
  res = true;
  return res;
}
```



```java
static boolean equalsIgnoreDashes(String s, String t) {
  int i = 0;
  int j = 0;
  int slength = s.length();
  int tlength = t.length();
  while (i < slength && j < tlength) {
    if (s.charAt(i) == '-') {
      i = i + 1;
    } else if (t.charAt(j) == '-') {
      j = j + 1;
    } else if (s.charAt(i) == t.charAt(j)) {
      i = i + 1;
      j = j + 1;
    } else {
      return false;
    }
  }
  while (i < slength) {
    if (s.charAt(i) != '-') {
      return false;
    }
    i = i + 1;
  }
  while (j < tlength) {
    if (t.charAt(j) != '-') {
      return false;
    }
    j = j + 1;
  }
  return true;
}
```

Pros

- Lowers the risk of experimentation
- Generated code is reviewed as usual
- Can be directly changed during operational event

Cons

- Language restricted (Dafny-Lite)
- Exceptions are hard

© 2022, Amazon Web Services, Inc. or its affiliates.

Other languages?

- Rust
  - Exception headaches go away entirely
  - Modeling lifetimes in Dafny is challenging

© 2022, Amazon Web Services, Inc. or its affiliates.

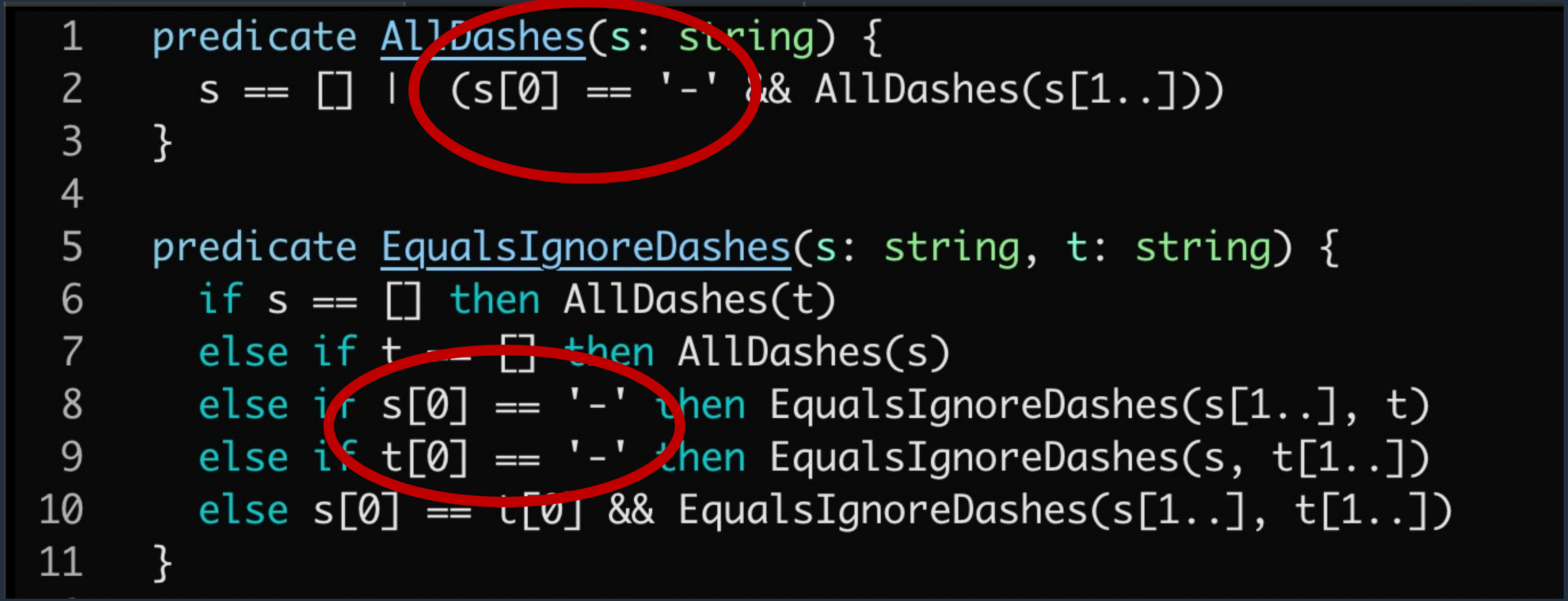
Verified Implementation

Idiomatic Compiler

Proof Evolution

© 2022, Amazon Web Services, Inc. or its affiliates.

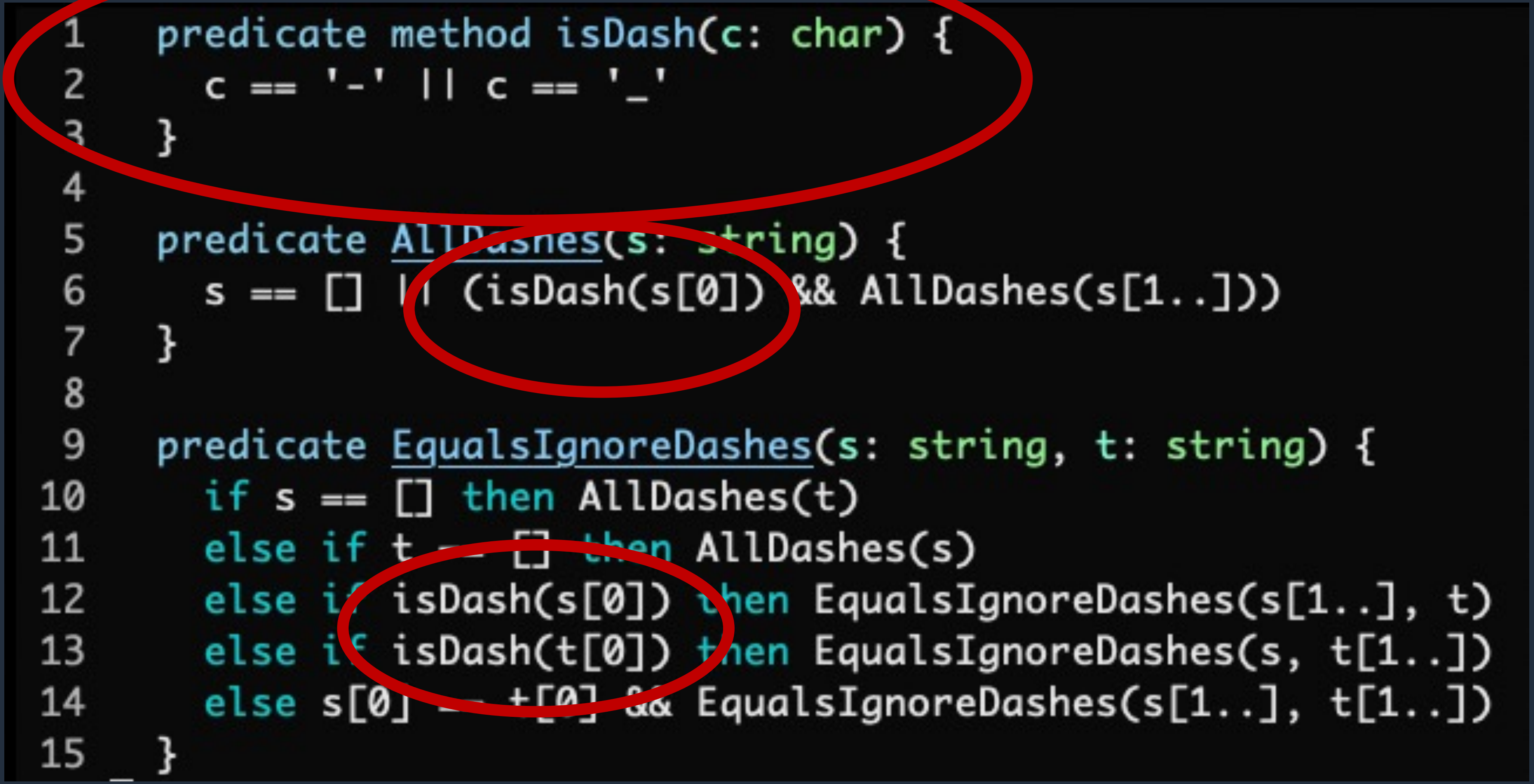
Specification



```
predicate AllDashes(s: string) {
  s == [] || (s[0] == '-' && AllDashes(s[1..]))
}

predicate EqualsIgnoreDashes(s: string, t: string) {
  if s == [] then AllDashes(t)
  else if t == [] then AllDashes(s)
  else if s[0] == '-' then EqualsIgnoreDashes(s[1..], t)
  else if t[0] == '-' then EqualsIgnoreDashes(s, t[1..])
  else s[0] == t[0] && EqualsIgnoreDashes(s[1..], t[1..])
}
```

© 2022, Amazon Web Services, Inc. or its affiliates.

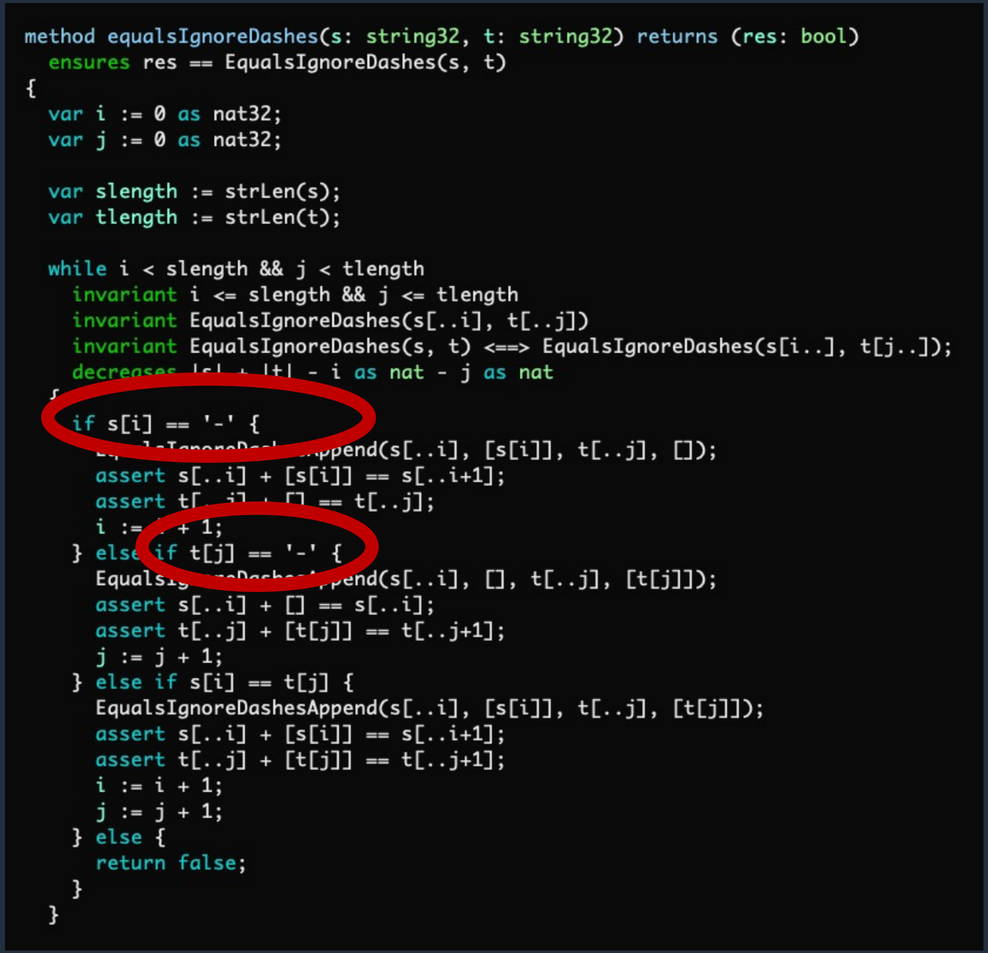
Specification



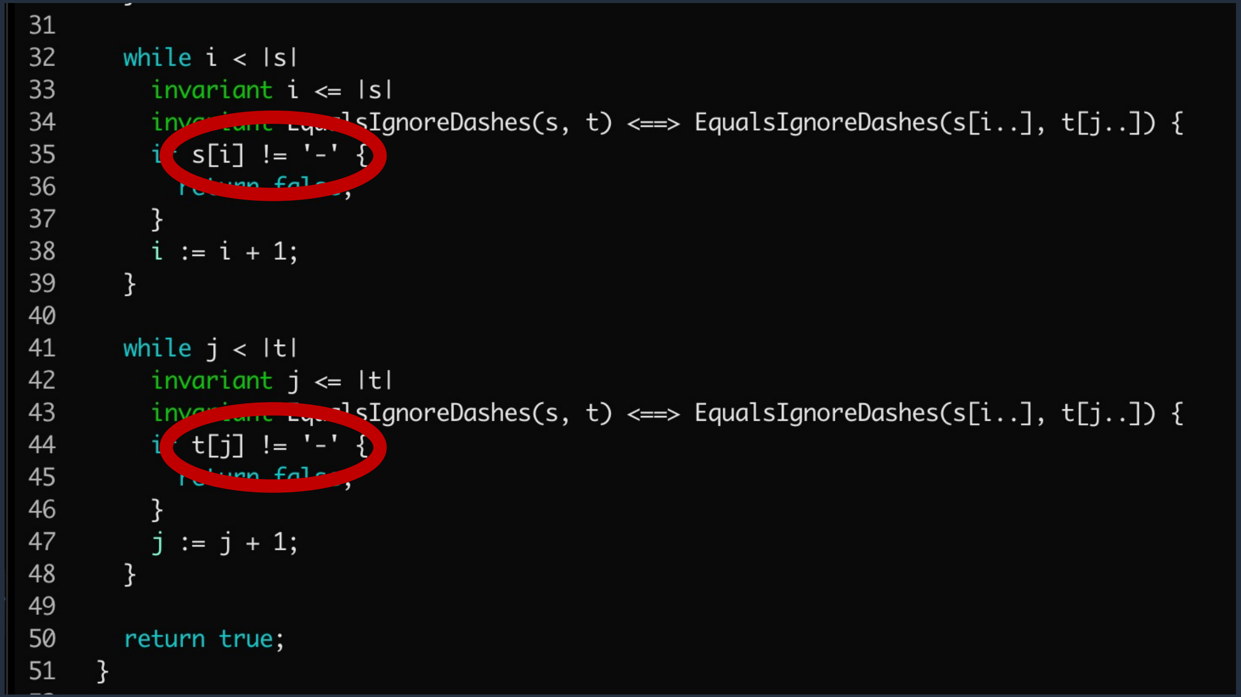
```
predicate method isDash(c: char) {
  c == '-' || c == '_'
}

predicate AllDashes(s: string) {
  s == [] || (isDash(s[0]) && AllDashes(s[1..]))
}

predicate EqualsIgnoreDashes(s: string, t: string) {
  if s == [] then AllDashes(t)
  else if t == [] then AllDashes(s)
  else if isDash(s[0]) then EqualsIgnoreDashes(s[1..], t)
  else if isDash(t[0]) then EqualsIgnoreDashes(s, t[1..])
  else s[0] == t[0] && EqualsIgnoreDashes(s[1..], t[1..])
}
```

© 2022, Amazon Web Services, Inc. or its affiliates.

Implementation

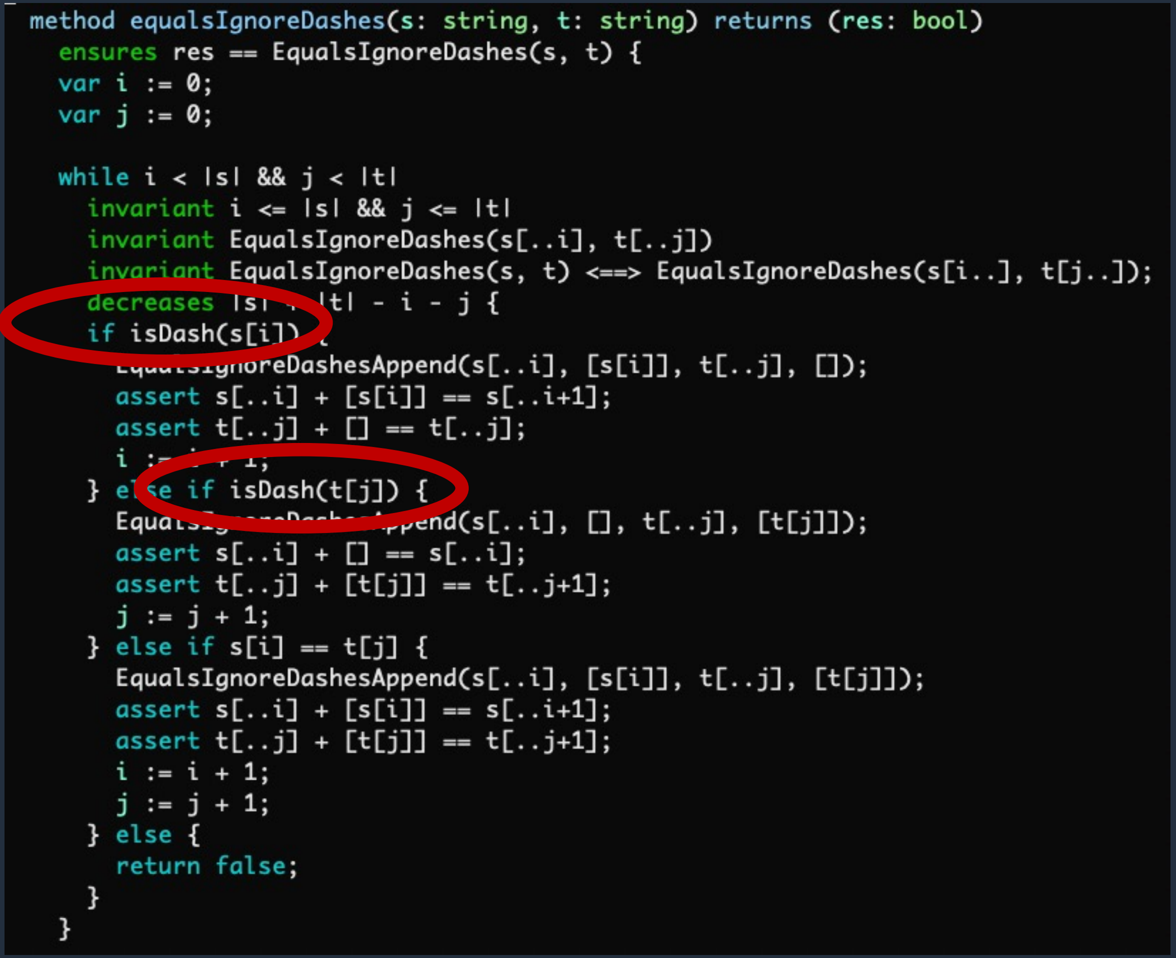


```
method equalsIgnoreDashes(s: string32, t: string32) returns (res: bool)
  ensures res == EqualsIgnoreDashes(s, t)
{
  var i := 0 as nat32;
  var j := 0 as nat32;

  var slength := strLen(s);
  var tlength := strLen(t);

  while i < slength && j < tlength
    invariant i <= slength && j <= tlength
    invariant EqualsIgnoreDashes(s[..i], t[..j])
    invariant EqualsIgnoreDashes(s, t) <==> EqualsIgnoreDashes(s[i..], t[j..]);
    decreases |s| + |t| - i as nat - j as nat
  {
    if s[i] == '-' {
      EqualsIgnoreDashesAppend(s[..i], [s[i]], t[..j], []);
      assert s[..i] + [s[i]] == s[..i+1];
      assert t[..j] + [] == t[..j];
      i := i + 1;
    } else if t[j] == '-' {
      EqualsIgnoreDashesAppend(s[..i], [], t[..j], [t[j]]);
      assert s[..i] + [] == s[..i];
      assert t[..j] + [t[j]] == t[..j+1];
      j := j + 1;
    } else if s[i] == t[j] {
      EqualsIgnoreDashesAppend(s[..i], [s[i]], t[..j], [t[j]]);
      assert s[..i] + [s[i]] == s[..i+1];
      assert t[..j] + [t[j]] == t[..j+1];
      i := i + 1;
      j := j + 1;
    } else {
      return false;
    }
  }
```

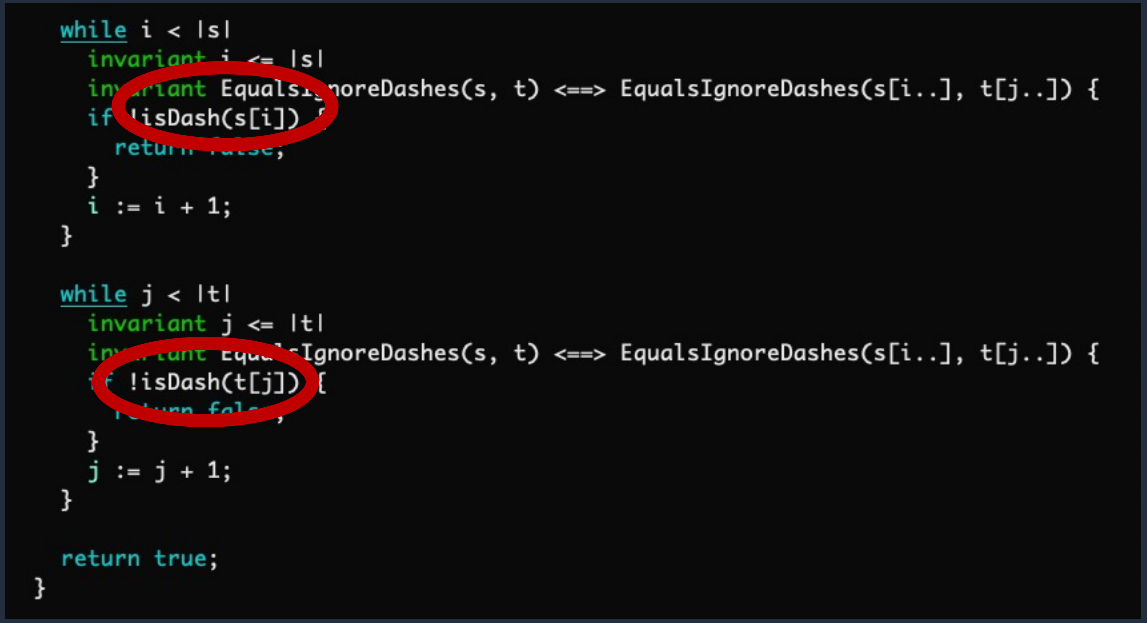


```
  while i < |s|
    invariant i <= |s|
    invariant EqualsIgnoreDashes(s, t) <==> EqualsIgnoreDashes(s[i..], t[j..]) {
    if s[i] != '-' {
      return false;
    }
    i := i + 1;
  }

  while j < |t|
    invariant j <= |t|
    invariant EqualsIgnoreDashes(s, t) <==> EqualsIgnoreDashes(s[i..], t[j..]) {
    if t[j] != '-' {
      return false;
    }
    j := j + 1;
  }

  return true;
}
```

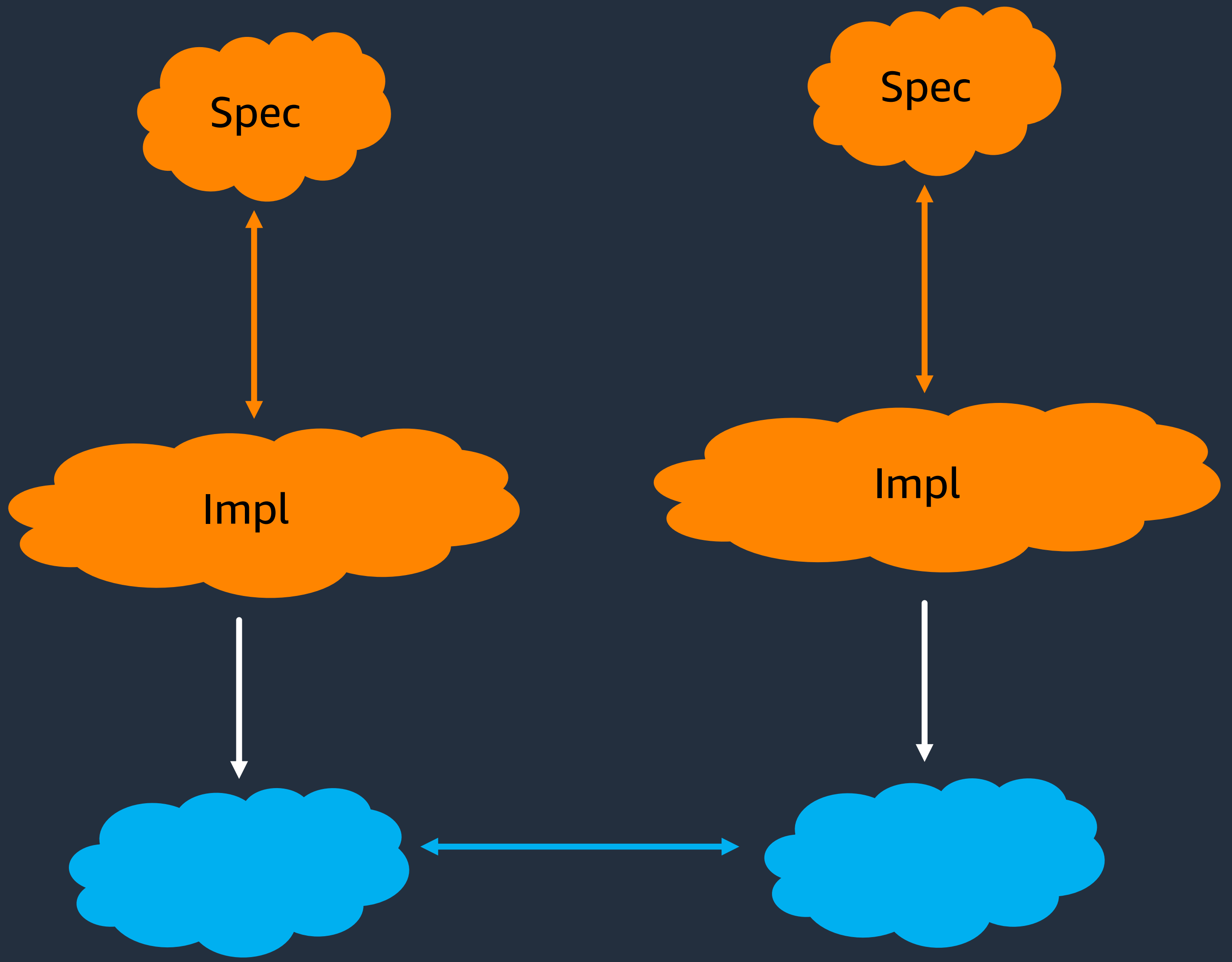
Implementation



```
method equalsIgnoreDashes(s: string, t: string) returns (res: bool)
  ensures res == EqualsIgnoreDashes(s, t) {
  var i := 0;
  var j := 0;

  while i < |s| && j < |t|
    invariant i <= |s| && j <= |t|
    invariant EqualsIgnoreDashes(s[..i], t[..j])
    invariant EqualsIgnoreDashes(s, t) <==> EqualsIgnoreDashes(s[i..], t[j..]);
    decreases |s| + |t| - i - j {
    if isDash(s[i]) {
      EqualsIgnoreDashesAppend(s[..i], [s[i]], t[..j], []);
      assert s[..i] + [s[i]] == s[..i+1];
      assert t[..j] + [] == t[..j];
      i := i + 1;
    } else if isDash(t[j]) {
      EqualsIgnoreDashesAppend(s[..i], [], t[..j], [t[j]]);
      assert s[..i] + [] == s[..i];
      assert t[..j] + [t[j]] == t[..j+1];
      j := j + 1;
    } else if s[i] == t[j] {
      EqualsIgnoreDashesAppend(s[..i], [s[i]], t[..j], [t[j]]);
      assert s[..i] + [s[i]] == s[..i+1];
      assert t[..j] + [t[j]] == t[..j+1];
      i := i + 1;
      j := j + 1;
    } else {
      return false;
    }
  }
```

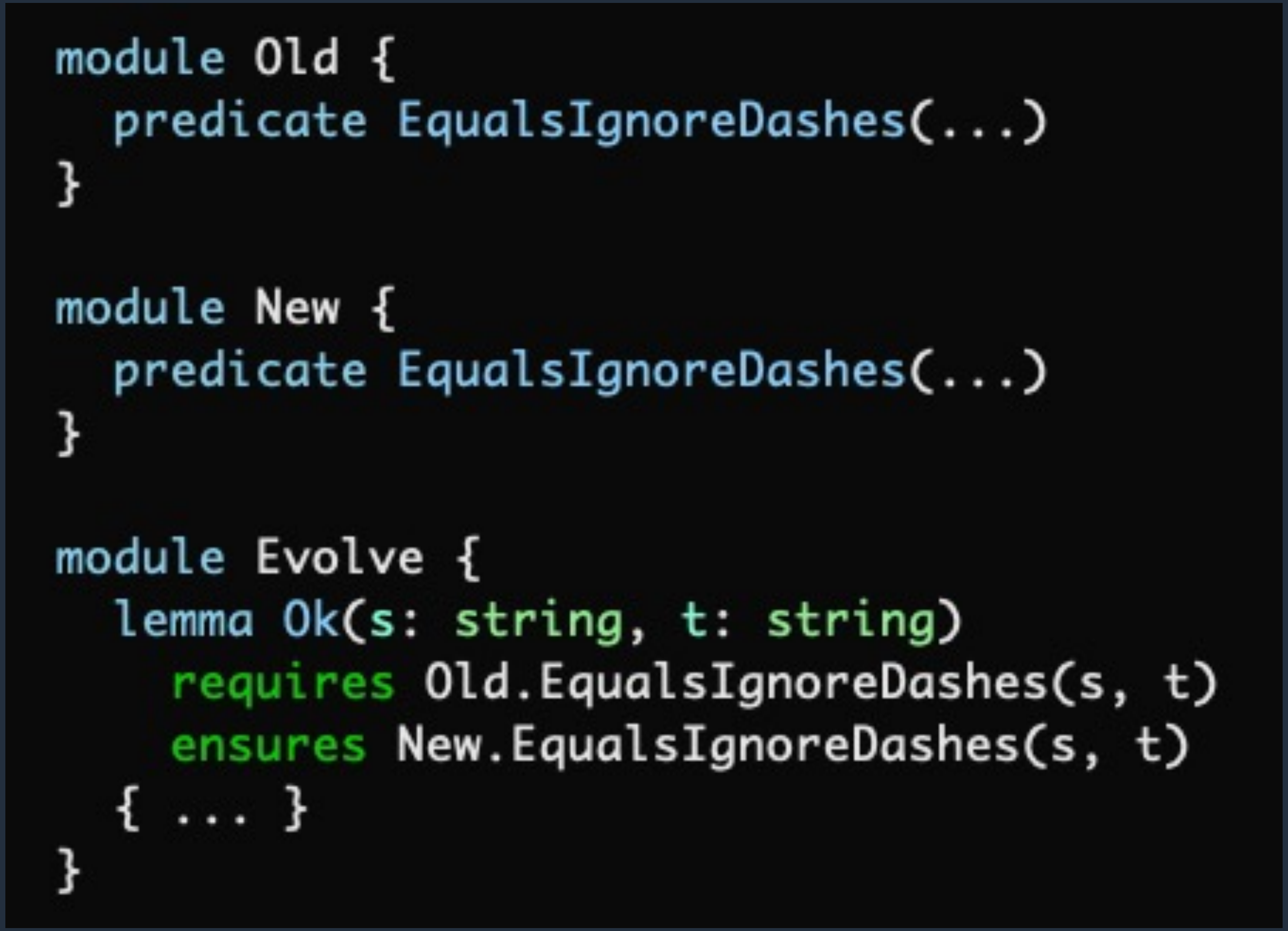


```
  while i < |s|
    invariant i <= |s|
    invariant EqualsIgnoreDashes(s, t) <==> EqualsIgnoreDashes(s[i..], t[j..]) {
    if !isDash(s[i]) {
      return false;
    }
    i := i + 1;
  }

  while j < |t|
    invariant j <= |t|
    invariant EqualsIgnoreDashes(s, t) <==> EqualsIgnoreDashes(s[i..], t[j..]) {
    if !isDash(t[j]) {
      return false;
    }
    j := j + 1;
  }

  return true;
}
```



© 2022, Amazon Web Services, Inc. or its affiliates.

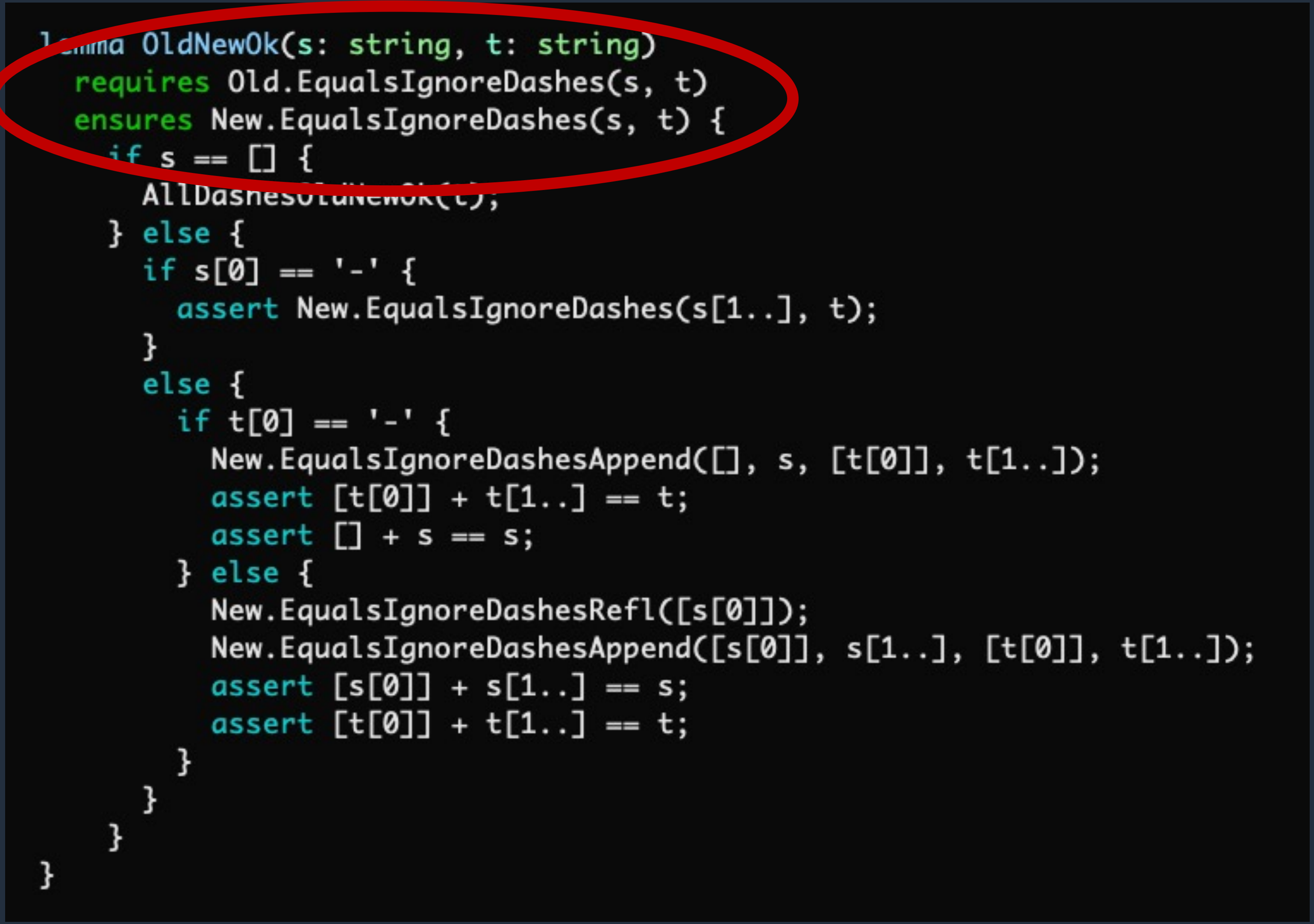
Proof Evolution



```
module Old {
  predicate EqualsIgnoreDashes(...)
}

module New {
  predicate EqualsIgnoreDashes(...)
}

module Evolve {
  lemma Ok(s: string, t: string)
    requires Old.EqualsIgnoreDashes(s, t)
    ensures New.EqualsIgnoreDashes(s, t)
  { ... }
}
```

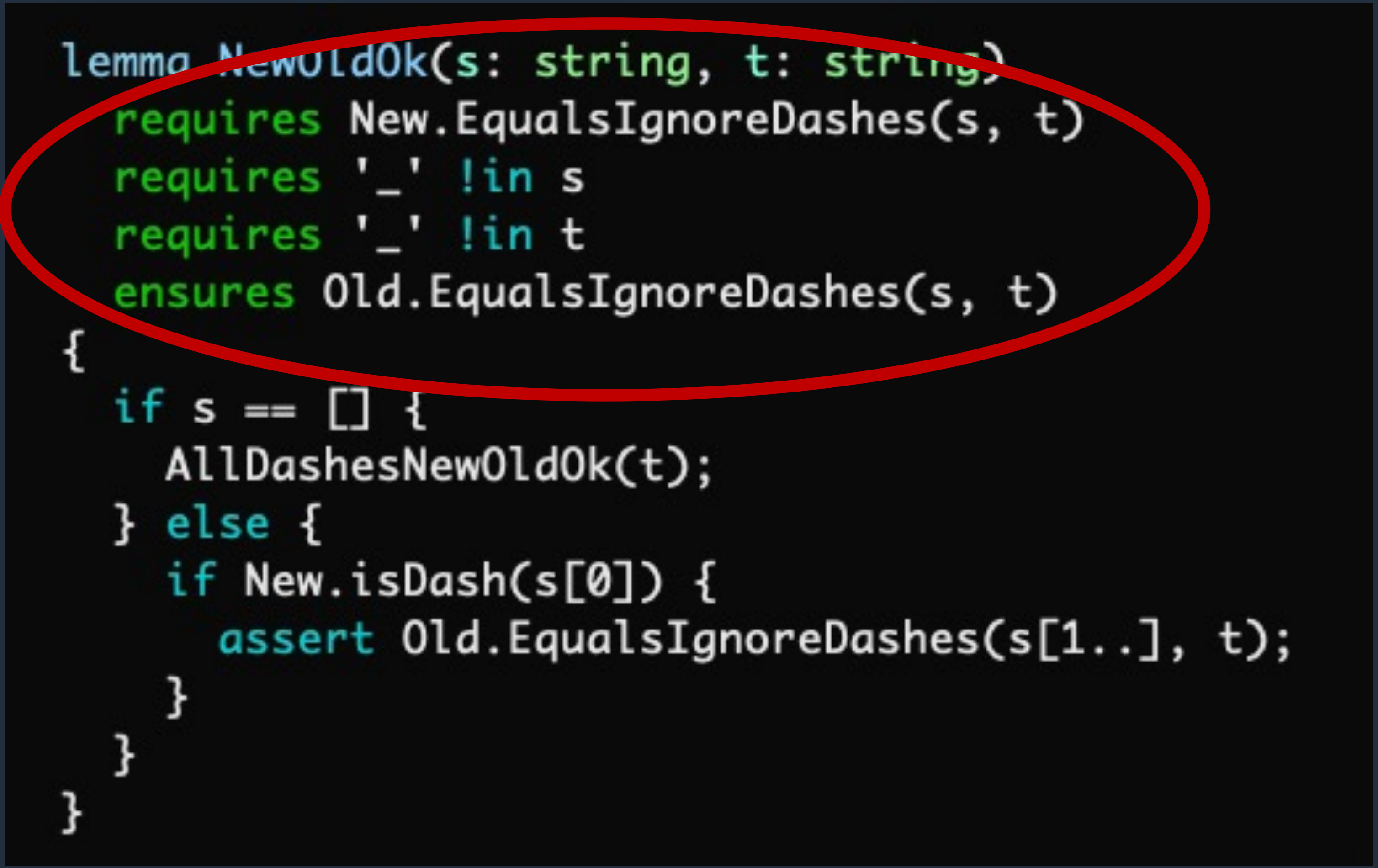
© 2022, Amazon Web Services, Inc. or its affiliates.

Proof Evolution



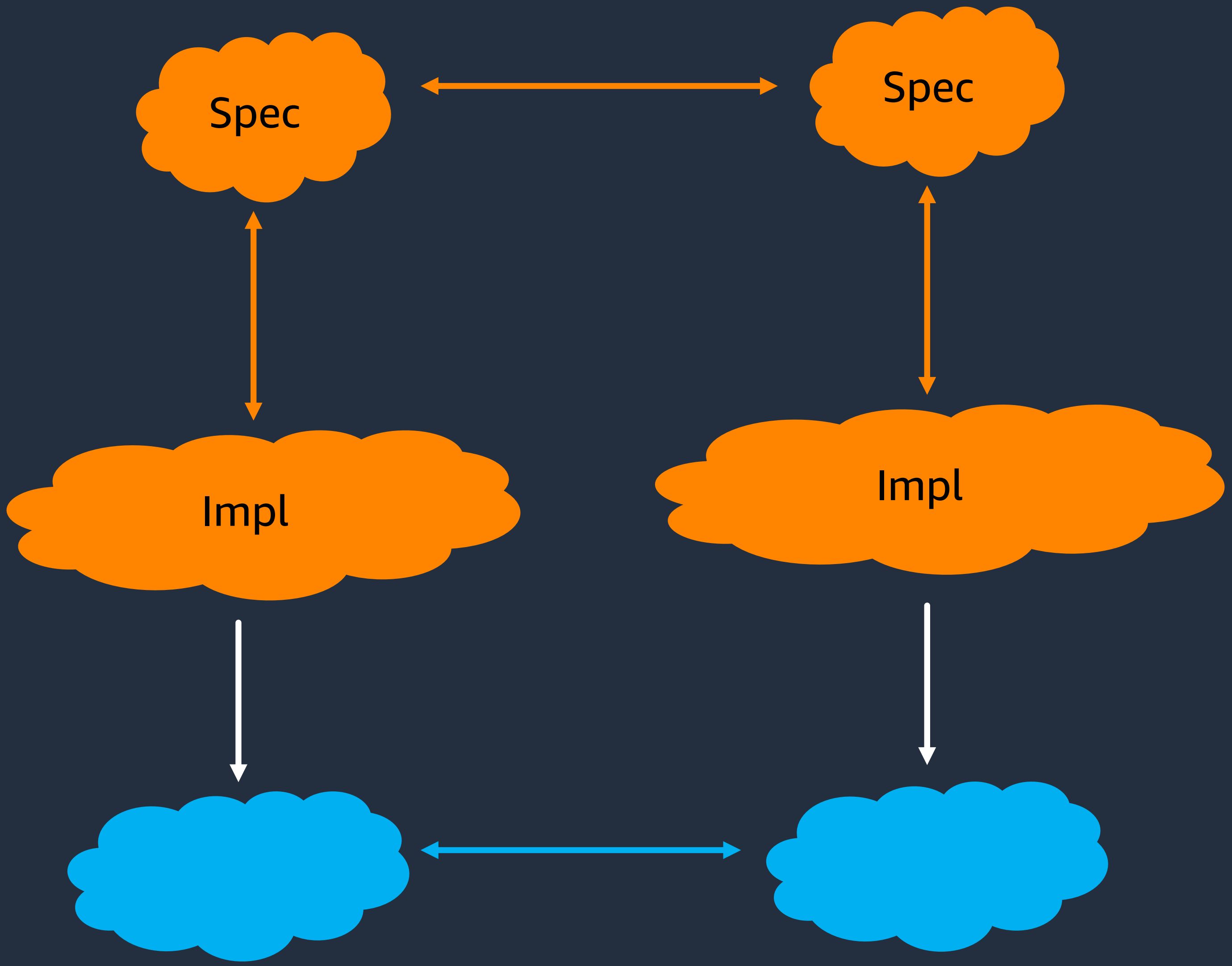
```
lemma OldNewOk(s: string, t: string)
  requires Old.EqualsIgnoreDashes(s, t)
  ensures New.EqualsIgnoreDashes(s, t) {
    if s == [] {
      AllDashesOldNewOk(t);
    } else {
      if s[0] == '-' {
        assert New.EqualsIgnoreDashes(s[1..], t);
      }
      else {
        if t[0] == '-' {
          New.EqualsIgnoreDashesAppend([], s, [t[0]], t[1..]);
          assert [t[0]] + t[1..] == t;
          assert [] + s == s;
        } else {
          New.EqualsIgnoreDashesRefl([s[0]]);
          New.EqualsIgnoreDashesAppend([s[0]], s[1..], [t[0]], t[1..]);
          assert [s[0]] + s[1..] == s;
          assert [t[0]] + t[1..] == t;
        }
      }
    }
}
```

Proof Evolution



```
lemma NewOldOk(s: string, t: string)
  requires New.EqualsIgnoreDashes(s, t)
  requires '_' !in s
  requires '_' !in t
  ensures Old.EqualsIgnoreDashes(s, t)
{
  if s == [] {
    AllDashesNewOldOk(t);
  } else {
    if New.isDash(s[0]) {
      assert Old.EqualsIgnoreDashes(s[1..], t);
    }
  }
}
```

© 2022, Amazon Web Services, Inc. or its affiliates.

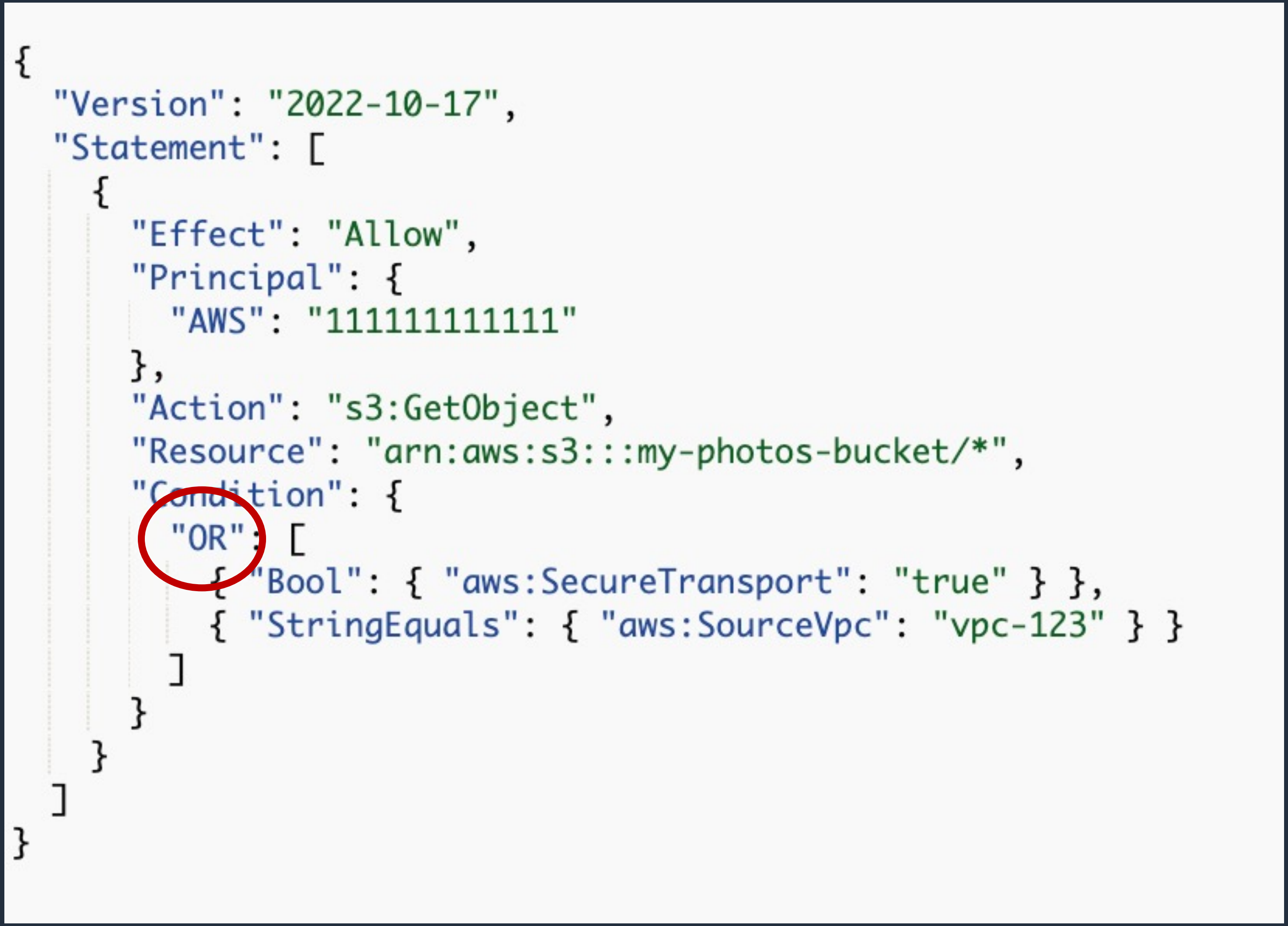


© 2022, Amazon Web Services, Inc. or its affiliates.

Conclusion

© 2022, Amazon Web Services, Inc. or its affiliates.

Making changes



```
{
  "Version": "2022-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111111111111"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::my-photos-bucket/*",
      "Condition": {
        "OR": [
          { "Bool": { "aws:SecureTransport": "true" } },
          { "StringEquals": { "aws:SourceVpc": "vpc-123" } }
        ]
      }
    }
  ]
}
```

© 2022, Amazon Web Services, Inc. or its affiliates.

Adding “OR”

Pros

- More direct, succinct policies

Cons

- Availability risk
- Security risk

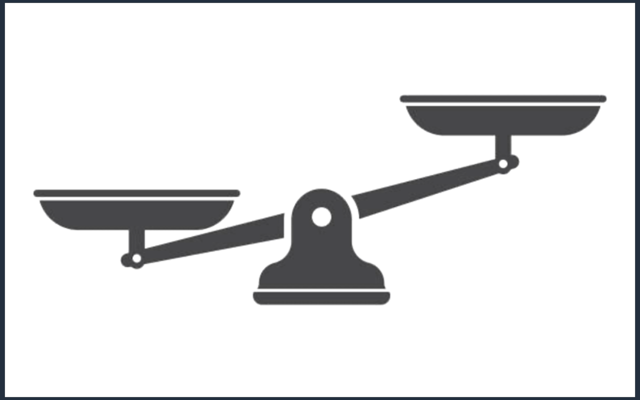
Adding “OR”

Pros

- More direct, succinct policies

Cons

- Need to prove backward compatibility



© 2022, Amazon Web Services, Inc. or its affiliates.

Formal verification enables faster evolution of critical systems

© 2022, Amazon Web Services, Inc. or its affiliates.

Customer-facing auth coming soon…

© 2022, Amazon Web Services, Inc. or its affiliates.

Thank you!

Sean McLaughlin

seanmcl@amazon.com

© 2022, Amazon Web Services, Inc. or its affiliates.