Patching with Matching

Alexander Schultheiß, Paul Bittner, Sandra Greiner,
Thomas Thüm, Timo Kehrer

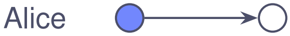
Paderborn University

April 11, 2024

Motivation: Change Propagation

Evolution of software variants

🔵 Feature A
🟠 Feature B

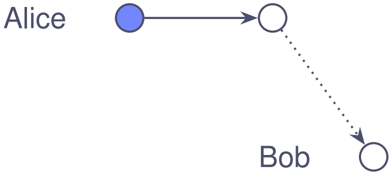


Alice

Evolution of software variants

🔵 Feature A

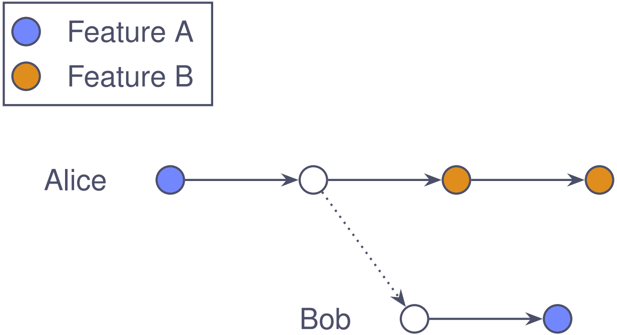
🟠 Feature B



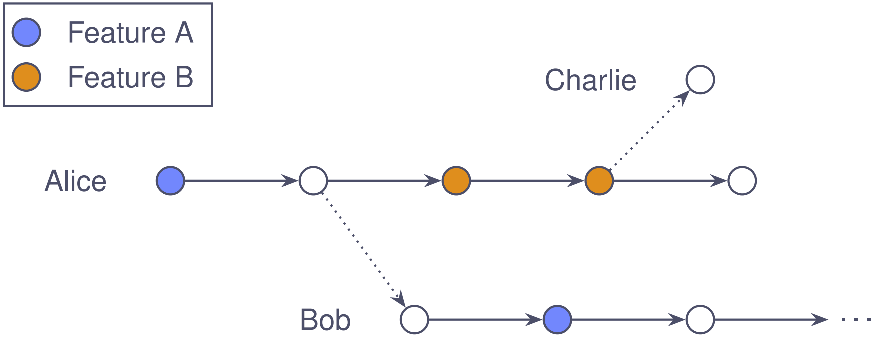
Alice

Bob

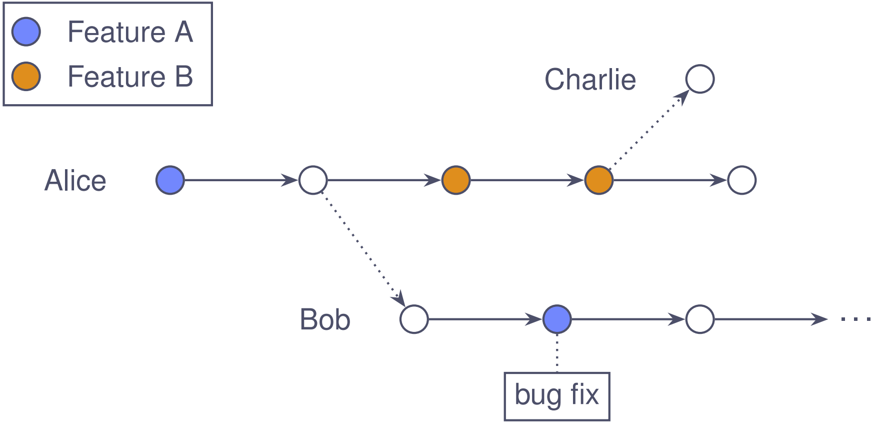
Evolution of software variants



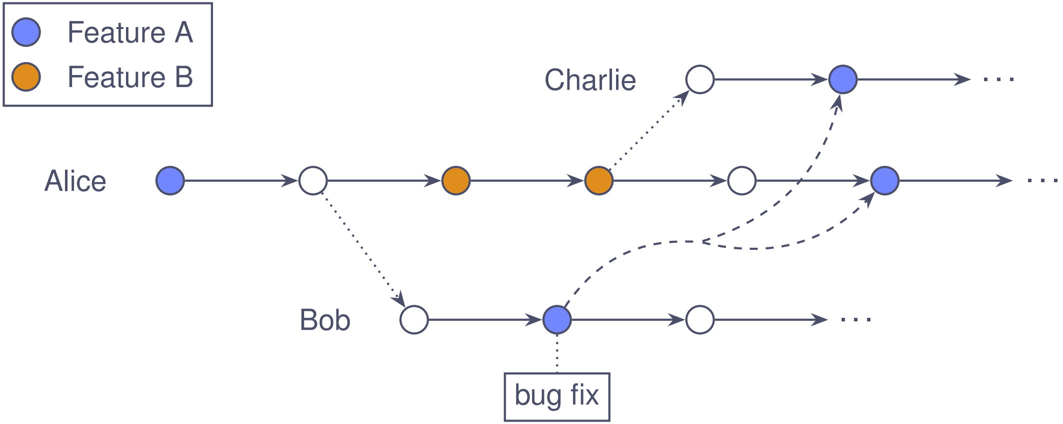
Evolution of software variants



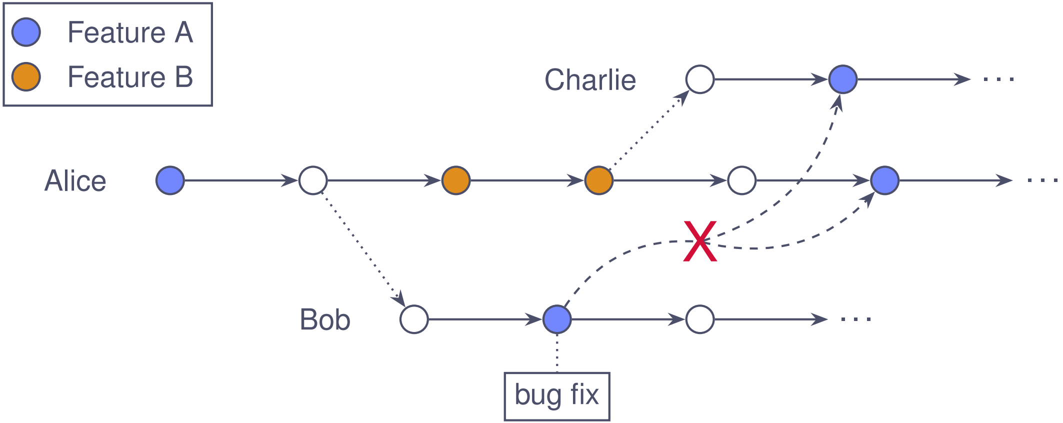
Evolution of software variants



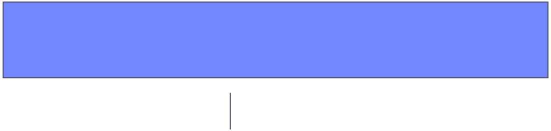
Evolution of software variants



Evolution of software variants

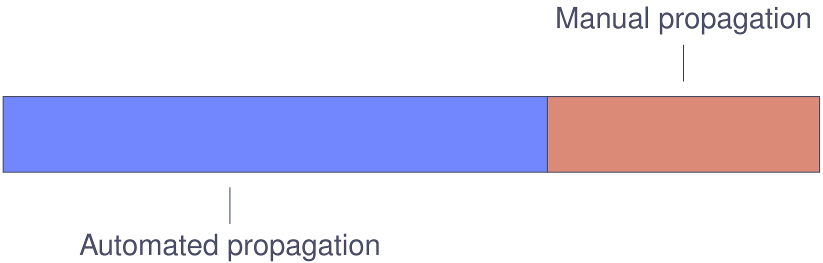


Goal: Reduce manual effort

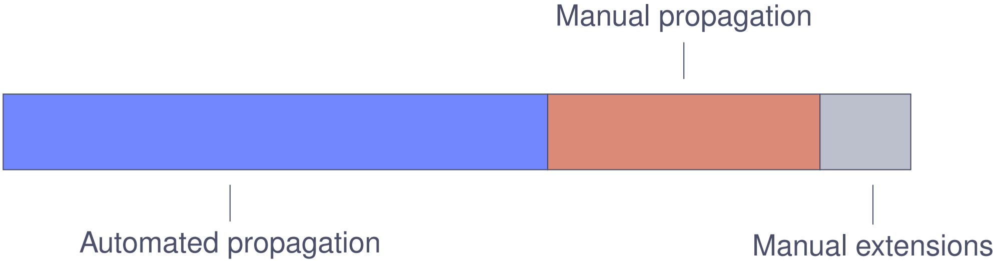


Automated propagation

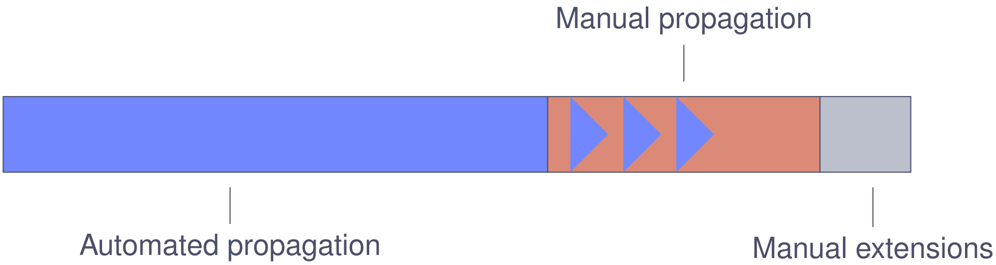
Goal: Reduce manual effort



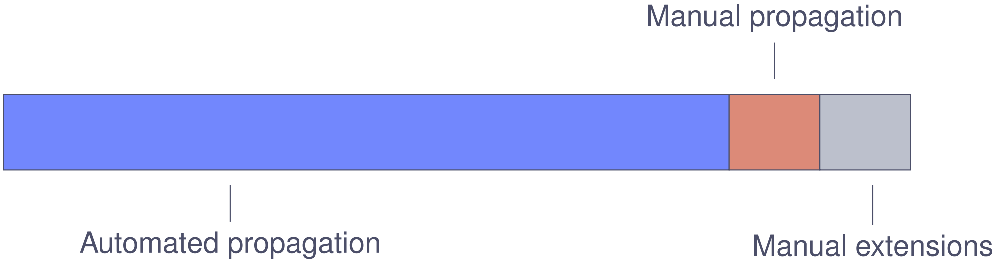
Goal: Reduce manual effort



Goal: Reduce manual effort



Goal: Reduce manual effort



Patching with Matching

Match source and target variants to align patch

```
  def welcome_message(username):
      greeting = f"Welcome, {username}!"
      return greeting

- def calculate_sum(num1, num2):
-     total = num1 + num2
-     return total

  def user_age_in_years(birth_year):
      current_year = 2024
+     age = current_year - birth_year
+     return age

  def multiply_numbers(x, y):
      product = x * y
      return product
```

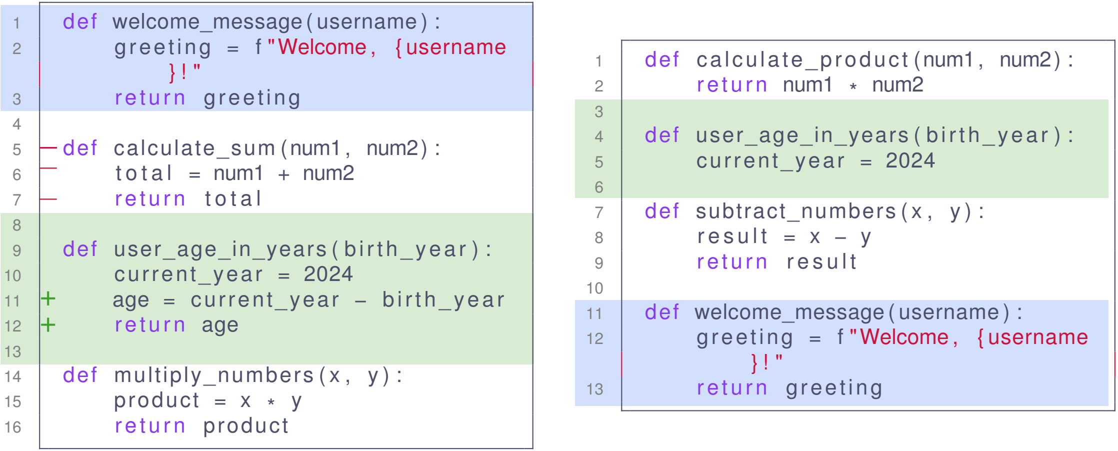
```
def calculate_product(num1, num2):
    return num1 * num2

def user_age_in_years(birth_year):
    current_year = 2024

def subtract_numbers(x, y):
    result = x - y
    return result

def welcome_message(username):
    greeting = f"Welcome, {username}!"
    return greeting
```

Match source and target variants to align patch



```
 def welcome_message(username):
     greeting = f"Welcome, {username}!"
     return greeting

-def calculate_sum(num1, num2):
-    total = num1 + num2
-    return total

 def user_age_in_years(birth_year):
     current_year = 2024
+    age = current_year - birth_year
+    return age

 def multiply_numbers(x, y):
     product = x * y
     return product
```

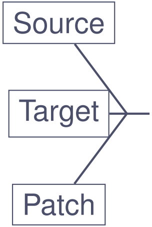
```
def calculate_product(num1, num2):
    return num1 * num2

def user_age_in_years(birth_year):
    current_year = 2024

def subtract_numbers(x, y):
    result = x - y
    return result

def welcome_message(username):
    greeting = f"Welcome, {username}!"
    return greeting
```

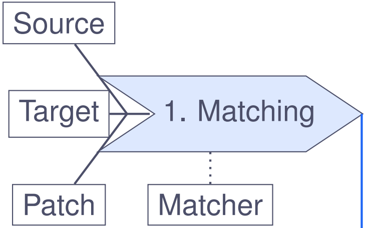
Overview of patching with matching



Input

- Source Variant (S)
- Target Variant (T)
- Patch (P)

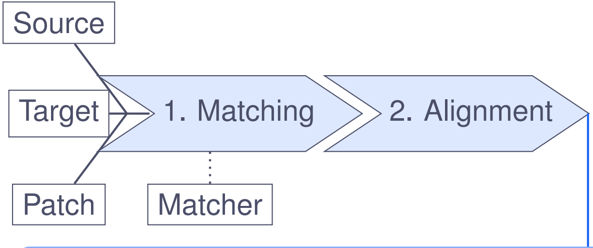
Overview of patching with matching



1. Matching

- Exchangable match function (e.g., LCS)
- Compare the source and target variant
- Determine a matching of common text (i.e., code)

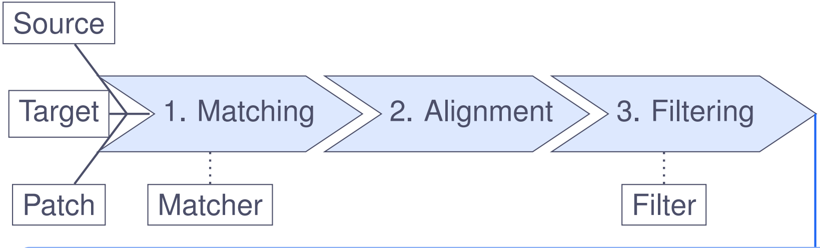
Overview of patching with matching



2. Alignment

- Align a patch to the target variant based on the matching
- Identifies the lines that should be removed
- Finds the best location for insertions

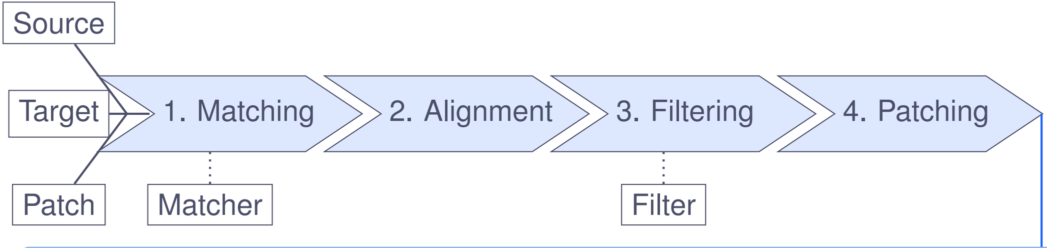
Overview of patching with matching



3. Filtering

- Exchangable filter function
- Filters the aligned patch
- e.g., filter based on affected features, or context similarity

Overview of patching with matching

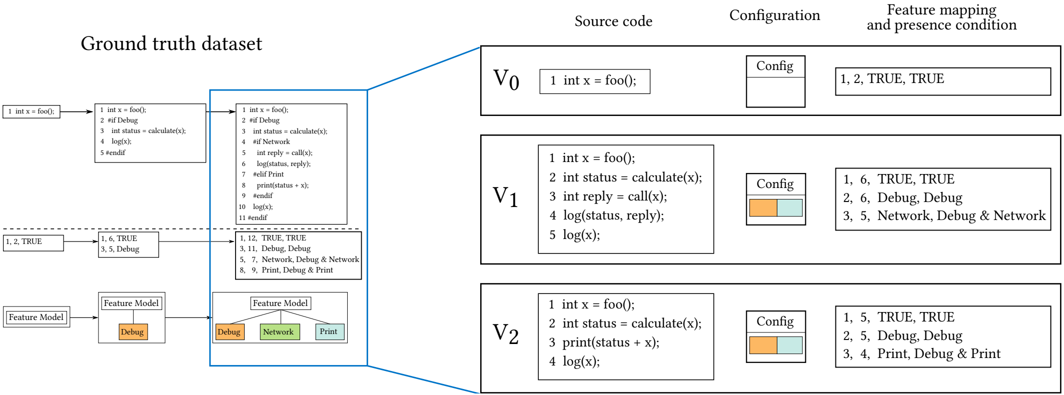


4. Patching

- Applies the prepared patch
- Changes are applied in their aligned order

Evaluation (In Progress)

1. Eval approach: SPL-based variant simulation



```
1  int x = foo();
```

```
1  int x = foo();
2  #if Debug
3    int status = calculate(x);
4    log(x);
5 #endif
```

```
1  int x = foo();
2  #if Debug
3    int status = calculate(x);
4    #if Network
5      int reply = call(x);
6      log(status, reply);
7    #elif Print
8      print(status + x);
9    #endif
10   log(x);
11 #endif
```

```
1, 2, TRUE
```

```
1, 6, TRUE
3, 5, Debug
```

```
1, 12,  TRUE, TRUE
3, 11,  Debug, Debug
5,   7,  Network, Debug & Network
8,   9,  Print, Debug & Print
```

V0

```
1  int x = foo();
```

```
1, 2, TRUE, TRUE
```

V1

```
1  int x = foo();
2  int status = calculate(x);
3  int reply = call(x);
4  log(status, reply);
5  log(x);
```

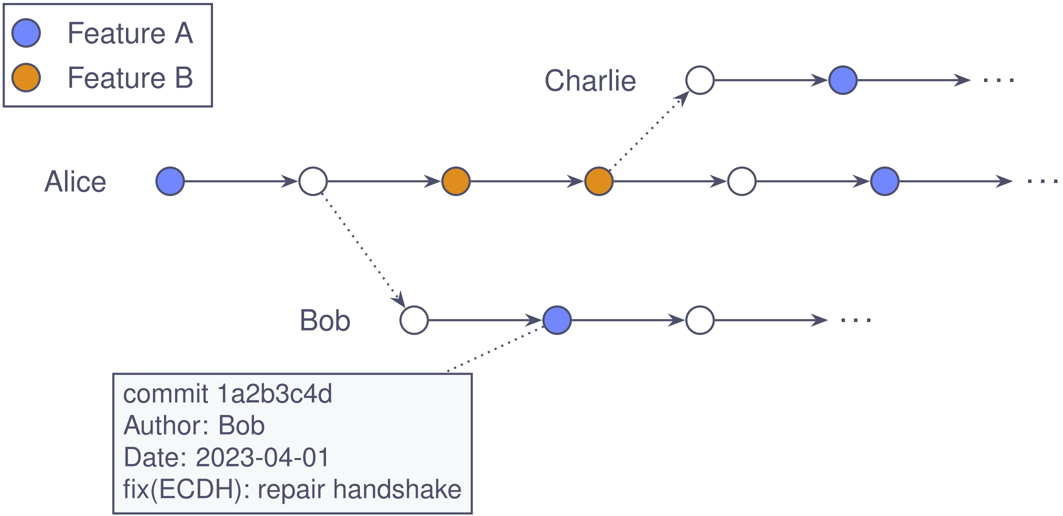
```
1,  6,  TRUE, TRUE
2,  6,  Debug, Debug
3,  5,  Network, Debug & Network
```

V2

```
1  int x = foo();
2  int status = calculate(x);
3  print(status + x);
4  log(x);
```

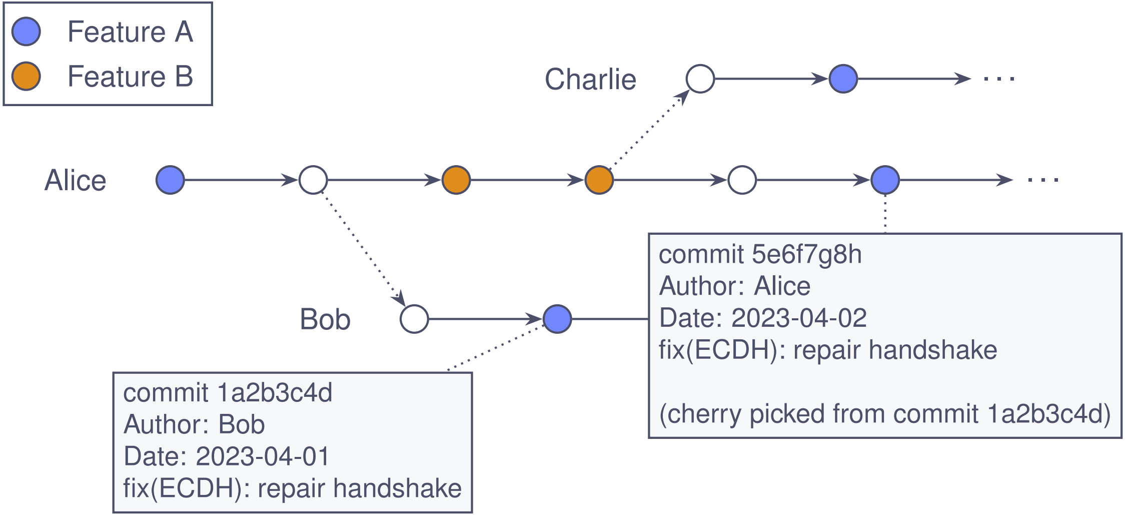
```
1,  5,  TRUE, TRUE
2,  5,  Debug, Debug
3,  4,  Print, Debug & Print
```

2. Eval approach: replay of real cherry picks



commit 1a2b3c4d
Author: Bob
Date: 2023-04-01
fix(ECDH): repair handshake

2. Eval approach: replay of real cherry picks



```
commit 1a2b3c4d
Author: Bob
Date: 2023-04-01
fix(ECDH): repair handshake
```

```
commit 5e6f7g8h
Author: Alice
Date: 2023-04-02
fix(ECDH): repair handshake

(cherry picked from commit 1a2b3c4d)
```

patching with matching vs. baseline

SPL variants (simulation)

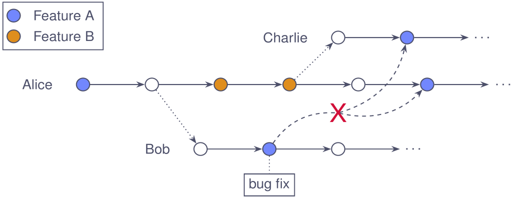
- $\sim$ Precision
- $\upuparrows$ Recall

Real cherry picks (replay)

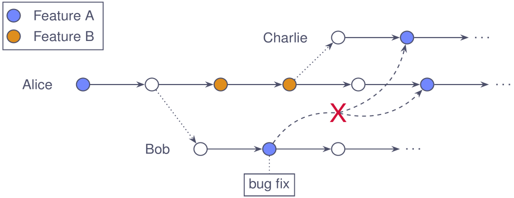
- $\sim$ Precision
- $\upuparrows$ Recall

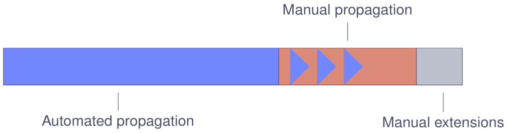
Estimated propagation quality relative to baseline approaches

Summary

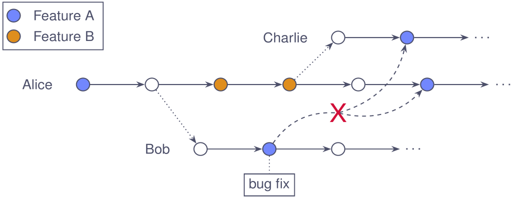


Summary

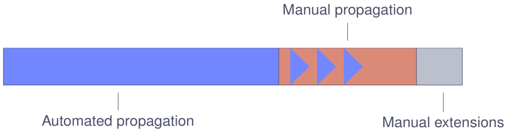

Feature A
Feature B
Charlie
Alice
Bob
bug fix


Manual propagation
Automated propagation
Manual extensions

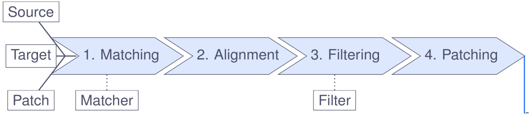
Summary



Feature A
Feature B
Charlie
Alice
Bob
bug fix
X

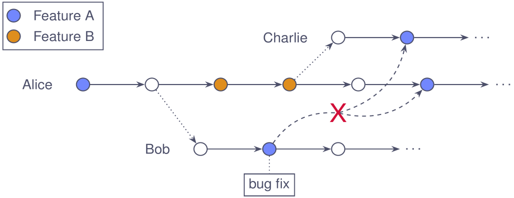


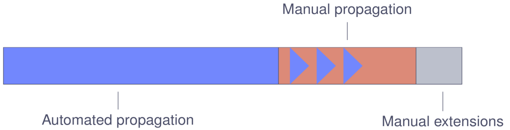
Manual propagation
Automated propagation
Manual extensions

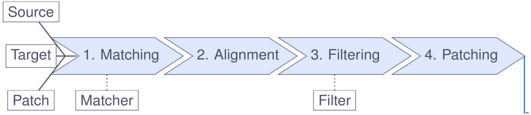


Source
Target
Patch
1. Matching
2. Alignment
3. Filtering
4. Patching
Matcher
Filter

Summary







SPL evolution replay

- SPL variants (no unintentional variant drift)
- simulated scenarios
- known variant drift

Cherry pick replay

- clone-and-own variants and versions
- real scenarios
- unknown variant drift

Conflicts during change propagation

```
@@ -4,6 +4,9 @@
    case SSH_KEX_DH_GROUP16_SHA512:
      rc = ssh_client_dh_init(session);
      break;
+   case SSH_KEX_CURVE25519_SHA256:
+     rc = curve25519_init(session);
+     break;
    case SSH_KEX_DH_GEX_SHA256:
      rc = ssh_client_dhgex_init(session);
      break;
```

```
int handshake(ssh_session session) {
  int rc = SSH_AGAIN;
  switch(session->next_crypto->kex_type){
    case SSH_KEX_DH_GROUP16_SHA512:
      rc = ssh_client_dh_init(session);
      break;
    case SSH_KEX_ECDH_SHA2_NISTP256:
      rc = ssh_client_ecdh_init(session);
      break;
    default:
      rc = SSH_ERROR;
  }
  return rc;
}


// more code
```

Conflicts during change propagation

```
@@ -4,6 +4,9 @@
    case SSH_KEX_DH_GROUP16_SHA512:
      rc = ssh_client_dh_init(session);
      break;
+   case SSH_KEX_CURVE25519_SHA256:
+     rc = curve25519_init(session);
+     break;
    case SSH_KEX_DH_GEX_SHA256:
      rc = ssh_client_dhgex_init(session);
      break;
```

```
int handshake(ssh_session session) {
  int rc = SSH_AGAIN;
  switch(session->next_crypto->kex_type){
    case SSH_KEX_DH_GROUP16_SHA512:
      rc = ssh_client_dh_init(session);
      break;
    case SSH_KEX_ECDH_SHA2_NISTP256:
      rc = ssh_client_ecdh_init(session);
      break;
    default:
      rc = SSH_ERROR;
  }
  return rc;
}


// more code
```

Conflicts during change propagation

```
@@ -4,6 +4,9 @@
      case SSH_KEX_DH_GROUP16_SHA512:
        rc = ssh_client_dh_init(session);
        break;
+     case SSH_KEX_CURVE25519_SHA256:
+       rc = curve25519_init(session);
+       break;
      case SSH_KEX_DH_GEX_SHA256:
        rc = ssh_client_dhgex_init(session);
        break;
```

```
int handshake(ssh_session session) {
  int rc = SSH_AGAIN;
  switch(session->next_crypto->kex_type){
    case SSH_KEX_DH_GROUP16_SHA512:
      rc = ssh_client_dh_init(session);
      break;
    case SSH_KEX_ECDH_SHA2_NISTP256:
      rc = ssh_client_ecdh_init(session);
      break;
    default:
      rc = SSH_ERROR;
  }
  return rc;
}


// more code
```

Conflicts during change propagation

```
@@ -4,6 +4,9 @@
    case SSH_KEX_DH_GROUP16_SHA512:
      rc = ssh_client_dh_init(session);
      break;
+   case SSH_KEX_CURVE25519_SHA256:
+     rc = curve25519_init(session);
+     break;
    case SSH_KEX_DH_GEX_SHA256:
      rc = ssh_client_dhgex_init(session);
      break;
```

```
int handshake(ssh_session session) {
  int rc = SSH_AGAIN;
  switch(session->next_crypto->kex_type){
    case SSH_KEX_DH_GROUP16_SHA512:
      rc = ssh_client_dh_init(session);
      break;
    case SSH_KEX_ECDH_SHA2_NISTP256:
      rc = ssh_client_ecdh_init(session);
      break;
    default:
      rc = SSH_ERROR;
  }
  return rc;
}


// more code
```

Eval approaches complement one another

SPL evolution replay

- SPL variants (no unintentional variant drift)
- simulated scenarios
- known variant drift

Cherry pick replay

- clone-and-own variants and versions
- real scenarios
- unknown variant drift

Two code variants

```c
int handshake(ssh_session session) {
  int rc = SSH_AGAIN;
  switch(session->next_crypto->kex_type){
    case SSH_KEX_DH_GROUP16_SHA512:
      rc = ssh_client_dh_init(session);
      break;
    case SSH_KEX_DH_GEX_SHA256:
      rc = ssh_client_dhgex_init(session);
      break;
    default:
      rc = SSH_ERROR;
  }
  return rc;
}
```

```c
int handshake(ssh_session session) {
  int rc = SSH_AGAIN;
  switch(session->next_crypto->kex_type){
    case SSH_KEX_DH_GROUP16_SHA512:
      rc = ssh_client_dh_init(session);
      break;
    case SSH_KEX_ECDH_SHA2_NISTP256:
      rc = ssh_client_ecdh_init(session);
      break;
    default:
      rc = SSH_ERROR;
  }
  return rc;
}
```

Handling a new case

```
int handshake(ssh_session session) {
  int rc = SSH_AGAIN;
  switch(session->next_crypto->
      kex_type){
    case SSH_KEX_DH_GROUP16_SHA512:
      rc = ssh_client_dh_init(session
          );
      break;
+   case SSH_KEX_CURVE25519_SHA256:
+     rc = curve25519_init(session);
+     break;
    case SSH_KEX_DH_GEX_SHA256:
      rc = ssh_client_dhgex_init(
          session);
      break;
    default:
      rc = SSH_ERROR;
  }
  return rc;
}
```

```
int handshake(ssh_session session) {
  int rc = SSH_AGAIN;
  switch(session->next_crypto->
      kex_type){
    case SSH_KEX_DH_GROUP16_SHA512:
      rc = ssh_client_dh_init(session
          );
      break;
    case SSH_KEX_ECDH_SHA2_NISTP256:
      rc = ssh_client_ecdh_init(
          session);
      break;
    default:
      rc = SSH_ERROR;
  }
  return rc;
}


// more code
```