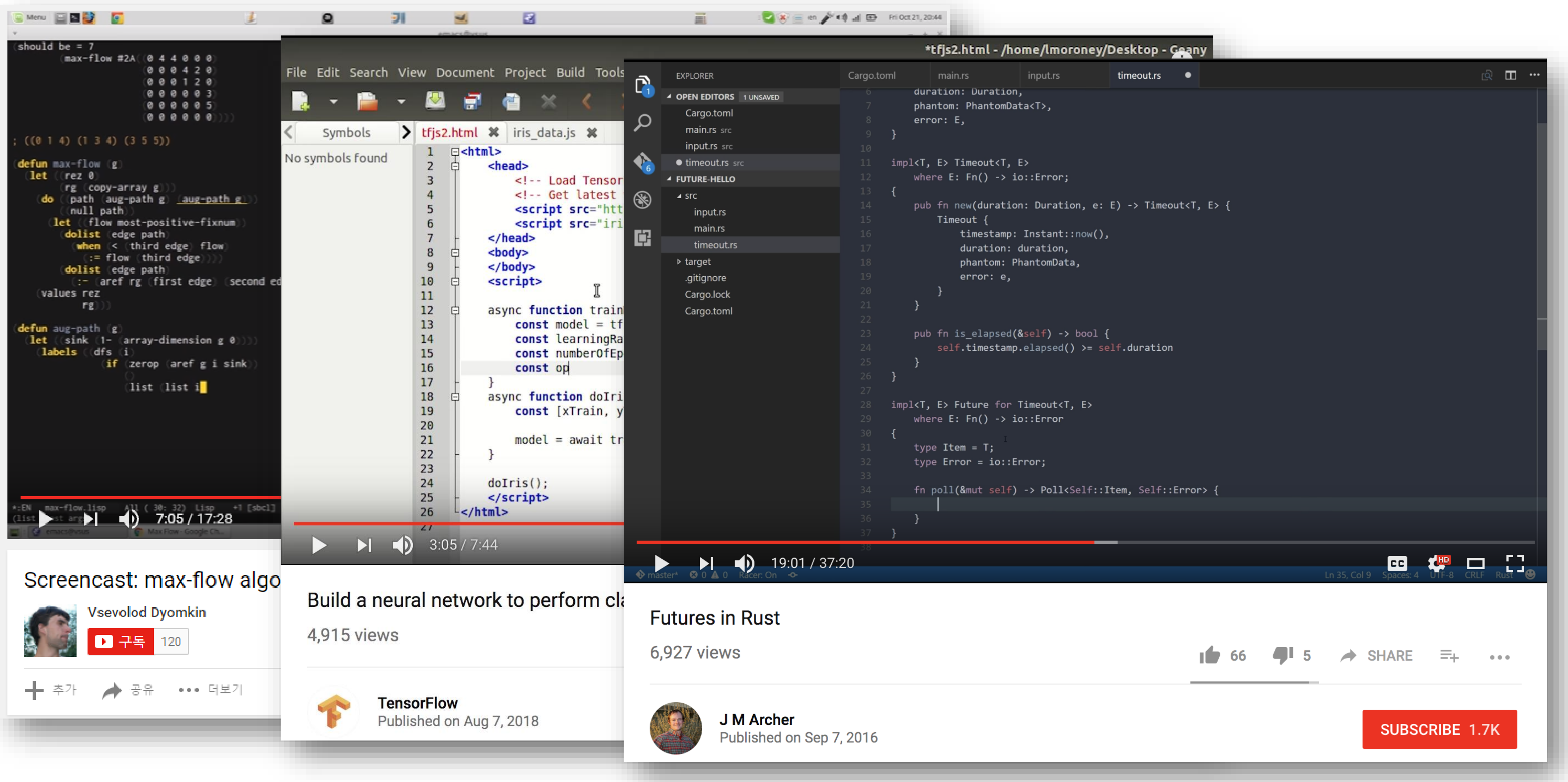
Non-Linear Editing of Text-Based Screencasts

Jungkook Park*	KAIST
Yeong Hoon Park*	University of Minnesota
Alice Oh	KAIST

Presenter : Yeong Hoon Park

10/16/2018, UIST 2018

Instructional Screencasts are Increasingly Popular



```
(should be = 7
        (max-flow #2A((0 4 4 0 0 0)
                      (0 0 0 4 2 0)
                      (0 0 0 1 2 0)
                      (0 0 0 0 0 3)
                      (0 0 0 0 0 5)
                      (0 0 0 0 0 0))))

; ((0 1 4) (1 3 4) (3 5 5))

(defun max-flow (g)
  (let ((rez 0)
        (rg (copy-array g)))
    (do ((path (aug-path g) (aug-path g)))
        ((null path))
      (let ((flow most-positive-fixnum))
        (dolist (edge path)
          (when (< (third edge) flow)
            (:= flow (third edge))))
        (dolist (edge path)
          (:- (aref rg (first edge) (second ed
    (values rez
            rg)))

(defun aug-path (g)
  (let ((sink (1- (array-dimension g 0))))
    (labels ((dfs (i)
               (if (zerop (aref g i sink))
                   ()
                   (list (list i
```

7:05 / 17:28

Screencast: max-flow algo

Vsevolod Dyomkin

구독 120

추가 공유 ••• 더보기

*tfjs2.html - /home/lmoroney/Desktop - Geany

```
<html>
    <head>
        <!-- Load Tensor
        <!-- Get latest
        <script src="htt
        <script src="iri
    </head>
    <body>
    </body>
    <script>

    async function train
        const model = tf
        const learningRa
        const numberOfEp
        const op
    }
    async function doIri
        const [xTrain, y

        model = await tr
    }

    doIris();
    </script>
</html>
```

3:05 / 7:44

Build a neural network to perform cl

4,915 views

TensorFlow
Published on Aug 7, 2018

```
    duration: Duration,
    phantom: PhantomData<T>,
    error: E,
}

impl<T, E> Timeout<T, E>
    where E: Fn() -> io::Error;
{
    pub fn new(duration: Duration, e: E) -> Timeout<T, E> {
        Timeout {
            timestamp: Instant::now(),
            duration: duration,
            phantom: PhantomData,
            error: e,
        }
    }

    pub fn is_elapsed(&self) -> bool {
        self.timestamp.elapsed() >= self.duration
    }
}

impl<T, E> Future for Timeout<T, E>
    where E: Fn() -> io::Error
{
    type Item = T;
    type Error = io::Error;

    fn poll(&mut self) -> Poll<Self::Item, Self::Error> {

    }
}
```

19:01 / 37:20

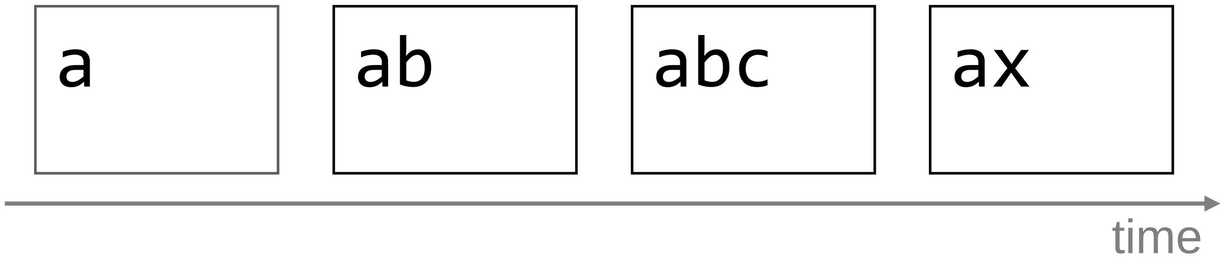
Futures in Rust

6,927 views

J M Archer
Published on Sep 7, 2016

SUBSCRIBE 1.7K

Screencast frames:



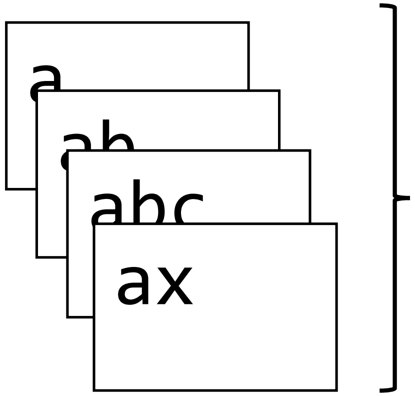


Image frames
(h.264, …)

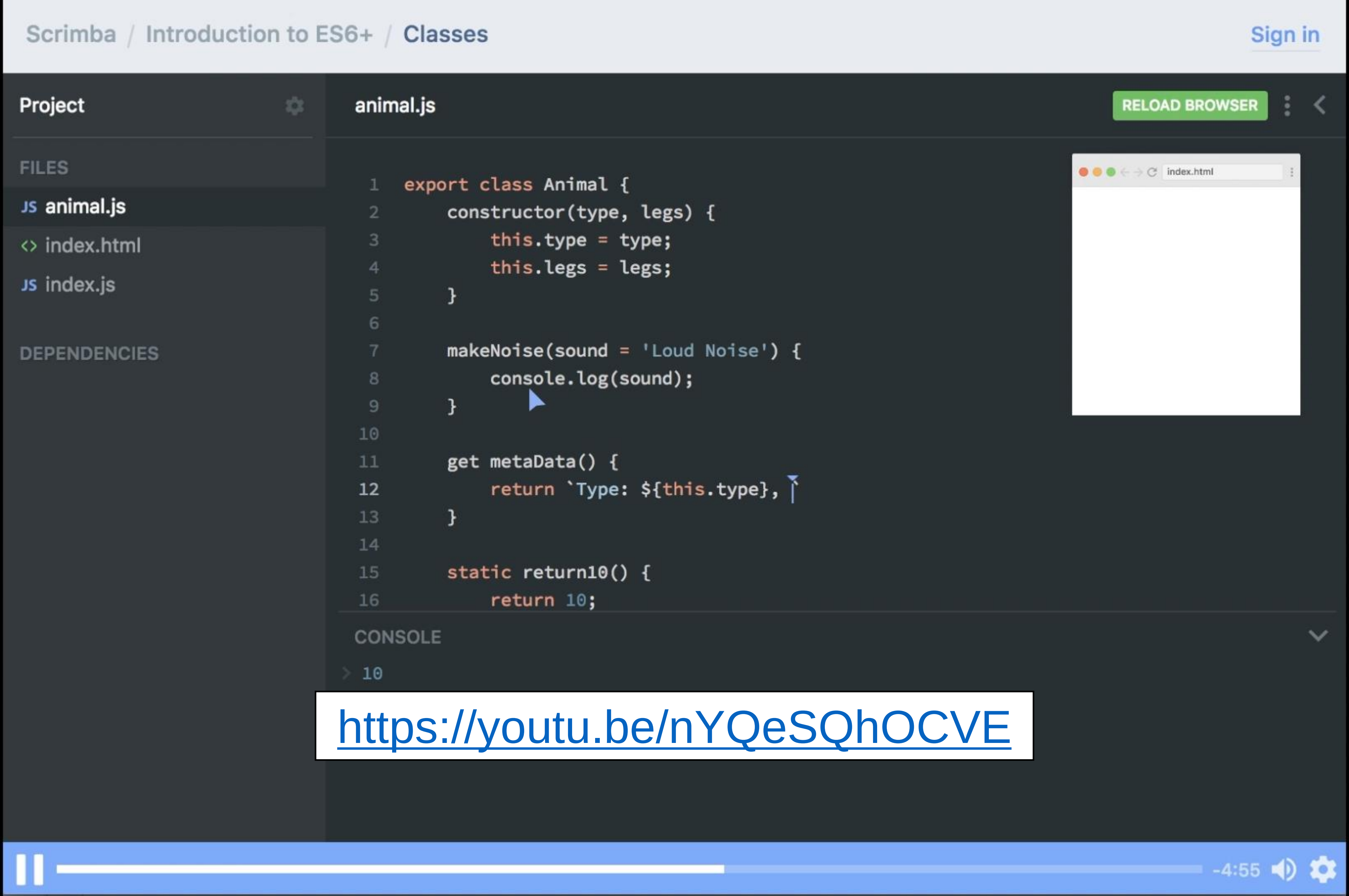
Video format
records screen

```
δ(0, 0, “a”) = a
δ(1, 1, “b”) = ab
δ(2, 2, “c”) = abc
δ(1, 3, “x”) = ax
```

Text edit operations
(OT, …)

Text-based screencast
records text changes

*OT: Operational Transformation



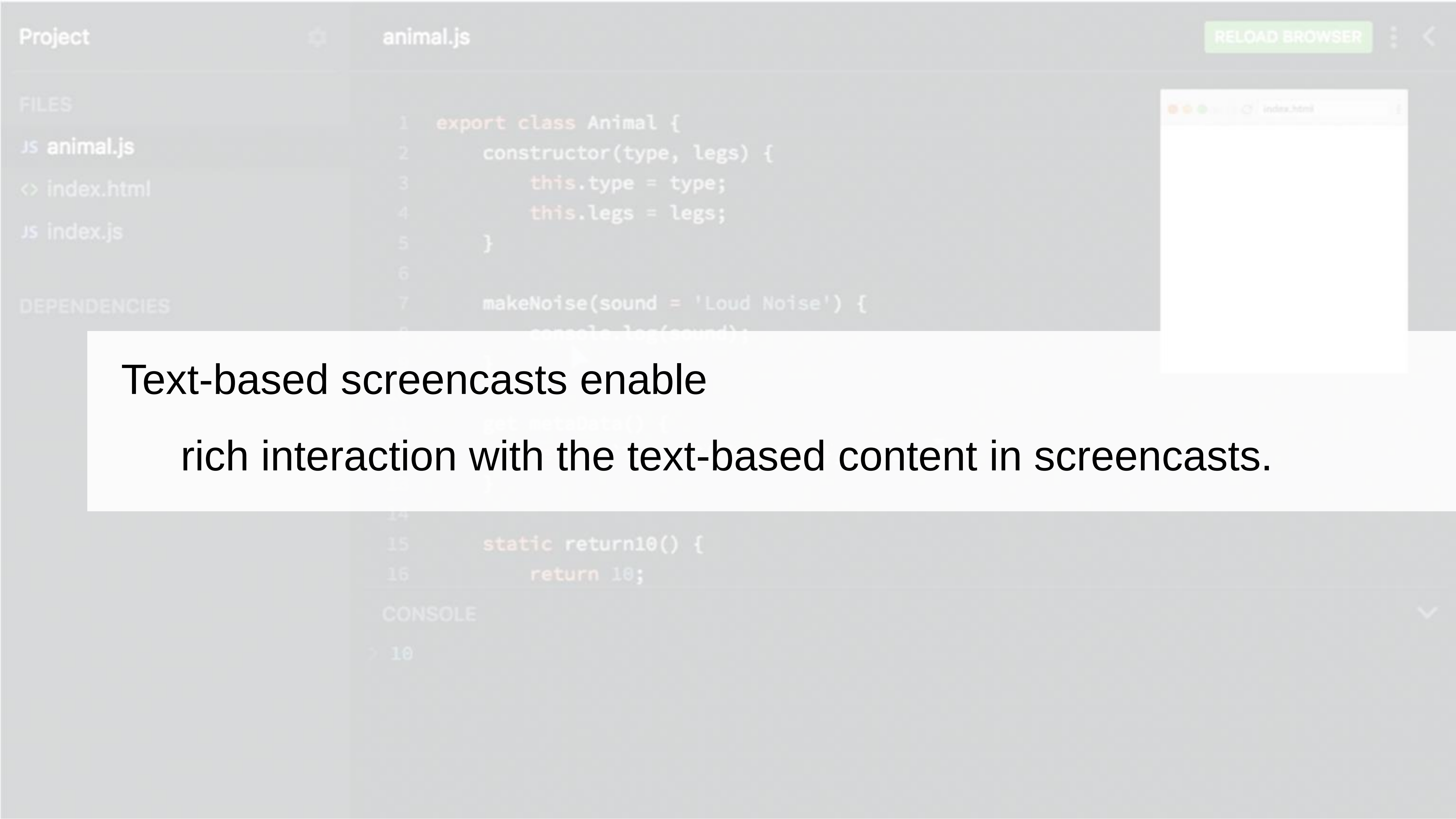
```
export class Animal {
    constructor(type, legs) {
        this.type = type;
        this.legs = legs;
    }

    makeNoise(sound = 'Loud Noise') {
        console.log(sound);
    }

    get metaData() {
        return `Type: ${this.type}, 
    }

    static return10() {
        return 10;
```

```
> 10
```

https://youtu.be/nYQeSQhOCVE

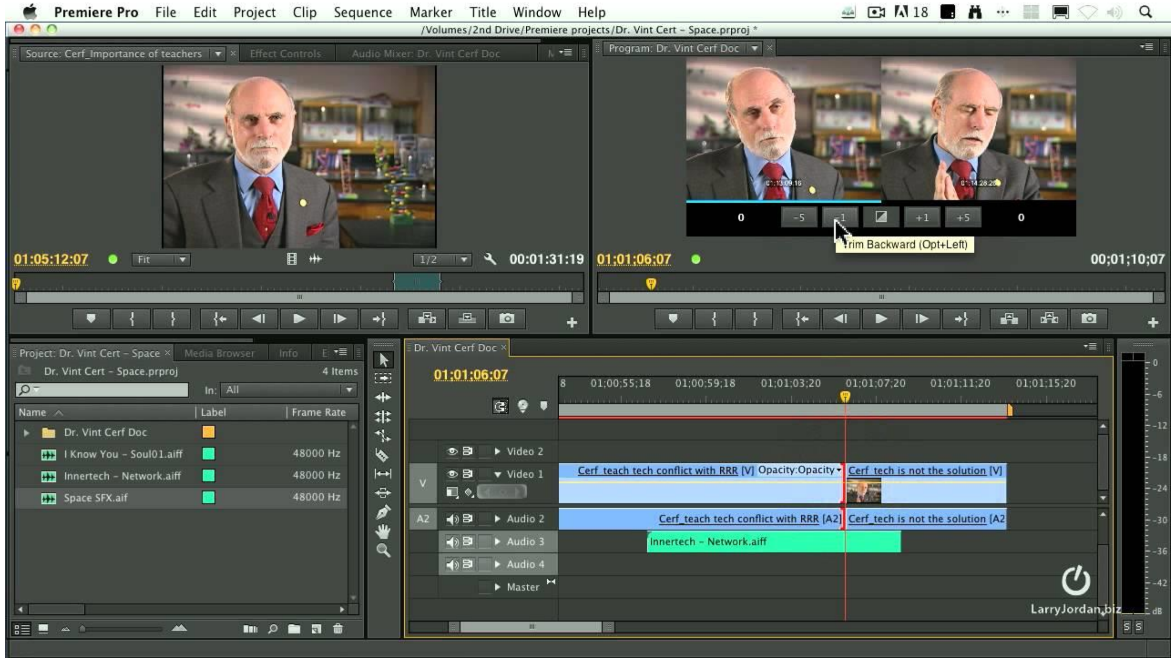
Text-based screencasts enable

rich interaction with the text-based content in screencasts.

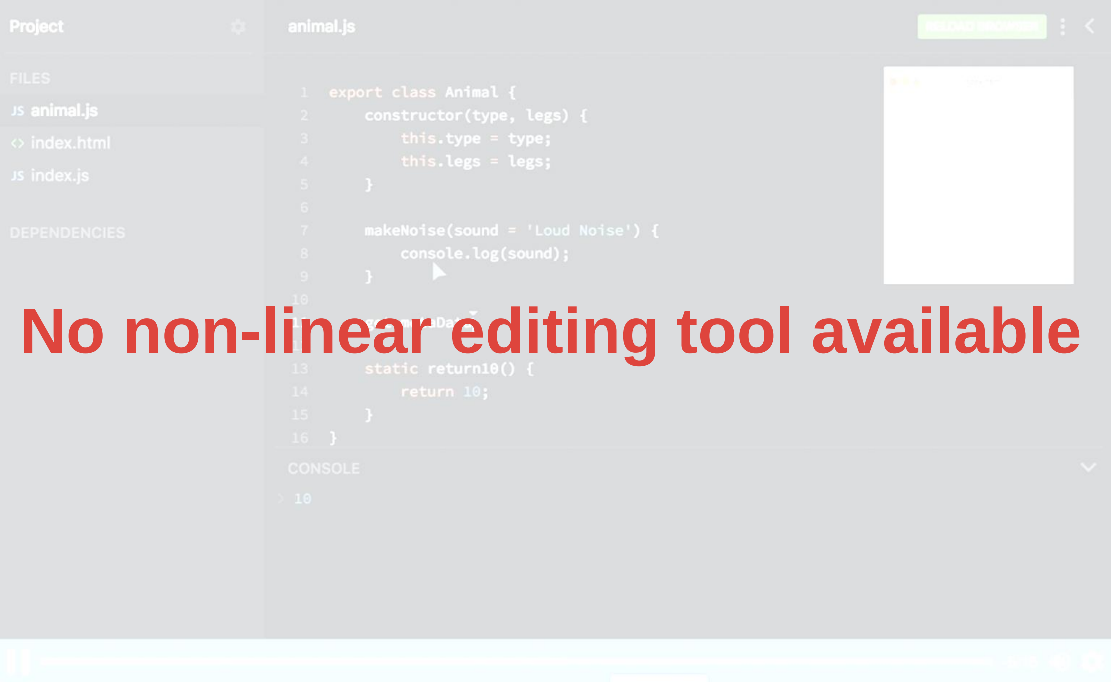


```
export class Animal {
    constructor(type, legs) {
        this.type = type;
        this.legs = legs;
    }

    makeNoise(sound = 'Loud Noise') {
        console.log(sound);

    get metaData() {

    }

    static return10() {
        return 10;
```

```
> 10
```


Premiere Pro File Edit Project Clip Sequence Marker Title Window Help
/Volumes/2nd Drive/Premiere projects/Dr. Vint Cert – Space.prproj *
Source: Cerf_Importance of teachers
Effect Controls
Audio Mixer: Dr. Vint Cerf Doc
Program: Dr. Vint Cerf Doc
01:05:12:07
00:01:31:19
01;01;06;07
00;01;10;07
Trim Backward (Opt+Left)
Project: Dr. Vint Cert – Space
Media Browser
Info
Dr. Vint Cert – Space.prproj
4 Items
Name
Label
Frame Rate
Dr. Vint Cerf Doc
I Know You – Soul01.aiff
48000 Hz
Innertech – Network.aiff
48000 Hz
Space SFX.aif
48000 Hz
Dr. Vint Cerf Doc
01;01;06;07
Video 2
Video 1
Audio 2
Audio 3
Audio 4
Master
Cerf_teach tech conflict with RRR [V] Opacity:Opacity
Cerf_tech is not the solution [V]
Cerf_teach tech conflict with RRR [A2]
Cerf_tech is not the solution [A2
Innertech – Network.aiff
LarryJordan.biz

Videos



```
export class Animal {
    constructor(type, legs) {
        this.type = type;
        this.legs = legs;
    }

    makeNoise(sound = 'Loud Noise') {
        console.log(sound);
    }


        static return10() {
            return 10;
        }
}
```

```
10
```

No non-linear editing tool available

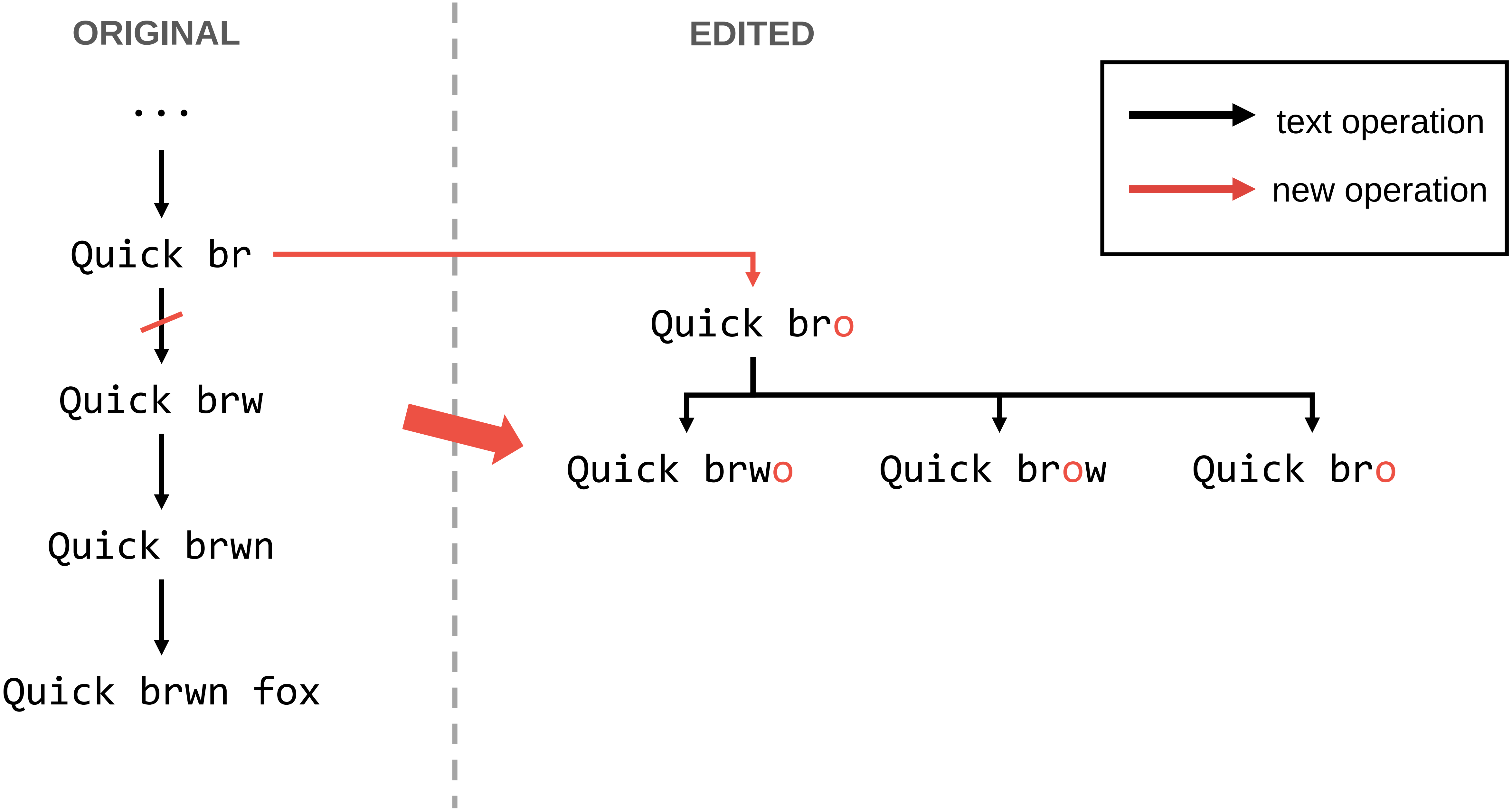
Text-based screencasts

Technical Challenge

Text operation sequence has a **causal** structure. Future events are influenced by the past!

Ambiguity: Changes made in the past often create **multiple possibilities** for diverging future timelines.

Resolution: Need some mechanism to resolve the ambiguity.



ORIGINAL

...

Quick br

Quick brw

Quick brwn

Quick brwn fox

EDITED

Quick bro

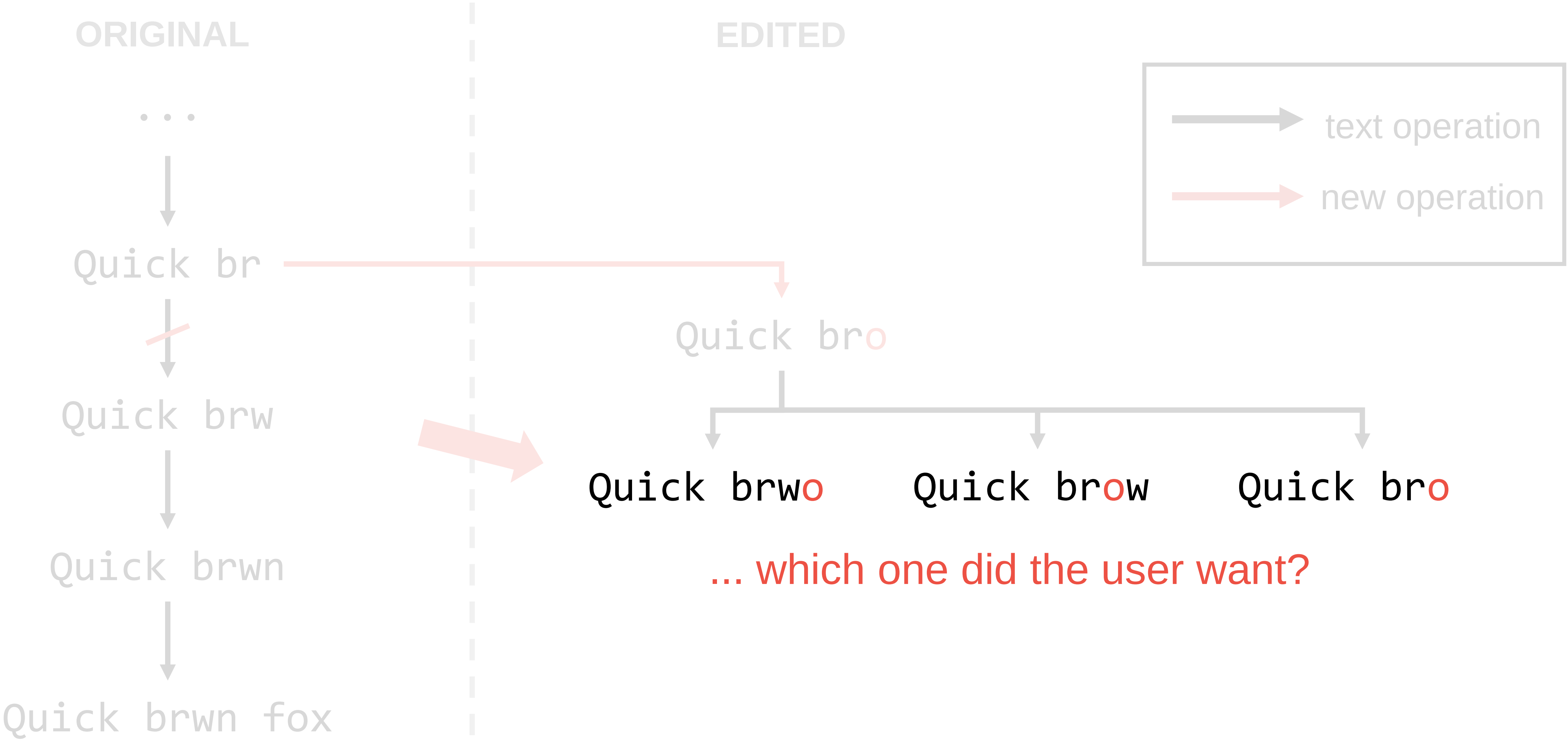
Quick brwo

Quick brow

Quick bro

text operation

new operation



ORIGINAL

EDITED

...

Quick br

Quick brw

Quick brwn

Quick brwn fox

Quick bro

Quick brwo

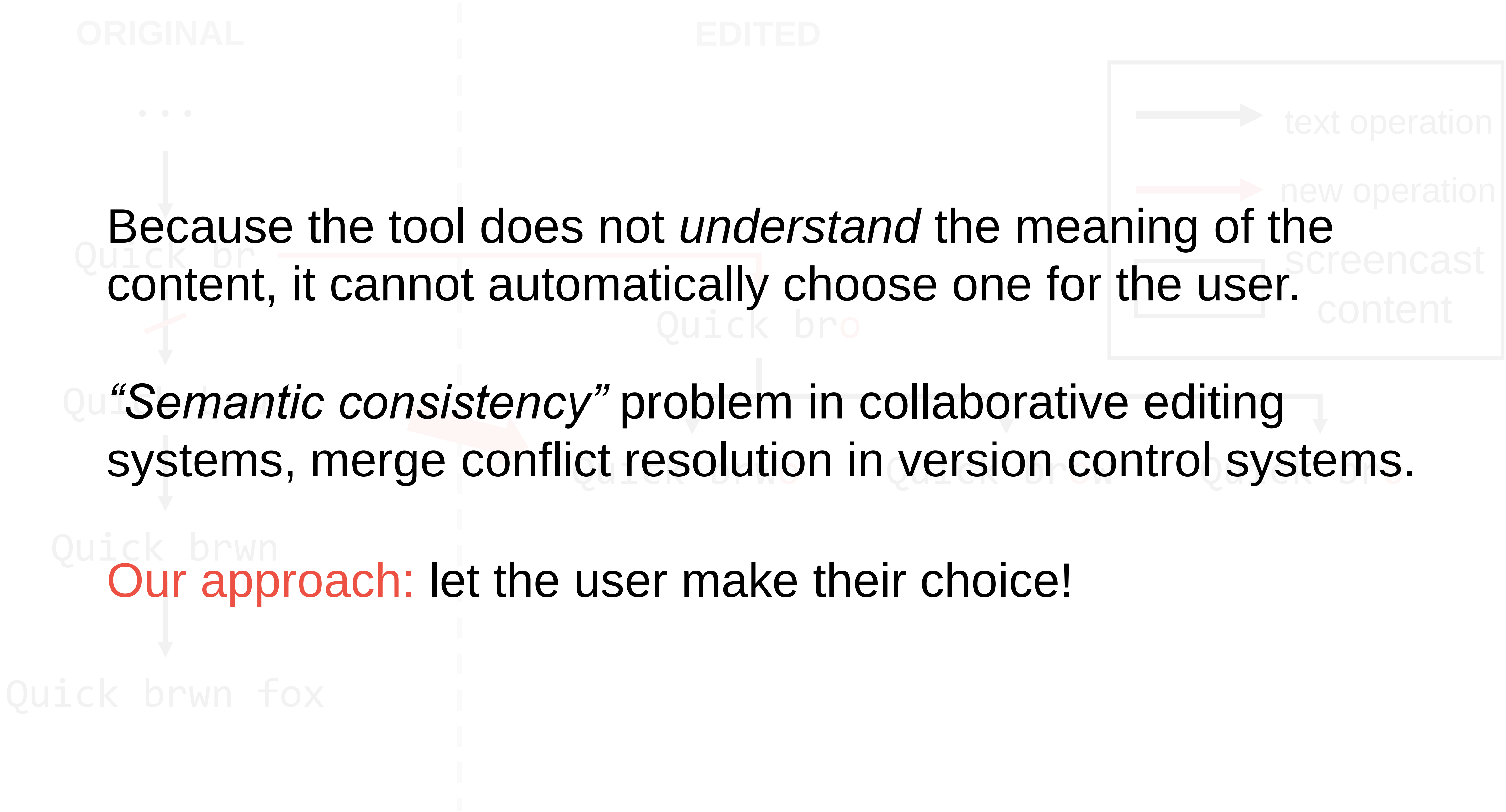
Quick brow

Quick bro

text operation

new operation

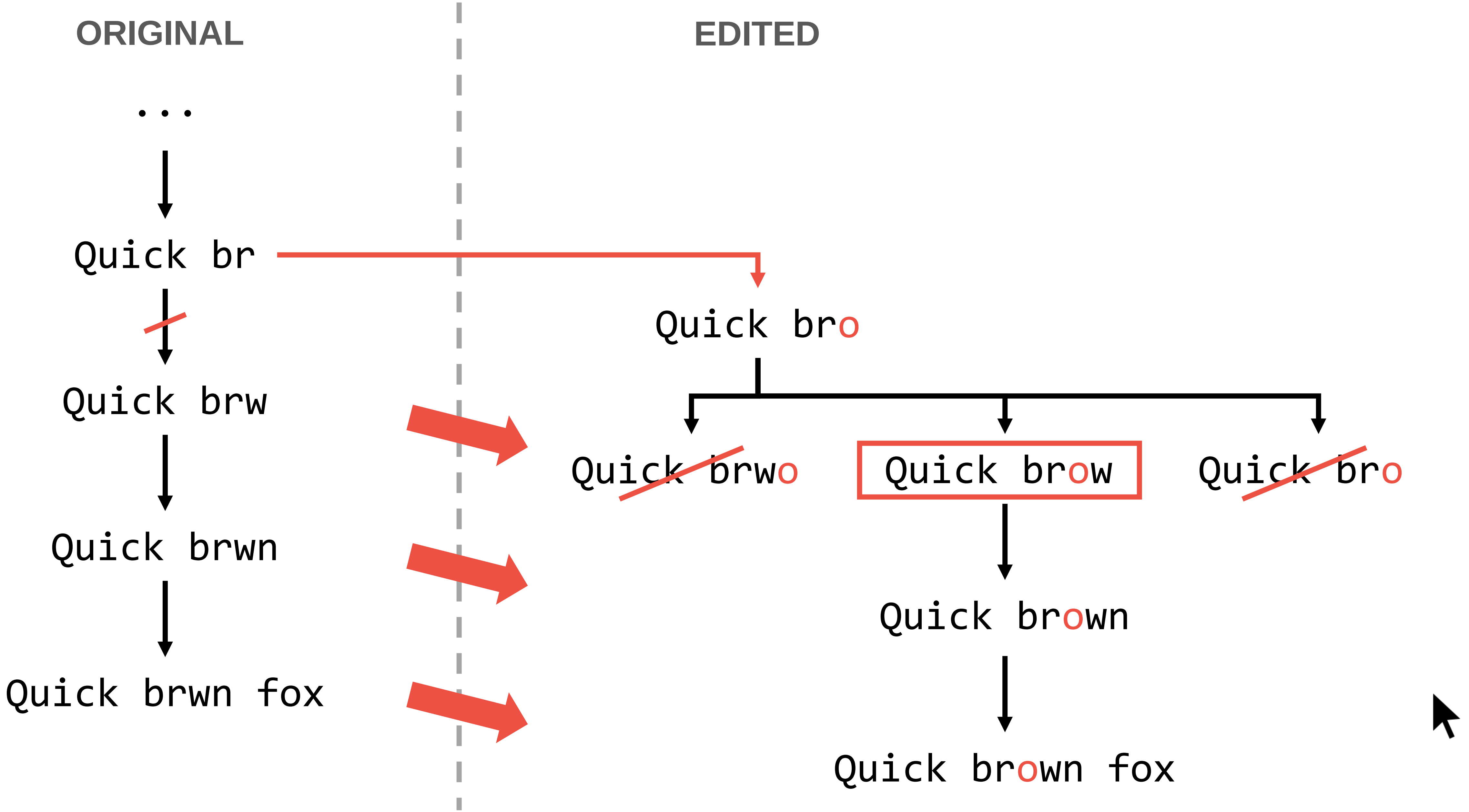
... which one did the user want?



Because the tool does not *understand* the meaning of the content, it cannot automatically choose one for the user.

“Semantic consistency” problem in collaborative editing systems, merge conflict resolution in version control systems.

Our approach: let the user make their choice!



ORIGINAL

EDITED

...

Quick br

Quick brw

Quick brwn

Quick brwn fox

Quick bro

~~Quick brwo~~

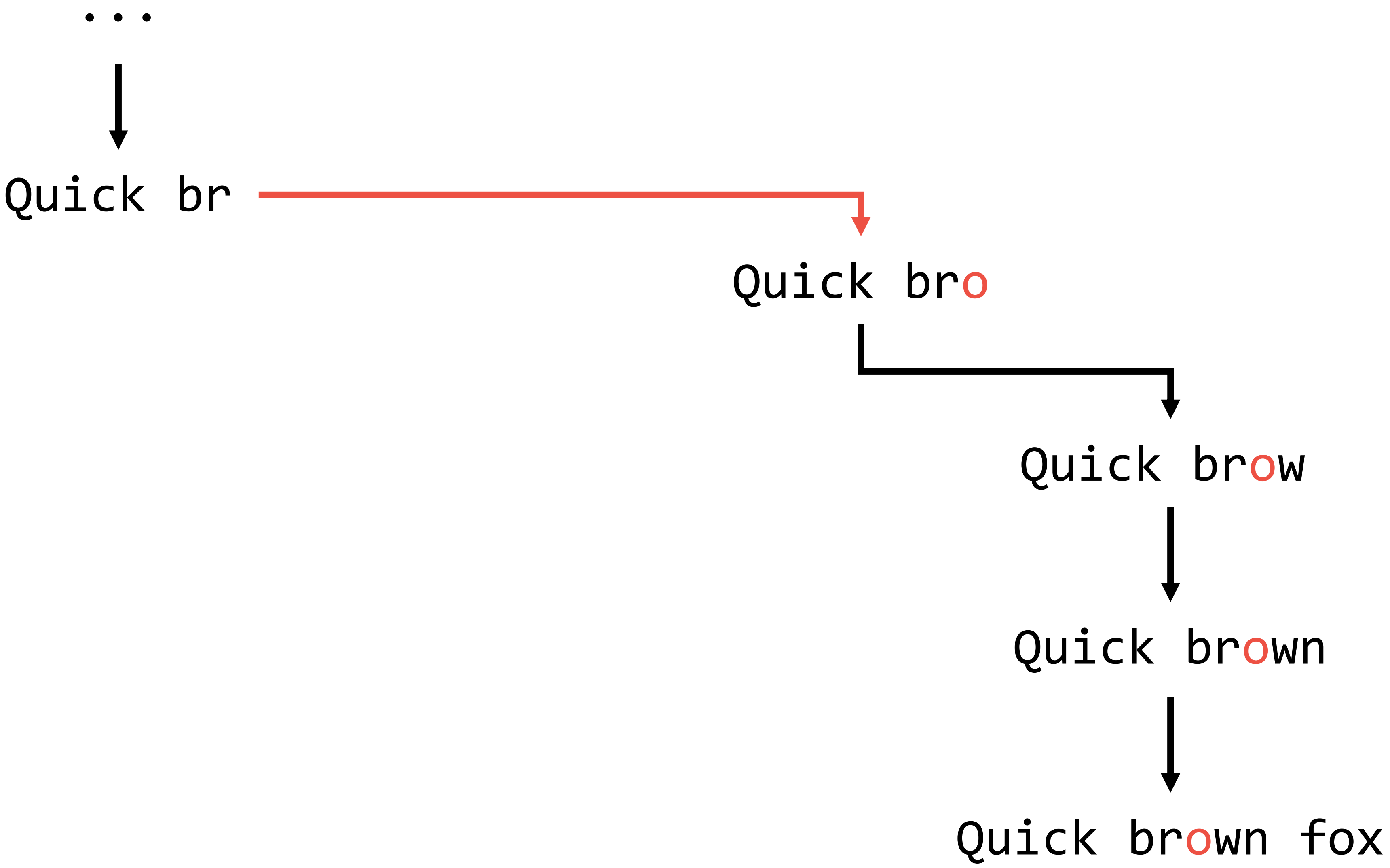
Quick brow

~~Quick bro~~

Quick brown

Quick brown fox

EDITED



...

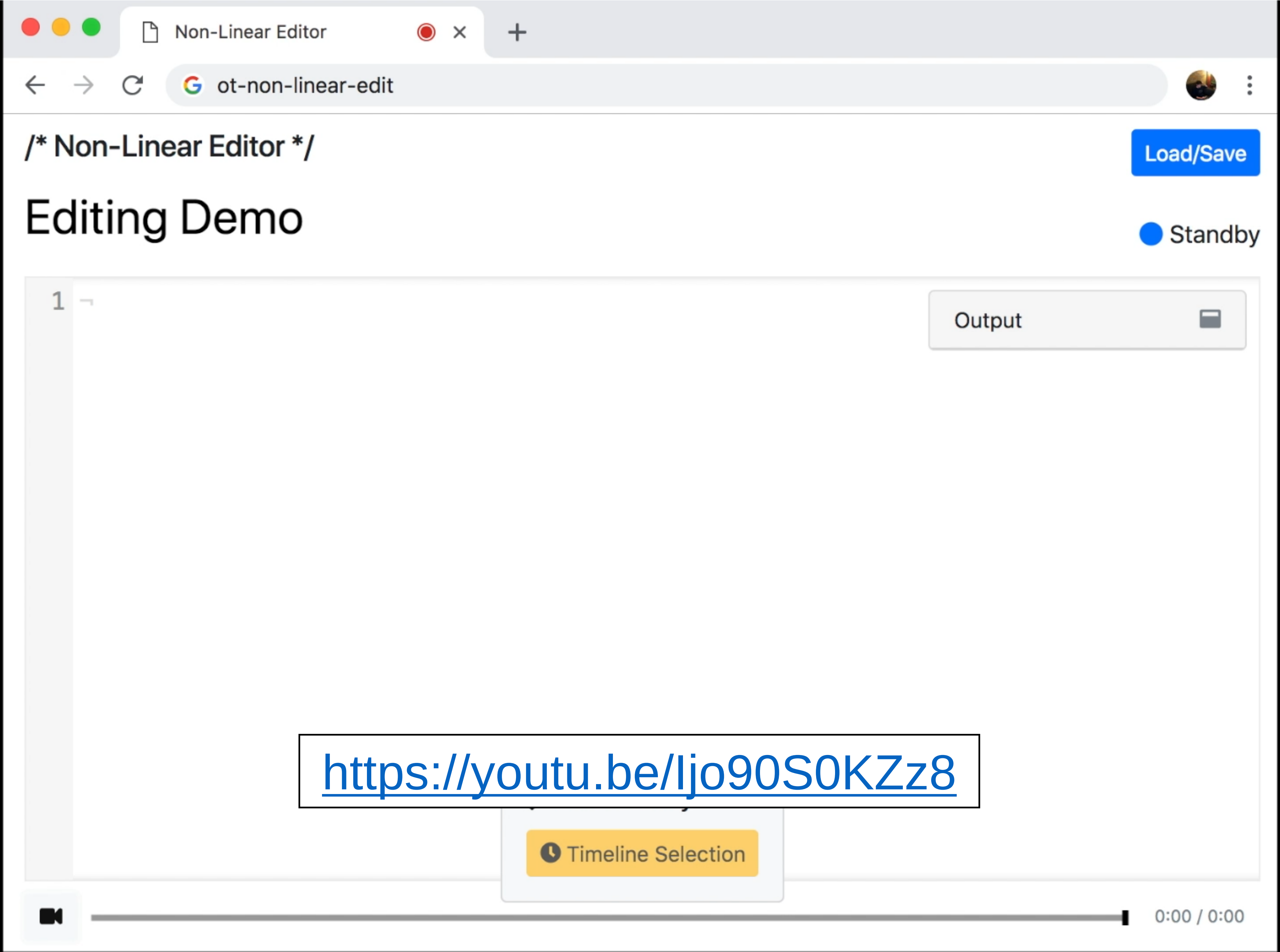
Quick br

Quick bro

Quick brow

Quick brown

Quick brown fox



https://youtu.be/Ijo90S0KZz8

Outline

Non-linear editing algorithm for text-based screencasts

A prototype editor that implements non-linear editing functionality for text-based screencasts

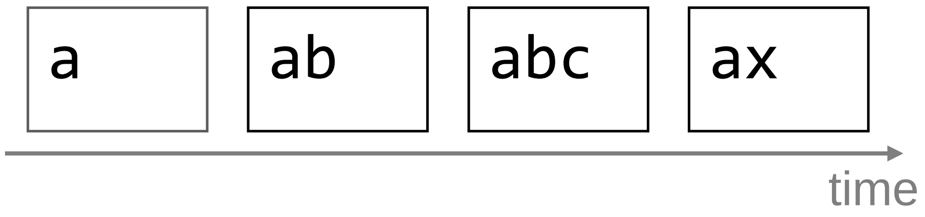
An exploratory study demonstrating that users can successfully edit a text-based screencast using our editor in various scenarios

Non-Linear Editing Algorithm
for text-based screencasts

DEFINITION

- A screencast is a sequence of text operations.
  (we use OT as a unit of text operations)

$$\text{Screencast } L = \left(\begin{array}{l} \delta(0,\ 0,\ \text{“a”}), \\ \delta(1,\ 1,\ \text{“b”}), \\ \delta(2,\ 2,\ \text{“c”}), \\ \delta(1,\ 3,\ \text{“x”}) \end{array} \right) =$$



Non-Linear Editing Algorithm
for text-based screencasts

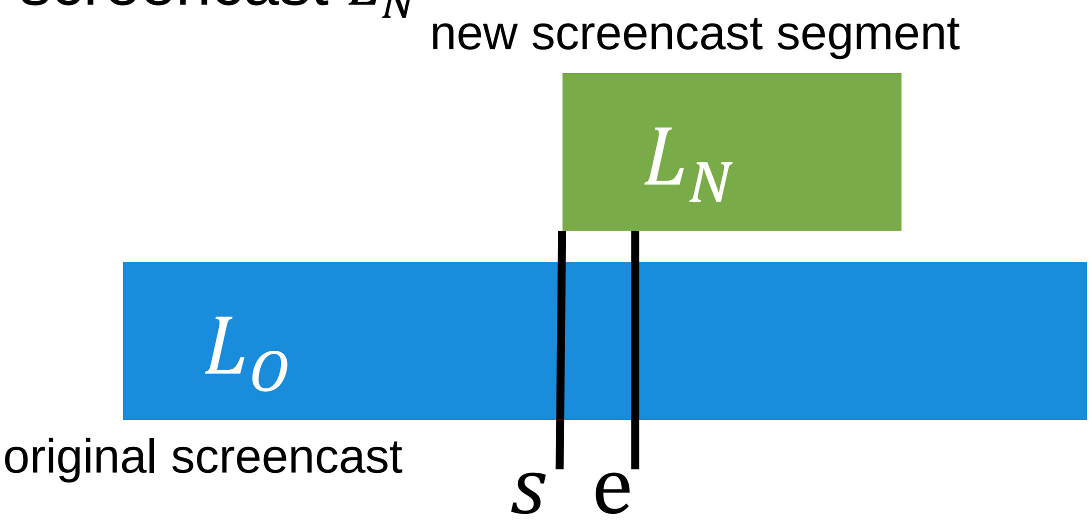
INPUT

- a screencast L_O
- a range $[s, \mathrm{e}]$ to re-record
- a re-recorded screencast L_N

OUTPUT

- a screencast L

$$L \coloneqq L_O[:s] + L_N + resolve(L_O[e:])$$



Non-Linear Editing Algorithm
for text-based screencasts

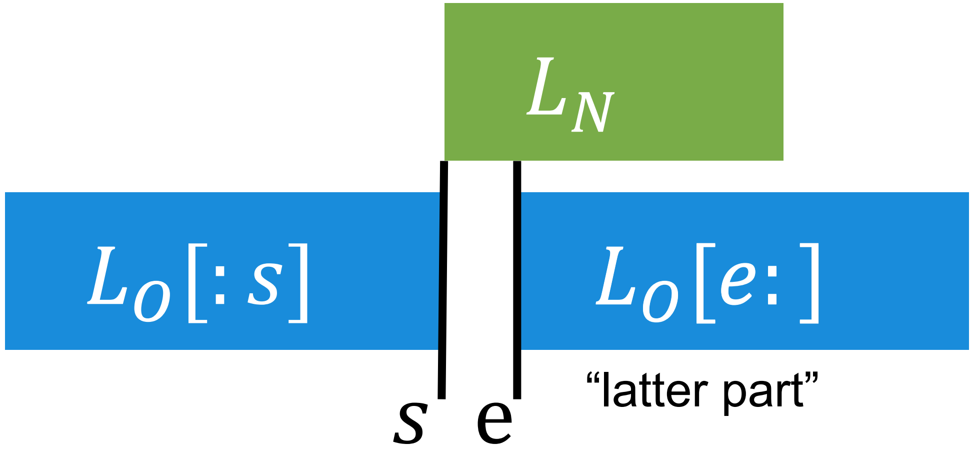
INPUT

- a screencast L_O
- a range $[s, \mathrm{e}]$ to re-record
- a re-recorded screencast L_N

OUTPUT

- a screencast L

$$L \coloneqq L_O[:s] + L_N + resolve(L_O[e:])$$



Non-Linear Editing Algorithm
for text-based screencasts

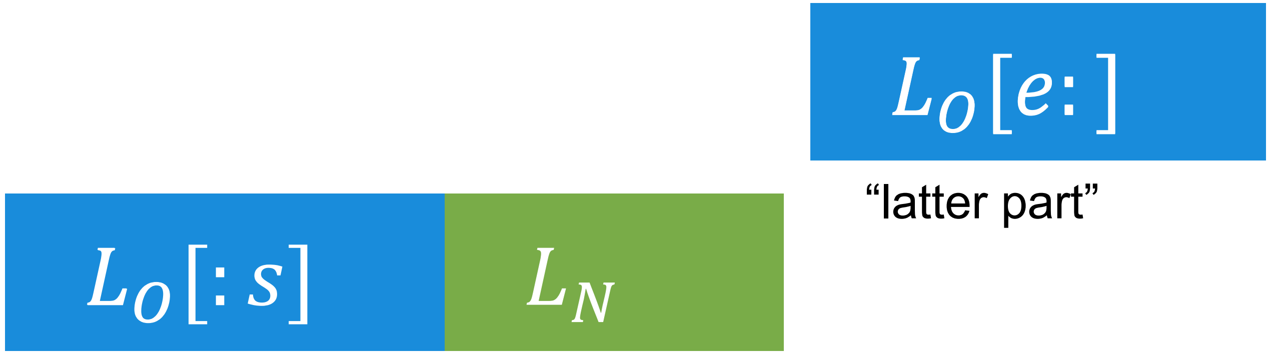
INPUT

- a screencast L_O
- a range $[s, \mathrm{e}]$ to re-record
- a re-recorded screencast L_N

OUTPUT

- a screencast L

$$L \coloneqq L_O[:s] + L_N + resolve(L_O[e:])$$



Non-Linear Editing Algorithm
for text-based screencasts

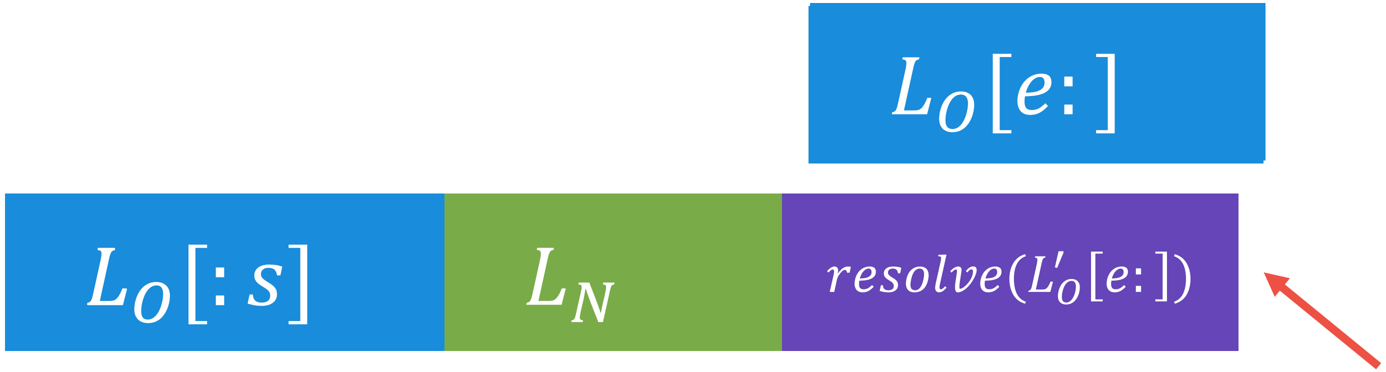
INPUT

- a screencast L_O
- a range $[s, \mathrm{e}]$ to re-record
- a re-recorded screencast L_N

OUTPUT

- a screencast L

$L \coloneqq L_O[:s] + L_N + resolve(L_O[e:])$

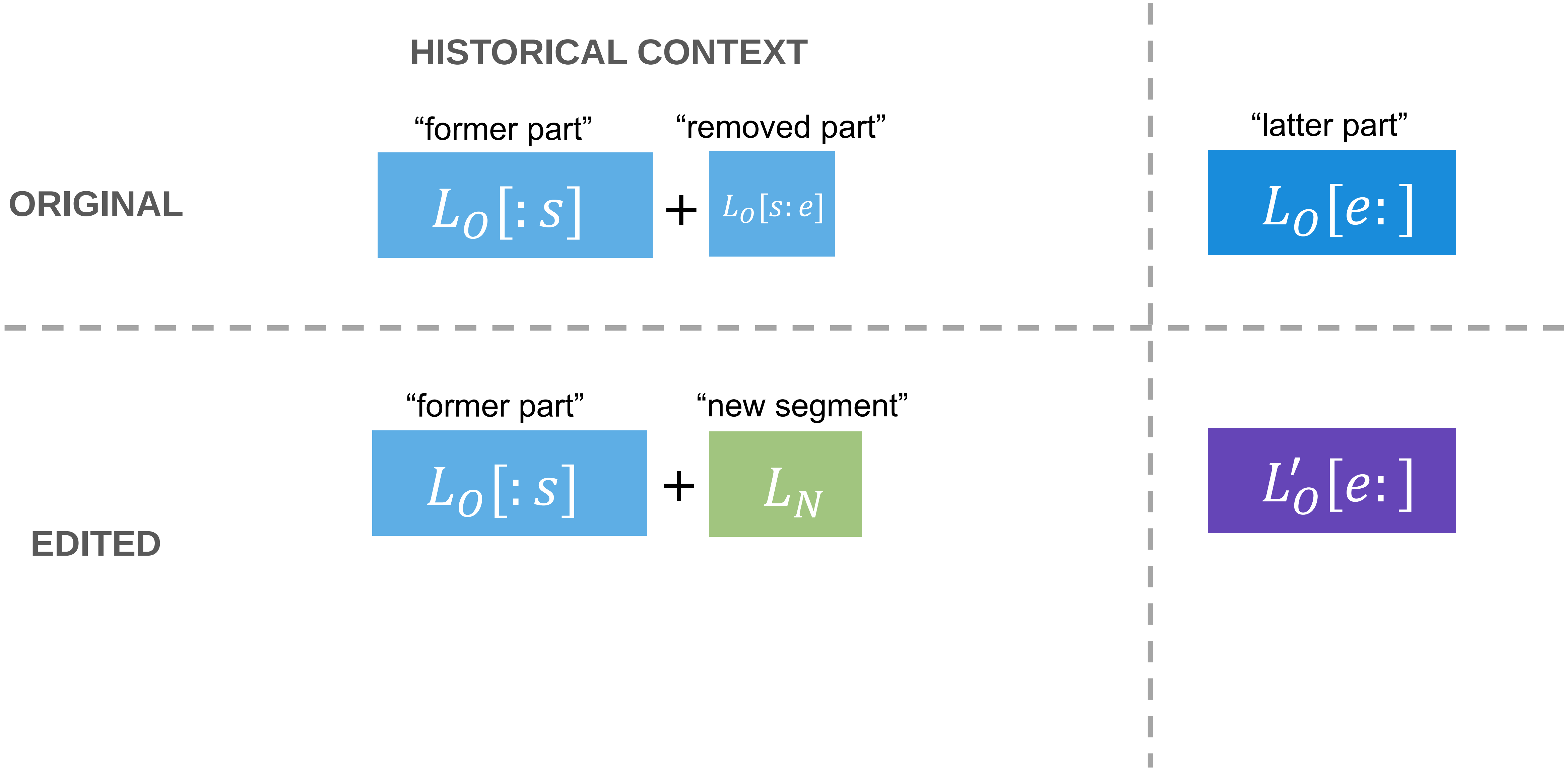


$L_O[e:]$

$L_O[:s]$ L_N $resolve(L'_O[e:])$

Transform the OTs.
In case of ambiguity, ask user.

$L_O[:e]$ Transformation (w/o Ambiguity Resolution)



HISTORICAL CONTEXT

ORIGINAL

“former part” $L_O[:s]$ + “removed part” $L_O[s:e]$

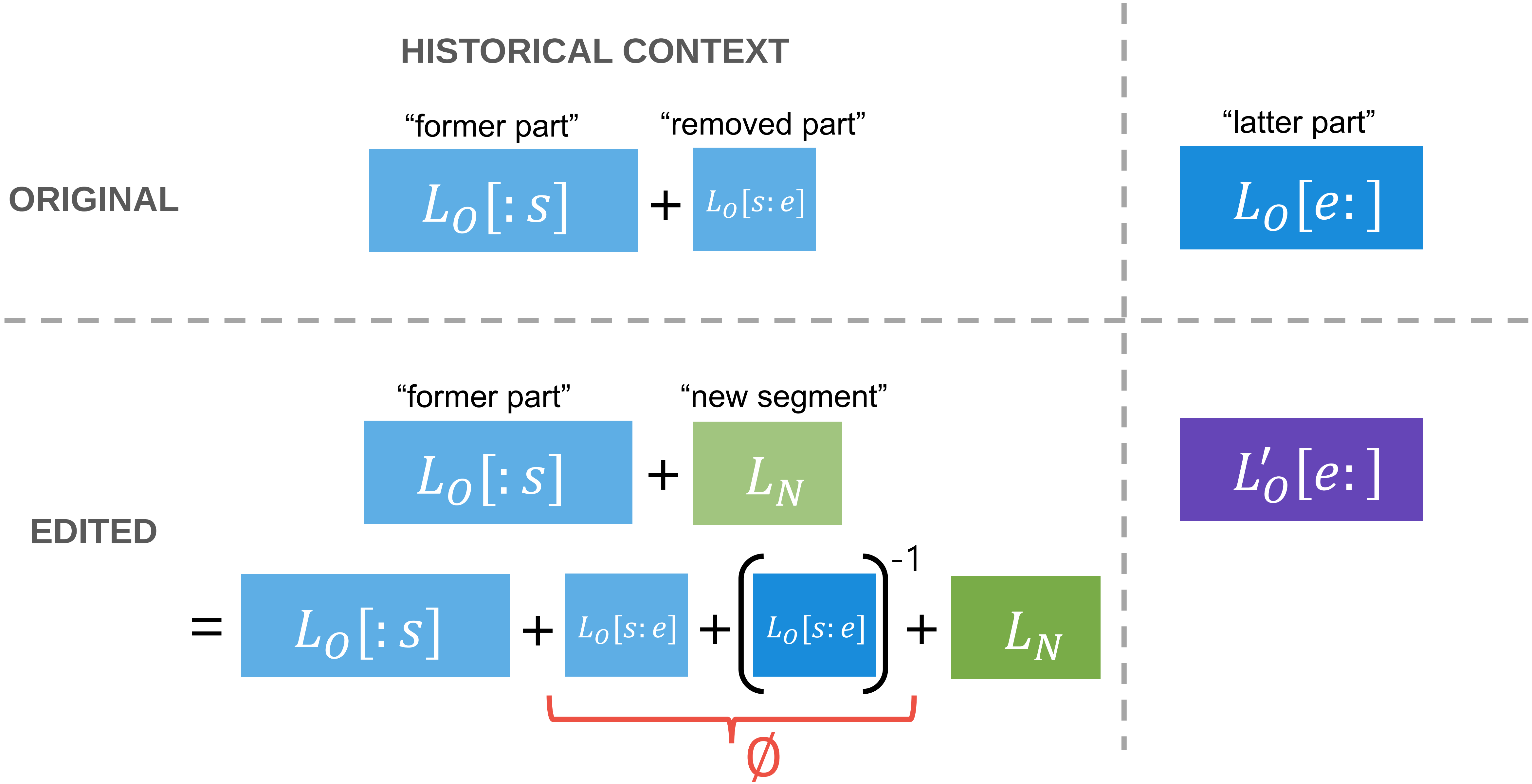
“latter part” $L_O[e:]$

EDITED

“former part” $L_O[:s]$ + “new segment” L_N

$L'_O[e:]$

$L_O[:e]$ Transformation (w/o Ambiguity Resolution)



HISTORICAL CONTEXT

ORIGINAL

"former part" $L_O[:s]$ + "removed part" $L_O[s:e]$

"latter part" $L_O[e:]$

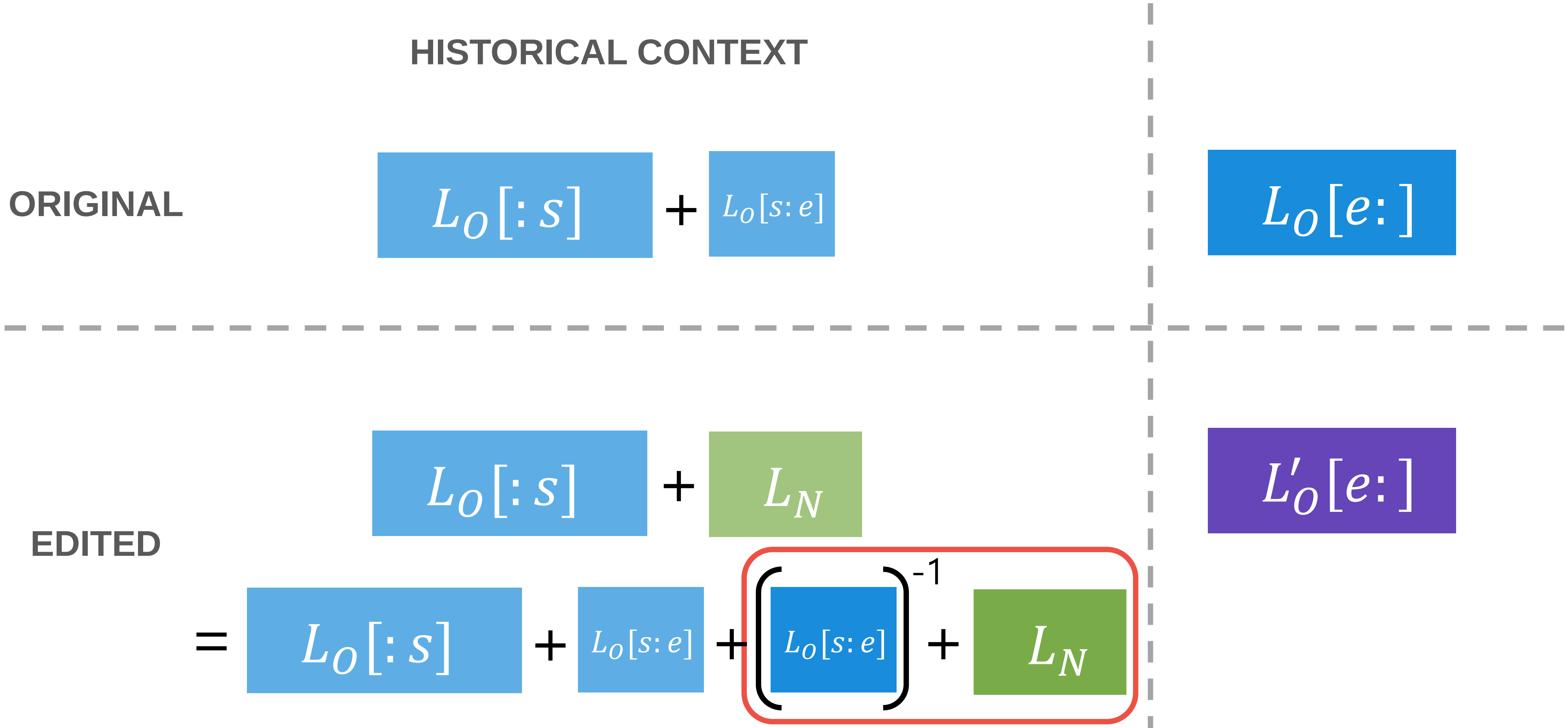
EDITED

"former part" $L_O[:s]$ + "new segment" L_N

$L'_O[e:]$

$$= L_O[:s] + \underbrace{L_O[s:e] + \left(L_O[s:e]\right)^{-1}}_{\emptyset} + L_N$$

$L_O[:e]$ Transformation (w/o Ambiguity Resolution)



HISTORICAL CONTEXT

ORIGINAL

$$L_O[:s] + L_O[s:e]$$

$$L_O[e:]$$

EDITED

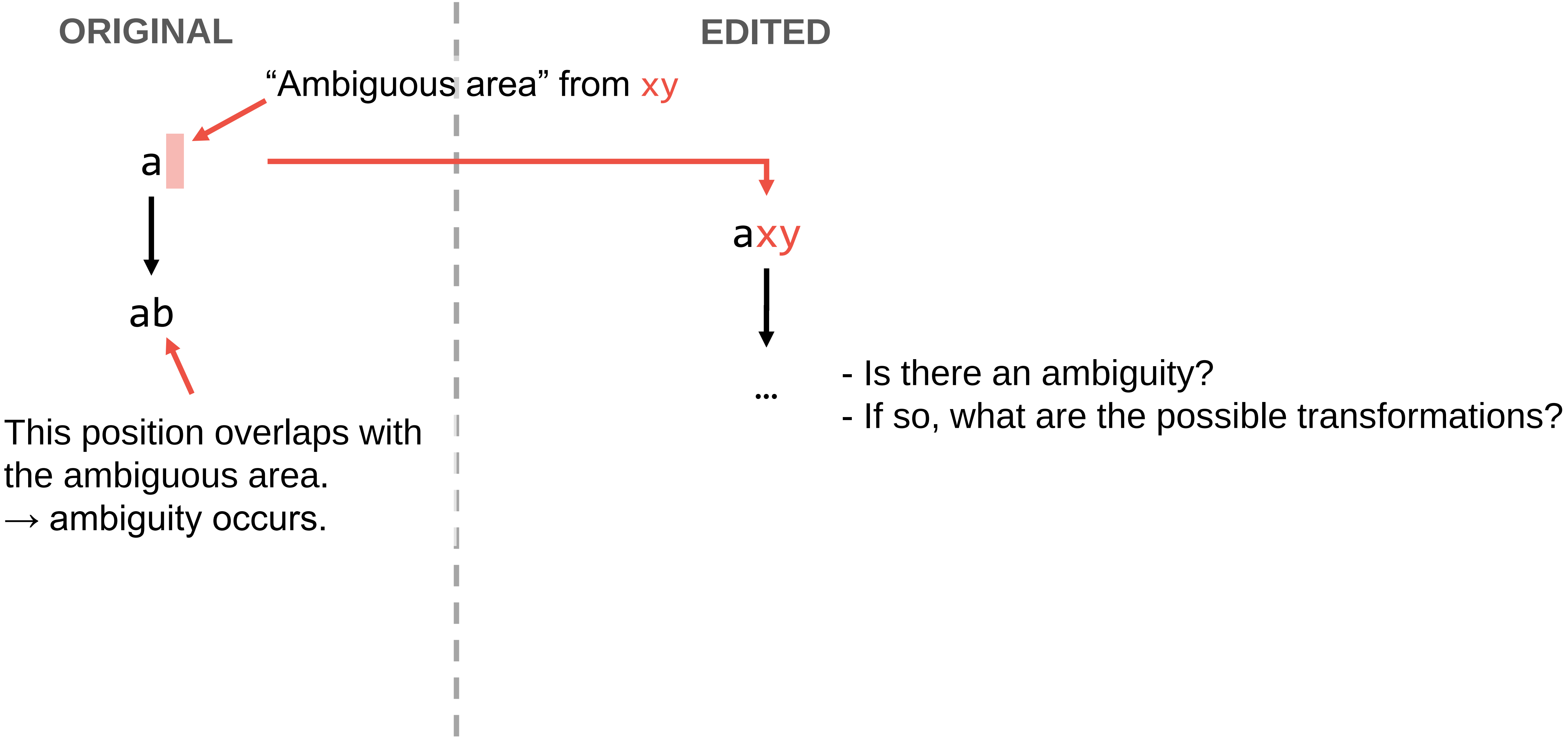
$$L_O[:s] + L_N$$

$$= L_O[:s] + L_O[s:e] + \left(L_O[s:e]\right)^{-1} + L_N$$

$$L'_O[e:]$$

Transform $L_O[e:]$ with …

$L_O[:e]$ Transformation (w/ Ambiguity Resolution)



ORIGINAL

EDITED

“Ambiguous area” from xy

a

ab

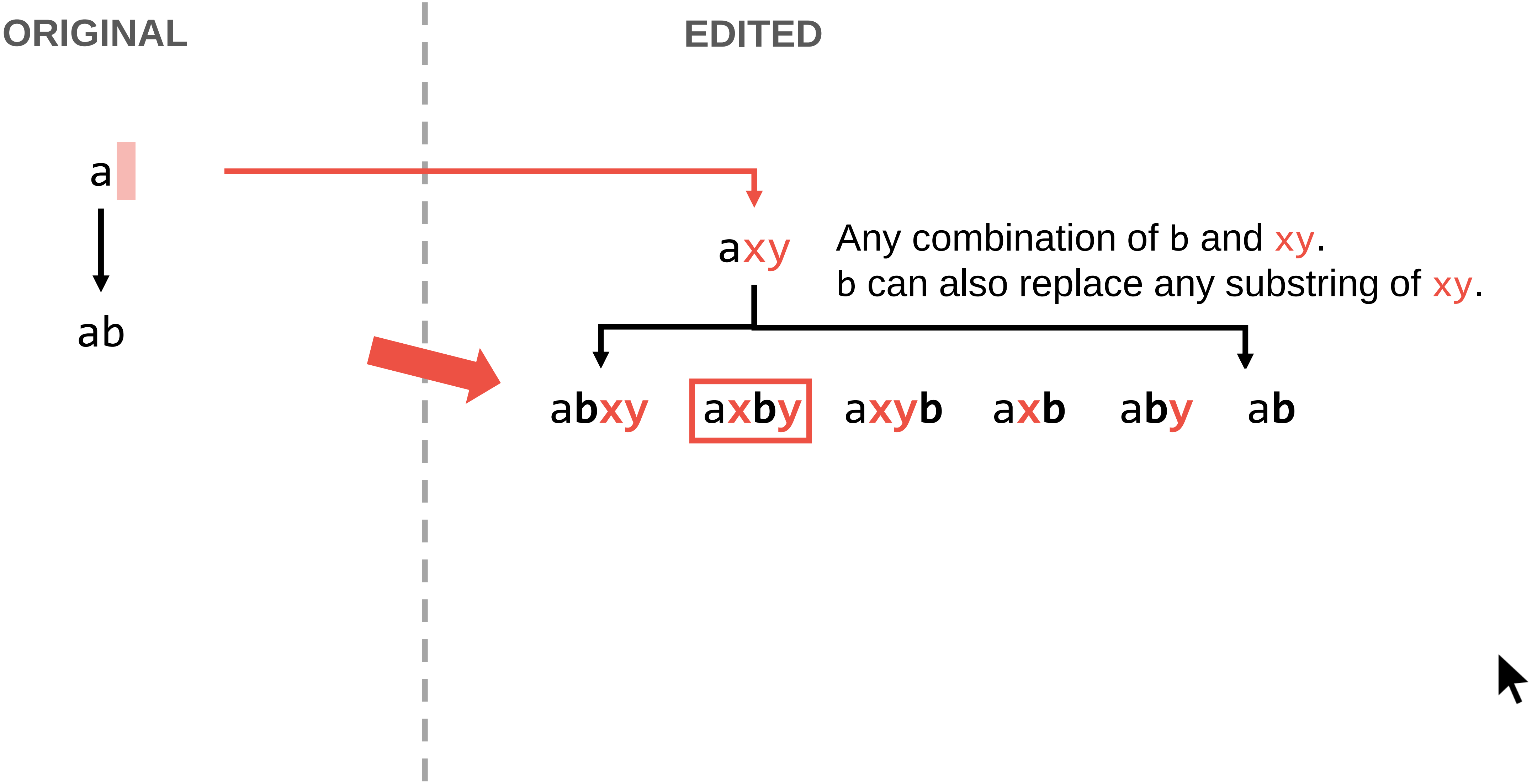
axy

...

This position overlaps with the ambiguous area.
→ ambiguity occurs.

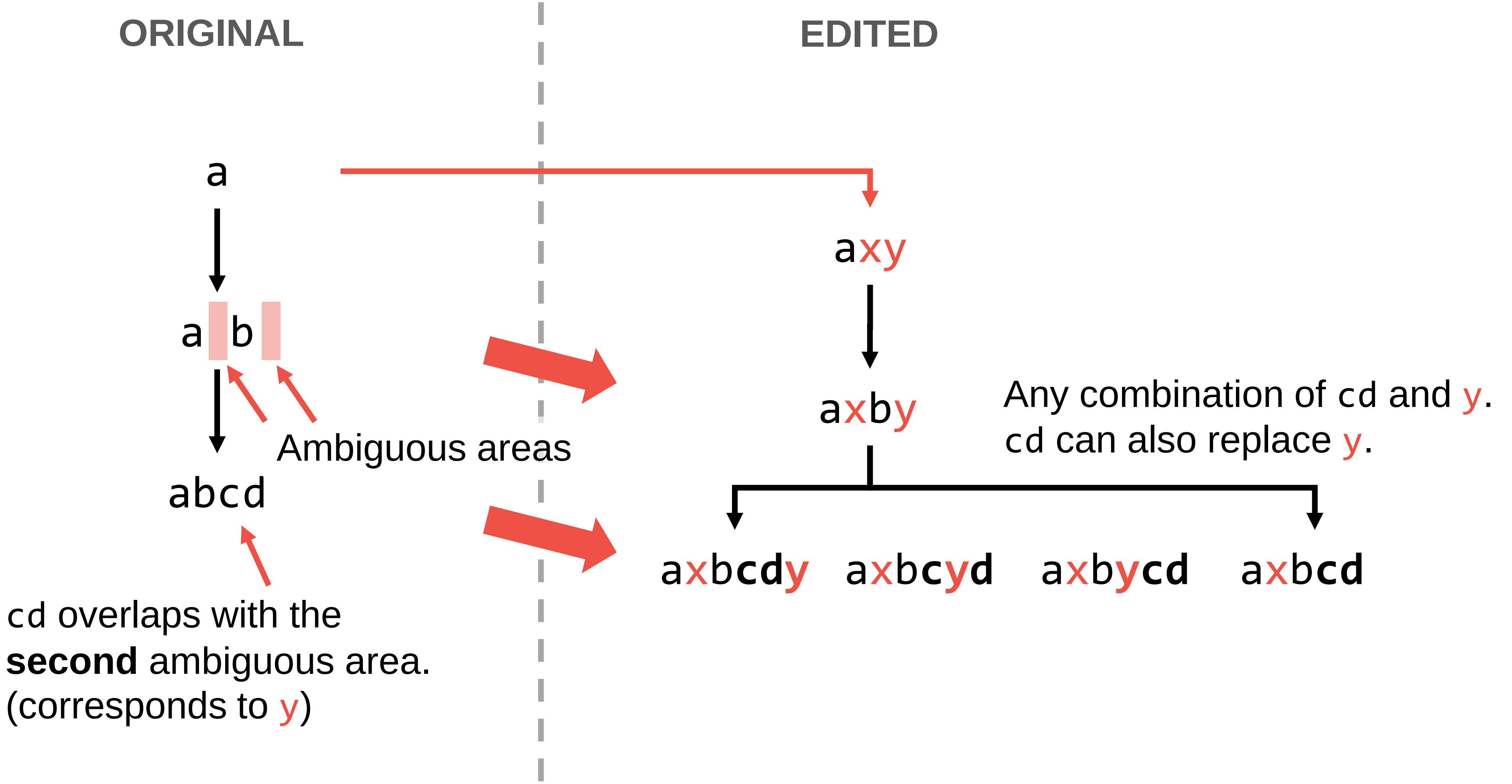
- Is there an ambiguity?
- If so, what are the possible transformations?

$L_O[:e]$ Transformation (w/ Ambiguity Resolution)



Any combination of b and xy.
b can also replace any substring of xy.

$L_O[:e]$ Transformation (w/ Ambiguity Resolution)



ORIGINAL

EDITED

a

a b

abcd

Ambiguous areas

cd overlaps with the **second** ambiguous area. (corresponds to y)

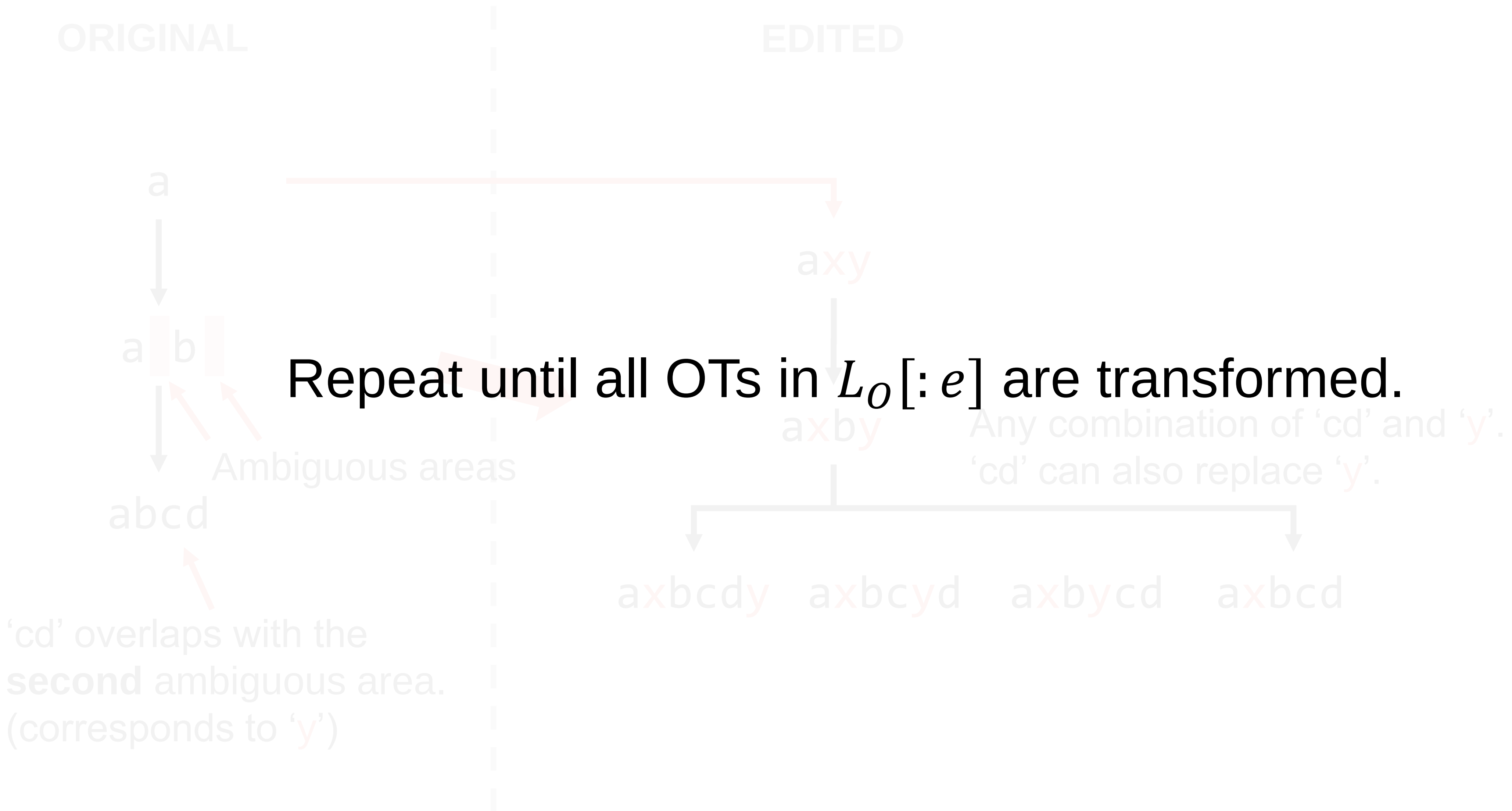
axy

axby

Any combination of cd and y. cd can also replace y.

axb**cd**y axb**c**y**d** axby**cd** axb**cd**

$L_O[:e]$ Transformation (w/ Ambiguity Resolution)



ORIGINAL

EDITED

a

axy

a b

axby

Ambiguous areas

Any combination of ‘cd’ and ‘y’.
‘cd’ can also replace ‘y’.

abcd

axbcdy axbcyd axbycd axbcd

‘cd’ overlaps with the **second** ambiguous area. (corresponds to ‘y’)

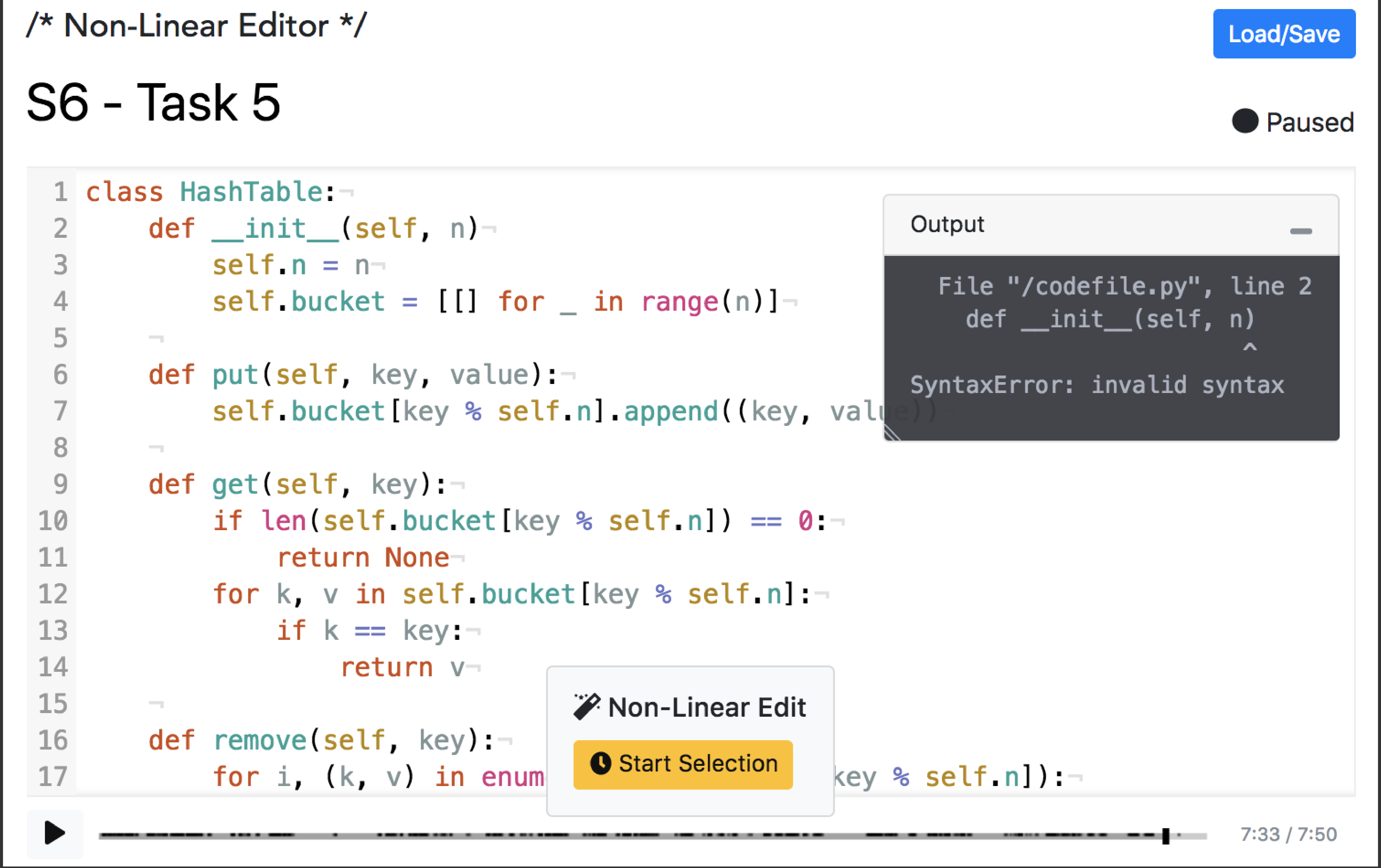
Repeat until all OTs in $L_O[:e]$ are transformed.

Outline

Non-linear editing algorithm for text-based screencasts

A prototype editor that implements non-linear editing functionality for text-based screencasts

An exploratory study demonstrating that users can successfully edit a text-based screencast using our editor in various scenarios

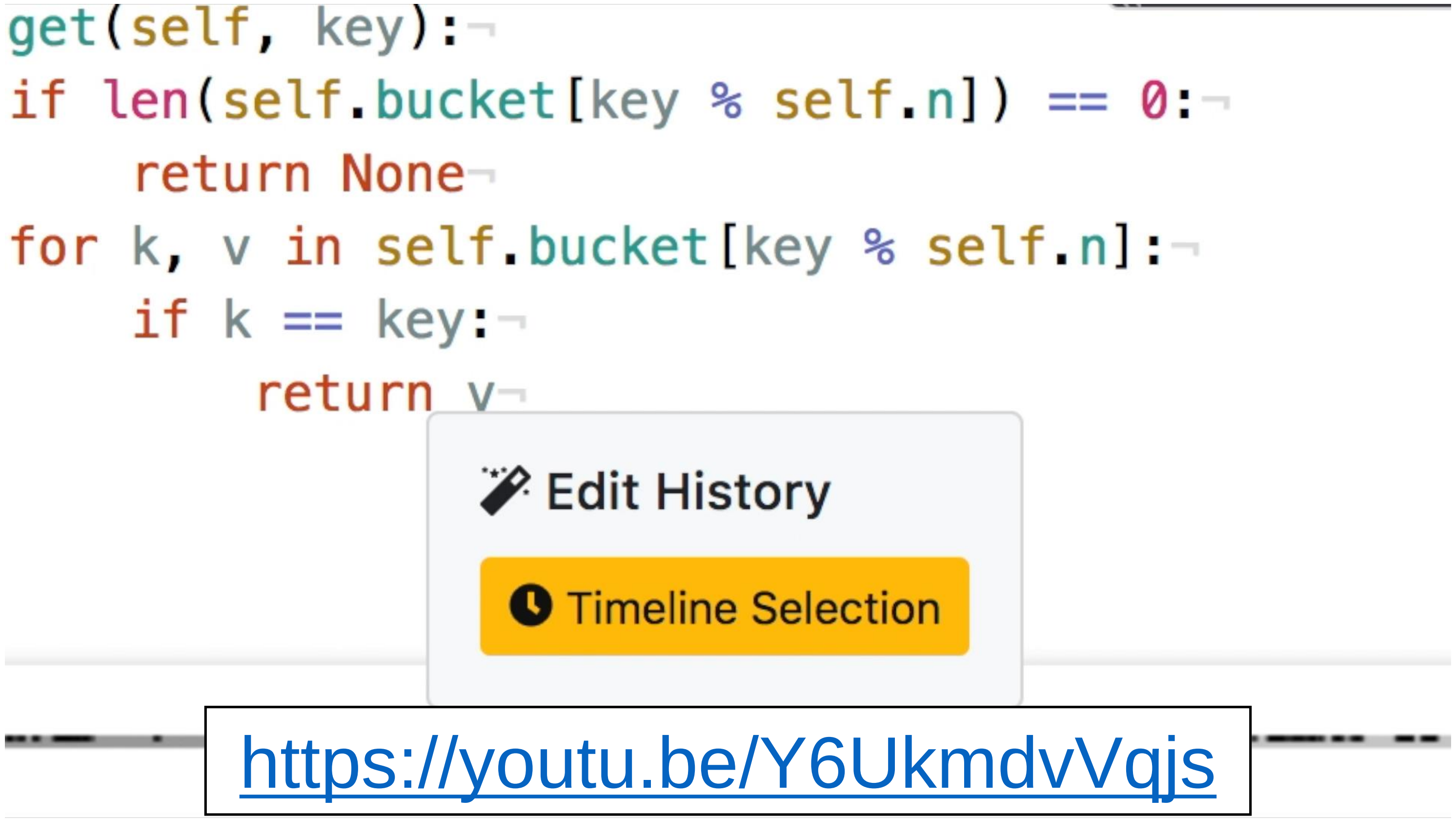


S6 - Task 5

```
class HashTable:
    def __init__(self, n)
        self.n = n
        self.bucket = [[] for _ in range(n)]

    def put(self, key, value):
        self.bucket[key % self.n].append((key, value))

    def get(self, key):
        if len(self.bucket[key % self.n]) == 0:
            return None
        for k, v in self.bucket[key % self.n]:
            if k == key:
                return v

    def remove(self, key):
        for i, (k, v) in enum                    key % self.n]):
```

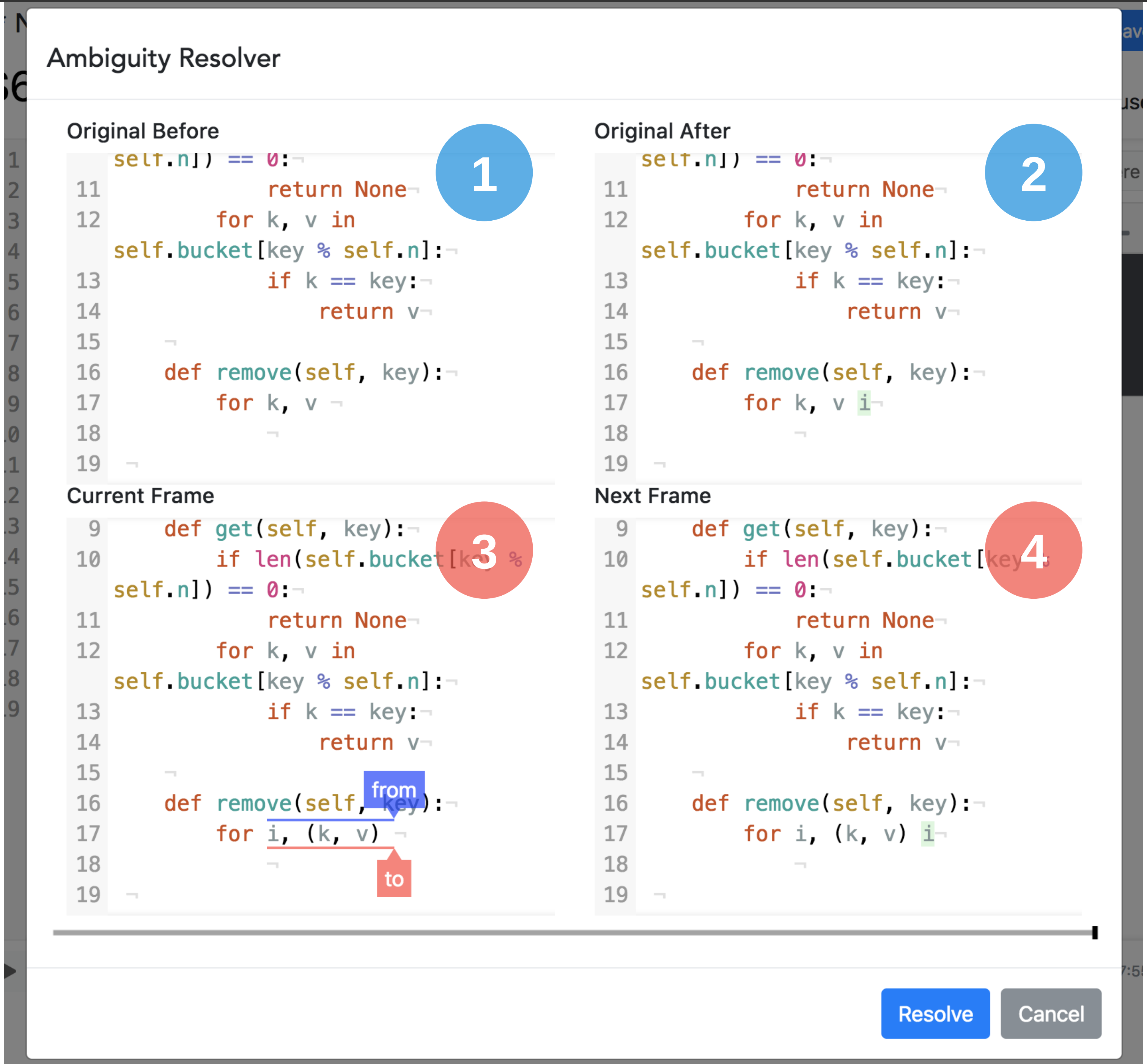
```
  File "/codefile.py", line 2
    def __init__(self, n)
                        ^
SyntaxError: invalid syntax
```

Time Range Selection



```
get(self, key):
if len(self.bucket[key % self.n]) == 0:
    return None
for k, v in self.bucket[key % self.n]:
    if k == key:
        return v
```

https://youtu.be/Y6UkmdvVqjs



Ambiguity Resolver

Original Before

```
self.n]) == 0:
            return None
        for k, v in
self.bucket[key % self.n]:
            if k == key:
                return v

    def remove(self, key):
        for k, v
```

Original After

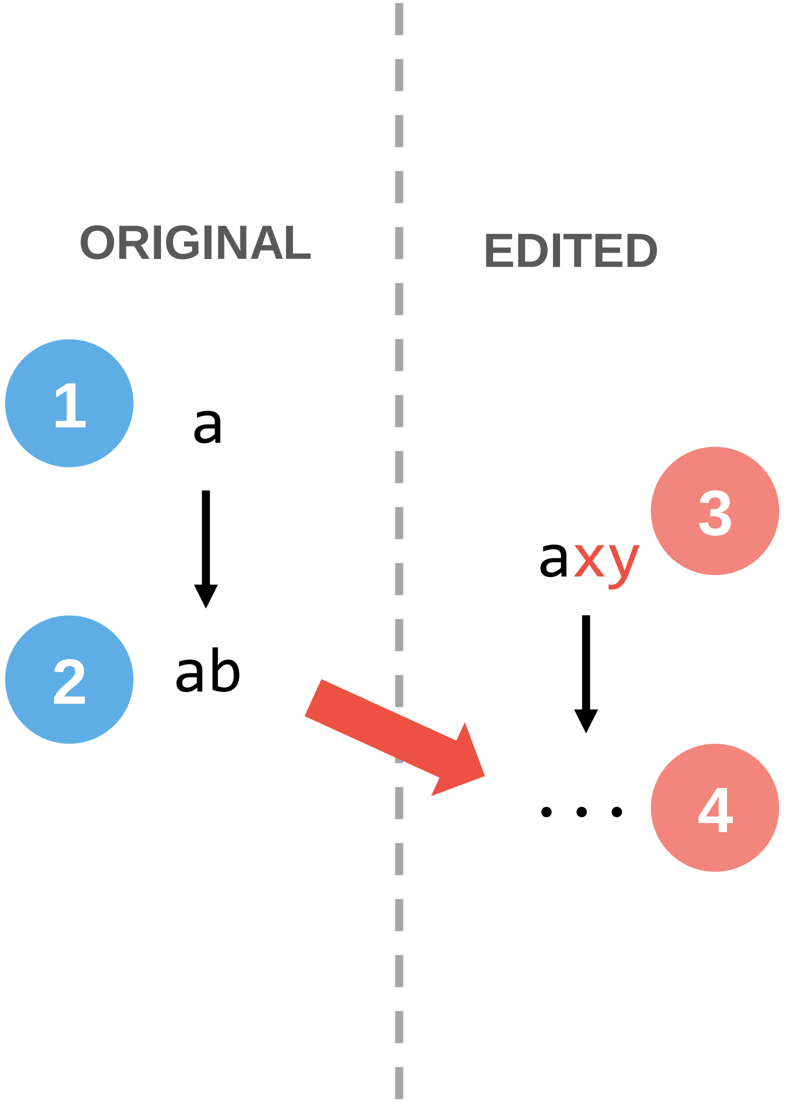
```
self.n]) == 0:
            return None
        for k, v in
self.bucket[key % self.n]:
            if k == key:
                return v

    def remove(self, key):
        for k, v i
```

Current Frame

```
    def get(self, key):
        if len(self.bucket[key %
self.n]) == 0:
            return None
        for k, v in
self.bucket[key % self.n]:
            if k == key:
                return v

    def remove(self, key):
        for i, (k, v)
```

Next Frame

```
    def get(self, key):
        if len(self.bucket[key %
self.n]) == 0:
            return None
        for k, v in
self.bucket[key % self.n]:
            if k == key:
                return v

    def remove(self, key):
        for i, (k, v) i
```



Outline

- Non-linear editing algorithm for text-based screencasts

- A prototype editor that implements non-linear editing functionality for text-based screencasts

- An exploratory study demonstrating that users can successfully edit a text-based screencast using our editor in various scenarios

Exploratory User Study
with simulated real-world use cases

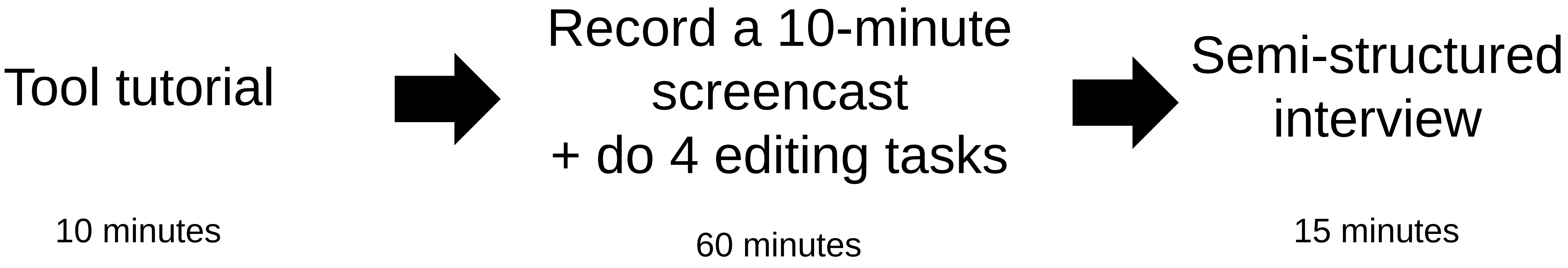
(i) Can users successfully edit a screencast in a diverse range of use cases?

(ii) What editing patterns emerge when users carry out different editing tasks?

(iii) How difficult is it for the users to perform ambiguity resolution?

Exploratory User Study
with simulated real-world use cases

6 participants



Tool tutorial

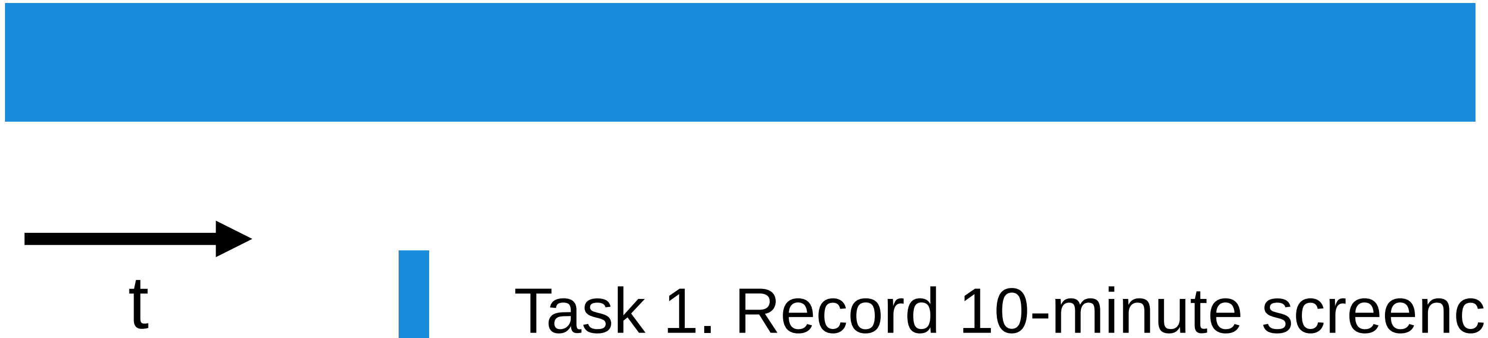
10 minutes

Record a 10-minute screencast
+ do 4 editing tasks

60 minutes

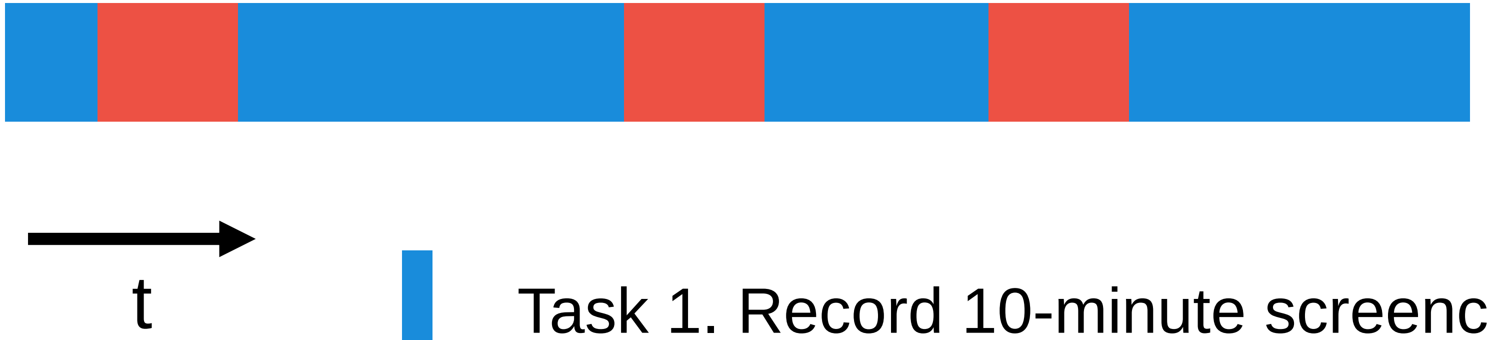
Semi-structured interview

15 minutes



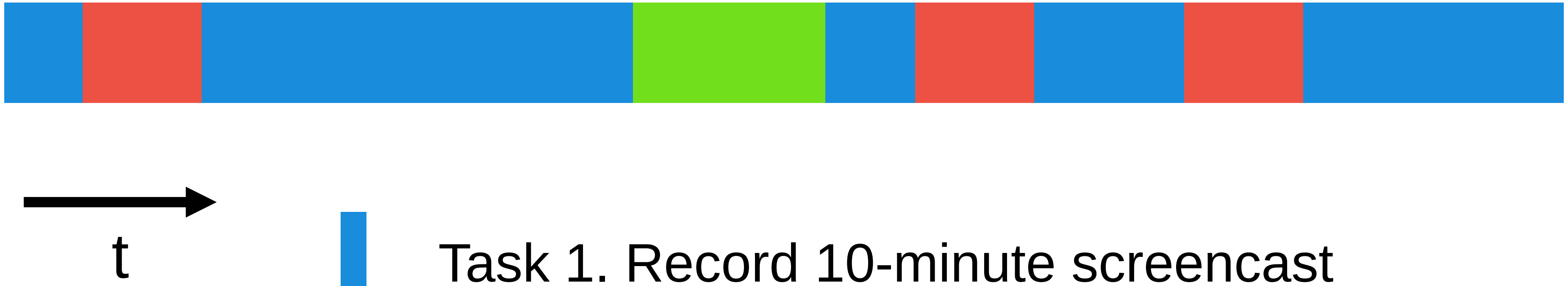
Task 1. Record 10-minute screencast

(write a simple hash table with put/get methods then test cases.)



🟦 Task 1. Record 10-minute screencast
(write a simple hash table with put/get methods then test cases.)

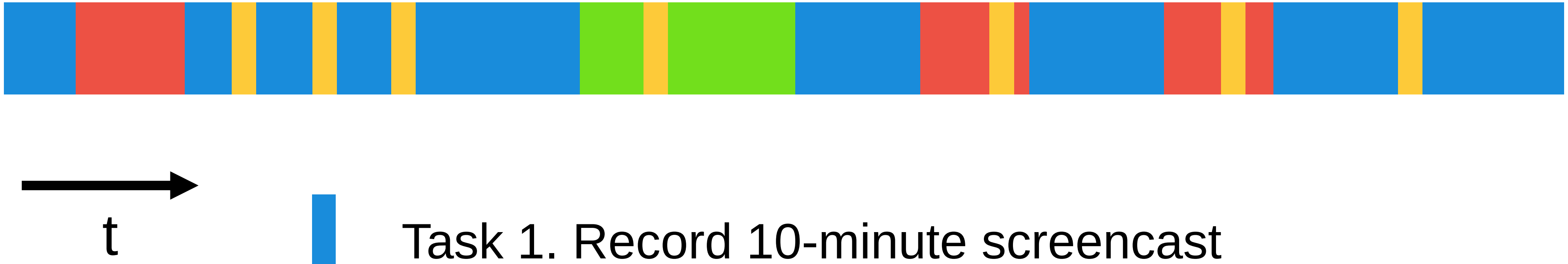
🟥 Task 2. Find and correct 3 mistakes



🟦 Task 1. Record 10-minute screencast
(write a simple hash table with put/get methods then test cases.)

🟥 Task 2. Find and correct 3 mistakes

🟩 Task 3. Add a *remove* method after implementing *put* and *get* methods but before writing test cases

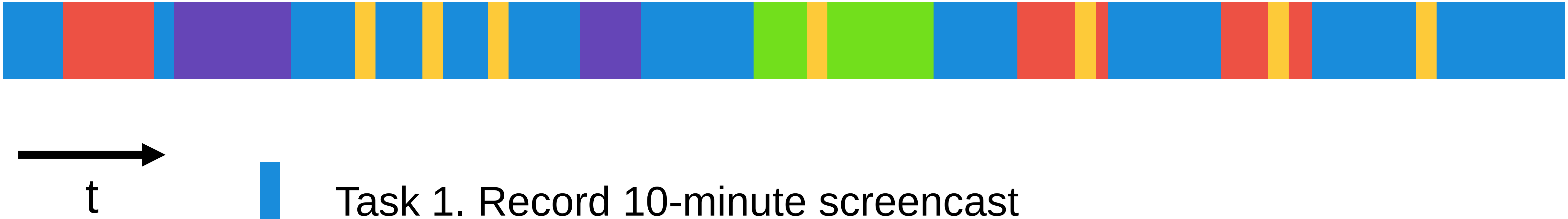


🟦 Task 1. Record 10-minute screencast
(write a simple hash table with put/get methods then test cases.)

🟥 Task 2. Find and correct 3 mistakes

🟩 Task 3. Add a *remove* method after implementing *put* and *get* methods but before writing test cases

🟨 Task 4. Rename one variable that is referenced at least 3 times



🟦 Task 1. Record 10-minute screencast
(write a simple hash table with put/get methods then test cases.)

🟥 Task 2. Find and correct 3 mistakes

🟩 Task 3. Add a *remove* method after implementing *put* and *get* methods but before writing test cases

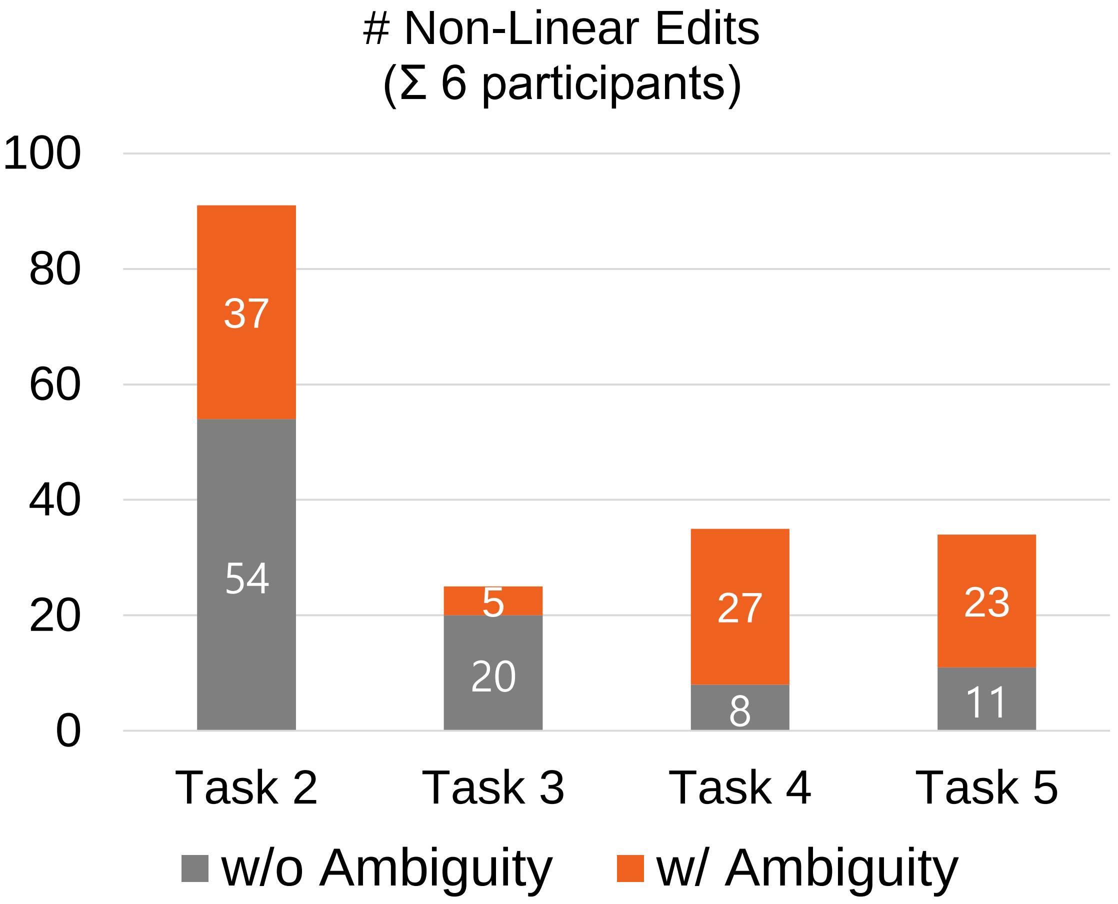
🟨 Task 4. Rename one variable that is referenced at least 3 times

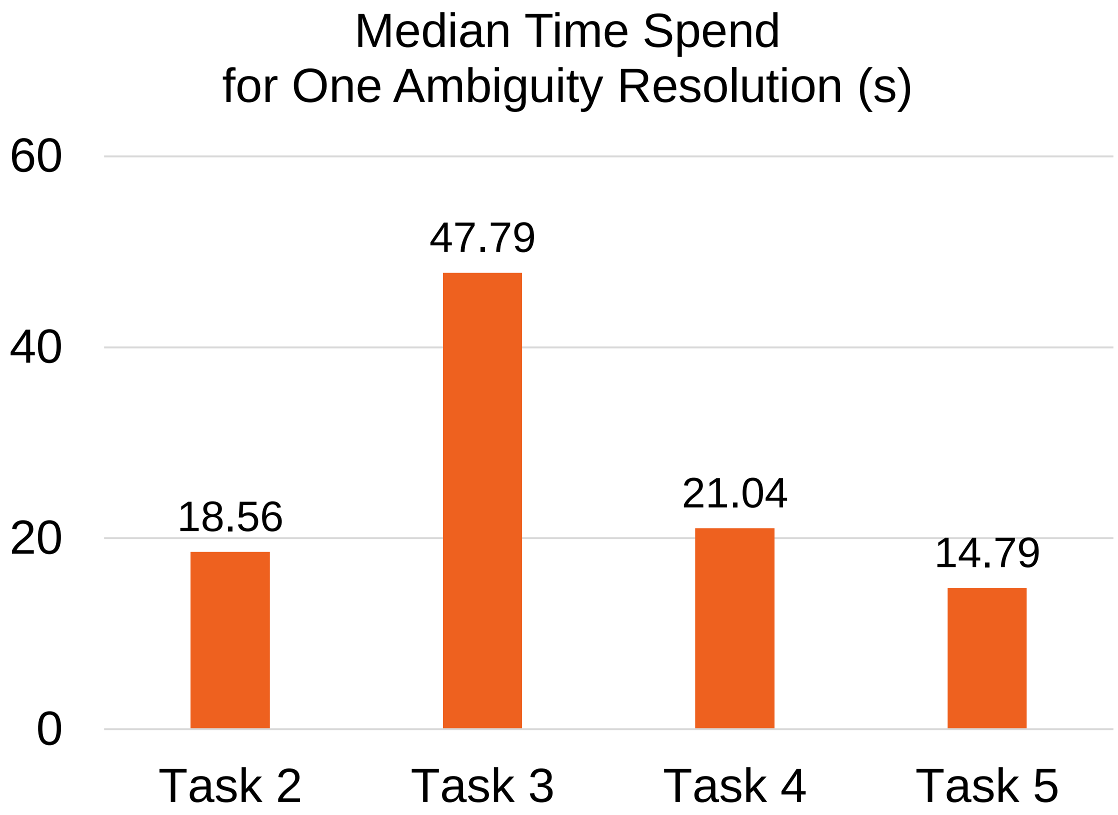
🟪 Task 5. Write docstring for *put* and *get* methods

Result Highlights

All participants completed the tasks successfully.

Participants found the ambiguity resolution process difficult.





Avg. 31 edits / task / participant
49.7% of the edits introduced an ambiguity

Each ambiguity requires avg. 1.45 decisions
Median 19.1 seconds to resolve an ambiguity

Result Highlights

All participants completed the tasks successfully.

Participants found the ambiguity resolution process difficult.

Difficulty of keeping track of all the changes from recording

“I cannot be conscious of the whole changes from the beginning to end when I'm editing.” (S4)

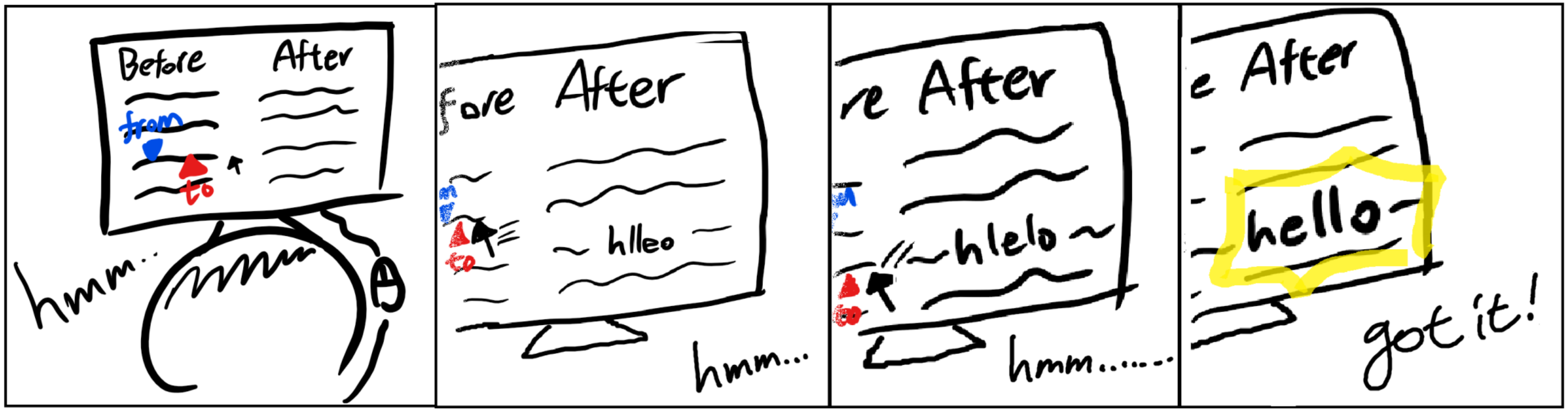
“It seems that I have a habit of unconsciously inserting characters and removing them while I'm thinking.” (S3)

Showing only the “Next Frame” is not enough

“I don’t know what comes next.. with this one character diff.” (S2)

Too much cognitive effort

> *"I didn't give too much thought into 'why' or 'how' before every move because it's complicated. ..." (S6)*



Some users shook the 'from' and 'to' handles until the Next Frame turned out as they desired

Further Research Directions

Need to improve interface design for ambiguity resolution.

- Give users much more context of what they are doing.

- Usability issues.

Reduce the cognitive burden of ambiguity resolution.

- Context-aware suggestions?

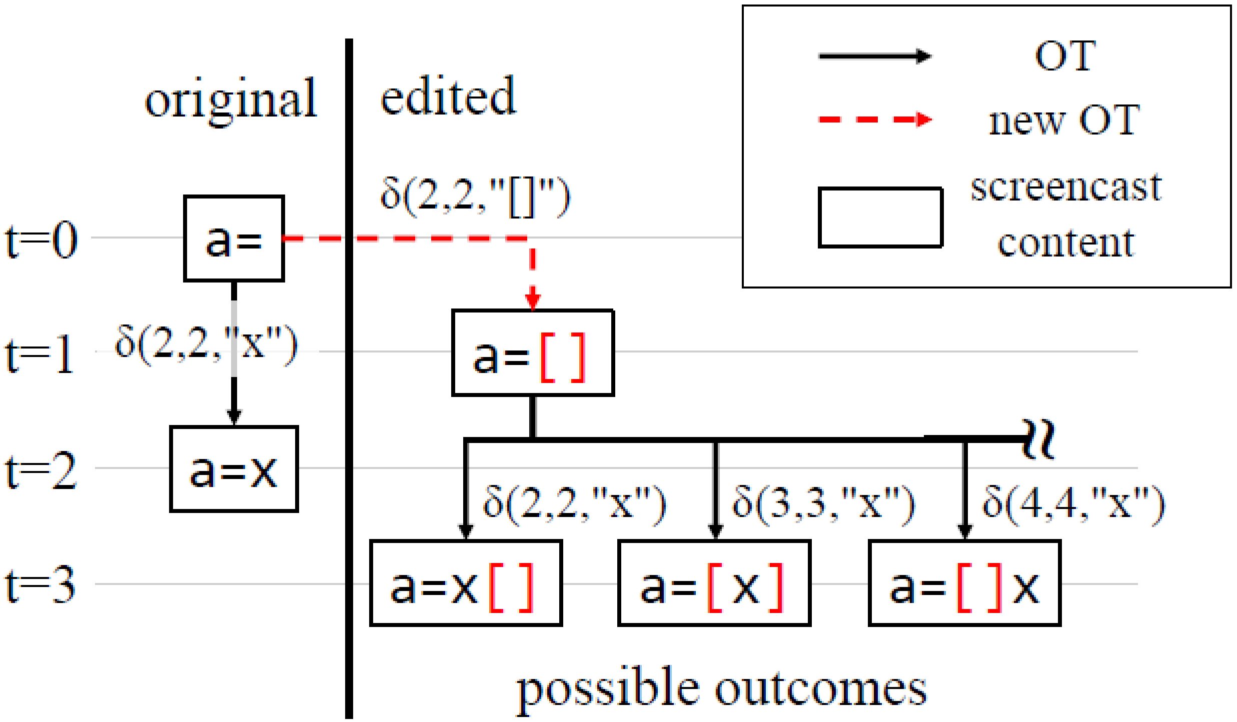
https://github.com/elicast-research/non-linear-edit https://github.com/elicast-research/non-linear-edit

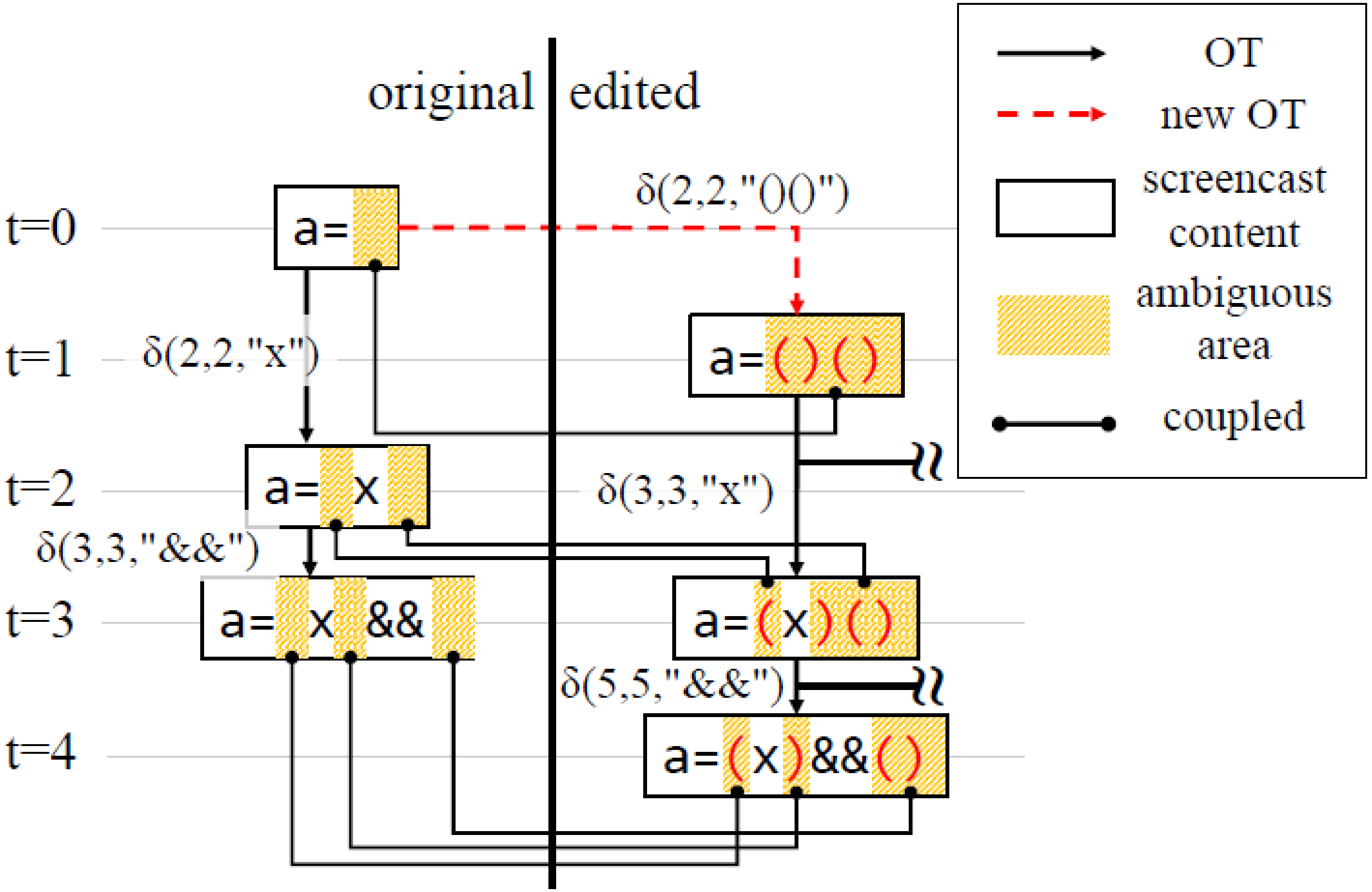
Non-Linear Editing of Text-Based Screencasts

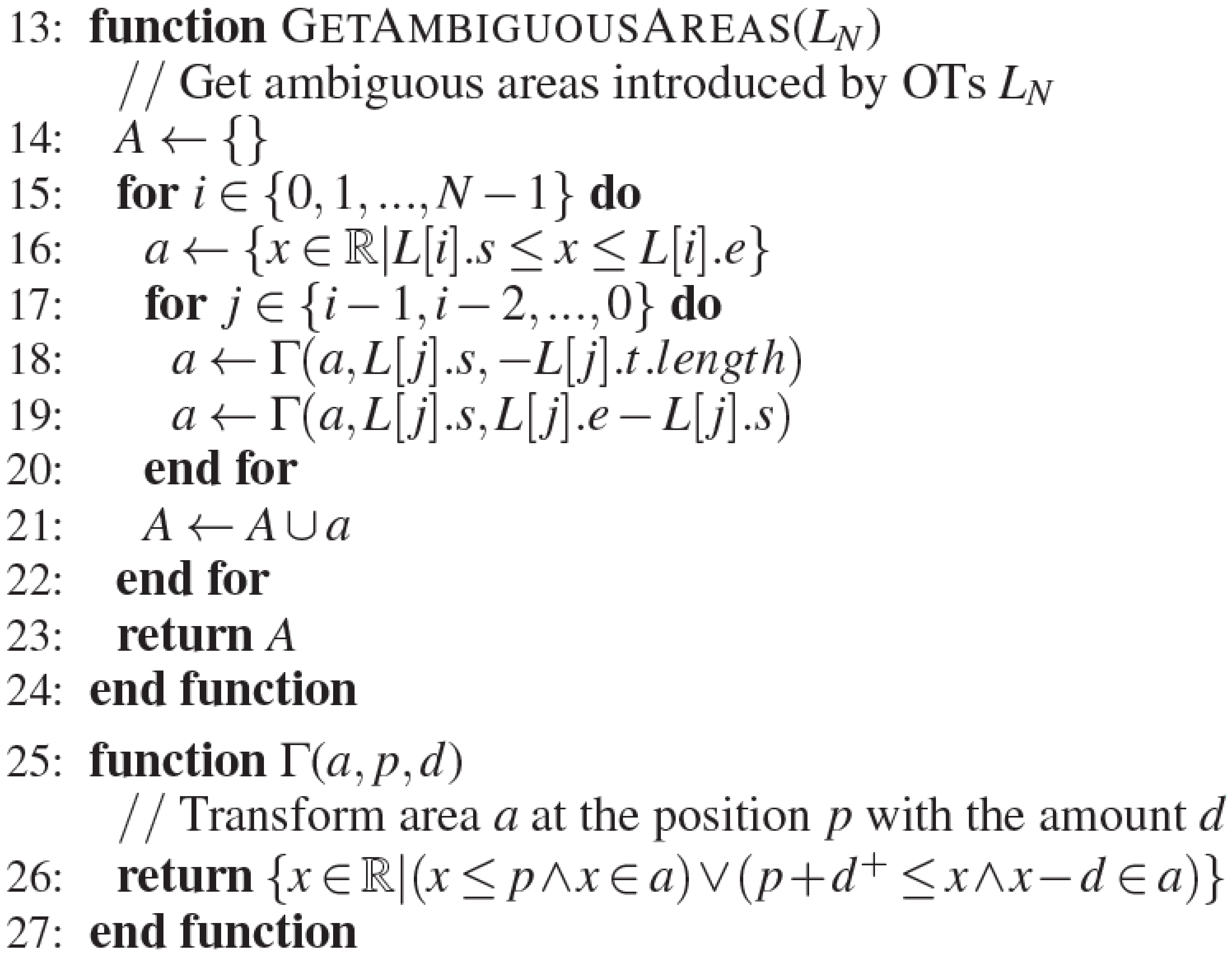
Jungkook Park* KAIST

Yeong Hoon Park* University of Minnesota

Alice Oh KAIST


original
edited
t=0
t=1
t=2
t=3
a=
δ(2,2,"[]")
δ(2,2,"x")
a=[]
a=x
δ(2,2,"x")
δ(3,3,"x")
δ(4,4,"x")
a=x[]
a=[x]
a=[]x
possible outcomes
OT
new OT
screencast content


original
edited
OT
new OT
screencast content
ambiguous area
coupled
t=0
t=1
t=2
t=3
t=4
a=
δ(2,2,"()()")
a=()()
δ(2,2,"x")
a= x
δ(3,3,"x")
δ(3,3,"&&")
a= x &&
a=(x)()
δ(5,5,"&&")
a=(x)&&()



```
function GetAmbiguousAreas($(L_N)$)
    // Get ambiguous areas introduced by OTs $L_N$
    $A \leftarrow \{\}$
    for $i \in \{0, 1, \ldots, N-1\}$ do
        $a \leftarrow \{x \in \mathbb{R} | L[i].s \leq x \leq L[i].e\}$
        for $j \in \{i-1, i-2, \ldots, 0\}$ do
            $a \leftarrow \Gamma(a, L[j].s, -L[j].t.length)$
            $a \leftarrow \Gamma(a, L[j].s, L[j].e - L[j].s)$
        end for
        $A \leftarrow A \cup a$
    end for
    return $A$
end function
function $\Gamma(a, p, d)$
    // Transform area $a$ at the position $p$ with the amount $d$
    return $\{x \in \mathbb{R} | (x \leq p \wedge x \in a) \vee (p + d^+ \leq x \wedge x - d \in a)\}$
end function
```

```
function GetCoupledAmbiguousArea($L_N$)
  // Get a coupled ambiguous area introduced by OTs $L_N$
  $A_{before} \leftarrow$ GetAmbiguousAreas($L_N$)
  $L_N^{inv} \leftarrow L_N.map(x \rightarrow x^{-1}).reverse()$
  $A_{after} \leftarrow$ GetAmbiguousAreas($L_N^{inv}$)
  $C \leftarrow []$
  while $A_{before} \neq \emptyset$ do  // always $|A_{before}| = |A_{after}|$
    Pop leftmost interval $a_{before}$ from $A_{before}$
    Pop leftmost interval $a_{after}$ from $A_{after}$
    $C.append((a_{before} \rightarrow a_{after}))$
  end while
  return $C$
end function
```

```
function GetPossibleTransform($(C, x)$)
  // Get all possible transformations of OT $x$ given by the coupled ambiguous area $C$
  $p_a \leftarrow a_{after}$ s.t. $(a_{before} \rightarrow a_{after}) \in C \wedge x.s \in a_{before}$
  $p_b \leftarrow a_{after}$ s.t. $(a_{before} \rightarrow a_{after}) \in C \wedge x.e \in a_{before}$
  if $p_a = \emptyset$, then $p_a \leftarrow [x.s + \Delta(C, x.s), x.s + \Delta(C, x.s)]$
  if $p_b = \emptyset$, then $p_b \leftarrow [x.e + \Delta(C, x.e), x.e + \Delta(C, x.e)]$
  $P_x \leftarrow []$
  for $a', b' \in p_a \times p_b$ do
    if $a' \leq b'$, then $P_x.append(\delta(a', b', x.t))$
  end for
  return $P_x$
end function
function $\Delta(C, p)$
  // Get the amount of position shift at $p$ by the coupled ambiguous area $C$
  $d \leftarrow \sum_{\substack{(a_{before} \rightarrow a_{after}) \in C \\ \wedge a_{before} \leq p}} (a_{after}.length - a_{before}.length)$
  return $d$
end function
```

```
function ReplaceScreencast($L_O, L_N, s, e$)
    // Replace a part of screencast $L_O[s:e]$ with OTs $L_N$
    $L_T^{inv} \leftarrow L_O[s:e].map(x \rightarrow x^{-1}).reverse()$
    $L'_N \leftarrow L_T^{inv} \cdot L_N$
    $L \leftarrow \text{InsertScreencast}(L_O, L'_N, e)$
    $L' \leftarrow L[0:s] \cdot L[2e-s:]$
    return $L'$
end function
```

```
function InsertScreencast($L_O, L_N, k$)
  // Insert a new screencast $L_N$ between two OTs $L_O[k-1]$ and $L_O[k]$
  $L \leftarrow L_O[0:k] \cdot L_N$
  $C \leftarrow$ GetCoupledAmbiguousArea($L_N$)
  for $i \in \{0, 1, ..., |L_O| - k - 1\}$ do
    $x := L_O[k+i]$
    $Y \leftarrow$ GetPossibleTransform($C, x$)
    if $|Y| > 1$ then
      Ask user to choose one $y$ from $Y$
    else
      $y \leftarrow Y[0]$  // No ambiguity
    end if
    $L.append(y)$
    for $(a_{before} \rightarrow a_{after}) \in C$ do
      $a_{before} \leftarrow \Gamma(\Gamma(a_{before}, x.s, x.s - x.e), x.s, x.t.length)$
      $a_{after} \leftarrow \Gamma(\Gamma(a_{after}, y.s, y.s - y.e), y.s, y.t.length)$
    end for
  end for
  return $L$
end function
```