Systematic Black-Box Analysis of Collaborative Web Applications

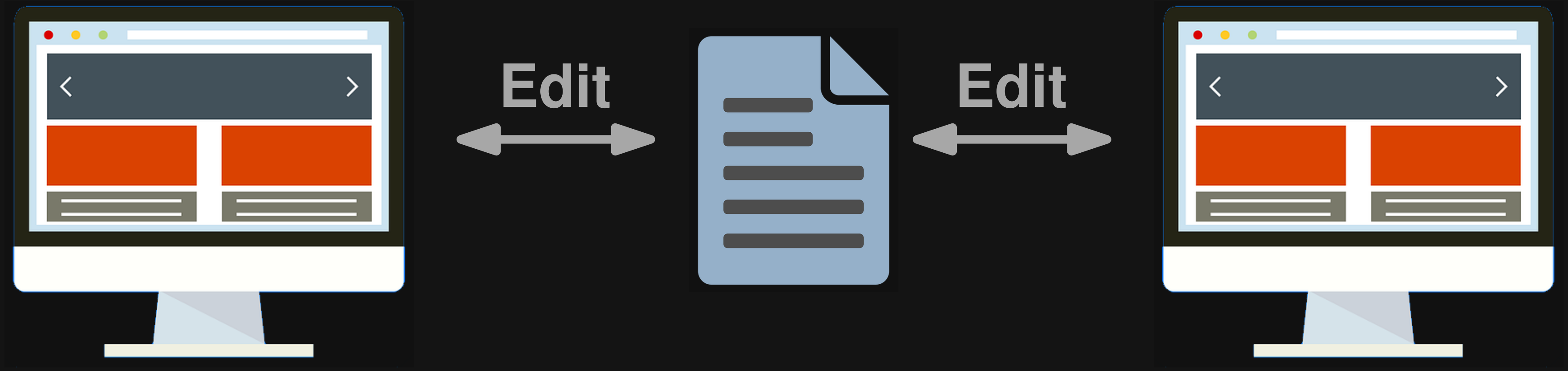
Michael Pradel
TU Darmstadt

Joint work with Marina Billes (TU Darmstadt)
and Anders Møller (Aarhus University)

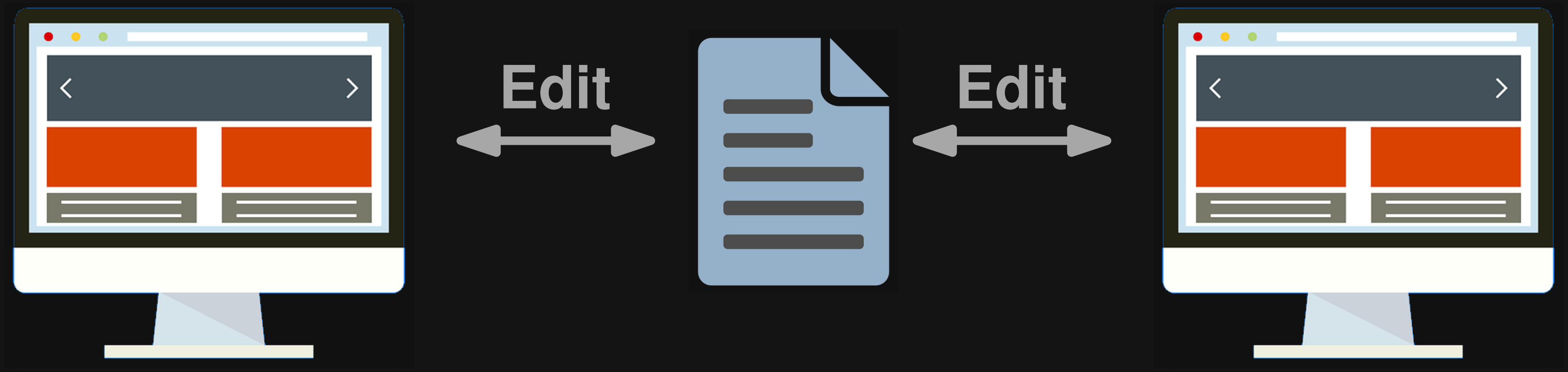
More details: Paper at PLDI 2017

Collaborative Web Applications

- Google Docs, MS Office Online, Cloud9 IDE, and many others
- Multiple interacting users
- Goal: Eventual consistency

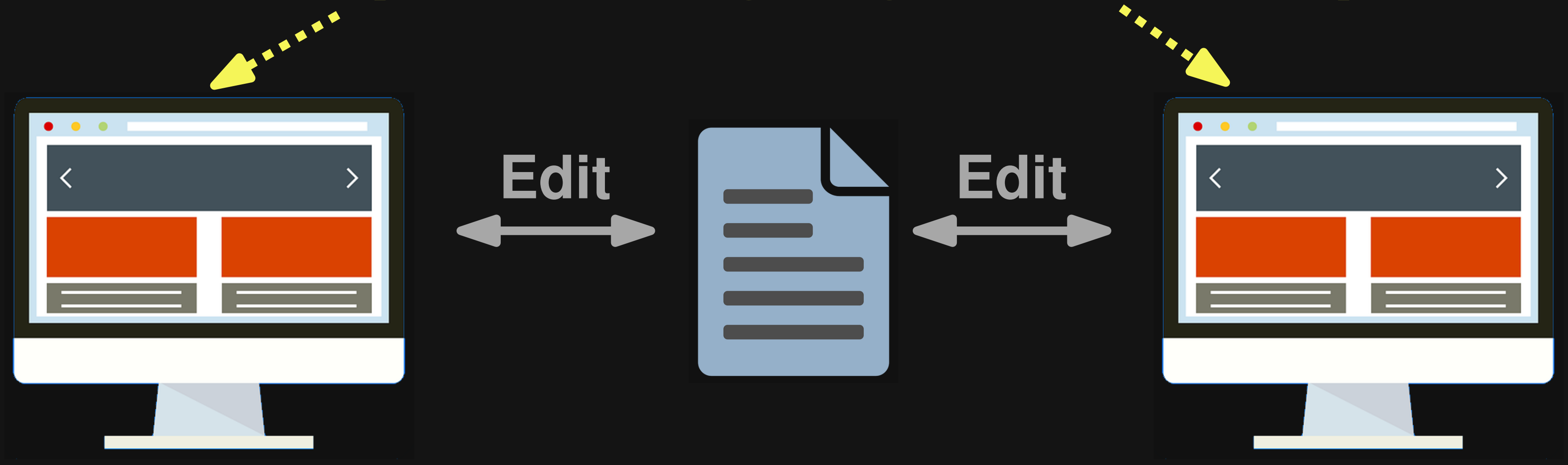


Hidden Complexity



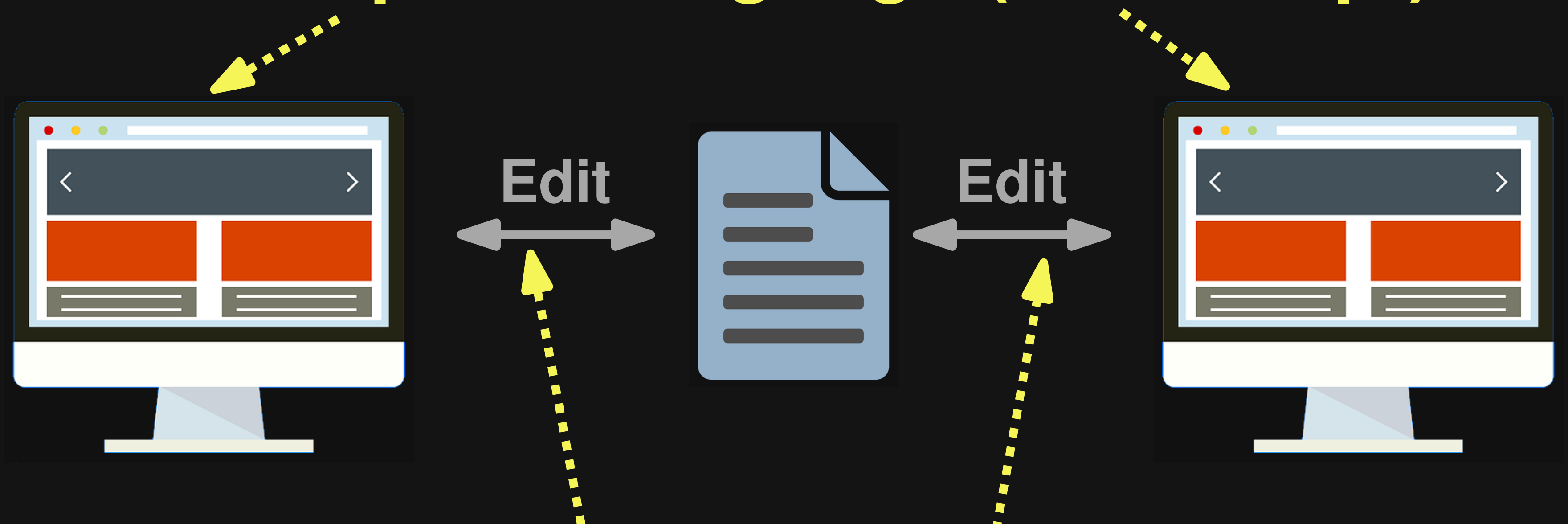
Hidden Complexity

Client-side: Heterogenous browsers with errorprone language (JavaScript)



Hidden Complexity

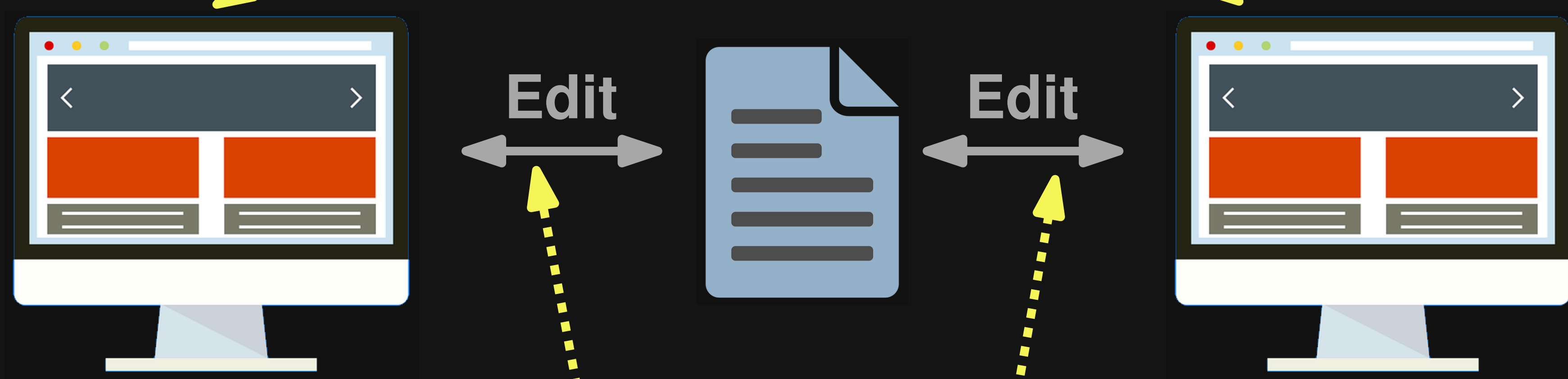
Client-side: Heterogenous browsers with errorprone language (JavaScript)



Concurrent interactions

Hidden Complexity

Client-side: Heterogenous browsers with errorprone language (JavaScript)



Concurrent interactions

Geographically distributed system

Example Bug

Client 1

Client 2

(Google Docs)

Example Bug

Client 1

Write ”this” ...

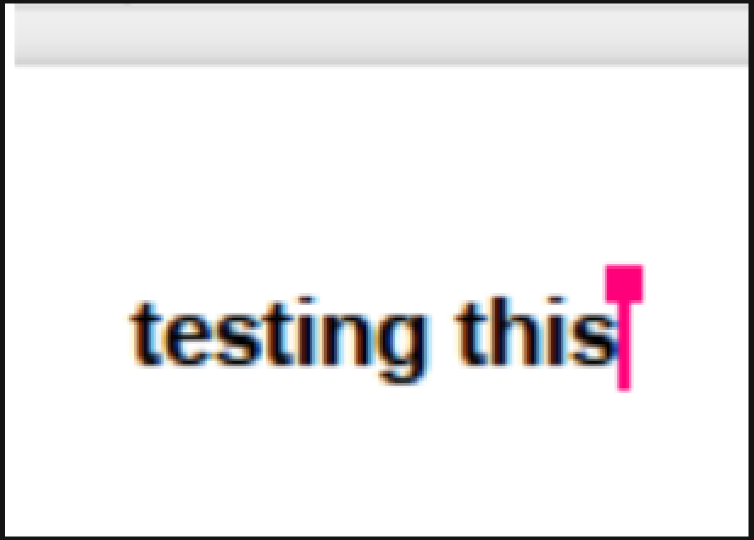
Client 2

Delete line ...

(Google Docs)

Example Bug

Client 1

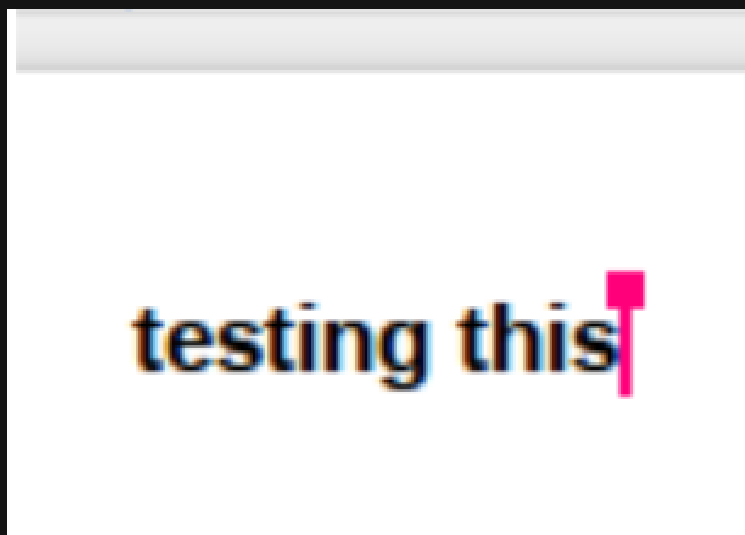


Client 2

(Google Docs)

Example Bug

Client 1



Client 2

Synchronize ...

(Google Docs)

Example Bug

Client 1

Client 2

Inconsistent state

(Google Docs)

Challenges for Analysis

How to analyze such a system?

- Huge space of client actions
- Huge space of concurrent interleavings
- Complex system with various components

Challenges for Analysis

How to analyze such a system?

- Huge space of client actions
- Huge space of concurrent interleavings
- Complex system with various components

Impossible to fully explore

Impossible to fully understand and control

This Talk: Simian

Technique for analyzing collaborative web applications

- **Automatic**: No need to specify interactions
- **Scalable**: Blackbox view of system
- **Systematic**: Bounded exploration of all potential conflicts
- **Precise**: No false positives

Correctness

Operational transformation [Ellis & Gibbs, 1989]

Non-blocking concurrency control

$$\begin{array}{ccc} \sigma_1 & \qquad & \sigma_2 \\ \big\downarrow op_1 & & \big\downarrow op_2 \end{array}$$

σ .. state, op .. operation, T .. transformation function

Correctness

Operational transformation [Ellis & Gibbs, 1989]

Non-blocking concurrency control

$$\begin{array}{ccc} \sigma_1 & \quad & \sigma_2 \\ \Big\downarrow {\scriptstyle op_1,\ T(op_2)} & & \Big\downarrow {\scriptstyle op_2,\ T(op_1)} \\ \sigma'_1 & & \sigma'_2 \end{array}$$

σ .. state, op .. operation, T .. transformation function

Correctness

Operational transformation [Ellis & Gibbs, 1989]

Non-blocking concurrency control

$$\sigma_1 \xrightarrow{op_1 \; T(op_2)} \sigma_1' \qquad \sigma_2 \xrightarrow{op_2 \; T(op_1)} \sigma_2'$$

Transformation ensures:

$$\sigma_1 \equiv \sigma_2 \implies \sigma_1' \equiv \sigma_2'$$

σ .. state, op .. operation, T .. transformation function

Correctness

Operational transformation [Ellis & Gibbs, 1989]

Non-blocking concurrency control

$$
\begin{tikzcd}
\sigma_1 \arrow[d, "\begin{matrix} op_1 \\ T(op_2) \end{matrix}"] \\
\sigma_1'
\end{tikzcd}
\qquad
\begin{tikzcd}
\sigma_2 \arrow[d, "\begin{matrix} op_2 \\ T(op_1) \end{matrix}"] \\
\sigma_2'
\end{tikzcd}
$$

Transformation ensures:

$$\sigma_1 \equiv \sigma_2 \implies \sigma_1' \equiv \sigma_2'$$

σ .. state, op .. operation, T .. transformation function

Correctness = Precedence + Convergence

Correctness

Operational transformation [Ellis & Gibbs, 1989]

Non-blocking concurrency control

$$
\begin{array}{ccc}
\sigma_1 & & \sigma_2 \\
\Big\downarrow {\scriptstyle op_1 \atop T(op_2)} & & \Big\downarrow {\scriptstyle op_2 \atop T(op_1)} \\
\sigma_1' & & \sigma_2'
\end{array}
$$

Transformation ensures:

$$\sigma_1 \equiv \sigma_2 \implies \sigma_1' \equiv \sigma_2'$$

σ .. state, op .. operation, T .. transformation function

Correctness = Precedence + Convergence

Focus of Simian

Overview of Simian

Set of user actions

Phase 1:
Sequential learning

Potential conflicts

Phase 2:
Concurrent analysis

Inconsistencies

Overview of Simian

Set of user actions

Phase 1:
Sequential learning

Potential conflicts

Phase 2:
Concurrent analysis

Inconsistencies

Blackbox reasoning about states and actions

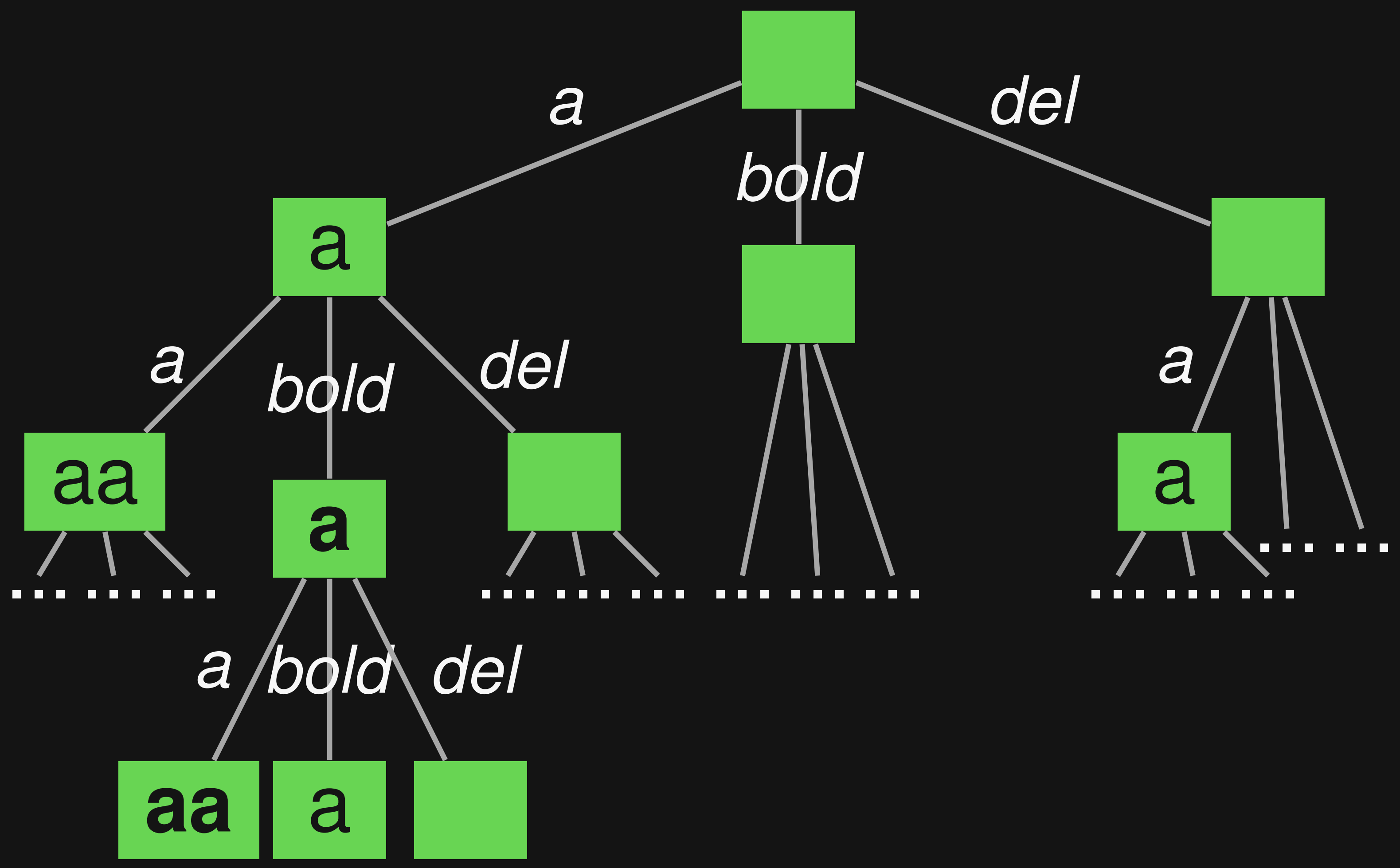
Actions

Action: Logical step triggered by a user

- **May consist of multiple implementation-level steps**
- **Examples**
  - Insert text ”a”
  - Mark current line and make it bold
  - Delete last character (backspace)

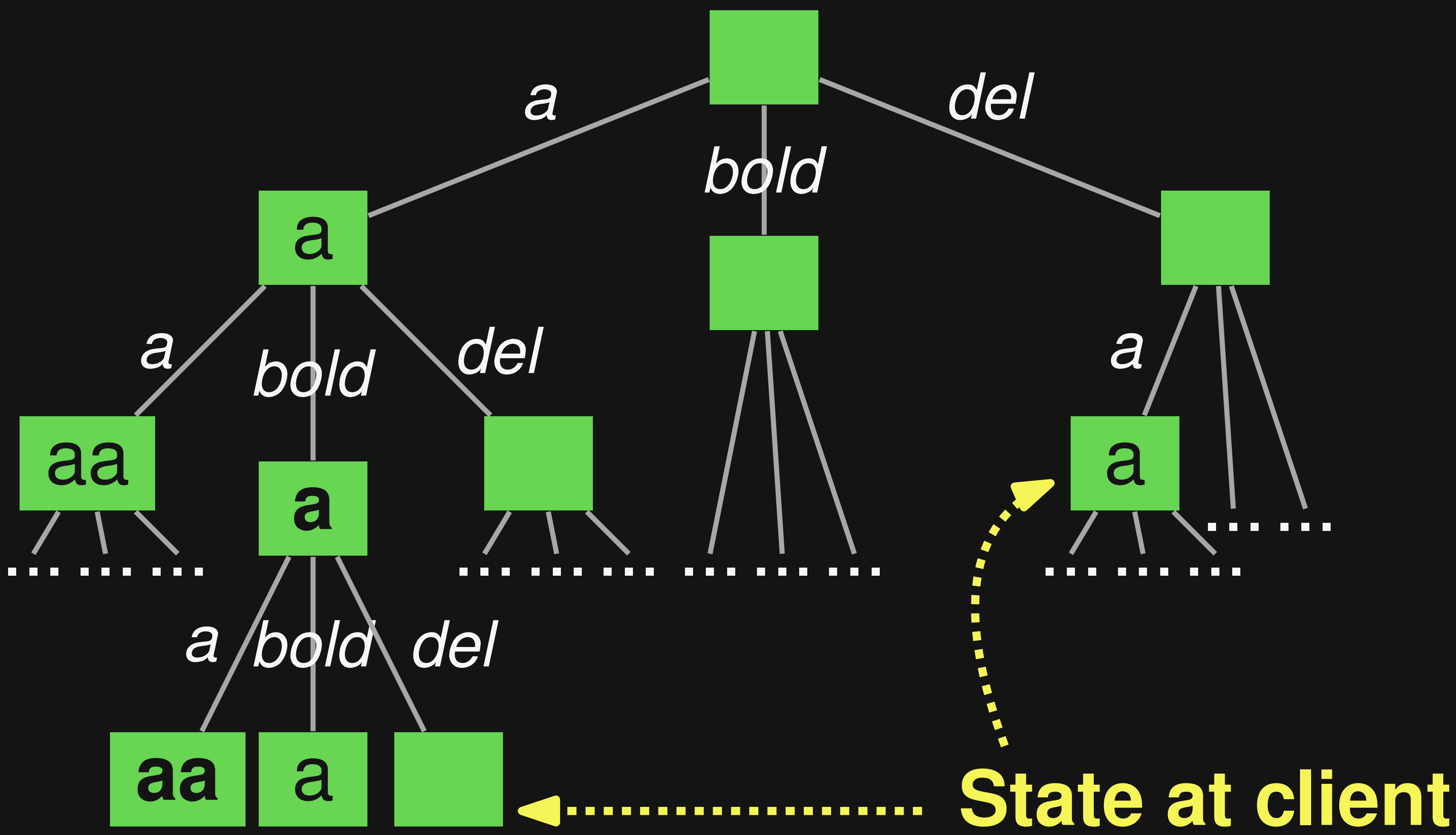
Action Tree

Sequences of actions by a single user



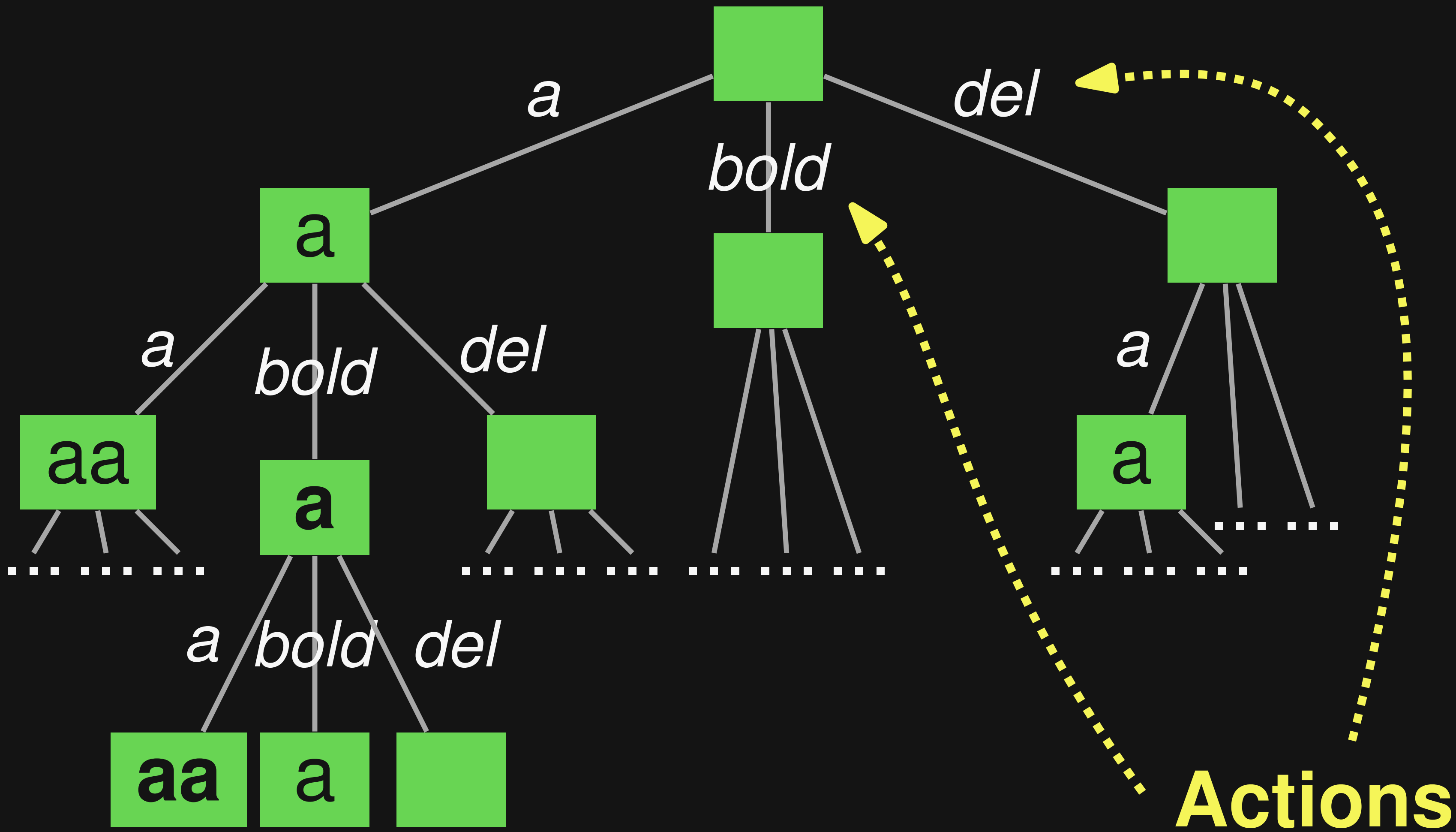
Action Tree

Sequences of actions by a single user



Action Tree

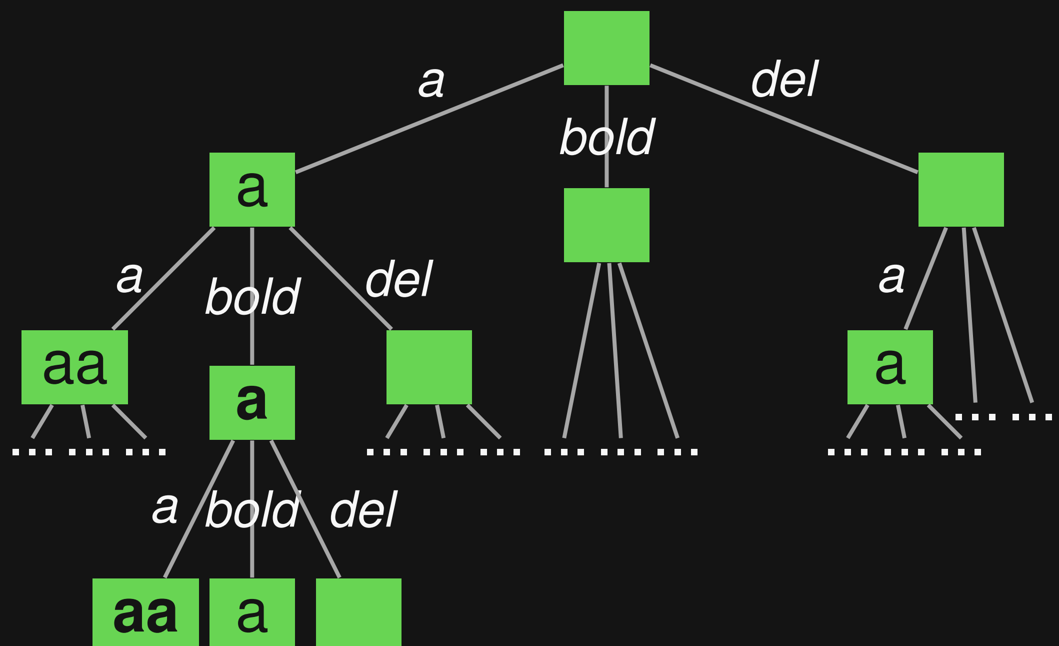
Sequences of actions by a single user



Phase 1: Sequential Learning

Systematic exploration of action tree

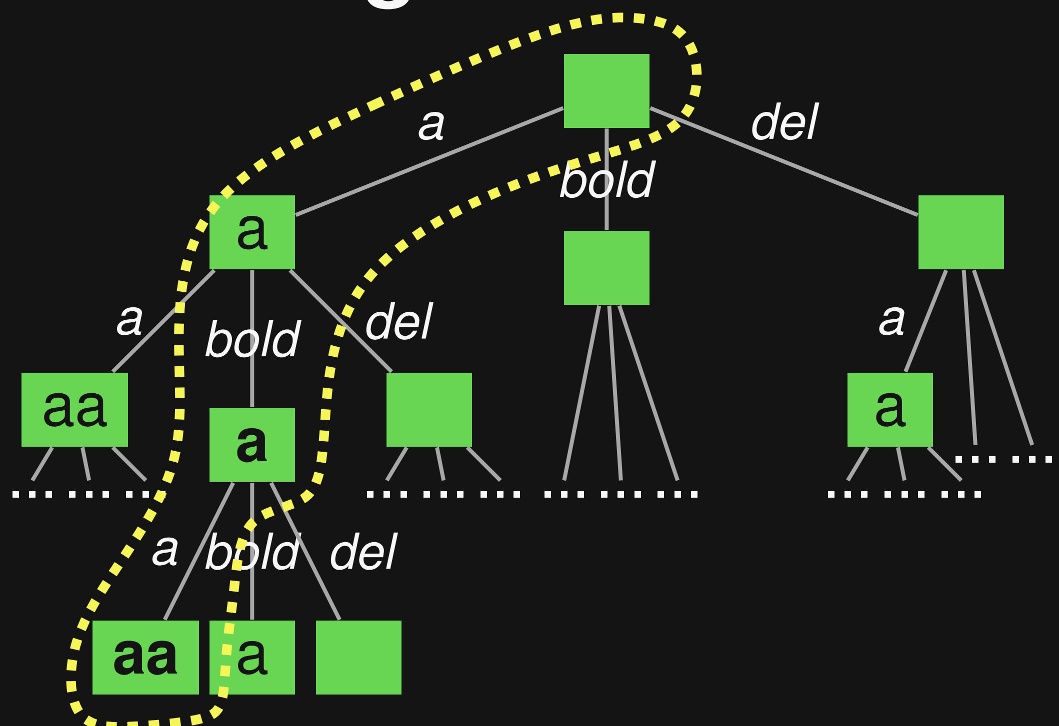
- **Full traversal up to maximum depth k**
- **Execute one single-client interaction per path through the tree**



Phase 1: Sequential Learning

Systematic exploration of action tree

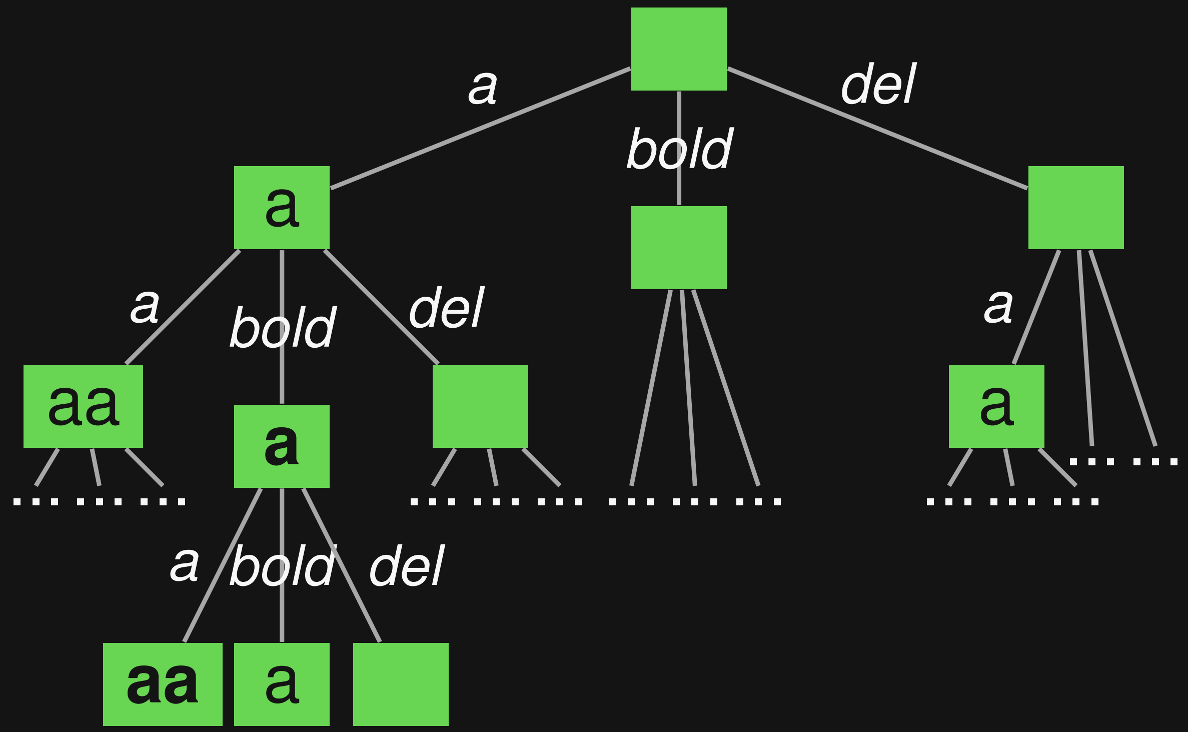
- **Full traversal up to maximum depth k**
- **Execute one single-client interaction per path through the tree**



Potential Conflicts

Identify **potential conflicts**

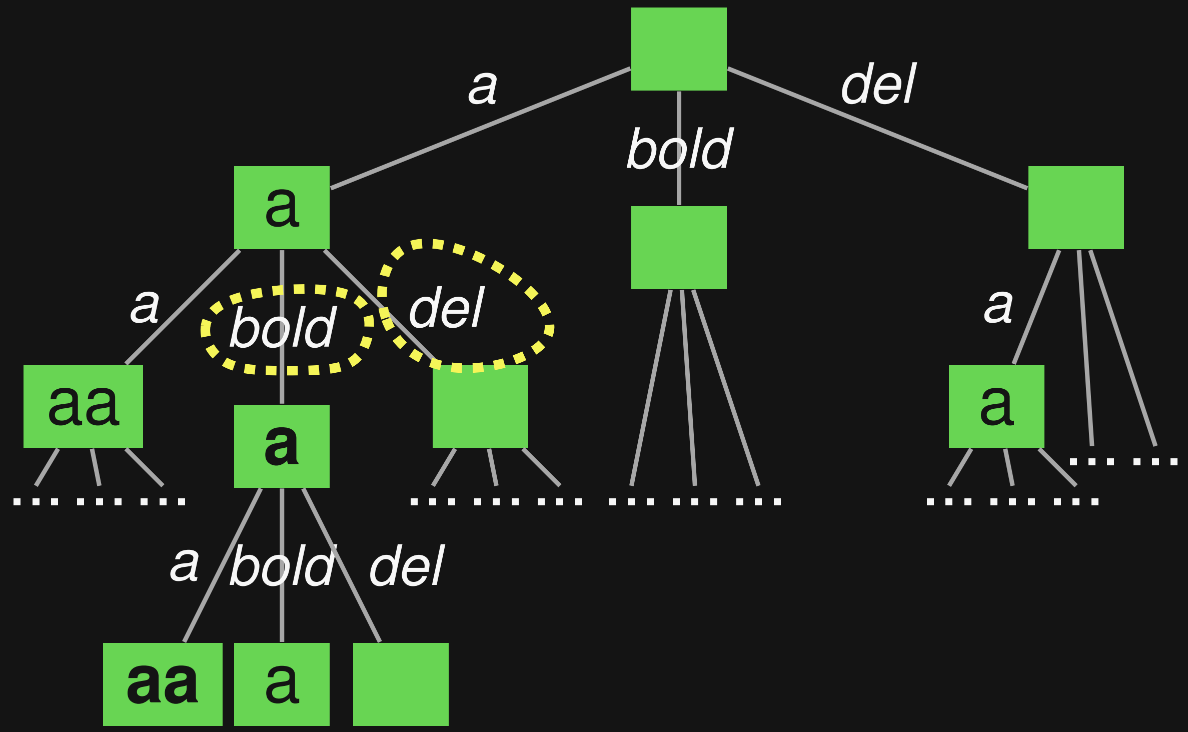
- **Actions that affect the same data when triggered in the same state**



Potential Conflicts

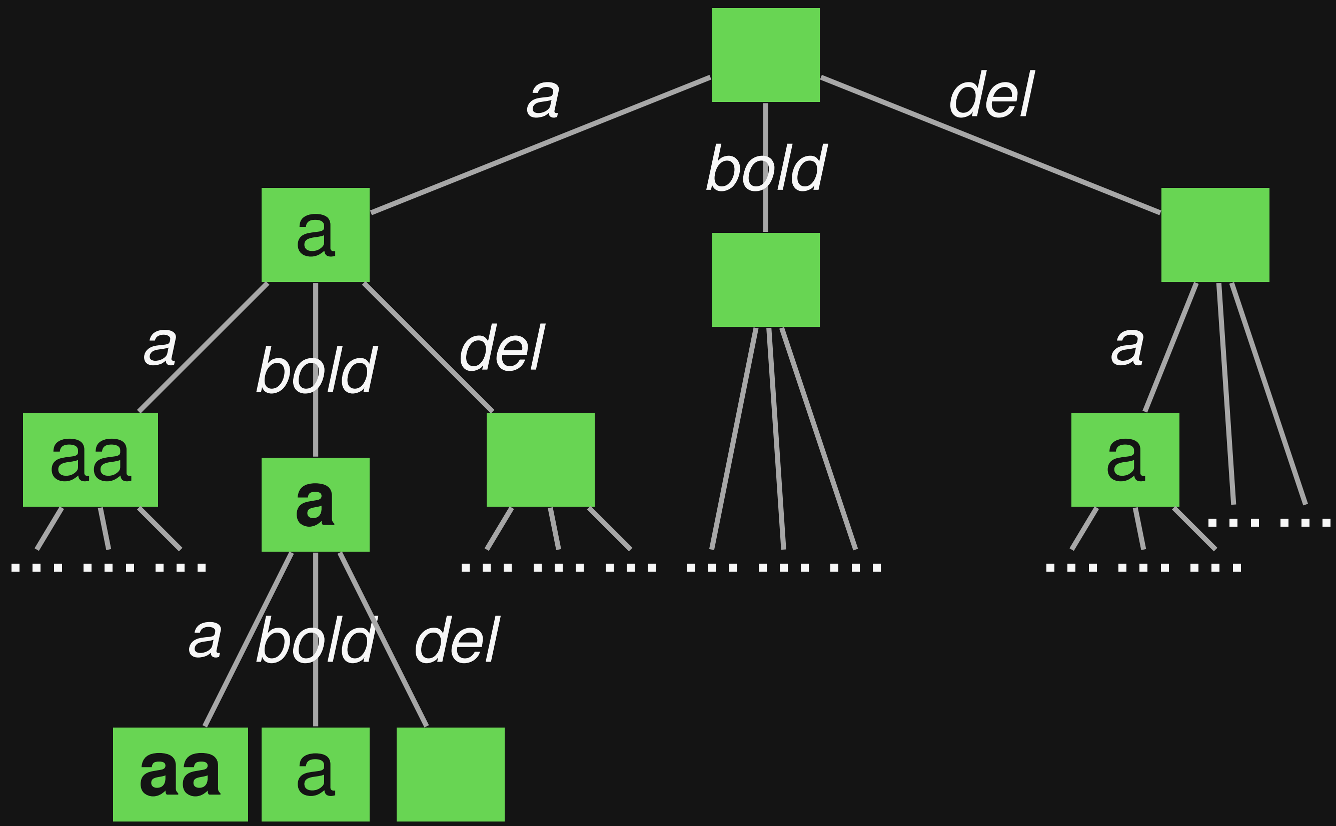
Identify potential conflicts

- Actions that affect the same data when triggered in the same state



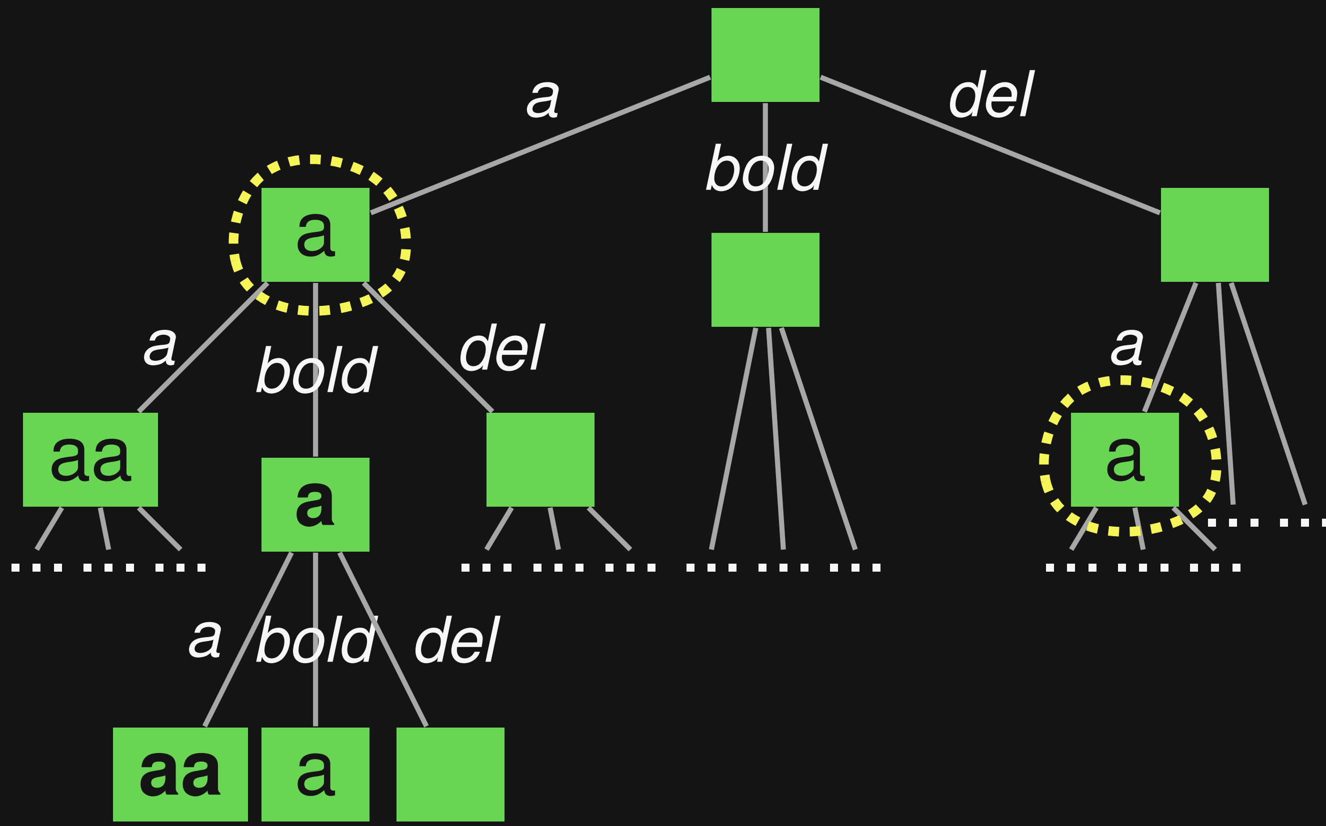
Equivalent States

Identify equivalent states



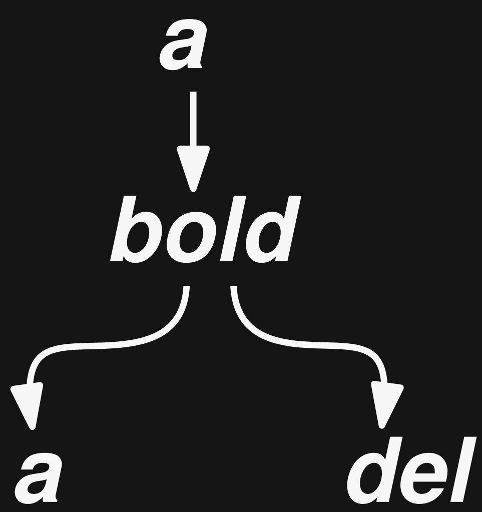
Equivalent States

Identify equivalent states



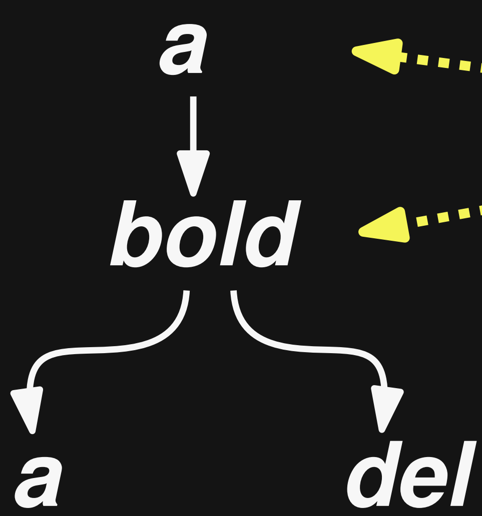
Multi-Client Interactions

Two clients trigger actions concurrently



Multi-Client Interactions

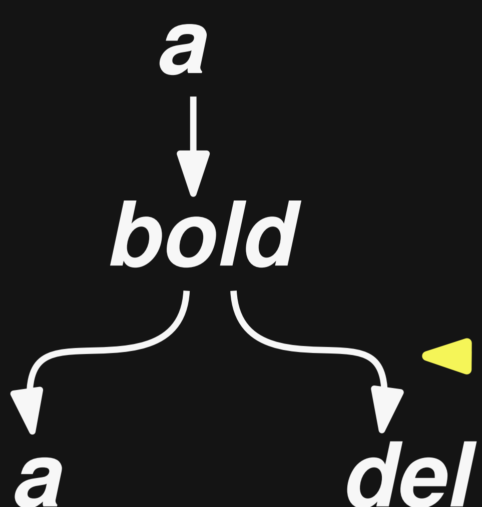
Two clients trigger actions concurrently



Sequential prefix:
Executed by client 1

Multi-Client Interactions

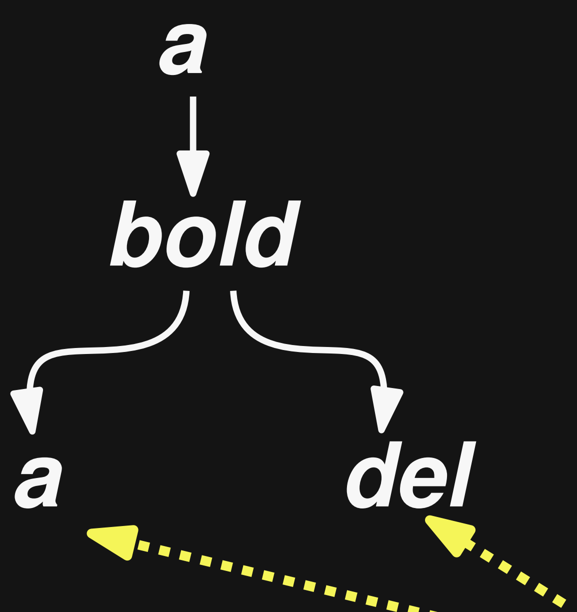
Two clients trigger actions concurrently



Wait until clients have synchronized

Multi-Client Interactions

Two clients trigger actions concurrently



Concurrent suffixes:
Executed by client 1 and 2

Phase 2: Concurrent Analysis

For each potential conflict:

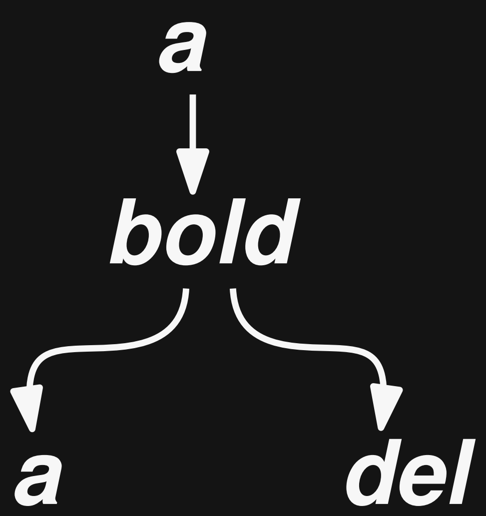
1) Synthesize a multi-client interaction

- Don’t repeat same suffixes in equivalent states

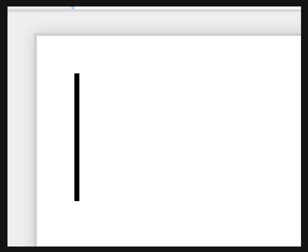
2) Check if clients eventually converge

Naive approach (for comparison): Synthesize and check all multi-client interactions

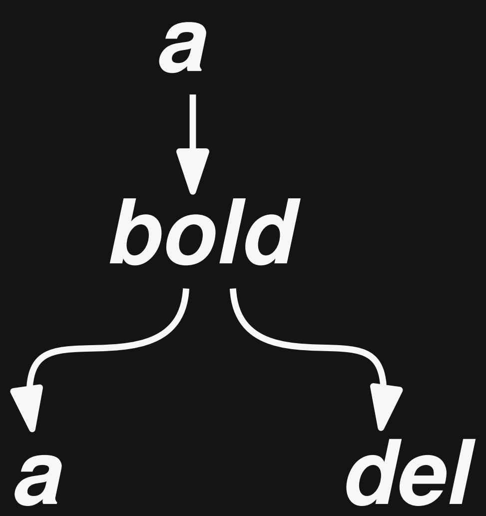
Example



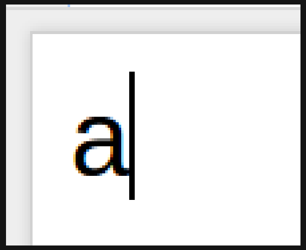
Client 1



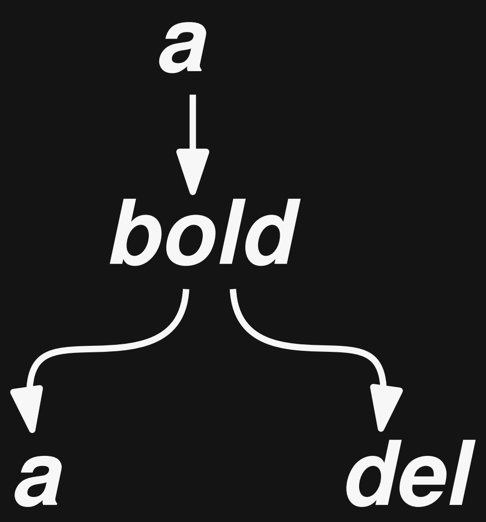
Example



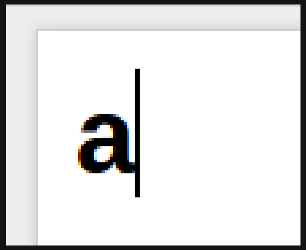
Client 1



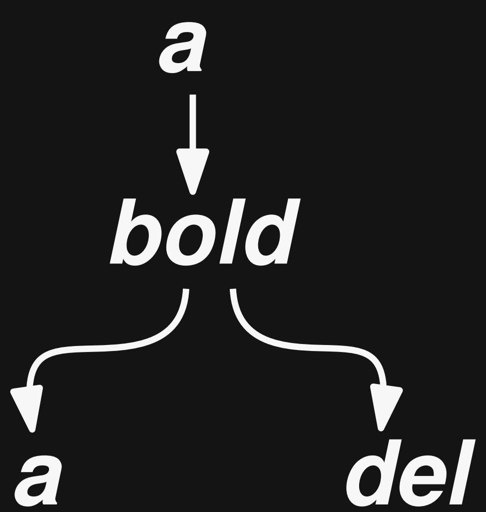
Example



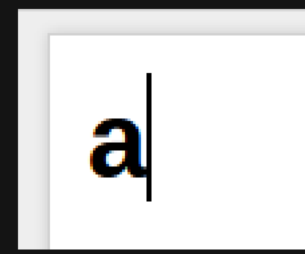
Client 1



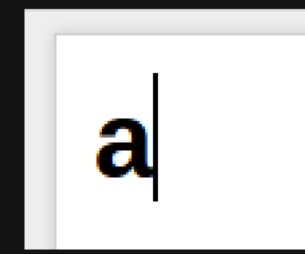
Example



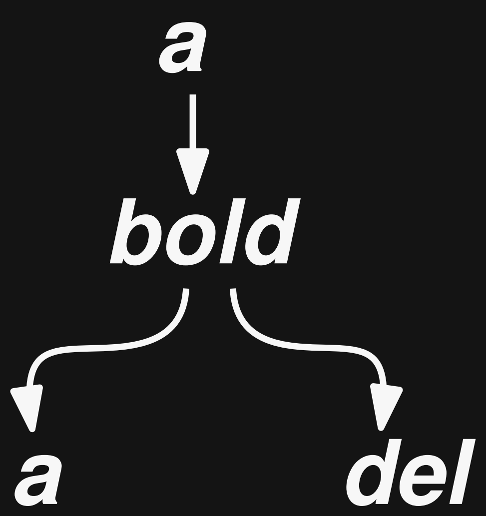
Client 1



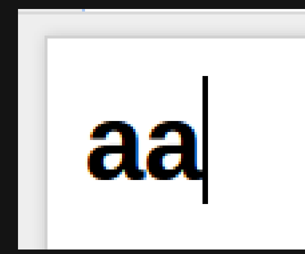
Client 2



Example

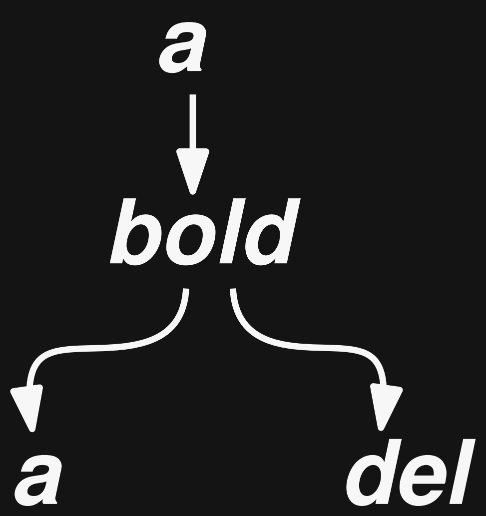


Client 1

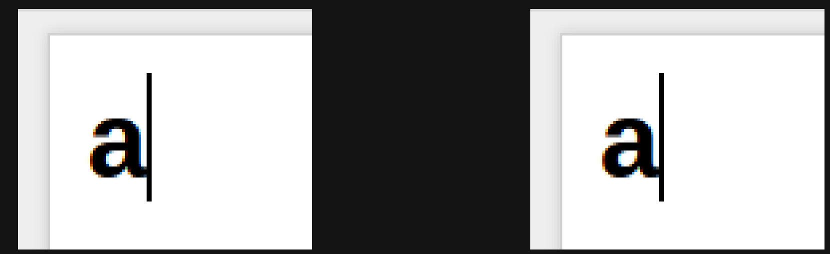


Client 2

Example

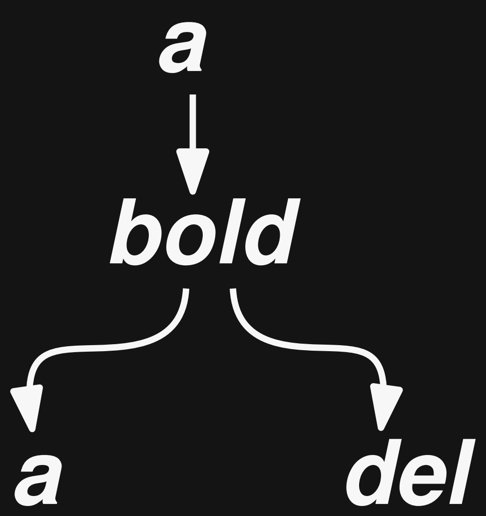


Client 1 Client 2

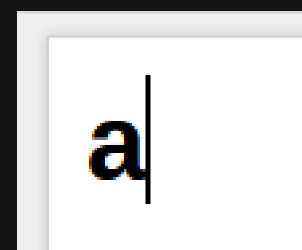


Correct outcome

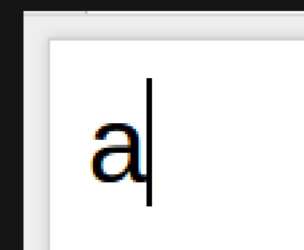
Example



Client 1



Client 2



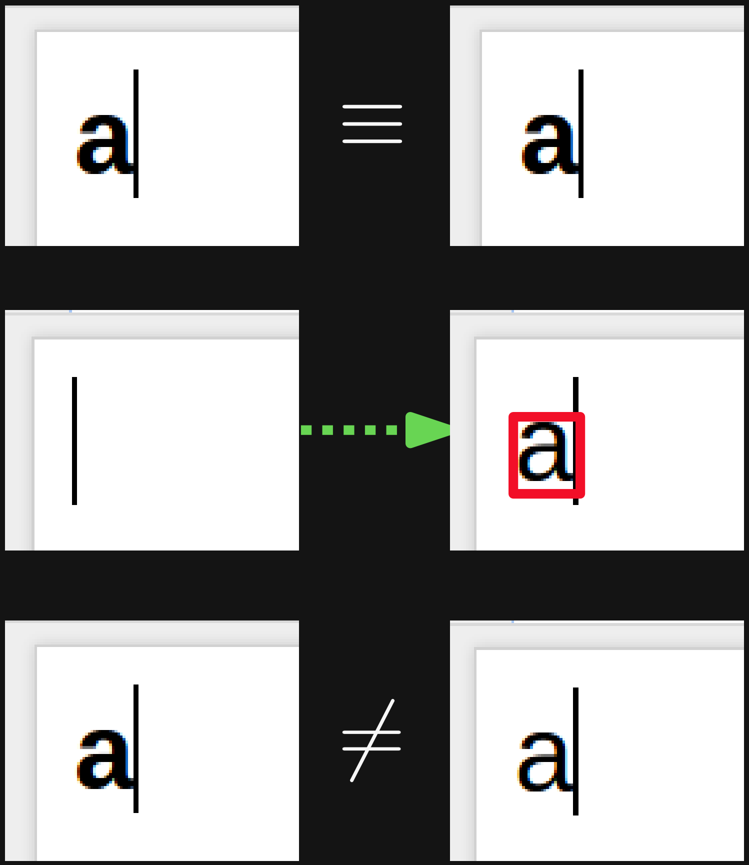
Incorrect outcome

Black-Box Reasoning

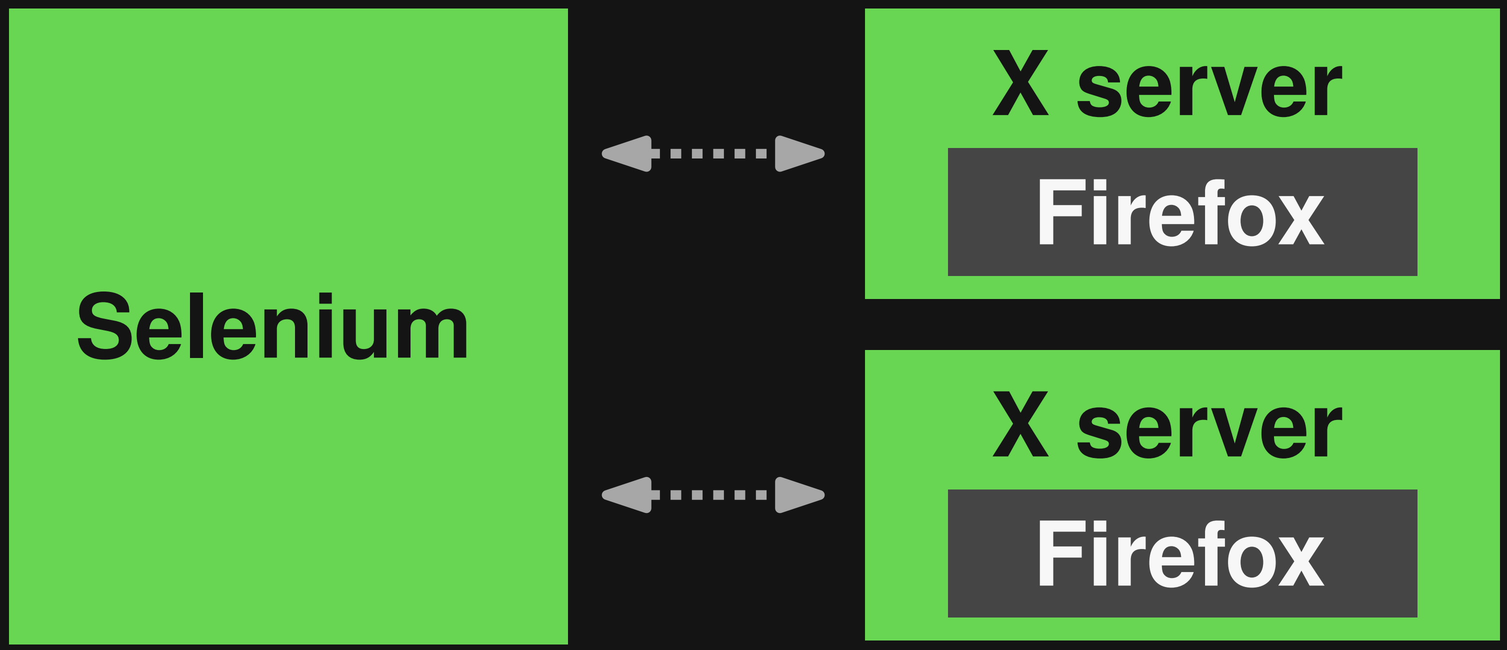
How to reason about states?

Pixel-based state abstraction

- Equivalent states if same pixels
- Conflicting actions if overlap of affected pixels
- Inconsistent states if different pixels



Implementation



- Approximate comparison of screenshots
- Blinded areas

https://github.com/marinabilles/simian

Evaluation

Google Docs

FIREPAD
Open source collaborative code and text editing

ownCloud

Ten actions:

Type “a”
Press Return
Toggle bold on line before cursor
Set font face to Verdana on line before cursor
Select and delete line before cursor
Press Tab
Press Space
Type “b”
Toggle italic on line after cursor
Set font size to 18 on line before cursor

Inconsistencies

Various issues in all three systems

	Exploration depth k		
	1	2	3
Google Docs	0	0	37
Firepad	0	5	32
ownCloud	1	15	126

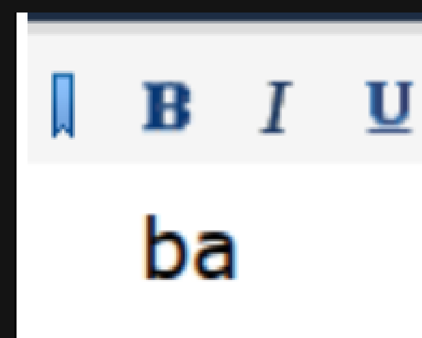
Examples

Inconsistent fonts

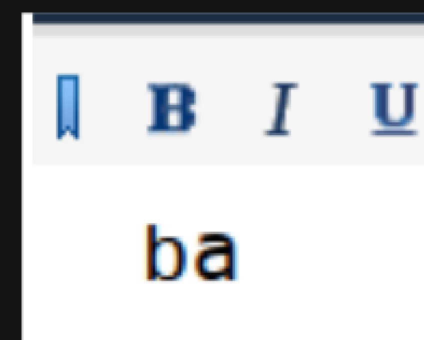
Set font "Verdana"

Type "a" ***Type "b"***

Client 1



Client 2



(ownCloud)

Examples

Incorrect selection shown

Type "a"

↓

Toggle bold

Type "a"   *Toggle bold*

Client 1

Client 2

(Firepad)

Examples

Text fragments are swapped

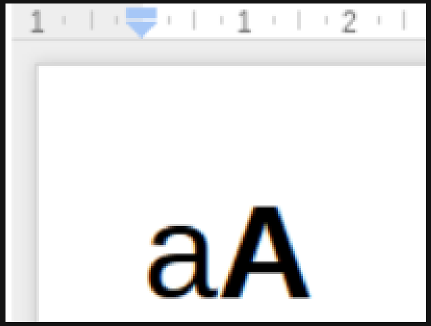
Both clients see incorrect state

Type "a"

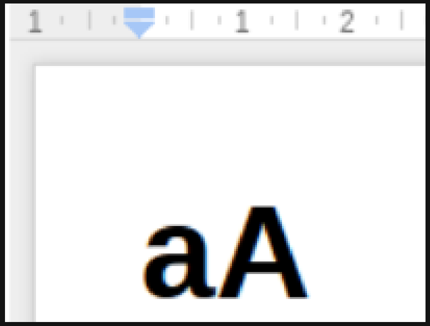
Toggle bold

Type "a" ***Press space***

Client 1



Client 2



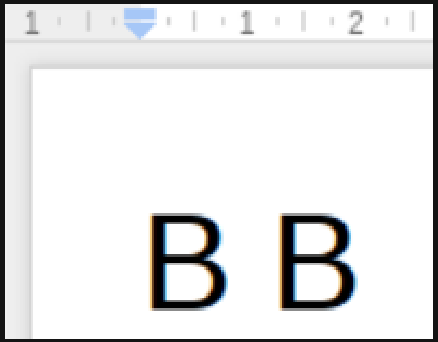
(Google Docs)

Examples

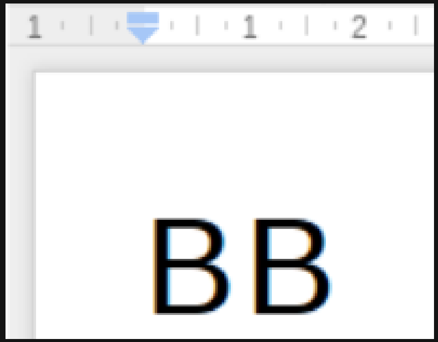
Duplicate text fragment

Both clients see incorrect state

Client 1



Client 2



Set font size to 18

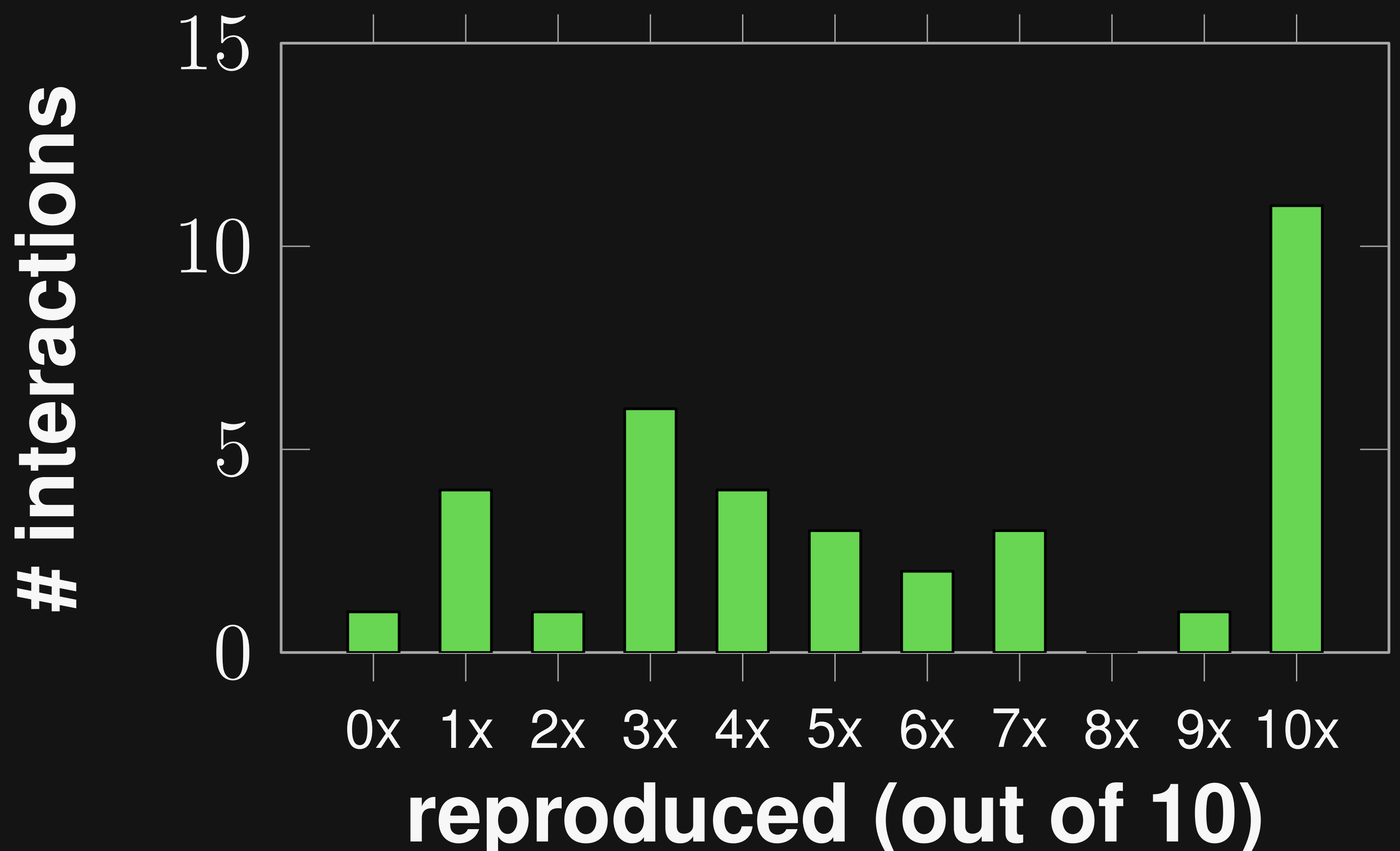
Type "b"

Press tab *Press space*

(Google Docs)

Influence of Non-Determinism

Can you reproduce these issues?



(Google Docs, 37 interactions with inconsistency in first run)

Performance

How long does it take?

- One inconsistency every 8:43 minutes
- 27–47% spent in first phase

What if we omit the first phase?

- About 10x slower

Conclusions

Analysis of collaborative web applications

- Automatic, scalable, systematic, precise
- Novel two-phase analysis of concurrent systems
- Blackbox reasoning about complex systems

Ongoing and future work

- Exploration of non-determinism
- Cluster inconsistencies by root cause

https://github.com/marinabilles/simian